Peter Griggs
Mini Project 3: Multimodal Battleship

1) see source code in this directory

2)

a)

The first challenge I had during the ship deployment phase was being able to easily reach all of the tiles on the board, including the bottom row. I just added an offset to the y coordinate of the cursor position to solve this, but when I had a friend play my implementation standing up, I realized that I had to add a larger offset than I had originally programmed when I was testing the system sitting down. This is a good example of user testing in different scenarios!

Another challenge I faced was that the ship rotated completely differently from what I expected when I used the unmodified hand.roll() angle as the rotation. I realized that the reference frame must have been reversed, so I took the negative of this angle. Once I fixed this, I quickly saw that the ship tended to rotate much further in the clockwise direction (from my reference frame), either because it is easier for me to physically move my hand in that direction or due to some setting in the system. To account for this, I added an offset.

b)

The only real challenge I faced while implementing the player's turn was recognizing the word 'fire'. The voice recognition API made several mistakes, recognizing words like ‘tire’ and ‘heart’ instead of ‘fire’. I didn’t completely solve this issue because I wasn’t sure if it was potentially due to issues with my mic or the voice recognition service, and because I normally only had to repeat myself once to get the system to recognize the ‘fire’ command.

c)

During the cpu turn, I faced a lot more issues with the speech recognition service recognizing the word ‘miss’. It would interpret my speech as many different words like ‘this’ or even the word ‘Miss’, which is the same spelling but different capitalization. I the fixed the capitalization problem by modifying the userSaid function to avoid any mismatches in capitalization between the commands and the speech transcription. Additional improvements could be made by using something like a distance metric between the desired commands and the recognized words and accepting words that are close enough to the desired command. For this, we could use something like edit distance. I’m not sure if this optimization is worth it for single-word commands, but maybe it is worth it for longer commands.

3)

I made an extension to the setup process that allows a user to point to a tile and deploy a ship there with a voice command. During the deployment phase, if the user points to a tile and says a phrase with either the word "place" or "here" and a valid type of ship, the system will attempt to place that ship in the specified location. For example, "place a battleship", "place a patrol boat here", and "battleship here" will all parse correctly. Also, if the user says "place a battleship vertically", the system will try to fit the battleship vertically instead of the default horizontal orientation.

I implemented this by recognizing if a user says "place" or "here", then checking for the words "battleship" or "patrol boat". If both of these checks pass, the system will place the ship by using the location of the cursor. This way, the user can say "place a battleship" or "battleship here", and both phrases will work. This allows for some flexibility without losing the meaning of what the user is trying to do. Additionally, if the user says the word 'vertical' or 'vertically', the system will set the ship rotation to pi/2. By default, it sets the ship's rotation to 0. The ship placement logic is implemented in the moveShip function, which enumerates the ship objects on the board and finds the one you want to move, then moves it to the desired location with the desired orientation. I had some issues where the ships would translate several columns to the right or left of the cursor, so I had to add a offset to correct for this phenomenon.

This extension helps the user because they don't have to deal with grabbing and rotating the ship if they don't want to. I found that using the point and deploy feature worked better for me. The rotation using hand.roll()

can be finicky, which is the main reason why I implemented this feature. Additionally, users might find it hard to use the regular deployment for various reasons. For example, if a user has a wrist injury, placing ships via point and deploy would be significantly less painful.