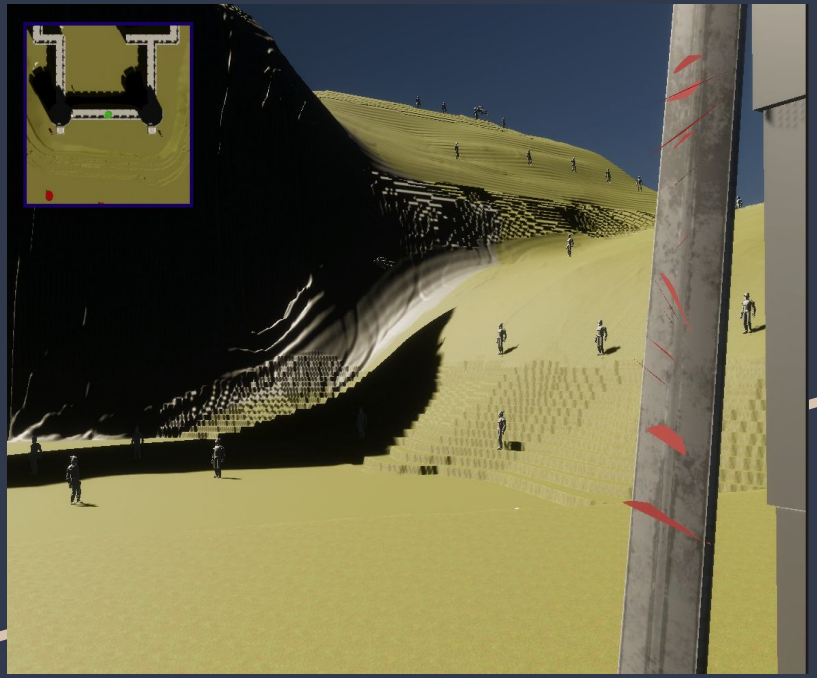
Slash: Unlimited

Samuel Ramos

Description



Slash: Unlimited is a single-player hack and slash game. It's similar to any other game that involves fighting hordes of enemy AI's while defending your castle. In a sense, Slash: Unlimited revolves around a stronghold invasion that serves as a strategic point to deploy troops from. The objective is to last as long as you can or destroy all the waves of enemies by retaking the mountains, hills, and flatlands in front of the stronghold. The game will end once the player dies or destroys all of the invaders. The game starts with a main menu screen that uses buttons to change the scene to the play scene.

Demonstration, Script Elaboration, Class Interaction all in Video

Class/ Objects

WeaponController.cs

```
using UnityEngine;

public class WeaponController : MonoBehaviour
{
    //references the specific GameObject and creates var
    public GameObject Sword;
    public bool CanAttack = true;
    public float AttackCooldown = 1.0f;
    public AudioClip SwordAttackSound;
    public bool IsAttacking = false;

    //method that runs by frame to attack and animate on
    //I made the pausemenu variable GameIsPaused to prev
    void Update()
    {
        if(!PauseMenu.GameIsPaused)
        {
            if(Input.GetMouseButtonDown(0))
            {
                if(CanAttack)
                {
                    SwordAttack();
                }
            }
        }
    }

    //public method that functions as a setter and gette
    //IEnumerators function as for each loops with an ar
    public void SwordAttack()
    {
        IsAttacking = true;
        CanAttack = false;
        Animator anim = Sword.GetComponent<Animator>();
        anim.SetTrigger("Attack");
        AudioSource ac = GetComponent<AudioSource>();
        ac.PlayOneShot(SwordAttackSound);
        StartCoroutine(ResetAttackCoolDown());
    }

    IEnumerator ResetAttackCoolDown()
    {
        StartCoroutine(ResetAttackBool());
        yield return new WaitForSeconds(AttackCooldown);
        CanAttack = true;
    }
```

```
}

IEnumerator ResetAttackBool()
{
    yield return new WaitForSeconds(1.0f);
    IsAttacking = false;
}
}
```

VolumeInGame.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Audio;

public class VolumeInGame : MonoBehaviour
{

    //creats a new AudioMixer by referencing the mixer you select
    //in the editor e.g. I reference the main mixer
    public AudioMixer audioMixer;

    //the values in the main mixer are exposed in the previous reference
    //so by dragging the volume slider,
    //we change the value which is exposed and is changed by the SetVolume function
    public void SetVolume(float volume)
    {
        audioMixer.SetFloat("volume", volume);
    }
}
```

CollisionDetection.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CollisionDetection : MonoBehaviour
{
    //references the WeaponController script and instantiates
    public WeaponController wc;

    //adds a slot where we can place our visual effect inside the script
    //component of the object housing the script
    public GameObject HitParticle;

    //method that uses an if conditional to reference the tag of the object
    //being hit and references the boolean of the imported WeaponController class
    //if the box collider component is hit, it sets off the trigger hit, triggering
    //the hit animation and the particle and after 3 seconds, it destroys the object

    private void OnTriggerEnter(Collider other)
    {
        if(other.tag == "Enemy" && wc.IsAttacking)
        {
            other.GetComponent<Animator>().SetTrigger("Hit");
            Instantiate(HitParticle, new Vector3(other.transform.position.x,
            transform.position.y, other.transform.position.z), other.transform.rotation);
            Destroy(other.gameObject, 3.0f);
        }
    }
}
```

Class/ Objects

SettingsMenu.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Audio;

//Copy of in-game value script

public class SettingsMenu : MonoBehaviour
{
    public AudioMixer audioMixer;

    public void SetVolume (float volume)
    {
        audioMixer.SetFloat("volume", volume);
    }
}
```

MainMenu.cs*

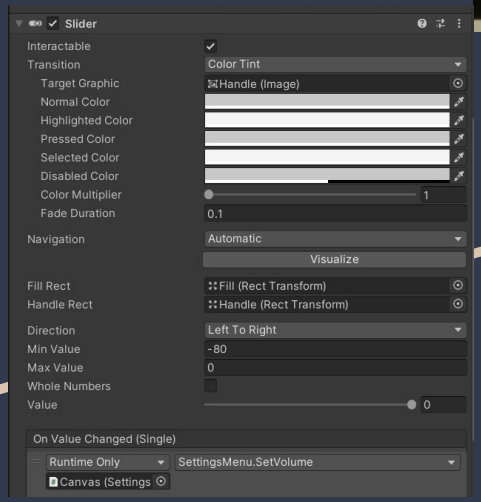
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class MainMenu : MonoBehaviour
{

    //public method that uses the index of the scenes in the build settings
    //and loads the next scene in the list
    public void PlayGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }

    //public method that can be used to quit the game
    public void QuitGame()
    {
        //will not close in unity editor so we use logs to show it quits
        Debug.Log("QUIT!");
        Application.Quit();
    }
}
```

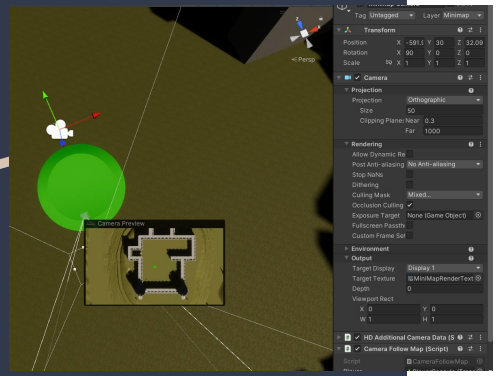
PauseMenu.cs*

```
using UnityEngine.SceneManagement;

public class PauseMenu : MonoBehaviour
{

    //made static to be used inside the game scene for game pause
    public static bool GameIsPaused = false;
    public GameObject pauseMenuUI;


    // Update is called once per frame
    //Uses the user events system to determine if escape key is pressed
    //and makes the canvas screen visible or non-visible
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if(GameIsPaused)
            {
                Resume();
            }
            else
            {
                Pause();
            }
        }

    //makes the canvas/menuscreen non-visible
    //time scale is just how fast the application
    //or realtime is e.g. 1f is 1 second meaning the game will move
    public void Resume()
    {
        pauseMenuUI.SetActive(false);
        Time.timeScale = 1f;
        GameIsPaused = false;
    }

    //makes the canvas/menuscreen non-visible
    void Pause()
    {
        pauseMenuUI.SetActive(true);
        Time.timeScale =  0f;
        GameIsPaused = true;
    }
```

```
//switches the scene to title screen with direct scene reference
public void LoadMenu()
{
    Time.timeScale = 1f;
    SceneManager.LoadScene("TitleScreen");
}

//quits the game
public void QuitGame()
{
    Debug.Log("Quitting game...");
    Application.Quit();
}
```

Special Features



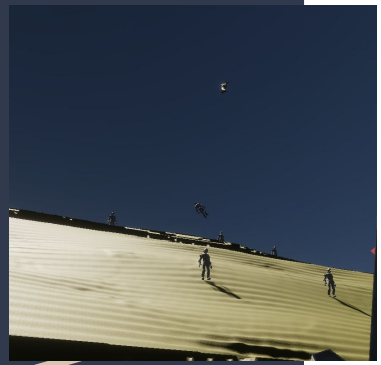


1. The minimap system is one of my biggest key features. I used the parent object's position e.g. the player location or enemy location and set a green or red icon at the position but a higher y-axis level to cover the model. The camera itself is positioned above the player so that the map will move with the player. I made a sprite image by referencing the minimap camera and adding it to the canvas, that's why it's able to update in real time and move around. On top of that the red dots (enemies) will be removed when the enemy is destroyed. I also set the icon image to only appear on the minimap camera layer so it doesn't appear when the player looks at enemies.

2. A second feature I added was the ability to change the volume of the game. By adding a slider that contains code, the main mixer is referenced and as the decibel levels are exposed to the code, the volume level updates. This allows the user to set the volume level to a comfortable position.

Known Bugs

Enemy hands not appearing when player gets close



Collider box touching the ground and sending enemies flying

Player being able to collide and push some enemies

Being able to go under the map through falling through the mountain spike area

The minimap not displaying enemy dots at the top of the mountain

What I want to Implement

- Enemy Movement
- Enemy Attack
- System that prevents player and enemy from moving upon collision with other objects
- Enemies respawning in waves
- Health pick up objects/regeneration
- More sound effects and music
- More UI e.g. health bar, enemy count, kill feed

Conclusion

Considering time spent and enjoyment, I think my project is decent, but it could of been better. I spent hours each day working on it using tutorials that I removed the script and objects of because it wouldn't work how I needed it. In the end, I mainly spent time testing tutorials and understanding it. Despite mainly following tutorials, I had to problem solve my own and move child objects to parent objects and reference difference objects in the scene to make the code work properly. It was a lot of problem-solving and even then, I spent hours following along and making the connections of hey, this is basically a getter and setter method or hey, this is referencing the object or class I created. In totality, the project was stressful, I spent a lot of time messing up, fixing code, understanding what I was doing and in general, everything took longer than I anticipated despite continuous hours of work almost every day. Despite being so stressful, I had a fun time with my project and am somewhat proud as it's my first time using unity.

What I Learned

Truthfully speaking, there was a lot to learn from this experience. Not only was it my first time using Unity, but I choose to use 3-dimensional as the base for my game which in my opinion was more difficult than 2-dimensional games. I had to learn how to import assets such as audio for the game and customize it in unity to fit requirements for the audio source script. I had to learn about different methods such as Ienumerator which functions like a while loop or for each loop, how to animate, how to add object components, and overall learn to code in C++. learning how to use C++ wasn't as difficult as I was able to make connections between different types of methods based on their function or because of the method type e.g. void.

Citation

All of my code and learning how to do import, animate, create came from parts of tutorials and implementing, re-arranging, and coding differently to fit my project's needs.

***Youtube*, Youtube** https://youtube.com/playlist?list=PLRV64V4dk5Y4nLhzD01wrv2P_Zc7qdeD-

Keep in mind that I still had to learn about each line of code while watching the tutorials, it's not about just following it and copying the code. I understood it so I could rearrange and problem solve which was a daily necessity while working on this project.

Questions?