Zombie Shooter Project 3f

Task 1. Create a TimerEvent script

Explanation

- We want to spawn zombies when the game is running
- Our Spawner script can do this for us, but we want to control how often zombies will spawn
- We can create a script with a timer to do this (and other things!)

Do this

- In the **Scripts** folder of the **Project view**, create a new script
- Name the script **TimerEvent**

Do this

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

```
using UnityEngine;
using UnityEngine.Events;

public class TimerEvent : MonoBehaviour {

    public float time = 1;
    public bool repeat = false;
    public UnityEvent onTimerComplete;

    private void Start () {
        if(repeat) {
            InvokeRepeating("OnTimerComplete", 0, time);
        }
        else {
            Invoke("OnTimerComplete", time);
        }
    }

    private void OnTimerComplete() {
        onTimerComplete.Invoke();
    }
}
```

Explanation - UnityEngine.Events

- Unity has an events system that will let GameObjects and components talk to each other
- We can set how these events communicate in the editor
- For example, our enemy may want to get the player transform when it spawns

```
using UnityEngine.Events;
```

Useful links

- More information about **UnityEvents** — UnityEvents - Scripting Reference

Explanation - time property

- The **time** is how long it takes for the timer to complete in seconds
- At default, **time** is set to 1 second

```
public float time = 1;
```

Explanation - repeat property

- Allows us to repeat the timer after it has finished
- If you only want the timer to run once, set this to **false**
- If you want it to run forever, set it to **true**

```
public bool repeat = false;
```

Explanation - onTimerComplete property

- When the timer completes, this event will send
- We can setup this event in the **editor** to make things to happen in the game - like spawning a zombie!

```
public UnityEvent onTimerComplete;
```

Explanation - Our Start method

- Here we either set a timer to run once, or forever
- If we set repeat to true, the timer will repeat forever
- If repeat is false, the timer runs once

```
private void Start () {
    if(repeat) {
        InvokeRepeating("OnTimerComplete", 0, time);
    }
    else {
        Invoke("OnTimerComplete", time);
    }
}
```

Explanation - Line 1

- If repeat is true
- NOTE: this is the same as stating “if(repeat == true)”

```
private void Start () {
    if(repeat) {
        InvokeRepeating("OnTimerComplete", 0, time);
    }
    else {
        Invoke("OnTimerComplete", time);
    }
}
```

Explanation - Line 2

- Here we call **InvokeRepeating**, a timer that will repeat forever!
- The first parameter is the method name (called **OnTimerComplete**) we will run every time the timer completes
- NOTE: the method name needs to be a **string**
- The second parameter is the time to wait before starting, we insert zero to start immediately
- The third parameter is the time to complete, we insert our public property, **time**

```
private void Start () {
    if(repeat) {
        InvokeRepeating("OnTimerComplete", 0, time);
    }
    else {
        Invoke("OnTimerComplete", time);
    }
}
```

Useful links

- More information about **InvokeRepeating** — InvokeRepeating - Scripting

Explanation - Line 3

- If our repeat is set to false, we will run the timer once only

```
private void Start () {
    if(repeat) {
        InvokeRepeating("OnTimerComplete", 0, time);
    }
    else {
        Invoke("OnTimerComplete", time);
    }
}
```

Explanation - Line 4

- Here we call **Invoke**, a timer that will run once
- The first parameter is the method name (called **OnTimerComplete**) we will run when the timer completes
- NOTE: the method name needs to be a **string**
- The second parameter is the time to complete, we insert our public property, **time**

```
private void Start () {
    if(repeat) {
        InvokeRepeating("OnTimerComplete", 0, time);
    }
    else {
        Invoke("OnTimerComplete", time);
    }
}
```

Explanation - Our OnTimerComplete method

- This method will run when the timer completes
- It will send the **onTimerComplete** event
- We can setup the **onTimerComplete** event in the editor to do things in the game - like spawn zombies!

```
private void OnTimerComplete() {
    onTimerComplete.Invoke();
}
```

Explanation - Line 1

- We use the **Invoke** method of **onTimerComplete** to send the event

```
private void OnTimerComplete() {
    onTimerComplete.Invoke();
}
```

Useful links

- More information about **UnityEvents** — UnityEvents - Scripting Reference

Task 2. Create a Zombie spawner in the scene

Do this

- Create an empty **GameObject** in the scene using the **Create** button in the Hierarchy
- **Create** > **Create Empty**
- Name it **Zombie Spawner** in the Inspector

Do this

- Add the **TimerEvent** script to the **Zombie Spawner**
- Add the **Spawner** script to the **Zombie Spawner**
- Add your **Zombie prefab** from the **Project view** to the **Prefab To Spawn** inlet of the **Spawner** component

Do this

- Create a new listener on the **On Timer Complete** event using the plus button on the event
- Drag the **Spawner** component to the event
- NOTE: do not drag the script, drag the component you just added to the **Zombie Spawner**
- Select the **Spawner** > **Spawn** method from the dropdown on the event

Check this

- Check you have setup the **Zombie Spawner** like the image here: