Zombie Shooter Project 4a

Task 1. Make the player send health updates to the game UI

Explanation

- We will edit the **Player** script to send its current health to the **GameUI** when the **Player** takes damage

Do this

- In the **Scripts** folder of the **Project view**, open the **Player** script

Do this

- Add the highlighted parts to your Player script
- Make sure you add them in the same places as shown below

```
using UnityEngine;

public class Player : MonoBehaviour {

    public delegate void UpdateHealth(int newHealth);
    public static event UpdateHealth OnUpdateHealth;

    private Animator gunAnim;

    private void Start () {
        gunAnim = GetComponent<Animator>();
    }

    private void Update() {
        if(Input.GetMouseButton(0)) {
            GetComponent<Animator>().SetBool("isFiring", true);
        }
        else {
            GetComponent<Animator>().SetBool("isFiring", false);
        }
    }

    public void SendHealthData(int health) {
        if (OnUpdateHealth != null) {
            OnUpdateHealth(health);
        }
    }
}
```

Explanation - Delegates

- The first **property** we declare in this class is a **Delegate**
- A Delegate is a way of running many methods with only one method call

Useful links

- More information about **Delegates** — Delegates - Video

Explanation - UpdateHealth delegate

- **UpdateHealth** will send the **current health** of the **player**
- This is be useful for the **UI**
  - Later we will have a simple **UI** with a player **health display**
- **UpdateHealth** can also be useful to **check** if the **player** is **dead**, so we can go to a **Game Over** screen
- **UpdateHealth** is a type of **delegate**

```
public delegate void UpdateHealth(int newHealth);
```

Explanation - Events

- An Event will broadcast a message that other scripts can listen for
- Using a delegate, other scripts can make their own custom methods that run when the event is set
- For example
  - The player is damaged by a zombie
  - The UpdateHealth event runs
  - The Game UI has a HandeHealth method that listens to the UpdateHealth method
  - HandeHealth runs, showing the players updated health

Explanation - onUpdateHealth

- **onUpdateHealth** is the **Event** attached to our **UpdateHealth** delegate
- Note we put the word **"on"** before the **Delegate** name
  - This is a standard way of showing an **Event** and **Delegate** are attached
- Note also the **Delegate UpdateHealth** is declared as the type of **Event** (or how they are attached)
  - So our declaration is:
  - UpdateHealth (Delegate)
  - onUpdateHealth (Event)
- Note the Event is public and static
  - A static property can be accessed from other scripts without using a GetComponent call

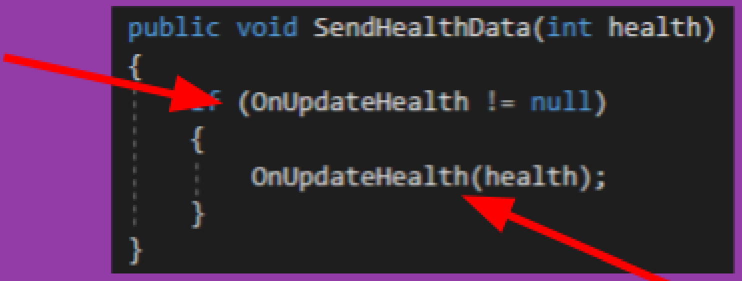
```
public static event UpdateHealth OnUpdateHealth;
```

Useful links

- More information about **Events** — Events - Video
- More information about **Statics** — Statics - Video

Explanation - code breakdown

if something is listening to our OnUpdateHealth message



```
public void SendHealthData(int health)
{
    if (OnUpdateHealth != null)
    {
        OnUpdateHealth(health);
    }
}
```

Send the message with the health value, provided as a parameter in OnHealthUpdate

Explanation - Line 1

- Check some other Gameobject is listening for the **OnUpdateHealth** event to fire
- A health bar would be interested in this data

```
public void SendHealthData(int health) {
    if (OnUpdateHealth != null) {
        OnUpdateHealth(health);
    }
}
```

Explanation - Line 2

- Send the **OnUpdateHealth** event with the health value

```
public void SendHealthData(int health) {
    if (OnUpdateHealth != null) {
        OnUpdateHealth(health);
    }
}
```

Task 1. Create an AddScore script

Explanation

- Here, we will add score to the player
- This script will enable the score to be sent for doing something
- In our case we will be adding score for killing a zombie
- We can easily use this script for adding score for collecting something, or lasting a certain amount of time or anything you want!

Do this

- In the **Project view**, create a new **C# Script** in the **Scripts Folder**
- Name the Script **AddScore**

Do this

- Type out this code into your script file
- Make sure your code is ***EXACTLY*** the same!

```
using UnityEngine;

public class AddScore : MonoBehaviour {

    public delegate void SendScore(int theScore);
    public static event SendScore OnSendScore;

    public int score = 10;

    private void OnDestroy() {
        if(OnSendScore != null) {
            OnSendScore(scoreToAdd);
        }
    }
}
```

Explanation - SendScore delegate

- **SendScore** will send the **score** stored in the **score** property
- This will be useful for the UI
    - We can display a total score that will be added to by this delegate
- **SendScore** is a type of **delegate**

```
public delegate void SendScore(int theScore);
```

Explanation - OnSendScore event

- **OnSendScore** is the event attached to our SendScore delegate
- **OnSendScore** will be the event that other scripts will listen to
    - The UI script will listen for the **OnSendScore**
- **OnSendScore** is a type of **event**
- **OnSendScore** is **public**, so it will be seen by other scripts
    - Note: it won't appear in the **Unity Editor**
- **OnSendScore** is static, so it can be accessed from other scripts without using **GetComponent**

```
public static event SendScore OnSendScore;
```

Explanation - score property

- **score** is the **message** that is sent with the **OnSendScore** event
- **score** is the amount of score you earn when sending the **OnSendScore** event
    - for example if you kill a zombie!
- **score** is a type of **int**
- **score** is a **public property**, so it is **editable** in the **Unity Editor**

```
public int score = 10;
```

Explanation - OnDestroy method

- **OnDestroy** is a method provided by Monobehaviour
- **OnDestroy** will run just before the GameObject is removed from the scene
- We can use this to send our score when a zombie dies

```
private void OnDestroy() {
    if(OnSendScore != null) {
        OnSendScore(score);
    }
}
```

Explanation - Line 1

- We check if the **OnSendScore** event has any scripts listening to it
  - If we don't, Unity will give us an error!

```
private void OnDestroy() {
    if(OnSendScore != null) {
        OnSendScore(score);
    }
}
```

Explanation - Line 2

- Here, we send the **OnSendScore** event to the listening scripts
- We send the **score** property with our **OnSendScore**

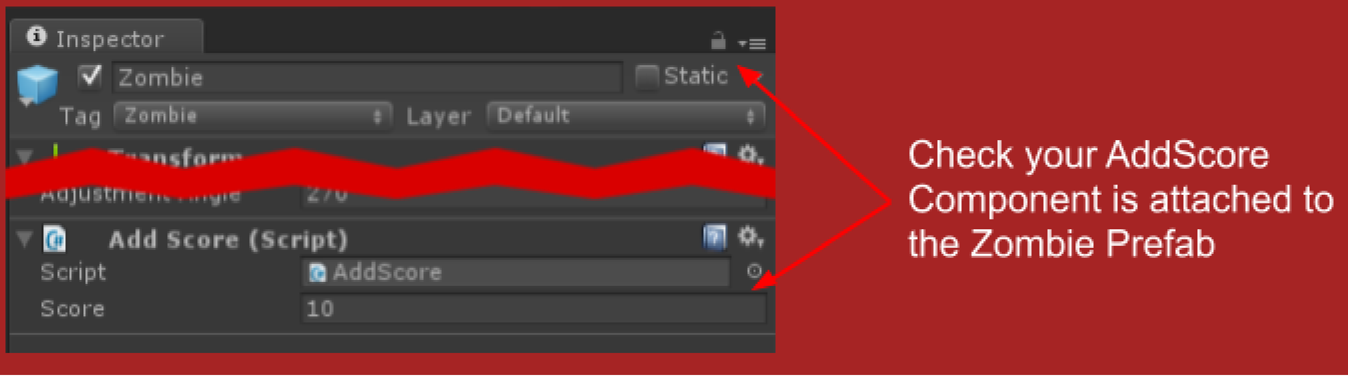
```
private void OnDestroy() {
    if(OnSendScore != null) {
        OnSendScore(score);
    }
}
```

Do this

- In the **Project view**, select the **Zombie** Prefab from the **Prefabs** folder
- Drag the **AddScore script** from the **Project view** onto the **Inspector**

Check this

- Check your **Zombie** Prefab has an **AddScore** Component



Script: AddScore
Score: 10

Check your AddScore Component is attached to the Zombie Prefab

Useful links

More information about **Delegates**	Delegates - Video
More information about **Events**	Events - Video
More information about **Statics**	Statics - Video