Multi Image Tracking AR- Unity

Step 1: Create a New Project

- Open Unity Hub
- Go to Projects → New Project
- Select 3D (Built-in Render Pipeline Core) → Click Create Project

Step 2: Install Required Packages

- Go to Window → Package Manager
- In the top-left dropdown, select Unity Registry
- Search and install the following packages:
    - AR Foundation
    - ARCore XR Plugin

Step 3: Project Setup

- Go to File → Build Settings
    - Platform: Select Android
    - Click Switch Platform
- Then go to File → Build Settings → Player Settings → Other Settings
    - Uncheck "Auto Graphics API"
    - Remove "Vulkan" from Graphics APIs
    - Scroll down to Minimum API Level → Select Android 7.0 (Nougat) API Level 24
    - Under Configuration:
        - Scripting Backend → Select IL2CPP
        - Target Architecture → Check ARM64

Step 4: XR Plugin Setup

- Go to XR Plugin Management
    - Under Android, check ARCore

Step 5: Hierarchy Setup

- In the Hierarchy:

    - Right-click → XR → AR Session Origin
    - Right-click → XR → AR Session
    - Delete the Main Camera

Step 6: Organize Project Folders

Inside the Assets folder, create:

- Reference Image Library folder
- Material folder
- Textures folder
- Video folder
- Scripts folder
- Prefabs folder

Step 7: Import Assets

- In the Video folder → Right-click → Import New Asset → Import your .mp4 videos
- In the Textures folder → Import your images

Step 8: Create Materials

- In the Material folder → Create a new material
- In the Inspector:
    - Shader → Unlit → Texture
    - Assign your texture to Base Map
- Create one material for each image you want to track.

Step 9: Setup Reference Image Library

- Open the Reference Image Library folder
- Create a Reference Image Library: Assets → Create → XR → Reference Image Library
- Add images:
    - Assign a texture
    - Use the same name as the material
    - Specify Physical Size (e.g., X: 0.1777, Y: 0.1)

Step 10: Add 3D Object (Quad)

- In SampleScene → Right-click → 3D Object → Quad
    - Rename it (e.g., AvatarQuad)
    - Transform:
        - Rotation X:-90
        - Scale X: 0.178, Y: 0.1 (match reference image size)
- On the Quad:
    - Add Component → Video Player
        - Assign the respective Video Clip
        - Check Loop
    - Drag and drop the Material to apply the texture

Repeat this process for each image and video.

Step 11: Scripting

In the Scripts folder, create a C# script named MultiImageTrackingManager.cs and paste the following code:

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

public class MultipleImageTrackingManager : MonoBehaviour
{
    [SerializeField] private GameObject[] prefabsToSpawn;
    private ARTrackedImageManager _arTrackedImageManager;
    private Dictionary<string, GameObject> _arObjects;

    private void Awake()
    {
        _arTrackedImageManager =
GetComponent<ARTrackedImageManager>();
        _arObjects = new Dictionary<string, GameObject>();
    }

    private void Start()
    {
```

```
        _arTrackedImageManager.trackedImagesChanged +=
OnTrackedImageChanged;

        foreach (GameObject prefab in prefabsToSpawn)
        {
            GameObject newARObject = Instantiate(prefab,
Vector3.zero, Quaternion.identity);
            newARObject.name = prefab.name;
            newARObject.SetActive(false);
            _arObjects.Add(newARObject.name, newARObject);
        }
    }

    private void OnDestroy()
    {
        _arTrackedImageManager.trackedImagesChanged -=
OnTrackedImageChanged;
    }

    private void
OnTrackedImageChanged(ARTrackedImagesChangedEventArgs
eventArgs)
    {
        foreach (ARTrackedImage trackedImage in
eventArgs.added)
        {
            UpdateTrackedImage(trackedImage);
        }

        foreach (ARTrackedImage trackedImage in
eventArgs.updated)
        {
            UpdateTrackedImage(trackedImage);
        }

        foreach (ARTrackedImage trackedImage in
eventArgs.removed)
        {

_arObjects[trackedImage.referenceImage.name].SetActive(false);
        }
    }

    private void UpdateTrackedImage(ARTrackedImage
trackedImage)
    {
        if (trackedImage.trackingState ==
TrackingState.Limited || trackedImage.trackingState ==
TrackingState.None)
        {
```

```
_arObjects[trackedImage.referenceImage.name].SetActive(false);
            return;
        }

        if
(_arObjects.ContainsKey(trackedImage.referenceImage.name))
        {
            GameObject obj =
_arObjects[trackedImage.referenceImage.name];
            obj.SetActive(true);
            obj.transform.position =
trackedImage.transform.position;
        }
    }
}
```

Step 12: Assign Script and Prefabs

- Select AR Session Origin
    - Add Component → AR Tracked Image Manager
        - Assign Serialized Library → Your Reference Image Library
        - Set Max Number of Moving Images to 1
    - Drag and drop the MultiImageTrackingManager script
    - Drag and drop your prefab Quads to the Prefabs to Spawn array
- Create Prefab:
    - Drag your 3D Quads into the Prefabs folder
    - Delete the Quads from the Scene

Final Steps

- Save the Scene
- Build and Run on your Android device