Computer Science Large Practical

Part 3

Submitted by:	**s1504942**
Submitted on:	Fri Dec-15-2017 at 14:17:16
Coursework deadline:	Fri Dec-15-2017 at 16:00:00
This document produced:	Fri Apr-06-2018 at 00:55:43

1 Introduction

This document contains a copy of your Part 3 report for the Computer Science Large Practical.

— ◇ —

Your submitted app has been tested; feedback and the mark awarded are presented in Section 2 at the end of this document.

Songle- Completion Report

s1504942

Summary

All features promised in the Design report submitted in Part 2 of the assignment have been implemented. Obvious improvements to the layout have been made at some places and some features are extended to improve user friendliness.

Points to Notice

- **Intro Activity is added** – When the user opens the app for the very first time, he is welcomed by an Introduction activity. Permission for Location is also added here - at the slide where Maps are explained. **(Addition)**
- **Visual Appeal** – Apart from making sure that general visual appearance is pleasing throughout the app, Guess activity's layout has been particularly improved aesthetically.
- **Feedback** – Feedback is an important part of a good design. Throughout the app, feedback is provided to user. For example – feedback when/with,
    - there are no words collected yet (in guess activity)
    - map level is upped
    - word is collected
    - data is successfully reset from settings + confirmation for deletion
    - Colouring of Song List based on which puzzles are unlocked (they follow a scheme)
    - a puzzle is guessed correctly – Feedback that more puzzles may be available now etc.
- **Thorough Testing** – Testing has been done not only to test the features and cases (like no internet) but also to see where improvements can be made. Surely, this will benefit the app experience. Consider –
    - Testing with device suggested that user must have put in a lot of work in collecting the words and must be given a chance to give up (explained below, how).
    - Map Updating to Levels has been tested
    - Unlocking of new Puzzles tested
    - Testing with device showed that there is initial noise when the map first opens, this incremented distance travelled without moving. Code added to avoid this
    - And more throughout the period of development (orientation, resetting progress properly, etc)
- **Small additions**
    - The progress bar in MapsActivity now also shows map level with words-collected to total-words-in-song ratio. Tapping the progress bar centres the map to playing area (George Square) (User is made aware of this in the intro activity).
    - Solved puzzles show song title at review screen (with 100% complete) and guess-field is filled with answer in Guess Activity for guessed puzzles.

Slight change in a feature + Addition of a feature

Hint was given by showing a very interesting word from Map-5 so that user could concentrate on it to guess the song. However, not all songs have very interesting words. So now, Hint button is only visible for Songs which have very-interesting words.

(Addition) Even though the design document said users can't give-up (& get free answers) and explained reasons for that decision, players can now give up once they have reached level 4 on Map. The give-up button becomes visible on Guess Activity if user is on map level 4 for that song. This is because by map 4, user has spent significant time on the puzzle and will be unhappy if he fails to know the correct answer.

Algorithms and Data Structures

Here are high level descriptions of methods which I consider important. The code is commented and I've made effort to make the code readable. However, a high-level description should be helpful.

- Getting songs from songs.xml
    - When the app starts, songs.xml is downloaded on Main Activity and parsed.
    - Parsing results in Song objects being created for each song and a list of Song(s) being created. The time-stamp is not checked because this parsing populates the Song(s)
    - Each Song at this stage is simply given number, title, artist, and YouTube link at this point and no other significant thing happens.
    - Each Song, in turn – (in init {..})
        - First run - Downloads the 5 maps and lyrics and saves them to internal storage.
        - For subsequent runs, simply check if these files exist in internal storage.
        - This isn't an issue as initially only 3 songs are playable in the beginning and all data for first three are downloaded first (and the required map1 first)
        - This has been tested thoroughly.
    - Each Song then goes on to read shared preference for words collected in past, guessed status, unlocked status, percentage complete etc. Only small computations are made as these are default for the first run initially and in later runs, no downloading must happen, so there is no lag at all.
    - **Timestamp is not checked because** - needed a method anyway to populate the list of Song(s). Parsing does this.
    - This list of Song is available to all activities as a companion object of main activity.

- Map – Collection of words, distance, level up
    - Distance from marker – wanted distance to be as close as possible. A marker will be collected if (a) it is at least 10 metres near and (b) it lies in the accuracy circle (~ 1 standard deviation confidence or 68% confidence that location lies within that circle.)
        - So, if accuracy is better than 10m, user must go even nearer to the marker. If it is worse, then at least 10m near.
    - Show markers corresponding to words which are not collected yet, and corresponding to map level
        - Choose KML layer based on song.mapLevel (stored as shared preference and updated when level changes)
        - Parse the KML object and plot only those markers which have property "name" not in the list of collected words
        - Add these markers to a hashmap of markers to their "name" property
    - Check if user is near a marker
        - Each time location is changed, check distance from each marker in the hashmap
        - If marker is close enough, remove from the hashmap, add to collected words and remove from the map.
    - Calculate distance travelled
        - Each time map activity is closed, distance in shared preference for the particular song is updated.
        - Each song must have already initialised the distance from shared preference during initialisation.
        - Each time location changes –
            - Calculate the distance between current location and past known location and update song.distance += this distance
    - How to Level Up on map
        - If num. of words collected == words in this level, it means all words for this map is collected (since words left don't include already-discovered-in-earlier-map)

- Other
    - Percentage complete is simply the number of words collected by total words * 100
    - Calories burnt (for stats activity) is tentative. Calculated by average of 100 calories per 1 mile of walking.
    - Unlocking of puzzle – Get the number of guessed songs.
        - When showing list of puzzles:
        - Unlock a new puzzle -> subtract 1 from number of guessed
        - Repeat until number of guessed reaches 0 or no more puzzles left to unlock.
    - Getting "row:column" to name
        - Use words.txt, split on \t and choose the second attribute. This gives the sentence.
        - Then split on "_" (space) and get the corresponding word.
        - Total words can also be counted in a similar manner.
        - A class Lyrics is defined to make this easy.

Experience with Kotlin

The first noticeable thing was that code written in Kotlin was much easier to read. So much so, that to understand a piece of code written in Java for Android, I would first copy it into android studio and convert to Kotlin! Having worked with Scala recently, I noticed some very welcome similarities like data classes (case classes in Scala), null safety (even though Kotlin does it better), and expressions-anywhere etc. However, even though Kotlin provides all these and more features like Scala, Kotlin was easier to learn and more natural to understand. Overall, Kotlin seemed like a new language – new as in, an improvement.

Even though most of the code examples available online are in Java and I faced hardship in the beginning, I quickly got used to it. I even used a repository written in all Java for android and had no issues – of course, I understand why – It is based on JVM. But this means that the language can be used pretty much anywhere, which makes it more attractive. I haven't used Android studio with Java, so it is hard for me to compare, but Kotlin's integration with Android studio was seamless and genuinely helpful.

Sometimes the Kotlin extension didn't recognise Java code or didn't translate it well, but it was clear that as Kotlin is adapted widely for android, these bugs will get fixed.

Most importantly for me, using Kotlin was fun! Trying some Kotlin Koans and exploring the language's offerings got me hooked since the beginning. I would advocate (as I've already done) new android developers to start with Kotlin. I am certain that I would like to work on my Kotlin skills as it becomes a mainstream programming language and employers actively start looking for Kotlin developers.

Acknowledgement

- AppIntro 4.2.2 - https://github.com/apl-devs/AppIntro

2 Feedback on your submission

2.1 Your implementation document overall

You were given a list of subjects for your implementation document to cover.

- *Your experience of Kotlin as a development language, including:*
  - *good features of the language which you found helpful;*
    - ▷ You have listed several of these, including improved support for protecting against null pointer errors, which is a major technical advantage of the Kotlin language.
  - *bad features of the language which you found to be problematic or confusing;*
    - ▷ You have not commented on any language syntax or semantic issues which might be seen as shortcomings of the Kotlin language.
  - *whether or not you would recommend that Java developers should start to migrate to Kotlin.*
    - ▷ You have given your recommendation here, but you have not included any reasons why you have made this recommendation.
- *Algorithms and data structures used for the core functions of the implementation.*
  - ▷ You have documented these thoroughly, and your descriptions are detailed.
- *Parts of your design which have not been realised in the implementation.*
  - ▷ You have reported that there are no major omissions from your design.
- *Additional features of your implementation which were not described in your design.*
  - ▷ You have added some additional features to improve your design.
- *Instructions for installing the server-side of your project, if you have one.*
  - ▷ Your project does not have a server-side component. This is fine.
- *Acknowledgements of code which comes from other sources.*
  - ▷ You have included acknowledgements for source code which you have used in creating your app but without much detail.

Overall, this is a very good submission and goes some way to providing a clear and accurate description of your implementation.

2.2 Functionality

Your app was to be a playable implementation of the Songle game, extended with bonus features of your choosing. There are five key requirements that any implementation of the game should fulfill.

1. The game should determine the location of the user and show where they are located on the map.

   ▷ Your game does this.

2. The game should load the songs to be guessed from the website.

   ▷ Your game does this.

3. The game should show the placemarks which tell the location of each word in the song.

   ▷ Your game does this.

4. The player should be able to collect a word by going to the location of a placemark.

   ▷ Your game does this.

5. The player should be able to guess the song based on the placemarks which they have collected.

   ▷ Your game does this.

2.3 Bonus features

You were asked to implement the *bonus features* which were described in your design document and (if there were any) to document any additional bonus features which you implemented which were not described in your design document. As detailed in the coursework specification, more significant bonus features are more credit-worthy.

— ◇ —

Your game has extensive bonus features which add a lot of value to the Songle game overall. It is clear that significant thought and effort have gone into implementing these, and they have delivered valuable improvements to the basic game.

2.4 Interface

The user interface of your app is an important feature in appealing to potential users. You were asked to follow the user interface design which you submitted for Part 2, enhanced by whatever alterations or improvements you made before this submission. Your interface was evaluated in terms of ease-of-use, completeness, and overall aesthetic.

— ◇ —

Your app does fairly well on most of the above criteria: it is easy to understand and use, it is functionally complete, and it has an appealing and consistent design. However, some of the design elements are slightly confusing. The link for opening the video in youtube is very small and the color used is very different from the color scheme of the game.

2.5 Code Quality

You were told that your code should be readable and clear, making use of private fields and methods and encapsulating code and data structures. You were told that, all else being equal, code with comments should receive a higher mark than code without comments.

— ◇ —

Your app does fairly well on the above criteria: you have structured your code quite well, and have included some comments which describe what your code is intended to do.

2.6 Use of your repository

You were told that, all else being equal, a project where version control using a source code repository such as BitBucket or GitHub has been used to manage the project code and resources throughout the duration of the project should receive a higher mark than one where version control has been used only a little (or has not been used at all).

— ◇ —

You have done this. You have been using your repository throughout the project and you have made regular commits to the repository which have provided you with continuous backups of your work.

2.7 Your mark for this submission

Taking into account the above assessment of your submission, your mark for Part 3 of the Computer Science Large Practical is **90%**