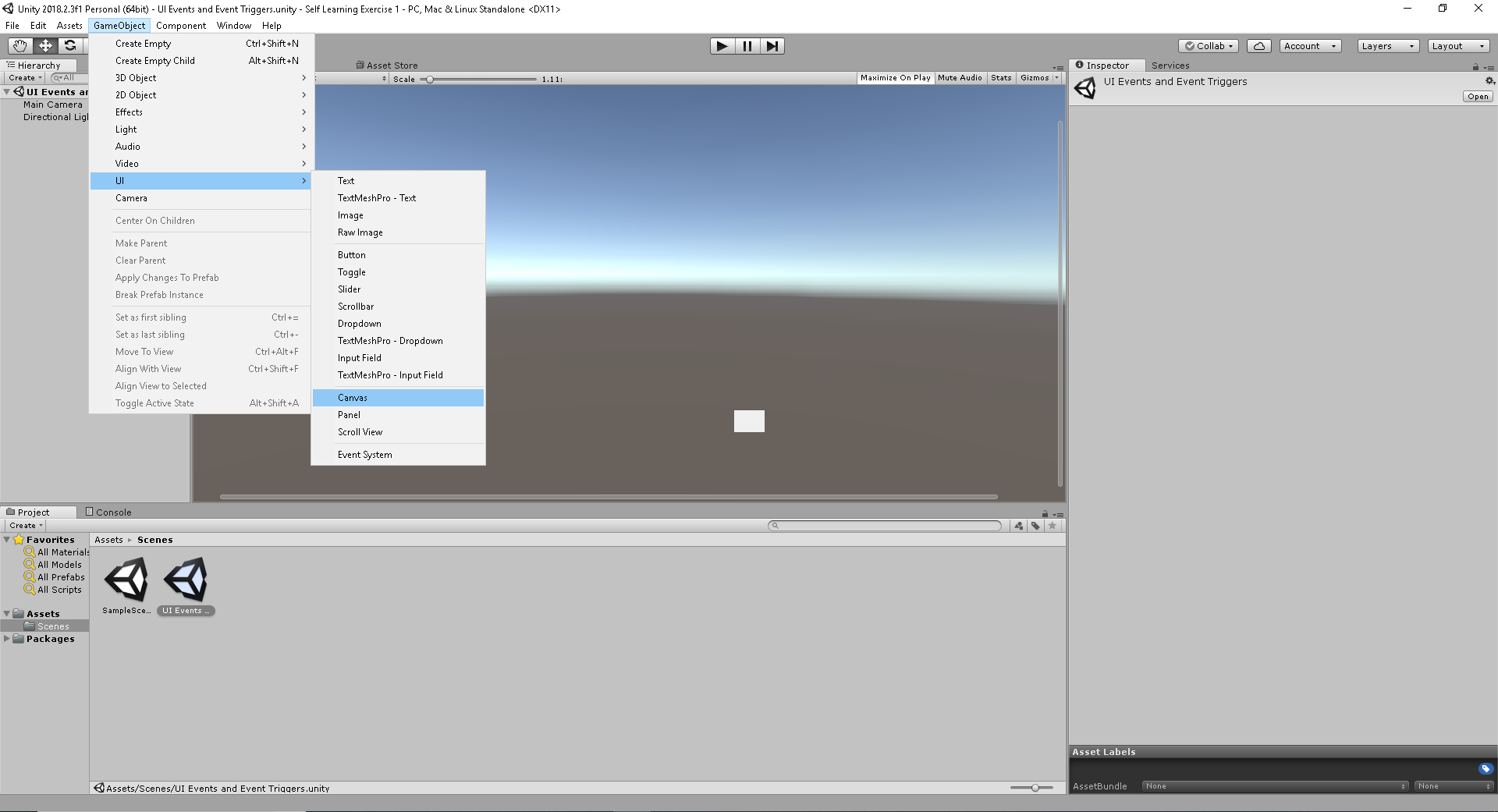
UI Events and Event Triggers

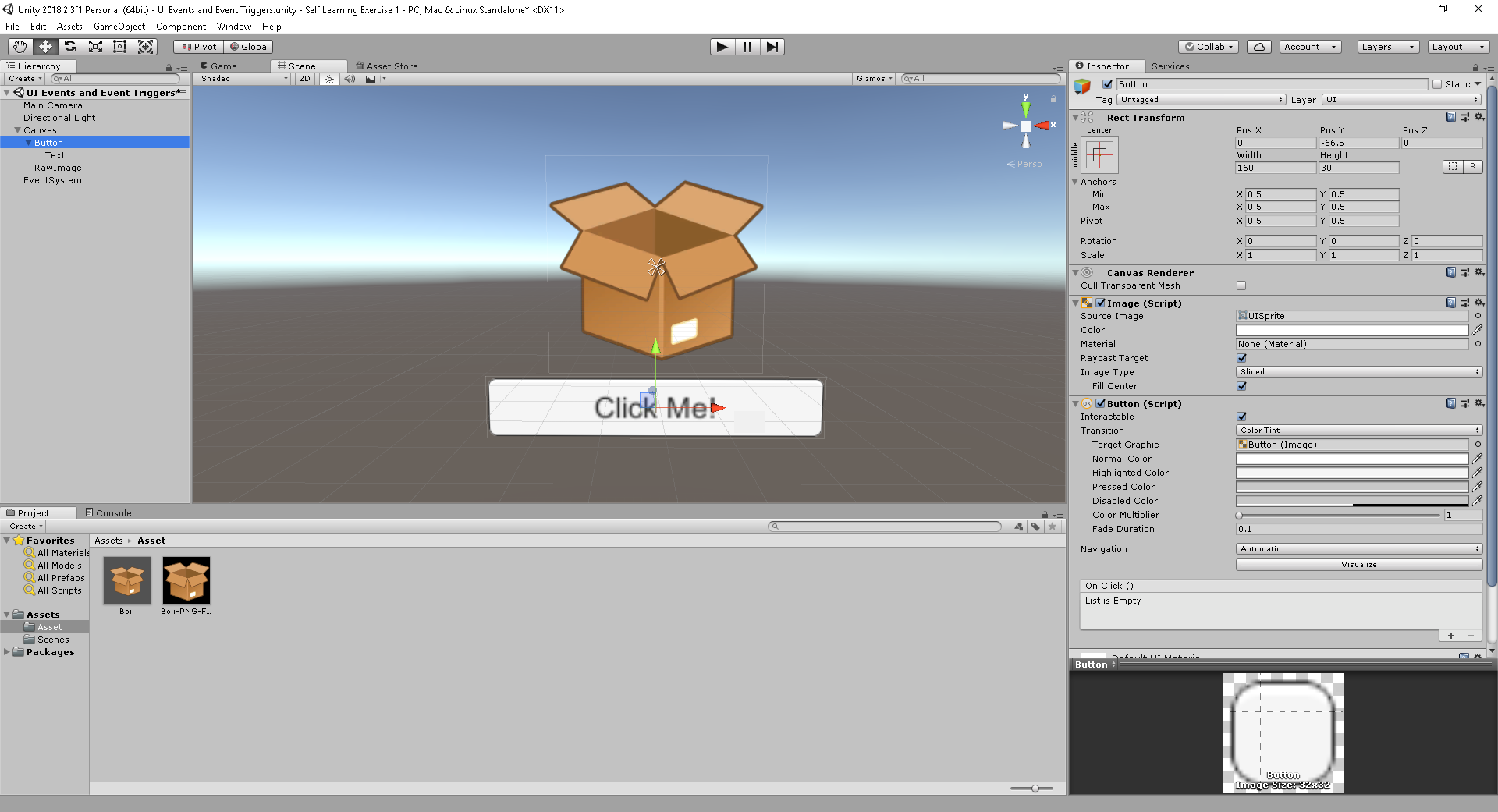
Whenever dealing with UI elements, the user tends to want the UI to be interactable. This lesson will cover the how to set up and create the events.

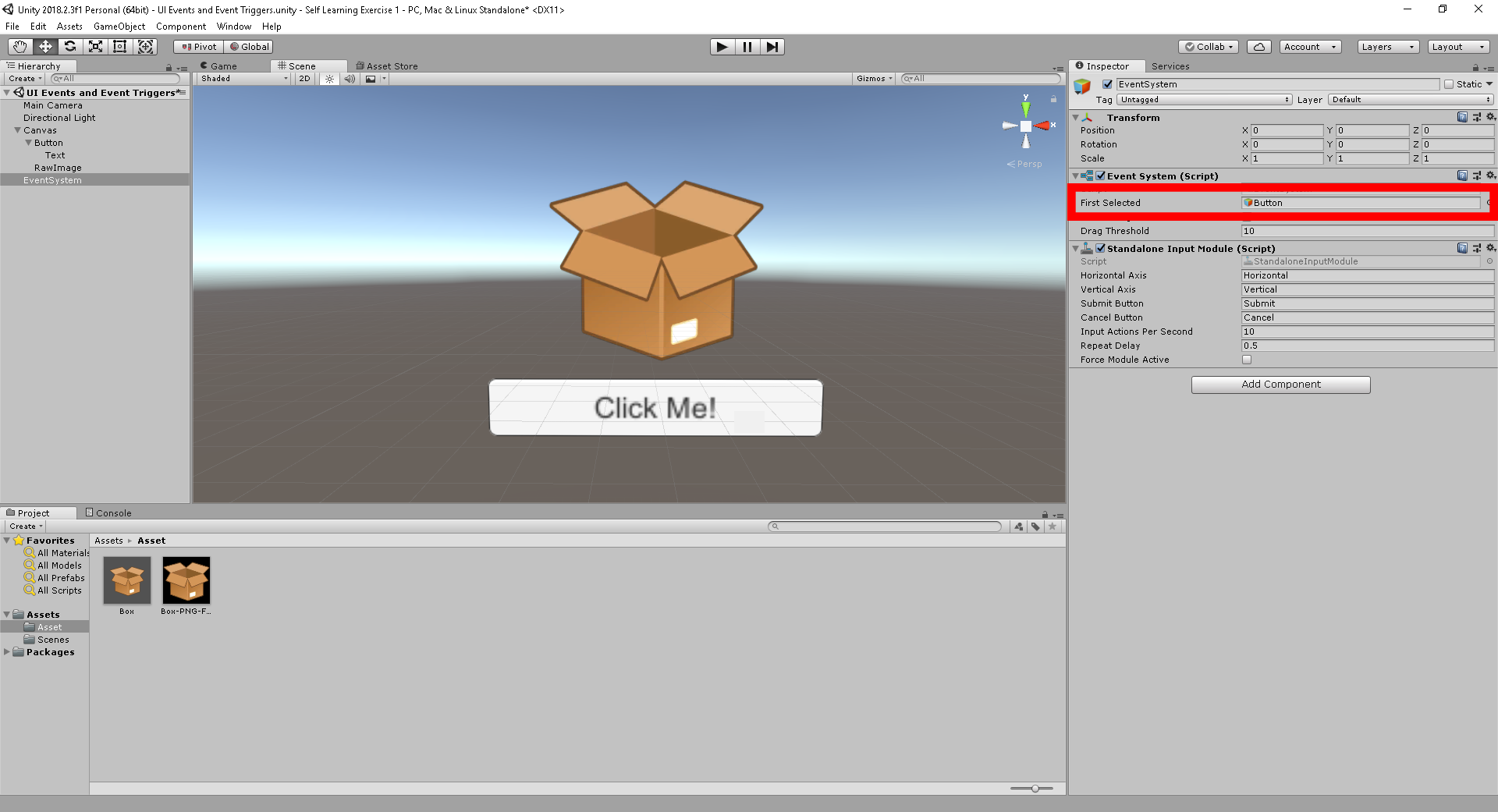
Start by creating a Canvas in the scene.



An EventSystem will automatically be created when a Canvas is created. Right Click on the Canvas go to UI and Button to create a button within the UI. The EventSystem looks for input modules and determines how each UI element should react.

To demonstrate this click on the EventSystem and look for a property in the Event System called **First Selected**. Any interactable UI elements can be selected and whatever elements is applied to this will be selected by default. Drag the Button element into the **First Selected** property and run Unity.





When you run the scene, you can see that the button is still interactive.

With the Event System there is also an Input Module. This is used for mouse, keyboard or gamepad. The Standalone Input Module is used to refer to different input axis that are used to interact with different UI elements. The Event System and Input Module determine what events are sent or fired to different game objects.

We will now create an event that will change the color of the box to demonstrate how this would work.

First create a script called BoxColor and apply this code snippet in the Update function.

using UnityEngine.UI;

public class BoxColor : MonoBehaviour {

Button button;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

}

void TurnRed()

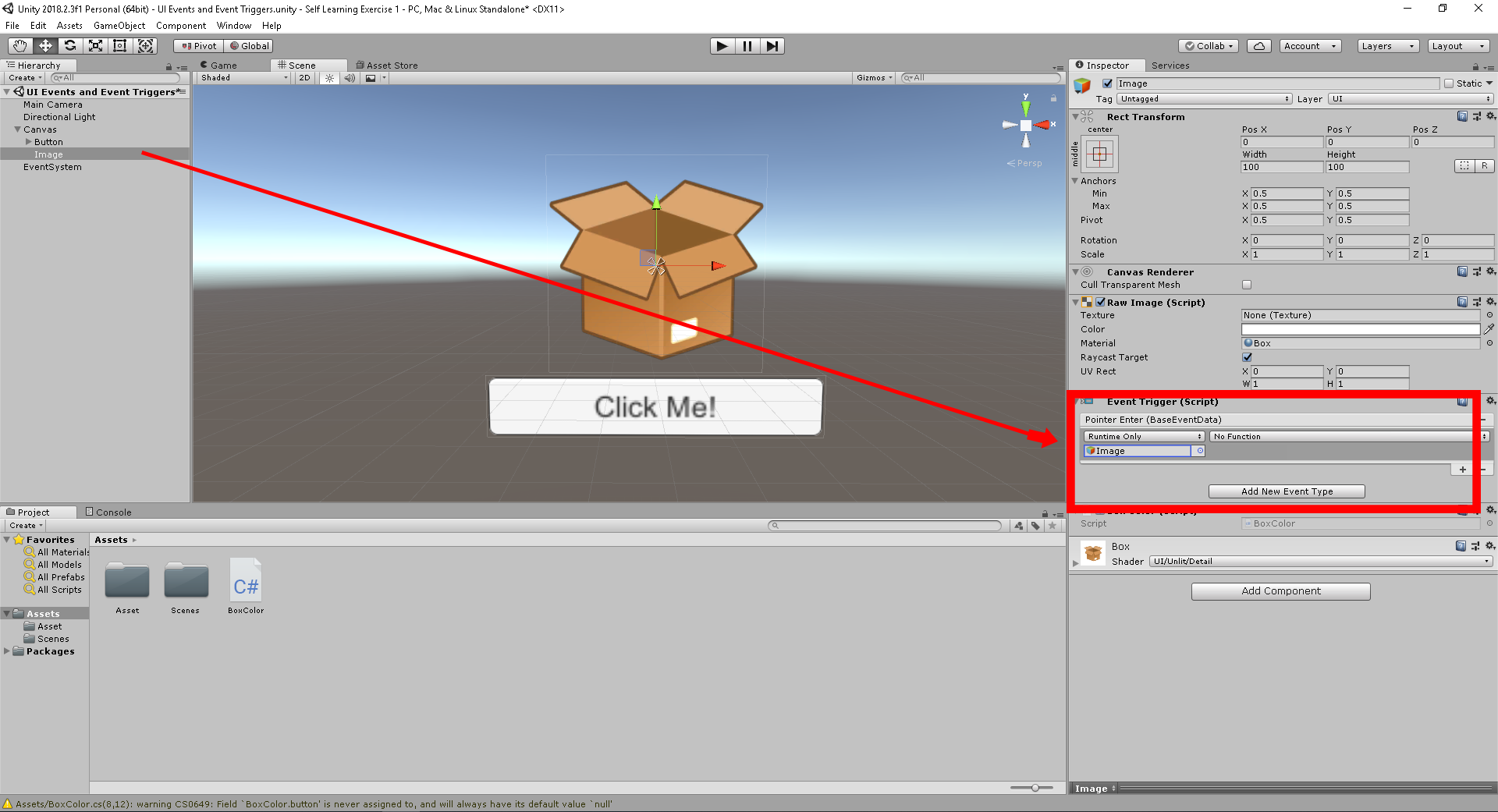
{

button.image.color = Color.red;

}

}

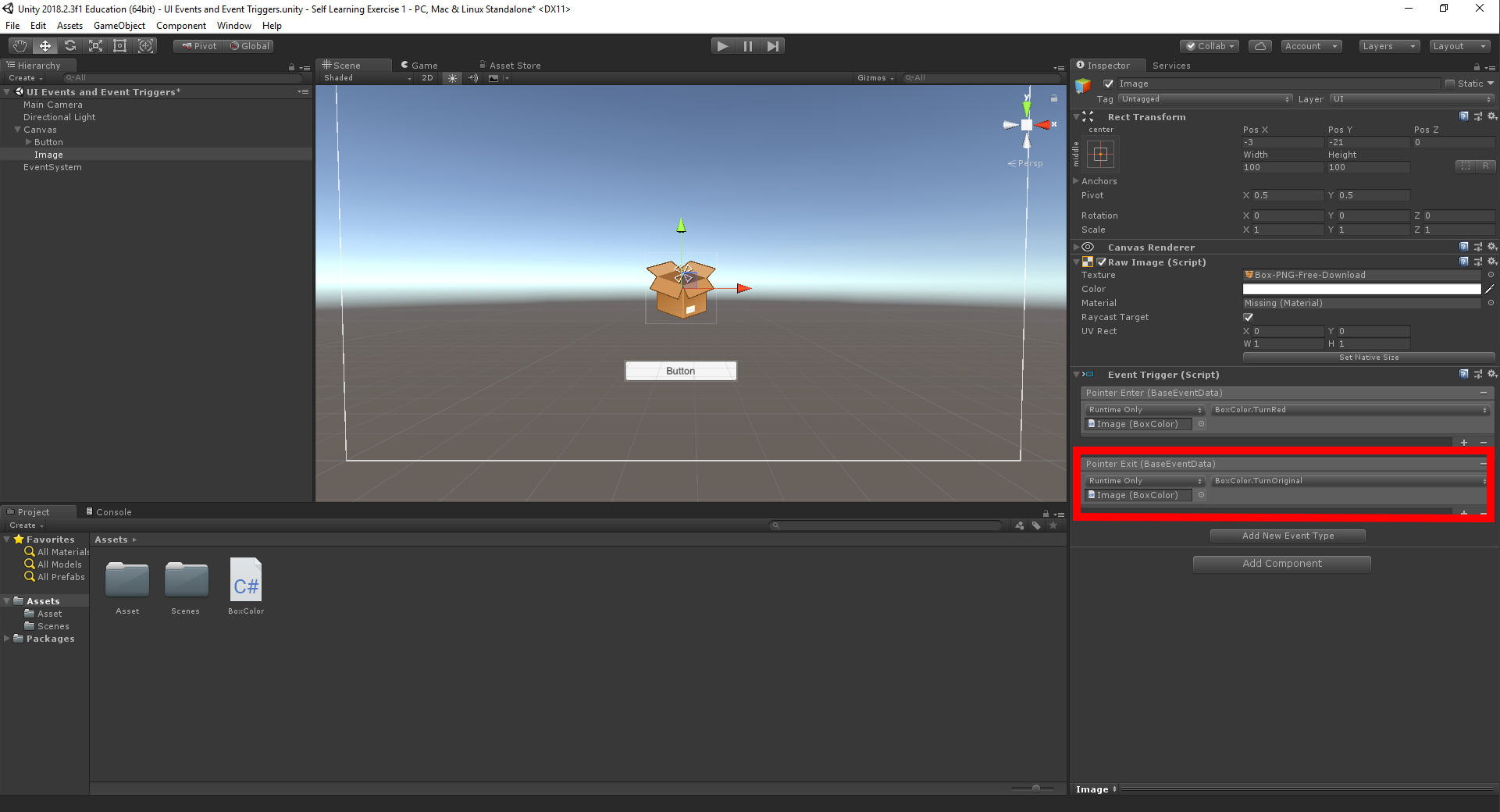
We will be using an OnClickEvent that is called when a button is pressed. Click on the **Image** within the Canvas, click on Add Component, Event and then Event Trigger. To add a new event click Add New and select **PointerEnter**. Now Add a function to the list by clicking on the + icon. Drag the Image into the object property.



Click on No Function and go to Box Color and click on the TurnRed() method. Click play and hover over the box. The color should now change to red.

To change the color back to normal once the user stops hovering over the box, click on the **Image** within the Canvas, click on **Add New Event Trigger**. To add a new event click add New and select **PointerExit**. Now Add a function to the list by clicking on the + icon. Drag the Image into the object property.

Click on No Function and go to Box Color and click on the TurnOriginal() method. Click play and hover over the box. The color should now change to red. Move the cursor away from the box and the box should go back to its normal state.



UI Slider

The Slider is a UI element that is controlled based on a min and a max value. The numbers are predetermined and controlled by dragging a handle for the slider. These controls are typically used for health bars, brightness, or sizing elements.

To start create a text box in the Canvas and name it ScaleText. Right Click the Canvas go to UI and then Text. Position the ScaleText as such: Pos X: 7 Pos Y: 141 Pos Z: 0 Width: 160 Height: 30. In the Text field name the Text: “**Scale the box:**” Set the Font: Arital, Font Size: 14 and Paragraph Alignment centre.

Next create a Slider. Right Click the Canvas go to UI and then Slider. Position the Slider as such: Pos X: 1 Pos Y: 119 Pos Z: 0 Width: 160 Height: 20. We are going to scale the Box object in game to scale big and small depending on the slider control.

Next let’s create a script that will build onto this functionality. Name the Script BoxSlider and apply this code into the Script.

using UnityEngine.UI;

public class BoxSlider : MonoBehaviour

{

public Slider slider;

public void SliderScale()

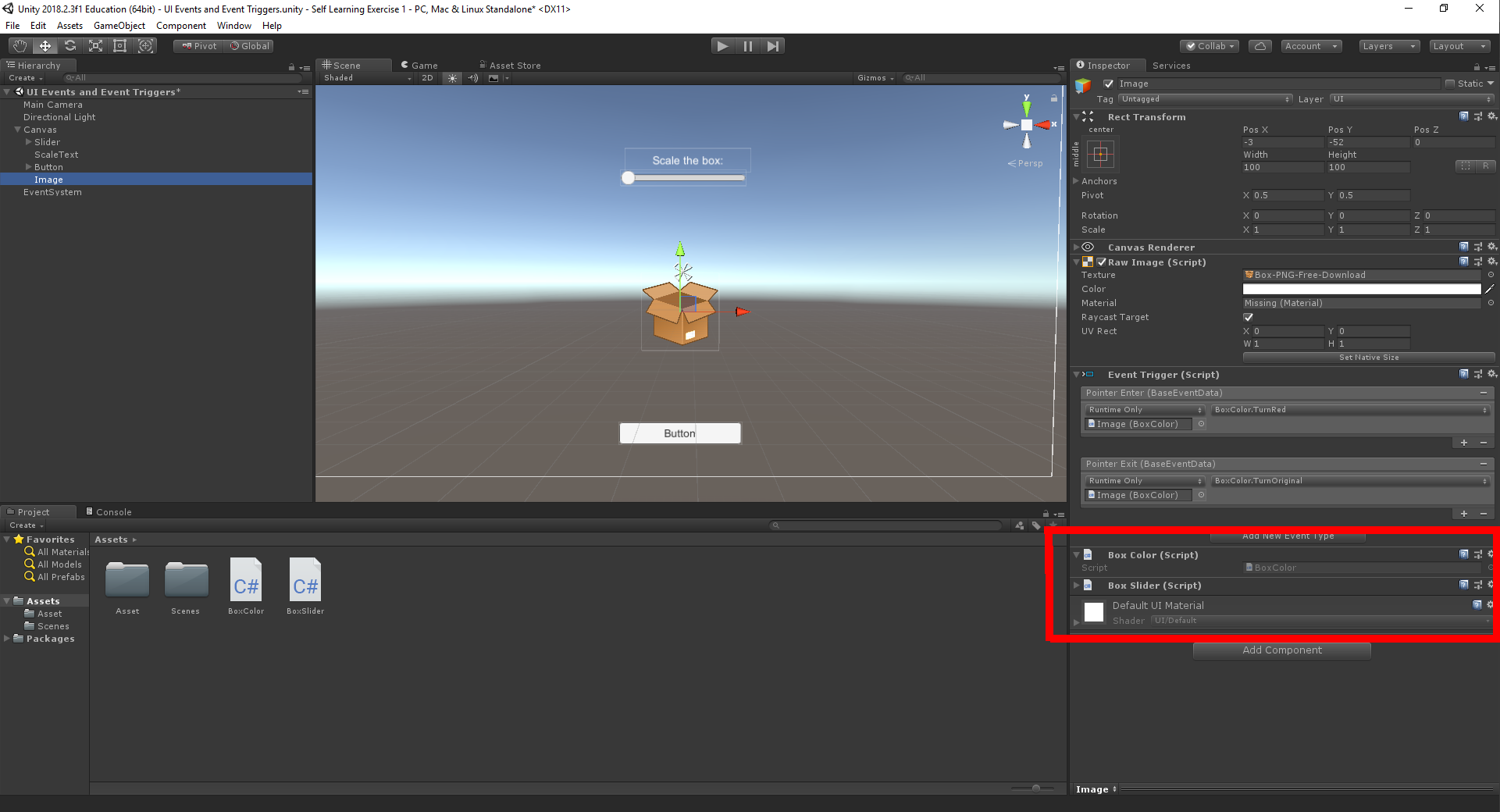
{

transform.localScale = new Vector3(slider.value, slider.value, slider.value);

}

}

Save the file and drag the script onto the Box Image. You’re box Image should now have two scripts.



Click on Slider and change the Min Value to 1 and Max Value to 2. Click the + sign on the On Value Changed (Single) and drag the Image component into the None (Object) spot. Click on the No Function and go to the BoxSlider and select the SliderScale() method.

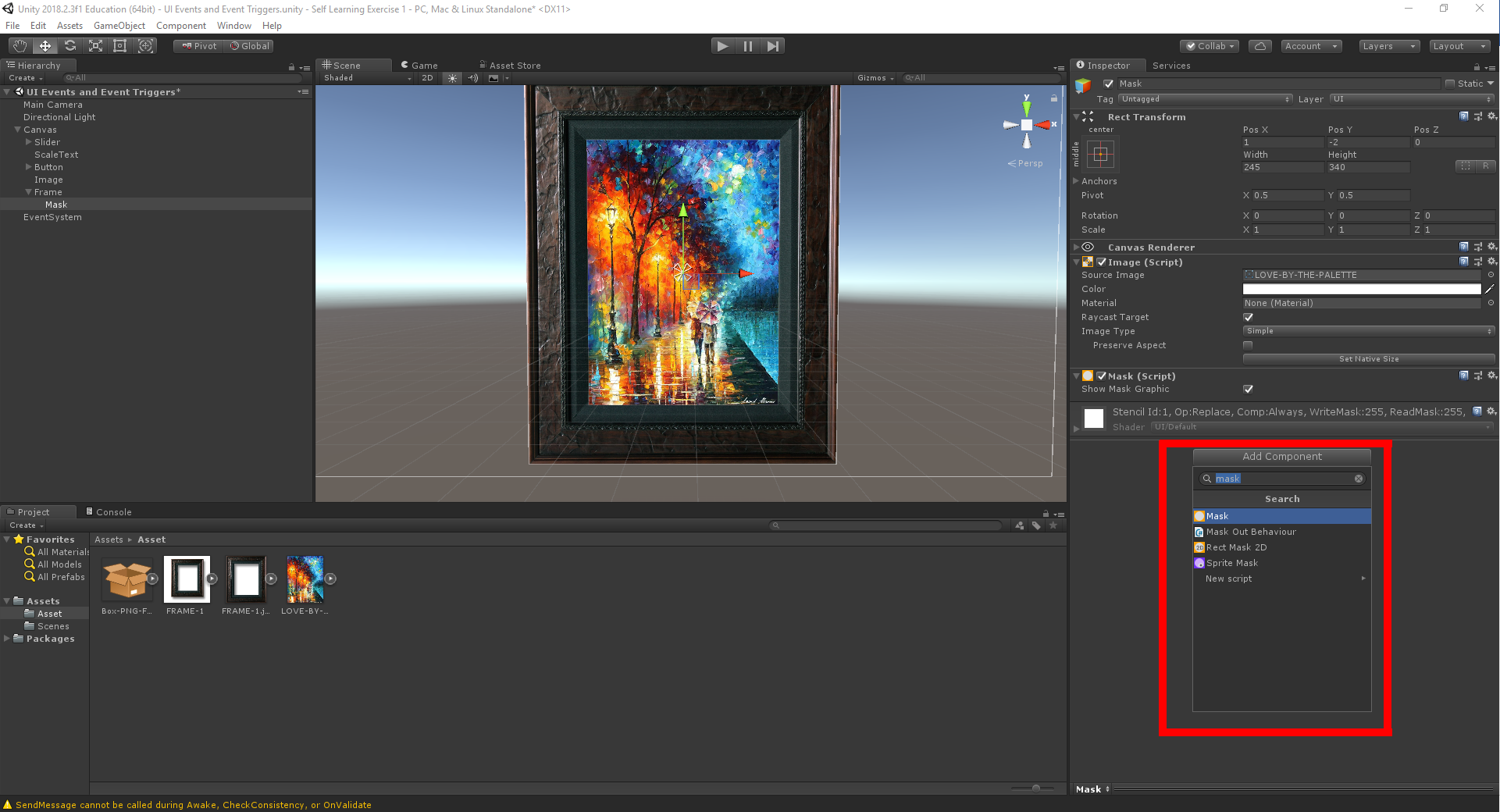
Click play and you can see that you’re box can change in size due to the slider component.

UI Mask

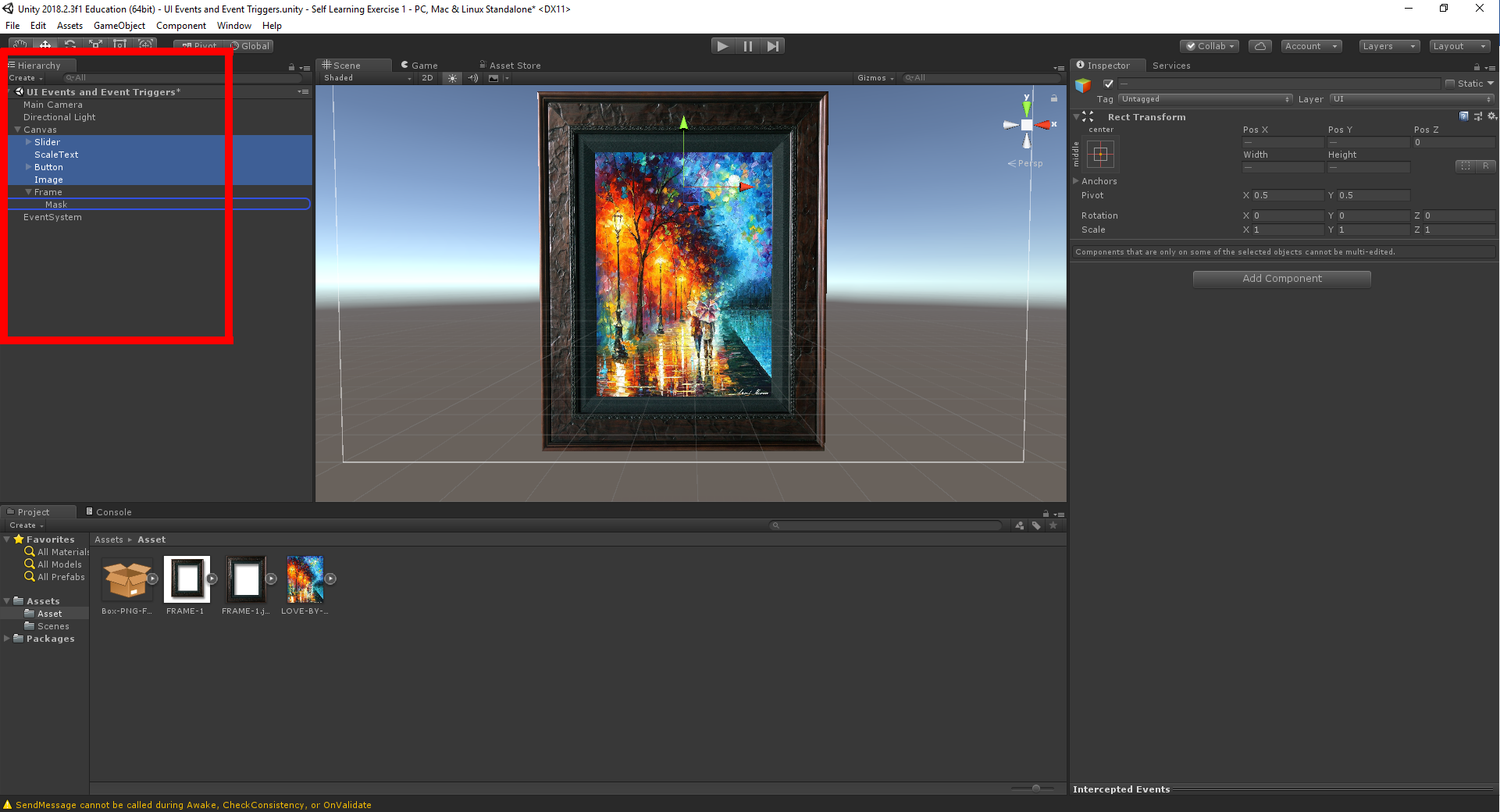
A UI Mask is used to create a shaped a mask over a UI element. A mask element would be used to clip a sized content to a desire size.

To start, first right click on the canvas and go to UI and click on Image. Rename the Image to Frame and place a frame picture for the Image. Position the Frame as such: Pos X: 0 Pos Y: -1 Pos Z: 0 Width: 400 Height: 500.

Next create another Image within the Frame game object. This is done by right clicking on Frame then go to UI and click on image. Place any art image you prefer in this section and call this Image Mask. Position the Frame as such: Pos X: 1 Pos Y: -2 Pos Z: 0 Width: 245 Height: 340. Next add the Mask (Script) component onto the Mask Game Object. To do this, click on the Add Component and search for Mask.



Now Select the Slider, ScaleText, Button and Image components in the Canvas and drag them into the Mask game object.



What you will see is the box image, the button, slider and slider text to appear in front of the Mask game object. Now if you keep all four objects selected and move left and right of the Scene, you will notice that the objects will disappear the moment it touches the frame.