

Program 1:

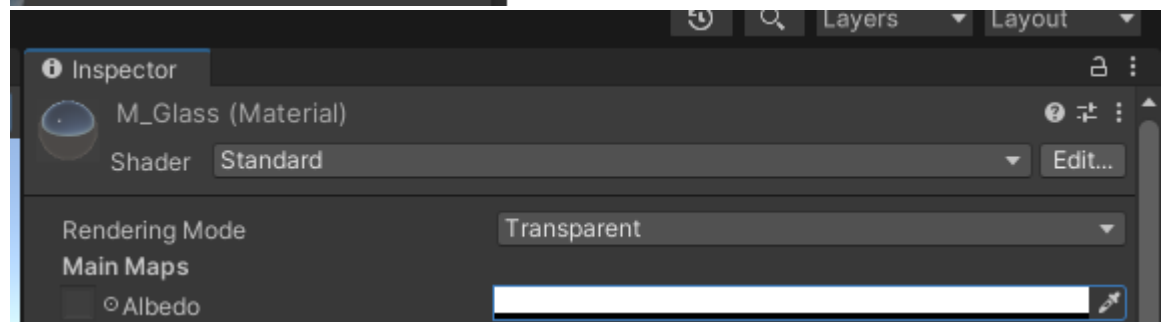
Create a 3D environment with particle effects, Materials and other effects.

Sol:

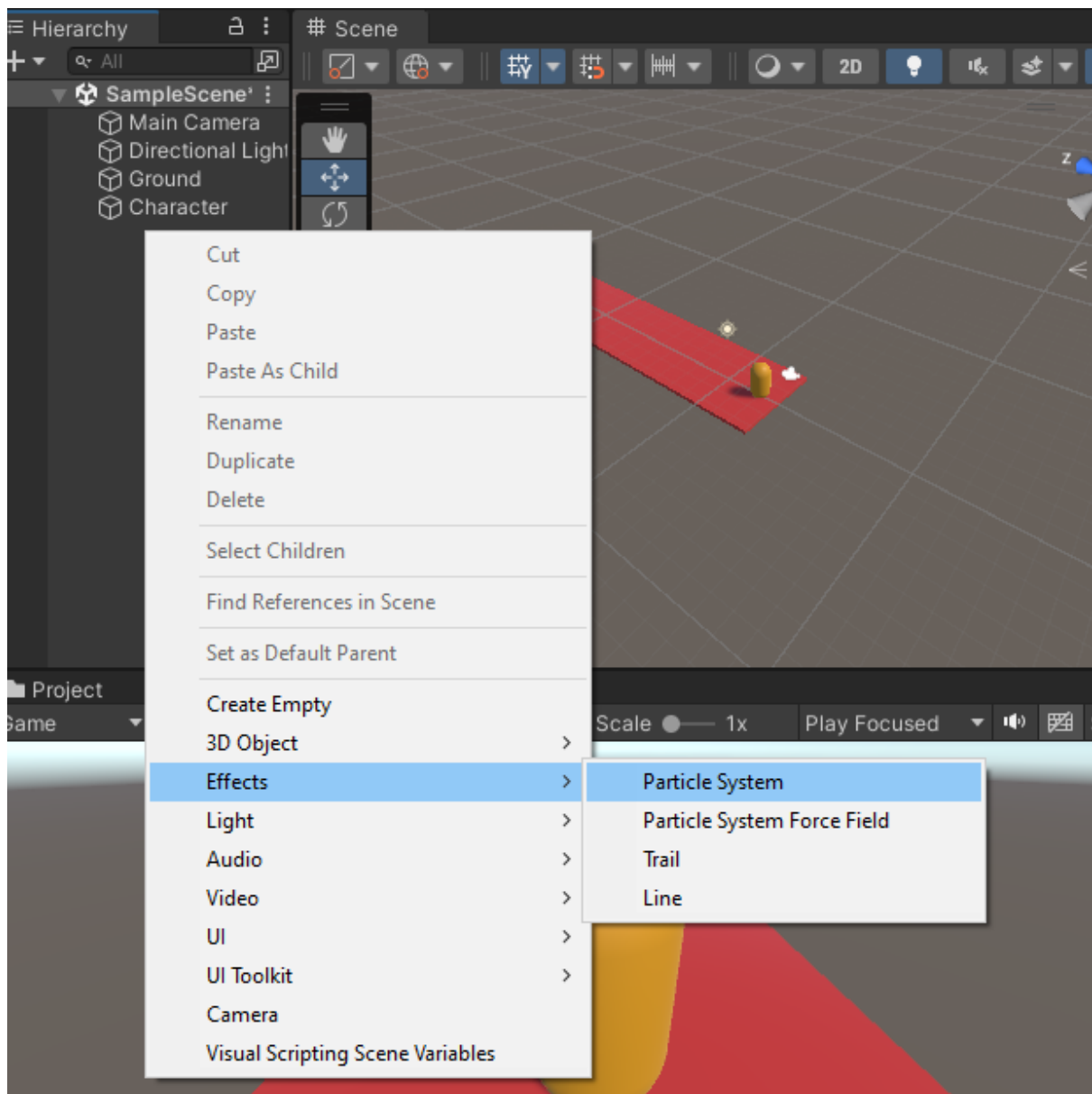
Steps:

1. Add 3D object cube and set (x,y,z) as 5,0.5,20
2. Create material
 - a. colour Blue and increase smoothness apply on Ground
 - b. Glass material: Increase smoothness and shine and change Rendering Mode to transparent

Click Albedo and set Apha to 0



3. Apply material on cube and name cube as Ground
4. Add Capsule and name it Character and Add material yellow colour
 - a. Add component Rigidbody to character
5. Right Click -> Effect-> ADD Particles



6. Change properties as below
 - Start speed=1
 - color=orange
 - Shape =cone
 - Degree =0
 - radius=0.3 to 0.5
7. Add the Empty game object and add box Collider as component
8. Add tag name End in empty game object
9. Create c# script and attach it to character
10. Open script in VS code and write code

```

using System.Collections;
using System.Collections.Generic;
using Unity.VisualScripting;
using UnityEngine;

public class FirstLab : MonoBehaviour
{
    public float speed = 100f;
    public ParticleSystem Particles;
    private void Update()
    {
        // Move the capsule forward in the Z direction
        if(Input.GetKeyDown(KeyCode.W))
        {
            transform.Translate(Vector3.forward * speed * Time.deltaTime);
        }
        if(Input.GetKeyDown(KeyCode.A))
        {
            transform.Translate(Vector3.left * speed * Time.deltaTime);
        }
        if(Input.GetKeyDown(KeyCode.D))
        {
            transform.Translate(Vector3.right * speed * Time.deltaTime);
        }
    }

    private void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.tag=="End")
        {
            Particles.Play(); // Activate particles
        }
    }
}

```

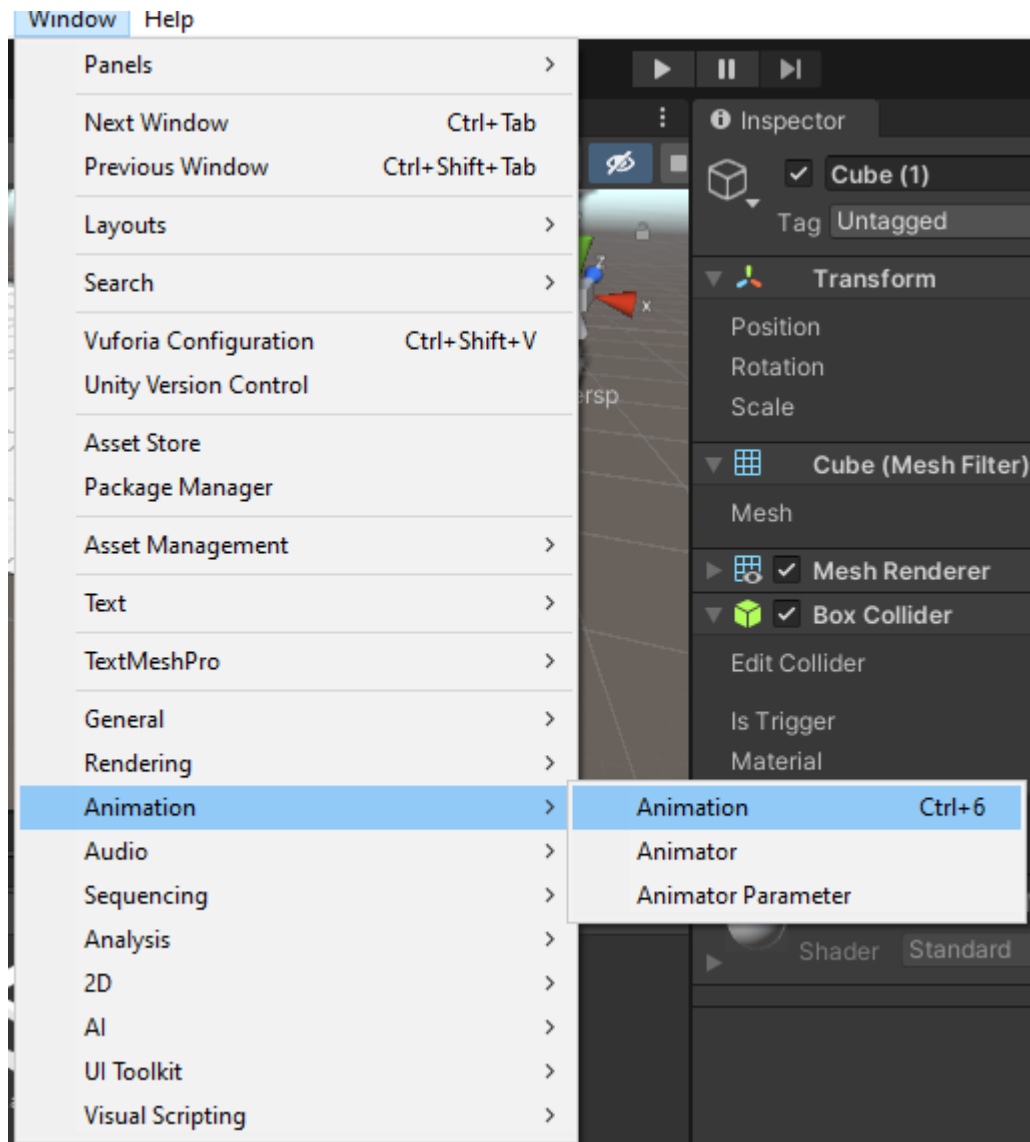
11. Run and check character movement.

LAB 2: Create an application that animates characters in the real world, triggered by user interactions

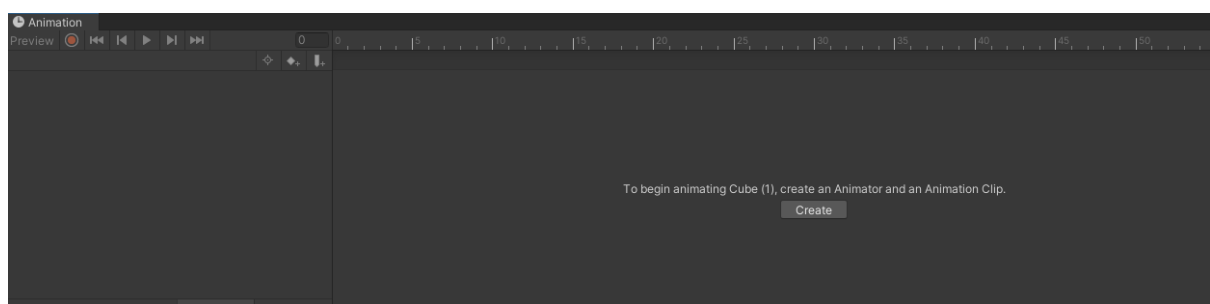
Step 1: Create 3D environment Like Ground

Step 2: Create Animation:

- Right Click and Add Cube
- Select cube and go to Windows->Animation->animation



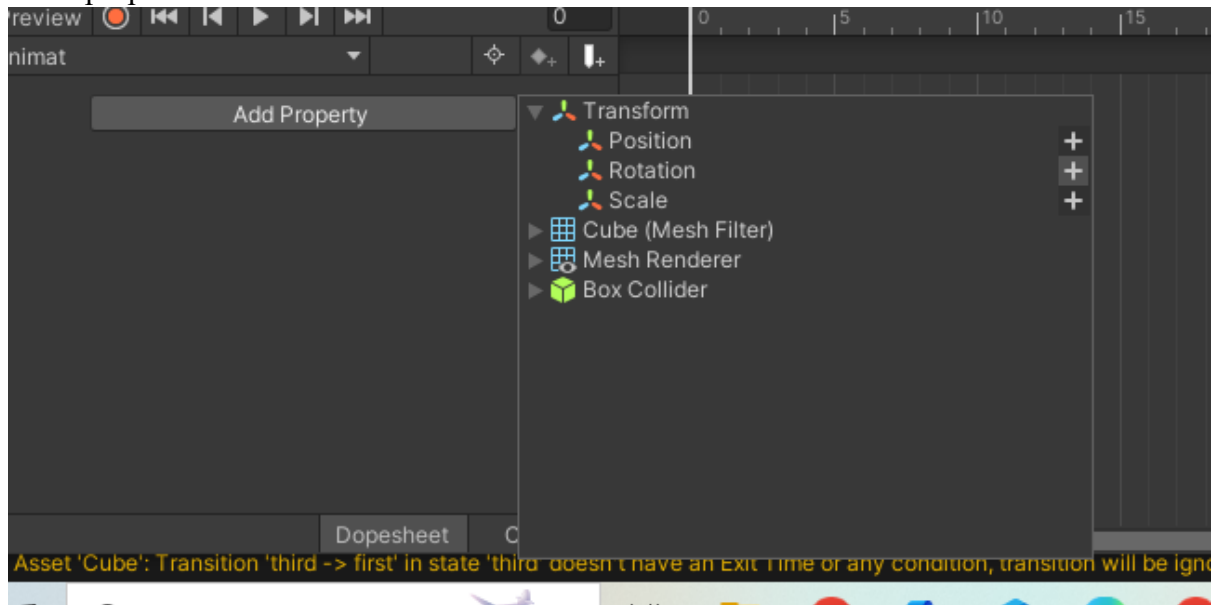
- Click Create



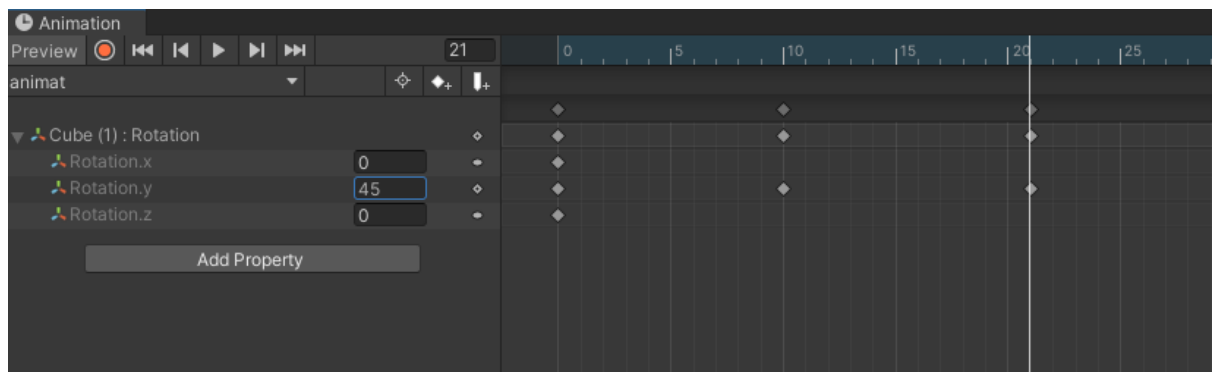
- Save First.anim

Step 3: Create Frames

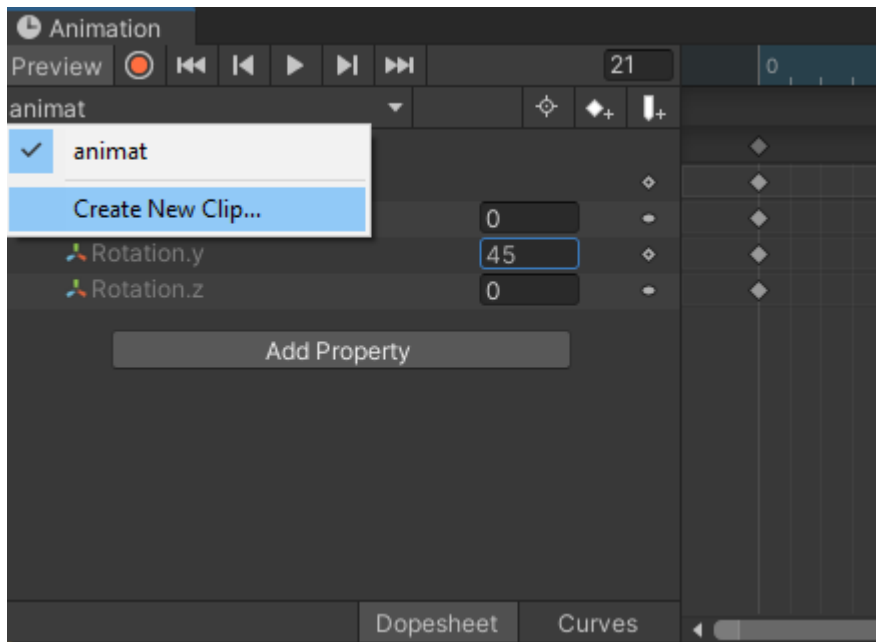
- Click properties->Transform->Rotation



- Move white line to 10 and add rotation in y 30, again move line to 20 and add 45 and so on



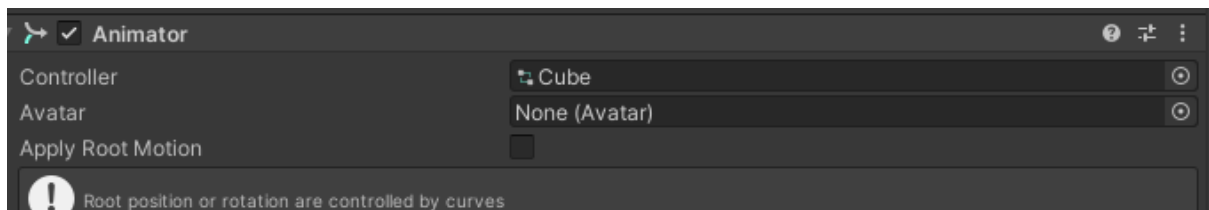
- Click on new clip and save animation second and repeat for four animation same



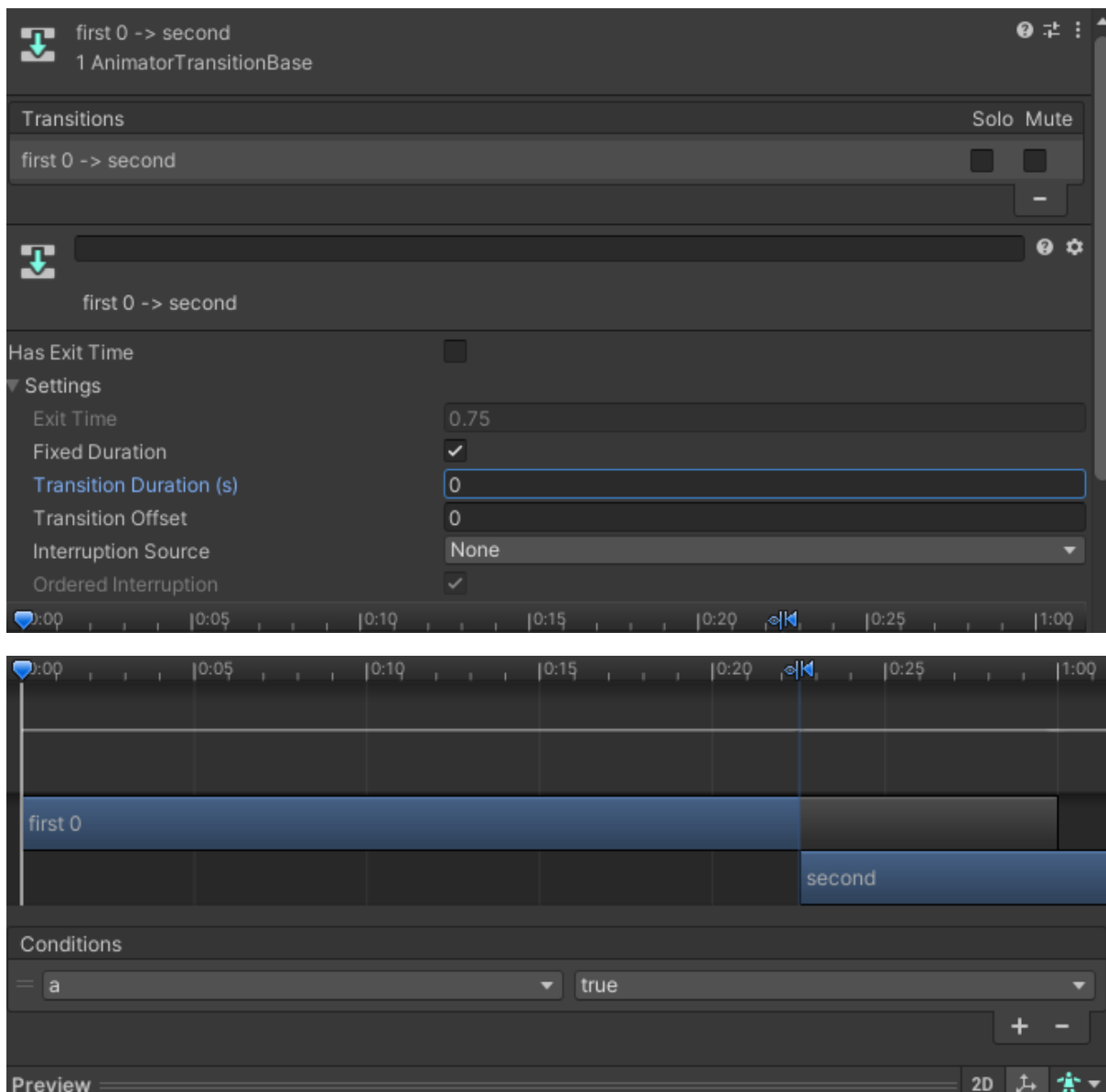
Step 4: Create Animation Graph

Note: it works like Finite automata

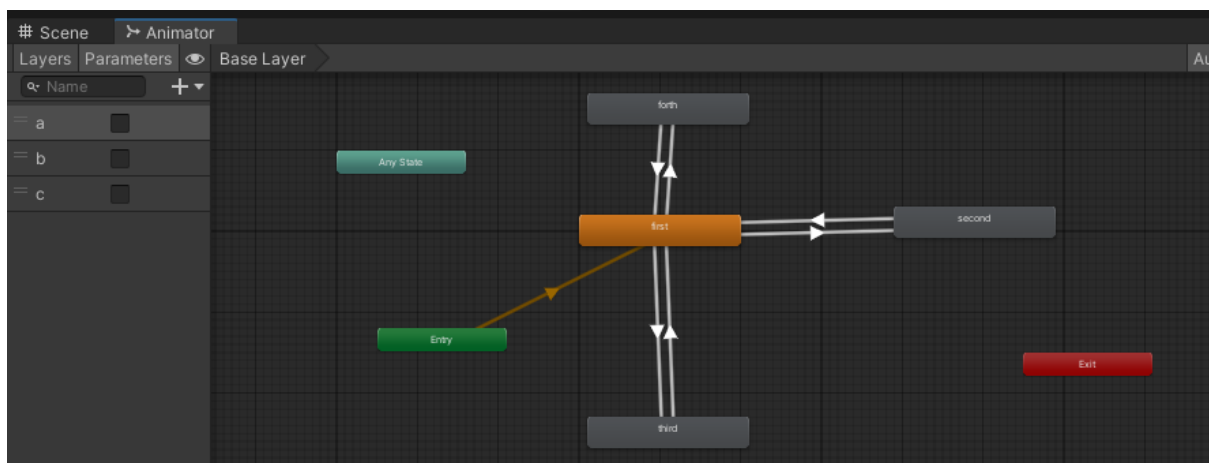
- Check u get animator in cube



- Open cube animator and connect transition
 - Create parameter as bool
 - First to second if a is true and second to first if a is false
 - Connect all animation same
 - Select each transition and perform setting as below

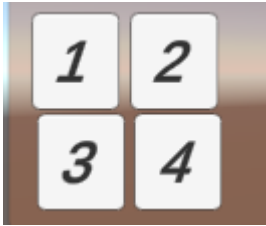


Create state diagram for all animation and add parameters



Step 4: Create UI by adding Button

- Right Click->UI->Add Panel
- Right Click-> UI->Button->Add Textmesh pro
- Click Import essentials
- Buttons are child of Panel

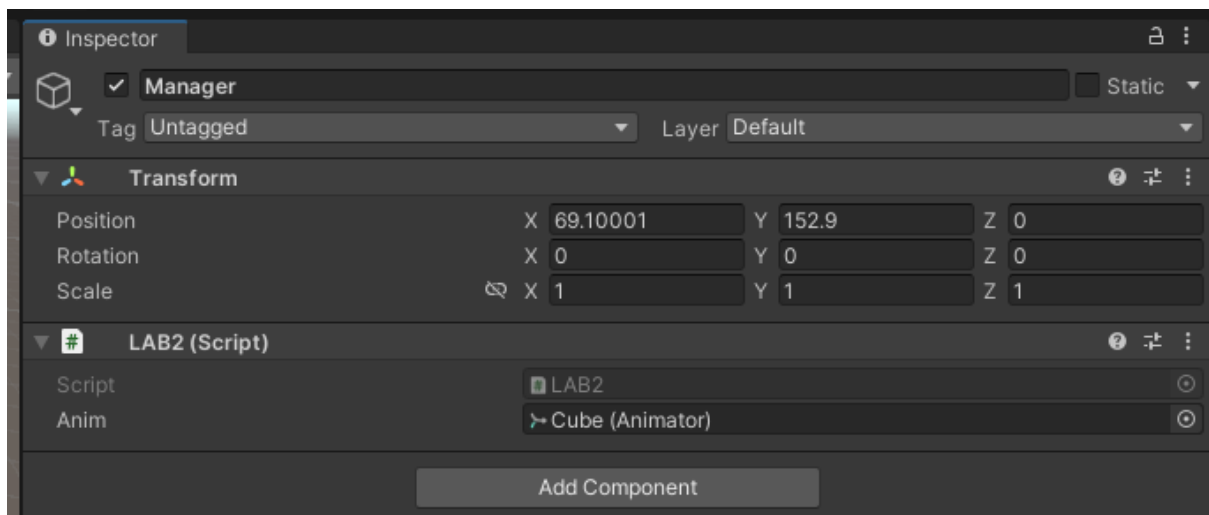


Step 5: Create c# script Lab2

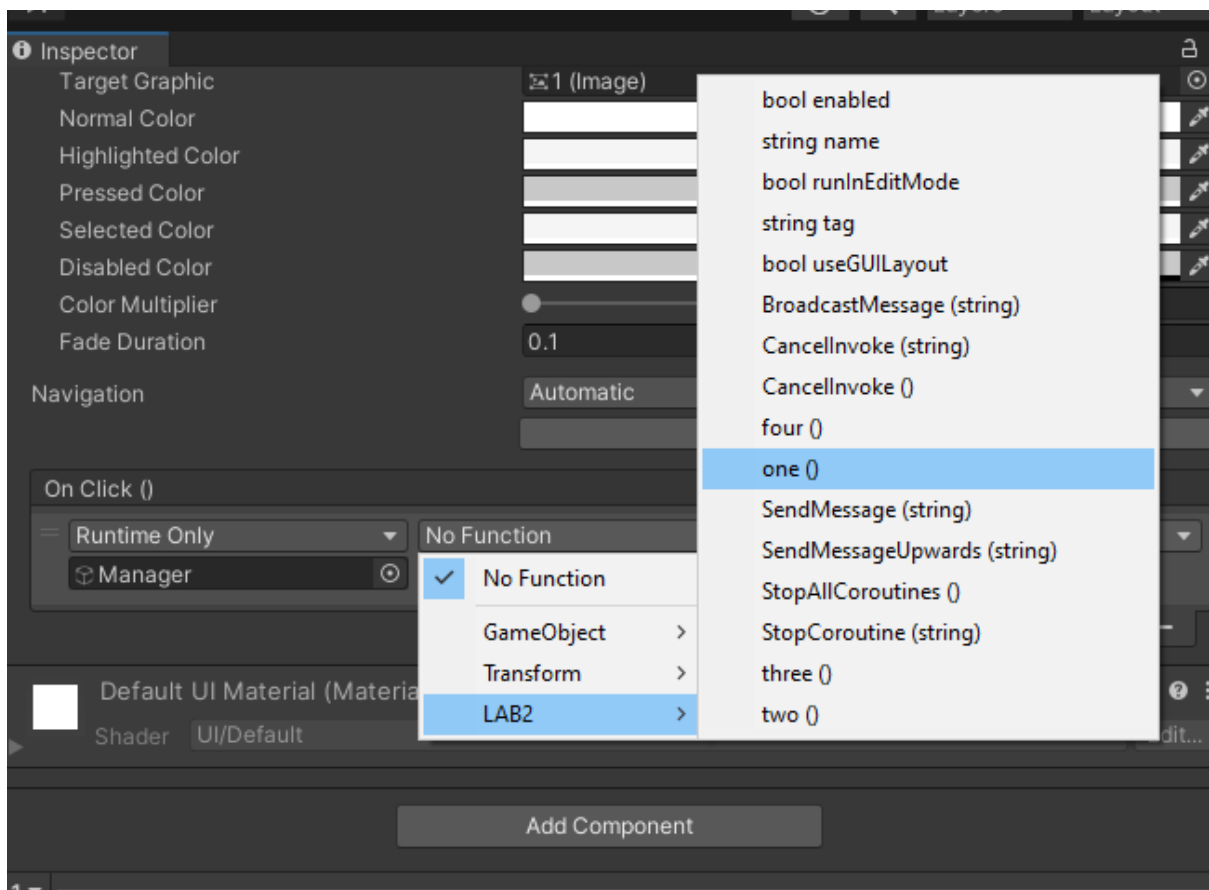
```
using UnityEngine;
public class LAB2 : MonoBehaviour
{
    public Animator anim;
    public void two()
    {
        anim.SetBool("a",true);
    }
    public void one()
    {
        anim.SetBool("a",false);
        anim.SetBool("b",false);
        anim.SetBool("c",false);
    }
    public void three()
    {
        anim.SetBool("b",true);
    }
    public void four()
    {
        anim.SetBool("c",true);
    }
}
```

Step6: Attach script

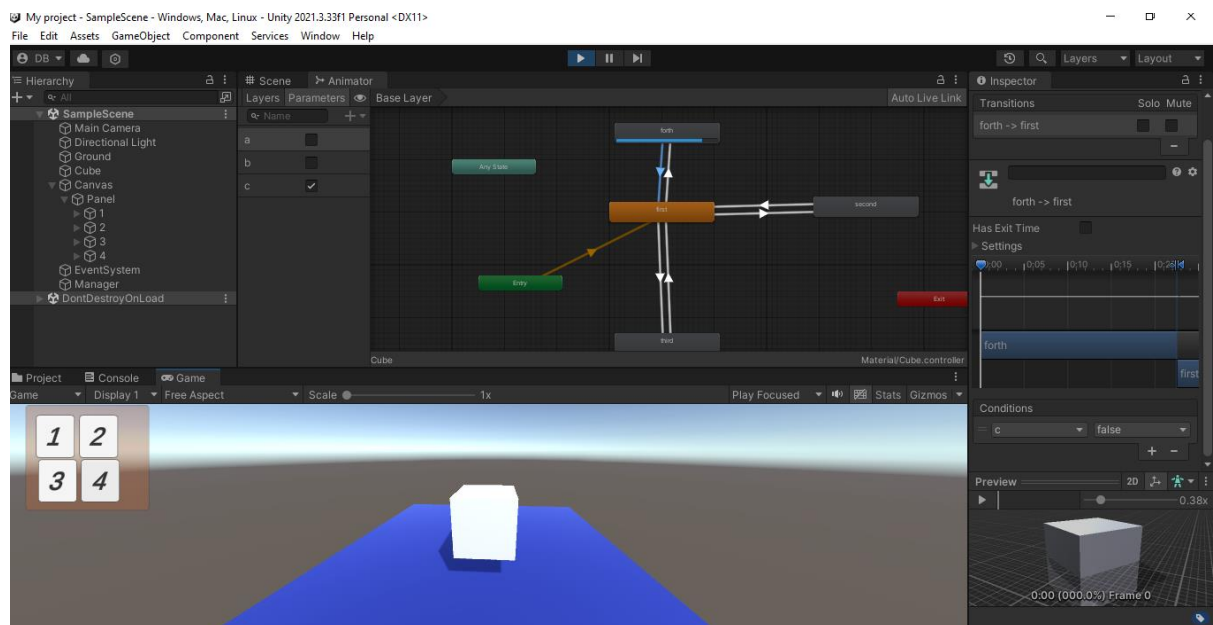
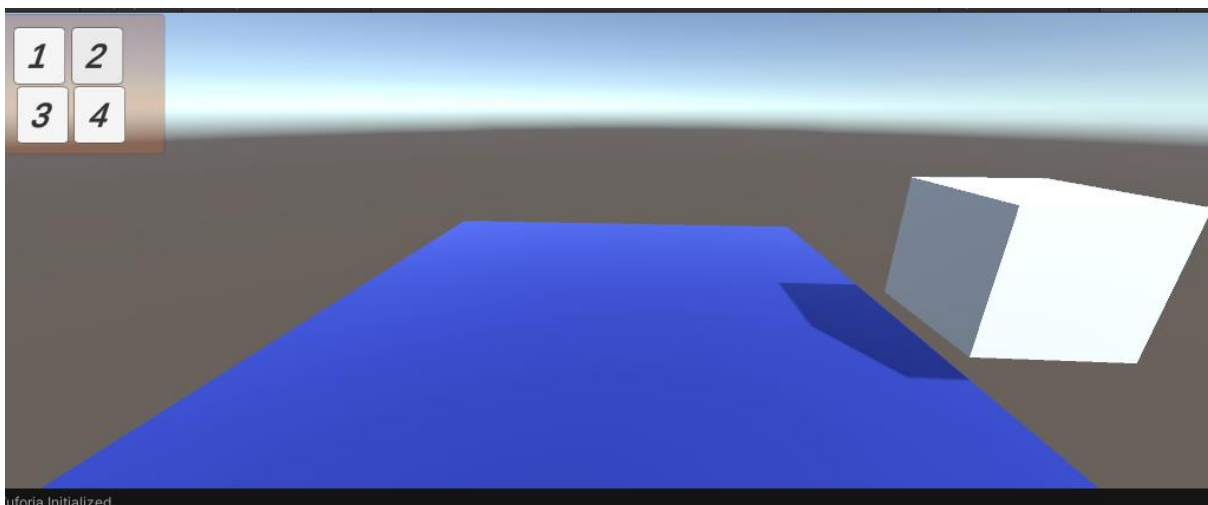
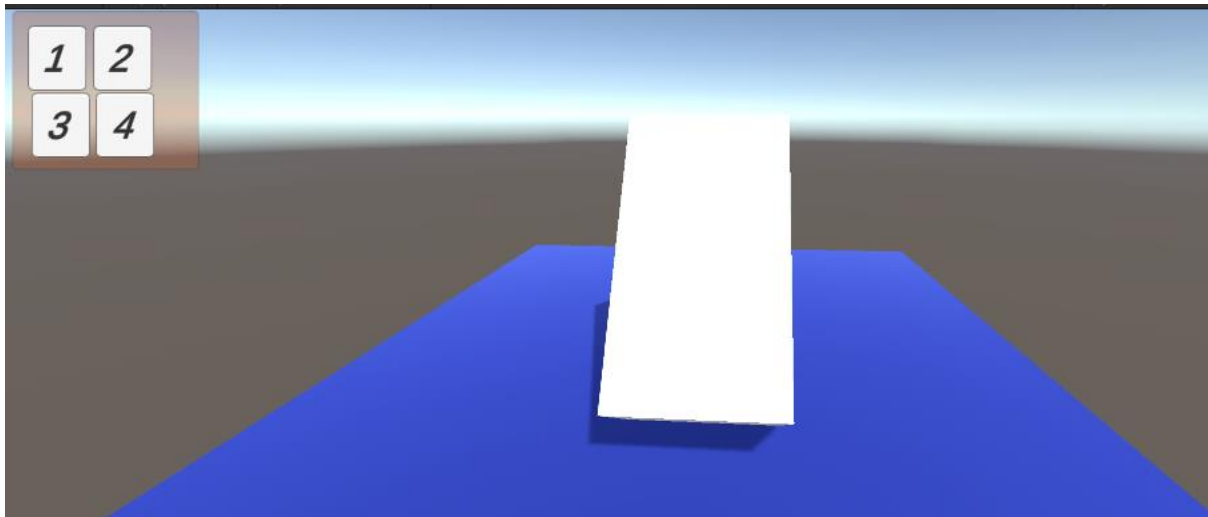
- Create empty game object name it manager
- Attach script to this and drag cube in place of animator



Step6: click button and attach function



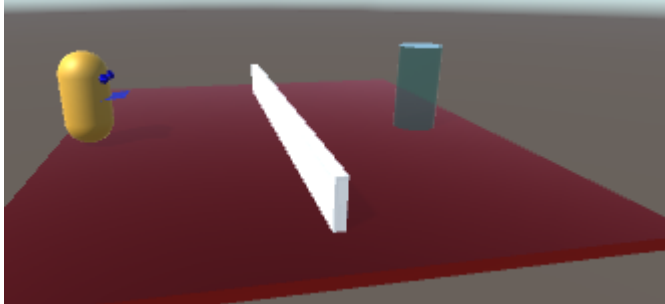
Output:



LAB 3: Create a player character as a 3D model and enable transformative actions such as moving left/right and forward/backward. (ADD AI)

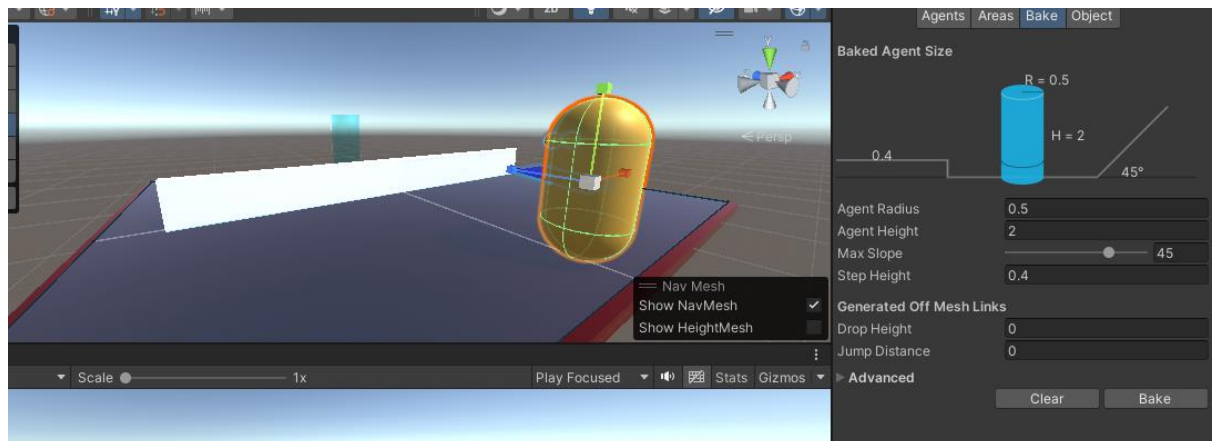
Sol:

Step 1: create environment for character, ground, Obstacle and AI

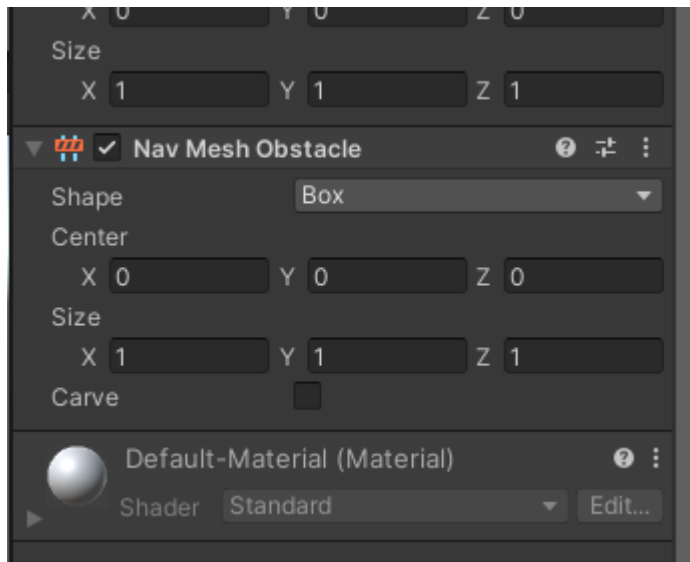


Step 2: Select Ground and click static

- Add Navigation path here it uses built-in A*
- Click bake it makes ground blue



Step 3: select white cube and add component Navmesh obstacle it will stop AI



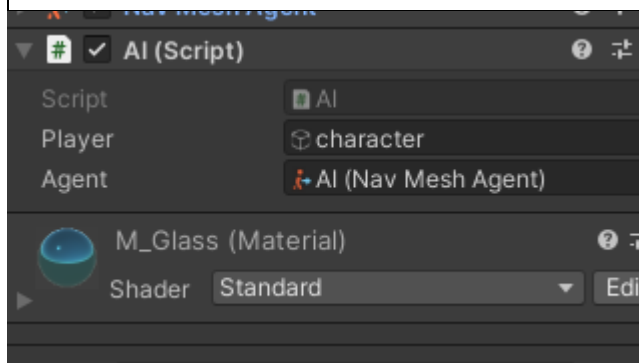
Step 4: Select AI ,cylinder and Add navmesh agent as component

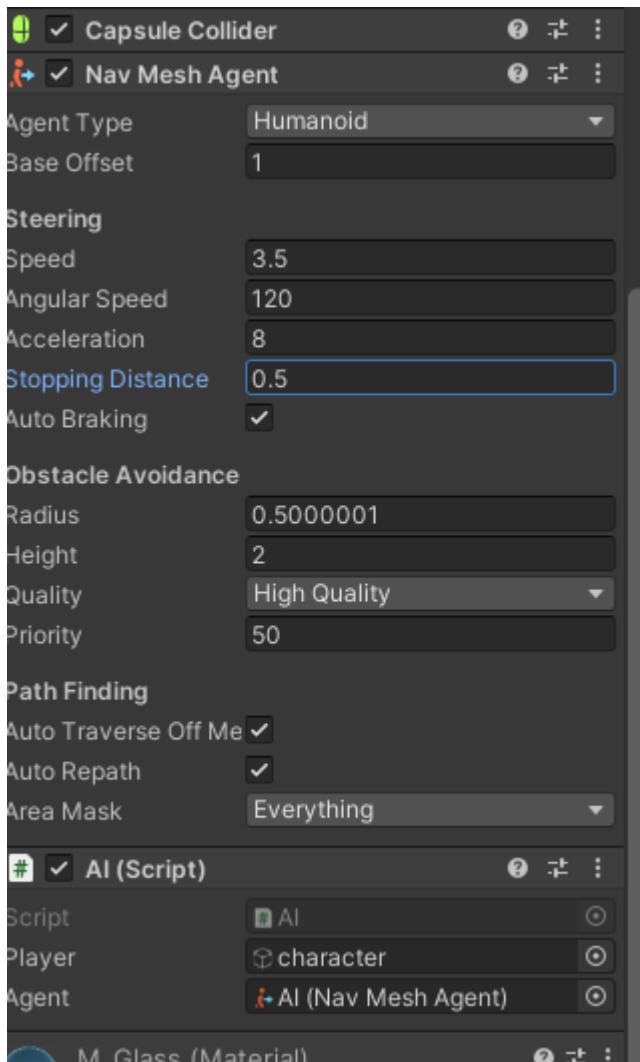
- Create c# script AI and attach to Cylinder i.e our AI

```
using UnityEngine;
using UnityEngine.AI;

public class AI : MonoBehaviour
{
    public GameObject player;
    public NavMeshAgent agent;

    void Update () {
        agent.SetDestination(player.transform.position);
    }
}
```





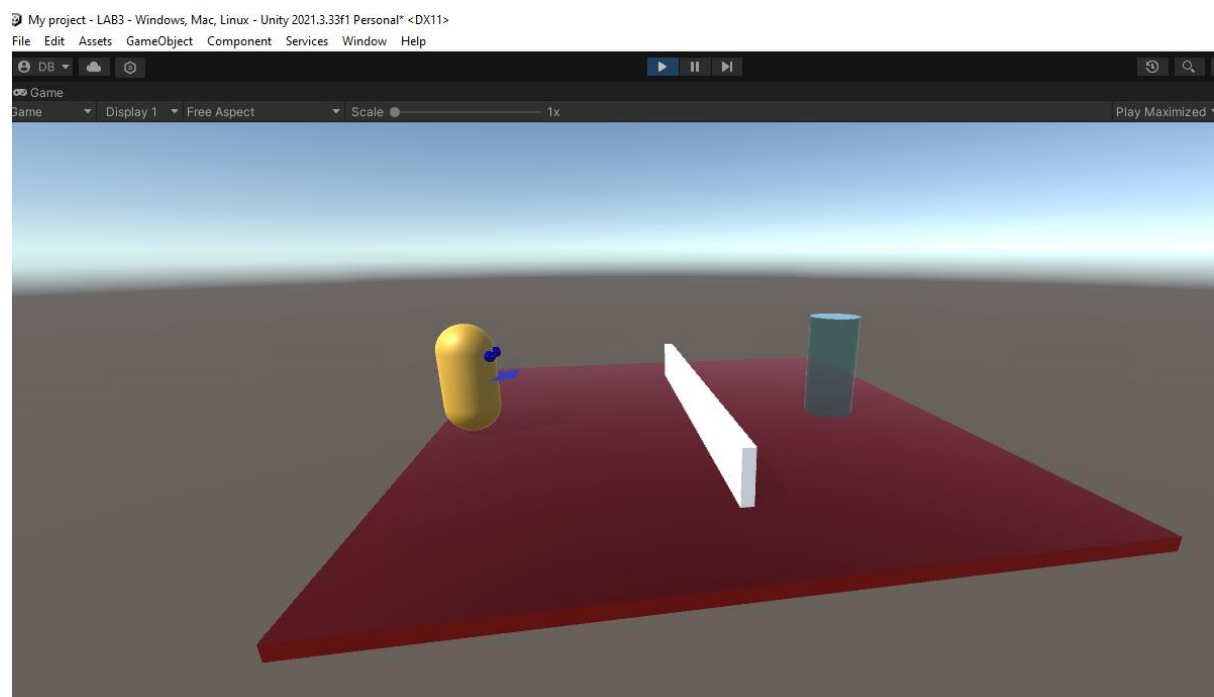
Step 5: Create C# Script ,Attach script to character and attach reference for character controller

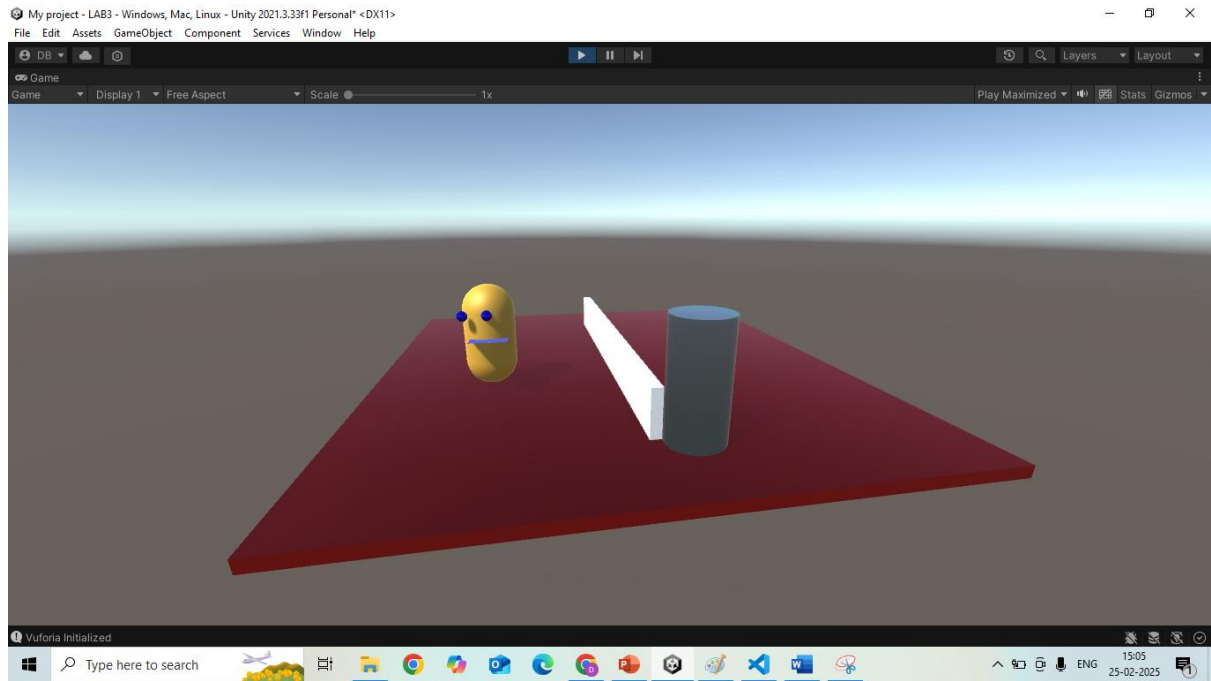
```
using UnityEngine;

public class LAB4 : MonoBehaviour
{
    public float moveSpeed = 50f;
    public float rotationSpeed = 700f;
    public CharacterController controller;
    private Vector3 moveDirection;
    void Update()
    {
        float moveX = Input.GetAxis("Horizontal");
        float moveZ = Input.GetAxis("Vertical");
        // Calculate movement direction based on input
        moveDirection = new Vector3(moveX, 0f, moveZ);
        if (moveDirection.magnitude > 0)
```

```
{
    Quaternion toRotation = Quaternion.LookRotation(moveDirection, Vector3.up);
    transform.rotation = Quaternion.RotateTowards(transform.rotation, toRotation,
rotationSpeed * Time.deltaTime);
}
// Apply the movement to the character
controller.Move(moveDirection * moveSpeed * Time.deltaTime);
}
}
```

Output:





LAB 4: Create an AR application with Vuforia, incorporating 3D models, interactive features using UI buttons

Sol:

Step 1: Create Vuforia Account

<https://developer.vuforia.com/home>

Step 2: Create Licence

- By click licence key
- Click Generate Basic Licence
- Give Name and Click Confirm

Add a license key to your Basic plan

License Name *
XYZ

You can change this later

☒ By checking this box, I acknowledge that this license key is subject to the Vuforia License Agreement

Cancel

Confirm

Step 3: Add Vuforia SDK from unity

<https://assetstore.unity.com/packages/templates/packs/vuforia-engine-163598?srltid=AfmBOoohbmlYwddAmIOAITZIC56DTKf4CSIsTma6A1zYXLWdXyOFvMMU>

or download from Vuforia portal and import

Vuforia Engine 11.0

Use the Vuforia Engine SDK to build augmented reality Android, iOS, and UWP applications for mobile devices and digital eyewear. Vuforia Engine can be used in projects built with [Unity](#), [Android Studio](#), [Xcode](#), and [Visual Studio](#).

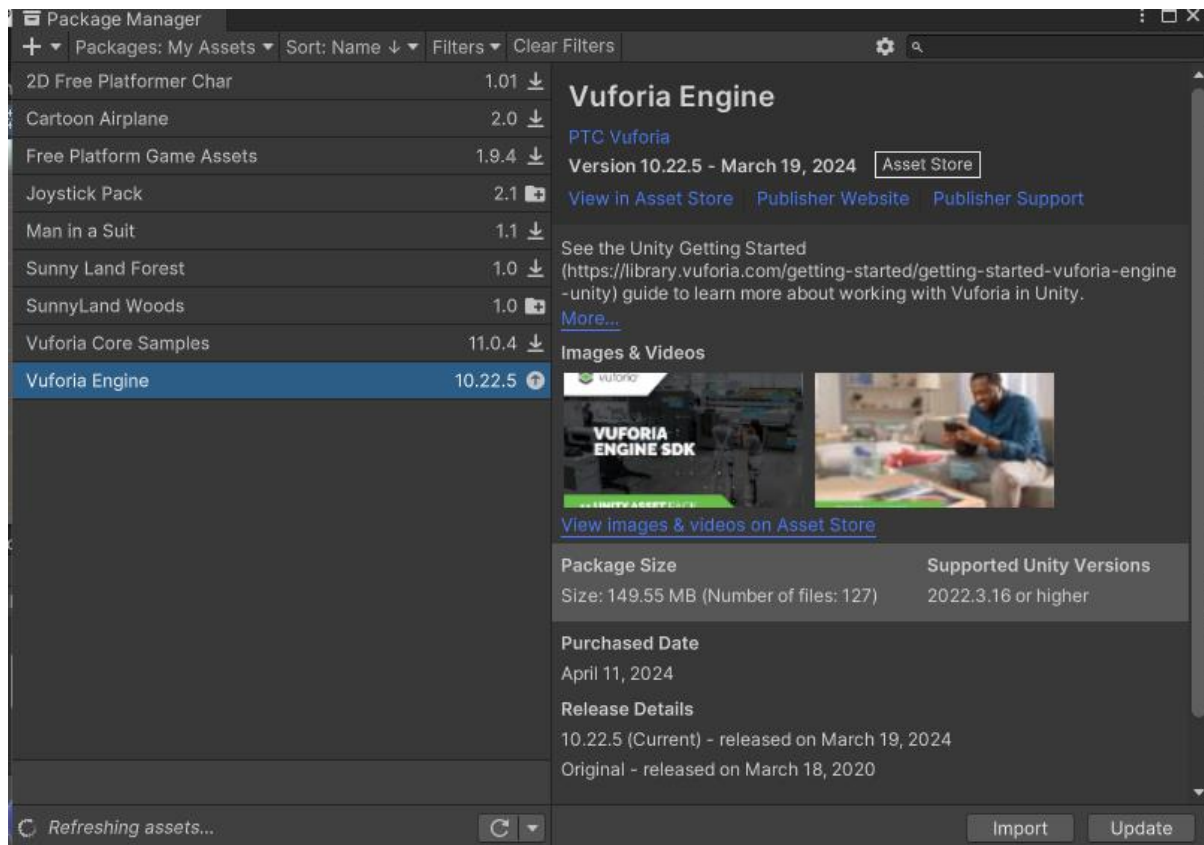


Download for Unity

add-vuforia-package-11-0-4.unitypackage

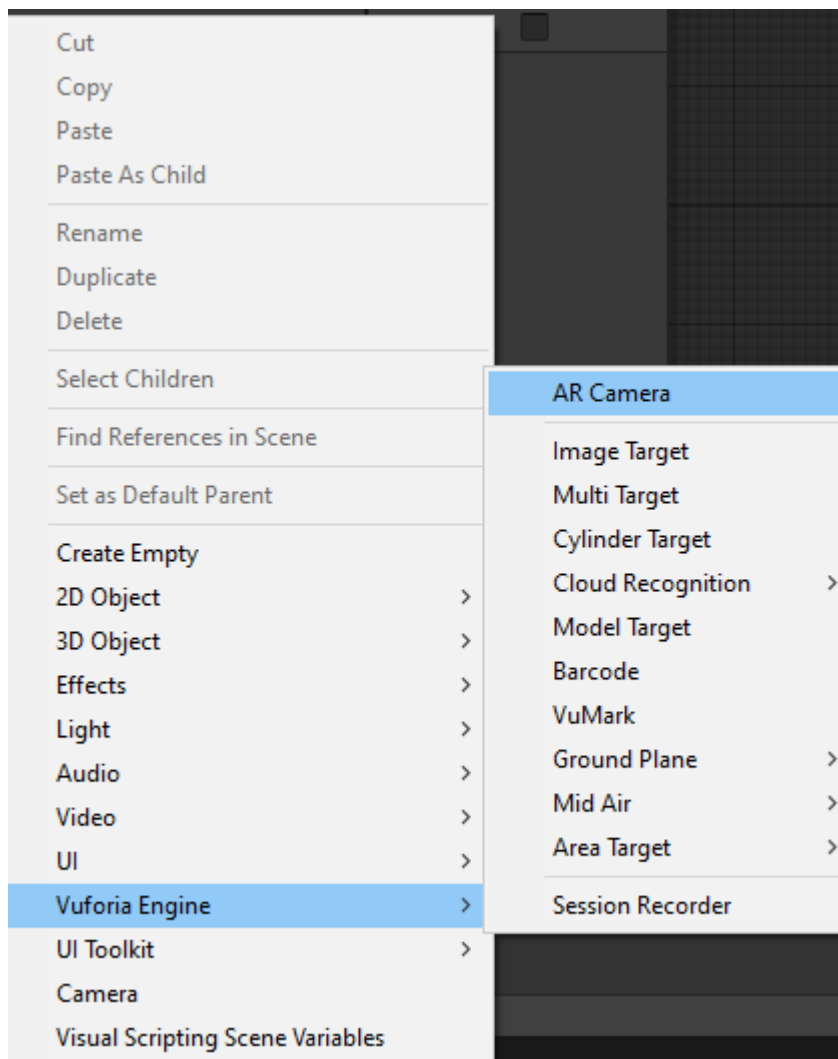
Download SHA-256

Step 4: Add asset in unity and click import

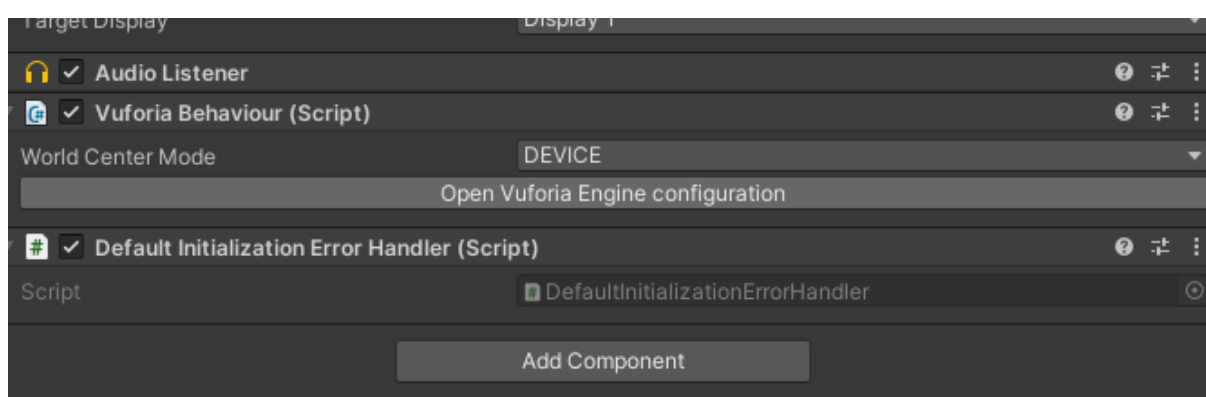


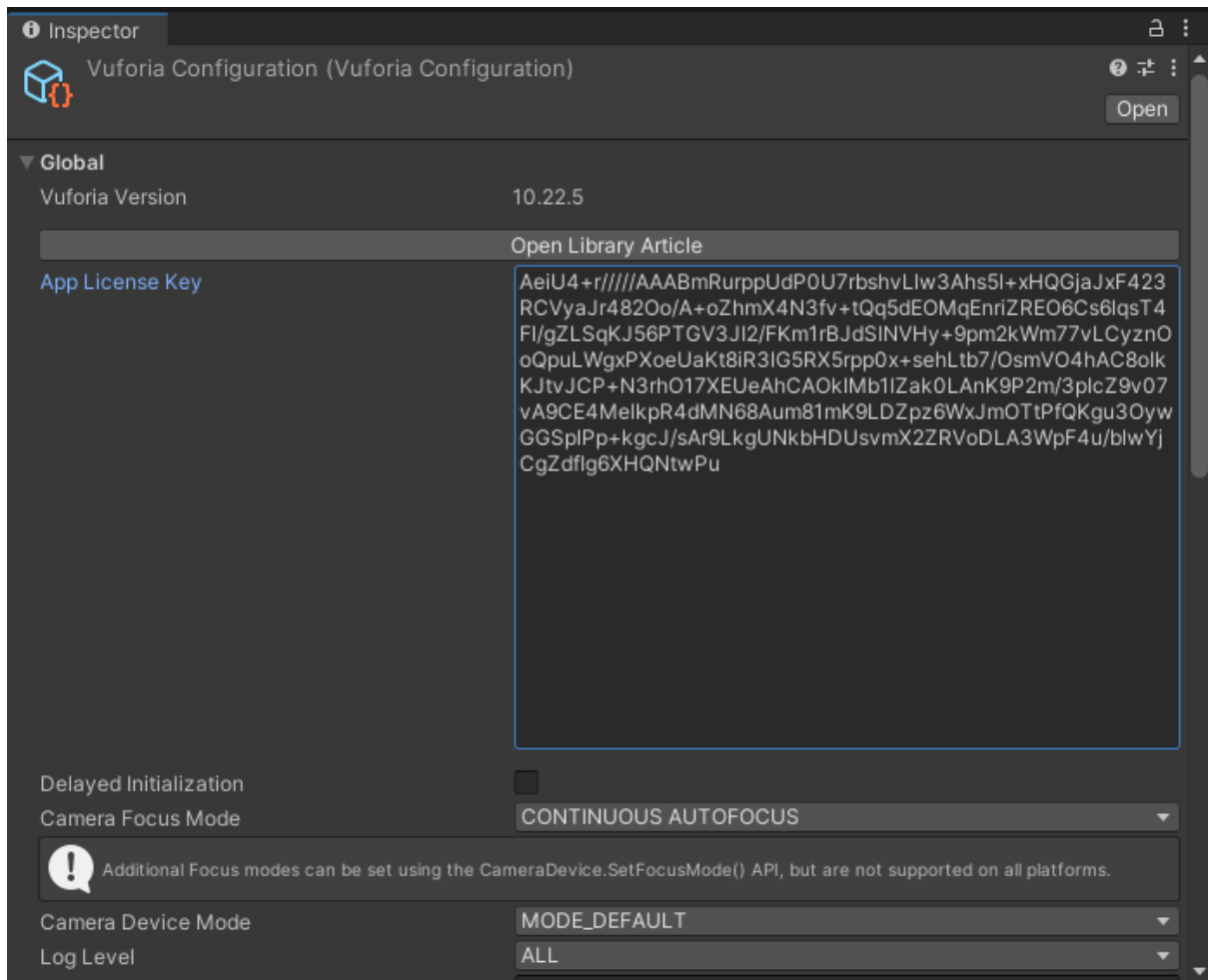
Step 5: Setup Vuforia Engine in unity

- Right click and Add AR camera and delete main camera



- Click AR camera from hierarchy and click open Vuforia configuration and add licence key from Vuforia account





Step 6: add image in Vuforia data base

- Click target manager
- Click generate database give name

Generate Database

Database Name *
DSCE2022

Type:

- ☒ Device
☐ Cloud
☐ VuMark

Cancel

Generate

- Click data base DSCE2022
- Click add target

DSCE2022 [Edit Name](#)

Type: Device

Targets (0)

Add Target

Download Database (All)



Image

Target Name

Type

Rating 

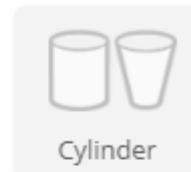
Status 

Date Modified

- Upload image

Add Target

Type:



File:

Choose file

watchtarget.jpeg

.jpg or .png (max file 2mb)

Width

5

Enter the width of your target in scene units. The size of the target should be on the same scale as your augmented virtual content. Vuforia uses meters as the default unit scale. The target's height will be calculated when you upload your image.

Name

watchtarget

Name must be unique to a database. When a target is detected in your application, this will be reported in the API.

Cancel

Add

- Click image and check ratings 4 or more is good

watchtarget

[Edit Name](#) [Remove](#)



[Update Target](#) [Show Features](#)

Type: Image
Status: Active
Target ID: 040163aaa83a4cebafee1cf4737b7ac3
Augmentable: ★★★★★
Added: Feb 25, 2025
Modified: Feb 25, 2025


- Download database and click unity editor

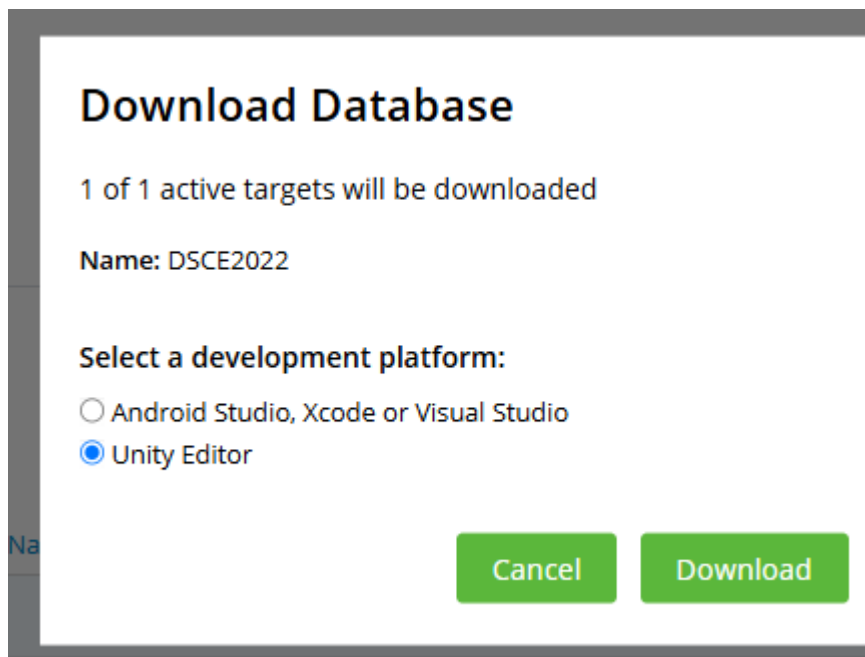
[Target Manager](#) > DSCE2022

DSCE2022

[Edit Name](#)

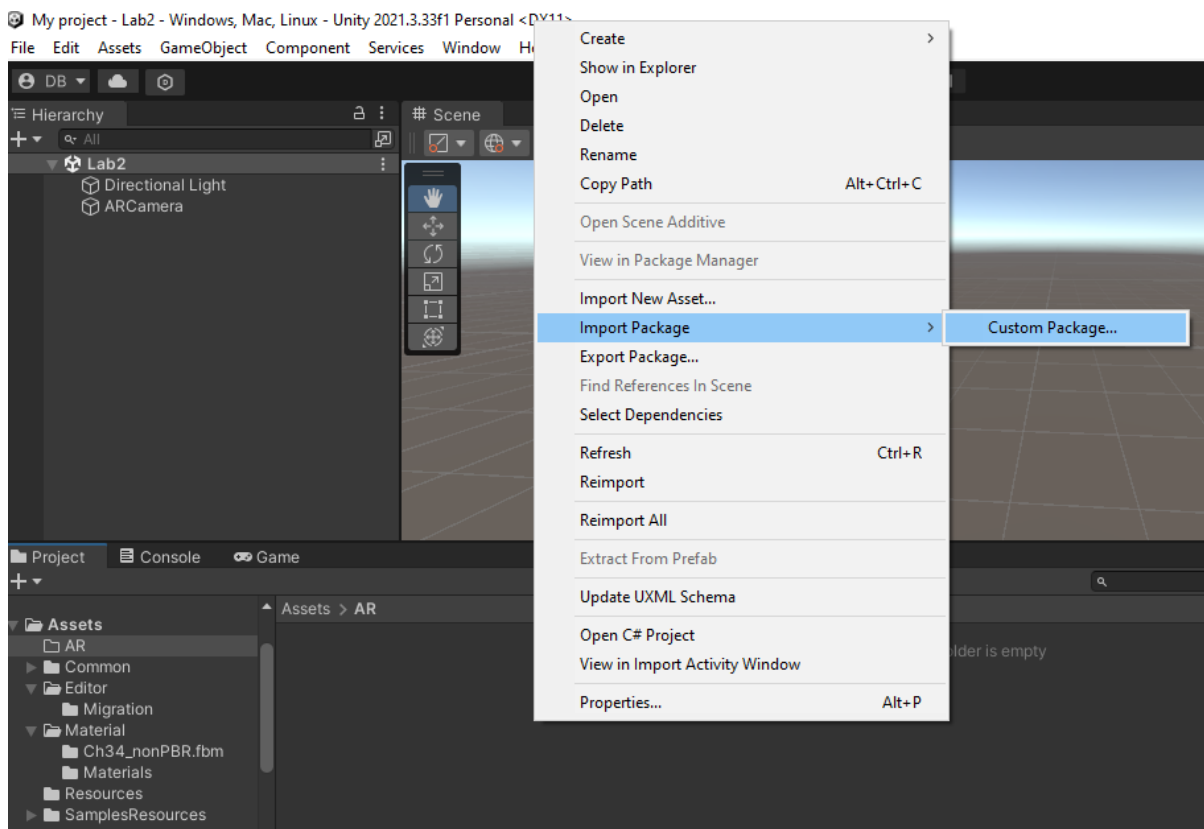
Type: Device

Targets (1)						
Add Target			Download Database (1)			
<input checked="" type="checkbox"/>	Image	Target Name	Type	Rating ⓘ	Status ▾	Date Modified
1 selected Delete						
<input checked="" type="checkbox"/>		watchtarget	Image	★★★★★	Active	Feb 25, 2025



Step 7: Import database in unity

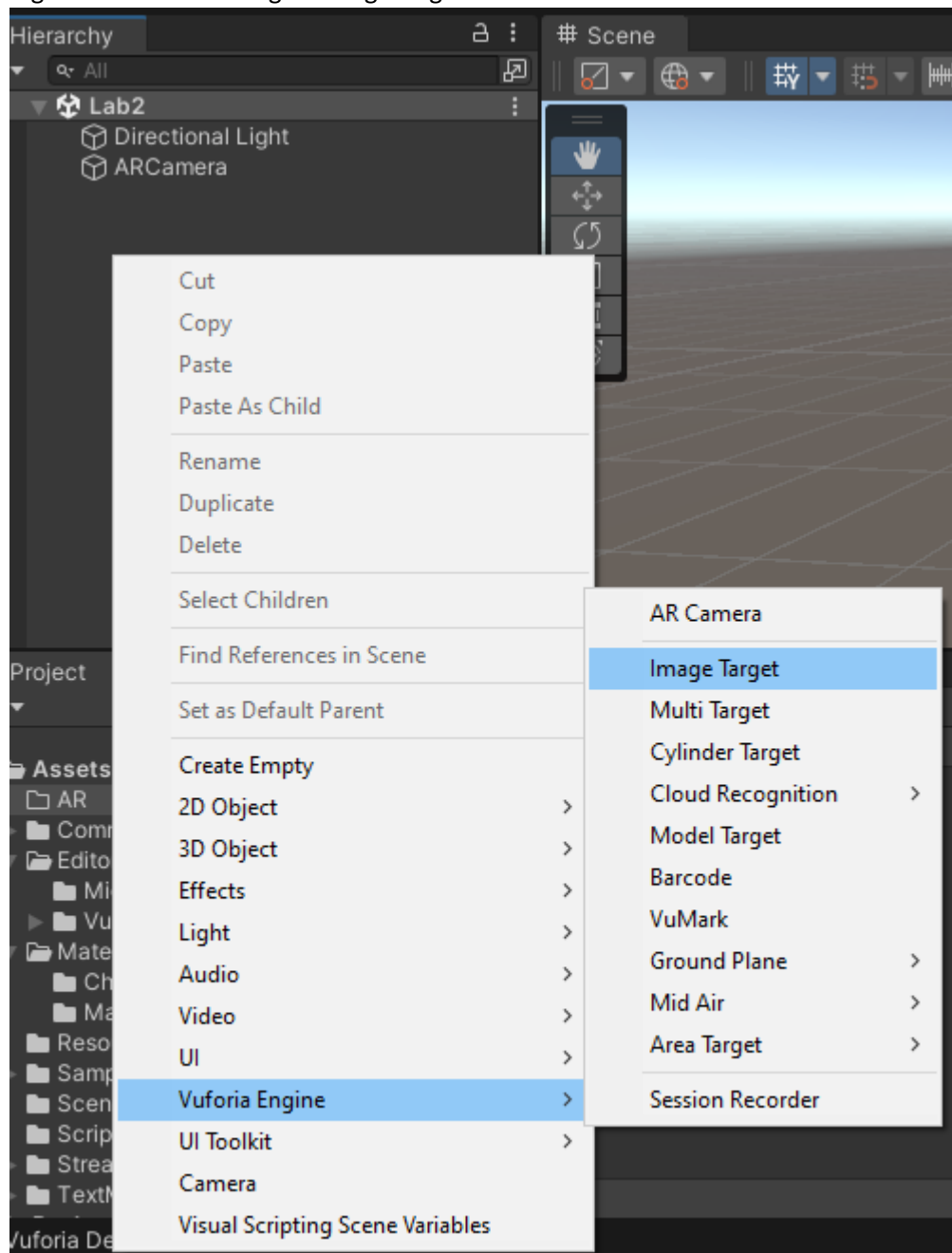
- Open unity create AR folder
- Right click import package->custom package



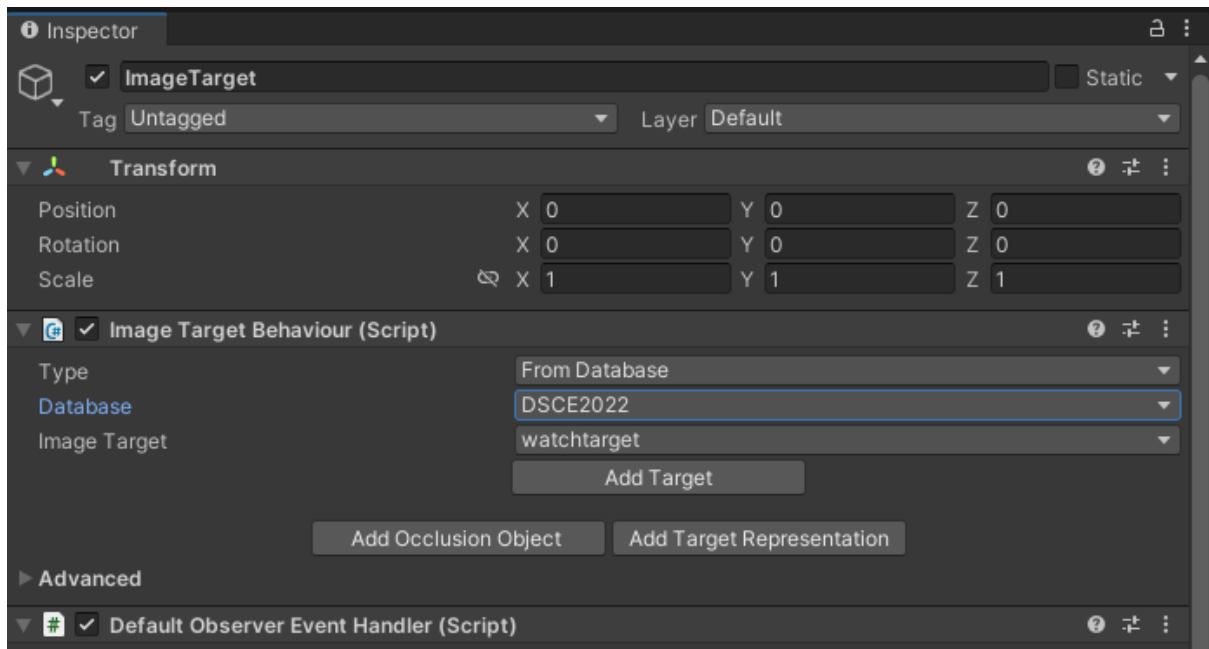
- Click downloaded package and click import

Step 8: setup in unity for image target

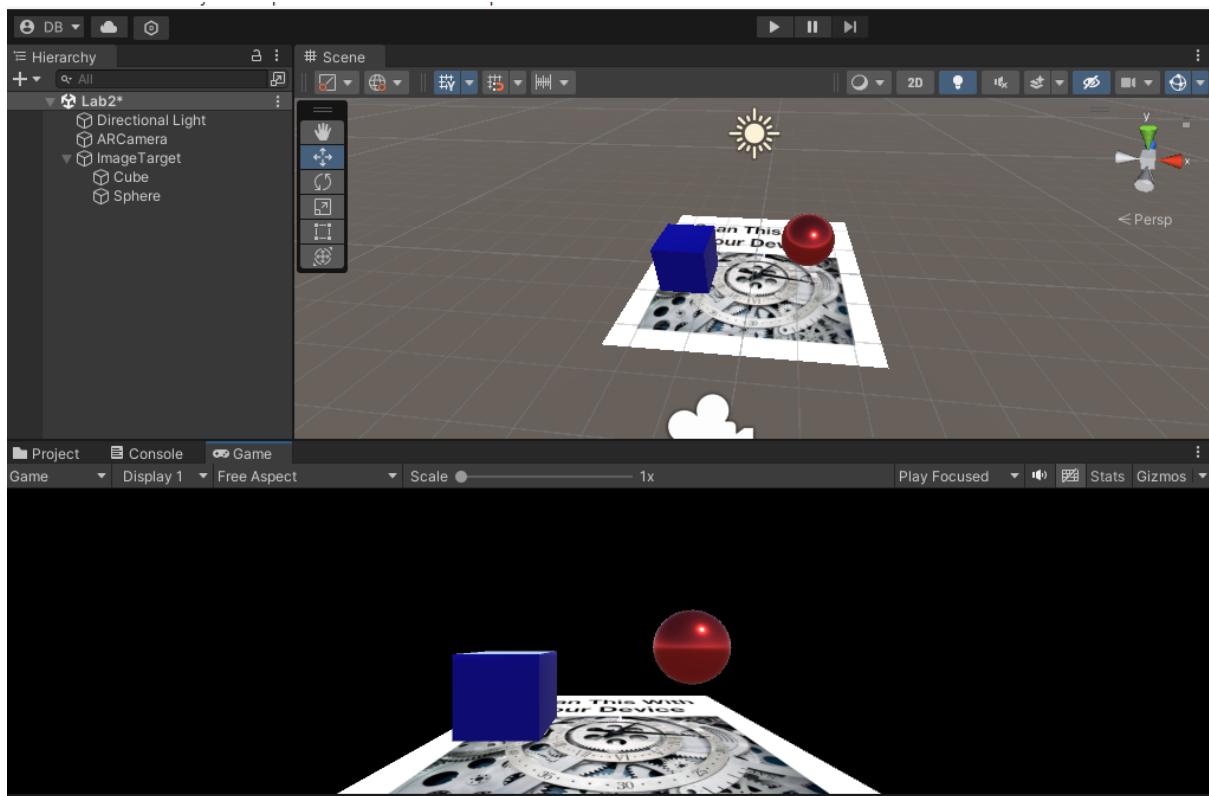
- Right click -> Vuforia engine image target



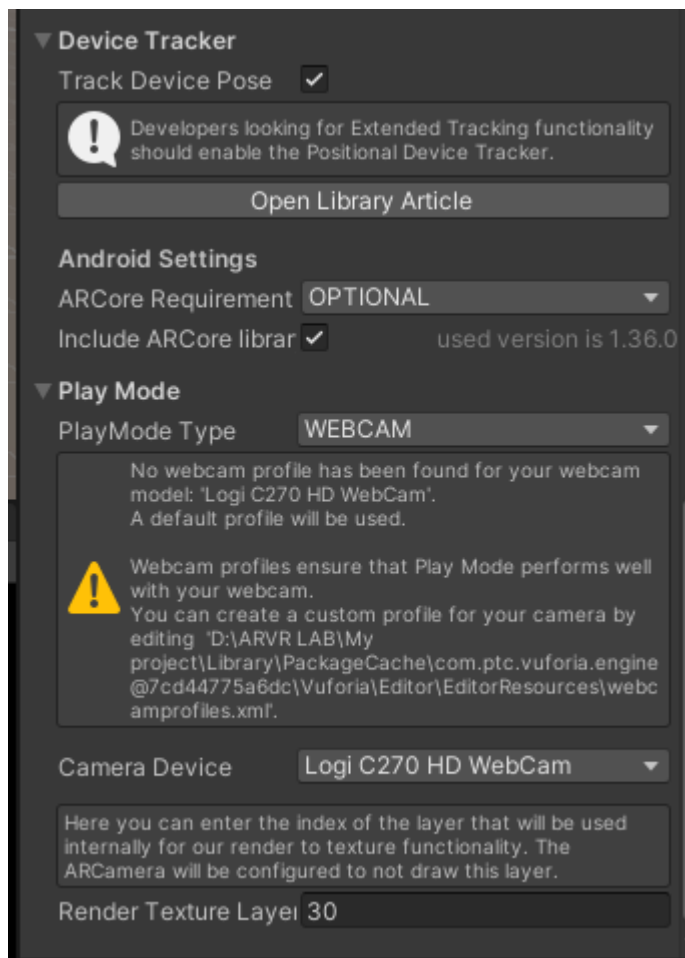
- Select image target and select database and your image



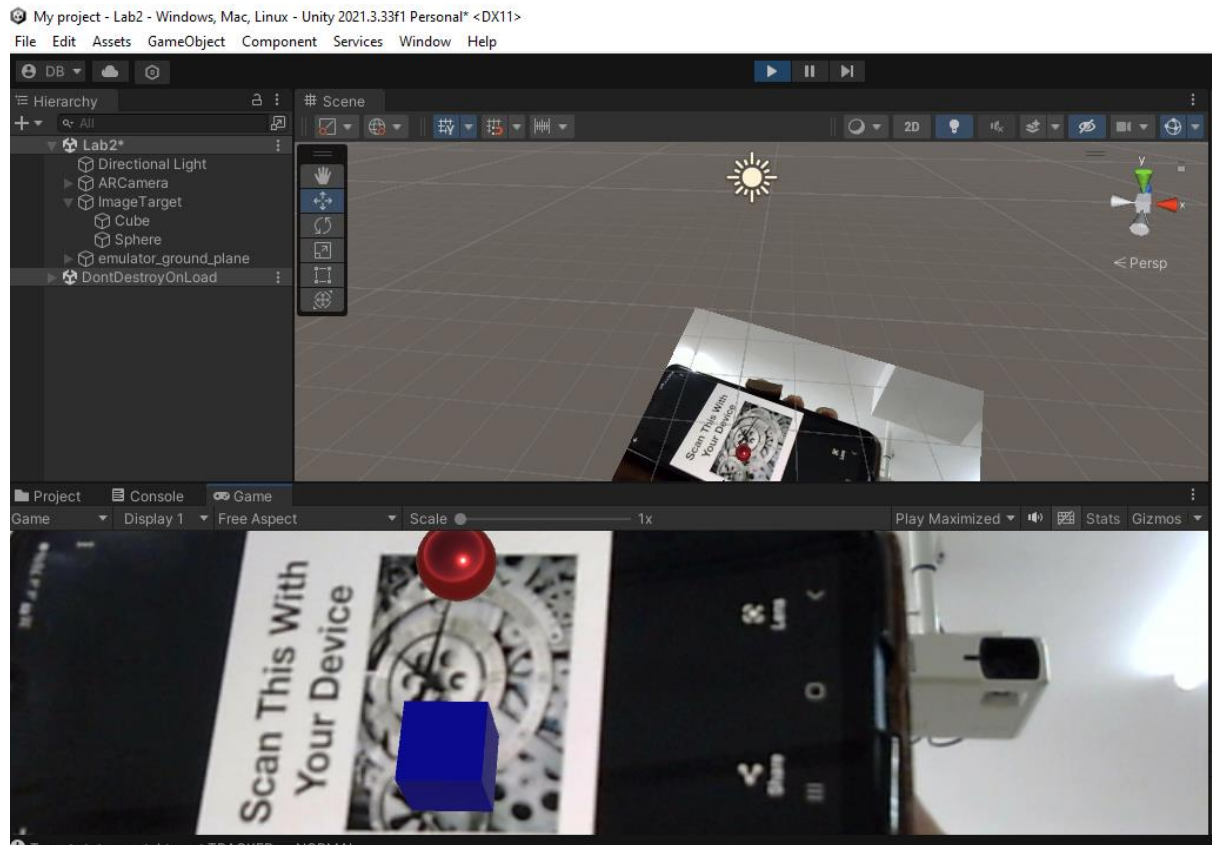
- Add cube and sphere in scene window and set up as below



for external webcam



- Now if u play while showing target image both gameobject will pop up



Step 9: create ui by creating buttons show and Hide



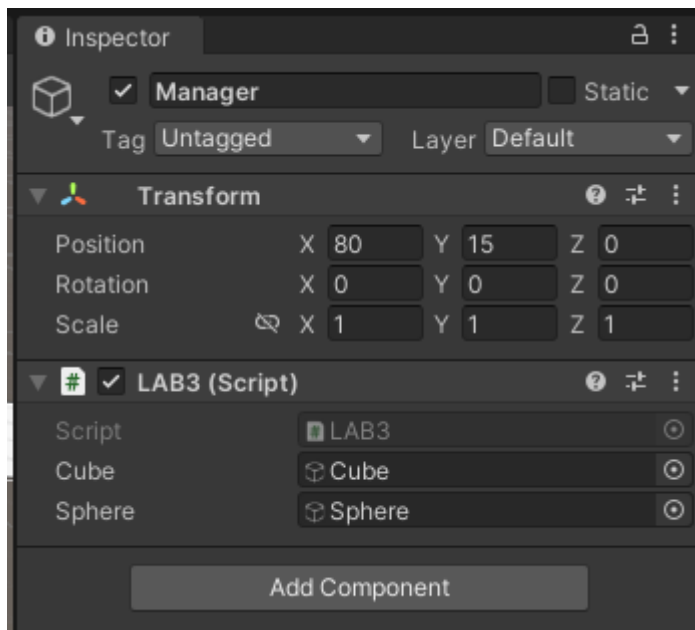
Step 10: create script and attach to empty game object manager and give reference of cube and sphere

```
using UnityEngine;

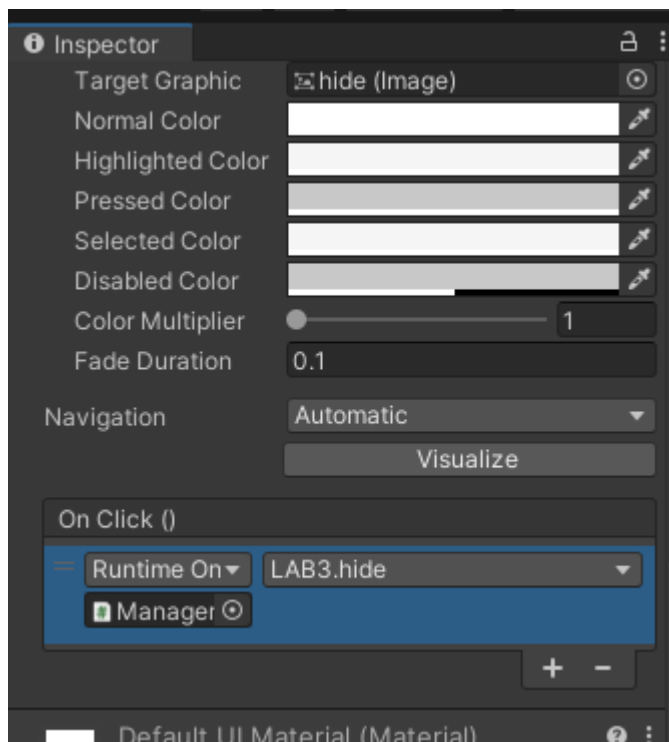
public class LAB3 : MonoBehaviour
{
    public GameObject cube;
    public GameObject sphere;
    void Start()
    {
        sphere.SetActive(false);
    }

    // Update is called once per frame
    void Update()
    {
        cube.transform.Rotate(0,30,0);
    }
    public void show()
```

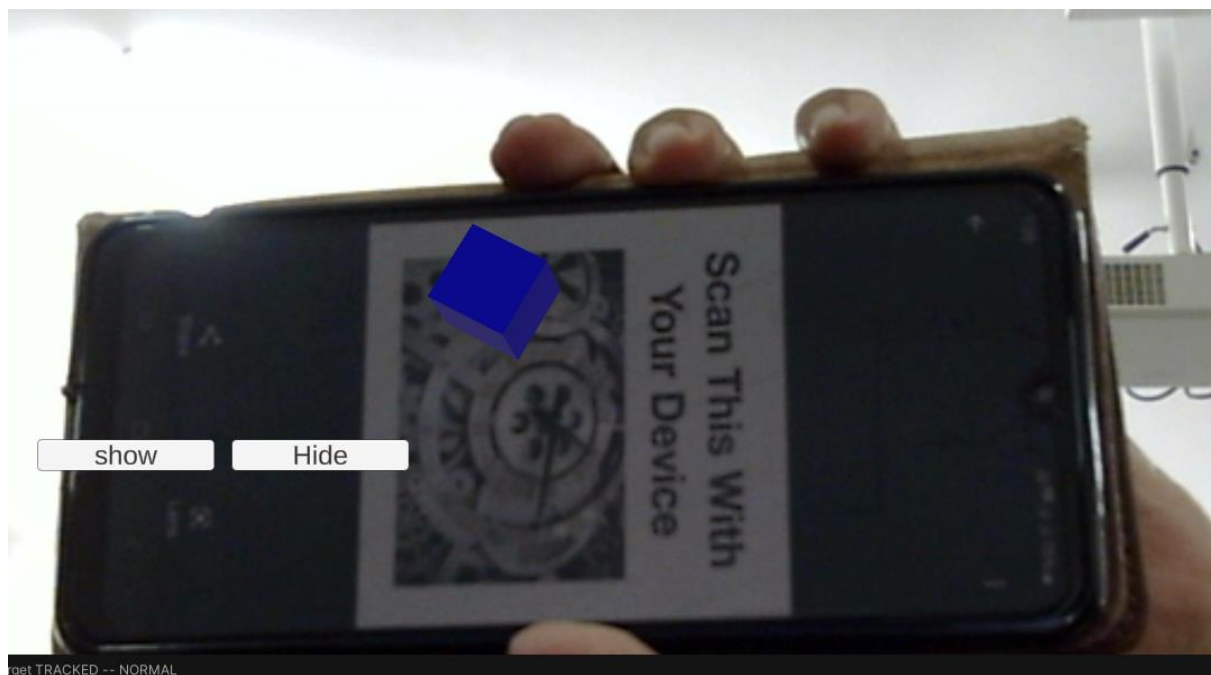
```
{  
    sphere.SetActive(true);  
}  
public void hide()  
{  
    sphere.SetActive(false);  
}  
}
```

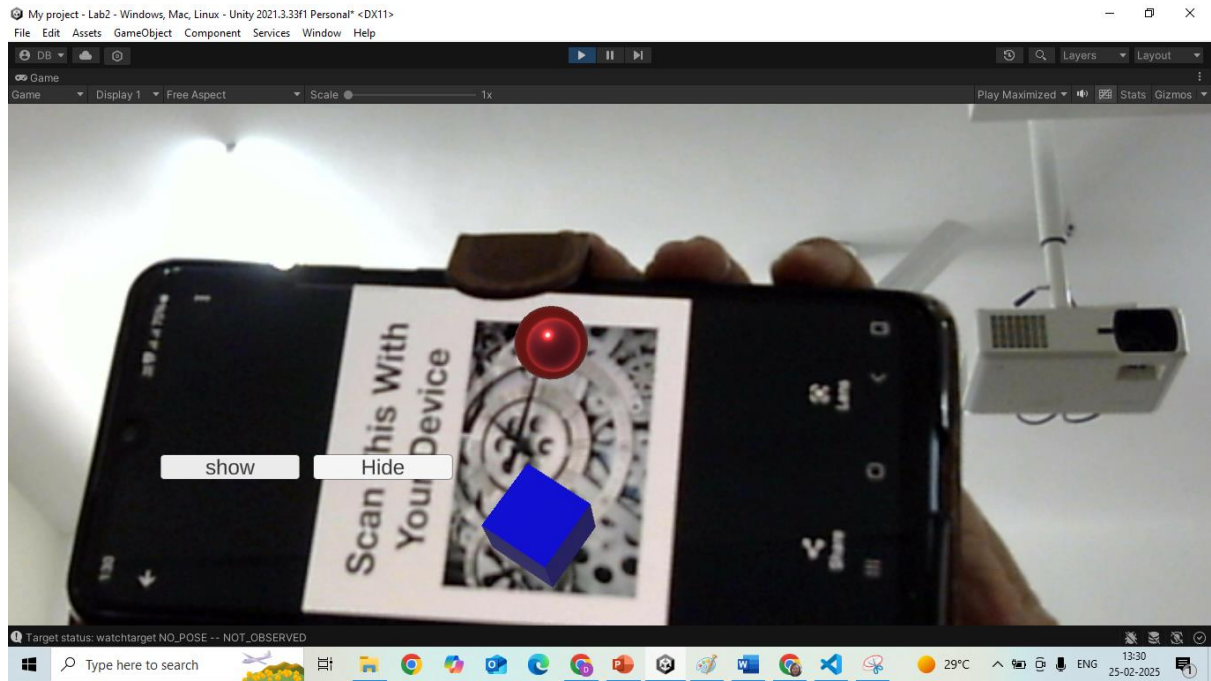


Step 11: Attach function to button



Output:





LAB 5: Create an AR application for Marker less application

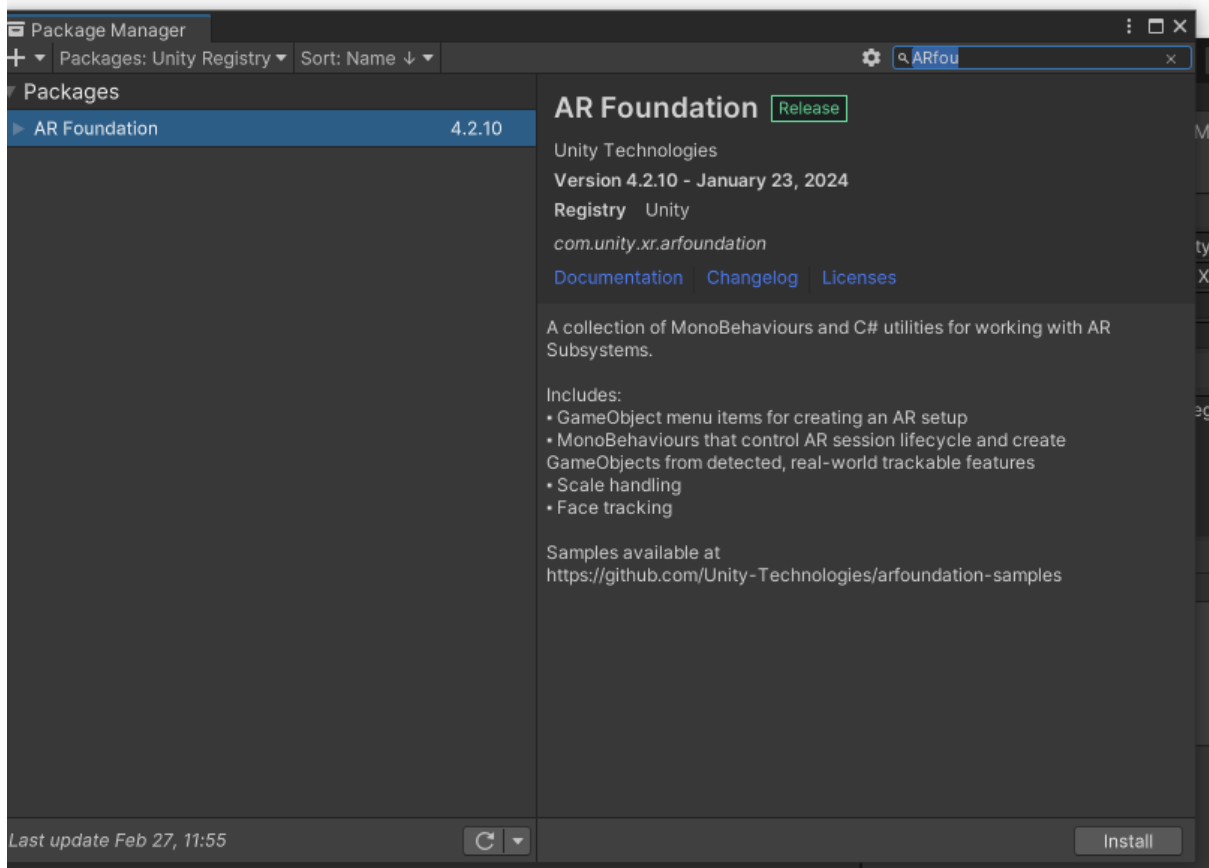
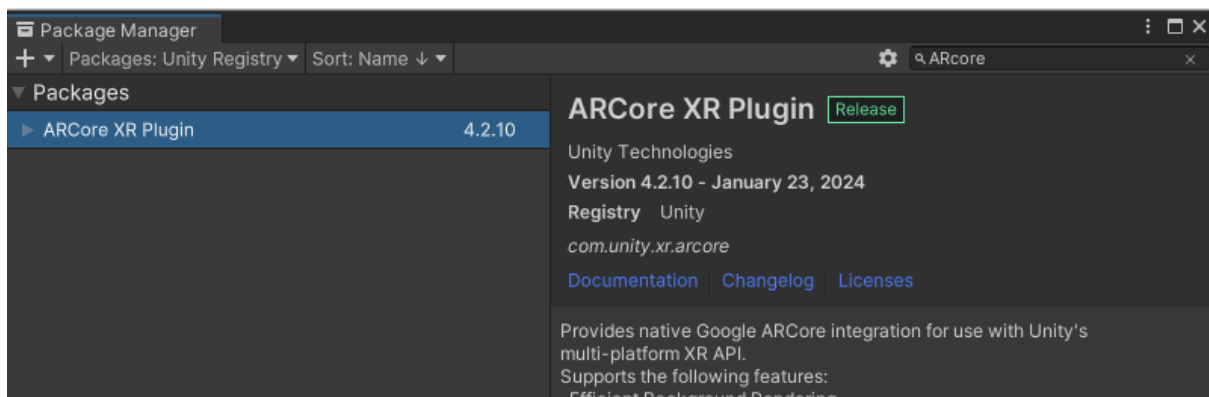
Sol:

Step 1: Create new scene LAB5

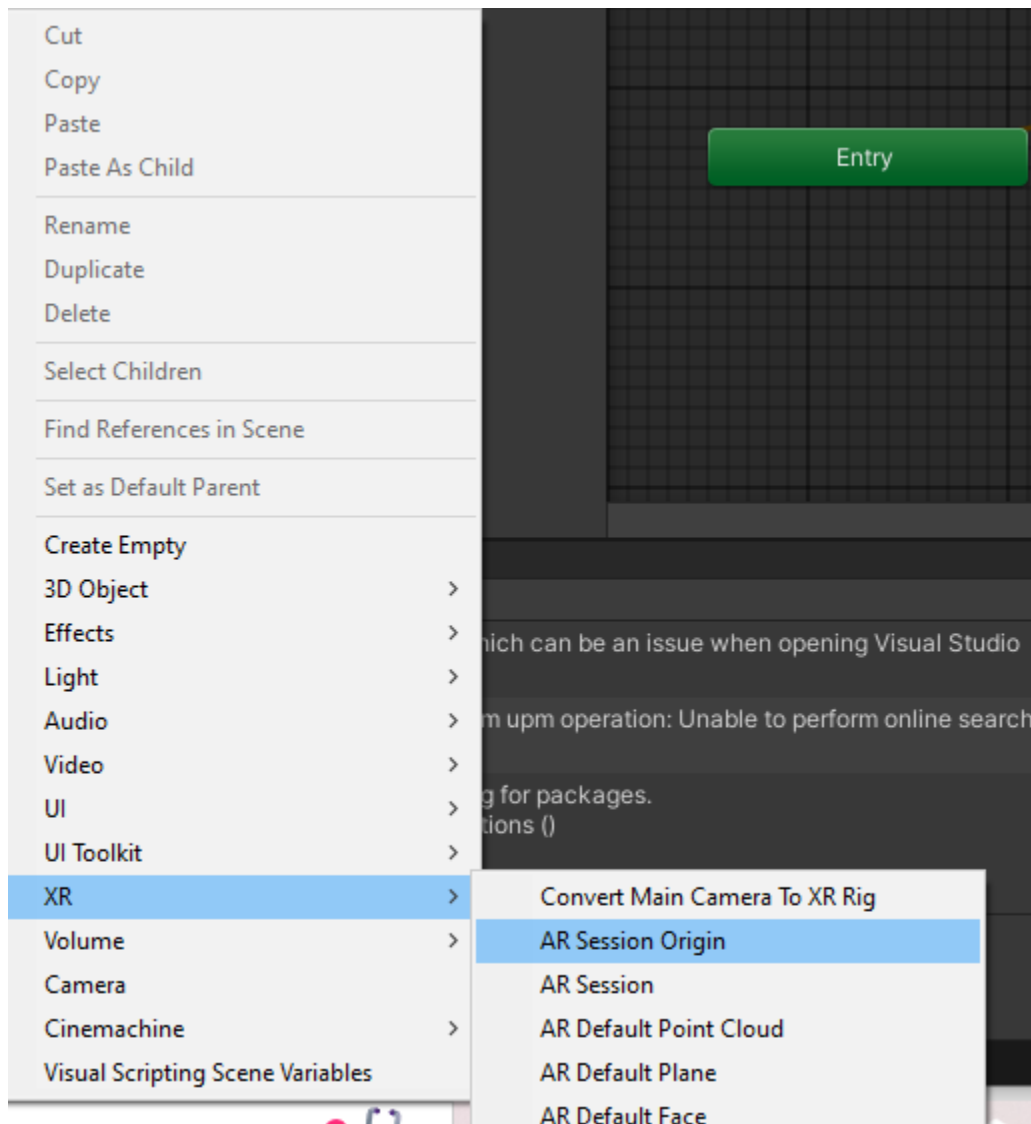
Step2: Delete Main Camera

Step3: Add package ARcore and AR Foundation

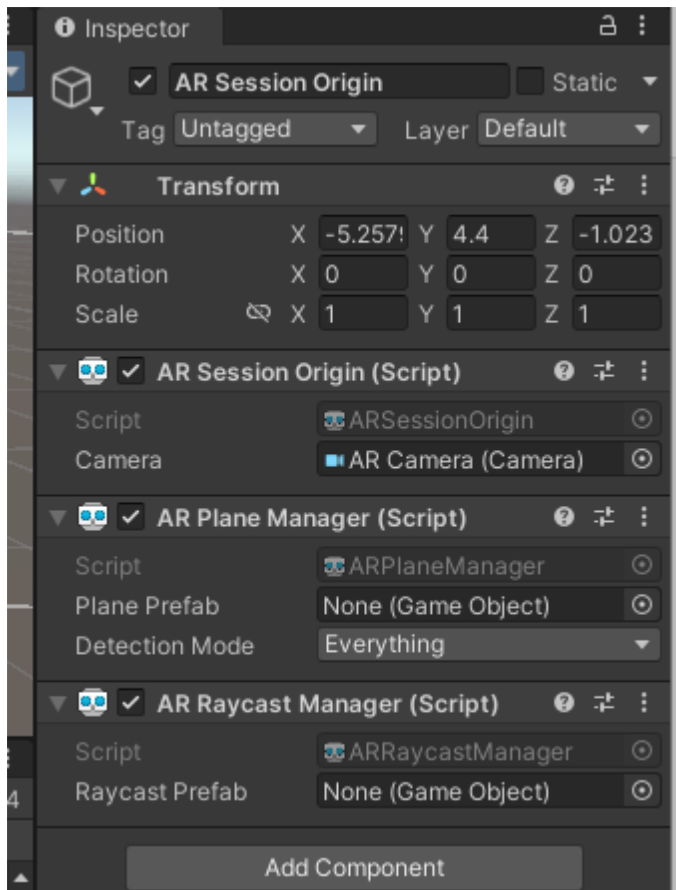
- **Windows-> Package manager->install ARcore XR Plugin and AR Foundation**



Step 4: Right Click- on hierarchy window> Add AR session origin or XR Session Origin

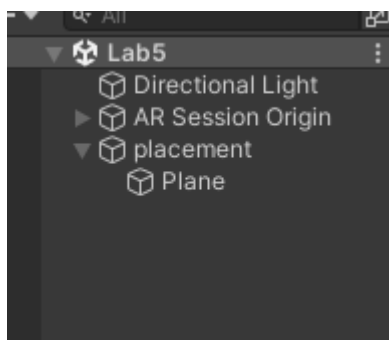


Step5: Select AR Session origin or XR origin and add component AR Raycast, AR plane Manager



Step 6: create empty gameobject name it placement and create plane as child of this placement

Uncheck mesh collider for plane



Step 7: create script name it PlacementIndicator and 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);
}
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);
    }
}
}

```

Step 8: create empty game object SpawnManager

- create script spawn_object attach to SpawnManager

```

using UnityEngine;

public class Spawn_object : MonoBehaviour
{
    public GameObject objectToSpawn;
    private PlacementIndicator placeIndicate;
    private GameObject spawnedObject; // Reference to the spawned object
    private float initialDistance; // Distance between fingers for scaling
    private Vector3 initialScale; // Initial scale of the object
    private bool isScaling = false; // Flag to check if scaling is active

    void Start()
    {
        placeIndicate = FindObjectOfType<PlacementIndicator>();
    }

    void Update()
    {

```

```

if (Input.touchCount > 0)
{
    Touch touch = Input.touches[0];

    // Check for object spawn on touch begin
    if (touch.phase == TouchPhase.Began && spawnedObject == null)
    {
        ShowObject();
    }

    // Handle scaling with pinch gesture
    if (Input.touchCount == 2)
    {
        ScaleObject();
    }
    // Handle rotation with single finger drag
    else if (Input.touchCount == 1 && spawnedObject != null)
    {
        RotateObject(touch);
    }
}

void ShowObject()
{
    spawnedObject = Instantiate(objectToSpawn, placeIndicate.transform.position,
placeIndicate.transform.rotation);
}

void ScaleObject()
{
    Touch touch1 = Input.GetTouch(0);
    Touch touch2 = Input.GetTouch(1);

    if (touch1.phase == TouchPhase.Began || touch2.phase == TouchPhase.Began)
    {
        initialDistance = Vector2.Distance(touch1.position, touch2.position);
        initialScale = spawnedObject.transform.localScale;
        isScaling = true;
    }

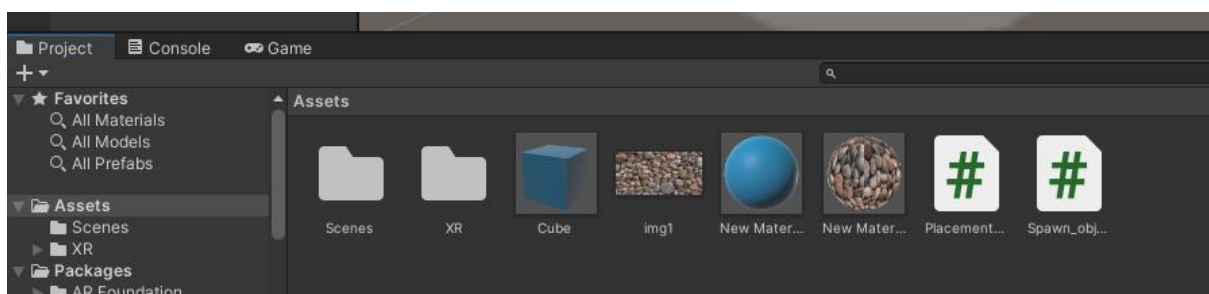
    if (touch1.phase == TouchPhase.Moved && touch2.phase == TouchPhase.Moved &&
isScaling)
    {
        float currentDistance = Vector2.Distance(touch1.position, touch2.position);
        float scaleFactor = currentDistance / initialDistance;
        spawnedObject.transform.localScale = initialScale * scaleFactor;
    }
}

void RotateObject(Touch touch)

```

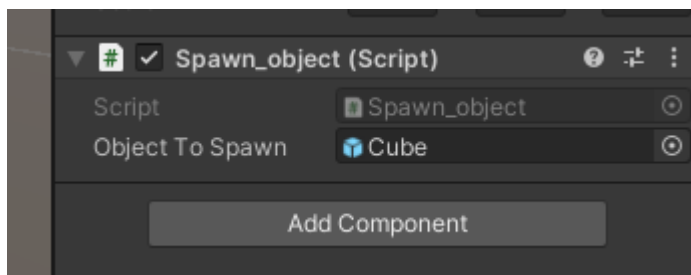
```
{  
    if (touch.phase == TouchPhase.Moved)  
    {  
        float rotationSpeed = 0.2f;  
        spawnedObject.transform.Rotate(Vector3.up, -touch.deltaPosition.x * rotationSpeed);  
    }  
}
```

step 9: create cube and make it prefabs by dragging cube from hierarchy to project and delete cube from hierarchy



Step 10:

Attach cube to script



Output:



Lab 6: Create 3D-Model using Three.js using A-Frame framework.

Sol:

Concept need for LAB:

- Three.js is a powerful, open-source JavaScript library that simplifies the process of creating and displaying 3D graphics and animations in web browsers using WebGL.
 - <https://threejs.org/>
- "A-Frame" refers to a web framework, written in JavaScript, that simplifies the creation of 3D and WebXR scenes, including AR experiences, using a HTML.
 - <https://aframe.io/docs/1.7.0/introduction/>
- <a-scene> is the main container where all 3D objects are placed.
 - It initializes the A-Frame environment.
- <a-entity position="0 1.6 4">: Creates an entity to hold the camera. The position attribute sets the initial position of the camera (0 meters on the X-axis, 1.6 meters on the Y-axis (eye level), and 4 meters back on the Z-axis).
- <a-camera></a-camera>: This tag defines the camera in the scene, through which the user views the VR environment. Since it's nested within the a-entity, it inherits the entity's position.
- **Directional Light** simulates a light source (like the sun) coming from (x=1, y=1, z=1).
- **Ambient Light** provides a general soft lighting across the scene.
- **cube** at (x=0, y=1, z=-3), rotated **45° around the Y-axis**.
- **Animation**: Rotates from 0 45 0 to 0 90 0 **every second (dur: 1000ms)** and loops infinitely.
 - **As code**

1. Create HTML file
2. Write code with CDN for A-Frame

```
<html>

<head>

  <script src="https://aframe.io/releases/1.4.0/aframe.min.js"></script>

</head>

<body>

  <a-scene>

    <!-- Camera and lighting -->

    <a-entity position="0 1.6 4">

      <a-camera></a-camera>

    </a-entity>

    <a-light type="directional" position="1 1 1" intensity="0.8"></a-light>

    <a-light type="ambient" intensity="0.5"></a-light>

    <!-- 3D Cube -->

    <a-box position="0 1 -3" rotation="0 45 0" color="red" animation="property: rotation; to: 0 90 0; loop: true; dur: 1000"></a-box>

    <a-box position="2 1 -3" rotation="0 0 45" color="green" animation="property: rotation; to: 0
```

```
90 0; loop: true; dur: 2000"></a-box>
```

```
<!-- Ground -->
```

```
<a-plane position="0 0 -4" rotation="-90 0 0" width="10" height="10" color="#7BC8A4"></a-plane>
```

```
</a-scene>
```

```
</body>
```

```
</html>
```

Output:



Lab 7: Create a GPS AR Camera and as per location detected 3D model should pop up

Sol:

1. Check the location of place from GPS and code accordingly
2. Here 3D model only visible if your GPS detected location is same as mentioned in code
3. Write HTML and Javascript code.

```
<!doctype html>
<html>
<head>
<title>A-Frame Geolocation</title>

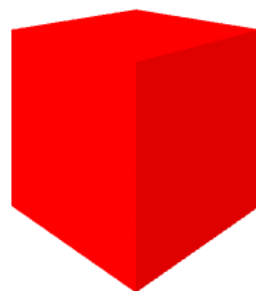
<script src="https://aframe.io/releases/1.2.0/aframe.min.js"></script>
<script>
document.addEventListener('DOMContentLoaded', function() {
function showShapes(position)
{
var currentLatitude = position.coords.latitude;
var currentLongitude = position.coords.longitude;
console.log("Latitude: " + currentLatitude);
console.log("Longitude: " + currentLongitude);

var locations = [
{ id: "box", lat:12.9957888,lon:77.6994816, threshold: 0.005 },
{ id: "cylinder", lat: 12.90509057, lon: 77.55971556, threshold: 0.005 },
{ id: "sphere", lat: 12.9564672,lon: 77.594624,threshold: 0.005}
];
locations.forEach(location => {
var shape = document.querySelector(`#${location.id}`);
if (shape && Math.abs(currentLatitude - location.lat) < location.threshold &&
Math.abs(currentLongitude - location.lon) < location.threshold) {
shape.setAttribute('visible', true);
}
});
}
function locationError(error) {
console.error("Error getting location: ", error);
document.getElementById('currentLocation').innerHTML =
`Error getting location: ${error.message}`;
}
function getLocation() {
if (navigator.geolocation) {
navigator.geolocation.getCurrentPosition(showShapes, locationError, { enableHighAccuracy: true,
timeout: 10000, maximumAge: 0 });
} else {
document.getElementById('currentLocation').innerHTML =
"Geolocation is not supported by this browser.";
}
}
getLocation();
```

```
});</script>
</head>
<body><h3 id="currentLocation">Fetching location...</h3>
<a-scene><a-box id="box" position="0 0.5 -3" rotation="0 45 0" color="red" visible="false"></a-
box>
<a-cylinder id="cylinder" position="2 0.5 -3" radius="0.5" height="1.5" color="blue"
visible="false"></a-cylinder>
<a-sphere id="sphere" position="-2 0.75 -3" radius="0.75" color="green" visible="false"></a-
sphere>
<a-camera gps-camera rotation-reader></a-camera>
</a-scene>
</body>
</html>
```

Output:

Current Location: 12.9957888, 77.6994816



Current Location: 12.9096941, 77.5733936



LAB 8: Create a Marker less AR to display 3D model in website or web-based application

Sol:

Step 1: download .Patt file and same .jpg file

<https://github.com/jeromeetienne/AR.js/blob/master/three.js/examples/marker-training/examples/pattern-files/pattern-letterA.patt>

Step 2: Write JavaScript and HTML code index.HTML

```
<!DOCTYPE html>
<html>
  <head>
    <!-- Include A-Frame -->
    <script src="https://aframe.io/releases/1.2.0/aframe.min.js"></script>

    <!-- Include AR.js for A-Frame -->
    <script src="https://cdn.jsdelivr.net/gh/jeromeetienne/ar.js/aframe/build/aframe-
ar.min.js"></script>
  </head>
  <body style="margin: 0px; overflow: hidden;">
    <a-scene embedded arjs>
      <!-- Marker -->
      <a-marker type="pattern" url="pattern-letterA.patt">
        <a-box position="0 0.5 0" material="opacity: 0.5;"></a-box>
        <a-cylinder color="green" height="1.0" radius="0.5" position="1 0.5 0"></a-cylinder>
      </a-marker>

      <!-- Camera -->
      <a-entity camera></a-entity>
    </a-scene>
  </body>
</html>
```

Step 3: create server or deploy application for testing

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 folder

2. Install express

To check if already install:

npm list 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. Start Your Server

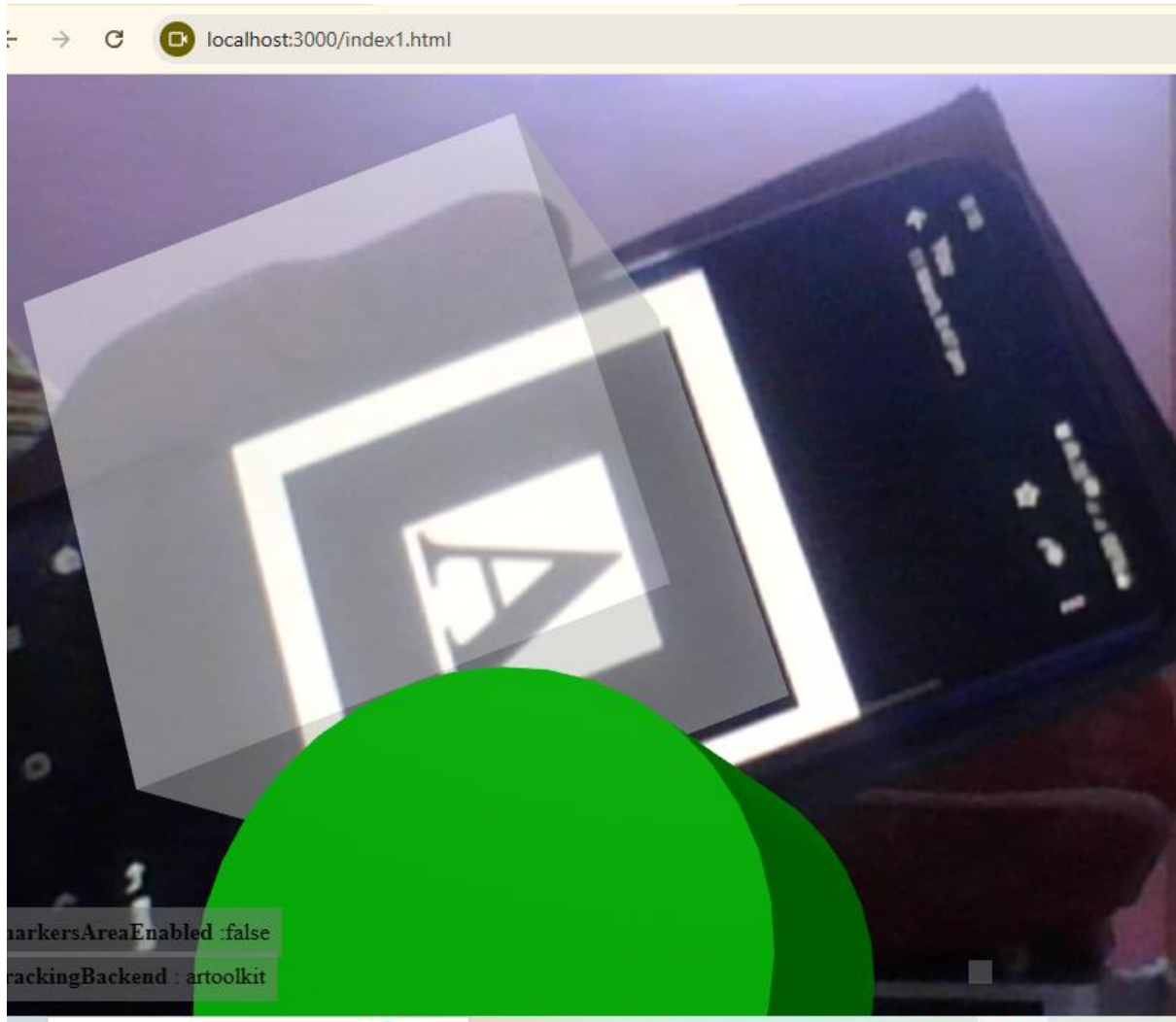
```
node server.js
```

//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
```

5. Type <http://localhost:3000/index.html> in the browser to run application

o/p:



Or Surge deployment if no web cam

//Deployment using surge

1. >npm install -g surge
2. >surge --version
3. >npm run bundle
4. In main folder check build file
5. Copy static_asset in build folder
6. Cd build
7. Type surge
8. Type enter
9. Type mail id and password

```
Welcome to surge! (surge.sh)
Login (or create surge account) by entering email & password.

    email: chirpy2710@gmail.com
    password:

Running as chirpy2710@gmail.com (Student)

    project: D:\MyVR\First\build\
    domain: excited-stream.surge.sh

Aborted - you do not have permission to publish to excited-stream.surge.s
```

10. Verify account

11. gain type surge then enter again enter

```
D:\MyVR\First>cd C:\Users\MY PC\AppData\Roaming\npm-cache\_logs\
    project: D:\MyVR\First\build\
    domain: neighborly-bead.surge.sh
    upload: [ ] 1% eta: 383.2s (29 files, 72302414 bytes)
C:\Users\MY PC\AppData\Roaming\npm-cache\_logs>npm run build

Domain is : neighborly-bead.surge.sh
```

For latest node version

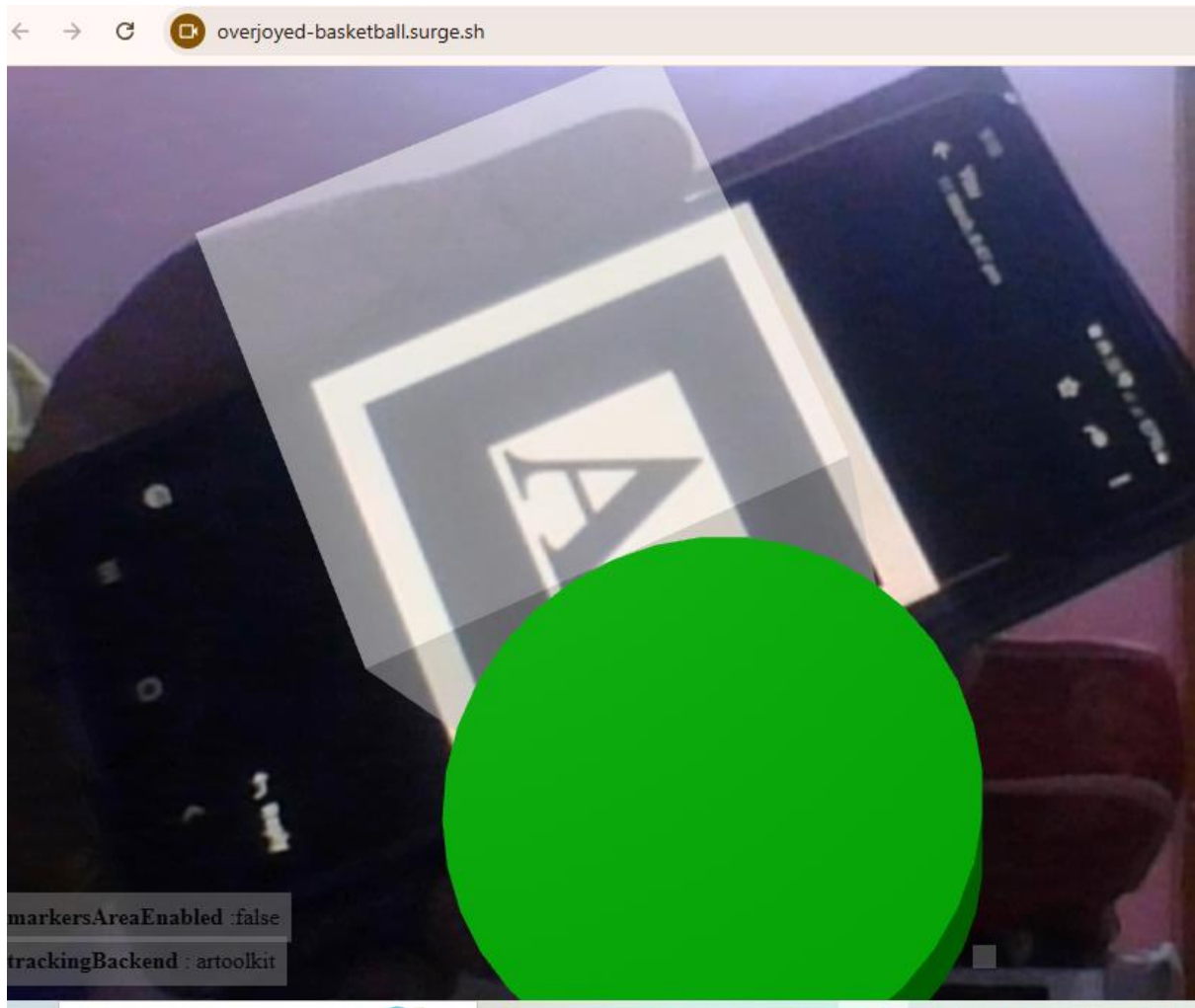
- npm init -y
- npm update
- npm install surge@latest --save
- webpack file
- index.js
- npm install webpack-cli --save-dev
- npm install html-webpack-plugin --save-dev
- npm run bundle or npx webpack
- Type surge

Edit: package.json

Go to build folder dist from cmd

Type surge and copy .sh URL with https://

Output



LAB-9: Create a Marker based AR to display 3D model in GLTF Format in website or web-based application

Sol:

Concept:

glTF (Graphics Library Transmission Format or GL Transmission Format and formerly known as WebGL Transmissions Format or WebGL TF) is a standard file format for three-dimensional scenes and models.

GLB is a file format for 3D models that includes textures, materials, and animations. It's a binary version of the GL Transmission Format (glTF) file.

1. Download GLTF model from website
2. Create index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple A-Frame Scene</title>
  <script src="https://aframe.io/releases/1.2.0/aframe.min.js"></script>
  <style>
    /* Ensure the body takes full height */
    body, html {
      margin: 0;
      padding: 0;
      height: 100%;
    }

    /* Style the button to make it visible and positioned correctly */
    button {
      position: absolute;
      top: 20px;
      left: 50%;
      transform: translateX(-50%);
      padding: 10px 20px;
      font-size: 16px;
      background-color: #4CC3D9;
      border: none;
      color: white;
      border-radius: 5px;
      cursor: pointer;
      z-index: 10; /* Ensure it's above the scene */
    }

    /* Add hover effect for the button */
    button:hover {
      background-color: #3a9ca1;
    }
  </style>
</head>
<body>
  <div>
    <img alt="3D model" data-bbox="124 363 697 905"/>
    <button>Click here to view the 3D model</button>
  </div>
</body>
</html>
```

```

    }
  </style>
</head>
<body>
  <a-scene>
    <a-assets>
      <a-asset-item id="value" src="Nike.glb"></a-asset-item>
    </a-assets>

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

    <a-entity id="model" gltf-model="#value" position="0 0 -5" rotation="0 45 0" scale="15 15 15"></a-entity>
  </a-scene>

  <!-- Button to trigger rotation -->
  <button onclick="rotateModel()">Rotate Model 45°</button>

  <script>
    // Function to rotate the model by 45 degrees each time
    function rotateModel() {
      const model = document.querySelector('#model');

      // Get the current rotation
      let currentRotation = model.getAttribute('rotation');

      // Increment the Y rotation by 45 degrees
      currentRotation.y += 45;

      // Apply the updated rotation
      model.setAttribute('rotation', currentRotation);
    }
  </script>
</body>
</html>

```

3. Create server if no webcam to test Deploy and check in mobile

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 folder

2. Install express

To check if already install:

`npm list 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}`);
});
```

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
```

2. Type `http://localhost:3000/index.html` in the browser to run application

Or deploy

Output:

