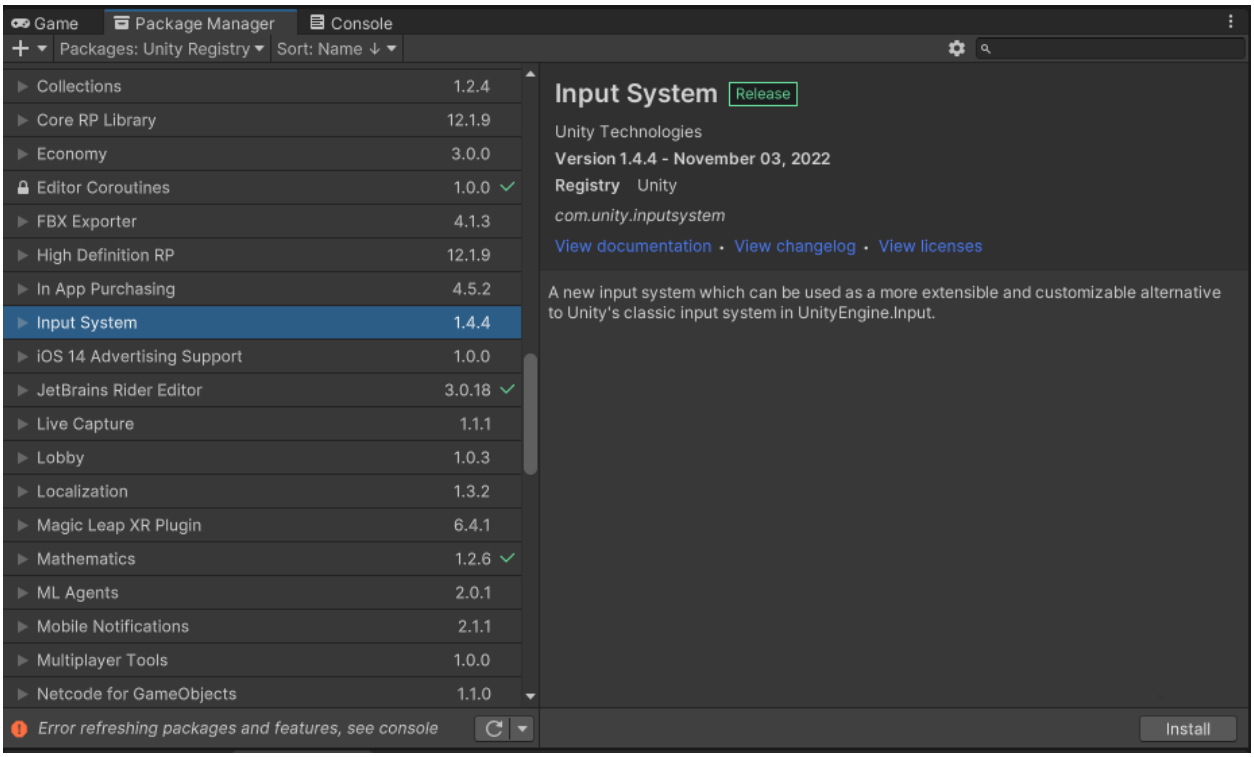
Using new Input System for testing

1. Download the new Input System from Package Manager

You can find the Unity Package Manager using the following: Window > Package Manager

Select “Packages: Unity Registry” on the top to list all the packages present in the Unity Registry.

Scroll down to find the Input System as follows:



Click on “Install”.

Note: This process will restart your project in order to enable the new input system with the backends.

2. Add the new test systems to manifest.json

In order to use the testing system in our test cases we have to add the entry of input system into the “testables” section. In order to do so navigate to the following files in your project:

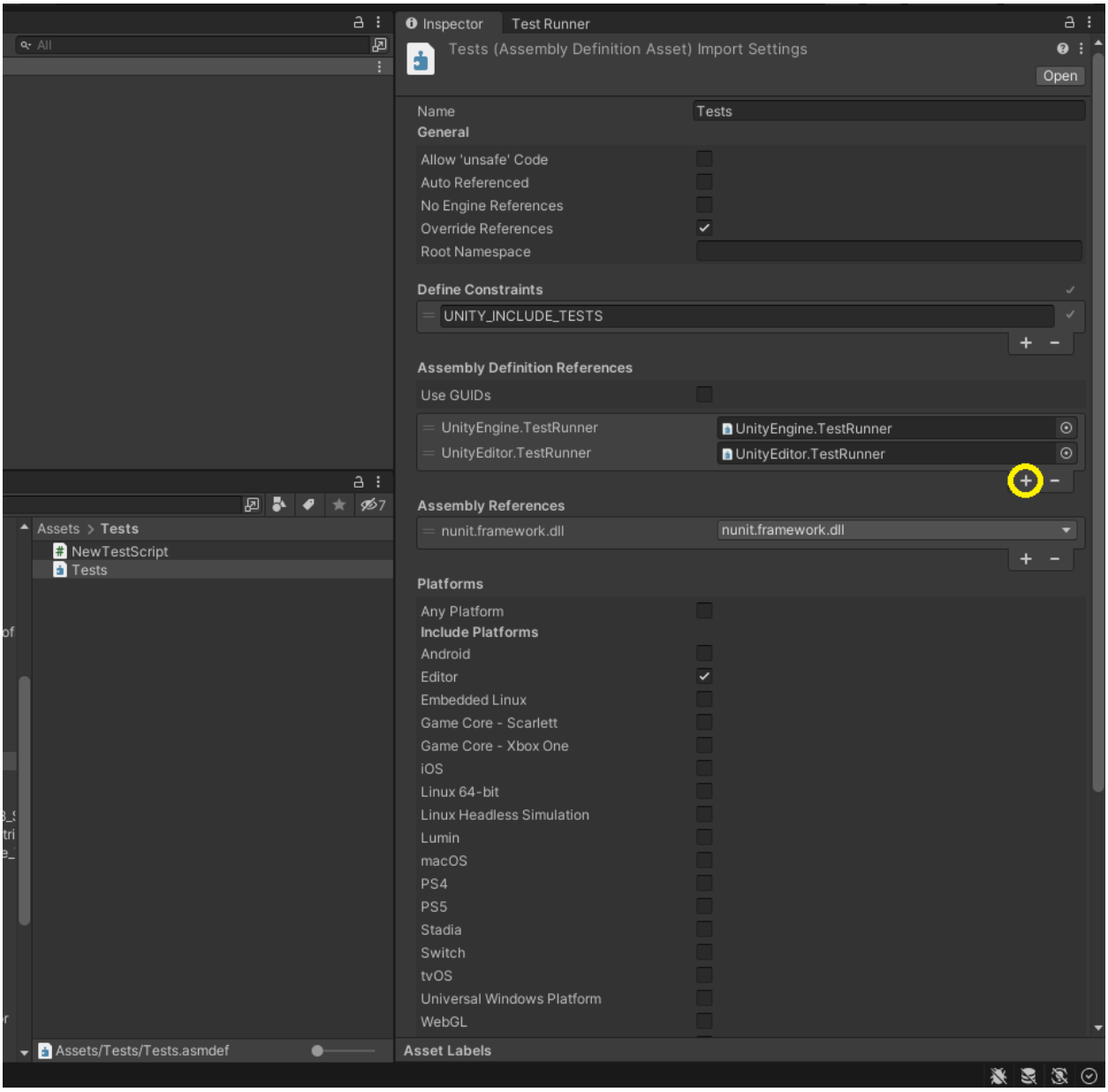
Packages > manifest.json

Add the following key after “dependencies”:


"com.unity.textmeshpro": "3.0.6",
"com.unity.timeline": "1.6.4",
"com.unity.toolchain.win-x86_64-linux-x86_64": "2.0.4
"com.unity.ugui": "1.0.0",
"com.unity.visualscripting": "1.8.0",
"com.unity.modules.ai": "1.0.0",
"com.unity.modules.androidjni": "1.0.0",
"com.unity.modules.animation": "1.0.0",
"com.unity.modules.assetbundle": "1.0.0",
"com.unity.modules.audio": "1.0.0",
"com.unity.modules.cloth": "1.0.0",
"com.unity.modules.director": "1.0.0",
"com.unity.modules.imageconversion": "1.0.0",
"com.unity.modules.imgui": "1.0.0",
"com.unity.modules.jsonserialize": "1.0.0",
"com.unity.modules.particlesystem": "1.0.0",
"com.unity.modules.physics": "1.0.0",
"com.unity.modules.physics2d": "1.0.0",
"com.unity.modules.screencapture": "1.0.0",
"com.unity.modules.terrain": "1.0.0",
"com.unity.modules.terrainphysics": "1.0.0",
"com.unity.modules.tilemap": "1.0.0",
"com.unity.modules.ui": "1.0.0",
"com.unity.modules.uielements": "1.0.0",
"com.unity.modules.umbra": "1.0.0",
"com.unity.modules.unityanalytics": "1.0.0",
"com.unity.modules.unitywebrequest": "1.0.0",
"com.unity.modules.unitywebrequestassetbundle": "1.0.
"com.unity.modules.unitywebrequestaudio": "1.0.0",
"com.unity.modules.unitywebrequesttexture": "1.0.0",
"com.unity.modules.unitywebrequestwww": "1.0.0",
"com.unity.modules.vehicles": "1.0.0",
"com.unity.modules.video": "1.0.0",
"com.unity.modules.vr": "1.0.0",
"com.unity.modules.wind": "1.0.0",
"com.unity.modules.xr": "1.0.0"
},
"testables": [
"com.unity.inputsystem"
]
}

3. Add the InputSystem assemblies to your testing assemblies.

In order to use the Input System in your test cases you have to add their assembly entries into your test case assembly entry or definition file. This is a “.asmdef” file that is present in your project and resides with your test script. Click on the “.asmdef” file and check the “Inspector”.



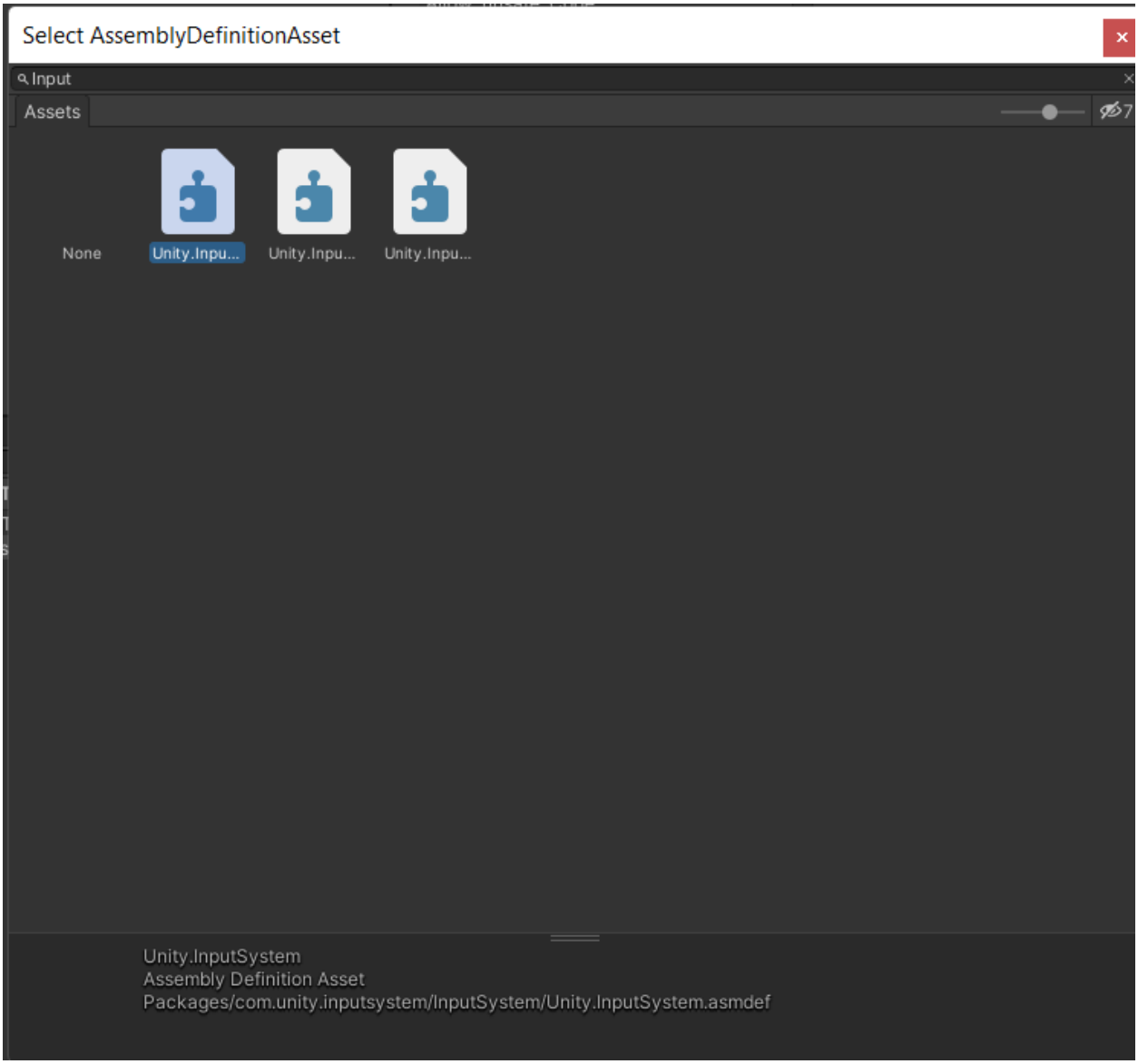
Search for the following keyword: “Input” and you will find 3 files namely:

a. Unity.InputSystem

b. Unity.InputSystem.IntegrationTests

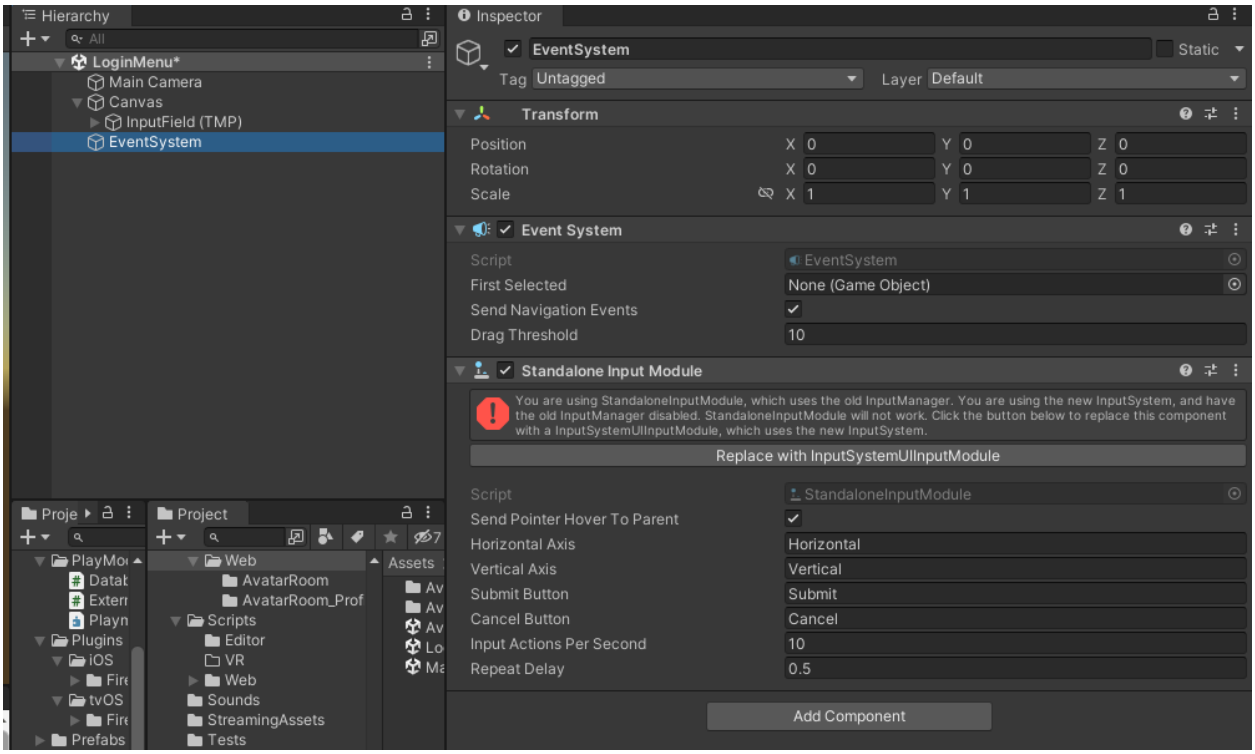
c. Unity.InputSystem.TestFramework

SELECT/ADD ONLY “Unity.InputSystem” AND “Unity.InputSystem.TestFramework” TO THE DEFINITIONS AND DONOT FORGET TO CLICK ON THE APPLY BUTTON IN THE INSPECTOR MENU.



4. Updating your in-game EventSystem

After updating the InputSystem and adding it to the test assembly file we have to update the in-game EventSystem to the new InputSystem. In-order to do so, Navigate to your scene and look for “EventSystem” and in the Inspector you will be able to see that the “Standalone Input Module” is being used which is the old input system. Click on “Replace with InputSystemUIInputModule” and this will replace it with the new Input System. Save your scene before writing test cases!!



4. Writing Test Case

Create a test script and write inherit the class with the InputTestFixture class.

A simple test case will be somewhat on these lines:

```
using System.Collections;
using NUnit.Framework;
using UnityEngine;
using UnityEngine.TestTools;
using UnityEngine.InputSystem;
using UnityEngine.SceneManagement;
using UnityEngine.UIElements;

[TestFixture]
0 references
public class InputTest : InputTestFixture
{


    0 references
    override public void Setup()
    {
        SceneManager.LoadScene(0);

    }
    // A UnityTest behaves like a coroutine in Play Mode. In Edit Mode you can use
    // `yield return null;` to skip a frame.
    [UnityTest]
    0 references
    public IEnumerator InputTestWithEnumeratorPasses()
    {
        yield return null;
        GameObject button = GameObject.Find("Canvas/Button");
        GameObject cube = GameObject.Find("TargetCube");
        var Mouse = InputSystem.AddDevice<Mouse>();
        Move(Mouse.position, button.transform.position);
        Click(Mouse.leftButton);
        yield return null;
        Debug.Log(cube.GetComponent<Renderer>().material.name);
        Assert.AreEqual(cube.name, "NewCube");
    }
}
```

This code above will check if the name of the cube has been changed to “NewCube” after clicking the button on screen.