# Discussion of Architectural and Design Patterns

This section lists the various design patterns being used in the project, where they are being used and why they are being used.

## Abstract Factory

The Abstract Factory is being used for the construction of Rooms, in this way it is easy to extend the game with the introduction of new types of rooms while maintaining encapsulation by distancing the client from the actual instantiation of the Rooms.

## Decorator

Is being used to create items that would be found within rooms, it is not being used to add the items to rooms rather it is being used to created the items themselves. By using the decorator design pattern we can allow the creation of a plethora of items from a smaller pool of methods, e.g. we can use “makeBroken(makeGolden(makeChair())))” , “makeBroken(makeChair())” , “makeGolden(makeChair())”, “makeChair()” to create 4 different objects using only 3 methods, rather than having to create 4 separate classes.

## Observer

The Observer is being used to notify NPC and the GUI as to when and where to the player moves, with the Player class being the subject and the NPC’s and GUI being observers. In this way both the GUI and NPC’s will be aware of the Player’s current position within the game, allowing 3rd party clients to create AI for npc’s that takes the player’s position into account.

## Strategy

The Strategy pattern is being used to allow for the AI of NPC’s, it will allow 3rd party developers to easily develop alternative AI programs for NPC and swap them out with easy, thus supporting extensibility.

## Façade

The Façade is being used to simplify the creation of the game world, as one class will be provided at the front end that will provide one method that calls all the commands for the creation of rooms(through the abstract factory), the creation of items(through use of the decorator) and the adding of items to rooms. This will make it far easier for 3rd party developers who simply want to create a replace GUI for the game rather than modify the game itself, as it will hide all the game’s construction behind 1 method in the Façade Class. This is the Pattern we selected through self-guided research.

## Interceptor

The Interceptor is being deployed in-between the game and the GUI, this is in order to provide 3rd party developers with hooks between the User-experience and the game program itself. These hooks will allow them to create interceptors to extend or listen for events and use these in their own implementation of the GUI.