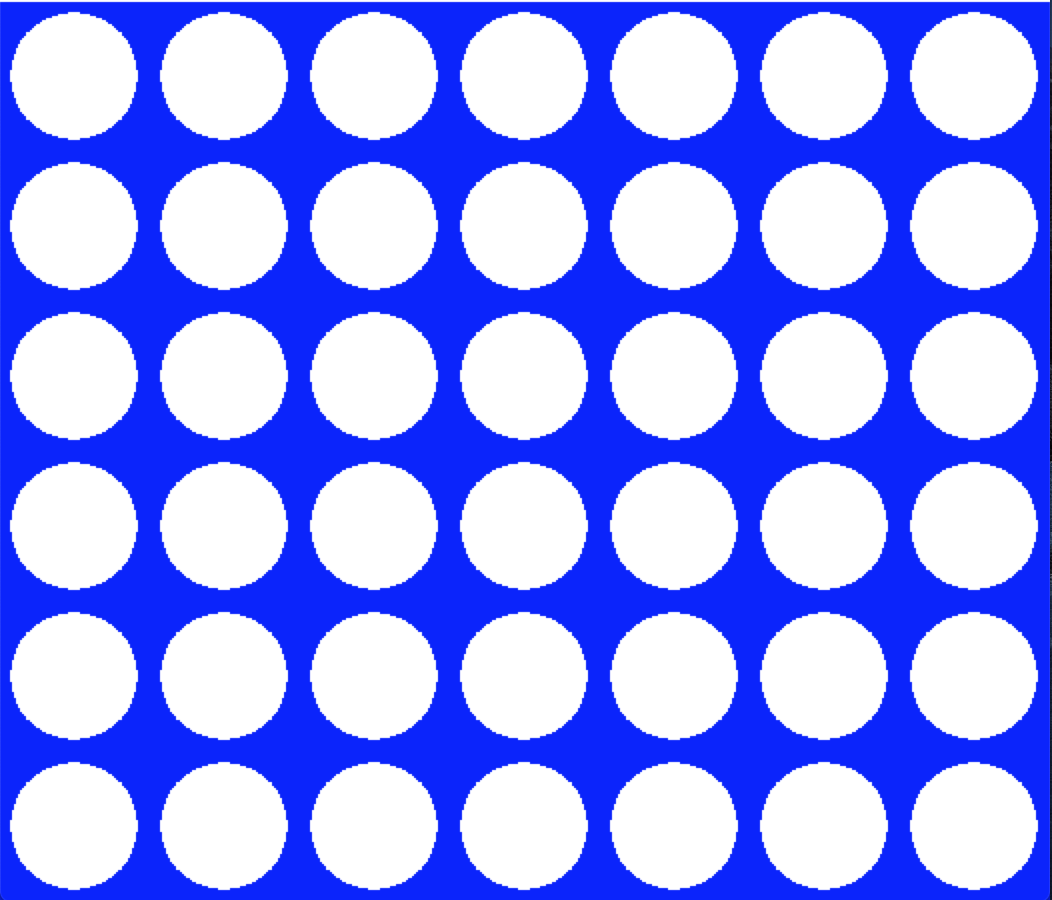
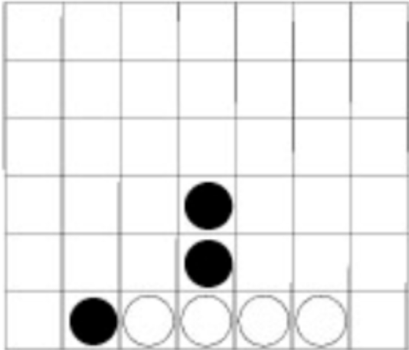
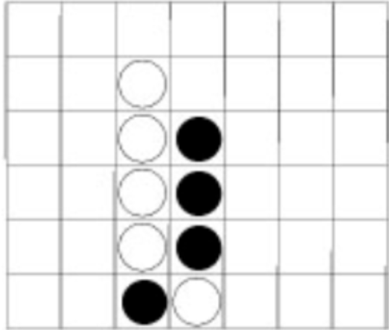
1. **Introduction**
   1. Description of Game
      1. The problem I chose to explore was the game of Connect Four because it was one of my favorite games to play as a child. I also chose this problem because of its simplistic rule set so that the agent algorithm would be easily implemented without a lot of time and space complexity added.
      2. Connect4 is a deterministic and static game as there are no random elements at work and no time constraints placed upon the agents.
   2. Rules of Connect Four
      1. Connect Four is a two player game where each player is allotted 21 identical circular pieces. The color of the players’ pieces does not matter but in this implementation the AI agent is assigned black and the human player is assigned red. The game is played on a rectangular board with 7 separated vertical columns with a depth of 6 so there are a total of 42 slots within the board as shown below. If a piece is dropped into a column it will fall to the first open row within that column. This is considered one move for the player.

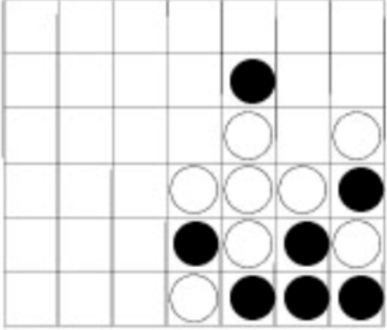


7

* + 1. The starting player is picked or assigned at random and the players take turns dropping their pieces into the board. Both players’ goal is to connect four of their colored pieces vertically, horizontally, or diagonally as shown below where the white has won horizontally(ii.1), vertically(ii.2), and last diagonally(ii.3). If both players have exhausted their 21 pieces the game ends as a draw.

 **ii.1 Horizontally**

 **ii.2 Vertically**

 **ii.3 Diagonally**

1. **Problem Solving Method**
   1. Minimax with alpha beta pruning
      1. After researching the possible algorithms to use it was decided that the minimax with pruning method would work the best since it is a static deterministic game where possible moves are able to be searched an created down to each possible winning or losing move. Because there are 4,531,985,219,092 possible states in the board it is best to only enumerate possible states from the current state of the board from the intelligent agent’s perspective and prune out possible terminal loss states.
      2. The depth that the algorithm is set to search down to can be manipulated but the best tested depth without slowing down the game too much in order to create a good playing environment as well as a good agent was 6.
      3. The way this minimax alpha beta prune works in this connect four is to set the agent (the computer) as the maximizing player and the human player as the minimum player. The maximizing player will search the next 6 possible moves including the minimizing player. It will simulate that the minimizing player is a ‘perfect’ player and that it is trying to minimize the agents moves and trying to deny it a win. If the maximizing player finds a winning terminal state within its depth of search meaning a win it will choose that path to as its next move. If it finds a losing terminal state within the path it will ‘prune’ that branch of moves out optimizing the number of states being explored saving time and space. If it finds a draw it will ignore that state.
      4. The heuristics were found by enumerating possible choices for the next depths. If the move would result in connecting four pieces adjacent to or diagonally to each other 100 points was added to the total for that branch. If the move connected 3 pieces 4 points were added to the score. If it connects 2 pieces it was given a score of 2. If the move connected 4 opponent pieces the score had 3 points subtracted from it. It did this across the board being examined in windows of length 4 in all spots on the board moving that window to account for all horizontal, vertical and diagonal sections. When the recursive call was finished the agent chooses the path with the highest total score or the branch with a terminal win state.
2. **PEAS Description of the environment**
   1. Performance Measure
      1. Put four pieces adjacent to or diagonal to each other and prevent the opponent from doing the same.
   2. Environment
      1. Game Board
      2. Agent’s Pieces
      3. Opponent’s Pieces
   3. Actuators
      1. Piece dropping function
   4. Sensors
      1. Access to the current state of the board
3. **Implementation**
   1. Language of choice
      1. The language chosen for this implementation was Python 3.7. The reason for this choice was ease of use and the use of PyGame for the GUI.
      2. It is acknowledged that python takes some time and space hits for some of the built in methods and classes however it was decided for ease of implementation that this would be ignored.
   2. Representation of Board and Pieces
      1. *Board*
         1. The board data structure representation is represented by a nested list of 6 where each list is a row of the board as shown below. Where each 0 represents an empty slot in the board.
            1. [[0, 0, 0, 0, 0, 0, 0]

[0, 0, 0, 0, 0, 0, 0]

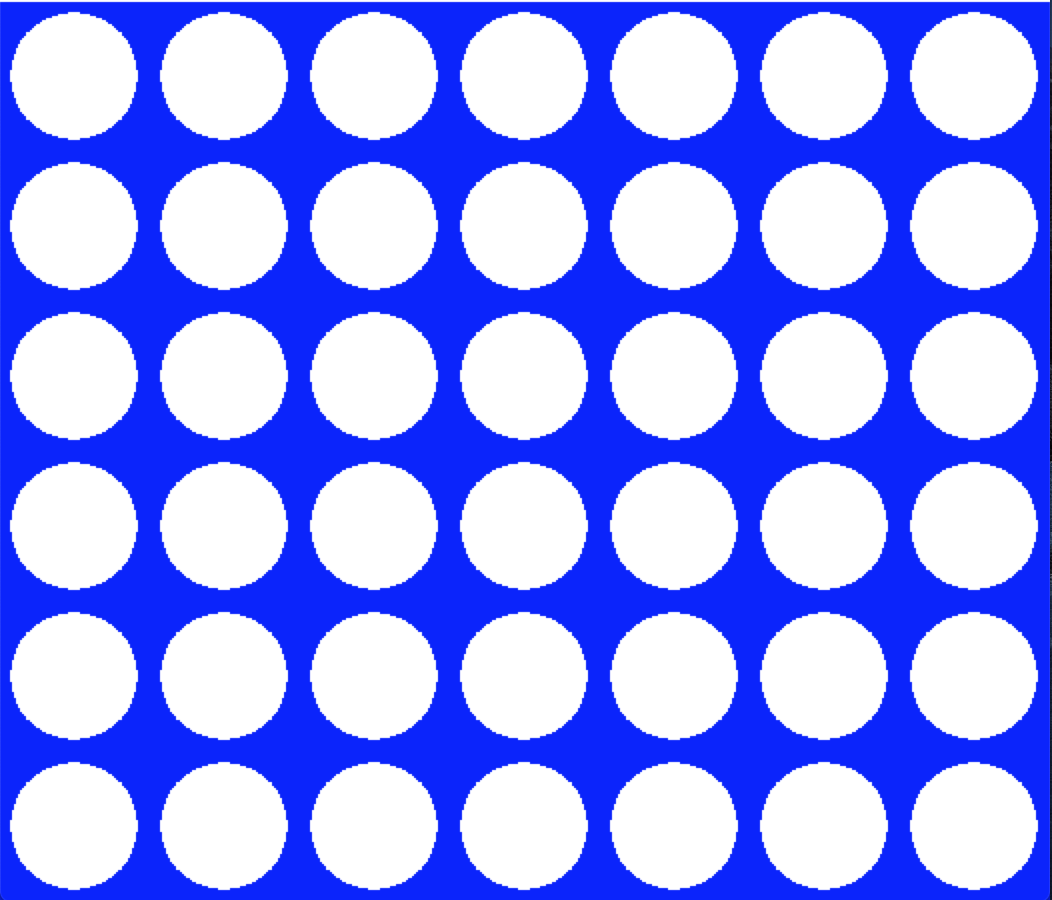
[0, 0, 0, 0, 0, 0, 0]

[0, 0, 0, 0, 0, 0, 0]

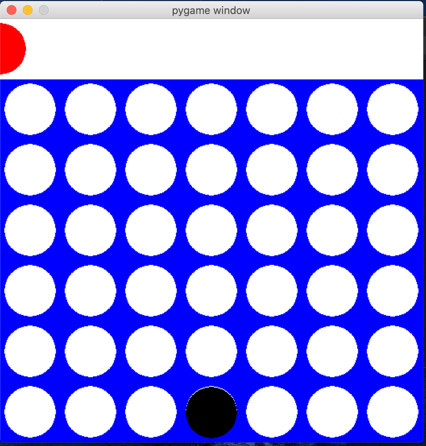
[0, 0, 0, 0, 0, 0, 0]

[0, 0, 0, 0, 0, 0, 0]]

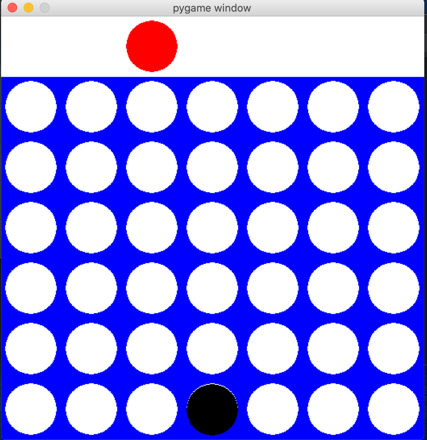
* + 1. *Player pieces* 
       1. A player piece was represented in the board by either a 1 or a 2. 2 being the computer agent and a 1 being the user piece
    2. *User Interface*
       1. The board is drawn to a separate pop up window as shown below where white space is a blank spot on the board.



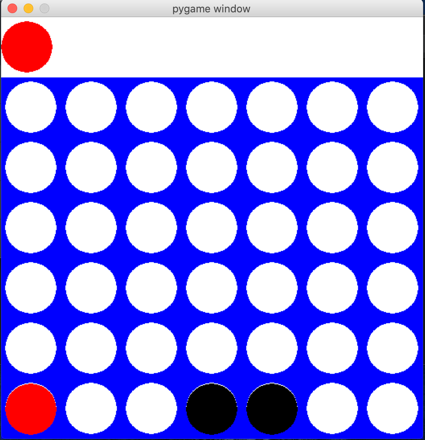
* + - 1. A piece is dropped into the board either by the agent(black) or the player (red). The computer will drop its piece in based on the algorithm discussed above by choosing the path with the highest score. For a player turn the red piece will follow the mouse until it is clicked over a valid drop location shown in figure 2 and when the mouse is clicked it will drop into the board in the column which it is above shown in figure 3. This will continue until either the program is exited, one of the players wins, or a draw occurs at which time it will say who wins and then automatically close the popup and exit the program.
         1. **Figure 1**



* + - * 1. **Figure 2**



* + - * 1. **Figure 3**



1. **Conclusion**
   1. In conclusion with the computer player and a minimax with alpha beta pruning applied to the computer’s next moves choice. It becomes extremely difficult to beat the computer especially with a depth of 6 applied to the algorithm. While this may not be a great player experience because of the difficulty, it goes to show just how powerful this AI can be for the game of Connect Four in a deterministic and static environment. This game could be modified to become less aggressive by changing the scores the algorithm uses or the depth could be decremented in order for the move space to be lessened for the computer in order to make it a more fair game and enhance user satisfaction.