Efectos de daño y sangre

Existe una forma de comunicación entre objetos un poco más compleja que el sistema de eventos que proporciona Unity.
Crearemos un script que se va a encargar de generar efectos audiovisuales en respuesta a determinados eventos que se den lugar durante el juego.

Trabajaremos con los eventos y delegados propios de C++, que son algo complicados pero muy prácticos.

Efectos de daño y sangre. Health

Usaremos un **Delegate** para definir un prototipo de función con el tipo que devuelve y los parámetros que necesitan.
Definimos la función delegada.

La segunda parte de la definición del evento consiste en definir el propio evento.

```
public delegate void _OnDamaged(GameObject go);
public static event _OnDamaged onDamaged;
```

Efectos de daño y sangre. Health

Un Delegate es una referencia a un método. Nos permite tratar ese método como una variable y pasarlo como variable para un callback. Cuando se invoca, notifica a todos los métodos que hacen referencia al Delegate. El funcionamiento es el del patrón Observador.

Ejemplo del mundo real es como el de un periódico. Una persona puede suscribirse a dicho periódico y cada vez que salga una nueva entrega todas las personas suscritas la recibirán.
Los eventos añaden una capa de abstracción y protección al Delegate. Esto previene al cliente del Delegate de resetearlo y eliminar la lista de suscriptores.

Efectos de daño y sangre. Health

Definimos el evento.

```
if (onDamaged != null)          Sabemos que alguien esta suscrito.
    onDamaged(gameObject);Lanzamos el evento y le pasamos el parámetro al G.O
if (currentHealth <= 0)
{
    onDead.Invoke();
}
```

Ahora debemos crear un Script para el creador de efectos y un GameObject vacío alv que lo asignaremos a la escena y el script.

Efectos de daño y sangre. Health

```
public class Health : MonoBehaviour
{
   public delegate void _OnDamaged(GameObject go);
   public static event _OnDamaged onDamaged;
   public float currentHealth = 5;
   public float maxHealth = 5;
   public UnityEvent onDamageTaken;
   public UnityEvent onDead;
   void DamageTaken (float amount)
   {
      currentHealth -= amount;
      onDamageTaken.Invoke();

      if (onDamaged != null)
         onDamaged(gameObject);

      if (currentHealth <= 0)
      {
         onDead.Invoke();
      }
   }
```

Efectos de daño y sangre. SFX

Cramos la función que se va a encargar del evento.

void HandleOnDamaged (GameObject go);

Debug.Log(go.name + «damaged»);

Nos suscribiremos al evento del script «Health» haciendo uso de dos funciones que forman parte de MonoBehaviour:

private void OnEnable()

{

Se ejecuta siempre que se activa o se crea un objeto.

}

private void OnDisable()

{

Lo contrario al anterior, se ejecuta siempre que se desactiva o justo antes de destruirse.

}

Efectos de daño y sangre. SFX

```
public class SFXmanager : MonoBehaviour
{
private void OnEnable()
   {
      Health.onDamaged += HandleOnDamaged;
   }
   private void OnDisable()
   {
Health.onDamaged -= HandleOnDamaged;
}
   void HandleOnDamaged (GameObject go)
   {
      Debug.Log (go.name + «damage»);
```

Efectos de daño y sangre. SFX

```
public class SFXmanager : MonoBehaviour
{
private void OnEnable()
   {
      Health.onDamaged += HandleOnDamaged;
   }
   private void OnDisable()
   {
Health.onDamaged -= HandleOnDamaged;
}
   void HandleOnDamaged (GameObject go)
   {
      Debug.Log (go.name + «damage»);
```

Creamos la partículas de sangre

Efectos de daño y sangre. SFX

```
public class SFXmanager : MonoBehaviour
{
private void OnEnable()
   {
      Health.onDamaged += HandleOnDamaged;
   }
   private void OnDisable()
   {
Health.onDamaged -= HandleOnDamaged;
}
   void HandleOnDamaged (GameObject go)
   {
      Debug.Log (go.name + «damage»);
```

Creamos la partículas de sangre y la añadimos como un Prefeb a nuestra carpeta de Prefabs.

Efectos de daño y sangre. SFX

```
public class SFXmanager : MonoBehaviour
{
    public GameObject BloodParticles;
    private void OnEnable()
    {
        Health.onDamaged += HandleOnDamaged;

    }

    private void OnDisable()
    {
        Health.onDamaged -= HandleOnDamaged;
    }
    void HandleOnDamaged (GameObject go)
    {
        GameObject.Instantiate(BloodParticles, go.transform.position, Quaternion.identity);.
```