**The Chinese University of Hong Kong**

**Department of Computer Science and Engineering**

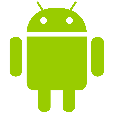
**CSCI 4140 Final Project Proposal**

Group 22

Poon Bing Chun 1155063232

Project Title: Tempus E Spatium

GitHub repository: <https://github.com/SoftFeta/tempusespatium>

# Overview

In this project, I will build a two-player Android quiz game app on one single device. A multitouch-enabled device is required for this app to run correctly. At the first week of the course, I came up with a Python game idea called ‘Rampant Sphinges’ *(please see Piazza for the game rules)*. I later downsized (but not scraping) it because I wished to work on my own, and I thought the workload of the original is too high for a single person. Still, the new idea remains to be a board game, retains the concept of scraping Wikipedia to generate quiz questions, and involves many facets of advanced Android programming.

The downsized project is named ‘Tempus E Spatium’ (Latin for Time and Space, which is often prefixed by “lost in”). Quiz types include: Fill in the blanks, drag and drop, listen to the national anthem and pin the map, multiple choice, and complete the word. Despite the name of the app, the last type can be of any question type. You can search for any topic as long as it is available in Wikipedia. There are 1000000+ possible questions generated from articles that belongs to 700+ WikiProjects: children’s literature, archaeology, Scottish castles, free software, animal anatomy, Canada roads, military history, relativity, constructed languages, Michael Jackson... you name it, you (probably) have it.

The questions are timed 5 seconds to 1 minute each, depending on the question type and difficulty settings. A game consists of 5 rounds. Each round has its own chessboard. A player advances a variable number of tiles on answering a question correctly, and retreats on answering it incorrectly or failing to answer it on time. The player who reaches the destination first wins the round. The player who wins the most rounds wins the game.

Programming languages used in this project includes Java, SQL, XPath and JavaScript. Markup languages used includes JSON, XML and HTML.

# Development schedule

This mid-scale project is designed to for one person’s workload. Owing to the scarcity of time, the rapid prototyping software development model is adopted. What’s special of this model is that implementation starts right at the beginning, simultaneous with design; instead of the waterfall model (coding only after the design is entirely finished). As a result, the outcome will be robust and bug-free. No Gantt charts 😊.

# Dependencies

The Android application uses support libraries, so it can backward support devices down to API 19 (Android 4.4 KitKat).

It uses the Picasso library for image downloading and caching, the OKHttp library for fetching Wikipedia webpages (GET requests), the Google Maps API, and the W3C XPath library for web scraping. For styling, the Android-Bootstrap library, and the NumberPicker library are used.

The built-in SQLite database consists of two tables URLs of Wikipedia articles (and corresponding question type for generating questions) and high scores respectively, but none of them will hold any image data. SharedPreference will be used for changing locale and difficulty.

It also uses the Natural Earth Data set, available in Google FusionTables for geographic data, which is used to generate country/state boundary polygons on a Google Maps fragment.

All resources used in this project are either in the public domain, or permitted for free use under the Creative Commons (CC) License.

# Impact of the application / Problems

* Not many games on the Play Store are educative, and most edutainment apps are for kids. By contrast, this game is designed to be **hard enough for adults**, including the intellectuals.
* Topic diversity: Any topic, as long as it is available in Wikipedia. 1000000+ possible questions generated from articles that belong to 700+ WikiProjects: children’s literature, archaeology, Scottish castles, free software, animal anatomy, Canada roads, military history, relativity, constructed languages, Michael Jackson... you name it, you (probably) have it.
* The questions themselves are multilingual, as their content are taken from French/German/Spanish Wikipedia. This means you can harness the app to **practise foreign languages as well**.

# Creativity / Goals

* *Tempus E Spatium* is one of the few apps that takes advantage of **multitouch**, which eliminates network latency in many 2-player games. It is a departure from the traditional ‘pass-and-play’ model, too.
* The app makes use of **web scraping** fetches and parses Wikipedia articles and turn them into questions, which virtually no games are using this technique at the moment.
* The app belongs to the new field of **‘edutainment’**: learning masquerading as playing. As the saying goes, ‘all work and no play makes Jack a dull boy.’ The app focuses on the holistic development of a person instead of turning him a nerd, by raising cultural awareness and sharing knowledge that is oft-neglected.
* **The soft keyboards are specially designed.** Keyboard layouts include **QWERTY, AZERTY (French) and QWERTZ (German)**, no additional downloads are needed. Note that there are two keyboards on each side, and each of them only enters text into their text fields. I use advanced techniques including Android’s @JavascriptInterface to create the **illusion of two cursors**, overcoming two major technical difficulties of having one cursor/focused item, and the lack of direct two-way communication channel between WebView widgets and JavaScript.
* One minigame involves **natural language processing**. You can find details in the Evaluation Plan part.

# Features

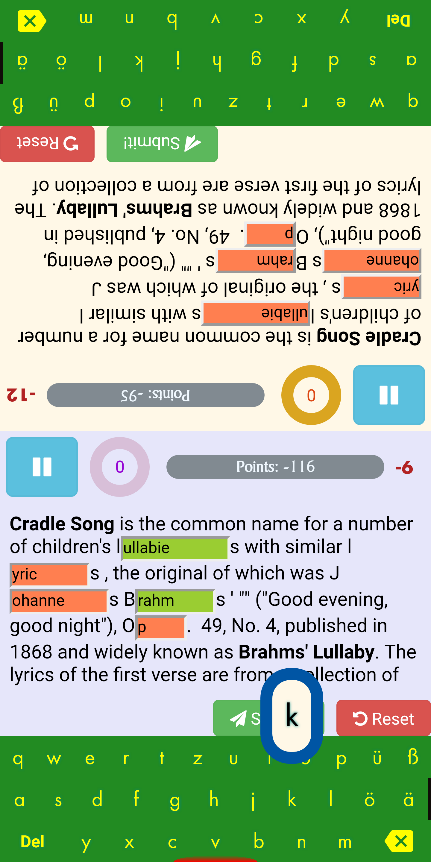
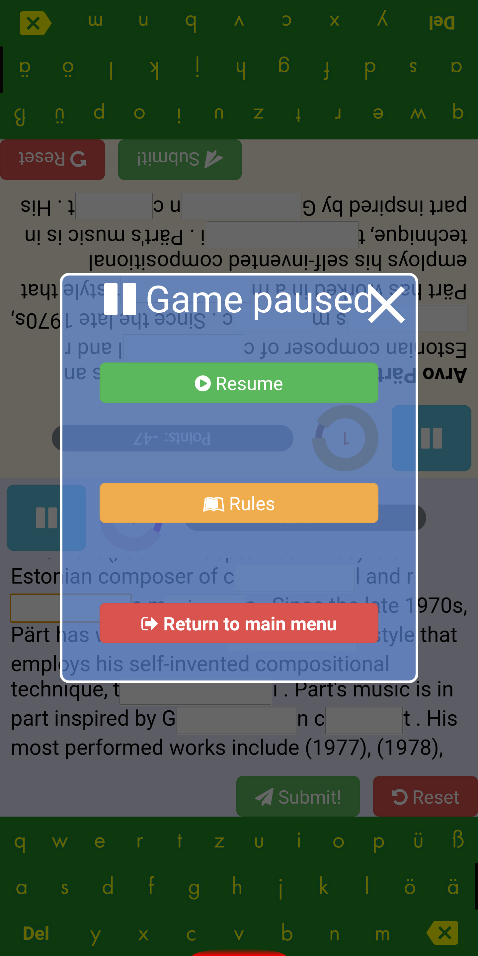
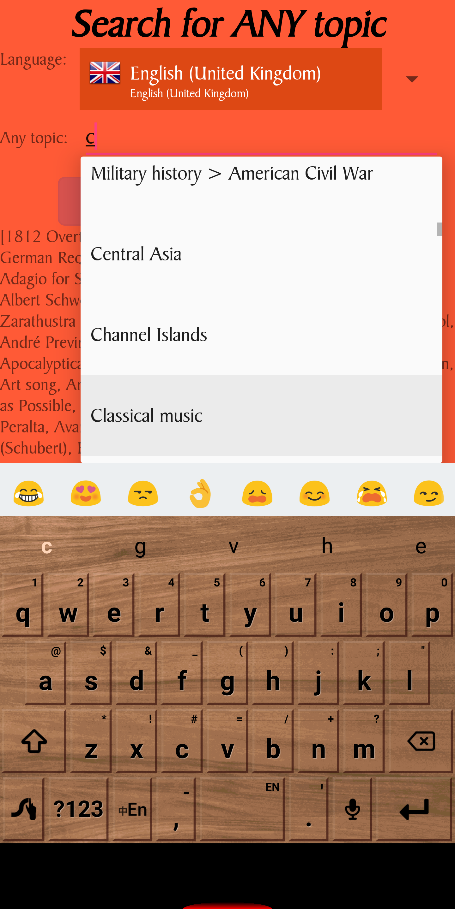
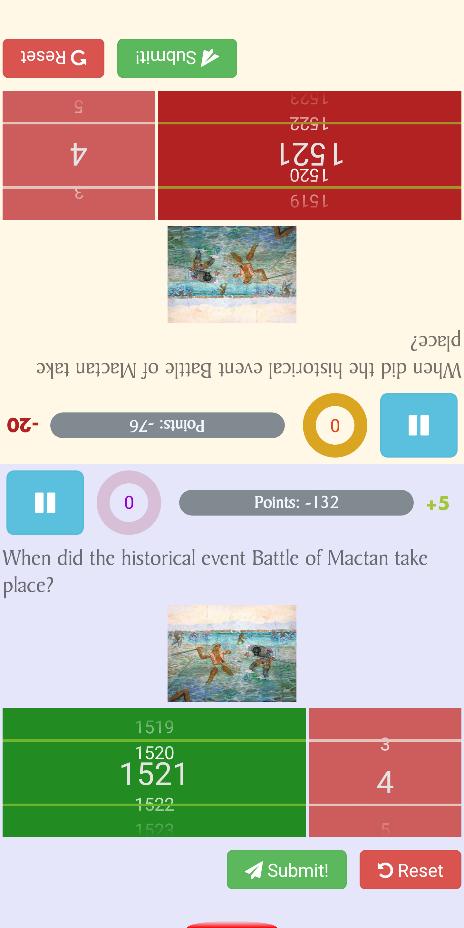
* 1000000+ possible questions generated from articles that belongs to 700+ WikiProjects
* Multilingual: supports (at least) 7 languages: Catalan, Chinese, English, French, German, Japanese and Spanish. UI locale and question language are separated.
* 6+ game types: complete the paragraph, drag and drop, double roller, listen to a tune and pin the map, choose the correct one, anagram, …
* The game is pausable **(asynchronous programming using a custom semaphore class)**
* Highscores (via Google Play Games API)
* 2 difficulty modes (Hard and Insane)
* Aesthetics
  + Animations: Physics-based animation (fling & spring) if API 25+.
  + Bootstrap controls and Font Awesome icons

# Evaluation plan

All features below has been realised in my prototype, yet they are in a preliminary stage and improvements are needed:

* I use OKHttp, and the rarely known *org.w3c.dom* and *javax.xml.xpath* APIs for web scraping. One WikiProject equals one topic. Various means (e.g. visit count in English Wikipedia, quality scale in German Wikipedia) are used to sort the articles. LinkedHashMap is used to record the insertion order of article entries.
* One minigame (pinning the map) involves Google Maps, Geocoder, and GeoJSON APIs.
* I use SQLite to store data like highscores.
* Audio streaming is involved.
* Even with @JavascriptInterface introduced in API 19, there is no direct way to get the contents in a WebView. Therefore, I exploit @JavascriptInterface interface by sending multiple alert()s so the callback function can get it. It is possible because the callbacks are non-blocking, i.e. not executed in the main thread.
* Pausing the game, countdown timers, and the above Javascript exploit (you have to make sure the number of correct answers is propagated by the callback function from another thread, before you get it in the main thread) are examples of asynchronous programming. A form of semaphore called the CountDownLatch is used. It is notable that widget objects cannot be shared among threads in Android.
* I have to be aware of memory leaks. Some operations, like OKHttp requests, cannot be done on the main thread.
* One minigame (matching relevant items) involves natural language processing. Snowball ([https://snowballstem.org/)](https://snowballstem.org/) is an old but useful Porter stemmer, most importantly, it supports English, German and Romance languages. It is not a library but a code generator and supports Java classes. Alternates include Apache Lucene but considerable efforts are needed to turn it Android-compatible. From a scraped Wikipedia article, we use XPath to extract plain text from HTML; use regular expression to remove symbols and whitespace; and use Snowball to **stem** words (lemmatisation works better, e.g. ‘better’ to ‘good’, but it involves much hard coding), with **stop words** (e.g. ‘whereas’, ‘whom’, ‘between’) removed, and obtain a **bag of words** ({‘constitut’: 5, ‘govern’: 4, ‘anthropolog’: 2}. Then we calculate the term frequency percentage. Repeat the process for two more topics. Irrelevant words can be generated from a predefined set of words, and search if they exist in the bag of words.

# Prototype

Since 31st January, I have been building a working prototype. Here are the screenshots: