Crow Hunt: A First-Person Shooter VR Game

Tate DeWeese
Stanford University
tdeweese@stanford.edu

Kairen Ye
Stanford University
kairenye@stanford.edu

1. Introduction

Our project is a virtual reality crow hunt game implemented in Unity. It is a first-person shooting game and utilizes the course's provided head-mounted display (HMD) and the VRduino board. The game was partially motivated by the Duck Hunt arcade game from Nintendo, where a user stands in a stationary location and shoots ducks that appear on the screen without hitting the dogs. Our game implements a similar 3D version, but with flying crows and cardinal birds. Similarly, in our game, the player should target crows without hitting cardinal birds. The player aims by turning his or her head and fires shots using keyboard controls.

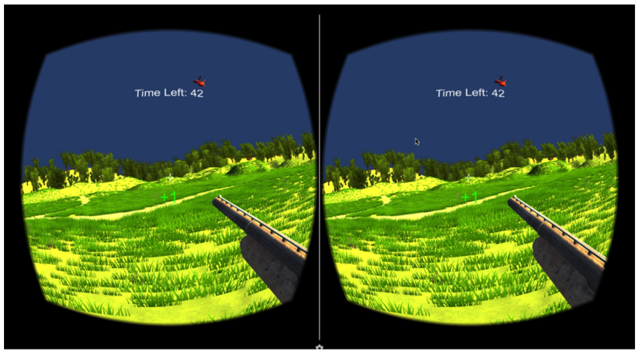


Figure 1: Stereo rendering of the Crow Hunt game in Unity editor's Play mode. A cardinal bird appears in the player's field of view. The time left in the game and the green +1 hit indicator are also shown.

2. Setup and Equipment

2.1. Orientation Tracking and Optimizations

Our game relies on orientation tracking so that the player can aim by rotating his head. Orientation tracking is achieved with the VRduino's on-board accelerometer and gyroscope. We used our own code implemented in homework 5, which only performs orientation-tracking related computations and not pose-tracking ones. This decreased the time to compute and stream the rotation quaternions. However, even with the homework code, we still observed a poor head rotation update rate within Unity. Knowing that the provided `ReadUSB.cs` script only reads the `QC` tag from the serial print, we removed all other unrelated print and function calls in `vrduino.ino` file, significantly improving the head rotation update rate in Unity.

2.2. Game Engine and Stereo Rendering SDK

We developed our game under Unity Version 2019.3.0a4 in MacOS with a Personal license. For stereo rendering, we used the Stereo Rendering SDK provided by the course staff, which is based off of the Google Cardboard SDK and adapted for the screen installed in the ViewMaster. With this setup, the game is run in Play mode within the Unity editor.

2.3. Input and Output Devices

The LCD screen serves as an external display to the PC and displays the maximized Unity "Game" window when the game is running. The external screen and the PC have to be connected via HDMI. The VRduino board needs to be plugged into the PC via USB so that the quaternion updates could be streamed over serial. The keyboard on the PC is also used in our game for firing shots.

3. Game Design

3.1. In-Game World

To make our in-game world realistic, we used an outdoor nature scene powered by the *Nature Starter Kit 2* Unity asset. We removed all tree objects in the scene to make an open space for the player. We also used the terrain colliders so the bird objects wouldn't vanish into the ground.

3.2. Bird Objects

In the game, birds spawn randomly and have random fly paths. We use the *Living Birds* Unity asset and utilize its `livingBirdsController` prefab to achieve this effect. We tested the controller and set the ideal number of birds parameter to 20, an unspawn distance to 40, and a bird scale of 20 for best visual experience. Two types of birds are used in our game: crows and cardinals. We also dispersed a total of 21 `groundTarget` around the player (see Fig. 2). The `groundTarget` attract birds and therefore help keep the birds around the user for better in-game experience.

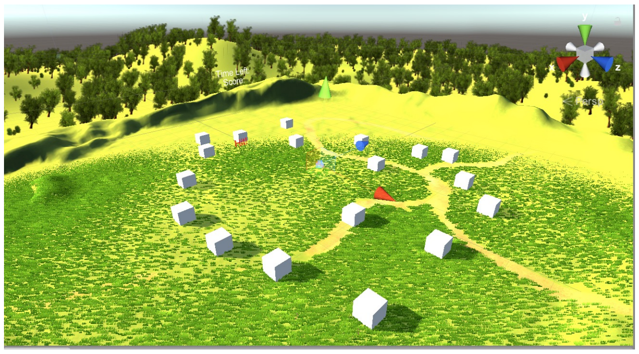


Figure 2: Bird's-eye view of the 21 `groundTarget` objects dispersed around the player to attract the birds. The Mesh Renderers have been turned on for readers to see these objects, but the player won't see them with the Mesh Renderers off.

3.3. Player Viewmodel

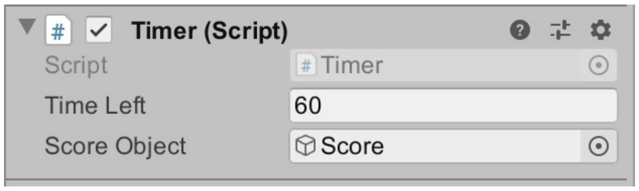
Two objects appear in the player's viewmodel: a shotgun and a 3D crosshair. They are powered by the *Hand Painted Shotgun* asset and the *Simple Modern Crosshairs: Pack 1* asset, respectively. We attached the shotgun model and the crosshair model under `CardboardMain`'s `Main Camera`. We fine-tuned the transform positions of these objects for best visual experience.

3.4. Player UI

The player's UI has three components: Time Left, Score, and Hit Indicator. These UI components are inserted as 3D Text objects and attached to `CardboardMain`'s `Main Camera`. Again, we fine-tuned the transform positions of these 3D texts for optimal visual experience. We wrote scripts for all three UI components to keep them updated in the game. The Time Left and Score indicators are managed by the `Timer.cs` script. The Hit Indicator shows a green +1 on the screen if the player hits a crow and a red -1 if the player hits a cardinal. The Hit Indicator's fade-out behavior is managed by the `HitFade.cs` script, which decrements the text's alpha value over time. The scores and the hit indicator's texts are updated upon hitting a bird, which is implemented in the `KillBird.cs` script.

3.5. Bullet Firing and Hit Detection

The gun model has a `Shell` and a `ShellTransform` attached to it. The `ShellTransform` is not a concrete object in the scene, but simply tracks the shell's current position and rotation. Whenever the player fires a bullet by pressing down space bar on the keyboard, the script `StartShot.cs` instantiates a new `Shell` object as a rigid body with the `ShellTransform`'s position and rotation. The new shell instance is then infused with an impulse force with an impulse parameter of 10, which we tuned for visually optimal bullet travel speed. Whenever the bullet collides with an object, the `KillBird.cs` script checks whether the other `Collider` object's name belongs to a crow or a cardinal. If so, the hit indicator and the score are updated accordingly. A shell object is destroyed upon hitting a bird or hitting the ground. Note that if game is over, these updates are not triggered any more.



Timer (Script)

Script	Timer
Time Left	60
Score Object	Score

Figure 3: You may adjust the `Time Left` field (in seconds) in Unity editor to adjust the length of the game. The default value is 60 seconds.

3.6. Sound Effects

The game has two sound effects: one upon firing a shot and one upon hitting a bird. The gunfire sound is powered by the *Post Apocalypse Guns Demo* asset, and the damage sound is powered by the *8bit SFX 01 - Shooting Game (Lite)* asset. The gunfire sound is triggered in `StartShot.cs`, and the damage sound is triggered in `KillBird.cs`.

4. Controls and Gameplay

4.1. Starting the Game

The game is started by clicking the Play button in Unity editor. The Time Left indicator will start counting down from 60 seconds (adjustable as seen in Fig. 3). After you start the game, the birds will start spawning, and it may take a few seconds before you can see any birds around you. Your score is initialized to 0 at the beginning of the game.

4.2. Playing the Game

Rotate your head to find birds around you. When you see a crow, use the crosshair to help with aim. Press the keyboard's space bar to fire a shot. Hitting a crow would add 1 point to your score. Hitting a cardinal would deduct 1 point from your score. Your score is tracked in the background and will be displayed once the game is over.

4.3. End of Game

Once the "Time Left" counter reaches 0, the game is over, and the "Time Left" text on the player UI will be replaced with "Game Over" text. Then, your final score is displayed. After game is over, while you can still fire shots, birds hit will not die or disappear. You can restart the game by ending and restarting the Play mode in Unity editor.