Zombie Shooter Project 3d

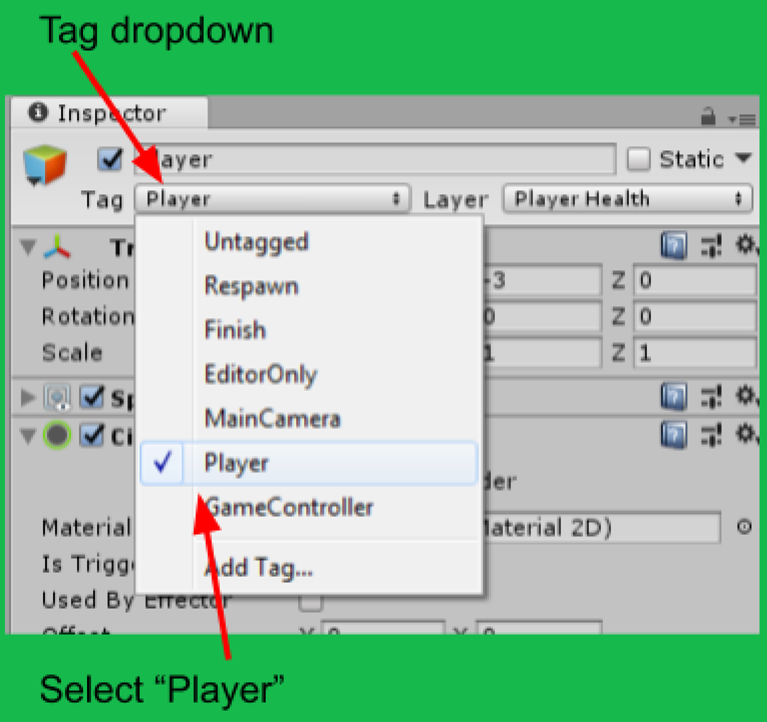
Task 1. Add a "Player" tag to the player GameObject in the scene

Explanation

- We want to identify the player's GameObject so zombies can look at and chase the player when they spawn
- In prefab form, zombies or any other GameObject won't "know" about other GameObjects, until they are spawned into the scene
- We can use a tag to identify a GameObject we want to find easily in the editor

Do this

- Select the **Hero** (or player) **GameObject** in the **Hierarchy**
- In the **Inspector** click the **Tags** dropdown
- Select **Player**



Useful links

- More information about **Tags** [Tags - Manual]

Task 2. Get a reference to the player when the Zombie spawns

Explanation

- We want our **zombie** to **find** the **player** in the **scene** by its **tag** so we can:
  - Follow the player
  - Face the player
- We have 2 components that do this - **MoveTowardsTarget** and **SmoothLookAtTarget2D**
- We need to give those components the player **Transform** when the zombie (or any enemy) spawns

Do this

- In the **Scripts** folder of the **Project view**, create a new script
- Name the script **Enemy**
- Drag the **Enemy** script onto the **Zombie** Gameobject in the **Hierarchy**

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 EnemySpawnedEvent : UnityEvent<Transform> { }

public class Enemy : MonoBehaviour {

    public EnemySpawnedEvent onSpawn;

    private void Start () {
        GameObject player = GameObject.FindWithTag("Player");
        onSpawn.Invoke(player.transform);
    }
}
```

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 - EnemySpawnedEvent class

- We create a custom event as a class.
- Our custom event is called **EnemySpawnedEvent**
- The class inherits from **UnityEvent**
- NOTE: the **EnemySpawnedEvent** has a **Transform** parameter in the **<>** brackets
- NOTE: we use **[System.Serializable]** to tell the unity editor we want to configure the event in the editor
- When we call the **EnemySpawnedEvent** using the Invoke method, we need to give it a **Transform** component

```
[System.Serializable]
public class EnemySpawnedEvent : UnityEvent<Transform> { }
```

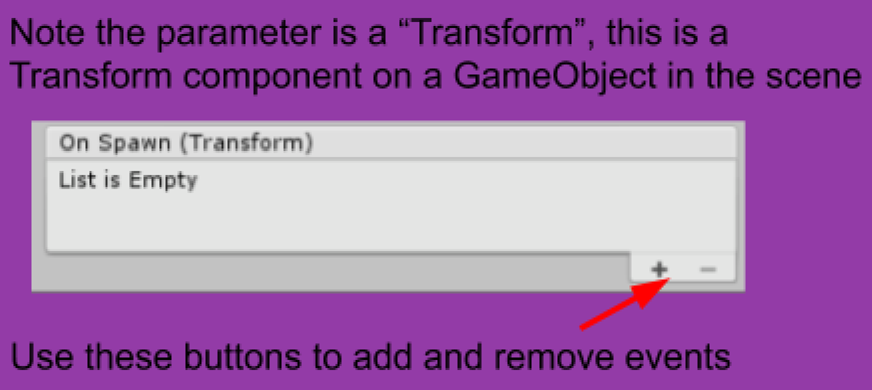
Useful links

More information about **Custom UnityEvents**	Custom UnityEvents - Scripting Reference
More information about **System.Serializable**	System.Serializable - Scripting reference

Explanation - onSpawn property

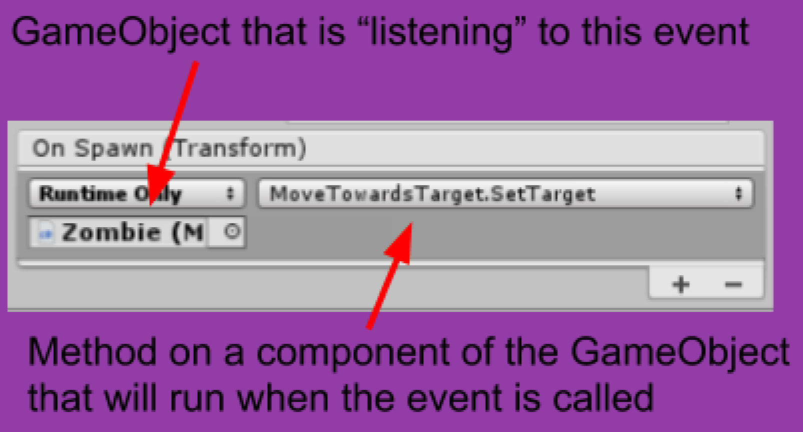
- **onSpawn** is a CUSTOM UnityEvent
- Custom UnityEvents can pass information, like a transform component
- NOTE: other value types can be used as the parameter - string, float, bool etc

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


Note the parameter is a "Transform", this is a Transform component on a GameObject in the scene

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!)

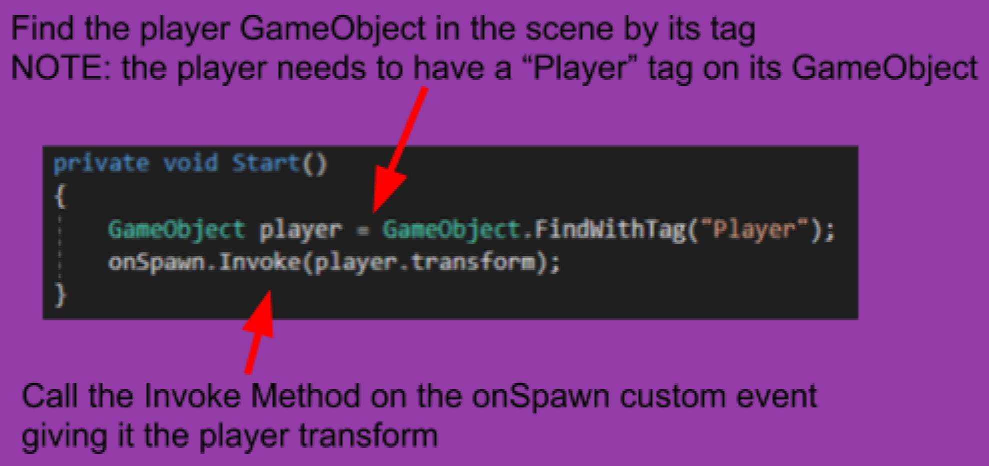

GameObject that is "listening" to this event

Method on a component of the GameObject that will run when the event is called

NOTE: the method called will have a Transform as a parameter!

```
public EnemySpawnedEvent onSpawn;
```

Explanation - Our Start method


Find the player GameObject in the scene by its tag
NOTE: the player needs to have a "Player" tag on its GameObject

```
private void Start()
{
    GameObject player = GameObject.FindWithTag("Player");
    onSpawn.Invoke(player.transform);
}
```

Call the Invoke Method on the onSpawn custom event giving it the player transform

Explanation - Line 1

- We create a **GameObject** variable to store our **player** from the scene
- **<u>NOTE: the player requires a "Player" tag on its GameObject in the scene!</u>**
- **<u>NOTE: make sure ONLY ONE GameObject in your scene has a "Player" tag on it!</u>**
- Use GameObject.FindWithTag to find the player by its tag

```
private void Start () {
    GameObject player = GameObject.FindWithTag("Player");
    onSpawn.Invoke(player.transform);
}
```

Useful links

- More information about **GameObject.FindWithTag** GameObject.FindWithTag - Scripting Reference

Explanation - Line 2

- Now we have the **player** GameObject in the scene, we give it to the **onSpawn** event
- We call the onSpawn event using Invoke and give it the Transform component of the player
- NOTE: the Transform component deals with the position of the player, which is what we are interested in to follow and look at
- Our custom event's **Invoke** method takes a **Transform** parameter, meaning we need the **Transform component** of the **player**
- We can get the player Transform component using **player.transform**

```
private void Start () {
    GameObject player = GameObject.FindWithTag("Player");
    onSpawn.Invoke(player.transform);
}
```

Useful links

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

Do this

- Select the **Zombie** in the **Hierarchy**
- Add the **Enemy** script to the **Zombie** in the **Inspector**

Task 3. Add a SetTarget method to the MoveTowardsTarget script

Explanation

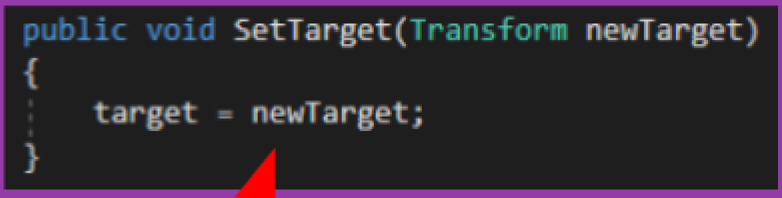
- We want to set the target for our **MoveTowards** component on the **Zombie** when it spawns
- We can add a public method called **SetTarget** to the script so the target can be set by another component (the Enemy component in this case)
- **SetTarget** will be called by the **onSpawn** event when the zombie spawns if we set the event up to do so in the editor

Do this

- **Open** the **MoveTowardsTarget** script in Visual Studio
- Add the following **HIGHLIGHTED** code

```
using UnityEngine;

public class MoveTowardsObject : MonoBehaviour {

    public Transform target;
    public float speed = 5.0f;

    private void Update() {
        if( target != null ) {
            transform.position = Vector3.MoveTowards( transform.position, target.position, speed * 0.01f );
        }
    }

   public void SetTarget(Transform newTarget) {
        target = newTarget;
   }
}
```

Explanation - Our custom SetTarget method



```
public void SetTarget(Transform newTarget)
{
    target = newTarget;
}
```

Set the public variable, target to the value of the parameter, newTarget

---

Task 4. Add a SetTarget method to the SmoothLookAtTarget2D script

Explanation

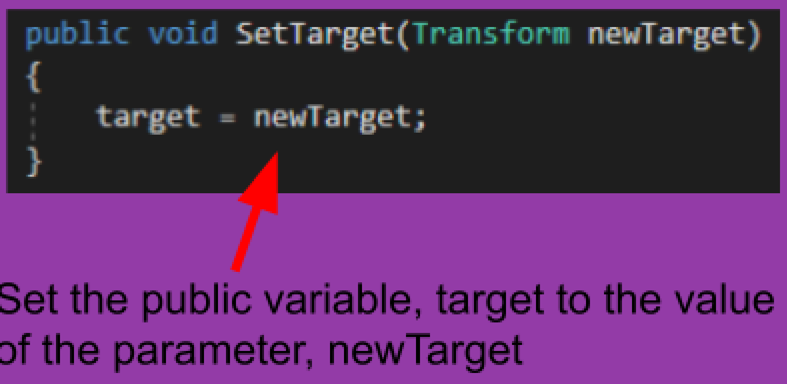
- We want to set the target for our **SmoothLookAtTarget2D** component on the **Zombie** when it spawns
- We can add a public method called **SetTarget** to the script so the target can be set by another component (the Enemy component in this case)
- **SetTarget** will be called by the **onSpawn** event when the zombie spawns if we set the event up to do so in the editor

Do this

- **Open** the **SmoothLookAtTarget2D** script in Visual Studio
- Add the following **HIGHLIGHTED** code

```
using UnityEngine;

public class SmoothLookAtTarget2D : MonoBehaviour {

    public Transform target;
    public float smoothing = 5.0f;
    public float adjustmentAngle = 0.0f;

    private void Update() {
        if( target != null ) {

            Vector3 difference = target.position - transform.position;

            float rotZ = Mathf.Atan2( difference.y, difference.x ) * Mathf.Rad2Deg;

            Quaternion newRot = Quaternion.Euler( new Vector3( 0.0f, 0.0f, rotZ + adjustmentAngle ));

            transform.rotation = Quaternion.Lerp( transform.rotation, newRot, Time.deltaTime * smoothing );
        }
    }

   public void SetTarget(Transform newTarget) {
        target = newTarget;
   }
}
```

Explanation - Our custom SetTarget method



```
public void SetTarget(Transform newTarget)
{
    target = newTarget;
}
```

Set the public variable, target to the value of the parameter, newTarget

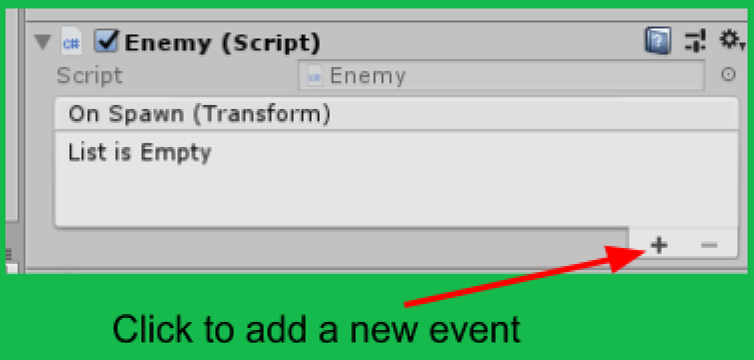
Task 5. Add MoveTowardsTarget and SmoothLookAtTarget as listeners to the onSpawn event

Explanation

- Our zombie has 2 components which need to know about the player Transform so they can move towards and look at it
- We can set each components public variable, **target** using their public **SetTarget** methods and the **Enemy onSpawn** event in the editor

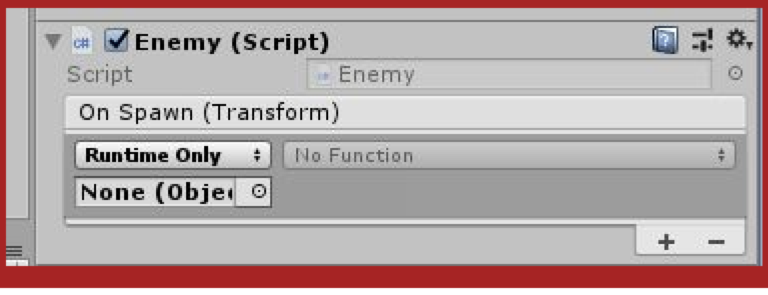
Do this

- Select the **Zombie** in the **Hierarchy**
- In the **Inspector**, on the **Enemy** component, click the “**+**" button on the **On Spawn** event



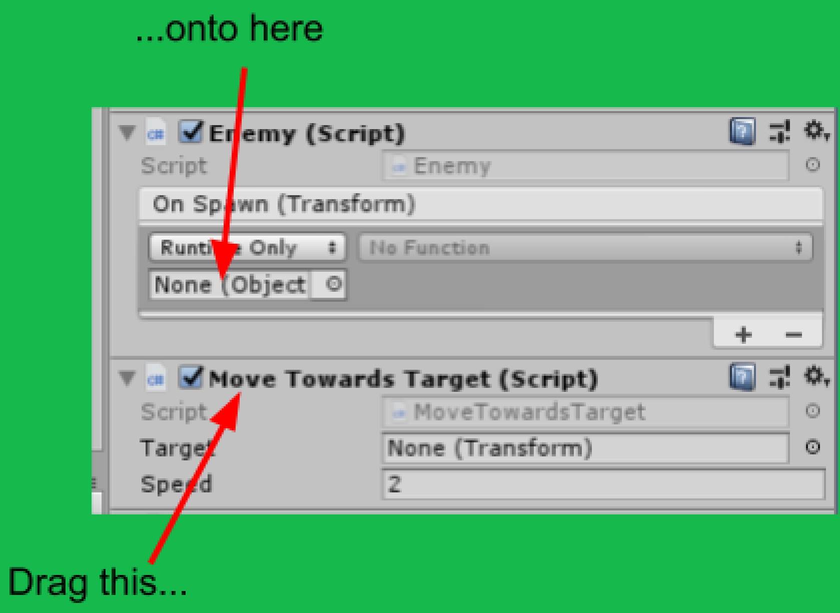
Check this

- Your Enemy component on the zombie GameObject should look like this
- The On Spawn event should have an empty listener



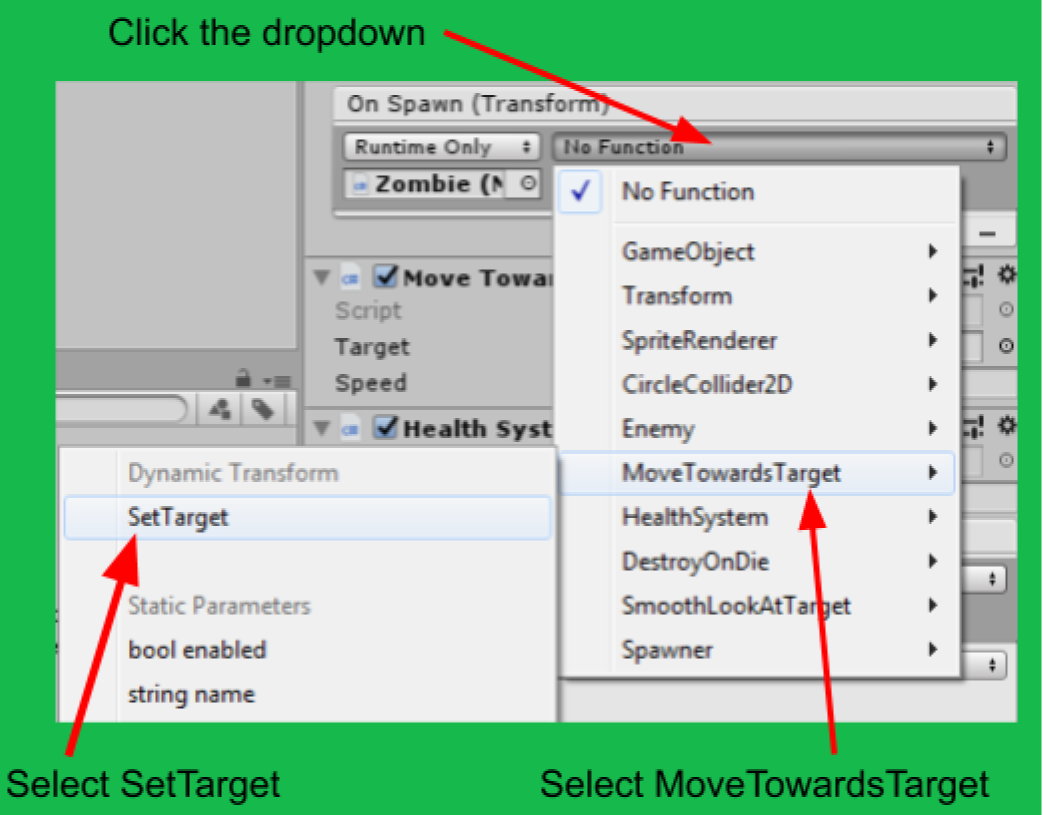
Do this

- Drag the **MoveTowardsTarget** <u>component</u> onto the empty inlet on the **On Spawn** event



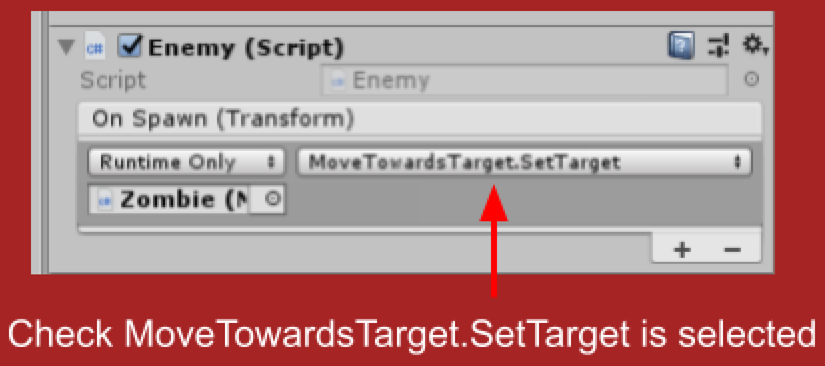
Do this

- Select the **SetTarget** method from the **MoveTowardsTarget** component on the event dropdown menu



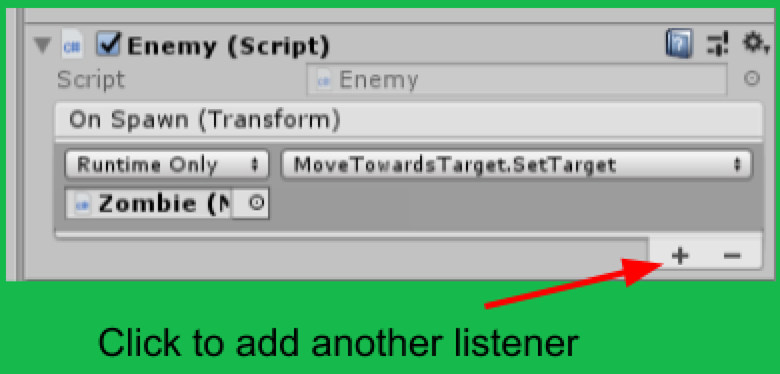
Check this

- Your Enemy component on the zombie GameObject should look like this
- The On Spawn event should have a listener setup for **MoveTowardsTarget.SetTarget**


Enemy (Script)
Script
Enemy
On Spawn (Transform)
Runtime Only
MoveTowardsTarget.SetTarget
Zombie
Check MoveTowardsTarget.SetTarget is selected

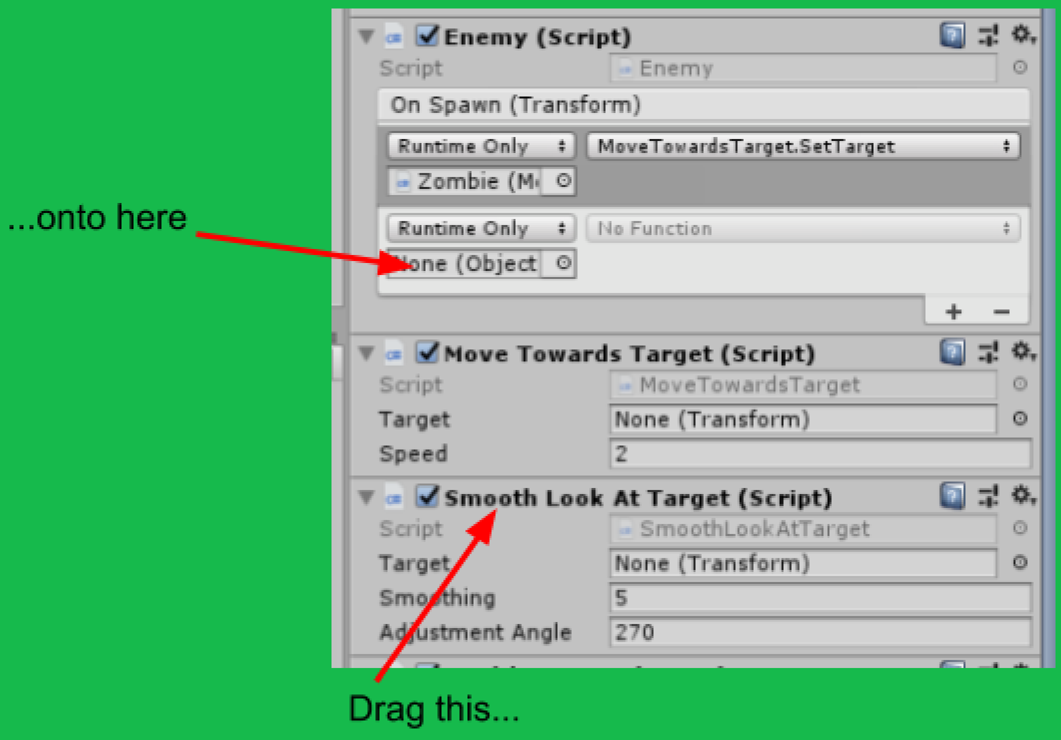
Do this

- In the **Inspector**, on the **Enemy** component, click the “**+**" button on the **On Spawn** event



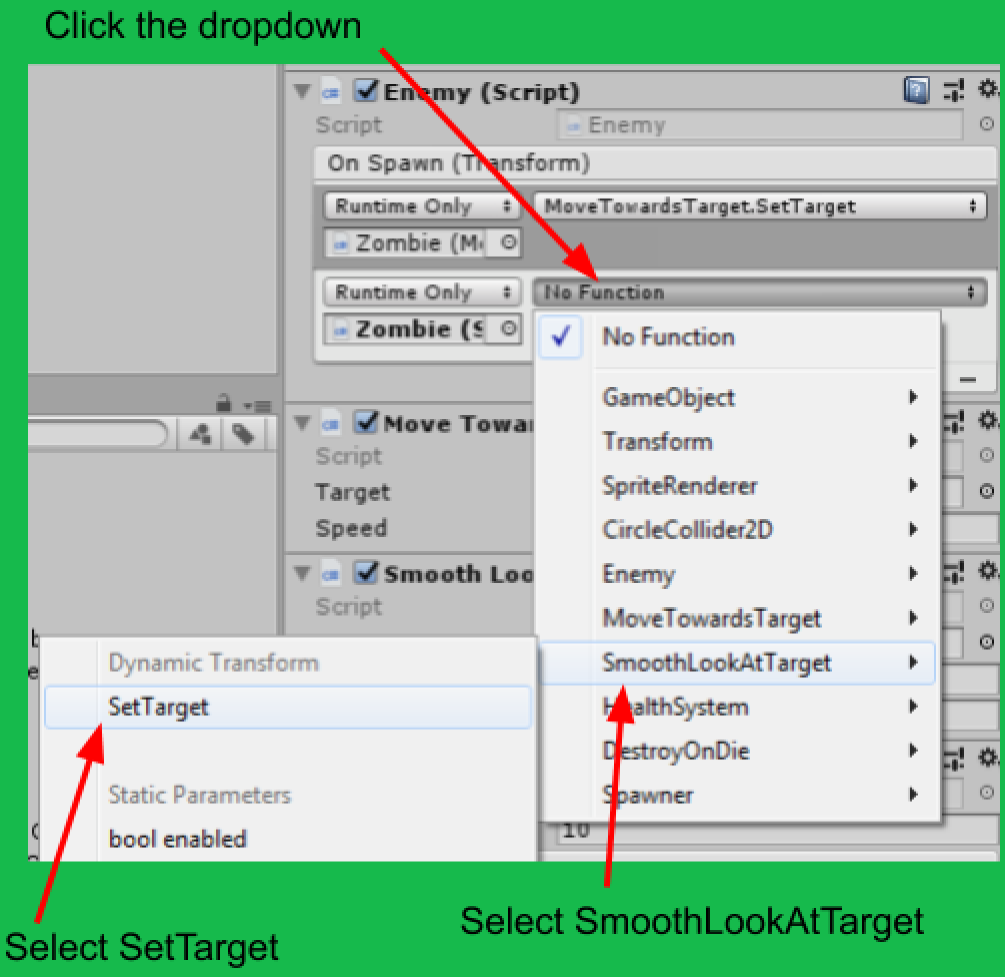
Do this

- Drag the **SmoothLookAtTarget** component onto the empty inlet on the **On Spawn** event



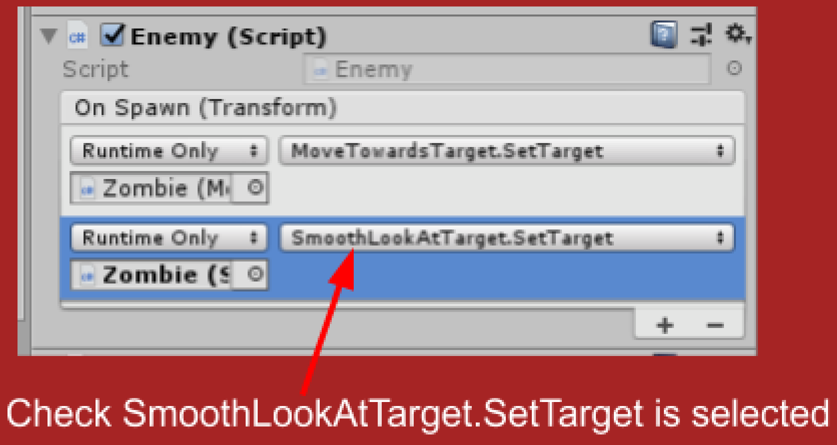
Do this

- Select the **SetTarget** method from the **MoveTowardsTarget** component on the event dropdown menu



Check this

- Your Enemy component on the zombie GameObject should look like this
- The On Spawn event should have a listener setup for **SmoothLookAtTarget.SetTarget**


Enemy (Script)
Script
Enemy
On Spawn (Transform)
Runtime Only
MoveTowardsTarget.SetTarget
Zombie (M
Runtime Only
SmoothLookAtTarget.SetTarget
Zombie (S
Check SmoothLookAtTarget.SetTarget is selected

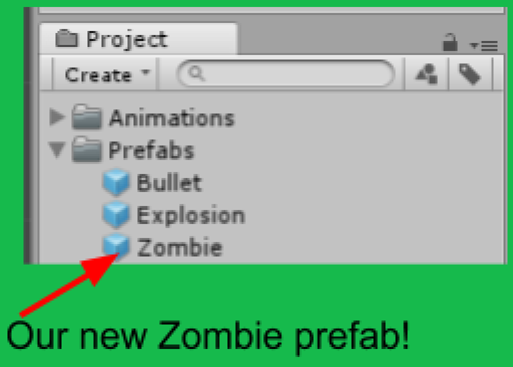
Task 6. Create a Prefab from the Zombie GameObject

Explanation

- Now our zombie will automatically find the player when it spawns, we can create a prefab of the zombie and spawn as many as we want from the prefab

Do this

- Select the **Zombie** in the **Hierarchy**
- Drag the **Zombie** into the **Prefabs** folder in the **Project view**


Project
Create
Animations
Prefabs
Bullet
Explosion
Zombie
Our new Zombie prefab!