EvolveTraffic

Generated by Doxygen 1.5.6

Wed Aug 20 00:48:33 2008

Contents

1 Class Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 Axle Class Reference

A class to represent an axle of a vehicle.

```
#include <Axle.h>
```

Public Member Functions

- Axle ()
- Axle (int i, double t, double v, double x, double w, int dirn)
- virtual ~Axle ()
- double getWeight ()
- double getSpacing ()
- void setWeight (double w)
- void setSpacing (double space)
- void UpdatePosition (double time)

Private Attributes

- double spacing

  The space to the next axle.

- double weight

  The weight of this axle.

4.1.1 Detailed Description

A class to represent an axle of a vehicle.

Definition at line 8 of file Axle.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Axle::Axle ()

Definition at line 14 of file Axle.cpp.

```
14           :spacing(0.0), weight(0.0)
15 {
16
17 }
```

4.1.2.2 Axle::Axle (int *i*, double *t*, double *v*, double *x*, double *w*, int *dirn*)

Definition at line 9 of file Axle.cpp.

```
10 {
11
12 }
```

4.1.2.3 Axle::~Axle () [virtual]

Definition at line 19 of file Axle.cpp.

```
19             {
20
21 }
```

4.1.3 Member Function Documentation

4.1.3.1 double Axle::getWeight ()

Definition at line 43 of file Axle.cpp.

References weight.

```
44 {
45         return weight;
46 }
```

4.1.3.2 double Axle::getSpacing ()

Definition at line 38 of file Axle.cpp.

References spacing.

```
39 {
40         return spacing;
41 }
```

4.1.3.3 void Axle::setWeight (double *w*)

Definition at line 33 of file Axle.cpp.

References weight.

```
34 {
35         weight = w;
36 }
```

4.1.3.4 void Axle::setSpacing (double *space*)

Definition at line 28 of file Axle.cpp.

References spacing.

```
29 {
30         spacing = space;
31 }
```

4.1.3.5 void Axle::UpdatePosition (double *time*)

Definition at line 23 of file Axle.cpp.

```
24 {
25
26 }
```

4.1.4 Member Data Documentation

4.1.4.1 double Axle::spacing [private]

The space to the next axle.

Definition at line 26 of file Axle.h.

Referenced by getSpacing(), and setSpacing().

4.1.4.2 double Axle::weight [private]

The weight of this axle.

Definition at line 28 of file Axle.h.

Referenced by getWeight(), and setWeight().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/Axle.h
- D:/~Research/Code/C++/EvolveTraffic/Axle.cpp

4.2 CAboutDlg Class Reference

A class for the About Dialog.

Public Types

- enum { IDD = IDD_ABOUTBOX }

Public Member Functions

- CAboutDlg ()

Public Attributes

- CString m_EditBox

Protected Member Functions

- virtual void DoDataExchange (CDataExchange *pDX)

4.2.1 Detailed Description

A class for the About Dialog.

Definition at line 122 of file EvolveTraffic.cpp.

4.2.2 Member Enumeration Documentation

4.2.2.1 anonymous enum

Enumerator:

IDD

Definition at line 129 of file EvolveTraffic.cpp.

```
129 { IDD = IDD_ABOUTBOX };
```

4.2.3 Constructor & Destructor Documentation

4.2.3.1 CAboutDlg::CAboutDlg ()

Definition at line 147 of file EvolveTraffic.cpp.

References m_EditBox.

```
147                     : CDialog(CAboutDlg::IDD)
148 {
149         //{{AFX_DATA_INIT(CAboutDlg)
150         CString str;
151         str += "Copyright (C) 2008:";                                                   str +=
152         str += "- University College Dublin";                                  str += "\r\n";
153         str += "- Laboratoire Central des Ponts et Chaus";   str += "\r\n";
154         str += "- Dublin Institute of Technology";                             str += "\r\n";
155
156         str += "Written by:";
157         str += "Dr Colin Caprani (colin.caprani@ucd.ie)";              str += "\r\n";
158         str += "Cillian Murphy";
159
160         str += "Supervised by:";
161         str += "Prof. Eugene OBrien (eugene.obrien@ucd.ie)";
162
163
164 /*      m_EditBox = _T("Copyright (C) 2008
165                                     \r\n- University College Dublin
166                                     \r\n- Laboratoire Central des Ponts et Chaus
167                                     \r\n- Dublin Institute of Technology
168                                     \r\n
169                                     \r\nWritten by:
170                                     \r\nDr Colin Caprani (colin.caprani@ucd.ie)
171                                     \r\nCillian Murphy
172                                     \r\n
173                                     \r\nSupervised by:
174                                     \r\nProf. Eugene OBrien (eugene.obrien@ucd.ie)");
175 */      m_EditBox = str;
176         //}}AFX_DATA_INIT
177 }
```

4.2.4 Member Function Documentation

4.2.4.1 void CAboutDlg::DoDataExchange (CDataExchange * *pDX*) [protected, virtual]

Definition at line 179 of file EvolveTraffic.cpp.

References IDC_EDIT1, and m_EditBox.

```
180 {
181         CDialog::DoDataExchange(pDX);
182         //{{AFX_DATA_MAP(CAboutDlg)
183         DDX_Text(pDX, IDC_EDIT1, m_EditBox);
184         //}}AFX_DATA_MAP
185 }
```

4.2.5 Member Data Documentation

4.2.5.1 CString CAboutDlg::m_EditBox

Definition at line 130 of file EvolveTraffic.cpp.

Referenced by CAboutDlg(), and DoDataExchange().

The documentation for this class was generated from the following file:

- D:/~Research/Code/C++/EvolveTraffic/EvolveTraffic.cpp

4.3 Car Class Reference

A class representing a car vehicle.

`#include <Car.h>`

Inheritance diagram for Car:

Collaboration diagram for Car:

Public Member Functions

- Car ()

  Constructor.

- virtual ~Car ()

  Default Destructor.

Static Private Attributes

- static int m_index = 0

4.3.1 Detailed Description

A class representing a car vehicle.

Definition at line 9 of file Car.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Car::Car ()

Constructor.

Definition at line 10 of file Car.cpp.

References Vehicle::m_ID, Vehicle::m_Tdelay, and VEH_ID_CAR.

```
11 {
12         m_Tdelay = 0.0;
13         m_ID = VEH_ID_CAR;
14 }
```

4.3.2.2 Car::~Car () [virtual]

Default Destructor.

Definition at line 17 of file Car.cpp.

References Vehicle::m_vAxles.

```
18 {
19         for(int i = 0; i < m_vAxles.size(); i++)
20                 delete m_vAxles.at(i);
21         m_vAxles.clear();
22 }
```

4.3.3 Member Data Documentation

4.3.3.1 int Car::m_index = 0 [static, private]

Definition at line 18 of file Car.h.

The documentation for this class was generated from the following files:

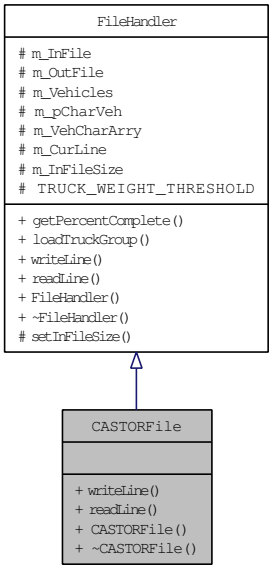
- D:/~Research/Code/C++/EvolveTraffic/Car.h
- D:/~Research/Code/C++/EvolveTraffic/Car.cpp

4.4 CASTORFile Class Reference

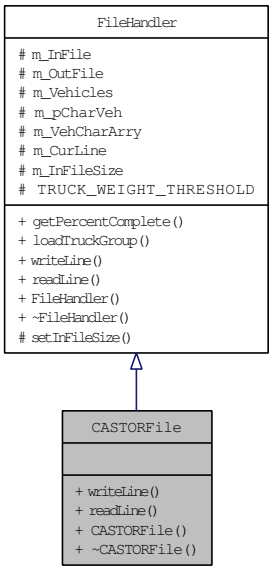
A class for reading from, and writing to, CASTOR format files.

`#include <CASTORFile.h>`

Inheritance diagram for CASTORFile:



Collaboration diagram for CASTORFile:



Public Member Functions

- void writeLine (Vehicle *pVeh)
  Writes a line representing a truck to a CASTOR file.
- Vehicle * readLine ()
  Reads a line representing a truck from a CASTOR file.
- CASTORFile (std::string inputFile, std::string outputFile)
  Constructor.
- virtual ~CASTORFile ()
  Default Destructor.

4.4.1 Detailed Description

A class for reading from, and writing to, CASTOR format files.

Definition at line 17 of file CASTORFile.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 CASTORFile::CASTORFile (std::string *inputFile*, std::string *outputFile*)

Constructor.

Parameters:

inputFile The file to read input from

outputFile The file to write output to

Definition at line 25 of file CASTORFile.cpp.

References FileHandler::m_InFile, FileHandler::m_OutFile, FileHandler::m_pCharVeh, FileHandler::m_VehCharArry, and FileHandler::setInFileSize().

```
26 {
27         m_InFile.open(inputFile.c_str(), std::ios::in);
28         setInFileSize();
29
30         m_OutFile.open(outputFile.c_str(), std::ios::out);
31         m_pCharVeh = m_VehCharArry;
32 }
```

Here is the call graph for this function:

4.4.2.2 CASTORFile::~CASTORFile () [virtual]

Default Destructor.

Definition at line 35 of file CASTORFile.cpp.

References FileHandler::m_InFile, FileHandler::m_OutFile, and FileHandler::m_-Vehicles.

```
36 {
37         m_InFile.close();
38         m_OutFile.flush();
39
40         m_Vehicles.clear();
41 }
```

4.4.3 Member Function Documentation

4.4.3.1 void CASTORFile::writeLine (Vehicle * *pVeh*) [virtual]

Writes a line representing a truck to a CASTOR file.

Parameters:

pVeh The vehicle to write to file

This function takes a vehicle as a parameter, and calls this vehicle's doCASTORDATA function, in order to obtain a string which represents all of the vehicle's physical properties. It then writes this string to a CASTOR format file

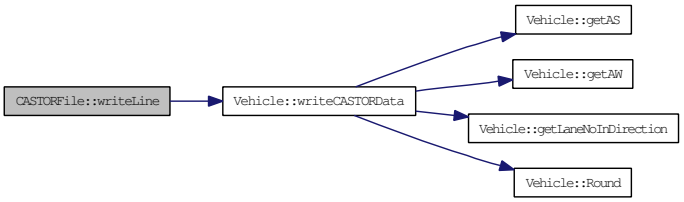
Implements FileHandler.

Definition at line 84 of file CASTORFile.cpp.

References FileHandler::m_OutFile, FileHandler::m_pCharVeh, and Vehicle::writeCASTORData().

```
85 {
86         pVeh->writeCASTORData(m_pCharVeh);
87         m_OutFile << m_pCharVeh << '\n';
88 }
```

Here is the call graph for this function:



4.4.3.2 Vehicle * CASTORFile::readLine () [virtual]

Reads a line representing a truck from a CASTOR file.

Returns:

A newly created truck

This function reads the next line available from a CASTOR format file and stores the information in a string. If the string is not null, a truck is created with the properties specified by the line from the file, and the truck is returned

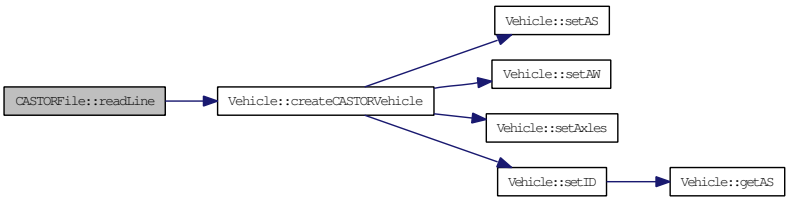
Implements FileHandler.

Definition at line 52 of file CASTORFile.cpp.

References Vehicle::createCASTORVehicle(), KG100_TO_KN, FileHandler::m_CurLine, FileHandler::m_InFile, and FileHandler::TRUCK_WEIGHT_THRESHOLD.

```
53 {
54         std::string str;
55         std::getline(m_InFile, str, '\n');
56 
57         if(str != "")
58         {
59                 m_CurLine++;
60                 Vehicle* pVeh;
61                 int GVW = atoi( str.substr(21,4).c_str() ); // TRUCK_WEIGHT_THRESHOLD in kg/100
62 
63                 if(GVW >= TRUCK_WEIGHT_THRESHOLD * KG100_TO_KN)
64                         pVeh = new Truck();
65                 else
66                         pVeh = new Car();
67 
68                 pVeh->createCASTORVehicle(str);
69                 return pVeh;
70         }
71 
72         return NULL;
73 }
```

Here is the call graph for this function:



The documentation for this class was generated from the following files:

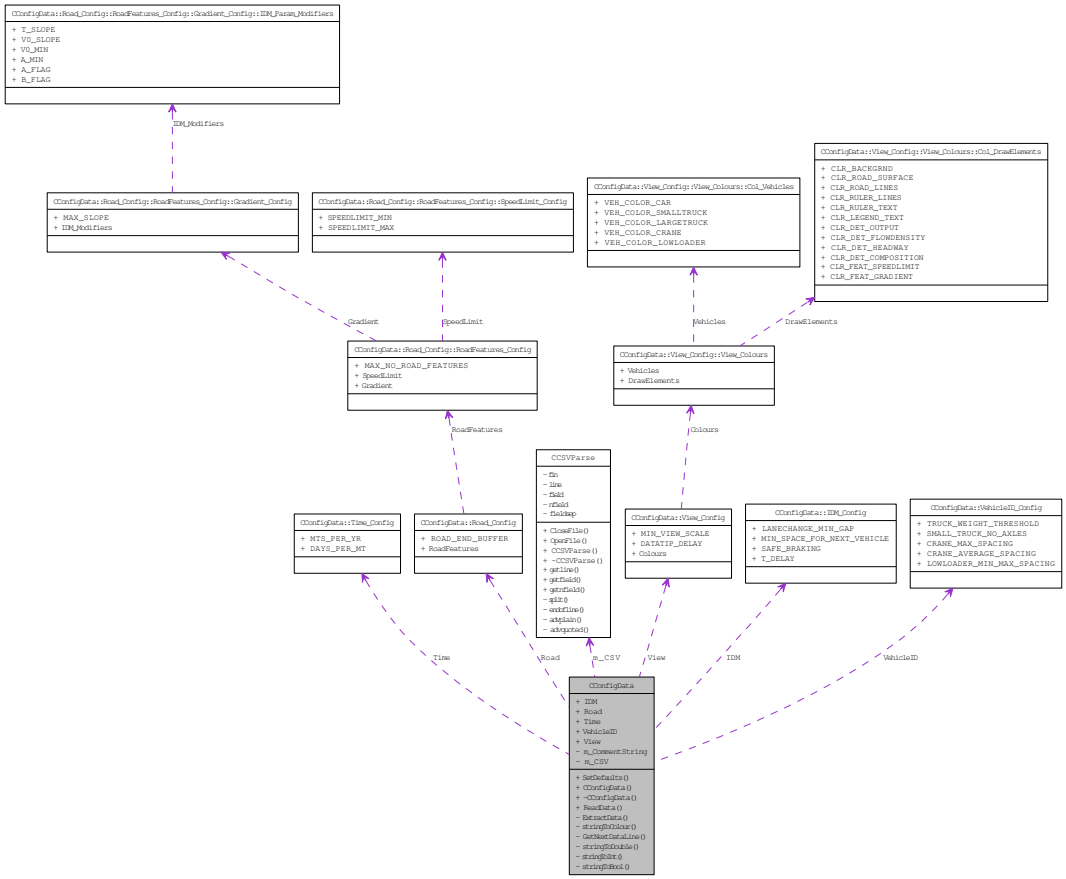
- D:/~Research/Code/C++/EvolveTraffic/CASTORFile.h
- D:/~Research/Code/C++/EvolveTraffic/CASTORFile.cpp

4.5 CConfigData Class Reference

A class for containing all of the configuration prameters for the program.

`#include <ConfigData.h>`

Collaboration diagram for CConfigData:



Public Member Functions

- void SetDefaults ()
- CConfigData ()
- virtual ~CConfigData ()
- bool ReadData (std::string inFile)

Public Attributes

- struct CConfigData::IDM_Config IDM
- struct CConfigData::Road_Config Road
- struct CConfigData::Time_Config Time

- struct CConfigData::VehicleID_Config VehicleID
- struct CConfigData::View_Config View

Private Member Functions

- void ExtractData ()
- COLORREF stringToColour (string line)
- string GetNextDataLine ()
- double stringToDouble (string line)
- int stringToInt (string line)
- bool stringToBool (string line)

Private Attributes

- string m_CommentString
- CCSVParse m_CSV

Classes

- struct IDM_Config
- struct Road_Config
- struct Time_Config
- struct VehicleID_Config
- struct View_Config

4.5.1 Detailed Description

A class for containing all of the configuration prameters for the program.

Definition at line 17 of file ConfigData.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 CConfigData::CConfigData ()

Definition at line 19 of file ConfigData.cpp.

References m_CommentString, and SetDefaults().

```
20 {
21         m_CommentString = "//";
22         SetDefaults();
23 }
```

Here is the call graph for this function:

4.5.2.2 CConfigData::~CConfigData () [virtual]

Definition at line 25 of file ConfigData.cpp.

```
26 {
27
28 }
```

4.5.3 Member Function Documentation

4.5.3.1 void CConfigData::SetDefaults ()

Definition at line 108 of file ConfigData.cpp.

References CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_-BACKGRND, CConfigData::View_Config::View_Colours::Col_-DrawElements::CLR_DET_COMPOSITION, CConfigData::View_-Config::View_Colours::Col_DrawElements::CLR_DET_FLOWDENSITY, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_-DET_HEADWAY, CConfigData::View_Config::View_Colours::Col_-DrawElements::CLR_DET_OUTPUT, CConfigData::View_Config::View_-Colours::Col_DrawElements::CLR_FEAT_GRADIENT, CConfigData::View_-Config::View_Colours::Col_DrawElements::CLR_FEAT_SPEEDLIMIT, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_-LEGEND_TEXT, CConfigData::View_Config::View_Colours::Col_-DrawElements::CLR_ROAD_LINES, CConfigData::View_Config::View_-Colours::Col_DrawElements::CLR_ROAD_SURFACE, CConfigData::View_-Config::View_Colours::Col_DrawElements::CLR_RULER_LINES, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_RULER_-TEXT, CConfigData::View_Config::Colours, CConfigData::VehicleID_-Config::CRANE_AVERAGE_SPACING, CConfigData::VehicleID_-Config::CRANE_MAX_SPACING, CConfigData::View_Config::DATATIP_DELAY, CConfigData::Time_Config::DAYS_PER_MT, CConfigData::View_Config::View_-Colours::DrawElements, CConfigData::VehicleID_Config::LOWLOADER_-MIN_MAX_SPACING, CConfigData::View_Config::MIN_VIEW_SCALE, CConfigData::Time_Config::MTS_PER_YR, CConfigData::VehicleID_-Config::SMALL_TRUCK_NO_AXLES, Time, CConfigData::VehicleID_-Config::TRUCK_WEIGHT_THRESHOLD, CConfigData::View_Config::View_-Colours::Col_Vehicles::VEH_COLOR_CAR, CConfigData::View_Config::View_-Colours::Col_Vehicles::VEH_COLOR_CRANE, CConfigData::View_Config::View_-Colours::Col_Vehicles::VEH_COLOR_LARGETRUCK, CConfigData::View_-Config::View_Colours::Col_Vehicles::VEH_COLOR_LOWLOADER, CConfigData::View_Config::View_Colours::Col_Vehicles::VEH_COLOR_-SMALLTRUCK, VehicleID, CConfigData::View_Config::View_Colours::Vehicles, and View.

Referenced by CConfigData().

```
109 {
110         IDM.LANECHANGE_MIN_GAP                  = 2.0;
111         IDM.MIN_SPACE_FOR_NEXT_VEHICLE  = 27.0;
112         IDM.SAFE_BRAKING                                = -12.0;
```

```
113         IDM.T_DELAY                                             = 1.6;
114
115
116         Road.ROAD_END_BUFFER                                                                    = 1000;
117
118         Road.RoadFeatures.MAX_NO_ROAD_FEATURES                          = 10;
119
120         Road.RoadFeatures.SpeedLimit.SPEEDLIMIT_MIN                     = 10;
121         Road.RoadFeatures.SpeedLimit.SPEEDLIMIT_MAX                     = 200;
122
123         Road.RoadFeatures.Gradient.MAX_SLOPE                            = 15;
124         Road.RoadFeatures.Gradient.IDM_Modifiers.A_FLAG         = true;
125         Road.RoadFeatures.Gradient.IDM_Modifiers.A_MIN          = 0.1;
126         Road.RoadFeatures.Gradient.IDM_Modifiers.B_FLAG         = true;
127         Road.RoadFeatures.Gradient.IDM_Modifiers.T_SLOPE        = 10.0;
128         Road.RoadFeatures.Gradient.IDM_Modifiers.V0_MIN         = 0.1;
129         Road.RoadFeatures.Gradient.IDM_Modifiers.V0_SLOPE       = 0.2;
130
131
132         Time.DAYS_PER_MT        = 25;
133         Time.MTS_PER_YR         = 10;
134
135
136         VehicleID.CRANE_AVERAGE_SPACING         = 2.5;
137         VehicleID.CRANE_MAX_SPACING                     = 4.5;
138         VehicleID.LOWLOADER_MIN_MAX_SPACING     = 7.5;
139         VehicleID.SMALL_TRUCK_NO_AXLES          = 3;
140         VehicleID.TRUCK_WEIGHT_THRESHOLD        = 35;
141
142
143         View.MIN_VIEW_SCALE                                     = 0.1;
144         View.DATATIP_DELAY                                      = 20.0;
145
146         View.Colours.Vehicles.VEH_COLOR_CAR                     = RGB(0,0,0);
147         View.Colours.Vehicles.VEH_COLOR_SMALLTRUCK      = RGB(0,0,255);
148         View.Colours.Vehicles.VEH_COLOR_LARGETRUCK      = RGB(255,0,0);
149         View.Colours.Vehicles.VEH_COLOR_CRANE           = RGB(0,255,0);
150         View.Colours.Vehicles.VEH_COLOR_LOWLOADER       = RGB(100,100,100);
151
152         View.Colours.DrawElements.CLR_BACKGRND                  = RGB(255,255,255);
153         View.Colours.DrawElements.CLR_ROAD_SURFACE              = RGB(200,200,200);
154         View.Colours.DrawElements.CLR_ROAD_LINES                = RGB(0,0,0);
155         View.Colours.DrawElements.CLR_RULER_LINES               = RGB(0,0,0);
156         View.Colours.DrawElements.CLR_RULER_TEXT                = RGB(0,0,0);
157         View.Colours.DrawElements.CLR_LEGEND_TEXT               = RGB(0,0,0);
158         View.Colours.DrawElements.CLR_DET_OUTPUT                = RGB(255,255,0);
159         View.Colours.DrawElements.CLR_DET_FLOWDENSITY   = RGB(0,255,0);
160         View.Colours.DrawElements.CLR_DET_HEADWAY               = RGB(0,0,255);
161         View.Colours.DrawElements.CLR_DET_COMPOSITION   = RGB(0,0,0);
162         View.Colours.DrawElements.CLR_FEAT_SPEEDLIMIT   = RGB(255,0,0);
163         View.Colours.DrawElements.CLR_FEAT_GRADIENT             = RGB(0,255,0);
164 }
```

4.5.3.2 bool CConfigData::ReadData (std::string *inFile*)

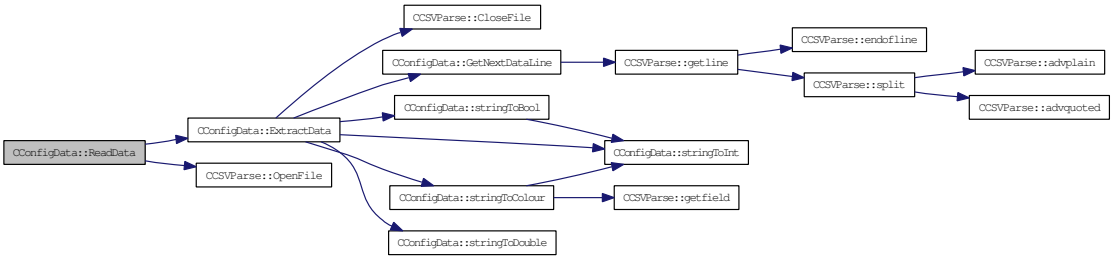
Definition at line 34 of file ConfigData.cpp.

References ExtractData(), m_CSV, and CCSVParse::OpenFile().

Referenced by CEvolveTrafficApp::InitInstance().

```
35 {
36         if( m_CSV.OpenFile(inFile, ",") )
37         {
38                 ExtractData();
39                 return true;
40         }
41         return false;
42 }
```

Here is the call graph for this function:



4.5.3.3 void CConfigData::ExtractData () [private]

Definition at line 44 of file ConfigData.cpp.

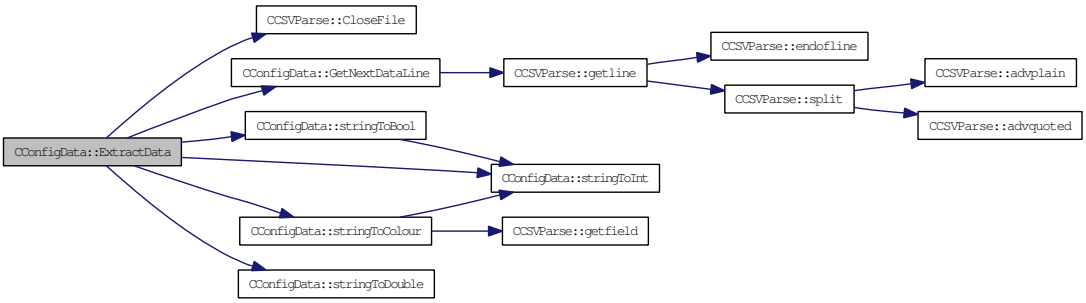
References CCSVParse::CloseFile(), CConfigData::View_Config::View_-Colours::Col_DrawElements::CLR_BACKGRND, CConfigData::View_-Config::View_Colours::Col_DrawElements::CLR_DET_COMPOSITION, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_-DET_FLOWDENSITY, CConfigData::View_Config::View_Colours::Col_-DrawElements::CLR_DET_HEADWAY, CConfigData::View_Config::View_-Colours::Col_DrawElements::CLR_DET_OUTPUT, CConfigData::View_-Config::View_Colours::Col_DrawElements::CLR_FEAT_GRADIENT, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_-FEAT_SPEEDLIMIT, CConfigData::View_Config::View_Colours::Col_-DrawElements::CLR_LEGEND_TEXT, CConfigData::View_Config::View_-Colours::Col_DrawElements::CLR_ROAD_LINES, CConfigData::View_-Config::View_Colours::Col_DrawElements::CLR_ROAD_SURFACE, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_RULER_-LINES, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_-RULER_TEXT, CConfigData::View_Config::Colours, CConfigData::VehicleID_-Config::CRANE_AVERAGE_SPACING, CConfigData::VehicleID_-Config::CRANE_MAX_SPACING, CConfigData::View_Config::DATATIP_DELAY, CConfigData::Time_Config::DAYS_PER_MT, CConfigData::View_Config::View_-Colours::DrawElements, GetNextDataLine(), CConfigData::VehicleID_-Config::LOWLOADER_MIN_MAX_SPACING, m_CSV, CConfigData::View_-Config::MIN_VIEW_SCALE, CConfigData::Time_Config::MTS_PER_YR, CConfigData::VehicleID_Config::SMALL_TRUCK_NO_AXLES, stringToBool(), stringToColour(), stringToDouble(), stringToInt(), Time, CConfigData::VehicleID_-Config::TRUCK_WEIGHT_THRESHOLD, CConfigData::View_Config::View_-

Colours::Col_Vehicles::VEH_COLOR_CAR, CConfigData::View_Config::View_-Colours::Col_Vehicles::VEH_COLOR_CRANE, CConfigData::View_Config::View_-Colours::Col_Vehicles::VEH_COLOR_LARGETRUCK, CConfigData::View_-Config::View_Colours::Col_Vehicles::VEH_COLOR_LOWLOADER, CConfigData::View_Config::View_Colours::Col_Vehicles::VEH_COLOR_-SMALLTRUCK, VehicleID, CConfigData::View_Config::View_Colours::Vehicles, and View.

Referenced by ReadData().

```
45 {
46                 string str;
47
48                 str = GetNextDataLine();        IDM.LANECHANGE_MIN_GAP                  = string
49                 str = GetNextDataLine();        IDM.MIN_SPACE_FOR_NEXT_VEHICLE  = stringToDouble
50                 str = GetNextDataLine();        IDM.SAFE_BRAKING
51                 str = GetNextDataLine();        IDM.T_DELAY
52
53                 str = GetNextDataLine();        Road.ROAD_END_BUFFER
54                 str = GetNextDataLine();        Road.RoadFeatures.SpeedLimit.SPEEDLIMIT_MIN
55                 str = GetNextDataLine();        Road.RoadFeatures.SpeedLimit.SPEEDLIMIT_MAX
56                 str = GetNextDataLine();        Road.RoadFeatures.Gradient.MAX_SLOPE
57                 str = GetNextDataLine();        Road.RoadFeatures.Gradient.IDM_Modifiers.A_FLAG
58                 str = GetNextDataLine();        Road.RoadFeatures.Gradient.IDM_Modifiers.A_MIN
59                 str = GetNextDataLine();        Road.RoadFeatures.Gradient.IDM_Modifiers.B_FLAG
60                 str = GetNextDataLine();        Road.RoadFeatures.Gradient.IDM_Modifiers.T_SLOPE
61                 str = GetNextDataLine();        Road.RoadFeatures.Gradient.IDM_Modifiers.V0_MIN
62                 str = GetNextDataLine();        Road.RoadFeatures.Gradient.IDM_Modifiers.V0_SLOP
63
64                 str = GetNextDataLine();        Time.DAYS_PER_MT        = stringToInt(str);
65                 str = GetNextDataLine();        Time.MTS_PER_YR         = stringToInt(str);
66
67                 str = GetNextDataLine();        VehicleID.CRANE_AVERAGE_SPACING         = string
68                 str = GetNextDataLine();        VehicleID.CRANE_MAX_SPACING
69                 str = GetNextDataLine();        VehicleID.LOWLOADER_MIN_MAX_SPACING     = string
70                 str = GetNextDataLine();        VehicleID.SMALL_TRUCK_NO_AXLES          = string
71                 str = GetNextDataLine();        VehicleID.TRUCK_WEIGHT_THRESHOLD        = string
72
73                 str = GetNextDataLine();        View.MIN_VIEW_SCALE
74                 str = GetNextDataLine();        View.DATATIP_DELAY
75                 str = GetNextDataLine();        View.Colours.Vehicles.VEH_COLOR_CAR
76                 str = GetNextDataLine();        View.Colours.Vehicles.VEH_COLOR_SMALLTRUCK
77                 str = GetNextDataLine();        View.Colours.Vehicles.VEH_COLOR_LARGETRUCK
78                 str = GetNextDataLine();        View.Colours.Vehicles.VEH_COLOR_CRANE
79                 str = GetNextDataLine();        View.Colours.Vehicles.VEH_COLOR_LOWLOADER
80                 str = GetNextDataLine();        View.Colours.DrawElements.CLR_BACKGRND
81                 str = GetNextDataLine();        View.Colours.DrawElements.CLR_ROAD_SURFACE
82                 str = GetNextDataLine();        View.Colours.DrawElements.CLR_ROAD_LINES
83                 str = GetNextDataLine();        View.Colours.DrawElements.CLR_RULER_LINES
84                 str = GetNextDataLine();        View.Colours.DrawElements.CLR_RULER_TEXT
85                 str = GetNextDataLine();        View.Colours.DrawElements.CLR_LEGEND_TEXT
86                 str = GetNextDataLine();        View.Colours.DrawElements.CLR_DET_OUTPUT
87                 str = GetNextDataLine();        View.Colours.DrawElements.CLR_DET_FLOWDENSITY
88                 str = GetNextDataLine();        View.Colours.DrawElements.CLR_DET_HEADWAY
89                 str = GetNextDataLine();        View.Colours.DrawElements.CLR_DET_COMPOSITION
90                 str = GetNextDataLine();        View.Colours.DrawElements.CLR_FEAT_SPEEDLIMIT
91                 str = GetNextDataLine();        View.Colours.DrawElements.CLR_FEAT_GRADIENT
92
93                 m_CSV.CloseFile();
94 }
```

Here is the call graph for this function:



4.5.3.4 COLORREF CConfigData::stringToColour (string *line*) [private]

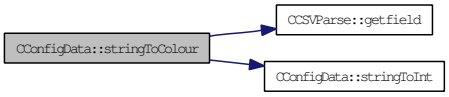
Definition at line 192 of file ConfigData.cpp.

References CCSVParse::getfield(), m_CSV, and stringToInt().

Referenced by ExtractData().

```
193 {
194         int red, green, blue;
195         string str;
196
197         str = m_CSV.getfield(0);        red = stringToInt(str);
198         str = m_CSV.getfield(1);        green = stringToInt(str);
199         str = m_CSV.getfield(2);        blue = stringToInt(str);
200
201         return RGB(red,green,blue);
202 }
```

Here is the call graph for this function:



4.5.3.5 string CConfigData::GetNextDataLine () [private]

Definition at line 96 of file ConfigData.cpp.

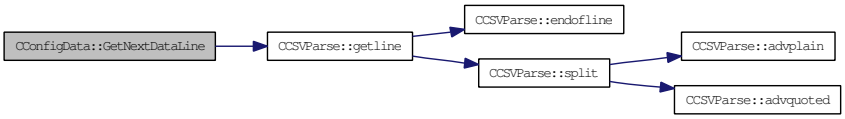
References CCSVParse::getline(), m_CommentString, and m_CSV.

Referenced by ExtractData().

```
97 {
98         string line;
99         m_CSV.getline(line);
```

```
100         while(line.substr(0,2) == m_CommentString)
101                 m_CSV.getline(line);
102
103         return line;
104 }
```

Here is the call graph for this function:



4.5.3.6 double CConfigData::stringToDouble (string *line*) [private]

Definition at line 166 of file ConfigData.cpp.

Referenced by ExtractData().

```
167 {
168         double val;     char data[15];
169         line.copy(data, 14, 0);
170         val = atof(data);
171         return val;
172 }
```

4.5.3.7 int CConfigData::stringToInt (string *line*) [private]

Definition at line 174 of file ConfigData.cpp.

Referenced by ExtractData(), stringToBool(), and stringToColour().

```
175 {
176         int val = 0;    char data[15];
177         line.copy(data, 14, 0);
178         val = atoi(data);
179         return val;
180 }
```

4.5.3.8 bool CConfigData::stringToBool (string *line*) [private]

Definition at line 182 of file ConfigData.cpp.

References stringToInt().

Referenced by ExtractData().

```
183 {
184         int val = stringToInt(line);
185         if(val == 1)
186                 return true;
187         else
188                 return false;
189 }
```

Here is the call graph for this function:

4.5.4 Member Data Documentation

4.5.4.1 struct CConfigData::IDM_Config CConfigData::IDM

Referenced by IDM::IDM(), Lane::Lane(), Road::Road(), and Vehicle::Vehicle().

4.5.4.2 struct CConfigData::Road_Config CConfigData::Road

Referenced by CRoadFeaturesDlg::CRoadFeaturesDlg(), IDM::IDM(), and Vehicle::Vehicle().

4.5.4.3 struct CConfigData::Time_Config CConfigData::Time

Referenced by ExtractData(), SetDefaults(), and Vehicle::Vehicle().

4.5.4.4 struct CConfigData::VehicleID_Config CConfigData::VehicleID

Referenced by ExtractData(), FileHandler::FileHandler(), SetDefaults(), and Vehicle::Vehicle().

4.5.4.5 struct CConfigData::View_Config CConfigData::View

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CPreferencesDlg::CPreferencesDlg(), ExtractData(), and SetDefaults().

4.5.4.6 string CConfigData::m_CommentString [private]

Definition at line 125 of file ConfigData.h.

Referenced by CConfigData(), and GetNextDataLine().

4.5.4.7 CCSVParse CConfigData::m_CSV [private]

Definition at line 127 of file ConfigData.h.

Referenced by ExtractData(), GetNextDataLine(), ReadData(), and stringToColour().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/ConfigData.h
- D:/~Research/Code/C++/EvolveTraffic/ConfigData.cpp

4.6 CConfigData::IDM_Config Struct Reference

`#include <ConfigData.h>`

Public Attributes

- double LANECHANGE_MIN_GAP
- double MIN_SPACE_FOR_NEXT_VEHICLE
- double SAFE_BRAKING
- double T_DELAY

4.6.1 Detailed Description

Definition at line 27 of file ConfigData.h.

4.6.2 Member Data Documentation

4.6.2.1 double CConfigData::IDM_Config::LANECHANGE_MIN_GAP

Definition at line 29 of file ConfigData.h.

Referenced by IDM::IDM().

4.6.2.2 double CConfigData::IDM_Config::MIN_SPACE_FOR_NEXT_VEHICLE

Definition at line 30 of file ConfigData.h.

Referenced by Lane::Lane(), Road::Road(), and Vehicle::Vehicle().

4.6.2.3 double CConfigData::IDM_Config::SAFE_BRAKING

Definition at line 31 of file ConfigData.h.

Referenced by IDM::IDM(), and Vehicle::Vehicle().

4.6.2.4 double CConfigData::IDM_Config::T_DELAY

Definition at line 32 of file ConfigData.h.

Referenced by Vehicle::Vehicle().

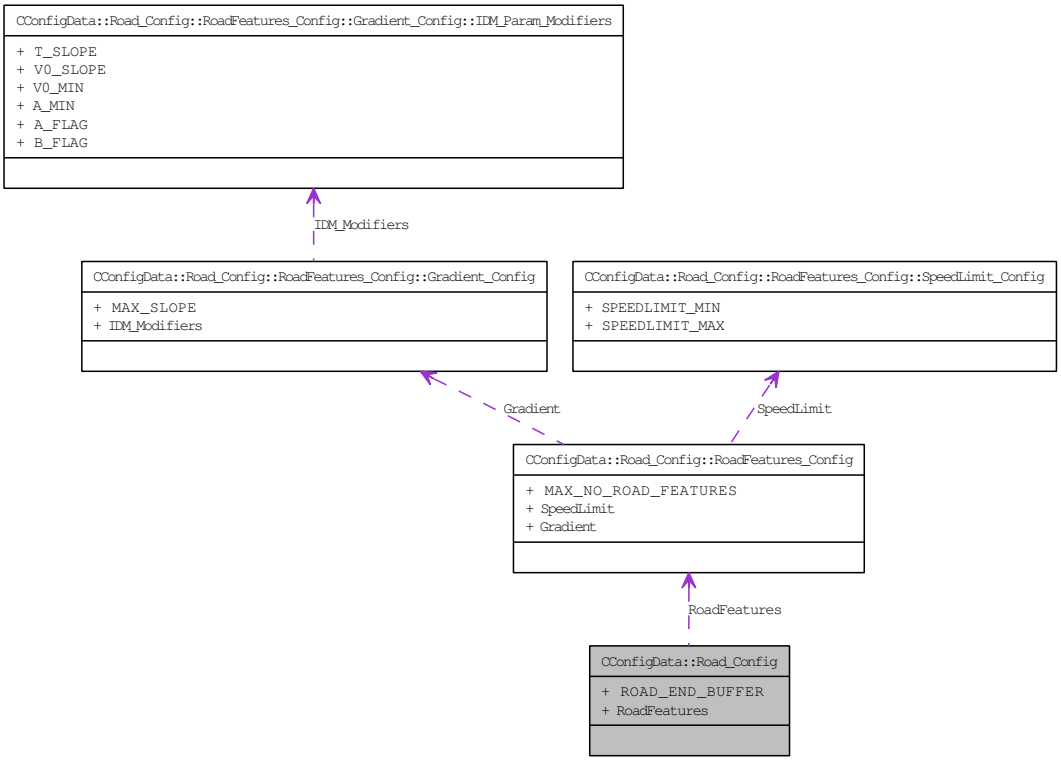
The documentation for this struct was generated from the following file:

- D:/~Research/Code/C++/EvolveTraffic/ConfigData.h

4.7 CConfigData::Road_Config Struct Reference

```
#include <ConfigData.h>
```

Collaboration diagram for CConfigData::Road_Config:



Public Attributes

- int ROAD_END_BUFFER
- struct CConfigData::Road_Config::RoadFeatures_Config RoadFeatures

Classes

- struct RoadFeatures_Config

4.7.1 Detailed Description

Definition at line 36 of file ConfigData.h.

4.7.2 Member Data Documentation

4.7.2.1 int CConfigData::Road_Config::ROAD_END_BUFFER

Definition at line 38 of file ConfigData.h.

Referenced by Vehicle::Vehicle().

4.7.2.2 struct CConfigData::Road_Config::RoadFeatures_Config CConfigData::Road_Config::RoadFeatures

Referenced by CRoadFeaturesDlg::CRoadFeaturesDlg(), and IDM::IDM().

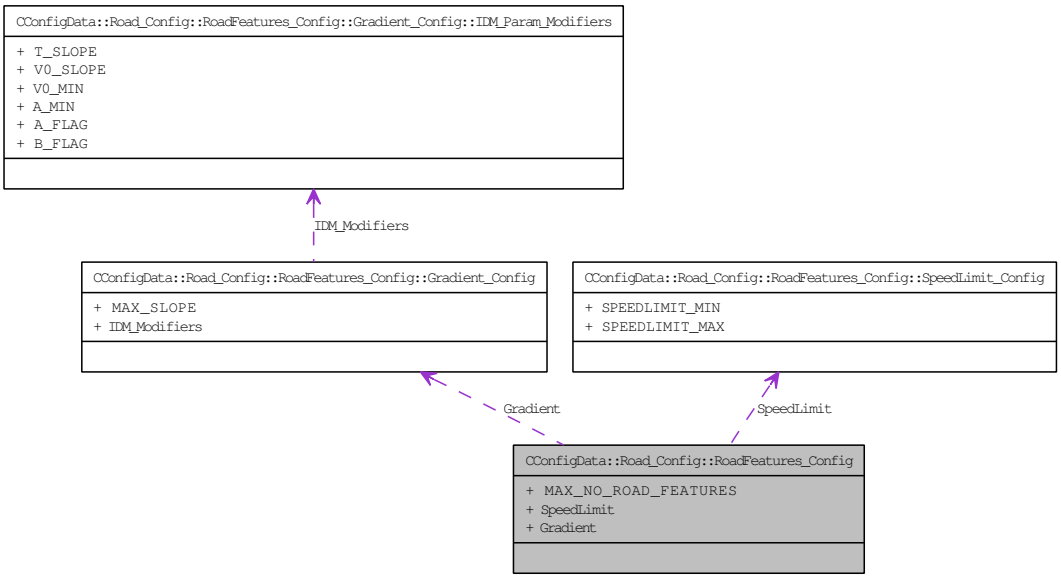
The documentation for this struct was generated from the following file:

- D:/~Research/Code/C++/EvolveTraffic/ConfigData.h

4.8 CConfigData::Road_Config::RoadFeatures_Config Struct Reference

`#include <ConfigData.h>`

Collaboration diagram for CConfigData::Road_Config::RoadFeatures_Config:



Public Attributes

- int MAX_NO_ROAD_FEATURES
- struct CConfigData::Road_Config::RoadFeatures_Config::SpeedLimit_Config SpeedLimit
- struct CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config Gradient

Classes

- struct Gradient_Config
- struct SpeedLimit_Config

4.8.1 Detailed Description

Definition at line 40 of file ConfigData.h.

4.8.2 Member Data Documentation

4.8.2.1 int CConfigData::Road_Config::RoadFeatures_Config::MAX_NO_ROAD_FEATURES

Definition at line 42 of file ConfigData.h.

4.8.2.2 struct CConfigData::Road_Config::RoadFeatures_Config::SpeedLimit_Config CConfigData::Road_Config::RoadFeatures_Config::SpeedLimit

Referenced by CRoadFeaturesDlg::CRoadFeaturesDlg().

4.8.2.3 struct CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config CConfigData::Road_Config::RoadFeatures_Config::Gradient

Referenced by CRoadFeaturesDlg::CRoadFeaturesDlg(), and IDM::IDM().

The documentation for this struct was generated from the following file:

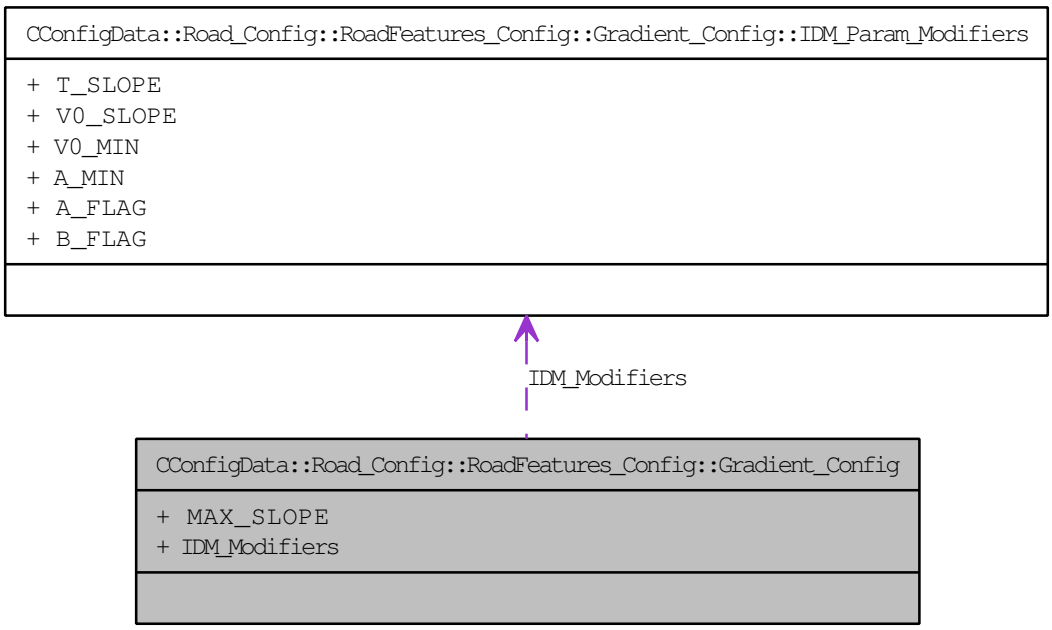
- D:/~Research/Code/C++/EvolveTraffic/ConfigData.h

4.9 CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config Struct Reference

`#include <ConfigData.h>`

Collaboration diagram for CConfigData::Road_Config::RoadFeatures_-

Config::Gradient_Config:



Public Attributes

- int MAX_SLOPE
- struct CConfigData::Road_Config::RoadFeatures_Config::Gradient_-Config::IDM_Param_Modifiers IDM_Modifiers

Classes

- struct IDM_Param_Modifiers

4.9.1 Detailed Description

Definition at line 52 of file ConfigData.h.

4.9.2 Member Data Documentation

4.9.2.1 int CConfigData::Road_Config::RoadFeatures_Config::Gradient_-Config::MAX_SLOPE

Definition at line 54 of file ConfigData.h.

Referenced by CRoadFeaturesDlg::CRoadFeaturesDlg().

4.9.2.2 struct CConfigData::Road_Config::RoadFeatures_Config::Gradient_-Config::IDM_Param_Modifiers CConfigData::Road_Config::RoadFeatures_-Config::Gradient_Config::IDM_Modifiers

Referenced by IDM::IDM().

The documentation for this struct was generated from the following file:

- D:/~Research/Code/C++/EvolveTraffic/ConfigData.h

4.10 CConfigData::Road_Config::RoadFeatures_-Config::Gradient_Config::IDM_Param_Modifiers Struct Reference

`#include <ConfigData.h>`

Public Attributes

- double T_SLOPE
- double V0_SLOPE
- double V0_MIN
- double A_MIN
- bool A_FLAG
- bool B_FLAG

4.10.1 Detailed Description

Definition at line 55 of file ConfigData.h.

4.10.2 Member Data Documentation

4.10.2.1 double CConfigData::Road_Config::RoadFeatures_-Config::Gradient_Config::IDM_Param_Modifiers::T_SLOPE

Definition at line 57 of file ConfigData.h.

Referenced by IDM::IDM().

4.10.2.2 double CConfigData::Road_Config::RoadFeatures_-Config::Gradient_Config::IDM_Param_Modifiers::V0_SLOPE

Definition at line 58 of file ConfigData.h.

Referenced by IDM::IDM().

4.10.2.3 double CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config::IDM_Param_Modifiers::V0_MIN

Definition at line 59 of file ConfigData.h.

Referenced by IDM::IDM().

4.10.2.4 double CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config::IDM_Param_Modifiers::A_MIN

Definition at line 60 of file ConfigData.h.

Referenced by IDM::IDM().

4.10.2.5 bool CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config::IDM_Param_Modifiers::A_FLAG

Definition at line 61 of file ConfigData.h.

Referenced by IDM::IDM().

4.10.2.6 bool CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config::IDM_Param_Modifiers::B_FLAG

Definition at line 62 of file ConfigData.h.

Referenced by IDM::IDM().

The documentation for this struct was generated from the following file:

- D:/~Research/Code/C++/EvolveTraffic/ConfigData.h

4.11 CConfigData::Road_Config::RoadFeatures_Config::SpeedLimit_Config Struct Reference

`#include <ConfigData.h>`

Public Attributes

- int SPEEDLIMIT_MIN
- int SPEEDLIMIT_MAX

4.11.1 Detailed Description

Definition at line 45 of file ConfigData.h.

4.11.2 Member Data Documentation

4.11.2.1 int CConfigData::Road_Config::RoadFeatures_Config::SpeedLimit_Config::SPEEDLIMIT_MIN

Definition at line 47 of file ConfigData.h.

Referenced by CRoadFeaturesDlg::CRoadFeaturesDlg().

4.11.2.2 int CConfigData::Road_Config::RoadFeatures_Config::SpeedLimit_Config::SPEEDLIMIT_MAX

Definition at line 48 of file ConfigData.h.

Referenced by CRoadFeaturesDlg::CRoadFeaturesDlg().

The documentation for this struct was generated from the following file:

- D:/~Research/Code/C++/EvolveTraffic/ConfigData.h

4.12 CConfigData::Time_Config Struct Reference

`#include <ConfigData.h>`

Public Attributes

- int MTS_PER_YR
- int DAYS_PER_MT

4.12.1 Detailed Description

Definition at line 69 of file ConfigData.h.

4.12.2 Member Data Documentation

4.12.2.1 int CConfigData::Time_Config::MTS_PER_YR

Definition at line 71 of file ConfigData.h.

Referenced by CConfigData::ExtractData(), CConfigData::SetDefaults(), and Vehicle::Vehicle().

4.12.2.2 int CConfigData::Time_Config::DAYS_PER_MT

Definition at line 72 of file ConfigData.h.

Referenced by CConfigData::ExtractData(), CConfigData::SetDefaults(), and Vehicle::Vehicle().

The documentation for this struct was generated from the following file:

- D:/~Research/Code/C++/EvolveTraffic/ConfigData.h

4.13 CConfigData::VehicleID_Config Struct Reference

`#include <ConfigData.h>`

Public Attributes

- int TRUCK_WEIGHT_THRESHOLD
- int SMALL_TRUCK_NO_AXLES
- int CRANE_MAX_SPACING
- double CRANE_AVERAGE_SPACING
- double LOWLOADER_MIN_MAX_SPACING

4.13.1 Detailed Description

Definition at line 76 of file ConfigData.h.

4.13.2 Member Data Documentation

4.13.2.1 int CConfigData::VehicleID_Config::TRUCK_WEIGHT_THRESHOLD

Definition at line 78 of file ConfigData.h.

Referenced by CConfigData::ExtractData(), FileHandler::FileHandler(), and CConfigData::SetDefaults().

4.13.2.2 int CConfigData::VehicleID_Config::SMALL_TRUCK_NO_AXLES

Definition at line 79 of file ConfigData.h.

Referenced by CConfigData::ExtractData(), CConfigData::SetDefaults(), and Vehicle::Vehicle().

4.13.2.3 int CConfigData::VehicleID_Config::CRANE_MAX_SPACING

Definition at line 80 of file ConfigData.h.

Referenced by CConfigData::ExtractData(), CConfigData::SetDefaults(), and Vehicle::Vehicle().

4.13.2.4 double CConfigData::VehicleID_Config::CRANE_AVERAGE_SPACING

Definition at line 81 of file ConfigData.h.

Referenced by CConfigData::ExtractData(), CConfigData::SetDefaults(), and Vehicle::Vehicle().

4.13.2.5 double CConfigData::VehicleID_Config::LOWLOADER_MIN_MAX_SPACING

Definition at line 82 of file ConfigData.h.

Referenced by CConfigData::ExtractData(), CConfigData::SetDefaults(), and Vehicle::Vehicle().

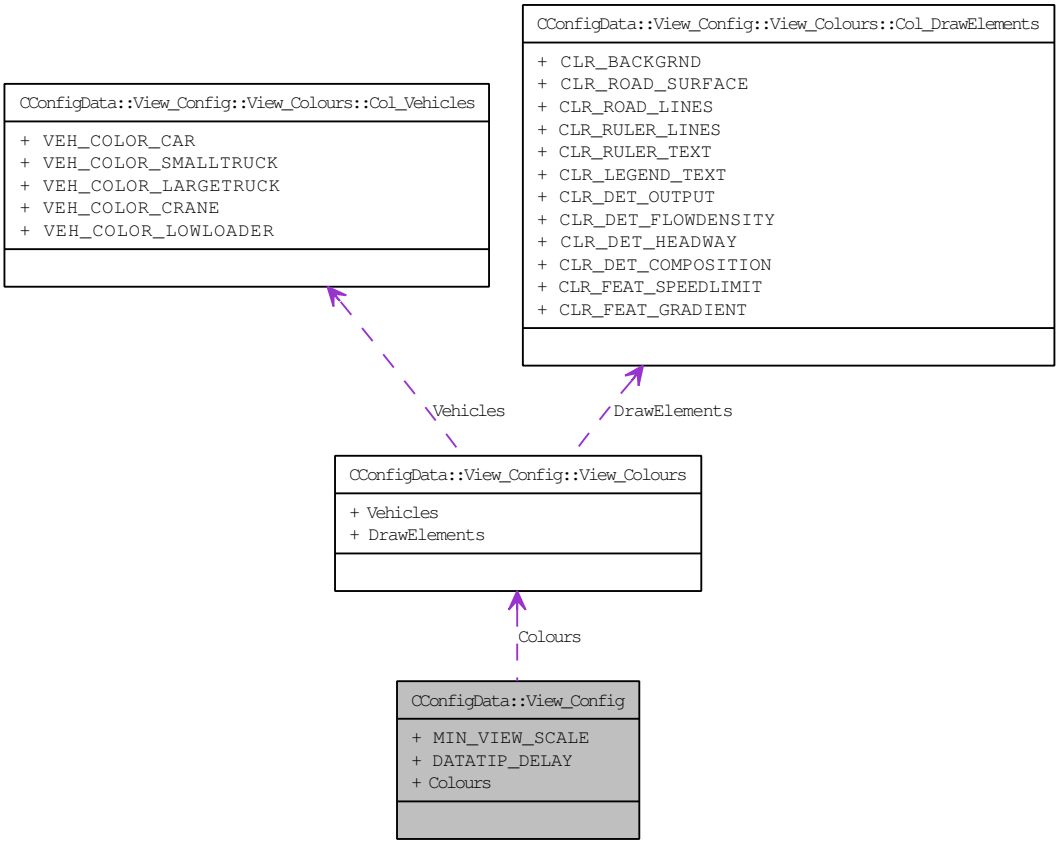
The documentation for this struct was generated from the following file:

- D:/~Research/Code/C++/EvolveTraffic/ConfigData.h

4.14 CConfigData::View_Config Struct Reference

`#include <ConfigData.h>`

Collaboration diagram for CConfigData::View_Config:



Public Attributes

- double MIN_VIEW_SCALE

- double DATATIP_DELAY
- struct CConfigData::View_Config::View_Colours Colours

Classes

- struct View_Colours

4.14.1 Detailed Description

Definition at line 86 of file ConfigData.h.

4.14.2 Member Data Documentation

4.14.2.1 double CConfigData::View_Config::MIN_VIEW_SCALE

Definition at line 89 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CPreferencesDlg::CPreferencesDlg(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.14.2.2 double CConfigData::View_Config::DATATIP_DELAY

Definition at line 90 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.14.2.3 struct CConfigData::View_Config::View_Colours CConfigData::View_Config::Colours

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

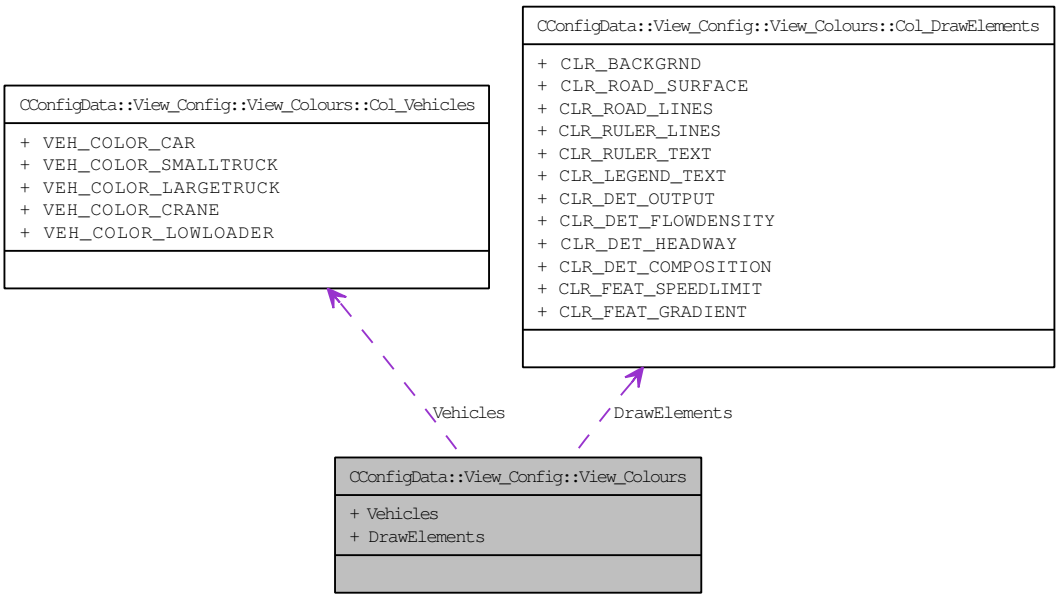
The documentation for this struct was generated from the following file:

- D:/~Research/Code/C++/EvolveTraffic/ConfigData.h

4.15 CConfigData::View_Config::View_Colours Struct Reference

`#include <ConfigData.h>`

Collaboration diagram for CConfigData::View_Config::View_Colours:



Public Attributes

- struct CConfigData::View_Config::View_Colours::Col_Vehicles Vehicles
- struct CConfigData::View_Config::View_Colours::Col_DrawElements DrawElements

Classes

- struct Col_DrawElements
- struct Col_Vehicles

4.15.1 Detailed Description

Definition at line 92 of file ConfigData.h.

4.15.2 Member Data Documentation

4.15.2.1 struct CConfigData::View_Config::View_Colours::Col_Vehicles CConfigData::View_Config::View_Colours::Vehicles

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.15.2.2 struct CConfigData::View_Config::View_Colours::Col_DrawElements CConfigData::View_Config::View_Colours::DrawElements

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

The documentation for this struct was generated from the following file:

- D:/~Research/Code/C++/EvolveTraffic/ConfigData.h

4.16 CConfigData::View_Config::View_Colours::Col_DrawElements Struct Reference

`#include <ConfigData.h>`

Public Attributes

- COLORREF CLR_BACKGRND
- COLORREF CLR_ROAD_SURFACE
- COLORREF CLR_ROAD_LINES
- COLORREF CLR_RULER_LINES
- COLORREF CLR_RULER_TEXT
- COLORREF CLR_LEGEND_TEXT
- COLORREF CLR_DET_OUTPUT
- COLORREF CLR_DET_FLOWDENSITY
- COLORREF CLR_DET_HEADWAY
- COLORREF CLR_DET_COMPOSITION
- COLORREF CLR_FEAT_SPEEDLIMIT
- COLORREF CLR_FEAT_GRADIENT

4.16.1 Detailed Description

Definition at line 104 of file ConfigData.h.

4.16.2 Member Data Documentation

4.16.2.1 COLORREF CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_BACKGRND

Definition at line 106 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.16.2.2 COLORREF CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_ROAD_SURFACE

Definition at line 107 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.16.2.3 COLORREF CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_ROAD_LINES

Definition at line 108 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.16.2.4 COLORREF CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_RULER_LINES

Definition at line 109 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.16.2.5 COLORREF CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_RULER_TEXT

Definition at line 110 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.16.2.6 COLORREF CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_LEGEND_TEXT

Definition at line 111 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.16.2.7 COLORREF CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_DET_OUTPUT

Definition at line 112 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.16.2.8 COLORREF CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_DET_FLOWDENSITY

Definition at line 113 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.16.2.9 COLORREF CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_DET_HEADWAY

Definition at line 114 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.16.2.10 COLORREF CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_DET_COMPOSITION

Definition at line 115 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.16.2.11 COLORREF CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_FEAT_SPEEDLIMIT

Definition at line 116 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.16.2.12 COLORREF CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_FEAT_GRADIENT

Definition at line 117 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

The documentation for this struct was generated from the following file:

- D:/~Research/Code/C++/EvolveTraffic/ConfigData.h

4.17 CConfigData::View_Config::View_Colours::Col_Vehicles Struct Reference

`#include <ConfigData.h>`

Public Attributes

- COLORREF VEH_COLOR_CAR
- COLORREF VEH_COLOR_SMALLTRUCK
- COLORREF VEH_COLOR_LARGETRUCK
- COLORREF VEH_COLOR_CRANE

- COLORREF VEH_COLOR_LOWLOADER

4.17.1 Detailed Description

Definition at line 94 of file ConfigData.h.

4.17.2 Member Data Documentation

4.17.2.1 COLORREF CConfigData::View_Config::View_Colours::Col_Vehicles::VEH_COLOR_CAR

Definition at line 96 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.17.2.2 COLORREF CConfigData::View_Config::View_Colours::Col_Vehicles::VEH_COLOR_SMALLTRUCK

Definition at line 97 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.17.2.3 COLORREF CConfigData::View_Config::View_Colours::Col_Vehicles::VEH_COLOR_LARGETRUCK

Definition at line 98 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.17.2.4 COLORREF CConfigData::View_Config::View_Colours::Col_Vehicles::VEH_COLOR_CRANE

Definition at line 99 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

4.17.2.5 COLORREF CConfigData::View_Config::View_Colours::Col_Vehicles::VEH_COLOR_LOWLOADER

Definition at line 100 of file ConfigData.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView(), CConfigData::ExtractData(), and CConfigData::SetDefaults().

The documentation for this struct was generated from the following file:

- D:/~Research/Code/C++/EvolveTraffic/ConfigData.h

4.18 CCSVParse Class Reference

A class to parse Comma Seperated Values input.

`#include <CSVParse.h>`

Public Member Functions

- void CloseFile ()
- bool OpenFile (string inFile, string sep)
- CCSVParse ()
- virtual ~CCSVParse ()
- int getline (string &)
- string getfield (int n)
- int getnfield () const

Private Member Functions

- int split ()
- int endofline (char)
- int advplain (const string &line, string &fld, int)
- int advquoted (const string &line, string &fld, int)

Private Attributes

- ifstream fin
- string line
- vector< string > field
- int nfield
- string fieldsep

4.18.1 Detailed Description

A class to parse Comma Seperated Values input.

Definition at line 26 of file CSVParse.h.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 CCSVParse::CCSVParse ()

Definition at line 19 of file CSVParse.cpp.

```
20 {
21
22 }
```

4.18.2.2 CCSVParse::~CCSVParse () [virtual]

Definition at line 24 of file CSVParse.cpp.

```
25 {
26
27 }
```

4.18.3 Member Function Documentation

4.18.3.1 void CCSVParse::CloseFile ()

Definition at line 40 of file CSVParse.cpp.

References fin.

Referenced by CConfigData::ExtractData().

```
41 {
42         fin.close();
43 }
```

4.18.3.2 bool CCSVParse::OpenFile (string *inFile*, string *sep*)

Definition at line 29 of file CSVParse.cpp.

References fieldsep, and fin.

Referenced by CConfigData::ReadData().

```
30 {
31         fin.open( inFile.c_str(), ios::in );
32
33         if(!fin)
34                 return false;
35
36         fieldsep = sep;
37         return true;
38 }
```

4.18.3.3 int CCSVParse::getline (string & *str*)

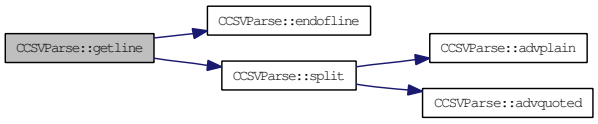
Definition at line 60 of file CSVParse.cpp.

References endofline(), fin, line, and split().

Referenced by CConfigData::GetNextDataLine().

```
61 {
62         char c;
63
64         for (line = ""; fin.get(c) && !endofline(c); )
65                 line += c;
66         split();
67         str = line;
68         return !fin.eof();
69 }
```

Here is the call graph for this function:



4.18.3.4 string CCSVParse::getfield (int *n*)

Definition at line 132 of file CSVParse.cpp.

References field, and nfield.

Referenced by CConfigData::stringToColour().

```
133 {
134         if (n < 0 || n >= nfield)
135                 return "";
136         else
137                 return field[n];
138 }
```

4.18.3.5 int CCSVParse::getnfield () const [inline]

Definition at line 37 of file CSVParse.h.

References nfield.

```
37 { return nfield; }
```

4.18.3.6 int CCSVParse::split () [private]

Definition at line 72 of file CSVParse.cpp.

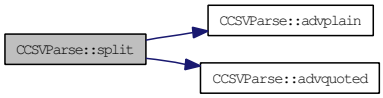
References advplain(), advquoted(), field, line, and nfield.

Referenced by getline().

```
73 {
74         string fld;
75         int i, j;
76
77         nfield = 0;
78         if (line.length() == 0)
79                 return 0;
80         i = 0;
81
82         do {
83                 if (i < line.length() && line[i] == '"')
84                         j = advquoted(line, fld, ++i);  // skip quote
85                 else
86                         j = advplain(line, fld, i);
```

```
87                 if (nfield >= field.size())
88                         field.push_back(fld);
89                 else
90                         field[nfield] = fld;
91                 nfield++;
92                 i = j + 1;
93         } while (j < line.length());
94
95         return nfield;
96 }
```

Here is the call graph for this function:



4.18.3.7 int CCSVParse::endofline (char *c*) [private]

Definition at line 46 of file CSVParse.cpp.

References fin.

Referenced by getline().

```
47 {
48         int eol;
49
50         eol = (c=='\r' || c=='\n');
51         if (c == '\r') {
52                 fin.get(c);
53                 if (!fin.eof() && c != '\n')
54                         fin.putback(c); // read too far
55         }
56         return eol;
57 }
```

4.18.3.8 int CCSVParse::advplain (const string & *line*, string & *fld*, int *i*) [private]

Definition at line 119 of file CSVParse.cpp.

References fieldsep.

Referenced by split().

```
120 {
121         int j;
122
123         j = s.find_first_of(fieldsep, i); // look for separator
124         if (j > s.length())               // none found
125                 j = s.length();
126         fld = string(s, i, j-i);
127         return j;
128 }
```

4.18.3.9 int CCSVParse::advquoted (const string & *line*, string & *fld*, int *i*) [private]

Definition at line 99 of file CSVParse.cpp.

References fieldsep.

Referenced by split().

```
100 {
101         int j;
102
103         fld = "";
104         for (j = i; j < s.length(); j++) {
105                 if (s[j] == '"' && s[++j] != '"') {
106                         int k = s.find_first_of(fieldsep, j);
107                         if (k > s.length())     // no separator found
108                                 k = s.length();
109                         for (k -= j; k-- > 0; )
110                                 fld += s[j++];
111                         break;
112                 }
113                 fld += s[j];
114         }
115         return j;
116 }
```

4.18.4 Member Data Documentation

4.18.4.1 ifstream CCSVParse::fin [private]

Definition at line 40 of file CSVParse.h.

Referenced by CloseFile(), endofline(), getline(), and OpenFile().

4.18.4.2 string CCSVParse::line [private]

Definition at line 41 of file CSVParse.h.

Referenced by getline(), and split().

4.18.4.3 vector<string> CCSVParse::field [private]

Definition at line 42 of file CSVParse.h.

Referenced by getfield(), and split().

4.18.4.4 int CCSVParse::nfield [private]

Definition at line 43 of file CSVParse.h.

Referenced by getfield(), getnfield(), and split().

4.18.4.5 string CCSVParse::fieldsep [private]

Definition at line 44 of file CSVParse.h.

Referenced by advplain(), advquoted(), and OpenFile().

The documentation for this class was generated from the following files:

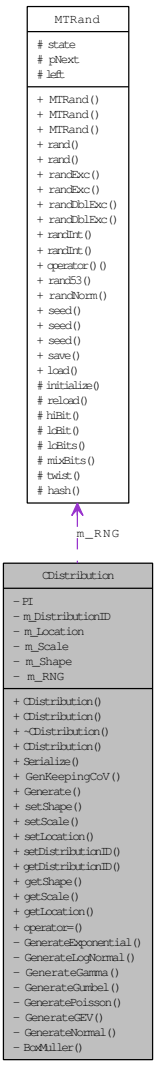
- D:/~Research/Code/C++/EvolveTraffic/CSVParse.h
- D:/~Research/Code/C++/EvolveTraffic/CSVParse.cpp

4.19 CDistribution Class Reference

A class representing a distribution that can be saved to file.

`#include <Distribution.h>`

Collaboration diagram for CDistribution:



Public Member Functions

- CDistribution ()
- CDistribution (double loc, double sc, double sh)
- virtual ~CDistribution ()
- CDistribution (const CDistribution &dist)
- void Serialize (CArchive &ar)
- double GenKeepingCoV (double newLocation)
- double Generate ()
- void setShape (double sh)
- void setScale (double sc)
- void setLocation (double loc)
- void setDistributionID (WORD id)
- WORD getDistributionID () const
- double getShape () const
- double getScale () const
- double getLocation () const
- CDistribution & operator= (const CDistribution &dist)

Private Member Functions

- double GenerateExponential ()
- double GenerateLogNormal ()
- double GenerateGamma ()
- double GenerateGumbel ()
- double GeneratePoisson ()
- double GenerateGEV ()
- double GenerateNormal ()
- double BoxMuller ()

Private Attributes

- const double PI
- WORD m_DistributionID
- double m_Location
- double m_Scale
- double m_Shape

Static Private Attributes

- static MTRand m_RNG

4.19.1 Detailed Description

A class representing a distribution that can be saved to file.

Definition at line 18 of file Distribution.h.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 CDistribution::CDistribution ()

Definition at line 19 of file Distribution.cpp.

References DIST_NORMAL.

```
19                              :PI(3.14159265359)
20 {
21         m_DistributionID = DIST_NORMAL;
22         m_Location = 10.0;
23         m_Scale = 2.0;
24         m_Shape = 0.0;
25 }
```

4.19.2.2 CDistribution::CDistribution (double *loc*, double *sc*, double *sh*)

Definition at line 27 of file Distribution.cpp.

References DIST_NORMAL, m_DistributionID, m_Location, m_Scale, and m_Shape.

```
27                                                                  :PI(3.14159265359)
28 {
29         m_DistributionID = DIST_NORMAL;
30         m_Location = loc;
31         m_Scale = sc;
32         m_Shape = sh;
33 }
```

4.19.2.3 CDistribution::~CDistribution () [virtual]

Definition at line 35 of file Distribution.cpp.

```
36 {
37
38 }
```

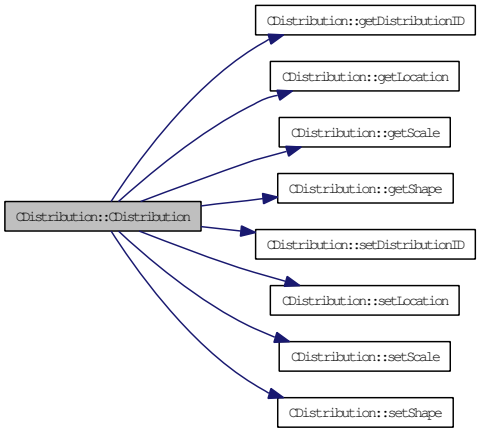
4.19.2.4 CDistribution::CDistribution (const CDistribution & *dist*)

Definition at line 40 of file Distribution.cpp.

References getDistributionID(), getLocation(), getScale(), getShape(), setDistributionID(), setLocation(), setScale(), and setShape().

```
40                                                          :PI(3.14159265359)
41 {
42         this->setDistributionID( dist.getDistributionID() );
43         this->setLocation( dist.getLocation() );
44         this->setScale( dist.getScale() );
45         this->setShape( dist.getShape() );
46 }
```

Here is the call graph for this function:



4.19.3 Member Function Documentation

4.19.3.1 void CDistribution::Serialize (CArchive & *ar*)

Definition at line 69 of file Distribution.cpp.

References m_DistributionID, m_Location, m_Scale, and m_Shape.

Referenced by CParameter::Serialize().

```
70 {
71         if (ar.IsStoring())
72         {
73                 ar      << m_DistributionID
74                         << m_Location
75                         << m_Scale
76                         << m_Shape;
77         }
78         else
79         {
80                 ar      >> m_DistributionID
81                         >> m_Location
82                         >> m_Scale
83                         >> m_Shape;
84         }
85 }
```

4.19.3.2 double CDistribution::GenKeepingCoV (double *newLocation*)

Definition at line 269 of file Distribution.cpp.

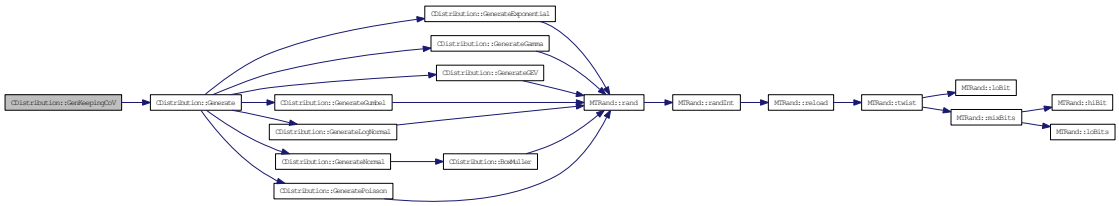
References Generate(), m_Location, and m_Scale.

Referenced by CParameter::GenKeepingCoV().

```
270 {
```

```
271         const double zero = 0.0001;
272         double val = m_Location;
273         if( (m_Scale >  zero || m_Scale < -zero) && (m_Location >  zero || m_Location < -zero)
274         {
275                 double CoV = m_Scale / m_Location;
276                 double OldScale = m_Scale;
277                 double OldLocation = m_Location;
278
279                 // set new params & generate new value
280                 m_Scale = newLocation * CoV;
281                 m_Location = newLocation;
282                 val = Generate();
283                 // restore old params
284                 m_Scale = OldScale;
285                 m_Location = OldLocation;
286         }
287         return val;
288 }
```

Here is the call graph for this function:



4.19.3.3 double CDistribution::Generate ()

Definition at line 141 of file Distribution.cpp.

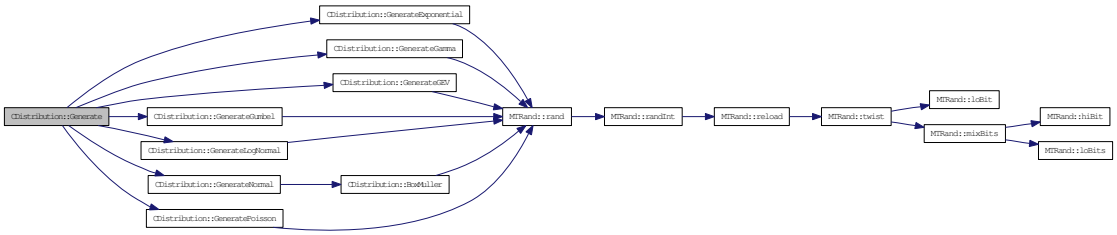
References DIST_CONST, DIST_EXPONENTIAL, DIST_GAMMA, DIST_GEV, DIST_GUMBEL, DIST_LOGNORMAL, DIST_NORMAL, DIST_POISSON, GenerateExponential(), GenerateGamma(), GenerateGEV(), GenerateGumbel(), GenerateLogNormal(), GenerateNormal(), GeneratePoisson(), m_DistributionID, and m_-Location.

Referenced by CParameter::Generate(), and GenKeepingCoV().

```
142 {
143         switch(m_DistributionID)
144         {
145                 case DIST_EXPONENTIAL:
146                         return GenerateExponential();
147                 case DIST_LOGNORMAL:
148                         return GenerateLogNormal();
149                 case DIST_GAMMA:
150                         return GenerateGamma();
151                 case DIST_GUMBEL:
152                         return GenerateGumbel();
153                 case DIST_POISSON:
154                         return GeneratePoisson();
155                 case DIST_GEV:
```

```
156                         return GenerateGEV();
157                 case DIST_NORMAL:
158                         return GenerateNormal();
159                 case DIST_CONST:
160                         return m_Location;
161                 default:
162                         return m_Location;              // constant
163         }
164 }
```

Here is the call graph for this function:



4.19.3.4 void CDistribution::setShape (double *sh*)

Definition at line 135 of file Distribution.cpp.

References m_Shape.

Referenced by CDistribution(), operator=(), and CTrafficConfigDlg::SetParamData().

```
136 {
137         m_Shape = sh;
138 }
```

4.19.3.5 void CDistribution::setScale (double *sc*)

Definition at line 130 of file Distribution.cpp.

References m_Scale.

Referenced by CDistribution(), operator=(), CParameter::SetDefaultParams(), and CTrafficConfigDlg::SetParamData().

```
131 {
132         m_Scale = sc;
133 }
```

4.19.3.6 void CDistribution::setLocation (double *loc*)

Definition at line 125 of file Distribution.cpp.

References m_Location.

Referenced by CDistribution(), operator=(), CParameter::SetDefaultParams(), and CTrafficConfigDlg::SetParamData().

```
126 {
127         m_Location = loc;
128 }
```

4.19.3.7 void CDistribution::setDistributionID (WORD *id*)

Definition at line 120 of file Distribution.cpp.

References m_DistributionID.

Referenced by CDistribution(), operator=(), CParameter::SetDefaultParams(), and CTrafficConfigDlg::SetParamData().

```
121 {
122         m_DistributionID = id;
123 }
```

4.19.3.8 WORD CDistribution::getDistributionID () const

Definition at line 100 of file Distribution.cpp.

References m_DistributionID.

Referenced by CDistribution(), CTrafficConfigDlg::LoadRow(), and operator=().

```
101 {
102         return m_DistributionID;
103 }
```

4.19.3.9 double CDistribution::getShape () const

Definition at line 115 of file Distribution.cpp.

References m_Shape.

Referenced by CDistribution(), CTrafficConfigDlg::LoadRow(), and operator=().

```
116 {
117         return m_Shape;
118 }
```

4.19.3.10 double CDistribution::getScale () const

Definition at line 110 of file Distribution.cpp.

References m_Scale.

Referenced by CDistribution(), CTrafficConfigDlg::LoadRow(), and operator=().

```
111 {
112         return m_Scale;
113 }
```

4.19.3.11 double CDistribution::getLocation () const

Definition at line 105 of file Distribution.cpp.

References m_Location.

Referenced by SpeedLimit::addVehicle(), CDistribution(), CTrafficConfigDlg::LoadRow(), and operator=().

```
106 {
107         return m_Location;
108 }
```

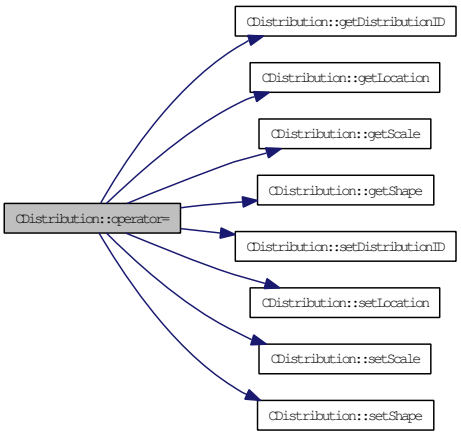
4.19.3.12 CDistribution & CDistribution::operator= (const CDistribution & *dist*)

Definition at line 90 of file Distribution.cpp.

References getDistributionID(), getLocation(), getScale(), getShape(), setDistributionID(), setLocation(), setScale(), and setShape().

```
91 {
92         this->setDistributionID( dist.getDistributionID() );
93         this->setLocation( dist.getLocation() );
94         this->setScale( dist.getScale() );
95         this->setShape( dist.getShape() );
96
97         return *this;
98 }
```

Here is the call graph for this function:



4.19.3.13 double CDistribution::GenerateExponential () [private]

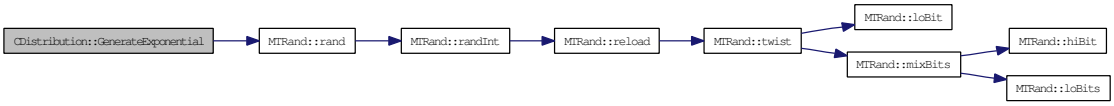
Definition at line 169 of file Distribution.cpp.

References m_RNG, and MTRand::rand().

Referenced by Generate().

```
170 {
171         double u01 = m_RNG.rand();
172         double val = -log(u01);
173         return val;
174 }
```

Here is the call graph for this function:



4.19.3.14 double CDistribution::GenerateLogNormal () [private]

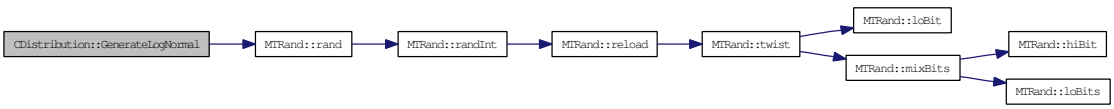
Definition at line 176 of file Distribution.cpp.

References m_Location, m_RNG, m_Scale, PI, and MTRand::rand().

Referenced by Generate().

```
177 {
178         double x[2], u[2];
179         u[0] = m_RNG.rand();
180         u[1] = m_RNG.rand();
181         x[0] = m_Location + m_Scale * sqrt(-2 * log(u[0])) * cos(2 * PI * u[1]);
182         x[1] = m_Location + m_Scale * sqrt(-2 * log(u[0])) * sin(2 * PI * u[1]);
183         double val = exp( x[ (int)(u[1] + 0.5) ] );
184
185         return val;
186 }
```

Here is the call graph for this function:



4.19.3.15 double CDistribution::GenerateGamma () [private]

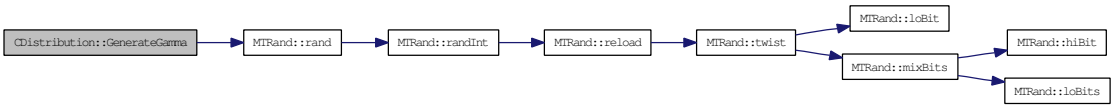
Definition at line 188 of file Distribution.cpp.

References m_Location, m_RNG, m_Scale, and MTRand::rand().

Referenced by Generate().

```
189 {
190         double u[2];
191         u[0] = m_RNG.rand();
192         u[1] = m_RNG.rand();
193         u[2] = m_RNG.rand();
194         double val = (1/m_Location)
195                                         * (-log(u[2]))
196                                         * pow( u[0], (1/m_Scale) )
197                                         / pow( u[0], (1/m_Scale)
198                                         + pow( u[1], (1/(1 - m_Scale)) ) );
199
200         return val;
201 }
```

Here is the call graph for this function:



4.19.3.16 double CDistribution::GenerateGumbel () [private]

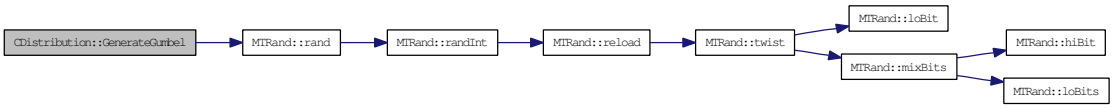
Definition at line 203 of file Distribution.cpp.

References m_Location, m_RNG, m_Scale, and MTRand::rand().

Referenced by Generate().

```
204 {
205         double u01 = m_RNG.rand();
206         double val = m_Location - (1 / m_Scale) * log(log(1 / u01));
207
208         return val;
209 }
```

Here is the call graph for this function:



4.19.3.17 double CDistribution::GeneratePoisson () [private]

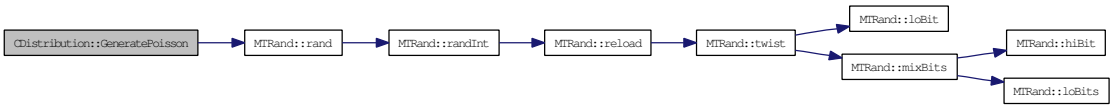
Definition at line 211 of file Distribution.cpp.

References m_Location, m_RNG, m_Scale, PI, and MTRand::rand().

Referenced by Generate().

```
212 {
213         double x[2], u[2];
214         u[0] = m_RNG.rand();
215         u[1] = m_RNG.rand();
216
217         double m = m_Location - 0.5;
218         double sigma = sqrt(m_Scale);
219
220         x[0] = m + sigma * sqrt(-2 * log(u[0])) * cos(2 * PI * u[1]);
221         x[1] = m + sigma * sqrt(-2 * log(u[0])) * sin(2 * PI * u[1]);
222         double val = x[ (int)(u[1] + 0.5) ];
223
224         return val;
225 }
```

Here is the call graph for this function:



4.19.3.18 double CDistribution::GenerateGEV () [private]

Definition at line 227 of file Distribution.cpp.

References m_Location, m_RNG, m_Scale, m_Shape, and MTRand::rand().

Referenced by Generate().

```
228 {
229         double u01 = m_RNG.rand();
230         double val = m_Location + m_Scale * (1 - pow(-log(u01),m_Shape)) / m_Shape;
231
232         return val;
233 }
```

Here is the call graph for this function:

4.19.3.19 double CDistribution::GenerateNormal () [private]

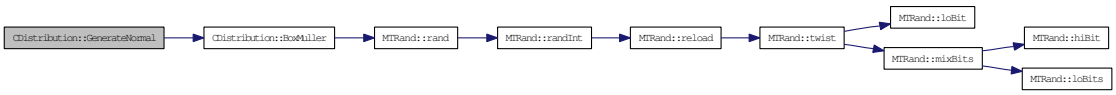
Definition at line 235 of file Distribution.cpp.

References BoxMuller(), m_Location, and m_Scale.

Referenced by Generate().

```
236 {
237         return m_Location + m_Scale * BoxMuller();
238 }
```

Here is the call graph for this function:



4.19.3.20 double CDistribution::BoxMuller () [private]

Definition at line 240 of file Distribution.cpp.

References m_RNG, and MTRand::rand().

Referenced by GenerateNormal().

```
241 {
242         // normal random variate generator
243
244         double x1, x2, w, y1;
245         static double y2;
246         static int use_last = 0;
247
248         if (use_last)                   // use value from previous call
249         {
250                 y1 = y2;
251                 use_last = 0;
252         }
253         else
254         {
255                 do {
256                         x1 = 2.0 * m_RNG.rand() - 1.0;
257                         x2 = 2.0 * m_RNG.rand() - 1.0;
258                         w = x1 * x1 + x2 * x2;
259                 } while ( w >= 1.0 );
260
261                 w = sqrt( (-2.0 * log( w ) ) / w );
262                 y1 = x1 * w;
263                 y2 = x2 * w;
264                 use_last = 1;
265         }
266         return( y1 );
267 }
```

Here is the call graph for this function:

4.19.4 Member Data Documentation

4.19.4.1 MTRand CDistribution::m_RNG [static, private]

Definition at line 66 of file Distribution.h.

Referenced by BoxMuller(), GenerateExponential(), GenerateGamma(), GenerateGEV(), GenerateGumbel(), GenerateLogNormal(), and GeneratePoisson().

4.19.4.2 const double CDistribution::PI [private]

Definition at line 67 of file Distribution.h.

Referenced by GenerateLogNormal(), and GeneratePoisson().

4.19.4.3 WORD CDistribution::m_DistributionID [private]

Definition at line 69 of file Distribution.h.

Referenced by CDistribution(), Generate(), getDistributionID(), Serialize(), and setDistributionID().

4.19.4.4 double CDistribution::m_Location [private]

Definition at line 70 of file Distribution.h.

Referenced by CDistribution(), Generate(), GenerateGamma(), GenerateGEV(), GenerateGumbel(), GenerateLogNormal(), GenerateNormal(), GeneratePoisson(), GenKeepingCoV(), getLocation(), Serialize(), and setLocation().

4.19.4.5 double CDistribution::m_Scale [private]

Definition at line 71 of file Distribution.h.

Referenced by CDistribution(), GenerateGamma(), GenerateGEV(), GenerateGumbel(), GenerateLogNormal(), GenerateNormal(), GeneratePoisson(), GenKeepingCoV(), getScale(), Serialize(), and setScale().

4.19.4.6 double CDistribution::m_Shape [private]

Definition at line 72 of file Distribution.h.

Referenced by CDistribution(), GenerateGEV(), getShape(), Serialize(), and setShape().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/Distribution.h
- D:/~Research/Code/C++/EvolveTraffic/Distribution.cpp

4.20 CEvolveTrafficApp Class Reference

A class for the applicaiton object.

```
#include <EvolveTraffic.h>
```

Public Member Functions

- void setInSimulation (bool InSim)
- CEvolveTrafficApp ()

  A class represetning the application object.

- virtual BOOL InitInstance ()
- virtual BOOL OnIdle (LONG lCount)
- afx_msg void OnAppAbout ()

Private Attributes

- bool m_bInSimulation

4.20.1 Detailed Description

A class for the applicaiton object.

Definition at line 25 of file EvolveTraffic.h.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 CEvolveTrafficApp::CEvolveTrafficApp ()

A class represetning the application object.

Definition at line 42 of file EvolveTraffic.cpp.

```
43 {
44         // TODO: add construction code here,
45         // Place all significant initialization in InitInstance
46 }
```

4.20.3 Member Function Documentation

4.20.3.1 void CEvolveTrafficApp::setInSimulation (bool *InSim*)

Definition at line 227 of file EvolveTraffic.cpp.

References m_bInSimulation.

```
228 {
229         m_bInSimulation = InSim;
230 }
```

4.20.3.2 BOOL CEvolveTrafficApp::InitInstance () [virtual]

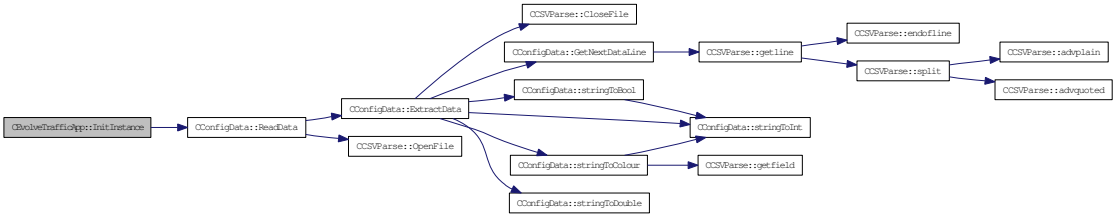
Definition at line 56 of file EvolveTraffic.cpp.

References IDR_MAINFRAME, m_bInSimulation, and CConfigData::ReadData().

```
57 {
58         AfxEnableControlContainer();
59
60         // Standard initialization
61         // If you are not using these features and wish to reduce the size
62         //  of your final executable, you should remove from the following
63         //  the specific initialization routines you do not need.
64
65 #ifdef _AFXDLL
66         Enable3dControls();                     // Call this when using MFC in a shared DLL
67 #else
68         Enable3dControlsStatic();       // Call this when linking to MFC statically
69 #endif
70
71         // Change the registry key under which our settings are stored.
72         // TODO: You should modify this string to be something appropriate
73         // such as the name of your company or organization.
74         SetRegistryKey(_T("Local AppWizard-Generated Applications"));
75
76         LoadStdProfileSettings();  // Load standard INI file options (including MRU)
77
78         if (!g_ConfigData.ReadData("EvolveTraffic.ini") )
79                 MessageBox(NULL,"Configuration file cannot be read", "EvolveTraffic", MB_OK|MB_
80
81         // Register the application's document templates.  Document templates
82         //  serve as the connection between documents, frame windows and views.
83
84         CSingleDocTemplate* pDocTemplate;
85         pDocTemplate = new CSingleDocTemplate(
86                 IDR_MAINFRAME,
87                 RUNTIME_CLASS(CEvolveTrafficDoc),
88                 RUNTIME_CLASS(CMainFrame),       // main SDI frame window
89                 RUNTIME_CLASS(CEvolveTrafficView));
90         AddDocTemplate(pDocTemplate);
91
92         // Enable DDE Execute open
93         EnableShellOpen();
94         RegisterShellFileTypes(TRUE);
95
96         // Parse command line for standard shell commands, DDE, file open
97         CCommandLineInfo cmdInfo;
98         ParseCommandLine(cmdInfo);
99
100         // Dispatch commands specified on the command line
101         if (!ProcessShellCommand(cmdInfo))
102                 return FALSE;
103
104         // The one and only window has been initialized, so show and update it.
105         m_pMainWnd->ShowWindow(SW_SHOW);
106         m_pMainWnd->UpdateWindow();
107
108         // Enable drag/drop open
109         m_pMainWnd->DragAcceptFiles();
110
111         m_bInSimulation = false;                // CC added code
112
113         return TRUE;
```

```
114 }
```

Here is the call graph for this function:



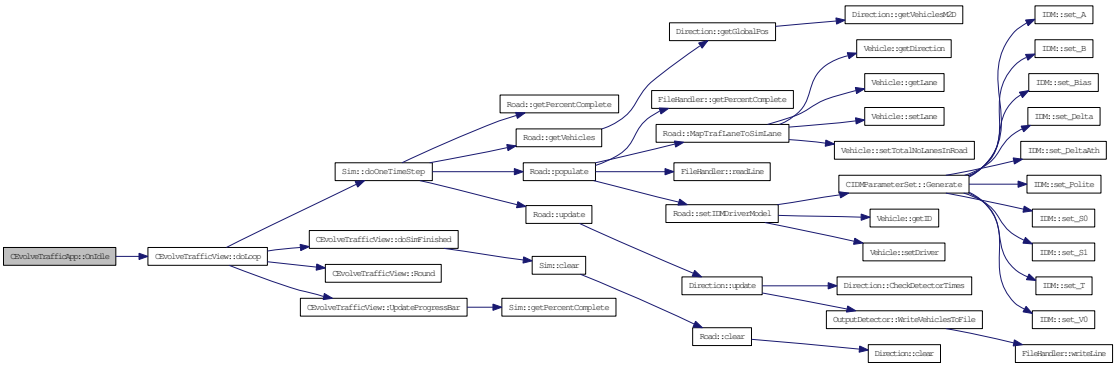
4.20.3.3 BOOL CEvolveTrafficApp::OnIdle (LONG *lCount*) [virtual]

Definition at line 204 of file EvolveTraffic.cpp.

References CEvolveTrafficView::doLoop(), and m_bInSimulation.

```
205 {
206         // The lCount param tells you how many times OnIdle has been called since the last
207         // message was processed.  Windows uses the  0th and 1st call to OnIdle for its own
208         // stuff, so we don't animate until those have been done, otherwise Windows has to
209         // wait after the 0th for the 1st call, and this could cause GUI lag. */
210         CWinApp::OnIdle(lCount);        // Do the base class WinApp processing.
211         if (lCount < 2)                         // Still doing Windows OnIdle processing,
212                 return TRUE;                    // so call OnIdle again.
213
214         CEvolveTrafficView* pView = (CEvolveTrafficView*)((CFrameWnd*)AfxGetMainWnd())->GetAct
215         ASSERT_VALID(pView);
216
217         if(m_bInSimulation)
218                 pView->doLoop( timeGetTime() ); // the main call
219
220         return true;    // loop back here again
221 }
```

Here is the call graph for this function:



4.20.3.4 void CEvolveTrafficApp::OnAppAbout ()

Definition at line 194 of file EvolveTraffic.cpp.

```
195 {
196         CAboutDlg aboutDlg;
197         aboutDlg.DoModal();
198 }
```

4.20.4 Member Data Documentation

4.20.4.1 bool CEvolveTrafficApp::m_bInSimulation [private]

Definition at line 47 of file EvolveTraffic.h.

Referenced by InitInstance(), OnIdle(), and setInSimulation().

The documentation for this class was generated from the following files:

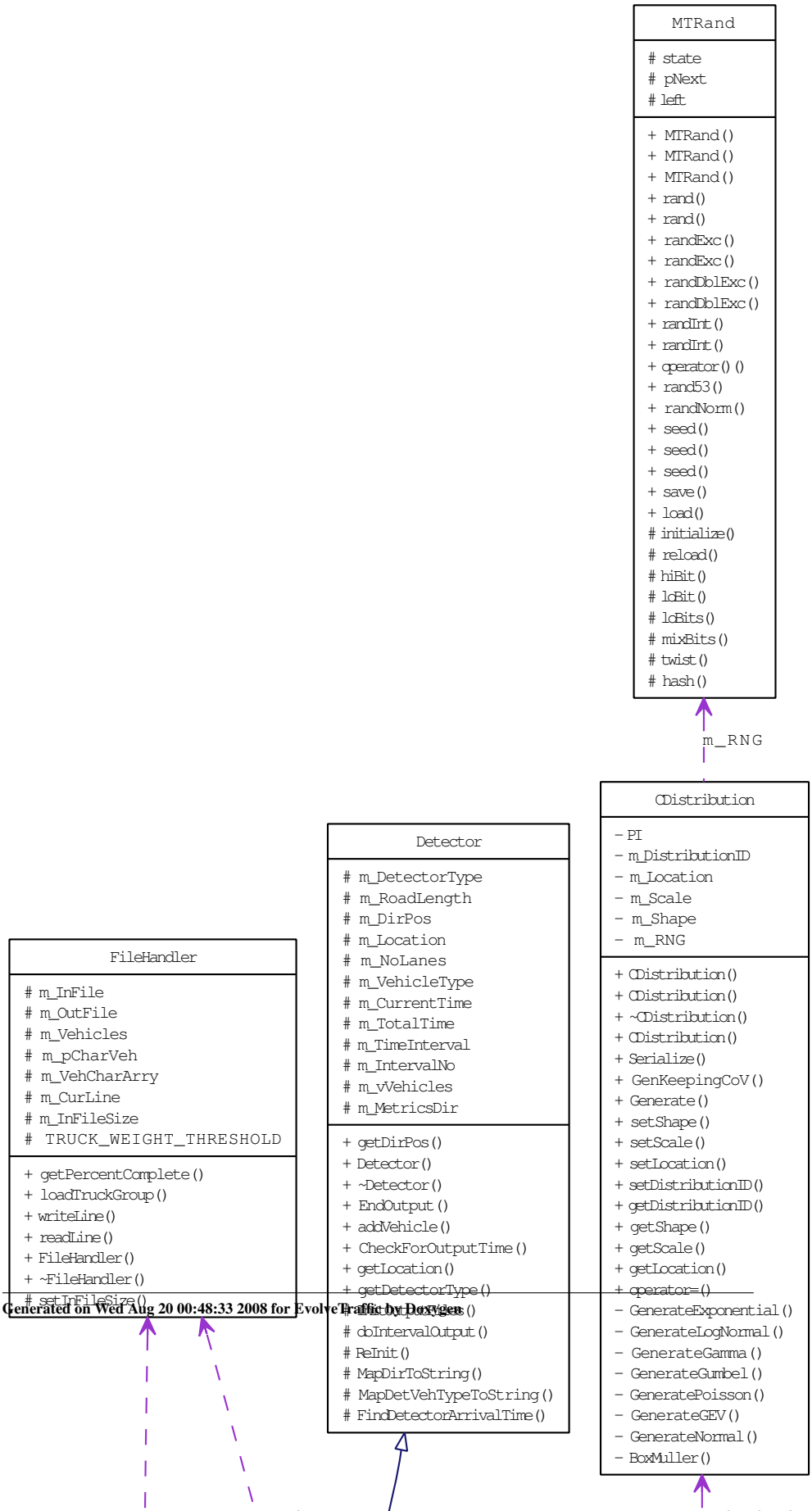
- D:/~Research/Code/C++/EvolveTraffic/EvolveTraffic.h
- D:/~Research/Code/C++/EvolveTraffic/EvolveTraffic.cpp

4.21 CEvolveTrafficDoc Class Reference

A class that stores data relating to the parameters of a simulation.

`#include <EvolveTrafficDoc.h>`

Collaboration diagram for CEvolveTrafficDoc:



Public Member Functions

- virtual BOOL OnNewDocument ()
- virtual void Serialize (CArchive &ar)
- void clear ()
- bool initSim ()
- void UpdateDerivedMembers (bool setMod)
- CObArray * getStatDetectors ()
- CObArray * getRoadFeatures ()
- WORD getFileType () const
- CString getFileIn () const
- CString getFileOut () const
- CString getMetricsDir () const
- int getNoLanesDirPos () const
- int getNoLanesDirNeg () const
- double getSimTimeStep () const
- int getNoDirections () const
- int getNoLanes () const
- int getRoadLength () const
- BOOL getDriveOnRight () const
- int getTrafFileNoLanesDirPos () const
- int getTrafFileNoLanesDirNeg () const
- int getLocOutputDetectorDirPos () const
- int getLocOutputDetectorDirNeg () const
- BOOL getAllowLaneChanging () const
- CIDMParameterSet getIDMParams_Car () const
- CIDMParameterSet getIDMParams_SmallTruck () const
- CIDMParameterSet getIDMParams_LargeTruck () const
- CIDMParameterSet getIDMParams_Crane () const
- CIDMParameterSet getIDMParams_Lowloader () const
- void setDriveOnRight (bool OnRight)
- void setStatDetectors (CObArray *pStatDetectors)
- void setRoadFeatures (CObArray *pRoadFeatures)
- void setFileType (WORD fileType)
- void setFileIn (CString file)
- void setFileOut (CString file)
- void setMetricsDir (CString dir)
- void setNoLanesDirPos (int nlpos)
- void setNoLanesDirNeg (int nlneg)
- void setSimTimeStep (double ts)
- void setNoDirections (int nd)
- void setNoLanes (int nl)
- void setRoadLength (int L)
- void setTrafFileNoLanesDirPos (int nl)
- void setTrafFileNoLanesDirNeg (int nl)
- void setLocOutputDetectorDirPos (int loc)
- void setLocOutputDetectorDirNeg (int loc)

- void setAllowLaneChanging (BOOL status)
- void setIDMParams_Car (CIDMParameterSet theSet)
- void setIDMParams_SmallTruck (CIDMParameterSet theSet)
- void setIDMParams_LargeTruck (CIDMParameterSet theSet)
- void setIDMParams_Crane (CIDMParameterSet theSet)
- void setIDMParams_Lowloader (CIDMParameterSet theSet)
- virtual ~CEvolveTrafficDoc ()

Public Attributes

- Sim m_Sim

Protected Member Functions

- CEvolveTrafficDoc ()

Private Attributes

- CString m_MetricsDir
- CObArray m_vStatDetectors
- CObArray m_vRoadFeatures
- BOOL m_DriveOnRight
- BOOL m_AllowLaneChanging
- int m_LocOutputDetectorDirPos
- int m_LocOutputDetectorDirNeg
- int m_TrafFileNoLanesDirPos
- int m_TrafFileNoLanesDirNeg
- int m_NoVehicleClasses
- CString m_FileIn
- CString m_FileOut
- WORD m_FileType
- int m_NoLanesDirPos
- int m_NoLanesDirNeg
- double m_SimTimeStep
- int m_NoDirections
- int m_NoLanes
- int m_RoadLength
- CIDMParameterSet m_IDMParams_Car
- CIDMParameterSet m_IDMParams_SmallTruck
- CIDMParameterSet m_IDMParams_LargeTruck
- CIDMParameterSet m_IDMParams_Crane
- CIDMParameterSet m_IDMParams_Lowloader

4.21.1 Detailed Description

A class that stores data relating to the parameters of a simulation.

Definition at line 20 of file EvolveTrafficDoc.h.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 CEvolveTrafficDoc::CEvolveTrafficDoc () [protected]

Definition at line 31 of file EvolveTrafficDoc.cpp.

References VEH_ID_CAR, VEH_ID_CRANE, VEH_ID_LARGETRUCK, VEH_ID_LOWLOADER, and VEH_ID_SMALLTRUCK.

```
32 {
33         m_IDMParams_Car                .setVehicleTypeID(VEH_ID_CAR);
34         m_IDMParams_SmallTruck  .setVehicleTypeID(VEH_ID_SMALLTRUCK);
35         m_IDMParams_LargeTruck  .setVehicleTypeID(VEH_ID_LARGETRUCK);
36         m_IDMParams_Crane              .setVehicleTypeID(VEH_ID_CRANE);
37         m_IDMParams_Lowloader   .setVehicleTypeID(VEH_ID_LOWLOADER);
38 }
```

4.21.2.2 CEvolveTrafficDoc::~CEvolveTrafficDoc () [virtual]

Definition at line 40 of file EvolveTrafficDoc.cpp.

References clear().

```
41 {
42         clear();
43 }
```

Here is the call graph for this function:

4.21.3 Member Function Documentation

4.21.3.1 BOOL CEvolveTrafficDoc::OnNewDocument () [virtual]

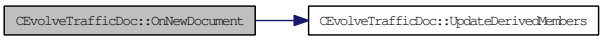
Definition at line 57 of file EvolveTrafficDoc.cpp.

References CASTOR, m_DriveOnRight, m_FileIn, m_FileOut, m_FileType, m_LocOutputDetectorDirNeg, m_LocOutputDetectorDirPos, m_NoLanesDirNeg, m_NoLanesDirPos, m_RoadLength, m_SimTimeStep, m_vRoadFeatures, m_vStatDetectors, and UpdateDerivedMembers().

```
58 {
59         if (!CDocument::OnNewDocument())
60                 return FALSE;
```

```
61
62          m_FileIn = _T("");
63          m_FileOut = _T("");
64          m_RoadLength = 3000;
65          m_NoLanesDirPos = 2;
66          m_NoLanesDirNeg = 1;
67          m_FileType = CASTOR;
68          m_SimTimeStep = 0.1;
69          m_DriveOnRight = true;
70          m_LocOutputDetectorDirPos = 2500;
71          m_LocOutputDetectorDirNeg = 2500;
72
73          CRoadFeature* pFeat = new CRoadFeature;
74          m_vRoadFeatures.Add(pFeat);
75
76          CStatDetector* pDet = new CStatDetector;
77          m_vStatDetectors.Add(pDet);
78
79          UpdateDerivedMembers(false);    // do not call SetModFlag
80
81          return TRUE;
82 }
```

Here is the call graph for this function:



4.21.3.2 void CEvolveTrafficDoc::Serialize (CArchive & *ar*) [virtual]

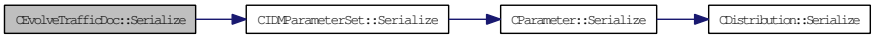
Definition at line 87 of file EvolveTrafficDoc.cpp.

References m_AllowLaneChanging, m_DriveOnRight, m_FileIn, m_FileOut, m_FileType, m_IDMParams_Car, m_IDMParams_Crane, m_IDMParams_LargeTruck, m_IDMParams_Lowloader, m_IDMParams_SmallTruck, m_LocOutputDetectorDirNeg, m_LocOutputDetectorDirPos, m_MetricsDir, m_NoDirections, m_NoLanes, m_NoLanesDirNeg, m_NoLanesDirPos, m_RoadLength, m_SimTimeStep, m_TrafFileNoLanesDirNeg, m_TrafFileNoLanesDirPos, m_vRoadFeatures, m_vStatDetectors, and CIDMParameterSet::Serialize().

```
88 {
89          m_vStatDetectors                .Serialize(ar);
90          m_vRoadFeatures                 .Serialize(ar);
91
92          m_IDMParams_Car                 .Serialize(ar);
93          m_IDMParams_SmallTruck  .Serialize(ar);
94          m_IDMParams_LargeTruck  .Serialize(ar);
95          m_IDMParams_Crane               .Serialize(ar);
96          m_IDMParams_Lowloader   .Serialize(ar);
97
98          if (ar.IsStoring())
99          {
100                 ar      << m_FileIn
101                         << m_FileOut
102                         << m_MetricsDir
103                         << m_FileType
```

```
104                         << m_TrafFileNoLanesDirPos
105                         << m_TrafFileNoLanesDirNeg
106                         << m_NoDirections
107                         << m_NoLanes
108                         << m_NoLanesDirPos
109                         << m_NoLanesDirNeg
110                         << m_RoadLength
111                         << m_DriveOnRight
112                         << m_SimTimeStep
113                         << m_LocOutputDetectorDirPos
114                         << m_LocOutputDetectorDirNeg
115                         << m_AllowLaneChanging;
116         }
117         else
118         {
119                 ar      >> m_FileIn
120                         >> m_FileOut
121                         >> m_MetricsDir
122                         >> m_FileType
123                         >> m_TrafFileNoLanesDirPos
124                         >> m_TrafFileNoLanesDirNeg
125                         >> m_NoDirections
126                         >> m_NoLanes
127                         >> m_NoLanesDirPos
128                         >> m_NoLanesDirNeg
129                         >> m_RoadLength
130                         >> m_DriveOnRight
131                         >> m_SimTimeStep
132                         >> m_LocOutputDetectorDirPos
133                         >> m_LocOutputDetectorDirNeg
134                         >> m_AllowLaneChanging;
135         }
136 }
```

Here is the call graph for this function:



4.21.3.3 void CEvolveTrafficDoc::clear ()

Definition at line 45 of file EvolveTrafficDoc.cpp.

References m_vRoadFeatures, and m_vStatDetectors.

Referenced by ~CEvolveTrafficDoc().

```
46 {
47         for(int i = 0; i < m_vRoadFeatures.GetSize(); i++)
48                 delete m_vRoadFeatures.GetAt(i);
49         m_vRoadFeatures.RemoveAll();
50
51         for(i = 0; i < m_vStatDetectors.GetSize(); i++)
52                 delete m_vStatDetectors.GetAt(i);
53         m_vStatDetectors.RemoveAll();
54
55 }
```

4.21.3.4 bool CEvolveTrafficDoc::initSim ()

Definition at line 165 of file EvolveTrafficDoc.cpp.

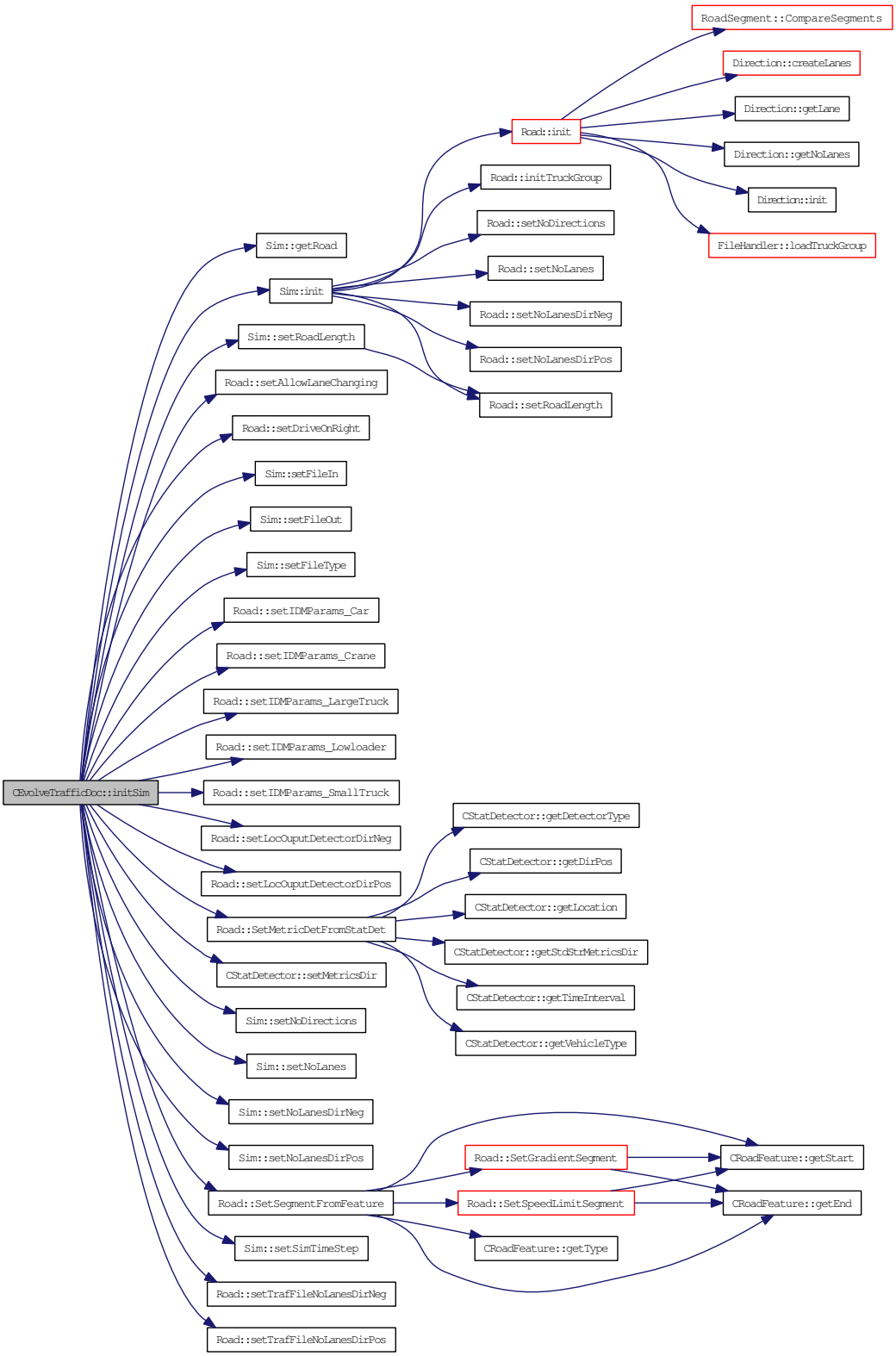
References Sim::getRoad(), Sim::init(), m_AllowLaneChanging, m_DriveOnRight, m_FileIn, m_FileOut, m_FileType, m_IDMParams_Car, m_IDMParams_Crane, m_IDMParams_LargeTruck, m_IDMParams_Lowloader, m_IDMParams_SmallTruck, m_LocOutputDetectorDirNeg, m_LocOutputDetectorDirPos, m_MetricsDir, m_NoDirections, m_NoLanes, m_NoLanesDirNeg, m_NoLanesDirPos, m_RoadLength, m_Sim, m_SimTimeStep, m_TrafFileNoLanesDirNeg, m_TrafFileNoLanesDirPos, m_vRoadFeatures, m_vStatDetectors, Road::setAllowLaneChanging(), Road::setDriveOnRight(), Sim::setFileIn(), Sim::setFileOut(), Sim::setFileType(), Road::setIDMParams_Car(), Road::setIDMParams_Crane(), Road::setIDMParams_LargeTruck(), Road::setIDMParams_Lowloader(), Road::setIDMParams_SmallTruck(), Road::setLocOuputDetectorDirNeg(), Road::setLocOuputDetectorDirPos(), Road::SetMetricDetFromStatDet(), CStatDetector::setMetricsDir(), Sim::setNoDirections(), Sim::setNoLanes(), Sim::setNoLanesDirNeg(), Sim::setNoLanesDirPos(), Sim::setRoadLength(), Road::SetSegmentFromFeature(), Sim::setSimTimeStep(), Road::setTrafFileNoLanesDirNeg(), and Road::setTrafFileNoLanesDirPos().

Referenced by CEvolveTrafficView::doSimStart().

```
166 {
167         CFile FileIn; CFileStatus status;
168         bool bFileExists = FileIn.GetStatus((LPCTSTR)m_FileIn,status);
169         if(!bFileExists)
170         {
171                 MessageBox(NULL, "The input traffic file does not exist","EvolveTraffic",MB_OK
172                 return false;
173         }
174
175         m_Sim.setFileIn( (LPCTSTR)m_FileIn );
176         m_Sim.setFileOut( (LPCTSTR)m_FileOut );
177         m_Sim.setFileType(m_FileType);
178         m_Sim.setNoDirections(m_NoDirections);
179         m_Sim.setNoLanes(m_NoLanes);
180         m_Sim.setNoLanesDirPos(m_NoLanesDirPos);
181         m_Sim.setNoLanesDirNeg(m_NoLanesDirNeg);
182         m_Sim.setRoadLength(m_RoadLength);
183         m_Sim.setSimTimeStep(m_SimTimeStep);
184
185         m_Sim.getRoad()->setIDMParams_Car               (&m_IDMParams_Car);
186         m_Sim.getRoad()->setIDMParams_SmallTruck(&m_IDMParams_SmallTruck);
187         m_Sim.getRoad()->setIDMParams_LargeTruck(&m_IDMParams_LargeTruck);
188         m_Sim.getRoad()->setIDMParams_Crane             (&m_IDMParams_Crane);
189         m_Sim.getRoad()->setIDMParams_Lowloader (&m_IDMParams_Lowloader);
190
191         for(int i = 0; i < m_vRoadFeatures.GetSize(); i++)
192                 m_Sim.getRoad()->SetSegmentFromFeature( reinterpret_cast<CRoadFeature*>(m_vRoad
193
194         for(i = 0; i < m_vStatDetectors.GetSize(); i++)
195         {
196                 CStatDetector* pSD = reinterpret_cast<CStatDetector*>(m_vStatDetectors.GetAt(i)
197                 pSD->setMetricsDir(m_MetricsDir);
198                 m_Sim.getRoad()->SetMetricDetFromStatDet( pSD, m_NoLanesDirPos, m_NoLanesDirNeg
199         }
```

```
200
201         m_Sim.getRoad()->setTrafFileNoLanesDirPos(m_TrafFileNoLanesDirPos);
202         m_Sim.getRoad()->setTrafFileNoLanesDirNeg(m_TrafFileNoLanesDirNeg);
203         m_Sim.getRoad()->setLocOuputDetectorDirPos(m_LocOutputDetectorDirPos);
204         m_Sim.getRoad()->setLocOuputDetectorDirNeg(m_LocOutputDetectorDirNeg);
205         m_Sim.getRoad()->setAllowLaneChanging(m_AllowLaneChanging);
206         m_Sim.getRoad()->setDriveOnRight(m_DriveOnRight);
207
208         m_Sim.init();   // must be called last when all other params are set
209
210         return true;
211 }
```

Here is the call graph for this function:



4.21.3.5 void CEvolveTrafficDoc::UpdateDerivedMembers (bool *setMod*)

Definition at line 156 of file EvolveTrafficDoc.cpp.

References m_NoDirections, m_NoLanes, m_NoLanesDirNeg, and m_NoLanesDirPos.

Referenced by CEvolveTrafficView::OnConfigSim(), and OnNewDocument().

```
157 {
158         m_NoLanes = m_NoLanesDirPos + m_NoLanesDirNeg;
159         m_NoDirections = (m_NoLanesDirPos > 0 && m_NoLanesDirNeg > 0) ? 2 : 1;
160
161         if(setMod)
162                 SetModifiedFlag();      // To implement Save Changes prompts
163 }
```

4.21.3.6 CObArray* CEvolveTrafficDoc::getStatDetectors () [inline]

Definition at line 48 of file EvolveTrafficDoc.h.

References m_vStatDetectors.

Referenced by CEvolveTrafficView::DrawDetectors(), and CEvolveTrafficView::OnConfigMetrics().

```
48 {return &m_vStatDetectors;};
```

4.21.3.7 CObArray* CEvolveTrafficDoc::getRoadFeatures () [inline]

Definition at line 49 of file EvolveTrafficDoc.h.

References m_vRoadFeatures.

Referenced by CEvolveTrafficView::DrawRoadSegments(), and CEvolveTrafficView::OnConfigFeatures().

```
49 {return &m_vRoadFeatures;};
```

4.21.3.8 WORD CEvolveTrafficDoc::getFileType () const [inline]

Definition at line 50 of file EvolveTrafficDoc.h.

References m_FileType.

Referenced by CEvolveTrafficView::OnConfigSim().

```
50 {return m_FileType;};
```

4.21.3.9 CString CEvolveTrafficDoc::getFileIn () const [inline]

Definition at line 51 of file EvolveTrafficDoc.h.

References m_FileIn.

Referenced by CEvolveTrafficView::OnConfigSim().

```
51 {return m_FileIn;};
```

4.21.3.10 CString CEvolveTrafficDoc::getFileOut () const [inline]

Definition at line 52 of file EvolveTrafficDoc.h.

References m_FileOut.

Referenced by CEvolveTrafficView::OnConfigSim().

```
52 {return m_FileOut;};
```

4.21.3.11 CString CEvolveTrafficDoc::getMetricsDir () const [inline]

Definition at line 53 of file EvolveTrafficDoc.h.

References m_MetricsDir.

Referenced by CEvolveTrafficView::OnConfigSim().

```
53 {return m_MetricsDir;};
```

4.21.3.12 int CEvolveTrafficDoc::getNoLanesDirPos () const [inline]

Definition at line 54 of file EvolveTrafficDoc.h.

References m_NoLanesDirPos.

Referenced by CEvolveTrafficView::initRoad(), and CEvolveTrafficView::OnConfigSim().

```
54 {return m_NoLanesDirPos;};
```

4.21.3.13 int CEvolveTrafficDoc::getNoLanesDirNeg () const [inline]

Definition at line 55 of file EvolveTrafficDoc.h.

References m_NoLanesDirNeg.

Referenced by CEvolveTrafficView::initRoad(), and CEvolveTrafficView::OnConfigSim().

```
55 {return m_NoLanesDirNeg;};
```

4.21.3.14 double CEvolveTrafficDoc::getSimTimeStep () const [inline]

Definition at line 56 of file EvolveTrafficDoc.h.

References m_SimTimeStep.

Referenced by CEvolveTrafficView::initRoad(), and CEvolveTrafficView::OnConfigSim().

```
56 {return m_SimTimeStep;};
```

4.21.3.15 int CEvolveTrafficDoc::getNoDirections () const [inline]

Definition at line 57 of file EvolveTrafficDoc.h.

References m_NoDirections.

Referenced by CEvolveTrafficView::initRoad().

```
57 {return m_NoDirections;};
```

4.21.3.16 int CEvolveTrafficDoc::getNoLanes () const [inline]

Definition at line 58 of file EvolveTrafficDoc.h.

References m_NoLanes.

Referenced by CEvolveTrafficView::initRoad().

```
58 {return m_NoLanes;};
```

4.21.3.17 int CEvolveTrafficDoc::getRoadLength () const [inline]

Definition at line 59 of file EvolveTrafficDoc.h.

References m_RoadLength.

Referenced by CEvolveTrafficView::initRoad(), and CEvolveTrafficView::OnConfigSim().

```
59 {return m_RoadLength;};
```

4.21.3.18 BOOL CEvolveTrafficDoc::getDriveOnRight () const [inline]

Definition at line 60 of file EvolveTrafficDoc.h.

References m_DriveOnRight.

Referenced by CEvolveTrafficView::initRoad(), and CEvolveTrafficView::OnConfigSim().

```
60 {return m_DriveOnRight;};
```

4.21.3.19 int CEvolveTrafficDoc::getTrafFileNoLanesDirPos () const [inline]

Definition at line 61 of file EvolveTrafficDoc.h.

References m_TrafFileNoLanesDirPos.

Referenced by CEvolveTrafficView::OnConfigSim().

```
61 {return m_TrafFileNoLanesDirPos;};
```

4.21.3.20 int CEvolveTrafficDoc::getTrafFileNoLanesDirNeg () const [inline]

Definition at line 62 of file EvolveTrafficDoc.h.

References m_TrafFileNoLanesDirNeg.

Referenced by CEvolveTrafficView::OnConfigSim().

```
62 {return m_TrafFileNoLanesDirNeg;};
```

4.21.3.21 int CEvolveTrafficDoc::getLocOutputDetectorDirPos () const [inline]

Definition at line 63 of file EvolveTrafficDoc.h.

References m_LocOutputDetectorDirPos.

Referenced by CEvolveTrafficView::DrawDetectors(), and CEvolveTrafficView::OnConfigSim().

```
63 {return m_LocOutputDetectorDirPos;};
```

4.21.3.22 int CEvolveTrafficDoc::getLocOutputDetectorDirNeg () const [inline]

Definition at line 64 of file EvolveTrafficDoc.h.

References m_LocOutputDetectorDirNeg.

Referenced by CEvolveTrafficView::DrawDetectors(), and CEvolveTrafficView::OnConfigSim().

```
64 {return m_LocOutputDetectorDirNeg;};
```

4.21.3.23 BOOL CEvolveTrafficDoc::getAllowLaneChanging () const [inline]

Definition at line 65 of file EvolveTrafficDoc.h.

References m_AllowLaneChanging.

Referenced by CEvolveTrafficView::OnConfigSim().

```
65 {return m_AllowLaneChanging;};
```

4.21.3.24 CIDMParameterSet CEvolveTrafficDoc::getIDMParams_Car () const [inline]

Definition at line 67 of file EvolveTrafficDoc.h.

References m_IDMParams_Car.

Referenced by CEvolveTrafficView::OnConfigTraf().

```
67 {return m_IDMParams_Car;};
```

4.21.3.25 CIDMParameterSet CEvolveTrafficDoc::getIDMParams_SmallTruck () const [inline]

Definition at line 68 of file EvolveTrafficDoc.h.

References m_IDMParams_SmallTruck.

Referenced by CEvolveTrafficView::OnConfigTraf().

```
68 {return m_IDMParams_SmallTruck;};
```

4.21.3.26 CIDMParameterSet CEvolveTrafficDoc::getIDMParams_LargeTruck () const [inline]

Definition at line 69 of file EvolveTrafficDoc.h.

References m_IDMParams_LargeTruck.

Referenced by CEvolveTrafficView::OnConfigTraf().

```
69 {return m_IDMParams_LargeTruck;};
```

4.21.3.27 CIDMParameterSet CEvolveTrafficDoc::getIDMParams_Crane () const [inline]

Definition at line 70 of file EvolveTrafficDoc.h.

References m_IDMParams_Crane.

Referenced by CEvolveTrafficView::OnConfigTraf().

```
70 {return m_IDMParams_Crane;};
```

4.21.3.28 CIDMParameterSet CEvolveTrafficDoc::getIDMParams_Lowloader () const [inline]

Definition at line 71 of file EvolveTrafficDoc.h.

References m_IDMParams_Lowloader.

Referenced by CEvolveTrafficView::OnConfigTraf().

```
71 {return m_IDMParams_Lowloader;};
```

4.21.3.29 void CEvolveTrafficDoc::setDriveOnRight (bool *OnRight*)

Definition at line 346 of file EvolveTrafficDoc.cpp.

References m_DriveOnRight.

Referenced by CEvolveTrafficView::OnConfigSim().

```
347 {
348         m_DriveOnRight = OnRight;
349 }
```

4.21.3.30 void CEvolveTrafficDoc::setStatDetectors (CObArray * *pStatDetectors*)

Definition at line 340 of file EvolveTrafficDoc.cpp.

References m_vStatDetectors.

Referenced by CEvolveTrafficView::OnConfigMetrics().

```
341 {
342         m_vStatDetectors.Copy( *pStatDetectors );
343         SetModifiedFlag();
344 }
```

4.21.3.31 void CEvolveTrafficDoc::setRoadFeatures (CObArray * *pRoadFeatures*)

Definition at line 334 of file EvolveTrafficDoc.cpp.

References m_vRoadFeatures.

Referenced by CEvolveTrafficView::OnConfigFeatures().

```
335 {
336         m_vRoadFeatures.Copy( *pRoadFeatures );
337         SetModifiedFlag();
338 }
```

4.21.3.32 void CEvolveTrafficDoc::setFileType (WORD *fileType*)

Definition at line 268 of file EvolveTrafficDoc.cpp.

References m_FileType.

Referenced by CEvolveTrafficView::OnConfigSim().

```
269 {
270         m_FileType = fileType;
271         SetModifiedFlag();      // To implement Save Changes prompts
272 }
```

4.21.3.33 void CEvolveTrafficDoc::setFileIn (CString *file*)

Definition at line 250 of file EvolveTrafficDoc.cpp.

References m_FileIn.

Referenced by CEvolveTrafficView::OnConfigSim().

```
251 {
252         m_FileIn = file;
253         SetModifiedFlag();      // To implement Save Changes prompts
254 }
```

4.21.3.34 void CEvolveTrafficDoc::setFileOut (CString *file*)

Definition at line 256 of file EvolveTrafficDoc.cpp.

References m_FileOut.

Referenced by CEvolveTrafficView::OnConfigSim().

```
257 {
258         m_FileOut = file;
259         SetModifiedFlag();      // To implement Save Changes prompts
260 }
```

4.21.3.35 void CEvolveTrafficDoc::setMetricsDir (CString *dir*)

Definition at line 262 of file EvolveTrafficDoc.cpp.

References m_MetricsDir.

Referenced by CEvolveTrafficView::OnConfigSim().

```
263 {
264         m_MetricsDir = dir;
265         SetModifiedFlag();      // To implement Save Changes prompts
266 }
```

4.21.3.36 void CEvolveTrafficDoc::setNoLanesDirPos (int *nlpos*)

Definition at line 238 of file EvolveTrafficDoc.cpp.

References m_NoLanesDirPos.

Referenced by CEvolveTrafficView::OnConfigSim().

```
239 {
240         m_NoLanesDirPos = nlpos;
241         SetModifiedFlag();      // To implement Save Changes prompts
242 }
```

4.21.3.37 void CEvolveTrafficDoc::setNoLanesDirNeg (int *nlneg*)

Definition at line 244 of file EvolveTrafficDoc.cpp.

References m_NoLanesDirNeg.

Referenced by CEvolveTrafficView::OnConfigSim().

```
245 {
246         m_NoLanesDirNeg = nlneg;
247         SetModifiedFlag();      // To implement Save Changes prompts
248 }
```

4.21.3.38 void CEvolveTrafficDoc::setSimTimeStep (double *ts*)

Definition at line 232 of file EvolveTrafficDoc.cpp.

References m_SimTimeStep.

Referenced by CEvolveTrafficView::OnConfigSim().

```
233 {
234         m_SimTimeStep = ts;
235         SetModifiedFlag();      // To implement Save Changes prompts
236 }
```

4.21.3.39 void CEvolveTrafficDoc::setNoDirections (int *nd*)

Definition at line 226 of file EvolveTrafficDoc.cpp.

References m_NoDirections.

```
227 {
228         m_NoDirections = nd;
229         SetModifiedFlag();      // To implement Save Changes prompts
230 }
```

4.21.3.40 void CEvolveTrafficDoc::setNoLanes (int *nl*)

Definition at line 220 of file EvolveTrafficDoc.cpp.

References m_NoLanes.

```
221 {
222         m_NoLanes = nl;
223         SetModifiedFlag();      // To implement Save Changes prompts
224 }
```

4.21.3.41 void CEvolveTrafficDoc::setRoadLength (int *L*)

Definition at line 214 of file EvolveTrafficDoc.cpp.

References m_RoadLength.

Referenced by CEvolveTrafficView::OnConfigSim().

```
215 {
216         m_RoadLength = L;
217         SetModifiedFlag();      // To implement Save Changes prompts
218 }
```

4.21.3.42 void CEvolveTrafficDoc::setTrafFileNoLanesDirPos (int *nl*)

Definition at line 304 of file EvolveTrafficDoc.cpp.

References m_TrafFileNoLanesDirPos.

Referenced by CEvolveTrafficView::OnConfigSim().

```
305 {
306         m_TrafFileNoLanesDirPos = nl;
307         SetModifiedFlag();
308 }
```

4.21.3.43 void CEvolveTrafficDoc::setTrafFileNoLanesDirNeg (int *nl*)

Definition at line 310 of file EvolveTrafficDoc.cpp.

References m_TrafFileNoLanesDirNeg.

Referenced by CEvolveTrafficView::OnConfigSim().

```
311 {
312         m_TrafFileNoLanesDirNeg = nl;
313         SetModifiedFlag();
314 }
```

4.21.3.44 void CEvolveTrafficDoc::setLocOutputDetectorDirPos (int *loc*)

Definition at line 316 of file EvolveTrafficDoc.cpp.

References m_LocOutputDetectorDirPos.

Referenced by CEvolveTrafficView::OnConfigSim().

```
317 {
318         m_LocOutputDetectorDirPos = loc;
319         SetModifiedFlag();
320 }
```

4.21.3.45 void CEvolveTrafficDoc::setLocOutputDetectorDirNeg (int *loc*)

Definition at line 322 of file EvolveTrafficDoc.cpp.

References m_LocOutputDetectorDirNeg.

Referenced by CEvolveTrafficView::OnConfigSim().

```
323 {
324         m_LocOutputDetectorDirNeg = loc;
325         SetModifiedFlag();
326 }
```

4.21.3.46 void CEvolveTrafficDoc::setAllowLaneChanging (BOOL *status*)

Definition at line 328 of file EvolveTrafficDoc.cpp.

References m_AllowLaneChanging.

Referenced by CEvolveTrafficView::OnConfigSim().

```
329 {
330         m_AllowLaneChanging = status;
331         SetModifiedFlag();
332 }
```

4.21.3.47 void CEvolveTrafficDoc::setIDMParams_Car (CIDMParameterSet *theSet*)

Definition at line 274 of file EvolveTrafficDoc.cpp.

References m_IDMParams_Car.

Referenced by CEvolveTrafficView::OnConfigTraf().

```
275 {
276         m_IDMParams_Car = theSet;
277         SetModifiedFlag();      // To implement Save Changes prompts
278 }
```

4.21.3.48 void CEvolveTrafficDoc::setIDMParams_SmallTruck (CIDMParameterSet *theSet*)

Definition at line 280 of file EvolveTrafficDoc.cpp.

References m_IDMParams_SmallTruck.

Referenced by CEvolveTrafficView::OnConfigTraf().

```
281 {
282         m_IDMParams_SmallTruck = theSet;
283         SetModifiedFlag();      // To implement Save Changes prompts
284 }
```

4.21.3.49 void CEvolveTrafficDoc::setIDMParams_LargeTruck (CIDMParameterSet *theSet*)

Definition at line 286 of file EvolveTrafficDoc.cpp.

References m_IDMParams_LargeTruck.

Referenced by CEvolveTrafficView::OnConfigTraf().

```
287 {
288         m_IDMParams_LargeTruck = theSet;
289         SetModifiedFlag();      // To implement Save Changes prompts
290 }
```

4.21.3.50 void CEvolveTrafficDoc::setIDMParams_Crane (CIDMParameterSet *theSet*)

Definition at line 292 of file EvolveTrafficDoc.cpp.

References m_IDMParams_Crane.

Referenced by CEvolveTrafficView::OnConfigTraf().

```
293 {
294         m_IDMParams_Crane = theSet;
295         SetModifiedFlag();      // To implement Save Changes prompts
296 }
```

4.21.3.51 void CEvolveTrafficDoc::setIDMParams_Lowloader (CIDMParameterSet *theSet*)

Definition at line 298 of file EvolveTrafficDoc.cpp.

References m_IDMParams_Lowloader.

Referenced by CEvolveTrafficView::OnConfigTraf().

```
299 {
300         m_IDMParams_Lowloader = theSet;
301         SetModifiedFlag();      // To implement Save Changes prompts
302 }
```

4.21.4 Member Data Documentation

4.21.4.1 Sim CEvolveTrafficDoc::m_Sim

Definition at line 44 of file EvolveTrafficDoc.h.

Referenced by CEvolveTrafficView::doLoop(), CEvolveTrafficView::doSimFinished(), CEvolveTrafficView::DrawTimer(), initSim(), CEvolveTrafficView::OnRunInvisible(), and CEvolveTrafficView::UpdateProgressBar().

4.21.4.2 CString CEvolveTrafficDoc::m_MetricsDir [private]

Definition at line 115 of file EvolveTrafficDoc.h.

Referenced by getMetricsDir(), initSim(), Serialize(), and setMetricsDir().

4.21.4.3 CObArray CEvolveTrafficDoc::m_vStatDetectors [private]

Definition at line 116 of file EvolveTrafficDoc.h.

Referenced by clear(), getStatDetectors(), initSim(), OnNewDocument(), Serialize(), and setStatDetectors().

4.21.4.4 CObArray CEvolveTrafficDoc::m_vRoadFeatures [private]

Definition at line 117 of file EvolveTrafficDoc.h.

Referenced by clear(), getRoadFeatures(), initSim(), OnNewDocument(), Serialize(), and setRoadFeatures().

4.21.4.5 BOOL CEvolveTrafficDoc::m_DriveOnRight [private]

Definition at line 118 of file EvolveTrafficDoc.h.

Referenced by getDriveOnRight(), initSim(), OnNewDocument(), Serialize(), and setDriveOnRight().

4.21.4.6 BOOL CEvolveTrafficDoc::m_AllowLaneChanging [private]

Definition at line 119 of file EvolveTrafficDoc.h.

Referenced by getAllowLaneChanging(), initSim(), Serialize(), and setAllowLaneChanging().

4.21.4.7 int CEvolveTrafficDoc::m_LocOutputDetectorDirPos [private]

Definition at line 120 of file EvolveTrafficDoc.h.

Referenced by getLocOutputDetectorDirPos(), initSim(), OnNewDocument(), Serialize(), and setLocOutputDetectorDirPos().

4.21.4.8 int CEvolveTrafficDoc::m_LocOutputDetectorDirNeg [private]

Definition at line 121 of file EvolveTrafficDoc.h.

Referenced by getLocOutputDetectorDirNeg(), initSim(), OnNewDocument(), Serialize(), and setLocOutputDetectorDirNeg().

4.21.4.9 int CEvolveTrafficDoc::m_TrafFileNoLanesDirPos [private]

Definition at line 122 of file EvolveTrafficDoc.h.

Referenced by getTrafFileNoLanesDirPos(), initSim(), Serialize(), and setTrafFileNoLanesDirPos().

4.21.4.10 int CEvolveTrafficDoc::m_TrafFileNoLanesDirNeg [private]

Definition at line 123 of file EvolveTrafficDoc.h.

Referenced by getTrafFileNoLanesDirNeg(), initSim(), Serialize(), and setTrafFileNoLanesDirNeg().

4.21.4.11 int CEvolveTrafficDoc::m_NoVehicleClasses [private]

Definition at line 124 of file EvolveTrafficDoc.h.

4.21.4.12 CString CEvolveTrafficDoc::m_FileIn [private]

Definition at line 125 of file EvolveTrafficDoc.h.

Referenced by getFileIn(), initSim(), OnNewDocument(), Serialize(), and setFileIn().

4.21.4.13 CString CEvolveTrafficDoc::m_FileOut [private]

Definition at line 126 of file EvolveTrafficDoc.h.

Referenced by getFileOut(), initSim(), OnNewDocument(), Serialize(), and setFileOut().

4.21.4.14 WORD CEvolveTrafficDoc::m_FileType [private]

Definition at line 127 of file EvolveTrafficDoc.h.

Referenced by getFileType(), initSim(), OnNewDocument(), Serialize(), and setFileType().

4.21.4.15 int CEvolveTrafficDoc::m_NoLanesDirPos [private]

Definition at line 128 of file EvolveTrafficDoc.h.

Referenced by getNoLanesDirPos(), initSim(), OnNewDocument(), Serialize(), setNoLanesDirPos(), and UpdateDerivedMembers().

4.21.4.16 int CEvolveTrafficDoc::m_NoLanesDirNeg [private]

Definition at line 129 of file EvolveTrafficDoc.h.

Referenced by getNoLanesDirNeg(), initSim(), OnNewDocument(), Serialize(), setNoLanesDirNeg(), and UpdateDerivedMembers().

4.21.4.17 double CEvolveTrafficDoc::m_SimTimeStep [private]

Definition at line 130 of file EvolveTrafficDoc.h.

Referenced by getSimTimeStep(), initSim(), OnNewDocument(), Serialize(), and setSimTimeStep().

4.21.4.18 int CEvolveTrafficDoc::m_NoDirections [private]

Definition at line 131 of file EvolveTrafficDoc.h.

Referenced by getNoDirections(), initSim(), Serialize(), setNoDirections(), and UpdateDerivedMembers().

4.21.4.19 int CEvolveTrafficDoc::m_NoLanes [private]

Definition at line 132 of file EvolveTrafficDoc.h.

Referenced by getNoLanes(), initSim(), Serialize(), setNoLanes(), and UpdateDerivedMembers().

4.21.4.20 int CEvolveTrafficDoc::m_RoadLength [private]

Definition at line 133 of file EvolveTrafficDoc.h.

Referenced by getRoadLength(), initSim(), OnNewDocument(), Serialize(), and setRoadLength().

4.21.4.21 CIDMParameterSet CEvolveTrafficDoc::m_IDMParams_Car [private]

Definition at line 134 of file EvolveTrafficDoc.h.

Referenced by getIDMParams_Car(), initSim(), Serialize(), and setIDMParams_Car().

4.21.4.22 CIDMParameterSet CEvolveTrafficDoc::m_IDMParams_SmallTruck [private]

Definition at line 135 of file EvolveTrafficDoc.h.

Referenced by getIDMParams_SmallTruck(), initSim(), Serialize(), and setIDMParams_SmallTruck().

4.21.4.23 CIDMParameterSet CEvolveTrafficDoc::m_IDMParams_LargeTruck [private]

Definition at line 136 of file EvolveTrafficDoc.h.

Referenced by getIDMParams_LargeTruck(), initSim(), Serialize(), and setIDMParams_LargeTruck().

4.21.4.24 CIDMParameterSet CEvolveTrafficDoc::m_IDMParams_Crane [private]

Definition at line 137 of file EvolveTrafficDoc.h.

Referenced by getIDMParams_Crane(), initSim(), Serialize(), and setIDMParams_Crane().

4.21.4.25 CIDMParameterSet CEvolveTrafficDoc::m_IDMParams_Lowloader [private]

Definition at line 138 of file EvolveTrafficDoc.h.

Referenced by getIDMParams_Lowloader(), initSim(), Serialize(), and setIDMParams_Lowloader().

The documentation for this class was generated from the following files:

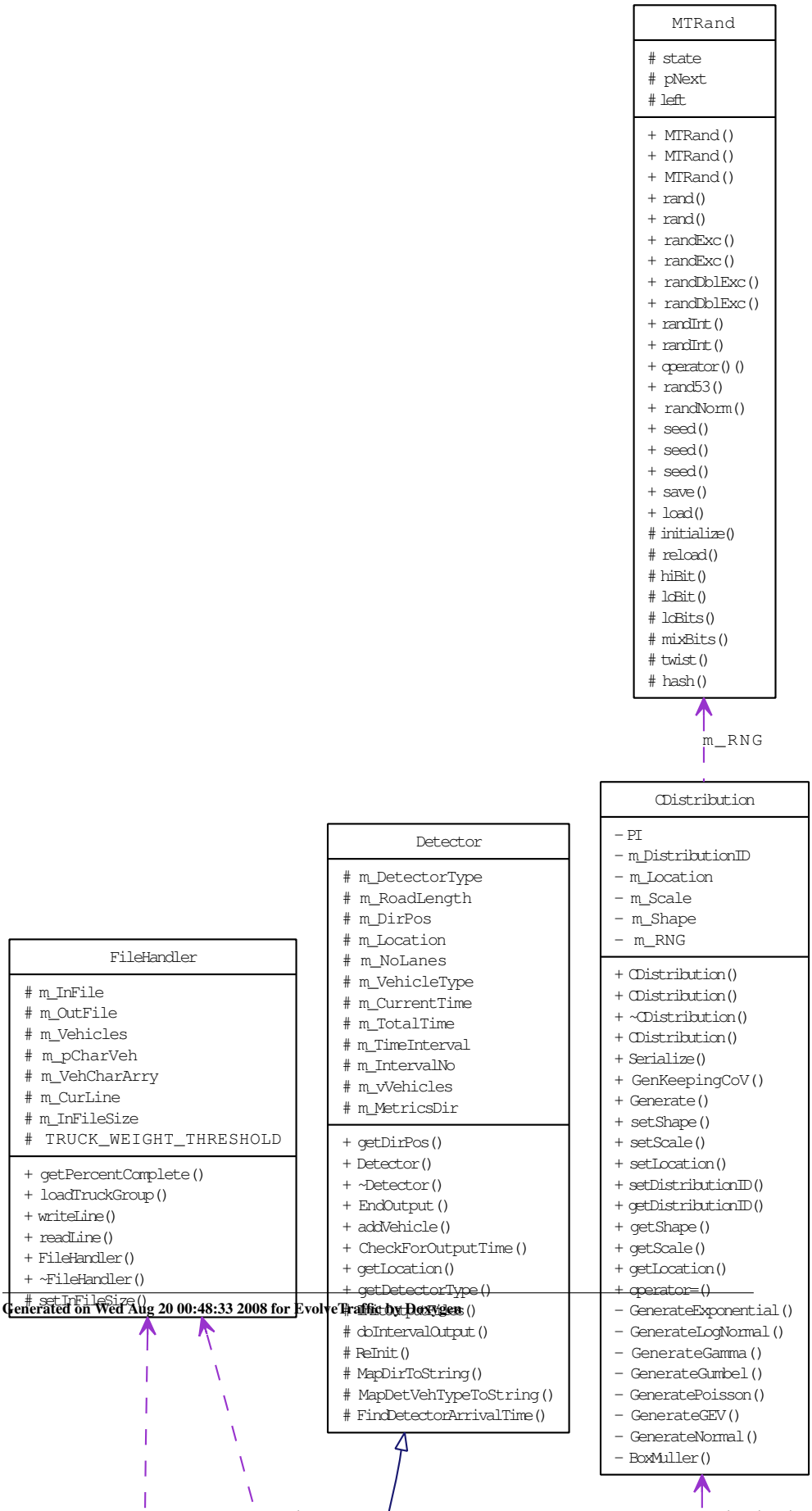
- D:/~Research/Code/C++/EvolveTraffic/EvolveTrafficDoc.h
- D:/~Research/Code/C++/EvolveTraffic/EvolveTrafficDoc.cpp

4.22 CEvolveTrafficView Class Reference

A class for user interaction and drawing functions.

`#include <EvolveTrafficView.h>`

Collaboration diagram for CEvolveTrafficView:



Public Member Functions

- CEvolveTrafficDoc ∗ GetDocument ()
- virtual void OnDraw (CDC ∗pDC)
- virtual BOOL PreCreateWindow (CREATESTRUCT &cs)
- virtual void OnPrepareDC (CDC ∗pDC, CPrintInfo ∗pInfo=NULL)
- void UpdateProgressBar ()
- void doLoop (int curTime)
- void setMaxTimeWarp ()
- virtual ∼CEvolveTrafficView ()

Protected Member Functions

- CEvolveTrafficView ()
- virtual void OnInitialUpdate ()
- virtual BOOL OnPreparePrinting (CPrintInfo ∗pInfo)
- virtual void OnBeginPrinting (CDC ∗pDC, CPrintInfo ∗pInfo)
- virtual void OnEndPrinting (CDC ∗pDC, CPrintInfo ∗pInfo)
- afx_msg void OnConfigSim ()
- afx_msg void OnRunVisible ()
- afx_msg void OnSize (UINT nType, int cx, int cy)
- afx_msg void OnConfigTraf ()
- afx_msg void OnToolsPrefs ()
- afx_msg void OnToolsPause ()
- afx_msg void OnToolsZoomin ()
- afx_msg void OnToolsZoomout ()
- afx_msg void OnToolsSpeedup ()
- afx_msg void OnToolsSlowdown ()
- afx_msg void OnKeyDown (UINT nChar, UINT nRepCnt, UINT nFlags)
- afx_msg void OnRunInvisible ()
- afx_msg void OnConfigFeatures ()
- afx_msg void OnConfigMetrics ()
- afx_msg void OnUpdateToolsPause (CCmdUI ∗pCmdUI)
- afx_msg void OnUpdateToolsSpeedup (CCmdUI ∗pCmdUI)
- afx_msg void OnUpdateToolsSlowdown (CCmdUI ∗pCmdUI)
- afx_msg void OnToolsStop ()
- afx_msg void OnUpdateToolsStop (CCmdUI ∗pCmdUI)
- afx_msg void OnLButtonDown (UINT nFlags, CPoint point)
- afx_msg void OnFileSaveImage ()

Private Member Functions

- Vehicle * FindVehicle (int iLane, int location)
- BOOL WriteWindowToDIB (LPTSTR szFile, CWnd *pWnd)
- void SetZoomScale (double newScale)
- void ValidateTimeWarp ()
- void doSimStart ()
- int Round (double val)
- void doSimFinished (bool bCompleted)
- void UpdateDrawRegion ()
- void SettingUpdateRedraw ()
- void initRoad ()
- void DrawRoad (CMemDC *pDC)
- void DrawRuler (CMemDC *pDC)
- void DrawTimer (CMemDC *pDC)
- void DrawVehicle (CMemDC *pDC, Vehicle *veh)
- void DrawVehicleAt (CMemDC *pDC, WORD VEH_ID, CRect VehRect)
- void DrawDetectors (CMemDC *pDC)
- void DrawLaneChangeDetectors (CMemDC *pDC, bool DirPos)
- void DrawLegend (CMemDC *pDC)
- void DrawLegendElement (CMemDC *pDC, int LineLength, int LineToText, int xpos, int ypos, CString str, COLORREF COL)
- void DrawSingleSegment (CMemDC *pDC, COLORREF COL, WORD HATCH, WORD FeatType, int val, CRect rect, int yTextOffset)
- void DrawRoadSegments (CMemDC *pDC)
- void DrawSingleDetector (CMemDC *pDC, COLORREF COL, int Loc, bool DirPos)
- void DrawLegendSegment (CMemDC *pDC, COLORREF COL, WORD HATCH, WORD FeatType, CString str, int value, int LineToText, int xpos, int ypos)
- void DrawLegendVehicle (CMemDC *pDC, WORD VehType, CString str, int LineToText, int xpos, int ypos)

Private Attributes

- bool m_bShowLegend
- bool m_bShowVelocity
- int m_BtmEdge
- int m_TopEdge
- int m_nLanesOnBtm
- int m_nLanesOnTop
- bool m_DriveOnRight
- double m_ExaggerateWidths
- CSize m_DrawArea
- int m_LastPercentComplete
- double m_MaxTimeWarp
- int m_SimTimeStep

- int m_PauseTime
- int m_StartRealTime
- int m_CurentRealTime
- double m_CurentSimTime
- BOOL m_bPause
- int m_TimerSpeed
- int m_PercentComplete
- CRect m_ClientRect
- int m_YCoordTop
- RECT m_RoadRect
- bool m_bInSimulation
- double m_Scale
- double m_TimeWarp
- double m_VehicleLengthScale
- double m_VehicleWidth
- double m_TickLength
- int m_TickStep
- double m_LaneWidth
- double m_Border_Top
- double m_Border_Btm
- double m_Border_Lhs
- double m_Border_Rhs
- int m_NoDirections
- int m_NoLanes
- int m_NoLanesDirPos
- int m_NoLanesDirNeg
- int m_RoadLength
- int m_NoTicks
- int m_RoadWidth
- M2D m_VehiclePositions
- CEvolveTrafficDoc * m_pDoc
- CProgressBar * m_pProgBar
- CMessageTip m_DataTip
- double MIN_VIEW_SCALE
- double DATATIP_DELAY
- COLORREF CLR_BACKGRND
- COLORREF CLR_DET_COMPOSITION
- COLORREF CLR_DET_FLOWDENSITY
- COLORREF CLR_DET_HEADWAY
- COLORREF CLR_DET_OUTPUT
- COLORREF CLR_FEAT_GRADIENT
- COLORREF CLR_FEAT_SPEEDLIMIT
- COLORREF CLR_LEGEND_TEXT
- COLORREF CLR_ROAD_LINES
- COLORREF CLR_ROAD_SURFACE
- COLORREF CLR_RULER_LINES

- COLORREF CLR_RULER_TEXT
- COLORREF VEH_COLOR_CAR
- COLORREF VEH_COLOR_SMALLTRUCK
- COLORREF VEH_COLOR_LARGETRUCK
- COLORREF VEH_COLOR_CRANE
- COLORREF VEH_COLOR_LOWLOADER

4.22.1 Detailed Description

A class for user interaction and drawing functions.

Definition at line 18 of file EvolveTrafficView.h.

4.22.2 Constructor & Destructor Documentation

4.22.2.1 CEvolveTrafficView::CEvolveTrafficView () [protected]

Definition at line 67 of file EvolveTrafficView.cpp.

References CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_BACKGRND, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_DET_COMPOSITION, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_DET_FLOWDENSITY, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_DET_HEADWAY, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_DET_OUTPUT, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_FEAT_GRADIENT, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_FEAT_SPEEDLIMIT, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_LEGEND_TEXT, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_ROAD_LINES, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_ROAD_SURFACE, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_RULER_LINES, CConfigData::View_Config::View_Colours::Col_DrawElements::CLR_RULER_TEXT, CConfigData::View_Config::Colours, CConfigData::View_Config::DATATIP_DELAY, DRAW_FRAMES_PER_SEC, CConfigData::View_Config::View_Colours::DrawElements, CConfigData::View_Config::MIN_VIEW_SCALE, CConfigData::View_Config::View_Colours::Col_Vehicles::VEH_COLOR_CAR, CConfigData::View_Config::View_Colours::Col_Vehicles::VEH_COLOR_CRANE, CConfigData::View_Config::View_Colours::Col_Vehicles::VEH_COLOR_LARGETRUCK, CConfigData::View_Config::View_Colours::Col_Vehicles::VEH_COLOR_LOWLOADER, CConfigData::View_Config::View_Colours::Col_Vehicles::VEH_COLOR_SMALLTRUCK, CConfigData::View_Config::View_Colours::Vehicles, and CConfigData::View.

```
68 {
69         // Configuration Settings
70         MIN_VIEW_SCALE          = g_ConfigData.View.MIN_VIEW_SCALE;
71         DATATIP_DELAY           = g_ConfigData.View.DATATIP_DELAY;
72
```

```
73          CLR_BACKGRND            = g_ConfigData.View.Colours.DrawElements.CLR_BACKGRND;
74          CLR_DET_COMPOSITION = g_ConfigData.View.Colours.DrawElements.CLR_DET_COMPOSITION;
75          CLR_DET_FLOWDENSITY = g_ConfigData.View.Colours.DrawElements.CLR_DET_FLOWDENSITY;
76          CLR_DET_HEADWAY         = g_ConfigData.View.Colours.DrawElements.CLR_DET_HEADWAY;
77          CLR_DET_OUTPUT          = g_ConfigData.View.Colours.DrawElements.CLR_DET_OUTPUT;
78          CLR_FEAT_GRADIENT       = g_ConfigData.View.Colours.DrawElements.CLR_FEAT_GRADIENT;
79          CLR_FEAT_SPEEDLIMIT     = g_ConfigData.View.Colours.DrawElements.CLR_FEAT_SPEEDLIMIT;
80          CLR_LEGEND_TEXT         = g_ConfigData.View.Colours.DrawElements.CLR_LEGEND_TEXT;
81          CLR_ROAD_LINES          = g_ConfigData.View.Colours.DrawElements.CLR_ROAD_LINES;
82          CLR_ROAD_SURFACE        = g_ConfigData.View.Colours.DrawElements.CLR_ROAD_SURFACE;
83          CLR_RULER_LINES         = g_ConfigData.View.Colours.DrawElements.CLR_RULER_LINES;
84          CLR_RULER_TEXT          = g_ConfigData.View.Colours.DrawElements.CLR_RULER_TEXT;
85
86          VEH_COLOR_CAR                   = g_ConfigData.View.Colours.Vehicles.VEH_COLOR_CAR;
87          VEH_COLOR_SMALLTRUCK    = g_ConfigData.View.Colours.Vehicles.VEH_COLOR_SMALLTRUCK;
88          VEH_COLOR_LARGETRUCK    = g_ConfigData.View.Colours.Vehicles.VEH_COLOR_LARGETRUCK;
89          VEH_COLOR_CRANE                 = g_ConfigData.View.Colours.Vehicles.VEH_COLOR_CRANE;
90          VEH_COLOR_LOWLOADER             = g_ConfigData.View.Colours.Vehicles.VEH_COLOR_LOWLOADER
91
92          // Initial view scale
93          m_Scale = 1.0;
94          m_ExaggerateWidths       = 10.0;
95
96          m_Border_Top                    = 20;
97          m_Border_Btm                    = 40;
98          m_Border_Lhs                    = 20;
99          m_Border_Rhs                    = 100;
100          m_TickStep                             = 500;
101          m_VehicleLengthScale    = 2.0;  // so they look ok - results in overlaps??
102          m_bShowVelocity                 = true;
103          m_bShowLegend                   = true;
104          m_TickLength                    = 1.0 * m_ExaggerateWidths;
105          m_LaneWidth                             = 4.0 * m_ExaggerateWidths;
106          m_VehicleWidth                  = 3.0 * m_ExaggerateWidths;             // less than la
107
108          m_bInSimulation = false;        // has a simulation begun
109          m_bPause = false;                       // is the simulation to progress one step
110          m_TimeWarp = 1;                         // multiplier of real time that the simulation
111          // the timer speed is constant since the refresh rate is constant
112          m_TimerSpeed = Round(1000/DRAW_FRAMES_PER_SEC); // in int ms
113          m_CurentRealTime = 0;           // actual time since simulation started
114          m_CurentSimTime = 0;            // sim time since sim started
115          m_PauseTime = 0;                        // the time the sim was paused at
116          m_MaxTimeWarp = 1000;           // a high number - will be reduced as sim progresses
117
118          m_LastPercentComplete = 0;
119          m_PercentComplete = 0;
120
121          m_YCoordTop = 0;        // legacy - need to remove
122 }
```

4.22.2.2 CEvolveTrafficView::~CEvolveTrafficView () [virtual]

Definition at line 124 of file EvolveTrafficView.cpp.

```
125 {
126
127 }
```

4.22.3 Member Function Documentation

4.22.3.1 CEvolveTrafficDoc * CEvolveTrafficView::GetDocument () [inline]

Definition at line 193 of file EvolveTrafficView.h.

Referenced by OnInitialUpdate().

```
194     { return (CEvolveTrafficDoc*)m_pDocument; }
```

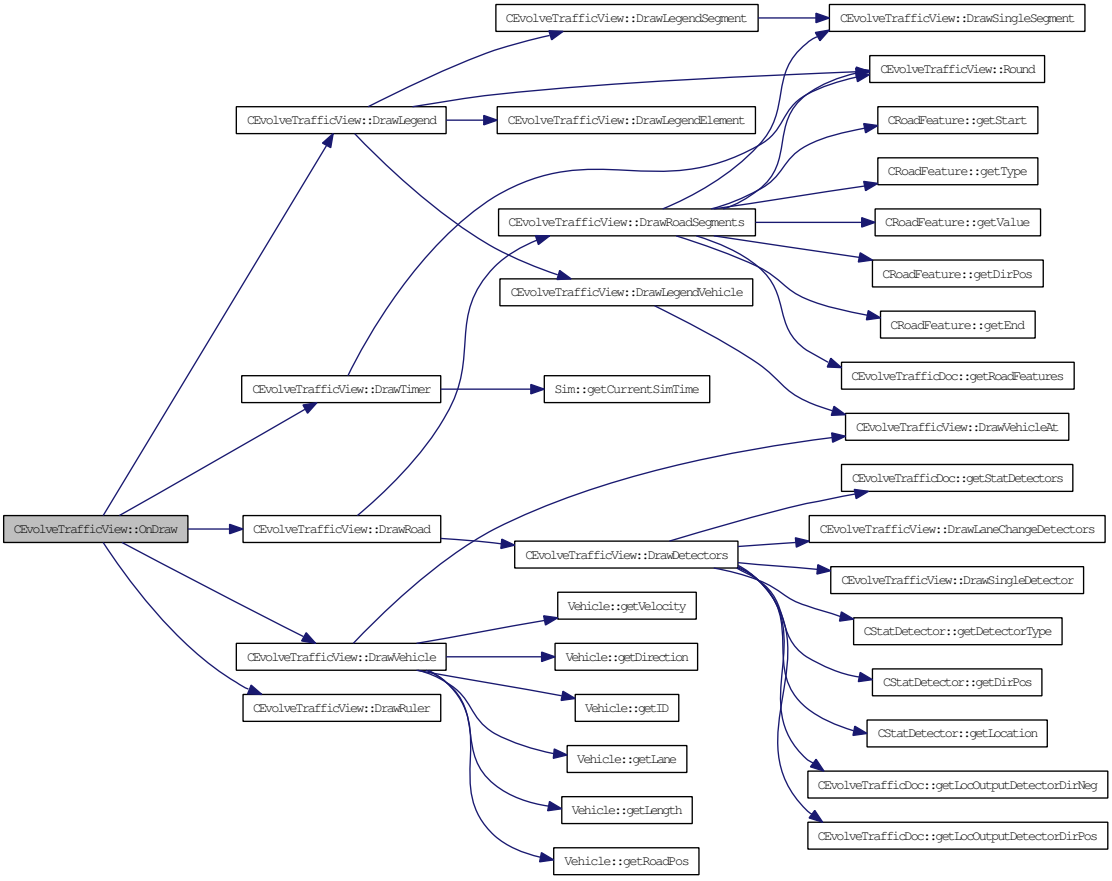
4.22.3.2 void CEvolveTrafficView::OnDraw (CDC * *pDC*) [virtual]

Definition at line 287 of file EvolveTrafficView.cpp.

References DrawLegend(), DrawRoad(), DrawRuler(), DrawTimer(), DrawVehicle(), m_bInSimulation, m_bShowLegend, and m_VehiclePositions.

```
288 {
289         CMemDC pDC(dc);
290
291         DrawRoad(pDC);
292         DrawRuler(pDC);
293
294         if(m_bShowLegend)
295                 DrawLegend(pDC);
296
297         if(m_bInSimulation)             // we're in a simulation
298         {
299                 // draw the vehicles where they are
300                 for (int i = 0; i < m_VehiclePositions.size(); i++)                     // Tota
301                 {
302                         for (int j = 0; j < m_VehiclePositions[i].size(); j++)  // No vehicles
303                                         DrawVehicle(pDC, m_VehiclePositions[i][j]);
304                 }
305         }
306
307         DrawTimer(pDC); // always draw & last so it's on top
308
309 }
```

Here is the call graph for this function:



4.22.3.3 BOOL CEvolveTrafficView::PreCreateWindow (CREATESTRUCT & *cs*) [virtual]

Definition at line 129 of file EvolveTrafficView.cpp.

```
130 {
131         // TODO: Modify the Window class or styles here by modifying
132         //  the CREATESTRUCT cs
133
134         return CScrollView::PreCreateWindow(cs);
135 }
```

4.22.3.4 void CEvolveTrafficView::OnPrepareDC (CDC * *pDC*, CPrintInfo * *pInfo* = NULL) [virtual]

Definition at line 159 of file EvolveTrafficView.cpp.

References m_Border_Lhs, m_ClientRect, m_DrawArea, and m_Scale.

Referenced by OnLButtonDown().

```
160 {
161         CScrollView::OnPrepareDC(pDC, pInfo);
162 
163         pDC->SetMapMode(MM_ANISOTROPIC);        // for scaling
164         pDC->SetWindowExt(m_DrawArea);          // in logical units
165 
166         int xExtent = m_Scale * m_DrawArea.cx;
167         int yExtent = m_Scale * m_DrawArea.cy;
168         pDC->SetViewportExt(xExtent, yExtent);
169 
170         // keep the road divider in the centre of screen
171         // by moving the viewport origin
172         GetClientRect(&m_ClientRect);   // the height of the window
173         CPoint viewOrg;
174         viewOrg.y = m_ClientRect.Height()/2;    // - yExtent/2;
175         viewOrg.x = m_Border_Lhs*m_Scale;
176 
177         viewOrg -= GetDeviceScrollPosition();   // so scrolling accounted for
178         pDC->SetViewportOrg(viewOrg);
179 }
```

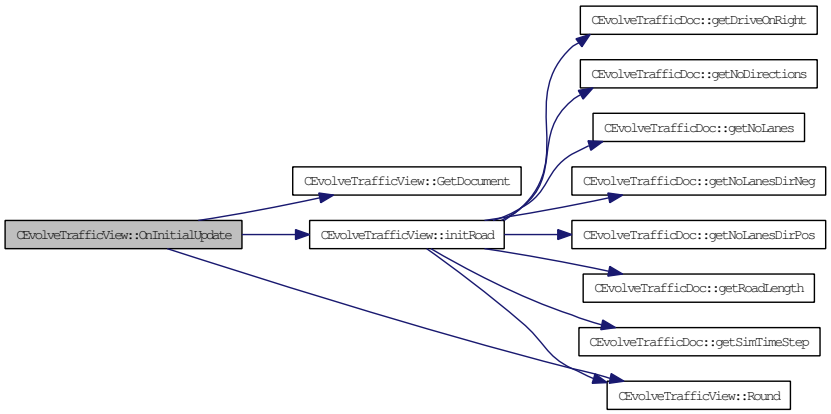
4.22.3.5 void CEvolveTrafficView::OnInitialUpdate () [protected, virtual]

Definition at line 140 of file EvolveTrafficView.cpp.

References DATATIP_DELAY, GetDocument(), initRoad(), m_DataTip, m_DrawArea, m_pDoc, and Round().

```
141 {
142         m_DrawArea.cx = 5000;
143         m_DrawArea.cy = 100;
144 
145         CScrollView::OnInitialUpdate();
146 
147         m_pDoc = GetDocument();
148         ASSERT_VALID(m_pDoc);
149 
150         initRoad();
151         if(m_DataTip.m_hWnd == NULL)
152         {
153                 m_DataTip.Create(this);
154                 m_DataTip.SetDisplayTime( Round(DATATIP_DELAY*1000) );
155         }
156 
157 }
```

Here is the call graph for this function:



4.22.3.6 BOOL CEvolveTrafficView::OnPreparePrinting (CPrintInfo * *pInfo*) [protected, virtual]

Definition at line 314 of file EvolveTrafficView.cpp.

```
315 {
316         // default preparation
317         return DoPreparePrinting(pInfo);
318 }
```

4.22.3.7 void CEvolveTrafficView::OnBeginPrinting (CDC * *pDC*, CPrintInfo * *pInfo*) [protected, virtual]

Definition at line 320 of file EvolveTrafficView.cpp.

```
321 {
322         // TODO: add extra initialization before printing
323 }
```

4.22.3.8 void CEvolveTrafficView::OnEndPrinting (CDC * *pDC*, CPrintInfo * *pInfo*) [protected, virtual]

Definition at line 325 of file EvolveTrafficView.cpp.

```
326 {
327         // TODO: add cleanup after printing
328 }
```

4.22.3.9 void CEvolveTrafficView::UpdateProgressBar ()

Definition at line 272 of file EvolveTrafficView.cpp.

References Sim::getPercentComplete(), m_LastPercentComplete, m_pDoc, m_PercentComplete, m_pProgBar, and CEvolveTrafficDoc::m_Sim.

Referenced by doLoop(), and OnRunInvisible().

```
273 {
274         m_PercentComplete = m_pDoc->m_Sim.getPercentComplete();
275
276         if(m_PercentComplete > m_LastPercentComplete)
277         {
278                 CString str;
279                 str.Format("%d%% complete", m_PercentComplete);
280                 m_pProgBar->SetText(str);
281                 m_pProgBar->StepIt();
282                 m_LastPercentComplete = m_PercentComplete;
283                 Invalidate(FALSE);      // called so that during invisible the bar is still see
284         }
285 }
```

Here is the call graph for this function:

4.22.3.10 void CEvolveTrafficView::doLoop (int *curTime*)

Definition at line 242 of file EvolveTrafficView.cpp.

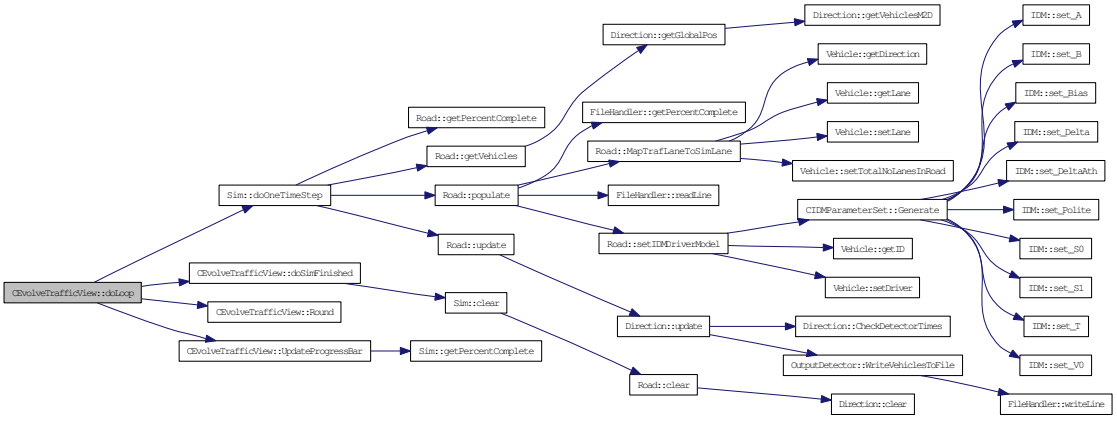
References Sim::doOneTimeStep(), doSimFinished(), m_bInSimulation, m_bPause, m_pDoc, CEvolveTrafficDoc::m_Sim, m_SimTimeStep, m_StartRealTime, m_TimerSpeed, m_TimeWarp, m_VehiclePositions, Round(), and UpdateProgressBar().

Referenced by CEvolveTrafficApp::OnIdle().

```
243 {
244         int StepTime = Round( (double)m_SimTimeStep/m_TimeWarp );
245         static int TimeNextSimStep = m_StartRealTime + StepTime;
246         static int TimeNextDraw = m_StartRealTime + m_TimerSpeed;
247
248         if(m_bInSimulation && !m_bPause)        // simulating & not paused
249         {
250                 if(curTime > TimeNextSimStep)   // ready for next simstep
251                 {
252                         // note m_bInSimulation will become false if it's the end of the simula
253                         m_VehiclePositions = m_pDoc->m_Sim.doOneTimeStep(&m_bInSimulation);
254                         UpdateProgressBar();
255
256                         if(!m_bInSimulation) // we were in a simulation and now we're not - ie
257                                 doSimFinished(true);    // true means completed
258
259                         while(TimeNextSimStep < curTime)
260                                 TimeNextSimStep += StepTime;
261                 }
```

```
262         }
263
264         if(curTime > TimeNextDraw)                          // ready for next frame?
265         {
266                 InvalidateRect(NULL,FALSE);                 // draw it
267                 while(TimeNextDraw < curTime)
268                         TimeNextDraw += m_TimerSpeed;
269         }
270 }
```

Here is the call graph for this function:



4.22.3.11 void CEvolveTrafficView::setMaxTimeWarp ()

Definition at line 428 of file EvolveTrafficView.cpp.

References m_MaxTimeWarp, and m_TimeWarp.

```
429 {
430         m_MaxTimeWarp = m_TimeWarp;
431 }
```

4.22.3.12 void CEvolveTrafficView::OnConfigSim () [protected]

Definition at line 594 of file EvolveTrafficView.cpp.

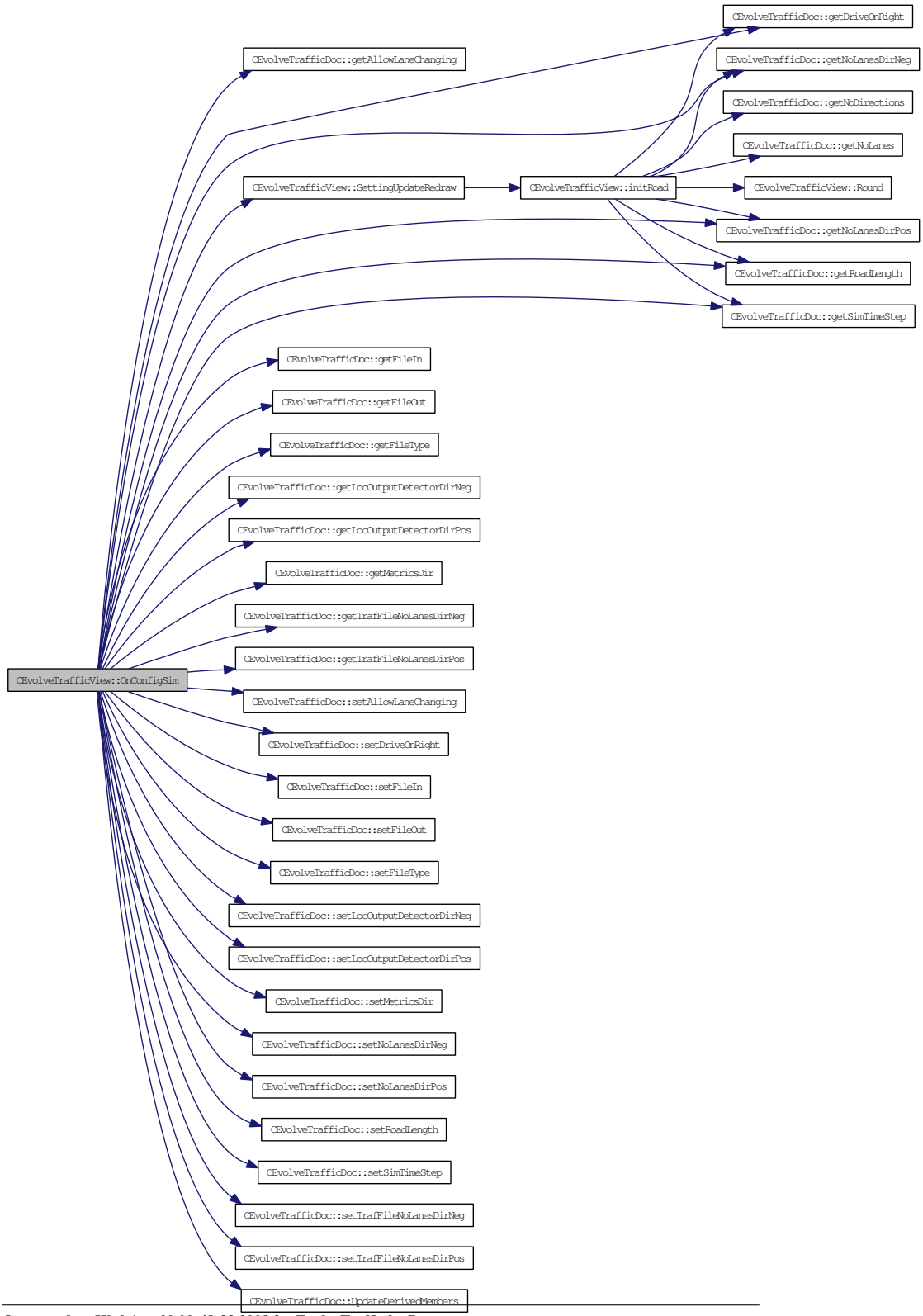
References CEvolveTrafficDoc::getAllowLaneChanging(), CEvolveTrafficDoc::getDriveOnRight(), CEvolveTrafficDoc::getFileIn(), CEvolveTrafficDoc::getFileOut(), CEvolveTrafficDoc::getFileType(), CEvolveTrafficDoc::getLocOutputDetectorDirNeg(), CEvolveTrafficDoc::getLocOutputDetectorDirPos(), CEvolveTrafficDoc::getMetricsDir(), CEvolveTrafficDoc::getNoLanesDirNeg(), CEvolveTrafficDoc::getNoLanesDirPos(), CEvolveTrafficDoc::getRoadLength(), CEvolveTrafficDoc::getSimTimeStep(), CEvolveTrafficDoc::getTrafFileNoLanesDirNeg(), CEvolveTrafficDoc::getTrafFileNoLanesDirPos(), CSimConfigDlg::m_AllowLaneChanging,

CSimConfigDlg::m_DriveOnRight, CSimConfigDlg::m_FileIn, CSimConfigDlg::m_FileOut, CSimConfigDlg::m_FileType, CSimConfigDlg::m_LocOutputDetectorDirNeg, CSimConfigDlg::m_LocOutputDetectorDirPos, CSimConfigDlg::m_MetricsDir, CSimConfigDlg::m_NoLanesDirNeg, CSimConfigDlg::m_NoLanesDirPos, m_pDoc, CSimConfigDlg::m_RoadLength, CSimConfigDlg::m_SimTimeStep, CSimConfigDlg::m_TrafFileNoLanesDirNeg, CSimConfigDlg::m_TrafFileNoLanesDirPos, CEvolveTrafficDoc::setAllowLaneChanging(), CEvolveTrafficDoc::setDriveOnRight(), CEvolveTrafficDoc::setFileIn(), CEvolveTrafficDoc::setFileOut(), CEvolveTrafficDoc::setFileType(), CEvolveTrafficDoc::setLocOutputDetectorDirNeg(), CEvolveTrafficDoc::setLocOutputDetectorDirPos(), CEvolveTrafficDoc::setMetricsDir(), CEvolveTrafficDoc::setNoLanesDirNeg(), CEvolveTrafficDoc::setNoLanesDirPos(), CEvolveTrafficDoc::setRoadLength(), CEvolveTrafficDoc::setSimTimeStep(), SettingUpdateRedraw(), CEvolveTrafficDoc::setTrafFileNoLanesDirNeg(), CEvolveTrafficDoc::setTrafFileNoLanesDirPos(), and CEvolveTrafficDoc::UpdateDerivedMembers().

```
595 {
596         CSimConfigDlg SimDlg;
597 
598         SimDlg.m_FileIn                 = m_pDoc->getFileIn();
599         SimDlg.m_FileOut                = m_pDoc->getFileOut();
600         SimDlg.m_MetricsDir             = m_pDoc->getMetricsDir();
601         SimDlg.m_FileType               = m_pDoc->getFileType();
602         SimDlg.m_RoadLength             = m_pDoc->getRoadLength();
603         SimDlg.m_DriveOnRight   = m_pDoc->getDriveOnRight();
604         SimDlg.m_NoLanesDirPos  = m_pDoc->getNoLanesDirPos();
605         SimDlg.m_NoLanesDirNeg  = m_pDoc->getNoLanesDirNeg();
606         SimDlg.m_SimTimeStep    = (int)(m_pDoc->getSimTimeStep() * 1000);       // seconds to m
607 
608         SimDlg.m_AllowLaneChanging                      = m_pDoc->getAllowLaneChanging();
609         SimDlg.m_TrafFileNoLanesDirPos          = m_pDoc->getTrafFileNoLanesDirPos();
610         SimDlg.m_TrafFileNoLanesDirNeg          = m_pDoc->getTrafFileNoLanesDirNeg();
611         SimDlg.m_LocOutputDetectorDirPos        = m_pDoc->getLocOutputDetectorDirPos();
612         SimDlg.m_LocOutputDetectorDirNeg        = m_pDoc->getLocOutputDetectorDirNeg();
613 
614         if(SimDlg.DoModal() == IDOK)
615         {
616                 m_pDoc->setFileIn(                      SimDlg.m_FileIn         );
617                 m_pDoc->setFileOut(                     SimDlg.m_FileOut        );
618                 m_pDoc->setMetricsDir(          SimDlg.m_MetricsDir     );
619                 m_pDoc->setFileType(            SimDlg.m_FileType       );
620                 m_pDoc->setRoadLength(          SimDlg.m_RoadLength     );
621                 m_pDoc->setDriveOnRight(        SimDlg.m_DriveOnRight );
622                 m_pDoc->setNoLanesDirPos(       SimDlg.m_NoLanesDirPos);
623                 m_pDoc->setNoLanesDirNeg(       SimDlg.m_NoLanesDirNeg);
624                 m_pDoc->setSimTimeStep(         (double)SimDlg.m_SimTimeStep / 1000 );  // mil
625 
626                 m_pDoc->setAllowLaneChanging(           SimDlg.m_AllowLaneChanging );
627                 m_pDoc->setTrafFileNoLanesDirPos(       SimDlg.m_TrafFileNoLanesDirPos );
628                 m_pDoc->setTrafFileNoLanesDirNeg(       SimDlg.m_TrafFileNoLanesDirNeg );
629                 m_pDoc->setLocOutputDetectorDirPos( SimDlg.m_LocOutputDetectorDirPos );
630                 m_pDoc->setLocOutputDetectorDirNeg( SimDlg.m_LocOutputDetectorDirNeg );
631 
632                 m_pDoc->UpdateDerivedMembers(true);     // call setMod flag
633 
634                 SettingUpdateRedraw();
635         };
```

```
636 }
```

Here is the call graph for this function:



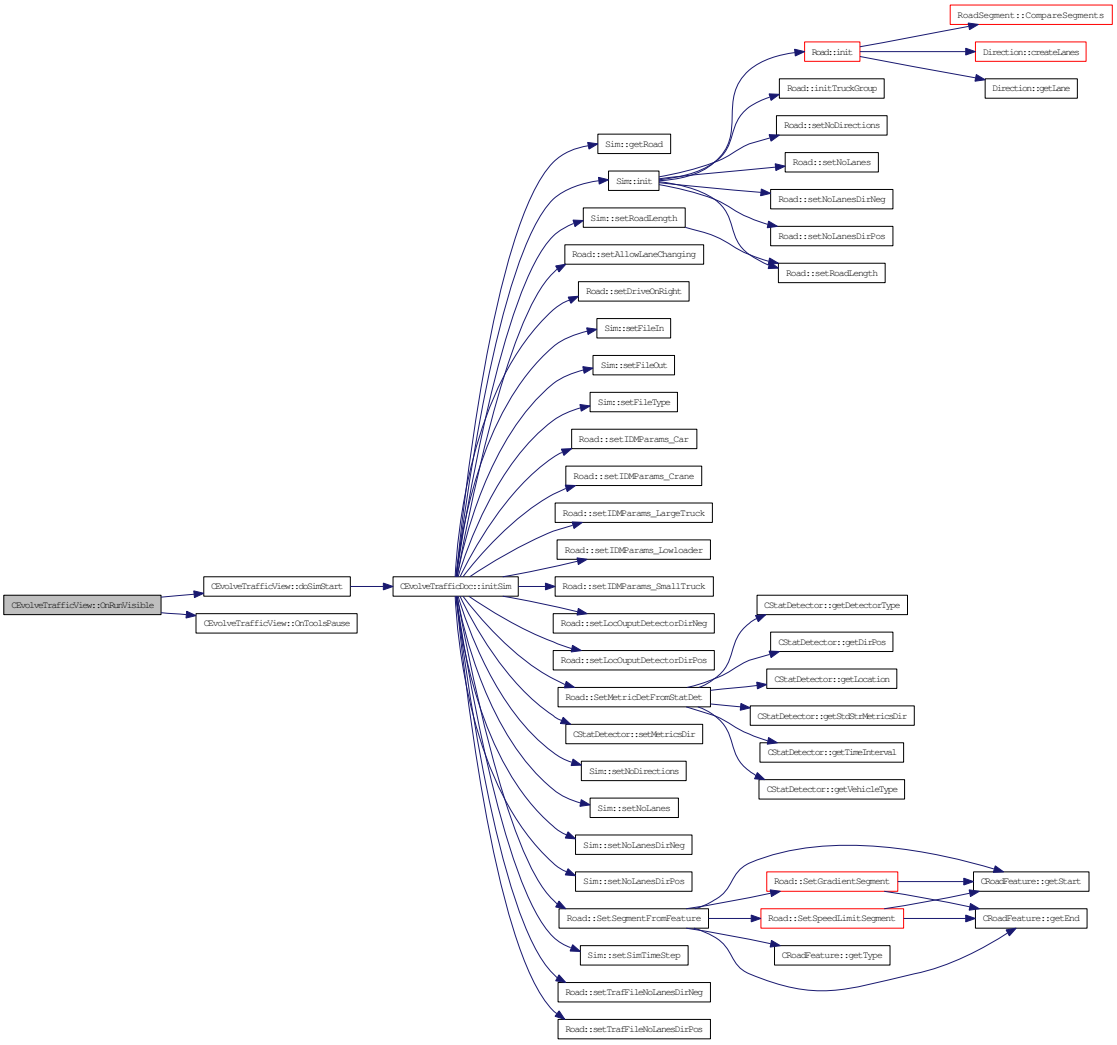
4.22.3.13 void CEvolveTrafficView::OnRunVisible () [protected]

Definition at line 354 of file EvolveTrafficView.cpp.

References doSimStart(), m_bInSimulation, m_bPause, and OnToolsPause().

```
355 {
356         if( !m_bInSimulation )
357                 doSimStart();
358         else
359         {
360                 if(m_bPause)            // give this button start stop behaviour as well
361                         OnToolsPause();
362                 else
363                         MessageBox("Simulation already running", "EvolveTraffic", MB_OK|MB_ICON
364         }
365 }
```

Here is the call graph for this function:



4.22.3.14 void CEvolveTrafficView::OnSize (UINT *nType*, int *cx*, int *cy*) [protected]

Definition at line 433 of file EvolveTrafficView.cpp.

```
434 {
435         CScrollView::OnSize(nType, cx, cy);
436 }
```

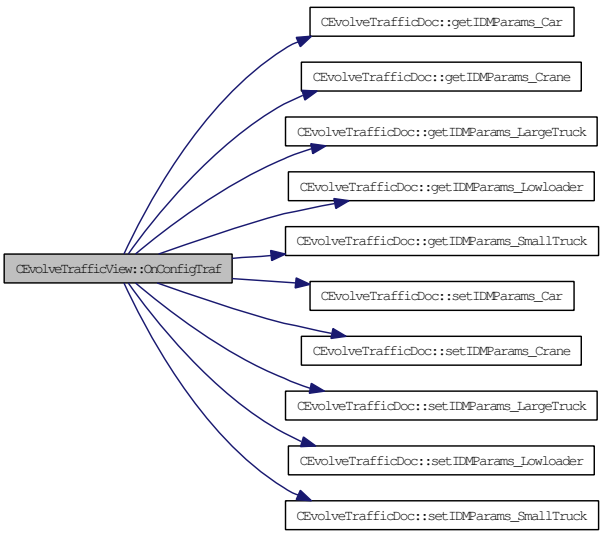
4.22.3.15 void CEvolveTrafficView::OnConfigTraf () [protected]

Definition at line 638 of file EvolveTrafficView.cpp.

References CEvolveTrafficDoc::getIDMParams_Car(), CEvolveTrafficDoc::getIDMParams_Crane(), CEvolveTrafficDoc::getIDMParams_LargeTruck(), CEvolveTrafficDoc::getIDMParams_Lowloader(), CEvolveTrafficDoc::getIDMParams_SmallTruck(), CTrafficConfigDlg::m_IDMParams_Car, CTrafficConfigDlg::m_IDMParams_Crane, CTrafficConfigDlg::m_IDMParams_LargeTruck, CTrafficConfigDlg::m_IDMParams_Lowloader, CTrafficConfigDlg::m_IDMParams_SmallTruck, m_pDoc, CEvolveTrafficDoc::setIDMParams_Car(), CEvolveTrafficDoc::setIDMParams_Crane(), CEvolveTrafficDoc::setIDMParams_LargeTruck(), CEvolveTrafficDoc::setIDMParams_Lowloader(), and CEvolveTrafficDoc::setIDMParams_SmallTruck().

```
639 {
640         CTrafficConfigDlg TrafDlg;
641
642         TrafDlg.m_IDMParams_Car                 = m_pDoc->getIDMParams_Car();
643         TrafDlg.m_IDMParams_SmallTruck  = m_pDoc->getIDMParams_SmallTruck();
644         TrafDlg.m_IDMParams_LargeTruck  = m_pDoc->getIDMParams_LargeTruck();
645         TrafDlg.m_IDMParams_Crane               = m_pDoc->getIDMParams_Crane();
646         TrafDlg.m_IDMParams_Lowloader   = m_pDoc->getIDMParams_Lowloader();
647
648         if(TrafDlg.DoModal() == IDOK)
649         {
650                 m_pDoc->setIDMParams_Car                (TrafDlg.m_IDMParams_Car);
651                 m_pDoc->setIDMParams_SmallTruck (TrafDlg.m_IDMParams_SmallTruck);
652                 m_pDoc->setIDMParams_LargeTruck (TrafDlg.m_IDMParams_LargeTruck);
653                 m_pDoc->setIDMParams_Crane              (TrafDlg.m_IDMParams_Crane);
654                 m_pDoc->setIDMParams_Lowloader  (TrafDlg.m_IDMParams_Lowloader);
655
656                 Invalidate();
657         }
658 }
```

Here is the call graph for this function:



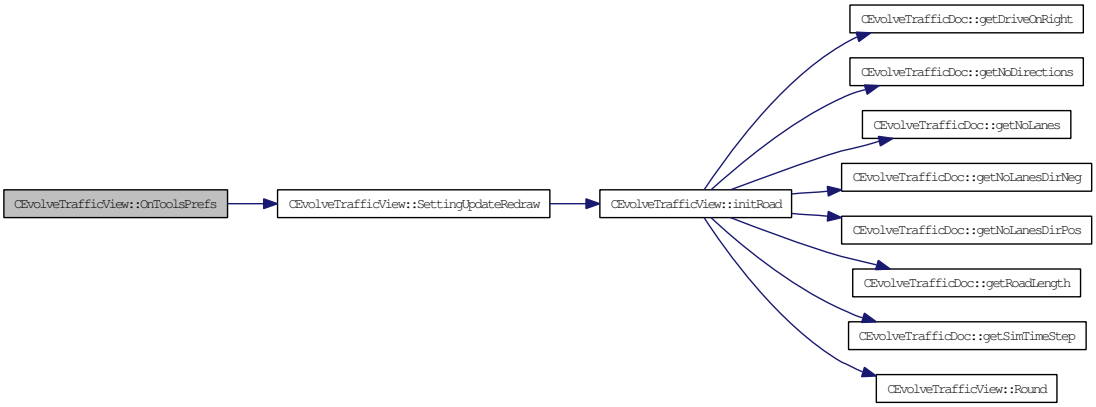
4.22.3.16 void CEvolveTrafficView::OnToolsPrefs () [protected]

Definition at line 660 of file EvolveTrafficView.cpp.

References m_Border_Btm, CPreferencesDlg::m_Border_Btm, m_Border_Lhs, CPreferencesDlg::m_Border_Lhs, m_Border_Rhs, CPreferencesDlg::m_Border_Rhs, m_Border_Top, CPreferencesDlg::m_Border_Top, m_bShowLegend, m_bShowVelocity, m_ExaggerateWidths, CPreferencesDlg::m_ExaggerateWidths, m_LaneWidth, CPreferencesDlg::m_LaneWidth, m_Scale, CPreferencesDlg::m_Scale, CPreferencesDlg::m_ShowLegend, CPreferencesDlg::m_ShowVelocity, m_TickLength, CPreferencesDlg::m_TickLength, m_TickStep, CPreferencesDlg::m_TickStep, m_TimeWarp, CPreferencesDlg::m_TimeWarp, m_VehicleLengthScale, CPreferencesDlg::m_VehicleLengthScale, m_VehicleWidth, CPreferencesDlg::m_VehicleWidth, and SettingUpdateRedraw().

```
661 {
662         CPreferencesDlg PrefsDlg;
663 
664         PrefsDlg.m_Scale                                = m_Scale;
665         PrefsDlg.m_ExaggerateWidths             = m_ExaggerateWidths;
666         PrefsDlg.m_TimeWarp                             = m_TimeWarp;
667         PrefsDlg.m_TickStep                             = m_TickStep;
668 
669         PrefsDlg.m_LaneWidth                    = m_LaneWidth   / m_ExaggerateWidths;
670         PrefsDlg.m_VehicleWidth                 = m_VehicleWidth/ m_ExaggerateWidths;
671         PrefsDlg.m_TickLength                   = m_TickLength  / m_ExaggerateWidths;
672 
673         PrefsDlg.m_VehicleLengthScale   = m_VehicleLengthScale;
674         PrefsDlg.m_ShowVelocity                 = m_bShowVelocity ? 1 : 0;
675         PrefsDlg.m_ShowLegend                   = m_bShowLegend ? 1 : 0;
676         PrefsDlg.m_Border_Top                   = m_Border_Top;
677         PrefsDlg.m_Border_Btm                   = m_Border_Btm;
678         PrefsDlg.m_Border_Lhs                   = m_Border_Lhs;
679         PrefsDlg.m_Border_Rhs                   = m_Border_Rhs;
680 
681         if(PrefsDlg.DoModal() == IDOK)
682         {
683                 m_Scale                                 =       PrefsDlg.m_Scale;
684                 m_ExaggerateWidths              =       PrefsDlg.m_ExaggerateWidths;
685                 m_TimeWarp                              =       PrefsDlg.m_TimeWarp;
686                 m_TickStep                              =       PrefsDlg.m_TickStep;
687 
688                 m_LaneWidth                             =       PrefsDlg.m_LaneWidth    * m_Exa
689                 m_VehicleWidth                  =       PrefsDlg.m_VehicleWidth * m_ExaggerateW
690                 m_TickLength                    =       PrefsDlg.m_TickLength   * m_ExaggerateW
691 
692                 m_VehicleLengthScale    =       PrefsDlg.m_VehicleLengthScale;
693                 m_bShowVelocity                 =       PrefsDlg.m_ShowVelocity == 0 ? false :
694                 m_bShowLegend                   =       PrefsDlg.m_ShowLegend == 0 ? false : tr
695                 m_Border_Top                    =       PrefsDlg.m_Border_Top;
696                 m_Border_Btm                    =       PrefsDlg.m_Border_Btm;
697                 m_Border_Lhs                    =       PrefsDlg.m_Border_Lhs;
698                 m_Border_Rhs                    =       PrefsDlg.m_Border_Rhs;
699 
700                 SettingUpdateRedraw();
701         }
702 }
```

Here is the call graph for this function:



4.22.3.17 void CEvolveTrafficView::OnToolsPause () [protected]

Definition at line 438 of file EvolveTrafficView.cpp.

References m_bPause, m_PauseTime, and m_StartRealTime.

Referenced by OnRunVisible().

```
439 {
440         if(m_bPause)    // currently paused - about to not be
441         {
442                 m_PauseTime = timeGetTime() - m_PauseTime;      // calc total pause time
443                 m_StartRealTime += m_PauseTime;                                 // so time just
444         }
445         else                    // not paused - but about to be
446                 m_PauseTime = timeGetTime();                            // save initial pause t
447
448         m_bPause = !m_bPause;           // so pause/start behaviour
449
450 }
```

4.22.3.18 void CEvolveTrafficView::OnToolsZoomin () [protected]

Definition at line 517 of file EvolveTrafficView.cpp.

References m_Scale, and SetZoomScale().

```
518 {
519         SetZoomScale(m_Scale + 0.2);    // MAGIC NUMBER - should it be an option?
520 }
```

Here is the call graph for this function:

4.22.3.19 void CEvolveTrafficView::OnToolsZoomout () [protected]

Definition at line 522 of file EvolveTrafficView.cpp.

References m_Scale, and SetZoomScale().

```
523 {
524         SetZoomScale(m_Scale - 0.2);    // MAGIC NUMBER - should it be an option?
525 }
```

Here is the call graph for this function:

4.22.3.20 void CEvolveTrafficView::OnToolsSpeedup () [protected]

Definition at line 527 of file EvolveTrafficView.cpp.

References m_TimeWarp, and ValidateTimeWarp().

```
528 {
529         m_TimeWarp += 1.0;                                  // MAGIC NUMBER - should it be an optio
530
531         ValidateTimeWarp();
532 }
```

Here is the call graph for this function:

4.22.3.21 void CEvolveTrafficView::OnToolsSlowdown () [protected]

Definition at line 534 of file EvolveTrafficView.cpp.

References m_TimeWarp, and ValidateTimeWarp().

```
535 {
536         m_TimeWarp -= 1.0;                                  // MAGIC NUMBER - should it be an optio
537
538         ValidateTimeWarp();
539 }
```

Here is the call graph for this function:

4.22.3.22 void CEvolveTrafficView::OnKeyDown (UINT *nChar*, UINT *nRepCnt*, UINT *nFlags*) [protected]

Definition at line 541 of file EvolveTrafficView.cpp.

References m_RoadLength, and Round().

```
542 {
543         CScrollView::OnKeyDown(nChar, nRepCnt, nFlags);
544
545         POINT curPoint = GetScrollPosition();
546         int deltaX = Round(m_RoadLength/10);    // MAGIC NUMBER - scroll 1/10 of road
547
548         switch(nChar)
549         {
550                 case VK_RIGHT:
551                         curPoint.x += deltaX;
552                         break;
553                 case VK_LEFT:
554                         curPoint.x -= deltaX;
555                         break;
556                 case VK_HOME:
557                         curPoint.x = 0;
558                         break;
559                 case VK_END:
560                         CSize temp = GetTotalSize();
561                         curPoint.x = temp.cx;
562                         break;
563         }
564         ScrollToPosition(curPoint);
565         Invalidate(FALSE);
566 }
```

Here is the call graph for this function:

4.22.3.23 void CEvolveTrafficView::OnRunInvisible () [protected]

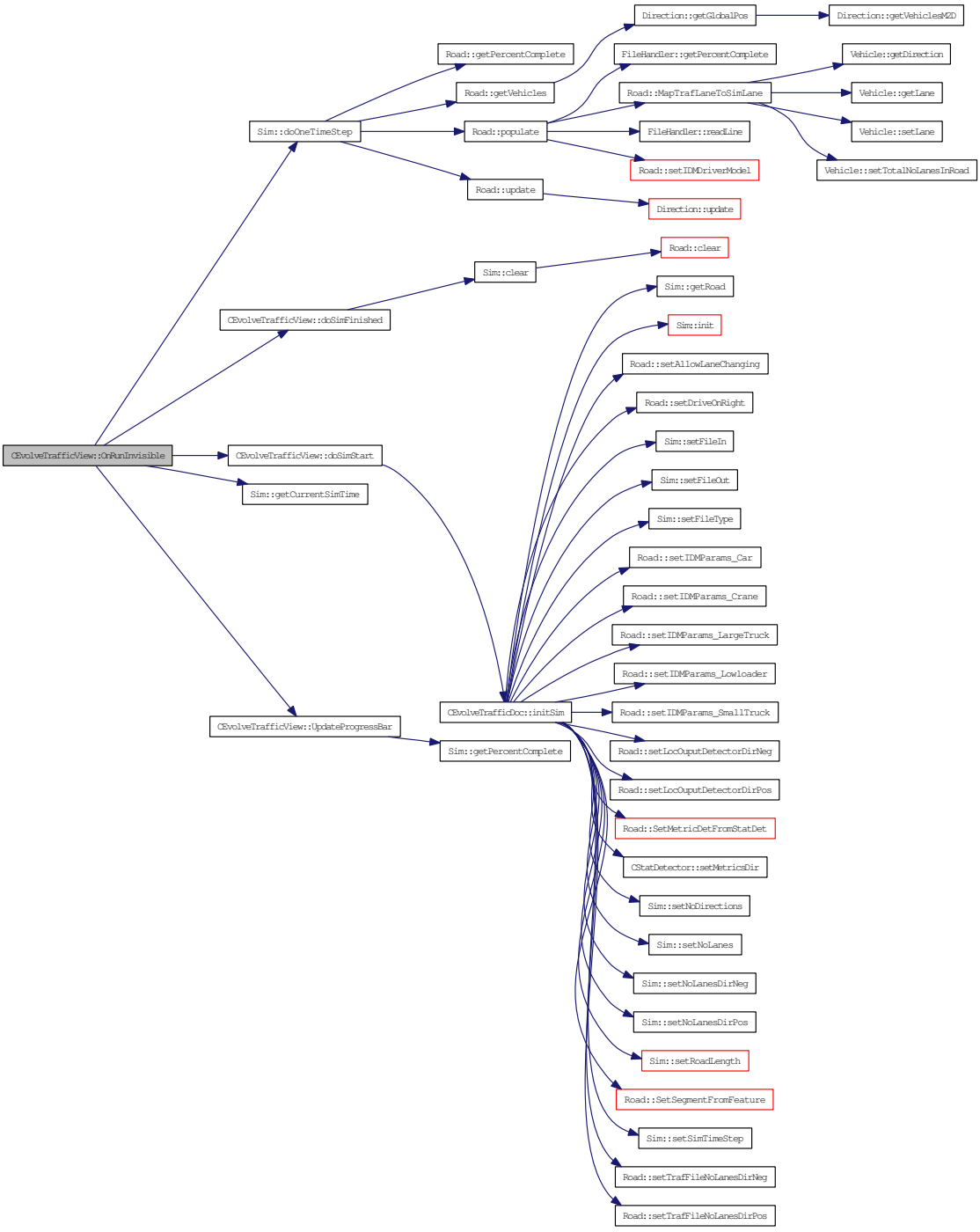
Definition at line 367 of file EvolveTrafficView.cpp.

References Sim::doOneTimeStep(), doSimFinished(), doSimStart(), Sim::getCurrentSimTime(), m_bInSimulation, m_CurentRealTime, m_CurentSimTime, m_pDoc, CEvolveTrafficDoc::m_Sim, m_StartRealTime, and UpdateProgressBar().

```
368 {
369         if( !m_bInSimulation )
370         {
371                 int result = MessageBox("Running EvolveTraffic in this mode means:\n"
372                                                                 "- only the progress bar is upd
373                                                                 "- the program will appear to h
374                                                                 "Are you sure that you want to
375                                                                 "EvolveTraffic", MB_YESNO|MB_IC
376
```

```
377                 if(result == IDYES)     // we're cleared to go
378                 {
379                         doSimStart();
380                         bool bInSim = true;             // local inSim - we want class-wide InS
381                         while(bInSim)
382                         {
383                                 m_pDoc->m_Sim.doOneTimeStep(&bInSim);   // bInSim becomes false
384                                 UpdateProgressBar();
385                         }
386                         m_CurentSimTime = m_pDoc->m_Sim.getCurrentSimTime();
387                         m_CurentRealTime = timeGetTime() - m_StartRealTime;     // no need for
388                         doSimFinished(true);    // no need to check if it's finished - true mea
389                 }
390         }
391         else
392                 MessageBox("Simulation already running", "EvolveTraffic", MB_OK|MB_ICONWARNING)
393
394 }
```

Here is the call graph for this function:



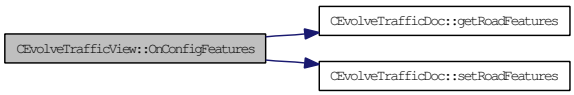
4.22.3.24 void CEvolveTrafficView::OnConfigFeatures () [protected]

Definition at line 704 of file EvolveTrafficView.cpp.

References CEvolveTrafficDoc::getRoadFeatures(), m_pDoc, m_RoadLength, CRoadFeaturesDlg::m_RoadLength, CRoadFeaturesDlg::m_vRoadFeatures, and CEvolveTrafficDoc::setRoadFeatures().

```
705 {
706         CRoadFeaturesDlg FeaturesDlg;
707
708         FeaturesDlg.m_RoadLength = m_RoadLength;        // for validation purposes only
709         FeaturesDlg.m_vRoadFeatures.Copy( *(m_pDoc->getRoadFeatures()) );
710
711         if( FeaturesDlg.DoModal() == IDOK )
712                 m_pDoc->setRoadFeatures( &(FeaturesDlg.m_vRoadFeatures) );
713
714         Invalidate();
715 }
```

Here is the call graph for this function:



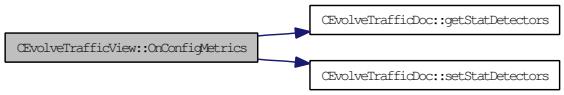
4.22.3.25 void CEvolveTrafficView::OnConfigMetrics () [protected]

Definition at line 717 of file EvolveTrafficView.cpp.

References CEvolveTrafficDoc::getStatDetectors(), m_pDoc, m_RoadLength, CStatDetectorDlg::m_RoadLength, CStatDetectorDlg::m_vStatDetectors, and CEvolveTrafficDoc::setStatDetectors().

```
718 {
719         CStatDetectorDlg StatDetDlg;
720
721         StatDetDlg.m_RoadLength = m_RoadLength; // for validation purposes only
722         StatDetDlg.m_vStatDetectors.Copy( *(m_pDoc->getStatDetectors()) );
723
724         if( StatDetDlg.DoModal() == IDOK )
725                 m_pDoc->setStatDetectors( &(StatDetDlg.m_vStatDetectors) );
726
727         Invalidate();
728 }
```

Here is the call graph for this function:



4.22.3.26 void CEvolveTrafficView::OnUpdateToolsPause (CCmdUI * *pCmdUI*) [protected]

Definition at line 465 of file EvolveTrafficView.cpp.

References m_bInSimulation, and m_bPause.

```
466 {
467         if(m_bInSimulation)
468         {
469                 pCmdUI->Enable(true);
470                 if(m_bPause)
471                         pCmdUI->SetText("&Resume\tF9");
472                 else
473                         pCmdUI->SetText("&Pause\tF9");
474         }
475         else
476                 pCmdUI->Enable(false);
477
478 }
```

4.22.3.27 void CEvolveTrafficView::OnUpdateToolsSpeedup (CCmdUI * *pCmdUI*) [protected]

Definition at line 480 of file EvolveTrafficView.cpp.

References m_bInSimulation.

```
481 {
482         if(m_bInSimulation)
483                 pCmdUI->Enable(true);
484         else
485                 pCmdUI->Enable(false);
486 }
```

4.22.3.28 void CEvolveTrafficView::OnUpdateToolsSlowdown (CCmdUI * *pCmdUI*) [protected]

Definition at line 488 of file EvolveTrafficView.cpp.

References m_bInSimulation.

```
489 {
490         if(m_bInSimulation)
491                 pCmdUI->Enable(true);
492         else
493                 pCmdUI->Enable(false);
494 }
```

4.22.3.29 void CEvolveTrafficView::OnToolsStop () [protected]

Definition at line 452 of file EvolveTrafficView.cpp.

References doSimFinished().

```
453 {
454         doSimFinished(false);   // false means not compelted, but stopped midway through
455 }
```

Here is the call graph for this function:

4.22.3.30 void CEvolveTrafficView::OnUpdateToolsStop (CCmdUI * *pCmdUI*) [protected]

Definition at line 457 of file EvolveTrafficView.cpp.

References m_bInSimulation.

```
458 {
459         if(m_bInSimulation)
460                 pCmdUI->Enable(true);
461         else
462                 pCmdUI->Enable(false);
463 }
```

4.22.3.31 void CEvolveTrafficView::OnLButtonDown (UINT *nFlags*, CPoint *point*) [protected]

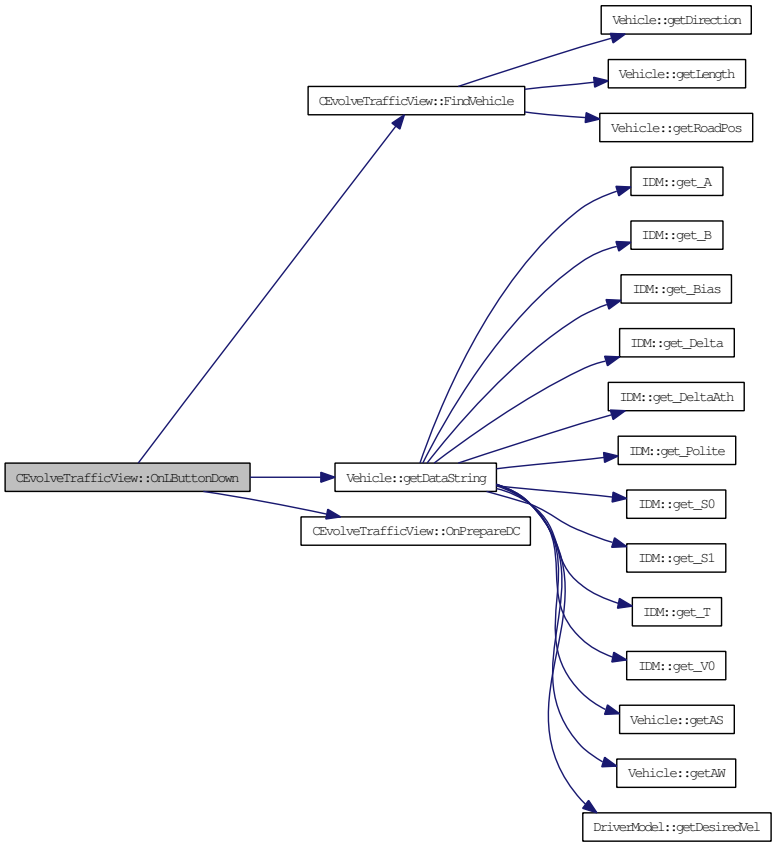
Definition at line 734 of file EvolveTrafficView.cpp.

References FindVehicle(), Vehicle::getDataString(), m_Border_Lhs, m_bPause, m_-BtmEdge, m_DataTip, m_DriveOnRight, m_LaneWidth, m_NoLanes, m_TopEdge, and OnPrepareDC().

```
735 {
736         if(m_bPause)    // only if paused
737         {
738                 // Get road coords
739                 CClientDC aDC(this);
740                 OnPrepareDC(&aDC);
741                 CPoint ptRoad = point;
742                 aDC.DPtoLP(&ptRoad);
743                 ptRoad.x -= m_Border_Lhs;
744
745                 if(ptRoad.y < m_BtmEdge && ptRoad.y > m_TopEdge) // only if within road
746                 {
747                         // Get lane
748                         int iLane = 1;
749                         int TopOfLane = m_BtmEdge;
750
751                         while(ptRoad.y < TopOfLane)
752                         {
753                                 TopOfLane = m_BtmEdge - iLane*m_LaneWidth;
754                                 iLane++;
755                         }
756                         // so it was in the previous lane, but zero based array, i.e.
```

```
757                         iLane -= 2;
758                         if(!m_DriveOnRight)
759                                 iLane = m_NoLanes - iLane -1;
760
761                         // Find vehicle at pos x in lane iLane
762                         Vehicle* pVeh = FindVehicle(iLane, ptRoad.x);
763
764                         if(pVeh != NULL)        // assemble datastring and display
765                         {
766                                 CRect rect(point.x,point.y,point.x+100, point.y+100);   // not
767                                 m_DataTip.Show(rect,pVeh->getDataString());
768                         }
769                 }
770         }
771
772         CScrollView::OnLButtonDown(nFlags, point);
773 }
```

Here is the call graph for this function:



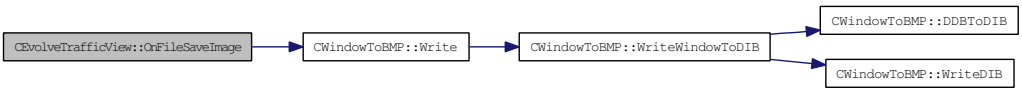
4.22.3.32 void CEvolveTrafficView::OnFileSaveImage () [protected]

Definition at line 572 of file EvolveTrafficView.cpp.

References CWindowToBMP::Write().

```
573 {
574         CFileDialog FileDlg(
575                 FALSE,                  // TRUE if File Open, FALSE if File Save As
576                 _T("*.bmp"),    // the default file extension
577                 NULL,                   // the initial filename that appears
578                 OFN_PATHMUSTEXIST | OFN_HIDEREADONLY | OFN_OVERWRITEPROMPT,//flags for customiz
579                 "Bitmap files (*.bmp) |*.bmp||",        // file filters
580                 this); // pointer to FileCialog's parent object
581
582         if(FileDlg.DoModal() == IDOK)
583         {
584                 CString file = FileDlg.GetPathName();
585                 UpdateWindow();         // to get rid of dialog
586                 CWindowToBMP convert;
587                 if( convert.Write(file, this) )
588                         MessageBox("Image saved.","EvolveTraffic", MB_OK|MB_ICONINFORMATION);
589                 else
590                         MessageBox("Image saving failed.","EvolveTraffic", MB_OK|MB_ICONWARNIN
591         }
592 }
```

Here is the call graph for this function:



4.22.3.33 Vehicle * CEvolveTrafficView::FindVehicle (int *iLane*, int *location*) [private]

Definition at line 775 of file EvolveTrafficView.cpp.

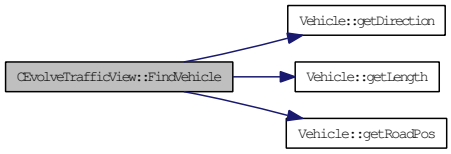
References Vehicle::getDirection(), Vehicle::getLength(), Vehicle::getRoadPos(), m_-VehicleLengthScale, and m_VehiclePositions.

Referenced by OnLButtonDown().

```
776 {
777         Vehicle* pVeh = NULL;
778         std::vector<Vehicle*> vLaneVehs = m_VehiclePositions.at(iLane);
779
780         for(int i = 0; i < vLaneVehs.size(); i++)
781         {
782                 pVeh = vLaneVehs.at(i);
783                 int front = pVeh->getRoadPos();
784                 int length = pVeh->getLength() * m_VehicleLengthScale;
785
786                 bool bOnVehicle = false;
787                 if(pVeh->getDirection())
788                 {
789                         if( location < front && location > front - length )
790                                 bOnVehicle = true;
791                 }
792                 else
```

```
793                 {
794                         if( location > front && location < front + length )
795                                 bOnVehicle = true;
796                 }
797
798                 if( bOnVehicle )
799                         return pVeh;
800         }
801
802         return NULL;
803 }
```

Here is the call graph for this function:



4.22.3.34 BOOL CEvolveTrafficView::WriteWindowToDIB (LPTSTR *szFile*, CWnd * *pWnd*) [private]

4.22.3.35 void CEvolveTrafficView::SetZoomScale (double *newScale*) [private]

Definition at line 181 of file EvolveTrafficView.cpp.

References m_ClientRect, m_DrawArea, m_Scale, MIN_VIEW_SCALE, and Round().

Referenced by OnToolsZoomin(), and OnToolsZoomout().

```
182 {
183         if(newScale > MIN_VIEW_SCALE)
184         {
185                 double oldZoomFactor = m_Scale;
186
187                 CPoint centerScrollPosition = GetScrollPosition();
188                 CRect ClientRect(0,0,0,0);
189                 centerScrollPosition.x += m_ClientRect.right/2;
190                 centerScrollPosition.y += m_ClientRect.bottom/2;
191
192                 m_Scale = newScale;
193
194                 CSize displaySize;
195                 displaySize.cx = Round(m_DrawArea.cx * m_Scale);
196                 displaySize.cy = Round(m_DrawArea.cy * m_Scale);
197                 SetScrollSizes(MM_TEXT,displaySize);
198
199                 int newXScrollPosition = Round((centerScrollPosition.x / oldZoomFactor) * m_Sca
200                 int newYScrollPosition = Round((centerScrollPosition.y / oldZoomFactor) * m_Sca
201
202                 newXScrollPosition -= m_ClientRect.right/2;
```

```
203                 newYScrollPosition -= m_ClientRect.bottom/2;
204                 ScrollToPosition(CPoint(newXScrollPosition,newYScrollPosition));
205         }
206         else
207         {
208                 CString str; str.Format("You cannot zoom out any further - minimum view scale i
209                 MessageBox(str, "EvolveTraffic", MB_OK|MB_ICONWARNING);
210         }
211
212         Invalidate(FALSE);
213 }
```

Here is the call graph for this function:

4.22.3.36 void CEvolveTrafficView::ValidateTimeWarp () [private]

Definition at line 502 of file EvolveTrafficView.cpp.

References m_MaxTimeWarp, and m_TimeWarp.

Referenced by OnToolsSlowdown(), and OnToolsSpeedup().

```
503 {
504         if(m_TimeWarp > m_MaxTimeWarp)
505         {
506                 m_TimeWarp = m_MaxTimeWarp;
507                 CString str;    str.Format("Maximum time warp reached (%.2f) for this simulatio
508                 MessageBox(str,"EvolveTraffic",MB_OK|MB_ICONINFORMATION);
509         }
510         if(m_TimeWarp < 0.5)    // set half speed as minimum
511         {
512                 m_TimeWarp = 1.0;
513                 MessageBox("Half of real-time speed is the minimum time warp.","EvolveTraffic",
514         }
515 }
```

4.22.3.37 void CEvolveTrafficView::doSimStart () [private]

Definition at line 396 of file EvolveTrafficView.cpp.

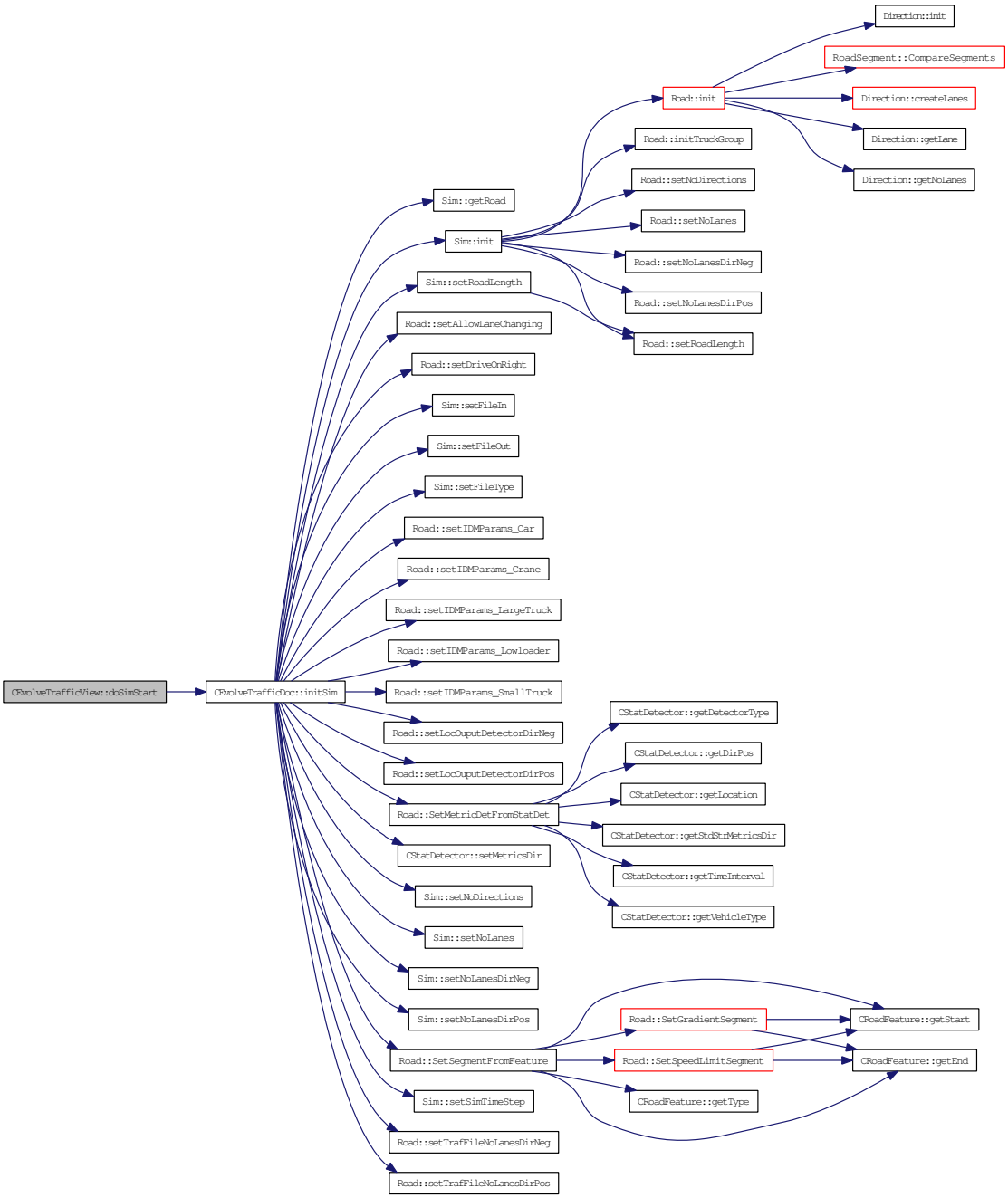
References CEvolveTrafficDoc::initSim(), m_bInSimulation, m_CurentSimTime, m_-pDoc, m_pProgBar, and m_StartRealTime.

Referenced by OnRunInvisible(), and OnRunVisible().

```
397 {
398         bool statusOK = m_pDoc->initSim();
399         if(statusOK)
400         {
401                 ((CEvolveTrafficApp*)AfxGetApp())->setInSimulation(true);
402                 m_StartRealTime = timeGetTime();
403                 m_bInSimulation = true;                         // this means the doLoop() beco
```

```
404                 m_CurentSimTime = 0.0;
405                 m_pProgBar = new CProgressBar;
406                 m_pProgBar->Create("Percent Complete 0%");
407                 Invalidate();   // refresh screen before starting
408         }
409 }
```

Here is the call graph for this function:



4.22.3.38 int CEvolveTrafficView::Round (double *val*) [inline, private]

Definition at line 103 of file EvolveTrafficView.h.

Referenced by doLoop(), DrawLegend(), DrawRoadSegments(), DrawTimer(), initRoad(), OnInitialUpdate(), OnKeyDown(), and SetZoomScale().

```
103 {return int(val + 0.5);};
```

4.22.3.39 void CEvolveTrafficView::doSimFinished (bool *bCompleted*) [private]

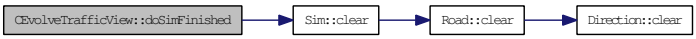
Definition at line 411 of file EvolveTrafficView.cpp.

References Sim::clear(), m_bInSimulation, m_LastPercentComplete, m_pDoc, m_pProgBar, CEvolveTrafficDoc::m_Sim, and m_VehiclePositions.

Referenced by doLoop(), OnRunInvisible(), and OnToolsStop().

```
412 {
413         Invalidate();
414         m_bInSimulation = false;
415         ((CEvolveTrafficApp*)AfxGetApp())->setInSimulation(false);      // tell the sim loop to
416         m_pProgBar->Clear();
417         delete m_pProgBar;
418         m_pDoc->m_Sim.clear();
419         m_LastPercentComplete = 0;
420         m_VehiclePositions.clear();
421
422         if(bCompleted)
423                 MessageBox("The simulation has finished.","EvolveTraffic",MB_OK|MB_ICONINFORMA
424 }
```

Here is the call graph for this function:



4.22.3.40 void CEvolveTrafficView::UpdateDrawRegion () [private]

4.22.3.41 void CEvolveTrafficView::SettingUpdateRedraw () [private]

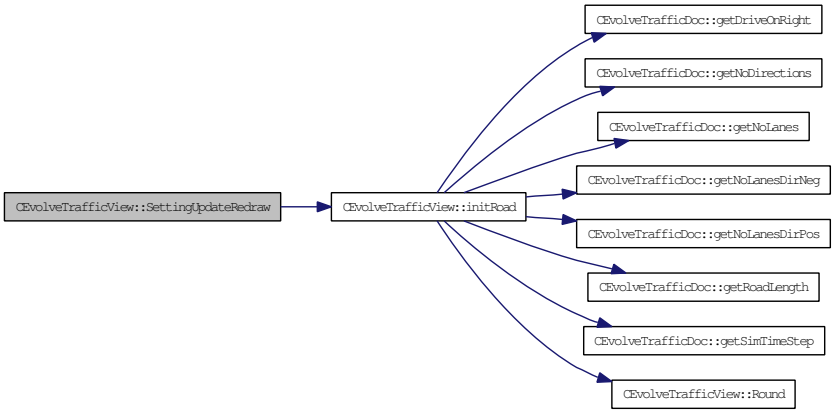
Definition at line 496 of file EvolveTrafficView.cpp.

References initRoad().

Referenced by OnConfigSim(), and OnToolsPrefs().

```
497 {
498         initRoad();             // re-do the road properties
499         Invalidate();   // draw it
500 }
```

Here is the call graph for this function:



4.22.3.42 void CEvolveTrafficView::initRoad () [private]

Definition at line 215 of file EvolveTrafficView.cpp.

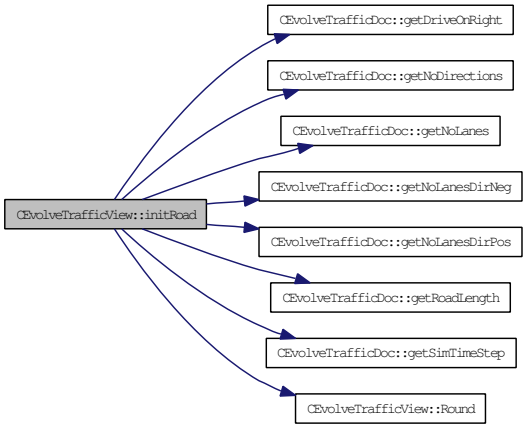
References CEvolveTrafficDoc::getDriveOnRight(), CEvolveTrafficDoc::getNoDirections(), CEvolveTrafficDoc::getNoLanes(), CEvolveTrafficDoc::getNoLanesDirNeg(), CEvolveTrafficDoc::getNoLanesDirPos(), CEvolveTrafficDoc::getRoadLength(), CEvolveTrafficDoc::getSimTimeStep(), m_Border_Btm, m_Border_Lhs, m_Border_Rhs, m_Border_Top, m_BtmEdge, m_DrawArea, m_DriveOnRight, m_LaneWidth, m_nLanesOnBtm, m_nLanesOnTop, m_NoDirections, m_NoLanes, m_NoLanesDirNeg, m_NoLanesDirPos, m_NoTicks, m_pDoc, m_RoadLength, m_RoadWidth, m_Scale, m_SimTimeStep, m_TickStep, m_TopEdge, and Round().

Referenced by OnInitialUpdate(), and SettingUpdateRedraw().

```
216 {
217         m_DriveOnRight  = m_pDoc->getDriveOnRight();
218         m_RoadLength    = m_pDoc->getRoadLength();
219         m_NoLanes               = m_pDoc->getNoLanes();
220         m_NoLanesDirPos = m_pDoc->getNoLanesDirPos();
221         m_NoLanesDirNeg = m_pDoc->getNoLanesDirNeg();
222         m_NoDirections  = m_pDoc->getNoDirections();
223         m_RoadWidth             = m_NoLanes * m_LaneWidth;
224         m_NoTicks               = (int)(m_RoadLength/m_TickStep)+1;
225
226         m_nLanesOnTop   = m_DriveOnRight ? m_NoLanesDirNeg : m_NoLanesDirPos;
227         m_nLanesOnBtm   = m_DriveOnRight ? m_NoLanesDirPos : m_NoLanesDirNeg;
228         m_TopEdge               = -m_nLanesOnTop*m_LaneWidth;
229         m_BtmEdge               = m_nLanesOnBtm*m_LaneWidth;
230
231         m_SimTimeStep = Round(1000 * m_pDoc->getSimTimeStep()); // constant sim time step
232
233         m_DrawArea.cy =  m_Border_Top + m_Border_Btm + m_RoadWidth;
234         m_DrawArea.cx =  m_Border_Lhs + m_Border_Rhs + m_RoadLength;
235
236         CSize scrolls;
```

```
237         scrolls.cx = m_DrawArea.cx*m_Scale;     // when scale is not 1.0
238         scrolls.cy = m_DrawArea.cy*m_Scale;
239         CScrollView::SetScrollSizes(MM_TEXT, scrolls);
240 }
```

Here is the call graph for this function:



4.22.3.43 void CEvolveTrafficView::DrawRoad (CMemDC * *pDC*) [private]

Definition at line 809 of file EvolveTrafficView.cpp.

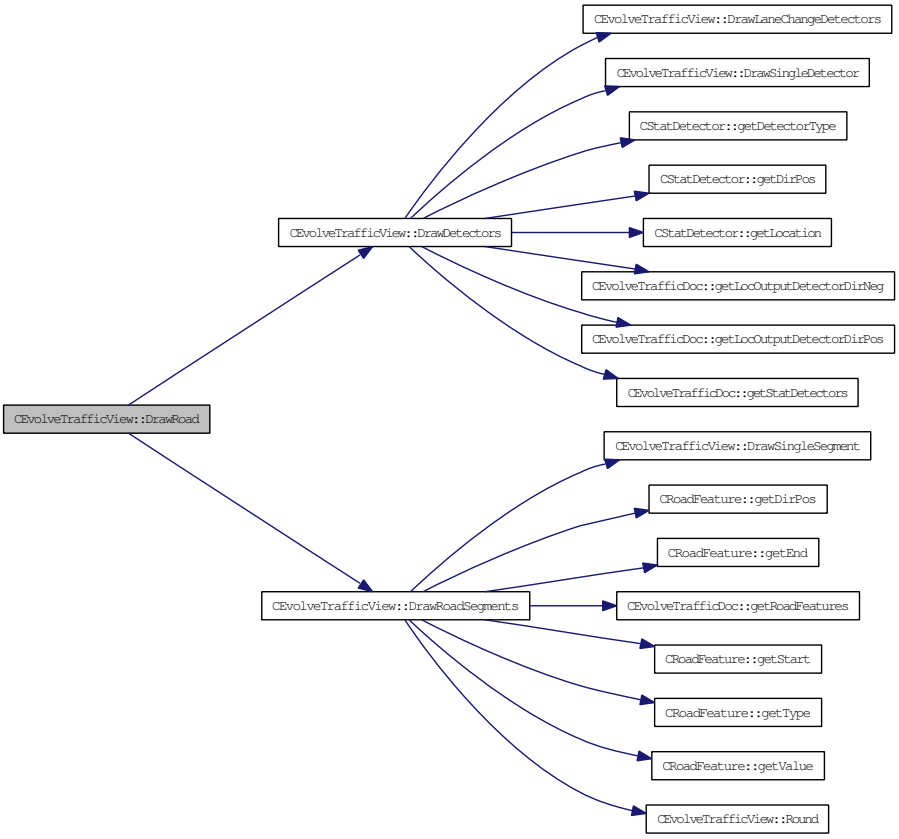
References CLR_ROAD_LINES, CLR_ROAD_SURFACE, DrawDetectors(), DrawRoadSegments(), m_Border_Lhs, m_BtmEdge, m_LaneWidth, m_NoDirections, m_NoLanes, m_RoadLength, m_RoadWidth, and m_TopEdge.

Referenced by OnDraw().

```
810 {
811         // Prepare brushes
812         CPen BlackPen(PS_SOLID, 0, CLR_ROAD_LINES);
813         CPen BlackDashPen(PS_DASH, 0, CLR_ROAD_LINES);
814 
815         // fill the road surface
816         pDC->FillSolidRect(m_Border_Lhs,m_TopEdge,m_RoadLength,m_RoadWidth, CLR_ROAD_SURFACE);
817         // draw road segements
818         DrawRoadSegments(pDC);
819 
820         pDC->SelectObject(&BlackPen);
821         // Top edge of the road
822         pDC->MoveTo(m_Border_Lhs,m_TopEdge);
823         pDC->LineTo(m_Border_Lhs+m_RoadLength,m_TopEdge);
824         // bottom edge of the road
825         pDC->MoveTo(m_Border_Lhs,m_BtmEdge);
826         pDC->LineTo(m_Border_Lhs+m_RoadLength,m_BtmEdge);
827         // for each lane
828         pDC->SelectObject(&BlackDashPen);
829         int LaneEdge =  m_BtmEdge;
830         for (int i = 0; i < m_NoLanes-1; i++)
```

```
831         {       // from bottom working the way up
832                 LaneEdge -= m_LaneWidth;
833                 pDC->MoveTo(m_Border_Lhs,LaneEdge);
834                 pDC->LineTo(m_Border_Lhs+m_RoadLength,LaneEdge);
835         }
836         // Divide carraigeway
837         pDC->SelectObject(&BlackPen);
838         if(m_NoDirections == 2)
839         {
840                 pDC->MoveTo(m_Border_Lhs,0);
841                 pDC->LineTo(m_Border_Lhs+m_RoadLength,0);
842         }
843
844         DrawDetectors(pDC);
845 }
```

Here is the call graph for this function:



4.22.3.44 void CEvolveTrafficView::DrawRuler (CMemDC * *pDC*) [private]

Definition at line 1175 of file EvolveTrafficView.cpp.

References CLR_BACKGRND, CLR_RULER_LINES, CLR_RULER_TEXT, m_Border_Lhs, m_BtmEdge, m_NoTicks, m_TickLength, and m_TickStep.

Referenced by OnDraw().

```
1176 {
1177         int ypos = m_BtmEdge;
1178
1179         //int ypos =    m_YCoordTop; //m_ClientRect.Height()/2 + m_RoadWidth/2;
1180         CPen RulerLinesPen(PS_SOLID, 0, CLR_RULER_LINES);
1181         pDC->SelectObject(&RulerLinesPen);
1182         pDC->SetTextColor(CLR_RULER_TEXT);
1183         pDC->SetBkColor(CLR_BACKGRND);
1184
1185         for(int i = 0; i < m_NoTicks; i++)
1186         {
1187                 int xpos = m_Border_Lhs+i*m_TickStep;
1188                 pDC->MoveTo(xpos,ypos);
1189                 pDC->LineTo(xpos,ypos+m_TickLength);
1190                 CString str; str.Format("%d m",i*m_TickStep);
1191                 pDC->TextOut(xpos+5,ypos+10,str);
1192         }
1193 }
```

4.22.3.45 void CEvolveTrafficView::DrawTimer (CMemDC * *pDC*) [private]

Definition at line 902 of file EvolveTrafficView.cpp.

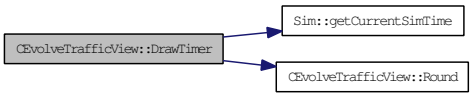
References CLR_BACKGRND, CLR_LEGEND_TEXT, Sim::getCurrentSimTime(), HOURS_PER_DAY, m_bInSimulation, m_Border_Lhs, m_Border_Top, m_bPause, m_ClientRect, m_CurentRealTime, m_CurentSimTime, m_pDoc, m_Scale, CEvolveTrafficDoc::m_Sim, m_StartRealTime, MINS_PER_HOUR, Round(), SECS_PER_HOUR, and SECS_PER_MIN.

Referenced by OnDraw().

```
903 {
904         pDC->SetTextColor(CLR_LEGEND_TEXT);
905         pDC->SetBkColor(CLR_BACKGRND);
906
907         // Draw the simulation time
908         int ypos = Round(-(m_ClientRect.Height()/2)/m_Scale)+m_Border_Top;      // this keeps t
909         int xpos = m_Border_Lhs;
910
911         if(m_bInSimulation)
912                 m_CurentSimTime = m_pDoc->m_Sim.getCurrentSimTime();
913
914         double temp = m_CurentSimTime;                                                  int day
915         temp = temp - day * HOURS_PER_DAY * SECS_PER_HOUR;              int hr  = (int)(temp/(S
916         temp = temp - hr * SECS_PER_HOUR;                                               int min
917         temp = temp - min * SECS_PER_MIN;                                               double
918
919         CString SimStr; SimStr.Format("Sim. Time - Day: %d - %d:%02d:%05.2f",day,hr,min,sec);
920         CSize strSimSize = pDC->GetTextExtent(SimStr);  // so text is always inside the window
921         pDC->TextOut(xpos,ypos,SimStr); // change + to - if drawn on bottom
922
923         // draw the real elapsed time
```

```
924         if(m_bInSimulation && !m_bPause)        // only update if we're simulating and not paus
925                 m_CurentRealTime = timeGetTime() - m_StartRealTime;
926
927         temp = (double)m_CurentRealTime/1000;                                   day     = (int)
928         temp = temp - day * HOURS_PER_DAY * SECS_PER_HOUR;              hr      = (int)(temp/(S
929         temp = temp - hr * SECS_PER_HOUR;                                       min
930         temp = temp - min * SECS_PER_MIN;                                       sec
931
932         CString RealStr; RealStr.Format("Real Time - Day: %d - %d:%02d:%05.2f",day,hr,min,sec);
933         CSize strRealSize = pDC->GetTextExtent(RealStr);        // so text is always inside the
934         pDC->TextOut(xpos,ypos + strRealSize.cy*1.2 ,RealStr);  // 20% sapce between strings |
935
936 }
```

Here is the call graph for this function:



4.22.3.46 void CEvolveTrafficView::DrawVehicle (CMemDC * *pDC*, Vehicle * *veh*) [private]

Definition at line 847 of file EvolveTrafficView.cpp.

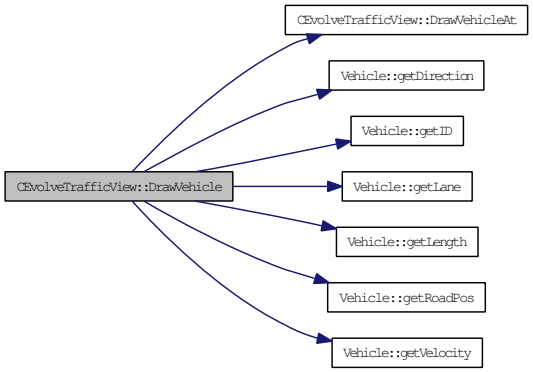
References DrawVehicleAt(), Vehicle::getDirection(), Vehicle::getID(), Vehicle::getLane(), Vehicle::getLength(), Vehicle::getRoadPos(), Vehicle::getVelocity(), m_Border_Lhs, m_bShowVelocity, m_BtmEdge, m_DriveOnRight, m_LaneWidth, M_PER_S_TO_KM_PER_H, m_TopEdge, m_VehicleLengthScale, and m_VehicleWidth.

Referenced by OnDraw().

```
848 {
849         bool DirPointsRight = pVeh->getDirection();
850         int lane = pVeh->getLane();
851         int x = pVeh->getRoadPos() + m_Border_Lhs;      // front of vehicle road position to sc
852         int y = 0;                                                                      // just
853
854         y = m_DriveOnRight ? m_BtmEdge - (lane-0.5)*m_LaneWidth : m_TopEdge + (lane-0.5)*m_Lane
855         // and subtract half the vehicle width to get the y pos of the top edge
856         y = y - m_VehicleWidth/2;
857
858         // So DirNeg vehicles' position corresponds with the front of the vehicle
859         int DirectionFactor = DirPointsRight == true ? -1 : +1;
860         int cx = DirectionFactor * m_VehicleLengthScale * pVeh->getLength();
861         CRect VehRect(x,y,x+cx,y+m_VehicleWidth);
862
863         DrawVehicleAt(pDC, pVeh->getID(), VehRect);
864
865         // draw the velocity on the vehicle
866         double velocity = pVeh->getVelocity()*M_PER_S_TO_KM_PER_H;
867
868         if(m_bShowVelocity)
869         {
```

```
870                 CString SpeedStr; SpeedStr.Format("%3.1f", velocity);
871                 CSize strSize = pDC->GetTextExtent(SpeedStr);
872                 x = x + cx/2 - strSize.cx/2;
873                 y = y + m_VehicleWidth/2 - strSize.cy/2;
874                 pDC->TextOut(x,y,SpeedStr);
875         }
876 }
```

Here is the call graph for this function:



4.22.3.47 void CEvolveTrafficView::DrawVehicleAt (CMemDC * *pDC*, WORD *VEH_ID*, CRect *VehRect*) [private]

Definition at line 878 of file EvolveTrafficView.cpp.

References VEH_COLOR_CAR, VEH_COLOR_CRANE, VEH_COLOR_-LARGETRUCK, VEH_COLOR_LOWLOADER, VEH_COLOR_SMALLTRUCK, VEH_ID_CAR, VEH_ID_CRANE, VEH_ID_LARGETRUCK, VEH_ID_-LOWLOADER, and VEH_ID_SMALLTRUCK.

Referenced by DrawLegendVehicle(), and DrawVehicle().

```
879 {
880         switch(VEH_ID)
881         {
882                 case VEH_ID_CAR:
883                         pDC->FillSolidRect(VehRect, VEH_COLOR_CAR);
884                         break;
885                 case VEH_ID_SMALLTRUCK:
886                         pDC->FillSolidRect(VehRect, VEH_COLOR_SMALLTRUCK);
887                         break;
888                 case VEH_ID_LARGETRUCK:
889                         pDC->FillSolidRect(VehRect, VEH_COLOR_LARGETRUCK);
890                         break;
891                 case VEH_ID_CRANE:
892                         pDC->FillSolidRect(VehRect, VEH_COLOR_CRANE);
893                         break;
894                 case VEH_ID_LOWLOADER:
895                         pDC->FillSolidRect(VehRect, VEH_COLOR_LOWLOADER);
896                         break;
897                 default:
```

```
898                         pDC->FillSolidRect(VehRect, VEH_COLOR_CAR);     // car
899         }
900 }
```

4.22.3.48 void CEvolveTrafficView::DrawDetectors (CMemDC * *pDC*) [private]

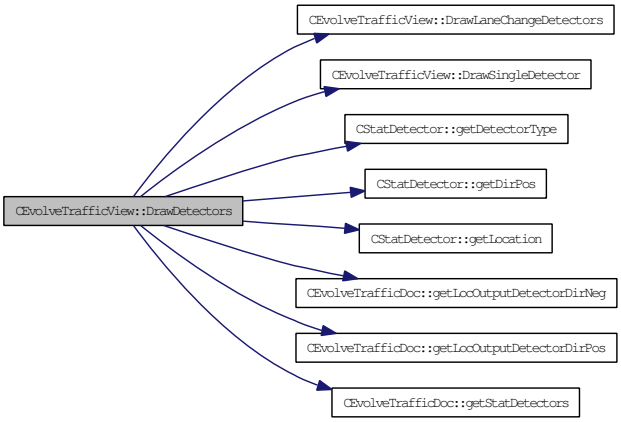
Definition at line 1102 of file EvolveTrafficView.cpp.

References CLR_DET_COMPOSITION, CLR_DET_FLOWDENSITY, CLR_DET_-HEADWAY, CLR_DET_OUTPUT, DrawLaneChangeDetectors(), DrawSingleDetector(), CStatDetector::getDetectorType(), CStatDetector::getDirPos(), CStatDetector::getLocation(), CEvolveTrafficDoc::getLocOutputDetectorDirNeg(), CEvolve-TrafficDoc::getLocOutputDetectorDirPos(), CEvolveTrafficDoc::getStatDetectors(), m_pDoc, METRICS_TYPE_COMPOSITION, METRICS_TYPE_FLOWDENSITY, METRICS_TYPE_HEADWAY, and METRICS_TYPE_LANE_CHANGE.

Referenced by DrawRoad().

```
1103 {
1104         // Output Detectors
1105         double LocDetDirPos = m_pDoc->getLocOutputDetectorDirPos();
1106         double LocDetDirNeg = m_pDoc->getLocOutputDetectorDirNeg();
1107         DrawSingleDetector(pDC, CLR_DET_OUTPUT, LocDetDirPos, true);
1108         DrawSingleDetector(pDC, CLR_DET_OUTPUT, LocDetDirNeg, false);
1109
1110         // Metrics Detectors
1111         CObArray vSD;   vSD.Copy( *(m_pDoc->getStatDetectors()) );
1112         for(int i = 0; i < vSD.GetSize(); i++)
1113         {
1114                 CStatDetector* pSD = reinterpret_cast<CStatDetector*>(vSD.GetAt(i));
1115                 WORD SDtype = pSD->getDetectorType();
1116                 bool bDraw = true;
1117                 COLORREF COL;
1118                 switch(SDtype)
1119                 {
1120                         case METRICS_TYPE_FLOWDENSITY:
1121                                 COL = CLR_DET_FLOWDENSITY;
1122                                 break;
1123                         case METRICS_TYPE_HEADWAY:
1124                                 COL = CLR_DET_HEADWAY;
1125                                 break;
1126                         case METRICS_TYPE_COMPOSITION:
1127                                 COL = CLR_DET_COMPOSITION;
1128                                 break;
1129                         case METRICS_TYPE_LANE_CHANGE:  // nothing to draw!
1130                                 DrawLaneChangeDetectors(pDC, pSD->getDirPos());
1131                                 bDraw = false;
1132                                 break;
1133                         default:
1134                                 COL = CLR_DET_FLOWDENSITY;
1135                 }
1136                 if(bDraw)
1137                         DrawSingleDetector(pDC, COL, pSD->getLocation(), pSD->getDirPos());
1138         }
1139 }
```

Here is the call graph for this function:



4.22.3.49 void CEvolveTrafficView::DrawLaneChangeDetectors (CMemDC * *pDC*, bool *DirPos*) [private]

Definition at line 1141 of file EvolveTrafficView.cpp.

References m_Border_Lhs, m_DriveOnRight, m_LaneWidth, m_NoLanesDirNeg, and m_NoLanesDirPos.

Referenced by DrawDetectors().

```
1142 {
1143         CString str = "LC";
1144         CSize strSize = pDC->GetTextExtent(str);
1145 
1146         int DriveSide = m_DriveOnRight ? +1 : -1;
1147         int DirFactor = DirPos ? +1 : -1;
1148         int nLanes = DirPos ? m_NoLanesDirPos : m_NoLanesDirNeg;
1149 
1150         int xScrnLoc = m_Border_Lhs - strSize.cx - 2;
1151         int ypos = DirFactor* DriveSide * nLanes * m_LaneWidth /2;
1152 
1153         pDC->TextOut(xScrnLoc,ypos-strSize.cy/2,str);
1154 }
```

4.22.3.50 void CEvolveTrafficView::DrawLegend (CMemDC * *pDC*) [private]

Definition at line 938 of file EvolveTrafficView.cpp.

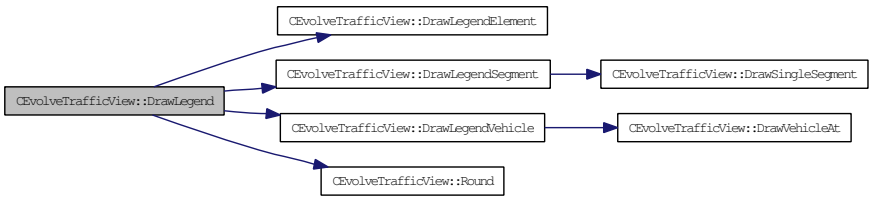
References CLR_BACKGRND, CLR_DET_COMPOSITION, CLR_DET_-FLOWDENSITY, CLR_DET_HEADWAY, CLR_DET_OUTPUT, CLR_FEAT_-GRADIENT, CLR_FEAT_SPEEDLIMIT, CLR_LEGEND_TEXT, DrawLegen-dElement(), DrawLegendSegment(), DrawLegendVehicle(), FEAT_GRADIENT, FEAT_SPEEDLIMIT, m_Border_Lhs, m_Border_Top, m_ClientRect, m_Scale,

Round(), VEH_ID_CAR, VEH_ID_CRANE, VEH_ID_LARGETRUCK, VEH_ID_-LOWLOADER, and VEH_ID_SMALLTRUCK.

Referenced by OnDraw().

```
939 {
940         pDC->SetTextColor(CLR_LEGEND_TEXT);
941         pDC->SetBkColor(CLR_BACKGRND);
942 
943         CString strTime = "Real Time - Day: 00 - 00:00:00.00";
944         CSize timeSize = pDC->GetTextExtent(strTime);
945 
946         // initial positions
947         int ypos = Round(-(m_ClientRect.Height()/2)/m_Scale) + m_Border_Top;    // this keeps t
948         int xpos = m_Border_Lhs + timeSize.cx + m_Border_Lhs;
949         int ygap = 20/m_Scale;
950         int LineLength = m_Border_Lhs/m_Scale;  // why not!
951         int LineToText = 20/m_Scale;
952 
953         DrawLegendVehicle(pDC, VEH_ID_CAR,                    "Car",                  LineTo
954         DrawLegendVehicle(pDC, VEH_ID_SMALLTRUCK,       "Small Truck",  LineToText, xpos, ypos)
955         DrawLegendVehicle(pDC, VEH_ID_LARGETRUCK,       "Large Truck",  LineToText, xpos, ypos)
956         DrawLegendVehicle(pDC, VEH_ID_CRANE,            "Crane",                LineToText, xpo
957         DrawLegendVehicle(pDC, VEH_ID_LOWLOADER,        "Low-loader",   LineToText, xpos, ypos)
958 
959         ypos -= 5*ygap; //reset ypos
960         CSize strSize = pDC->GetTextExtent("Large Truck");
961         xpos += LineLength+LineToText+strSize.cx+2*m_Border_Lhs;
962 
963         DrawLegendElement(pDC, LineLength, LineToText, xpos, ypos, "Output Detector", CLR_DET_C
964         DrawLegendElement(pDC, LineLength, LineToText, xpos, ypos, "Flow-Density Detector", CLR
965         DrawLegendElement(pDC, LineLength, LineToText, xpos, ypos, "Headway Detector", CLR_DET_
966         DrawLegendElement(pDC, LineLength, LineToText, xpos, ypos, "Composition Detector", CLR_
967 
968         pDC->TextOut(xpos,ypos,"LC");
969         pDC->TextOut(xpos+LineLength+LineToText,ypos,"Lane Change Detector");
970 
971         ypos -= 4*ygap; //reset ypos
972         strSize = pDC->GetTextExtent("Lane Change Detector");
973         xpos += LineLength+LineToText+strSize.cx+2*m_Border_Lhs;
974 
975         DrawLegendSegment(pDC, CLR_FEAT_SPEEDLIMIT, HS_DIAGCROSS, FEAT_SPEEDLIMIT,
976                 "Speed limit section", 50, LineToText, xpos, ypos);
977         ypos += 2*ygap+5;
978         DrawLegendSegment(pDC, CLR_FEAT_GRADIENT, HS_VERTICAL, FEAT_GRADIENT,
979                 "Gradient section", 5, LineToText, xpos, ypos);
980 
981 }
```

Here is the call graph for this function:



4.22.3.51 void CEvolveTrafficView::DrawLegendElement (CMemDC * *pDC*, int *LineLength*, int *LineToText*, int *xpos*, int *ypos*, CString *str*, COLORREF *COL*) [private]

Definition at line 1008 of file EvolveTrafficView.cpp.

Referenced by DrawLegend().

```
1010 {
1011         CPen DetPen;
1012
1013         DetPen.CreatePen(PS_SOLID, 3, COL);
1014         pDC->SelectObject(&DetPen);
1015         CSize strSize = pDC->GetTextExtent(str);
1016         pDC->TextOut(xpos+LineLength+LineToText,ypos,str);
1017         ypos += strSize.cy/2;
1018         pDC->MoveTo(xpos,ypos);
1019         pDC->LineTo(xpos+LineLength,ypos);
1020 }
```

4.22.3.52 void CEvolveTrafficView::DrawSingleSegment (CMemDC * *pDC*, COLORREF *COL*, WORD *HATCH*, WORD *FeatType*, int *val*, CRect *rect*, int *yTextOffset*) [private]

Definition at line 1062 of file EvolveTrafficView.cpp.

References FEAT_GRADIENT, and FEAT_SPEEDLIMIT.

Referenced by DrawLegendSegment(), and DrawRoadSegments().

```
1064 {
1065         int x1 = rect.left;
1066         int x2 = rect.right;
1067         int y1 = rect.top;
1068         int y2 = rect.bottom;
1069
1070         CPen SegEdgePen(PS_SOLID, 1, COL);
1071         CPen* pOldPen = pDC->SelectObject(&SegEdgePen);         // store previous pen
1072         CBrush SegBrush(HATCH,COL);
1073         CRect SegRect(x1,y1,x2,y2);
1074         CBrush* pOldBrush = pDC->SelectObject(&SegBrush);       // select drawing object and s
1075         pDC->SetBkMode(TRANSPARENT);    // set background mode
1076         pDC->Rectangle(SegRect);                // draw a rectangle which will be patterned /
1077         pDC->SelectObject(pOldBrush);   // reset previous objects
1078
1079         CPen SegEndsPen(PS_SOLID, 3, COL);      // MAGIC NUMBER - seg pen width - keep same as
1080         pDC->SelectObject(&SegEndsPen);
1081         pDC->MoveTo(x1,y1);     pDC->LineTo(x1,y2);
1082         pDC->MoveTo(x2,y1);     pDC->LineTo(x2,y2);
1083
1084         pDC->SelectObject(pOldPen);                     // now reselect old pen
1085
1086         CString str;
1087         switch(FeatType)
1088         {
1089                 case FEAT_SPEEDLIMIT:
1090                         str.Format("%d km/h",val);
1091                         break;
1092                 case FEAT_GRADIENT:
```

```
1093                        str.Format("%d%%",val);
1094                        break;
1095                default:
1096                        str.Format("%d km/h",val);
1097        }
1098        CSize strSize = pDC->GetTextExtent(str);
1099        pDC->TextOut((x1+x2)/2-strSize.cx/2,(y1+y2)/2-strSize.cy/2+yTextOffset,str);
1100 }
```

4.22.3.53 void CEvolveTrafficView::DrawRoadSegments (CMemDC * *pDC*) [private]

Definition at line 1022 of file EvolveTrafficView.cpp.

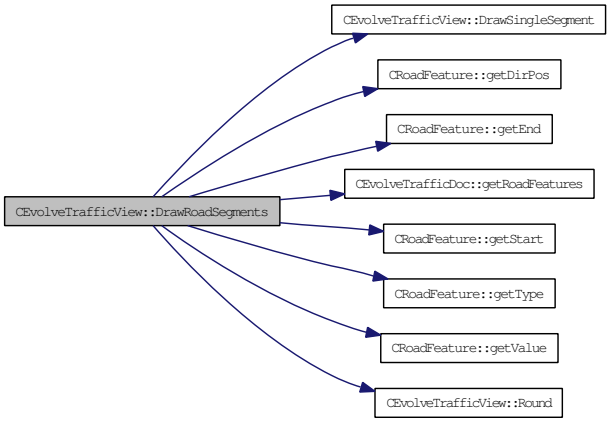
References CLR_FEAT_GRADIENT, CLR_FEAT_SPEEDLIMIT, DrawSingleSegment(), FEAT_GRADIENT, FEAT_SPEEDLIMIT, CRoadFeature::getDirPos(), CRoadFeature::getEnd(), CEvolveTrafficDoc::getRoadFeatures(), CRoadFeature::getStart(), CRoadFeature::getType(), CRoadFeature::getValue(), m_Border_Lhs, m_DriveOnRight, m_LaneWidth, m_NoLanesDirNeg, m_NoLanesDirPos, m_pDoc, M_PER_S_TO_KM_PER_H, and Round().

Referenced by DrawRoad().

```
1023 {
1024        CObArray vSeg;  vSeg.Copy( *(m_pDoc->getRoadFeatures()) );
1025        for(int i = 0; i < vSeg.GetSize(); i++)
1026        {
1027                CRoadFeature* pFeat = reinterpret_cast<CRoadFeature*>(vSeg.GetAt(i));
1028                WORD FeatType = pFeat->getType();
1029                int FeatValue = 0;
1030                COLORREF COL;
1031                WORD HATCH;
1032                switch(FeatType)
1033                {
1034                        case FEAT_SPEEDLIMIT:
1035                                COL = CLR_FEAT_SPEEDLIMIT;
1036                                HATCH = HS_DIAGCROSS;
1037                                FeatValue = Round(pFeat->getValue()*M_PER_S_TO_KM_PER_H);
1038                                break;
1039                        case FEAT_GRADIENT:
1040                                COL = CLR_FEAT_GRADIENT;
1041                                HATCH = HS_VERTICAL;
1042                                FeatValue = Round(pFeat->getValue());
1043                                break;
1044                        default:
1045                                COL = CLR_FEAT_SPEEDLIMIT;
1046                }
1047
1048                bool DirPos = pFeat->getDirPos();
1049                int DriveSide = m_DriveOnRight ? +1 : -1;
1050                int DirFactor = DirPos ? +1 : -1;
1051                int nLanes = DirPos ? m_NoLanesDirPos : m_NoLanesDirNeg;
1052
1053                int x1 = pFeat->getStart() + m_Border_Lhs;
1054                int x2 = pFeat->getEnd() + m_Border_Lhs;
1055                int y1 = 0;
1056                int y2 = DirFactor * DriveSide * nLanes * m_LaneWidth;
1057                CRect segRect(x1,y1,x2,y2);
1058                DrawSingleSegment(pDC, COL, HATCH, FeatType, FeatValue, segRect,m_LaneWidth/2)
```

```
1059         }
1060 }
```

Here is the call graph for this function:



4.22.3.54 void CEvolveTrafficView::DrawSingleDetector (CMemDC * *pDC*, COLORREF *COL*, int *Loc*, bool *DirPos*) [private]

Definition at line 1156 of file EvolveTrafficView.cpp.

References m_Border_Lhs, m_DriveOnRight, m_LaneWidth, m_NoLanesDirNeg, and m_NoLanesDirPos.

Referenced by DrawDetectors().

```
1157 {
1158         // Draw detectors
1159         CPen DetectorPen(PS_SOLID, 3, COL);     // MAGIC NUMBER - detector pen width
1160         pDC->SelectObject(&DetectorPen);
1161 
1162         int ScrnLoc = Loc + m_Border_Lhs;
1163 
1164         int DriveSide = m_DriveOnRight ? +1 : -1;
1165         int DirFactor = DirPos ? +1 : -1;
1166         int nLanes = DirPos ? m_NoLanesDirPos : m_NoLanesDirNeg;
1167 
1168         int ypos = 0;//DirFactor*DriveSide*m_RoadWidth/2;// + m_ClientRect.Height()/2;  // bot
1169         pDC->MoveTo(ScrnLoc,ypos);
1170         //ypos = ypos - DirFactor* DriveSide * nLanes * m_LaneWidth;
1171         ypos = DirFactor* DriveSide * nLanes * m_LaneWidth;
1172         pDC->LineTo(ScrnLoc,ypos);
1173 }
```

4.22.3.55 void CEvolveTrafficView::DrawLegendSegment (CMemDC * *pDC*, COLORREF *COL*, WORD *HATCH*, WORD *FeatType*, CString *str*, int *value*, int *LineToText*, int *xpos*, int *ypos*) [private]

Definition at line 995 of file EvolveTrafficView.cpp.

References DrawSingleSegment(), and m_Scale.

Referenced by DrawLegend().

```
997 {
998         int rectWidth = 80/m_Scale;
999         int rectHeight = 30/m_Scale;
1000
1001         CRect segRect(xpos,ypos,xpos+rectWidth, ypos+rectHeight);
1002         DrawSingleSegment(pDC, COL, HATCH, FeatType, value, segRect, 0);
1003
1004         CSize strSize = pDC->GetTextExtent(str);
1005         pDC->TextOut(xpos+rectWidth+LineToText, ypos+rectHeight/2-strSize.cy/2,str);
1006 }
```

Here is the call graph for this function:

4.22.3.56 void CEvolveTrafficView::DrawLegendVehicle (CMemDC * *pDC*, WORD *VehType*, CString *str*, int *LineToText*, int *xpos*, int *ypos*) [private]

Definition at line 983 of file EvolveTrafficView.cpp.

References DrawVehicleAt(), and m_Scale.

Referenced by DrawLegend().

```
984 {
985         int rectWidth = 20/m_Scale;
986         int rectHeight = 10/m_Scale;
987
988         CSize strSize = pDC->GetTextExtent(str);
989         pDC->TextOut(xpos+rectWidth+LineToText, ypos,str);
990         ypos += strSize.cy/2 - rectHeight/2;
991         CRect VehRect(xpos,ypos,xpos+rectWidth, ypos+rectHeight);
992         DrawVehicleAt(pDC, VehType, VehRect);
993 }
```

Here is the call graph for this function:

4.22.4 Member Data Documentation

4.22.4.1 bool CEvolveTrafficView::m_bShowLegend [private]

Definition at line 85 of file EvolveTrafficView.h.

Referenced by OnDraw(), and OnToolsPrefs().

4.22.4.2 bool CEvolveTrafficView::m_bShowVelocity [private]

Definition at line 86 of file EvolveTrafficView.h.

Referenced by DrawVehicle(), and OnToolsPrefs().

4.22.4.3 int CEvolveTrafficView::m_BtmEdge [private]

Definition at line 90 of file EvolveTrafficView.h.

Referenced by DrawRoad(), DrawRuler(), DrawVehicle(), initRoad(), and OnLButtonDown().

4.22.4.4 int CEvolveTrafficView::m_TopEdge [private]

Definition at line 91 of file EvolveTrafficView.h.

Referenced by DrawRoad(), DrawVehicle(), initRoad(), and OnLButtonDown().

4.22.4.5 int CEvolveTrafficView::m_nLanesOnBtm [private]

Definition at line 92 of file EvolveTrafficView.h.

Referenced by initRoad().

4.22.4.6 int CEvolveTrafficView::m_nLanesOnTop [private]

Definition at line 93 of file EvolveTrafficView.h.

Referenced by initRoad().

4.22.4.7 bool CEvolveTrafficView::m_DriveOnRight [private]

Definition at line 94 of file EvolveTrafficView.h.

Referenced by DrawLaneChangeDetectors(), DrawRoadSegments(), DrawSingleDetector(), DrawVehicle(), initRoad(), and OnLButtonDown().

4.22.4.8 double CEvolveTrafficView::m_ExaggerateWidths [private]

Definition at line 95 of file EvolveTrafficView.h.

Referenced by OnToolsPrefs().

4.22.4.9 CSize CEvolveTrafficView::m_DrawArea [private]

Definition at line 98 of file EvolveTrafficView.h.

Referenced by initRoad(), OnInitialUpdate(), OnPrepareDC(), and SetZoomScale().

4.22.4.10 int CEvolveTrafficView::m_LastPercentComplete [private]

Definition at line 101 of file EvolveTrafficView.h.

Referenced by doSimFinished(), and UpdateProgressBar().

4.22.4.11 double CEvolveTrafficView::m_MaxTimeWarp [private]

Definition at line 128 of file EvolveTrafficView.h.

Referenced by setMaxTimeWarp(), and ValidateTimeWarp().

4.22.4.12 int CEvolveTrafficView::m_SimTimeStep [private]

Definition at line 129 of file EvolveTrafficView.h.

Referenced by doLoop(), and initRoad().

4.22.4.13 int CEvolveTrafficView::m_PauseTime [private]

Definition at line 130 of file EvolveTrafficView.h.

Referenced by OnToolsPause().

4.22.4.14 int CEvolveTrafficView::m_StartRealTime [private]

Definition at line 131 of file EvolveTrafficView.h.

Referenced by doLoop(), doSimStart(), DrawTimer(), OnRunInvisible(), and OnToolsPause().

4.22.4.15 int CEvolveTrafficView::m_CurentRealTime [private]

Definition at line 132 of file EvolveTrafficView.h.

Referenced by DrawTimer(), and OnRunInvisible().

4.22.4.16 double CEvolveTrafficView::m_CurentSimTime [private]

Definition at line 133 of file EvolveTrafficView.h.

Referenced by doSimStart(), DrawTimer(), and OnRunInvisible().

4.22.4.17 BOOL CEvolveTrafficView::m_bPause [private]

Definition at line 134 of file EvolveTrafficView.h.

Referenced by doLoop(), DrawTimer(), OnLButtonDown(), OnRunVisible(), OnToolsPause(), and OnUpdateToolsPause().

4.22.4.18 int CEvolveTrafficView::m_TimerSpeed [private]

Definition at line 135 of file EvolveTrafficView.h.

Referenced by doLoop().

4.22.4.19 int CEvolveTrafficView::m_PercentComplete [private]

Definition at line 136 of file EvolveTrafficView.h.

Referenced by UpdateProgressBar().

4.22.4.20 CRect CEvolveTrafficView::m_ClientRect [private]

Definition at line 138 of file EvolveTrafficView.h.

Referenced by DrawLegend(), DrawTimer(), OnPrepareDC(), and SetZoomScale().

4.22.4.21 int CEvolveTrafficView::m_YCoordTop [private]

Definition at line 139 of file EvolveTrafficView.h.

4.22.4.22 RECT CEvolveTrafficView::m_RoadRect [private]

Definition at line 140 of file EvolveTrafficView.h.

4.22.4.23 bool CEvolveTrafficView::m_bInSimulation [private]

Definition at line 141 of file EvolveTrafficView.h.

Referenced by doLoop(), doSimFinished(), doSimStart(), DrawTimer(), OnDraw(), OnRunInvisible(), OnRunVisible(), OnUpdateToolsPause(), OnUpdateToolsSlowdown(), OnUpdateToolsSpeedup(), and OnUpdateToolsStop().

4.22.4.24 double CEvolveTrafficView::m_Scale [private]

Definition at line 143 of file EvolveTrafficView.h.

Referenced by DrawLegend(), DrawLegendSegment(), DrawLegendVehicle(), DrawTimer(), initRoad(), OnPrepareDC(), OnToolsPrefs(), OnToolsZoomin(), OnToolsZoomout(), and SetZoomScale().

4.22.4.25 double CEvolveTrafficView::m_TimeWarp [private]

Definition at line 144 of file EvolveTrafficView.h.

Referenced by doLoop(), OnToolsPrefs(), OnToolsSlowdown(), OnToolsSpeedup(), setMaxTimeWarp(), and ValidateTimeWarp().

4.22.4.26 double CEvolveTrafficView::m_VehicleLengthScale [private]

Definition at line 145 of file EvolveTrafficView.h.

Referenced by DrawVehicle(), FindVehicle(), and OnToolsPrefs().

4.22.4.27 double CEvolveTrafficView::m_VehicleWidth [private]

Definition at line 146 of file EvolveTrafficView.h.

Referenced by DrawVehicle(), and OnToolsPrefs().

4.22.4.28 double CEvolveTrafficView::m_TickLength [private]

Definition at line 147 of file EvolveTrafficView.h.

Referenced by DrawRuler(), and OnToolsPrefs().

4.22.4.29 int CEvolveTrafficView::m_TickStep [private]

Definition at line 148 of file EvolveTrafficView.h.

Referenced by DrawRuler(), initRoad(), and OnToolsPrefs().

4.22.4.30 double CEvolveTrafficView::m_LaneWidth [private]

Definition at line 149 of file EvolveTrafficView.h.

Referenced by DrawLaneChangeDetectors(), DrawRoad(), DrawRoadSegments(), DrawSingleDetector(), DrawVehicle(), initRoad(), OnLButtonDown(), and OnToolsPrefs().

4.22.4.31 double CEvolveTrafficView::m_Border_Top [private]

Definition at line 150 of file EvolveTrafficView.h.

Referenced by DrawLegend(), DrawTimer(), initRoad(), and OnToolsPrefs().

4.22.4.32 double CEvolveTrafficView::m_Border_Btm [private]

Definition at line 151 of file EvolveTrafficView.h.

Referenced by initRoad(), and OnToolsPrefs().

4.22.4.33 double CEvolveTrafficView::m_Border_Lhs [private]

Definition at line 152 of file EvolveTrafficView.h.

Referenced by DrawLaneChangeDetectors(), DrawLegend(), DrawRoad(), DrawRoadSegments(), DrawRuler(), DrawSingleDetector(), DrawTimer(), DrawVehicle(), initRoad(), OnLButtonDown(), OnPrepareDC(), and OnToolsPrefs().

4.22.4.34 double CEvolveTrafficView::m_Border_Rhs [private]

Definition at line 153 of file EvolveTrafficView.h.

Referenced by initRoad(), and OnToolsPrefs().

4.22.4.35 int CEvolveTrafficView::m_NoDirections [private]

Definition at line 155 of file EvolveTrafficView.h.

Referenced by DrawRoad(), and initRoad().

4.22.4.36 int CEvolveTrafficView::m_NoLanes [private]

Definition at line 156 of file EvolveTrafficView.h.

Referenced by DrawRoad(), initRoad(), and OnLButtonDown().

4.22.4.37 int CEvolveTrafficView::m_NoLanesDirPos [private]

Definition at line 157 of file EvolveTrafficView.h.

Referenced by DrawLaneChangeDetectors(), DrawRoadSegments(), DrawSingleDetector(), and initRoad().

4.22.4.38 int CEvolveTrafficView::m_NoLanesDirNeg [private]

Definition at line 158 of file EvolveTrafficView.h.

Referenced by DrawLaneChangeDetectors(), DrawRoadSegments(), DrawSingleDetector(), and initRoad().

4.22.4.39 int CEvolveTrafficView::m_RoadLength [private]

Definition at line 159 of file EvolveTrafficView.h.

Referenced by DrawRoad(), initRoad(), OnConfigFeatures(), OnConfigMetrics(), and OnKeyDown().

4.22.4.40 int CEvolveTrafficView::m_NoTicks [private]

Definition at line 160 of file EvolveTrafficView.h.

Referenced by DrawRuler(), and initRoad().

4.22.4.41 int CEvolveTrafficView::m_RoadWidth [private]

Definition at line 161 of file EvolveTrafficView.h.

Referenced by DrawRoad(), and initRoad().

4.22.4.42 M2D CEvolveTrafficView::m_VehiclePositions [private]

Definition at line 163 of file EvolveTrafficView.h.

Referenced by doLoop(), doSimFinished(), FindVehicle(), and OnDraw().

4.22.4.43 CEvolveTrafficDoc* CEvolveTrafficView::m_pDoc [private]

Definition at line 165 of file EvolveTrafficView.h.

Referenced by doLoop(), doSimFinished(), doSimStart(), DrawDetectors(), DrawRoadSegments(), DrawTimer(), initRoad(), OnConfigFeatures(), OnConfigMetrics(), OnConfigSim(), OnConfigTraf(), OnInitialUpdate(), OnRunInvisible(), and UpdateProgressBar().

4.22.4.44 CProgressBar* CEvolveTrafficView::m_pProgBar [private]

Definition at line 166 of file EvolveTrafficView.h.

Referenced by doSimFinished(), doSimStart(), and UpdateProgressBar().

4.22.4.45 CMessageTip CEvolveTrafficView::m_DataTip [private]

Definition at line 167 of file EvolveTrafficView.h.

Referenced by OnInitialUpdate(), and OnLButtonDown().

4.22.4.46 double CEvolveTrafficView::MIN_VIEW_SCALE [private]

Definition at line 169 of file EvolveTrafficView.h.

Referenced by SetZoomScale().

4.22.4.47 double CEvolveTrafficView::DATATIP_DELAY [private]

Definition at line 170 of file EvolveTrafficView.h.

Referenced by OnInitialUpdate().

4.22.4.48 COLORREF CEvolveTrafficView::CLR_BACKGRND [private]

Definition at line 172 of file EvolveTrafficView.h.

Referenced by DrawLegend(), DrawRuler(), and DrawTimer().

4.22.4.49 COLORREF CEvolveTrafficView::CLR_DET_COMPOSITION [private]

Definition at line 173 of file EvolveTrafficView.h.

Referenced by DrawDetectors(), and DrawLegend().

4.22.4.50 COLORREF CEvolveTrafficView::CLR_DET_FLOWDENSITY [private]

Definition at line 174 of file EvolveTrafficView.h.

Referenced by DrawDetectors(), and DrawLegend().

4.22.4.51 COLORREF CEvolveTrafficView::CLR_DET_HEADWAY [private]

Definition at line 175 of file EvolveTrafficView.h.

Referenced by DrawDetectors(), and DrawLegend().

4.22.4.52 COLORREF CEvolveTrafficView::CLR_DET_OUTPUT [private]

Definition at line 176 of file EvolveTrafficView.h.

Referenced by DrawDetectors(), and DrawLegend().

4.22.4.53 COLORREF CEvolveTrafficView::CLR_FEAT_GRADIENT [private]

Definition at line 177 of file EvolveTrafficView.h.

Referenced by DrawLegend(), and DrawRoadSegments().

4.22.4.54 COLORREF CEvolveTrafficView::CLR_FEAT_SPEEDLIMIT [private]

Definition at line 178 of file EvolveTrafficView.h.

Referenced by DrawLegend(), and DrawRoadSegments().

4.22.4.55 COLORREF CEvolveTrafficView::CLR_LEGEND_TEXT [private]

Definition at line 179 of file EvolveTrafficView.h.

Referenced by DrawLegend(), and DrawTimer().

4.22.4.56 COLORREF CEvolveTrafficView::CLR_ROAD_LINES [private]

Definition at line 180 of file EvolveTrafficView.h.

Referenced by DrawRoad().

4.22.4.57 COLORREF CEvolveTrafficView::CLR_ROAD_SURFACE [private]

Definition at line 181 of file EvolveTrafficView.h.

Referenced by DrawRoad().

4.22.4.58 COLORREF CEvolveTrafficView::CLR_RULER_LINES [private]

Definition at line 182 of file EvolveTrafficView.h.

Referenced by DrawRuler().

4.22.4.59 COLORREF CEvolveTrafficView::CLR_RULER_TEXT [private]

Definition at line 183 of file EvolveTrafficView.h.

Referenced by DrawRuler().

4.22.4.60 COLORREF CEvolveTrafficView::VEH_COLOR_CAR [private]

Definition at line 185 of file EvolveTrafficView.h.

Referenced by DrawVehicleAt().

4.22.4.61 COLORREF CEvolveTrafficView::VEH_COLOR_SMALLTRUCK [private]

Definition at line 186 of file EvolveTrafficView.h.

Referenced by DrawVehicleAt().

4.22.4.62 COLORREF CEvolveTrafficView::VEH_COLOR_LARGETRUCK [private]

Definition at line 187 of file EvolveTrafficView.h.

Referenced by DrawVehicleAt().

4.22.4.63 COLORREF CEvolveTrafficView::VEH_COLOR_CRANE [private]

Definition at line 188 of file EvolveTrafficView.h.

Referenced by DrawVehicleAt().

4.22.4.64 COLORREF CEvolveTrafficView::VEH_COLOR_LOWLOADER [private]

Definition at line 189 of file EvolveTrafficView.h.

Referenced by DrawVehicleAt().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/EvolveTrafficView.h
- D:/~Research/Code/C++/EvolveTraffic/EvolveTrafficView.cpp

4.23 CIDMParameterSet Class Reference

A class representing a set of IDM parameters that can be saved to file.

`#include <IDMParameterSet.h>`

Collaboration diagram for CIDMParameterSet:

Public Member Functions

- CIDMParameterSet ()
- CIDMParameterSet (WORD id)
- virtual ~CIDMParameterSet ()
- CIDMParameterSet (const CIDMParameterSet &ParamSet)
- void Serialize (CArchive &ar)
- CIDMParameterSet & operator= (const CIDMParameterSet &param)
- IDM Generate ()
- CParameter * get_T ()
- CParameter * get_A ()

- CParameter * get_B ()
- CParameter * get_S0 ()
- CParameter * get_S1 ()
- CParameter * get_V0 ()
- CParameter * get_Delta ()
- CParameter * get_Polite ()
- CParameter * get_Bias ()
- CParameter * get_DeltaAth ()
- int getVehicleTypeID () const
- void set_T (CParameter *param)
- void set_A (CParameter *param)
- void set_B (CParameter *param)
- void set_S0 (CParameter *param)
- void set_S1 (CParameter *param)
- void set_V0 (CParameter *param)
- void set_Delta (CParameter *param)
- void set_Polite (CParameter *param)
- void set_Bias (CParameter *param)
- void set_DeltaAth (CParameter *param)
- void setVehicleTypeID (int id)

Private Member Functions

- void setAll (const CIDMParameterSet &ParamSet)
- void buildParamSet ()

Private Attributes

- int m_VehicleTypeID
- CParameter m_T
- CParameter m_A
- CParameter m_B
- CParameter m_S0
- CParameter m_S1
- CParameter m_V0
- CParameter m_Delta
- CParameter m_Polite
- CParameter m_Bias
- CParameter m_DeltaAth

4.23.1 Detailed Description

A class representing a set of IDM parameters that can be saved to file.

Definition at line 19 of file IDMParameterSet.h.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 CIDMParameterSet::CIDMParameterSet ()

Definition at line 19 of file IDMParameterSet.cpp.

```
20 {
21         buildParamSet();
22 }
```

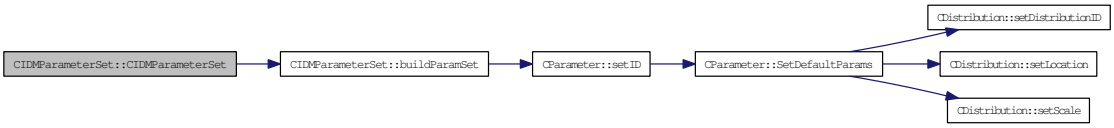
4.23.2.2 CIDMParameterSet::CIDMParameterSet (WORD *id*)

Definition at line 24 of file IDMParameterSet.cpp.

References buildParamSet(), and m_VehicleTypeID.

```
25 {
26         m_VehicleTypeID = id;
27         buildParamSet();
28 }
```

Here is the call graph for this function:



4.23.2.3 CIDMParameterSet::~CIDMParameterSet () [virtual]

Definition at line 30 of file IDMParameterSet.cpp.

```
31 {
32
33 }
```

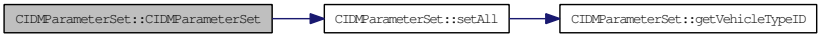
4.23.2.4 CIDMParameterSet::CIDMParameterSet (const CIDMParameterSet & *ParamSet*)

Definition at line 35 of file IDMParameterSet.cpp.

References setAll().

```
36 {
37         setAll(ParamSet);
38 }
```

Here is the call graph for this function:



4.23.3 Member Function Documentation

4.23.3.1 void CIDMParameterSet::Serialize (CArchive & *ar*)

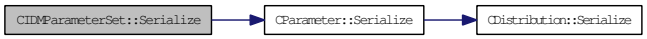
Definition at line 58 of file IDMParameterSet.cpp.

References m_A, m_B, m_Bias, m_Delta, m_DeltaAth, m_Polite, m_S0, m_S1, m_T, m_V0, m_VehicleTypeID, and CParameter::Serialize().

Referenced by CEvolveTrafficDoc::Serialize().

```
59 {
60         m_T                     .Serialize(ar);
61         m_A                     .Serialize(ar);
62         m_B                     .Serialize(ar);
63         m_S0            .Serialize(ar);
64         m_S1            .Serialize(ar);
65         m_V0            .Serialize(ar);
66         m_Delta         .Serialize(ar);
67         m_Polite        .Serialize(ar);
68         m_Bias          .Serialize(ar);
69         m_DeltaAth      .Serialize(ar);
70
71         if (ar.IsStoring())
72                 ar << m_VehicleTypeID;
73         else
74                 ar >> m_VehicleTypeID;
75 }
```

Here is the call graph for this function:



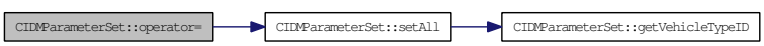
4.23.3.2 CIDMParameterSet & CIDMParameterSet::operator= (const CIDMParameterSet & *param*)

Definition at line 114 of file IDMParameterSet.cpp.

References setAll().

```
115 {
116         setAll(ParamSet);
117         return *this;
118 }
```

Here is the call graph for this function:



4.23.3.3 IDM CIDMParameterSet::Generate ()

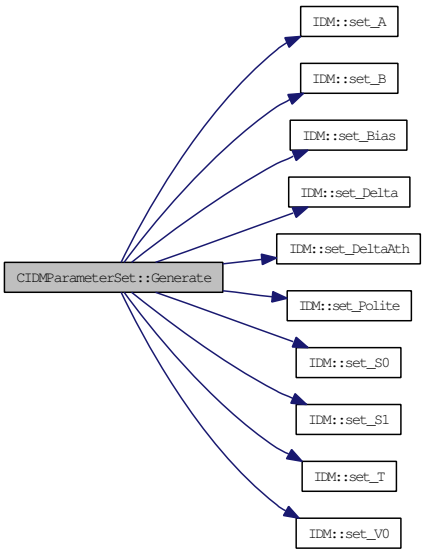
Definition at line 232 of file IDMParameterSet.cpp.

References m_A, m_B, m_Bias, m_Delta, m_DeltaAth, m_Polite, m_S0, m_S1, m_T, m_V0, IDM::set_A(), IDM::set_B(), IDM::set_Bias(), IDM::set_Delta(), IDM::set_DeltaAth(), IDM::set_Polite(), IDM::set_S0(), IDM::set_S1(), IDM::set_T(), and IDM::set_V0().

Referenced by Road::setIDMDriverModel().

```
233 {
234         IDM tempIDM;
235 
236         tempIDM.set_T           (m_T            .Generate() );
237         tempIDM.set_A           (m_A            .Generate() );
238         tempIDM.set_B           (m_B            .Generate() );
239         tempIDM.set_S0          (m_S0           .Generate() );
240         tempIDM.set_S1          (m_S1           .Generate() );
241         tempIDM.set_V0          (m_V0           .Generate() );
242         tempIDM.set_Delta       (m_Delta        .Generate() );
243         tempIDM.set_Polite      (m_Polite       .Generate() );
244         tempIDM.set_Bias        (m_Bias         .Generate() );
245         tempIDM.set_DeltaAth(m_DeltaAth .Generate() );
246 
247         return tempIDM;
248 }
```

Here is the call graph for this function:



4.23.3.4 CParameter * CIDMParameterSet::get_T ()

Definition at line 125 of file IDMParameterSet.cpp.

References m_T.

Referenced by CTrafficConfigDlg::MapRowToParameter().

```
126 {
127         return &m_T;
128 }
```

4.23.3.5 CParameter * CIDMParameterSet::get_A ()

Definition at line 135 of file IDMParameterSet.cpp.

References m_A.

Referenced by CTrafficConfigDlg::MapRowToParameter().

```
136 {
137         return &m_A;
138 }
```

4.23.3.6 CParameter * CIDMParameterSet::get_B ()

Definition at line 145 of file IDMParameterSet.cpp.

References m_B.

Referenced by CTrafficConfigDlg::MapRowToParameter().

```
146 {
147         return &m_B;
148 }
```

4.23.3.7 CParameter * CIDMParameterSet::get_S0 ()

Definition at line 155 of file IDMParameterSet.cpp.

References m_S0.

Referenced by CTrafficConfigDlg::MapRowToParameter().

```
156 {
157         return &m_S0;
158 }
```

4.23.3.8 CParameter * CIDMParameterSet::get_S1 ()

Definition at line 165 of file IDMParameterSet.cpp.

References m_S1.

Referenced by CTrafficConfigDlg::MapRowToParameter().

```
166 {
167         return &m_S1;
168 }
```

4.23.3.9 CParameter * CIDMParameterSet::get_V0 ()

Definition at line 175 of file IDMParameterSet.cpp.

References m_V0.

Referenced by SpeedLimit::addVehicle(), and CTrafficConfigDlg::MapRowToParameter().

```
176 {
177         return &m_V0;
178 }
```

4.23.3.10 CParameter * CIDMParameterSet::get_Delta ()

Definition at line 185 of file IDMParameterSet.cpp.

References m_Delta.

Referenced by CTrafficConfigDlg::MapRowToParameter().

```
186 {
187         return &m_Delta;
188 }
```

4.23.3.11 CParameter * CIDMParameterSet::get_Polite ()

Definition at line 195 of file IDMParameterSet.cpp.

References m_Polite.

Referenced by CTrafficConfigDlg::MapRowToParameter().

```
196 {
197         return &m_Polite;
198 }
```

4.23.3.12 CParameter * CIDMParameterSet::get_Bias ()

Definition at line 205 of file IDMParameterSet.cpp.

References m_Bias.

Referenced by CTrafficConfigDlg::MapRowToParameter().

```
206 {
207         return &m_Bias;
208 }
```

4.23.3.13 CParameter * CIDMParameterSet::get_DeltaAth ()

Definition at line 215 of file IDMParameterSet.cpp.

References m_DeltaAth.

Referenced by CTrafficConfigDlg::MapRowToParameter().

```
216 {
217         return &m_DeltaAth;
218 }
```

4.23.3.14 int CIDMParameterSet::getVehicleTypeID () const

Definition at line 225 of file IDMParameterSet.cpp.

References m_VehicleTypeID.

Referenced by setAll().

```
226 {
227         return m_VehicleTypeID;
228 }
```

4.23.3.15 void CIDMParameterSet::set_T (CParameter * *param*)

Definition at line 120 of file IDMParameterSet.cpp.

References m_T.

```
121 {
122         m_T = *param;
123 }
```

4.23.3.16 void CIDMParameterSet::set_A (CParameter * *param*)

Definition at line 130 of file IDMParameterSet.cpp.

References m_A.

```
131 {
132         m_A = *param;
133 }
```

4.23.3.17 void CIDMParameterSet::set_B (CParameter * *param*)

Definition at line 140 of file IDMParameterSet.cpp.

References m_B.

```
141 {
142         m_B = *param;
143 }
```

4.23.3.18 void CIDMParameterSet::set_S0 (CParameter * *param*)

Definition at line 150 of file IDMParameterSet.cpp.

References m_S0.

```
151 {
152         m_S0 = *param;
153 }
```

4.23.3.19 void CIDMParameterSet::set_S1 (CParameter * *param*)

Definition at line 160 of file IDMParameterSet.cpp.

References m_S1.

```
161 {
162         m_S1 = *param;
163 }
```

4.23.3.20 void CIDMParameterSet::set_V0 (CParameter * *param*)

Definition at line 170 of file IDMParameterSet.cpp.

References m_V0.

```
171 {
172         m_V0 = *param;
173 }
```

4.23.3.21 void CIDMParameterSet::set_Delta (CParameter * *param*)

Definition at line 180 of file IDMParameterSet.cpp.

References m_Delta.

```
181 {
182         m_Delta = *param;
183 }
```

4.23.3.22 void CIDMParameterSet::set_Polite (CParameter * *param*)

Definition at line 190 of file IDMParameterSet.cpp.

References m_Polite.

```
191 {
192         m_Polite = *param;
193 }
```

4.23.3.23 void CIDMParameterSet::set_Bias (CParameter * *param*)

Definition at line 200 of file IDMParameterSet.cpp.

References m_Bias.

```
201 {
202         m_Bias = *param;
203 }
```

4.23.3.24 void CIDMParameterSet::set_DeltaAth (CParameter * *param*)

Definition at line 210 of file IDMParameterSet.cpp.

References m_DeltaAth.

```
211 {
212         m_DeltaAth = *param;
213 }
```

4.23.3.25 void CIDMParameterSet::setVehicleTypeID (int *id*)

Definition at line 220 of file IDMParameterSet.cpp.

References m_VehicleTypeID.

```
221 {
222         m_VehicleTypeID = id;
223 }
```

4.23.3.26 void CIDMParameterSet::setAll (const CIDMParameterSet & *ParamSet*) [private]

Definition at line 81 of file IDMParameterSet.cpp.

References getVehicleTypeID(), m_A, m_B, m_Bias, m_Delta, m_DeltaAth, m_Polite, m_S0, m_S1, m_T, m_V0, and m_VehicleTypeID.

Referenced by CIDMParameterSet(), and operator=().

```
82 {
83         this->m_T               = ParamSet.m_T;
84         this->m_A               = ParamSet.m_A;
85         this->m_B               = ParamSet.m_B;
86         this->m_S0              = ParamSet.m_S0;
87         this->m_S1              = ParamSet.m_S1;
88         this->m_V0              = ParamSet.m_V0;
89         this->m_Delta   = ParamSet.m_Delta;
90         this->m_Polite  = ParamSet.m_Polite;
91         this->m_Bias    = ParamSet.m_Bias;
92         this->m_DeltaAth= ParamSet.m_DeltaAth;
93
94         this->m_VehicleTypeID = ParamSet.getVehicleTypeID();
95 }
```

Here is the call graph for this function:

4.23.3.27 void CIDMParameterSet::buildParamSet () [private]

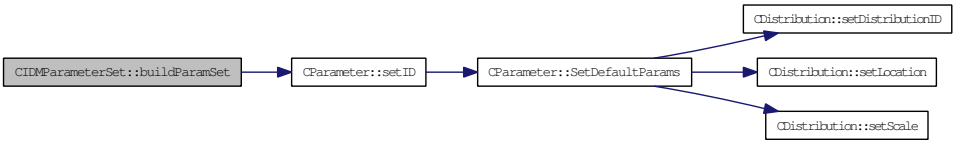
Definition at line 97 of file IDMParameterSet.cpp.

References IDM_PARAM_A, IDM_PARAM_B, IDM_PARAM_BIAS, IDM_-PARAM_DELTA, IDM_PARAM_DELTAATH, IDM_PARAM_POLITE, IDM_-PARAM_S0, IDM_PARAM_S1, IDM_PARAM_T, IDM_PARAM_V0, m_A, m_B, m_Bias, m_Delta, m_DeltaAth, m_Polite, m_S0, m_S1, m_T, m_V0, and CParameter::setID().

Referenced by CIDMParameterSet().

```
98 {
99         // Set each of the parameters of the IDM model
100         // Note appropriate default values are specified in the
101         // CDistribution(WORD id) constructor
102         m_T                     .setID(IDM_PARAM_T);
103         m_A                     .setID(IDM_PARAM_A);
104         m_B                     .setID(IDM_PARAM_B);
105         m_S0            .setID(IDM_PARAM_S0);
106         m_S1            .setID(IDM_PARAM_S1);
107         m_V0            .setID(IDM_PARAM_V0);
108         m_Delta         .setID(IDM_PARAM_DELTA);
109         m_Polite        .setID(IDM_PARAM_POLITE);
110         m_Bias          .setID(IDM_PARAM_BIAS);
111         m_DeltaAth      .setID(IDM_PARAM_DELTAATH);
112 }
```

Here is the call graph for this function:



4.23.4 Member Data Documentation

4.23.4.1 int CIDMParameterSet::m_VehicleTypeID [private]

Definition at line 75 of file IDMParameterSet.h.

Referenced by CIDMParameterSet(), getVehicleTypeID(), Serialize(), setAll(), and setVehicleTypeID().

4.23.4.2 CParameter CIDMParameterSet::m_T [private]

Definition at line 77 of file IDMParameterSet.h.

Referenced by buildParamSet(), Generate(), get_T(), Serialize(), set_T(), and setAll().

4.23.4.3 CParameter CIDMParameterSet::m_A [private]

Definition at line 78 of file IDMParameterSet.h.

Referenced by buildParamSet(), Generate(), get_A(), Serialize(), set_A(), and setAll().

4.23.4.4 CParameter CIDMParameterSet::m_B [private]

Definition at line 79 of file IDMParameterSet.h.

Referenced by buildParamSet(), Generate(), get_B(), Serialize(), set_B(), and setAll().

4.23.4.5 CParameter CIDMParameterSet::m_S0 [private]

Definition at line 80 of file IDMParameterSet.h.

Referenced by buildParamSet(), Generate(), get_S0(), Serialize(), set_S0(), and setAll().

4.23.4.6 CParameter CIDMParameterSet::m_S1 [private]

Definition at line 81 of file IDMParameterSet.h.

Referenced by buildParamSet(), Generate(), get_S1(), Serialize(), set_S1(), and setAll().

4.23.4.7 CParameter CIDMParameterSet::m_V0 [private]

Definition at line 82 of file IDMParameterSet.h.

Referenced by buildParamSet(), Generate(), get_V0(), Serialize(), set_V0(), and setAll().

4.23.4.8 CParameter CIDMParameterSet::m_Delta [private]

Definition at line 83 of file IDMParameterSet.h.

Referenced by buildParamSet(), Generate(), get_Delta(), Serialize(), set_Delta(), and setAll().

4.23.4.9 CParameter CIDMParameterSet::m_Polite [private]

Definition at line 84 of file IDMParameterSet.h.

Referenced by buildParamSet(), Generate(), get_Polite(), Serialize(), set_Polite(), and setAll().

4.23.4.10 CParameter CIDMParameterSet::m_Bias [private]

Definition at line 85 of file IDMParameterSet.h.

Referenced by buildParamSet(), Generate(), get_Bias(), Serialize(), set_Bias(), and setAll().

4.23.4.11 CParameter CIDMParameterSet::m_DeltaAth [private]

Definition at line 86 of file IDMParameterSet.h.

Referenced by buildParamSet(), Generate(), get_DeltaAth(), Serialize(), set_DeltaAth(), and setAll().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/IDMParameterSet.h
- D:/~Research/Code/C++/EvolveTraffic/IDMParameterSet.cpp

4.24 CMainFrame Class Reference

`#include <MainFrm.h>`

Public Member Functions

- virtual BOOL OnCreateClient (LPCREATESTRUCT lpcs, CCreateContext *pContext)
- virtual BOOL PreCreateWindow (CREATESTRUCT &cs)
- virtual ~CMainFrame ()

Protected Member Functions

- CMainFrame ()
- afx_msg int OnCreate (LPCREATESTRUCT lpCreateStruct)

Protected Attributes

- CSplitterWnd m_wndSplitter
- CStatusBar m_wndStatusBar
- CToolBar m_wndToolBar

4.24.1 Detailed Description

Definition at line 12 of file MainFrm.h.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 CMainFrame::CMainFrame () [protected]

Definition at line 37 of file MainFrm.cpp.

```
38 {
39
40 }
```

4.24.2.2 CMainFrame::~CMainFrame () [virtual]

Definition at line 42 of file MainFrm.cpp.

```
43 {
44 }
```

4.24.3 Member Function Documentation

4.24.3.1 BOOL CMainFrame::OnCreateClient (LPCREATESTRUCT *lpcs*, CCreateContext * *pContext*) [virtual]

Definition at line 76 of file MainFrm.cpp.

References m_wndSplitter.

```
78 {
79         return m_wndSplitter.Create(this,
80                 2, 2,               // TODO: adjust the number of rows, columns
81                 CSize(10, 10),      // TODO: adjust the minimum pane size
82                 pContext);
83 }
```

4.24.3.2 BOOL CMainFrame::PreCreateWindow (CREATESTRUCT & *cs*) [virtual]

Definition at line 85 of file MainFrm.cpp.

```
86 {
87         if( !CFrameWnd::PreCreateWindow(cs) )
88                 return FALSE;
89         // TODO: Modify the Window class or styles here by modifying
90         //  the CREATESTRUCT cs
91
92         return TRUE;
93 }
```

4.24.3.3 int CMainFrame::OnCreate (LPCREATESTRUCT *lpCreateStruct*) [protected]

Definition at line 46 of file MainFrm.cpp.

References IDR_MAINFRAME, indicators, m_wndStatusBar, and m_wndToolBar.

```
47 {
48         if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
49                 return -1;
50
51         if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD | WS_VISIBLE | CBRS_TOP
52                 | CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC) ||
53                 !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
54         {
55                 TRACE0("Failed to create toolbar\n");
56                 return -1;      // fail to create
57         }
58
59         if (!m_wndStatusBar.Create(this) ||
60                 !m_wndStatusBar.SetIndicators(indicators,
61                   sizeof(indicators)/sizeof(UINT)))
62         {
63                 TRACE0("Failed to create status bar\n");
64                 return -1;      // fail to create
65         }
66
67         // TODO: Delete these three lines if you don't want the toolbar to
68         //  be dockable
69         m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
70         EnableDocking(CBRS_ALIGN_ANY);
71         DockControlBar(&m_wndToolBar);
72
73         return 0;
74 }
```

4.24.4 Member Data Documentation

4.24.4.1 CSplitterWnd CMainFrame::m_wndSplitter [protected]

Definition at line 21 of file MainFrm.h.

Referenced by OnCreateClient().

4.24.4.2 CStatusBar CMainFrame::m_wndStatusBar [protected]

Definition at line 44 of file MainFrm.h.

Referenced by OnCreate().

4.24.4.3 CToolBar CMainFrame::m_wndToolBar [protected]

Definition at line 45 of file MainFrm.h.

Referenced by OnCreate().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/MainFrm.h
- D:/~Research/Code/C++/EvolveTraffic/MainFrm.cpp

4.25 CMemDC Class Reference

A class for flicker-free drawing in the window.

`#include <memdc.h>`

Public Member Functions

- CMemDC (CDC *pDC, const CRect *pRect=NULL)
- ~CMemDC ()
- CMemDC * operator → ()
- operator CMemDC * ()

Private Attributes

- CBitmap m_bitmap
- CBitmap * m_oldBitmap
- CDC * m_pDC
- CRect m_rect
- BOOL m_bMemDC

4.25.1 Detailed Description

A class for flicker-free drawing in the window.

Definition at line 31 of file memdc.h.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 CMemDC::CMemDC (CDC * *pDC*, const CRect * *pRect* = NULL) [inline]

Definition at line 40 of file memdc.h.

References m_bitmap, m_bMemDC, m_oldBitmap, m_pDC, and m_rect.

```
40                                                      : CDC()
41         {
42                 ASSERT(pDC != NULL);
43
44                 // Some initialization
45                 m_pDC = pDC;
46                 m_oldBitmap = NULL;
47                 m_bMemDC = !pDC->IsPrinting();
48
49                 // Get the rectangle to draw
50                 if (pRect == NULL) {
51                         pDC->GetClipBox(&m_rect);
52                 } else {
53                         m_rect = *pRect;
54                 }
55
```

```
56                 if (m_bMemDC) {
57                         // Create a Memory DC
58                         CreateCompatibleDC(pDC);
59                         pDC->LPtoDP(&m_rect);
60 
61                         m_bitmap.CreateCompatibleBitmap(pDC, m_rect.Width(), m_rect.Height());
62                         m_oldBitmap = SelectObject(&m_bitmap);
63 
64                         SetMapMode(pDC->GetMapMode());
65 
66                         SetWindowExt(pDC->GetWindowExt());
67                         SetViewportExt(pDC->GetViewportExt());
68 
69                         pDC->DPtoLP(&m_rect);
70                         SetWindowOrg(m_rect.left, m_rect.top);
71                 } else {
72                         // Make a copy of the relevent parts of the current DC for printing
73                         m_bPrinting = pDC->m_bPrinting;
74                         m_hDC       = pDC->m_hDC;
75                         m_hAttribDC = pDC->m_hAttribDC;
76                 }
77 
78                 // Fill background
79                 FillSolidRect(m_rect, pDC->GetBkColor());
80         }
```

4.25.2.2 CMemDC::~CMemDC () [inline]

Definition at line 82 of file memdc.h.

References m_bMemDC, m_oldBitmap, m_pDC, and m_rect.

```
83         {
84                 if (m_bMemDC) {
85                         // Copy the offscreen bitmap onto the screen.
86                         m_pDC->BitBlt(m_rect.left, m_rect.top, m_rect.Width(), m_rect.Height(),
87                                 this, m_rect.left, m_rect.top, SRCCOPY);
88 
89                         //Swap back the original bitmap.
90                         SelectObject(m_oldBitmap);
91                 } else {
92                         // All we need to do is replace the DC with an illegal value,
93                         // this keeps us from accidently deleting the handles associated with
94                         // the CDC that was passed to the constructor.
95                         m_hDC = m_hAttribDC = NULL;
96                 }
97         }
```

4.25.3 Member Function Documentation

4.25.3.1 CMemDC* CMemDC::operator → () [inline]

Definition at line 100 of file memdc.h.

```
101         {
102                  return this;
103         }
```

4.25.3.2 CMemDC::operator CMemDC * () [inline]

Definition at line 106 of file memdc.h.

```
107         {
108                 return this;
109         }
```

4.25.4 Member Data Documentation

4.25.4.1 CBitmap CMemDC::m_bitmap [private]

Definition at line 33 of file memdc.h.

Referenced by CMemDC().

4.25.4.2 CBitmap* CMemDC::m_oldBitmap [private]

Definition at line 34 of file memdc.h.

Referenced by CMemDC(), and ~CMemDC().

4.25.4.3 CDC* CMemDC::m_pDC [private]

Definition at line 35 of file memdc.h.

Referenced by CMemDC(), and ~CMemDC().

4.25.4.4 CRect CMemDC::m_rect [private]

Definition at line 36 of file memdc.h.

Referenced by CMemDC(), and ~CMemDC().

4.25.4.5 BOOL CMemDC::m_bMemDC [private]

Definition at line 37 of file memdc.h.

Referenced by CMemDC(), and ~CMemDC().

The documentation for this class was generated from the following file:

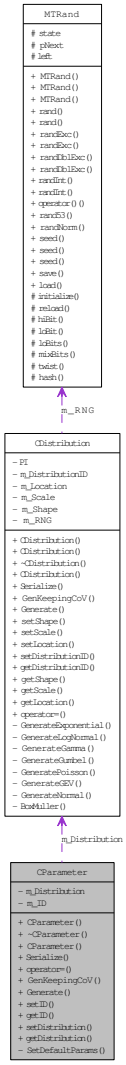
- D:/~Research/Code/C++/EvolveTraffic/memdc.h

4.26 CParameter Class Reference

A class represetning a single IDM parameter that can be saved to file.

`#include <Parameter.h>`

Collaboration diagram for CParameter:



Public Member Functions

- CParameter ()
- virtual ~CParameter ()
- CParameter (const CParameter &param)
- void Serialize (CArchive &ar)
- CParameter & operator= (const CParameter &param)
- double GenKeepingCoV (double newLocation)
- double Generate ()
- void setID (WORD id)
- WORD getID () const

- void setDistribution (CDistribution *dist)
- CDistribution * getDistribution ()

Private Member Functions

- void SetDefaultParams ()

Private Attributes

- CDistribution m_Distribution
- WORD m_ID

4.26.1 Detailed Description

A class represetning a single IDM parameter that can be saved to file.

Definition at line 17 of file Parameter.h.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 CParameter::CParameter ()

Definition at line 19 of file Parameter.cpp.

```
20 {
21
22 }
```

4.26.2.2 CParameter::~CParameter () [virtual]

Definition at line 24 of file Parameter.cpp.

```
25 {
26
27 }
```

4.26.2.3 CParameter::CParameter (const CParameter & *param*)

Definition at line 29 of file Parameter.cpp.

References getID(), m_Distribution, and m_ID.

```
30 {
31         this->m_Distribution = param.m_Distribution;
32         this->m_ID = param.getID();
33 }
```

Here is the call graph for this function:

4.26.3 Member Function Documentation

4.26.3.1 void CParameter::Serialize (CArchive & *ar*)

Definition at line 53 of file Parameter.cpp.

References m_Distribution, m_ID, and CDistribution::Serialize().

Referenced by CIDMParameterSet::Serialize().

```
54 {
55         m_Distribution.Serialize(ar);
56         if (ar.IsStoring())
57                 ar << m_ID;
58         else
59                 ar >> m_ID;
60 }
```

Here is the call graph for this function:

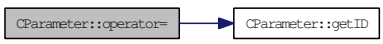
4.26.3.2 CParameter & CParameter::operator= (const CParameter & *param*)

Definition at line 140 of file Parameter.cpp.

References getID(), m_Distribution, and m_ID.

```
141 {
142         this->m_Distribution = param.m_Distribution;
143         this->m_ID = param.getID();
144
145         return *this;
146 }
```

Here is the call graph for this function:



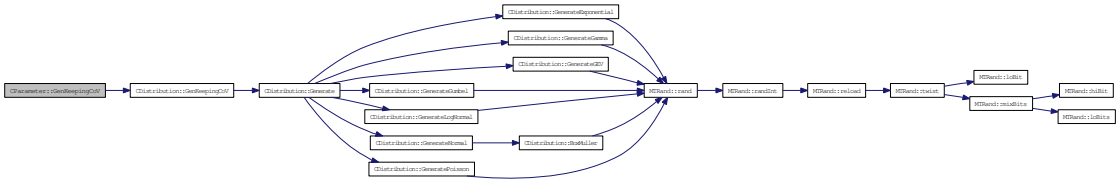
4.26.3.3 double CParameter::GenKeepingCoV (double *newLocation*)

Definition at line 175 of file Parameter.cpp.

References CDistribution::GenKeepingCoV(), and m_Distribution.

```
176 {
177         return m_Distribution.GenKeepingCoV(newLocation);
178 }
```

Here is the call graph for this function:



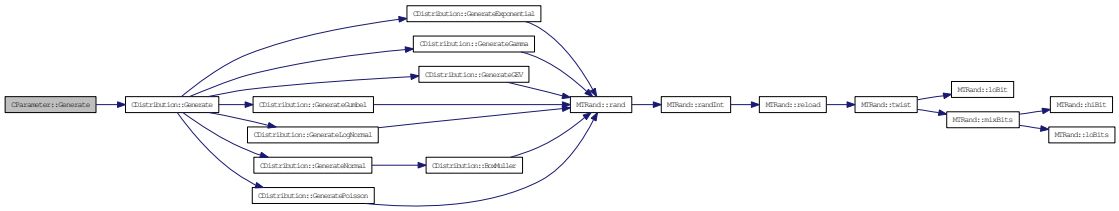
4.26.3.4 double CParameter::Generate ()

Definition at line 170 of file Parameter.cpp.

References CDistribution::Generate(), and m_Distribution.

```
171 {
172         return m_Distribution.Generate();
173 }
```

Here is the call graph for this function:



4.26.3.5 void CParameter::setID (WORD *id*)

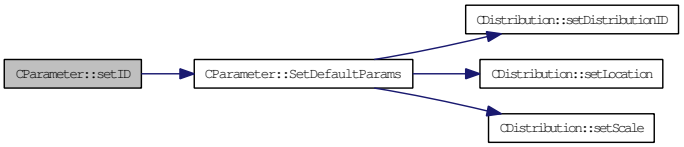
Definition at line 163 of file Parameter.cpp.

References m_ID, and SetDefaultParams().

Referenced by CIDMParameterSet::buildParamSet().

```
164 {
165         m_ID = id;
166         SetDefaultParams();
167 }
```

Here is the call graph for this function:



4.26.3.6 WORD CParameter::getID () const

Definition at line 158 of file Parameter.cpp.

References m_ID.

Referenced by CParameter(), and operator=().

```
159 {
160         return m_ID;
161 }
```

4.26.3.7 void CParameter::setDistribution (CDistribution * *dist*)

Definition at line 153 of file Parameter.cpp.

References m_Distribution.

```
154 {
155         m_Distribution = *dist;
156 }
```

4.26.3.8 CDistribution * CParameter::getDistribution ()

Definition at line 148 of file Parameter.cpp.

References m_Distribution.

Referenced by SpeedLimit::addVehicle(), CTrafficConfigDlg::LoadRow(), and CTrafficConfigDlg::SetParamData().

```
149 {
150         return &m_Distribution;
151 }
```

4.26.3.9 void CParameter::SetDefaultParams () [private]

Definition at line 65 of file Parameter.cpp.

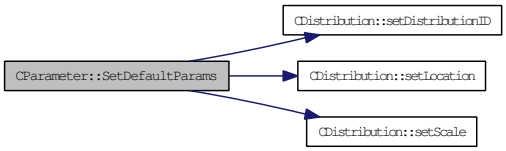
References DIST_CONST, IDM_PARAM_A, IDM_PARAM_B, IDM_PARAM_-BIAS, IDM_PARAM_DELTA, IDM_PARAM_DELTAATH, IDM_PARAM_-POLITE, IDM_PARAM_S0, IDM_PARAM_S1, IDM_PARAM_T, IDM_-PARAM_V0, m_Distribution, m_ID, CDistribution::setDistributionID(), CDistribution::setLocation(), and CDistribution::setScale().

Referenced by setID().

```
66 {
67         // Here we set the default IDM properties based on ID
68         // No set shapes since default distribvution is DIST_NORMAL
69         switch(m_ID)
70         {
71                 case IDM_PARAM_T:
72                         {
73                                 m_Distribution.setLocation(1.2);
74                                 m_Distribution.setScale(0.05);
75                         }
76                         break;
77                 case IDM_PARAM_A:
78                         {
79                                 m_Distribution.setLocation(1.0);
80                                 m_Distribution.setScale(0.05);
81                         }
82                         break;
83                 case IDM_PARAM_B:
84                         {
85                                 m_Distribution.setLocation(2.0);
86                                 m_Distribution.setScale(0.1);
87                         }
88                         break;
89                 case IDM_PARAM_S0:
90                         {
91                                 m_Distribution.setLocation(1.0);
92                                 m_Distribution.setScale(0.05);
93                         }
94                         break;
95                 case IDM_PARAM_S1:
96                         {
97                                 m_Distribution.setLocation(10.0);
98                                 m_Distribution.setScale(0.7);
99                         }
100                         break;
101                 case IDM_PARAM_V0:
102                         {       // Defined as km/h
103                                 m_Distribution.setLocation(80);
104                                 m_Distribution.setScale(3.0);
105                         }
106                         break;
107                 case IDM_PARAM_DELTA:
108                         {       // Leave as a constant
109                                 m_Distribution.setDistributionID(DIST_CONST);
110                                 m_Distribution.setLocation(4);
111                         }
112                         break;
113                 case IDM_PARAM_POLITE:
114                         {       // Leave as a constant
115                                 m_Distribution.setDistributionID(DIST_CONST);
116                                 m_Distribution.setLocation(0.0);
117                         }
118                         break;
```

```
119                 case IDM_PARAM_BIAS:
120                         {       // Leave as a constant
121                                 m_Distribution.setDistributionID(DIST_CONST);
122                                 m_Distribution.setLocation(0.1);
123                         }
124                         break;
125                 case IDM_PARAM_DELTAATH:
126                         {       // Leave as a constant
127                                 m_Distribution.setDistributionID(DIST_CONST);
128                                 m_Distribution.setLocation(0.3);
129                         }
130                         break;
131                 default: // IDM_PARAM_V0:
132                         {       // Defined as km/h
133                                 m_Distribution.setLocation(80);
134                                 m_Distribution.setScale(3.0);
135                         }
136                         break;
137         }
138 }
```

Here is the call graph for this function:



4.26.4 Member Data Documentation

4.26.4.1 CDistribution CParameter::m_Distribution [private]

Definition at line 51 of file Parameter.h.

Referenced by CParameter(), Generate(), GenKeepingCoV(), getDistribution(), operator=(), Serialize(), SetDefaultParams(), and setDistribution().

4.26.4.2 WORD CParameter::m_ID [private]

Definition at line 52 of file Parameter.h.

Referenced by CParameter(), getID(), operator=(), Serialize(), SetDefaultParams(), and setID().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/Parameter.h
- D:/~Research/Code/C++/EvolveTraffic/Parameter.cpp

4.27 CPreferencesDlg Class Reference

A class for the User Preferences Dialog.

`#include <PreferencesDlg.h>`

Public Types

- enum { IDD = IDD_PREFERENCES }

Public Member Functions

- CPreferencesDlg (CWnd *pParent=NULL)

Public Attributes

- double m_Scale
- double m_ExaggerateWidths
- double m_TimeWarp
- double m_LaneWidth
- double m_VehicleWidth
- double m_VehicleLengthScale
- double m_ShowVelocity
- double m_ShowLegend
- double m_TickLength
- double m_TickStep
- double m_Border_Top
- double m_Border_Btm
- double m_Border_Lhs
- double m_Border_Rhs
- CGridCtrl m_Grid

Protected Member Functions

- virtual void DoDataExchange (CDataExchange *pDX)
- virtual void OnOK ()

Private Member Functions

- BOOL OnInitDialog ()
- void SetGridHeadings ()
- void SetCells ()
- void LoadSettingsIntoGrid ()
- void OnGridEndEdit (NMHDR *pNotifyStruct, LRESULT *pResult)
- double * MapRowToValue (int row)

Private Attributes

- int m_nFixCols
- int m_nFixRows
- CStringArray m_sRowHeads
- CStringArray m_sColHeads
- int m_nCols
- int m_nRows
- double MIN_VIEW_SCALE

4.27.1 Detailed Description

A class for the User Preferences Dialog.

Definition at line 17 of file PreferencesDlg.h.

4.27.2 Member Enumeration Documentation

4.27.2.1 anonymous enum

Enumerator:

IDD

Definition at line 25 of file PreferencesDlg.h.

```
25 { IDD = IDD_PREFERENCES };
```

4.27.3 Constructor & Destructor Documentation

4.27.3.1 CPreferencesDlg::CPreferencesDlg (CWnd * *pParent* = NULL)

Definition at line 23 of file PreferencesDlg.cpp.

References FIXED_COLUMNS, FIXED_ROWS, m_nCols, m_nFixCols, m_nFixRows, m_nRows, m_sColHeads, m_sRowHeads, CConfigData::View_Config::MIN_VIEW_SCALE, MIN_VIEW_SCALE, and CConfigData::View.

```
24         : CDialog(CPreferencesDlg::IDD, pParent)
25 {
26         MIN_VIEW_SCALE = g_ConfigData.View.MIN_VIEW_SCALE;
27
28         m_sRowHeads.Add("Overall scale (meters/pixel)");
29         m_sRowHeads.Add("Widths Scale");
30         m_sRowHeads.Add("Time warp factor");
31         m_sRowHeads.Add("Lane width (m)");
32         m_sRowHeads.Add("Vehicle width (m)");
33         m_sRowHeads.Add("Vehicle Length Scale");
34         m_sRowHeads.Add("Show Velocities");
35         m_sRowHeads.Add("Show Legend");
36         m_sRowHeads.Add("Tick Length (m)");
```

```
37          m_sRowHeads.Add("Tick Interval (m)");
38          m_sRowHeads.Add("Border Top (m)");
39          m_sRowHeads.Add("Border Bottom (m)");
40          m_sRowHeads.Add("Border Left (m)");
41          m_sRowHeads.Add("Border Right (m)");
42
43          m_sColHeads.Add("Setting");
44          m_sColHeads.Add("Value");
45
46          m_nFixRows = FIXED_ROWS;        // MAGIC NUMBER - row and column headers
47          m_nFixCols = FIXED_COLUMNS;
48
49          m_nCols = 1 + m_nFixCols;       // MAGIC NUMBER - obviously 1 extra column
50          m_nRows = m_sRowHeads.GetSize() + m_nFixRows;
51
52          //{{AFX_DATA_INIT(CPreferencesDlg)
53                  // NOTE: the ClassWizard will add member initialization here
54          //}}AFX_DATA_INIT
55  }
```

4.27.4 Member Function Documentation

4.27.4.1 void CPreferencesDlg::DoDataExchange (CDataExchange * *pDX*) [protected, virtual]

Definition at line 58 of file PreferencesDlg.cpp.

References IDC_GRID, and m_Grid.

```
59  {
60          CDialog::DoDataExchange(pDX);
61          //{{AFX_DATA_MAP(CPreferencesDlg)
62          DDX_Control(pDX, IDC_GRID, m_Grid);             // associate the grid window with a C++
63          //}}AFX_DATA_MAP
64  }
```

4.27.4.2 void CPreferencesDlg::OnOK () [protected, virtual]

Definition at line 77 of file PreferencesDlg.cpp.

References m_ExaggerateWidths, m_LaneWidth, m_Scale, m_TimeWarp, m_VehicleWidth, and MIN_VIEW_SCALE.

```
78  {
79          CDialog::OnOK();
80          if(m_Scale < MIN_VIEW_SCALE)
81          {
82                  MessageBox("View scale cannot be less than minimum - default minimum set", "Evol
83                  m_Scale = MIN_VIEW_SCALE;
84          }
85          if(m_ExaggerateWidths < 1.0)
86          {
87                  MessageBox("Widths scale cannot be less than 1.0 - default set", "EvolveTraffic"
88                  m_ExaggerateWidths = 1.0;
89          }
90          if(m_TimeWarp < 1.0)
91          {
```

```
92                 MessageBox("Widths scale cannot be less than 1.0 - default set", "EvolveTraffic"
93                 m_ExaggerateWidths = 1.0;
94         }
95         if(m_VehicleWidth > m_LaneWidth)
96         {
97                 MessageBox("Vehicle width should be less than lane width - default of 80% of lan
98                 m_VehicleWidth = 0.8 * m_LaneWidth;     // MAGIC NUMBER - no harm having a magic
99         }
100 }
```

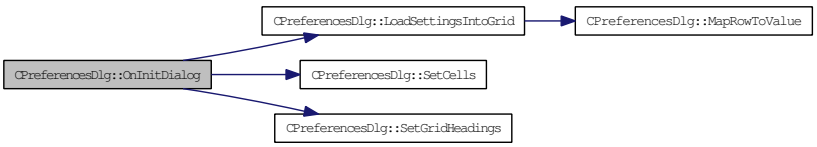
4.27.4.3 BOOL CPreferencesDlg::OnInitDialog () [private]

Definition at line 102 of file PreferencesDlg.cpp.

References LoadSettingsIntoGrid(), m_Grid, m_nCols, m_nFixCols, m_nFixRows, m_nRows, SetCells(), and SetGridHeadings().

```
103 {
104         CDialog::OnInitDialog();
105
106         TRY {
107                 m_Grid.SetRowCount(m_nRows);
108                 m_Grid.SetColumnCount(m_nCols);
109                 m_Grid.SetFixedRowCount(m_nFixRows);
110                 m_Grid.SetFixedColumnCount(m_nFixCols);
111         }
112         CATCH (CMemoryException, e)
113         {
114                 e->ReportError();
115                 return FALSE;
116         }
117         END_CATCH
118
119         SetGridHeadings();
120         SetCells();
121         m_Grid.ExpandToFit();
122
123         LoadSettingsIntoGrid();
124
125         return TRUE;  // return TRUE  unless you set the focus to a control
126 }
```

Here is the call graph for this function:



4.27.4.4 void CPreferencesDlg::SetGridHeadings () [private]

Definition at line 128 of file PreferencesDlg.cpp.

References m_Grid, m_nFixCols, m_sColHeads, and m_sRowHeads.

Referenced by OnInitDialog().

```
129 {
130         int row = 0;
131         int col = 0;
132
133         // Set fixed column text
134         m_Grid.SetItemText(row,m_nFixCols-1,m_sColHeads.GetAt(0));
135         m_Grid.SetItemText(row,m_nFixCols,m_sColHeads.GetAt(1));
136
137         // Set fixed row text
138         col = 0;
139         for (row = 1; row < m_Grid.GetRowCount(); row++)
140                 m_Grid.SetItemText(row,col,m_sRowHeads.GetAt(row-1));
141 }
```

4.27.4.5 void CPreferencesDlg::SetCells () [private]

Definition at line 143 of file PreferencesDlg.cpp.

References m_Grid, and m_nRows.

Referenced by OnInitDialog().

```
144 {
145         for (int row = 1; row < m_nRows; row++)
146         {
147                 int col = 1;
148                 m_Grid.SetCellType(row,col, RUNTIME_CLASS(CGridCellNumeric));
149                 m_Grid.GetCell(row,col)->SetFormat(DT_CENTER|DT_VCENTER|DT_SINGLELINE|DT_NOPRE
150                 CGridCellNumeric *pCell = (CGridCellNumeric*)m_Grid.GetCell(row,col);
151                 pCell->SetFlags(CGridCellNumeric::Real);        // no negative numbers
152         }
153 }
```

4.27.4.6 void CPreferencesDlg::LoadSettingsIntoGrid () [private]

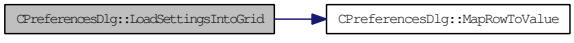
Definition at line 155 of file PreferencesDlg.cpp.

References m_Grid, m_nRows, and MapRowToValue().

Referenced by OnInitDialog().

```
156 {
157         int col = 1;
158         for (int row = 1; row < m_nRows; row++)
159         {
160                 CGridCellNumeric* pCell = (CGridCellNumeric *)(m_Grid.GetCell(row,col));
161                 pCell->SetNumber( *(MapRowToValue(row)) );
162         }
163 }
```

Here is the call graph for this function:



4.27.4.7 void CPreferencesDlg::OnGridEndEdit (NMHDR * *pNotifyStruct*, LRESULT * *pResult*) [private]

Definition at line 165 of file PreferencesDlg.cpp.

References m_Grid, and MapRowToValue().

```
166 {
167         // if change is ok, then *pResult 0, else *pResult -1
168
169         // pItem is the cell that has just been edited
170     NM_GRIDVIEW* pItem = (NM_GRIDVIEW*) pNotifyStruct;
171
172     int row = pItem->iRow;
173         int col = pItem->iColumn;
174
175         double val = ((CGridCellNumeric *)(m_Grid.GetCell(row,col)))->GetNumber();
176         *(MapRowToValue(row)) = val;
177
178         *pResult = 0;
179 }
```

Here is the call graph for this function:

4.27.4.8 double * CPreferencesDlg::MapRowToValue (int *row*) [private]

Definition at line 181 of file PreferencesDlg.cpp.

References m_Border_Btm, m_Border_Lhs, m_Border_Rhs, m_Border_Top, m_ExaggerateWidths, m_LaneWidth, m_nFixRows, m_Scale, m_ShowLegend, m_ShowVelocity, m_TickLength, m_TickStep, m_TimeWarp, m_VehicleLengthScale, and m_VehicleWidth.

Referenced by LoadSettingsIntoGrid(), and OnGridEndEdit().

```
182 {
183         // This functions assumes the variable order as per m_sRowHeads
184
185         row = row - m_nFixRows; // keeps independence of m_nFixRows
186         switch(row)
187         {
188                 case 0:
189                         return &m_Scale;
190                 case 1:
191                         return &m_ExaggerateWidths;
192                 case 2:
193                         return &m_TimeWarp;
194                 case 3:
195                         return &m_LaneWidth;
196                 case 4:
197                         return &m_VehicleWidth;
198                 case 5:
199                         return &m_VehicleLengthScale;
```

```
200                 case 6:
201                         return &m_ShowVelocity;
202                 case 7:
203                         return &m_ShowLegend;
204                 case 8:
205                         return &m_TickLength;
206                 case 9:
207                         return &m_TickStep;
208                 case 10:
209                         return &m_Border_Top;
210                 case 11:
211                         return &m_Border_Btm;
212                 case 12:
213                         return &m_Border_Lhs;
214                 case 13:
215                         return &m_Border_Rhs;
216                 default:
217                         return &m_Border_Top;
218         }
219 }
```

4.27.5 Member Data Documentation

4.27.5.1 double CPreferencesDlg::m_Scale

Definition at line 26 of file PreferencesDlg.h.

Referenced by MapRowToValue(), OnOK(), and CEvolveTrafficView::OnToolsPrefs().

4.27.5.2 double CPreferencesDlg::m_ExaggerateWidths

Definition at line 27 of file PreferencesDlg.h.

Referenced by MapRowToValue(), OnOK(), and CEvolveTrafficView::OnToolsPrefs().

4.27.5.3 double CPreferencesDlg::m_TimeWarp

Definition at line 28 of file PreferencesDlg.h.

Referenced by MapRowToValue(), OnOK(), and CEvolveTrafficView::OnToolsPrefs().

4.27.5.4 double CPreferencesDlg::m_LaneWidth

Definition at line 29 of file PreferencesDlg.h.

Referenced by MapRowToValue(), OnOK(), and CEvolveTrafficView::OnToolsPrefs().

4.27.5.5 double CPreferencesDlg::m_VehicleWidth

Definition at line 30 of file PreferencesDlg.h.

Referenced by MapRowToValue(), OnOK(), and CEvolveTrafficView::OnToolsPrefs().

4.27.5.6 double CPreferencesDlg::m_VehicleLengthScale

Definition at line 31 of file PreferencesDlg.h.

Referenced by MapRowToValue(), and CEvolveTrafficView::OnToolsPrefs().

4.27.5.7 double CPreferencesDlg::m_ShowVelocity

Definition at line 32 of file PreferencesDlg.h.

Referenced by MapRowToValue(), and CEvolveTrafficView::OnToolsPrefs().

4.27.5.8 double CPreferencesDlg::m_ShowLegend

Definition at line 33 of file PreferencesDlg.h.

Referenced by MapRowToValue(), and CEvolveTrafficView::OnToolsPrefs().

4.27.5.9 double CPreferencesDlg::m_TickLength

Definition at line 34 of file PreferencesDlg.h.

Referenced by MapRowToValue(), and CEvolveTrafficView::OnToolsPrefs().

4.27.5.10 double CPreferencesDlg::m_TickStep

Definition at line 35 of file PreferencesDlg.h.

Referenced by MapRowToValue(), and CEvolveTrafficView::OnToolsPrefs().

4.27.5.11 double CPreferencesDlg::m_Border_Top

Definition at line 36 of file PreferencesDlg.h.

Referenced by MapRowToValue(), and CEvolveTrafficView::OnToolsPrefs().

4.27.5.12 double CPreferencesDlg::m_Border_Btm

Definition at line 37 of file PreferencesDlg.h.

Referenced by MapRowToValue(), and CEvolveTrafficView::OnToolsPrefs().

4.27.5.13 double CPreferencesDlg::m_Border_Lhs

Definition at line 38 of file PreferencesDlg.h.

Referenced by MapRowToValue(), and CEvolveTrafficView::OnToolsPrefs().

4.27.5.14 double CPreferencesDlg::m_Border_Rhs

Definition at line 39 of file PreferencesDlg.h.

Referenced by MapRowToValue(), and CEvolveTrafficView::OnToolsPrefs().

4.27.5.15 CGridCtrl CPreferencesDlg::m_Grid

Definition at line 41 of file PreferencesDlg.h.

Referenced by DoDataExchange(), LoadSettingsIntoGrid(), OnGridEndEdit(), OnInitDialog(), SetCells(), and SetGridHeadings().

4.27.5.16 int CPreferencesDlg::m_nFixCols [private]

Definition at line 59 of file PreferencesDlg.h.

Referenced by CPreferencesDlg(), OnInitDialog(), and SetGridHeadings().

4.27.5.17 int CPreferencesDlg::m_nFixRows [private]

Definition at line 60 of file PreferencesDlg.h.

Referenced by CPreferencesDlg(), MapRowToValue(), and OnInitDialog().

4.27.5.18 CStringArray CPreferencesDlg::m_sRowHeads [private]

Definition at line 61 of file PreferencesDlg.h.

Referenced by CPreferencesDlg(), and SetGridHeadings().

4.27.5.19 CStringArray CPreferencesDlg::m_sColHeads [private]

Definition at line 62 of file PreferencesDlg.h.

Referenced by CPreferencesDlg(), and SetGridHeadings().

4.27.5.20 int CPreferencesDlg::m_nCols [private]

Definition at line 63 of file PreferencesDlg.h.

Referenced by CPreferencesDlg(), and OnInitDialog().

4.27.5.21 int CPreferencesDlg::m_nRows [private]

Definition at line 64 of file PreferencesDlg.h.

Referenced by CPreferencesDlg(), LoadSettingsIntoGrid(), OnInitDialog(), and SetCells().

4.27.5.22 double CPreferencesDlg::MIN_VIEW_SCALE [private]

Definition at line 73 of file PreferencesDlg.h.

Referenced by CPreferencesDlg(), and OnOK().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/PreferencesDlg.h
- D:/~Research/Code/C++/EvolveTraffic/PreferencesDlg.cpp

4.28 CRoadFeature Class Reference

A class for storing general Road Feature objects to file.

`#include <RoadFeature.h>`

Public Member Functions

- CRoadFeature ()
- virtual ~CRoadFeature ()
- void Serialize (CArchive &ar)
- WORD getType ()
- bool getDirPos ()
- int getStart ()
- int getEnd ()
- double getValue ()
- void setType (WORD type)
- void setDirPos (bool dirpos)
- void setStart (int start)
- void setEnd (int end)
- void setValue (double val)

Private Attributes

- double m_Value
- int m_End
- int m_Start
- bool m_DirPos
- WORD m_Type

4.28.1 Detailed Description

A class for storing general Road Feature objects to file.

Definition at line 14 of file RoadFeature.h.

4.28.2 Constructor & Destructor Documentation

4.28.2.1 CRoadFeature::CRoadFeature ()

Definition at line 21 of file RoadFeature.cpp.

References FEAT_SPEEDLIMIT.

```
22 {
23         m_Type          = FEAT_SPEEDLIMIT;
24         m_DirPos        = true;
25         m_Start         = 0;
26         m_End           = 1000;
27         m_Value         = 41.7; //  in m/s - about 150 km/h!
28 }
```

4.28.2.2 CRoadFeature::~CRoadFeature () [virtual]

Definition at line 30 of file RoadFeature.cpp.

```
31 {
32
33 }
```

4.28.3 Member Function Documentation

4.28.3.1 void CRoadFeature::Serialize (CArchive & *ar*)

Definition at line 35 of file RoadFeature.cpp.

References m_DirPos, m_End, m_Start, m_Type, and m_Value.

```
36 {
37     if (ar.IsStoring())
38     {
39         ar      << m_Type
40                         << static_cast<int>(m_DirPos)   // because I read in an int below
41                         << m_Start
42                         << m_End
43                         << m_Value;
44     }
45     else
46         {
47                 int temp;
48         ar      >> m_Type
49                         >> temp
50                         >> m_Start
51                         >> m_End
52                         >> m_Value;
53
54                 m_DirPos = temp == 0 ? false : true;
55         }
56 }
```

4.28.3.2 WORD CRoadFeature::getType ()

Definition at line 58 of file RoadFeature.cpp.

References m_Type.

Referenced by CEvolveTrafficView::DrawRoadSegments(), CRoadFeaturesDlg::LoadFeaturesIntoGrid(), CRoadFeaturesDlg::OnValidate(), CRoadFeaturesDlg::SetParamData(), and Road::SetSegmentFromFeature().

```
59 {
60         return m_Type;
61 }
```

4.28.3.3 bool CRoadFeature::getDirPos ()

Definition at line 63 of file RoadFeature.cpp.

References m_DirPos.

Referenced by CEvolveTrafficView::DrawRoadSegments(), CRoadFeaturesDlg::LoadFeaturesIntoGrid(), CRoadFeaturesDlg::OnValidate(), Road::SetGradientSegment(), and Road::SetSpeedLimitSegment().

```
64 {
65         return m_DirPos;
66 }
```

4.28.3.4 int CRoadFeature::getStart ()

Definition at line 68 of file RoadFeature.cpp.

References m_Start.

Referenced by CEvolveTrafficView::DrawRoadSegments(), CRoadFeaturesDlg::LoadFeaturesIntoGrid(), CRoadFeaturesDlg::OnValidate(), Road::SetGradientSegment(), Road::SetSegmentFromFeature(), and Road::SetSpeedLimitSegment().

```
69 {
70         return m_Start;
71 }
```

4.28.3.5 int CRoadFeature::getEnd ()

Definition at line 73 of file RoadFeature.cpp.

References m_End.

Referenced by CEvolveTrafficView::DrawRoadSegments(), CRoadFeaturesDlg::LoadFeaturesIntoGrid(), CRoadFeaturesDlg::OnValidate(), Road::SetGradientSegment(), Road::SetSegmentFromFeature(), and Road::SetSpeedLimitSegment().

```
74 {
75         return m_End;
76 }
```

4.28.3.6 double CRoadFeature::getValue ()

Definition at line 78 of file RoadFeature.cpp.

References m_Value.

Referenced by CEvolveTrafficView::DrawRoadSegments(), CRoadFeaturesDlg::LoadFeaturesIntoGrid(), CRoadFeaturesDlg::OnValidate(), Road::SetGradientSegment(), and Road::SetSpeedLimitSegment().

```
79 {
80         return m_Value;
81 }
```

4.28.3.7 void CRoadFeature::setType (WORD *type*)

Definition at line 83 of file RoadFeature.cpp.

References m_Type.

Referenced by CRoadFeaturesDlg::SetParamData().

```
84 {
85         m_Type = type;
86 }
```

4.28.3.8 void CRoadFeature::setDirPos (bool *dirpos*)

Definition at line 88 of file RoadFeature.cpp.

References m_DirPos.

Referenced by CRoadFeaturesDlg::SetParamData().

```
89 {
90         m_DirPos = dirpos;
91 }
```

4.28.3.9 void CRoadFeature::setStart (int *start*)

Definition at line 93 of file RoadFeature.cpp.

References m_Start.

Referenced by CRoadFeaturesDlg::OnValidate(), and CRoadFeaturesDlg::SetParamData().

```
94 {
95         m_Start = start;
96 }
```

4.28.3.10 void CRoadFeature::setEnd (int *end*)

Definition at line 98 of file RoadFeature.cpp.

References m_End.

Referenced by CRoadFeaturesDlg::OnValidate(), and CRoadFeaturesDlg::SetParamData().

```
99 {
100         m_End = end;
101 }
```

4.28.3.11 void CRoadFeature::setValue (double *val*)

Definition at line 103 of file RoadFeature.cpp.

References m_Value.

Referenced by CRoadFeaturesDlg::SetParamData().

```
104 {
105         m_Value = val;
106 }
```

4.28.4 Member Data Documentation

4.28.4.1 double CRoadFeature::m_Value [private]

Definition at line 39 of file RoadFeature.h.

Referenced by getValue(), Serialize(), and setValue().

4.28.4.2 int CRoadFeature::m_End [private]

Definition at line 40 of file RoadFeature.h.

Referenced by getEnd(), Serialize(), and setEnd().

4.28.4.3 int CRoadFeature::m_Start [private]

Definition at line 41 of file RoadFeature.h.

Referenced by getStart(), Serialize(), and setStart().

4.28.4.4 bool CRoadFeature::m_DirPos [private]

Definition at line 42 of file RoadFeature.h.

Referenced by getDirPos(), Serialize(), and setDirPos().

4.28.4.5 WORD CRoadFeature::m_Type [private]

Definition at line 43 of file RoadFeature.h.

Referenced by getType(), Serialize(), and setType().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/RoadFeature.h
- D:/~Research/Code/C++/EvolveTraffic/RoadFeature.cpp

4.29 CRoadFeaturesDlg Class Reference

A class for the Road Features Dialog.

`#include <RoadFeaturesDlg.h>`

Public Types

- enum { IDD = IDD_ROADFEATURES }

Public Member Functions

- CRoadFeaturesDlg (CWnd *pParent=NULL)
- virtual void OnValidate (UINT &nID, CString &strMessage)

Public Attributes

- int m_RoadLength
- int m_nFixCols
- int m_nFixRows
- int m_nCols
- int m_nRows
- CObArray m_vRoadFeatures
- CGridCtrl m_Grid

Protected Member Functions

- virtual void DoDataExchange (CDataExchange *pDX)
- virtual BOOL OnInitDialog ()
- afx_msg void OnBtnAddfeat ()
- afx_msg void OnBtnDelfeat ()
- void OnGridEndEdit (NMHDR *pNotifyStruct, LRESULT *pResult)

Private Member Functions

- WORD MapStringToType (CString str)
- void SetParamData (int row, int col, double val)
- void SetParamData (int row, int col, CString str)
- CString MapTypeToString (WORD type)
- void LoadFeaturesIntoGrid ()
- void SetCells (bool bAddDelete)
- void SetGridHeadings ()

Private Attributes

- int m_NoFeatures
- CStringArray m_sDirections
- CStringArray m_sFeatureTypes
- CStringArray m_sColumnHeaders
- int MAX_SLOPE
- int SPEEDLIMIT_MIN
- int SPEEDLIMIT_MAX

4.29.1 Detailed Description

A class for the Road Features Dialog.

Definition at line 19 of file RoadFeaturesDlg.h.

4.29.2 Member Enumeration Documentation

4.29.2.1 anonymous enum

Enumerator:

IDD

Definition at line 28 of file RoadFeaturesDlg.h.

```
28 { IDD = IDD_ROADFEATURES };
```

4.29.3 Constructor & Destructor Documentation

4.29.3.1 CRoadFeaturesDlg::CRoadFeaturesDlg (CWnd * *pParent* = NULL)

Definition at line 25 of file RoadFeaturesDlg.cpp.

References FIXED_COLUMNS, FIXED_ROWS, CConfigData::Road_-Config::RoadFeatures_Config::Gradient, m_nCols, m_nFixCols, m_-nFixRows, m_sColumnHeaders, m_sDirections, m_sFeatureTypes,

CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config::MAX_-SLOPE, MAX_SLOPE, CConfigData::Road, CConfigData::Road_-Config::RoadFeatures, CConfigData::Road_Config::RoadFeatures_-Config::SpeedLimit, CConfigData::Road_Config::RoadFeatures_-Config::SpeedLimit_Config::SPEEDLIMIT_MAX, SPEEDLIMIT_-MAX, CConfigData::Road_Config::RoadFeatures_Config::SpeedLimit_-Config::SPEEDLIMIT_MIN, and SPEEDLIMIT_MIN.

```
26          : CDialogExt(CRoadFeaturesDlg::IDD, pParent)
27 {
28          //{{AFX_DATA_INIT(CRoadFeaturesDlg)
29                  // NOTE: the ClassWizard will add member initialization here
30          //}}AFX_DATA_INIT
31
32          MAX_SLOPE               = g_ConfigData.Road.RoadFeatures.Gradient.MAX_SLOPE;
33          SPEEDLIMIT_MIN  = g_ConfigData.Road.RoadFeatures.SpeedLimit.SPEEDLIMIT_MIN;
34          SPEEDLIMIT_MAX  = g_ConfigData.Road.RoadFeatures.SpeedLimit.SPEEDLIMIT_MAX;
35
36          m_sFeatureTypes.Add("Speed Limit");
37          m_sFeatureTypes.Add("Gradient");
38
39          m_sDirections.Add("Positive x");
40          m_sDirections.Add("Negative x");
41
42          m_sColumnHeaders.Add("Type");
43          m_sColumnHeaders.Add("Direction");
44          m_sColumnHeaders.Add("Start (m)");
45          m_sColumnHeaders.Add("End (m)");
46          m_sColumnHeaders.Add("Value");
47
48          m_nFixCols = FIXED_COLUMNS;
49          m_nFixRows = FIXED_ROWS;
50          m_nCols = m_sColumnHeaders.GetSize() + m_nFixCols;
51 }
```

4.29.4 Member Function Documentation

4.29.4.1 void CRoadFeaturesDlg::OnValidate (UINT & *nID*, CString & *strMessage*) [virtual]

Definition at line 103 of file RoadFeaturesDlg.cpp.

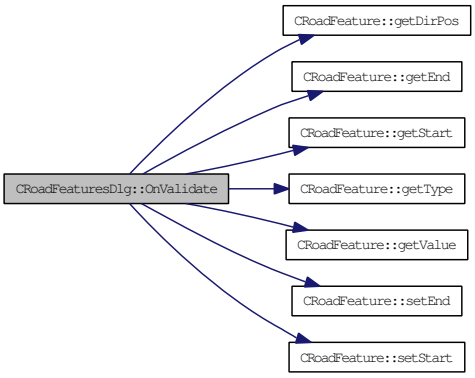
References FEAT_GRADIENT, FEAT_SPEEDLIMIT, CRoadFeature::getDirPos(), CRoadFeature::getEnd(), CRoadFeature::getStart(), CRoadFeature::getType(), CRoadFeature::getValue(), IDC_GRID, M_PER_S_TO_KM_PER_H, m_-RoadLength, m_vRoadFeatures, MAX_SLOPE, CRoadFeature::setEnd(), CRoadFeature::setStart(), SPEEDLIMIT_MAX, and SPEEDLIMIT_MIN.

```
104 {
105          int nFeats = m_vRoadFeatures.GetSize();
106
107          for(int i = 0; i < nFeats;  i++)
108          {
109                  CRoadFeature *pFeat = reinterpret_cast<CRoadFeature*>(m_vRoadFeatures.GetAt(i))
110
111                  // In case the start specified is after the end, swap them
112                  int FeatStart = pFeat->getStart();
```

```
113                 int FeatEnd = pFeat->getEnd();
114                 if(FeatStart > FeatEnd)
115                 {
116                         pFeat->setStart(FeatEnd);
117                         pFeat->setEnd(FeatStart);
118                         int temp = FeatStart; FeatStart = FeatEnd; FeatEnd = temp;      // for
119                 }
120
121                 // check on bounds
122                 if(FeatStart > m_RoadLength || FeatEnd > m_RoadLength)
123                 {
124                         strMessage.Format("Feature %d: Starts or ends outside the road length."
125                         nID = IDC_GRID;
126                         return;
127                 }
128
129                 int FeatType = pFeat->getType();        // for further use
130                 // check on values
131                 if(FeatType == FEAT_GRADIENT && pFeat->getValue() > MAX_SLOPE)
132                 {
133                         strMessage.Format("Feature %d: Gradient specified is greater \nthan the
134                         nID = IDC_GRID;
135                         return;
136                 }
137
138                 double speed_limit = pFeat->getValue()*M_PER_S_TO_KM_PER_H;
139                 if(FeatType == FEAT_SPEEDLIMIT && speed_limit < SPEEDLIMIT_MIN)
140                 {
141                         strMessage.Format("Feature %d: Speed limit specified is smaller \nthan
142                         nID = IDC_GRID;
143                         return;
144                 }
145
146                 if(FeatType == FEAT_SPEEDLIMIT && speed_limit > SPEEDLIMIT_MAX)
147                 {
148                         strMessage.Format("Feature %d: Speed limit specified is greater \nthan
149                         nID = IDC_GRID;
150                         return;
151                 }
152
153                 bool FeatDirPos = pFeat->getDirPos();
154                 // check on overlaps
155                 for(int j = 0; j < nFeats; j++) // because they are not sorted
156                 {
157                         if( j != i)     // better than j++ which causes bounds problem
158                         {
159                                 CRoadFeature *pFeatJ = reinterpret_cast<CRoadFeature*>(m_vRoadF
160
161                                 // if the same type and same direction
162                                 if(FeatType == pFeatJ->getType() && FeatDirPos == pFeatJ->getDi
163                                 {
164                                         int FeatJStart = pFeatJ->getStart();
165                                         int FeatJEnd = pFeatJ->getEnd();
166                                         if( (FeatStart > FeatJStart && FeatEnd < FeatJEnd) ||
167                                                 (FeatJStart > FeatStart && FeatJStart < FeatEnd
168                                         {
169                                                 strMessage.Format("Feature %d overlaps with Fea
170                                                 nID = IDC_GRID;
171                                                 return;
172                                         }
173                                 }
174                         } // end of if j != i
```

```
175                 }
176         } // end of main for loop
177
178 }
```

Here is the call graph for this function:



4.29.4.2 void CRoadFeaturesDlg::DoDataExchange (CDataExchange * *pDX*) [protected, virtual]

Definition at line 53 of file RoadFeaturesDlg.cpp.

References IDC_GRID, and m_Grid.

```
54 {
55         CDialogExt::DoDataExchange(pDX);
56         //{{AFX_DATA_MAP(CRoadFeaturesDlg)
57         DDX_Control(pDX, IDC_GRID, m_Grid);             // associate the grid window with a C++
58         //}}AFX_DATA_MAP
59 }
```

4.29.4.3 BOOL CRoadFeaturesDlg::OnInitDialog () [protected, virtual]

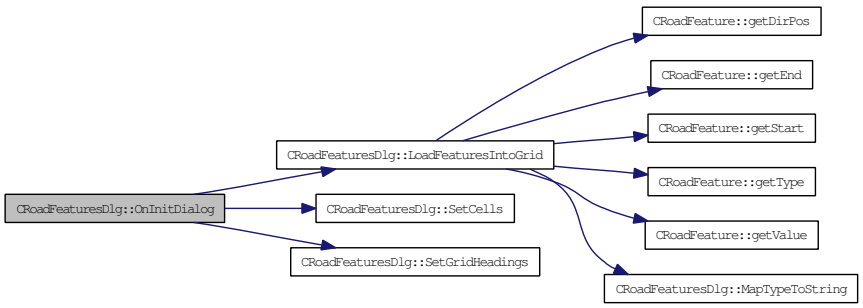
Definition at line 73 of file RoadFeaturesDlg.cpp.

References LoadFeaturesIntoGrid(), m_Grid, m_nCols, m_nFixCols, m_nFixRows, m_NoFeatures, m_nRows, m_vRoadFeatures, SetCells(), and SetGridHeadings().

```
74 {
75         CDialogExt::OnInitDialog();
76
77         // init rows
78         m_NoFeatures = m_vRoadFeatures.GetSize();
79         m_nRows = m_nFixRows + m_NoFeatures;
80
81         TRY {
82                 m_Grid.SetRowCount(m_nRows);
```

```
83                 m_Grid.SetColumnCount(m_nCols);
84                 m_Grid.SetFixedRowCount(m_nFixRows);
85                 m_Grid.SetFixedColumnCount(m_nFixCols);
86         }
87         CATCH (CMemoryException, e)
88         {
89                 e->ReportError();
90                 return FALSE;
91         }
92         END_CATCH
93
94         SetGridHeadings();
95         SetCells(false);        // false because not addition or detletion
96         m_Grid.ExpandColumnsToFit();
97
98         LoadFeaturesIntoGrid();
99
100         return TRUE;  // return TRUE  unless you set the focus to a control
101 }
```

Here is the call graph for this function:



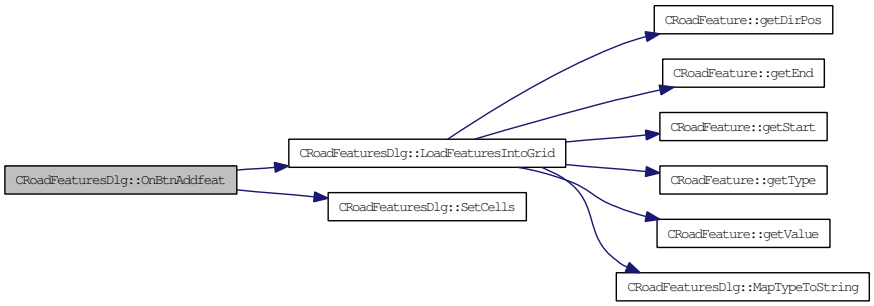
4.29.4.4 void CRoadFeaturesDlg::OnBtnAddfeat () [protected]

Definition at line 277 of file RoadFeaturesDlg.cpp.

References LoadFeaturesIntoGrid(), m_NoFeatures, m_vRoadFeatures, and SetCells().

```
278 {
279         // Add the feature to the vector
280         m_NoFeatures++;
281         CRoadFeature* pFeat = new CRoadFeature; // create one to use
282         m_vRoadFeatures.Add(pFeat);
283
284         // add a row to the table
285         SetCells(true);
286         LoadFeaturesIntoGrid();
287 }
```

Here is the call graph for this function:



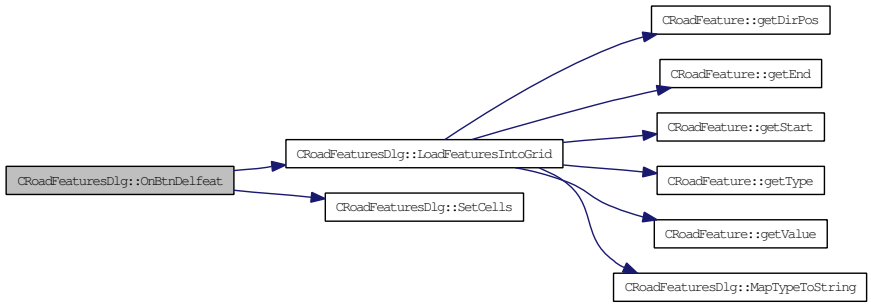
4.29.4.5 void CRoadFeaturesDlg::OnBtnDelfeat () [protected]

Definition at line 289 of file RoadFeaturesDlg.cpp.

References LoadFeaturesIntoGrid(), m_NoFeatures, m_vRoadFeatures, and SetCells().

```
290 {
291         if(m_NoFeatures != 0)
292         {
293                 // Get the ptr and remove from the vector
294                 CRoadFeature* pFeat = reinterpret_cast<CRoadFeature*>(m_vRoadFeatures.GetAt(m_N
295                 m_vRoadFeatures.RemoveAt( m_NoFeatures-1 );
296
297                 delete pFeat;   // remove feature from heap
298                 m_NoFeatures--; // decrement only now
299
300                 // update the table
301                 SetCells(true);
302                 LoadFeaturesIntoGrid();
303         }
304         else
305                 MessageBox("No road feature to delete.", "EvolveTraffic", MB_OK|MB_ICONWARNING)
306 }
```

Here is the call graph for this function:



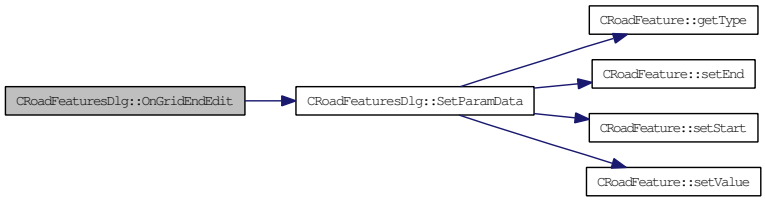
4.29.4.6 void CRoadFeaturesDlg::OnGridEndEdit (NMHDR * *pNotifyStruct*, LRESULT * *pResult*) [protected]

Definition at line 309 of file RoadFeaturesDlg.cpp.

References m_Grid, m_nFixRows, and SetParamData().

```
310 {
311         // if change is ok, then *pResult 0, else *pResult -1
312
313         // pItem is the cell that has just been edited
314     NM_GRIDVIEW* pItem = (NM_GRIDVIEW*) pNotifyStruct;
315
316     int row = pItem->iRow;      int col = pItem->iColumn;
317
318         if(col < 2 + m_nFixRows )       // is the type or direction column
319         {
320                 CString strCell = m_Grid.GetCell(row,col)->GetText();
321                 if(strCell != "")
322                 {
323                         // data is ok
324                         SetParamData(row,col,strCell);
325                         *pResult = 0;
326                 }
327                 else
328                         *pResult = -1;
329         }
330         else
331         {
332                 // no need to verify CNumericCells since will always be a number
333                 // even deleting a cell's data results in a zero
334                 double val = ((CGridCellNumeric *)(m_Grid.GetCell(row,col)))->GetNumber();
335                 SetParamData(row,col,val);
336                 *pResult = 0;
337         }
338
339 }
```

Here is the call graph for this function:



4.29.4.7 WORD CRoadFeaturesDlg::MapStringToType (CString *str*) [private]

Definition at line 392 of file RoadFeaturesDlg.cpp.

References FEAT_GRADIENT, FEAT_SPEEDLIMIT, and m_sFeatureTypes.

Referenced by SetParamData().

```
393 {
394         // Note this function reqires the order of the types
395         // to not change
396         int i = 0;
397         while(str != m_sFeatureTypes.GetAt(i))
398                 i++;
399
400         switch(i)
401         {
402                 case 0:         return FEAT_SPEEDLIMIT;
403                 case 1:         return FEAT_GRADIENT;
404                 default:        return FEAT_SPEEDLIMIT;
405         }
406 }
```

4.29.4.8 void CRoadFeaturesDlg::SetParamData (int *row*, int *col*, double *val*) [private]

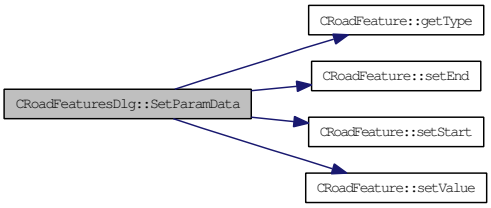
Definition at line 358 of file RoadFeaturesDlg.cpp.

References FEAT_SPEEDLIMIT, CRoadFeature::getType(), KM_PER_H_TO_M_PER_S, m_nFixRows, m_vRoadFeatures, CRoadFeature::setEnd(), CRoadFeature::setStart(), and CRoadFeature::setValue().

Referenced by OnGridEndEdit().

```
359 {
360         CRoadFeature* pFeat = reinterpret_cast<CRoadFeature*>(m_vRoadFeatures[row-m_nFixRows]);
361
362         switch(col)
363         {
364                 case 3:
365                         pFeat->setStart( (int)val );
366                         break;
367                 case 4:
368                         pFeat->setEnd( (int)val );
369                         break;
370                 case 5:
371                         {
372                                 if(pFeat->getType() == FEAT_SPEEDLIMIT)
373                                         val = val * KM_PER_H_TO_M_PER_S;        // note the uni
374                                 pFeat->setValue( val );
375                                 break;
376                         }
377                 default:
378                         pFeat->setValue( val );
379         }
380 }
```

Here is the call graph for this function:



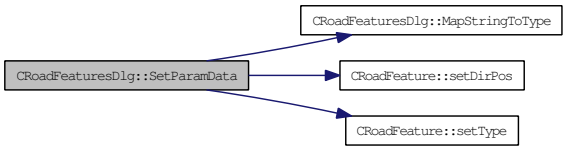
4.29.4.9 void CRoadFeaturesDlg::SetParamData (int *row*, int *col*, CString *str*) [private]

Definition at line 341 of file RoadFeaturesDlg.cpp.

References m_nFixCols, m_nFixRows, m_sDirections, m_vRoadFeatures, MapStringToType(), CRoadFeature::setDirPos(), and CRoadFeature::setType().

```
342 {
343         // if it's a string it's either col 1 or 2
344         CRoadFeature* pFeat = reinterpret_cast<CRoadFeature*>(m_vRoadFeatures[row-m_nFixRows]);
345 
346         col = col - m_nFixCols;
347 
348         if(col == 0)            // feature type
349                 pFeat->setType( MapStringToType(str) );
350 
351         else if(col == 1)       // direction
352         {
353                 bool DirPos = str == m_sDirections.GetAt(0) ? true : false;
354                 pFeat->setDirPos(DirPos);
355         }
356 }
```

Here is the call graph for this function:



4.29.4.10 CString CRoadFeaturesDlg::MapTypeToString (WORD *type*) [private]

Definition at line 382 of file RoadFeaturesDlg.cpp.

References FEAT_GRADIENT, FEAT_SPEEDLIMIT, and m_sFeatureTypes.

Referenced by LoadFeaturesIntoGrid().

```
383 {
384         switch(type)
385         {
386                 case FEAT_SPEEDLIMIT:   return m_sFeatureTypes.GetAt(0);
387                 case FEAT_GRADIENT:             return m_sFeatureTypes.GetAt(1);
388                 default:                                return m_sFeatureTypes.GetAt(0);
389         }
390 }
```

4.29.4.11 void CRoadFeaturesDlg::LoadFeaturesIntoGrid () [private]

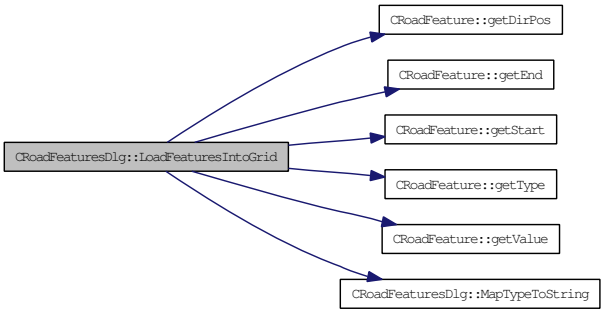
Definition at line 246 of file RoadFeaturesDlg.cpp.

References FEAT_SPEEDLIMIT, CRoadFeature::getDirPos(), CRoadFeature::getEnd(), CRoadFeature::getStart(), CRoadFeature::getType(), CRoadFeature::getValue(), m_Grid, m_nFixCols, m_nFixRows, m_NoFeatures, M_PER_S_TO_KM_PER_H, m_sDirections, m_vRoadFeatures, and MapTypeToString().

Referenced by OnBtnAddfeat(), OnBtnDelfeat(), and OnInitDialog().

```
247 {
248         for(int i = 0; i < m_NoFeatures; i++)
249         {
250                 int col = m_nFixCols;
251                 int row = m_nFixRows + i;
252                 CRoadFeature* pFeat = reinterpret_cast<CRoadFeature*>(m_vRoadFeatures[i]);
253
254                 CString txt = MapTypeToString( pFeat->getType() );
255                 m_Grid.SetItemText(row,col,txt);
256
257                 // get direction text
258                 txt = pFeat->getDirPos() == true ? m_sDirections.GetAt(0) : m_sDirections.GetAt
259                 m_Grid.SetItemText(row,col+1,txt);
260
261                 CGridCellNumeric* pCell1 = (CGridCellNumeric *)(m_Grid.GetCell(row,col+2));
262                 CGridCellNumeric* pCell2 = (CGridCellNumeric *)(m_Grid.GetCell(row,col+3));
263                 CGridCellNumeric* pCell3 = (CGridCellNumeric *)(m_Grid.GetCell(row,col+4));
264                 pCell1->SetNumber( pFeat->getStart() );
265                 pCell2->SetNumber( pFeat->getEnd() );
266
267                 double val = pFeat->getValue();
268                 if(pFeat->getType() == FEAT_SPEEDLIMIT)
269                         val = val * M_PER_S_TO_KM_PER_H;        // note the unit conversion don
270
271                 pCell3->SetNumber( val );
272         }
273         m_Grid.AutoSizeRows();
274         m_Grid.Invalidate();
275 }
```

Here is the call graph for this function:



4.29.4.12 void CRoadFeaturesDlg::SetCells (bool *bAddDelete*) [private]

Definition at line 199 of file RoadFeaturesDlg.cpp.

References m_Grid, m_nCols, m_nFixCols, m_nFixRows, m_NoFeatures, m_nRows, m_sDirections, m_sFeatureTypes, and m_vRoadFeatures.

Referenced by OnBtnAddfeat(), OnBtnDelfeat(), and OnInitDialog().

```
200 {
201         // these lines are for the resizing of the rows due to addtions/deletions
202         if(bAddDelete)
203         {
204                 m_NoFeatures = m_vRoadFeatures.GetSize();
205                 m_nRows = m_nFixRows + m_NoFeatures;
206                 m_Grid.SetRowCount(m_nRows);
207                 // update row headings
208                 for(int row = m_nFixRows; row < m_nRows; row++)
209                 {
210                         CString str;
211                         str.Format("Feature %d", row);
212                         m_Grid.SetItemText(row,0,str);
213                 }
214         }
215
216         for (int row = m_nFixRows; row < m_nRows; row++)
217         {
218                 int col = 1;    // Do Feature-type drop down
219                 m_Grid.SetCellType(row,col, RUNTIME_CLASS(CGridCellCombo));
220             CGridCellCombo *pCell = (CGridCellCombo*) m_Grid.GetCell(row,col);
221                 pCell->SetOptions(m_sFeatureTypes);
222                 pCell->SetStyle(CBS_DROPDOWNLIST);
223
224                 col = 2;                // Do directions drop down
225                 m_Grid.SetCellType(row,col, RUNTIME_CLASS(CGridCellCombo));
226             pCell = (CGridCellCombo*) m_Grid.GetCell(row,col);
227                 pCell->SetOptions(m_sDirections);
228                 pCell->SetStyle(CBS_DROPDOWNLIST);
229         }
230
231         for (row = m_nFixRows; row < m_nRows; row++)
232         {
233                 for (int col = m_nFixCols+2; col < m_nCols; col++)
```

```
234                 {
235                         m_Grid.SetCellType(row,col, RUNTIME_CLASS(CGridCellNumeric));
236                         m_Grid.GetCell(row,col)->SetFormat(DT_CENTER|DT_VCENTER|DT_SINGLELINE|D
237                         CGridCellNumeric *pCell = (CGridCellNumeric*)m_Grid.GetCell(row,col);
238                         if(col == m_nCols - 1)
239                                 pCell->SetFlags(CGridCellNumeric::Real | CGridCellNumeric::Nega
240                         else
241                                 pCell->SetFlags(CGridCellNumeric::Integer);     // all others a
242                 }
243         }
244 }
```

4.29.4.13 void CRoadFeaturesDlg::SetGridHeadings () [private]

Definition at line 180 of file RoadFeaturesDlg.cpp.

References m_Grid, m_nCols, m_nFixRows, m_nRows, and m_sColumnHeaders.

Referenced by OnInitDialog().

```
181 {
182         int row = 0;
183         int col = 0;
184
185         // Set fixed column text
186         for (col = 1; col < m_nCols; col++)
187                 m_Grid.SetItemText(row,col,m_sColumnHeaders.GetAt(col-1));
188
189         // Set fixed row text
190         col = 0;
191         for(row = m_nFixRows; row < m_nRows; row++)
192         {
193                 CString str;
194                 str.Format("Feature %d", row);
195                 m_Grid.SetItemText(row,col,str);
196         }
197 }
```

4.29.5 Member Data Documentation

4.29.5.1 int CRoadFeaturesDlg::m_RoadLength

Definition at line 23 of file RoadFeaturesDlg.h.

Referenced by CEvolveTrafficView::OnConfigFeatures(), and OnValidate().

4.29.5.2 int CRoadFeaturesDlg::m_nFixCols

Definition at line 29 of file RoadFeaturesDlg.h.

Referenced by CRoadFeaturesDlg(), LoadFeaturesIntoGrid(), OnInitDialog(), SetCells(), and SetParamData().

4.29.5.3 int CRoadFeaturesDlg::m_nFixRows

Definition at line 30 of file RoadFeaturesDlg.h.

Referenced by CRoadFeaturesDlg(), LoadFeaturesIntoGrid(), OnGridEndEdit(), OnInitDialog(), SetCells(), SetGridHeadings(), and SetParamData().

4.29.5.4 int CRoadFeaturesDlg::m_nCols

Definition at line 31 of file RoadFeaturesDlg.h.

Referenced by CRoadFeaturesDlg(), OnInitDialog(), SetCells(), and SetGridHeadings().

4.29.5.5 int CRoadFeaturesDlg::m_nRows

Definition at line 32 of file RoadFeaturesDlg.h.

Referenced by OnInitDialog(), SetCells(), and SetGridHeadings().

4.29.5.6 CObArray CRoadFeaturesDlg::m_vRoadFeatures

Definition at line 33 of file RoadFeaturesDlg.h.

Referenced by LoadFeaturesIntoGrid(), OnBtnAddfeat(), OnBtnDelfeat(), CEvolveTrafficView::OnConfigFeatures(), OnInitDialog(), OnValidate(), SetCells(), and SetParamData().

4.29.5.7 CGridCtrl CRoadFeaturesDlg::m_Grid

Definition at line 35 of file RoadFeaturesDlg.h.

Referenced by DoDataExchange(), LoadFeaturesIntoGrid(), OnGridEndEdit(), OnInitDialog(), SetCells(), and SetGridHeadings().

4.29.5.8 int CRoadFeaturesDlg::m_NoFeatures [private]

Definition at line 67 of file RoadFeaturesDlg.h.

Referenced by LoadFeaturesIntoGrid(), OnBtnAddfeat(), OnBtnDelfeat(), OnInitDialog(), and SetCells().

4.29.5.9 CStringArray CRoadFeaturesDlg::m_sDirections [private]

Definition at line 68 of file RoadFeaturesDlg.h.

Referenced by CRoadFeaturesDlg(), LoadFeaturesIntoGrid(), SetCells(), and SetParamData().

4.29.5.10 CStringArray CRoadFeaturesDlg::m_sFeatureTypes [private]

Definition at line 69 of file RoadFeaturesDlg.h.

Referenced by CRoadFeaturesDlg(), MapStringToType(), MapTypeToString(), and SetCells().

4.29.5.11 CStringArray CRoadFeaturesDlg::m_sColumnHeaders [private]

Definition at line 70 of file RoadFeaturesDlg.h.

Referenced by CRoadFeaturesDlg(), and SetGridHeadings().

4.29.5.12 int CRoadFeaturesDlg::MAX_SLOPE [private]

Definition at line 72 of file RoadFeaturesDlg.h.

Referenced by CRoadFeaturesDlg(), and OnValidate().

4.29.5.13 int CRoadFeaturesDlg::SPEEDLIMIT_MIN [private]

Definition at line 73 of file RoadFeaturesDlg.h.

Referenced by CRoadFeaturesDlg(), and OnValidate().

4.29.5.14 int CRoadFeaturesDlg::SPEEDLIMIT_MAX [private]

Definition at line 74 of file RoadFeaturesDlg.h.

Referenced by CRoadFeaturesDlg(), and OnValidate().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/RoadFeaturesDlg.h
- D:/~Research/Code/C++/EvolveTraffic/RoadFeaturesDlg.cpp

4.30 CSimConfigDlg Class Reference

A class for the Simulation Configurations Dialog.

`#include <SimConfigDlg.h>`

Public Types

- enum { IDD = IDD_SIMCONFIG }

Public Member Functions

- CSimConfigDlg (CWnd *pParent=NULL)
- virtual void OnValidate (UINT &nID, CString &strMessage)

Public Attributes

- CString m_FileIn
- CString m_FileOut
- CString m_MetricsDir

- int m_RoadLength
- int m_NoLanesDirPos
- int m_NoLanesDirNeg
- int m_FileType
- int m_SimTimeStep
- int m_TrafFileNoLanesDirPos
- int m_TrafFileNoLanesDirNeg
- bool m_DriveOnRight
- int m_LocOutputDetectorDirPos
- int m_LocOutputDetectorDirNeg
- BOOL m_AllowLaneChanging
- int m_intDriveSide

Protected Member Functions

- virtual void DoDataExchange (CDataExchange *pDX)
- BOOL OnInitDialog ()
- afx_msg void OnBtnFileInPick ()
- afx_msg void OnBtnFileOutPick ()
- afx_msg void OnRadCastor ()
- afx_msg void OnRadSaft ()
- afx_msg void OnSelchangeCmbDriveside ()
- afx_msg void OnBtnMetricsout ()

4.30.1 Detailed Description

A class for the Simulation Configurations Dialog.

Definition at line 18 of file SimConfigDlg.h.

4.30.2 Member Enumeration Documentation

4.30.2.1 anonymous enum

Enumerator:

IDD

Definition at line 26 of file SimConfigDlg.h.

```
26 { IDD = IDD_SIMCONFIG };
```

4.30.3 Constructor & Destructor Documentation

4.30.3.1 CSimConfigDlg::CSimConfigDlg (CWnd * *pParent* = NULL)

Definition at line 18 of file SimConfigDlg.cpp.

References CASTOR, m_AllowLaneChanging, m_DriveOnRight, m_FileIn, m_FileOut, m_FileType, m_intDriveSide, m_LocOutputDetectorDirNeg, m_LocOutputDetectorDirPos, m_MetricsDir, m_NoLanesDirNeg, m_NoLanesDirPos, m_RoadLength, m_SimTimeStep, m_TrafFileNoLanesDirNeg, and m_TrafFileNoLanesDirPos.

```
19         : CDialogExt(CSimConfigDlg::IDD, pParent)
20 {
21         //{{AFX_DATA_INIT(CSimConfigDlg)
22         m_FileIn = _T("");
23         m_RoadLength = 5000;
24         m_NoLanesDirPos = 1;
25         m_NoLanesDirNeg = 1;
26         m_FileType = CASTOR;
27         m_SimTimeStep = 100;
28         m_TrafFileNoLanesDirPos = -1;
29         m_TrafFileNoLanesDirNeg = -1;
30         m_FileOut = _T("");
31         m_DriveOnRight = true;
32         m_LocOutputDetectorDirPos = 0;
33         m_LocOutputDetectorDirNeg = 0;
34         m_AllowLaneChanging = true;
35         m_intDriveSide = -1;
36         m_MetricsDir = _T("");
37         //}}AFX_DATA_INIT
38 }
```

4.30.4 Member Function Documentation

4.30.4.1 void CSimConfigDlg::OnValidate (UINT & *nID*, CString & *strMessage*) [virtual]

Definition at line 78 of file SimConfigDlg.cpp.

References IDC_CMB_LANESDIRNEG, IDC_CMB_LANESDIRPOS, IDC_CMB_TRAFFILE_LANESDIR1, IDC_CMB_TRAFFILE_LANESDIR2, IDC_EDT_DIRNEGDETECTOR, IDC_EDT_DIRPOSDETECTOR, IDC_EDT_FILE_IN, IDC_EDT_FILE_OUT, m_LocOutputDetectorDirNeg, m_LocOutputDetectorDirPos, m_NoLanesDirNeg, m_NoLanesDirPos, m_RoadLength, m_TrafFileNoLanesDirNeg, and m_TrafFileNoLanesDirPos.

```
79 {
80         UpdateData();
81
82         CString strFileIn, strFileOut, strNoLanesDirPos, strNoLanesDirNeg;
83         CString strSimTimeStep, strTrafFileNoLanesDirPos, strTrafFileNoLanesDirNeg;
84         CString strDriveSide, strLocOutputDetectorDirPos, strLocOutputDetectorDirNeg;
85
86         GetDlgItemText(IDC_EDT_FILE_IN, strFileIn);
87         GetDlgItemText(IDC_EDT_FILE_OUT, strFileOut);
88         GetDlgItemText(IDC_CMB_LANESDIRPOS, strNoLanesDirPos);
```

```
89          GetDlgItemText(IDC_CMB_LANESDIRNEG, strNoLanesDirNeg);
90          GetDlgItemText(IDC_CMB_TRAFFILE_LANESDIR1, strTrafFileNoLanesDirPos);
91          GetDlgItemText(IDC_CMB_TRAFFILE_LANESDIR2, strTrafFileNoLanesDirNeg);
92
93          if(strFileIn.IsEmpty())
94          {
95                  strMessage = "Please select a file.";
96                  nID = IDC_EDT_FILE_IN;
97                  return;
98          }
99
100          if(strFileOut.IsEmpty())
101          {
102                  strMessage = "Please specify a filename.";
103                  nID = IDC_EDT_FILE_OUT;
104                  return;
105          }
106
107          if(strNoLanesDirPos.IsEmpty())
108          {
109                  strMessage = "Please specify the number of lanes.";
110                  nID = IDC_CMB_LANESDIRPOS;
111                  return;
112          }
113
114          if(strNoLanesDirNeg.IsEmpty())
115          {
116                  strMessage = "Please specify the number of lanes.";
117                  nID = IDC_CMB_LANESDIRNEG;
118                  return;
119          }
120
121          if(strTrafFileNoLanesDirPos.IsEmpty())
122          {
123                  strMessage = "Please specify the number of lanes.";
124                  nID = IDC_CMB_TRAFFILE_LANESDIR1;
125                  return;
126          }
127
128          if(strTrafFileNoLanesDirNeg.IsEmpty())
129          {
130                  strMessage = "Please specify the number of lanes.";
131                  nID = IDC_CMB_TRAFFILE_LANESDIR2;
132                  return;
133          }
134
135          if(m_TrafFileNoLanesDirPos == 0 && m_TrafFileNoLanesDirNeg == 0)
136          {
137                  strMessage = "There cannot be zero lanes in both directions";
138                  nID = IDC_CMB_TRAFFILE_LANESDIR1;
139                  return;
140          }
141
142          if(m_TrafFileNoLanesDirPos > m_NoLanesDirPos)
143          {
144                  strMessage = "The number of simulation lanes must not \nbe less than that of th
145                  nID = IDC_CMB_LANESDIRPOS;
146                  return;
147          }
148
149          if(m_TrafFileNoLanesDirNeg > m_NoLanesDirNeg)
150          {
```

```
151                 strMessage = "The number of simulation lanes must not \nbe less than that of th
152                 nID = IDC_CMB_LANESDIRNEG;
153                 return;
154         }
155
156         if(m_LocOutputDetectorDirPos > m_RoadLength)    // don't need to check < 0 since only p
157         {
158                 strMessage = "The output dector must be located on the road \n- please check th
159                 nID = IDC_EDT_DIRPOSDETECTOR;
160                 return;
161         }
162
163         if(m_LocOutputDetectorDirNeg > m_RoadLength)
164         {
165                 strMessage = "The output dector must be located on the road \n- please check th
166                 nID = IDC_EDT_DIRNEGDETECTOR;
167                 return;
168         }
169
170         if(m_TrafFileNoLanesDirPos == 0 && m_NoLanesDirPos > 0)
171         {
172                 strMessage = "For this direction, there will be no vehicles \nsince there are n
173                 nID = IDC_CMB_LANESDIRPOS;
174                 return;
175         }
176
177         if(m_TrafFileNoLanesDirNeg == 0 && m_NoLanesDirNeg > 0)
178         {
179                 strMessage = "For this direction, there will be no vehicles \nsince there are n
180                 nID = IDC_CMB_LANESDIRNEG;
181                 return;
182         }
183 }
```

4.30.4.2 void CSimConfigDlg::DoDataExchange (CDataExchange * *pDX*) [protected, virtual]

Definition at line 41 of file SimConfigDlg.cpp.

References IDC_CHECK_LANECHNG, IDC_CMB_DRIVESIDE, IDC_-CMB_LANESDIRNEG, IDC_CMB_LANESDIRPOS, IDC_CMB_-TRAFFILE_LANESDIR1, IDC_CMB_TRAFFILE_LANESDIR2, IDC_EDT_-DIRNEGDETECTOR, IDC_EDT_DIRPOSDETECTOR, IDC_EDT_FILE_IN, IDC_EDT_FILE_OUT, IDC_EDT_METRICSOUT, IDC_EDT_ROADLENGTH, IDC_EDT_SIMTIMESTEP, m_AllowLaneChanging, m_FileIn, m_FileOut, m_intDriveSide, m_LocOutputDetectorDirNeg, m_LocOutputDetectorDirPos, m_MetricsDir, m_NoLanesDirNeg, m_NoLanesDirPos, m_RoadLength, m_-SimTimeStep, m_TrafFileNoLanesDirNeg, and m_TrafFileNoLanesDirPos.

```
42 {
43         CDialogExt::DoDataExchange(pDX);
44         //{{AFX_DATA_MAP(CSimConfigDlg)
45         DDX_Text(pDX, IDC_EDT_FILE_IN, m_FileIn);
46         DDX_Text(pDX, IDC_EDT_ROADLENGTH, m_RoadLength);
47         DDV_MinMaxInt(pDX, m_RoadLength, 500, 100000);
48         DDX_CBIndex(pDX, IDC_CMB_LANESDIRPOS, m_NoLanesDirPos);
49         DDX_CBIndex(pDX, IDC_CMB_LANESDIRNEG, m_NoLanesDirNeg);
50         DDX_Text(pDX, IDC_EDT_SIMTIMESTEP, m_SimTimeStep);
```

```
51          DDV_MinMaxInt(pDX, m_SimTimeStep, 10, 2000);
52          DDX_CBIndex(pDX, IDC_CMB_TRAFFILE_LANESDIR1, m_TrafFileNoLanesDirPos);
53          DDX_CBIndex(pDX, IDC_CMB_TRAFFILE_LANESDIR2, m_TrafFileNoLanesDirNeg);
54          DDX_Text(pDX, IDC_EDT_FILE_OUT, m_FileOut);
55          DDX_Text(pDX, IDC_EDT_DIRPOSDETECTOR, m_LocOutputDetectorDirPos);
56          DDX_Text(pDX, IDC_EDT_DIRNEGDETECTOR, m_LocOutputDetectorDirNeg);
57          DDX_Check(pDX, IDC_CHECK_LANECHNG, m_AllowLaneChanging);
58          DDX_CBIndex(pDX, IDC_CMB_DRIVESIDE, m_intDriveSide);
59          DDX_Text(pDX, IDC_EDT_METRICSOUT, m_MetricsDir);
60          //}}AFX_DATA_MAP
61  }
```

4.30.4.3 BOOL CSimConfigDlg::OnInitDialog () [protected]

Definition at line 185 of file SimConfigDlg.cpp.

References CASTOR, IDC_CMB_DRIVESIDE, IDC_RAD_CASTOR, IDC_RAD_-SAFT, m_DriveOnRight, m_FileType, m_intDriveSide, and SAFT.

```
186 {
187         CDialogExt::OnInitDialog();
188
189         switch (m_FileType)
190         {
191                 case SAFT:
192                         CheckDlgButton(IDC_RAD_SAFT,1);
193                         break;
194                 case CASTOR:
195                         CheckDlgButton(IDC_RAD_CASTOR,1);
196                         break;
197                 default:
198                         CheckDlgButton(IDC_RAD_CASTOR,1);
199         }
200
201         m_intDriveSide = 1;     // default to "Right"
202         if(!m_DriveOnRight)
203                 m_intDriveSide = 0;     // but "Left" if needs be
204         CComboBox * pCmb = (CComboBox*)GetDlgItem(IDC_CMB_DRIVESIDE);
205         pCmb->SetCurSel(m_intDriveSide);
206
207         return TRUE;
208 }
```

4.30.4.4 void CSimConfigDlg::OnBtnFileInPick () [protected]

Definition at line 210 of file SimConfigDlg.cpp.

References m_FileIn.

```
211 {
212         // szFilters is a text string that includes two file name filters:
213         // "*.my" for "MyType Files" and "*.*' for "All Files."
214         //char CChildFrame::szFilters[] = ;
215
216         CFileDialog FileDlg(
217                 TRUE,   // TRUE if File Open, FALSE if File Save As
218                 NULL,   // the default file extension
```

```
219                 NULL,   // the initial filename that appears
220                 OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST | OFN_HIDEREADONLY,//flags for customizin
221                 "All Files (*.*)|*.*||",        // file filters
222                 this); // pointer to FileCialog's parent object
223
224    if(FileDlg.DoModal() == IDOK)
225    {
226       m_FileIn = FileDlg.GetPathName();
227           UpdateData(FALSE); // The FALSE switches the direction of the DDX
228    }
229 }
```

4.30.4.5 void CSimConfigDlg::OnBtnFileOutPick () [protected]

Definition at line 231 of file SimConfigDlg.cpp.

References m_FileOut.

```
232 {
233         CFileDialog FileDlg(
234                 FALSE,                  // TRUE if File Open, FALSE if File Save As
235                 _T("*.txt"),    // the default file extension
236                 NULL,                   // the initial filename that appears
237                 OFN_PATHMUSTEXIST | OFN_HIDEREADONLY | OFN_OVERWRITEPROMPT,//flags for customiz
238                 "text files (*.txt) |*.txt|All Files (*.*)|*.*||",      // file filters
239                 this); // pointer to FileCialog's parent object
240
241    if(FileDlg.DoModal() == IDOK)
242    {
243       m_FileOut = FileDlg.GetPathName();
244           UpdateData(FALSE); // The FALSE switches the direction of the DDX
245    }
246 }
```

4.30.4.6 void CSimConfigDlg::OnRadCastor () [protected]

Definition at line 277 of file SimConfigDlg.cpp.

References CASTOR, and m_FileType.

```
278 {
279         m_FileType = CASTOR;
280 }
```

4.30.4.7 void CSimConfigDlg::OnRadSaft () [protected]

Definition at line 282 of file SimConfigDlg.cpp.

References m_FileType, and SAFT.

```
283 {
284         m_FileType = SAFT;
285 }
```

4.30.4.8 void CSimConfigDlg::OnSelchangeCmbDriveside () [protected]

Definition at line 287 of file SimConfigDlg.cpp.

References IDC_CMB_DRIVESIDE, and m_DriveOnRight.

```
288 {
289         CString str;
290         GetDlgItemText(IDC_CMB_DRIVESIDE, str);
291 
292         if(str == "Left")
293                 m_DriveOnRight = false;
294         else
295                 m_DriveOnRight = true;
296 }
```

4.30.4.9 void CSimConfigDlg::OnBtnMetricsout () [protected]

Definition at line 248 of file SimConfigDlg.cpp.

References m_MetricsDir.

```
249 {
250     BROWSEINFO bi = { 0 };
251     bi.lpszTitle = _T("Pick a Directory");
252     LPITEMIDLIST pidl = SHBrowseForFolder ( &bi );
253     if ( pidl != 0 )
254     {
255         // get the name of the folder
256         TCHAR path[MAX_PATH];
257         if ( SHGetPathFromIDList ( pidl, path ) )
258         {
259             //TRACE("Selected Folder: %s\n", path );
260                         m_MetricsDir = path;
261                         m_MetricsDir += "\\";
262                         UpdateData(FALSE);
263         }
264                 else
265                         MessageBox("Error in selecting folder - please type the name","EvolveTr
266 
267         // free memory used
268         IMalloc * imalloc = 0;
269         if ( SUCCEEDED( SHGetMalloc ( &imalloc )) )
270         {
271             imalloc->Free ( pidl );
272             imalloc->Release ( );
273         }
274     }
275 }
```

4.30.5 Member Data Documentation

4.30.5.1 CString CSimConfigDlg::m_FileIn

Definition at line 27 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), DoDataExchange(), OnBtnFileInPick(), and CEvolveTrafficView::OnConfigSim().

4.30.5.2 CString CSimConfigDlg::m_FileOut

Definition at line 28 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), DoDataExchange(), OnBtnFileOutPick(), and CEvolveTrafficView::OnConfigSim().

4.30.5.3 CString CSimConfigDlg::m_MetricsDir

Definition at line 29 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), DoDataExchange(), OnBtnMetricsout(), and CEvolveTrafficView::OnConfigSim().

4.30.5.4 int CSimConfigDlg::m_RoadLength

Definition at line 30 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), DoDataExchange(), CEvolveTrafficView::OnConfigSim(), and OnValidate().

4.30.5.5 int CSimConfigDlg::m_NoLanesDirPos

Definition at line 31 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), DoDataExchange(), CEvolveTrafficView::OnConfigSim(), and OnValidate().

4.30.5.6 int CSimConfigDlg::m_NoLanesDirNeg

Definition at line 32 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), DoDataExchange(), CEvolveTrafficView::OnConfigSim(), and OnValidate().

4.30.5.7 int CSimConfigDlg::m_FileType

Definition at line 33 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), CEvolveTrafficView::OnConfigSim(), OnInitDialog(), OnRadCastor(), and OnRadSaft().

4.30.5.8 int CSimConfigDlg::m_SimTimeStep

Definition at line 34 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), DoDataExchange(), and CEvolveTrafficView::OnConfigSim().

4.30.5.9 int CSimConfigDlg::m_TrafFileNoLanesDirPos

Definition at line 35 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), DoDataExchange(), CEvolveTrafficView::OnConfigSim(), and OnValidate().

4.30.5.10 int CSimConfigDlg::m_TrafFileNoLanesDirNeg

Definition at line 36 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), DoDataExchange(), CEvolveTrafficView::OnConfigSim(), and OnValidate().

4.30.5.11 bool CSimConfigDlg::m_DriveOnRight

Definition at line 37 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), CEvolveTrafficView::OnConfigSim(), OnInitDialog(), and OnSelchangeCmbDriveside().

4.30.5.12 int CSimConfigDlg::m_LocOutputDetectorDirPos

Definition at line 38 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), DoDataExchange(), CEvolveTrafficView::OnConfigSim(), and OnValidate().

4.30.5.13 int CSimConfigDlg::m_LocOutputDetectorDirNeg

Definition at line 39 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), DoDataExchange(), CEvolveTrafficView::OnConfigSim(), and OnValidate().

4.30.5.14 BOOL CSimConfigDlg::m_AllowLaneChanging

Definition at line 40 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), DoDataExchange(), and CEvolveTrafficView::OnConfigSim().

4.30.5.15 int CSimConfigDlg::m_intDriveSide

Definition at line 41 of file SimConfigDlg.h.

Referenced by CSimConfigDlg(), DoDataExchange(), and OnInitDialog().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/SimConfigDlg.h
- D:/~Research/Code/C++/EvolveTraffic/SimConfigDlg.cpp

4.31 CStatDetector Class Reference

A class for the general Detector object that can be saved to file.

```
#include <StatDetector.h>
```

Public Member Functions

- CStatDetector ()
- virtual ~CStatDetector ()
- void Serialize (CArchive &ar)
- std::string getStdStrMetricsDir ()
- WORD getDetectorType ()
- bool getDirPos ()
- WORD getVehicleType ()
- int getTimeInterval ()
- int getLocation ()
- void setMetricsDir (CString dir)
- void setDetectorType (WORD type)
- void setDirPos (bool dirpos)
- void setVehicleType (WORD type)
- void setTimeInterval (int interval)
- void setLocation (int loc)

Private Attributes

- CString m_MetricsDir
- WORD m_DetectorType
- bool m_DirPos
- WORD m_VehicleType
- int m_TimeInterval
- int m_Location

4.31.1 Detailed Description

A class for the general Detector object that can be saved to file.

Definition at line 16 of file StatDetector.h.

4.31.2 Constructor & Destructor Documentation

4.31.2.1 CStatDetector::CStatDetector ()

Definition at line 21 of file StatDetector.cpp.

References METRICS_TYPE_FLOWDENSITY, and METRICS_VEH_ALL.

```
22 {
23         m_DetectorType  = METRICS_TYPE_FLOWDENSITY;
24         m_DirPos                = true;
25         m_VehicleType   = METRICS_VEH_ALL;
26         m_TimeInterval  = 60;
27         m_Location              = 1000;
28 }
```

4.31.2.2 CStatDetector::~CStatDetector () [virtual]

Definition at line 30 of file StatDetector.cpp.

```
31 {
32
33 }
```

4.31.3 Member Function Documentation

4.31.3.1 void CStatDetector::Serialize (CArchive & *ar*)

Definition at line 35 of file StatDetector.cpp.

References m_DetectorType, m_DirPos, m_Location, m_TimeInterval, and m_VehicleType.

```
36 {
37     if (ar.IsStoring())
38     {
39         ar      << m_DetectorType
40                         << static_cast<int>(m_DirPos)   // because I read in an int below
41                         << m_VehicleType
42                         << m_TimeInterval
43                         << m_Location;
44     }
45     else
46         {
47                 int temp;
48         ar      >> m_DetectorType
49                         >> temp
50                         >> m_VehicleType
51                         >> m_TimeInterval
52                         >> m_Location;
53
54                 m_DirPos = temp == 0 ? false : true;
55         }
56 }
```

4.31.3.2 std::string CStatDetector::getStdStrMetricsDir ()

Definition at line 113 of file StatDetector.cpp.

References m_MetricsDir.

Referenced by Road::SetMetricDetFromStatDet().

```
114 {
115          std::string str = (LPCTSTR)m_MetricsDir;
116          return str;
117 }
```

4.31.3.3 WORD CStatDetector::getDetectorType ()

Definition at line 58 of file StatDetector.cpp.

References m_DetectorType.

Referenced by CEvolveTrafficView::DrawDetectors(), CStatDetectorDlg::LoadStatDetectorsIntoGrid(), CStatDetectorDlg::OnValidate(), and Road::SetMetricDetFromStatDet().

```
59 {
60         return m_DetectorType;
61 }
```

4.31.3.4 bool CStatDetector::getDirPos ()

Definition at line 63 of file StatDetector.cpp.

References m_DirPos.

Referenced by CEvolveTrafficView::DrawDetectors(), CStatDetectorDlg::LoadStatDetectorsIntoGrid(), and Road::SetMetricDetFromStatDet().

```
64 {
65         return m_DirPos;
66 }
```

4.31.3.5 WORD CStatDetector::getVehicleType ()

Definition at line 68 of file StatDetector.cpp.

References m_VehicleType.

Referenced by CStatDetectorDlg::LoadStatDetectorsIntoGrid(), and Road::SetMetricDetFromStatDet().

```
69 {
70         return m_VehicleType;
71 }
```

4.31.3.6 int CStatDetector::getTimeInterval ()

Definition at line 73 of file StatDetector.cpp.

References m_TimeInterval.

Referenced by CStatDetectorDlg::LoadStatDetectorsIntoGrid(), CStatDetectorDlg::OnValidate(), and Road::SetMetricDetFromStatDet().

```
74 {
75         return m_TimeInterval;
76 }
```

4.31.3.7 int CStatDetector::getLocation ()

Definition at line 78 of file StatDetector.cpp.

References m_Location.

Referenced by CEvolveTrafficView::DrawDetectors(), CStatDetectorDlg::LoadStatDetectorsIntoGrid(), CStatDetectorDlg::OnValidate(), and Road::SetMetricDetFromStatDet().

```
79 {
80         return m_Location;
81 }
```

4.31.3.8 void CStatDetector::setMetricsDir (CString *dir*)

Definition at line 108 of file StatDetector.cpp.

References m_MetricsDir.

Referenced by CEvolveTrafficDoc::initSim().

```
109 {
110          m_MetricsDir = dir;
111 }
```

4.31.3.9 void CStatDetector::setDetectorType (WORD *type*)

Definition at line 83 of file StatDetector.cpp.

References m_DetectorType.

Referenced by CStatDetectorDlg::SetParamData().

```
84 {
85         m_DetectorType = type;
86 }
```

4.31.3.10 void CStatDetector::setDirPos (bool *dirpos*)

Definition at line 88 of file StatDetector.cpp.

References m_DirPos.

Referenced by CStatDetectorDlg::SetParamData().

```
89 {
90         m_DirPos = dirpos;
91 }
```

4.31.3.11 void CStatDetector::setVehicleType (WORD *type*)

Definition at line 93 of file StatDetector.cpp.

References m_VehicleType.

Referenced by CStatDetectorDlg::SetParamData().

```
94 {
95         m_VehicleType = type;
96 }
```

4.31.3.12 void CStatDetector::setTimeInterval (int *interval*)

Definition at line 98 of file StatDetector.cpp.

References m_TimeInterval.

Referenced by CStatDetectorDlg::SetParamData().

```
99 {
100         m_TimeInterval = interval;
101 }
```

4.31.3.13 void CStatDetector::setLocation (int *loc*)

Definition at line 103 of file StatDetector.cpp.

References m_Location.

Referenced by CStatDetectorDlg::SetParamData().

```
104 {
105         m_Location = loc;
106 }
```

4.31.4 Member Data Documentation

4.31.4.1 CString CStatDetector::m_MetricsDir [private]

Definition at line 43 of file StatDetector.h.

Referenced by getStdStrMetricsDir(), and setMetricsDir().

4.31.4.2 WORD CStatDetector::m_DetectorType [private]

Definition at line 44 of file StatDetector.h.

Referenced by getDetectorType(), Serialize(), and setDetectorType().

4.31.4.3 bool CStatDetector::m_DirPos [private]

Definition at line 45 of file StatDetector.h.

Referenced by getDirPos(), Serialize(), and setDirPos().

4.31.4.4 WORD CStatDetector::m_VehicleType [private]

Definition at line 46 of file StatDetector.h.

Referenced by getVehicleType(), Serialize(), and setVehicleType().

4.31.4.5 int CStatDetector::m_TimeInterval [private]

Definition at line 47 of file StatDetector.h.

Referenced by getTimeInterval(), Serialize(), and setTimeInterval().

4.31.4.6 int CStatDetector::m_Location [private]

Definition at line 48 of file StatDetector.h.

Referenced by getLocation(), Serialize(), and setLocation().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/StatDetector.h
- D:/~Research/Code/C++/EvolveTraffic/StatDetector.cpp

4.32 CStatDetectorDlg Class Reference

A class for the Detector Dialog.

`#include <StatDetectorDlg.h>`

Public Types

- enum { IDD = IDD_STATDETECTORS }

Public Member Functions

- CStatDetectorDlg (CWnd *pParent=NULL)
- virtual void OnValidate (UINT &nID, CString &strMessage)

Public Attributes

- int m_RoadLength
- int m_nFixCols
- int m_nFixRows
- int m_nCols
- int m_nRows
- CObArray m_vStatDetectors
- CGridCtrl m_Grid

Protected Member Functions

- virtual void DoDataExchange (CDataExchange ∗pDX)
- virtual BOOL OnInitDialog ()
- afx_msg void OnBtnAdddet ()
- afx_msg void OnBtnDeldet ()
- void OnGridEndEdit (NMHDR ∗pNotifyStruct, LRESULT ∗pResult)

Private Member Functions

- WORD MapStringToDetType (CString str)
- WORD MapStringToVehType (CString str)
- void SetParamData (int row, int col, int val)
- void SetParamData (int row, int col, CString str)
- CString MapDetTypeToString (WORD type)
- CString MapVehTypeToString (WORD type)
- void LoadStatDetectorsIntoGrid ()
- void SetCells (bool bAddDelete)
- void SetGridHeadings ()

Private Attributes

- int m_NoStatDetectors
- CStringArray m_sDirections
- CStringArray m_sStatDetectorTypes
- CStringArray m_sVehicleTypes
- CStringArray m_sColumnHeaders

4.32.1 Detailed Description

A class for the Detector Dialog.

Definition at line 19 of file StatDetectorDlg.h.

4.32.2 Member Enumeration Documentation

4.32.2.1 anonymous enum

Enumerator:

IDD

Definition at line 28 of file StatDetectorDlg.h.

```
28 { IDD = IDD_STATDETECTORS };
```

4.32.3 Constructor & Destructor Documentation

4.32.3.1 CStatDetectorDlg::CStatDetectorDlg (CWnd * *pParent* = NULL)

Definition at line 23 of file StatDetectorDlg.cpp.

References FIXED_COLUMNS, FIXED_ROWS, m_nCols, m_nFixCols, m_nFixRows, m_sColumnHeaders, m_sDirections, m_sStatDetectorTypes, and m_sVehicleTypes.

```
24         : CDialogExt(CStatDetectorDlg::IDD, pParent)
25 {
26         //{{AFX_DATA_INIT(CStatDetectorDlg)
27                 // NOTE: the ClassWizard will add member initialization here
28         //}}AFX_DATA_INIT
29
30         m_sStatDetectorTypes.Add("Flow & Density");
31         m_sStatDetectorTypes.Add("Headway");
32         m_sStatDetectorTypes.Add("Composition");
33         m_sStatDetectorTypes.Add("Lane Changes");
34
35         m_sDirections.Add("Positive x");
36         m_sDirections.Add("Negative x");
37
38         m_sVehicleTypes.Add("All Vehicles");
39         m_sVehicleTypes.Add("Cars");
40         m_sVehicleTypes.Add("Small Trucks");
41         m_sVehicleTypes.Add("Large Trucks");
42         m_sVehicleTypes.Add("Cranes");
43         m_sVehicleTypes.Add("Low-loaders");
44
45         m_sColumnHeaders.Add("Type");
46         m_sColumnHeaders.Add("Direction");
47         m_sColumnHeaders.Add("Vehicle Types");
48         m_sColumnHeaders.Add("Time Interval (s)");
49         m_sColumnHeaders.Add("Location / Step (m)");
50
51         m_nFixCols = FIXED_COLUMNS;
52         m_nFixRows = FIXED_ROWS;
53         m_nCols = m_sColumnHeaders.GetSize() + m_nFixCols;
54 }
```

4.32.4 Member Function Documentation

4.32.4.1 void CStatDetectorDlg::OnValidate (UINT & *nID*, CString & *strMessage*) [virtual]

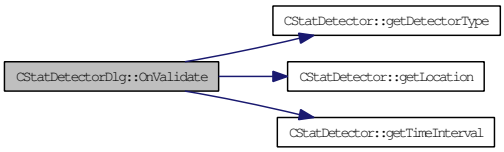
Definition at line 107 of file StatDetectorDlg.cpp.

References CStatDetector::getDetectorType(), CStatDetector::getLocation(), CStatDetector::getTimeInterval(), IDC_GRID, m_RoadLength, m_vStatDetectors, and METRICS_TYPE_LANE_CHANGE.

```
108 {
109          int nSD = m_vStatDetectors.GetSize();
110
111          for(int i = 0; i < nSD;  i++)
112          {
```

```
113                 CStatDetector* pSD = reinterpret_cast<CStatDetector*>(m_vStatDetectors.GetAt(i)
114
115                 int location = pSD->getLocation();
116                 if(location > m_RoadLength)     // since negative numbers can't occur - only ch
117                 {
118                         strMessage.Format("Detector %d: must be located on the road.\nPlease ch
119                         nID = IDC_GRID;
120                         return;
121                 }
122
123                 int time = pSD->getTimeInterval();
124                 if(time < 10 && time > 0)
125                 {
126                         CString str; str.Format("Detector %d: The time interval is very small a
127                         int result = MessageBox(str + "\nAre you sure you want to keep this val
128                         if(result == IDNO)
129                         {
130                                 strMessage.Format("Detector %d: Change the time interval.", i+1
131                                 nID = IDC_GRID;
132                                 return;
133                         }
134                 }
135                 else if(time == 0)
136                 {
137                         strMessage.Format("Detector %d: The time interval must not be zero.", i
138                         nID = IDC_GRID;
139                         return;
140                 }
141
142                 WORD DetType = pSD->getDetectorType();
143                 int dist_interval = location;
144                 if(DetType == METRICS_TYPE_LANE_CHANGE && dist_interval > 1000)
145                 {
146                         CString str; str.Format("Detector %d: The lane change detector step is
147                         MessageBox(str,"EvolveTraffic", MB_OK | MB_ICONWARNING);
148                 }
149         }
150 }
```

Here is the call graph for this function:



4.32.4.2 void CStatDetectorDlg::DoDataExchange (CDataExchange * *pDX*) [protected, virtual]

Definition at line 57 of file StatDetectorDlg.cpp.

References IDC_GRID, and m_Grid.

```
58 {
59         CDialogExt::DoDataExchange(pDX);
```

```
60          //{{AFX_DATA_MAP(CStatDetectorDlg)
61          DDX_Control(pDX, IDC_GRID, m_Grid);             // associate the grid window with a C++
62          //}}AFX_DATA_MAP
63 }
```

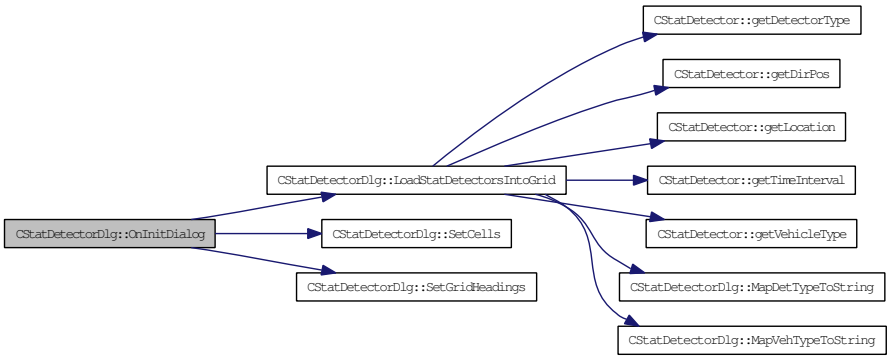
4.32.4.3 BOOL CStatDetectorDlg::OnInitDialog () [protected, virtual]

Definition at line 77 of file StatDetectorDlg.cpp.

References LoadStatDetectorsIntoGrid(), m_Grid, m_nCols, m_nFixCols, m_nFixRows, m_NoStatDetectors, m_nRows, m_vStatDetectors, SetCells(), and SetGridHeadings().

```
78 {
79          CDialogExt::OnInitDialog();
80
81          // init rows
82          m_NoStatDetectors = m_vStatDetectors.GetSize();
83          m_nRows = m_nFixRows + m_NoStatDetectors;
84
85          TRY {
86                  m_Grid.SetRowCount(m_nRows);
87                  m_Grid.SetColumnCount(m_nCols);
88                  m_Grid.SetFixedRowCount(m_nFixRows);
89                  m_Grid.SetFixedColumnCount(m_nFixCols);
90          }
91          CATCH (CMemoryException, e)
92          {
93                  e->ReportError();
94                  return FALSE;
95          }
96          END_CATCH
97
98          SetGridHeadings();
99          SetCells(false);        // false because not addition or detletion
100          m_Grid.ExpandColumnsToFit();
101
102          LoadStatDetectorsIntoGrid();
103
104          return TRUE;  // return TRUE  unless you set the focus to a control
105 }
```

Here is the call graph for this function:



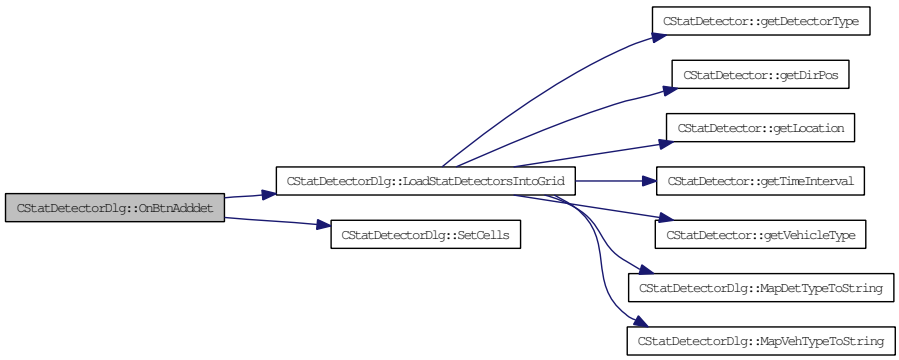
4.32.4.4 void CStatDetectorDlg::OnBtnAdddet () [protected]

Definition at line 253 of file StatDetectorDlg.cpp.

References LoadStatDetectorsIntoGrid(), m_NoStatDetectors, m_vStatDetectors, and SetCells().

```
254 {
255         // Add the feature to the vector
256         m_NoStatDetectors++;
257         CStatDetector* pStatDet = new CStatDetector;    // create one to use
258         m_vStatDetectors.Add(pStatDet);
259
260         // add a row to the table
261         SetCells(true);
262         LoadStatDetectorsIntoGrid();
263 }
```

Here is the call graph for this function:



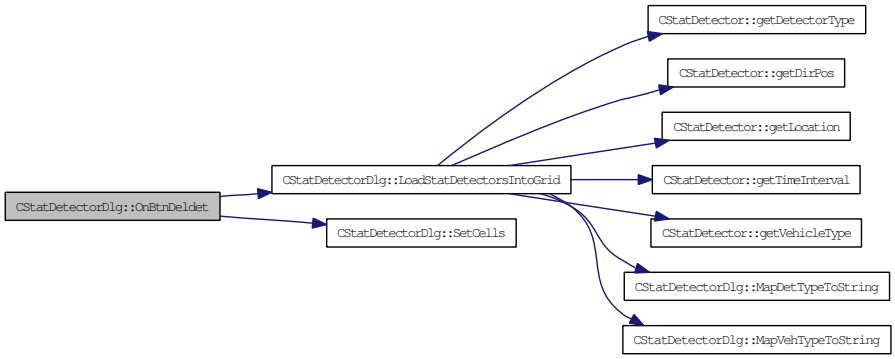
4.32.4.5 void CStatDetectorDlg::OnBtnDeldet () [protected]

Definition at line 265 of file StatDetectorDlg.cpp.

References LoadStatDetectorsIntoGrid(), m_NoStatDetectors, m_vStatDetectors, and SetCells().

```
266 {
267         if(m_NoStatDetectors != 0)
268         {
269                 // Get the ptr and remove from the vector
270                 CStatDetector* pStatDet = reinterpret_cast<CStatDetector*>(m_vStatDetectors.Get
271                 m_vStatDetectors.RemoveAt( m_NoStatDetectors-1 );
272
273                 delete pStatDet;        // remove feature from heap
274                 m_NoStatDetectors--;    // decrement only now
275
276                 // update the table
277                 SetCells(true);
278                 LoadStatDetectorsIntoGrid();
279         }
280         else
281                 MessageBox("No statistics detector to delete.", "EvolveTraffic", MB_OK|MB_ICONW
282 }
```

Here is the call graph for this function:



4.32.4.6 void CStatDetectorDlg::OnGridEndEdit (NMHDR * *pNotifyStruct*, LRESULT * *pResult*) [protected]

Definition at line 285 of file StatDetectorDlg.cpp.

References m_Grid, m_nFixRows, and SetParamData().

```
286 {
287         // if change is ok, then *pResult 0, else *pResult -1
288
289         // pItem is the cell that has just been edited
290     NM_GRIDVIEW* pItem = (NM_GRIDVIEW*) pNotifyStruct;
291
292     int row = pItem->iRow;      int col = pItem->iColumn;
```

```
293
294         if(col < 3 + m_nFixRows )       // is the type, direction, or vehicle column
295         {
296                 CString strCell = m_Grid.GetCell(row,col)->GetText();
297                 if(strCell != "")
298                 {
299                         // data is ok
300                         SetParamData(row,col,strCell);
301                         *pResult = 0;
302                 }
303                 else
304                         *pResult = -1;
305         }
306         else
307         {
308                 int val = (int)((CGridCellNumeric *)(m_Grid.GetCell(row,col)))->GetNumber();
309                 SetParamData(row,col,val);
310                 *pResult = 0;
311         }
312
313 }
```

Here is the call graph for this function:

4.32.4.7 WORD CStatDetectorDlg::MapStringToDetType (CString *str*) [private]

Definition at line 401 of file StatDetectorDlg.cpp.

References m_sStatDetectorTypes, METRICS_TYPE_COMPOSITION, METRICS_-TYPE_FLOWDENSITY, METRICS_TYPE_HEADWAY, and METRICS_TYPE_-LANE_CHANGE.

Referenced by SetParamData().

```
402 {
403         // Note this function reqires the order of the types
404         // to not change
405         int i = 0;
406         while(str != m_sStatDetectorTypes.GetAt(i))
407                 i++;
408
409         switch(i)
410         {
411                 case 0:         return METRICS_TYPE_FLOWDENSITY;
412                 case 1:         return METRICS_TYPE_HEADWAY;
413                 case 2:         return METRICS_TYPE_COMPOSITION;
414                 case 3:         return METRICS_TYPE_LANE_CHANGE;
415                 default:        return METRICS_TYPE_FLOWDENSITY;
416         }
417 }
```

4.32.4.8 WORD CStatDetectorDlg::MapStringToVehType (CString *str*) [private]

Definition at line 381 of file StatDetectorDlg.cpp.

References m_sVehicleTypes, METRICS_VEH_ALL, METRICS_VEH_CAR, METRICS_VEH_CRANE, METRICS_VEH_LARGETRUCK, METRICS_VEH_-LOWLOADER, and METRICS_VEH_SMALLTRUCK.

Referenced by SetParamData().

```
382 {
383         // Note this function reqires the order of the types
384         // to not change
385         int i = 0;
386         while(str != m_sVehicleTypes.GetAt(i))
387                 i++;
388
389         switch(i)
390         {
391                 case 0:         return METRICS_VEH_ALL;
392                 case 1:         return METRICS_VEH_CAR;
393                 case 2:         return METRICS_VEH_SMALLTRUCK;
394                 case 3:         return METRICS_VEH_LARGETRUCK;
395                 case 4:         return METRICS_VEH_CRANE;
396                 case 5:         return METRICS_VEH_LOWLOADER;
397                 default:        return METRICS_VEH_ALL;
398         }
399 }
```

4.32.4.9 void CStatDetectorDlg::SetParamData (int *row*, int *col*, int *val*) [private]

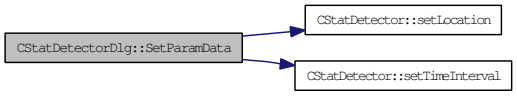
Definition at line 339 of file StatDetectorDlg.cpp.

References m_nFixCols, m_nFixRows, m_vStatDetectors, CStatDetec-tor::setLocation(), and CStatDetector::setTimeInterval().

Referenced by OnGridEndEdit().

```
340 {
341         CStatDetector* pStatDet = reinterpret_cast<CStatDetector*>(m_vStatDetectors[row-m_nFixR
342
343         col = col - m_nFixCols;
344
345         switch(col)
346         {
347                 case 3:         // time
348                         pStatDet->setTimeInterval(val);
349                         break;
350                 default:        // location
351                         pStatDet->setLocation(val);
352         }
353 }
```

Here is the call graph for this function:



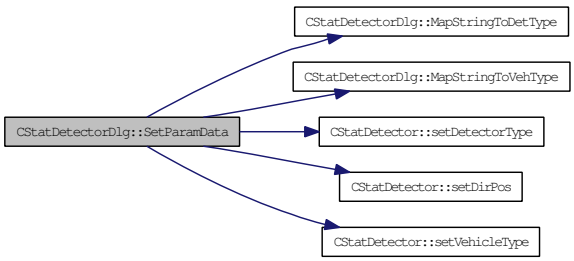
4.32.4.10 void CStatDetectorDlg::SetParamData (int *row*, int *col*, CString *str*) [private]

Definition at line 315 of file StatDetectorDlg.cpp.

References m_nFixCols, m_nFixRows, m_sDirections, m_vStatDetectors, MapStringToDetType(), MapStringToVehType(), CStatDetector::setDetectorType(), CStatDetector::setDirPos(), and CStatDetector::setVehicleType().

```
316 {
317         // if it's a string it's either col 1 or 2
318         CStatDetector* pStatDet = reinterpret_cast<CStatDetector*>(m_vStatDetectors[row-m_nFixR
319
320         col = col - m_nFixCols;
321
322
323         switch(col)
324         {
325                 case 0:                 // detector type
326                         pStatDet->setDetectorType( MapStringToDetType(str) );
327                         break;
328                 case 1:                 // direction
329                         {
330                                 bool DirPos = str == m_sDirections.GetAt(0) ? true : false;
331                                 pStatDet->setDirPos(DirPos);
332                                 break;
333                         }
334                 default:                // vehicle
335                         pStatDet->setVehicleType( MapStringToVehType(str) );
336         }
337 }
```

Here is the call graph for this function:



4.32.4.11 CString CStatDetectorDlg::MapDetTypeToString (WORD *type*) [private]

Definition at line 369 of file StatDetectorDlg.cpp.

References m_sStatDetectorTypes, METRICS_TYPE_COMPOSITION, METRICS_-TYPE_FLOWDENSITY, METRICS_TYPE_HEADWAY, and METRICS_TYPE_-LANE_CHANGE.

Referenced by LoadStatDetectorsIntoGrid().

```
370 {
371         switch(type)
372         {
373                 case METRICS_TYPE_FLOWDENSITY:  return m_sStatDetectorTypes.GetAt(0);
374                 case METRICS_TYPE_HEADWAY:              return m_sStatDetectorTypes.GetAt(1);
375                 case METRICS_TYPE_COMPOSITION:  return m_sStatDetectorTypes.GetAt(2);
376                 case METRICS_TYPE_LANE_CHANGE:  return m_sStatDetectorTypes.GetAt(3);
377                 default:                                                        return m_sStatDetector
378         }
379 }
```

4.32.4.12 CString CStatDetectorDlg::MapVehTypeToString (WORD *type*) [private]

Definition at line 355 of file StatDetectorDlg.cpp.

References m_sVehicleTypes, METRICS_VEH_ALL, METRICS_VEH_CAR, METRICS_VEH_CRANE, METRICS_VEH_LARGETRUCK, METRICS_VEH_-LOWLOADER, and METRICS_VEH_SMALLTRUCK.

Referenced by LoadStatDetectorsIntoGrid().

```
356 {
357         switch(type)
358         {
359                 case METRICS_VEH_ALL:                   return m_sVehicleTypes.GetAt(0);
360                 case METRICS_VEH_CAR:                   return m_sVehicleTypes.GetAt(1);
361                 case METRICS_VEH_SMALLTRUCK:    return m_sVehicleTypes.GetAt(2);
362                 case METRICS_VEH_LARGETRUCK:    return m_sVehicleTypes.GetAt(3);
363                 case METRICS_VEH_CRANE:                 return m_sVehicleTypes.GetAt(4);
364                 case METRICS_VEH_LOWLOADER:             return m_sVehicleTypes.GetAt(5);
365                 default:                                                        return m_sVehicleTypes
366         }
367 }
```

4.32.4.13 void CStatDetectorDlg::LoadStatDetectorsIntoGrid () [private]

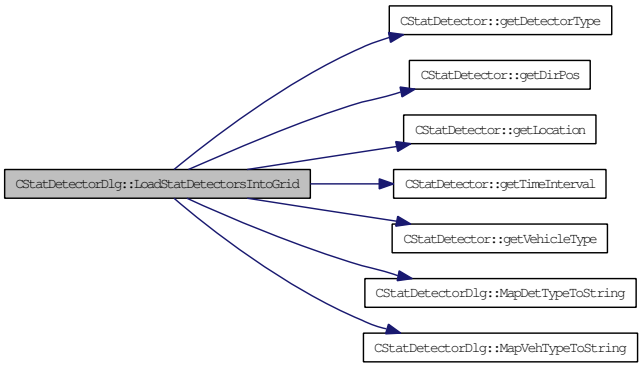
Definition at line 222 of file StatDetectorDlg.cpp.

References CStatDetector::getDetectorType(), CStatDetector::getDirPos(), CStatDetector::getLocation(), CStatDetector::getTimeInterval(), CStatDetector::getVehicleType(), m_Grid, m_nFixCols, m_nFixRows, m_NoStatDetectors, m_sDirections, m_vStatDetectors, MapDetTypeToString(), and MapVehType-ToString().

Referenced by OnBtnAdddet(), OnBtnDeldet(), and OnInitDialog().

```
223 {
224         for(int i = 0; i < m_NoStatDetectors; i++)
225         {
226                 int col = m_nFixCols;
227                 int row = m_nFixRows + i;
228                 CStatDetector* pStatDet = reinterpret_cast<CStatDetector*>(m_vStatDetectors[i])
229
230                 CString txt = MapDetTypeToString( pStatDet->getDetectorType() );
231                 m_Grid.SetItemText(row,col,txt);
232
233                 col++;  // next col - set direction text
234                 txt = pStatDet->getDirPos() == true ? m_sDirections.GetAt(0) : m_sDirections.Ge
235                 m_Grid.SetItemText(row,col,txt);
236
237                 col++;  // next col - set vehicle text
238                 txt = MapVehTypeToString( pStatDet->getVehicleType() );
239                 m_Grid.SetItemText(row,col,txt);
240
241                 col++;  // next col - set time interval
242                 CGridCellNumeric* pCell = (CGridCellNumeric *)(m_Grid.GetCell(row,col));
243                 pCell->SetNumber( pStatDet->getTimeInterval() );
244
245                 col++;  // next col - set location
246                 pCell = (CGridCellNumeric *)(m_Grid.GetCell(row,col));
247                 pCell->SetNumber( pStatDet->getLocation() );
248         }
249         m_Grid.AutoSizeRows();
250         m_Grid.Invalidate();
251 }
```

Here is the call graph for this function:



4.32.4.14 void CStatDetectorDlg::SetCells (bool *bAddDelete*) [private]

Definition at line 171 of file StatDetectorDlg.cpp.

References m_Grid, m_nFixRows, m_NoStatDetectors, m_nRows, m_sDirections, m_sStatDetectorTypes, m_sVehicleTypes, and m_vStatDetectors.

Referenced by OnBtnAdddet(), OnBtnDeldet(), and OnInitDialog().

```
172 {
173         // these lines are for the resizing of the rows due to addtions/deletions
174         if(bAddDelete)
175         {
176                 m_NoStatDetectors = m_vStatDetectors.GetSize();
177                 m_nRows = m_nFixRows + m_NoStatDetectors;
178                 m_Grid.SetRowCount(m_nRows);
179                 // update row headings
180                 for(int row = m_nFixRows; row < m_nRows; row++)
181                 {
182                         CString str;
183                         str.Format("Detector %d", row);
184                         m_Grid.SetItemText(row,0,str);
185                 }
186         }
187
188         for (int row = m_nFixRows; row < m_nRows; row++)
189         {
190                 int col = 1;    // Do Detector-type drop down
191                 m_Grid.SetCellType(row,col, RUNTIME_CLASS(CGridCellCombo));
192             CGridCellCombo *pCell = (CGridCellCombo*) m_Grid.GetCell(row,col);
193                 pCell->SetOptions(m_sStatDetectorTypes);
194                 pCell->SetStyle(CBS_DROPDOWNLIST);
195
196                 col = 2;                // Do directions drop down
197                 m_Grid.SetCellType(row,col, RUNTIME_CLASS(CGridCellCombo));
198             pCell = (CGridCellCombo*) m_Grid.GetCell(row,col);
199                 pCell->SetOptions(m_sDirections);
200                 pCell->SetStyle(CBS_DROPDOWNLIST);
201
202                 col = 3;                // Do vehicles drop down
203                 m_Grid.SetCellType(row,col, RUNTIME_CLASS(CGridCellCombo));
204             pCell = (CGridCellCombo*) m_Grid.GetCell(row,col);
205                 pCell->SetOptions(m_sVehicleTypes);
206                 pCell->SetStyle(CBS_DROPDOWNLIST);
207
208                 col = 4;                // Do time interval cells
209                 m_Grid.SetCellType(row,col, RUNTIME_CLASS(CGridCellNumeric));
210                 m_Grid.GetCell(row,col)->SetFormat(DT_CENTER|DT_VCENTER|DT_SINGLELINE|DT_NOPRE
211                 CGridCellNumeric *pCellN = (CGridCellNumeric*)m_Grid.GetCell(row,col);
212                 pCellN->SetFlags(CGridCellNumeric::Integer);
213
214                 col = 5;                // Do location cells
215                 m_Grid.SetCellType(row,col, RUNTIME_CLASS(CGridCellNumeric));
216                 m_Grid.GetCell(row,col)->SetFormat(DT_CENTER|DT_VCENTER|DT_SINGLELINE|DT_NOPRE
217                 pCellN = (CGridCellNumeric*)m_Grid.GetCell(row,col);
218                 pCellN->SetFlags(CGridCellNumeric::Integer);
219         }
220 }
```

4.32.4.15 void CStatDetectorDlg::SetGridHeadings () [private]

Definition at line 152 of file StatDetectorDlg.cpp.

References m_Grid, m_nCols, m_nFixRows, m_nRows, and m_sColumnHeaders.

Referenced by OnInitDialog().

```
153 {
154         int row = 0;
```

```
155         int col = 0;
156
157         // Set fixed column text
158         for (col = 1; col < m_nCols; col++)
159                 m_Grid.SetItemText(row,col,m_sColumnHeaders.GetAt(col-1));
160
161         // Set fixed row text
162         col = 0;
163         for(row = m_nFixRows; row < m_nRows; row++)
164         {
165                 CString str;
166                 str.Format("Detector %d", row);
167                 m_Grid.SetItemText(row,col,str);
168         }
169 }
```

4.32.5 Member Data Documentation

4.32.5.1 int CStatDetectorDlg::m_RoadLength

Definition at line 23 of file StatDetectorDlg.h.

Referenced by CEvolveTrafficView::OnConfigMetrics(), and OnValidate().

4.32.5.2 int CStatDetectorDlg::m_nFixCols

Definition at line 29 of file StatDetectorDlg.h.

Referenced by CStatDetectorDlg(), LoadStatDetectorsIntoGrid(), OnInitDialog(), and SetParamData().

4.32.5.3 int CStatDetectorDlg::m_nFixRows

Definition at line 30 of file StatDetectorDlg.h.

Referenced by CStatDetectorDlg(), LoadStatDetectorsIntoGrid(), OnGridEndEdit(), OnInitDialog(), SetCells(), SetGridHeadings(), and SetParamData().

4.32.5.4 int CStatDetectorDlg::m_nCols

Definition at line 31 of file StatDetectorDlg.h.

Referenced by CStatDetectorDlg(), OnInitDialog(), and SetGridHeadings().

4.32.5.5 int CStatDetectorDlg::m_nRows

Definition at line 32 of file StatDetectorDlg.h.

Referenced by OnInitDialog(), SetCells(), and SetGridHeadings().

4.32.5.6 CObArray CStatDetectorDlg::m_vStatDetectors

Definition at line 33 of file StatDetectorDlg.h.

Referenced by LoadStatDetectorsIntoGrid(), OnBtnAdddet(), OnBtnDeldet(), CEvolveTrafficView::OnConfigMetrics(), OnInitDialog(), OnValidate(), SetCells(), and SetParamData().

4.32.5.7 CGridCtrl CStatDetectorDlg::m_Grid

Definition at line 35 of file StatDetectorDlg.h.

Referenced by DoDataExchange(), LoadStatDetectorsIntoGrid(), OnGridEndEdit(), OnInitDialog(), SetCells(), and SetGridHeadings().

4.32.5.8 int CStatDetectorDlg::m_NoStatDetectors [private]

Definition at line 68 of file StatDetectorDlg.h.

Referenced by LoadStatDetectorsIntoGrid(), OnBtnAdddet(), OnBtnDeldet(), OnInitDialog(), and SetCells().

4.32.5.9 CStringArray CStatDetectorDlg::m_sDirections [private]

Definition at line 69 of file StatDetectorDlg.h.

Referenced by CStatDetectorDlg(), LoadStatDetectorsIntoGrid(), SetCells(), and SetParamData().

4.32.5.10 CStringArray CStatDetectorDlg::m_sStatDetectorTypes [private]

Definition at line 70 of file StatDetectorDlg.h.

Referenced by CStatDetectorDlg(), MapDetTypeToString(), MapStringToDetType(), and SetCells().

4.32.5.11 CStringArray CStatDetectorDlg::m_sVehicleTypes [private]

Definition at line 71 of file StatDetectorDlg.h.

Referenced by CStatDetectorDlg(), MapStringToVehType(), MapVehTypeToString(), and SetCells().

4.32.5.12 CStringArray CStatDetectorDlg::m_sColumnHeaders [private]

Definition at line 72 of file StatDetectorDlg.h.

Referenced by CStatDetectorDlg(), and SetGridHeadings().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/StatDetectorDlg.h
- D:/~Research/Code/C++/EvolveTraffic/StatDetectorDlg.cpp

4.33 CTrafficConfigDlg Class Reference

A class for the IDM Model Configuration Dialog.

`#include <TrafficConfigDlg.h>`

Collaboration diagram for CTrafficConfigDlg:

Public Types

- enum { IDD = IDD_TRAFFICCONFIG }

Public Member Functions

- CTrafficConfigDlg (CWnd *pParent=NULL)

Public Attributes

- CComboBox m_cmbVehicleClassCopy
- CComboBox m_cmbVehicleClassDefine
- int m_nFixCols
- int m_nFixRows
- int m_nCols
- int m_nRows
- CStringArray m_sDistributions
- CIDMParameterSet m_IDMParams_Car
- CIDMParameterSet m_IDMParams_SmallTruck
- CIDMParameterSet m_IDMParams_LargeTruck
- CIDMParameterSet m_IDMParams_Crane
- CIDMParameterSet m_IDMParams_Lowloader
- CGridCtrl m_Grid

Protected Member Functions

- virtual void DoDataExchange (CDataExchange *pDX)
- virtual BOOL OnInitDialog ()
- afx_msg void OnBtnCopy ()
- afx_msg void OnSelchangeCmbClassCopy ()
- afx_msg void OnSelchangeCmbClassDefine ()
- void OnGridEndEdit (NMHDR *pNotifyStruct, LRESULT *pResult)

Private Member Functions

- CParameter * MapRowToParameter (int row)

  MAPPING FUNCTIONS FOR TEXT/OBJECT TO ID & BACK.

- void SetParamData (int row, int col, double val)
- void SetParamData (int row, int col, CString str)
- CIDMParameterSet * MapIDMParamSet (int iSelect)
- void LoadRow (int row)
- WORD MapDistributionString (CString dist)
- CString MapDistributionID (WORD distID)
- void LoadParamsIntoGrid ()
- void SetCells ()
- void SetGridHeadings ()

Private Attributes

- CIDMParameterSet * m_pCurrentIDMParamSet
- CIDMParameterSet * m_pCopyIDMParamSet
- CStringArray m_sCmbOptions
- CStringArray m_sRowHeaders
- CStringArray m_sColumnHeaders

4.33.1 Detailed Description

A class for the IDM Model Configuration Dialog.

Definition at line 20 of file TrafficConfigDlg.h.

4.33.2 Member Enumeration Documentation

4.33.2.1 anonymous enum

Enumerator:

IDD

Definition at line 28 of file TrafficConfigDlg.h.

```
28 { IDD = IDD_TRAFFICCONFIG };
```

4.33.3 Constructor & Destructor Documentation

4.33.3.1 CTrafficConfigDlg::CTrafficConfigDlg (CWnd * *pParent* = NULL)

Definition at line 21 of file TrafficConfigDlg.cpp.

References FIXED_COLUMNS, FIXED_ROWS, m_nCols, m_nFixCols, m_nFixRows, m_nRows, m_sCmbOptions, m_sColumnHeaders, m_sDistributions, and m_sRowHeaders.

```
22         : CDialog(CTrafficConfigDlg::IDD, pParent)
23 {
24     m_sDistributions.Add("Exponential");
25     m_sDistributions.Add("Log-Normal");
26     m_sDistributions.Add("Gamma");
27         m_sDistributions.Add("Gumbel");
28         m_sDistributions.Add("Poisson");
29         m_sDistributions.Add("GEV");
30         m_sDistributions.Add("Normal");
31         m_sDistributions.Add("Constant");
32 
33         m_sColumnHeaders.Add("Distribution");
34         m_sColumnHeaders.Add("Location");
35         m_sColumnHeaders.Add("Scale");
36         m_sColumnHeaders.Add("Shape");
37 
```

```
38          m_sRowHeaders.Add("Safe time headway, T (s)");
39          m_sRowHeaders.Add("Maximum acceleration, a (m/s^2)");
40          m_sRowHeaders.Add("Comfortable decceleration, b (m/s^2)");
41          m_sRowHeaders.Add("Minimum jam distance, s0 (m)");
42          m_sRowHeaders.Add("Elastic jam distance, s1 (m)");
43          m_sRowHeaders.Add("Desired velocity, v0 (km/h)");
44          m_sRowHeaders.Add("Acceleration exponent, delta");
45          m_sRowHeaders.Add("Lane change politeness factor, p");
46          m_sRowHeaders.Add("Outside lane bias factor, deltaAbias");
47          m_sRowHeaders.Add("Lane change threshold, deltaAth (m/s^2)");
48
49          m_sCmbOptions.Add("Class 0 - Cars");
50          m_sCmbOptions.Add("Class 1 - Small Truck");
51          m_sCmbOptions.Add("Class 2 - Large Truck");
52          m_sCmbOptions.Add("Class 3 - Crane");
53          m_sCmbOptions.Add("Class 4 - Low-loader");
54
55          m_nFixCols = FIXED_COLUMNS;     // MAGIC NUMBER for row & column heads
56          m_nFixRows = FIXED_ROWS;
57          m_nCols = m_sColumnHeaders.GetSize() + m_nFixCols;
58          m_nRows = m_sRowHeaders.GetSize() + m_nFixRows;
59
60          //{{AFX_DATA_INIT(CTrafficConfigDlg)
61                  // NOTE: the ClassWizard will add member initialization here
62          //}}AFX_DATA_INIT
63  }
```

4.33.4 Member Function Documentation

4.33.4.1 void CTrafficConfigDlg::DoDataExchange (CDataExchange * *pDX*) [protected, virtual]

Definition at line 66 of file TrafficConfigDlg.cpp.

References IDC_CMB_CLASSCOPY, IDC_CMB_CLASSDEFINE, IDC_GRID, m_cmbVehicleClassCopy, m_cmbVehicleClassDefine, and m_Grid.

```
67  {
68          CDialog::DoDataExchange(pDX);
69          //{{AFX_DATA_MAP(CTrafficConfigDlg)
70          DDX_Control(pDX, IDC_CMB_CLASSCOPY, m_cmbVehicleClassCopy);
71          DDX_Control(pDX, IDC_CMB_CLASSDEFINE, m_cmbVehicleClassDefine);
72          DDX_Control(pDX, IDC_GRID, m_Grid);             // associate the grid window with a C++
73          //}}AFX_DATA_MAP
74  }
```

4.33.4.2 BOOL CTrafficConfigDlg::OnInitDialog () [protected, virtual]

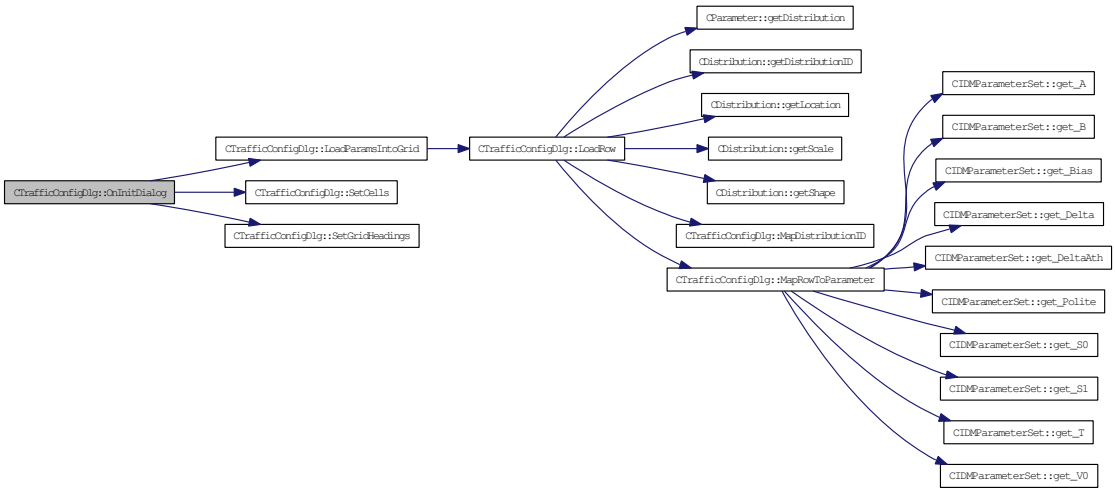
Definition at line 89 of file TrafficConfigDlg.cpp.

References LoadParamsIntoGrid(), m_cmbVehicleClassCopy, m_cmbVehicleClassDefine, m_Grid, m_IDMParams_Car, m_nCols, m_nFixCols, m_nFixRows, m_nRows, m_pCurrentIDMParamSet, m_sCmbOptions, SetCells(), and SetGridHeadings().

```
90  {
```

```
91          CDialog::OnInitDialog();
92
93          TRY {
94                  m_Grid.SetRowCount(m_nRows);
95                  m_Grid.SetColumnCount(m_nCols);
96                  m_Grid.SetFixedRowCount(m_nFixRows);
97                  m_Grid.SetFixedColumnCount(m_nFixCols);
98          }
99          CATCH (CMemoryException, e)
100          {
101                  e->ReportError();
102                  return FALSE;
103          }
104          END_CATCH
105
106          SetGridHeadings();
107          SetCells();
108          m_Grid.AutoSizeColumn(0);       // headings
109          m_Grid.ExpandToFit();
110
111          m_pCurrentIDMParamSet = &m_IDMParams_Car;
112          LoadParamsIntoGrid();
113
114          // Load ComboBox text dynamically
115          for(int i = 0; i < m_sCmbOptions.GetSize(); i++)
116          {
117                  m_cmbVehicleClassDefine.AddString(m_sCmbOptions.GetAt(i));
118                  m_cmbVehicleClassCopy.AddString(m_sCmbOptions.GetAt(i));
119          }
120          // Select the first vehicle class
121          m_cmbVehicleClassDefine.SelectString(0,m_sCmbOptions.GetAt(0));
122
123          return TRUE;  // return TRUE  unless you set the focus to a control
124 }
```

Here is the call graph for this function:



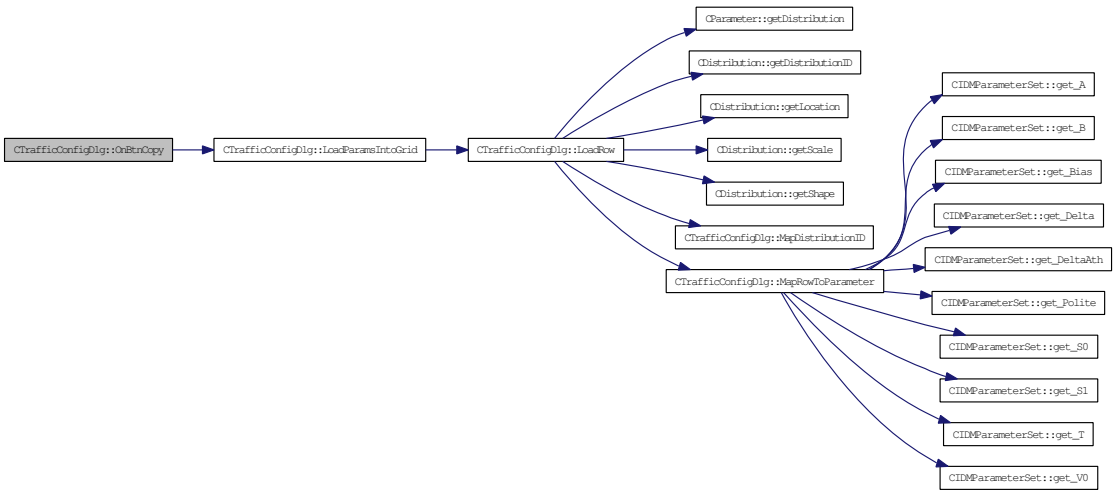
4.33.4.3 void CTrafficConfigDlg::OnBtnCopy () [protected]

Definition at line 200 of file TrafficConfigDlg.cpp.

References LoadParamsIntoGrid(), m_pCopyIDMParamSet, and m_pCurrentIDMParamSet.

```
201 {
202         // Once something is chosen
203         if( m_pCurrentIDMParamSet != NULL && m_pCopyIDMParamSet != NULL)
204         {
205                 // and they are not the same objects
206                 if( m_pCurrentIDMParamSet != m_pCopyIDMParamSet )
207                 {
208                         // copy them
209                         *m_pCurrentIDMParamSet = *m_pCopyIDMParamSet;
210                         LoadParamsIntoGrid();
211                 }
212         }
213 }
```

Here is the call graph for this function:



4.33.4.4 void CTrafficConfigDlg::OnSelchangeCmbClassCopy () [protected]

Definition at line 215 of file TrafficConfigDlg.cpp.

References m_cmbVehicleClassCopy, m_pCopyIDMParamSet, and MapIDMParamSet().

```
216 {
217         int iSelect = m_cmbVehicleClassCopy.GetCurSel();
218         m_pCopyIDMParamSet = MapIDMParamSet(iSelect);
219 }
```

Here is the call graph for this function:

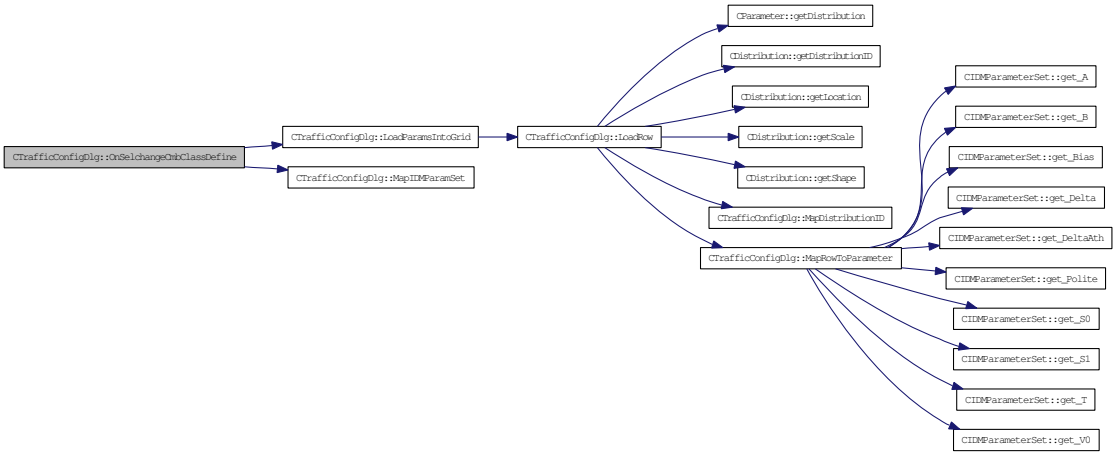
4.33.4.5 void CTrafficConfigDlg::OnSelchangeCmbClassDefine () [protected]

Definition at line 221 of file TrafficConfigDlg.cpp.

References LoadParamsIntoGrid(), m_cmbVehicleClassDefine, m_pCurrentIDMParamSet, and MapIDMParamSet().

```
222 {
223         // Load chosen paramset into m_pCurrentIDMParamSet
224         int iSelect = m_cmbVehicleClassDefine.GetCurSel();
225         m_pCurrentIDMParamSet = MapIDMParamSet(iSelect);
226
227         LoadParamsIntoGrid();
228 }
```

Here is the call graph for this function:



4.33.4.6 void CTrafficConfigDlg::OnGridEndEdit (NMHDR * *pNotifyStruct*, LRESULT * *pResult*) [protected]

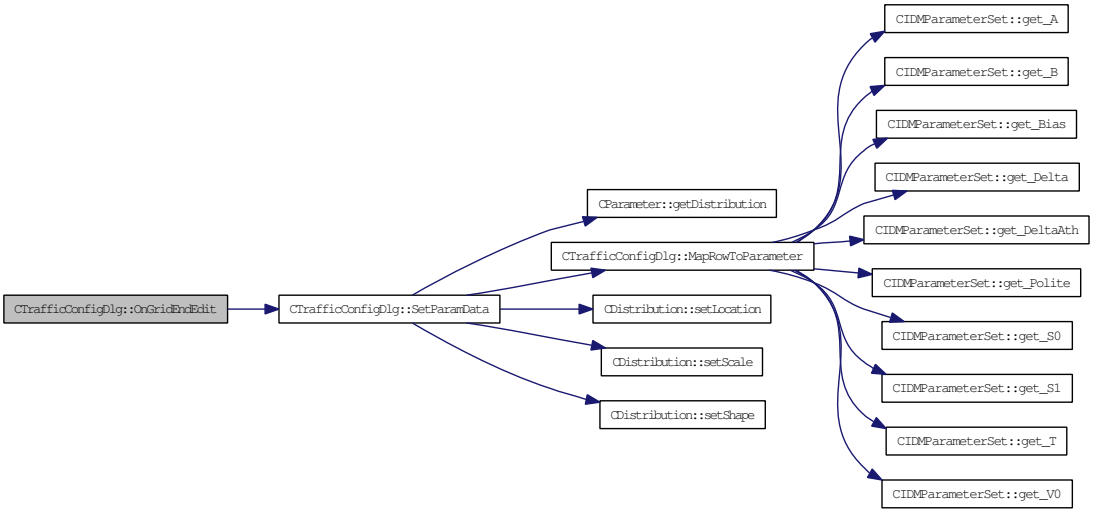
Definition at line 231 of file TrafficConfigDlg.cpp.

References m_Grid, m_nFixCols, and SetParamData().

```
232 {
233         // if change is ok, then *pResult 0, else *pResult -1
234
```

```
235         // pItem is the cell that has just been edited
236     NM_GRIDVIEW* pItem = (NM_GRIDVIEW*) pNotifyStruct;
237
238     int row = pItem->iRow;
239         int col = pItem->iColumn;
240
241         if(col == m_nFixCols)   // is the distriubtion column
242         {
243                 CString strCell = m_Grid.GetCell(row,col)->GetText();
244                 if(strCell != "")
245                 {
246                         // data is ok
247                         SetParamData(row,col,strCell);
248                         *pResult = 0;
249                 }
250                 else
251                         *pResult = -1;
252         }
253         else
254         {
255                 // no need to verify CNumericCells since will always be a number
256                 // even deleting a cell's data results in a zero
257                 double val = ((CGridCellNumeric *)(m_Grid.GetCell(row,col)))->GetNumber();
258                 SetParamData(row,col,val);
259                 *pResult = 0;
260         }
261 }
```

Here is the call graph for this function:



4.33.4.7 CParameter * CTrafficConfigDlg::MapRowToParameter (int *row*) [private]

MAPPING FUNCTIONS FOR TEXT/OBJECT TO ID & BACK.

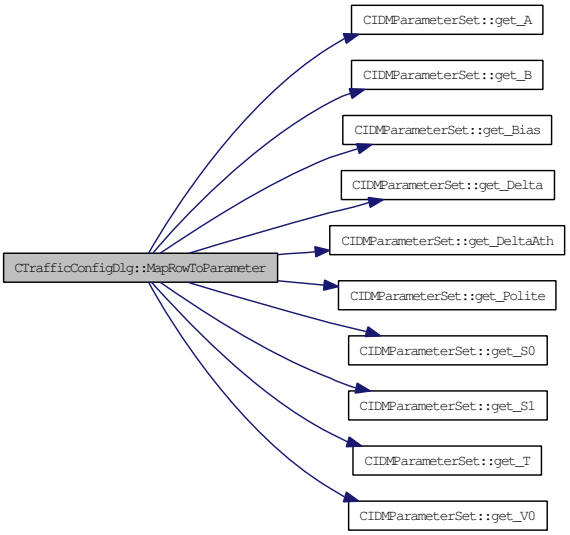
Definition at line 295 of file TrafficConfigDlg.cpp.

References CIDMParameterSet::get_A(), CIDMParameterSet::get_B(), CIDMParameterSet::get_Bias(), CIDMParameterSet::get_Delta(), CIDMParameterSet::get_DeltaAth(), CIDMParameterSet::get_Polite(), CIDMParameterSet::get_S0(), CIDMParameterSet::get_S1(), CIDMParameterSet::get_T(), CIDMParameterSet::get_V0(), m_nFixRows, and m_pCurrentIDMParamSet.

Referenced by LoadRow(), and SetParamData().

```
296 {
297         // This function assumes m_pCurrentIDMParamSet != NULL
298         // This function assumes the order of the parameters does not change
299
300         // this means we're independent of m_nFixCols in the cases
301         row = row - m_nFixRows;
302         switch(row)
303         {
304                 case 0:         return m_pCurrentIDMParamSet->get_T();
305                 case 1:         return m_pCurrentIDMParamSet->get_A();
306                 case 2:         return m_pCurrentIDMParamSet->get_B();
307                 case 3:         return m_pCurrentIDMParamSet->get_S0();
308                 case 4:         return m_pCurrentIDMParamSet->get_S1();
309                 case 5:         return m_pCurrentIDMParamSet->get_V0();
310                 case 6:         return m_pCurrentIDMParamSet->get_Delta();
311                 case 7:         return m_pCurrentIDMParamSet->get_Polite();
312                 case 8:         return m_pCurrentIDMParamSet->get_Bias();
313                 case 9:         return m_pCurrentIDMParamSet->get_DeltaAth();
314                 default:        return m_pCurrentIDMParamSet->get_T();  // T
315         }
316 }
```

Here is the call graph for this function:



4.33.4.8 void CTrafficConfigDlg::SetParamData (int *row*, int *col*, double *val*) [private]

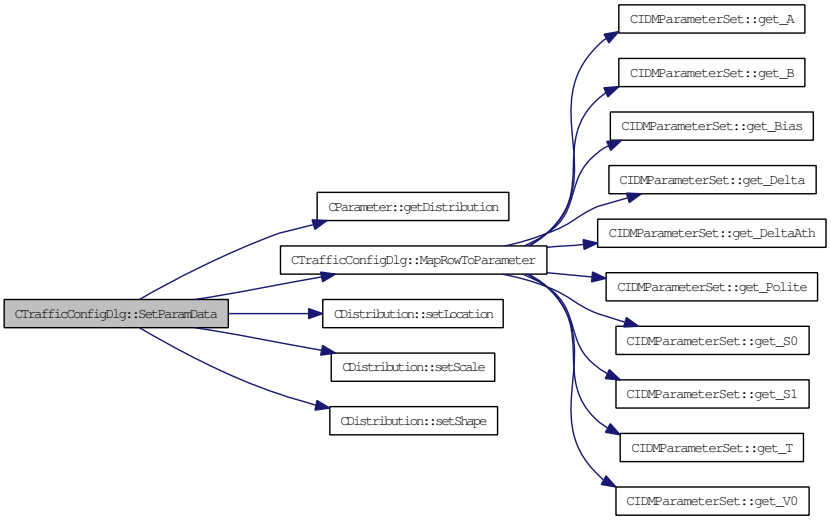
Definition at line 270 of file TrafficConfigDlg.cpp.

References CParameter::getDistribution(), m_nFixCols, MapRowToParameter(), CDistribution::setLocation(), CDistribution::setScale(), and CDistribution::setShape().

Referenced by OnGridEndEdit().

```
271 {
272         CParameter* currentParam = MapRowToParameter(row);
273
274         // this means we're independent of m_nFixCols in the cases
275         col = col - m_nFixCols;
276         switch(col)
277         {
278                 case 1:
279                         currentParam->getDistribution()->setLocation(val);
280                         break;
281                 case 2:
282                         currentParam->getDistribution()->setScale(val);
283                         break;
284                 case 3:
285                         currentParam->getDistribution()->setShape(val);
286                         break;
287                 default:
288                         currentParam->getDistribution()->setLocation(val);
289         }
290 }
```

Here is the call graph for this function:



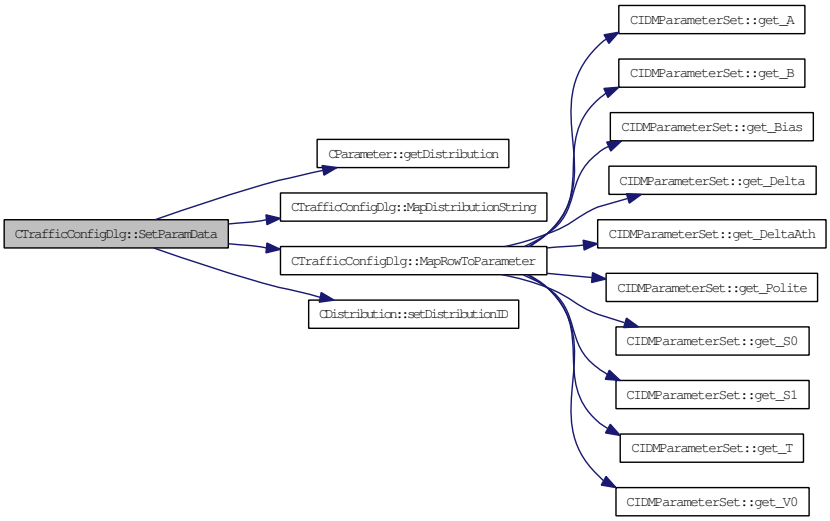
4.33.4.9 void CTrafficConfigDlg::SetParamData (int *row*, int *col*, CString *str*) [private]

Definition at line 263 of file TrafficConfigDlg.cpp.

References CParameter::getDistribution(), MapDistributionString(), MapRowToParameter(), and CDistribution::setDistributionID().

```
264 {
265         CParameter* currentParam = MapRowToParameter(row);
266         WORD dist = MapDistributionString(str);
267         currentParam->getDistribution()->setDistributionID(dist);
268 }
```

Here is the call graph for this function:



4.33.4.10 CIDMParameterSet * CTrafficConfigDlg::MapIDMParamSet (int *iSelect*) [private]

Definition at line 318 of file TrafficConfigDlg.cpp.

References m_IDMParams_Car, m_IDMParams_Crane, m_IDMParams_LargeTruck, m_IDMParams_Lowloader, and m_IDMParams_SmallTruck.

Referenced by OnSelchangeCmbClassCopy(), and OnSelchangeCmbClassDefine().

```
319 {
320         // Note that this function requires the index of the
321         // m_sCmbOptions text to not change. There is no way
322         // to avoid this since some function must map the txt
323         // string to the m-IDMParams_ classes
324
325         switch(iSelect)
326         {
327                 case 0:         return &m_IDMParams_Car;
328                 case 1:         return &m_IDMParams_SmallTruck;
329                 case 2:         return &m_IDMParams_LargeTruck;
330                 case 3:         return &m_IDMParams_Crane;
```

```
331                 case 4:         return &m_IDMParams_Lowloader;
332                 default:        return &m_IDMParams_Car;
333         }
334 }
```

4.33.4.11 void CTrafficConfigDlg::LoadRow (int *row*) [private]

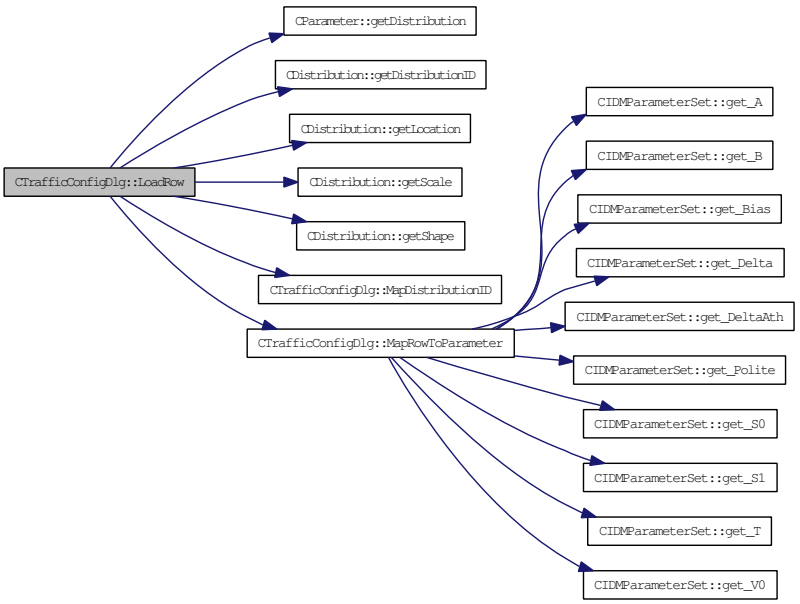
Definition at line 184 of file TrafficConfigDlg.cpp.

References CParameter::getDistribution(), CDistribution::getDistributionID(), CDistribution::getLocation(), CDistribution::getScale(), CDistribution::getShape(), m_Grid, m_nFixCols, MapDistributionID(), and MapRowToParameter().

Referenced by LoadParamsIntoGrid().

```
185 {
186         CParameter* pParam = MapRowToParameter(row);
187
188         int col = m_nFixCols;
189         CString txt = MapDistributionID( pParam->getDistribution()->getDistributionID() );
190         m_Grid.SetItemText(row,col,txt);
191
192         CGridCellNumeric* pCell1 = (CGridCellNumeric *)(m_Grid.GetCell(row,col+1));
193         CGridCellNumeric* pCell2 = (CGridCellNumeric *)(m_Grid.GetCell(row,col+2));
194         CGridCellNumeric* pCell3 = (CGridCellNumeric *)(m_Grid.GetCell(row,col+3));
195         pCell1->SetNumber(pParam->getDistribution()->getLocation());
196         pCell2->SetNumber(pParam->getDistribution()->getScale());
197         pCell3->SetNumber(pParam->getDistribution()->getShape());
198 }
```

Here is the call graph for this function:



4.33.4.12 WORD CTrafficConfigDlg::MapDistributionString (CString *dist*) [private]

Definition at line 354 of file TrafficConfigDlg.cpp.

References DIST_CONST, DIST_EXPONENTIAL, DIST_GAMMA, DIST_GEV, DIST_GUMBEL, DIST_LOGNORMAL, DIST_NORMAL, DIST_POISSON, and m_sDistributions.

Referenced by SetParamData().

```
355 {
356         // Note this function reqires the order of the distributions
357         // to not change
358         int i = 0;
359         while(dist != m_sDistributions.GetAt(i))
360                 i++;
361
362         switch(i)
363         {
364                 case 0:         return DIST_EXPONENTIAL;
365                 case 1:         return DIST_LOGNORMAL;
366                 case 2:         return DIST_GAMMA;
367                 case 3:         return DIST_GUMBEL;
368                 case 4:         return DIST_POISSON;
369                 case 5:         return DIST_GEV;
370                 case 6:         return DIST_NORMAL;
371                 case 7:         return DIST_CONST;
372                 default:        return DIST_CONST;
373         }
374 }
```

4.33.4.13 CString CTrafficConfigDlg::MapDistributionID (WORD *distID*) [private]

Definition at line 336 of file TrafficConfigDlg.cpp.

References DIST_CONST, DIST_EXPONENTIAL, DIST_GAMMA, DIST_GEV, DIST_GUMBEL, DIST_LOGNORMAL, DIST_NORMAL, DIST_POISSON, and m_sDistributions.

Referenced by LoadRow().

```
337 {
338         // Note this function reqires the order of the distributions
339         // to not change
340         switch(distID)
341         {
342                 case DIST_EXPONENTIAL:  return m_sDistributions.GetAt(0);
343                 case DIST_LOGNORMAL:    return m_sDistributions.GetAt(1);
344                 case DIST_GAMMA:                return m_sDistributions.GetAt(2);
345                 case DIST_GUMBEL:               return m_sDistributions.GetAt(3);
346                 case DIST_POISSON:              return m_sDistributions.GetAt(4);
347                 case DIST_GEV:                  return m_sDistributions.GetAt(5);
348                 case DIST_NORMAL:               return m_sDistributions.GetAt(6);
349                 case DIST_CONST:                return m_sDistributions.GetAt(7);
350                 default:                                return m_sDistributions.GetAt(7);
351         }
352 }
```

4.33.4.14 void CTrafficConfigDlg::LoadParamsIntoGrid () [private]

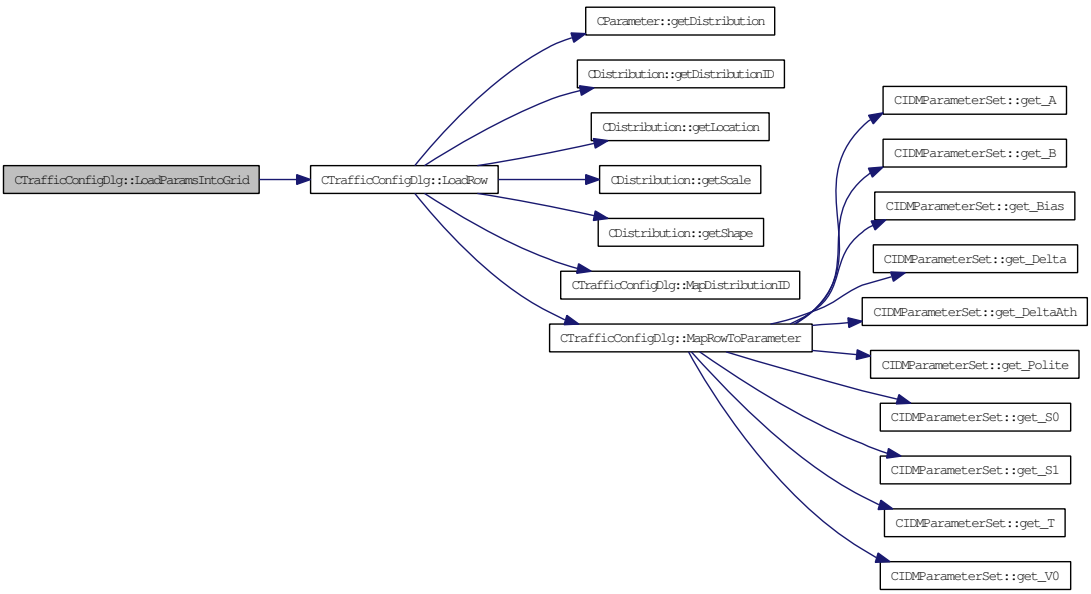
Definition at line 164 of file TrafficConfigDlg.cpp.

References LoadRow(), m_Grid, and m_nFixRows.

Referenced by OnBtnCopy(), OnInitDialog(), and OnSelchangeCmbClassDefine().

```
165 {
166         // First, clear the grid
167 //      CCellRange range(m_nFixRows,m_nFixCols,m_Grid.GetRowCount()-m_nFixRows, m_Grid.GetColu
168 //      m_Grid.ClearCells( range );
169
170         LoadRow(m_nFixRows);
171         LoadRow(m_nFixRows+1);
172         LoadRow(m_nFixRows+2);
173         LoadRow(m_nFixRows+3);
174         LoadRow(m_nFixRows+4);
175         LoadRow(m_nFixRows+5);
176         LoadRow(m_nFixRows+6);
177         LoadRow(m_nFixRows+7);
178         LoadRow(m_nFixRows+8);
179         LoadRow(m_nFixRows+9);
180
181         m_Grid.Refresh();
182 }
```

Here is the call graph for this function:



4.33.4.15 void CTrafficConfigDlg::SetCells () [private]

Definition at line 141 of file TrafficConfigDlg.cpp.

References m_Grid, m_nCols, m_nRows, and m_sDistributions.

Referenced by OnInitDialog().

```
142 {
143         for (int row = 1; row < m_nRows; row++)
144         {
145                 int col = 1;
146                 m_Grid.SetCellType(row,col, RUNTIME_CLASS(CGridCellCombo));
147             CGridCellCombo *pCell = (CGridCellCombo*) m_Grid.GetCell(row,col);
148                 pCell->SetOptions(m_sDistributions);
149                 pCell->SetStyle(CBS_DROPDOWNLIST); //CBS_DROPDOWN, CBS_DROPDOWNLIST, CBS_SIMPLE
150         }
151
152         for (row = 1; row < m_nRows; row++)
153         {
154                 for (int col = 2; col < m_nCols; col++)
155                 {
156                         m_Grid.SetCellType(row,col, RUNTIME_CLASS(CGridCellNumeric));
157                         m_Grid.GetCell(row,col)->SetFormat(DT_CENTER|DT_VCENTER|DT_SINGLELINE|D
158                         CGridCellNumeric *pCell = (CGridCellNumeric*)m_Grid.GetCell(row,col);
159                         pCell->SetFlags(CGridCellNumeric::Real | CGridCellNumeric::Negative);
160                 }
161         }
162 }
```

4.33.4.16 void CTrafficConfigDlg::SetGridHeadings () [private]

Definition at line 126 of file TrafficConfigDlg.cpp.

References m_Grid, m_nCols, m_nRows, m_sColumnHeaders, and m_sRowHeaders.

Referenced by OnInitDialog().

```
127 {
128         int row = 0;
129         int col = 0;
130
131         // Set fixed column text
132         for (col = 1; col < m_nCols; col++)
133                 m_Grid.SetItemText(row,col,m_sColumnHeaders.GetAt(col-1));
134
135         // Set fixed row text
136         col = 0;
137         for (row = 1; row < m_nRows; row++)
138                 m_Grid.SetItemText(row,col,m_sRowHeaders.GetAt(row-1));
139 }
```

4.33.5 Member Data Documentation

4.33.5.1 CComboBox CTrafficConfigDlg::m_cmbVehicleClassCopy

Definition at line 29 of file TrafficConfigDlg.h.

Referenced by DoDataExchange(), OnInitDialog(), and OnSelchangeCmbClassCopy().

4.33.5.2 CComboBox CTrafficConfigDlg::m_cmbVehicleClassDefine

Definition at line 30 of file TrafficConfigDlg.h.

Referenced by DoDataExchange(), OnInitDialog(), and OnSelchangeCmbClassDefine().

4.33.5.3 int CTrafficConfigDlg::m_nFixCols

Definition at line 31 of file TrafficConfigDlg.h.

Referenced by CTrafficConfigDlg(), LoadRow(), OnGridEndEdit(), OnInitDialog(), and SetParamData().

4.33.5.4 int CTrafficConfigDlg::m_nFixRows

Definition at line 32 of file TrafficConfigDlg.h.

Referenced by CTrafficConfigDlg(), LoadParamsIntoGrid(), MapRowToParameter(), and OnInitDialog().

4.33.5.5 int CTrafficConfigDlg::m_nCols

Definition at line 33 of file TrafficConfigDlg.h.

Referenced by CTrafficConfigDlg(), OnInitDialog(), SetCells(), and SetGridHeadings().

4.33.5.6 int CTrafficConfigDlg::m_nRows

Definition at line 34 of file TrafficConfigDlg.h.

Referenced by CTrafficConfigDlg(), OnInitDialog(), SetCells(), and SetGridHeadings().

4.33.5.7 CStringArray CTrafficConfigDlg::m_sDistributions

Definition at line 35 of file TrafficConfigDlg.h.

Referenced by CTrafficConfigDlg(), MapDistributionID(), MapDistributionString(), and SetCells().

4.33.5.8 CIDMParameterSet CTrafficConfigDlg::m_IDMParams_Car

Definition at line 36 of file TrafficConfigDlg.h.

Referenced by MapIDMParamSet(), CEvolveTrafficView::OnConfigTraf(), and OnInitDialog().

4.33.5.9 CIDMParameterSet CTrafficConfigDlg::m_IDMParams_SmallTruck

Definition at line 37 of file TrafficConfigDlg.h.

Referenced by MapIDMParamSet(), and CEvolveTrafficView::OnConfigTraf().

4.33.5.10 CIDMParameterSet CTrafficConfigDlg::m_IDMParams_LargeTruck

Definition at line 38 of file TrafficConfigDlg.h.

Referenced by MapIDMParamSet(), and CEvolveTrafficView::OnConfigTraf().

4.33.5.11 CIDMParameterSet CTrafficConfigDlg::m_IDMParams_Crane

Definition at line 39 of file TrafficConfigDlg.h.

Referenced by MapIDMParamSet(), and CEvolveTrafficView::OnConfigTraf().

4.33.5.12 CIDMParameterSet CTrafficConfigDlg::m_IDMParams_Lowloader

Definition at line 40 of file TrafficConfigDlg.h.

Referenced by MapIDMParamSet(), and CEvolveTrafficView::OnConfigTraf().

4.33.5.13 CGridCtrl CTrafficConfigDlg::m_Grid

Definition at line 42 of file TrafficConfigDlg.h.

Referenced by DoDataExchange(), LoadParamsIntoGrid(), LoadRow(), OnGridEndEdit(), OnInitDialog(), SetCells(), and SetGridHeadings().

4.33.5.14 CIDMParameterSet* CTrafficConfigDlg::m_pCurrentIDMParamSet [private]

Definition at line 75 of file TrafficConfigDlg.h.

Referenced by MapRowToParameter(), OnBtnCopy(), OnInitDialog(), and OnSelchangeCmbClassDefine().

4.33.5.15 CIDMParameterSet* CTrafficConfigDlg::m_pCopyIDMParamSet [private]

Definition at line 76 of file TrafficConfigDlg.h.

Referenced by OnBtnCopy(), and OnSelchangeCmbClassCopy().

4.33.5.16 CStringArray CTrafficConfigDlg::m_sCmbOptions [private]

Definition at line 77 of file TrafficConfigDlg.h.

Referenced by CTrafficConfigDlg(), and OnInitDialog().

4.33.5.17 CStringArray CTrafficConfigDlg::m_sRowHeaders [private]

Definition at line 78 of file TrafficConfigDlg.h.

Referenced by CTrafficConfigDlg(), and SetGridHeadings().

4.33.5.18 CStringArray CTrafficConfigDlg::m_sColumnHeaders

`[private]`

Definition at line 79 of file TrafficConfigDlg.h.

Referenced by CTrafficConfigDlg(), and SetGridHeadings().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/TrafficConfigDlg.h
- D:/~Research/Code/C++/EvolveTraffic/TrafficConfigDlg.cpp

4.34 CWindowToBMP Class Reference

A class for writing a windows content to a BMP file.

`#include <WindowToBMP.h>`

Public Member Functions

- bool Write (CString file, CWnd *oWnd)
- CWindowToBMP ()
- virtual ~CWindowToBMP ()

Private Member Functions

- BOOL WriteWindowToDIB (LPTSTR szFile, CWnd *pWnd)
- HANDLE DDBToDIB (CBitmap &bitmap, DWORD dwCompression, CPalette *pPal)
- BOOL WriteDIB (LPTSTR szFile, HANDLE hDIB)

4.34.1 Detailed Description

A class for writing a windows content to a BMP file.

Definition at line 14 of file WindowToBMP.h.

4.34.2 Constructor & Destructor Documentation

4.34.2.1 CWindowToBMP::CWindowToBMP ()

Definition at line 19 of file WindowToBMP.cpp.

```
20 {
21
22 }
```

4.34.2.2 CWindowToBMP::~CWindowToBMP () [virtual]

Definition at line 24 of file WindowToBMP.cpp.

```
25 {
26
27 }
```

4.34.3 Member Function Documentation

4.34.3.1 bool CWindowToBMP::Write (CString *file*, CWnd * *oWnd*)

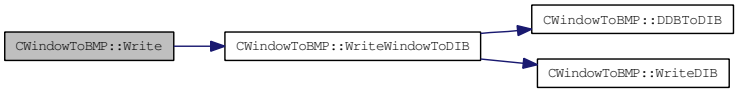
Definition at line 29 of file WindowToBMP.cpp.

References WriteWindowToDIB().

Referenced by CEvolveTrafficView::OnFileSaveImage().

```
30 {
31           char* ch = file.GetBuffer(0);
32           return WriteWindowToDIB( ch, pWnd );
33 }
```

Here is the call graph for this function:



4.34.3.2 BOOL CWindowToBMP::WriteWindowToDIB (LPTSTR *szFile*, CWnd * *pWnd*) [private]

Definition at line 35 of file WindowToBMP.cpp.

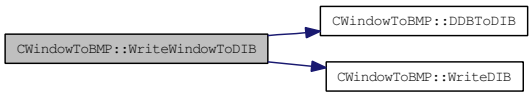
References DDBToDIB(), and WriteDIB().

Referenced by Write().

```
36 {
37         CBitmap         bitmap;
38         CWindowDC       dc(pWnd);
39         CDC             memDC;
40         CRect           rect;
41
42         memDC.CreateCompatibleDC(&dc);
43
44         pWnd->GetWindowRect(rect);
45
46         bitmap.CreateCompatibleBitmap(&dc, rect.Width(),rect.Height() );
47
48         CBitmap* pOldBitmap = memDC.SelectObject(&bitmap);
49         memDC.BitBlt(0, 0, rect.Width(),rect.Height(), &dc, 0, 0, SRCCOPY);
50
51         // Create logical palette if device support a palette
```

```
52          CPalette pal;
53          if( dc.GetDeviceCaps(RASTERCAPS) & RC_PALETTE )
54          {
55                  UINT nSize = sizeof(LOGPALETTE) + (sizeof(PALETTEENTRY) * 256);
56                  LOGPALETTE *pLP = (LOGPALETTE *) new BYTE[nSize];
57                  pLP->palVersion = 0x300;
58
59                  pLP->palNumEntries =
60                          GetSystemPaletteEntries( dc, 0, 255, pLP->palPalEntry );
61
62                  // Create the palette
63                  pal.CreatePalette( pLP );
64
65                  delete[] pLP;
66          }
67
68          memDC.SelectObject(pOldBitmap);
69
70          // Convert the bitmap to a DIB
71          HANDLE hDIB = DDBToDIB( bitmap, BI_RGB, &pal );
72
73          if( hDIB == NULL )
74                  return FALSE;
75
76          // Write it to file
77          WriteDIB( szFile, hDIB );
78
79          // Free the memory allocated by DDBToDIB for the DIB
80          GlobalFree( hDIB );
81          return TRUE;
82  }
```

Here is the call graph for this function:



4.34.3.3 HANDLE CWindowToBMP::DDBToDIB (CBitmap & *bitmap*, DWORD *dwCompression*, CPalette * *pPal*) [private]

Definition at line 112 of file WindowToBMP.cpp.

Referenced by WriteWindowToDIB().

```
113 {
114          BITMAP                  bm;
115          BITMAPINFOHEADER        bi;
116          LPBITMAPINFOHEADER      lpbi;
117          DWORD                   dwLen;
118          HANDLE                  hDIB;
119          HANDLE                  handle;
120          HDC                     hDC;
121          HPALETTE                hPal;
122
123
```

```
124         ASSERT( bitmap.GetSafeHandle() );
125
126         // The function has no arg for bitfields
127         if( dwCompression == BI_BITFIELDS )
128                 return NULL;
129
130         // If a palette has not been supplied use defaul palette
131         hPal = (HPALETTE) pPal->GetSafeHandle();
132         if (hPal==NULL)
133                 hPal = (HPALETTE) GetStockObject(DEFAULT_PALETTE);
134
135         // Get bitmap information
136         bitmap.GetObject(sizeof(bm),(LPSTR)&bm);
137
138         // Initialize the bitmapinfoheader
139         bi.biSize               = sizeof(BITMAPINFOHEADER);
140         bi.biWidth              = bm.bmWidth;
141         bi.biHeight             = bm.bmHeight;
142         bi.biPlanes             = 1;
143         bi.biBitCount           = bm.bmPlanes * bm.bmBitsPixel;
144         bi.biCompression        = dwCompression;
145         bi.biSizeImage          = 0;
146         bi.biXPelsPerMeter      = 0;
147         bi.biYPelsPerMeter      = 0;
148         bi.biClrUsed            = 0;
149         bi.biClrImportant       = 0;
150
151         // Compute the size of the  infoheader and the color table
152         int nColors = (1 << bi.biBitCount);
153         if( nColors > 256 )
154                 nColors = 0;
155         dwLen  = bi.biSize + nColors * sizeof(RGBQUAD);
156
157         // We need a device context to get the DIB from
158         hDC = GetDC(NULL);
159         hPal = SelectPalette(hDC,hPal,FALSE);
160         RealizePalette(hDC);
161
162         // Allocate enough memory to hold bitmapinfoheader and color table
163         hDIB = GlobalAlloc(GMEM_FIXED,dwLen);
164
165         if (!hDIB){
166                 SelectPalette(hDC,hPal,FALSE);
167                 ReleaseDC(NULL,hDC);
168                 return NULL;
169         }
170
171         lpbi = (LPBITMAPINFOHEADER)hDIB;
172
173         *lpbi = bi;
174
175         // Call GetDIBits with a NULL lpBits param, so the device driver
176         // will calculate the biSizeImage field
177         GetDIBits(hDC, (HBITMAP)bitmap.GetSafeHandle(), 0L, (DWORD)bi.biHeight,
178                         (LPBYTE)NULL, (LPBITMAPINFO)lpbi, (DWORD)DIB_RGB_COLORS);
179
180         bi = *lpbi;
181
182         // If the driver did not fill in the biSizeImage field, then compute it
183         // Each scan line of the image is aligned on a DWORD (32bit) boundary
184         if (bi.biSizeImage == 0){
185                 bi.biSizeImage = ((((bi.biWidth * bi.biBitCount) + 31) & ~31) / 8)
```

```
186                                                 * bi.biHeight;
187
188                 // If a compression scheme is used the result may infact be larger
189                 // Increase the size to account for this.
190                 if (dwCompression != BI_RGB)
191                         bi.biSizeImage = (bi.biSizeImage * 3) / 2;
192         }
193
194         // Realloc the buffer so that it can hold all the bits
195         dwLen += bi.biSizeImage;
196         if (handle = GlobalReAlloc(hDIB, dwLen, GMEM_MOVEABLE))
197                 hDIB = handle;
198         else{
199                 GlobalFree(hDIB);
200
201                 // Reselect the original palette
202                 SelectPalette(hDC,hPal,FALSE);
203                 ReleaseDC(NULL,hDC);
204                 return NULL;
205         }
206
207         // Get the bitmap bits
208         lpbi = (LPBITMAPINFOHEADER)hDIB;
209
210         // FINALLY get the DIB
211         BOOL bGotBits = GetDIBits( hDC, (HBITMAP)bitmap.GetSafeHandle(),
212                                 0L,                                 // Start scan line
213                                 (DWORD)bi.biHeight,                 // # of scan lines
214                                 (LPBYTE)lpbi                        // address for bitmap bits
215                                 + (bi.biSize + nColors * sizeof(RGBQUAD)),
216                                 (LPBITMAPINFO)lpbi,                 // address of bitmapinfo
217                                 (DWORD)DIB_RGB_COLORS);             // Use RGB for color table
218
219         if( !bGotBits )
220         {
221                 GlobalFree(hDIB);
222
223                 SelectPalette(hDC,hPal,FALSE);
224                 ReleaseDC(NULL,hDC);
225                 return NULL;
226         }
227
228         SelectPalette(hDC,hPal,FALSE);
229         ReleaseDC(NULL,hDC);
230         return hDIB;
231 }
```

4.34.3.4 BOOL CWindowToBMP::WriteDIB (LPTSTR *szFile*, HANDLE *hDIB*) [private]

Definition at line 84 of file WindowToBMP.cpp.

Referenced by WriteWindowToDIB().

```
85 {
86         BITMAPFILEHEADER hdr;
87         LPBITMAPINFOHEADER lpbi;
88
89         if (!hDIB)
90         return FALSE;
```

```
91
92          CFile file;
93          if (!file.Open (szFile, CFile::modeWrite | CFile::modeCreate))
94          return FALSE;
95
96          lpbi = (LPBITMAPINFOHEADER)hDIB;
97          int nColors = lpbi->biBitCount;
98
99          hdr.bfType = ((WORD) ('M' << 8) | 'B');
100          hdr.bfSize = GlobalSize (hDIB) + sizeof( hdr );
101          hdr.bfReserved1 = 0;
102          hdr.bfReserved2 = 0;
103          hdr.bfOffBits = (DWORD) (sizeof( hdr ) + sizeof(BITMAPINFOHEADER));
104
105          file.Write( &hdr, sizeof(hdr) );
106          file.Write( lpbi, GlobalSize(hDIB) );
107          file.Close ();
108
109          return TRUE;
110 }
```

The documentation for this class was generated from the following files:

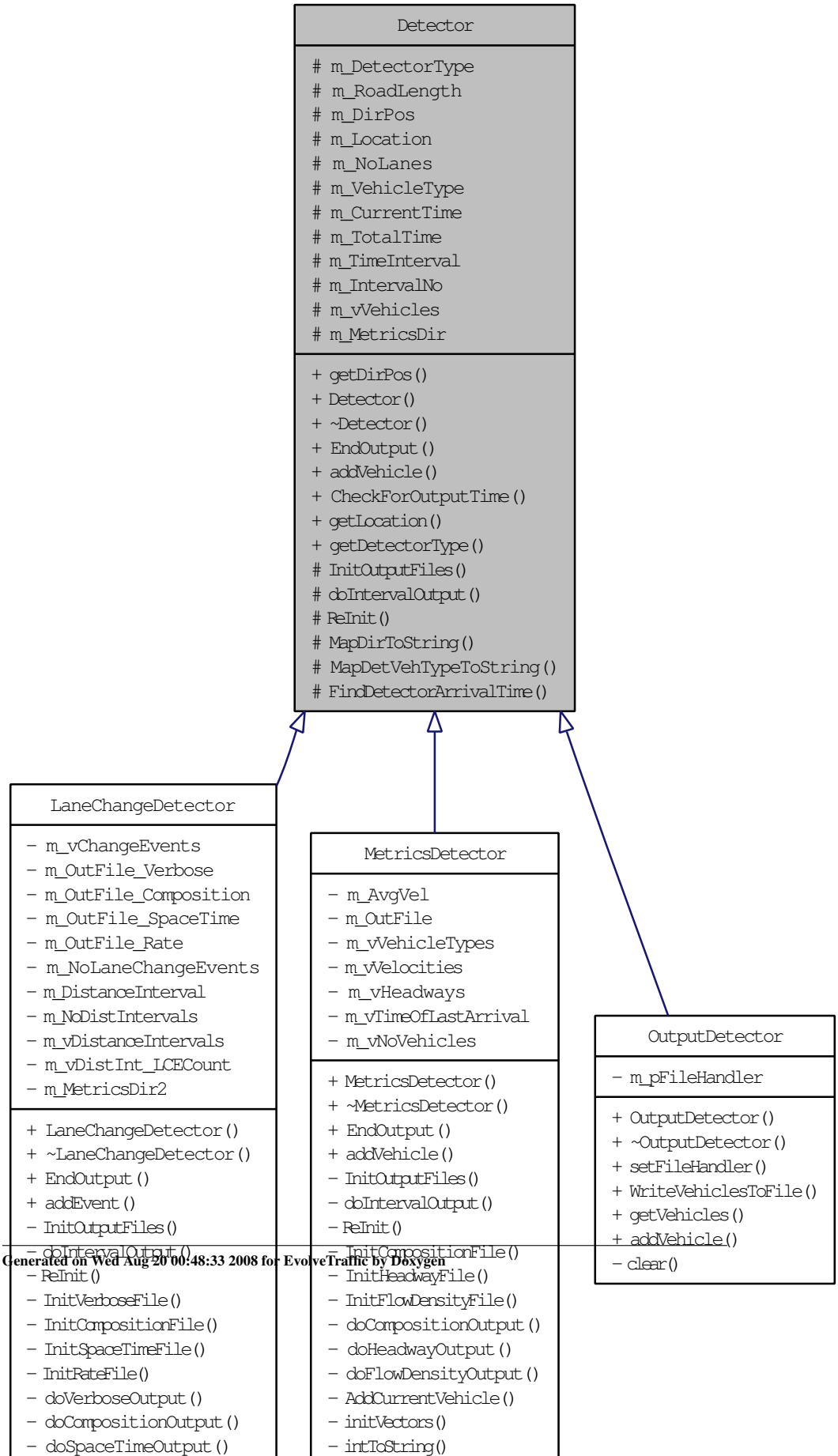
- D:/~Research/Code/C++/EvolveTraffic/WindowToBMP.h
- D:/~Research/Code/C++/EvolveTraffic/WindowToBMP.cpp

4.35 Detector Class Reference

A base class from which other, specific detectors are derived.

`#include <Detector.h>`

Inheritance diagram for Detector:



Public Member Functions

- bool getDirPos ()
- Detector ()
- virtual ~Detector ()
- virtual void EndOutput ()
- virtual void addVehicle (Vehicle *pVeh, double curTime)
- void CheckForOutputTime (double step)

  Checks if it is time for the detector to output.

- int getLocation ()
- WORD getDetectorType ()

Protected Member Functions

- virtual void InitOutputFiles ()
- virtual void doIntervalOutput ()
- virtual void ReInit ()
- std::string MapDirToString (bool DirPos)
- std::string MapDetVehTypeToString (WORD VehType)
- double FindDetectorArrivalTime (Vehicle *pVeh, const double curTime)

  Finds the arrival time of a vehicle.

Protected Attributes

- WORD m_DetectorType
- int m_RoadLength
- bool m_DirPos
- int m_Location
- int m_NoLanes
- WORD m_VehicleType
- double m_CurrentTime
- double m_TotalTime
- double m_TimeInterval
- int m_IntervalNo
- std::vector< Vehicle * > m_vVehicles
- std::string m_MetricsDir

4.35.1 Detailed Description

A base class from which other, specific detectors are derived.

Definition at line 21 of file Detector.h.

4.35.2 Constructor & Destructor Documentation

4.35.2.1 Detector::Detector ()

Definition at line 19 of file Detector.cpp.

References m_CurrentTime, m_IntervalNo, m_MetricsDir, and m_TotalTime.

```
20 {
21         m_TotalTime = 0.0;
22         m_CurrentTime = 0.0;
23         m_IntervalNo = 0;
24         m_MetricsDir = "C:\\EvolveTraffic\\Metrics\\";
25 }
```

4.35.2.2 Detector::~Detector () [virtual]

Definition at line 27 of file Detector.cpp.

```
28 {
29
30 }
```

4.35.3 Member Function Documentation

4.35.3.1 bool Detector::getDirPos ()

Definition at line 95 of file Detector.cpp.

References m_DirPos.

Referenced by Lane::Lane().

```
96 {
97         return m_DirPos;
98 }
```

4.35.3.2 virtual void Detector::EndOutput () [inline, virtual]

Reimplemented in LaneChangeDetector, and MetricsDetector.

Definition at line 28 of file Detector.h.

```
28 {};
```

4.35.3.3 virtual void Detector::addVehicle (Vehicle * *pVeh*, double *curTime*) [inline, virtual]

Reimplemented in MetricsDetector, and OutputDetector.

Definition at line 29 of file Detector.h.

```
29 {};
```

4.35.3.4 void Detector::CheckForOutputTime (double *step*)

Checks if it is time for the detector to output.

Parameters:

step The timestep

Given the timestep and the time at which the detector should output, this function is called each iteration. When the output time has been reached, the detector outputs

Definition at line 39 of file Detector.cpp.

References doIntervalOutput(), m_CurrentTime, m_IntervalNo, m_TimeInterval, and m_TotalTime.

```
40 {
41         m_TotalTime += step;
42         m_CurrentTime += step;
43         if(m_CurrentTime >= m_TimeInterval)
44         {
45                 m_CurrentTime -= m_TimeInterval;        // We reset the timer
46                 m_IntervalNo++;                                         // increase the index
47                 doIntervalOutput();                                     // we call the virtual
48         }
49 }
```

Here is the call graph for this function:

4.35.3.5 int Detector::getLocation ()

Definition at line 90 of file Detector.cpp.

References m_Location.

Referenced by Lane::CheckDetectors().

```
91 {
92         return m_Location;
93 }
```

4.35.3.6 WORD Detector::getDetectorType ()

Definition at line 100 of file Detector.cpp.

References m_DetectorType.

Referenced by Lane::Lane().

```
101 {
102          return m_DetectorType;
103 }
```

4.35.3.7 virtual void Detector::InitOutputFiles () [inline, protected, virtual]

Reimplemented in LaneChangeDetector, and MetricsDetector.

Definition at line 36 of file Detector.h.

```
36 {};
```

4.35.3.8 virtual void Detector::doIntervalOutput () [inline, protected, virtual]

Reimplemented in LaneChangeDetector, and MetricsDetector.

Definition at line 37 of file Detector.h.

Referenced by CheckForOutputTime().

```
37 {};
```

4.35.3.9 virtual void Detector::ReInit () [inline, protected, virtual]

Reimplemented in LaneChangeDetector, and MetricsDetector.

Definition at line 38 of file Detector.h.

```
38 {};
```

4.35.3.10 std::string Detector::MapDirToString (bool *DirPos*) [protected]

Definition at line 65 of file Detector.cpp.

References m_DirPos.

Referenced by MetricsDetector::InitCompositionFile(), LaneChangeDetector::InitCompositionFile(), MetricsDetector::InitFlowDensityFile(), MetricsDetector::InitHeadwayFile(), LaneChangeDetector::InitRateFile(), LaneChangeDetector::InitSpaceTimeFile(), and LaneChangeDetector::InitVerboseFile().

```
66 {
67         if(m_DirPos)
68                 return "Pos";
69         else
70                 return "Neg";
71 }
```

4.35.3.11 std::string Detector::MapDetVehTypeToString (WORD *VehType*) [protected]

Definition at line 51 of file Detector.cpp.

References METRICS_VEH_ALL, METRICS_VEH_CAR, METRICS_VEH_CRANE, METRICS_VEH_LARGETRUCK, METRICS_VEH_LOWLOADER, and METRICS_VEH_SMALLTRUCK.

Referenced by MetricsDetector::InitCompositionFile(), LaneChangeDetector::InitCompositionFile(), MetricsDetector::InitFlowDensityFile(), MetricsDetector::InitHeadwayFile(), LaneChangeDetector::InitRateFile(), LaneChangeDetector::InitSpaceTimeFile(), and LaneChangeDetector::InitVerboseFile().

```
52 {
53         switch(DetVehType)
54         {
55                 case METRICS_VEH_ALL:                   return "All";
56                 case METRICS_VEH_CAR:                   return "Car";
57                 case METRICS_VEH_SMALLTRUCK:    return "ST";
58                 case METRICS_VEH_LARGETRUCK:    return "LT";
59                 case METRICS_VEH_CRANE:                 return "Crane";
60                 case METRICS_VEH_LOWLOADER:             return "LL";
61                 default: return "All";
62         }
63 }
```

4.35.3.12 double Detector::FindDetectorArrivalTime (Vehicle * *pVeh*, const double *curTime*) [protected]

Finds the arrival time of a vehicle.

Parameters:

pVeh The vehicle

curTime The current time

Returns:

The time of arrival

This function finds the time that a vehicle arrived at a detector by interpolating between simulation steps

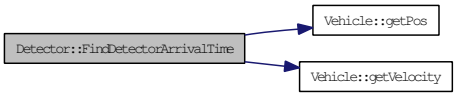
Definition at line 82 of file Detector.cpp.

References Vehicle::getPos(), Vehicle::getVelocity(), and m_Location.

Referenced by MetricsDetector::AddCurrentVehicle(), and OutputDetector::addVehicle().

```
83 {
84         double distBeyondDetector = pVeh->getPos() - m_Location;
85         double ArrivalTime = curTime - distBeyondDetector/pVeh->getVelocity();
86
87         return ArrivalTime;
88 }
```

Here is the call graph for this function:



4.35.4 Member Data Documentation

4.35.4.1 WORD Detector::m_DetectorType [protected]

Definition at line 44 of file Detector.h.

Referenced by MetricsDetector::AddCurrentVehicle(), MetricsDetector::doIntervalOutput(), getDetectorType(), MetricsDetector::InitOutputFiles(), LaneChangeDetector::LaneChangeDetector(), and MetricsDetector::MetricsDetector().

4.35.4.2 int Detector::m_RoadLength [protected]

Definition at line 45 of file Detector.h.

Referenced by MetricsDetector::InitCompositionFile(), MetricsDetector::InitFlowDensityFile(), MetricsDetector::InitHeadwayFile(), LaneChangeDetector::LaneChangeDetector(), and MetricsDetector::MetricsDetector().

4.35.4.3 bool Detector::m_DirPos [protected]

Definition at line 46 of file Detector.h.

Referenced by getDirPos(), MetricsDetector::InitCompositionFile(), LaneChangeDetector::InitCompositionFile(), MetricsDetector::InitFlowDensityFile(), MetricsDetector::InitHeadwayFile(), LaneChangeDetector::InitRateFile(), LaneChangeDetector::InitSpaceTimeFile(), LaneChangeDetector::InitVerboseFile(), LaneChangeDetector::LaneChangeDetector(), MapDirToString(), MetricsDetector::MetricsDetector(), and OutputDetector::OutputDetector().

4.35.4.4 int Detector::m_Location [protected]

Definition at line 47 of file Detector.h.

Referenced by FindDetectorArrivalTime(), getLocation(), MetricsDetector::InitCompositionFile(), MetricsDetector::InitFlowDensityFile(), MetricsDetector::InitHeadwayFile(), MetricsDetector::MetricsDetector(), and OutputDetector::OutputDetector().

4.35.4.5 int Detector::m_NoLanes [protected]

Definition at line 48 of file Detector.h.

Referenced by MetricsDetector::doFlowDensityOutput(), MetricsDetector::doHeadwayOutput(), MetricsDetector::InitFlowDensityFile(), MetricsDetector::InitHeadwayFile(), MetricsDetector::initVectors(), MetricsDetector::MetricsDetector(), and MetricsDetector::ReInit().

4.35.4.6 WORD Detector::m_VehicleType [protected]

Definition at line 49 of file Detector.h.

Referenced by MetricsDetector::AddCurrentVehicle(), LaneChangeDetector::addEvent(), MetricsDetector::addVehicle(), MetricsDetector::doIntervalOutput(), LaneChangeDetector::EndOutput(), MetricsDetector::InitCompositionFile(), LaneChangeDetector::InitCompositionFile(), MetricsDetector::InitFlowDensityFile(), MetricsDetector::InitHeadwayFile(), MetricsDetector::InitOutputFiles(), LaneChangeDetector::InitOutputFiles(), LaneChangeDetector::InitRateFile(), LaneChangeDetector::InitSpaceTimeFile(), LaneChangeDetector::InitVerboseFile(), LaneChangeDetector::LaneChangeDetector(), and MetricsDetector::MetricsDetector().

4.35.4.7 double Detector::m_CurrentTime [protected]

Definition at line 50 of file Detector.h.

Referenced by CheckForOutputTime(), Detector(), and MetricsDetector::MetricsDetector().

4.35.4.8 double Detector::m_TotalTime [protected]

Definition at line 51 of file Detector.h.

Referenced by CheckForOutputTime(), Detector(), LaneChangeDetector::doCompositionOutput(), and LaneChangeDetector::doRateOutput().

4.35.4.9 double Detector::m_TimeInterval [protected]

Definition at line 52 of file Detector.h.

Referenced by CheckForOutputTime(), MetricsDetector::doFlowDensityOutput(), LaneChangeDetector::doRateOutput(), LaneChangeDetector::LaneChangeDetector(), and MetricsDetector::MetricsDetector().

4.35.4.10 int Detector::m_IntervalNo [protected]

Definition at line 53 of file Detector.h.

Referenced by CheckForOutputTime(), Detector(), MetricsDetector::doFlowDensityOutput(), and MetricsDetector::MetricsDetector().

4.35.4.11 std::vector<Vehicle*> Detector::m_vVehicles [protected]

Definition at line 54 of file Detector.h.

Referenced by OutputDetector::addVehicle(), OutputDetector::clear(), and OutputDetector::WriteVehiclesToFile().

4.35.4.12 std::string Detector::m_MetricsDir [protected]

Definition at line 55 of file Detector.h.

Referenced by Detector(), MetricsDetector::InitCompositionFile(), LaneChangeDetector::InitCompositionFile(), MetricsDetector::InitFlowDensityFile(), MetricsDetector::InitHeadwayFile(), LaneChangeDetector::InitRateFile(), LaneChangeDetector::InitSpaceTimeFile(), LaneChangeDetector::InitVerboseFile(), LaneChangeDetector::LaneChangeDetector(), and MetricsDetector::MetricsDetector().

The documentation for this class was generated from the following files:

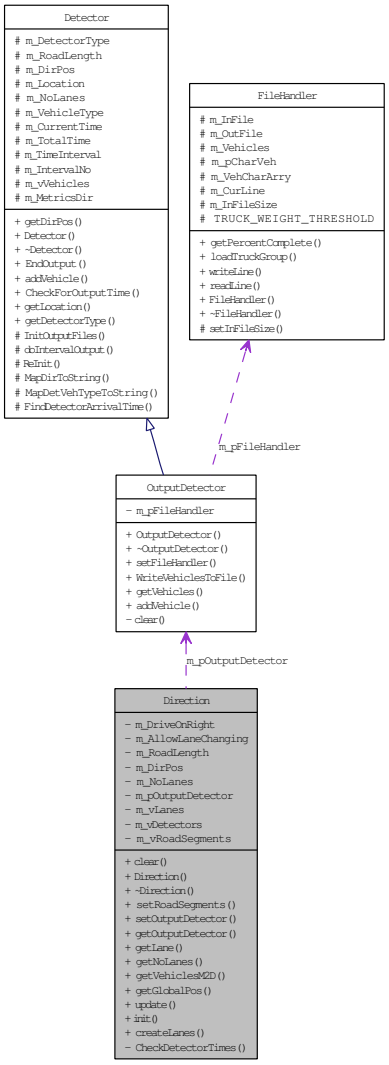
- D:/~Research/Code/C++/EvolveTraffic/Detector.h
- D:/~Research/Code/C++/EvolveTraffic/Detector.cpp

4.36 Direction Class Reference

A class representing a direction in a road.

`#include <Direction.h>`

Collaboration diagram for Direction:



Public Member Functions

- void clear ()

  Clears the direction so it can be used in another simulation.

- Direction ()

  Default constructor.

- virtual ~Direction ()

  Default destructor.

- void setRoadSegments (std::vector< RoadSegment * > segments)
- void setOutputDetector (OutputDetector *OutputDet)

  Sets the direction's output detector.

- OutputDetector * getOutputDetector ()

  Gets the direction's output detector.

- Lane & getLane (int i)

  Gets a specific lane.

- int getNoLanes ()

  Gets the number of lanes.

- M2D getVehiclesM2D ()

  Gets all the vehicles on the road.

- M2D getGlobalPos ()

  Updates all the vehicle positions to take account of direction.

- bool update (const double step, const double curTime)

  Handles the updating of all the lanes in the direction.

- void init (bool DirPos, OutputDetector *OutDet, bool AllowLaneChange, int RoadLength, bool DriveOnRight)

  Initialises the direction.

- void createLanes (int iFirstLane, std::vector< int > vLaneLengths, std::vector< Detector * > detectors, std::vector< RoadSegment * > segments)

  Creates all the lanes running in a particular direction.

Private Member Functions

- void CheckDetectorTimes (double step)

  Checks all the detectors for their output times.

Private Attributes

- bool m_DriveOnRight
- bool m_AllowLaneChanging
- double m_RoadLength
- bool m_DirPos
- int m_NoLanes
- OutputDetector * m_pOutputDetector
- std::vector< Lane > m_vLanes
- std::vector< Detector * > m_vDetectors
- std::vector< RoadSegment * > m_vRoadSegments

4.36.1 Detailed Description

A class representing a direction in a road.

Definition at line 13 of file Direction.h.

4.36.2 Constructor & Destructor Documentation

4.36.2.1 Direction::Direction ()

Default constructor.

Definition at line 6 of file Direction.cpp.

References m_pOutputDetector.

```
7 {
8         m_pOutputDetector = NULL;
9 }
```

4.36.2.2 Direction::~Direction () [virtual]

Default destructor.

Definition at line 12 of file Direction.cpp.

References m_vDetectors, and m_vRoadSegments.

```
13 {
14         int i;
15
16         // Delete all of this direction's metrics detectors
17         for(i = 0; i < m_vDetectors.size(); i++)
18                 delete m_vDetectors.at(i);
19
20
21         // Delete all of this direction's special segments
22         for(i = 0; i < m_vRoadSegments.size(); i++)
23                 delete m_vRoadSegments.at(i);
24
25 }
```

4.36.3 Member Function Documentation

4.36.3.1 void Direction::clear ()

Clears the direction so it can be used in another simulation.

This function clears all of the direction's information so that it may be used in another simulation without re-instanciating.

Definition at line 220 of file Direction.cpp.

References m_pOutputDetector, m_vDetectors, m_vLanes, and m_vRoadSegments.

Referenced by Road::clear().

```
221 {
222         int i;
223
224         for(i = 0; i < m_vLanes.size(); i++)
225                 m_vLanes.at(i).clear();
226
227         m_vLanes.clear();
228
229         delete m_pOutputDetector; // Delete this direction's output detector
230
231         // Delete all of this direction's metrics detectors
232         for(i = 0; i < m_vDetectors.size(); i++)
233                 delete m_vDetectors.at(i);
234
235         m_vDetectors.clear();
236
237         // Delete all of this direction's special segments
238         for(i = 0; i < m_vRoadSegments.size(); i++)
239                 delete m_vRoadSegments.at(i);
240
241         m_vRoadSegments.clear();
242 }
```

4.36.3.2 void Direction::setRoadSegments (std::vector< RoadSegment * > *segments*)

4.36.3.3 void Direction::setOutputDetector (OutputDetector * *OutputDet*)

Sets the direction's output detector.

Parameters:

OutputDet The direction's output detector

Definition at line 196 of file Direction.cpp.

References m_pOutputDetector.

```
197 {
198         m_pOutputDetector = OutputDet;
199 }
```

4.36.3.4 OutputDetector * Direction::getOutputDetector ()

Gets the direction's output detector.

Returns:

The direction's output detector

Definition at line 187 of file Direction.cpp.

References m_pOutputDetector.

```
188 {
189         return m_pOutputDetector;
190 }
```

4.36.3.5 Lane & Direction::getLane (int *i*)

Gets a specific lane.

Parameters:

i The lane to choose

Returns:

The chosen lane

Definition at line 178 of file Direction.cpp.

References m_vLanes.

Referenced by Road::init().

```
179 {
180         return m_vLanes.at(i);
181 }
```

4.36.3.6 int Direction::getNoLanes ()

Gets the number of lanes.

Returns:

The number of lanes

Definition at line 168 of file Direction.cpp.

References m_NoLanes.

Referenced by Road::init().

```
169 {
170         return m_NoLanes;
171 }
```

4.36.3.7 M2D Direction::getVehiclesM2D ()

Gets all the vehicles on the road.

Returns:

All the vehicles on the road

Definition at line 151 of file Direction.cpp.

References m_NoLanes, and m_vLanes.

Referenced by getGlobalPos().

```
152 {
153         // returns vehicles for position from the start of the lane extraction
154         M2D vVeh2D;
155
156         for(int i = 0; i < m_NoLanes; i++)
157         {
158                 std::vector<Vehicle*> temp = m_vLanes[i].getPos();
159                 vVeh2D.push_back(temp);
160         }
161         return vVeh2D;
162 }
```

4.36.3.8 M2D Direction::getGlobalPos ()

Updates all the vehicle positions to take account of direction.

Returns:

All adjusted vehicle positions

This function updates the position that a vehicle should have on screen

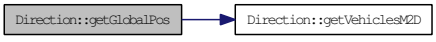
Definition at line 130 of file Direction.cpp.

References getVehiclesM2D(), m_DirPos, m_NoLanes, and m_vLanes.

Referenced by Road::getVehicles().

```
131 {
132         // this modifies the lane position to take accound of direction
133         M2D vVeh2D = getVehiclesM2D();
134
135         if(!m_DirPos)   // if in negative x-direction
136         {
137                 for(int i = 0; i < m_NoLanes; i++)
138                 {
139                         int d2 = vVeh2D[i].size();
140                         for(int j = 0; j < d2; j++)
141                                 vVeh2D[i][j]->setRoadPos( m_vLanes[i].getLength() - vVeh2D[i][
142                 }
143         }
144         return vVeh2D;
145 }
```

Here is the call graph for this function:



4.36.3.9 bool Direction::update (const double *step*, const double *curTime*)

Handles the updating of all the lanes in the direction.

Parameters:

step The timestep

curTime The current simulation time

Returns:

Whether or not the direction is empty

This function updates all the lanes that are running in a particular direction. If necessary, the elements in the direction's output detector are also output.

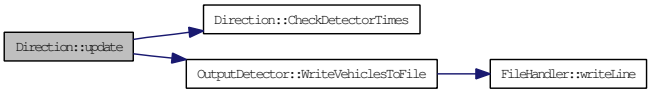
Definition at line 37 of file Direction.cpp.

References CheckDetectorTimes(), m_NoLanes, m_pOutputDetector, m_vLanes, and OutputDetector::WriteVehiclesToFile().

Referenced by Road::update().

```
38 {
39         CheckDetectorTimes(step);       // check if it's time to output what we have
40         bool DirectionEmpty = false;
41         int NoEmptyLanes = 0;
42         for(int i = 0; i < m_NoLanes; i++)
43         {
44                 bool laneEmpty = m_vLanes[i].update(step, curTime);
45                 if(laneEmpty)
46                         NoEmptyLanes++;
47         }
48
49         // output any vehicles past the detector
50         m_pOutputDetector->WriteVehiclesToFile();
51
52         if(NoEmptyLanes == m_NoLanes)
53                 DirectionEmpty = true;
54
55         return DirectionEmpty;
56 }
```

Here is the call graph for this function:



4.36.3.10 void Direction::init (bool *DirPos*, OutputDetector * *pOutDet*, bool *AllowLaneChange*, int *RoadLength*, bool *DriveOnRight*)

Initialises the direction.

Parameters:

DirPos Whether or not the direction is in a positive direction

pOutDet The direction's output detector

AllowLaneChange Whether or not lane changing is allowed

RoadLength The length of the road

DriveOnRight Whether or not vehicles drive on the right

Definition at line 66 of file Direction.cpp.

References m_AllowLaneChanging, m_DirPos, m_DriveOnRight, m_pOutputDetector, and m_RoadLength.

Referenced by Road::init().

```
67 {
68         m_DirPos = DirPos;
69         m_pOutputDetector = pOutDet;
70         m_AllowLaneChanging = AllowLaneChange;
71         m_RoadLength = RoadLength;
72         m_DriveOnRight = DriveOnRight;
73 }
```

4.36.3.11 void Direction::createLanes (int *iFirstLane*, std::vector< int > *vLaneLengths*, std::vector< Detector * > *detectors*, std::vector< RoadSegment * > *segments*)

Creates all the lanes running in a particular direction.

Parameters:

iFirstLane The index of the first lane in this direction

vLaneLengths The number of lanes to create

detectors The detectors in the direction

segments The segments in this direction

This function handles the creating of lanes that are running in a certain direction and sets up the links between a lane and its two adjacent lanes. In addition, all necessary detectors and/or road segments are passed to the lane

Definition at line 86 of file Direction.cpp.

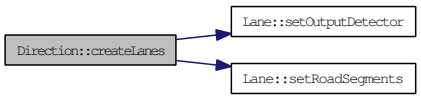
References m_AllowLaneChanging, m_DirPos, m_DriveOnRight, m_NoLanes, m_pOutputDetector, m_vDetectors, m_vLanes, m_vRoadSegments, Lane::setOutputDetector(), and Lane::setRoadSegments().

Referenced by Road::init().

```
88 {
89         m_NoLanes = vLaneLengths.size();
90         m_vDetectors = detectors;
91         m_vRoadSegments = segments;
```

```
92
93          for(int i = 0; i < m_NoLanes; i++)
94          {
95                  Lane lane(iFirstLane + i, vLaneLengths[i], m_vDetectors, m_DirPos, m_AllowLaneCh
96                  lane.setOutputDetector(m_pOutputDetector);
97                  lane.setRoadSegments(m_vRoadSegments);
98                  m_vLanes.push_back(lane);
99          }
100
101          // set adjacent lanes if more than 1 lane
102          if(m_NoLanes > 1)
103          {
104                  // If(DRIVE_ON_RIGHT), lanes are numbered from the BTM of the screen as:
105                  // Dir Pos - 0, 1, 2, 3 -  Dir Neg - 4, 5, 6, 7
106                  // else lanes are numbered from the TOP of the screen
107                  if(m_DriveOnRight)
108                  {
109                          for(i = 0; i < m_NoLanes - 1; i++)
110                                  m_vLanes.at(i).setLeftLane(&m_vLanes.at(i+1));  // next lane up
111                          for(i = 1; i < m_NoLanes; i++)
112                                  m_vLanes.at(i).setRightLane(&m_vLanes.at(i-1)); // one down the
113                  }
114                  else
115                  {
116                          for(i = 0; i < m_NoLanes - 1; i++)
117                                  m_vLanes.at(i).setRightLane(&m_vLanes.at(i+1)); // next lane do
118                          for(i = 1; i < m_NoLanes; i++)
119                                  m_vLanes.at(i).setLeftLane(&m_vLanes.at(i-1));  // next one up
120                  }
121          }
122 }
```

Here is the call graph for this function:



4.36.3.12 void Direction::CheckDetectorTimes (double *step*) [private]

Checks all the detectors for their output times.

Parameters:

step The timestep

This function checks, for each detector, whether it is time for that detector to gather information and output.

Definition at line 208 of file Direction.cpp.

References m_vDetectors.

Referenced by update().

```
209 {
210         for(int i = 0; i < m_vDetectors.size(); i++)
211         {
212                 m_vDetectors.at(i)->CheckForOutputTime(step);
213         }
214 }
```

4.36.4 Member Data Documentation

4.36.4.1 bool Direction::m_DriveOnRight [private]

Definition at line 39 of file Direction.h.

Referenced by createLanes(), and init().

4.36.4.2 bool Direction::m_AllowLaneChanging [private]

Definition at line 40 of file Direction.h.

Referenced by createLanes(), and init().

4.36.4.3 double Direction::m_RoadLength [private]

Definition at line 41 of file Direction.h.

Referenced by init().

4.36.4.4 bool Direction::m_DirPos [private]

Definition at line 42 of file Direction.h.

Referenced by createLanes(), getGlobalPos(), and init().

4.36.4.5 int Direction::m_NoLanes [private]

Definition at line 43 of file Direction.h.

Referenced by createLanes(), getGlobalPos(), getNoLanes(), getVehiclesM2D(), and update().

4.36.4.6 OutputDetector* Direction::m_pOutputDetector [private]

Definition at line 45 of file Direction.h.

Referenced by clear(), createLanes(), Direction(), getOutputDetector(), init(), setOutputDetector(), and update().

4.36.4.7 std::vector<Lane> Direction::m_vLanes [private]

Definition at line 46 of file Direction.h.

Referenced by clear(), createLanes(), getGlobalPos(), getLane(), getVehiclesM2D(), and update().

4.36.4.8 std::vector<Detector*> Direction::m_vDetectors [private]

Definition at line 47 of file Direction.h.

Referenced by CheckDetectorTimes(), clear(), createLanes(), and ~Direction().

4.36.4.9 std::vector<RoadSegment*> Direction::m_vRoadSegments [private]

Definition at line 48 of file Direction.h.

Referenced by clear(), createLanes(), and ~Direction().

The documentation for this class was generated from the following files:

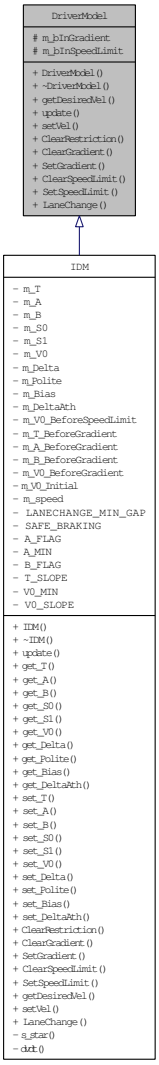
- D:/~Research/Code/C++/EvolveTraffic/Direction.h
- D:/~Research/Code/C++/EvolveTraffic/Direction.cpp

4.37 DriverModel Class Reference

A base class for representing driver models.

`#include <DriverModel.h>`

Inheritance diagram for DriverModel:



Public Member Functions

- DriverModel ()

  Default Constructor.

- virtual ~DriverModel ()

  Default Destructor.

- virtual double getDesiredVel ()=0
- virtual double update (double vel, double dist)=0
- virtual void setVel (double vel)=0

- virtual void ClearRestriction ()
- virtual void ClearGradient ()
- virtual void SetGradient (double gradient)
- virtual void ClearSpeedLimit ()
- virtual void SetSpeedLimit (double limit)
- virtual double LaneChange (double GapToFront, double GapToBack, double FrontChangeAccel, double CurrentBackAccel, double ProposedBackAccel, bool overtake)=0

Protected Attributes

- bool m_bInGradient
- bool m_bInSpeedLimit

4.37.1 Detailed Description

A base class for representing driver models.

Definition at line 10 of file DriverModel.h.

4.37.2 Constructor & Destructor Documentation

4.37.2.1 DriverModel::DriverModel ()

Default Constructor.

Definition at line 6 of file DriverModel.cpp.

References m_bInGradient, and m_bInSpeedLimit.

```
7 {
8         m_bInGradient = false;
9         m_bInSpeedLimit = false;
10 }
```

4.37.2.2 DriverModel::~DriverModel () [virtual]

Default Destructor.

Definition at line 12 of file DriverModel.cpp.

```
12                          {
13
14 }
```

4.37.3 Member Function Documentation

4.37.3.1 virtual double DriverModel::getDesiredVel () [pure virtual]

Implemented in IDM.

Referenced by Vehicle::getDataString(), and Vehicle::getDesiredVel().

4.37.3.2 virtual double DriverModel::update (double *vel*, double *dist*) [pure virtual]

Implemented in IDM.

Referenced by Vehicle::calcAccel().

4.37.3.3 virtual void DriverModel::setVel (double *vel*) [pure virtual]

Implemented in IDM.

Referenced by Vehicle::calcAccel().

4.37.3.4 virtual void DriverModel::ClearRestriction () [inline, virtual]

Reimplemented in IDM.

Definition at line 20 of file DriverModel.h.

Referenced by Lane::CheckRoadSegments().

```
20 {};
```

4.37.3.5 virtual void DriverModel::ClearGradient () [inline, virtual]

Reimplemented in IDM.

Definition at line 21 of file DriverModel.h.

Referenced by Gradient::removeVehicle().

```
21 {};
```

4.37.3.6 virtual void DriverModel::SetGradient (double *gradient*) [inline, virtual]

Reimplemented in IDM.

Definition at line 22 of file DriverModel.h.

Referenced by Gradient::addVehicle().

```
22 {};
```

4.37.3.7 virtual void DriverModel::ClearSpeedLimit () [inline, virtual]

Reimplemented in IDM.

Definition at line 23 of file DriverModel.h.

Referenced by SpeedLimit::removeVehicle().

```
23 {};
```

4.37.3.8 virtual void DriverModel::SetSpeedLimit (double *limit*) [inline, virtual]

Reimplemented in IDM.

Definition at line 24 of file DriverModel.h.

Referenced by SpeedLimit::addVehicle().

```
24 {};
```

4.37.3.9 virtual double DriverModel::LaneChange (double *GapToFront*, double *GapToBack*, double *FrontChangeAccel*, double *CurrentBackAccel*, double *ProposedBackAccel*, bool *overtake*) [pure virtual]

Implemented in IDM.

Referenced by Vehicle::LaneChangeAdvantage().

4.37.4 Member Data Documentation

4.37.4.1 bool DriverModel::m_bInGradient [protected]

Definition at line 36 of file DriverModel.h.

Referenced by IDM::ClearGradient(), DriverModel(), and IDM::SetGradient().

4.37.4.2 bool DriverModel::m_bInSpeedLimit [protected]

Definition at line 37 of file DriverModel.h.

Referenced by IDM::ClearSpeedLimit(), DriverModel(), IDM::SetGradient(), and IDM::SetSpeedLimit().

The documentation for this class was generated from the following files:

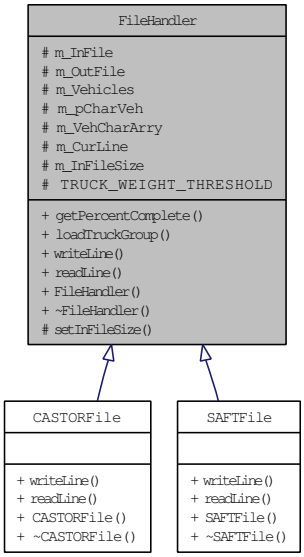
- D:/~Research/Code/C++/EvolveTraffic/DriverModel.h
- D:/~Research/Code/C++/EvolveTraffic/DriverModel.cpp

4.38 FileHandler Class Reference

A base class for handling file input and output.

```
#include <FileHandler.h>
```

Inheritance diagram for FileHandler:



Public Member Functions

- int getPercentComplete ()
- std::vector< Vehicle * > loadTruckGroup (int no)

  Facilitates the buffering of a number of vehicles, rather than line by line input.

- virtual void writeLine (Vehicle *pVeh)=0
- virtual Vehicle * readLine ()=0
- FileHandler ()

  Default Constructor.

- virtual ~FileHandler ()

  Default Destructor.

Protected Member Functions

- void setInFileSize ()

Protected Attributes

- std::ifstream m_InFile
- std::ofstream m_OutFile
- std::vector< Vehicle * > m_Vehicles
- char * m_pCharVeh

- char m_VehCharArry [78]
- int m_CurLine
- int m_InFileSize
- int TRUCK_WEIGHT_THRESHOLD

4.38.1 Detailed Description

A base class for handling file input and output.

Definition at line 21 of file FileHandler.h.

4.38.2 Constructor & Destructor Documentation

4.38.2.1 FileHandler::FileHandler ()

Default Constructor.

Definition at line 23 of file FileHandler.cpp.

References m_CurLine, CConfigData::VehicleID_Config::TRUCK_WEIGHT_-THRESHOLD, TRUCK_WEIGHT_THRESHOLD, and CConfigData::VehicleID.

```
24 {
25         TRUCK_WEIGHT_THRESHOLD = g_ConfigData.VehicleID.TRUCK_WEIGHT_THRESHOLD;
26
27         m_CurLine = 0;
28 }
```

4.38.2.2 FileHandler::~FileHandler () [virtual]

Default Destructor.

Definition at line 31 of file FileHandler.cpp.

```
32 {
33
34 }
```

4.38.3 Member Function Documentation

4.38.3.1 int FileHandler::getPercentComplete ()

Definition at line 61 of file FileHandler.cpp.

References m_CurLine, and m_InFileSize.

Referenced by Road::populate().

```
62 {
63         double prop = (double)(m_CurLine) / m_InFileSize;
64
65         int percent = int(prop * 100 + 0.5);
```

```
66
67          return percent;
68 }
```

4.38.3.2 std::vector< Vehicle * > FileHandler::loadTruckGroup (int *no*)

Facilitates the buffering of a number of vehicles, rather than line by line input.

Parameters:

no The number of vehicles to buffer

Returns:

The buffer of vehicles

Definition at line 41 of file FileHandler.cpp.

References m_Vehicles, and readLine().

Referenced by Road::init().

```
42 {
43          for(int i = 0; i < no; i++)
44          {
45                  Vehicle* pVeh = readLine();
46                  m_Vehicles.push_back(pVeh);
47          }
48
49          return m_Vehicles;
50 }
```

Here is the call graph for this function:

4.38.3.3 virtual void FileHandler::writeLine (Vehicle * *pVeh*) [pure virtual]

Implemented in CASTORFile, and SAFTFile.

Referenced by OutputDetector::WriteVehiclesToFile().

4.38.3.4 virtual Vehicle* FileHandler::readLine () [pure virtual]

Implemented in CASTORFile, and SAFTFile.

Referenced by loadTruckGroup(), and Road::populate().

4.38.3.5 void FileHandler::setInFileSize () [protected]

Definition at line 52 of file FileHandler.cpp.

References m_InFile, and m_InFileSize.

Referenced by CASTORFile::CASTORFile(), and SAFTFile::SAFTFile().

```
53 {
54     m_InFileSize = std::count(  std::istreambuf_iterator<char>(m_InFile),
55                                                                 std::istreambuf_iterator<char>()
56                                                                 '\n');
57
58         m_InFile.seekg(0,std::ios::beg);        // go back to start of file
59 }
```

4.38.4 Member Data Documentation

4.38.4.1 std::ifstream FileHandler::m_InFile [protected]

Definition at line 33 of file FileHandler.h.

Referenced by CASTORFile::CASTORFile(), SAFTFile::readLine(), CASTORFile::readLine(), SAFTFile::SAFTFile(), setInFileSize(), CASTORFile::~CASTORFile(), and SAFTFile::~SAFTFile().

4.38.4.2 std::ofstream FileHandler::m_OutFile [protected]

Definition at line 34 of file FileHandler.h.

Referenced by CASTORFile::CASTORFile(), SAFTFile::SAFTFile(), SAFTFile::writeLine(), CASTORFile::writeLine(), CASTORFile::~CASTORFile(), and SAFTFile::~SAFTFile().

4.38.4.3 std::vector<Vehicle*> FileHandler::m_Vehicles [protected]

Definition at line 35 of file FileHandler.h.

Referenced by loadTruckGroup(), CASTORFile::~CASTORFile(), and SAFTFile::~SAFTFile().

4.38.4.4 char* FileHandler::m_pCharVeh [protected]

Definition at line 36 of file FileHandler.h.

Referenced by CASTORFile::CASTORFile(), SAFTFile::SAFTFile(), SAFTFile::writeLine(), and CASTORFile::writeLine().

4.38.4.5 char FileHandler::m_VehCharArry[78] [protected]

Definition at line 37 of file FileHandler.h.

Referenced by CASTORFile::CASTORFile(), and SAFTFile::SAFTFile().

4.38.4.6 int FileHandler::m_CurLine [protected]

Definition at line 39 of file FileHandler.h.

Referenced by FileHandler(), getPercentComplete(), SAFTFile::readLine(), and CASTORFile::readLine().

4.38.4.7 int FileHandler::m_InFileSize [protected]

Definition at line 40 of file FileHandler.h.

Referenced by getPercentComplete(), and setInFileSize().

4.38.4.8 int FileHandler::TRUCK_WEIGHT_THRESHOLD [protected]

Definition at line 42 of file FileHandler.h.

Referenced by FileHandler(), SAFTFile::readLine(), and CASTORFile::readLine().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/FileHandler.h
- D:/~Research/Code/C++/EvolveTraffic/FileHandler.cpp

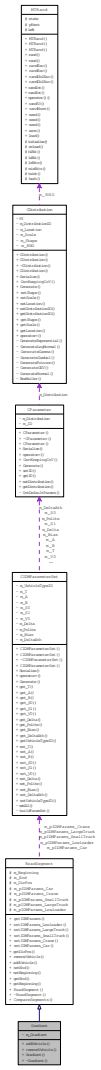
4.39 Gradient Class Reference

A derived class to represent a gradient on a road.

`#include <Gradient.h>`

Inheritance diagram for Gradient:

Collaboration diagram for Gradient:



Public Member Functions

- void addVehicle (Vehicle ∗pVeh)
- void removeVehicle (Vehicle ∗pVeh)
- Gradient (int start, int end, double grad, bool DirPos)
- virtual ∼Gradient ()

Private Attributes

- double m_Gradient

4.39.1 Detailed Description

A derived class to represent a gradient on a road.

Definition at line 17 of file Gradient.h.

4.39.2 Constructor & Destructor Documentation

4.39.2.1 Gradient::Gradient (int *start*, int *end*, double *grad*, bool *DirPos*)

Definition at line 19 of file Gradient.cpp.

References RoadSegment::m_Beginning, RoadSegment::m_DirPos, RoadSegment::m_End, and m_Gradient.

```
20 {
21         m_Beginning = start;
22         m_End = end;
23         m_DirPos = DirPos;
24         m_Gradient = grad;
25 }
```

4.39.2.2 Gradient::~Gradient () [virtual]

Definition at line 27 of file Gradient.cpp.

```
28 {
29
30 }
```

4.39.3 Member Function Documentation

4.39.3.1 void Gradient::addVehicle (Vehicle * *pVeh*) [virtual]

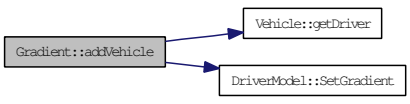
Reimplemented from RoadSegment.

Definition at line 32 of file Gradient.cpp.

References Vehicle::getDriver(), m_Gradient, and DriverModel::SetGradient().

```
33 {
34         pVeh->getDriver()->SetGradient(m_Gradient);
35 }
```

Here is the call graph for this function:



4.39.3.2 void Gradient::removeVehicle (Vehicle * *pVeh*) [virtual]

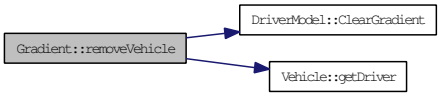
Reimplemented from RoadSegment.

Definition at line 37 of file Gradient.cpp.

References DriverModel::ClearGradient(), and Vehicle::getDriver().

```
38 {
39 //      TRACE("Remove vehicle - Direction: %d Position: %f\tVelocity: %f\n",pVeh->getDirection()
40         DriverModel* driver = pVeh->getDriver();                // called before add vehicle!!
41         driver->ClearGradient();                                          // causes vehicle st
42 }
```

Here is the call graph for this function:



4.39.4 Member Data Documentation

4.39.4.1 double Gradient::m_Gradient [private]

Definition at line 26 of file Gradient.h.

Referenced by addVehicle(), and Gradient().

The documentation for this class was generated from the following files:

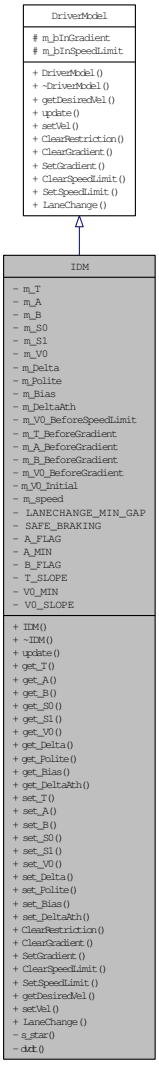
- D:/~Research/Code/C++/EvolveTraffic/Gradient.h
- D:/~Research/Code/C++/EvolveTraffic/Gradient.cpp

4.40 IDM Class Reference

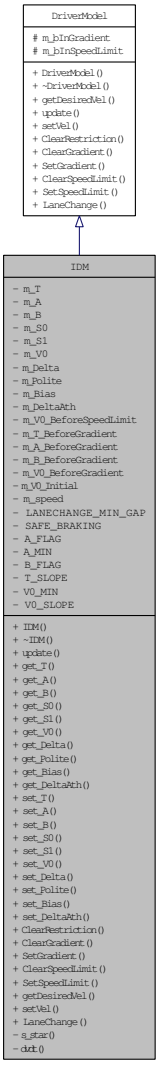
A class representing the IDM driver model.

`#include <IDM.h>`

Inheritance diagram for IDM:



Collaboration diagram for IDM:



Public Member Functions

- IDM ()
- virtual ~IDM ()
- double update (double vel, double dist)

  Updates the IDM's acceleration.

- double get_T ()
- double get_A ()
- double get_B ()
- double get_S0 ()

- double get_S1 ()
- double get_V0 ()
- double get_Delta ()
- double get_Polite ()
- double get_Bias ()
- double get_DeltaAth ()
- void set_T (double val)
- void set_A (double val)
- void set_B (double val)
- void set_S0 (double val)
- void set_S1 (double val)
- void set_V0 (double val)
- void set_Delta (double val)
- void set_Polite (double val)
- void set_Bias (double val)
- void set_DeltaAth (double val)
- virtual void ClearRestriction ()

  Clears all restrictions on the IDM.

- virtual void ClearGradient ()

  Clears all gradient restrictions on the IDM.

- virtual void SetGradient (double gradient)
- virtual void ClearSpeedLimit ()

  Clears all speedlimit restrictions on the IDM.

- virtual void SetSpeedLimit (double newV0)

  Sets the properties for an IDM entering a speedlimit.

- virtual double getDesiredVel ()

  Gets the desired velocity of the IDM.

- virtual void setVel (double vel)

  Sets the velocity of the IDM.

- double LaneChange (double GapToFront, double GapToBack, double FrontChangeAccel, double CurrentBackAccel, double ProposedBackAccel, bool overtake)

  Finds the advantage that a vehicle would gain from changing lane.

Private Member Functions

- double s_star (double vel)
- double dvdt (double m_speed, double s)

Private Attributes

- double m_T

  Safe time headway.

- double m_A

  Maximum acceleration.

- double m_B

  Desired deceleration.

- double m_S0
- double m_S1
- double m_V0
- double m_Delta
- double m_Polite

  MOBIL: Consideration towards other drivers.

- double m_Bias

  MOBIL: Desire to keep to slow lane.

- double m_DeltaAth

  MOBIL: Dampens overtaking occurances.

- double m_V0_BeforeSpeedLimit
- double m_T_BeforeGradient
- double m_A_BeforeGradient
- double m_B_BeforeGradient
- double m_V0_BeforeGradient
- double m_V0_Initial
- double m_speed
- double LANECHANGE_MIN_GAP
- double SAFE_BRAKING
- bool A_FLAG
- double A_MIN
- bool B_FLAG
- double T_SLOPE
- double V0_MIN
- double V0_SLOPE

4.40.1 Detailed Description

A class representing the IDM driver model.

Definition at line 9 of file IDM.h.

4.40.2 Constructor & Destructor Documentation

4.40.2.1 IDM::IDM ()

Definition at line 8 of file IDM.cpp.

References CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config::IDM_Param_Modifiers::A_FLAG, A_FLAG, CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config::IDM_Param_Modifiers::A_MIN, A_MIN, CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config::IDM_Param_Modifiers::B_FLAG, B_FLAG, CConfigData::Road_Config::RoadFeatures_Config::Gradient, CConfigData::IDM, CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config::IDM_Modifiers, CConfigData::IDM_Config::LANECHANGE_MIN_GAP, LANECHANGE_MIN_GAP, m_A, m_A_BeforeGradient, m_B, m_B_BeforeGradient, m_Bias, m_Delta, m_DeltaAth, m_Polite, m_S0, m_S1, m_T, m_T_BeforeGradient, m_V0, m_V0_BeforeGradient, m_V0_BeforeSpeedLimit, m_V0_Initial, CConfigData::Road, CConfigData::Road_Config::RoadFeatures, CConfigData::IDM_Config::SAFE_BRAKING, SAFE_BRAKING, CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config::IDM_Param_Modifiers::T_SLOPE, T_SLOPE, CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config::IDM_Param_Modifiers::V0_MIN, V0_MIN, CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config::IDM_Param_Modifiers::V0_SLOPE, and V0_SLOPE.

```
9 {
10          LANECHANGE_MIN_GAP      = g_ConfigData.IDM.LANECHANGE_MIN_GAP;
11          SAFE_BRAKING            = g_ConfigData.IDM.SAFE_BRAKING;
12
13          A_FLAG          = g_ConfigData.Road.RoadFeatures.Gradient.IDM_Modifiers.A_FLAG;
14          A_MIN           = g_ConfigData.Road.RoadFeatures.Gradient.IDM_Modifiers.A_MIN;
15          B_FLAG          = g_ConfigData.Road.RoadFeatures.Gradient.IDM_Modifiers.B_FLAG;
16          T_SLOPE         = g_ConfigData.Road.RoadFeatures.Gradient.IDM_Modifiers.T_SLOPE;
17          V0_MIN          = g_ConfigData.Road.RoadFeatures.Gradient.IDM_Modifiers.V0_MIN;
18          V0_SLOPE        = g_ConfigData.Road.RoadFeatures.Gradient.IDM_Modifiers.V0_SLOPE;
19
20          m_T                     = 0.0;
21          m_A                     = 0.0;
22          m_B                     = 0.0;
23          m_S0            = 0.0;
24          m_S1            = 0.0;
25          m_V0            = 0.0;
26          m_Delta         = 0.0;
27          m_Polite        = 0.0;
28          m_Bias          = 0.0;
29          m_DeltaAth      = 0.0;
30
31          m_V0_BeforeSpeedLimit   = 0.0;
32          m_V0_BeforeGradient             = 0.0;
33          m_T_BeforeGradient              = 0.0;
34          m_A_BeforeGradient              = 0.0;
35          m_B_BeforeGradient              = 0.0;
36
37          m_V0_Initial                    = 0.0;
38 }
```

4.40.2.2 IDM::~IDM () [virtual]

Definition at line 40 of file IDM.cpp.

```
41 {
42
43 }
```

4.40.3 Member Function Documentation

4.40.3.1 double IDM::update (double *vel*, double *dist*) [virtual]

Updates the IDM's acceleration.

Parameters:

vel The vehicle's velocity

dist The distance to the next vehicle

Returns:

The new acceleration

This function updates the acceleration of the IDM based on the current velocity and the distance to the next vehicle

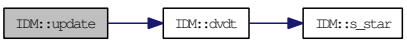
Implements DriverModel.

Definition at line 56 of file IDM.cpp.

References dvdt().

```
57 {
58         double accel = dvdt(vel, dist);
59         return accel;
60 }
```

Here is the call graph for this function:



4.40.3.2 double IDM::get_T ()

Definition at line 169 of file IDM.cpp.

References m_T.

Referenced by Vehicle::getDataString().

```
170 {
171         // Unit - input: secs
172         //              - output: secs
173         return m_T;
174 }
```

4.40.3.3 double IDM::get_A ()

Definition at line 186 of file IDM.cpp.

References m_A.

Referenced by Vehicle::getDataString().

```
187 {
188         // Unit - input: m/s^2
189         //              - output: m/s^2
190         return m_A;
191 }
```

4.40.3.4 double IDM::get_B ()

Definition at line 203 of file IDM.cpp.

References m_B.

Referenced by Vehicle::getDataString().

```
204 {
205         // Unit - input: m/s^2
206         //              - output: m/s^2
207         return m_B;
208 }
```

4.40.3.5 double IDM::get_S0 ()

Definition at line 219 of file IDM.cpp.

References m_S0.

Referenced by Vehicle::getDataString().

```
220 {
221         // Unit - input: m
222         //              - output: m
223         return m_S0;
224 }
```

4.40.3.6 double IDM::get_S1 ()

Definition at line 235 of file IDM.cpp.

References m_S1.

Referenced by Vehicle::getDataString().

```
236 {
237         // Unit - input: m
238         //              - output: m
239         return m_S1;
240 }
```

4.40.3.7 double IDM::get_V0 ()

Definition at line 253 of file IDM.cpp.

References m_V0.

Referenced by Vehicle::getDataString().

```
254 {
255         // Unit - input: m/s
256         //              - output: m/s //km/h
257         return m_V0;// * M_PER_S_TO_KM_PER_H;
258 }
```

4.40.3.8 double IDM::get_Delta ()

Definition at line 269 of file IDM.cpp.

References m_Delta.

Referenced by Vehicle::getDataString().

```
270 {
271         // Unit - input: none
272         //              - output: none
273         return m_Delta;
274 }
```

4.40.3.9 double IDM::get_Polite ()

Definition at line 285 of file IDM.cpp.

References m_Polite.

Referenced by Vehicle::getDataString().

```
286 {
287         // Unit - input: none
288         //              - output: none
289         return m_Polite;
290 }
```

4.40.3.10 double IDM::get_Bias ()

Definition at line 301 of file IDM.cpp.

References m_Bias.

Referenced by Vehicle::getDataString().

```
302 {
303         // Unit - input: none
304         //              - output: none
305         return m_Bias;
306 }
```

4.40.3.11 double IDM::get_DeltaAth ()

Definition at line 317 of file IDM.cpp.

References m_DeltaAth.

Referenced by Vehicle::getDataString().

```
318 {
319         // Unit - input: none
320         //              - output: none
321         return m_DeltaAth;
322 }
```

4.40.3.12 void IDM::set_T (double *val*)

Definition at line 162 of file IDM.cpp.

References m_T.

Referenced by CIDMParameterSet::Generate().

```
163 {
164         // Unit - input: secs
165         //              - output: secs
166         m_T = val;
167 }
```

4.40.3.13 void IDM::set_A (double *val*)

Definition at line 178 of file IDM.cpp.

References m_A.

Referenced by CIDMParameterSet::Generate().

```
179 {
180         // Unit - input: m/s^2
181         //              - output: m/s^2
182         m_A = val;
183         ASSERT(m_A > 0);
184 }
```

4.40.3.14 void IDM::set_B (double *val*)

Definition at line 195 of file IDM.cpp.

References m_B.

Referenced by CIDMParameterSet::Generate().

```
196 {
197         // Unit - input: m/s^2
198         //              - output: m/s^2
199         m_B = val;
200         ASSERT(m_B > 0);
201 }
```

4.40.3.15 void IDM::set_S0 (double *val*)

Definition at line 212 of file IDM.cpp.

References m_S0.

Referenced by CIDMParameterSet::Generate().

```
213 {
214         // Unit - input: m
215         //              - output: m
216         m_S0 = val;
217 }
```

4.40.3.16 void IDM::set_S1 (double *val*)

Definition at line 228 of file IDM.cpp.

References m_S1.

Referenced by CIDMParameterSet::Generate().

```
229 {
230         // Unit - input: m
231         //              - output: m
232         m_S1 = val;
233 }
```

4.40.3.17 void IDM::set_V0 (double *val*)

Definition at line 244 of file IDM.cpp.

References KM_PER_H_TO_M_PER_S, m_V0, and m_V0_Initial.

Referenced by CIDMParameterSet::Generate().

```
245 {
246         // Unit - input: km/h
247         //              - output: m/s
248         m_V0 = val * KM_PER_H_TO_M_PER_S;
249         m_V0_Initial = m_V0;
250         ASSERT(m_V0 > 0);
251 }
```

4.40.3.18 void IDM::set_Delta (double *val*)

Definition at line 262 of file IDM.cpp.

References m_Delta.

Referenced by CIDMParameterSet::Generate().

```
263 {
264         // Unit - input: none
265         //              - output: none
266         m_Delta = val;
267 }
```

4.40.3.19 void IDM::set_Polite (double *val*)

Definition at line 278 of file IDM.cpp.

References m_Polite.

Referenced by CIDMParameterSet::Generate().

```
279 {
280         // Unit - input: none
281         //              - output: none
282         m_Polite = val;
283 }
```

4.40.3.20 void IDM::set_Bias (double *val*)

Definition at line 294 of file IDM.cpp.

References m_Bias.

Referenced by CIDMParameterSet::Generate().

```
295 {
296         // Unit - input: none
297         //              - output: none
298         m_Bias = val;
299 }
```

4.40.3.21 void IDM::set_DeltaAth (double *val*)

Definition at line 310 of file IDM.cpp.

References m_DeltaAth.

Referenced by CIDMParameterSet::Generate().

```
311 {
312         // Unit - input: none
313         //              - output: none
314         m_DeltaAth = val;
315 }
```

4.40.3.22 void IDM::ClearRestriction () [virtual]

Clears all restrictions on the IDM.

This function removes all of the speed/gradient restrictions on an IDM, and is only called once the IDM's vehicle has left all previous road segments and has not entered another one. The desired velocity of the IDM is set back to its original desired velocity

Reimplemented from DriverModel.

Definition at line 439 of file IDM.cpp.

References m_V0, and m_V0_Initial.

```
440 {
441         // This is only called if we are entering a clear road and
442         // have already been removed from all segments.
443         m_V0 = m_V0_Initial;
444 }
```

4.40.3.23 void IDM::ClearGradient () [virtual]

Clears all gradient restrictions on the IDM.

This function removes all of the restrictions imposed upon an IDM when the IDM's vehicle enters a gradient road segment. The variables affected are all set back to the value they were previous to the vehicle entering the gradient segment.

Reimplemented from DriverModel.

Definition at line 417 of file IDM.cpp.

References m_A, m_A_BeforeGradient, m_B, m_B_BeforeGradient, DriverModel::m_bInGradient, m_T, m_T_BeforeGradient, m_V0, and m_V0_-BeforeGradient.

```
418 {
419         if(!m_bInGradient)
420                 TRACE("***Grad: Removed but not in\n");
421         ASSERT(m_V0_BeforeGradient > 0);        // if this happens it means vehicle wasn't adde
422
423         m_bInGradient = false;
424         m_T = m_T_BeforeGradient;
425         m_A = m_A_BeforeGradient;
426         m_B = m_B_BeforeGradient;
427         m_V0 = m_V0_BeforeGradient;
428 }
```

4.40.3.24 void IDM::SetGradient (double *gradient*) [virtual]

Reimplemented from DriverModel.

Definition at line 348 of file IDM.cpp.

References A_FLAG, A_MIN, B_FLAG, KM_PER_H_TO_M_PER_S, m_A, m_A_BeforeGradient, m_B, m_B_BeforeGradient, DriverModel::m_bInGradient, DriverModel::m_bInSpeedLimit, m_T, m_T_BeforeGradient, m_V0, m_V0_-BeforeGradient, m_V0_Initial, T_SLOPE, V0_MIN, and V0_SLOPE.

```
349 {
350         m_bInGradient = true;
351         m_A_BeforeGradient = m_A;       m_T_BeforeGradient = m_T;
352         m_B_BeforeGradient = m_B;       m_V0_BeforeGradient = m_V0;
353
354         // increase T - regardless of uphill or downhill
355         double ABSgradient = gradient < 0 ? -gradient : gradient;
356         m_T += ABSgradient/T_SLOPE;
357
358         // Reduce A - for uphill
359         // MAGIC NUMBER 10 accounts for component of gravity acceleration
```

```
360         // parallel to road surface - sin of small angle
361         if(A_FLAG)
362         {
363                 m_A -= gradient/10;
364                 m_A = m_A < A_MIN ? A_MIN : m_A;        // set to min at least
365         }
366
367         // Increase B for uphill only - MAGIC NUMBER 10 as for A
368         if(B_FLAG && gradient > 0)
369                 m_B += gradient/10;
370
371         // Reduce V0 for uphill only
372         if(gradient > 0)
373         {
374                 // reduce V0 from the 'free' desired velocity
375                 // so if not in speed limit zone, we're free to reduce to gradient, else
376                 if(!m_bInSpeedLimit)
377                         m_V0 -= (gradient/V0_SLOPE)*KM_PER_H_TO_M_PER_S;
378                 else    // we're in a speed limit so see if speed limit or gradient governs V0
379                 {
380                         double FreeV0 = m_V0_Initial;   // Do NOT change value of m_V0_BeforeSp
381                         double SpeedLimitV0 = m_V0;
382                         double GradientV0 = FreeV0 - (gradient/V0_SLOPE)*KM_PER_H_TO_M_PER_S;
383                         m_V0 = SpeedLimitV0 < GradientV0 ? SpeedLimitV0 : GradientV0;
384                 }
385                 if(m_V0 < V0_MIN)
386                 {
387                         //TRACE("*** V0 of: %f\t set to: %f\n",m_V0, V0_MIN);
388                         m_V0 = V0_MIN;  // set to min at least
389                 }
390         }       // end if gradient uphill
391 }
```

4.40.3.25 void IDM::ClearSpeedLimit () [virtual]

Clears all speedlimit restrictions on the IDM.

This function removes the speed restriction imposed upon an IDM when the IDM's vehicle enters a speedlimit road segment. The desired velocity of the vehicle is set back to the value it was previous to the vehicle entering the speedlimit segment

Reimplemented from DriverModel.

Definition at line 400 of file IDM.cpp.

References DriverModel::m_bInSpeedLimit, m_V0, and m_V0_BeforeSpeedLimit.

```
401 {
402         if(!m_bInSpeedLimit)
403                 TRACE("***SL: Removed but not in\n");
404         ASSERT(m_V0_BeforeSpeedLimit > 0);      // if this happens it means vehicle wasn't adde
405         m_bInSpeedLimit = false;
406         m_V0 = m_V0_BeforeSpeedLimit;
407 }
```

4.40.3.26 void IDM::SetSpeedLimit (double *newV0*) [virtual]

Sets the properties for an IDM entering a speedlimit.

Parameters:

newV0 The new desired velocity of the IDM

This function deals with setting the properties for when an IDM enters a speedlimit segment. It takes note that it is currently in a speedlimit, and takes note of the desired velocity before the segment was entered. The desired velocity of the IDM is set to the given desired velocity

Reimplemented from DriverModel.

Definition at line 335 of file IDM.cpp.

References DriverModel::m_bInSpeedLimit, m_V0, and m_V0_BeforeSpeedLimit.

```
336 {
337         m_bInSpeedLimit = true;
338         m_V0_BeforeSpeedLimit = m_V0;
339
340         m_V0 = newV0;
341         ASSERT(m_V0 > 0);
342         ASSERT(m_V0_BeforeSpeedLimit > 0);
343 }
```

4.40.3.27 double IDM::getDesiredVel () [virtual]

Gets the desired velocity of the IDM.

Returns:

The IDM's desired velocity

Implements DriverModel.

Definition at line 105 of file IDM.cpp.

References m_V0.

```
106 {
107         return m_V0;
108 }
```

4.40.3.28 void IDM::setVel (double *vel*) [virtual]

Sets the velocity of the IDM.

Parameters:

vel The velocity to set

Implements DriverModel.

Definition at line 96 of file IDM.cpp.

References m_speed.

```
97 {
98         m_speed = vel;
99 }
```

4.40.3.29 double IDM::LaneChange (double *GapToFront*, double *GapToBack*, double *FrontChangeAccel*, double *CurrentBackAccel*, double *ProposedBackAccel*, bool *overtake*) [virtual]

Finds the advantage that a vehicle would gain from changing lane.

Parameters:

GapToFront The gap to the vehicle in front in the new lane

GapToBack The gap to the vehicle behind in the new lane

FrontChangeAccel The change in acceleration the vehicle will experience with a new front vehicle

CurrentBackAccel The current acceleration of the vehicle behind in the new lane

ProposedBackAccel The new acceleration the vehicle behind will experience with a new front vehicle

overtake Whether or not the vehicle is overtaking

Returns:

The net advantage of the change

This function finds the net advantage (the vehicle's own advantage minus the disadvantage of others) that occurs due to a vehicle changing lane. The advantage is seen as the extra acceleration gained from changing lane, while the disadvantage is considered to be the deceleration that other vehicles have to undertake in order to allow the change to take place.

Implements DriverModel.

Definition at line 129 of file IDM.cpp.

References LANECHANGE_MIN_GAP, m_Bias, m_DeltaAth, m_Polite, and SAFE_BRAKING.

```
131 {
132         double advantage = 0.0;
133         double othersDisadv = 0.0;
134         double netAdvantage = 0.0;
135 
136         if(GapToFront >= LANECHANGE_MIN_GAP && GapToBack >= LANECHANGE_MIN_GAP && ProposedBackA
137         {
138                 advantage = FrontChangeAccel + (overtake ? -1 : 1) * m_Bias;
139                 othersDisadv = CurrentBackAccel - ProposedBackAccel;
140         }
141 
142         if(othersDisadv < 0.0)
143                 othersDisadv = 0.0;
144 
145         netAdvantage = advantage - othersDisadv * m_Polite;
```

```
146
147         if(netAdvantage <= m_DeltaAth)  // if it's under the threshold
148                 netAdvantage = 0.0;
149
150         return netAdvantage;
151 }
```

4.40.3.30 double IDM::s_star (double *vel*) [private]

Definition at line 78 of file IDM.cpp.

References m_A, m_B, m_S0, m_S1, m_speed, m_T, and m_V0.

Referenced by dvdt().

```
79 {
80         // Desired gap
81         ASSERT(m_V0 > 0);       // it has to be because of the division & sqrt below
82         ASSERT(m_A > 0);
83         ASSERT(m_B > 0);
84         double s_str = m_S0 + m_S1 * sqrt(m_speed/m_V0)
85                 + m_T*m_speed + (m_speed*dv)/(2*sqrt(m_A*m_B));
86
87         return s_str;
88 }
```

4.40.3.31 double IDM::dvdt (double *m_speed*, double *s*) [private]

Definition at line 62 of file IDM.cpp.

References m_A, m_Delta, m_speed, m_V0, s_star(), and SAFE_BRAKING.

Referenced by update().

```
63 {
64         double s_str = s_star(dv);
65         double intel_acc = m_A*(1-pow(m_speed/m_V0,m_Delta));
66         double intel_dec = -m_A*pow(s_str/s,2);
67
68         double net_acc = intel_acc + intel_dec;
69
70         // THIS NEEDS TO BE CAPPED OR BREAKDOWN OCCURS (net_acc ~ -30000)
71         // Test value of -100 given
72         if(net_acc < SAFE_BRAKING)
73                 net_acc = SAFE_BRAKING;
74
75         return net_acc;
76 }
```

Here is the call graph for this function:

4.40.4 Member Data Documentation

4.40.4.1 double IDM::m_T [private]

Safe time headway.

Definition at line 44 of file IDM.h.

Referenced by ClearGradient(), get_T(), IDM(), s_star(), set_T(), and SetGradient().

4.40.4.2 double IDM::m_A [private]

Maximum acceleration.

Definition at line 46 of file IDM.h.

Referenced by ClearGradient(), dvdt(), get_A(), IDM(), s_star(), set_A(), and SetGradient().

4.40.4.3 double IDM::m_B [private]

Desired deceleration.

Definition at line 48 of file IDM.h.

Referenced by ClearGradient(), get_B(), IDM(), s_star(), set_B(), and SetGradient().

4.40.4.4 double IDM::m_S0 [private]

Definition at line 50 of file IDM.h.

Referenced by get_S0(), IDM(), s_star(), and set_S0().

4.40.4.5 double IDM::m_S1 [private]

Definition at line 52 of file IDM.h.

Referenced by get_S1(), IDM(), s_star(), and set_S1().

4.40.4.6 double IDM::m_V0 [private]

Definition at line 54 of file IDM.h.

Referenced by ClearGradient(), ClearRestriction(), ClearSpeedLimit(), dvdt(), get_V0(), getDesiredVel(), IDM(), s_star(), set_V0(), SetGradient(), and SetSpeedLimit().

4.40.4.7 double IDM::m_Delta [private]

Definition at line 56 of file IDM.h.

Referenced by dvdt(), get_Delta(), IDM(), and set_Delta().

4.40.4.8 double IDM::m_Polite [private]

MOBIL: Consideration towards other drivers.

Definition at line 59 of file IDM.h.

Referenced by get_Polite(), IDM(), LaneChange(), and set_Polite().

4.40.4.9 double IDM::m_Bias [private]

MOBIL: Desire to keep to slow lane.

Definition at line 61 of file IDM.h.

Referenced by get_Bias(), IDM(), LaneChange(), and set_Bias().

4.40.4.10 double IDM::m_DeltaAth [private]

MOBIL: Dampens overtaking occurances.

Definition at line 63 of file IDM.h.

Referenced by get_DeltaAth(), IDM(), LaneChange(), and set_DeltaAth().

4.40.4.11 double IDM::m_V0_BeforeSpeedLimit [private]

Definition at line 65 of file IDM.h.

Referenced by ClearSpeedLimit(), IDM(), and SetSpeedLimit().

4.40.4.12 double IDM::m_T_BeforeGradient [private]

Definition at line 66 of file IDM.h.

Referenced by ClearGradient(), IDM(), and SetGradient().

4.40.4.13 double IDM::m_A_BeforeGradient [private]

Definition at line 67 of file IDM.h.

Referenced by ClearGradient(), IDM(), and SetGradient().

4.40.4.14 double IDM::m_B_BeforeGradient [private]

Definition at line 68 of file IDM.h.

Referenced by ClearGradient(), IDM(), and SetGradient().

4.40.4.15 double IDM::m_V0_BeforeGradient [private]

Definition at line 69 of file IDM.h.

Referenced by ClearGradient(), IDM(), and SetGradient().

4.40.4.16 double IDM::m_V0_Initial [private]

Definition at line 70 of file IDM.h.

Referenced by ClearRestriction(), IDM(), set_V0(), and SetGradient().

4.40.4.17 double IDM::m_speed [private]

Definition at line 93 of file IDM.h.

Referenced by dvdt(), s_star(), and setVel().

4.40.4.18 double IDM::LANECHANGE_MIN_GAP [private]

Definition at line 97 of file IDM.h.

Referenced by IDM(), and LaneChange().

4.40.4.19 double IDM::SAFE_BRAKING [private]

Definition at line 98 of file IDM.h.

Referenced by dvdt(), IDM(), and LaneChange().

4.40.4.20 bool IDM::A_FLAG [private]

Definition at line 99 of file IDM.h.

Referenced by IDM(), and SetGradient().

4.40.4.21 double IDM::A_MIN [private]

Definition at line 100 of file IDM.h.

Referenced by IDM(), and SetGradient().

4.40.4.22 bool IDM::B_FLAG [private]

Definition at line 101 of file IDM.h.

Referenced by IDM(), and SetGradient().

4.40.4.23 double IDM::T_SLOPE [private]

Definition at line 102 of file IDM.h.

Referenced by IDM(), and SetGradient().

4.40.4.24 double IDM::V0_MIN [private]

Definition at line 103 of file IDM.h.

Referenced by IDM(), and SetGradient().

4.40.4.25 double IDM::V0_SLOPE [private]

Definition at line 104 of file IDM.h.

Referenced by IDM(), and SetGradient().

The documentation for this class was generated from the following files:

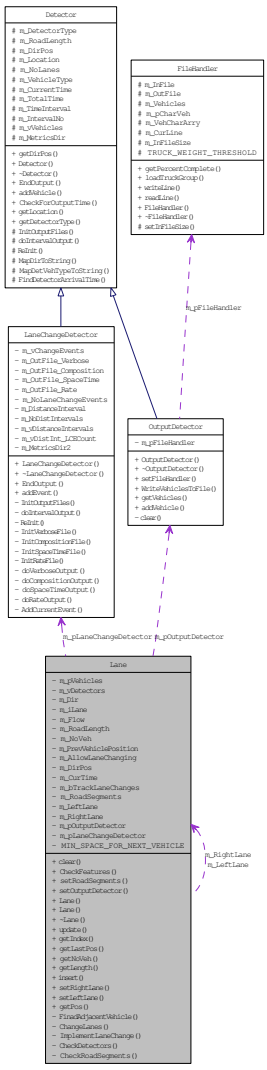
- D:/~Research/Code/C++/EvolveTraffic/IDM.h
- D:/~Research/Code/C++/EvolveTraffic/IDM.cpp

4.41 Lane Class Reference

A class representing a lane in a road.

`#include <Lane.h>`

Collaboration diagram for Lane:



Public Member Functions

- void clear ()

  Clears the lane so it can be used in another simulation.

- void CheckFeatures (const double curTime, double prevPosition, double curPosition, Vehicle *pVeh)

  Checks all features on the road.

- void setRoadSegments (std::vector< RoadSegment * > segments)

  Sets the road segments for the direction.

- void setOutputDetector (OutputDetector *out)

  Sets the output detector for the direction.

- Lane ()

  Default constructor.

- Lane (int iLane, int length, std::vector< Detector * > detectors, bool DirPos, bool AllowLaneChange)

  Main constructor.

- virtual ~Lane ()

  Default destructor.

- bool update (const double step, const double curTime)

  Updates the positions of all Vehicles in the Lane.

- int getIndex ()

  Gets the index of the lane.

- double getLastPos ()

  Gets the position of the last vehicle in the lane.

- int getNoVeh ()

  Gets the number of vehicles in the lane.

- double getLength ()

  Gets the length of the Lane.

- void insert (int i, Vehicle *pVeh)

  Inserts a Vehicle into the Lane.

- void setRightLane (Lane *right)

  Sets a pointer to the Lane to the right of the current Lane The Lane index is in global coords:
  - *For DriveOnRight the lanes are numbered from bottom to top 0.*

- void setLeftLane (Lane *left)

  Sets a pointer to the Lane to the left of the current Lane The Lane index is in global coords:
  - *For DriveOnRight the lanes are numbered from bottom to top 0.*

- std::vector< Vehicle * > getPos ()

  Gets all the Vehicles currently in the Lane.

Private Member Functions

- Vehicle * FinadAdjacentVehicle (Lane *AdjacentLane, double location, bool front, int *iAdjacentLane=NULL)

  Finds a vehicle in the adjacent lane.

- Lane * ChangeLanes (Vehicle *pVeh)

  Changes a Vehicle's Lane if advantageous and possible.

- Lane * ImplementLaneChange (Vehicle *pVeh, int LaneChangeID, int iVehLeftLane, int iVehRightLane)

  Changes a vehicle to another lane.

- void CheckDetectors (const double curTime, double prevPosition, double curPosition, Vehicle *pVeh)

  Adds a vehicle to the detectors list of vehicles for that step.

- void CheckRoadSegments (double prevPosition, double curPosition, Vehicle *pVeh)

  Checks whether a vehicle has entered or exited any road segments.

Private Attributes

- std::vector< Vehicle * > m_pVehicles
- std::vector< Detector * > m_vDetectors
- int m_Dir
- int m_iLane
- double m_Flow
- double m_RoadLength
- int m_NoVeh
- double m_PrevVehiclePosition
- bool m_AllowLaneChanging
- bool m_DirPos
- double m_CurTime
- bool m_bTrackLaneChanges
- std::vector< RoadSegment * > m_RoadSegments
- Lane * m_LeftLane
- Lane * m_RightLane
- OutputDetector * m_pOutputDetector
- LaneChangeDetector * m_pLaneChangeDetector
- double MIN_SPACE_FOR_NEXT_VEHICLE

4.41.1 Detailed Description

A class representing a lane in a road.

Definition at line 24 of file Lane.h.

4.41.2 Constructor & Destructor Documentation

4.41.2.1 Lane::Lane ()

Default constructor.

Definition at line 16 of file Lane.cpp.

References m_LeftLane, and m_RightLane.

```
17 {
18         m_LeftLane = NULL;
19         m_RightLane = NULL;
20 }
```

4.41.2.2 Lane::Lane (int *iLane*, int *RoadLength*, std::vector< Detector * > *detectors*, bool *DirPos*, bool *AllowLaneChange*)

Main constructor.

The main constructor used for creating a Lane object.

Parameters:

iLane The lane's index

RoadLength The length of the Road

detectors The detectors to be used in the lane

DirPos Whether the lane is in the positive direction or not

AllowLaneChange Whether lane changing is allowed

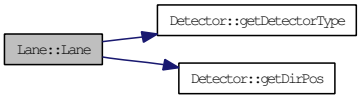
Definition at line 38 of file Lane.cpp.

References Detector::getDetectorType(), Detector::getDirPos(), CConfigData::IDM, m_AllowLaneChanging, m_bTrackLaneChanges, m_DirPos, m_iLane, m_LeftLane, m_NoVeh, m_pLaneChangeDetector, m_PrevVehiclePosition, m_RightLane, m_RoadLength, m_vDetectors, METRICS_TYPE_LANE_CHANGE, CConfigData::IDM_Config::MIN_SPACE_FOR_NEXT_VEHICLE, and MIN_SPACE_FOR_NEXT_VEHICLE.

```
39 {
40
41         MIN_SPACE_FOR_NEXT_VEHICLE = g_ConfigData.IDM.MIN_SPACE_FOR_NEXT_VEHICLE;
42         m_PrevVehiclePosition = MIN_SPACE_FOR_NEXT_VEHICLE;
43
44         m_NoVeh = 0;
45         m_RoadLength = RoadLength;
46         m_LeftLane = NULL;
47         m_RightLane = NULL;
48         m_bTrackLaneChanges = false;
49
50         m_iLane = iLane;
51         m_vDetectors = detectors;
52         m_DirPos = DirPos;
53         m_AllowLaneChanging = AllowLaneChange;
```

```
54
55          for(int i = 0; i < m_vDetectors.size(); i++)
56          {
57                  Detector* pDet = m_vDetectors.at(i);
58                  if( (pDet->getDetectorType() == METRICS_TYPE_LANE_CHANGE) && (pDet->getDirPos()
59                  {
60                          m_bTrackLaneChanges = true;
61                          m_pLaneChangeDetector = dynamic_cast<LaneChangeDetector*>(pDet);
62                          break;
63                  }
64          }
65 }
```

Here is the call graph for this function:



4.41.2.3 Lane::~Lane () [virtual]

Default destructor.

Definition at line 23 of file Lane.cpp.

```
24 {
25
26 }
```

4.41.3 Member Function Documentation

4.41.3.1 void Lane::clear ()

Clears the lane so it can be used in another simulation.

This function clears all of the lane's information so that it may be used in another simulation without re-instanciating.

Definition at line 529 of file Lane.cpp.

References m_pVehicles, m_RoadSegments, and m_vDetectors.

```
530 {
531
532          // If we exit in mid-sim, clear all vehicles currently on the road
533          for(int i = 0; i < m_pVehicles.size(); i++)
534                  delete m_pVehicles.at(i);
535
536          // Free the vector for vehicles
537          m_pVehicles.clear();
538
539          // Free the vectors for detectors and segments (both deleted in direction)
540          m_vDetectors.clear();
541          m_RoadSegments.clear();
542 }
```

4.41.3.2 void Lane::CheckFeatures (const double *curTime*, double *prevPosition*, double *curPosition*, Vehicle * *pVeh*)

Checks all features on the road.

Parameters:

curTime The current time

prevPosition The vehicle's position in the previous step

curPosition The vehicle's current position

pVeh The vehicle

See also:

CheckDetectors()
CheckRoadSegments()

This function checks all of the special features on the road

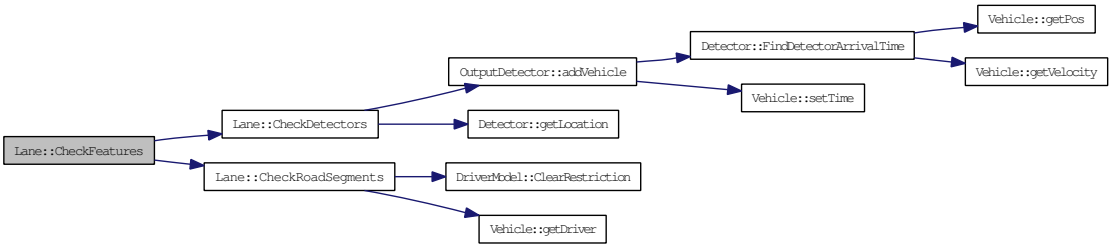
Definition at line 161 of file Lane.cpp.

References CheckDetectors(), and CheckRoadSegments().

Referenced by update().

```
162 {
163         CheckDetectors(curTime, prevPosition, curPosition, pVeh);
164         CheckRoadSegments(prevPosition, curPosition, pVeh);
165 }
```

Here is the call graph for this function:



4.41.3.3 void Lane::setRoadSegments (std::vector< RoadSegment * > *segments*)

Sets the road segments for the direction.

Parameters:

segments The road segments

Definition at line 464 of file Lane.cpp.

References m_RoadSegments.

Referenced by Direction::createLanes().

```
465 {
466         m_RoadSegments = segments;
467 }
```

4.41.3.4 void Lane::setOutputDetector (OutputDetector * *out*)

Sets the output detector for the direction.

Parameters:

out The output detector

Definition at line 455 of file Lane.cpp.

References m_pOutputDetector.

Referenced by Direction::createLanes().

```
456 {
457         m_pOutputDetector = out;
458 }
```

4.41.3.5 bool Lane::update (const double *step*, const double *curTime*)

Updates the positions of all Vehicles in the Lane.

This function is called for each iteration of the simulation, and handles the updating of the positions of Vehicles, and the removal of Vehicles which are no longer on the Road. If there are any Vehicles on the Road, the Vehicle at the head of the Lane is updated with respect to the clear Road ahead of it. Following this, all Vehicles following the Vehicle at the head of the Road are updated with respect to the preceding Vehicle. These Vehicles are also checked to see if it is time to determine if changing to another Lane would be advantageous. After all Vehicles have been updated, it is checked to see whether the Vehicle at the head of the Lane has yet passed the end of the Road. If it has, it is removed from the Lane and the simulation. Vehicles are also checked to see if they have passed a detector during the update

Parameters:

step The timestep

curTime The current simulation time

Returns:

Whether the lane is empty or not

See also:

ChangeLanes()
CheckFeatures()

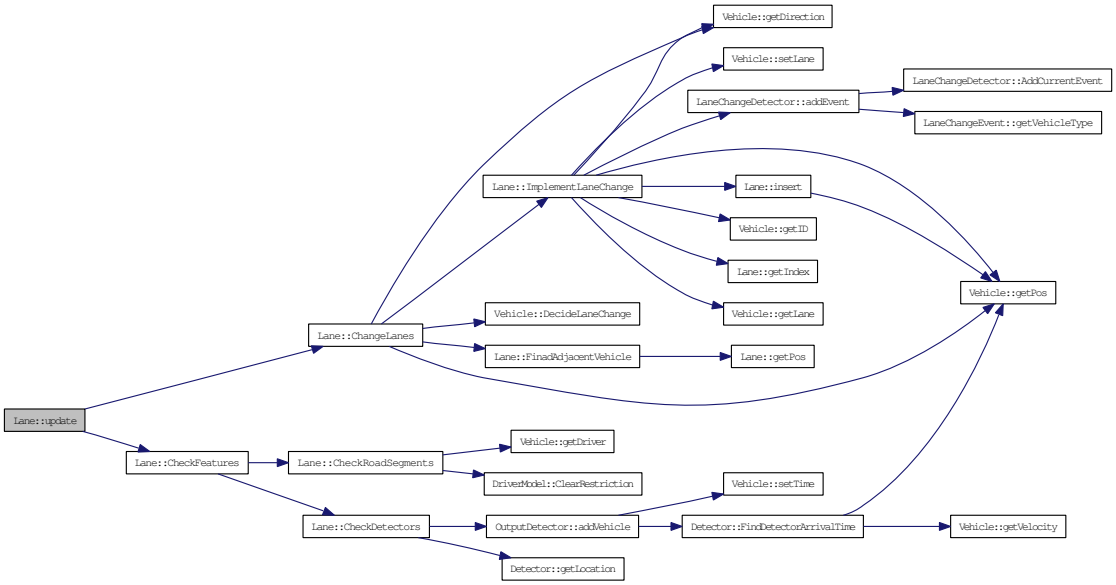
Definition at line 90 of file Lane.cpp.

References ChangeLanes(), CheckFeatures(), m_AllowLaneChanging, m_CurTime, m_NoVeh, m_PrevVehiclePosition, m_pVehicles, m_RoadLength, and MIN_SPACE_FOR_NEXT_VEHICLE.

```
91 {
92         m_CurTime = curTime;
93         bool laneEmpty = true;
94
95         if(m_pVehicles.size() > 0)
96         {
97                 // Update the head of the lane to drive towards the end of the road
98                 double previousPos = m_pVehicles.front()->getPos();
99                 m_pVehicles.front()->update(step);
100                 double currentPos = m_pVehicles.front()->getPos();
101                 CheckFeatures(curTime, previousPos, currentPos, m_pVehicles.front());
102
103                 // Update each vehicle based on the properties of the preceding vehicle
104                 for(int i = 1; i < m_pVehicles.size(); i++)
105                 {
106                         previousPos = m_pVehicles[i]->getPos();
107                         m_pVehicles[i]->update(step, m_pVehicles[i-1]);
108                         currentPos = m_pVehicles[i]->getPos();
109
110                         bool bChangedLane = false;
111                         if(m_AllowLaneChanging)
112                         {
113                                 bool bChangeStatus = m_pVehicles.at(i)->getChangeStatus();
114                                 if(bChangeStatus)
115                                 {
116                                         Lane* pNewLane = ChangeLanes(m_pVehicles.at(i));
117                                         bChangedLane = pNewLane == NULL ? false : true;
118                                         if(bChangedLane)
119                                         {
120                                                 // get the new lane to check the vehicle for a
121                                                 pNewLane->CheckFeatures(curTime, previousPos, c
122                                                 m_pVehicles.at(i)->setChangeStatus(false);
123                                                 m_pVehicles.erase(m_pVehicles.begin()+i);
124                                         }
125                                 }
126                         }
127                         if(!bChangedLane)
128                                 CheckFeatures(curTime, previousPos, currentPos, m_pVehicles.at
129
130                 }
131                 // update position of last vehicle in lane
132                 // so more vehicles can get on the road
133                 m_PrevVehiclePosition = m_pVehicles.back()->getPos();
134
135                 //Remove any vehicles that are no longer on the road
136                 if(m_pVehicles.at(0)->getPos() > m_RoadLength + MIN_SPACE_FOR_NEXT_VEHICLE)
137                 {
138                         delete m_pVehicles.front();
139                         m_pVehicles.erase(m_pVehicles.begin());
140                         m_NoVeh--;
```

```
141                 }
142                 laneEmpty = false;;
143         }
144         return laneEmpty;
145 }
```

Here is the call graph for this function:



4.41.3.6 int Lane::getIndex ()

Gets the index of the lane.

Returns:

The index of the lane

Definition at line 268 of file Lane.cpp.

References m_iLane.

Referenced by ImplementLaneChange().

```
269 {
270         return m_iLane;
271 }
```

4.41.3.7 double Lane::getLastPos ()

Gets the position of the last vehicle in the lane.

Returns:

The position of the last vehicle in the lane

Definition at line 259 of file Lane.cpp.

References m_PrevVehiclePosition.

```
260 {
261         return m_PrevVehiclePosition;
262 }
```

4.41.3.8 int Lane::getNoVeh ()

Gets the number of vehicles in the lane.

Returns:

The number of vehicles in the lane

Definition at line 250 of file Lane.cpp.

References m_pVehicles.

```
251 {
252         return m_pVehicles.size();
253 }
```

4.41.3.9 double Lane::getLength ()

Gets the length of the Lane.

Returns:

The length of the Lane

Definition at line 205 of file Lane.cpp.

References m_RoadLength.

```
206 {
207         return m_RoadLength;
208 }
```

4.41.3.10 void Lane::insert (int *i*, Vehicle * *pVeh*)

Inserts a Vehicle into the Lane.

This function inserts a given Vehicle into a particular position in a Lane

Parameters:

i The position to insert the Vehicle into

pVeh The Vehicle to insert

Definition at line 436 of file Lane.cpp.

References Vehicle::getPos(), m_NoVeh, m_PrevVehiclePosition, and m_pVehicles.

Referenced by ImplementLaneChange().

```
437 {
438         if(i == m_pVehicles.size())     // insert at end
439         {
440                 m_pVehicles.push_back(pVeh);
441                 m_PrevVehiclePosition = pVeh->getPos();
442         }
443         else if(i <= 0)                         // insert at front
444                 m_pVehicles.insert(m_pVehicles.begin(), pVeh);
445         else                                                    // insert between front & back, i.e. la
446                 m_pVehicles.insert(m_pVehicles.begin()+i, pVeh);
447
448         m_NoVeh++;
449 }
```

Here is the call graph for this function:

4.41.3.11 void Lane::setRightLane (Lane * *right*)

Sets a pointer to the Lane to the right of the current Lane The Lane index is in global coords:

- For DriveOnRight the lanes are numbered from bottom to top 0.

..n

- For left driving the lanes are numbered from top to bottom 0...n

Parameters:

right The Lane to point to

Definition at line 241 of file Lane.cpp.

References m_RightLane.

```
242 {
243         m_RightLane = right;
244 }
```

4.41.3.12 void Lane::setLeftLane (Lane * *left*)

Sets a pointer to the Lane to the left of the current Lane The Lane index is in global coords:

- For DriveOnRight the lanes are numbered from bottom to top 0.

..n

- For left driving the lanes are numbered from top to bottom 0...n

Parameters:

left The Lane to point to

Definition at line 229 of file Lane.cpp.

References m_LeftLane.

```
230 {
231         m_LeftLane = left;
232 }
```

4.41.3.13 std::vector< Vehicle * > Lane::getPos ()

Gets all the Vehicles currently in the Lane.

This function returns the vector which contains pointers to all Vehicles which are currently in the Lane

Returns:

All of the Vehicles in the Lane

Definition at line 217 of file Lane.cpp.

References m_pVehicles.

Referenced by FinadAdjacentVehicle().

```
218 {
219         return m_pVehicles;
220 }
```

4.41.3.14 Vehicle * Lane::FinadAdjacentVehicle (Lane * *AdjacentLane*, double *location*, bool *front*, int * *iAdjacentLane* = NULL) [private]

Finds a vehicle in the adjacent lane.

Parameters:

AdjacentLane The lane at which we are looking

location The location of the vehicle wishing to change

front Whether or not we are looking for the vehicle in front of the current vehicle's position

iAdjacentLane The index to switch into in the adjacent lane

Returns:

The vehicle desired

This function finds the vehicle that in the adjacent lane, in front of (front == true) the given location, or behind (front == false) the given location. It returns the appropriate vehicle, and sets iAdjacentIndex to the index into which the current vehicle should change if it changes lane.

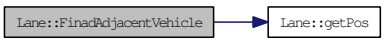
Definition at line 400 of file Lane.cpp.

References getPos().

Referenced by ChangeLanes().

```
401 {
402         std::vector<Vehicle*> AdjacentLaneVeh = AdjacentLane->getPos();
403
404         if(AdjacentLaneVeh.size() == 0)
405                 return NULL;
406
407         // searches from back of nextLane up to the vehicle in front
408         int i = AdjacentLaneVeh.size() - 1;
409
410         while(i >= 0 && AdjacentLaneVeh.at(i)->getPos() < location)
411                 i--;
412
413         if(iAdjacentLane != NULL)       // i points to the vehicle in front of us in the other
414                 *iAdjacentLane = i+1;   // place the position in iAdjacentLane
415
416         //We need to check for bounds here, since we're returning a Vehicle, not an index
417
418         if(front && i >= 0)
419                 return AdjacentLaneVeh.at(i);
420         else if(front && i < 0)
421                 return NULL;
422         else if(!front && i == AdjacentLaneVeh.size() - 1)
423                 return NULL;
424         else
425                 return AdjacentLaneVeh.at(i+1);
426 }
```

Here is the call graph for this function:



4.41.3.15 Lane * Lane::ChangeLanes (Vehicle * *pVeh*) [private]

Changes a Vehicle's Lane if advantageous and possible.

Parameters:

pVeh The vehicle changing lanes

Returns:

Whether or not the vehicle has changed lane

This function is called whenever it is time for a Vehicle to check whether or not it would be advantageous to move to another Lane. The vehicle finds its neighbours in the left and right lanes, and then decides whether to change lane, or which lane to change to, based on their properties.

Definition at line 285 of file Lane.cpp.

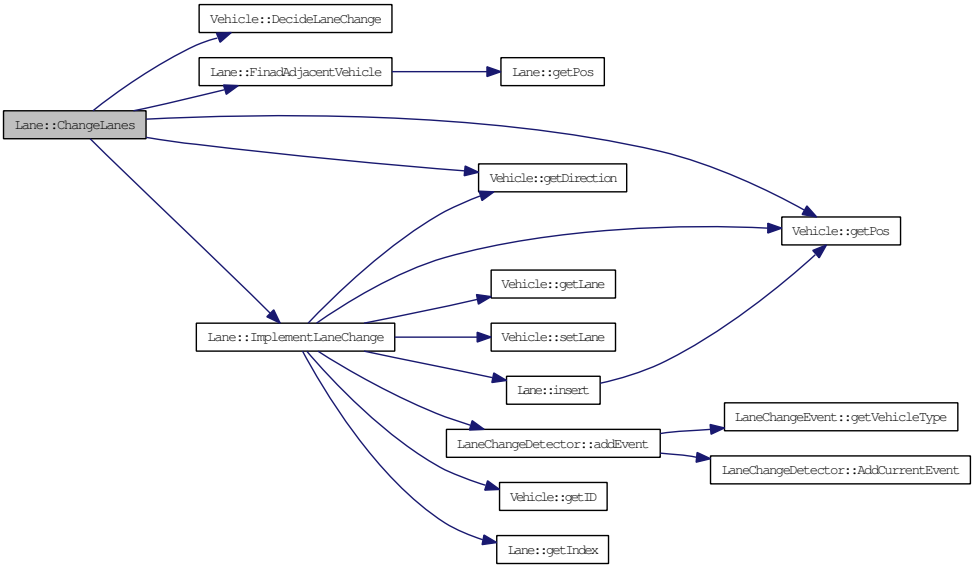
References Vehicle::DecideLaneChange(), FinadAdjacentVehicle(), Vehicle::getDirection(), Vehicle::getPos(), ImplementLaneChange(), m_LeftLane, and m_RightLane.

Referenced by update().

```
286 {
287         // since if it's a single lane you cannot change so let's not even bother
288         if(m_RightLane == NULL && m_LeftLane == NULL)
289                 return false;
290
291         bool DirPos = pVeh->getDirection();
292         Vehicle* LeftFrontVehicle = NULL;       Vehicle* LeftBackVehicle = NULL;
293         Vehicle* RightFrontVehicle = NULL;  Vehicle* RightBackVehicle = NULL;
294         int iLeftLane = 0;                                      int iRightLane = 0;
295         bool LeftLaneExists = m_LeftLane != NULL ? true : false;
296         bool RightLaneExists = m_RightLane != NULL ? true : false;
297
298         if(LeftLaneExists)
299         {
300                 LeftFrontVehicle = FinadAdjacentVehicle(m_LeftLane, pVeh->getPos(), true, &iLef
301                 LeftBackVehicle = FinadAdjacentVehicle(m_LeftLane, pVeh->getPos(), false);
302         }
303
304         if(RightLaneExists)
305         {
306                 RightFrontVehicle = FinadAdjacentVehicle(m_RightLane, pVeh->getPos(), true, &iR
307                 RightBackVehicle = FinadAdjacentVehicle(m_RightLane, pVeh->getPos(), false);
308         }
309         int LaneChangeID = 0;
310         if(DirPos)      // in DirPos
311                 LaneChangeID = pVeh->DecideLaneChange(  LeftFrontVehicle,       LeftBackVehicle
312
313
314         else    // in DirNeg so swap vehicles - see comment in Vehicle::DecideLaneChange()
315                 LaneChangeID = pVeh->DecideLaneChange(  RightFrontVehicle,      RightBackVehicl
316
317
318
319         if(LaneChangeID == 0)
320                 return NULL;
321         else
322         {
323                 Lane* pNewLane = ImplementLaneChange(pVeh, LaneChangeID, iLeftLane, iRightLane)
324                 return pNewLane;
```

```
325         }
326 }
```

Here is the call graph for this function:



4.41.3.16 Lane * Lane::ImplementLaneChange (Vehicle * *pVeh*, int *LaneChangeID*, int *iVehLeftLane*, int *iVehRightLane*) [private]

Changes a vehicle to another lane.

Parameters:

pVeh The vehicle changing lane

LaneChangeID The lane to change to

iVehLeftLane The index of the vehicle if moving to the left lane

iVehRightLane The index of the vehicle if moving to the right lane

This function inserts the vehicle that is changing lane into the appropriate lane. It also takes care of tracking lane changes if the user specifies that lane changes should be tracked.

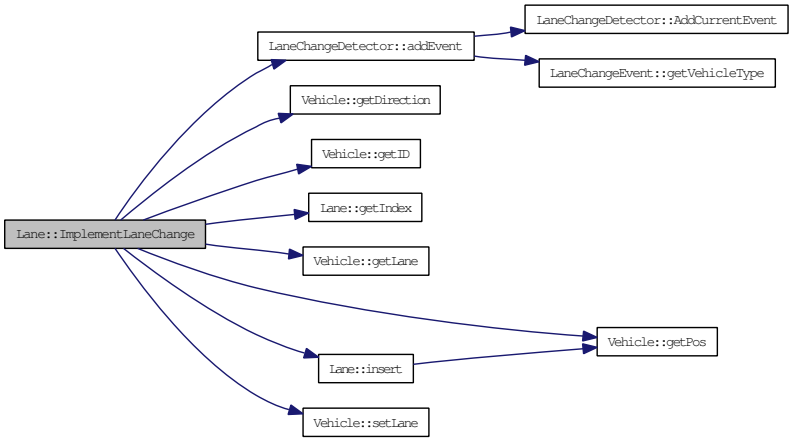
Definition at line 339 of file Lane.cpp.

References LaneChangeDetector::addEvent(), Vehicle::getDirection(), Vehicle::getID(), getIndex(), Vehicle::getLane(), Vehicle::getPos(), insert(), m_bTrackLaneChanges, m_CurTime, m_LeftLane, m_pLaneChangeDetector, m_RightLane, m_RoadLength, and Vehicle::setLane().

Referenced by ChangeLanes().

```
340 {
341         Lane* VehiclesLeftLane;         Lane* VehiclesRightLane;        Lane* LaneToChange;
342
343         int fromLane = pVeh->getLane();         // for lane change detector
344         double position = pVeh->getPos();
345         bool DirPos = pVeh->getDirection();
346         bool changeLeft = true;
347
348         if(DirPos)
349         {
350                 VehiclesLeftLane = m_LeftLane;          VehiclesRightLane = m_RightLane;
351         }
352         else
353         {
354                 VehiclesLeftLane = m_RightLane;         VehiclesRightLane = m_LeftLane;
355
356                 int temp = iVehLeftLane;                        // Swapping vehicle index in ne
357                 iVehLeftLane = iVehRightLane;
358                 iVehRightLane = temp;
359
360                 position = m_RoadLength - position;
361         }
362         if(LaneChangeID == 1)
363         {
364                 // change to the vehicle's left lane
365                 pVeh->setLane(VehiclesLeftLane->getIndex()+1);  // vehicle lane numbers are 1-h
366                 VehiclesLeftLane->insert(iVehLeftLane, pVeh);
367                 LaneToChange = VehiclesLeftLane;
368         }
369         else
370         {
371                 changeLeft = false;
372                 // change to vehicle's right lane
373                 pVeh->setLane(VehiclesRightLane->getIndex()+1); // vehicle lane numbers are 1-h
374                 VehiclesRightLane->insert(iVehRightLane, pVeh);
375                 LaneToChange = VehiclesRightLane;
376         }
377
378         if(m_bTrackLaneChanges) // If we are tracking changes, add this event
379         {
380                 int toLane = (LaneToChange->getIndex() + 1);
381                 LaneChangeEvent* lce = new LaneChangeEvent(position, fromLane, toLane, pVeh->ge
382                 m_pLaneChangeDetector->addEvent(lce);
383         }
384
385         return LaneToChange;
386 }
```

Here is the call graph for this function:



4.41.3.17 void Lane::CheckDetectors (const double *curTime*, double *prevPosition*, double *curPosition*, Vehicle * *pVeh*) [private]

Adds a vehicle to the detectors list of vehicles for that step.

Parameters:

curTime The current simulation time

prevPosition The vehicle's previous position

curPosition The vehicle's current position

pVeh The vehicle

This function takes account of all detectors that are currently on the road and checks to see if a given vehicle has passed any detectors in the last simulation step. If a vehicle has passed a detector, it is added to that detector

Definition at line 179 of file Lane.cpp.

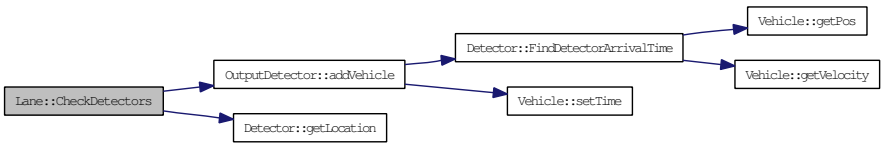
References OutputDetector::addVehicle(), Detector::getLocation(), m_pOutputDetector, and m_vDetectors.

Referenced by CheckFeatures().

```
180 {
181         // if the vehicle has moved past it in the last time step
182         // add the vehicle to the detector
183
185         int DetectorLocation = m_pOutputDetector->getLocation();
186         if(prevPosition <= DetectorLocation && curPosition >= DetectorLocation)
187                 m_pOutputDetector->addVehicle(pVeh, curTime);
188
190
191         for(int i = 0; i < m_vDetectors.size(); i++)
```

```
192         {
193                 DetectorLocation = m_vDetectors.at(i)->getLocation();
194                 if(prevPosition <= DetectorLocation && curPosition >= DetectorLocation)
195                         m_vDetectors.at(i)->addVehicle(pVeh, curTime);
196         }
197 }
```

Here is the call graph for this function:



4.41.3.18 void Lane::CheckRoadSegments (double *prevPosition*, double *curPosition*, Vehicle * *pVeh*) [private]

Checks whether a vehicle has entered or exited any road segments.

Parameters:

prevPosition The position of the vehicle before the update

curPosition The position of the vehicle after the update

pVeh The vehicle involved

This function takes account of all segments in the road - speedlimits, gradients, etc - and checks each update to see if a vehicle has entered or exited any of the segments on the road. Overlapping segments are accounted for, as if a vehicle exits a segment and has not entered a subsequent segment, it is checked to see whether the vehicle is still in the bounds of a previous segment

Definition at line 482 of file Lane.cpp.

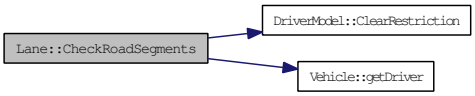
References DriverModel::ClearRestriction(), Vehicle::getDriver(), and m_RoadSegments.

Referenced by CheckFeatures().

```
483 {
484         bool entered = false;
485         bool exited = false;
486         int nSegments = m_RoadSegments.size();
487
488         for(int i = 0; i < nSegments; i++)                                          // for
489         {
490                 int start = m_RoadSegments.at(i)->getBeginning();       // its start and end
491                 int end = m_RoadSegments.at(i)->getEnd();
492
493                 if(prevPosition <= start && curPosition >= start)       // if we have entered t
494                 {
495                         m_RoadSegments.at(i)->addVehicle(pVeh);
```

```
496                         entered = true;
497                 }
498                 else if(prevPosition < end && curPosition >= end)       // else if we have exit
499                 {
500                         m_RoadSegments.at(i)->removeVehicle(pVeh);
501                         exited = true;
502                 }
503         }
504
505         if(exited && !entered) // if we have exited a segment, but not entered a subsequent one
506         {
507                 for(int j = 0; j < nSegments; j++)
508                 {
509                         int start = m_RoadSegments.at(j)->getBeginning();
510                         int end = m_RoadSegments.at(j)->getEnd();
511
512                         if(curPosition >= start && curPosition < end)
513                         {
514                                 m_RoadSegments.at(j)->addVehicle(pVeh);
515                                 entered = true;
516                                 break;
517                         }
518                 }
519                 if(!entered)
520                         pVeh->getDriver()->ClearRestriction();
521         }
522 }
```

Here is the call graph for this function:



4.41.4 Member Data Documentation

4.41.4.1 std::vector<Vehicle*> Lane::m_pVehicles [private]

Definition at line 58 of file Lane.h.

Referenced by clear(), getNoVeh(), getPos(), insert(), and update().

4.41.4.2 std::vector<Detector*> Lane::m_vDetectors [private]

Definition at line 59 of file Lane.h.

Referenced by CheckDetectors(), clear(), and Lane().

4.41.4.3 int Lane::m_Dir [private]

Definition at line 61 of file Lane.h.

4.41.4.4 int Lane::m_iLane [private]

Definition at line 62 of file Lane.h.

Referenced by getIndex(), and Lane().

4.41.4.5 double Lane::m_Flow [private]

Definition at line 63 of file Lane.h.

4.41.4.6 double Lane::m_RoadLength [private]

Definition at line 64 of file Lane.h.

Referenced by getLength(), ImplementLaneChange(), Lane(), and update().

4.41.4.7 int Lane::m_NoVeh [private]

Definition at line 65 of file Lane.h.

Referenced by insert(), Lane(), and update().

4.41.4.8 double Lane::m_PrevVehiclePosition [private]

Definition at line 66 of file Lane.h.

Referenced by getLastPos(), insert(), Lane(), and update().

4.41.4.9 bool Lane::m_AllowLaneChanging [private]

Definition at line 67 of file Lane.h.

Referenced by Lane(), and update().

4.41.4.10 bool Lane::m_DirPos [private]

Definition at line 68 of file Lane.h.

Referenced by Lane().

4.41.4.11 double Lane::m_CurTime [private]

Definition at line 69 of file Lane.h.

Referenced by ImplementLaneChange(), and update().

4.41.4.12 bool Lane::m_bTrackLaneChanges [private]

Definition at line 70 of file Lane.h.

Referenced by ImplementLaneChange(), and Lane().

4.41.4.13 std::vector<RoadSegment*> Lane::m_RoadSegments [private]

Definition at line 72 of file Lane.h.

Referenced by CheckRoadSegments(), clear(), and setRoadSegments().

4.41.4.14 Lane* Lane::m_LeftLane [private]

Definition at line 74 of file Lane.h.

Referenced by ChangeLanes(), ImplementLaneChange(), Lane(), and setLeftLane().

4.41.4.15 Lane* Lane::m_RightLane [private]

Definition at line 75 of file Lane.h.

Referenced by ChangeLanes(), ImplementLaneChange(), Lane(), and setRightLane().

4.41.4.16 OutputDetector* Lane::m_pOutputDetector [private]

Definition at line 76 of file Lane.h.

Referenced by CheckDetectors(), and setOutputDetector().

4.41.4.17 LaneChangeDetector* Lane::m_pLaneChangeDetector [private]

Definition at line 77 of file Lane.h.

Referenced by ImplementLaneChange(), and Lane().

4.41.4.18 double Lane::MIN_SPACE_FOR_NEXT_VEHICLE [private]

Definition at line 79 of file Lane.h.

Referenced by Lane(), and update().

The documentation for this class was generated from the following files:

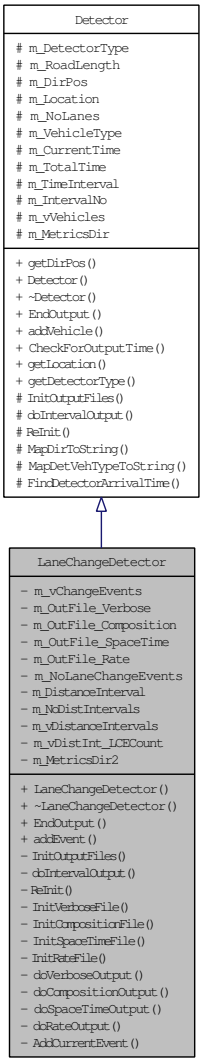
- D:/~Research/Code/C++/EvolveTraffic/Lane.h
- D:/~Research/Code/C++/EvolveTraffic/Lane.cpp

4.42 LaneChangeDetector Class Reference

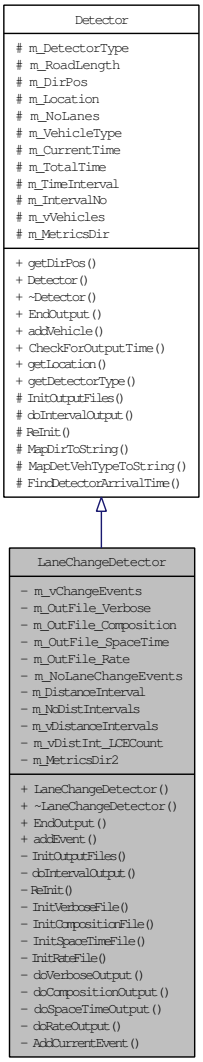
A derived class to represent a detector that tracks lane changes.

`#include <LaneChangeDetector.h>`

Inheritance diagram for LaneChangeDetector:



Collaboration diagram for LaneChangeDetector:



Public Member Functions

- LaneChangeDetector (std::string dir, double dt, double dx, int roadLen, bool DirPos, WORD VehType)

  Constructor.

- virtual ~LaneChangeDetector ()

  Destructor.

- virtual void EndOutput ()

  Cleans up end of file output.

- void addEvent (LaneChangeEvent *lce)

  Adds an event to the detector.

Private Member Functions

- virtual void InitOutputFiles ()

  Initialises the output files.

- virtual void doIntervalOutput ()

  Processes output.

- virtual void ReInit ()

  Re-initialises the detector after output.

- void InitVerboseFile ()

  Initialises the verbose file for output.

- void InitCompositionFile ()

  Initialises the composition file for output.

- void InitSpaceTimeFile ()

  Initialises the space-time file for output.

- void InitRateFile ()

  Initialises the rate file for output.

- void doVerboseOutput ()

  Processes verbose output.

- void doCompositionOutput ()

  Processes composition output.

- void doSpaceTimeOutput ()

  Processes space-time output.

- void doRateOutput ()

  Processes rate output.

- void AddCurrentEvent (LaneChangeEvent *lce)

Private Attributes

- std::vector< LaneChangeEvent * > m_vChangeEvents
- std::ofstream m_OutFile_Verbose
- std::ofstream m_OutFile_Composition
- std::ofstream m_OutFile_SpaceTime
- std::ofstream m_OutFile_Rate
- int m_NoLaneChangeEvents
- int m_DistanceInterval
- int m_NoDistIntervals
- std::vector< int > m_vDistanceIntervals
- std::vector< int > m_vDistInt_LCECount
- std::string m_MetricsDir2

4.42.1 Detailed Description

A derived class to represent a detector that tracks lane changes.

Definition at line 18 of file LaneChangeDetector.h.

4.42.2 Constructor & Destructor Documentation

4.42.2.1 LaneChangeDetector::LaneChangeDetector (std::string *dir*, double *dt*, double *dx*, int *roadLen*, bool *DirPos*, WORD *VehType*)

Constructor.

Parameters:

dir The detector output directory

dt The time interval to use

dx The distance interval to use

roadLen The length of the road

DirPos Whether the detector is in the positive direction

VehType The type of vehicles to track

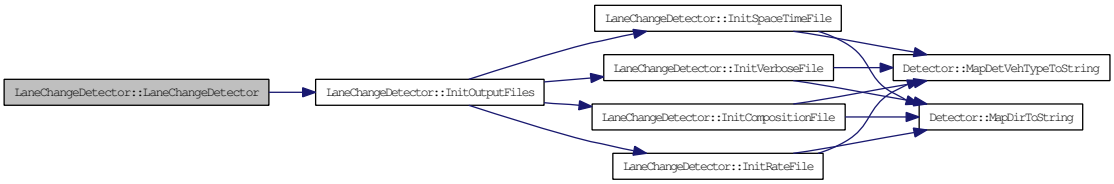
Definition at line 27 of file LaneChangeDetector.cpp.

References InitOutputFiles(), Detector::m_DetectorType, Detector::m_DirPos, m_DistanceInterval, Detector::m_MetricsDir, m_NoDistIntervals, m_NoLaneChangeEvents, Detector::m_RoadLength, Detector::m_TimeInterval, m_vDistanceIntervals, m_vDistInt_LCECount, Detector::m_VehicleType, and METRICS_TYPE_LANE_CHANGE.

```
29 {
30         m_MetricsDir = dir;
31         m_TimeInterval = dt;
32         m_DistanceInterval = dx;
33         m_RoadLength = roadLen;
```

```
34          m_DirPos = DirPos;
35          m_VehicleType = VehType;
36
37          m_NoLaneChangeEvents = 0;
38          m_DetectorType = METRICS_TYPE_LANE_CHANGE;
39
40          int curPos = 0;
41          while(curPos < m_RoadLength)
42          {
43                  m_vDistanceIntervals.push_back(curPos);
44                  curPos += m_DistanceInterval;
45          }
46          m_vDistanceIntervals.push_back(m_RoadLength); // enclose the last interval
47          m_NoDistIntervals = m_vDistanceIntervals.size();
48          m_vDistInt_LCECount.assign(m_NoDistIntervals, 0);
49
50          InitOutputFiles();
51  }
```

Here is the call graph for this function:



4.42.2.2 LaneChangeDetector::~LaneChangeDetector () [virtual]

Destructor.

Definition at line 54 of file LaneChangeDetector.cpp.

```
55  {
56
57  }
```

4.42.3 Member Function Documentation

4.42.3.1 void LaneChangeDetector::EndOutput () [virtual]

Cleans up end of file output.

Reimplemented from Detector.

Definition at line 129 of file LaneChangeDetector.cpp.

References m_OutFile_Composition, m_OutFile_Rate, m_OutFile_SpaceTime, m_OutFile_Verbose, Detector::m_VehicleType, and METRICS_VEH_ALL.

```
130 {
131         if(m_VehicleType == METRICS_VEH_ALL)
132                 m_OutFile_Composition.close();
```

```
133
134         m_OutFile_Verbose.close();
135         m_OutFile_SpaceTime.close();
136         m_OutFile_Rate.close();
137 }
```

4.42.3.2 void LaneChangeDetector::addEvent (LaneChangeEvent * *pLCE*)

Adds an event to the detector.

Parameters:

pLCE The event to add

Depending on the type of vehicles being detected, this function will add the event if it contains the correct type of vehicle

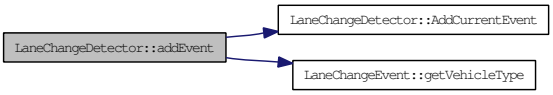
Definition at line 77 of file LaneChangeDetector.cpp.

References AddCurrentEvent(), LaneChangeEvent::getVehicleType(), Detector::m_-VehicleType, METRICS_VEH_ALL, METRICS_VEH_CAR, METRICS_VEH_-CRANE, METRICS_VEH_LARGETRUCK, METRICS_VEH_LOWLOADER, METRICS_VEH_SMALLTRUCK, VEH_ID_CAR, VEH_ID_CRANE, VEH_ID_-LARGETRUCK, VEH_ID_LOWLOADER, and VEH_ID_SMALLTRUCK.

Referenced by Lane::ImplementLaneChange().

```
78 {
79         WORD vehID = pLCE->getVehicleType();
80
81         switch(m_VehicleType)
82         {
83                 case METRICS_VEH_ALL:
84                         AddCurrentEvent(pLCE);
85                         break;
86                 case METRICS_VEH_CAR:
87                         if(vehID == VEH_ID_CAR)
88                                 AddCurrentEvent(pLCE);
89                         break;
90                 case METRICS_VEH_SMALLTRUCK:
91                         if(vehID == VEH_ID_SMALLTRUCK)
92                                 AddCurrentEvent(pLCE);
93                         break;
94                 case METRICS_VEH_LARGETRUCK:
95                         if(vehID == VEH_ID_LARGETRUCK)
96                                 AddCurrentEvent(pLCE);
97                         break;
98                 case METRICS_VEH_CRANE:
99                         if(vehID == VEH_ID_CRANE)
100                                 AddCurrentEvent(pLCE);
101                         break;
102                 case METRICS_VEH_LOWLOADER:
103                         if(vehID == VEH_ID_LOWLOADER)
104                                 AddCurrentEvent(pLCE);
105                         break;
106                 default:
107                         AddCurrentEvent(pLCE);
108         }
109 }
```

Here is the call graph for this function:



4.42.3.3 void LaneChangeDetector::InitOutputFiles () [private, virtual]

Initialises the output files.

Reimplemented from Detector.

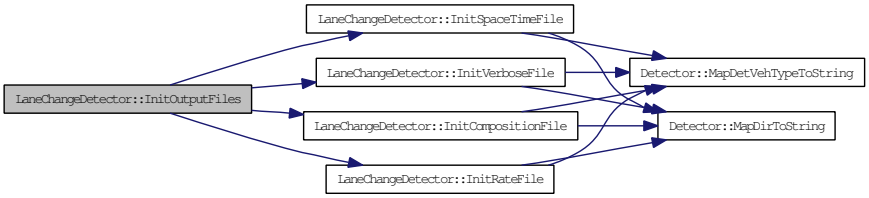
Definition at line 60 of file LaneChangeDetector.cpp.

References InitCompositionFile(), InitRateFile(), InitSpaceTimeFile(), InitVerboseFile(), Detector::m_VehicleType, and METRICS_VEH_ALL.

Referenced by LaneChangeDetector().

```
61 {
62         if(m_VehicleType == METRICS_VEH_ALL)
63                 InitCompositionFile();  // only makes sense for all vehicles
64
65         InitVerboseFile();
66         InitSpaceTimeFile();
67         InitRateFile();
68 }
```

Here is the call graph for this function:



4.42.3.4 void LaneChangeDetector::doIntervalOutput () [private, virtual]

Processes output.

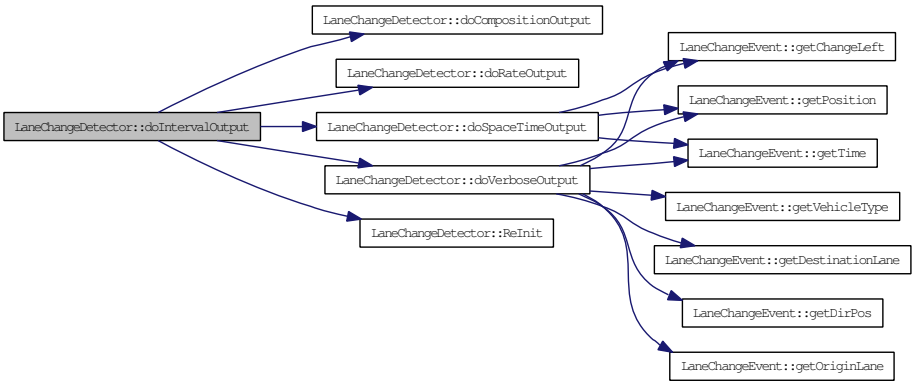
Reimplemented from Detector.

Definition at line 118 of file LaneChangeDetector.cpp.

References doCompositionOutput(), doRateOutput(), doSpaceTimeOutput(), doVerboseOutput(), and ReInit().

```
119 {
120         doVerboseOutput();
121         doCompositionOutput();
122         doSpaceTimeOutput();
123         doRateOutput();
124
125         ReInit();
126 }
```

Here is the call graph for this function:



4.42.3.5 void LaneChangeDetector::ReInit () [private, virtual]

Re-initialises the detector after output.

Reimplemented from Detector.

Definition at line 140 of file LaneChangeDetector.cpp.

References m_NoDistIntervals, m_NoLaneChangeEvents, m_vChangeEvents, and m_vDistInt_LCECount.

Referenced by doIntervalOutput().

```
141 {
142         m_NoLaneChangeEvents = 0;
143
144         m_vDistInt_LCECount.clear();
145         m_vDistInt_LCECount.assign(m_NoDistIntervals, 0);
146
147         for(int i = 0; i < m_vChangeEvents.size(); i++)
148                 delete m_vChangeEvents.at(i);
149         m_vChangeEvents.clear();
150
151 }
```

4.42.3.6 void LaneChangeDetector::InitVerboseFile () [private]

Initialises the verbose file for output.

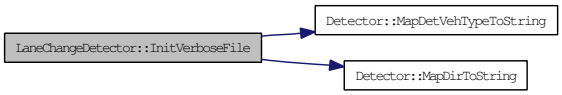
Definition at line 154 of file LaneChangeDetector.cpp.

References Detector::m_DirPos, Detector::m_MetricsDir, m_OutFile_Verbose, Detector::m_VehicleType, Detector::MapDetVehTypeToString(), and Detector::MapDirToString().

Referenced by InitOutputFiles().

```
155 {
156         std::string file;
157         file = m_MetricsDir;
158         file += "LC_All_" + MapDetVehTypeToString(m_VehicleType) + "_"
159                                     + MapDirToString(m_DirPos) + ".csv";
160
161         m_OutFile_Verbose.open(file.c_str(), std::ios::out);
162
163         m_OutFile_Verbose << "LANE CHANGE - VERBOSE OUTPUT OF EVENTS" << '\n';
164         m_OutFile_Verbose << "Time,";
165         m_OutFile_Verbose << "Position,";
166         m_OutFile_Verbose << "From Lane,";
167         m_OutFile_Verbose << "To Lane,";
168         m_OutFile_Verbose << "Right/Left,";
169         m_OutFile_Verbose << "Type" << '\n';
170         m_OutFile_Verbose << std::endl;
171 }
```

Here is the call graph for this function:



4.42.3.7 void LaneChangeDetector::InitCompositionFile () [private]

Initialises the composition file for output.

Definition at line 174 of file LaneChangeDetector.cpp.

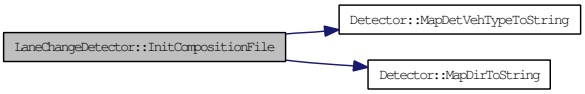
References Detector::m_DirPos, Detector::m_MetricsDir, m_OutFile_Composition, Detector::m_VehicleType, Detector::MapDetVehTypeToString(), and Detector::MapDirToString().

Referenced by InitOutputFiles().

```
175 {
176         std::string file;
177         file = m_MetricsDir;
178         file += "LC_Comp_" + MapDetVehTypeToString(m_VehicleType) + "_"
179                                     + MapDirToString(m_DirPos) + ".csv";
180
181         m_OutFile_Composition.open(file.c_str(), std::ios::out);
182
183         m_OutFile_Composition << "LANE CHANGE - COMPOSITION OF EVENTS" << "\n";
184         m_OutFile_Composition << "Time Interval"        << ",";
185         m_OutFile_Composition << "Car Changes"          << ",";
```

```
186         m_OutFile_Composition << "SmallTr Changes"      << ",";
187         m_OutFile_Composition << "LargeTr Changes"      << ",";
188         m_OutFile_Composition << "Crane Changes"        << ",";
189         m_OutFile_Composition << "LowLoad Changes"      << '\n';
190
191 }
```

Here is the call graph for this function:



4.42.3.8 void LaneChangeDetector::InitSpaceTimeFile () [private]

Initialises the space-time file for output.

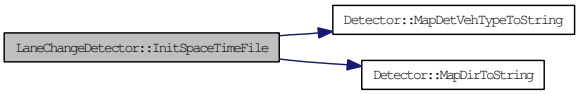
Definition at line 194 of file LaneChangeDetector.cpp.

References Detector::m_DirPos, Detector::m_MetricsDir, m_OutFile_SpaceTime, Detector::m_VehicleType, Detector::MapDetVehTypeToString(), and Detector::MapDirToString().

Referenced by InitOutputFiles().

```
195 {
196         std::string file;
197         file = m_MetricsDir;
198         file += "LC_ST_" + MapDetVehTypeToString(m_VehicleType) + "_"
199                                  + MapDirToString(m_DirPos) + ".csv";
200
201         m_OutFile_SpaceTime.open(file.c_str(), std::ios::out);
202
203         m_OutFile_SpaceTime << "LANE CHANGE - SPACE TIME OF EVENTS" << '\n';
204         m_OutFile_SpaceTime << "Left To Right," << ',' << "Right To Left" << '\n';
205         m_OutFile_SpaceTime << "Time (s),Location (m),Time (s),Location (m)" << '\n';
206 }
```

Here is the call graph for this function:



4.42.3.9 void LaneChangeDetector::InitRateFile () [private]

Initialises the rate file for output.

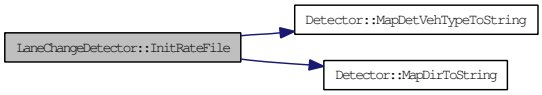
Definition at line 209 of file LaneChangeDetector.cpp.

References Detector::m_DirPos, Detector::m_MetricsDir, m_NoDistIntervals, m_OutFile_Rate, m_vDistanceIntervals, Detector::m_VehicleType, Detector::MapDetVehTypeToString(), and Detector::MapDirToString().

Referenced by InitOutputFiles().

```
210 {
211         std::string file;
212         file = m_MetricsDir;
213         file += "LC_Rate_" + MapDetVehTypeToString(m_VehicleType) + "_"
214                                     + MapDirToString(m_DirPos) + ".csv";
215
216         m_OutFile_Rate.open(file.c_str(), std::ios::out);
217
218         m_OutFile_Rate << "LANE CHANGE - RATES BY LOCATION AND TIME" << '\n';
219         m_OutFile_Rate << "Time (s),Location (m)" << '\n';
220         m_OutFile_Rate << ",";
221         for(int i = 0; i < m_NoDistIntervals; i++)
222                 m_OutFile_Rate << m_vDistanceIntervals[i] << ",";
223         m_OutFile_Rate << '\n';
224 }
```

Here is the call graph for this function:



4.42.3.10 void LaneChangeDetector::doVerboseOutput () [private]

Processes verbose output.

Definition at line 227 of file LaneChangeDetector.cpp.

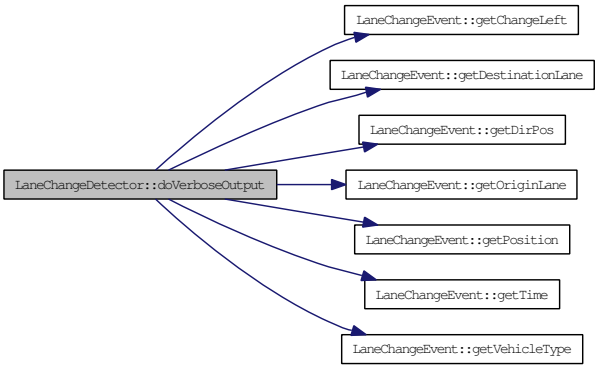
References LaneChangeEvent::getChangeLeft(), LaneChangeEvent::getDestinationLane(), LaneChangeEvent::getDirPos(), LaneChangeEvent::getOriginLane(), LaneChangeEvent::getPosition(), LaneChangeEvent::getTime(), LaneChangeEvent::getVehicleType(), m_NoLaneChangeEvents, m_OutFile_Verbose, m_vChangeEvents, VEH_ID_CAR, VEH_ID_CRANE, VEH_ID_LARGETRUCK, VEH_ID_LOWLOADER, and VEH_ID_SMALLTRUCK.

Referenced by doIntervalOutput().

```
228 {
229         for(int i = 0; i < m_NoLaneChangeEvents; i++)
230         {
231                 LaneChangeEvent* curEvent = m_vChangeEvents.at(i);
232
233                 std::string change = (curEvent->getChangeLeft()) ? "Left" : "Right";
234                 std::string dir = (curEvent->getDirPos()) ? "Pos" : "Neg";
235
236                 std::string type;
237
238                 switch(curEvent->getVehicleType())
```

```
239                 {
240                         case VEH_ID_CAR:                type = "Car";           break;
241                         case VEH_ID_SMALLTRUCK: type = "SmallTr";       break;
242                         case VEH_ID_LARGETRUCK: type = "LargeTr";       break;
243                         case VEH_ID_CRANE:              type = "Crane";         break;
244                         case VEH_ID_LOWLOADER:  type = "Lowload";       break;
245                 }
246
247                 m_OutFile_Verbose << curEvent->getTime() << " ,"
248                                                 << curEvent->getPosition()              << " ,"
249                                                 << curEvent->getOriginLane()            << " ,"
250                                                 << curEvent->getDestinationLane()       << " ,"
251                                                 << change
252                                                 << type << '\n';
253         }
254 }
```

Here is the call graph for this function:



4.42.3.11 void LaneChangeDetector::doCompositionOutput () [private]

Processes composition output.

Definition at line 257 of file LaneChangeDetector.cpp.

References m_NoLaneChangeEvents, m_OutFile_Composition, Detector::m_TotalTime, m_vChangeEvents, VEH_ID_CAR, VEH_ID_CRANE, VEH_ID_LARGETRUCK, VEH_ID_LOWLOADER, and VEH_ID_SMALLTRUCK.

Referenced by doIntervalOutput().

```
258 {
259         int nCar = 0;
260         int nSmallTruck = 0;
261         int nLargeTruck = 0;
262         int nCrane = 0;
263         int nLowLoader = 0;
264
265         for(int i = 0; i < m_NoLaneChangeEvents; i++)
266         {
267                 WORD VehType = m_vChangeEvents.at(i)->getVehicleType();
```

```
268
269                 switch(VehType)
270                 {
271                         case VEH_ID_CAR:                nCar++;                 break;
272                         case VEH_ID_SMALLTRUCK: nSmallTruck++;  break;
273                         case VEH_ID_LARGETRUCK: nLargeTruck++;  break;
274                         case VEH_ID_CRANE:              nCrane++;               break;
275                         case VEH_ID_LOWLOADER:  nLowLoader++;   break;
276                         default: nCar++;
277                 }
278         }
279
280         m_OutFile_Composition << m_TotalTime    << ",";
281         m_OutFile_Composition << nCar                   << ",";
282         m_OutFile_Composition << nSmallTruck    << ",";
283         m_OutFile_Composition << nLargeTruck    << ",";
284         m_OutFile_Composition << nCrane                 << ",";
285         m_OutFile_Composition << nLowLoader             << '\n';
286 }
```

4.42.3.12 void LaneChangeDetector::doSpaceTimeOutput () [private]

Processes space-time output.

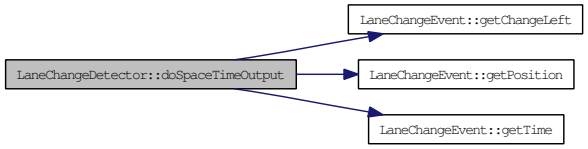
Definition at line 289 of file LaneChangeDetector.cpp.

References LaneChangeEvent::getChangeLeft(), LaneChangeEvent::getPosition(), LaneChangeEvent::getTime(), m_NoLaneChangeEvents, m_OutFile_SpaceTime, and m_vChangeEvents.

Referenced by doIntervalOutput().

```
290 {
291         for(int i = 0; i < m_NoLaneChangeEvents; i++)
292         {
293                 LaneChangeEvent* pLCE = m_vChangeEvents.at(i);
294                 double time                     = pLCE->getTime();
295                 double position         = pLCE->getPosition();
296                 bool bChangeToLeft      = pLCE->getChangeLeft();
297
298                 if(bChangeToLeft)               // send to right hand column
299                         m_OutFile_SpaceTime << ",,";
300
301                 m_OutFile_SpaceTime << time << ',' << position << '\n';
302         }
303 }
```

Here is the call graph for this function:



4.42.3.13 void LaneChangeDetector::doRateOutput () [private]

Processes rate output.

Definition at line 306 of file LaneChangeDetector.cpp.

References m_DistanceInterval, m_NoDistIntervals, m_NoLaneChangeEvents, m_OutFile_Rate, Detector::m_TimeInterval, Detector::m_TotalTime, m_vChangeEvents, m_vDistanceIntervals, m_vDistInt_LCECount, and SECS_PER_HOUR.

Referenced by doIntervalOutput().

```
307 {
308         for(int i = 0; i < m_NoLaneChangeEvents; i++)
309         {
310                 double position = m_vChangeEvents.at(i)->getPosition();
311                 int iDistInterval = 0;
312                 // find the interval to be incremented
313                 while( !(position < m_vDistanceIntervals[iDistInterval]) )
314                         iDistInterval++;
315
316                 m_vDistInt_LCECount[iDistInterval]++;   // count it
317         }
318
319         double deltaX = (double)m_DistanceInterval/1000;                // change to km
320         double deltaT = (double)m_TimeInterval/SECS_PER_HOUR;   // change to hour
321
322         m_OutFile_Rate << m_TotalTime;
323         for(i = 0; i < m_NoDistIntervals; i++)
324         {
325                 double n = m_vDistInt_LCECount[i];
326                 double rate = n/(deltaX*deltaT);
327                 m_OutFile_Rate << ',' << rate;
328         }
329         m_OutFile_Rate << '\n';
330 }
```

4.42.3.14 void LaneChangeDetector::AddCurrentEvent (LaneChangeEvent * *lce*) [private]

Definition at line 111 of file LaneChangeDetector.cpp.

References m_NoLaneChangeEvents, and m_vChangeEvents.

Referenced by addEvent().

```
112 {
113         m_vChangeEvents.push_back(lce);
114         m_NoLaneChangeEvents++;
115 }
```

4.42.4 Member Data Documentation

4.42.4.1 std::vector<LaneChangeEvent*> LaneChangeDetector::m_vChangeEvents [private]

Definition at line 46 of file LaneChangeDetector.h.

Referenced by AddCurrentEvent(), doCompositionOutput(), doRateOutput(), doSpaceTimeOutput(), doVerboseOutput(), and ReInit().

4.42.4.2 std::ofstream LaneChangeDetector::m_OutFile_Verbose [private]

Definition at line 48 of file LaneChangeDetector.h.

Referenced by doVerboseOutput(), EndOutput(), and InitVerboseFile().

4.42.4.3 std::ofstream LaneChangeDetector::m_OutFile_Composition [private]

Definition at line 49 of file LaneChangeDetector.h.

Referenced by doCompositionOutput(), EndOutput(), and InitCompositionFile().

4.42.4.4 std::ofstream LaneChangeDetector::m_OutFile_SpaceTime [private]

Definition at line 50 of file LaneChangeDetector.h.

Referenced by doSpaceTimeOutput(), EndOutput(), and InitSpaceTimeFile().

4.42.4.5 std::ofstream LaneChangeDetector::m_OutFile_Rate [private]

Definition at line 51 of file LaneChangeDetector.h.

Referenced by doRateOutput(), EndOutput(), and InitRateFile().

4.42.4.6 int LaneChangeDetector::m_NoLaneChangeEvents [private]

Definition at line 53 of file LaneChangeDetector.h.

Referenced by AddCurrentEvent(), doCompositionOutput(), doRateOutput(), doSpaceTimeOutput(), doVerboseOutput(), LaneChangeDetector(), and ReInit().

4.42.4.7 int LaneChangeDetector::m_DistanceInterval [private]

Definition at line 54 of file LaneChangeDetector.h.

Referenced by doRateOutput(), and LaneChangeDetector().

4.42.4.8 int LaneChangeDetector::m_NoDistIntervals [private]

Definition at line 55 of file LaneChangeDetector.h.

Referenced by doRateOutput(), InitRateFile(), LaneChangeDetector(), and ReInit().

4.42.4.9 std::vector<int> LaneChangeDetector::m_vDistanceIntervals [private]

Definition at line 56 of file LaneChangeDetector.h.

Referenced by doRateOutput(), InitRateFile(), and LaneChangeDetector().

4.42.4.10 std::vector<int> LaneChangeDetector::m_vDistInt_LCECount [private]

Definition at line 57 of file LaneChangeDetector.h.

Referenced by doRateOutput(), LaneChangeDetector(), and ReInit().

4.42.4.11 std::string LaneChangeDetector::m_MetricsDir2 [private]

Definition at line 59 of file LaneChangeDetector.h.

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/LaneChangeDetector.h
- D:/~Research/Code/C++/EvolveTraffic/LaneChangeDetector.cpp

4.43 LaneChangeEvent Class Reference

A container class to represent the relevant information from a lane change event.

`#include <LaneChangeEvent.h>`

Public Member Functions

- double getTime ()
- WORD getVehicleType ()
- bool getDirPos ()
- int getOriginLane ()
- int getPosition ()
- int getDestinationLane ()
- bool getChangeLeft ()
- LaneChangeEvent (int pos, int fromLane, int toLane, int type, bool left, bool dirPos, double time)

  Constructor.

- virtual ~LaneChangeEvent ()

  Destructor.

Private Attributes

- int m_EventPosition

  The position of the event.

- int m_OriginLane

  The lane from which the vehicle changed.

- int m_DestinationLane

  The lane to which the vehicle changed.

- int m_VehType

  The type of vehicle involved.

- bool m_MovingLeft

  Whether the vehicle changed into a left lane or not.

- bool m_DirPos

  Whether the vehicle is travelling in a positive direction or not.

- double m_EventTime

  The time at which the event occured.

4.43.1 Detailed Description

A container class to represent the relevant information from a lane change event.

Definition at line 17 of file LaneChangeEvent.h.

4.43.2 Constructor & Destructor Documentation

4.43.2.1 LaneChangeEvent::LaneChangeEvent (int *pos*, int *fromLane*, int *toLane*, int *type*, bool *moveLeft*, bool *dirPos*, double *time*)

Constructor.

Parameters:

pos The position of the event

fromLane The lane from which the vehicle changed

toLane The lane to which the vehicle changed

type The type of vehicle involved

moveLeft Whether the vehicle changed into a left lane or not

dirPos Whether the vehicle is travelling in a positive direction or not

time The time at which the event occured

Definition at line 29 of file LaneChangeEvent.cpp.

References m_DestinationLane, m_DirPos, m_EventPosition, m_EventTime, m_MovingLeft, m_OriginLane, and m_VehType.

```
30 {
31         m_EventPosition = pos;
32         m_OriginLane = fromLane;
33         m_DestinationLane = toLane;
34         m_VehType = type;
35         m_MovingLeft = moveLeft;
36         m_DirPos = dirPos;
37         m_EventTime = time;
38 }
```

4.43.2.2 LaneChangeEvent::~LaneChangeEvent () [virtual]

Destructor.

Definition at line 41 of file LaneChangeEvent.cpp.

```
42 {
43
44 }
```

4.43.3 Member Function Documentation

4.43.3.1 double LaneChangeEvent::getTime ()

Definition at line 76 of file LaneChangeEvent.cpp.

References m_EventTime.

Referenced by LaneChangeDetector::doSpaceTimeOutput(), and LaneChangeDetector::doVerboseOutput().

```
77 {
78         return m_EventTime;
79 }
```

4.43.3.2 WORD LaneChangeEvent::getVehicleType ()

Definition at line 71 of file LaneChangeEvent.cpp.

References m_VehType.

Referenced by LaneChangeDetector::addEvent(), and LaneChangeDetector::doVerboseOutput().

```
72 {
73         return m_VehType;
74 }
```

4.43.3.3 bool LaneChangeEvent::getDirPos ()

Definition at line 66 of file LaneChangeEvent.cpp.

References m_DirPos.

Referenced by LaneChangeDetector::doVerboseOutput().

```
67 {
68         return m_DirPos;
69 }
```

4.43.3.4 int LaneChangeEvent::getOriginLane ()

Definition at line 61 of file LaneChangeEvent.cpp.

References m_OriginLane.

Referenced by LaneChangeDetector::doVerboseOutput().

```
62 {
63         return m_OriginLane;
64 }
```

4.43.3.5 int LaneChangeEvent::getPosition ()

Definition at line 56 of file LaneChangeEvent.cpp.

References m_EventPosition.

Referenced by LaneChangeDetector::doSpaceTimeOutput(), and LaneChangeDetector::doVerboseOutput().

```
57 {
58         return m_EventPosition;
59 }
```

4.43.3.6 int LaneChangeEvent::getDestinationLane ()

Definition at line 51 of file LaneChangeEvent.cpp.

References m_DestinationLane.

Referenced by LaneChangeDetector::doVerboseOutput().

```
52 {
53         return m_DestinationLane;
54 }
```

4.43.3.7 bool LaneChangeEvent::getChangeLeft ()

Definition at line 46 of file LaneChangeEvent.cpp.

References m_MovingLeft.

Referenced by LaneChangeDetector::doSpaceTimeOutput(), and LaneChangeDetector::doVerboseOutput().

```
47 {
48         return m_MovingLeft;
49 }
```

4.43.4 Member Data Documentation

4.43.4.1 int LaneChangeEvent::m_EventPosition [private]

The position of the event.

Definition at line 32 of file LaneChangeEvent.h.

Referenced by getPosition(), and LaneChangeEvent().

4.43.4.2 int LaneChangeEvent::m_OriginLane [private]

The lane from which the vehicle changed.

Definition at line 34 of file LaneChangeEvent.h.

Referenced by getOriginLane(), and LaneChangeEvent().

4.43.4.3 int LaneChangeEvent::m_DestinationLane [private]

The lane to which the vehicle changed.

Definition at line 36 of file LaneChangeEvent.h.

Referenced by getDestinationLane(), and LaneChangeEvent().

4.43.4.4 int LaneChangeEvent::m_VehType [private]

The type of vehicle involved.

Definition at line 38 of file LaneChangeEvent.h.

Referenced by getVehicleType(), and LaneChangeEvent().

4.43.4.5 bool LaneChangeEvent::m_MovingLeft [private]

Whether the vehicle changed into a left lane or not.

Definition at line 40 of file LaneChangeEvent.h.

Referenced by getChangeLeft(), and LaneChangeEvent().

4.43.4.6 bool LaneChangeEvent::m_DirPos [private]

Whether the vehicle is travelling in a positive direction or not.

Definition at line 42 of file LaneChangeEvent.h.

Referenced by getDirPos(), and LaneChangeEvent().

4.43.4.7 double LaneChangeEvent::m_EventTime [private]

The time at which the event occured.

Definition at line 44 of file LaneChangeEvent.h.

Referenced by getTime(), and LaneChangeEvent().

The documentation for this class was generated from the following files:

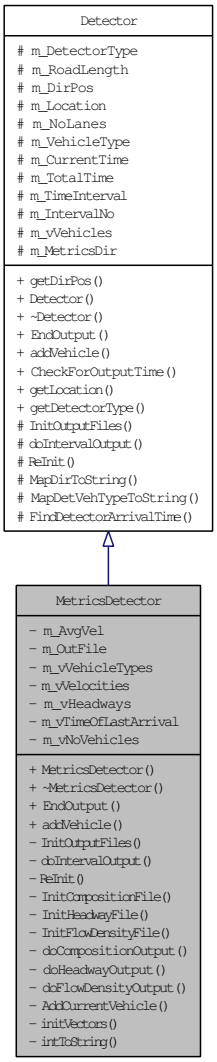
- D:/~Research/Code/C++/EvolveTraffic/LaneChangeEvent.h
- D:/~Research/Code/C++/EvolveTraffic/LaneChangeEvent.cpp

4.44 MetricsDetector Class Reference

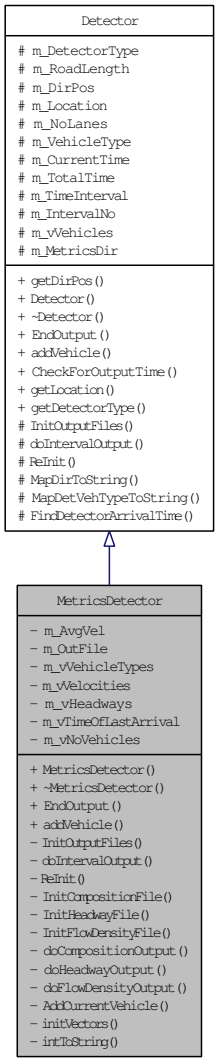
A derived class to represent a detector that tracks metrics information.

`#include <MetricsDetector.h>`

Inheritance diagram for MetricsDetector:



Collaboration diagram for MetricsDetector:



Public Member Functions

- MetricsDetector (std::string dir, WORD DetType, bool DirPos, int nLanes, WORD VehType, int timeInterval, int loc, int RoadLength)

  Constructor.

- virtual ~MetricsDetector ()

  Destructor.

- virtual void EndOutput ()

  Close the output file.

- virtual void addVehicle (Vehicle *pVeh, double curTime)
  Check if we are to add the current vehicle to the detector.

Private Member Functions

- virtual void InitOutputFiles ()
  Initialise the output files.
- virtual void doIntervalOutput ()
  Do the output at the end of each time interval.
- virtual void ReInit ()
  At the end of each time interval reset vectors for next time period.
- void InitCompositionFile ()
  Initialise the Composition output file.
- void InitHeadwayFile ()
  Initialise the Headway output file.
- void InitFlowDensityFile ()
  Initialise the Flow-Density output file.
- void doCompositionOutput ()
  Do the Composition output.
- void doHeadwayOutput ()
  Do the Headway output.
- void doFlowDensityOutput ()
  Do the Flow-Density output.
- void AddCurrentVehicle (Vehicle *pVeh, const double curTime)
  Add the current vehicle to the detector.
- void initVectors ()
  Initialise storage vectors.
- std::string intToString (int d)
  Change an int to a string object.

Private Attributes

- double m_AvgVel
- std::ofstream m_OutFile
- std::vector< WORD > m_vVehicleTypes
- M2Ddbl m_vVelocities
- M2Ddbl m_vHeadways
- std::vector< double > m_vTimeOfLastArrival
- std::vector< int > m_vNoVehicles

4.44.1 Detailed Description

A derived class to represent a detector that tracks metrics information.

Definition at line 19 of file MetricsDetector.h.

4.44.2 Constructor & Destructor Documentation

4.44.2.1 MetricsDetector::MetricsDetector (std::string *dir*, WORD *DetType*, bool *DirPos*, int *nLanes*, WORD *VehType*, int *timeInterval*, int *loc*, int *RoadLength*)

Constructor.

Parameters:

dir The detector output directory

DetType The particular type of detector

DirPos Whether the detector is in the positive direction

nLanes The number of lanes the detector covers

VehType The type of vehicles to track

timeInterval The time interval to use

loc The location of the detector

RoadLength The length of the road

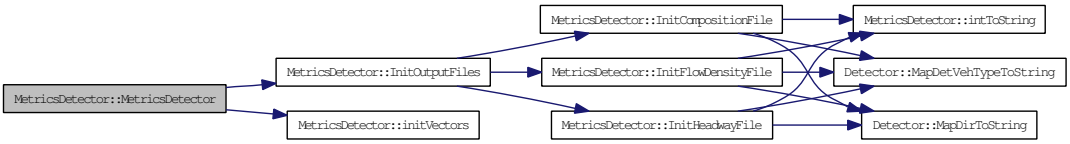
Definition at line 30 of file MetricsDetector.cpp.

References InitOutputFiles(), initVectors(), Detector::m_CurrentTime, Detector::m_DetectorType, Detector::m_DirPos, Detector::m_IntervalNo, Detector::m_Location, Detector::m_MetricsDir, Detector::m_NoLanes, Detector::m_RoadLength, Detector::m_TimeInterval, Detector::m_VehicleType, and m_vTimeOfLastArrival.

```
32 {
33         m_MetricsDir = dir;
34         m_DetectorType = DetType;
35         m_DirPos = DirPos;
36         m_NoLanes = nLanes;
37         m_VehicleType = VehType;
38         m_TimeInterval = timeInterval;
```

```
39          m_Location = loc;
40          m_RoadLength = RoadLength;
41
42          m_CurrentTime = 0;
43          m_IntervalNo = 0;
44
45          m_vTimeOfLastArrival.assign(m_NoLanes,0.0);     // vector of last arrival times by lane
46          initVectors();
47          InitOutputFiles();
48  }
```

Here is the call graph for this function:



4.44.2.2 MetricsDetector::~MetricsDetector () [virtual]

Destructor.

Definition at line 51 of file MetricsDetector.cpp.

```
52 {
53
54 }
```

4.44.3 Member Function Documentation

4.44.3.1 void MetricsDetector::EndOutput () [virtual]

Close the output file.

Reimplemented from Detector.

Definition at line 367 of file MetricsDetector.cpp.

References m_OutFile.

```
368 {
369          m_OutFile.close();
370 }
```

4.44.3.2 void MetricsDetector::addVehicle (Vehicle * *pVeh*, double *curTime*) [virtual]

Check if we are to add the current vehicle to the detector.

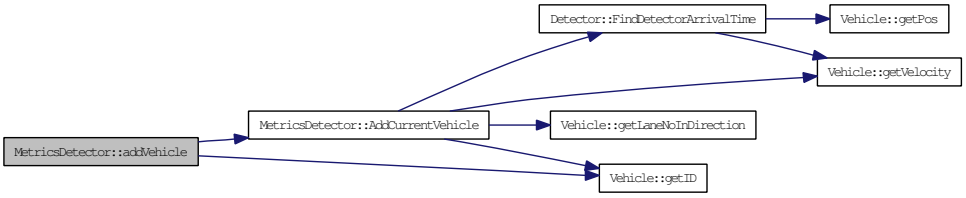
Reimplemented from Detector.

Definition at line 77 of file MetricsDetector.cpp.

References AddCurrentVehicle(), Vehicle::getID(), Detector::m_VehicleType, METRICS_VEH_ALL, METRICS_VEH_CAR, METRICS_VEH_CRANE, METRICS_VEH_LARGETRUCK, METRICS_VEH_LOWLOADER, METRICS_-VEH_SMALLTRUCK, VEH_ID_CAR, VEH_ID_CRANE, VEH_ID_-LARGETRUCK, VEH_ID_LOWLOADER, and VEH_ID_SMALLTRUCK.

```
78 {
79         WORD vehID = pVeh->getID();
80
81         switch(m_VehicleType)
82         {
83                 case METRICS_VEH_ALL:
84                         AddCurrentVehicle(pVeh, curTime);
85                         break;
86                 case METRICS_VEH_CAR:
87                         if(vehID == VEH_ID_CAR)
88                                 AddCurrentVehicle(pVeh, curTime);
89                         break;
90                 case METRICS_VEH_SMALLTRUCK:
91                         if(vehID == VEH_ID_SMALLTRUCK)
92                                 AddCurrentVehicle(pVeh, curTime);
93                         break;
94                 case METRICS_VEH_LARGETRUCK:
95                         if(vehID == VEH_ID_LARGETRUCK)
96                                 AddCurrentVehicle(pVeh, curTime);
97                         break;
98                 case METRICS_VEH_CRANE:
99                         if(vehID == VEH_ID_CRANE)
100                                 AddCurrentVehicle(pVeh, curTime);
101                         break;
102                  case METRICS_VEH_LOWLOADER:
103                         if(vehID == VEH_ID_LOWLOADER)
104                                 AddCurrentVehicle(pVeh, curTime);
105                         break;
106                  default:
107                         AddCurrentVehicle(pVeh, curTime);
108          }
109
110 }
```

Here is the call graph for this function:



4.44.3.3 void MetricsDetector::InitOutputFiles () [private, virtual]

Initialise the output files.

Reimplemented from Detector.

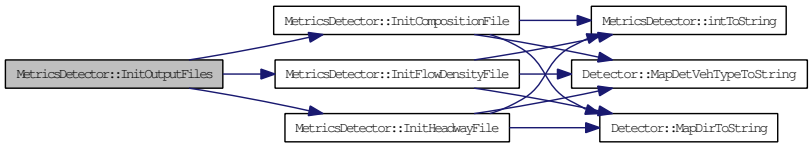
Definition at line 172 of file MetricsDetector.cpp.

References InitCompositionFile(), InitFlowDensityFile(), InitHeadwayFile(), Detector::m_DetectorType, Detector::m_VehicleType, METRICS_TYPE_-COMPOSITION, METRICS_TYPE_FLOWDENSITY, METRICS_TYPE_-HEADWAY, and METRICS_VEH_ALL.

Referenced by MetricsDetector().

```
173 {
174         switch(m_DetectorType)
175         {
176                 case METRICS_TYPE_FLOWDENSITY:
177                         InitFlowDensityFile();
178                         break;
179                 case METRICS_TYPE_HEADWAY:
180                         InitHeadwayFile();
181                         break;
182                 case METRICS_TYPE_COMPOSITION:
183                         if(m_VehicleType == METRICS_VEH_ALL)    // since composition only makes
184                                 InitCompositionFile();
185                         break;
186                 default:
187                         InitFlowDensityFile();
188         }
189 }
```

Here is the call graph for this function:



4.44.3.4 void MetricsDetector::doIntervalOutput () [private, virtual]

Do the output at the end of each time interval.

Reimplemented from Detector.

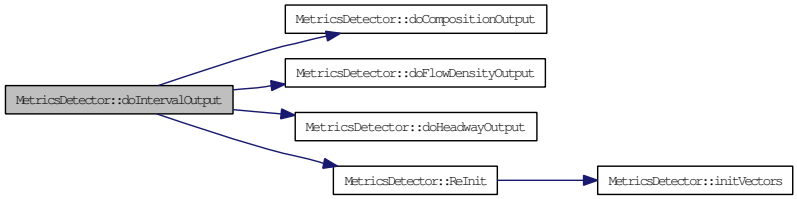
Definition at line 56 of file MetricsDetector.cpp.

References doCompositionOutput(), doFlowDensityOutput(), doHeadwayOut-put(), Detector::m_DetectorType, Detector::m_VehicleType, METRICS_TYPE_-COMPOSITION, METRICS_TYPE_FLOWDENSITY, METRICS_TYPE_-HEADWAY, METRICS_VEH_ALL, and ReInit().

```
57 {
58         switch(m_DetectorType)
59         {
60                 case METRICS_TYPE_FLOWDENSITY:
61                         doFlowDensityOutput();
62                         break;
```

```
63                  case METRICS_TYPE_HEADWAY:
64                          doHeadwayOutput();
65                          break;
66                  case METRICS_TYPE_COMPOSITION:
67                          if(m_VehicleType == METRICS_VEH_ALL)
68                                  doCompositionOutput();  // since composition only makes sense wi
69                          break;
70                  default:
71                          doFlowDensityOutput();
72          }
73
74          ReInit();
75  }
```

Here is the call graph for this function:



4.44.3.5 void MetricsDetector::ReInit () [private, virtual]

At the end of each time interval reset vectors for next time period.

Reimplemented from Detector.

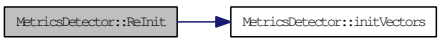
Definition at line 147 of file MetricsDetector.cpp.

References initVectors(), Detector::m_NoLanes, m_vHeadways, m_vNoVehicles, m_-vVehicleTypes, and m_vVelocities.

Referenced by doIntervalOutput().

```
148 {
149          m_vVehicleTypes.clear();
150          m_vNoVehicles.clear();
151          for(int i = 0; i < m_NoLanes; i++)
152          {
153                  m_vVelocities[i].clear();
154                  m_vHeadways[i].clear();
155          }
156          m_vVelocities.clear();
157          m_vHeadways.clear();
158
159          initVectors();
160  }
```

Here is the call graph for this function:



4.44.3.6 void MetricsDetector::InitCompositionFile () [private]

Initialise the Composition output file.

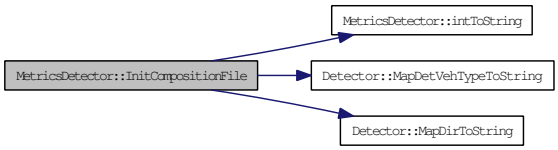
Definition at line 239 of file MetricsDetector.cpp.

References intToString(), Detector::m_DirPos, Detector::m_Location, Detector::m_-MetricsDir, m_OutFile, Detector::m_RoadLength, Detector::m_VehicleType, Detector::MapDetVehTypeToString(), and Detector::MapDirToString().

Referenced by InitOutputFiles().

```
240 {
241         // Create file name and open it
242         int loc = m_DirPos == true ? m_Location : m_RoadLength - m_Location;
243         std::string file;
244         file = m_MetricsDir;
245         file += "Comp_" + MapDetVehTypeToString(m_VehicleType) + "_"
246                                 + MapDirToString(m_DirPos) +"_"
247                                 + intToString(loc) + ".csv";
248         m_OutFile.open(file.c_str(), std::ios::out);
249
250         // let's set some headers
251         m_OutFile << "No. Cars"                 << ",";
252         m_OutFile << "No. Small Trucks" << ",";
253         m_OutFile << "No. Large Trucks" << ",";
254         m_OutFile << "No. Cranes"               << ",";
255         m_OutFile << "No. Low-Loaders"  << '\n';
256 }
```

Here is the call graph for this function:



4.44.3.7 void MetricsDetector::InitHeadwayFile () [private]

Initialise the Headway output file.

Definition at line 220 of file MetricsDetector.cpp.

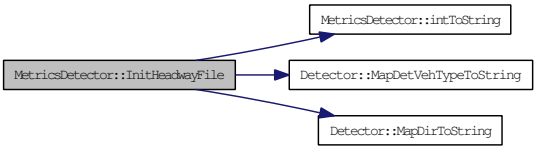
References intToString(), Detector::m_DirPos, Detector::m_Location, Detector::m_-MetricsDir, Detector::m_NoLanes, m_OutFile, Detector::m_RoadLength, Detector::m_VehicleType, Detector::MapDetVehTypeToString(), and Detector::MapDirToString().

Referenced by InitOutputFiles().

```
221 {
222         // Create file name and open it
223         int loc = m_DirPos == true ? m_Location : m_RoadLength - m_Location;
224         std::string file;
```

```
225         file = m_MetricsDir;
226         file += "Head_" + MapDetVehTypeToString(m_VehicleType) + "_"
227                                 + MapDirToString(m_DirPos) +"_"
228                                 + intToString(loc) + ".csv";
229         m_OutFile.open(file.c_str(), std::ios::out);
230
231         // let's set some headers
232         m_OutFile << "Headways (s)"     << '\n';
233         for(int i = 1; i < m_NoLanes; i++)
234                 m_OutFile << "Lane " <<  intToString(i) << ',';
235         m_OutFile << "Lane " <<  intToString(m_NoLanes) << '\n';
236 }
```

Here is the call graph for this function:



4.44.3.8 void MetricsDetector::InitFlowDensityFile () [private]

Initialise the Flow-Density output file.

Definition at line 192 of file MetricsDetector.cpp.

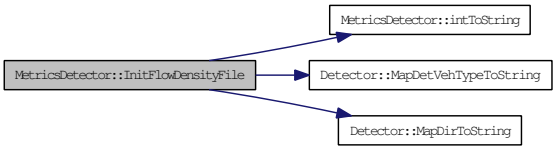
References intToString(), Detector::m_DirPos, Detector::m_Location, Detector::m_-MetricsDir, Detector::m_NoLanes, m_OutFile, Detector::m_RoadLength, Detector::m_VehicleType, Detector::MapDetVehTypeToString(), and Detector::MapDirToString().

Referenced by InitOutputFiles().

```
193 {
194         // Create file name and open it
195         int loc = m_DirPos == true ? m_Location : m_RoadLength - m_Location;
196         std::string file;
197         file = m_MetricsDir;
198         file += "FD_" + MapDetVehTypeToString(m_VehicleType) + "_"
199                                 + MapDirToString(m_DirPos) +"_"
200                                 + intToString(loc) + ".csv";
201         m_OutFile.open(file.c_str(), std::ios::out);
202
203         // let's set some headers
204         m_OutFile << " " << ',';        //Interval no column
205         for(int i = 1; i <= m_NoLanes; i++)
206                 m_OutFile << "Lane " <<  intToString(i) << ", , ,";
207         m_OutFile << "Totals" << '\n';
208
209         m_OutFile << "Interval No."             << ',';
210         for(i = 0; i < m_NoLanes+1; i++)        // + 1 is for the totals columns
211         {
212                 char end = i == m_NoLanes ? '\n' : ',';
213                 m_OutFile << "Density (veh/km)" << ',';
214                 m_OutFile << "Flow (veh/hr)"    << ',';
```

```
215                 m_OutFile << "Avg. Vel. (km/h)" << end;
216         }
217 }
```

Here is the call graph for this function:



4.44.3.9 void MetricsDetector::doCompositionOutput () [private]

Do the Composition output.

Definition at line 326 of file MetricsDetector.cpp.

References m_OutFile, m_vVehicleTypes, VEH_ID_CAR, VEH_ID_CRANE, VEH_-ID_LARGETRUCK, VEH_ID_LOWLOADER, and VEH_ID_SMALLTRUCK.

Referenced by doIntervalOutput().

```
327 {
328         int nCar = 0;
329         int nSmallTruck = 0;
330         int nLargeTruck = 0;
331         int nCrane = 0;
332         int nLowLoader = 0;
333
334         for(int i = 0; i < m_vVehicleTypes.size(); i++)
335         {
336                 WORD VehType = m_vVehicleTypes.at(i);
337
338                 switch(VehType)
339                 {
340                         case VEH_ID_CAR:                nCar++;                 break;
341                         case VEH_ID_SMALLTRUCK: nSmallTruck++;  break;
342                         case VEH_ID_LARGETRUCK: nLargeTruck++;  break;
343                         case VEH_ID_CRANE:              nCrane++;               break;
344                         case VEH_ID_LOWLOADER:  nLowLoader++;   break;
345                         default: nCar++;
346                 }
347         }
348
349         m_OutFile << nCar                       << ",";
350         m_OutFile << nSmallTruck        << ",";
351         m_OutFile << nLargeTruck        << ",";
352         m_OutFile << nCrane                     << ",";
353         m_OutFile << nLowLoader         << '\n';
354 }
```

4.44.3.10 void MetricsDetector::doHeadwayOutput () [private]

Do the Headway output.

Definition at line 300 of file MetricsDetector.cpp.

References Detector::m_NoLanes, m_OutFile, and m_vHeadways.

Referenced by doIntervalOutput().

```
301 {
302         // get size of largest headway values vector
303         int jmax = m_vHeadways[0].size();
304         for(int i = 1; i < m_NoLanes; i++)
305         {
306                 if(m_vHeadways[i].size() > jmax)
307                         jmax = m_vHeadways[i].size();
308         }
309
310         // write across then down, leaving blanks where no value
311         for(int j = 0; j < jmax; j++)
312         {
313                 for(i = 0; i < m_NoLanes; i++)
314                 {
315                         char end = i == m_NoLanes-1 ? '\n' : ',';
316
317                         if(j < m_vHeadways[i].size())
318                                 m_OutFile << m_vHeadways[i].at(j) << end;
319                         else
320                                 m_OutFile << " " << end;
321                 }
322         }
323 }
```

4.44.3.11 void MetricsDetector::doFlowDensityOutput () [private]

Do the Flow-Density output.

Definition at line 259 of file MetricsDetector.cpp.

References Detector::m_IntervalNo, Detector::m_NoLanes, m_OutFile, M_PER_S_TO_KM_PER_H, Detector::m_TimeInterval, m_vNoVehicles, m_vVelocities, and SECS_PER_HOUR.

Referenced by doIntervalOutput().

```
260 {
261         // first find the density, flow and ave velocity
262         std::vector<double> vAvgVel(m_NoLanes,0.0);
263         std::vector<double> vFlowPerHour(m_NoLanes,0.0);
264         std::vector<double> vDensity(m_NoLanes,0.0);
265
266         for(int i = 0; i < m_NoLanes; i++)
267         {
268                 for(int j = 0; j < m_vNoVehicles[i]; j++)
269                         vAvgVel[i] += m_vVelocities[i].at(j);
270                 // since we have the sum of velocites in lane i now
271                 // we can get the average now, in km/h
272                 vAvgVel[i] = M_PER_S_TO_KM_PER_H * vAvgVel[i] / m_vNoVehicles[i];
273                 vFlowPerHour[i] = m_vNoVehicles[i] * (SECS_PER_HOUR/m_TimeInterval);
274                 vDensity[i] = vFlowPerHour[i]/vAvgVel[i];       // units of veh/km
275         }
```

```
276
277         // lastly, do the sum of lanes for total direction properties
278         double totAvgVel = 0.0; double totFlowPerHour = 0.0; double totDensity = 0.0;
279         for(i = 0; i < m_NoLanes; i++)
280         {
281                 totAvgVel               += vAvgVel[i];
282                 totFlowPerHour  += vFlowPerHour[i];
283                 totDensity              += vDensity[i];
284         }
285
286         // then do the output and reset the variables
287         m_OutFile << m_IntervalNo       << ',';
288         for(i = 0; i < m_NoLanes; i++)
289         {
290                 m_OutFile << vDensity[i]                << ',';
291                 m_OutFile << vFlowPerHour[i]    << ',';
292                 m_OutFile << vAvgVel[i]                 << ',';
293         }
294         m_OutFile << totDensity         << ',';
295         m_OutFile << totFlowPerHour     << ',';
296         m_OutFile << totAvgVel          << '\n';
297 }
```

4.44.3.12 void MetricsDetector::AddCurrentVehicle (Vehicle * *pVeh*, const double *curTime*) [private]

Add the current vehicle to the detector.

Definition at line 116 of file MetricsDetector.cpp.

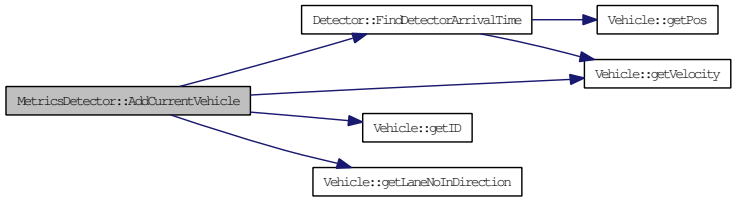
References Detector::FindDetectorArrivalTime(), Vehicle::getID(), Vehicle::getLaneNoInDirection(), Vehicle::getVelocity(), Detector::m_DetectorType, Detector::m_VehicleType, m_vHeadways, m_vNoVehicles, m_vTimeOfLastArrival, m_vVehicleTypes, m_vVelocities, METRICS_TYPE_COMPOSITION, METRICS_TYPE_FLOWDENSITY, METRICS_TYPE_HEADWAY, and METRICS_VEH_ALL.

Referenced by addVehicle().

```
117 {
118         int iLane = pVeh->getLaneNoInDirection();       // get the local lane number in its dir
119         switch(m_DetectorType)
120         {
121                 case METRICS_TYPE_FLOWDENSITY:
122                 {
123                         m_vNoVehicles[iLane-1]++;
124                         m_vVelocities[iLane-1].push_back(pVeh->getVelocity());          // and
125                         break;
126                 }
127                 case METRICS_TYPE_HEADWAY:
128                 {
129                         double ArrivalTime = FindDetectorArrivalTime(pVeh, curTime);
130                         double headway = ArrivalTime - m_vTimeOfLastArrival[iLane-1];
131                         m_vHeadways[iLane-1].push_back(headway);
132                         m_vTimeOfLastArrival[iLane-1] = ArrivalTime;    // update last arrival
133                         break;
134                 }
135                 case METRICS_TYPE_COMPOSITION:
136                 {
```

```
137                         // since composition only makes sense with all vehicles
138                         if(m_VehicleType == METRICS_VEH_ALL)
139                                 m_vVehicleTypes.push_back( pVeh->getID() );
140                         break;
141                 }
142         }
143
144 }
```

Here is the call graph for this function:



4.44.3.13 void MetricsDetector::initVectors () [private]

Initialise storage vectors.

Definition at line 163 of file MetricsDetector.cpp.

References Detector::m_NoLanes, m_vHeadways, m_vNoVehicles, and m_vVelocities.

Referenced by MetricsDetector(), and ReInit().

```
164 {
165         m_vNoVehicles.assign(m_NoLanes,0);                      // no of vehicles by lane
166         std::vector<double> temp;
167         m_vHeadways.assign(m_NoLanes, temp);
168         m_vVelocities.assign(m_NoLanes, temp);
169 }
```

4.44.3.14 std::string MetricsDetector::intToString (int *d*) [private]

Change an int to a string object.

Definition at line 357 of file MetricsDetector.cpp.

Referenced by InitCompositionFile(), InitFlowDensityFile(), and InitHeadwayFile().

```
358 {
359         std::string a;
360         char buffer; char* pBuffer = &buffer;
361         itoa( d, pBuffer, 10 );
362         a.append(&buffer);
363         return a;
364 }
```

4.44.4 Member Data Documentation

4.44.4.1 double MetricsDetector::m_AvgVel [private]

Definition at line 46 of file MetricsDetector.h.

4.44.4.2 std::ofstream MetricsDetector::m_OutFile [private]

Definition at line 47 of file MetricsDetector.h.

Referenced by doCompositionOutput(), doFlowDensityOutput(), doHeadwayOutput(), EndOutput(), InitCompositionFile(), InitFlowDensityFile(), and InitHeadwayFile().

4.44.4.3 std::vector<WORD> MetricsDetector::m_vVehicleTypes [private]

Definition at line 48 of file MetricsDetector.h.

Referenced by AddCurrentVehicle(), doCompositionOutput(), and ReInit().

4.44.4.4 M2Ddbl MetricsDetector::m_vVelocities [private]

Definition at line 49 of file MetricsDetector.h.

Referenced by AddCurrentVehicle(), doFlowDensityOutput(), initVectors(), and ReInit().

4.44.4.5 M2Ddbl MetricsDetector::m_vHeadways [private]

Definition at line 50 of file MetricsDetector.h.

Referenced by AddCurrentVehicle(), doHeadwayOutput(), initVectors(), and ReInit().

4.44.4.6 std::vector<double> MetricsDetector::m_vTimeOfLastArrival [private]

Definition at line 51 of file MetricsDetector.h.

Referenced by AddCurrentVehicle(), and MetricsDetector().

4.44.4.7 std::vector<int> MetricsDetector::m_vNoVehicles [private]

Definition at line 52 of file MetricsDetector.h.

Referenced by AddCurrentVehicle(), doFlowDensityOutput(), initVectors(), and ReInit().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/MetricsDetector.h
- D:/~Research/Code/C++/EvolveTraffic/MetricsDetector.cpp

4.45 MTRand Class Reference

A class for generating random numbers - Mersenne Twister.

`#include <MersenneTwister.h>`

Public Types

- enum { N = 624 }
- enum { SAVE = N + 1 }
- typedef unsigned long uint32

Public Member Functions

- MTRand (const uint32 &oneSeed)
- MTRand (uint32 *const bigSeed, uint32 const seedLength=N)
- MTRand ()
- double rand ()
- double rand (const double &n)
- double randExc ()
- double randExc (const double &n)
- double randDblExc ()
- double randDblExc (const double &n)
- uint32 randInt ()
- uint32 randInt (const uint32 &n)
- double operator() ()
- double rand53 ()
- double randNorm (const double &mean=0.0, const double &variance=0.0)
- void seed (const uint32 oneSeed)
- void seed (uint32 *const bigSeed, const uint32 seedLength=N)
- void seed ()
- void save (uint32 *saveArray) const
- void load (uint32 *const loadArray)

Protected Types

- enum { M = 397 }

Protected Member Functions

- void initialize (const uint32 oneSeed)
- void reload ()
- uint32 hiBit (const uint32 &u) const
- uint32 loBit (const uint32 &u) const
- uint32 loBits (const uint32 &u) const
- uint32 mixBits (const uint32 &u, const uint32 &v) const
- uint32 twist (const uint32 &m, const uint32 &s0, const uint32 &s1) const

Static Protected Member Functions

- static uint32 hash (time_t t, clock_t c)

Protected Attributes

- uint32 state [N]
- uint32 * pNext
- int left

4.45.1 Detailed Description

A class for generating random numbers - Mersenne Twister.

Definition at line 71 of file MersenneTwister.h.

4.45.2 Member Typedef Documentation

4.45.2.1 typedef unsigned long MTRand::uint32

Definition at line 74 of file MersenneTwister.h.

4.45.3 Member Enumeration Documentation

4.45.3.1 anonymous enum

Enumerator:

N

Definition at line 76 of file MersenneTwister.h.

```
76 { N = 624 };       // length of state vector
```

4.45.3.2 anonymous enum

Enumerator:

SAVE

Definition at line 77 of file MersenneTwister.h.

```
77 { SAVE = N + 1 };  // length of array for save()
```

4.45.3.3 anonymous enum [protected]

Enumerator:

M

Definition at line 80 of file MersenneTwister.h.

```
80 { M = 397 };  // period parameter
```

4.45.4 Constructor & Destructor Documentation

4.45.4.1 MTRand::MTRand (const uint32 & *oneSeed*) [inline]

Definition at line 139 of file MersenneTwister.h.

References seed().

```
140         { seed(oneSeed); }
```

Here is the call graph for this function:

4.45.4.2 MTRand::MTRand (uint32 *const *bigSeed*, uint32 const *seedLength* = N) [inline]

Definition at line 142 of file MersenneTwister.h.

References seed().

```
143         { seed(bigSeed,seedLength); }
```

Here is the call graph for this function:

4.45.4.3 MTRand::MTRand () [inline]

Definition at line 145 of file MersenneTwister.h.

References seed().

```
146         { seed(); }
```

Here is the call graph for this function:

4.45.5 Member Function Documentation

4.45.5.1 double MTRand::rand () [inline]

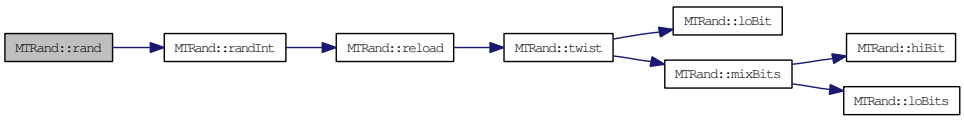
Definition at line 148 of file MersenneTwister.h.

References randInt().

Referenced by CDistribution::BoxMuller(), CDistribution::GenerateExponential(), CDistribution::GenerateGamma(), CDistribution::GenerateGEV(), CDistribution::GenerateGumbel(), CDistribution::GenerateLogNormal(), CDistribution::GeneratePoisson(), operator()(), and rand().

```
149         { return double(randInt()) * (1.0/4294967295.0); }
```

Here is the call graph for this function:



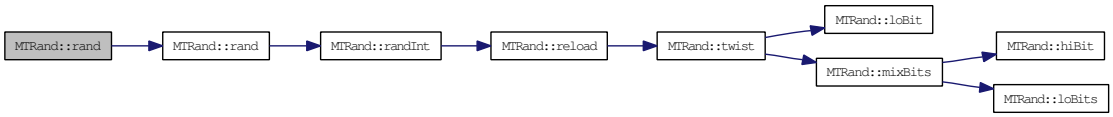
4.45.5.2 double MTRand::rand (const double & *n*) [inline]

Definition at line 151 of file MersenneTwister.h.

References rand().

```
152         { return rand() * n; }
```

Here is the call graph for this function:



4.45.5.3 double MTRand::randExc () [inline]

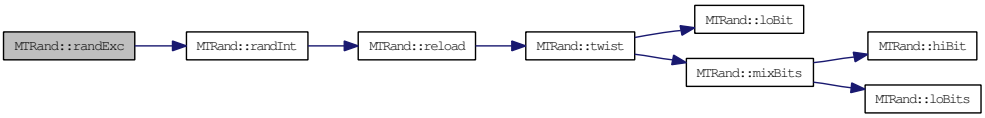
Definition at line 154 of file MersenneTwister.h.

References randInt().

Referenced by randExc(), and randNorm().

```
155         { return double(randInt()) * (1.0/4294967296.0); }
```

Here is the call graph for this function:



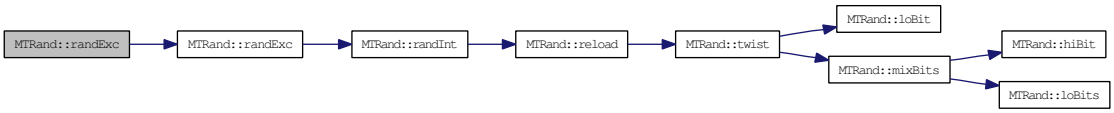
4.45.5.4 double MTRand::randExc (const double & *n*) [inline]

Definition at line 157 of file MersenneTwister.h.

References randExc().

```
158         { return randExc() * n; }
```

Here is the call graph for this function:



4.45.5.5 double MTRand::randDblExc () [inline]

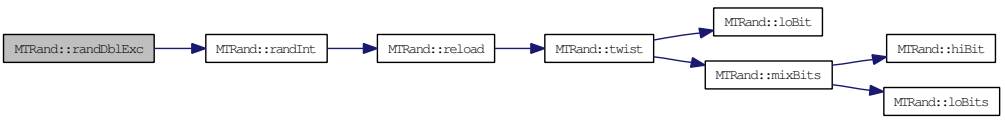
Definition at line 160 of file MersenneTwister.h.

References randInt().

Referenced by randDblExc(), and randNorm().

```
161         { return ( double(randInt()) + 0.5 ) * (1.0/4294967296.0); }
```

Here is the call graph for this function:



4.45.5.6 double MTRand::randDblExc (const double & *n*) [inline]

Definition at line 163 of file MersenneTwister.h.

References randDblExc().

```
164         { return randDblExc() * n; }
```

Here is the call graph for this function:

4.45.5.7 MTRand::uint32 MTRand::randInt () [inline]

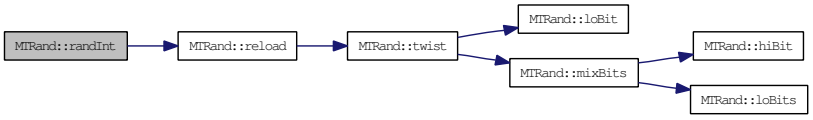
Definition at line 181 of file MersenneTwister.h.

References left, pNext, and reload().

Referenced by rand(), rand53(), randDblExc(), randExc(), and randInt().

```
182 {
183         // Pull a 32-bit integer from the generator state
184         // Every other access function simply transforms the numbers extracted here
185
186         if( left == 0 ) reload();
187         --left;
188
189         register uint32 s1;
190         s1 = *pNext++;
191         s1 ^= (s1 >> 11);
192         s1 ^= (s1 <<  7) & 0x9d2c5680UL;
193         s1 ^= (s1 << 15) & 0xefc60000UL;
194         return ( s1 ^ (s1 >> 18) );
195 }
```

Here is the call graph for this function:



4.45.5.8 MTRand::uint32 MTRand::randInt (const uint32 & *n*) [inline]

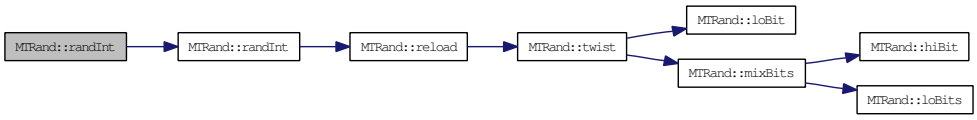
Definition at line 197 of file MersenneTwister.h.

References randInt().

```
198 {
199         // Find which bits are used in n
200         // Optimized by Magnus Jonsson (magnus@smartelectronix.com)
201         uint32 used = n;
202         used |= used >> 1;
203         used |= used >> 2;
204         used |= used >> 4;
205         used |= used >> 8;
206         used |= used >> 16;
```

```
207
208         // Draw numbers until one is found in [0,n]
209         uint32 i;
210         do
211                 i = randInt() & used;  // toss unused bits to shorten search
212         while( i > n );
213         return i;
214 }
```

Here is the call graph for this function:



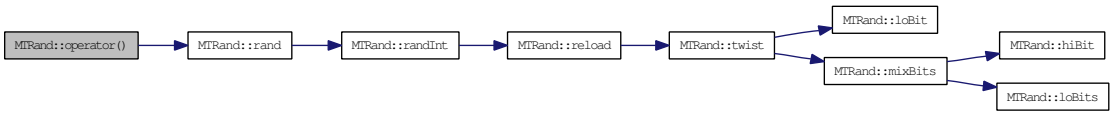
4.45.5.9 double MTRand::operator() () [inline]

Definition at line 106 of file MersenneTwister.h.

References rand().

```
106 { return rand(); }  // same as rand()
```

Here is the call graph for this function:



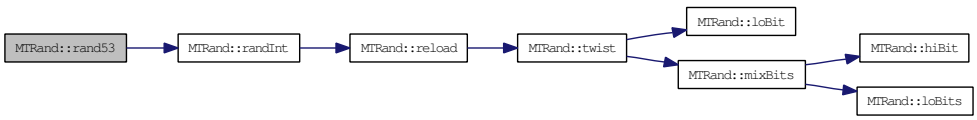
4.45.5.10 double MTRand::rand53 () [inline]

Definition at line 166 of file MersenneTwister.h.

References randInt().

```
167 {
168         uint32 a = randInt() >> 5, b = randInt() >> 6;
169         return ( a * 67108864.0 + b ) * (1.0/9007199254740992.0);  // by Isaku Wada
170 }
```

Here is the call graph for this function:



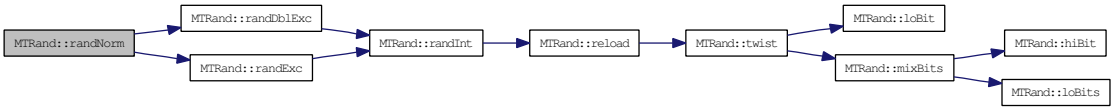
4.45.5.11 double MTRand::randNorm (const double & *mean* = 0.0, const double & *variance* = 0.0) [inline]

Definition at line 172 of file MersenneTwister.h.

References randDblExc(), and randExc().

```
173 {
174         // Return a real number from a normal (Gaussian) distribution with given
175         // mean and variance by Box-Muller method
176         double r = sqrt( -2.0 * log( 1.0-randDblExc()) ) * variance;
177         double phi = 2.0 * 3.14159265358979323846264338328 * randExc();
178         return mean + r * cos(phi);
179 }
```

Here is the call graph for this function:



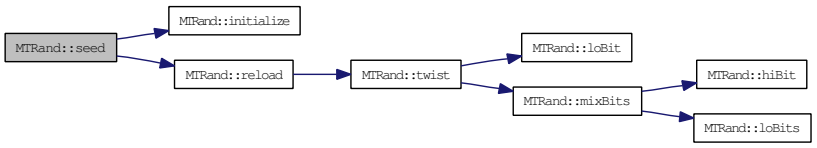
4.45.5.12 void MTRand::seed (const uint32 *oneSeed*) [inline]

Definition at line 217 of file MersenneTwister.h.

References initialize(), and reload().

```
218 {
219         // Seed the generator with a simple uint32
220         initialize(oneSeed);
221         reload();
222 }
```

Here is the call graph for this function:



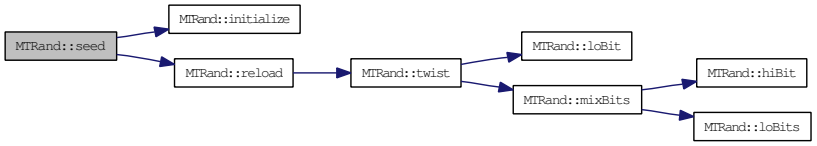
4.45.5.13 void MTRand::seed (uint32 *const *bigSeed*, const uint32 *seedLength* = N) [inline]

Definition at line 225 of file MersenneTwister.h.

References initialize(), N, reload(), and state.

```
226 {
227         // Seed the generator with an array of uint32's
228         // There are 2^19937-1 possible initial states.  This function allows
229         // all of those to be accessed by providing at least 19937 bits (with a
230         // default seed length of N = 624 uint32's).  Any bits above the lower 32
231         // in each element are discarded.
232         // Just call seed() if you want to get array from /dev/urandom
233         initialize(19650218UL);
234         register int i = 1;
235         register uint32 j = 0;
236         register int k = ( N > seedLength ? N : seedLength );
237         for( ; k; --k )
238         {
239                 state[i] =
240                         state[i] ^ ( (state[i-1] ^ (state[i-1] >> 30)) * 1664525UL );
241                 state[i] += ( bigSeed[j] & 0xffffffffUL ) + j;
242                 state[i] &= 0xffffffffUL;
243                 ++i;  ++j;
244                 if( i >= N ) { state[0] = state[N-1];  i = 1; }
245                 if( j >= seedLength ) j = 0;
246         }
247         for( k = N - 1; k; --k )
248         {
249                 state[i] =
250                         state[i] ^ ( (state[i-1] ^ (state[i-1] >> 30)) * 1566083941UL );
251                 state[i] -= i;
252                 state[i] &= 0xffffffffUL;
253                 ++i;
254                 if( i >= N ) { state[0] = state[N-1];  i = 1; }
255         }
256         state[0] = 0x80000000UL;  // MSB is 1, assuring non-zero initial array
257         reload();
258 }
```

Here is the call graph for this function:



4.45.5.14 void MTRand::seed () [inline]

Definition at line 261 of file MersenneTwister.h.

References hash(), and N.

Referenced by MTRand().

```
262 {
263         // Seed the generator with an array from /dev/urandom if available
264         // Otherwise use a hash of time() and clock() values
265
266         // First try getting an array from /dev/urandom
267         FILE* urandom = fopen( "/dev/urandom", "rb" );
268         if( urandom )
```

```
269         {
270                 uint32 bigSeed[N];
271                 register uint32 *s = bigSeed;
272                 register int i = N;
273                 register bool success = true;
274                 while( success && i-- )
275                         success = fread( s++, sizeof(uint32), 1, urandom ) == 0 ? false: true;
276                 fclose(urandom);
277                 if( success ) { seed( bigSeed, N );  return; }
278         }
279
280         // Was not successful, so use time() and clock() instead
281         seed( hash( time(NULL), clock() ) );
282 }
```

Here is the call graph for this function:

4.45.5.15 void MTRand::save (uint32 * *saveArray*) const [inline]

Definition at line 345 of file MersenneTwister.h.

References left, N, and state.

```
346 {
347         register uint32 *sa = saveArray;
348         register const uint32 *s = state;
349         register int i = N;
350         for( ; i--; *sa++ = *s++ ) {}
351         *sa = left;
352 }
```

4.45.5.16 void MTRand::load (uint32 *const *loadArray*) [inline]

Definition at line 355 of file MersenneTwister.h.

References left, N, pNext, and state.

```
356 {
357         register uint32 *s = state;
358         register uint32 *la = loadArray;
359         register int i = N;
360         for( ; i--; *s++ = *la++ ) {}
361         left = *la;
362         pNext = &state[N-left];
363 }
```

4.45.5.17 void MTRand::initialize (const uint32 *oneSeed*) [inline, protected]

Definition at line 285 of file MersenneTwister.h.

References N, and state.

Referenced by seed().

```
286 {
287         // Initialize generator state with seed
288         // See Knuth TAOCP Vol 2, 3rd Ed, p.106 for multiplier.
289         // In previous versions, most significant bits (MSBs) of the seed affect
290         // only MSBs of the state array.  Modified 9 Jan 2002 by Makoto Matsumoto.
291         register uint32 *s = state;
292         register uint32 *r = state;
293         register int i = 1;
294         *s++ = seed & 0xffffffffUL;
295         for( ; i < N; ++i )
296         {
297                 *s++ = ( 1812433253UL * ( *r ^ (*r >> 30) ) + i ) & 0xffffffffUL;
298                 r++;
299         }
300 }
```

4.45.5.18 void MTRand::reload () [inline, protected]

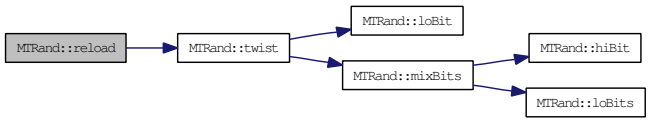
Definition at line 303 of file MersenneTwister.h.

References left, M, N, pNext, state, and twist().

Referenced by randInt(), and seed().

```
304 {
305         // Generate N new values in state
306         // Made clearer and faster by Matthew Bellew (matthew.bellew@home.com)
307         register uint32 *p = state;
308         register int i;
309         for( i = N - M; i--; ++p )
310                 *p = twist( p[M], p[0], p[1] );
311         for( i = M; --i; ++p )
312                 *p = twist( p[M-N], p[0], p[1] );
313         *p = twist( p[M-N], p[0], state[0] );
314
315         left = N, pNext = state;
316 }
```

Here is the call graph for this function:



4.45.5.19 uint32 MTRand::hiBit (const uint32 & *u*) const [inline, protected]

Definition at line 128 of file MersenneTwister.h.

Referenced by mixBits().

```
128 { return u & 0x80000000UL; }
```

4.45.5.20 uint32 MTRand::loBit (const uint32 & *u*) const [inline, protected]

Definition at line 129 of file MersenneTwister.h.

Referenced by twist().

```
129 { return u & 0x00000001UL; }
```

4.45.5.21 uint32 MTRand::loBits (const uint32 & *u*) const [inline, protected]

Definition at line 130 of file MersenneTwister.h.

Referenced by mixBits().

```
130 { return u & 0x7fffffffUL; }
```

4.45.5.22 uint32 MTRand::mixBits (const uint32 & *u*, const uint32 & *v*) const [inline, protected]

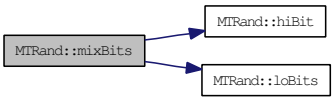
Definition at line 131 of file MersenneTwister.h.

References hiBit(), and loBits().

Referenced by twist().

```
132                 { return hiBit(u) | loBits(v); }
```

Here is the call graph for this function:



4.45.5.23 uint32 MTRand::twist (const uint32 & *m*, const uint32 & *s0*, const uint32 & *s1*) const [inline, protected]

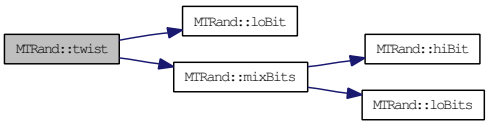
Definition at line 133 of file MersenneTwister.h.

References loBit(), and mixBits().

Referenced by reload().

```
134                 { return m ^ (mixBits(s0,s1)>>1) ^ (-loBit(s1) & 0x9908b0dfUL); }
```

Here is the call graph for this function:



4.45.5.24 MTRand::uint32 MTRand::hash (time_t *t*, clock_t *c*) [inline, static, protected]

Definition at line 319 of file MersenneTwister.h.

Referenced by seed().

```
320 {
321         // Get a uint32 from t and c
322         // Better than uint32(x) in case x is floating point in [0,1]
323         // Based on code by Lawrence Kirby (fred@genesis.demon.co.uk)
324 
325         static uint32 differ = 0;  // guarantee time-based seeds will change
326 
327         uint32 h1 = 0;
328         unsigned char *p = (unsigned char *) &t;
329         for( size_t i = 0; i < sizeof(t); ++i )
330         {
331                 h1 *= UCHAR_MAX + 2U;
332                 h1 += p[i];
333         }
334         uint32 h2 = 0;
335         p = (unsigned char *) &c;
336         for( size_t j = 0; j < sizeof(c); ++j )
337         {
338                 h2 *= UCHAR_MAX + 2U;
339                 h2 += p[j];
340         }
341         return ( h1 + differ++ ) ^ h2;
342 }
```

4.45.6 Member Data Documentation

4.45.6.1 uint32 MTRand::state[N] [protected]

Definition at line 82 of file MersenneTwister.h.

Referenced by initialize(), load(), reload(), save(), and seed().

4.45.6.2 uint32* MTRand::pNext [protected]

Definition at line 83 of file MersenneTwister.h.

Referenced by load(), randInt(), and reload().

4.45.6.3 int MTRand::left [protected]

Definition at line 84 of file MersenneTwister.h.

Referenced by load(), randInt(), reload(), and save().

The documentation for this class was generated from the following file:

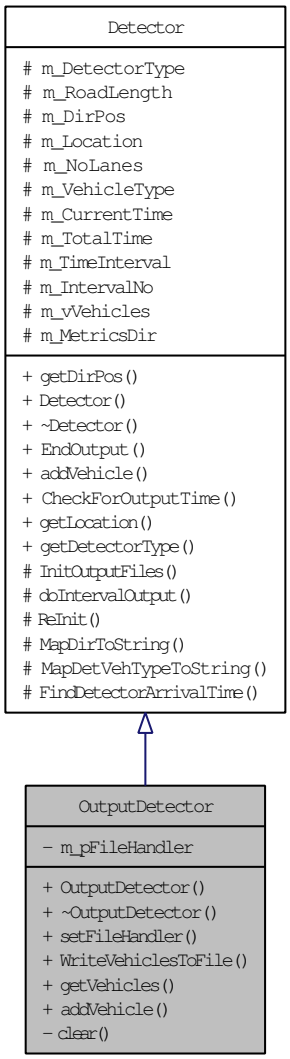
- D:/~Research/Code/C++/EvolveTraffic/MersenneTwister.h

4.46 OutputDetector Class Reference

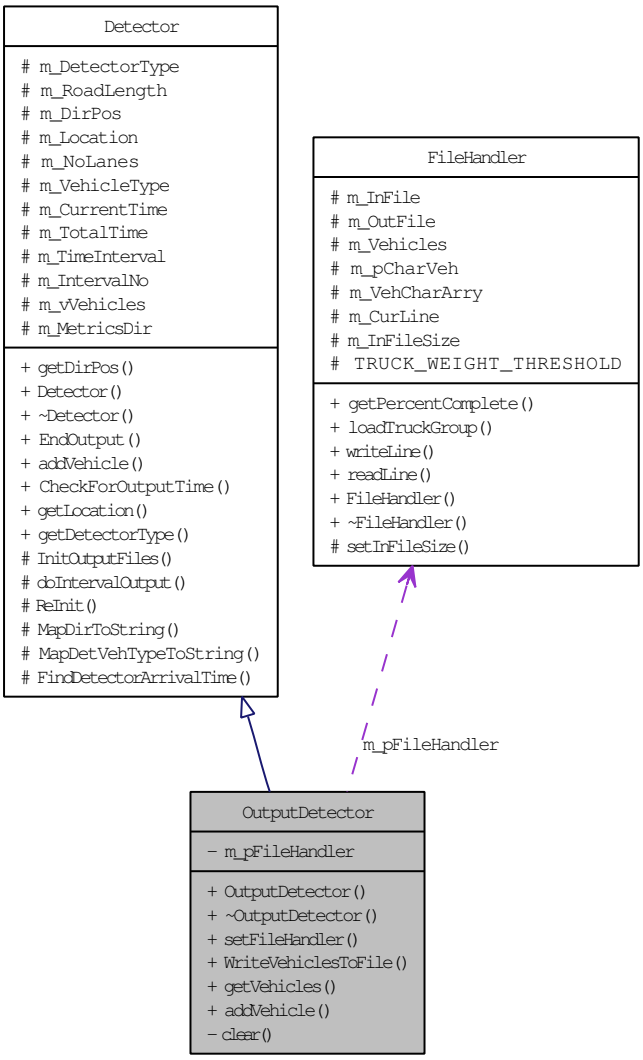
A derived class representing a detector which tracks when vehicles pass a point.

`#include <OutputDetector.h>`

Inheritance diagram for OutputDetector:



Collaboration diagram for OutputDetector:



Public Member Functions

- OutputDetector (int loc, bool DirPos, FileHandler *fh)

  Constructor.

- virtual ~OutputDetector ()

  Default Desctructor.

- void setFileHandler (FileHandler *fh)

  Sets the OutputDetector's filehandler.

- void WriteVehiclesToFile ()

  Writes the OutputDetector's vehicles to file.

- std::vector< Vehicle * > getVehicles ()
- void addVehicle (Vehicle *pVeh, double curTime)

  Adds a vehicle to the OutputDetector.

Private Member Functions

- void clear ()

Private Attributes

- FileHandler * m_pFileHandler

4.46.1 Detailed Description

A derived class representing a detector which tracks when vehicles pass a point.

Definition at line 19 of file OutputDetector.h.

4.46.2 Constructor & Destructor Documentation

4.46.2.1 OutputDetector::OutputDetector (int *loc*, bool *DirPos*, FileHandler * *fh*)

Constructor.

Parameters:

loc The location of the OutputDetector

DirPos Whether or not the OutputDetector is in the positive direction

fh The filehandler to use

Definition at line 24 of file OutputDetector.cpp.

References Detector::m_DirPos, Detector::m_Location, and m_pFileHandler.

```
25 {
26         m_Location = loc;
27         m_DirPos = DirPos;
28         m_pFileHandler = fh;
29 }
```

4.46.2.2 OutputDetector::~OutputDetector () [virtual]

Default Desctructor.

Definition at line 32 of file OutputDetector.cpp.

```
33 {
34
35 }
```

4.46.3 Member Function Documentation

4.46.3.1 void OutputDetector::setFileHandler (FileHandler * *fh*)

Sets the OutputDetector's filehandler.

Parameters:

fh The filehandler to use

Definition at line 64 of file OutputDetector.cpp.

References m_pFileHandler.

```
65 {
66         m_pFileHandler = fh;
67 }
```

4.46.3.2 void OutputDetector::WriteVehiclesToFile ()

Writes the OutputDetector's vehicles to file.

Definition at line 52 of file OutputDetector.cpp.

References m_pFileHandler, Detector::m_vVehicles, and FileHandler::writeLine().

Referenced by Direction::update().

```
53 {
54         for(int i = 0; i < m_vVehicles.size(); i++)
55                 m_pFileHandler->writeLine(m_vVehicles.at(i));
56         // once written, they must be deleted
57         m_vVehicles.clear();
58 }
```

Here is the call graph for this function:

4.46.3.3 std::vector<Vehicle*> OutputDetector::getVehicles ()

4.46.3.4 void OutputDetector::addVehicle (Vehicle * *pVeh*, double *curTime*) [virtual]

Adds a vehicle to the OutputDetector.

Parameters:

pVeh The vehicle to add

curTime The current simulation time

Reimplemented from Detector.

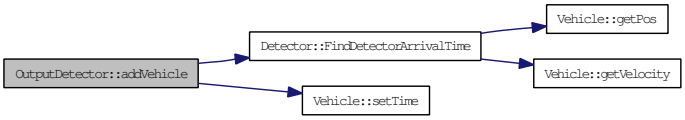
Definition at line 42 of file OutputDetector.cpp.

References Detector::FindDetectorArrivalTime(), Detector::m_vVehicles, and Vehicle::setTime().

Referenced by Lane::CheckDetectors().

```
43 {
44         // set the vehicle time to the arrival time at the detector
45         double ArrivalTime = FindDetectorArrivalTime(pVeh, curTime);
46         pVeh->setTime(ArrivalTime);
47         // now add to vector prior to writing
48         m_vVehicles.push_back(pVeh);
49 }
```

Here is the call graph for this function:



4.46.3.5 void OutputDetector::clear () [private]

Definition at line 69 of file OutputDetector.cpp.

References Detector::m_vVehicles.

```
70 {
71         m_vVehicles.clear();
72 }
```

4.46.4 Member Data Documentation

4.46.4.1 FileHandler* OutputDetector::m_pFileHandler [private]

Definition at line 33 of file OutputDetector.h.

Referenced by OutputDetector(), setFileHandler(), and WriteVehiclesToFile().

The documentation for this class was generated from the following files:

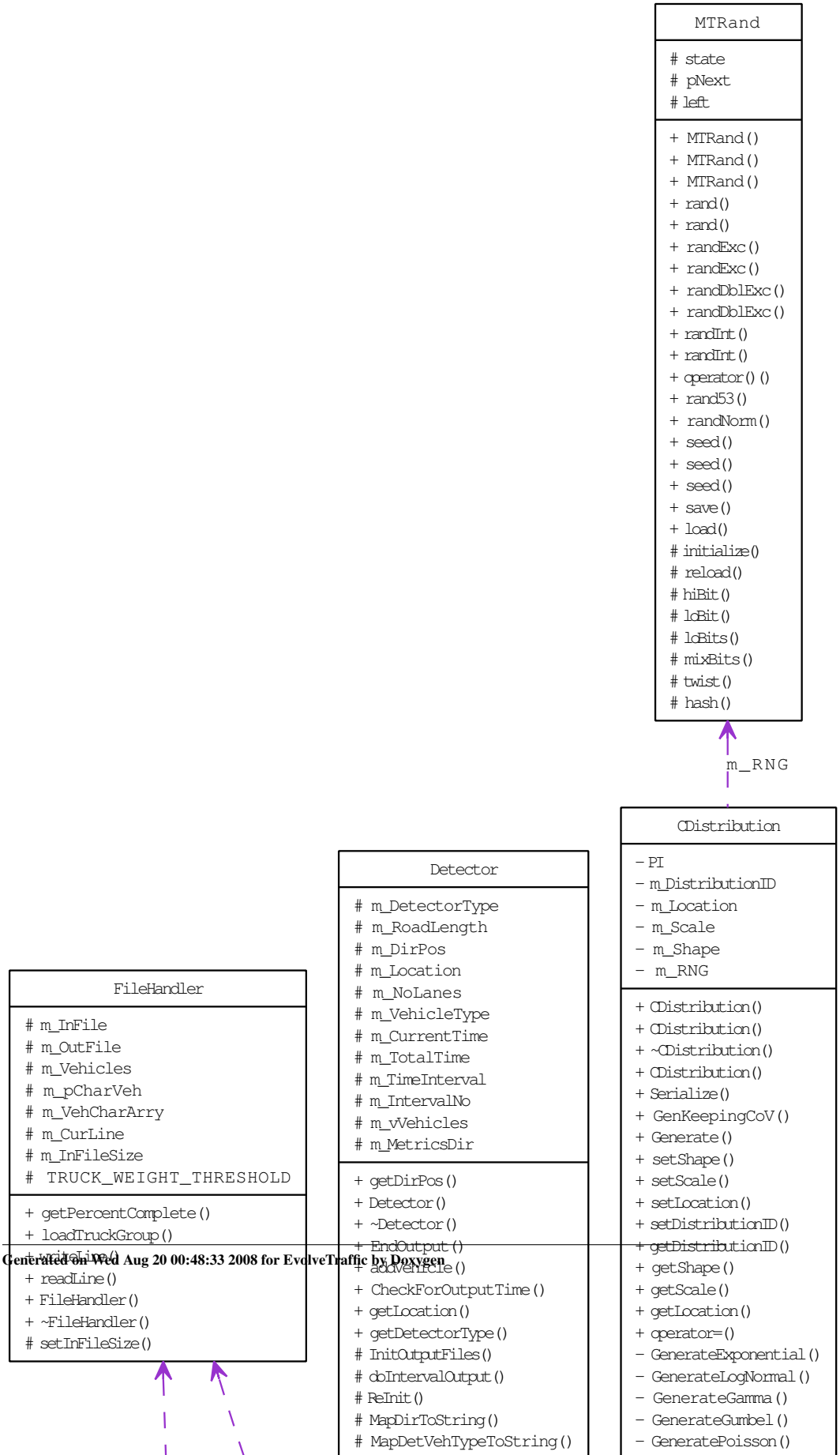
- D:/∼Research/Code/C++/EvolveTraffic/OutputDetector.h
- D:/∼Research/Code/C++/EvolveTraffic/OutputDetector.cpp

4.47 Road Class Reference

A class to represent a road in the simulation.

`#include <Road.h>`

Collaboration diagram for Road:



Public Member Functions

- void setLaneChangeReg (bool trackChanges, double outputTime, int roadLen, WORD registerType)
- void setDriveOnRight (bool OnRight)

  Sets whether or not the vehicles should drive on the right.

- void clear ()

  Clears the road so it can be used in another simulation.

- int getPercentComplete ()
- bool getAllowLaneChanging ()

  Gets whether or not lane changing is allowed.

- void setAllowLaneChanging (bool status)

  Sets whether or not lane changing is allowed.

- void setLocOuputDetectorDirPos (int loc)

  Sets the position of output detector in the positive direction.

- void setLocOuputDetectorDirNeg (int loc)

  Sets the position of output detector in the negative direction.

- void setNoLanesDirPos (int nlpos)

  Sets the number of lanes in the positive direction.

- void setNoLanesDirNeg (int nlneg)

  Sets the number of lanes in the negative direction.

- void setNoDirections (int nd)

  Sets the number of directions.

- void setNoLanes (int nl)

  Sets the number of lanes.

- void setRoadLength (int L)

  Sets the length of the road.

- void setTrafFileNoLanesDirPos (int nl)

  Sets the number of lanes in the positive direction in the traffic file.

- void setTrafFileNoLanesDirNeg (int nl)

  Sets the number of lanes in the negative direction in the traffic file.

- void setIDMParams_Car (CIDMParameterSet *Params)

  Sets the Car IDM Parameter Set.

- void setIDMParams_SmallTruck (CIDMParameterSet *Params)

  Sets the Small Truck IDM Parameter Set.

- void setIDMParams_LargeTruck (CIDMParameterSet *Params)

  Sets the Large Truck IDM Parameter Set.

- void setIDMParams_Crane (CIDMParameterSet *Params)

  Sets the Crane IDM Parameter Set.

- void setIDMParams_Lowloader (CIDMParameterSet *Params)

  Sets the Lowloader IDM Parameter Set.

- void SetSegmentFromFeature (CRoadFeature *pFeat)

  Sets road segments based on data specified by the user to the GUI.

- void SetMetricDetFromStatDet (CStatDetector *pDet, int nLanesPos, int nLanesNeg)

  Sets the road metrics detector data from input through the GUI.

- double getLength ()

  Gets the length of the road.

- M2D getVehicles ()

  Gets all the vehicles on the road.

- void initTruckGroup (string inFile, string outFile, WORD fileType)

  Initialises the file handler.

- bool update (double step, const double curTime)

  Updates the properties of the road.

- void populate (double step, const double curTime)

  Populates the road.

- double init ()

  Initialises the road.

- Road ()

  Constructor.

- virtual ~Road ()

  Destructor.

Private Member Functions

- void passIDMParameters (RoadSegment *segment)

  Passes the IDM parameters to a given road segment.

- void MapTrafLaneToSimLane (Vehicle *pVeh)

  Maps vehicles to appropriate lanes.

- void setIDMDriverModel (Vehicle *pVeh)

  Sets a vehicle's drivermodel as IDM.

- void SetSpeedLimitSegment (CRoadFeature *pFeat)

  Sets a speed limit based on data specified by the user to the GUI.

- void SetGradientSegment (CRoadFeature *pFeat)

  Sets a gradient based on data specified by the user to the GUI.

Private Attributes

- bool m_DriveOnRight
- int m_PercentComplete
- CIDMParameterSet * m_pIDMParams_Car
- CIDMParameterSet * m_pIDMParams_SmallTruck
- CIDMParameterSet * m_pIDMParams_LargeTruck
- CIDMParameterSet * m_pIDMParams_Crane
- CIDMParameterSet * m_pIDMParams_Lowloader
- int m_LocOutputDetectorDirPos
- int m_LocOutputDetectorDirNeg
- Direction DirectionPos
- Direction DirectionNeg
- int m_TrafFileNoLanesDirPos
- int m_TrafFileNoLanesDirNeg
- FileHandler * m_pFileHandler
- std::vector< Vehicle * > m_BufferVehicles
- std::vector< Lane * > m_vLanes
- std::vector< Detector * > m_vDetectorsPos
- std::vector< Detector * > m_vDetectorsNeg
- std::vector< int > m_vLaneLengthsPos
- std::vector< int > m_vLaneLengthsNeg
- std::vector< RoadSegment * > m_vRoadSegmentsPos
- std::vector< RoadSegment * > m_vRoadSegmentsNeg
- bool m_bEndOfFile
- bool m_AllowLaneChanging
- int m_NoLanesDirPos
- int m_NoLanesDirNeg

- int m_NoDirections
- int m_NoLanes
- int m_RoadLength
- double MIN_SPACE_FOR_NEXT_VEHICLE

4.47.1 Detailed Description

A class to represent a road in the simulation.

Definition at line 21 of file Road.h.

4.47.2 Constructor & Destructor Documentation

4.47.2.1 Road::Road ()

Constructor.

Definition at line 9 of file Road.cpp.

References CConfigData::IDM, m_bEndOfFile, m_pFileHandler, m_pIDMParams_Car, m_pIDMParams_Crane, m_pIDMParams_LargeTruck, m_pIDMParams_Lowloader, m_pIDMParams_SmallTruck, CConfigData::IDM_Config::MIN_SPACE_FOR_NEXT_VEHICLE, and MIN_SPACE_FOR_NEXT_VEHICLE.

```
10 {
11         MIN_SPACE_FOR_NEXT_VEHICLE = g_ConfigData.IDM.MIN_SPACE_FOR_NEXT_VEHICLE;
12
13         m_pIDMParams_Car               = NULL;
14         m_pIDMParams_SmallTruck = NULL;
15         m_pIDMParams_LargeTruck = NULL;
16         m_pIDMParams_Crane             = NULL;
17         m_pIDMParams_Lowloader  = NULL;
18
19         m_pFileHandler                 = NULL;
20
21         m_bEndOfFile = false;
22 }
```

4.47.2.2 Road::~Road () [virtual]

Destructor.

Definition at line 25 of file Road.cpp.

References m_pIDMParams_Car, m_pIDMParams_Crane, m_pIDMParams_LargeTruck, m_pIDMParams_Lowloader, and m_pIDMParams_SmallTruck.

```
26 {
27         m_pIDMParams_Car               = NULL;
28         m_pIDMParams_SmallTruck = NULL;
29         m_pIDMParams_LargeTruck = NULL;
30         m_pIDMParams_Crane             = NULL;
31         m_pIDMParams_Lowloader  = NULL;
32 }
```

4.47.3 Member Function Documentation

4.47.3.1 void Road::setLaneChangeReg (bool *trackChanges*, double *outputTime*, int *roadLen*, WORD *registerType*)

4.47.3.2 void Road::setDriveOnRight (bool *OnRight*)

Sets whether or not the vehicles should drive on the right.

Parameters:

OnRight Whether or not the vehicles should drive on the right

Definition at line 445 of file Road.cpp.

References m_DriveOnRight.

Referenced by CEvolveTrafficDoc::initSim().

```
446 {
447         m_DriveOnRight = OnRight;
448 }
```

4.47.3.3 void Road::clear ()

Clears the road so it can be used in another simulation.

This function clears all of the road's information so that it may be used in another simulation without re-instanciating.

Definition at line 608 of file Road.cpp.

References Direction::clear(), DirectionNeg, DirectionPos, m_BufferVehicles, m_NoDirections, m_pFileHandler, m_pIDMParams_Car, m_pIDMParams_Crane, m_pIDMParams_LargeTruck, m_pIDMParams_Lowloader, m_pIDMParams_SmallTruck, m_vDetectorsNeg, m_vDetectorsPos, m_vLanes, m_vRoadSegmentsNeg, and m_vRoadSegmentsPos.

Referenced by Sim::clear().

```
609 {
610         DirectionPos.clear();
611         if(m_NoDirections == 2)
612                 DirectionNeg.clear();
613
614         delete m_pFileHandler;
615
616         m_vLanes.clear();
617
618         m_BufferVehicles.clear();
619
620         m_vDetectorsPos.clear();
621         m_vDetectorsNeg.clear();
622
623         m_vRoadSegmentsPos.clear();
```

```
624         m_vRoadSegmentsNeg.clear();
625
626         // why don't we just delete these?
627         m_pIDMParams_Car               = NULL;
628         m_pIDMParams_SmallTruck = NULL;
629         m_pIDMParams_LargeTruck = NULL;
630         m_pIDMParams_Crane             = NULL;
631         m_pIDMParams_Lowloader  = NULL;
632 }
```

Here is the call graph for this function:

4.47.3.4 int Road::getPercentComplete ()

Definition at line 418 of file Road.cpp.

References m_PercentComplete.

Referenced by Sim::doOneTimeStep().

```
419 {
420         return m_PercentComplete;
421 }
```

4.47.3.5 bool Road::getAllowLaneChanging ()

Gets whether or not lane changing is allowed.

Returns:

Whether or not lane changing is allowed

Definition at line 427 of file Road.cpp.

References m_AllowLaneChanging.

```
428 {
429         return m_AllowLaneChanging;
430 }
```

4.47.3.6 void Road::setAllowLaneChanging (bool *status*)

Sets whether or not lane changing is allowed.

Parameters:

status Whether or not lane changing is allowed

Definition at line 436 of file Road.cpp.

References m_AllowLaneChanging.

Referenced by CEvolveTrafficDoc::initSim().

```
437 {
438         m_AllowLaneChanging = status;
439 }
```

4.47.3.7 void Road::setLocOuputDetectorDirPos (int *loc*)

Sets the position of output detector in the positive direction.

Parameters:

loc The position of output detector

Definition at line 402 of file Road.cpp.

References m_LocOutputDetectorDirPos.

Referenced by CEvolveTrafficDoc::initSim().

```
403 {
404         m_LocOutputDetectorDirPos = loc;
405 }
```

4.47.3.8 void Road::setLocOuputDetectorDirNeg (int *loc*)

Sets the position of output detector in the negative direction.

Parameters:

loc The position of output detector

Definition at line 412 of file Road.cpp.

References m_LocOutputDetectorDirNeg, and m_RoadLength.

Referenced by CEvolveTrafficDoc::initSim().

```
413 {
414         // change from overall road coordinates to lane coordinates
415         m_LocOutputDetectorDirNeg = m_RoadLength - loc;
416 }
```

4.47.3.9 void Road::setNoLanesDirPos (int *nlpos*)

Sets the number of lanes in the positive direction.

Parameters:

nlpos The number of lanes in the positive direction

Definition at line 271 of file Road.cpp.

References m_NoLanesDirPos.

Referenced by Sim::init().

```
272 {
273         m_NoLanesDirPos = nlpos;
274 }
```

4.47.3.10 void Road::setNoLanesDirNeg (int *nlneg*)

Sets the number of lanes in the negative direction.

Parameters:

nlneg The number of lanes in the negative direction

Definition at line 280 of file Road.cpp.

References m_NoLanesDirNeg.

Referenced by Sim::init().

```
281 {
282         m_NoLanesDirNeg = nlneg;
283 }
```

4.47.3.11 void Road::setNoDirections (int *nd*)

Sets the number of directions.

Parameters:

nd The number of directions

Definition at line 289 of file Road.cpp.

References m_NoDirections.

Referenced by Sim::init().

```
290 {
291         m_NoDirections = nd;
292 }
```

4.47.3.12 void Road::setNoLanes (int *nl*)

Sets the number of lanes.

Parameters:

nl The number of lanes

Definition at line 298 of file Road.cpp.

References m_NoLanes.

Referenced by Sim::init().

```
299 {
300         m_NoLanes = nl;
301 }
```

4.47.3.13 void Road::setRoadLength (int *L*)

Sets the length of the road.

Parameters:

L The length of the road

Definition at line 307 of file Road.cpp.

References m_RoadLength.

Referenced by Sim::init(), and Sim::setRoadLength().

```
308 {
309         m_RoadLength = L;
310 }
```

4.47.3.14 void Road::setTrafFileNoLanesDirPos (int *nl*)

Sets the number of lanes in the positive direction in the traffic file.

Parameters:

nl The number of lanes

Definition at line 384 of file Road.cpp.

References m_TrafFileNoLanesDirPos.

Referenced by CEvolveTrafficDoc::initSim().

```
385 {
386         m_TrafFileNoLanesDirPos = nl;
387 }
```

4.47.3.15 void Road::setTrafFileNoLanesDirNeg (int *nl*)

Sets the number of lanes in the negative direction in the traffic file.

Parameters:

nl The number of lanes

Definition at line 393 of file Road.cpp.

References m_TrafFileNoLanesDirNeg.

Referenced by CEvolveTrafficDoc::initSim().

```
394 {
395         m_TrafFileNoLanesDirNeg = nl;
396 }
```

4.47.3.16 void Road::setIDMParams_Car (CIDMParameterSet * *Params*)

Sets the Car IDM Parameter Set.

Parameters:

Params The IDM Parameter Set

Definition at line 339 of file Road.cpp.

References m_pIDMParams_Car.

Referenced by CEvolveTrafficDoc::initSim().

```
340 {
341         m_pIDMParams_Car = Params;
342 }
```

4.47.3.17 void Road::setIDMParams_SmallTruck (CIDMParameterSet * *Params*)

Sets the Small Truck IDM Parameter Set.

Parameters:

Params The IDM Parameter Set

Definition at line 348 of file Road.cpp.

References m_pIDMParams_SmallTruck.

Referenced by CEvolveTrafficDoc::initSim().

```
349 {
350         m_pIDMParams_SmallTruck = Params;
351 }
```

4.47.3.18 void Road::setIDMParams_LargeTruck (CIDMParameterSet * *Params*)

Sets the Large Truck IDM Parameter Set.

Parameters:

Params The IDM Parameter Set

Definition at line 357 of file Road.cpp.

References m_pIDMParams_LargeTruck.

Referenced by CEvolveTrafficDoc::initSim().

```
358 {
359         m_pIDMParams_LargeTruck = Params;
360 }
```

4.47.3.19 void Road::setIDMParams_Crane (CIDMParameterSet * *Params*)

Sets the Crane IDM Parameter Set.

Parameters:

Params The IDM Parameter Set

Definition at line 366 of file Road.cpp.

References m_pIDMParams_Crane.

Referenced by CEvolveTrafficDoc::initSim().

```
367 {
368         m_pIDMParams_Crane = Params;
369 }
```

4.47.3.20 void Road::setIDMParams_Lowloader (CIDMParameterSet * *Params*)

Sets the Lowloader IDM Parameter Set.

Parameters:

Params The IDM Parameter Set

Definition at line 375 of file Road.cpp.

References m_pIDMParams_Lowloader.

Referenced by CEvolveTrafficDoc::initSim().

```
376 {
377         m_pIDMParams_Lowloader = Params;
378 }
```

4.47.3.21 void Road::SetSegmentFromFeature (CRoadFeature * *pFeat*)

Sets road segments based on data specified by the user to the GUI.

Parameters:

pFeat The road feature to create

This function takes the data inputted by the user into the GUI and creates the correct road segment based on the given data

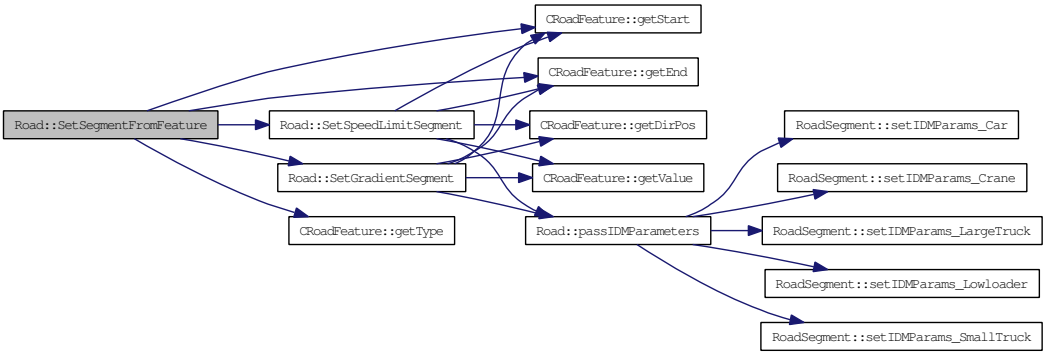
Definition at line 470 of file Road.cpp.

References FEAT_GRADIENT, FEAT_SPEEDLIMIT, CRoadFeature::getEnd(), CRoadFeature::getStart(), CRoadFeature::getType(), SetGradientSegment(), and SetSpeedLimitSegment().

Referenced by CEvolveTrafficDoc::initSim().

```
471 {
472         ASSERT(pFeat->getStart() < pFeat->getEnd());    // should be since done in CRoadFeature
473
474         switch( pFeat->getType() )
475         {
476                 case FEAT_SPEEDLIMIT:
477                         SetSpeedLimitSegment(pFeat);
478                         break;
479                 case FEAT_GRADIENT:
480                         SetGradientSegment(pFeat);
481                         break;
482                 default:
483                         break;
484         }
485
486 }
```

Here is the call graph for this function:



4.47.3.22 void Road::SetMetricDetFromStatDet (CStatDetector * *pDet*, int *nLanesPos*, int *nLanesNeg*)

Sets the road metrics detector data from input through the GUI.

Parameters:

pDet The detector to create

nLanesPos The number of lanes in the positive direction

nLanesNeg The number of lanes in the negative direction

This function takes the data inputted by the user into the GUI and creates a detector based on the given data. The detector is then passed to the correct direction.

Definition at line 555 of file Road.cpp.

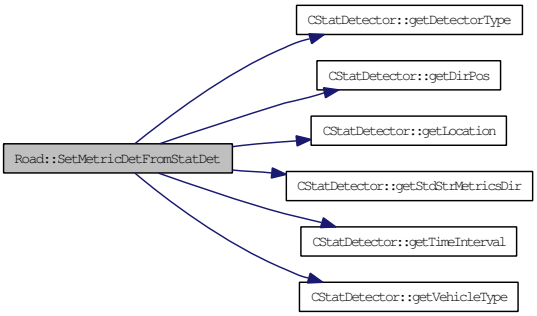
References CStatDetector::getDetectorType(), CStatDetector::getDirPos(), CStatDetector::getLocation(), CStatDetector::getStdStrMetricsDir(), CStatDetector::getTimeInterval(), CStatDetector::getVehicleType(), m_RoadLength, m_vDetectorsNeg, m_vDetectorsPos, and METRICS_TYPE_LANE_CHANGE.

Referenced by CEvolveTrafficDoc::initSim().

```
556 {
557         bool bDirPos = pDet->getDirPos();
558 
559         if(pDet->getDetectorType() == METRICS_TYPE_LANE_CHANGE)
560         {
561                 LaneChangeDetector* pLCD;
562                 pLCD = new LaneChangeDetector(  pDet->getStdStrMetricsDir(),
563                                                                                 pDet->getTimeIn
564                                                                                 pDet->getLocati
565                                                                                 m_RoadLength,
566                                                                                 bDirPos,
567                                                                                 pDet->getVehicl
568                 if( bDirPos )
569                         m_vDetectorsPos.push_back(pLCD);
570                 else
571                         m_vDetectorsNeg.push_back(pLCD);
572         }
573         else
574         {
575                 MetricsDetector* pMD;
576                 if( bDirPos )
577                 {
578                         pMD = new MetricsDetector(      pDet->getStdStrMetricsDir(),
579                                                                                 pDet->getDetect
580                                                                                 bDirPos,
581                                                                                 nLanesPos,
582                                                                                 pDet->getVehicl
583                                                                                 pDet->getTimeIn
584                                                                                 pDet->getLocati
585                                                                                 m_RoadLength);
586                         m_vDetectorsPos.push_back(pMD);
587                 }
588                 else
589                 {
590                         pMD = new MetricsDetector(      pDet->getStdStrMetricsDir(),
591                                                                                 pDet->getDetect
592                                                                                 bDirPos,
593                                                                                 nLanesNeg,
594                                                                                 pDet->getVehicl
595                                                                                 pDet->getTimeIn
596                                                                                 m_RoadLength -
597                                                                                 m_RoadLength);
```

```
598                         m_vDetectorsNeg.push_back(pMD);
599                 }
600         }
601 }
```

Here is the call graph for this function:



4.47.3.23 double Road::getLength ()

Gets the length of the road.

Returns:

The length of the road

Definition at line 262 of file Road.cpp.

References m_RoadLength.

Referenced by Sim::getRoadLength().

```
263 {
264         return m_RoadLength;
265 }
```

4.47.3.24 M2D Road::getVehicles ()

Gets all the vehicles on the road.

Returns:

All the vehicles on the road

Definition at line 238 of file Road.cpp.

References DirectionNeg, DirectionPos, Direction::getGlobalPos(), m_NoDirections, m_NoLanesDirNeg, and m_NoLanesDirPos.

Referenced by Sim::doOneTimeStep().

```
239 {
240         M2D pos;
241
242         M2D temp = DirectionPos.getGlobalPos(); // get DirPos positions
243         for(int i = 0; i < m_NoLanesDirPos; i++)                // and add lane by lane
244                 pos.push_back(temp[i]);
245
246         if(m_NoDirections == 2)         // Uni-directional roads are in Dir1
247         {
248                 temp.clear();
249
250                 temp = DirectionNeg.getGlobalPos();     // get DirNeg positions
251                 for(i = 0; i < m_NoLanesDirNeg; i++)            // and add lane by lane
252                         pos.push_back(temp[i]);
253         }
254
255         return pos;
256 }
```

Here is the call graph for this function:

4.47.3.25 void Road::initTruckGroup (string *inFile*, string *outFile*, WORD *fileType*)

Initialises the file handler.

Parameters:

inFile The input file to read from

outFile The output file to write to

fileType The type of file

This function determines whether the type of file being used is CASTOR or SAFT format, and creates an appropriate filehandler based on the input provided.

Definition at line 322 of file Road.cpp.

References CASTOR, m_pFileHandler, and SAFT.

Referenced by Sim::init().

```
323 {
324         if(fileType == CASTOR)
325         {
326                 m_pFileHandler = new CASTORFile(inFile, outFile);
327
328         }
329         else if(fileType == SAFT)
330         {
331                 m_pFileHandler = new SAFTFile(inFile, outFile);
332         }
333 }
```

4.47.3.26 bool Road::update (double *step*, const double *curTime*)

Updates the properties of the road.

Parameters:

step The timestep

curTime The current time

Returns:

Whether or not the simulation is still running

This function is called for each timestep of the simulation and updates its directions, once the directions are not empty

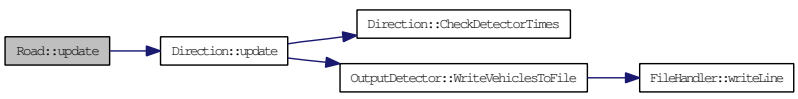
Definition at line 43 of file Road.cpp.

References DirectionNeg, DirectionPos, m_bEndOfFile, m_NoDirections, m_vDetectorsNeg, m_vDetectorsPos, and Direction::update().

Referenced by Sim::doOneTimeStep().

```
44 {
45         bool SimulationOver = false;
46         bool RoadEmpty = false;
47         bool DirPosEmpty = false;
48         bool DirNegEmpty = false;
49
50         // since there's always one direction
51         DirPosEmpty = DirectionPos.update(step, curTime);
52
53         if(m_NoDirections == 2)
54                 DirNegEmpty = DirectionNeg.update(step, curTime);
55
56         if(DirPosEmpty && DirNegEmpty)
57                 RoadEmpty = true;
58
59         if(RoadEmpty && m_bEndOfFile)   // if no more vehicles and road is empty
60                 SimulationOver = true;          // we are finished!
61
62         if(SimulationOver)      // tidy up detector output
63         {
64                 for(int i = 0; i < m_vDetectorsPos.size(); i++)
65                         m_vDetectorsPos[i]->EndOutput();
66                 for(i = 0; i < m_vDetectorsNeg.size(); i++)
67                         m_vDetectorsNeg[i]->EndOutput();
68         }
69
70         return SimulationOver;
71 }
```

Here is the call graph for this function:



4.47.3.27 void Road::populate (double *step*, const double *curTime*)

Populates the road.

Parameters:

step The timestep

curTime The simulation's current time

This function handles the population of the road which is called in each timestep. The main loop handles the processing of the vehicles in the buffer. If the head of the buffer is currently entering the road, then the other vehicles are also checked. However, since the vehicles are sorted in order of time, if the head of the buffer is not currently entering the road, then no other vehicle is either. When a vehicle enters the road from the buffer, the next vehicle is read from the file being used, and added to the buffer.

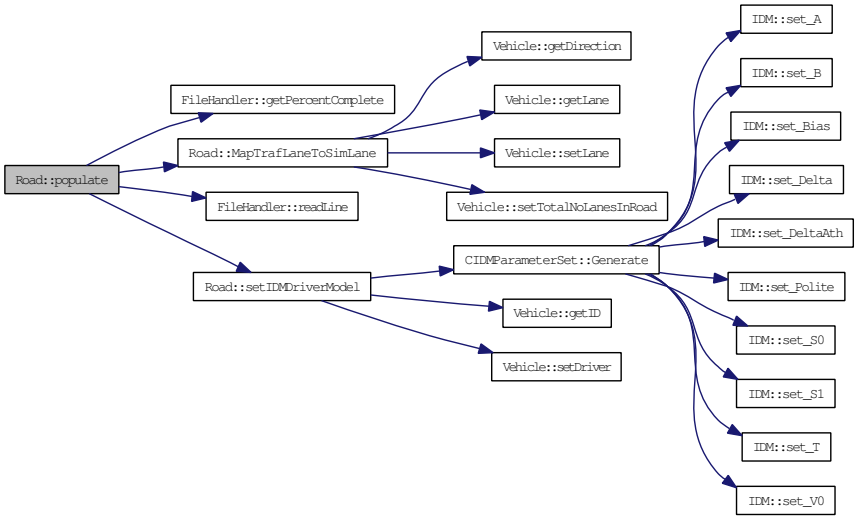
Definition at line 141 of file Road.cpp.

References FileHandler::getPercentComplete(), m_bEndOfFile, m_BufferVehicles, m_DriveOnRight, m_PercentComplete, m_pFileHandler, m_RoadLength, m_vLanes, MapTrafLaneToSimLane(), MIN_SPACE_FOR_NEXT_VEHICLE, FileHandler::readLine(), and setIDMDriverModel().

Referenced by Sim::doOneTimeStep().

```
142 {
143         bool frontOn = true;
144         while(m_BufferVehicles.size() > 0 && frontOn)
145         {
146                 frontOn = false;
147                 int lane = m_BufferVehicles.front()->getLane() - 1;             // the cumulati
148
149                 if(m_vLanes.at(lane)->getLastPos() >= MIN_SPACE_FOR_NEXT_VEHICLE)
150                 {
151                         if(m_BufferVehicles.front()->getTime() <= curTime)      // If it is tim
152                         {
153                                 int noVeh = m_vLanes.at(lane)->getNoVeh();              // cras
154
155                                 // set the driver, roadlength, insert into lane and remove fron
156                                 setIDMDriverModel(m_BufferVehicles.front());
157                                 m_BufferVehicles.front()->init(m_RoadLength, m_DriveOnRight);
158                                 m_vLanes.at(lane)->insert(noVeh, m_BufferVehicles.front());
159                                 m_BufferVehicles.erase(m_BufferVehicles.begin());
160
161                                 Vehicle* tempVeh = m_pFileHandler->readLine();
162                                 m_PercentComplete = m_pFileHandler->getPercentComplete();
163                                 if(tempVeh != NULL)
164                                 {
165                                         MapTrafLaneToSimLane(tempVeh);
166                                         m_BufferVehicles.push_back(tempVeh);
167                                 }
168                                 else
169                                         m_bEndOfFile = true;
170                                 frontOn = true;
171                         }
172                 }
173
174         }
175 }
```

Here is the call graph for this function:



4.47.3.28 double Road::init ()

Initialises the road.

Returns:

The time to the first vehicle

This function initialises the road object. Firstly, the special segments of the road are sorted depending on their road position. Next, we load a buffer of vehicles which the road processes, and maps these vehicles to their respective lanes. Directions are initialised, and the road then takes account of cumulative lane identifiers by loading each direction's lanes into a vector of cumulative lanes. Detectors, if specified by the user, are also set in each direction to facilitate the retrieval of updated vehicle information when a vehicle passes a specified point.

Definition at line 85 of file Road.cpp.

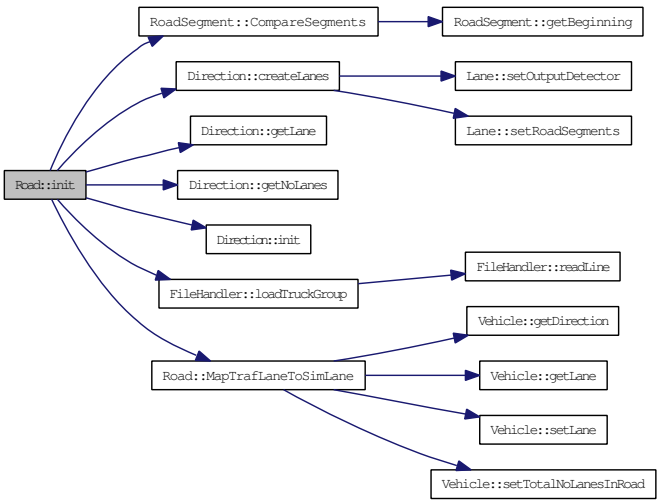
References RoadSegment::CompareSegments(), Direction::createLanes(), DirectionNeg, DirectionPos, Direction::getLane(), Direction::getNoLanes(), Direction::init(), INPUT_FILE_BUFFER_SIZE, FileHandler::loadTruckGroup(), m_AllowLaneChanging, m_BufferVehicles, m_DriveOnRight, m_LocOutputDetectorDirNeg, m_LocOutputDetectorDirPos, m_NoDirections, m_NoLanesDirNeg, m_NoLanesDirPos, m_pFileHandler, m_RoadLength, m_vDetectorsNeg, m_vDetectorsPos, m_vLanes, m_vRoadSegmentsNeg, m_vRoadSegmentsPos, and MapTrafLaneToSimLane().

Referenced by Sim::init().

```
86 {
```

```
87          sort(m_vRoadSegmentsPos.begin(), m_vRoadSegmentsPos.end(), RoadSegment::CompareSegments)
88          sort(m_vRoadSegmentsNeg.begin(), m_vRoadSegmentsNeg.end(), RoadSegment::CompareSegments)
89
90          // Initially set up vehicles and their sim lanes
91          m_BufferVehicles = m_pFileHandler->loadTruckGroup(INPUT_FILE_BUFFER_SIZE);
92          for(int i = 0; i < m_BufferVehicles.size(); i++)
93                  MapTrafLaneToSimLane(m_BufferVehicles.at(i));
94
96          // lane vector
97          std::vector<int> vLaneLengthsPos(m_NoLanesDirPos,m_RoadLength); // this is for future ex
98
99          // true means positive x-direction
100          OutputDetector* pDetDirPos = new OutputDetector(m_LocOutputDetectorDirPos, true, m_pFil
101          DirectionPos.init(true, pDetDirPos, m_AllowLaneChanging, m_RoadLength, m_DriveOnRight);
102          DirectionPos.createLanes(0, vLaneLengthsPos, m_vDetectorsPos, m_vRoadSegmentsPos);
103
104          // Using cumulative lane identifiers
105          for(i = 0; i < DirectionPos.getNoLanes(); i++)
106                  m_vLanes.push_back(&DirectionPos.getLane(i));
107
109          if(m_NoDirections == 2)
110          {
111                  // lane vector
112                  std::vector<int> vLaneLengthsNeg(m_NoLanesDirNeg,m_RoadLength);
113
114                  // false means the negative x-direction
115                  OutputDetector* pDetDirNeg = new OutputDetector(m_LocOutputDetectorDirNeg, fals
116                  DirectionNeg.init(false, pDetDirNeg, m_AllowLaneChanging, m_RoadLength, m_Drive
117                  DirectionNeg.createLanes(m_NoLanesDirPos, vLaneLengthsNeg, m_vDetectorsNeg, m_v
118
119                  for(i = 0; i < DirectionNeg.getNoLanes(); i++)
120                          m_vLanes.push_back(&DirectionNeg.getLane(i));
121          }
122
123          // Return the time of the first truck of the traffic file
124          return m_BufferVehicles.front()->getTime();
125  }
```

Here is the call graph for this function:



4.47.3.29 void Road::passIDMParameters (RoadSegment * *segment*) [private]

Passes the IDM parameters to a given road segment.

Parameters:

segment The segment to pass the parameters to

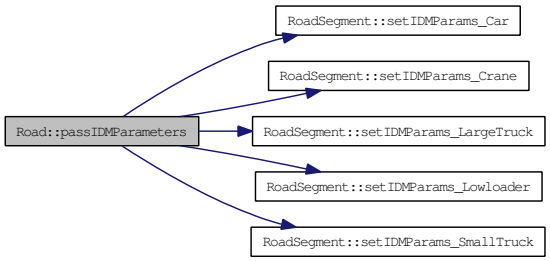
Definition at line 454 of file Road.cpp.

References m_pIDMParams_Car, m_pIDMParams_Crane, m_pIDMParams_LargeTruck, m_pIDMParams_Lowloader, m_pIDMParams_SmallTruck, RoadSegment::setIDMParams_Car(), RoadSegment::setIDMParams_Crane(), RoadSegment::setIDMParams_LargeTruck(), RoadSegment::setIDMParams_Lowloader(), and RoadSegment::setIDMParams_SmallTruck().

Referenced by SetGradientSegment(), and SetSpeedLimitSegment().

```
455 {
456         segment->setIDMParams_Car(m_pIDMParams_Car);
457         segment->setIDMParams_Crane(m_pIDMParams_Crane);
458         segment->setIDMParams_LargeTruck(m_pIDMParams_LargeTruck);
459         segment->setIDMParams_Lowloader(m_pIDMParams_Lowloader);
460         segment->setIDMParams_SmallTruck(m_pIDMParams_SmallTruck);
461 }
```

Here is the call graph for this function:



4.47.3.30 void Road::MapTrafLaneToSimLane (Vehicle * *pVeh*) [private]

Maps vehicles to appropriate lanes.

Parameters:

pVeh The vehicle to map

This function updates the lane provided by the vehicle's original traffic file assignment to an appropriate lane in the simulation. This is to facilitate the simulation of more lanes than the input file allows for

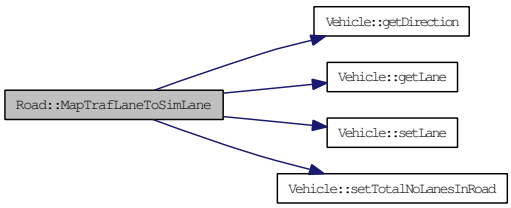
Definition at line 185 of file Road.cpp.

References Vehicle::getDirection(), Vehicle::getLane(), m_NoLanes, m_TrafFileNoLanesDirNeg, m_TrafFileNoLanesDirPos, Vehicle::setLane(), and Vehicle::setTotalNoLanesInRoad().

Referenced by init(), and populate().

```
186 {
187         // first tell the vehicle how many lanes the road has for later use
188         pVeh->setTotalNoLanesInRoad(m_NoLanes);
189
190         // then map its current local to global lane number
191         if(!pVeh->getDirection())       // only if it's in the negative direction
192         {
193                 int TrafFileNoLanes = m_TrafFileNoLanesDirPos + m_TrafFileNoLanesDirNeg;
194                 int lanesFromNegVerge = TrafFileNoLanes - pVeh->getLane();
195                 int newLane = m_NoLanes - lanesFromNegVerge;    // vehicle lane numbers are 1-b
196                 pVeh->setLane(newLane);
197         }
198 }
```

Here is the call graph for this function:



4.47.3.31 void Road::setIDMDriverModel (Vehicle * *pVeh*) [private]

Sets a vehicle's drivermodel as IDM.

Parameters:

pVeh The vehicle to set

This function sets the drivermodel of a particular vehicle based on the type of that vehicle

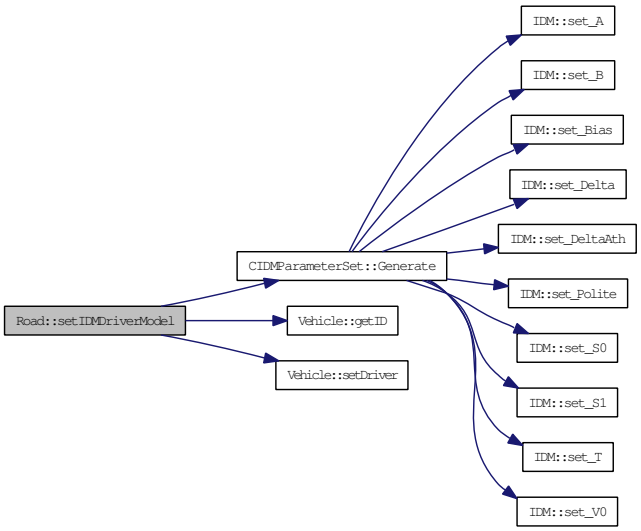
Definition at line 207 of file Road.cpp.

References CIDMParameterSet::Generate(), Vehicle::getID(), m_pIDMParams_Car, m_pIDMParams_Crane, m_pIDMParams_LargeTruck, m_pIDMParams_Lowloader, m_pIDMParams_SmallTruck, Vehicle::setDriver(), VEH_ID_CAR, VEH_ID_CRANE, VEH_ID_LARGETRUCK, VEH_ID_LOWLOADER, and VEH_ID_SMALLTRUCK.

Referenced by populate().

```
208 {
209         WORD VEH_ID = pVeh->getID();
210 
211         switch(VEH_ID)
212         {
213                 case VEH_ID_CAR:
214                         pVeh->setDriver( m_pIDMParams_Car->Generate() );
215                         break;
216                 case VEH_ID_SMALLTRUCK:
217                         pVeh->setDriver( m_pIDMParams_SmallTruck->Generate() );
218                         break;
219                 case VEH_ID_LARGETRUCK:
220                         pVeh->setDriver( m_pIDMParams_LargeTruck->Generate() );
221                         break;
222                 case VEH_ID_CRANE:
223                         pVeh->setDriver( m_pIDMParams_Crane->Generate() );
224                         break;
225                 case VEH_ID_LOWLOADER:
226                         pVeh->setDriver( m_pIDMParams_Lowloader->Generate() );
227                         break;
228                 default:
229                         pVeh->setDriver( m_pIDMParams_Car->Generate() );
230         };
231 }
```

Here is the call graph for this function:



4.47.3.32 void Road::SetSpeedLimitSegment (CRoadFeature * *pFeat*) [private]

Sets a speed limit based on data specified by the user to the GUI.

Parameters:

pFeat The speed limit to create

This function takes the data inputted by the user into the GUI and creates a speed limit based on the given data

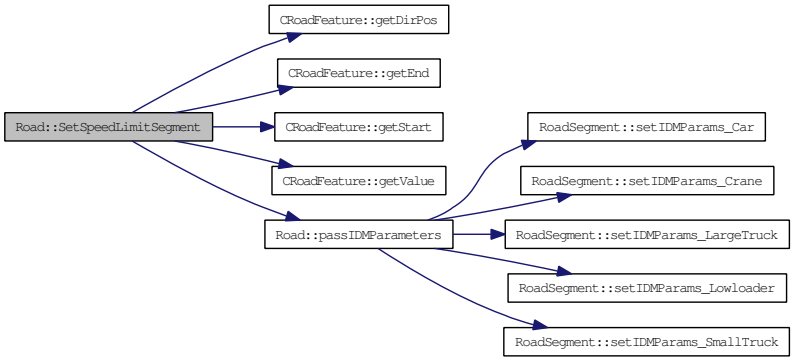
Definition at line 495 of file Road.cpp.

References CRoadFeature::getDirPos(), CRoadFeature::getEnd(), CRoadFeature::getStart(), CRoadFeature::getValue(), m_RoadLength, m_vRoadSegmentsNeg, m_vRoadSegmentsPos, and passIDMParameters().

Referenced by SetSegmentFromFeature().

```
496 {
497         bool bDirPos = pFeat->getDirPos();
498         SpeedLimit* pSL;
499         if( bDirPos )
500         {
501                 pSL = new SpeedLimit(pFeat->getStart(), pFeat->getEnd(), pFeat->getValue(), bDi
502                 m_vRoadSegmentsPos.push_back(pSL);
503         }
504         else
505         {
506                 int start = pFeat->getStart();
507                 int end = pFeat->getEnd();
508
509                 pSL = new SpeedLimit(m_RoadLength - end, m_RoadLength - start, pFeat->getValue
510
511                 m_vRoadSegmentsNeg.push_back(pSL);
512         }
513         passIDMParameters(pSL);
514 }
```

Here is the call graph for this function:



4.47.3.33 void Road::SetGradientSegment (CRoadFeature * *pFeat*) [private]

Sets a gradient based on data specified by the user to the GUI.

Parameters:

pFeat The gradient to create

This function takes the data inputted by the user into the GUI and creates a gradient based on the given data

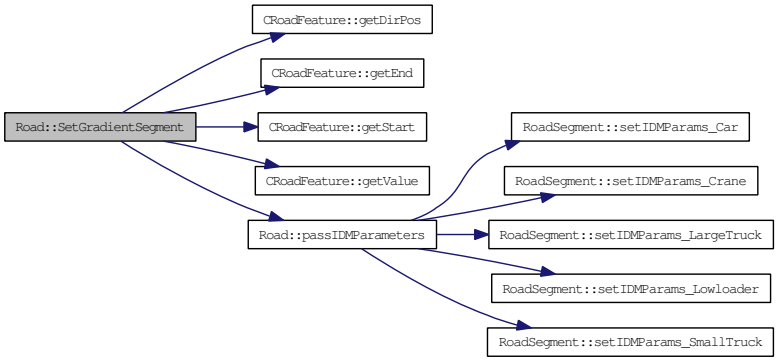
Definition at line 523 of file Road.cpp.

References CRoadFeature::getDirPos(), CRoadFeature::getEnd(), CRoadFeature::getStart(), CRoadFeature::getValue(), m_RoadLength, m_vRoadSegmentsNeg, m_vRoadSegmentsPos, and passIDMParameters().

Referenced by SetSegmentFromFeature().

```
524 {
525         bool bDirPos = pFeat->getDirPos();
526         Gradient* pGr;
527 
528         if( bDirPos )
529         {
530                 pGr = new Gradient(pFeat->getStart(), pFeat->getEnd(), pFeat->getValue(), bDirP
531                 m_vRoadSegmentsPos.push_back(pGr);
532         }
533         else
534         {
535                 int start = pFeat->getStart();
536                 int end = pFeat->getEnd();
537 
538                 pGr = new Gradient(m_RoadLength - end, m_RoadLength - start, pFeat->getValue(),
539 
540                 m_vRoadSegmentsNeg.push_back(pGr);
541         }
542         passIDMParameters(pGr);
543 }
```

Here is the call graph for this function:



4.47.4 Member Data Documentation

4.47.4.1 bool Road::m_DriveOnRight [private]

Definition at line 64 of file Road.h.

Referenced by init(), populate(), and setDriveOnRight().

4.47.4.2 int Road::m_PercentComplete [private]

Definition at line 65 of file Road.h.

Referenced by getPercentComplete(), and populate().

4.47.4.3 CIDMParameterSet* Road::m_pIDMParams_Car [private]

Definition at line 72 of file Road.h.

Referenced by clear(), passIDMParameters(), Road(), setIDMDriverModel(), setIDMParams_Car(), and ~Road().

4.47.4.4 CIDMParameterSet* Road::m_pIDMParams_SmallTruck [private]

Definition at line 73 of file Road.h.

Referenced by clear(), passIDMParameters(), Road(), setIDMDriverModel(), setIDMParams_SmallTruck(), and ~Road().

4.47.4.5 CIDMParameterSet* Road::m_pIDMParams_LargeTruck [private]

Definition at line 74 of file Road.h.

Referenced by clear(), passIDMParameters(), Road(), setIDMDriverModel(), setIDMParams_LargeTruck(), and ~Road().

4.47.4.6 CIDMParameterSet* Road::m_pIDMParams_Crane [private]

Definition at line 75 of file Road.h.

Referenced by clear(), passIDMParameters(), Road(), setIDMDriverModel(), setIDMParams_Crane(), and ~Road().

4.47.4.7 CIDMParameterSet* Road::m_pIDMParams_Lowloader [private]

Definition at line 76 of file Road.h.

Referenced by clear(), passIDMParameters(), Road(), setIDMDriverModel(), setIDMParams_Lowloader(), and ~Road().

4.47.4.8 int Road::m_LocOutputDetectorDirPos [private]

Definition at line 78 of file Road.h.

Referenced by init(), and setLocOuputDetectorDirPos().

4.47.4.9 int Road::m_LocOutputDetectorDirNeg [private]

Definition at line 79 of file Road.h.

Referenced by init(), and setLocOuputDetectorDirNeg().

4.47.4.10 Direction Road::DirectionPos [private]

Definition at line 81 of file Road.h.

Referenced by clear(), getVehicles(), init(), and update().

4.47.4.11 Direction Road::DirectionNeg [private]

Definition at line 82 of file Road.h.

Referenced by clear(), getVehicles(), init(), and update().

4.47.4.12 int Road::m_TrafFileNoLanesDirPos [private]

Definition at line 84 of file Road.h.

Referenced by MapTrafLaneToSimLane(), and setTrafFileNoLanesDirPos().

4.47.4.13 int Road::m_TrafFileNoLanesDirNeg [private]

Definition at line 85 of file Road.h.

Referenced by MapTrafLaneToSimLane(), and setTrafFileNoLanesDirNeg().

4.47.4.14 FileHandler* Road::m_pFileHandler [private]

Definition at line 87 of file Road.h.

Referenced by clear(), init(), initTruckGroup(), populate(), and Road().

4.47.4.15 std::vector<Vehicle*> Road::m_BufferVehicles [private]

Definition at line 88 of file Road.h.

Referenced by clear(), init(), and populate().

4.47.4.16 std::vector<Lane*> Road::m_vLanes [private]

Definition at line 89 of file Road.h.

Referenced by clear(), init(), and populate().

4.47.4.17 std::vector<Detector*> Road::m_vDetectorsPos [private]

Definition at line 90 of file Road.h.

Referenced by clear(), init(), SetMetricDetFromStatDet(), and update().

4.47.4.18 std::vector<Detector*> Road::m_vDetectorsNeg [private]

Definition at line 91 of file Road.h.

Referenced by clear(), init(), SetMetricDetFromStatDet(), and update().

4.47.4.19 std::vector<int> Road::m_vLaneLengthsPos [private]

Definition at line 93 of file Road.h.

4.47.4.20 std::vector<int> Road::m_vLaneLengthsNeg [private]

Definition at line 94 of file Road.h.

4.47.4.21 std::vector<RoadSegment*> Road::m_vRoadSegmentsPos [private]

Definition at line 96 of file Road.h.

Referenced by clear(), init(), SetGradientSegment(), and SetSpeedLimitSegment().

4.47.4.22 std::vector<RoadSegment*> Road::m_vRoadSegmentsNeg [private]

Definition at line 97 of file Road.h.

Referenced by clear(), init(), SetGradientSegment(), and SetSpeedLimitSegment().

4.47.4.23 bool Road::m_bEndOfFile [private]

Definition at line 99 of file Road.h.

Referenced by populate(), Road(), and update().

4.47.4.24 bool Road::m_AllowLaneChanging [private]

Definition at line 100 of file Road.h.

Referenced by getAllowLaneChanging(), init(), and setAllowLaneChanging().

4.47.4.25 int Road::m_NoLanesDirPos [private]

Definition at line 101 of file Road.h.

Referenced by getVehicles(), init(), and setNoLanesDirPos().

4.47.4.26 int Road::m_NoLanesDirNeg [private]

Definition at line 102 of file Road.h.

Referenced by getVehicles(), init(), and setNoLanesDirNeg().

4.47.4.27 int Road::m_NoDirections [private]

Definition at line 103 of file Road.h.

Referenced by clear(), getVehicles(), init(), setNoDirections(), and update().

4.47.4.28 int Road::m_NoLanes [private]

Definition at line 104 of file Road.h.

Referenced by MapTrafLaneToSimLane(), and setNoLanes().

4.47.4.29 int Road::m_RoadLength [private]

Definition at line 105 of file Road.h.

Referenced by getLength(), init(), populate(), SetGradientSegment(), setLocOuputDetectorDirNeg(), SetMetricDetFromStatDet(), setRoadLength(), and SetSpeedLimitSegment().

4.47.4.30 double Road::MIN_SPACE_FOR_NEXT_VEHICLE [private]

Definition at line 107 of file Road.h.

Referenced by populate(), and Road().

The documentation for this class was generated from the following files:

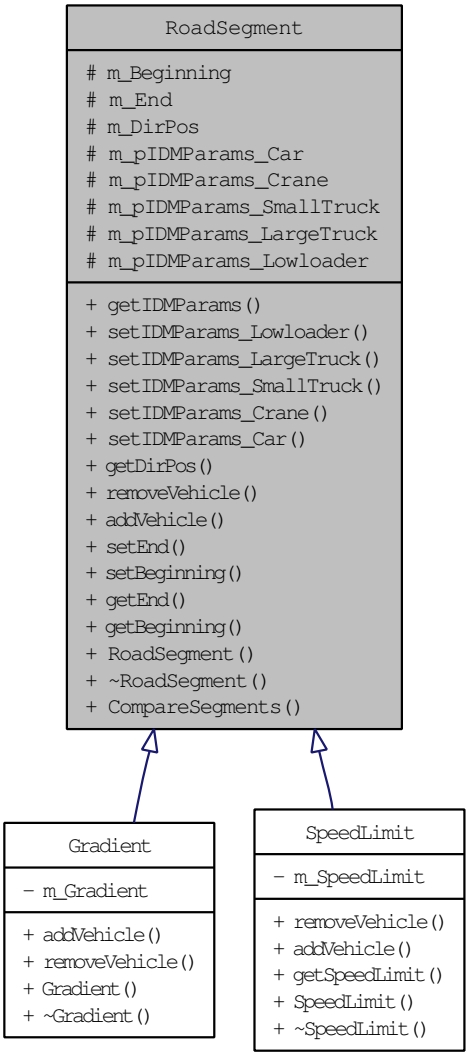
- D:/~Research/Code/C++/EvolveTraffic/Road.h
- D:/~Research/Code/C++/EvolveTraffic/Road.cpp

4.48 RoadSegment Class Reference

A base class from which specific special road segments are derived.

`#include <RoadSegment.h>`

Inheritance diagram for RoadSegment:



Collaboration diagram for RoadSegment:

Public Member Functions

- CIDMParameterSet * getIDMParams (WORD veh_id)
- void setIDMParams_Lowloader (CIDMParameterSet *Params)
- void setIDMParams_LargeTruck (CIDMParameterSet *Params)
- void setIDMParams_SmallTruck (CIDMParameterSet *Params)
- void setIDMParams_Crane (CIDMParameterSet *Params)
- void setIDMParams_Car (CIDMParameterSet *Params)
- bool getDirPos ()
- virtual void removeVehicle (Vehicle *pVeh)
- virtual void addVehicle (Vehicle *pVeh)

- void setEnd (int end)
- void setBeginning (int beginning)
- int getEnd () const
- int getBeginning () const
- RoadSegment ()
- virtual ~RoadSegment ()

Static Public Member Functions

- static bool CompareSegments (const RoadSegment *a, const RoadSegment *b)

  A function to compare RoadSegments.

Protected Attributes

- int m_Beginning
- int m_End
- bool m_DirPos
- CIDMParameterSet * m_pIDMParams_Car
- CIDMParameterSet * m_pIDMParams_Crane
- CIDMParameterSet * m_pIDMParams_SmallTruck
- CIDMParameterSet * m_pIDMParams_LargeTruck
- CIDMParameterSet * m_pIDMParams_Lowloader

4.48.1 Detailed Description

A base class from which specific special road segments are derived.

Definition at line 20 of file RoadSegment.h.

4.48.2 Constructor & Destructor Documentation

4.48.2.1 RoadSegment::RoadSegment ()

Definition at line 18 of file RoadSegment.cpp.

```
19 {
20
21 }
```

4.48.2.2 RoadSegment::~RoadSegment () [virtual]

Definition at line 23 of file RoadSegment.cpp.

```
24 {
25
26 }
```

4.48.3 Member Function Documentation

4.48.3.1 CIDMParameterSet * RoadSegment::getIDMParams (WORD *veh_id*)

Definition at line 88 of file RoadSegment.cpp.

References m_pIDMParams_Car, m_pIDMParams_Crane, m_pIDMParams_-LargeTruck, m_pIDMParams_Lowloader, m_pIDMParams_SmallTruck, VEH_ID_-CAR, VEH_ID_CRANE, VEH_ID_LARGETRUCK, VEH_ID_LOWLOADER, and VEH_ID_SMALLTRUCK.

Referenced by SpeedLimit::addVehicle().

```
89 {
90         switch(veh_id)
91         {
92                 case VEH_ID_CAR:
93                         return m_pIDMParams_Car;
94                 case VEH_ID_SMALLTRUCK:
95                         return m_pIDMParams_SmallTruck;
96                 case VEH_ID_LARGETRUCK:
97                         return m_pIDMParams_LargeTruck;
98                 case VEH_ID_CRANE:
99                         return m_pIDMParams_Crane;
100                 case VEH_ID_LOWLOADER:
101                         return m_pIDMParams_Lowloader;
102                 default:
103                         return m_pIDMParams_Car;
104         }
105 }
```

4.48.3.2 void RoadSegment::setIDMParams_Lowloader (CIDMParameterSet * *Params*)

Definition at line 83 of file RoadSegment.cpp.

References m_pIDMParams_Lowloader.

Referenced by Road::passIDMParameters().

```
84 {
85         m_pIDMParams_Lowloader = Params;
86 }
```

4.48.3.3 void RoadSegment::setIDMParams_LargeTruck (CIDMParameterSet * *Params*)

Definition at line 78 of file RoadSegment.cpp.

References m_pIDMParams_LargeTruck.

Referenced by Road::passIDMParameters().

```
79 {
80         m_pIDMParams_LargeTruck = Params;
81 }
```

4.48.3.4 void RoadSegment::setIDMParams_SmallTruck (CIDMParameterSet * *Params*)

Definition at line 73 of file RoadSegment.cpp.

References m_pIDMParams_SmallTruck.

Referenced by Road::passIDMParameters().

```
74 {
75         m_pIDMParams_SmallTruck = Params;
76 }
```

4.48.3.5 void RoadSegment::setIDMParams_Crane (CIDMParameterSet * *Params*)

Definition at line 68 of file RoadSegment.cpp.

References m_pIDMParams_Crane.

Referenced by Road::passIDMParameters().

```
69 {
70         m_pIDMParams_Crane = Params;
71 }
```

4.48.3.6 void RoadSegment::setIDMParams_Car (CIDMParameterSet * *Params*)

Definition at line 63 of file RoadSegment.cpp.

References m_pIDMParams_Car.

Referenced by Road::passIDMParameters().

```
64 {
65         m_pIDMParams_Car = Params;
66 }
```

4.48.3.7 bool RoadSegment::getDirPos ()

Definition at line 58 of file RoadSegment.cpp.

References m_DirPos.

```
59 {
60         return m_DirPos;
61 }
```

4.48.3.8 bool RoadSegment::CompareSegments (const RoadSegment * *a*, const RoadSegment * *b*) [static]

A function to compare RoadSegments.

Parameters:

- ***a*** The first RoadSegment
- ***b*** The second RoadSegment

Returns:

Whether the first segment's starting point is less than the second.

Definition at line 114 of file RoadSegment.cpp.

References getBeginning().

Referenced by Road::init().

```
115 {
116         int startA = a->getBeginning();
117         int startB = b->getBeginning();
118         return startA < startB;
119 }
```

Here is the call graph for this function:

4.48.3.9 void RoadSegment::removeVehicle (Vehicle * *pVeh*) [virtual]

Reimplemented in Gradient, and SpeedLimit.

Definition at line 53 of file RoadSegment.cpp.

```
54 {
55
56 }
```

4.48.3.10 void RoadSegment::addVehicle (Vehicle * *pVeh*) [virtual]

Reimplemented in Gradient, and SpeedLimit.

Definition at line 48 of file RoadSegment.cpp.

```
49 {
50
51 }
```

4.48.3.11 void RoadSegment::setEnd (int *end*)

Definition at line 43 of file RoadSegment.cpp.

References m_End.

```
44 {
45         m_End = end;
46 }
```

4.48.3.12 void RoadSegment::setBeginning (int *beginning*)

Definition at line 38 of file RoadSegment.cpp.

References m_Beginning.

```
39 {
40         m_Beginning = beginning;
41 }
```

4.48.3.13 int RoadSegment::getEnd () const

Definition at line 33 of file RoadSegment.cpp.

References m_End.

```
34 {
35         return m_End;
36 }
```

4.48.3.14 int RoadSegment::getBeginning () const

Definition at line 28 of file RoadSegment.cpp.

References m_Beginning.

Referenced by CompareSegments().

```
29 {
30         return m_Beginning;
31 }
```

4.48.4 Member Data Documentation

4.48.4.1 int RoadSegment::m_Beginning [protected]

Definition at line 43 of file RoadSegment.h.

Referenced by getBeginning(), Gradient::Gradient(), setBeginning(), and SpeedLimit::SpeedLimit().

4.48.4.2 int RoadSegment::m_End [protected]

Definition at line 44 of file RoadSegment.h.

Referenced by getEnd(), Gradient::Gradient(), setEnd(), and SpeedLimit::SpeedLimit().

4.48.4.3 bool RoadSegment::m_DirPos [protected]

Definition at line 45 of file RoadSegment.h.

Referenced by getDirPos(), Gradient::Gradient(), and SpeedLimit::SpeedLimit().

4.48.4.4 CIDMParameterSet* RoadSegment::m_pIDMParams_Car [protected]

Definition at line 46 of file RoadSegment.h.

Referenced by getIDMParams(), and setIDMParams_Car().

4.48.4.5 CIDMParameterSet* RoadSegment::m_pIDMParams_Crane [protected]

Definition at line 47 of file RoadSegment.h.

Referenced by getIDMParams(), and setIDMParams_Crane().

4.48.4.6 CIDMParameterSet* RoadSegment::m_pIDMParams_SmallTruck [protected]

Definition at line 48 of file RoadSegment.h.

Referenced by getIDMParams(), and setIDMParams_SmallTruck().

4.48.4.7 CIDMParameterSet* RoadSegment::m_pIDMParams_LargeTruck [protected]

Definition at line 49 of file RoadSegment.h.

Referenced by getIDMParams(), and setIDMParams_LargeTruck().

4.48.4.8 CIDMParameterSet* RoadSegment::m_pIDMParams_Lowloader [protected]

Definition at line 50 of file RoadSegment.h.

Referenced by getIDMParams(), and setIDMParams_Lowloader().

The documentation for this class was generated from the following files:

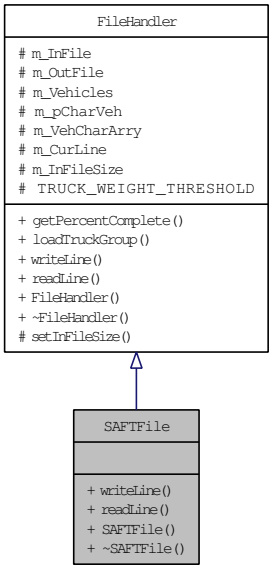
- D:/~Research/Code/C++/EvolveTraffic/RoadSegment.h
- D:/~Research/Code/C++/EvolveTraffic/RoadSegment.cpp

4.49 SAFTFile Class Reference

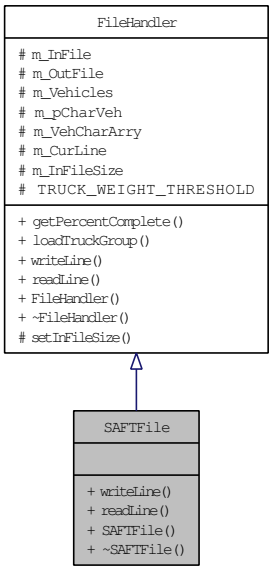
A class for reading from, and writing to, SAFT format files.

`#include <SAFTFile.h>`

Inheritance diagram for SAFTFile:



Collaboration diagram for SAFTFile:



Public Member Functions

- void writeLine (Vehicle *pVeh)
  Writes a line representing a truck to a SAFT file.
- Vehicle * readLine ()
  Reads a line representing a truck from a SAFT file.
- SAFTFile (std::string inputFile, std::string outputFile)
  Constructor.
- virtual ~SAFTFile ()
  Default destructor.

4.49.1 Detailed Description

A class for reading from, and writing to, SAFT format files.

Definition at line 17 of file SAFTFile.h.

4.49.2 Constructor & Destructor Documentation

4.49.2.1 SAFTFile::SAFTFile (std::string *inputFile*, std::string *outputFile*)

Constructor.

Parameters:

inputFile The file to read input from

outputFile The file to write output to

Definition at line 25 of file SAFTFile.cpp.

References FileHandler::m_InFile, FileHandler::m_OutFile, FileHandler::m_pCharVeh, FileHandler::m_VehCharArry, and FileHandler::setInFileSize().

```
26 {
27         m_InFile.open(inputFile.c_str(), std::ios::in);
28         setInFileSize();
29
30         m_OutFile.open(outputFile.c_str(), std::ios::out);
31         m_pCharVeh = m_VehCharArry;
32 }
```

Here is the call graph for this function:

4.49.2.2 SAFTFile::~SAFTFile () [virtual]

Default destructor.

Definition at line 35 of file SAFTFile.cpp.

References FileHandler::m_InFile, FileHandler::m_OutFile, and FileHandler::m_-Vehicles.

```
36 {
37         m_InFile.close();
38         m_OutFile.flush();
39
40         m_Vehicles.clear();
41 }
```

4.49.3 Member Function Documentation

4.49.3.1 void SAFTFile::writeLine (Vehicle * *pVeh*) [virtual]

Writes a line representing a truck to a SAFT file.

Parameters:

pVeh The vehicle to write to file

This function takes a vehicle as a parameter, and calls this vehicle's doSAFTDATA function, in order to obtain a string which represents all of the vehicle's physical properties. It then writes this string to a SAFT format file

Implements FileHandler.

Definition at line 86 of file SAFTFile.cpp.

References FileHandler::m_OutFile, FileHandler::m_pCharVeh, and Vehicle::writeSAFTData().

```
87 {
88         pVeh->writeSAFTData(m_pCharVeh);
89         m_OutFile << m_pCharVeh << '\n';
90 }
```

Here is the call graph for this function:

4.49.3.2 Vehicle * SAFTFile::readLine () [virtual]

Reads a line representing a truck from a SAFT file.

Returns:

A newly created truck

This function reads the next line available from a SAFT format file and stores the information in a string. If the string is not null, a truck is created with the properties specified by the line from the file, and the truck is returned

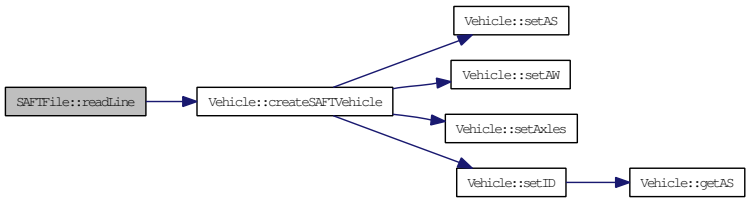
Implements FileHandler.

Definition at line 52 of file SAFTFile.cpp.

References Vehicle::createSAFTVehicle(), KG100_TO_KN, FileHandler::m_CurLine, FileHandler::m_InFile, and FileHandler::TRUCK_WEIGHT_THRESHOLD.

```
53 {
54         std::string str;
55
56         std::getline(m_InFile, str, '\n');
57
58         if(str != "")
59         {
60                 m_CurLine++;
61                 Vehicle* pVeh;
62                 double GVW = atoi( str.substr(27,4).c_str() ) * KG100_TO_KN;     // convert kg/10
63
64                 if(GVW >= TRUCK_WEIGHT_THRESHOLD)
65                         pVeh = new Truck();
66
67                 else
68                         pVeh = new Car();
69
70                 pVeh->createSAFTVehicle(str);
71                 return pVeh;
72         }
73
74         return NULL;
75 }
```

Here is the call graph for this function:



The documentation for this class was generated from the following files:

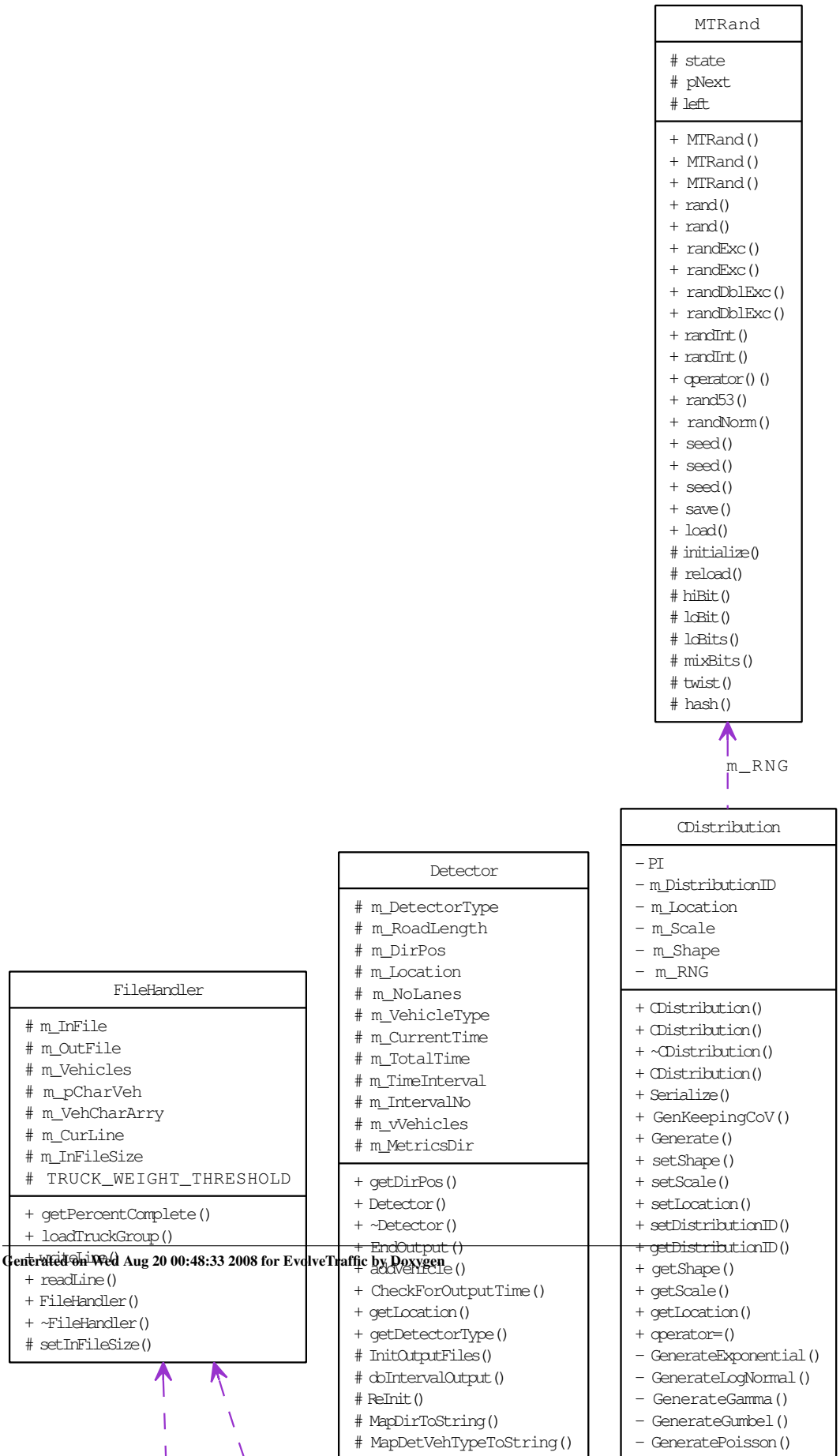
- D:/~Research/Code/C++/EvolveTraffic/SAFTFile.h
- D:/~Research/Code/C++/EvolveTraffic/SAFTFile.cpp

4.50 Sim Class Reference

A class to represent the simulation handler.

`#include <Sim.h>`

Collaboration diagram for Sim:



Public Member Functions

- void clear ()

  Clears the sim so it can be used in another simulation.

- bool doFullSimulation ()

  Does an entire simulation.

- void setFileIn (string file)

  Sets the file to read input from.

- void setFileOut (string file)

  Sets the file to write output to.

- void setFileType (WORD fileType)

  Sets the file type of the input.

- void setNoLanesDirPos (int nlpos)

  Sets the number of lanes in the positive direction.

- void setNoLanesDirNeg (int nlneg)

  Sets the number of lanes in the negative direction.

- void setSimTimeStep (double ts)

  Sets the size of the timestep.

- void setNoDirections (int nd)

  Sets the number of directions.

- void setNoLanes (int nl)

  Sets the number of lanes.

- void setRoadLength (int L)

  Sets the length of the road.

- double getCurrentSimTime ()

  Gets the current time of the simulation.

- int getPercentComplete ()
- Road * getRoad ()

  Gets the simulation's road object.

- double getRoadLength ()

  Gets the length of the road.

- M2D getPositions ()

Gets all the vehicles on the road.

- M2D doOneTimeStep (bool *pInSimulation)
  Steps the simulation forward by one timestep.
- void init ()
  Initialises the simulation.
- Sim ()
  Default Constructor.
- virtual ~Sim ()
  Default Destructor.

Private Attributes

- double m_CurrentSimTime
- Road m_Road
- M2D m_vVehicles
- string m_FileIn
- string m_FileOut
- WORD m_FileType
- int m_NoLanesDirPos
- int m_NoLanesDirNeg
- double m_SimTimeStep
- int m_NoDirections
- int m_NoLanes
- int m_RoadLength
- int m_PercentComplete

4.50.1 Detailed Description

A class to represent the simulation handler.

Definition at line 18 of file Sim.h.

4.50.2 Constructor & Destructor Documentation

4.50.2.1 Sim::Sim ()

Default Constructor.

Definition at line 6 of file Sim.cpp.

References m_CurrentSimTime.

```
7 {
8         m_CurrentSimTime = 0.0;
9 }
```

4.50.2.2 Sim::~Sim () [virtual]

Default Destructor.

Definition at line 12 of file Sim.cpp.

```
13 {
14
15 }
```

4.50.3 Member Function Documentation

4.50.3.1 void Sim::clear ()

Clears the sim so it can be used in another simulation.

This function clears all of the sim's information so that it may be used in another simulation without re-instanciating.

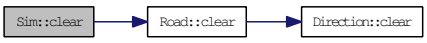
Definition at line 204 of file Sim.cpp.

References Road::clear(), m_CurrentSimTime, and m_Road.

Referenced by CEvolveTrafficView::doSimFinished().

```
205 {
206         m_CurrentSimTime = 0.0;
207         m_Road.clear();
208 }
```

Here is the call graph for this function:



4.50.3.2 bool Sim::doFullSimulation ()

Does an entire simulation.

Returns:

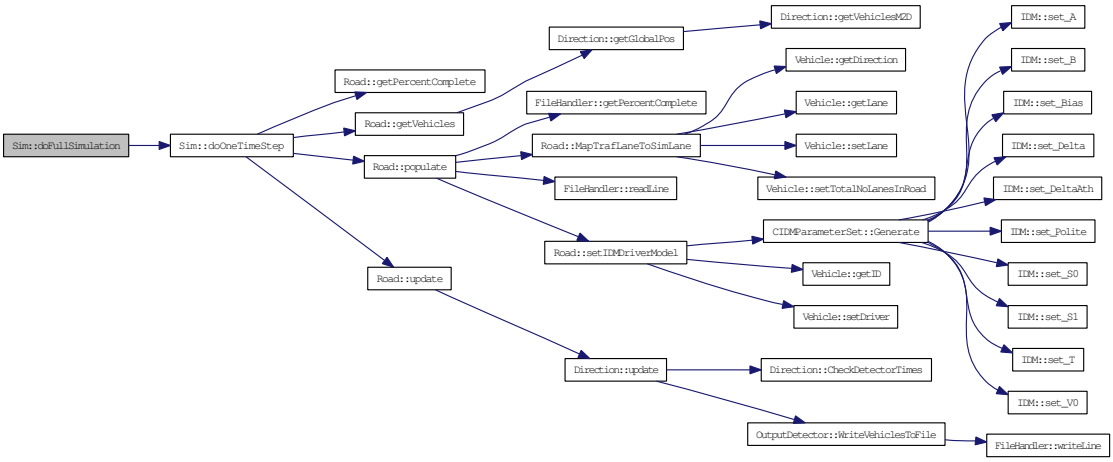
Whether the sim has finished

Definition at line 64 of file Sim.cpp.

References doOneTimeStep().

```
65 {
66         bool InSim = true;
67         while(InSim)
68                 doOneTimeStep(&InSim);  // InSim becomes false when sim is over
69
70         return false;                           // so we know when it's finished
71 }
```

Here is the call graph for this function:



4.50.3.3 void Sim::setFileIn (string *file*)

Sets the file to read input from.

Parameters:

file The file to read input from

Definition at line 103 of file Sim.cpp.

References m_FileIn.

Referenced by CEvolveTrafficDoc::initSim().

```
104 {
105         m_FileIn = file;
106 }
```

4.50.3.4 void Sim::setFileOut (string *file*)

Sets the file to write output to.

Parameters:

file The file to write output to

Definition at line 112 of file Sim.cpp.

References m_FileOut.

Referenced by CEvolveTrafficDoc::initSim().

```
113 {
114         m_FileOut = file;
115 }
```

4.50.3.5 void Sim::setFileType (WORD *fileType*)

Sets the file type of the input.

Parameters:

fileType The file type of the input

Definition at line 121 of file Sim.cpp.

References m_FileType.

Referenced by CEvolveTrafficDoc::initSim().

```
122 {
123         m_FileType = fileType;
124 }
```

4.50.3.6 void Sim::setNoLanesDirPos (int *nlpos*)

Sets the number of lanes in the positive direction.

Parameters:

nlpos The number of lanes in the positive direction

Definition at line 130 of file Sim.cpp.

References m_NoLanesDirPos.

Referenced by CEvolveTrafficDoc::initSim().

```
131 {
132         m_NoLanesDirPos = nlpos;
133 }
```

4.50.3.7 void Sim::setNoLanesDirNeg (int *nlneg*)

Sets the number of lanes in the negative direction.

Parameters:

nlneg The number of lanes in the negative direction

Definition at line 139 of file Sim.cpp.

References m_NoLanesDirNeg.

Referenced by CEvolveTrafficDoc::initSim().

```
140 {
141         m_NoLanesDirNeg = nlneg;
142 }
```

4.50.3.8 void Sim::setSimTimeStep (double *ts*)

Sets the size of the timestep.

Parameters:

ts The size of the timestep

Definition at line 148 of file Sim.cpp.

References m_SimTimeStep.

Referenced by CEvolveTrafficDoc::initSim().

```
149 {
150         m_SimTimeStep = ts;
151 }
```

4.50.3.9 void Sim::setNoDirections (int *nd*)

Sets the number of directions.

Parameters:

nd The number of directions

Definition at line 157 of file Sim.cpp.

References m_NoDirections.

Referenced by CEvolveTrafficDoc::initSim().

```
158 {
159         m_NoDirections = nd;
160 }
```

4.50.3.10 void Sim::setNoLanes (int *nl*)

Sets the number of lanes.

Parameters:

nl The number of lanes

Definition at line 166 of file Sim.cpp.

References m_NoLanes.

Referenced by CEvolveTrafficDoc::initSim().

```
167 {
168         m_NoLanes = nl;
169 }
```

4.50.3.11 void Sim::setRoadLength (int *L*)

Sets the length of the road.

Parameters:

L The length of the road

Definition at line 175 of file Sim.cpp.

References m_Road, m_RoadLength, and Road::setRoadLength().

Referenced by CEvolveTrafficDoc::initSim().

```
176 {
177         m_RoadLength = L;
178         m_Road.setRoadLength(L);
179 }
```

Here is the call graph for this function:

4.50.3.12 double Sim::getCurrentSimTime ()

Gets the current time of the simulation.

Returns:

The current time of the simulation

Definition at line 185 of file Sim.cpp.

References m_CurrentSimTime.

Referenced by CEvolveTrafficView::DrawTimer(), and CEvolveTrafficView::OnRunInvisible().

```
186 {
187         return m_CurrentSimTime;
188 }
```

4.50.3.13 int Sim::getPercentComplete ()

Definition at line 82 of file Sim.cpp.

References m_PercentComplete.

Referenced by CEvolveTrafficView::UpdateProgressBar().

```
83 {
84         return m_PercentComplete;
85 }
```

4.50.3.14 Road * Sim::getRoad ()

Gets the simulation's road object.

Returns:

The simulation's road object

Definition at line 194 of file Sim.cpp.

References m_Road.

Referenced by CEvolveTrafficDoc::initSim().

```
195 {
196          return &m_Road;
197 }
```

4.50.3.15 double Sim::getRoadLength ()

Gets the length of the road.

Returns:

The length of the road

Definition at line 91 of file Sim.cpp.

References Road::getLength(), and m_Road.

```
92 {
93         return m_Road.getLength();
94 }
```

Here is the call graph for this function:

4.50.3.16 M2D Sim::getPositions ()

Gets all the vehicles on the road.

Returns:

All the vehicles on the road

Definition at line 77 of file Sim.cpp.

References m_vVehicles.

```
78 {
79         return m_vVehicles;
80 }
```

4.50.3.17 M2D Sim::doOneTimeStep (bool * *pInSimulation*)

Steps the simulation forward by one timestep.

Parameters:

pInSimulation Whether or not the simulation is still running

Returns:

All of the vehicles currently on the Road

This function is called while the simulation is running. It firstly populates the road each step, by calling Road.populate(), and then updates each vehicle on the road by calling Road.update(). In addition to this, the function return all of the vehicles currently on the road.

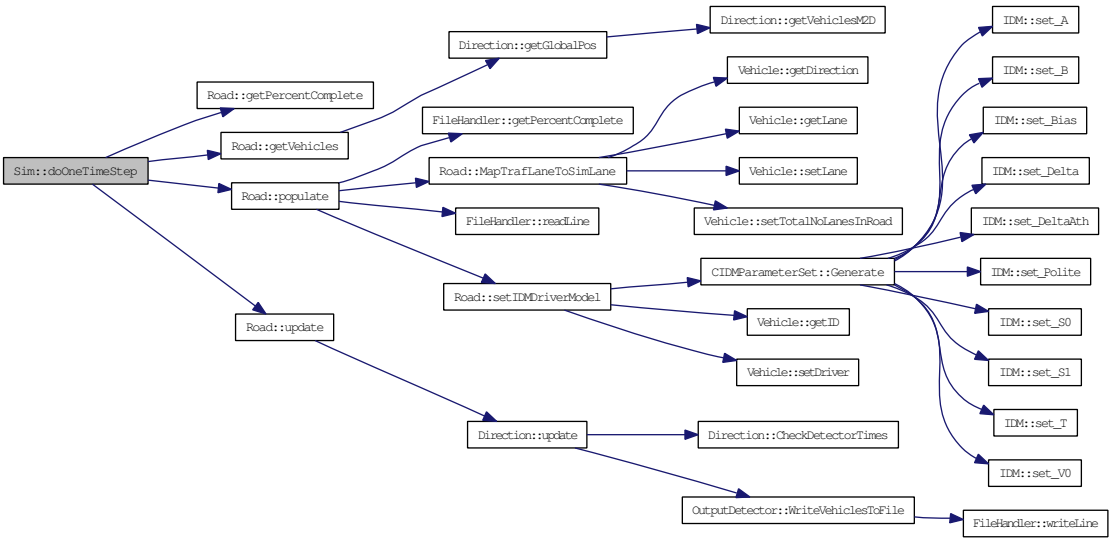
Definition at line 45 of file Sim.cpp.

References Road::getPercentComplete(), Road::getVehicles(), m_CurrentSimTime, m_PercentComplete, m_Road, m_SimTimeStep, m_vVehicles, Road::populate(), and Road::update().

Referenced by doFullSimulation(), CEvolveTrafficView::doLoop(), and CEvolveTrafficView::OnRunInvisible().

```
46 {
47         m_Road.populate(m_SimTimeStep,m_CurrentSimTime);
48 
49         bool SimOver = m_Road.update(m_SimTimeStep, m_CurrentSimTime);
50         if(SimOver)
51                 (*pInSimulation) = false;       // we're no longer in a simulation
52 
53         m_vVehicles = m_Road.getVehicles();
54         m_CurrentSimTime += m_SimTimeStep;
55         m_PercentComplete = m_Road.getPercentComplete();
56 
57         return m_vVehicles;
58 }
```

Here is the call graph for this function:



4.50.3.18 void Sim::init ()

Initialises the simulation.

This function takes data which has been passed from the Graphical User Interface, and initialises the Road object with this data.

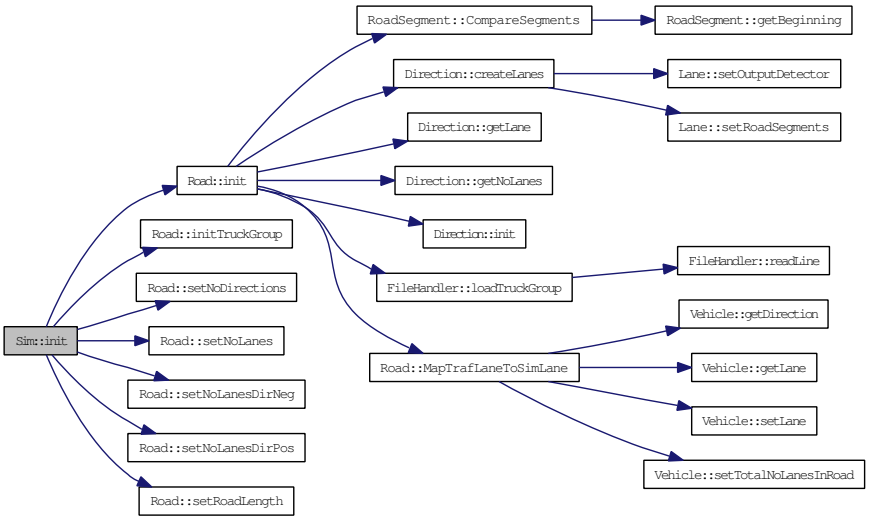
Definition at line 22 of file Sim.cpp.

References Road::init(), Road::initTruckGroup(), m_CurrentSimTime, m_FileIn, m_FileOut, m_FileType, m_NoDirections, m_NoLanes, m_NoLanesDirNeg, m_NoLanesDirPos, m_Road, m_RoadLength, Road::setNoDirections(), Road::setNoLanes(), Road::setNoLanesDirNeg(), Road::setNoLanesDirPos(), and Road::setRoadLength().

Referenced by CEvolveTrafficDoc::initSim().

```
23 {
24         m_Road.setNoDirections(m_NoDirections);
25         m_Road.setNoLanes(m_NoLanes);
26         m_Road.setNoLanesDirPos(m_NoLanesDirPos);
27         m_Road.setNoLanesDirNeg(m_NoLanesDirNeg);
28         m_Road.setRoadLength(m_RoadLength);
29
30         m_Road.initTruckGroup(m_FileIn, m_FileOut, m_FileType);
31         m_CurrentSimTime = m_Road.init();
32 }
```

Here is the call graph for this function:



4.50.4 Member Data Documentation

4.50.4.1 double Sim::m_CurrentSimTime [private]

Definition at line 46 of file Sim.h.

Referenced by clear(), doOneTimeStep(), getCurrentSimTime(), init(), and Sim().

4.50.4.2 Road Sim::m_Road [private]

Definition at line 47 of file Sim.h.

Referenced by clear(), doOneTimeStep(), getRoad(), getRoadLength(), init(), and setRoadLength().

4.50.4.3 M2D Sim::m_vVehicles [private]

Definition at line 48 of file Sim.h.

Referenced by doOneTimeStep(), and getPositions().

4.50.4.4 string Sim::m_FileIn [private]

Definition at line 49 of file Sim.h.

Referenced by init(), and setFileIn().

4.50.4.5 string Sim::m_FileOut [private]

Definition at line 50 of file Sim.h.

Referenced by init(), and setFileOut().

4.50.4.6 WORD Sim::m_FileType [private]

Definition at line 51 of file Sim.h.

Referenced by init(), and setFileType().

4.50.4.7 int Sim::m_NoLanesDirPos [private]

Definition at line 52 of file Sim.h.

Referenced by init(), and setNoLanesDirPos().

4.50.4.8 int Sim::m_NoLanesDirNeg [private]

Definition at line 53 of file Sim.h.

Referenced by init(), and setNoLanesDirNeg().

4.50.4.9 double Sim::m_SimTimeStep [private]

Definition at line 54 of file Sim.h.

Referenced by doOneTimeStep(), and setSimTimeStep().

4.50.4.10 int Sim::m_NoDirections [private]

Definition at line 55 of file Sim.h.

Referenced by init(), and setNoDirections().

4.50.4.11 int Sim::m_NoLanes [private]

Definition at line 56 of file Sim.h.

Referenced by init(), and setNoLanes().

4.50.4.12 int Sim::m_RoadLength [private]

Definition at line 57 of file Sim.h.

Referenced by init(), and setRoadLength().

4.50.4.13 int Sim::m_PercentComplete [private]

Definition at line 58 of file Sim.h.

Referenced by doOneTimeStep(), and getPercentComplete().

The documentation for this class was generated from the following files:

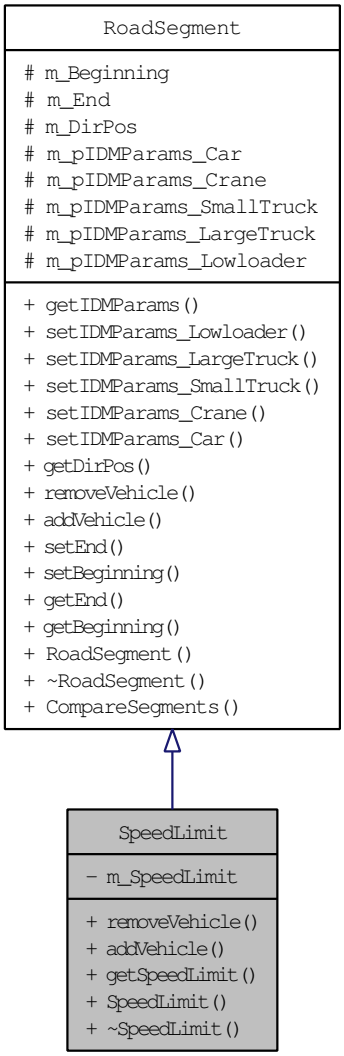
- D:/~Research/Code/C++/EvolveTraffic/Sim.h
- D:/~Research/Code/C++/EvolveTraffic/Sim.cpp

4.51 SpeedLimit Class Reference

A derived class to represent a speed-limit section on a road.

`#include <SpeedLimit.h>`

Inheritance diagram for SpeedLimit:



Collaboration diagram for SpeedLimit:

Public Member Functions

- void removeVehicle (Vehicle *pVeh)

  Removes a vehicle from a speed limit.

- void addVehicle (Vehicle *pVeh)

  Adds a vehicle to the speed limit.

- double getSpeedLimit ()
- SpeedLimit (int start, int end, double limit, bool DirPos)

  Constructor.

- virtual ~SpeedLimit ()
  Destructor.

Private Attributes

- double m_SpeedLimit
  The speed limit for this segment.

4.51.1 Detailed Description

A derived class to represent a speed-limit section on a road.

Definition at line 17 of file SpeedLimit.h.

4.51.2 Constructor & Destructor Documentation

4.51.2.1 SpeedLimit::SpeedLimit (int *start*, int *end*, double *limit*, bool *DirPos*)

Constructor.

Definition at line 19 of file SpeedLimit.cpp.

References RoadSegment::m_Beginning, RoadSegment::m_DirPos, RoadSegment::m_End, and m_SpeedLimit.

```
20 {
21         m_Beginning             = start;
22         m_End                   = end;
23         m_DirPos                = DirPos;
24         m_SpeedLimit    = limit;
25 }
```

4.51.2.2 SpeedLimit::~SpeedLimit () [virtual]

Destructor.

Definition at line 28 of file SpeedLimit.cpp.

```
29 {
30
31 }
```

4.51.3 Member Function Documentation

4.51.3.1 void SpeedLimit::removeVehicle (Vehicle * *pVeh*) [virtual]

Removes a vehicle from a speed limit.

Parameters:

pVeh The Vehicle to remove

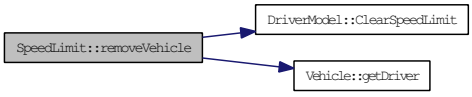
Reimplemented from RoadSegment.

Definition at line 75 of file SpeedLimit.cpp.

References DriverModel::ClearSpeedLimit(), and Vehicle::getDriver().

```
76 {
77         DriverModel* driver = pVeh->getDriver();
78         driver->ClearSpeedLimit();
79 }
```

Here is the call graph for this function:



4.51.3.2 void SpeedLimit::addVehicle (Vehicle * *pVeh*) [virtual]

Adds a vehicle to the speed limit.

Parameters:

pVeh The vehicle to add

This function adds a vehicle to a speed limit segment in the road. A new desired acceleration for the vehicle is created based on the type of vehicle. If, however, the vehicle's desired acceleration is below that of the speed limit, then the desired acceleration does not change

Reimplemented from RoadSegment.

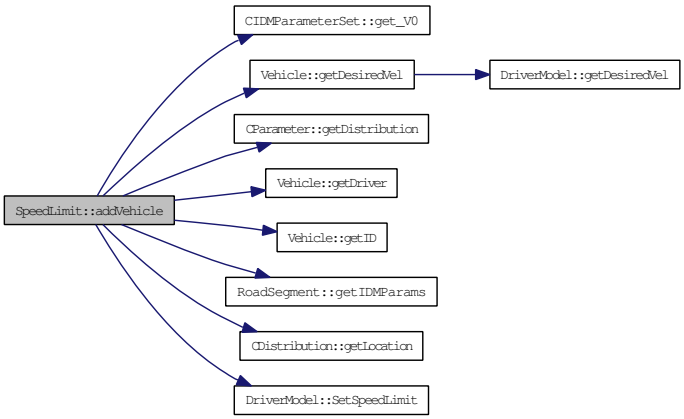
Definition at line 47 of file SpeedLimit.cpp.

References CIDMParameterSet::get_V0(), Vehicle::getDesiredVel(), CParameter::getDistribution(), Vehicle::getDriver(), Vehicle::getID(), RoadSegment::getIDMParams(), CDistribution::getLocation(), KM_PER_H_TO_M_PER_S, m_SpeedLimit, and DriverModel::SetSpeedLimit().

```
48 {
49         double curV0 = pVeh->getDesiredVel();
50         double newV0 = curV0;   // set limit to current V0 if curV0 < limit
51
52         // do we need to generate a new lower V0?
53         if(curV0 > m_SpeedLimit)
54         {
55                 WORD veh_id = pVeh->getID();
56                 CIDMParameterSet* paramSet = getIDMParams(veh_id);
57                 CParameter V0 = (*paramSet->get_V0());
```

```
58                  // Initial method - generate new V0
59                  //newV0 = V0.GenKeepingCoV(m_SpeedLimit)*KM_PER_H_TO_M_PER_S;
60                  // revised method - keep ratio of v0/location constant
61                  double meanV0 = (V0.getDistribution()->getLocation()*KM_PER_H_TO_M_PER_S);
62                  double v0ratio = curV0/meanV0;
63                  newV0 = v0ratio * m_SpeedLimit;
64                  //TRACE("***Speed limit - m_SpeedLimit: %f\tcurV0: %f\tmeanV0: %f\tv0ratio: %f\t
65          }
66          // this method allows for other driver models
67          DriverModel* driver = pVeh->getDriver();
68          driver->SetSpeedLimit(newV0);   // always call this function, even if not changing V0
69 }
```

Here is the call graph for this function:



4.51.3.3 double SpeedLimit::getSpeedLimit ()

Definition at line 33 of file SpeedLimit.cpp.

References m_SpeedLimit.

```
34 {
35          return m_SpeedLimit;
36 }
```

4.51.4 Member Data Documentation

4.51.4.1 double SpeedLimit::m_SpeedLimit [private]

The speed limit for this segment.

Definition at line 28 of file SpeedLimit.h.

Referenced by addVehicle(), getSpeedLimit(), and SpeedLimit().

The documentation for this class was generated from the following files:

- D:/~Research/Code/C++/EvolveTraffic/SpeedLimit.h
- D:/~Research/Code/C++/EvolveTraffic/SpeedLimit.cpp

4.52 Truck Class Reference

A class to represent a Truck.

`#include <Truck.h>`

Inheritance diagram for Truck:

Collaboration diagram for Truck:

Public Member Functions

- Truck ()

  Default constructor.

- virtual ~Truck ()

  Default destructor.

- void setAW (int i, double w)

  Sets the weight of a given axle.

- void setAS (int i, double s)
  Sets the spacing of a given axle.
- int getNoAxles ()
  Gets the vehicle's number of axles.
- double getAS (int i)
  Gets the spacing of a given axle.
- double getAW (int i)
  Gets the weight of a given axle.
- bool operator== (Truck &x)
- bool operator< (const Truck &x)
- void returnTruckData (int truckData[ ])
  Parses the Truck's data into an array.
- void doSAFTData (char *pTruck)
- void doCASTORData (char *pTruck)
- void createSAFTVehicle (std::string data)
  Creates a truck from a given SAFT string.
- void createCASTORVehicle (std::string data)
  Creates a truck from a given CASTOR string.
- void setDriver ()
- void setGVW (double weight)
  Sets the GVW of the vehicle to a given weight.

Private Member Functions

- int getID ()
  Gets the class of the Vehicle.

Private Attributes

- std::vector< Axle * > axles
- double m_OverhangFront
- double m_OverhangBack

4.52.1 Detailed Description

A class to represent a Truck.

Definition at line 22 of file Truck.h.

4.52.2 Constructor & Destructor Documentation

4.52.2.1 Truck::Truck ()

Default constructor.

Default Constructor.

Definition at line 15 of file Truck.cpp.

```
15              {
16 /*
17          HEAD    = 0;    DAY             = 0;    MONTH   = 0;    YEAR    = 5;    HOUR    = 0;
18          MIN             = 0;    SEC             = 0;    hNDT    = 0;    SPEED   = 0;    GVW
19          LENGTH  = 0;    NAXLES  = 0;    DIRN    = 0;    LANE    = 0;    TRNS    = 0;
20 */
21
22 }
```

4.52.2.2 Truck::~Truck () [virtual]

Default destructor.

Default Destructor.

Definition at line 25 of file Truck.cpp.

References Vehicle::m_vAxles.

```
26 {
27          for(int i = 0; i < m_vAxles.size(); i++)
28                  delete m_vAxles.at(i);
29          m_vAxles.clear();
30 }
```

4.52.3 Member Function Documentation

4.52.3.1 void Truck::setAW (int *i*, double *w*)

Sets the weight of a given axle.

Parameters:

i the axle to operate upon

w the desired weight

Reimplemented from Vehicle.

4.52.3.2 void Truck::setAS (int *i*, double *s*)

Sets the spacing of a given axle.

The space of a given axle to the next axle

Parameters:

i the axle to operate upon

s the desired spacing

Reimplemented from Vehicle.

4.52.3.3 int Truck::getNoAxles ()

Gets the vehicle's number of axles.

Returns:

the vehicle's number of axles

Reimplemented from Vehicle.

4.52.3.4 double Truck::getAS (int *i*)

Gets the spacing of a given axle.

The space of a given axle to the next axle

Parameters:

i the axle to operate upon

Returns:

the space between this axle and the next

Reimplemented from Vehicle.

4.52.3.5 double Truck::getAW (int *i*)

Gets the weight of a given axle.

Parameters:

i the axle to operate upon

Returns:

the weight of the axle

Reimplemented from Vehicle.

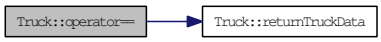
4.52.3.6 bool Truck::operator== (Truck & *x*)

Definition at line 65 of file Truck.cpp.

References returnTruckData().

```
66 {
67         const int N = 33;
68         int truck1[N]; int truck2[N];
69         returnTruckData(truck1);
70         x.returnTruckData(truck2);
71
72         bool same = true; int i = 0;
73         while(same && i < N)
74         {
75                 if(truck1[i] != truck2[i])
76                         same = false;
77                 i++;
78         }
79         return same;
80 }
```

Here is the call graph for this function:



4.52.3.7 bool Truck::operator< (const Truck & *x*)

Definition at line 58 of file Truck.cpp.

References Vehicle::getTime().

```
59 {
60         double a = this->getTime();
61         double b = x.getTime();
62         return a < b;
63 }
```

Here is the call graph for this function:

4.52.3.8 void Truck::returnTruckData (int *truckData*[ ])

Parses the Truck's data into an array.

Definition at line 33 of file Truck.cpp.

References Vehicle::m_Day, Vehicle::m_GVW, Vehicle::m_Head, Vehicle::m_-Hndt, Vehicle::m_Hour, Vehicle::m_intDir, Vehicle::m_Lane, Vehicle::m_Length, Vehicle::m_Min, Vehicle::m_Month, Vehicle::m_NoAxles, Vehicle::m_Sec, Vehicle::m_Trns, Vehicle::m_Velocity, and Vehicle::m_Year.

Referenced by operator==().

```
34 {
```

```
35
36          truckData[0]    = m_Head;
37          truckData[1]    = m_Day;
38          truckData[2]    = m_Month;
39          truckData[3]    = m_Year;
40          truckData[4]    = m_Hour;
41          truckData[5]    = m_Min;
42          truckData[6]    = m_Sec;
43          truckData[7]    = m_Hndt;
44          truckData[8]    = m_Velocity;   // must change to dm/s
45          truckData[9]    = m_GVW;                // change to kg/100
46          truckData[10]   = m_Length;             // change to dm
47          truckData[11]   = m_NoAxles;
48          truckData[12]   = m_intDir;             //m_DirPointsToRight == true ? 1 : 2;
49          truckData[13]   = m_Lane;
50          truckData[14]   = m_Trns;
51
52          for(int i = 0; i <= 33; i++)
53          {
54                  if( (truckData[i] < 0) || (truckData[i] > 9999) ) truckData[i] = 0;
55          }
56 }
```

4.52.3.9 void Truck::doSAFTData (char * *pTruck*)

4.52.3.10 void Truck::doCASTORData (char * *pTruck*)

4.52.3.11 void Truck::createSAFTVehicle (std::string *data*)

Creates a truck from a given SAFT string.

Parameters:

data The SAFT string of data

This function takes in a string from a SAFT file and creates a truck based on the properties supplied by the data string

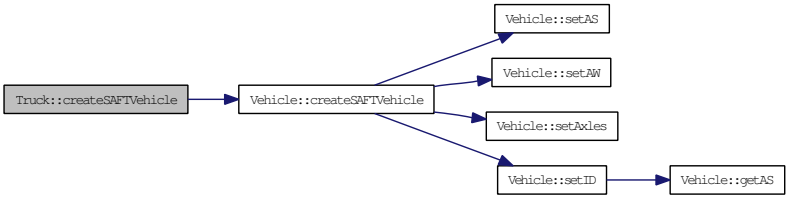
Reimplemented from Vehicle.

Definition at line 89 of file Truck.cpp.

References Vehicle::createSAFTVehicle(), m_OverhangBack, and m_OverhangFront.

```
90 {
91          Vehicle::createSAFTVehicle(data);
92          m_OverhangFront = 0.0;
93          m_OverhangBack = 0.0;
94 }
```

Here is the call graph for this function:



4.52.3.12 void Truck::createCASTORVehicle (std::string *data*)

Creates a truck from a given CASTOR string.

Parameters:

data The CASTOR string of data

This function takes in a string from a CASTOR file and creates a truck based on the properties supplied by the data string

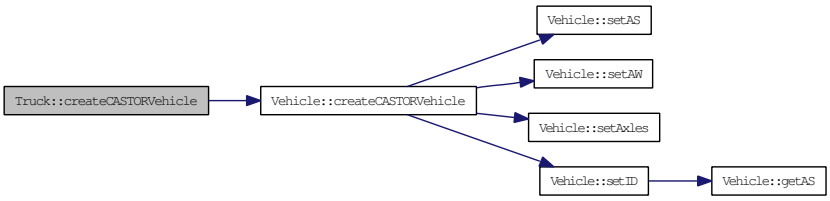
Reimplemented from Vehicle.

Definition at line 103 of file Truck.cpp.

References Vehicle::createCASTORVehicle(), m_OverhangBack, and m_-OverhangFront.

```
104 {
105         Vehicle::createCASTORVehicle(data);
106         m_OverhangFront = 0.0;
107         m_OverhangBack = 0.0;
108 }
```

Here is the call graph for this function:



4.52.3.13 void Truck::setDriver ()

4.52.3.14 void Truck::setGVW (double *weight*)

Sets the GVW of the vehicle to a given weight.

Parameters:

weight the desired GVW

Reimplemented from Vehicle.

4.52.3.15 int Truck::getID () [private]

Gets the class of the Vehicle.

Returns:

ID the Vehicle's class

Reimplemented from Vehicle.

4.52.4 Member Data Documentation

4.52.4.1 std::vector<Axle*> Truck::axles [private]

Definition at line 51 of file Truck.h.

4.52.4.2 double Truck::m_OverhangFront [private]

Definition at line 52 of file Truck.h.

Referenced by createCASTORVehicle(), and createSAFTVehicle().

4.52.4.3 double Truck::m_OverhangBack [private]

Definition at line 53 of file Truck.h.

Referenced by createCASTORVehicle(), and createSAFTVehicle().

The documentation for this class was generated from the following files:

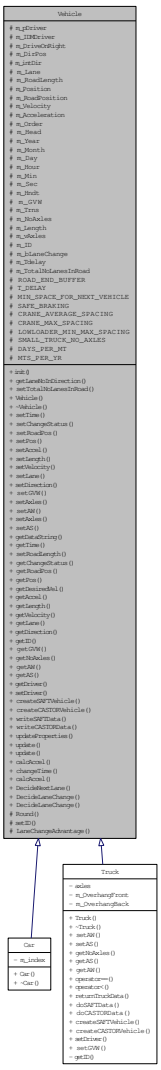
- D:/~Research/Code/C++/EvolveTraffic/Truck.h
- D:/~Research/Code/C++/EvolveTraffic/Truck.cpp

4.53 Vehicle Class Reference

A base class from which specific Vehicle types are derived.

`#include <Vehicle.h>`

Inheritance diagram for Vehicle:



Collaboration diagram for Vehicle:

Public Member Functions

- void init (int RoadLength, bool DriveOnRIght)
- int getLaneNoInDirection ()
- void setTotalNoLanesInRoad (int nLanes)
- Vehicle ()

  Default Constructor.

- virtual ~Vehicle ()

  Default Destructor.

- void setTime (double time)
- void setChangeStatus (bool stat)

  Sets whether or not the Vehicle wishes to change lanes.

- void setRoadPos (double r)

  Sets the position of the Vehicle on the road.

- void setPos (double p)

  Sets the position of the Vehicle to a given value.

- void setAccel (double acceler)

  Sets the acceleration of the Vehicle to a given value.

- void setLength (double length)

  Sets the length of the Vehicle to a given value.

- void setVelocity (double velocity)

  Sets the velocity of the Vehicle to a given value.

- void setLane (int l)

  Sets the lane that the Vehicle is in.

- void setDirection (bool DirPointsRight)

  Sets the direction of the Vehicle.

- void setGVW (double weight)

  Sets the GVW of the vehicle to a given weight.

- void setAxles ()

  Sets the vehicle's number of axles to a given number.

- void setAW (int i, double w)

  Sets the weight of a given axle.

- void setAxles (int axNo)
- void setAS (int i, double s)

  Sets the spacing of a given axle.

- CString getDataString ()
- virtual double getTime () const

  Gets the time of arrival of the vehicle in seconds.

- void setRoadLength (int length)
- bool getChangeStatus ()

  Gets whether or not the Vehicle wishes to change lane.

- double getRoadPos ()
  Gets the position of the Vehicle on the road.
- double getPos ()
  Gets the Vehicle's position.
- double getDesiredVel ()
  Gets the desired velocity of the Vehicle's driver model.
- double getAccel ()
  Gets the acceleration of the Vehicle.
- double getLength ()
  Gets the length of the Vehicle.
- double getVelocity ()
  Gets the velocity of the Vehicle.
- int getLane ()
  Gets the lane that the Vehicle is in.
- bool getDirection ()
  Gets the direction that the Vehicle is travelling in.
- WORD getID ()
  Gets the class of the Vehicle.
- double getGVW ()
  Gets the GVW of the vehicle.
- int getNoAxles ()
  Gets the vehicle's number of axles.
- double getAW (int i)
  Gets the weight of a given axle.
- double getAS (int i)
  Gets the spacing of a given axle.
- DriverModel * getDriver ()
  Gets the Vehicle's DriverModel.
- void setDriver (IDM driver)
  Sets the Vehicle's DriverModel.
- void createSAFTVehicle (std::string data)

- void createCASTORVehicle (std::string data)
- void writeSAFTData (char *pTruck)

  Prepares a vehicle for printing to a SAFT file.

- void writeCASTORData (char *pTruck)

  Prepares a vehicle for printing to a CASTOR file.

- void updateProperties (double step, double accel)

  Updates the velocity, acceleration and position of the Vehicle based its acceleration and the timestep.

- void update (double step, Vehicle *pV)

  Updates the position of the Vehicle based on the timestep and preceding Vehicle.

- void update (double step)

  Updates the position of the Vehicle based on the timestep.

- double calcAccel ()

  Calculates a Vehicle's acceleration when unimpeded by other Vehicles.

- bool changeTime (double step)

  Handles the timer for checking Lane changes.

- double calcAccel (Vehicle *pV)

  Calculates a Vehicle's acceleration based on properties of the preceding Vehicle.

- int DecideNextLane (double LeftAdv, double RightAdv)
- double DecideLaneChange (Vehicle *frontVeh, Vehicle *backVeh, bool overtake)
- int DecideLaneChange (Vehicle *LeftFrontVehicle, Vehicle *LeftBackVehicle, Vehicle *RightFrontVehicle, Vehicle *RightBackVehicle, bool LeftLaneExists, bool RightLaneExists)

  Handles whether or not the Vehicle will change Lane.

Protected Member Functions

- int Round (double val)
- WORD setID ()
- double LaneChangeAdvantage (Vehicle *FrontVehicle, Vehicle *BackVehicle, bool overtake)

  Gets the advantage a Vehicle would gain from switching Lanes.

Protected Attributes

- DriverModel * m_pDriver
- IDM m_IDMDriver
- bool m_DriveOnRight
- bool m_DirPos
- int m_intDir
- int m_Lane
- int m_RoadLength
- double m_Position
- double m_RoadPosition
- double m_Velocity
- double m_Acceleration
- int m_Order
- int m_Head
- int m_Year
- int m_Month
- int m_Day
- int m_Hour
- int m_Min
- int m_Sec
- int m_Hndt
- double m_GVW
- int m_Trns
- int m_NoAxles
- double m_Length
- std::vector< Axle * > m_vAxles
- WORD m_ID
- bool m_bLaneChange
- double m_Tdelay
- int m_TotalNoLanesInRoad
- int ROAD_END_BUFFER
- double T_DELAY
- double MIN_SPACE_FOR_NEXT_VEHICLE
- double SAFE_BRAKING
- double CRANE_AVERAGE_SPACING
- double CRANE_MAX_SPACING
- double LOWLOADER_MIN_MAX_SPACING
- int SMALL_TRUCK_NO_AXLES
- int DAYS_PER_MT
- int MTS_PER_YR

4.53.1 Detailed Description

A base class from which specific Vehicle types are derived.

Definition at line 14 of file Vehicle.h.

4.53.2 Constructor & Destructor Documentation

4.53.2.1 Vehicle::Vehicle ()

Default Constructor.

Definition at line 12 of file Vehicle.cpp.

References CConfigData::VehicleID_Config::CRANE_AVERAGE_SPACING, CRANE_AVERAGE_SPACING, CConfigData::VehicleID_Config::CRANE_-MAX_SPACING, CRANE_MAX_SPACING, CConfigData::Time_Config::DAYS_-PER_MT, DAYS_PER_MT, CConfigData::IDM, CConfigData::VehicleID_-Config::LOWLOADER_MIN_MAX_SPACING, LOWLOADER_MIN_MAX_-SPACING, CConfigData::IDM_Config::MIN_SPACE_FOR_NEXT_VEHICLE, MIN_SPACE_FOR_NEXT_VEHICLE, CConfigData::Time_Config::MTS_PER_YR, MTS_PER_YR, CConfigData::Road, CConfigData::Road_Config::ROAD_END_-BUFFER, ROAD_END_BUFFER, CConfigData::IDM_Config::SAFE_BRAKING, SAFE_BRAKING, CConfigData::VehicleID_Config::SMALL_TRUCK_NO_-AXLES, SMALL_TRUCK_NO_AXLES, CConfigData::IDM_Config::T_DELAY, T_DELAY, CConfigData::Time, and CConfigData::VehicleID.

```
13 {
14         ROAD_END_BUFFER                         = g_ConfigData.Road.ROAD_END_BUFFER;
15
16         T_DELAY                                         = g_ConfigData.IDM.T_DELAY;
17         MIN_SPACE_FOR_NEXT_VEHICLE      = g_ConfigData.IDM.MIN_SPACE_FOR_NEXT_VEHICLE;
18         SAFE_BRAKING                            = g_ConfigData.IDM.SAFE_BRAKING;
19
20         CRANE_AVERAGE_SPACING           = g_ConfigData.VehicleID.CRANE_AVERAGE_SPACING;
21         CRANE_MAX_SPACING                       = g_ConfigData.VehicleID.CRANE_MAX_SPACING;
22         LOWLOADER_MIN_MAX_SPACING       = g_ConfigData.VehicleID.LOWLOADER_MIN_MAX_SPACING;
23         SMALL_TRUCK_NO_AXLES            = g_ConfigData.VehicleID.SMALL_TRUCK_NO_AXLES;
24
25         DAYS_PER_MT                                     = g_ConfigData.Time.DAYS_PER_MT;
26         MTS_PER_YR                                      = g_ConfigData.Time.MTS_PER_YR;
27 }
```

4.53.2.2 Vehicle::~Vehicle () [virtual]

Default Destructor.

Definition at line 30 of file Vehicle.cpp.

```
31 {
32
33 }
```

4.53.3 Member Function Documentation

4.53.3.1 void Vehicle::init (int *RoadLength*, bool *DriveOnRIght*)

Definition at line 561 of file Vehicle.cpp.

References m_DriveOnRight, and m_RoadLength.

```
562 {
563         m_RoadLength = RoadLength;
564         m_DriveOnRight = DriveOnRight;
565 }
```

4.53.3.2 int Vehicle::getLaneNoInDirection ()

Definition at line 645 of file Vehicle.cpp.

References m_DirPos, m_Lane, and m_TotalNoLanesInRoad.

Referenced by MetricsDetector::AddCurrentVehicle(), and writeCASTORData().

```
646 {
647         if(m_DirPos)            // if pos direction, local = global
648                 return m_Lane;
649         else                            // if neg, it's not
650         {
651                 ASSERT(m_Lane <= m_TotalNoLanesInRoad); // if it isn't we have a negative index
652                 int iLocalLane = m_TotalNoLanesInRoad - m_Lane + 1;     // must be 1-based lane
653                 return iLocalLane;
654         }
655 }
```

4.53.3.3 void Vehicle::setTotalNoLanesInRoad (int *nLanes*)

Definition at line 639 of file Vehicle.cpp.

References m_TotalNoLanesInRoad.

Referenced by Road::MapTrafLaneToSimLane().

```
640 {
641         m_TotalNoLanesInRoad = nLanes;
642 }
```

4.53.3.4 void Vehicle::setTime (double *time*)

Definition at line 534 of file Vehicle.cpp.

References DAYS_PER_MT, HOURS_PER_DAY, m_Day, m_Hndt, m_Hour, m_Min, m_Month, m_Sec, m_Year, MINS_PER_HOUR, MTS_PER_YR, and SECS_PER_-HOUR.

Referenced by OutputDetector::addVehicle().

```
535 {
536         double temp = time;
537         const int mpy = MTS_PER_YR;     const int dpm = DAYS_PER_MT;
538
539         m_Year  = (int)(temp/(MTS_PER_YR * DAYS_PER_MT * HOURS_PER_DAY * SECS_PER_HOUR));
540
541         temp    = time - m_Year * MTS_PER_YR * DAYS_PER_MT * HOURS_PER_DAY * SECS_PER_HOUR;
542         m_Month = (int)(temp/(DAYS_PER_MT * HOURS_PER_DAY * SECS_PER_HOUR));
```

```
543
544         temp    = temp - m_Month * DAYS_PER_MT * HOURS_PER_DAY * SECS_PER_HOUR;
545         m_Day   = (int)(temp/(HOURS_PER_DAY * SECS_PER_HOUR));
546
547         temp    = temp - m_Day * HOURS_PER_DAY * SECS_PER_HOUR;
548         m_Hour  = (int)(temp/(SECS_PER_HOUR));
549
550         temp    = temp - m_Hour * SECS_PER_HOUR;
551         m_Min   = (int)(temp/(MINS_PER_HOUR));
552
553         temp    = temp - m_Min * MINS_PER_HOUR;
554         m_Sec   = (int)(temp);
555
556         temp    = temp - m_Sec;
557         m_Hndt  = (int)(100 * temp);
558
559 }
```

4.53.3.5 void Vehicle::setChangeStatus (bool *stat*)

Sets whether or not the Vehicle wishes to change lanes.

Parameters:

stat whether or not the Vehicle wants to change lanes

Definition at line 360 of file Vehicle.cpp.

References m_bLaneChange.

Referenced by update().

```
361 {
362         m_bLaneChange = stat;
363 }
```

4.53.3.6 void Vehicle::setRoadPos (double *r*)

Sets the position of the Vehicle on the road.

For graphical use

Parameters:

r the Vehicle's position on the road

Definition at line 369 of file Vehicle.cpp.

References m_RoadPosition.

```
370 {
371         m_RoadPosition = r;
372 }
```

4.53.3.7 void Vehicle::setPos (double *p*)

Sets the position of the Vehicle to a given value.

Parameters:

p the Vehicle's position

Definition at line 351 of file Vehicle.cpp.

References m_Position.

```
352 {
353         m_Position = p;
354 }
```

4.53.3.8 void Vehicle::setAccel (double *acc*)

Sets the acceleration of the Vehicle to a given value.

Parameters:

acc the Vehicle's new acceleration

Definition at line 322 of file Vehicle.cpp.

References m_Acceleration.

```
323 {
324         m_Acceleration = acc;
325 }
```

4.53.3.9 void Vehicle::setLength (double *length*)

Sets the length of the Vehicle to a given value.

Parameters:

length The Vehicle's new length

Definition at line 313 of file Vehicle.cpp.

References m_Length.

```
314 {
315         m_Length = length;
316 }
```

4.53.3.10 void Vehicle::setVelocity (double *vel*)

Sets the velocity of the Vehicle to a given value.

Parameters:

vel the Vehicle's new velocity

Definition at line 304 of file Vehicle.cpp.

References m_Velocity.

```
305 {
306         m_Velocity = vel;
307 }
```

4.53.3.11 void Vehicle::setLane (int *lane*)

Sets the lane that the Vehicle is in.

Parameters:

lane the Vehicle's lane

Definition at line 331 of file Vehicle.cpp.

References m_Lane.

Referenced by Lane::ImplementLaneChange(), and Road::MapTrafLaneToSimLane().

```
332 {
333         m_Lane = lane;
334 }
```

4.53.3.12 void Vehicle::setDirection (bool *DirPos*)

Sets the direction of the Vehicle.

Parameters:

DirPos Whether or not the vehicle is travelling in a positive direction

Definition at line 341 of file Vehicle.cpp.

References m_DirPos.

```
342 {
343         m_DirPos = DirPos;
344 }
```

4.53.3.13 void Vehicle::setGVW (double *weight*)

Sets the GVW of the vehicle to a given weight.

Parameters:

weight the desired GVW

Reimplemented in Truck.

Definition at line 477 of file Vehicle.cpp.

References m_GVW.

```
478 {
479         m_GVW = weight;
480 }
```

4.53.3.14 void Vehicle::setAxles ()

Sets the vehicle's number of axles to a given number.

This function sets the vehicle's number of Axles and current differentiates between large and small trucks based on the number of Axles

Definition at line 607 of file Vehicle.cpp.

References m_NoAxles, and m_vAxles.

Referenced by createCASTORVehicle(), and createSAFTVehicle().

```
608 {
609         for(int i = 0; i < m_NoAxles; i++)
610         {
611                 Axle* axle = new Axle();
612                 m_vAxles.push_back(axle);
613         }
614 }
```

4.53.3.15 void Vehicle::setAW (int *i*, double *w*)

Sets the weight of a given axle.

Parameters:

i the axle to operate upon

w the desired weight

Reimplemented in Truck.

Definition at line 597 of file Vehicle.cpp.

References m_vAxles.

Referenced by createCASTORVehicle(), and createSAFTVehicle().

```
598 {
599         m_vAxles.at(i)->setWeight(w);
600 }
```

4.53.3.16 void Vehicle::setAxles (int *axNo*)

4.53.3.17 void Vehicle::setAS (int *i*, double *s*)

Sets the spacing of a given axle.

The space of a given axle to the next axle

Parameters:

i the axle to operate upon

s the desired spacing

Reimplemented in Truck.

Definition at line 588 of file Vehicle.cpp.

References m_vAxles.

Referenced by createCASTORVehicle(), and createSAFTVehicle().

```
589 {
590         m_vAxles.at(i)->setSpacing(s);
591 }
```

4.53.3.18 CString Vehicle::getDataString ()

Definition at line 881 of file Vehicle.cpp.

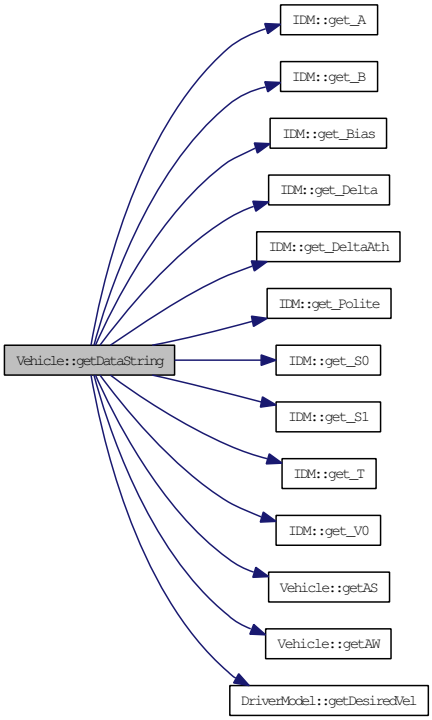
References IDM::get_A(), IDM::get_B(), IDM::get_Bias(), IDM::get_Delta(), IDM::get_DeltaAth(), IDM::get_Polite(), IDM::get_S0(), IDM::get_S1(), IDM::get_T(), IDM::get_V0(), getAS(), getAW(), DriverModel::getDesiredVel(), m_Acceleration, m_DirPos, m_GVW, m_IDMDriver, m_Lane, m_Length, m_NoAxles, m_pDriver, M_PER_S_TO_KM_PER_H, m_RoadPosition, and m_Velocity.

Referenced by CEvolveTrafficView::OnLButtonDown().

```
882 {
883         CString dataStr, tempStr, strSplitter;
884         strSplitter = "-----------------------------------\n";
885 
886         dataStr = "Vehicle Data:\n";
887         dataStr += strSplitter;
888         dataStr += "Direction: \t";
889         tempStr = m_DirPos ? "Positive\n" : "Negative\n";
890         dataStr += tempStr;
891 
892         tempStr.Format("Lane:        \t%d\n",                  m_Lane);
893         tempStr.Format("Position:    \t%4.1f m\n",             m_RoadPosition);
894         tempStr.Format("Velocity:    \t%3.1f km/h\n",   m_Velocity*M_PER_S_TO_KM_PER_H);
```

```
895         tempStr.Format("Acceleration:\t%2.2f m/s^2\n",  m_Acceleration);
896         tempStr.Format("Desired Vel.:\t%3.1f km/h\n",   m_pDriver->getDesiredVel()*M_PER_S_TO_K
897
898         dataStr += strSplitter;
899
900         tempStr.Format("IDM T: \t\t%1.2f s\n",          m_IDMDriver.get_T() );
901         tempStr.Format("IDM A: \t\t%2.2f m/s^2\n",      m_IDMDriver.get_A() );
902         tempStr.Format("IDM B: \t\t%2.2f m/s^2\n",      m_IDMDriver.get_B() );
903         tempStr.Format("IDM S0:\t\t%2.2f m\n",          m_IDMDriver.get_S0() );
904         tempStr.Format("IDM S1:\t\t%2.2f m\n",          m_IDMDriver.get_S1() );
905         tempStr.Format("IDM V0:\t\t%3.1f m/s\n",        m_IDMDriver.get_V0() );
906         tempStr.Format("IDM Delta:\t%2.1f\n",           m_IDMDriver.get_Delta() );
907         tempStr.Format("IDM Polite:\t%2.1f\n",          m_IDMDriver.get_Polite() );
908         tempStr.Format("IDM Bias:\t%2.1f\n",            m_IDMDriver.get_Bias() );
909         tempStr.Format("IDM DeltaAth:\t%2.1f\n",        m_IDMDriver.get_DeltaAth() );
910
911         dataStr += strSplitter;
912
913         tempStr.Format("GVW:         \t%3.1f kN\n",             m_GVW);
914         tempStr.Format("Length:      \t%2.1f m\n",              m_Length);
915         tempStr.Format("No of Axles: \t%d\n",                   m_NoAxles);
916
917         for(int i = 0; i < m_NoAxles; i++)
918         {
919                 tempStr.Format("Axle %d:\t", i+1);                      dataStr += tempStr;
920                 tempStr.Format("%2.1f kN",  getAW(i) ); dataStr += tempStr;
921                 if(i < m_NoAxles - 1)
922                 {
923                         tempStr.Format(" | %2.1f m\n",getAS(i) );        dataStr += tempStr;
924                 }
925         }
926
927         return dataStr;
928 }
```

Here is the call graph for this function:



4.53.3.19 double Vehicle::getTime () const [virtual]

Gets the time of arrival of the vehicle in seconds.

Returns:

The time of arrival of the vehicle in seconds

Definition at line 522 of file Vehicle.cpp.

References DAYS_PER_MT, HOURS_PER_DAY, m_Day, m_Hndt, m_Hour, m_Min, m_Month, m_Sec, m_Year, MTS_PER_YR, SECS_PER_HOUR, and SECS_PER_-MIN.

Referenced by Truck::operator<().

```
523 {
524         // working off a 10 month year & a 25 day month
525         int s_per_hr = SECS_PER_HOUR;
526         int s_per_day = SECS_PER_HOUR*HOURS_PER_DAY;
527 
528         int no_days = (m_Year - 5) * MTS_PER_YR * DAYS_PER_MT + (m_Month - 1) * DAYS_PER_MT +
529         double time = no_days * s_per_day + m_Hour * s_per_hr + m_Min * SECS_PER_MIN + m_Sec +
530 
531         return time;
532 }
```

4.53.3.20 void Vehicle::setRoadLength (int *length*)

Definition at line 567 of file Vehicle.cpp.

References m_RoadLength.

```
568 {
569         m_RoadLength = length;
570 }
```

4.53.3.21 bool Vehicle::getChangeStatus ()

Gets whether or not the Vehicle wishes to change lane.

Returns:

laneChange whether or not the Vehicle wishes to change lane

Definition at line 415 of file Vehicle.cpp.

References m_bLaneChange.

```
416 {
417         return m_bLaneChange;
418 }
```

4.53.3.22 double Vehicle::getRoadPos ()

Gets the position of the Vehicle on the road.

Graphical use

Returns:

roadPos the position of the Vehicle on the road

Definition at line 397 of file Vehicle.cpp.

References m_RoadPosition.

Referenced by CEvolveTrafficView::DrawVehicle(), and CEvolveTrafficView::FindVehicle().

```
398 {
399         return m_RoadPosition;
400 }
```

4.53.3.23 double Vehicle::getPos ()

Gets the Vehicle's position.

Returns:

m_x the Vehicle's position

Definition at line 379 of file Vehicle.cpp.

References m_Position.

Referenced by calcAccel(), Lane::ChangeLanes(), Detector::FindDetectorArrivalTime(), Lane::ImplementLaneChange(), Lane::insert(), and LaneChangeAdvantage().

```
380 {
381         return m_Position;
382 }
```

4.53.3.24 double Vehicle::getDesiredVel ()

Gets the desired velocity of the Vehicle's driver model.

Returns:

the driver model's desired velocity

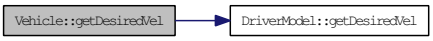
Definition at line 388 of file Vehicle.cpp.

References DriverModel::getDesiredVel(), and m_pDriver.

Referenced by SpeedLimit::addVehicle().

```
389 {
390         return m_pDriver->getDesiredVel();
391 }
```

Here is the call graph for this function:



4.53.3.25 double Vehicle::getAccel ()

Gets the acceleration of the Vehicle.

Returns:

acc the acceleration of the Vehicle

Definition at line 442 of file Vehicle.cpp.

References m_Acceleration.

Referenced by LaneChangeAdvantage().

```
443 {
444         return m_Acceleration;
445 }
```

4.53.3.26 double Vehicle::getLength ()

Gets the length of the Vehicle.

Returns:

len the length of the Vehicle

Definition at line 433 of file Vehicle.cpp.

References m_Length.

Referenced by calcAccel(), CEvolveTrafficView::DrawVehicle(), CEvolveTrafficView::FindVehicle(), and LaneChangeAdvantage().

```
434 {
435         return m_Length;
436 }
```

4.53.3.27 double Vehicle::getVelocity ()

Gets the velocity of the Vehicle.

Returns:

vel the velocity of the Vehicle

Definition at line 424 of file Vehicle.cpp.

References m_Velocity.

Referenced by MetricsDetector::AddCurrentVehicle(), calcAccel(), CEvolveTrafficView::DrawVehicle(), and Detector::FindDetectorArrivalTime().

```
425 {
426         return m_Velocity;
427 }
```

4.53.3.28 int Vehicle::getLane ()

Gets the lane that the Vehicle is in.

Returns:

lane the Vehicle's lane

Definition at line 451 of file Vehicle.cpp.

References m_Lane.

Referenced by CEvolveTrafficView::DrawVehicle(), Lane::ImplementLaneChange(), and Road::MapTrafLaneToSimLane().

```
452 {
453         return m_Lane;
454 }
```

4.53.3.29 bool Vehicle::getDirection ()

Gets the direction that the Vehicle is travelling in.

Returns:

dir the Vehicle's direction

Definition at line 460 of file Vehicle.cpp.

References m_DirPos.

Referenced by Lane::ChangeLanes(), CEvolveTrafficView::DrawVehicle(), CEvolveTrafficView::FindVehicle(), Lane::ImplementLaneChange(), and Road::MapTrafLaneToSimLane().

```
461 {
462         return m_DirPos;
463 }
```

4.53.3.30 WORD Vehicle::getID ()

Gets the class of the Vehicle.

Returns:

ID the Vehicle's class

Reimplemented in Truck.

Definition at line 406 of file Vehicle.cpp.

References m_ID.

Referenced by MetricsDetector::AddCurrentVehicle(), SpeedLimit::addVehicle(), MetricsDetector::addVehicle(), CEvolveTrafficView::DrawVehicle(), Lane::ImplementLaneChange(), and Road::setIDMDriverModel().

```
407 {
408         return m_ID;
409 }
```

4.53.3.31 double Vehicle::getGVW ()

Gets the GVW of the vehicle.

Returns:

the GVW of the vehicle

Definition at line 469 of file Vehicle.cpp.

References m_GVW.

```
470 {
471         return m_GVW;
472 }
```

4.53.3.32 int Vehicle::getNoAxles ()

Gets the vehicle's number of axles.

Returns:

the vehicle's number of axles

Reimplemented in Truck.

Definition at line 576 of file Vehicle.cpp.

References m_NoAxles.

```
577 {
578         return m_NoAxles;
579 }
```

4.53.3.33 double Vehicle::getAW (int *i*)

Gets the weight of a given axle.

Parameters:

i the axle to operate upon

Returns:

the weight of the axle

Reimplemented in Truck.

Definition at line 633 of file Vehicle.cpp.

References m_vAxles.

Referenced by getDataString(), writeCASTORData(), and writeSAFTData().

```
634 {
635         return m_vAxles.at(i)->getWeight();
636 }
```

4.53.3.34 double Vehicle::getAS (int *i*)

Gets the spacing of a given axle.

The space of a given axle to the next axle

Parameters:

i the axle to operate upon

Returns:

the space between this axle and the next

Reimplemented in Truck.

Definition at line 623 of file Vehicle.cpp.

References m_vAxles.

Referenced by getDataString(), setID(), writeCASTORData(), and writeSAFTData().

```
624 {
625         return m_vAxles.at(i)->getSpacing();
626 }
```

4.53.3.35 DriverModel * Vehicle::getDriver ()

Gets the Vehicle's DriverModel.

Returns:

the Vehicle's driver

Definition at line 169 of file Vehicle.cpp.

References m_pDriver.

Referenced by SpeedLimit::addVehicle(), Gradient::addVehicle(), Lane::CheckRoadSegments(), SpeedLimit::removeVehicle(), and Gradient::removeVehicle().

```
170 {
171          return m_pDriver;
172 }
```

4.53.3.36 void Vehicle::setDriver (IDM *driver*)

Sets the Vehicle's DriverModel.

Parameters:

driver The Vehicle's driver

Definition at line 177 of file Vehicle.cpp.

References m_IDMDriver, and m_pDriver.

Referenced by Road::setIDMDriverModel().

```
178 {
179         m_IDMDriver = driver;
180         m_pDriver = &m_IDMDriver;
181 }
```

4.53.3.37 void Vehicle::createSAFTVehicle (std::string *data*)

Reimplemented in Truck.

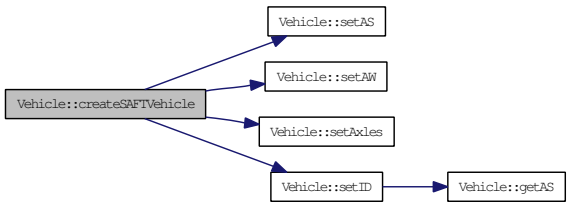
Definition at line 717 of file Vehicle.cpp.

References m_Day, m_GVW, m_Head, m_Hndt, m_Hour, m_ID, m_intDir, m_Lane, m_Length, m_Min, m_Month, m_NoAxles, m_Order, m_Position, m_RoadPosition, m_Sec, m_Tdelay, m_Velocity, m_Year, setAS(), setAW(), setAxles(), and setID().

Referenced by Truck::createSAFTVehicle(), and SAFTFile::readLine().

```
718 {
719         m_Position = 0.0;
720         m_RoadPosition = m_Position;
721 
722         m_Order         = atoi( data.substr(0,5).c_str() );
723         m_Head          = 20005;
724         m_Day           = atoi( data.substr(10,2).c_str() );
725         m_Month         = atoi( data.substr(12,2).c_str() );
726         m_Year          = atoi( data.substr(14,2).c_str() );
727         m_Hour          = atoi( data.substr(16,2).c_str() );
728         m_Min           = atoi( data.substr(18,2).c_str() );
729         m_Sec           = atoi( data.substr(20,2).c_str() );
730         m_Hndt          = atoi( data.substr(22,2).c_str() );
731         m_Velocity      = atoi( data.substr(24,3).c_str() );
732         m_GVW           = atoi( data.substr(27,4).c_str() );
733         m_Length        = atoi( data.substr(31,3).c_str() );
734         m_NoAxles       = atoi( data.substr(34,1).c_str() );
735 
736         m_Lane = 1;
737         m_intDir = 1;   // m_DirPointsToRight = true;
738 
739         // Length = length/10 for dm to meters
740         // Vel = vel / 10 for dm/s to meters/second
741         m_Length        /= 10;
742         m_Velocity      /= 10;
743 
744         m_Tdelay        = 0.0;
745 
746         setAxles();
747         double W;
748         double S;
749         int j = 33;
750 
751         for(int i = 0; i < m_NoAxles; i++)
752         {
753                 j += 2;
754                 W = atoi( data.substr(j,3).c_str() );
755                 setAW(i, W);
756 
757                 if(i == m_NoAxles-1) break;
758 
759                 j += 3;
760                 S = atoi( data.substr(j,2).c_str() );
761                 S /= 10;                // convert to m
762                 setAS(i, S);
763         }
764 
765         m_ID = setID();         // why is this here
766 }
```

Here is the call graph for this function:



4.53.3.38 void Vehicle::createCASTORVehicle (std::string *data*)

Reimplemented in Truck.

Definition at line 661 of file Vehicle.cpp.

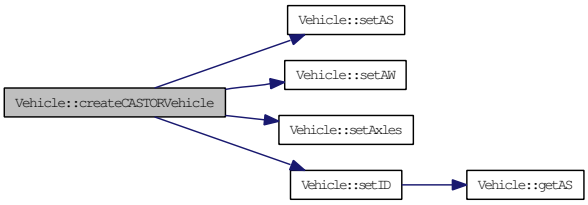
References KG100_TO_KN, m_Day, m_DirPos, m_GVW, m_Head, m_Hndt, m_Hour, m_ID, m_intDir, m_Lane, m_Length, m_Min, m_Month, m_NoAxles, m_Position, m_RoadPosition, m_Sec, m_Tdelay, m_Trns, m_Velocity, m_Year, setAS(), setAW(), setAxles(), and setID().

Referenced by Truck::createCASTORVehicle(), and CASTORFile::readLine().

```
662 {
663         m_Position = 0.0;
664         m_RoadPosition = m_Position;
665
666         m_Head          = atoi( data.substr(0,4).c_str() );
667         m_Day           = atoi( data.substr(4,2).c_str() );
668         m_Month         = atoi( data.substr(6,2).c_str() );
669         m_Year          = atoi( data.substr(8,2).c_str() );
670         m_Hour          = atoi( data.substr(10,2).c_str() );
671         m_Min           = atoi( data.substr(12,2).c_str() );
672         m_Sec           = atoi( data.substr(14,2).c_str() );
673         m_Hndt          = atoi( data.substr(16,2).c_str() );
674         m_Velocity      = atoi( data.substr(18,3).c_str() );
675         m_GVW           = atoi( data.substr(21,4).c_str() );
676         m_Length        = atoi( data.substr(25,3).c_str() );
677         m_NoAxles       = atoi( data.substr(28,1).c_str() );
678         m_intDir        = atoi( data.substr(29,1).c_str() );
679         m_Lane          = atoi( data.substr(30,1).c_str() );
680         m_Trns          = atoi( data.substr(31,3).c_str() );
681
682         m_DirPos = m_intDir == 1 ? true : false;                // Required for compatibility w
683
684         // Length = length/10 for dm to meters
685         // Vel = vel / 10 for dm/s to meters/second
686         // GVW * 0.981 for kg/100 to kN
687         // Trns = trns/10 for dm to meters
688
689         m_Length        /= 10;
690         m_Velocity      /= 10;
691         m_GVW           *= KG100_TO_KN;
692         m_Trns          /= 10;
693
694         m_Tdelay        = 0.0;
695
```

```
696         setAxles();
697         double W;
698         double S;
699         int j = 32;
700
701         for(int i = 0; i < m_NoAxles; i++)
702         {
703                 j += 2;
704                 W = atoi( data.substr(j,3).c_str() );
705                 W *= KG100_TO_KN;       // convert to kN
706                 setAW(i, W);
707
708                 j += 3;
709                 S = atoi( data.substr(j,2).c_str() );
710                 S /= 10;                // convert to m
711                 setAS(i, S);
712         }
713
714         m_ID = setID();
715 }
```

Here is the call graph for this function:



4.53.3.39 void Vehicle::writeSAFTData (char * *pVehChar*)

Prepares a vehicle for printing to a SAFT file.

Parameters:

pVehChar The string of data to represent the truck

This function takes in a string, and writes all of its properties to the string in SAFT format, in order to prepare for printing of the vehicle

Definition at line 839 of file Vehicle.cpp.

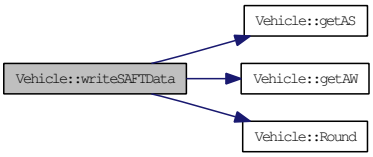
References getAS(), getAW(), KG100_TO_KN, m_Day, m_GVW, m_Head, m_Hndt, m_Hour, m_Length, m_Min, m_Month, m_NoAxles, m_Order, m_Sec, m_Velocity, m_Year, and Round().

Referenced by SAFTFile::writeLine().

```
840 {
841         // Length = length*10 for meters to decimeters
842         // Vel = vel * 10 for metres/second to decimetres/second
843
```

```
844         int velocity    = Round(m_Velocity*10);
845         int length              = Round(m_Length*10);
846
847         ostrstream oFile( pVehChar, sizeof(pVehChar)*78 );
848
849         oFile.fill('0');
850         oFile.width(5); oFile << m_Order;
851         oFile.width(5); oFile << m_Head;
852         oFile.width(2); oFile << m_Day;
853         oFile.width(2); oFile << m_Month;
854         oFile.width(2); oFile << m_Year;
855         oFile.width(2); oFile << m_Hour;
856         oFile.width(2); oFile << m_Min;
857         oFile.width(2); oFile << m_Sec;
858         oFile.width(2); oFile << m_Hndt;
859         oFile.fill(' ');
860         oFile.width(3); oFile << velocity;
861         oFile.width(4); oFile << m_GVW;
862         oFile.width(3); oFile << length;
863         oFile.width(1); oFile << m_NoAxles;
864
865         int j = 3;
866         for(int i = 0; i < m_NoAxles; i++)
867         {
868                 oFile.width(j); oFile << Round(this->getAW(i)/KG100_TO_KN);     // convert fron
869                 j = 2;
870
871                 if(i == m_NoAxles-1)
872                         break;
873
874                 oFile.width(j); oFile << Round(this->getAS(i)*10);              // convert fron
875                 j = 3;
876         }
877
878         oFile << ends;
879 }
```

Here is the call graph for this function:



4.53.3.40 void Vehicle::writeCASTORData (char * *pVehChar*)

Prepares a vehicle for printing to a CASTOR file.

Parameters:

pVehChar The string of data to represent the truck

This function takes in a string, and writes all of its properties to the string in CASTOR format, in order to prepare for printing of the vehicle

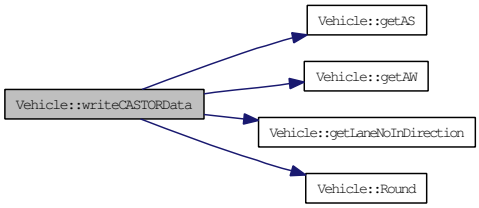
Definition at line 776 of file Vehicle.cpp.

References CASTOR_MAX_AXLES, getAS(), getAW(), getLaneNoInDirection(), KG100_TO_KN, m_Day, m_GVW, m_Head, m_Hndt, m_Hour, m_intDir, m_Length, m_Min, m_Month, m_NoAxles, m_Sec, m_Trns, m_Velocity, m_Year, and Round().

Referenced by CASTORFile::writeLine().

```
777 {
778         // Length = length*10 for meters to decimeters
779         // Vel = vel * 10 for metres/second to decimetres/second
780         // GVW / 0.981 for KN to KG/100
781         // Trns = trns/10 for metres to decimetres
782 
783         int velocity    = Round(m_Velocity*10);
784         int grossWeight = Round(m_GVW/KG100_TO_KN);
785         int length              = Round(m_Length*10);
786         int transPos    = Round(m_Trns*10);
787 
788         ostrstream oFile( pVehChar, sizeof(pVehChar)*78 );
789 
790         oFile.width(4); oFile << m_Head;
791         oFile.width(2); oFile << m_Day;
792         oFile.width(2); oFile << m_Month;
793         oFile.width(2); oFile << m_Year;
794         oFile.width(2); oFile << m_Hour;
795         oFile.width(2); oFile << m_Min;
796         oFile.width(2); oFile << m_Sec;
797         oFile.width(2); oFile << m_Hndt;
798         oFile.width(3); oFile << velocity;
799         oFile.width(4); oFile << grossWeight;
800         oFile.width(3); oFile << length;
801         oFile.width(1); oFile << m_NoAxles;
802         oFile.width(1); oFile << m_intDir;
803         oFile.width(1); oFile << getLaneNoInDirection();        // local lane no in direction
804         oFile.width(3); oFile << transPos;
805 
806         int j = 3;
807         for(int i = 0; i < m_NoAxles; i++)
808         {
809                 oFile.width(j); oFile << Round(this->getAW(i)/KG100_TO_KN);     // convert fron
810                 j = 2;
811 
812                 if(i == 8) break;
813 
814                 oFile.width(j); oFile << Round(this->getAS(i)*10);              // convert fron
815                 j = 3;
816         }
817 
818         for(i = m_NoAxles; i < CASTOR_MAX_AXLES; i++)
819         {
820                 oFile.width(j); oFile << "0";
821                 j = 2;
822 
823                 if(i == 8) break;
824 
825                 oFile.width(j); oFile << "0";
826                 j = 3;
827         }
828 
829         oFile << ends;
830 }
```

Here is the call graph for this function:



4.53.3.41 void Vehicle::updateProperties (double *step*, double *accel*)

Updates the velocity, acceleration and position of the Vehicle based its acceleration and the timestep.

Parameters:

step The timestep

accel The Vehicle's acceleration

This function updates the properties of the Vehicle based on how much it has accelerated in the previous timestep, assuming average velocity over the timestep. The velocity, and acceleration of the Vehicle are updated, as well as the Vehicle's position.

Definition at line 127 of file Vehicle.cpp.

References m_Acceleration, m_Position, m_RoadPosition, and m_Velocity.

Referenced by update().

```
128 {
129         // Assume average velocity over timestep
130         double v_old = m_Velocity;
131         double v_new = m_Velocity + accel*step;
132 
133         // zero speeds may occur when very close to stationary obstacle
134         // eliminate as per Treiber's comments to Carpool Tunnel
135         if(v_new < 0)
136         {
137                 //TRACE("*** Backwards move prevented\n");
138                 v_new = 0.0;
139         }
140 
141         m_Position += 0.5*(v_old + v_new)*step;
142 
143         m_RoadPosition = m_Position;
144         m_Velocity = v_new;
145 
146         m_Acceleration = accel;
147 }
```

4.53.3.42 void Vehicle::update (double *step*, Vehicle * *pV*)

Updates the position of the Vehicle based on the timestep and preceding Vehicle.

Parameters:

step the timestep

pV a pointer to the Vehicle in front of this one

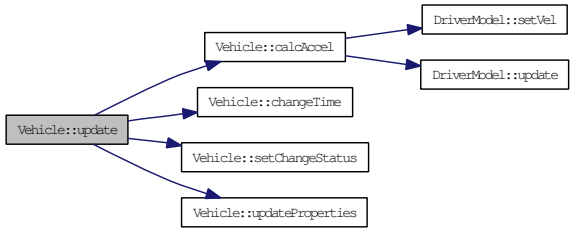
Every Vehicle in the Lane will call this function except the head of the Lane. The velocity and acceleration of the Vehicle will be altered based on the velocity of, and distance to, the Vehicle in front. This function also handles the timer for when the Vehicle will check to see if changing Lane would be advantageous.

Definition at line 70 of file Vehicle.cpp.

References calcAccel(), changeTime(), setChangeStatus(), and updateProperties().

```
71 {
72         bool change = changeTime(step);
73
74         if(change)
75                 setChangeStatus(true);
76
77         double accel = calcAccel(pV);
78
79         updateProperties(step, accel);
80 }
```

Here is the call graph for this function:



4.53.3.43 void Vehicle::update (double *step*)

Updates the position of the Vehicle based on the timestep.

Parameters:

step the timestep

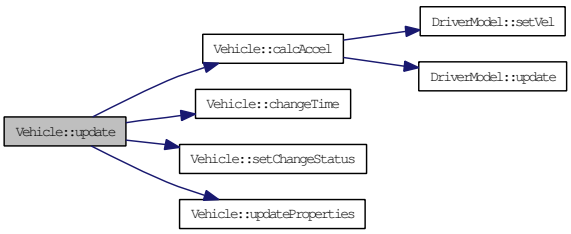
The Vehicle calling this function will always be the head of traffic in its particular Lane, as it has no Vehicle in front to reference and instead drives towards the end of the Road This function also handles the timer for when the Vehicle will check to see if changing Lane would be advantageous.

Definition at line 49 of file Vehicle.cpp.

References calcAccel(), changeTime(), setChangeStatus(), and updateProperties().

```
50 {
51         if( changeTime(step) )
52                 setChangeStatus(true);
53
54         double accel = calcAccel();
55
56         updateProperties(step, accel);
57 }
```

Here is the call graph for this function:



4.53.3.44 double Vehicle::calcAccel ()

Calculates a Vehicle's acceleration when unimpeded by other Vehicles.

This function serves the purpose of determining the acceleration of a Vehicle when there are no other Vehicles in front of it in the same Lane. Based on the distance to the end of the Road, the DriverModel alters acceleration accordingly

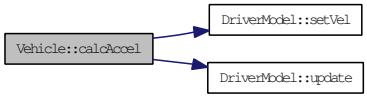
Definition at line 88 of file Vehicle.cpp.

References m_pDriver, m_Position, m_RoadLength, m_Velocity, ROAD_END_-BUFFER, DriverModel::setVel(), and DriverModel::update().

Referenced by LaneChangeAdvantage(), and update().

```
89 {
90         m_pDriver->setVel(m_Velocity);
91
92         //MAGIC NUMBER: ROAD END
93         double accel = m_pDriver->update(m_Velocity, (m_RoadLength + ROAD_END_BUFFER) - m_Positi
94
95         return accel;
96 }
```

Here is the call graph for this function:



4.53.3.45 bool Vehicle::changeTime (double *step*)

Handles the timer for checking Lane changes.

Parameters:

step The timestep

Returns:

Whether or not it is time to check for a Lane change

Definition at line 154 of file Vehicle.cpp.

References m_Tdelay, and T_DELAY.

Referenced by update().

```
155 {
156         m_Tdelay += step;
157
158         if (m_Tdelay >= T_DELAY)
159         {
160                 m_Tdelay = m_Tdelay - T_DELAY;
161                 return true;
162         }
163         else {return false;}
164 }
```

4.53.3.46 double Vehicle::calcAccel (Vehicle * *pV*)

Calculates a Vehicle's acceleration based on properties of the preceding Vehicle.

Parameters:

pV A pointer to the preceding Vehicle

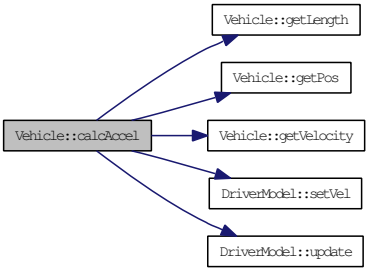
This function serves the purpose of determining the acceleration of a Vehicle when there is another Vehicle in front of it in the same Lane. Based on the distance between the two vehicles, and the difference in their velocity, the DriverModel alters acceleration accordingly.

Definition at line 106 of file Vehicle.cpp.

References getLength(), getPos(), getVelocity(), m_pDriver, m_Position, m_Velocity, DriverModel::setVel(), and DriverModel::update().

```
107 {
108         m_pDriver->setVel(m_Velocity);
109
110         double s = pV->getPos() - pV->getLength() - m_Position;
111
112         double delta_v = m_Velocity - pV->getVelocity();
113         double accel = m_pDriver->update(delta_v, s);
114
115         return accel;
116 }
```

Here is the call graph for this function:



4.53.3.47 int Vehicle::DecideNextLane (double *LeftAdv*, double *RightAdv*)

4.53.3.48 double Vehicle::DecideLaneChange (Vehicle * *frontVeh*, Vehicle * *backVeh*, bool *overtake*)

Referenced by Lane::ChangeLanes().

4.53.3.49 int Vehicle::DecideLaneChange (Vehicle * *LeftFrontVehicle*, Vehicle * *LeftBackVehicle*, Vehicle * *RightFrontVehicle*, Vehicle * *RightBackVehicle*, bool *LeftLaneExists*, bool *RightLaneExists*)

Handles whether or not the Vehicle will change Lane.

Parameters:

LeftFrontVehicle The Vehicle in front in the Left Lane

LeftBackVehicle The Vehicle behind in the Left Lane

RightFrontVehicle The Vehicle in front in the Right Lane

RightBackVehicle The Vehicle behind in the Right Lane

LeftLaneExists Whether or not there is a Lane to the left

RightLaneExists Whether or not there is a Lane to the Right

Returns:

The Lane to change to

See also:

LaneChangeAdvantage()

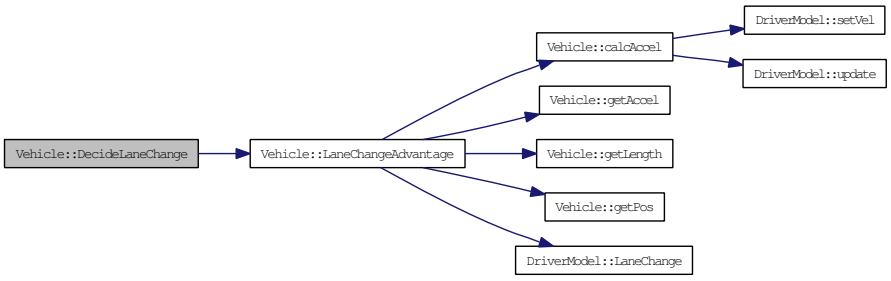
This function determines which Lane the Vehicle should change to if it changes Lane at all. Pointers to Vehicles that would be in front of, and behind, the current Vehicle if it were in either the right or left Lane are passed to the function, along with identifiers which specify whether or not there exists a Lane to the left or right of the current Lane.

Definition at line 204 of file Vehicle.cpp.

References LaneChangeAdvantage(), and m_DriveOnRight.

```
207 {
208         // Note that a possible conflict between what is the lane's leftLane and rightLane
209         // and the Vehicle's leftLane and Rightlane. For DirPos they are the same, but not
210         // for DirNeg. e.g.
211         // DirPos - Lane 1: leftLane = 2: rightLane = 0;
212         // DirNeg - Lane 5: leftLane = 6: rightLane = 4
213         // Whereas for the Vehicle:
214         // DirPos - in lane 1: lane to left = 2: lane to right = 0;
215         // DirNeg - in lane 5: lane to left = 4: lane to right = 6;
216         // Therefore:
217         // THIS FUNCTION NAMES LANES AS VIEWED BY THE VEHICLE
218
219         double LeftLaneAdvantage = 0.0;         double RightLaneAdvantage = 0.0;
220
221         if(m_DriveOnRight)
222         {       // we are overtaking properly
223                 if(LeftLaneExists)
224                         LeftLaneAdvantage = LaneChangeAdvantage(LeftFrontVehicle, LeftBackVehic
225                 // or we are being bold
226                 if(RightLaneExists)
227                         RightLaneAdvantage = LaneChangeAdvantage(RightFrontVehicle, RightBackVe
228         }
229         else
230         {       // we're boldly undertaking
231                 if(LeftLaneExists)
232                         LeftLaneAdvantage = LaneChangeAdvantage(LeftFrontVehicle, LeftBackVehic
233                 // we are overtaking properly
234                 if(RightLaneExists)
235                         RightLaneAdvantage = LaneChangeAdvantage(RightFrontVehicle, RightBackVe
236         }
237
238         const double zero = 0.0001;
239         if(LeftLaneAdvantage < zero && RightLaneAdvantage < zero)
240                 return 0;
241         else if(LeftLaneAdvantage > RightLaneAdvantage)
242                 return 1;
243         else
244                 return 2;
245 }
```

Here is the call graph for this function:



4.53.3.50 int Vehicle::Round (double *val*) [inline, protected]

Definition at line 86 of file Vehicle.h.

Referenced by writeCASTORData(), and writeSAFTData().

```
86 {return int(val + 0.5);};
```

4.53.3.51 WORD Vehicle::setID () [protected]

Definition at line 482 of file Vehicle.cpp.

References CRANE_AVERAGE_SPACING, getAS(), LOWLOADER_MIN_MAX_-SPACING, m_ID, m_NoAxles, m_vAxles, SMALL_TRUCK_NO_AXLES, VEH_-ID_CRANE, VEH_ID_LARGETRUCK, VEH_ID_LOWLOADER, and VEH_ID_-SMALLTRUCK.

Referenced by createCASTORVehicle(), and createSAFTVehicle().

```
483 {
484         // Small Truck or large truck
485         m_ID = (m_NoAxles <= SMALL_TRUCK_NO_AXLES) ?
486                                                 VEH_ID_SMALLTRUCK : VEH_ID_LARGETRUCK;
487
488
489         double averageSpace = getAS(0);
490         double maxSpace = 0;
491
492         // For all the vehicle's axles
493         for(int i = 1; i < m_NoAxles; i++)
494         {
495                 // Obtain the average spacing
496                 averageSpace += getAS(i);
497
498                 // Obtain the maximum spacing
499                 if(getAS(i) > getAS(i-1))
500                         maxSpace = getAS(i);
501         }
502
503         // If the maximum spacing is less than the maximum spacing for a crane and there are mo
504         if(maxSpace < CRANE_MAX_SPACING && m_NoAxles > 2)
505         {
506                 if(averageSpace / m_vAxles.size() < CRANE_AVERAGE_SPACING) // And the average s
507                         m_ID = VEH_ID_CRANE;    // It is a crane
508         }
509
510         // If the maximum spacing is at least the maximum for a low-loader and there are more t
511         if(maxSpace >= LOWLOADER_MIN_MAX_SPACING && m_NoAxles > 2)
512                 m_ID = VEH_ID_LOWLOADER;        //It is a low-loader
513
514         return m_ID;
515 }
```

Here is the call graph for this function:

4.53.3.52 double Vehicle::LaneChangeAdvantage (Vehicle * *FrontVehicle*, Vehicle * *BackVehicle*, bool *overtake*) [protected]

Gets the advantage a Vehicle would gain from switching Lanes.

Parameters:

FrontVehicle The Vehicle in front in the new Lane

BackVehicle The Vehicle behind in the new Lane

overtake Whether the Vehicle is overtaking or undertaking

Returns:

The net advantage of the Lane change

This function takes pointers to the Vehicles both in front, and behind, the position that the current Vehicle would have in the new Lane. If the Vehicle that would be in front is null, then the change in acceleration is decided with relation to acceleration on an open Road. Otherwise, the acceleration is decided with relation to the Vehicle in front.

Definition at line 260 of file Vehicle.cpp.

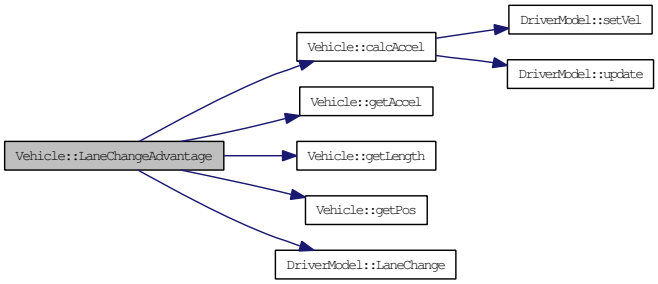
References calcAccel(), getAccel(), getLength(), getPos(), DriverModel::LaneChange(), m_pDriver, MIN_SPACE_FOR_NEXT_VEHICLE, and SAFE_BRAKING.

Referenced by DecideLaneChange().

```
261 {
262         // These are IDM properties and may be reomved from Constants.h
263         double GapToFront = MIN_SPACE_FOR_NEXT_VEHICLE;                 double GapToBack = MIN_
264         double CurrentBackAccel = SAFE_BRAKING;                                 double Proposed
265         double FrontChangeAccel = 0.0;
266
267         // calculate the above lane change parameters
268         if(FrontVehicle != NULL)
269         {
270                 GapToFront = FrontVehicle->getPos() - FrontVehicle->getLength() - this->getPos
271                 FrontChangeAccel = this->calcAccel(FrontVehicle) - this->getAccel();
272         }
273         else
274                 FrontChangeAccel = this->calcAccel() - this->getAccel();        // no front Veh
275
276         if(BackVehicle != NULL)
277         {
278                 GapToBack = this->getPos() - this->getLength() - BackVehicle->getPos();
279                 ProposedBackAccel = BackVehicle->calcAccel(this);
280                 // Since the acceleration of the back Vehicle at the next step
281                 // is affected by the presence of the Vehicle in front of it
282                 // we must take that into account and not an open-road acceleration
283                 // but only if there is a Vehicle in front!
284                 if(FrontVehicle != NULL)
285                         CurrentBackAccel = BackVehicle->calcAccel(FrontVehicle);
286                 else
287                         CurrentBackAccel = BackVehicle->getAccel();
288         }
289
290         double advantage =  m_pDriver->LaneChange(GapToFront, GapToBack, FrontChangeAccel,
```

```
291
292         return advantage;
293 }
```

Here is the call graph for this function:



4.53.4 Member Data Documentation

4.53.4.1 DriverModel* Vehicle::m_pDriver [protected]

Definition at line 91 of file Vehicle.h.

Referenced by calcAccel(), getDataString(), getDesiredVel(), getDriver(), LaneChangeAdvantage(), and setDriver().

4.53.4.2 IDM Vehicle::m_IDMDriver [protected]

Definition at line 92 of file Vehicle.h.

Referenced by getDataString(), and setDriver().

4.53.4.3 bool Vehicle::m_DriveOnRight [protected]

Definition at line 94 of file Vehicle.h.

Referenced by DecideLaneChange(), and init().

4.53.4.4 bool Vehicle::m_DirPos [protected]

Definition at line 95 of file Vehicle.h.

Referenced by createCASTORVehicle(), getDataString(), getDirection(), getLaneNoInDirection(), and setDirection().

4.53.4.5 int Vehicle::m_intDir [protected]

Definition at line 96 of file Vehicle.h.

Referenced by createCASTORVehicle(), createSAFTVehicle(), Truck::returnTruckData(), and writeCASTORData().

4.53.4.6 int Vehicle::m_Lane [protected]

Definition at line 97 of file Vehicle.h.

Referenced by createCASTORVehicle(), createSAFTVehicle(), getDataString(), getLane(), getLaneNoInDirection(), Truck::returnTruckData(), and setLane().

4.53.4.7 int Vehicle::m_RoadLength [protected]

Definition at line 98 of file Vehicle.h.

Referenced by calcAccel(), init(), and setRoadLength().

4.53.4.8 double Vehicle::m_Position [protected]

Definition at line 99 of file Vehicle.h.

Referenced by calcAccel(), createCASTORVehicle(), createSAFTVehicle(), getPos(), setPos(), and updateProperties().

4.53.4.9 double Vehicle::m_RoadPosition [protected]

Definition at line 100 of file Vehicle.h.

Referenced by createCASTORVehicle(), createSAFTVehicle(), getDataString(), getRoadPos(), setRoadPos(), and updateProperties().

4.53.4.10 double Vehicle::m_Velocity [protected]

Definition at line 102 of file Vehicle.h.

Referenced by calcAccel(), createCASTORVehicle(), createSAFTVehicle(), getDataString(), getVelocity(), Truck::returnTruckData(), setVelocity(), updateProperties(), writeCASTORData(), and writeSAFTData().

4.53.4.11 double Vehicle::m_Acceleration [protected]

Definition at line 103 of file Vehicle.h.

Referenced by getAccel(), getDataString(), setAccel(), and updateProperties().

4.53.4.12 int Vehicle::m_Order [protected]

Definition at line 105 of file Vehicle.h.

Referenced by createSAFTVehicle(), and writeSAFTData().

4.53.4.13 int Vehicle::m_Head [protected]

Definition at line 106 of file Vehicle.h.

Referenced by createCASTORVehicle(), createSAFTVehicle(), Truck::returnTruckData(), writeCASTORData(), and writeSAFTData().

4.53.4.14 int Vehicle::m_Year [protected]

Definition at line 108 of file Vehicle.h.

Referenced by createCASTORVehicle(), createSAFTVehicle(), getTime(), Truck::returnTruckData(), setTime(), writeCASTORData(), and writeSAFTData().

4.53.4.15 int Vehicle::m_Month [protected]

Definition at line 109 of file Vehicle.h.

Referenced by createCASTORVehicle(), createSAFTVehicle(), getTime(), Truck::returnTruckData(), setTime(), writeCASTORData(), and writeSAFTData().

4.53.4.16 int Vehicle::m_Day [protected]

Definition at line 110 of file Vehicle.h.

Referenced by createCASTORVehicle(), createSAFTVehicle(), getTime(), Truck::returnTruckData(), setTime(), writeCASTORData(), and writeSAFTData().

4.53.4.17 int Vehicle::m_Hour [protected]

Definition at line 111 of file Vehicle.h.

Referenced by createCASTORVehicle(), createSAFTVehicle(), getTime(), Truck::returnTruckData(), setTime(), writeCASTORData(), and writeSAFTData().

4.53.4.18 int Vehicle::m_Min [protected]

Definition at line 112 of file Vehicle.h.

Referenced by createCASTORVehicle(), createSAFTVehicle(), getTime(), Truck::returnTruckData(), setTime(), writeCASTORData(), and writeSAFTData().

4.53.4.19 int Vehicle::m_Sec [protected]

Definition at line 113 of file Vehicle.h.

Referenced by createCASTORVehicle(), createSAFTVehicle(), getTime(), Truck::returnTruckData(), setTime(), writeCASTORData(), and writeSAFTData().

4.53.4.20 int Vehicle::m_Hndt [protected]

Definition at line 114 of file Vehicle.h.

Referenced by createCASTORVehicle(), createSAFTVehicle(), getTime(), Truck::returnTruckData(), setTime(), writeCASTORData(), and writeSAFTData().

4.53.4.21 double Vehicle::m_GVW [protected]

Definition at line 116 of file Vehicle.h.

Referenced by createCASTORVehicle(), createSAFTVehicle(), getDataString(), getGVW(), Truck::returnTruckData(), setGVW(), writeCASTORData(), and writeSAFTData().

4.53.4.22 int Vehicle::m_Trns [protected]

Definition at line 117 of file Vehicle.h.

Referenced by createCASTORVehicle(), Truck::returnTruckData(), and writeCASTORData().

4.53.4.23 int Vehicle::m_NoAxles [protected]

Definition at line 118 of file Vehicle.h.

Referenced by createCASTORVehicle(), createSAFTVehicle(), getDataString(), getNoAxles(), Truck::returnTruckData(), setAxles(), setID(), writeCASTORData(), and writeSAFTData().

4.53.4.24 double Vehicle::m_Length [protected]

Definition at line 119 of file Vehicle.h.

Referenced by createCASTORVehicle(), createSAFTVehicle(), getDataString(), getLength(), Truck::returnTruckData(), setLength(), writeCASTORData(), and writeSAFTData().

4.53.4.25 std::vector<Axle*> Vehicle::m_vAxles [protected]

Definition at line 120 of file Vehicle.h.

Referenced by getAS(), getAW(), setAS(), setAW(), setAxles(), setID(), Car::~Car(), and Truck::~Truck().

4.53.4.26 WORD Vehicle::m_ID [protected]

Definition at line 122 of file Vehicle.h.

Referenced by Car::Car(), createCASTORVehicle(), createSAFTVehicle(), getID(), and setID().

4.53.4.27 bool Vehicle::m_bLaneChange [protected]

Definition at line 123 of file Vehicle.h.

Referenced by getChangeStatus(), and setChangeStatus().

4.53.4.28 double Vehicle::m_Tdelay [protected]

Definition at line 124 of file Vehicle.h.

Referenced by Car::Car(), changeTime(), createCASTORVehicle(), and createSAFTVehicle().

4.53.4.29 int Vehicle::m_TotalNoLanesInRoad [protected]

Definition at line 125 of file Vehicle.h.

Referenced by getLaneNoInDirection(), and setTotalNoLanesInRoad().

4.53.4.30 int Vehicle::ROAD_END_BUFFER [protected]

Definition at line 127 of file Vehicle.h.

Referenced by calcAccel(), and Vehicle().

4.53.4.31 double Vehicle::T_DELAY [protected]

Definition at line 128 of file Vehicle.h.

Referenced by changeTime(), and Vehicle().

4.53.4.32 double Vehicle::MIN_SPACE_FOR_NEXT_VEHICLE [protected]

Definition at line 129 of file Vehicle.h.

Referenced by LaneChangeAdvantage(), and Vehicle().

4.53.4.33 double Vehicle::SAFE_BRAKING [protected]

Definition at line 130 of file Vehicle.h.

Referenced by LaneChangeAdvantage(), and Vehicle().

4.53.4.34 double Vehicle::CRANE_AVERAGE_SPACING [protected]

Definition at line 131 of file Vehicle.h.

Referenced by setID(), and Vehicle().

4.53.4.35 double Vehicle::CRANE_MAX_SPACING [protected]

Definition at line 132 of file Vehicle.h.

Referenced by Vehicle().

4.53.4.36 double Vehicle::LOWLOADER_MIN_MAX_SPACING [protected]

Definition at line 133 of file Vehicle.h.

Referenced by setID(), and Vehicle().

4.53.4.37 int Vehicle::SMALL_TRUCK_NO_AXLES [protected]

Definition at line 134 of file Vehicle.h.

Referenced by setID(), and Vehicle().

4.53.4.38 int Vehicle::DAYS_PER_MT [protected]

Definition at line 135 of file Vehicle.h.

Referenced by getTime(), setTime(), and Vehicle().

4.53.4.39 int Vehicle::MTS_PER_YR [protected]

Definition at line 136 of file Vehicle.h.

Referenced by getTime(), setTime(), and Vehicle().

The documentation for this class was generated from the following files:

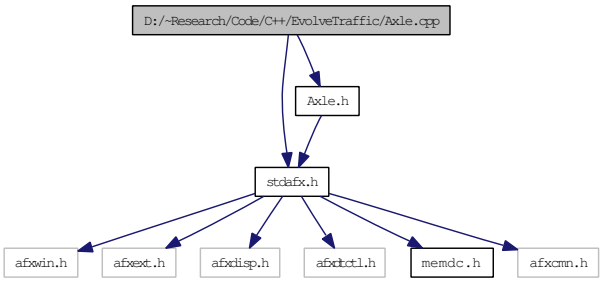
- D:/~Research/Code/C++/EvolveTraffic/Vehicle.h
- D:/~Research/Code/C++/EvolveTraffic/Vehicle.cpp

5 File Documentation

5.1 D:/~Research/Code/C++/EvolveTraffic/Axle.cpp File Reference

```
#include "stdafx.h"
#include "Axle.h"
```

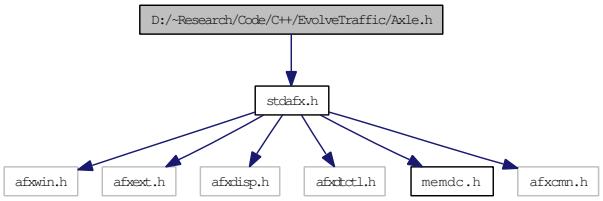
Include dependency graph for Axle.cpp:



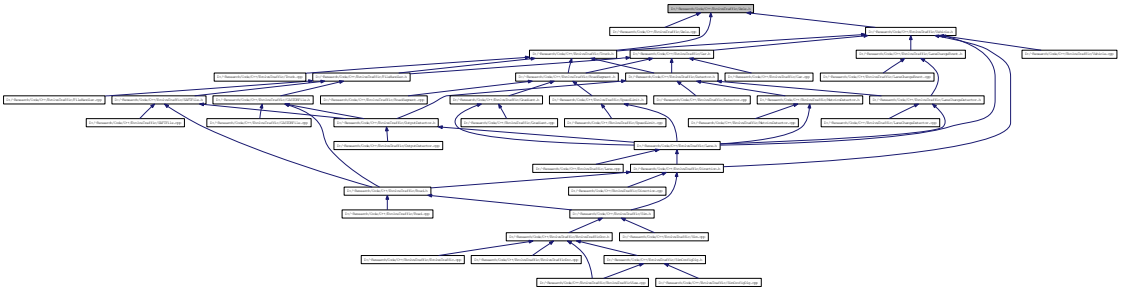
5.2 D:/~Research/Code/C++/EvolveTraffic/Axle.h File Reference

```
#include "stdafx.h"
```

Include dependency graph for Axle.h:



This graph shows which files directly or indirectly include this file:



Classes

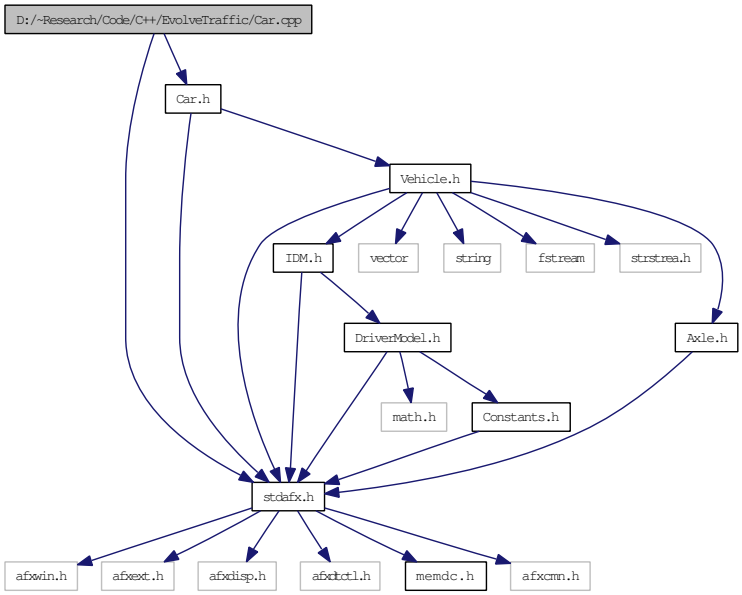
- class Axle

  A class to represent an axle of a vehicle.

5.3 D:/~Research/Code/C++/EvolveTraffic/Car.cpp File Reference

```
#include "stdafx.h"
#include "Car.h"
```

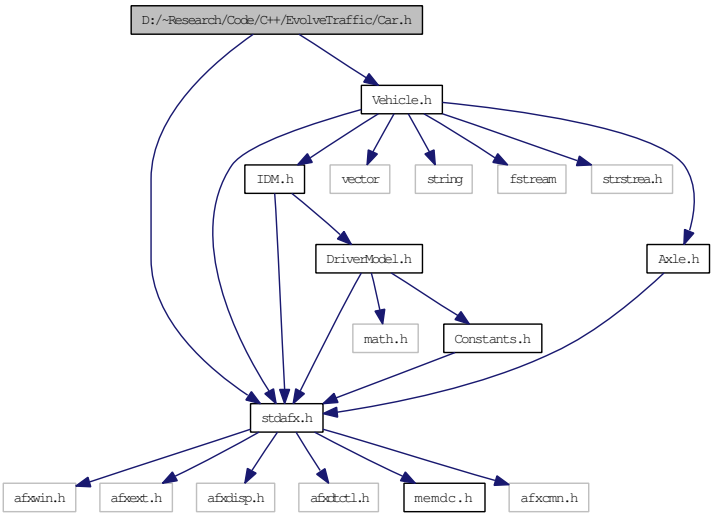
Include dependency graph for Car.cpp:



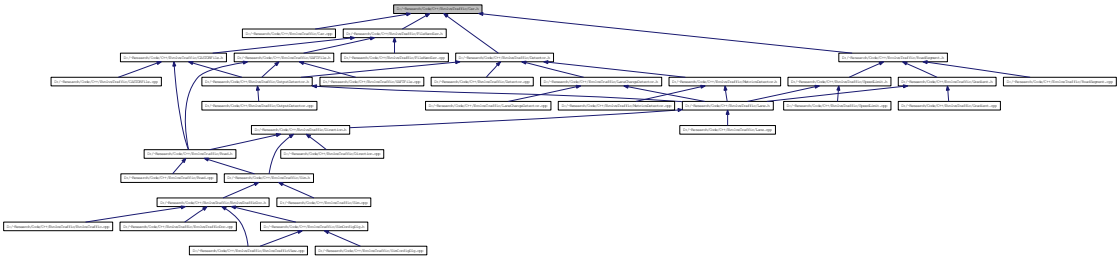
5.4 D:/~Research/Code/C++/EvolveTraffic/Car.h File Reference

```
#include "stdafx.h"
#include "Vehicle.h"
```

Include dependency graph for Car.h:



This graph shows which files directly or indirectly include this file:



Classes

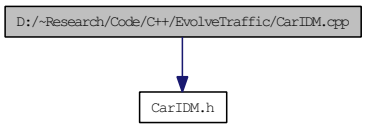
- class Car

  A class representing a car vehicle.

5.5 D:/~Research/Code/C++/EvolveTraffic/CarIDM.cpp File Reference

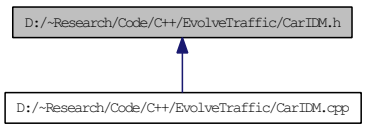
```
#include "CarIDM.h"
```

Include dependency graph for CarIDM.cpp:



5.6 D:/~Research/Code/C++/EvolveTraffic/CarIDM.h File Reference

This graph shows which files directly or indirectly include this file:

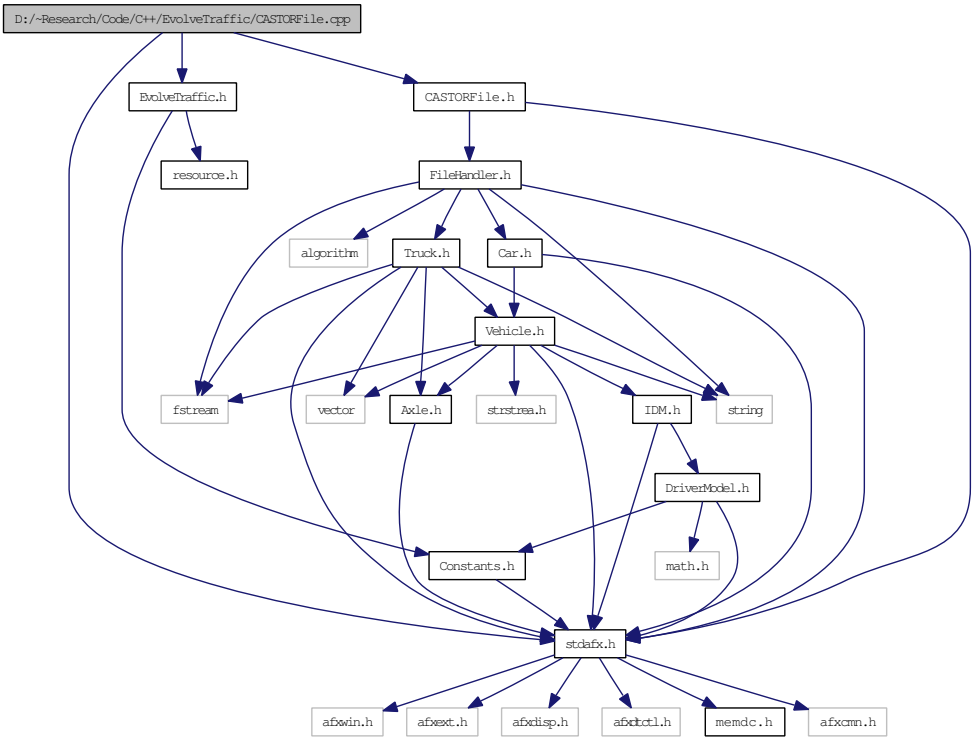


5.7 D:/~Research/Code/C++/EvolveTraffic/CASTORFile.cpp File Reference

```
#include "stdafx.h"
```

```
#include "EvolveTraffic.h"
#include "CASTORFile.h"
```

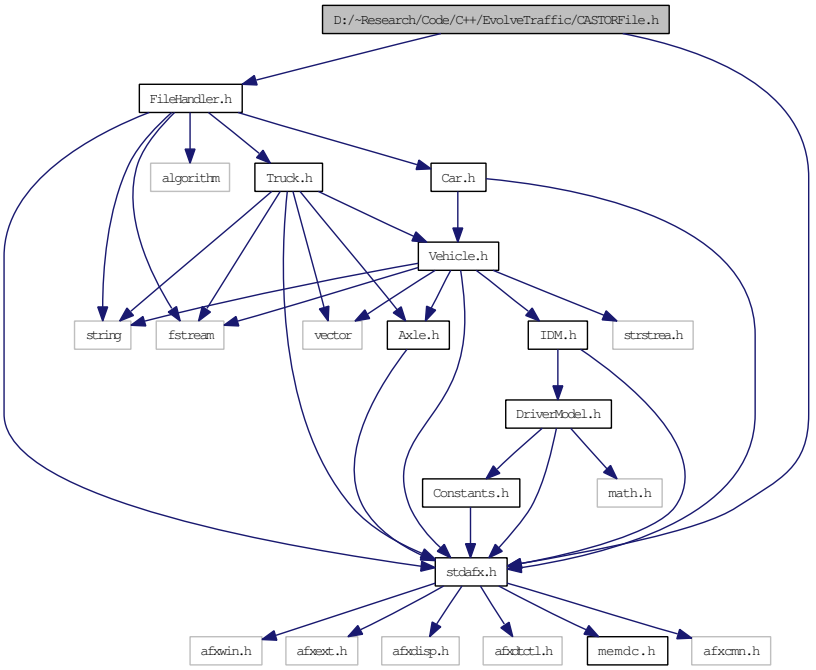
Include dependency graph for CASTORFile.cpp:



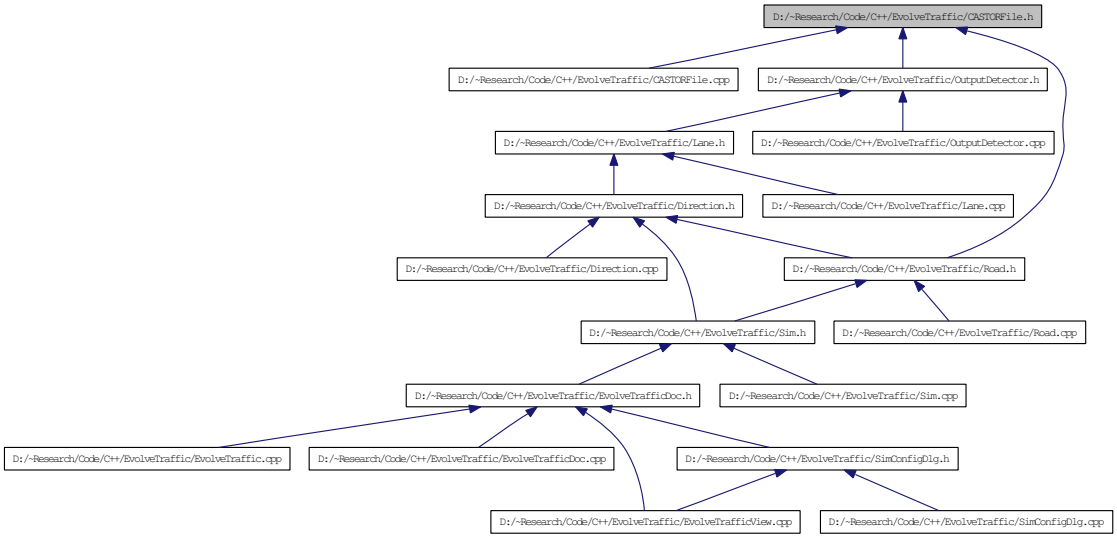
5.8 D:/~Research/Code/C++/EvolveTraffic/CASTORFile.h File Reference

```
#include "stdafx.h"
#include "FileHandler.h"
```

Include dependency graph for CASTORFile.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CASTORFile

A class for reading from, and writing to, CASTOR format files.

Defines

- #define AFX_CASTORFILE_H__A65FA0F3_9CD8_4697_A7B5_-E3B76744C37B__INCLUDED_

5.8.1 Define Documentation

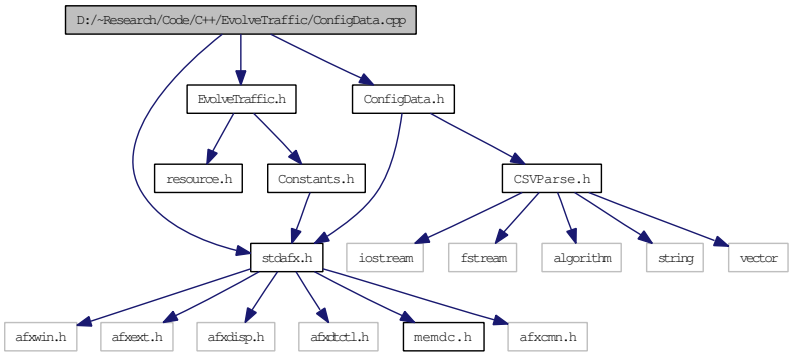
5.8.1.1 #define AFX_CASTORFILE_H__A65FA0F3_9CD8_4697_A7B5_-E3B76744C37B__INCLUDED_

Definition at line 6 of file CASTORFile.h.

5.9 D:/~Research/Code/C++/EvolveTraffic/ConfigData.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "ConfigData.h"
```

Include dependency graph for ConfigData.cpp:



Variables

- CConfigData g_ConfigData

5.9.1 Variable Documentation

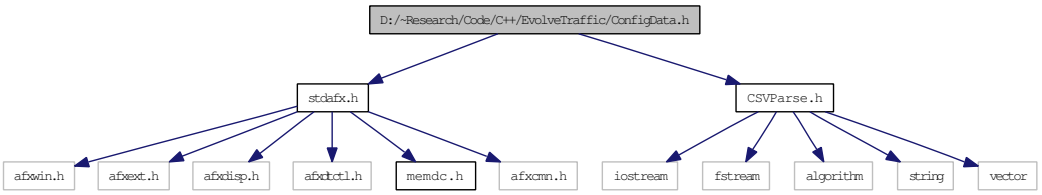
5.9.1.1 CConfigData g_ConfigData

Definition at line 32 of file ConfigData.cpp.

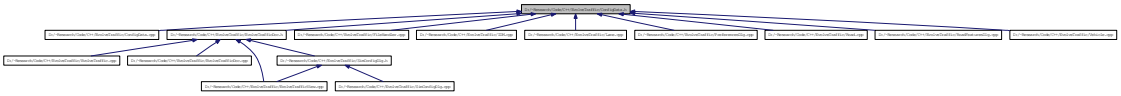
5.10 D:/~Research/Code/C++/EvolveTraffic/ConfigData.h File Reference

```
#include "stdafx.h"
#include "CSVParse.h"
```

Include dependency graph for ConfigData.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CConfigData

  A class for containing all of the configuration prameters for the program.

- struct CConfigData::IDM_Config
- struct CConfigData::Road_Config
- struct CConfigData::Road_Config::RoadFeatures_Config
- struct CConfigData::Road_Config::RoadFeatures_Config::SpeedLimit_Config
- struct CConfigData::Road_Config::RoadFeatures_Config::Gradient_Config
- struct CConfigData::Road_Config::RoadFeatures_Config::Gradient_-Config::IDM_Param_Modifiers
- struct CConfigData::Time_Config
- struct CConfigData::VehicleID_Config
- struct CConfigData::View_Config
- struct CConfigData::View_Config::View_Colours
- struct CConfigData::View_Config::View_Colours::Col_Vehicles
- struct CConfigData::View_Config::View_Colours::Col_DrawElements

Defines

- #define AFX_CONFIGDATA_H__F17D3AEB_F9C5_4B29_A869_-E50D84EA9B0A__INCLUDED_

5.10.1 Define Documentation

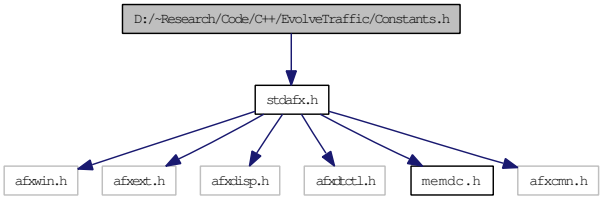
5.10.1.1 #define AFX_CONFIGDATA_H__F17D3AEB_F9C5_4B29_A869_-E50D84EA9B0A__INCLUDED_

Definition at line 6 of file ConfigData.h.

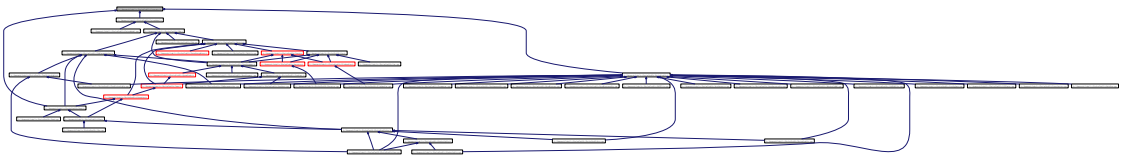
5.11 D:/~Research/Code/C++/EvolveTraffic/Constants.h File Reference

```
#include "stdafx.h"
```

Include dependency graph for Constants.h:



This graph shows which files directly or indirectly include this file:



Variables

- const WORD DIST_EXPONENTIAL = 200U
- const WORD DIST_LOGNORMAL = 201U
- const WORD DIST_GAMMA = 202U
- const WORD DIST_GUMBEL = 203U
- const WORD DIST_POISSON = 204U
- const WORD DIST_GEV = 205U
- const WORD DIST_NORMAL = 206U
- const WORD DIST_CONST = 207U
- const WORD IDM_PARAM_T = 300U
- const WORD IDM_PARAM_A = 301U
- const WORD IDM_PARAM_B = 302U
- const WORD IDM_PARAM_S0 = 303U
- const WORD IDM_PARAM_S1 = 304U
- const WORD IDM_PARAM_V0 = 305U

- const WORD IDM_PARAM_DELTA = 306U
- const WORD IDM_PARAM_POLITE = 307U
- const WORD IDM_PARAM_BIAS = 308U
- const WORD IDM_PARAM_DELTAATH = 309U
- const WORD VEH_ID_CAR = 0
- const WORD VEH_ID_SMALLTRUCK = 1
- const WORD VEH_ID_LARGETRUCK = 2
- const WORD VEH_ID_CRANE = 3
- const WORD VEH_ID_LOWLOADER = 4
- const WORD FEAT_SPEEDLIMIT = 400U
- const WORD FEAT_GRADIENT = 401U
- const int CASTOR_MAX_AXLES = 9
- const UINT VERSION_NUMBER = 1
- const double M_PER_S_TO_KM_PER_H = 3.6
- const double KM_PER_H_TO_M_PER_S = 0.27777778
- const double KG100_TO_KN = 0.981
- const int SECS_PER_HOUR = 3600
- const int HOURS_PER_DAY = 24
- const int MINS_PER_HOUR = 60
- const int SECS_PER_MIN = 60
- const int FIXED_ROWS = 1
- const int FIXED_COLUMNS = 1
- const WORD METRICS_VEH_ALL = 500U
- const WORD METRICS_VEH_CAR = 501U
- const WORD METRICS_VEH_SMALLTRUCK = 502U
- const WORD METRICS_VEH_LARGETRUCK = 503U
- const WORD METRICS_VEH_CRANE = 504U
- const WORD METRICS_VEH_LOWLOADER = 505U
- const WORD METRICS_TYPE_FLOWDENSITY = 510U
- const WORD METRICS_TYPE_HEADWAY = 511U
- const WORD METRICS_TYPE_COMPOSITION = 512U
- const WORD METRICS_TYPE_LANE_CHANGE = 513U
- const WORD SAFT = 100U
- const WORD CASTOR = 101U
- const WORD INPUT_FILE_BUFFER_SIZE = 10U
- const WORD DRAW_FRAMES_PER_SEC = 20U

5.11.1 Variable Documentation

5.11.1.1 const WORD CASTOR = 101U

Definition at line 82 of file Constants.h.

Referenced by CSimConfigDlg::CSimConfigDlg(), Road::initTruckGroup(), CSimConfigDlg::OnInitDialog(), CEvolveTrafficDoc::OnNewDocument(), and CSimConfigDlg::OnRadCastor().

5.11.1.2 const int CASTOR_MAX_AXLES = 9

Definition at line 46 of file Constants.h.

Referenced by Vehicle::writeCASTORData().

5.11.1.3 const WORD DIST_CONST = 207U

Definition at line 20 of file Constants.h.

Referenced by CDistribution::Generate(), CTrafficConfigDlg::MapDistributionID(), CTrafficConfigDlg::MapDistributionString(), and CParameter::SetDefaultParams().

5.11.1.4 const WORD DIST_EXPONENTIAL = 200U

Definition at line 13 of file Constants.h.

Referenced by CDistribution::Generate(), CTrafficConfigDlg::MapDistributionID(), and CTrafficConfigDlg::MapDistributionString().

5.11.1.5 const WORD DIST_GAMMA = 202U

Definition at line 15 of file Constants.h.

Referenced by CDistribution::Generate(), CTrafficConfigDlg::MapDistributionID(), and CTrafficConfigDlg::MapDistributionString().

5.11.1.6 const WORD DIST_GEV = 205U

Definition at line 18 of file Constants.h.

Referenced by CDistribution::Generate(), CTrafficConfigDlg::MapDistributionID(), and CTrafficConfigDlg::MapDistributionString().

5.11.1.7 const WORD DIST_GUMBEL = 203U

Definition at line 16 of file Constants.h.

Referenced by CDistribution::Generate(), CTrafficConfigDlg::MapDistributionID(), and CTrafficConfigDlg::MapDistributionString().

5.11.1.8 const WORD DIST_LOGNORMAL = 201U

Definition at line 14 of file Constants.h.

Referenced by CDistribution::Generate(), CTrafficConfigDlg::MapDistributionID(), and CTrafficConfigDlg::MapDistributionString().

5.11.1.9 const WORD DIST_NORMAL = 206U

Definition at line 19 of file Constants.h.

Referenced by CDistribution::CDistribution(), CDistribution::Generate(), CTrafficConfigDlg::MapDistributionID(), and CTrafficConfigDlg::MapDistributionString().

5.11.1.10 const WORD DIST_POISSON = 204U

Definition at line 17 of file Constants.h.

Referenced by CDistribution::Generate(), CTrafficConfigDlg::MapDistributionID(), and CTrafficConfigDlg::MapDistributionString().

5.11.1.11 const WORD DRAW_FRAMES_PER_SEC = 20U

Definition at line 88 of file Constants.h.

Referenced by CEvolveTrafficView::CEvolveTrafficView().

5.11.1.12 const WORD FEAT_GRADIENT = 401U

Definition at line 43 of file Constants.h.

Referenced by CEvolveTrafficView::DrawLegend(), CEvolveTrafficView::DrawRoadSegments(), CEvolveTrafficView::DrawSingleSegment(), CRoadFeaturesDlg::MapStringToType(), CRoadFeaturesDlg::MapTypeToString(), CRoadFeaturesDlg::OnValidate(), and Road::SetSegmentFromFeature().

5.11.1.13 const WORD FEAT_SPEEDLIMIT = 400U

Definition at line 42 of file Constants.h.

Referenced by CRoadFeature::CRoadFeature(), CEvolveTrafficView::DrawLegend(), CEvolveTrafficView::DrawRoadSegments(), CEvolveTrafficView::DrawSingleSegment(), CRoadFeaturesDlg::LoadFeaturesIntoGrid(), CRoadFeaturesDlg::MapStringToType(), CRoadFeaturesDlg::MapTypeToString(), CRoadFeaturesDlg::OnValidate(), CRoadFeaturesDlg::SetParamData(), and Road::SetSegmentFromFeature().

5.11.1.14 const int FIXED_COLUMNS = 1

Definition at line 65 of file Constants.h.

Referenced by CPreferencesDlg::CPreferencesDlg(), CRoadFeaturesDlg::CRoadFeaturesDlg(), CStatDetectorDlg::CStatDetectorDlg(), and CTrafficConfigDlg::CTrafficConfigDlg().

5.11.1.15 const int FIXED_ROWS = 1

Definition at line 64 of file Constants.h.

Referenced by CPreferencesDlg::CPreferencesDlg(), CRoadFeaturesDlg::CRoadFeaturesDlg(), CStatDetectorDlg::CStatDetectorDlg(), and CTrafficConfigDlg::CTrafficConfigDlg().

5.11.1.16 const int HOURS_PER_DAY = 24

Definition at line 59 of file Constants.h.

Referenced by CEvolveTrafficView::DrawTimer(), Vehicle::getTime(), and Vehicle::setTime().

5.11.1.17 const WORD IDM_PARAM_A = 301U

Definition at line 24 of file Constants.h.

Referenced by CIDMParameterSet::buildParamSet(), and CParameter::SetDefaultParams().

5.11.1.18 const WORD IDM_PARAM_B = 302U

Definition at line 25 of file Constants.h.

Referenced by CIDMParameterSet::buildParamSet(), and CParameter::SetDefaultParams().

5.11.1.19 const WORD IDM_PARAM_BIAS = 308U

Definition at line 31 of file Constants.h.

Referenced by CIDMParameterSet::buildParamSet(), and CParameter::SetDefaultParams().

5.11.1.20 const WORD IDM_PARAM_DELTA = 306U

Definition at line 29 of file Constants.h.

Referenced by CIDMParameterSet::buildParamSet(), and CParameter::SetDefaultParams().

5.11.1.21 const WORD IDM_PARAM_DELTAATH = 309U

Definition at line 32 of file Constants.h.

Referenced by CIDMParameterSet::buildParamSet(), and CParameter::SetDefaultParams().

5.11.1.22 const WORD IDM_PARAM_POLITE = 307U

Definition at line 30 of file Constants.h.

Referenced by CIDMParameterSet::buildParamSet(), and CParameter::SetDefaultParams().

5.11.1.23 const WORD IDM_PARAM_S0 = 303U

Definition at line 26 of file Constants.h.

Referenced by CIDMParameterSet::buildParamSet(), and CParameter::SetDefaultParams().

5.11.1.24 const WORD IDM_PARAM_S1 = 304U

Definition at line 27 of file Constants.h.

Referenced by CIDMParameterSet::buildParamSet(), and CParameter::SetDefaultParams().

5.11.1.25 const WORD IDM_PARAM_T = 300U

Definition at line 23 of file Constants.h.

Referenced by CIDMParameterSet::buildParamSet(), and CParameter::SetDefaultParams().

5.11.1.26 const WORD IDM_PARAM_V0 = 305U

Definition at line 28 of file Constants.h.

Referenced by CIDMParameterSet::buildParamSet(), and CParameter::SetDefaultParams().

5.11.1.27 const WORD INPUT_FILE_BUFFER_SIZE = 10U

Definition at line 85 of file Constants.h.

Referenced by Road::init().

5.11.1.28 const double KG100_TO_KN = 0.981

Definition at line 56 of file Constants.h.

Referenced by Vehicle::createCASTORVehicle(), SAFTFile::readLine(), CASTORFile::readLine(), Vehicle::writeCASTORData(), and Vehicle::writeSAFTData().

5.11.1.29 const double KM_PER_H_TO_M_PER_S = 0.27777778

Definition at line 53 of file Constants.h.

Referenced by SpeedLimit::addVehicle(), IDM::set_V0(), IDM::SetGradient(), and CRoadFeaturesDlg::SetParamData().

5.11.1.30 const double M_PER_S_TO_KM_PER_H = 3.6

Definition at line 52 of file Constants.h.

Referenced by MetricsDetector::doFlowDensityOutput(), CEvolveTrafficView::DrawRoadSegments(), CEvolveTrafficView::DrawVehicle(), Vehicle::getDataString(), CRoadFeaturesDlg::LoadFeaturesIntoGrid(), and CRoadFeaturesDlg::OnValidate().

5.11.1.31 const WORD METRICS_TYPE_COMPOSITION = 512U

Definition at line 77 of file Constants.h.

Referenced by MetricsDetector::AddCurrentVehicle(), MetricsDetector::doIntervalOutput(), CEvolveTrafficView::DrawDetectors(), MetricsDetector::InitOutputFiles(), CStatDetectorDlg::MapDetTypeToString(), and CStatDetectorDlg::MapStringToDetType().

5.11.1.32 const WORD METRICS_TYPE_FLOWDENSITY = 510U

Definition at line 75 of file Constants.h.

Referenced by MetricsDetector::AddCurrentVehicle(), CStatDetector::CStatDetector(), MetricsDetector::doIntervalOutput(), CEvolveTrafficView::DrawDetectors(), MetricsDetector::InitOutputFiles(), CStatDetectorDlg::MapDetTypeToString(), and CStatDetectorDlg::MapStringToDetType().

5.11.1.33 const WORD METRICS_TYPE_HEADWAY = 511U

Definition at line 76 of file Constants.h.

Referenced by MetricsDetector::AddCurrentVehicle(), MetricsDetector::doIntervalOutput(), CEvolveTrafficView::DrawDetectors(), MetricsDetector::InitOutputFiles(), CStatDetectorDlg::MapDetTypeToString(), and CStatDetectorDlg::MapStringToDetType().

5.11.1.34 const WORD METRICS_TYPE_LANE_CHANGE = 513U

Definition at line 78 of file Constants.h.

Referenced by CEvolveTrafficView::DrawDetectors(), Lane::Lane(), LaneChangeDetector::LaneChangeDetector(), CStatDetectorDlg::MapDetTypeToString(), CStatDetectorDlg::MapStringToDetType(), CStatDetectorDlg::OnValidate(), and Road::SetMetricDetFromStatDet().

5.11.1.35 const WORD METRICS_VEH_ALL = 500U

Definition at line 68 of file Constants.h.

Referenced by MetricsDetector::AddCurrentVehicle(), LaneChangeDetector::addEvent(), MetricsDetector::addVehicle(), CStatDetector::CStatDetector(), MetricsDetector::doIntervalOutput(), LaneChangeDetector::EndOutput(), MetricsDetector::InitOutputFiles(), LaneChangeDetector::InitOutputFiles(), Detector::MapDetVehTypeToString(), CStatDetectorDlg::MapStringToVehType(), and CStatDetectorDlg::MapVehTypeToString().

5.11.1.36 const WORD METRICS_VEH_CAR = 501U

Definition at line 69 of file Constants.h.

Referenced by LaneChangeDetector::addEvent(), MetricsDetector::addVehicle(), Detector::MapDetVehTypeToString(), CStatDetectorDlg::MapStringToVehType(), and CStatDetectorDlg::MapVehTypeToString().

5.11.1.37 const WORD METRICS_VEH_CRANE = 504U

Definition at line 72 of file Constants.h.

Referenced by LaneChangeDetector::addEvent(), MetricsDetector::addVehicle(), Detector::MapDetVehTypeToString(), CStatDetectorDlg::MapStringToVehType(), and CStatDetectorDlg::MapVehTypeToString().

5.11.1.38 const WORD METRICS_VEH_LARGETRUCK = 503U

Definition at line 71 of file Constants.h.

Referenced by LaneChangeDetector::addEvent(), MetricsDetector::addVehicle(), Detector::MapDetVehTypeToString(), CStatDetectorDlg::MapStringToVehType(), and CStatDetectorDlg::MapVehTypeToString().

5.11.1.39 const WORD METRICS_VEH_LOWLOADER = 505U

Definition at line 73 of file Constants.h.

Referenced by LaneChangeDetector::addEvent(), MetricsDetector::addVehicle(), Detector::MapDetVehTypeToString(), CStatDetectorDlg::MapStringToVehType(), and CStatDetectorDlg::MapVehTypeToString().

5.11.1.40 const WORD METRICS_VEH_SMALLTRUCK = 502U

Definition at line 70 of file Constants.h.

Referenced by LaneChangeDetector::addEvent(), MetricsDetector::addVehicle(), Detector::MapDetVehTypeToString(), CStatDetectorDlg::MapStringToVehType(), and CStatDetectorDlg::MapVehTypeToString().

5.11.1.41 const int MINS_PER_HOUR = 60

Definition at line 60 of file Constants.h.

Referenced by CEvolveTrafficView::DrawTimer(), and Vehicle::setTime().

5.11.1.42 const WORD SAFT = 100U

Definition at line 81 of file Constants.h.

Referenced by Road::initTruckGroup(), CSimConfigDlg::OnInitDialog(), and CSimConfigDlg::OnRadSaft().

5.11.1.43 const int SECS_PER_HOUR = 3600

Definition at line 58 of file Constants.h.

Referenced by MetricsDetector::doFlowDensityOutput(), LaneChangeDetector::doRateOutput(), CEvolveTrafficView::DrawTimer(), Vehicle::getTime(), and Vehicle::setTime().

5.11.1.44 const int SECS_PER_MIN = 60

Definition at line 61 of file Constants.h.

Referenced by CEvolveTrafficView::DrawTimer(), and Vehicle::getTime().

5.11.1.45 const WORD VEH_ID_CAR = 0

Definition at line 35 of file Constants.h.

Referenced by LaneChangeDetector::addEvent(), MetricsDetector::addVehicle(), Car::Car(), CEvolveTrafficDoc::CEvolveTrafficDoc(), MetricsDetector::doCompositionOutput(), LaneChangeDetector::doCompositionOutput(), LaneChangeDetector::doVerboseOutput(), CEvolveTrafficView::DrawLegend(), CEvolveTrafficView::DrawVehicleAt(), RoadSegment::getIDMParams(), and Road::setIDMDriverModel().

5.11.1.46 const WORD VEH_ID_CRANE = 3

Definition at line 38 of file Constants.h.

Referenced by LaneChangeDetector::addEvent(), MetricsDetector::addVehicle(), CEvolveTrafficDoc::CEvolveTrafficDoc(), MetricsDetector::doCompositionOutput(), LaneChangeDetector::doCompositionOutput(), LaneChangeDetector::doVerboseOutput(), CEvolveTrafficView::DrawLegend(), CEvolveTrafficView::DrawVehicleAt(), RoadSegment::getIDMParams(), Vehicle::setID(), and Road::setIDMDriverModel().

5.11.1.47 const WORD VEH_ID_LARGETRUCK = 2

Definition at line 37 of file Constants.h.

Referenced by LaneChangeDetector::addEvent(), MetricsDetector::addVehicle(), CEvolveTrafficDoc::CEvolveTrafficDoc(), MetricsDetector::doCompositionOutput(), LaneChangeDetector::doCompositionOutput(), LaneChangeDetector::doVerboseOutput(), CEvolveTrafficView::DrawLegend(), CEvolveTrafficView::DrawVehicleAt(), RoadSegment::getIDMParams(), Vehicle::setID(), and Road::setIDMDriverModel().

5.11.1.48 const WORD VEH_ID_LOWLOADER = 4

Definition at line 39 of file Constants.h.

Referenced by LaneChangeDetector::addEvent(), MetricsDetector::addVehicle(), CEvolveTrafficDoc::CEvolveTrafficDoc(), MetricsDetector::doCompositionOutput(), LaneChangeDetector::doCompositionOutput(), LaneChangeDetector::doVerboseOutput(), CEvolveTrafficView::DrawLegend(), CEvolveTrafficView::DrawVehicleAt(), RoadSegment::getIDMParams(), Vehicle::setID(), and Road::setIDMDriverModel().

5.11.1.49 const WORD VEH_ID_SMALLTRUCK = 1

Definition at line 36 of file Constants.h.

Referenced by LaneChangeDetector::addEvent(), MetricsDetector::addVehicle(), CEvolveTrafficDoc::CEvolveTrafficDoc(), MetricsDetector::doCompositionOutput(), LaneChangeDetector::doCompositionOutput(), LaneChangeDetector::doVerboseOutput(), CEvolveTrafficView::DrawLegend(), CEvolveTrafficView::DrawVehicleAt(), RoadSegment::getIDMParams(), Vehicle::setID(), and Road::setIDMDriverModel().

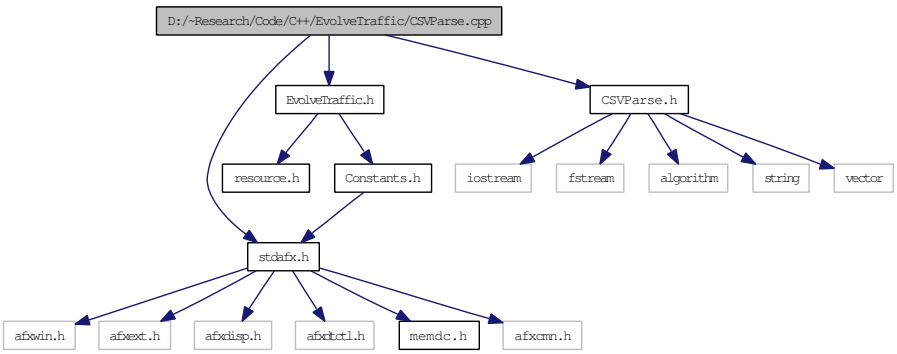
5.11.1.50 const UINT VERSION_NUMBER = 1

Definition at line 49 of file Constants.h.

5.12 D:/~Research/Code/C++/EvolveTraffic/CSVParse.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "CSVParse.h"
```

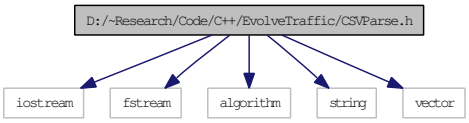
Include dependency graph for CSVParse.cpp:



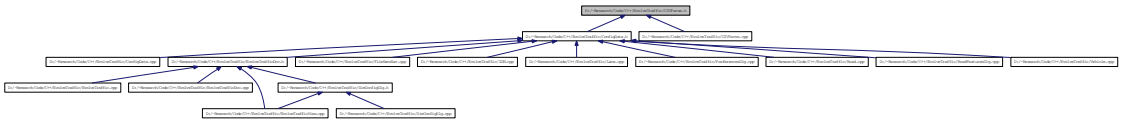
5.13 D:/~Research/Code/C++/EvolveTraffic/CSVParse.h File Reference

```
#include <iostream>
#include <fstream>
#include <algorithm>
#include <string>
#include <vector>
```

Include dependency graph for CSVParse.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CCSVParse

  A class to parse Comma Seperated Values input.

Defines

- #define AFX_CSVPARSE_H__A5DED40F_C9D8_4B26_9333_-14BF8D583637__INCLUDED_

5.13.1 Define Documentation

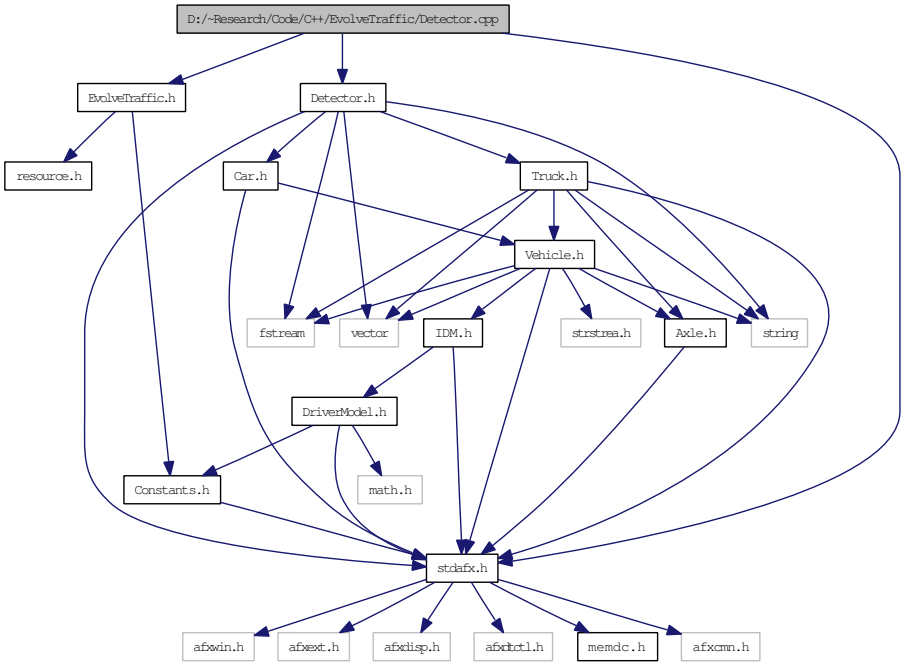
5.13.1.1 #define AFX_CSVPARSE_H__A5DED40F_C9D8_4B26_9333_-14BF8D583637__INCLUDED_

Definition at line 6 of file CSVParse.h.

5.14 D:/~Research/Code/C++/EvolveTraffic/Detector.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "Detector.h"
```

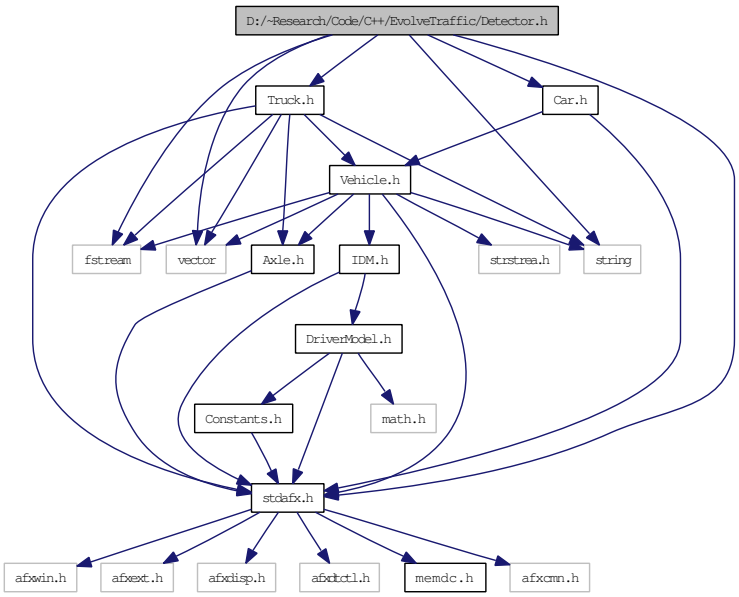
Include dependency graph for Detector.cpp:



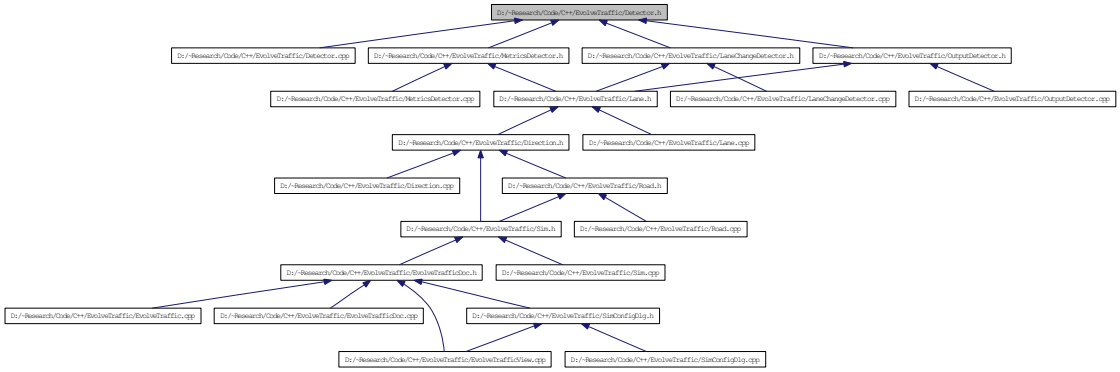
5.15 D:/~Research/Code/C++/EvolveTraffic/Detector.h File Reference

```
#include "stdafx.h"
#include <string>
#include <fstream>
#include <vector>
#include "Truck.h"
#include "Car.h"
```

Include dependency graph for Detector.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Detector

  A base class from which other, specific detectors are derived.

Defines

- #define AFX_DETECTOR_H__338D3ED7_2EE5_4DD1_9434_-3E210AB4E60C__INCLUDED_

5.15.1 Define Documentation

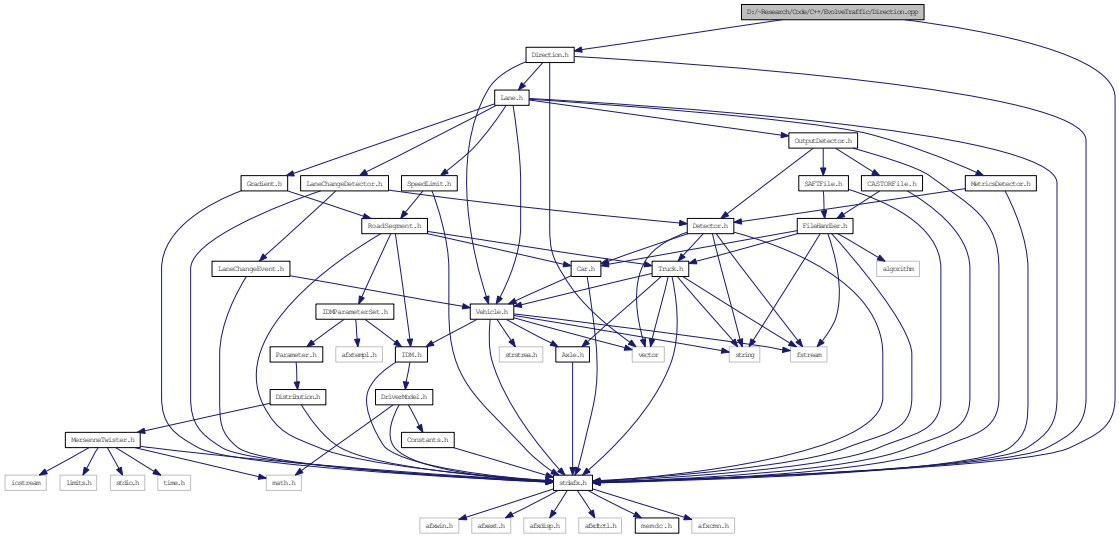
5.15.1.1 #define AFX_DETECTOR_H__338D3ED7_2EE5_4DD1_9434_-3E210AB4E60C__INCLUDED_

Definition at line 6 of file Detector.h.

5.16 D:/∼Research/Code/C++/EvolveTraffic/Direction.cpp File Reference

```
#include "stdafx.h"
#include "Direction.h"
```

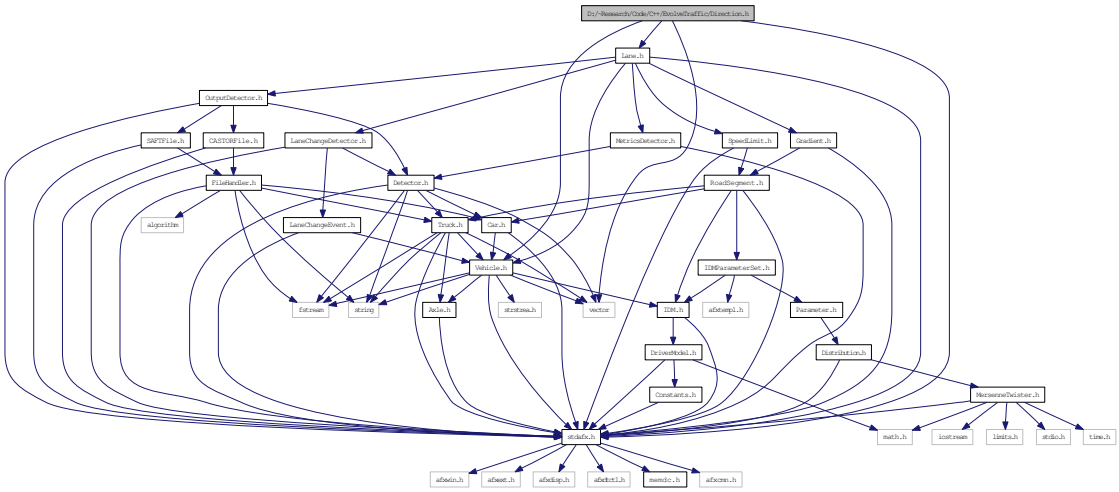
Include dependency graph for Direction.cpp:



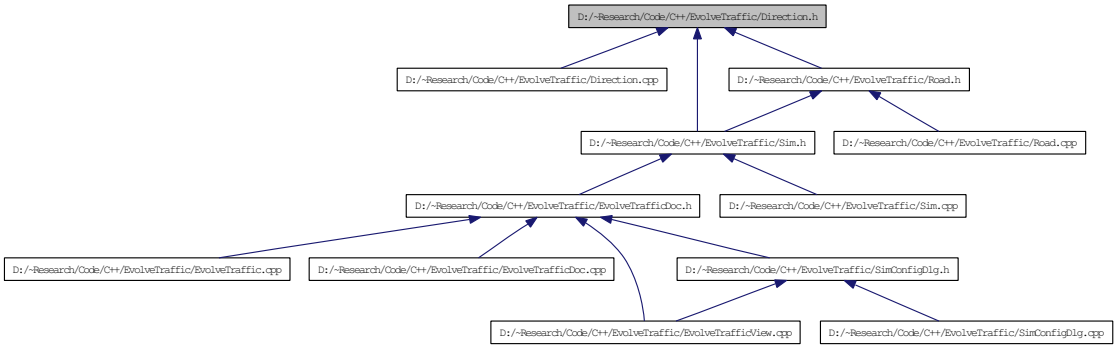
5.17 D:/∼Research/Code/C++/EvolveTraffic/Direction.h File Reference

```
#include "stdafx.h"
#include <vector>
#include "Vehicle.h"
#include "Lane.h"
```

Include dependency graph for Direction.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Direction

  A class representing a direction in a road.

Typedefs

- typedef std::vector< std::vector< Vehicle * > > M2D

5.17.1 Typedef Documentation

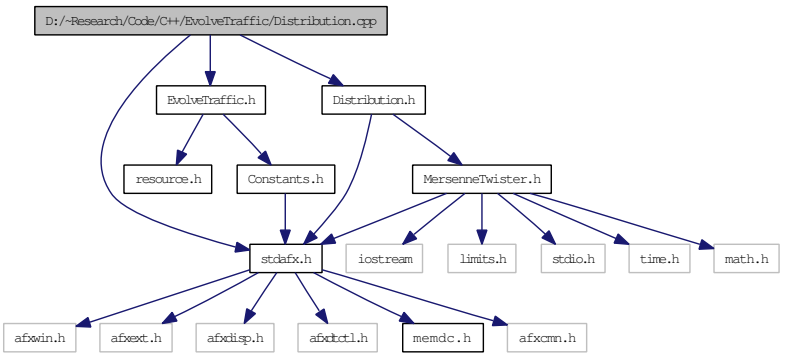
5.17.1.1 typedef std::vector< std::vector<Vehicle*> > M2D

Definition at line 9 of file Direction.h.

5.18 D:/~Research/Code/C++/EvolveTraffic/Distribution.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "Distribution.h"
```

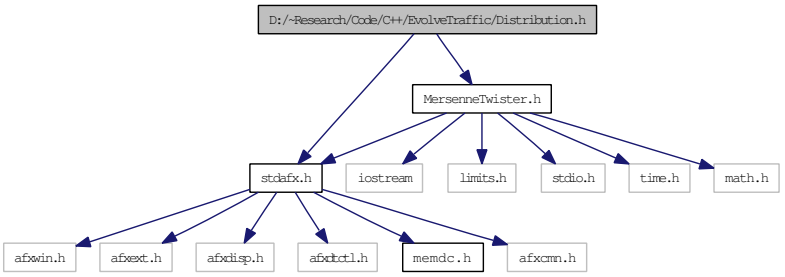
Include dependency graph for Distribution.cpp:



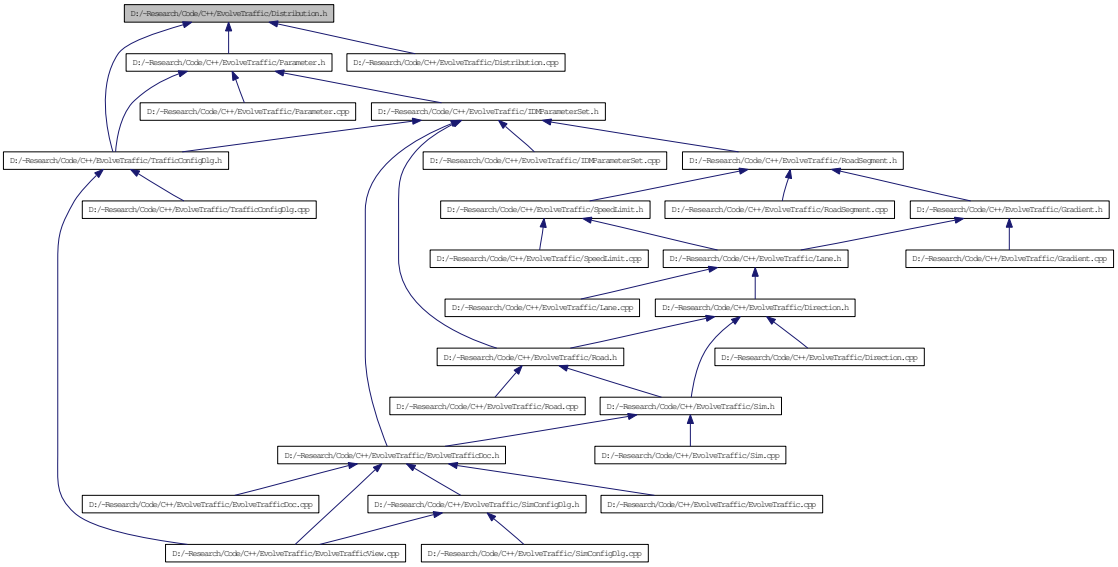
5.19 D:/~Research/Code/C++/EvolveTraffic/Distribution.h File Reference

```
#include "stdafx.h"
#include "MersenneTwister.h"
```

Include dependency graph for Distribution.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CDistribution

  A class representing a distribution that can be saved to file.

Defines

- #define AFX_DISTRIBUTION_H__EE052C75_E901_4AB0_85F4_-6FA67C4F40C5__INCLUDED_

5.19.1 Define Documentation

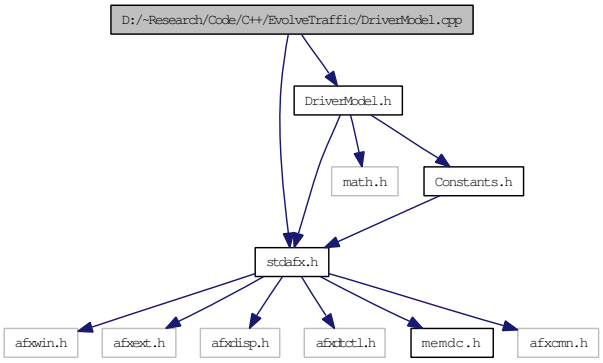
5.19.1.1 #define AFX_DISTRIBUTION_H__EE052C75_E901_4AB0_85F4_-6FA67C4F40C5__INCLUDED_

Definition at line 2 of file Distribution.h.

5.20 D:/~Research/Code/C++/EvolveTraffic/DriverModel.cpp File Reference

```
#include "stdafx.h"
#include "DriverModel.h"
```

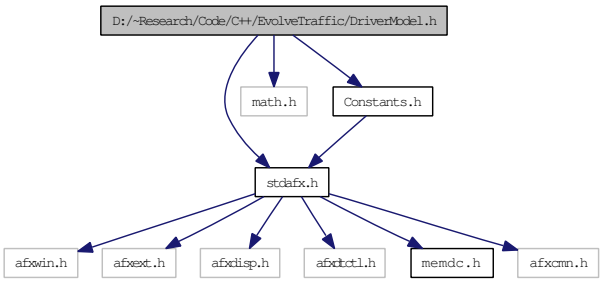
Include dependency graph for DriverModel.cpp:



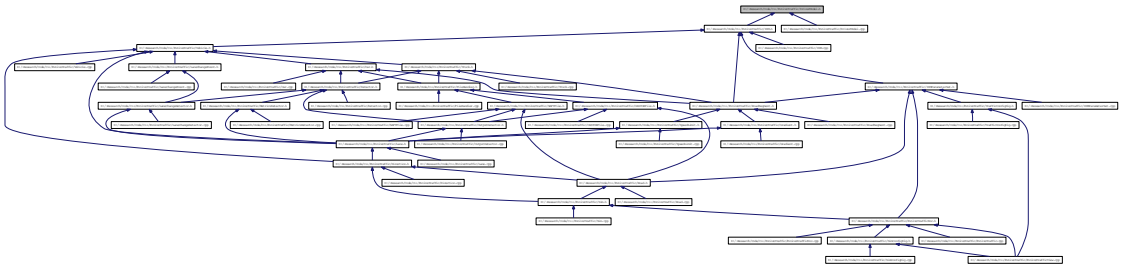
5.21 D:/~Research/Code/C++/EvolveTraffic/DriverModel.h File Reference

```
#include "stdafx.h"
#include <math.h>
#include "Constants.h"
```

Include dependency graph for DriverModel.h:



This graph shows which files directly or indirectly include this file:



Classes

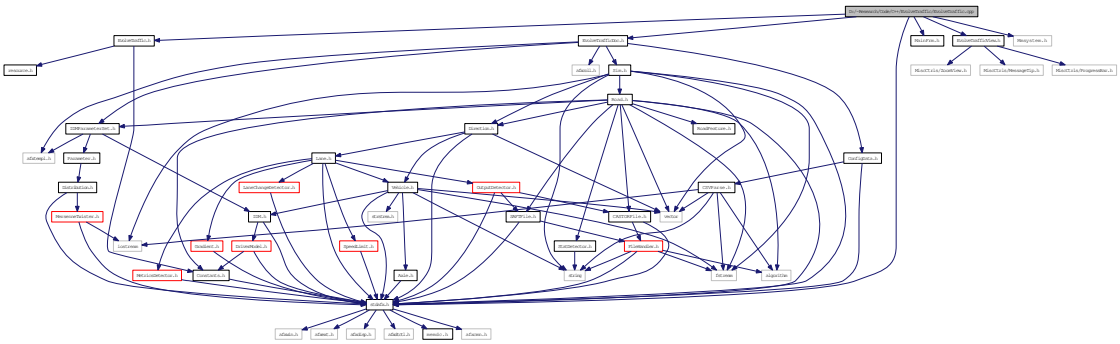
- class DriverModel

  A base class for representing driver models.

5.22 D:/~Research/Code/C++/EvolveTraffic/EvolveTraffic.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "MainFrm.h"
#include "EvolveTrafficDoc.h"
#include "EvolveTrafficView.h"
#include <Mmsystem.h>
```

Include dependency graph for EvolveTraffic.cpp:



Classes

- class CAboutDlg

  A class for the About Dialog.

Variables

- CConfigData g_ConfigData
- CEvolveTrafficApp theApp

5.22.1 Variable Documentation

5.22.1.1 CConfigData g_ConfigData

Definition at line 32 of file ConfigData.cpp.

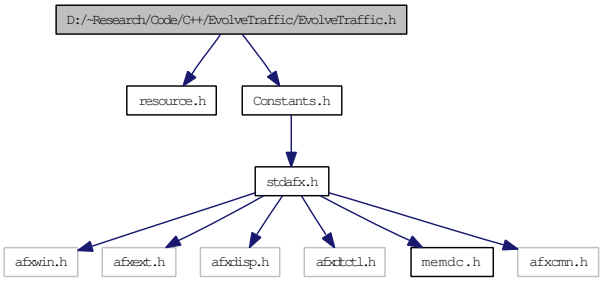
5.22.1.2 CEvolveTrafficApp theApp

Definition at line 51 of file EvolveTraffic.cpp.

5.23 D:/~Research/Code/C++/EvolveTraffic/EvolveTraffic.h File Reference

```
#include "resource.h"
#include "Constants.h"
```

Include dependency graph for EvolveTraffic.h:



This graph shows which files directly or indirectly include this file:

Classes

- class CEvolveTrafficApp

  A class for the applicaiton object.

Defines

- #define AFX_EVOLVETRAFFIC_H__9F6D6F9F_447A_4531_BDB2_3C60611EC34E__INCLUDED_

5.23.1 Define Documentation

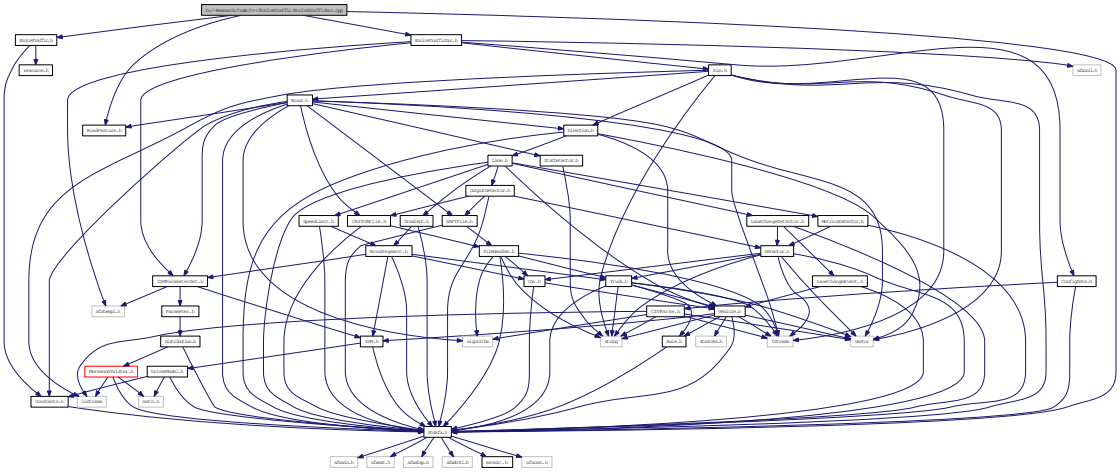
5.23.1.1 #define AFX_EVOLVETRAFFIC_H__9F6D6F9F_447A_4531_BDB2_3C60611EC34E__INCLUDED_

Definition at line 5 of file EvolveTraffic.h.

5.24 D:/∼Research/Code/C++/EvolveTraffic/EvolveTrafficDoc.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "EvolveTrafficDoc.h"
#include "RoadFeature.h"
```

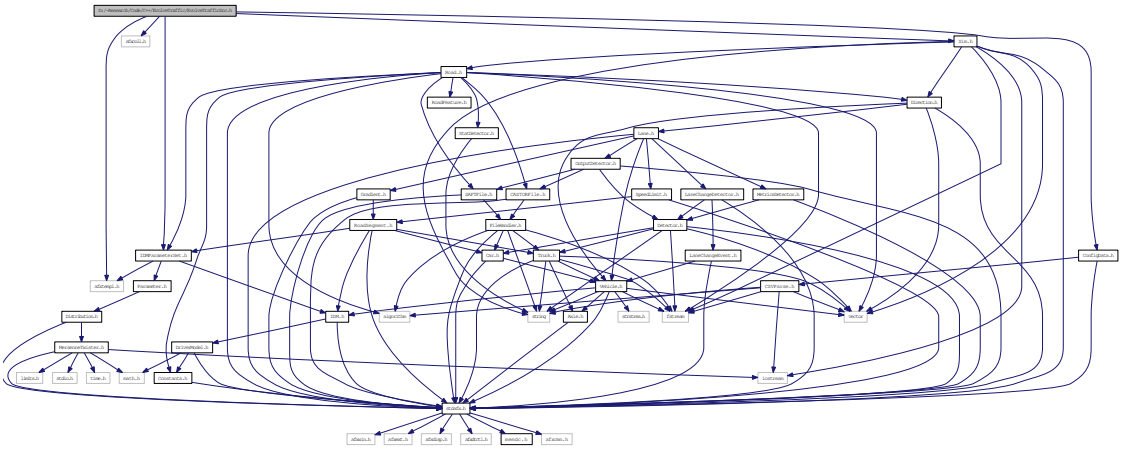
Include dependency graph for EvolveTrafficDoc.cpp:



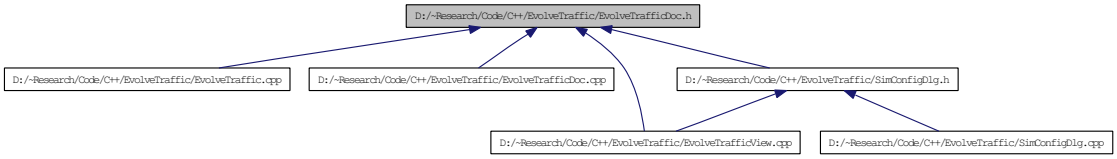
5.25 D:/∼Research/Code/C++/EvolveTraffic/EvolveTrafficDoc.h File Reference

```
#include "afxtempl.h"
#include "afxcoll.h"
#include "Sim.h"
#include "IDMParameterSet.h"
#include "ConfigData.h"
```

Include dependency graph for EvolveTrafficDoc.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CEvolveTrafficDoc

  A class that stores data relating to the parameters of a simulation.

Defines

- #define AFX_EVOLVETRAFFICDOC_H__E9AF4E58_CB5C_4E2F_94AC_-B1498A1C476A__INCLUDED_

5.25.1 Define Documentation

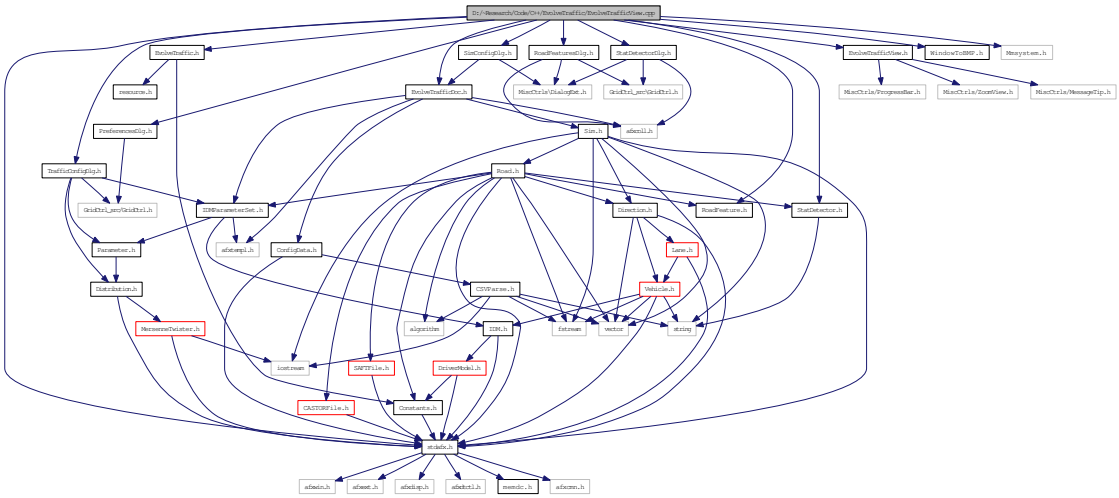
5.25.1.1 #define AFX_EVOLVETRAFFICDOC_H__E9AF4E58_CB5C_4E2F_-94AC_B1498A1C476A__INCLUDED_

Definition at line 6 of file EvolveTrafficDoc.h.

5.26 D:/~Research/Code/C++/EvolveTraffic/EvolveTrafficView.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "EvolveTrafficDoc.h"
#include "EvolveTrafficView.h"
#include "SimConfigDlg.h"
#include "TrafficConfigDlg.h"
#include "PreferencesDlg.h"
#include "RoadFeaturesDlg.h"
#include "RoadFeature.h"
#include "StatDetectorDlg.h"
#include "StatDetector.h"
#include "WindowToBMP.h"
#include <Mmsystem.h>
```

Include dependency graph for EvolveTrafficView.cpp:



Variables

- CConfigData g_ConfigData

5.26.1 Variable Documentation

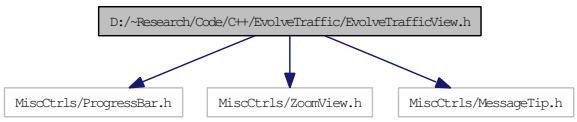
5.26.1.1 CConfigData g_ConfigData

Definition at line 32 of file ConfigData.cpp.

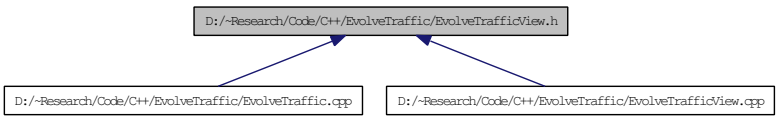
5.27 D:/~Research/Code/C++/EvolveTraffic/EvolveTrafficView.h File Reference

```
#include "MiscCtrls/ProgressBar.h"
#include "MiscCtrls/ZoomView.h"
#include "MiscCtrls/MessageTip.h"
```

Include dependency graph for EvolveTrafficView.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CEvolveTrafficView

  A class for user interaction and drawing functions.

Defines

- #define AFX_EVOLVETRAFFICVIEW_H__A8EBCCBD_29B8_4AA3_-BF81_C9D991006520__INCLUDED_

5.27.1 Define Documentation

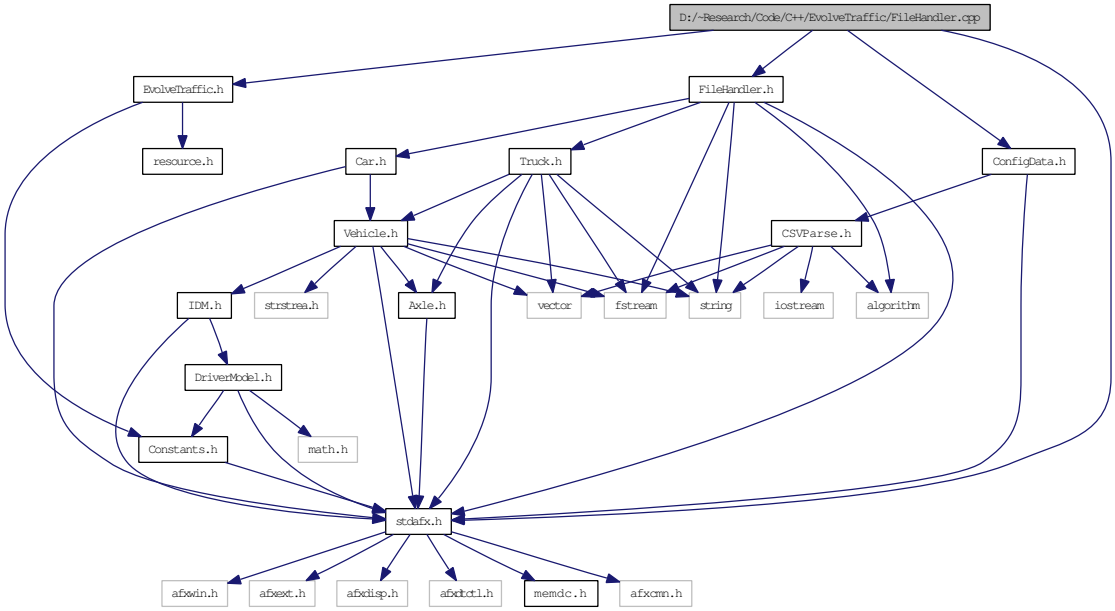
5.27.1.1 #define AFX_EVOLVETRAFFICVIEW_H__A8EBCCBD_29B8_-4AA3_BF81_C9D991006520__INCLUDED_

Definition at line 6 of file EvolveTrafficView.h.

5.28 D:/~Research/Code/C++/EvolveTraffic/FileHandler.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "FileHandler.h"
#include "ConfigData.h"
```

Include dependency graph for FileHandler.cpp:



Variables

- CConfigData g_ConfigData

5.28.1 Variable Documentation

5.28.1.1 CConfigData g_ConfigData

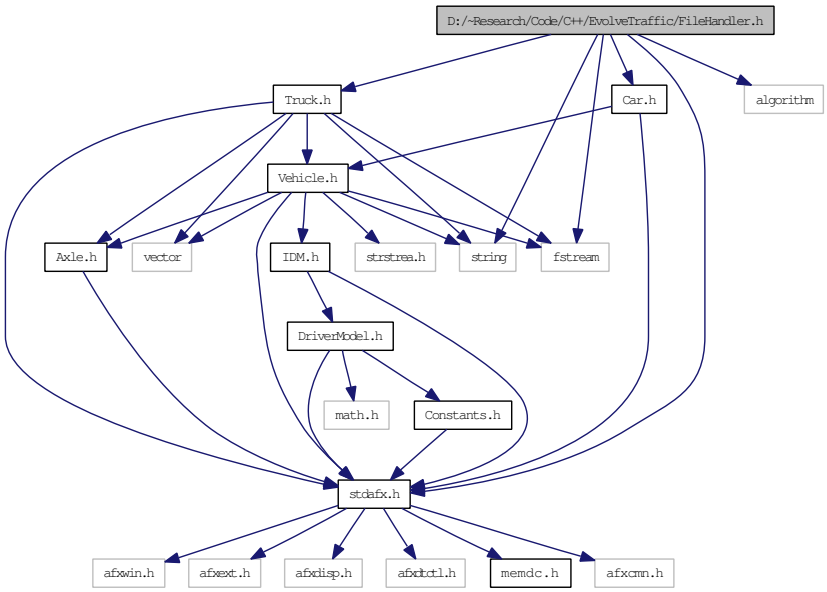
Definition at line 32 of file ConfigData.cpp.

5.29 D:/~Research/Code/C++/EvolveTraffic/FileHandler.h File Reference

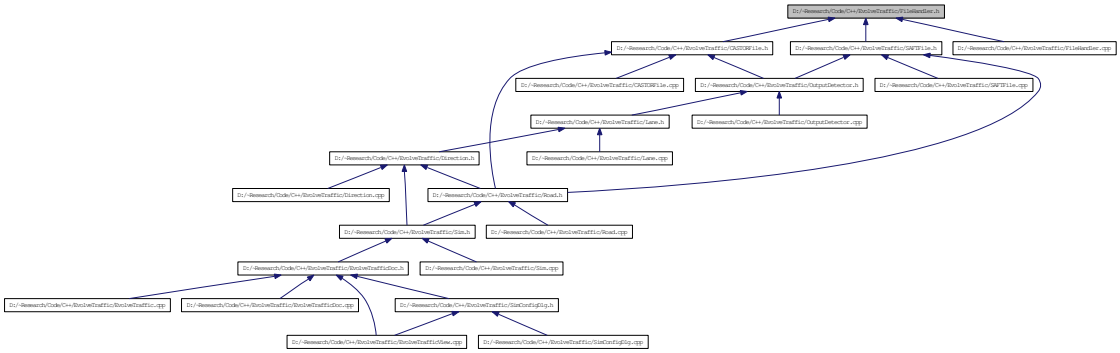
```
#include "stdafx.h"
#include <string>
```

```
#include <fstream>
#include <algorithm>
#include "Truck.h"
#include "Car.h"
```

Include dependency graph for FileHandler.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FileHandler

  A base class for handling file input and output.

Defines

- #define AFX_FILEHANDLER_H__1D4DB653_00A1_4904_803E_-A9AFD95DA75D__INCLUDED_

5.29.1 Define Documentation

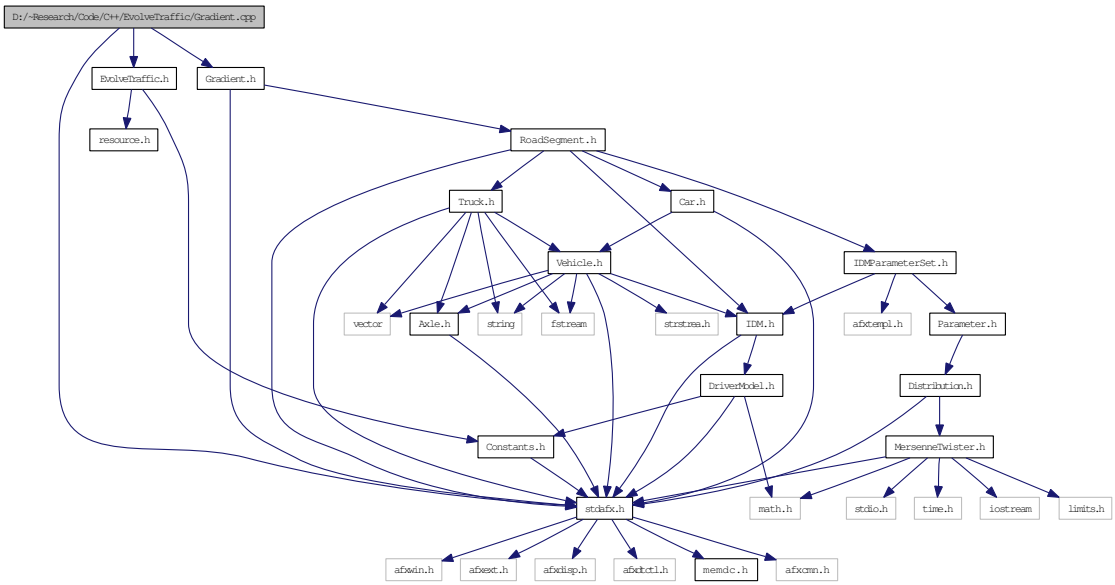
5.29.1.1 #define AFX_FILEHANDLER_H__1D4DB653_00A1_4904_803E_-A9AFD95DA75D__INCLUDED_

Definition at line 6 of file FileHandler.h.

5.30 D:/~Research/Code/C++/EvolveTraffic/Gradient.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "Gradient.h"
```

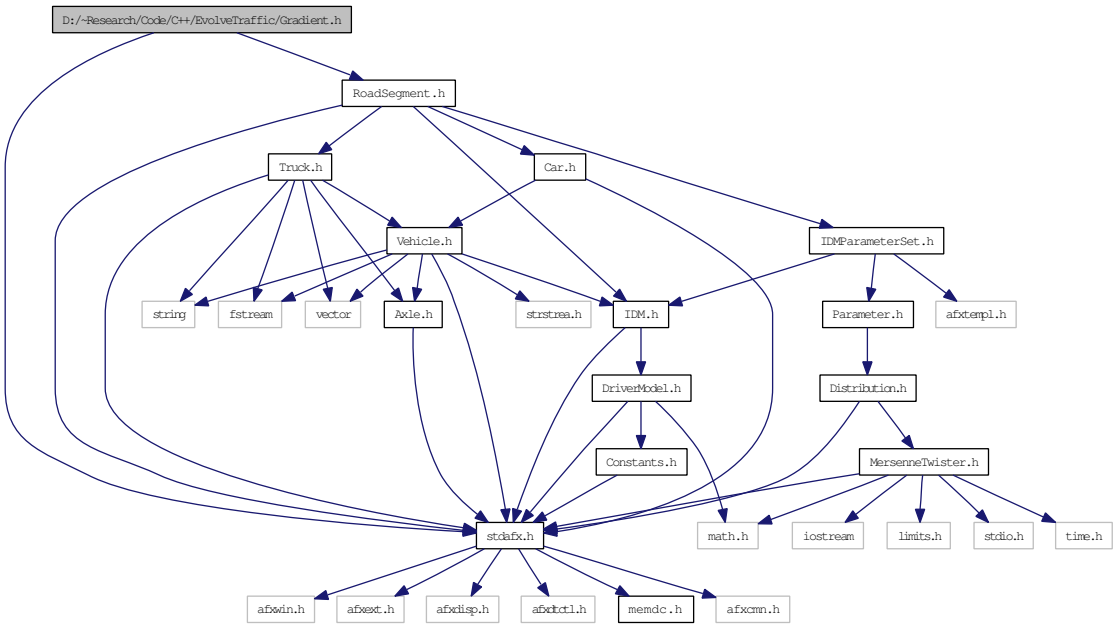
Include dependency graph for Gradient.cpp:



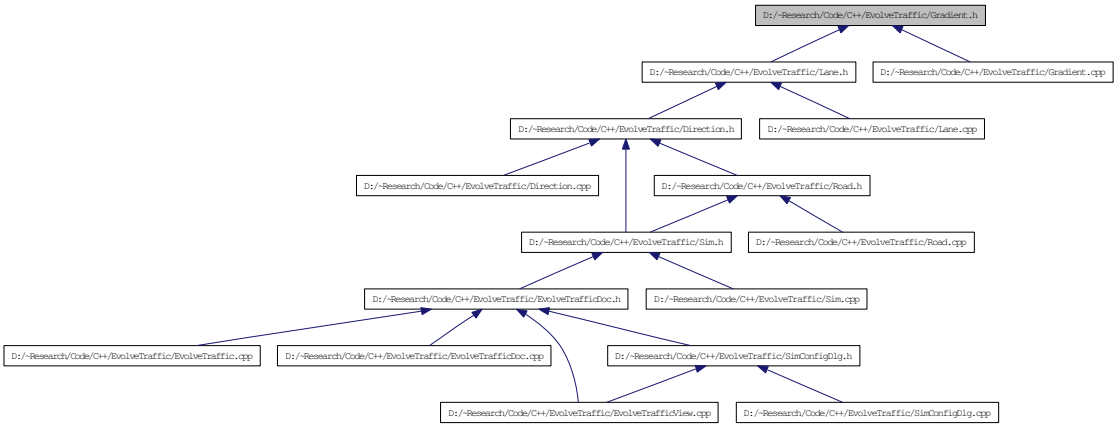
5.31 D:/~Research/Code/C++/EvolveTraffic/Gradient.h File Reference

```
#include "stdafx.h"
#include "RoadSegment.h"
```

Include dependency graph for Gradient.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Gradient

  A derived class to represent a gradient on a road.

Defines

- #define AFX_GRADIENT_H__AA2E912A_B7E3_4B9E_A253_-C350F42F6278__INCLUDED_

5.31.1 Define Documentation

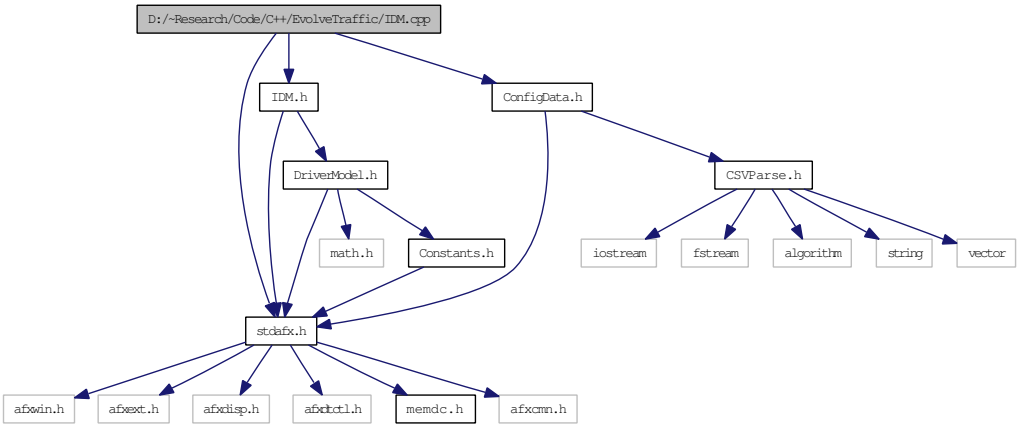
5.31.1.1 #define AFX_GRADIENT_H__AA2E912A_B7E3_4B9E_A253_-C350F42F6278__INCLUDED_

Definition at line 6 of file Gradient.h.

5.32 D:/~Research/Code/C++/EvolveTraffic/IDM.cpp File Reference

```
#include "stdafx.h"
#include "IDM.h"
#include "ConfigData.h"
```

Include dependency graph for IDM.cpp:



Variables

- CConfigData g_ConfigData

5.32.1 Variable Documentation

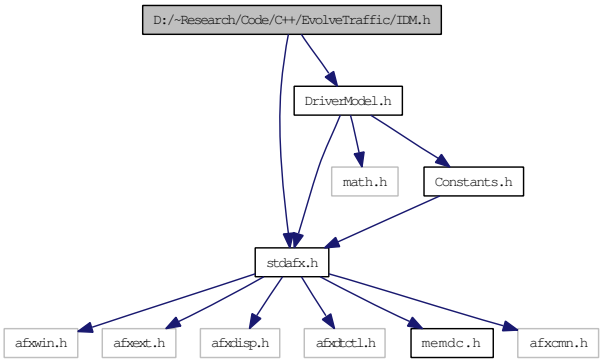
5.32.1.1 CConfigData g_ConfigData

Definition at line 32 of file ConfigData.cpp.

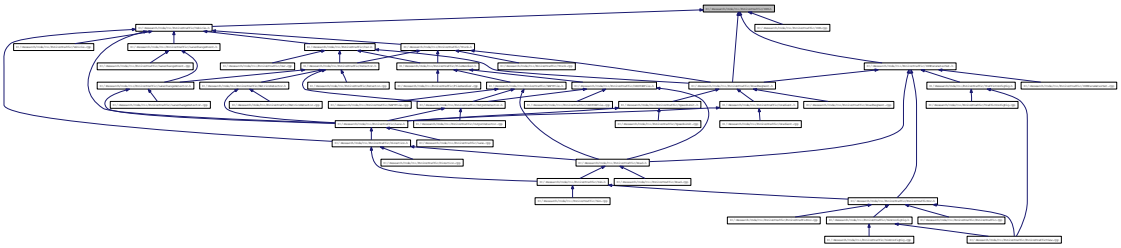
5.33 D:/~Research/Code/C++/EvolveTraffic/IDM.h File Reference

```
#include "stdafx.h"
#include "DriverModel.h"
```

Include dependency graph for IDM.h:



This graph shows which files directly or indirectly include this file:



Classes

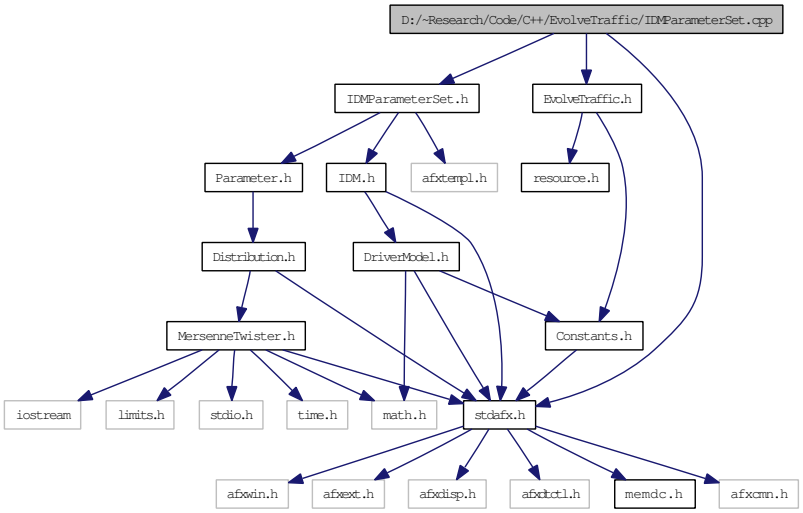
- class IDM

  A class representing the IDM driver model.

5.34 D:/~Research/Code/C++/EvolveTraffic/IDMParameterSet.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "IDMParameterSet.h"
```

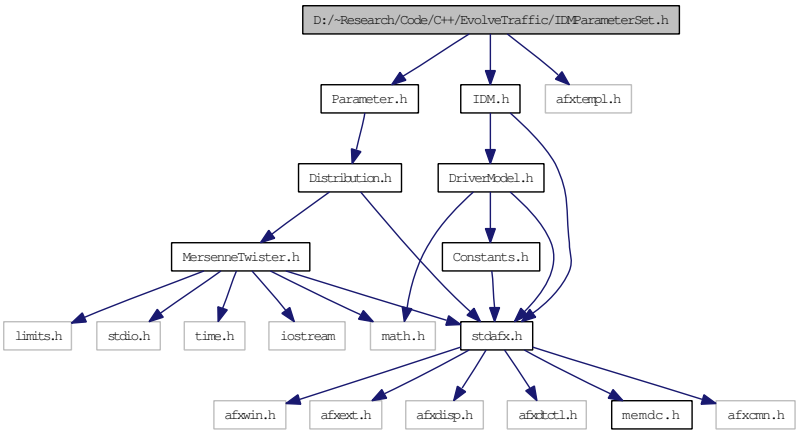
Include dependency graph for IDMParameterSet.cpp:



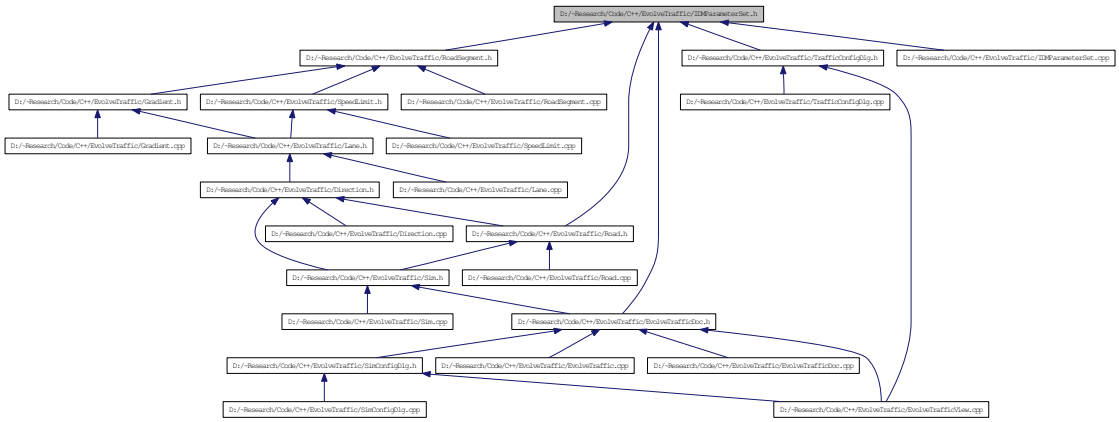
5.35 D:/~Research/Code/C++/EvolveTraffic/IDMParameterSet.h File Reference

```
#include "Parameter.h"
#include "IDM.h"
#include "afxtempl.h"
```

Include dependency graph for IDMParameterSet.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CIDMParameterSet

  A class representing a set of IDM parameters that can be saved to file.

Defines

- #define AFX_IDMPARAMETERSET_H__134C1BE2_9D40_4A62_B20D_-65C2D5901FE6__INCLUDED_

5.35.1 Define Documentation

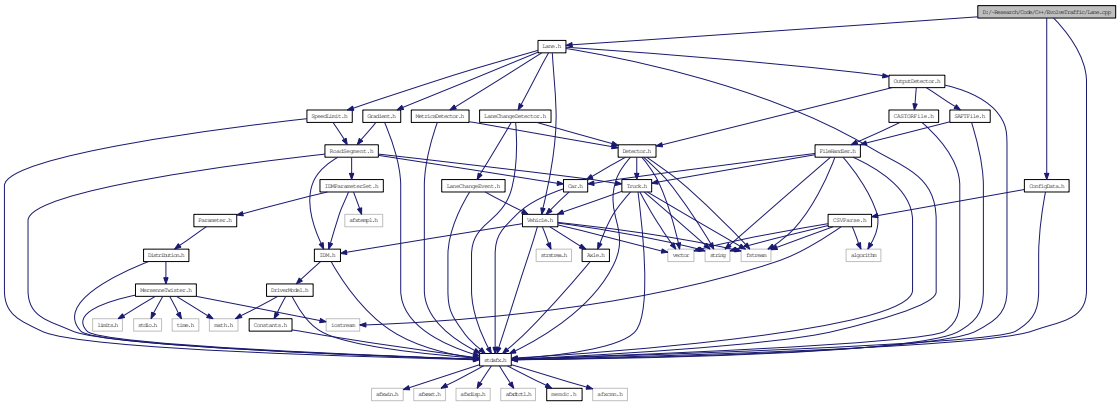
5.35.1.1 #define AFX_IDMPARAMETERSET_H__134C1BE2_9D40_4A62_-B20D_65C2D5901FE6__INCLUDED_

Definition at line 2 of file IDMParameterSet.h.

5.36 D:/~Research/Code/C++/EvolveTraffic/Lane.cpp File Reference

```
#include "stdafx.h"
#include "Lane.h"
#include "ConfigData.h"
```

Include dependency graph for Lane.cpp:



Variables

- CConfigData g_ConfigData

5.36.1 Variable Documentation

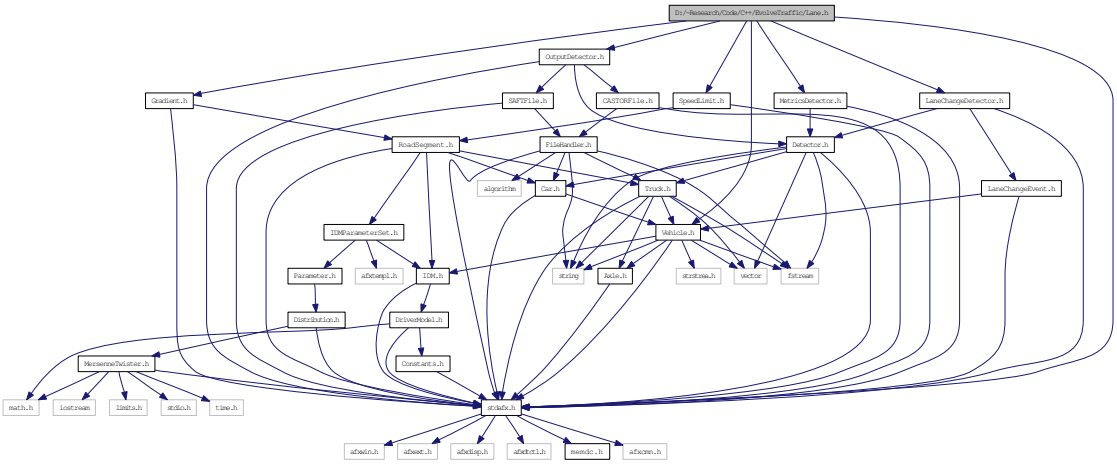
5.36.1.1 CConfigData g_ConfigData

Definition at line 32 of file ConfigData.cpp.

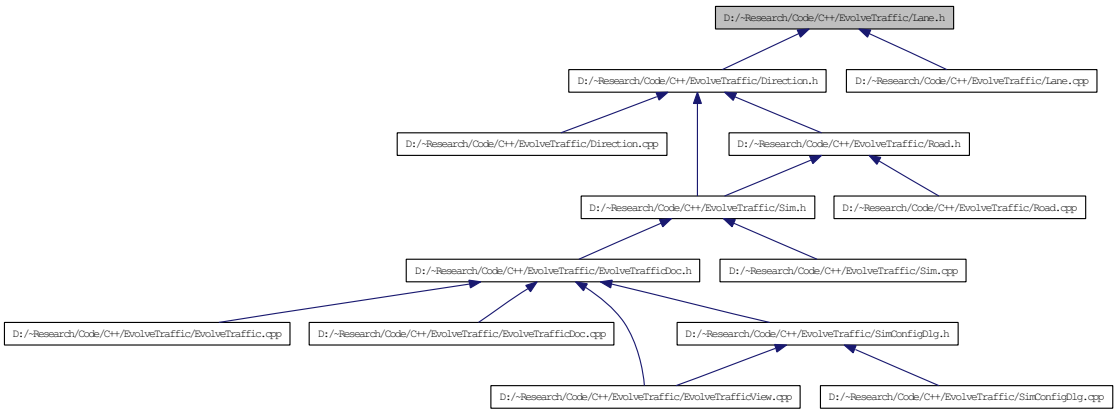
5.37 D:/~Research/Code/C++/EvolveTraffic/Lane.h File Reference

```
#include "stdafx.h"
#include "Vehicle.h"
#include "OutputDetector.h"
#include "SpeedLimit.h"
#include "MetricsDetector.h"
#include "Gradient.h"
#include "LaneChangeDetector.h"
```

Include dependency graph for Lane.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Lane

  A class representing a lane in a road.

Defines

- #define AFX_LANE_H__3506F588_71F6_4DF9_AF0A_10B8BA4B9F50__-INCLUDED_

5.37.1 Define Documentation

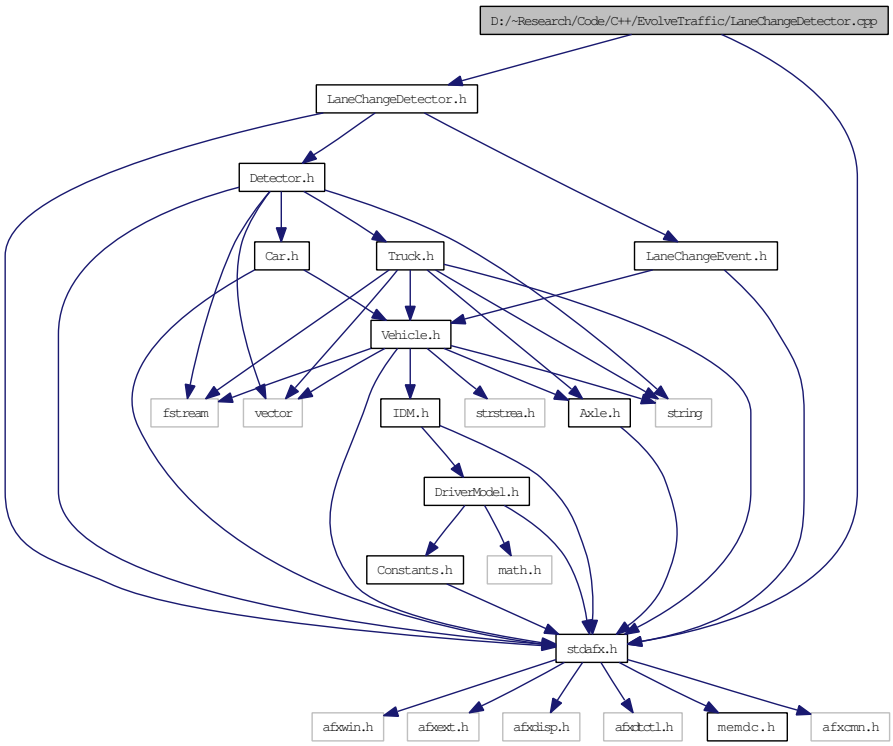
5.37.1.1 #define AFX_LANE_H__3506F588_71F6_4DF9_AF0A_10B8BA4B9F50__INCLUDED_

Definition at line 6 of file Lane.h.

5.38 D:/~Research/Code/C++/EvolveTraffic/LaneChangeDetector.cpp File Reference

```
#include "stdafx.h"
#include "LaneChangeDetector.h"
```

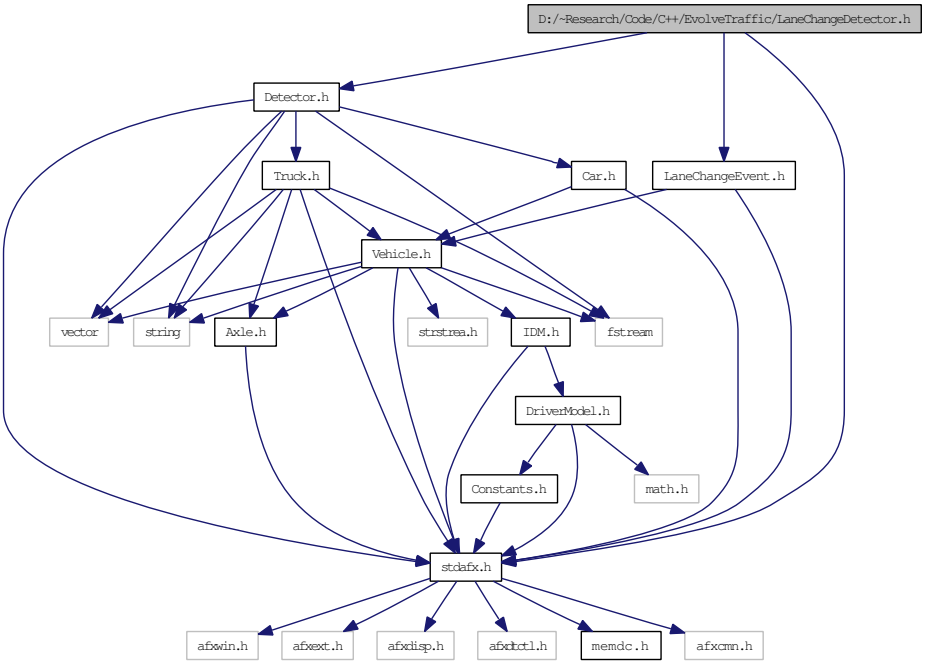
Include dependency graph for LaneChangeDetector.cpp:



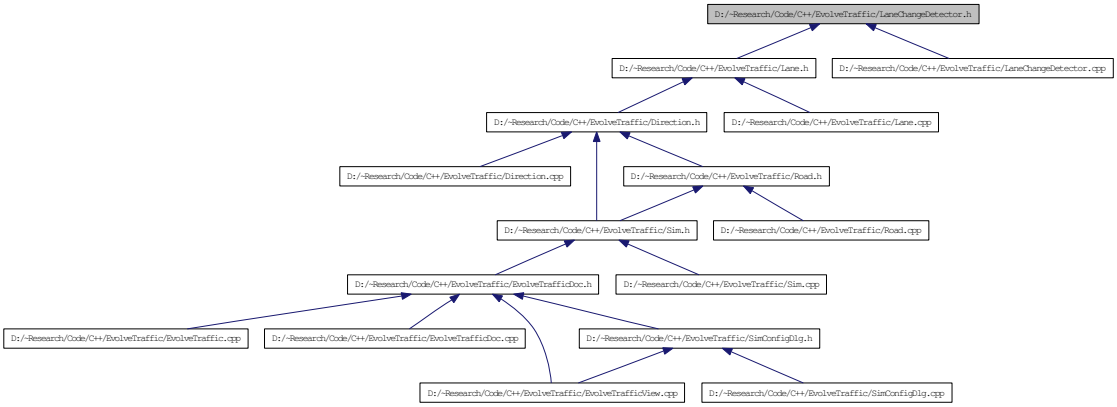
5.39 D:/~Research/Code/C++/EvolveTraffic/LaneChangeDetector.h File Reference

```
#include "stdafx.h"
#include "Detector.h"
#include "LaneChangeEvent.h"
```

Include dependency graph for LaneChangeDetector.h:



This graph shows which files directly or indirectly include this file:



Classes

- class LaneChangeDetector

  A derived class to represent a detector that tracks lane changes.

Defines

- #define AFX_LaneChangeDetector_H__639DB76E_AAB0_4635_9E5E_-2770E45286C3__INCLUDED_

5.39.1 Define Documentation

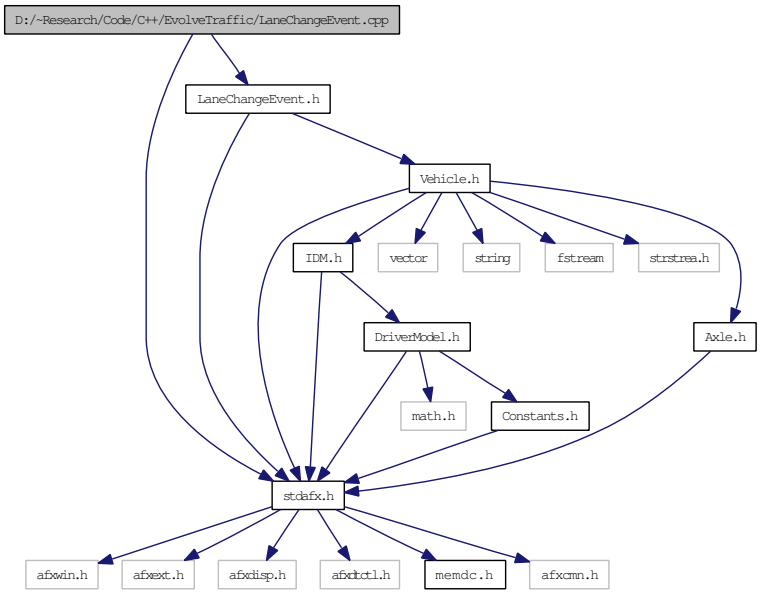
5.39.1.1 #define AFX_LaneChangeDetector_H__639DB76E_AAB0_4635_-9E5E_2770E45286C3__INCLUDED_

Definition at line 6 of file LaneChangeDetector.h.

5.40 D:/~Research/Code/C++/EvolveTraffic/LaneChangeEvent.cpp File Reference

```
#include "stdafx.h"
#include "LaneChangeEvent.h"
```

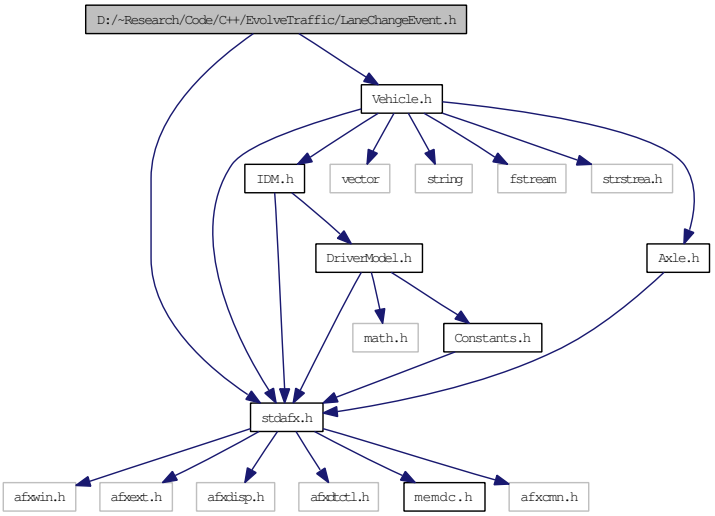
Include dependency graph for LaneChangeEvent.cpp:



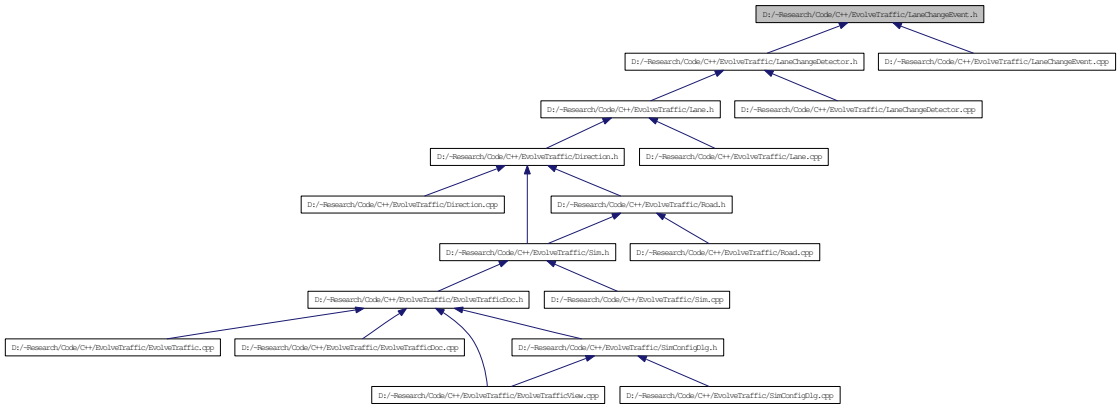
5.41 D:/~Research/Code/C++/EvolveTraffic/LaneChangeEvent.h File Reference

```
#include "stdafx.h"
#include "Vehicle.h"
```

Include dependency graph for LaneChangeEvent.h:



This graph shows which files directly or indirectly include this file:



Classes

- class LaneChangeEvent

  A container class to represent the relevant information from a lane change event.

Defines

- #define AFX_LANECHANGEEVENT_H__96B66F57_CE4E_4E26_A451_-345709048732__INCLUDED_

5.41.1 Define Documentation

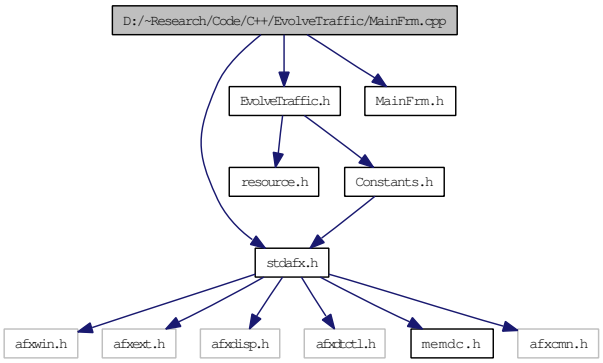
5.41.1.1 #define AFX_LANECHANGEEVENT_H__96B66F57_CE4E_4E26_-A451_345709048732__INCLUDED_

Definition at line 6 of file LaneChangeEvent.h.

5.42 D:/~Research/Code/C++/EvolveTraffic/MainFrm.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "MainFrm.h"
```

Include dependency graph for MainFrm.cpp:



Variables

- static UINT indicators [ ]

5.42.1 Variable Documentation

5.42.1.1 UINT indicators[ ] [static]

Initial value:

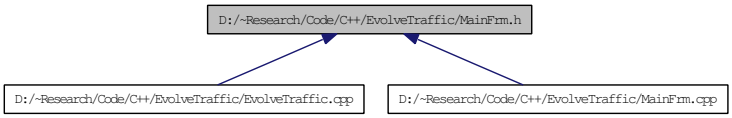
```
{
        ID_SEPARATOR,
        ID_INDICATOR_CAPS,
        ID_INDICATOR_NUM,
        ID_INDICATOR_SCRL,
}
```

Definition at line 26 of file MainFrm.cpp.

Referenced by CMainFrame::OnCreate().

5.43 D:/~Research/Code/C++/EvolveTraffic/MainFrm.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class CMainFrame

Defines

- #define AFX_MAINFRM_H__844A8630_1A25_49B0_8BD9_-F5BB57F1CCF6__INCLUDED_

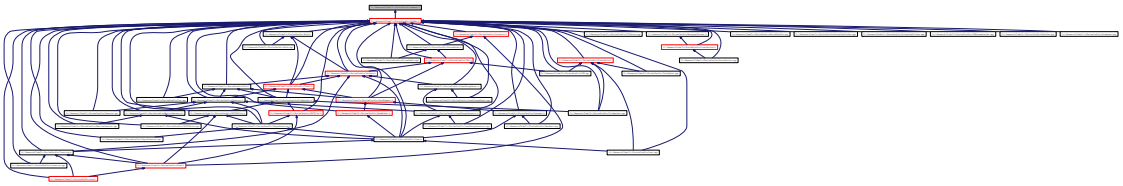
5.43.1 Define Documentation

5.43.1.1 #define AFX_MAINFRM_H__844A8630_1A25_49B0_8BD9_-F5BB57F1CCF6__INCLUDED_

Definition at line 6 of file MainFrm.h.

5.44 D:/~Research/Code/C++/EvolveTraffic/memdc.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

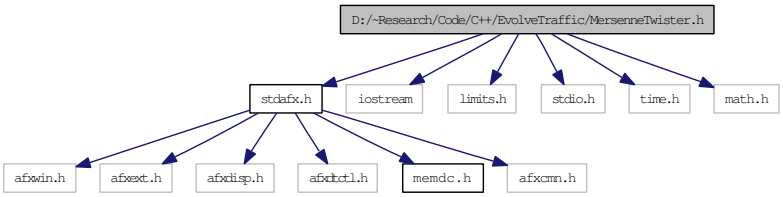
- class CMemDC

  A class for flicker-free drawing in the window.

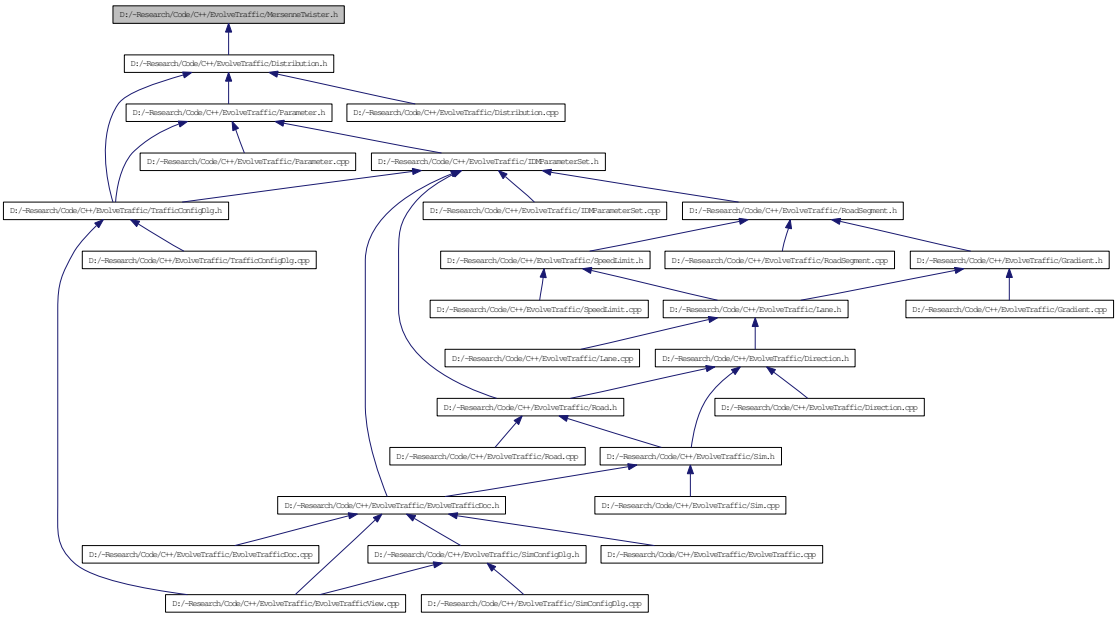
5.45 D:/~Research/Code/C++/EvolveTraffic/MersenneTwister.h File Reference

```
#include "stdafx.h"
#include <iostream>
#include <limits.h>
#include <stdio.h>
#include <time.h>
#include <math.h>
```

Include dependency graph for MersenneTwister.h:



This graph shows which files directly or indirectly include this file:



Classes

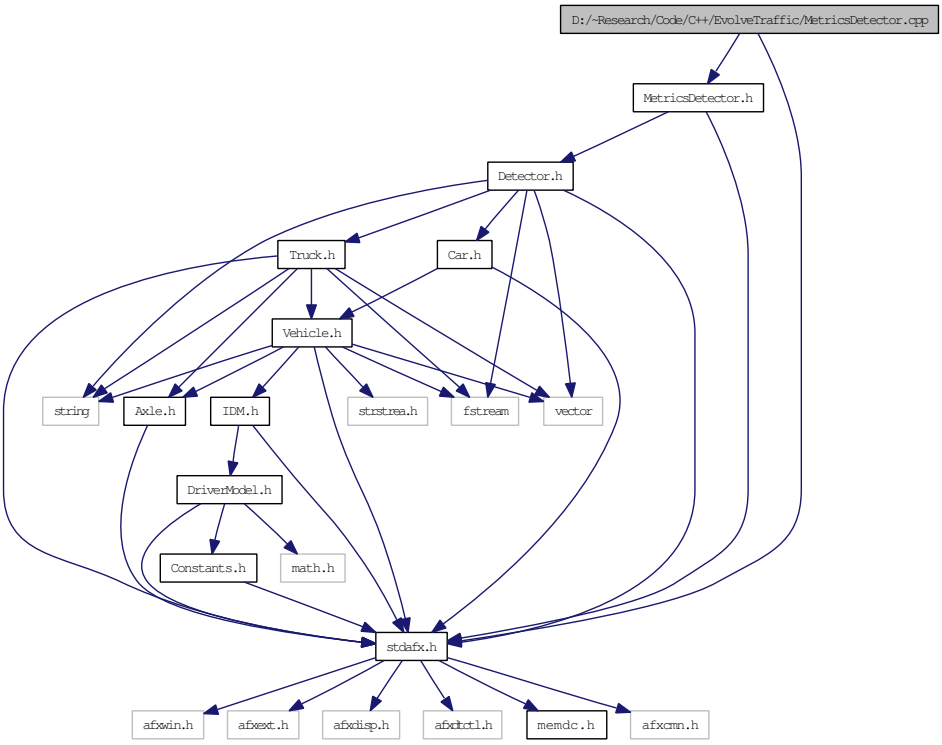
- class MTRand

A class for generating random numbers - Mersenne Twister.

5.46 D:/~Research/Code/C++/EvolveTraffic/MetricsDetector.cpp File Reference

```
#include "stdafx.h"
#include "MetricsDetector.h"
```

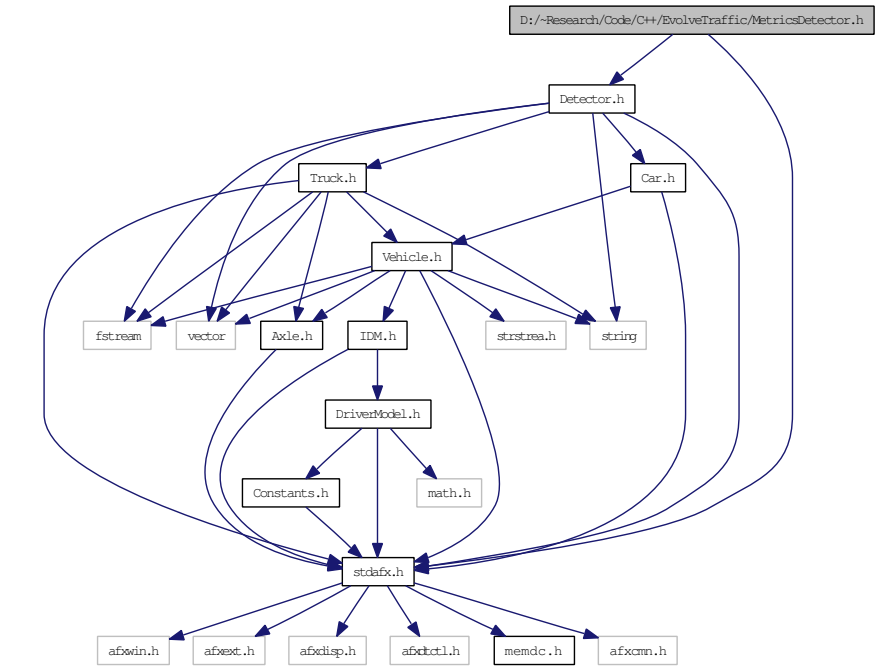
Include dependency graph for MetricsDetector.cpp:



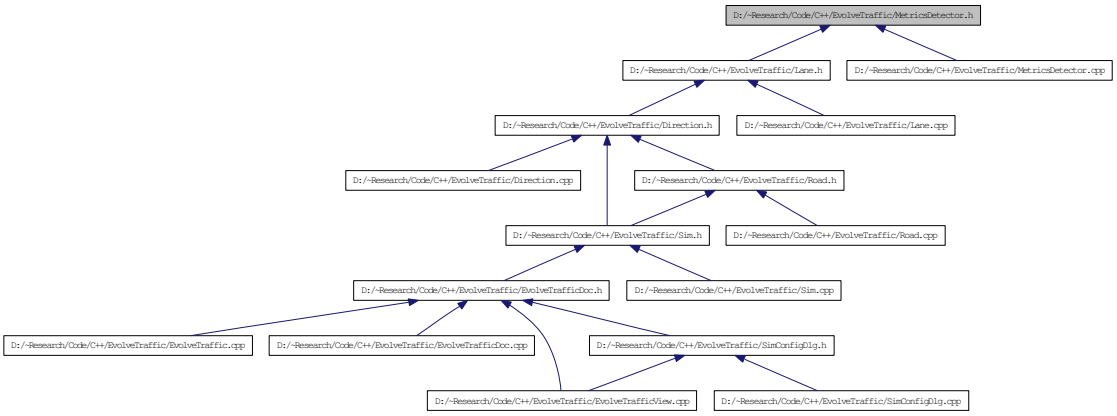
5.47 D:/~Research/Code/C++/EvolveTraffic/MetricsDetector.h File Reference

```
#include "stdafx.h"
#include "Detector.h"
```

Include dependency graph for MetricsDetector.h:



This graph shows which files directly or indirectly include this file:



Classes

- class MetricsDetector

  A derived class to represent a detector that tracks metrics information.

Defines

- #define AFX_METRICSDETECTOR_H__06A2EEA5_6E1B_4581_BB61_-6FEC99FBABCC__INCLUDED_

Typedefs

- typedef std::vector< std::vector< double > > M2Ddbl

5.47.1 Define Documentation

5.47.1.1 #define AFX_METRICSDETECTOR_H__06A2EEA5_6E1B_4581_-BB61_6FEC99FBABCC__INCLUDED_

Definition at line 6 of file MetricsDetector.h.

5.47.2 Typedef Documentation

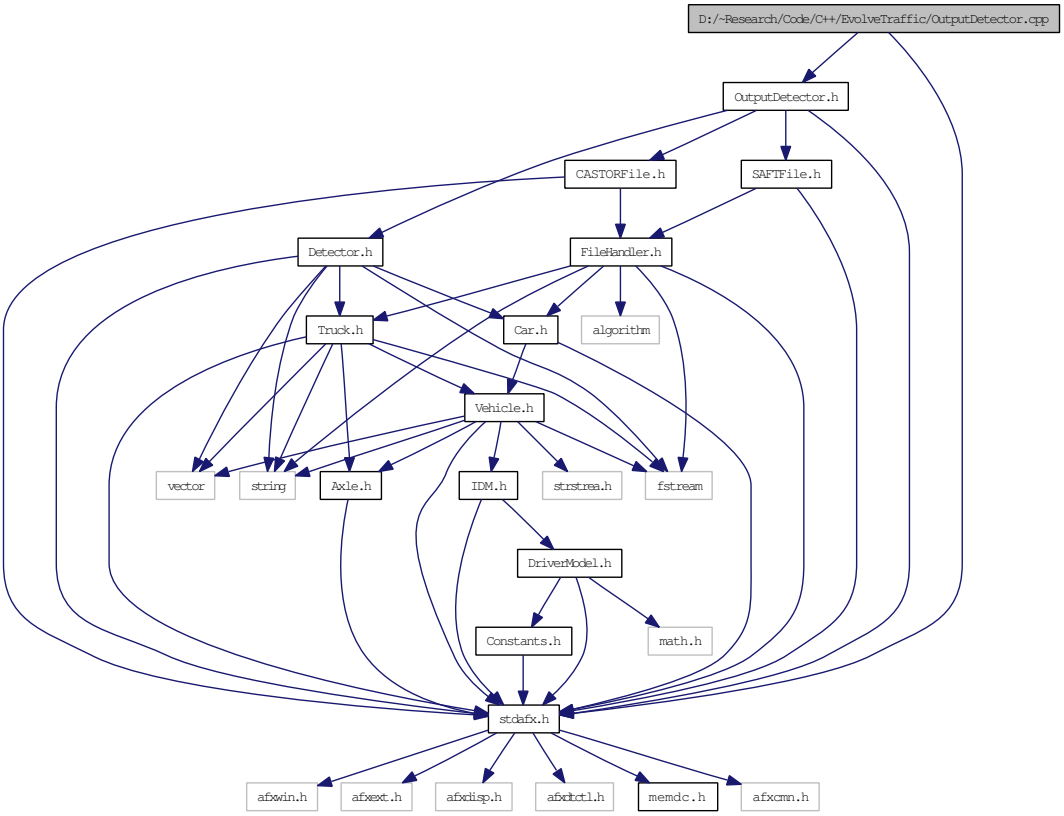
5.47.2.1 typedef std::vector< std::vector<double> > M2Ddbl

Definition at line 15 of file MetricsDetector.h.

5.48 D:/~Research/Code/C++/EvolveTraffic/OutputDetector.cpp File Reference

```
#include "stdafx.h"
#include "OutputDetector.h"
```

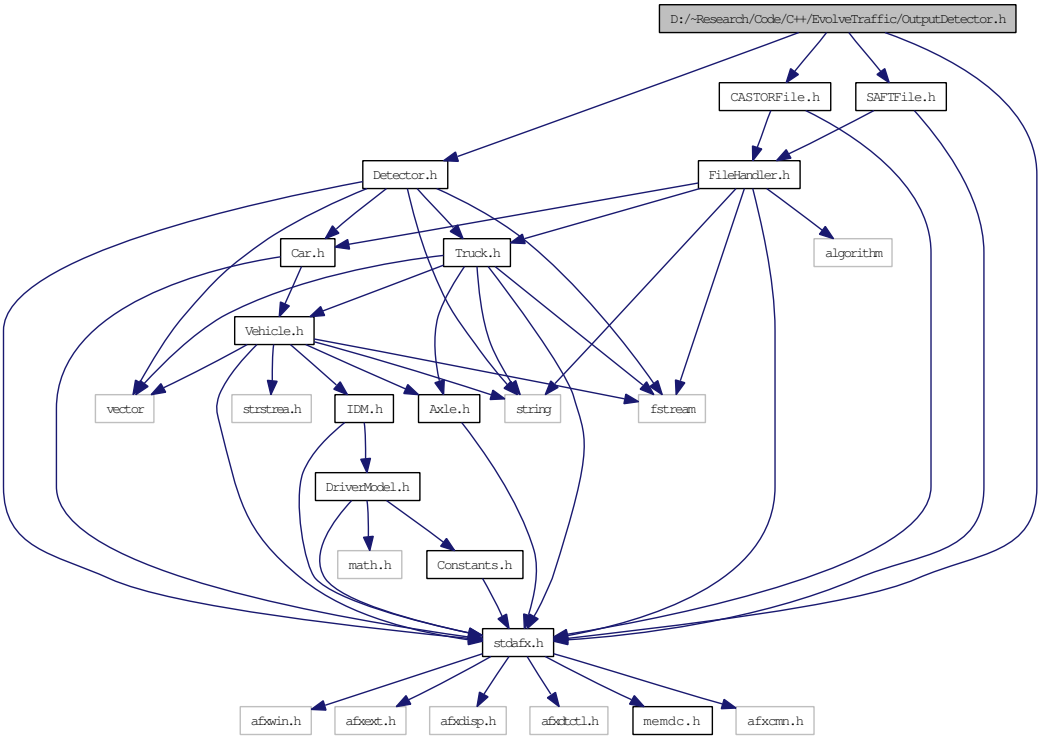
Include dependency graph for OutputDetector.cpp:



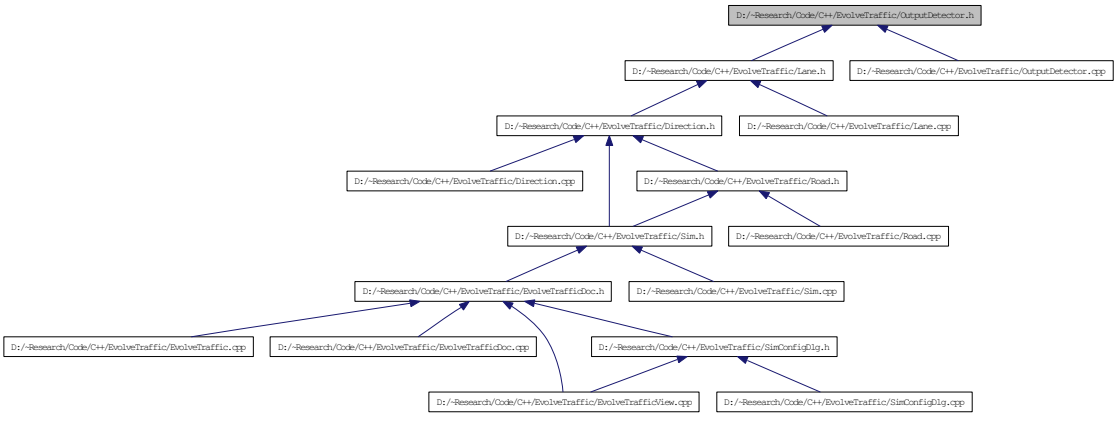
5.49 D:/~Research/Code/C++/EvolveTraffic/OutputDetector.h File Reference

```
#include "stdafx.h"
#include "Detector.h"
#include "CASTORFile.h"
#include "SAFTFile.h"
```

Include dependency graph for OutputDetector.h:



This graph shows which files directly or indirectly include this file:



Classes

- class OutputDetector

  A derived class representing a detector which tracks when vehicles pass a point.

Defines

- #define AFX_DETECTOR_H__F90280E9_BFD2_40CF_A999_-86FF1E9DDE15__INCLUDED_

5.49.1 Define Documentation

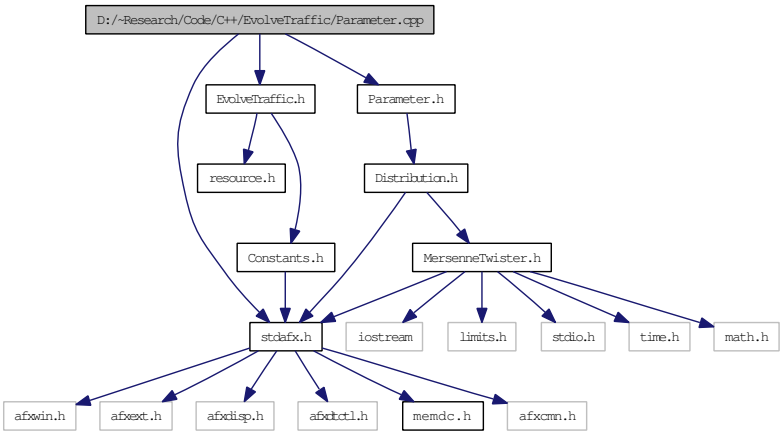
5.49.1.1 #define AFX_DETECTOR_H__F90280E9_BFD2_40CF_A999_-86FF1E9DDE15__INCLUDED_

Definition at line 6 of file OutputDetector.h.

5.50 D:/~Research/Code/C++/EvolveTraffic/Parameter.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "Parameter.h"
```

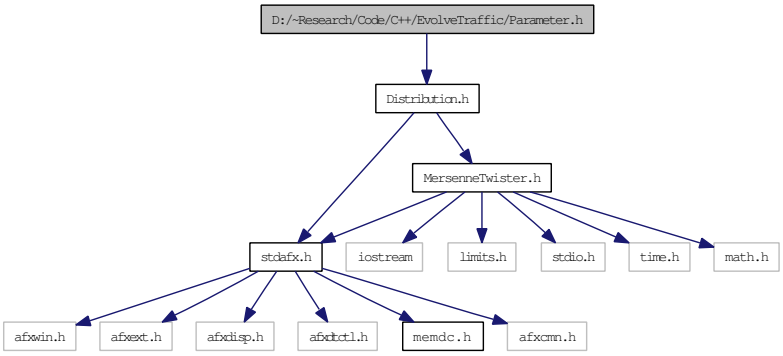
Include dependency graph for Parameter.cpp:



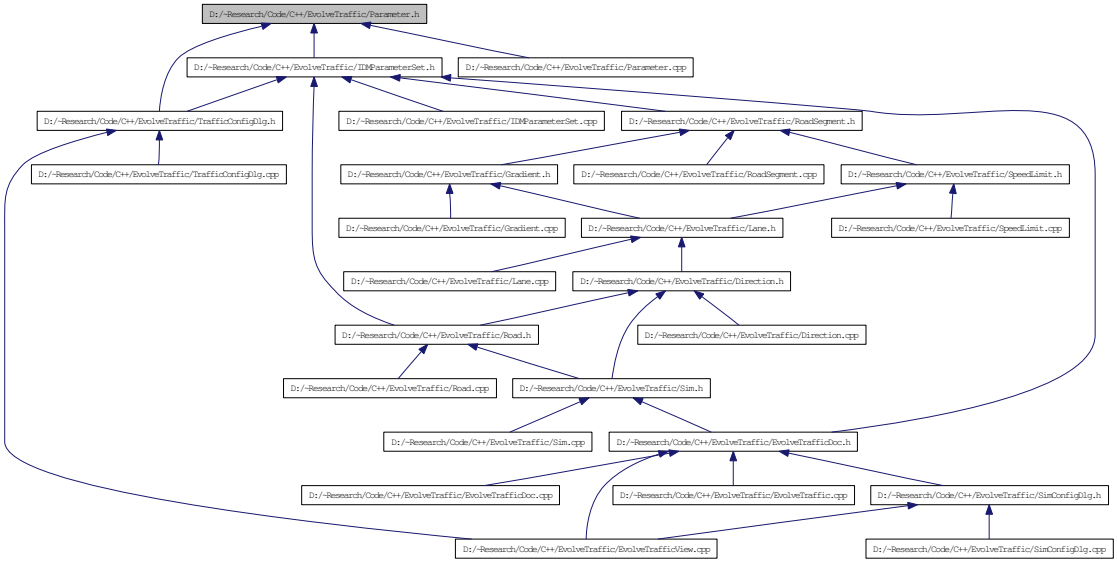
5.51 D:/~Research/Code/C++/EvolveTraffic/Parameter.h File Reference

```
#include "Distribution.h"
```

Include dependency graph for Parameter.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CParameter

  A class represetning a single IDM parameter that can be saved to file.

Defines

- #define AFX_PARAMETER_H__1F448E46_482F_4100_97E4_-9A28DCF398AE__INCLUDED_

5.51.1 Define Documentation

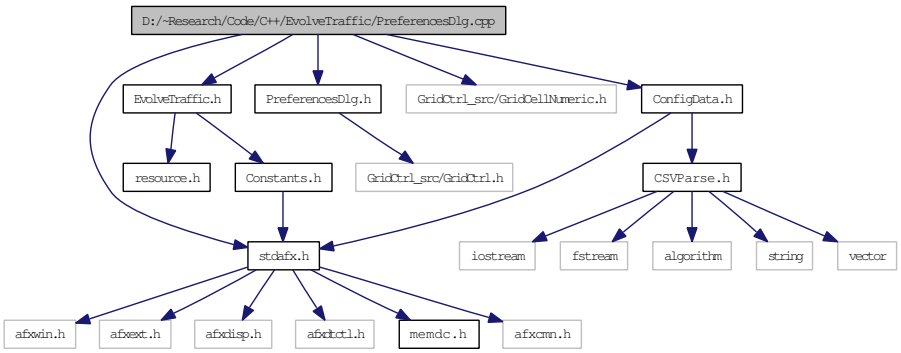
5.51.1.1 #define AFX_PARAMETER_H__1F448E46_482F_4100_97E4_-9A28DCF398AE__INCLUDED_

Definition at line 2 of file Parameter.h.

5.52 D:/~Research/Code/C++/EvolveTraffic/PreferencesDlg.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "PreferencesDlg.h"
#include "GridCtrl_src/GridCellNumeric.h"
#include "ConfigData.h"
```

Include dependency graph for PreferencesDlg.cpp:



Variables

- CConfigData g_ConfigData

5.52.1 Variable Documentation

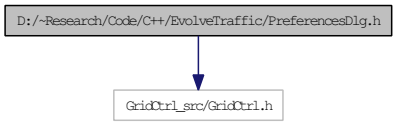
5.52.1.1 CConfigData g_ConfigData

Definition at line 32 of file ConfigData.cpp.

5.53 D:/~Research/Code/C++/EvolveTraffic/PreferencesDlg.h File Reference

```
#include "GridCtrl_src/GridCtrl.h"
```

Include dependency graph for PreferencesDlg.h:



This graph shows which files directly or indirectly include this file:

Classes

- class CPreferencesDlg

  A class for the User Preferences Dialog.

Defines

- #define AFX_PREFERENCESDLG_H__E42A9827_1C70_4C4D_A234_-4645C7FC5464__INCLUDED_

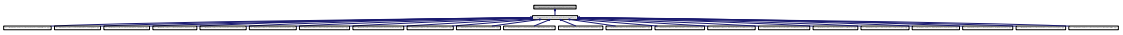
5.53.1 Define Documentation

5.53.1.1 #define AFX_PREFERENCESDLG_H__E42A9827_1C70_4C4D_-A234_4645C7FC5464__INCLUDED_

Definition at line 2 of file PreferencesDlg.h.

5.54 D:/~Research/Code/C++/EvolveTraffic/resource.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define IDD_ABOUTBOX 100
- #define IDR_MAINFRAME 128

- #define IDR_EVOLVETYPE 129
- #define IDD_SIMCONFIG 130
- #define IDD_TRAFFICCONFIG 132
- #define IDD_PREFERENCES 134
- #define IDD_ROADFEATURES 135
- #define IDD_STATDETECTORS 136
- #define IDC_EDT_FILE 1000
- #define IDC_EDT_FILE_IN 1000
- #define IDC_BTN_FILEPICK 1001
- #define IDC_BTN_FILEINPICK 1001
- #define IDC_EDT_ROADLENGTH 1002
- #define IDC_RAD_SAFT 1003
- #define IDC_RAD_CASTOR 1004
- #define IDC_CMB_LANESDIR1 1005
- #define IDC_CMB_LANESDIRPOS 1005
- #define IDC_CMB_LANESDIR2 1006
- #define IDC_CMB_LANESDIRNEG 1006
- #define IDC_EDT_FILE_OUT 1007
- #define IDC_CMB_DRIVESIDE 1008
- #define IDC_EDT_SIMTIMESTEP 1009
- #define IDC_GRID 1010
- #define IDC_CMB_TRAFFILE_LANESDIR1 1010
- #define IDC_CMB_CLASSDEFINE 1011
- #define IDC_CMB_TRAFFILE_LANESDIR2 1011
- #define IDC_BTN_COPY 1012
- #define IDC_BTN_FILEOUTPICK 1012
- #define IDC_CMB_CLASSCOPY 1013
- #define IDC_EDT_DIRPOSDETECTOR 1013
- #define IDC_EDT_DIRNEGDETECTOR 1014
- #define IDC_BTN_METRICSOUT 1015
- #define IDC_EDT_METRICSOUT 1016
- #define IDC_CHECK_LANECHNG 1024
- #define IDC_BTN_ADDFEAT 1025
- #define IDC_BTN_DELFEAT 1026
- #define IDC_BTN_ADDDET 1027
- #define IDC_BTN_DELDET 1028
- #define IDC_EDIT1 1029
- #define ID_CONFIG_TRAF 32771
- #define ID_CONFIG_SIM 32772
- #define ID_RUN_RUNALL 32773
- #define ID_RUN_READINTRUCKFILE 32774
- #define ID_RUN_EVOLVETRAFFIC 32775
- #define ID_TOOLS_PREFS 32776
- #define ID_TOOLS_PAUSE 32777
- #define ID_TOOLS_ZOOMIN 32778
- #define ID_TOOLS_ZOOMOUT 32779

- #define ID_TOOLS_SPEEDUP 32780
- #define ID_TOOLS_SLOWDOWN 32781
- #define ID_RUN_VISIBLE 32786
- #define ID_RUN_INVISIBLE 32787
- #define ID_CONFIG_FEATURES 32788
- #define ID_CONFIG_METRICS 32789
- #define ID_TOOLS_STOP 32794
- #define ID_BUTTON32801 32801
- #define ID_BUTTON32805 32805
- #define ID_BUTTON32806 32806
- #define ID_FILE_SAVEIMAGETOFILE 32808
- #define ID_FILE_SAVEIMAGE 32809

5.54.1 Define Documentation

5.54.1.1 #define ID_BUTTON32801 32801

Definition at line 60 of file resource.h.

5.54.1.2 #define ID_BUTTON32805 32805

Definition at line 61 of file resource.h.

5.54.1.3 #define ID_BUTTON32806 32806

Definition at line 62 of file resource.h.

5.54.1.4 #define ID_CONFIG_FEATURES 32788

Definition at line 57 of file resource.h.

5.54.1.5 #define ID_CONFIG_METRICS 32789

Definition at line 58 of file resource.h.

5.54.1.6 #define ID_CONFIG_SIM 32772

Definition at line 45 of file resource.h.

5.54.1.7 #define ID_CONFIG_TRAF 32771

Definition at line 44 of file resource.h.

5.54.1.8 #define ID_FILE_SAVEIMAGE 32809

Definition at line 64 of file resource.h.

5.54.1.9 #define ID_FILE_SAVEIMAGETOFILE 32808

Definition at line 63 of file resource.h.

5.54.1.10 #define ID_RUN_EVOLVETRAFFIC 32775

Definition at line 48 of file resource.h.

5.54.1.11 #define ID_RUN_INVISIBLE 32787

Definition at line 56 of file resource.h.

5.54.1.12 #define ID_RUN_READINTRUCKFILE 32774

Definition at line 47 of file resource.h.

5.54.1.13 #define ID_RUN_RUNALL 32773

Definition at line 46 of file resource.h.

5.54.1.14 #define ID_RUN_VISIBLE 32786

Definition at line 55 of file resource.h.

5.54.1.15 #define ID_TOOLS_PAUSE 32777

Definition at line 50 of file resource.h.

5.54.1.16 #define ID_TOOLS_PREFS 32776

Definition at line 49 of file resource.h.

5.54.1.17 #define ID_TOOLS_SLOWDOWN 32781

Definition at line 54 of file resource.h.

5.54.1.18 #define ID_TOOLS_SPEEDUP 32780

Definition at line 53 of file resource.h.

5.54.1.19 #define ID_TOOLS_STOP 32794

Definition at line 59 of file resource.h.

5.54.1.20 #define ID_TOOLS_ZOOMIN 32778

Definition at line 51 of file resource.h.

5.54.1.21 #define ID_TOOLS_ZOOMOUT 32779

Definition at line 52 of file resource.h.

5.54.1.22 #define IDC_BTN_ADDDET 1027

Definition at line 41 of file resource.h.

5.54.1.23 #define IDC_BTN_ADDFEAT 1025

Definition at line 39 of file resource.h.

5.54.1.24 #define IDC_BTN_COPY 1012

Definition at line 31 of file resource.h.

5.54.1.25 #define IDC_BTN_DELDET 1028

Definition at line 42 of file resource.h.

5.54.1.26 #define IDC_BTN_DELFEAT 1026

Definition at line 40 of file resource.h.

5.54.1.27 #define IDC_BTN_FILEINPICK 1001

Definition at line 16 of file resource.h.

5.54.1.28 #define IDC_BTN_FILEOUTPICK 1012

Definition at line 32 of file resource.h.

5.54.1.29 #define IDC_BTN_FILEPICK 1001

Definition at line 15 of file resource.h.

5.54.1.30 #define IDC_BTN_METRICSOUT 1015

Definition at line 36 of file resource.h.

5.54.1.31 #define IDC_CHECK_LANECHNG 1024

Definition at line 38 of file resource.h.

Referenced by CSimConfigDlg::DoDataExchange().

5.54.1.32 #define IDC_CMB_CLASSCOPY 1013

Definition at line 33 of file resource.h.

Referenced by CTrafficConfigDlg::DoDataExchange().

5.54.1.33 #define IDC_CMB_CLASSDEFINE 1011

Definition at line 29 of file resource.h.

Referenced by CTrafficConfigDlg::DoDataExchange().

5.54.1.34 #define IDC_CMB_DRIVESIDE 1008

Definition at line 25 of file resource.h.

Referenced by CSimConfigDlg::DoDataExchange(), CSimConfigDlg::OnInitDialog(), and CSimConfigDlg::OnSelchangeCmbDriveside().

5.54.1.35 #define IDC_CMB_LANESDIR1 1005

Definition at line 20 of file resource.h.

5.54.1.36 #define IDC_CMB_LANESDIR2 1006

Definition at line 22 of file resource.h.

5.54.1.37 #define IDC_CMB_LANESDIRNEG 1006

Definition at line 23 of file resource.h.

Referenced by CSimConfigDlg::DoDataExchange(), and CSimConfigDlg::OnValidate().

5.54.1.38 #define IDC_CMB_LANESDIRPOS 1005

Definition at line 21 of file resource.h.

Referenced by CSimConfigDlg::DoDataExchange(), and CSimConfigDlg::OnValidate().

5.54.1.39 #define IDC_CMB_TRAFFILE_LANESDIR1 1010

Definition at line 28 of file resource.h.

Referenced by CSimConfigDlg::DoDataExchange(), and CSimConfigDlg::OnValidate().

5.54.1.40 #define IDC_CMB_TRAFFILE_LANESDIR2 1011

Definition at line 30 of file resource.h.

Referenced by CSimConfigDlg::DoDataExchange(), and CSimConfigDlg::OnValidate().

5.54.1.41 #define IDC_EDIT1 1029

Definition at line 43 of file resource.h.

Referenced by CAboutDlg::DoDataExchange().

5.54.1.42 #define IDC_EDT_DIRNEGDETECTOR 1014

Definition at line 35 of file resource.h.

Referenced by CSimConfigDlg::DoDataExchange(), and CSimConfigDlg::OnValidate().

5.54.1.43 #define IDC_EDT_DIRPOSDETECTOR 1013

Definition at line 34 of file resource.h.

Referenced by CSimConfigDlg::DoDataExchange(), and CSimConfigDlg::OnValidate().

5.54.1.44 #define IDC_EDT_FILE 1000

Definition at line 13 of file resource.h.

5.54.1.45 #define IDC_EDT_FILE_IN 1000

Definition at line 14 of file resource.h.

Referenced by CSimConfigDlg::DoDataExchange(), and CSimConfigDlg::OnValidate().

5.54.1.46 #define IDC_EDT_FILE_OUT 1007

Definition at line 24 of file resource.h.

Referenced by CSimConfigDlg::DoDataExchange(), and CSimConfigDlg::OnValidate().

5.54.1.47 #define IDC_EDT_METRICSOUT 1016

Definition at line 37 of file resource.h.

Referenced by CSimConfigDlg::DoDataExchange().

5.54.1.48 #define IDC_EDT_ROADLENGTH 1002

Definition at line 17 of file resource.h.

Referenced by CSimConfigDlg::DoDataExchange().

5.54.1.49 #define IDC_EDT_SIMTIMESTEP 1009

Definition at line 26 of file resource.h.

Referenced by CSimConfigDlg::DoDataExchange().

5.54.1.50 #define IDC_GRID 1010

Definition at line 27 of file resource.h.

Referenced by CTrafficConfigDlg::DoDataExchange(), CStatDetectorDlg::DoDataExchange(), CRoadFeaturesDlg::DoDataExchange(), CPreferencesDlg::DoDataExchange(), CStatDetectorDlg::OnValidate(), and CRoadFeaturesDlg::OnValidate().

5.54.1.51 #define IDC_RAD_CASTOR 1004

Definition at line 19 of file resource.h.

Referenced by CSimConfigDlg::OnInitDialog().

5.54.1.52 #define IDC_RAD_SAFT 1003

Definition at line 18 of file resource.h.

Referenced by CSimConfigDlg::OnInitDialog().

5.54.1.53 #define IDD_ABOUTBOX 100

Definition at line 5 of file resource.h.

5.54.1.54 #define IDD_PREFERENCES 134

Definition at line 10 of file resource.h.

5.54.1.55 #define IDD_ROADFEATURES 135

Definition at line 11 of file resource.h.

5.54.1.56 #define IDD_SIMCONFIG 130

Definition at line 8 of file resource.h.

5.54.1.57 #define IDD_STATDETECTORS 136

Definition at line 12 of file resource.h.

5.54.1.58 #define IDD_TRAFFICCONFIG 132

Definition at line 9 of file resource.h.

5.54.1.59 #define IDR_EVOLVETYPE 129

Definition at line 7 of file resource.h.

5.54.1.60 #define IDR_MAINFRAME 128

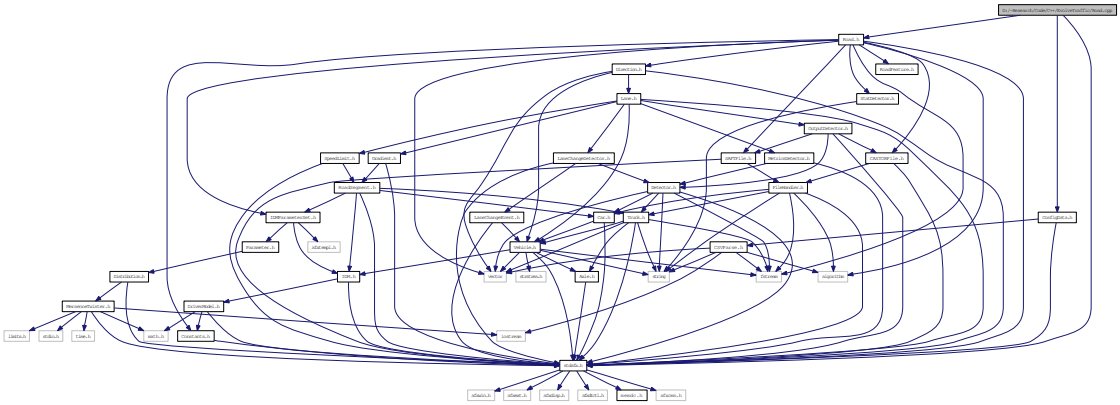
Definition at line 6 of file resource.h.

Referenced by CEvolveTrafficApp::InitInstance(), and CMainFrame::OnCreate().

5.55 D:/~Research/Code/C++/EvolveTraffic/Road.cpp File Reference

```
#include "stdafx.h"
#include "Road.h"
#include "ConfigData.h"
```

Include dependency graph for Road.cpp:



Variables

- CConfigData g_ConfigData

5.55.1 Variable Documentation

5.55.1.1 CConfigData g_ConfigData

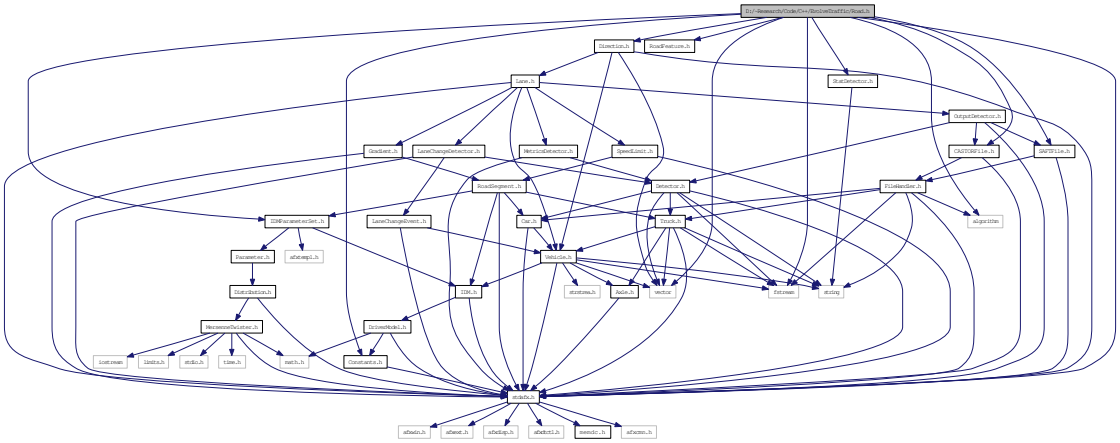
Definition at line 32 of file ConfigData.cpp.

5.56 D:/~Research/Code/C++/EvolveTraffic/Road.h File Reference

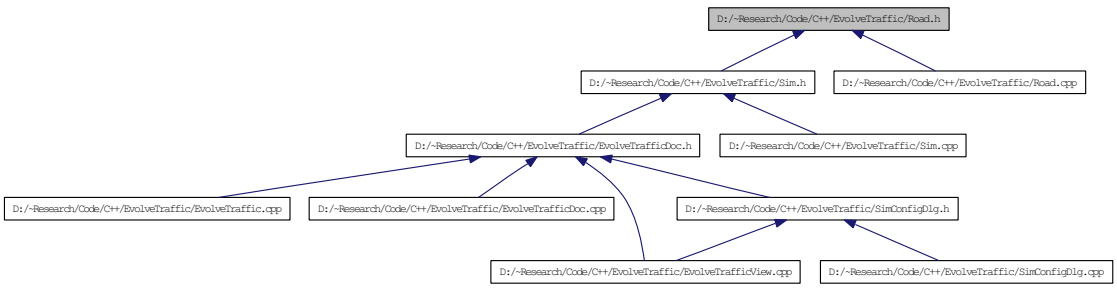
```
#include "stdafx.h"
```

```
#include "IDMParameterSet.h"
#include "RoadFeature.h"
#include "StatDetector.h"
#include "Direction.h"
#include "CASTORFile.h"
#include "SAFTFile.h"
#include "Constants.h"
#include <vector>
#include <fstream>
#include <algorithm>
```

Include dependency graph for Road.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Road

A class to represent a road in the simulation.

Typedefs

- typedef std::vector< std::vector< Vehicle * > > M2D

5.56.1 Typedef Documentation

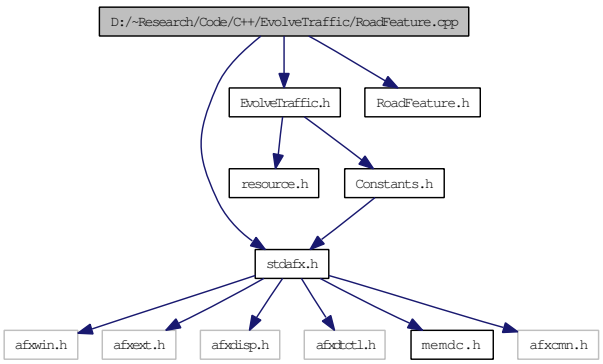
5.56.1.1 typedef std::vector< std::vector<Vehicle*> > M2D

Definition at line 17 of file Road.h.

5.57 D:/~Research/Code/C++/EvolveTraffic/RoadFeature.cpp File Reference

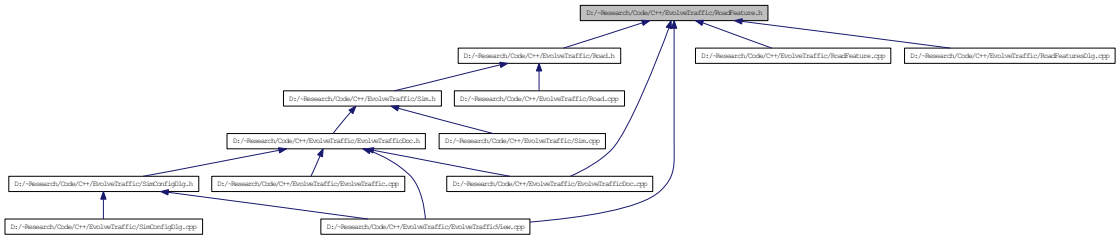
```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "RoadFeature.h"
```

Include dependency graph for RoadFeature.cpp:



5.58 D:/~Research/Code/C++/EvolveTraffic/RoadFeature.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class CRoadFeature

  A class for storing general Road Feature objects to file.

Defines

- #define AFX_ROADFEATURE_H__F64DE59C_1374_4951_8D85_-765E63B664DD__INCLUDED_

5.58.1 Define Documentation

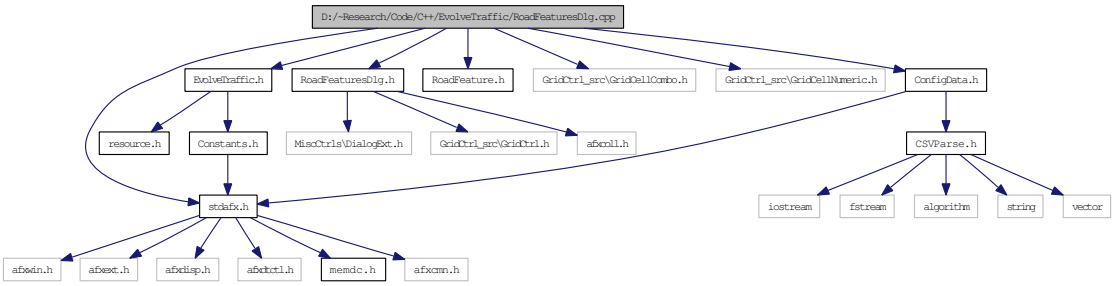
5.58.1.1 #define AFX_ROADFEATURE_H__F64DE59C_1374_4951_8D85_-765E63B664DD__INCLUDED_

Definition at line 6 of file RoadFeature.h.

5.59 D:/~Research/Code/C++/EvolveTraffic/RoadFeaturesDlg.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "RoadFeaturesDlg.h"
#include "RoadFeature.h"
#include "GridCtrl_src\GridCellCombo.h"
#include "GridCtrl_src\GridCellNumeric.h"
#include "ConfigData.h"
```

Include dependency graph for RoadFeaturesDlg.cpp:



Variables

- CConfigData g_ConfigData

5.59.1 Variable Documentation

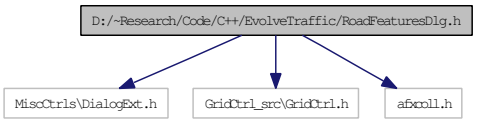
5.59.1.1 CConfigData g_ConfigData

Definition at line 32 of file ConfigData.cpp.

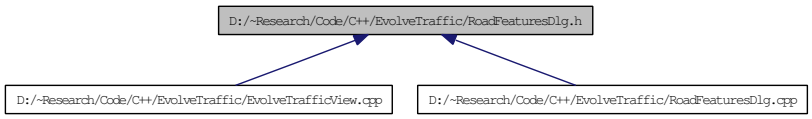
5.60 D:/~Research/Code/C++/EvolveTraffic/RoadFeaturesDlg.h File Reference

```
#include "MiscCtrls\DialogExt.h"
#include "GridCtrl_src\GridCtrl.h"
#include "afxcoll.h"
```

Include dependency graph for RoadFeaturesDlg.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CRoadFeaturesDlg

  A class for the Road Features Dialog.

Defines

- #define AFX_ROADFEATURESDLG_H__72BAD1EF_0F8E_4D4B_98E1_-C93C7732E4AD__INCLUDED_

5.60.1 Define Documentation

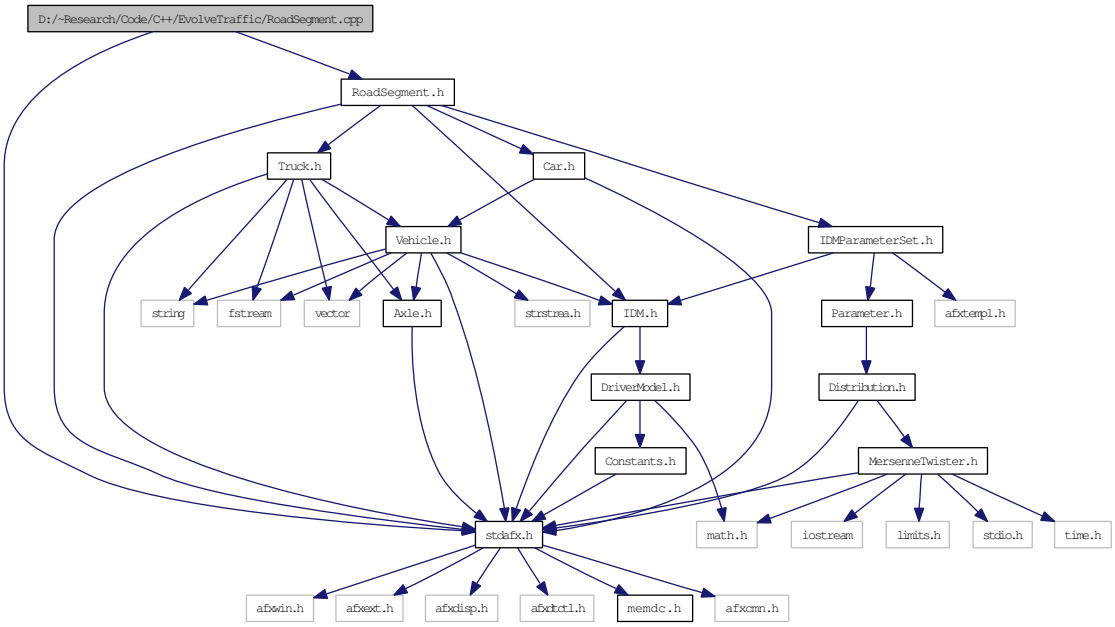
5.60.1.1 #define AFX_ROADFEATURESDLG_H__72BAD1EF_0F8E_4D4B_-98E1_C93C7732E4AD__INCLUDED_

Definition at line 2 of file RoadFeaturesDlg.h.

5.61 D:/~Research/Code/C++/EvolveTraffic/RoadSegment.cpp File Reference

```
#include "stdafx.h"
#include "RoadSegment.h"
```

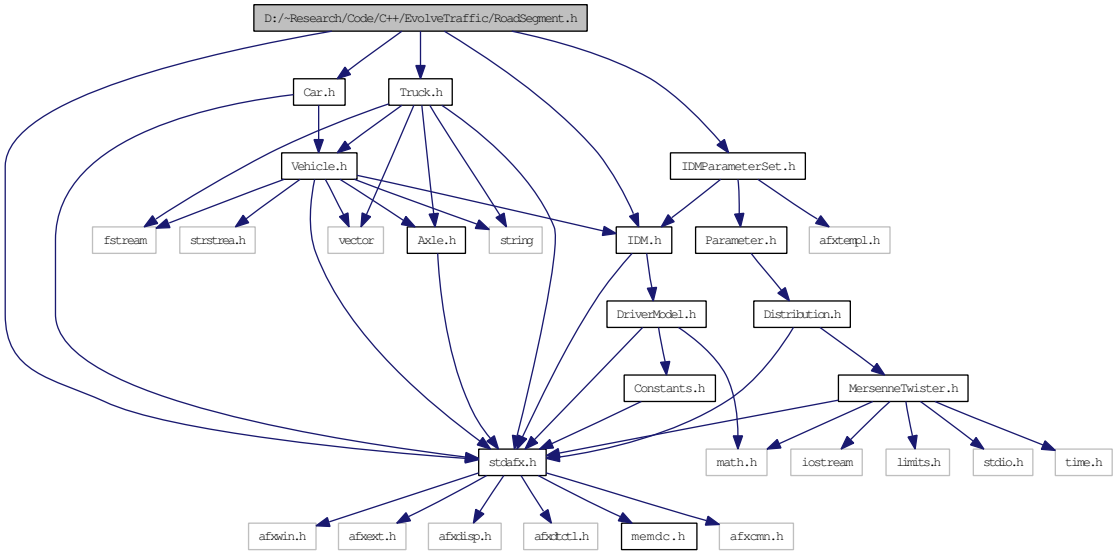
Include dependency graph for RoadSegment.cpp:



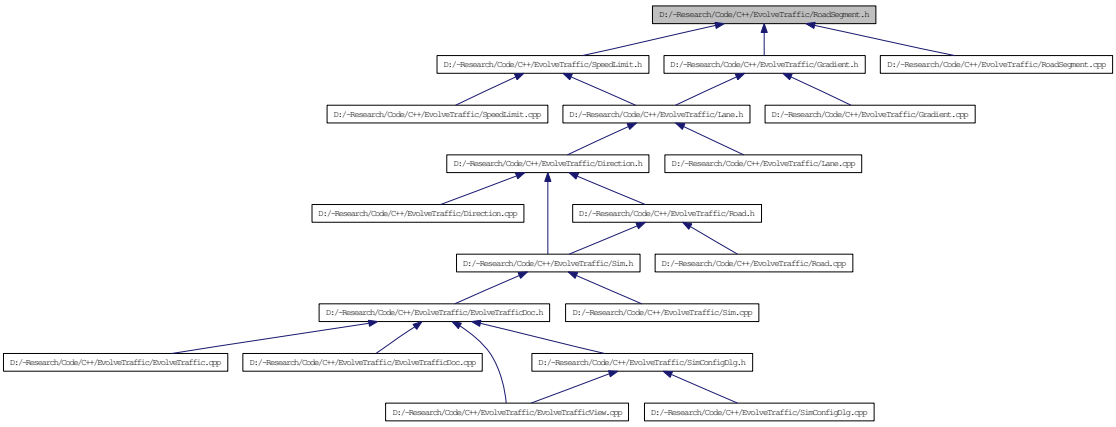
5.62 D:/~Research/Code/C++/EvolveTraffic/RoadSegment.h File Reference

```
#include "stdafx.h"
#include "Truck.h"
#include "Car.h"
#include "IDM.h"
#include "IDMParameterSet.h"
```

Include dependency graph for RoadSegment.h:



This graph shows which files directly or indirectly include this file:



Classes

- class RoadSegment

  A base class from which specific special road segments are derived.

Defines

- #define AFX_ROADSEGMENT_H__DFD0E9F6_3EB2_47CB_9D5E_-408357D1CF04__INCLUDED_

5.62.1 Define Documentation

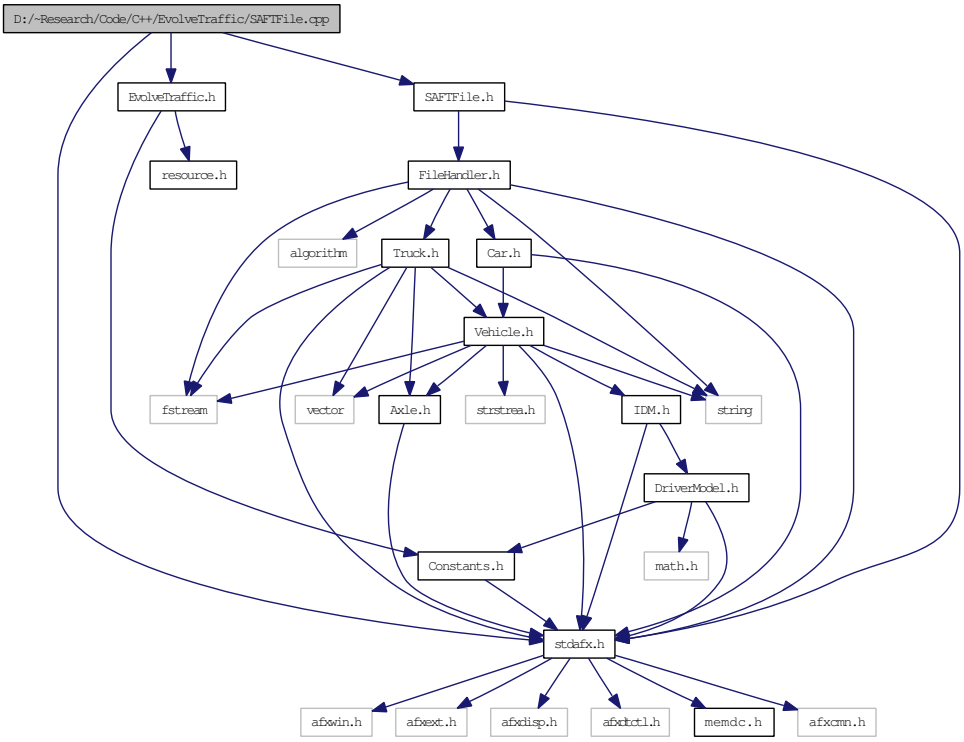
5.62.1.1 #define AFX_ROADSEGMENT_H__DFD0E9F6_3EB2_47CB_9D5E_-408357D1CF04__INCLUDED_

Definition at line 6 of file RoadSegment.h.

5.63 D:/~Research/Code/C++/EvolveTraffic/SAFTFile.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "SAFTFile.h"
```

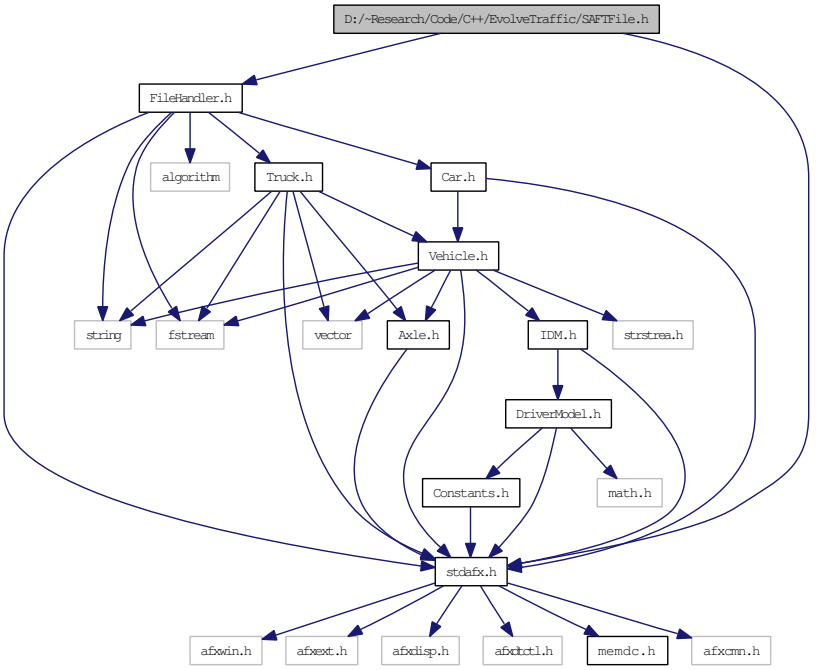
Include dependency graph for SAFTFile.cpp:



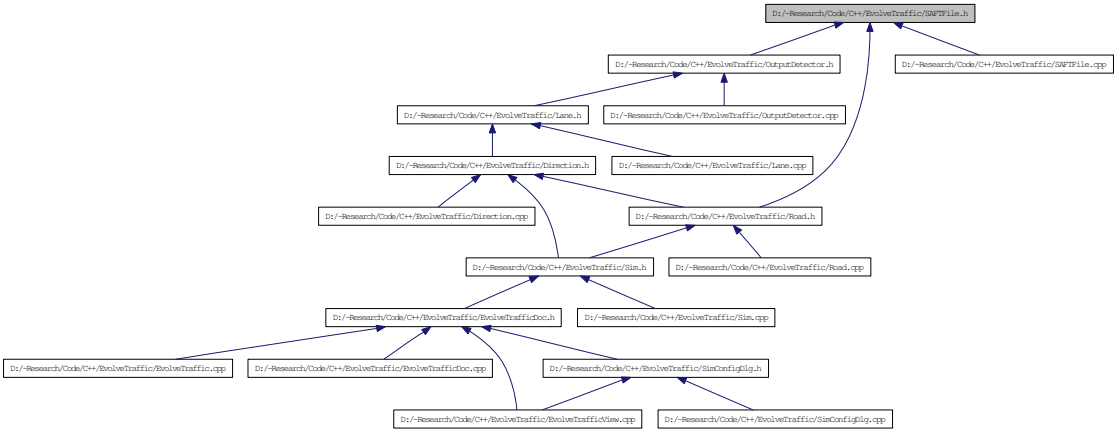
5.64 D:/~Research/Code/C++/EvolveTraffic/SAFTFile.h File Reference

```
#include "stdafx.h"
#include "FileHandler.h"
```

Include dependency graph for SAFTFile.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SAFTFile

  A class for reading from, and writing to, SAFT format files.

Defines

- #define AFX_SAFTFILE_H__33D68077_6D7E_4FDF_AEBB_-2B831E92D21D__INCLUDED_

5.64.1 Define Documentation

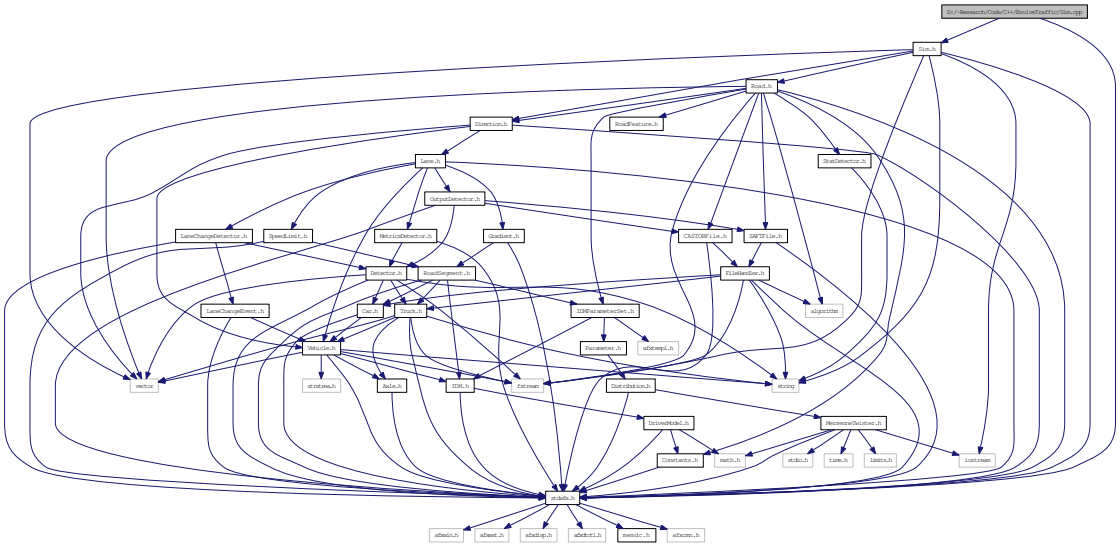
5.64.1.1 #define AFX_SAFTFILE_H__33D68077_6D7E_4FDF_AEBB_-2B831E92D21D__INCLUDED_

Definition at line 6 of file SAFTFile.h.

5.65 D:/~Research/Code/C++/EvolveTraffic/Sim.cpp File Reference

```
#include "stdafx.h"
#include "Sim.h"
```

Include dependency graph for Sim.cpp:

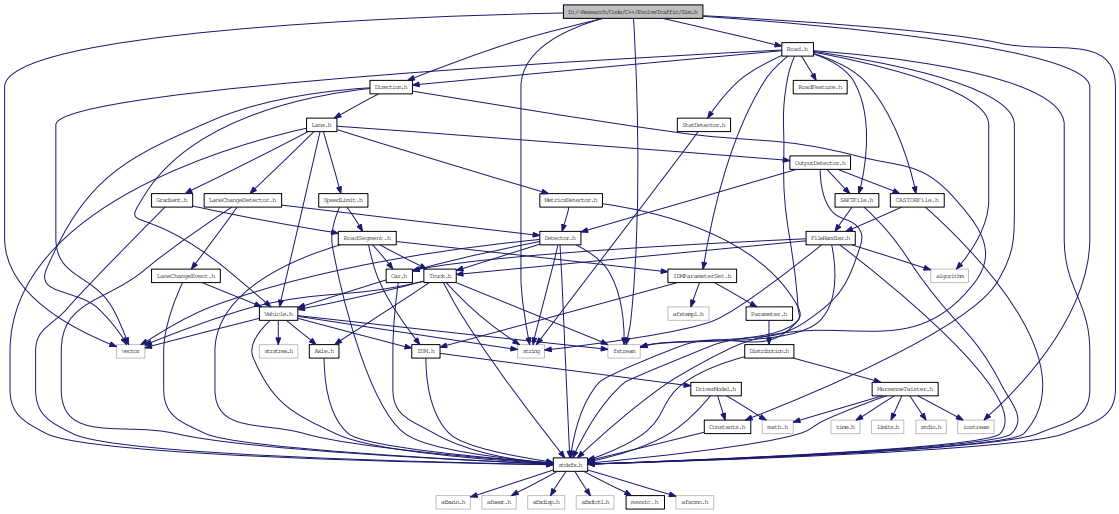


5.66 D:/~Research/Code/C++/EvolveTraffic/Sim.h File Reference

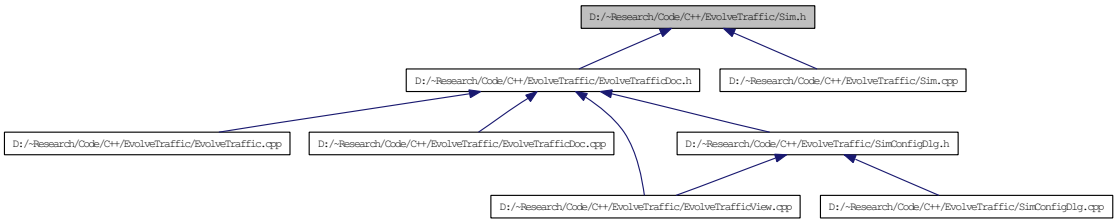
```
#include "stdafx.h"
#include "Direction.h"
#include "Road.h"
#include <vector>
#include <iostream>
```

```
#include <fstream>
#include <string>
```

Include dependency graph for Sim.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Sim

  A class to represent the simulation handler.

Typedefs

- typedef std::vector< std::vector< Vehicle * > > M2D

5.66.1 Typedef Documentation

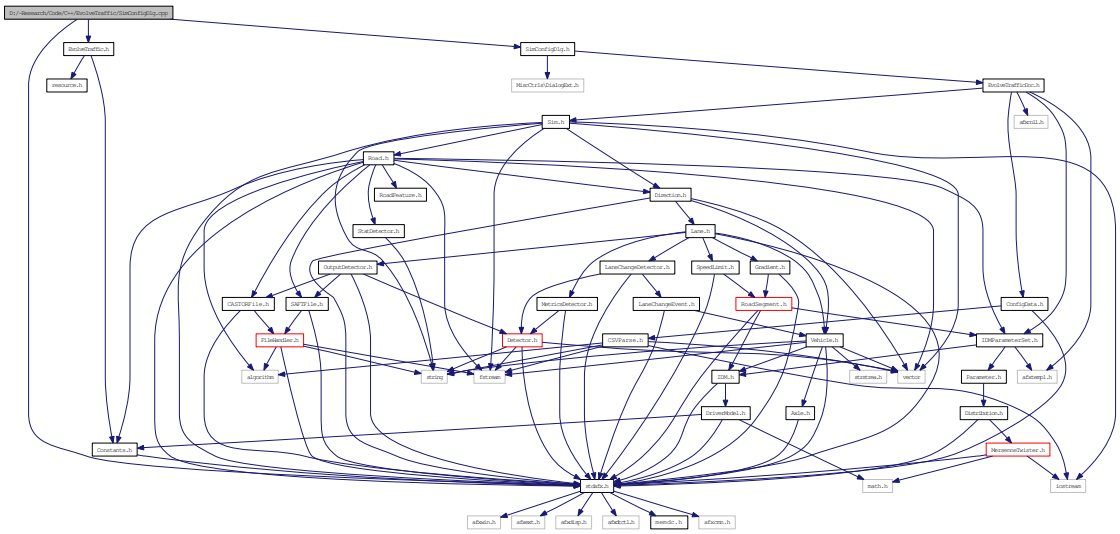
5.66.1.1 typedef std::vector< std::vector<Vehicle*> > M2D

Definition at line 14 of file Sim.h.

5.67 D:/~Research/Code/C++/EvolveTraffic/SimConfigDlg.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "SimConfigDlg.h"
```

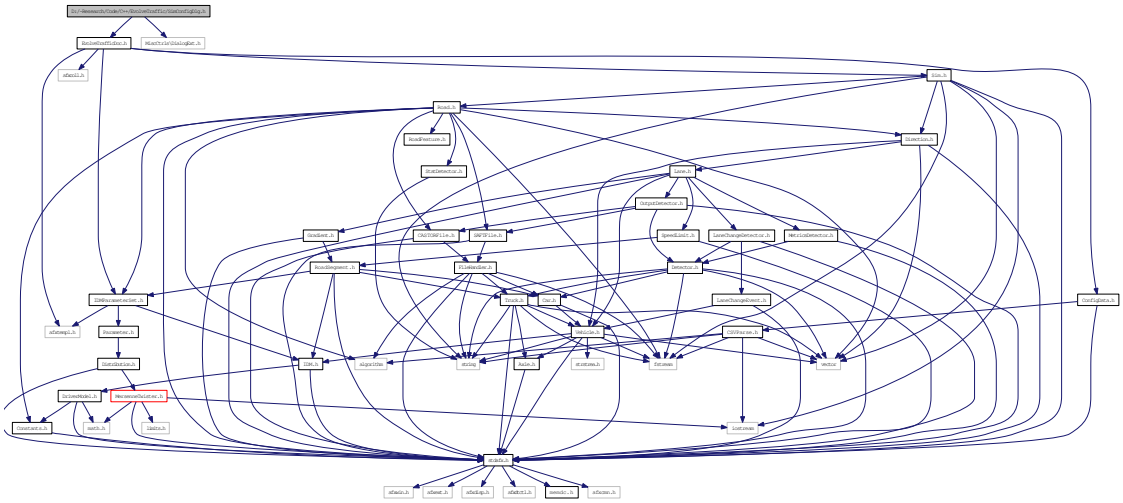
Include dependency graph for SimConfigDlg.cpp:



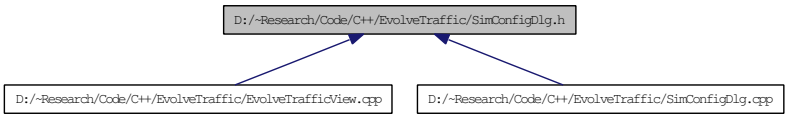
5.68 D:/~Research/Code/C++/EvolveTraffic/SimConfigDlg.h File Reference

```
#include "EvolveTrafficDoc.h"
#include "MiscCtrls\DialogExt.h"
```

Include dependency graph for SimConfigDlg.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CSimConfigDlg

  A class for the Simulation Configurations Dialog.

Defines

- #define AFX_SIMCONFIGDLG_H__66BC747D_1FD0_4187_9593_-A061AB8D38E2__INCLUDED_

5.68.1 Define Documentation

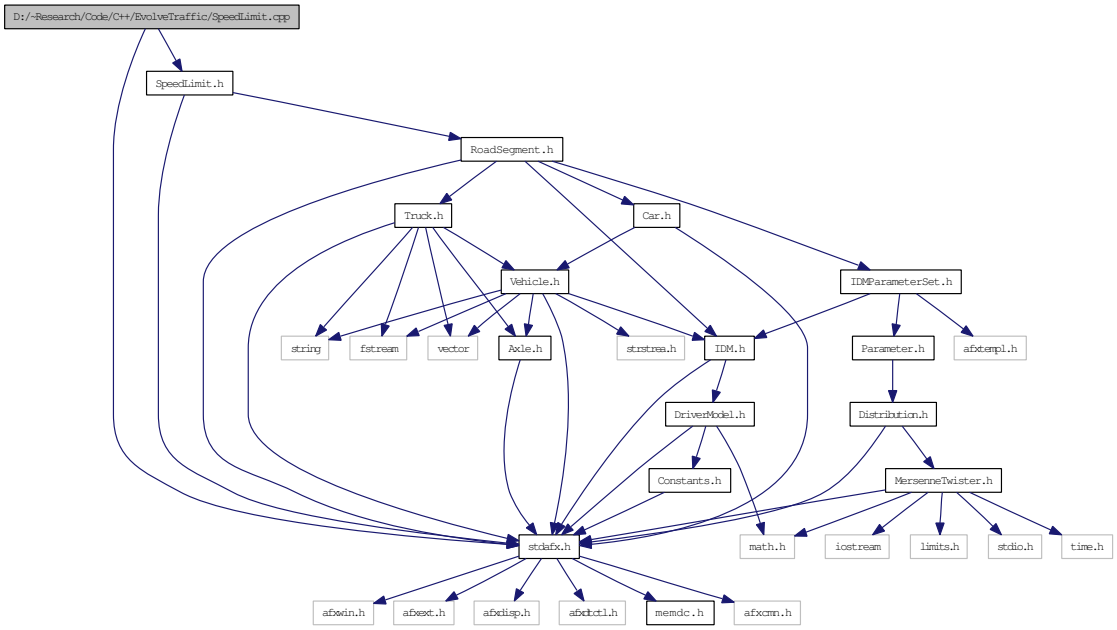
5.68.1.1 #define AFX_SIMCONFIGDLG_H__66BC747D_1FD0_4187_9593_-A061AB8D38E2__INCLUDED_

Definition at line 2 of file SimConfigDlg.h.

5.69 D:/~Research/Code/C++/EvolveTraffic/SpeedLimit.cpp File Reference

```
#include "stdafx.h"
#include "SpeedLimit.h"
```

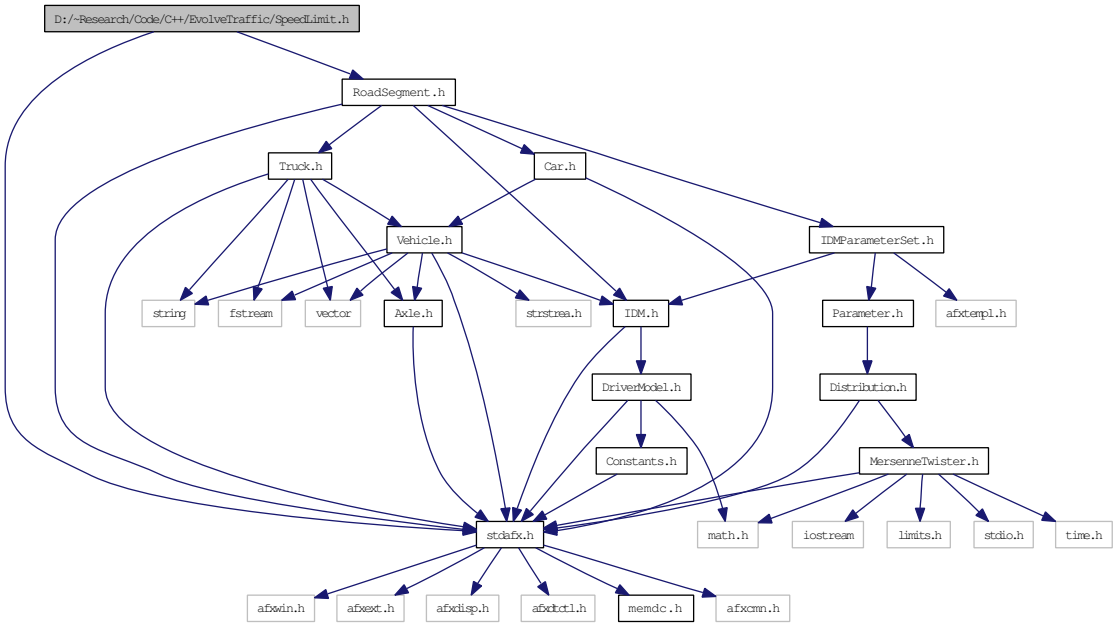
Include dependency graph for SpeedLimit.cpp:



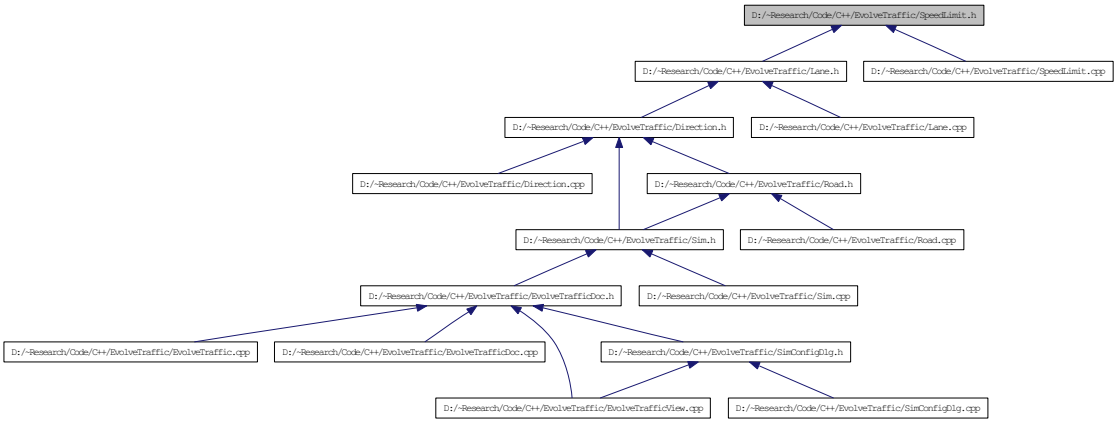
5.70 D:/~Research/Code/C++/EvolveTraffic/SpeedLimit.h File Reference

```
#include "stdafx.h"
#include "RoadSegment.h"
```

Include dependency graph for SpeedLimit.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SpeedLimit

  A derived class to represent a speed-limit section on a road.

Defines

- #define AFX_SPEEDLIMIT_H__1E65A464_295B_4549_A0A1_-B9174C5AEE6B__INCLUDED_

5.70.1 Define Documentation

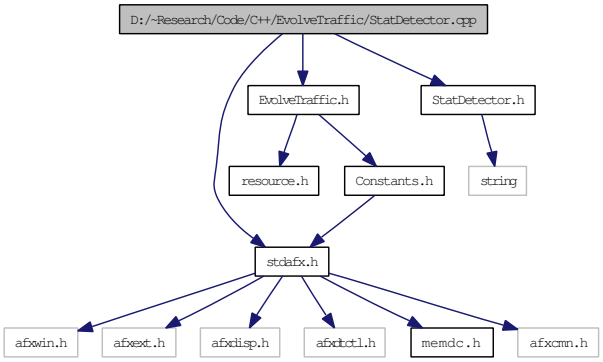
5.70.1.1 #define AFX_SPEEDLIMIT_H__1E65A464_295B_4549_A0A1_-B9174C5AEE6B__INCLUDED_

Definition at line 6 of file SpeedLimit.h.

5.71 D:/~Research/Code/C++/EvolveTraffic/StatDetector.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "StatDetector.h"
```

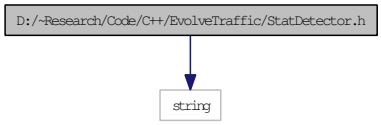
Include dependency graph for StatDetector.cpp:



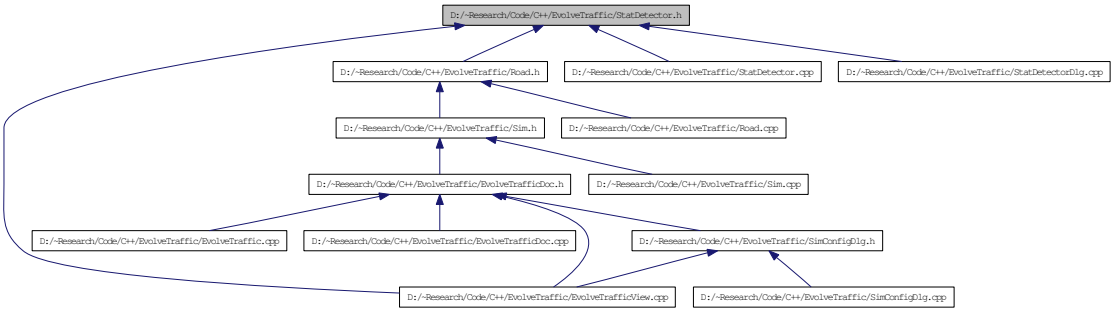
5.72 D:/~Research/Code/C++/EvolveTraffic/StatDetector.h File Reference

```
#include <string>
```

Include dependency graph for StatDetector.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CStatDetector

  A class for the general Detector object that can be saved to file.

Defines

- #define AFX_STATDETECTOR_H__09E4D8B1_AFFB_46DA_B96A_-5AC5192CA1EE__INCLUDED_

5.72.1 Define Documentation

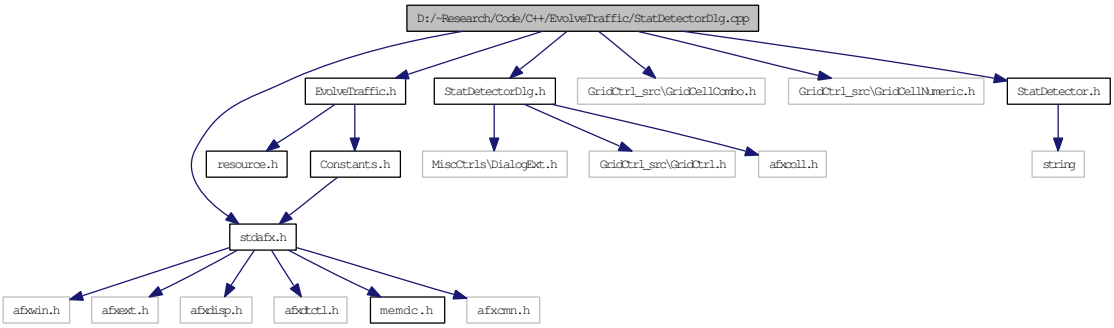
5.72.1.1 #define AFX_STATDETECTOR_H__09E4D8B1_AFFB_46DA_-B96A_5AC5192CA1EE__INCLUDED_

Definition at line 6 of file StatDetector.h.

5.73 D:/~Research/Code/C++/EvolveTraffic/StatDetectorDlg.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "StatDetectorDlg.h"
#include "GridCtrl_src\GridCellCombo.h"
#include "GridCtrl_src\GridCellNumeric.h"
#include "StatDetector.h"
```

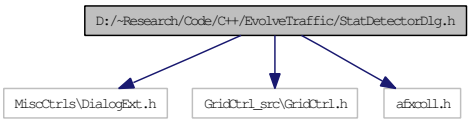
Include dependency graph for StatDetectorDlg.cpp:



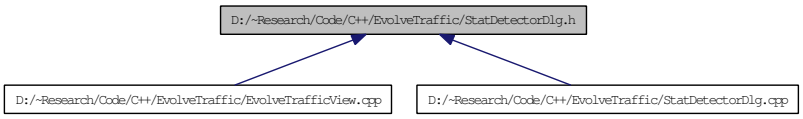
5.74 D:/~Research/Code/C++/EvolveTraffic/StatDetectorDlg.h File Reference

```
#include "MiscCtrls\DialogExt.h"
#include "GridCtrl_src\GridCtrl.h"
#include "afxcoll.h"
```

Include dependency graph for StatDetectorDlg.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CStatDetectorDlg

  A class for the Detector Dialog.

Defines

- #define AFX_STATDETECTORDLG_H__80DF71D6_83E9_4446_BD1B_-D6E8985A9548__INCLUDED_

5.74.1 Define Documentation

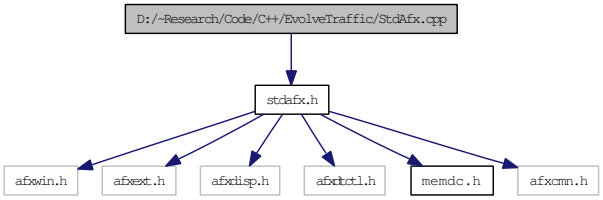
5.74.1.1 #define AFX_STATDETECTORDLG_H__80DF71D6_83E9_4446_-BD1B_D6E8985A9548__INCLUDED_

Definition at line 2 of file StatDetectorDlg.h.

5.75 D:/~Research/Code/C++/EvolveTraffic/StdAfx.cpp File Reference

```
#include "stdafx.h"
```

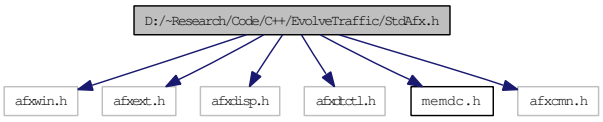
Include dependency graph for StdAfx.cpp:



5.76 D:/~Research/Code/C++/EvolveTraffic/StdAfx.h File Reference

```
#include <afxwin.h>
#include <afxext.h>
#include <afxdisp.h>
#include <afxdtctl.h>
#include "memdc.h"
#include <afxcmn.h>
```

Include dependency graph for StdAfx.h:



Defines

- #define AFX_STDAFX_H__61721CC1_F41A_456E_AF82_F05D2EF92301_-_INCLUDED_
- #define VC_EXTRALEAN

5.76.1 Define Documentation

5.76.1.1 #define AFX_STDAFX_H__61721CC1_F41A_456E_AF82_F05D2EF92301__INCLUDED_

Definition at line 7 of file StdAfx.h.

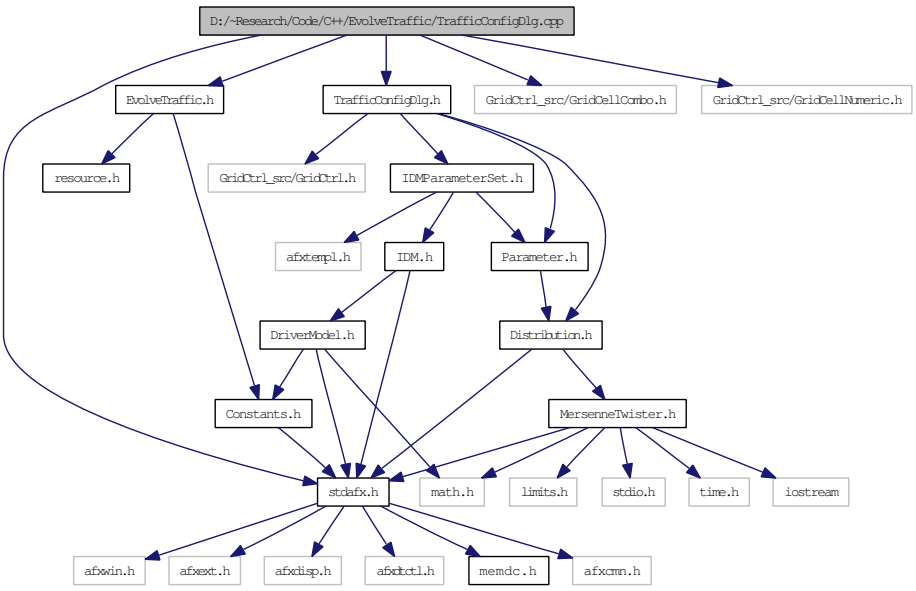
5.76.1.2 #define VC_EXTRALEAN

Definition at line 13 of file StdAfx.h.

5.77 D:/~Research/Code/C++/EvolveTraffic/TrafficConfigDlg.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
#include "TrafficConfigDlg.h"
#include "GridCtrl_src/GridCellCombo.h"
#include "GridCtrl_src/GridCellNumeric.h"
```

Include dependency graph for TrafficConfigDlg.cpp:

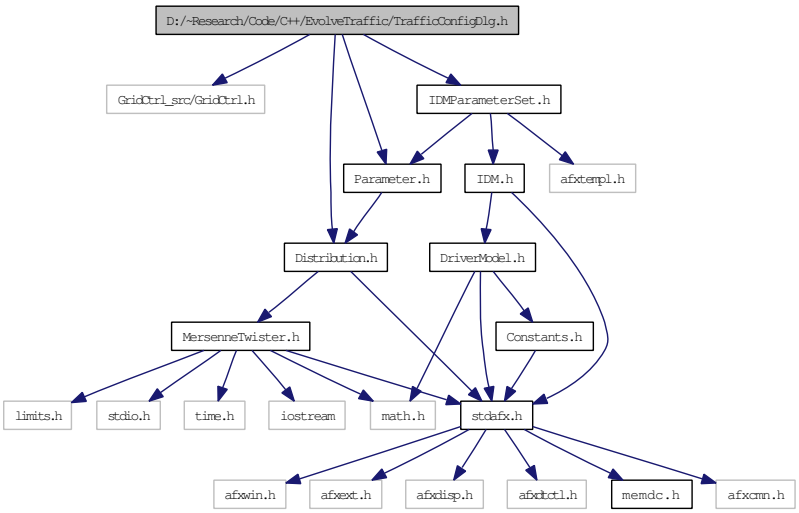


5.78 D:/~Research/Code/C++/EvolveTraffic/TrafficConfigDlg.h File Reference

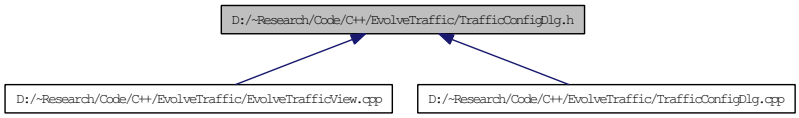
```
#include "GridCtrl_src/GridCtrl.h"
```

```
#include "IDMParameterSet.h"
#include "Parameter.h"
#include "Distribution.h"
```

Include dependency graph for TrafficConfigDlg.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CTrafficConfigDlg

  A class for the IDM Model Configuration Dialog.

Defines

- #define AFX_TRAFFICCONFIGDLG_H__6AA97537_057C_46E9_B1CC_B08FC4C93A1D__INCLUDED_

5.78.1 Define Documentation

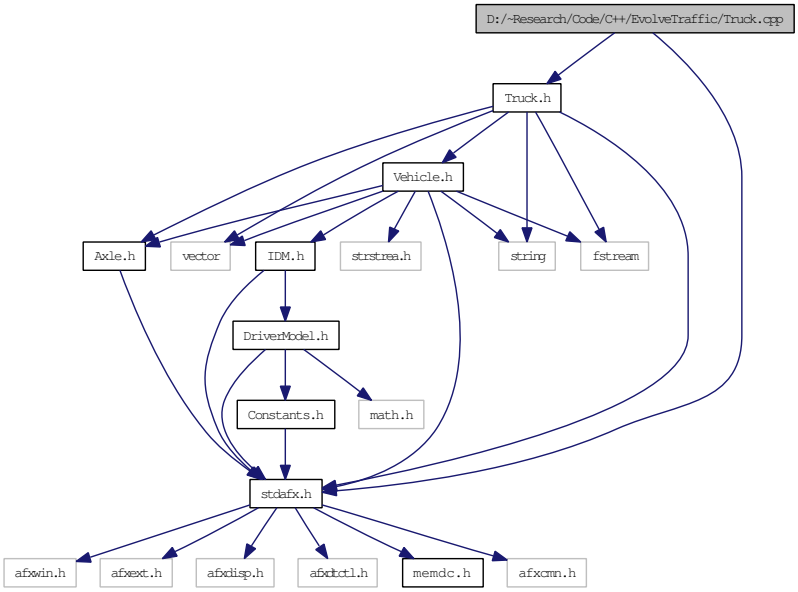
5.78.1.1 #define AFX_TRAFFICCONFIGDLG_H__6AA97537_057C_46E9_B1CC_B08FC4C93A1D__INCLUDED_

Definition at line 2 of file TrafficConfigDlg.h.

5.79 D:/~Research/Code/C++/EvolveTraffic/Truck.cpp File Reference

```
#include "stdafx.h"
#include "Truck.h"
```

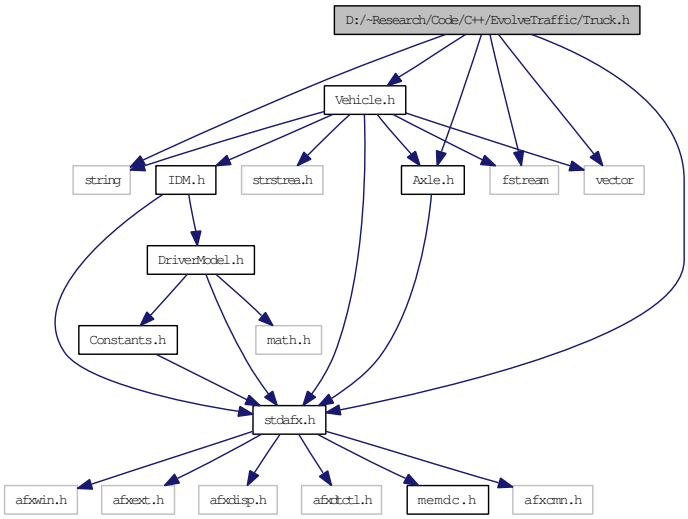
Include dependency graph for Truck.cpp:



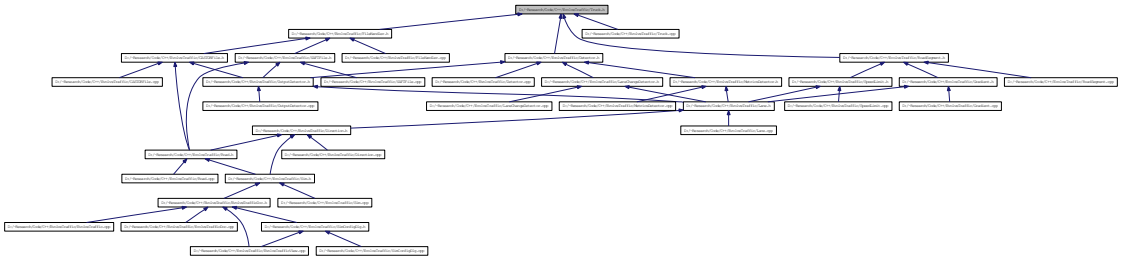
5.80 D:/~Research/Code/C++/EvolveTraffic/Truck.h File Reference

```
#include "stdafx.h"
#include <string>
#include <fstream>
#include <vector>
#include "Axle.h"
#include "Vehicle.h"
```

Include dependency graph for Truck.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Truck

  A class to represent a Truck.

Defines

- #define AFX_TRUCK_H__4FAD4F73_3B9C_4D49_A94D_-548AED8B7A7B__INCLUDED_

5.80.1 Define Documentation

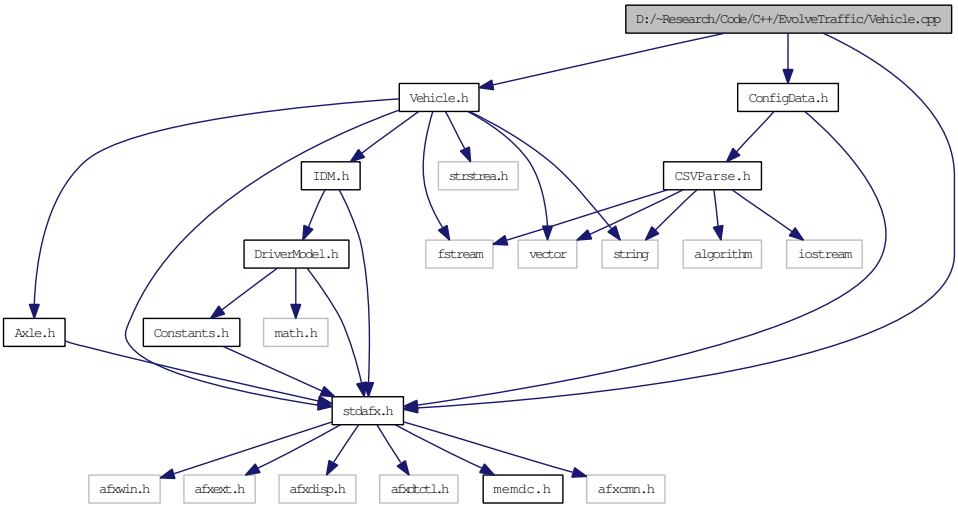
5.80.1.1 #define AFX_TRUCK_H__4FAD4F73_3B9C_4D49_A94D_-548AED8B7A7B__INCLUDED_

Definition at line 6 of file Truck.h.

5.81 D:/~Research/Code/C++/EvolveTraffic/Vehicle.cpp File Reference

```
#include "stdafx.h"
#include "Vehicle.h"
#include "ConfigData.h"
```

Include dependency graph for Vehicle.cpp:



Variables

- CConfigData g_ConfigData

5.81.1 Variable Documentation

5.81.1.1 CConfigData g_ConfigData

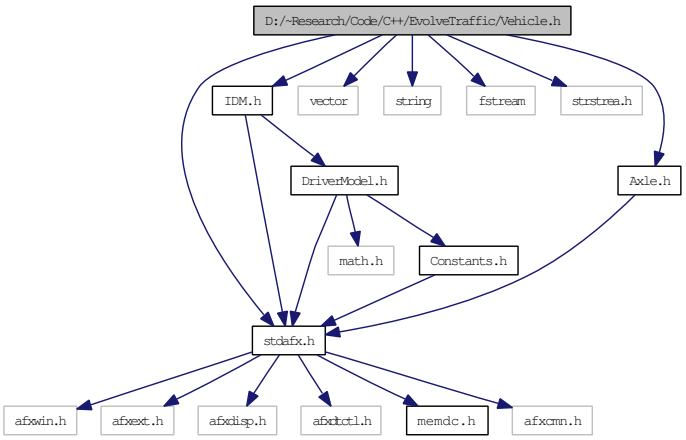
Definition at line 32 of file ConfigData.cpp.

5.82 D:/~Research/Code/C++/EvolveTraffic/Vehicle.h File Reference

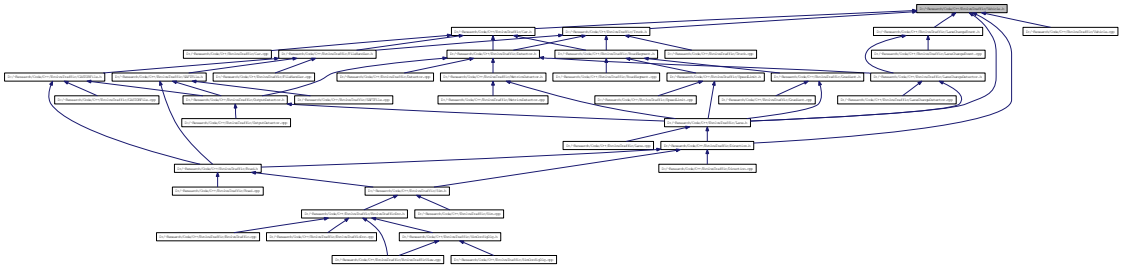
```
#include "stdafx.h"
#include "IDM.h"
#include <vector>
```

```
#include <string>
#include <fstream>
#include <strstrea.h>
#include "Axle.h"
```

Include dependency graph for Vehicle.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Vehicle

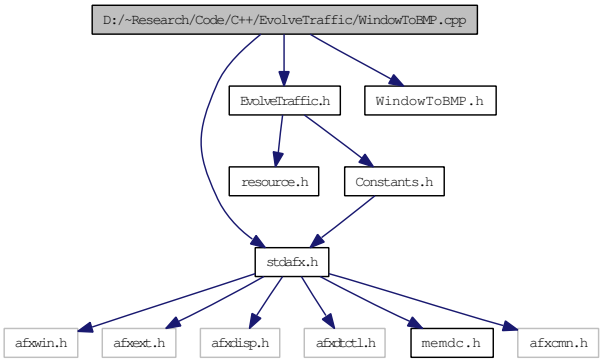
  A base class from which specific Vehicle types are derived.

5.83 D:/~Research/Code/C++/EvolveTraffic/WindowToBMP.cpp File Reference

```
#include "stdafx.h"
#include "EvolveTraffic.h"
```

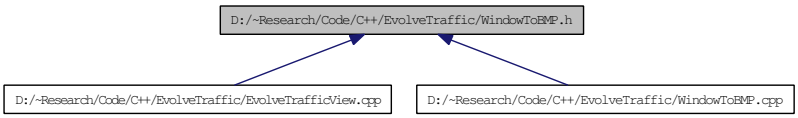
```
#include "WindowToBMP.h"
```

Include dependency graph for WindowToBMP.cpp:



5.84 D:/~Research/Code/C++/EvolveTraffic/WindowToBMP.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class CWindowToBMP

  A class for writing a windows content to a BMP file.

Defines

- #define AFX_WINDOWTOBMP_H__0A2334F8_B274_49DB_BEDB_-E043A546DB61__INCLUDED_

5.84.1 Define Documentation

5.84.1.1 #define AFX_WINDOWTOBMP_H__0A2334F8_B274_49DB_BEDB_-E043A546DB61__INCLUDED_

Definition at line 6 of file WindowToBMP.h.