VR Assignment Report

Date: 16/05/2021

Student: Pozzoni Davide

Project Supervisor: Norman Murray

Title: Zombie Repair

Introduction

The VR game I developed is a Tower Defense VR game with randomization of enemy spawn.

The game supports SteamVR, and any HMD capable of running over SteamVR.

In the VR game, a player is a person surrounded by zombies with only some guns to defend themselves as long as they can. The guns are spawned into pieces, so the player needs to quickly assemble figure out how to assemble them to protect themselves.

The number of zombies killed and the time they survived and are shown as the final score at each round.

Gameplay

The initial state of the game has the Menu active containing colliders buttons to start the game or close the application as well as the title and a short tutorial text. While in the menu, a table is also active with three parts of a basic gun that the player can interact with and use to get comfortable with the assembling of the guns mechanic before the start of the game. That table is positioned right in front of the menu so that is easily visible to the player.

The player can assemble guns parts by grabbing them one on each hand. When the two components are correctly positioned and matched, they will snap together. Once a gun handle has at least one barrel attached to it, then the trigger can be used to shoot from that gun. If the player also adds augmentations the bun will be more powerful.

Once the player interacts with the Play button the game is started. All the menu specific items are disabled with the help of the ‘MenuToggle’ script, and the ‘Spawner’ script is activated. The ‘Spawner’ will activate on randomized intervals and, on each interval, it will spawn one or two enemies and a fixed amount of gun parts. On a bullet collision with an enemy, that enemy will take damage till it reaches 0 hp, at which point it dies. The bullets will also collide with the player causing a game over if the player shot themselves. If the player fails to kill an enemy before it comes too close, on collision with the enemy the player will die causing a game over. The ‘Spawner’ script also keeps track of the number of enemies killed in total.

On game over the spawner will stop the game and clear all active enemies and items and restore the state of the menu by calling the ‘EnableMenu()’ method of ‘MenuToggle’. It will also pass in the value of the total survived time and enemies killed so that the ‘MenuToggle’ script can update the

scoreboard text in the ‘GameOver’ object and activate it. From this stage, the player can interact again with the Play button to start a new game or exit.

Scenes, Components and Sounds

Main Scene

The game is entirely contained inside one single scene that is managed by scripts where necessary.

The scene contains multiple UI elements,

The player is surrounded by sci-fi scenery with neon cyberpunk stylization, a style first popularized by ‘TRON’, with a dark cybernetic environment highlighted by bright colours. All shapes are simple and essential resembling early computer graphics.

Components

SceneManager

The ‘SceneManager’ prefab is a simple game object containing all the main scripts necessary for the gameplay.

Spawner

It contains the ‘Spawner’ that is the main script responsible for managing all object in the scene during the game. It can spawn new instances of enemies and weapon parts and keep track of all spawned objects in lists. On each ‘Update()’ it will keep track of timers using ‘Time.deltaTime’ and when the enemy or items interval passes it will spawn new objects of the respective type.

The items spawn will spawn two augmentations, two barrels and one handle of the weapons. Of each of those item types, the final game object is picked at random from a corresponding list of prefabs. That allows to have various types of handles, barrels and augmentations and, by adding them to the corresponding list type, the ‘Spawner’ script will automatically randomize the spawn of the items so that the player can have different combination to choose from, but also making certain that at least a full set is always available to the player.

Similarly, the enemy spawn picks one random type of enemy to spawn from a corresponding list of prefabs, although at the moment only one enemy type is implemented

For both of those spawn behaviours, a random position will be determined based on a set distance from the player.

ToggleMenu

The ‘ToggleMenu’ script is responsible for enabling and disabling all menu related items.

The main methods are ‘EnableMenu()’ and ‘DisableMenu()’. As the name implies they will respectively enable and disable all objects that are necessary only for the menu. On top of that, the ‘EnableMenu()’ method will also update the content of the scoreboard at each game over.

UI Elements

The ‘Menu’ prefab contains two collider buttons to start the game or close the application as well as the title and a short tutorial text. The buttons events are linked by the ‘ToggleMenu’ script on ‘Awake()’ to the ‘SteamVR_LaserPointer’ event handlers of each hand. The handler ‘PointerIn’ and ‘PointerOut’ are used to highlight the button when the laser is inside that button by changing the scale slightly. The handler ‘PointerClick’ is instead used to trigger the action related to that button.

The ‘GameOver’ prefab contains the text for game over message and score. The text in the score is updated by the ‘ToggleMenu’ script at each game over to show the correct last score.

Weapons

The weapons a currently the most complex part of the game. They are divided into three parts that need to be assembled for a gun to work.

All parts are interactable object using the SteamVR ‘Interactable’ script that let you handle the object.

On top of that, they all have the ‘Item’ script that implements the method ‘Die()’. That method is used by the ‘Spawner’ script to destroy that item. Using that an item will remove itself from the list of active items and then Destroy itself.

Handle

The handle, or base of the gun, is the main component of a gun. It has two types of attachment point that act as an anchor point to snap other parts of the gun into position. Depending on the gun-type it can have multiple of each type.

The handle part has an additional modified version of the SteamVR interactable script called ‘WeaponInteractable’ that is responsible to call the ‘Shooting()’ action of the handle that the player is holding.

The ‘Shooting’ method is defined in a separate script attached to the handle called ‘WeaponScaffold’. That script defines the main behaviour of a gun, such as fire rate and the number of ammo.

If any augmentation is attached to the hand, the script will also keep track of additional parameters retrieved from that augmentation that will change the behaviour of the gun.

The handle will not actually shoot bullets as that is delegated to any barrel that is attached to the hangle.

This script is also responsible for destroying the gun and any attached parts when the weapon is out of ammo.

Augmentation

An augmentation has two scripts,

The main script is the ‘WeaponPluggable’ script. This script is responsible for attaching that augmentation to the matching slot of the handle. The stotting action is archived on the proximity of a pluggable object with the correct slot type of the handle. However, the slot action can proceed only if the player has a handle equipped and the augmentation is being held in the player hand, that way the parts cannot slot together by accident.

The secondary script, ‘WeaponAugment’ is a simple script that only has some public variables that let you declare the modifier values that that augmentation will apply to the gun.

Barrel

The barrel is an extension of an augmentation. It will indeed have all scripts described above at its root. On top of that, it has one or more barrel heads object that they have attached to it the ‘WeaponBarrel’ script.

The ‘WeaponBarrel’ script is the script responsible for shooting bullets. The ‘Shoot()’ action will take in the spread parameter and instantiate the bullet objects in various directions depending on the spread parameter and also play the shooting sound.

Bullet

The bullet is a simple rigid object and collider game object with attached to it a script, ‘Bullet’, that defines the speed, damage, and lifetime of that bullet.

Enemies

The ‘Emeny’ prefab has an ‘Enemy’ script that will handle the behaviour of that enemy.

On ‘Start()’ the script will find the player in the scene and set it as a target for that enemy. If the enemy has its target on ‘Update()’ it will gradually move towards the player by the amount determined by its speed.

The ‘Enemy’ also has a collider. On collision using ‘OnTriggerEnter()’ it will check the source of the other collider. If the other collider is the ‘Player’ collider then it will cause the player to take damage. If the other collider is a bullet, it will instead take damage by the amount defined on that bulled and die once the hp of that enemy is down to 0.

It also implements the ‘Die()’ method. Similarly to the items that method will remove the enemy from the list of active enemies and destroy that object.

Player

The Plater is a modified prefab of the SteamVR Player script. It has a VR camera and hands and a body collider.

On top of the SteamVR default player scripts, it also has the ‘PlayerHandler’ script. The main role of that script is to keep track of the life of the player. When the player takes damage it will play an associated sound and it will check if the player still has remaining hp, if not it will end the game.

Sounds

The sound design is intentionally silly using sounds effect all dubbed by me to match the cartoonish scenery. Each object that needs to do a sound contain an ‘AudioSource’ and is responsible to play their sound to maintain spatial directionality.

Where necessary the playing of the sound is triggered by the script of that object. For example, the barrel part of a gun on shooting the script will also play the sound, which pitch is also randomized to offer a small sound variation between each shot. Each barrel is associated with a different sound so depending on the barrels combinations that the player installed on a gun the resulting shooting sound will be different.

Movement and Collision

Movement is not necessary for this game as the concept is that the player is surrounded, and they need to hold their position. The player can only move within their real space. The only locomotion characteristic that is implemented is the snap angle rotation that will let the user turn their camera.

Collisions are supported by all components. Bullets collide with both player and enemies to give them damage. Enemies collide with the bullets and the player. The hand of the player object can also collide with all solid object in the scene, even the enemies but they will keep walking toward you even if you try and stop them with your hands.

Performance

The graphic design of the game is intentionally low poly due to stylization, so all models do not require a lot of memory to run.

All unused objects are always destroyed. The ‘Spawner’ script will keep track of all instances of items and enemies that are spawned during a game in lists and at the end of that game, it will destroy all objects contained in those lists. Those lists are also used to limit the number of objects that can be in the scene at once. Only a maximum of 200 enemies and 50 items can be in the scene at once. If the number of active objects is above that the spawn routine will not create any new enemy or item.