The University of Texas at Dallas
CS 6364
Artificial Intelligence
Fall 2012

Class Project PROPOSAL

4 – in a row

Students: Siddartha Guthikonda, Chakravarthy Reddy Sirigireddy, Raghavendar Erupaka

Email Contact:

cxs126930@utdallas.edu
sxg122630@utdallas.edu
rxe110230@utdallas.edu

Contents

Table of Contents

1. Introduction

1.1 Game:

Implementing a game using Artificial Intelligence technique is our main task. In order to fulfill this we have developed a game called 4 in a row using Minimax algorithm and also compared it with other game playing algorithms.

1.2 Rules of the game:

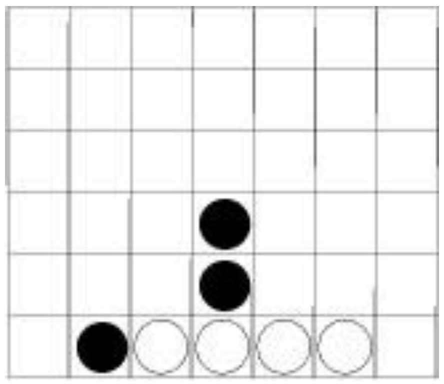
This is a two player game, each player having 21 identical pieces. Each player chooses his color of interest for his pieces or it can be pre done by the system. The game is played on a vertical rectangular board with 7 columns and 6 rows each. A piece put in one of the column will fall down and occupy the lowest unoccupied position in that column. A column can accommodate no more than 6 pieces. There is no possibility to place a piece at any random position other than the lowest possible position in a column.

The players make their moves in turn. No rule to specify a starting player. For our project, Player 1 player first followed by Player 2. Player 1 and Player 2 can be chosen by human.

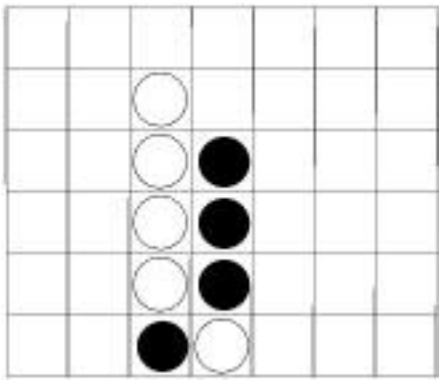
1.3 Goal:

Goal of player is to connect four of his pieces in a row vertically, horizontally or diagonally. The first player to achieve this goal wins the game. If all the 42 pieces are played and the goal is not achieved then the game is draw. It should also be noted that illegal moves are not allowed and therefore are not penalized. A win gets a value of +100 and lose gets a value of -100.

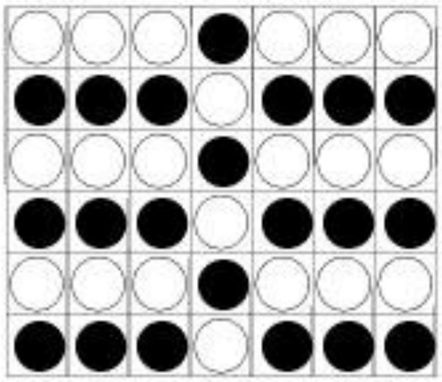
1.4 Block Diagrams:



Win State: Four white pieces in a row horizontally.



Win State: Four white pieces in a row vertically.



Draw State: Board is filled and all 42 pieces are placed.

2. Task Environment

The goal of this project is to create a agent that plays 4 in a row game. In order to design an appropriate agent, we made the agent as four different agents whose main task is to play the game. The four agents are given as:

- Performance Measure agent
- Environment
- Actuators
- Sensors

2.1 Performance Measure agent:

The aim of this agent is to measure performance of the whole agent and choose decisions in order to put four of its pieces adjacent to each other and prevent opponent from doing the same.

2.2 Environment:

Environment agent observes the environment of the game, i.e, the Game Board, Agent's pieces and Opponent's pieces. It is a fully observable agent which always has access to the game environment. Though it cannot use the Opponent's pieces, it can observe them and know how many more are left.

2.3 Actuators:

Actuator is a function which takes data from the performance agent and places the piece in a position in order to satisfy the performance measure agent.

2.4 Sensors:

Sensors are the agents which have direct access to the state of the board.

3. Development

3.1 Interface:

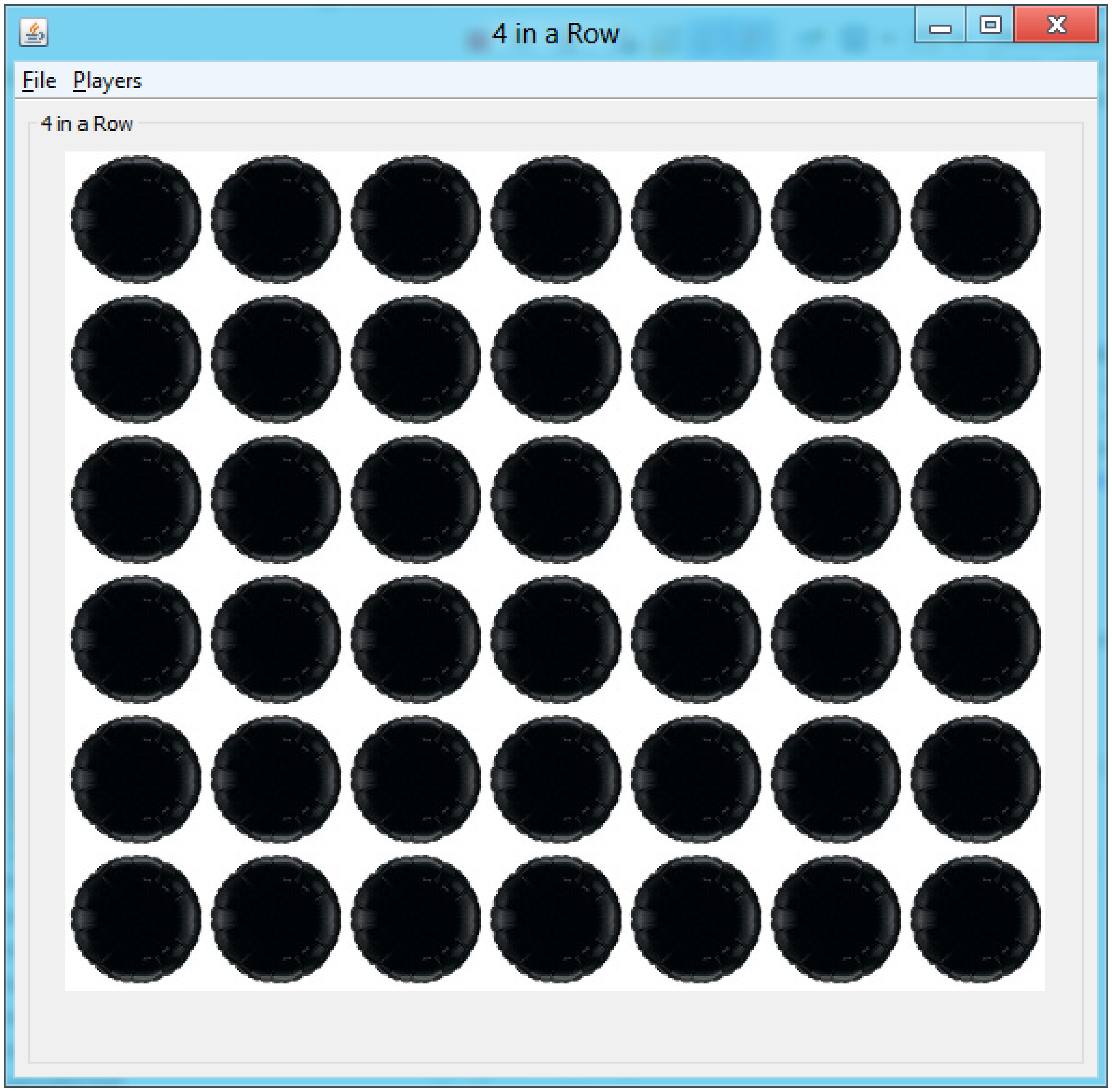
The interface of the game consists of the game menu and the game board. Menu gives options to start a new game and exit a game. Also the menu gives option to choose players as player 1 and player 2. Player 1 has always the first chance to move his piece.

Once the players are set, the user can begin a new game. The game board is set to initial configuration of all empty squares. Player can drop pieces by clicking on the board on the desired column. The pieces by default occupy the lowest unoccupied position.

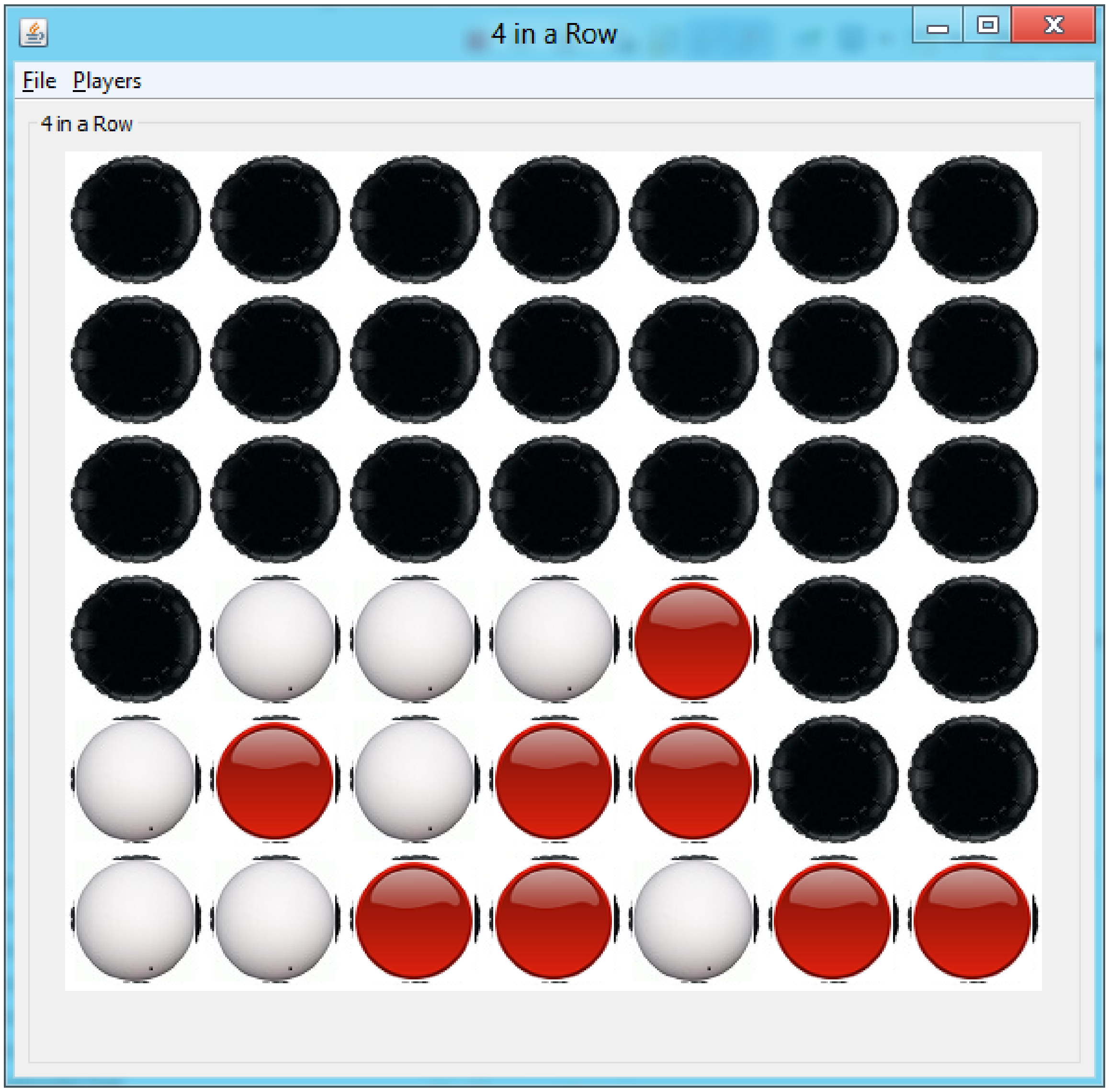
When the game is over, a window pops up to notify the user of the outcome. If the game results in a win for some player, the message states that the player has won. If the game is a draw, the message states the same.

3.2 Screenshots

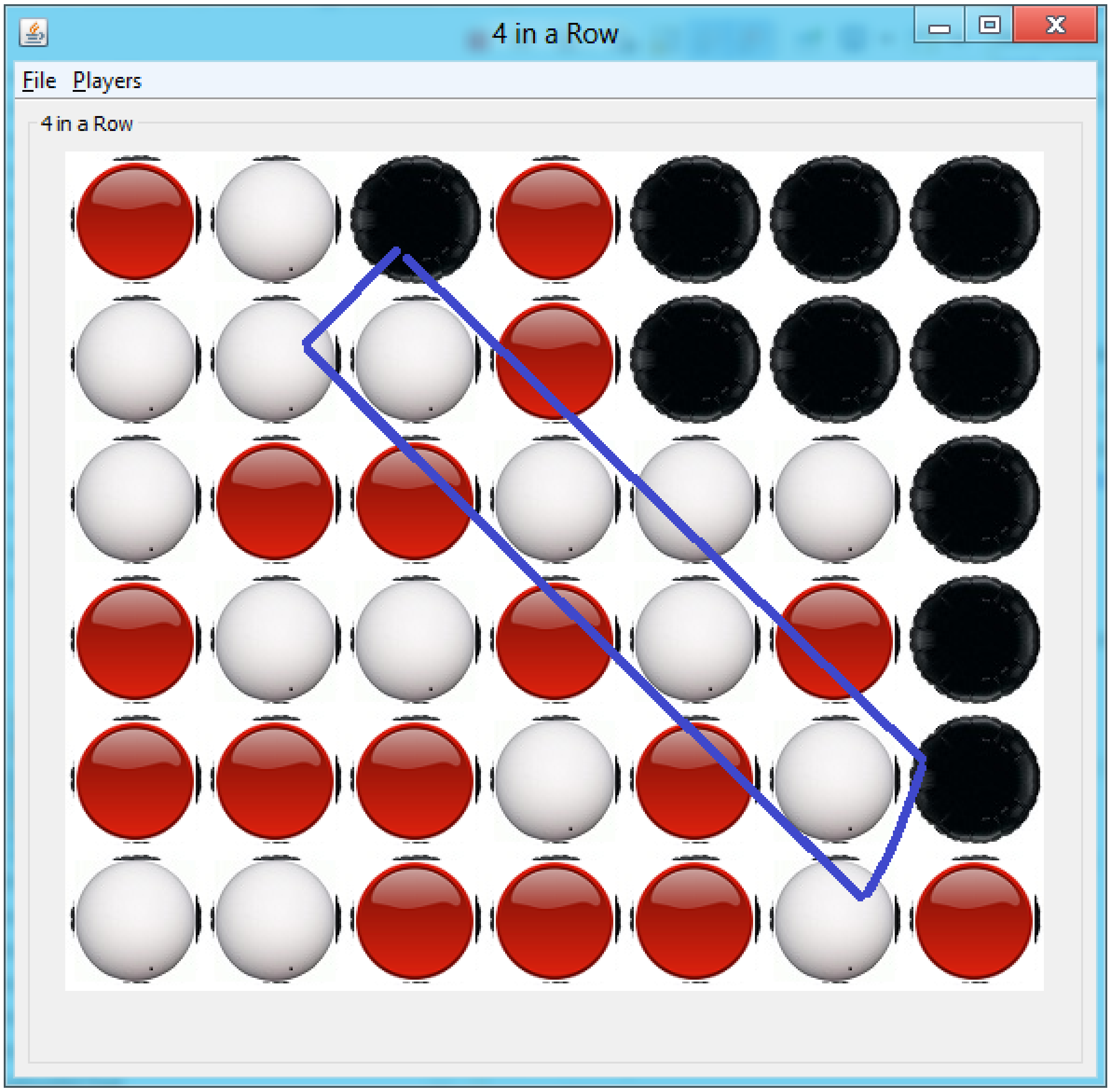
Initial state when no player made a move



State showing two players playing Four in a Row



Final State – Winning Position



4. Artificial Intelligence

The game is developed in such a way that it has more than one intelligent agents and different algorithms are used to compare results. An AI vs AI game can also be played but it finishes within seconds because of the speed of the agent.

4.1 Defensive AI:

This AI uses a heuristic function to determine what the next move should be. For every instance of the board, it assigns a value to each of available move. For example, if the opponent has three pieces in a row, a value of 8 is given to the space that would complete the opponent's goal. The lower the number of opponent's pieces the lower the values assigned would be. The aim of this algorithm is to block the opponent from achieving the goal.

4.2 Aggressive AI:

This AI uses the same type of heuristic as the Defensive AI with a difference. The defensive algorithm applies heuristic function to opponent's pieces but the Aggressive algorithm also applies the heuristic to its pieces also.

4.3 Minimax AI:

This algorithm constructs a minimax tree and instead of searching the whole tree, it searches upto four levels and makes the move. This limitation is because the computational constraints as the number possible trees are massive. The decision to limit look ahead to four levels is that it is the minimum number of moves a skilled player takes to complete the game. This algorithm yields better results than Defensive and Aggressive player.

4.4 Utility Function for Minimax:

In order to choose a path in the Minimax tree, a utility function is chosen to give values for the terminal node. The calculation of utility function is given as follows:

- Win: Returns a utility value of +100.
- Lose: Returns a utility value of -100.
- Draw: Returns a utility value of 0.
- 3 pieces of AI in a row: Returns a value of Heuristic + 16.
- 3 pieces of Opponent in a row: Returns a value of Heuristic + 8.
- 2 pieces of AI in a row: Returns a value of Heuristic + 4.
- 2 pieces of Opponent in a row: Returns a value of Heuristic +4.

Depending on this Heuristic values the Intelligent Agent to play our game calls the Actuator to place a piece in the right position. These values are calculated and maintained by the Performance Measure agent.

5. Results

We compared the results of our three players, i.e Aggressive agent, Minimax agent and Defensive agent. The observation from our project is that Minimax is one such algorithm that never loses a game while playing with other AI players. This is because unlike Aggressive and Defensive players, our Minimax player looks ahead four steps in advance to make a move. It is also observed that Alpha beta pruning significantly reduces the number of nodes.

References:

[1] Java 2D graphics: http://docs.oracle.com/javase/tutorial/2d/index.html

[2] Milton-Bradley, Inc. (). Connect Four Game, *Hasbro, Inc.*

[3] Russell, S., Norvig, P. (2003). Artificial Intelligence, A Modern Approach, Third Edition. *Pearson Education, Inc., Prentice Hall*.