NAVRACHANA
UNIVERSITY

AR MAZE PROJECT

Group:

Hamza Saiyed (22000229)
Huzaifa Patel (22000260)

Project description:

This Augmented Reality (AR) Maze Game utilizes image recognition technology to project a 3D maze into the real-world environment upon scanning a designated image marker. The game integrates realistic physics to simulate a virtual ball, which users can control to navigate through the maze by tilting their device. The AR experience combines interactive gameplay with real-world immersion, allowing players to engage directly with virtual elements in their physical surroundings. Developed using Unity and C#, this project demonstrates the effective fusion of AR technology with interactive gaming to create an immersive, real-time experience.

System Requirements:

Hardware: Mobile device with a minimum quad-core processor and at least 3 GB of RAM for smooth AR performance.

Operating System: Android 8.0+ or iOS 12.0+ with support for ARCore or ARKit.

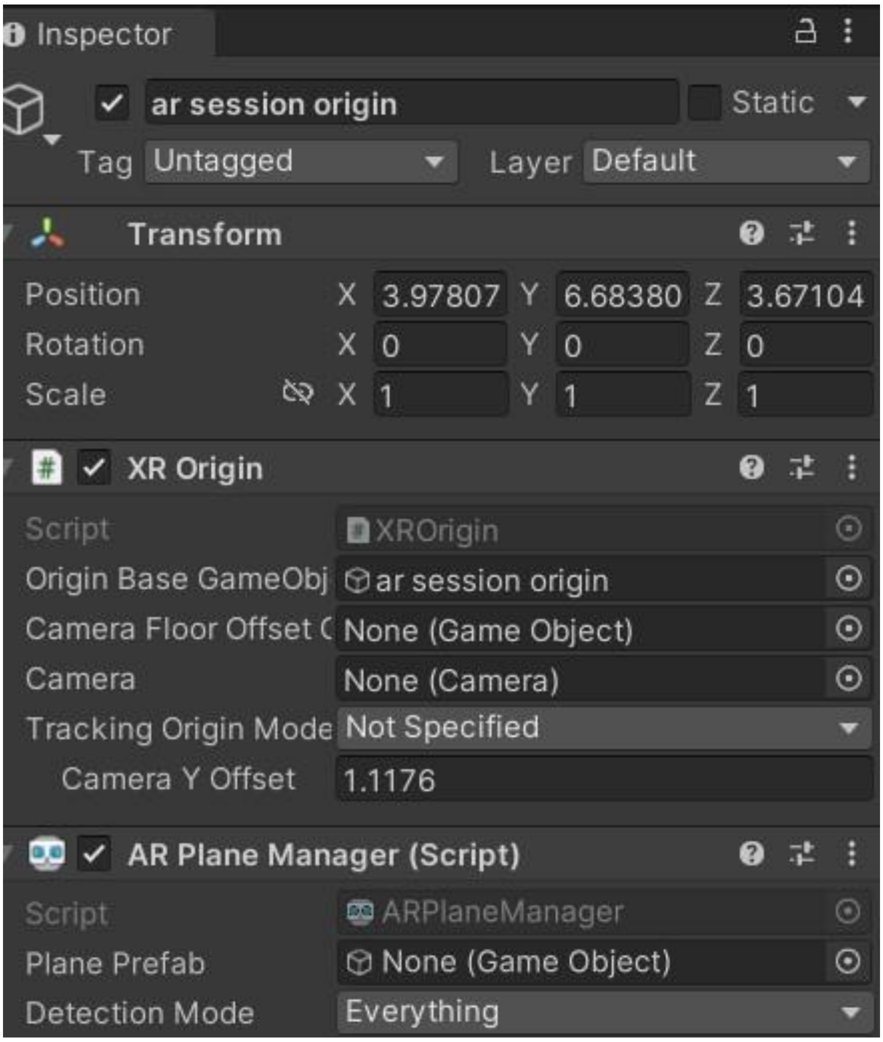
Software Dependencies: Unity 2020.3 or higher, AR Foundation, ARCore/ARKit SDKs, and Vuforia SDK for image recognition.

Graphics: GPU support with OpenGL ES 3.0 or Metal for optimal rendering.

Camera: Rear camera with a minimum resolution of 8 MP for effective image tracking and AR projection.

STEPS:

1) open unity
2) create new 3D project
3) install ar foundation in unity
4) install arcorexr plugin
5) download an maze image from google
6) create ar session
7) create ar session origin
8) add ar camera
9) add ar plane manager component
10) add plane in the scene
11) in assets import new asset of maze that was downloaded
12) drag that maze to plane
13) now add cube to plane
14) scale the cube according to maze
15) now duplicate the cube and cover all the border of maze by cube with scale
16) add material to asset
17) give the color to material
18) drag that material to cube so that cube get colored
19) add spehere to the maze
20) add script to ar session origin for plane detection



☑ ar session origin □ Static

Tag: Untagged

Layer: Default

Transform

	X	Y	Z
Position	3.97807	6.68380	3.67104
Rotation	0	0	0
Scale	1	1	1

☑ **XR Origin**

Script: XROrigin

Origin Base GameObj: ar session origin

Camera Floor Offset (: None (Game Object)

Camera: None (Camera)

Tracking Origin Mode: Not Specified

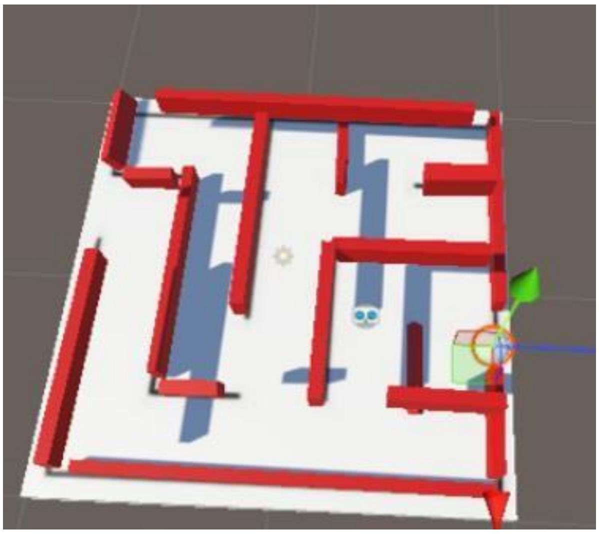
Camera Y Offset: 1.1176

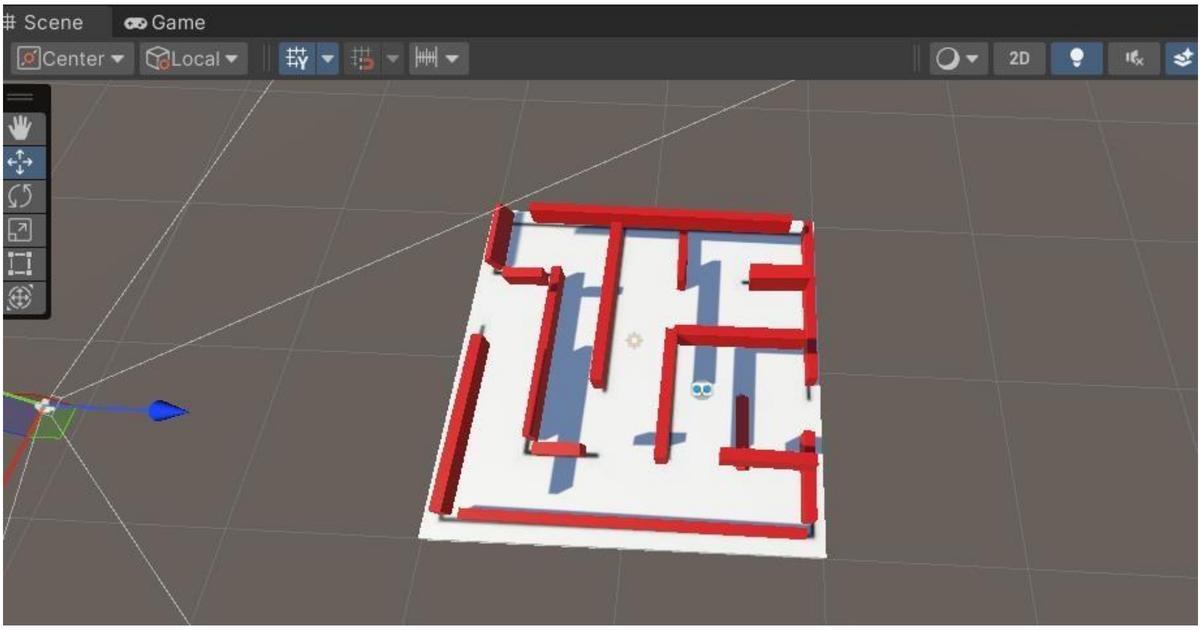
☑ **AR Plane Manager (Script)**

Script: ARPlaneManager

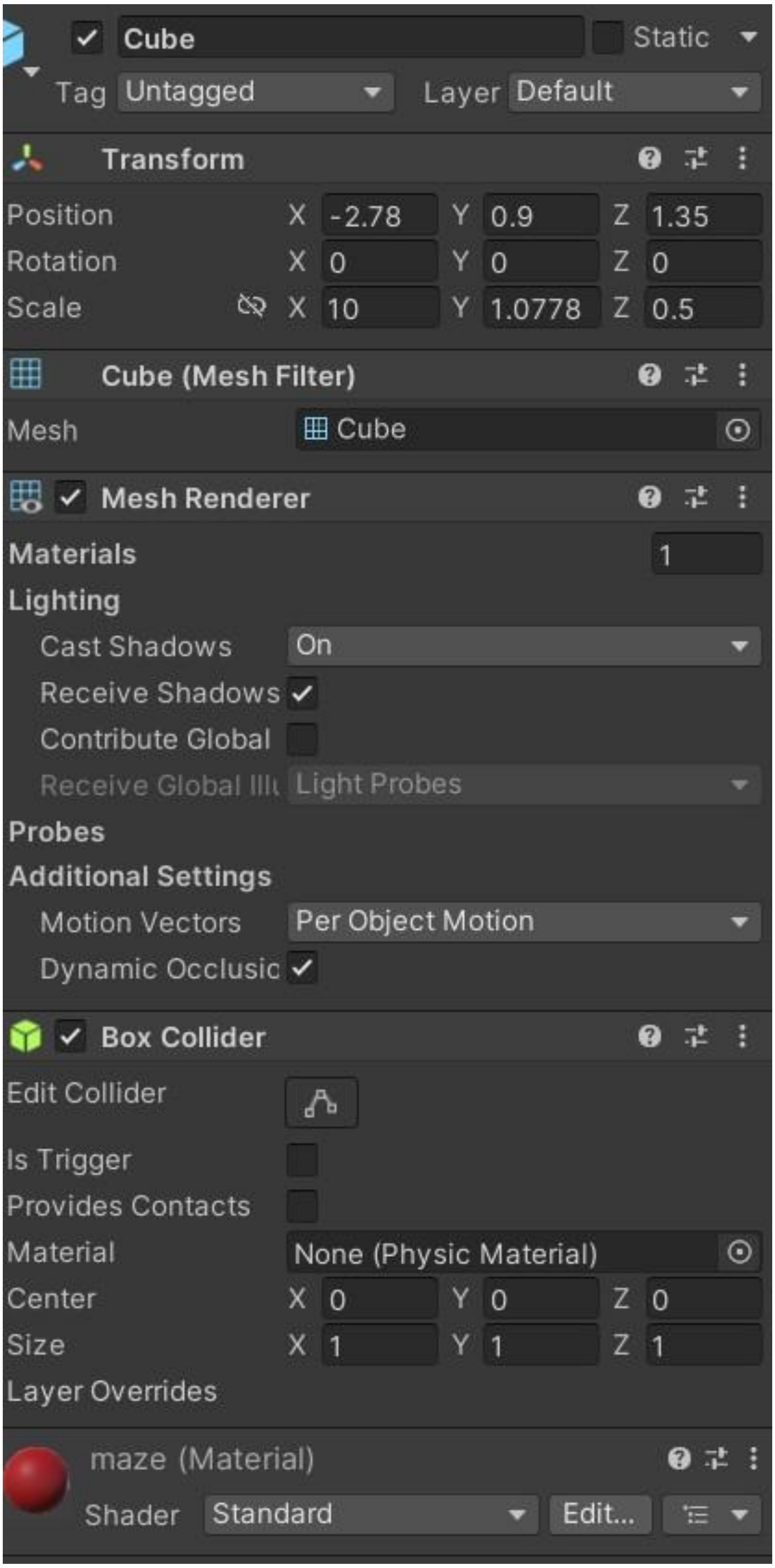
Plane Prefab: None (Game Object)

Detection Mode: Everything





Scene
Game
Center
Local
2D



☑ Cube □ Static

Tag Untagged Layer Default

Transform

	X	Y	Z
Position	-2.78	0.9	1.35
Rotation	0	0	0
Scale	10	1.0778	0.5

Cube (Mesh Filter)

Mesh Cube

☑ **Mesh Renderer**

Materials 1

Lighting

Cast Shadows On

Receive Shadows ☑

Contribute Global □

Receive Global Illı Light Probes

Probes

Additional Settings

Motion Vectors Per Object Motion

Dynamic Occlusic ☑

☑ **Box Collider**

Edit Collider

Is Trigger □

Provides Contacts □

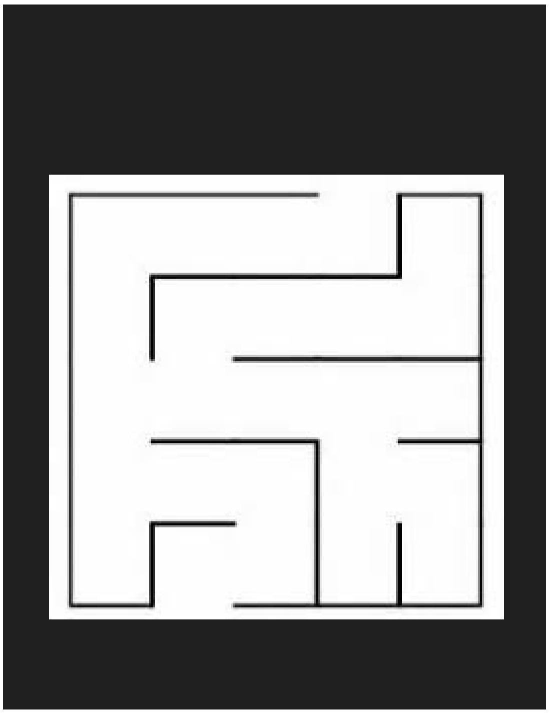
Material None (Physic Material)

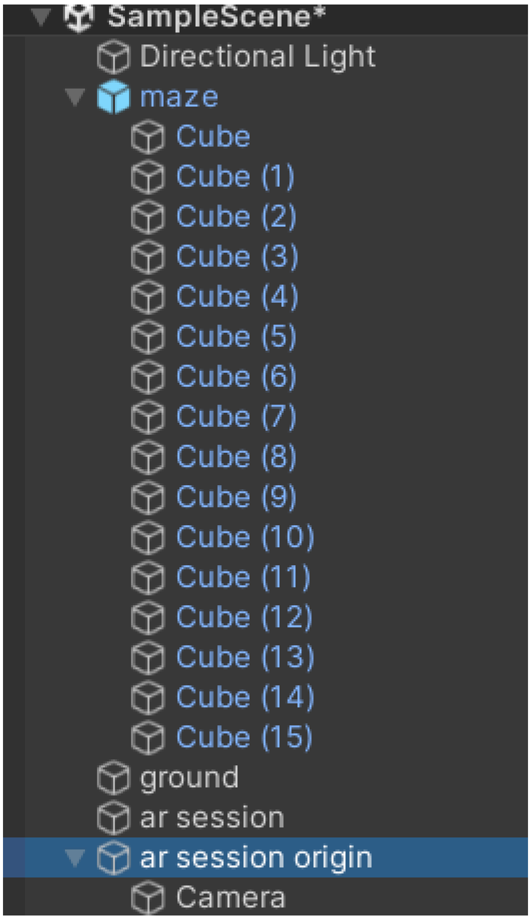
	X	Y	Z
Center	0	0	0
Size	1	1	1

Layer Overrides

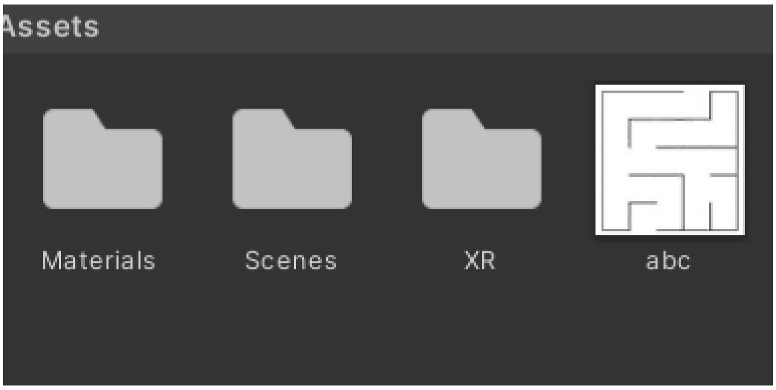
maze (Material)

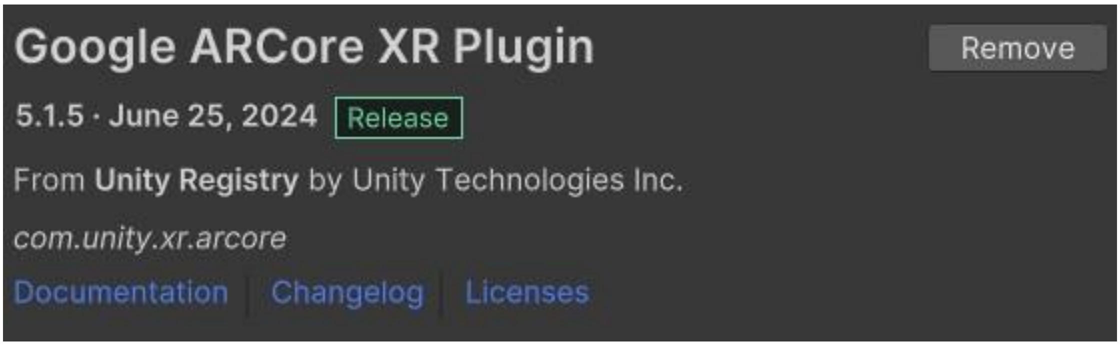
Shader Standard Edit...





- SampleScene*
  - Directional Light
  - maze
    - Cube
    - Cube (1)
    - Cube (2)
    - Cube (3)
    - Cube (4)
    - Cube (5)
    - Cube (6)
    - Cube (7)
    - Cube (8)
    - Cube (9)
    - Cube (10)
    - Cube (11)
    - Cube (12)
    - Cube (13)
    - Cube (14)
    - Cube (15)
  - ground
  - ar session
  - ar session origin
    - Camera



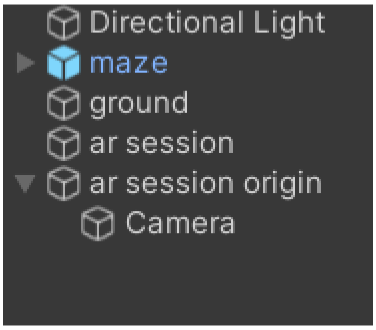


Google ARCore XR Plugin

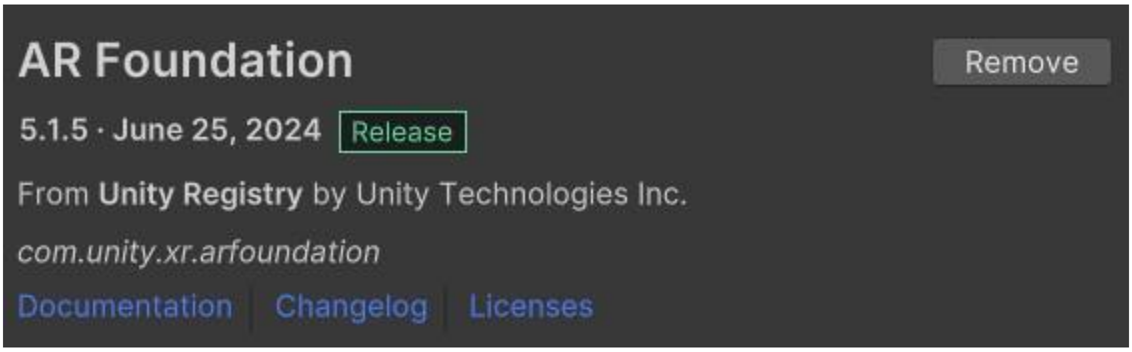
5.1.5 · June 25, 2024 Release

From **Unity Registry** by Unity Technologies Inc.

com.unity.xr.arcore



Directional Light
▶ maze
ground
ar session
▼ ar session origin
    Camera

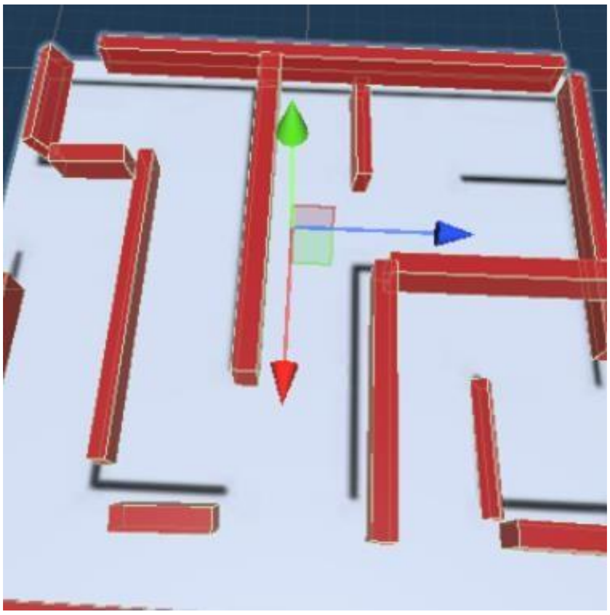


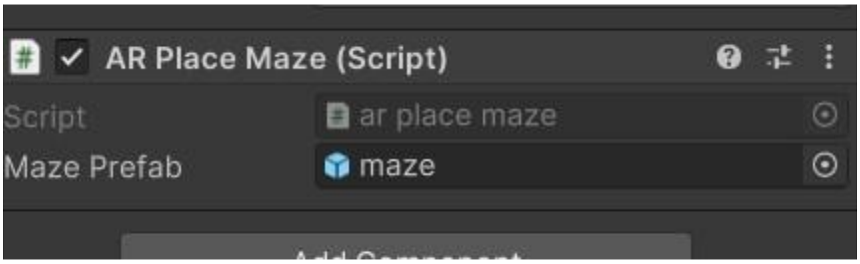
AR Foundation

5.1.5 · June 25, 2024 Release

From **Unity Registry** by Unity Technologies Inc.

com.unity.xr.arfoundation



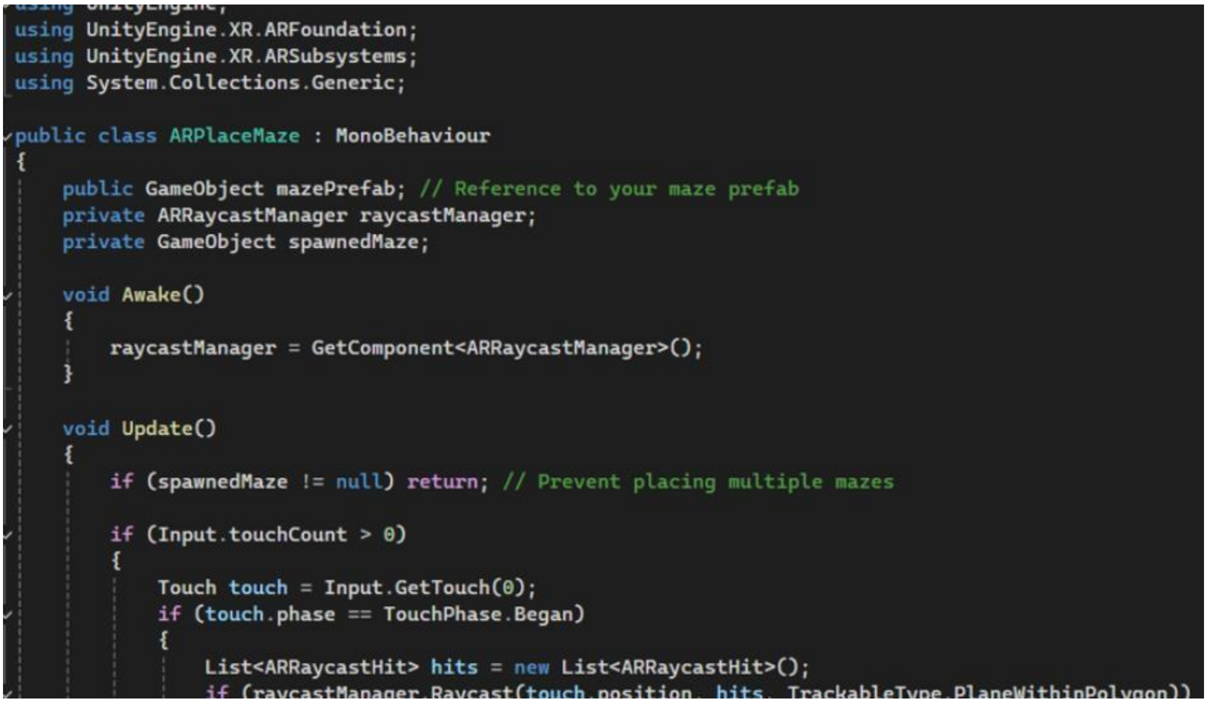


AR Place Maze (Script)

Script ar place maze

Maze Prefab maze

Add Component



```
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
using System.Collections.Generic;

public class ARPlaceMaze : MonoBehaviour
{
    public GameObject mazePrefab; // Reference to your maze prefab
    private ARRaycastManager raycastManager;
    private GameObject spawnedMaze;

    void Awake()
    {
        raycastManager = GetComponent<ARRaycastManager>();
    }

    void Update()
    {
        if (spawnedMaze != null) return; // Prevent placing multiple mazes

        if (Input.touchCount > 0)
        {
            Touch touch = Input.GetTouch(0);
            if (touch.phase == TouchPhase.Began)
            {
                List<ARRaycastHit> hits = new List<ARRaycastHit>();
                if (raycastManager.Raycast(touch.position, hits, TrackableType.PlaneWithinPolygon))
```