CSE6730 Modeling & Simulation : Project-1

Generated by Doxygen 1.8.3.1

Thu Feb 21 2013 16:58:54

Contents

1 Hierarchical Index **1**

1.1 Class Hierarchy 1

2 Class Index **3**

2.1 Class List 3

3 File Index **5**

3.1 File List 5

4 Class Documentation **7**

4.1 _Topology Class Reference 7

4.1.1 Constructor & Destructor Documentation 8

4.1.1.1 _Topology 8

4.1.2 Member Data Documentation 8

4.1.2.1 ExitQ 8

4.1.2.2 I1 8

4.1.2.3 I2 8

4.1.2.4 I3 8

4.1.2.5 I4 8

4.1.2.6 I5 8

4.2 calender_queue Class Reference 9

4.2.1 Detailed Description 9

4.2.2 Constructor & Destructor Documentation 9

4.2.2.1 calender_queue 9

4.2.2.2 calender_queue 9

4.2.3 Member Function Documentation 9

4.2.3.1 check659bucket 9

4.2.3.2 check659bucket 10

4.2.3.3 dequeue 10

4.2.3.4 dequeue 10

4.2.3.5 get_bucket_count 10

4.2.3.6 get_bucket_count 10

4.2.3.7 getQsize 10
4.2.3.8 getQsize 10
4.2.3.9 gettimeframe 10
4.2.3.10 gettimeframe 11
4.2.3.11 init 11
4.2.3.12 insert 11
4.2.3.13 isEmpty 11
4.2.3.14 isEmpty 11
4.2.3.15 next_event 11
4.2.3.16 next_event 11
4.2.3.17 PopNext 11
4.2.3.18 PopNext 12
4.2.3.19 remove_event 12
4.2.3.20 remove_event 12
4.2.3.21 resize 12
4.3 Event0< T, OBJ > Class Template Reference 13
4.4 Event1< T, OBJ, U1, T1 > Class Template Reference 14
4.5 Event2< T, OBJ, U1, T1, U2, T2 > Class Template Reference 15
4.6 Event3< T, OBJ, U1, T1, U2, T2, U3, T3 > Class Template Reference 16
4.7 event_compare Class Reference 17
4.8 EventBase Class Reference 17
4.9 eventDsc Struct Reference 18
4.10 Intersection Class Reference 18
4.10.1 Constructor & Destructor Documentation 20
4.10.1.1 Intersection 20
4.10.1.2 Intersection 20
4.10.1.3 ~Intersection 20
4.10.2 Member Function Documentation 20
4.10.2.1 addVehicletoQueue 20
4.10.2.2 EvictQ 20
4.10.2.3 getID 21
4.10.2.4 getQdirection 21
4.10.2.5 getQlane 21
4.10.2.6 NextQInfo 21
4.10.2.7 QCanGo 22
4.10.2.8 VehicleDeparture 22
4.10.2.9 VehiclePass 23
4.10.3 Member Data Documentation 23
4.10.3.1 busy 23
4.10.3.2 EBI1 24

4.10.3.3 EBI2 24
4.10.3.4 ExitQ 24
4.10.3.5 haveSignal 24
4.10.3.6 ID 24
4.10.3.7 NBI1 24
4.10.3.8 NBI2 24
4.10.3.9 NInter 24
4.10.3.10 routingtable 24
4.10.3.11 SBI1 24
4.10.3.12 SBI2 24
4.10.3.13 SInter 24
4.10.3.14 WBI1 25
4.10.3.15 WBI2 25
4.11 IntersectionwithoutSignal Class Reference 25
4.11.1 Constructor & Destructor Documentation 26
4.11.1.1 IntersectionwithoutSignal 26
4.11.1.2 IntersectionwithoutSignal 26
4.11.1.3 ∼IntersectionwithoutSignal 27
4.11.2 Member Function Documentation 27
4.11.2.1 addVehicletoQueue 27
4.11.2.2 QCanGo 27
4.12 IntersectionwithSignal Class Reference 28
4.12.1 Constructor & Destructor Documentation 30
4.12.1.1 IntersectionwithSignal 30
4.12.1.2 IntersectionwithSignal 30
4.12.1.3 ∼IntersectionwithSignal 30
4.12.2 Member Function Documentation 30
4.12.2.1 addVehicletoQueue 30
4.12.2.2 changeSignalTrigger 30
4.12.2.3 QCanGo 31
4.12.3 Member Data Documentation 31
4.12.3.1 EB 32
4.12.3.2 NB 32
4.12.3.3 SB 32
4.12.3.4 WB 32
4.13 node Class Reference 32
4.13.1 Detailed Description 32
4.13.2 Constructor & Destructor Documentation 32
4.13.2.1 node 32
4.13.3 Member Function Documentation 32

4.13.3.1 getTime 32
4.14 prioqueue Class Reference 33
4.15 RandomNumGen Class Reference 33
4.15.1 Constructor & Destructor Documentation 33
4.15.1.1 RandomNumGen 33
4.15.1.2 RandomNumGen 33
4.15.1.3 ∼RandomNumGen 34
4.15.2 Member Function Documentation 34
4.15.2.1 GetState 34
4.15.2.2 Next 34
4.15.2.3 Reset 34
4.16 RoadSegment Class Reference 34
4.16.1 Constructor & Destructor Documentation 35
4.16.1.1 RoadSegment 35
4.16.2 Member Function Documentation 35
4.16.2.1 AddVehicle 35
4.16.2.2 EvictVehicle 35
4.16.3 Member Data Documentation 35
4.16.3.1 qleft 35
4.16.3.2 qright 35
4.17 Simulator Class Reference 36
4.17.1 Constructor & Destructor Documentation 36
4.17.1.1 Simulator 36
4.17.2 Member Function Documentation 36
4.17.2.1 getNow 36
4.17.2.2 Now 37
4.17.2.3 Run 37
4.17.2.4 Schedule 37
4.17.2.5 Schedule 37
4.17.2.6 Schedule 38
4.17.2.7 Schedule 38
4.17.2.8 Stop 38
4.17.2.9 StopAt 39
4.17.3 Member Data Documentation 39
4.17.3.1 instance 39
4.18 TrafficLight Class Reference 39
4.18.1 Constructor & Destructor Documentation 39
4.18.1.1 TrafficLight 39
4.18.1.2 TrafficLight 40
4.18.1.3 ∼TrafficLight 40

4.18.2 Member Function Documentation . . . 40
4.18.2.1 cyclestate . . . 40
4.18.2.2 getLeftState . . . 40
4.18.2.3 getState . . . 40
4.18.2.4 getType . . . 40
4.18.3 Member Data Documentation . . . 41
4.18.3.1 myid . . . 41
4.18.3.2 type . . . 41
4.19 VehicleClass Class Reference . . . 41
4.19.1 Constructor & Destructor Documentation . . . 41
4.19.1.1 VehicleClass . . . 41
4.19.1.2 ∼VehicleClass . . . 42
4.19.2 Member Function Documentation . . . 42
4.19.2.1 EndTime . . . 42
4.19.2.2 getDestination . . . 42
4.19.2.3 getDirection . . . 42
4.19.2.4 getID . . . 42
4.19.2.5 getLastQ . . . 42
4.19.2.6 getSource . . . 42
4.19.2.7 setEndTime . . . 42
4.19.2.8 setLastQ . . . 42
4.19.2.9 StartTime . . . 43
4.19.2.10 updateDirection . . . 43
4.19.3 Member Data Documentation . . . 43
4.19.3.1 endTime . . . 43
4.19.3.2 startTime . . . 43
4.20 VehicleQueue Class Reference . . . 43
4.20.1 Constructor & Destructor Documentation . . . 43
4.20.1.1 VehicleQueue . . . 43
4.20.1.2 VehicleQueue . . . 44
4.20.1.3 VehicleQueue . . . 44
4.20.2 Member Function Documentation . . . 44
4.20.2.1 back . . . 44
4.20.2.2 empty . . . 44
4.20.2.3 front . . . 44
4.20.2.4 GetLen . . . 44
4.20.2.5 GetMaxLen . . . 44
4.20.2.6 isBusy . . . 44
4.20.2.7 pop . . . 44
4.20.2.8 push . . . 44

4.20.3 Member Data Documentation 44
4.20.3.1 busy 45
4.20.3.2 LastSentCar 45
4.20.3.3 maxLength 45
4.20.3.4 Q1 45

5 File Documentation 47

5.1 calender_queue.h File Reference 47
5.1.1 Detailed Description 48
5.1.2 Macro Definition Documentation 48
5.1.2.1 BUCKET_COUNT 48
5.1.2.2 CALENDER_PERIOD 48
5.1.2.3 TOTAL_TIME 49
5.2 CommonDefs.h File Reference 49
5.2.1 Detailed Description 50
5.2.2 Macro Definition Documentation 50
5.2.2.1 BurstTime 50
5.2.2.2 checkQinterval 50
5.2.2.3 LPassTime 50
5.2.2.4 PassTime 50
5.2.2.5 roadSegTime 50
5.2.2.6 startToPass 51
5.2.3 Typedef Documentation 51
5.2.3.1 Time_t 51
5.2.4 Function Documentation 51
5.2.4.1 reg 51
5.2.4.2 turn 51
5.3 Events.h File Reference 52
5.3.1 Detailed Description 53
5.4 Intersection.h File Reference 53
5.4.1 Detailed Description 55
5.5 IntersectionwithSignal.h File Reference 55
5.5.1 Detailed Description 56
5.6 IntersectionwoSignal.h File Reference 57
5.6.1 Detailed Description 58
5.7 main.cpp File Reference 58
5.8 PostProcessing.h File Reference 59
5.8.1 Detailed Description 60
5.8.2 Function Documentation 61
5.8.2.1 PostProcStats 61

5.9 RandomNum.cc File Reference 61
5.9.1 Detailed Description 62
5.10 RandomNum.h File Reference 62
5.10.1 Detailed Description 63
5.10.2 Macro Definition Documentation 63
5.10.2.1 MINSTDG 63
5.10.2.2 MINSTDM 63
5.10.2.3 MINSTDX 63
5.10.3 Function Documentation 63
5.10.3.1 gettime 63
5.11 RoadSegment.h File Reference 63
5.11.1 Detailed Description 63
5.12 scheduleVehicles.h File Reference 63
5.12.1 Detailed Description 64
5.12.2 Function Documentation 64
5.12.2.1 scheduleVehicles 65
5.13 Simulator.h File Reference 65
5.13.1 Detailed Description 66
5.14 Topology.h File Reference 66
5.14.1 Detailed Description 67
5.15 TrafficLight.h File Reference 68
5.15.1 Detailed Description 69
5.16 VehicleClass.h File Reference 69
5.16.1 Detailed Description 71

Index **71**

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

_Topology 7
calender_queue 9
event_compare 17
EventBase 17
    Event0< T, OBJ > 13
    Event1< T, OBJ, U1, T1 > 14
    Event2< T, OBJ, U1, T1, U2, T2 > 15
    Event3< T, OBJ, U1, T1, U2, T2, U3, T3 > 16
eventDsc 18
Intersection 18
    IntersectionwithoutSignal 25
    IntersectionwithSignal 28
node 32
prioqueue 33
RandomNumGen 33
RoadSegment 34
Simulator 36
TrafficLight 39
VehicleClass 41
VehicleQueue 43

Chapter 2

Class Index

2.1 Class List

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

_Topology 7
calender_queue 9
Event0$<$ T, OBJ $>$ 13
Event1$<$ T, OBJ, U1, T1 $>$ 14
Event2$<$ T, OBJ, U1, T1, U2, T2 $>$ 15
Event3$<$ T, OBJ, U1, T1, U2, T2, U3, T3 $>$ 16
event_compare 17
EventBase 17
eventDsc 18
Intersection 18
IntersectionwithoutSignal 25
IntersectionwithSignal 28
node 32
prioqueue 33
RandomNumGen 33
RoadSegment 34
Simulator 36
TrafficLight 39
VehicleClass 41
VehicleQueue 43

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

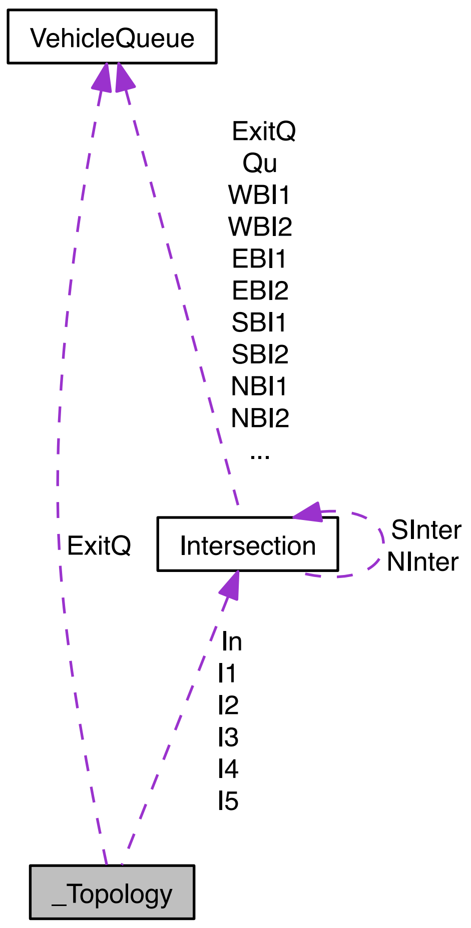
calender_queue.h
Declartion of the class calender queue 47
CommonDefs.h 49
Events.h
Declaration of various types of events 52
Intersection.h 53
IntersectionwithSignal.h 55
IntersectionwoSignal.h 57
main.cpp 58
PostProcessing.h 59
prioqueue.h . . . . . **??**
RandomNum.cc 61
RandomNum.h 62
RoadSegment.h 63
scheduleVehicles.h 63
Simulator.h 65
stdafx.h . . . . . **??**
Topology.h 66
TrafficLight.h
Description of functionality of traffic light 68
VehicleClass.h 69
VehicleQueue.h . . . . . **??**
testing/**calender_queue_testing.h** . . . . . **??**
testing/**test1.h** . . . . . **??**
testing/**testing.h** . . . . . **??**

Chapter 4

Class Documentation

4.1 _Topology Class Reference

Collaboration diagram for _Topology:



Public Member Functions

- _Topology ()

Public Attributes

- Intersection ∗ I1
- Intersection ∗ I2
- Intersection ∗ I3
- Intersection ∗ I4
- Intersection ∗ I5
- Intersection ∗ **In** [5]
- VehicleQueue ∗ ExitQ

4.1.1 Constructor & Destructor Documentation

4.1.1.1 _Topology::_Topology ( ) [inline]

Default constructor Initializes the topology with intersections and

4.1.2 Member Data Documentation

4.1.2.1 VehicleQueue∗ _Topology::ExitQ

Holds a vehicles queue for post processing

4.1.2.2 Intersection∗ _Topology::I1

10th street

4.1.2.3 Intersection∗ _Topology::I2

11th street

4.1.2.4 Intersection∗ _Topology::I3

12th street

4.1.2.5 Intersection∗ _Topology::I4

13th street

4.1.2.6 Intersection∗ _Topology::I5

14th street

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

- Topology.h

4.2 calender_queue Class Reference

```
#include <calender_queue.h>
```

Public Member Functions

- void insert (EventBase ∗E1)
- void dequeue (EventBase ∗E1)
- EventBase ∗ PopNext ()
- EventBase ∗ next_event (int bucket_num)
- void remove_event (int bucket_num, EventBase ∗E1)
- int isEmpty ()
- int get_bucket_count ()
- calender_queue ()
- int getQsize ()
- int gettimeframe ()
- void check659bucket ()
- void init (int num, double wid, double earliest)
- void resize ()
- void **insert** (node ∗E1)
- void dequeue (node ∗E1)
- node ∗ PopNext ()
- node ∗ next_event (int bucket_num)
- void remove_event (int bucket_num, node ∗E1)
- int isEmpty ()
- int get_bucket_count ()
- calender_queue (int bk, double int_width, double bk_sz)
- int getQsize ()
- int gettimeframe ()
- void check659bucket ()

4.2.1 Detailed Description

Calender Queue is priority queue, Ref: Calendar queues: a fast 0(1) priority queue implementation for the simulation event set problem

4.2.2 Constructor & Destructor Documentation

4.2.2.1 calender_queue::calender_queue ( )

Default constructor

4.2.2.2 calender_queue::calender_queue (int *bk,* double *int_width,* double *bk_sz*)

Default constructor constructs an calender queue with buck_count number of buckets

4.2.3 Member Function Documentation

4.2.3.1 void calender_queue::check659bucket ( )

is there debuggin purpose

4.2.3.2 void calender_queue::check659bucket ( )

is there debuggin purpose

4.2.3.3 void calender_queue::dequeue (EventBase * *E1*)

Removes an event E1 from the list (Hence it won't be scheduled)

Parameters

E1	: event pointer to removed from the list

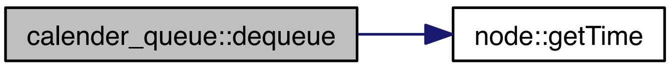
4.2.3.4 void calender_queue::dequeue (node * *E1*)

Removes an event E1 from the list (Hence it won't be scheduled)

Parameters

E1	: event pointer to removed from the list

Here is the call graph for this function:



4.2.3.5 int calender_queue::get_bucket_count ( )

Get number of buckets in the calender queue

4.2.3.6 int calender_queue::get_bucket_count ( )

Get number of buckets in the calender queue

4.2.3.7 int calender_queue::getQsize ( )

Returns number of element in the Q

4.2.3.8 int calender_queue::getQsize ( )

Returns number of element in the Q

4.2.3.9 int calender_queue::gettimeframe ( )

Returns the time frame from which last event was popped

4.2.3.10 int calender_queue::gettimeframe ( )

Returns the time frame from which last event was popped

4.2.3.11 void calender_queue::init (int *num,* double *wid,* double *earliest*)

Initilizes the queue with num buckets

4.2.3.12 void calender_queue::insert (EventBase * *E1*)

Inserts an event into the priority list

Parameters

E1	is the even to be inserted into the list

4.2.3.13 int calender_queue::isEmpty ( )

Checks if there are anymore events left in the list

4.2.3.14 int calender_queue::isEmpty ( )

Checks if there are anymore events left in the list

4.2.3.15 node * calender_queue::next_event (int *bucket_num*)

Returns event with minimum time from bucket-num

Parameters

bucket_num	is the number of bucket from which you want to get the min time stamp event

4.2.3.16 node* calender_queue::next_event (int *bucket_num*)

Returns event with minimum time from bucket-num

Parameters

bucket_num	is the number of bucket from which you want to get the min time stamp event

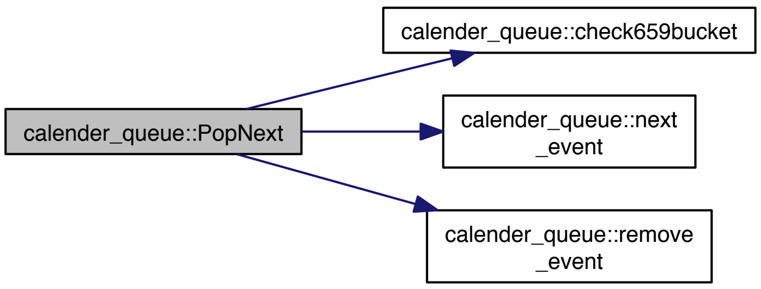
4.2.3.17 node * calender_queue::PopNext ( )

Pops Next event(event with minimum time stamp) in the list (and removes it as well)

Returns

Event pointer with minimum time stamp

Here is the call graph for this function:


calender_queue::check659bucket
calender_queue::PopNext
calender_queue::next _event
calender_queue::remove _event

4.2.3.18 node∗ calender_queue::PopNext ( )

Pops Next event(event with minimum time stamp) in the list (and removes it as well)

Returns

Event pointer with minimum time stamp

4.2.3.19 void calender_queue::remove_event (int *bucket_num,* EventBase ∗ *E1*)

Removes event E1 from bucket_num^th bucket

Parameters

BUcket_num	is the id of bucket from which event is to be removed
E1	is pointer to event to be removed

4.2.3.20 void calender_queue::remove_event (int *bucket_num,* node ∗ *E1*)

Removes event E1 from bucket_num^th bucket

Parameters

BUcket_num	is the id of bucket from which event is to be removed
E1	is pointer to event to be removed

4.2.3.21 void calender_queue::resize ( )

Inserts an event into the priority list

Parameters

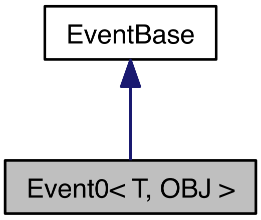
E1	is the even to be inserted into the list Resizes the calender queue based on

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

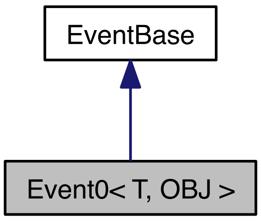
- calender_queue.h
- testing/calender_queue_testing.h
- calender_queue.cc
- testing/calender_queue_testing.cc

4.3 Event0< T, OBJ > Class Template Reference

Inheritance diagram for Event0< T, OBJ >:



Collaboration diagram for Event0< T, OBJ >:



Public Member Functions

- **Event0** (double t, void(T::∗f)(), OBJ ∗obj0)
- void **CallHandler** ()

Public Attributes

- void(T::∗ **handler**)(void)

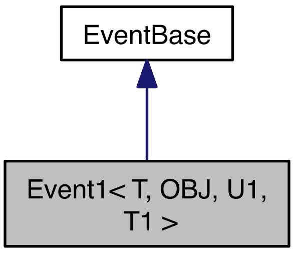
- OBJ ∗ **obj**

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

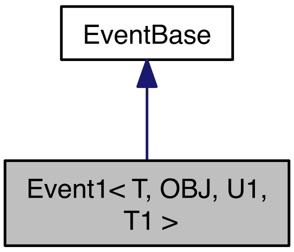
- Events.h

4.4 Event1< T, OBJ, U1, T1 > Class Template Reference

Inheritance diagram for Event1< T, OBJ, U1, T1 >:



Collaboration diagram for Event1< T, OBJ, U1, T1 >:



Public Member Functions

- **Event1** (double t, void(T::∗f)(U1), OBJ ∗obj0, T1 t1_0)
- void **CallHandler** ()

Public Attributes

- void(T::∗ **handler**)(U1)
- OBJ ∗ **obj**

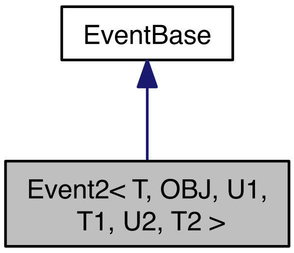
- T1 **t1**

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

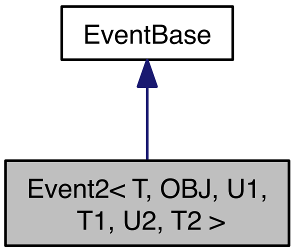
- Events.h

4.5 Event2< T, OBJ, U1, T1, U2, T2 > Class Template Reference

Inheritance diagram for Event2< T, OBJ, U1, T1, U2, T2 >:



Collaboration diagram for Event2< T, OBJ, U1, T1, U2, T2 >:



Public Member Functions

- **Event2** (double t, void(T::∗f)(U1, U2), OBJ ∗obj0, T1 t1_0, T2 t2_0)
- void **CallHandler** ()

Public Attributes

- void(T::∗ **handler**)(U1, U2)
- OBJ ∗ **obj**

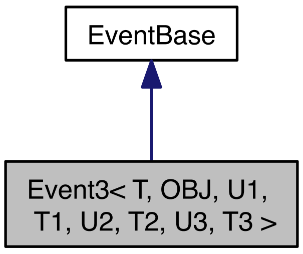
- T1 **t1**
- T2 **t2**

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

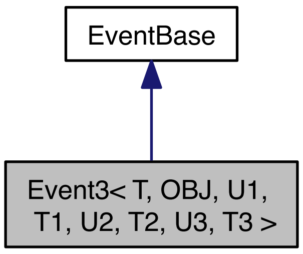
- Events.h

4.6 Event3< T, OBJ, U1, T1, U2, T2, U3, T3 > Class Template Reference

Inheritance diagram for Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >:



Collaboration diagram for Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >:



Public Member Functions

- **Event3** (double t, void(T::∗f)(U1, U2, U3), OBJ ∗obj0, T1 t1_0, T2 t2_0, T3 t3_0)
- void **CallHandler** ()

Public Attributes

- void(T::∗ **handler**)(U1, U2, U3)

- OBJ ∗ **obj**
- T1 **t1**
- T2 **t2**
- T3 **t3**

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

- Events.h

4.7 event_compare Class Reference

Public Member Functions

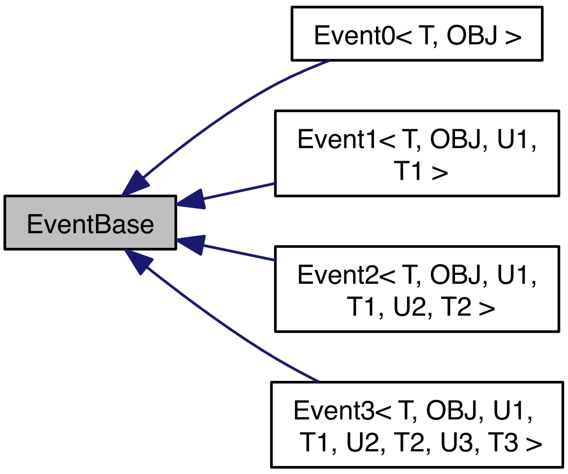
- bool **operator()** (EventBase ∗const &l, const EventBase ∗const &r) const

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

- Events.h

4.8 EventBase Class Reference

Inheritance diagram for EventBase:


Event0< T, OBJ >
Event1< T, OBJ, U1, T1 >
EventBase
Event2< T, OBJ, U1, T1, U2, T2 >
Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >

Public Member Functions

- **EventBase** (Time_t t)
- virtual void **CallHandler** ()=0
- Time_t **getTime** ()

Public Attributes

- Time_t **time**

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

- Events.h

4.9 eventDsc Struct Reference

Public Attributes

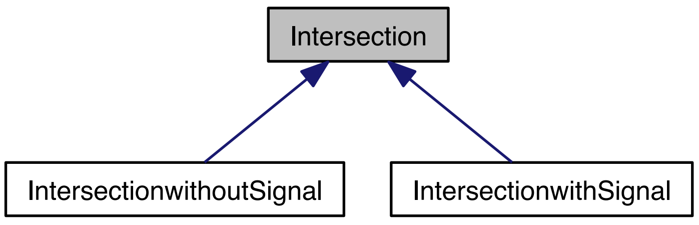
- int **type**
- int **InterID**
- int **QDir**
- int **QLane**
- int **QSize**
- double **timetag**

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

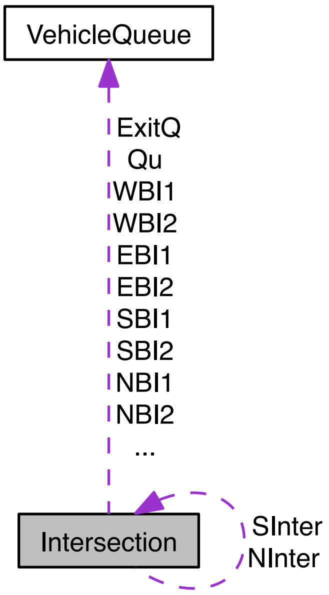
- testing/test1.h

4.10 Intersection Class Reference

Inheritance diagram for Intersection:



Collaboration diagram for Intersection:



Public Member Functions

- Intersection ()
- Intersection (int num)
- ∼Intersection ()
- int getID ()
- void VehiclePass (VehicleClass ∗vehicle, int Turn)
- void VehicleDeparture (VehicleClass ∗vehicle)
- void EvictQ (VehicleQueue ∗joinqueue)
- virtual void addVehicletoQueue (VehicleQueue ∗joinqueue, VehicleClass ∗vehicle)=0
- virtual int QCanGo (int direction, int lane)=0
- int getQdirection (Intersection ∗inter, VehicleQueue ∗Q)
- int getQlane (Intersection ∗inter, VehicleQueue ∗Q)
- void NextQInfo (VehicleQueue ∗currentQ, VehicleClass ∗vehicle, Intersection ∗&NextInter, VehicleQueue ∗&FutureQ, bool &isfull, int &Turn)

Public Attributes

- VehicleQueue ∗ EBI1
- VehicleQueue ∗ EBI2
- VehicleQueue ∗ WBI1
- VehicleQueue ∗ WBI2
- VehicleQueue ∗ NBI1
- VehicleQueue ∗ NBI2
- VehicleQueue ∗ SBI1
- VehicleQueue ∗ SBI2

- VehicleQueue ∗ **Qu** [4][2]
- dir routingtable [12]
- int **NBIlength**
- int **SBIlength**
- VehicleQueue ∗ ExitQ
- Intersection ∗ NInter
- Intersection ∗ SInter

Protected Attributes

- int ID
- bool haveSignal
- bool busy

4.10.1 Constructor & Destructor Documentation

4.10.1.1 Intersection::Intersection ( )

Default Constructor

4.10.1.2 Intersection::Intersection (int *num*)

Constructor

Parameters

num	

4.10.1.3 Intersection::∼Intersection ( )

Default destructor

4.10.2 Member Function Documentation

4.10.2.1 virtual void Intersection::addVehicletoQueue (VehicleQueue ∗ *joinqueue,* VehicleClass ∗ *vehicle*) `[pure virtual]`

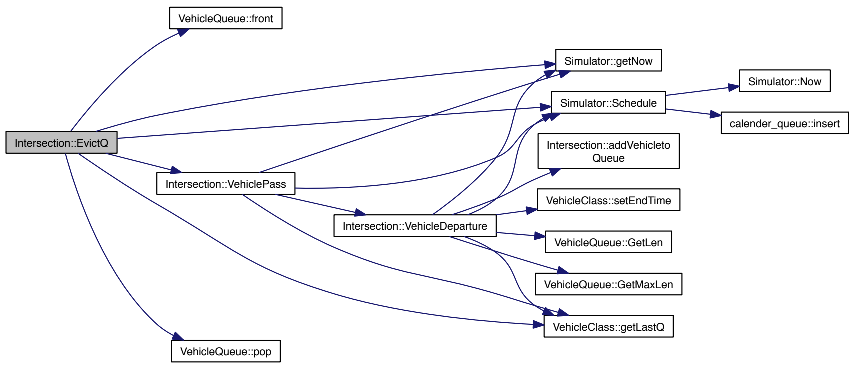
Virtual function Adds vehicle into queue

Implemented in IntersectionwithSignal, and IntersectionwithoutSignal.

4.10.2.2 void Intersection::EvictQ (VehicleQueue ∗ *joinqueue*)

Evicts the Vehicle Queue

Here is the call graph for this function:



4.10.2.3 int Intersection::getID ( ) [inline]

Returns the ID of the intersection

4.10.2.4 int Intersection::getQdirection (**Intersection** ∗ *inter*, **VehicleQueue** ∗ *Q*)

Gets the direction of the queue

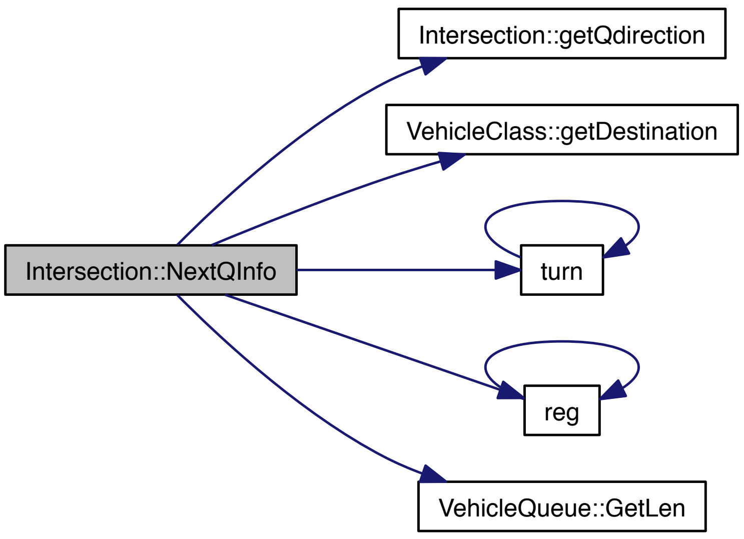
4.10.2.5 int Intersection::getQlane (**Intersection** ∗ *inter*, **VehicleQueue** ∗ *Q*)

Get the queue lane

4.10.2.6 void Intersection::NextQInfo (**VehicleQueue** ∗ *currentQ*, **VehicleClass** ∗ *vehicle*, **Intersection** ∗& *NextInter*, **VehicleQueue** ∗& *FutureQ*, bool & *isfull*, int & *Turn*)

Gets the next queue info

Here is the call graph for this function:



4.10.2.7 virtual int Intersection::QCanGo (int *direction,* int *lane*) `[pure virtual]`

QCAnGo

Implemented in IntersectionwithSignal, and IntersectionwithoutSignal.

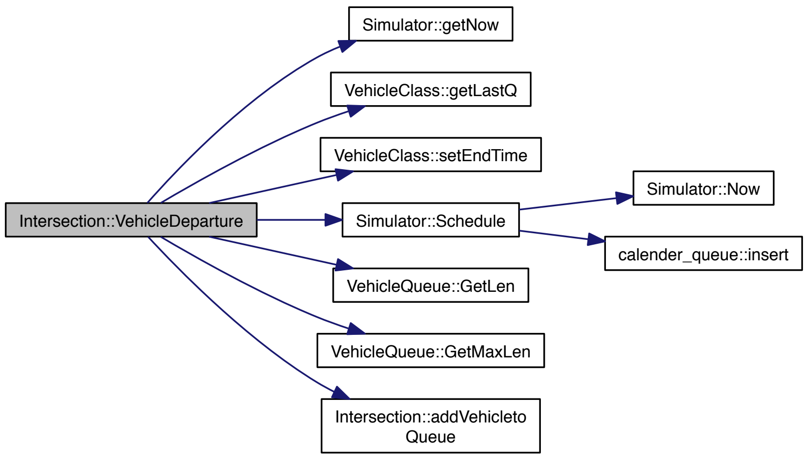
4.10.2.8 void Intersection::VehicleDeparture (VehicleClass ∗ *vehicle*)

Departs Vehicle from the intersection

Parameters

vehicle	is the vehicle to be departed

Here is the call graph for this function:



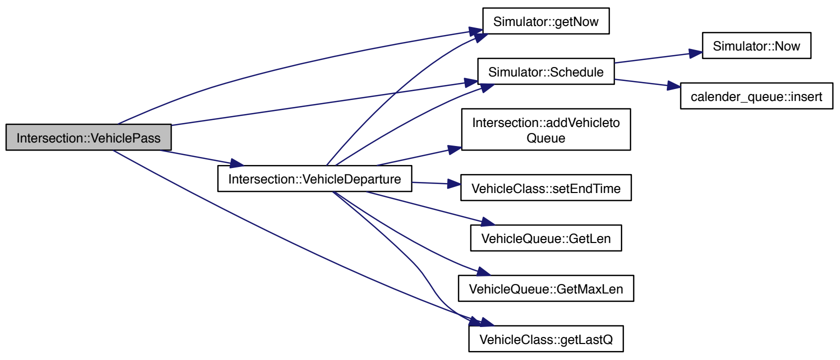
4.10.2.9 void Intersection::VehiclePass (**VehicleClass** ∗ *vehicle,* int *Turn*)

Logic of vehicle passing through this intersection

Parameters

Vehicle	
turn	

Here is the call graph for this function:



4.10.3 Member Data Documentation

4.10.3.1 bool Intersection::busy [protected]

Busy or not

4.10.3.2 VehicleQueue∗ Intersection::EBl1

Vehicle Queue East bound lane 1

4.10.3.3 VehicleQueue∗ Intersection::EBl2

Vehicle Queue East bound lane 2

4.10.3.4 VehicleQueue∗ Intersection::ExitQ

all vehicle exiting the system are queued into Exit queue for post processing

4.10.3.5 bool Intersection::haveSignal [protected]

Have traffic signal or not

4.10.3.6 int Intersection::ID [protected]

Intersection Id

4.10.3.7 VehicleQueue∗ Intersection::NBl1

Vehicle Queue North bound lane 1

4.10.3.8 VehicleQueue∗ Intersection::NBl2

Vehicle Queue North bound lane 2

4.10.3.9 Intersection∗ Intersection::NInter

Neighboring intersection in the North

4.10.3.10 dir Intersection::routingtable[12]

Acts as trnslator for routing cars

4.10.3.11 VehicleQueue∗ Intersection::SBl1

Vehicle Queue South bound lane 1

4.10.3.12 VehicleQueue∗ Intersection::SBl2

Vehicle Queue South bound lane 2

4.10.3.13 Intersection∗ Intersection::SInter

Neighboring intersection in the South

4.10.3.14 VehicleQueue∗ Intersection::WBl1

Vehicle Queue West bound lane 1

4.10.3.15 VehicleQueue∗ Intersection::WBl2

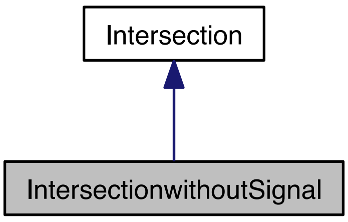
Vehicle Queue West bound lane 2

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

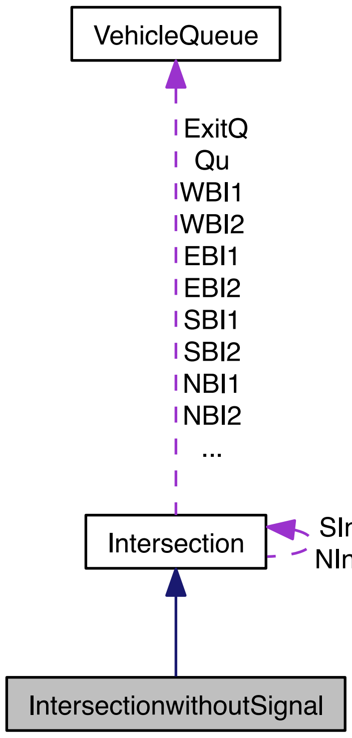
- Intersection.h
- Intersection.cpp

4.11 IntersectionwithoutSignal Class Reference

Inheritance diagram for IntersectionwithoutSignal:



Collaboration diagram for IntersectionwithoutSignal:



Public Member Functions

- virtual void addVehicletoQueue (VehicleQueue ∗joinqueue, VehicleClass ∗vehicle)
- virtual int QCanGo (int direction, int lane)
- IntersectionwithoutSignal ()
- IntersectionwithoutSignal (int)
- ∼IntersectionwithoutSignal ()

Additional Inherited Members

4.11.1 Constructor & Destructor Documentation

4.11.1.1 IntersectionwithoutSignal::IntersectionwithoutSignal ( )

Default constructor

4.11.1.2 IntersectionwithoutSignal::IntersectionwithoutSignal (int *nID*)

Constructor with setting ID

4.11.1.3 IntersectionwithoutSignal::∼IntersectionwithoutSignal (void)

Default Destructor

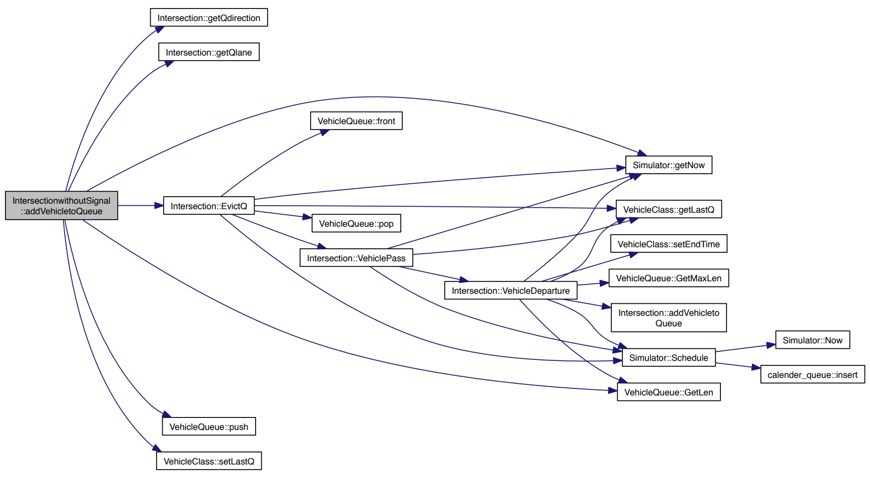
4.11.2 Member Function Documentation

4.11.2.1 void IntersectionwithoutSignal::addVehicletoQueue (**VehicleQueue** ∗ *joinqueue,* **VehicleClass** ∗ *vehicle*) `[virtual]`

Adds to outgoing queue or removes vehicles

Implements Intersection.

Here is the call graph for this function:

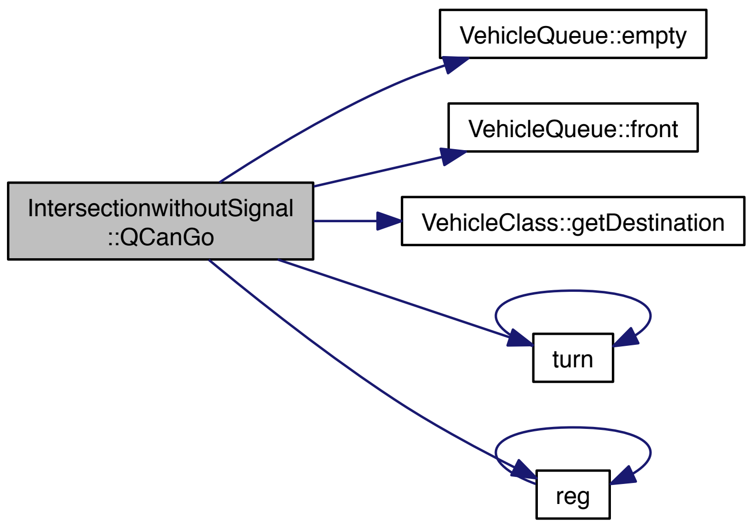


4.11.2.2 int IntersectionwithoutSignal::QCanGo (int *direction,* int *lane*) `[virtual]`

Figures if the Q(direction,lane) can starts moving

Implements Intersection.

Here is the call graph for this function:

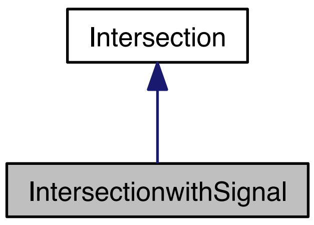


The documentation for this class was generated from the following files:

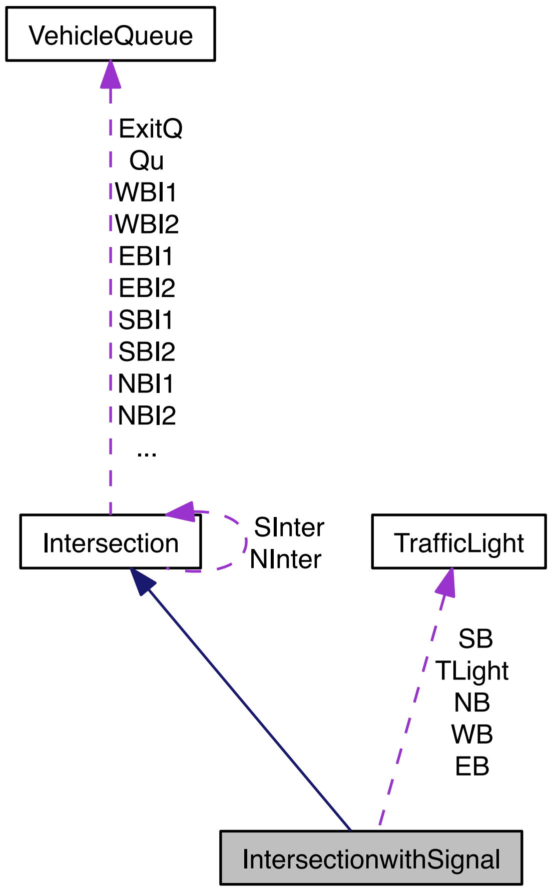
- IntersectionwoSignal.h
- IntersectionwoSignal.cpp

4.12 IntersectionwithSignal Class Reference

Inheritance diagram for IntersectionwithSignal:



Collaboration diagram for IntersectionwithSignal:



Public Member Functions

- void changeSignalTrigger (int LightID, int leftorthru)
- virtual void addVehicletoQueue (VehicleQueue ∗joinqueue, VehicleClass ∗vehicle)
- virtual int QCanGo (int direction, int lane)
- IntersectionwithSignal ()
- IntersectionwithSignal (int)
- ∼IntersectionwithSignal ()

Public Attributes

- TrafficLight ∗ EB
- TrafficLight ∗ WB
- TrafficLight ∗ NB
- TrafficLight ∗ SB
- TrafficLight ∗ **TLight** [4]

Additional Inherited Members

4.12.1 Constructor & Destructor Documentation

4.12.1.1 IntersectionwithSignal::IntersectionwithSignal ( )

Default Constructor

4.12.1.2 IntersectionwithSignal::IntersectionwithSignal (int *nID*)

COnstror sets the ID of the intersection

4.12.1.3 IntersectionwithSignal::∼IntersectionwithSignal (void)

Destructor

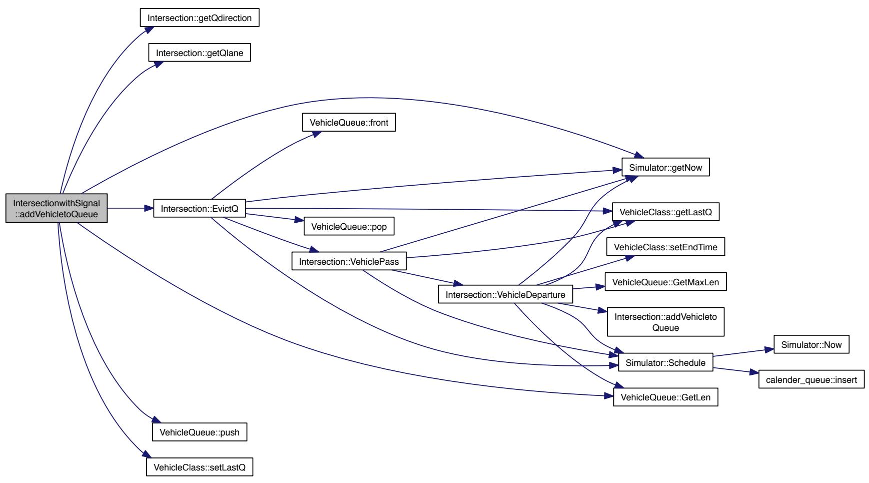
4.12.2 Member Function Documentation

4.12.2.1 void IntersectionwithSignal::addVehicletoQueue (**VehicleQueue** ∗ *joinqueue,* **VehicleClass** ∗ *vehicle*) [virtual]

Adds to outgoing queue or removes vehicles

Implements Intersection.

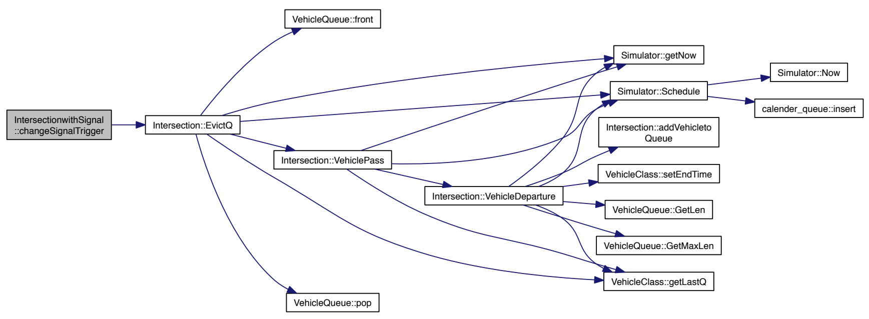
Here is the call graph for this function:



4.12.2.2 void IntersectionwithSignal::changeSignalTrigger (int *LightID,* int *leftorthru*)

checks its own signals leftortur=1: left

Here is the call graph for this function:

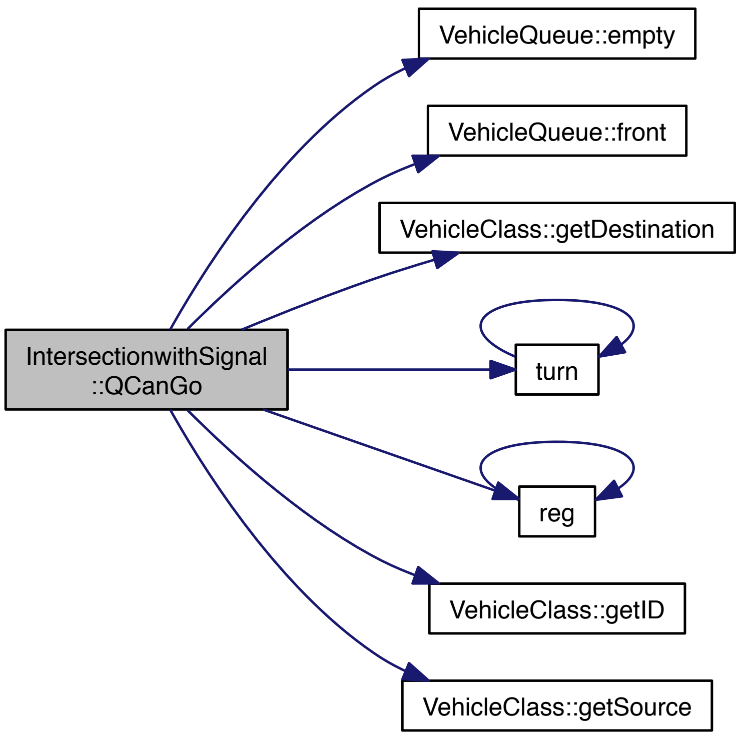


4.12.2.3 int IntersectionwithSignal::QCanGo (int *direction*, int *lane*) [virtual]

If the queue can go in "direction,lane"

Implements Intersection.

Here is the call graph for this function:



4.12.3 Member Data Documentation

4.12.3.1 TrafficLight∗ IntersectionwithSignal::EB

East bound Traffic lights

4.12.3.2 TrafficLight∗ IntersectionwithSignal::NB

North bound Traffic lights

4.12.3.3 TrafficLight∗ IntersectionwithSignal::SB

South bound Traffic lights

4.12.3.4 TrafficLight∗ IntersectionwithSignal::WB

West bound Traffic lights

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

- IntersectionwithSignal.h
- IntersectionwithSignal.cpp

4.13 node Class Reference

```
#include <calender_queue_testing.h>
```

Public Member Functions

- double getTime ()
- node (int v, double d)

4.13.1 Detailed Description

let's describe a small data structure for testing calender queue function

4.13.2 Constructor & Destructor Documentation

4.13.2.1 node::node (int *v*, double *d*) [inline]

constructor

4.13.3 Member Function Documentation

4.13.3.1 double node::getTime ( ) [inline]

Gets time of the function

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

- testing/calender_queue_testing.h

4.14 prioqueue Class Reference

Public Member Functions

- void **enqueue** (EventBase ∗)
- EventBase ∗ **dequeue** (EventBase ∗)
- EventBase ∗ **PopNext** ()
- bool **isEmpty** ()

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

- prioqueue.h

4.15 RandomNumGen Class Reference

Public Member Functions

- RandomNumGen ()
- RandomNumGen (unsigned long x0)
- double Next ()
- void Reset ()
- unsigned long GetState ()
- ∼RandomNumGen ()

4.15.1 Constructor & Destructor Documentation

4.15.1.1 RandomNumGen::RandomNumGen ( )

Constructor: Initializes the default parameters of Random number genrator

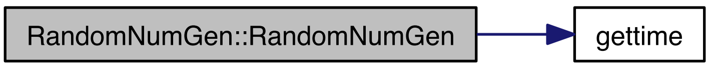
4.15.1.2 RandomNumGen::RandomNumGen (unsigned long *x0*)

Constuctor: Initializes the starting state with x0

Parameters

x0	is long input, if 0 takes starting point seed as time, otherwise sets x0 as the internal state

Here is the call graph for this function:



4.15.1.3 RandomNumGen::∼RandomNumGen ( )

Destructor for random number genrator

4.15.2 Member Function Documentation

4.15.2.1 unsigned long RandomNumGen::GetState ( )

Gives state of random genrator. Used for debugging purpose

4.15.2.2 double RandomNumGen::Next ( )

Genrates next random number

4.15.2.3 void RandomNumGen::Reset ( )

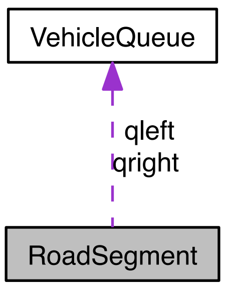
Resets the random number generator

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

- RandomNum.h
- RandomNum.cc

4.16 RoadSegment Class Reference

Collaboration diagram for RoadSegment:



Public Member Functions

- RoadSegment (dir direction, Intersection ∗par, int cap)
- void AddVehicle (VehicleClass ∗vehicle)
- void EvictVehicle ()

Public Attributes

- VehicleQueue qright
- VehicleQueue qleft

4.16.1 Constructor & Destructor Documentation

4.16.1.1 RoadSegment::RoadSegment (dir *direction*, **Intersection** ∗ *par*, int *cap*) [inline]

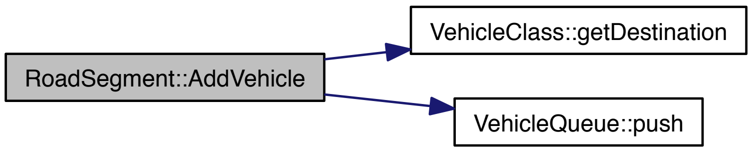
Constructor

4.16.2 Member Function Documentation

4.16.2.1 void RoadSegment::AddVehicle (**VehicleClass** ∗ *vehicle*)

Adds vehicle to the road segment

Here is the call graph for this function:



4.16.2.2 void RoadSegment::EvictVehicle ( )

Evicts vehicle from the Road Segment

4.16.3 Member Data Documentation

4.16.3.1 VehicleQueue RoadSegment::qleft

Left lane (Vehicle Queue)

4.16.3.2 VehicleQueue RoadSegment::qright

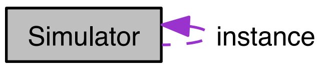
Right lane (Vehicle queue)

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

- RoadSegment.h
- RoadSegment.cpp

4.17 Simulator Class Reference

Collaboration diagram for Simulator:



Public Member Functions

- Simulator ()
- void Stop ()
- Time_t getNow ()
- template<typename T , typename OBJ , typename U1 , typename T1 >
  void Schedule (double t, void(T::∗handler)(U1), OBJ ∗obj, T1 t1)
- template<typename T , typename OBJ , typename U1 , typename T1 , typename U2 , typename T2 >
  void Schedule (double t, void(T::∗handler)(U1, U2), OBJ ∗obj, T1 t1, T2 t2)
- template<typename T , typename OBJ , typename U1 , typename T1 , typename U2 , typename T2 , typename U3 , typename T3 >
  void Schedule (double t, void(T::∗handler)(U1, U2, U3), OBJ ∗obj, T1 t1, T2 t2, T3 t3)

Static Public Member Functions

- static void Run ()
- static void StopAt (Time_t)
- template<typename T , typename OBJ >
  static void Schedule (double t, void(T::∗handler)(void), OBJ ∗obj)
- static Time_t Now ()

Static Public Attributes

- static Simulator ∗ instance =0

4.17.1 Constructor & Destructor Documentation

4.17.1.1 Simulator::Simulator ( )

Default constructor

4.17.2 Member Function Documentation

4.17.2.1 Time_t Simulator::getNow ( ) [inline]

Returns the current time of the simulation

4.17.2.2 Time_t Simulator::Now ( ) [static]

Returns the time NOW

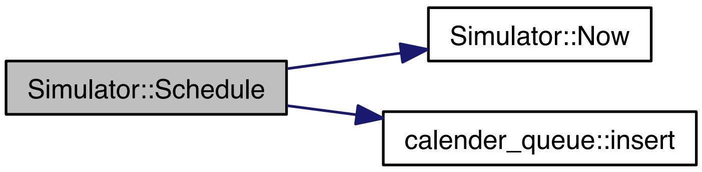
4.17.2.3 void Simulator::Run ( ) [static]

Start executing events

4.17.2.4 template<typename T , typename OBJ > static void Simulator::Schedule (double *t*, void(T::∗)(void) *handler*, OBJ ∗ *obj*) [inline], [static]

Schedules the events type 0

Here is the call graph for this function:



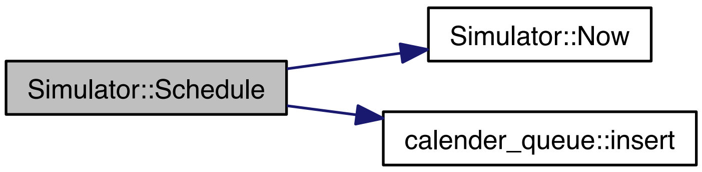
4.17.2.5 template<typename T , typename OBJ , typename U1 , typename T1 > void Simulator::Schedule (double *t*, void(T::∗)(U1) *handler*, OBJ ∗ *obj*, T1 *t1*) [inline]

Schedules the event type1

See Also

Events.h

Here is the call graph for this function:



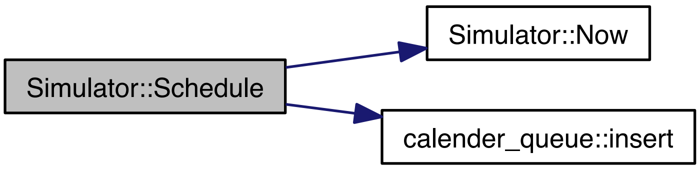
4.17.2.6 template<typename T , typename OBJ , typename U1 , typename T1 , typename U2 , typename T2 > void Simulator::Schedule (double *t*, void(T::∗)(U1, U2) *handler*, OBJ ∗ *obj*, T1 *t1*, T2 *t2*) [inline]

Schedules the event type2

See Also

Events.h

Here is the call graph for this function:



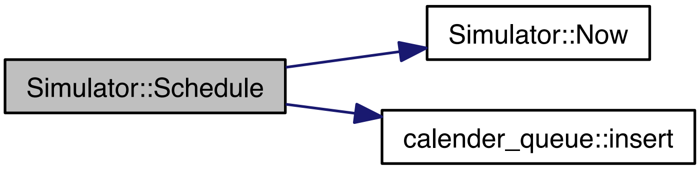
4.17.2.7 template<typename T , typename OBJ , typename U1 , typename T1 , typename U2 , typename T2 , typename U3 , typename T3 > void Simulator::Schedule (double *t*, void(T::∗)(U1, U2, U3) *handler*, OBJ ∗ *obj*, T1 *t1*, T2 *t2*, T3 *t3*) [inline]

Schedules the event type3

See Also

Events.h

Here is the call graph for this function:



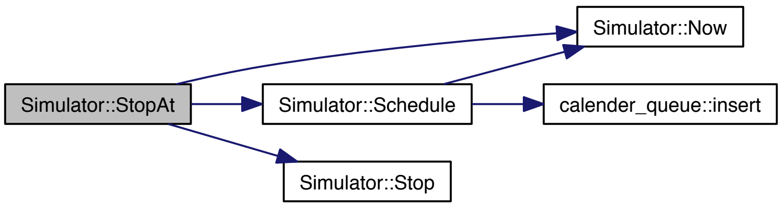
4.17.2.8 void Simulator::Stop ( )

Stops executing events

4.17.2.9 void Simulator::StopAt (Time_t *t*) [static]

Defines stopping time

Here is the call graph for this function:



4.17.3 Member Data Documentation

4.17.3.1 Simulator ∗ Simulator::instance =0 [static]

Pointer to simulator

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

- Simulator.h
- Simulator.cpp

4.18 TrafficLight Class Reference

Public Member Functions

- int getType ()
- state getState ()
- state getLeftState ()
- TrafficLight ()
- TrafficLight (int id, int typ, state initialState, state initialstate2, double Ph1, double Ph2, double Ph3, double Ph4, double Ph5, double Ph6, IntersectionwithSignal ∗p, Time_t timetoStart, Time_t timetoStart2)
- ∼TrafficLight ()
- void cyclestate (int leftorthru)

Public Attributes

- int type
- int myid

4.18.1 Constructor & Destructor Documentation

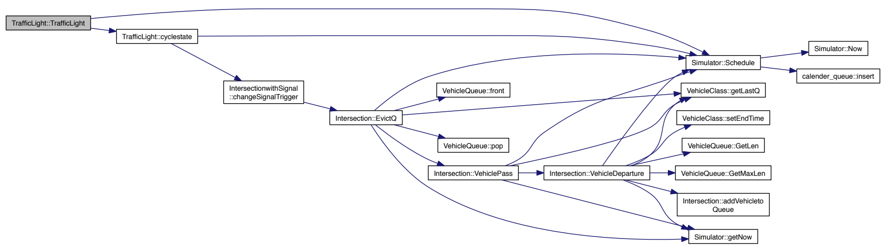
4.18.1.1 TrafficLight::TrafficLight ( )

Default constructor

4.18.1.2 TrafficLight::TrafficLight (int *id,* int *typ,* state *initialState,* state *initialstate2,* double *Ph1,* double *Ph2,* double *Ph3,* double *Ph4,* double *Ph5,* double *Ph6,* IntersectionwithSignal $*$ *p,* Time_t *timetoStart,* Time_t *timetoStart2*)

Constructor with initial states and everything (type, initialState GLT, YLT, RLT, GTR, YTR, RTR) put zeros if any was inapplicable

Here is the call graph for this function:



4.18.1.3 TrafficLight::∼TrafficLight ( )

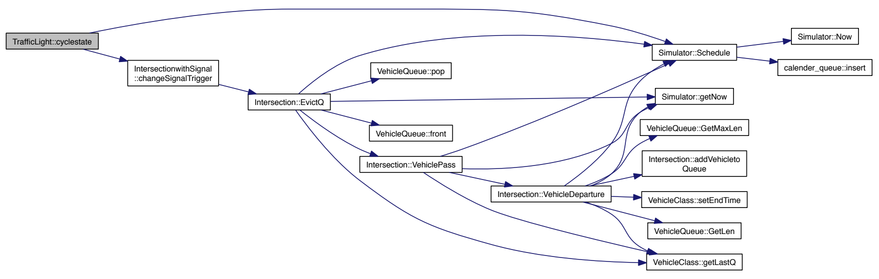
Default destructor

4.18.2 Member Function Documentation

4.18.2.1 void TrafficLight::cyclestate (int *leftorthru*)

cyclestate

Here is the call graph for this function:



4.18.2.2 state TrafficLight::getLeftState ( ) [inline]

Returns the present left state of the traffic light

4.18.2.3 state TrafficLight::getState ( ) [inline]

Returns the present state of the traffic light

4.18.2.4 int TrafficLight::getType ( ) [inline]

Returns the type of traffic light

4.18.3 Member Data Documentation

4.18.3.1 int TrafficLight::myid

Id of the traffic signal

4.18.3.2 int TrafficLight::type

0 if 3 states and 1 if 6 states and 2 if there are two independent signals

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

- TrafficLight.h
- TrafficLight.cpp

4.19 VehicleClass Class Reference

Public Member Functions

- void setEndTime (Time_t t)
- int getID ()
- void updateDirection (dir Direction)

  !Constructor
- dir getDirection ()
- void setLastQ (VehicleQueue ∗Q)
- VehicleQueue ∗ getLastQ ()
- Time_t StartTime ()
- Time_t EndTime ()
- int getDestination ()
- int getSource ()
- VehicleClass (int id, int start, int Dest, Time_t starttime)
- ∼VehicleClass ()

Public Attributes

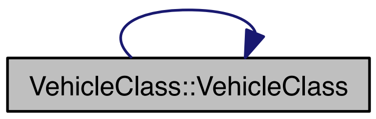
- Time_t startTime
- Time_t endTime
- std::list< eventDsc > **EventList**

4.19.1 Constructor & Destructor Documentation

4.19.1.1 VehicleClass::VehicleClass (int *id,* int *start,* int *Dest,* Time_t *starttime*)

Default constructor

Here is the call graph for this function:



4.19.1.2 VehicleClass::∼VehicleClass ( )

Default destructor

4.19.2 Member Function Documentation

4.19.2.1 Time_t VehicleClass::EndTime ( ) [inline]

Returns endtime of the car

4.19.2.2 int VehicleClass::getDestination ( )

Returns destination of the car

4.19.2.3 dir VehicleClass::getDirection ( )

outputs the direction of the car

4.19.2.4 int VehicleClass::getID ( )

Gets the ID of the car

4.19.2.5 VehicleQueue ∗ VehicleClass::getLastQ ( )

outputs the last queue

4.19.2.6 int VehicleClass::getSource ( )

Returns the source of the car

4.19.2.7 void VehicleClass::setEndTime (Time_t *t*)

Sets the end time

4.19.2.8 void VehicleClass::setLastQ (VehicleQueue ∗ *Q*)

Sets the Vehicle queue to which the car eblonged

4.19.2.9 Time_t VehicleClass::StartTime ( ) [inline]

Returns the start time

4.19.2.10 void VehicleClass::updateDirection (dir *Direction*)

!Constructor

Updated the direction of the car

4.19.3 Member Data Documentation

4.19.3.1 Time_t VehicleClass::endTime

Time when a car exits out of the system

4.19.3.2 Time_t VehicleClass::startTime

Time when a car comes into existance

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

- VehicleClass.h
- VehicleClass.cpp

4.20 VehicleQueue Class Reference

Public Member Functions

- VehicleQueue ()
- VehicleQueue (VehicleQueue ∗Q)
- VehicleQueue (int maxL)
- VehicleClass ∗ front ()
- bool empty ()
- void push (VehicleClass ∗V1)
- void pop ()
- VehicleClass ∗ back ()
- int GetMaxLen ()
- int GetLen ()
- bool isBusy ()

Public Attributes

- int maxLength
- std::queue< VehicleClass ∗ > Q1
- int busy
- double LastSentCar

4.20.1 Constructor & Destructor Documentation

4.20.1.1 VehicleQueue::VehicleQueue ( )

Default constuctor

4.20.1.2 VehicleQueue::VehicleQueue (VehicleQueue ∗ *Q*)

Constuctor with Pointer argument

4.20.1.3 VehicleQueue::VehicleQueue (int *maxL*)

Constuctor with int argument assigning maxLength

4.20.2 Member Function Documentation

4.20.2.1 VehicleClass ∗ VehicleQueue::back ( )

Returns pointer to the vehicle that is at the end of the queue

4.20.2.2 bool VehicleQueue::empty ( )

Returns "true" if the queue is empty

4.20.2.3 VehicleClass ∗ VehicleQueue::front ( )

Returns pointer of the vehicle which is front of the queue

4.20.2.4 int VehicleQueue::GetLen ( )

Returns lenght of the queue (i.e. how many vehicles are there in the queue)

4.20.2.5 int VehicleQueue::GetMaxLen ( )

Returns Maximum possible lenght of the queue

4.20.2.6 bool VehicleQueue::isBusy ( )

Returns if the queue is busy or not

4.20.2.7 void VehicleQueue::pop ( )

Returns (and removes) vehicle that was latest existing vehicle in the queue

4.20.2.8 void VehicleQueue::push (VehicleClass ∗ *V1*)

Adds vehicle to the back of the queue

Parameters

V1	is pointer of the vehicle to be pushed into the queue

4.20.3 Member Data Documentation

4.20.3.1 int VehicleQueue::busy

to check if the queue is busy/not

4.20.3.2 double VehicleQueue::LastSentCar

Holds time for vehicle that was sent last

4.20.3.3 int VehicleQueue::maxLength

To hold the maximum length of the Queue

4.20.3.4 std::queue<VehicleClass* > VehicleQueue::Q1

std::Queue for holding the vehicle queue

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

- VehicleQueue.h
- VehicleQueue.cpp

Chapter 5

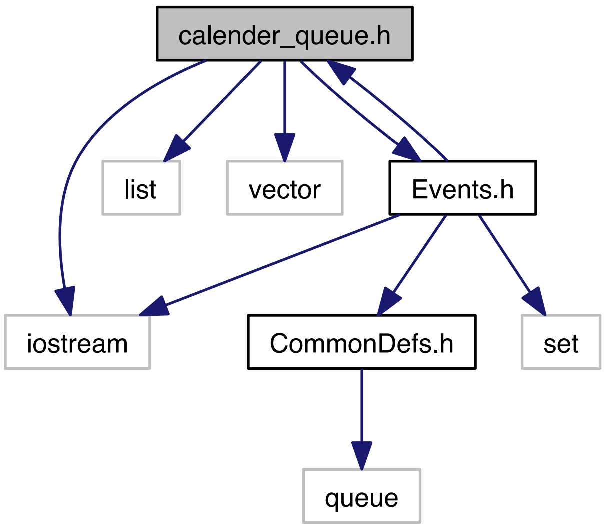
File Documentation

5.1 calender_queue.h File Reference

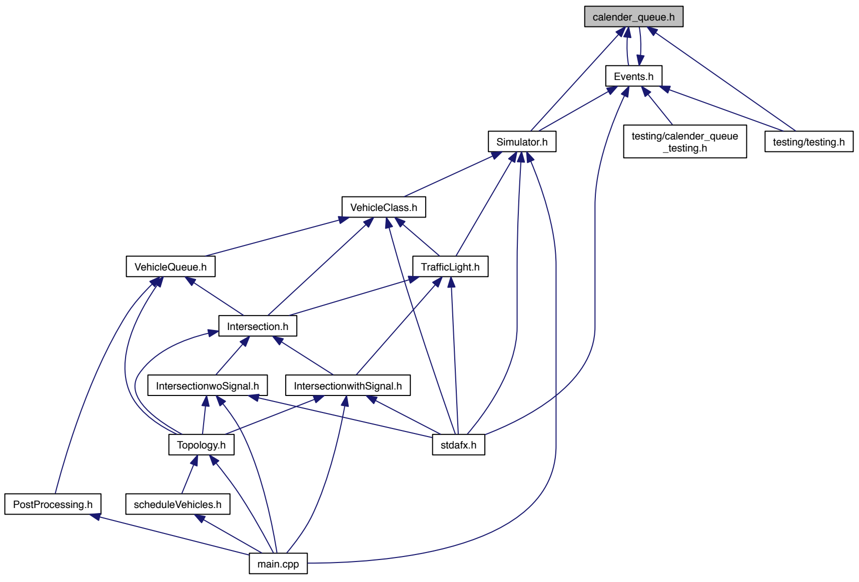
declartion of the class calender queue

```
#include <iostream>
#include <list>
#include <vector>
#include "Events.h"
```

Include dependency graph for calender_queue.h:



This graph shows which files directly or indirectly include this file:



Classes

- class calender_queue

Macros

- #define TOTAL_TIME 120∗60
- #define BUCKET_COUNT 72000
- #define **BUCKET_SIZE** 0.1
- #define CALENDER_PERIOD BUCKET_COUNT∗BUCKET_SIZE

Typedefs

- typedef std::list< EventBase ∗ > **bucket**

5.1.1 Detailed Description

declartion of the class calender queue

5.1.2 Macro Definition Documentation

5.1.2.1 #define BUCKET_COUNT 72000

Number of Buckets for Calender Queue

5.1.2.2 #define CALENDER_PERIOD BUCKET_COUNT∗BUCKET_SIZE

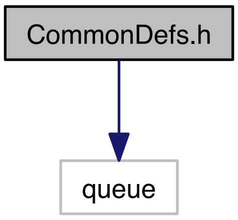
how much is a "year" for this calender

5.1.2.3 #define TOTAL_TIME 120∗60

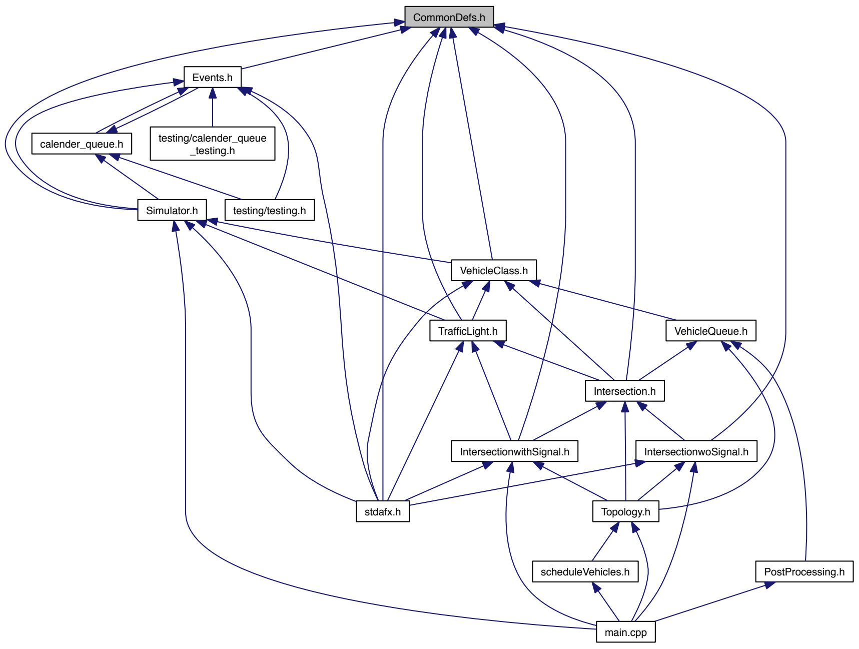
Total time of the simulation

5.2 CommonDefs.h File Reference

`#include <queue>`
Include dependency graph for CommonDefs.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **__COMMON_DEFS_H__**
- #define PassTime 5.0

- #define startToPass 2.0
- #define LPassTime 3.0
- #define roadSegTime 36.0
- #define checkQinterval 2.0
- #define BurstTime 2.0

Typedefs

- typedef double Time_t

Enumerations

- enum **state** {
  GLT, **YLT**, **RLT**, **GTR**,
  YTR, **RTR** }
- enum **dir** { **N**, **S**, **E**, **W** }

Functions

- int reg (int i)
- int turn (dir globalDir, int QDirection)

5.2.1 Detailed Description

Contains commmon definations of various parameters used in different functions

5.2.2 Macro Definition Documentation

5.2.2.1 #define BurstTime 2.0

time for the next vehicle to depart when cars are going in groups

5.2.2.2 #define checkQinterval 2.0

if the next Q is full, check again in this amout of time

5.2.2.3 #define LPassTime 3.0

service time to turn left in seconds (debug)

5.2.2.4 #define PassTime 5.0

service time to go straight in seconds

5.2.2.5 #define roadSegTime 36.0

time to travel one road segment

5.2.2.6 #define startToPass 2.0

when a queue is empty and a vehicle arrives, it takes this much to depart

5.2.3 Typedef Documentation

5.2.3.1 typedef double Time_t

Type for storing simulation times

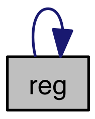
5.2.4 Function Documentation

5.2.4.1 int reg (int *i*)

See Also

intersection.cpp

Here is the call graph for this function:



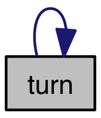
5.2.4.2 int turn (dir *globalDir,* int *QDirection*)

Returns routing address for Vehicle

See Also

intersection.cpp

Here is the call graph for this function:

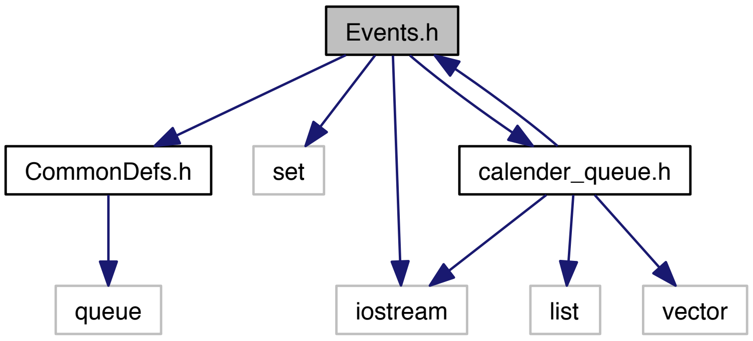


5.3 Events.h File Reference

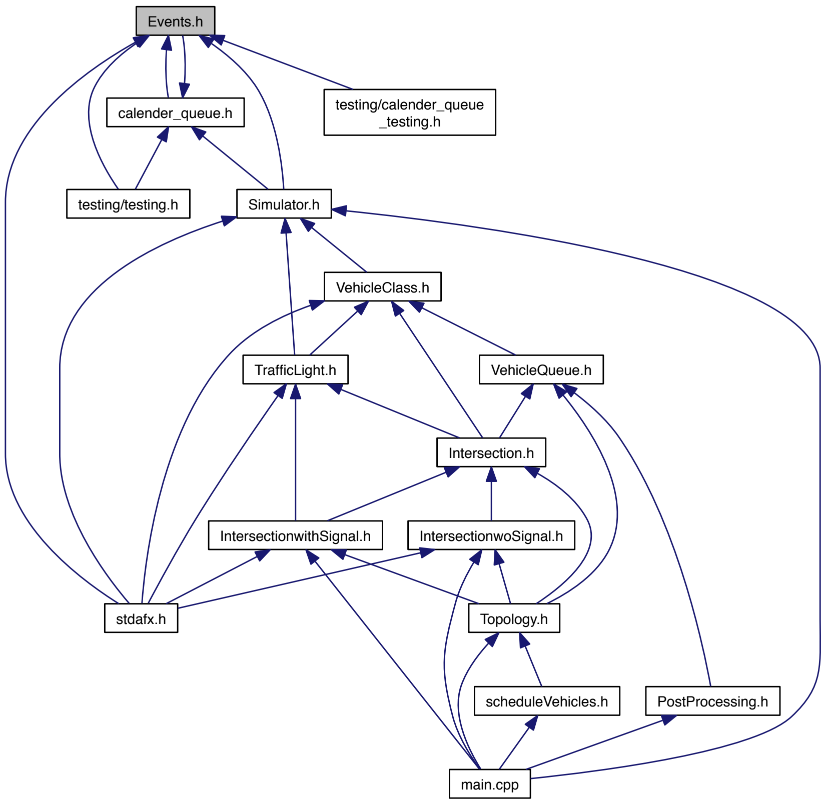
declaration of various types of events

```
#include "CommonDefs.h"
#include <set>
#include <iostream>
#include "calender_queue.h"
```

Include dependency graph for Events.h:



This graph shows which files directly or indirectly include this file:



Classes

- class EventBase
- class Event0< T, OBJ >
- class Event1< T, OBJ, U1, T1 >
- class Event2< T, OBJ, U1, T1, U2, T2 >
- class Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >
- class event_compare

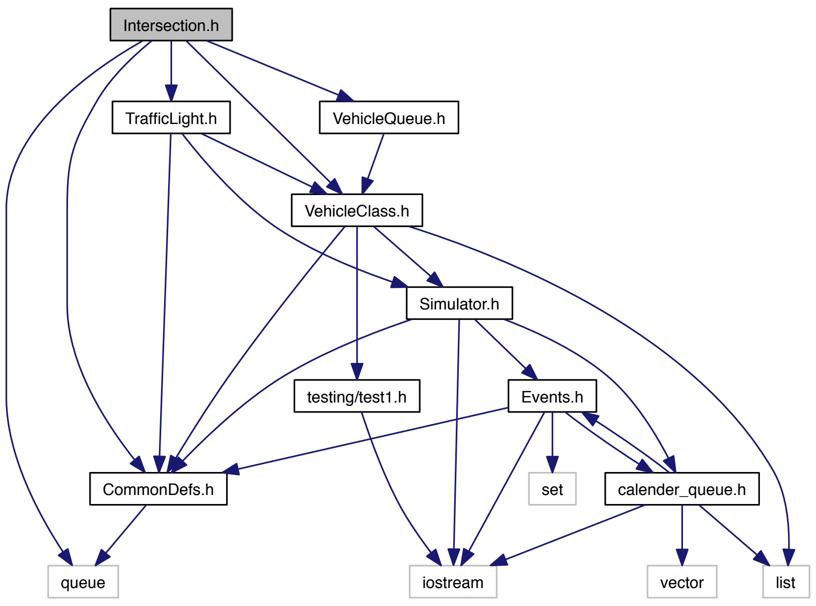
5.3.1 Detailed Description

declaration of various types of events

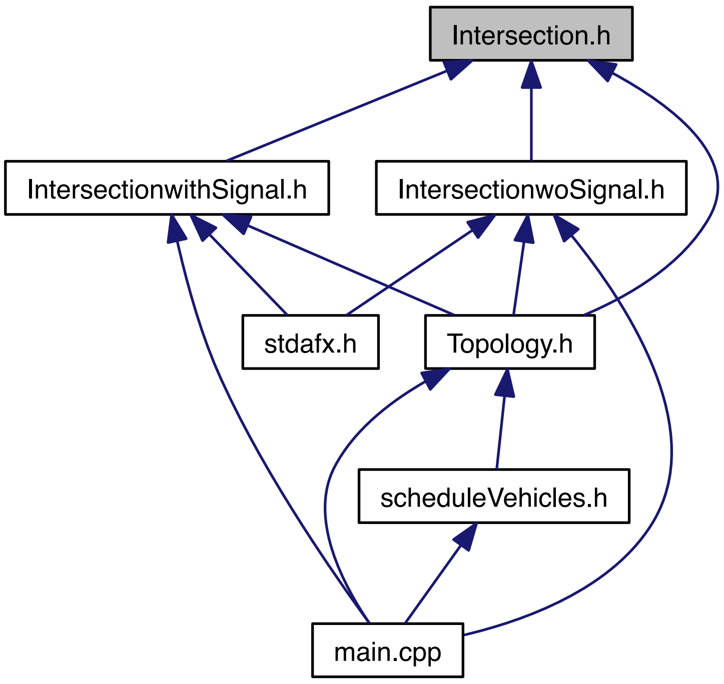
5.4 Intersection.h File Reference

```
#include <queue>
#include "CommonDefs.h"
#include "TrafficLight.h"
#include "VehicleClass.h"
#include "VehicleQueue.h"
```

Include dependency graph for Intersection.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Intersection

5.4.1 Detailed Description

Contains base class intersection from which both intersectionwithsignal and intersectionwosignal inherit

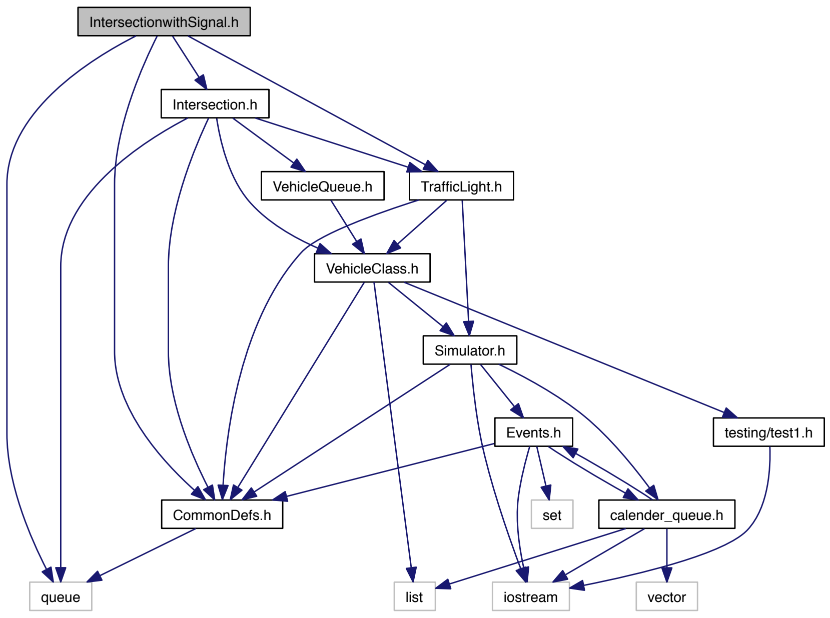
See Also

IntersectionwithSignal.h
IntersectionwoSignal.h

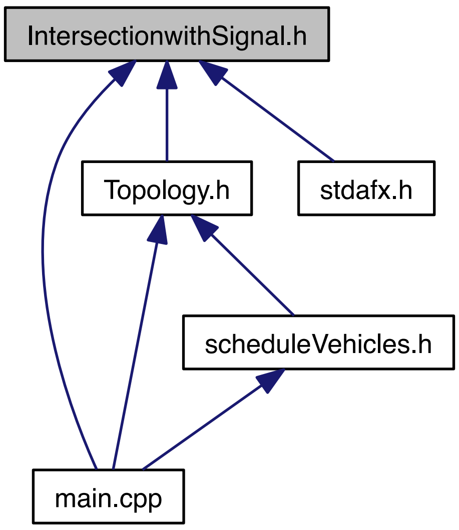
5.5 IntersectionwithSignal.h File Reference

```
#include <queue>
#include "CommonDefs.h"
#include "TrafficLight.h"
#include "Intersection.h"
```

Include dependency graph for IntersectionwithSignal.h:



This graph shows which files directly or indirectly include this file:



Classes

- class IntersectionwithSignal

5.5.1 Detailed Description

Description of Intersection with traffic signals class

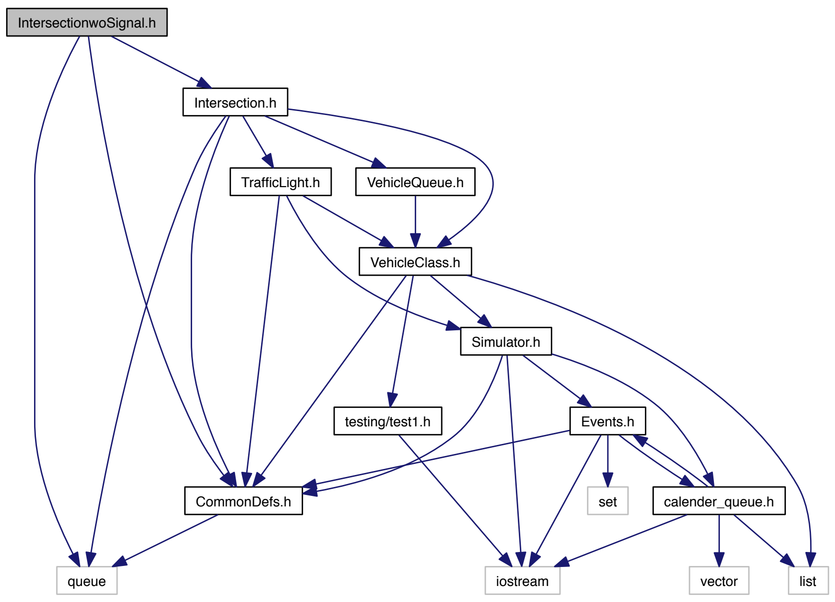
See Also

Intersection.h

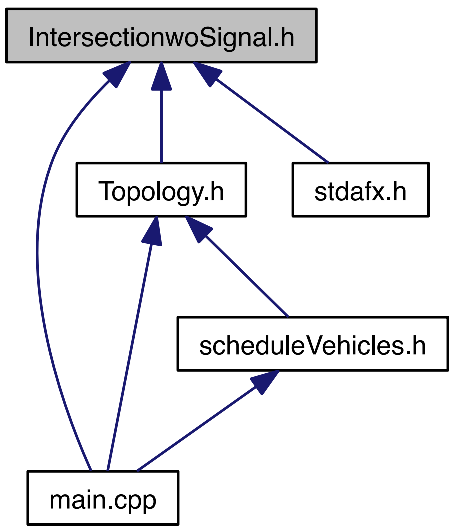
5.6 IntersectionwoSignal.h File Reference

```
#include <queue>
#include "CommonDefs.h"
#include "Intersection.h"
```

Include dependency graph for IntersectionwoSignal.h:



This graph shows which files directly or indirectly include this file:



Classes

- class IntersectionwithoutSignal

5.6.1 Detailed Description

Description of Intersection with out traffic signals class

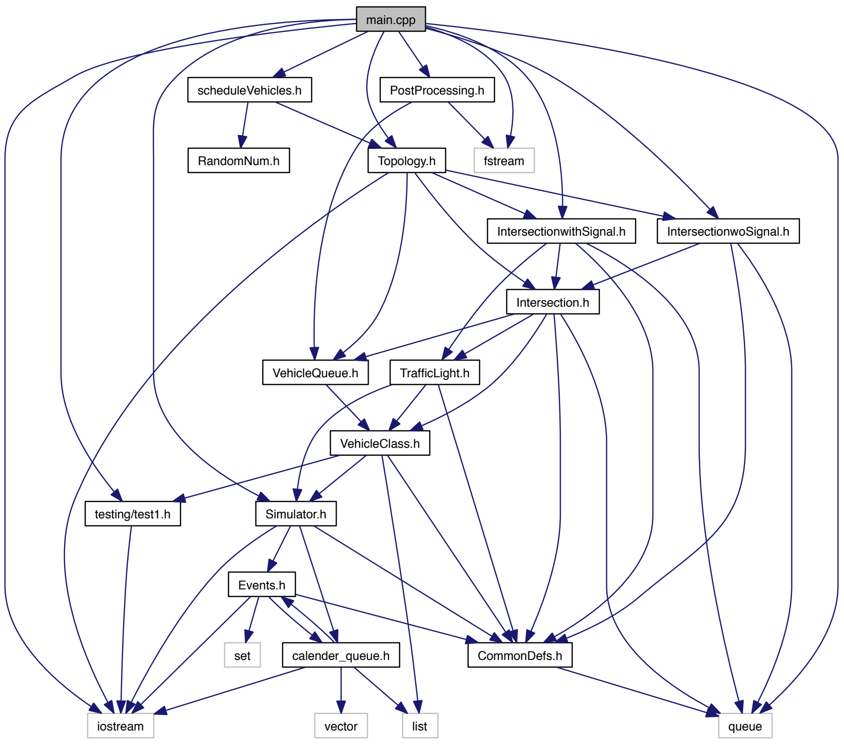
See Also

Intersection.h

5.7 main.cpp File Reference

```
#include <iostream>
#include <fstream>
#include "Simulator.h"
#include "IntersectionwithSignal.h"
#include "IntersectionwoSignal.h"
#include "Topology.h"
#include "scheduleVehicles.h"
#include "PostProcessing.h"
#include "testing/test1.h"
#include <queue>
```

Include dependency graph for main.cpp:



Functions

- int **main** ()

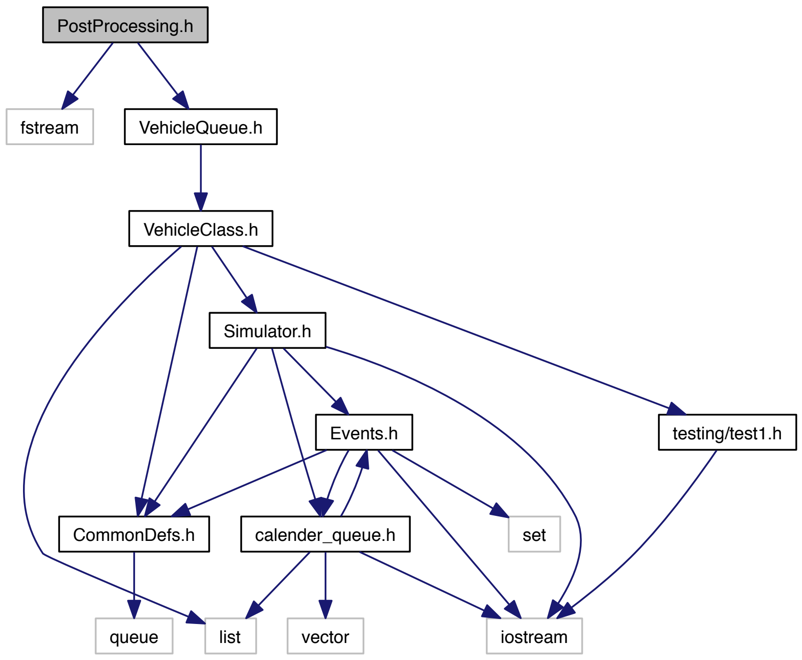
Variables

- Simulator ∗ **sim** = new Simulator()

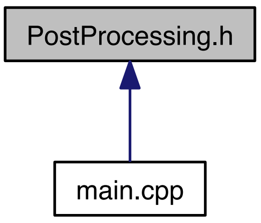
5.8 PostProcessing.h File Reference

```
#include <fstream>
#include "VehicleQueue.h"
```

Include dependency graph for PostProcessing.h:



This graph shows which files directly or indirectly include this file:



Functions

- void PostProcStats (VehicleQueue ∗ExQ, double timeval, int buckets, int source, int dest, ofstream &fh)

5.8.1 Detailed Description

Takes Exit Queue As Argument and print Various following things

1. Histogram of simulation time

5.8.2 Function Documentation

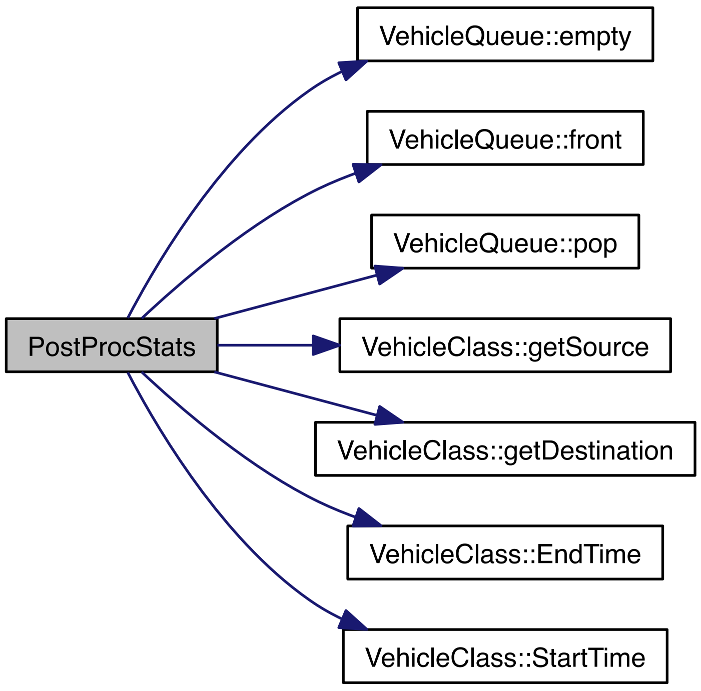
5.8.2.1 void PostProcStats (**VehicleQueue** ∗ *ExQ*, double *timeval*, int *buckets*, int *source*, int *dest*, ofstream & *fh*)

Takes exit queue and prints histogram of time takes to cover between source and destination Also prints stats like, average time , standard deviation etc.

Parameters

EQ	is exit Q
buckets	is number of buckets for histogram
timeval	is the period of time which is divided into buckets
source	is starting point of the journey
dest	is input for describing

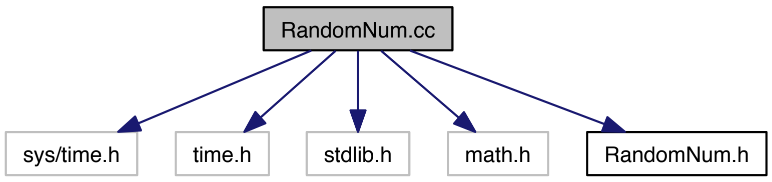
Here is the call graph for this function:



5.9 RandomNum.cc File Reference

```
#include <sys/time.h>
#include <time.h>
#include <stdlib.h>
#include <math.h>
#include "RandomNum.h"
```

Include dependency graph for RandomNum.cc:



Functions

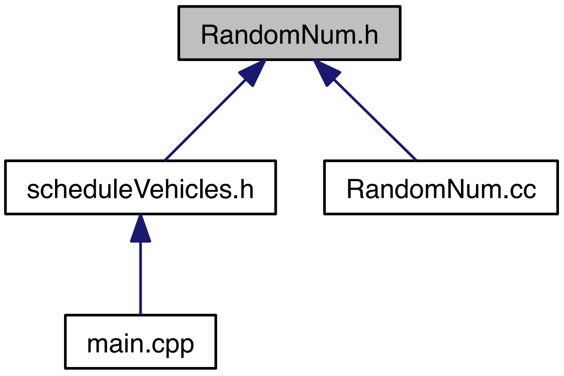
- unsigned long **gettime** (void)

5.9.1 Detailed Description

contains defination of randomnumber generator class And brief testing of the random numbers

5.10 RandomNum.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class RandomNumGen

Macros

- #define MINSTDX 1

- #define MINSTDM 2147483647
- #define MINSTDG 16807

Functions

- unsigned long gettime ()

5.10.1 Detailed Description

A Random number generator class. This class describes the implementation number genrator

5.10.2 Macro Definition Documentation

5.10.2.1 #define MINSTDG 16807

Multiplier for random number genrator

5.10.2.2 #define MINSTDM 2147483647

Modulus for random number genrator

5.10.2.3 #define MINSTDX 1

Default starting state

5.10.3 Function Documentation

5.10.3.1 unsigned long gettime (void)

Time function to measure time

5.11 RoadSegment.h File Reference

Classes

- class RoadSegment

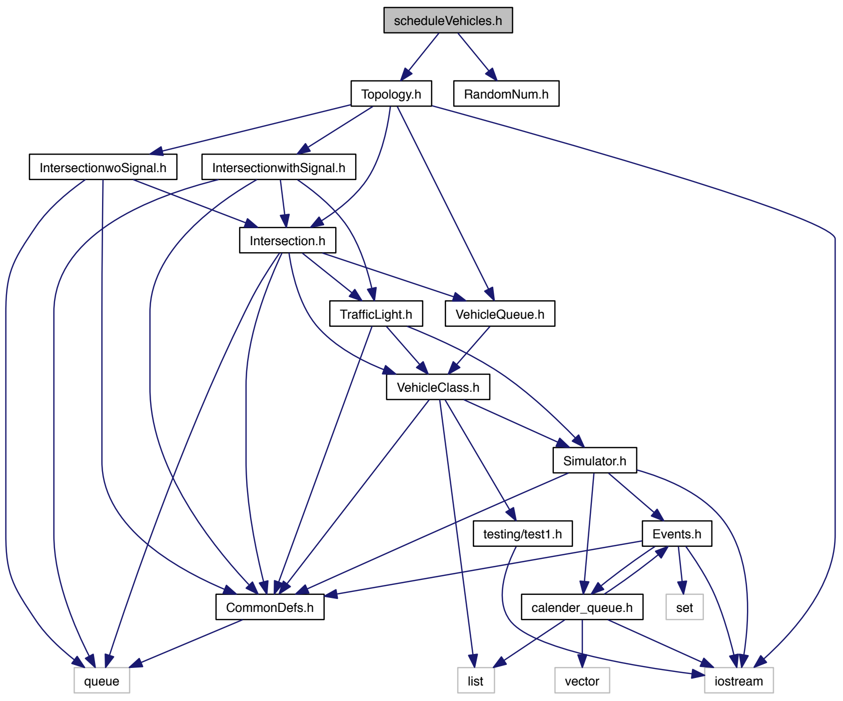
5.11.1 Detailed Description

Define a segment of the Road

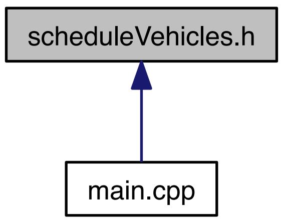
5.12 scheduleVehicles.h File Reference

```
#include "Topology.h"
#include "RandomNum.h"
```

Include dependency graph for scheduleVehicles.h:



This graph shows which files directly or indirectly include this file:



Functions

- void scheduleVehicles (_Topology *Topology, double maxTime)

5.12.1 Detailed Description

It initializes the scheduling of vehicle during the simulation

5.12.2 Function Documentation

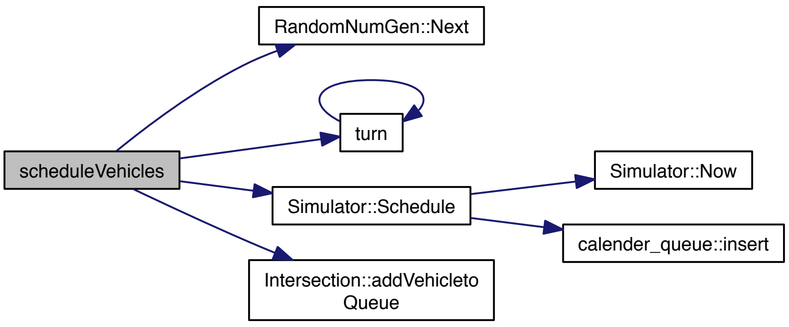
5.12.2.1 void scheduleVehicles (_Topology ∗ *Topology*, double *maxTime*)

It initializes the scheduling of vehicle during the simulation

Parameters

Topology	of the westpeachtree street
max-Time,maximum	time of till which we have to schedule vehicles

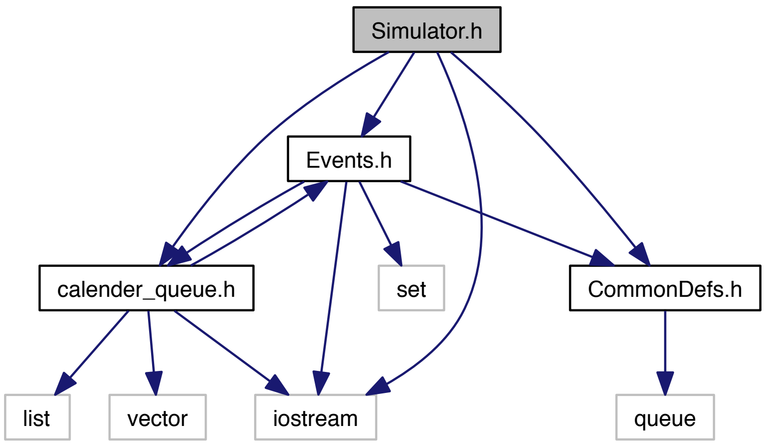
Here is the call graph for this function:



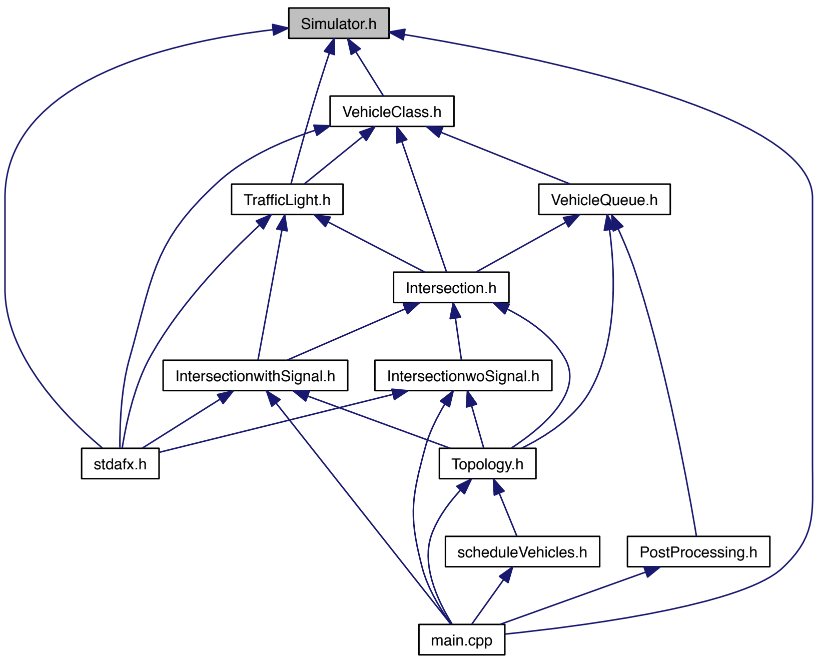
5.13 Simulator.h File Reference

```
#include <iostream>
#include "CommonDefs.h"
#include "Events.h"
#include "calender_queue.h"
```

Include dependency graph for Simulator.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Simulator

Typedefs

- typedef calender_queue **EventSet_t**

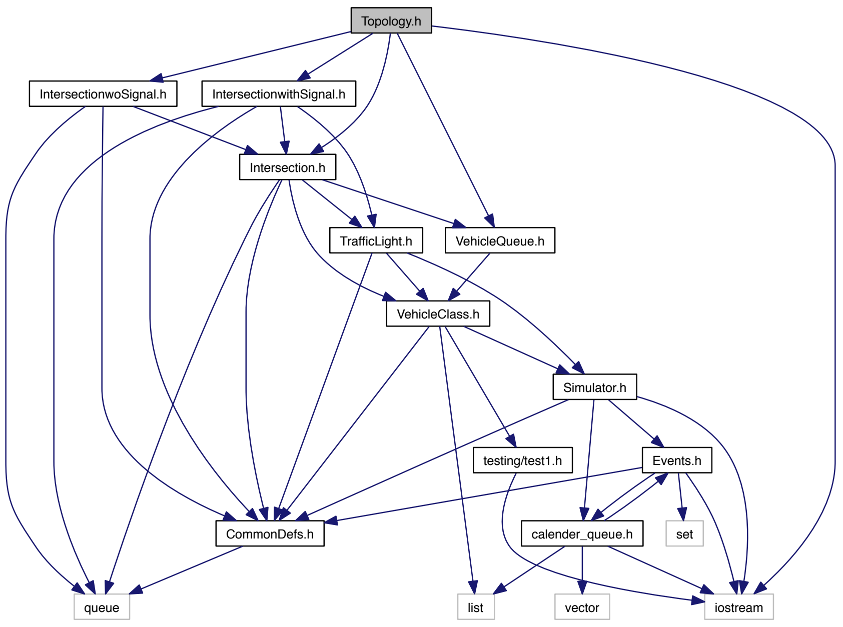
5.13.1 Detailed Description

Contains description of Simulator class and various functions of simulator class

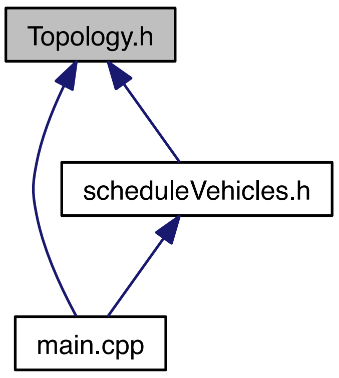
5.14 Topology.h File Reference

```
#include <iostream>
#include "Intersection.h"
#include "IntersectionwithSignal.h"
#include "IntersectionwoSignal.h"
#include "VehicleQueue.h"
```

Include dependency graph for Topology.h:



This graph shows which files directly or indirectly include this file:



Classes

- class _Topology

5.14.1 Detailed Description

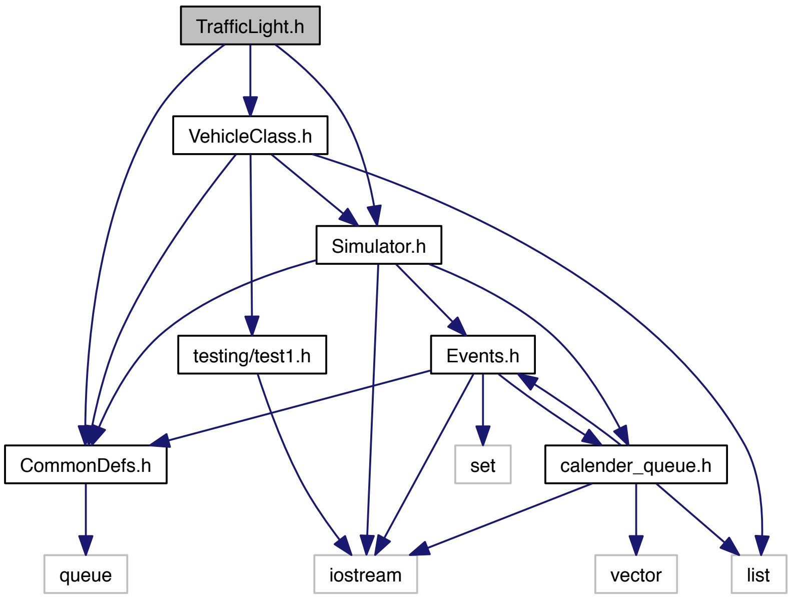
To describe the topology of the street to be simulated i.e. peachtree street for this project

5.15 TrafficLight.h File Reference

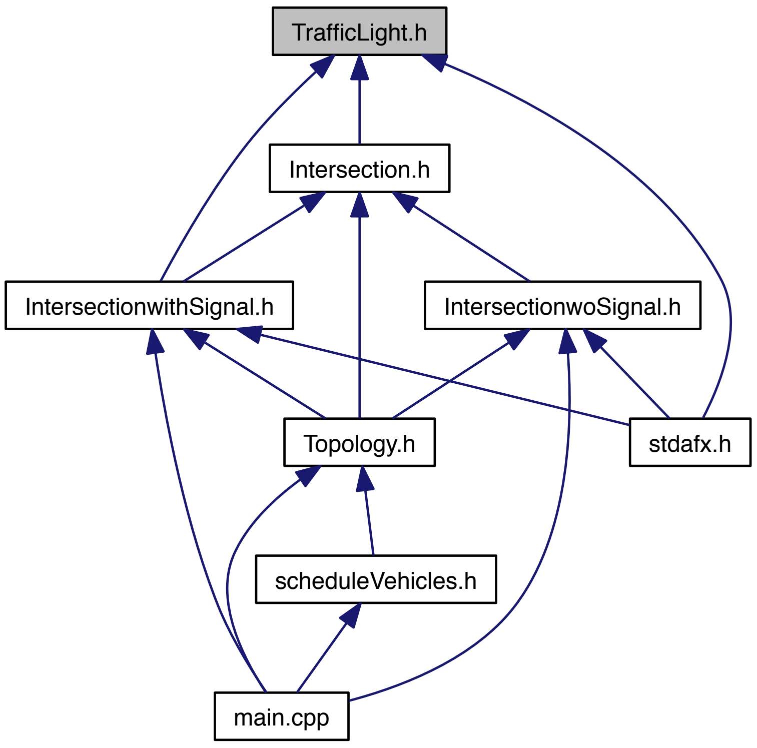
description of functionality of traffic light

```
#include "CommonDefs.h"
#include "VehicleClass.h"
#include "Simulator.h"
```

Include dependency graph for TrafficLight.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TrafficLight

Variables

- Simulator ∗ **sim**

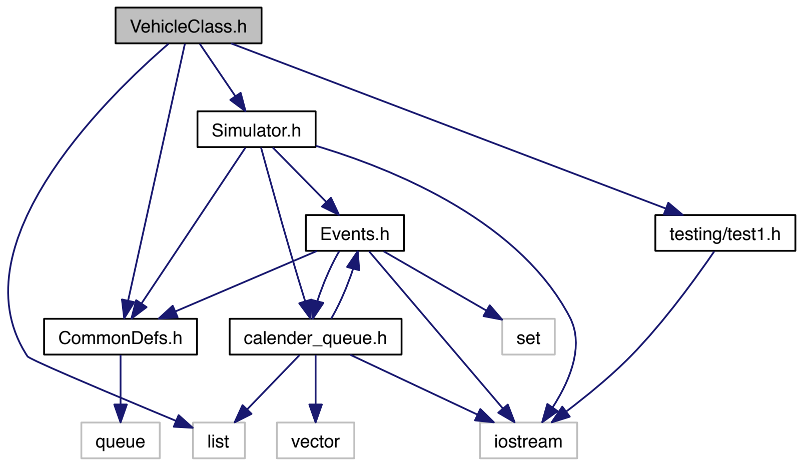
5.15.1 Detailed Description

description of functionality of traffic light

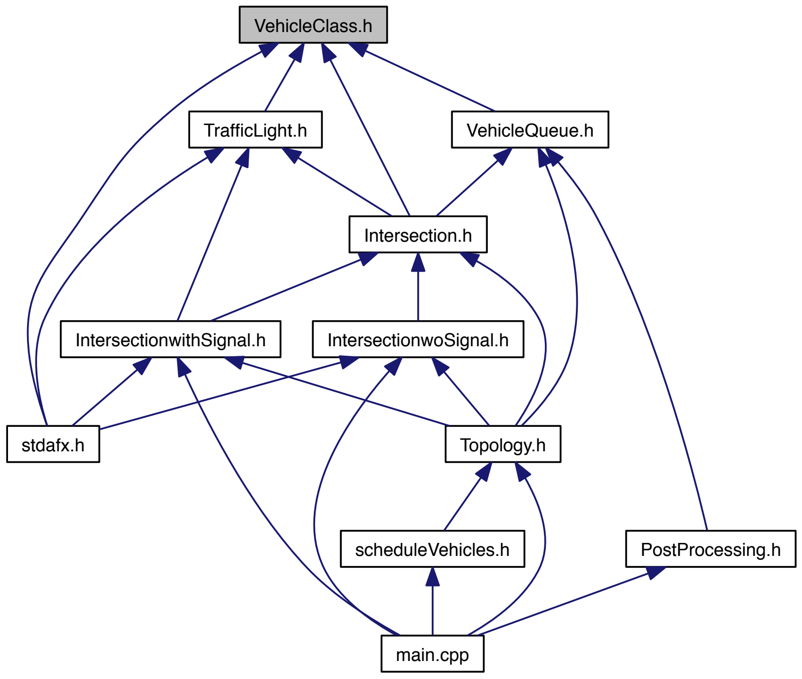
5.16 VehicleClass.h File Reference

```
#include "CommonDefs.h"
#include "testing/test1.h"
#include <list>
#include "Simulator.h"
```

Include dependency graph for VehicleClass.h:



This graph shows which files directly or indirectly include this file:



Classes

- class VehicleClass

Variables

- Simulator ∗ **sim**

5.16.1 Detailed Description

Contains description of vehicle class

Index

~Intersection
    Intersection, 20
~IntersectionwithSignal
    IntersectionwithSignal, 30
~IntersectionwithoutSignal
    IntersectionwithoutSignal, 26
~RandomNumGen
    RandomNumGen, 33
~TrafficLight
    TrafficLight, 40
~VehicleClass
    VehicleClass, 42
_Topology, 7
    _Topology, 8
    _Topology, 8
    ExitQ, 8
    I1, 8
    I2, 8
    I3, 8
    I4, 8
    I5, 8

AddVehicle
    RoadSegment, 35
addVehicletoQueue
    Intersection, 20
    IntersectionwithoutSignal, 27
    IntersectionwithSignal, 30

BUCKET_COUNT
    calender_queue.h, 48
back
    VehicleQueue, 44
BurstTime
    CommonDefs.h, 50
busy
    Intersection, 23
    VehicleQueue, 44

CALENDER_PERIOD
    calender_queue.h, 48
calender_queue, 9
    calender_queue, 9
    calender_queue, 9
    check659bucket, 9
    dequeue, 10
    get_bucket_count, 10
    getQsize, 10
    gettimeframe, 10
    init, 11
    insert, 11
    isEmpty, 11
    next_event, 11
    PopNext, 11, 12
    remove_event, 12
    resize, 12
calender_queue.h, 47
    BUCKET_COUNT, 48
    CALENDER_PERIOD, 48
    TOTAL_TIME, 48
changeSignalTrigger
    IntersectionwithSignal, 30
check659bucket
    calender_queue, 9
checkQinterval
    CommonDefs.h, 50
CommonDefs.h, 49
    BurstTime, 50
    checkQinterval, 50
    LPassTime, 50
    PassTime, 50
    reg, 51
    roadSegTime, 50
    startToPass, 50
    Time_t, 51
    turn, 51
cyclestate
    TrafficLight, 40

dequeue
    calender_queue, 10

EB
    IntersectionwithSignal, 31
EBI1
    Intersection, 23
EBI2
    Intersection, 24
empty
    VehicleQueue, 44
EndTime
    VehicleClass, 42
endTime
    VehicleClass, 43
Event0< T, OBJ >, 13
Event1< T, OBJ, U1, T1 >, 14
Event2< T, OBJ, U1, T1, U2, T2 >, 15
Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >, 16
event_compare, 17
EventBase, 17

eventDsc, 18
Events.h, 52
EvictQ
    Intersection, 20
EvictVehicle
    RoadSegment, 35
ExitQ
    _Topology, 8
    Intersection, 24

front
    VehicleQueue, 44

get_bucket_count
    calender_queue, 10
getDestination
    VehicleClass, 42
getDirection
    VehicleClass, 42
getID
    Intersection, 21
    VehicleClass, 42
getLastQ
    VehicleClass, 42
getLeftState
    TrafficLight, 40
GetLen
    VehicleQueue, 44
GetMaxLen
    VehicleQueue, 44
getNow
    Simulator, 36
getQdirection
    Intersection, 21
getQlane
    Intersection, 21
getQsize
    calender_queue, 10
getSource
    VehicleClass, 42
GetState
    RandomNumGen, 34
getState
    TrafficLight, 40
getTime
    node, 32
getType
    TrafficLight, 40
gettime
    RandomNum.h, 63
gettimeframe
    calender_queue, 10

haveSignal
    Intersection, 24

I1
    _Topology, 8
I2
    _Topology, 8
I3
    _Topology, 8
I4
    _Topology, 8
I5
    _Topology, 8
ID
    Intersection, 24
init
    calender_queue, 11
insert
    calender_queue, 11
instance
    Simulator, 39
Intersection, 18
    ∼Intersection, 20
    addVehicletoQueue, 20
    busy, 23
    EBI1, 23
    EBI2, 24
    EvictQ, 20
    ExitQ, 24
    getID, 21
    getQdirection, 21
    getQlane, 21
    haveSignal, 24
    ID, 24
    Intersection, 20
    NBI1, 24
    NBI2, 24
    NInter, 24
    NextQInfo, 21
    QCanGo, 22
    routingtable, 24
    SBI1, 24
    SBI2, 24
    SInter, 24
    VehicleDeparture, 22
    VehiclePass, 23
    WBI1, 24
    WBI2, 25
Intersection.h, 53
IntersectionwithSignal, 28
    ∼IntersectionwithSignal, 30
    addVehicletoQueue, 30
    changeSignalTrigger, 30
    EB, 31
    IntersectionwithSignal, 30
    IntersectionwithSignal, 30
    NB, 32
    QCanGo, 31
    SB, 32
    WB, 32
IntersectionwithSignal.h, 55
IntersectionwithoutSignal, 25
    ∼IntersectionwithoutSignal, 26
    addVehicletoQueue, 27

IntersectionwithoutSignal, 26
IntersectionwithoutSignal, 26
QCanGo, 27
IntersectionwoSignal.h, 57
isBusy
VehicleQueue, 44
isEmpty
calender_queue, 11

LPassTime
CommonDefs.h, 50
LastSentCar
VehicleQueue, 45

MINSTDG
RandomNum.h, 63
MINSTDM
RandomNum.h, 63
MINSTDX
RandomNum.h, 63
main.cpp, 58
maxLength
VehicleQueue, 45
myid
TrafficLight, 41

NB
IntersectionwithSignal, 32
NBI1
Intersection, 24
NBI2
Intersection, 24
NInter
Intersection, 24
Next
RandomNumGen, 34
next_event
calender_queue, 11
NextQInfo
Intersection, 21
node, 32
getTime, 32
node, 32
Now
Simulator, 36

PassTime
CommonDefs.h, 50
pop
VehicleQueue, 44
PopNext
calender_queue, 11, 12
PostProcStats
PostProcessing.h, 61
PostProcessing.h, 59
PostProcStats, 61
prioqueue, 33
push
VehicleQueue, 44

Q1
VehicleQueue, 45
QCanGo
Intersection, 22
IntersectionwithoutSignal, 27
IntersectionwithSignal, 31
qleft
RoadSegment, 35
qright
RoadSegment, 35

RandomNum.cc, 61
RandomNum.h, 62
gettime, 63
MINSTDG, 63
MINSTDM, 63
MINSTDX, 63
RandomNumGen, 33
∼RandomNumGen, 33
GetState, 34
Next, 34
RandomNumGen, 33
RandomNumGen, 33
Reset, 34
reg
CommonDefs.h, 51
remove_event
calender_queue, 12
Reset
RandomNumGen, 34
resize
calender_queue, 12
roadSegTime
CommonDefs.h, 50
RoadSegment, 34
AddVehicle, 35
EvictVehicle, 35
qleft, 35
qright, 35
RoadSegment, 35
RoadSegment, 35
RoadSegment.h, 63
routingtable
Intersection, 24
Run
Simulator, 37

SB
IntersectionwithSignal, 32
SBI1
Intersection, 24
SBI2
Intersection, 24
SInter
Intersection, 24
Schedule
Simulator, 37, 38
scheduleVehicles
scheduleVehicles.h, 64

scheduleVehicles.h, 63
  scheduleVehicles, 64
setEndTime
  VehicleClass, 42
setLastQ
  VehicleClass, 42
Simulator, 36
  getNow, 36
  instance, 39
  Now, 36
  Run, 37
  Schedule, 37, 38
  Simulator, 36
  Stop, 38
  StopAt, 38
Simulator.h, 65
StartTime
  VehicleClass, 42
startTime
  VehicleClass, 43
startToPass
  CommonDefs.h, 50
Stop
  Simulator, 38
StopAt
  Simulator, 38

TOTAL_TIME
  calender_queue.h, 48
Time_t
  CommonDefs.h, 51
Topology.h, 66
TrafficLight, 39
  ∼TrafficLight, 40
  cyclestate, 40
  getLeftState, 40
  getState, 40
  getType, 40
  myid, 41
  TrafficLight, 39
  TrafficLight, 39
  type, 41
TrafficLight.h, 68
turn
  CommonDefs.h, 51
type
  TrafficLight, 41

updateDirection
  VehicleClass, 43

VehicleClass, 41
  ∼VehicleClass, 42
  EndTime, 42
  endTime, 43
  getDestination, 42
  getDirection, 42
  getID, 42
  getLastQ, 42
  getSource, 42
  setEndTime, 42
  setLastQ, 42
  StartTime, 42
  startTime, 43
  updateDirection, 43
  VehicleClass, 41
  VehicleClass, 41
VehicleClass.h, 69
VehicleDeparture
  Intersection, 22
VehiclePass
  Intersection, 23
VehicleQueue, 43
  back, 44
  busy, 44
  empty, 44
  front, 44
  GetLen, 44
  GetMaxLen, 44
  isBusy, 44
  LastSentCar, 45
  maxLength, 45
  pop, 44
  push, 44
  Q1, 45
  VehicleQueue, 43, 44
  VehicleQueue, 43, 44

WB
  IntersectionwithSignal, 32
WBI1
  Intersection, 24
WBI2
  Intersection, 25