RBR: Run Boy Run

Introduction

We will be designing a single-player platform game similar to Temple Run. The player will be confined to a rectangular playing field with a set width and infinite length. This playing field will be surrounded lengthwise by a forest scene that is purely aesthetic. The player will be continuously moving in the +Z axis at a pre-set speed while avoiding obstacles by moving around them or jumping over them. The player will be followed by a chaser moving slightly slower than the player. The object of the game is to continue moving forward to prevent the chaser from catching up. If the player hits an obstacle, their forward speed will stop until they move around the obstacle. The chaser will ignore all obstacles and continue moving forward at a set speed. If the chaser collides with the player, the game ends.

If we have time, we will implement a coin system for the user to pick up as they move along the path. A user's score will be a combination of their elapsed time playing and the number of coins they pick up along the way. Our project focuses on user interaction with our graphical interface. We expect our graphics to be blocky, similar to minecraft.

We will incorporate a nature-adventure theme into our project whenever possible. Most of this will come from the design of our obstacles and background.

Objects

Character Objects:

Player - A human or stick figure controlled by user, can collide with other objects (that are not ambient)

Chaser - A monkey or rock that will chase the player from behind, if the chaser comes in contact with the player, game over.

Objects that slow down the player:

Objects to jump over:

Fallen Trees - Logs will be along the path that the player must jump over.

Pits - Holes in the ground that the player must jump over.

Object to move around:

Ruin wall - A wall that is on the path, player must move left or right to avoid

Shrubbery - A shrub that takes up an amount of space along a path, player must move left or right to avoid

Standing Trees - A tree on the path, player must move left or right to avoid

Objects that don't affect gameplay:

Ambience Objects: water, trees, shrubbery,

Water - Add in water to background, does not affect gameplay

Trees - Add in trees to background, does not affect gameplay

Shrubbery - Add in shrubbery to background, does not affect gameplay

Game Mechanics

Game logic:

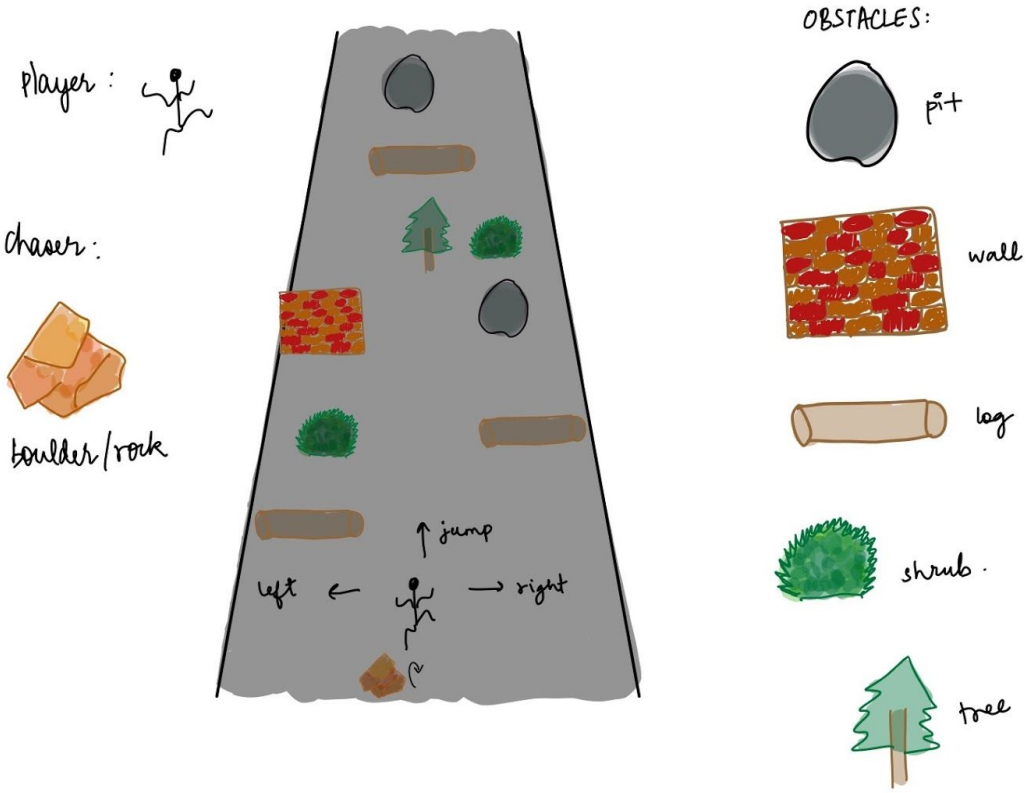

Figure 1. Showing the general scene and objects of the game. The actual game may look different.

We expect our game to look similar to the scene depicted in the figure above, but the actual details may vary depending on what we can get working. For example, right now the player (shown by stick figure in figure 1) is being chased by a boulder, but we may end up using a "Monkey" object to replace the chaser. The general idea of our game is similar to that of Temple Run: a player continually moves forward without colliding with the chaser and has to simultaneously avoid running into certain objects placed in the scene. The basic rules of the game are described below:

1. There is a clock running in the background that records the time a player spends "staying alive". If there are multiple people playing the game, whoever spends the longest time in the game wins.

2. The player can use the up, down, and right keys to jump, move left, and move right respectively.
3. If the player collides with an object from the environment, the player comes to a halt and has to use the up, left, or right keys to go around the obstacle without wasting too much time so that the player does not collide with the chaser.
4. If the player collides with the chaser or falls into a pit (one of the obstacles depicted in figure 1), the game ends immediately.
5. There are multiple levels, with each one being a little bit more challenging than the last. Each new level introduces extra difficulty by increasing the number of obstacles on the road as well as the speed of the chaser.
6. Before the game begins, the screen will display a countdown starting at 3, and at the end of the game, the screen will display the text “Game Over” over a black background. These visuals will be achieved using screen overlay.

Camera Positions

There will be two camera positions available for the user. The first one is from the point of view of the chaser looking towards the player. It will be a ground level point of view looking straight into the +Z axis. The second camera position will be an aerial view from behind the chaser. It will be slightly angled downwards towards the player and the map in front of the player. The player may cycle between the two camera positions as they see fit to do so during gameplay.

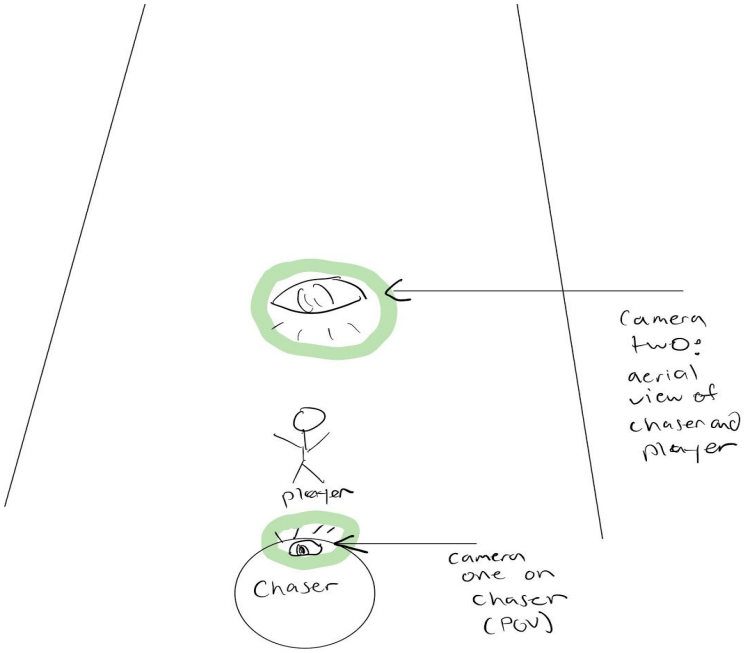

Figure 2: Shows camera positions

Course Topics Used
-week 1-4 topics all
-smooth shading, Mappings: Texture, Bump, Displacement, Environment
Shadows: 2-pass z-buffer, shadow volumes
-Lighting/Illumination: Ambient, Diffuse, Specular

Features:

Shadowing:

Our graphics will inherit a blocky model (Minecraft-esque) so shadowing will be constructed with respect to the models of each object. When it comes to shadowing, the most important is to first create shadows on static objects (trees, walls, shrubbery, etc.) . Once each static object has been implemented with its own shadow, we will focus on dynamic objects such as the player and the chaser.

Collision detection:

Collision detection will be implemented by checking whether a player and an object's axes overlap (or if the player falls within a certain range within the object's axes). If so, we will detect a collision has been made between the player and the object.

Screen overlay:

When the game is booted up, we will have a menu that at least includes the "Start" button (begins the game) and the "Exit" button (quits the game). When the "Start" button is clicked, a countdown will begin at three decreasing each second until the countdown is less than 1. While the current game is running, the player's score will be constantly calculated and displayed. When the player dies, a black screen will overlay with the player's final score and the text "Game Over" that will indicate the player has died.

Physics based simulation (only if we have time, but probably not):

If we have the basic and other advanced features properly working, with the extra time we have, we will try to implement physics based simulation into our game. If a player hits an obstacle, we will decelerate the player and slowly accelerate back to their normal speed. The objective of this implementation is to allow for the chaser to close in on the player, making it harder for the player to succeed in avoiding the chaser.

Workload Division:

There is no definite workload division so far, but we have a rough idea of what each team member should contribute. This workload division will be changed throughout the project. For now, Akanksha will handle our map generation and design. She will work with Devin and Kia to design the map and then implement it. We will decide between random map generation

with increasing obstacles as time elapses and a pre-filled map that will be reused for each game. Kia will lead our object model creation. We hope to use pre-existing models online for more complicated objects and generate simpler objects, like pits, ourselves. Devin will implement some game mechanics, such as user movement and scoring. All team members will help with collision detection and implementation of advanced features. All team members will also help test and debug the project.

Team Information:

Github Handle, Email ID, Student ID:

Devin Yerasi - dyerasi, devinyerasi@yahoo.com
Akanksha Dhanwal - akanksha003, akankshad@g.ucla.edu, 304923935
Kia Mohager - kiamohager, kiamohager@g.ucla.edu

Team Liaison: Akanksha Dhanwal (akankshad@g.ucla.edu)