

**INFO 5900**  
**SPECIAL PROBLEMS**  
**VIRTUAL REALITY AND ITS APPLICATIONS**  
  
**FINAL PROJECT**  
  
**THE MAZE MASTER**

Under the guidance of

**Dr. Sharad Sharma**

Team Members – 1. Sravani Reddy Meda (11593747)  
2. Pramodh Athota (11611995)  
3. Naga Sirisha Ponnaganti (11563226)

## CONTENTS

1. ABSTRACT	3
2. INTRODUCTION	4
2.1. GOALS OF THE PROJECT	4
2.2. MODELLING	4
2.3. APPLICATION USAGE	6
2.4. PROGRAMMING	6
3. FUNCTIONALITY	10
3.1 VISION	10
3.2 SOUND	12
3.3 ANIMATIONS	12
3.4 INTERACTIVITY	13
3.5 CHARACTERS/AVTARS	18
3.6 SENSORS	18
3.7 PLAYER	20
3.8 INTERFACE ELEMENTS	21
4. APPLICATION SIGNIFICANCE	24
5. WHY VR IS APPROPRIATE TECHNOLOGY?	24
6. PROBLEMS ENCOUNTERED	24
7. FUTURE WORK	25
8. SOFTWARES USED	26
9. ACKNOWLEDGEMENT	26
10. REFERENCES	26
11. CONCLUSION	26

**ABSTRACT:****Need for the Work –**

A VR experience that is both immersive and interactive was developed using Unity for the Maze Master project. The goal of this project is to give players a fun and interesting approach to go into a maze-like environment and solve puzzles.

**Importance of Maze Master –**

The Maze Master game has significance because it offers players an exciting and satisfying puzzle-solving experience that is both tough and engrossing. Players must use their imagination and critical thinking to find their way through the maze and solve the riddles in this visually stunning and intellectually demanding game.

**VR Environment of Maze Master –**

The VR environment of The Maze Master shows an intricate and challenging maze that users must navigate and solve puzzles within. The environment is designed to be visually appealing and immersive, with intricate details and challenging obstacles that require users to think creatively and strategically to progress.

**Target Audience for Maze Master –**

The target audience for Maze Master game are primarily individuals who enjoy puzzle-solving and are interested in exploring new and innovative virtual environments. This may include gamers, enthusiasts of virtual reality technology, and individuals seeking to improve their problem-solving and critical thinking skills. Additionally, the game may be appealing to individuals who enjoy challenging themselves and pushing their cognitive limits.

**Application of the project –**

The application is useful because it provides a fun and interactive way for individuals to improve their problem-solving skills while also engaging with cutting-edge technology. The Maze Master VR application offers a unique and innovative way to explore and interact with virtual environments, making it an exciting and valuable tool for education, entertainment, and personal growth.

## INTRODUCTION:

### Goals of the Project –

The goals of the Maze Master Virtual Reality project are:

- To provide an engaging and challenging game experience for players.
- To showcase the capabilities of virtual reality technology in creating immersive game environments.
- To improve the problem-solving and critical thinking skills of players through maze-solving challenges.
- To provide a customizable game experience through character selection and difficulty levels.
- To demonstrate the use of various tools and techniques in creating virtual environments, such as terrain tools, pro-builder tool, and asset store packages.
- To incorporate sound effects and visual cues to enhance the overall game experience.
- To create a user-friendly interface for navigation and interaction with the game.
- To provide an enjoyable and entertaining game experience for players of all ages.

### Modelling (Planned Geometry, Textures, Animations and Functionality)–

- The modelling for the Maze Master game was done using Unity's terrain tools, Pro-Builder, and assets from the Unity Asset Store.
- The Maze Master virtual environment consists of three different levels with different themes, namely a desert theme, a forest theme, and a city theme. The levels are designed as mazes, with walls and paths forming a complex structure that the player needs to navigate through.

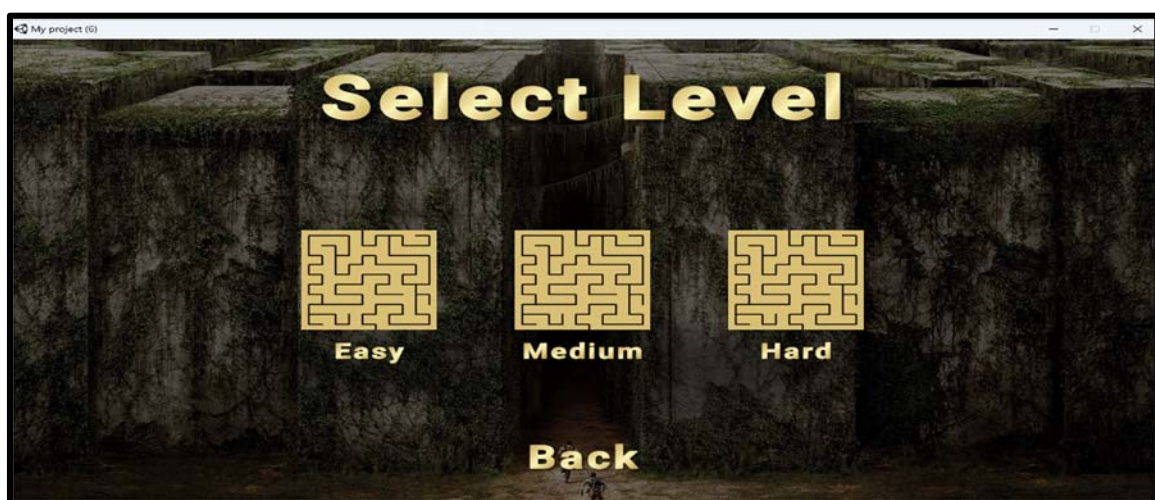


Fig1. Select the Level of Difficulty

- For the first level, the desert theme, the walls, and ground are created using the terrain tools in Unity, with a sand texture applied to the ground. The surrounding mountains are also created using the terrain tools. All textures used in the game are 4K textures from Blender and Poly haven. For the second level, the forest theme, a similar approach is used, but with a green forest texture applied to the ground and walls. Volumetric fog generated using particle generator plugins is also added to this level.

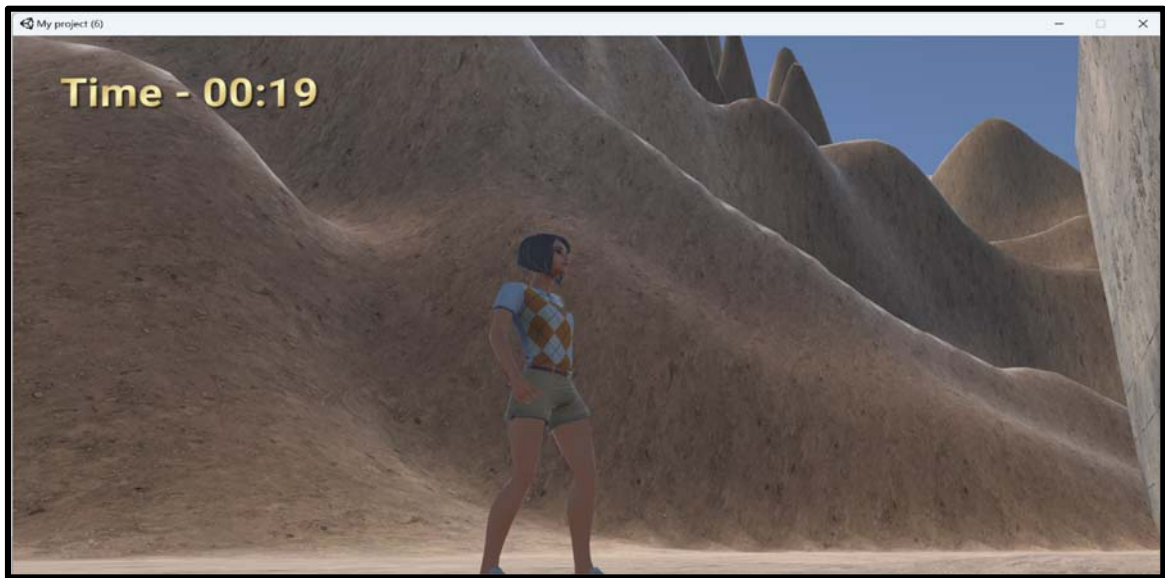


Fig2. Desert Theme

- For the third level, the city theme, the environment is designed using the White City package from the Unity Asset Store, which includes over 6000 buildings. A maze is then created in between the buildings for the player to navigate. The maze construction is done using the Pro-Builder tool in Unity.



Fig3. City Theme

- Animations are used for the player character, including walking, running, and idle animations. The player can also select between a male or female character avatar.
- The functionality of the environment includes the ability for the player to select the level difficulty and avatar, adjust the volume and sound effects, and access a map of the maze when they are stuck or lost. The player's progress is tracked with a timer for each level.

### **Application Usage –**

To start the game, the user will see a canvas scene with options to play, options, and exit. The user can select the play option and then select a character avatar (feminine or masculine). The user can also access the options menu to adjust the sound settings.

After selecting the avatar, the user can select one of three levels: Easy, Medium, or Hard. Each level will present the user with a different maze environment and difficulty level. The user will navigate through the maze using the VR controllers and try to find the way out.

If the user gets stuck in the maze, a help notification will automatically appear, indicating that the user can hold the E key to view the maze map. The map will be displayed in the form of an overlay and can be turned on or off by holding the E key or leaving the key.

The application also includes proximity sensors to prevent the user from walking through walls or buildings. If the user collides with a wall or building, they will be stopped from going any further in that direction.

At certain positions in the maze, the user will encounter sounds indicating footsteps and sprinting. The sounds of getting stuck or blocked will also be audible. If the user completes a level, a congratulatory message will appear on the screen, and the user can choose to go back to the level selection menu or quit the game.

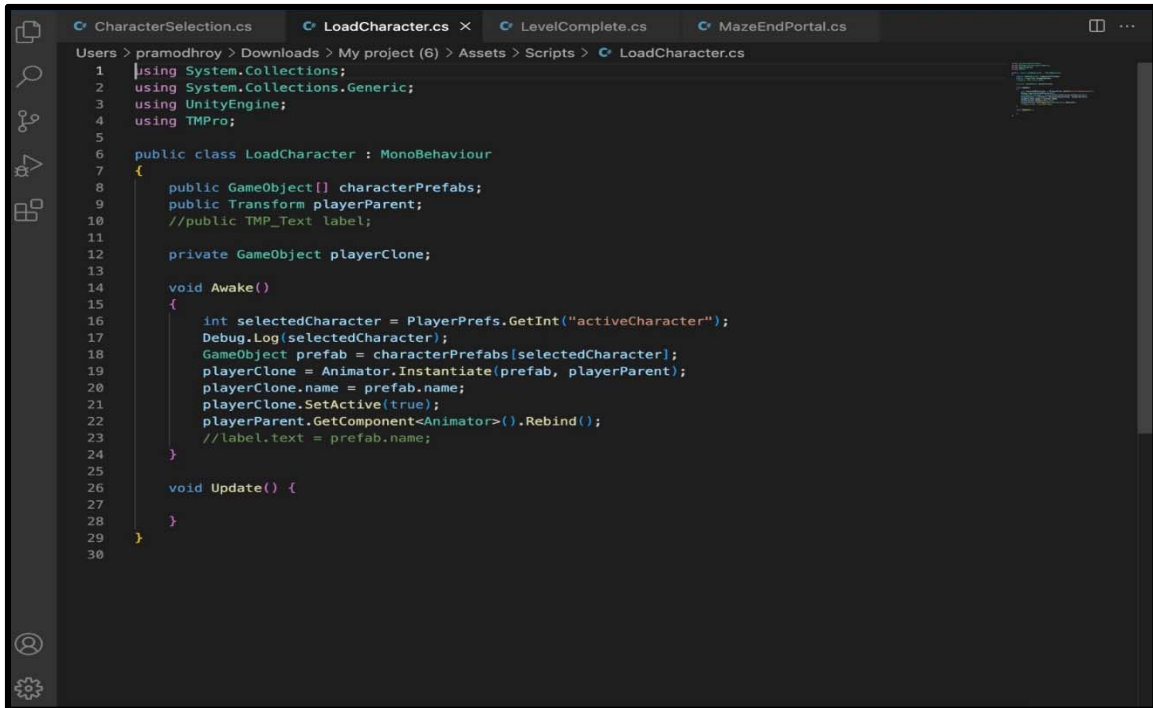
In summary, the Maze Master VR application will allow users to enjoy a virtual reality maze puzzle experience with various difficulty levels and features like map overlay, sound effects, and collision detection.

### **Programming –**

The game mechanics, user interface, and interactions were all programmed using C#.

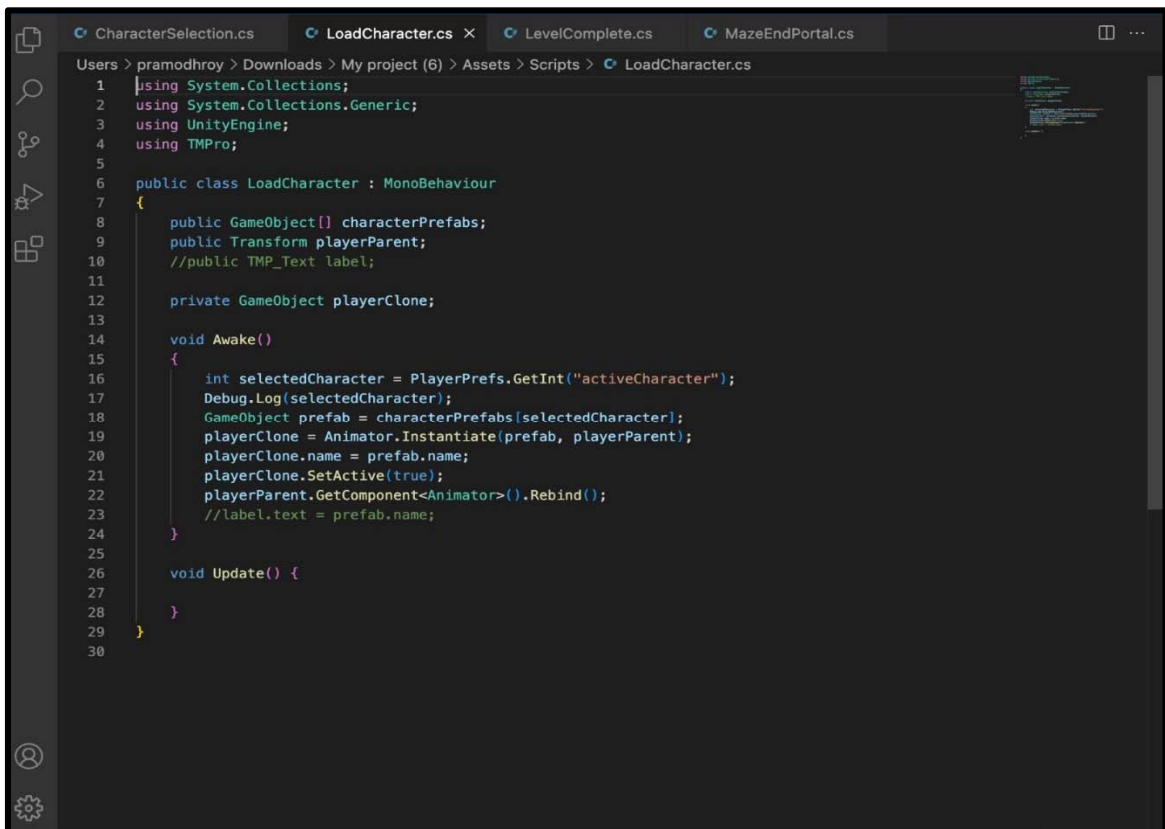
Some of the key programming features used in the game include:

- Third-person controller:  
The player movement and control of the avatar were programmed using the third-person controller feature in Unity. This allows the player to move the avatar using the keyboard keys and interact with the game world.



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using TMPro;
5
6 public class LoadCharacter : MonoBehaviour
7 {
8     public GameObject[] characterPrefabs;
9     public Transform playerParent;
10    //public TMP_Text label;
11
12    private GameObject playerClone;
13
14    void Awake()
15    {
16        int selectedCharacter = PlayerPrefs.GetInt("activeCharacter");
17        Debug.Log(selectedCharacter);
18        GameObject prefab = characterPrefabs[selectedCharacter];
19        playerClone = Animator.Instantiate(prefab, playerParent);
20        playerClone.name = prefab.name;
21        playerClone.SetActive(true);
22        playerParent.GetComponent<Animator>().Rebind();
23        //label.text = prefab.name;
24    }
25
26    void Update() {
27    }
28 }
29
30
```

Fig4. Character Selection Script



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using TMPro;
5
6 public class LoadCharacter : MonoBehaviour
7 {
8     public GameObject[] characterPrefabs;
9     public Transform playerParent;
10    //public TMP_Text label;
11
12    private GameObject playerClone;
13
14    void Awake()
15    {
16        int selectedCharacter = PlayerPrefs.GetInt("activeCharacter");
17        Debug.Log(selectedCharacter);
18        GameObject prefab = characterPrefabs[selectedCharacter];
19        playerClone = Animator.Instantiate(prefab, playerParent);
20        playerClone.name = prefab.name;
21        playerClone.SetActive(true);
22        playerParent.GetComponent<Animator>().Rebind();
23        //label.text = prefab.name;
24    }
25
26    void Update() {
27    }
28 }
29
30
```

Fig5. Load Character Script

➤ Canvas UI:

The canvas user interface was programmed using C#, including the options menu and the level completion screen. The canvas UI allows the player to interact with the game options and progress through the levels.

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using TMPro;
5
6  public class LevelComplete : MonoBehaviour //Unused Script
7  {
8      public TMP_Text contextLabel;
9      public TMP_Text scoresLabel;
10
11     void StartNot()
12     {
13         string completionTime = "0" + PlayerPrefs.GetString("completedLevelTime") + ":00";
14         string completionScore = PlayerPrefs.GetString("completedLevelScore") + "00";
15         scoresLabel.text = completionTime + "\n" + completionScore;
16         PlayerPrefs.SetInt("completedLevel", 0);
17     }
18
19     // Update is called once per frame
20     void Update()
21     {
22     }
23 }
24
25

```

Fig6. Level Completion Canvas Script

```

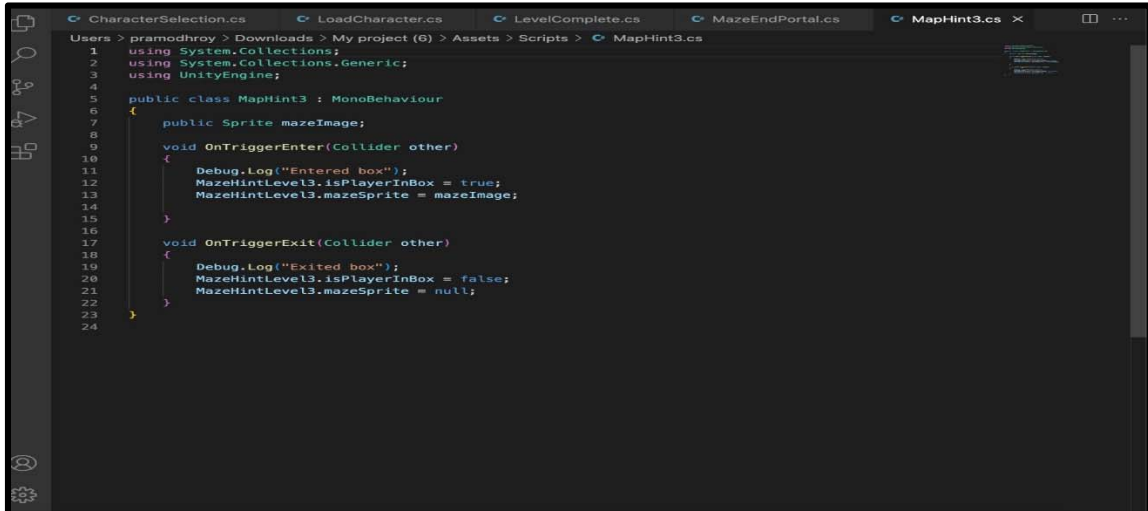
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5  using TMPro;
6
7  public class MenuScript : MonoBehaviour
8  {
9      public GameObject[] menus;
10     public TMP_Text levelCompleteContextLabel;
11     public TMP_Text scoresLabel;
12
13     void Awake()
14     {
15         Cursor.lockState = CursorLockMode.None;
16         foreach (GameObject menu in menus)
17         {
18             menu.SetActive(false);
19         }
20         if (!PlayerPrefs.HasKey("completedLevel"))
21         {
22             PlayerPrefs.SetInt("completedLevel", 0);
23         }
24         if (PlayerPrefs.GetInt("completedLevel") == 0)
25         {
26             menus[1].SetActive(true);
27             PlayerPrefs.SetInt("activeCharacter", 0);
28             int selectedCharacter = PlayerPrefs.GetInt("activeCharacter");
29         }
30         else
31         {
32             LevelCompleteMenu();
33         }
34     }
35
36     public void quitGame()
37     {
38         Debug.Log("QUIT!");
39         Application.Quit();
40     }
41 }

```

Fig7. Menu Canvas Script



- Hint and map display:  
When the player reaches a certain point in the maze, a hint is displayed on the screen instructing them to press the "E" key to display the map. This was programmed using C# to trigger the display of the map when the "E" key is pressed.

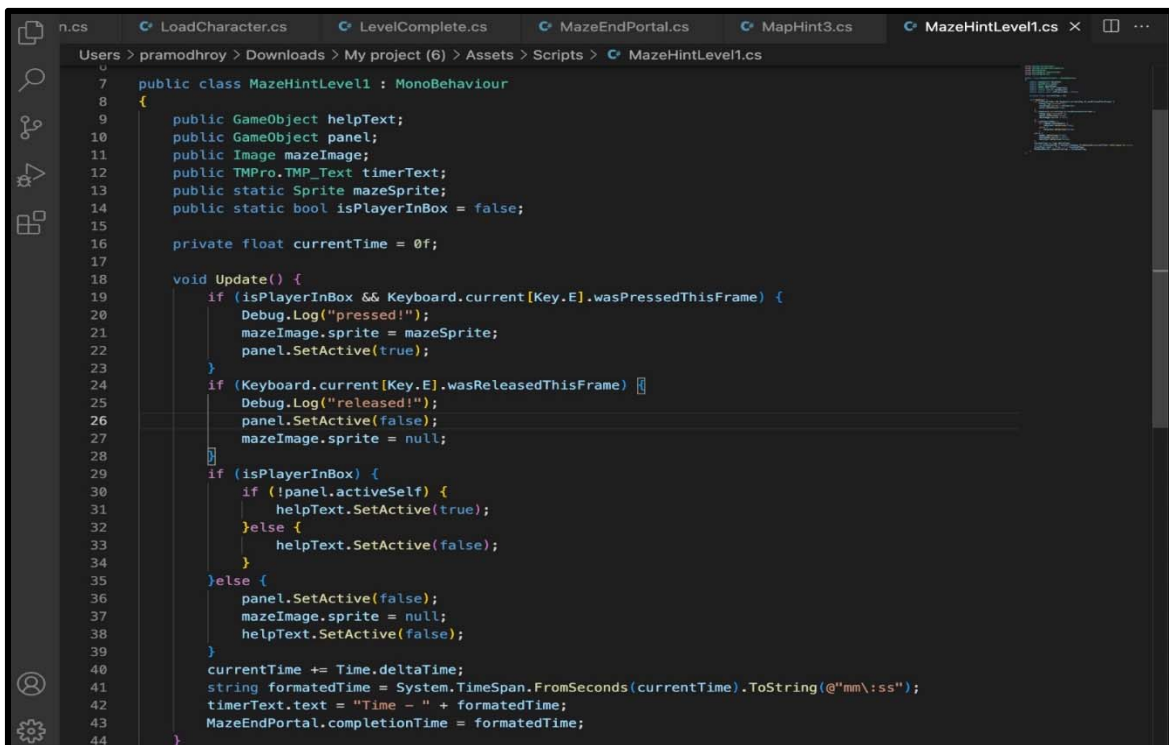


```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class MapHint3 : MonoBehaviour
6  {
7      public Sprite mazeImage;
8
9      void OnTriggerEnter(Collider other)
10     {
11         Debug.Log("Entered box");
12         MazeHintLevel3.isPlayerInBox = true;
13         MazeHintLevel3.mazeSprite = mazeImage;
14     }
15
16     void OnTriggerExit(Collider other)
17     {
18         Debug.Log("Exited box");
19         MazeHintLevel3.isPlayerInBox = false;
20         MazeHintLevel3.mazeSprite = null;
21     }
22 }
23
24

```

Fig8. Map Hint Script



```

7  public class MazeHintLevel1 : MonoBehaviour
8  {
9      public GameObject helpText;
10     public GameObject panel;
11     public Image mazeImage;
12     public TMPPro.TMP_Text timerText;
13     public static Sprite mazeSprite;
14     public static bool isPlayerInBox = false;
15
16     private float currentTime = 0f;
17
18     void Update() {
19         if (isPlayerInBox && Keyboard.current[Key.E].wasPressedThisFrame) {
20             Debug.Log("pressed!");
21             mazeImage.sprite = mazeSprite;
22             panel.SetActive(true);
23         }
24         if (Keyboard.current[Key.E].wasReleasedThisFrame) {
25             Debug.Log("released!");
26             panel.SetActive(false);
27             mazeImage.sprite = null;
28         }
29         if (isPlayerInBox) {
30             if (!panel.activeSelf) {
31                 helpText.SetActive(true);
32             } else {
33                 helpText.SetActive(false);
34             }
35         } else {
36             panel.SetActive(false);
37             mazeImage.sprite = null;
38             helpText.SetActive(false);
39         }
40         currentTime += Time.deltaTime;
41         string formattedTime = System.TimeSpan.FromSeconds(currentTime).ToString(@"mm\:ss");
42         timerText.text = "Time - " + formattedTime;
43         MazeEndPortal.completionTime = formattedTime;
44     }
45 }

```

Fig9. Maze Map Display for Key Trigger Script

Overall, C# programming was used extensively in the Maze Master game to create the game mechanics, user interface, and interactions.

## FUNCTIONALITY:

### Vision –

In the Maze Master game, textures and 3D models are used to provide a visually appealing and detailed virtual environment. The terrain tools in Unity are used to create the maze structure, and textures are applied to the walls and ground of the maze to create different environments for each level (e.g. desert sand texture for the Easy level, forest texture for the Medium level, and city texture for the Hard level).

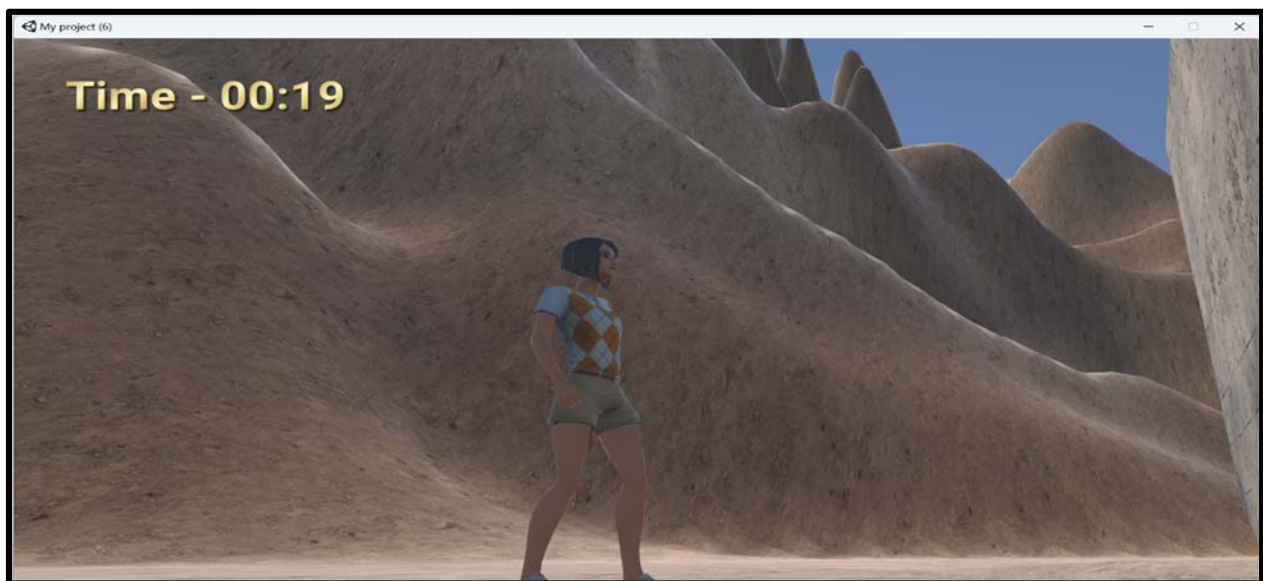


Fig10. Desert Theme for Level1



Fig11. Forest Theme for Level2

In addition to textures, 3D models are used to add details to the environment. For example, the mountain models in the Easy level are added using terrain tools, while the buildings in the Hard level are added using the White City package from the Unity Asset Store.

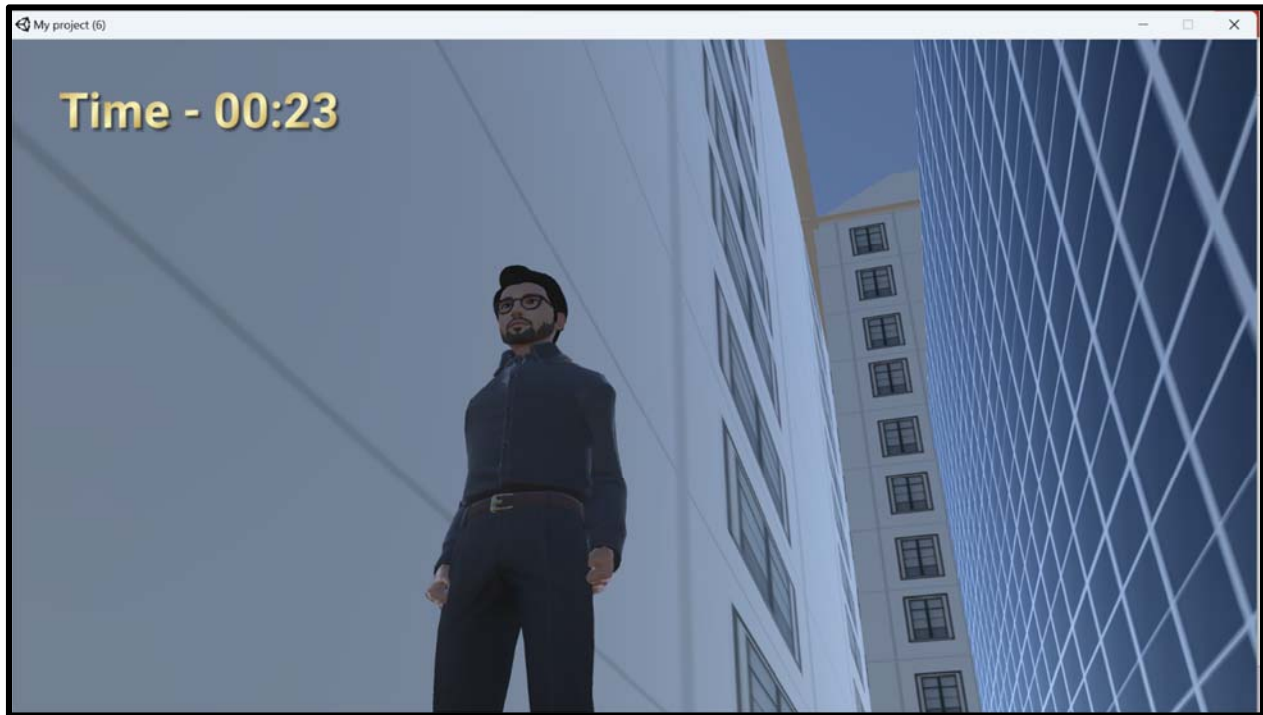


Fig12. City Theme for Level3

Furthermore, animations and particle effects are used to provide additional visual feedback to the player. For instance, volumetric fog is generated using particle generator plugins in the Medium level, and a help notification is displayed with an animation when the player reaches certain positions in the maze.



Fig13. Hint for Level 1

Overall, the use of textures, 3D models, and animations in the Maze Master game enhances the player's immersion in the virtual environment and provides detailed information about the game's surroundings.

### **Sound –**

Firstly, there are footstep sounds that play when the player moves around in the game. These sounds give the player feedback on their movement and make the game feel more immersive.

Secondly, there are sound effects that play when the player reaches certain points in the maze, such as when they are stuck and need help, or when they complete a level. These sounds provide important feedback to the player and help to keep them engaged in the game.

Finally, there is also background music that plays throughout the game. The music is designed to create a sense of tension and suspense, which helps to keep the player engaged and motivated to complete the maze.

### **Animations –**

The player animation is implemented to provide visual feedback to the player regarding their movement and actions in the game. The player avatar is rigged and animated. The player's avatar is animated to move forward, backward, left, or right based on the player's input. Additionally, the avatar is animated to perform other actions such as jumping, running, or climbing based on the game mechanics.

The fog animation is implemented to create an eerie and mysterious atmosphere in the game. The fog is created using a particle system in the game engine, which emits particles that simulate the appearance of fog. The fog's properties, such as density, color, and movement, can be adjusted to achieve the desired effect.



Fig14. Fog Animation

Both the player and the fog animations enhance the overall immersive experience of the game and make it more engaging for the player.

### Interactivity –

The Maze Master game has several user-triggered events that enhance interactivity and engagement. Here are five examples:

➤ **Avatar selection:**

When the player clicks on the "Play" button, a canvas scene appears with options to play, options, and exit. If the player clicks on "Play," it asks the player to select the avatar. The player can choose between a feminine or masculine avatar. This user-triggered event allows the player to customize the game experience according to their preference.



Fig15. Avatar Selection

➤ Level selection:

After selecting the avatar, the player can select the level of the game. There are three levels: Easy, Medium, and Hard. This user-triggered event allows the player to choose the level of difficulty they want to play.

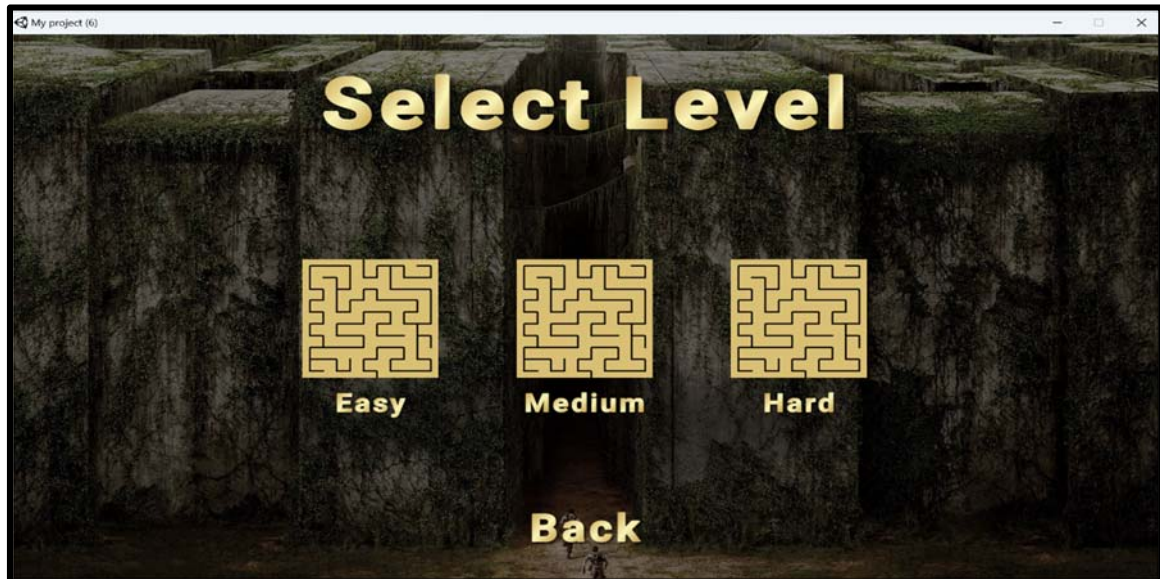


Fig16. Level Selection

➤ Help notification:

In all three levels, when the player reaches or gets stuck at certain positions, a help notification automatically appears, asking the player to hold the "E" key to observe the map. This user-triggered event provides assistance to the player when needed.



Fig17. Help Hint for Level2



Fig18. Help Hint for Level3

- **Maze map display:**  
When the player holds the "E" key at the specified locations, a map is displayed. The mesh renderer is used for the on and off toggle of the maze map. This user-triggered event allows the player to observe the map and navigate the maze effectively.

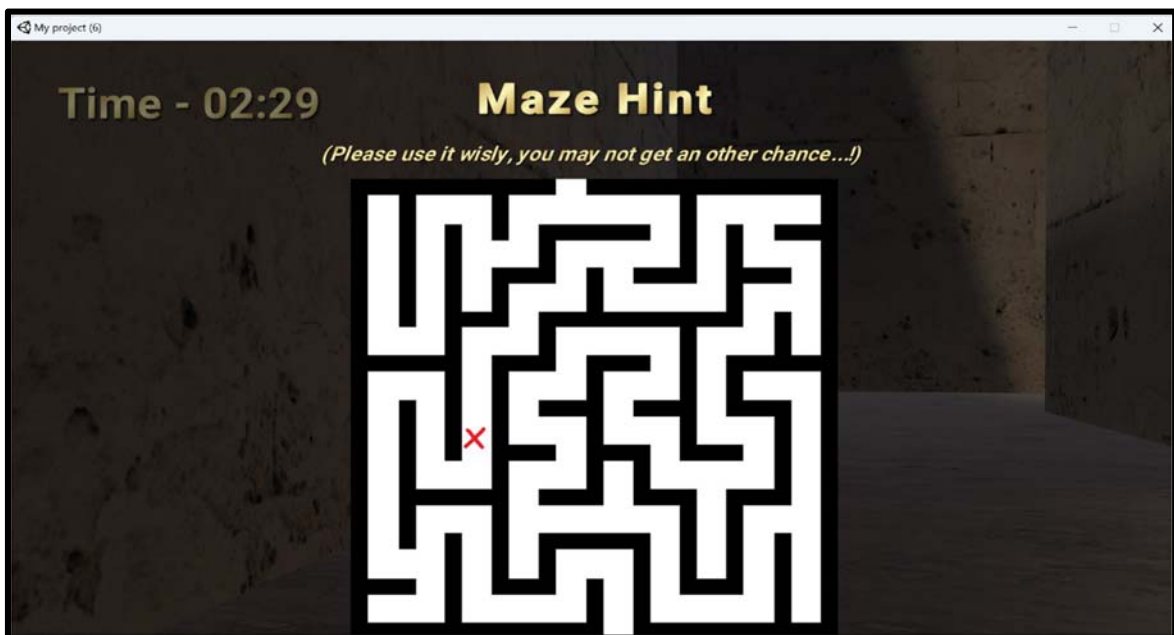


Fig19. Maze Map for Level1



Fig20. Maze Map for Level2

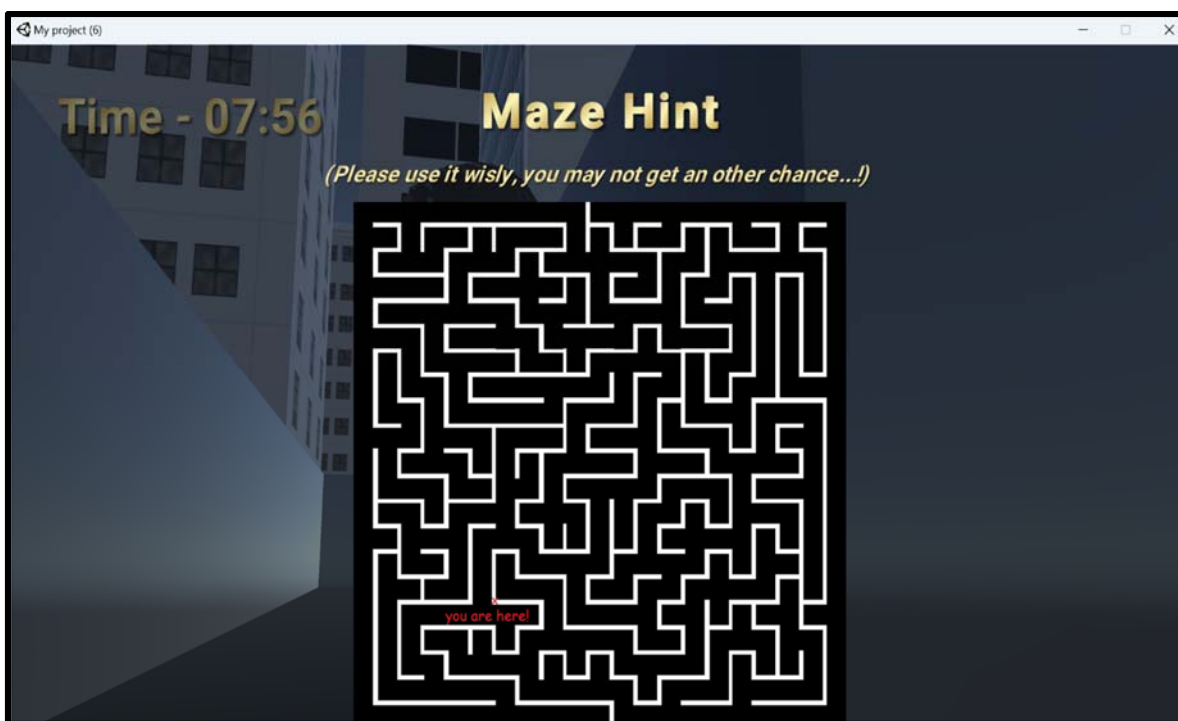


Fig20. Maze Map for Level3



➤ Teleportation:

After the player solves each level, a canvas UI appears automatically, mentioning congratulations and asking the player to click "Back" to select another level or "Quit" to end the game. This user-triggered event allows the player to navigate through the game and choose the next level or quit the game.



Fig21. Teleportation after completion of Easy Level



Fig. Teleportation after completion of Medium Level

Overall, these user-triggered events enhance the game's interactivity and provide a more engaging experience for the player. The code is written in C# for these events, such as teleportation, third-person controller, avatar selection, canvas, menu button, sliders, and displaying hints (Press E for the map).

### **Characters/Avatars –**

There are two options for the avatars: one is a feminine avatar, and the other is a masculine avatar.

Path-following behavior is used in game development to make the character move along a predetermined path. The avatars are programmed to move along the paths in the maze. The movement of the avatars is controlled using the keyboard. The player can use the W, A, S, D keys to move in four directions and the left shift key to sprint.



Fig22. Avtar Controls

The avatars are also interactive with the environment. For example, if the player collides with a wall or a building, the proximity sensor is triggered, and the avatar stops moving. Similarly, when the player reaches a specific location in the maze and presses the E key on the keyboard, the avatar stops moving, and a map is displayed on the screen. This interaction is programmed using C# code in Unity.

### **Sensors –**

In this project, there are three different types of sensors used: Proximity Sensor, Time Sensor, and Keyboard Input (which can be considered as Touch Sensor).

➤ **Proximity Sensor:**

This sensor is used to detect the player's distance from the walls, buildings or any other obstacle in the game. It is implemented using colliders. When the player collides with any obstacle, the proximity sensor detects it and prevents the player from going further. This ensures that the player does not pass through the walls or buildings and stays within the maze.



Fig23. Proximity Sensors

➤ Time Sensor:

Each level in the game has a timer that tracks the time taken by the player to complete the level. The time sensor is implemented using code in C# that records the starting time of the level and continuously updates the time taken by the player until the level is completed. The timer adds an element of challenge and competition to the game, motivating the player to complete the level as fast as possible.

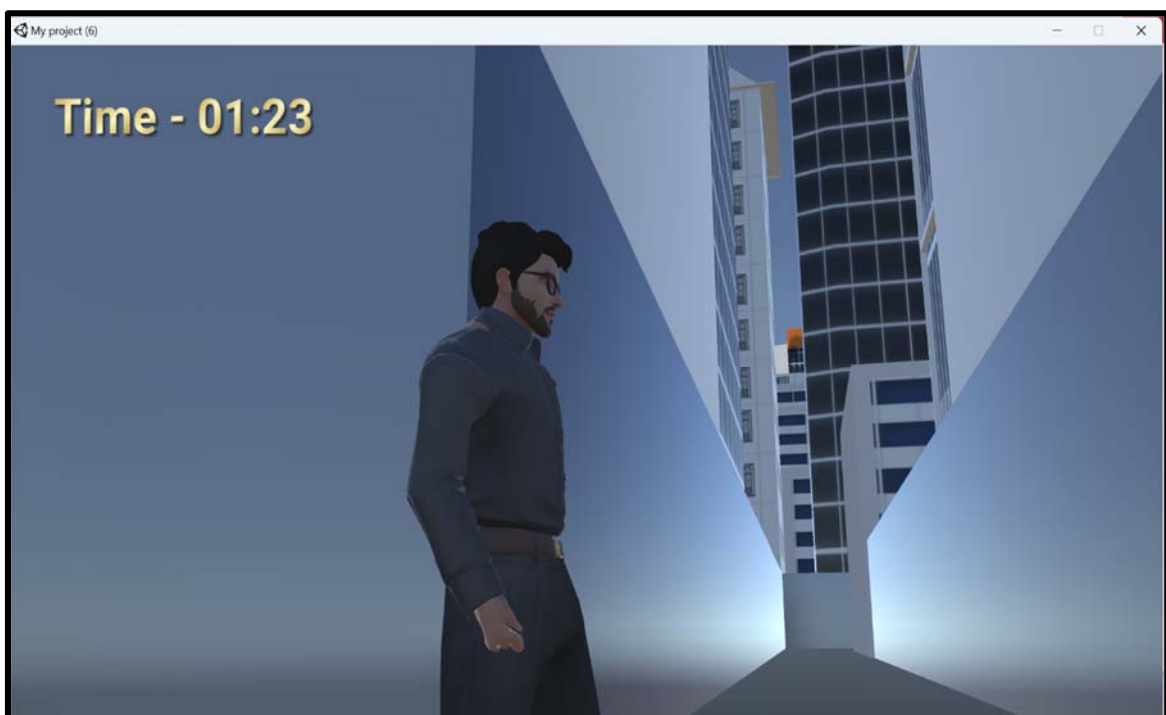


Fig24. Time Sensor

➤ Keyboard Input (Touch Sensor):

In the game, the player can trigger events by pressing and holding the E key on the keyboard. This is implemented using keyboard input, which can be considered as a touch sensor. When the player reaches a certain location in the maze, a hint is displayed on the screen, asking the player to hold the E key to display the map. This feature helps the player in navigating through the maze and finding the correct path to reach the end of the level. The keyboard input also allows the player to move the avatar in four directions and sprint using the left shift key.



Fig25. KeyBoard Input

### **Player –**

A Third Person Controller is used. This allows the player to see their character from a third-person perspective, which is appropriate for exploring the maze and solving puzzles. The player can control the character's movement using the W, A, S, and D keys on the keyboard, and the left shift key to sprint. The player can also select the character's gender based on their preference.



Fig26. Third Person Controller

### **Interface Elements –**

- **Main Menu:**  
The main menu appears when the game starts and includes three options - Play, Options, and Exit.

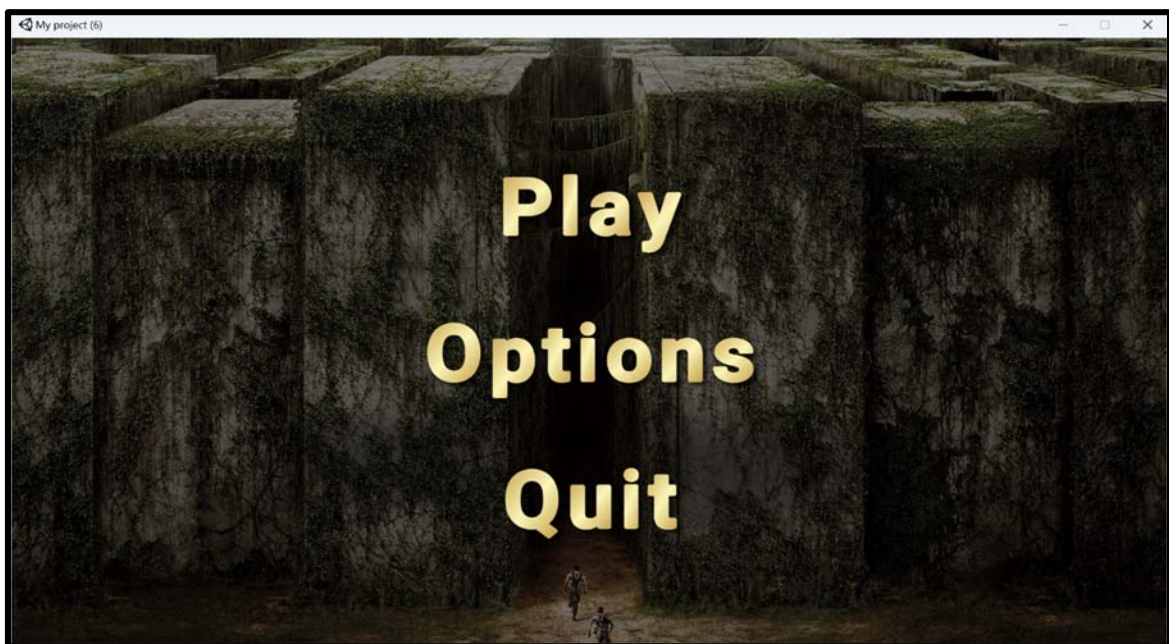


Fig27. Main Menu

- **Character Selection:**  
When the player clicks on the "Play" button, a character selection screen appears, allowing the player to choose their avatar. This screen includes two options - a feminine and a masculine avatar as shown in Fig15.
- **Options:**  
The "Options" button leads to a screen where the player can toggle the sound on and off, adjust the volume using a slider, and adjust the quality of graphics.

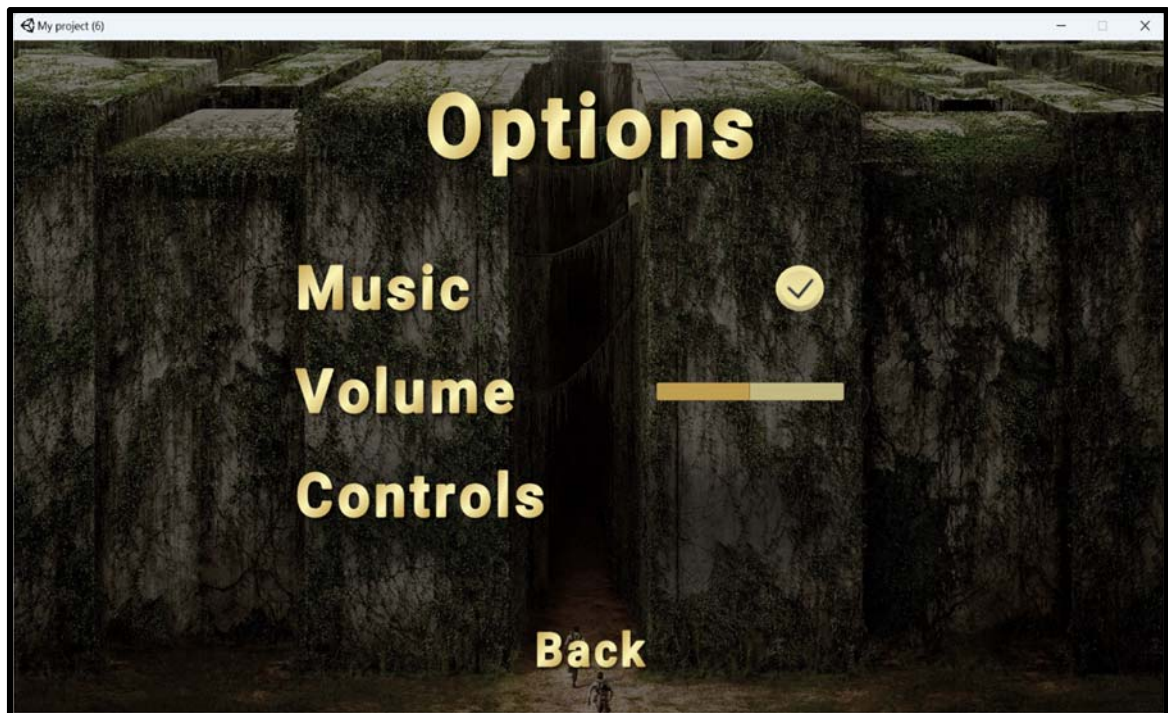


Fig29. Level Selection

- **Help Notification:**  
When the player reaches or gets stuck at certain positions in the maze, a help notification appears, instructing the player to hold the E key to observe the map.



Fig30. Help Notification

➤ **Maze Map:**

When the player holds the E key, a maze map appears, showing the layout of the maze.

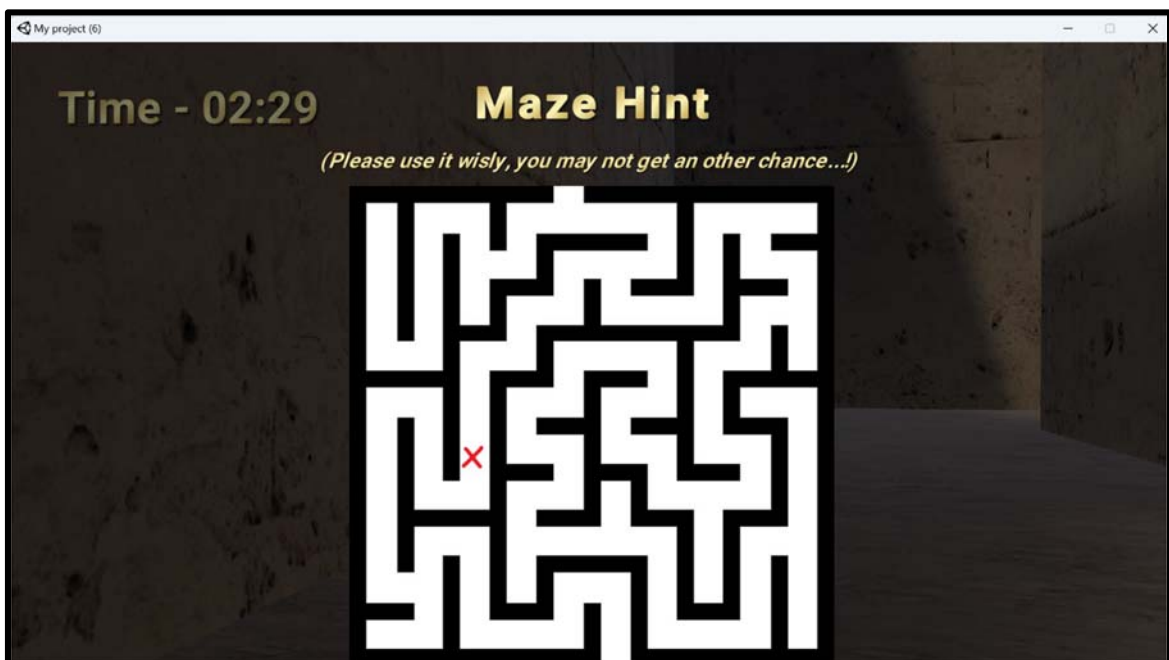


Fig31. Maze Map

➤ **Teleportation:**

After the player solves each level, a canvas UI appears, giving the option to go back to the level selection screen or quit the game.



Fig32. Teleportation to scene after completing Level1

- **Timer:**  
A timer runs for each level separately to calculate the time taken by the player to solve the maze.

**This application is useful** for providing an immersive and engaging experience to the players by allowing them to solve different levels of mazes with varying difficulties. It not only challenges their problem-solving skills but also provides an opportunity to explore different environments such as desert, forest, and a cityscape. The use of sound effects and visual cues enhances the overall experience and makes it more interactive. Additionally, the timer feature allows players to compete against themselves and try to solve the maze in the shortest possible time.

**Virtual reality is the appropriate technology for this project because** it provides a highly immersive and interactive environment for the players. By using a VR headset and controllers, players can move around and interact with the environment as if they are actually present in it. This level of immersion cannot be achieved with traditional 2D screens and input devices. Furthermore, VR technology allows for a high level of customization and interactivity, which is necessary for creating a complex and engaging maze game like this one. Overall, VR technology is a perfect fit for this project as it provides a unique and highly engaging experience to the players.

#### **PROBLEMS ENCOUNTERED:**

In the course of developing our maze game project, we encountered a few challenges and have identified some remaining shortcomings that we would like to report. One of the challenges we faced was the implementation of the torch feature. Initially, we planned to include a key that would enable the player to turn on the torch and navigate through the maze in the dark. However, we



encountered difficulties while adding the light and building that level, which led to a delay in implementing this feature. As a result, we were unable to include this feature in our final project.

Another challenge we encountered was the rendering of level 3, which took approximately 6 hours to complete. This significantly slowed down our development process and impacted our ability to focus on other areas of the project. If we had more time, we would have attempted to find a more efficient way to render this level.

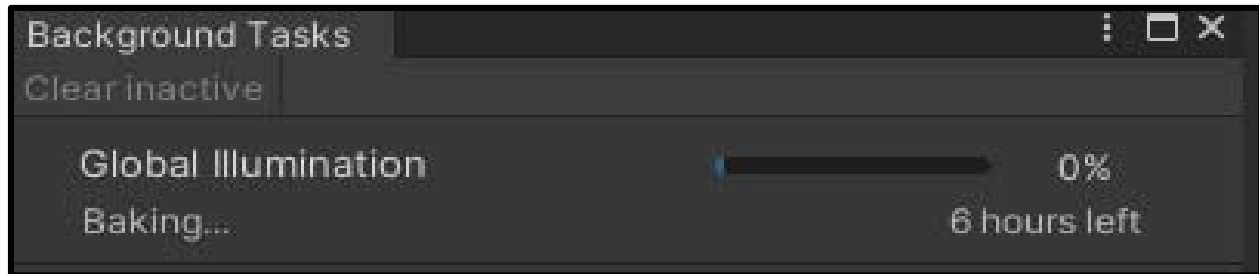


Fig33. Difficulty Faced

### **FUTURE WORK:**

The future work for this project could include several enhancements and improvements. Here are some possible ideas:

- **Multiplayer Mode:**  
Adding a multiplayer mode could make the game more engaging and challenging. Players could compete with each other to solve the maze in the shortest amount of time.
- **More Levels:**  
Adding more levels to the game could make it more interesting and challenging. The new levels could be designed with different textures and terrains to keep the players engaged.
- **Adding Opponent:**  
Adding an opponent that follows the player could make the game more challenging. The opponent could try to prevent the player from reaching the end of the maze, making it more difficult for the player.
- **Achievements and Rewards:**  
Adding achievements and rewards for completing levels within a certain amount of time could make the game more rewarding and enjoyable for players.

These are just a few ideas for future work on the project. Depending on the goals and resources, there could be many other features and improvements that could be added to the game.

**SOFTWARE USED:**

- Unity
- Adobe XD
- Photoshop

**ACKNOWLEDGMENT:**

- [White City | 3D Urban | Unity Asset Store](#)
- [https://polyhaven.com/a/coast\\_sand\\_rocks\\_02](https://polyhaven.com/a/coast_sand_rocks_02)
- [https://polyhaven.com/a/dirt\\_floor](https://polyhaven.com/a/dirt_floor)
- [https://polyhaven.com/a/aerial\\_rocks\\_02](https://polyhaven.com/a/aerial_rocks_02)
- [https://polyhaven.com/a/sandstone\\_blocks\\_04](https://polyhaven.com/a/sandstone_blocks_04)

**REFERENCES:**

- <https://helpx.adobe.com/in/xd/get-started.html>
- <https://www.mazegenerator.net/>
- [Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine](#)

**CONCLUSION:**

In conclusion, the project is a maze game with three levels of increasing difficulty. The game allows players to select their avatar, adjust sound settings, and navigate through different terrains. The project is developed using Unity and written in C#.