Design Document

Final Version



Group B

Todor Tsekov (2228777)

Wen Zhang (2351420)

Jiaqi Ni (2217774)

Yidi Wu (2197301)

Kalina Petrova (2221667)

Jianfei Feng (2204529)

Class: Ei8s1/Ei8s2

Course: Project C-phase

Mentor: Bert Van Gestel

Contents

[1.Class Diagram 2](#_Toc368839600)

[2.Class Description 3-6](#_Toc368839607)

[3.Sequence Diagram](#_Toc368839608) 7-10

[4.Appendix A](#_Toc368839608) 10

Class Diagram:



Class Description:

**Main\_Screen**: this is the form user will see when they open the application. It will show user two options, “load a project” or “create a new project”.

**Methods:**

**+CreateNewProject()**: allow user to create a new project.

**+OpenExsitProject(string path)**: allow user to load an existing project.

**Project\_Grid\_Screen:** the main interface the user will be using when he/she is using this application. There will be a grid, two type of crossing, the simulation control panel and the statistic list box in this form.

**Variables:**

**-simulation**: instance from class **Simulation**.

**-filepath:** the path that the project will be saved/loaded.

-**myGrid:** instance from the class **Grid.**

**-CurrentType**: the current selected type of crossing (Type A or Type B).

**-saved:** local variable, indicating whether this project is saved or not.

**Methods:**

**+Project\_Grid\_Screen():** constructor of this form.

**SetMaxFlow:** the form that will pop up when the user puts a crossing to the grid or when the user want to change the maximum flow of a crossing.

**Variables:**

**-cell:** instance from the class **Cell**, value given by parameters passed through the constructor.

**Methods:**

**+SetMaxFlow(Cell cell):** the constructor of the form.

**SetTrafficLight:** the form that will pop up when the user wants to change the interval of the traffic lights of some specific crossing in the grid.

**Variables:**

**-cell:** instance of the class **Cell**, value given by parameters passed through the constructor.

**Methods:**

**+SetTrafficLight(Cell cell):** the constructor of the form.

**AlterFlow:** the form that will pop up when the user wants to change the flow of the lanes of some specific crossing in the grid.

**Variables:**

**-cell:** instance of the class **Cell**, value given by parameters passed through the constructor.

**Methods:**

**+AlterFlow(Cell cell):** the constructor of the form.

**Simulation:** the class which will take control of the whole simulation.

**Variables:**

**+Grid:** the grid shown in the project\_Gird\_Screen.

**+Status:** the status of the simulation.

**+Logbook:** the logbook which will record all the statistic of the simulation.

**Methods:**

**+Simulation ():** the constructor of the class.

**+SetGrid(Grid g):** set the grid.

**+SetUP():** check whether it is ready to start the simulation.

**+SetStatus(Status s):** change the status of the current simulation.

**+NextSecond():** go to the next second of the simulation.

**Status<enumeration>:** the status the simulation may have.

Initialization: 1, Paused: 2, Running: 3

**Grid:** the class which will create a grid with specific number of specific size cells.

**Variables:**

**-HorizontalCells:** the number of horizontal cells(column numbers).

**-VertivalVells:** the number of vertical cells (row numbers).

**-Orignal:** the left up point of this grid.

**-GridCellSize:** the size of each cell.

**+Cells:**

**Methods:**

**+Grid(Point p, Size s, int v, int h):** the constructor which will initialize the whole grid.

**+Draw(Graphics g, Pen p):** the method which will draw the grid.

**+AddOneCell(Cell c):** add one cell to the grid cell list.

**+RemoveOneCell(Cell c)** remove one cell to the grid cell list.

**+GetCellByPosition(int x, int y):** get the cell from the grid cell list with its location.

**Cell:** the class for cells in the grid.

**Variables:**

**+X:** the location of the cell in X dimension.

**+Y:** the location of the cell in Y dimension.

**+Image:** the image of crossing in this cell.

**+Crossing:** the crossing that the cell contains.

**Methods:**

**+Cell(int x, int y):** the constructor of this class, initialize the cell with its location in the grid.

**+SettingImageAndCrossing(Image I, Crossing c):** set the crossing it contains and the image of the crossing.

**FileHelper:** the class which will help us to save the project to a file or load a project from a file.

**Variables:**

**+Filepath:** the path where the project is saved or where a project is saved.

**Methods:**

**+Load(string path):** load a project from some file.

**+Save(string path):** save the project to some file.

**Crossing<abstract>:** the class which contains the information of the crossings.

**Variables:**

**+Lanes:** the lanes of the crossing, in this case, there are 8 lanes.

**-X:** the location of this crossing in X dimension.

**-Y:** the location of this crossing in Y dimension.

**Methods:**

**+Crossing():** the constructor of this class which initialize the crossing.

**+Initialization():** the method which will initialize the crossing again, will be called when some property of the crossing or the lanes of the crossing change and require reinitialization.

**+SetLocation(int x, int y):** set the location of this crossing, same as the cell which contains this crossing.

**+SetMaxFlowOfLanes(int L1, int L2, int U1, int U2, int R1, int R2, int D1, int D2):**  set the maximum flow of the 8 lanes those around the crossing.

**+SetCurrentFlowOfLanes(int L1, int L2, int U1, int U2, int R1, int R2, int D1, int D2):** set the current flow of the 8 lanes those around the crossing.

**+<abstract>NextSecond(Crossing left, Crossing up, Crossing Right, Crossing Down):** go to the next second, do all necessary changes with the lanes around this crossing.

**CrossingTypeA:**

**Variables:**

**+Status:** the status of the crossing, indicating which lanes are NOT with red traffic lights.

**Methods:**

**+CrossingTypeA():** the constructor of the class, initialize the crossing.

**+ NextSecond(Crossing left, Crossing up, Crossing Right, Crossing Down):** go to the next second, do all necessary changes with the lanes around this crossing.

**StatusTypeA<Enumeration>:** the status of the crossingTypeA.

**Lane3and8**: only lane 3 and lane 8 do NOT have traffic lights as RED.

**Lane4and7**: only lane 4 and lane 7 do NOT have traffic lights as RED.

**Lane2and5**: only lane 2 and lane 5 do NOT have traffic lights as RED.

**Lane1and6**: only lane 1 and lane 6 do NOT have traffic lights as RED.

**CrossingTypeB:**

**Variables:**

**+Status:** the status of the crossing, indicating which lanes are NOT with red traffic lights.

**Methods:**

**+CrossingTypeB():** the constructor of the class, initialize the crossing.

**+ NextSecond(Crossing left, Crossing up, Crossing Right, Crossing Down):** go to the next second, do all necessary changes with the lanes around this crossing.

**StatusTypeB<Enumeration>:** the status of the crossingTypeB.

**Lane3and8**: only lane 3 and lane 8 do NOT have traffic lights as RED.

**Lane4and7**: only lane 4 and lane 7 do NOT have traffic lights as RED.

**Lane2and5**: only lane 2 and lane 5 do NOT have traffic lights as RED.

**Lane1and6**: only lane 1 and lane 6 do NOT have traffic lights as RED.

**Pedestrian:** only pedestrians are allowed to go.

**Lane:** the class for lanes in the crossing.

**Variables:**

**+MaxFlow:** the maximum flow of the lane

**+CurrentFlow:** the current flow of the lane.

**+Light:** the traffic of this lane.

**+Color:** the color of this lane, if the current flow is larger than its maximum flow, then its lane color becomes RED, otherwise it is GREEN.

**Methods:**

**+Lane():** the constructor of this class, initialize the class.

**+SetMaxFlow:** change the maximum flow of this lane.

**+IncreaseFlow:** increase the current flow of this lane.

**+DecreaseFlow:** decrease the current flow of this lane.

**+SetCurrentLightToGreen():** set the traffic light of this lane to Green.

**+SetCurrentFlow(int i):** set the current flow of this lane.

**TrafficLight:** the class for the traffic lights in the lanes.

**Variables:**

**+CurrentLight:** current traffic light color.

**-RedInterval:** the interval for RED light.

**-YellowInterval:** the interval for YELLOW light.

**-GreenInterval:** the interval for GREEN light.

**+RedTimer:** if current light is red, then this is the remaining time for the red light.

**+YellowTimer:** if current light is yellow, then this is the remaining time for the yellow light.

**+GreenTimer:** if current light is green, then this is the remaining time for the green light.

**Methods:**

**+TrafficLight():** the constructor for this class.

**+SetInterval(int g, int r, int y):** change the intervals for the traffic lights.

**+SetCurrentLightToGreen():** change the current light to Green.

**+NextSecond():** go to next second anddo all necessary changes.

**LightStatus<Enumeration>**

**Red, Green, Yellow.**

Sequence Diagrams: **Add crossing:**

**Delete crossing:**



**Play simulation:**



**Pause simulation:**

**Stop simulation:**

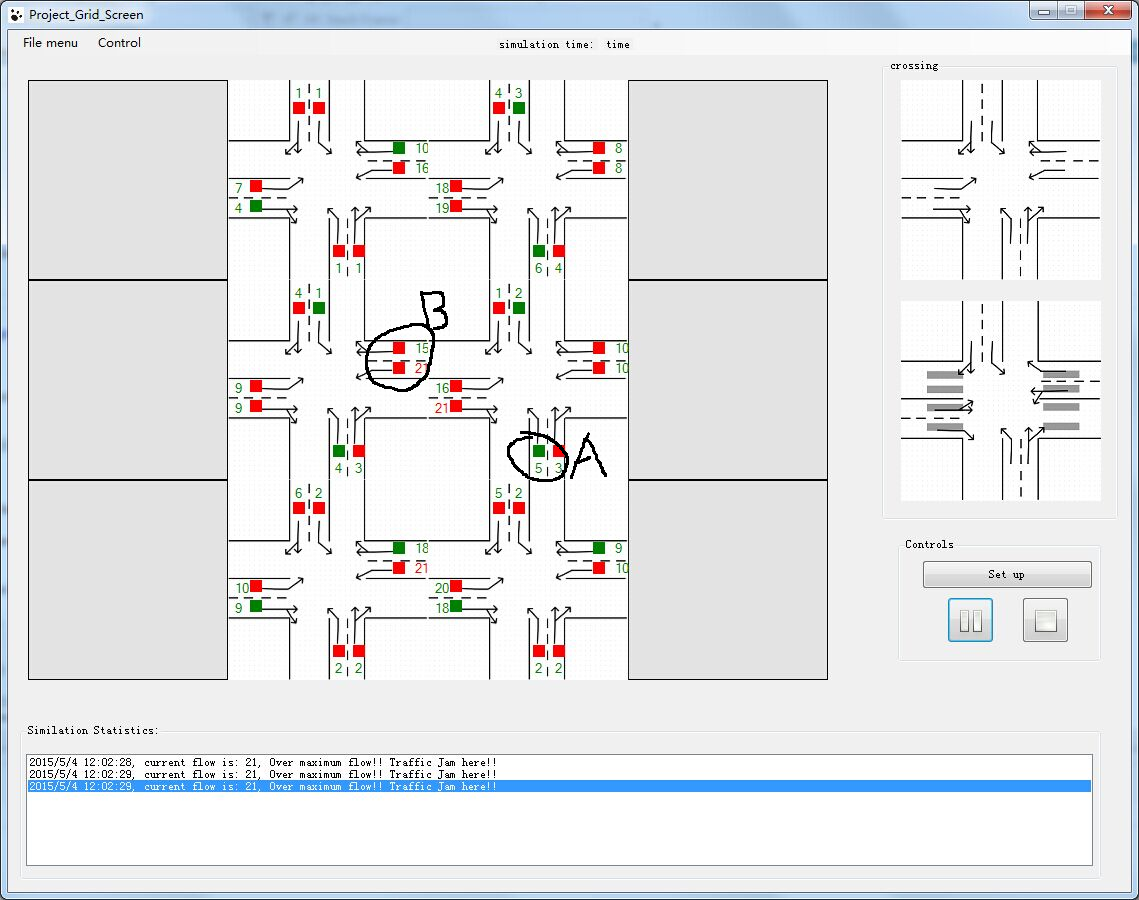
**Save new project:**

**Create new project:**

**Load project:**

**Exit:**

Appendix A: additional explanation for sequence diagram “Play simulation”.



For example, in this case, the flow of left lane in area “A” will decrease, since it has green traffic light.

But the lanes in area “B” will increase since the cars from left lane in area “A” will have to go there.