Samantha Kusner and Pravin Khanal
CSCI 3907 Section 87
VR Project Final Report

Section 1:

Welcome to Dumbledore's Labyrinth, fellow wizard! Our VR app brings the maze from Harry Potter and the Goblet of Fire to life in a way that has yet to be seen before. Dumbledore knows us wizards have a lot to do to prepare for the upcoming Triwizard Tournament, which is a place where our magical skills will be put to the test, so he is doing his best to help us prepare. Dumbledore was clued in that the final event of the Tournament is a complicated labyrinth that only one wizard will be fortunate enough to solve, so he did us a favor and set up a practice maze just outside of Hogwarts grounds. Within this maze, we can challenge our magical skills in determining the quickest path to the end of the labyrinth where we will reach the Triwizard Cup and be victorious!

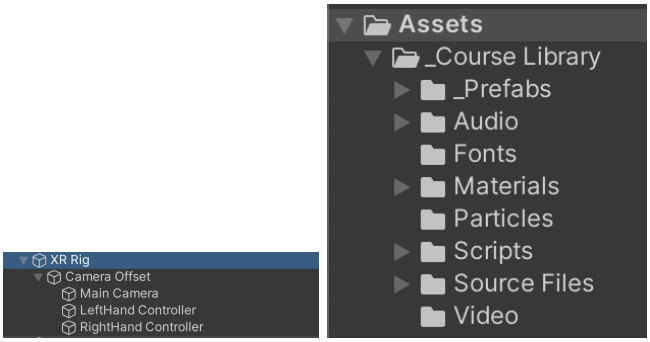
We developed this app in Unity version 2021.3.8f1 on a Mac OS for our Meta Quest 2.

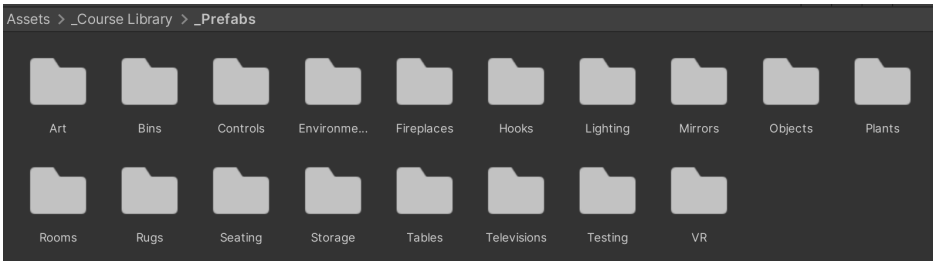
Section 2:

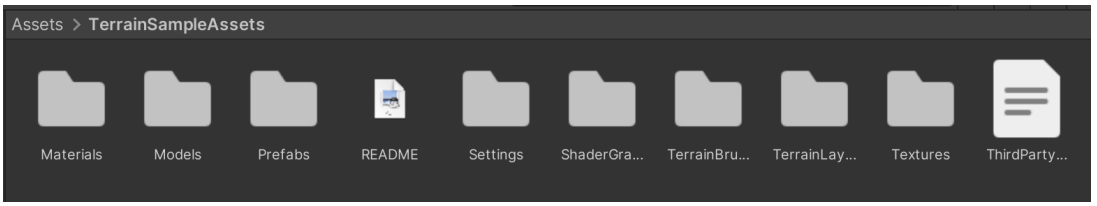
The main inspiration for our VR app was Harry Potter, more specifically, Harry Potter and the Goblet of Fire. We wanted to create a labyrinth that players could maneuver through to reach the end where there would be a goblet waiting for them. One major design that we heavily debated over was how the users would move through the terrain. We were initially thinking of having the players walk/run through the terrain, but we agreed that having them teleport through the terrain might be more fun. When it came to actually building the maze, we agreed to make the terrain more natural with mountains that have snow on the peaks. Along with this we wanted users to be able to see Hogwarts as they progress through the terrain, so we put Hogwarts on one of the mountain peaks. We also added instructions on the main menu so that users know what their objective is upon entering the game. We also incorporated 6th degree of freedom in our application so that users would be more comfortable in the environment. This significantly reduced feelings of nausea.

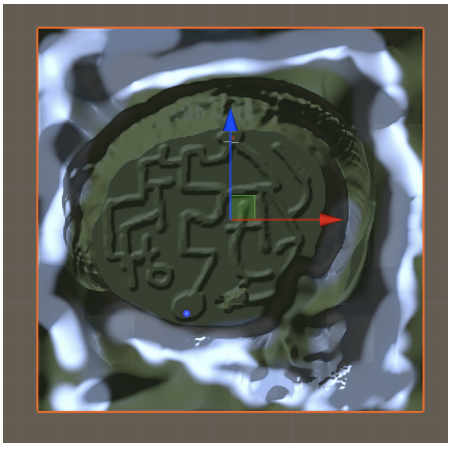
Section 3:

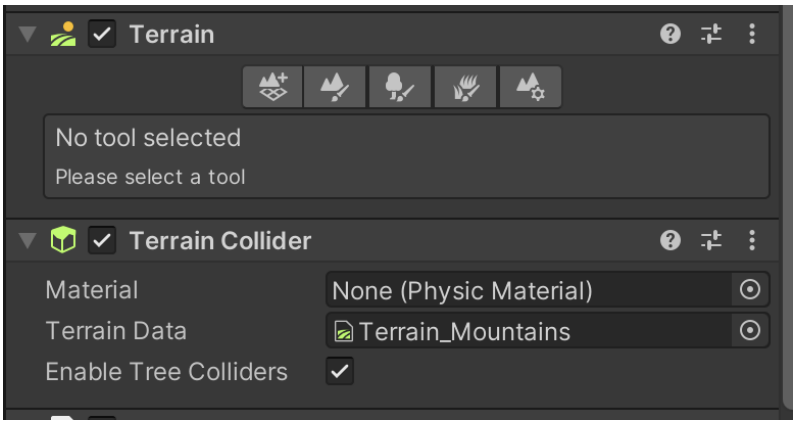
1. We used Unity's create with VR folder to start the project since it had all the VR and XR packages pre-installed as well as access to the XR Rig which played a crucial part in our project.
    a. https://learn.unity.com/course/create-with-vr - Link to the folder as well as the video guide
    b. https://pixabay.com/sound-effects/success-fanfare-trumpets-6185/ - Audio link

c. 

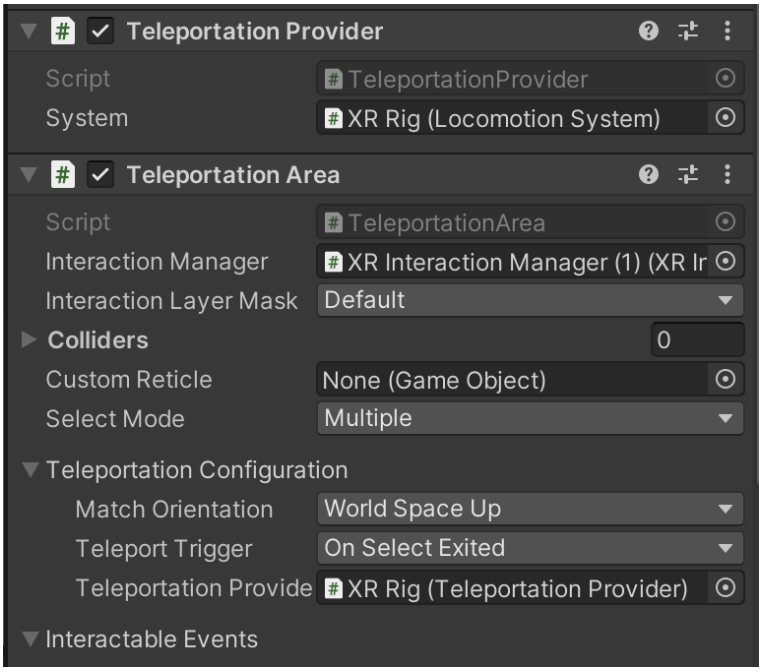


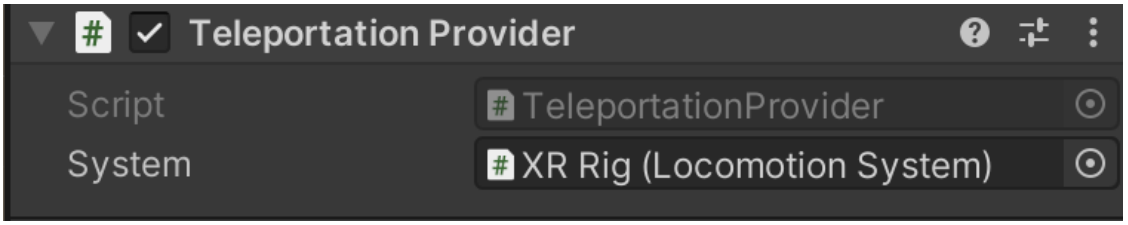
2. We built the terrain using Unity's terrain asset package, which can be found in the Unity asset store. The package gave us access to some textures and brushes which made building the terrain a whole lot easier. Along with this we also utilized YouTube videos, which provided a step-by-step instruction on how to create the terrain and add some textures.
    a. https://www.youtube.com/watch?v=MWQv2Bagwgk
    b. https://www.youtube.com/watch?v=2XdQkwSw-bA - We used this guide mainly
    c. https://assetstore.unity.com/packages/3d/environments/landscapes/terrain-sample-asset-pack-145808 - This is the terrain assets package
    d. 

    e. 

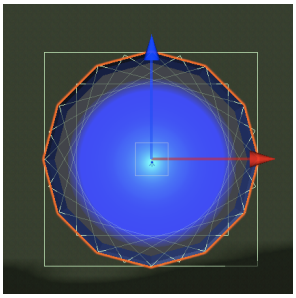
f. 

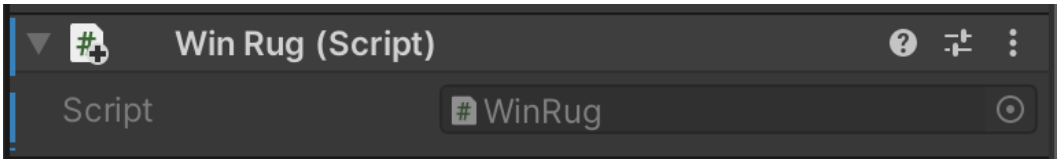
3. Next we implement locomotion to maneuver through the terrain. We followed Unity's locomotion tutorial that is provided in Unity's create with VR guide.
   a. https://learn.unity.com/course/create-with-vr

b. 



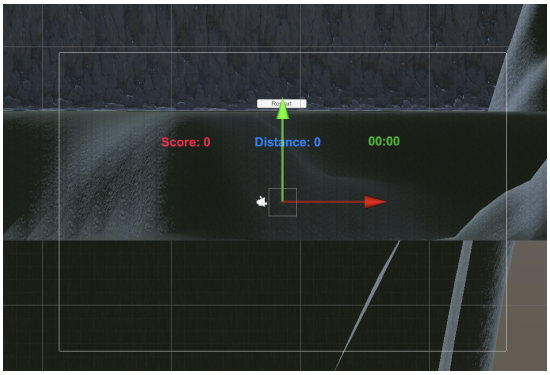
4. After this we started working on the navigation function. We created an object at the end of the maze and made that our target goal, which we then implemented into our script.
   a. https://www.cnblogs.com/kao-la-bao-bei/p/13503413.html

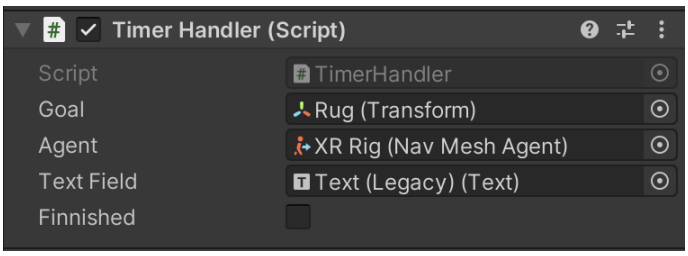
b. 

c. 

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

public class WinRug : MonoBehaviour
{
    // Start is called before the first frame update
    private void OnTriggerEnter(Collider other){
        GameObject.Find("Rug").SendMessage("Finnish");
    }
}

```

5. We also added a timer and distance calculator using guides found on YouTube and forums that included scripts.
    a. https://www.youtube.com/watch?v=Txtma2S7aNs
    b. https://docs.unity3d.com/ScriptReference/Vector3.Distance.html

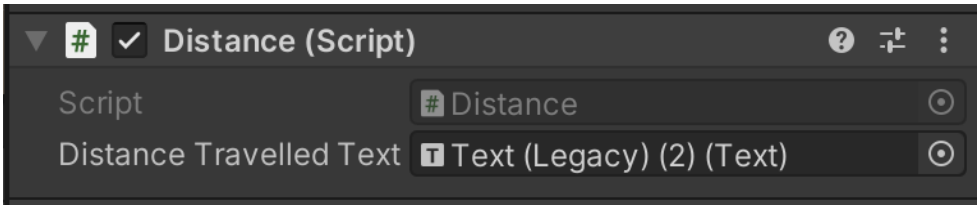
c. 

d. 

e.

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

public class TimerHandler : MonoBehaviour
{
    [SerializeField]
    private Transform goal;
    [SerializeField]
    private UnityEngine.AI.NavMeshAgent agent;
    // [SerializeField]
    // private Transform player;

    private float startTime;
    private string textTime;
    private float guiTime;

    private int minutes;
    private int seconds;
    private int fraction;

    public Text textField;

    public bool finnished = false;



    void startScene(){
        if(SceneManager.GetActiveScene().name.Equals("HogwartsRedone")){
            startTime = Time.timeSinceLevelLoad;
        }
    }

    void Update(){

        if(finnished) return;


        if(SceneManager.GetActiveScene().name.Equals("HogwartsRedone")){
            guiTime = Time.timeSinceLevelLoad - startTime;

            minutes= (int)guiTime / 60;
            seconds = (int)guiTime % 60;
            fraction = (int)(guiTime * 100) % 100;
            textTime = string.Format("{0:00}:{1:00}", minutes, seconds, fraction);

            if(finnished == true){
                return;
            }
            else{
                textField.text = textTime;
            }
        }
    }

    public void FinishTime()
    {
        textField.text = "Final: " + textTime;
        finnished = true;
        return;
    }


}

```

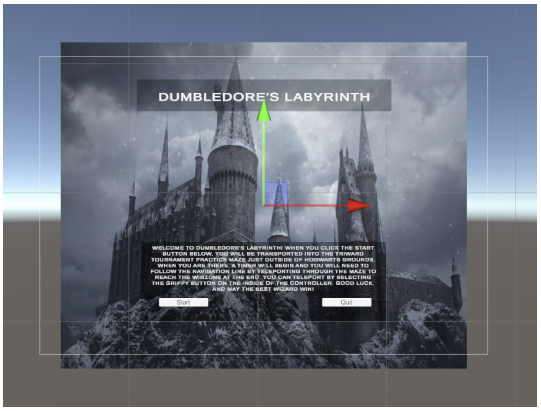
f.

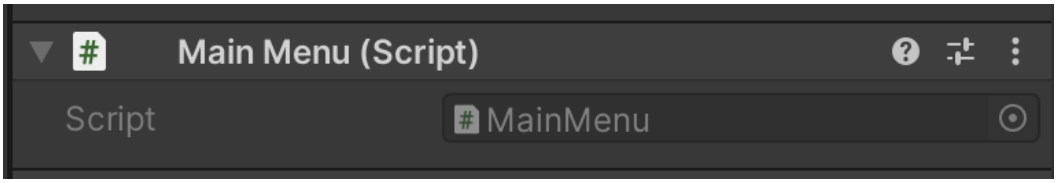


g.

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

public class Distance : MonoBehaviour
{
    float distanceTravelled = 0;
    Vector3 lastPosition;
    public Text distanceTravelledText;


    void Start()
        {
        if(SceneManager.GetActiveScene().name.Equals("HogwartsRedone"))
          lastPosition = transform.position;
        }


    public void UpdateDistance()
        {
            if(SceneManager.GetActiveScene().name.Equals("HogwartsRedone")){
                distanceTravelled += Vector3.Distance(transform.position, lastPosition);
                distanceTravelledText.text = string.Format("Distance: {0}", distanceTravelled.ToString());
                lastPosition = transform.position;
            }
        }
}

```

6. The last thing we implemented was a main menu. We did this by creating a new scene and adding a canvas with buttons that would either start the game or quit. Upon starting the game, it changes scenes to the game scene.

a. 

b. 

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

public class MainMenu : MonoBehaviour
{
    // Start is called before the first frame update
    public void ExitButton()
    {
        Application.Quit();
        Debug.Log("Game Closed");
    }

    // Update is called once per frame
    public void StartGame()
    {
        SceneManager.LoadScene("HogwartsRedone");
    }

    public void ReStartGame()
    {
        SceneManager.LoadScene("MainMenu");
    }
}
```

Section 4:

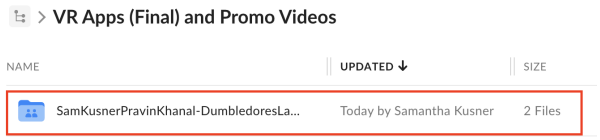
When creating this VR app we realized that the user experience would be one of our top priorities. Upon testing our app we realized that the teleportation feature moves too fast which was causing us to nauseate a bit. To fix this we slowed down the teleportation time so the users have a greater time to react to the teleportation. We also created the terrain so that there aren't sharp turns or any reason to make a quick 360 turn. We also created ample lighting with the terrain so that users have a clear view of where they are going at all times. Along with this the lighting never changes or flashes so users don't have to worry about photosensitive epilepsy. Another major optimization was having the scoreboard move with the camera so that players know their stats at all times, this means that players don't have to turn their head up, or to the

side in order to see the time or distance. One big optimization we have for users is having the restart button incorporated into the scoreboard, this allows users to restart the game at anypoint if they mess up. This also allows users who took the wrong path to reset the game and follow the correct path. The idea was that if a player gets lost, they should have to try and go till the end before resetting the game.

Section 5:

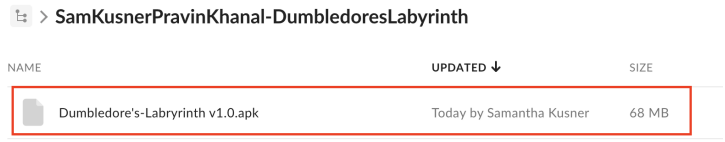
Step 1:

- From the Box folder, go to SamKusnerPravinKhanal-DumbledoresLabyrinth



Step 2:

- Once in our folder, download “Dumbledore’s-Labyrinth v1.0.apk” to your computer

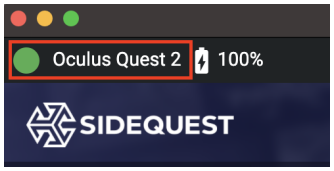


Step 3:

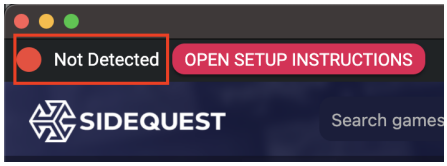
- Download SideQuest advanced installer from https://sidequestvr.com/setup-howto
- Download the appropriate version for your OS

Step 4:

- Open SideQuest
- Connect your VR headset to your computer
- Check that SideQuest recognizes your headset as connected
- If it is connected, you will see a green circle in the top left corner



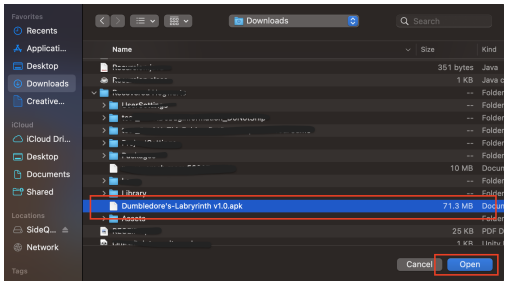
- If it is not connected, you will see a red dot in the same place
- Follow the setup instructions to connect your headset



Step 5:

- Locate the box with the arrow pointed down in the menu along the top of the screen

- From here, locate the apk you want to install to the headset and upload it

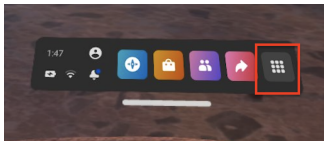


- Check the bottom of the screen for this message:

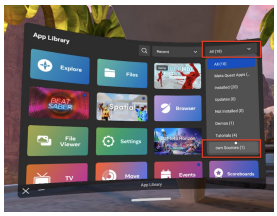
- If you receive this, you have installed the app successfully! If not, try again.

Step 6:

- Open your headset (this next section's instructions are specific to Meta Quest)

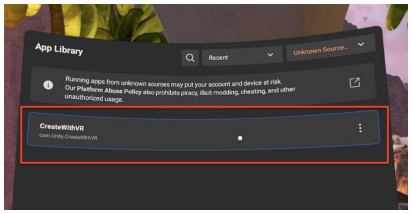


- Select the apps menu from the dock
- From here, click on the apps dropdown menu in the top right hand corner
- Select Unknown Sources from the bottom of this menu



Step 7:

- Once here, you should see the apk you installed
- Click on it to enter the application



Step 8:

- When you select the app, you should see a screen that say “made with unity” as it loads
- After that, you should see the main menu for the app
- Read the instructions
- Start the game with this button

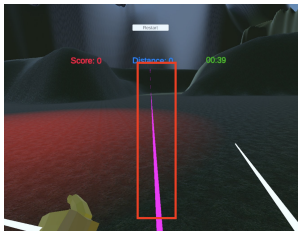
- Quit the game with this button

Step 9:

- When you enter the game, you will see your score, distance, and time statistics
- The timer begins when you enter the game, so don’t wait long to start moving!

Step 10:

- Follow the pink navigation line to help you reach the winzone at the end

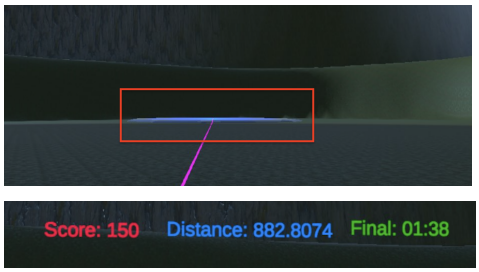


- Note: The line does not follow the navigation mesh of the terrain, so be aware that you cannot climb on walls to get to the end. When you move, the line will readjust and continue to lead you to the end.

- Teleport through the maze using the button on the inside of either controller
- Look around by moving the joystick

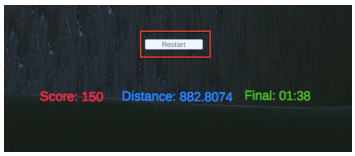
Step 11:

- When you reach the end, teleport within the blue winzone on the ground to score and get your end of game statistics!



Step 12:

- Go back to the main menu with the Restart button to start the game over or quit
- Note: you can do this at any point during the game!



Section 6:

Starting this project we had high expectations and aspirations of how we wanted the end product to be, however, we soon realized the challenges of using unity and how its functions are heavily dependent on each other. As far as what works, everything we implemented works to a certain degree. The terrain really worked in our favor, it was simple enough to create and there were many things you could customize. Adding texture and some patterns to the terrain was pretty simple upon following guides on YouTube. Locomotion was also another thing that was simple to implement since unity itself has a guide on implementing locomotion. Testing out different types of locomotion and customizing to fit the terrain was also hassle free. Making the scoreboard, timer and distance calculator was pretty simple, however, implementing code to

make them update was quite difficult. We ran into many issues whether it was with the code or with how unity handles its various objects. One thing we would like to see improvements on is our navigation pointer. Our guide has a straight path to the end goal that curves upwards into the terrain. We would like to have the directional guide on the ground at all times and curve through the terrain floor and not the terrains hills, which would align with our navigation mesh. Aside from that, everything seems to work pretty well.

Section 7:

Something that we would love to see implemented in this app further down the road is interactions between the player and obstacles in their path, which would be more true to the book which we adopted this idea from. Ideally, a player would encounter a magical creature or some other type of roadblock and they would have the option to choose between different spells to cast upon it. When the user is prompted to choose a spell, upon choosing a spell it would eliminate the enemy and give the users a shorter path to the end of the maze. We were also thinking of having the users have a magic hand that they could use to cast spells. The idea was to have a mana and health bar on the users and everytime they casted a spell it would reduce their mana but upon defeating the enemy the gauge would refill. A cool addition would be adding potions around the maze, like if you reached a deadend there would be potions there that would help the users later on. This would give users an opportunity to explore alternative paths and give diversity to the game. As far as the potions go, the potions would be able to unlock new spells, regenerate health or mana, or even potentially relocate the users closer to the end of the labyrinth. Some further improvements that could be made are, having the timer stop when the user reaches the end of the maze and to have the scoreboard update. Other further improvements would be adding more texture and details to the terrain and maybe even adding different lighting to different regions in the maze.

Appendix I: Lessons Learned:

When we began this project, we were extremely enthusiastic about it to the point where we may have accidentally lost sight of realistic expectations. It was very easy for us to dream up big and bold ideas for this project despite never having used the software before or knowing anything about game design. We are both juniors continuing our undergraduate education, and we were definitely a bit over ambitious in our initial design overview, which had us biting our tongues as days went by and we were unable to get past certain roadblocks along the way with Unity. That being said, we both found this project very rewarding and interesting when it was all said and done. During the development of our app, we had some very frustrating moments, such as when we accidentally deleted the whole scene we had been working on for weeks and had to scramble to try to find a way to get it back. At times, we doubted our skills entirely and had to take several steps back to reevaluate our progress. We also found ourselves pretty frequently overwhelmed, but we learned to take things one step at a time and rank the priority of our tasks based on functionality over aesthetic. We also learned how to utilize a lot of neat features within

Unity and the Unity community forum. We definitely found ourselves consulting Google as well as our classmates who we were friendly with to aid us when we hit challenging road blocks. It was really nice to see the way we came together to help each other overcome our challenges with the development environment. We do wish that we had more time to finish everything we had originally planned and hoped to create, but we are satisfied with the end product. We were both entirely new to Unity when we came into this class, so it took us a lot of time and effort to get to where we are. We are very proud of each other and the work we put in along the way despite the end product not being entirely what we envisioned when we began. We can confidently say at the end of this project that we appreciate the opportunity to expand our knowledge and skill set in this area, and hope to continue to grow as we move forward with school and our careers.

Appendix 2: Feedback to the instructor

Overall it was a great experience, and we learned a lot through the process. It was definitely a learning curve since unity was explicitly taught in class, but the product we produced paid off at the end. Professor Ok was very knowledgeable on the subject or AR/VR and its application in the real world, however, the class is designed to be very self paced when it comes to the projects. It would have been great to have more time in class to actually work on our projects, so that way we could have gotten some help debugging any errors we were having. It would also have been great if we had gotten more time to interact with some of Unity's tutorials so that we would have felt more comfortable going into the VR project. Showing some C# scripts and its interaction with the VR application. We would also have liked to have check-ins so that way we know we are on track, and if we weren't getting assistance so we could catch up.