Zombie Shooter Project 4c

Task 1. Create a Game UI Script

Explanation

- Here we will code the player score and health UI
- This script will receive events for the player health from the **Player** component and the score from the **AddScore** component
- We will learn how to add and remove methods as listeners from a delegate
- We will use the Text and Slider UI components we setup in the previous section to display our score and health

Do this

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

Do this

- Select the **Canvas** in the **Hierarchy**
- Add the **GameUI script** to the **Canvas** GameObject as a component

Do this

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

```
using UnityEngine;
using UnityEngine.UI;

public class GameUI : MonoBehaviour {

    public Slider healthBar;
    public Text scoreText;

    public int playerScore = 0;

    private void OnEnable() {
        Player.OnUpdateHealth += UpdateHealthBar;
        AddScore.OnSendScore += UpdateScore;
    }

    private void OnDisable() {
        Player.OnUpdateHealth -= UpdateHealthBar;
        AddScore.OnSendScore -= UpdateScore;
    }

    private void UpdateHealthBar(int health) {
        healthBar.value = health;
    }

    private void UpdateScore(int theScore) {
        playerScore += theScore;
        scoreText.text = "SCORE: " + playerScore.ToString();
    }
}
```

Explanation - UnityEngine.UI

- Unity has a code library for dealing with UI components
- We have to include this whenever we deal with the UI components
- In our script we will be setting values on **Slider** and **Text** UI components, so we need to use the UI library to do so

```
using UnityEngine.UI;
```

Useful links

- More information about **Unity UI** — Unity UI - Manual

Explanation - healthBar property

- This is a reference to the **Slider** in the **Hierarchy**
- We will set its **Value** property to our players current health

```
public Slider healthBar;
```

Useful links

- More information about **Slider** — Slider - Manual
- More information about **Slider** — Slider - scripting reference

Explanation - scoreText property

- This is a reference to the **Text** in the **Hierarchy**
- We will set its **text** property to our players current score

```
public Text scoreText;
```

Useful links

- More information about **Text** — Text - Manual
- More information about **Text** — Text - scripting reference

Explanation - OnEnable method

- The **OnEnable** method is a MonoBehaviour method
- **OnEnable** is called when a GameObject has been created
  - If the GameObject is **disabled**, then **re-enabled**, it will be called again
- **OnEnable** runs just **BEFORE Start**
- Because it is only called once, we can use it to set things up ready for any update methods

```
private void OnEnable() {

}
```

Useful links

- More information about **OnEnable** — OnEnable - Scripting

Explanation - Line 1

- Here we add the GameUI script as a listener to the **OnHealthUpdate** event on the **Player**
- Note how we don't have to get the GameObject the **Player** script is attached to!
  - This is because the **Player OnUpdateHealth** event is **static**
- We add a custom method, **UpdateHealthBar** as our listener
  - This means, when the **OnUpdateHealth** method runs on the **Player**, **UpdateHealthBar** will run to!
- We use the **+=** operator to assign the **UpdateHealthBar** as a listener

```
private void OnEnable() {
    Player.OnUpdateHealth += UpdateHealthBar;
    AddScore.OnSendScore += UpdateScore;
}
```

Useful links

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

Explanation - Line 2

- Here we add the **GameUI** script as a listener to the **OnSendScore** event on the **AddScore** script
- There can be many **AddScore** scripts, when any one of them sends a message, this will receive it
- We add the custom method, **UpdateScore** to run when **OnSendScore** runs
- We once again use the **+=** to assign **UpdateScore** as a listener

```
private void OnEnable() {
    Player.OnUpdateHealth += UpdateHealthBar;
    AddScore.OnSendScore += UpdateScore;
}
```

Explanation - OnDisable method

- The **OnDisable** method is a MonoBehaviour method
- **OnDisable** is called when a GameObject has been disabled
    - If the GameObject is **enabled**, then **disabled**, it will be called
- **OnDisable** runs only after a GameObject has been **disabled**
- **OnDisable** will run every time a GameObject is **disabled**

```
private void OnDisable() {

}
```

Useful links

- More information about **OnDisable** — OnDisable - Scripting

Explanation - Line 1

- When we attach an event, we need to detach it when we are done
- If we don't we could get errors about memory
- This is because Unity doesn't know if we are done using the GameObject listening to the event
- So we need to detach events the same way we attached them (nearly!)
- We use the **-=** operator to detach our **UpdateHealthBar** method from the **Player.OnUpdateHealth** event

```
private void OnDisable() {
    Player.OnUpdateHealth -= UpdateHealthBar;
    AddScore.OnSendScore -= UpdateScore;
}
```

Explanation - Line 2

- We detach the **UpdateScore** event in the same way as the health

```
private void OnDisable() {
    Player.OnUpdateHealth -= UpdateHealthBar;
    AddScore.OnSendScore -= UpdateScore;
}
```

Explanation - UpdateHealthBar method

- We set the health amount here
- The healthBar is a Slider, which has a value property we can set to our players current health

```
private void UpdateHealthBar(int health) {
    healthBar.value = health;
}
```

Explanation - Line 1

- We set our **healthBar**'s value property to the **health** parameter provided by our **UpdateHealthBar** method
- The **Player** component will send us the current health value, as shown in the **OnEnable** method earlier

```
private void UpdateHealthBar(int health) {
    healthBar.value = health;
}
```

Useful links

- More information about **Slider.value** — Slider.value - Scripting

Explanation - UpdateScore method

- We add to the score amount here
- We add the new score (**theScore**) to the current score, **playerScore**
- The **scoreText** is a **Text**, which has a **text** property we can set to our players current score, **playerScore**

```
private void UpdateScore(int theScore) {
    playerScore += theScore;
    scoreText.text = "SCORE: "+ playerScore.ToString();
}
```

Explanation - Line 1

- We add our **theScore** parameter to our **playerScore** property
- The **playerScore** property is the total score for the game

```
private void UpdateScore(int theScore) {
    playerScore += theScore;
    scoreText.text = "SCORE: " + playerScore.ToString();
}
```

Explanation - Line 2

- We set the text property of scoreText to the current player score, **playerScore**
- We add the string “SCORE: ” to our **scoreText.text** value, so users know what the number represents
- Adding strings together like this is called **concatenation**
- The **playerScore** is an **int**, so we need to convert it to a **string** to display on our **Text**
- We use **.ToString()** to convert the **int** to a **string**

```
private void UpdateScore(int theScore) {
    playerScore += theScore;
    scoreText.text = "SCORE: " + playerScore.ToString();
}
```

Useful links

- More information about **Concatenation** — Concatenation
- More information about **ToString** — ToString