University of Edinburgh

Software Engineering Large Pracitcal

Part 2: Grabble Design Document

Author:
s1317642 Kendeas Theofanous

November 9, 2016

[Document "Design-Document.pdf" submitted by s1317642]

[This page intentionally left blank]

1 Introduction

Grabble is a mobile game implemented to allow users to collect letters from around the University of Edinburgh's Central Area and make words. The features as well as the aspects of the system were discussed in the Proposal Document. Android Studio allows to create applications targeting many devices. My implementation of Grabble will target phones and tablets with minimum SDK Android 4.4. Doing so, Grabble can be run on approximately 73.9% of the devices active on the Google Play Store. Grabble will extensively use GPS location of the device it runs on, consuming battery of the device quickly. Also, its features will potentially use the devices memory. Android 4.4 with its improved memory usage will benefit the users' gameplay experience, reduce lagging and allow the user to use other features of their device more smoothly. In the following sections, I will discuss the structure of my implementation of Grabble, which comprise the software elements externally visible to the users and the relationships among them. I will also discuss the bonus features that I will include in my implementation.

Nice thought. To optimize for power, think when do you need to turn GPS on? There are many granularities of location available in Android library. They have different power requirements too.

2 Software Architecture

2.1 Activities

The application will consist of four activities which correspond to the four different layouts the application will run on.

2.1.1 MainActivity

MainActivity is the launcher activity of Grabble. It is basically a simple login page. The user will have to log in with their unique username and password. The application checks whether the android has access to Wifi or mobile data and if it doesn't, it prompts the user to connect to a network. The application will insist to ask for a network connection, since the KML files and dictionary are not stored locally but are accessed through a URL. Once a connection is confirmed, the user can follow to click the login button. When the login button is clicked, the user is transferred to a splash screen until the map loads. In my further implementation of the application, I plan to modify the MainActivity in a way such that when the user opens the application for the first time, they will have to register and the application will store locally their credentials and only accept those as login information. i.e The application will only be used by one user.

(a) View: Layout (b) Prompt to turn on Wifi (c) Screenshot

Figure 1: MainActivity

Would it work on mobile data?

[Document "Design-Document.pdf" submitted by s1317642]

Suggestion: Look for Google Play Games Services APIs for sign in feature. (auth)

2.1.2 SplashActivity

SplashActivity is a splash screen I implemented to load after clicking the login button on the MainActivity in preparation of the maps loading in the next activity. The layout includes a text section where a tip is displayed. I have also included an Instructions button where a user can click and a dialogue will display further instructions and rules of the game. A small image was selected to be displayed in the middle of the screen. A progress bar is shown below the image, indicating the loading of the map. Once the map is ready to be displayed, the application goes to the MapActivity.

✓decorative feature. But a nice feature. Most professional apps include "loading screens" in their product.

(a) View: Layout

(b) Screenshot

Figure 2: SplashActivity

2.1.3 MapActivity

Very nice detailed thoughts.

✓

MapActivity is the activity where the map is loaded and the KML layer is imported for the Central Area of the University of Edinburgh. The KML layer loaded in gameplay is determined by the day of the week when the application is started. The application asks for permission to access the calendar of the users device in order to load the map. If the user clicks yes, the application goes to its next functionalities, otherwise it prompts again.

Suggestion: You can do away with this permission by taking server time (as you are connecting to it all the time) and take user's timezone ~~info~~ information. But this is not critical.

✓

The application checks whether the user has the location settings of their device turned on and if they are off, the application asks for permission to turn them on. If the user clicks yes, they can proceed and play the game. Otherwise, the application can't recognise the current location of the user and acts as a simple Google Maps view.

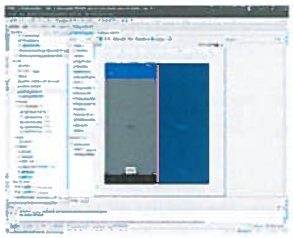
This activity only contains the support map fragment for Google Maps and an Options button which when clicked, it transfers the user to the CollectionOfLettersActivity for more options.

Suggestion for future extensions: When there are too many placemarkers (>> 1000) then how do you efficiently find out which one is near? Do you check distance with every placemarker? - How do you store the placemarkers in memory?

✓

The Android LocationManager API of the application will determine how close the player physically is to a certain Placemark and once they are at a range 10m from the Placemark, it will indicate they are able to grab the letter. The indication I chose for this situation is a single vibration of the phone. Once the user grabs the letter, the Placemark will change colour.

I used code from `http://www.androidtutorialpoint.com/intermediate/android-map-app-showing-current-location-android/`, for the implementation of Google Maps.



(a) View: Layout

(b) Screenshot

(c) Prompt to turn on Location

(d) Kml Layer on Centra Area

Figure 3: MapActivity

2.1.4 CollectionOfLettersActivity

CollectionOfLettersActivity is displayed when the user clicks the Options button in the MapActivity. Its implementation is incomplete but a draft design is displayed below. It contains a button Letters, which when clicked displays a dialogue with the letters the user has collected so far during his game time on a certain day. It also contains a button History, which when clicked displays a dialogue with the words the user has formed since the day they started playing the game in descending order of the words' score. Finally, it contains a button Dictionary, which when clicked displays a dialogue with all the seven-letter sequences of characters forming words as specified by the *Official Grabble Dictionary 2016*.

Not very clear!

Showing all words from the dictionary is not a good idea. Think why?

(a) View: Layout

(b) Draft Screenshot

Figure 4: CollectionOfLettersActivity

2.2 Bonus Features

The main Bonus Feature of the application is that it will keep a history of the words the player has created and their score. The words are stored in descending

order, such that the highest-score word is first. Users who like the challenge will want to surpass their previous scores and aim to create words with higher scores.

I have implemented a MusicService Java class and music is played throughout the activities of the application. The song I chose is *Rise of the Machines* by *West One Music*.

When the player enters the range of a Placemark, the device will vibrate once, notifying the player they can grab a letter.

Finally, when the player grabs a letter, the Placemark will change colour (or disappear) from the map indicating that the player can't go to the same Placemark and grab the same letter on the same day map.

2.3 Libraries

In the implementation of Grabble, despite the android libraries used for widgets, buttons and other UI objects, I will use libraries for specific Google Maps funcitonalities and network accesses. I will also use media player libraries for addition of music in my activities. Other useful libraries are animation libraries for smoother image animations. I plan to use libraries to store information for the letters and words the user will form during their gameplay. The libraries I have so far are the following:

- android.content.ComponentName;
- android.content.Context;
- android.content.DialogInterface;
- android.content.ServiceConnection;
- android.net.ConnectivityManager;
- android.net.NetworkInfo;
- android.os.IBinder;
- android.provider.Settings;
- android.support.v7.app.AlertDialog;
- android.location.Location;
- android.os.Build;
- android.os.StrictMode;
- android.support.v4.app.ActivityCompat;
- android.support.v4.app.FragmentActivity;
- android.os.Bundle;
- android.support.v4.content.ContextCompat;
- com.google.android.gms.common.ConnectionResult;

- com.google.android.gms.common.api.GoogleApiClient;
- com.google.android.gms.common.api.PendingResult;
- com.google.android.gms.common.api.ResultCallback;
- com.google.android.gms.common.api.Status;
- com.google.android.gms.location.LocationListener;
- com.google.android.gms.location.LocationRequest;
- com.google.android.gms.location.LocationServices;
- com.google.android.gms.location.LocationSettingsRequest;
- com.google.android.gms.location.LocationSettingsResult;
- com.google.android.gms.location.LocationSettingsStates;
- com.google.android.gms.location.LocationSettingsStatusCodes;
- com.google.android.gms.maps.CameraUpdateFactory;
- com.google.android.gms.maps.GoogleMap;
- com.google.android.gms.maps.OnMapReadyCallback;
- com.google.android.gms.maps.SupportMapFragment;
- com.google.android.gms.maps.model.LatLng;
- com.google.android.gms.maps.model.Marker;
- com.google.android.gms.maps.model.MarkerOptions;
- com.google.maps.android.kml.KmlContainer;
- com.google.maps.android.kml.KmlLayer;
- com.google.maps.android.kml.KmlPlacemark;
- org.xmlpull.v1.XmlPullParserException;
- java.io.IOException;
- java.io.InputStream;
- java.net.URL;
- android.os.Bundle;
- android.os.Handler;
- android.animation.ObjectAnimator;
- android.view.animation.LinearInterpolator;
- android.widget.ProgressBar;
- android.media.MediaPlayer;
- android.media.MediaPlayer.OnErrorListener;

2.4 Dependencies

The following dependencies are vital for the operation of Grabble:

- 'com.android.support:appcompat-v7:24.2.1'
- 'com.google.android.gms:play-services-maps:9.6.1'
- 'com.android.support:design:24.2.1'
- 'com.google.android.gms:play-services-appindexing:9.6.1'
- 'com.android.support:support-v4:24.2.1'
- 'com.google.android.gms:play-services:9.6.1'
- 'com.google.maps.android:android-maps-utils:0.4+'

Part 2 of the Software Engineering Large Practical has been marked in accordance with the marking criteria given in the specification document for the practical, as detailed below.

---

1. The coverage of the activities which the app will include.

- *Have any major application functions been forgotten?*
- *Is each activity a coherent, self-contained item of work which needs to be done?*

20/25

You didn't provide detailed explanation of
- How does a user form a word?
- How do you validate a word?

---

2. The completeness of the libraries and dependencies which were listed.

- *Do they include everything which will be needed to implement the bonus features which are promised?*

25/25

---

3. The added value provided by the bonus features which are offered.

- *Are they just some additional decorations or do they actually make the game more interesting to play, or more rewarding?*

25/25

---

4. The coherency and consistency of the screens which you shown from the application.

- *Do they clearly belong to the same app, giving the impression of an app which has been designed thoughtfully?*
- *Do they represent aesthetically-pleasing design?*

22/25

Better screenshots for word formation and letter inventory was ~~expected~~ expected.

Mark awarded: 92 %

[This page intentionally left blank]