TILES – ANDROID 2D GAME

Software Engineering Individual Project – CS 7059

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science & Statistics (SCSS)

JANUARY 16, 2017

SUBMITTED BY – SAURABH PATHAK (16336962)
Under Supervision of – Dr. Fergal Shevlin

Table of Contents

Abstract

Tiles is a 2D android game developed using android SDK, Java and Android Studio. The game is supported on android versions > 6.0 (Marshmallow). It has been tested on both android 6 and android 7 (Nougat) versions. The game has been designed using Adobe Illustrator.

My background

I am an Information Technology graduate. I didn't have any experience with android development until this project, but I had some software development experience. So I was able to relate the android framework and work on it. It took me 3 weeks to get hold of the framework.

1) Introduction

Tiles is a memory game where the player needs find the pairs of tiles with same number or pattern behind them. The game contains a grid of tiles. Each tile has a placeholder on the front side which is visible initially to the player and a number or pattern behind them. Each of these unique number or pattern exists twice in the game. So the motive is to successively tap on the tiles with the same number or pattern behind them. When a player taps a tile it flips and shows the number behind it. If the second click is made on a tile with same number or pattern behind them, then the player has successfully found a pair. Both the tiles freeze and will constantly show the number as a placeholder. In case the second click results in a tile with different number or pattern behind it, both the tiles flip back and hide the pattern.

2) Functionality (Approach)

I will now discuss my approach to implement the above functionalities. Since I only used the standard android SDK, I only needed to learn about its API documentation and little bit about JAVA. My approach was fairly simple. I first implemented the game functionality in a test web app as I have some proficiency with the technology, so it was easier to identify the issues and getting an easy solution for all of them. When the game became completely functional I froze my design there and made the same implementation in android.

2.1 Welcome Screen

The activity associated with the welcome screen is *IndexActivity***.** This screen has 3 buttons in total – *Start Game, Resume Game & Best Scores.*

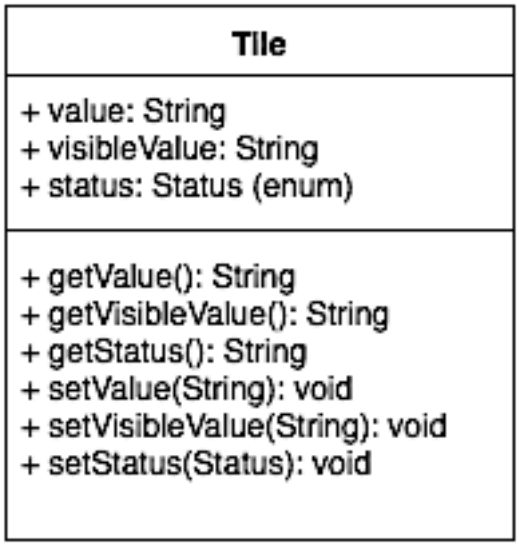
For *Start Game*, the functionality was straightforward. I just needed to initialize a new tiles grid (*explained in 2.2*), a stopwatch to record the time and a click counter.

For *Resume Game*, I needed to implement some data structure that records my game's last state and makes the new grid from recorded data. I used Shared Preferences to implement this button's functionality. When this IndexActivity loads, it checks for a key "tileList" stored in Shared Preferences which contains the game state. In case "tileList" is found, I show the Resume Game button and load the previous game, else set its visibility to GONE.

For *Best Scores*, I simply redirected welcome screen to best scores screen. The implementation for saving and getting scores is *explained in 2.4*.

2.2 The Tiles Grid

On starting or Resuming a game, app lands to the *MainActivity* page. On this main game page we needed a data structure to represent each of the tiles. For this a **Tile** class was created for which the class diagram is shown on the right side. Now let's go through what each property of the Tile class represents.



2.2.1 value: The value of a tile which is hidden and needs to be matched.

2.2.2 visibleValue: The placeholder value on the tile behind which the value is hidden.

2.2.3 status: The status of a tile. Its and enum with 3 values (*locked, visible, unlocked*). Until the values are matched a tile's status is locked. Once a tile's pair is found and matched its status is changed to unlocked.

All the methods in Tile class are just getters and setters. Moving on, for the Tiles grid an ArrayList of such Tile objects was made which stored the game status. Now we had the basic data structure for our grid and we are left with implementing the UI for the grid. For the UI view and interactions, I created an adapter, ButtonAdapter which extended the android native

BaseAdapter. Now we have the UI as well as the data stored for a grid.

2.3 Saving the Game State

Saving the game state was a very tricky problem. First challenge was to decide which option to use for saving the grid data. I finally chose the option of saving data as key-value pairs in shared preferences.
The reason for choosing shared preferences over SQLite was that I didn't wanted to invest much time and effort learning about database management. Shared preferences offers a very simple API to save data. It is very well documented on the Android developer website and it only took half an hour to implement and test it.
The next challenge was saving the arraylist in shared preferences. Shared preferences only offers saving data in primitive data types and Set<String> type structures. So to tackle this problem I made a utility function to convert my arraylist to JSONArray type object which may further be parsed to string and consequently be saved using shared preferences. Following is the parsing code.

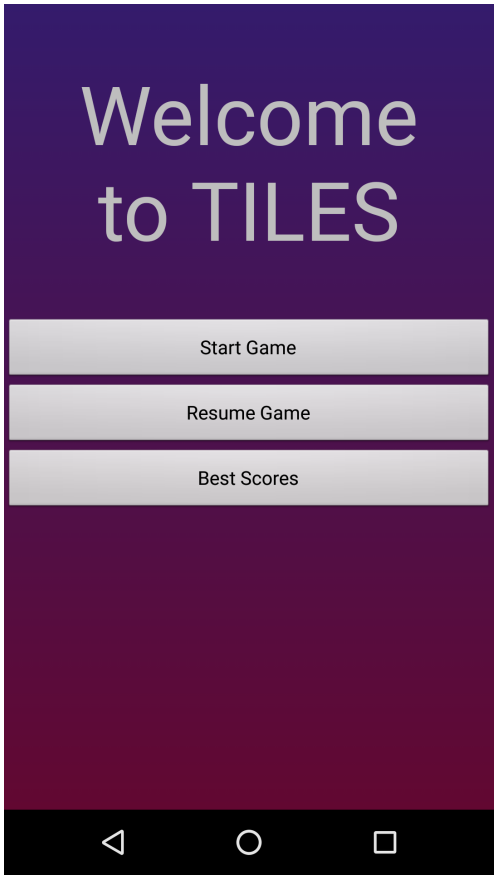
```
public static JSONArray
getJsonListFromArrayList(ArrayList<Tile> tileList) throws
JSONException {
    JSONArray list = new JSONArray();
    JSONObject obj;
    for (int i = 0; i < tileList.size(); i++) {
        obj = new JSONObject();
        obj.put("value", tileList.get(i).getValue() == null
? "" : tileList.get(i).getValue());
        obj.put("visibleValue",
tileList.get(i).getVisibleValue() == null ? "" :
tileList.get(i).getVisibleValue());
        obj.put("status", tileList.get(i).getStatus());
        list.put(obj);
    }
    return list;
}
```

2.4 The Scoring Screen

Implementing the scoring screen was the easiest module of the project. In shared preferences I kept another key for saving the highest scores – "scores". Every time a game finishes, I just add another score to scores array, sort it and remove the 4th element from the array since I am only showing top 3 scores.

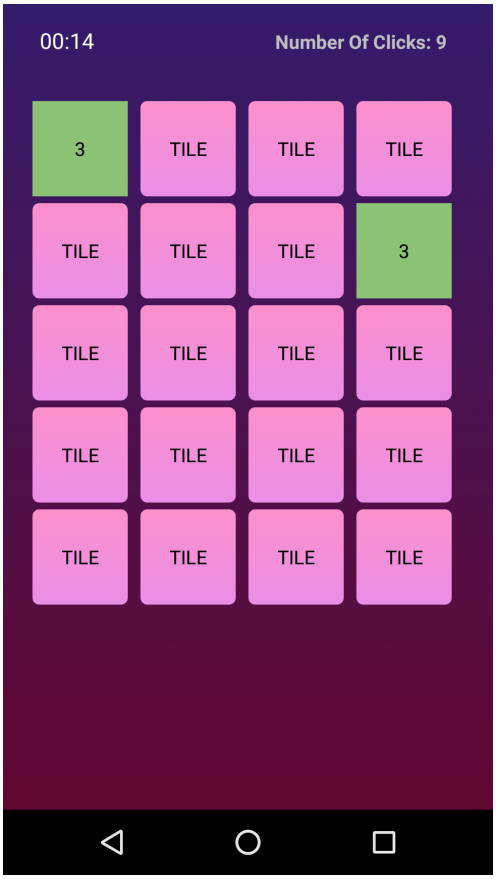
3) Appearance and Interaction

Designing the UI was the main challenge since I don't have any designing background. Still I looked up a few tutorials and made use of Adobe Illustrator and designed the basic theme for the app. An icon was also designed for the app which is available in the coming section.

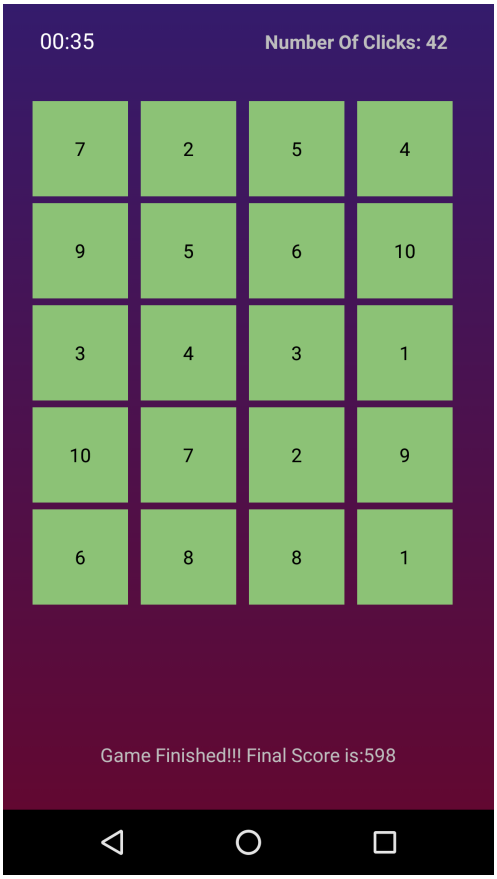


This is the **Game Home Screen**. The player is presented with three options – Start Game, Resume Game, Best Scores.

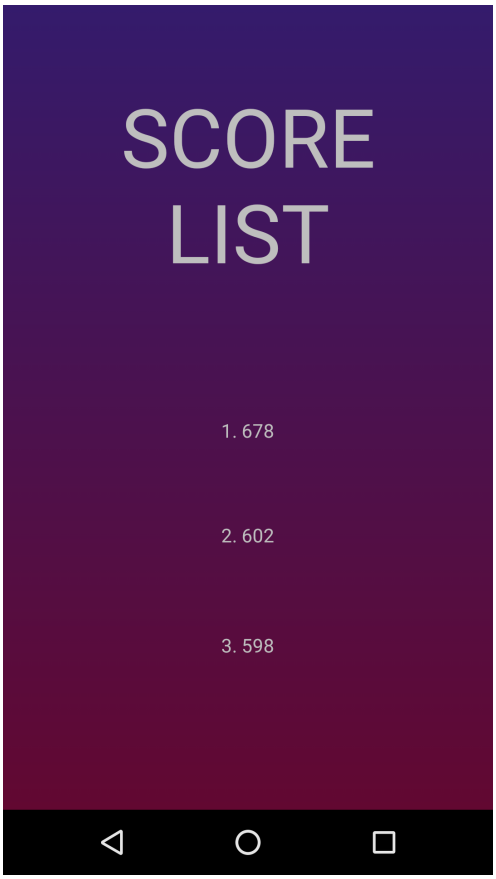
This is the **Game Start Screen**. It appears on tapping Start Game Button on the home screen.



This is the **Game Resume Screen**. It appears when a player taps on the Resume Game button on the home screen. The last saved Game will be resumed from where it was left.



This is the **Game Completion Screen**. Total time taken to complete the game is shown on top left, number of clicks is on top right and at the bottom of the screen, final score is displayed.



This is the list of best scores screen. Top 3 scores achieved on a device are shown here.

4) Testing

Similar to designing, I don't have any background in testing. I implemented JUNIT for testing individual units to verify whether functionality is stable to extend or not.
Following were the JUNIT test cases that were implemented.

4.1) IsListUnlocked – Game Finished trigger

```
@Test
public void isListUnlocked() {
    assertNotEquals(Utils.isListUnlocked(tileList), true);
}
```

4.2) IsListLocked – If any one tile is unlocked, save the game

```
@Test
public void isListLocked() {
    assertEquals(Utils.isListLocked(tileList), true);
}
```

4.3) IsArrayMatching – Whether the grid generation function is generating correct pattern or not

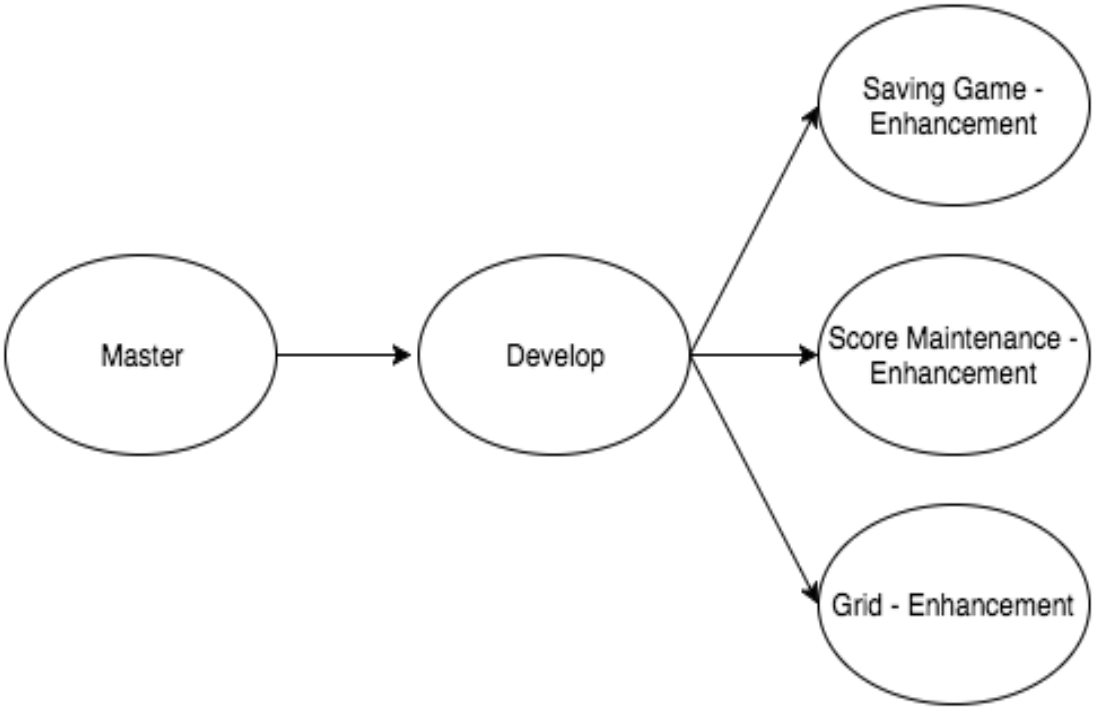
```
@Test
public void isArrayMatching() throws Exception {
    assertArrayEquals(arrayOfInts, arrayOfInts2);
}
```

4.4) IsJsonValid – Whether json is valid and lint properly

```
@Test
public void isJsonValid() throws JSONException {
    assertEquals(Utils.getJsonListFromArrayList(tileList),
jsonList);
}
```

5) Management

I used gitlab for maintaining and versioning my code base. In total there were **35** commits. I made enhancement specific branches. All these branches were made from a develop branch, which is the main development branch. The develop branch was originated from the master branch, which is the main release branch. Following diagram represents the branching structure for my repository.



6) Planning and Documentation

I used AGILE development for the development and maintenance of my android game. In total there were 5 sprints -

SPRINT 1 – Deciding a Game Topic

- Research on previous built games
- Coming up with game ideas
- Finalizing the Game
- Week 4 – Presentation
- Understanding Technical Requirements
- Learning about popular Game building Frameworks
- Finalizing technical tools which will be used

SPRINT 2 – Setting up the Project

- Install Java
- Installing Android Studio
- Creating a new project with blank activity
- Setting up Virtual device
- Configuring drivers for connected android device
- Running the project

SPRINT 3 – Development

- Create Game Page
- Create Grids

- Add Timer
- Add Click Count functionality
- Add Animation on clicked tiles
- Implement Corresponding click functionality
- Create Start Page
- Calculate and show scores
- Retain game status on close
- User customized grid option
- UI Improvements

SPRINT 4 – QA and Stabilization

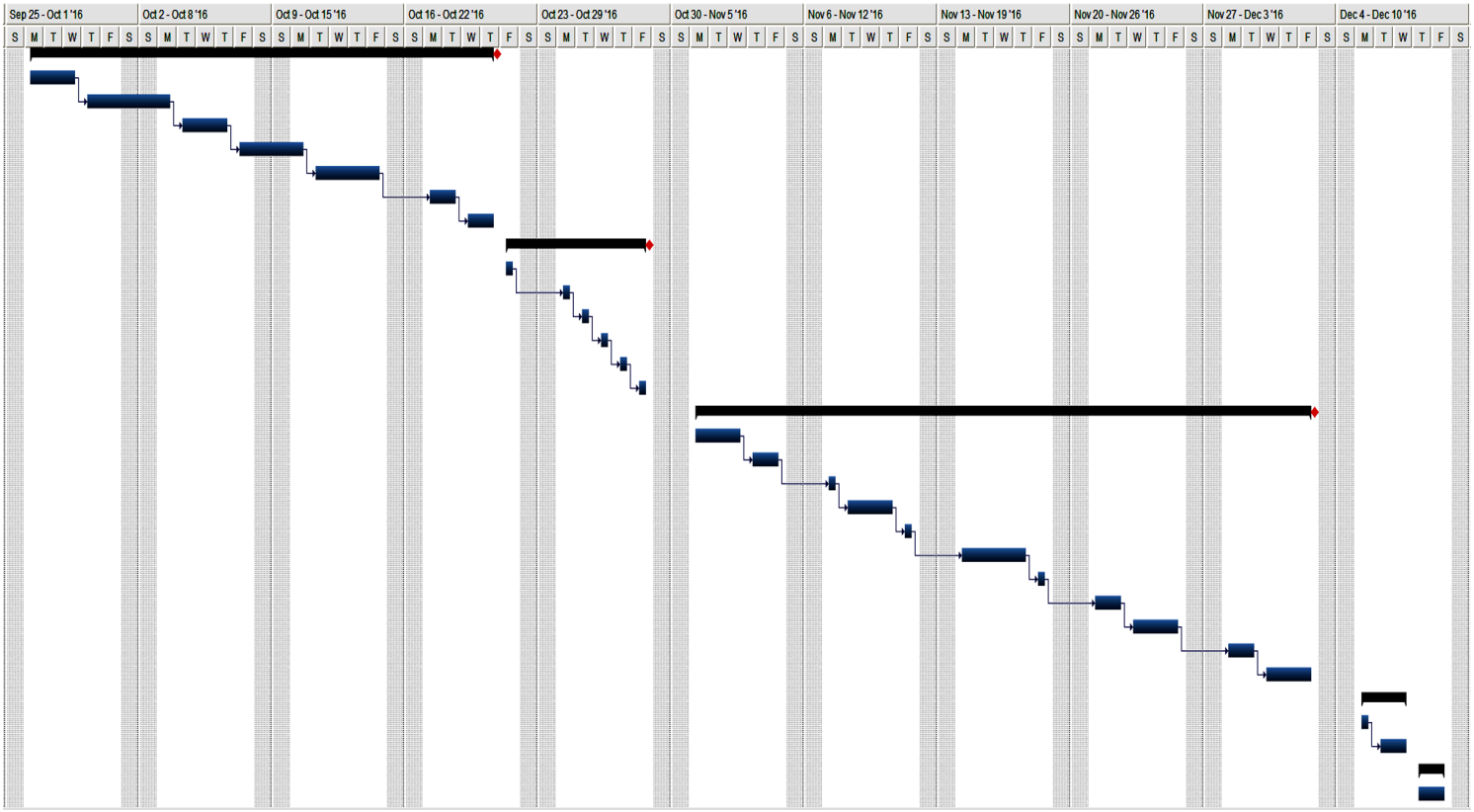
- QA and bug logging
- Stabilization - Fixing Issues

SPRINT 5 – Deployment

- Deploy on App store

For keeping a track on the progress of the game I used Gantt chart to schedule the sprints and their respective enhancements.

(Figure 2: Gantt Chart showing progress throughout the Project)



7) Further Work

The basic model of the game has been implemented. There is still scope for extension in the game. We may use some 3D graphics library to enhance the experience to 3D, something like the mahjong tiles.
Apart from this, the game can still extend in the levels of difficulty. For now we only have 3 levels of difficulty. So there can be further levels with increasing difficulty of patterns and complicated grids.

8) Conclusion

Developing an Android game was real fun. Development in android is not very complex, so it was the right choice for a 12-week project. I, myself, have worked as a software engineer for 3 years and android really fascinated me. Learning android was really fun and this is definitely going on the skills listed in my resume.

9) References

None of the contents in this report has been taken from anywhere. All the images and textual content have been designed and written by me.