TEAM Run Boy Run: Report

Our Idea

Our idea for the game was to create an endless running game inspired by Temple Run. The playing field is surrounded lengthwise by a forest scene that is purely aesthetic. The player moves continuously in the +Z axis at a pre-set speed while avoiding obstacles by moving around them or jumping over them. Like Temple Run, our game also has a chaser following the player closely. The player is a black rectangle and the chaser is a rotating boulder. The objective of the game is to continue moving forward without colliding with the chaser or any other obstacles on the path. There are also coins on the path for the player to collect as they move forward. Finally, as one would expect, the game ends when the player collides with any obstacle.

How to Play the Game

As described above the player has 3 abilities - the ability to move left, right and to jump. These actions can be achieved by pressing the <a> <d> and <space> keys, respectively. The player's job is to use a combination of these movements to avoid running into an obstacle and to collect coins as they move forward in the game.

Fig. 1. View of the game when it begins, showing the player, chaser, some coins and some obstacles.

If at any time, the player wants to restart the game, they can use the <1> key to reload the game. If the player collides with an object (that is not a coin) from the environment, the player comes to a halt which causes the game to end. At the same time, an alert message appears on the screen with a description of the amount of time the player spent in the game, as well as their

total score (#coins). The player can select ‘OK’ to restart the game if they wish to do so. There is also an optional rainforest ambience music than can be turned on/off using the music bar above the scene.

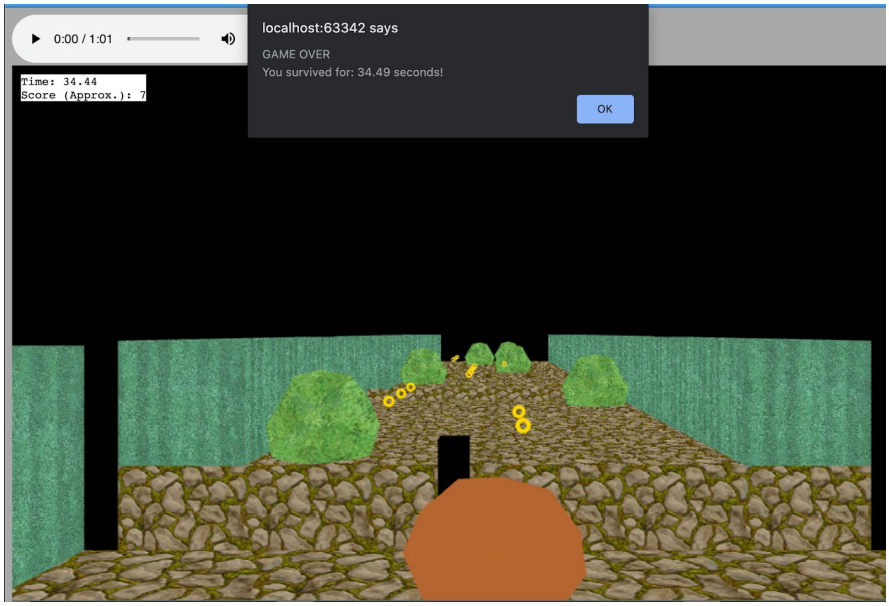

Fig. 2. View of the game when the player “collides” with the pit and causes the game to end.

Video of Gameplay

https://drive.google.com/file/d/1CgsxyqLhWsPUgnpmVx01kNIvaAqDZ37F/view?usp=sharing

Features and Implementation

Path and Obstacles

The path was created by creating a template path where a cube was translated and drawn in the x and y coordinates. The template path was then translated in the z direction to create an almost indefinite pathway. The game originally draws a path 100 units in the z direction on startup and then constantly gets drawn as the player progresses along the pathway. Therefore, the path is constantly being drawn and is hardly noticeable to the player. The pits were created by utilizing a modulus operation where at certain intervals the path would not be drawn and instead be translated an extra 2 units in the z direction before being drawn again. The shrubs were made in a similar fashion using the modulus operator on the coordinates of the path. Instead of translating forward and breaking the path, we translate upwards and then call the draw function on the “sphere” shape. We created a flat shaded sphere with only 2 subdivision spheres and added a grass-like texture to make it look more like a shrub.

Coin System

A coin system that would allow the player to increase their score for every coin picked up. The coin system represents a player's performance throughout the game, the more coins picked up equates to a higher score. The coins were made through manipulation of a torus that was translated, scaled, and then rotated along its own axis. To pick up the coins, the collision detection feature was utilized where it allowed the player to collide with coins and increment the player's score. The coins were placed by using a modulus operation similar to the pits and were translated among different locations along the path.

Player/Chaser/Camera Movements

We also had 3 separate model transformations for the chaser, player, and camera. Each of these had to be modified as the player moves in the game based on inputs from the user. The camera matrix is attached to the chaser and therefore moves with the chaser as it follows the player.

Text Overlay

To keep track of the elapsed time as well as the number of coins collected by the player, we added some text overlay to our scene, displaying the scoreboard. This was achieved using Javascript and CSS.

Collision Detection

Our advanced feature for this project was collision detection. Collision detection was essential for our game. This was used to determine if the player had hit any obstacles or the path boundary, fallen into a pit, or picked up a coin. To implement this, a function "collision_detected(x_coord, y_coord, z_coord, obstacle_x, obstacle_y, obstacle_z, obstacle_width)" was implemented and called before drawing any obstacle, pit, or coin. When calling this function, the coordinates of two objects and the object's width were passed in. This function returned true if there was a collision and false if the two objects did not intersect. We used bounding volume collision detection, in which an imaginary rectangle surrounded the center of each object with a width we specified while calling "collision_detected(...)". The function checked if each coordinate value of the first object was within a certain width of the second object. If the x and z coordinate were within range and the player was not in the middle of a jump, the function returned true.

Problems Faced

One large problem we faced throughout this project was mitigating lag due to path generation. Due to the iterative nature of our path generation and large size of the path

discussed above, each frame in the scene was computationally expensive. We mitigated some of the computation by only rendering the path up to 100 blocks in front of the user. This allowed the game to run smoothly for the first 60-100 seconds, as it never rendered the full path available to the user. However, our path generation began at the origin for each frame, so as the player moved further and further along the path, away from the origin, our program had to compute more and more blocks of the path, causing the game to slow down as time increased. In the future, we would like to stop rendering the parts of the path that the player had passed and are off screen. For now, we did not do that as each part of the path relied on the transforms we had performed previously. We could solve this by having a function that generates the needed model_transform based on the values of i and j in the for-loop, rather than have each iteration of the loop slowly build up the model_transform.

One other small problem we had was keeping track of a player's coin collection. Every time a player collided with a coin, the collision_detected function would record a hit for each frame the player was colliding with the coin in. Therefore, each coin collected would add 1+ coins to a Player's score depending on how long they were colliding with the coin for. To somewhat mitigate this, we added a small time element needed to increase a player's coin score. We also only added 0.1 coins to a player's coin score each time collision_detected returned true. This made our coin collection system much more accurate, as most times a player's coin count would increase by 1 for each coin collected. However, this was not perfect and sometimes 2-3 coins were counted when a single coin was collected. To completely eliminate this error in the future, we plan to use booleans for each path row to see if a coin has been collected. If so, no other coin collisions would be recognized until the next path row was reached.

Lastly, if you played our game, you may have noticed that the player does not increase its speed during the game at all. We would like to improve on this in the future as having this feature would have made the game a bit more challenging and fun. However, we decided against increasing the speed of the player because this seemingly small change would have caused our game to run slower. Not only would this have meant calling our computationally heavy draw_path function many more times, but this would have also made the current coin collection algorithm much less accurate. Therefore, instead of actually increasing the speed of the player, chaser and camera, in the future, we would like to create the illusion of the player having increased speed by making the textures on the path scroll in the opposite direction to the player.

Contributions and Commit Links

Kia Mohager -
https://github.com/intro-graphics/team-project-team-rbr/commit/1a209477a8b6cc0994a5cbc769f

3efbe3e036dd1(Pits Implementation),
https://github.com/intro-graphics/team-project-team-rbr/commit/de87ae8c6b66fe70040541f3227184480024c0ec(Coin System Implementation

Devin Yerasi -
https://github.com/intro-graphics/team-project-team-rbr/commit/31b2104d8a5929c42a2e61d7a538a6d669140c6f (Collision Detection)
https://github.com/intro-graphics/team-project-team-rbr/commit/9c0bac317b826063146c89e699e5b8793f333b5c (Player Left/Right Movement)
https://github.com/intro-graphics/team-project-team-rbr/commit/ae2545de6f31515f9749485ba80227053bb4ee1b (Path Generation Function)

Akanksha Dhanwal -
https://github.com/intro-graphics/team-project-team-rbr/commit/769ee242a3a499ce3316ef254b8b6025698cdf2c(Screen Overlay for score/time)

https://github.com/intro-graphics/team-project-team-rbr/commit/40196a28e277f370591e5db20fd41e9b9996a179 (Coin Collisions and Score)

https://github.com/intro-graphics/team-project-team-rbr/commit/67fc10db1082ed0295520ffa46f8f6b6d24f6f1a (Textures)

https://github.com/intro-graphics/team-project-team-rbr/commit/a8b9e04312fc0af04d577efe5027ee7290025c23 (Shrubs)

https://github.com/intro-graphics/team-project-team-rbr/commit/dbe1b5ffdfdf2c8ab84aa2e6d807e3eced076cdc (Player, Chaser, Camera basic movements)

https://github.com/intro-graphics/team-project-team-rbr/commit/0af70d0a74fa80d34a2868665a2ab8fba675469d (Player Jump Movement)

Member Information

Akanksha Dhanwal: akanksha003; 304923935; akankshad@g.ucla.edu
Devin Yerasi: dyerasi; 305167818; devinyerasi@yahoo.com
Kia Mohager: kiamohager; 005355260; kiamohager@g.ucla.edu