EECS 448 Team 2

Design Patterns Document

We have a wide range of design patterns included in our Project 3 codebase. Patterns included in our design include Wrappers, multiple different instances of state, and an iterator. Each of these design patterns are used in a unique way in our project to make our code function properly and efficiently.

Our code includes multiple instances of declaring a function, which at its core is a wrapper. Specifically, the first function called in our game is called startGame(). All this function does is call other functions and object constructors. This function itself does absolutely no computation and just hands off the work to other classes and parts of the code. This is a prime example of a wrapper.

There are multiple examples of state in our code. One obvious one is where our EventListeners are listening for a keyup. When this happens the state instantly changes (with the change of a bool called “key”) and the character is raised, and depending on when the key variable is changed, the game with either end or progress. Another instance of a state is whether the game is over or not. While the player is running the game is in a state of playing. If the player crosses the finish line or runs into an obstacle the state will change to the game is over.

The final example of a design pattern is an iterator. In our project, the player must jump over multiple obstacles to reach the finish line. The obstacles are stored into a list object, and is iterated over throughout the course, placing the obstacles.

It is has been made clear that our project 3 includes multiple different instances of design patterns including one iterator, multiple changes in state, and multiple different wrappers.