CSE1242 Computer Programming II, Spring 2024

JavaFX Project: TRAFFIC SIMULATOR GAME

date submitted : 10/05/2024

Name-Surname	ID
Semih DAĞDELEN	150123023
Baran ÇİL	150123020
Ahmet Faruk DEMİR	150122035

Problem Definition

In this project our mission was to implement a traffic simulator game which players can influence the traffic by turning the traffic lights on or off. Firstly we needed to have a map which had buildings, road tiles and traffic lights. Secondly we were ought to implement car spawning logic and creating traffic.We were given 5 levels and .txt files for each one. We read from files and constructed our levels accordingly.

Implementation Details - UML Diagrams

Building class UML diagram

Building	
~	buildingType : int
~	rotation : int
~	colorIndex : int
~	gridX : int
~	gridY : int
~	pane : Pane
~	rectangle :Rectangle
~	rectangle1 : Rectangle
~	rectangle2 : Rectangle
~	circle1 : Circle
~	circle2 : Circle
~	colors : Color[4]
-	SQUARE_SIZE : double {readOnly}
+	Building(int buildingType, int rotation, int colorIndex, int row, int col)
+	getPane() : Pane
+	getStrokeColor(): Color
+	getColor() : Color
-	createCircle() :Circle
-	createRectangle() : Rectangle
+	createBuildingType0() : void
+	createBuildingType1() : void
+	createBuildingType2() : void
+	createSquare() : Rectangle
+	getters / setters

Building class has a constructor that takes 5 parameters,**buildingType** that specifies the type, **rotation** for rotating the pane that we used to place our rectangles,**colorIndex** for choosing the color of the rectangle,**row** and **col** parameters for placing the pane that has our rectangle

to gridpane, which is our main pane.In constructor we create rectangle objects based on the parameters and add it to objects pane,which is a normal Pane type.

getPane method returns a pane,which we already constructed a building and replaced it into the objects pane. This is a crucial method for almost every object in the project. Main objective of returning a pane was to add it into our main pane and being able to freely rotate the rectangles.

createBuildingType0(), createBuildingType1(), createBuildingType2() methods are used in constructor to create specific types of buildings.

createCircle() , createSquare() , createRectangle() by the names indicate, these methods are used to create circles rectangles and squares which make up buildings.

getColor() and **getStrokeColor()** methods return a rgb value based on colorIndex property

TrafficLight class UML diagram

TrafficLight	
~ ~ ~ ~ ~ ~	startX : double startY : double endX : double endY : double line :Line pane :StackPane
+ + +	TrafficLight(double startX,double startY, double endX,double endY) getPane() : StackPane getters/setters

TrafficLight constructor has 4 fields, **startX,startY,endX,endY**. these fields are used to create a line and a circle which are on top of each other. we added line and circle to TrafficLight' s **StackPane** and then returned it using **getPane()** method in test class.

Car class UML diagram

Car	
~ ~ ~ ~ ~ ~ ~	path :Path pane : Pane pathTransition : pathTransition car : Rectangle time : double X :double Y: double

~	pathLength : double
+	Car (Path path, Pane pane)
+	getPathLength() : double
+	calculatePathLength(Path[] paths) :double
+	equals(Object o) : boolean
+	getters/ setters

Our **car** constructor takes two parameters, **path** which the car goes on and **pane** which is the parent that car is added into.In car constructor we create a rectangle by specifying height width and color of the car and we set the **orientation, interpolator** and **cycle count** of path transition the car takes.

In **getPathLength()** method we search for path elements one by one and calculate the distance from start to end.This method is a crucial one because we used this to overcome the different car speeds problem.

RoadTile class UML diagram

RoadTile	
~	roadType : int
~	rotation : int
~	GridX : int
~	GridY : int
+	RoadTile(int roadType, int rotation,int GridX,int GridY)
+	getRoadTile() : Pane
+	getters / setters

RoadTile constructor takes 4 arguments,roadType argument decides the type,rotation is the value that decides pane' s rotation value, GridX and GridY decide the position of the roadTile relative to gridPane

getRoadTile() method constructs the shape of the roadTile using rectangles and calculating x and y coordinates of RoadTile.

Metadata class UML diagram

Metadata	
~	x : double
~	y : double
~	numOfGridX : int
~	numOfGridY : int
~	destinationsToWin : int
~	crashesToLose : int
~	numberOfPath : int
~	grids : Rectangle [][]
~	gridPane : GridPane
+	Metadata(double x, double y , int numOfGridX, int numOfGridY , int
+	destinationsToWin , int crashesToLose , int numberOfPath)
+	getPane : GridPane
+	getters / setters

Metadata class is used to construct the map that all our buildings, roadTiles, TrafficLights and cars land into.We set our map from **gridPane**,we adjusted colors of tiles and set borders.

Test class UML diagram

Test	
-	primaryStage: Stage
-	cars : Arraylist <Car>
-	trafficLights: Arraylist<TrafficLight>
-	pane :Gridpane
-	paneForLabel: Pane
-	pathArray :Path[]
-	textLabels : Label
-	winCondition: Label
-	loseCondition: Label
-	metadata : Metadata
-	time : double
-	crashCount: int
-	scoreCount : int
-	crashesToLose : int
-	destinationsToWin : int
-	level: int
-	nextLevelButton: Button
-	backToMenuButton: Button
-	imageview2: ImageView
-	winHandled: boolean
-	levelCompleted: boolean

+	start(Stage primaryStage) :void
+	readFile(int level): void
+	main(String[] args) : void
-	spawnCar() : void
-	calculateDistance(double x1, double y1, double x2, double y2): double
-	createTraffic() : void
-	update() : void
+	handleWinCondition(): void
+	moveNextLevel(): void

Test class is the class where we combine all of the classes to get the desired result.

First of all we made a start page using a different scene and pane. we set up a button that changes scenes into level 1.We used **scanner** in readFile to read from level files and make our levels one by one. we took the words line by line into an array and constructed our objects accordingly.
We took our objects which were already created to a pane and we used objects getPane() methods to get the pane and add it into our pane. For **trafficLight** issue, we used **StackPane** instead of normal Pane and we used setLayout methods to carry the stackpane into its desired position and that fixed traffic light issue.

We put score and crashes count up left and we updated the crashes and score count. Whenever a **pathTransition** is finished we upped the score and whenever two cars **collided** we upped the crash count.

Handling the interactions between cars and general car movement checking happened in **spawnCar, createTraffic and update** methods. We used the code that we were given in term project guideline and modified it. We had nested car loops checking car and otherCar. We used intersects() method to check whether a car touched otherCar or not. And if they did we updated crashCount. We had another loop that checked every traffic light and if car intersected the light and light was green cars would move on. Else they all stop and wait in a line.

calculateDistance is a helper method for calculating between two lines.

handleWinCondition and **moveNextLevel** methods are used for switching the panes whenever user won or lost the game.We clear the pane and call readFile method to construct the other level.

Which parts are complete/incomplete in our project?
In our project everything except some traffic control bugs are complete and set. Our problem is if a car is from a different path it does not see the other car and stop at a traffic light and crash the car waiting in the traffic light

What are the difficulties we have encountered during the implementation?
Car handling situations took us a lot of time since it was a little bit more complicated than other parts.Traffic light issue also costs us a lot of time, we knew the problem but we had to discover a lot of methods in order to overcome the problem.

TEST CASES
Level 1

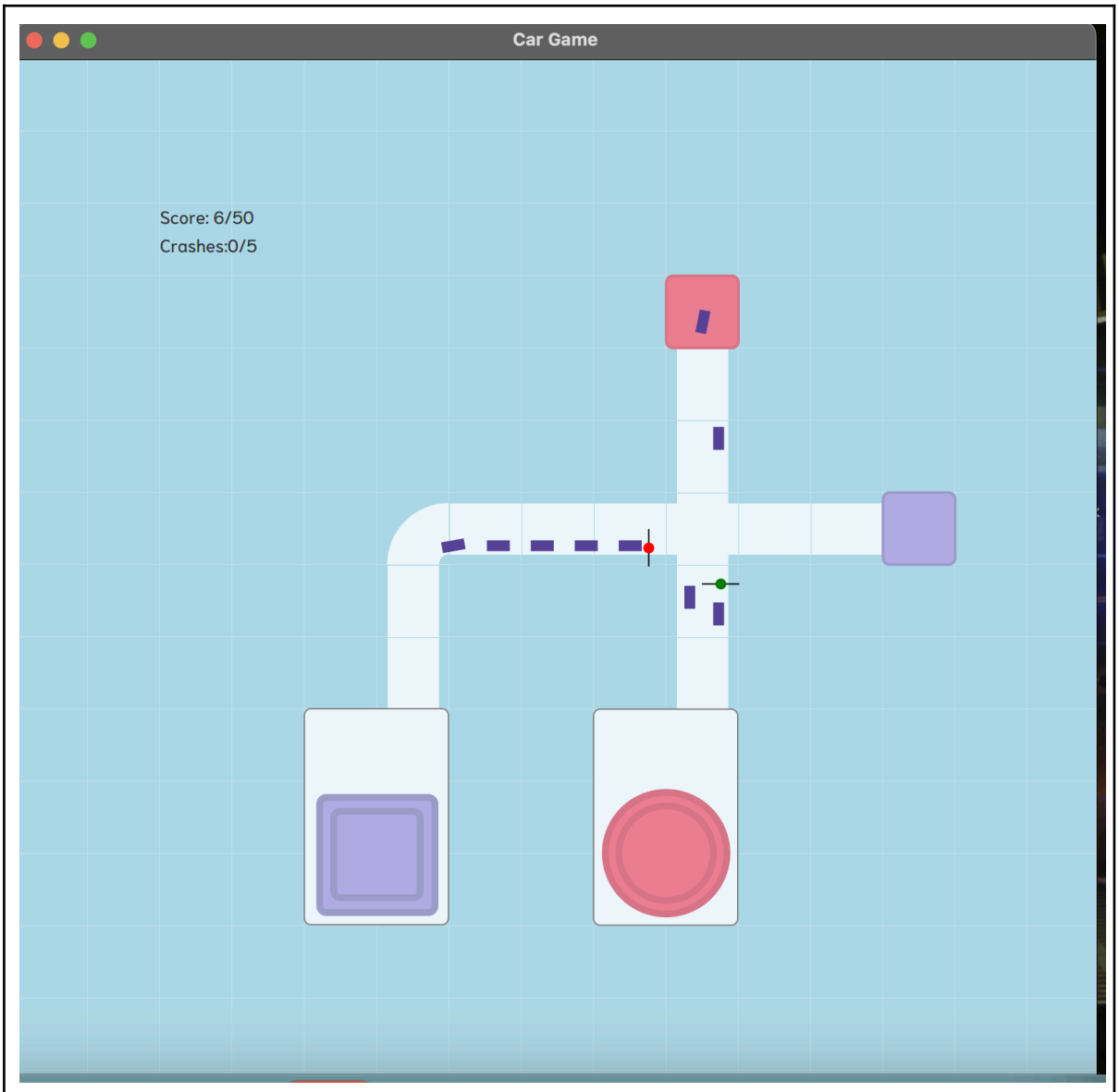

Image 1 - Footage of game 1, cars waiting in a line, traffic light is working as expected

Level 2

Image 2 - Game over text being displayed after user failed the game.

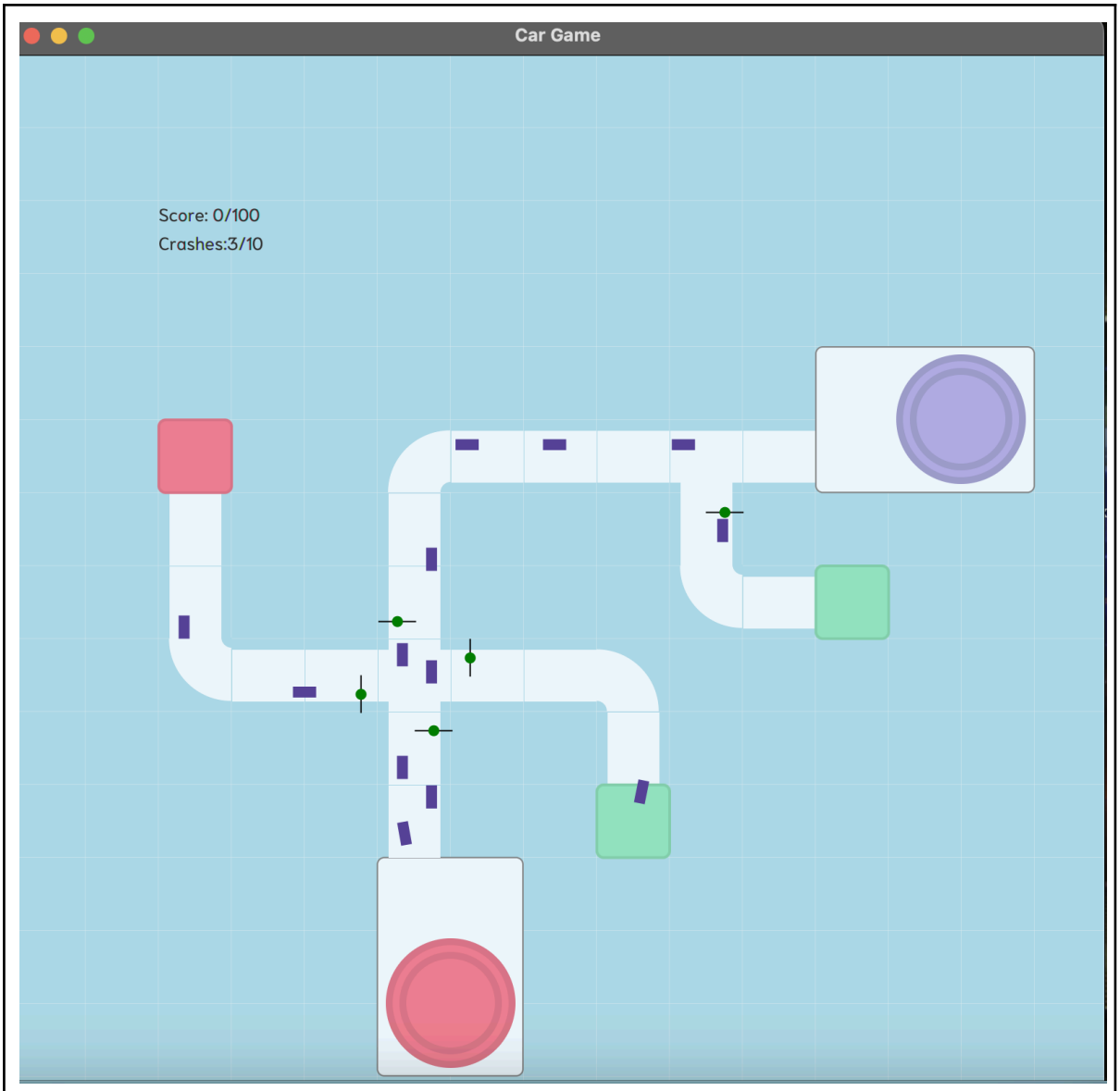

Image 3 - level 3 traffic running

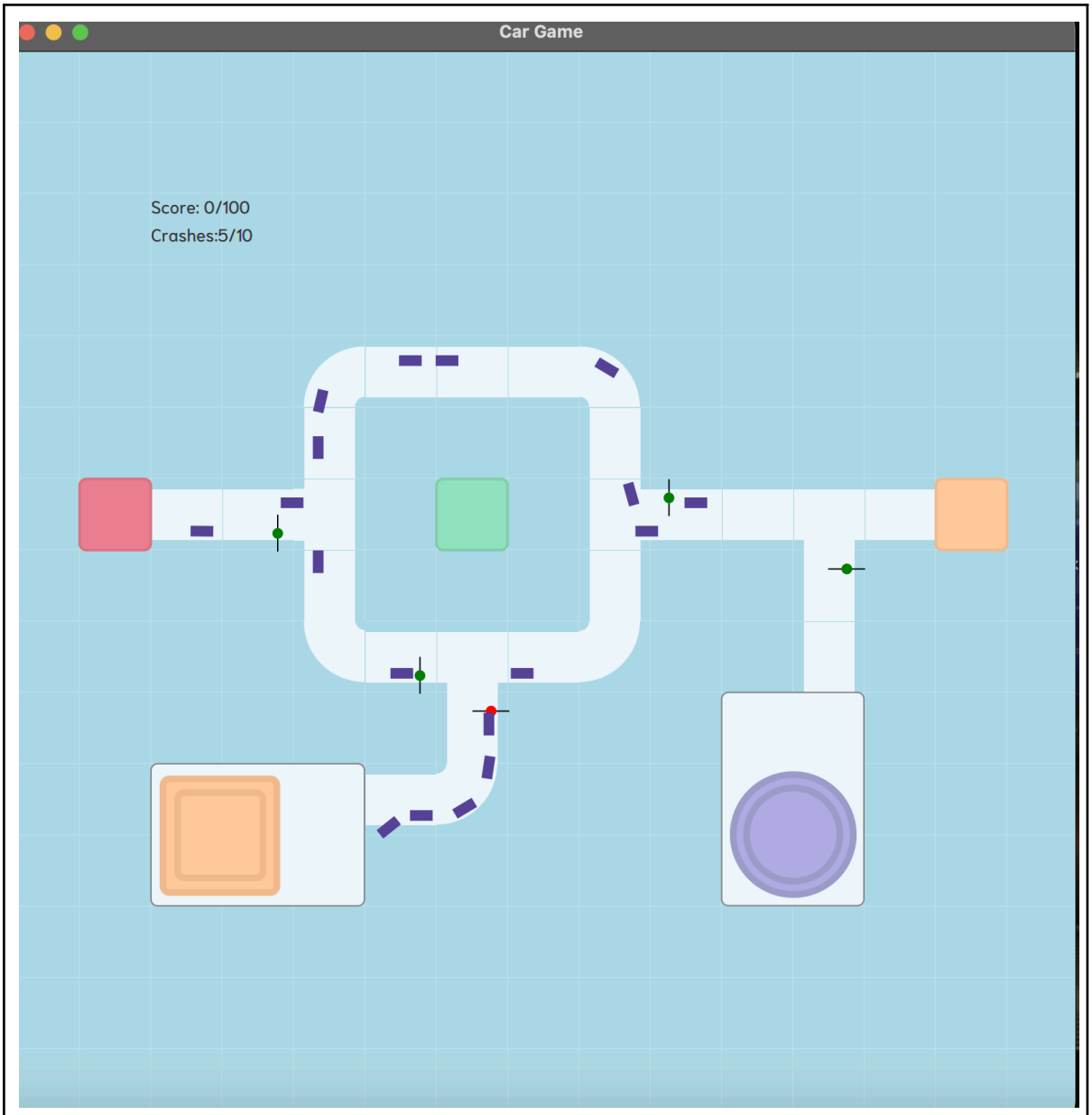

Image 4 - level 4 traffic

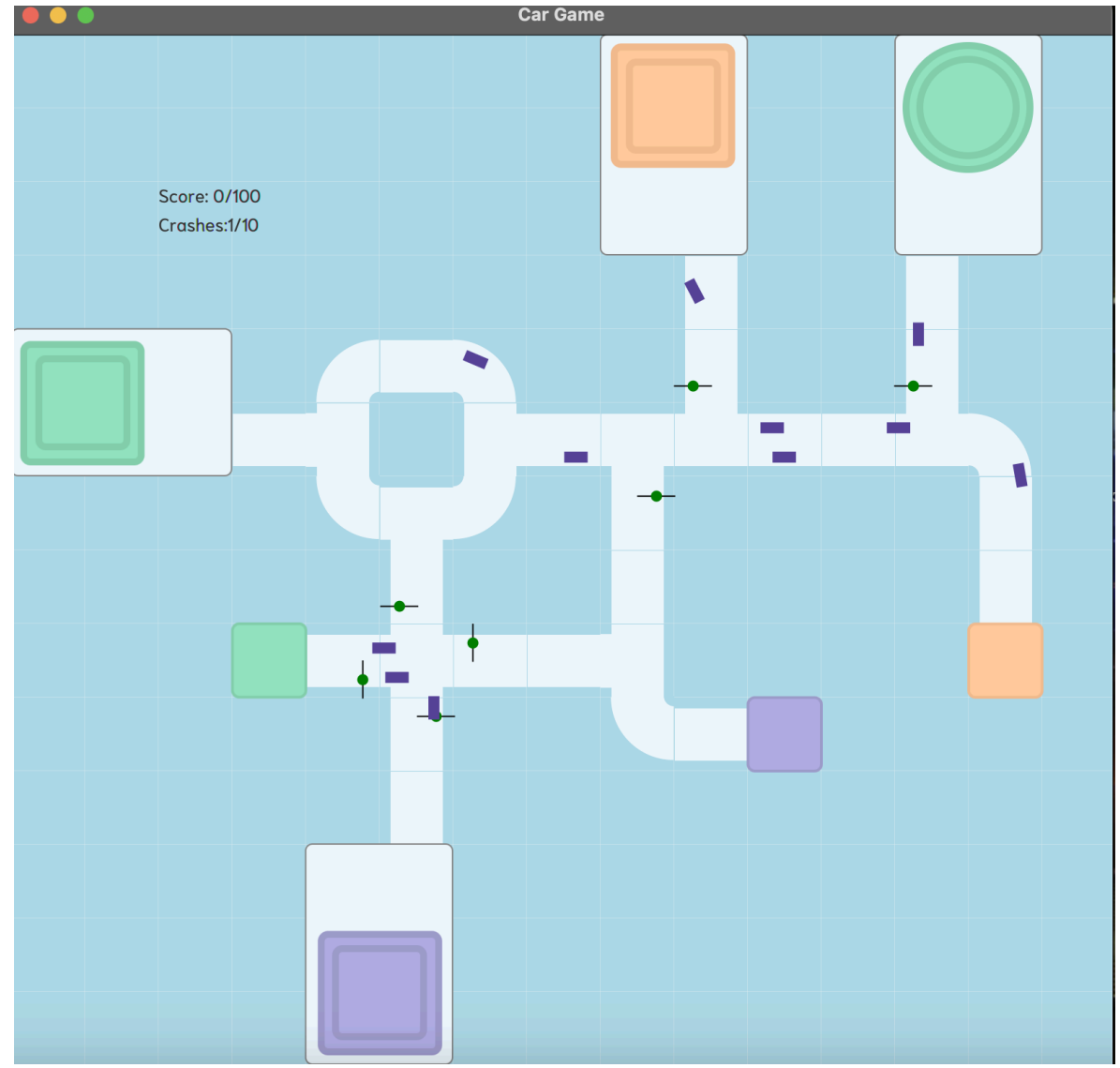

Image 5 - level 5 lights and traffic