

**CSE1242 COMPUTER PROGRAMMING-II**

**(SPRING 2024)**

**TERM PROJECT**

**TRAFFIC CONTROL SIMULATOR**

**09.05.2024**

|  |  |  |
| --- | --- | --- |
|  | Student ID Number | Name & Surname |
| 1 | 150123852 | İlayda İlhan |
| 2 | 150123081 | Sultan Kocagöz |
| 3 | 150123075 | Dilek İrem Çıldır |

TABLE OF CONTENTS

Problem Definition…….…...…………………………………………………..........3

Implementation Details …………………………………………….…………..........3

Test Cases…………………………………………………………………………….9.

**PROBLEM DEFINITION**

In the traffic light control simulator project, a road was created for vehicles to proceed and buildings where vehicles can exit and enter were constructed. After all these background edits were made, traffic lights were created with the given coordinates and then vehicles moving on the road were created. The movement of vehicles depends on the status of traffic lights. If the traffic light is green, the vehicles continue to move on the road, while the vehicles stop at the red light, creating traffic congestion. Control of traffic lights is provided by the user. When you click on the traffic light, it changes its current color to the opposite color.

On some roads where vehicles pass, vehicles may collide. To prevent this, traffic lights must be controlled by the user. While the vehicles completing the road are added to the score table, the number of vehicles colliding with each other is also added to the collision table created in the game. The game is lost when the number of collisions reaches 10. This game has 5 different levels, and when the code is run, you can start the game directly from the 1st level by pressing the start button in the window that opens, or you can choose the level you want according to the import level. The aim of the game is to control the traffic lights to prevent the vehicles from colliding. You can stop the game with the stop button created in the game, if desired. You can exit the game or continue the game accordingly.

**Implementation Details**

**UML Diagrams:**

|  |  |
| --- | --- |
| BUILDING | |
| -  -  -  -  - | type: int  degrees: int  colorIndex: int  cellX: int  cellY: int |
| +  +  +  +  +  +  +  +  +  + | Building(type: int, degrees: int, colorIndex: int, cellX: int, cellY: int)getType() : int  setType(type:int):void  getDegrees(getType(): int  getDegrees(): int  getColorIndex(): int  setColorIndex(colorIndex: int): void  getCellX(): int  setCellX(cellX: int): void  getCellY(): int  setCellY(cellY: int): void |

|  |  |
| --- | --- |
| METADATA | |
| -  -  -  -  -  -  - | width: double  height: double  gridCellsX: int  gridCellsY: int  numOfPaths: int  numOfCars: int  carAccident: int |
| +  +  +  +  +  +  +  +  +  +  +  +  +  +  + | Metadata(width: double, height: double, gridCellsX: int,gridCellsY: int, numOfPaths: int, numOfCars: int, carAccident: int)  getWidth(): double  setWidth(width: double): void  getHeight(): double  setHeight(height: double): void  getGridCellsX(): int  setGridCellsX(gridCellsX: int): void  getGridCellsY(): int  setGridCellsY(gridCellsY: int): void  getNumOfPaths(): int  setNumOfPaths(numOfPaths: int): void  getNumOfCars(): int  setNumOfCars(numOfCars: int): void  getCarAccident(): int  setCarAccident(carAccident: int): void |

|  |  |
| --- | --- |
| PATH | |
| -  -  -  - | indexOfPath: int  element: String  x: double  y: double |
| +  +  +  +  +  +  +  +  + | Path(indexOfPath: int, element: String, x: double,y: double)  getIndexOfPath(): int  setIndexOfPath(indexOfPath: int): void  getElement(): String  setElement(element: String): void  getX(): double  setX(x: double): void  getY(): double  setY(y: double): void |

|  |  |
| --- | --- |
| ROADTILE | |
| -  -  -  - | type: int  degrees: int  cellX: int  cellY: int |
| +  +  +  +  +  +  +  +  + | RoadTile(type: int,degrees: int,cellX: int,cellY: int)  getType(): int  setType(type: int): void  getDegrees(): int  setDegrees(degrees: int): void  getCellX(): int  setCellX(cellX: int): void  getCellY(): int  setCellY(cellY: int): void |

|  |  |
| --- | --- |
| TRAFFICLIGHT | |
| -  -  -  -  -  -  - | x1: double  y1: double  x2: double  y2: double  isRed: boolean  LightCheckArea: Rectangle  isGreen: boolean |
| +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  + | TrafficLight()  TrafficLight(x1: double, y1: double, x2: double,y2: double)  setGreen(isGreen: boolean): void  isGreen(): boolean  isRed(): boolean  setRed(isRed: boolean): void  getLightCheckArea(): Rectangle  setLightCheckArea(lightCheckArea: Rectangle): void  getX1(): double  setX1(x1: double): void  getY1(): double  setY1(y1: double): void  getX2(): double  setX2(x2: double): void  getY2(): double  setY2(y2: double): void |

|  |  |
| --- | --- |
| TEST | |
| -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  - | s: Stage  pane1: Pane  gp: Pane  image: ImageView  vbox: VBox  scenes: ArrayList<Scene>  FxPaths: ArrayList<Path>  time: double  timer: AnimationTimer  cars: ArrayList<Rectangle>  carTransitions: ArrayList<PathTransition>  pausedCars: ArrayList<PathTransition>  lights: ArrayList<Circle>  removedCars: ArrayList<Bounds>  crashes: int  score: int  nextLevel: File  next: Button  files: File[] |
| +  +  +  -  -  -  -  + | start(Stage): void  showScene(Scene): void  startGame(File): Scene  update(File, Pane, Label, Label, Button, Button): void  spawnCar(Pane, Label, Label, Button, Button): void  checkCrashes(Label): void  checkLights(): void  main(String[]): void |

First, building, metadata, path, roadtile and trafficlight classes are written together with getter and setter methods according to the instructions in the term project file.

First for Test Class, a stage was added in the test class. Since this stage belongs to all of them, access to the stage is provided from anywhere. Then, Pane1 was created and access to this pane1 is provided from anywhere, just like the stage. Pane 1 gives the illustrated screen with the traffic lights that are first seen when the game is started. The created GP gives the game screen.

It includes returns such as Vbox back, so it should be accessible from everywhere. Vbox belongs to the main screen. When we return to the main screen, we first delete the pane and then reach Vbox, and there is access to Vbox at all levels.

We added all the scenes we created in the scene screen into this scene array. Fxpaths contain paths for levels and these paths are updated for each level. Then, the necessary input files were read using file reading operations.

**Start method:**

We created the file array as final, the reason why it is final is because we can only change the final ones in the set on action. We added images and buttons for the main screen and these buttons were taken to the Vbox defined at the beginning. We took all of them to the main pane (pane1) and added them to the scene.

The functions of Button1 and Button2 have been written. When Button1 is pressed, the user is presented with two buttons: back and start buttons. Clicking the back button returns to the main screen, while selecting the start button allows the game to start. Back and start buttons were added to Vbox because if the game is won, the back and next level buttons will appear before the user. When the next level button is clicked, the next level runs from the current level and deletes the arrays belonging to the current level and ensures that the file belongs to the next level defined at the beginning. Button2, when the button named import level is pressed, the data in the start panel is deleted and the previous image and box are added. There are buttons for the levels in this box. It takes one of the file arrays belonging to the initial file level defined at the beginning of the code.

**Startgame method:**

It takes the file as a parameter and the next file from the file array determined at the beginning is thrown to the next level. Then the file reading and file reading closing operations are performed. Pane2 is created. In order, it is required for metadata, roadtile, building, trafficlight.

The scores and crashes of the game are started and the score and crashes labels that will appear at the beginning of the game are defined. A stop button has been designed to allow exiting the game while in the game. When this stop button is clicked, 'Do you want to continue?' appears on the screen. ' Question and two buttons appear below the question. These are yes and no expressions. If the user selects the yes button, the game continues to run from where it was paused. If the user selects the no button, the game returns to the main screen. Go label was created. Music was added to play in the background when the game is run.

Time is set equal to 3 because the cars exit immediately at startup. The field created in the data field is defined. The Hande method is overridden, and the update method is called. In the update, checkcrashes and check light methods are called.

**Check Crashes:**

The positions of the two cars are taken and the distance between them is determined. If the distance between the two cars is less than a tolerable distance and they do not belong to the same path, then the crashes of these cars are increased by one and the crashes label is updated in the upper left corner of the game screen. The crashed cars are displayed both from the panel and from the path transition. is removed. In this process, first the path transition stops and then the car is removed from the panel. To not be counted in the next control, car1 and car2 are removed from the transition.

**Checklight**:

Checklights contain the locations of cars and traffic lights. If the distance between cars and traffic lights is smaller than the tolerable distance and the color of the light is red, the path transition for that car is paused. If it is smaller than the tolerable distance but green, then play occurs and then the positions of the cars relative to each other are checked. In the next cycle, if the distance between the cars is smaller than the tolerable distance and the previous car is paused, then the car behind is also paused. If the car in front is moving forward, the car behind is also played.

**Update method:**

In this method, cars appear randomly. If the score is greater than or equal to 100, the path transition of all cars on the screen is stopped and all cars are deleted from the screen and the music is also stopped. When the user wins the game, a new Vbox is added, labels are added and if the last level is not selected, a 'next level' button is added and a back button is added.All of these are taken to the pane. If the score is less than 100 and the crash is equal to or greater than 10, then all path transitions and music are stopped. Again, a new pane is created, and the leaves and back button are created in the pane. If the score is less than 100 and crashes If da is less than 10, the game continues, and the Spawn method is called.

**SpawnCar**

Cars are created and the colors of the cars are assigned randomly and the path for that level is determined randomly. The path transition is created. If the path transition ends, the score is increased. The path transition is removed from the car transition and the score label is updated. The cars that have completed the road are removed from the panel, and at the same time, the cars that have been created are also removed from the cars. The timer is started.

**Questions:**

**Which parts are complete/incomplete in your project?**

Our project works completely according to the term project file instructions given by our academician. Additionally, extra things have been added to the project by our team.

**What are the difficulties you have encountered during the implementation?**

We first created the project using gridpane, and that's why we spent a lot of time and gridpane was causing the coordinates of many things to mismatch. We created the project much more easily by using pane again, but during this time, we spent a lot of effort and reached the wrong result. Due to the gridpane we used in the beginning, the traffic lights were blurred. We had difficulty in making the coordinates and the path of the cars. It turns out that our mistake was due to not using panels.

We also had some difficulty in the beginning in creating the rotation logic of large-sized buildings. Despite all these difficulties and weeks of effort, we were very happy when we achieved the result and completed the project without any problems.

**What are the additional functionalities of your project added by your team?**

Some buttons have been added to the project by our team to make the game easier to use. Thanks to the stop button, the game is stopped, and a scene opens. In this scene, the user is asked whether he wants to continue the game. If the user clicks the yes button, the game continues to run from where he left off. If the user selects the no button, he exits the game and goes to the main screen, and from there the user can select the desired level and start the game again with the desired level. In addition, when the game is started by our team, a game music plays in the background, which provides an effective gaming environment to the game users.

**TEST CASES**

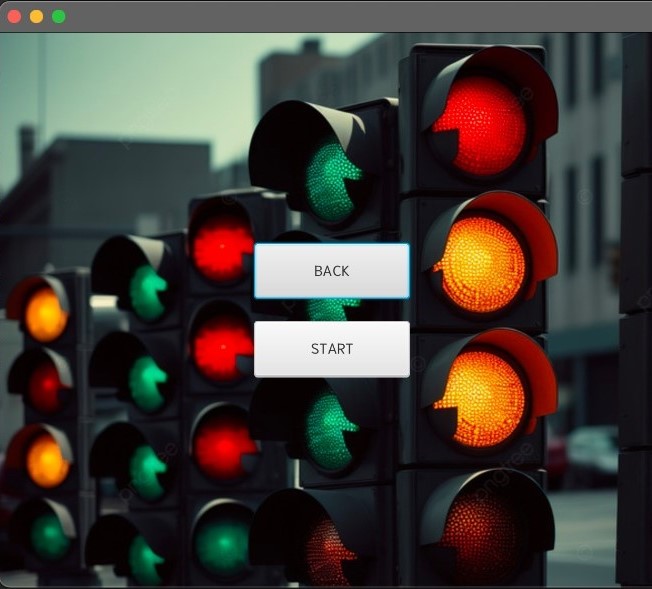
trafik ışığı, hafif, taşımak, nakletmek, sinyal verme cihazı içeren bir resim

Açıklama otomatik olarak oluşturuldu

First, when the code is run, the user is greeted by two buttons with start game and import level options. When start game is selected from these buttons, the game starts directly from the 1st level, and when import level is selected, buttons with 5 different levels welcome the user to select.

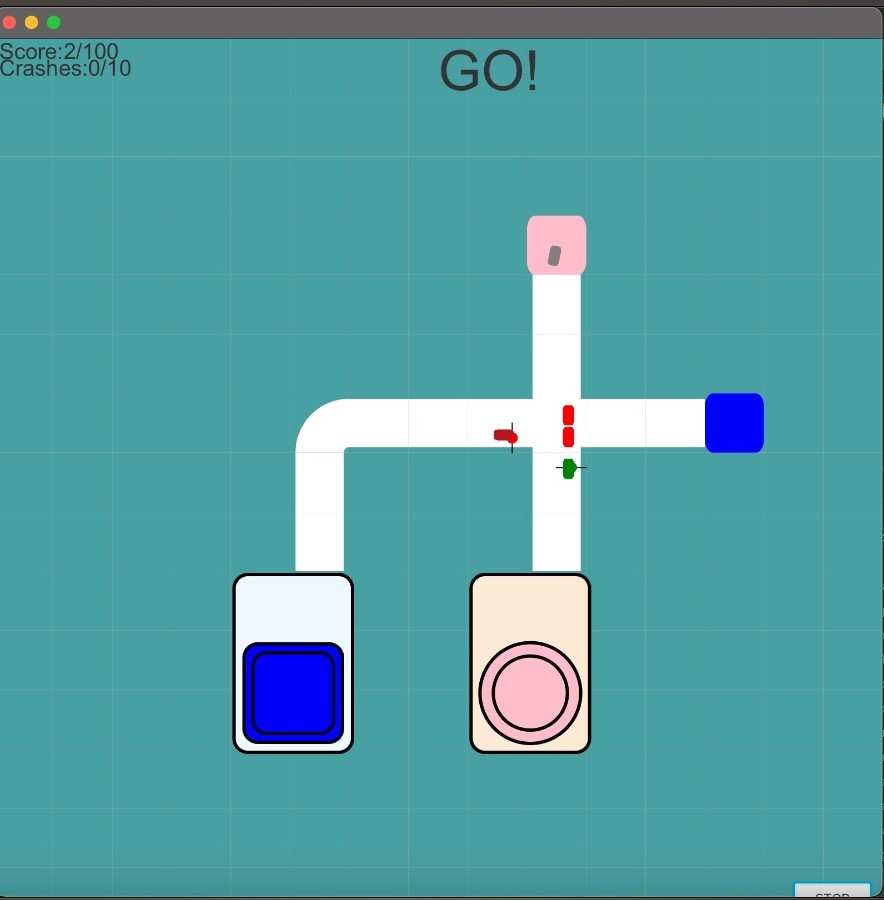


After the user selects the import level, this screen with the buttons for the 5 levels welcomes the user. These buttons allow the user to choose the level they want to start with.

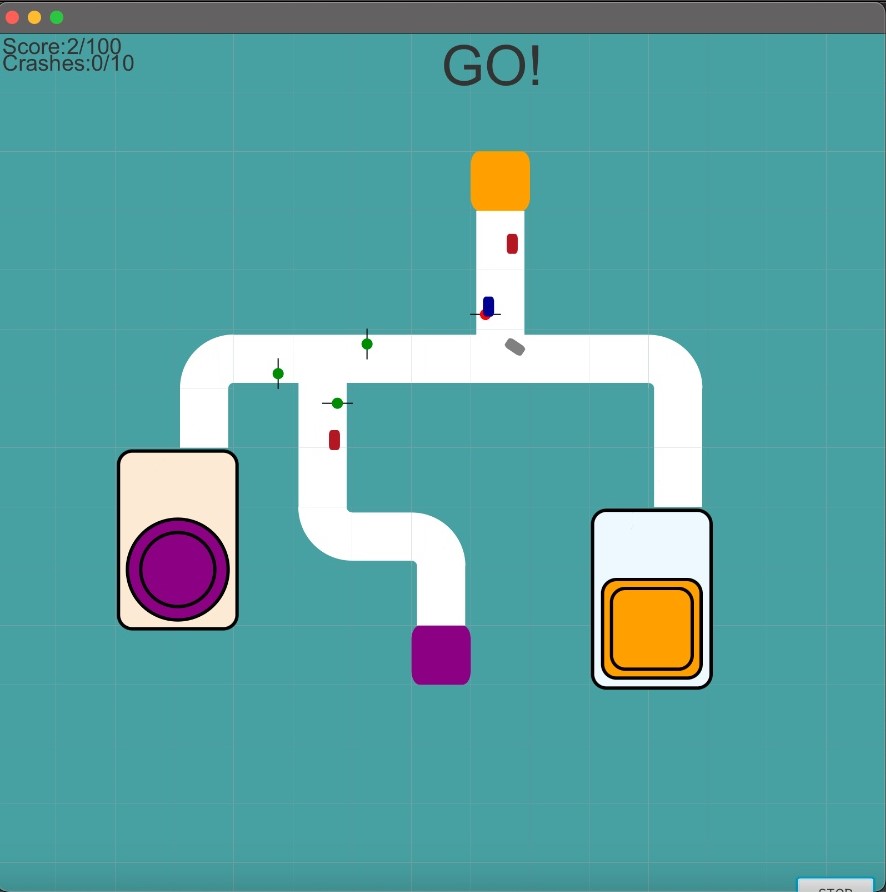


After selecting the level, the user wants to play, if the user wants to make changes again, the user can return to the main screen by pressing the back button and select the level again, or can run the level selected by the user by pressing the start button.

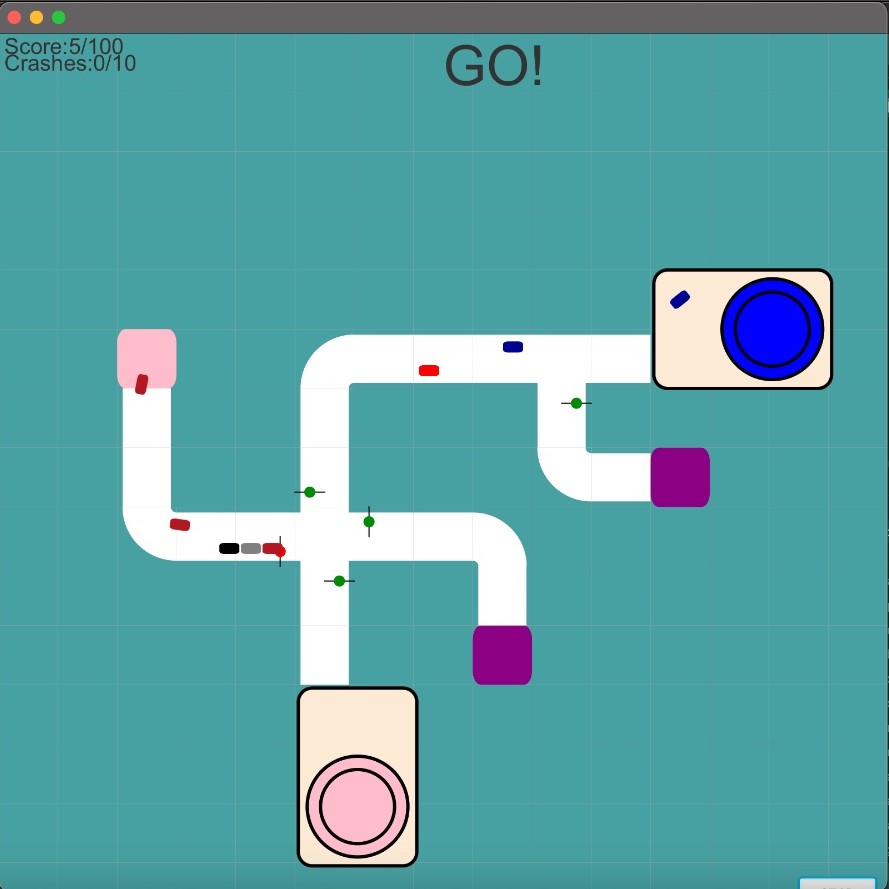
Depending on the level chosen by the user, after selecting stat game, the level of the game opens and there are 5 different levels to choose from in the game.



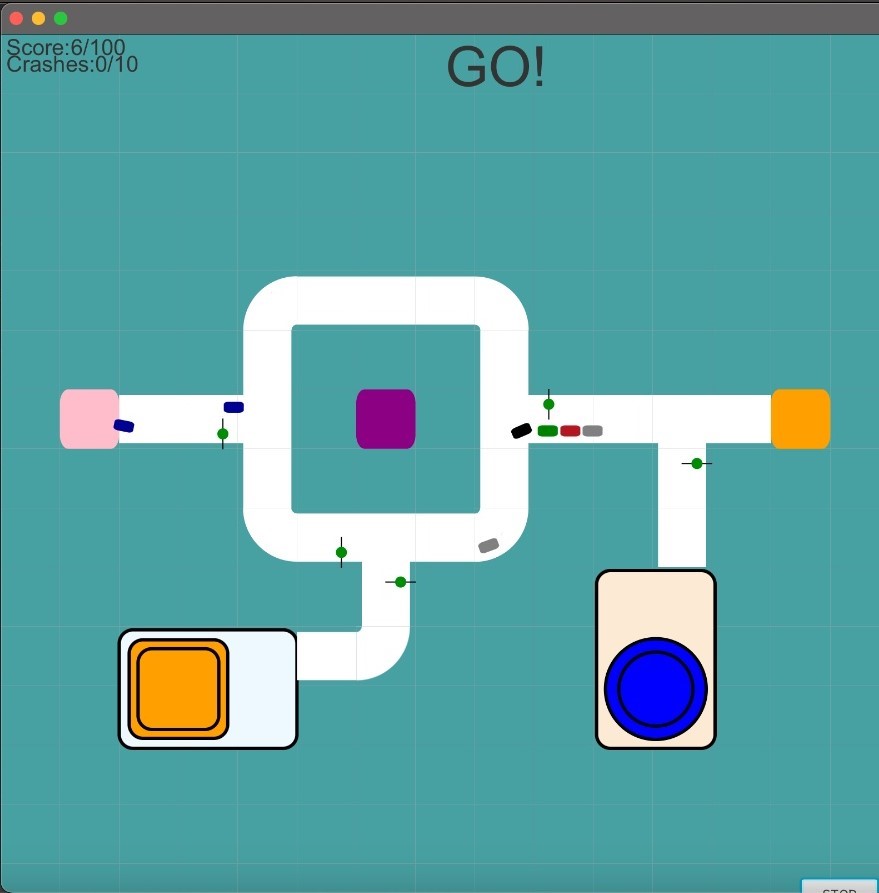
Level-1



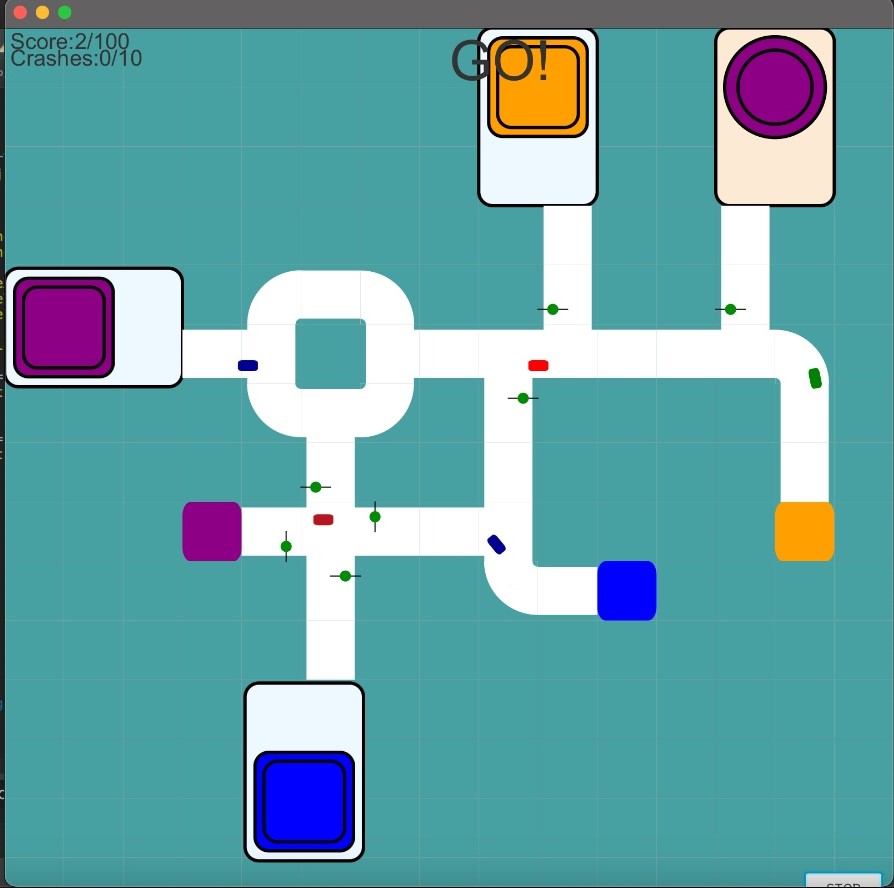
Level-2



Level-3



Level-4

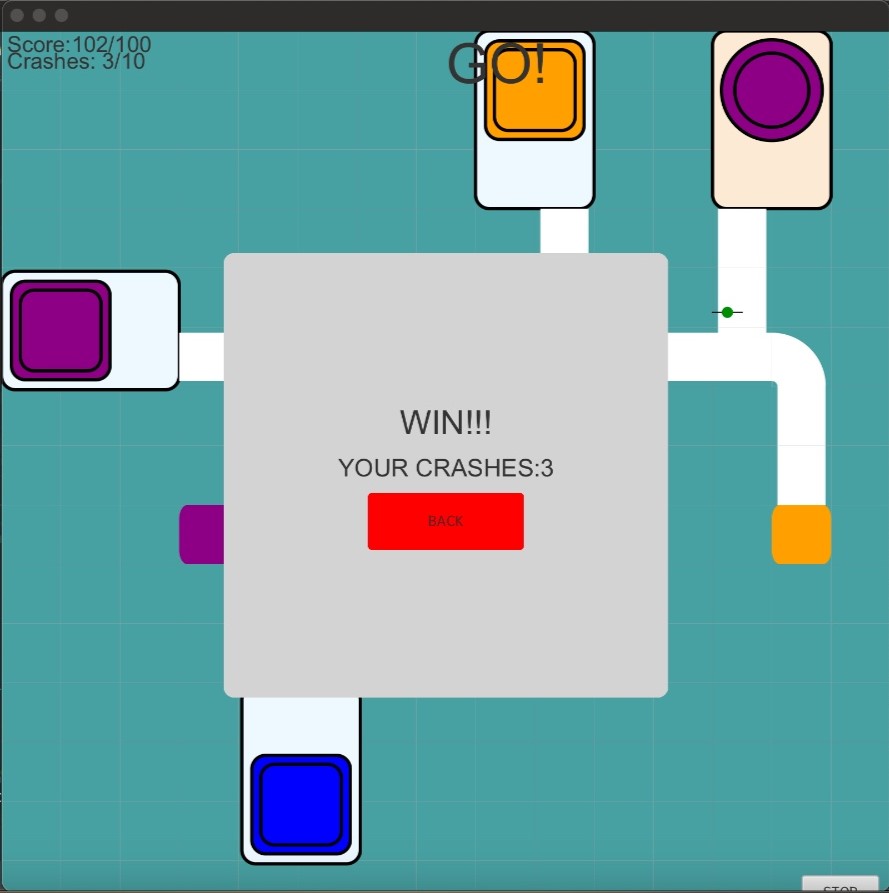


Level-5

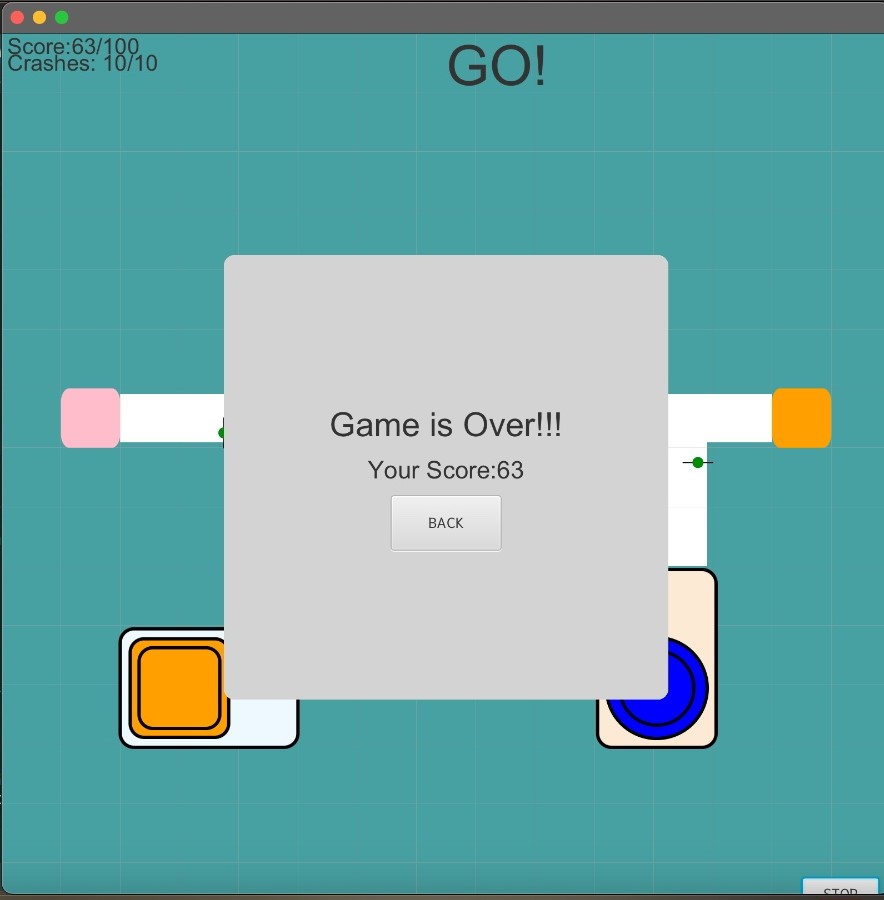
If the user presses the stop button in the lower right corner of the levels, the user will be greeted with a screen like this.



When the user presses the stop button, the game is stopped, and a screen appears. On this screen, it asks the user if they want to continue the game and has buttons named yes and no. If the user selects the yes button, the game will continue running from where it was paused, but if the user selects the no button, the game will be exited.



If the number of vehicles completing the path reaches 100 and the number of collisions is less than 10, the user wins the game.



If the number of colliding cars reaches 10 before the number of vehicles completing the path reaches 100, the game is lost.