**Project 1**

**[Connect Four]**

**CIS17A**

**Name: Prieto, Heriberto**

**Date: 05/12/2017**

**Introduction**

Title: Connect Four

Classic board game consisting of two players.

Choose a column to drop a palette. Attempt to connect 4 palettes in any format before your opponent.

When either player 1 or player 2 have connected four, they have won and the game is over.

**Summary**

Project Size: Around 128 lines(Including spacing)

Number of Variables: 7 variables

Number of Methods: 6

Conceptually this game isn’t the too difficult to build. Main focus for this project was to build an efficient project with respect to space. I used ternary operators as much as possible, attempted to steer from if-statements as much as possible. Im EXTREMELY comfortable with working with 2D Arrays now and I hope it shows. I am really happy with how short the project is and I believe the code is very reader friendly.

**Pseudo Code**

-> Declare Variables

-> Allocate Memory for 2D Array with 6 rows and 7 columns.

-> Begin while loop that takes in user input and performs Connect Four Game Logic.

**Program:**

#include <cstdlib>

#include <iostream>

using namespace std;

int \*\*fillBrd(int,int);

void prntBrd(int \*\*,int,int);

void rsetBrd(int \*\*,int,int);

void dropChp(int \*\*,int,int,int);

int gameWon(int \*\*,int,int);

void endGame(int \*\*,int,int);

int main(int argc,char\*\* argv) {

//Seed

srand(static\_cast<unsigned int>(time(0)));

//Declare variables

int won = 0; //False while four have not been connected

int columns = 7; //Columns in board

int rows = 6; //Rows in board

int turns = columns \* rows;//Max number of turns

int \*\*board = fillBrd(columns,rows);

int place; //Position where chip will be dropped

int player; //Depending on turn, keeps track of player turns

while (turns >= 0 && !won) {

//Output to player(s)

cout<<"Turns left: "<<turns<<endl;

cout<<"Choose a column from 1 - 7 and drop a chip."<<endl;

prntBrd(board,columns,rows);

(turns % 2 == 0)?cout<<"Player 1's turn:"<<endl:cout<<"Player 2's turn:"<<endl;

cin.clear();

cin>>place;

player = (turns%2==0)?1:2;

//"Drop Chip" into array based on players decision

if (place > 0 && place < 8) dropChp(board,rows,place,player); else ++turns;

//Check if player won

won = gameWon(board,rows,columns);

//Announce winner

if (won != 0) cout<<"Player: "<<won<<" wins!"<<endl; else turns--;

}

//Print gameboard

prntBrd(board,columns,rows);

//Deallocate memory

endGame(board,columns,rows);

}

/\* Allocates memory for 2DArray/Connect Four Game Board \*/

int \*\*fillBrd(int cols,int rows) {

//Initialize 2DArray

int \*\*board = new int\*[rows];

for (int r = 0; r < rows; r++)

board[r] = new int[cols];

//Set board to all zeros

rsetBrd(board,cols,rows);

return board;

}

/\* Prints Connect Four Board/2D Array \*/

void prntBrd(int \*\*board,int cols,int rows) {

cout<<"----------------------------------"<<endl;

for (int r = 0; r < rows; r++) {

for (int c = 0; c < cols; c++)

cout<<board[r][c]<<" || ";

cout<<endl;

}

cout<<endl;

}

/\* Resets values of 2D Array/Connect Four Board to 0 \*/

void rsetBrd(int \*\*board,int cols,int rows) {

for (int r = 0; r < rows; r++)

for (int c = 0; c < cols; c++)

board[r][c] = 0;

}

/\* Places value at given column, lowest value row without value \*/

void dropChp(int \*\*board,int rows,int place,int player) {

for (int r = rows - 1; r >= 0; r--)

if (board[r][place - 1] == 0) {

board[r][place - 1] = player;

break;

}

}

/\* Game Winning Logic, returns player that won. \*/

int gameWon(int \*\*board,int rows,int cols) {

for (int r = 0; r < rows; r++)

for (int c = 0; c < cols; c++)

if (board[r][c] != 0) {

//Bottom Half Vertical Win

if (r > 2)

if (board[r][c] == board[r - 1][c])

if (board[r - 1][c] == board[r - 2][c])

if (board[r - 2][c] == board[r - 3][c])return board[r][c];

//Top Half Vertical Win

if (r < 3)

if (board[r][c] == board[r + 1][c])

if (board[r + 1][c] == board[r + 2][c])

if (board[r + 2][c] == board[r + 3][c])return board[r][c];

//Reverse Diagonal Win

if (r < 3 && c < 4)

if (board[r][c] == board[r + 1][c + 1])

if (board[r + 1][c + 1] == board[r + 2][c + 2])

if (board[r + 2][c + 2] == board[r + 3][c + 3])return board[r][c];

//Diagonal Win

if (r < 3 && c < 4)

if (board[r][c] == board[r + 1][c - 1])

if (board[r + 1][c - 1] == board[r + 2][c - 2])

if (board[r + 2][c - 2] == board[r + 3][c - 3])return board[r][c];

//Horizontal Win

if (c < 4)

if (board[r][c] == board[r][c + 1])

if (board[r][c + 1] == board[r][c + 2])

if (board[r][c + 2] == board[r][c + 3])return board[r][c];

}

return 0;

}

/\* Deallocates 2D Array memory/End of Game \*/

void endGame(int \*\*board,int cols,int rows) {

for (int r = 0; r < rows; r++)

delete []board[r];

delete []board;

}