Name: Siddharth Vaknalli

Game: Ultimate Chess

ICS4UE Final Project – Work Log

|  |  |  |
| --- | --- | --- |
| **Date** | **Activities** | **Time taken** |
| 17/12/2014 | Initial design and drawing initial UML diagram | 1.5 hours |
| 18/12/2014 | Initial design and drawing initial UML diagram | 1 hour |
| 19/12/2014 | Finishing initial design and presenting it to Mr. Ridout | 1.5 hours |
| 21/12/2014 | Drawing the Bishop chess piece image | 2 hours |
| 22/12/2014 | Drawing the Queen chess piece image | 2 hours |
| 23/12/2014 | Drawing the King and Knight chess piece images | * 1. hours |
| 25/12/2014 | - Setting up the Piece and Square classes and writing code for a few methods (including constructors) in these classes  - Setting up all the other main classes including all subclasses of Piece with their constructors  - Writing all code required for drawing the board and its pieces at the start of the game (including setting up the basic JFrame and JPanel subclasses) | 4 hours |
| 26/12/2014 | Building on and improving code written the previous day (25/12/2014) | 1.5 hours |
| 27/12/2014 | -Writing code and testing methods for the Rook class, as well as writing the basic code for a few of the implemented Mouse methods in the JPanel subclass  - Adding extra methods and variables as and when needed to the Piece and Square classes for writing the above mentioned code | 5 hours |
| 28/12/2014 | -Writing code for the Pawn class (mainly for generating moves), which includes handling special moves such as the en passant and promotion  -Adding code to other classes to make the Pawn class work as and when required | 4.5 hours |
| 30/12/2014 | -Designing the graphics and writing code for capturing pieces and making them visible on the screen  -Writing code for the Queen class | 4.5 hours |
| 31/12/2014 | -Writing the basic code for the King class, including handling checks and generating moves according to the arrangement of the opponent pieces  -Adding code to the other classes whenever necessary | 3.5 hours |
| 1/1/2015 | Improving and finishing code for the King class and working on handling situations for moving other Pieces when the King is under check | 4 hours |
| 2/1/2015 | Finishing all code for other Pieces for handling situations for checking opponent King or defending own King in check | 3.5 hours |
| 7/1/2015 | Discussing further programming plans with my partner and improving/building on previous code written with help from my partner | 1.75 hours |
| 8/1/2015 | -In class work with my partner to help handle castling moves  -Finishing code for the Move class and implementing it in the JPanel and JFrame classes, including handling undoes  -Writing code for unchecking Kings after they have been moved out of a cheque  -Finishing code for promoting pawns, including giving the user piece choices to promote the pawn to using a dialog box | 4.25 hours |
| 9/1/2015 | -Finishing all the code for checking the King, unchecking the King and moving other Pieces to prevent/counter/block checks according to the arrangement of other Pieces in the game  -Making required adjustments to all classes for the above mentioned code  -Debugging and handling a few null pointer exceptions | 5 hours |
| 10/1/2015 | -Handling code for checkmates and stalemates and planning the code for other ways a game can end  -Adding, simplifying and adjusting other code as required, including adding a few required methods in the JPanel class  -Debugging and testing | 3.5 hours |
| 11/1/2015 | -Writing code for handling certain ways a game can be drawn, such as the fifty-move rule  -Adding an option to the JFrame to draw a game, which is enabled if a draw is possible, and writing supporting methods  -Debugging and testing earlier code | 2 hours |
| 12/1/2015 | -AI research and pseudo code planning  -Debugging and testing basic game | 4.5 hours |
| 13/1/2015 | -Writing the minimax algorithm and adding other code/methods when required  -Working on improving the efficiency and accuracy of the minimax algorithm  -Debugging and testing other code | 4.5 hours |
| 14/1/2015 | -Testing, debugging and improving the minimax algorithm  -Debugging and testing the overall game | 4 hours |
| 15/1/2015 | -Adding required code to the minimax algorithm to handle situations involving checking and unchecking kings  -Testing and debugging the AI component  -Alpha beta pruning research and planning | 5.5 hours |
| 16/1/2015 | -Alpha beta pruning on the minimax algorithm, and adding other code necessary to properly run the pruning process  -Debugging and testing the AI and improving its quality/efficiency  -Improving/designing original graphics for the game | 6 hours |
| 17/1/2015 |  | 6 hours |
| 18/1/2015 | - Working on graphics, including creating those for the main screen, choice of color and difficulty screens, side panels, chess board frame and numbering. Coding the display of appropriate messages correctly  -Debugging and testing | 8 hours |
| 19/1/2015 | - Improving and perfecting castling in both the single and two player versions of the game  - Adding graphics, such as undo, new game, statistics and quit buttons, and writing code for their function  - Debugging and testing all parts of the game  - Preparing testing report for beta testing | 5 hours |
| 20/1/2015 | -Finalizing game for submission  -Debugging and testing | 3.5 hours |
| 21/1/2015 | -Commenting remaining sections of code  -Fixing a few bugs  -Submission of game | 4 hours |

Summary:

I mainly contributed to setting down the layout and design for the project, which included making UML diagrams and designs for all the classes and their methods/variables. Further, I set up the basic code for most classes in the project, which was then later built on by me and my partner. This included writing the base code for the Piece class and all its subclasses (such as constructors, generating moves etc.). I also wrote the main code for putting and removing Kings in and out of checks as well as well as for restricting the moves of pieces when in checks. I wrote the code for checking end game scenarios (wins and draws), and the code in the JPanel class for undoing moves and drawing games. For the AI component, I researched the minimax algorithm and implemented its code, in addition to alpha-beta pruning it. I wrote all the code for the Move class, which includes the method for undoing moves. Finally, I worked on most of the original graphics in the game, including images for the side panels, board squares and the game logo.

From my point of view, I and Sean worked quite well together as a team. We planned and divided the tasks between ourselves well and collaborated well in class and during the Christmas break by communicating well with each other and using technology (Google docs) to work together. We were able to adjusting our thinking and programming styles well to suit each other. Whenever there was a difference of opinion or conflicting views, we were able to sort it out relatively well by looking at the solution best suited for the interest of the game. We should improve on our timings and adjusting each other’s schedules so that we can spend more time coding and thinking on solutions together.

My contribution: 65 – 70%

Sean’s contribution: 30-35%