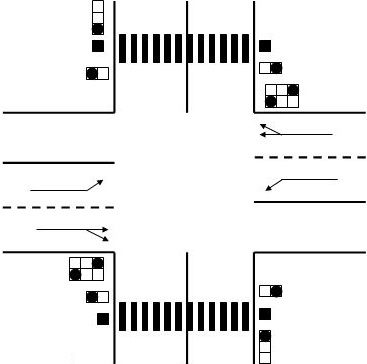
Traffic Simulation



Members:

Govinda

Melese

Ruth

Juarni

Wenhua

Shuying

Contents

[Introduction 3](#_Toc369533119)

[Description of Class Diagram and the Method 5](#_Toc369533120)

[class Traffic 5](#_Toc369533121)

[abstract class Crossing 8](#_Toc369533122)

[class Crossing\_WithPedestarian 10](#_Toc369533123)

[class Crossing\_withoutPedestarian 10](#_Toc369533124)

[class Group 11](#_Toc369533125)

[class Lane 11](#_Toc369533126)

[class Traffic\_light 12](#_Toc369533127)

[class Lane 13](#_Toc369533128)

[class MovingObject 14](#_Toc369533129)

[Sequence Diagram 16](#_Toc369533130)

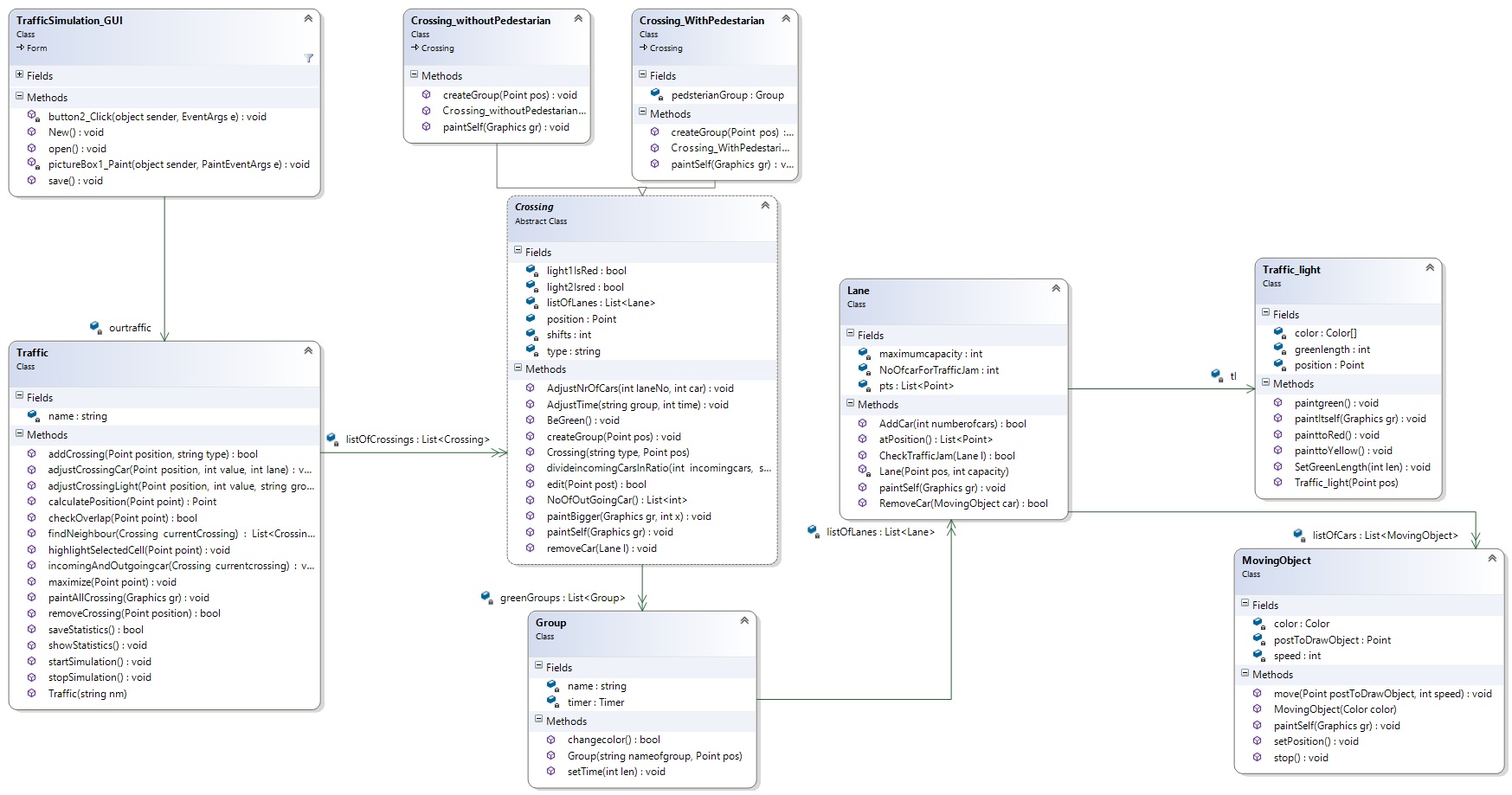
[Add Crossing 16](#_Toc369533131)

[Remove Crossing 17](#_Toc369533132)

[Adjust Traffic (Customize) 18](#_Toc369533133)

# Introduction

This Design document is focused on how software looks like and how it works. This document is all about Class Diagram, description about the method and some Sequence Diagrams.



# Description of Class Diagram and the Method

## class Traffic

{

List<Crossing> listOfCrossings;

String name;

/// <summary>

/// This is the constructor of the traffic class

/// </summary>

/// <param name="crossings">This is the list of crossings contained in a traffic simulation</param>

public Traffic(String nm)

{

this.name = nm;

}

/// <summary>

/// This method will be called to add a crossing to the traffic system

/// </summary>

/// <param name="position">The position of the crossing tht the user wants to place the crossing</param>

/// <param name="type">the type of crossing</param>

/// <returns>returns true if the crossing is successfully added and false otherwise</returns>

public bool addCrossing(Crossing crossing)

{

calculatePosition(position);

//checks if no overlap

//create cros

//listOfCrossings.Add();

// calls incomingandoutgoingcar(created crossing)

return true;

}

/// <summary>

/// This method removes a crossing from the traffic system

/// </summary>

/// <param name="position">The position where the crossing that should be removed is</param>

/// <returns>returns true if the crossing is successfully removed and false otherwise</returns>

public bool removeCrossing(Point position)

{

calculatePosition(position);

return true;

}

/// <summary>

/// This method is called to start the simulation i.e,the cars start moving,the traffic lights start controlling..etc

/// </summary>

public void startSimulation() { }

/// <summary>

/// This method will be called to stop a simulation that is running

/// </summary>

public void stopSimulation()

{

}

/// <summary>

/// This method saves the statistics about a specific simulation,the statistics will be saved in notepad

/// </summary>

/// <returns>The method returns true if the statistics were succesfully saved and false otherwise</returns>

public bool saveStatistics()

{

return true;

}

/// <summary>

/// This method is called to show the statistics that had initially been saved

/// </summary>

public void showStatistics()

{

}

/// <summary>

/// This method will be used to adjust the time for a particular traffic light

/// </summary>

/// <param name="point">This is the position of the traffic light that the user wants to adjust</param>

/// <param name="value">This is the length of green time </param>

/// <param name="group">This is the name of the group which the time is to be set</param>

public void adjustCrossingLight(Point position, int value, string group)

{

//search for crossing from list with that position

//listOfCrossings[position].AdjustTime(group, value);

}

/// <summary>

/// This method will be used to adjust the car for a particular lane

/// </summary>

/// <param name="point">This is the position of the traffic light that the user wants to adjust</param>

/// <param name="value">This is the number of cars on the lane </param>

/// <param name="point">This is the position of the lane in the list where cars is to be set</param>

public void adjustCrossingCar(Point position, int value, int lane)

{

//search for crossing from list with that position

// listOfCrossings[position].AdjustNrOfCars(Lane tobeedited, value);

}

/// <summary>

/// This method will be used to increase the size of the objects in a simulation

/// </summary>

/// <param name="point">The position provided by the user of the objects that should be increased</param>

public void maximize(Point point)

{

calculatePosition(point);

}

/// <summary>

/// This method will be used to find the position of a point(that the user provides) on the grid

/// </summary>

/// <param name="point">The point selected by the user</param>

/// <returns>The calculated position (start point of a cell) on the grid is retuned </returns>

public Point calculatePosition(Point point)

{

Point calculatedPoint=point;

return point;

}

public bool checkOverlap(Point point)

{

calculatePosition(point);

//first calls calculate position method and then check if that cell is empty or not

return false;

}

public void highlightSelectedCell(Point point)

{

calculatePosition(point);

//calculate the position and mark the cell as a selected with some painting

}

/// <summary>

/// This method paints the list of crossings

/// </summary>

/// <param name="gr">The graphics on which the crossings will be painted</param>

public void paintAllCrossing(Graphics gr)

{

}

/// <summary>

/// a method which checks the neighbours of a given crossing corresponding to their direction

/// </summary>

/// <param name="currentCrossing"> the crossing which its neighbours are to be found</param>

/// <returns>a number of crossing which are neighbours to the parameter crossing</returns>

public List<Crossing> findNeighbour(Crossing currentCrossing)

{

List<Crossing> mynbr = new List<Crossing>(4); // the index will satrt according to the direction North, East, South, west

// These will help to know where exactly the neighbour is and if objects are coming in or out of the current crossing

return mynbr;

}

/// <summary>

///

/// </summary>

/// <param name="currentcrossing"></param>

public void incomingAndOutgoingcar(Crossing currentcrossing)

{

// findNeighbour(current);

//it will call the function NrOfOutgoingCar and divideincomingCarsInRatio of the current and neighbour crossings as needed

}

}

# 

## abstract class Crossing

{

List<Lane> listOfLanes;

public Point position;

bool light1IsRed;

bool light2Isred;

int shifts;

string type;

List<Group> greenGroups;

/// <summary>

/// This is the constructor of the crossing class

/// </summary>

/// <param name="type">This is the type of crossing</param>

public Crossing(string type, Point pos)

{

this.type = type;

position = pos;

}

/// <summary>

/// This methods is will be used to edit the crossing

/// </summary>

/// <param name="post">The position provided by the user </param>

/// <returns>Returns true if the eiditing is successful</returns>

public bool edit(Point post)

{

return true;

}

/// <summary>

/// This method will be called to tell a group to change to green in sequence of group list

/// </summary>

public void BeGreen()

{

//loop to call all through group

}

/// <summary>

/// This method will be used to adjust the time of the green light

/// </summary>

/// <param name="group">The name of the group</param>

/// <param name="time">The time it is being adjusted to</param>

public void AdjustTime(string group,int time)

{

// find the group and set the time length

}

/// <summary>

/// This method will be used to adjust the number of cars in a lane

/// </summary>

/// <param name="laneNo">The position of lane in the list whose number of cars is being adjusted</param>

/// <param name="car">The number of cars the lane is being adjusted to</param>

public void AdjustNrOfCars(int laneNo, int car)

{

//finds the lane from list and set number of cars

}

/// <summary>

/// This method removes a car from a given lane

/// </summary>

/// <param name="l">The lane with the car that is to be removed</param>

public void removeCar(Lane l)

{

}

/// <summary>

/// This method paints a crossing object

/// </summary>

/// <param name="gr">The graphics to that the crossing should be painted on</param>

public virtual void paintSelf(Graphics gr)

{

//........

}

/// <summary>

/// This method paint a crossing on separate graphics than the traffic

/// </summary>

/// <param name="gr">The graphics to that the crossing should be painted on</param>

/// <param name="x"> the size that every parameter of the objects should change</param>

public void paintBigger(Graphics gr, int x)

{

//........

}

/// <summary>

/// creates 4 groups of lanes, and if type2 it adds one more

/// </summary>

public virtual void createGroup(Point pos)

{

/\* if()//

{

}\*/

}

/// <summary>

/// this methods calculates the number of cars goingout of this crossing which might be comingin for others neighbours

/// </summary>

/// <returns>list of cars going out in four direction</returns>

public List<int> NoOfOutGoingCar()

{

List<int> myoutgoingcars = new List<int>(4);

// the sequence of the list is according to the direction North out, east out, south out, west out

return myoutgoingcars;

}

/// <summary>

/// it divides the in coming cars according to ratio of 1:2 for 1st and 2nd lane, and the remaing to 3rd lane

/// </summary>

/// <param name="incomingcars"> number of coming in to to the road</param>

/// <param name="direction">the direction of the road</param>

public void divideincomingCarsInRatio(int incomingcars, string direction)

{

//calculates where each lane are and add cars according to ratio

}

}

class Crossing\_WithPedestarian : Crossing

{

public override void paintSelf(System.Drawing.Graphics gr)

{

throw new NotImplementedException();

}

}

class Crossing\_withoutPedestarian : Crossing

{

public override void paintSelf(System.Drawing.Graphics gr)

{

throw new NotImplementedException();

}

}

## class Group

{

string name;

List<Lane> listOfLanes;

Timer timer;

/// <summary>

/// create a lane with a given position and assigns name and default timer 20sec for the group

/// </summary>

/// <param name="nameofgroup"></param>

/// <param name="pos">the position of the lane</param>

public Group(string nameofgroup, Point pos)

{

this.name = nameofgroup;

//create lane wi

}

/// <summary>

/// changes the color of light of lights in the group to green and Red according to timer

/// </summary>

/// <returns>it retuns true if all are changed to red</returns>

public bool changecolor()

{

return true;

}

/// <summary>

/// sets the time length of each light in the group

/// </summary>

public void setTime(int len)

{

//loops through all lanes and changes their green time length...

}

## class Lane

{

Traffic\_light tl;

int NoOfcarForTrafficJam; // cars stopped in the lane beyond maximum

int maximumcapacity;

List<Point> pts; //every lane has start position, stop position, turning position and end position

List<MovingObject> movingobjects; //objects coming into the lane with some constant time

/// <summary>

/// This method paints the Lane object on a graphics

/// </summary>

/// <param name="gr">The graphics that the lane should be painted on</param>

public void paintSelf(Graphics gr)

{

//the lane paints itself

}

/// <summary>

///

/// </summary>

/// <returns></returns>

public List<Point> atPosition()

{ //every lane has start position, stop position, turning position and end position

return pts;

}

/// <summary>

/// This method is used to add a car to a lane

/// </summary>

/// <param name="car">The car that is to be added</param>

/// <returns>Returns true if the car object is successfully added</returns>

public bool SetNoOfCar (MovingObject car)

{

return true;

}

/// <summary>

/// This method removes a car from a lane when it reaches the end of the lane and there is no connection

/// </summary>

/// <param name="car">The car that is to be removed</param>

/// <returns>Returns true if the car is successfully removed and false otherwise</returns>

public bool RemoveCar(MovingObject car)

{

//sets the NoOfcarForTrafficJam using the green/red time and the noofcars in lane and compare it withmaximum capacity

return true;

}

}

## class Traffic\_light

{

Color[] color;

Point position; //x and y axisposition

private int greenlength;

/// <summary>

/// creates a light object

/// </summary>

/// <param name="pos">position of light on lane</param>

public Traffic\_light(Point pos)

{

this.position = pos;

}

/// <summary>

/// paints the color green in the top of the light

/// </summary>

public void paintgreen()

{

//change light to green

//start timer

}

/// <summary>

/// paintsthe color yellow in the middle of light

/// </summary>

public void painttoYellow()

{

//checks the timer

//paints yellow color

}

/// <summary>

/// paints red in the bottom of light

/// </summary>

public void painttoRed()

{

//paints red color

//also send message to crossing???????? can it????

}

/// <summary>

/// calls all paint methods in step first green, yellow, then red

/// </summary>

/// <param name="gr"></param>

public void paintItself(Graphics gr)

{

//do all the painting of traffic light

}

/// <summary>

/// this the length of green paint other than yellow

/// </summary>

/// <param name="len"></param>

public void SetGreenLength(int len)

{

//set length value

}

}

## class Lane

{

Traffic\_light tl;

List<Point> pts; //every lane has start position, stop position, turning position and end position

List<MovingObject> listOfCars; //no of objects coming in the lane in some constant time

int NoOfcarForTrafficJam; //no of cars exceeding capacity

int maximumcapacity;

Lane(Point pos, int capacity)

{

this.maximumcapacity = capacity;

}

List<Traffic\_light> listOfLights;

/// <summary>

/// This method paints the Lane object on a graphics

/// </summary>

/// <param name="gr">The graphics that the lane should be painted on</param>

public void paintSelf(Graphics gr)

{

//the lane paints itself

}

/// <summary>

///

/// </summary>

/// <returns></returns>

public List<Point> atPosition()

{ //every lane has start position, stop position, turning position and end position

return pts;

}

/// <summary>

/// This method is used to add a car to a lane

/// </summary>

/// <param name="numberofcars">The number of cars that are to be added in the lane </param>

/// <returns>Returns true if the car object is successfully added</returns>

public bool AddCars(int numberofcars)

{

//makes changes in the carobject list

return true;

}

/// <summary>

/// This method removes a car from a lane when it reaches the end of the lane and there is no connection

/// </summary>

/// <param name="car">The car that is to be removed</param>

/// <returns>Returns true if the car is successfully removed and false otherwise</returns>

public bool RemoveCar(MovingObject car)

{

return true;

}

/// <summary>

/// This method checks the traffic to see if there is a jam or not

/// </summary>

/// <param name="l">The lane that is to be checked </param>

/// <returns>Returns true if there's a traffic jam and false otherwise</returns>

public bool CheckTrafficJam(Lane l)

{

return true;

}

}

## class MovingObject

{

private Point postToDrawObject;

private int speed;

private Color color;

/// <summary>

/// This is the constuctor of the class MovingObject

/// </summary>

///<param name="color">The color of the object</param>

public MovingObject(Color color)

{

this.color = color;

} /// <summary>

/// This method is responsible for moving the moving objects

/// </summary>

/// <param name="postToDrawObject">The initial position of the object before it is moved</param>

/// <param name="speed">The speed at which the object should be moved</param>

public void move(Point postToDrawObject,int speed)

{

}

/// <summary>

/// This method paints the moving objects on a graphics

/// </summary>

/// <param name="gr">The graphics on which the objects should be painted</param>

public void paintSelf(Graphics gr)

{

}

public void setPosition()

{

}

/// <summary>

/// This method stops a moving MovingObject

/// </summary>

public void stop()

{

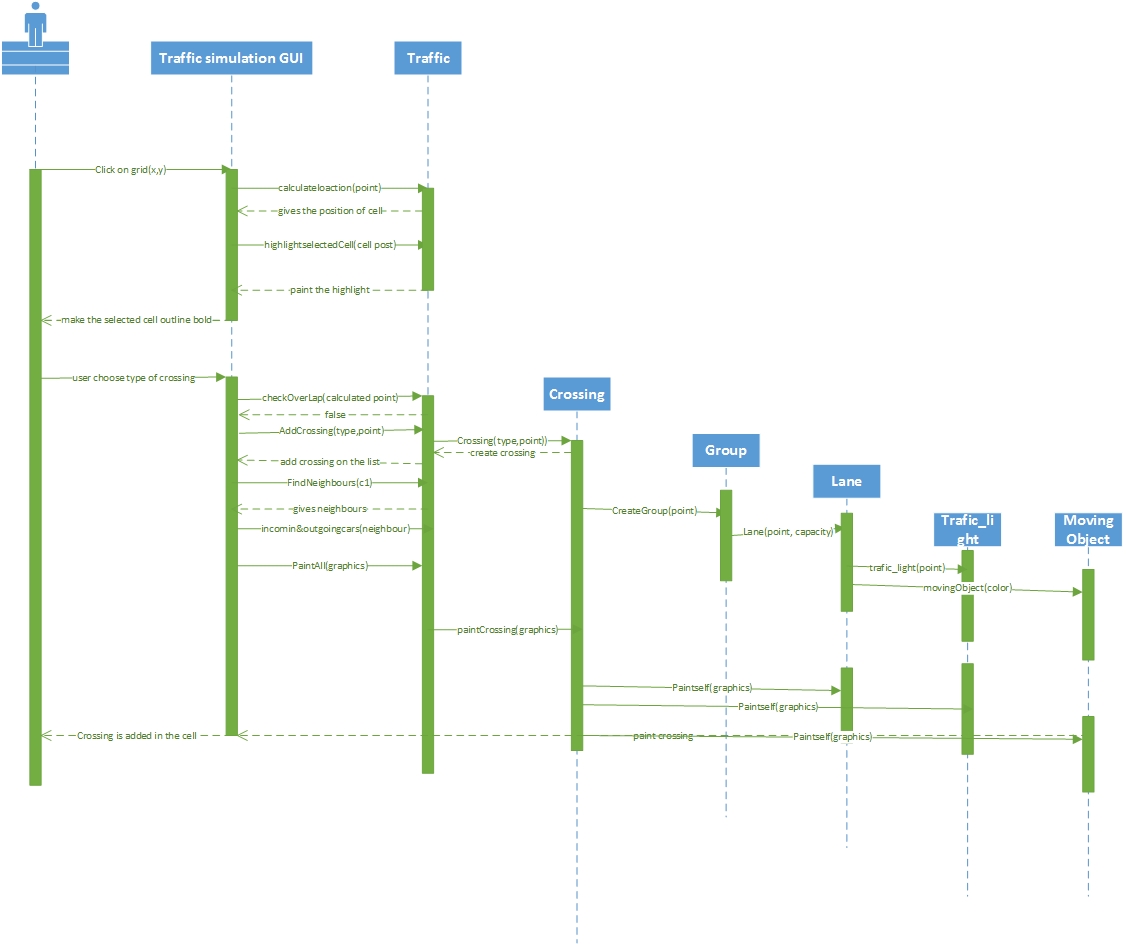
}

}

# 

# Sequence Diagram

## Add Crossing



## Remove Crossing

## Adjust Traffic (Customize)

