# Design Patterns

**Singleton**

An example of how I used a singleton was by using the Level Blueprints that are built in to unreal engine. I used this to setup the environment in the scene, load the HUD and setup the player controller object, for other scenes It was used to setup the UI and the menus

Screenshot

Advantages

One of the advantages of the level blueprint is the fact that all the functions in it will always be called before the rest of the game. This meant I could use it to ensure that the required objects had the correct values and were always setup properly.

Disadvantages

One of the disadvantages of the level blueprint I think is that there isn’t really a way to view the active values of certain variables without opening the blueprint up fully. And I had an issue early on where my character wouldn’t sync with the player controller because I had to search for the player index and it wouldn’t let me find the existing player in the scene.

**State Pattern**

I used state patterns for my player and enemy controllers. For my enemy the state was configured using an enum so it would change depending on whether certain requirements were met. The Player only had a couple so it could be controlled by a set of booleans

**Advantages**

One of the advantages of using the states was when I started implementing the animation. In the animation event graph I could setup a set of checks so the animation controller knew what state the enemy or player was in and play the set animation accordingly. An example of this is where I used the arming and disarming method to play the same animation in reverse and attach the staff to different parts of the player skeleton mesh depending on which Boolean was active.

Screenshot of graph

Disadvantages

A disadvantage to the state pattern was probably in trying to implement an animation where a character could do different motions at once. The limit to using the enum was that it was only allowed to be one at a time and it meant my enemy wouldn’t have been able to play the attack animation while it was also running towards the player.

# Programming Principles

Dry Code

I kept my code dry by separating into functions and eevents and calling them instead of using the same code / BP repeatedly

Here are some examples

explain

Encapsulation

I kept data that didn’t need to be easily changed from the inspector hidden and private and only allowed access to the parts that needed it.

Here are some examples

Explain