EXPERIMENT-1

- Installation and basic understanding of Unity software for Augmented Reality.

Methodology - Unity Technologies created a game engine named "Unity" which was first announced and launched in June 2005. This engine provides a platform for developers to create 2D or 3D games and is suitable for multiple operating systems. Over time, Unity has expanded its capabilities to include support for various desktop, mobile, console, and virtual reality platforms. It is especially well-liked for mobile game development on iOS and Android and is known for being user-friendly for beginner developers, which has made it a popular choice for indie game development.

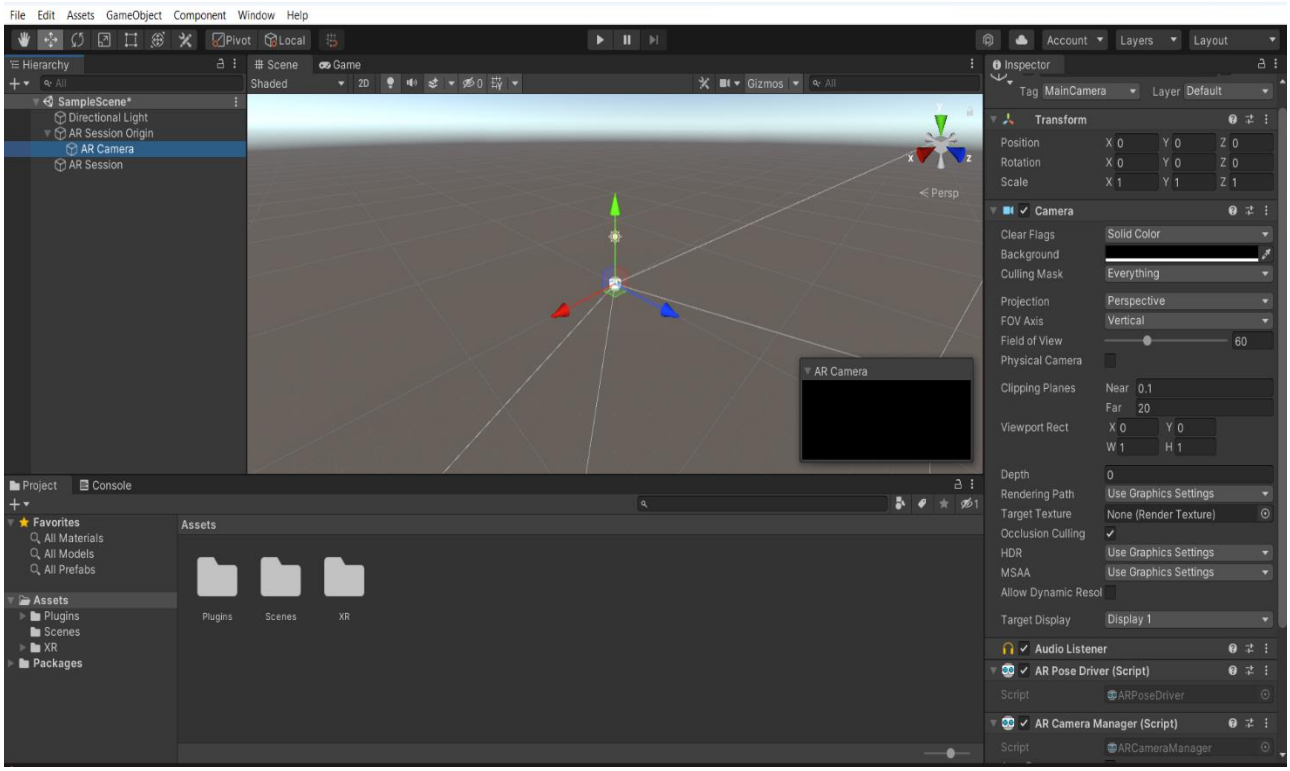
The Unity Hub is a separate software program that provides access to the Unity ecosystem. It is utilized for various tasks, including managing Unity projects, installing different versions of the Unity Editor, as well as licensing and installing additional components.

Steps for installing and AR app development-

1. Open unity.com website and click on download button.
2. Open the downloaded file and then click on install.
3. Now open the unity hub and click on install editor.
4. Select the version of editor.
5. Select the SDK and JDK and then click on install.
6. During installation it asks to install visual studio, select it, and select the c# editor in visual studio and then install it too.
7. Once it is downloaded select 3d core and click on new project
8. Type the name of project and then press enter.
9. To create an AR application, you will need to install the AR Foundation package from the Unity Asset Store. This package provides a set of tools

for creating AR applications that can run on different platforms, such as iOS and Android.

10. After the AR Foundation package is installed, you can start building your AR application by adding AR components to your scene, such as an AR camera and AR plane manager.
11. You can test your AR application by running it in the Unity Editor or by building it for a specific platform, such as iOS or Android.
12. Once your AR application is complete, you can use Unity Hub to manage your projects and share them with others.



Challenges Overcome-

1. Installing and setting up Unity Hub.
2. Installing the different versions of Unity.
3. Familiarizing oneself with the project environment and checking out various features.
4. Learning to deploy applications on your android device.

EXPERIMENT-2

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

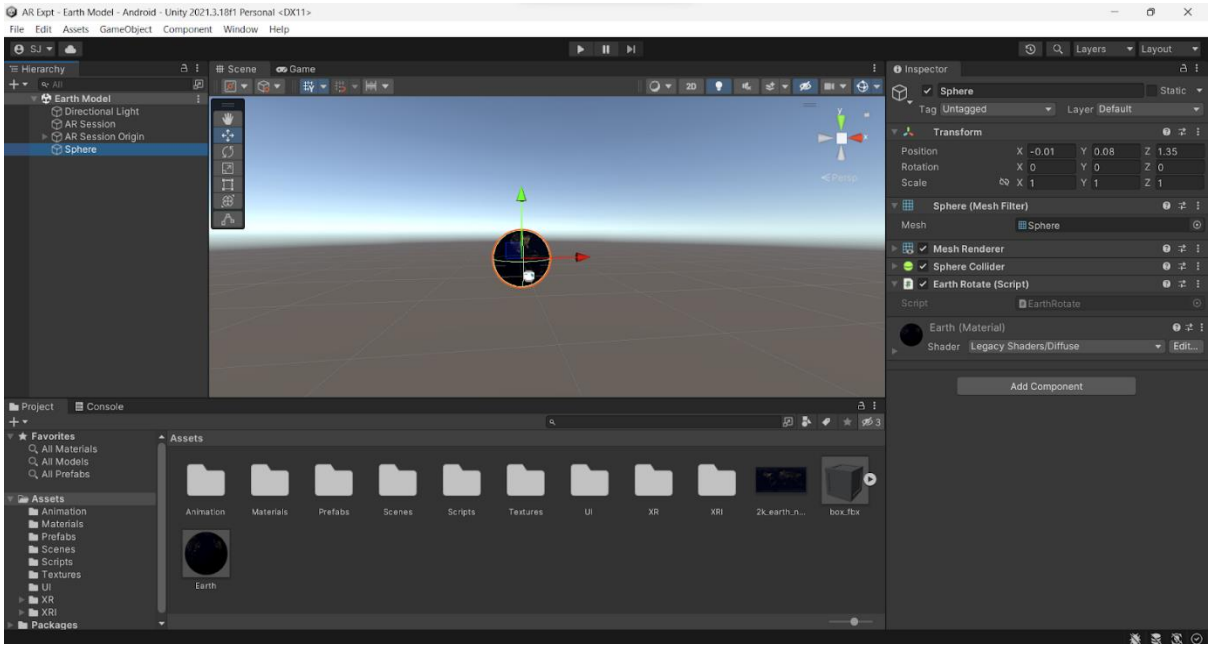
Requirements-

- UNITY HUB
- 3D-object (Sphere)
- Visual Studio
- Android Phone

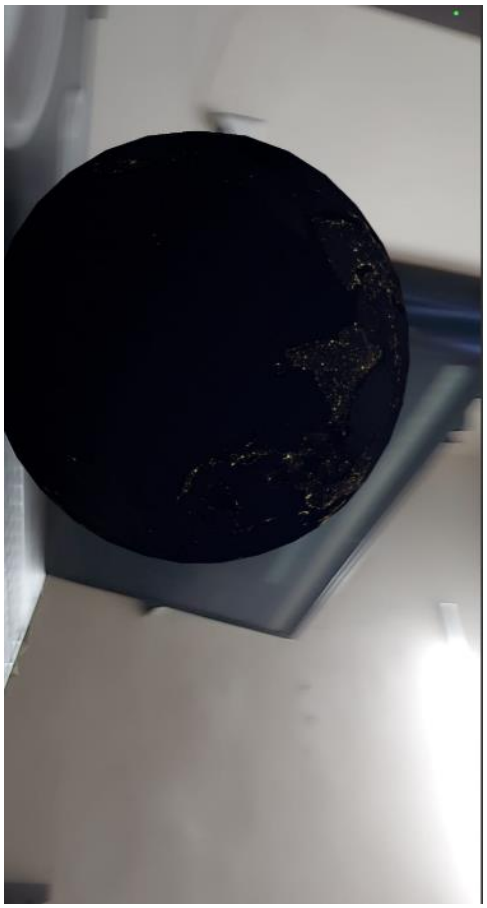
Steps-

1. open the unity and select 3d and create new project
2. download the AR package AR foundation, AR core, AR kit plugin 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. Add a script to rotate the sphere along y axis so that earth looks like moving along y axis.
9. Build and run the application by setting it for android.

Unity Scene-


AR Expt - Earth Model - Android - Unity 2021.3.18f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
Earth Model
Directional Light
AR Session
AR Session Origin
Sphere
Scene
Game
Inspector
Sphere
Tag Untagged
Layer Default
Transform
Position
Rotation
Scale
Sphere (Mesh Filter)
Mesh
Mesh Renderer
Sphere Collider
Earth Rotate (Script)
Script
EarthRotate
Earth (Material)
Shader
Legacy Shaders/Diffuse
Add Component
Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
Animation
Materials
Prefabs
Scenes
Scripts
Textures
UI
XR
XRI
Packages
Earth

Output-



EXPERIMENT-3

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

Requirements-

- Unity Hub
- Placement indicator
- Asset from unity asset store
- Visual Studio
- Android phone

Steps-

1. we have to step up your project for AR application and download the necessary package like AR Foundation, ARKit , ARCore plugin.
2. Then we need to step 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 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 spider asset from the asset store to create its prefab to be displayed as the 3D object.

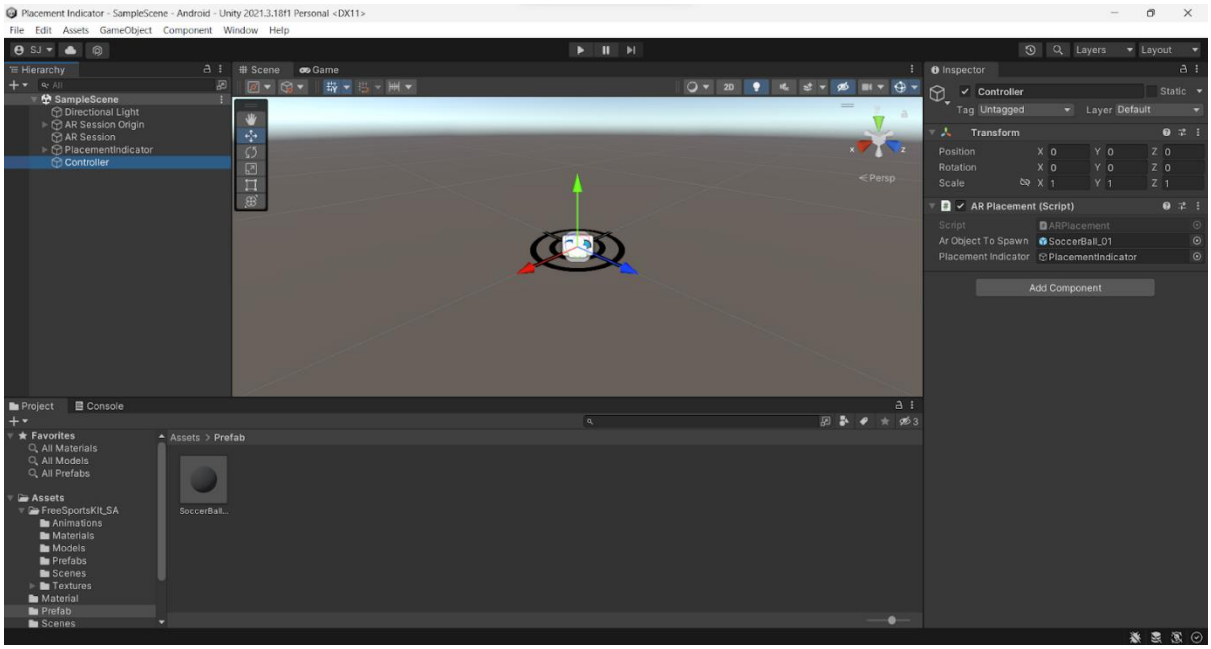
7. Finally we link the spider prefab and the placement indicator to the Game Object for script for the final working.
8. We can now build and run this application using any android device.

Code-

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.ARSubsystems;
using UnityEngine.XR.ARFoundation;
public class ARPlacement : MonoBehaviour
{
    public GameObject arObjectToSpawm;
    public GameObject placementIndicator;
    private GameObject spawnObject;
    private Pose PlacementPose;
    private ARRaycastManager aRRaycastManager;
    private bool placementPoseIsValid = false;
    void Start()
    {
        aRRaycastManager = FindObjectOfType<ARRaycastManager>();
        placementIndicator.SetActive(false);
    }
    void Update()
    {
        if(spawnObject==null && placementPoseIsValid &&
Input.touchCount > 0 && Input.GetTouch(0).phase == TouchPhase.Began)
        {
            ARPlaceObject();
        }
        UpdatePlacementPose();
        UpdatePlacementIndicator();
    }
    void UpdatePlacementIndicator()
    {
        if(spawnObject==null && placementPoseIsValid)
        {
```

```
            placementIndicator.SetActive(true);
placementIndicator.transform.SetPositionAndRotation(PlacementPose.posit
ion,PlacementPose.rotation);
        }
        else
        {
            placementIndicator.SetActive(false)
        }
    }
    void UpdatePlacementPose()
    {
        var screenCenter = Camera.main.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()
    {
        spawnObject =
Instantiate(arObjectToSpawm,PlacementPose.position,PlacementPose.rotati
on);
    }
}
```

Unity scene-


Placement Indicator - SampleScene - Android - Unity 2021.3.18f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
SampleScene
Directional Light
AR Session Origin
AR Session
PlacementIndicator
Controller
Scene
Game
Persp
Inspector
Controller
Tag Untagged
Layer Default
Transform
Position
Rotation
Scale
AR Placement (Script)
Script ARPlacement
Ar Object To Spawn SoccerBall_01
Placement Indicator PlacementIndicator
Add Component
Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
FreeSportsKit_SA
Animations
Materials
Models
Prefabs
Scenes
Textures
Material
Prefab
Scenes
Assets > Prefab
SoccerBall...

Output-

Challenges-

1. Understanding the use of Plane Manager and RayCast Manager
2. Learning to create prefabs and materials.
3. Understanding how GameObject works.
4. Understanding how to write a C# script and learning the basics of its semantics.
5. Setting up the different components for the application.

EXPERIMENT-4

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

Requirements-

- Unity Hub
- Visual Studio
- 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.

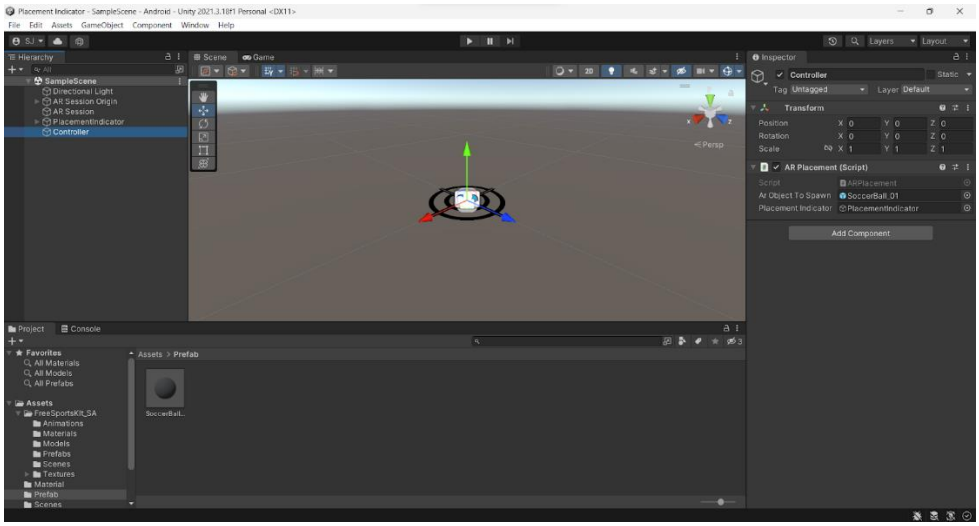
4. Then we created AR default plane and then converted it into a prefab
5. Then we create a script called ARRaycastPlace to place multiple objects on the ground
6. We add AR session origin to raycast manager, then add an AR camera and a Cube to be the object to be placed

7. We can now build and run this application using any android device.

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);
         }
      }
   }
}
```

Unity Scene-



Output-

Challenges-

1. Understanding the use of Plane Manager and RayCast Manager
2. Learning to create prefabs and materials.
3. Understanding how GameObject works.
4. Understanding how to write a C# script and learning the basics of its semantics.

5. Setting up the different components for the application.

EXPERIMENT-5 and 6

- Create an Augmented Reality (AR) app with Unity that utilizes arrow markers to indicate the placement of multiple AR objects. This app is specifically designed for the 5t experiment which involves summoning multiple objects in an AR environment.

Requirements-

- Unity hub
- Placement indicator image
- Spider asset from asset store
- Visual Studio
- Android Phone

Steps-

1. First we have to step up your project for AR application and download the necessary package like AR Foundation, ARKit, 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 spider 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.

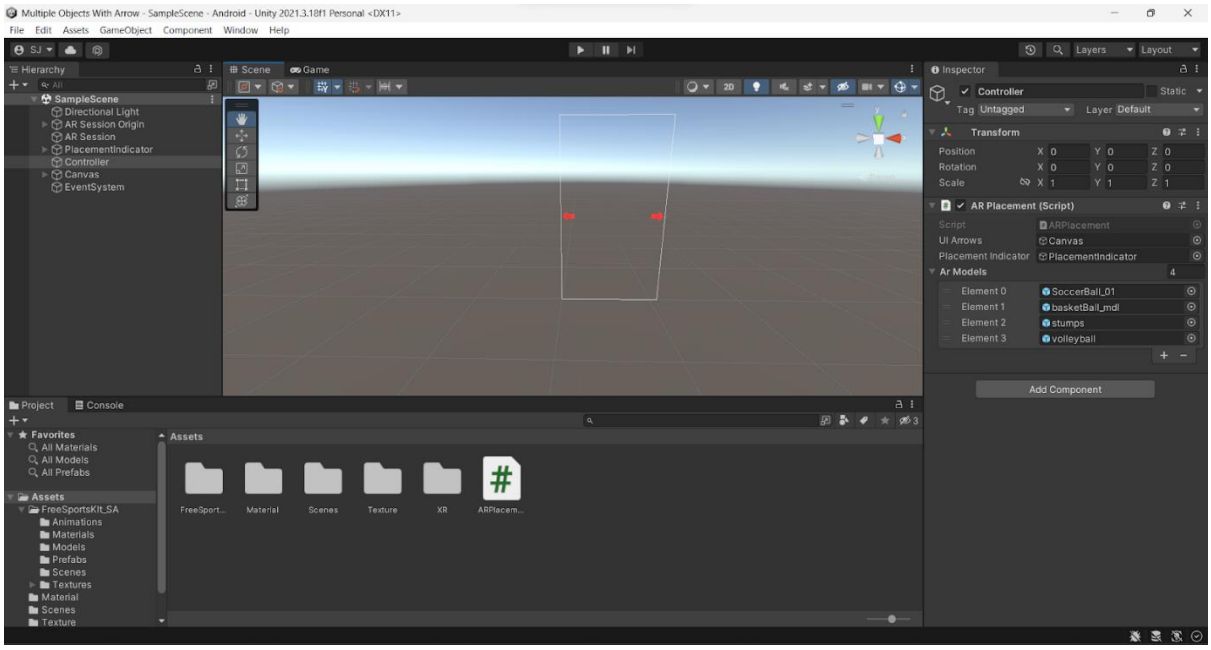
C# Code-

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

```
using UnityEngine.XR.ARFoundation;
public class ARPlacement : MonoBehaviour
{
    public GameObject UIArrows;
    public GameObject placementIndicator;
    private GameObject spawnObject;
    private Pose PlacementPose;
    private ARRaycastManager aRRaycastManager;
    private bool placementPoseIsValid = false;
    public GameObject[] arModels;
    int modelIndex = 0;
    void Start()
    {
        aRRaycastManager = FindObjectOfType<ARRaycastManager>();
        UIArrows.SetActive(false);
        placementIndicator.SetActive(false);
    }
    void Update()
    {
        if(spawnObject==null && placementPoseIsValid &&
Input.touchCount > 0 && Input.GetTouch(0).phase == TouchPhase.Began)
        {
            ARPlaceObject(modelIndex);
            UIArrows.SetActive(true);
        }
        UpdatePlacementPose();
        UpdatePlacementIndicator();
    }
    void UpdatePlacementIndicator()
    {
        if(spawnObject==null && placementPoseIsValid)
        {
            placementIndicator.SetActive(true);
placementIndicator.transform.SetPositionAndRotation(PlacementPose.posit
ion,PlacementPose.rotation);
        }
        else
        {
            placementIndicator.SetActive(false);
        }
    }
    void UpdatePlacementPose()
    {
```

```
        var screenCenter = Camera.main.ViewportToScreenPoint(new
Vector3(0.5f,0.5f));
        var hits = new List<ARRaycastHit>();
aRRaycastManager.Raycast(screenCenter,hits,TrackableType.Planes);
        placementPoseIsValid = hits.Count > 0;
        if(placementPoseIsValid && spawnObject == 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);
                spawnObject =
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);
    }
}
```

Unity Scene-



Output-

Challenges-

1. Understanding the use of Plane Manager and RayCast Manager
2. Learning to create prefabs and materials.
3. Understanding how GameObject works.
4. Understanding how to write a C# script and learning the basics of its semantics.
5. Understanding the working of UI buttons and canvas.
6. Setting up the different components for the application.