

ECE114L – PROJECT 1

TIC TAC TOE

Kim Nguyen – Jim Son

November 18, 2017

## **I. OBJECTIVE**

Build a C++ program for the game Tic Tac Toe, with 2 different modes: player vs. computer, and player vs. player.

## **II. BACKGROUND**

We had initially started this program using specific user inputs to select specific boxes on the playing field. Using this method brought many problems because it allowed the user to input characters that should not be inputted. We had run into an infinite loop when there was a wrong input. This was fixed, but other bugs remained. This idea eventually became scratched when further research was done on a user-friendly menu which was supposed to only be an addition to make the program more presentable after the game was finished. This menu used arrow keys to choose an option as well as the return/enter key to select. After understanding how the virtual keys were utilized for a menu, we realized it could be used on the actual game. This had ultimately made the program very user-friendly, and it also avoided the bugs from the previous idea of specific user inputs since the only available inputs are the arrow keys and return/enter.

## **III. RESULTS**

We created a program that asks user to use the arrow keys to move to and select the desired positions, and the Esc key to escape the program.

The steps of the program are illustrated in Figure 3.1:

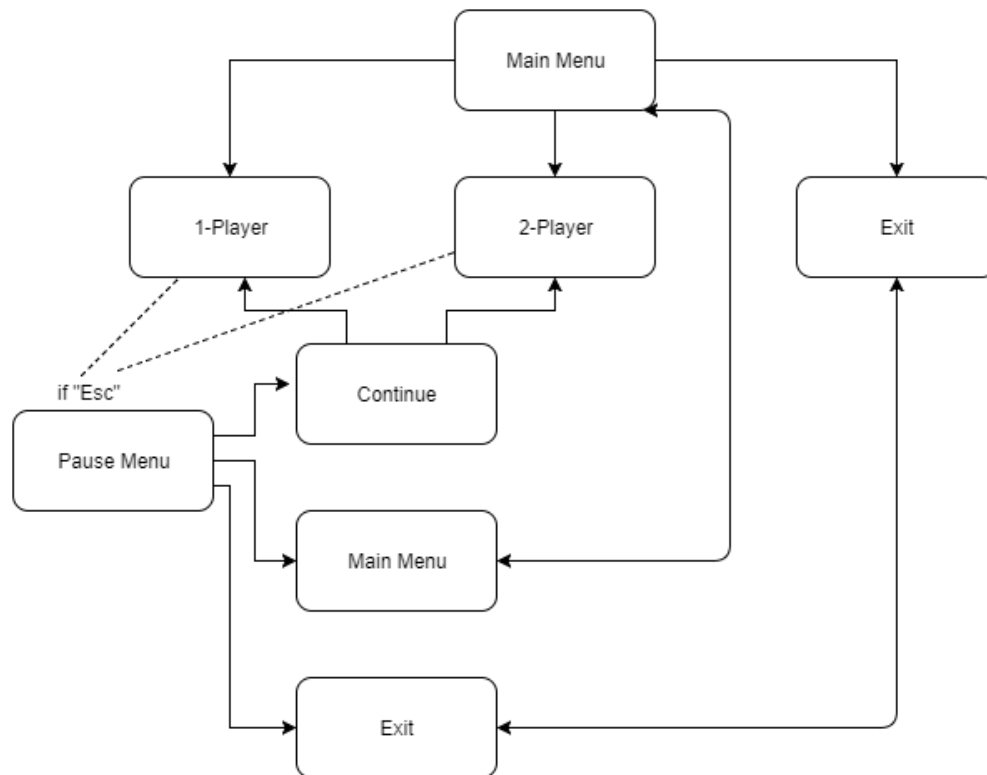


Figure 3.1 – Program Flow Chart



Figure 3.2 – Screenshot of the program output – Main menu

```
main
*** TIC TAC TOE ***
Please move the arrow keys to the position you want, enter to choose:
Fatfat 1's turn (X):
*-----*
|  X  | -  | -  |
|-----|-----|-----|
|  -  | X  | -  |
|  ^^^  |
|-----|-----|-----|
|  -  | 0  | 0  |
*-----*

Current Score
=====
Fatfat 1  Computer
0          0

Press 'Escape' to pause the game.
```

Figure 3.3 – Screenshot of the program output – “1-Player” (player vs. computer) mode

```
main
*** TIC TAC TOE ***
Please move the arrow keys to the position you want, enter to choose:
Game ended! Fatfat 1 Wins!
*-----*
|  X  | 0  | 0  |
|-----|-----|-----|
|  X  | X  | X  |
|  ^^^  |
|-----|-----|-----|
|  0  | X  | 0  |
*-----*

Current Score
=====
Fatfat 1  Computer
1          0

Press 'Escape' to pause the game.

Would you like to play again?
Yes      No
^^^^_
```

Figure 3.4 – Screenshot of the program output – One player wins a round

```

C:\main
*** TIC TAC TOE ***
Please move the arrow keys to the position you want, enter to choose:
DRAW GAME!

*-----*
|  X  |  O  |  X  |
|-----|-----|-----|
|  O  |  X  |  X  |
|-----|-----|-----|
|  O  |  X  |  O  |
|-----|-----|-----|

      ^^^

*-----*

Current Score
=====
Fatfat 1 Fatfat 2
2           0

*-----*


Press 'Escape' to pause the game.

Would you like to play again?

Yes      No
^^^

```

Figure 3.5 – Screenshot of the program output – “2-Player” (player vs. player) mode



```
*-----*
|
|          PAUSED
|          -----
|          *-----*
|          |Continue|
|          *-----*
|          Main Menu
|
|          Quit
|
*-----*
```

Figure 3.6 – Screenshot of the program output – Pause menu

#### **IV. ANALYSIS**

##### **Role of each member:**

- **Jim:**
  - Set up the program's layout, research on using arrow keys in displaying the program, write the menus and taking user's inputs
  - Write the 2-player mode
  - Research on doing the computer's moves
- **Kim:**
  - Set up the class and its functions
  - Write the 1-player mode
  - Research on doing the computer's moves

##### **Result Discussion:**

We were successful in creating a program that runs how it was planned and even more.

##### **Calculations Discussion:**

Simple addition and subtraction with respect to x and y coordinates are used to place items at specific positions on the console. Because we set the coordinates for the play field and menus ourselves, we are able to set limits telling the program how many times or how far the directional arrow keys can be pressed using "if" statements. As a directional key is pressed, either the x or y variable will increase or decrease moving the cursor. All the while variables controlling the multidimensional array are also being increased by 1 or decreased by 1 to signify where on the play field to place an X or O. This all depends on what key is pressed. This same method is used to move the cursor for choices on the menus.

## V. CONCLUSION:

- During this project, we practiced what we have learned in class, and also learned to use array, use random function to generate a random number, the Windows libraries to use the arrow keys and to place the cursor at a specific position in the screen, system() functions to handle the screen output as well as how to use class.
- We intended to create another mode, in which the computer is “smarter” by applying Minimax Algorithm in game theory, but we did not have enough time to do research and include this part in the program.
- We used object-oriented programming in this project, but it was not the most effective, as the program is highly dependent on the arrow keys and we was not successful in finding a way to write the code more effectively.
- The program uses system() functions which does what it needs to do on Windows only, so it does not run as smoothly on other platforms.
- Besides the difficulties that has been discussed above, there are other things we can improve on, such as we should make a clearer strategy before actually starting to write the code.

## VI. APPENDIX:

### tictactoe.h

```
#ifndef TICTACTOE_CLASS
#define TICTACTOE_CLASS

#include <iostream>
#include <string>
#include <cstdlib>
#include <Windows.h>
using namespace std;

HANDLE screen = GetStdHandle(STD_OUTPUT_HANDLE);
```

```
COORD coordinate;

class TicTacToe {
public:
    TicTacToe();
    TicTacToe(string, string);
    int onePlayer();
    void computerMove(int a, int b, char pF[3][3]);
    int twoPlayer();
    void menu();
    void drawBoard(string name1, string name2, int gameMode);
    void setP(int x, int y);

private:
    int gameMode[2];
    bool menuOn;
    int row, menuChoice, pause_menu;
    int x = 12, y = 6, turn = 0, a = 0, b = 0;
    bool inGame = true, drawBoardOn = true, choice = true, pause;
    int player1Count = 0, player2Count = 0;
    string name1, name2;
};

#endif
```

## tictactoe.cpp

```
#include "tictactoe.h"

//-----default constructor-----//
/*
    Description: if no parameters, the initial values for name1 and name2 is "Player 1" and
    "Player 2"
*/
TicTacToe::TicTacToe() {
    this->name1 = "Player 1";
    this->name2 = "Player 2";
}

//-----converting constructor-----//
/*
    Description: if there are parameters, set the given values for name1 and name2
*/
TicTacToe::TicTacToe(string name1, string name2) {
    this->name1 = name1;
    this->name2 = name2;
}

//-----onePlayer()-----//
/*
    Description: - called when the user choose the player vs. computer mode
                - name2 is replaced with "Computer"
*/
```



```

    Note: needs to be developed more to add the AI computer mode
*/
int TicTacToe::onePlayer() {
    drawBoard(name1, "Computer", 0);
}
//*****

//-----computerMove()-----//
/*
    Description: user random to determine computer's move and place 'O' in the current
    position
    Variables:
        - int a, b: the current values of the pF array that is being worked on
*/
void TicTacToe::computerMove(int a, int b, char pF[3][3]) {
    while (pF[a][b] != '-') {
        a = rand() % 3;
        b = rand() % 3;
    }
    pF[a][b] = 'O';
}
//*****
*****

//-----twoPlayer()-----//
/*
    Description: called when the user choose the player vs. player mode
*/
int TicTacToe::twoPlayer() {
    drawBoard(name1, name2, 1);
}
//*****
*****

//-----menu()-----//
/*
    Variables:
        + bool menuOn: true if the menu is displayed
        + int row: the position of the current user's choice
        + int menuChoice: the current user's choice
    Description: start the game by drawing a menu with 3 choices:
        + "1 Player" (menuChoice == 0): human player with computer
        + "2 Player" (menuChoice == 1): 2 human players
        + "Exit" (menuChoice == 2): exit the program
*/
void TicTacToe::menu() {
    this->menuOn = true;
    row = 9, menuChoice = 0;

    setP(30, 9); cout << "*-----*";
    setP(30, 10); cout << "|"; setP(39, 10); cout << "|";
    setP(30, 11); cout << "*-----*";

    while (menuOn == true)
    {

```

```

if (menuChoice == 0)
{
    setP(23, 18); cout << " ";
    setP(23, 18); cout << "Player Against Computer.";
}

else if (menuChoice == 1)
{
    setP(23, 18); cout << " ";
    setP(23, 18); cout << "Play Against A Friend!";
}
else
{
    setP(23, 18); cout << " ";
    setP(23, 18); cout << "Exit The Game.";
}

setP(20, 5); cout << "*-----*";
setP(20, 6); cout << "|"; setP(50, 6); cout << "|";
setP(20, 7); cout << "|"; setP(30, 7); cout << "TIC-TAC-TOE"; setP(50, 7); cout <<
"|";
setP(20, 8); cout << "|"; setP(30, 8); cout << "-----"; setP(50, 8); cout <<
"|";
setP(20, 9); cout << "|"; setP(50, 9); cout << "|";
setP(20, 10); cout << "|"; setP(31, 10); cout << "1-Player"; setP(50, 10); cout <<
"|";
setP(20, 11); cout << "|"; setP(50, 11); cout << "|";
setP(20, 12); cout << "|"; setP(31, 12); cout << "2-Player"; setP(50, 12); cout <<
"|";
setP(20, 13); cout << "|"; setP(50, 13); cout << "|";
setP(20, 14); cout << "|"; setP(31, 14); cout << "Exit"; setP(50, 14); cout << "|";
setP(20, 15); cout << "|"; setP(50, 15); cout << "|";
setP(20, 16); cout << "*-----*";

system("pause>nul");

if (GetAsyncKeyState(VK_DOWN) && row != 13)
{
    setP(30, row); cout << " ";
    setP(30, row + 1); cout << " "; setP(39, row + 1); cout << " ";
    setP(30, row + 2); cout << " ";
    row += 2;
    menuChoice++;

    if (row == 11)
    {
        setP(30, row); cout << "*-----*";
        setP(30, row + 1); cout << "|"; setP(39, row + 1); cout << "|";
        setP(30, row + 2); cout << "*-----*";
    }

    if (row == 13)
    {
        setP(30, row); cout << "*-----*";
        setP(30, row + 1); cout << "|"; setP(35, row + 1); cout << "|";
        setP(30, row + 2); cout << "*-----*";
    }
}
}

```

```

if (GetAsyncKeyState(VK_UP) && row != 9)
{
    if (row == 13)
    {
        setP(35, row + 1); cout << " ";
    }

    setP(30, row); cout << " ";
    setP(30, row + 1); cout << " "; setP(39, row + 1); cout << " ";
    setP(30, row + 2); cout << " ";
    row -= 2;
    menuChoice--;
    setP(30, row); cout << "*-----*";
    setP(30, row + 1); cout << "|"; setP(39, row + 1); cout << "|";
    setP(30, row + 2); cout << "*-----*";
}

if (GetAsyncKeyState(VK_RETURN))
{
    if (menuChoice == 0)
        onePlayer();
    if (menuChoice == 1)
        twoPlayer();
    if (menuChoice == 2)
    {
        menuOn = false;
        break;
    }
}
}

setP(0, 23);
}
//*****

//-----drawBoard()-----//
/*
Description: draw the play field and play the game after calling the menu and choosing
the gameMode, also display the pause menu if the RETURN key is pressed
Variables:
- bool inGame: main loop for game inputs
- bool drawBoardOn: loop to display the play field or exit the play field
- bool choice: loop to choose to replay or end program
- bool pause: to display the pause menu
- int row: current position of user's choice
- int player1Count, player2Count: current scores of 2 players (+1 per winner per match)
- int pause_menu: user's choice in the pause menu (0 = continue, 1 = main menu, 2 =
exit)
- char pF[][]: the multidimensional array that store the players' mark in a match
(initial value = '-', name1's mark = 'X', name2's mark = 'O')
- int turn: count the turn
- int x, y: screen coordinate
- int a, b: current position in the play field (pF)
*/
void TicTacToe::drawBoard(string name1, string name2, int gameMode) {
    inGame = true, drawBoardOn = true, choice = true, pause;

```

```

row = 0, player1Count = 0, player2Count = 0, pause_menu = 0;

while (drawBoardOn == true) //Loops the whole program, choice at the end to break
{
    system("cls");          // clear screen

    char pF[3][3] = { '-', '-', '-', '-', '-', '-', '-', '-', '-' }; //The array which holds the
X's and O's
    x = 12, y = 6, turn = 0, a = 0, b = 0;

    setP(x, y); cout << "^^^";

    while (inGame == true)
    {
        //Draws a playing field.
        setP(10, 3); cout << "*-----*";
        setP(10, 5); cout << "| " << pF[0][0] << " | " << pF[0][1] << " | " <<
pF[0][2] << " | ";
        setP(10, 7); cout << "|----|----|----|";
        setP(10, 9); cout << "| " << pF[1][0] << " | " << pF[1][1] << " | " <<
pF[1][2] << " | ";
        setP(10, 11); cout << "|----|----|----|";
        setP(10, 13); cout << "| " << pF[2][0] << " | " << pF[2][1] << " | " <<
pF[2][2] << " | ";
        setP(10, 15); cout << "*-----*";

        setP(36, 5); cout << "Current Score";
        setP(33, 7); cout << name1;
        setP(43, 7); cout << name2;
        setP(36, 9); cout << player1Count;
        setP(47, 9); cout << player2Count;
        setP(32, 4); cout << "*-----*";
        setP(32, 5); cout << "|"; setP(51, 5); cout << "|";
        setP(32, 6); cout << "|"; setP(51, 6); cout << "|"; setP(36, 6); cout <<
"=====
        setP(32, 7); cout << "|"; setP(51, 7); cout << "|";
        setP(32, 8); cout << "|"; setP(51, 8); cout << "|";
        setP(32, 9); cout << "|"; setP(51, 9); cout << "|";
        setP(32, 10); cout << "*-----*";

        setP(32, 13); cout << "Press 'Escape' to pause the game.";

        if (inGame == true)
        {
            setP(10, 0); cout << "*** TIC TAC TOE ***" << endl;

            if (turn % 2 == 0) //Checks who's turn it is
            {
                setP(10, 2); cout << name1 << "'s turn (X): " << endl;
            }
            else
            {
                setP(10, 2); cout << name2 << "'s turn (O): " << endl;
            }

            setP(2, 1); cout << "Please move the arrow keys to the position you want, enter
to choose: ";

```

```

        setP(0, 17);
    }
    if (turn >= 5)        //Checks for a win on turn 5
    {
        if ( //Possible positions of X to win.
            ((pF[0][0] == 'X') && (pF[1][0] == 'X') && (pF[2][0] == 'X')) || ((pF[0][1] ==
            'X') && (pF[1][1] == 'X') && (pF[2][1] == 'X')) ||
            ((pF[0][2] == 'X') && (pF[1][2] == 'X') && (pF[2][2] == 'X')) || ((pF[0][0] ==
            'X') && (pF[0][1] == 'X') && (pF[0][2] == 'X')) ||
            ((pF[1][0] == 'X') && (pF[1][1] == 'X') && (pF[1][2] == 'X')) || ((pF[2][0] ==
            'X') && (pF[2][1] == 'X') && (pF[2][2] == 'X')) ||
            ((pF[0][0] == 'X') && (pF[1][1] == 'X') && (pF[2][2] == 'X')) || ((pF[0][2] ==
            'X') && (pF[1][1] == 'X') && (pF[2][0] == 'X'))
        )
        {
            setP(7, 2); cout << "Game ended! " << name1 << " Wins!" << endl;
            player1Count++;
            inGame = false;
            break;
        }
        else if ( //Possible positions of 0 to win.
            ((pF[0][0] == 'O') && (pF[1][0] == 'O') && (pF[2][0] == 'O')) || ((pF[0][1] ==
            'O') && (pF[1][1] == 'O') && (pF[2][1] == 'O')) ||
            ((pF[0][2] == 'O') && (pF[1][2] == 'O') && (pF[2][2] == 'O')) || ((pF[0][0] ==
            'O') && (pF[0][1] == 'O') && (pF[0][2] == 'O')) ||
            ((pF[1][0] == 'O') && (pF[1][1] == 'O') && (pF[1][2] == 'O')) || ((pF[2][0] ==
            'O') && (pF[2][1] == 'O') && (pF[2][2] == 'O')) ||
            ((pF[0][0] == 'O') && (pF[1][1] == 'O') && (pF[2][2] == 'O')) || ((pF[0][2] ==
            'O') && (pF[1][1] == 'O') && (pF[2][0] == 'O'))
        )
        {
            setP(7, 2); cout << "Game ended! " << name2 << " Wins!" << endl;
            player2Count++;
            inGame = false;
            break;
        }
    }

    if (turn == 9) //If no one wins in 9 turns, the game ends in a draw.
    {
        setP(7, 2); cout << " " << endl;
        setP(15, 2); cout << "DRAW GAME!" << endl;
        inGame = false;
        break;
    }
}

//cin.ignore(); (if pause can't be used)
//cin.get(); (if pause can't be used)
system("pause>nul"); //Pauses the active screen

```

```

    if (GetAsyncKeyState(VK_DOWN) && y != 14) //Moves the cursor down until y = 14 and
increases the "a" value for pF[a][b]
    {
        setP(x, y); cout << " ";
    }

```

```

        y += 4;
        setP(x, y); cout << "^^^";
        a++;
        continue;
    }

    if (GetAsyncKeyState(VK_UP) && y != 6)    //Moves the cursor down until y = 6 and
decreases the "a" value for pF[a][b]
    {
        setP(x, y); cout << "   ";
        y -= 4;
        setP(x, y); cout << "^^^";
        a--;
        continue;
    }

    if (GetAsyncKeyState(VK_LEFT) && x != 12) //Moves the cursor left until x = 12 and
decreases the "b" value for pF[a][b]
    {
        setP(x, y); cout << "   ";
        x -= 6;
        setP(x, y); cout << "^^^";
        b--;
        continue;
    }

    if (GetAsyncKeyState(VK_RIGHT) && x != 24) //Moves the cursor right until x = 24 and
increases the "b" value for pF[a][b]
    {
        setP(x, y); cout << "   ";
        x += 6;
        setP(x, y); cout << "^^^";
        b++;
        continue;
    }

    if (GetAsyncKeyState(VK_RETURN)) //When enter/return is pressed, checks for even #
rounds and checks if the element is already an 'X' or 'O'
    {
        if (gameMode == 0) {
            if ((turn % 2 == 0) && (pF[a][b] == '-'))    //Places an X into pF[a][b] if the
round is even
            {
                pF[a][b] = 'X';
                turn++;

                if (turn < 8) {
                    computerMove(a, b, pF);
                    turn++;
                }
            }
        }

        if (gameMode == 1) {
            if ((turn % 2 == 0) && (pF[a][b] == '-'))    //Places an X into pF[a][b] if the
round is even
            {
                pF[a][b] = 'X';
                turn++;
            }
        }
    }

```

```

    }
    else if (!(turn % 2 == 0) && (pF[a][b] == '-')) //Places an O into pF[a][b]
if the round is Odd
    {
        pF[a][b] = 'O';
        turn++;
    }
}

}

if (GetAsyncKeyState(VK_ESCAPE)) //Opens the pause menu
{
    pause = true;
    int z = 9;
    system("cls");
    setP(30, 9); cout << "*-----*";
    setP(30, 10); cout << "|"; setP(39, 10); cout << "|";
    setP(30, 11); cout << "*-----*";
    while (pause == true)
    {
        setP(20, 5); cout << "*-----*";
        setP(20, 6); cout << "|"; setP(49, 6); cout << "|";
        setP(20, 7); cout << "|"; setP(32, 7); cout << "PAUSED"; setP(49, 7); cout <<
"|";
        setP(20, 8); cout << "|"; setP(32, 8); cout << "-----"; setP(49, 8); cout <<
"|";
        setP(20, 9); cout << "|"; setP(49, 9); cout << "|";
        setP(20, 10); cout << "|"; setP(31, 10); cout << "Continue"; setP(49, 10); cout
<< "|";
        setP(20, 11); cout << "|"; setP(49, 11); cout << "|";
        setP(20, 12); cout << "|"; setP(31, 12); cout << "Main Menu"; setP(49, 12);
cout << "|";
        setP(20, 13); cout << "|"; setP(49, 13); cout << "|";
        setP(20, 14); cout << "|"; setP(31, 14); cout << "Quit"; setP(49, 14); cout <<
"|";
        setP(20, 15); cout << "|"; setP(49, 15); cout << "|";
        setP(20, 16); cout << "*-----*";

        system("pause>nul");

        if (GetAsyncKeyState(VK_DOWN) && z != 13)
        {
            setP(30, z); cout << " ";
            setP(30, z + 1); cout << " "; setP(39, z + 1); cout << " ";
            setP(30, z + 2); cout << " ";
            z += 2;
            pause_menu++;

            if (z == 11)
            {
                setP(30, z); cout << "*-----*";
                setP(30, z + 1); cout << "|"; setP(40, z + 1); cout << "|";
                setP(30, z + 2); cout << "*-----*";
            }
        }
    }
}

```

```

    if (z == 13)
    {
        setP(30, z); cout << "*-----*";
        setP(30, z + 1); cout << "|"; setP(35, z + 1); cout << "|";
        setP(30, z + 2); cout << "*-----*";
    }
}

if (GetAsyncKeyState(VK_UP) && z != 9)
{
    if (z == 13)
    {
        setP(35, 14); cout << " ";
    }

    setP(30, z); cout << " ";
    setP(30, z + 1); cout << " "; setP(40, z + 1); cout << " ";
    setP(30, z + 2); cout << " ";
    z -= 2;
    pause_menu--;

    if (z == 9)
    {
        setP(30, z); cout << "*-----*";
        setP(30, z + 1); cout << "|"; setP(39, z + 1); cout << "|";
        setP(30, z + 2); cout << "*-----*";
    }

    if (z == 11)
    {
        setP(30, z); cout << "*-----*";
        setP(30, z + 1); cout << "|"; setP(40, z + 1); cout << "|";
        setP(30, z + 2); cout << "*-----*";
    }
}

if (GetAsyncKeyState(VK_RETURN))
{
    if (pause_menu == 0)
    {
        pause = false;
        system("cls");
        setP(x, y); cout << "^^^";
        break;
    }
    if (pause_menu == 1)
    {
        pause = false;
        inGame = false;
        drawBoardOn = false;
        choice = false;
        system("cls");
        row = 9;
        menuChoice = 0;
        setP(30, row); cout << "*-----*";
        setP(30, row + 1); cout << "|"; setP(39, row + 1); cout << "|";
        setP(30, row + 2); cout << "*-----*";
    }
}

```



```

        break;
    }
    if (pause_menu == 2)
    {
        pause = false;
        inGame = false;
        drawBoardOn = false;
        choice = false;
        menuOn = false;
        break;
    }
} //End of if(GetAsyncKeyState(VK_RETURN))
} //End of while pause == true
} // End of if(GetAsyncKeyState(VK_ESCAPE))
}

if (!(GetAsyncKeyState(VK_ESCAPE)) && choice == true)
{
    setP(0, 18); cout << "Would you like to play again?" << endl;
    setP(2, 20); cout << "Yes";
    setP(9, 20); cout << "No";
    x = 2;
    setP(2, 21); cout << "^^^";
}

while (choice == true) //Choice of playing more or closing the program.
{
    system("PAUSE>nul");

    if (GetAsyncKeyState(VK_LEFT) && x != 2) //Left or right selection for yes or no,
increases or decreases "row"
    {
        setP(x, 21); cout << "    ";
        x -= 7;
        setP(x, 21); cout << "^^^";
        row--;
    }
    else if (GetAsyncKeyState(VK_RIGHT) && x != 9)
    {
        setP(x, 21); cout << "    ";
        x += 7;
        setP(x, 21); cout << "^^";
        row++;
    }
}

if (GetAsyncKeyState(VK_RETURN)) //Takes the "row" value to drawBoardOn or end
the program
{
    if (row == 0)
    {
        inGame = true;
        break;
    }
    else if (row == 1)
    {
        drawBoardOn = false;
        system("cls");
        row = 9;
    }
}

```

