3/14/2016

# Design document

# Traffic Lights

# Version I

Monica Stoica Rosen Danev Alexandru Vinerean Blagovest Tsarev Dymtro Bunin Preslav Gherchev Dimitar Vikentiev Ventsislav Yotov

fONTYS UNIVERSITY OF APPLIED SCIENCE

Contents

[Introduction 2](#_Toc445745892)

[Class diagram 4](#_Toc445745893)

[User Interface 4](#_Toc445745894)

[Sequence diagrams 4](#_Toc445745895)

# 

# Introduction

Our group consists of six members: Rosen Danev, Blagovest Tsarev,Monica Stoica, Alexandru Vinerean, Ventsislav Yotov and Dmyro Bunin students of class EI7S2.

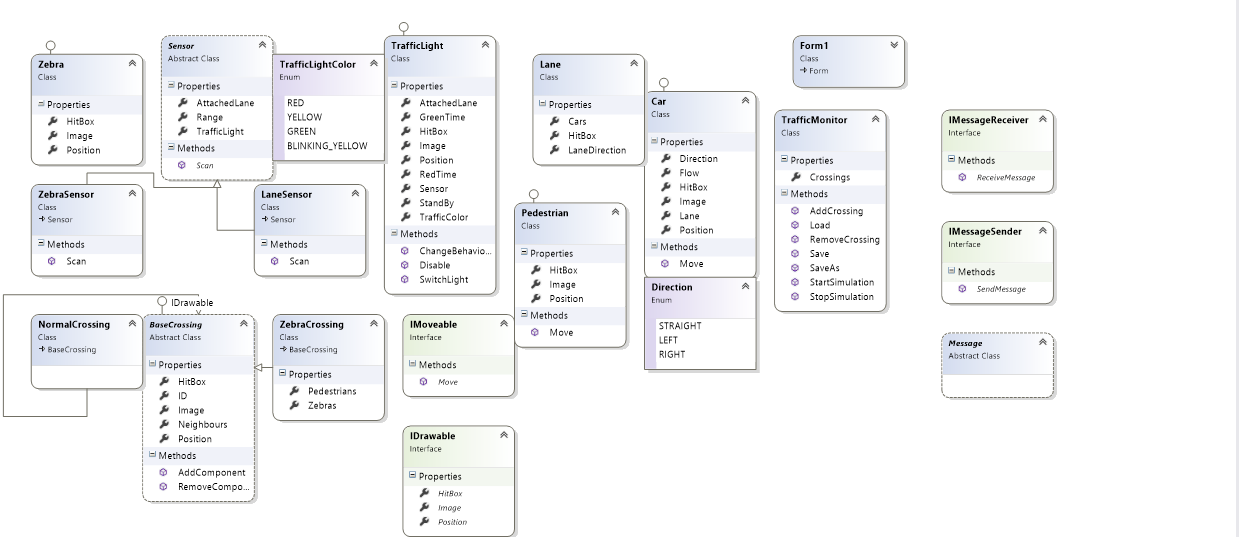
The purpose of this document is to identify the design of our traffic simulator software system. The system’s structure will be defined using the Unified Modelling Language.

A class diagram and the description of each class’ members such as fields, properties and methods will represent the first chapter. The class diagram provides an overview of the software system by describing the classes inside the system and the relationship between them.

Moreover, to have a better understanding of how the objects interact with others in a particular scenario (use-case), few sequence diagrams will be explained.

In the ‘User-interface’ chapter, the process of creating a simple traffic situation is explained.

# Class diagram



Below, you can find a description of the fields, properties and methods used in our class diagram.

***TrafficMonitor***

**Properties:**

* Crossings : List<Crossing> - represents a list of the crossings drawn on the screen

**Constructor**

* TrafficMonitor()
* TrafficMonitor(Serialization info, StreamingContext context)

**Methods:**

* AddCrossing(Crossing c, int locx, int locy): bool – Adds a crossing to the list and draws it on the screen in the specified location. Returns true if the component has been successfully added; otherwise false.
* Load(String path):TrafficMonitor – Deserializes a saved file and returns the loaded traffic simulation.
* RemoveCrossing(Crossing c): void –Removes a component from the list.
* SaveAs(Network net, String path): static void – converts the file to binary and saves it in .xml format
* SartSimulation():void – starts the simulation.
* StopSimulation():void – stops the simulation.
* GetObjectData(Serialization info, StreamingContext):void – objects to be serialized and added to the serialization info.

***Car***

**Properties:**

* Direction:Direction – returns the direction of the car
* Flow:int – return the number of cars that are used for the simulation.
* HitBox:Rectangle – returns a rectangle representing the current position of a car
* Image: Image –returns the image of the car
* Lane: Lane – returns the lane on which the car is positioned.
* Position: Point – returns the current position of the car

**Methods:**

-Move():void – enables the cars to move.

***Pedestrian***

**Properties:**

* HitBox:Rectangle –returns a rectangle representing the position of the pedestrian
* Image:Image – returns the image of the pedestrian.
* Position:Point – returns the current position of the car.

**Methods:**

* Move():void – enables the pedestrians to move.

**TrafficLight**

**Properties:**

* AttachedLine:Line – returns the line for which the traffic light functions
* GreenTime:Int – returns the amount of seconds of green time of a traffic light
* HitBox:Rectangle – returns a rectangle representing the position of the traffic light
* Image: Image – returns an image representing the traffic light
* Position:Point – returns a point with the X and Y coordinates of the traffic light
* RedTime:int – returns the amount of seconds of red time of a traffic light
* Sensor:Sensor – returns the sensor of the traffic light
* StandBy:Bool – returns false if the traffic light works within normal parameters (not yellow intermittent) and true if the traffic light is disabled
* TrafficColor:Color – returns the current color of the traffic light

**Methods:**

* ChangeBehaviour():void – Changes the behavior of a traffic light into a sensor?
* Disable():void – puts the traffic light on standby.
* SwitchLight():void – switches between the colors of the traffic light depending on the current color.

***Sensor***

**Properties:**

* Range:int – returns the maximum distance which will trigger the sensor
* TrafficLight:TrafficLight – returns the traffic light which the sensor will call

**Methods:**

Scan():bool – scans and checks if there are any cars/pedestrians coming. If yes, it changes the color of the traffic light (or add another method to do that in case this method returns a bool?)

***LaneSensor: Inherits from Sensor class and implements differently the method Scan(). The difference is that its checking the assigned lane if there are any cars approaching. The class also contains an AttachedLane:Lane property which returns the lane for which the sensor is triggered.***

***ZabraSensor: Inherits from Sensor class and implements differently the method Scan(). The difference is that it is checking the assigned zebra if there are any pedestrians who wish to cross.The class also contains an AttachedZebra:Zebra property which returns the zebra for which the sensor is triggered.***

***Zebra:***

**Properties:**

* HitBox:Rectangle – returns a rectangle representing the position of the zebra

***BaseCrossing: abstract class***

**Properties:**

* HitBox:Rectangle – returns a rectangle representing the position of the crossing
* ID:int – returns the identification number of the crossing (from 1 to 12)
* Image: Image – returns the image of the crossing
* Neighbors:List<Crossing> returns a list with the connected crossings
* Position:Point – returns a point with the current position of the crossing

**Methods:**

* AddComponent()?
* RemoveComponent?

***NormalCrossing: This class inherits from the BaseCrossing class and represents the first type of crossing without zebras.***

***ZebraCrossing: This class inherits from the BaseCrossing class and represents the second type of crossing with zebras and contains the following additions:***

**Properties:**

* Pedestrians:bool – returns true if there are pedestrians who wish to cross the street
* Zebras:List<Zebra> -returns a list of zebras.

***Message: abstract class***

***IMessageSender***

**Methods:**

* SendMessage():void –Sends a message to one of the components announcing that its state has changed

***IMessageReceiver***

* ReceiveMessage():bool – confirms that the component has received the message

***Direction: Enum –STRAIGHT, LEFT, RIGHT***

***TrafficLightColor: Enum- RED, YELLOW, GREEN, BLINKING\_YELLOW***

***IDrawable: interface***

**Properties:**

* *Hitbox: Rectangle*
* *Image: Image*
* *Position: Point*

***IMovable: interface***

***Methods:***

* *Move():void*

# User Interface

# Sequence diagrams