"LIVIU REBREANU" NATIONAL COLLEGE BISTRIŢA



CONNECT FOUR

*Certificate work, year 2020*

Author: Alexandra Ilovan

Coordinating teacher: Constantin Gălățan

Introduction

The Connect Four application consists of an interactive game for two, the goal of which is to connect 4 disks of the same color in a horizontal, vertical or diagonal line in front of the opponent. This is an accessible game for all ages that is not very difficult, but which nevertheless puts the mind to the test. Connect Four, often compared to "X and O" or checkers, is suitable for those who feel the need to enter into a competition with friends, but in a relaxing setting.

The inspiration for the app came from the physical form of the game and borrows the same principles.

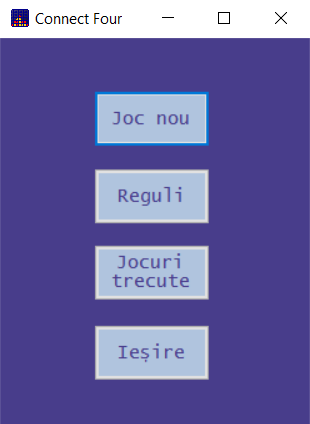
 

Presentation of the application

The Connect Four application is a C# application, of the Windows Forms App type, made with the Microsoft Visual Studio 2019 program and using the .NET 4.5.2 platform.

The interface is made entirely in Romanian and consists of 5 forms.

When launching the application, the main form (menu) will open, which is equipped with 4 buttons:



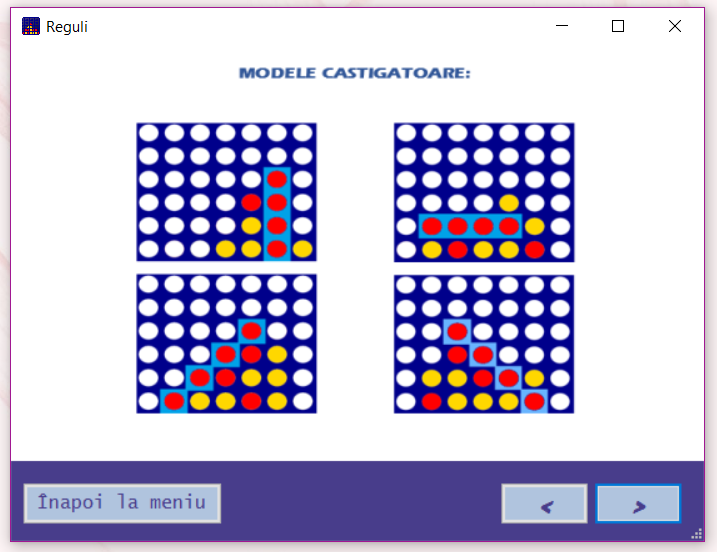
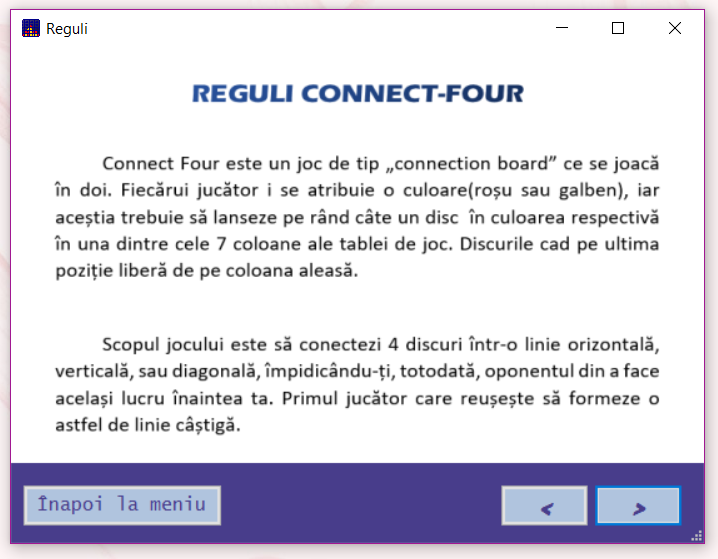
1. New Game – opens a new form for entering player names before starting the game

2. Rules – takes you to a form where the rules of the game are briefly presented

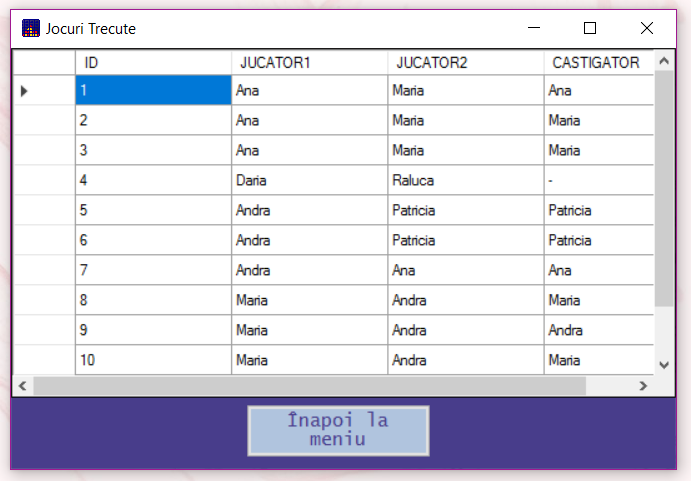
3. Past games – opens a form where all the games played so far are presented

4. Exit – exiting the application

Pressing the Rules button will open this form:



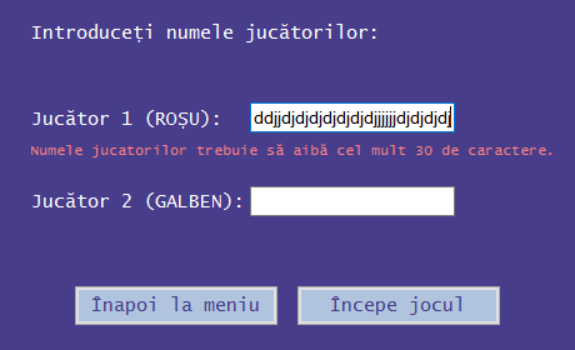
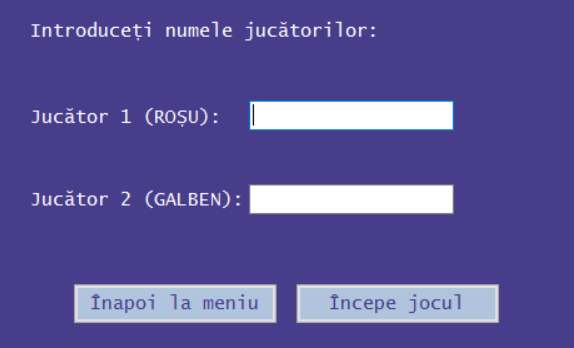
This form is made like a slideshow, offering the possibility to view the rules by using the ">" (forward) and "<" (back) buttons, or if the users have familiarized themselves with the rules of the game, they can return to menu by pressing the Back to Menu button. (The rules pages are images extracted from the Images folder found in bin -> Debug.)

Pressing the Past Games button will open the following window:

Within it, there is a dataGridView and a back menu button. The DataGridView displays information about each game, data that is stored in a SQL database, in the Games table after each game. The Games table has the following structure:

* ID: an individual index (which is also the primary key), representing the game number
* PLAYER\_NAME1
* PLAYER\_NAME2
* WINNER: Holds the name of the winning player

Pressing the New Game button will open a form with two textBoxes for entering the names of the two players. Both have a 30 character limit, to match the limit set in the database fields. If this limit is exceeded, a warning message will be displayed.

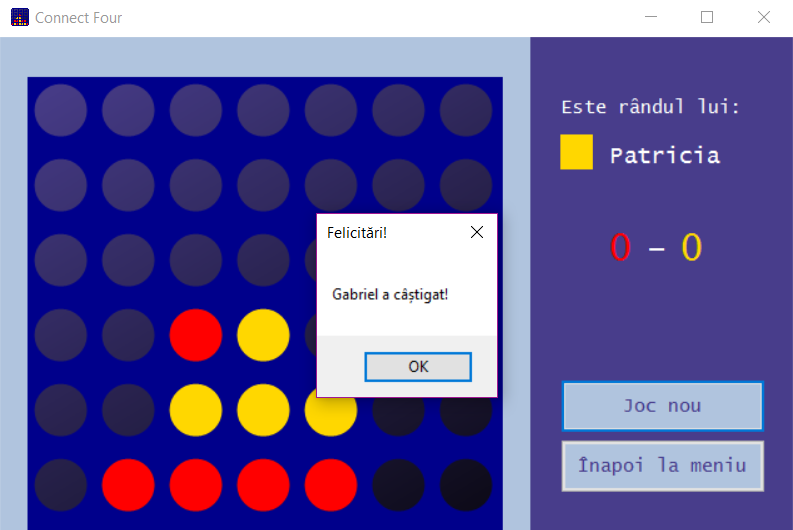


Also, starting the game will not be possible without filling in both text boxes.

The Start Game button will finally open the window dedicated to the game:



On the left side of the window we have the game board, and on the right side a label that specifies who is the next player (when opening the shape, player 1 will always start - associated with the color red, but when starting a new game, the previous winning player will start first ), a label specifying the score, a button to start a new game (which can only be used if the current game is finished) and a button to return to the menu. Either by pressing the New Game button or by pressing the "X" or Back to the menu buttons, the data specific to the last game (the names of the players and the winner) are saved in the database. After forming one of the 4 winning patterns, a MessageBox will appear to convey who the winner is, and after pressing the "OK" button, the score will update.





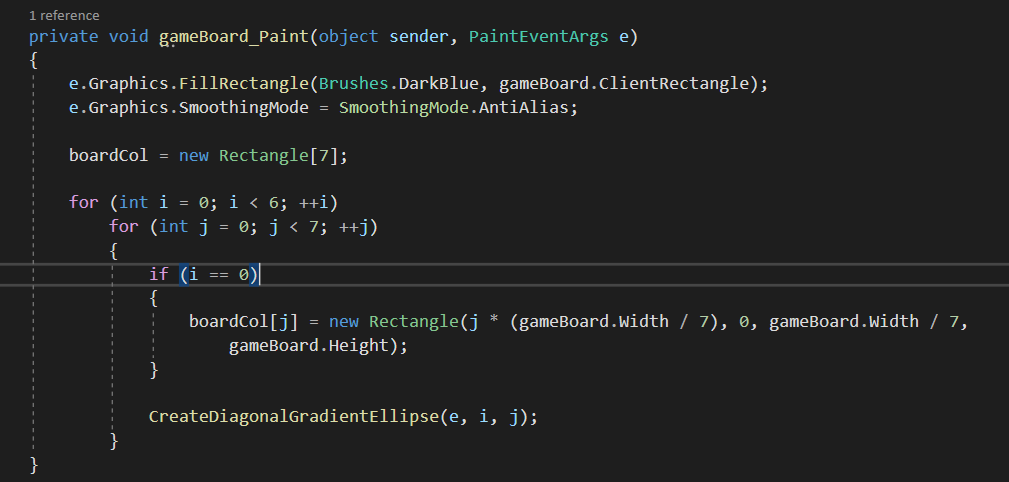
Description of the way the program works and the source code

Most of the code, both for the game mechanics and saving the necessary information to the database, took place in Form1 (the form in which the game is played), so I'll focus on explaining that.

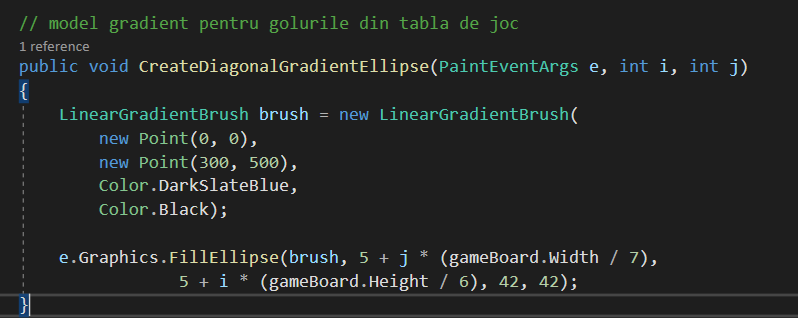
First of all, both the game board and the red and yellow disks were made by drawing with the help of Graphics objects g1(associated with the game board -> panel gameBoard), g2(associated with the panel used to indicate the next player row -> panelTurn). I saw the game board as a matrix with 6 rows and 7 columns.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 00 | 01 | 02 | 03 | 04 | 05 | 06 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 20 | 21 | 22 | 2. 3 | 24 | 25 | 26 |
| 30 | 31 | 32 | 33 | 34 | 35 | 36 |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 |
| 50 | 51 | 52 | 53 | 54 | 55 | 56 |

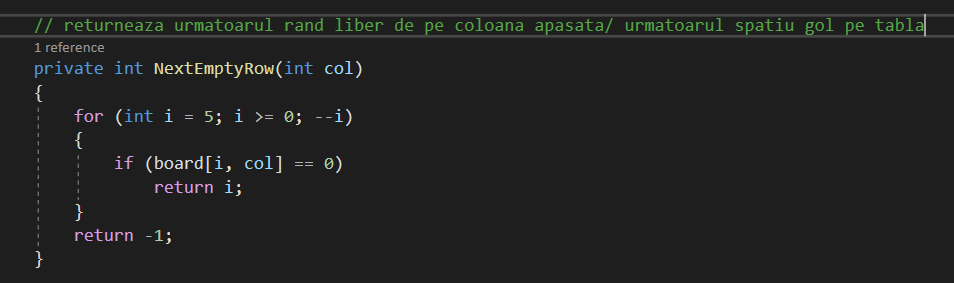
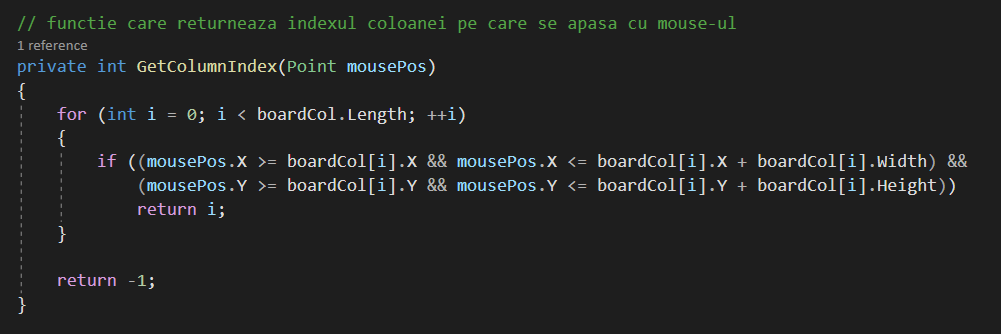
I created an array (boardCol) of Rectangle objects with 7 components (since the game board has 7 columns). We looped through the array as normal, and every time we encountered an i = 0, we created a new rectangle on column j – so we could easily identify which column we were on later. (As seen in the table above, for each i = 0 we have one rectangle from top to bottom)



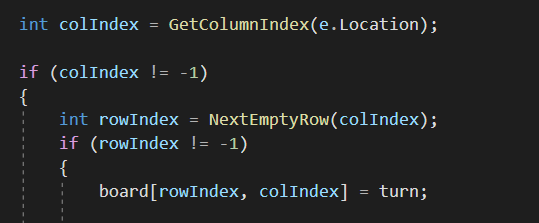
Also, within this for cycle, the function is called CreateDiagonalGradientEllipse(PaintEventArgs is,intand,intj) which deals with drawing the ellipses on the game board (the free spaces on which the disks can be launched).



In the physical format of the game, disks are launched from the top and fall to the lowest free position on the column they were launched on. Using the same reasoning, a function that determines the column the player clicks on with the mouse (the player will be able to click anywhere on the column they want to place the disk on, and the program will instantly find the first free position on it) and one that finds the most free row down (the next place the disk can land) on that column are essential.



In the MouseClick event above the gameBoard, it is checked after the placement of each disk if the last player who launched one won.



I created a board matrix with 6 lines and 7 columns, and if the clicked mouse position is not outside the game board, then board[i, j] = tower where:

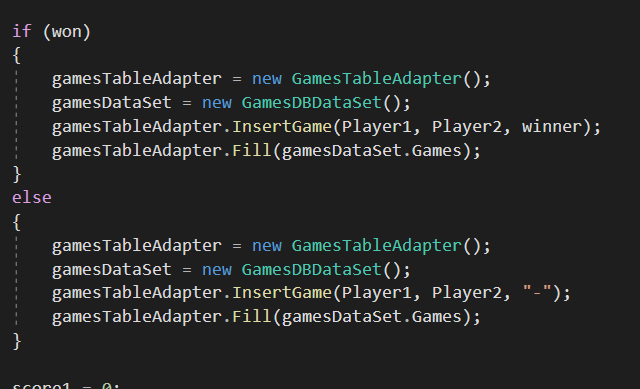
* i = index of the next free row, j = index of the pressed column
* turn (type int) – indicates the player's turn
  + if turn = 1 => it is the first player's turn
  + if turn = 2 => it is the second player's turn

However, when it comes to filling the board matrix, the meaning is slightly different:

board[i, j] = 1; (turn default = 1) => player 1 placed a disk on the board in the space at position (i, j)

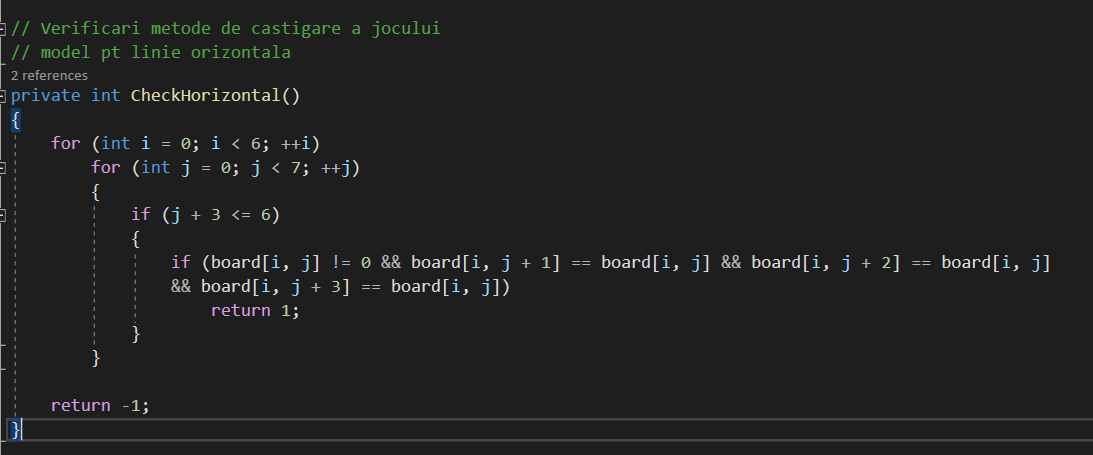
* at the start of the game all array elements are initialized to 0

Saving data to the database:



* if the game is not finished and the window is closed, instead of the winner's name the character string "-" will be saved.

Afterwards, I created 4 functions to determine the winning patterns, one function for each pattern (horizontal line, vertical line, diagonal in front, diagonal in back).

*For example:*

Going through the matrix board, I checked that there was no possibility of exiting the matrix, and if I found 4 equal and non-zero values ​​on consecutive positions in the matrix, it means that I found a winning model => in which case the function returns value 1.