Connect4

Written for : BadGames, Inc
Written by : Marc Chevaldonné, IUT Informatique, Université d'Auvergne Clermont1

November the 25th of 2008
Functional requirements version 1.0

Index

1. Introduction

1.1. Context

This document has been written for students of the second year of “Génie Informatique” of the IUT Informatique in the University d’Auvergne. Its goal is to be used as a simulation of a real case for a project. In this context, these requirements describe the software the students will developp.

Connect4 is a simple player vs. player strategy and combinatory game, in which the players take turns in dropping alternating colored discs into a seven-column, six-row vertically-suspended grid. The object of the game is to connect four singly-colored discs in a row -- vertically, horizontally, or diagonally -- before your opponent can do likewise.

1.2. Glossary

Connect4 is also known as Puissance4 in french.

1.3. Referenced documents

This document is refering to the following documents:

Title	Version
Connect4 - Book of specifications	1.1

2. General Requirements

2.1. Software presentation

The product to develop is made of one software application.

Its goal is to allow playing the Connect4 game in “player versus player” mode :

- human player versus human player
- human player versus computer player
- computer player versus computer player.

The computer player is made of an Artificial Intelligence (AI).

2.2. Operational constraints

2.2.1. Exploitation

The software shall allow loading different Articifial Intelligence for computer players. It includes adding new Artificial Intelligence to the application, without the help of the developer (the program can load new AI without new building).

2.2.2. Performances

The software should run on any PC computer with at least Windows 2000.

2.2.3. Installation

The application will be delivered as an exe file and a pack of dlls.

2.3. Realisation constraints

2.3.1. Structural constraints

The product is made of an exe file, some dlls (particularly for Artificial Intelligence) and XML files for saving high scores.

2.3.2. Evolution constraints

This software should be evolutive: users should be able to add new Artificial Intelligence to the application by adding new library dlls in a particular folder of the application.

2.3.3. Development constraints

The software application will be developed on PC with Windows 2000 or higher and Visual .Net 2005, in C# and XML (for high scores files).

2.3.4. Quality constraints

To be defined.

3. Glossary

term	definition
Connect4	a two-player game in which the players take turns in dropping alternating colored discs into a seven-column, six-rox vertically-suspended grid. The object of the game is to connect four singly-colored discs in a row (vertically, horizontally, or diagonally) before your opponent can do likewise. It is called Puissance4 in french.
Board	the vertical grid in which the colored discs are dropped
Chip	Colored disc
Artificial Intelligence	the intelligence of the machines and the branch of computer science which aims to create it. In this document, we use these two words to point an algorithm calculating the best column to drop a disc in.

4. Functional requirements

This part describes the different requirements of the application. The main function F0 is “to play to Connect4”. It is composed of the following principal functions :

- F1 : to define a player
- F2 : to choose temporary characteristics to link to the players during the game
- F3 : to play the game
- F4 : to manage scores
- F5 : to load Artificial Intelligence algorithms

These functions and their sub-functions as well, are described in details below.

Function F1 : to define a player

Description

Allows defining a player (old or new), i.e. its name, type (human or AI), chip color, ... By default, the two players are described with name “anonymous” and type “human”.

Input(s)

None or player’s XML file (optional).

Processing

A player is created (new player) or modified (old player) with the help of the sub-functions described below.

Ouput(s)

A new player recorded in the player’s XML file or an existing player modified or not, ready to play.

Fonction F1.1 : to load an existing player

Description

Allows choosing one existing player in the list of players that have already played a game on this computer.

Input(s)

None.

Processing

The user can choose among a list of existing player one of them, and the characteristics of this player are displayed.

Output(s)

An existing player is selected and ready to play.

Fonction F1.2 : to modify the player

Description

In the beggining, the players are default players (called “anonymous” and of type “human”). The players can then be modified. If an existing player has been loaded, it can also be modified. Then, the following sub-functions apply to both cases.

Input(s)

A player (default player or already loaded or modified player).

Processing

The user is able to modify characteristics of a player.

Output(s)

- same player, but modified

Fonction F1.2.1 : to modify the player’s type

Description

Allows modifying the type of the player. The type can be one of the following ones : “Human”, “AI” (for Artificial Intelligence) or “Random”.

Input(s)

- A player.
- List of possible types : Human, AI, Random.

Processing

This action modifies the type of the player. However, it can have consequences on another charateristic of the player: its name. If the type is modified :

- from “Human” or “Random” to “AI”: the name is the one corresponding to the default AI algorithm (first one found by the application, see function ?).
- from “AI” or “Human” to “Random”: the name is set to “No Name”
- from “Random” or “AI” to “Human” : the name is set to “Anonymous”

Output(s)

- the player with the type and name modified.

Fonction F1.2.2 : to modify the name of the player

Description

Allows to change the player’s name, only if the player is of type “Human”.

Input(s)

- A “Human” player.

Processing

If the player to modify is of type “Human”, the user can modify its name using one of the two sub-functions described below.

Output(s)

- Modified “Human” player.

Fonction F1.2.2.1 : to choose the name of the player among a list of existing players

Description

Allows choosing the name of the player if it is of type “Human”, among a list of existing players.

Input(s)

- A player
- Player’s XML file

Processing

This function does exactly the same processing than the function F1.1 but works like a filter among players. Instead of choosing a player of any type in the list of known players (in the player’s XML file), it allows you choosing the player only among the players of type “Human” in this file.

Output(s)
The player with a modified name.

Fonction F1.2.2.2 : to choose a new name for a player

Description
Alllows the user to choose a new name (not existing in the player's XML file) for the player to modify.

Input(s)
- A player
- Player's XML file

Processing
The user inserts a new name for the player to modify and the application checks if this name is not owned by another registered player. If not, the name is modified; else, the player's name remains unchanged.

Output(s)
The player with a modified name (or not).

Fonction F2: to choose temporarly characteristics to link to the players during the game

Description
A game needs to link some characteristics to the players temporarly like the color of the player's chips and the name of the player that will begin the game. They are needed for starting a game.

Input(s)
2 players.
2 colors.

Processing
The user links a color to a player. The two colors must be different enough to be distinguished by the players. The user chooses one of the two players that will begin the game.

Output(s)
2 players, each one linked to a particular color.
The player that will begin the game.

Fonction F2.1: to link a player to a color

Description
Allows linking a color to each player. If this is the first game since the running of the application, default colors are red for first player, and yellow for last one. If not, the colors are the same than the last game.

Input(s)
2 players.
2 colors.

Processing

The user links a color to a player. The two colors must be different enough to be distinguished by the players.

Output(s)

- 2 players, each one linked to a particular color.

Fonction F2.2: to choose which player will begin the game

Description

Allows choosing which player begins the game.

Input(s)

2 players.

Processing

The user chooses one of the two players that will begin the game.

Output(s)

- the player that will begin the game.

Fonction F3: to play the game

Description

Groups the sub-functions needed for playing.

Input(s)

- A game
- Two players
- A board

Processing

Allows users to insert chips in the board, to go back one turn, to display if the game is even, of if it is won, the name of the winner.

Output(s)

The winner or the message saying that the game is even.

Fonction F3.1: to start a game

Description

Allows starting a game between two players.

Input(s)

- 2 valid players.

Processing

If the two players are valid (i.e. of any type except “Human” or of type “Human” but with a name different than “Anonymous”), the game is starting : the first player is asked to insert a chip (see function 3.2).

Output(s)
an empty board.

Fonction F3.2: to choose a column where to insert a chip

Description
Allows a player to choose one of the board's column.

Input(s)
- a board
- a player.

Processing
The player chooses a board's column to insert a chip in, but the insertion is not done yet (see function 3.3 for insertion), it is only a choice..

Output(s)
a column id.

Fonction F3.3: to insert a chip in a column

Description
Allows a player to insert one of its chip in the chosen column.

Input(s)
- a board
- a player
- the name of the next player
- a column id.

Processing
The player chooses a board's column (see function 3.2) to insert a chip in. If the column number is valid (not too low, not to high, not corresponding to a full column), a chip is inserted and the next player is the other one. Else, no chip is inserted, and the next player is still the same. Then the "next player" is asked to insert a chip in the board (still functions 3.2 and 3.3).

Output(s)
modified board
the name of the next player.

Fonction F3.4: to go back one turn

Description
It should be possible to go back one turn (multiple times) in order to avoid unvoluntary chip insertion.

Input(s)
- the board (not empty)

- the next player.

Processing

If the board is not empty, this function allows going back one turn and changes “next player” to the other one.

Output(s)

the board with one chip less.

the next player

Fonction F3.5: to check if there is a winner

Description

Checks if one player has made an n-chips line.

Input(s)

- board.

Processing(s)

If one player has an n-chips line, outputs his name (see function F3.7)

Ouput(s)

optional: winner’s name.

Fonction F3.6: to check if the board is full

Description

Checks if the board is full.

Input(s)

- board.

Processing(s)

Checks if all columns are full. If there are full, checks if there is a winner (see function 3.5).

Ouput(s)

None.

Fonction F3.7: to display the winner’s name or that the game is even

Description

When the game is over with a winner, displays the winner’s name. When the game is over without any winner (board full and no chip connection), displays that the game is even.

Input(s)

- board
- optional: winner’s name.

Processing(s)

If there is a winner, displays his name. If there is no winner and the board is full, displays that the game is even.

Ouput(s)
Message with the winner's name or that the game is even.

Fonction F4: to manage scores between players

Description
Groups the sub-functions needed for managing scores between players.

Input(s)
- Two players
- player's XML file.

Processing
Stores the scores between two players in the player's XML file.

Output(s)
score
player's XML file.

Fonction F4.1: to load scores between two players

Description
Allows starting getting the scores between two players (mainly in order to display it, see function F4.2).

Input(s)
- 2 existing different players.

Processing
If the two players are existing and different, checks if they have already played one against the other, and if yes, loads their score (else, loads 0-0).

Output(s)
score.

Fonction F4.2: to display the score between two players

Description
Displays the score between two players.

Input(s)
- score.

Processing
Everytime a player is modified, the application loads the score between the two current players (see function F4.1) and displays it.

Output(s)
None.

Fonction F4.3: to update the score between two players.

Description

At the end of the game, the winner's score between its opponent is incremented.

Input(s)

- 2 players
- the winner's name
- XML player's file.

Processing

If the players have already played together, the application looks for their score in the player's XML file and increments the score of the winner. Else, it creates a new score between them in the file with 1 for the winner and 0 for the looser.

Output(s)

modified XML player's file.

Fonction F5: to load Artificial Intelligence algorithms

Description

Loads AI algorithms at the launching of the application.

The Artificial Algorithm are contained in Dynamic Linked Library (dll). These DLL should be copied in the main executable folder. Then, when the application is launched, it should parse the base folder, import the DLL and loads the different AI algorithms. One DLL can contain different algorithm. A unique name is then created for each algorithm with the help of the name of the AI player, the name of the creator, and the version number.

Input(s)

- .exe file of the game
- .dll files containing AI algorithms.

Processing

At the launching of the application, the program parse the base folder and import the DLL containing the AI algorithms. The algorithms are now usable through "AI" players (see F1.1 to load one). Their name is made of the name given by the creator, the creator's name and the version number.

Output(s)

a set of AI algorithms usable as "AI" players.

5. Summary of the functional requirements

	Functional requirements
F1	to define a player
F1 1	to load an existing player
F1 2	to modify the player
F1 2 1	to modify the player's type
F1 2 2	to modify the name of the player
F1 2 2 1	to choose the name of the player among a list of existing players
F1 2 2 2	to choose a new name for a player
F2	to choose temporarly characteristics to link to the players during the game
F2 1	to link a player to a color
F2 2	to choose which player will begin the game
F3	to play the game
F3 1	to start the game
F3 2	to choose a column where to insert a chip
F3 3	to insert a chip in a column
F3 4	to go back one turn
F3 5	to check if there is a winner
F3 6	to check if the board is full
F3 7	to display the winner's name or that the game is even
F4	to manage scores between players
F4 1	to load scores between players
F4 2	to display scores between players
F4 3	to update scores between players
F5	to load Artificial Intelligence algorithms

6. Technical requirements

6.1. Player's XML file

On each computer running the Connect4 game, some information must be saved about the players and the previous games. These information should be saved in an XML file and should be made of:

- information about the players:
  - type: "Human", "AI" or "Random"
  - name (name of the algorithm or the library in the case of "AI" type, name of the player in the case of "Human" type, "No name" in the case of "Random" type), must be unique in the document
- information about the previous games:
  - the name of the opponents
  - the score: number of wins for each opponent against the other one

6.2. Loading of the different AI algorithms

The Artificial Algorithm are contained in Dynamic Linked Library (dll). These DLL should be copied in the main executable folder. Then, when the application is launched, it should parse the base folder, import the DLL and loads the different AI algorithms. One DLL can contain different algorithm. A unique name is then created for each algorithm with the help of the name of the AI player, the name of the creator, and the version number.

7. Interface requirements

This part describes the requirements for the application interface.

7.1. Human/Software graphical interface

The draft of this part presents the main parts that the graphical interface of the application should contain. However, the appearence is not fixed and can naturally be modified.

The following of this part presents the different parts of the interface.

7.1.1. Players definition graphical interface

This first part of the software should be the one the users see when running the progam (see figure below). It presents a three columns tab: the first one presents the characteristics of the players (name, type, score, chip color and the indication of the player beginning the game), and the two other ones the two players.

The name and type of the players can be modified with the help of ComboBox-like components. The colors can be modified with the help of ColorDialog-like components. The indication of the player beggining the game can be done with Checkbox-like components, one disactivating the other when it is clicked.

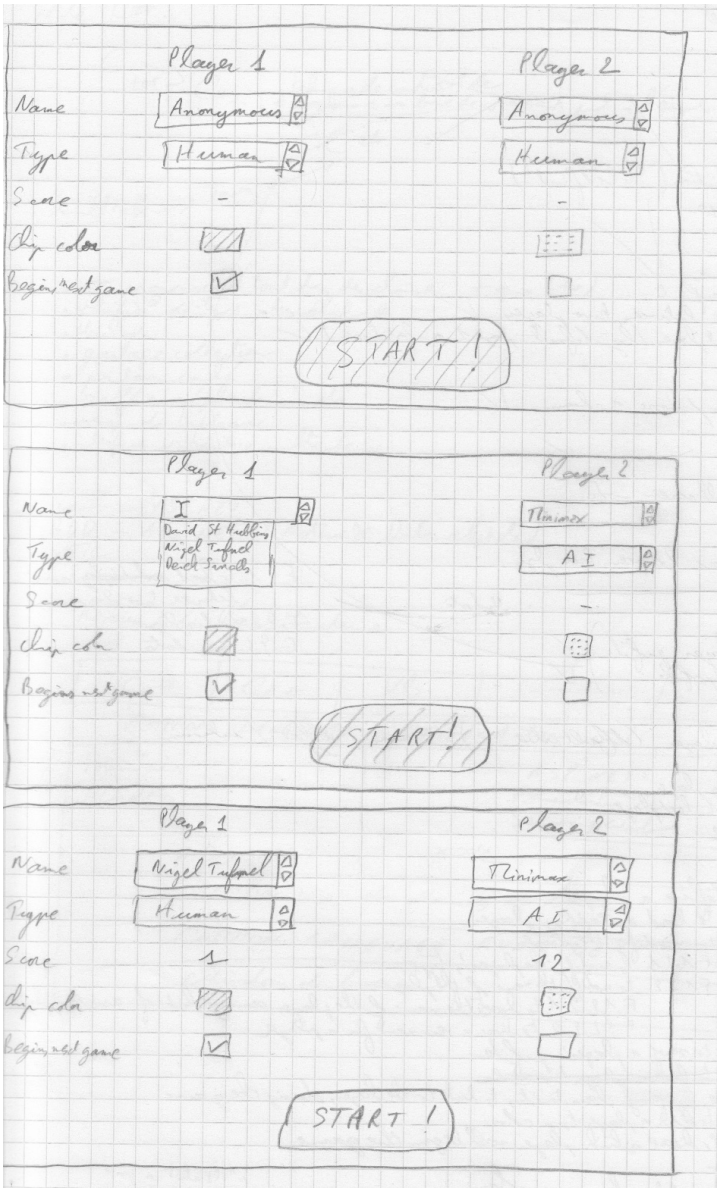
The score is automatically updated when the program recognizes a pair of opponents.

At last, a big "START !" is activated if two valid players are created; else it is not activated.

The functions associated to this part are the following ones (and their sub-functions):

- F1 to define a player

- F2 to choose temporarly characteristics to link to the players during the game
- F31 to start the game
- F41 to load scores between players
- F42 to display scores between players



7.1.2. Game graphical interface

On the top of the interface, the players names, types, chip colors and scores are displayed. An arrow or any system is indicating which player is about to play.

The main part of the screen represents the board with empty cases or cases with colored chips. On the top of the board a flying chip represents the current column selected by the next player before insertion. This could be done only by dragging the mouse cursor on th e columns. When the selection is validated

(by mouse clicking for instance), the flying chip falls to the lowest empty case of the selected column (only if the column is not full).

If one of the players has won, a big message (with colors and some graphical effects) indicates his name and the score is updated (and saved in the player's XML file). If the game is even, a more simple message indicates "no winner" or something like that.

After some time or when a user clicks on the mouse, it goes back to the "defintion player screen".

The functions associated to this part are the following ones (and their sub-functions):

- F32 to choose a column where to insert a chip
- F33 to insert a chip in a column
- F34 to go back one turn
- F35 to check if there is a winner
- F36 to check if the board is full
- F37 to display the winner's name or that the game is even
- F42 to display scores between players
- F43 to update scores between players