Journal of Computer Science 5 (4): 283-289, 2009
ISSN 1549-3636
© 2009 Science Publications

Real-Time Connect 4 Game Using Artificial Intelligence

[1]Ahmad M. Sarhan, [2]Adnan Shaout and [2]Michele Shock
[1]Department of Computer Engineering,
University of Jordan, P.O. Box 17966, Amman-11195, Jordan
[2]Department of Electrical and Computer Engineering,
University of Michigan-Dearborn, USA

Abstract: Problem statement: The study presented a design that converted connect 4 game into a real-time game by incorporating time restraints. **Approach:** The design used Artificial Intelligence (AI) in implementing the connect 4 game. The AI for this game was based on influence mapping. **Results:** A waterfall-based AI software was developed for a Connect 4 game. **Conclusion:** A real time connect 4 game was successfully designed and implanted with GUI using C++ programming language.

Key words: Real-time system, connect 4 game, search techniques, AI techniques, influence mapping

INTRODUCTION

Connect 4 was first published by Milton[1]. The waterfall model was used in the creation of the software[2]. The concept of the Connect 4 game is to get, before your opponent, four chips in a row, arranged either diagonally, vertically, or horizontally.

There are many ways to solve the Connect 4 game. There are several levels of AI difficulty as in[3]. They are the random, defensive and aggressive AI. Random, as the name implies, randomly picks a play and is the easiest to beat. Defensive AI makes blocking a win a priority, while aggressive AI makes winning a priority. Both are harder to beat than the random AI.

Solution algorithms for AI are numerous and can be complex. Some that were considered are minimax, minimax with alpha-beta pruning, A* and influence maps. Minimax is a recursive tree that uses backtracking to find the optimal move and is the hardest to beat[3-5]. The opponents are referred to as MIN and MAX. MIN tries to minimize MAX’s score and MAX tries to maximize his score[6]. The algorithm then uses this rule and looks several moves ahead for the best move possible. Minimax may be the best way of getting the optimal move, but it requires a lot of processing, so pruning methods are used[5]. The pruning method that was considered is the alpha-beta method. Since minimax looks at all possible plays including ones that can be ignored, alpha-beta pruning is used to improve the efficiency of the minimax algorithm[6]. It scores the possible plays and if it is at a MAX node, it only looks down the branches that have a score greater than or equal to that of the MAX node. Furthermore, if it’s a MIN node, it looks for a score that is less than or equal to the MIN node[6]. Even with these pruning methods found in[4-6], this type of AI is too complex. The A* algorithm was also looked at. A* is a best first search that combines the path cost from the start to the current and the estimated cost from the current to the end of the cheapest path[7]. The A* algorithm may not be a fit for this real time game because the A* algorithm may not meet the game deadline.

The game has been solved by Allen, using a brute-force approach[9]. Using a knowledge-based approach, the game has also been solved by Uiterwijk *et al*.[10] and by Allis in his Master's thesis[11].

In[11], Allis wrote a program called VICTOR. In combination with conspiracy-number search, search tables and depth-first search, VICTOR was able to show that White can win on the standard 7□6 board. It was also shown that, by using a database of approximately half a million positions, VICTOR can play real time against opponents on the 7□6 board, always winning with White.

VICTOR is very similar to Velena, which was designed by Bertoletti and was based on eight mathematical rules[12]. Velena is a Shannon-C type expert system written to play Connect 4. Allis also showed that the game is a first player win. Velena is always able to win if she plays first[10].

A neural-network approach to the game was designed by Schneider *et al*.[13]. The system employed the multilayer perceptron architecture which learns through the supervised backpropagation algorithm. The

Corresponding Author: Ahmad M. Sarhan, Department of Computer Engineering, University of Jordan, P.O. Box 17966, Amman-11195, Jordan

required knowledge for training was obtained from saved games.

This game was implemented as a real-time game by Shaout and Shock[14]. The program was written in C using LabWindows/CVI by National Instruments. It was written for a windows environment. It was a 2 player game and had a 5 sec timer for play. The plays were made on an interactive GUI. The current player was displayed on the top of the GUI. Once the current player makes a play, the chip is placed in that column within 1 sec. When the chip is in place, the timer is reset. When the timer expires, the current player loses his/her turn. In[14], the computer player was not implemented, however.

MATERIALS AND METHODS

The algorithm used in this study is based on using heuristics with influence mapping that is seen in[8]. This was chosen because it is the method that fits in with the original code for the game. It is a greedy algorithm and may not return the optimal move, but it meets the requirements of the system without too much overhead. In[8], there are influence maps and heuristics of Tic-Tac-Toe and Pente, a larger scale Tic-Tac-Toe. These methods look at and evaluate the entire board, whereas in Connect 4, only the top available spot in each column needs to be looked at. The algorithm used in this study uses the basics from[8], transforms them into use for Connect 4 and integrates the aggressive, defensive and random AI from[5].

The layout is a grid of six rows by seven columns[1]. This layout is shown in Fig. 1 and the original design flow is shown in Fig. 2. When it is the player's turn to play, he will click on one of the boxes above the circle LEDs to select the column for chip placement. The chip will then be placed at the bottom of the chosen column. After each play, the program checks for a winner or a tie.

The program starts with the screen shown in Fig. 3. This screen gives the player (s) a choice of one player, two players, or to quit the game.

Fig. 4 shows the flow of start and difficulty screens. The program allows the players to choose to have a timer or not to have one.

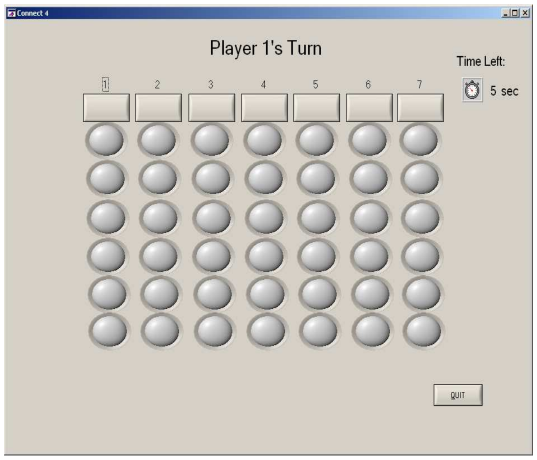


Fig. 1: Board layout

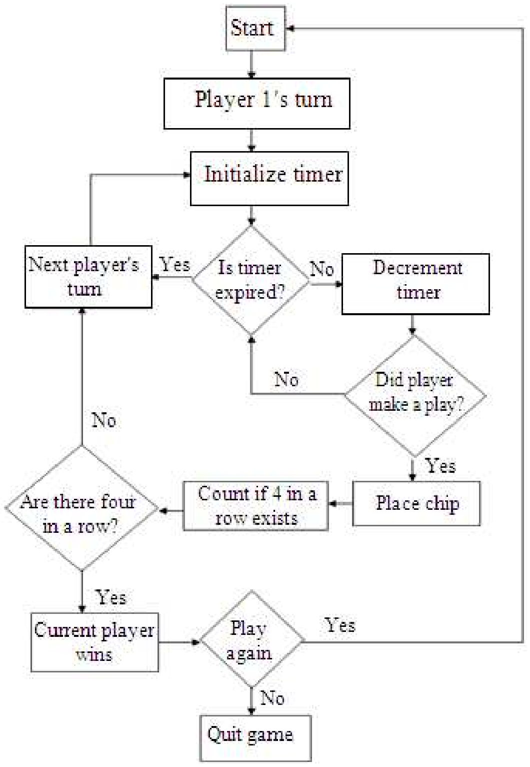
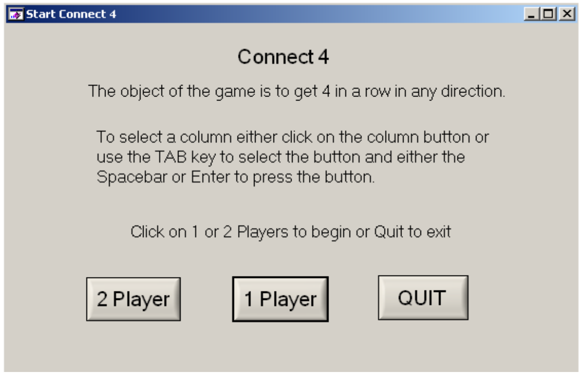


Fig. 2: Original software design flow chart



Connect 4

The object of the game is to get 4 in a row in any direction.

To select a column either click on the column button or use the TAB key to select the button and either the Spacebar or Enter to press the button.

Click on 1 or 2 Players to begin or Quit to exit

Fig. 3: Start screen

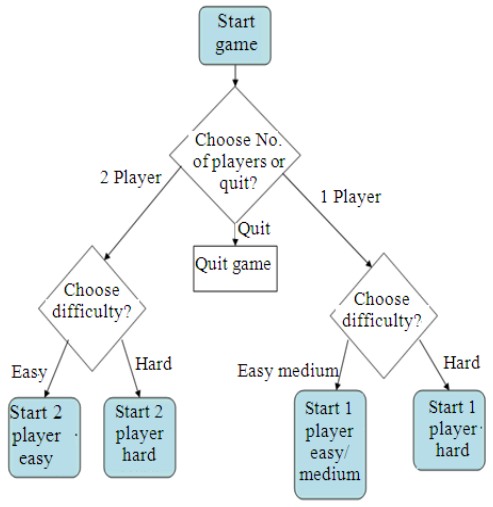
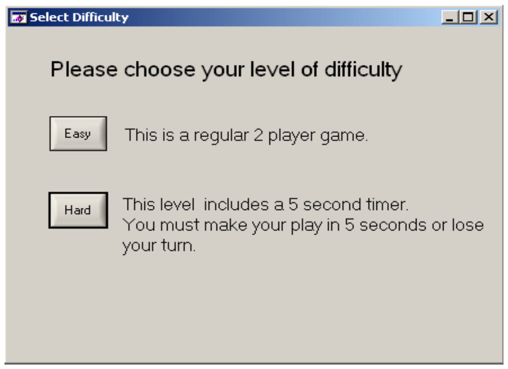


Fig. 4: Software flow of start and difficulty screens


Select Difficulty

Please choose your level of difficulty

Easy This is a regular 2 player game.

Hard This level includes a 5 second timer. You must make your play in 5 seconds or lose your turn.

Fig. 5: Two player difficulty screen

Figure 5 shows the difficulty choice screen and Fig. 6 and 7 show the flow of the Easy and Hard modes, respectively. The easy level has no timer and allows for a more leisurely game. In this mode, the timer is invisible and disabled. The program will wait until the current player makes a move or selects quit. After a play is made, it checks if that player has won or caused a tie

The hard level, on the other hand, initiates the timer and makes it visible to the players. It plays the same as the two player easy mode except for the lost turn when the timer expires.

The one-player game has three levels to choose from. The difficulty screen for the one-player option is shown in Fig. 8. It allows the player to choose from an easy, medium and hard levels. The Easy mode has an AI that is easier to beat. The Medium mode includes the AI from the easy level and is modified, as shown in Fig. 9, making it harder to beat.

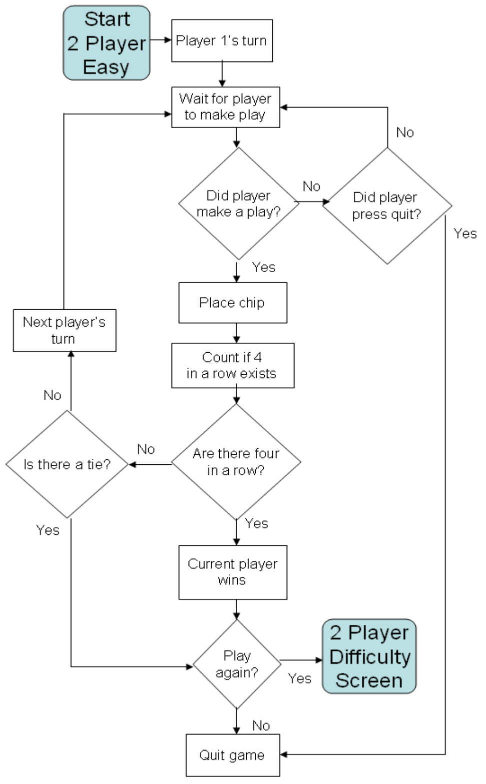


Fig. 6: Software flow for two player easy mode

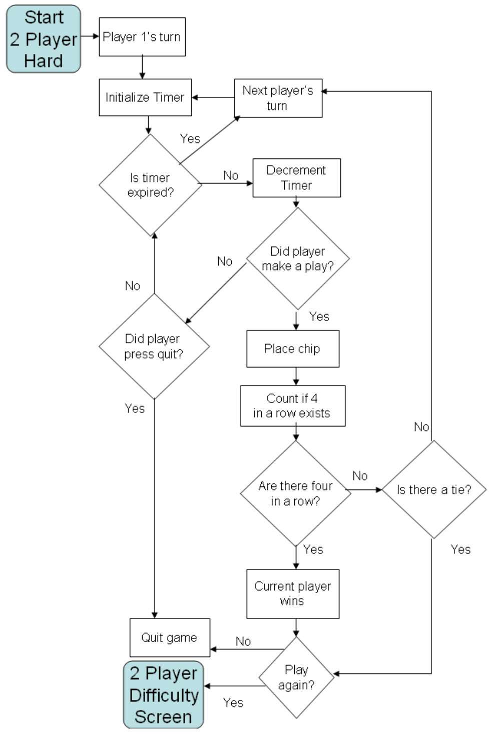
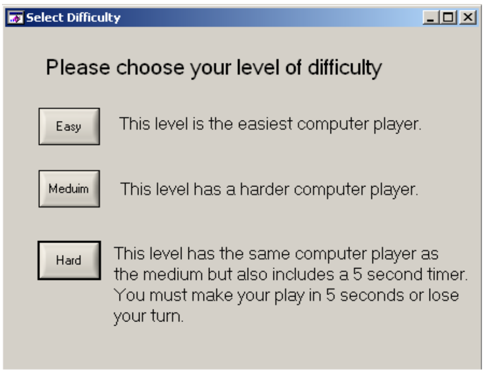


Fig. 7: Software flow for two player hard mode


Select Difficulty

Please choose your level of difficulty

Easy — This level is the easiest computer player.

Meduim — This level has a harder computer player.

Hard — This level has the same computer player as the medium but also includes a 5 second timer. You must make your play in 5 seconds or lose your turn.

Fig. 8: One player difficulty screen

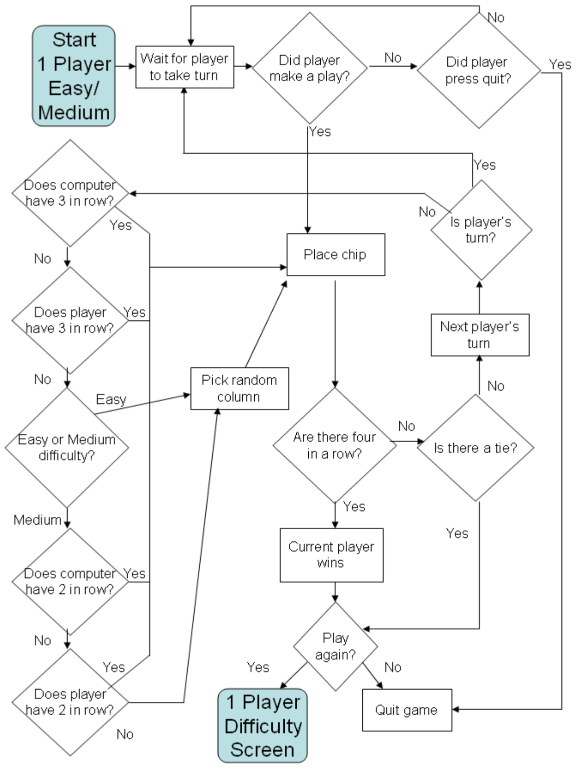


Fig. 9: Software flow for the one player easy/medium mode

The Hard level has the same AI as in the Medium level, in addition to a timer to step-up the skill level needed to beat the computer, as shown in Fig. 10.

All of the difficulty levels have the same algorithm to score the possible moves. There are only seven possible moves, the bottom spot available for each column.

The score is stored in a double array and it keeps track of both the computer player's score and the human player's score for each available move.

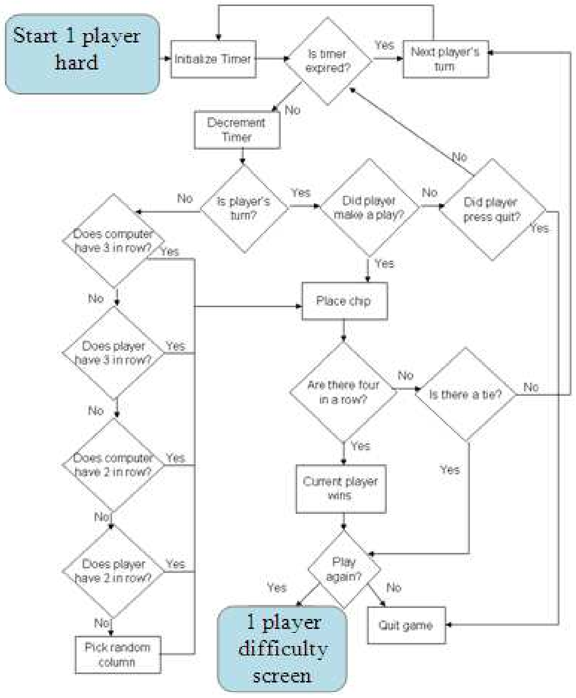


Fig. 10: Software flow for one player hard mode

It starts by looking at the horizontal direction for the move it is scoring. It examines the space to the left of the move (as long as the spot is on the board, it does not check spots that are off the board) and sees if the player it is scoring has a chip there. If that player has a chip there, the count is increased. It continues to count until either the space is not occupied for the player being scored or it reaches the end of the board. It then looks to the right and does the same thing.

For example, while scoring the computer player, the move it is examining has a chip to the left that is the computer's and a chip to the right that is the human player's. It sees the chip on the left and increases the count to 1. It then looks another space to the left and sees it empty and stops looking left. It then looks right and sees that the other player has a chip there, so it stops looking to the right. The score for this move is 1. Scoring is then done in the vertical direction. Each possible move looks down until it hits the other players chip or the bottom of the board. If the score for this column is higher than the score for the horizontal direction, the horizontal score is replaced with the vertical score.

The moves are then scored in the left and right diagonal directions. The scoring algorithm checks down and up the diagonal for the number of chips in a row that are connected to that move until they hit the other player's chip, an empty space, or the end of the board.

The AI for the easy level starts with an aggressive move. It takes the scores for the computer and checks if there is a score of 3 or more, which would be a win the game. If there is a score of 3, it picks that column. If there is no move, it makes a defensive move by looking at the score of the player to see if they have a score of 3 or more. If neither the computer nor the player has a score of 3 or more, a random column is selected.

The medium level is built off the easy level. Like the easy level, it first looks for the computer or the player to have a score of 3 or more. If a move is not found yet, the AI will make another aggressive move by looking to see if the computer has a score of 2 anywhere. This will allow the computer to get closer to a win and force the player to block instead of working toward a win. If that move does not exist, it will then play another defensive move by blocking the 2 in a row that the player has. Finally, if neither player has 2 in a row, a random column is chosen. Fig. 9 shows the software flow for the easy and the medium levels.

Picking the hard level gives you the same AI that the medium level has. What makes it harder is that a 5 sec timer is added. The player has 5 sec to make a play after the computer plays. This requires quick thinking in order to beat the computer.

A delay was added for the computer player so that the move could be seen easily by the human player. Fig. 10 shows the software flow for the hard level.

RESULTS

The improvements in this version of the game over the older version[14] include the following new additions:

- Timer: In this version, the timer is made an option whereas in the old program, the timer was always activated
- Computer player AI: There is now a choice between one and two players, whereas the old version was just a two-player game

Implementing these changes required the addition of two new screens and updating the start screen. The start screen now has two buttons that start the game (Fig. 3), instead of one. Figure 5 and 8 are new screens added to implement the difficulty levels and the timer.

DISCUSSION

The game was implemented using C++ running on Windows platform. A brief description of the functions used to compose the program is shown in Table 1.

Table 1: Description of the Main Functions

Function name	Operation
Initialize variable ()	Initialize global variables that are constant
Pick best play ()	AI for computer player. Allows for the program to pick column for different difficulty levels
Horiz score ()	Used only with the computer player, scores the computer and player in the horizontal direction to find the best move
Vert score ()	Used only with the computer player, scores the computer and player in vertical direction to the best move
Left diag score ()	Used only with the computer player, scores the computer and player in left diagonal direction to find the best move
Right diag score ()	Used only with the computer player, scores the computer and player in right diagonal direction to find the best move
Change player ()	Switches the current player
Is horiz win ()	Returns 1 if there is 4 in a row in the horizontal direction
Is vert win ()	Returns 1 if there is 4 in a row in the vertical direction
Is leftdia gwin ()	Returns 1 if there is 4 in a row in a left slant direction
Is right dia gwin()	Returns 1 if there is 4 in a row in a right slant direction
Is winner ()	Returns a 1 if there is a winner. Also, suspends the timer if there is a winner
Load board ()	Places a 1 for player 1 and 2 for player 2 in the spot chosen by the current player
Start program ()	Called when a player chooses to start the game with 2 players from the initial start-up panel
Start computer program	Called when a player chooses to start the game with 1 player
Play again	Called when the player chooses to play the game again, after there is a tie or a winner.
Quit game	Called when player chooses to quit the game again after there is a tie or a winner
Timer expired	Called when the timer clicks every second and then will change the display to decrement the time shown
Quit call back	Called when the quit button is pressed and ends the game
Hard setting ()	Called when the hard setting is chosen in the 1 player mode
Medium setting()	Called when the medium setting is chosen in the 1 player mode
Easy setting ()	Called when the easy setting is chosen in the 1 player mode
Hard2 player ()	Called when the hard setting is chosen in the 2 player mode
Easy2 player ()	Called when the easy setting is chosen in the 2 player mode
Column1()	Called when player selects the first column and depending on what row is currently available, it stores the selection in the board
Column2()	Called when player selects the second column and depending on what row is currently available, it stores the selection in the board
Column3()	Called when player selects the third column and depending on what row is currently available, it stores the selection in the board
Column4()	Called when player selects the fourth column and depending on what row is currently available, it stores the selection in the board
Column5()	Called when player selects the fifth column and depending on what row is currently available, it stores the selection in the board
Column6()	Called when player selects the sixth column and depending on what row is currently available, it stores the selection in the board
Column7()	Called when player selects the seventh column and depending on what row is currently available, it stores the selection in the board

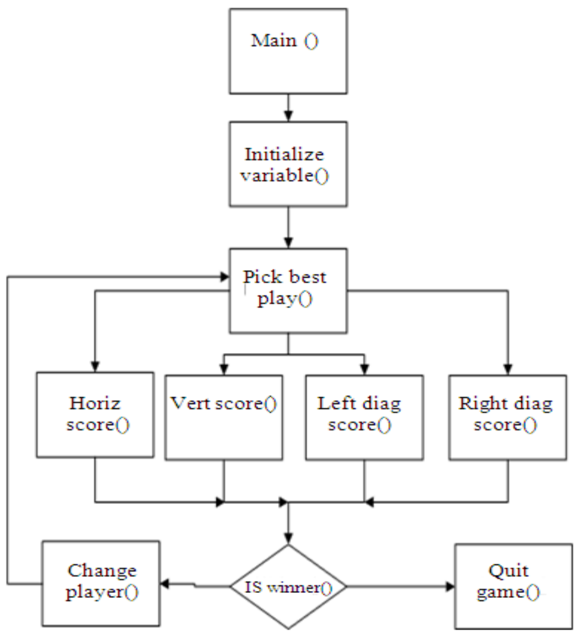


Fig. 11: Simplified main functions calls

The interactions of the Main functions are shown in Fig. 11. The complete C++ Code of this implementation can be requested from the researchers.

The test procedures consist of testing the game in all the modes, as follows:

Select the 2 player mode:

- Select easy mode and play the game. Make sure that there are no bugs and that it flows as if you were playing with a live board and chips
- Select the hard mode and make sure that it flows like the easy mode except for the timer. Make sure that when the timer expires that the current player changes. Make sure that when the current player makes a move that the next player is made current and the timer resets

Select the 1 player mode:

- Select the easy mode, play the game and see where the computer chooses to play. It should try to win or block three in a row, but randomly pick any other play. Run with the debugger and make sure that it never gets to part of the AI that is not intended for it
- Select the medium mode, play the game and see where the computer chooses to play. It should try to win or block three in a row, create three in a row or block 2 in a row, then choose randomly. Run with the debugger and make sure that it never gets to part of the AI that is not intended for it
- Select the hard mode and run the same test as the medium mode and add in a test for the timer. Make sure that when the timer expires that the current player changes. Make sure that when the current player makes a move that the next player is made current and the timer resets.

For this game system, there were several black box testers. Their feedback consisted of:

- Add a delay to computer player, it is hard to keep track of where the computer plays when it is right after the player makes a play. After the delay was added, it was noted that the delay might be too long
- When the game is over and the splash screen asks to play again. If the player chooses to play again, the game should either re-start the game at the same level, or go back to the screen were it asks for the difficulty level, but it should not go back to the main screen were it has to choose the number of players

There was also a white box tester. This tester gave the following feedback:

- The game first recognizes that player 1 won and then it continued playing even when the player has won. This resulted in two messages, one that says that the computer won and the other one says that player 1 won

These items were then investigated, corrected as needed and retested. The testers found no further bugs that needed to be resolved.

CONCLUSION

AI for Connect 4 game was integrated into a real-time version using the waterfall software development model. The game was updated to include the choice of one or two players. If two players were chosen, there was a choice of having a timer. If one player was chosen, there was the choice of easy AI, medium AI or hard AI. The only level in the one player mode that has a timer is the hard level. Testing verified that the requirements were met. Minor details were overlooked and they were found and fixed during testing.

REFERENCES

1. Milton Bradley, 1974. Connect Four (also known as Plot Four, Find Four, Four in a Row and Four in a line), Wikipedia; http://en.wikipedia.org/wiki/Connect_Four
2. Wikipedia, 2009. Waterfall Model. http://en.wikipedia.org/wiki/Waterfall_model
3. Adams, R., E. Ibsen and C. Zhang, 2003. A Connect Four Playing AI Agent: Algorithm and Creation Process. http://www.ccs.neu.edu/home/eclip5e/classes/csu520/SmartConnectFour-Final_Paper-v1.0.doc
4. Pomakis, K., 2005. Connect-4 Algorithm; http://www.pomakis.com/c4/connect_generic/c4.txt
5. Labib, B., 2003. Connect4 using Alpha-Beta Search algorithm. http://www.codeproject.com/KB/mobile/Connect4AB.aspx
6. Padhy, N.P., 2005. State Space Search: Implementation and Applications. In: Artificial Intelligence and Intelligent Systems. Oxford University Press, New York, pp: 161-171.
7. Russell, S.J. and P. Norvig, 1995. Informed Search Methods. In: Artificial Intelligence: A Modern Approach. Upper Saddle River, Prentice-Hall, New Jersey, pp: 96-98.
8. Matthews, J., 2000. A* for the Masses. http://www.generation5.org/content/2000/astar.asp
9. Allen , J., 1989, A Note on the Computer Solution of Connect-Four. In: Heuristic Programming in Artificial Intelligence: The First Computer Olympiad, Levy, D.N.L. and D.F. Beal (Eds.). Ellis Horwood, Chichester, ISBN: 13: 978-0470216590, pp. 134-135.
10. Uiterwijk, J.W.H.M., L.V. Allis and H.J. Van Den Herik, 1989. A Knowledge-Based Approach to Connect-Four. The Game is Solved!. In: Heuristic Programming in Artificial Intelligence: The First Computer Olympiad, Levy, D.N.L. and D.F. Beal (Eds.). Ellis Horwood, Chichester, ISBN: 13: 978-0470216590, pp: 113-133.
11. Allis, V., 1989. A Knowledge-based Approach of Connect-Four, The Game is Solved: White Wins. MS. Thesis, Vrije University, Amsterdam, The Netherlands.
12. Velena, B.G., 1997. A Shannon C-type program which plays connect four perfectly. http://www.ce.unipr.it/~gbe/velena.html
13. Schneider, M.O. and J.L. Garcia Rosa, 2002. Neural connect 4-A connectionist approach to the game. Proceedings of the 7th Brazilian Symposium on Neural Networks, Brazil.
14. Shaout, A., 2007. Real-time connect 4 game. http://www-ersonal.engin.umd.umich.edu/~shaout/connect4.pdf