|  |  |
| --- | --- |
| Frustration  A SOFTWARE ENGINEERING DESIGN PROJECT | assignment 2017: gROUP pROJECT  Part 2: User Interface  GROUP  Rubab Ramzan - S00162293 Brain Mc Gowan - S00165159 Bryan Kerruish - S00173160 |

Table of Contents

[The users 2](#_Toc481695272)

[Beginning a game 2](#_Toc481695273)

[Rolling the dice 3](#_Toc481695274)

[The first move 4](#_Toc481695275)

[Options menu 5](#_Toc481695276)

[The last move 6](#_Toc481695277)

[The end of the game 7](#_Toc481695278)

[XAML Interface 7](#_Toc481695279)

[Save Class 8](#_Toc481695280)

[Pseudo Code for Save-Class 8](#_Toc481695281)

## The users

Frustration is a family game for all ages and abilities. The game has a simple set of rules and virtual no learning curb.

We have aimed to make our interface as simple and intuitive to use as possible to facilitate all age categories and computer ability.

## Beginning a game



On starting the application; the user has the options to start a new game or quit the game; or continue if a saved game is present.

Selecting new game, the number of players’ screen will be displayed.

Selecting continue will load the saved game and continue where left off.

Selecting quit will exit the application.



The number of players is the only requirement in setting up a game. To make this as simple as possible, the number of players’ choice is visually represented with the game pieces.

## Rolling the dice



When a player needs to roll the dice, a dialog with the players coloured piece and a die is displayed. When the dice is clicked, the dice roll is animated and the player is shown the board.

## The first move



The first move onto the board will always be as a six. In the above screen, red has rolled a six, all four red pieces are currently “home” and are highlighted. The player can click any of the four red pieces and it will move to the highlighted red “start position”.

The interface also has three areas below the board with the game information.

The left side contains the game log. This will display a log of all the rolls, moves and actions of the game.

The centre contains the dice result.

The right side contains the current player.

The game options can be accessed by clicking the options icon in the top left corner.

## Options menu



The save game option allows the player to save the current game. When a game is saved, the continue option on the initial screen is visible.

Change language provides the player the ability to change the game language.

Quit game provides a means to end the application.

## The last move



The game will end when a player moves all his pieces’ home. In the above screen, the green player has rolled a three and is able to move his final piece home.

## The end of the game

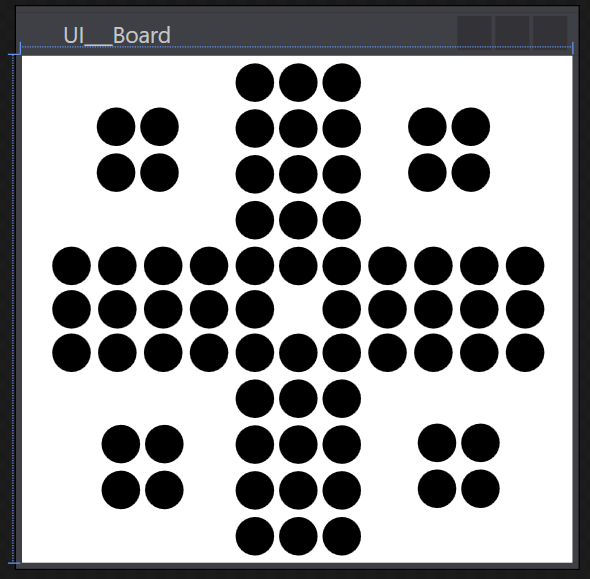


When the game ends, the game over message will be displayed along with the winner player. The player will have the options to play again or quit the game.

If play again is selected, we restart from the beginning.

Quit game will end the application.

## XAML Interface



<Window x:Class="Frustration.UI\_\_\_Board"

        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

        xmlns:local="clr-namespace:Frustration"

        mc:Ignorable="d"

        Title="UI\_\_\_Board" Height="300" Width="300">

    <Grid>

        <Viewbox xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation" Stretch="Uniform">

            <Canvas Name="svg5230" Width="1481.63" Height="1489">

                <Canvas.RenderTransform>

                    <TranslateTransform X="0" Y="0"/>

                </Canvas.RenderTransform>

                <Canvas.Resources/>

                <Canvas Name="g5228">

                    <Canvas Name="shape51">

                        <Canvas.RenderTransform>

                            <TranslateTransform X="18.12" Y="-815.364"/>

                        </Canvas.RenderTransform>

                        <Path xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Name="path4870" Fill="#000000">

                            <Path.Data>

                                <PathGeometry Figures="M0 1432.3 A56.6929 56.6929 0 0 1 113.39 1432.3 A56.6929 56.6929 0 1 1 0 1432.3 Z" FillRule="NonZero"/>

                            </Path.Data>

                        </Path>

                    </Canvas>

                    <Canvas Name="shape63">

                        <Canvas.RenderTransform>

                            <TranslateTransform X="152.766" Y="-815.364"/>

                        </Canvas.RenderTransform>

                        <Path xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Name="path4875" Fill="#000000">

                            <Path.Data>

                                <PathGeometry Figures="M0 1432.3 A56.6929 56.6929 0 0 1 113.39 1432.3 A56.6929 56.6929 0 1 1 0 1432.3 Z" FillRule="NonZero"/>

                            </Path.Data>

                        </Path>

                    </Canvas>

Code snippet of the beginning of the XAML board interface. Canvas is drawn and each board position is its’ own individual canvas – a vector based circle.

## Save Class

Our game allows for the game state to be saved to file / database. The user can load a game if one is present when the application loads and during the game they can enter the options screen and save the game.

## Pseudo Code for Save-Class

Instantiate the FileManager Class **FileManager fm = new FileManager();**

From there, the game would call it like **fm.SaveGame(this);** passing this as a reference to the manager class which can then extract whatever it needs from the game in order to save it. The content of the method within the fm class isn’t important, and it will return a message to say the game has been saved.

To load a game, you could call **fm.LoadGame();** this method could return some data and apply it to the game to load.

**Fm.ClearGame()** could just return void, as its only to clear the game state.