Tom Dale

CS-273

FinalSpec.doc

7-20-2016

Traffic Junction Simulation

**Project Requirement:**

The people of 273*ville* are planning to construct new intersections within the city limits. At these intersections there will be either a roundabout, or a four-way traffic light constructed. To understand the efficiency of both of these intersection types, and to determine which should be used at a given junction a computer simulation was requested. The user will input the total cars using the intersection per hour as well as the percent of cars entering the four-way from the north, south, east, and west direction. The simulation will run a 4 hour block and give the user information about the efficiency of both types of intersections and a recommendation for which intersection type is preferred.

**Requirement Specifications**

* Program needs to simulate traffic stops in the town of 273*ville*, population 2000
* The names of all the residents of 273*ville* are stored in our class folder.
* Every person in 273*ville* could drive through the intersection.
  + User sets average amount of vehicles using intersection per hour.
  + Vehicles can be cars or emergency vehicles.
  + On average 1 out of 300 cars will be an emergency vehicle.
* Each intersection has four incoming streets labelled North, South, East, and West
  + User sets a percentage of vehicles that drive from each direction. (All totaling 100%)
* A car is 50% likely to drive straight, 25% likely to turn right, and 25% likely to turn left.
* U-turns are illegal.
* Time to go through intersection is based on intersections with speed limits of 30 mph.
* After the user enters the 5 inputs the simulation runs for a traffic light and roundabout.
  + Both intersection types have different rules (mentioned below).
  + When emergency vehicles approach all cars exit the intersection until the emergency vehicle is through.
* The simulations are run for 4 hours
* Results are displayed for both simulation
  + Stats Displayed: Total cars through, total emergency vehicles through, average waiting time from N,S,E,W, and total average waiting time
* Rules for Roundabouts:
  + Vehicles turning right = 4s, driving straight = 8s, turning left = 12s
  + Cars will not enter if portion of roundabout in front of them is taken
  + 4 portions inside roundabout (NE,NW,SE,SW) each hold 2 cars driving through
* Rules for Traffic Lights:
  + Vehicles turning right = 3s, driving straight = 4s, turning left = 6s
  + Lights can be GO for north/south or east/west, there are no special turning lights.
  + Light will change after 2 minutes if it is needed, or it will change if it’s lanes are empty.
  + Yellow light time (cars will not enter but they will leave) = 4s
  + No car will enter the intersection if the space in front of them is taken
  + If space required to move into intersection is open:
    - Right turns are always allowed
    - Driving straight must have a green light
    - Turning left must have an empty que or car turning left at the opposite road
* The program will recommend that the intersection with the lowest wait times be constructed.

**Design**

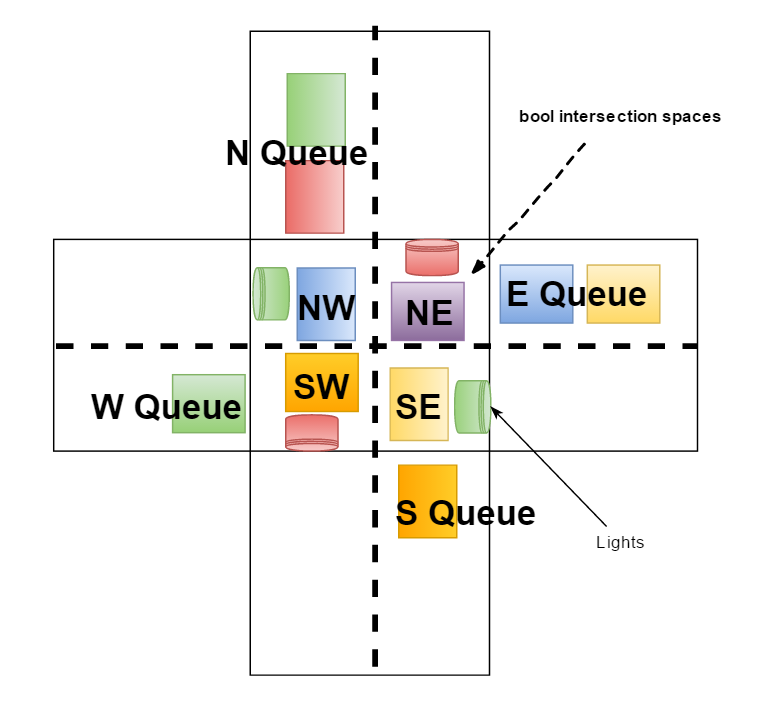
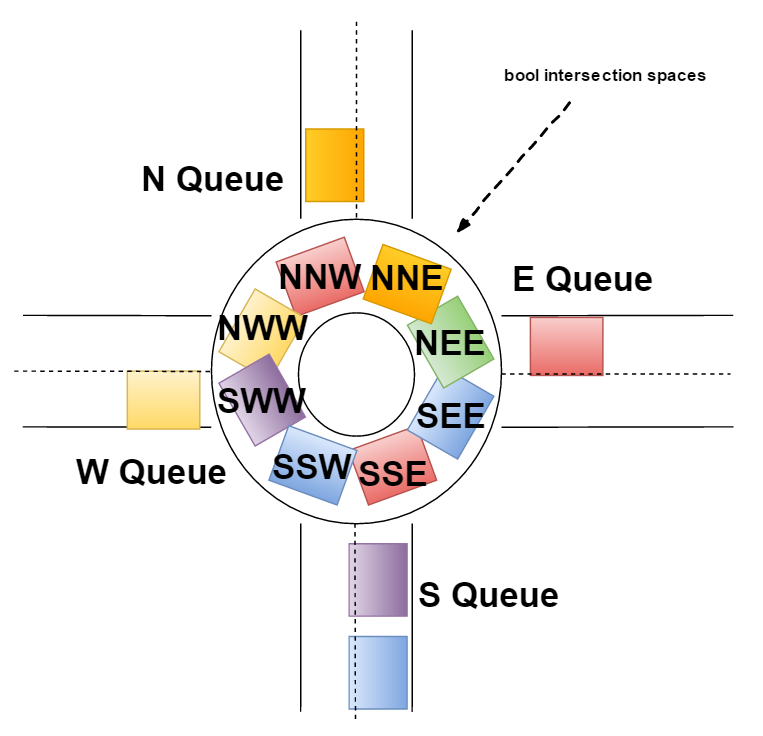
The main function will have the user input the average amount of cars that go through the intersection each out as well as the percent of those cars coming from each of the four directions. Main will also ask the user if he would like to be shown all the data of cars moving through traffic. The amount of cars driving through from each road as well as a Boolean show value to designate to show the exact traffic or not. The main function will call both roundabout and trafficlight to run. After this is finished main will call the .rate() function from both intersection simulations and compare them. Rate will be an integer number that tests the overall wait time of cars. The intersection type with the lower .rate() will be suggested to the user as the recommended traffic junction.

The traffic simulation will be run mainly within the two classes Roundabout and TrafficLight each inheriting from the intersection class. The intersection .run() virtual function will be implemented differently be both child classes because of the different nature of roundabouts and traffic lights. Both intersections have for queues N,S,E,W that hold cars waiting to enter junction. The roundabout has eight bool values NNE, NEE,SEE, SSE,SWW,SSW,NNW, NWW that represent the space required for a new car to enter the roundabout. The lighted intersection works differently, there are only four spaces NE,SE,SW,NW that cars move though, but there is also a NStrafficlight bool that represents a north-south traffic light when true and east-west light when false. Both run functions are a while loop that increments the clock with each loop representing one second until 14,400 second (four hours) have been run. The loop in both intersections works the same. Call a function input cars into the queues if a new car arrives, then call a function to move any cars in the intersection out or towards their next space. If bool showCars is true then call each car upon leaving the intersection to .print\_data().. Next the new car is checked to be an emergency vehicle, if it is then set emergency count to the vehicle .get\_time\_through(). Now check to see if all spaces in intersection are empty, and if so decrement emergency count once per loop. If the new vehicle was not an emergency vehicle then call a moveCarsIn function to move any car that is able out of the front of the queue and into a space in the intersection.

The random class is used to execute any random function. It contains a bool rand(int x) function that returns true x/3600 % of the time (the amount of seconds in one hour). This is used by the addCars function to designate when a new car approaches. It also has a string randDirection() function that returns left 25%, right %25, and straight 50% of the time to be used by a car object to decide which way it is turning. It also uses an int rand\_int() function to return a random integer between 0 and RAND\_MAX to be used in all other needs for randomness.

The car class and inheriting Emergency\_Vehicle class have a name, vehicle type, as well as a direction for turning and time it takes to do so. Within the Vehicle.h file are functions called initList() and getnext() which organize the list car is created from. InitList() initializes a global list that reads in names from “residents of 273vill.txt” and pairs them with a vehicle type, cars can be trucks, SUVs, or Sedans, and emergency vehicles can be firetruck, ambulance, or police car. The no argument constructor for car and emergency vehicle pulls a random place in the set and imports that name and vehicle type. A random turning signal is assigned from randDirection() and the proper time it takes is assigned along with it.

**Intersection Diagrams:**



**UML Diagram:**

****

**Use Cases:**

-Enter variables (no intersection user data)

|  |  |  |
| --- | --- | --- |
| 1 | **Use Input** | **System Response** |
| 2 |  | Print welcome message and prompt for average cars per hour |
| 3 | Enter average cars per hour hour | If entry < 0 ask user to try again |
| 4 |  | Ask user to enter % traffic from North street |
| 5 | Enter number for percent traffic from North | If entry < 0 or > 100 ask user to try again |
| 6 | Same as above for South, East, and West | Repeat last sequence for South, East, and West car traffic. |
| **7** |  | If North + South + East + West are not equal to 100 ask user to try again |
| **8** |  | Convert from percent traffic N,S,E,W to amount of cars coming from N,S,E,W. |
| **9** |  | Send N,S,E,W to roundabout and traffic light tests. |
| **10** |  | Ask user if they would like to view traffic user data. |
| **11** | Enter ‘n’ (or anything else) | Continue as normal |
| **12** |  | Run roundabout and traffic light and return simulation data |
| **13** |  | Print which intersection type is recommended from simulation |

**-**Choose to see intersection users (inserted starting at row 10 from previous use case);

|  |  |  |
| --- | --- | --- |
| 1 |  | Ask user if they would like to view traffic user data |
| 2 | Enter ‘y’ |  |
| 3 |  | Run roundabout and traffic light and return each vehicle name type turn and time taken |
| 4 |  | Return simulation data |
| 5 |  | Print which intersection type is recommended from simulation |

**Psuedo-code:**

**In Source.cpp**

Get\_input()

1. While (true)
2. Cout << enter cars per hour
3. Cin >> avgcars
4. If input restart loop < 0 restart loop
5. Cout << Enter north south east west percent traffic
6. Cin Ncars Scars Ecars Wcars
7. If any inputs are < 0 or > 100 or if all four value are equal to 100 then
8. Cout << do you want to see individual vehicle information
9. Cin >> ‘y’
10. Set show to true
11. Pass roundabout and lightedintersection .set\_cars(Ncars,Scars,Ecars,Wcars,show)
12. Return

**In intersection.h**

Run()

1. Int tick = 0
2. While (tick < 14400)
3. addCars()
4. moveCarsOut()
5. if emergency count is equal to 0
6. move cars in
7. else if all spaces in intersection are empty
8. emergency – 1
9. return

addCars()

1. if(random(Ncars)) //car approach rate
2. car\* entering = new car
3. if( new car is emergency vehicle)
4. emergency = entering->get\_time\_through()
5. else
6. Nqueue.push\_back(entering)

moveCarsOut()

1. if space is taken in intersection
2. call that cars .move(string location) function an