Soundscape Android framework choices

Project goal

Although this is the "Soundscape for Android" project, it's highly desirable that at the end of it we have code that can be run on Android and also targetted at other platforms. The original iOS Soundscape was written in Swift and is tightly bound to iOS. It's a useful starting point, but the code isn't easily reusable on Android. The overall aim is that the code that comes out of this project will run on Android and iOS with just a re-compile.

Project components

Here's a partial list of the components that make up the Soundscape application:

- Turning OpenStreetMap data into audio prompts
- Displaying map on screen with waypoints overlayed
- Audio engine - including 3D beacons and text to speech
- UI - waypoint editor and generally app navigation
- Compass/Head tracking for direction
- GPS smoothing

Frameworks

As with most software projects, the aim is to write as little code as possible, use the most widely used and widely tested libraries available and have a fast and easy to use development environment. The chosen framework needs to have good documentation and the more developers the better.

Flutter

Google flutter uses the Dart programming language and the development environment is Visual Studio Code. This is a good introduction to writing apps using flutter. One of the most compelling features of flutter is the ability to change code whilst the app is running and the app behaviour will alter whilst it's still running. This allows for fast UI development with very little time spent waiting for compiles/loads. However, it only works when targetting desktop applications, and not when targetting Android/Web/iOS. For a UI only app this is great, but isn't a significant gain for us.

There are loads of great libraries for Dart but many of them only work on a subset of the platforms that flutter supports - Web, Desktop, iOS or Android. FMOD which we're hoping to use for the audio component has a library but like many of the libraries I looked at it fails many integration tests.

The development environment was straightforward, and though Dart has a slightly steeper learning curve than I had expected the documentation was excellent. I didn't try integrating any native (C++) code into the flutter project.

Kotlin multiplatform

This is the most obvious platform for writing Android apps. Kotlin is the language and the development environment is Android Studio. There are plenty of simple apps in github that can be used, but I found it straightfoward enough just to start from a new project and write some code. I'm sure computer scientists

would throw their hand up, but Kotlin just felt like C++ from 30 years ago without the semi-colons... Coming from C++/JavaScript/Python the syntax is straightforward and the documentation is good.

Android Studio seems as powerful as Visual Studio Code and made running on target devices straightforward. It was also fairly trivial to add in native (C/C++) code libraries and to call them from Kotlin. The syntax highlighting in Android Studio indicates un-called functions and that works across languages, so any C++ APIs that were not called from Kotlin were obvious. This helps with switching between languages, and in general the Kotlin errors and warnings were very helpful.

The chosen framework

Kotlin multiplatform seems like the framework that we should choose. It's widely used, and Google has announced official support for it. Kotlin is a slighly easier language to use, seems to have better library support and Android being the main target and is very well supported. There is an obvious path to iOS support, though I think that requires further investigation to best understand how to pull together our libraries into an iOS app vs an Android app.

Whilst we should aim to write new code in Kotlin, libraries from other languages can be easily used and this increases the breadth of available code and available developers.