**MARMARA UNIVERSITY COMPUTER SCIENCE ENGINEERING**

**TERM PROJECT : TRAFFIC GAME**

CSE1242 Computer Programming II, Spring 2024

Submitted to: Asst. Prof. Sanem Arslan Yılmaz

Due Date: 10.05.2024

|  |  |  |
| --- | --- | --- |
| 1 | 150122059 | Alper Kaan Arslan |

Table of Contents

[**Problem Definition** 3](#_Toc166261016)

[**Implementation Details** 3](#_Toc166261017)

[**Test Cases** 7](#_Toc166261018)

[**Beginning of The Game** 7](#_Toc166261019)

[**Traffic Control** 8](#_Toc166261020)

[**Level Maps** 9](#_Toc166261021)

Table of Figures

[Figure 1: Interface of the game 7](#_Toc166260906)

[Figure 2: Level Options inside Import Level Button 7](#_Toc166260907)

[Figure 3 Traffic in Level 4 8](#_Toc166260908)

[Figure 4 Level 1 Map 9](#_Toc166260909)

[Figure 5 Level 2 Map 9](#_Toc166260910)

[Figure 6 Level 3 Map 9](#_Toc166260911)

[Figure 7 Level 4 Map 9](#_Toc166260912)

[Figure 8 Level 5 Map 10](#_Toc166260913)

# **Problem Definition**

Traffic Game is a Traffic simulator based on the traffic system in everyday life, consists of buildings, cars, roads, and traffic lights. Cars emerge from the buildings they produced and attempt to reach another building by following specific routes. Each car increases the score when it successfully reaches It’s destination without colliding with another car and bringing the player closer to winning. The player is able to change the traffic lights' colors using the mouse to prevent car collisions and strive to win. Total collision and successful arrival counts are monitored within certain limits that may vary according to the level. If the player reaches the targeted total successful arrival count before the collision count that needs to be avoided, the level is considered successfully completed. There are total of 5 levels, and one cannot proceed to the next level without completing the current one. To win the game, all levels must be completed.

# **Implementation Details**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Building | |  |
|  |  |  | |
| +  -  - |  | Building(int type, double rotation, int color, int gridCellX, int gridCellY )  setStroke(color: int, c: Circle): void  setStroke( color: int, r: Rectangle): void | |

Building class **extends** Pane

|  |  |  |  |
| --- | --- | --- | --- |
|  | Car | |  |
| -  -  -  -  -  -  -  ~  ~  -  ~  -  +  - |  | car: Rectangle final CARSPEED: int cars: ArrayList<Rectangle> transition: PathTransition x: double double y; path: Path yCoordinate:double xCoordinate: double elements: List<PathElement> sceneCoords: Point2D stopped : boolean score: int alreadyFinished : boolean | |
| +  +  +  +  +  +  +  -  +  +  + |  | Car(path: Path)  stop(): void  move(): void  getX(): void  getY(): void  isTooCloseTo(otherCar : Car): boolean  isStopped(): boolean  calculateLength(path: Path): double  isCollidingWith(otherCar: Car)  isOnPath(otherCar: Car) : boolean  remove(): void | |

Car class **extends** Group

|  |  |  |  |
| --- | --- | --- | --- |
|  | CarSpawner | |  |
| +  -  -  -  -  -  -  -  -  -  -  - |  | cars: ArrayList<Car>  paths: ArrayList<Path>  trafficLights: ArrayList<TrafficLight>  time: double  crashCount: int  crashCountLabel: Label  scoreLabel: Label  lastCar1: Car  lastCar2 : Car  space: Region  timer: AnimationTimer  labelBox: HBox | |
| +  -  -  -  + |  | CarSpawner(ArrayList<Path> paths, ArrayList<TrafficLight> trafficLights)  update():void  handleCrash(car1: Car, car2: Car): void  spawnCar(paths: List<Path>): void  restartGame(): void | |

CarSpawner class **extends** Pane

|  |  |  |  |
| --- | --- | --- | --- |
|  | GameBackground | |  |
| +  + + + + + + |  | width: double  height: double  numberOfGridCellsX: int  numberOfGridCellsY: int  pathCount: int  succesfullyArrived: int  crashedCars: int | |
| + |  | GameBackground(double width, double height, int numberOfGridCellsX, int numberOfGridCellsY, int pathCount, int succesfullyArrived, int crashedCars) | |

GameBackground class **extends** Pane

|  |  |  |  |
| --- | --- | --- | --- |
|  | Level1 | |  |
|  |  |  | |
| +  - |  | Level1()  PathParts(parts: String[], path: Path): void | |

Level1 class **extends** Pane

|  |  |  |  |
| --- | --- | --- | --- |
|  | Level2 | |  |
|  |  |  | |
| +  - |  | Level2()  PathParts(parts: String[], path: Path): void | |

Level2 class **extends** Pane

|  |  |  |  |
| --- | --- | --- | --- |
|  | Level3 | |  |
|  |  |  | |
| +  - |  | Level3()  PathParts(parts: String[], path: Path): void | |

Level3 class **extends** Pane

|  |  |  |  |
| --- | --- | --- | --- |
|  | Level4 | |  |
|  |  |  | |
| +  - |  | Level4()  PathParts(parts: String[], path: Path): void | |

Level4 class **extends** Pane

|  |  |  |  |
| --- | --- | --- | --- |
|  | Level5 | |  |
|  |  |  | |
| +  - |  | Level5()  PathParts(parts: String[], path: Path): void | |

Level5 class **extends** Pane

|  |  |  |  |
| --- | --- | --- | --- |
|  | RoadTile | |  |
|  |  |  | |
| + |  | RoadTile(int type, double rotation, int gridCellX, int gridCellY) | |

RoadTile class **extends** Pane

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test | |  |
| -  -  -  -  -  - - |  | instance:Test  primaryStage: Stage  level1: Level1  level2: Level2  level3: Level3  level4: Level4  level5: Level5 | |
| +  -  +  +  + |  | start(primaryStage: Stage)  showLevelButtons(primaryStage: Stage, backgroundImageView: ImageView): void  loadLevelContent(selectedLevel: String): void  getInstance(): Test  main(args: String[]): void | |

Test class **extends** Application

|  |  |  |  |
| --- | --- | --- | --- |
|  | TryAgainScreen | |  |
| -  - |  | carSpawner: CarSpawner  stage: Stage | |
| +  + |  | TryAgainScreen(carSpawner: CarSpawner)  show(): void | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Win | |  |
| -  - - |  | stage: Stage  currentLevel: int  testInstance: Test | |
| +  +  -  - |  | Win(CarSpawner carSpawner)  show(): void  loadNextLevel(): void  showCongratsMessageAndExit(): void | |

In our project, we have a total of 14 classes.

We have 2 different classes, "Building" and "RoadTile", for creating buildings and roads. Using these classes, we position the objects we create onto a grid pane using another class named "GameBackground". Additionally, we have 3 additional classes, "Car", "CarSpawner", and "TrafficLight", for initiating the starting animation and producing the items used in the animation.

In addition to these, we have 5 classes representing 5 different levels: "Level1", "Level2", "Level3", "Level4", and "Level5". Each level class has separate text files from which we read values to determine various aspects such as the positions of buildings and traffic lights, the buildings where cars will be produced and reach, and the roads cars will follow in each level.

We have a "Test" class where we examine processes related to winning and losing conditions and create instances of the levels. Finally, we have "Win" and "TryAgainScreen" classes which provide outcomes based on winning and losing conditions.

In the "TryAgainScreen" class, when a level is lost, a "Try Again" button is displayed, allowing the player to retry the level. Each time a level is won, the player can progress to the next level using the "Win" class, receiving a congratulatory message at the end of the game before exiting.

Except for minor bugs, our project is complete. It is currently operational without encountering any errors.

During the implementation, we encountered several difficulties as follows:

When attempting to control the movement of cars using traffic lights in animations (stop/go), we faced issues activating the lights and couldn't achieve the cars' movement based on the traffic lights.

Another challenge was that when turning any traffic light red, instead of cars lining up one after another, they overlapped.

Lastly, we encountered a problem with cars crashing during corner turns.

However, eventually, we managed to solve nearly all of these problems.

In our project, an additional functionality implemented by our team is the “import level” button utilized on the starting screen of the TrafficGame. This button provides access to all levels of the game seperetaly on the screen, enabling players to start the game from any desired level.

Some bugs that we can't find a solution for:

1-In menu, player encounters 2 button named as “Import Level” and “Start Game”. If player chooses the “Import Level” button ; level options will be seen. After any level preference and initiation ,If player exceeds the level, with “Next Level” button clicked , player will start to play second level. In case of succes of that level as well, “Next Level” button will work as it should be.

2- When cars come from different path instances and meet at the same traffic light, they can become intertwined or unexpected accidents can occur. (Ex: 5th TrafficLight in Level 4.)

3- Even if the cars reach the buildings where they are produced due to the duration of red light they have encountered, car production does not stop in that building.

4- When cars start to move after waiting too long at traffic lights, some cars sometimes changed their position or teleported several units.

# **Test Cases**

## **Beginning of The Game**

Figure 1: Interface of the game Figure 2: Level Options inside Import Level Button

As it is seen , there are 2 options in the interface of the game. When the "Import Level" button is selected, the player is offered with 5 levels that can be played independently of each other. The player can choose the level they want and play. When the "Start Game" button is selected, the game automatically starts from the first level and progresses to the last level by pressing the "Next Level" button on the screen as long as levels are successfuly accomplished. If the level cannot be passed, the same level can be tried again by clicking the "Try Again" button on the screen.

## **Traffic Control**

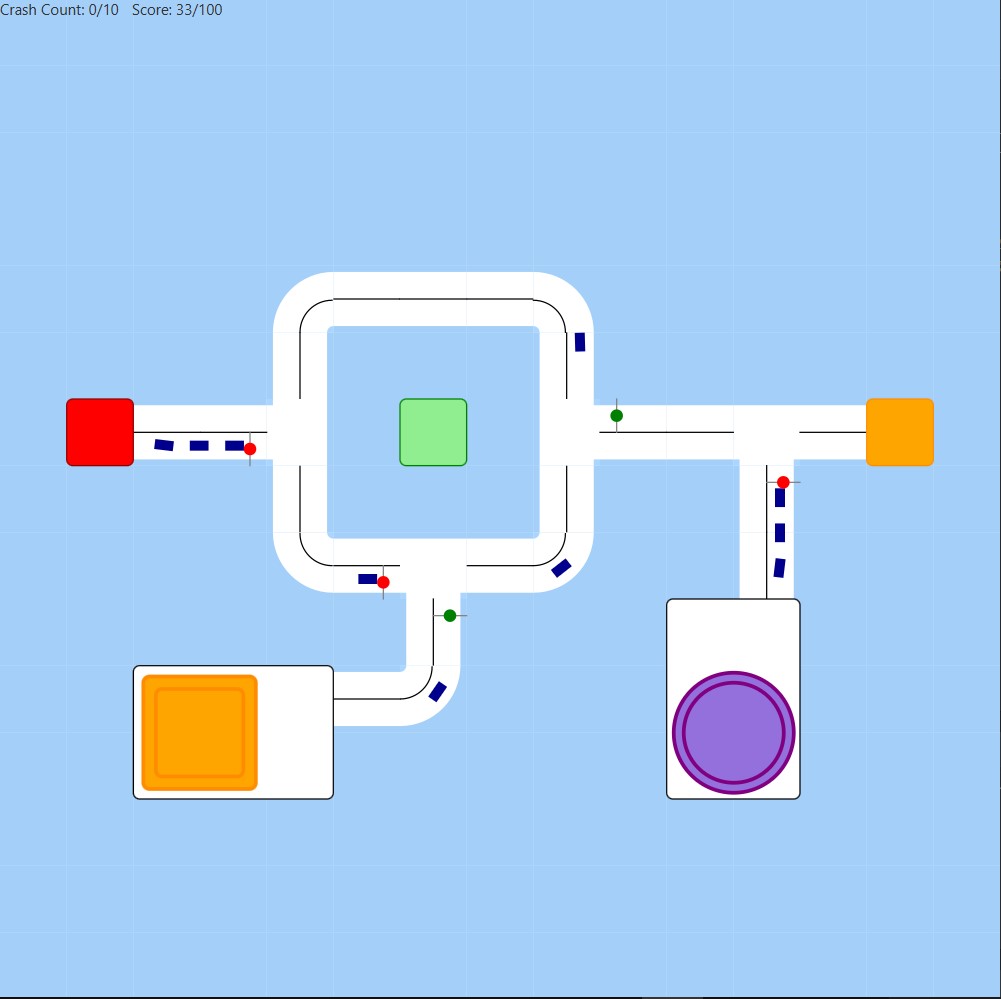


Figure 3 Traffic in Level 4

As seen above, the cars follow a certain path from the building where they are produced to their destination. If there is a red traffic light on their way, they stop in turn at a certain distance. When the cars successfully reach their destination, the "Score" value on the top-left increases. If two cars collide and an accident occurs, they disappear from the screen and the Crash Count at the top-left increases.

## **Level Maps**

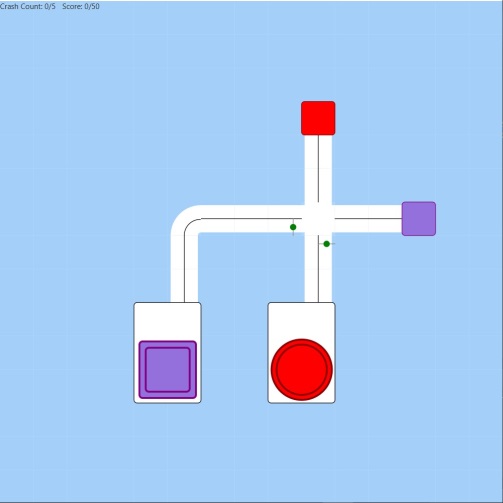
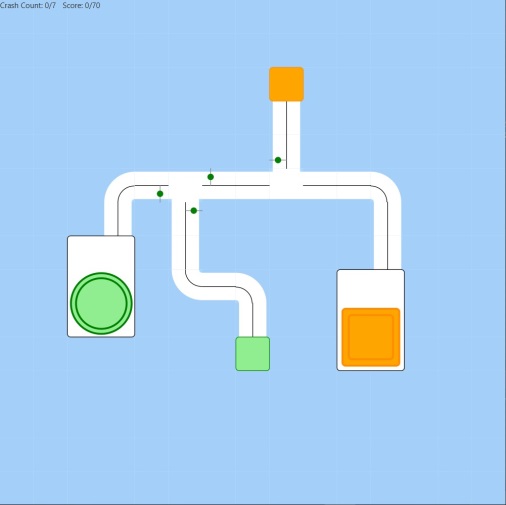
 

Figure 4 Level 1 Map  Figure 5 Level 2 Map

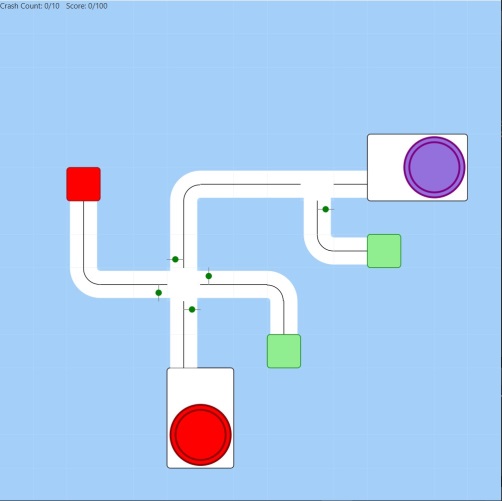
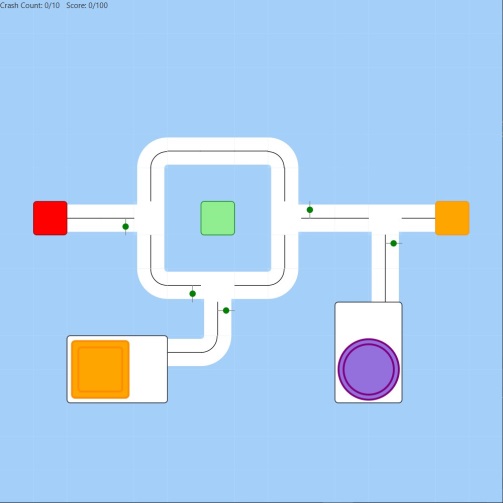
 

Figure 6 Level 3 Map Figure 7 Level 4 Map

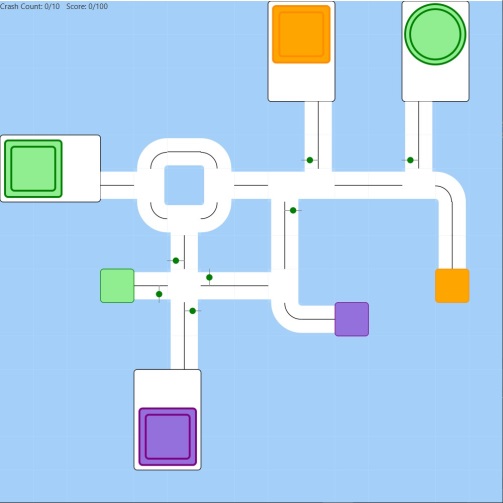


Figure 8 Level 5 Map

There are 5 levels in the game and each level has different maps.