Project Report

Connect4FX Game

Yassr Shaar

**Student ID**: 14328571

**MSc. in Computer Science NL**

**Module Code:** COMP20300

**Module** **Title**: Java programming

**Lecturer:** Dr Simon Caton

Logo

Description automatically generated

UCD School of Computer Science

University College Dublin

December 18, 2020

Table of Contents

[Introduction 3](#_Toc59211455)

[Game Implementation 3](#_Toc59211456)

[Connect4 trivia 3](#_Toc59211457)

[Class Structure 4](#_Toc59211458)

[Class Diagram 5](#_Toc59211459)

[Game Design thoughts 6](#_Toc59211460)

[Encapsulation 6](#_Toc59211461)

[Model View Controller 6](#_Toc59211462)

[Game Unit Testing 7](#_Toc59211463)

[Improvements over Traditional game 8](#_Toc59211464)

# Introduction

The game chosen for this JavaFX project was Connect4.

Connect4 is a two player, turn-based board game that is typically comprised of a board that is seven columns in length and six rows in height.

This game was chosen as it offered a simple concept of gameplay, the players only need to choose a column where a disc would drop, however, the complexity of the game comes with the different ways a player can win. This level of complexity coupled with the desire to “make more” of this traditional game is what compelled me towards the project.

## Game Implementation

This project’s implementation of Connect4 provides further options to players which include customisation of the player name, colour, board size and finally the options to pause or play music. This is a very visual implementation of the traditional game and is aimed to be aesthetically pleasing and enjoyable.

The game rules are in line with the traditional game in that any player is capable of winning by having a combination of four discs of the same colour in any of the following directions:

* Horizontally
* Vertically
* Diagonally (to the left or right)

This implementation does also take into consideration the possibility of a draw occurring where neither player wins and there are no empty tiles on the board to place a disc.

## Connect4 trivia

**Connect4 trivia** that I discovered purely through the designing of this game (quite possibly the creators of the game did not even realise this).

The original colours for Connect4 are Yellow & Red.

When we look at a HEX colour picker, we can see something very interesting:

When we select #ff0 we get the colour **Yellow** & if we were to **Connect4** 0’s (discs) to that hex colour #ff0000 we get **Red**!

Graphical user interface

Description automatically generated Graphical user interface

Description automatically generated

# Class Structure

The class structure of the JavaFX game begins with the GameMain. This is where we create our Stage and scene which we call to it the Main Menu preference method.

The Main Menu preference method returns a GridPane which is filled with customisation and graphics tailored to the player’s choices. Here the players can choose their name & colour for a basic game, however, they can also choose to have the background music playing or not, they can also optionally increase the size of the board to create a more complex game.

Whether the players chose to customise the game or go with the default settings, clicking the Start Game button will launch the game. This button performs the following:

1. Sets the music button image to display a No music sign. (this is relevant as the music stops during the end screen to allow for a different music clip to play)
2. Sets the Column and Row based on the player’s choice
3. Creates the two players and takes in their name and colour choices.
4. Changes Scenes by calling StartGame.

StartGame is a method within MainMenu which is instructed to firstly call the method within Disc which clears the board of any previous game attempts and retrieves the required elements to be displayed within the game. This includes a grid for the board and the discs that will be used to play the game along with the selector that aids with identifying which column the player is about to drop a disc in. The method will return a gameroot pane which will result in changing scenes to the main game.

The logic for the main part of the game is constructed within he GameDesign class while any actions that occur related to the game discs occurs within the Disc class.

GameDesign firstly holds the ‘not so constant’ constants for the diameters of the game. This would mainly include the Columns, rows and tile size (which is a nonadjustable size used to keep a consistent sizing of the board). Most importantly the player1Move Boolean is declared here and asserted to be true.

Some fundamentals are created here such as the board shape. This includes some visualisation elements to make the board appear 3D. Such things as punching holes in the board shape and sliding the disc behind them to appear like the traditional game in 3D. Also, the choice of lighting and shadows amplifies this effect.

The selector is created within this method to allow for a visual representation of where the mouse is hoovering on the board and which column would the disc drop on if the player were to click down. This would return a list of rows within the column for which the selector is hovering over.

The createPlayer method takes in the player's name and colour when they are assigned in the MainMenu class. The name is checked and manipulated to best suit the structure of the game. This is done by removing any spaces in the name (helps with saving the leader board), checking if the name is empty, if it is then a default name is given such as Player1; ensuring that the player name is no longer than 8 characters (for visual purposes).

The colour is also checked to determine if no colour is selected or returned, by default the colour RED will be assigned for player 1 and YELLOW for player 2. Since the Colour picker is used to retrieve the player colour choices this should never be an issue, but the precautions are being taken here.

The final two aspects of the game design are focused on consideration of whether the game has been won or if there are further discs to be played.

Whenever a disc is played, the gameEnd method is called and takes in the column and row of where the new disc landed. This method then uses the new disc coordinates to create 4 lists which take 3 coordinates off of where this disc is placed. The 4 lists are the coordinates of points vertical, horizontal and diagonal (from bottom left and bottom right) of our disc.

Once these are collected the checkWin method is called to determine if any of these lists would result in a win.

The Checkwin method takes the list of points from gameEnd and determines if the 4 coordinates are for discs of the same colour. If they are then the player who placed that last disc is the winner.

The logic for how the win is handled and the discs dropping is handled with in the Disc class.

Firstly, the constructor for Disc checks firstColour, which is a Boolean to determine which player's turn once that is known, the colour is assigned to the disc to be dropped.

## Class Diagram

Diagram

Description automatically generated

## Game Design thoughts

### Encapsulation

### Model View Controller

despite not implementing MVC as traditionally would be expected, the project does borrow some design considerations when approaching the handling of some elements. This can be more prominently seen with regards to the displaying of the End Screen of application. The creation of the model elements such as the pane, buttons, text and labels is all handled within a single method dedicated to the View, while the logic behind it is controlled within the body of the main method.

# Game Unit Testing

# Improvements over Traditional game