**Enable a GameObject from C#**

**Objective:** Students will arrange an explosion and a level reload when the spaceship crashes into anything.

1. Open **CollisionHandler.cs** script
2. Lets create a [SerializeField] variable to control Load Delay and set it to 1 second.

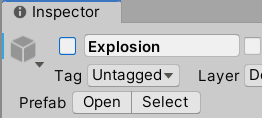


1. Lets import **UnityEngine.SceneManagement** and put a comment next to it **//ok as long as this is the only script that loads scenes.**



In our **StartDeathSequence** method we want to trigger the explosion.

1. Go to Unity and find the **Explosion** particle prefab. Remember this prefab has a particle system and an Audio Source.
2. Lets attach this **Explosion** to our **Spaceship** and **reset its position** so its right at the root of the ship. Lets also **increase the scale** of the Explosion to **10x10x10**
3. **Turn on Play On Awake** for the Explosion Particle System and the Audio Clip
4. Apply the overrides to your SpaceShip.
5. If you click Play to test your game you should immediately see the explosion particles and hear the explosion.
6. Now since we don’t want it to do that we can disable the entire Game Object by Unchecking the box in the Inspector.



1. Then what can happen when the game is running, is we can pragmatically turn Game object on in our code.

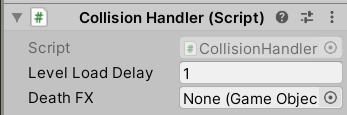
**Challenge**: Enable a Game Object during play

* Ensure particles and audio “Play On Awake”
* Have a reference to the GameObject ex. deathFX
* Use deathFX.SetActive(true);

**Solution:** Under the first [SerializeField] create a second type GameObject and name it deathFX



1. Go back to **Unity** and make sure both SerializeFields popped up for your SpaceShip



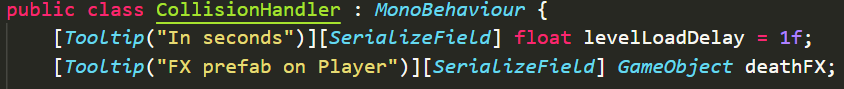
1. Lock the **Inspector** by clicking on the **Lock** icon in the top right

|  |  |
| --- | --- |
| Unlocked |  |
| Locked |  |

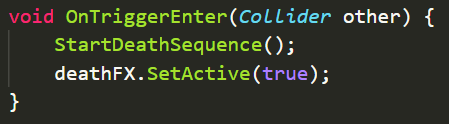
1. Drag the **Explosion** from the Hierarchy to your **DeathFX** field



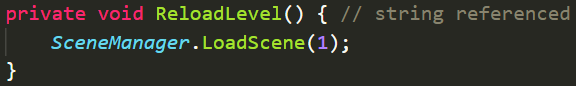
1. Lets add ToolTips to these new Serialize Fields



1. Inside of your OnTriggerEnter method lets activate the deathFX

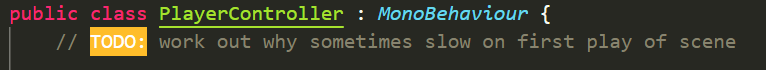


1. Go to Unity and test your changes. Make sure your particles play when you crash.
2. Create a method called **ReloadLevel** and load scene and include a comment so you remember its referenced elsewhere.



1. Go to Unity and test changes. Crash and game should reload the level

**Optional**: At this point my super speed is annoying. So lets go to **PlayerController.cs** and place a note



**Possible Bug:** if you have a build index can’t be loaded error go back to Build Settings remove the scenes and re add them

