Zombie Shooter Project 2c

Task 1. Shoot the bullet!

Do this

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

Do this

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

```
using UnityEngine;

public class Weapon : MonoBehaviour {

    public GameObject bulletPrefab;
    public Transform bulletSpawn;
    public float fireTime = 0.5f;

    private bool isFiring = false;

    private void SetFiring(){
        isFiring = false;
    }

    private void Fire(){
        isFiring = true;
        Instantiate( bulletPrefab, bulletSpawn.position, bulletSpawn.rotation );

        if( GetComponent<AudioSource>() != null ) {
            GetComponent<AudioSource>().Play();
        }

        Invoke( "SetFiring", fireTime );
    }

    private void Update(){
        if( Input.GetMouseButton(0) ) {
            if( !isFiring ) {
                Fire();
            }
        }
    }
}
```

Explanation - bulletPrefab property

- The **Prefab** we will fire as a **Bullet**
- **bulletPrefab** is a type of **GameObject**
- **bulletPrefab** is a **public** property so it is **editable** in the **Unity Editor**

```
public GameObject bulletPrefab;
```

Explanation - bulletSpawn property

- The position and rotation in the scene your **Bullet** is spawn from
- **bulletSpawn** is a type of **Transform**
- **bulletSpawn** is a **public** property, so it is **editable** in the **Unity Editor**

```
public Transform bulletSpawn;
```

Explanation - fireTime property

- The time in seconds between firing Bullets
- fireTime has a default setting of 0.5, meaning it will fire a Bullet every half a second
- **fireTime** is a **float**, a **decimal number**
- **fireTime** is a **public** property, so it is **editable** from the **Unity Editor**

```
public float fireTime = 0.5f;
```

Explanation - isFiring property

- **isFiring** acts like a **throttle** for firing our **Bullet**
- When it is **false**, a **Bullet** can be **fired**
- When it is **true**, no **Bullets** can be **fired**
- Because of this, the **default** value **has** to be **false**
- **isFiring** is a type of bool, a **true** or **false** value
- isFiring is a **private** property, so it **CANNOT** be **edited** in the **Unity Editor**

```
private bool isFiring = false;
```

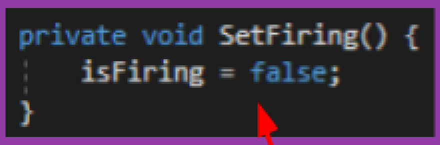
Explanation - SetFiring method

- **SetFiring** is a **custom** method, meaning we made it up for our **Weapon** Component!
- It will set the **isFiring** property to false, allowing another **Bullet** to be fired

```
private void SetFiring() {

}
```

Explanation - code breakdown



```
private void SetFiring() {
    isFiring = false;
}
```

Set isFiring to false

Explanation - Line 1

- Our isFiring property is set to false
- This will allow other methods in the **Weapon** class to fire another **Bullet**

```
private void SetFiring() {
    isFiring = false;
}
```

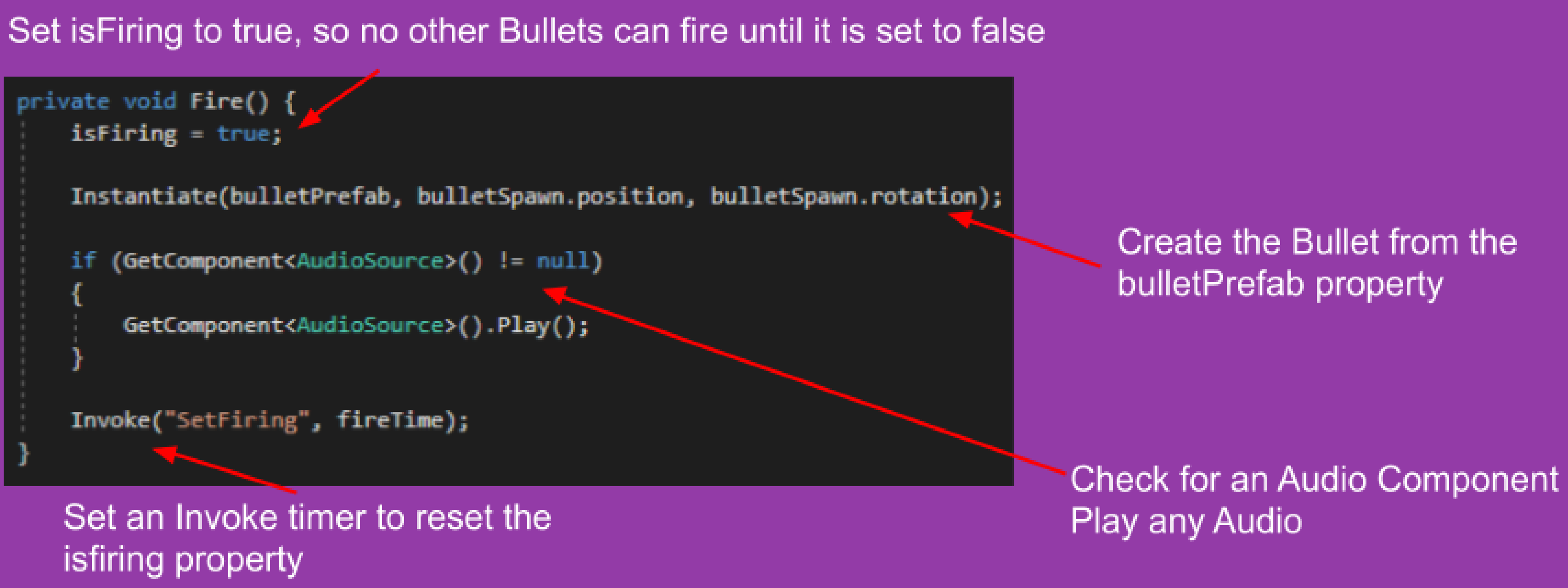
Explanation - SetFiring method

- **Fire** is a **custom** method, meaning we made it up for our **Weapon** Component!
- It will do the following:
  - Reset the isFiring ready for the next Bullet
  - Create the Bullet
  - Check for an Audio Component, Play the Audio if there is one
  - Set an Invoke timer to call the SetFiring method, using fireTime

```
private void Fire() {

}
```

Explanation - code breakdown


Set isFiring to true, so no other Bullets can fire until it is set to false
Create the Bullet from the bulletPrefab property
Check for an Audio Component Play any Audio
Set an Invoke timer to reset the isfiring property

```
private void Fire() {
    isFiring = true;

    Instantiate(bulletPrefab, bulletSpawn.position, bulletSpawn.rotation);

    if (GetComponent<AudioSource>() != null)
    {
        GetComponent<AudioSource>().Play();
    }

    Invoke("SetFiring", fireTime);
}
```

Explanation - Line 1

- We set the **isFiring** property to **true**
- This will **stop** the **gun firing** again (see the **Update** method) until isFiring is **set to false**

```
private void Fire() {
    isFiring = true;
    Instantiate( bulletPrefab, bulletSpawn.position, bulletSpawn.rotation );

    if( GetComponent<AudioSource>() != null ) {
        GetComponent<AudioSource>().Play();
    }

    Invoke( "SetFiring", fireTime );
}
```

Explanation - Line 2

- We create the Bullet GameObject here
- Using the **Instantiate** method, we **create** a new **GameObject** by **cloning** a **Prefab**
- **Instantiate** also has options about **positioning** and **rotating** the new **GameObject**
- We use the **bulletSpawn's Position** and **Rotation** to spawn the **Bullet** in the correct place
- We get the **Position** from **bulletSpawn.position**
- We get the **Rotation** from **bulletSpawn.rotation**

```
private void Fire() {
    isFiring = true;
    Instantiate( bulletPrefab, bulletSpawn.position, bulletSpawn.rotation );

    if( GetComponent<AudioSource>() != null) {
        GetComponent<AudioSource>().Play();
    }

    Invoke( "SetFiring", fireTime );
}
```

Useful links

More information about **Instantiate**	Instantiate
More information about **Transform.position**	Transform.position
More information about **Transform.rotation**	Transform.rotation

Explanation - Line 3

- Here we check if there’s an **Audio Source Component** attached

```
private void Fire() {
    isFiring = true;
    Instantiate( bulletPrefab, bulletSpawn.position, bulletSpawn.rotation );

    if( GetComponent<AudioSource>() != null) {
        GetComponent<AudioSource>().Play();
    }

    Invoke( "SetFiring", fireTime );
}
```

Useful links

- More information about the **AudioSource** Component AudioSource

Explanation - Line 4

- **Play** the **audio clip** using the Audio source’s Play() method

```
private void Fire() {
    isFiring = true;
    Instantiate( bulletPrefab, bulletSpawn.position, bulletSpawn.rotation );

    if( GetComponent<AudioSource>() != null ) {
        GetComponent<AudioSource>().Play();
    }

    Invoke( "SetFiring", fireTime );
}
```

Useful links

- More information about the **AudioSource** Component AudioSource.Play()

Explanation - Line 5

- Call **Invoke**
- The method to call is **SetFiring**, another custom method
- The timer is **fireTime**, this will control the **rate of fire**

```
private void Fire() {
    isFiring = true;
    Instantiate( bulletPrefab, bulletSpawn.position, bulletSpawn.rotation );

    if( GetComponent<AudioSource>() != null) {
        GetComponent<AudioSource>().Play();
    }

    Invoke( "SetFiring", fireTime );
}
```

Useful links

- More information about **Invoke** Invoke

Explanation - Update method

- The event function we are using is **Update**
- **Update runs constantly** while the game is running, so any code inside the method will be running constantly!
- The syntax to use the **Update** method looks like this:

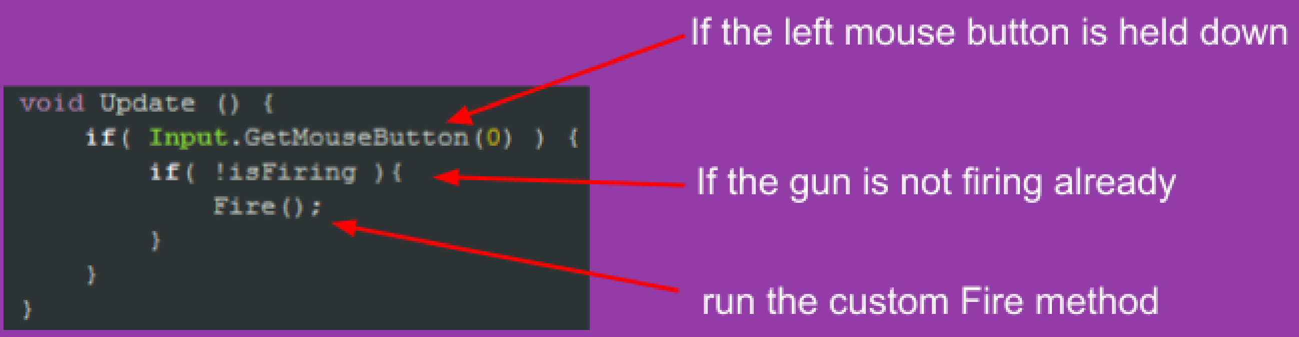
```
private void Update(){

}
```

Useful links

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

Explanation - code breakdown



```
void Update () {
    if( Input.GetMouseButton(0) ) {
        if( !isFiring ){
            Fire();
        }
    }
}
```

If the left mouse button is held down
If the gun is not firing already
run the custom Fire method

Explanation - Line 1

- First, we check if the **left mouse button** is currently **being held down** using the **GetMouseButton** method
- We specify the **left mouse button** by entering a **0** in the **GetMouseButton** method call

```
private void Update(){
    if( Input.GetMouseButton(0) ) {
        if( !isFiring ) {
            Fire();
        }
    }
}
```

Useful links

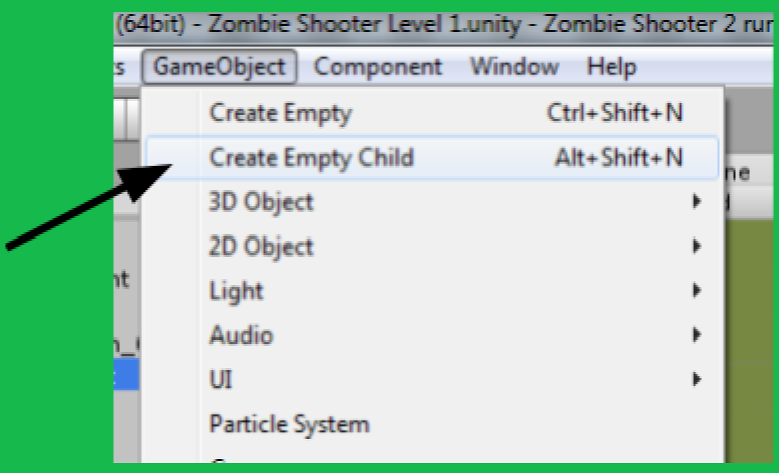
- More information about **Input.GetMouseButton** [Input.GetMouseButton]

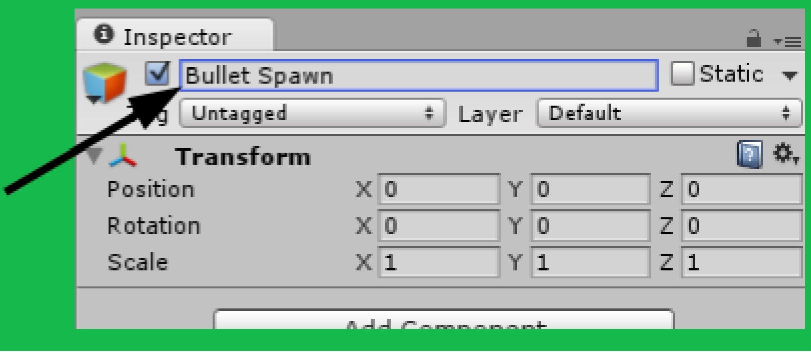
Do this

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

Do this

- Create an empty **GameObject**
- **Top Menu** : **GameObject** > **Create Empty Child**
- Name the empty GameObject **Bullet Spawn**


(64bit) - Zombie Shooter Level 1.unity - Zombie Shooter 2 run
GameObject
Component
Window
Help
Create Empty Ctrl+Shift+N
Create Empty Child Alt+Shift+N
3D Object
2D Object
Light
Audio
UI
Particle System


Inspector

Bullet Spawn ☐ Static

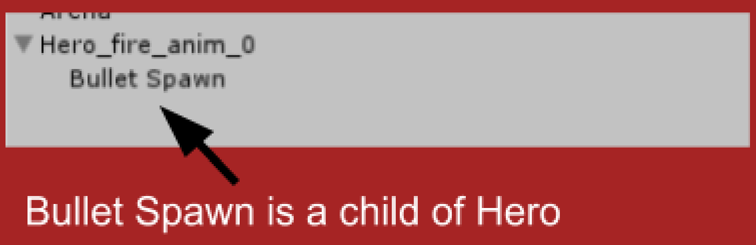
Tag Untagged Layer Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

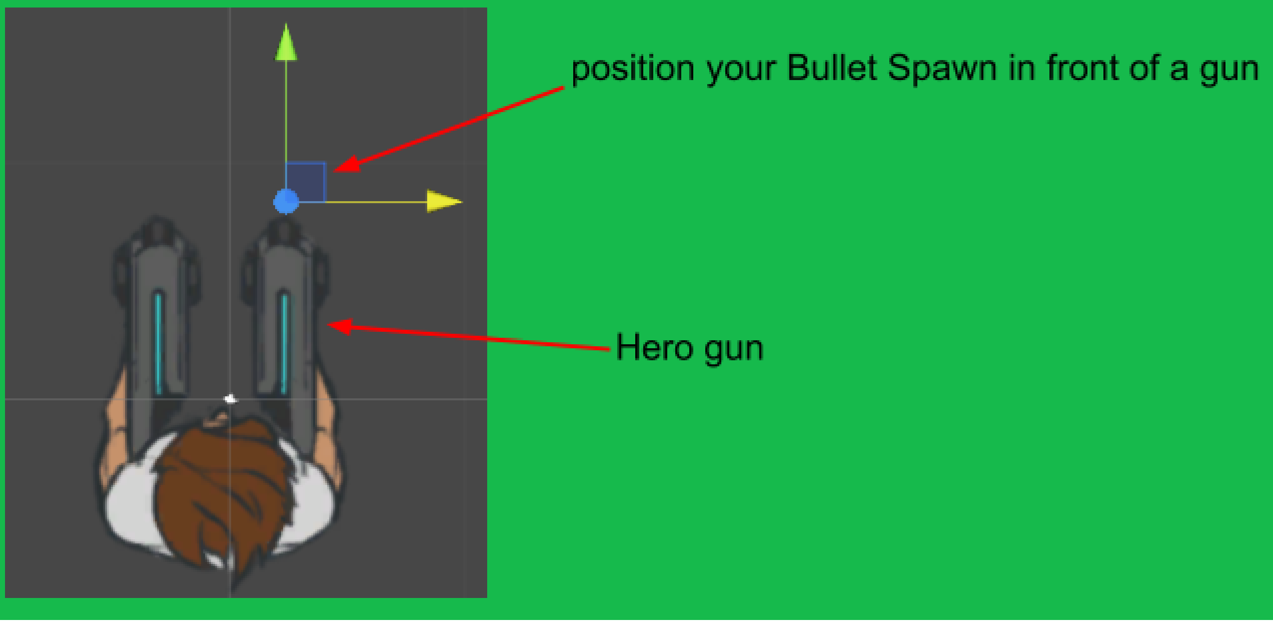
Check this

- Your Bullet Spawn GameObject is parented to the Hero GameObject



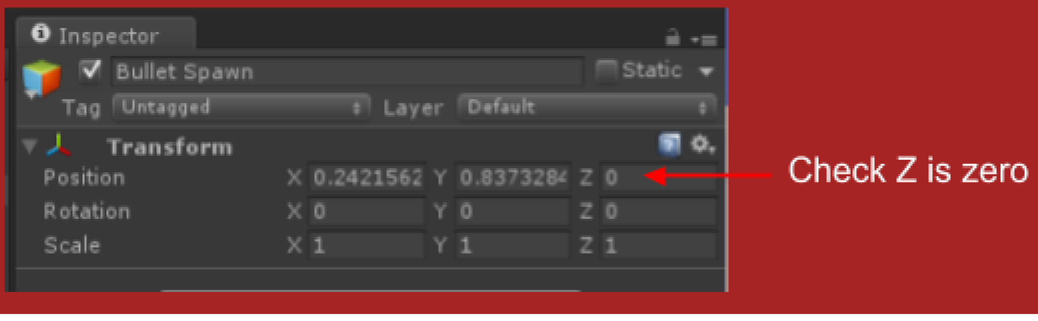
Do this

- In the **Scene view**, **position** the **Bullet Spawn** in front of one of the **Hero's guns**



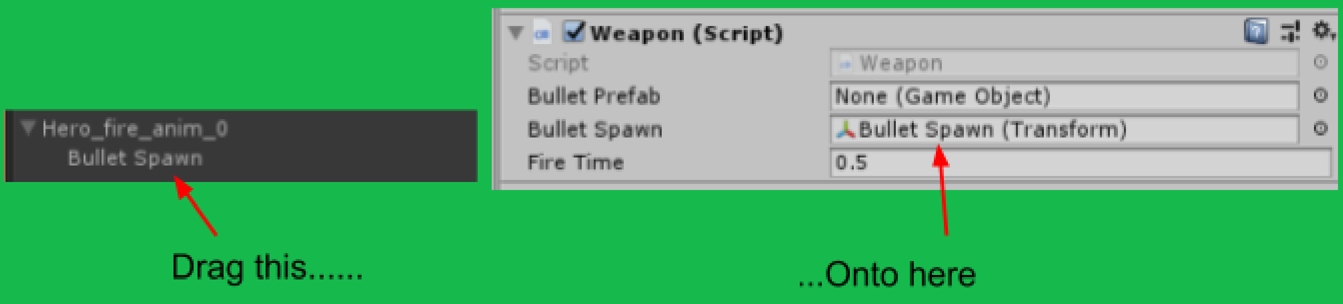
Check this

- On the **Bullet Spawn**, check the **Transform** Component
- The position's **Z axis** should be **zero**



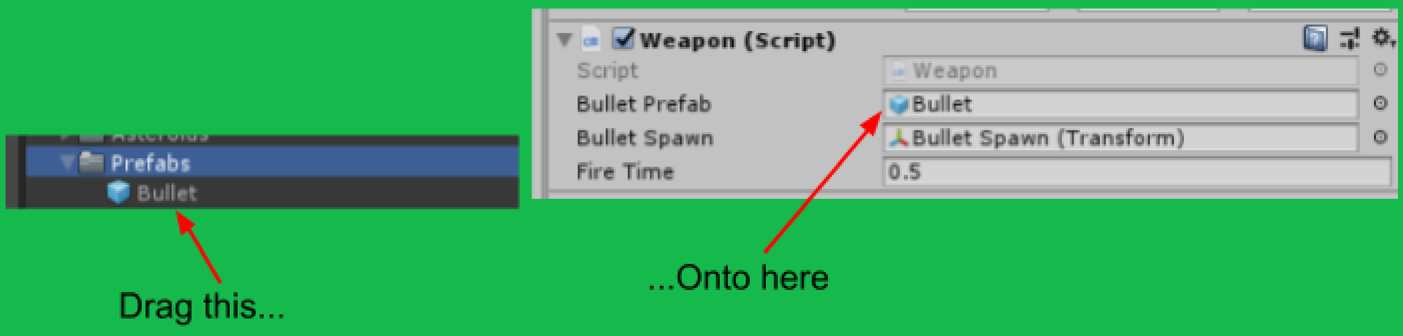
Do this

- Select the **Hero** GameObject in the **Hierarchy**
- **Drag** the **Bullet Spawn** GameObject into the **Bullet Spawn inlet** on the **Weapon**Component



Do this

- Select the **Hero** GameObject in the **Hierarchy**
- **Drag** the **Bullet Prefab** from the **Project view** into the **Bullet Prefab inlet** on the **Weapon** Component

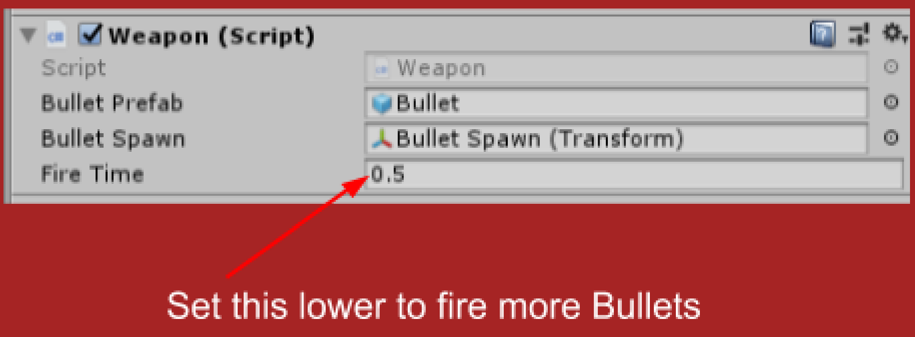


Do this

- Test the Game!
- Check the **Bullets** fire from the **Heros** gun

Check this

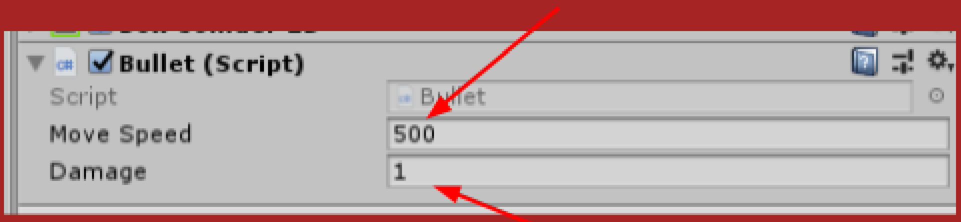
- If you want more **Bullets** to fire, set the **fireTime** to a **lower value**
  - **fireTime** is the amount of **Bullets** fired per second
- **Weapon** should be attached to the Hero GameObject


Weapon (Script)
Script Weapon
Bullet Prefab Bullet
Bullet Spawn Bullet Spawn (Transform)
Fire Time 0.5
Set this lower to fire more Bullets

Check this

- If you want your **Bullets** to **move faster**, set **speed** to a **higher value**
- **Bullet2D** should be attached to the **Bullet Prefab** in the **Project view**

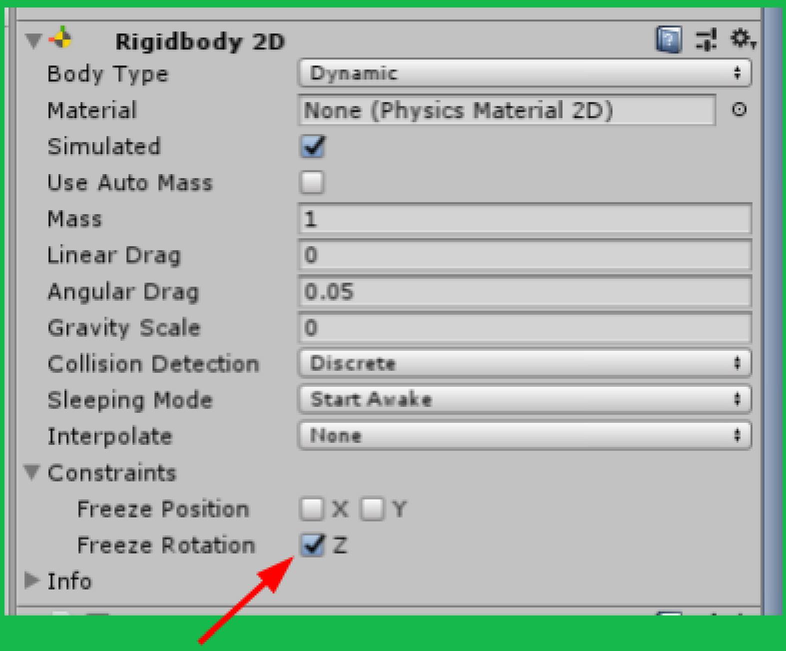
Set speed higher for faster Bullets


Bullet (Script)
Script: Bullet
Move Speed: 500
Damage: 1

Set this higher for more damage

Do this

- In the **Unity Editor**, select the **Hero** script in the **Hierarchy**
- On the **Rigidbody2D** component, open the **Constraints** section by clicking on the small arrow by the word constraints
- Tick the box next to **Freeze Rotation**

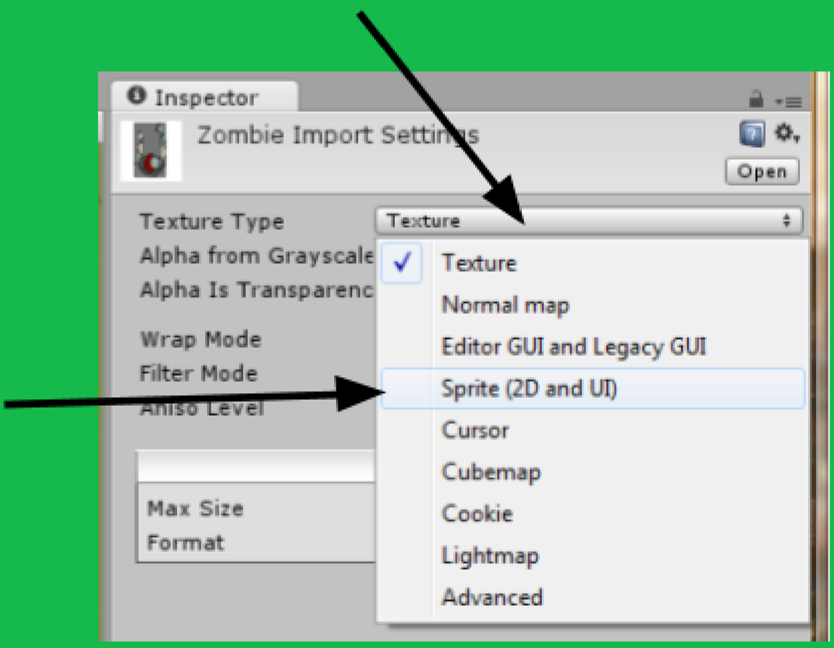

Rigidbody 2D
Body Type: Dynamic
Material: None (Physics Material 2D)
Simulated: ✓
Use Auto Mass: ☐
Mass: 1
Linear Drag: 0
Angular Drag: 0.05
Gravity Scale: 0
Collision Detection: Discrete
Sleeping Mode: Start Awake
Interpolate: None
Constraints
Freeze Position: ☐ X ☐ Y
Freeze Rotation: ✓ Z
Info

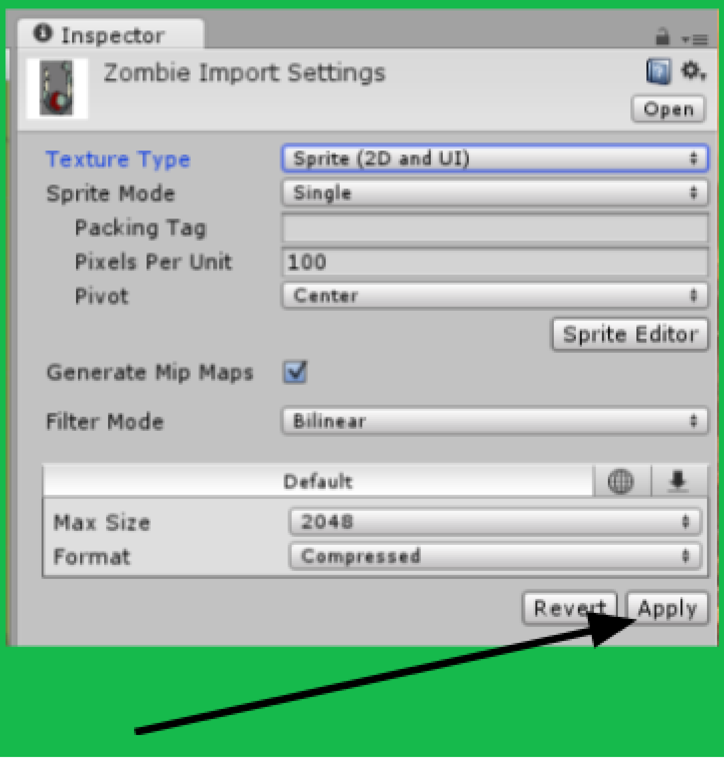
Tick this box

Task 2. Create a Zombie

Do this

- In the **Sprites folder** of the **Project view**, select the **Zombie** artwork
- In the **Inspector**, Set the **Texture Type** to **Sprite**
- Click **Apply**

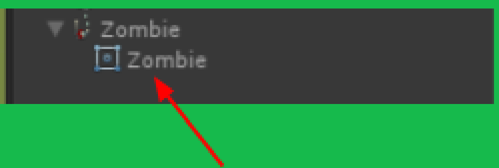



Inspector
Zombie Import Settings

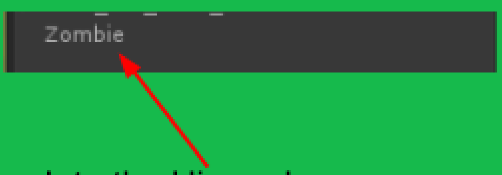
Texture Type: Sprite (2D and UI)
Sprite Mode: Single
Packing Tag:
Pixels Per Unit: 100
Pivot: Center
Generate Mip Maps: ✓
Filter Mode: Bilinear
Default
Max Size: 2048
Format: Compressed

Do this

- Select the **Zombie** Artwork in the **Project view**
- **Drag** it onto the **Hierarchy** to create a **Zombie** GameObject



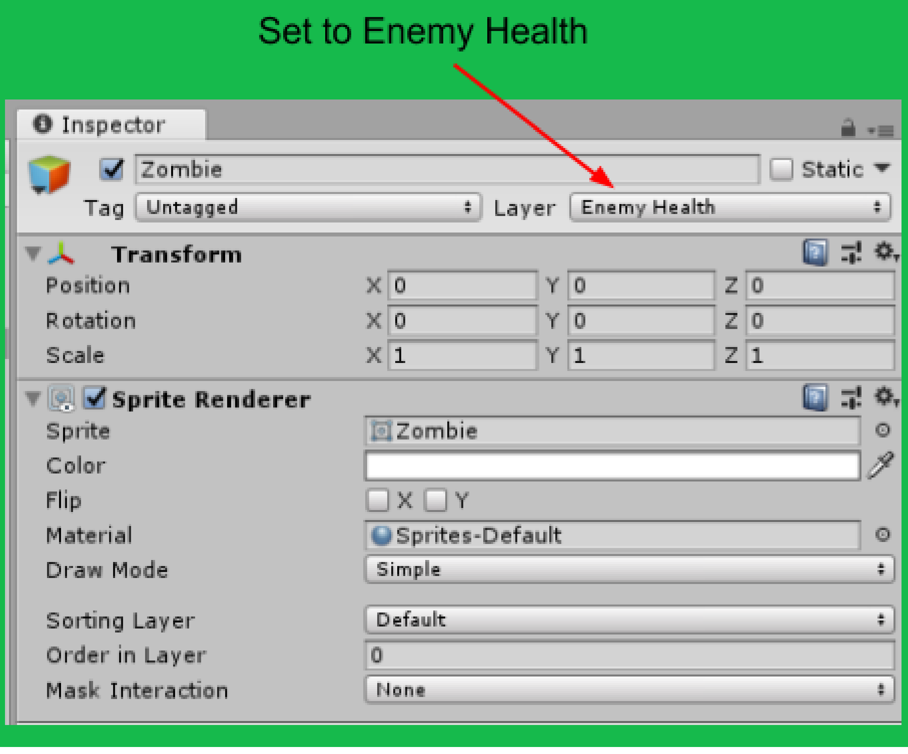
Drag this from the Project view....



...Into the Hierarchy

Do this

- Select the **Zombie** GameObject in the **Hierarchy**
- In the **Inspector**, set the **Layer** on the GameObject to **Enemy Health**



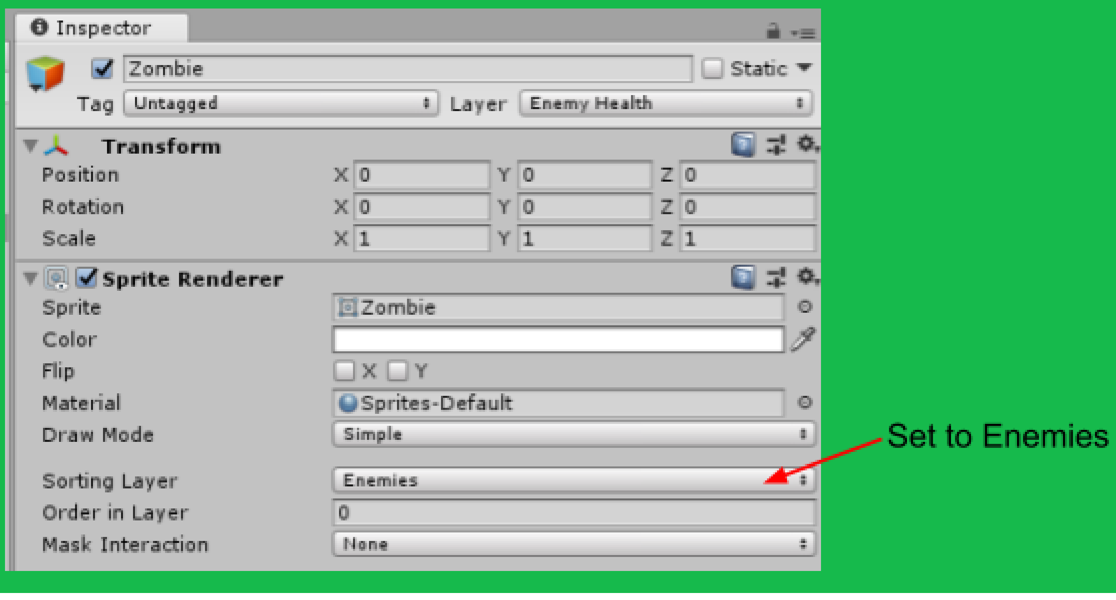
Zombie — Static
Tag: Untagged — Layer: Enemy Health

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Sprite Renderer
Sprite: Zombie
Color
Flip: X Y
Material: Sprites-Default
Draw Mode: Simple
Sorting Layer: Default
Order in Layer: 0
Mask Interaction: None

Do this

- Select the **Zombie** GameObject in the **Hierarchy**
- In the **Inspector**, set the **Sorting Layer** on the **Sprite Renderer** to **Enemies**



Zombie — Static
Tag: Untagged — Layer: Enemy Health

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

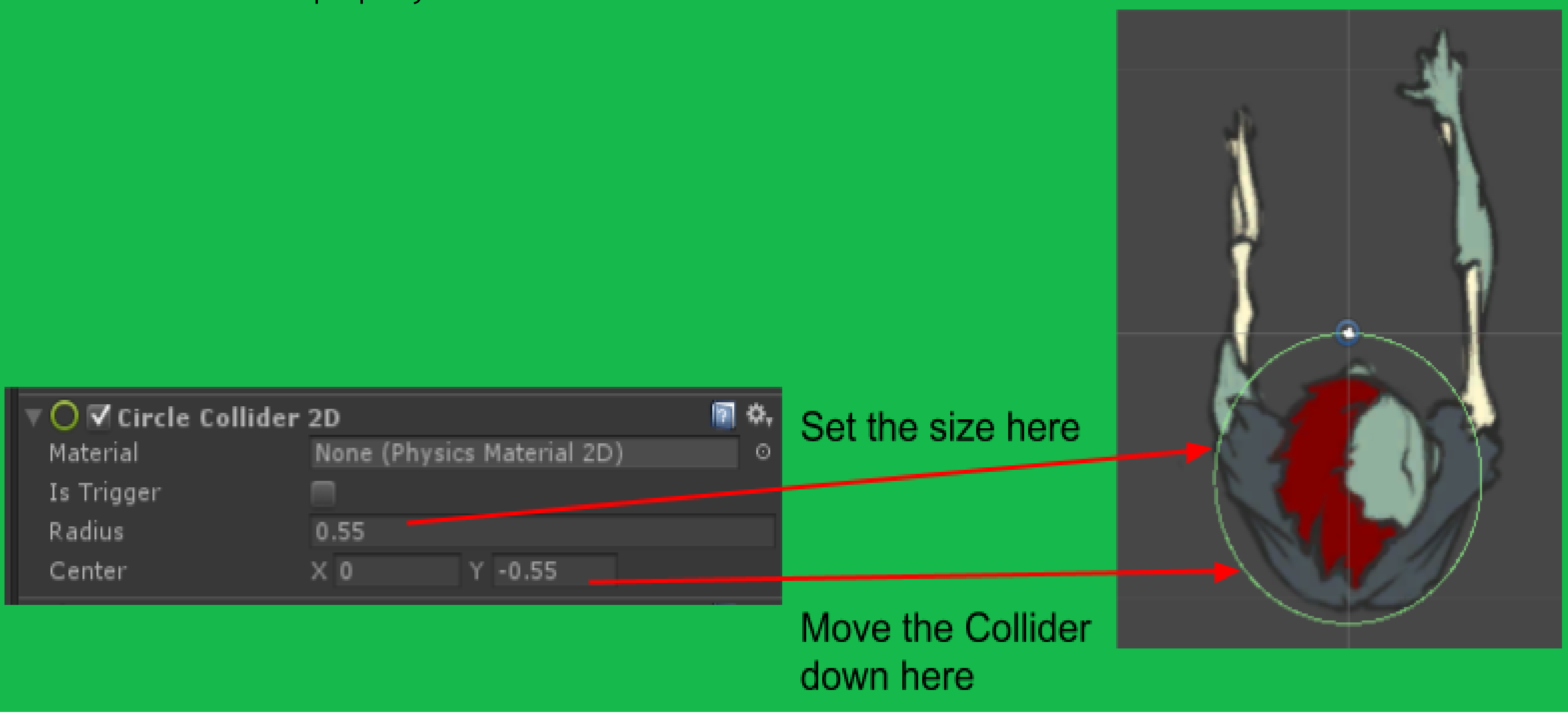
Sprite Renderer
Sprite: Zombie
Color
Flip: X Y
Material: Sprites-Default
Draw Mode: Simple
Sorting Layer: Enemies
Order in Layer: 0
Mask Interaction: None

Do this

- Using the **Add Component** button, add a **Circle Collider 2D** to the **Zombie**
- **Add Component** > **Physics 2D** > **Circle Collider 2D**

Do this

- Set the **Radius** property of the **Circle Collider 2D** to cover the Zombies head and body
- Set the **Y** value of the **Center** property to the middle of the Zombies head



Circle Collider 2D
Material: None (Physics Material 2D)
Is Trigger
Radius: 0.55
Center: X 0 Y -0.55

INTERACTIVE
SYSTEMS STUDIO

CGD