**Report**

**Introduction 595 Words**

We began our project by looking at a number of tile-based games for our initial research. This involved looking at checkers, battleships, chess, memory games etc. and evaluating each games feasibility based on our current skills in visual studio and the time frame we had been given. This led to us agreeing to make a checkers or draughts game for 2 players. We made sure we were both familiar with the rules of the game before we began our initial design.

**Approach/Solutions**

After completing our design, we began working on the actual checkers board. We wanted the board to be grey and red and start the players off with 12 checkers each. The approach we took to making the actual tiles was creating a 2D array of buttons using code in the main form.cs file. This involved a method with 2 for loops that traverse the array of buttons setting the background of each button to either a red tile, grey tile, black counter or red counter. We decided to make all our buttons code generated. Our Menu was simple and involved 3 buttons, a play button, instructions button and a quit button. The play button opened the main checkers form and the instructions button opened an instruction form.

Once we had the basics completed (initialised game board and menu) we began work on the more complicated methods in the game. In terms of movement for our game we had two main methods for this, CanBlackMove() and CanRedMove(). These methods basically check to see if the counter can move based on what is diagonal to it. Keeping tracks of each users turn was handled by a method called swithTurn() this basically checks to see if a variable called blackturn is true and sets the turn based on that, this is displayed to the user along with the score of each player on the side of the screen. The method CankingMove() is handled in a similar method to CanBlackMove() except the king can move forward and backwards so we had to make code to accommodate this. In terms of winning, the number of counters for each player is constantly kept track of and this made it easy to design and implement finding a winner.

**Problems**

The first main problem we faced was actually trying to create a circular button but it turns out this isn’t directly possible in visual studio, which is why we had to resort to using background images we created ourselves for the counters.

Getting each background image to align properly on the buttons was also an issue, especially for the when we wanted to indicate to the user that they had selected a counter. We got around this by simply outlining the counter on the image itself as instead of the whole image.

Another issue we faced was the capturing of multiple counters in one move which is possible in a game of checkers in real life. This was very difficult to implement as we had to account for multiple diagonal moves at once which proved to be as time consuming as it was complicated.

**Conclusion/Future Ideas**

In conclusion checkers was very challenging to implement and at times frustrating in visual studio, however due to solid teamwork and organisation we felt we did a good job completing this assignment. In terms of future ideas, if we had more time we would have spent more time on the actual U.I, making it look better because although we feel it looks good enough as it is, it could look nicer.