Graphic interface setup

Before we look at how to use Canvas buttons in Unity let's create the minimum elements needed to be able to solve the problem.

Canvas

The Canvas is the object in which we can place elements belonging to the graphic interface. To create a Canvas we right click on the hierarchy, go to UI and then click on Canvas, as shown in figure 1.

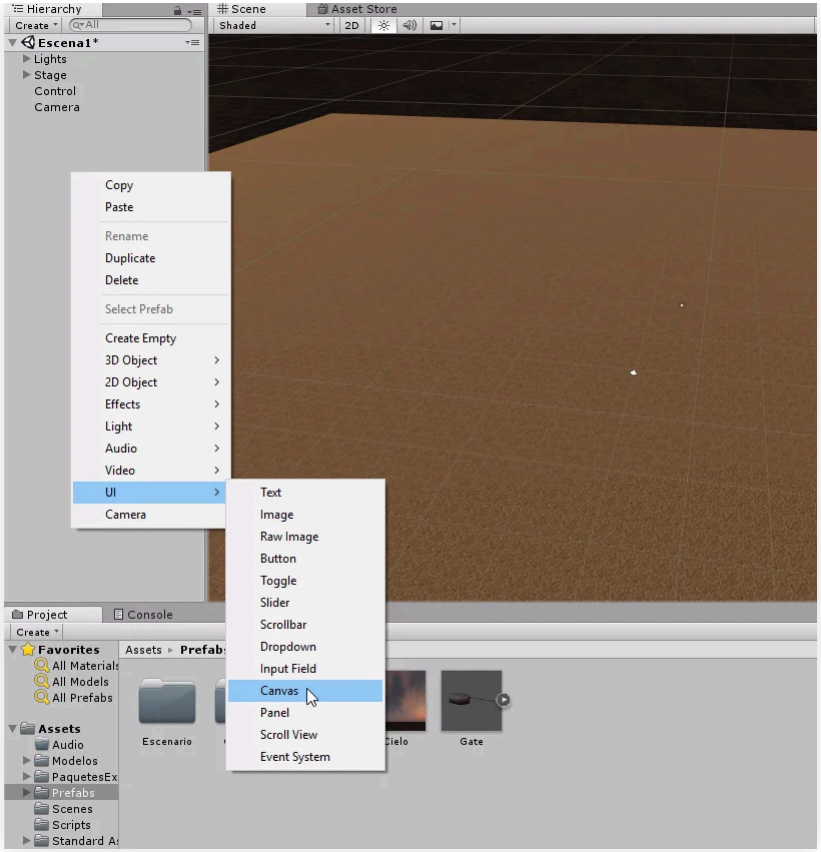

Fig. 1: Creating a Canvas GameObject in Unity

Before going any further, I recommend to configure the Canvas climbing component according to your needs, this component is assigned to the Canvas and you can see it in the inspector.

In my case I will set the “UI Scale Mode” parameter to “Scale With Screen Size” and indicate a reference resolution of 1920×1080, as shown in figures 2 and 3.

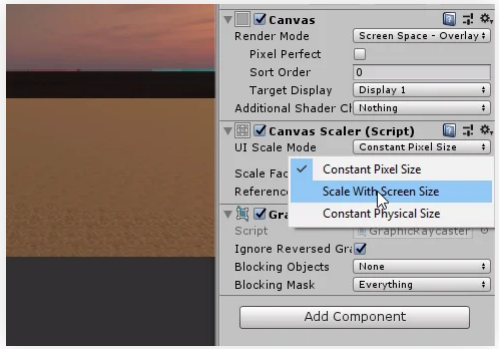

Fig. 2: Canvas Scaler component of a Canvas type object

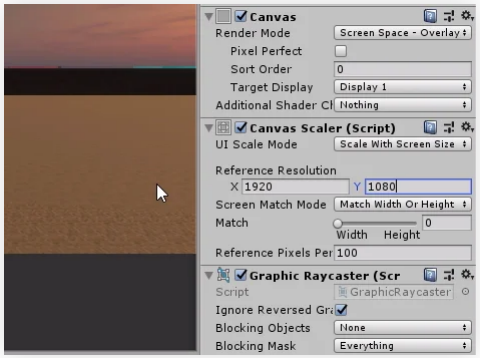

Fig. 3: Reference resolution for the Canvas climber

Button Object

Now inside the Canvas we create a Button. In the hierarchy, right click on the Canvas object, go to UI and choose the “Button” option, this will add a Button type object to the Canvas.

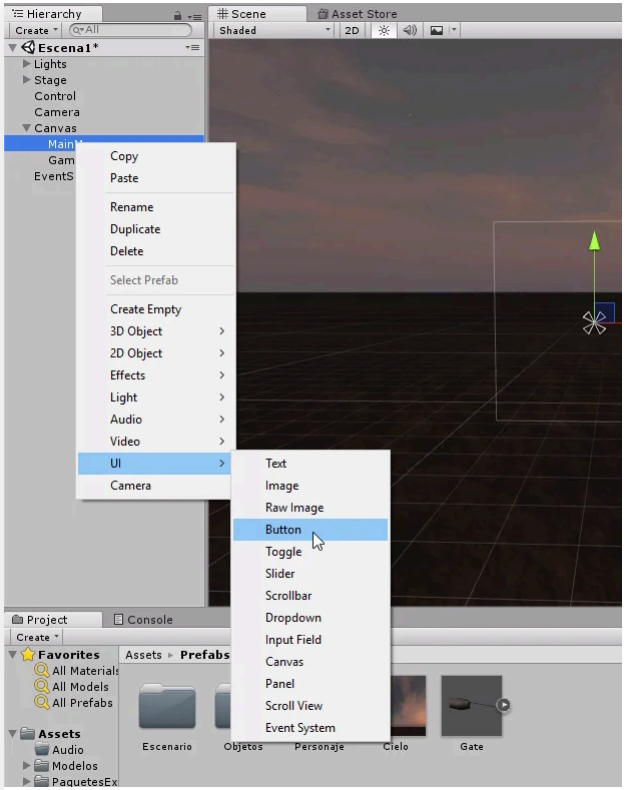

Fig. 4: Creation of a button object in Unity

Then I will create a panel that will have, as a child, a Text type object, both can be created from the same menu as shown in figure 4.

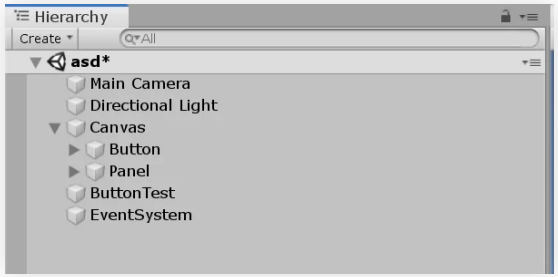

Fig. 5: The Canvas contains a button and a message. An empty GameObject is created to assign the script with the action.

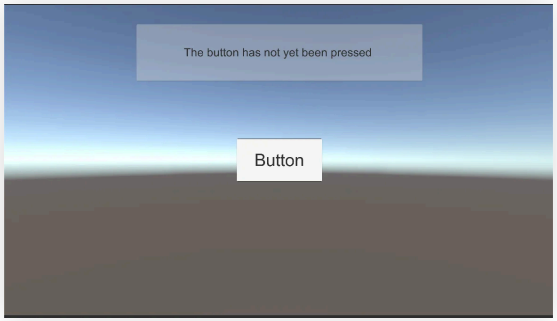

Fig. 6: Elements of the graphic interface

Create an action for the button

As an example, the action will simply consist of making a sign appear saying that the button has been pressed.

For the button to be able to perform any action when pressed, we need to define it in a Script, so let's create one by right clicking on the project folder, then go to Create and choose the option C# Script, as shown in figure 5.

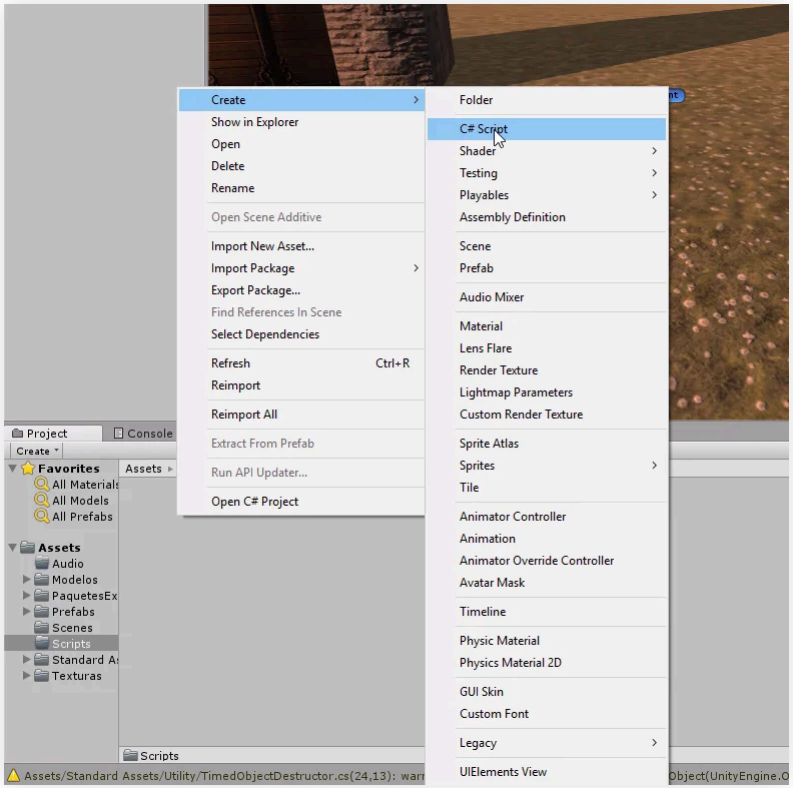

Fig. 7: Creating a new script in Unity

We give it a name and then create an empty GameObject and assign the Script to it. In figure 5 you can see that I have created an empty GameObject called “ButtonTest”.

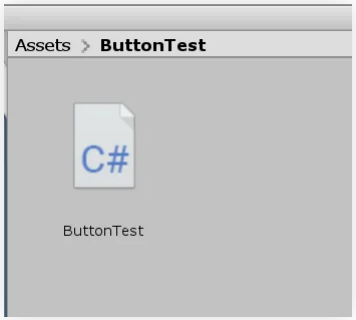

Fig. 8: We created a Script called ButtonTest to program the action of the button when pressed.

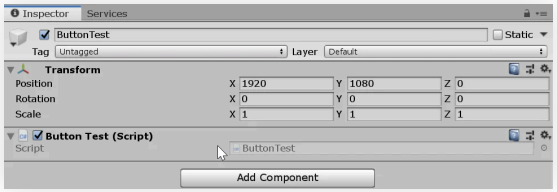

Fig. 9: Asignamos el Script al GameObject ButtonTest.

To describe all the actions that a button must do when pressed we create a method or public function within the Script, in figure 10 we see the function called “MyButtonFunction”.

It is VERY IMPORTANT that this function is declared as public, otherwise we cannot assign it to the button.

The function is responsible for incrementing a counter called “buttonPressedCounter” and then writing a message on the screen indicating how many times the button was pressed. The if I had to add it because when the counter is 1, the word “time” goes in singular.

You can try this same example for, remember to add the namespace that is in line 4, “UnityEngine.UI”, otherwise you will not be able to access the Text class.

Assembly-CSharp ButtonTest

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class ButtonTest : MonoBehaviour
7  {
8
9      [SerializeField]
10     private Text message;
11
12     private int buttonPressedCounter = 0;
13
14
15     // Start is called before the first frame update
16     void Start()
17     {
18
19     }
20
21     // Update is called once per frame
22     void Update()
23     {
24
25     }
26
27     public void MyButtonFunction()
28     {
29         buttonPressedCounter++;
30         if (buttonPressedCounter == 1)
31         {
32             message.text = "The button has been pressed "+buttonPressedCounter+" time.";
33         }
34         else
35         {
36             message.text = "The button has been pressed " + buttonPressedCounter + " times.";
37         }
38
39
40     }
41
42
43 }
44
```

Fig. 10: Code within the ButtonTest Script.

We go to the inspector and place the Canvas Text component in the script field.

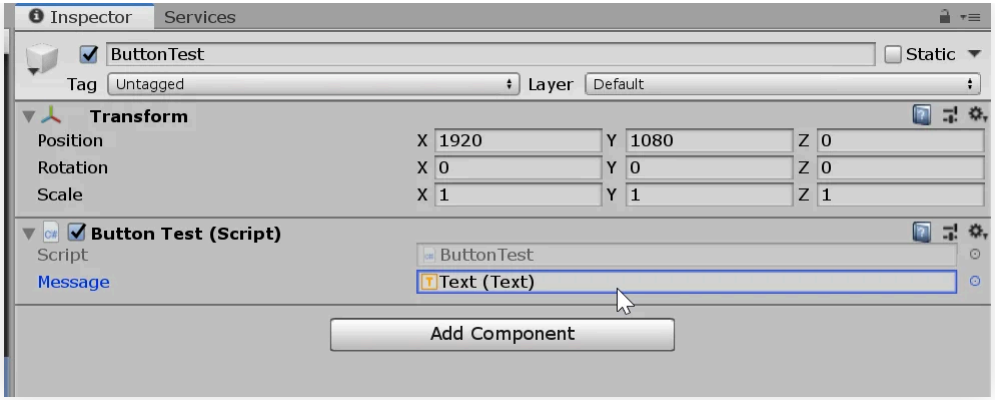

Fig. 11: I assign the text requested by the script to the inspector.

Now we select the button and go to the Button component in the inspector, here we must click on the + button in the OnClick() panel, this will add a new action when an OnClick() event occurs, that is when the button is pressed.

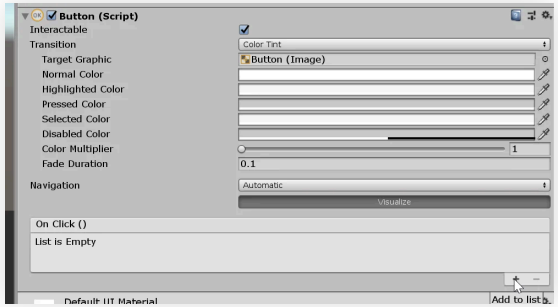

Fig. 12: In the OnClick panel we click on the + button to add a new action.

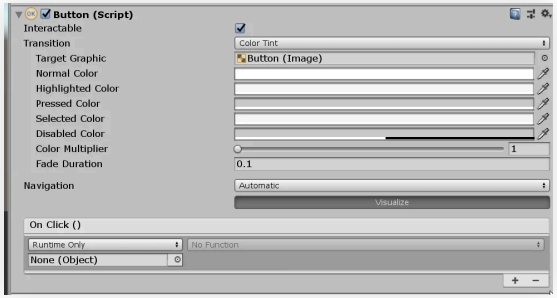

Fig. 13: A field appears in which we can assign a GameObject.

As we see in figure 13, we have a field where we can assign a GameObject, here we drag from the hierarchy, the GameObject that has assigned the script with the action. As we see in figure 14, we have dragged the GameObject “ButtonTest”.

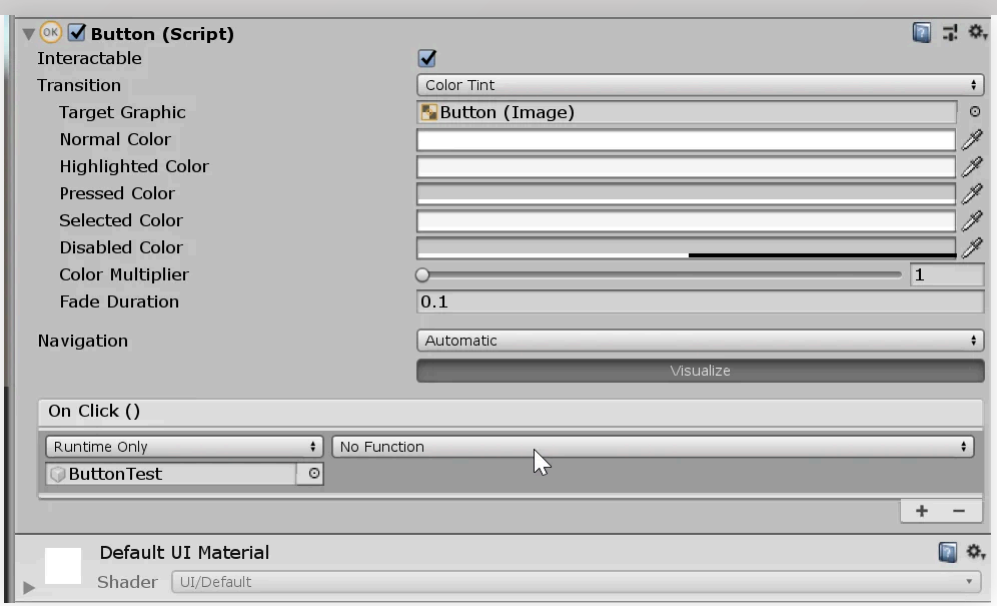

Fig. 14: We assign the GameObject containing the Script with the action we want the button to perform when pressed.

Then using the drop-down menu, which initially says “No Function”, we’ll first select the Script and then the public function we’ve created for the button, as shown in Figure 15.

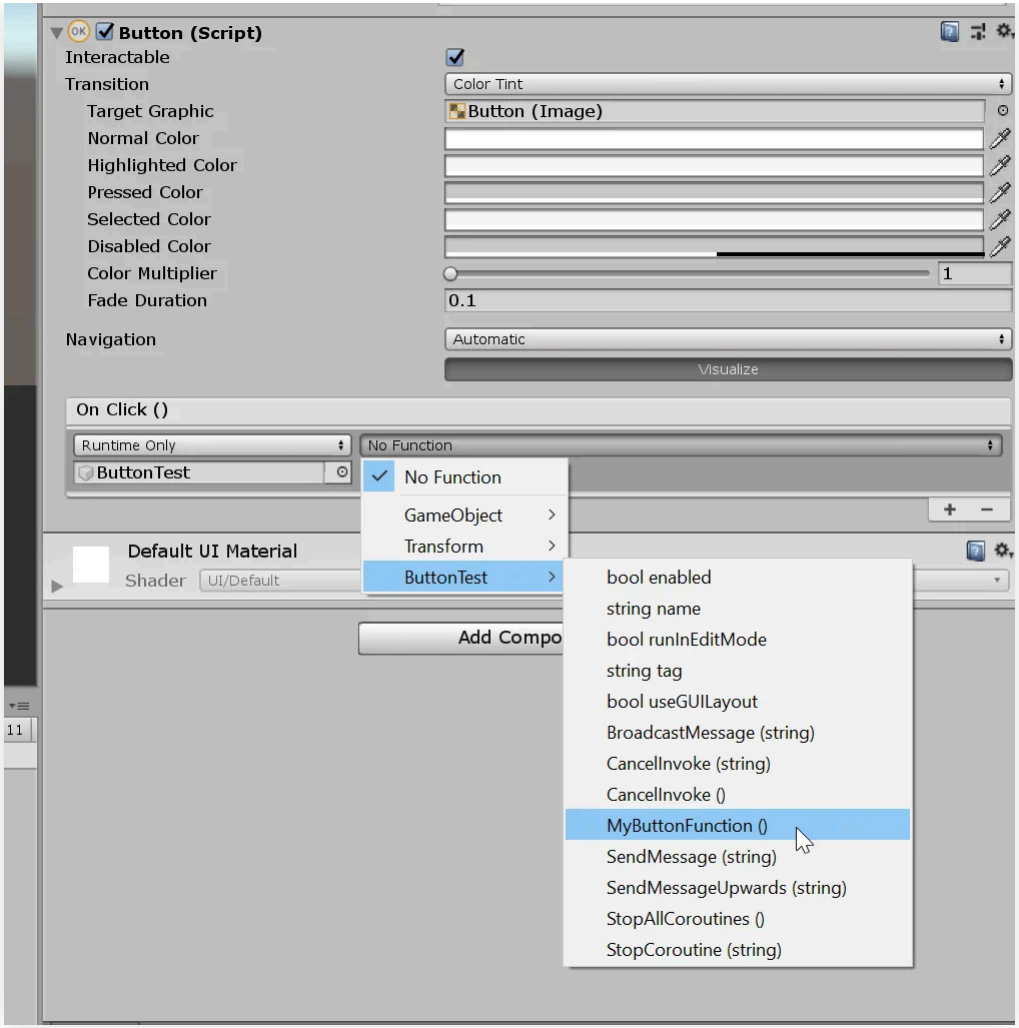

Fig. 15: Using the drop-down menu, we choose the function that we want to be executed when the button is pressed.

Each time the button is pressed the poster changes showing the number of times it has been pressed.

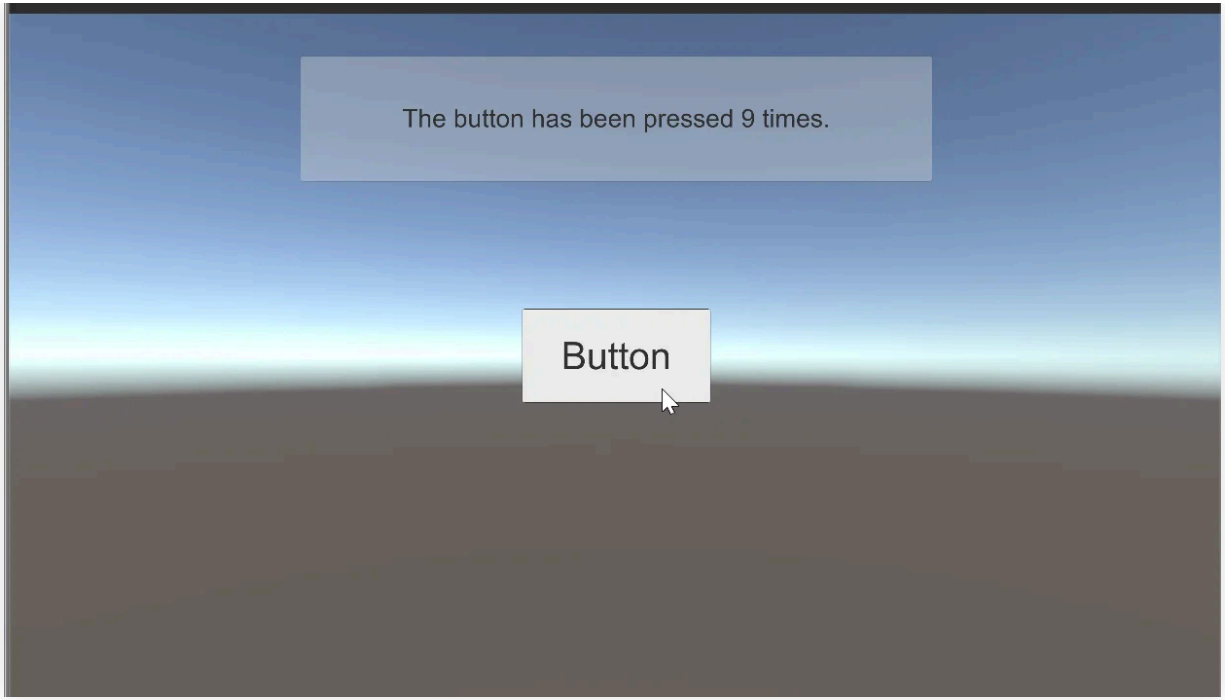

Fig. 16: Every time the button is pressed the counter increases and the display changes.

Conclusion

We've seen how to use the Canvas buttons in unity. The action must be programmed within a script using a programming method and must be declared as public.

Then we must add a new field in the OnClick() panel of the Button component and assign the GameObject that the script has with the action.

Finally, using the drop-down menu in OnClick(), we must select the function we have defined in the script.

← PREVIOUS
What is a MonoBehaviour in Unity

NEXT →
Buttons not working in Unity – 3 common reasons