Maddie Shea

Are Oelsner

Ryan Jennings

Final 240 Project

Class Design:

main.cpp

// Utility class for random number generation
Random.h

// Class for collecting statistics data
Statistics.h

Intersection.h:
-> includes Lane.h, Section.h, Vehicle.h, TrafficLight.h

- Vehicle[] vehicles
- Intersection intersection
- Lane*[4] lanes
- Clock simClock

Vehicle.h:
-> includes Section.h, Lane.h, Random.h

-Vehicle(int size, double _rightProb, double _leftProb, Lane* _lane);

- vector<Section*> sections
- int size
- boolean inIntersection
- boolean nearIntersection
- boolean exited

- void move()
- boolean canMove()

-printVehicle()   // debugging purposes

Section.h:

* Neighbor Sections:

- Section* up
- Section* right
- Section* down
- Section* left
- boolean occupied
- boolean nearEdge
- boolean nearIntersection
- boolean inIntersection

* Getting sections relative to the Intersection
- Section* getUpSection()
- Section* getRightSection()
- Section* getDownSection()
- Section* getLeftSection()

- bool getOccupied()
- bool getNearEdge()
- bool getNearIntersection()
- bool getInIntersection()

* Getting sections relative to a Vehicle in the Section
- Section* getStraight()
- Section* getRight()
- Section* getBack()
- Section* getLeft()

- void setNeighbor(Section* _section, int direction)
- void setOccupied(bool _occupied)
- void setNearEdge(bool _nearEdge)
- void setNearIntersection(bool _nearIntersection)
- void setInIntersection(bool _inIntersection)

*Prints sections in line form
- printSection()
*Prints sections in t form
- printSection2()

Lane.h:
-> includes Section.h, TrafficLight.h

- Lane(int length, Section* intSec1, Section* intSec2, int direction)
- vector<Section*> sections // Size given by user
- int direction // 1/N, 2/E, 3/S, 4/W
- TrafficLight trafficLight

- Section* getSection(int i)
- int getDirection()
- TrafficLight getTrafficLight()

- vector<Section*> allocSections(int size)
- bool canAllocSections(int size)

TrafficLight.h:

- TrafficLight(double green, double red, double yellow, Color startColor);

- enum Color { red, green, yellow }
- Color currentColor

- double greenTime // Given by user
- double yellowTime // Given by user
- double redTime // Given by user

- double getGreenTime()
- double getYellowTime()
- double getRedTime()
- Color getColor()
- double timeRemaining(double timeElapsed)

Intersection.h:

-> includes Lane.h, Section.h, Vehicle.h, TrafficLight.h

- Lane north;
- Lane east;
- Lane south;
- Lane west;

* Intersection Sections
- Section NW;
- Section NE;
- Section SE;

- Section SW;

-vector<Vehicle> vehicles

-TrafficLight northTrafficLight;
- TrafficLight eastTrafficLight;
- TrafficLight southTrafficLight;
- TrafficLight westTrafficLight;

-int length

Testing

We tested each class individually to make sure that it compiled. Then we ran the main continuously adding the classes. First, we tested the parser, then the traffic light and so on. We built up the main by adding more components and making sure each class compiled and executed the correct methods. We created print functions for Vehicle, Section, and Lane to assist in debugging. Sections are represented as brackets [ ] and contain an O if it is occupied, an N if it is near the intersection and n if both near the intersection and occupied, I if it is in the intersection and i if it is both in the intersection and occupied, and E if it is near the edge of the lane and e if it is both near the edge of the lane and occupied. If none of these conditions are true, then it is printed as just an open bracket. The vehicle print function prints the values of the variables in Vehicle, which is done for each vehicle in the intersection.

Additionally, we created a separate section tester file because the sections are so important to the program. We needed to make sure that all of the sections were correctly labeled and had correct values.

How to Run

Provided is a makefile and an input.config file. The input.config file is set up so that comments are indicated using a #. The inputs can be changed by changing the values after the equals sign. The parameters are specified in the configuration file, and there is no negative input. We have set a minimum lane length requirement to 12, so that Vehicles spawn at least one section away from the section closest to the entrance to the intersection. The program should be compiled using “make”, and then executed using “make r”.