Interactive Graphics Final Project: RoboRun

Hamna Moieez, 1968733

September 23, 2021

1 Introduction

For the purpose of final project for the Interactive Graphics course, I chose to design a basic endless runner game. The design of the game is considerably simple since the goal is to construct a character and animate it to move on a terrain, avoid certain obstacles and collect a reward on the way. The game is endless and greater the number of rewards collected, the greater is the overall score. On each obstacle interaction, a life line is taken away from the character. There are a total of three life lines and once they are exhausted, the game ends with the score being equal to the number of rewards collected.

This project report is divided into various sections. In the next section 2, we discuss the design of each of the elements of the game and also the overall game design, the color combinations, etc. This is followed by the discussion of the technical construction of the game in Section 3, and discussion on game play in the same section. Eventually, we discuss the advantages and disadvantages of the approaches in Section 4 and conclude it in Section 5.

2 Game Design

The first and foremost consideration in game design is to come up with a basic idea of all the elements in the game, their interactions with each other, the user interaction with the game and incorporating the sense of a scene that all these elements reside in. I have named the game, RoboRun, since it is an endless runner game where a robot runs on the streets avoiding obstacles and tries to collect as many coins as it can.

2.1 Elements in the Game

There are three main elements in the game: the character, the punishment, and the reward. The character is a robot which is running on a road. The character design is explained in the construction section 3, however the character is shown below, see Figure 1.

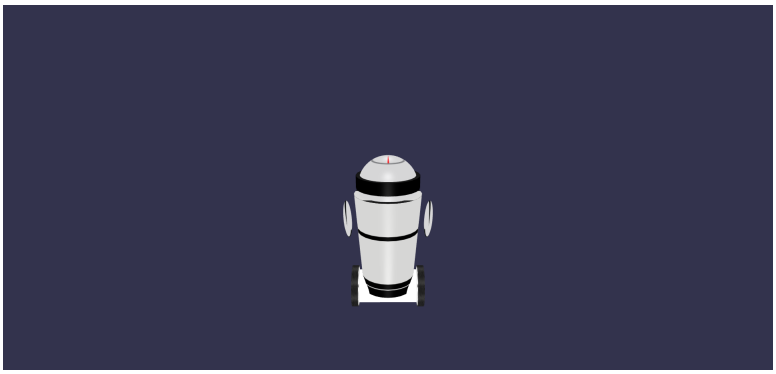


Figure 1: The robot character of the game RoboRun.

Figure 2: The obstacle which is a stop sign. Similar obstacles are placed along the way to block the character's way.

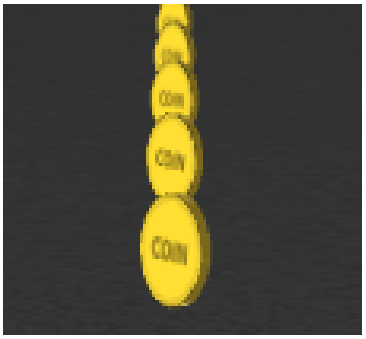


Figure 3: The reward as coins. This snippet is taken from the start of the game so they appear in a line since the game gets progressively difficult with random coin placement.

The next element is the punishment which I encoded as an obstacle. The robot (character) has three life lines and everytime the character hits an obstacle, see Figure 2, one life line is reduced. Once all the life lines are exhausted, the game ends. The reward is a coin which pops up at random places on the terrain as the robot runs and tackles the obstacles placed along the way, see Figure 3. Each coin is worth +1 and the score is added as more coins are collected as the robot runs through the game.

2.2 Inter-Element Interactions

Since I have explained the elements in the previous section, now we move onto defining the interactions. The character interacts with the environment in two possible ways: i) interaction with the obstacles ii) interaction with the coin, see Figure 4. In the first case (i), on each interaction of the character with the obstacle, a life line is exhausted. In the second case (ii), on each interaction of character with the coin, the coin is consumed and a score is earned.

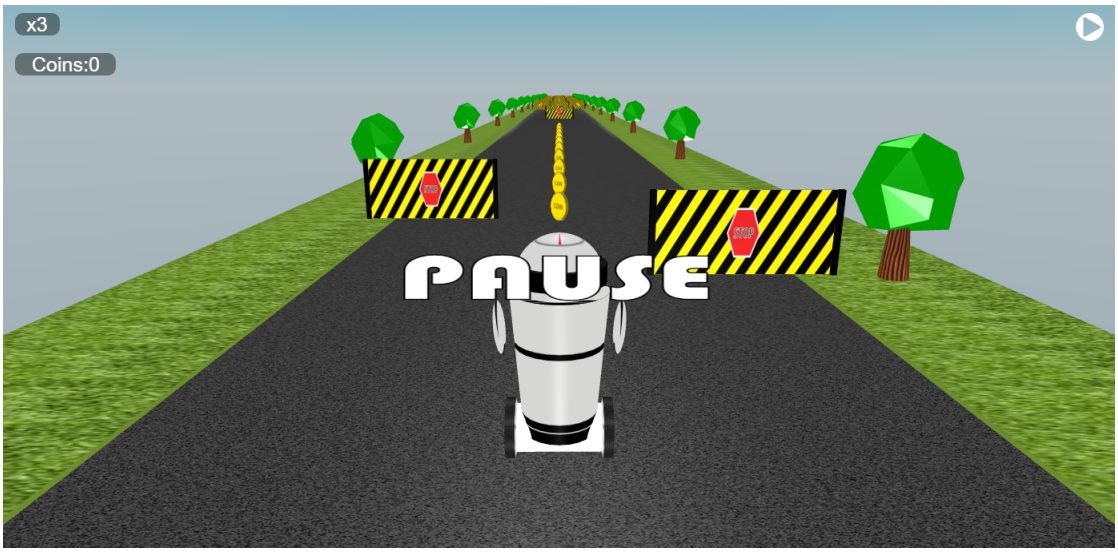


Figure 4: Snippet taken from the running game, on pause for the purpose of the taking a screenshot. Note that this is the start of the game. The robot is running on the terrain and the obstacles and coins are placed randomly on the terrain for the robot to avoid. The game has a progressive difficulty and therefore the obstacle placement is easier at the start and overall game play speed is slow. Once time progresses and user is still playing the game, the obstacle placement becomes faster, coins sparser and game play faster as well.

2.3 User Interaction

The other aspect of the design is the user interaction. The user interaction is the way the user interacts with the robot since essentially the user is controlling the robot and user has to avoid the obstacles and collect coins along the way. There are three main user actions: up, right, left. The up action corresponds to jump, the right to moving right and left to moving left, see Figure 5. These actions allow the robot to navigate the obstacles and collect coins.

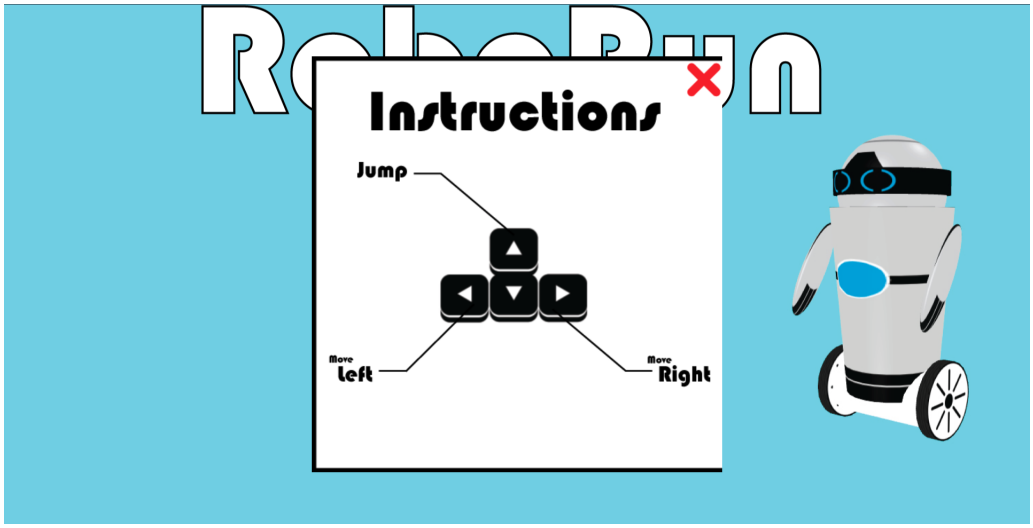


Figure 5: The instruction manual describing the user interactions. This is also accessible through the first page once the game has started.

2.4 Scene

The last component of design is the scene. For the scene aspect, I went with a simple endless road which has barricades placed as obstacles. There are certain trees on the side. The scene with the robot, sidewalk (with grass and trees on it), the road on which the robot runs, and sky as background is shown in Figure 6.

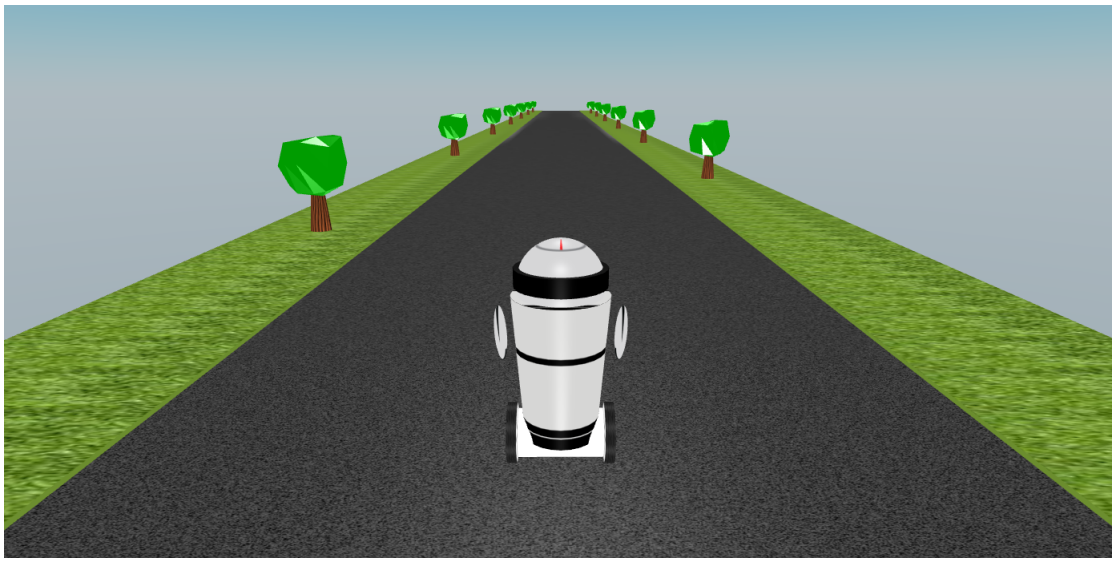


Figure 6: The overall scene of the game. Note that the obstacles are not placed yet.

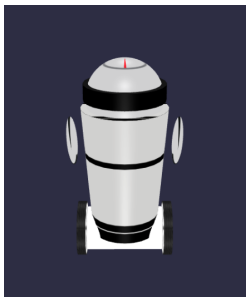


Figure 7: Textureless robot.

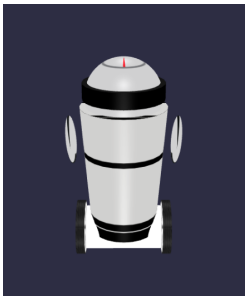
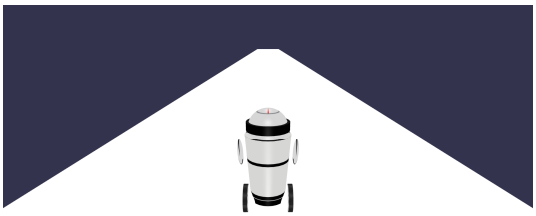


Figure 8: The textured robot with metallic texture applied.

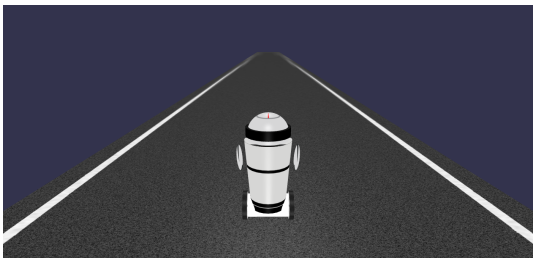
3 Game Construction

In this section we discuss the construction of the game. We have discussed the construction of each of the elements of the game. The main character is a robot which runs on a road and navigates the obstacles and has to consume coins to increase score. The robot is a hierarchical model following the instructions given in HW # 02. The game has been built in WebGL using Babylon.JS as the main library. The character, scene, and all the interactions are implemented in Babylon.JS. The base of the robot was built textureless and then the texture was added. The texture of the robot gives a metallic feel and has light added to it to expose certain parts. The camera is added to follow the robot from backside since it is running away from the screen into the road. The textureless and textured robot are shown in the Figure 7 and 8, respectively. Note that the textureless robot refers to the character without the current texture (metallic body) i.e. there was still a base checkered texture applied in the start.

The second aspect is the platform (a base) on which the robot runs. I constructed this following HW # 02, however swapped the grassy-patch with unending road with a grassy sidewalk. Furthermore, I also added some trees (taken from Babylon.JS) to the scene to add a bit more flavor to it. The construction process was pretty similar wherein I first design a base platform and then added a road texture and repeated the process for the sidewalk, see Figure 9.



(a) Textureless base ground



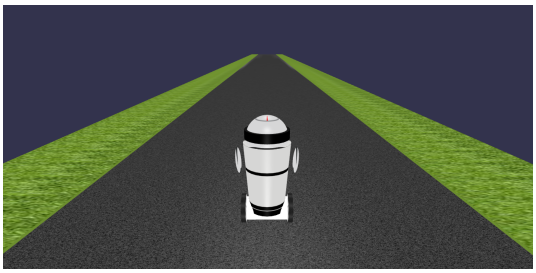
(b) Road texture added to ground

Figure 9: Textureless and Textured ground with the character, robot, placed on it. Note that in the second image, the sidewalk is still texture less.

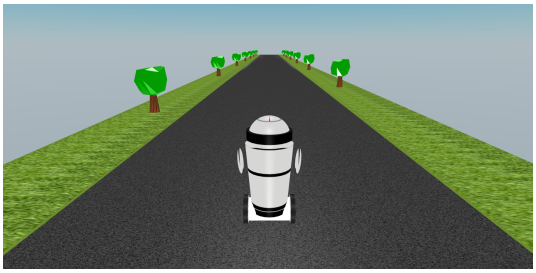
For the sidewalk, I have just added some grassy texture and then went on to add some trees, see Figure 10.

3.1 Animation

Once all of these objects were constructed, I went on to implement the animations. The animations are swivelling coins, robot moving forward, robot moving left and right and jumping. The left and right motion is basic translation along the X axis in addition to rotating the head of the robot in the direction of the motion to mimick the *looking towards that direction*. The code snippet for the motion towards left is given below. The one for right is similar with all the signs reversed and therefore the snippet is not given here.



(a) Grassy texture to sidewalk.



(b) Added trees to sidewalk.

Figure 10: The sidewalk with texture added and then with trees and some background (sky).

```
function robotMoveLeft() {
  if (robotHead.position.x == 0 || robotHead.position.x == 3) {
    moveRobotAnim(
      robotHead,
      robotHead.position.x,
      robotHead.position.x - 3
    );
    robotHead.addRotation(0, -0.3, 0);
    setTimeout(function () {
      robotHead.addRotation(0, 0.3, 0);
    }, 250);
  }
}
```

For the jump animation, it is translation along both X and Y axis with a bit of floating animation to add some sense of realistic animation. However, none of the animations are imported and are implemented with basic operations.

```
function robotJump() {
  if (
    robotHead.position.x == 0 ||
    robotHead.position.x == -3 ||
    robotHead.position.x == 3
  ) {
    aniPosY(robotHead, robotHead.position.y);

    running = false;
    setTimeout(function () {
      if (gameRunning) {
        running = true;
      }
    }, 680);
  }
}
```

All of these are triggered on event listeners on the user input (via the keys). These are the base animations and rest of the functions are helper functions i.e. re-rendering the model after translating, rotating, etc. Therefore, that code is not given. However, the entire code base is uploaded on Github and can be viewed at this link: CODE. Similarly, the game can be played at this link: PLAY.

4 Advantages & Disadvantages

The major advantage of the design is that since it is hierarchical, the entire construction is highly modular. The animation sequence is also modular and calling each of the movement functions triggers specific motion. One such instance could be adding more obstacles in the path of the robot with a diverse profile instead of just barricades. Doing this would be simple since we could re-purpose the previously used placing functions and by importing a model, it should be a few more lines of code. Similar is the case with animation, which can be extended by re-purposing previously written code with minor modifications. The disadvantage is mostly based out of WebGL's limitation that it does not allow for configuring complex scenarios and adding elements could lead to cluttering. However, overall the gameplay is smooth and can easily run in a basic browser. I have tested it on Chrome and it works just as expected.

5 Conclusion

In this project, I built a basic game as a requirement for the final project in WebGL using Babylon.JS. The game is a basic endless runner game where a robot runs on a road and tries to collect as many coins as it can while avoiding crashing into the obstacles. The game is designed following the specified requirements, with specific textures, lights and a hierarchical model of the main character, a robot in this case. The game is capable of running in any basic browser, we have tested it on Chrome and it works fine. The game can be played by clicking here: PLAY. The entire code is documented and is uploaded to the Github repository, link provided above.