

Tuesday 29th of Mar 2022

Unity WORKSHOP


Unity Engine Fundamentals

Prepared by the Artificial Intelligence Research Center (AIRC), Ajman
University, Ajman, UAE



Workshop Overview

The workshop aim

-  The workshop aims to build up your knowledge in game development, transform and deploy your ideas into high-quality games using Unity.

The workshop Duration

10 - 12 hours (5-6 Days)

Workshop Objectives

- Learn good coding of C# and controlling patterns.
- Gain a general knowledge of videogame design.
- Get experience using Unity (very versatile 3D tool).
- Develop a 3D rocket game.
- Build and Deploy the accomplished game.

Workshop Agenda

- Day 1
 - 🕒 **Workshop Overview**
 - 🕒 **Project Demo**
 - 🕒 **Introduction to Unity 3D Engine**
 - 🕒 **Basic Game Units**
 - 🕒 **Community and Support**
 - 🕒 **Introducing Prefabs**
 - 🕒 **Your First Script**
 - 🕒 **Basic Input Controlling**
- Day 2
 - 🕒 **Practicing Methods**
 - 🕒 **Physics Laws**
 - 🕒 **Rocket Vertical movement**
 - 🕒 **Rocket Rotation**
- Day 3
 - 🕒 **Rocket Movement and Rotation Constraints**
 - 🕒 **Unity Audio Introduction**

- Day 3
 - 🕒 **Play Audio Clip**
 - 🕒 **Objects Collision**
 - 🕒 **Switch Statement**
- Day 4
 - 🕒 **Swap between Scenes (Game Levels)**
 - 🕒 **Using Invoke**
 - 🕒 **Multiple Audio Clips**
 - 🕒 **Introduction to Unity Assets Store**
- Day 5
 - 🕒 **Make Rocket Look Spiffy**
 - 🕒 **Practicing with Particles**
 - 🕒 **Move Obstacles**
 - 🕒 **Quit Application**
- Day 6
 - 🕒 **Design The Game Levels**
 - 🕒 **Cinemachine and Camera**
 - 🕒 **Build and Publish the game**



Day 1 Outlines

- ④ **Workshop Overview**
- ④ **Project Demo**
- ④ **Introduction to Unity
3D Engine**
- ④ **Basic Game Units**
- ④ **Community and
Support**
- ④ **Introducing Prefabs**
- ④ **Your First Script**
- ④ **Basic Input Controlling**

1. Project Demo

What is the accomplished game after finishing the workshop



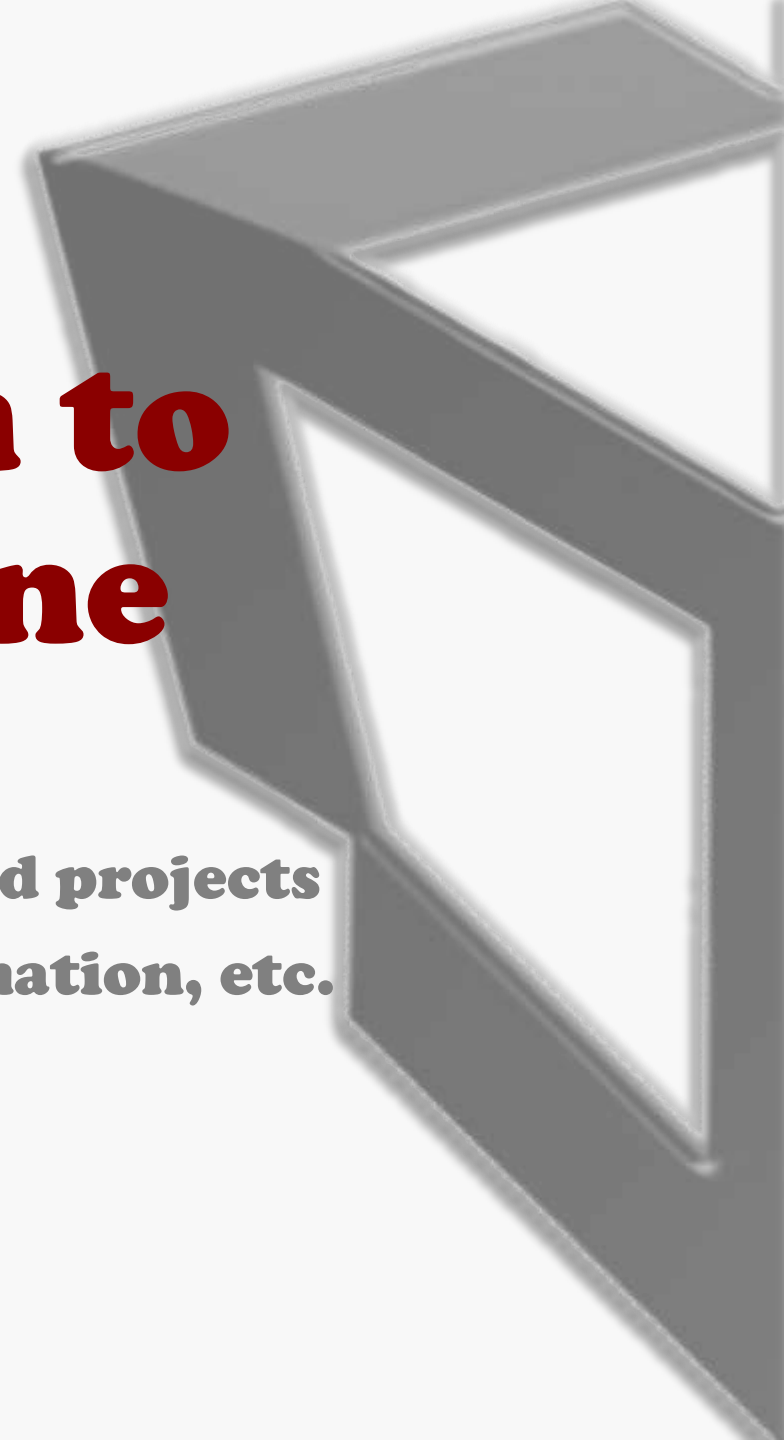
The Workshop Project

- ④ Build and Design a 3D rocket game using unity engine along with covering the basics of programming with C# language. In this game, the student will learn how to move and rotate the rocket, add sound effects, develop multiple game levels and exploit in case of obstacles hits.



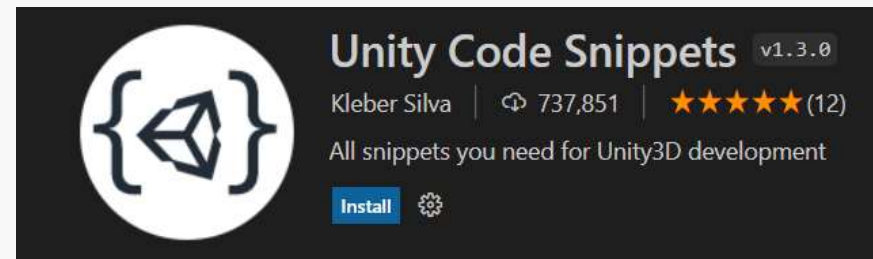
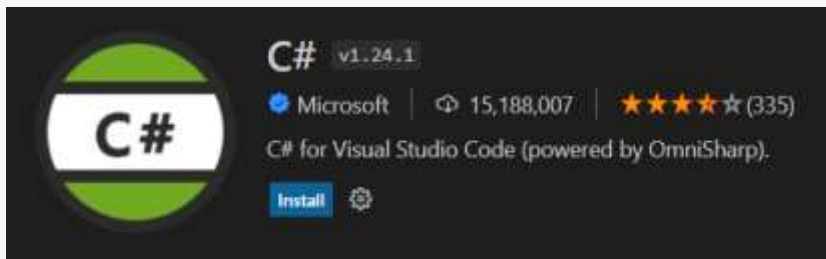
2. Introduction to Unity 3D Engine

**A 2D & 3D development platform to build projects
for various industries across games, animation, etc.**



Main steps

- 📦 Download Unity hub (<https://unity3d.com/get-unity/download>)
- 📦 Download Unity release
- 📦 Download Visual Studio code IDE (<https://code.visualstudio.com/download>)
 - Check the unity Editor to be Visual studio code (Edit → Preferences → External tools)



Game Engine and IDE

🕹️ Game Engine



- Visual interface for creating games
- Systems of existing code we can use (Physics, audio, etc)

🕹️ Integrated Development Environment (IDE)



Visual Studio Code

- Helps us to write code to tell the game engine what to do
- Auto-complete, color coding, Syntax error checking

Introduce the following

Unity hub

- Project Types
- Create project

Unity

- Project name
- Save Project
- Layout

• Hierarchy

- Object
- Camera
- Light

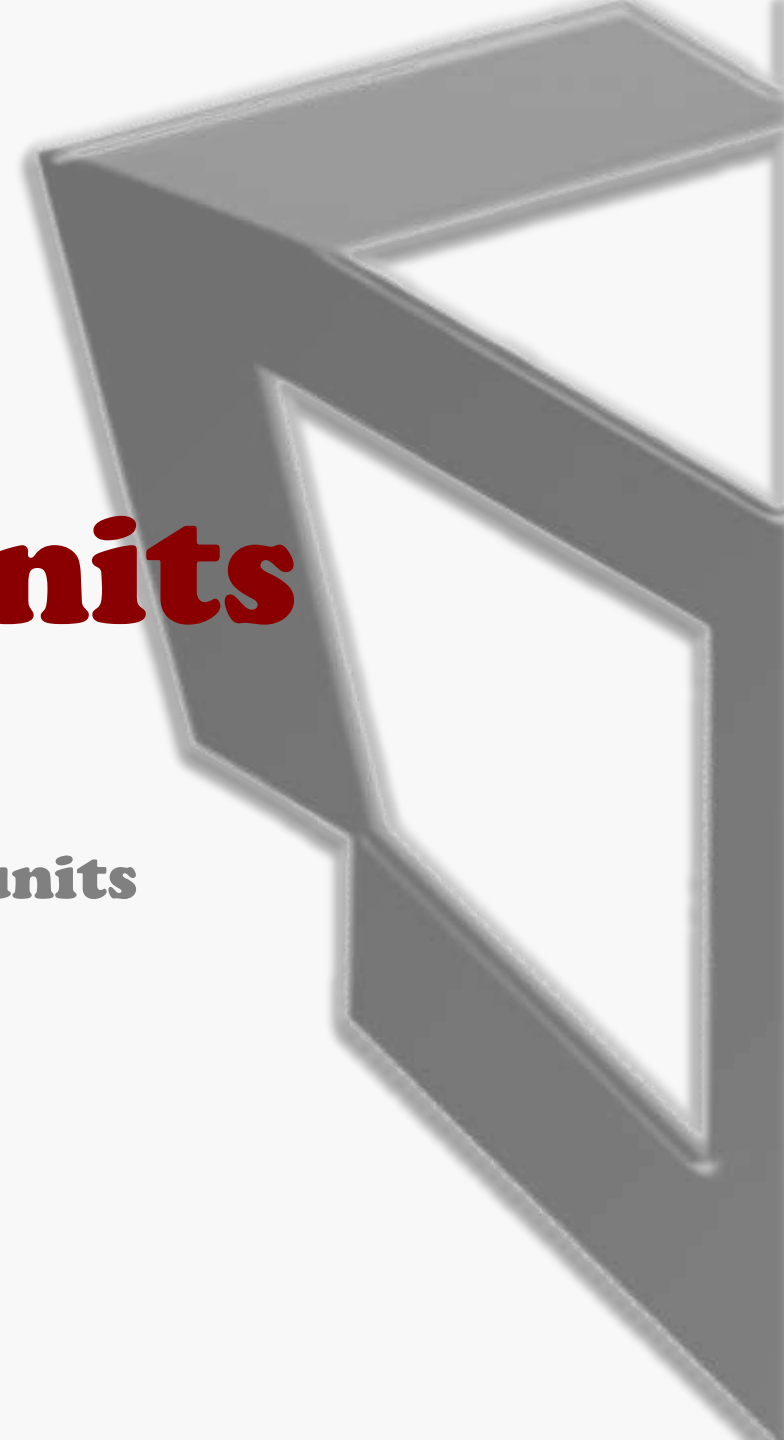
• Inspector

- Object components
- Transform

• Project Assets

3. Basic Game Units

Start Practicing unity basic game units



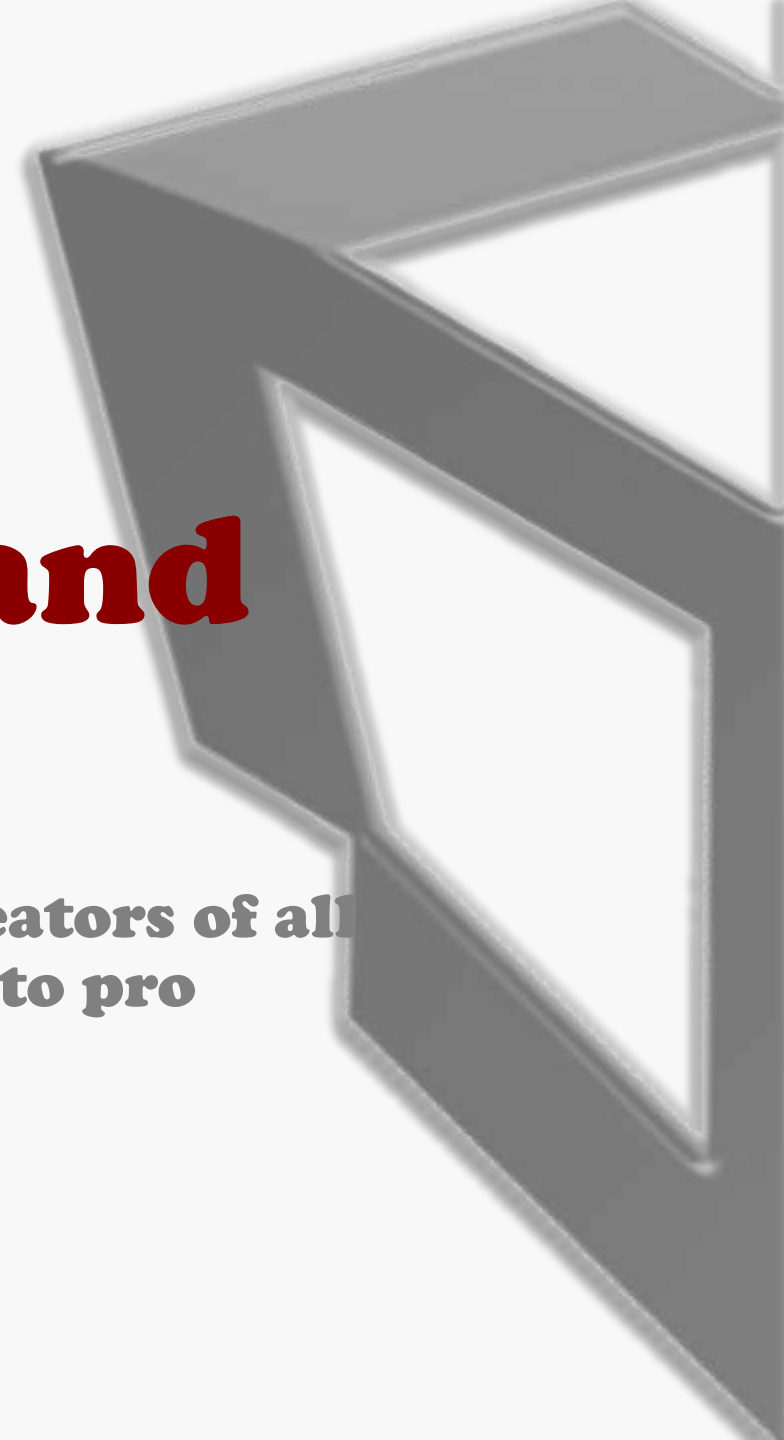
Navigation and Basic Actions

- 🕒 Create a cube shaped object
- 🕒 Create Nested/compound objects
- 🕒 Look around (Left and right)
- 🕒 Change view angle
- 🕒 Create a quick scene
- 🕒 Zoom in/out
- 🕒 Focus on object

4. Community and Support

the largest community of Unity users. Creators of all types – beginner to expert, hobbyist to pro

<https://unity.com/community>



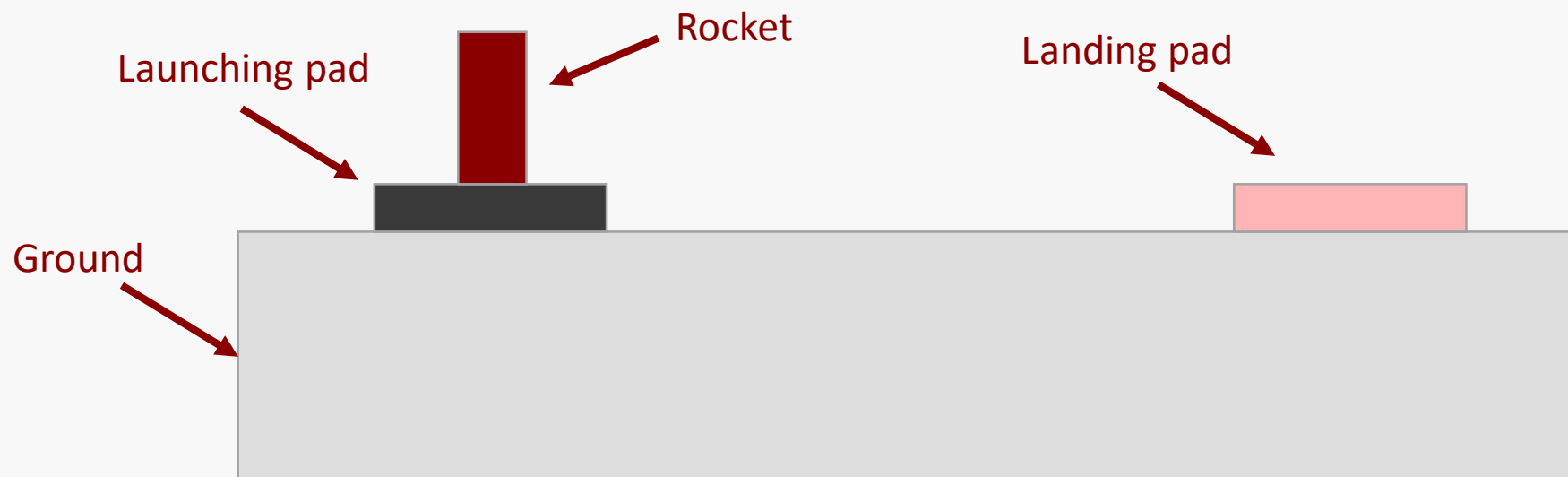
5. Introducing Prefabs

Rocket Game Prefabs



Building our starting pieces

- 🎯 Create a prefab from your Rocket
- 🎯 Create the launch pad
- 🎯 Create a quick scene using your prefabs (Rocket – launching pad – landing pad)



6. Your First Script

What is a script

Control scenes, objects, actions using C# Scripts



Unity Scripting

🕹️ Integrated Development Environment (IDE)



Visual Studio Code

- Helps us to write code to tell the game engine what to do
- Auto-complete, color coding, Syntax error checking

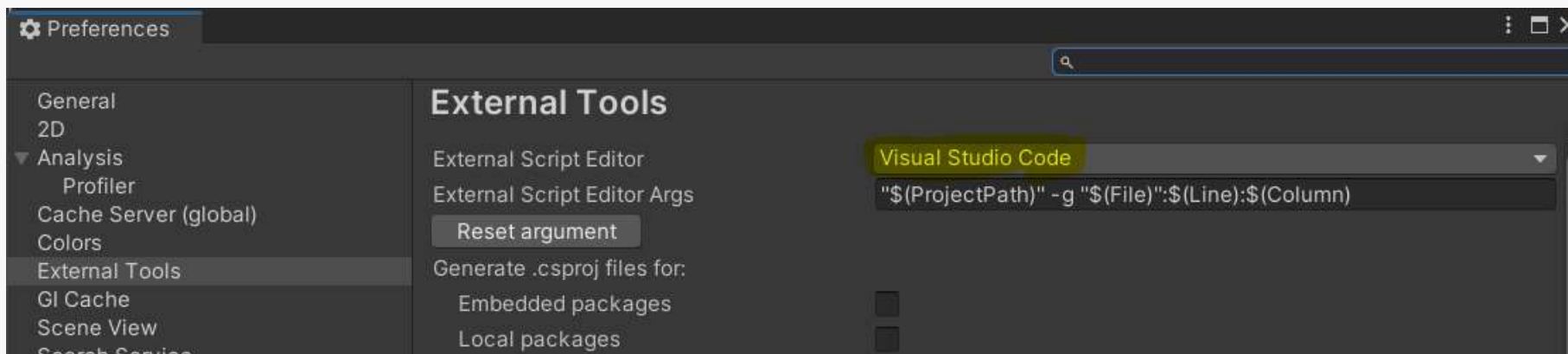
🕹️ Scripting Language



- is a programming language developed by Microsoft, capable of a wide range of tasks,
- it's the programming language used by Unity Game Engine

Before Scripting

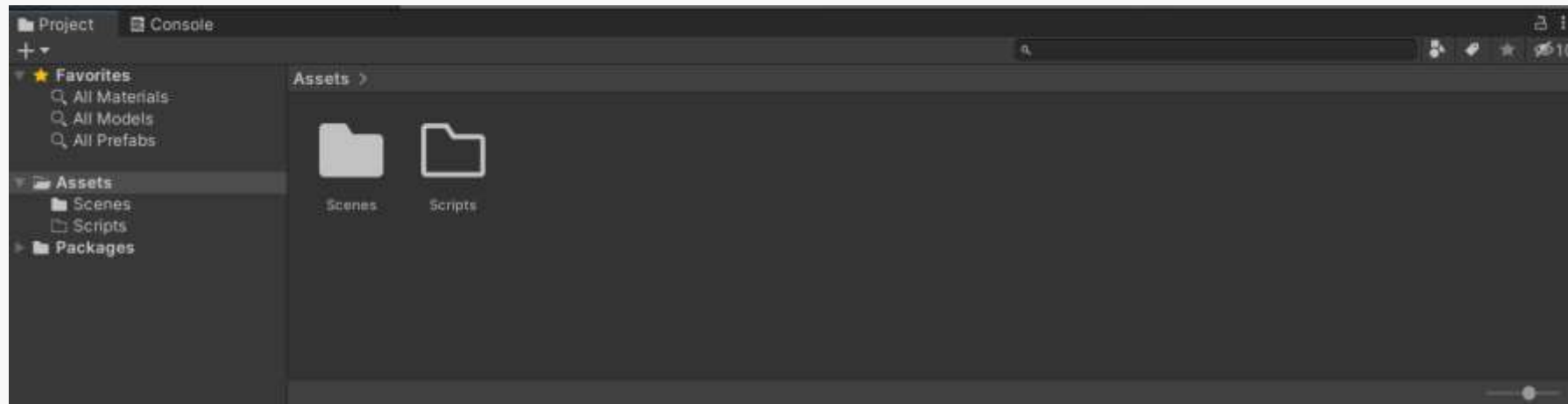
- 🕒 Make sure you have installed C# extension on visual studio code
- 🕒 Make sure you have installed Unity Code Snippet on visual studio code
- 🕒 Set Visual Studio Code to default script editor
 - Edit > Preferences > External Tools



Before Scripting

📁 Create folder for Scripts

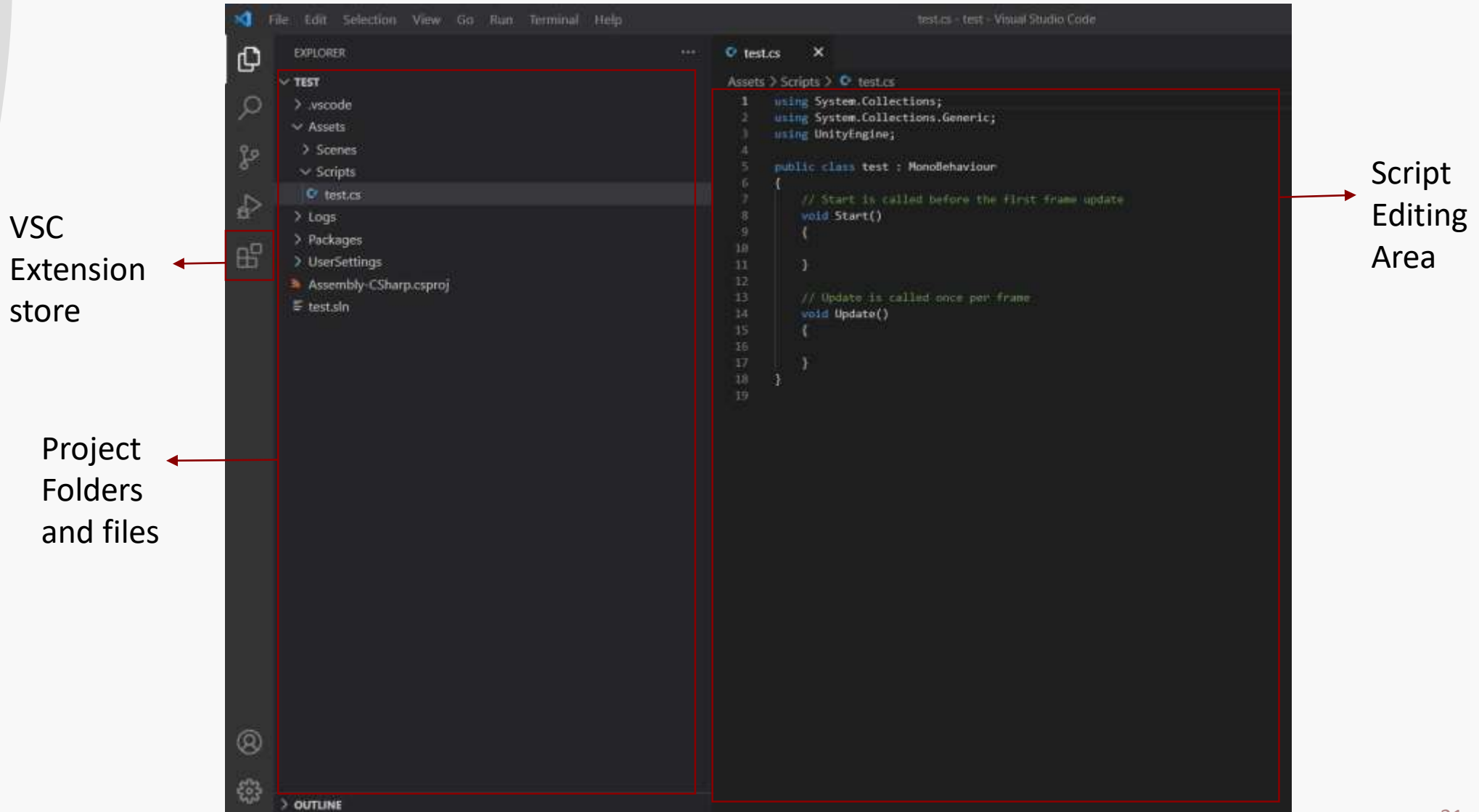
- Right-click
- Choose create > Folder
- Name it "Scripts"



Creating A Script

- 👉 Navigate to the Assets > Scripts folder
- 👉 Right click on the mouse > create > C# Script
- 👉 Rename your script Immediately
 - Why ?
 - How to solve the problem?
 - Delete the Script file and create a new one with the name desired
 - Change the class name to match the Script file name
- 👉 Open the script in VS Code by double clicking on the script file icon
 - The C# script file extension is .cs

Visual Studio Code UI



Anatomy of a Script file

The image shows a code editor window for a file named 'test.cs'. The code is as follows:

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class test : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start()
9     {
10
11     }
12
13     // Update is called once per frame
14     void Update()
15     {
16
17     }
18 }
19
```

Annotations on the right side of the image point to specific parts of the code:

- Packages:** Points to the three 'using' statements (lines 1-3).
- Class Scope:** Points to the entire class definition from line 5 to line 18.
- Start() Scope:** Points to the 'Start()' method definition from line 8 to line 11.
- Update() Scope:** Points to the 'Update()' method definition from line 14 to line 17.

Start and Update Functions

⌚ Start()

Run only once the Scene is loaded (first frame only)

⌚ Update()

Runs once for each frame

<https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html>

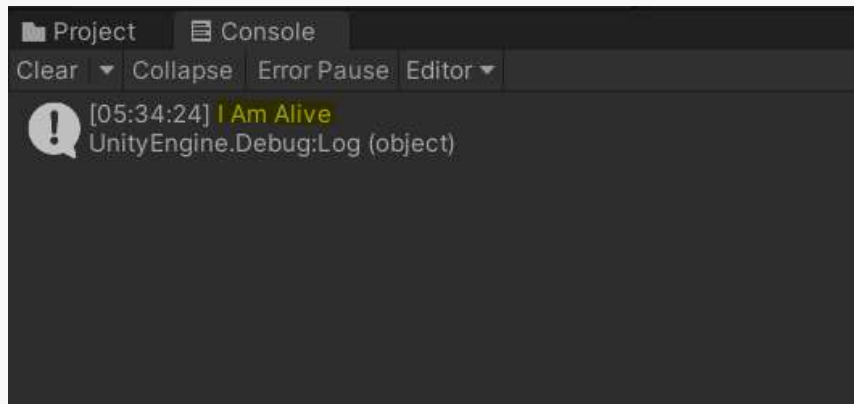
Debug Statement

🔗 Syntax:

- `Debug.Log("Messag you want to display")`

🔗 Used for debugging

🔗 The debug message is displayed in the console area



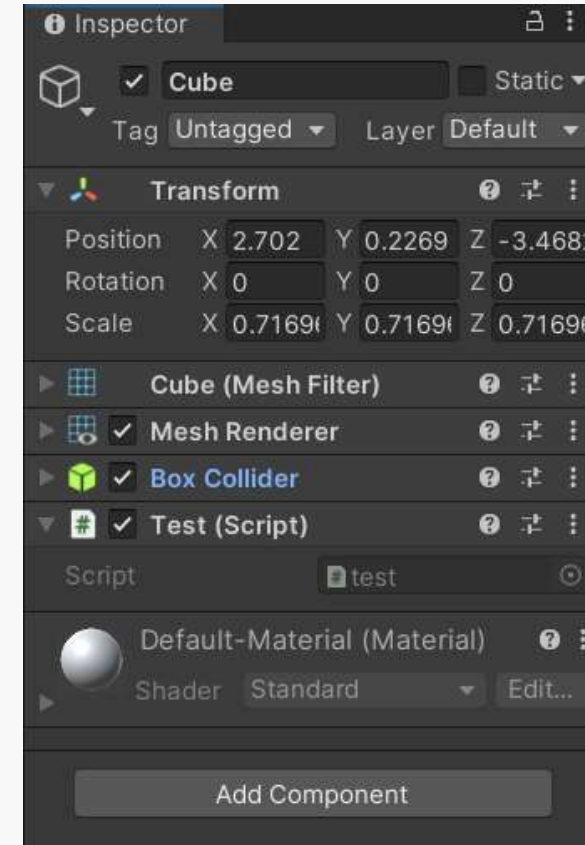
<https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html>

Link/Map a script

🕒 Attach the Script to the object instance

(not the prefab)we desire to control

- Drag and Drop to the inspector of the object
- Add component > add > scripts > scriptName



7. Basic Input Controlling

Get the keyboard, mouse, arrows keys input.



Get Input

- ⌚ When we push “space”, then print “Thrusting”
- ⌚ When we push “A”, then print “rotate left”
- ⌚ else if we push “D”, then print “rotate right”

Get Input – If statement

- ⌚ When we push “space”, then print “Thrusting”
- ⌚ When we push “A”, then print “rotate left”
- ⌚ else if we push “D”, then print “rotate right”

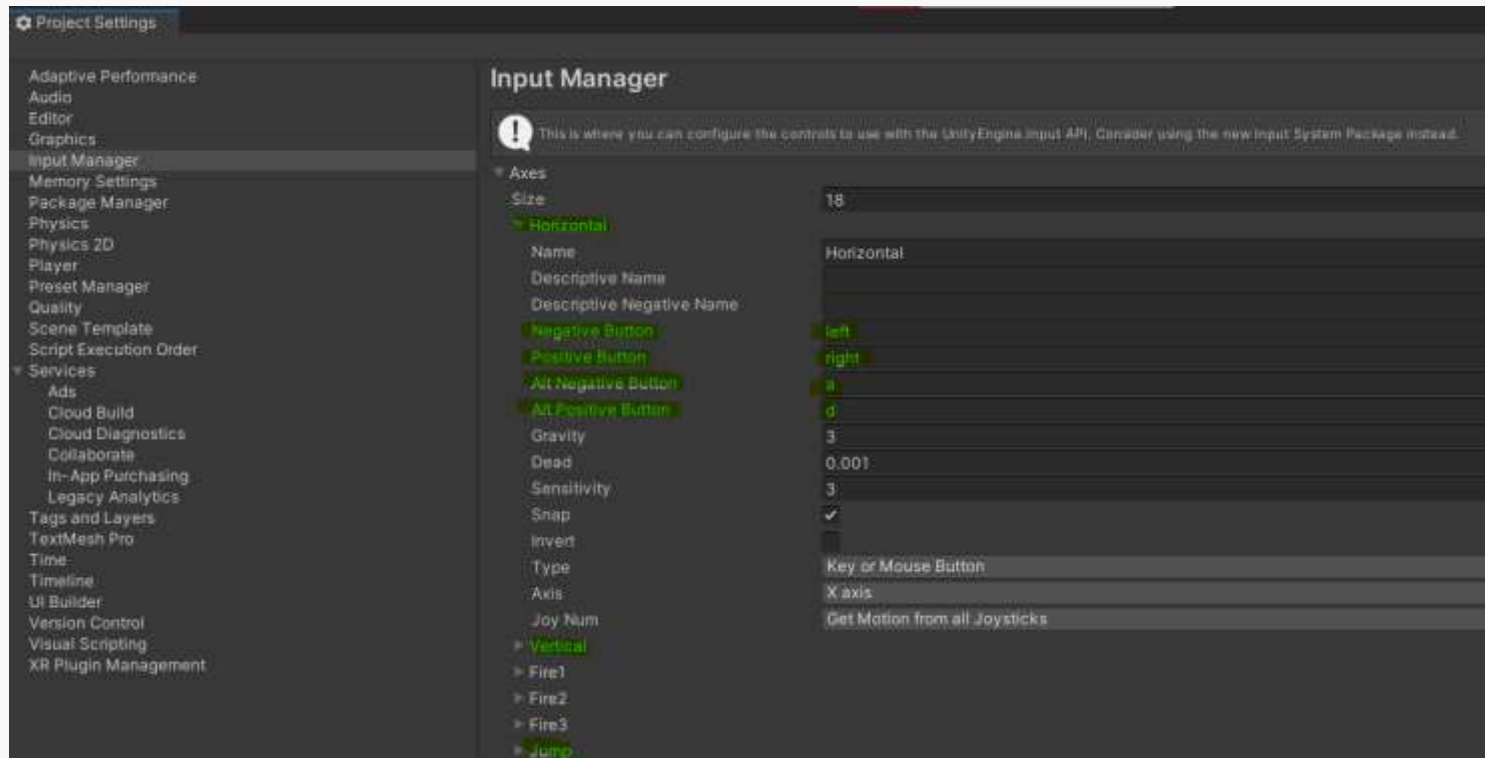
```
if (we push “space”)  
{  
    Debug.Log(“Thrusting”)  
}
```

```
if (we push “A”)  
{  
    Debug.Log(“rotate left”)  
}
```

```
else if (we push “D”)  
{  
    Debug.Log(“rotate right”)  
}
```

Input Manager

🔗 Edit → Project Settings → Input Manager



Get Input

🔗 Basic Syntax

```
Input.GetKey(KeyCode.YourKey)
```

```
Input.GetKey("YourKey")
```

Get Input

- ⌚ When we push “space”, then print “Thrusting”
- ⌚ When we push “A”, then print “rotate left”
- ⌚ else if we push “D”, then print “rotate right”

```
If(Input.GetKey(KeyCode.Space))  
{  
    Debug.Log("Thrusting");  
}
```

```
if (Input.GetKey(KeyCode.A))  
{  
    Debug.Log("rotate left");  
}
```

```
else if (Input.GetKey(KeyCode.D))  
{  
    Debug.Log("rotate right");  
}
```

The final piece of code

```
void Update()
{
    if (Input.GetKey(KeyCode.Space))
    {
        Debug.Log("Thrusting");
    }

    if (Input.GetKey(KeyCode.A))
    {
        Debug.Log("Rotate left");
    }

    else if (Input.GetKey(KeyCode.D))
    {
        Debug.Log("Rotate Right");
    }
}
```