PLANE DETECTION

Step 1: Open Unity Hub,Create a New Project with 3D (Built-In Render Pipeline) template.

Step 2: In the Top Left corner click on Windows < Package Manager,

1) In the Package Manager on the top left click the drop down menu and click on Unity Registry and search for AR in the Search Box .
2) Now Install the AR Foundation and ARCore XR Plugin
3) After installing, close the window and go to File < Build Settings in the Top Left of Unity window.

Step 3: In the Build Settings,

1) Click the Player Settings
2) Click on Android Logo and go for other settings
3) Uncheck the Auto Graphics API and click on the Vulkan then the minus (-) symbol so that it is removed and also uncheck the Multithreaded Rendering.
4) Scroll down and select the Android version 7.0 in the drop down menu of Minimum API level
5) Click the drop down menu in Scripting Backend and click IL2CPP and check the ARM64 box
6) In the left click on XR Plug-in Management and in Android tab check the ARCore and close this window
7) Click on Android and click Switch Platform after finishing close this window

Step 4: Now Delete the Main Camera from the Hierarchy.

1. Right click in the Hierarchy and click XR < AR Session Origin
2. Right click in the Hierarchy and click XR < AR Session
3. Right click in the Hierarchy and click XR < AR Default Plane

Step 5: Drag the AR Default Plane to Assets and delete the AR Default Plane in the Hierarchy

Step 6: Click on AR Session Origin and Go to Inspector Panel and click on Add component and search for the AR Plane Manager and click it.

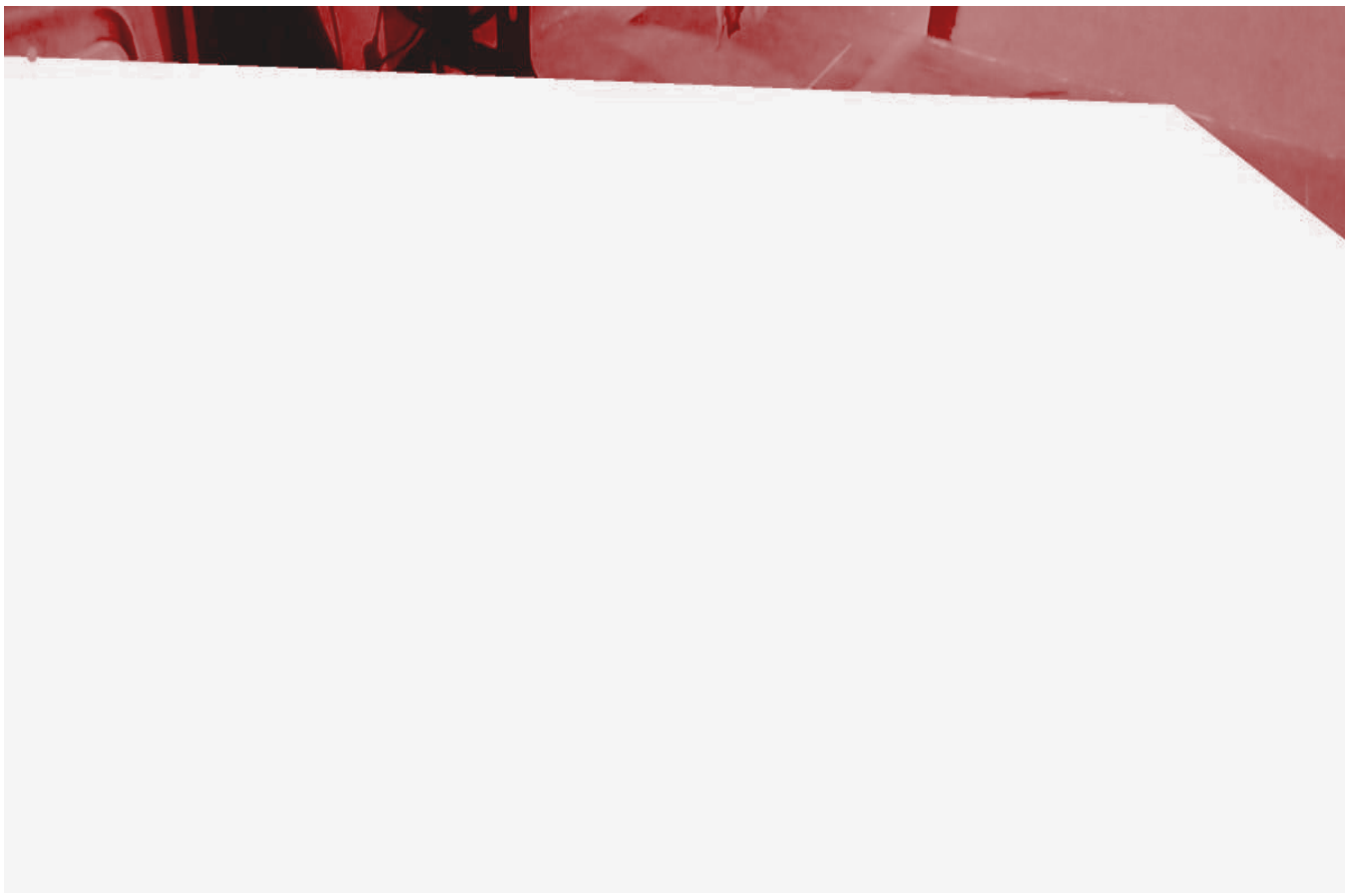
Step 7: Drag the AR Default Plane to the Plane Prefab in the Inspector Panel.

Step 8: Make sure the Detection Mode is checked for the following and save the file (Ctrl+S)

Step 9: Go to File < Build Settings in the Top Left of Unity window.

Step 10: Click on the Build option and the Save As dialog box will open choose your desired directory and name of the file and click save.

Output:



OBJECT PLACEMENT

Step 1. Create or Open a Unity Project

- Open Unity Hub → New Project → Choose 2D or 3D → Name it → Create.

Step 2. Add or Create the Object

- Right-click in the Hierarchy → 3D Object > Cube (for 3D)
  or → 2D Object > Sprite (for 2D).
- Or drag and drop a prefab into the scene.

Step 3. Set the Plane (Reference Area)

- 3D: Use Plane from 3D Object > Plane.
- 2D: Use the Camera view as your "plane."

Step 4. Set Position via Inspector

- Select the object in Hierarchy.
- In Inspector → Transform:
    - Set Position (X, Y, Z) to where you want the object.

Example for 3D: (0, 0, 0) places the object at the origin.
Example for 2D: (0, 0, 0) places it at screen center.

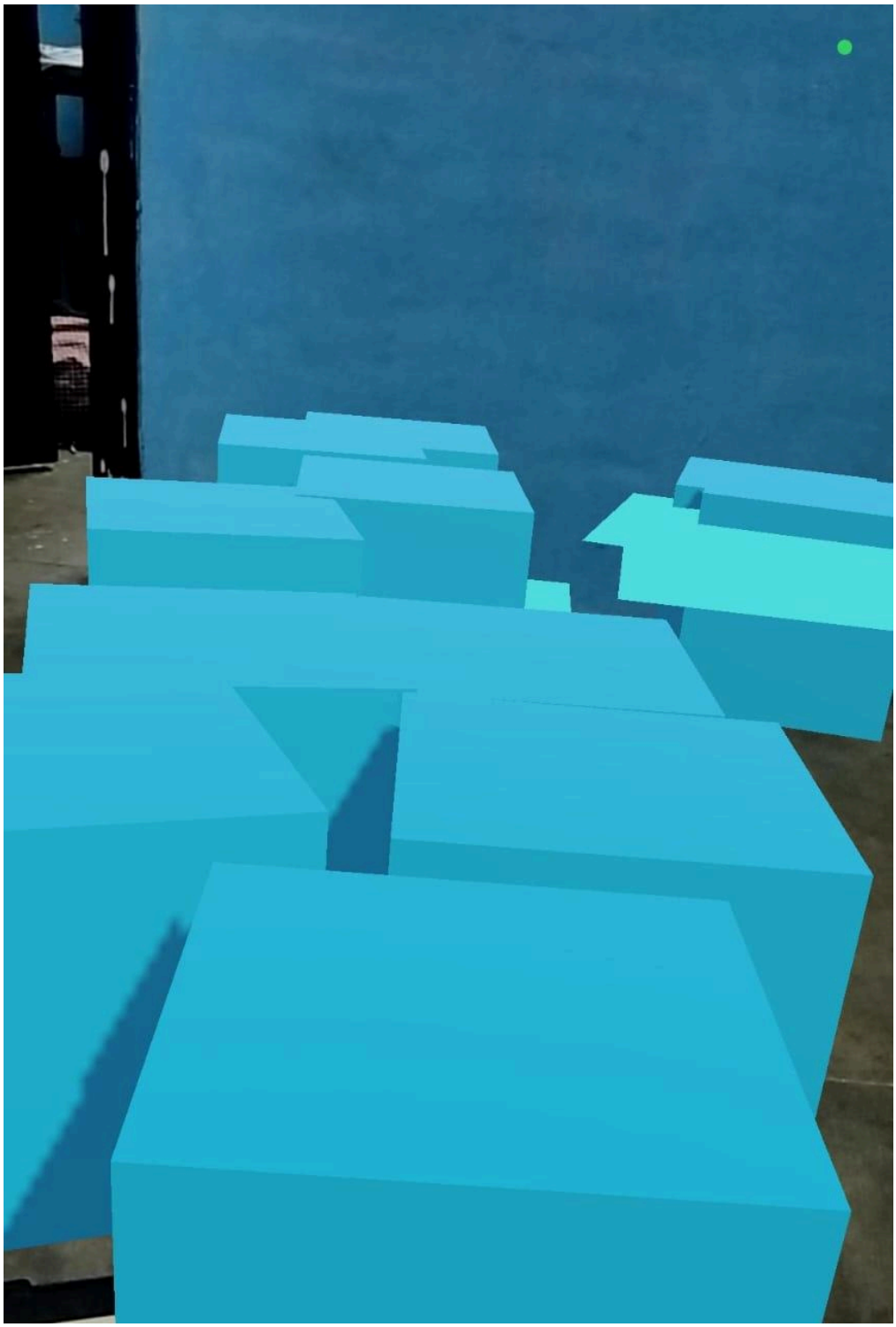
Step 6. Play & Test

- Click Play to see object placement.
- Adjust positions in real time if needed.

Step 7. (Optional) Label or Group Objects

- Rename in Hierarchy.
- Use Empty GameObjects as folders/groups.

Output:



UI SLIDER

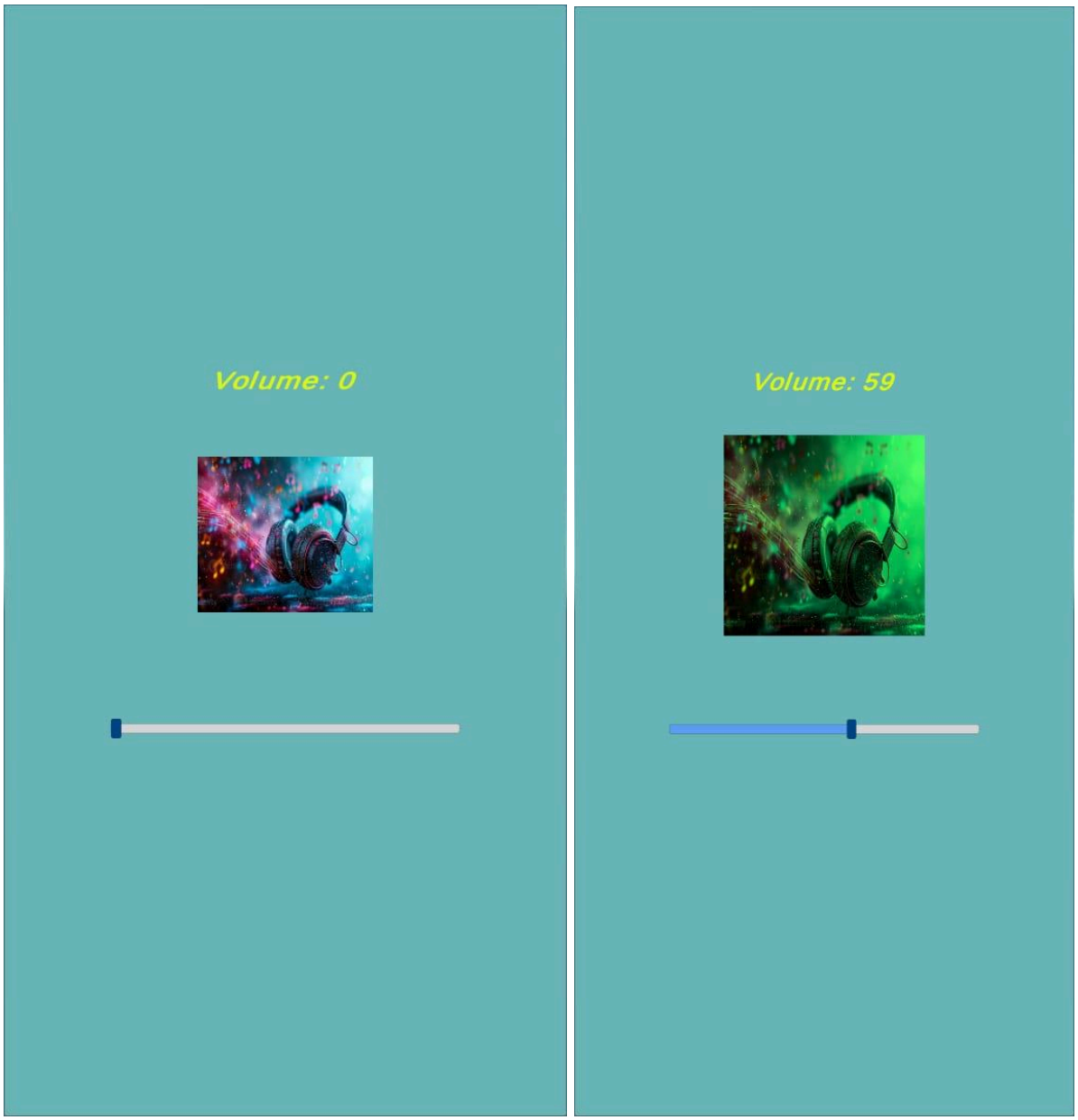
Step 1. Create a New Unity Project (2D or 3D).
Step 2. Add a Canvas (Right-click in Hierarchy → UI → Canvas).
Step 3. Add UI elements (Button, Slider, Toggle, Scrollbar, Dropdown) from GameObject → UI.
Step 4. Attach this script to an empty GameObject in your scene.

Source Code:

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class MainMenuUI : MonoBehaviour
{
public Button startButton, exitButton;
public Slider volumeSlider;
public Toggle fullscreenToggle;
public Scrollbar scrollBar;
public Dropdown resolutionDropdown;
void Start()
{
startButton.onClick.AddListener(StartGame);
exitButton.onClick.AddListener(ExitGame);
volumeSlider.onValueChanged.AddListener(AdjustVolume);
fullscreenToggle.onValueChanged.AddListener(SetFullscreen);
resolutionDropdown.onValueChanged.AddListener(ChangeResolution);
}
void StartGame()
{
Debug.Log(&quot;Start Button Clicked! Load Game...&quot;);
}
void ExitGame()
{
Debug.Log(&quot;Exit Button Clicked! Quitting...&quot;);
Application.Quit();}
void AdjustVolume(float value)
{
Debug.Log(&quot;Volume: &quot; + value);
AudioListener.volume = value;
}
void SetFullscreen(bool isFull)
{
Screen.fullScreen = isFull;
Debug.Log(&quot;Fullscreen: &quot; + isFull);
```

```
}
void ChangeResolution(int index)
{
if (index == 0) Screen.SetResolution(1920, 1080, Screen.fullScreen);
else if (index == 1) Screen.SetResolution(1280, 720, Screen.fullScreen);
else if (index == 2) Screen.SetResolution(800, 600, Screen.fullScreen);
Debug.Log(&amp;quot;Resolution Changed&amp;quot;);
}}
```

Output:



UI -FADE IN FADE OUT

Step 1:Create a New Unity Project (2D or 3D).

Step 2:Create UI Elements (GameObject → UI):

- Canvas (Automatically added with UI elements)
- Panel (For fade-in effect)
- Button (For event handling)
- Text (For hover effect)
- Image with Mask (For masking)

Step 3:Attach Script to an Empty GameObject

Step 4:Assign UI Elements in the Inspector

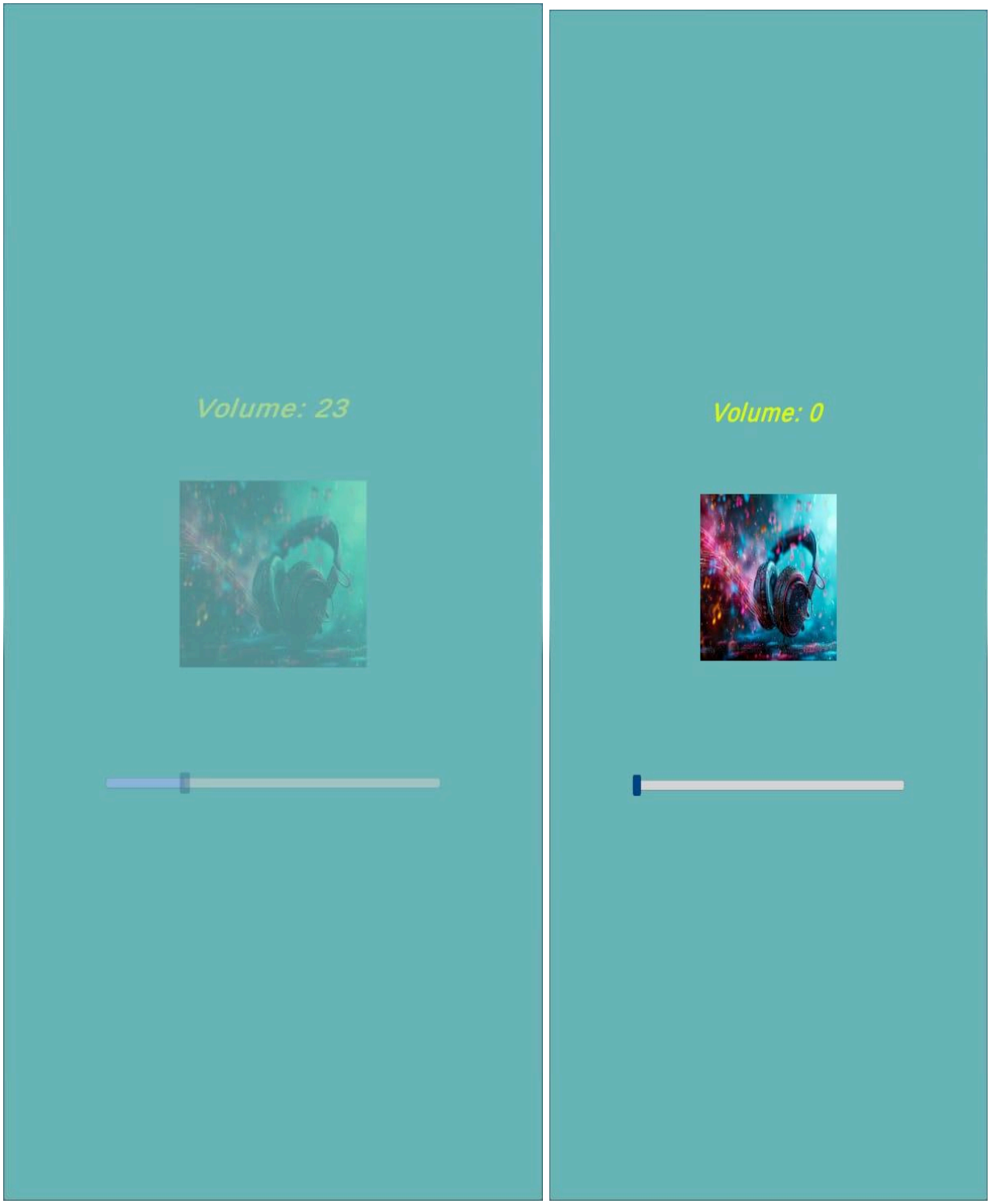
Step 5: Run the Scene

Source Code:

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using System.Collections;
public class UIManager : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler
{public Button myButton;
public Text labelText;
public Mask maskComponent;
public CanvasGroup uiPanel;
public Vector3 normalScale = Vector3.one;
public Vector3 hoverScale = new Vector3(1.2f, 1.2f, 1.2f);
public float animationSpeed = 0.2f;
void Start()
{
myButton.onClick.AddListener(OnButtonClick);
StartCoroutine(FadeInUI());}
void OnButtonClick()
{
Debug.Log(&amp;quot;Button Clicked!&amp;quot;);
}
IEnumerator FadeInUI()
```

```
{
uiPanel.alpha = 0;
while (uiPanel.alpha &lt; 1)
{
uiPanel.alpha += Time.deltaTime;
yield return null;}}
public void OnPointerEnter(PointerEventData eventData){
labelText.color = Color.red; // Hover Effect
StopAllCoroutines();
StartCoroutine(ScaleOverTime(hoverScale));
}
public void OnPointerExit(PointerEventData eventData){
labelText.color = Color.white; // Reset Color
StopAllCoroutines();
StartCoroutine(ScaleOverTime(normalScale));}
IEnumerator ScaleOverTime(Vector3 targetScale)
{
Vector3 startScale = myButton.transform.localScale;
float time = 0;
while (time &lt; animationSpeed){
myButton.transform.localScale = Vector3.Lerp(startScale, targetScale, time /
animationSpeed);
time += Time.deltaTime;
yield return null;
}
myButton.transform.localScale = targetScale;
}
void Update(){
if (Input.GetKeyDown(KeyCode.Space))
{
maskComponent.enabled = !maskComponent.enabled;
Debug.Log(&quot;Mask Toggled!&quot;);}}}
```

Output:



OBJECT ROTATION

1. Create a UI Canvas:
   - In Hierarchy, right-click → UI → Canvas.
   - Select the Canvas and in the Inspector, set UI Scale Mode to Scale With Screen Size.

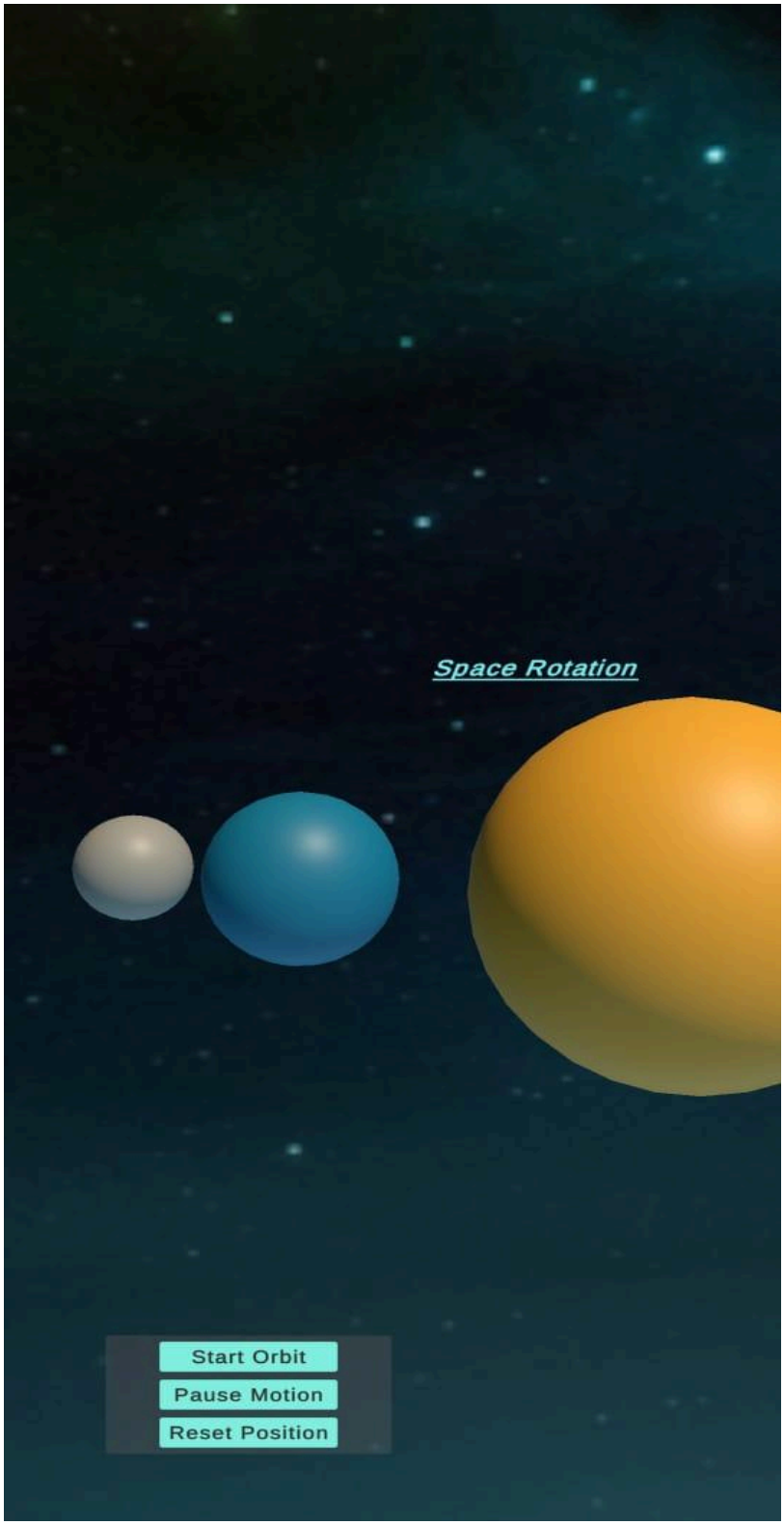
2. Add Two Buttons:
   - Right-click Canvas → UI → Button (do this twice for Left and Right buttons).
   - Rename them LeftButton and RightButton.
   - Select each button and change their Position:
   - Left Button: Position X = -400 (move to the left).
   - Right Button: Position X = 400 (move to the right).
   - Change Button Text:
   - Expand each button → Click Text → Change text to "LEFT" and "RIGHT".

Source Code:

```
using UnityEngine;
using UnityEngine.UI;
public class OrbitController : MonoBehaviour
{
    public GameObject sun;
    public GameObject earth;
    public GameObject moon;
    public Button startOrbitBtn;
    public Button pauseMotionBtn;
    public Button resetBtn;
    private bool isOrbiting = false;
    private Vector3 initialEarthPos;
    private Vector3 initialMoonPos;
    void Start()
    {   initialEarthPos = earth.transform.position;
        initialMoonPos = moon.transform.position;
        startOrbitBtn.onClick.AddListener(StartOrbit);
        pauseMotionBtn.onClick.AddListener(PauseOrbit);
        resetBtn.onClick.AddListener(ResetPositions);
```

```
    }
    void Update()
    {
        if (isOrbiting)
        {
             earth.transform.RotateAround(sun.transform.position, Vector3.up, 20 * Time.deltaTime);
            moon.transform.RotateAround(earth.transform.position, Vector3.up, 60 *
Time.deltaTime);
        }}
    void StartOrbit()
    {
        isOrbiting = true;
    }
    void PauseOrbit()
    {
        isOrbiting = false;
    }
    void ResetPositions()
    {
        isOrbiting = false;
        earth.transform.position = initialEarthPos;
        moon.transform.position = initialMoonPos;
    }
}
```

Output:



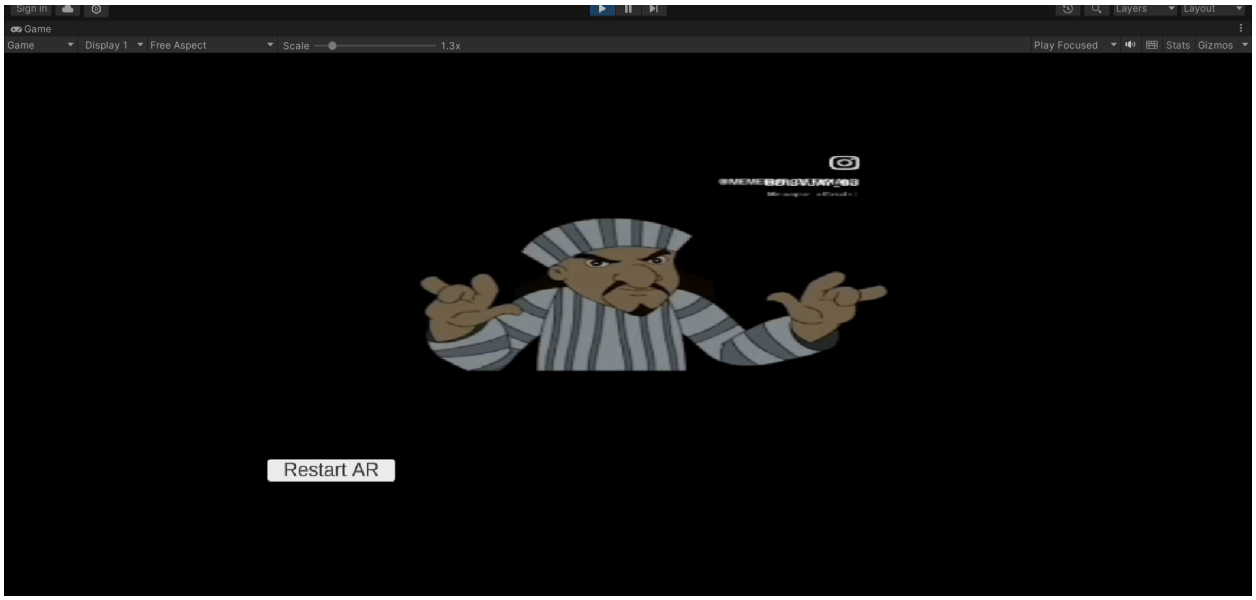
SINGLETON SCRIPT

Source Code:

```
using UnityEngine;
using UnityEngine.Video;

public class GameManager : MonoBehaviour
{
    public static GameManager Instance { get; private set; }
    private VideoPlayer videoPlayer;

    private void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
        }
        else
        {
            Destroy(gameObject);
            return;
        }

        videoPlayer = FindObjectOfType<VideoPlayer>();

        if (videoPlayer == null)
        {
            Debug.LogError("VideoPlayer not found in the scene!");
        }
    }
```

```
    public void RestartGame()
    {
        if (videoPlayer != null)
        {
            videoPlayer.Stop(); // Stop the video
            videoPlayer.frame = 0; // Reset to first frame
            videoPlayer.Play(); // Play again from the start
            Debug.Log("Video Restarted!");
        }
        else
        {
            Debug.LogError("No VideoPlayer found!");
        }
    }
}
```

Output:



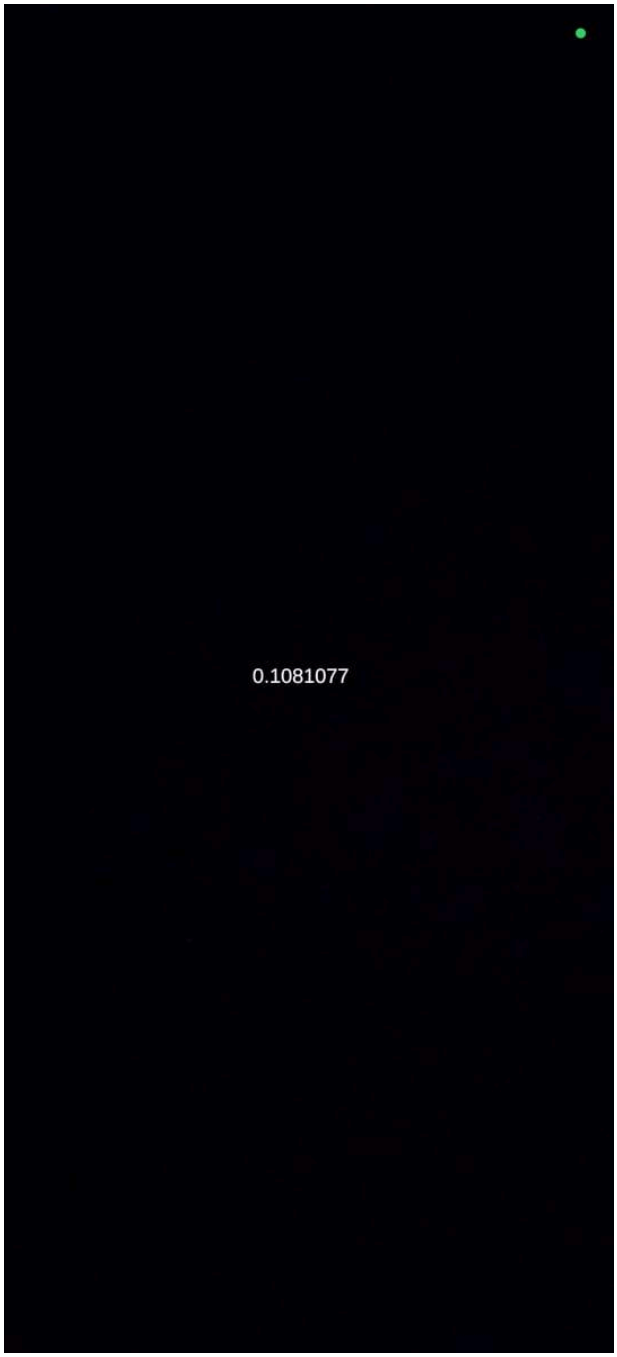
LIGHT ESTIMATION

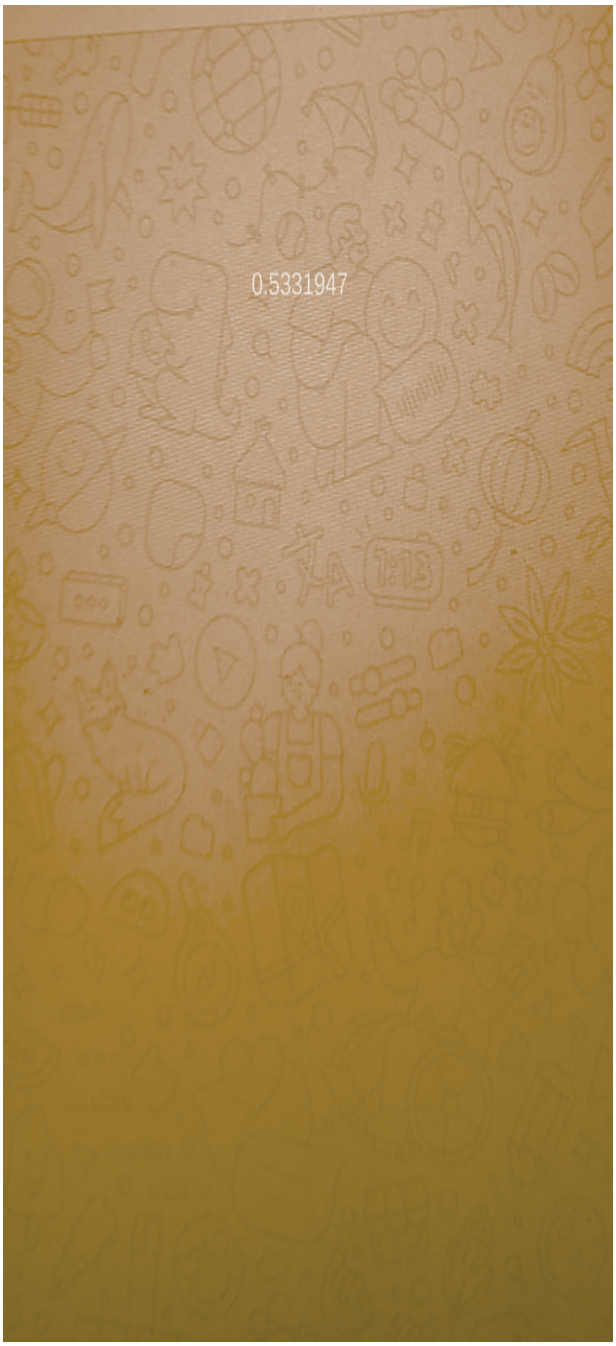
Source code:

```
using UnityEngine;
using UnityEngine.XR.ARFoundation;

public class LightEstimationManager : MonoBehaviour
{
    [SerializeField] private Light sceneLight;
    private ARCameraManager arCameraManager;

    void Start()
    {
        arCameraManager = FindObjectOfType<ARCameraManager>();
        if (arCameraManager != null)
        {
            arCameraManager.frameReceived += OnCameraFrameReceived;
            Debug.Log("Subscribed to frameReceived event.");
        }
        else
        {
            Debug.LogError("ARCameraManager not found.");
        }
    }

    private void OnCameraFrameReceived(ARCameraFrameEventArgs args)
    {
        // Adjust light intensity based on estimated brightness
        if (args.lightEstimation.averageBrightness.HasValue)
        {
            sceneLight.intensity = args.lightEstimation.averageBrightness.Value;
            Debug.Log($"Light intensity adjusted to {sceneLight.intensity}.");
        }
```

```
        else
        {
            Debug.Log("No brightness estimation data available.");
        }

        // Adjust light color temperature if available
        if (args.lightEstimation.averageColorTemperature.HasValue)
        {
            sceneLight.colorTemperature = args.lightEstimation.averageColorTemperature.Value;
            Debug.Log($"Light color temperature adjusted to {sceneLight.colorTemperature}.");
        }
        else
        {
            Debug.Log("No color temperature estimation data available.");
        }
    }

    private void OnDestroy()
    {
        if (arCameraManager != null)
        {
            arCameraManager.frameReceived -= OnCameraFrameReceived;
            Debug.Log("Unsubscribed from frameReceived event.");
        }
    }
}
```

Output:





INTERACTION CONTROLLER MODE

AddPhotoController.cs

```
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
using System.Collections.Generic;
public class AddPhotoController : MonoBehaviour {
public List photoPrefabs;
private ARRaycastManager raycastManager;
private List hits = new List();
private int selectedPhotoIndex = 0;
void Start()
{
raycastManager = FindObjectOfType();
}
public void OnAddPhotoButtonClicked() {
Vector2 screenCenter = new Vector2(Screen.width / 2f, Screen.height / 2f);
if (raycastManager.Raycast(screenCenter, hits, TrackableType.PlaneWithinPolygon))
{ Pose hitPose = hits[0].pose;
GameObject spawned = Instantiate(photoPrefabs[selectedPhotoIndex], hitPose.position,
hitPose.rotation);
spawned.AddComponent(); // For interaction } }
public void SetSelectedPhotoIndex(int index)
{
selectedPhotoIndex = index; } }
```

GestureController.cs

```
using UnityEngine;
using UnityEngine.EventSystems;
public class GestureController : MonoBehaviour
{
```

```
private float initialDistance;
private Vector3 initialScale;
private Quaternion initialRotation;
private float rotationSpeed = 0.2f;
void Update() {
if (Input.touchCount == 2 && !IsPointerOverUIObject())
{
Touch touch0 = Input.GetTouch(0);
Touch touch1 = Input.GetTouch(1); // Scale
if (touch1.phase == TouchPhase.Began)
{
initialDistance = Vector2.Distance(touch0.position, touch1.position);
initialScale = transform.localScale;
initialRotation = transform.rotation; }
else if (touch0.phase == TouchPhase.Moved || touch1.phase == TouchPhase.Moved)
{
float currentDistance = Vector2.Distance(touch0.position, touch1.position);
if (Mathf.Approximately(initialDistance, 0))
return;
float factor = currentDistance / initialDistance;
transform.localScale = initialScale * factor; // Rotate
Vector2 prevDir = (touch0.position - touch0.deltaPosition) - (touch1.position -
touch1.deltaPosition);
Vector2 currentDir = touch0.position - touch1.position;
float angle = Vector2.SignedAngle(prevDir, currentDir);
transform.rotation = initialRotation * Quaternion.Euler(0, -angle * rotationSpeed, 0);
} } }
private bool IsPointerOverUIObject()
{
if (EventSystem.current == null)
return false;
return EventSystem.current.IsPointerOverGameObject(Input.GetTouch(0).fingerId );
} }
```

Output:

AR FRAMEWORK

Source code:

```
using UnityEngine;
using UnityEngine.UI;
public class ARModeController : MonoBehaviour
{
[Header("UI Panels")]
public GameObject startupUI;
public GameObject scanUI;
public GameObject mainUI;
public GameObject placeObjectUI;
public GameObject nonARUI;
[Header("Buttons")]
public Button startButton;
void Start()
{
ShowStartupUI();
startButton.onClick.AddListener(StartARExperience);
}
void ShowStartupUI()
{
startupUI.SetActive(true);
scanUI.SetActive(false);
mainUI.SetActive(false);
placeObjectUI.SetActive(false);
nonARUI.SetActive(false);
}
void StartARExperience() {
startupUI.SetActive(false);
```

```
scanUI.SetActive(true);
} }
```

Create the Plane Detection Script

```
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
public class PlaneDetectionHandler : MonoBehaviour
{
public ARPlaneManager planeManager;
public GameObject scanUI;
public GameObject mainUI;
private bool planeDetected = false;
void OnEnable()
{
planeManager.planesChanged += OnPlanesChanged; }
void OnDisable() {
planeManager.planesChanged -= OnPlanesChanged;
}
void OnPlanesChanged(ARPlanesChangedEventArgs args)
{
 if (!planeDetected && planeManager.trackables.count > 0)
{ planeDetected = true;
scanUI.SetActive(false);
mainUI.SetActive(true);
} } }
```

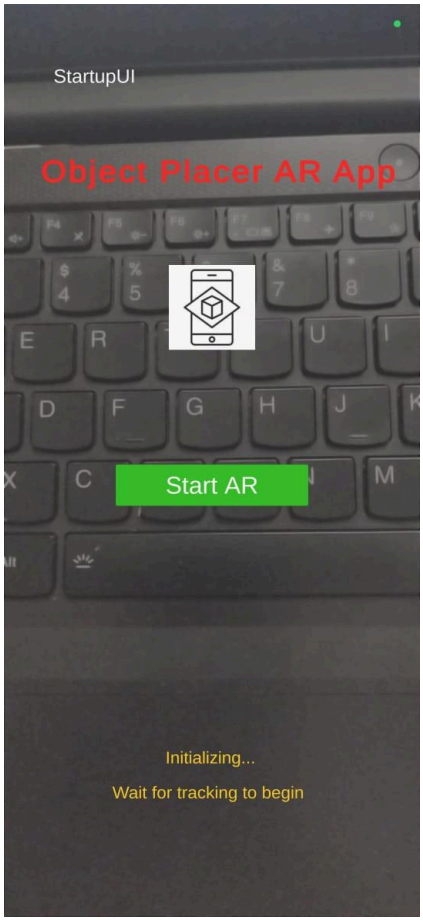
Object Placement Script

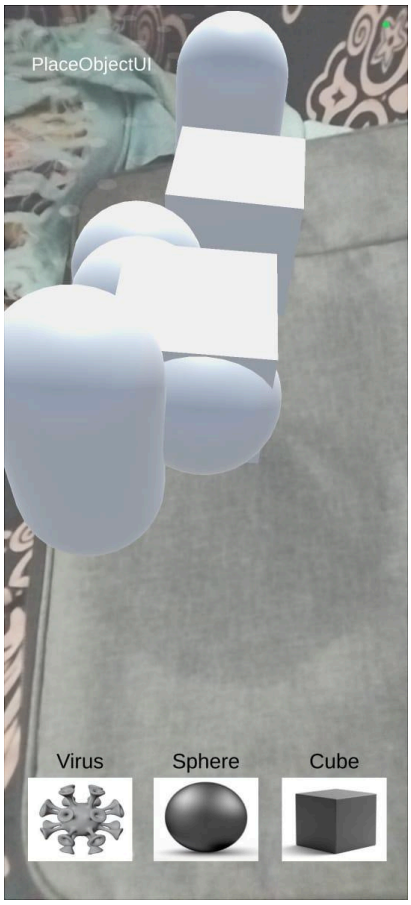
```
using System.Collections.Generic;
using UnityEngine;
```

```
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
public class ObjectPlacer : MonoBehaviour
{
public ARRaycastManager raycastManager;
public GameObject cubePrefab;
public GameObject spherePrefab;
public GameObject virusPrefab;
public GameObject scanUI;
public GameObject placeObjectUI;
private GameObject selectedPrefab;
private static List hits = new List();
private bool hasPlacedUI = false;
public void SelectCube()
{
selectedPrefab = cubePrefab;
}
public void SelectSphere()
{
selectedPrefab = spherePrefab;
}
public void SelectVirus() {
selectedPrefab = virusPrefab;
}
void Update() {
if (!hasPlacedUI && raycastManager.Raycast(new Vector2(Screen.width / 2,
Screen.height / 2), hits, TrackableType.PlaneWithinPolygon))
{ hasPlacedUI = true;
scanUI.SetActive(false);
```

```
placeObjectUI.SetActive(true);
}
if (Input.touchCount == 0 || selectedPrefab == null)
return;
Touch touch = Input.GetTouch(0);
if (touch.phase != TouchPhase.Began) return;
if (raycastManager.Raycast(touch.position, hits,
TrackableType.PlaneWithinPolygon))
{
Pose hitPose = hits[0].pose;
Instantiate(selectedPrefab, hitPose.position, hitPose.rotation); } } }
```

Output:





ANIMATED PROMPT