VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

COMPUTER ARCHITECTURE (CO2007)

Assignment

FOUR IN A ROW

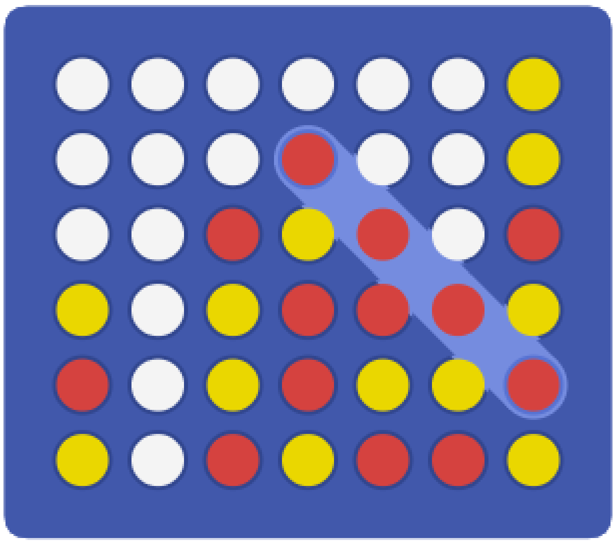
Advisor: Pham Quoc Cuong

Students: Nguyen Minh Hung - 2052504

HO CHI MINH CITY, NOVEMBER 2022

I. INTRODUCTION

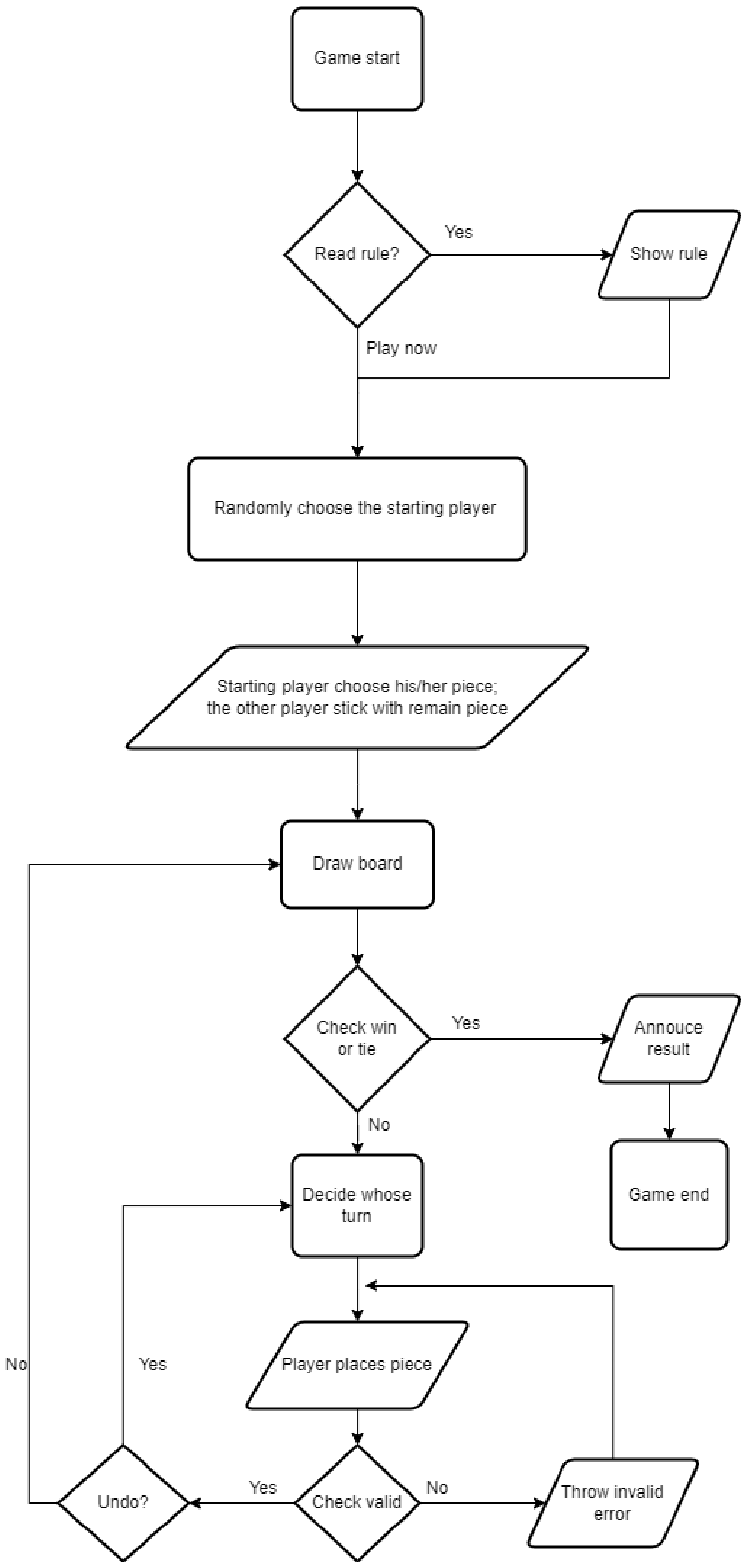
Four in a Row is the classic two-player game where you take turns to place a counter in an upright grid and try and beat your opponent to place 4 counters in a row.



The game is played with a **seven-column** and **six-row** grid, which is arranged upright. **The starting player is randomly chosen**, picks a game piece color (yellow or red), and can place a piece in any column. Each player then alternately takes a turn placing a piece in any column that is not already full. The piece falls straight down, occupying the lowest available spot within the column, or is stopped by another piece. The aim is to be the first of the two players to connect four pieces of the same color vertically, horizontally, or diagonally. If each cell of the grid is filled and no player has already connected four pieces, the game ends in a draw, so no player wins.

II. IMPLEMENTATION

The game's general flow will be presented in the following chart:



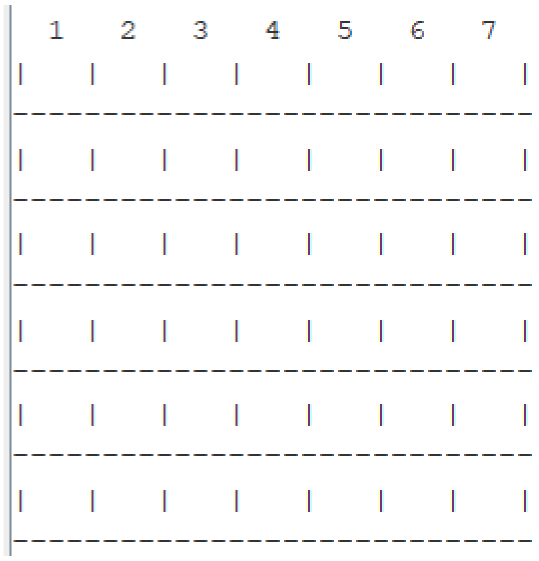
Next, we will go into detail about the important components of the game.

1. Choose the starting player

In this game, the starting player will be chosen randomly. The random mechanic I use is using MIPS syscall instruction number 42 to generate a number from 0 to 9; if the generated number then the starting player will be Player 1, else Player 2.

2. Draw board

The board in this game is a seven-column and six-row board. I use 5 registers, $t0 to $t5, as 6 rows with $t0 hold the lowest row and then go up. Between the rows are multiple '-' act as a "floor" of the row.



```
  1   2   3   4   5   6   7
|   |   |   |   |   |   |   |
-----------------------------
|   |   |   |   |   |   |   |
-----------------------------
|   |   |   |   |   |   |   |
-----------------------------
|   |   |   |   |   |   |   |
-----------------------------
|   |   |   |   |   |   |   |
-----------------------------
|   |   |   |   |   |   |   |
-----------------------------
```

In this game of mine, the index of elements that hold the piece for 7 columns are 3,7,11,15,19,23,27 from left to right accordingly.

3. Decide turn

When the first player is chosen, a register will be assigned to remember who goes first, I choose $a1 in this case. If $a1= 1, player 1 goes first; else ($a1 =2) player 2.

Then when going to the *decideturn* function, $a1 will be divided by 2, if the remainder is 0, it is player 2 turn; else (remainder =1) it is player 1 turn.

Also in this part, $s1 will be assigned the piece of player 1, and $s2 will be assigned that of player 2. These registers will be used in the next part.

4. Place piece

For in decide turn, we see if this is player 1 or player 2 turn, then decide X or Y will be placed

The game then will ask the player to type in which column he/she wants to place the piece. If the input is invalid, the game throws an invalid error, and the player inputs the column again.

If the input is valid, the game will go to the column that the player chooses and start to check if the cell is empty, starting from the bottom cell. If the cell has a piece in it, it will go up until reaching the highest row. If the cell on the highest row is not empty, the game will notify the player that the column is full and ask the player to choose another column.

For example, Player chooses column 1, the game will check 3($t0). If 3($t0) full, game go to 3($t1) and check if 3($t1) is empty or not, …

5. Undo

From the start of the game, there are 2 registers to count undo turns of the two players, both are equal to 3. Each time right after inputting the column to place the piece, the game will ask the player if they want to undo the move or not. If the player undoes, the undo register assigned to that player will be minus 1 and the player can type in the new move. If the undo register reaches 0, the game will not ask the player to undo anymore.

6. Check Tie

From the start of the game, there is a register to count turns, it is initially set to 0. After a player makes a move successfully, count will be plus 1. If the count reaches 42 (6 x 7) and there is still no winner, the game will announce the game is tied.

7. Check Win

a. Horizontally

First the game start from the first cell on the left of the bottom row, 3($t0).

- If 3($t0) is empty, go to the next case

- If the next cell, 7($t0), is not equal 3($t0), go to the next case

- If 11($t0) not equal 3($t0), go to the next case

- If 15($t0) not equal 3($t0), go to the next case

- If still not jump then it mean there are “4 in a row”, the game jump to the winning case

Then the game repeats this checking from 7($t0) until 15($t0) then go to the next row, $t1. After checking all possible horizontal cases, if there is any winning case, the game goes to checking vertically.

b. Vertically

Example:

- If 3($t0) is empty, go to the next case

- If the next cell, 3($t1), is not equal 3($t0), go to the next case

- If 3($t2) not equal 3($t0), go to the next case

- If 3($t3) not equal 3($t0), go to the next case

- If still not jump then it mean there are “4 in a row”, the game jump to the winning case

With the same mechanic as horizontally, the game checks all possible vertical cases. If there is no winning situation, the game goes to diagonally check.

c. Diagonally

First, we go from left to right, for example: checking 3($t0), 7($t1), 11($t2), and 15($t3).

Then if there is no winning case, we go from right to left. For example: checking 27($t0), 23($t1), 19 ($t2), 15($t3).

If there is still no winning case, the game goes back to *decideturn*. The game will repeat with the other player’s turn.