Plan of Attack

- July 15 - Everyone will initiate the .cc and .h files for each class.
- July 20 –
  - Bo will implement ChessPiece (and all inherited classes), Posn, GameControl, and game history.
  - Chris will implement AI and TextDisplay.
  - Chuck will implement Board, GraphicDisplay, and Observer.
- July 23 –
  - Implementation of undoing any chess move by Bo.
  - Main Function by Chris/Chuck.
- July 25 - Final Design Brief to be completed by all group members.

Implementation of 4 player chess option if time allows. Each group member is responsible for adding necessary modifications to their classes they created originally. Group members will be required to assist each other should any finish early.

Chess Questions

Question:

Chess programs usually come with a book of standard opening move sequences, which list accepted opening moves and responses to opponents' moves, for the first dozen or so moves of the game. Although you are not required to support this, discuss how you would implement a book of standard openings if required.

Answer:

Create a StandardOpening class with the following fields:

- Stack of moves in sequential order (which will represent a standard opening if played)

Create a StandardOpeningBook with:

- Vector containing all desired StandardOpenings
- A method to check if the last move made corresponds to any StandardOpening

After every move, a check will be run via the Book's (StandardOpeningBook's) method.

This check will pop top move off of each StandardOpening's stack and compare it to the actual move in game. If the move is the same, that StandardOpening is preserved and the next move in the stack could be applied by an AI (depending on AI). There is a possibly that multiple different StandardOpenings would apply and the AI would then need to choose between possible moves. If the move is not the same this openings is removed from StandardOpeningBooks vector of StandardOpenings.

Question:

How would you implement a feature that would allow a player to undo his/her last

move? What about an unlimited number of undos?

Answer:

Create a CMove class with the following fields:

- Original coordinates of chess piece that was moved
- New coordinate of chess piece that was moved
- Colour of chess piece that was moved
- Chess piece that was captured (if any)
- Castling flag
- Capture en passant flag

Create a History class with:

- A stack of CMove objects as a private field within
- An undo Method that pops a move off the stack and reverses the last move on the board

-After every move, a corresponding CMove object will be pushed onto the History object's stack

-If the user indicates they want an undo (presumably through typing a command "undo") the History object's undo method will respond appropriately.

-If no moves have been made, undo does nothing

-Last moved piece will return to its prior position

-If the last move was a castling move, logic for moving the king and rook back to their appropriate positions is run

-If the last move was an en passant capture, special logic for replacing the captured pawn is run

-If the last move was a normal capturing move, the piece that was removed is replaced at the spot the moved piece is returning from

-If a book of openings is being used by AI, the AI would shift one move back.

In the case that we are not allowing unlimited undos (only last move can be undone), there would be no need for a stack, the history class would store only one move as a field. All the other logic would remain similar.

Question:

Variations on chess abound. For example, four-handed chess is a variant that is played by four players (search for it!). Outline the changes that would be necessary to make your program into a four-handed chess game.

Answer:

The current board implementation is a vector (theBoard) that contains 8 vectors (theRows) which each contain 8 piece pointers (theColumns).Additional rows would simply be created by pushing additional row vectors to the board. Rows are customizable and need not all contain the same number of columns.

In the context of four handed chess:

- theBoard would contain 3 rows of 8 columns followed by
    - 8 rows of 14 columns, followed by 3 rows of 8 columns.
- Pieces would be created and pointed to as appropriate on this custom board
- Pieces would be able to be assigned to any of four different colour allegiances
- Turn order would be passed in the appropriate order.
- Win condition would have to be changed appropriately
- Assuming allegiances exist between players could exist, conditions for check
    - would have to change from any piece of other color being able to take attack ones king, to any piece from opposing team being able to attack ones king.
    - Capture logic would also need to be altered similarly.
- Pawn promotion logic would need to be altered
- AI would need to be overhauled