CPE 400: Distributed Transport Network

1.0

Generated by Doxygen 1.8.12

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 Intersect1 Class Reference

`#include <intersect.h>`

Public Member Functions

- Intersect1 ()
- Intersect1 (char id, bool holdsAsterisk, VehicleDir dir)
- ~Intersect1 ()
- void setIntersection (char id, bool holdsAsterisk, VehicleDir dir)
- std::string operator[ ] (int row)
- std::string getIntersection (int row)

Static Public Attributes

- static const int NUM_ROWS = 5
- static const std::string UPPER_SCORE = "\u23ba"
- static const std::string VERT_ROADWAY [5] = { "_", " ", " ", " ", UPPER_SCORE }
- static const std::string HORI_ROADWAY [1] = { "" }

Private Member Functions

- void constructIntersection ()

Private Attributes

- char objId
- bool hasAsterisk
- VehicleDir objDir
- std::string intersection [5]

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Intersect1() [1/2]

```
Intersect1::Intersect1 ( )
```

4.1.1.2 Intersect1() [2/2]

```
Intersect1::Intersect1 (
            char id,
            bool holdsAsterisk,
            VehicleDir dir )
```

4.1.1.3 ~Intersect1()

```
Intersect1::~Intersect1 ( )
```

4.1.2 Member Function Documentation

4.1.2.1 constructIntersection()

```
void Intersect1::constructIntersection ( )  [private]
```

4.1.2.2 getIntersection()

```
string Intersect1::getIntersection (
            int row )
```

4.1.2.3 operator[]()

```
string Intersect1::operator[] (
            int row )
```

4.1.2.4 setIntersection()

```
void Intersect1::setIntersection (
            char id,
            bool holdsAsterisk,
            VehicleDir dir )
```

4.1.3 Field Documentation

4.1.3.1 hasAsterisk

```
bool Intersect1::hasAsterisk  [private]
```

4.1.3.2 HORI_ROADWAY

```
const string Intersect1::HORI_ROADWAY = { "" }  [static]
```

4.1.3.3 intersection

```
std::string Intersect1::intersection[5]  [private]
```

4.1.3.4 NUM_ROWS

```
const int Intersect1::NUM_ROWS = 5  [static]
```

4.1.3.5 objDir

```
VehicleDir Intersect1::objDir  [private]
```

4.1.3.6 objId

```
char Intersect1::objId  [private]
```

4.1.3.7 UPPER_SCORE

```
const string Intersect1::UPPER_SCORE = "\u23ba"  [static]
```

4.1.3.8 VERT_ROADWAY

```
const string Intersect1::VERT_ROADWAY = { "_", " ", " ", " ", UPPER_SCORE }  [static]
```

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

- include/intersect.h
- src/intersect.cpp

4.2 Packet Struct Reference

```
#include <vehicle.h>
```

Data Fields

- int packetId
- int srcId
- int destX = 0
- int destY = 0
- int destId
- int srcX
- int srcY
- int age
- std::string message
- std::vector< int > ids
- bool thrown = false
- bool atDest = false

4.2.1 Field Documentation

4.2.1.1 age

```
int Packet::age
```

4.2.1.2 atDest

```
bool Packet::atDest = false
```

4.2.1.3 destId

```
int Packet::destId
```

4.2.1.4 destX

```
int Packet::destX = 0
```

4.2.1.5 destY

```
int Packet::destY = 0
```

4.2.1.6 ids

```
std::vector<int> Packet::ids
```

4.2.1.7 message

```
std::string Packet::message
```

4.2.1.8 packetId

```
int Packet::packetId
```

4.2.1.9 srcId

```
int Packet::srcId
```

4.2.1.10 srcX

```
int Packet::srcX
```

4.2.1.11 srcY

```
int Packet::srcY
```

4.2.1.12 thrown

```
bool Packet::thrown = false
```

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

- include/vehicle.h

4.3 Simulator Class Reference

```
#include <simulator.h>
```

Public Member Functions

- Simulator ()
  Simulator Constructor.
- ~Simulator ()
  Simulator Destructor.
- void run ()
  Run.
- char displayMenu ()
  Display Menu.
- void initWorld (int sizeX, int sizeY)
  Initialize world.
- bool moveVehicle (int xCoorFrom, int yCoorFrom, int xCoorTo, int yCoorTo)
- void displayWorld ()

Private Attributes

- World< Taxi > world

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Simulator()

```
Simulator::Simulator ( )
```

Simulator Constructor.

Constructs the Simulator Class object

Precondition

None

Postcondition

Simulator class object called

Algorithm

None

Exceptions

None	

Parameters

None	

Returns

None

Note

None

4.3.1.2 ~Simulator()

```
Simulator::~Simulator ( )
```

Simulator Destructor.

Implicitly destructs all data structures

Precondition

Assume initialized class object

Postcondition

None

Algorithm

Simulator class data member destructors called implicitely upon deletion of the simulator class

Exceptions

None	

Parameters

None	

Returns

None

Note

None

4.3.2 Member Function Documentation

4.3.2.1 displayMenu()

```
char Simulator::displayMenu ( )
```

Display Menu.

Prints the list of available commands

Precondition

None

Postcondition

Displays the menu

Algorithm

Standard I/O stream operations

Exceptions

None

Parameters

None

Returns

Char

Note

None

4.3.2.2 displayWorld()

```
void Simulator::displayWorld ( )
```

4.3.2.3 initWorld()

```
void Simulator::initWorld (
            int sizeX,
            int sizeY )
```

Initialize world.

Dynamically constructs the world with given parameters

Precondition

Assume initialized world object

Postcondition

World initialized

Algorithm

Call the world class initWorld function

Exceptions

None

Parameters

in	*sizeX*	Max x-axis world size to be set

[in] sizeY Max y-axos world size to be set

Returns

None

Note

None

4.3.2.4 moveVehicle()

```
bool Simulator::moveVehicle (
            int xCoorFrom,
            int yCoorFrom,
            int xCoorTo,
            int yCoorTo )
```

4.3.2.5 run()

```
void Simulator::run ( )
```

Run.

Runs the simulation

Precondition

None

Postcondition

Simulation ran

Algorithm

Run the simulation in a loop executing commands from the user until the user terminates the program

Exceptions

None	

Parameters

Parameters

None	

Returns

None

Note

Essentially the main driver

4.3.3 Field Documentation

4.3.3.1 world

```
World<Taxi> Simulator::world  [private]
```

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

- include/simulator.h
- src/simulator.cpp

4.4 Taxi Class Reference

```
#include <vehicle.h>
```

Inheritance diagram for Taxi:

Public Member Functions

- Taxi (int x, int y, int rowMax, int colMax, bool hasPkt=false)

  Constructor for Taxi class derived from abstrat Vehicle class.
- char getId ()
- bool inTransition ()

  In Transisition.
- void calculateDestination ()

  Calculates Taxi's destination.

Private Attributes

- int ticksToMove
- int tickCounter

Additional Inherited Members

4.4.1 Constructor & Destructor Documentation

4.4.1.1 Taxi()

```
Taxi::Taxi (
            int x,
            int y,
            int rowMax,
            int colNum,
            bool hasPkt = false )
```

Constructor for Taxi class derived from abstrat Vehicle class.

Calculates destination and next location

Precondition

None

Postcondition

None

Algorithm

Calculates destination Calculates next location

Exceptions

None	

Parameters

in	*x*	X position of vehicle

[in] y Y position of vehicle

[in] rowMax Number of rows allowed

[in] colNum Number of columns allowed

[in] hasPkt Contains packet

Note

None

4.4.2 Member Function Documentation

4.4.2.1 calculateDestination()

```
void Taxi::calculateDestination ( )  [virtual]
```

Calculates Taxi's destination.

Randomly assigns destination within given row and column boundaries using rand() generator

Precondition

number of columns $>$ 0 number of rows $>$ 0

Postcondition

New destination != old location

Algorithm

Generate random xy coordinates until coordinates != old destination

Exceptions

None	

Parameters

None	

Returns

None

Note

Recommend setting current location coordinates = old destination before call

Implements Vehicle.

4.4.2.2 getId()

```
char Taxi::getId ( )  [inline], [virtual]
```

Reimplemented from Vehicle.

4.4.2.3 inTransition()

```
bool Taxi::inTransition ( )
```

In Transisition.

Checks if the vehicle is in transition between interesections and pseudo-moves it a tick closer to the next intersection

Precondition

None

Postcondition

Returns a bool stating if the vehicle is in transition and pseudo-moves it a tick closer to the next intersection

Algorithm

Check to see if tick counter is less than ticks to move, and if so increments the tick counter and returns true, otherwise, reset the tick counter and return false

Exceptions

None	

Parameters

None	

Note

None

4.4.3 Field Documentation

4.4.3.1 tickCounter

```
int Taxi::tickCounter  [private]
```

4.4.3.2 ticksToMove

```
int Taxi::ticksToMove  [private]
```

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

- include/vehicle.h
- src/vehicle.cpp

4.5 Vehicle Class Reference

`#include <vehicle.h>`

Inheritance diagram for Vehicle:

Public Member Functions

- Vehicle (int x, int y, int rowNum, int colNum, bool hasPkt=false)

  Constructor for abstract Vehicle class.
- ~Vehicle ()
- virtual char getId ()
- void getLocation (int &x, int &y)
- void getNextLocation (int &x, int &y)
- void getDestination (int &x, int &y)
- int getPacketSize ()
- VehicleDir getDirection ()
- bool hasPacket ()
- void move ()

  Moves vehicle.
- void redirect ()

  Redirects the vehicle.
- void setPacket (bool holdsPacket)
- bool throwPacket (Vehicle *targetVehicle, Packet thrownPacket, bool update=false)

  Throws packet to the passed vehicle and ouptputs a successful transfer message.
- bool packetCaught (Packet thrownPacket)

  checks to see if passed paket has reacched its destination or if it can be taken in
- bool updatePacketCaught (Packet thrownPacket)

  checks to see if passed paket has reacched its destination or if it can be taken in
- bool vehicleRun ()

  throws packets if they exist
- bool bestDestinationAlgorithm ()

  searches for best vehicle to throw to
- void updateLocation ()
- int getVehicleId () const

  gets vehicle id

Data Fields

- std::vector< Packet * > packets
- std::vector< Packet * > updates
- Vehicle * nearByVehicles [8]

Protected Member Functions

- void calcNextLocation ()

  Calculates vehicles's next location.
- bool calcAltDirection ()

  Calculates an alterative direction.
- void stop ()

  Stop vehicle.
- bool planUp ()

  Sets next location and direction to go up.
- bool planDown ()

  Sets next location and direction to go down.
- bool planLeft ()

  Sets next location and direction to go left.
- bool planRight ()

  Sets next location and direction to go right.
- virtual void calculateDestination ()=0

Protected Attributes

- int xPos
- int yPos
- int xDest
- int yDest
- int xNextPos
- int yNextPos
- bool hasPkt
- bool hasUpdate
- int vehicleId
- Packet * newPacket
- unsigned int redirectCounter
- VehicleDir vehicleDir
- std::vector< vehicleLocation > locations
- int rowMax
- int colMax

Static Protected Attributes

- static int vehicleCount = 0

4.5.1 Constructor & Destructor Documentation

4.5.1.1 Vehicle()

```
Vehicle::Vehicle (
            int x,
            int y,
            int rowMax,
            int columnMax,
            bool hasPkt = false )
```

Constructor for abstract Vehicle class.

Sets up class member variables. Destination set to given position. recommend calling setDestination() from derived class constructors.

Precondition

None

Postcondition

None

Algorithm

Sets destination to given position

Exceptions

4.5.1.2 ~Vehicle()

```
Vehicle::~Vehicle ( )  [inline]
```

4.5.2 Member Function Documentation

4.5.2.1 bestDestinationAlgorithm()

```
bool Vehicle::bestDestinationAlgorithm ( )
```

searches for best vehicle to throw to

gives each vehicle in proximity of current vehicle a score based on how well the vehicle will get the packet to the appropriate location. This score is based on the destination of the target vehicle, the direction the vehicles are moving and where the vehicle is going(destination)

Precondition

called from vehicle Run function

Postcondition

Packet is thrown to ideal vehicle and packet is removed from vehicle

None

Exceptions

None	

Parameters

None	

Returns

Returns true if packet is thrown and false if packet is kept

Note

None

4.5.2.2 calcAltDirection()

```
bool Vehicle::calcAltDirection ( )  [protected]
```

Calculates an alterative direction.

Determines an optimal alterative way to the destination ignoring the possiblity of going fowards or backward

Precondition

None

Postcondition

Vehicle direction not forward or backward; next position is modified

Algorithm

Find optimal 90 degree alternate direction If no other direction possible, then direction set to NaN

Exceptions

None	

Parameters

None	

Returns

Whether vehicle is boxed in on its sides

Note

If vehicle is boxed in on both sides left/right or up/down, then stops vehicle

4.5.2.3 calcNextLocation()

```
void Vehicle::calcNextLocation ( )  [protected]
```

Calculates vehicles's next location.

Using current location, next location, and direction, recalculates the location the car should be next

Precondition

None

Postcondition

None

Algorithm

Based on the direction, move towards destination along x axis first If wanted direction is behind, then make alternate turn If direction set to NaN, then any turn is possible

Exceptions

None	

Parameters

None	

Returns

None

Note

Does not allow immediate U-turns. If no other options are available two move() calls are required to complete a U-turn. If direction set to NaN, then any turn is possible

4.5.2.4 calculateDestination()

```
virtual void Vehicle::calculateDestination ( )  [protected], [pure virtual]
```

Implemented in Taxi.

4.5.2.5 getDestination()

```
void Vehicle::getDestination (
            int & x,
            int & y )  [inline]
```

4.5.2.6 getDirection()

```
VehicleDir Vehicle::getDirection ( )  [inline]
```

4.5.2.7 getId()

```
virtual char Vehicle::getId ( )  [inline], [virtual]
```

Reimplemented in Taxi.

4.5.2.8 getLocation()

```
void Vehicle::getLocation (
            int & x,
            int & y )  [inline]
```

4.5.2.9 getNextLocation()

```
void Vehicle::getNextLocation (
            int & x,
            int & y )  [inline]
```

4.5.2.10 getPacketSize()

```
int Vehicle::getPacketSize ( )  [inline]
```

4.5.2.11 getVehicleId()

```
int Vehicle::getVehicleId ( ) const
```

gets vehicle id

Returns id

Precondition

None

Postcondition

None

None

Exceptions

None	

Parameters

None	

Returns

vehicle id (int)

Note

None

4.5.2.12 hasPacket()

```
bool Vehicle::hasPacket ( )  [inline]
```

4.5.2.13 move()

```
void Vehicle::move ( )
```

Moves vehicle.

Sets current location to next location and recalcuates next location

Precondition

None

Postcondition

Current location, next location, and direction are updated

Algorithm

Set current location to next location If current location is destination, then change destination Calculate new next location and direction

Exceptions

None	

Parameters

None	

Returns

None

Note

Warning: Does NOT know about other vehicles

4.5.2.14 packetCaught()

```
bool Vehicle::packetCaught (
            Packet thrownPacket )
```

checks to see if passed paket has reacched its destination or if it can be taken in

Looks at packet destination , if not current packet checks to see if vehicle id is in packets id list If Id list is empty, knows to create initial packet information If packet is to be added copies function members and adds to vehicles packet list

Precondition

throwPacket()

Postcondition

Packet is added to packet list

None

Exceptions

None	

Parameters

None	

Returns

Returns true if packet was added and false otherwise

Note

None

4.5.2.15 planDown()

```
bool Vehicle::planDown ( )  [protected]
```

Sets next location and direction to go down.

Increments X next position and sets direction down if within boundaries

Precondition

None

Postcondition

Next location and direction are modified

Algorithm

Checks if next position would be out of bounds Alters vehicle direction and next position

Exceptions

None	

Parameters

None	

Returns

Returns true if turn is not out of bounds; false otherwise

Note

None

4.5.2.16 planLeft()

```
bool Vehicle::planLeft ( )  [protected]
```

Sets next location and direction to go left.

Decrements X next position and sets direction left if within boundaries

Precondition

None

Postcondition

Next location and direction are modified

Algorithm

Checks if next position would be out of bounds Alters vehicle direction and next position

Exceptions

None	

Parameters

None	

Returns

Returns true if turn is not out of bounds; false otherwise

Note

None

4.5.2.17 planRight()

```
bool Vehicle::planRight ( )  [protected]
```

Sets next location and direction to go right.

Increments X next position and sets direction right if within boundaries

Precondition

None

Postcondition

Next location and direction are modified

Algorithm

Checks if next position would be out of bounds Alters vehicle direction and next position

Exceptions

None	

Parameters

None	

Returns

Returns true if turn is not out of bounds; false otherwise

Note

None

4.5.2.18 planUp()

```
bool Vehicle::planUp ( )  [protected]
```

Sets next location and direction to go up.

Increments X next position and sets direction up if within boundaries

Precondition

None

Postcondition

Next location and direction are modified

Algorithm

Checks if next position would be out of bounds Alters vehicle direction and next position

Exceptions

None	

Parameters

None	

Returns

Returns true if turn is not out of bounds; false otherwise

Note

None

4.5.2.19 redirect()

```
void Vehicle::redirect ( )
```

Redirects the vehicle.

Attempts to have the car to take an alternative route

Precondition

None

Postcondition

Vehicle direction and next position is modified

Algorithm

Stage 1: Find route alternative Stage 2: Attempt to wiggle your way out Stage 3: Give up all hope

Exceptions

None	

Parameters

None	

Returns

None

Note

Calling multiple times while not calling move will invoke different behaviors

4.5.2.20 setPacket()

```
void Vehicle::setPacket (
            bool holdsPacket )  [inline]
```

4.5.2.21 stop()

```
void Vehicle::stop ( )  [protected]
```

Stop vehicle.

Sets next position equal to current position and sets direction to NaN

Precondition

None

Postcondition

Next location = current position; direction = NaN;

Algorithm

Set vehicle direction = NaN Set next position equal = current position

Exceptions

None	

Parameters

None	

Returns

None

Note

None

4.5.2.22 throwPacket()

```
bool Vehicle::throwPacket (
            Vehicle * targetVehicle,
            Packet thrownPacket,
            bool update = false )
```

Throws packet to the passed vehicle and ouptputs a successful transfer message.

Helper function

Precondition

None

Postcondition

Packet is added to target vehicle

None

Exceptions

None	

Parameters

None	

Returns

Returns true if turn is not out of bounds; false otherwise

Note

None

4.5.2.23 updateLocation()

```
void Vehicle::updateLocation ( )
```

4.5.2.24 updatePacketCaught()

```
bool Vehicle::updatePacketCaught (
            Packet thrownPacket )
```

checks to see if passed paket has reacched its destination or if it can be taken in

Looks at packet destination , if not current packet checks to see if vehicle id is in packets id list If Id list is empty, knows to create initial packet information If packet is to be added copies function members and adds to vehicles packet list

Precondition

throwPacket()

Postcondition

Packet is added to packet list

None

Exceptions

None	

Parameters

None	

Returns

Returns true if packet was added and false otherwise

Note

None

4.5.2.25 vehicleRun()

```
bool Vehicle::vehicleRun ( )
```

throws packets if they exist

throws regular packets using bestDest and throws updates to all nearby vehicles

Precondition

called from World Run

Postcondition

Packets are thrown

None

Exceptions

None	

Parameters

None	

Returns

Returns false only right now

Note

None

4.5.3 Field Documentation

4.5.3.1 colMax

```
int Vehicle::colMax  [protected]
```

4.5.3.2 hasPkt

```
bool Vehicle::hasPkt  [protected]
```

4.5.3.3 hasUpdate

```
bool Vehicle::hasUpdate  [protected]
```

4.5.3.4 locations

```
std::vector<vehicleLocation> Vehicle::locations  [protected]
```

4.5.3.5 nearByVehicles

```
Vehicle* Vehicle::nearByVehicles[8]
```

4.5.3.6 newPacket

```
Packet* Vehicle::newPacket  [protected]
```

4.5.3.7 packets

```
std::vector<Packet *> Vehicle::packets
```

4.5.3.8 redirectCounter

```
unsigned int Vehicle::redirectCounter  [protected]
```

4.5.3.9 rowMax

```
int Vehicle::rowMax  [protected]
```

4.5.3.10 updates

```
std::vector<Packet *> Vehicle::updates
```

4.5.3.11 vehicleCount

```
int Vehicle::vehicleCount = 0  [static], [protected]
```

4.5.3.12 vehicleDir

```
VehicleDir Vehicle::vehicleDir  [protected]
```

4.5.3.13 vehicleId

```
int Vehicle::vehicleId  [protected]
```

4.5.3.14 xDest

```
int Vehicle::xDest  [protected]
```

4.5.3.15 xNextPos

```
int Vehicle::xNextPos  [protected]
```

4.5.3.16 xPos

```
int Vehicle::xPos  [protected]
```

4.5.3.17 yDest

```
int Vehicle::yDest  [protected]
```

4.5.3.18 yNextPos

```
int Vehicle::yNextPos  [protected]
```

4.5.3.19 yPos

```
int Vehicle::yPos  [protected]
```

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

- include/vehicle.h
- src/vehicle.cpp

4.6 vehicleLocation Struct Reference

`#include <vehicle.h>`

Data Fields

- int vehicleID
- int destX
- int destY
- int srcX
- int srcY
- bool thrown

4.6.1 Field Documentation

4.6.1.1 destX

`int vehicleLocation::destX`

4.6.1.2 destY

`int vehicleLocation::destY`

4.6.1.3 srcX

`int vehicleLocation::srcX`

4.6.1.4 srcY

`int vehicleLocation::srcY`

4.6.1.5 thrown

`bool vehicleLocation::thrown`

4.6.1.6 vehicleID

`int vehicleLocation::vehicleID`

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

- include/vehicle.h

4.7 World< DataType > Class Template Reference

`#include <world.h>`

Public Member Functions

- World ()

  Default/Initialization constructor.
- ~World ()

  World Destructor.
- bool initWorld (int sizeX, int sizeY)

  Initialize world.
- void displayWorld ()

  Displays world in ASCII format.
- bool populateWorld (int numObjects)

  Populate World.
- void clearWorld ()

  Clear World.
- void runWorld (int ticks)

  Run World.
- bool isObjectPresent (int xCoor, int yCoor)

  Is object present.
- bool getObject (int xCoor, int yCoor, DataType *&object)

  Get object.
- bool findObject (int id, DataType *object)

  Remove from list.
- int getNumObjects ()

  Get Number of Objects.
- std::vector< DataType * > & getObjectList ()

  Get Object List.
- bool insertObject (int xCoor, int yCoor, DataType *object)

  Insert Object.
- bool removeObject (int xCoor, int yCoor, DataType *object)

  Remove Object.
- bool deleteObject (int xCoor, int yCoor)
- bool generatePacket ()

  generates packet

Data Fields

- TransferType RUNALGORITH

Private Member Functions

- int findFromList (DataType *object)

  Find From List.
- bool removeFromList (int index)

  Remove from list.
- void runFlood ()

  transfers packet in a flood style algorithm
- void runDest ()

  Uses the destination search algorithm to transfer packets.
- void moveVehicles ()

  Moves vehicles if possible.
- void updateAdjacency ()

  updates eaqch vehicles adjacancy list

Private Attributes

- int worldSizeX
- int worldSizeY
- int numObjects
- int packetids = 0
- bool initializedLocations = false
- std::vector< DataType * > objectList
- std::vector< unsigned int > objectActionCounter
- DataType *** world

4.7.1 Constructor & Destructor Documentation

4.7.1.1 World()

```
template<class DataType >
World< DataType >::World ( )
```

Default/Initialization constructor.

Constructs World Class

Precondition

None

Postcondition

Initializes all data quantities

Algorithm

Standard initialization operation

Exceptions

None

Parameters

None

Returns

None

Note

None

4.7.1.2 ~World()

```
template<class DataType >
World< DataType >::~World ( )
```

World Destructor.

Deletes the world and all its objects

Precondition

Assumes at least one node constructed

Postcondition

All linked list nodes are removed

Algorithm

Dynamically allocate an array of pointers (x-axis) then dynamically

Exceptions

None

Parameters

None

Returns

None

Note

None

4.7.2 Member Function Documentation

4.7.2.1 clearWorld()

```
template<class DataType >
void World< DataType >::clearWorld ( )
```

Clear World.

Deletes all objects present in the world

Precondition

Assume initialized class object

Postcondition

All elements in world deleted

Algorithm

Go through each world array element and delete object. Set the object pointer to NULL

Exceptions

None	

Parameters

None	

Returns

None

Note

None

4.7.2.2 deleteObject()

```
template<class DataType>
bool World< DataType >::deleteObject (
            int xCoor,
            int yCoor )
```

4.7.2.3 displayWorld()

```
template<class DataType >
void World< DataType >::displayWorld ( )
```

Displays world in ASCII format.

Outputs to terminal all intersections and interminate roadways

Precondition

Assumes initialized world object and world data member

Postcondition

World filled with objects

Algorithm

Loop through intersections within world displaying their symbols

Exceptions

Cannot	display before world initialization

Parameters

None	

Returns

None

Note

None

4.7.2.4 findFromList()

```
template<class DataType>
int World< DataType >::findFromList (
            DataType * object )  [private]
```

Find From List.

Finds the matching object in the object address list

Precondition

None

Postcondition

Vector index at marching object returned

Algorithm

Go through the vector searching for a matching object address

Exceptions

None	

Parameters

in	*object*	Object address to search for

Returns

Vector index at marching object

Note

Returns -1 if no object found

4.7.2.5 findObject()

```
template<class DataType>
bool World< DataType >::findObject (
            int id,
            DataType * object )
```

Remove from list.

Removes item from the vector

Precondition

None

Postcondition

Item removed from list

Algorithm

Shift vector elements forward overwriting the element at index specified

Exceptions

None	

Parameters

in	*index*	Vector index to delete

Returns

Boolean stated if deletion is sucessful

Note

None

4.7.2.6 generatePacket()

```
template<class DataType >
bool World< DataType >::generatePacket ( )
```

generates packet

takes use input to deterimne the message and destination and src for a packet

Precondition

World must be initilized and populated

Postcondition

src vehicle has new packet

Algorithm None

Exceptions

None	

Parameters

in	*None*	

Returns

Boolean stated if packet is added succesfully

Note

None

4.7.2.7 getNumObjects()

```
template<class DataType >
int World< DataType >::getNumObjects ( )
```

Get Number of Objects.

Returns the number of objects present in the world

Precondition

Assumes initialized class object

Postcondition

Number of objects returned

Algorithm

Return numObject varible

Exceptions

None	

Parameters

None	

Returns

Integer specifying of objects present in the world

Note

None

4.7.2.8 getObject()

```
template<class DataType>
bool World< DataType >::getObject (
            int xCoor,
            int yCoor,
            DataType *& object )
```

Get object.

Returns the object from the world at the specified coordinates

Precondition

Assume initialized world object

Postcondition

Object from world returned

Algorithm

Check to see if the coordinates given are in range and see if there is an object present then return the address of the object

Exceptions

None	

Parameters

in	*xCoor*	X-axis coordinate

[in] yCoor Y-axis coordinate

[out] object Object returned from the world

Returns

boolean specifying if object returned sucessfully

Note

Returns false if the coordinates given are out of range and if there is no object at specified coordinates

4.7.2.9 getObjectList()

```
template<class DataType >
std::vector< DataType * > & World< DataType >::getObjectList ( )
```

Get Object List.

Returns the list of objects in the world

Precondition

Assume initialized object class

Postcondition

List of objects returned

Algorithm

Return the list of objects's address

Exceptions

None	

Parameters

None	

Returns

Address of list of object pointers

Note

None

4.7.2.10 initWorld()

```
template<class DataType >
bool World< DataType >::initWorld (
            int sizeX,
            int sizeY )
```

Initialize world.

Dynamically constructs the world with given parameters

Precondition

None

Postcondition

World initialized

Algorithm

Dynamically allocates the xCoor pointer and the yCoor pointers based on the parameters and sets all object pointers to NULL

Exceptions

None	

Parameters

in	*sizeX*	Max x-axis world size to be set

[in] sizeY Max y-axos world size to be set

Returns

None

Note

Can be ran multiple times
World is empty after initialization

4.7.2.11 insertObject()

```
template<class DataType>
bool World< DataType >::insertObject (
            int xCoor,
            int yCoor,
            DataType * object )
```

Insert Object.

Inserts object at specified coordinates

Precondition

Assume initialized class object

Postcondition

Object inserted into world

Algorithm

Check if there is not already an object present at given coordinates and insert the object into the world and push that object into the list

Exceptions

None	

Parameters

in	*xCoor*	X-axis coordinate

[in] yCoor Y-axis coordinate

[out] object Object to insert into the world

Returns

Bool stating insertion success

Note

None

4.7.2.12 isObjectPresent()

```
template<class DataType >
bool World< DataType >::isObjectPresent (
            int xCoor,
            int yCoor )
```

Is object present.

Checks if an object is present at given location

Precondition

Assumes initialized class object

Postcondition

Returns if the object is present or not

Algorithm

Check if the coordinates are in range, then check if a NULL ptr is not present at given coordinates

Exceptions

None	

Parameters

in	*xCoor*	X-axis coordinate

[in] yCoor Y-axis coordinate

Returns

Boolean signifying if an object is present

Note

None

4.7.2.13 moveVehicles()

```
template<class DataType >
void World< DataType >::moveVehicles ( )  [private]
```

Moves vehicles if possible.

iterates through each vehicle and moves it towards its destination

Precondition

None

Postcondition

Vehicles move

Algorithm

Exceptions

None	

Parameters

in	*None*	

Returns

Boolean stated if deletion is sucessful

Note

None

4.7.2.14 populateWorld()

```
template<class DataType >
bool World< DataType >::populateWorld (
            int numObjectsToInsert )
```

Populate World.

Fills the world with objects

Precondition

Assumes initialized world object and world data member

Postcondition

World filled with objects

Algorithm

Create n amound of new objects at random coordinates without overlapping previously created objects

Exceptions

Cannot	populate before world initialization

Parameters

in	*numObjects*	Number of objects to fill the world with

Returns

Boolean signifying populating success

Note

Number of objects can not exceed world capacity
INCOMPLETE FUNCTION: Need something to assign the coordinates to the vehicle

4.7.2.15 removeFromList()

```
template<class DataType >
bool World< DataType >::removeFromList (
            int index )  [private]
```

Remove from list.

Removes item from the vector

Precondition

None

Postcondition

Item removed from list

Algorithm

Shift vector elements forward overwriting the element at index specified

Exceptions

None	

Parameters

in	*index*	Vector index to delete

Returns

Boolean stated if deletion is sucessful

Note

None

4.7.2.16 removeObject()

```
template<class DataType>
bool World< DataType >::removeObject (
            int xCoor,
            int yCoor,
            DataType * object )
```

Remove Object.

Returns object and removes it from the world

Precondition

Assume initialized class object

Postcondition

Object returned and removed from world and list of objects present

Algorithm

Check to see if the coordinates given are in range and see if there is an object present then return the address of the object and remove it from the world and list of objects present

Exceptions

None	

Parameters

in	*xCoor*	X-axis coordinate

[in] yCoor Y-axis coordinate

[out] object Object returned from the world

Returns

Bool indicating success

Note

Returns false if the coordinates given are out of range and if there is no object at specified coordinates
Function does not delete the object, only removes and returns it

4.7.2.17 runDest()

```
template<class DataType >
void World< DataType >::runDest ( )  [private]
```

Uses the destination search algorithm to transfer packets.

each vehicle will start its life by broadcasting its starting point and its current destination, this will be spread by the flood algorithm. Each vehicle then uses the dest search algotim inside the vehicle class

Precondition

None

Postcondition

packets are moved

Algorithm None

Exceptions

None	

Parameters

in	*None*	

Returns

None

Note

None

4.7.2.18 runFlood()

```
template<class DataType >
void World< DataType >::runFlood ( )  [private]
```

transfers packet in a flood style algorithm

each vehicle broadcasts its packets to every vehicle in its range

Precondition

None

Postcondition

Item removed from list

Algorithm

iterate through each vehicle, iterates through the surrounding intersections if a vehicle is dicovered teh packet is passed to that vehicle If the throwing vehicle does not throw the packet then it will try again next tick A packet may live with a vehicle for 5 ticks before ageing out

Exceptions

None	

Parameters

in	*None*	

Returns

None

Note

None

4.7.2.19 runWorld()

```
template<class DataType >
void World< DataType >::runWorld (
            int ticks )
```

Run World.

Runs the world an arbituary amount of ticks

Precondition

Assume initialized class object

Postcondition

World ran

Algorithm

Go through each array element checking each

Exceptions

None	

Parameters

in	*ticks*	Number of times to run each element's function

Returns

None

Note

None

4.7.2.20 updateAdjacency()

```
template<class DataType >
void World< DataType >::updateAdjacency ( )  [private]
```

updates eaqch vehicles adjacancy list

finds all vehicles nearby and adds them to the list of near by vehicles

Precondition

None

Postcondition

Each vehicle knows what vehicles are in the immediate area

Algorithm None

Exceptions

None	

Parameters

in	*None*	

Returns

None

Note

None

4.7.3 Field Documentation

4.7.3.1 initializedLocations

```
template<class DataType>
bool World< DataType >::initializedLocations = false  [private]
```

4.7.3.2 numObjects

```
template<class DataType>
int World< DataType >::numObjects  [private]
```

4.7.3.3 objectActionCounter

```
template<class DataType>
std::vector<unsigned int> World< DataType >::objectActionCounter  [private]
```

4.7.3.4 objectList

```
template<class DataType>
std::vector<DataType *> World< DataType >::objectList  [private]
```

4.7.3.5 packetids

```
template<class DataType>
int World< DataType >::packetids = 0  [private]
```

4.7.3.6 RUNALGORITH

```
template<class DataType>
TransferType World< DataType >::RUNALGORITH
```

4.7.3.7 world

```
template<class DataType>
DataType*** World< DataType >::world  [private]
```

4.7.3.8 worldSizeX

```
template<class DataType>
int World< DataType >::worldSizeX  [private]
```

4.7.3.9 worldSizeY

```
template<class DataType>
int World< DataType >::worldSizeY  [private]
```

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

- include/world.h
- src/world.cpp

Chapter 5

File Documentation

5.1 include/intersect.h File Reference

Declaration file for intersect class(es)

```
#include <sstream>
#include <string>
#include "vehicle.h"
```

Data Structures

- class Intersect1

5.1.1 Detailed Description

Declaration file for intersect class(es)

Author

Tyler Michael Goffinet

Declares all member methods and structures of the intersect class(es)

Version

1.0 Tyler Michael Goffinet (20 October 2016) Original Code

Note

None

5.2 include/simulator.h File Reference

Definition file for simulator class.

```
#include <iostream>
#include "../src/world.cpp"
#include "vehicle.h"
```

Data Structures

- class Simulator

5.2.1 Detailed Description

Definition file for simulator class.

Author

Brandon Thai Nguyen

Specifies all member methods of the simulator class

Version

1.0 Brandon Thai Nguyen (03 October 2016) Original Code

Note

Requires vehicle.h

5.3 include/vehicle.h File Reference

Definition file for vehicle classes.

```
#include <iostream>
#include <vector>
```

Data Structures

- struct Packet
- struct vehicleLocation
- class Vehicle
- class Taxi

Enumerations

- enum VehicleDir { NaN, UP, DOWN, LEFT, RIGHT }

5.3.1 Detailed Description

Definition file for vehicle classes.

Author

Brandon Thai Nguyen

Specifies all member methods of the vehicle classes

Version

2.00 Tyler Michael Goffinet (24 October 2016) Added getId(), getLocation(), getDirection(), hasPacket(), getNextLocation(), move(), setPacket(), protected functions, and member variables

1.00 Brandon Thai Nguyen (03 October 2016) Original Code

Note

None

5.3.2 Enumeration Type Documentation

5.3.2.1 VehicleDir

`enum VehicleDir`

Enumerator

NaN	
UP	
DOWN	
LEFT	
RIGHT	

5.4 include/world.h File Reference

Definition file for world class.

```
#include <time.h>
#include <cstdlib>
```

```
#include <sstream>
#include <vector>
#include "intersect.h"
#include "vehicle.h"
```

Data Structures

- class World< DataType >

Enumerations

- enum TransferType { FLOOD, DESTSEARCH }

5.4.1 Detailed Description

Definition file for world class.

Author

Brandon Thai Nguyen

Specifies all member methods of the world class

Version

1.00 Brandon Thai Nguyen (03 October 2016) Original Code

Note

Requires world.h, vehicle.h

5.4.2 Enumeration Type Documentation

5.4.2.1 TransferType

```
enum TransferType
```

Enumerator

FLOOD	
DESTSEARCH	

5.5 src/intersect.cpp File Reference

Implementation file for intersect class(es)

```
#include "intersect.h"
```

Macros

- #define CLASS_INTERSECT_CPP

5.5.1 Detailed Description

Implementation file for intersect class(es)

Author

Tyler Michael Goffinet

Implements all member methods of the intersect class(es)

Version

1.0 Tyler Goffinet (03 October 2016) Original Code

Note

Requires intersect.h

5.5.2 Macro Definition Documentation

5.5.2.1 CLASS_INTERSECT_CPP

```
#define CLASS_INTERSECT_CPP
```

5.6 src/main.cpp File Reference

Driver program.

```
#include <iostream>
#include "simulator.h"
```

Functions

- int main (int argc, char **argv)

5.6.1 Detailed Description

Driver program.

Version

1.1 Brandon Thai Nguyen (06 September 2016) Updated for use with simulator class

1.0 Tyler Goffinet (28 September 2016) Original Code

Note

5.6.2 Function Documentation

5.6.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

5.7 src/simulator.cpp File Reference

Implementation file for simulator class.

```
#include <sys/wait.h>
#include "simulator.h"
```

Macros

- #define CLASS_SIMULATOR_CPP

5.7.1 Detailed Description

Implementation file for simulator class.

Author

Brandon Thai Nguyen

Implements all member methods of the simulator class

Version

1.0 Brandon Thai Nguyen (03 October 2016) Original Code

Note

Requires simulator.h

5.7.2 Macro Definition Documentation

5.7.2.1 CLASS_SIMULATOR_CPP

```
#define CLASS_SIMULATOR_CPP
```

5.8 src/vehicle.cpp File Reference

Implementation file for vehicle classes.

```
#include "vehicle.h"
```

Macros

- #define CLASS_VEHICLE_CPP

5.8.1 Detailed Description

Implementation file for vehicle classes.

Author

Brandon Thai Nguyen

Implements all member methods of the vehicle classes

Version

2.0 Tyler Goffinet (24 October 2016) Implemented functions

1.0 Brandon Thai Nguyen (03 October 2016) Original Code

Note

Requires vehicle.h

5.8.2 Macro Definition Documentation

5.8.2.1 CLASS_VEHICLE_CPP

```
#define CLASS_VEHICLE_CPP
```

5.9 src/world.cpp File Reference

Implementation file for world class.

```
#include "world.h"
```

Macros

- #define CLASS_WORLD_CPP

5.9.1 Detailed Description

Implementation file for world class.

Author

Brandon Thai Nguyen

Implements all member methods of the world class

Version

1.3 Daniel Smith Added generate packet and packet transfering capabilities

1.2 Tyler Goffinet Implemented display and updated runWorld()

1.1 Tyler Goffinet Updated for use with vehicle classes

1.0 Brandon Thai Nguyen (03 October 2016) Original Code

Note

Requires world.h

5.9.2 Macro Definition Documentation

5.9.2.1 CLASS_WORLD_CPP

```
#define CLASS_WORLD_CPP
```