Space Shooter VR

Victoria Chiu and Michael Kossyrev

I. Introduction

For the final project, we decided to construct a VR space shooter game. This used the provided HMD with Vrduino to track the user's head motion. Mouse and keyboard controls were used to play the game.

II. Gameplay

The player is set as the first-person controller of an invisible spaceship. The game consists of flying the ship through space with 6 degrees of freedom of movement and shooting asteroids with the two attached laser guns. Asteroids periodically spawn in a randomized zone around the player with their velocity direction pointing towards the player's position at spawn time. Each asteroid destroyed adds 1 point to the score. The game is over when the player gets hit by an asteroid. Figure 1 shows an asteroid exploding after being hit by a bullet.

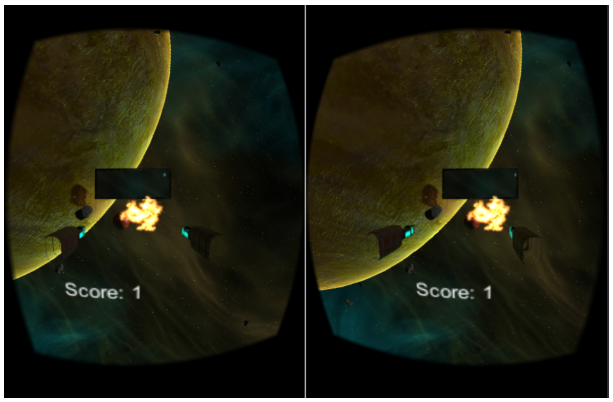

Fig. 1. Asteroid Explosion.
Certain artifacts, like the disappearing gun components, are not present when viewed through the HMD.

III. Environment and User Interface

The environment consists of a 50-unit radius boundary sphere. Any objects (except the player) that come into contact with the boundary sphere are destroyed. The player has their position clamped to make sure they stay inside the boundary. This can lead to some discomfort due to rapid velocity changes at the border, but this is mitigated by having a wide-open skybox. Our skybox uses a space theme with a large planet on one side. Lighting is rudimentary, with a single directional source providing sufficient illumination for the scene. The player UI consists of a score counter at the bottom of a standard (ie straight-forward) Field of View (FOV), which stays static with respect to the ship, but not the player camera rotation. This also applies to the rear-view mirror at the top of the standard FOV. In this way, these fixtures are essentially part of the ship, and not part of the player Heads-Up Display (HUD). The player also has two gun objects in their HUD, one on each side. These do rotate with the player head, so the player shoots in the direction they look.

There are also 6 planetoids around the boundary with one in the middle, to help let the player know they are moving; we did not use space dust, so without the planetoids, and if no asteroids are in sight, it's hard for the player to know that they are moving. Figure 2 shows a typical scene in the game.

We added audio effects for when a bullet is shot and when an explosion occurs. In general gameplay, there is background music to enhance the game experience.

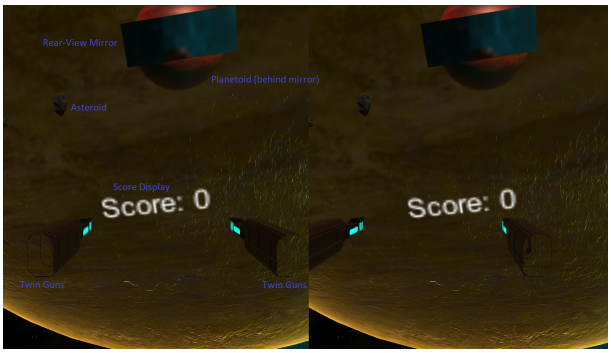

Fig. 2. Typical scene.

IV. Controls

Control is handled using the "WASD" keyboard keys to adjust the pitch and yaw of the ship. The space bar and left shift keys handle acceleration forward and backward. The left mouse button shoots bullets out of the guns. Once a "Game Over" state is reached, the player can either close the game by pressing "ESC" or restart by pressing "R".

V. VRduino

The VRduino contains an IMU consisting of a gyroscope and accelerometer. We use basic sensor fusion to create an accurate quaternion rendering of the player's orientation. Position is not tracked, since we implemented the controls on a computer's keyboard and mouse. The VRduino sends quaternion information over a serial port, which is then grabbed by Unity and implemented in the player camera using the ReadUSB2.cs script. The camera's orientation is set to be the same as the ship's, with the added offset

of information from the VRduino. Google's carboard SDK for Unity was used out-of-the-box to implement the stereo vision.