Midnight Mountains - Final Project Write-up

David Paulk, Channing Huang, Christopher Giglio

Honor Code: This paper represents our own work in accordance with University regulations.
Signature: /s/ David Paulk, Channing Huang, Christopher Giglio

1 Introduction

1.1 Goal

The aim of our project was to implement a simple video game with a focus on quality graphics. The user's objective is to collect as many coins as possible while navigating a mountain terrain. When the user collides with the mountain, the game ends. As the user performs the coin-collecting task, the scene changes in several ways: a day-night cycle reflects the passage of time, the coins float about their spawn-points according to a periodic function, and the player gains speed as more coins are collected. Also, the light emitted by the sun, the players, and the coins affects the surface of objects in the scene such as the mountainside and the fog.

1.2 Previous Work

Our project builds on the work of several other projects. Our infinite mountain range was implemented using the three.js WebGL Terrain Demo (need citation - $http://threejs.org/examples/\#webgl\_geometry\_terrain$) as start code. The day-night cycle that makes the sky dynamic was implemented using [a WebGL demo] (need citation) as start code. The coins float according to a periodic function used to render Three.PointLight objects in the three.js WebGL Point Lights Demo (need citation - $http://threejs.org/examples/webgl\_lights\_pointlights.html$). The game music was taken from the Mario videogame (need citation).

1.3 Approach

We tried to implement normative mapping using a mountain texture. Our implementation approach for adding this feature did not work because it could not be used simotaneously with the game menus and continuously rendering

the mountain texture as new mountain terrain is spawned proved to be computationally expensive, resulting in noticable latency.

2 Methodology

The following subsections describe each part of the video implementation in further detail. The parts include sky rendering, coin rendering, player perspective, navigation controls, mountain terrain spawning, collision detection, and extras (i.e. game menus and audio tracks).

2.1 Sky Rendering

2.2 Coin Rendering

It is the goal of the user to collect as many coins as possible before crashing into the mountainside. A coin is represented in Javascript as a THREE.Object3D in the shape of a sphere and contains a THREE.PointLight to illuminate it, which is particularly convenient during the night when it is harder to navigate the terrain. The coins are animated by moving the x, y and z coordinates of their positions according the periodic function, relative to their spawn position.

$$x+ = sin(time * 7) * 3$$

$$y+ = cos(time * 5) * 4$$

$$z+ = Math.cos(time * 3) * 3$$

The position update is small enough so that animating them does not make it more difficult for the user to collect coins. When the user's player collides with a coin, the score increases by one and the coin is removed from the scene. One implementation challenge that we faced was figuring out how to spawn and update coins in a way that would avoid noticable latency. While one approach would be to limit the amount of memory occupied by each coin, this approach would prevent us from adding extra coin features such as lighting (glow). A better approach for avoiding latency that we chose was to limit the number of coins present in the scene at any point in time. We did this by first adding a maximum distance from the player that the coins can be spawned and updated at (7000 units). Second, we limited the number of coins that could be present at any time to 10, so that after 10 coins are spawned, no more can be spawned until one or more is collected by the player.