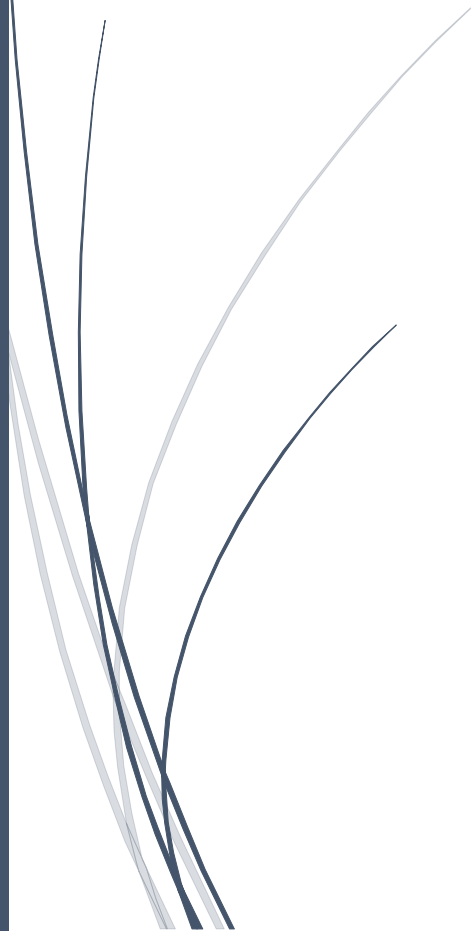
12/11/2017

Documentation

Software Engineering Large Practical – Part 3



Finn Zhan Chen
UNIVERSITY OF EDINBURGH

Contents

1 Additional Bonus Feature

There were some additional bonus features which were not described in my design document.

1.1 Play Anywhere!

The user can choose to play the game in his current location. This bonus feature makes use of the relative coordinate of the placemarks in George Square to the user's location.

1.2 Play in Edinburgh, New York, London, Canary Island and Los Angeles!

The user can choose to play the game in a predefined set of streets (George Square, Wall Street, Imperial College, Las Palmas and Hollywood). This bonus feature makes use of the relative coordinate of the placemarks in George Square to the street's coordinate.

2 Bonus Features not realised

There were parts of my design which have not been realised in the implementation.

2.1 Drawing a red rectangle

With the additional bonus features from previous section, drawing a red rectangle for the game map adds redundant complexity and does not add any additional value for the user. Therefore, I decided to implement it.

3. Acknowledgement

3.1 Code Reuse

- *AlertDialog askOption()* from StackOverflow: https://stackoverflow.com/questions/24359667/how-to-disable-back-button-for-particular-activity

- Navigation Drawer template from Android Developer: https://developer.android.com/training/implementing-navigation/nav-drawer.html

- Class Connectivity.java by Emil from StackOverflow: https://stackoverflow.com/questions/2802472/detect-network-connection-type-on-android

- Count Down Timer Template from Android Developer: https://developer.android.com/reference/android/os/CountDownTimer.html

3.2 Other

- Logo has been generated and customised from a free logo generator Logaster: https://www.logaster.co.uk/

- The source of the placemark collecting sound is royalty free from ZapSplat. https://www.zapsplat.com/music/game-win-or-gain-stars-x-3-end-of-level-tone/

4 Storage

4.1 SharedPreferences

SharedPreferences “mysettings” stores the following variable:

String:

- user_name: This personalises the game to the user

Integer

- superpower_remaining: This is the accumulated superpower from past victories
- guess_remaining: This is the accumulated guesses from past victories

Boolean

- map_exists: this determines whether the user can play the game. When installed map_exists is set to false and map_exists is set to true after the successful completion of the first download task.

Set<String>

- Completed_songs: Each string in this set is the combined formatted information of a cleared song. The information includes date, where, user name, difficulty, song details, and YouTube URL. This format is decoded when outputting to the user in Activity_8_Achievement.

4.2 Internal Storage

The internal storage stores all the following files:

- 5 maps KML file for each song in the formatted name “Song01-Map1”
- 1 lyrics TXT file for each song in the formatted name “Lyrics01”
- 1 database of songs XML file in the name “SongsXML”

5 Algorithms and Data structures

Algorithms and data structures for the core functions of the implementation are documented.

5.0 Self-Defined Class

A *Song* object contains 4 String attributes: song number, title, artist, and YouTube link.

A *Placemark* object contains 3 String attributes (position, description, styleUrl) and 1 LatLng attribute (point).

Both class implements Serializable interface so that the objects can be passed between activities.

5.1 Activity_1_Start

5.1.a Download All Song, Maps and Lyrics XML to internal storage

At the start of the starting activity, if the WiFi or 4G is available then executes *DownloadEverythingTask.java*.

1. If songs.xml does not exist in the internal storage or version is different:

   i. Download all files from Informatics server
   ii. *List<Song>* is used to store all songs parsed from the online songs.xml because the exact number of available songs is not known.
   iii. 5 maps and lyrics of each song in *List<Song>* downloaded and saved to internal storage with indefinable filenames such as “Song01-Map1” and “Lyrics01”.
   iv. Save the live songs.xml internally in the end. This makes sure if download is interrupted, user needs to download everything again and not let user play the game.
   v. Set map_exists to true in SharedPreferences mysettings.

2. If 1 is not the case then if map_exists is true, then allow user to play the game otherwise output an error message.

5.2 Activity_2_Choose_Song

5.2.a Passing the chosen Song object in Activity_2_Choose_Song to Activity_3_Game

HashMap<String, Song> songMap is used to map the chosen song number from User Interface to the actual Song object. Song object can retrieved at O(1) time and is passed to Activity_3_Game as a Serializable object.

5.2.b Updating UI with available songs

songMap is passed by reference and is initialised in LoadSongFromFileTask.java by parsing the songs.xml from internal storage. All available songs for user to choose is the *songMap’s* keys and is updated on the Spinner object in the UI.

5.3 Activity_3_Game

This activity contains the bulk of the app and contains 3 important data structures:

- *HashMap<String, String[]> lyrics*: The key is the line number of the lyrics txt file and the value is the words corresponding to that line.

- *ArrayList<Placemark> collectedPlacemarks*: The list of placemarks collected.

- *ConcurrentHashMap<Marker, Placemark> markerMap*: The key is a Marker reference so that it can be removed from map when collected. The value is the Placemark and is used to calculate the distance between user and the placemark.

- *LatLng gameStartPosition*: This determines where the placemarks are plotted in relative to the original coordinates (i.e. this is the centre of all Placemark). This is crucial for the additional bonus feature in section 1. *ConcurrentHashMap* is important because it allows concurrent modification of the Map during the iteration loop without the need to block them.

5.3.a Load Song Lyrics

Hasmap<String, String[]> lyrics is initialised in task *LoadLyricsFromFileTask.java*.

The lyric text file for the corresponding song is retrieved from internal storage. Each line from the text file is read and split into line number and words corresponding to that line. These information is put into *HashMap lyrics*.

5.3.b Load Placemarks On Map

Depending on where the user wants to play (User's current location, George Square, Imperial College, Wall Street, Las Palmas or Hollywood), *LatLng gameStartPosition* is set to the corresponding coordinate.

The actual plotting takes place in *LoadPlacemarksFromFileTask.java.*

Depending on the difficulty chosen, 1 of the 5 maps KML from internal storage is parsed to obtain *List<Placemark> placemarks.*

For each Placemark in *placemarks*:

1. Modify Placemark's coordinate in relative to the *gameStartPosition*.
2. Plot a Marker to the map with the Placemark's coordinate and style.
3. Put the Marker as the key and the Placemark as the value in the HashMap *markerMap*. This is needed to unplot *Markers* on map when collecting them.

5.3.c Automatic Placemarks Capture Within Range

On every location update, run *capturePlacemarksWithinRange* function which collects placemarks within range. The capture range is set according to the difficulty chosen.

For each Marker in the *ConcurrentHashMap markerMap* keyset do:

1. If the distance in metres between the user's location and the Placemark is smaller than the capture radius then:
    a. If the game is not muted, play a pickup sound once only in the entire iteration
    b. Add the placemark to *ArrayList<Placemark> collectedPlacemarks*
    c. Un-plot *Marker* from map
    d. Remove this key-value set in *markerMap*. This is only achievable because of *ConcurrentHashMap* data structure.

5.3.d View Collected Words

To view the collected words, the following are done:

1. Create *LinkedHashSet<String>* for each category of words.
2. For each Placemark in *collectedPlacemarks* do:
    a. Decode Placemark's position into line number and word index.
    b. Use *HashMap lyrics* to get the word at that line and index.
    c. Append the word to the *LinkedHashSet<String>* of the corresponding category.
3. Pass these *LinkedHashSet as ArrayList<String>* in the new intent to *Activity_4_View_Collected_Words* to display the words in screen. *ArrayList<String>* is necessary because intent does not support passing *LinkedHashSet* to the new activity.

HashSet is used to avoid repetition of words collected and *LinkedHashSet* to preserve the order in which words are collected.

5.3.e Superpower concept

This takes place in method *activateSuperpower*.

If the superpower is not active, then:

1. Decrement remaining superpower
2. Save a copy of *markerMap* to *markerMapOld*
3. Clear the map
4. Load Placemarks on Map (see section 5.3.d) with a difficulty one level lower than the selected difficulty.
5. Activate and set the countdown timer to 60 seconds.
6. During this 60 seconds period, game is played like normal.
7. When timer finishes, set *markerMap* to *markerMapOld*
8. Clear the map and reloads the placemarks from *markerMap* in *ReloadPlacemarksToMapTask*