Lab 9: Experiment using AR

1. Go to Window > Package Manager.

2. At the top of the Package Manager window, look for the Packages dropdown menu.

3. Select Unity Registry from this dropdown. This view shows all official Unity packages, including AR Foundation.

4. With Unity Registry selected, use the search bar to type “AR Foundation”.

5. You should see AR Foundation in the results. Select it and click Install.

6. Install Platform-Specific XR Plugin. (ARCore/ARKit: SDKs provided by Google (for Android) and Apple (for iOS) that enable AR experiences.)

7. Go to File > Build Settings.

8. Choose either iOS or Android as your platform and click Switch Platform.

9. In Player Settings (located in the Build Settings window):

- Go to Other Settings and set the Minimum API Level to a version that supports AR (e.g., API Level 24 or higher for Android).
- Enable ARCore or ARKit support under XR Plug-in Management (in Project Settings > XR Plug-in Management).

10. Delete the default Main Camera.

11. Right-click in the Hierarchy and select XR > AR Session Origin. This will automatically add:

- AR Session Origin: Handles the positioning and scaling of AR content.
- AR Camera: Acts as the main camera for AR viewing.

12. Add an AR Session to the scene. Right-click in the Hierarchy and select XR > AR Session. This object manages the lifecycle of the AR experience.

13. In the AR Session Origin object, add an AR Plane Manager component (to detect planes) and an AR Raycast Manager (for user taps).

14. In the AR Plane Manager component, assign a Plane Prefab. This prefab visually represents detected planes.

- To create one, make a new 3D Plane (GameObject > 3D Object > Plane), customize its appearance (like a transparent material), and save it as a prefab.

15. Create or import a 3D object you'd like to place (e.g., a simple Cube).

16. Save this object as a Prefab by dragging it from the Hierarchy to the Assets folder.

17. Create a new script (e.g., ARObjectPlacer.cs) to handle object placement.

18. Attach the ARObjectPlacer script to the AR Session Origin.

19. Write a simple script to place the object when a plane is tapped. Here's a basic script to get started:

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

public class ARObjectPlacer : MonoBehaviour
{
    public GameObject objectToPlace;
    private ARRaycastManager raycastManager;
    private List<ARRaycastHit> hits = new List<ARRaycastHit>();
    void Start()
    {
        raycastManager = GetComponent<ARRaycastManager>();
    }
    void Update()
    {
        if (Input.touchCount > 0)
        {
            Touch touch = Input.GetTouch(0);
            if (touch.phase == TouchPhase.Began)
            {
                if (raycastManager.Raycast(touch.position, hits,
            TrackableType.Planes))
                {
                    Pose hitPose = hits[0].pose;

                    Instantiate(objectToPlace, hitPose.position,
                hitPose.rotation);
                }
            }
        }
    }
}
```

20. Drag the object prefab (e.g., Cube) into the Object To Place field in the Inspector.

21. Connect your device to your computer via USB.

22. Set up the device for Developer Mode (Android) or Xcode Development (iOS).

23. Go to File > Build Settings > Build and Run.

24. Unity will build the project, install it on the device, and launch it.

25. Point your device at a flat surface, and when it detects a plane, tap the screen to place the object.

Lab 10 – Create a physics based game play to realize all basic Newtonian effects

Create a physics based unity game play to realize all basic Newtonian effects.

- Keep It Simple: Focus on a few key Newtonian effects and implement them well.
- Iterate: Test and adjust forces, gravity, and object interactions for realistic behavior.
- Physics Material: Adjust the bounciness, friction, and other material properties to simulate different real-world surfaces.

Steps for a demo of ball sliding game

1. Start a new 3D project.

2. Unity starts with a default camera and light. You'll be adding objects like the ball and a sloped platform for the ball to slide on.

3. Create the Ground/Platform:

- Right-click in the Hierarchy and choose 3D Object > Plane. This will serve as the ground.
- Scale and position it as needed (use the Scale Tool or set values like 10, 1, 10 to make it larger).
- Optionally, create a ramp for the ball to slide down by using a cube (3D Object > Cube), scale it along the X or Z axis, and rotate it slightly to create an incline.

4. Create the Ball:

- Right-click in the Hierarchy and choose 3D Object > Sphere. This will beyour ball.
- b. Position the sphere above the ramp/plane so it can slide down.
- c. Scale it to a desired size, like 0.5 for both X, Y, and Z axes.
- d. Rename the sphere object to "Ball" for clarity.

Add Physics to the Ball

5. Add a Rigidbody Component:

- Select the ball (sphere) and click Component > Physics > Rigidbody.
- b. The Rigidbody will allow the ball to interact with the physics engine, including gravity, forces, and collisions.
- c. Set the mass of the ball to a reasonable value, such as 1, which will simulate realistic movement.

6. Add Collider Components:

- The ball already has a SphereCollider. The ground should have a

  BoxCollider (since a Plane comes with a BoxCollider by default).

  Apply Newtonian Mechanics

Newton's First Law (Inertia):

- The Rigidbody component automatically applies inertia: the ball will stay at rest or continue sliding unless a force (like friction or collision) acts upon it.

Newton's Second Law (Force = Mass × Acceleration):

- To apply force directly to the ball, you can use Unity's AddForce method to simulate acceleration. However, for sliding, the force will mainly come from gravity and friction.
- Gravity will naturally pull the ball down the inclined plane, causing it to accelerate. By adjusting the slope, you control how fast it moves.

Newton's Third Law (Action-Reaction):

- When the ball hits the ground or other objects, it will react with opposite forces, causing it to either bounce or stop, depending on friction and material settings.

Adjust Physics Settings

7. Global Gravity: Unity’s default gravity is set to Earth's gravity (-9.81 m/s2 on the Y-axis). You can adjust gravity in the Project Settings:

- Go to Edit > Project Settings > Physics and change the gravity settings if needed.

8. Friction: To control sliding, adjust the friction on the plane and ball:

- Create a Physic Material by right-clicking in the Project window and selecting Create > Physic Material.
- Set the Dynamic Friction and Static Friction properties. Lower values make the ball slide more easily.
- Assign this physic material to the plane's and ball's colliders to control how easily the ball slides.

9. Implementing Sliding Mechanics

- Position the Ball: Place the ball at the top of the inclined platform so it will slide down under the influence of gravity.
- Test: Press Play to test if the ball slides down the ramp. Gravity should naturally pull the ball down the slope due to the Rigidbody component.
- Adjust Friction and Mass: Adjust the friction values to control how fast or slow the ball slides. You can tweak the mass of the ball if needed to simulate different sliding behaviors.

10. User Input (Optional)

- If you want to let the user control the ball, you can add some basic user input:

o Attach a script to the ball for keyboard controls:

```
Using UnityEngine;
public class BallControl : MonoBehaviour
{
	public float force = 10f;
	private Rigidbody rb;
	void Start()
	{
		rb = GetComponent<Rigidbody>();
	}
	void Update()
	{
		// Use arrow keys to apply force in different directions
		if (Input.GetKey(KeyCode.UpArrow))
		{
			rb.AddForce(Vector3.forward * force);
		}
		if (Input.GetKey(KeyCode.DownArrow))
		{
			rb.AddForce(Vector3.back * force);
		}
		if (Input.GetKey(KeyCode.LeftArrow))
		{
			rb.AddForce(Vector3.left * force);
		}
		if (Input.GetKey(KeyCode.RightArrow))
		{
			rb.AddForce(Vector3.right * force);
		}
	}
}
```

- This script allows the player to control the ball's movement by applying force in the forward, backward, left, and right directions using the arrow keys.

11. Collisions and Bouncing

- If the ball hits walls or objects, it will react according to Newton's third law.
- You can create walls by adding cubes (3D Object > Cube) around the plane and assigning a BoxCollider to them.
- To make the ball bounce more upon collision, adjust the Bounciness property of the Physic Material applied to the ball's collider.

```
if (Input.GetKey(KeyCode.RightArrow))
{
        rb.AddForce(Vector3.right * force);
}
```

- This script allows the player to control the ball's movement by applying force in the forward, backward, left, and right directions using the arrow keys.