Peiyun Zeng
CSE436
LAB5—BattleShip

Design/Method

I implement this game following the MVC design model: model, view and controller. The model here is composed of Ship, Player and Game. Game is a whole logic control of how the player and AI play the game, when it's win, and judge if the click on the view is a hit or miss. Ship class stores everything about a ship including its size, type and the state of hit. Player class does the same thing as ship class, it stores a player type and what ships he owns.
As for views, I choose to implement a customized Game View to display our ship state and use "long press" gesture to rotate the ship and "pan" gesture to move the ship to locations we want, which is controlled by the game view controller. To make every view looks better and could respond to our operations, I use customized view for ships.
Two viewControllers are implemented for the app, game viewController for AI and game viewController for human vs human. In both GameView controllers, they control each part's logic for display (tons of code for transitions) and some very trivial logic to do the computing parts. I choose to use protocol which is defined in Game model, in this way, the game control is much more scalable. Also, I use the "delegate" to invoke some methods, which are not defined in the class. A main viewController is implemented to control the main page of the game. When button "one player" is pressed, the view controller for AI is pushed to nav. When button "two players" is pressed, the gameViewController for human vs human is pushed to nav.

Game Instructions
In this Battleship game, when the user starts the game there are two versions for the user to choose from. The user can either play with the AI or choose the Two People version of the game to play with another human. After the user enters the game mode of his/her choice, the first user would start by filling the five ships of different sizes onto the game board. In the version that the user is playing with the AI, there would be a view letting the user wait for the AI without showing the choices of AI. After the two players both finish placing the ships onto the board, the program would randomly generate a user to start battling. If the user failed to hit on any of the ships of the other user, then the other user automatically takes the game over. The sound effects in this game would be demonstrated in four different formats: one sound for failing hitting on any ship, one sound for hitting on one ship, and another sound comes into effect either when one player wins the game or when one ship sinks. Players may only win the game when they have successfully beaten all of the other user's ships on the board first. The sound effect is implemented using *AudioServices*

References

Images for battleships are provided by Jingxuan Zhao and Shangqing Li.

Consulted with Todd Sproull