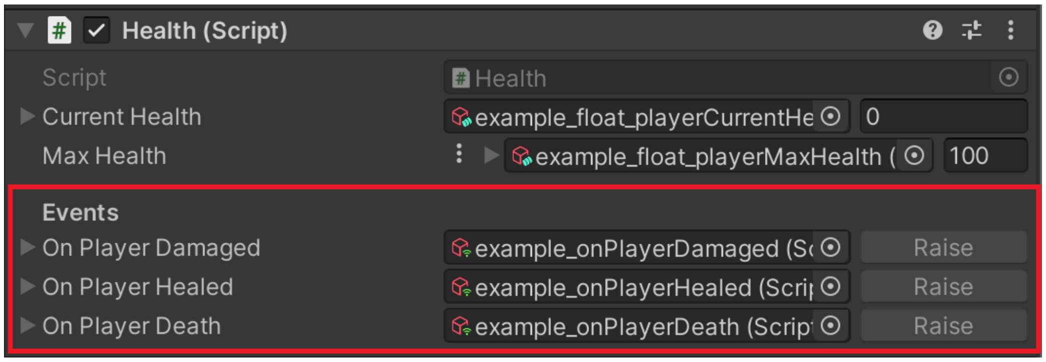
4.Scriptable Events

Table of Content

Firing events from code

Events are useful 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**.

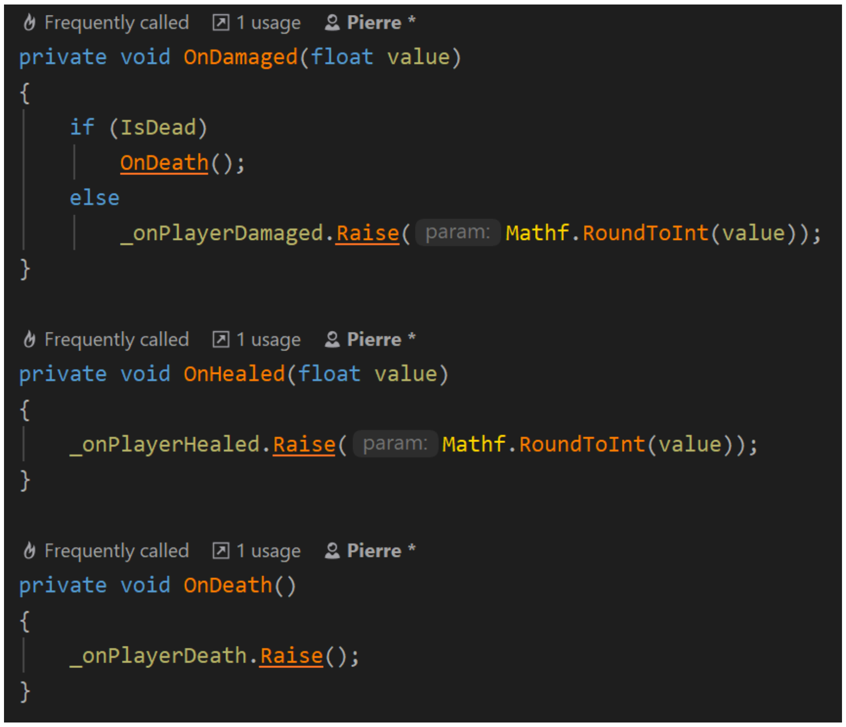
- OnPlayerDamaged -> ScriptableEventInt
- OnPlayerHealed -> ScriptableEventInt
- OnPlayerDeath -> ScriptableEventNoParam



Health (Script)

Script	Health	
Current Health	example_float_playerCurrentHe	0
Max Health	example_float_playerMaxHealth (	100
Events		
On Player Damaged	example_onPlayerDamaged (Sc	Raise
On Player Healed	example_onPlayerHealed (Scri	Raise
On Player Death	example_onPlayerDeath (Scrip	Raise

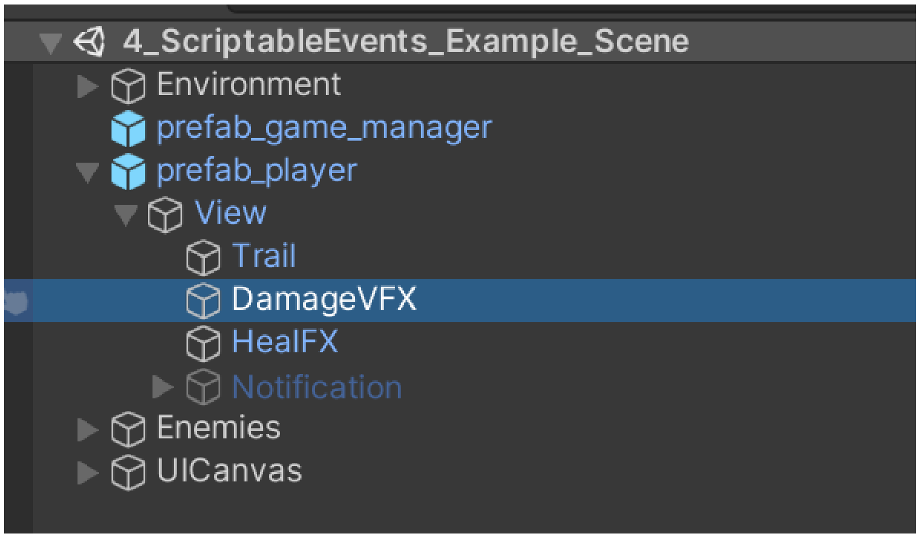
It is simple to fire these events from code, you simply call the **Raise**() method. Here is an example in **Health.cs**:

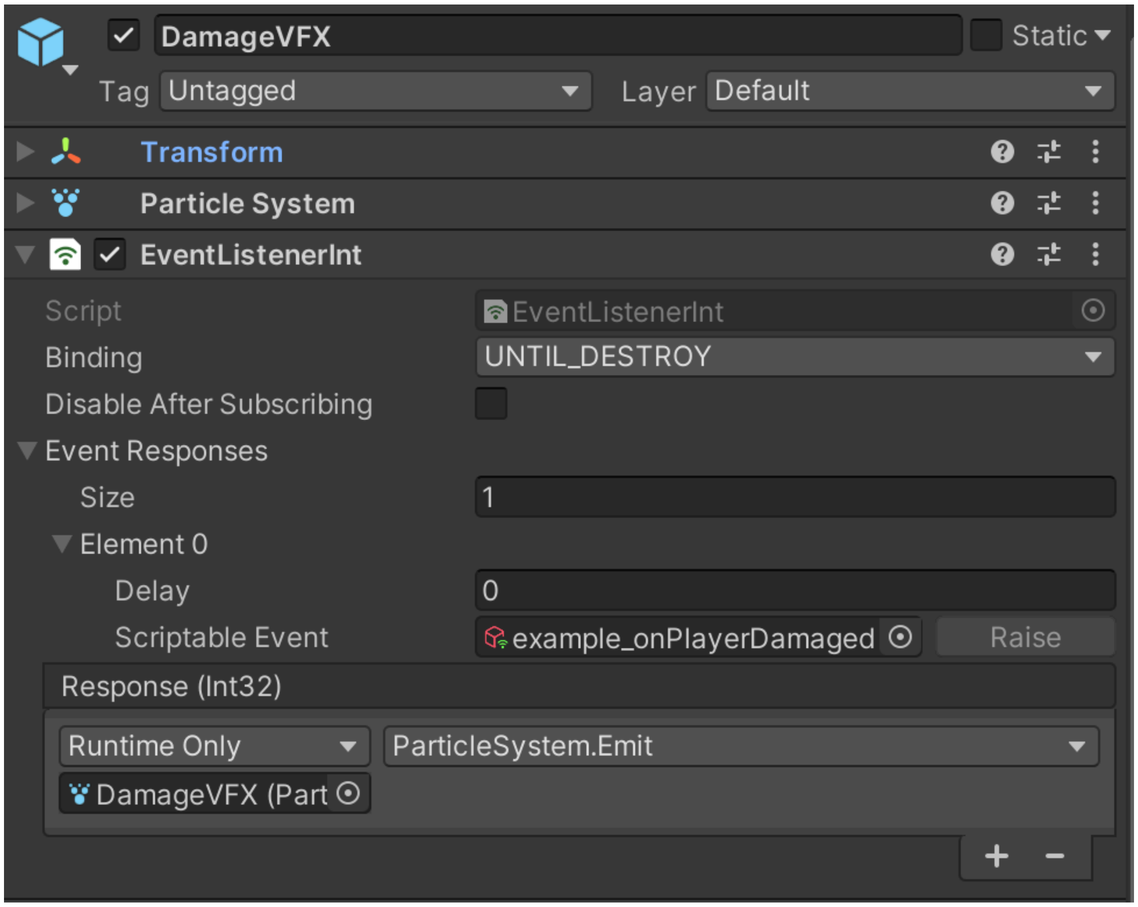


```
🔥 Frequently called   ↗ 1 usage   👤 Pierre *
private void OnDamaged(float value)
{
    if (IsDead)
        OnDeath();
    else
        _onPlayerDamaged.Raise( param: Mathf.RoundToInt(value));
}

🔥 Frequently called   ↗ 1 usage   👤 Pierre *
private void OnHealed(float value)
{
    _onPlayerHealed.Raise( param: Mathf.RoundToInt(value));
}

🔥 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**.



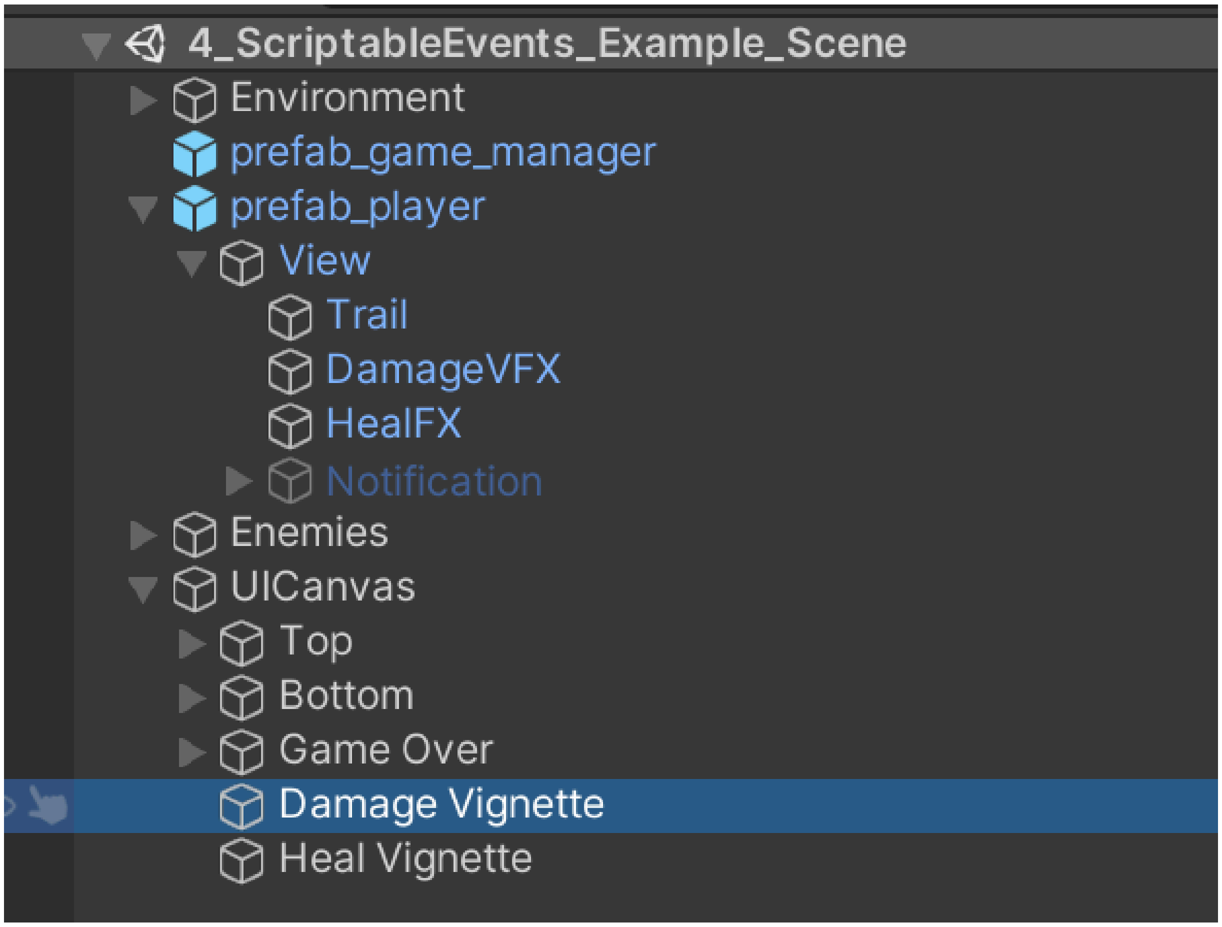

DamageVFX
Static
Tag Untagged
Layer Default
Transform
Particle System
EventListenerInt
Script EventListenerInt
Binding UNTIL_DESTROY
Disable After Subscribing
Event Responses
Size 1
Element 0
Delay 0
Scriptable Event example_onPlayerDamaged
Raise
Response (Int32)
Runtime Only
ParticleSystem.Emit
DamageVFX (Part

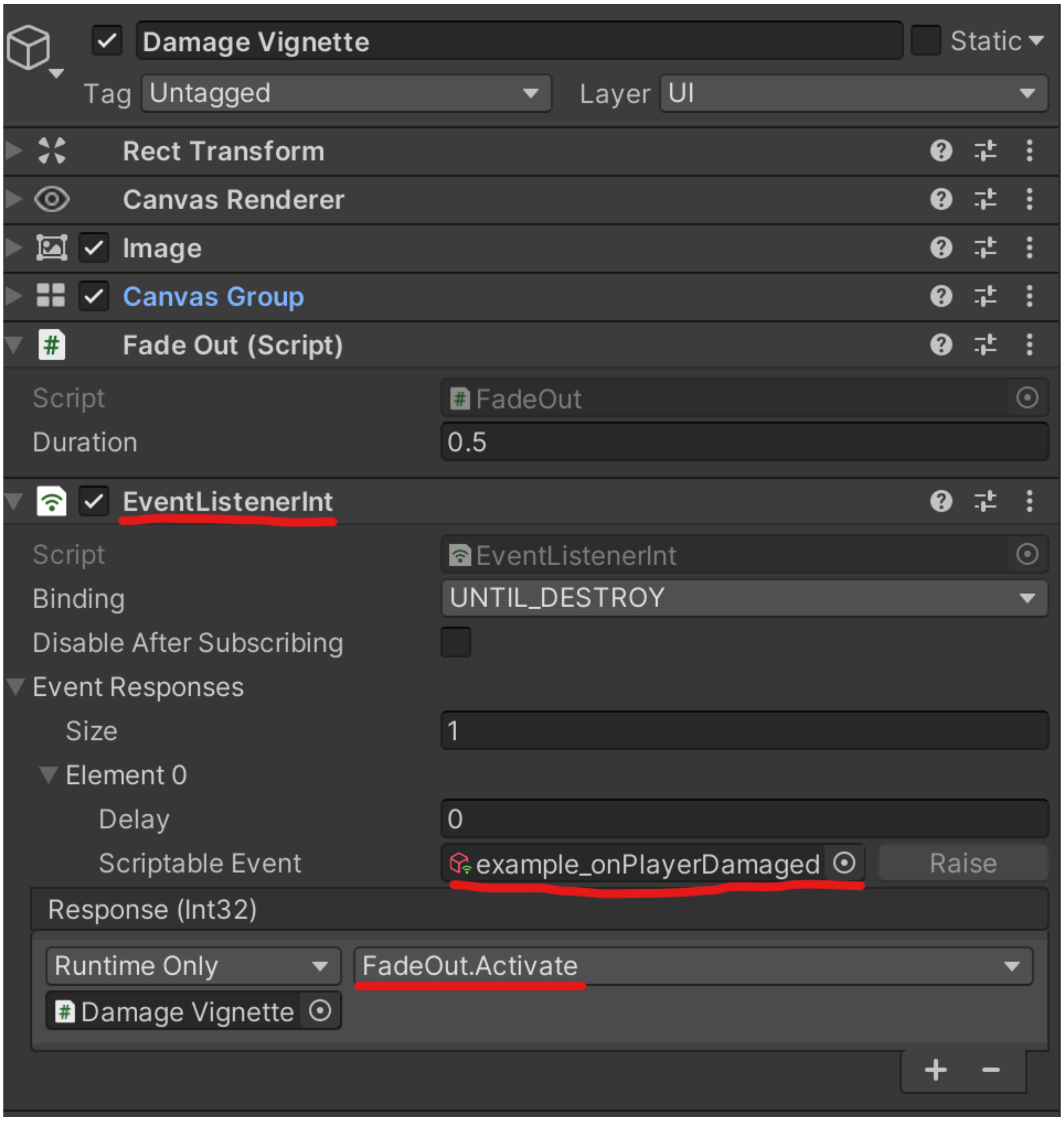
When the player is damaged, this event is fired and it will trigger the particle system **dynamic method** "Emit" that takes an int as a parameter. With this simple trick, we can visually show a relation between the amount of damage and the intensity of the VFX without code. A damage of 10 will emit 10 particles, 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 amounts of damage, but in Hyper Casual 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, there is a red vignette effect appearing then fading out.

UICanvas/Damage Vignette


4_ScriptableEvents_Example_Scene
Environment
prefab_game_manager
prefab_player
View
Trail
DamageVFX
HealFX
Notification
Enemies
UICanvas
Top
Bottom
Game Over
Damage Vignette
Heal Vignette


Damage Vignette
Static
Tag Untagged
Layer UI
Rect Transform
Canvas Renderer
Image
Canvas Group
Fade Out (Script)
Script FadeOut
Duration 0.5
EventListenerInt
Script EventListenerInt
Binding UNTIL_DESTROY
Disable After Subscribing
Event Responses
Size 1
Element 0
Delay 0
Scriptable Event example_onPlayerDamaged
Raise
Response (Int32)
Runtime Only
FadeOut.Activate
Damage Vignette

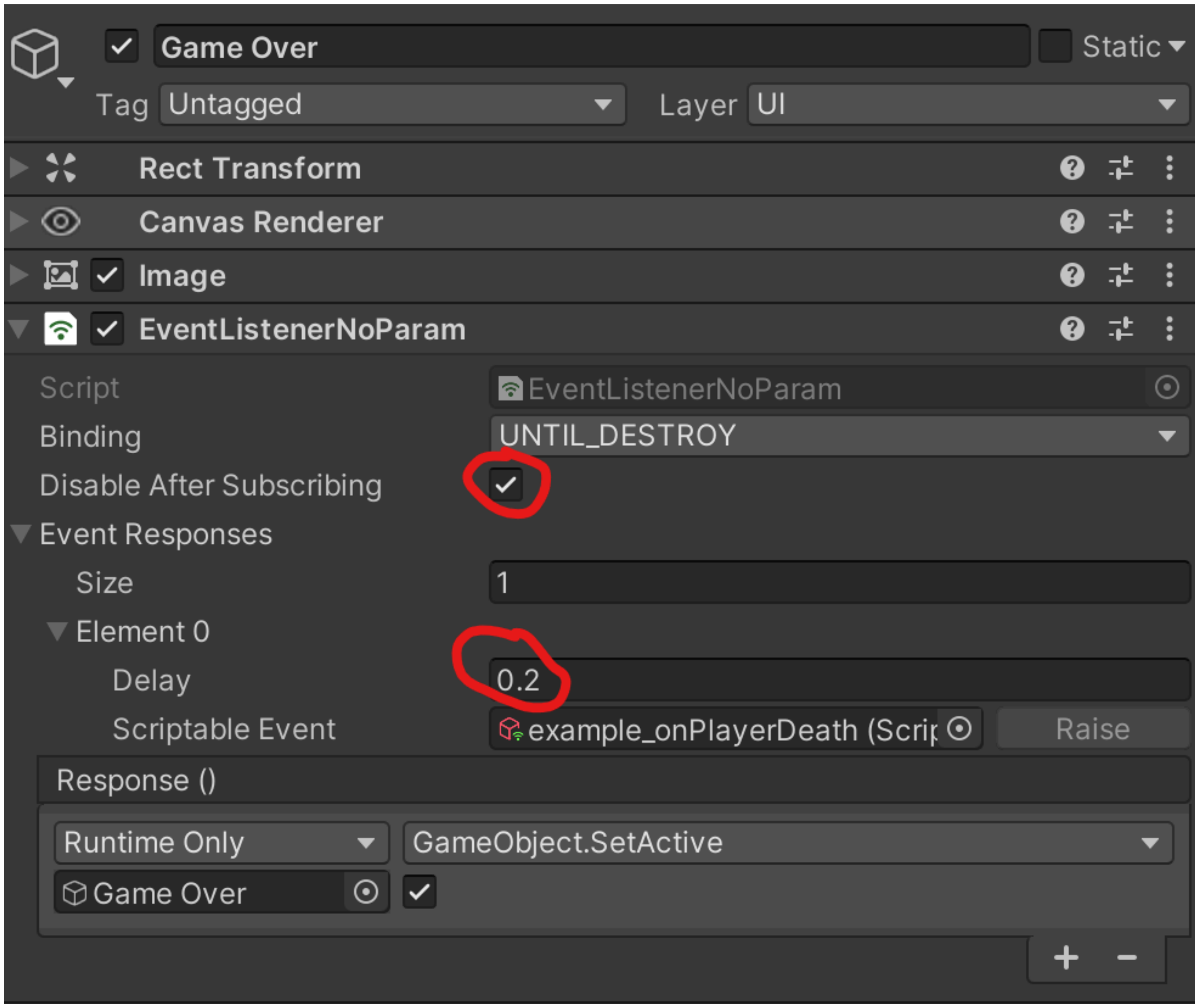
As we can see, here we listen to the **OnPlayerDamaged** event. When it is fired, we activate the Fade out.

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 listed by the Game Over screen.

You will notice that the Game Over Screen is visible in the editor and gets disabled when the game starts. Indeed, it needs to be enabled to register to the event at the beginning of the game.

This is 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().



✓ Game Over Static
Tag: Untagged Layer: UI

Rect Transform
Canvas Renderer
✓ Image
✓ EventListenerNoParam

Script	EventListenerNoParam
Binding	UNTIL_DESTROY
Disable After Subscribing	✓
Event Responses	
Size	1
Element 0	
Delay	0.2
Scriptable Event	example_onPlayerDeath (Scrip

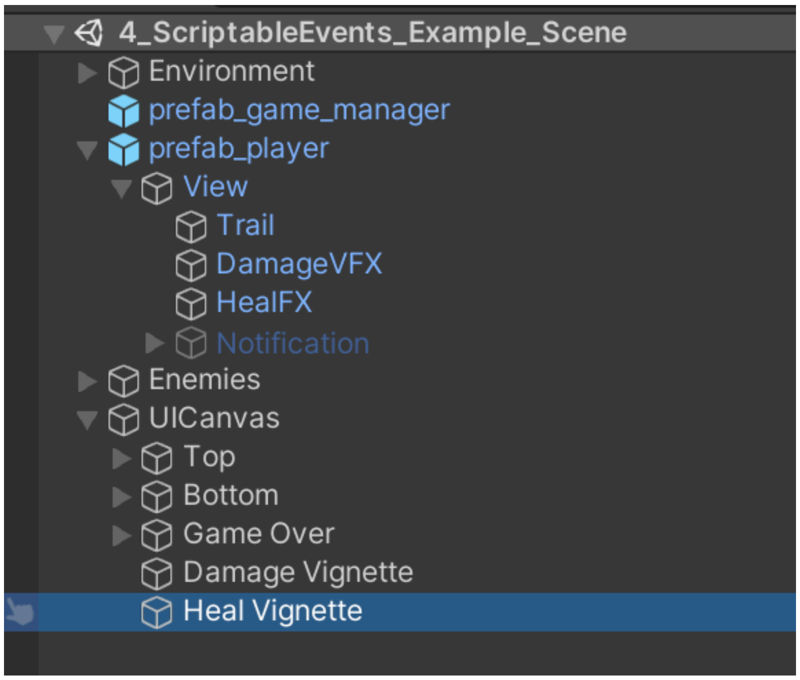
Response ()
Runtime Only — GameObject.SetActive
Game Over — ✓

You will notice also that we added a small delay before invoking the response. This is useful to add small delays without needing to write code to do it.

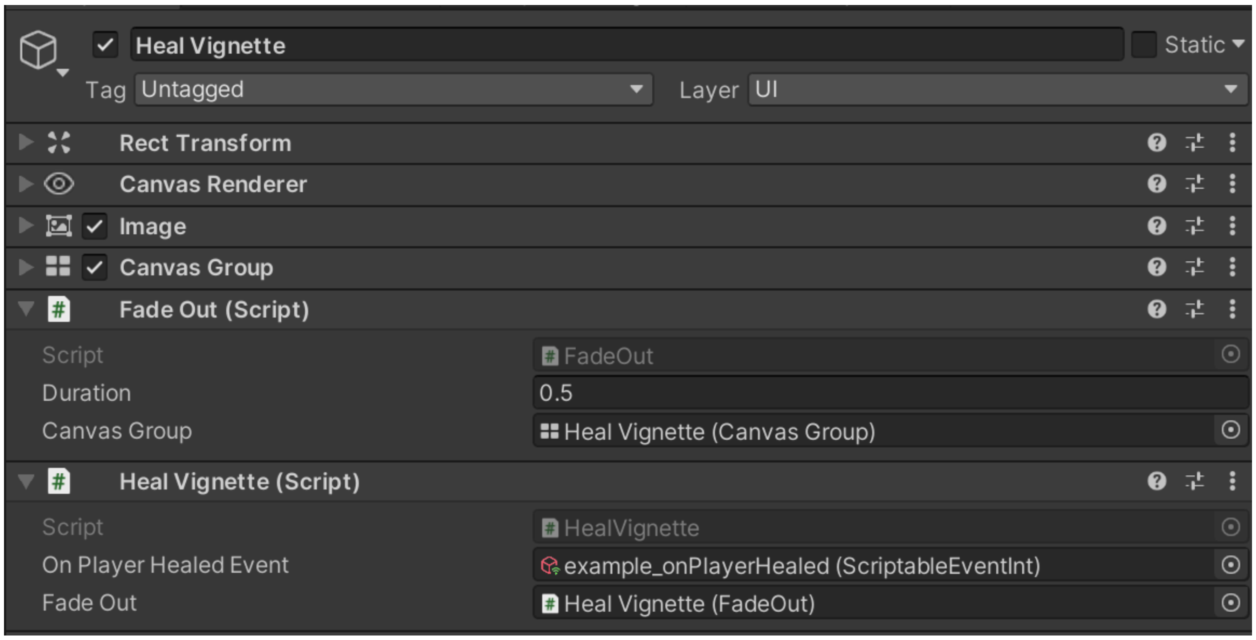
Registering to events from code

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

UICanvas/Heal Vignette



```
4_ScriptableEvents_Example_Scene
  Environment
  prefab_game_manager
  prefab_player
    View
      Trail
      DamageVFX
      HealFX
      Notification
  Enemies
  UICanvas
    Top
    Bottom
    Game Over
    Damage Vignette
    Heal Vignette
```



Heal Vignette — Static
Tag: Untagged — Layer: UI

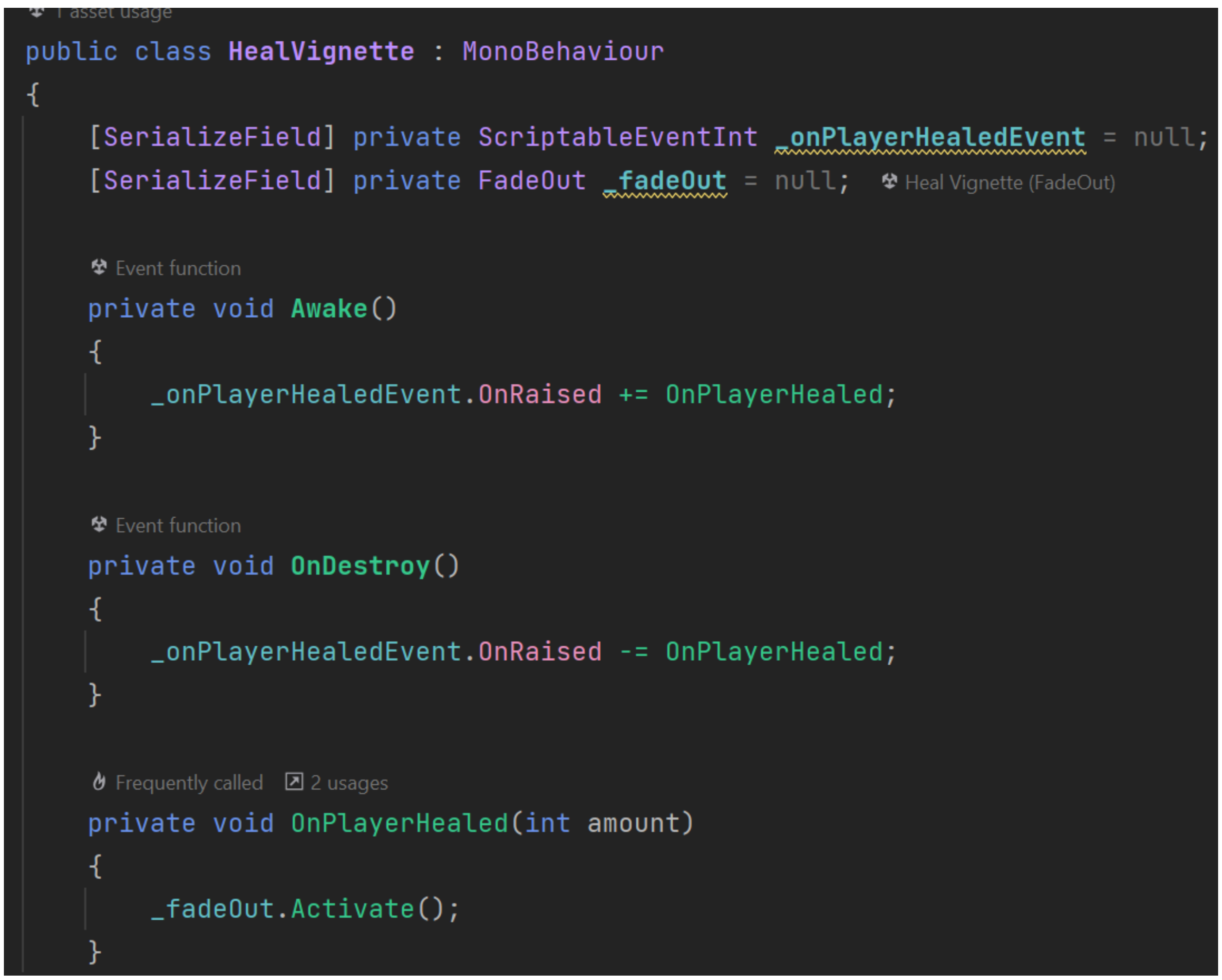
Rect Transform
Canvas Renderer
Image
Canvas Group
Fade Out (Script)

Script	FadeOut
Duration	0.5
Canvas Group	Heal Vignette (Canvas Group)

Heal Vignette (Script)

Script	HealVignette
On Player Healed Event	example_onPlayerHealed (ScriptableEventInt)
Fade Out	Heal Vignette (FadeOut)

Let's inspect the code of the HealVignette.cs. Registering to an event is straightforward, 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 method OnPlayerHealed will be called and activate the fade out.

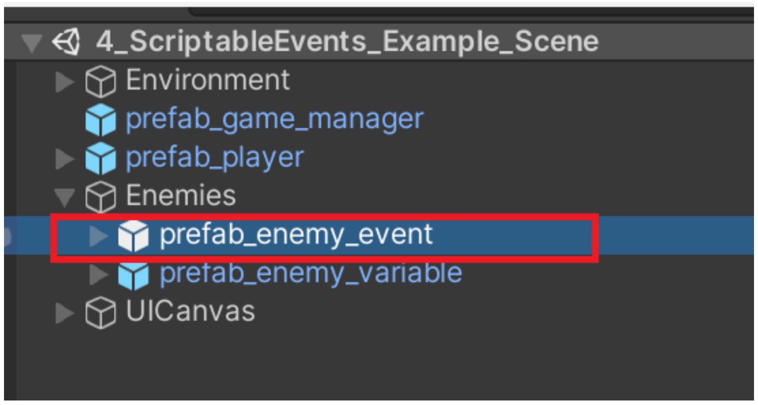


```
public class HealVignette : MonoBehaviour
{
    [SerializeField] private ScriptableEventInt _onPlayerHealedEvent = null;
    [SerializeField] private FadeOut _fadeOut = null;  Heal Vignette (FadeOut)

    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();
    }
```

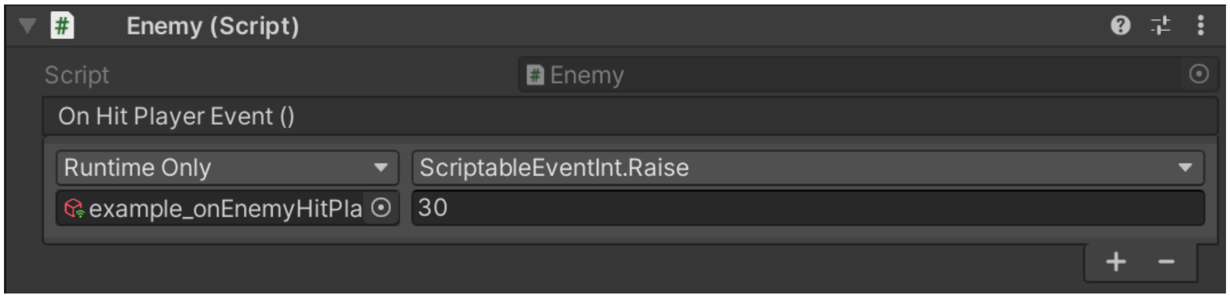
Note : don't forget to unsubscribe from the event OnDestroy() or OnDisable() for safety.

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_enemy_event**, you can see that the event **example_onEnemyHitPlayer** is raised directly from a unity event.

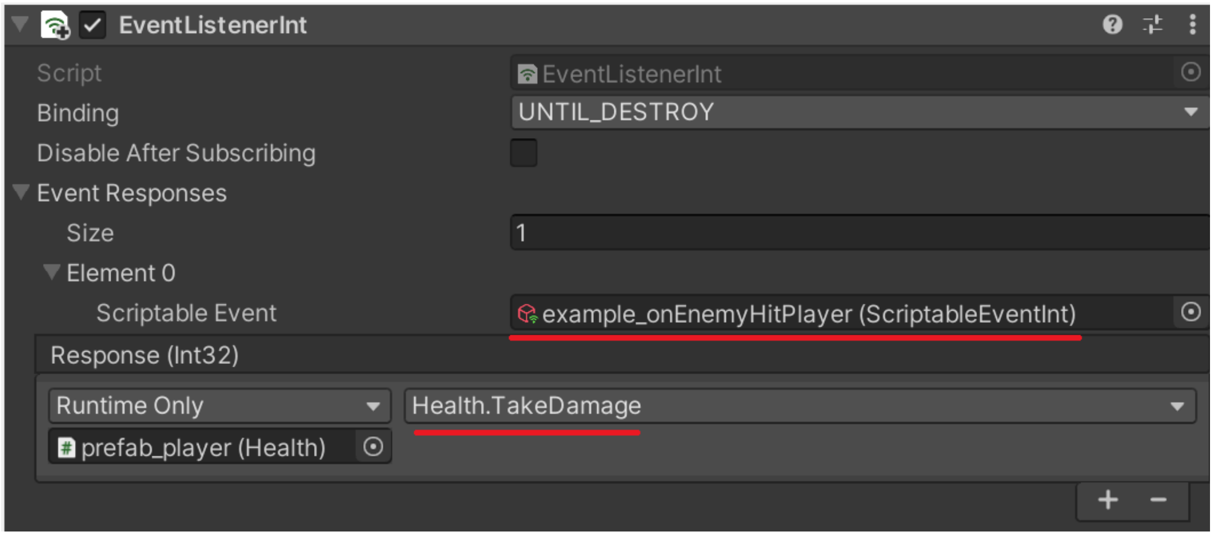


4_ScriptableEvents_Example_Scene
Environment
prefab_game_manager
prefab_player
Enemies
prefab_enemy_event
prefab_enemy_variable
UICanvas



Enemy (Script)
Script: Enemy
On Hit Player Event ()
Runtime Only | ScriptableEventInt.Raise
example_onEnemyHitPla | 30

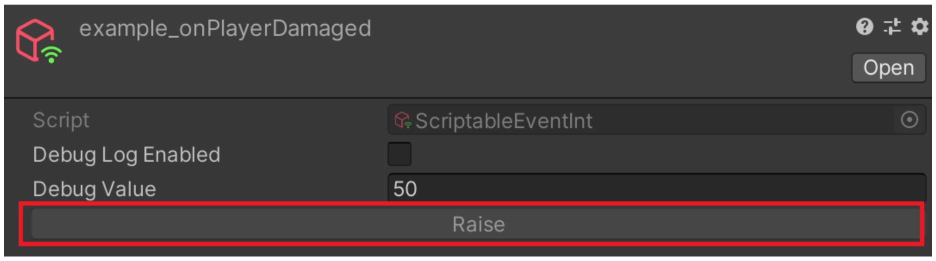
There is an EventListenerInt on the **prefab_player** that listens to it.



EventListenerInt
Script: EventListenerInt
Binding: UNTIL_DESTROY
Disable After Subscribing: ☐
Event Responses
Size: 1
Element 0
Scriptable Event: example_onEnemyHitPlayer (ScriptableEventInt)
Response (Int32)
Runtime Only | Health.TakeDamage
prefab_player (Health)

Debug

You can also easily debug events by raising them from the editor. Just select any of your scriptable events and click on the Raise button at runtime. If you want to see console logs when this event is fired and which method are called from it then set "debug log enabled" to true.



example_onPlayerDamaged

Open

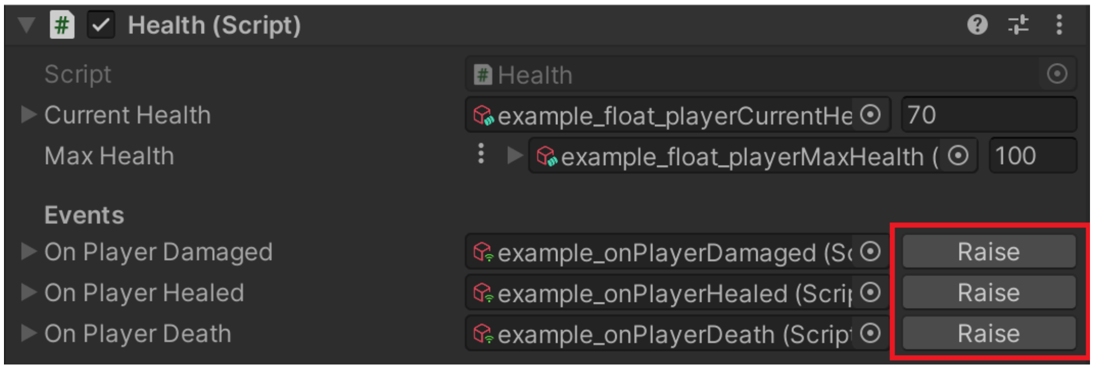
Script: ScriptableEventInt

Debug Log Enabled: ☐

Debug Value: 50

Raise

Or using the shortcut when it is exposed in your class:



Health (Script)

Script: Health

Current Health: example_float_playerCurrentHe 70

Max Health: example_float_playerMaxHealth (100

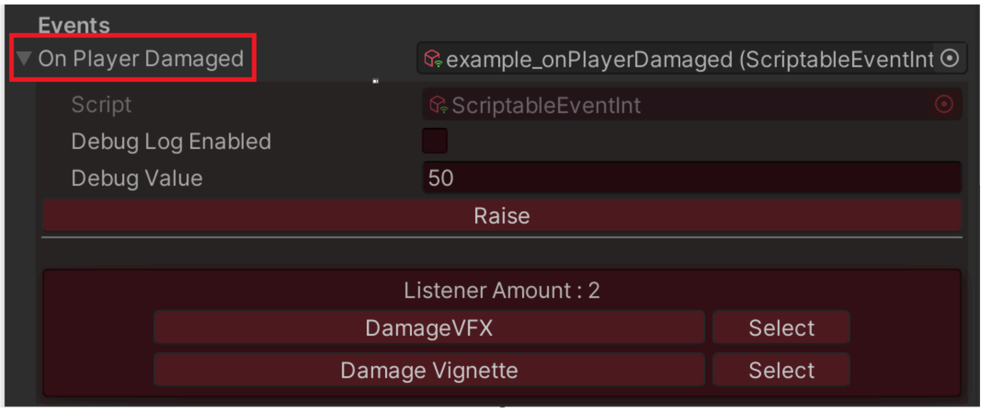
Events

On Player Damaged: example_onPlayerDamaged (S(Raise

On Player Healed: example_onPlayerHealed (Scri| Raise

On Player Death: example_onPlayerDeath (Scrip' Raise

Finally, you can quickly display an inline inspector from classes using the small arrow next to the variable name. Similarly to scriptable variables and lists. This is very useful to debug your events quickly.



Events

On Player Damaged: example_onPlayerDamaged (ScriptableEventInt

Script: ScriptableEventInt

Debug Log Enabled: ☐

Debug Value: 50

Raise

Listener Amount : 2

DamageVFX Select

Damage Vignette Select