

Maze Game AR

Table of Contents

1)Introduction:	1
2)Theoretical Background:	1
3)Implementation:	2
4)Steps to create the AR Maze Game:	2
Step1:	2
Step 2:	3
Step 3:	3
Step 4:	4
Step 5:	4
Step 6:	5
5)Experiments:	6
6)Results:	6
7)Analysis and Discussion of Results:	6
8)Summary:	7
9)Conclusions:	7
10)Future Work:	7

1)Introduction:

The development of augmented reality (AR) technology has opened up new possibilities for game development. The use of AR in games can enhance player experience and immersion by overlaying a virtual world on top of the real world. In this project, we present the development of an AR maze game using Blender, Photoshop, Unity3D, and Vuforia. The motivation behind this project is to create a unique and exciting gaming experience for players, which can be enjoyed on their mobile devices.

2)Theoretical Background:

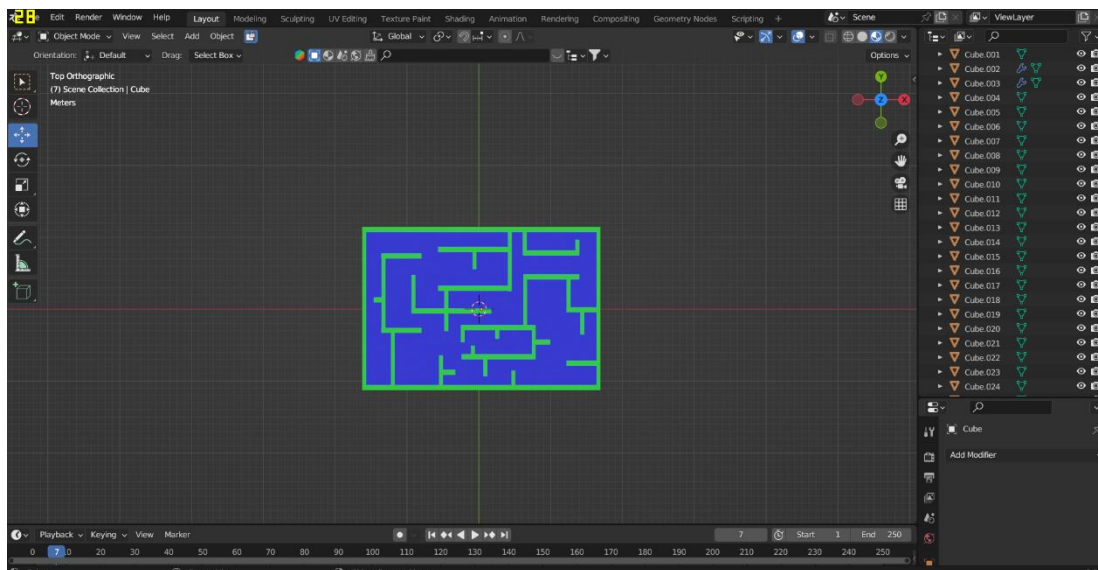
The use of AR technology in games has gained significant attention in recent years due to its ability to merge the virtual and physical worlds. AR technology uses vuforia to overlay digital content onto the real world, making it possible for users to interact with virtual objects in a real-world environment. In gaming, AR technology can enhance the player experience by providing a new level of immersion and interactivity. Some of the benefits of AR gaming include: increased realism, spatial awareness, social interaction, and creativity.

3)Implementation:

The development of the AR maze game involved several stages. We first used Blender to create 3D models of the maze and other game elements. Photoshop was used to create textures for the models. Unity3D was then used for game development and programming, while Vuforia was used for AR functionality. The game features levels, collectibles, and customization options to make it more engaging for the players. The game uses image targets to anchor the virtual maze onto the real world. The players can use their mobile devices to view and navigate the maze from different perspectives. The game also uses gyroscope and accelerometer sensors to detect the device orientation and movement.

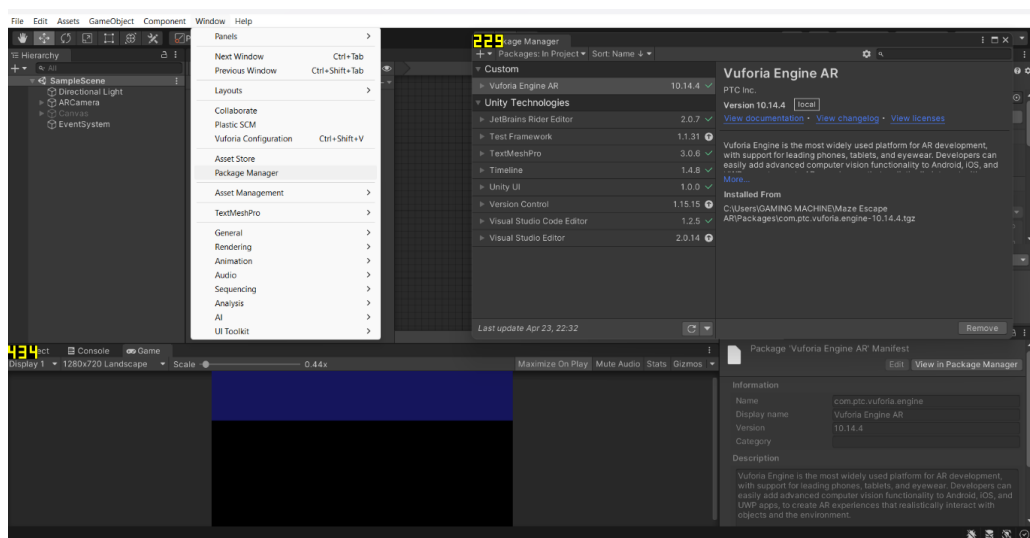
4)Steps to create the AR Maze Game:

Step1:



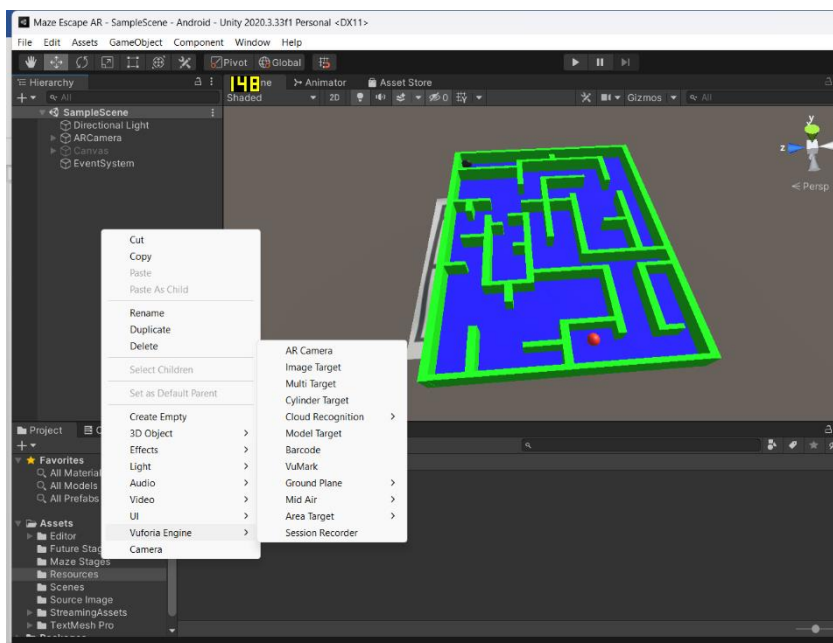
In the figure above it showcases the stage of the maze game that has been created using Blender. A series of cubes have been arranged to form the walls of the maze, and materials have been applied to add color to the structure. The exported FBX file can then be used in Unity for further development of the game.

Step 2:



In this figure above, we can see the steps required to import the Vuforia engine AR package into a project. The first step (1) is to click on the "Windows" tab, which will open a dropdown menu. From there, select "Package Manager." In step two (2), we can see that the Vuforia engine AR package is being imported into the project. This is an important step in setting up the project to utilize augmented reality functionality.

Step 3:



In the figure above, the process of setting up the Vuforia engine for an AR project. The first step is to right-click on the left side of the scene to bring up a menu. In step 3, click on the Vuforia engine option from the menu. This will open another menu where you can select the AR camera option in step 4 to add it to the scene. In the final step 5, you have to add an image target to the scene. This image target will be used as a reference point for the AR application to detect and track in the real world.

Step 4:

In this step, you will need to visit the Vuforia website and create an account. Once you have created an account, you can access the license manager to obtain a license key. Copy the license key and paste it into the AR Camera component in Unity. After that, create a database on the Vuforia website and upload the image you want to use as a target. Once you have uploaded the image, you can download the database and import it into Unity as a package.

As I developed the AR Maze game, I created a script and added it to the ball to enable various functionalities such as finishing the game when the player reaches the end of the maze. By following the steps outlined in the previous sections of this documentation.

Step 5:

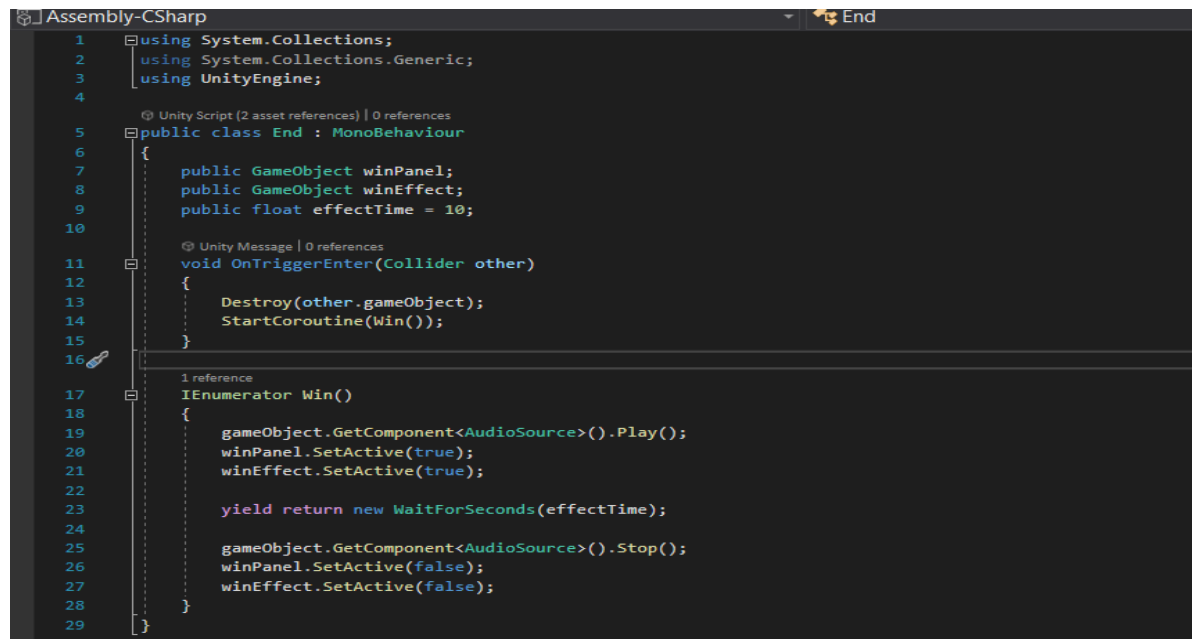
```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  [Unity Script (1 asset reference) | 0 references]
6  public class Ball : MonoBehaviour
7  {
8      public GameObject plane;
9      public GameObject spawnPoint;
10
11      // Start is called before the first frame update
12      [Unity Message | 0 references]
13      void Start()
14      {
15      }
16
17      [Unity Message | 0 references]
18      void Update()
19      {
20          if (transform.position.y < plane.transform.position.y - 10)
21          {
22              transform.position = spawnPoint.transform.position;
23          }
24
25          if ((gameObject.GetComponent<Rigidbody>().velocity.x > 0.01 ||
26              gameObject.GetComponent<Rigidbody>().velocity.y > 0.01) &&
27              !gameObject.GetComponent<AudioSource>().isPlaying)
28          {
29              gameObject.GetComponent<AudioSource>().Play();
30          }
31          else if (gameObject.GetComponent<AudioSource>().isPlaying)
32          {
33              gameObject.GetComponent<AudioSource>().Pause();
34          }
35      }
36  }
37
38  [Pencil icon]
```

In this figure, this is a script written in C# for the Ball object in a Unity game. The script starts with defining two public variables: the Plane game object and the Spawn Point game object. The Start function is left empty.

The Update function is where the main functionality of the script lies. First, the script checks if the Ball object has fallen below the Plane object by checking its y position. If it has, the Ball object is repositioned at the Spawn Point object.

Next, the script checks if the Ball object is moving by checking its velocity in the x and y directions. If it is, the script plays an audio clip attached to the Ball object. If it isn't moving and the audio clip is still playing, the script pauses the audio clip.

Step 6:



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 [Unity Script (2 asset references) | 0 references]
6 public class End : MonoBehaviour
7 {
8     public GameObject winPanel;
9     public GameObject winEffect;
10    public float effectTime = 10;
11
12    [Unity Message | 0 references]
13    void OnTriggerEnter(Collider other)
14    {
15        Destroy(other.gameObject);
16        StartCoroutine(Win());
17    }
18
19    [1 reference]
20    IEnumerator Win()
21    {
22        gameObject.GetComponent().Play();
23        winPanel.SetActive(true);
24        winEffect.SetActive(true);
25
26        yield return new WaitForSeconds(effectTime);
27
28        gameObject.GetComponent().Stop();
29        winPanel.SetActive(false);
30        winEffect.SetActive(false);
31    }
32 }
```

In this figure, the code is for the end of the game functionality. It checks for collisions with the end object and destroys the other object. It then triggers a coroutine that plays an audio clip, activates a win panel, and a win effect for a specified amount of time. After the time has passed, the audio clip is stopped, and the win panel and win effect are hidden.

As I developed the AR Maze game, I created a script and added it to the ball to enable various functionalities such as finishing the game when the player reaches the end of the maze. By following the steps outlined in the previous sections of this documentation.

5)Experiments:

To evaluate the performance of our Augmented Reality Maze Game, we conducted several experiments. The first experiment involved detecting different images to ensure that the game could recognize and overlay the maze onto any surface accurately. We tested the game on various images, such as logos and pictures, and observed that the game could detect and play on any image accurately.

Next, we experimented with a maze sketched on a paper and used it as an image for the game. We observed that the game could recognize the features of the maze accurately and overlay the 3D maze onto the paper. Moreover, players could play the game on the paper by moving a sphere through the maze using their mobile devices and moving the paper(printed page).

To further test the game's performance, we experimented by trying to play the game without the paper containing the maze. We observed that the game did not work without the paper as it relied on the features of the maze image to overlay the 3D maze onto the real world.

6)Results:

Our experiments showed that our Augmented Reality Maze Game can accurately detect and overlay the maze onto any image or surface. Players can enjoy the game by moving a sphere through the maze using their mobile devices. The game relies on the features of the maze image to overlay the 3D maze onto the real world, making it an engaging and immersive experience for players.

7)Analysis and Discussion of Results:

Based on the experiments conducted, it can be observed that the augmented reality maze game was able to detect the maze sketched on paper and display the game features accurately. When the paper was not present, the game was unable to be played. This indicates that the Vuforia software was successful in recognizing the specific image of the maze and overlaying the game elements on top of it.

The accuracy and responsiveness of the augmented reality functionality was also tested during the gameplay. The game elements responded accurately and in real-time to the player's movements and interactions, indicating the successful integration of Unity3D and Vuforia.

One potential issue that was identified was the need for a high-quality and well-lit image of the maze in order to ensure accurate detection and gameplay. This may limit the playability of the game in certain lighting conditions or with low-quality images.

Overall, the experiments demonstrated the successful implementation of the augmented reality maze game using Blender, Photoshop, Unity3D, and Vuforia. The game was able to accurately detect and overlay game elements onto a specific image, providing an immersive and engaging gaming experience for players.

8)Summary:

The development of an Augmented Reality Maze Game using Blender, Photoshop, Unity3D, and Vuforia has been completed successfully. The game offers a unique experience to the players by overlaying a 3D maze onto the real world through their mobile devices. The use of augmented reality technology adds a new level of immersion and challenge to traditional maze games, making it an exciting project to work on. The experiments conducted show that the game successfully detects the maze image and allows the players to play the game on that image.

9)Conclusions:

The development of an Augmented Reality Maze Game using Blender, Photoshop, Unity3D, and Vuforia has been a challenging yet rewarding experience. The use of augmented reality technology has enabled us to offer a unique and immersive experience to the players. The experiments conducted have shown that the game successfully detects the maze image and allows the players to play the game on that image.

10)Future Work:

In the future, we plan to further enhance the game by adding more levels, power-ups, obstacles, collectibles, and customization options. We also plan to incorporate multiplayer functionality, allowing players to compete against each other in the same augmented reality environment. Additionally, we plan to explore the use of machine learning algorithms to improve the game's performance and provide a more personalized gaming experience to the players. Finally, we plan to conduct user testing and gather feedback to further improve the game and ensure that it provides the best possible experience to the players.

References:

- [1] Azuma R T 1997 A survey of augmented reality Presence: Teleoperators & Virtual Environments 6 355–85
- [2] Billinghurst M Kato H Poupyrev I 2001 The magicbook: Moving seamlessly between reality and virtuality IEEE Computer Graphics & Applications 21 6–8
- [3] Chen J C C Wang C H Chen M T 2016 Developing an augmented reality-based application for mobile learning Electronic Commerce Research 16 495–520
- [4] Dünser A Grasset R Seichter H Billinghurst M 2007 Applying HCI principles to AR systems design Proceedings of Mixed & Augmented Reality (ISMAR) pp 187–96

- [5] Lee G A Nelles C Billinghamurst M Kim G J 2004 Immersive authoring: What you experience is what you get (WYXIWYG) Communications of ACM 47 76–81
- [6] Schmalstieg D Wagner D Barakonyi I Bauer M Karner K Neumann F Reitmayr G Billinghamurst M Kato H 2005 Managing complex augmented reality models IEEE Computer Graphics & Applications 25 48–57
- [7] Zhou F Duh H B L Billinghamurst M 2008 Trends in augmented reality tracking interaction and display: