

## CSE6730 Modeling & Simulation : Project-1

Generated by Doxygen 1.8.3.1

Mon Feb 18 2013 21:11:52



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Class Documentation</b>	<b>7</b>
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
4.2.3	Member Function Documentation . . . . .	9
4.2.3.1	check659bucket . . . . .	9
4.2.3.2	dequeue . . . . .	9
4.2.3.3	get_bucket_count . . . . .	9
4.2.3.4	getQsize . . . . .	9
4.2.3.5	gettimeframe . . . . .	10
4.2.3.6	insert . . . . .	10
4.2.3.7	isEmpty . . . . .	10

4.2.3.8	next_event	10
4.2.3.9	PopNext	10
4.2.3.10	remove_event	10
4.3	Event0< T, OBJ > Class Template Reference	11
4.4	Event1< T, OBJ, U1, T1 > Class Template Reference	12
4.5	Event2< T, OBJ, U1, T1, U2, T2 > Class Template Reference	13
4.6	Event3< T, OBJ, U1, T1, U2, T2, U3, T3 > Class Template Reference	14
4.7	event_compare Class Reference	15
4.8	EventBase Class Reference	15
4.9	eventDsc Struct Reference	16
4.10	Intersection Class Reference	16
4.10.1	Constructor & Destructor Documentation	18
4.10.1.1	Intersection	18
4.10.1.2	Intersection	18
4.10.1.3	~Intersection	18
4.10.2	Member Function Documentation	18
4.10.2.1	addVehicleToQueue	18
4.10.2.2	EvictQ	18
4.10.2.3	getID	19
4.10.2.4	getQdirection	19
4.10.2.5	getQlane	19
4.10.2.6	NextQInfo	19
4.10.2.7	QCanGo	20
4.10.2.8	VehicleDeparture	20
4.10.2.9	VehiclePass	21
4.10.3	Member Data Documentation	21
4.10.3.1	busy	21
4.10.3.2	EBI1	21
4.10.3.3	EBI2	22
4.10.3.4	ExitQ	22
4.10.3.5	haveSignal	22
4.10.3.6	ID	22
4.10.3.7	NBI1	22
4.10.3.8	NBI2	22
4.10.3.9	NInter	22
4.10.3.10	routingtable	22
4.10.3.11	SBI1	22
4.10.3.12	SBI2	22
4.10.3.13	SInter	22
4.10.3.14	WBI1	22

4.10.3.15 WBI2 . . . . .	23
4.11 IntersectionwithoutSignal Class Reference . . . . .	23
4.11.1 Constructor & Destructor Documentation . . . . .	24
4.11.1.1 IntersectionwithoutSignal . . . . .	24
4.11.1.2 IntersectionwithoutSignal . . . . .	24
4.11.1.3 ~IntersectionwithoutSignal . . . . .	25
4.11.2 Member Function Documentation . . . . .	25
4.11.2.1 addVehicletoQueue . . . . .	25
4.11.2.2 QCanGo . . . . .	25
4.12 IntersectionwithSignal Class Reference . . . . .	26
4.12.1 Constructor & Destructor Documentation . . . . .	28
4.12.1.1 IntersectionwithSignal . . . . .	28
4.12.1.2 IntersectionwithSignal . . . . .	28
4.12.1.3 ~IntersectionwithSignal . . . . .	28
4.12.2 Member Function Documentation . . . . .	28
4.12.2.1 addVehicletoQueue . . . . .	28
4.12.2.2 changeSignalTrigger . . . . .	28
4.12.2.3 QCanGo . . . . .	29
4.12.3 Member Data Documentation . . . . .	29
4.12.3.1 EB . . . . .	29
4.12.3.2 NB . . . . .	30
4.12.3.3 SB . . . . .	30
4.12.3.4 WB . . . . .	30
4.13 prioqueue Class Reference . . . . .	30
4.14 RandomNumGen Class Reference . . . . .	30
4.14.1 Constructor & Destructor Documentation . . . . .	30
4.14.1.1 RandomNumGen . . . . .	30
4.14.1.2 RandomNumGen . . . . .	31
4.14.1.3 ~RandomNumGen . . . . .	31
4.14.2 Member Function Documentation . . . . .	31
4.14.2.1 GetState . . . . .	31
4.14.2.2 Next . . . . .	31
4.14.2.3 Reset . . . . .	31
4.15 RoadSegment Class Reference . . . . .	31
4.15.1 Constructor & Destructor Documentation . . . . .	32
4.15.1.1 RoadSegment . . . . .	32
4.15.2 Member Function Documentation . . . . .	32
4.15.2.1 AddVehicle . . . . .	32
4.15.2.2 EvictVehicle . . . . .	32
4.15.3 Member Data Documentation . . . . .	32

4.15.3.1	qleft	32
4.15.3.2	qright	32
4.16	Simulator Class Reference	33
4.16.1	Constructor & Destructor Documentation	33
4.16.1.1	Simulator	33
4.16.2	Member Function Documentation	33
4.16.2.1	getNow	33
4.16.2.2	Now	34
4.16.2.3	Run	34
4.16.2.4	Schedule	34
4.16.2.5	Schedule	34
4.16.2.6	Schedule	35
4.16.2.7	Schedule	35
4.16.2.8	Stop	35
4.16.2.9	StopAt	36
4.16.3	Member Data Documentation	36
4.16.3.1	instance	36
4.17	TrafficLight Class Reference	36
4.17.1	Constructor & Destructor Documentation	36
4.17.1.1	TrafficLight	36
4.17.1.2	TrafficLight	37
4.17.1.3	~TrafficLight	37
4.17.2	Member Function Documentation	37
4.17.2.1	cyclestate	37
4.17.2.2	getLeftState	37
4.17.2.3	getState	37
4.17.2.4	getType	38
4.17.3	Member Data Documentation	38
4.17.3.1	myid	38
4.17.3.2	type	38
4.18	VehicleClass Class Reference	38
4.18.1	Constructor & Destructor Documentation	38
4.18.1.1	VehicleClass	38
4.18.1.2	~VehicleClass	39
4.18.2	Member Function Documentation	39
4.18.2.1	EndTime	39
4.18.2.2	getDestination	39
4.18.2.3	getDirection	39
4.18.2.4	getID	39
4.18.2.5	getLastQ	39

4.18.2.6	getSource	39
4.18.2.7	setEndTime	39
4.18.2.8	setLastQ	39
4.18.2.9	StartTime	40
4.18.2.10	updateDirection	40
4.18.3	Member Data Documentation	40
4.18.3.1	endTime	40
4.18.3.2	startTime	40
4.19	VehicleQueue Class Reference	40
4.19.1	Constructor & Destructor Documentation	40
4.19.1.1	VehicleQueue	40
4.19.2	Member Function Documentation	41
4.19.2.1	back	41
4.19.2.2	empty	41
4.19.2.3	front	41
4.19.2.4	GetLen	41
4.19.2.5	GetMaxLen	41
4.19.2.6	isBusy	41
4.19.2.7	pop	41
4.19.2.8	push	41
4.19.3	Member Data Documentation	41
4.19.3.1	busy	41
4.19.3.2	LastSentCar	41
4.19.3.3	Q1	42
<b>5</b>	<b>File Documentation</b>	<b>43</b>
5.1	calender_queue.h File Reference	43
5.1.1	Detailed Description	44
5.1.2	Macro Definition Documentation	44
5.1.2.1	BUCKET_COUNT	45
5.1.2.2	CALENDER_PERIOD	45
5.1.2.3	TOTAL_TIME	45
5.2	CommonDefs.h File Reference	45
5.2.1	Detailed Description	47
5.2.2	Macro Definition Documentation	47
5.2.2.1	BurstTime	47
5.2.2.2	checkQinterval	47
5.2.2.3	LPassTime	47
5.2.2.4	PassTime	47
5.2.2.5	roadSegTime	47

5.2.2.6	startToPass	47
5.2.3	Typedef Documentation	47
5.2.3.1	Time_t	47
5.2.4	Function Documentation	47
5.2.4.1	reg	47
5.2.4.2	turn	48
5.3	Events.h File Reference	48
5.3.1	Detailed Description	50
5.4	Intersection.h File Reference	50
5.4.1	Detailed Description	51
5.5	IntersectionwithSignal.h File Reference	51
5.5.1	Detailed Description	53
5.6	IntersectionwoSignal.h File Reference	53
5.6.1	Detailed Description	54
5.7	main.cpp File Reference	54
5.8	PostProcessing.h File Reference	56
5.8.1	Detailed Description	56
5.8.2	Function Documentation	57
5.8.2.1	PostProcStats	57
5.9	RandomNum.h File Reference	58
5.9.1	Detailed Description	58
5.9.2	Macro Definition Documentation	58
5.9.2.1	MINSTDG	58
5.9.2.2	MINSTDM	58
5.9.2.3	MINSTDY	59
5.10	RoadSegment.h File Reference	59
5.10.1	Detailed Description	59
5.11	scheduleVehicles.h File Reference	59
5.11.1	Detailed Description	60
5.11.2	Function Documentation	60
5.11.2.1	scheduleVehicles	60
5.12	Simulator.h File Reference	61
5.12.1	Detailed Description	62
5.13	Topology.h File Reference	62
5.13.1	Detailed Description	63
5.14	TrafficLight.h File Reference	63
5.14.1	Detailed Description	65
5.15	VehicleClass.h File Reference	65
5.15.1	Detailed Description	67







# 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 . . . . .	15
EventBase . . . . .	15
Event0< T, OBJ > . . . . .	11
Event1< T, OBJ, U1, T1 > . . . . .	12
Event2< T, OBJ, U1, T1, U2, T2 > . . . . .	13
Event3< T, OBJ, U1, T1, U2, T2, U3, T3 > . . . . .	14
eventDsc . . . . .	16
Intersection . . . . .	16
IntersectionwithoutSignal . . . . .	23
IntersectionwithSignal . . . . .	26
prioqueue . . . . .	30
RandomNumGen . . . . .	30
RoadSegment . . . . .	31
Simulator . . . . .	33
TrafficLight . . . . .	36
VehicleClass . . . . .	38
VehicleQueue . . . . .	40



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">_Topology</a>	7
<a href="#">calender_queue</a>	9
<a href="#">Event0&lt; T, OBJ &gt;</a>	11
<a href="#">Event1&lt; T, OBJ, U1, T1 &gt;</a>	12
<a href="#">Event2&lt; T, OBJ, U1, T1, U2, T2 &gt;</a>	13
<a href="#">Event3&lt; T, OBJ, U1, T1, U2, T2, U3, T3 &gt;</a>	14
<a href="#">event_compare</a>	15
<a href="#">EventBase</a>	15
<a href="#">eventDsc</a>	16
<a href="#">Intersection</a>	16
<a href="#">IntersectionwithoutSignal</a>	23
<a href="#">IntersectionwithSignal</a>	26
<a href="#">prioqueue</a>	30
<a href="#">RandomNumGen</a>	30
<a href="#">RoadSegment</a>	31
<a href="#">Simulator</a>	33
<a href="#">TrafficLight</a>	36
<a href="#">VehicleClass</a>	38
<a href="#">VehicleQueue</a>	40



## Chapter 3

# File Index

### 3.1 File List

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

<a href="#">calender_queue.h</a>	
Declaration of the class calender queue . . . . .	43
<a href="#">CommonDefs.h</a> . . . . .	45
<a href="#">Events.h</a>	
Declaration of various types of events . . . . .	48
<a href="#">Intersection.h</a> . . . . .	50
<a href="#">IntersectionwithSignal.h</a> . . . . .	51
<a href="#">IntersectionwoSignal.h</a> . . . . .	53
<a href="#">main.cpp</a> . . . . .	54
<a href="#">PostProcessing.h</a> . . . . .	56
<b>prioqueue.h</b> . . . . .	??
<a href="#">RandomNum.h</a> . . . . .	58
<a href="#">RoadSegment.h</a> . . . . .	59
<a href="#">scheduleVehicles.h</a> . . . . .	59
<a href="#">Simulator.h</a> . . . . .	61
<b>stdafx.h</b> . . . . .	??
<a href="#">Topology.h</a> . . . . .	62
<a href="#">TrafficLight.h</a>	
Description of functionality of traffic light . . . . .	63
<a href="#">VehicleClass.h</a> . . . . .	65
<b>VehicleQueue.h</b> . . . . .	??
testing/ <b>test1.h</b> . . . . .	??
testing/ <b>testing.h</b> . . . . .	??



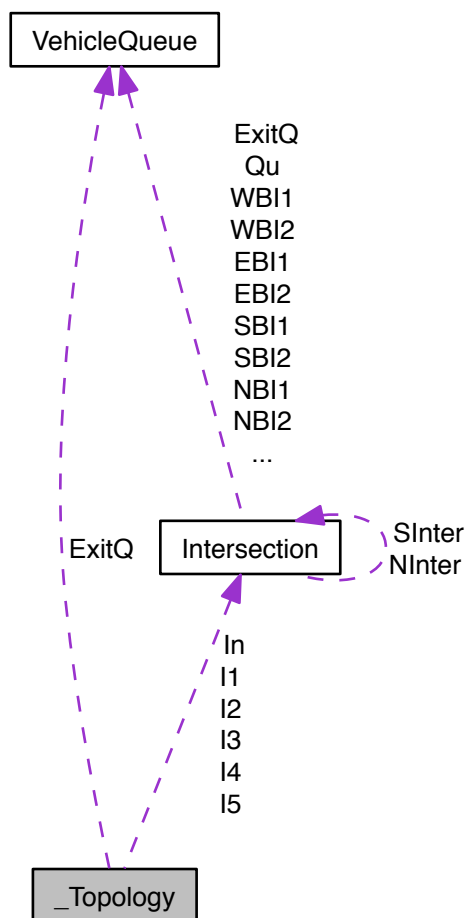


## Chapter 4

# 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](#) ()

### 4.2.1 Detailed Description

Calender Queue is priority queue, Ref: Calendar queues: a fast  $O(1)$  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

### 4.2.3 Member Function Documentation

#### 4.2.3.1 [void calender\\_queue::check659bucket](#) ( )

is there debuggin purpose

#### 4.2.3.2 [void calender\\_queue::dequeue](#) ( [EventBase](#) \* *E1* )

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

#### Parameters

<i>E1</i>	: event pointer to removed from the list
-----------	--

#### 4.2.3.3 [int calender\\_queue::get\\_bucket\\_count](#) ( )

Get number of buckets in the calender queue

#### 4.2.3.4 [int calender\\_queue::getQsize](#) ( )

Returns number of element in the Q

#### 4.2.3.5 int calender\_queue::gettimeframe ( )

Returns the time frame from which last event was popped

#### 4.2.3.6 void calender\_queue::insert ( **EventBase** \* *E1* )

Inserts an event into the priority list

##### Parameters

<i>E1</i>	is the even to be inserted into the list
-----------	--

#### 4.2.3.7 int calender\_queue::isEmpty ( )

Checks if there are anymore events left in the list

#### 4.2.3.8 **EventBase** \* calender\_queue::next\_event ( int *bucket\_num* )

Returns event with minimum time from bucket-num

##### Parameters

<i>bucket_num</i>	is the number of bucket from which you want to get the min time stamp event
-------------------	---

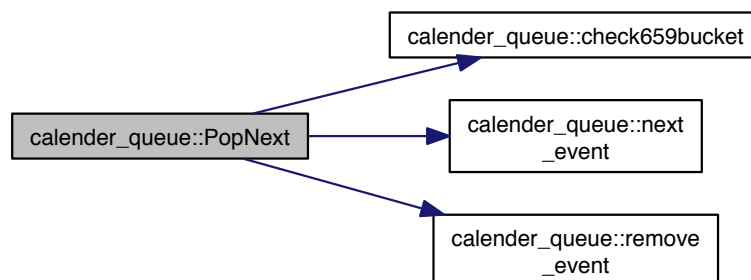
#### 4.2.3.9 **EventBase** \* calender\_queue::PopNext ( )

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:



#### 4.2.3.10 void calender\_queue::remove\_event ( int *bucket\_num*, **EventBase** \* *E1* )

Removes event *E1* from *bucket\_num*<sup>th</sup> bucket

## Parameters

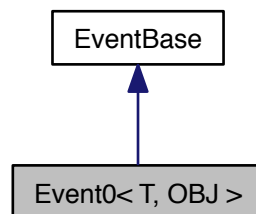
<i>BUcket_num</i>	is the id of bucket from which event is to be removed
<i>E1</i>	is pointer to event to be removed

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

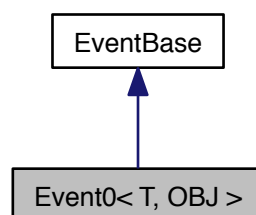
- [calender\\_queue.h](#)
- [calender\\_queue.cc](#)

### 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)

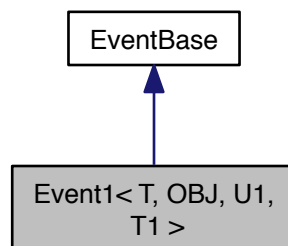
- OBJ \* **obj**

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

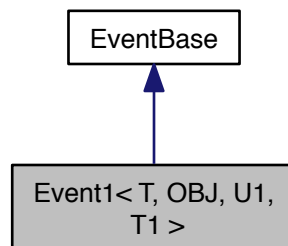
- [Events.h](#)

#### 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**

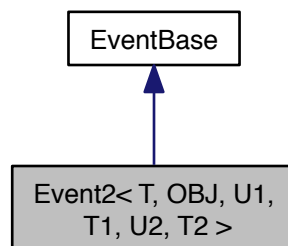
- T1 t1

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

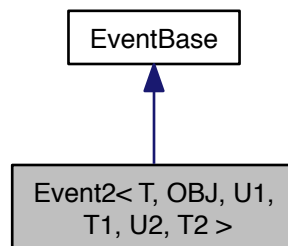
- [Events.h](#)

## 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**

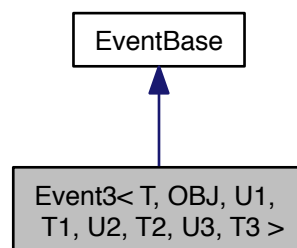
- T1 **t1**
- T2 **t2**

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

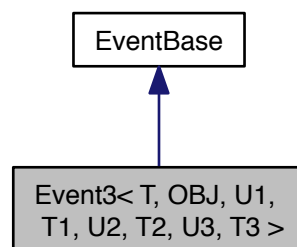
- [Events.h](#)

## 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

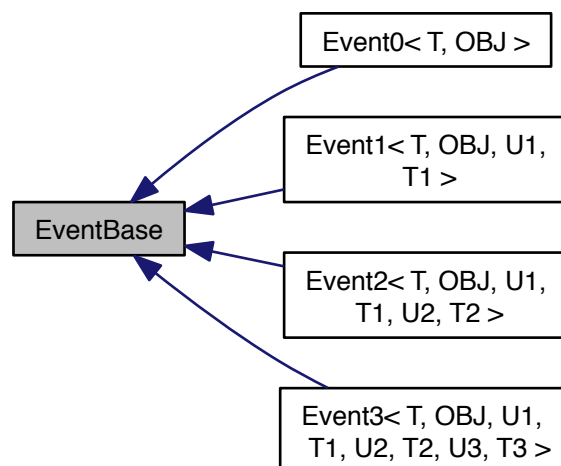
- bool **operator()** ([EventBase](#) \*const &l, const [EventBase](#) \*const &r) const

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

- [Events.h](#)

## 4.8 EventBase Class Reference

Inheritance diagram for EventBase:



### 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

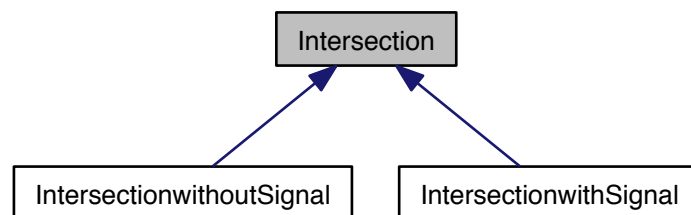
- int **type**
- int **InterID**
- int **QDir**
- int **QLane**
- int **QSize**
- double **timetag**

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

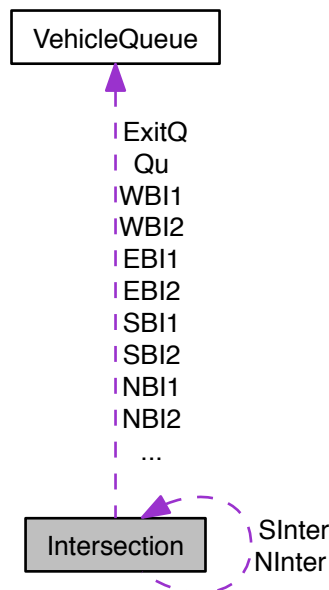
- [testing/test1.h](#)

## 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 **NBllength**
- int **SBllength**
- [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

<i>num</i>	
------------	--

### 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]

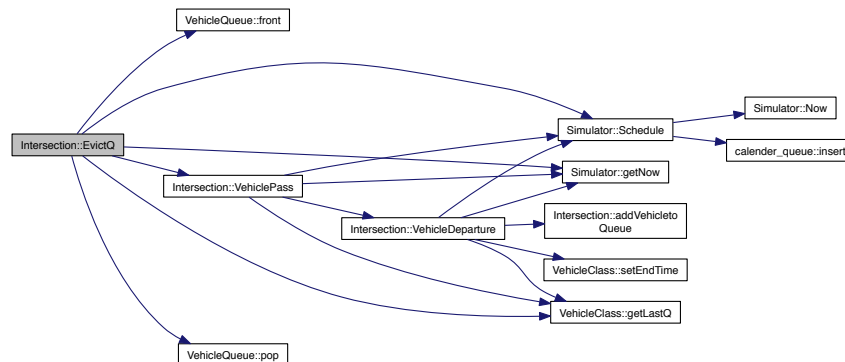
Virtual function Adds vehicle into queue

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

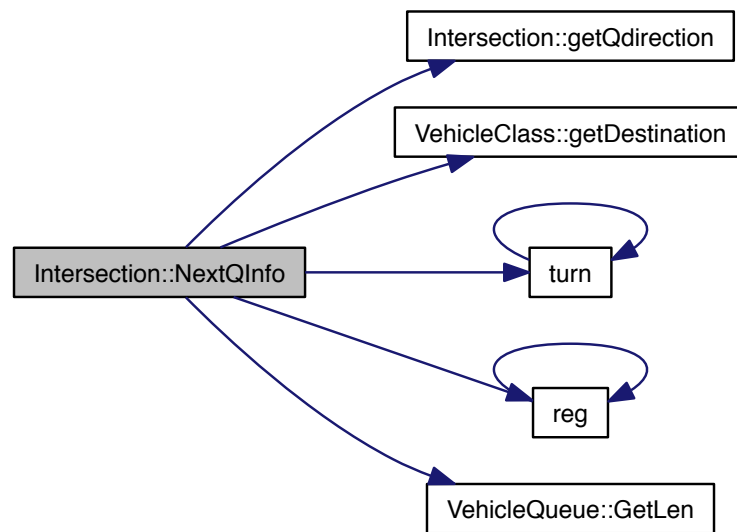
#### 4.10.2.5 `int Intersection::getQlane ( Intersection * inter, VehicleQueue * Q )`

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](#).

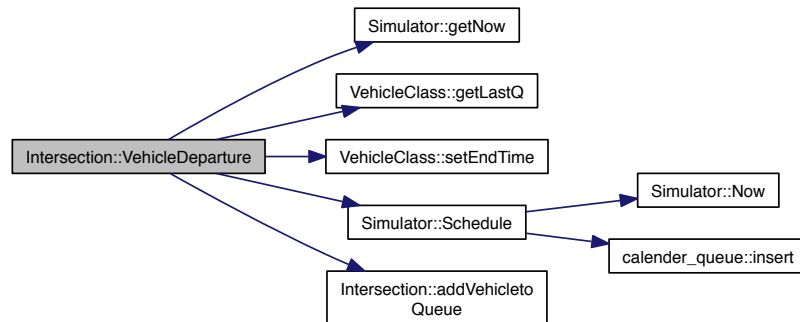
4.10.2.8 `void Intersection::VehicleDeparture ( VehicleClass * vehicle )`

Departs Vehicle from the intersection

Parameters

<i>vehicle</i>	is the vehicle to be departed
----------------	-------------------------------

Here is the call graph for this function:



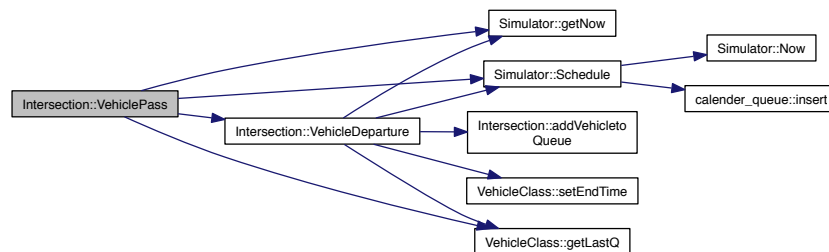
#### 4.10.2.9 void Intersection::VehiclePass ( VehicleClass \* vehicle, int Turn )

Logic of vehicle passing through this intersection

##### Parameters

<i>Vehicle</i>	
<i>turn</i>	

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::EBI1

Vehicle Queue East bound lane 1

#### 4.10.3.3 **VehicleQueue\*** Intersection::**EBI2**

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::**NBI1**

Vehicle Queue North bound lane 1

#### 4.10.3.8 **VehicleQueue\*** Intersection::**NBI2**

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::**SBI1**

Vehicle Queue South bound lane 1

#### 4.10.3.12 **VehicleQueue\*** Intersection::**SBI2**

Vehicle Queue South bound lane 2

#### 4.10.3.13 **Intersection\*** Intersection::**SInter**

Neighboring intersection in the South

#### 4.10.3.14 **VehicleQueue\*** Intersection::**WBI1**

Vehicle Queue West bound lane 1



## 4.10.3.15 VehicleQueue\* Intersection::WBI2

Vehicle Queue West bound lane 2

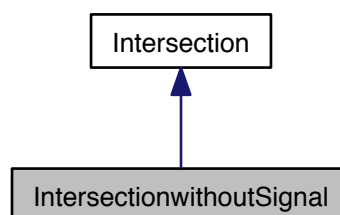
The documentation for this class was generated from the following files:

- [Intersection.h](#)

- [Intersection.cpp](#)

## 4.11 IntersectionwithoutSignal Class Reference

Inheritance diagram for IntersectionwithoutSignal:





## 4.11.1.3 IntersectionwithoutSignal::~~IntersectionwithoutSignal ( void )

Default Destructor

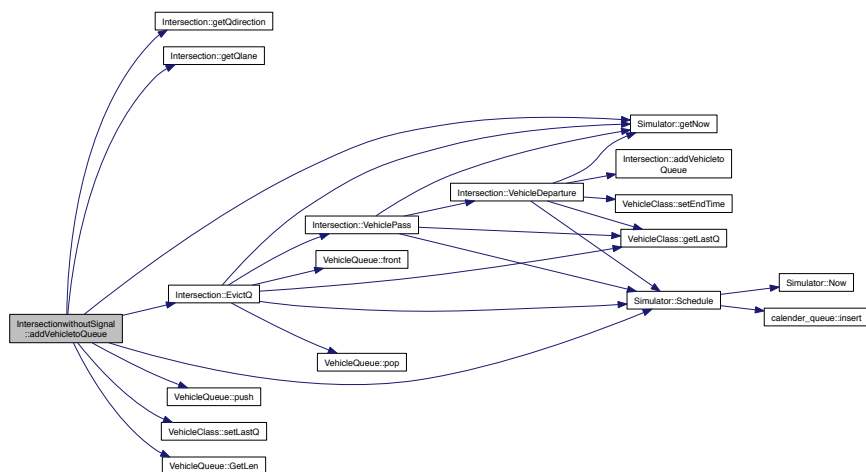
## 4.11.2 Member Function Documentation

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

Adds to outgoing queue or removes vehicles

Implements [Intersection](#).

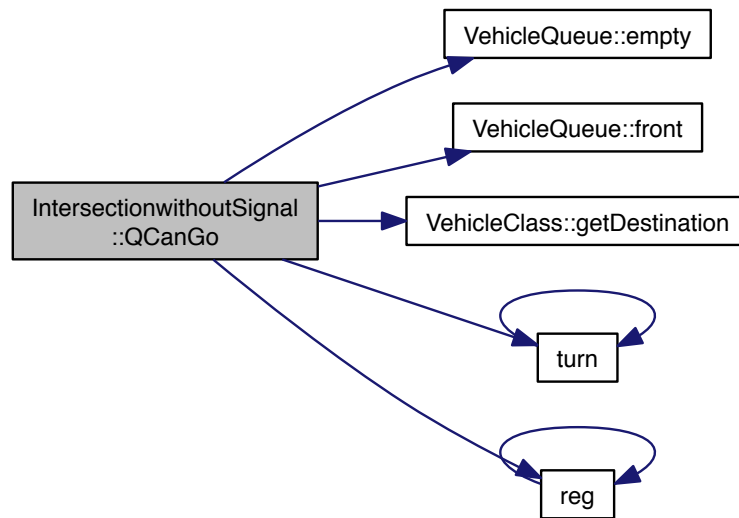
Here is the call graph for this function:

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:

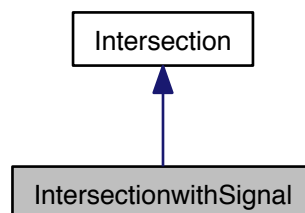


The documentation for this class was generated from the following files:

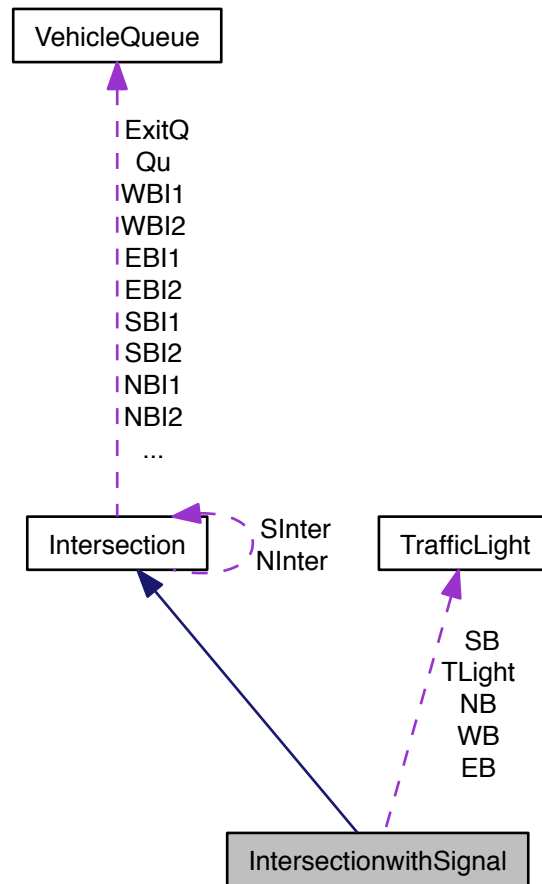
- [IntersectionwoSignal.h](#)
- `IntersectionwoSignal.cpp`

## 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

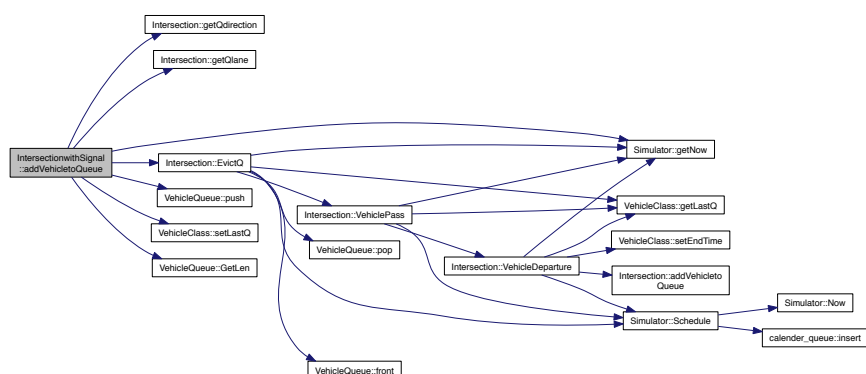
### 4.12.2 Member Function Documentation

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

Adds to outgoing queue or removes vehicles

Implements [Intersection](#).

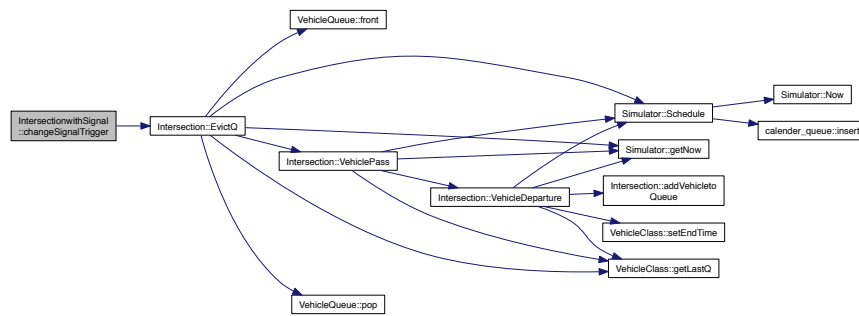
Here is the call graph for this function:



#### 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:

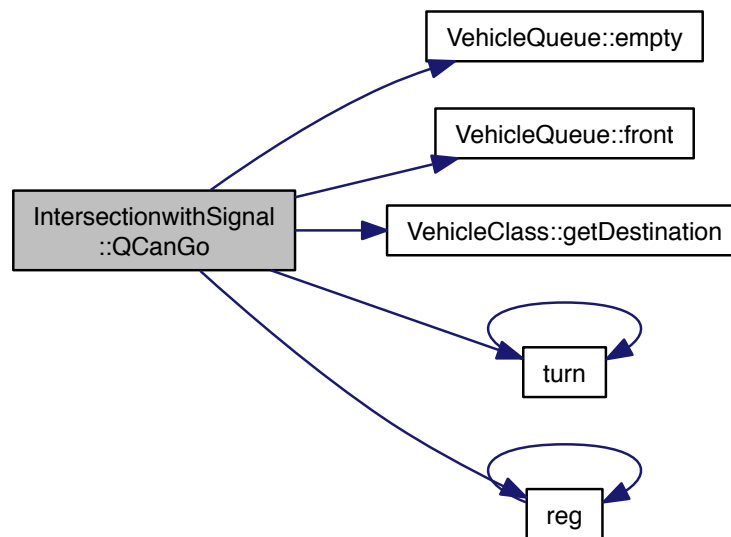


#### 4.12.2.3 int IntersectionwithSignal::QCanGo ( int *direction*, int *lane* ) [virtual]

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 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

### 4.14 RandomNumGen Class Reference

#### Public Member Functions

- [RandomNumGen](#) ()
- [RandomNumGen](#) (unsigned long x0)
- double [Next](#) ()
- void [Reset](#) ()
- unsigned long [GetState](#) ()
- [~RandomNumGen](#) ()

#### 4.14.1 Constructor & Destructor Documentation

##### 4.14.1.1 RandomNumGen::RandomNumGen ( )

Constructor: Initializes the default parameters of Random number generator



## 4.14.1.2 RandomNumGen::RandomNumGen ( unsigned long x0 )

Constructor: 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
----	--

## 4.14.1.3 RandomNumGen::~~RandomNumGen ( )

Destructor for random number generator

## 4.14.2 Member Function Documentation

## 4.14.2.1 unsigned long RandomNumGen::GetState ( )

Gives state of random genrator. Used for debugging purpose

## 4.14.2.2 double RandomNumGen::Next ( )

Genrates next random number

## 4.14.2.3 void RandomNumGen::Reset ( )

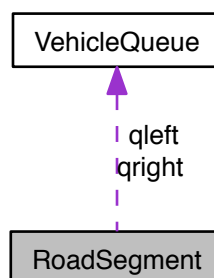
Resets the random number generator

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

- [RandomNum.h](#)
- RandomNum.cc

## 4.15 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

### 4.15.1 Constructor & Destructor Documentation

4.15.1.1 `RoadSegment::RoadSegment ( dir direction, Intersection * par, int cap )` `[inline]`

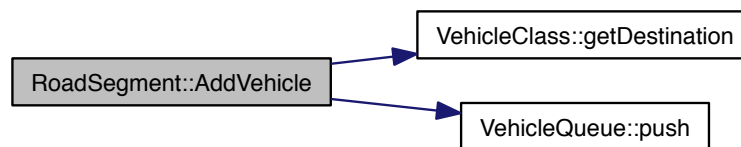
Constructor

### 4.15.2 Member Function Documentation

4.15.2.1 `void RoadSegment::AddVehicle ( VehicleClass * vehicle )`

Adds vehicle to the road segment

Here is the call graph for this function:



4.15.2.2 `void RoadSegment::EvictVehicle ( )`

Evicts vehicle from the Road Segment

### 4.15.3 Member Data Documentation

4.15.3.1 `VehicleQueue RoadSegment::qleft`

Left lane (Vehicle Queue)

4.15.3.2 `VehicleQueue RoadSegment::qright`

Right lane (Vehicle queue)

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

- [RoadSegment.h](#)
- [RoadSegment.cpp](#)

## 4.16 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

#### 4.16.1 Constructor & Destructor Documentation

##### 4.16.1.1 Simulator::Simulator ( )

Default constructor

#### 4.16.2 Member Function Documentation

##### 4.16.2.1 Time\_t Simulator::getNow ( ) [inline]

Returns the current time of the simulation

#### 4.16.2.2 `Time_t Simulator::Now ( ) [static]`

Returns the time NOW

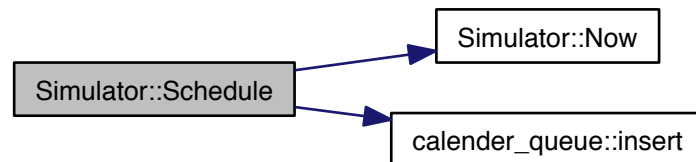
#### 4.16.2.3 `void Simulator::Run ( ) [static]`

Start executing events

#### 4.16.2.4 `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:



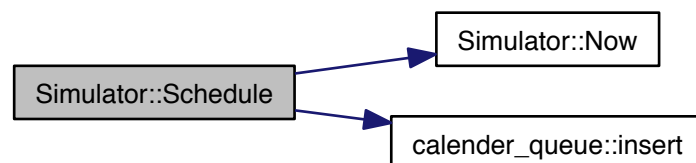
#### 4.16.2.5 `template<typename T , typename OBJ , typename U1 , typename T1 > void Simulator::Schedule ( double t, void(T::*)(U1) handler, OBJ * obj, T1 t1 ) [inline]`

Schedules the event type1

See Also

[Events.h](#)

Here is the call graph for this function:



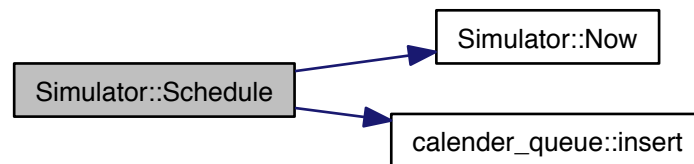
4.16.2.6 `template<typename T , typename OBJ , typename U1 , typename T1 , typename U2 , typename T2 > void Simulator::Schedule ( double t, void(T::*)(U1, U2) handler, OBJ * obj, T1 t1, T2 t2 ) [inline]`

Schedules the event type2

See Also

[Events.h](#)

Here is the call graph for this function:



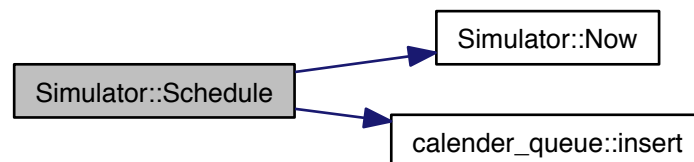
4.16.2.7 `template<typename T , typename OBJ , typename U1 , typename T1 , typename U2 , typename T2 , typename U3 , typename T3 > void Simulator::Schedule ( double t, void(T::*)(U1, U2, U3) handler, OBJ * obj, T1 t1, T2 t2, T3 t3 ) [inline]`

Schedules the event type3

See Also

[Events.h](#)

Here is the call graph for this function:



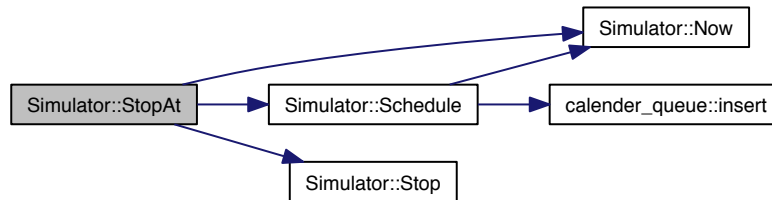
4.16.2.8 `void Simulator::Stop ( )`

Stops executing events

#### 4.16.2.9 void Simulator::StopAt ( Time\_t t ) [static]

Defines stopping time

Here is the call graph for this function:



### 4.16.3 Member Data Documentation

#### 4.16.3.1 Simulator \* Simulator::instance =0 [static]

Pointer to simulator

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

- [Simulator.h](#)
- [Simulator.cpp](#)

## 4.17 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](#)

### 4.17.1 Constructor & Destructor Documentation

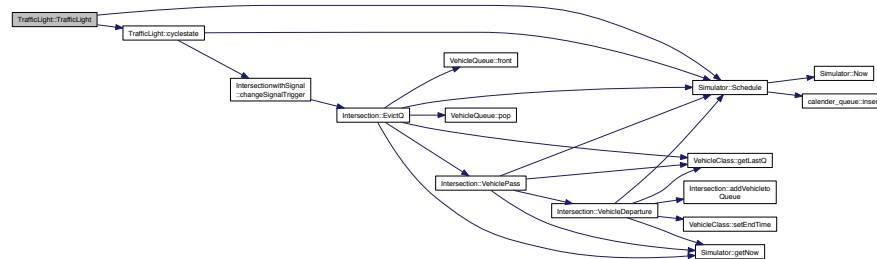
#### 4.17.1.1 TrafficLight::TrafficLight ( )

Default constructor

4.17.1.2 `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:



4.17.1.3 `TrafficLight::~~TrafficLight ( )`

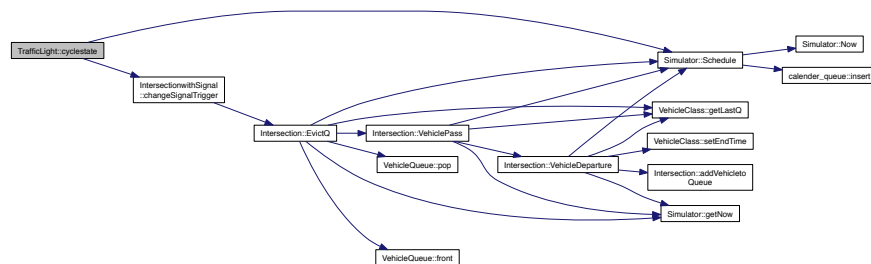
Default destructor

## 4.17.2 Member Function Documentation

4.17.2.1 `void TrafficLight::cyclestate ( int leftorthru )`

cyclestate

Here is the call graph for this function:



4.17.2.2 `state TrafficLight::getLeftState ( ) [inline]`

Returns the present left state of the traffic light

4.17.2.3 `state TrafficLight::getState ( ) [inline]`

Returns the present state of the traffic light

#### 4.17.2.4 int TrafficLight::getType ( ) [inline]

Returns the type of traffic light

### 4.17.3 Member Data Documentation

#### 4.17.3.1 int TrafficLight::myid

Id of the traffic signal

#### 4.17.3.2 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](#)

## 4.18 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](#)

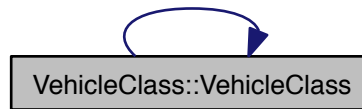
### 4.18.1 Constructor & Destructor Documentation

#### 4.18.1.1 VehicleClass::VehicleClass ( int id, int start, int Dest, Time\_t starttime )

Default constructor



Here is the call graph for this function:



#### 4.18.1.2 VehicleClass::~~VehicleClass ( )

Default destructor

### 4.18.2 Member Function Documentation

#### 4.18.2.1 Time\_t VehicleClass::EndTime ( ) [inline]

Returns endtime of the car

#### 4.18.2.2 int VehicleClass::getDestination ( )

Returns destination of the car

#### 4.18.2.3 dir VehicleClass::getDirection ( )

outputs the direction of the car

#### 4.18.2.4 int VehicleClass::getID ( )

Gets the ID of the car

#### 4.18.2.5 VehicleQueue \* VehicleClass::getLastQ ( )

outputs the last queue

#### 4.18.2.6 int VehicleClass::getSource ( )

Returns the source of the car

#### 4.18.2.7 void VehicleClass::setEndTime ( Time\_t t )

Sets the end time

#### 4.18.2.8 void VehicleClass::setLastQ ( VehicleQueue \* Q )

Sets the Vehicle queue to which the car eblonged

#### 4.18.2.9 Time\_t VehicleClass::StartTime ( ) [inline]

Returns the start time

#### 4.18.2.10 void VehicleClass::updateDirection ( dir *Direction* )

!Constructor

Updated the direction of the car

### 4.18.3 Member Data Documentation

#### 4.18.3.1 Time\_t VehicleClass::endTime

Time when a car exits out of the system

#### 4.18.3.2 Time\_t VehicleClass::startTime

Time when a car comes into existence

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

- [VehicleClass.h](#)
- [VehicleClass.cpp](#)

## 4.19 VehicleQueue Class Reference

### Public Member Functions

- [VehicleQueue](#) ( )
- [VehicleClass](#) \* [front](#) ( )
- bool [empty](#) ( )
- void [push](#) ([VehicleClass](#) \*V1)
- void [pop](#) ( )
- [VehicleClass](#) \* [back](#) ( )
- int [GetMaxLen](#) ( )
- int [GetLen](#) ( )
- bool [isBusy](#) ( )

### Public Attributes

- std::queue< [VehicleClass](#) \* > [Q1](#)
- int [busy](#)
- double [LastSentCar](#)

### 4.19.1 Constructor & Destructor Documentation

#### 4.19.1.1 VehicleQueue::VehicleQueue ( )

Default constructor

## 4.19.2 Member Function Documentation

### 4.19.2.1 VehicleClass \* VehicleQueue::back ( )

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

### 4.19.2.2 bool VehicleQueue::empty ( )

Returns "true" if the queue is empty

### 4.19.2.3 VehicleClass \* VehicleQueue::front ( )

Returns pointer of the vehicle which is front of the queue

### 4.19.2.4 int VehicleQueue::GetLen ( )

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

### 4.19.2.5 int VehicleQueue::GetMaxLen ( )

Returns Maximum possible length of the queue

### 4.19.2.6 bool VehicleQueue::isBusy ( )

Returns if the queue is busy or not

### 4.19.2.7 void VehicleQueue::pop ( )

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

### 4.19.2.8 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
----	---

## 4.19.3 Member Data Documentation

### 4.19.3.1 int VehicleQueue::busy

to check if the queue is busy/not

### 4.19.3.2 double VehicleQueue::LastSentCar

Holds time for vehicle that was sent last

#### 4.19.3.3 `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

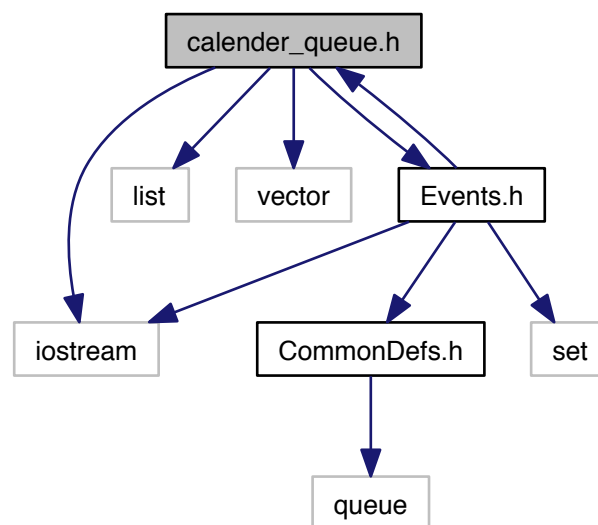
# File Documentation

### 5.1 calender\_queue.h File Reference

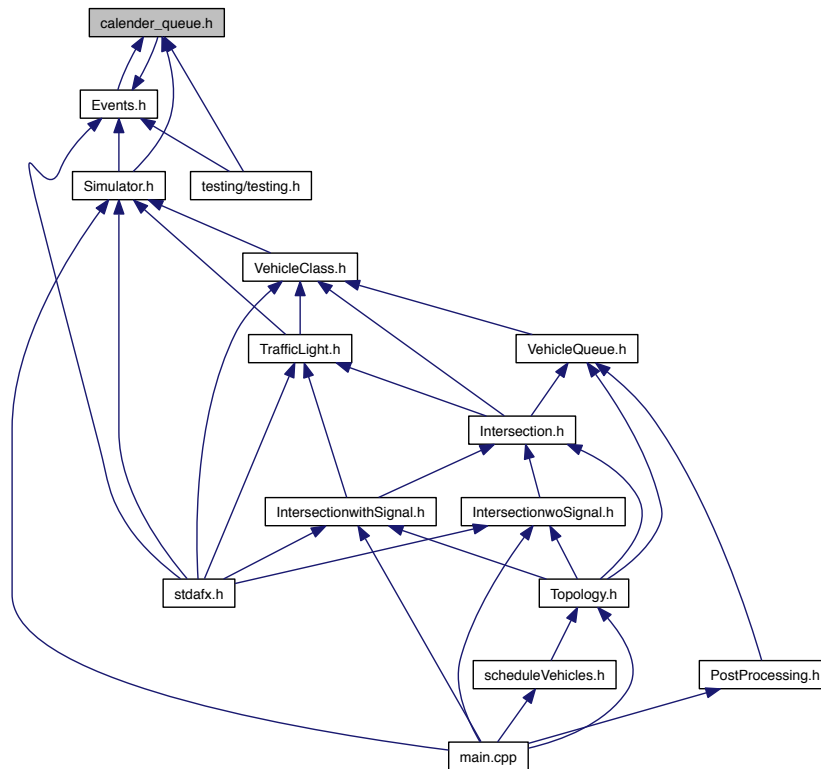
declartion of the class calender queue

```
#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

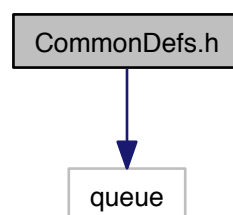
### 5.1.2.3 `#define TOTAL_TIME 120*60`

Total time of the simulation

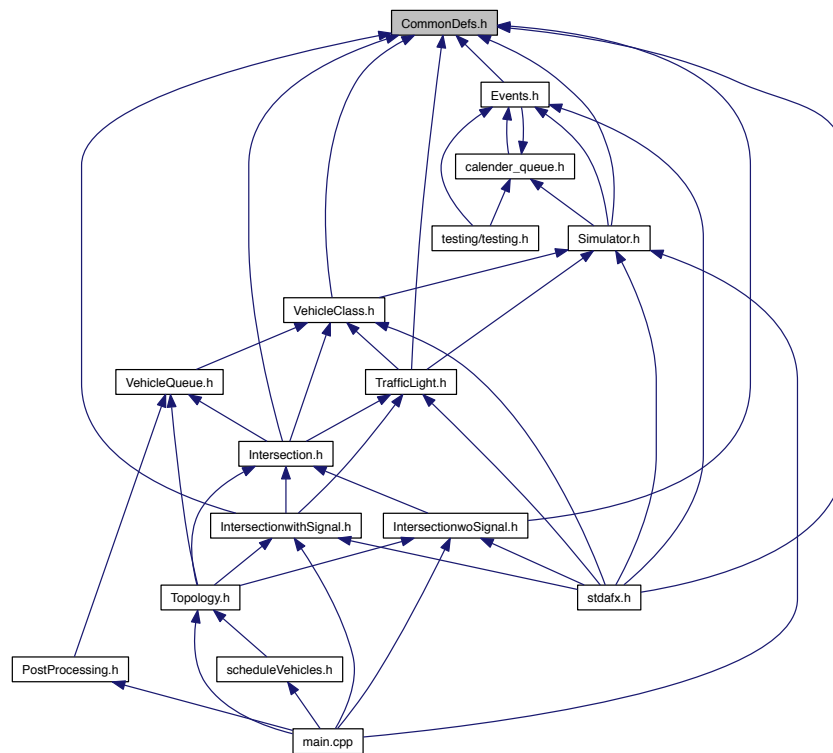
## 5.2 CommonDefs.h File Reference

```
#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 common definitions 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 amount 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

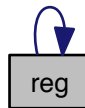
### 5.2.4 Function Documentation

#### 5.2.4.1 `int reg ( int i )`

**See Also**

intersection.cpp

Here is the call graph for this function:

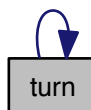
**5.2.4.2 int turn ( dir *globalDir*, int *QDirection* )**

Returns routing address for Vehicle

**See Also**

intersection.cpp

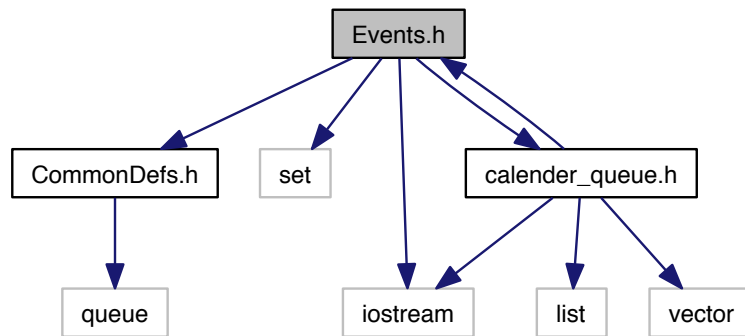
Here is the call graph for this function:

**5.3 Events.h File Reference**

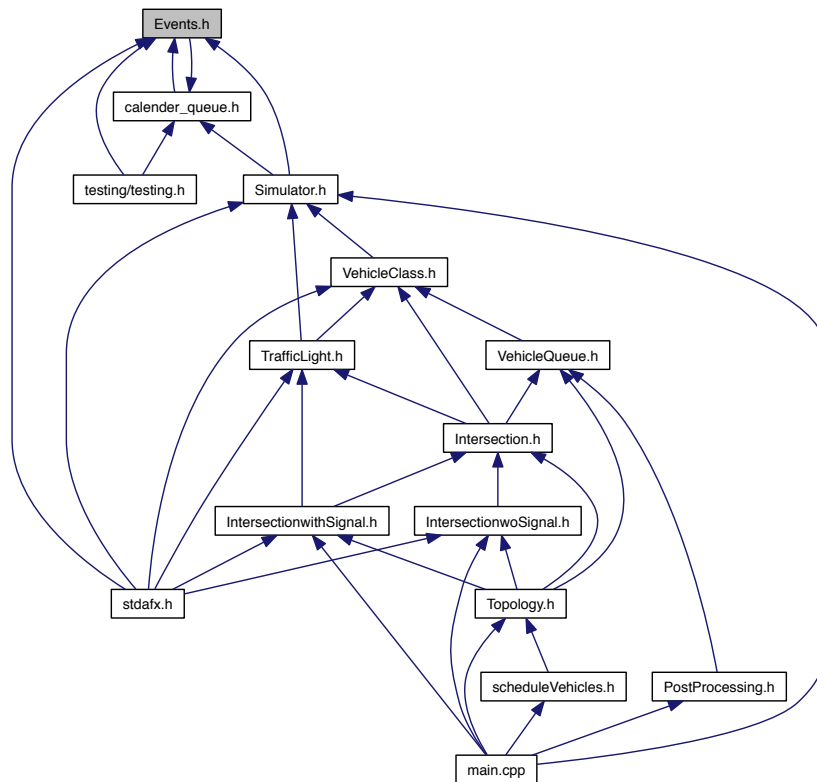
declaration of various types of events

```
#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](#)

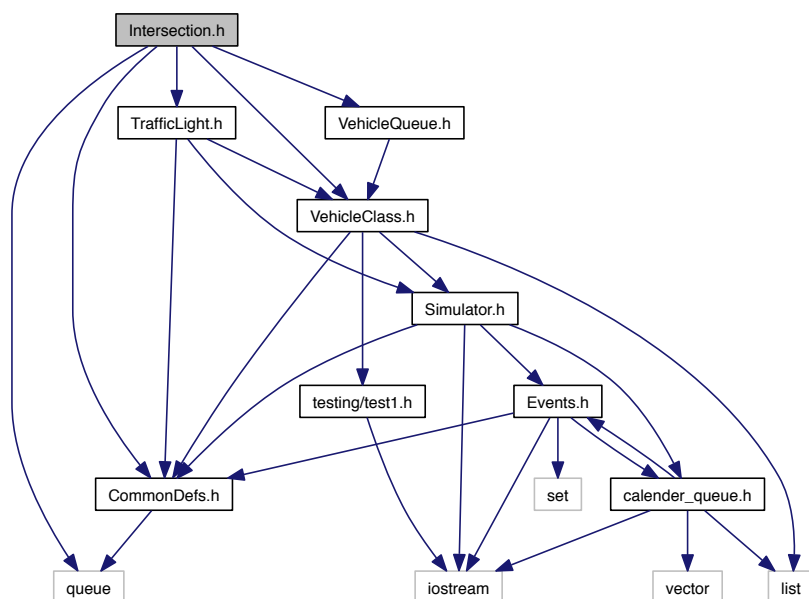
### 5.3.1 Detailed Description

declaration of various types of events

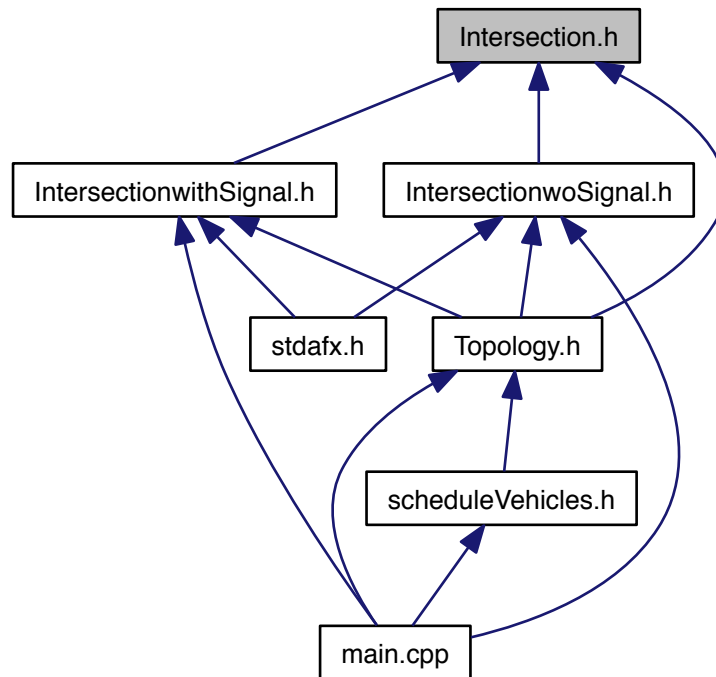
## 5.4 Intersection.h File Reference

```
#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](#)

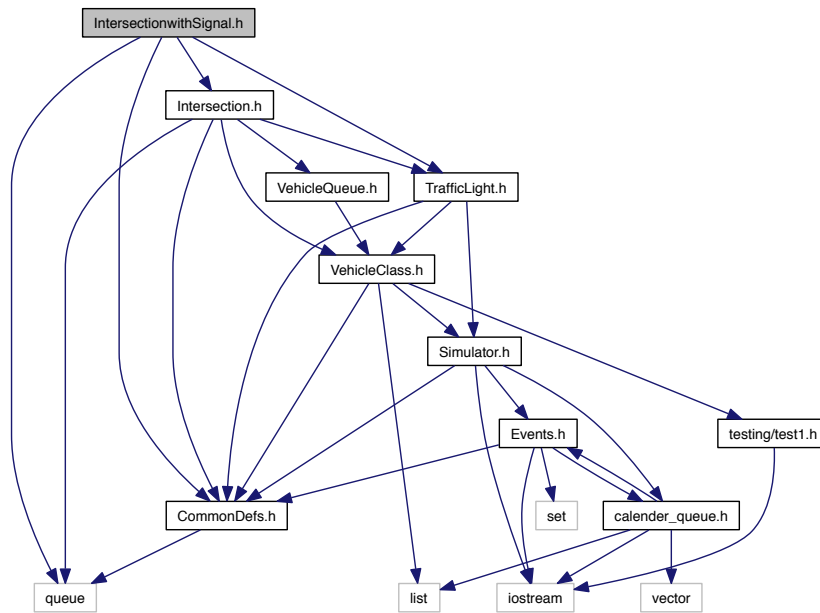
## 5.5 IntersectionwithSignal.h File Reference

```

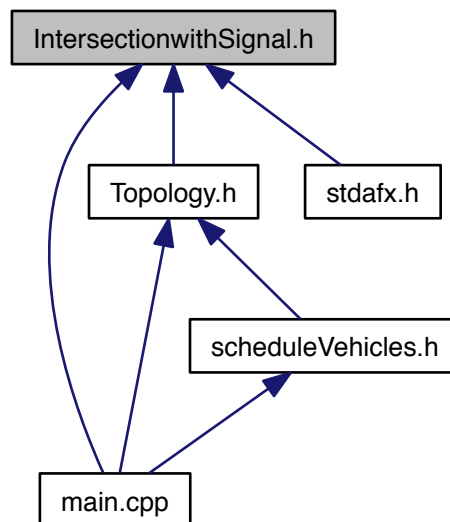
#include <queue>
#include "CommonDefs.h"
#include "TrafficLight.h"
#include "Intersection.h"

```

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

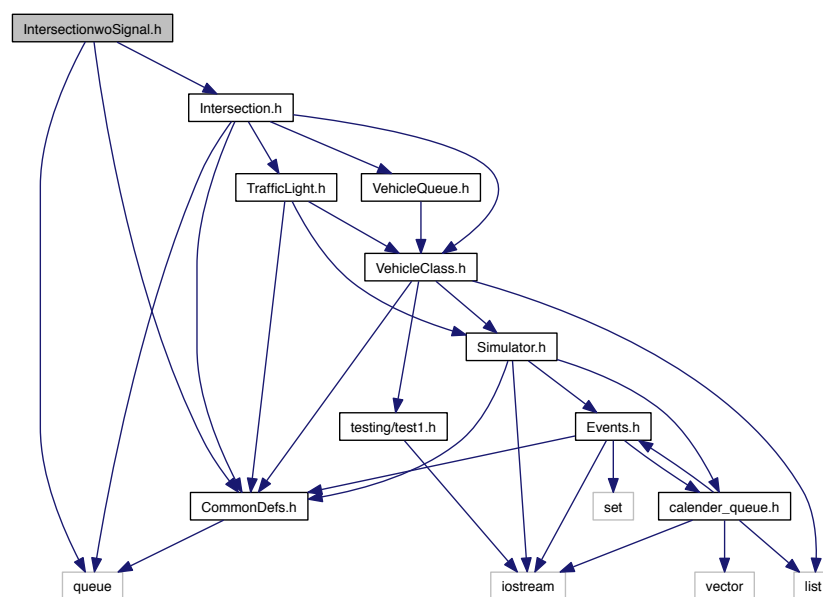
See Also

[Intersection.h](#)

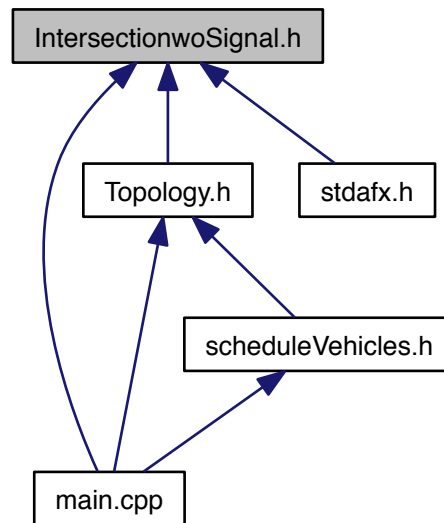
## 5.6 IntersectionwoSignal.h File Reference

```
#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

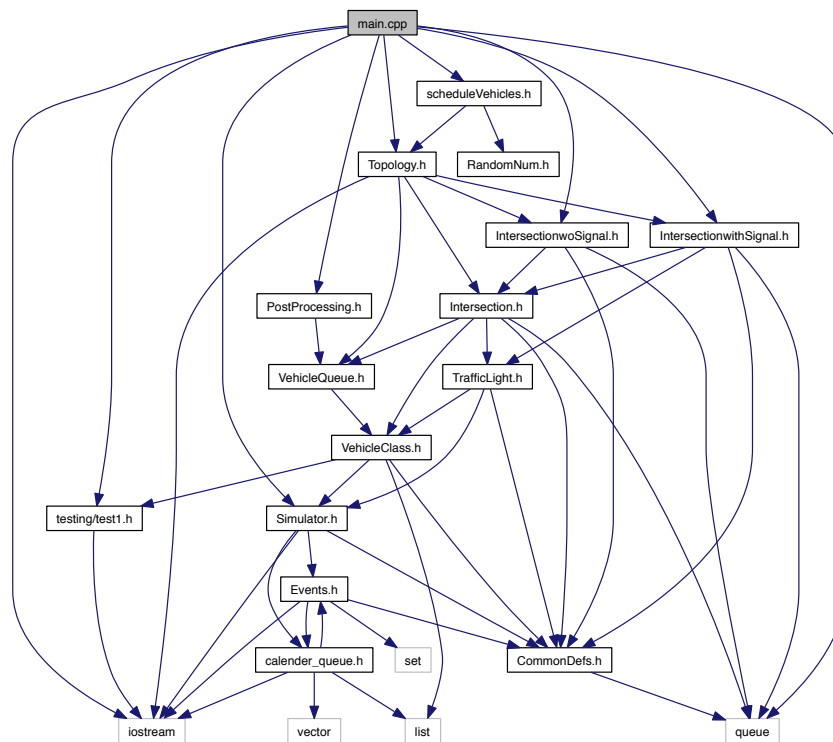
```

#include <iostream>
#include "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 ()`

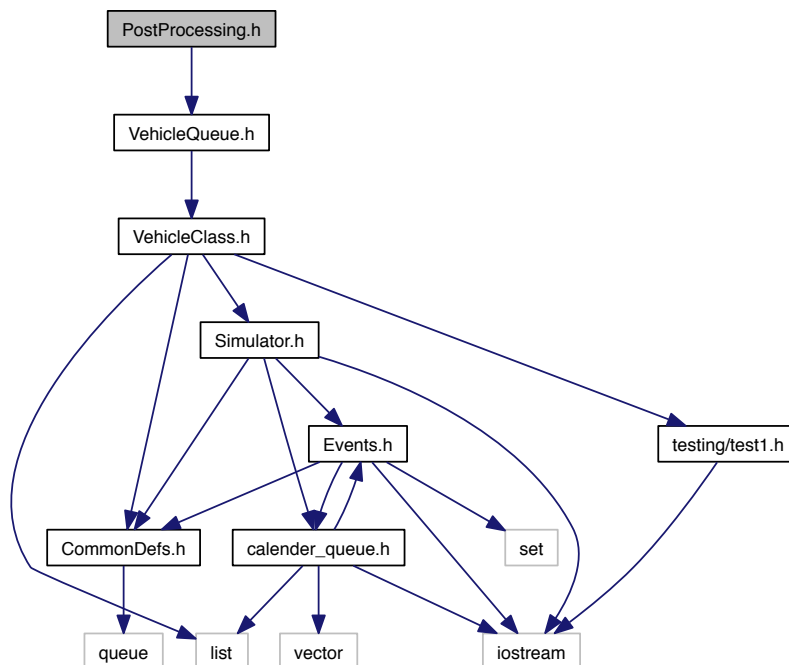
## Variables

- `Simulator * sim = new Simulator()`

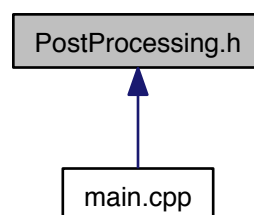
## 5.8 PostProcessing.h File Reference

```
#include "VehicleQueue.h"
```

Include dependency graph for PostProcessing.h:



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



### Functions

- void [PostProcStats](#) ([VehicleQueue](#) \*EQ, double timeval, int buckets, int source, int dest)

#### 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

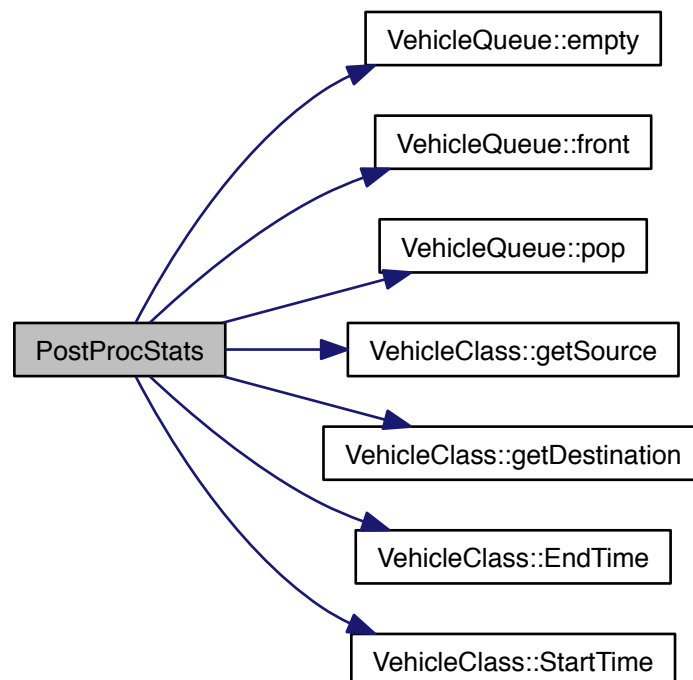
### 5.8.2.1 void PostProcStats ( VehicleQueue \* EQ, double timeval, int buckets, int source, int dest )

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

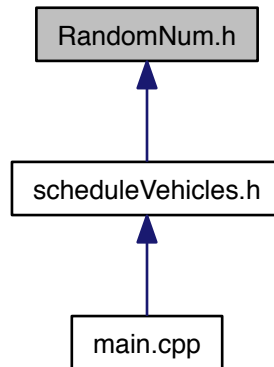
<i>EQ</i>	is exit Q
<i>buckets</i>	is number of buckets for histogram
<i>timeval</i>	is the period of time which is divided into buckets
<i>source</i>	is starting point of the journey
<i>dest</i>	is input for describing

Here is the call graph for this function:



## 5.9 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](#) ()

#### 5.9.1 Detailed Description

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

#### 5.9.2 Macro Definition Documentation

##### 5.9.2.1 #define MINSTDG 16807

Multiplier for random number genrator

##### 5.9.2.2 #define MINSTDM 2147483647

Modulus for random number genrator

## 5.9.2.3 #define MINSTDX 1

Default starting state

## 5.10 RoadSegment.h File Reference

## Classes

- class [RoadSegment](#)

## 5.10.1 Detailed Description

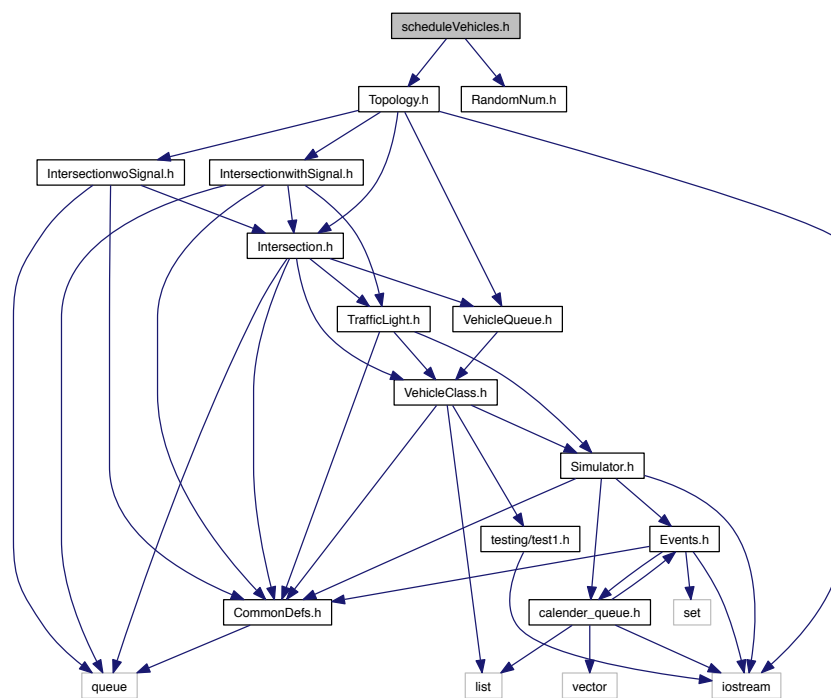
Define a segment of the Road

## 5.11 scheduleVehicles.h File Reference

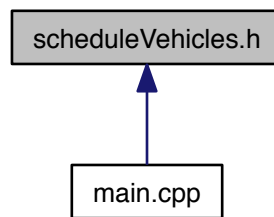
```
#include "Topology.h"
```

```
#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)

### 5.11.1 Detailed Description

It initializes the scheduling of vehicle during the simulation

### 5.11.2 Function Documentation

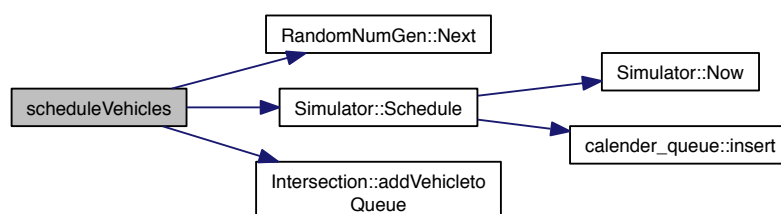
#### 5.11.2.1 void `scheduleVehicles` ( `_Topology` \* *Topology*, double *maxTime* )

It initializes the scheduling of vehicle during the simulation

#### Parameters

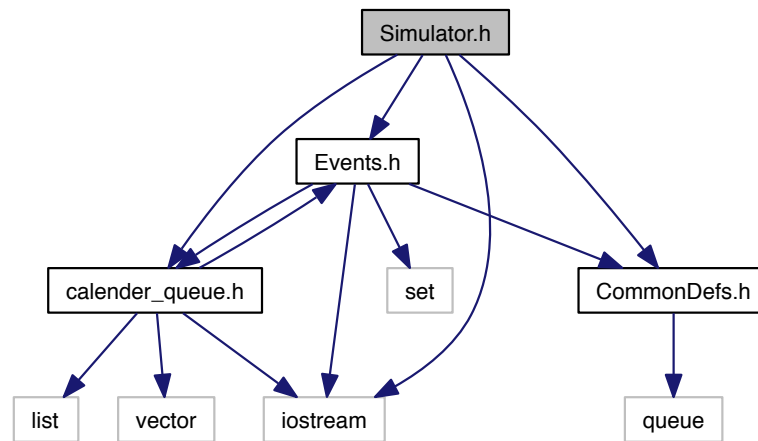
<i>Topology</i>	of the westpeachtree street
<i>max-Time, maximum</i>	time of till which we have to schedule vehicles

Here is the call graph for this function:

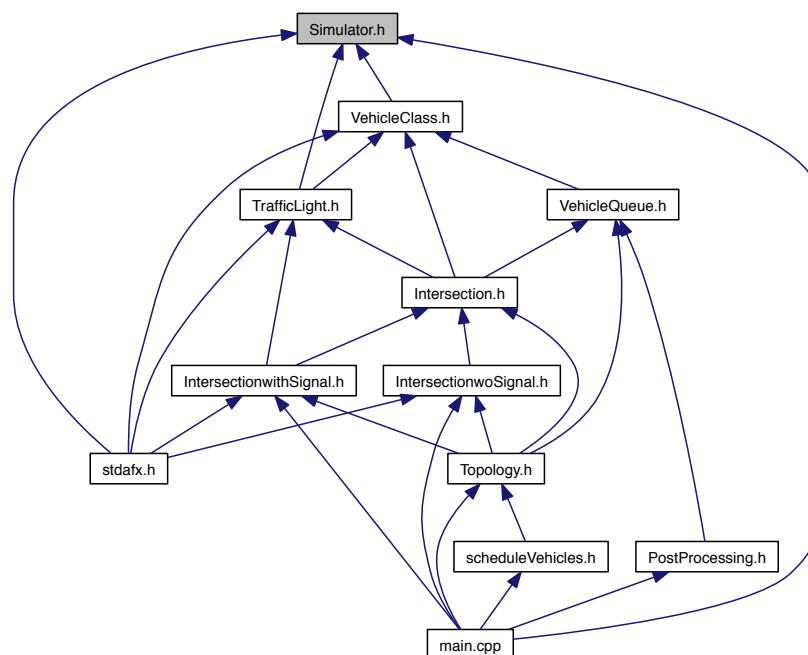


## 5.12 Simulator.h File Reference

```
#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**

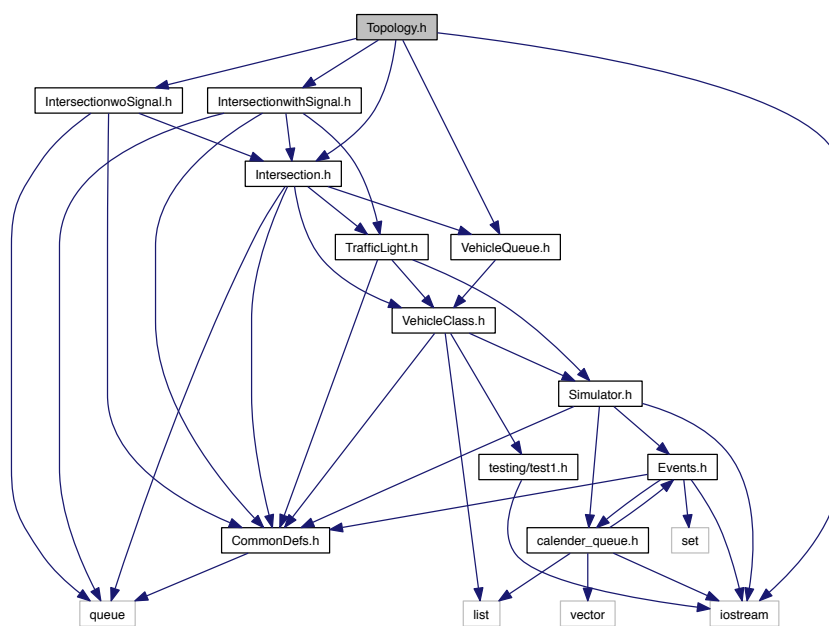
### 5.12.1 Detailed Description

Contains description of [Simulator](#) class and various functions of simulator class

## 5.13 Topology.h File Reference

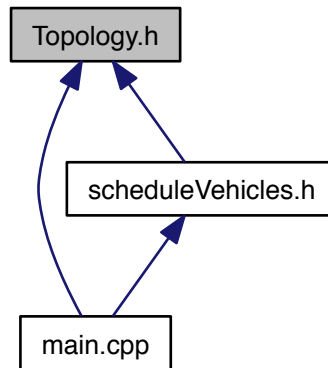
```
#include <iostream>
#include "Intersection.h"
#include "IntersectionwithSignal.h"
#include "IntersectionwoSignal.h"
#include "VehicleQueue.h"
```

Include dependency graph for Topology.h:





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



## Classes

- [class `\_Topology`](#)

### 5.13.1 Detailed Description

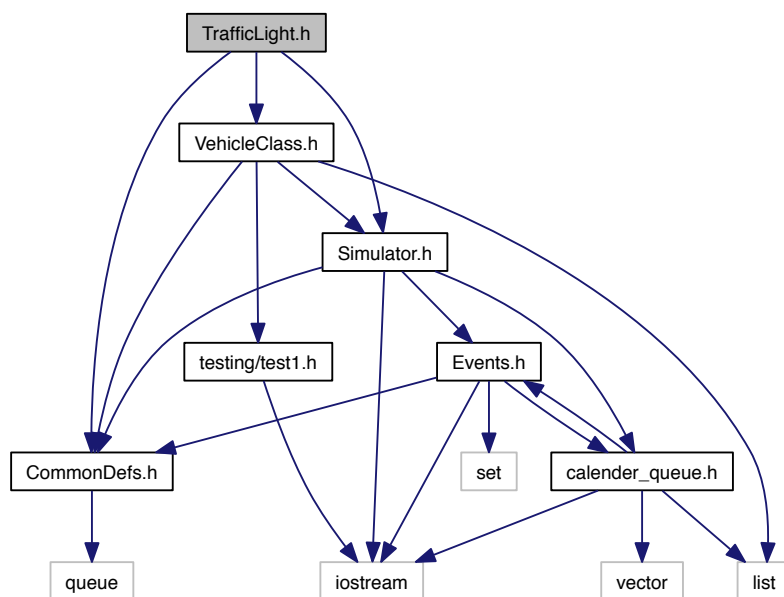
To describe the topology of the street to be simulated i.e. peachtree street for this project

## 5.14 TrafficLight.h File Reference

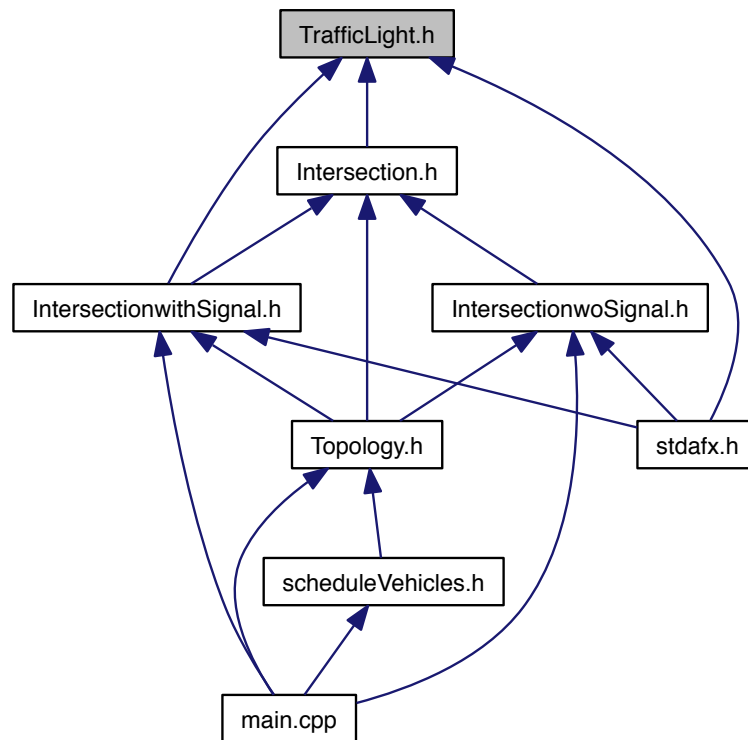
description of functionality of traffic light

```
#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**

### 5.14.1 Detailed Description

description of functionality of traffic light

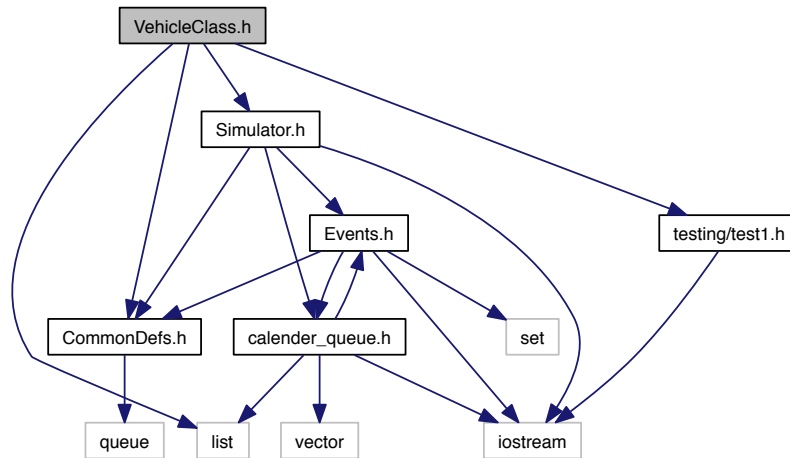
## 5.15 VehicleClass.h File Reference

```

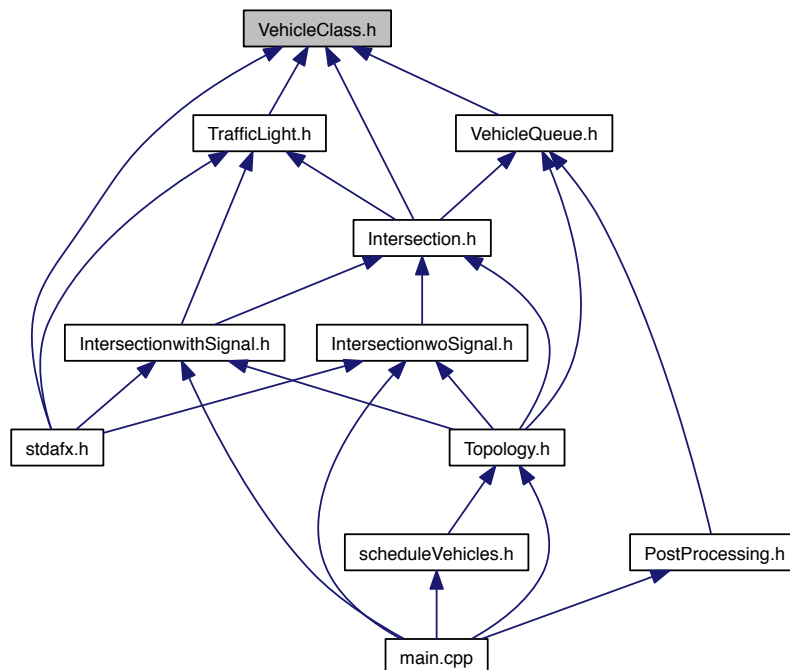
#include "CommonDefs.h"
#include "testing/test1.h"
#include <list>
#include "Simulator.h"

```

Include dependency graph for VehicleClass.h:



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



## Classes

- class [VehicleClass](#)

## Variables

- [Simulator](#) \* **sim**

### 5.15.1 Detailed Description

Contains description of vehicle class

# Index

- ~Intersection
  - Intersection, [18](#)
- ~IntersectionwithSignal
  - IntersectionwithSignal, [28](#)
- ~IntersectionwithoutSignal
  - IntersectionwithoutSignal, [24](#)
- ~RandomNumGen
  - RandomNumGen, [31](#)
- ~TrafficLight
  - TrafficLight, [37](#)
- ~VehicleClass
  - VehicleClass, [39](#)
- \_Topology, [7](#)
  - \_Topology, [8](#)
  - \_Topology, [8](#)
  - ExitQ, [8](#)
  - I1, [8](#)
  - I2, [8](#)
  - I3, [8](#)
  - I4, [8](#)
  - I5, [8](#)
- AddVehicle
  - RoadSegment, [32](#)
- addVehicletoQueue
  - Intersection, [18](#)
  - IntersectionwithoutSignal, [25](#)
  - IntersectionwithSignal, [28](#)
- BUCKET\_COUNT
  - calender\_queue.h, [44](#)
- back
  - VehicleQueue, [41](#)
- BurstTime
  - CommonDefs.h, [47](#)
- busy
  - Intersection, [21](#)
  - VehicleQueue, [41](#)
- CALENDER\_PERIOD
  - calender\_queue.h, [45](#)
- calender\_queue, [9](#)
  - calender\_queue, [9](#)
  - calender\_queue, [9](#)
  - check659bucket, [9](#)
  - dequeue, [9](#)
  - get\_bucket\_count, [9](#)
  - getQsize, [9](#)
  - gettimeframe, [9](#)
  - insert, [10](#)
  - isEmpty, [10](#)
  - next\_event, [10](#)
  - PopNext, [10](#)
  - remove\_event, [10](#)
- calender\_queue.h, [43](#)
  - BUCKET\_COUNT, [44](#)
  - CALENDER\_PERIOD, [45](#)
  - TOTAL\_TIME, [45](#)
- changeSignalTrigger
  - IntersectionwithSignal, [28](#)
- check659bucket
  - calender\_queue, [9](#)
- checkQinterval
  - CommonDefs.h, [47](#)
- CommonDefs.h, [45](#)
  - BurstTime, [47](#)
  - checkQinterval, [47](#)
  - LPassTime, [47](#)
  - PassTime, [47](#)
  - reg, [47](#)
  - roadSegTime, [47](#)
  - startToPass, [47](#)
  - Time\_t, [47](#)
  - turn, [48](#)
- cyclestate
  - TrafficLight, [37](#)
- dequeue
  - calender\_queue, [9](#)
- EB
  - IntersectionwithSignal, [29](#)
- EBI1
  - Intersection, [21](#)
- EBI2
  - Intersection, [21](#)
- empty
  - VehicleQueue, [41](#)
- EndTime
  - VehicleClass, [39](#)
- endTime
  - VehicleClass, [40](#)
- Event0< T, OBJ >, [11](#)
- Event1< T, OBJ, U1, T1 >, [12](#)
- Event2< T, OBJ, U1, T1, U2, T2 >, [13](#)
- Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >, [14](#)
- event\_compare, [15](#)
- EventBase, [15](#)
- eventDsc, [16](#)
- Events.h, [48](#)

- EvictQ
  - Intersection, [18](#)
- EvictVehicle
  - RoadSegment, [32](#)
- ExitQ
  - \_Topology, [8](#)
  - Intersection, [22](#)
- front
  - VehicleQueue, [41](#)
- get\_bucket\_count
  - calender\_queue, [9](#)
- getDestination
  - VehicleClass, [39](#)
- getDirection
  - VehicleClass, [39](#)
- getID
  - Intersection, [19](#)
  - VehicleClass, [39](#)
- getLastQ
  - VehicleClass, [39](#)
- getLeftState
  - TrafficLight, [37](#)
- GetLen
  - VehicleQueue, [41](#)
- GetMaxLen
  - VehicleQueue, [41](#)
- getNow
  - Simulator, [33](#)
- getQdirection
  - Intersection, [19](#)
- getQlane
  - Intersection, [19](#)
- getQsize
  - calender\_queue, [9](#)
- getSource
  - VehicleClass, [39](#)
- GetState
  - RandomNumGen, [31](#)
- getState
  - TrafficLight, [37](#)
- getType
  - TrafficLight, [37](#)
- gettimeframe
  - calender\_queue, [9](#)
- haveSignal
  - Intersection, [22](#)
- I1
  - \_Topology, [8](#)
- I2
  - \_Topology, [8](#)
- I3
  - \_Topology, [8](#)
- I4
  - \_Topology, [8](#)
- I5
  - \_Topology, [8](#)
- ID
  - Intersection, [22](#)
- insert
  - calender\_queue, [10](#)
- instance
  - Simulator, [36](#)
- Intersection, [16](#)
  - ~Intersection, [18](#)
  - addVehicleToQueue, [18](#)
  - busy, [21](#)
  - EBI1, [21](#)
  - EBI2, [21](#)
  - EvictQ, [18](#)
  - ExitQ, [22](#)
  - getID, [19](#)
  - getQdirection, [19](#)
  - getQlane, [19](#)
  - haveSignal, [22](#)
  - ID, [22](#)
  - Intersection, [18](#)
  - NBI1, [22](#)
  - NBI2, [22](#)
  - NInter, [22](#)
  - NextQInfo, [19](#)
  - QCanGo, [20](#)
  - routingtable, [22](#)
  - SBI1, [22](#)
  - SBI2, [22](#)
  - SInter, [22](#)
  - VehicleDeparture, [20](#)
  - VehiclePass, [21](#)
  - WBI1, [22](#)
  - WBI2, [22](#)
- Intersection.h, [50](#)
- IntersectionwithSignal, [26](#)
  - ~IntersectionwithSignal, [28](#)
  - addVehicleToQueue, [28](#)
  - changeSignalTrigger, [28](#)
  - EB, [29](#)
  - IntersectionwithSignal, [28](#)
  - IntersectionwithSignal, [28](#)
  - NB, [29](#)
  - QCanGo, [29](#)
  - SB, [30](#)
  - WB, [30](#)
- IntersectionwithSignal.h, [51](#)
- IntersectionwithoutSignal, [23](#)
  - ~IntersectionwithoutSignal, [24](#)
  - addVehicleToQueue, [25](#)
  - IntersectionwithoutSignal, [24](#)
  - IntersectionwithoutSignal, [24](#)
  - QCanGo, [25](#)
- IntersectionwoSignal.h, [53](#)
- isBusy
  - VehicleQueue, [41](#)
- isEmpty
  - calender\_queue, [10](#)

- LPassTime
  - CommonDefs.h, 47
- LastSentCar
  - VehicleQueue, 41
- MINSTDG
  - RandomNum.h, 58
- MINSTDM
  - RandomNum.h, 58
- MINSTDX
  - RandomNum.h, 58
- main.cpp, 54
- myid
  - TrafficLight, 38
- NB
  - IntersectionwithSignal, 29
- NBI1
  - Intersection, 22
- NBI2
  - Intersection, 22
- NInter
  - Intersection, 22
- Next
  - RandomNumGen, 31
- next\_event
  - calender\_queue, 10
- NextQInfo
  - Intersection, 19
- Now
  - Simulator, 33
- PassTime
  - CommonDefs.h, 47
- pop
  - VehicleQueue, 41
- PopNext
  - calender\_queue, 10
- PostProcStats
  - PostProcessing.h, 57
- PostProcessing.h, 56
  - PostProcStats, 57
- prioqueue, 30
- push
  - VehicleQueue, 41
- Q1
  - VehicleQueue, 41
- QCanGo
  - Intersection, 20
  - IntersectionwithoutSignal, 25
  - IntersectionwithSignal, 29
- qlleft
  - RoadSegment, 32
- qright
  - RoadSegment, 32
- RandomNum.h, 58
  - MINSTDG, 58
  - MINSTDM, 58
  - MINSTDX, 58
- RandomNumGen, 30
  - ~RandomNumGen, 31
  - GetState, 31
  - Next, 31
  - RandomNumGen, 30
  - RandomNumGen, 30
  - Reset, 31
- reg
  - CommonDefs.h, 47
- remove\_event
  - calender\_queue, 10
- Reset
  - RandomNumGen, 31
- roadSegTime
  - CommonDefs.h, 47
- RoadSegment, 31
  - AddVehicle, 32
  - EvictVehicle, 32
  - qlleft, 32
  - qright, 32
  - RoadSegment, 32
  - RoadSegment, 32
- RoadSegment.h, 59
- routingtable
  - Intersection, 22
- Run
  - Simulator, 34
- SB
  - IntersectionwithSignal, 30
- SBI1
  - Intersection, 22
- SBI2
  - Intersection, 22
- SInter
  - Intersection, 22
- Schedule
  - Simulator, 34, 35
- scheduleVehicles
  - scheduleVehicles.h, 60
- scheduleVehicles.h, 59
  - scheduleVehicles, 60
- setEndTime
  - VehicleClass, 39
- setLastQ
  - VehicleClass, 39
- Simulator, 33
  - getNow, 33
  - instance, 36
  - Now, 33
  - Run, 34
  - Schedule, 34, 35
  - Simulator, 33
  - Stop, 35
  - StopAt, 35
- Simulator.h, 61
- StartTime



- VehicleClass, 39
- startTime
  - VehicleClass, 40
- startToPass
  - CommonDefs.h, 47
- Stop
  - Simulator, 35
- StopAt
  - Simulator, 35
- TOTAL\_TIME
  - calender\_queue.h, 45
- Time\_t
  - CommonDefs.h, 47
- Topology.h, 62
- TrafficLight, 36
  - ~TrafficLight, 37
  - cyclestate, 37
  - getLeftState, 37
  - getState, 37
  - getType, 37
  - myid, 38
  - TrafficLight, 36
  - TrafficLight, 36
  - type, 38
- TrafficLight.h, 63
- turn
  - CommonDefs.h, 48
- type
  - TrafficLight, 38
- updateDirection
  - VehicleClass, 40
- VehicleClass, 38
  - ~VehicleClass, 39
  - EndTime, 39
  - endTime, 40
  - getDestination, 39
  - getDirection, 39
  - getID, 39
  - getLastQ, 39
  - getSource, 39
  - setEndTime, 39
  - setLastQ, 39
  - StartTime, 39
  - startTime, 40
  - updateDirection, 40
  - VehicleClass, 38
  - VehicleClass, 38
- VehicleClass.h, 65
- VehicleDeparture
  - Intersection, 20
- VehiclePass
  - Intersection, 21
- VehicleQueue, 40
  - back, 41
  - busy, 41
  - empty, 41
  - front, 41
  - GetLen, 41
  - GetMaxLen, 41
  - isBusy, 41
  - LastSentCar, 41
  - pop, 41
  - push, 41
  - Q1, 41
  - VehicleQueue, 40
  - VehicleQueue, 40
- WB
  - IntersectionwithSignal, 30
- WBI1
  - Intersection, 22
- WBI2
  - Intersection, 22