Whitt Kinley and Simeon Fritz

Dr. Hooper

CSC 3400

3 December 2016

Connect 4 Final Project

We decided to work on a Connect 4 game for this project. We made this decision because we already had a solid understanding of how the game works, so we believed that we would be able to more efficiently write the appropriate code and algorithms in Python that would be needed to successfully complete this project.

We started this project by examining the Connect 4 games that we had worked on earlier in the semester, and attempt to understand the various aspects of the programs to understand how they were implemented to play the game correctly. First we had to determine how to write algorithms that would correctly play the game. These algorithms include knowing which columns still have space available, knowing if one of the players has won, and knowing if the game is over from a tie. We found that by using nested for and if statements it was not difficult to determine wins when the winning pieces were vertical or horizontal. The more difficult part came when trying to determine wins on a diagonal. However, by carefully thinking through the various actions required to assess pieces on a diagonal and by doing some research, we were able to come up with a method that correctly assess the state of the game after each move. This implementation always accurately determines the winner if there is one (Norvig).

During our initial work, we used a print method that allowed us to play the game directly in the console with a visual printout of the game state. This allowed us to efficiently test a lot of varying situations to make sure our code was working correctly. Moreover, we only allowed for a structure in which two people can play against each other.

The second phase of the project was to implement a graphical user interface to allow the player the ability to choose a move by simply clicking on the column in which he would like to drop a piece. We used TKinter to create the graphical user interface (TKinter). We created three buttons that the player can click; One for each player that allows the computer to make a move for that player, and one that clears the game board and starts a new game. Figuring out how to obtain the desired action from each click took some work, but by looking at other examples of and researching on the TKinter information page, we eventually achieved our goal. During our initial pass through, we created a variable that would pick a random number between 1 and 7 to start the basis for our computer player.

Our final hurdle was to create an algorithm that would allow the computer player to make smart moves. While doing this, we realized that it would most likely have been easier to complete this step before we created the graphical user interface. After much research and work, we were able to implement an algorithm that allowed the computer to play with some intelligence. To accomplish this, we used the alpha beta search algorithm from the aima python code (AIMAcode). To use this algorithm, we created a utility method that passed the alpha beta search the required information. This resulted in a return statement of either 1, -1, or 0 depending on the results of the search. The depth of the search that alpha beta performs is determined by a variable that can be changed in the GUI class. So far, we have been able to get the level of the algorithm intelligence to a point where it will look ahead and block the opponent when presented with an immediate threat of three in a row. Therefore when you pit Player 1, the random computer player, against Player 2, the intelligent computer player, Player 2 will almost always win.

One of the additions we want to accomplish in the future to further improve the performance of this Connect 4 game is to implement the use of heuristics. The benefit of using heuristics in a game of Connect 4 is that the AI can rank the quality of each possible move based on that move’s likelihood to result in a win. For example, the best possible first move is to place a token in center column, as opposed to picking a column on the sides. The center column has more future opportunities resulting in obtaining 4 in a row than a side column will. Once there are pieces on the board, the locations of each of the players will have an impact on the rank that the heuristic assigns to each possible move. For example, if the computer has two options that each would result in obtaining 3 in a row, but one of these options also results in blocking the opponent the heurist will assign this option the higher rank. Thus, once this heuristic is implemented, the computer player will likely be significantly smarter than the average player, and will be able to win the game the majority of the time.

The final addition we would like to implement is alpha beta pruning. In the current version, it is very inefficient to use an alpha beta depth of more than five or six. As the depth grows, the game becomes exponentially slower. If we were to implement alpha beta pruning, we would be able to allow the computer to search much deeper while still running efficiently and returning a response relatively quickly.

A couple of the things we learned while working on this project include the importance of thoroughly understanding the implementation of each aspect of a project, as well as how it will interact with the rest of the project. Creating useful algorithms depends on carefully thinking through what you want to accomplish, and how your tools can allow you to do this.

**Works Cited**

AIMAcode. "Aima-python." *GitHub*., 27 Sept. 2016. Web. 01 Dec. 2016.

Allen, James Dow. "Lessons in Expert Programming (C Language)." *Lessons in Expert Programming*., Nov. 2008. Web. 01 Dec. 2016.

PascalPons. "Connect 4 Solver." *GitHub*., 01 Oct. 2016. Web. 01 Dec. 2016.

Russell, Stuart J., and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice Hall, 1995. Print.

Pearce, Adam. "Connect 4 AI: How It Works." *Connect 4 AI: How It Works*., 19 July 2014. Web. 01 Dec. 2016.

Pearl, Judea. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, MA: Addison-Wesley Pub., 1984. Print.

"TKinter." *TkInter - Python Wiki.*, 08 June 2014. Web. 01 Dec. 2016.

Tromp, John. "John's Connect Four Playground." *John's Connect Four Playground*., 15 Jan. 2016. Web. 01 Dec. 2016.