**Aim: To understand Unity 3D Effects and 3D vectors**

a.) Develop a 3D environment with particle effects.

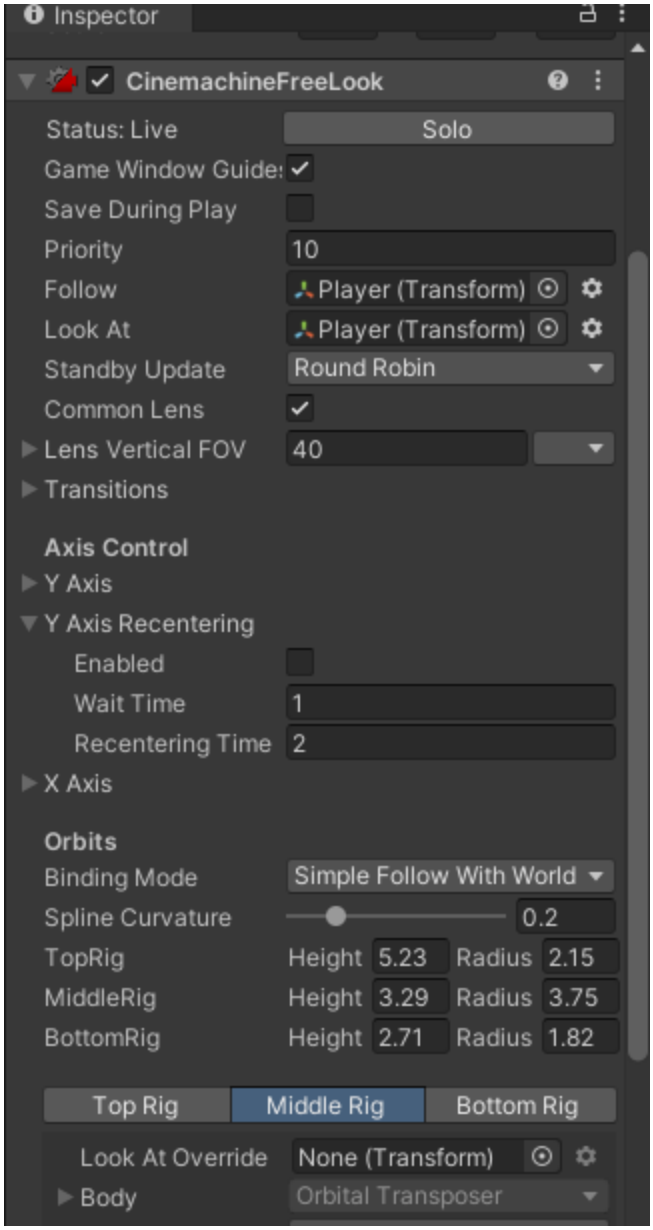
b.) Integrate a player character as a 3D model and enable transformative actions such as moving left/right and forward/backward.

**Solution**
**Part-A**

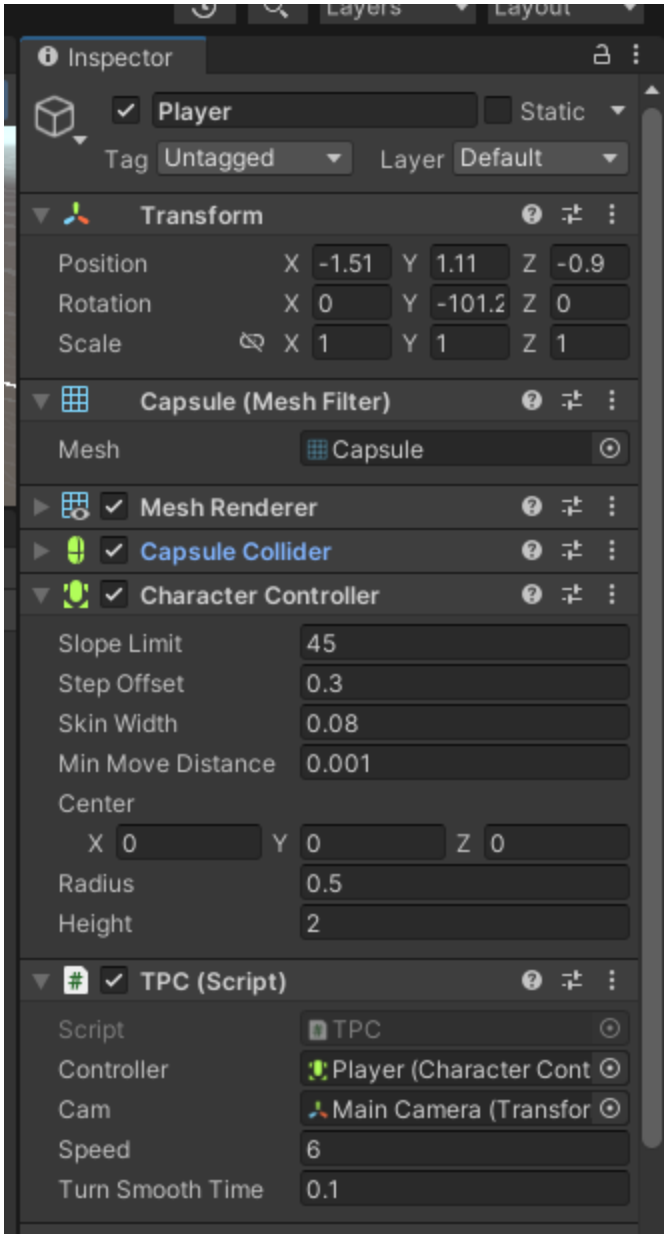
1. Add 3D Object plane and Player
2. Apply Material
    a. Smooth
    b. Metallic
    c. Rough
3. Add Particles and edit properties like
    a. Size
    b. Shape
    c. Duration
    d. Color
    e. Loop
    f. Delay

**Part -B**

1. Add free look camera from cinemachine and edit
    a. Rig
    b. Follow
    c. Look




2. Add character controller as component for player
3. Create c# script  and attach to player




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

public class TPC : MonoBehaviour
{
    public CharacterController controller;
    public Transform cam;
```

```
    public float speed = 6f;
    public float turnSmoothTime = 0.1f;
    float turnSmoothVelocity;

    void Update()
    {
        float horizontal = Input.GetAxisRaw("Horizontal");
        float vertical = Input.GetAxisRaw("Vertical");
        Vector3 direction = new Vector3(horizontal, 0f,
vertical).normalized;

        if (direction.magnitude >= 0.1f)
        {
            float targetAngle = Mathf.Atan2(direction.x, direction.z) *
Mathf.Rad2Deg + cam.eulerAngles.y;
            float angle = Mathf.SmoothDampAngle(transform.eulerAngles.y,
targetAngle, ref turnSmoothVelocity, turnSmoothTime);
            transform.rotation = Quaternion.Euler(0f, angle, 0f);

            Vector3 moveDir = Quaternion.Euler(0f, targetAngle, 0f) *
Vector3.forward;
            controller.Move(moveDir.normalized * speed *
Time.deltaTime);
        }
    }
}
```

LAB2:

**Aim: Showcase 3D modeling, UI/UX, and multimedia integration in AR.**

**Program**:

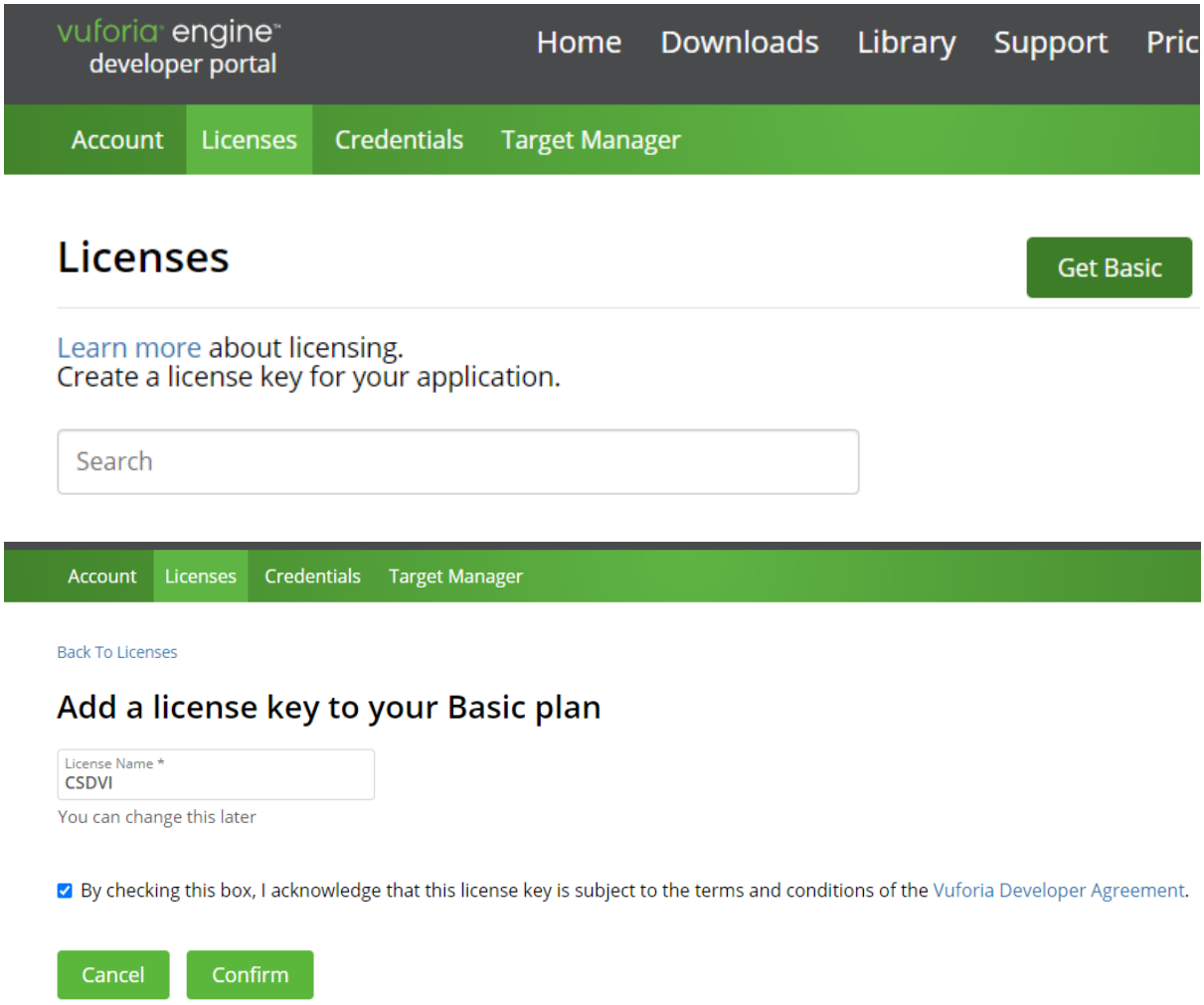
Create an AR application with Vuforia, incorporating 3D models, interactive features using UI buttons.

**Solution:**

AR VR lab 2

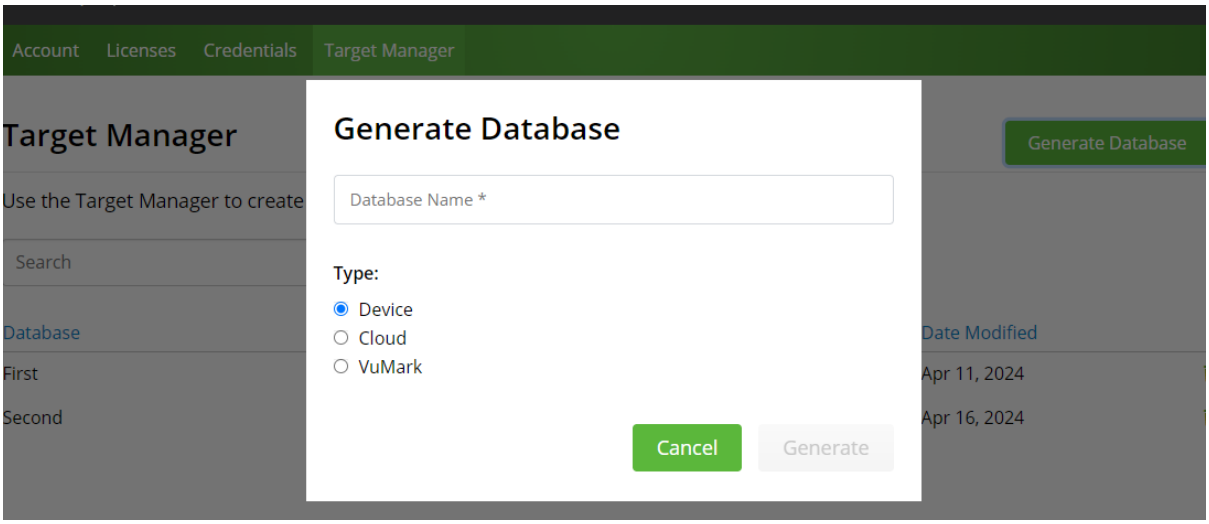
**Step 1: Vuforia Setup**:

* Create Vuforia account> Navigate to licenses> Get Basic -> click confirm  license key

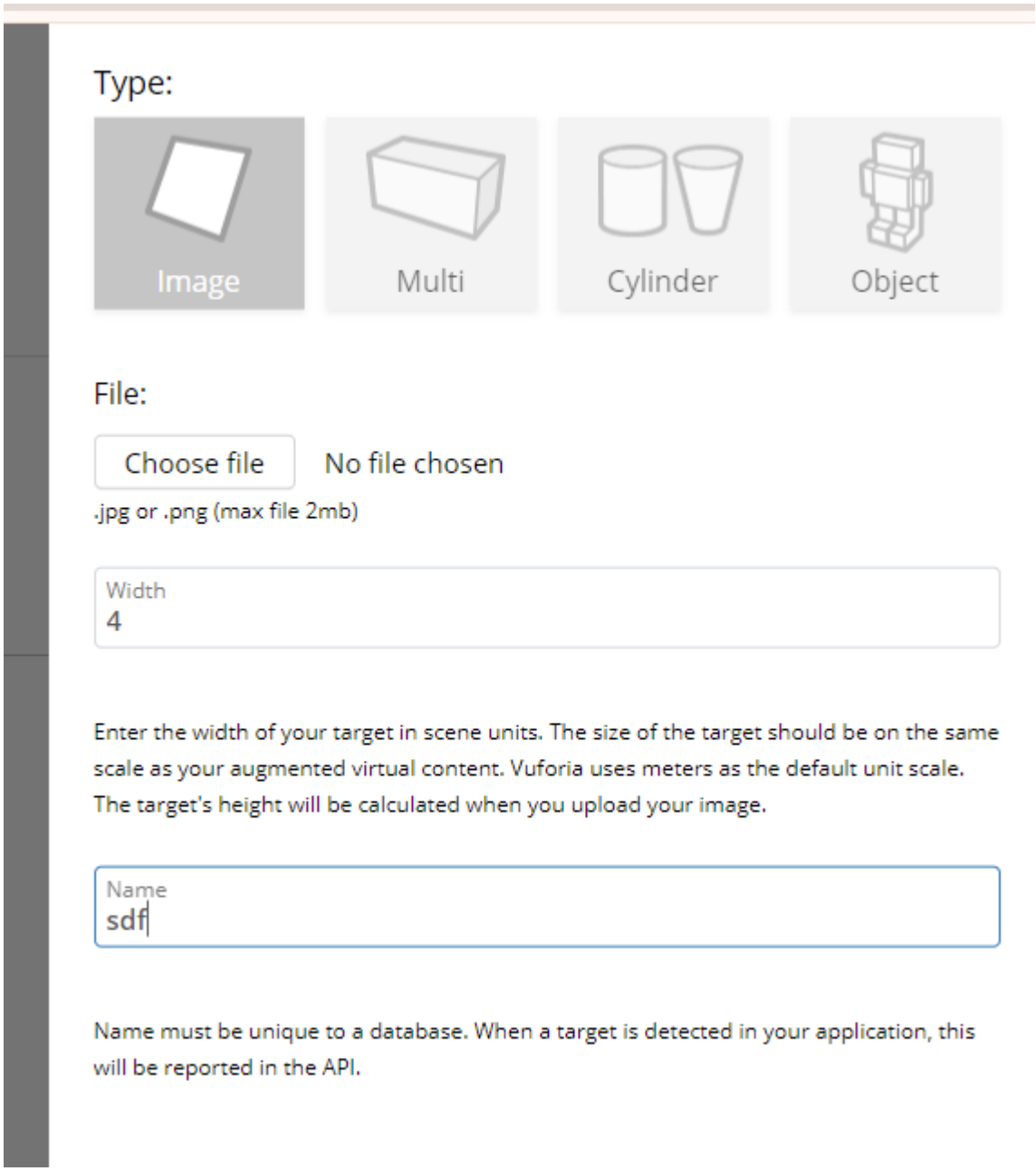



- Click on license>-> Copy key

❖ Click My account ->Target manager->Generate Database select Device

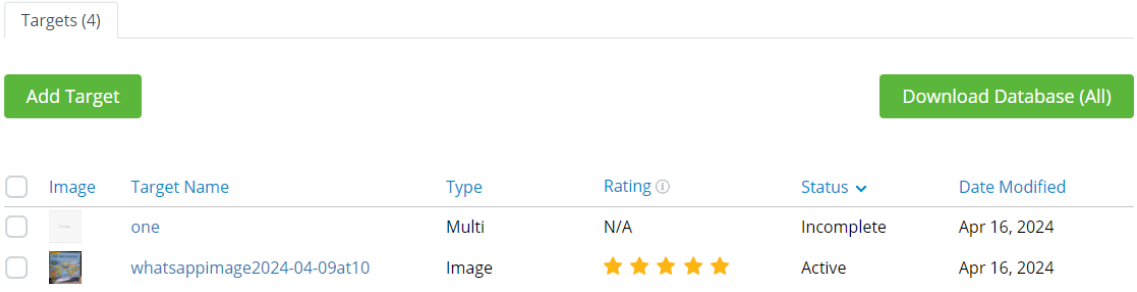



* Generate a Vuforia database to store image target




* load the target image sent on GCR or any img of choice

* Download the created database and vuforia engine for Unity(Link in Downloads Tab)

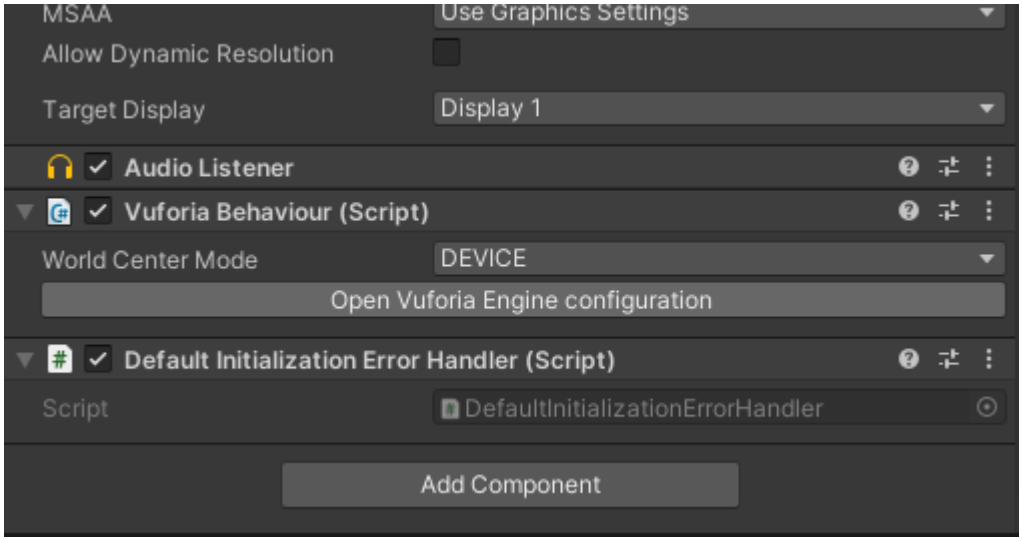

**Step 2: Import Vuforia package and Database in Unity:**

* create a new 3D core project in unity

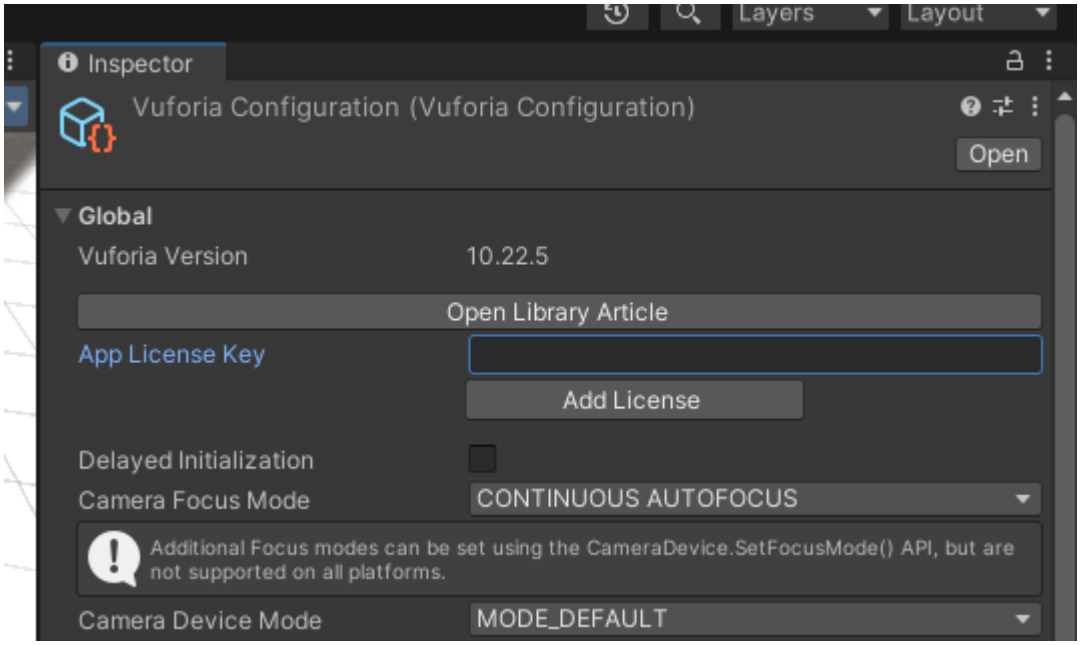
* add the vuforia package and downloaded database to unity project

* add AR camera by right clicking on hierarchy window (on Unity)[Vuforia Engine>AR camera]

* on AR camera properties: Navigate to OPEN VUFORIA ENGINE CONFIGURATION




* under license section: copy the license key copied from Vuforia

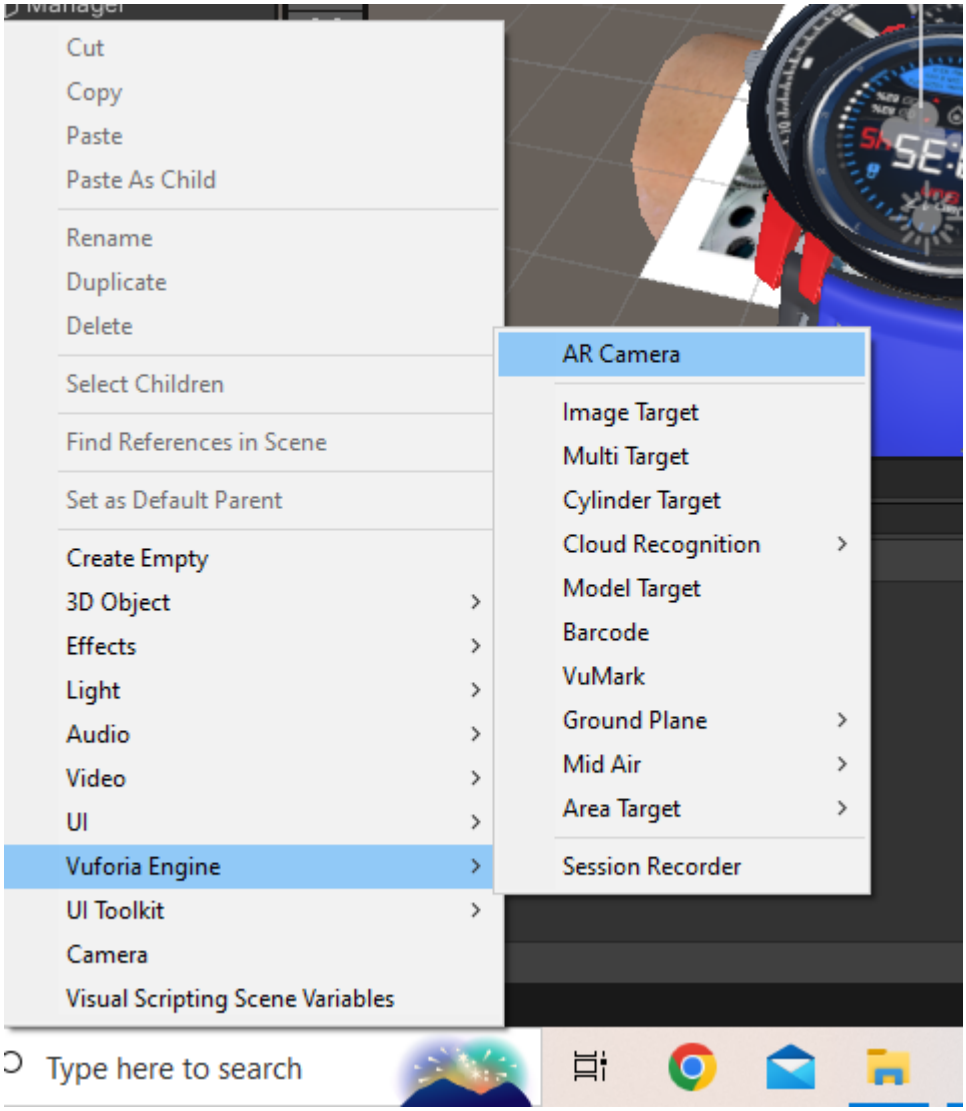



* right click on hierarchy> vuforia engine> image target> in inspector window

* image target type script> type: from database

Databse: name of vuforia database created earlier

Image target: Image loaded in database


Cut
Copy
Paste
Paste As Child
Rename
Duplicate
Delete
Select Children
Find References in Scene
Set as Default Parent
Create Empty
3D Object
Effects
Light
Audio
Video
UI
Vuforia Engine
UI Toolkit
Camera
Visual Scripting Scene Variables
AR Camera
Image Target
Multi Target
Cylinder Target
Cloud Recognition
Model Target
Barcode
VuMark
Ground Plane
Mid Air
Area Target
Session Recorder
Type here to search

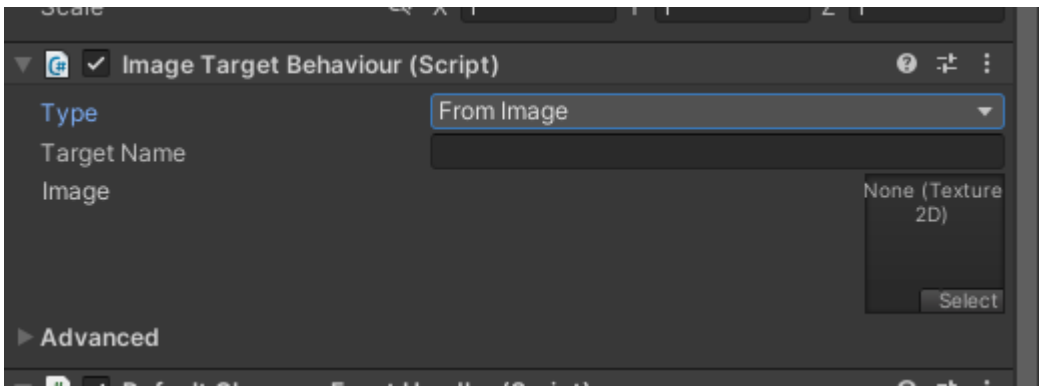
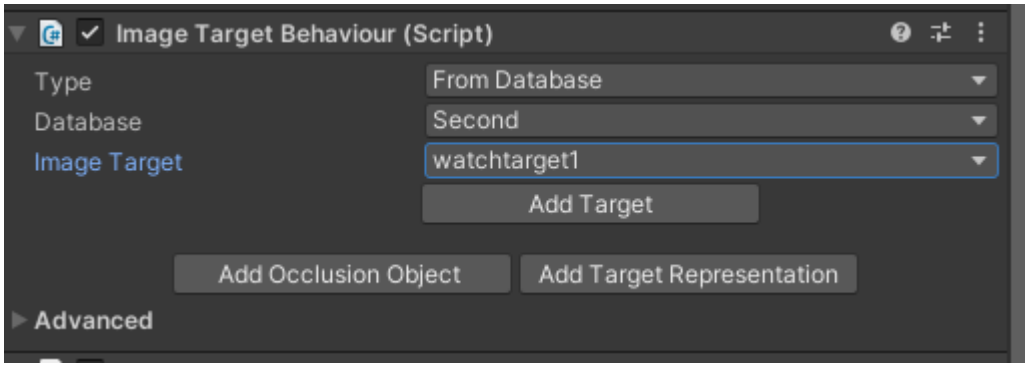


Image Target Behaviour (Script)
Type
From Image
Target Name
Image
None (Texture 2D)
Select
Advanced




* Download and extract files from Lec6 zip file uploaded on GCR and import folder to unity

* import the assets in Lec6 folder on unity and drag and drop the Watch 1 prefab to scene

* on hierachy window ensure to make the watch model as child component of image target

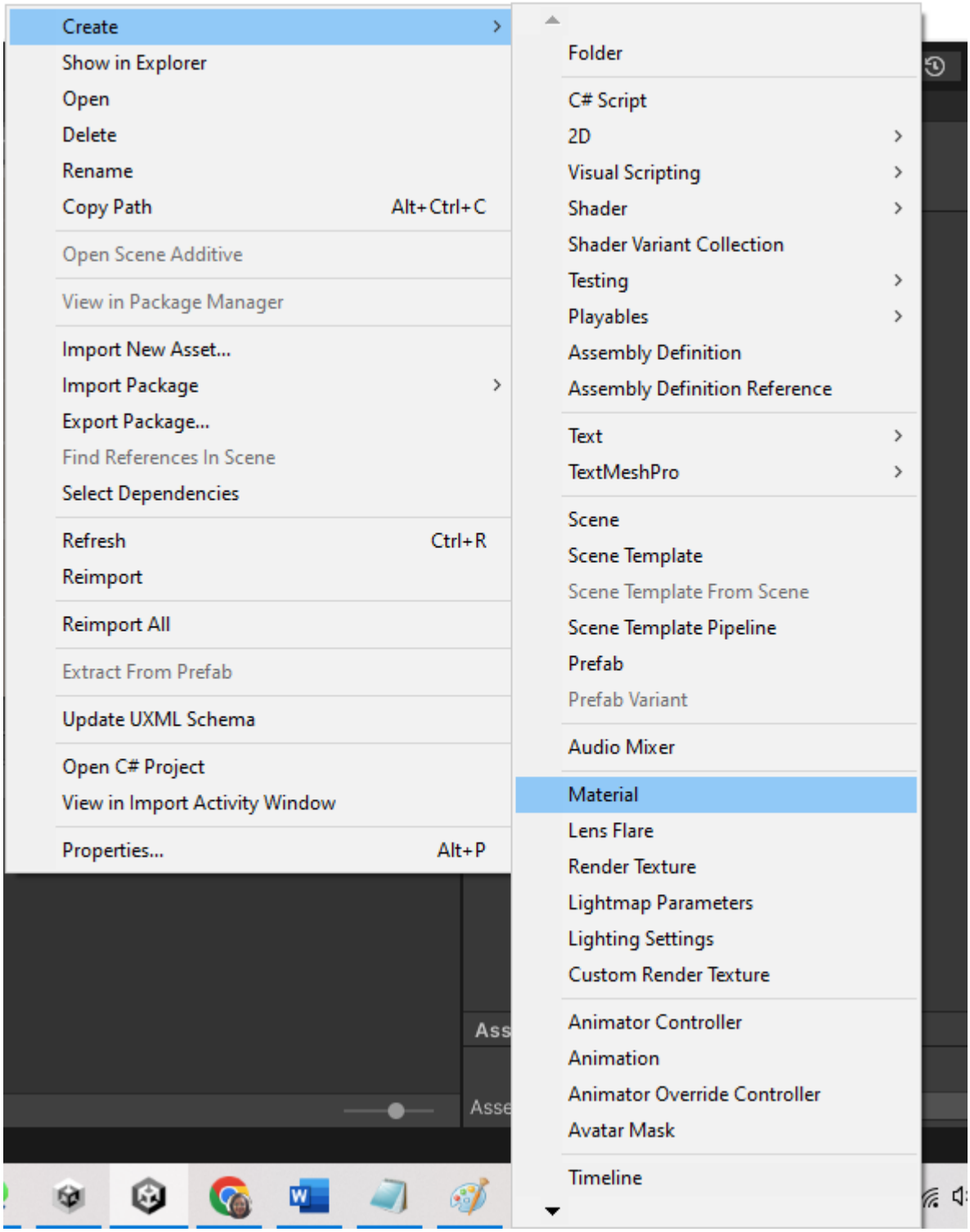
* shade the watch model using materials:

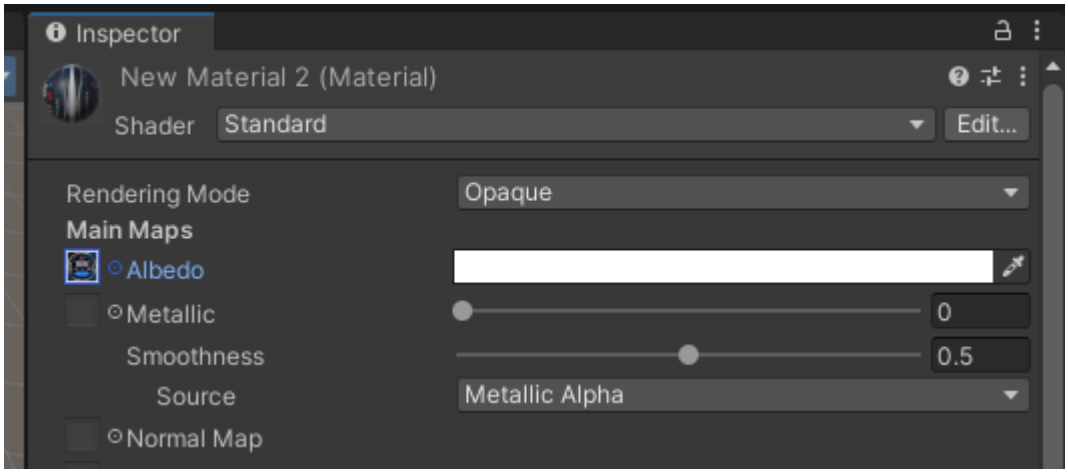
  buckle: any color

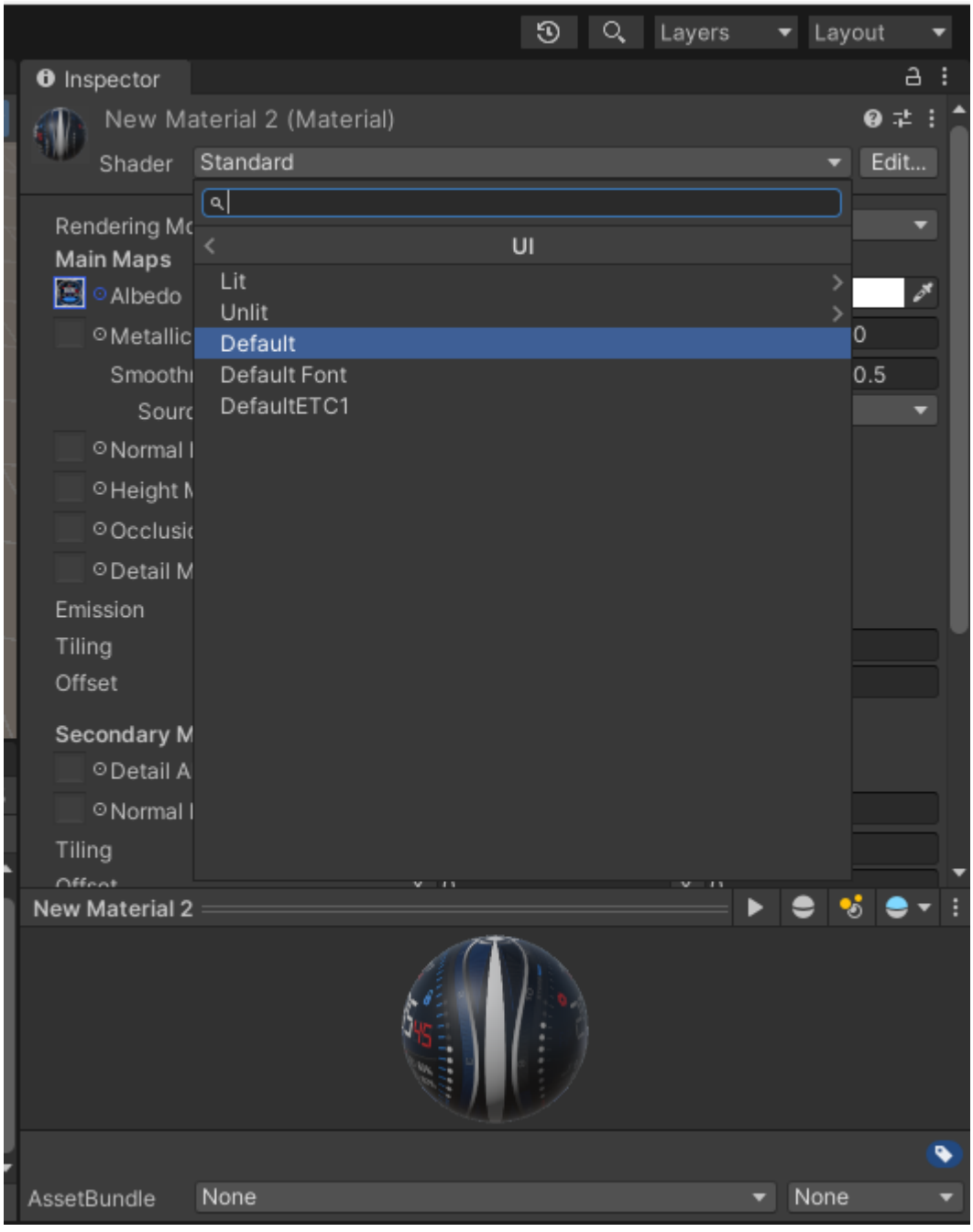
  case: black material with more smoothness and metallic texture

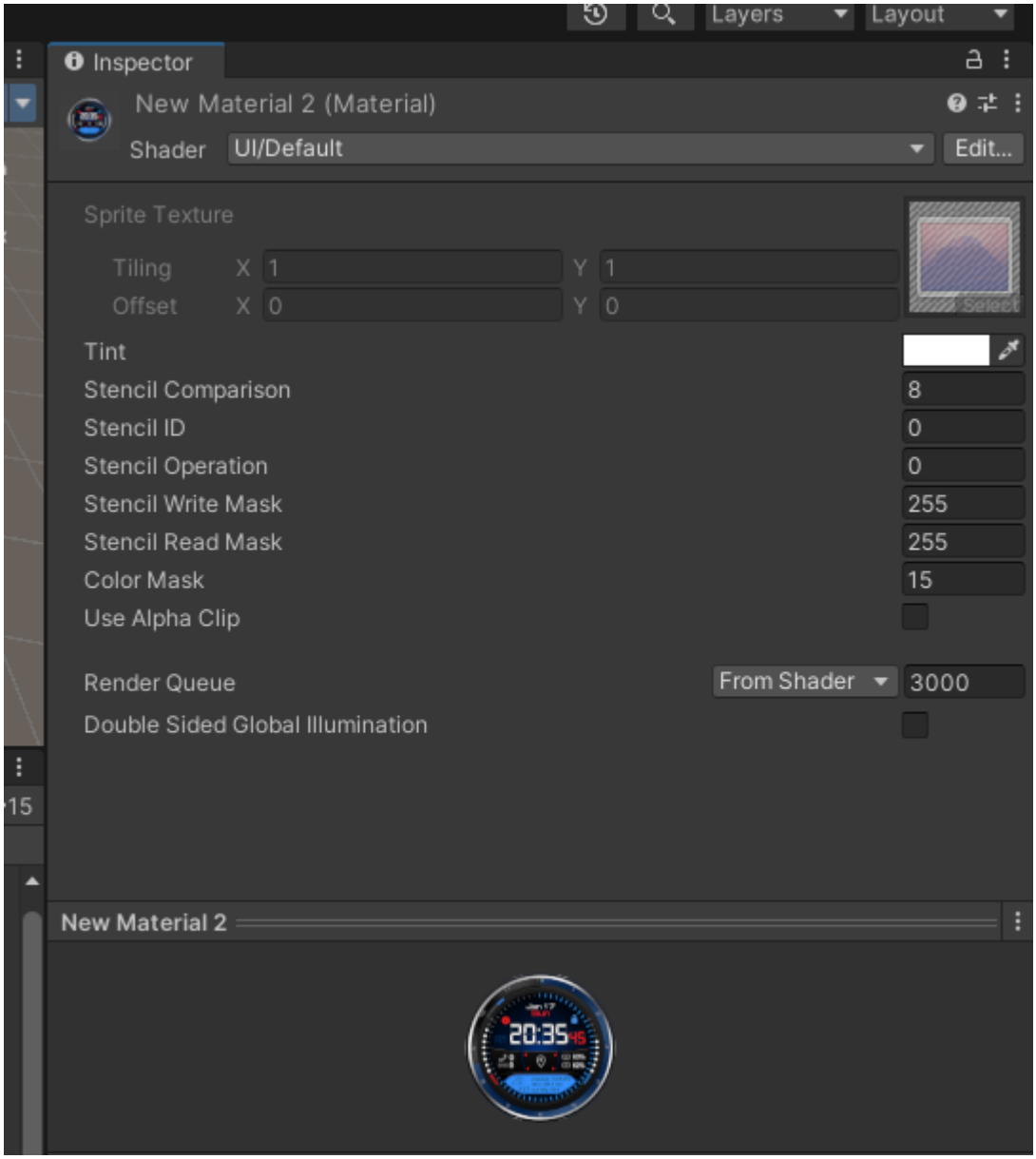
  glass: rendering mode: transparent albedo: alpha value= 0

  dial: textures> select dial image > shader: UI > default


Create
Show in Explorer
Open
Delete
Rename
Copy Path Alt+Ctrl+C
Open Scene Additive
View in Package Manager
Import New Asset...
Import Package
Export Package...
Find References In Scene
Select Dependencies
Refresh Ctrl+R
Reimport
Reimport All
Extract From Prefab
Update UXML Schema
Open C# Project
View in Import Activity Window
Properties... Alt+P
Folder
C# Script
2D
Visual Scripting
Shader
Shader Variant Collection
Testing
Playables
Assembly Definition
Assembly Definition Reference
Text
TextMeshPro
Scene
Scene Template
Scene Template From Scene
Scene Template Pipeline
Prefab
Prefab Variant
Audio Mixer
Material
Lens Flare
Render Texture
Lightmap Parameters
Lighting Settings
Custom Render Texture
Animator Controller
Animation
Animator Override Controller
Avatar Mask
Timeline


Inspector
New Material 2 (Material)
Shader
Standard
Edit...
Rendering Mode
Opaque
Main Maps
Albedo
Metallic
0
Smoothness
0.5
Source
Metallic Alpha
Normal Map


Layers
Layout
Inspector
New Material 2 (Material)
Shader
Standard
Edit...
UI
Lit
Unlit
Default
Default Font
DefaultETC1
Main Maps
Albedo
Emission
Tiling
Offset
Secondary M
Tiling
New Material 2
AssetBundle
None
None




* repeat the same steps to add other watch models

* create 2 UI buttons to change the watch model on click

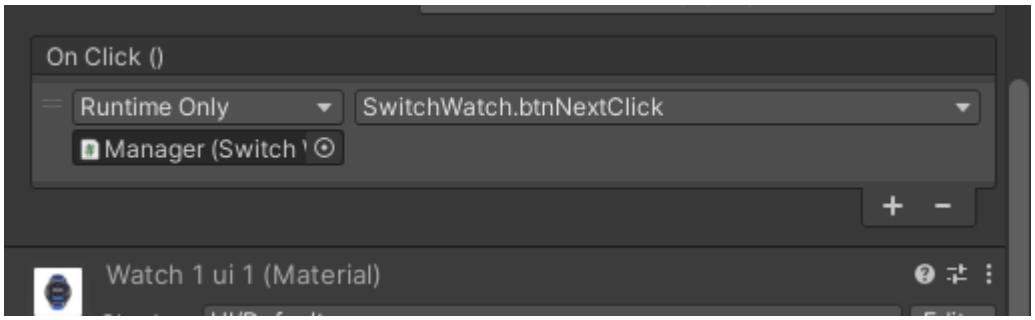
* attach script to buttons or create empty game object say manager

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

public class SwitchWatch : MonoBehaviour
{
    public GameObject obj1,obj2;
    // Start is called before the first frame update
    void Start()
    {
```

```
        obj1.SetActive(true);
        obj2.SetActive(false);
    }

    // Update is called once per frame
    public void btnNextClick()
    {
        obj1.SetActive(false);
        obj2.SetActive(true);
    }
    public void btnPrevClick()
    {
         obj1.SetActive(true);
        obj2.SetActive(false);
    }
}
```

* initialise the watches as gameobject references to the script

* add on click condition for both buttons and tag the respective functions




Testing:

* save and run project

* notice the webcam become active

* via a mobile phone, show the target image to webcam

* check if watch model is visible

* test buttons to check if watch models change on button click

Imp Links:

https://www.youtube.com/watch?v=RMOMTyfECTk

https://www.youtube.com/watch?v=-bF0oxgtt6A

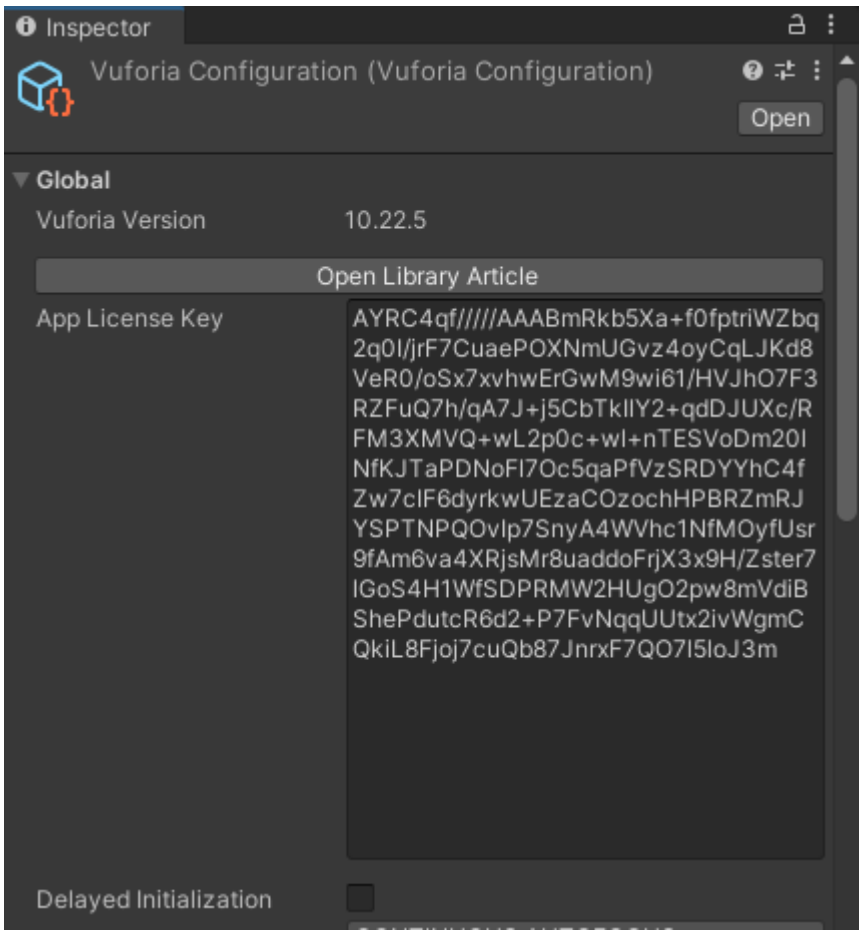
LAB:3

Develop AR applications using image targets and design suitable UI for watch-related experiences based on UX guidelines.

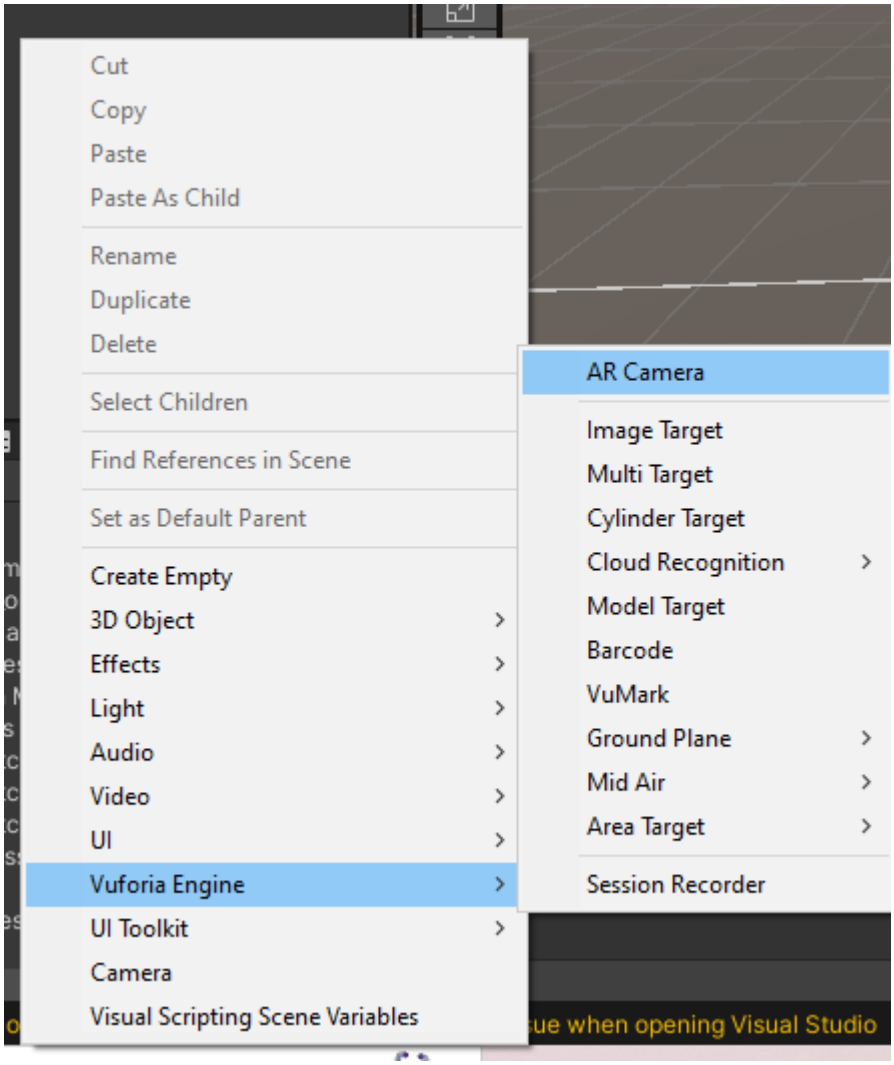
Solutions:

Steps:

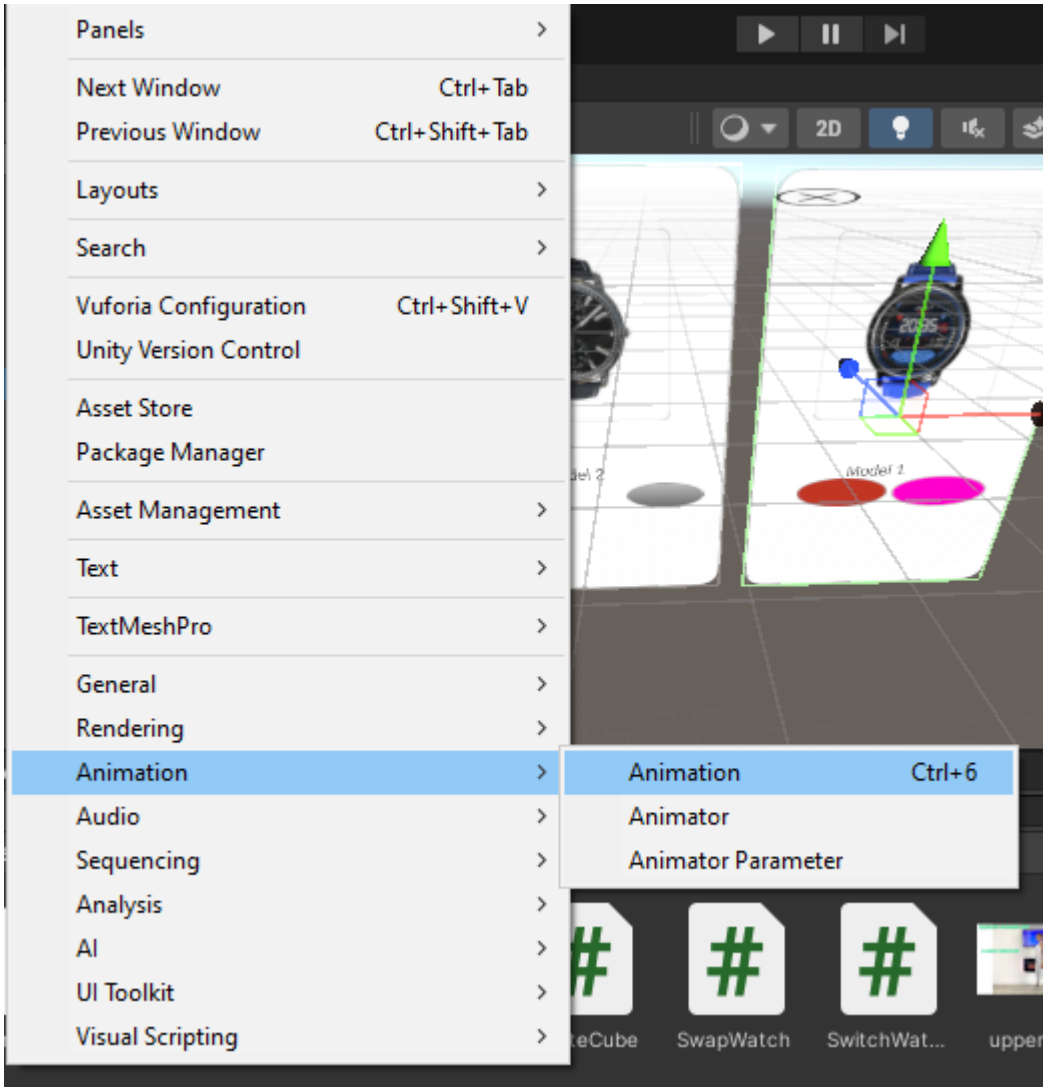
1. Create Vuforia database add Image target
2. Download database
3. Add the Vuforia to Unity engine by adding licence key




4. Add AR Camera
5. Add Image Target

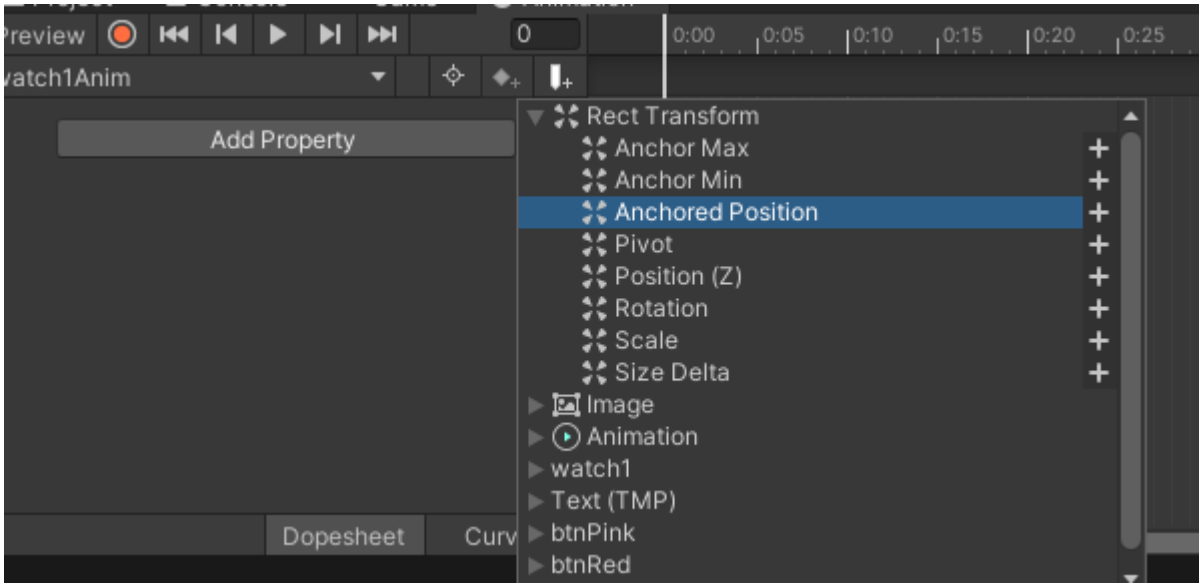



6. Add 3D Model watch on the scene as previous lab
7. Create two Windows for animation
    a. Right click add image
    b. On image add text add cost of watch
    c. Animate window




Name animation as `watch1Anim`

Select image(watch window) and click add property select React Transform->position




Select image and animate window for both image

8.Create c# script name it SwapWatch

```
using UnityEngine;
using UnityEngine.EventSystems;
public class SwapWatch : MonoBehaviour
{
    public GameObject[] Watchwindow;
    public GameObject[] WatchModel;
        Animation anim1,anim2;
    // Start is called before the first frame update
    void Start()
    {
        WatchModel[0].SetActive(true);
        Watchwindow[0].SetActive(true);
        WatchModel[1].SetActive(false);
        Watchwindow[1].SetActive(false);
        //get component of animation
        anim1=Watchwindow[0].GetComponent<Animation>();
        anim2=Watchwindow[1].GetComponent<Animation>();

    }

    public void Watch1()
    {
        WatchModel[0].SetActive(true);
        Watchwindow[0].SetActive(true);
        WatchModel[1].SetActive(false);
        Watchwindow[1].SetActive(false);
        anim1["watch1Anim"].speed=1.0f;
        anim1.Play();
```

```
    }
    public void watch2()
    {
        WatchModel[0].SetActive(false);
        Watchwindow[0].SetActive(false);
        WatchModel[1].SetActive(true);
        Watchwindow[1].SetActive(true);
        anim2["watch2Anim"].speed=1.0f;
        anim2.Play();
    }
    public void buttonCloseClick()
    {
        string
buttonName=EventSystem.current.currentSelectedGameObject.name;
        if(buttonName=="close1")
        {
            //reverse of close animation
            anim1["watch1Anim"].speed=-1.0f;
            anim1["watch1Anim"].time=
            anim1["watch1Anim"].length;
            anim1.Play();
        }
        else if(buttonName=="close2")
        {
            anim2["watch2Anim"].speed=-1.0f;
            anim2["watch2Anim"].time=
            anim2["watch2Anim"].length;
            anim2.Play();
        }
    }


}
```

8. Create empty game object manager and attach script to manager
9. Give reference of watch and window in script
10. Run and check output  on laptop camera or build apk and check in mobile

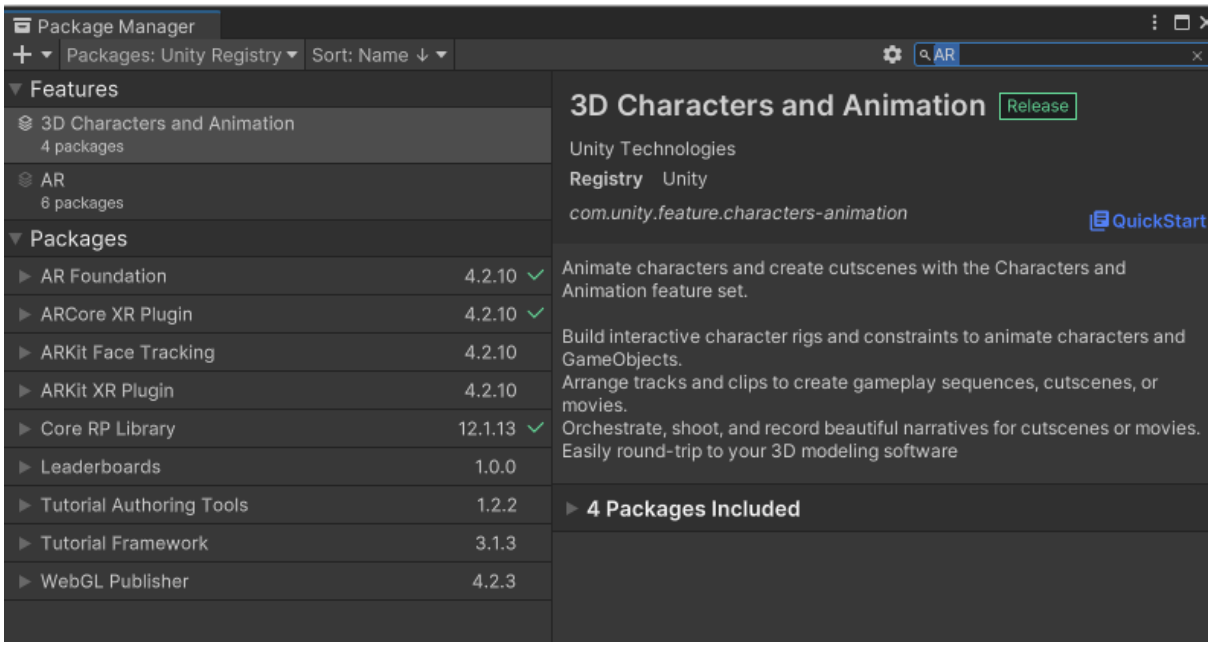
Note: In exam building apk is compulsory

Lab-4:

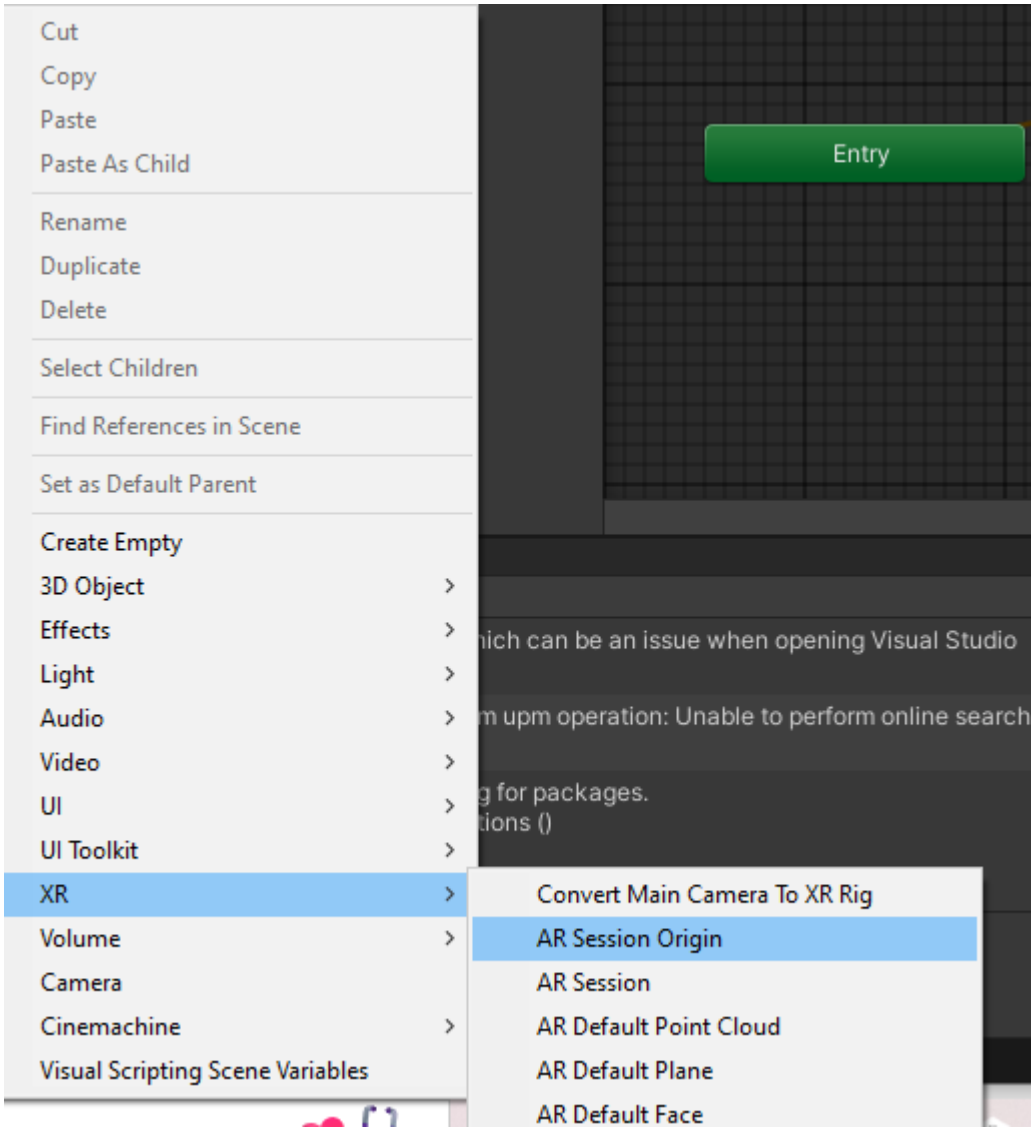
Create an AR application that animates characters in the real world, triggered by user interactions

Solution Steps:

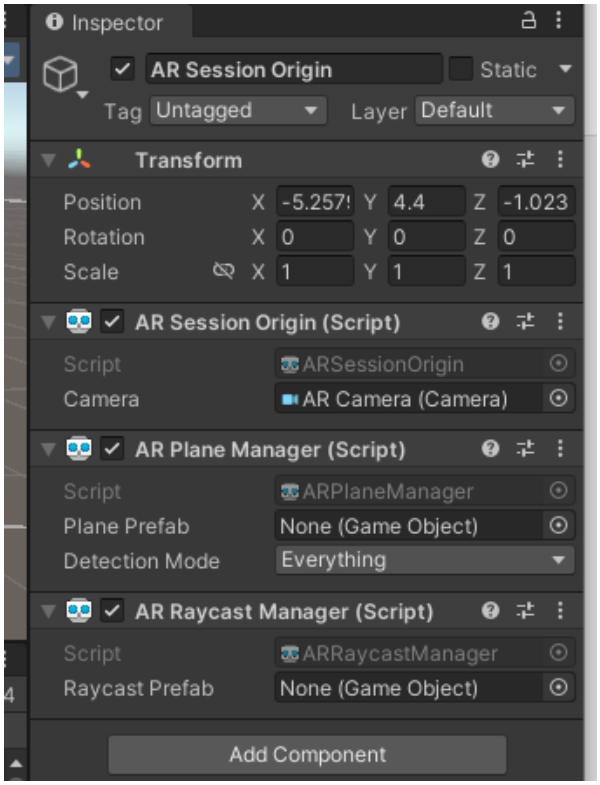
1. Windows-> Package manager->install ARcore XR Plugin
2. AR Foundation



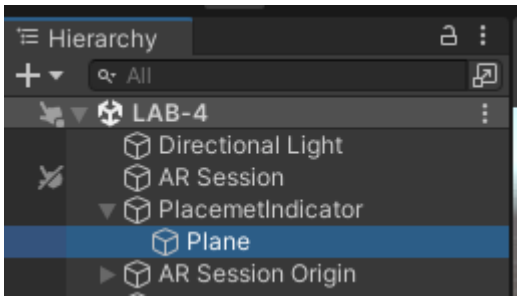

3. Right Click- on hierarchy window> Add AR session origin or XR Session Origin
4. AR/XR session




5. Delete main camera
6. Select AR Session origin or XR origin and add component AR Raycast,AR plane Manager



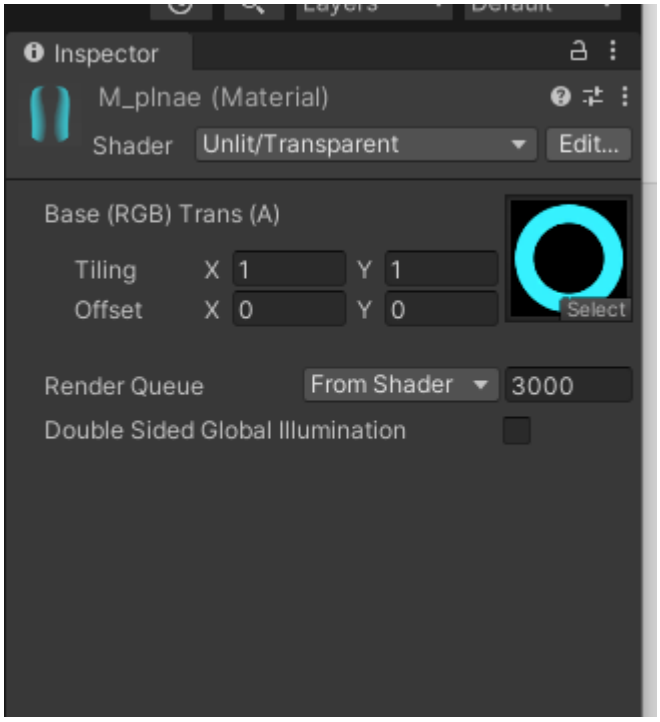

9. Create Empty Game Object placement and create plane as child of this




10. select plane and remove mesh collider component

11. import 2D image in unity

12. create material give this image as reference in albedo and change shader unlit transparent




13. add material to plane

14 Import 3D model from Maximo (https://www.mixamo.com/ ) and import two animation idle and run

15. Add model in scene and create animator

16. create animation graph

17 add animator controller as component to 3D model and create it as prefab by dragging it in project window and delete it from hierarchy window

18. Create empty game object SpawnManager

19 create script `PlacementIndicator` attach to placement

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

public class PlacementIndicator : MonoBehaviour
{
        private ARRaycastManager rayManager;
    private GameObject visual;

    // Start is called before the first frame update
    void Start()
    {
        rayManager = FindObjectOfType<ARRaycastManager>();
        visual = transform.GetChild(0).gameObject;
        //hide placement indicator
```

```
        visual.SetActive(false);
    }

    // Update is called once per frame
    void Update()
    {
        List<ARRaycastHit> hits = new List<ARRaycastHit>();
        //shoot raycast from center of screen
        rayManager.Raycast(new Vector2(Screen.width / 2, Screen.height / 2),
         hits, TrackableType.Planes);
        //if we hit AR plane update position and rotation
        if (hits.Count > 0)
        {
            transform.position = hits[0].pose.position;
            transform.rotation = hits[0].pose.rotation;
            if (!visual.activeInHierarchy)
                visual.SetActive(true);
        }
    }
}
```

20 create script spawn_object attach to SpawnManager

```
using UnityEngine;

public class spawn_object : MonoBehaviour
{
   public GameObject objectToSpawn;
    private PlacementIndicator PlaceIndicate;

    void Start()
    {
        PlaceIndicate = FindObjectOfType<PlacementIndicator>();

    }

    // Update is called once per frame
    void Update()
    {


        if (Input.touchCount > 0 &&
```

```
        Input.touches[0].phase == TouchPhase.Began)
        {
            showobject();
        }

    }
    void showobject()
    {
        GameObject obj = Instantiate(objectToSpawn,
PlaceIndicate.transform.position,
        PlaceIndicate.transform.rotation);
    }

}
```

21. create reference in spawn_object for 3D model

22. build apk and test application in mobile

Note: your mobile must support Google ARcore

LAB-5

Create an AR application that animates characters in the real world, triggered by user interactions on some algorithm

Solution:

Here perceptron algorithm is used.

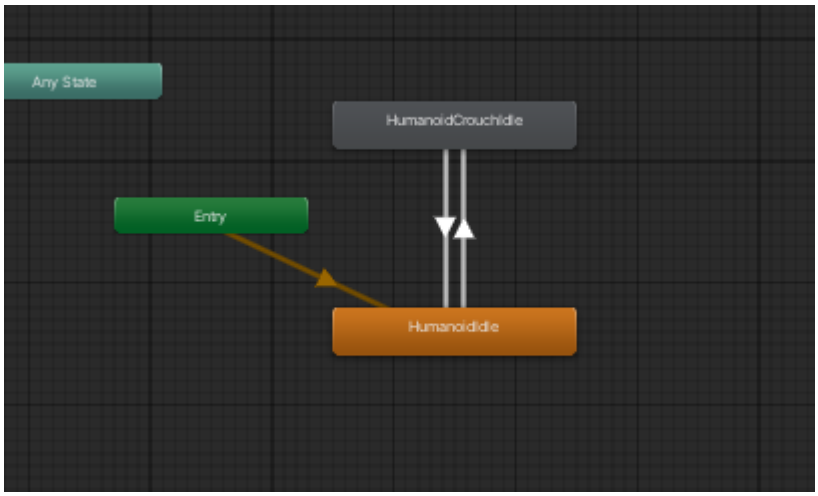
Steps 1. Add 3D mesh(model) from Maximo download two different animations

//here one animation is default and other model learn when read ball will be generated

2. Add model in scene name it as player

3. Create animator controller and attach it to player

4. Create parameter of type trigger and change transition




5. select player and add component rigid body

6. create two c# script (perceptron and Throw) and attach to player

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

[System.Serializable]
public class TrainingSet
{
    public double[] input;
    public double output;
}

public class Perceptron : MonoBehaviour {

    List<TrainingSet> ts = new List<TrainingSet>();
    double[] weights = {0,0};
    double bias = 0;
    double totalError = 0;
```

```
    //public GameObject npc;
    public Rigidbody rb;
    public Animator anim;
    void Start ()
    {
        rb=GetComponent<Rigidbody>();
        anim=GetComponent<Animator>();
        InitialiseWeights();
    }
    public void SendInput(double i1, double i2, double o)
    {
        //react
        double result = CalcOutput(i1, i2);
        Debug.Log(result);
        if(result == 0)
        {
            anim.SetTrigger("Crouch");
            rb.isKinematic=false;
        }
        else
        {
            rb.isKinematic = true;
        }

        //learn from it for next time
        TrainingSet s = new TrainingSet();
        s.input = new double[2] {i1,i2};
        s.output = o;
        ts.Add(s);
        Train();
    }

    double DotProductBias(double[] w1, double[] v2)
    {
        if (w1 == null || v2 == null)
            return -1;

        if (w1.Length != v2.Length)
            return -1;

        double y = 0;
        for (int x = 0; x < w1.Length; x++)
        {
            y += w1[x] * v2[x];
        }

        y += bias;
```

```
        return y;
    }

    double CalcOutput(int i)
    {
        return(ActivationFunction(DotProductBias(weights,ts[i].input)));
    }

    double CalcOutput(double i1, double i2)
    {
        double[] inp = new double[] {i1, i2};
        return(ActivationFunction(DotProductBias(weights,inp)));
    }

    double ActivationFunction(double y)
    {
        if(y > 0) return (1);
        return(0);
    }

    void InitialiseWeights()
    {
        for(int i = 0; i < weights.Length; i++)
        {
            weights[i] = Random.Range(-1.0f,1.0f);
        }
        bias = Random.Range(-1.0f,1.0f);
    }

    void UpdateWeights(int j)
    {
        double error = ts[j].output - CalcOutput(j);
        totalError += Mathf.Abs((float)error);
        for(int i = 0; i < weights.Length; i++)
        {
            weights[i] = weights[i] + error*ts[j].input[i];
        }
        bias += error;
    }

    void Train()
    {
        for(int t = 0; t < ts.Count; t++)
        {
            UpdateWeights(t);
        }
    }
```

```
    void Update () {
        if(Input.GetKeyDown("space"))
        {
            InitialiseWeights();
            ts.Clear();
        }
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
public class Throw : MonoBehaviour {

    public GameObject spherePrefab;
    public GameObject cubePrefab;
    public Material green;
    public Material red;

    Perceptron p;

    // Use this for initialization
    void Start () {
        p = GetComponent<Perceptron>();
    }
    public void inputButton()
    {
         string
buttonName=EventSystem.current.currentSelectedGameObject.name;
         if(buttonName=="1")
         {
            GameObject g =
Instantiate(spherePrefab,Camera.main.transform.position,Camera.main.transfor
m.rotation);
            g.GetComponent<Renderer>().material = red;
            g.GetComponent<Rigidbody>().AddForce(0,0,500);
            p.SendInput(0,0,0);
         }
         else if(buttonName=="2")
        {
```

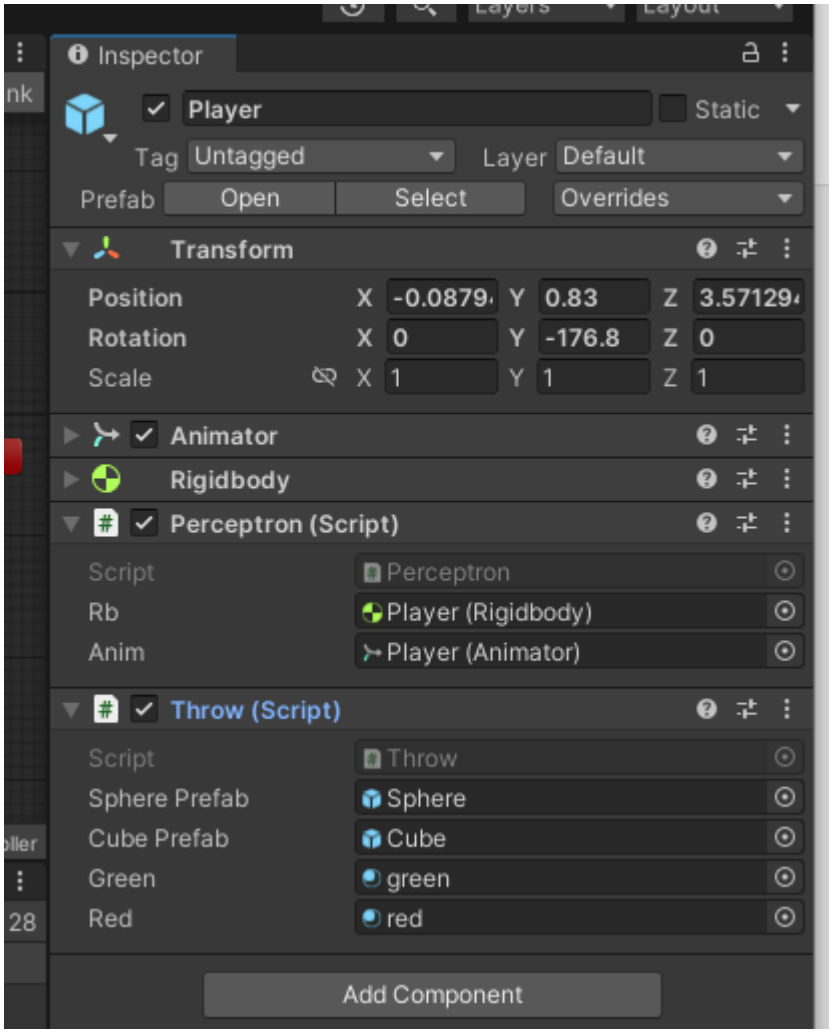
```
            GameObject g =
Instantiate(spherePrefab,Camera.main.transform.position,Camera.main.transfor
m.rotation);
            g.GetComponent<Renderer>().material = green;
            g.GetComponent<Rigidbody>().AddForce(0,0,500);
            p.SendInput(0,1,1);
        }
        else if(buttonName=="3")
        {
            GameObject g =
Instantiate(cubePrefab,Camera.main.transform.position,Camera.main.transform.
rotation);
            g.GetComponent<Renderer>().material = red;
            g.GetComponent<Rigidbody>().AddForce(0,0,500);
            p.SendInput(1,0,1);
        }
        else if(buttonName=="4")
        {
            GameObject g =
Instantiate(cubePrefab,Camera.main.transform.position,Camera.main.transform.
rotation);
            g.GetComponent<Renderer>().material = green;
            g.GetComponent<Rigidbody>().AddForce(0,0,500);
            p.SendInput(1,1,1);
        }
    }
    // Update is called once per frame
    void Update () {
        if(Input.GetKeyDown("1"))
        {
            GameObject g =
Instantiate(spherePrefab,Camera.main.transform.position,Camera.main.transfor
m.rotation);
            g.GetComponent<Renderer>().material = red;
            g.GetComponent<Rigidbody>().AddForce(0,0,500);
            p.SendInput(0,0,0);
        }
        else if(Input.GetKeyDown("2"))
        {
            GameObject g =
Instantiate(spherePrefab,Camera.main.transform.position,Camera.main.transfor
m.rotation);
            g.GetComponent<Renderer>().material = green;
            g.GetComponent<Rigidbody>().AddForce(0,0,500);
            p.SendInput(0,1,1);
        }
        else if(Input.GetKeyDown("3"))
        {
```

```
            GameObject g =
Instantiate(cubePrefab,Camera.main.transform.position,Camera.main.transform.
rotation);
            g.GetComponent<Renderer>().material = red;
            g.GetComponent<Rigidbody>().AddForce(0,0,500);
            p.SendInput(1,0,1);
        }
        else if(Input.GetKeyDown("4"))
        {
            GameObject g =
Instantiate(cubePrefab,Camera.main.transform.position,Camera.main.transform.
rotation);
            g.GetComponent<Renderer>().material = green;
            g.GetComponent<Rigidbody>().AddForce(0,0,500);
            p.SendInput(1,1,1);
        }
    }
}
```

7. create two prefab cube and sphere

9. create two materials red and green

10. Attach prefab, rigidbody, material two scripts




11. Run the application and test model change animation at red ball(sphere)

LAB-6

Create a GPS AR Camera and As per location detected 3D model should pop up.

To create Express server

1. Create Project

2. mkdir my-aframe-project
3. cd my-aframe-project
4. npm init -y
5. create folder public inside project flder

2. Install express

```
npm install express
```

3. Create server.js script for creating server

```
const express = require('express');
const path = require('path');
const app = express();

// Serve static files from the "public" directory
app.use(express.static(path.join(__dirname, 'public')));

// Start the server
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

4. Create Location.html

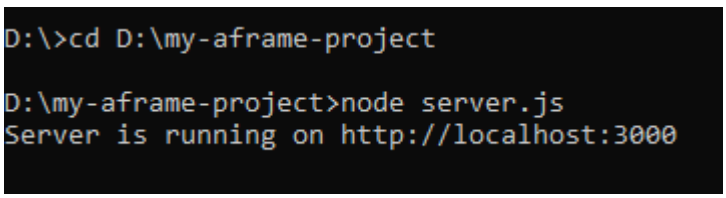
```
        <!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>A-Frame Scene</title>
    <script src="https://aframe.io/releases/1.2.0/aframe.min.js"></script>
    <script>
      document.addEventListener('DOMContentLoaded', function() {
        // Function to handle success in getting location
        function showCube(position) {
          var currentLatitude = position.coords.latitude;
          var currentLongitude = position.coords.longitude;
          console.log("Latitude: " + currentLatitude);
```

```
        console.log("Longitude: " + currentLongitude);

        // Make the cube visible by setting its `visible` attribute to true
             const targetLatitude = 12.9096941;
        const targetLongitude = 77.5733936;
             const threshold = 0.0001;
        var cube = document.querySelector('#locationCube');
             if (Math.abs(currentLatitude - targetLatitude) < threshold &&
Math.abs(currentLongitude - targetLongitude) < threshold)
            {
        cube.setAttribute('visible', true);
            }
      }

      // Function to handle error in getting location
      function locationError(error) {
       console.error("Error getting location: ", error);
      }

      // Get current position
      navigator.geolocation.getCurrentPosition(showCube, locationError);
     });
   </script>
  </head>
  <body>
   <a-scene>
    <a-box id="locationCube" position="0 0.5 -2" rotation="0 45 0" color="#4CC3D9"
visible="false"></a-box>
     <a-camera gps-camera rotation-reader></a-camera>
   </a-scene>
  </body>
</html>
```

5. Start Your Server

```
node server.js
```

//here you need to use command prompt ->locate your folder and run server




6. Type http://localhost:3000/location.html in the browser to run application

LAB-7

Create a Marker less AR to display 3D model in GLTF Format in website or web-based application

To create Express server

1. Create Project

2. mkdir my-aframe-project
3. cd my-aframe-project
4. npm init -y
5. create folder public inside project flder

2. Install express

```
npm install express
```

3. Create server.js script for creating server

```
const express = require('express');
const path = require('path');
const app = express();

// Serve static files from the "public" directory
app.use(express.static(path.join(__dirname, 'public')));

// Start the server
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

4. Download 3D model and keep in public folder or create folder Markers and keep that folder in public folder and save your 3D model inside markers folder

➔ Create MarkessLess.html and write code

```
 <!DOCTYPE html>
<html>
<head>
  <title>Simple A-Frame Scene</title>
  <script src="https://aframe.io/releases/1.2.0/aframe.min.js"></script>
</head>
<body>
  <a-scene>
    <a-assets>
      <!-- Make sure the path to your GLTF model is correct -->
      <a-asset-item id="tree" src="./markers/TankTop.gltf"></a-asset-item>
    </a-assets>
```

```
    <!-- Camera entity -->
    <a-entity position="0 1.6 0">
      <a-camera></a-camera>
    </a-entity>

    <!-- GLTF Model entity -->
    <a-entity gltf-model="#tree" position="-1 0 -3" rotation="0 45 0" scale="2.5 2.5 2.5"
material="color: red"></a-entity>

    <!-- Example of a simple box to ensure the scene is working -->
    <a-box position="0 0.5 -2" rotation="0 45 0" color="#4CC3D9"></a-box>
  </a-scene>
</body>
</html>
```

5. Start Your Server

```
node server.js
```

//here you need to use command prompt ->locate your folder and run server

```
D:\>cd D:\my-aframe-project

D:\my-aframe-project>node server.js
Server is running on http://localhost:3000
```

6. Type http://localhost:3000/MarkessLess.html in the browser to run application