Reversi

Documentation

I. Design

This part will explain how and why the app is designed the way it by going through the different layouts.

This layout shows the status of the game.

If the black rectangle is stroked, it means that it's the black player turn.

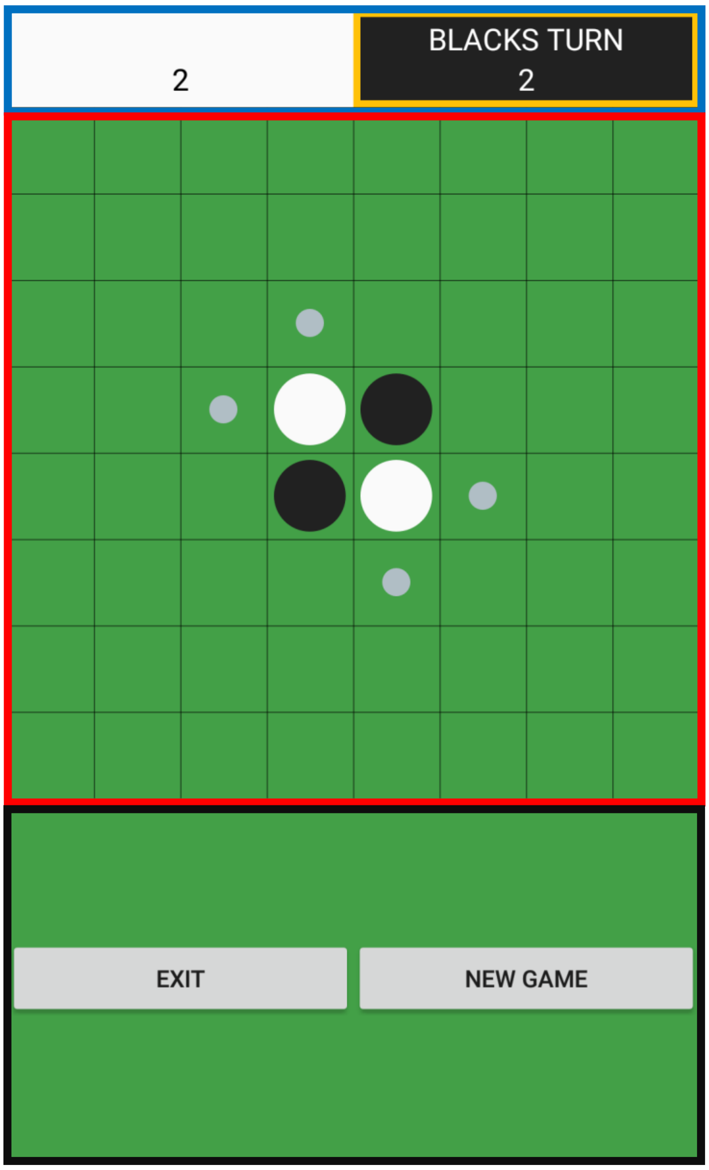
The numbers displayed represent the number of pieces of each player.

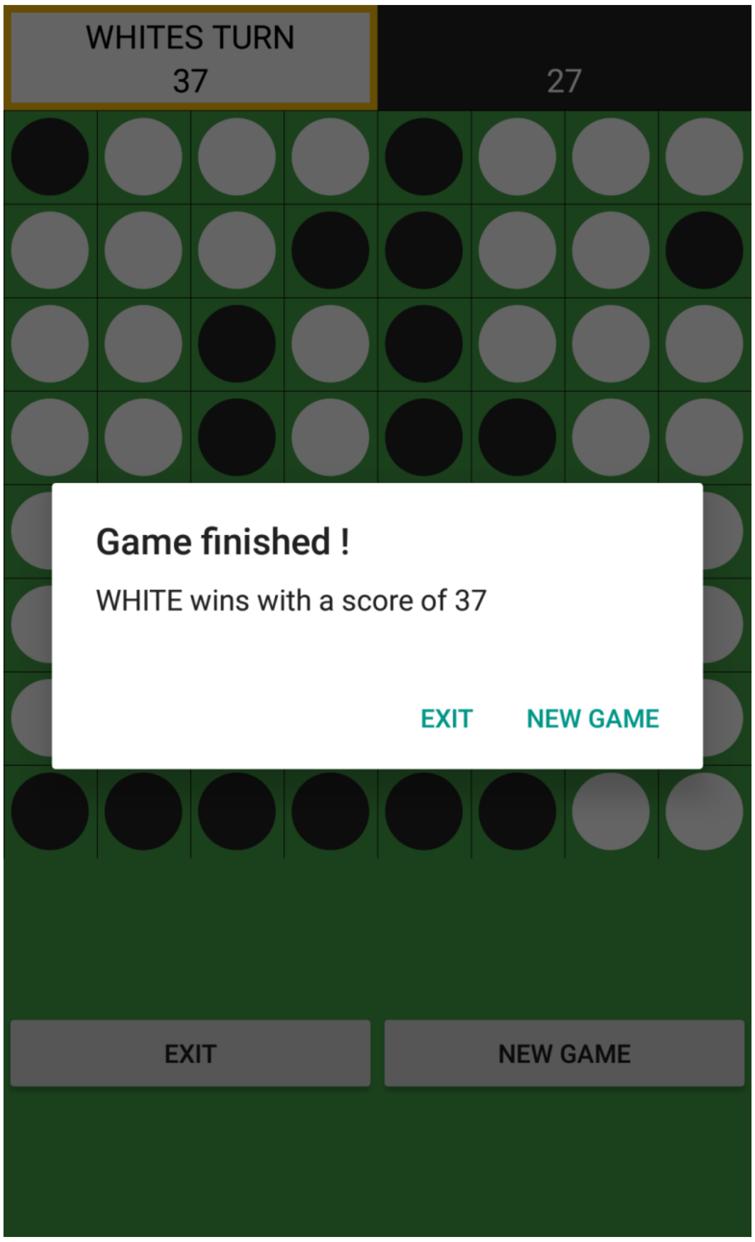
This layout is a custom view that is the actual board game. The custom view will be forced to be square by taking the max width available if it's in PORTRAIT mode and the max height available in LANDSCAPE mode. So a cell will be the size of a side of the custom view divided by 8.

The white circles represent the white player pieces, same goes for the black player. A circle is 0,416 the size of a cell.

The little grey ones represent a cell that is available to put a piece on. A little grey circle is 0,16 the size of a cell.

This layout is the control panel. By clicking "New Game" you reset the game. If you press "Exit", you quit the application.





Game finished !

WHITE wins with a score of 37

When both players can’t play anymore, the game ends.

The winner is the player with the most pieces on the board.

A Dialog appears to announce the winner.

From there, you can either quit the game by pressing “Exit” or start a new one by pressing “New Game”.

II. Code

This part will explain the code.

MainActivity:

initComponents:

This method will initialize all the components contained in the "activity_main.xml" and set the correct listeners.

updateScores:

This method that take an array of integers as parameters will change the text of the TextViews in the "player_board_layout.xml" used to display the score.

changePlayer:

This method will also update the "player_board_layout" by setting the correct background and the stroke for the current player.

Override of "onConfigurationChanged":

This method will be called when the phone's orientation changed. It will set the correct orientation of the LinearLayout that contains the board game and the control panel.

BoardGame (custom view):

init:

This method will initialize the lines that will be drawn to create the board game. It will also create the array of “Paint” that will be used to paint the pieces. It also set the listener for when a grey piece is touched.

reset:

This method will reset the board game as its initial state: 4 pieces on the board and 2 pieces for each player.

checkWin:

This method will check if the game is over with 2 checks:

- if the variable “noMoves” is equal to 2 (which means that both players run out of available move. (it also checks that one player is reduced to 0 piece)
- if the variable “isFilled” is set to “true” (which means that the board is full).

displayWinner:

When the game ends, this method will be called and will display the DialogAlert to inform the players of the winner and offer them the option to quit or to make a new game.

calcDimensionsVertical:

This method is called whenever the phone is in PORTRAIT mode, it will set the cellSize to the viewWidth divided by 8.

calcDimensionsHorizontal:

This method is pretty much the same as the previous but is called when the phone is in LANDSCAPE mode and will set the cellSize to viewHeight divided by 8.

calcPieces:

This method will be called in the onDraw method to calculate the coordinates and the size of the pieces. It will also call the “checkOptions” method.

Override of onDraw:

In this method, the components needed to create the board (the lines and the pieces) are drawn.

Override of onMeasure:

This method will be called whenever a change on the screen size is detected. It will be used to force the custom view to be squared.

GameEngine:

The GameEngine is not an Activity, it's a class used to manage how the game works.

The representation of the board game is a double array of "Piece". I used a double array so it will be easier to check the available moves.

Constructor:

It will set the currentPlayer to BLACK, create the double array and the array of scores.

reset:

It will reset the game by setting the variables used to detect a win to their initial states and reset the double array.

countScores:

It's a method used to know how much of each piece is there on the board.

action:

This is the method that's called when a grey piece is pressed. It will place the piece of the color of the current player and check the options available for the other player.

cleanOptions:

This method is used to replace the grey pieces by EMPTY ones and clear each LinkedList in these pieces.

checkOptions:

This method is used to check which cells can be used to play a move. It will be called recursively on the empty cells to check if a reverse chain is available for the current player. If there is, the LinkedList of an empty cell will be filled and this cell will pass from EMPTY to OPTION, which graphically results in a "grey piece".

Piece:

This class is not an Activity or a View. It's just a Model class that represents a piece on the game board. It has some attributes that are usefull for both GameEngine and BoardGame like the coordinates of the piece (in pixel) and the type of the piece.

It also stores an array of LinkedList that is only used when a Piece is of the type "OPTION". Each LinkedList represents a reverse chain in a direction.