Using the Pool Manager with bullets (OPTIONAL) AINT152

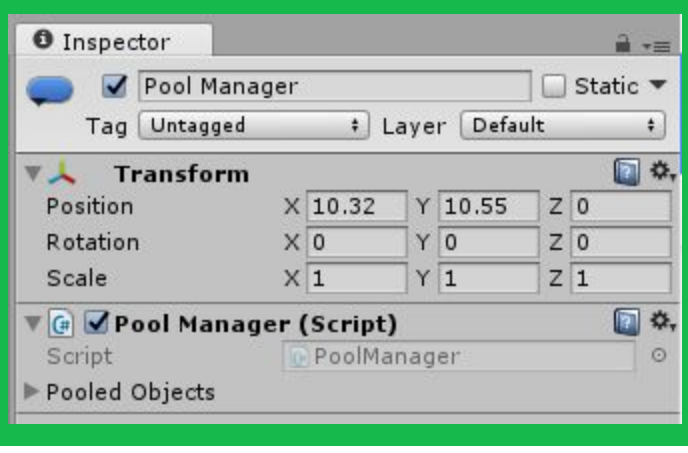
Task 1. Setup the Pool Manager

Explanation

- To use the Pool Manager with our bullets we need to add one to the **Scene**
- Make sure you only add **ONE** Pool Manager to a Scene!
- We can pool as many different **GameObjects** as we want using this one Pool Manager
- Copy the Script from the **Pool Manager Script breakdown** on the **DLE**

Do this

- In the **Scene,** create an empty GameObject
- Name it **Pool Manager**
- Add the **PoolManager** Script to it in the Inspector



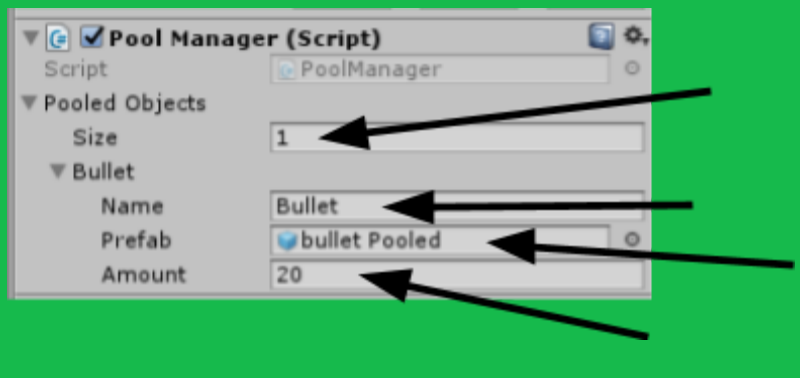
Task 2. Add the bullet prefab to the Pool Manager

Explanation

- We now need to add our Bullet prefab and tell the Pool Manager how many Bullets we need at runtime

Do this

- In the **Inspector**, type 1 in the **Size** section
- Type **Bullet** in the **Name** Section
  - You will use the **Name** to get a bullet from other scripts
- Add your Bullet Prefab to the **Prefab** section
  - The **Prefab** is the GameObject you want to spawn from your pool
- Type 20 in the **Amount** section
  - The **Amount** is the total amount of bullets allowed to spawn



Task 3. Edit the Bullet2D script to run every time it is enabled

Explanation

- We need the **Bullet** to run code every time it is enabled
- The **Start** method only runs once when the **Bullet** is created
- **OnEnable** will run every time the **Bullet** is enabled

Do this

- Open the **Bullet2D** script
- Change **void Start()**
- To **void OnEnable()**

This will run our Invoke method every time the bullet enables

Change this...

```
void Start()
{
    Invoke("Die", destroyTime);
}
```

...to this

```
void OnEnable()
{
    Invoke("Die", destroyTime);
}
```

Do this

- Change **Destroy(gameObject);**
- To **gameObject.SetActive(false);**

This will disable our GameObject instead of destroying it

Change this...

```
void Die()
{
    Destroy(gameObject);
}
```

...to this

```
void Die()
{
    gameObject.SetActive(false);
}
```

Do this

- Change **void OnDestroy()**
- To **void OnDisable()**

This will Cancel out Invoke every time the GameObject is disabled, instead of when it is destroyed

Change this...

```
void OnDestroy()
{
    CancelInvoke("Die");
}
```

...to this

```
void OnDisable()
{
    CancelInvoke("Die");
}
```

Task 4. Edit the BulletHit2D script to disable the GameObject when triggered

Explanation

- When the **Bullet** hits something and the **OnTriggerEnter2D** runs, we want the **Bullet** to disable instead of destroying itself

Do this

- Open the **BulletHit2D** script
- Change **Destroy(gameObject);**
- To **gameObject.SetActive(false);**

This will disable the GameObject instead of destroying it

```
void OnTriggerEnter2D(Collider2D other)
{
    if (other.CompareTag(damageTag))
    {
        other.SendMessage("TakeDamage", damage);
    }

    Destroy(gameObject);
}
```

Change this...

```
void OnTriggerEnter2D(Collider2D other)
{
    if (other.CompareTag(damageTag))
    {
        other.SendMessage("TakeDamage", damage);
    }

    gameObject.SetActive(false);
}
```

...to this

Task 5. Edit the ShootBullet script to use the PoolManager

Explanation

- We can now use the PoolManager to get our bullet
- First, we can remove our bulletPrefab, because the GameObject will come from the PoolManager

Do this

- Open the **ShootBullet** script
- Remove **public GameObject bulletPrefab;**

We don't need this property now, so delete it

Change this...

```
public class ShootBullet : MonoBehaviour {

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

...to this

```
public class ShootBullet : MonoBehaviour {

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

Explanation

- Now we will set up the Fire method to get a Bullet from the PoolManager, position it, rotate it and enable it!

Do this

- Remove **Instantiate(bulletPrefab, bulletSpawn.position, bulletSpawn.rotation);**

We aren't creating a new GameObject now, so we don't need this code

Remove this line!

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

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

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

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

Do this

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

    GameObject bullet = PoolManager.current.GetPooledObject("Bullet");
    if (bullet != null)
    {
        bullet.transform.position = bulletSpawn.position;
        bullet.transform.rotation = bulletSpawn.rotation;
        bullet.SetActive(true);
    }

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

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

Add all this code

Explanation - What the code in the Fire method does

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

    GameObject bullet = PoolManager.current.GetPooledObject("Bullet");
    if (bullet != null)
    {
        bullet.transform.position = bulletSpawn.position;
        bullet.transform.rotation = bulletSpawn.rotation;
        bullet.SetActive(true);
    }

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

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

Get the Bullet GameObject from the PoolManager

Set position

Set rotation

Enable the Bullet