

Software Engineering Large Practical

Stephen Gilmore

School of Informatics

October 9, 2017

About the Software Engineering Large Practical

- The Software Engineering Large Practical is a programming practical which seeks to develop your **independence and confidence** in programming by giving you more practice on a larger project.
- Different from practicals in second year, the SELP gives you some freedom to **shape the direction of the practical**; not everything is specified for you.
- The course promotes **independent, creative study and active learning**, encouraging you to research concepts which can be used to develop extensions to improve your project.
- The final implementation which you deliver should be **uniquely yours**, incorporating your own ideas and design.

Availability

- The Software Engineering Large Practical is a 20 point Level 9 course which is available for Year 3 undergraduate students in Informatics.
- It is not available to visiting undergraduate students or students in Year 4 or Year 5 of their undergraduate studies.
- It is not available to postgraduate students.
- Year 4, Year 5 and postgraduate students have other practical courses which are provided for them.

- The Software Engineering Large Practical consists of one large design and implementation project, done in three parts.
- The first part is **administrative only**, requiring setting up and populating a source code repository for the practical.
- The second part is a **design document**, presenting the plan of the implementation work which will realise the design.
- The third part is the **implementation**. This should be a well-engineered implementation of the previously-supplied design.

Course timing

	Deadline	Out of	Weight
Part 1	16:00 on Wednesday 11th October	0/1	0%
Part 2	16:00 on Wednesday 8th November	100	50%
Part 3	16:00 on Wednesday 13th December	100	50%

- Parts 2 and 3 are equally weighted and constitute the assessment for the Software Engineering Large Practical.
- There is no exam paper for this course.

Software Engineering Large Practical 2017/2018

- The requirement for the Software Engineering Large Practical is to use the **Android Studio** development environment to create an app implemented in Java and XML for an Android device.
- The app implements a **location-based mobile phone puzzle game** which allows users to follow a map and collect words which have been scattered at random around the University of Edinburgh's Central Area.
- The words make up the lyrics of a well-known song and the puzzle aspect of the game is to guess the song from the words which have been found. Given that it is a song-based puzzle, the game is called **Songle**.

Frequently asked questions

- *I don't have an Android device. I've never written an app before. How can I do this practical?*
 - You don't need to have an Android device to do this practical exercise. The software which you develop will run on an emulator which is freely available for Windows, Mac OS X, and GNU/Linux platforms. There is no expectation that you have written an app before: you will learn how to do this in the course of this practical. You may also need to learn more about Java programming.

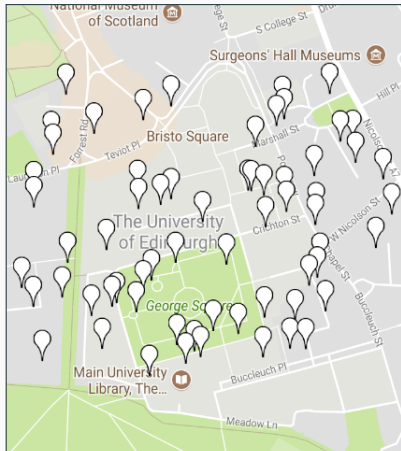
Frequently asked questions

- *I do have an Android device. Is there a specified device for this practical or a specified Android version?*
 - No; you can choose the Android version. If you have an Android device then choose the specification for your device, to allow you to test your app on a real phone. If you do not have an Android device then choose the emulator for a relatively recent device and a relatively recent version of the Android platform.

Songle: a song-based puzzle game

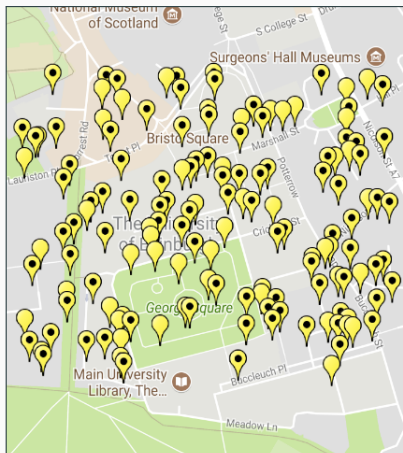
- Words are collected by *visiting their location on the map*, by which we mean that the player literally moves to that location with their mobile phone.
- There are five different versions of the map for each song, each giving *progressively more hints* to help the user guess the song more easily.
- Words are classified as either *boring*, *notboring*, *interesting* or *veryinteresting*.
 - Words which are classified as *boring* are short, common words (such as 'the', 'an', 'and', etc.)
 - Words which are classified as *veryinteresting* are longer, less common words (such as 'Scaramouche', 'thunderbolts' and 'lightning').

Map version 1 (25% of the words)



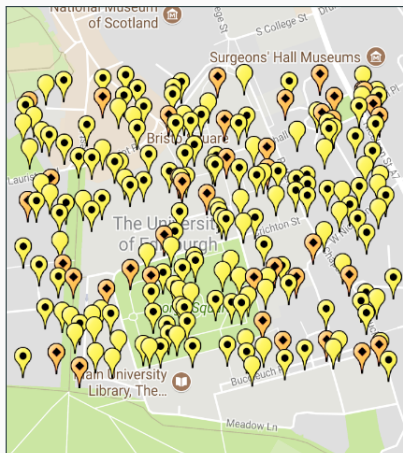
Key: *unclassified* (📍); *boring* (🟡); *notboring* (🟠);
interesting (🟡); *veryinteresting* (🔴)

Map version 2 (50% of the words)



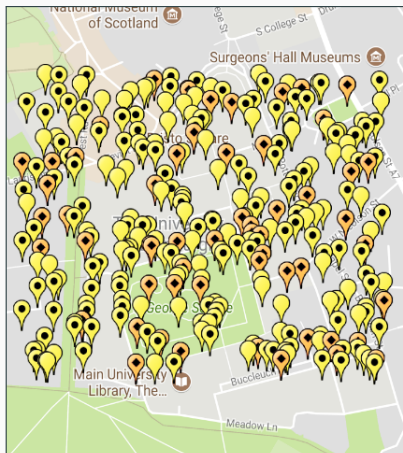
Key: *unclassified* (📍); *boring* (🟡); *notboring* (🟠);
interesting (🟡); *veryinteresting* (🔴)

Map version 3 (75% of the words)



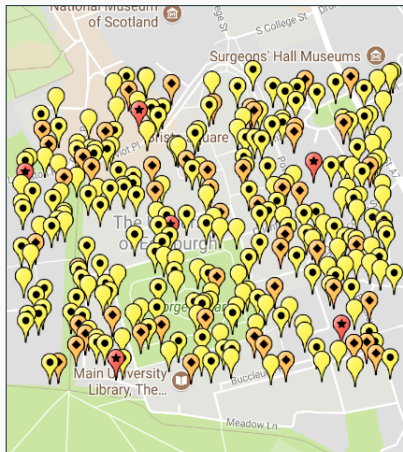
Key: *unclassified* (📍); *boring* (🟡); *notboring* (🟠);
interesting (🟡); *veryinteresting* (🔴)

Map version 4 (100% of the words)



Key: *unclassified* (📍); *boring* (🟡); *notboring* (🟠);
interesting (🟡); *veryinteresting* (🔴)

Map version 5 (100% of the words)



Key: *unclassified* (📍); *boring* (🟡); *notboring* (🟠);
interesting (🟡); *veryinteresting* (🔴)

- The game is backed by a collection of songs with associated maps, stored in XML format on an HTTP server.
- This collection of songs is periodically extended with additional songs in order to keep players interested in the game as time goes by.
- Your app should check for a new version of this XML document every time the user begins a new game.
- The XML document contains a timestamp which is updated every time a new version of the document is released.

<?xml version="1.0" encoding="UTF-8"?>

<Songs timestamp="2017-09-10T20:10:22.563+01:00[Europe/London]"
root="http://www.inf.ed.ac.uk/teaching/courses/selp/data/songs/" >

<Song>

<Number>01</Number>

<Artist>Queen</Artist>

<Title>Bohemian Rhapsody</Title>

<Link><https://youtu.be/fJ9rUzIMcZQ></Link>

</Song>

<Song>

<Number>02</Number>

<Artist>Blur</Artist>

<Title>Song 2</Title>

<Link><https://youtu.be/SSbBvKaM6sk></Link>

</Song>

...

</Songs>

Maps in KML format

Maps

- There is a **Songle Word Map** for each song, made available in the Keyhole Markup Language (KML) format used by Google Earth and other geographic visualisation software.
- The word maps are also available in **HTML and text formats** for ease of reading and browsing online.
- A file **words.txt** contains the **song lyrics with line numbers** which count the lines of the song.

Contents of a Sngle map

- A KML document is a list of **Placemarks** with optional **Style** decorations.
- Each Placemark contains a name giving the unique numerical identifier of the place, a description giving the letter which is available here, and a **Point**.
- A Point has coordinates in the format `<longitude, latitude, height>` where the height is always 0 in our maps, and thus can safely be ignored.

Example Songle map (1/3)

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document> ← previously omitted in error
```

```
<Style id="unclassified">
```

```
<IconStyle>
```

```
<scale>1.75</scale>
```

```
<Icon>
```

```
<href>
```

```
http://maps.google.com/mapfiles/kml/paddle/wht-blank.png
```

```
</href>
```

```
</Icon>
```

```
</IconStyle>
```

```
</Style>
```

wht-blank.png = 

...

Example Sngle map (2/3)

```
<Placemark>  
  <name>1:2</name>  
  <description>unclassified</description>  
  <styleUrl>#unclassified</styleUrl>  
  <Point>  
    <coordinates>-3.1920514249267513,55.94610672008997,0</coordinates>  
  </Point>  
</Placemark>
```

3.192° W, 55.946° N

...

Example Songle map (3/3)

```
<Placemark>  
  <name>26:4</name>  
  <description>unclassified</description>  
  <styleUrl>#unclassified</styleUrl>  
  <Point>  
    <coordinates>-3.188107575106777,55.94423294710405,0</coordinates>  
  </Point>  
</Placemark>
```

3.188° W, 55.944° N

```
</Document> ← previously omitted in error  
</kml>
```

Getting “near” to a word

- In designing your game you should decide **how near a Placemark the player physically needs to be** before they can be considered to have collected that word.
- GPS-based devices cannot determine your true location perfectly but the Android LocationManager API at least attempts to determine the accuracy of its estimated location.

Underspecification

- The above description of the Songle game provides you with a list of requirements for the game but it **does not specify the game design completely**.
 - For example, it does not specify when or how the game is made easier for the player by switching to one of the more informative maps, nor whether this is controlled by the player or it happens automatically on some condition.
- **Underspecification** like this occurs quite often in software projects and it is deliberately included in this practical in an attempt to make the practical more realistic and in order to allow you to be creative in how you design your application.

Bonus features

- In addition to the game features described above you should design and implement some **bonus features**, such as:
 - giving hints such as free words, or even displaying a line from the song;
 - recording the distance walked by the user while playing the game;
 - timing-based play (play against the clock);
 - play modes (easy, moderate, difficult) or attainment levels (beginner, advanced, expert);
 - a scoring system which reflects the attainment level, or play mode;
 - ...
- You should aim to add **two additional bonus features**, but games with more than two additional bonus features are also welcome.

Software Engineering aspects

This practical helps you to develop three useful software engineering skills:

- **using version control systems:** you are to use the Git version control system to manage the source code of your application—learning how much and when to commit code is a useful skill;
- **writing automated tests:** you are to write automated tests for your code and submit these together with the source code of your application; and
- **writing readable source code:** the Java source code which you submit will be inspected for clarity and readability (as well as correctness) so you should try to write clear, easy-to-read code.

Frequently asked questions

Frequently asked questions

- *Can I develop my app on my laptop?*
 - Yes. You are strongly encouraged to do this because it will encourage you to investigate the Android SDK and related libraries. Of course, we recommend taking regular, well-organised backups, which is an important service provided by a version control system.

Frequently asked questions

- *Can I develop my app on my laptop?*
 - Yes. You are strongly encouraged to do this because it will encourage you to investigate the Android SDK and related libraries. Of course, we recommend taking regular, well-organised backups, which is an important service provided by a version control system.
- *Instead of Java, can I implement my app in Ruby/Python/Scala/C#?*
 - No, not for this practical. We need all students to be working in the same programming language in order to make a fair assessment.

Frequently asked questions

- *Do I have to develop in Android Studio? I much prefer Eclipse/Emacs/vi etc.*
 - You are required to submit an Android Studio project so we strongly recommend developing in Android Studio for this practical exercise.

Frequently asked questions

- *Do I have to develop in Android Studio? I much prefer Eclipse/Emacs/vi etc.*
 - You are required to submit an Android Studio project so we strongly recommend developing in Android Studio for this practical exercise.
- *Do I have to use Git? I much prefer Subversion/Mercurial/Darcs etc.*
 - Yes. Git is the chosen version control system for this practical because it is supported by Android Studio. However, please note that we are using a **private Git repository**: do not upload your code to a public repository where anyone can see it!

Software Engineering Large Practical

Stephen Gilmore (Stephen.Gilmore@ed.ac.uk)
School of Informatics

Document version 1.0.3

Issued on: October 9, 2017

About

The Software Engineering Large Practical is a 20 point Level 9 course which is available for Year 3 undergraduate students on Informatics degrees. It is not available to visiting undergraduate students or students in Year 4 or Year 5 of their undergraduate studies. It is not available to postgraduate students. Year 4, Year 5 and postgraduate students have other practical courses which are provided for them.

Scope

The Software Engineering Large Practical consists of one large design and implementation project, done in three parts. The first part is administrative only, requiring setting up and populating a source code repository for the practical. The second part is a design document, presenting the plan of the implementation work which will realise the design. The third part is the implementation, which should be a well-engineered implementation of the previously-supplied design.

Part of the SELP	Deadline	Out of	Weight
Part 1 (Admin)	16:00 on Wednesday 11th October	0/1	0%
Part 2 (Design document)	16:00 on Wednesday 8th November	100	50%
Part 3 (Implementation)	16:00 on Wednesday 13th December	100	50%

Please note that although Part 1 of this practical is not weighted, you are strongly encouraged to do it both in order to ensure that you are keeping up with the material of the course as it progresses and because your use of your source code repository is a factor in the assessment for Part 3 of the practical. Parts 2 and 3 are equally weighted

Any questions?
