Randall Krueger
Chip’s Challenge
Report
10/13/18

Between the midway deliverable and the final product, several changes were made, especially with respect to the patterns used. The decorator pattern was removed, with a key simply containing its color as an integer. The strategy pattern was also removed, as buttons and the corresponding strategies were not implemented. Instead, the singleton pattern was adopted by the Chip class. As there is only ever one Chip, and the same Chip can be used on multiple levels, this pattern was a perfect fit. Chip’s constructor no longer has functionality, but instead, the class itself is accessed, and a method inside returns a reference to the single Chip object. Additionally, the state pattern was adopted by the level classes. The level that the game is currently in is the active state, and the one referred to by the game engine. When the state changes, the level returns a new level object referring to the next level to be played, and the level state changes to the new level object. Ice, an exit, and exit keys were implemented, adding additional obstacles for Chip to overcome.

Patterns:

Pattern Name: Observer	
Class Name	Role
Entity	Observer
Chip	Observable
Purpose: The entities on the board need to observe Chip to see when Chip comes in contact with the entity.	

Pattern Name: Singleton	
Class Name	Role
Chip	Singleton
Purpose: There only ever needs to exist one Chip. This way, the program is prevented from creating multiple Chips that are all being moved by the player and being observed by the entities.	

Pattern Name: State	
Class Name	Role
GameEngine	Acts differently based on state
Level	General State
LevelOne/LevelTwo	Specific State
Purpose: Allows the game engine to have a current level state, with corresponding map and start position for Chip. Once a level is beaten, the current state returns the next level to play, and the engine can transition to the next state, and begin play.	

I liked the final design of Chip’s Challenge. The patterns are outlined above and worked well with the pieces of the program. I based the program off of the Columbus game. I modified the movement to have the edges scroll, and added multiple images for the player. Then I added entities that perform various actions when touched by Chip. Finally, I added multiple levels and a way to transition between levels. I also had to make changes to the main classes, like the map and the engine in order to use a standard map instead of something generated, the printing of various objects that move, and even a UI that shows how many of each key Chip has. In the end, I was very happy with the result, but if I were to start over, I would limit the references to other classes and streamline processes. For example, the exit determines when to transition to the next level, and it calls the map’s nextLevel method, which in turn calls the engine’s nextLevel method. I would like to standardize which classes reference which classes and limit this “baton passing”. I also would like to just add more obstacles and more levels.