Soap: 4_ScriptableEvents_Example_Scene

Table of Contents
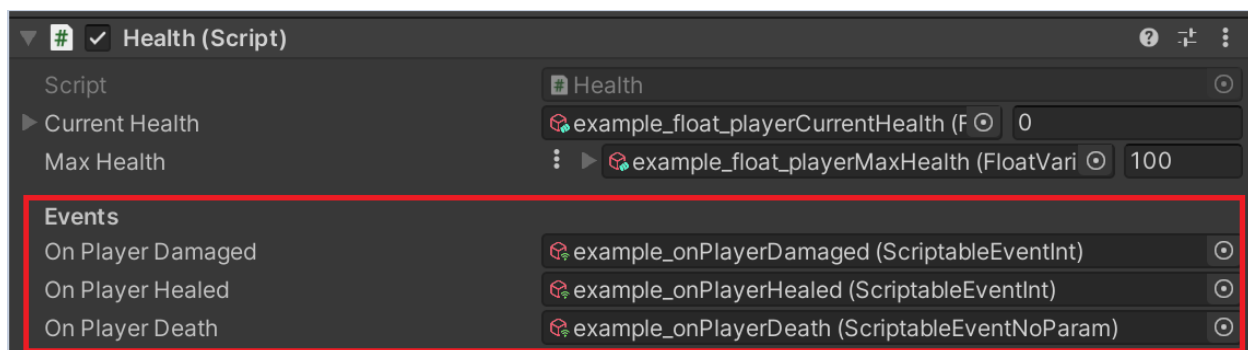
Firing events from Code

Events are nice to trigger things like VFX, menus, or even gameplay elements.

Let's look at the events that are fired from **Health.cs** on the **prefab_player**.

– OnPlayerDamaged -> ScriptableEventInt

– OnPlayerHealed -> ScriptableEventInt

– OnPlayerDeath -> ScriptableEvent (no parameters)

It is simple to fire these events from code, you simply call the **Raise**() method. You can check in **Health.cs** for more details:

```csharp
私 Frequently called    ☑ 1 usage    ☺ Pierre *
private void OnDamaged(float value)
{
    if (IsDead)
        OnDeath();
    else
        _onPlayerDamaged.Raise( param: Mathf.RoundToInt(value));
}

私 Frequently called    ☑ 1 usage    ☺ Pierre *
private void OnHealed(float value)
{
    _onPlayerHealed.Raise( param: Mathf.RoundToInt(value));
}

私 Frequently called    ☑ 1 usage    ☺ Pierre *
private void OnDeath()
{
    _onPlayerDeath.Raise();
}
```

Event Listeners

Health.cs is where those events are fired, but let's find out who is listening to these events. Let's check the **prefab_player** and examine the **DamageVFX**.
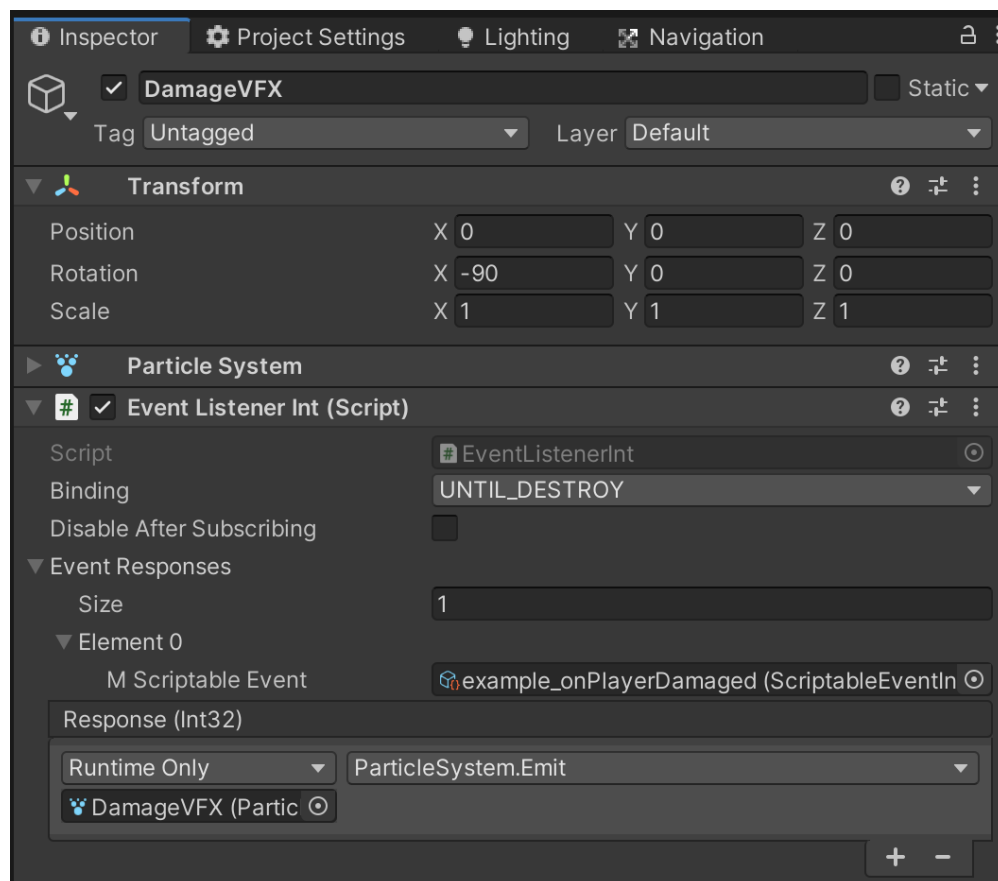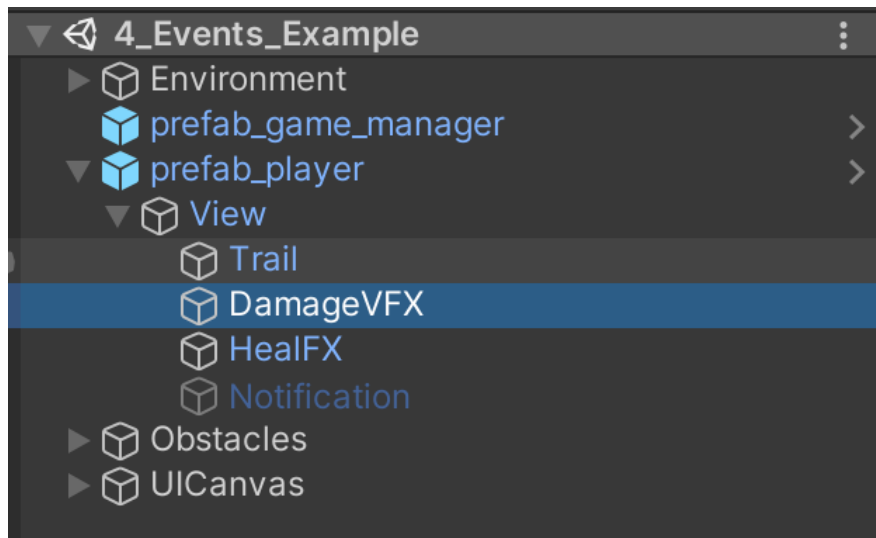
When the player is damaged, this event is fired and it will trigger the particle system **dynamic method** "Emit" that takes an int as a parameters. With this simple trick, we can show visually a relation between the amount of damage and the intensity of the VFX without code. A damage of 10 will emit 10 paritcles, and a damage of 50 , maybe a critical hit, will emit 50 particles.
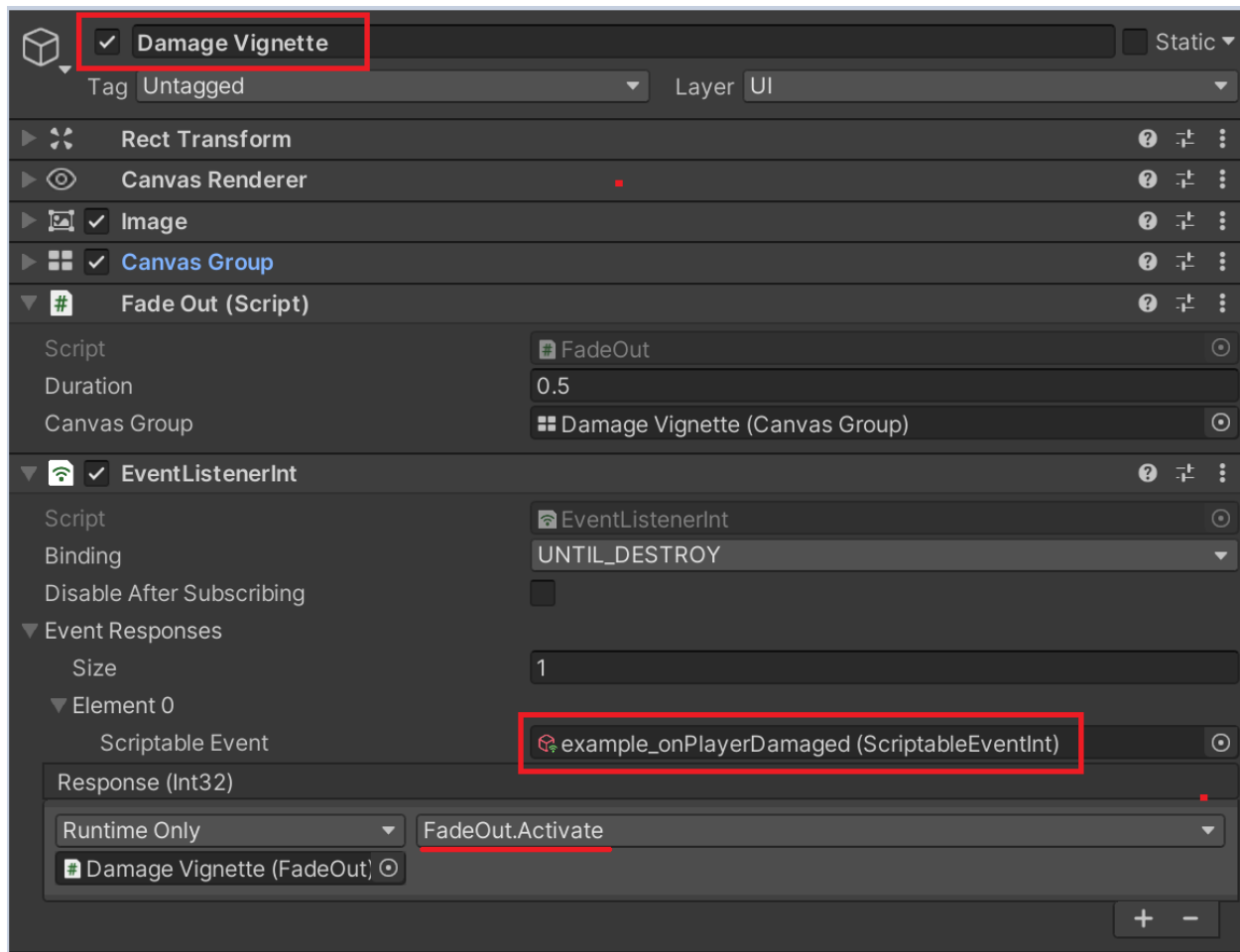
In a bigger game, you would probably have a different VFX for different amount of damages, but in Hypercasual games, you can get away with this.

The Heal VFX in the prefab_player does the opposite but works the same way.

You may have noticed that when the player gets damaged, we trigger a hurt damage vignette effect.

UICanvas/Damage Vignette
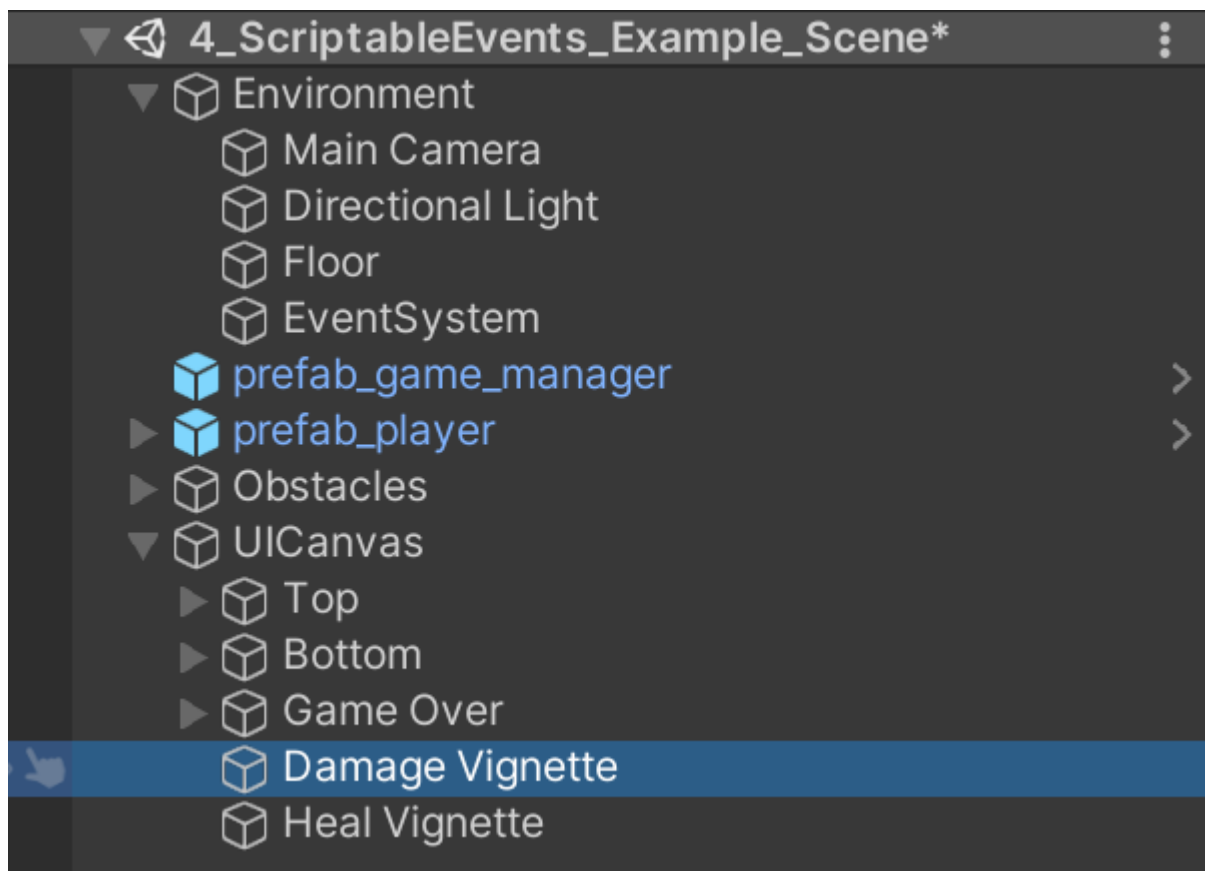
As we can see, here we listen to the **OnPlayerDamaged** event. When it is fired, we activate the Fade out.

Note : we ignore the int parameter as we don't need it and call a regular method instead of a dynamic method through the unity event.

The **example_OnPlayerDeath** event is listened by the Game Over screen.

You will notice that the Game Over Screen is visible in editor and gets disabled when we start the game. This is because we need to have him enabled to register to the event at the beginning of the game.

This what the boolean variable called "Disable after subscribing" is for. The purpose of this variable is to disable the GameObject after subscribing to the event in Awake().
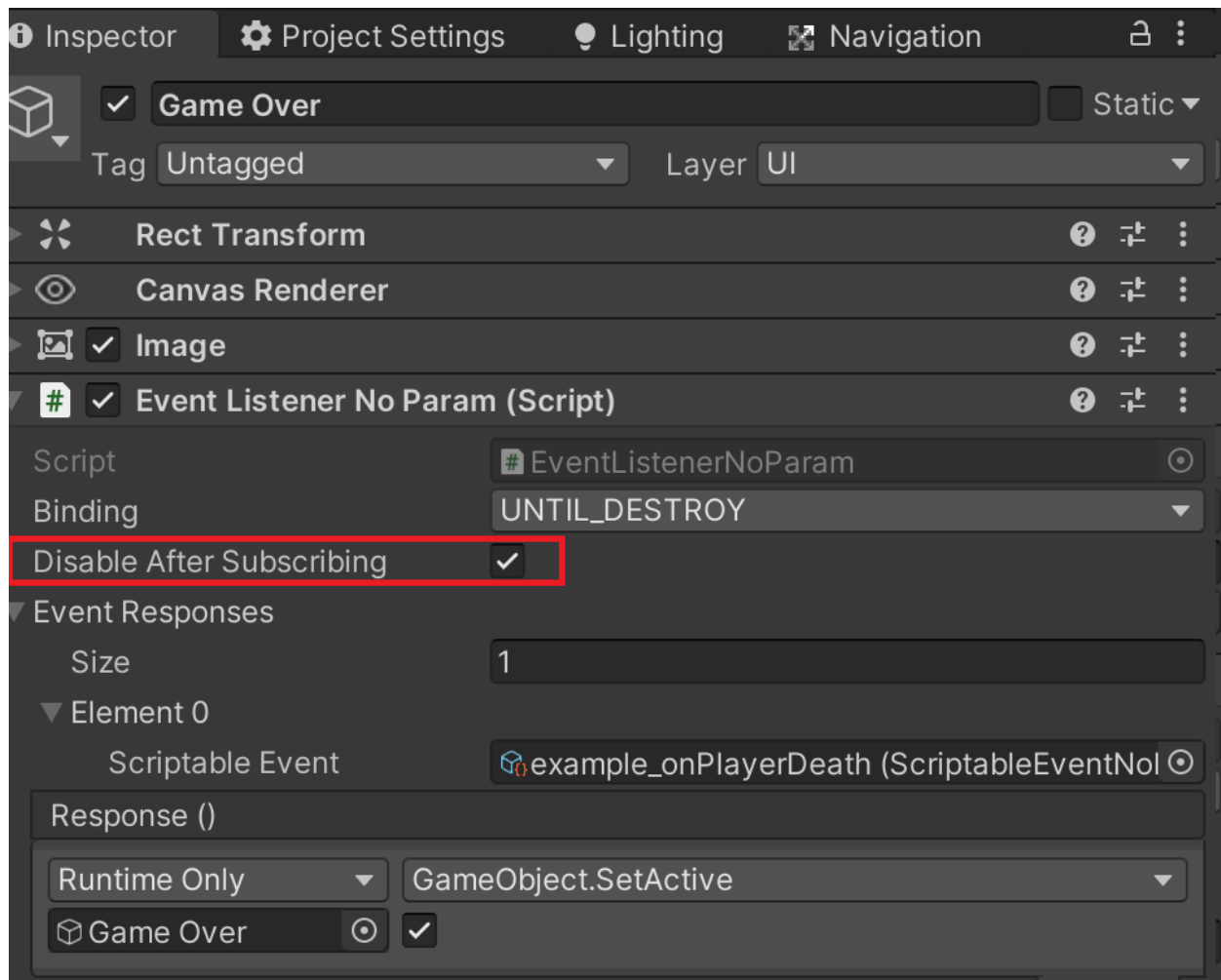
Listening to events from code

Alternatively to using the Event Listeners, you can directly subscribe to an event when it is triggered by code. If you select the **Heal vignette**, we can see an example.

UICanvas/Heal Vignette
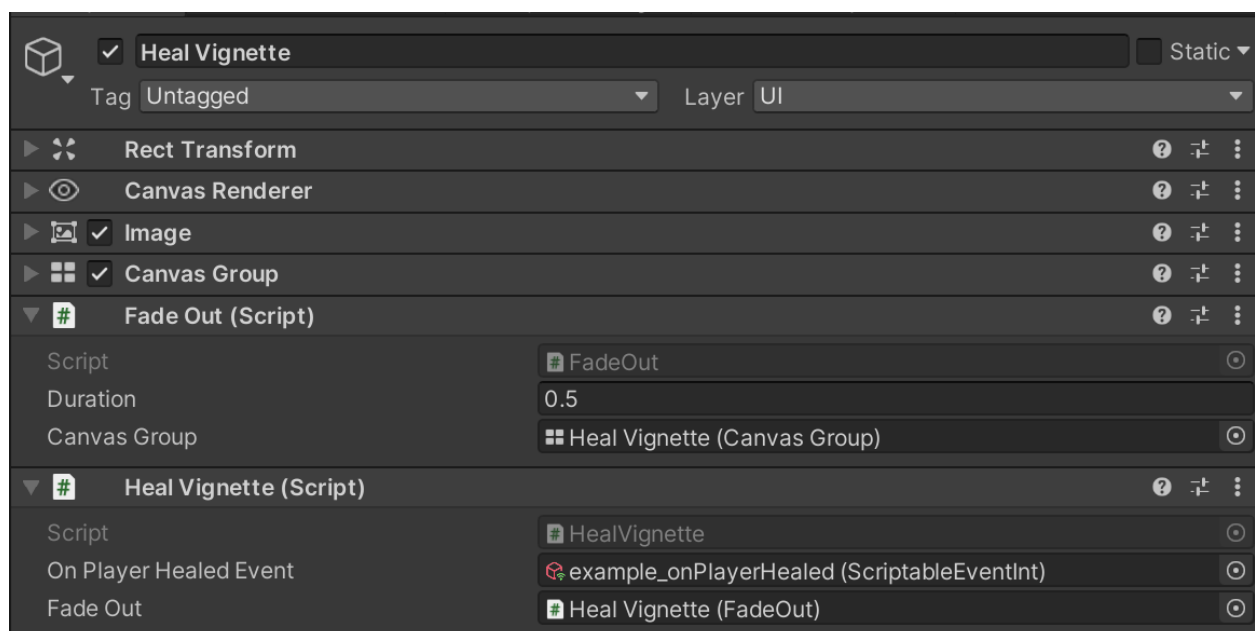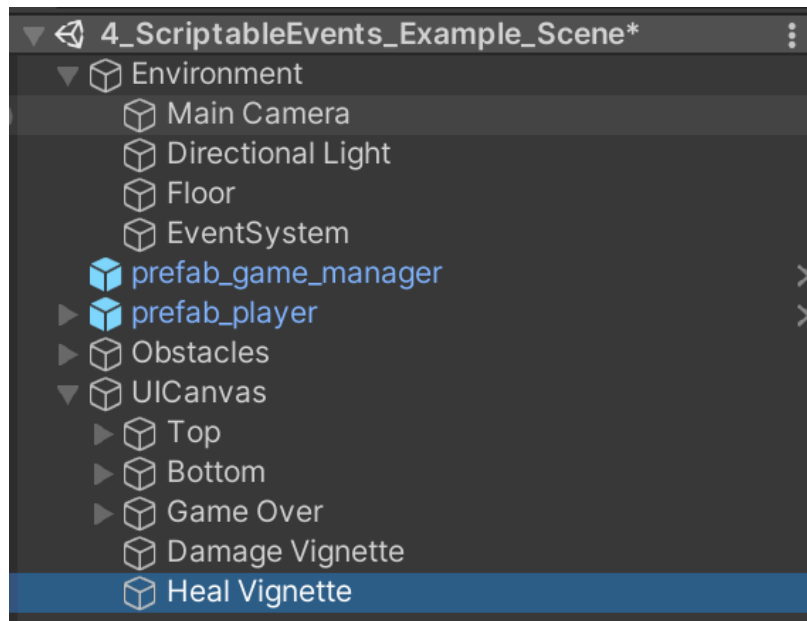
Lets inspect the code of the HealVignette.cs. Registering to an event is straighforward, simply subscribe your method to the **OnRaised** action of the event. Your method will then be called when the event is raised. In this case, the metod OnPlayerHealed will be called and activate the fade out.

```csharp
// 1 asset usage
public class HealVignette : MonoBehaviour
{
    [SerializeField] private ScriptableEventInt _onPlayerHealedEvent = null;
    [SerializeField] private FadeOut _fadeOut = null;    // Heal Vignette (FadeOut)

    // Event function
    private void Awake()
    {
        _onPlayerHealedEvent.OnRaised += OnPlayerHealed;
    }

    // Event function
    private void OnDestroy()
    {
        _onPlayerHealedEvent.OnRaised -= OnPlayerHealed;
    }

    // Frequently called    // 2 usages
    private void OnPlayerHealed(int amount)
    {
        _fadeOut.Activate();
    }
}
```
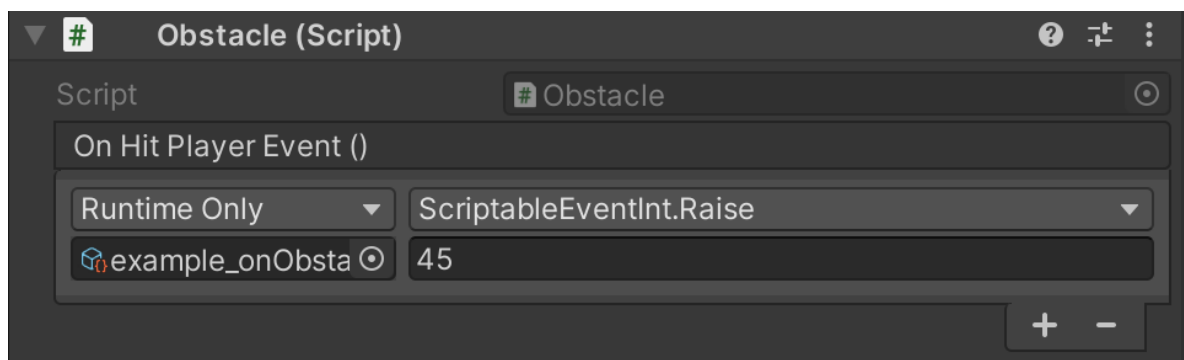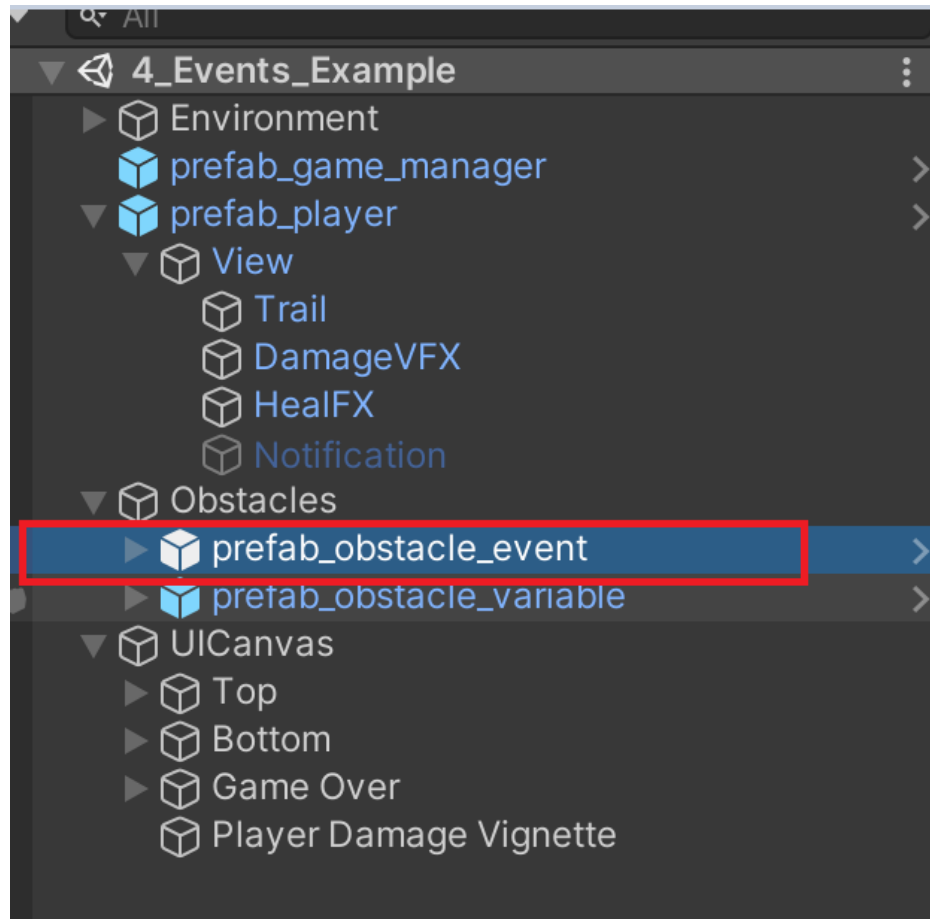
Note : don't forget to unsubscribe from the event OnDestroy() or OnDisable() for safety.

Firing events from the editor

You don't always need to use code to fire an event. Because it is a Scriptable Object, you can call it from the editor. If you select the **prefab_obstacle_event**, you can see that the event **example_onObstacleHitPlayer** is raised directly from a unity event.

There is an EventListener int on the **prefab_player** that listens to it.
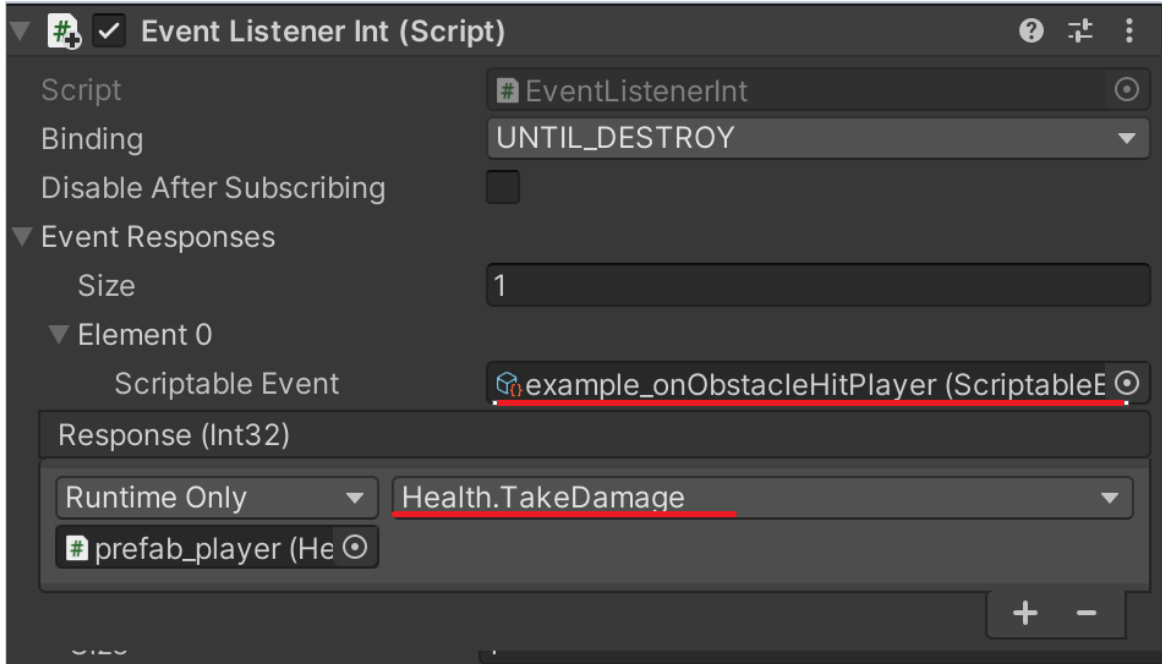
Debug

You can also easily debug events by raising them from the editor. Just select any of your scriptable event and click on the Raise button at runtime.