**Set up design document**

**Krijn Hoogendorp 8724970**

**November 14, 2014**

Goal

Create a two-dimensional puzzle with one empty space into which some numbered tiles can slide horizontally or vertically to occupy. The goal is to arrange the board from smallest to largest, as shown in the example below.

Requirements

1. N-puzzle
2. 3 levels of difficulty: “easy” (n=8), “medium” (n=15), “hard”(n=24)
3. Users must have choice of images
4. Image must appear for 3 seconds
5. During play user is not allowed to hit MENUbutton
6. When changing difficulty a new game should start
7. The game´s state should survive after user quits

Optional

1. Ensure pseudorandom shuffle after start/resest game
2. Allow user to use own image (collect through URL)
3. Implement automatic solver

Implementation

The implementation has 2 main parts:

1. Puzzle layout
2. Puzzle rules

The layout will be done partly in Java and partly in XML

The rules will be coded in Java

Screens

main

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  | start game |  |  |
|  |  |  |  |  |
|  |  | continue previous game |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Start: screen 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  | Choose level |  |  |
|  |  |  |  |  |
|  |  | easy |  |  |
|  |  |  |  |  |
|  |  | medium |  |  |
|  |  |  |  |  |
|  |  | hard |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Start: screen 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  | Choose picture |  |  |
|  |  |  |  |  |
|  |  | UvA logo |  |  |
|  |  |  |  |  |
|  |  | football |  |  |
|  |  |  |  |  |
|  |  | pretty girls |  |  |
|  |  |  |  |  |
|  |  | import own |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Start: screen 3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  | PICTURE | | |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  | quit |  | reset |  | solve |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Activities

mainActivity: onCreate (create First screen), onClick (when choosing an option)

playLevelActivity: onCreate (create screen 2), onClick (When choosing an option )

playPictureActivity: onCreate(create screen 3), onClick (When choosing an option)

startGameActivity: onCreate, onClick, onQuit (touching quit button), onReset (touching reset button), onSolve (touching solve button)

statusGameActivity: onCreate, onClick, onReset, onSolve

automaticGameActivity

Classes

MainActivity, PlayLevelActivity, PlayPictureActivity, PlayGameActivity, PlayGameRules, PlayGameStart

Data

The data important is:

Difficulty, picture, game state

At the moment I am exploring the following option to collect the game´s state:

- assigning a variable to each of the parts of the picture (picture has n parts).

- assigning a variable to each of the possible locations.

The games state is saved by the tupple of these 2 variables.