CS175 Final Project Writeup

Dhamar Carrillo

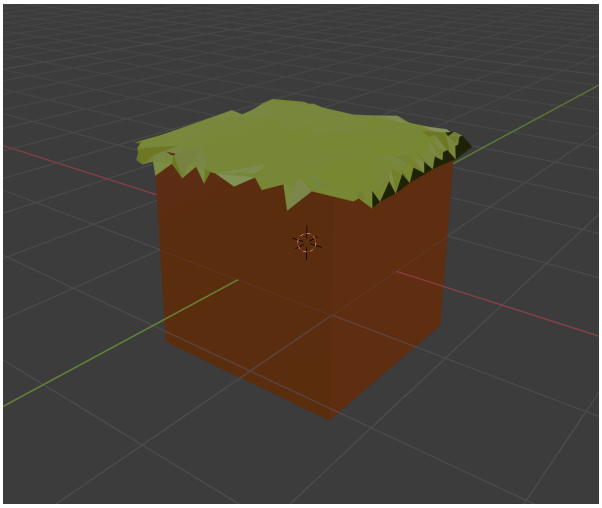
May 2024

Overview

For this project, I aimed to create a runner video game using Unity for the game implementation and Blender for the character design and modeling. The game incorporates an automatic running character that continuously moves forward, navigating through a terrace of obstacles, while being controlled by the user to perform jumps. The player's goal in the game is to stay alive as long as possible in order to reach the end of the game. The game can be played continuously without limit by restarting again.

Character Design

Starting off with the character design, I took inspiration from the last CS175 OpenGL assignment which involved animating a bunny's fur. Thus, the character created for this game is a bunny. I used Blender in order to design and model this character. I began by following tutorials on how to use Blender's interface and environment. After getting comfortable with the basic Blender features, I began practicing to model characters and objects by following YouTube tutorials. Taking inspiration from the animal characters, I began creating the Bunny character from scratch by borrowing and modifying features from the previous tutorials I had followed. In the end, the result was a 3D bunny standing on two legs that would be able to run across the video game canvas once implemented. In addition to the Bunny model, I also created dirt and grass blocks to include in the video game in order to give the game an environment and more imaginative feel, as opposed to using regular gray cubes for the terrace and obstacles that the character would face.

Unity Learning Process

As I was learning to use Blender, I was also learning how to use Unity. Similarly to how I modeled the Bunny, I also followed Youtube tutorials on Unity projects. I followed videos on the interface and basics of Unity as well as on some C# syntax. In order to practice for the video game implementation, I implemented Unity versions of Flappy Bird, 2048, and a rolling ball in a maze. After watching video tutorials on the endless runner game and the 2D Mario game, I began implementing my final game project.

Game Implementation

The game implementation took me several days to complete due to debugging and trial and error. I began testing my game with capsule and cube model placeholders first rather than with the character and environment assets I had created on Blender.

One of the very first issues that I ran into was that my capsule character would glitch and go through the cubes, rather than stop when it bumped against a cube. This became a problem especially when my character would go straight through the ground and disappear. I was able to resolve this by adding a box collider to each cube and setting the ground as a solid layer. Another issue that I struggled with was that my character wouldn't jump high enough to reach some of the cubes. In my code, I have a variable called 'jumpingHeight' which determines

how high the character can jump up. ALthough it might've been easy to simply increase the jumping height measurement, I wanted the game to still feel like a challenge and didn't want to risk my character shooting up too high into the sky every time that it jumped. Therefore, I decided to not change the jumping height from what I had originally set it to. Instead, I modified my code so that whenever the user presses the spacebar, the character will jump. Pressing the spacebar multiple times will allow the character to jump more than once and continue jumping higher with each time that the spacebar is pressed.

Throughout the process of completing the game, I combined ideas from the previous video games I had implemented and tutorials I had watched. After I had finished implementing the logic and mechanisms of the game, I replaced the capsule and cube placeholder models with the finalized Blender models I had created. Although the game is far from perfect or completely finalized, it runs smoothly and doesn't run into any major issues. If the character falls from the ground, the game presents a 'Game Over' screen and offers the opportunity to play again.

Take Aways

I learned a lot from this project and really enjoyed it. I had previously used Maya software in the past before and Blender gave me the opportunity to try out the same type of 3D animation modeling again. I enjoyed following tutorials to create characters and objects in Blender, which didn't feel too confusing overall, although some features were indeed challenging. This project also gave me the opportunity to learn how to use Unity and code in C#, both being skills that I had been wanting to learn for quite a while. I found that Unity's interface and development environment wasn't particularly difficult to work with. Moreso, it was the different settings and components that needed to be added or changed that made it a bit challenging, simply for the fact that it was a lot to keep track of. That being said, I definitely do think that working with Unity is much easier and flexible than working with OpenGL. Unity automatically provides a lot of the features that need to be manually coded for in OpenGL, and it does so in a much cleaner and user-friendly interface. I'll definitely be continuing to explore more of Unity over the summer thanks to this project.

Open Game

To open the game, please download the ZIP file from the Canvas submission (or alternatively, clone the GitHub repository into your desktop). Using UnityHub, upload the project folder into the Projects Workspace and open it. The game should open on Unity's game platform in fullscreen, for both MacBook and Windows. The character will start running immediately once the game begins. To make the character jump, press the SpaceBar. To have the Bunny jump higher or multiple times, simply click the SpaceBar a couple of times while playing. Play while avoiding to fall into any gaps or holes in the ground terrace. You'll know you've reached the end of the game when you encounter a downhill slide of dirt.