**Lab 11. UI Component, Scrolling, and Simple Events**

Steps to Create Unity Project with UI Component, Scrolling, and Simple Events:

1. Create a New Unity Project

* Open Unity Hub and create a new project.
* Choose a template (3D or 2D, depending on your need).
* Name your project and set its location, then click "Create."

2. Add a Canvas and UI Components

* In the Unity editor, right-click in the Hierarchy panel.
* Select UI → Canvas. This will automatically create an EventSystem and a Canvas object.
* Right-click on the Canvas and add the UI elements you need:
  + Button: UI → Button
  + Text: UI → Text
  + Scroll View: UI → Scroll View

3. Configure the Scroll View

* Unity's Scroll View contains the following parts:
  + Viewport: This is where the content will be displayed.
  + Content: This will contain all the elements inside the scroll view.
  + Scrollbar (optional): Adds a vertical/horizontal scrollbar.
  + You can adjust the size of the content inside the Scroll View to simulate scrolling.
  + You can place multiple UI elements (like buttons, images, etc.) inside the content for scrolling.

4. Add a C# Script for Event Handling

* Create a C# script to handle events like button clicks or UI interactions.
*  Right-click in the Project window, select Create → C# Script, and name it UIManager.

Sample C# Script for UIManager

This script handles button clicks and scroll view interactions:

using UnityEngine;

using UnityEngine.UI; // For working with UI elements

using UnityEngine.SceneManagement;

public class UIManager : MonoBehaviour

{

// Reference to the Button and Scroll View

public Button myButton;

public ScrollRect scrollRect;

public Text infoText; // Reference to a Text object for displaying

messages

void Start()

{

// Assign a function to the button click event

myButton.onClick.AddListener(OnButtonClick);

// Add listener for scroll view events (optional)

scrollRect.onValueChanged.AddListener(OnScrollChanged);

}

// Function to handle button click

void OnButtonClick()

{

Debug.Log("Button clicked!");

infoText.text = "You clicked the button!";

}

// Function to handle scrolling events (optional)

void OnScrollChanged(Vector2 position)

{

Debug.Log("Scrolling... Position: " + position);

}

// Optionally, you can load another scene when the button is clicked

public void LoadNewScene(string sceneName)

{

SceneManager.LoadScene(sceneName);

}

}

5. Linking the Script to UI Elements

* Attach the UIManager script to an empty GameObject in the Hierarchy panel.
* Drag and drop the UI elements (Button, ScrollRect, and Text) into the script's public fields in the Inspector.

6. Testing the Project

* Press the Play button in the Unity editor.
* Interact with the UI components:
  + Clicking the button will update the text and log a message.
  + Scrolling the Scroll View will log the current scroll position.

**Lab 12: AI as NPC with Navigation Mesh**

1. Open Unity Hub and create a new 3D Project. Give it an appropriate name (e.g., "NavMeshNPCExperiment").

2. In the Hierarchy, create a simple environment:

* Right-click in the Hierarchy > 3D Object > Plane (this will be the floor).
* Resize the plane as necessary (e.g., 20x20 units) by selecting it and adjusting thescale in the Inspector window.

3. Create Obstacles and Player Character. Add obstacles to your environment:

* Right-click in the Hierarchy > 3D Object > Cube.
* Position and scale the cubes around the plane to create walls or barriers.

4. Create a Player Character:

* Add a Capsule to represent the player: Right-click in the Hierarchy > 3D Object > Capsule.
* Rename the capsule to "Player".
* Add a basic camera and movement control for the player:

1. Right-click > Camera (to view from player's perspective).

2. You can write a simple script to control the player's movement (optional).

5. Creating the NPC. Add an NPC Character:

* Right-click in the Hierarchy > 3D Object > Capsule (this will act as the NPC).
* Rename the capsule to "NPC".

6. Add a NavMeshAgent:

* With the NPC selected, click Add Component in the Inspector > Search for NavMesh Agent and add it.
* This component allows the NPC to move along a NavMesh.

7. Enable Navigation Mesh Settings:Go to Window > AI > Navigation to open the Navigation window.

8. Mark Walkable Surfaces:

* Select the Plane (floor) and go to the Navigation window.
* Under the Object tab in the Navigation window, set the Plane to Navigation Static and ensure it's marked as "Walkable".

9. Bake the NavMesh:

* Go to the Bake tab in the Navigation window.
* Click Bake to generate a navigation mesh. The walkable area should be highlighted in blue.
* Ensure the obstacles are not part of the walkable area, but NPCs can navigate around them.

10. Write NPC Movement Script. Create a new C# script to control NPC movement.

* In the Project window, right-click > Create > C# Script. Name it NPCMovement.
* Double-click the script to open it in your code editor (e.g., Visual Studio).

11. Write the Script: Here’s a basic script to make the NPC follow the player using NavMeshAgent:

using UnityEngine;

using UnityEngine.AI;

public class NPCMovement : MonoBehaviour

{

public Transform player; // Reference to the player's position

private NavMeshAgent agent; // Reference to the NPC's NavMeshAgent

void Start()

{

// Get the NavMeshAgent component

agent = GetComponent<NavMeshAgent>();

}

void Update()

{

// Set the NPC's destination to the player's position

agent.SetDestination(player.position);

}

}

12. Assign the Player to the Script:

* Save the script and return to Unity.
* Drag and drop the NPCMovement script onto the NPC in the Hierarchy.
* In the Inspector, you will see the Player field in the script. Drag and drop the

Player object from the Hierarchy into this field.

13. Run the Scene:

* Press Play in Unity. The NPC should now follow the player character by navigating around obstacles using the NavMesh.

14. Refine the NPC's Behavior:

* The NPC will follow the player continuously as long as the game is running.
* You can further refine the NPC’s behavior by adding conditions like distance checks or patrol states, which we’ll add in the next steps.

15. Add AI Behavior (Patrol and Chase). Modify the NPCMovement Script to add patrol and chase behavior.

* Here is an extended script that lets the NPC patrol between points and chase the player when they get close:

public enum AIState { Patrol, Chase }

public class NPCMovement : MonoBehaviour

{

public AIState currentState = AIState.Patrol;

public Transform[] patrolPoints;

public Transform player;

private NavMeshAgent agent;

private int patrolIndex;

void Start()

{

agent = GetComponent<NavMeshAgent>();

patrolIndex = 0;

GotoNextPatrolPoint();

}

void Update()

{

if (currentState == AIState.Patrol)

{

Patrol();

}

else if (currentState == AIState.Chase)

{

ChasePlayer();

}

float distanceToPlayer = Vector3.Distance(transform.position,

player.position);

if (distanceToPlayer < 5.0f)

{

currentState = AIState.Chase;

}

else if (distanceToPlayer > 7.0f)

{

currentState = AIState.Patrol;

}

}

void Patrol()

{

if (!agent.pathPending && agent.remainingDistance < 0.5f)

{

GotoNextPatrolPoint();

}

}

void GotoNextPatrolPoint()

{

if (patrolPoints.Length == 0)

return;

agent.destination = patrolPoints[patrolIndex].position;

patrolIndex = (patrolIndex + 1) % patrolPoints.Length;

}

void ChasePlayer()

{

agent.SetDestination(player.position);

}

}

16. Set Patrol Points:

o In the Inspector, add empty GameObjects as patrol points:

1. Right-click in the Hierarchy > Create Empty.

2. Place the empty objects where you want the NPC to patrol.

3. Assign them in the NPCMovement script’s Patrol Points array in the

Inspector.

17. Run the Scene:

* Press Play.
* The NPC should now patrol between the set points when the player is far away.

When the player gets close, the NPC should chase them.

18. Optional Enhancements

1. Dynamic Obstacles: Make certain objects dynamic (e.g., moving doors), and update the

NavMesh in real-time.

2. Additional AI: Implement more advanced behaviors like fleeing from the player or

cooperating with other NPCs.