# Design Pattern’s

[Introduction](#_Introduction)

[Why add design patterns](#_Why_add_design)

[Observer Pattern](#_Observer_Pattern)

[Singleton Pattern](#_Singleton_Pattern)

[Summary](#_Summary)

# Introduction

I was making a 3D project in Unity a game called “BrainFizz” it’s an adventure type of game, the game contains a quest system where you can interact with NPC and talk to them there is a multiple dialogue options to choose from and there is a combat system where you fight monsters of that word.

# Why add design patterns

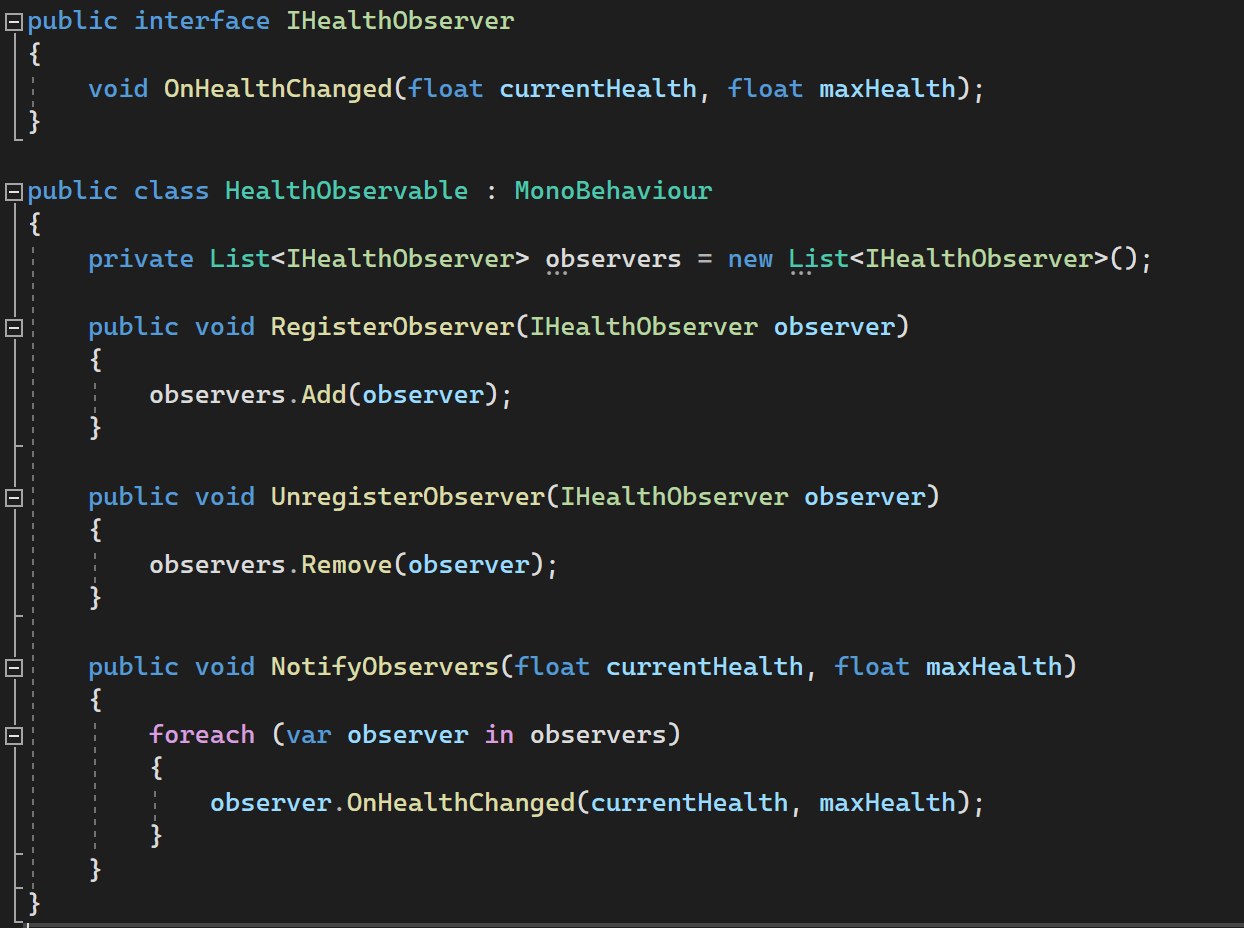
The Design patterns are very useless for this project for being a open world 3D.  
Its more easier to maintain code provide standardized solutions to common problems so the code is going to be more understandable and easier to maintain, when developers encounter a familiar pattern, they can quickly understand its intent and implementation.  
Design patterns makes implementing systems easier and more modifiable, its easier to extend that component as well like for combat adding new weapons.

# Observer Pattern

I added a Observer Pattern for the “HealthController” script. The Observer Pattern defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

There are 2 scripts the “Health Controller” and the “HealthObservable”.

Observable class to manage observers and notify changes

Code:  


“IHealthObserver” Interface

This interface defines a contract that observers must adhere to. It declares a single method OnHealthChanged which will be called whenever the observed object's health changes.

“HealthObservable” This class serves as the "observable" or the subject. It keeps track of all the observers (objects interested in health changes) and notifies them when the health changes.

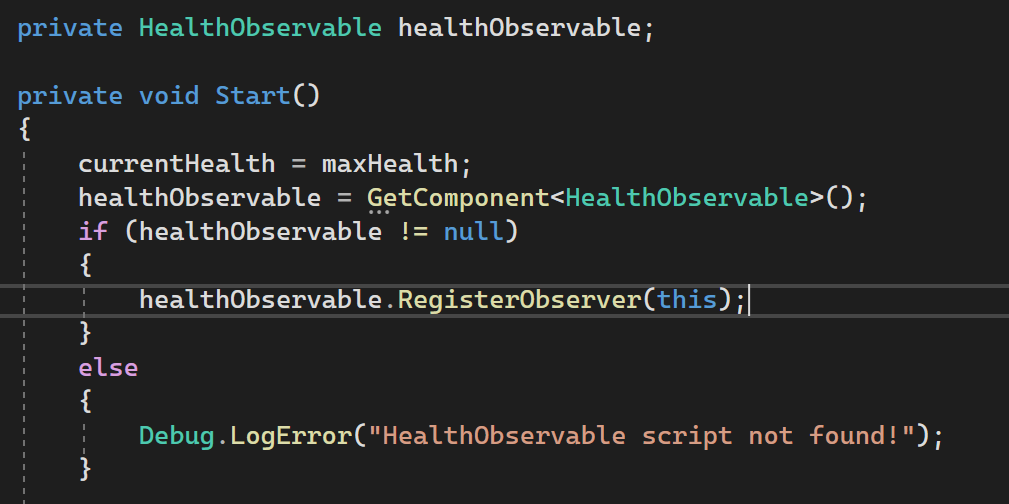
Methods:

* RegisterObserver: Adds an observer to the list of observers.
* UnregisterObserver: Removes an observer from the list.
* NotifyObservers: Iterates through all registered observers and calls their OnHealthChanged method, passing the current health and max health as parameters.

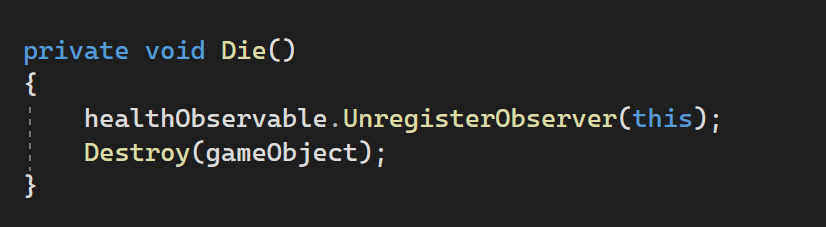
Now for the “Health Controller”

Properties:

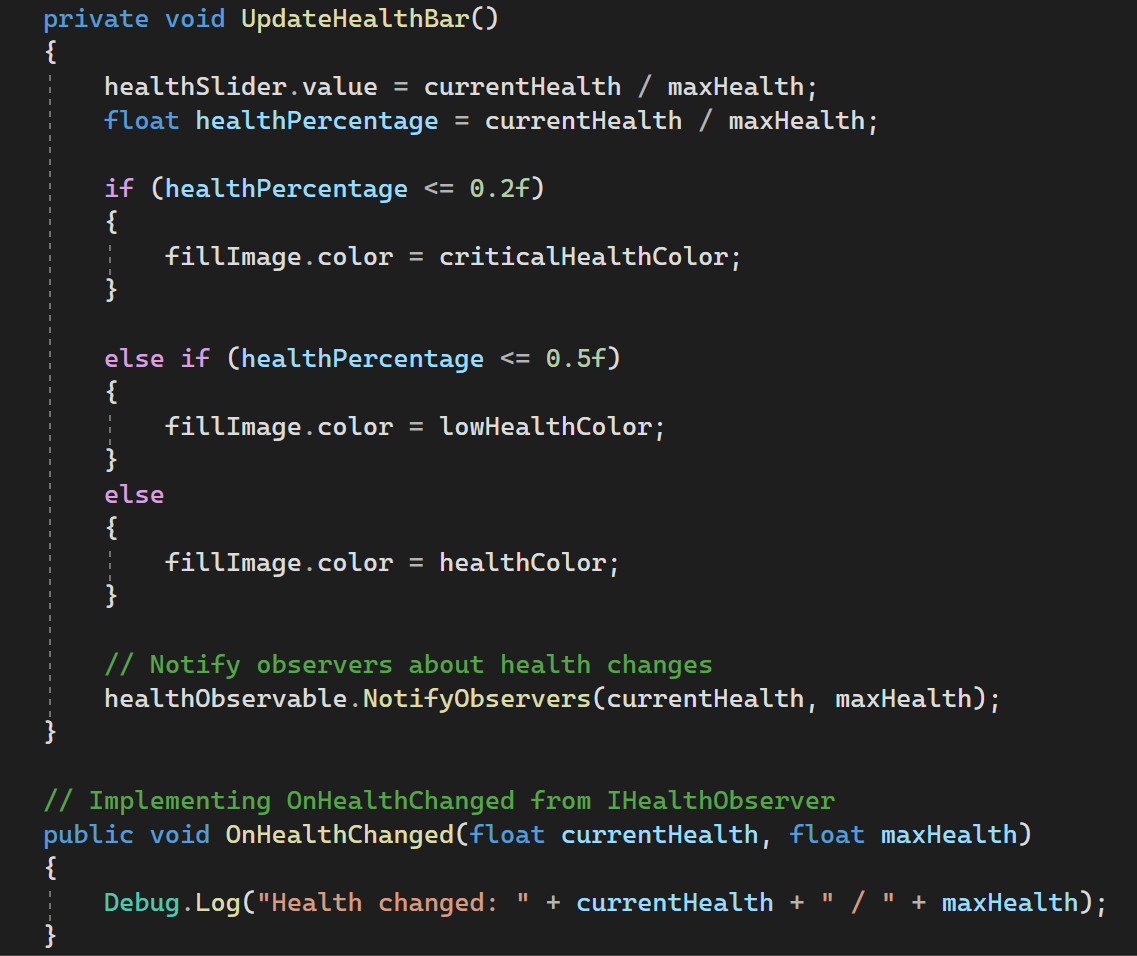
* “maxHealth”: Maximum health value.
* “currentHealth”: Current health value.
* “healthSlider” and “fillImage”: References to UI elements for health representation.
* “healthColor”, “lowHealthColor” and “criticalHealthColor”: Colors for health indication.



The start basically finds that script that is attached to the object and adds to the observer.



If the object dies it destroys the object and removes from the health observer script.

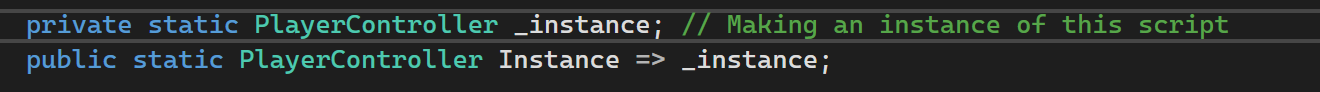


“UpdateHealthBar” well updates the health bar and changes its color depending how much it has, and it calls the “NotifyObservers” notifies the observers about the health changes.

“OnHealthChanged” takes the current health and the max and displays what it has for that specific object.

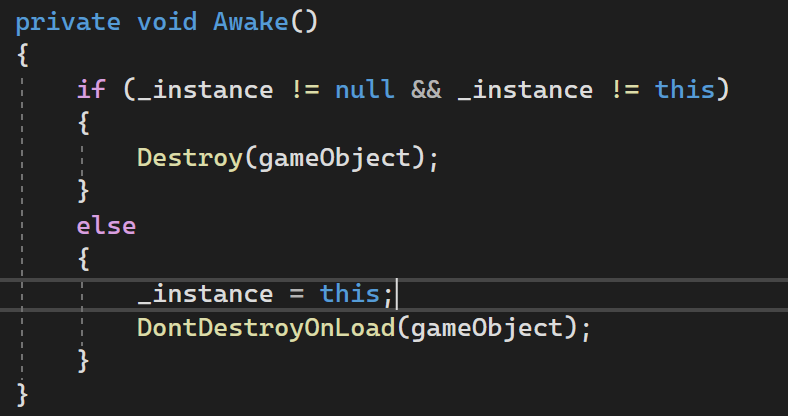
# Singleton Pattern

Implementing the Singleton pattern can help ensure that only one instance of the “PlayerController” exists throughout the game.



Making the singleton instance – “private static PlayerController \_instance”

And this “public static PlayerController Instance => \_instance” is a public property to access the singleton instance.



The code inside Awake() ensures that only one instance (singleton) of the MonoBehaviour exists in the game, Awake function is the very first thing that gets called before any other function.

This function makes sure that there is only one instance exists throughout the game.

# Summary

In summary, the “HealthObservable” class manages a list of objects that implement the “IHealthObserver” interface. It allows objects to register themselves as observers and then notifies them when there's a change in health by calling their “OnHealthChanged” method.

This design enables a decoupled way for multiple components or classes to react to changes in health without directly knowing about each other.

Singleton is used because you only need 1 of that object/script, like the player you only need 1, there isn’t multiple players and there shouldn’t be in that particular project.