3D Adventure Game

Gendong Zhang
Stanford University
zgdsh29@stanford.edu

Abstract

In this project, I designed and implemented a 3D adventure game in three.js, similar to the popular games such as Fortnite or Counter-Strike. Users can explore the game by choosing their favorite weapons and following the instructions, such as acquiring the key and annihilating enemies, to win the game. The equipment provided by the course are sufficient to play the game.

1. Introduction

Nowadays, game players, from ordinary people to professional gamers, are more demanding in image quality, sound effects and immersive experience. Virtual reality provides a way for them to fully enjoy the game in a 3D environment. Inspired by the movie Ready Player One, which describes a virtual reality entertainment universe, I created my own preliminary game that contains most general virtual reality game features including IMU tracking and general keyboard control.

2. Related Work

Many people design the games on different platforms with different tools. Unity is one of the popular real-time creation platforms that people can use to create games. Space shooter games like Galaxian and other shooting games are created using Unity. Other tools like Three.js are also used in game design and 3D computer graphics and animations. Since most of the coursework are done in Three.js, so I decide to use Three.js as my design tool. Many of the games created by Three.js are 2-D or at least not for VR hardware, and my goal is to use the knowledge learned from class to create a VR game that can let the players immerse in the game.

3. Game Design

3.1. Game Plot

At beginning, players are put in a small village guided by some evil characters. They have to obtain the banner to save the village from the hand of demons, meanwhile they also have to eliminate the enemies. They can choose their favorite weapons locked in the tents, and the key to the tents are hidden in a secret place surrounded by the enemies. Players have to obtain the key in order to activate the weapons to kill the enemies and bring peace to the village. However, players cannot have contact with the enemies or their health will drop and eventually lose the game. Therefore, at first players has to dodge the enemies to get the key. Once the player unlocks the weapon tents, the weapons can be equipped, such as uzi, machine gun or even rocket launcher. Different weapon has different degree of damage on the enemies, and the rocket launcher only has ammo (rocket). Once all the enemies have been killed, the door will disappear and the winning banner is ready to be grabbed by players. Players simply go and grab to banner to win the game.

3.2. Interaction Design

3.2.1 Contact detection

In this game, in order to grab the key, weapons or banner, players have to make contact with the 3D models rendered in the scene. I use the bounding box of each 3D model to mark the boundary of the object, and check whether or not players' all three coordinates (x,y,z) are within the bounding box. If so, the contact is made, and corresponding consequence will show up. Even for checking whether the bullet has hit the enemies, the same idea is used except that several offsets are added in order to make the hit easier.

3.2.2 Building Blocks and Enemies Allocation

I use the models in kenny natural park extend pack [5] to build the villages including mountains, trees, flowers, walls, doors and etc. I also deploy several different characters to serve as enemies in different location in the village. The

character models are taken from Kenny.nl, and I used MTLLoader and ObjLoader from three.js to load the object and mtl files. For each character, different skins are also added. Some enemies moving back and forth, and some are moving in both x and y axis. In this way, the difficulties of obtaining the key or the banner and shooting are increased, so the player has to be more focused to win the game.

3.2.3 Weapon Selection and Control

Once players have the key, they can choose their weapons. Weapons are spinning around the y-axis, indicating that they are available. Players can use the weapons by running into the weapons, and even they are not satisfied with their current weapon or simply run out of ammo, they can swap the weapon with the ones spinning on the ground. Different weapon has different sound effect, and these are added by the audio elements in Javascript. All the sound track are downloaded from [7] and [4]. For each weapon, the corresponding ammo is used. In order to make the firing effect realistic, each bullet is set alive for one second, and after one second they will be removed from the scene. Each bullet also has an initial velocity and orientation pointing towards the target where the weapons points at. Each weapon has a gun-sign-on mode to help players aim the target more precisely, when the mode is off, the weapon is placed on the right hand side of the player's view and pointing toward where the camera looks at; when the mode is on, it is placed in front of the camera.

3.2.4 Action Control and IMU Tracking

Players have to use "WASD" keys on the keyboard to move forward, left, right and backward by changing the position of the camera, in another word add a translation in the corresponding direction. I was planning to use Lighthouse position tracking, but because players in the game have to turn around, and the Lighthouse might not be able to accurately track their positions, so I simply use "WASD" keys. For the orientation of view, IMU tracking is utilized. This part is simply taken from Homework 6 [1], and use a quaternion variable to store the reading from VRduino. Players are able to turn around to change the view. For turning on/off the gun-sign-on mode, players can hit "V" on their keyboard, and by hitting "B", the weapon can be fired. In addition, in order to jump over an obstacle or fence, "Space" key is used for the player to jump.

4. Equipment

The hardware for this project are VRduino and ViewMaster provided by EE 267 course.

5. Conclusions

This game has been successfully implemented and it also has the general features of immersive VR game. Although the game effect might not be as good as using other modules like Unity, it still can offer user good experience when playing the game.

6. Game Screenshots

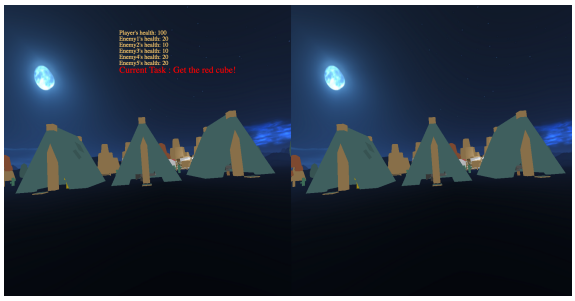

Figure 1. At the start of the game and the weapon tents, current task to grab the red cube

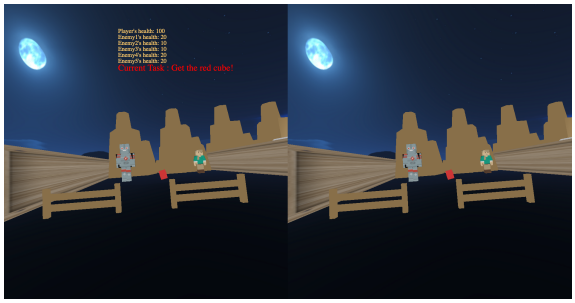

Figure 2. The red cube (key) to open the weapon tents

Figure 3. Enemies guided the red cube and banner, current task is to kill the enemies

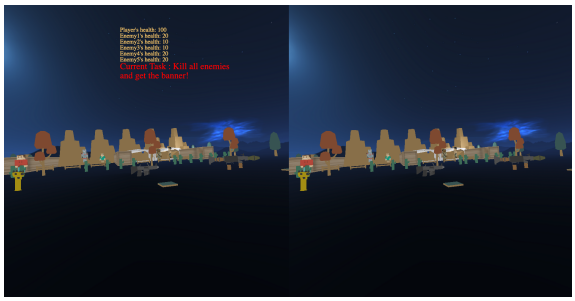

Figure 4. Available Weapons

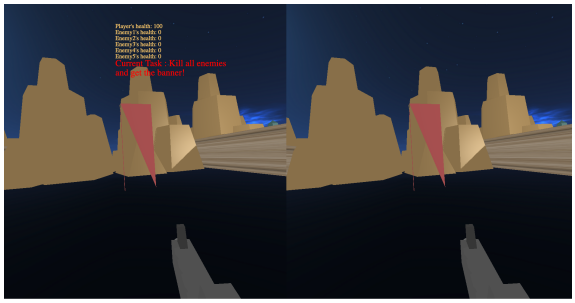

Figure 5. Banner to win the game, current task is to grab the banner

7. Future Work

The future work can be include motion detection such as using handlers to control the action of the character instead of using the keyboard. Also the lighthouse can be integrated wisely despite the fact that the player has to turn around and move in all directions.

8. Acknowledgements

For this project, I took the suggestions of TA Hayato Ikoma, who asked me to follow online tutorials of how to design games with Three.js and how to add sounds to game. I went through one online tutorial [6] and learned how to load object file, design the bullet trajectory and the use of loading manager. I took a small portion of the code from the online tutorial [3] and modified it based on my own design. For the necessary files such as MTLLoader.js, OBJLloader.js and three.min.js, I downloaded them from the open-source Github of threejs.org [2] The majority of the design and implementation were original.

References

[1] https://github.com/computational-imaging/EE267 Spring2019/tree/master/homework6.

[2] https://github.com/mrdoob/three.js/.

[3] https://github.com/saucecode/threejs demos.

[4] https://opengameart.org/.

[5] https://www.kenney.nl/.

[6] https://www.youtube.com/watch?v=axGQAMqsxdw.

[7] http://www.audiomicro.com.