Zombie Shooter Project 2d

Task 1. Health System

Do this

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

Do this

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

```
using UnityEngine;
using UnityEngine.Events;

[System.Serializable]
public class OnDamagedEvent : UnityEvent<int> { }

public class HealthSystem : MonoBehaviour {

    public int health = 10;
    public UnityEvent onDie;
    public OnDamagedEvent onDamaged;

    public void TakeDamage( int damage ){
        health -= damage;

        onDamaged.Invoke(health);

        if( health < 1 ) {
            onDie.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 spawn a blood explosion and destroy itself when its health runs out

```
using UnityEngine.Events;
```

Useful links

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

Explanation - health property

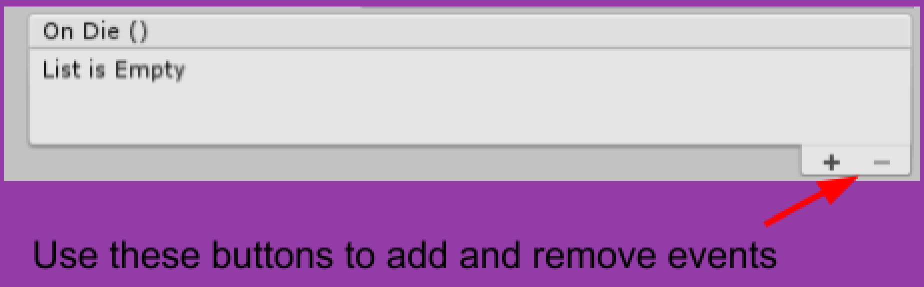
- The **Zombie** will have **health** that **reduces** when it is hit by a **Bullet**
- **health** is a type of **int** (a whole number like 1 or 39)
- **health** is a **public** property so it is **editable** in the **Unity Editor**

```
public int health = 10;
```

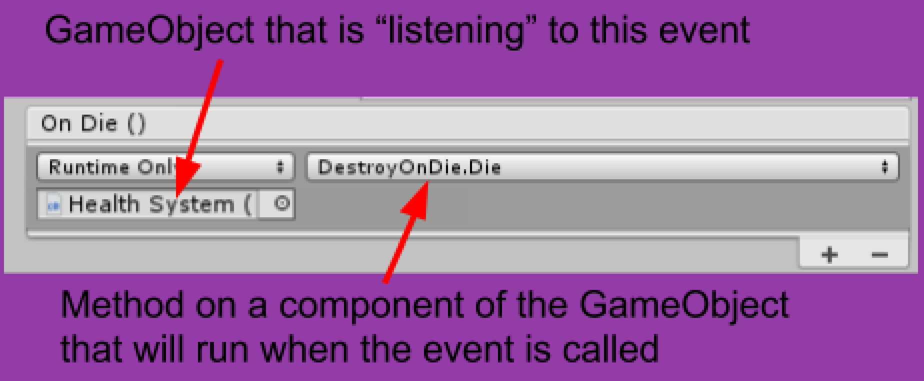
Explanation - onDie property

- onDie is a UnityEvent
- onDie can be configured in the Editor to do something when the event is called (when the zombie runs out of health)

Here is what an EMPTY event looks like in the Editor


On Die ()
List is Empty

Here is what an event looks like with a "listener" added (you can add as many "listeners" as you want!)

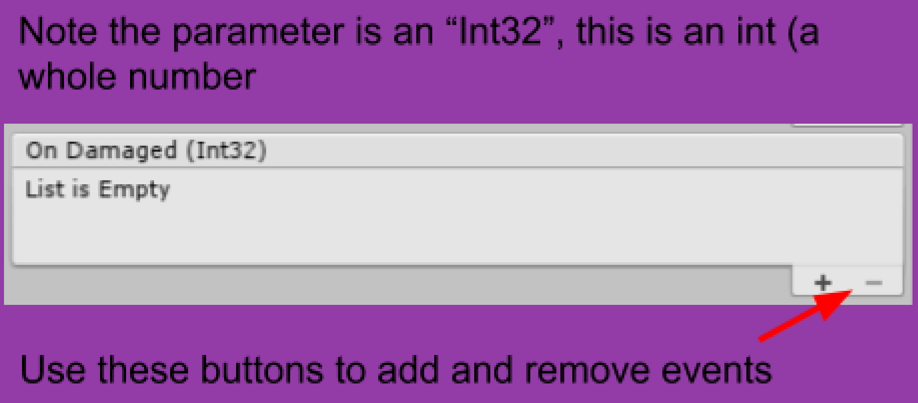

On Die ()
Runtime Only
DestroyOnDie.Die
Health System (
GameObject that is "listening" to this event
Method on a component of the GameObject that will run when the event is called
+
−

```
public UnityEvent onDie;
```

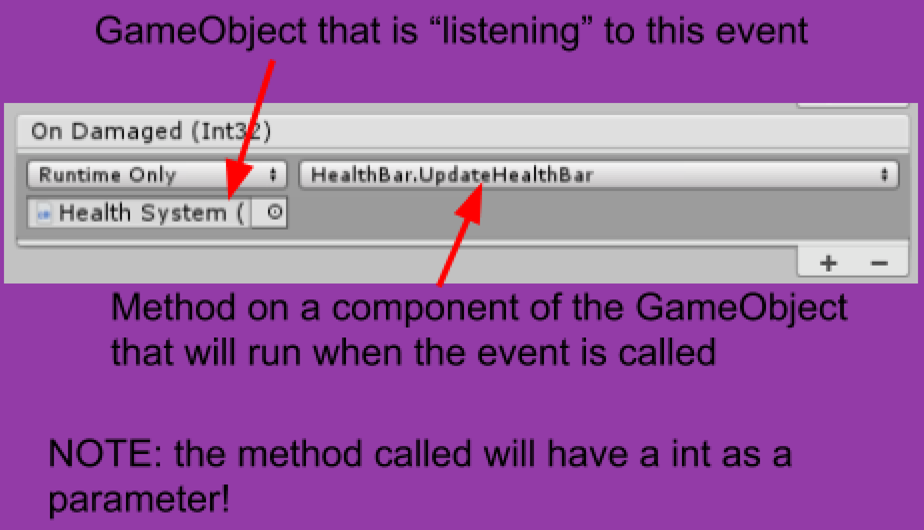
Explanation - onDamaged property

- **onDamaged** is a CUSTOM UnityEvent
- Custom UnityEvents can pass information, like how much health is left on the enemy
- NOTE: other value types can be used as the parameter - string, float, bool etc

Here is what an EMPTY custom event looks like in the Editor


On Damaged (Int32)
List is Empty
Note the parameter is an "Int32", this is an int (a whole number
+
−
Use these buttons to add and remove events

Here is what a custom event looks like with a "listener" added (you can add as many "listeners" as you want!)


On Damaged (Int32)
Runtime Only
HealthBar.UpdateHealthBar
Health System (
GameObject that is "listening" to this event
Method on a component of the GameObject that will run when the event is called
NOTE: the method called will have a int as a parameter!
+
−

```
public OnDamagedEvent onDamaged;
```

Useful links

- More information about **Custom UnityEvents** [Custom UnityEvents (1 parameter) - Scripting Reference]

Explanation - TakeDamage method signature

- **TakeDamage** will handle the Zombies health going down after being hit by a **Bullet**
- **TakeDamage** is a **public** method
    - This means it can be called by other classes
- The Bullet will call this method if it hits the Zombie
- TakeDamage has 1 **parameter** called **damage**
- **damage** is a type of int
- **damage** will remove health from the Zombie's health property

```
public void TakeDamage( int damage ){

}
```

Explanation - Our custom TakeDamage method

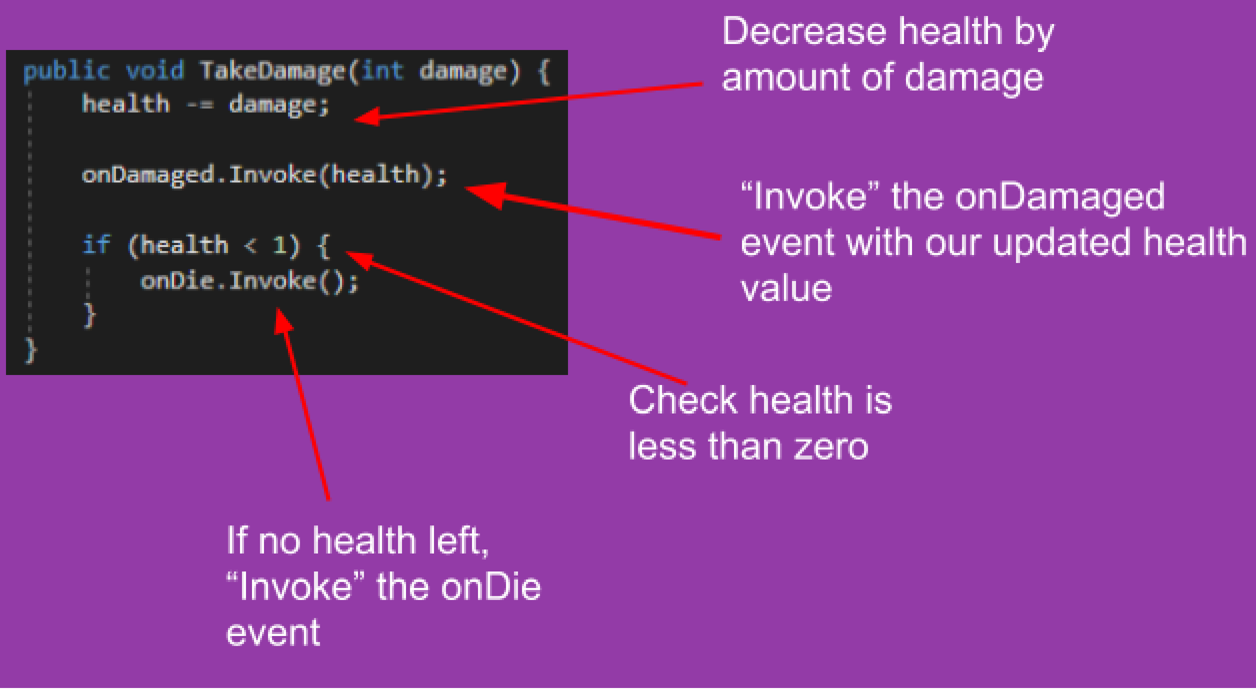
- Here is our custom **TakeDamage** method in full

```
public void TakeDamage( int damage ){
    health -= damage;

    onDamaged.Invoke(health);

    if( health < 1 ) {
        onDie.Invoke();
    }
}
```

Explanation - code breakdown



```
public void TakeDamage(int damage) {
    health -= damage;

    onDamaged.Invoke(health);

    if (health < 1) {
        onDie.Invoke();
    }
}
```

Decrease health by amount of damage
“Invoke” the onDamaged event with our updated health value
Check health is less than zero
If no health left, “Invoke” the onDie event

Explanation - Line 1

- The **Zombies health property** is **reduced** by the amount of **damage** given by the **damage parameter**
- The **health property** and **damage parameter** are both of type **int**
- We use the **-=** operator to minus the damage from the current health
    - “health -= damage” **is short for** “health = health - damage”

```
public void TakeDamage( int damage ){
    health -= damage;

    onDamaged.Invoke(health);

    if( health < 1 ) {
        onDie.Invoke();
    }
}
```

Useful links

- More information about **-= operator** -= operator

Explanation - Line 2

- All UnityEvents including custom ones have an **Invoke** method that will send the event to any “listeners” we added in the Editor
- Our custom event, **onDamaged** sends our current health value to any listeners

```
public void TakeDamage( int damage ){
    health -= damage;

    onDamaged.Invoke(health);

    if( health < 1 ) {
        onDie.Invoke();
    }
}
```

Useful links

- More information about **Invoke** [Invoke - Scripting Reference]

Explanation - Line 3

- Now the **Zombies health** has been **reduced**, we **check** its **new value** to see if it is **zero or lower**
- We use the **<= operator** to check if the **health** is **zero** or **below**
  - <= means “Less than or equal to”

```
public void TakeDamage( int damage ){
    health -= damage;

    onDamaged.Invoke(health);

    if( health < 1 ) {
        onDie.Invoke();
    }
}
```

Useful links

- More information about <= **operator** [<= operator]

Explanation - Line 4

- All UnityEvents including custom ones have an **Invoke** method that will send the event to any “listeners” we added in the Editor
- Our onDie event has no data to send, so we call Invoke with no parameters

```
public void TakeDamage( int damage ){
    health -= damage;

    onDamaged.Invoke(health);

    if( health <= 0 ) {
        onDie.Invoke();
    }
}
```

Useful links

- More information about **Invoke** [Invoke - Scripting Reference]

Do this

- In the **Unity Editor**, select the **HealthSystem** script in the **Project view**
- **Drag** the **HealthSystem** script onto the **Zombie GameObject** in the **Hierarchy**

INTERACTIVE SYSTEMS STUDIO

CGD