**Instantiate at Runtime**

**Objective:** In this lesson students are going to learn how to instantiate at runtime, make some more things explode, and we’re going to make sure that GameObjects don’t stick around after being destroyed and use up resources.

Now the reason we can’t apply the same plan to enemies that we did to our PlayerShip is because we know when the Player crashes the game restarts so there’s no leftover mess to deal with. Also when our Player dies our Particle System is triggered but if we use the Destroy() method on the Enemies their Particle System won’t play.

1. In the game lets duplicate an enemy.

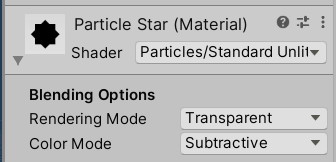


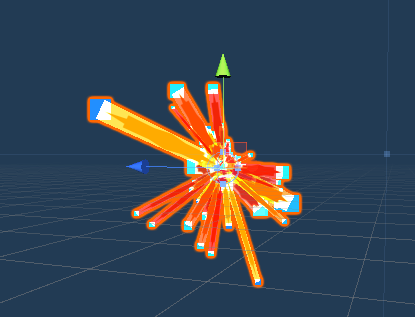
1. Rename the **Explosion** Particle System in your **Prefabs** folder to **Player Explosion VFX** and make a copy out of it. Name this new copy **Enemy Explosion VFX**
2. Double click the **Enemy Explosion VFX** to open the prefab.
3. Reset its position and look for it.
4. Click on start color and lets change it to something foreign. We don’t want it to look at all similar to the Player Ship Explosion.



Figure 1Left is the Player Explosion. Right is the new Enemy Explosion

1. I changed the Particle System to Particles> Standard Unlit
2. Then changes Rendering Mode to Transparent and Color Mode to Subtractive

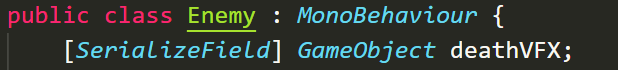




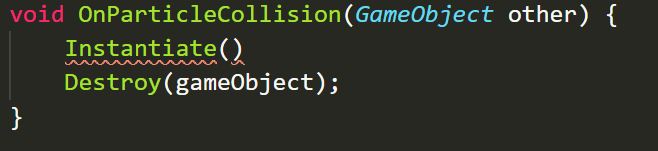
1. Leave the Prefab Edit mode

We’re not going to add the Enemy Explosion prefab to every enemy bc that’s going to get tedious.

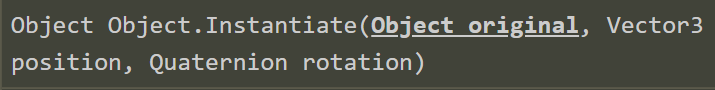
1. Open your **Enemy.cs**
2. Delete the Debug line
3. Create a **[SerializeField]** of type **GameObject** called **deathVFX**



1. Go to Unity and highlight all the Enemy ships in the Hierarchy and if you add the **Enemy Explosion particles** to the **Death VFX** field in the Inspector it will add it to all of them
2. Go back to Enemy.cs
3. Type Instantiate() above Destroy()



Instantiate takes 3 arguments

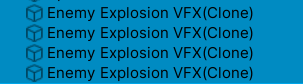


The Object, the location, and the rotation

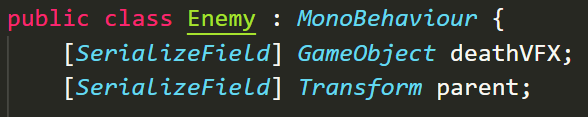
1. The Object is deathVFX, the location is wherever the enemy is, and then we don’t need any rotation.



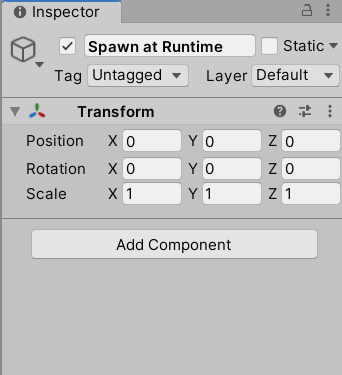
1. Find the **Enemy Explosion VFX** prefab, open its Prefab Editor, and **turn on Play on Awake**
2. Play test the game
3. Lets play test again this time pause it after destroying some enemies. You should notice that we’re making a lot of clones for the Enemy Explosion VFX that stick around.



1. Were going to fix this by adding these clones to an Empty Parent GameObject and getting rid of them as time goes on.
2. Create Empty GameObject and name it **Spawn at Runtime**. Reset its position
3. In Enemy.cs create a new [SerializeField] called parent of type Transform.



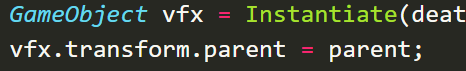
Now the reason its Transform is bc the Empty GameObject JUST has a Transform Module



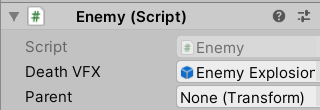
1. Lets save the Instantiate method to a new variable called vfx



1. Now this new GameObject we’re going to set its parent to the Spawn at Runtime Gameobject.



1. Go back to Unity highlight all the Enemies and look for the **Parent** field in the Inspector



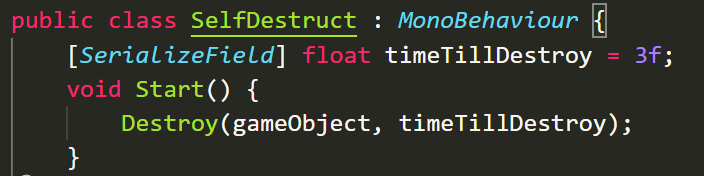
1. Grab your **Spawn at Runtime** GameObject and drag it to this field.
2. Now if you test the game again and pause it all the clones are children of this GameObject

**Challenge:**

* Create a SelfDestruct.cs script and attach it to something…
* Use Destroy() to make sure none of the particle effects stick around. Look up what parameters Destroy accepts

**Solution:**

1. Create a SelfDestruct.cs script and open it. Delete the Update method
2. Inside of Start call the Destroy method, give it the GameObject and created a [SerializeField] timer variable.



1. Open the Enemy Explosion Prefab editor
2. Click Add Component
3. Attach SelfDestruct.cs script
4. Now if you test the Game the Particles should disappear from the Hierarchy after 3 seconds.