Refined Design – 11/19/2017

Since our previous design submission, we have been able to implement a solid prototype of our board using Pyro, allowing distributed access to the same board object that can be updated and read exactly in the way we want. Currently, we host the nameserver locally on one of our systems, but we hope to eventually move it to a remote server that we'll have running. We hope to be able to re-use that server for our "extra" additions (high-score board, allowing gamer-tags for players, etc.). We managed to get past one of our initial hurdles of automating the nameserver start-up & initialization of our board object upon a host starting a game.

Besides setting up our board object with Pyro, we have also been able to simulate movements and updates to the board by testing sharks. One major implementation obstacle we faced was the process of updating and rendering the board to emulate real-time movement of the sharks on the board. We discovered that the issue originated from our algorithm used to update the board given a new current position of an object. Every time a shark moves, it first read the board before writing its position, and then re-read the board after it wrote to it, which slowed down overall performance considerably. To address this issue, we changed it so that to update the board, the object sends its current location and the board itself will write the shark onto itself.

For the rest of our minimal deliverable, we still need to create a clear distinction between the two user threads, one for controlling the fish and writing to the board, and one thread for simply reading the board. We have begun working to make sharks entirely independent of the user, treating them almost as an extension of the board (except with no influence from the user). Once we clean up the user control of the fish, it is simply a matter of connecting all the pieces into a polished product (i.e. hosting nameserver not locally, creating a game lobby to act as a barrier for players waiting to play, handling how the game terminates).