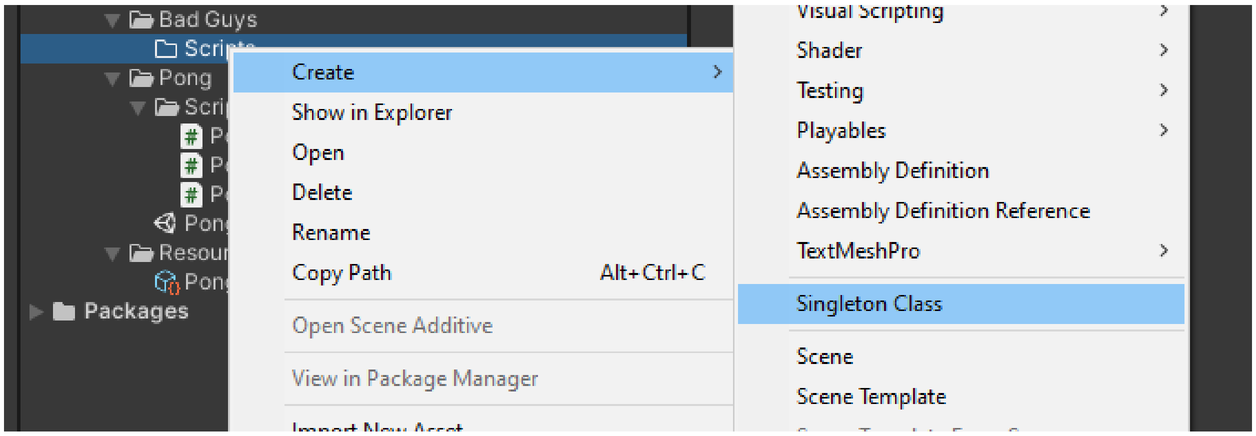
Singleton System Tutorial

Links and Resources

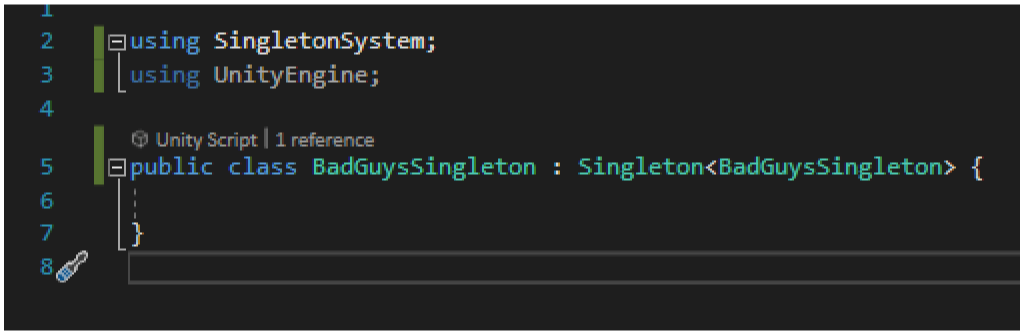
- [Singleton System on the Itch.io]

Create a Singleton in 3 Steps!

1. Create a new class that extends from ‘Singleton’
   - The easiest, best way to do this is by right-clicking in the Project view where you want to the new script to go, and selecting ‘Create -> Singleton Class’



   - The second way to create a Singleton class is to create a new C# script and edit it to have the following structure:

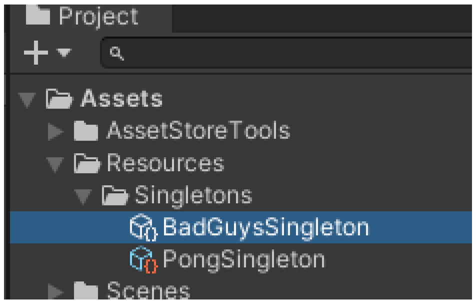


```
using SingletonSystem;
using UnityEngine;

Unity Script | 1 reference
public class BadGuysSingleton : Singleton<BadGuysSingleton> {

}
```

i. The class itself can be called anything, but we recommend putting the term 'Singleton' in it as best practice.
ii. The Type within Singleton<Type> is the class itself.
iii. The class must include the 'SingletonSystem' namespace by declaring 'using SingletonSystem;' at the top.

2. Allow Unity to reload

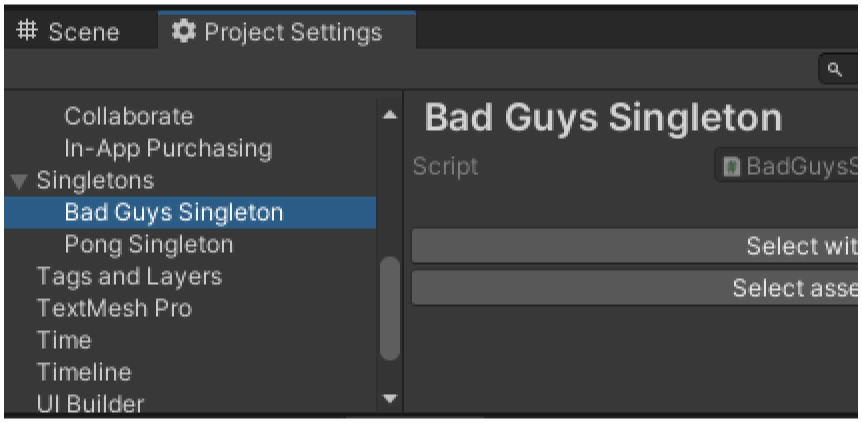
- After Unity reloads and/or recompiles, it should have automatically created a ScriptableObject for the new Singleton class within 'Assets/Resources/Singletons'



- Feel free to move this asset anywhere within your project's asset folder **as long as it is still within 'Resources'**.

3. Add fields and methods to your Singleton and start using it

- Access and edit the fields on the Singleton by either selecting the ScriptableObject in the Project view OR by using the Project Settings:



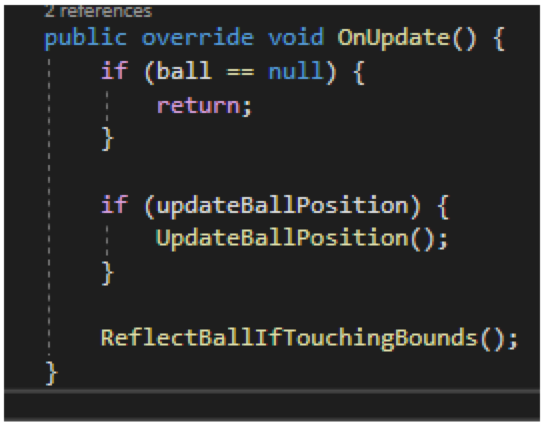
- The 'Instance' of a Singleton can be acquired in one of two ways:
  i. Using MyCustomSingleton.Instance (static property)
  ii. Using Singletons.Get<MyCustomSingleton>() (static method)

Pong Example

For an example of how Singletons can be used in practice, check out the 'Pong' example scene within 'Assets/Singleton System/Examples (Safe to delete)/Pong'. Please note that there are no controls or input for the Pong Example, and the ball will simply bounce around the screen indefinitely.

Aspects highlighted by the Pong Example

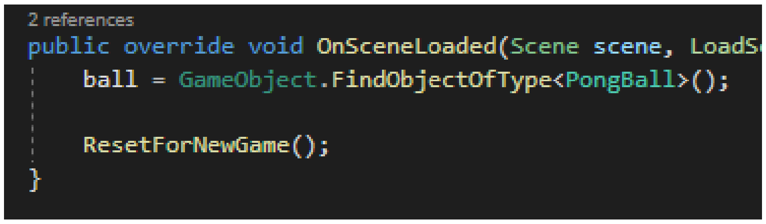
- Game logic handled within the Singleton



```
2 references
public override void OnUpdate() {
    if (ball == null) {
        return;
    }

    if (updateBallPosition) {
        UpdateBallPosition();
    }

    ReflectBallIfTouchingBounds();
}
```

A snippet from PongSingleton.cs highlighting the OnUpdate() functionality of Singletons.

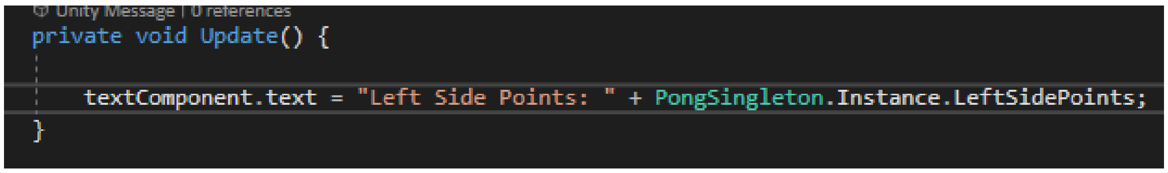
- The Singleton accessing GameObjects within the scene



```
2 references
public override void OnSceneLoaded(Scene scene, LoadS
    ball = GameObject.FindObjectOfType<PongBall>();

    ResetForNewGame();
}
```

A snippet from PongSingleton.cs highlighting the use of OnSceneLoaded() to get GameObjects.

- GameObjects within the scene accessing the Singleton



```
Unity Message | 0 references
private void Update() {

    textComponent.text = "Left Side Points: " + PongSingleton.Instance.LeftSidePoints;
}
```

A snippet from PongPointDisplay.cs showing how to access the Singleton and its fields.