Untitled three.js Skiing Game

Alex (Ronny) Menendez, Stanley C. Kemp

December 16, 2021

1 What Each Member Worked On

Each group member worked on what they wanted to. Here is a general description of the jobs that each person worked on:

Alex Research, Presentation

Stanley Paper, Game Coding/Design

2 Objectives & Motivation

This group's objective was to create a game capable of running in the browser using JavaScript and the graphics library three.js, because the members of the group wanted experience with interactive graphical games.

3 Project Description

The completed project would be a game about skiing that would run in the browser. It would be a top down 2D game navigated with the arrow keys.

3.1 Premise Details

The score increases with every tick of the clock. The game is to dodge forward-moving trees while snowboarding in order to stay up as long as possible, as the player would take damage to a finite healthbar if they were to hit a tree. Another hazard that we would add is the monster which chases after the player. The player would avoid the monster, shooting projectiles that eventually knock it out of the game for a while.

3.2 Project History and Challenges

Originally, the game that would be implemented was a game where a player would fit an irregular object through a hole in a surface in order to progress to another level. It was later decided that there was too little time to implement it before the end of the semester, so instead another premise had to be chosen for the game. The chosen idea was inspired by a youtube video on a 3D Dinosaur Game project. [1]

One challenge was the UI of the application. By design, three.js does not natively support 2D graphical overlays. There are multiple ways of overcoming this:

- Draw the UI using a canvas and exporting to an in-game texture.
- Overlay some native HTML elements and then update them.
- Overlay a 2D canvas, then draw the UI on top of it.

The group chose the third of these options.

Another challenge was having and designing assets for the game. Originally the group wanted to import models for our visuals. It was decided that to work around the CORS request issue, the program had to generate its own graphics out of the built-in graphical primitives, such as spheres, boxes, and cylinders, as well as the built-in materials, using the classes MeshPhongMaterial and MeshLambertMaterial, among others. This was the method used to make the assets of the game.

4 Implementation Details

This project was designed to pass a memory state holding information about through each function in a draw loop. The state of a game gets passed through each function in the structure as a variable, getting read and modified in order to produce change.

At the end, the state gets processed and turned into a visual representation on the screen according to the changes made and constants defined on the files, using a different state called the tapestry, which holds objects relevent to the graphics side of the application.

These are the functions of various files:

[1] This one: `https://www.youtube.com/watch?v=KJ38qCwFdy8`

`index.html` This is the file that the user sees. It contains the main loop of the code, input handling, the general HTML structure of the application, and constants that are necessary to the functioning of the code.

`three.min.js` This is a minimized JavaScript library made for production code. It contains the important graphics functions that we need to interface with WebGL and GLSL. It is included as a tag from `index.html`.

`internalgame.js` This file gets included from `index.html` as a tag. This is how the internal model of the game acts. The game executes the functions in this code to know how the game should act internally, and it holds details about hit detection, damage, and motion.

`initialState.js` This state holds details about the state that gets initialized at the start of the game, and saves it to another variable that holds a copy of the initial state for next rounds.

`externalrepresentation.js` This file is also included from `index.html` as a tag. It translates the current game state from an internal representation to an external one on the screen.

`graphicsInit.js` This file contains the initialization code for all the graphics. It configures the tapestry, lighting, and sets up several graphical assets, as well as the UI canvas.

5 Results Analysis

At the time of writing, this project has not yet been finished. However, we have succeeded in getting the graphical portion of the game mostly coded. Some of the code that wasn't linked up was the hit detection portion of the code, as well as the motion of the trees forward, although they had been implemented, they were not linked up to the code.

6 Next Steps

To progress forward, the group should:

- ☐ Figure out why the game crashes occasionally
- ☐ Connect hit detection to the rest of the game

- ☐ Spawn new trees
- ☐ Add the monster
- ☐ Make the graphics better