LAB 5: Create an AR application for Marker less application

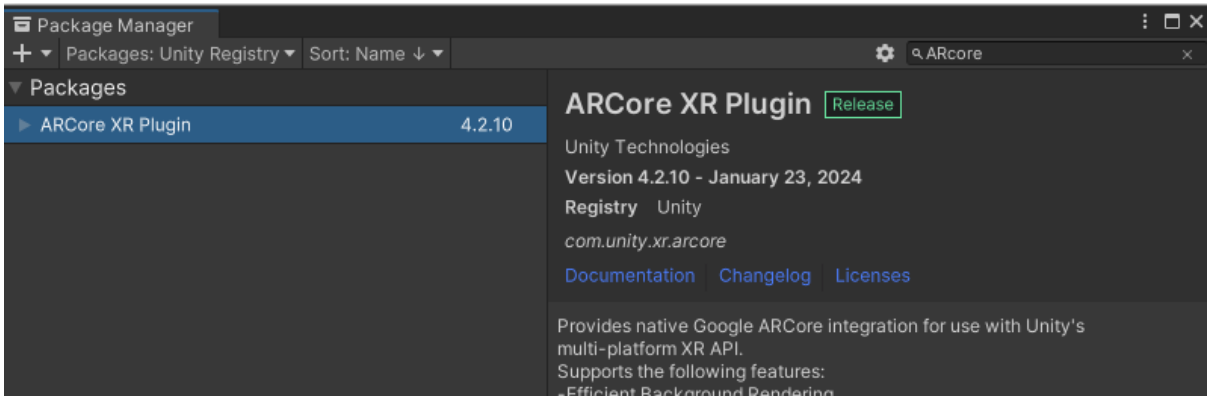
Sol:

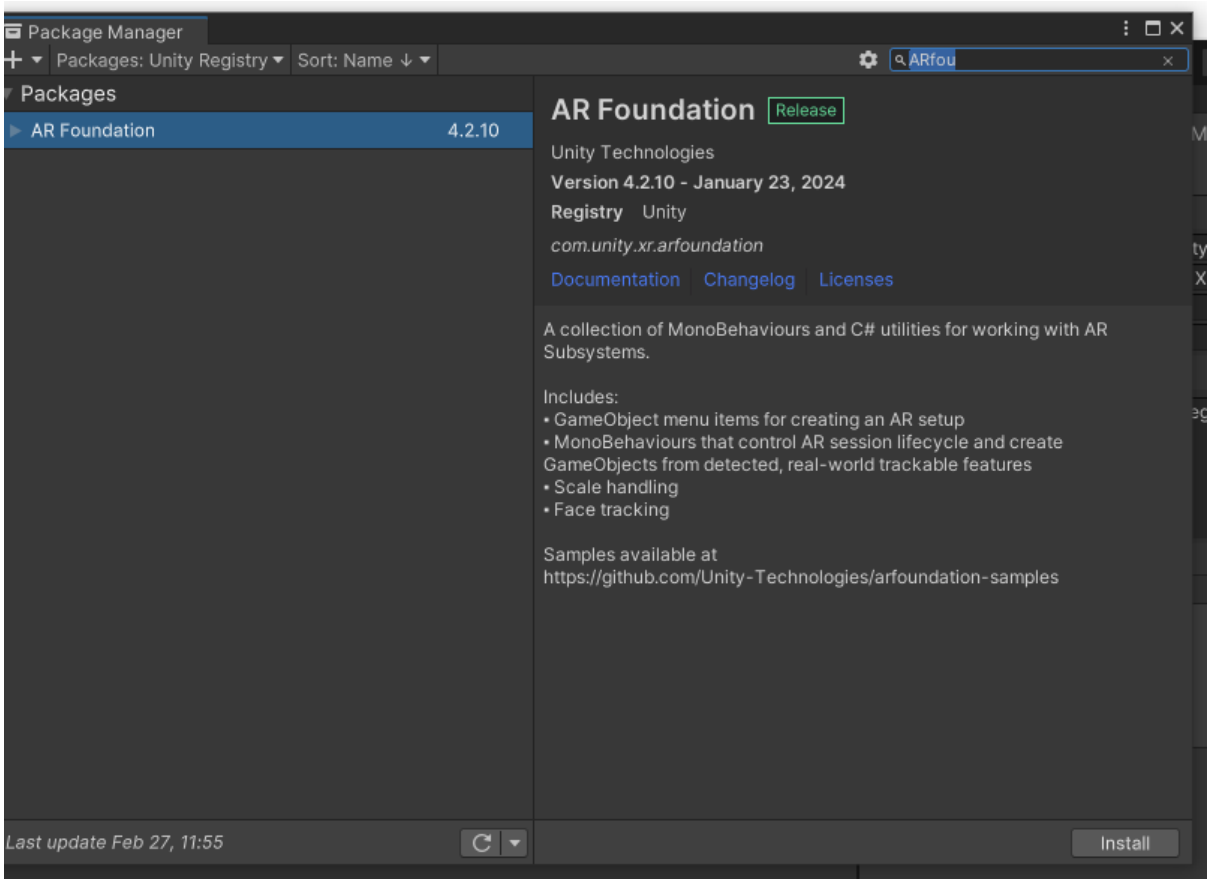
Step 1: Create new scene LAB5

Step2: Delete Main Camera

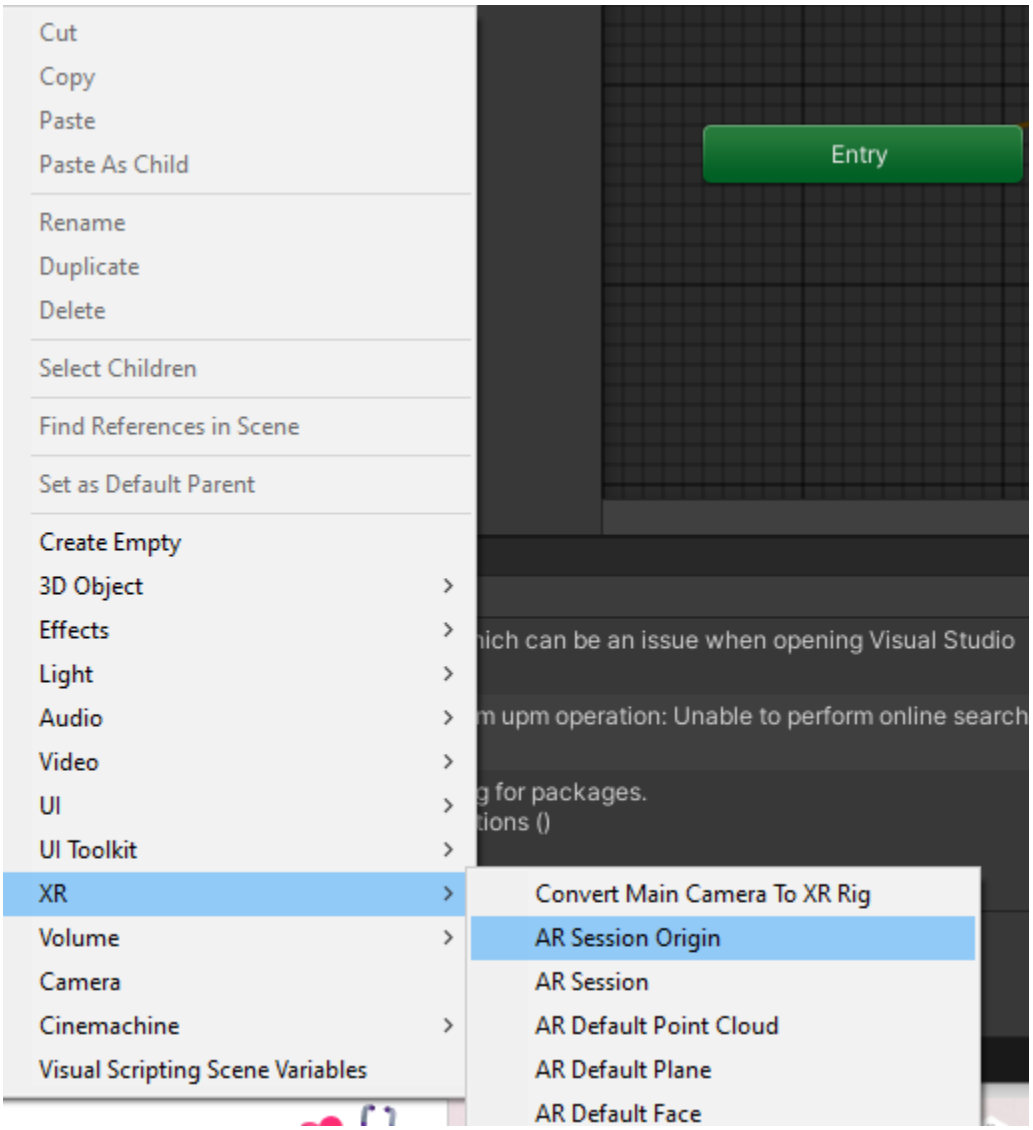
Step3: Add package ARcore and AR Foundation

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

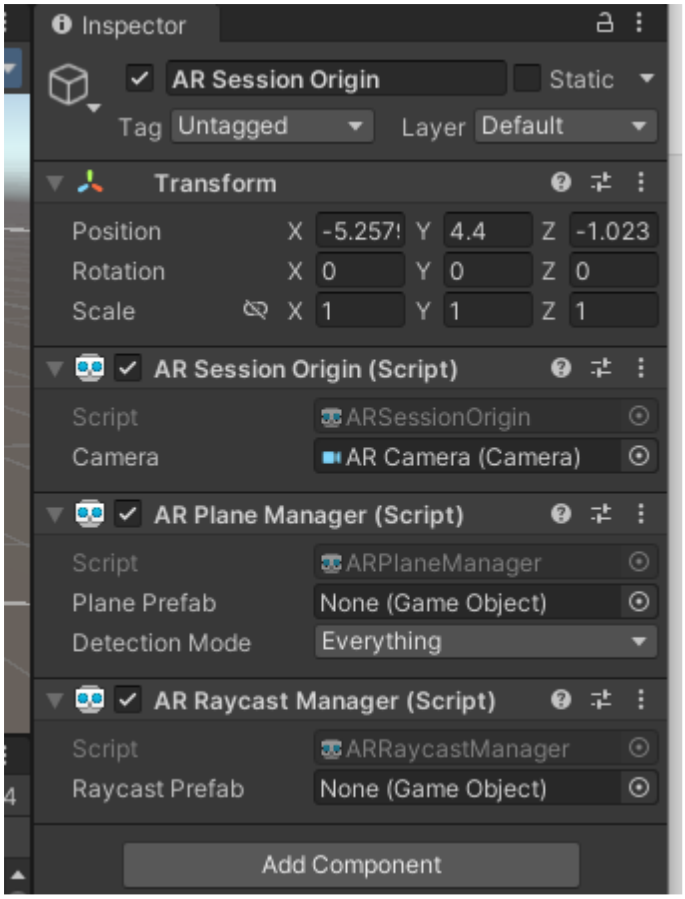




Step 4: Right Click- on hierarchy window> Add AR session origin or XR Session Origin

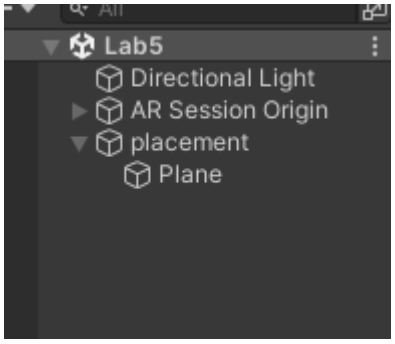


Step5: Select AR Session origin or XR origin and add component AR Raycast, AR plane Manager



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

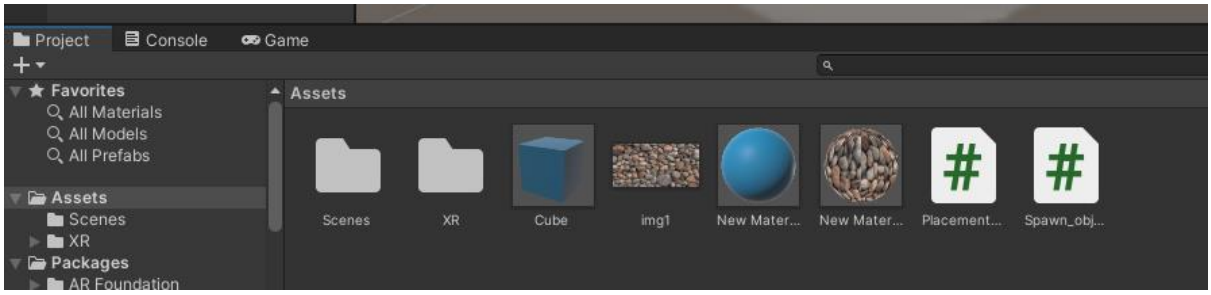
- create script spawn_object attach to SpawnManager

```
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)
```

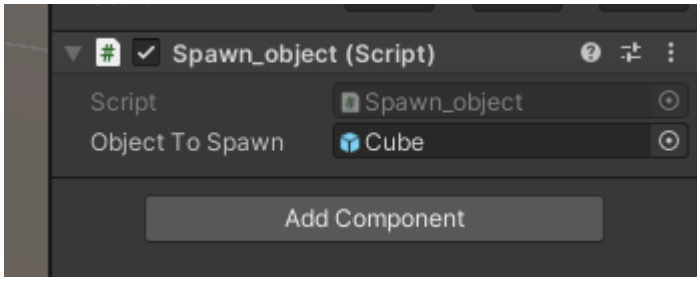
```
    {
        if (touch.phase == TouchPhase.Moved)
        {
            float rotationSpeed = 0.2f;
            spawnedObject.transform.Rotate(Vector3.up, -touch.deltaPosition.x * rotationSpeed);
        }
    }
}
```

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: