AUGMENTED REALITY

(COCSE57)

AARYAN RAJ SARDA

ROLL NO: 2019UCO1684

COE 3

Batch - 2

SEMESTER 8

NSUT, DELHI

NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY, DELHI

EXPERIMENT 1

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

SOLUTION

The app has been created by using Unity Hub and the 3D object used is the Cube which is a free 3D object provided on the unity asset manager online.

The unity scene space contains has 5 components that are the following :

1) Directional Light
2) AR Session
3) XR Origin
4) AR Default
5) AR Controller

To set up the AR environment we require the following.

1) The main camera controller that unity provides has to be removed for the working of the XR Origin Camera for creating augmented reality objects.
2) Two components the AR Plane Manager, and AR Raycast Manager are added as new components to XR Origin
3) Finally, the AR Controller is created which is a custom controller thatwould contain a C# script for the 3D object. The object is also scripted while programming the aspect with a Raycast manager object as well.
4) Finally, the build of the app is done and the app is securelydeployed on the phone.

Components Required:

1) Unity Hub
2) 3D Object (Cube Unity 3D Asset)
3) Vs Code
4) Simulator or a Phone

Code

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

public class ARController : MonoBehaviour
{
    public GameObject CubeObject;
    public ARRaycastManager RaycastManager;

    // Update is called once per frame
    void Update()
    {
        if(Input.touchCount > 0 && Input.GetTouch(0).phase == TouchPhase.Began){
            List<ARRaycastHit> touches = new List<ARRaycastHit>();
            RaycastManager.Raycast(Input.GetTouch(0).position, touches,
UnityEngine.XR.ARSubsystems.TrackableType.Planes);
            if(touches.Count > 0){
                GameObject.Instantiate(CubeObject, touches[0].pose.position,
touches[0].pose.rotation);
            }
        }
    }

    void Start(){

    }
}
```

Images of the app:

Challenges Overcome

1) Installing unity hub and creating build settings for the environment were specific to the system and the simulator environments.
2) The components and unity system were an aspect to familiarizewith for creating the app.
3) Setting a script for the controller in C# was interesting and its object-oriented feature provides modularity.

“I have done this assignment on my own. I have not copied anything from another student or any online source. I understand if my work is found like somebody else’s code, my case can be sent to the Disciplinary committee of the institute for appropriate action”.

EXPERIMENT 2

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

SOLUTION

The augmented reality app was developed using Unity Hub and utilizes a free 3D object of a dumbbell from the Unity Asset Store. The Unity scene includes 5 components: Directional Light, AR Session, XR Origin, AR Default, and AR Controller.

In order to set up the AR environment, the default camera controller provided by Unity must be removed and replaced with the XR Origin Camera. Additionally, the XR Origin must be equipped with two new components: AR Plane Manager and AR Raycast Manager.

The custom controller, AR Controller, contains a C# script that handles the placement of the 3D object and utilizes the Raycast Manager. Once the scripting and setup are complete, the app is built and securely deployed to a mobile device.

Components Required:

1) Unity Hub
2) 3D Object
3) Vs Code
4) Simulator or a Phone

Code

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

public class PlaceManager : MonoBehaviour
{
    private PlaceIndicator placeIndicator;
    public GameObject objectToPlace;
    private GameObject newPlacedObject;
    // Start is called before the first frame update
    void Start()
    {
        placeIndicator = FindObjectOfType<PlaceIndicator> ();
```

```
    }

    public void ClickToPlace() {
        newPlacedObject = Instantiate(objectToPlace, placeIndicator.transform.position,
placeIndicator.transform.rotation);
    }
}
```

Images of the app:

Challenges Overcome

1) Understanding raycast and plane manager was an interesting task
2) Understanding the Augmented Reality Spatial regions also gave insights.
3) A major benefit was that in the last lab assignment I had already added the feature of creating an image on tap of placement indicator thus the knowledge helped to perform this lab assignment.

“I have done this assignment on my own. I have not copied anything from another student or any online source. I understand if my work is found like somebody else’s code, my case can be sent to the Disciplinary committee of the institute for appropriate action”.

EXPERIMENT 3

Build an Augmented Reality application using Unity and insert multiple AR objects

SOLUTION

The augmented reality app was developed using Unity Hub and utilizes a free 3D object from the Unity Asset Store.The Unity scene includes 8 components: Directional Light, AR Session, XROrigin, AR Default, and AR Controller, Placement Indicator, Canvas controller for the 2 arrows and event manager.

In order to set up the AR environment, the default camera controller provided by Unity must be removed and replaced with the XR Origin Camera. Additionally, the XR Origin must be equipped with two new components: AR Plane Manager and AR Raycast Manager.

The custom controller, AR Controller, contains a C# script that handles the placement of the ARMultimodel objects and utilizes the Raycast Manager. Once the scripting and setup are complete, the app is built and securely deployed to a mobile device.

Components Required:

1) Unity Hub
2) 3D Object
3) Vs Code
4) Simulator or a Phone

The code script allows switching between multiple 3D models in the real world

using their mobile device. The script uses the ARFoundation and ARSubsystems of Unity to track surfaces in the environment and place virtual objects on them.

The script starts by initializing variables for the UI arrows, placement indicator, and the 3D models to be placed. It also sets up the ARRaycastManager and hides the UI arrows at the beginning.

In the Update() method, the script checks if the user has initiated a touch event to place a 3D model. If so, it calls the ARPlaceObject() method to place the selected model and displays the UI arrows. It then updates the placement pose and placement indicator on the screen.

The UpdatePlacementIndicator() method checks if a 3D model has not been placed yet and the placement pose is valid. If so, it activates the placement indicator at the position of the placement pose.

The UpdatePlacementPose() method calculates the placement pose of the virtual object based on the camera view and the detected surfaces in the environment.

The ARPlaceObject() method is called to place a selected 3D model. It first destroys any previously spawned models and then instantiates the selected model at the placement pose.

The ModelChangeRight() and ModelChangeLeft() methods are called when the user swipes left or right on the screen to switch between the available 3D models. They update the modelIndex variable and call the ARPlaceObject() method to place the selected model.

Code

```
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 arObjectToSpawn;
    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);

    }

    // need to update placement indicator, placement pose and spawn
    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);
    }

}
```

Images of the app:

Challenges Overcome

1) Having arrows to summon and change object in view was a task
2) Understanding the Augmented Reality Spatial regions also gave insights.
3) Understanding the ability to create a placement indicator with multiple objects is also interesting

“I have done this assignment on my own. I have not copied anything from another student or any online source. I understand if my work is found like somebody else’s code, my case can be sent to the Disciplinary committee of the institute for appropriate action

EXPERIMENT 4

Build an Augmented Reality application using Unity to Summon multiple AR objects

SOLUTION

The augmented reality app was developed using Unity Hub and utilizes a free 3D object from the Unity Asset Store.The Unity scene includes 8 components: Directional Light, AR Session, XROrigin, AR Default, and AR Controller, Placement Indicator, Canvas controller for the 2 arrows and event manager.

In order to set up the AR environment, the default camera controller provided by Unity must be removed and replaced with the XR Origin Camera. Additionally, the XR Origin must be equipped with two new components: AR Plane Manager and AR Raycast Manager.

The custom controller, AR Controller, contains a C# script that handles the placement of the ARMultimodel objects and utilizes the Raycast Manager. Once the scripting and setup are complete, the app is built and securely deployed to a mobile device.

Components Required:

1. Unity Hub
2. 3D Object
3. Vs Code
4. Simulator or a Phone

The code script allows switching between multiple 3D models in the real world

using their mobile device. The script uses the ARFoundation and ARSubsystems of Unity to track surfaces in the environment and place virtual objects on them.

The script starts by initializing variables for the UI arrows, placement indicator, and the 3D models to be placed. It also sets up the ARRaycastManager and hides the UI arrows at the beginning.

In the Update() method, the script checks if the user has initiated a touch event to place a 3D model. If so, it calls the ARPlaceObject() method to place the selected model and displays the UI arrows. It then updates the placement pose and placement indicator on the screen.

The UpdatePlacementIndicator() method checks if a 3D model has not been placed yet and the placement pose is valid. If so, it activates the placement indicator at the position of the placement pose.

The UpdatePlacementPose() method calculates the placement pose of the virtual object based on the camera view and the detected surfaces in the environment.

The ARPlaceObject() method is called to place a selected 3D model. It first destroys any previously spawned models and then instantiates the selected model at the placement pose.

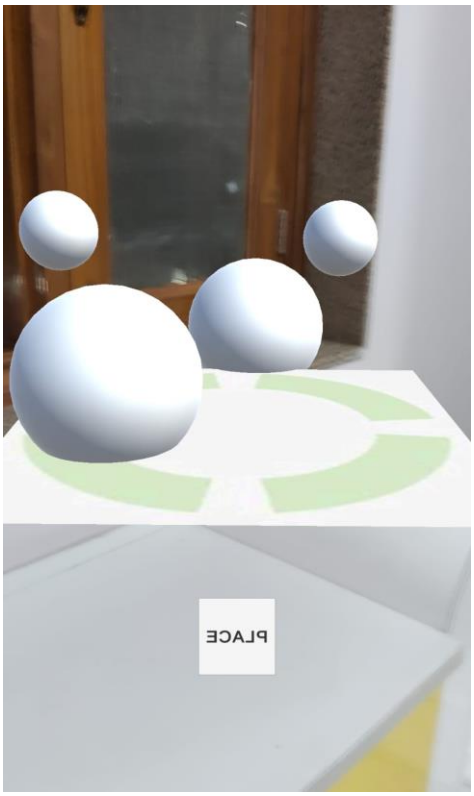
The ModelChangeRight() and ModelChangeLeft() methods are called when the user swipes left or right on the screen to switch between the available 3D models. They update the modelIndex variable and call the ARPlaceObject() method to place the selected model.

Code

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

public class PlaceIndicator : MonoBehaviour
{
    private ARRaycastManager raycastManager;
    private GameObject indicator;
    private List<ARRaycastHit> hits = new List<ARRaycastHit> ();

    // Start is called before the first frame update
    void Start()
    {
        raycastManager = FindObjectOfType<ARRaycastManager> ();
        indicator = transform.GetChild(0).gameObject;
        indicator.SetActive(false);
    }
```

```
    // Update is called once per frame
    void Update()
    {
        var ray = new Vector2(Screen.width / 2, Screen.height / 2);
        if(raycastManager.Raycast(ray, hits, TrackableType.Planes)) {
            Pose hitPose = hits[0].pose;
            transform.position = hitPose.position;
            transform.rotation = hitPose.rotation;

            if(!indicator.activeInHierarchy) {
                indicator.SetActive(true);
            }
        }
    }
}
```

Images of the app:



Challenges Overcome

1. Having arrows to summon and change object in view was a task
2. Understanding the Augmented Reality Spatial regions also gave insights.
3. Understanding the ability to create a placement indicator with multiple objects is also interesting

“I have done this assignment on my own. I have not copied anything from another student or any online source. I understand if my work is found like somebody else’s code, my case can be sent to the Disciplinary committee of the institute for appropriate action”.

EXPERIMENT 5

Build an Augmented Reality application using Unity and Use arrows as placement indicators to summon multiple AR objects

SOLUTION

The augmented reality app was developed using Unity Hub and utilizes a free 3D object from the Unity Asset Store.The Unity scene includes 8 components: Directional Light, AR Session, XROrigin, AR Default, AR Controller, Placement Indicator, Canvas controller for the 2 arrows and event manager.

In order to set up the AR environment, the default camera controller provided by Unity must be removed and replaced with the XR Origin Camera. Additionally, the XR Origin must be equipped with two new components: AR Plane Manager and AR Raycast Manager.

The custom controller, AR Controller, contains a C# script that handles the placement of the ARMultimodel objects and utilizes the Raycast Manager. Once the scripting and setup are complete, the app is built and securely deployed to a mobile device.

Components Required:

1. Unity Hub
2. 3D Object
3. Vs Code
4. Simulator or a Phone

The code script allows switching between multiple 3D models in the real world using their mobile device. The script uses the ARFoundation and ARSubsystems of Unity to track surfaces in the environment and place virtualobjects on them.

The script starts by initializing variables for the UI arrows, placement indicator, and the 3D models to be placed. It also sets up the ARRaycastManager and hides the UI arrows at the beginning.

In the Update() method, the script checks if the user has initiated a touch event to place a 3D model. If so, it calls the ARPlaceObject() method to place the selected model and displays the UI arrows. It then updates the placement pose and placement indicator on the screen.

The UpdatePlacementIndicator() method checks if a 3D model has not been placed yet and the placement pose is valid. If so, it activates the placement indicator at the position of the placement pose.

The UpdatePlacementPose() method calculates the placement pose of the virtual object based on the camera view and the detected surfaces in the environment.

The ARPlaceObject() method is called to place a selected 3D model. It first destroys any previously spawned models and then instantiates the selected model at the placement pose.

The ModelChangeRight() and ModelChangeLeft() methods are called when the user swipes left or right on the screen to switch between the available 3D models. They update the modelIndex variable and call the ARPlaceObject() method to place the selected model.

Code

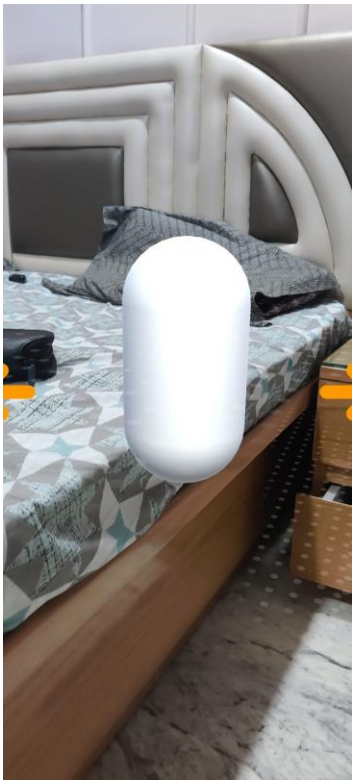
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;

public class ARPlacement : MonoBehaviour
{
    public GameObject UIArrows;

   // public GameObject arObjectToSpawn;
    public GameObject placementIndicator;
```

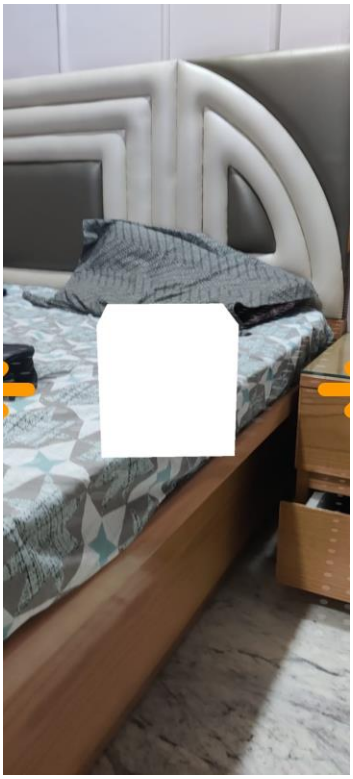
```
    private GameObject spawnedObject;
    private Pose PlacementPose;
    private ARRaycastManager aRRaycastManager;
    private bool placementPoseIsValid = false;

    public GameObject[] arModels;
    int modelIndex = 0;

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

    }

    // need to update placement indicator, placement pose and spawn
    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);
    }

}
```

Images of the app:





Challenges Overcome

4. Having arrows to summon and change object in view was a task
5. Understanding the Augmented Reality Spatial regions also gave insights.
6. Understanding the ability to create a placement indicator with multiple objects is also interesting

“I have done this assignment on my own. I have not copied anything from another student or any online source. I understand if my work is found like somebody else’s code, my case can be sent to the Disciplinary committee of the institute for appropriate action”.

EXPERIMENT 6

Build an Augmented Reality application using Unity –
1. Tracking an Image on plane to play video
2. Insert Image to track and video to play in Unity
3. Use Unity AR Foundation packages to play a video using AR Image Tracking

SOLUTION

The augmented reality app was developed using Unity Hub and utilizes a video and image from the video as objects. The Unity scene includes 3 components: Directional Light, AR Session, XR Origin with a video player object as the Prefab.

The new material widget for the image of the photo has been set with the same ratio as the plane view. The AR trackedImage manager utilizes the ReferenceImage Library and a Parent Prefab object.

Components Required:

1) Unity Hub
2) Video
3) Image Tracking
4) Vs Code
5) Simulator or a Phone

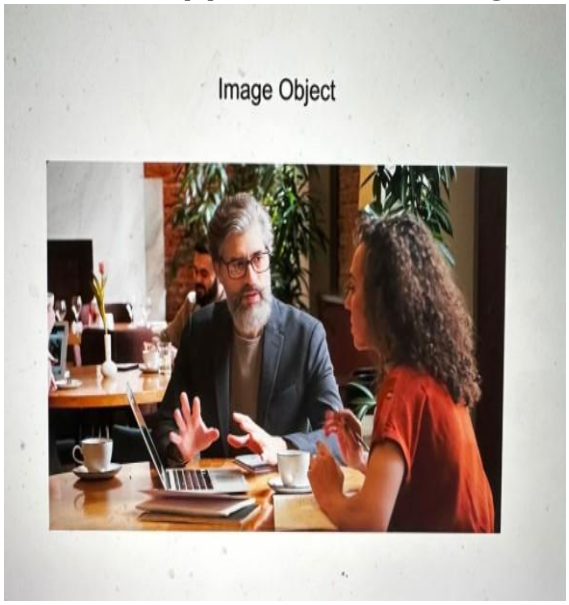
Steps

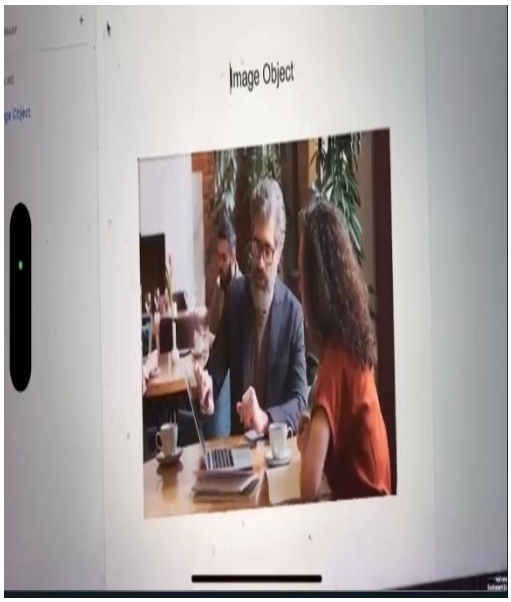
1. Create a unity project with AR Core foundation
2. Import the AR Foundation package into the project.
3. Enable the AR core kit and set the build setting for ios AR.
4. Create an image target from the video.
5. Find or create an image that you want to use as the target for your AR

experience. Make sure it has enough detail for the AR system to track it accurately.

6. Add an AR Session Origin object to the scene.
7. Add an AR Image Tracking object to the scene. This object will be used totrack the image target you created earlier.
8. Add a video player object to the scene and attach it to the plane object thatwill display the video when the image target is tracked. You can do this bydragging the video player object onto the plane object in the hierarchy, and then setting the Render Mode field in the Video Player component to "Material Override." Finally, drag the plane object onto the Material Override field.
9. Attach the AR Image Tracking object to the AR Session Origin object.
10. Drag the AR Image Tracking object onto the Tracking Reference field inthe AR Session Origin component to attach it to the AR Session Origin object.
11. Set the image target to track.
12. Select the AR Image Tracking object and set the Image Library field tothe image target created earlier.
13. The prefab of the AR object has the segment of the video player object.
14. Test and Build the AR app.

Screenshot of app and sheet image





Challenges Overcome

The main difficulty issues you may face while building this AR app are related to image tracking accuracy and video playback performance. Make sure to use a high-quality image target with enough detail for accurate tracking.

“I have done this assignment on my own. I have not copied anything from another student or any online source. I understand if my work is found like somebody else’s code, my case can be sent to the Disciplinary committee of the institute for appropriate action”.