The state chart begins by turning on the game, which puts the software in the **Ready** state. From here the user/player has two choices; first, input a move (one peg that jumps another peg) or second, request the system to solve a puzzle (initially) as optimally as possible. Following the second choice, the system is now in the **Solved** state as the program has determined an optimal way in which to solve the puzzle. This path may only be taken if the game board state is in an initial configuration, meaning no moves have been input via the user. Again the user has reached a fork and must determine whether to switch of the software (**final state**) or if he/she chooses, play again (go back to Ready).

Playing again will return the system back to the Ready state. Now, back to the first set of paths. After the software has been switched on (from start state) and is in the Ready state, the user will input a pair of pegs to execute a move. The system’s state becomes, Check Move and now the system will attempt to validate
the user’s move. An invalid move will push the system into the Error state and the program will output the error (complaint) to the user. The user is then asked to re-input a valid peg pair placing the system back in Ready. If another invalid combination is submitted, the process repeats: Check Move → Error → Ready. If the user/player submits a valid move and is verified in Check Move the program determines if there are more moves that exist. If the system concludes that there are in fact more moves, the state changes to Ready and waits for more input from the user. If the system decides that last move submitted was a “final” move and there is only one peg remaining on the game board, the state changes to Solved denoting a successful finish. Conversely, for a “final” move that has more than one peg remaining on the board, the state becomes Unsolved. This means that the user has input a sequence of valid moves, but they have not yielded an optimal solution. Therefore the puzzle has reached an end but not the desired finale.

When the system has reached one of these states: Solved or Unsolved, the player is prompted via a system message whether or not he/she should play again or quit (turn off the software). If a player chooses to replay or play again the state changes back to ready and the board is reset to an initial state.

As a side note, whenever the system is in the Ready state and the user so wishes he/she may choose to clear or reset the game board to an initial state. This can only be accomplished in while in Ready. Reasoning behind the mode of operation, is simple. The user may have gotten frustrated with the current board state (as a product of their sequence of moves) or may have mistakenly made a move that afterward he/she regretted. There of course may be other reasons, but the option remains open for a player.

GUI for the Implementation
The following pertains to figure g2:
1). The main component of the interface is this section, as the game board and its current state will be displayed here. Initially graphical representation of the data structure within the program will be shown to the user. Then, as moves are made (either by the user or the system) the resulting board state will be shown, replacing the previous state.

2). This is where dialog between the system and the user will be output. Upon the input of an erroneous move, unexpected values (characters, rather than integers) entered as peg numbers, or more general errors will be relayed to the user in here. Other non-error related messages will also be passed through here as well. These messages may include
(but not limited to) successful completion of the game with a score based on number of pegs remaining. Conformation of a command, i.e. clearing the board, or solving the puzzle via the system's optimal algorithm.

3). In this section, the user will be expected to input a move via the textboxes, "Force Peg" and "Target Peg" and then press (click) the move button. When the user inputs into the textbox, they will input an integer corresponding to a peg number as displayed on the game board. If he/she inputs something other that an integer, an error will be returned. The system is responsible for validating the move once the move button is pressed. Upon a successfully input and valid move the system will update the structure within the program and display the change (updated) board state in the GUI. If the system is not able to validate the move (invalid peg numbers, pegs not present, move not possible, etc.) then an error will be returned and shown in the system messages textarea.

Other elements in figure g2 include the clear/reset button and the solve puzzle button. The clear/reset button makes itself quite obvious. It clears any previous moves from the system’s memory and reinitializes the board as a “new game”. The solve puzzle button can only be used when the game begins and a user has not input any moves, as long as these conditions hold, the system will solve the puzzle optimally.

The following pertains to figure g1:
Upon completion of the puzzle, this means no more valid moves with 1 or more pegs left on the game board. This dialog box will pop-up and ask the user whether he/she would like to continue playing (Play Again) or shut the software down (Quit). If the play again option is selected, the board and move history is cleared. Essentially, the game is reinitialized.