Augmented Reality
Practical file

Name: Chirag Gupta
Roll No.: 2019UCO1689
Subject Code: COCSE57

EXPERIMENT 1

Installation and basic understanding of Unity software for Augmented Reality.

Methodology

Unity Technologies developed Unity, which is a game engine that was first released in June 2005. The engine is capable of creating both 3D and 2D games and is compatible with various desktop, mobile, console, and virtual reality platforms. Unity is highly favored by beginner developers, and it is popular for iOS and Android mobile game development. Moreover, the Unity Hub is an independent application that is utilized for managing Unity projects, installing Editor versions, as well as licensing and installing additional components.

Here are the steps for using Unity Hub to create an AR project:

1. Download and install Unity Hub from the Unity Website
2. Open Unity Hub and Click add button to install new version of unity from Installs tab
3. Once the version of unity is installed, create new Project
4. Select the name and location for your project and choose the Unity version
5. Now, add AR Foundation and AR Core XR Package from Package manager. It is a cross-platform framework that allows you to build augmented reality experiences once, then build for either Android or iOS devices.
6. After the package is installed we can start building our AR application by adding various components to our scene like the AR Camera.

Experiment 2

Build an Augmented Reality application having 3D object in it using Unity.

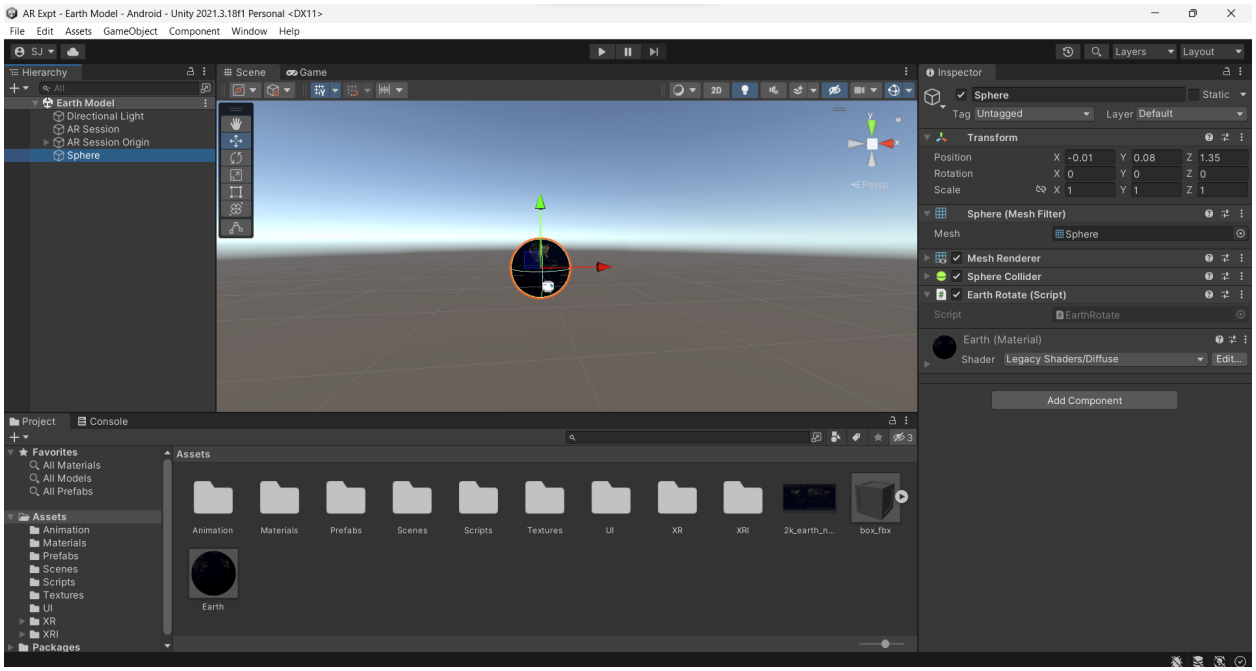
Requirements

1. Unity Hub
2. Earth material
3. VS Code
4. Android Phone

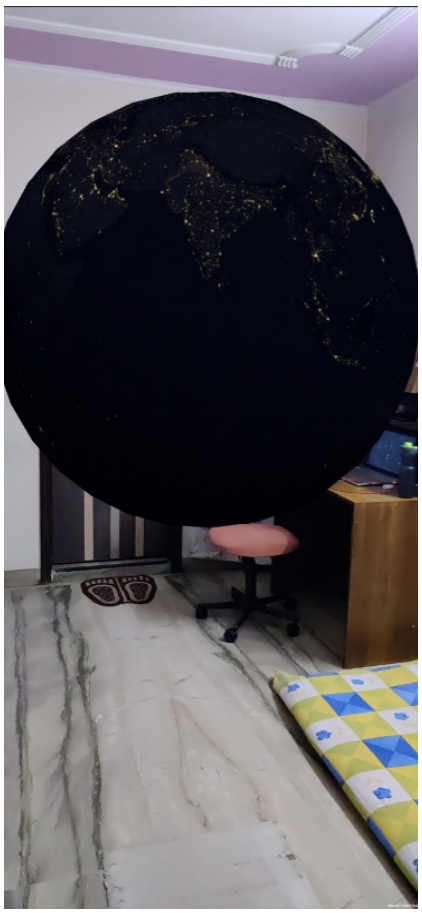
Methodology

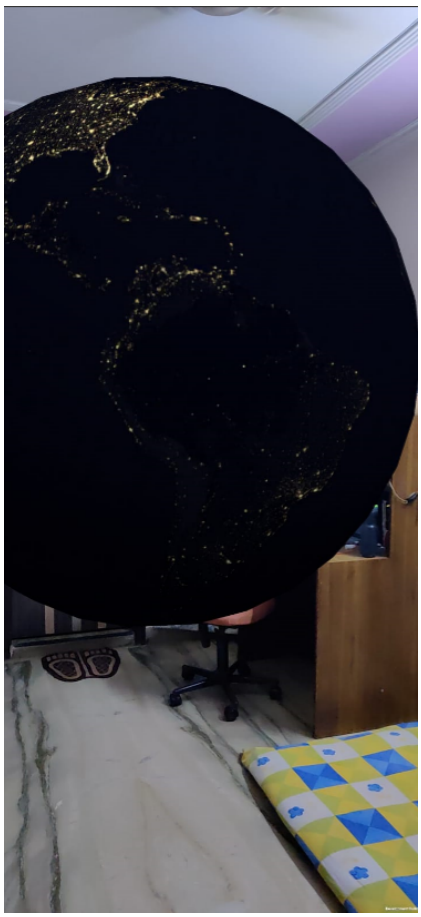
1. Open the unity and select 3d and create new project
2. Download the AR package AR foundation, AR core from windows package manager.
3. Go to XR and add AR session and AR session origin in hierarchy window.
4. Delete the main camera
5. Add sphere for earth and download earth texture online
6. Make a material having texture added to it and then add this material to sphere.
7. Bring the camera out of sphere so that sphere is in view of camera.
8. Build and run the application by setting it for android.

Unity Scene



Output





Experiment 3

Build an Augmented Reality application having Placement Indicator in it to summon 3D object in it using Unity.

Requirements

1. Unity Hub
2. Placement Indicator Image
3. Sports Equipment asset from Asset Store
4. VS Code
5. Android Phone

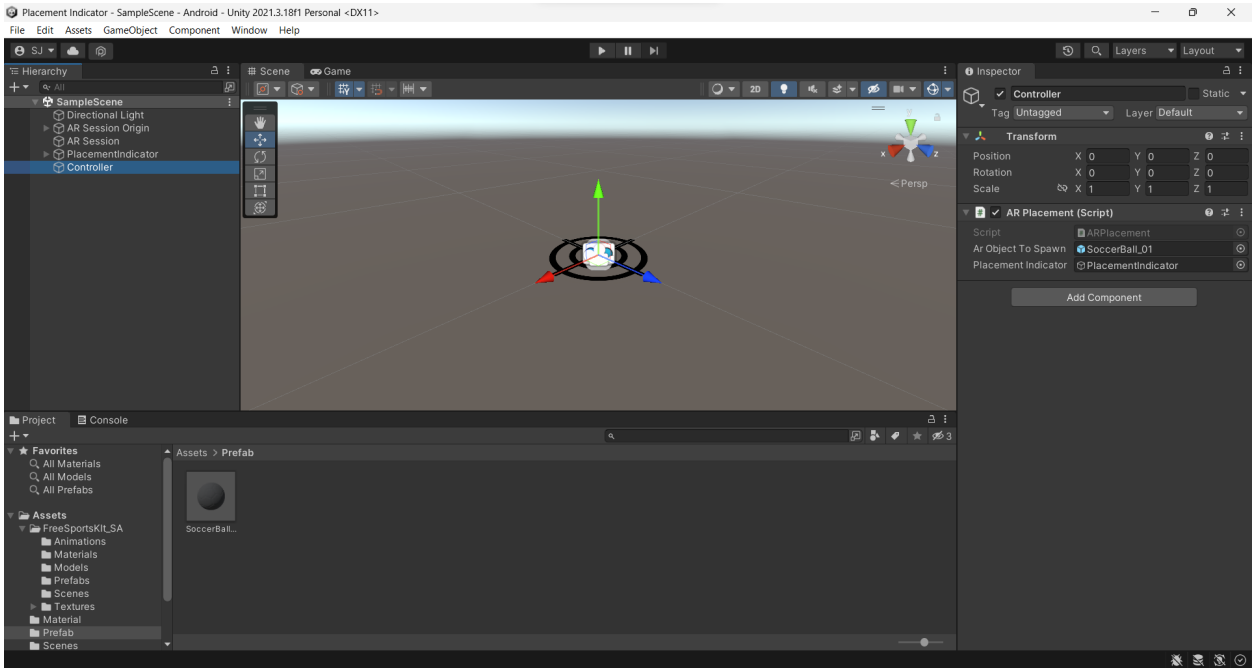
Methodology

To build the application you have to follow the following steps:

1. First we have to step up your project for AR application and download the necessary package like AR Foundation, ARCore XR plugin.
2. Then we need to set up a AR session by using AR Session Origin and AR Session. AR Session Origin contains the camera so we need to delete the default Main Camera from the project.
3. Then we add the necessary components for the application to our AR Session Origin. AR Plane Manager is added to manage the plane generated on which the object will be displayed and AR RayCastManager is added to get the location of the position to place the placement indicator.
4. Then we create a Game Object called placement indicator. We then create a prefab for using a target png image and create a material using it to be added to the placement indicator as a material which will act as the target.

5. Then create a GameObject named Controller which will contain the script (ARPlcement.cs) for the working of the application which is given below.
6. Then we imported a sports equipment asset from the asset store to create its prefab to be displayed as the 3D object.
7. Finally we link the sports equipment prefab and the placement indicator to the Game Object for script for the final working.
8. We can now build and run this application using any android device.

Unity Scene



C# Script

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

public class ARPlacement : MonoBehaviour
{
```

```
    public GameObject arObjectToSpawn;
    public GameObject placementIndicator;
    private GameObject spawnedObject;
    private Pose PlacementPose;
    private ARRaycastManager aRRaycastManager;
    private bool placementPoseIsValid = false;
    void Start()
    {
        aRRaycastManager = FindObjectOfType<ARRaycastManager>();
    }
    void Update()
    {
        if(spawnedObject == null && placementPoseIsValid &&
Input.touchCount > 0 && Input.GetTouch(0).phase == TouchPhase.Began)
        {
            ARPlaceObject();
        }
        UpdatePlacementPose();
        UpdatePlacementIndicator();
    }
    void UpdatePlacementIndicator()
    {
        if(spawnedObject == null && placementPoseIsValid)
        {
            placementIndicator.SetActive(true);

placementIndicator.transform.SetPositionAndRotation(PlacementPose.position
, PlacementPose.rotation);
        }
        else
        {
            placementIndicator.SetActive(false);
        }
    }
    void UpdatePlacementPose()
    {
        var screenCenter = Camera.current.ViewportToScreenPoint(new
Vector3(0.5f, 0.5f));
        var hits = new List<ARRaycastHit>();
        aRRaycastManager.Raycast(screenCenter, hits,
TrackableType.Planes);
        placementPoseIsValid = hits.Count > 0;
        if(placementPoseIsValid)
        {
            PlacementPose = hits[0].pose;
        }
    }
```

```
    void ARPlaceObject()
    {
        spawnedObject = Instantiate(arObjectToSpawn,
PlacementPose.position, PlacementPose.rotation);
    }
}
```

Output

Experiment 4

Build an Augmented Reality application using Unity for inserting multiple AR objects.

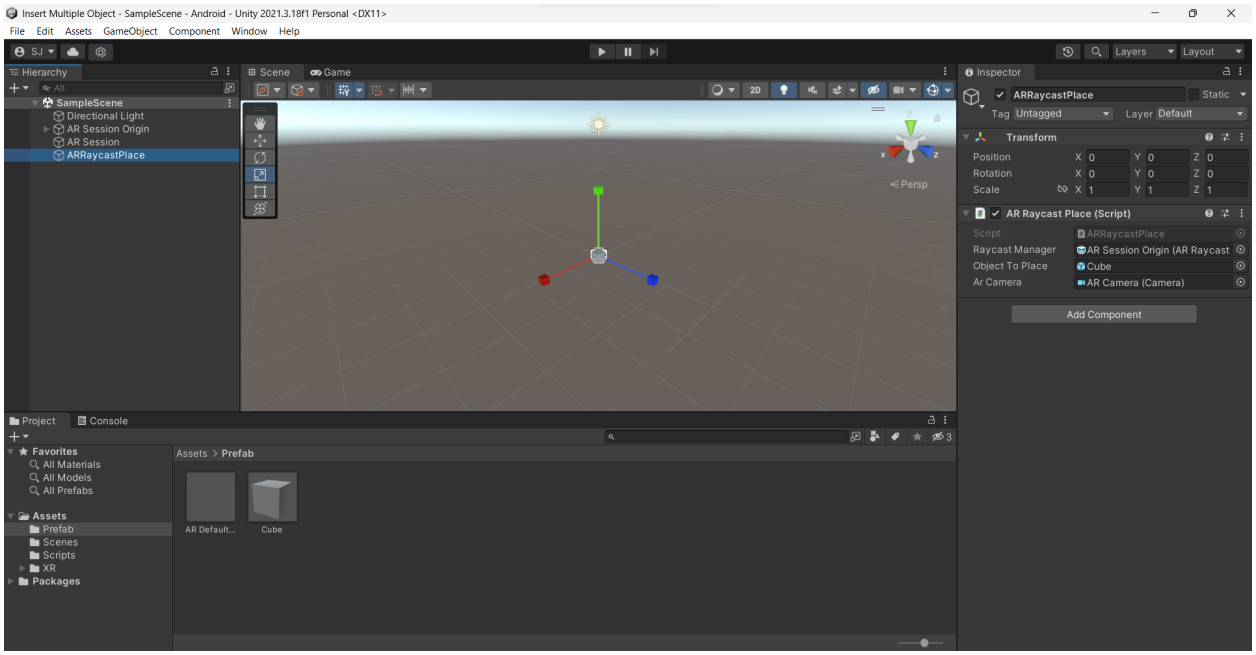
Requirements

1. Unity Hub
2. VS Code
3. Android Phone

Methodology

1. First we have to step up your project for AR application and download the necessary package like AR Foundation, ARKit or ARCore plugin.
2. Then we need to setup an AR session by using AR Session Origin and AR Session. AR Session Origin contains the camera so we need to delete the default Main Camera from the project.
3. Then we add the necessary components for the application to our AR Session Origin. AR Plane Manager is added to manage the plane generated on which the object will be displayed and AR RayCastManager is added to get the location of the position to place the placement indicator.
4. Then we created AR default plane and then converted it into a prefab
5. Then we create a script called ARRaycastPlace to place multiple objects on the ground
6. We add AR session origin to raycast manager, then add an AR camera and a Cube to be the object to be placed
7. We can now build and run this application using any android device.

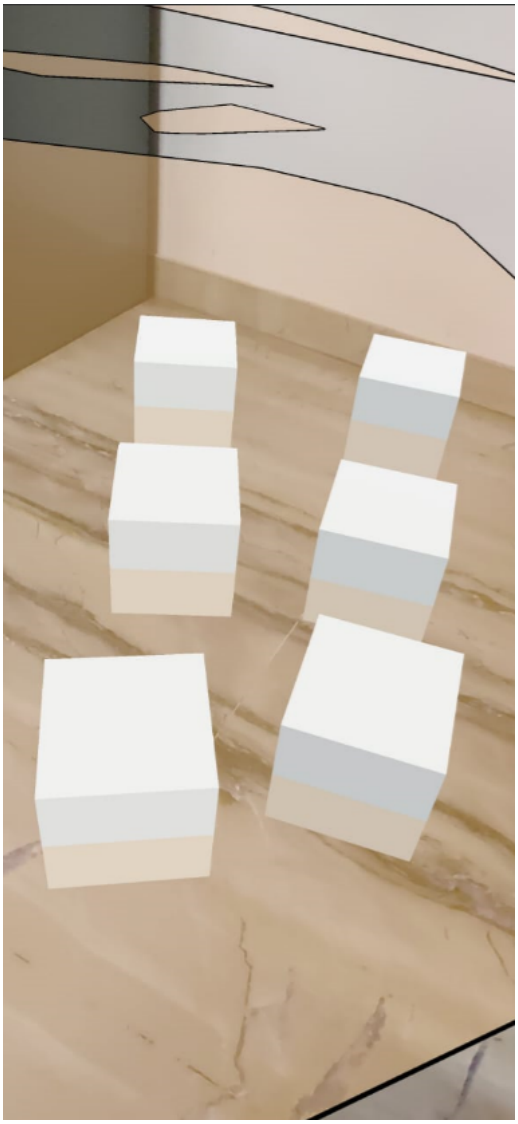
Unity Scene



C# Script

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

public class ARRaycastPlace : MonoBehaviour
{
    public ARRaycastManager raycastManager;
    public GameObject objectToPlace;
    public Camera arCamera;

    private List<ARRaycastHit> hits = new List<ARRaycastHit>();

    void Update()
    {
        Ray ray = arCamera.ScreenPointToRay(Input.mousePosition);

        if(Input.GetMouseButton(0)) {
            if(raycastManager.Raycast(ray, hits, TrackableType.Planes)) {
                Pose hitPose = hits[0].pose;

                Instantiate(objectToPlace, hitPose.position,
hitPose.rotation);
            }
        }
    }}
```

Output



Experiment 5 & 6

Build an Augmented Reality application using Unity to use arrows as placement indicators to summon multiple AR objects. (It includes the 5th experiment in which we have to summon multiple objects in AR Environment)

Requirements

1. Unity Hub
2. Placement Indicator Image
3. Sports Equipment asset from Asset Store
4. VS Code
5. Android Phone

Methodology

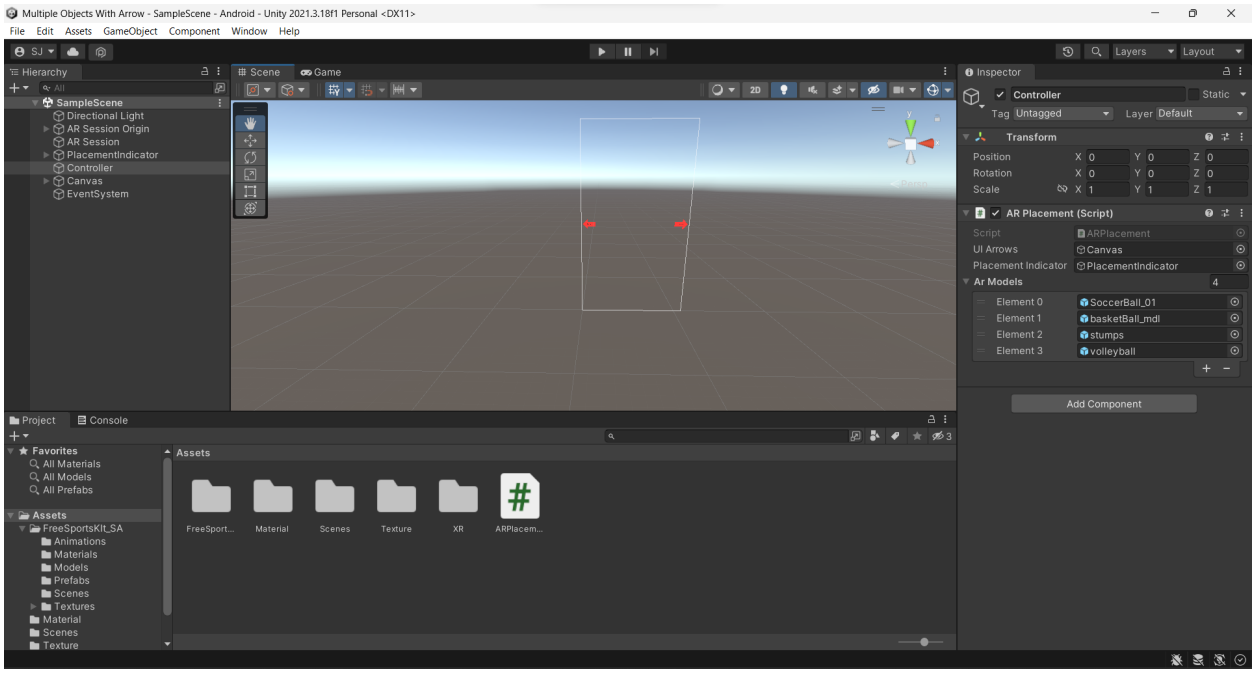
To build the application you have to follow the following steps:

1. First we have to step up your project for AR application and download the necessary package like AR Foundation, ARKit o r ARCore plugin.
2. Then we need to step up an AR session by using AR Session Origin and AR Session. AR Session Origin contains the camera so we need to delete the default Main Camera from the project.
3. Then we add the necessary components for the application to our AR Session Origin. AR Plane Manager is added to manage the plane generated on which the object will be displayed and AR RayCastManager is added to get the location of the position to place the placement indicator.
4. Then we create a Game Object called placement indicator which acts as the indicator for the object to be displayed. We then create a prefab for using a target png image and create

a material using it to be added to the placement indicator as a material which will act as the target.

5. Then create a GameObject named Controller which will contain the script (ARPlcement.cs) for the working of the application which is given below.
6. Then we imported a sports equipment asset from the asset store to create its prefab to be displayed as the 3D object. Also we make two more prefabs for a capsule and a sphere.
7. Add a UI button to the scene that the user can tap to place an arrow. When the button is pressed, create an instance of the arrow model and attach it to a touch point on the screen. Use the AR Raycast Manager to determine the position and orientation of the surface where the arrow should be placed.
8. When an arrow is placed, display a menu of 3D objects that the user can summon by tapping on the arrow. You can use a simple UI panel to display the menu, with each object represented by a button.
9. When the user taps on a button in the menu, instantiate the corresponding 3D object and attach it to the arrow. Use the position and orientation of the arrow to determine where the object should be placed.
10. We can now build and run this application using any android device.

Unity Scene



C# Script

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

public class ARPlacement : MonoBehaviour
{
    public GameObject UIArrows;

    public GameObject placementIndicator;
    private GameObject spawnedObject;
    private Pose PlacementPose;
    private ARRaycastManager aRRaycastManager;
    private bool placementPoseIsValid = false;

    public GameObject[] arModels;
    int modelIndex = 0;

    void Start()
    {
        aRRaycastManager = FindObjectOfType<ARRaycastManager>();
        UIArrows.SetActive(false);

    }

    void Update()
```

```
    {
        if (spawnedObject == null && placementPoseIsValid &&
Input.touchCount > 0 && Input.GetTouch(0).phase == TouchPhase.Began)
        {
            ARPlaceObject(modelIndex);
            UIArrows.SetActive(true);
        }


        UpdatePlacementPose();
        UpdatePlacementIndicator();


    }
    void UpdatePlacementIndicator()
    {
        if (spawnedObject == null && placementPoseIsValid)
        {
            placementIndicator.SetActive(true);

placementIndicator.transform.SetPositionAndRotation(PlacementPose.position
, PlacementPose.rotation);
        }
        else
        {
            placementIndicator.SetActive(false);
        }
    }

    void UpdatePlacementPose()
    {
        var screenCenter = Camera.current.ViewportToScreenPoint(new
Vector3(0.5f, 0.5f));
        var hits = new List<ARRaycastHit>();
        aRRaycastManager.Raycast(screenCenter, hits,
TrackableType.Planes);

        placementPoseIsValid = hits.Count > 0;
        if (placementPoseIsValid && spawnedObject == null)
        {
            PlacementPose = hits[0].pose;
        }
    }

    void ARPlaceObject(int id)
    {
        for(int i = 0; i < arModels.Length; i++)
        {
            if(i == id)
            {
                GameObject clearUp =
GameObject.FindGameObjectWithTag("ARMultiModel");
                Destroy(clearUp);
```

```
                spawnedObject = Instantiate(arModels[i],
PlacementPose.position, PlacementPose.rotation);
            }
        }
    }

    public void ModelChangeRight()
    {
        if (modelIndex < arModels.Length - 1)
            modelIndex++;
        else
            modelIndex = 0;

        ARPlaceObject(modelIndex);
    }
    public void ModelChangeLeft()
    {
        if (modelIndex > 0)
            modelIndex--;
        else
            modelIndex = arModels.Length - 1;

        ARPlaceObject(modelIndex);
    }
}
```

Output


FLASH
FLASH
FLASH