



**Unity**를 이용한 2D 게임프로그래밍

# Lecture 1

## Unity 사용해 보기

강영민

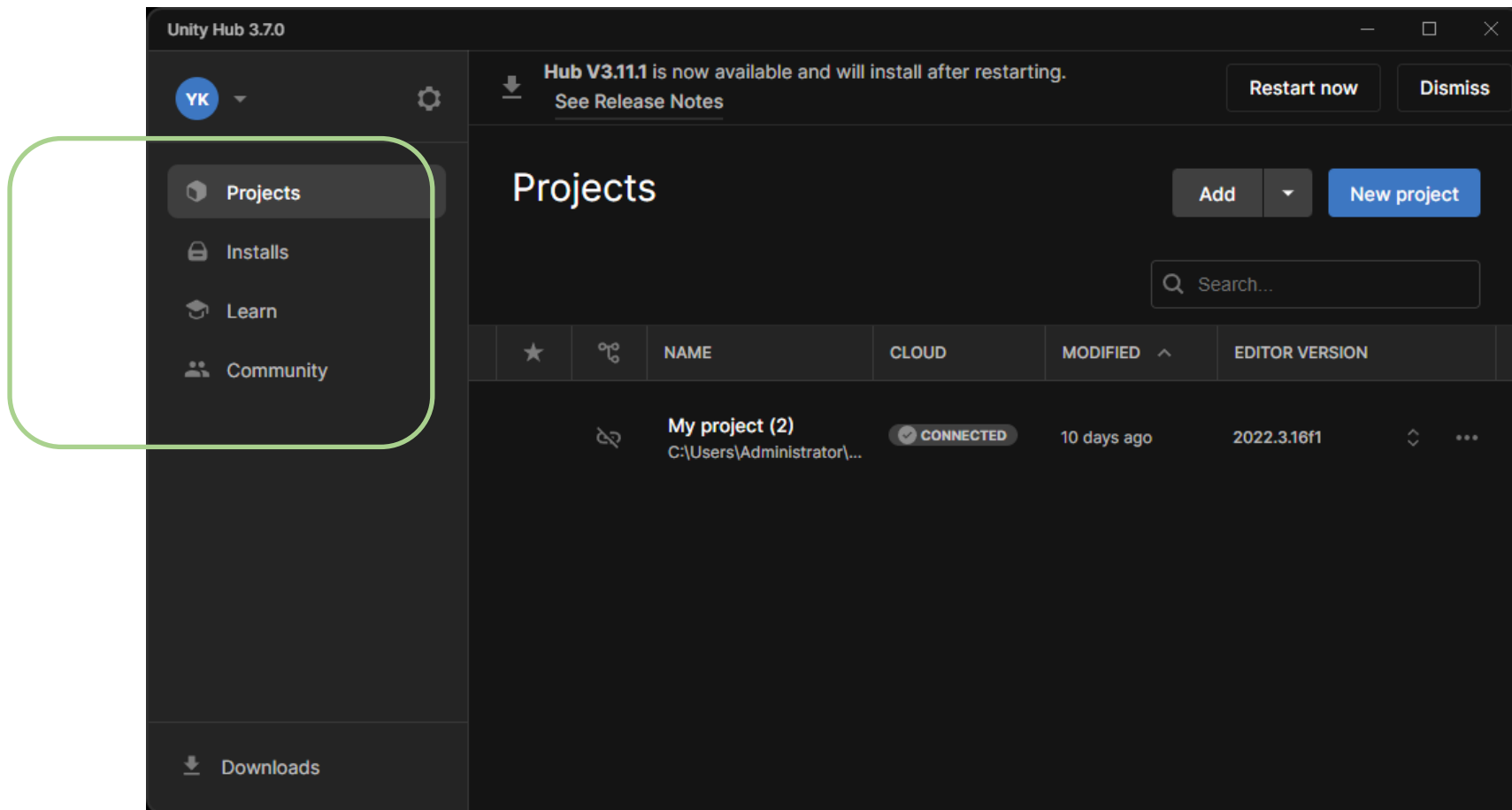
동명대학교 게임공학과

## 학습목표

- 유니티에 익숙해 진다.
- 간단한 프로젝트를 만들어 본다.
- 프로젝트에 C# 코드를 추가해 본다.
- 2D 게임과 3D 게임의 차이에 대해 생각해 본다.

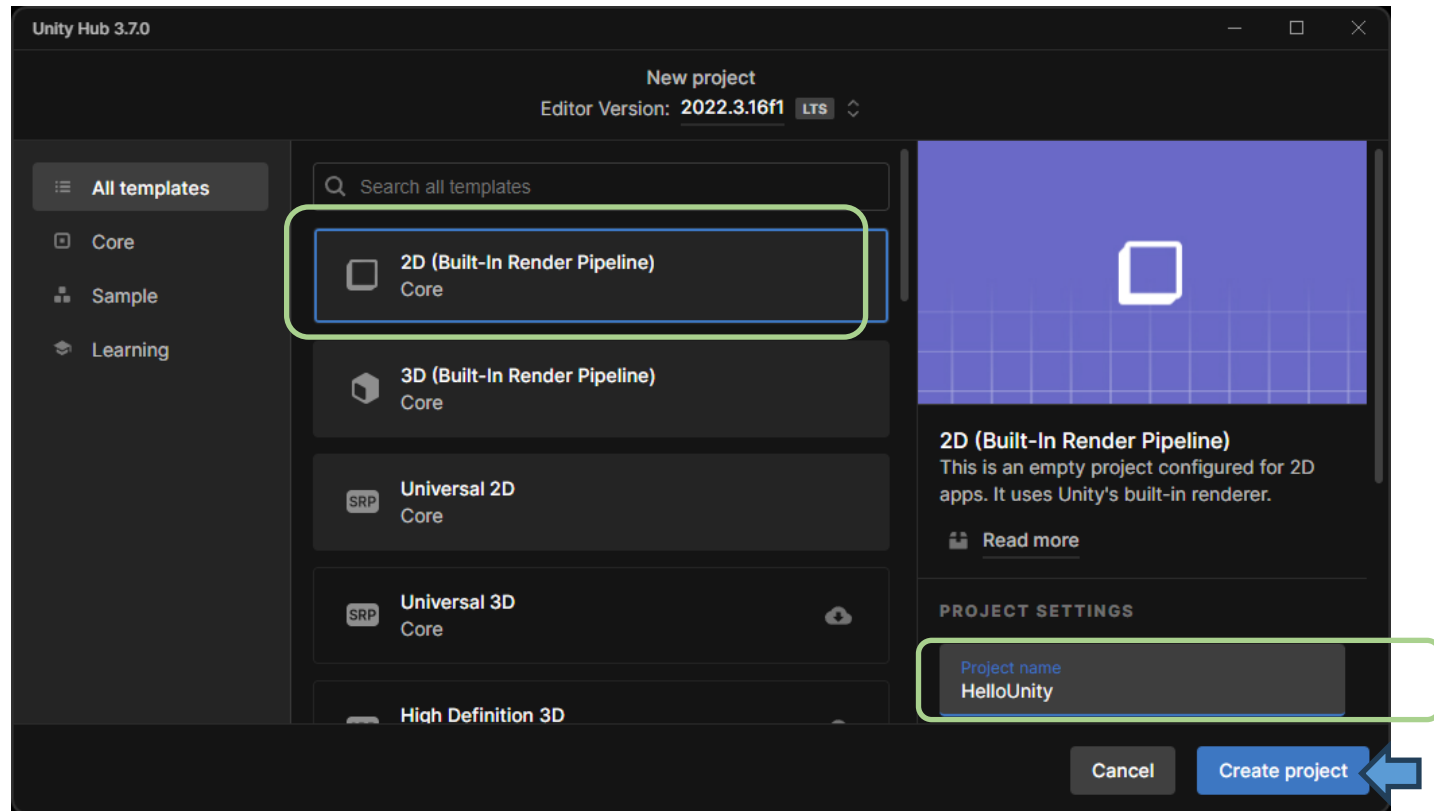
# 설치한 Unity Hub를 실행해 보자

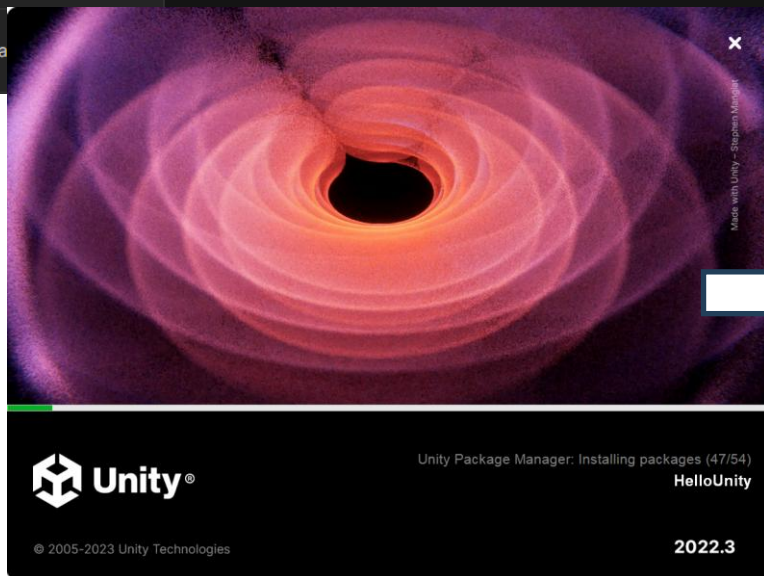
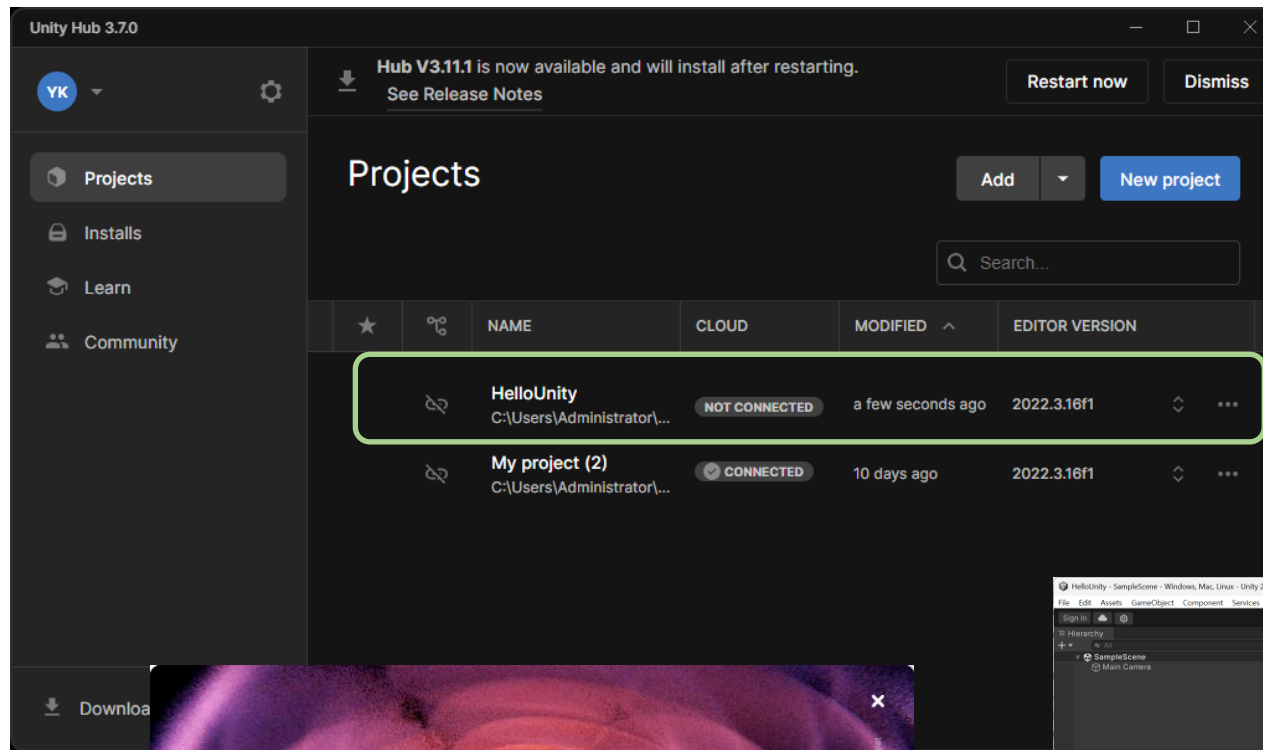
- 네 개의 탭이 탭 메뉴가 존재



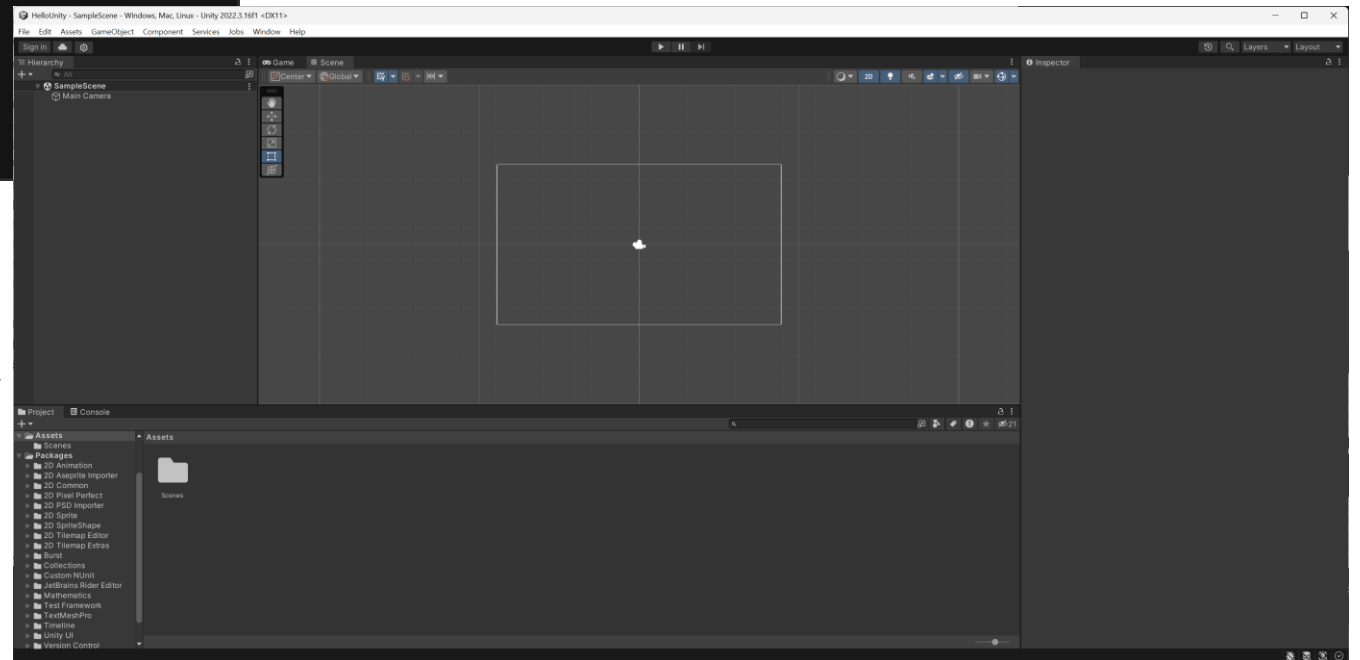
# 2D 게임프로젝트를 생성해 보자

- 네 개의 탭이 탭 메뉴가 존재



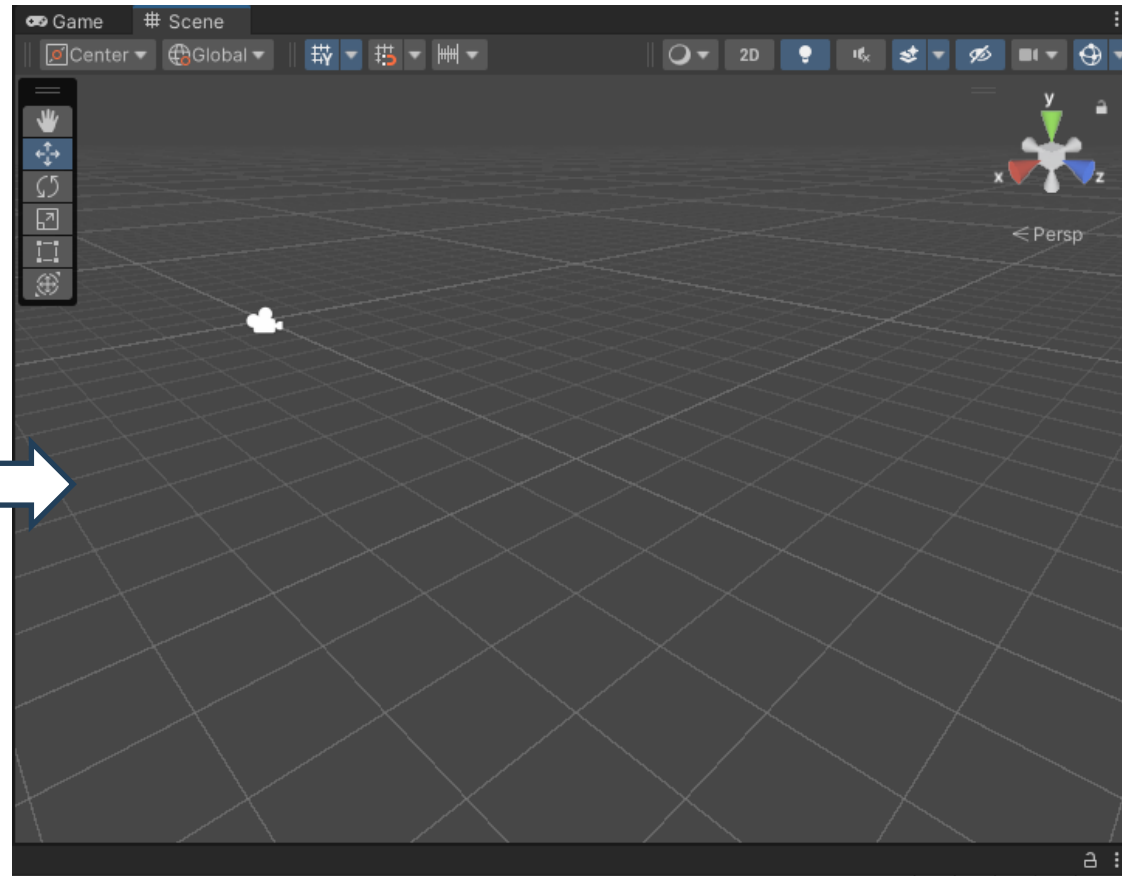
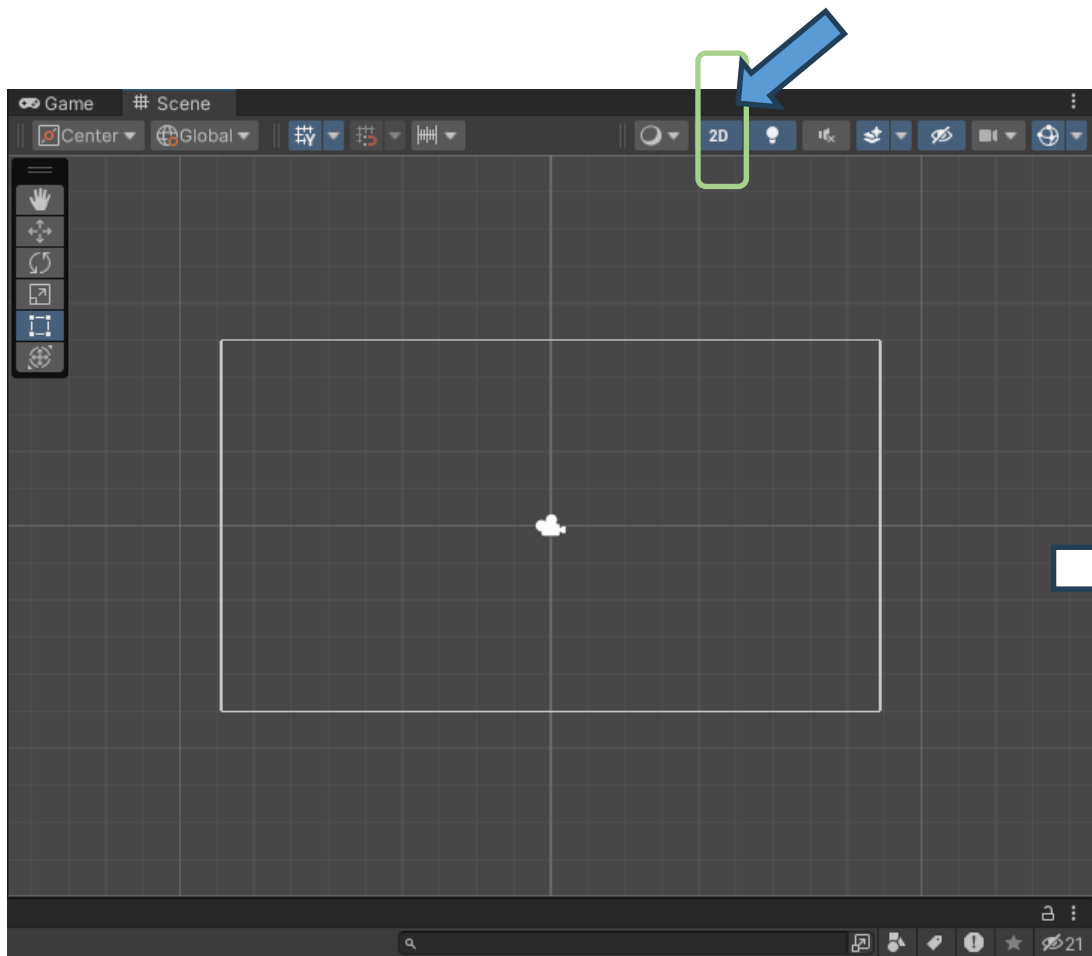


유니티 엔진 실행됨

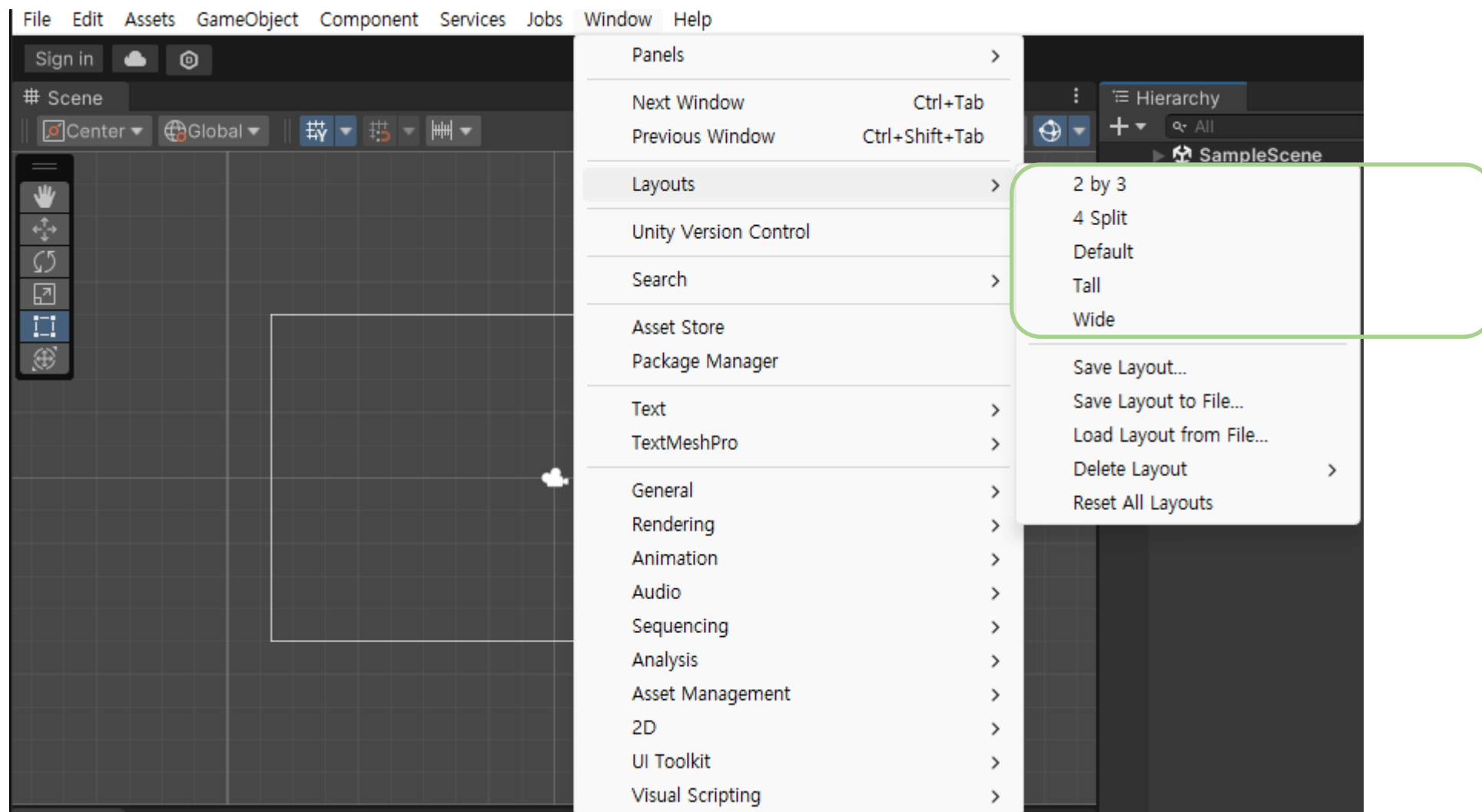


유니티 엔진 실행되고 프로젝트가 로딩된 상태

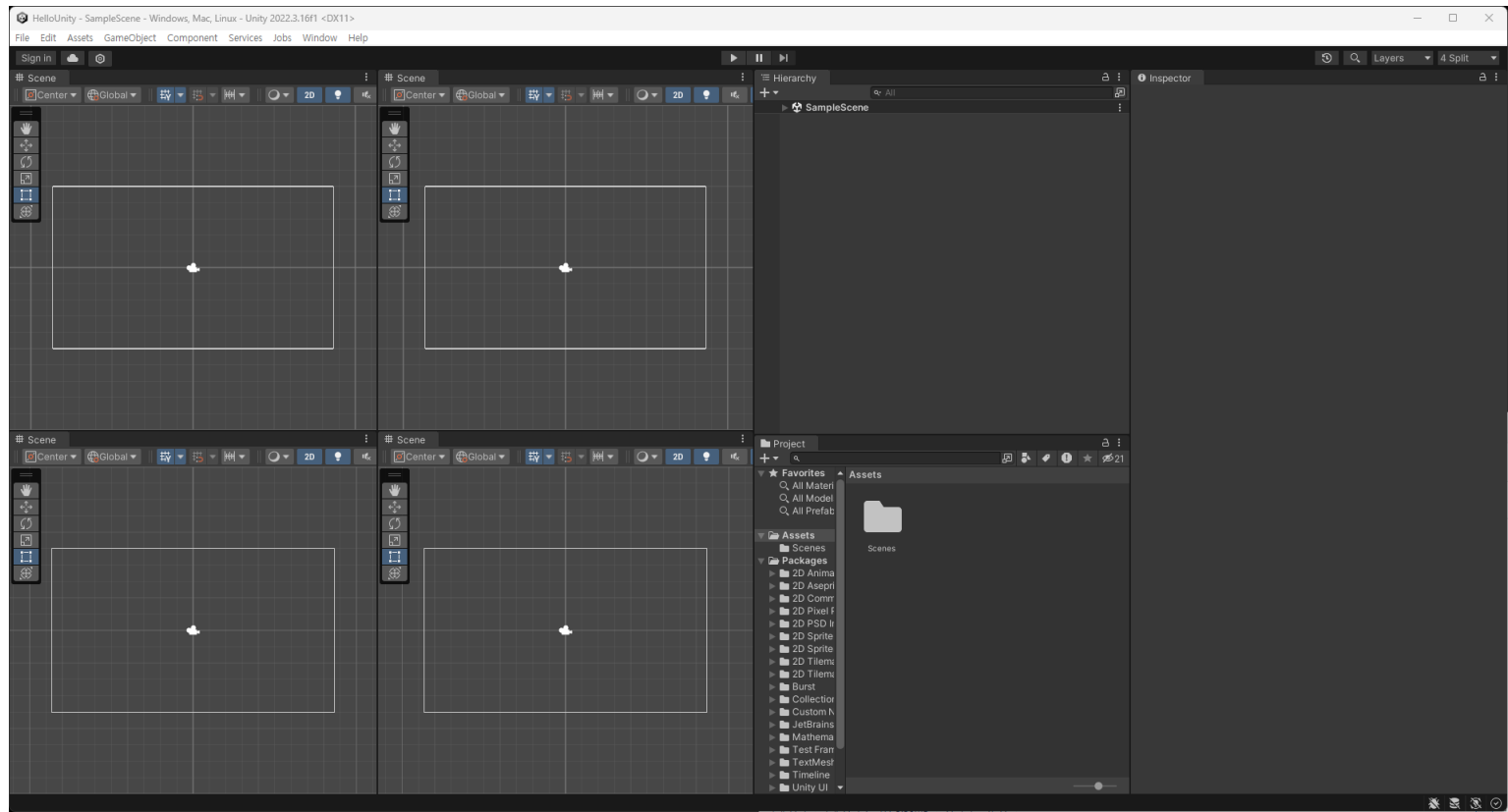
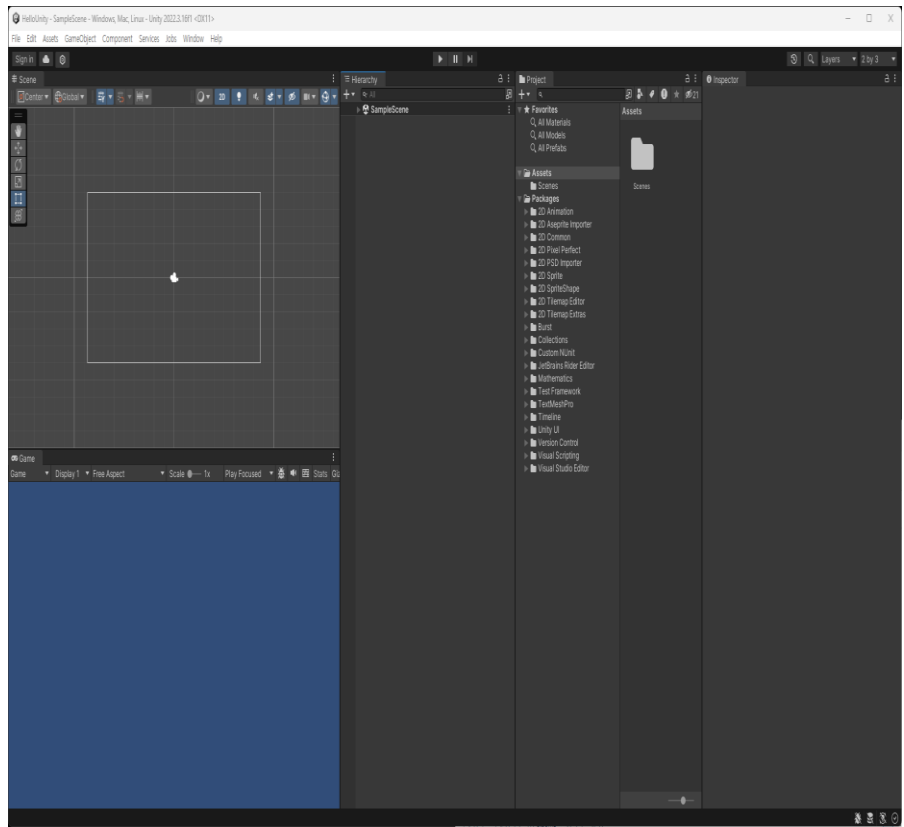
- 2D 공간은 3D 공간의 일부



# 다양한 윈도우 레이아웃 테스트



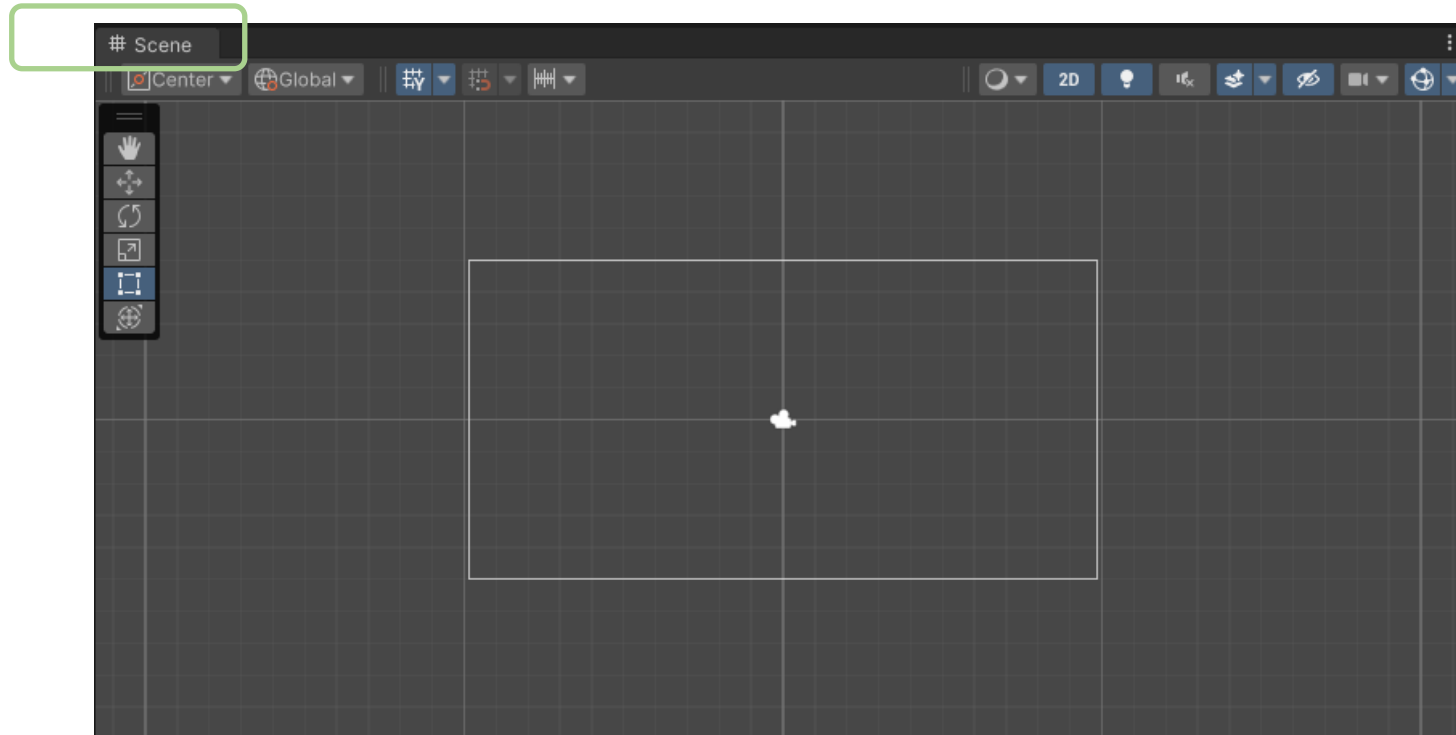
# 다양한 윈도 레이아웃 테스트





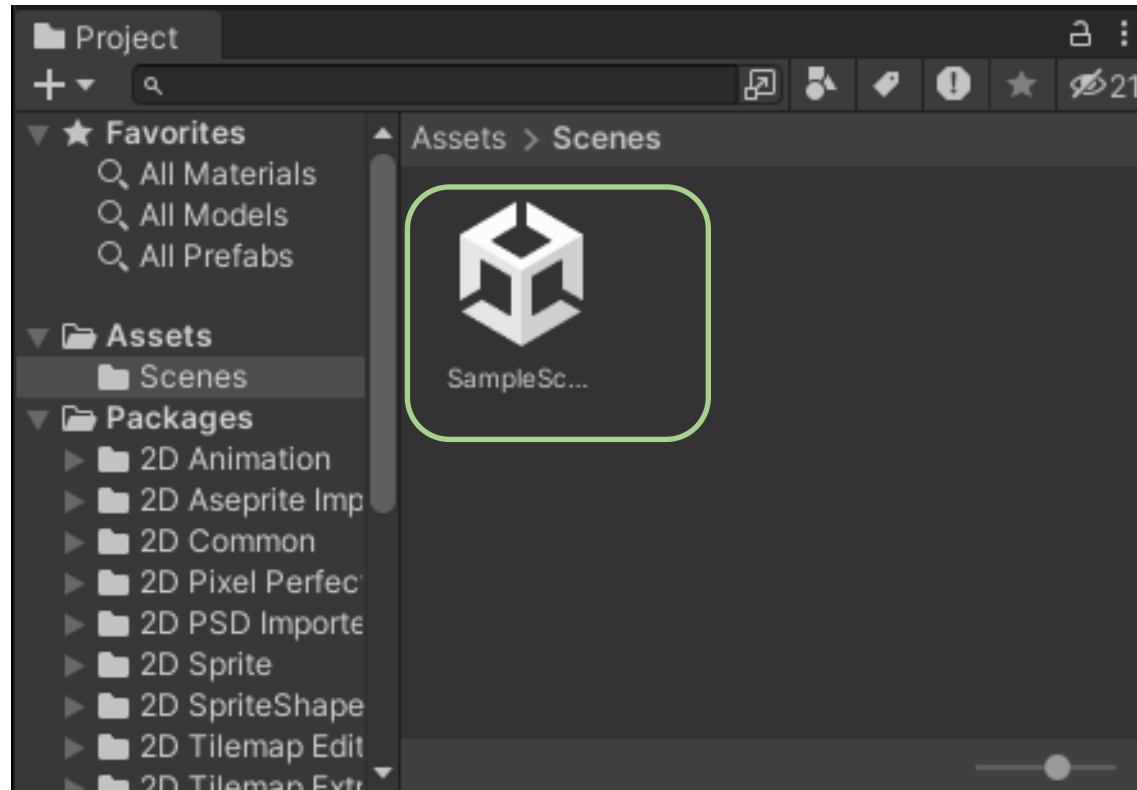
# 장면(Scene) – 게임 레벨

- 장면은 유니티에서 사용되는 일종의 Asset
- 장면에는 유니티 게임 프로젝트의 전부 혹은 일부 콘텐츠가 담김
- 유니티 프로젝트에서 개발은 이 장면 단위로 이루어짐

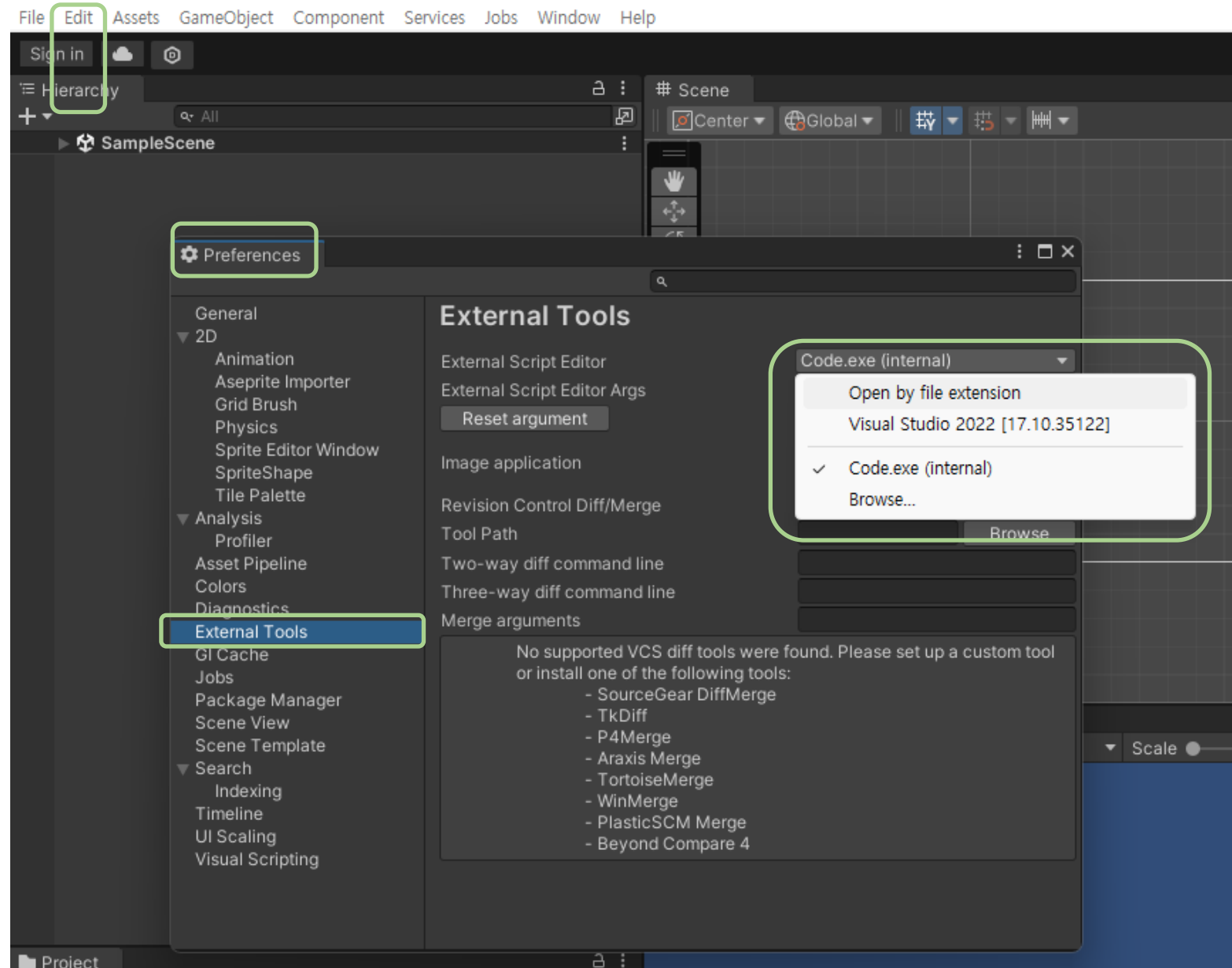


# 기본 장면 – SampleScene

- 유니티 프로젝트를 생성하면 SampleScene이 생성됨



# 설정

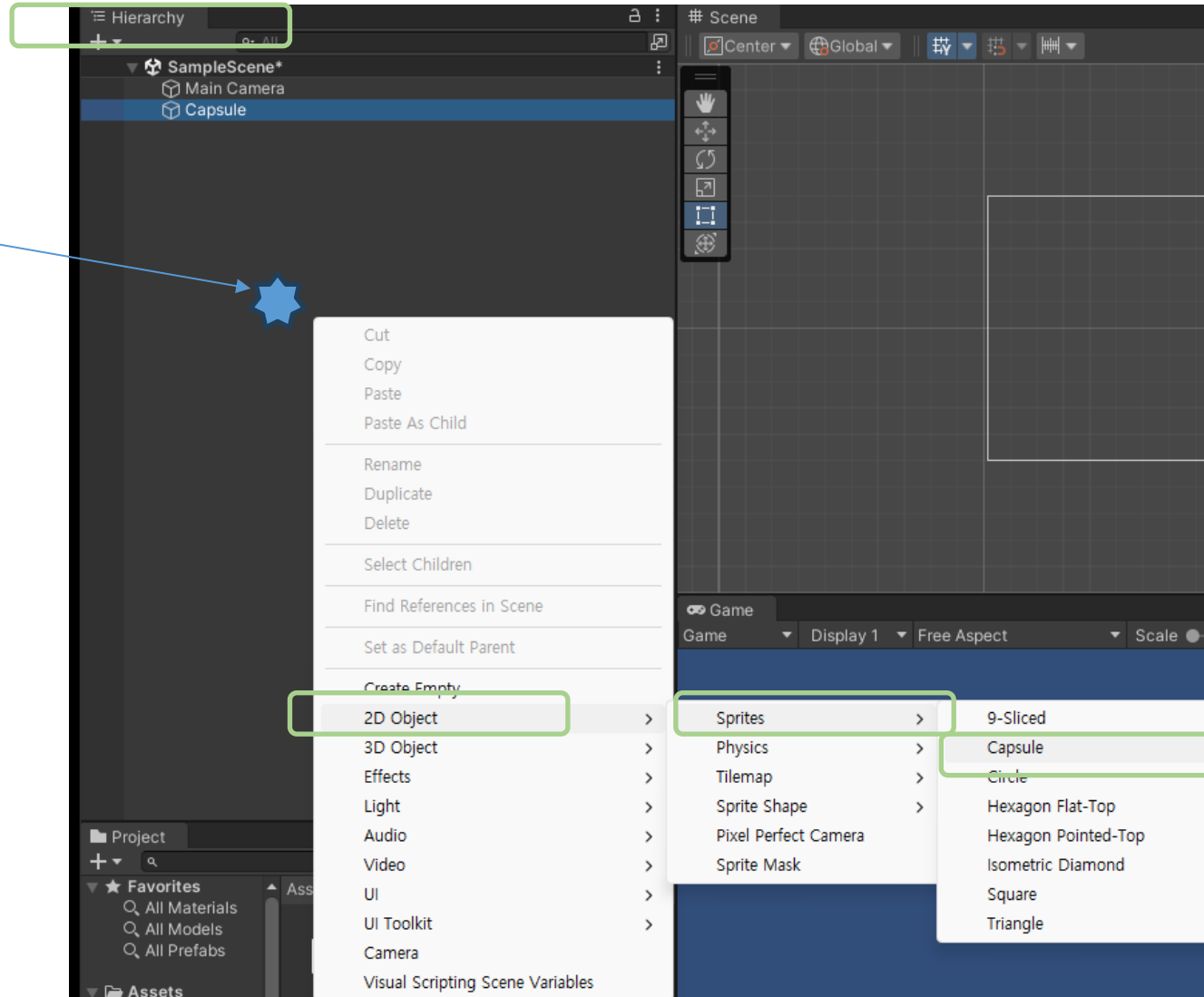


# 간단한 객체를 넣고 움직여 보자

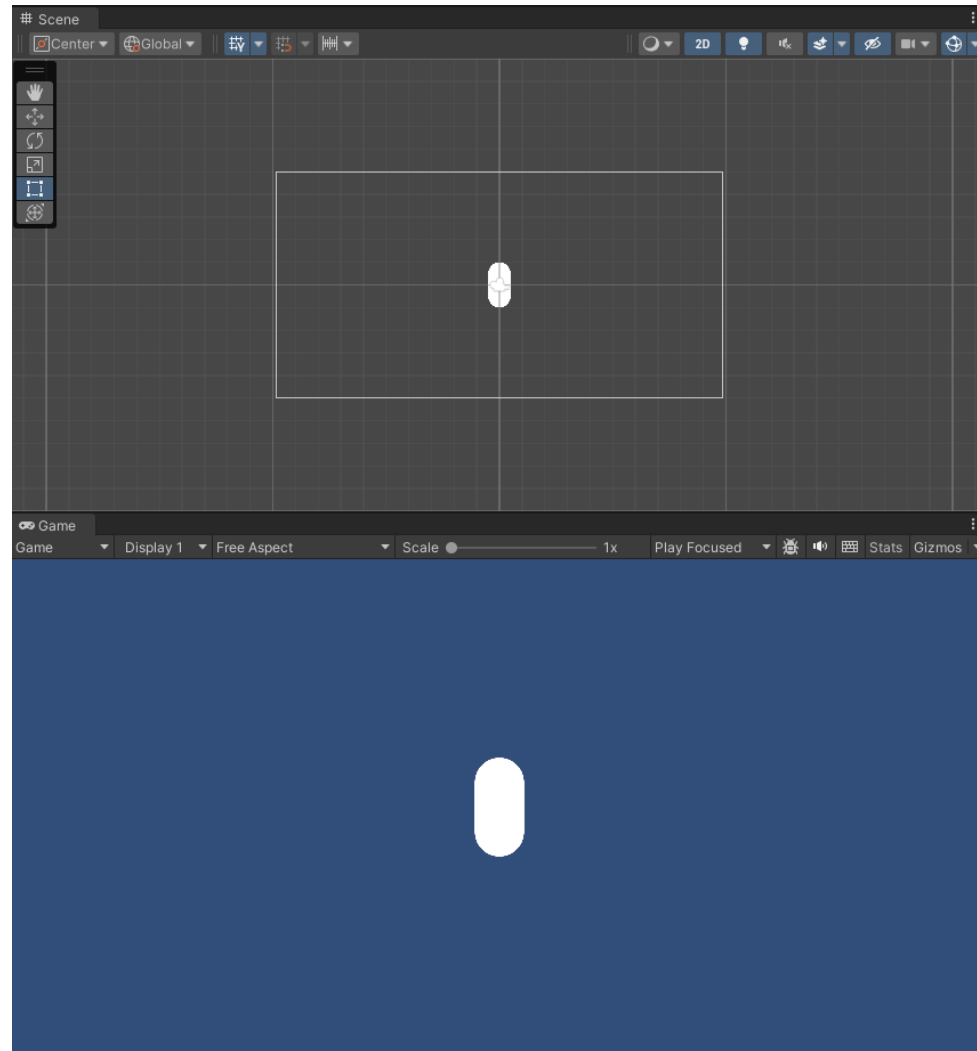
- 2D 게임 → 스프라이트 객체 사용
  - 캡슐 스프라이트 추가
  - 캡슐 스프라이트에 C# 코드 연결
  - 코딩을 통해 스프라이트 움직이기

# 캡슐 스프라이트 추가

Right click!



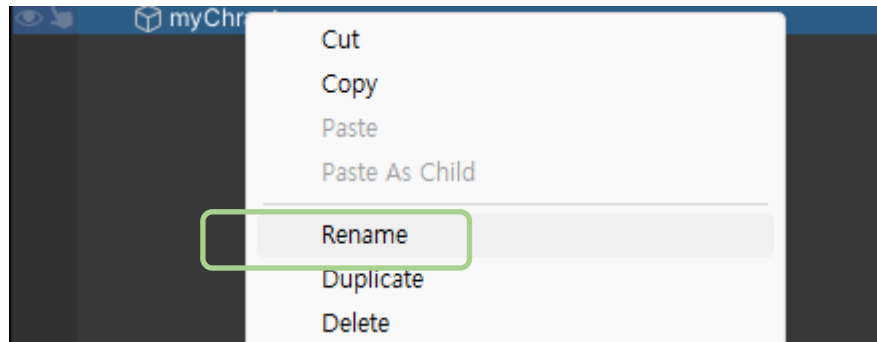
# 캡슐 스프라이트 확인



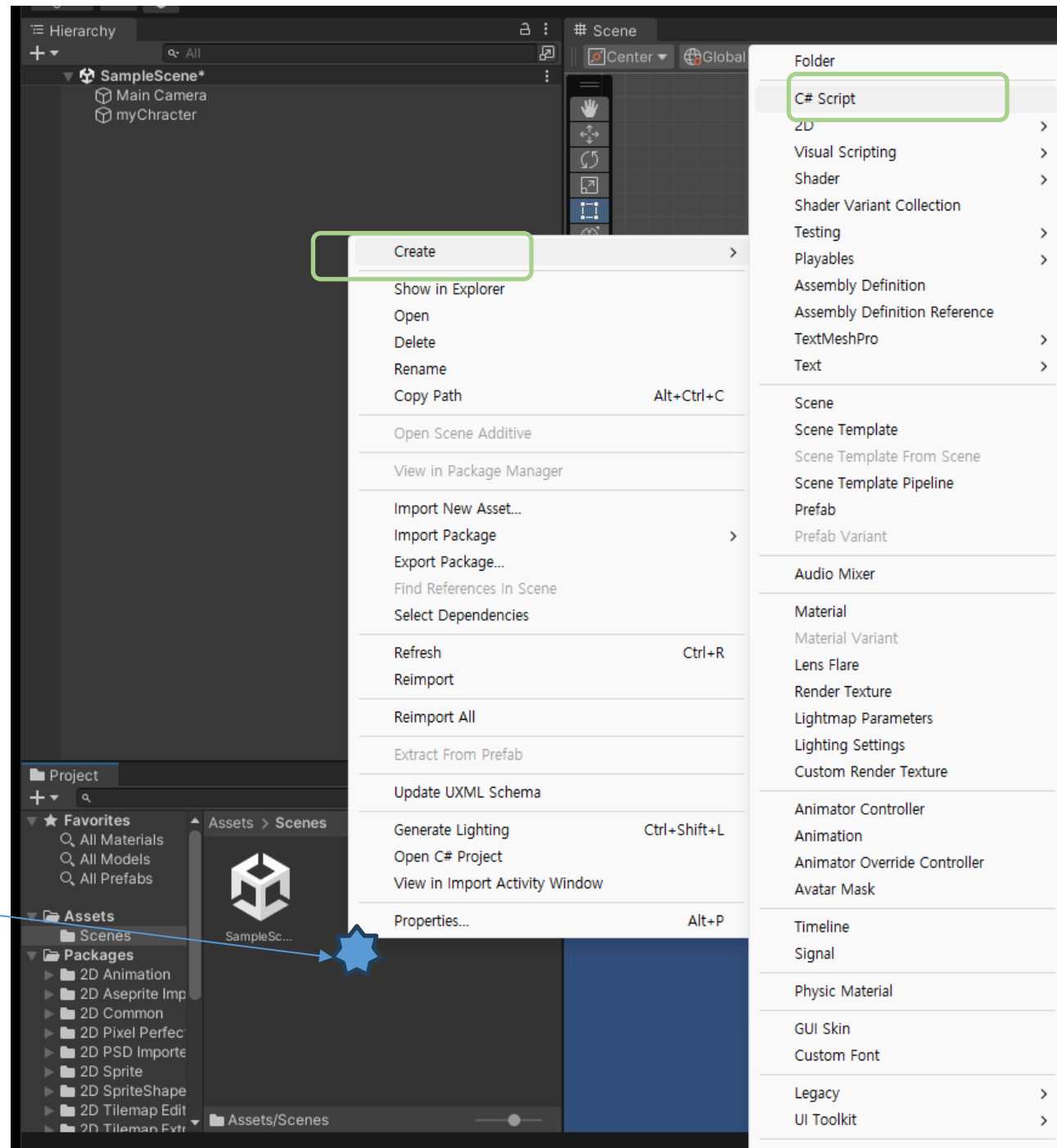
# 캡슐 스프라이트 움직이기 위한 “method” 사용

- 유니티의 메소드
  - 우리의 게임이 무엇인가를 할 수 있도록 하는 코드의 덩어리를 실행시키는 도구
  - 함수와 비슷한 개념 – 메소드는 객체에 딸린 함수
- 사용가능한 메소드
  - 유니티에서 제공하는 메소드
  - 우리가 직접 만든 메소드
- 메소드를 만들고 실행하는 것
  - 만들기: 무엇인가를 할 수 있는 코드를 메소드로 작성해 두는 것
  - 실행: 만들어 놓은 메소드를 호출하여 해당 동작이 이루어지게 하는 것

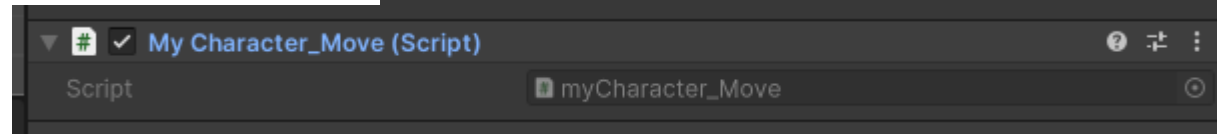
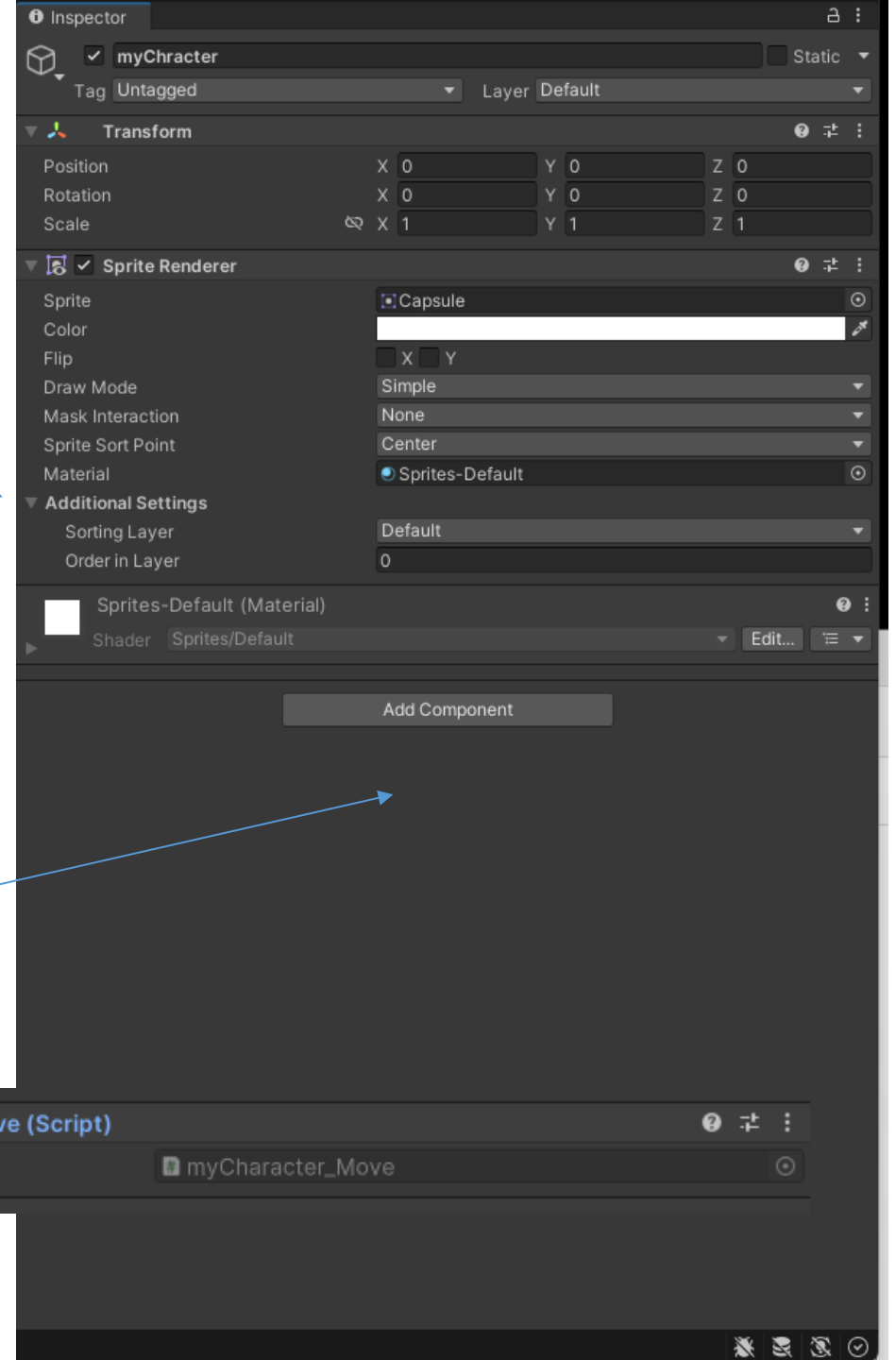
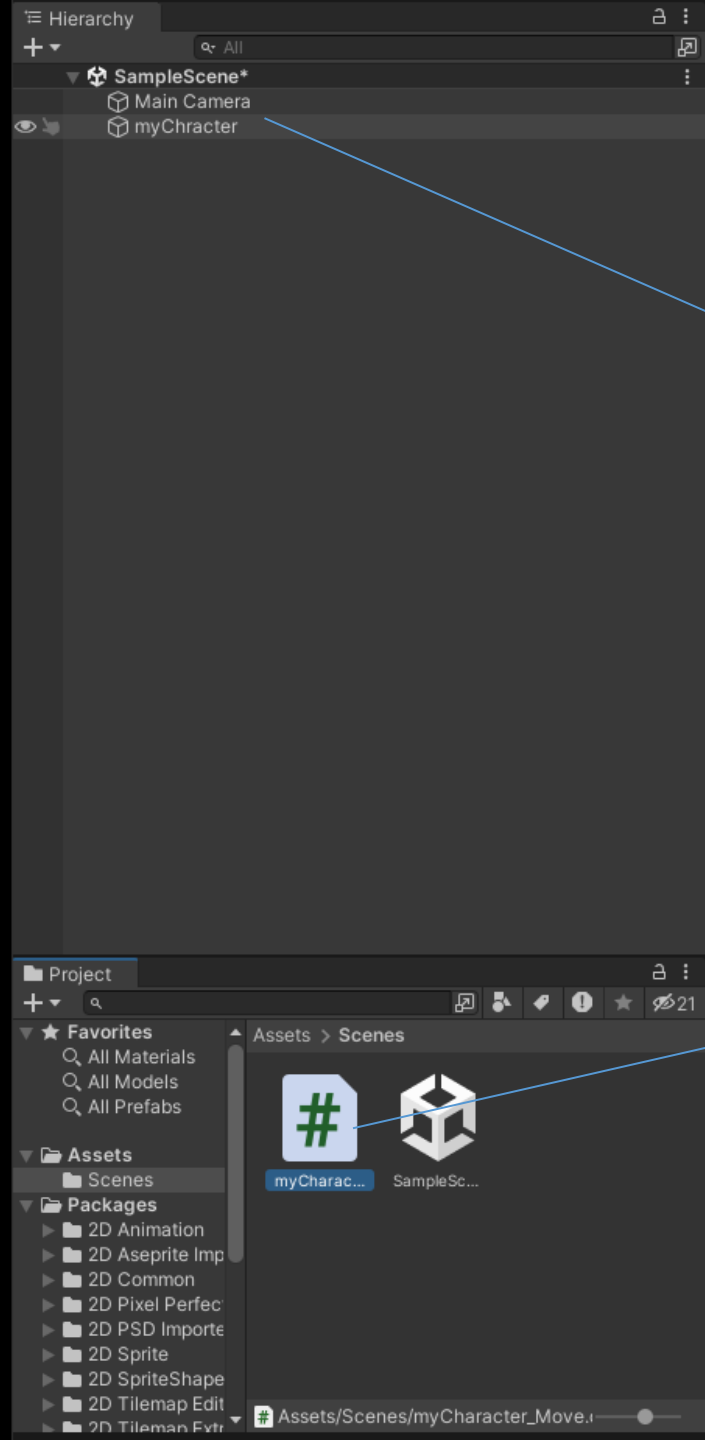
# 스크립트를 연결하기



Right click!

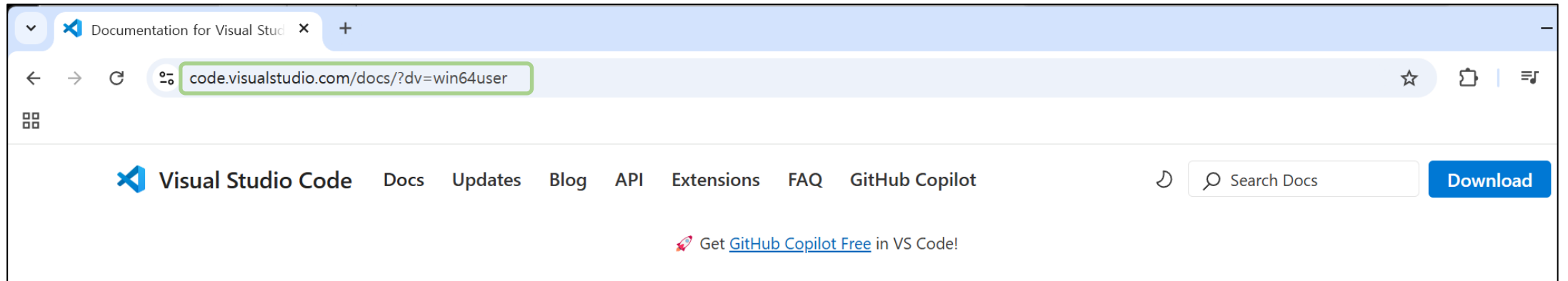




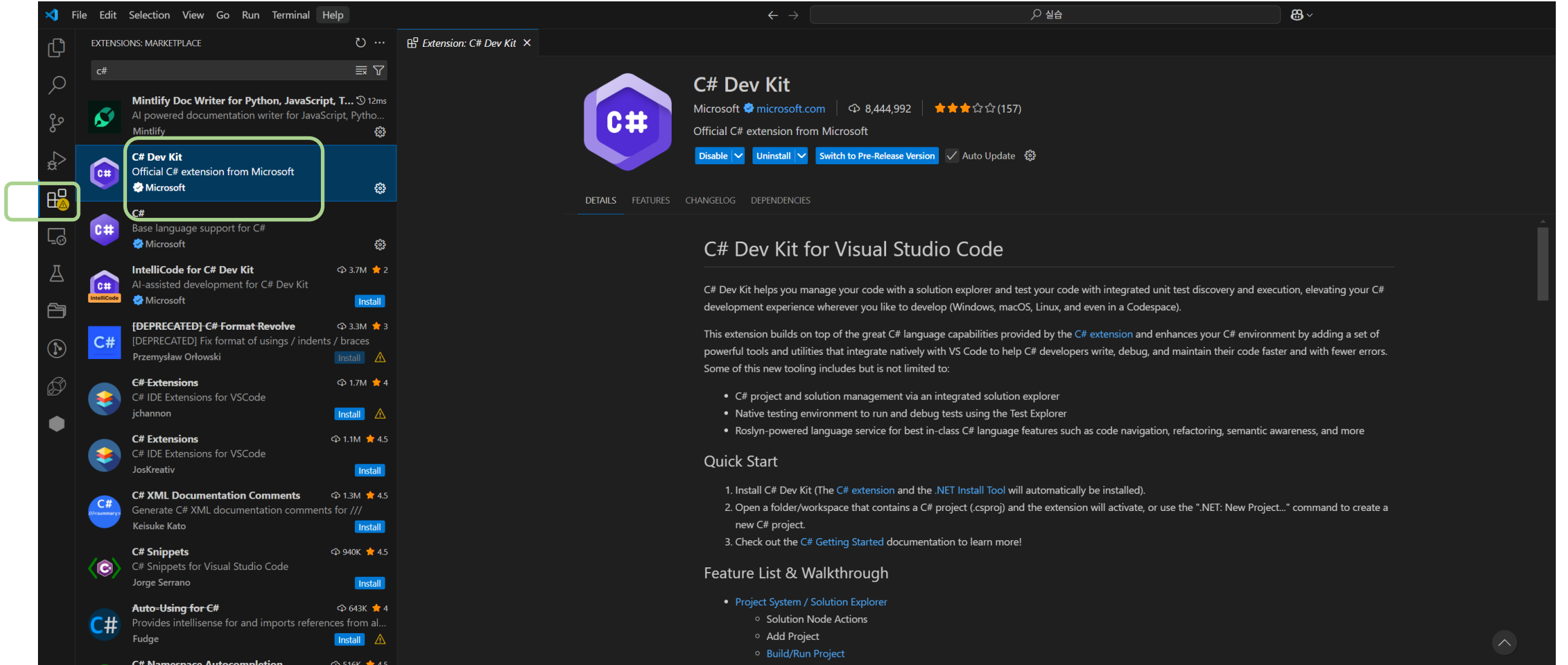


# 스크립트를 수정해 보자

- 스크립트를 두 번 클릭
  - 연결된 외부 개발 도구가 열림
  - Visual Studio Code를 연결해 두었음
  - 열리지 않는 경우에는 도구가 설치되지 않은 상태임
- 필요한 경우 설치



## 필요한 Extension



# 필요한 Extension

The screenshot displays the Visual Studio Code interface with the Unity extension marketplace open. The left sidebar shows a list of extensions, with 'Unity' highlighted. The main panel shows the details for the 'Unity' extension by Microsoft, which integrates Visual Studio Code with Unity. The extension is installed and has a 4.5-star rating. Below the extension details, there is a section titled 'Unity for Visual Studio Code' which describes the extension's purpose and lists its features.

**Unity**  
Integrates Visual Studio Code with Unity  
Microsoft

**Debugger for Unity**  
Unity debugger extension  
Unity Technologies

**Unity Tools**  
Various tools to help with Unity development  
Avin Zarlez

**Unity Code Snippets**  
All snippets you need for Unity3D development  
Kleber Silva

**Unity Snippets**  
Unity MonoBehaviour function snippets with document...  
Ycleptic Studios

**eppzi! (C# theme for Unity)**  
Carefully designed colors with meanings.  
eppzi!

**Unity Toolbox**  
Awesome tools for VSCode & Unity  
maxsroka

**Unity Snippets Modified**  
Unity MonoBehaviour function snippets with document...  
lionize

**Unity Code Snippets**  
All snippets you need for Unity3D development  
cemuka

**Unity Dev Pack**  
Some extensions that will help you use VSCode with U...  
fabriciohod

**Unity Code**  
All snippets you need for Unity3D, Unity2D, UnityGame...  
Bakken Bui-Gemini

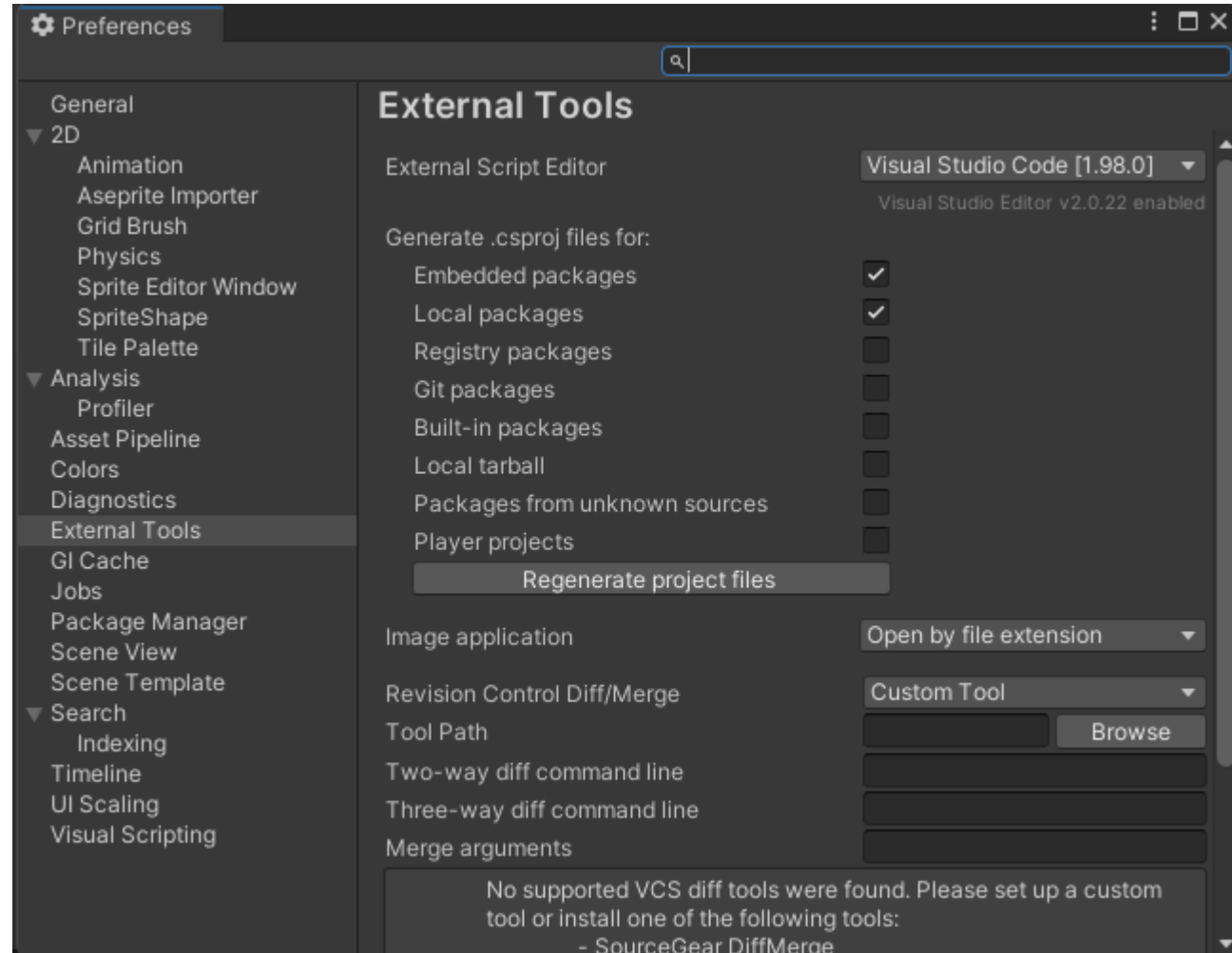
## Unity for Visual Studio Code

The Unity extension provides C# developers with a lightweight and streamlined [Unity](#) development experience on Visual Studio Code. It builds on top of the rich C# capabilities provided by the [C# Dev Kit](#) and [C#](#) extensions. This extension integrates natively with Visual Studio Code and includes multiple productivity features including:

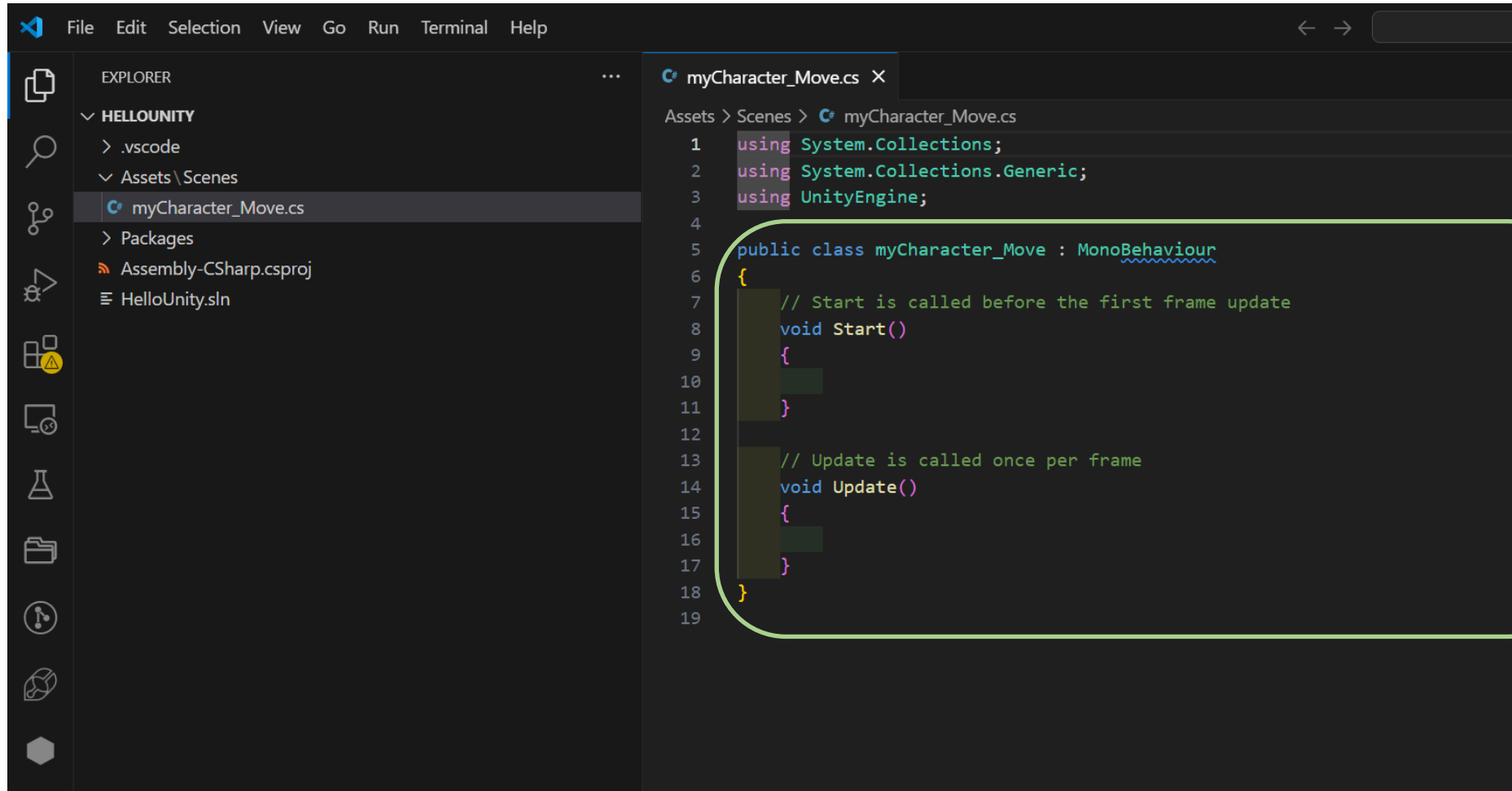
- A Unity debugger to debug your Unity editor and your Unity players.
- Unity specific C# analyzers and refactorings.
- Code coloration for Unity file formats (.asmdef, .shader, .uss, .uxml).

The screenshot also shows a preview of the Unity editor running a game, and a code editor displaying C# code for a Unity script.

# Unity Preference



# 코드 편집



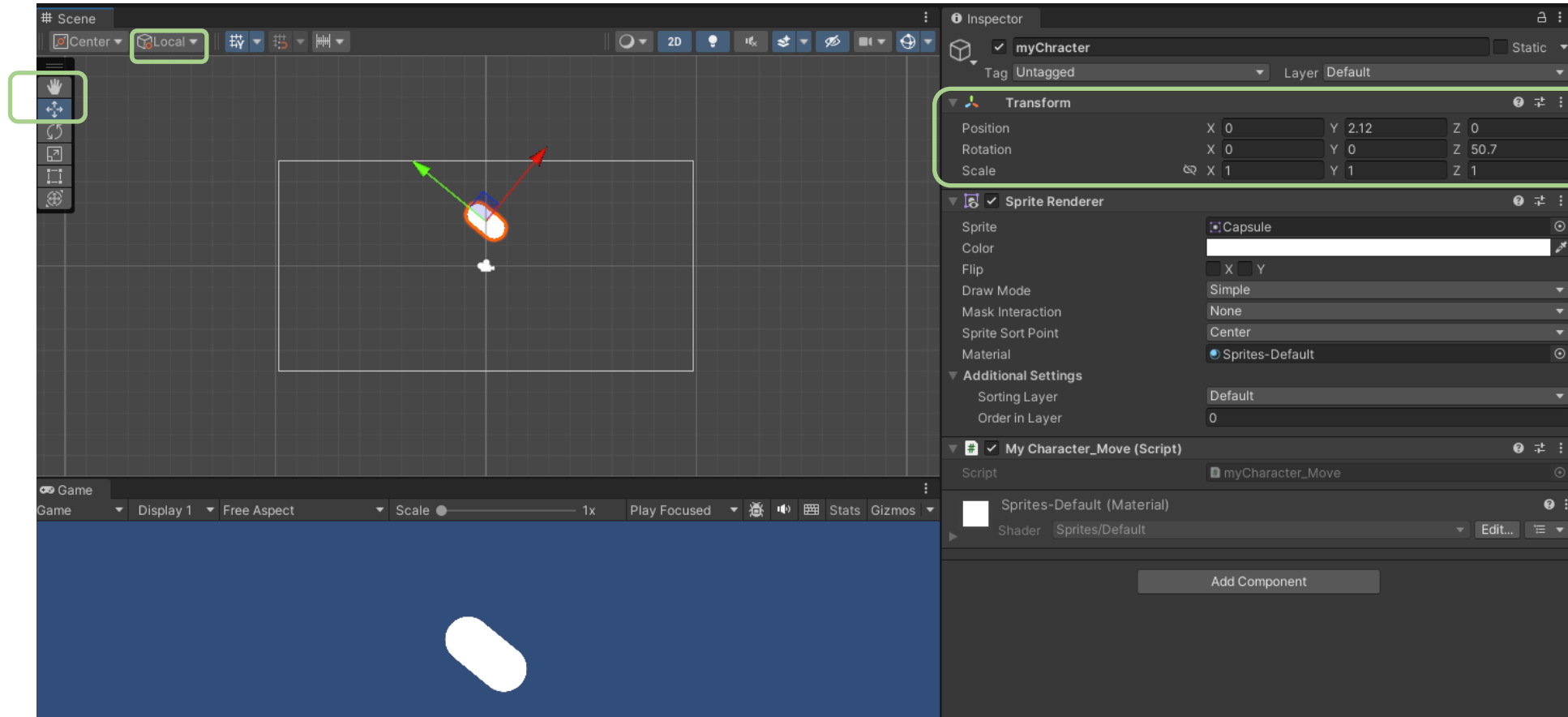
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class myCharacter_Move : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     void Start()
9     {
10
11     }
12
13     // Update is called once per frame
14     void Update()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

이 클래스가  
우리의 캐릭터에  
연결되어 있음

두 개의 클래스  
Start, Update는  
자동으로 호출되는  
메소드

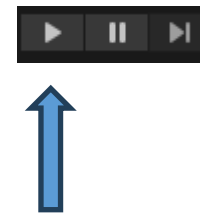
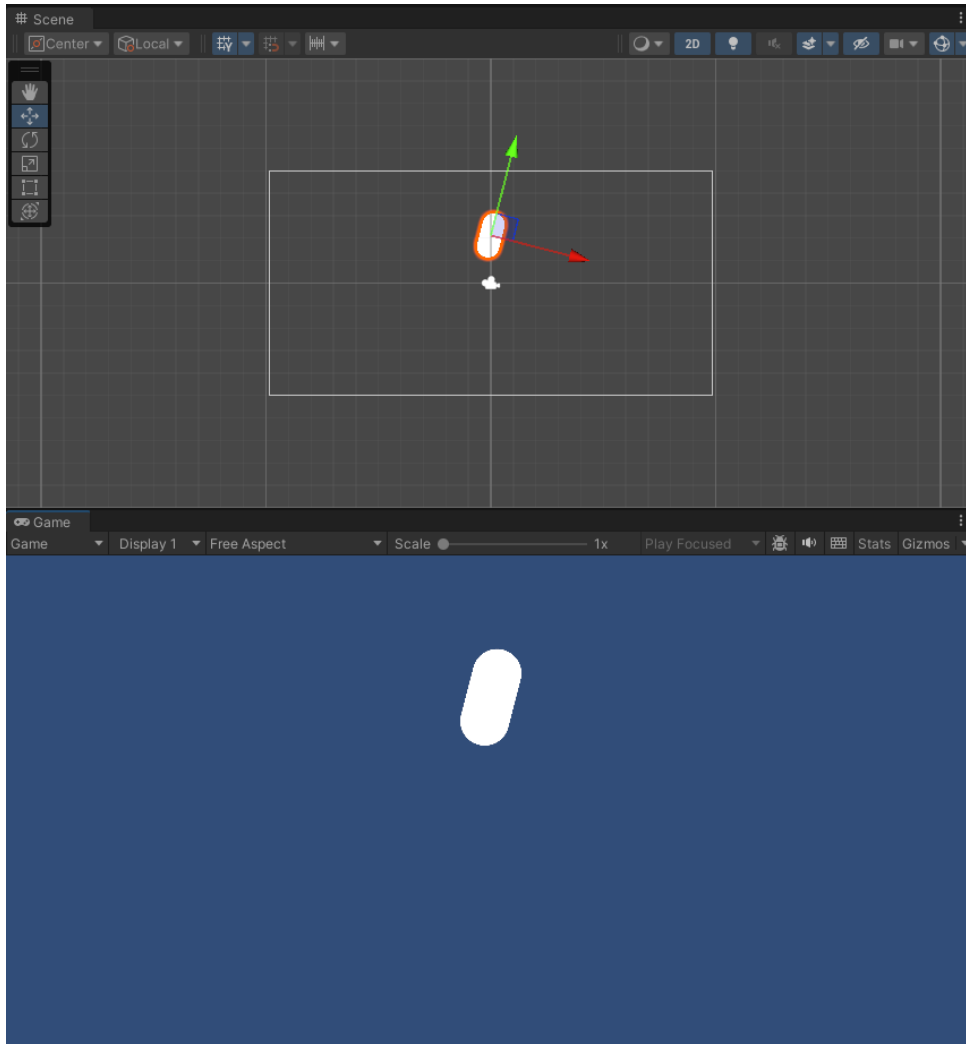
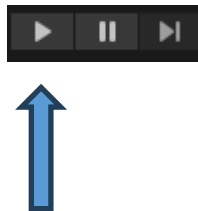
Start: 플레이될 때  
Update: 매 프레임

# 인스펙터(Inspector)로 값의 변화 확인



# 움직이는 첫 프로젝트

Play 버튼을 누른다



멈춰야지 변경이 가능

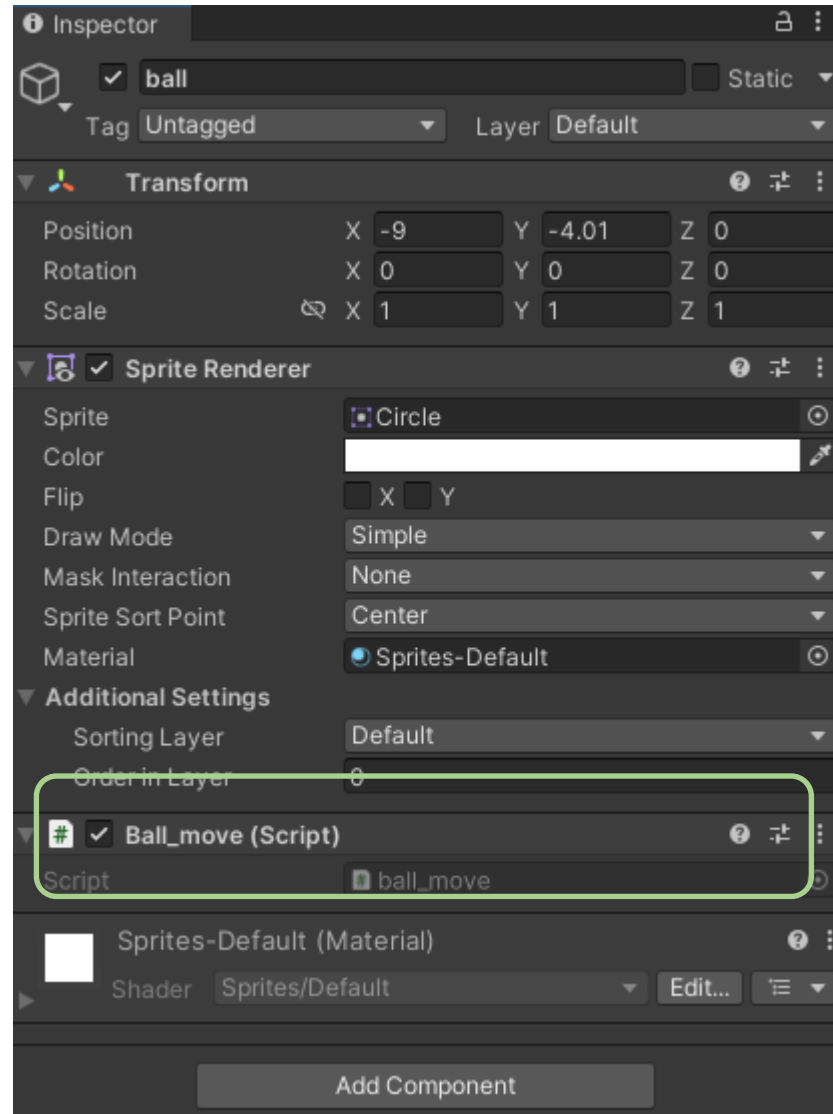
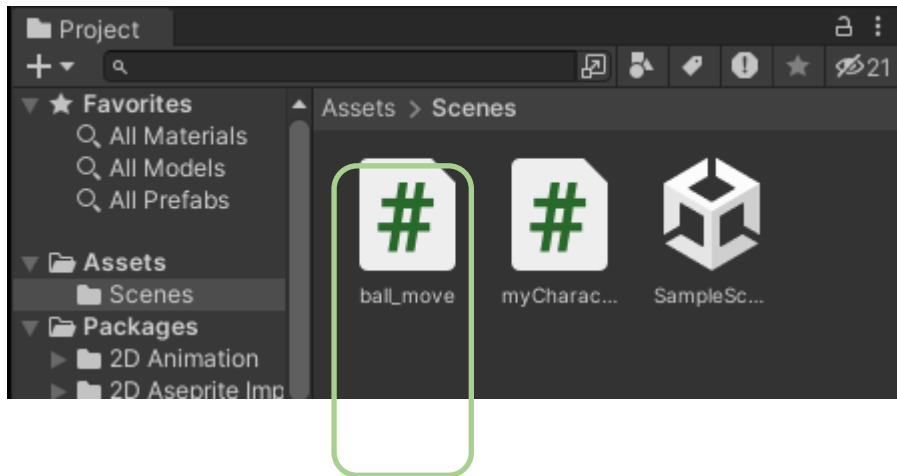
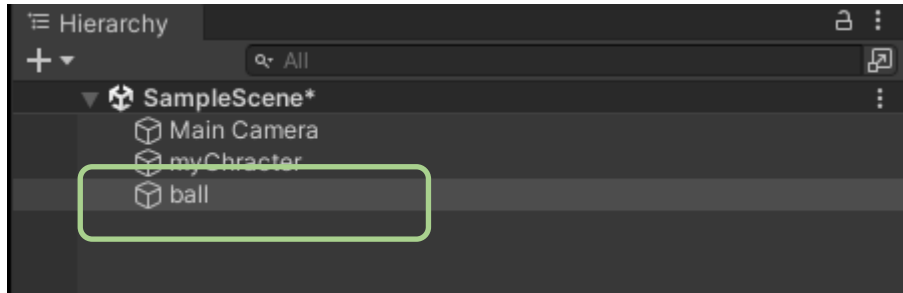


# 새로운 메소드: transform.Translate

```
13 // Update is called once per frame
    0 references
14 void Update()
15 {
16     transform.Rotate(0, 0.0f, 0.5f);
17     transform.Translate(0.0f, 0.01f, 0.0f);
18 }
19 }
20
```



# 새로운 캐릭터 만들어 보기



# 변수를 써서 코딩해 보기

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ball_move : MonoBehaviour
{
    float mass = 2.0f;
    float x = -9.0f;
    float y = -4.0f;
    float vx = 0.0f;
    float vy = 0.0f;
    float fx = 0.0f;
    float fy = 0.0f;
    float g = 9.8f;
    float total_time = 0.0f;

    // Start is called before the first frame update
    void Start()
    {
        fx = 40.0f;
        fy = 40.0f;
    }
}
```

```
// Update is called once per frame
void Update()
{
    float dt = Time.deltaTime;
    total_time += dt;
    if (total_time > 0.5f)
    {
        fx = 0.0f;
        fy = 0.0f;
    }
    float ax = fx / mass;
    float ay = fy / mass - g;
    vx += ax * dt; // v = u + at
    vy += ay * dt;
    x += vx * dt; // s = s + vt
    y += vy * dt;
    if (y < -4.5f) {
        vy = -vy;
        y = -4.5f + (-4.5f - y);
    }
    if (x > 10.0f) {
        vx = -vx;
        x = 10.0f - (x - 10.0f);
    }
    if (x < -10.0f) {
        vx = -vx;
        x = -10.0f + (-10.0f - x);
    }
    transform.position = new Vector3(x, y, 0);
}
}
```



Questions?