Project 3 Backlog

GUI

* Start screen to pick vs another human or vs AI (assuming player is a human, since they’re using a GUI.
* Represent board for game
* Player 2, if a local human, should be able to input moves via the GPU
* GUI can handle any number of pits 4<x<9
* Human player should have buttons to input the pie rule; the second player chooses whether to continue with the starting setup, or to switch sides with the first player.
* A graphical representation of the time remaining for the player’s move
* Representation of who won the game, and possibly a “You win/lose! Congratulations!” sort of thing
* Represent communications with server to local human
* Represent moves and game actions made by remote player to the game state

Server/Client

* Able to connect to remote location (as server or client)
* Write input from remote location to local buffer
* Write local input to local buffer
* Read from local buffer, and perform actions accordingly
* Transfer commands from remote and local to a local buffer
* Read items from local buffer, and perform appropriate actions
* Info command: show the game’s current configuration
* Move command: choose the desired pit for the next move.
* Ack: acknowledge client/server’s message (or something?)
* As server, configure the with user (console) input (probably don’t link with the GUI, too ambitious)
* As server, choose for second player to be client, AI player, or local human input
* Receive moves from client player (if any)
* As server, send ILLEGAL or TIMEOUT acknowledgements if either is applicable, then end game
* Send information (current game state at each turn, server player’s moves, etc.)
* As client, connect to remote host with hostname and port
* Get remote game config
* Receive moves from server
* Send moves to server
* Be able to respond to server with moves; both human player and AI should be possible as client (**Figure out if we want to make two clients, or one that is multi-purpose)**
* Be able to handle and respond to WIN, LOSE, DRAW, TIMEOUT, ILLEGAL acknowledgements from server
* Send INFO statements
* Receive and output INFO statements

Game Manager

* Handle 4<=x<=9 pits
* Handle different number of seeds in each pit, with standard variation (each pit has same # of seeds)
* Handle different number of seeds in each pit, with random variation (each pit has a random # of seeds, but their distribution is equal for either player)
* Implement capture rule
* Initialize with standard or random distribution of pieces in pits
* Allow switching of player position’s, depending on player 2’s choice re: pie rule
* Keep track of time elapsed in current move
* Interface with the client-server system
* Check when game is over (no pieces remain on one side of the board)
* Know which player won

AI

* Computer can randomly choose valid move
* Create a future game node
* Create min/max tree with future game nodes
* Computer can look 1 move into future
* Computer can look any amount of moves into future
* Alpha-beta pruning at 1 level deep
* Alpha-beta pruning at any level deep
* Heuristic improvement