Object Oriented Programming

Project

Highway Simulation

Names and Registration Numbers of the Students who developed the Project

· **Costas Chatzopoulos** 1115201300202

· **Anxhelino Mehmeti** 1115201400233

Compilation

g++ -o highway *.cpp

Execution Examples

- **$ ./highway 100 4 0 67**
- **$ ./highway 507 9 7 48 0 1 1 1 0**

Parameters

1. **[N: >= 0]**
   Number of cycles that the simulation will run.
2. **[NSegs: >= 0]**
   Number of segments in the highway (There will be NSegs + 1 Nodes).
3. **[K: >= 0]**
   In each cycle, K vehicles will try to exit from a Toll (For e-pass this number is 2 times K).
4. **[Percent: 0/100]**
   Percentage of vehicles that will get ready to leave a segment, in each cycle.
5. **[Toll size input: 0/1]** (optional)
   If 1, every time a Toll is created, the program will ask for the size of the Toll. Else, it's random.
6. **[Initialize: 0/1]** (optional)
   If 1, the Highway will be initialized with vehicles. Else, it will be empty.
7. **[Manually: 0/1]** (optional)
   If 1, every cycle will start manually. Else, all the cycles run until the end of the program.
8. **[Print: 0/1]** (optional)
   If 1, prints everything that happens in the Highway. Else, only the prints that were asked.
9. **[Debug: 0/1]** (optional)
   Execute the program in Debug Mode. (Constructors, Destructors, extra Debugging Prints)

Information about the Methodology that was followed

CLASS VEHICLE

Every vehicle has its own plate number. This way we can distinguish every vehicle from the others while they are moving in the Highway. If it's printed, a **ready** Vehicle is **green**, if not then it's **blue**.

CLASS TOLL

In each Toll there can be a maximum of **capacity** Vehicles waiting in its **dequeue** (double ended queue). A Toll is considered to be inactive when it's created with capacity equal to 0. Also, a Toll's **percentage** of vehicles in it, can be printed (from the class Entry), compared to its capacity.

CLASS ENTRY

K becomes 0 when the segment is full. This way, congestion is prevented because the tolls won't let any other vehicles pass until the situation in the segment is better, while K is incremented by 1 each time. If K is 0, no new vehicles will enter the Tolls, because they have been "informed" about the traffic in the segment that they want to enter, so they choose not to enter the highway! Also, the number of vehicles that will enter any Toll, is a random number from 0 to the number vehicles that can enter that toll - which depends on its capacity and on the number of vehicles that are already there. Also, an Entry can print the **percentage** that has vehicles, compared to the number of maximum vehicles possible for that Entry. Another initiative we took, was that when a new Vehicle goes to the Entry, it decides to go in the queue of the Toll with the least amount of Vehicles (as long as the capacity is not violated).

CLASS SEGMENT

A Segment has a dequeue of Vehicles, from which a percentage will get ready to exit from that Segment and pass to the next one, or exit the Highway. This queue has a specified capacity which cannot be violated. Also a Segment has a pointer to the Previous Segment (Left) and the Next Segment (Right). The first Segment has a NULL pointer to its Left and the last one, a NULL pointer to its Right. Also, a Segment can print the **percentage** of vehicles in it, compared to its capacity.

CLASS NODE

A Node contains 4 pointers. One to its Previous Node (Left), one to its Next Node (Right), one to its Entrance and one to the Segment that's after it. The last node has a NULL pointer to Segment, since there is no Segment after it. This means there are NSegs+1 Nodes. When we create a Node, we also create an Entry (its Entry) and a Segment (the Segment after it), unless it's the last Node.

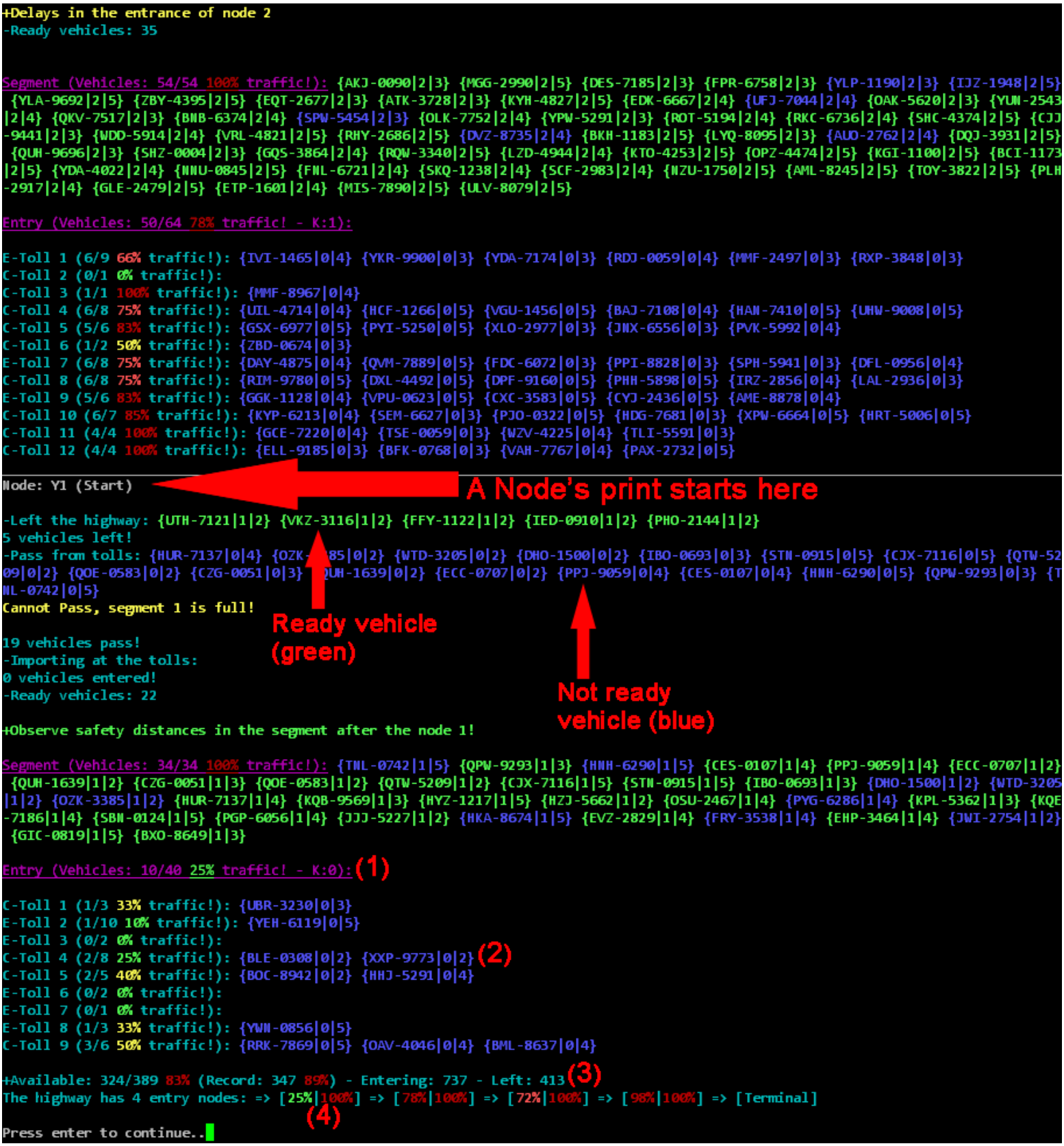
CLASS HIGHWAY

The Highway has a pointer to pointers to Node. This allows dynamic creation of a row of Nodes of a specific size, even if this size is different every time. Also, a Highway can print the **percentage** of vehicles that move in it, compared to the number of maximum vehicles possible in it.

CLASS UNIQUE_RAND (OPTIONAL)

For the simulation's needs, we used the class Unique_Rand. This class acquires usefulness for **set_ready()** function, which randomly makes a vehicle **ready** to exit in the next cycle of the simulation so as not to make a Vehicle ready, if it already is.

Information from screenshot of one Cycle



(1)

We see 10 out of 40 Vehicles, at most in this Entry. Which means 25%, so we print that.

(2)

Same as Entry, but instead it's data for the Toll. We can also see that a Vehicle is printed like this: {XXP-9773|0|2}. The first part (before the |) is the plate number. The second part is the current position (if it's 0, it hasn't entered the Highway yet), and the third is the destination. The numbers correspond to Nodes.

(3)

Here, first, we see 324 out of 389, which is 83%. This means that there are 324 Vehicles currently in the Highway, out of the maximum number of 389 Vehicles (in the current simulation). Then, we see "Record" which tells us the most Vehicles that have been in the Highway, as well as the percentage that they cover. In the end, "Entering" is a counter of how many Vehicles have entered the Highway until now, and "Left" is also a counter for Vehicles that have exited the Highway.

(4)

This is a representation of the Highway in the current Cycle, in which we can almost visualize the Highway. After an open bracket we see the percentage of an Entry and after a vertical bar | we see the percentage of the Segment.

Screenshot without setting parameter Print as 1