Programming of this project was done in Java using the Eclipse tool environment.

Examples where Alpha Beta gives savings over MinMax (here depth searched is 3)

1. MiniMaxOpening

Input Board is -XXXXXXXXXBXXBXXXWXXXXXX

Output Board is -XXXXXXXXXBXXBXXXWXXXXXW

Minmax Estimate - 0

Positions Evaluated - 14512

ABOpening

Input Board is -XXXXXXXXXBXXBXXXWXXXXXX

Output Board is -XXXXXXXXXBXXBXXXWXXXXXW

Minmax Estimate - 0

Positions Evaluated – 2976

1. MiniMaxGame

Input Board is - XXWXWXBXXBXXBXXXWXBXXWX

Output Board is -XXWXWXBXXBXXBXXXXXBWXWX

Minmax Estimate - -12

Positions Evaluated – 3046

ABGame

Input Board is -XXWXWXBXXBXXBXXXWXBXXWX

Output Board is -XXWXWXBXXBXXBXXXWXBXXXW

Minmax Estimate - -12

Positions Evaluated - 436

My improved static estimation function is as follows -

A static estimation for MidgameEndgame:

if (numBlackPieces ≤ 2) return(10000)

else if (numWhitePieces ≤ 2) return(-10000)

else if (numBlackMoves==0) return(10000)

else return ( 1000(numWhitePieces + numPossibleMills − numBlackPieces) - numBlackMoves)

A static estimation for Opening:

return (numWhitePieces + numPossibleMills − numBlackPieces)

where-

numPossibleMills is the number of possible mills at the current board.

If we play the game for white, this estimation will give better results because there will be better chances of white winning the game if it has better chances of making a mill in the next turn. The more the chances are of making a mill over the next move the better the chances will be of winning the game. So my estimation function accounts for this factor in opening as well as the midgame/endgame phase. Thus providing a better result.