Michael Guan(mhg99), Enoch Kim(ek537), Jeffrey Shen(jcs528)

Progress report

Vision:

As of now our vision seems to be to create a fully functional chess game. The chess game must be fully functional with working castling, en passant, queening, pawns moving two tiles as their first move, etc. Additionally, the game must be able to recognize when checks are given, when a move is illegal due to a pin, and when checkmate occurs. All the move sets of pieces are given by the standard chess rules. This vision differs from our initial vision substantially by limiting ourselves to one game: chess.

Summary of Progress:

So far, we have been able to display the chess board, put the pieces on the board, and take turns in the system. We have properly bound the movement of the pieces to the board as they cannot move off the board. Additionally, all movement of the pieces are bound by the movesets that are defined by chess rules with a coordinate system defined by 0,0 at the top left; however, this is not a completely flushed system. Pieces may take other pieces and pass through other pieces instead of being blocked, as in an actual game of chess. Special movement has not been implemented as castling, en passant, queening, and allowing the pawns to move two tiles as their first move. Additionally, the pawn is able to move diagonally whenever it pleases instead of being bound by the rule that it may only move to take another piece diagonally. Commands that are not properly worded will lead to errors whereas if the command is properly formatted but are invalid do result in the system to ask for another move. In the demo, we demonstrated proper non-special movement of all the pieces and that they are displayed on the board.

Activity breakdown:

Enoch:

1. Worked on making pieces game.ml and the corresponding specification in game.mli
    a. Defined all the pieces and the game's abstract representation
2. Worked on the state.ml and the corresponding specification state.mli
    a. Defined the state's abstract representation (board and turns). Implemented movement of pieces and updating the board to be displayed properly, initial to_string, and placement of all the pieces of the board

Michael:

1. Worked on the chess pieces for game.ml
    a. Made piece representation and a function to check if a move is valid for certain pieces
2. Created test cases and make files to automate building and testing
    a. Automated the running of make test and make build
    b. Tested pieces to see if they were moving according to their moveset.

Jeff:

1. Worked on making the various commands in command.ml and the corresponding command.mli for the game and checked how they interacted with the main engine
    a. Defined the various commands and the parsing function for the command.ml
2. Worked on main.ml and created the game engine that ran the program

a. Caught the possible commands for the game that ran through the engine

Productivity analysis:

We were productive in the sense that I believe we spent a considerable amount of time problem solving, developing and brainstorming as well as making changes to our original plan. We realized that some of the goals that we made were too high and that we haven't realized the work that goes into making a fully functioning chess game. By making chess the goal of the project, I believe that we are well on our way and on track to building a fully functioning chess game. This goal that we set for ourselves is flexible as we also have room to expand if time allows us in the future.

Scope Grade:

We, as a team, decided that we have achieved good scope for this sprint of the project. For the goals we set out originally in the charter for this project, we have created both the board and pieces that we defined in the satisfactory and good scope for Alpha. However, we were unable to achieve the idea of uploading a specific json with the game information. After working on the project, we discovered the difficulty of using just a json to store all the information on the rules of the game without coding our own lexer to parse the language in order to fully define an abstract game. Thus, our original goals of creating a single game engine that can take jsons to create various instances of different types of chess games with different has been changed. We did not give ourselves an excellent grade for this reason, as we were unable to continue to create a more advanced system that we originally set out to make. Instead, we think we have received a good scope for our ability to complete the board and moveable pieces that we first originally set out to do. Beyond that, we also think that we will rethink some of our goals and try to make a more feasible project.

<u>Goals for next sprint:</u>

1. Satisfactory Scope: Make pawns behave correctly and piece blocking
    a. We want to add additional logic to allow pawns to be able to move forward twice on the first turn and for them to only be able to move diagonal one space when capturing another piece. Next, pieces that can move a range of spaces (queen, rook, bishop) should not be able to pass through other pieces.
2. Good Scope: Implement checks and checkmate
    a. We want our chess program to be able to check the current state of the board and determine whether or not there is a check. Also, the program should be able to detect a checkmate, and decide that the game is over.
3. Excellent Scope: Special Moves
    a. For special moves, we want to be able to have players be able to do a castling move or en passant when the conditions are appropriate.