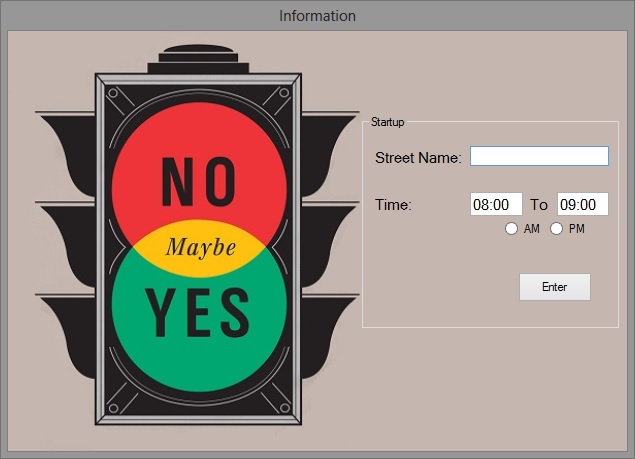
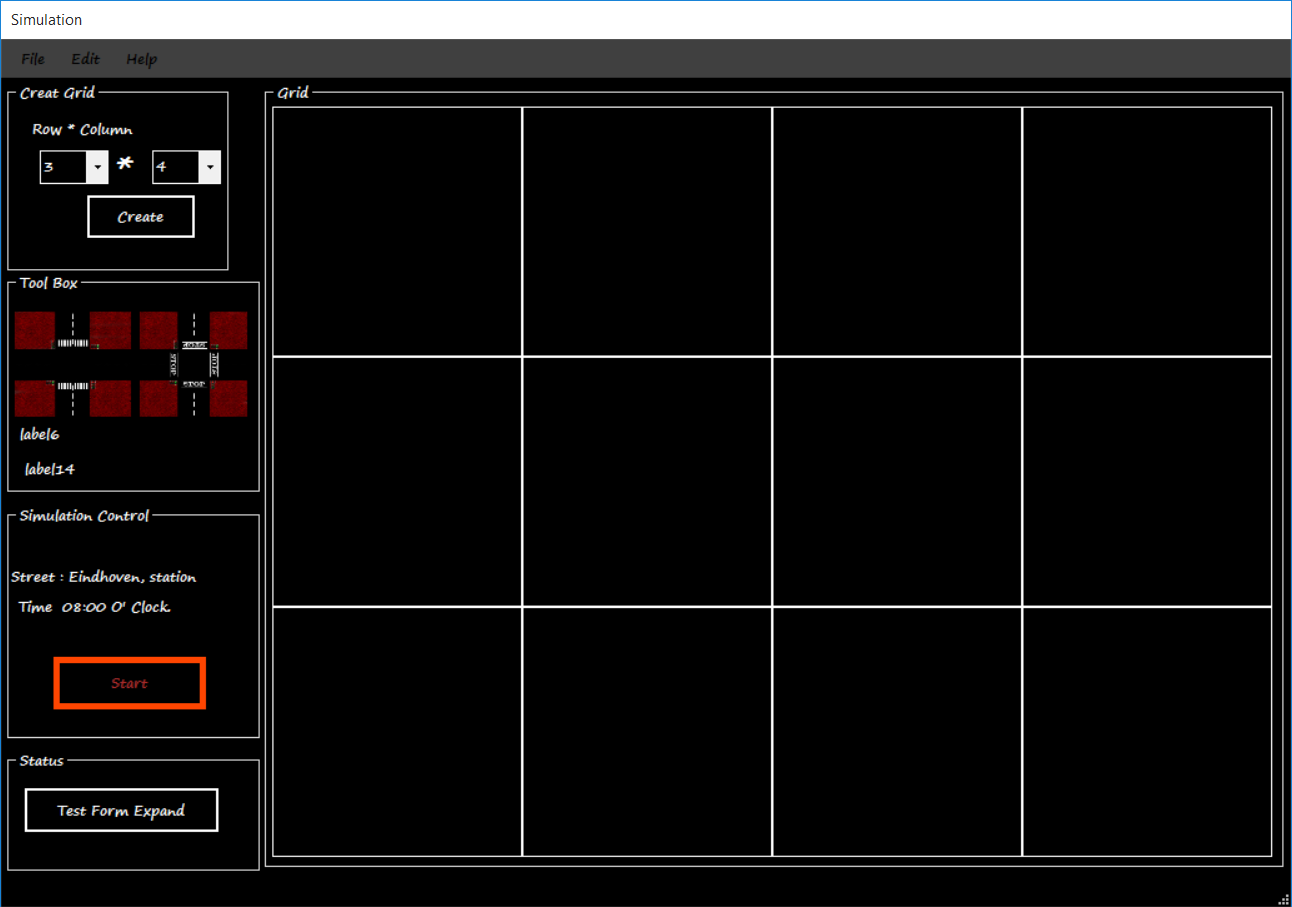
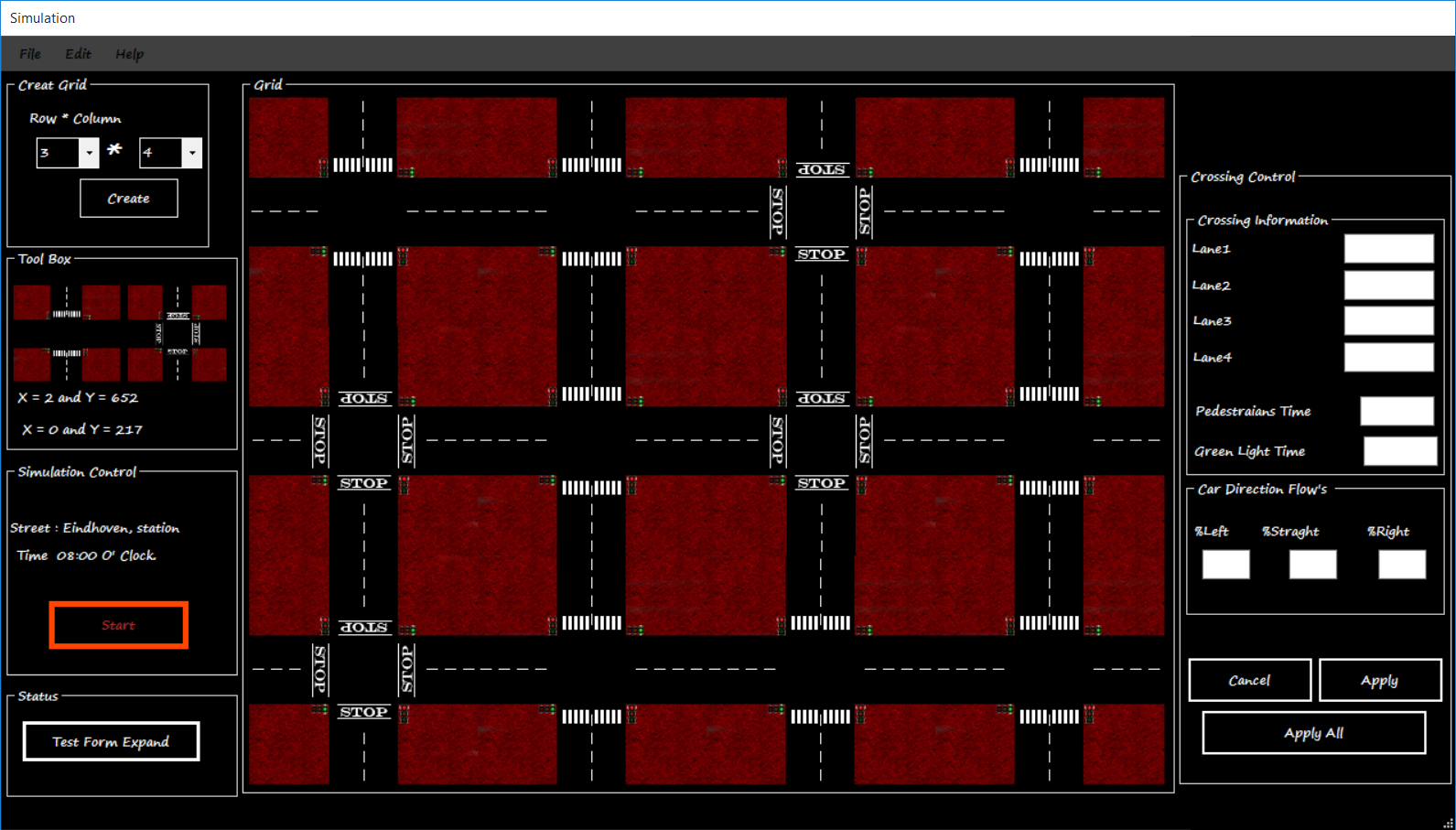
|  |  |
| --- | --- |
| DeSIGN DOCUMENT  Mikaeil Shaghelani Lor - Mervin Vrolijk - Bolarinwa Iruemiobe - Nibras Shawy - Tarwiya Al Ismaili | group C  Class diagram-Description of the classes and their members along with Some sequence diagrams.  **[ProCP- Traffic Light simulator ]**  26/OCT/2015 V2.0.1 |

1. **Table of Contents**
2. Table of content-------------------------------------------------------------1
3. Introduction------------------------------------------------------------------2
4. Class Diagrams--------------------------------------------------------------4
5. Description of classes--------------------------------------------------5-10
6. Interactions /Some sequence diagrams-----------------------------11-12
7. Assumptions and Risks--------------------------------------------------13
8. Appendix1--------------------------------------------------------------14-17
9. **Introduction**

****



**Traffic Light Simulations**

Purpose

This design document describes the architecture and system design of a

Traffic simulator along with elaborating the entities, attributes

and their relations in respect to how they could be used to develop this

program most efficiently in order to achieve the decent simulator that can simulate

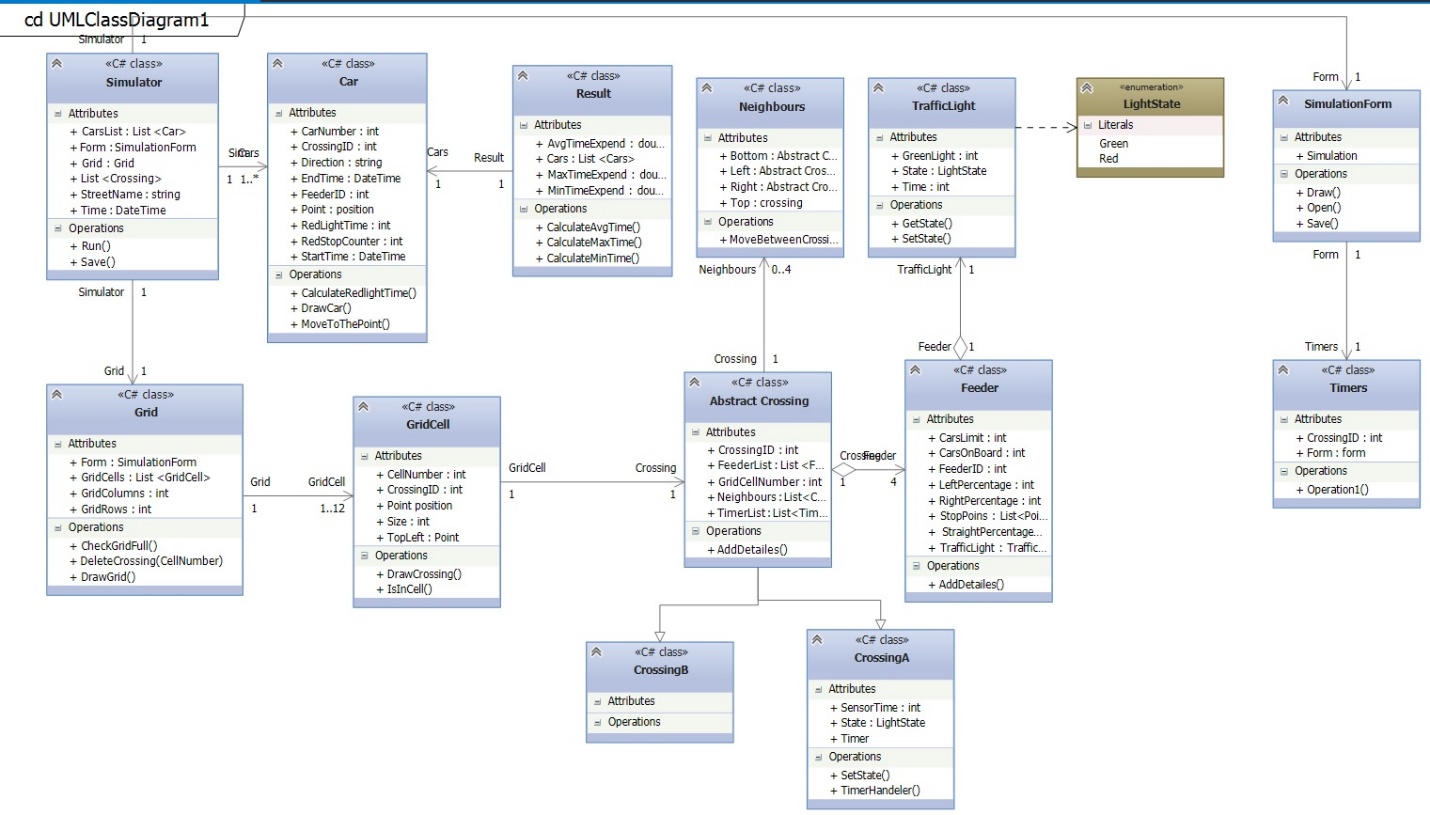
every possibilities.

1. Scope

The objective and the goal of this product is to simulate diffrent traffic flow situations to tests and find out the best usage of the traffic lights in order to decrease the number of accident in the town.

In this design document we provide description of our high level entities among with some design visualizations such as **class diagrams** and few of the most important **sequence diagram**s ,which would explain the interactions for some specific set of actions.

1. **Class diagram**

****

***Figure 3.1***

1. **Description of the classes and their members**

**Classes:**

* Simulation
* SimulationForm
* Grid
* GridCell
* Cars
* Neighbour
* TrafficLight
* Crossing (CrossingA, CrossingB)
* Feeder
* Timers
* Result

**/// Class Members description see figure 3.1**

# Class Simulator

Attributes

// contains the list of car objects

CarsList : List<Car>

//object of form class that enable Simulator to communicate with Form

sForm:SmulationForm

//a grid of type grid

Grid : Grid

// a list of all the crossing that have been placed on the simulator

List : <Crossing>

//name of the street the simulator will be running on

StreetName : string

//the real time of when the simulator will be run

Time : DateTime

Operations

//method to run the simulator

Run()

//method to save the simulator that has already been run

Save()

# Grid

Attributes

//class Form attributes to help draw the grid

Form : SimulationForm

//a list of object GridCell

GridCells : List<GridCell>

//number of columns chosen

GridColumns : int

//number of rows chosen

GridRows : int

Operations

//method to check if all the cells in the grid are full

CheckGridFull()

//method to delete a crossing from a certain cell number

DeleteCrossing(int CellNr)

//method to draw the grid

DrawGrid()

# GridCell

Attributes

//number of certain cell on the grid

CellNumber : int

//the ID of a crossing

CrossingID : int

//class Form attribute to help draw the cell

Form : SimulationForm

//size of the cell

Size: int

//point x and y of the top left corner of the cell

TopLeft : point

Operations

//method to draw a crossing on a cell

DrawCrossing(int GridCellNr, Crossing crossingType)

//check where a certain point is

IsInCell(Point)

# Feeder

The object that contains information about cars, limits and percentages they can go to on the simulation.

Attributes

CarsLimit: int

// The maximum number of cars allowed on the feeds

CarsOnBoard: int

// The number of cars on board at a time

FeederID: int

// The identification number of the feeder

LeftPercentage: int

//The percentage of the cars that should go to the left.

RightPercentage: int

//The percentage of the cars that should go to the right.

StopPoints: List <Points>

// This is where the car will stop on the red light

StraightPercentage: int

//The percentage of the cars that should go to the left.

TrafficLight : Traffic

//The object of type traffic light to be placed on the feeder.

Operations

AddDetails(CarLimit,LeftPercentage,Rightpercentage, straightPercentage)

// The method that accepts the input from the user to create the feeder object based on the entered specification.

# Timer

The timer is used to control the duration of the traffic light

CrossingID : int

// The id of the crossing where the timer is located.

Form: form

The form object.

# Class TrafficLight

This class represents a traffic ligth that can determine a green traffic light (how many time ). And the state of the traffic light.

Constructor

Constructor of the class TrafficLight (Green light , state, time).

Green light: cross point.

State : to determine the state of the traffic light if it is green or not.

Time: to determine the time of waiting in millisecond.

Method:

Public int Getstate:

this method determines the current state of the traffic light and it returns a state.

Public Setstate(int GreenLight, traficColor state):

This method will give ability to execute in the value of the traffic light.

# Class Neighbours.

This class represents the neighbours of every crossing that can determine how many neighbours nearby the crossing.

Constructor

Constructor of the class Neighbours (left, right, bottom, top)

crossing is object from the class Crossing can take a the state (right , bottom, left, top).

Method:

MoveBetweenCrossing(left, right, bottom, top)

This method will determine the movement of the crossing and that depends on which feeder is running and returns the direction of movement.

# Class Cars:

This Class represents the car, every car has to move in the street of the crossing and when there is a direction the car can go into this direction and that is depends on the probability that the user has chosen. if there is a redlight the car has to stop in specific time and after that can complete the way.

Constructor

Car(Time Starttime, int carNumber, int CrossingID, string direction, Time Endtime , int RedlightTime, int RedstopCount, int X, int Y ).

Method:

CalculatRedlightTime (Time Starttime, Time Endtime )

This method is going to calculate the time that the car spends on waiting redlight to finish.

Drawcar (int X,int Y)

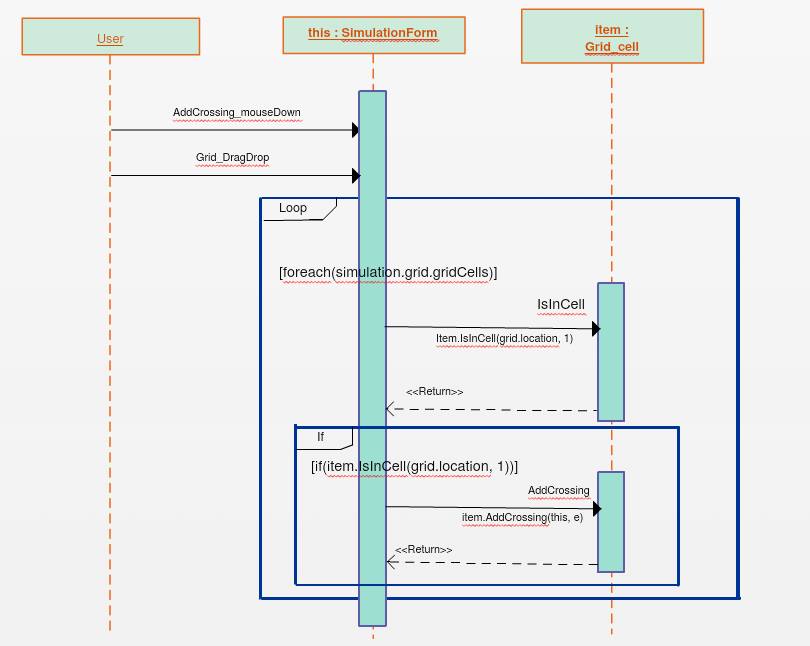
This care is used to draw the car in the simulation and that depends on the position and the direction.

MoveTo the Point(int X, Time Starttime, int Y, Time Endtime )

This mythos is used to move the car to the next point

1. **Interactions/sequence diagrams**

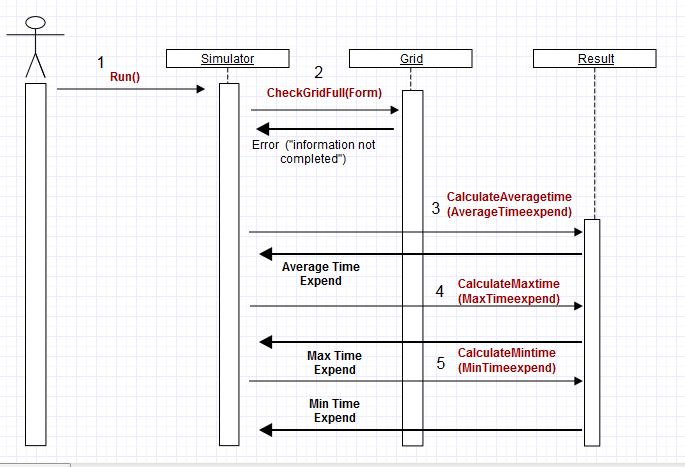
**Draw a crossing on the Grid cell**

****

***Figure 5.1***

* In Figure 5.1 the sequence diagram shows how a crossing will be draw in the grid cell

**Run Simulation**



***Figure 5.2***

* Figure 5.2 shows how simulation will be run

**Draw Grid**

See Appendix 1

1. **Assumptions/Risks**

* While drawing a crossing if the selected grid cell is already filled, the simulator replaces it with the new one.
* Before run the simulation all required crossing detailes has to be completely filled in, if not simulation will not run.

***Appendix 1:***

***Draw Grid sequence diagram***

