Andrew Ngo

December 10th 2018

CS 4773

Assignment 4

1. The observer pattern is identified by a subject that notifies its dependent object observers of a state changes in the subject through a function call. The observer objects call an update function once the subject’s state has changed. I chose to implement the observer pattern because I felt that using it in conjunction with the state pattern would be simple. In my project, the observer notices a state change in the Game object and proceeds to print which state the game object is in through its update function. The pattern was fairly straightforward to implement in my project because of the distinct states that the observable Game object has. An interesting aspect about the observer pattern is that any method in the observable object can change the state with the setChanged() function call. An issue I came across when implementing this design pattern is deciding what should be observed and when the observable object notifies its observers.

2. The command pattern encapsulates an action in an object that can be used multiple times or performed at a later time. These command objects can also be switched with other commands. I chose the command pattern because I was already considering implementing this pattern with my previous project ideas. Implementing the pattern was similar to the state pattern in the way that the individual commands are separate classes like the different states in the state pattern. This command pattern was interesting as it allowed me to change states of the Game object within a command. Additionally, the concept of a remote control for the command is interesting. I did not run into any issues while implementing the command design pattern.

3. The state pattern is identifiable by different behaviors that an object can show based on its different states. I chose to implement the state pattern as I determined that it would be the most intuitive design pattern. This pattern allows for the game object to have different behaviors depending on its state. State pattern is the most intuitive pattern because of my familiarity of finite state machines. With that in mind this pattern was not as interesting as the other ones. An issue that had occurred during the state pattern design implementation was setting the state of the game through the setState() method. My initial attempt had different state objects initialized outside of the setState() method which caused a number of headaches. This was fixed by creating a new object in each setState() method call.