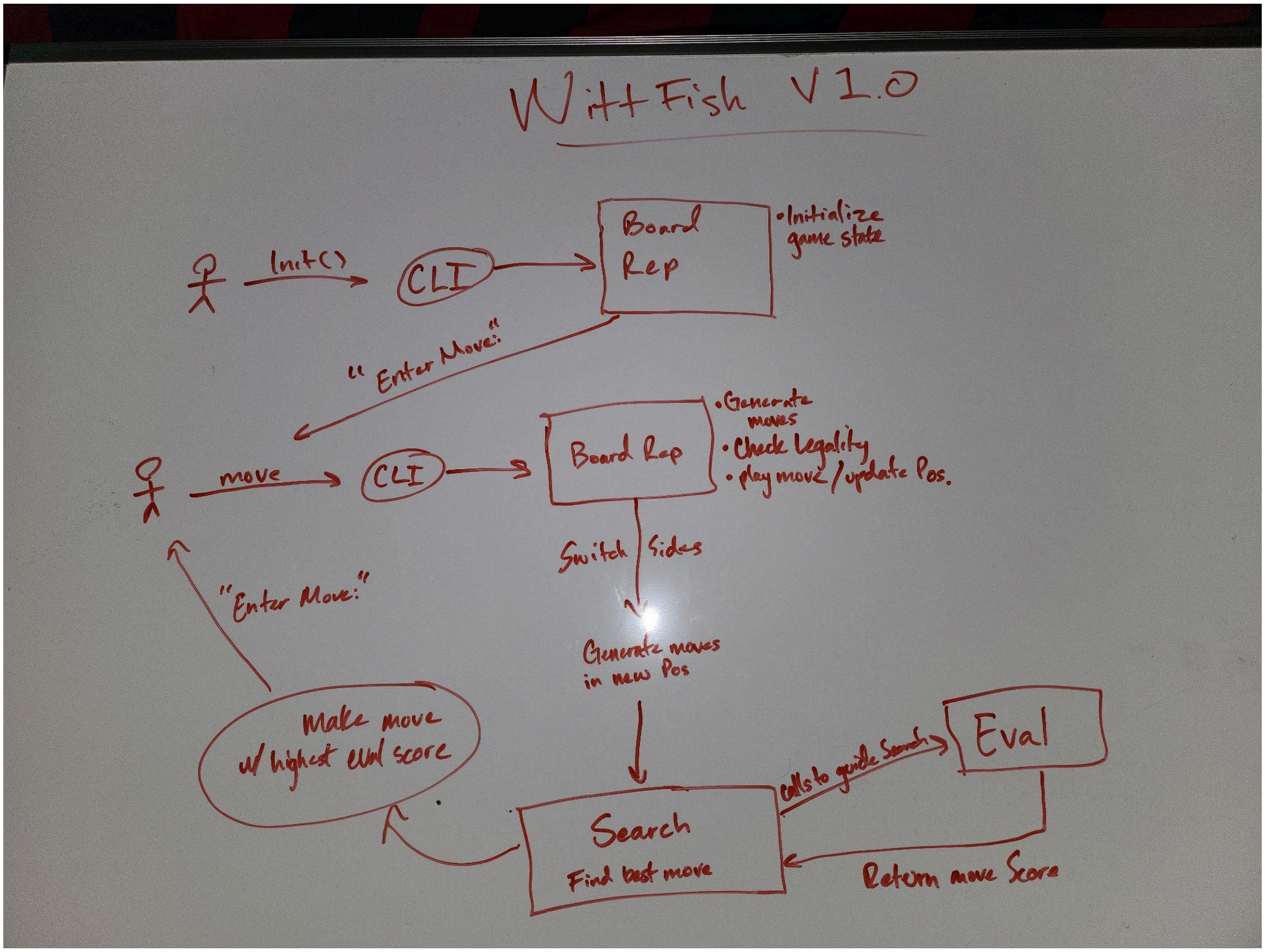
WittFish Design Document

Outline

- Overall system design
- Description of each component
- Road map
- Next Steps

Design



Description of above document:

- A user will start the program via command line, specifying how the engine should initialize.
    - Ex: white/black assignment, search depth, eval coefficients.
- The board representation initializes the game and either plays an opening move or prompts the user.
- If the player makes a move, the board representation checks its legality. If legal, the move is made, otherwise the user is prompted again.
- Once the user makes a move, the engine will generate all legal moves in the current position and search for the best one.
    - The search algorithm uses the Eval function to guide its path.
- The eval function will take a board position and return a score for the current side based on variables like: material advantage, positional advantage, time advantage, etc.
- The flow will repeat once the engine makes a move and prompts the user again.

Board Representation:

- The board rep class contains all information needed about the current state of the game board.
    - The approach will be primarily piece centric (bitboards).
        - uint64_t
        - The benefit of bitboards is that it allows for the use of clever macros that vastly speed up computation.
    - The following bitboards(and possibly more) will be needed:
        - One for each piece and each color (bishop, knight, etc.) – this would be 12 total
        - One for the overall board
        - Others for special moves
    -
    - May need square centric representations in the future, but bitboards are fine for now.
- Will also need (at least) the following functions:
    - Update function
    - Move/demove function for each piece

Search:

- The search algorithm will be an implementation of minimax, searching moves up to a specified depth of positions.
- Goal is to use an alpha-beta algorithm to search depth-first.
- Will use eval function to score resulting board positions from moves the search tries out.
- Bad moves will be discarded, and the algo will attempt to search as far as it can in a ‘good’ direction.

Evaluation:

- The evaluation function takes a board position and the current color to move, and returns a number representing the advantage of that position for the current color.

Road Map:

1. Board Representation
    a. Nothing can work before the board is complete
    b. For every piece:
        i. Implement its representation
        ii. Implement its move generation
    c. Implement special moves (castling, en passant, promotion)
    d. Implement checks/checkmates

2. Debugging Framework
    a. Not sure how this will look yet, but need to get it done before searching and evaluating can be touched.

3. Search/Eval
    a. Search and evaluation can be developed in relative concurrency, but not until the board rep is complete.

4. Iteratively improve Search and eval functions

5. Implement Universal Chess Interface (UCI)

Next Steps:

- Board is all that matters right now