

Project

Connect Four

Guidelines

While we acknowledge that beauty is in the eye of the beholder, you should nonetheless strive for elegance in your code. Not every program which runs deserves full credit. Make sure to state invariants in comments which are sometimes implicit in the informal presentation of a class. Do not reinvent wheels, and try to make your functions small and easy to understand. Use tasteful layout and avoid long winded and contorted code. You are expected to work alone, but of course, you can discuss problems, solutions and strategies with others.

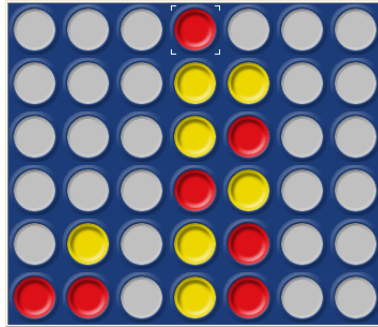
Goal

The goal of this project is to implement the game Connect Four. The project is designed as a four week project. Every week you are asked to hand in a homework assignment, which will be graded. So all in all, your project will be worth four homework assignments. In the last class, we will have a Connect Four competition, where your programs will play against one another. The winner will receive a price.

Game Description

Connect Four is a two-player board game in which the players take turns in dropping tokens into a vertical grid with the objective of getting four of one's own tokens into a line. The game was published by Milton Bradley in 1974; a non-proprietary version is known as "The Captain's Mistress". See http://en.wikipedia.org/wiki/Connect_Four for more information.

In the game, there are two colors of tokens. One player plays yellow, the other red. The following depicts a game configurations that may occurred in a concrete game.



If we enumerate the columns with 1..7, we see that if yellow plays 3 and red plays 3, red will have won, because red has managed to get four pieces into a line. This one is diagonal, but it could also be horizontal or vertical.

Rules

1. The game board, has seven columns and six rows.
2. There are 21 red and 21 yellow tokens.
3. One player plays with red tokens, the other with yellow tokens.
4. The tokens are inserted at the top of a column, and they will fall down and land on the ground (if the column was empty) or on top of a previously inserted token.
5. Red starts.
6. Red and yellow take turns.
7. One can only insert tokens in one of the seven columns.
8. One cannot insert a token into a column that is full.
9. A line consists of several tokens, either in vertical, horizontal, or diagonal form, which contain only tokens of the same color.
10. A player wins if he or she manages to form a line of four tokens of his or her color.
11. The game ends if one of the player wins.
12. The game ends in a tie if none of the two players can make a valid move and none of the players has won.
13. A move cannot take more than 180 seconds.

Project Goals

You are asked to implement two players for the Connect Four game. A human player and an automatic player. We will provide you with one interface of what a player is expected to do, you are supposed to implement it. No more rules are given.

To implement the human player, you will need to design an implement a human computer user interface. This can be a graphical user interface or a text base interfaces, your choice. One of the critical steps in this part of the assignment is to think of a good representation of the board. For the second assignment you need to hand in a program that allows two human players to play each other.

The automatic player is like a human player except that it uses an algorithm to determine where to put one the next token. This could be a probabilistic algorithm that randomly tosses tokens into some not yet full column into the board. Or it could be a search algorithm, just like the ones we discussed in class, depth first search, breadth first search, iterative deepening, or maybe even $\alpha\beta$ -pruning search. (This will likely be my choice for my player!)

Interfaces

Before telling you what you have to do, we establish some common basic rules for the implementation. First, the color red and yellow are represented by Boolean values, where

red is represented as **true**
yellow is represented as **false** .

Every player you write, human or automatic, should implement the **Player** interface. Your automatic player needs to have an internal representation of the board, otherwise it will not be able to efficiently search for the best move. The human player doesn't do it, because we have memory, and are used to have the game situation in our heads. But this also means, that the automatic player needs to be informed about the move of the opponent. The human player observes that directly. Therefore, the **Player** interface that we ask you to implement the following four methods.

```
interface Player {  
    void init (Boolean color);  
    String name ();  
    int move ();  
    void inform (int i);  
}
```

init is the method that the Game referee (implemented and administered by the teaching staff) will use in order to inform a player about what color he/she/it is playing. The **name** method serves a player to identify himself or herself for the logs. The **move** method is called when a player is asked to make a move, and the **inform** method is called to inform the player about the move of the opponent.

Assignment 10

(due, Monday 27.11.2006)

Hand in your design of a board, with all methods that you think are necessary and important.

Assignment 11

(due, Monday 4.12.2006)

Implement a human player. Hand in your code in such a way that when started, two human players play against each other. Use a graphical user interfaces if you like!

Assignment 12

(due, Monday 11.12.2006)

Implement an automatic player. Hand in your code in such a way that by default your human player plays the automatic player.

Assignment 13

(due, Monday 18.12.2006)

Have your code ready, in time for the first semi annual OOP championship. Simply hand in your automated player in one file. Stay tuned for more information from the TAs on how to package your code in such a way that they can run the competition smoothly. (If you don't follow these instructions, you will be automatically disqualified.)