Patrick Falvey
CSE 40793
Chip's Challenge Homework #6

My final design came together quite well given the scale of the design. Writing from scratch was a little daunting at first so to help I used the basis of my design from the Christopher Columbus game to help get me started. I decided to keep the random "island" generation from that homework to add a new element to this game that was different from the original. These islands are now the blocks that serve as obstacles for chip to go around.

Implementing my three design patterns helped keep my design clear as I was writing more complex code. My three patterns were MVC, Observer, and Singleton. MVC was the overarching design. The view consisted of the map/grid, chip, and the level drawing which generated blocks, chips, chip gate, blue key gate, blue key, and obstacles. Chip's view (CCView) implemented the observer pattern as the observer. It would update every time the model changed chip's position. The model (CCModel) implemented the observer pattern as the observable in this case. It also kept track of chip's game progress as he collected chips and the blue key. Finally, the controller (CCController) handled the key events and the user interaction. The controller also implemented the Singleton pattern because it was a static class and only one of them could be instantiated per game. I had the controller be a singleton so that the user interaction would not get messed up if there were multiple objects trying to control user input. Overall, I was pleased with my design and with how my three design patterns all smoothly interacted with each other.

If I was starting from scratch again there would be a few things that I would change. First would be to write a few more functions to clean up the level making. I have a lot of lines regarding making new image views for specific walls and obstacles. However, a lot of these have hard coded values in them because they are for specific obstacles between the two levels. Next, I would improve the combination of the chip generation and the wall generation so that there would be no potential at all of gridlock on the map. At its current point, the chips and the wall block generation is random to mix up the game. However, theoretically the wall blocks could entrap one of the chips, the blue key, or the entrance either obstacle. I eliminated some of this by adding buffers on the wall block generation to make sure that the immediate entrances the the obstacles were not blocked, but adding more checking and less entropy would help ensure no blockage would ever occur.

Design Templates:

Pattern Name: Observer	
Class Name	**Role in Pattern**
CCView	Observer
CCModel	Observable
Purpose: Chip's image (CCView) needed to update on the map when the user moved him. CCModel kept track of chip's movements and what needed to happened based on where he was in each cell. CCModel would then notify CCView of this and CCView would change the view accordingly.	

Pattern Name: MVC	
Class Name	**Role in Pattern**
CCModel	Model
CCView	View
MapDisplay	View
CCController	Controller
Purpose: Given that I was having a graphical display, I wanted to split it up using MVC. CCController accepted user input and decided what to do with it. CCModel kept track of the internals of the game and handled what to do when chip encountered certain tiles or obstacles. CCView displayed chip. MapDisplay also displayed the map which consisted of the tiles, obstacles, and chip himself.	

Pattern Name: Singleton	
Class Name	**Role in Pattern**
CCController	Singleton
Purpose: Given that this was the only class that decided on what to do with user input and key events, I made it a singleton so that there would be only one class and one instance of that class that dealt with direct user input and key events. If there were any more, I felt that there could have been some conflicts or potential duplicate events.	