

Visual Studio Tools for Unity

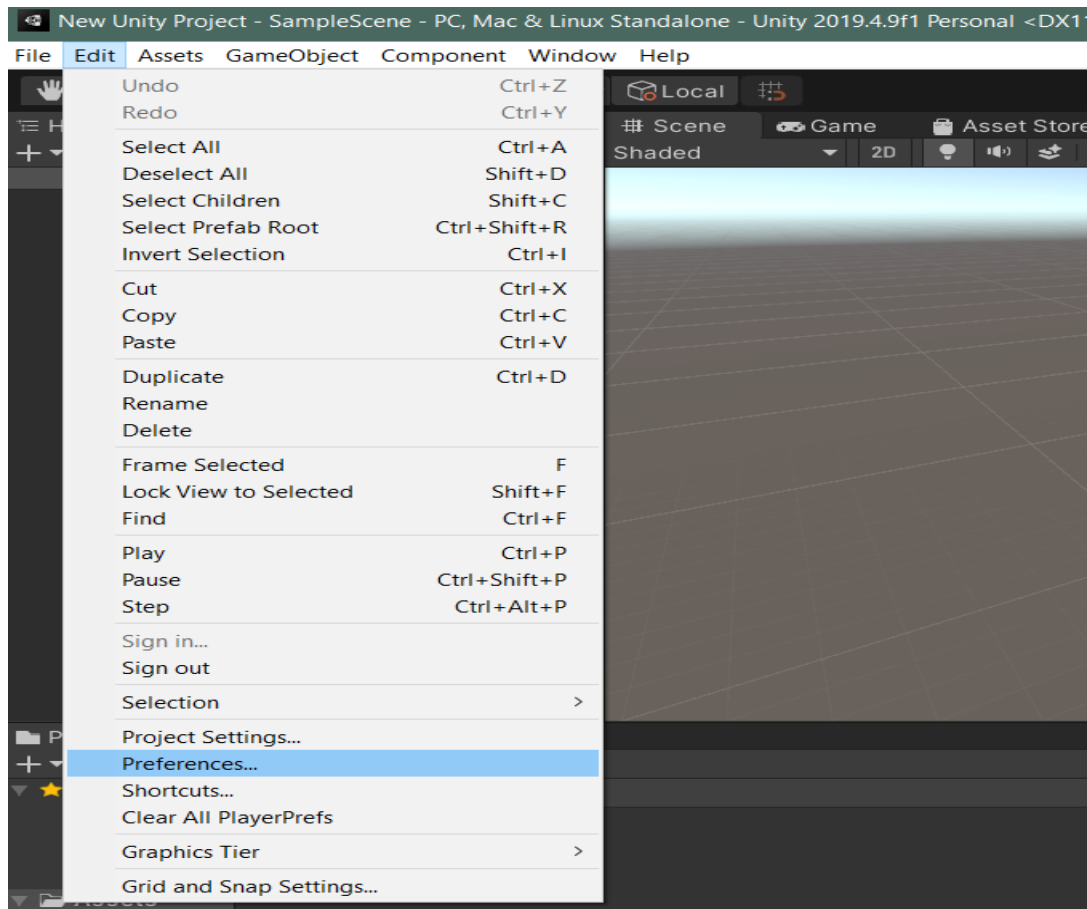
Visual Studio Tools for Unity is a free Visual Studio extension that turns Visual Studio into a powerful tool for developing cross-platform games and apps with Unity.

While the Unity editor is great for putting your game world together, you can't write your code in it. With Visual Studio Tools for Unity, you can use the familiar code editing, debugging and productivity features of Visual Studio to create editor and game scripts for your Unity project using C#, and you can debug them using Visual Studio's powerful debugging capabilities.

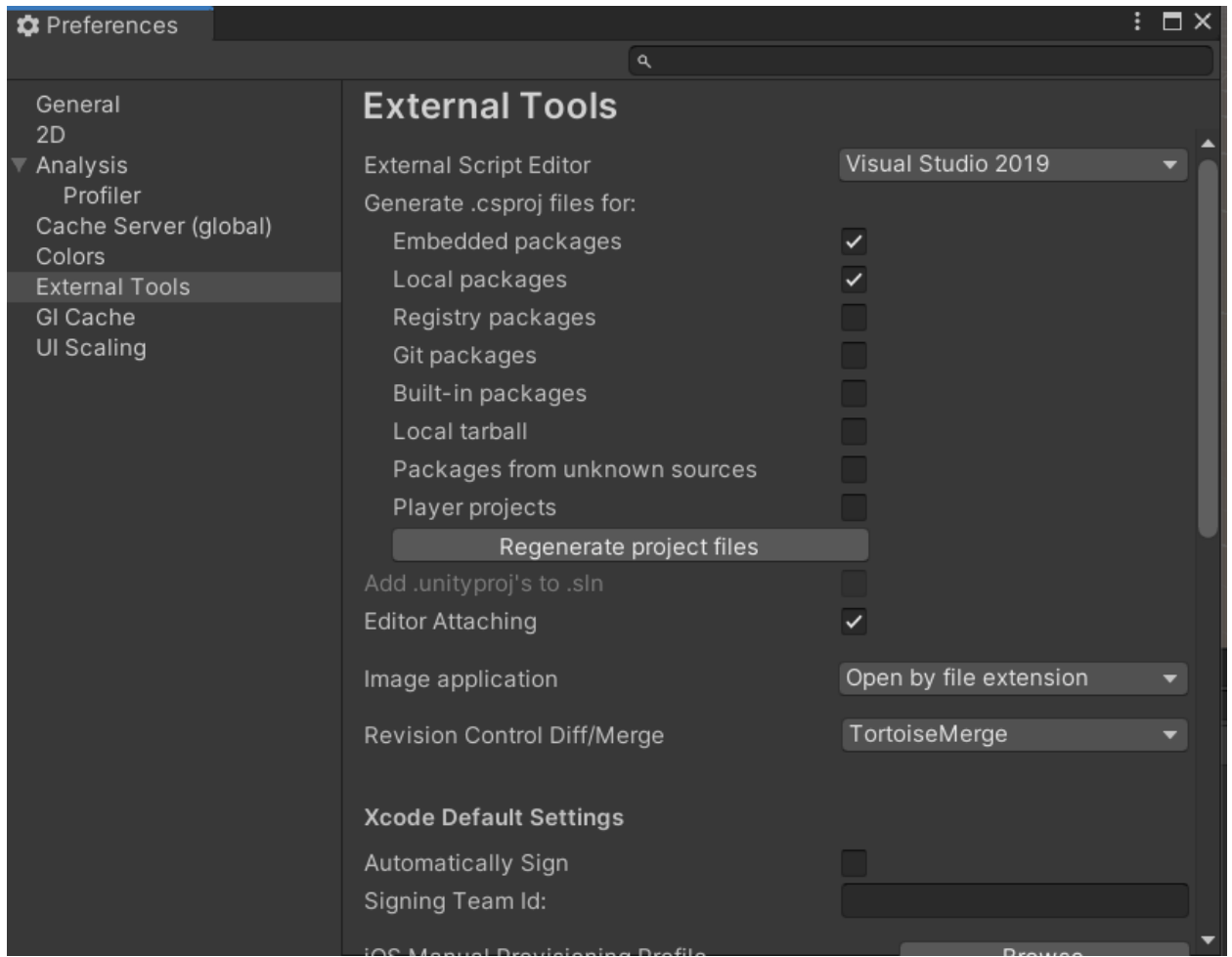
Configure Unity for use with Visual Studio

Starting with Unity 2018.1, Visual Studio should be the default external script editor in Unity. You can confirm this or change the external script editor to a specific version of Visual Studio:

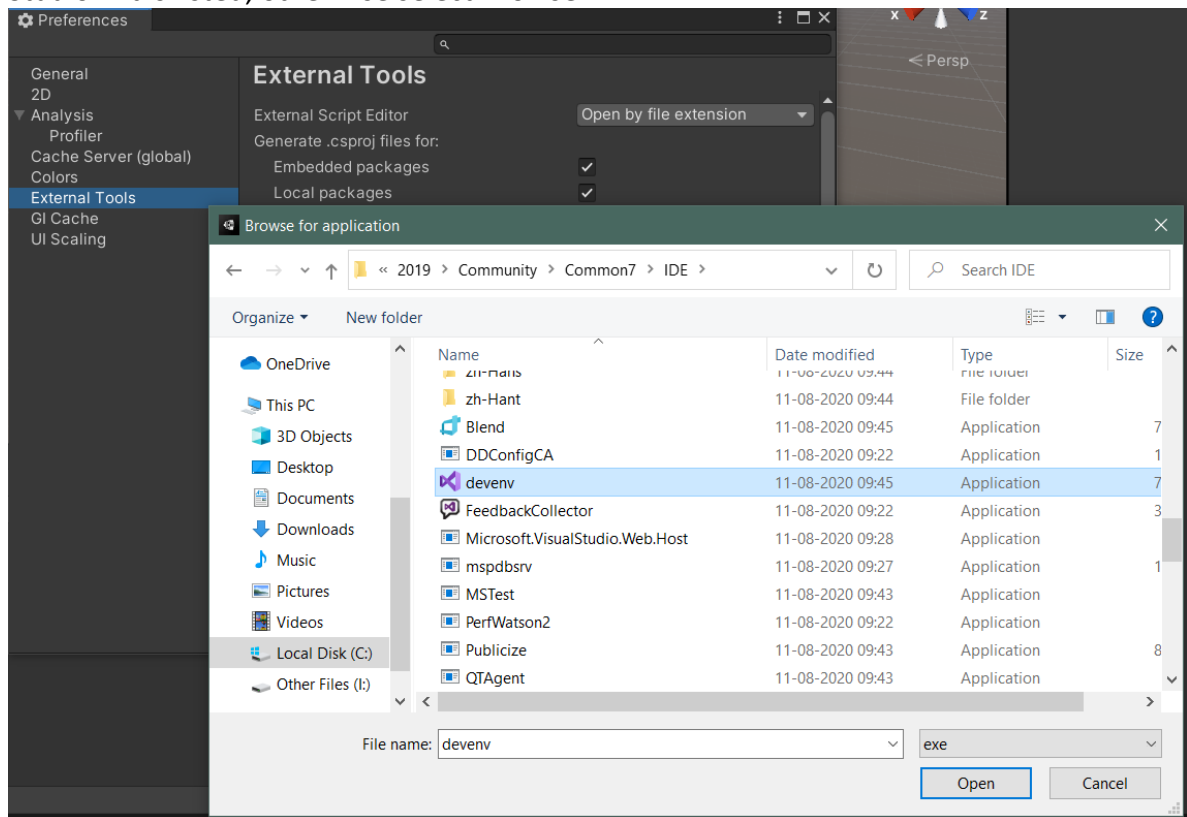
1. Open Unity
2. Select Preferences from the Edit menu.



3. In the Preferences dialog, select the External Tools tab.



- From the External Script Editor dropdown list, choose your desired version of Visual Studio if it is listed, otherwise select Browse.



- Once Visual Studio is selected in the External Script Editor list, confirm that the Editor Attaching checkbox is selected.
- Close the Preferences dialog to complete the configuration process.

Note:

Even though Visual Studio comes with its own C# compiler, and you can use it to check if you have errors in your c# scripts, Unity still uses its own C# compiler to compile your scripts. Using the Visual Studio compiler is still quite useful, because it means you don't have to switch to Unity all the time to check if you have any errors or not.

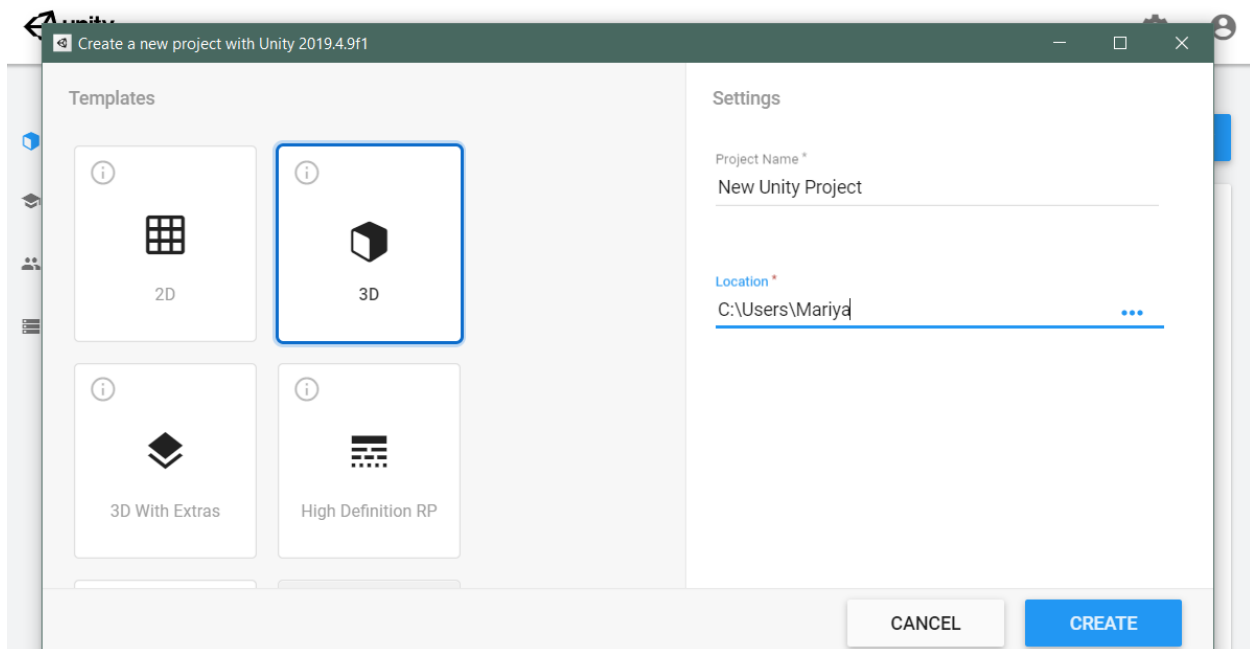
Visual Studio's C# compiler has some more features than Unity's C# compiler currently supports. This means that some code (especially newer c# features) will not throw an error in Visual Studio but will in Unity.

Sample C# script to test the integration of Unity and Visual Studio

Scenes in Unity

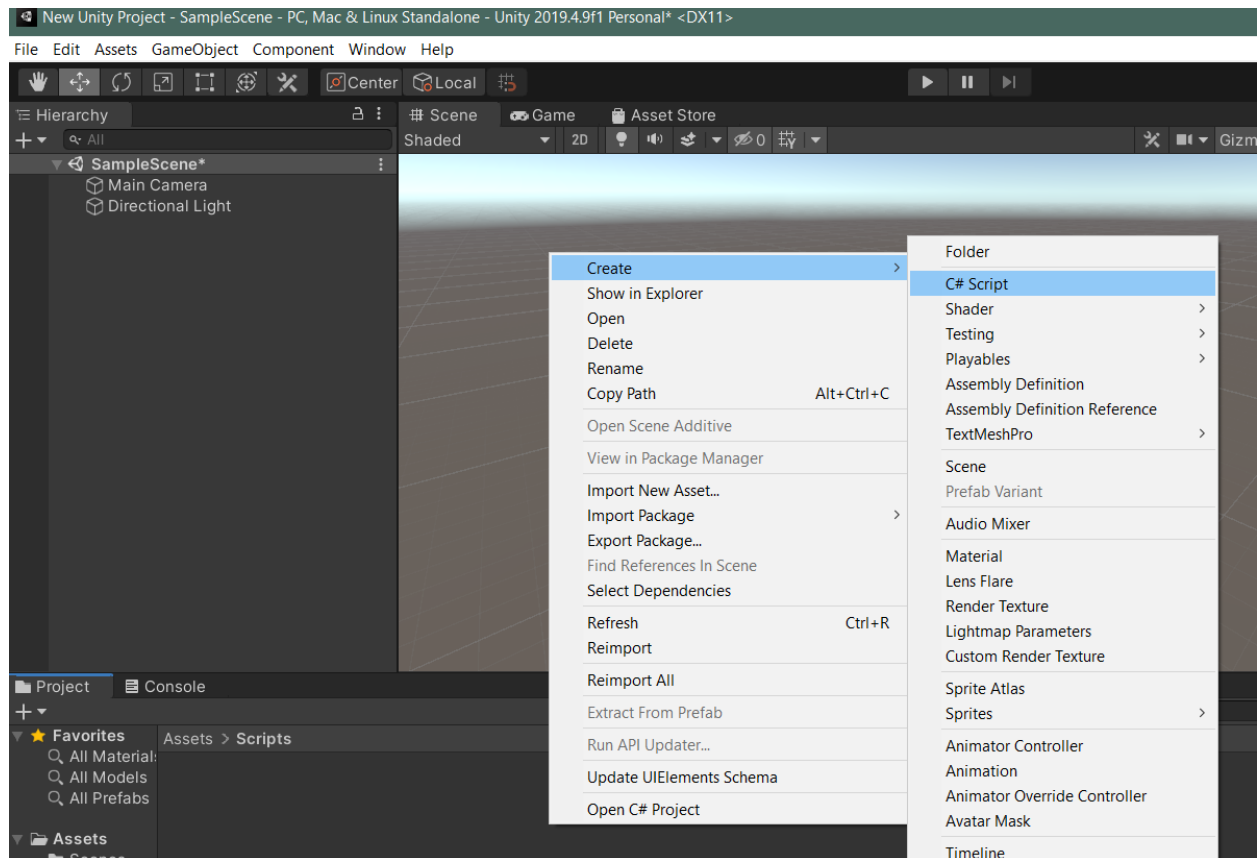
Everything that runs in your unity game exists in a scene. When you package your game for a platform, the resulting game is a collection of one or more scenes, plus any platform--dependent code you add. You can have as many scenes as you want in a project. A scene can be thought of as a level in a game, though you can have multiple levels in one scene file by just moving the player/camera to different points in the scene.

Add a new 3D project in Unity



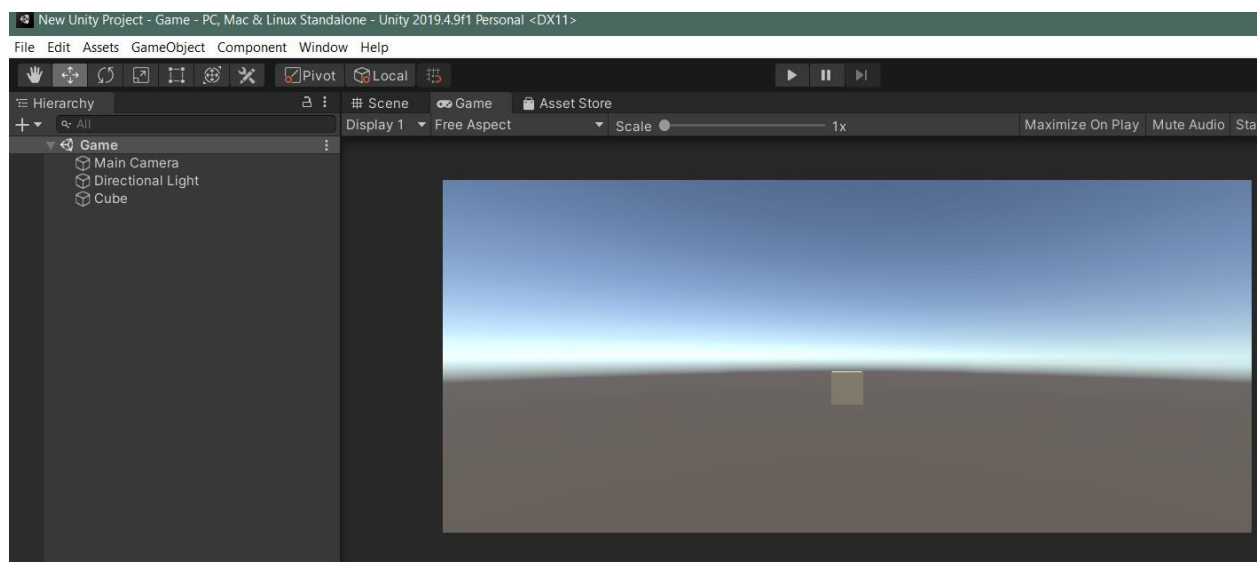
In the new project window Create a folder Scripts and open it.

Inside the opened folder right click and select create and choose "C# Script" from the list.

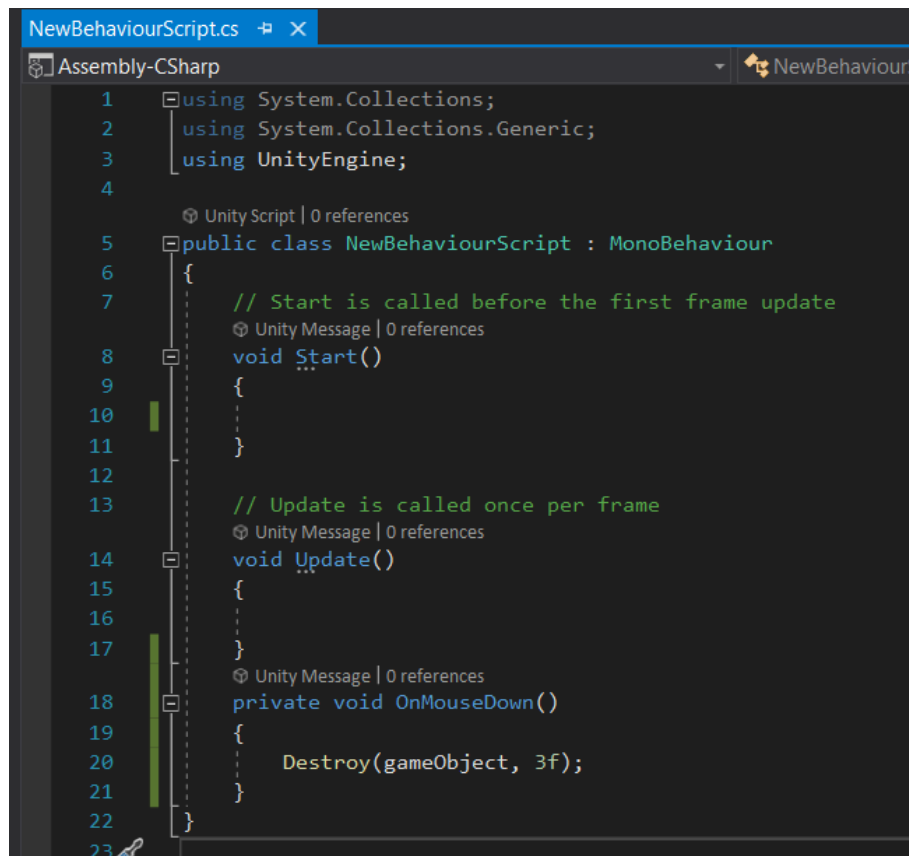


Double click on the newly created script, then it will be opened in the Visual Studio.

Add a new game object cube in the Unity project.



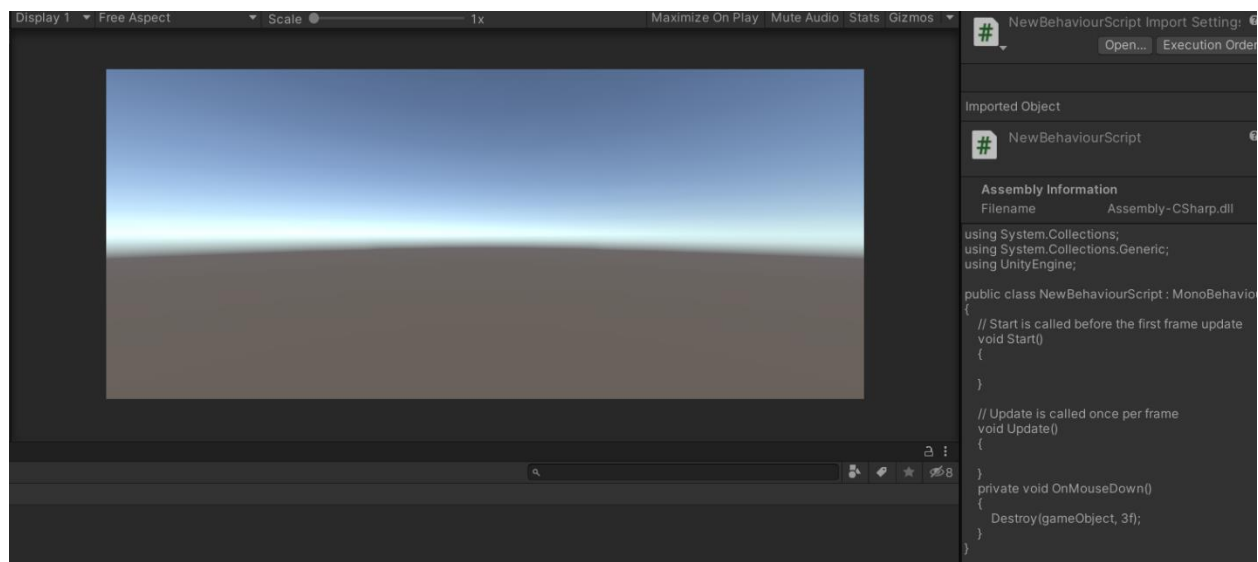
Add code in the C# script to hide the cube after 3seconds of mouse click on the cube.



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 [Unity Script | 0 references]
6 public class NewBehaviourScript : MonoBehaviour
7 {
8     // Start is called before the first frame update
9     [Unity Message | 0 references]
10    void Start()
11    {
12    }
13
14    // Update is called once per frame
15    [Unity Message | 0 references]
16    void Update()
17    {
18    }
19
20    [Unity Message | 0 references]
21    private void OnMouseDown()
22    {
23        Destroy(gameObject, 3f);
24    }
25 }
```

Then play the project and click on the cube.

After 3 seconds, the cube will be removed.



This is a basic sample C# script run on unity to test the integration of visual studio and unity.