1.0 Introduction

1.1 Goals and objectives

The objective of this programming project was to design and implement an automobile traffic simulation program. Simutraffic will take a set of parameter values (which define the average arrival rate, the number of queues, the average departure rate per car, the time interval and the running time) , and then simulate a traffic queue depending on these values. It will go through discrete timing and will display what's happening to the user by visual means.

This project was done to show that reality could be modeled by a doing a simulation program. Though, it is impossible to model every aspect of our lives and also impossible to perfect each of those aspects, it's a great idea to try to model things from real life. By modeling some of our life's aspects, we could evaluate many behaviors (traffic light sequence, flight simulation, car simulation ... etc.) to learn more about them and try to think of some suitable solution. This is one of the main jobs of an engineer.

The project was done through several steps. First, a specification was evaluated until an agreement was met with the client. Second, the system architecture was designed and made to model the simulation programs every aspect. And then, the program was implemented using C++ for the program mains engine and using the fltk classes for the interface design.

Finally, the program was tested thoroughly and validated to see if it complies with the specification.

1.2 Background

Traffic congestion is a major concern in many countries. Populations are getting bigger, and more traffic jams are created everyday. The only way to solve this is through the use of traffic engineering. Traffic engineers work day and night to solve these traffic congestion problems. As it is known, traffic congestion is a hard thing to solve in developing countries, and it causes many problems to the standards of living in them. But traffic engineers alone will not be able to solve these problems; they are in need of some of models of real life situations to do their research on and to discover new solutions. This is where a traffic simulator program becomes useful.

Traffic simulation programs help traffic engineers them in work. Their main job is to create for these engineers a dynamic traffic environment that is modeled from real life.

The main problem in designing and implementing these simulation programs is that driver's behavior is very hard to model. This is due to the fact that modeling any thing from real life is hard and needs a lot of research and work. Even if the program has been designed and implemented, it may have some flaws on modeling the driver's reactions. Research is still being done on how to model the driver's interactions perfectly to develop these simulation programs.

One of the best packages used to simulate traffic behavior is "Paramics" by Quadstone limited software.This program has a very large customer base and is used extensively in many of the world's government and research centers. More information is on http://www.paramics-online.com/.

1.3 Report overview

This report has been written in 8 chapters, a brief description of each chapter is shown below:

Chapter 1: Introduction:
This chapter will explain simutraffic in a brief way and will give a background on the subject of the project. It will also include an overview of the reports chapters.

Chapter 2: System overview:
This chapter will explain the system requirements, specifications and a summary of changes that occurred on the project.

Chapter 3: System architecture and model:
This chapter will show the system architecture diagrams that include the functional decomposition diagram, the DFD and the state–transition diagram.

Chapter 4: System design:
This chapter will explain every function and store used in the simulation program.

Chapter 5: Implementation:
This chapter will explain the platform used to develop the program, the algorithm models for the arrival and departure of the cars, the interface design, implementation issues. It will also show on input / output example.

Chapter 6: Testing:
This chapter will show how the program has been verified and evaluated to meet the specification that has been set for it.

Chapter 7: Documentation:

This chapter will explain how the program can be used including several snapshot of the simulation in work.

Chapter 8: Project discussion:

The final chapter that will include a summary of the work done, some critical proposal and a section to show some enhancements that could be add to the system.

2.0 System overview

2.1 System requirements

The requirements of the client was to design and implement a program that will do three things:

- Simulate traffic junction queue and show their behavior as a function of:
  1- Arrival rate per unit time
  2- Departure rate per car
  3- Traffic light sequence
- Display the queue as they are simulated
- Give a set of statistics for each lane on each road after the simulation has finished.

2.2 System specification

The program will be divided into a category of function. They will be described in the following sections.

2.2.1 Input reading

This set of functions will read the traffic sequence from a file and the needed parameters from the GUI stores using a set of ordinary C++ functions.

2.2.2 Geometry drawing

This set of function will draw the environment of the simulation (cars, traffic light ... etc) and will also draw the user interface and set it up. This will be done using fltk's classes and widgets which are a free set of classes specifically done for GUI designing.

2.2.3 Initialize GUI parameters and queues

This set of functions will initialize the needed parameters values before and after the program starts. It will also setup queue data structures for each of the car lanes.The queue data structures will be set using a set of linked struct lists.These kinds of data structures are a good for queued data ,each including a set of needed info. (In this case,the cars color code and the cars time of arrival to the queue).

2.2.4 Simulate queues

This set of functions will simulate the queues by increasing and decreasing as needed. They will also walk the simulation through discrete timing, updating as needed. This will be done using a control loop which will control the steps of how the increase, decrease and car drawing functions are called.

2.2.5 Statistical recording

This set of functions will record and output the simulation statistics in a formatted table for easy reading. It will use ordinary ways of calculating the statistical values and print them directly to an output file.

2.3 User inputs

The inputs that will be provided by the user are:

- The number of lanes
- The traffic light sequence
- The average car arrival rate/min

2.4 Changes in project

While working with the programming project, some changes occurred to the specifications occurred to the specifications. A summary of these changes is shown below:

2.4.1 Input change

The new inputs that were added are:

1 Time interval

2 Running Time

3 Average Departure time per car

- Time interval

  This input is used to specify how much each time interval duration is.
  This will help adjust the simulation's timing depending on the user's needs, e.g. if he wants a much finer simulation (i.e. very detailed), the user should select the smallest interval.

- Running time

  This input is needed to specify the duration of the virtual environment of the simulation. This will NOT BE BASED ON REAL TIME i.e. if 1 min is specified, that doesn't mean the program will run for one minute.

- Average Departure time per car

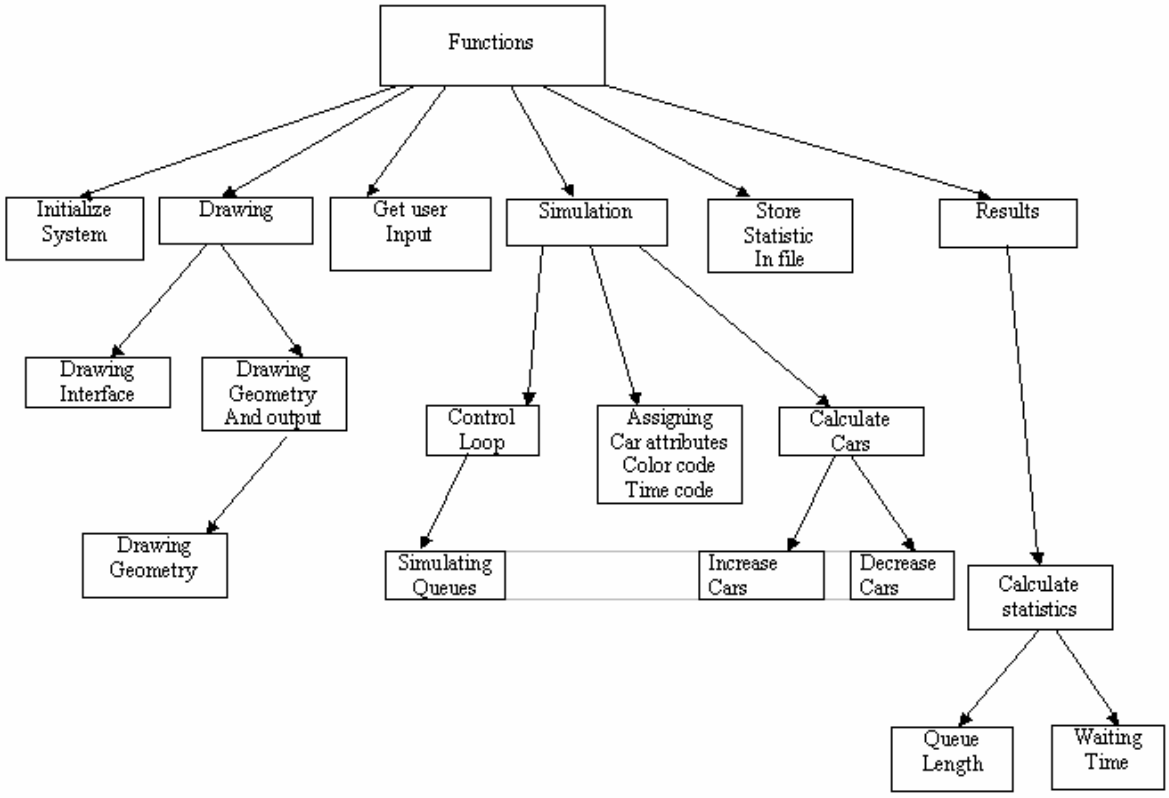
  This input is needed to calculate number of cars that departure from the queue. Before, the departure rate was stated in the requirements to be fixed (Refer to Appendix A: Requirements Paper, constraints section). But now this has changed. We will use instead a standard normal distribution model for the departure part.

2.4.2 Function change

The only change that occurred to the functions is that the output will be in a text file, which will be formatted in Ms Excel® format for the user to see.

3.0 System Architecture

3.1 Functional Decomposition



3.2 DFD

The DFD can be seen on the next page.

3.3 Behavioral Model

This can be seen on the DFD page,where each color specifies a state:

Orange: Input state

Red: Simulate State

Blue: Output State

4.0 System Design

4.1 Functions:

4.1.1 Initialize System

This function will initialize the traffic light sequence array and car queues linked list. Its only input is the traffic light sequence file and its outputs are the initialized traffic light sequence array and linked car queues lists.

4.1.2 Draw Interface

This function will draw the GUI as shown below:

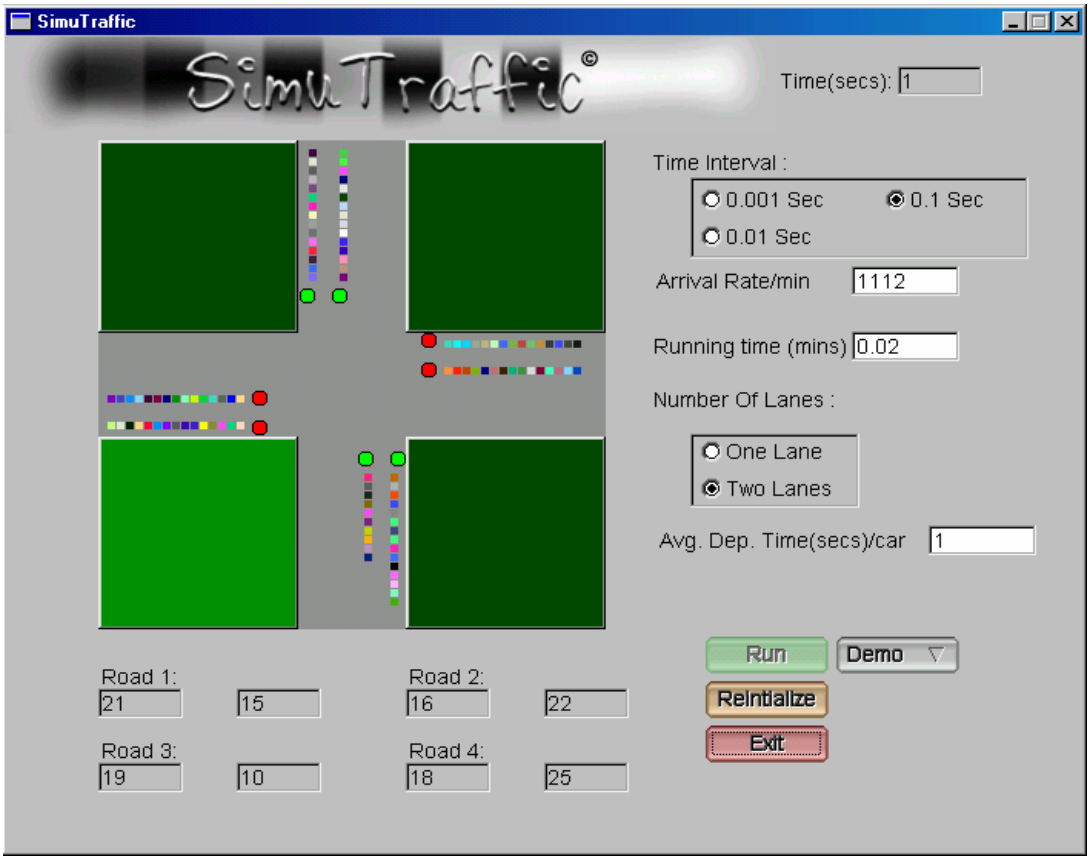


Figure 1 The Interface

It will not have any inputs except that it will output the systems parameters, which are the arrival rate, the running time, time interval, the number of lanes, and the average departure time per car.

4.1.3 Get User Inputs

This function will get the GUI parameters and store them in memory .It's input are the parameters of the system and it's output is the store of each of the parameters (each store has the same name as the parameter it's going to store).

4.1.4 Draw Geometry

This function contains the main drawing components of fltk to draw the traffic queues on screen as they increase and decrease. Its inputs are the number of lanes, the car queues, and the traffic light sequence. Its output is the drawing of the car queues and traffic light status.

4.1.5 Simulate Queue

The main core of the program, this function controls the main loop and how will the program flow for each function. It goes through n times that correspond to the running time given by the user. Its inputs are the running time, the number of lanes and the time interval. As it's only a control loop, it does not have any outputs.

4.1.6 Increase cars

This function increases the number of the cars in the queues depending on their situation, it uses the Poisson statistical model to compute the number of cars that arrive in a specific time interval. Its inputs are the arrival rate, the time interval, and the car queues. Its output is the accumulation of queue lengths and the modified car queues.

4.1.7 Assign color code and time for each car

This function is part of the increase car function, as it modifies the cars that have been added in each queue, by assigning them random color codes and there arrival time. Its input is the car queues and its output is the modified car queues.

4.1.8 Decrease cars

This function decreases the number of cars after a certain time which corresponds to the average departure time per car entered by the user. At each time interval, it checks if the time is more or equal to the average departure time and removes cars as needed. Its inputs are the car queues and the average departure time. Its output is the accumulation of waiting times and the modified car queues.

4.1.9 Calculate queue length

This function calculates the maximum, minimum and average car queue lengths. Its input is the accumulation of car queue lengths. Its outputs are the maximum, minimum and average car queue lengths.

4.1.10 Calculate waiting time

This function calculates the Maximum, Minimum and average car waiting times. Its input is the accumulation of car waiting times. Its outputs are the maximum, minimum and average car waiting times.

4.1.11 Store statistics in file

This function stores and prints the statistics calculated (Manxmen and Average queue lengths and waiting times) in a file. Its inputs are the Queue lengths and waiting time statistics and its output is the statistic file.

4.2 Stores

4.2.1 Traffic Light Sequence

In this store, the traffic light sequence (which is written in a file) is stored by initializing function. It is requested when the traffic environment is drawn or updated. It is only updated once by the initializing function as said above and only one function reads it which is the Draw Geometry function.

4.2.2 Car queues

This store is sued to store the car queues for each lane. It is requested when the queues are updated by adding or removing cars. Several functions update this store. It is first initialized by the Initializing Function then it's updated by the increase car function and the decrease car function. The latter functions also read what's in the car queues store before changing anything in it. The only function that only reads this store is the Drawing Geometry function.

4.2.3 Arrival Rate

This store is used to store the arrival rate of the cars entered by the user. It's requested when the increase in cars is needed to be calculated. The only function that updates it is the Get Use Input function. Also, the only function that reads it is the Increase car function.

4.2.4 Running time

This store is used to store the running duration of the simulation entered by the user. It is requested when the program needs to know or to check if the running time of the simulation has ended. The only function that updates it is the Get User Input function and the only function that reads it is the Simulating Queues function.

4.2.5 Time interval

This store is used to store the time for each discrete interval of the simulation. It is requested when the number of cars that arrived is needed to be calculated and to know how much each discrete interval takes time. It's only updated by the Get User Input function and it's read by the Increase car function and the simulating queues function.

4.2.6 Number of lanes

This store is used to store the number of lanes that are going to be used for the simulation and drawn on screen. It is requested when the screen is drawn for the first time and when the queues are simulated to know how many queues are used. The only function that updates it is the get user input function, and the two functions that read it are the draw geometry and simulate queues functions.

4.2.7 Average departure time per car

This store is used to store the average time it takes a car to departure. It is requested to check if it is the right time to remove a car or not the only function that updates it is the get user input function and the only function that requests it is the decrease car function.

4.2.8 Increase status (Queue Length Accumulator)

This store is used to store the accumulation of queue lengths of each queue. It is requested when the program needs to calculate the average queue length. It's updated by the Increase car function and read only by the calculate queue length function.

4.2.9 Decrease status (Waiting times accumulator)

This store is used to store the accumulation of waiting times of each car in each queue. It is requested when the program needs to calculate the average waiting time. Its updated by the decrease car function and read only by the calculate waiting time function.

4.2.10 Max, Min and Average queue length

These stores are used to store the queue length statistics (Maximum, Minimum and average queue length respectively).They are requested when the program is ready to print the statistics in a file. They are updated by the "Calculate Queue length" function and requested by the "Store Statistics in file" function.

4.2.11 Max, Min and Average waiting time

These stores are used to store the waiting time statistics (Maximum, Minimum and average waiting times respectively).They are requested when the program is ready to print the statistics in a file. They are updated by the "Calculate Waiting Time" function and requested by the "Store Statistics in file" function.

5.0 Implementation

5.1 Development platform

Simutraffic was developed using MS visual C++ ®. Also the interface was designed in c++ using the FLTK classes (release 1.1.3), which are a free set of classes under the GNU license.

5.1.1 Header files

1-Fltk header files:

```
#include <FL/fl_draw.H>
#include <FL/fl_ask.H>
#include <FL/Fl.H>
#include <FL/Fl_Box.H>
#include <FL/Fl_Button.H>
#include <FL/Fl_Round_Button.H>
#include <FL/Fl_Int_Input.H>
#include <FL/Fl_Input.H>
#include <FL/Fl_Output.H>
#include <FL/Fl_Widget.H>
#include <FL/Fl_Group.H>
#include <FL/Fl_Float_Input.H>
#include <FL/Fl_Value_Output.H>
#include <FL/Fl_Double_Window.H>
#include <FL/Fl_Gl_Window.H>
#include <FL/Fl_BMP_Image.H>
#include <FL/Fl_Menu_Button.H>
#include <FL/Fl_Image.H>
```

Refer to fltk's documentation at http://www.fltk.org to know what each header file.(Its also included on the projects CD)

2-C++ header files:

```
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <iomanip>
```

5.2 Model Algorithms

The arrival rate and departure rate were modeled using good statistic models. There are explained as follows:

5.2.1 Poisson Distribution Model

The car arrival rate was modeled using the Poisson distribution model. The model is good in modeling events that occur at random times. It's used to model random events that occur in real life (car traffic, telephone calls arriving to a phone booth …etc) which occur it random in time.

The problem is that the Poisson distribution model doesn't give us the number of events per unit time, instead, it give us the probability of a random event happening in one unit time.

So, what was needed to be done is to sample the distribution to give the needed number, which is "how many events occur per unit time".

The algorithm first takes a random number (using rand()) and samples it to generate a random sample that can be used in the program.

5.2.2 Normal distribution algorithm

The departure rate was modeled using a standard normal distribution model. The standard normal distribution is a statistical model which is known to be good to model events that occur between a known average. This model is good to model real life situation where an average is found to approximately the same for approximately any sample size (e.g. height of population, the number of discarded chips in a processor company .etc).

The same problem with the letter distribution occurs here which is that this distribution specifies "the probability of an event in each unit time" and not " how many events occur per unit time" so it must be sampled to give the needed result as it is done in the in the latter algorithm.

The algorithms codes are in the appendix for reference use and include comments on how can they be used.

5.3 Interface design

The interface was implemented using C++ by FLTK classes (release 1.1.3) as noted in the beginning. An example of the interface is shown below: -

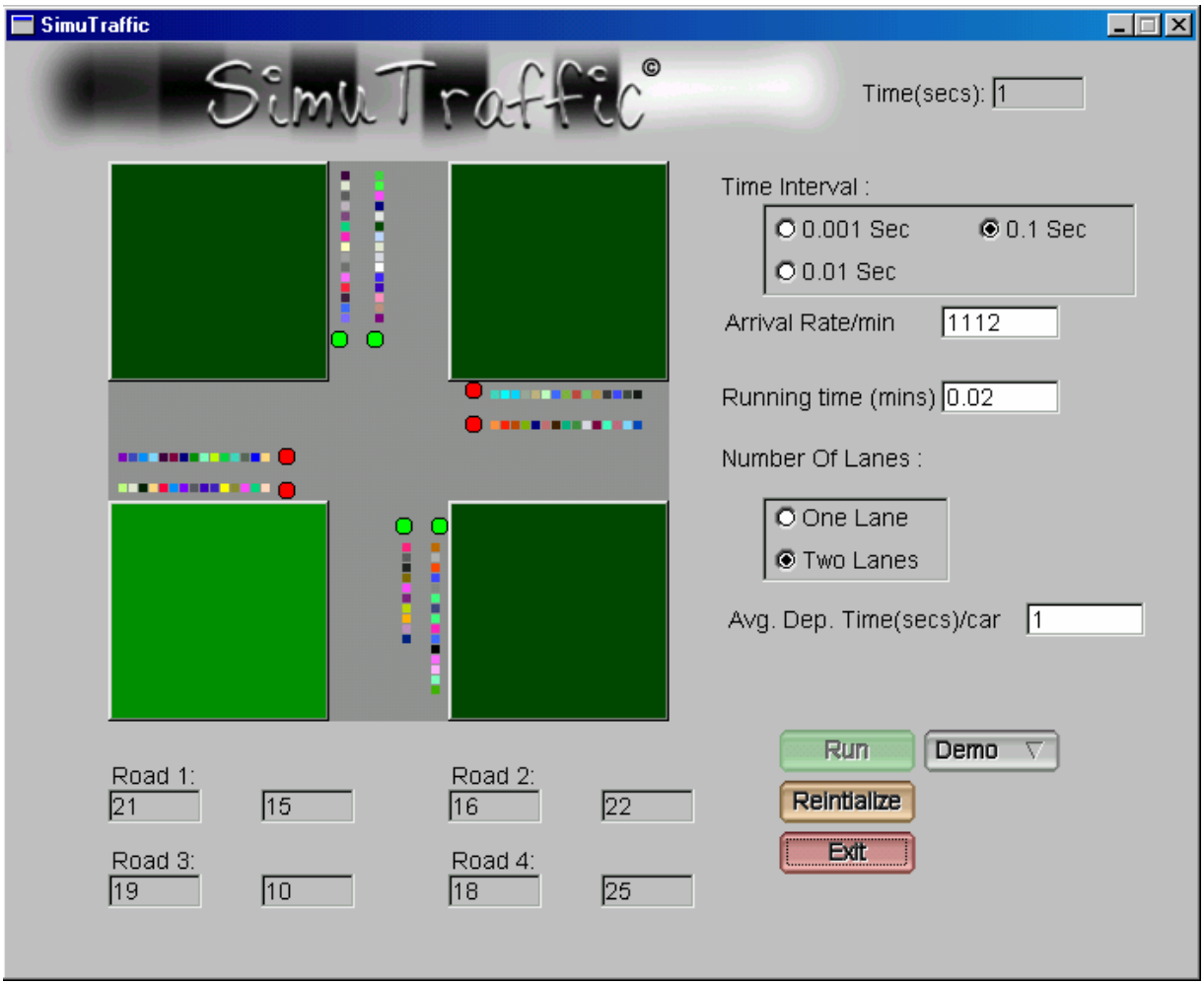


Figure 2 The Interface

The interface will be explained in detail in chapter 7 (Documentation) to show how it is used.

5.4 Output/input example

5.4.1 Input:

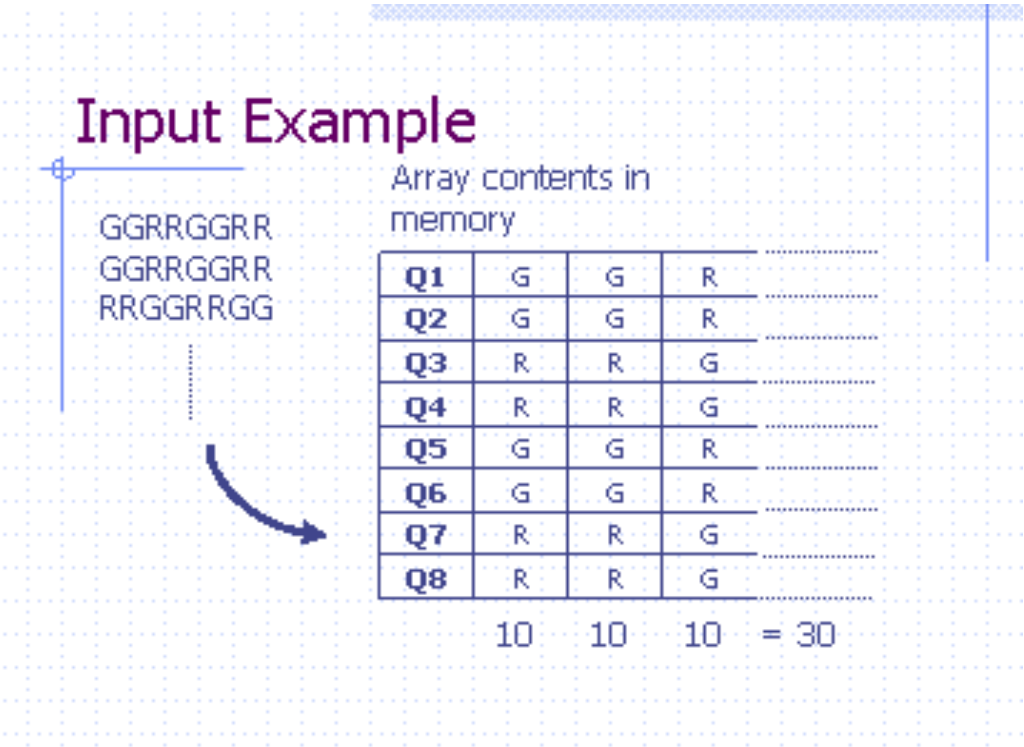


Figure 3 Input Example

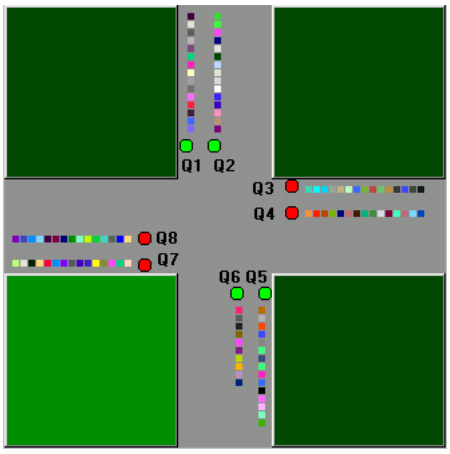


Figure 4 Queue placements

As you can see above, the input will be copied from a file, into the memory by the program. Each line in the input will represent a 10 Sec interval of the simulation e.g. if the user needs Q1 traffic light to be 20 seconds green, he should write the letter G twice separated by 2 lines. Another thing is that the user requires to put 8 characters per line e.g. if he only needs 4 queues (1 lane per road), this means it should be entered this way:

```
G-R-G-R-
G-R-G-R-
R-G-R-G-
R-G-R-G-
```

The first line means that Q1 (the first G) will be green for 10 secs, Q2 will be red for 10 secs, Q3 will be green for ten seconds and Q4 will be red for 10 seconds. The same goes for the other lines. Another this is the dashes (-). This could be anything, because the program wont read these characters. This is based on that the user only needs 4 queues. If the user for instance, selects 2 lanes which means 8 queues and provides this file, the program will wont work in a right way. Also, there must be a character between each G and R if it was 4 queues.

Also,if the user needs more than 10 seconds for each traffic light,the user needs to repeat the line several times in the file (each after the other) to achieve the needed time.(for instance,a 30 second green traffic light means three lines of G for that specific queue)

What the program does with this text file is copy it to an array in memory, read from it as it goes through the simulation (as each 10 seconds go from the simulation time it changes the lights as needed, and if the traffic light didn't change,its recorded by the program to continue on the second line),and repeat it when it arrives at the end of the line.

5.4.2 Output

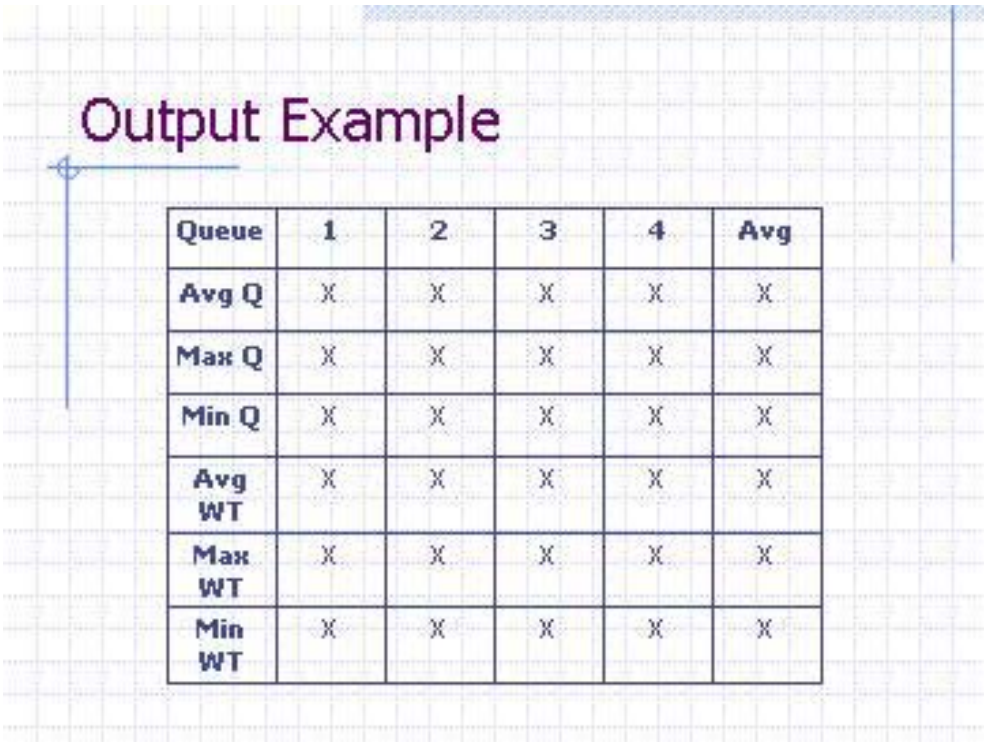


Figure 5 Output Example

This is an example of a statistic table that will be generated by the program after it finishes the simulation. This example is for a one lane per road option.

5.5 Simulation algorithm

The simulation of the queues is done by a simple algorithim,which is main core of how the program works. This algorithm is explained in steps below:

1-The program first read the traffic light sequence to an array in memory.

2- Read the parameters of the values entered by the user from the GUI.

3- Simulate the queues by calling a function (which is a control loop of the program),that calls the increase, decrease and draw cars function for *n* times.(*n* is a number that refers to the number of loops that control loop has to go. Its calculated using the running time value entered by the user)

4- After finishing from the loops, the program then calculates the needed statistical values needed to be written in the table that will be generated by the program

Below is a diagram that shows an overview of the operation.

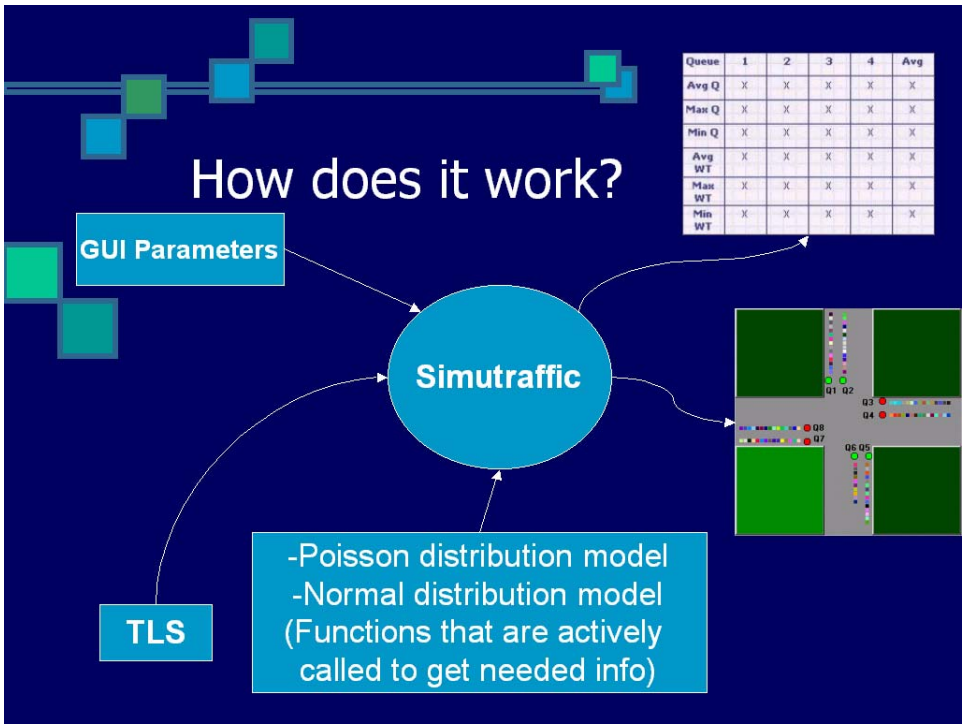


Figure 6 how the program works

5.6 Implementation Issues

Some problems have been encountered, but they were solved successfully. A list of theses problems with a brief explanation follows:

5.6.1 FLTK classes

One of main problems with the implementation is designing interface with FLTK. Because there was no back knowledge about using the classes, learning how to use the classes took some time, but the problem was solved successfully.

5.6.2 Drawing the car queues

One big problem (but easily solved) was the drawing of cars as they arrived and left from the queue. It was thought that this was only done using the (f l_ draw) function of FLTK. But then, it was discovered that ordinary box widgets could be used instead. (Refer to the FLTK documentation for more information at (((http://www.fltk .org)) or the CD that is included with the report).

5.6.3 Average Departure time

The last problem (which was discovered before the last project week) was that all cars in one queue had the same departure time which changes with each interval. This a serious mistake because this doesn't give each car a unique departure time. This means that for each interval in the discrete timing of the simulation, all the drivers will be the same, which is completely unrealistic. This was solved by giving each car a different departure rate as it is with assigning different color code for each of the cars.

6.0 Testing

6.1 Verification:

Simutraffic was tested thoroughly to know if it does meet the required requirements. The results of these testing show that the system satisfies the requirements and behaviors as it was intended do. An example of how to operate the program is shown in chapter 7.

6.2 Validation:

The data that is generated by simutraffic has been validated by using the test cases shown in following sections:

Default values: Time interval *0.1*
Running time *1 min*
2 lanes

6.2.1 Test case 1:

Average arrival: *25 cars/min*
Average departure: *5 sec/car*

	Queue 1	Queue 2	Queue 3	Queue 4	Queue 5	Queue 6	Queue 7	Queue 8	Average
Average Queue Length(cars)	9.54167	6.09667	5.545	10.145	9.595	6.18833	3.23	6.42333	7.09562
Max Queue Length(cars)	20	13	12	17	22	15	6	13	14
Min Queue Length(cars)	1	1	1	1	1	1	1	1	1
Average Waiting Time(sec)	18.5109	20.7369	21.1759	15.8231	21.3315	28.4358	24.5785	21.2089	21.4752
Max Waiting Time(sec)	35	27	40	46	35	33	40	41	37.125
Min Waiting Time(sec)	2	2	10	10	0	1	10	10	5.625

6.2.2 Test case 2:

Average arrival: 25 cars/min

Average departure: 10000 sec/car

	Queue 1	Queue 2	Queue 3	Queue 4	Queue 5	Queue 6	Queue 7	Queue 8	Average
Average Queue Length(cars)	5.76167	15.4633	8.26	10.3117	13.2817	11.4767	8.65167	9.285	10.3115
Max Queue Length(cars)	13	30	17	23	23	22	18	20	20
Min Queue Length(cars)	1	1	1	1	1	1	1	1	1
Average Waiting Time(sec)	∞	∞	∞	∞	∞	∞	∞	∞	∞
Max Waiting Time(sec)	∞	∞	∞	∞	∞	∞	∞	∞	∞
Min Waiting Time(sec)	∞	∞	∞	∞	∞	∞	∞	∞	∞

6.2.3 Test case 3:

Average arrival: 0 cars/min

Average departure: 5 sec/car

	Queue 1	Queue 2	Queue 3	Queue 4	Queue 5	Queue 6	Queue 7	Queue 8	Average
Average Queue Length(cars)	0	0	0	0	0	0	0	0	0
Max Queue Length(cars)	0	0	0	0	0	0	0	0	0
Min Queue Length(cars)	0	0	0	0	0	0	0	0	0
Average Waiting Time(sec)	0	0	0	0	0	0	0	0	0
Max Waiting Time(sec)	0	0	0	0	0	0	0	0	0
Min Waiting Time(sec)	0	0	0	0	0	0	0	0	0

6.2.4 Test case 4:

Average arrival: 1000 cars/min

Average departure: 0.1 sec/car

	Queue 1	Queue 2	Queue 3	Queue 4	Queue 5	Queue 6	Queue 7	Queue 8	Average
Average Queue Length(cars)	492	468.825	514.178	482.702	481.262	487.262	492.088	498.478	489.599
Max Queue Length(cars)	978	961	1012	961	1011	935	981	951	973
Min Queue Length(cars)	1	0	0	0	0	0	0	0	0
Average Waiting Time(sec)	28.3409	27.93	28.2833	29.5093	26.5123	27.9079	28.0692	28.3325	28.1107
Max Waiting Time(sec)	48	48	56	57	46	48	57	57	52.125
Min Waiting Time(sec)	0	0	0	0	0	0	0	0	0

6.3 Test conclusion

As shown in the above test cases, simutraffic generate's good result and are near to what they are expected to be. Also, the GUI and visual interface meet to what have been written in the specification and are working in good condition. Some protection has been implemented (e.g. the simulation will not work if not all the needed fields are not filled) to keep it from crashing unexpectedly.

7.0 Documentation

Simutraffic is a fairly easy program to use thanks to its simple GUI. But to use the program efficiently the user needs to know about each of the program parameters and functions and how to operate them exactly as intended.

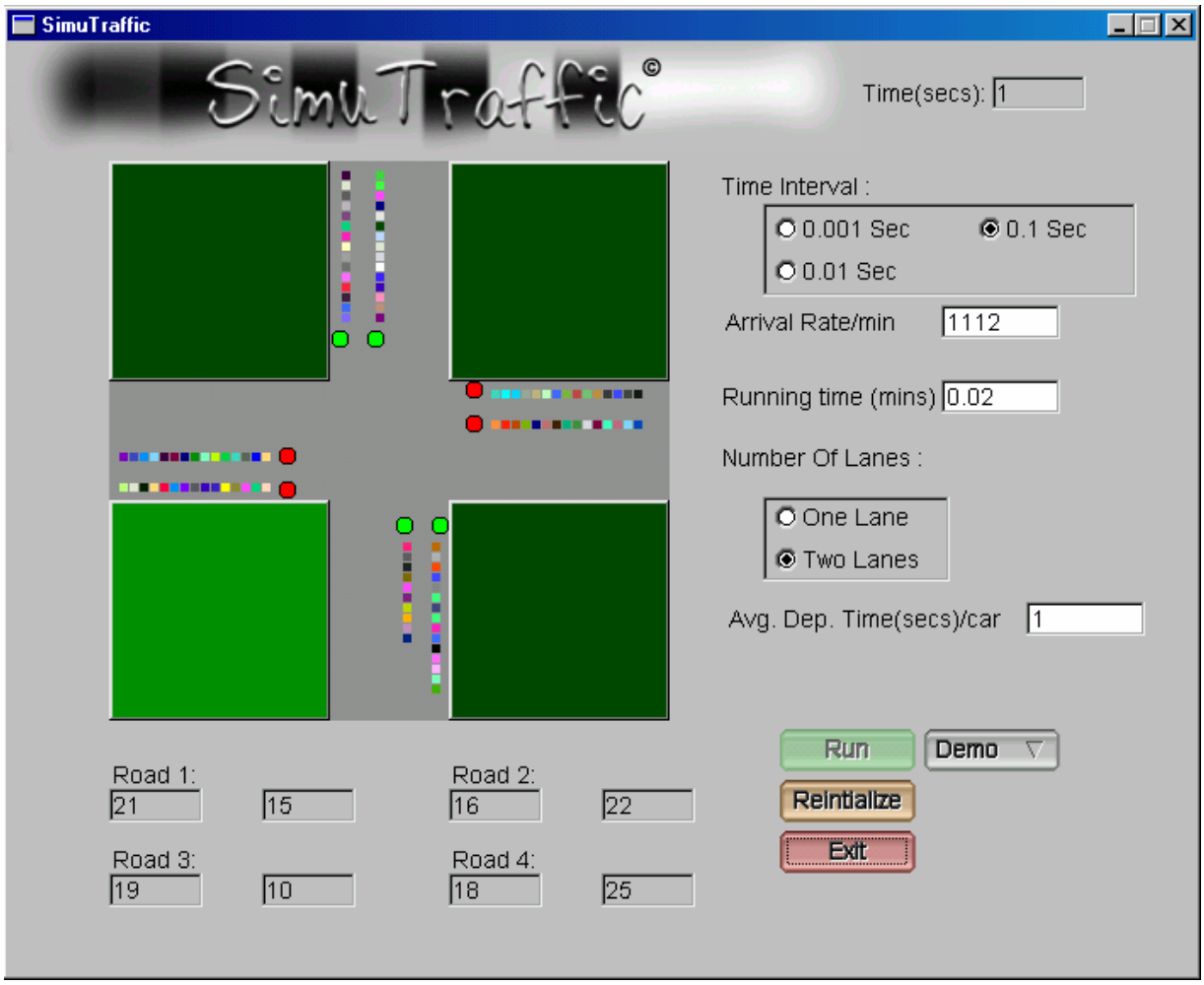
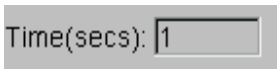


Figure 7 Simuttrafic's interface

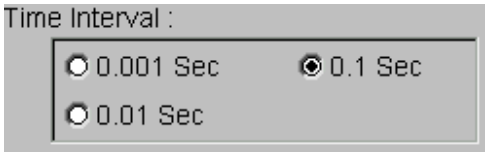
7.1 Interface overview:

7.1.1 Time counter:



This output value shows how much time has passed since the start of the simulation. Its incrementation depends specially on the time interval selected.

7.1.2 Time interval:



This set of buttons lets the user choose a desired time interval for the discrete intervals of the simulation. A smaller number means the simulation runs slower but gives a much finer detailed simulation.

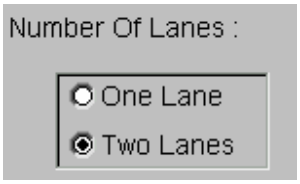
7.1.3 Average Arrival Rate:

This input parameter specifies on average rate of cars per minute. It's used to calculate the number of cars that arrive in each of the queues.

7.1.4 Running time:

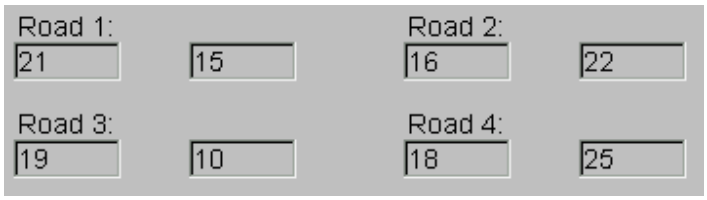
This input specifies how much "simulation time" is needed in minutes. It is used to specify how many time intervals does the user needs the simulation to go through.E.g. if the 0.1 time interval is chosen and 1 minute of running time has been typed the program will go through 600 discrete time intervals.

7.1.5 Number of lanes



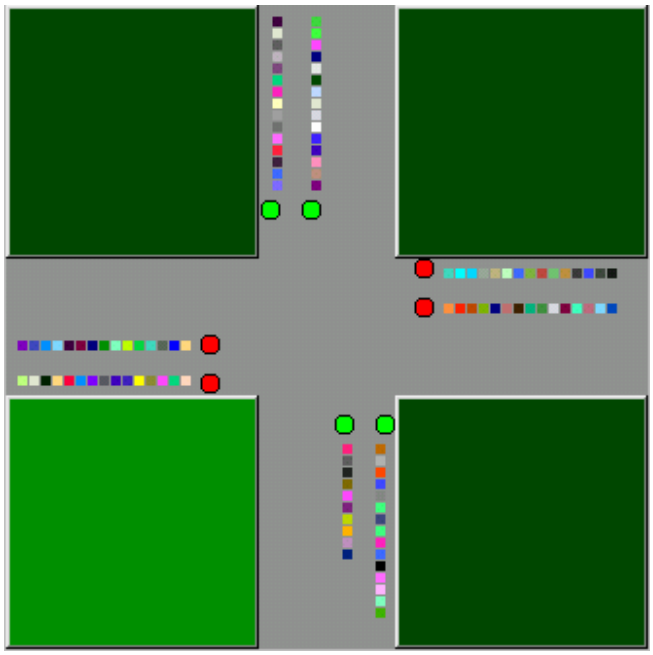
This set of buttons specify the needed number of lanes to perform the simulation on.

7.1.6 Queue value output



This set of output values specifies how much cars are in each car queue of the roads. The boxes are nun before the simulation is executed to specify queue of each box. The queue running is shown is the next section.

7.1.7 Visual Environment:



This is where the car queues are visually represented as they are simulated. The queue length of each queue is shown on the output boxes below it. The numbering of the queue is shown on the image.

7.1.8 Control buttons:

These buttons control the behavior of simutraffic. The first buttons runs the simulation. The second one reinitializes the simulation to its original settings. The

third button exits from the simutraffic program. The last button on the right is a sub menu for ready-made demos to try the program out.

7.1.9 Average Departure time per car:

This input is used to enter the average departure time per car. It defines an average for the departure times of the cars in the queues, which is used by the program to generate a departure time for each car in a queue around that average.

7.2 Operation Overview:

The program operation is very easy. The user is needed to fill the needed parameters and press the run button to observe the behavior. Note that the last input needed is the light-sequencing file for each of the queue. This file's example was shown in chapter 5 in the output and input examples. The file must be named (TLS.txt), and the program must be reloaded every time it is changed. Simutraffic will simulate the queue and then output the result in an Ms Excel® format file. The table will contain the statistics about the simulation of the car queue and their waiting times that are very useful to observe how each lightning strategy is.

8.0 Project discussion

8.1 Summary of work

The program that's described in this report is Simutraffic. Simutraffic is a program that simulates the behavior of queues in cross road traffic junctions. The behavior of the car queues changes and differs depending on the parameters value entered by the user. Simutraffic also visualizes the car queues to show their behavior and then generate a table of the statistics for each of the queues.

The implementation of the program went into several steps, which were successfully executed during the semester's course. This report documents all these steps results from what have been written in the system overview until what diagrams have been drawn. It also shows how each step was done and shows how the program is implemented. Some problems were confronted, but they were solved before and during the last testing week.

Moreover, a documentation of the made program (Simutraffic) has been made to help new users use the program efficiently. It includes an overview of the interface and shows how the program can be operated efficiently.

8.2 Constraints

The programming project has several constraints that are listed in the following sections.

8.2.1 Two Lane limit

Simutraffic has a limit to display and simulate car traffic queues of only 2 lanes. Implementing it with more would be time consuming and not easy.

8.2.2 Drawn cars limit:

Simutraffic has a limit of drawing only 15 cars for each queue on the GUI. This is due to the screen size limitation of the designed GUI.

8.2.3 Traffic Light Sequence Manually validated

The traffic light sequence of the traffic lights is manually validated by the user and not by the program (e.g. the user will be sure that no 2 opposing traffic lights are green at the same time.). It was first thought that an AI (Artificial Intelligence) would be implemented to validate the traffic light sequence as needed. But, it was found that an AI implementation will be very tedious and hard to implement in the time specified for us.

8.2.4 Crossroad junction

The only junction type that was implemented in simutraffic is the crossroad junction.

8.2.5 One distribution model for each queue

Only one distribution model will be used for all queues. This will make the simulation not as realistic as needed but implementing this feature will be very time consuming, as each queue will need its own random number generator.

8.3 Future enhancements

Many enhancements can be added to evolve simutraffic. These sections will show some enhancements that are highly recommended.

8.3.1 Different distribution model for each queue

One of the main problems of simutraffic (which makes it not a very realistic simulation) is its use of one distribution model for all queues. This means that all queues will use one random number generator that tends to make the simulation unnatural. To solve this, it is needed to give each queue its own random number generator each with its own seed.

8.3.2 AI for the traffic light sequences

This enhancement will make the program a very good one by making it find the most suitable traffic light sequence for a specific traffic junction. This will a lot and will be a great addition to the programs other functions.

8.3.3 Different types of traffic junctions with variable lanes

Simutraffic only supports one type of traffic junctions (crossroad) and a limit of 2 lanes per road. The obvious enhancement would be to make it dynamic and let it support what junction the user has drawn for it or from a set of default junctions.

8.3.4 City-sized traffic simulation

Simutraffic can be enhanced by also making it simulate more than one junction at the same time. Literaly, this means that a user can draw a city traffic map for the program and make it simulate the behavior of how each traffic junction affects the other.