# CS 135 VR Final Project

**Group members:** Kai Wen Tsai, Yulin Zhang

## Abstract

This is a first-person shooting game. It introduces a VR shooting perspective with two different kinds of gaming experience (air/ground). The game uses many Oculus functions such as controller haptic feedback, aiming system, distance grab, and moving speed adjustment on the hand trigger.

## Links to project video and github

https://www.youtube.com/watch?v=nkFMDApJ8iA

https://github.com/kevint1221/CS135/tree/master/Final_project

## Introduction

The project is a FPS game. There are different enemies with various shapes, amount of health and attacks in several levels. We incorporated 2 game modes, both in the air and on the ground. The project is interesting in ways that it utilizes Oculus specific controlling methods with the Touch controllers.

VR puts you right there in the scene, as you see the orcs in the ground mode rushing to you and shooting bullets at you, you feel the pressure and risks not found when playing on a monitor. Part of the challenge is how to let the player move around in the scene without changing the landscape too dramatically, which easily causes motion sickness. The movement speed and the rotation speed of the character had to be toned down to mitigate that issue.

In a traditional FPS game, the gun is fixed to the character camera angle, and the crosshair is fixed to the center of the screen. In a VR setting, the direction of the weapon being pointed at can, and is much preferred to move independently from the camera angle, which mimics the real life situation and provides a more realistic experience. In fact, not only the weapon, but the character movement is also independent from the camera movement, which again increases the realistic aspect of the game.

There is interaction with items in the game. Ammo crates give plyers more ammo, and the red shining potions replenish players health. We explored different options in the two play modes, where the air game stays in a scene and enemies keep spawning there, and the ground game switches scenes after certain conditions are met (level up with success or reset with failure).

We made the game in a way that it can be both challenging and casual. Both combat modes have increasing difficulties with the level progression. In the ground game, players start with a lot of ammo crates and health potions, and the amount of supplies decrease as the level goes on. By level 5, there is only 30 ammo total and 25 enemies to defeat, so players need to make every shot count. On the other hand, players have unlimited health and ammo in the air game, so players can enjoy a spray and pray experience.

We made efforts to deliver a VR friendly experience, such as using UIs that are fixed in the world space, as having texts moving along with the camera rather feels like having dirt stuck on the lens. We also adjusted the sizes of the enemies, items, buildings and landscape to make everything appear life-sized. These contribute to providing an immersive VR game.

## Related works

One of the groups created a basketball shooting game. The player can also dunk if they want.  Another group created a baseball game. Both games try to simulate real sports experience. The downside of it is the lack of content or challenges. These two games' scales are relatively small. Players can only experience one kind of activity. Take the basketball game as an example, there is no opponent to compete with the player (think Wii Sports). The only feedback is the score the player gets from hitting the wall.

The basketball game uses the Touch controllers' grab function as well. Our game is set in a rather open scene, where the basketball game happens in a downsized court and there is a ball delivery mechanism for the player, so the basic grab function is sufficient for that game. We implemented the distance grab function with the weapon, which is a more convenient grab method. The upcoming VR game Half-Life: Alyx, which of course is a much more completed and polished game, uses distance grab with the in-game items "gravity gloves", which sounds futuristic and convincing.

## Design

Details:

We customized the game scene landscape by using both the environment package from Unity Standard Assets, open source files and the Unity Terrain Tool.

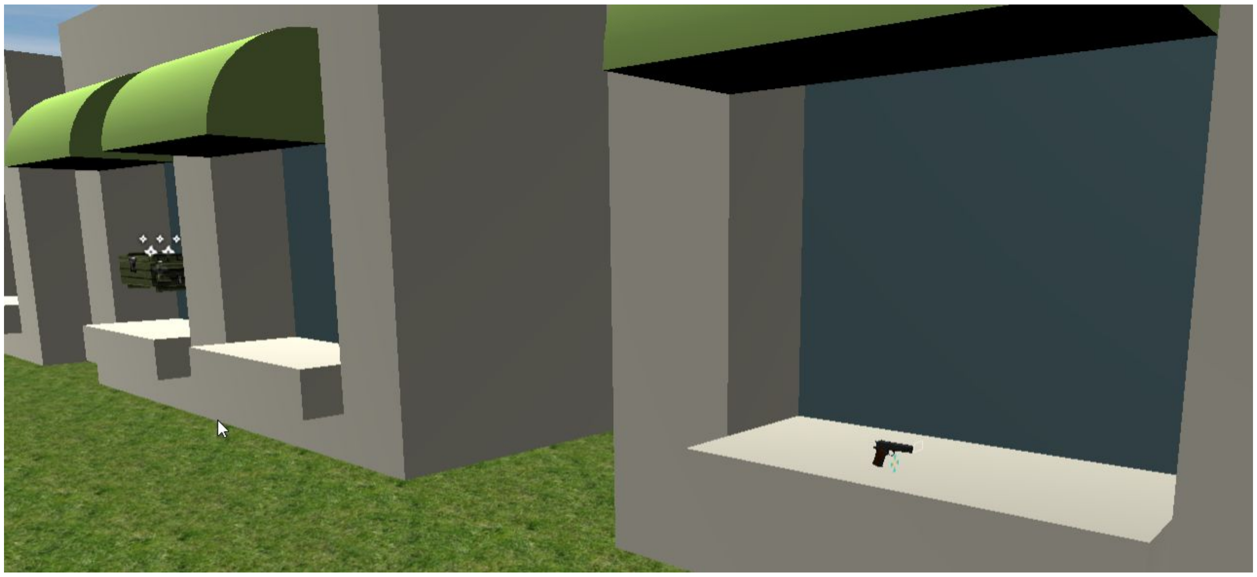
Ground mode:

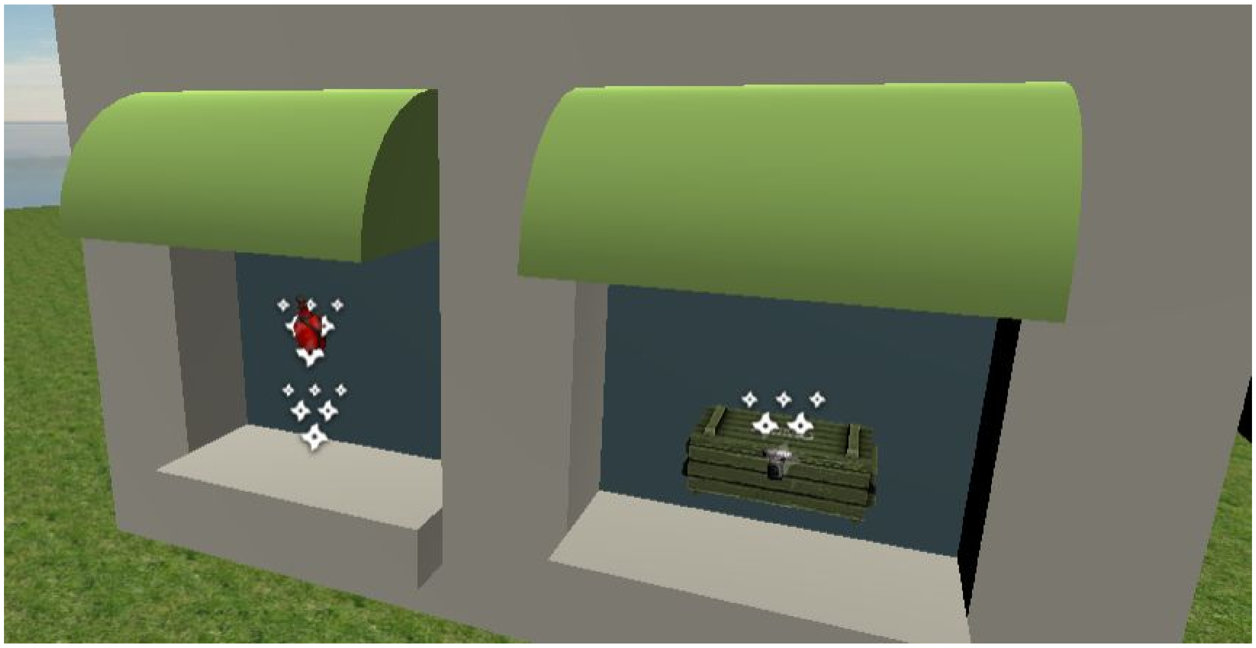
- OVR character controller - this is the VR character that players use to interact with the game. Players start with a certain amount of health and ammo and try to defeat all enemies through all levels.
- Weapon: A handgun from Unity Asset Store, has a provided script that provides animation. It can be grabbed from a distance.
- Ammo crate: replenish player ammo.
- Health potion: replenish player health.
- Enemies: chase and shoot at players within a certain distance. Bullet damages and touch damages are preset. Players are invincible are a brief moment after taking damage.
- Levels: 5 levels in total. Killing all enemies in the level will enable players to proceed to the next one. Player health dropping to or below 0, or having enemies left while all ammo are used up (including the ones in the ammo crates) will result in a failure and scene reset back to the menu room.

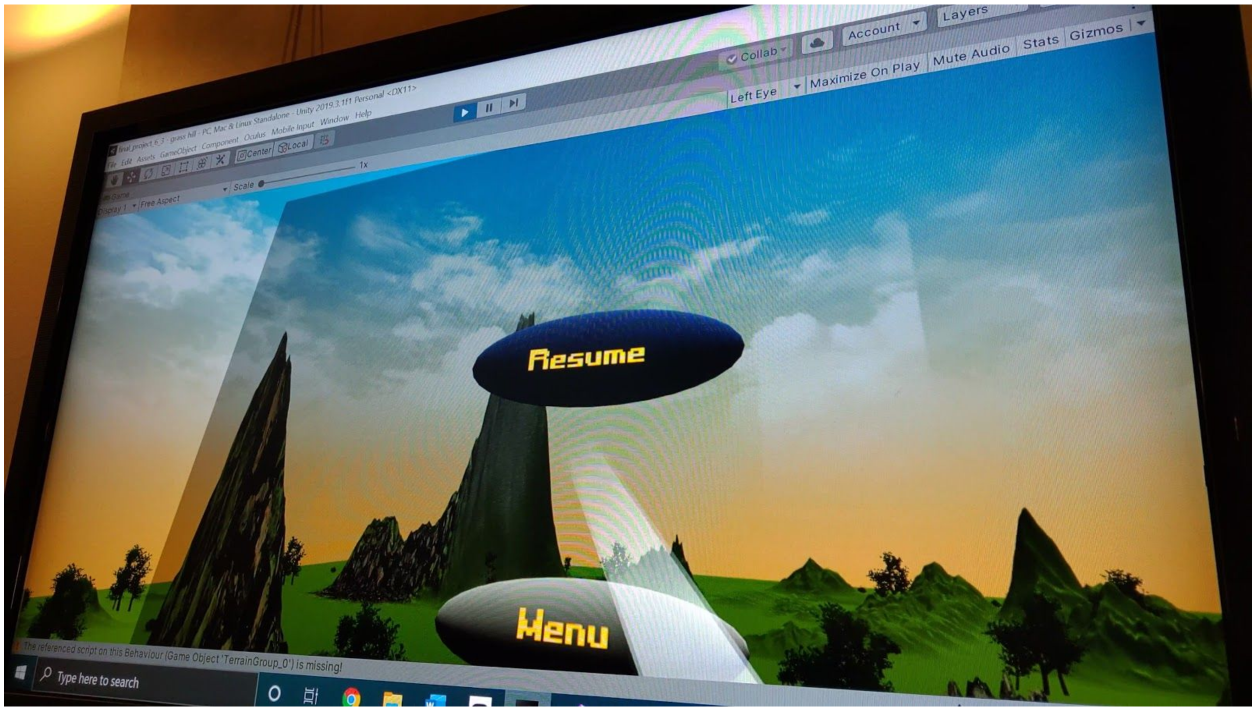
<u>Air mode:</u>

- OVRController:
    - haptic feedback when player shoot
    - hand trigger to increment speed
    - thumbstick to rotate aircraft
    - start button to pause the game
- Levels:
    - each level has small minions and one large minion
    - when use defeat all enemies, game automatically generates next level enemies
    - there is a final boss in the last level
        - stronger health
        - bigger attack hitbox
- Pause Menu:
    - pause the game while playing
        - player can resume the game or move to restart menu scene
- Terrain:
    - apply terrain tools to create terrain and objects
        - mountain
        - lake
        - tree
        - bush
- enemies behavior
    - sheep moves in circle around player
    - small aircraft moves in specific route in them map
    - cow moves around lake

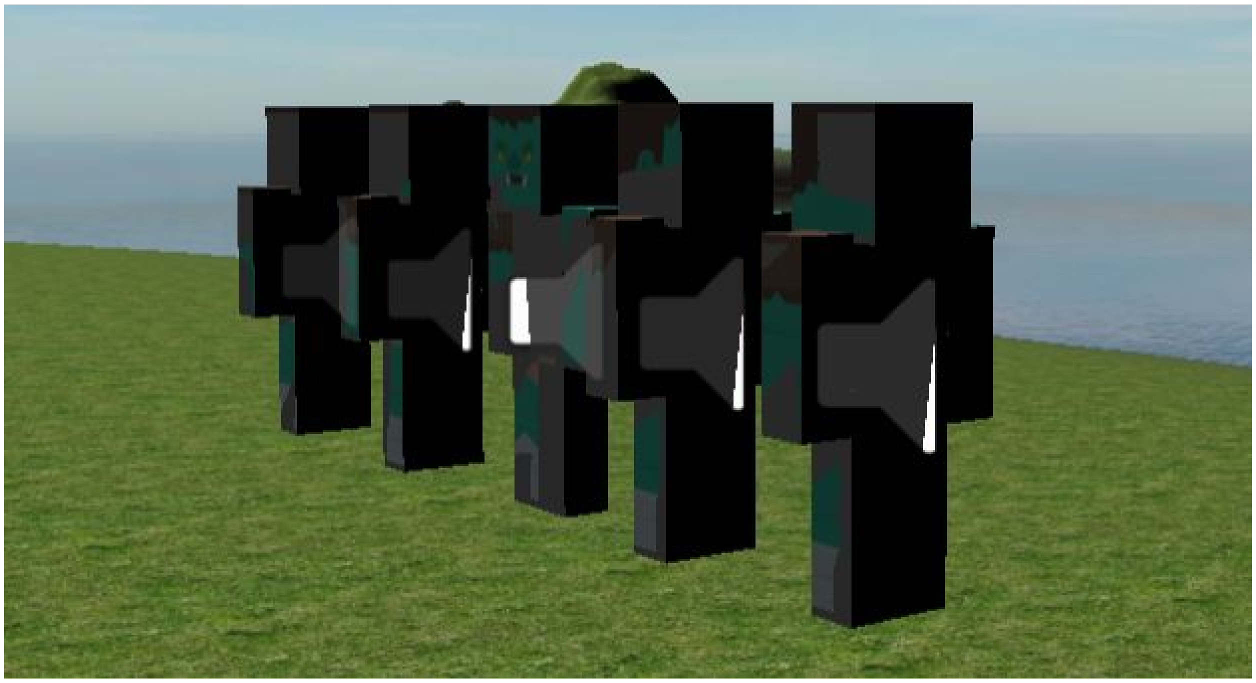
    - big aircraft follows user with minimum range
- player behavior
    - flare is created when player shoot
    - smoke and explosion are created when bullet hits an object
- 
    -







The referenced script on this Behaviour (Game Object 'TerrainGroup_0') is missing!



Interface: Oculus Touch controller

- The independent hand and weapon movement from the camera and character movement is more realistic when shooting

  - Being able to move hands freely, and seeing their motions and positions in the virtual world simultaneously feels much more intuitive compared to sitting still and using keyboard and mouse
  - Controller vibration provides better feedback and interaction
  - The thumbsticks and triggers on the controller provide precise control as ranges of motion instead of simply on and off toggle, for example players can move at different speeds with how deep the triggers are pressed
  - Players with Touch controllers can move around while playing, which is probably more healthy

Oculus Best Practices:

- We use world space UI to provide VR friendly user experience
- We put up warnings at the beginning of the game about possible eye strain, disorientation and nausea along with other VR side effects caused by excessive use without breaks
- We tested the project countless times to ensure the game mechanics go smoothly and the UI are user-friendly

Use conditions:

- If the users are easily prone to motion sickness, our game might not be the most polished options for them, since our scenes are set in a rather open world

**Implementation**
The project is implemented with concepts learned from the labs, Unity Asset Store free assets, and customized scripts. It is developed for the Oculus Rift platform.

Ground scripts and assets:

- Enemy: parent class of shootingEnemy (Open source with modifications suitable for our project)
- ShootingEnemy: defines the behaviors of the enemies seen in the ground game mode, such as chasing players and shooting at players. UnityEngine.AI is used to enable navigation with Nav Mesh. Scenes are baked with restraints on slopes greater than 10 degrees to prevent enemies from going into the water. (Open source with modifications suitable for our project)
- ObjectPoolingManagers: bullets are pre-made and set inactive by this script. A bullet from the bullet pool is activated whenever the player shoots and deactivated after a preset time. When more bullets are needed than what’s in the pool, this script instantiates

more clones of them. (Open source with modifications suitable for our project)

  - LevelManager: checks if the level success or fail conditions are met and updates the level UI text. It uses coroutine methods to put a preset delay before switching scenes
  - GameController: updates the UI text for player ammo and HP
  - AmmoCrates: spins the ammo crates. The ammo crates in the scenes have particle effects attached. Materials are from Unity Asset Store.
  - HealthPotion: spins the health potions. The potions in the scenes have particle effects attached. Materials are from Unity Asset Store.
  - GroundBullet: defines bullet speed, damage and how long they exist.
  - GroundPlayer: defines player initial health and ammo, calls the handgun animation, set active bullets, interact with ammo crates and health potions and take damage from enemy bullets and touching. (Open source with modifications suitable for our project)
  - LevelCanvas: updates the level UI and displays the current level number.
  - DistanceGrabber & DistanceGrabbable: enables grabbing objects from a distance, is provided as a part of the Oculus Integration Assets. (Open source with modifications suitable for our project)
  - OVRPlayerController: enables VR control. (Open source as part of the Oculus Integration Asset)

Air:

- script
  - Bullet_effect: create visual effect when bullet hit enemy or object
  - Controller_mapping: all the mapping to oculus controller and keyboard for testing
  - EnemyGenerator: generate enemy with different level and response time
  - EnemyStats: store all the enemies stats such as damage, attack speed, bullet travel speed, health etc
  - ambient_sound: create audio sound for player when enemy near by
  - chase_player: move toward player/object with desire movement speed
  - enemy_attack: attack player, get enemy stats and create bullet
  - enemy_route: create enemy travel path
  - global_variable: variable that can be control through entire game

        - shoot_bullet: player can shoot bullet with desire sound/visual effect/shoot speed. Also control many audio such as hit sound
        - stay_around_player: move toward player, and keep in certain distance
    - prefab: prefab model such as sheep cow, aircrafts
        - 3DTOYS
        - Hessburg - stealth Bomber
        - StarSparrow
        - LowPolyWater_Pack
        - Wispy Sky
        - NatureStarterKit2
        - VertexColorFarmAnimals
        - Standard Asset

Other challenges:

- adjust bullet position/speed/hit point when it is generated
    - find the vector of the front end of gun, apply that to the bullet
- shoot visual effect does not align with gun position when player is moving
    - make effect become child of gun object
- when bullet hits terrain, the visual effect does not correctly position
    - adjust offset between hit point and terrain height
- Hand tracking and distance grab was a pain to implement, as the open source tutorials online often are outdated with slightly different scripts and object attributes.

**Lessons Learned:**

First of all we learned that developing a game is fun! We also learned that there are many different ways to implement high level designs, some more efficient than the others. VR is a new and fast-evolving industry, we used our own Oculus Quest for the development of the project, because Oculus Link enables the desktop use like a Rift S.

Kai is responsible for developing the air mode and Yulin is responsible for the ground mode development. The scenes and level designs vary from one to another, but in the end many of our scripts serve very similar functions, such as the ones that manage bullet-shooting. Since some scripts are more efficient doing their jobs, we could have had better communication to enable recycling the scripts. Saving the developement time and cost may enable us to further polish the game.