Just Snek Things

Anne Oursler, Lexi Galantino, Matt Turner

Our project can best be described as distributed Snake. We use erlang gen_servers to allow users to create and join multiplayer games of Snake. Actual calculation for a given game is done in a python program, which is in regular contact with the game server via erlport. The server then communicates with every client gen_server attached to the given game to send updated boards and receive moves from each player. The board is then represented on users' screens via erlport translation to a python gui.

The concurrency in our project comes from the various players' ability to move and function at the same time across a shared domain (the board), which is handled via erlang messaging.

The biggest issue with this project is finding a way to handle so many updates on the shared board, consistently, and quickly. Presumably, this board as a data structure will end up being a large 2d array. Our idea is to limit interactions per player to 1 per .5 seconds. Even with that limit, latency and computation time are huge considerations. We could program everything in erlang, but as all objects in erlang are immutable, the sheer space constraints of so many list updates on a large 2d array seems unfeasible. It might be that a python game server would be easier to program, but it lacks the innate concurrency support that erlang messaging provides. Our current idea is to have a running python process accessed via erlport which is capable of doing a large number of small matrix updates in very short time (something like numpy), with all network traffic handled by the more foolproof erlang gen_servers.

So far, we've spent a considerable amount of time porting over our erlang gen_server code, as well as reviewing possible solutions for the client-side GUI. The next major task is to get a working python implementation of our Snake game running so we can start to test latency and efficacy of erlport. This should be done in the next week. Once that's done, we can start looking at linking up via the gui, fine tuning player number per game, powerups, and other more complex features.

For stretch goals, we're considering things like adding a matchmaking room...a sort of game lobby to coordinate players and give a list of current games, better latency hiding, high score counts, and larger, more complex boards.