# Portierung des Advanced Map Viewers

# Projektplan

##### Problemstellung

Die Anwendung AdvancedMapViewer (AMV) wurde programmiert für Smartphones mit dem Android Betriebssystem. Obwohl als Programmiersprache ausschließlich Java benutzt wurde, ist sie nicht kompatibel zu anderen Betriebssystemen, wie iOS, Linux oder Windows.

In der Hoffnung, das zukünftige Smartphone Generationen sich der PC-Architektur immer weiter annähern und somit PC-Anwendungen leicht übertragen werden können, begründet sich die Motivation, den AMV auf den PC zu portieren.

Damit dies gelingen kann, müssen alle Android spezifischen Klassen ersetzt werden durch Klassen, die für die PC-Architektur bestimmt sind.

Dabei muss natürlich darauf geachtet werden, dass eventuelle Abhängigkeiten erhalten bleiben und so wenig wie möglich Funktionalität verloren geht.

Dieser Projektplan beschreibt nun die allgemeine Vorgehensweise dieser Portierung.

##### Lösungsweg

Wie schon erwähnt, ist es leider nicht möglich, die Android-Bibliothek komplett zu übernehmen. Dies hat den Grund, dass das Betriebssystem Android sich zu Windows/Linux besonders in der Speicherverwaltung, der Prozessverwaltung sowie der Anzeige unterscheidet.

Daraus folgt, dass eine solche Portierung der Android-Bibliothek, ohne nahezu jede Klasse anzupassen oder umzuschreiben, nicht funktioniert. Es muss also für jede Android-spezifische Klasse ein passender Ersatz gefunden werden, der auf jedem PC funktioniert und kaum Funktionalität einbüßt.

Stattdessen wird versucht, die Android-Bibliothek soweit es geht durch die Java-Standard-Bibliothek abzubilden (z.B. Speicher- und Prozessverwaltung). Die Nutzung von anderen Bibliotheken ist aber nicht ausgeschlossen, wobei die Java-Standard-Bibliothek höchste Priorität hat.

Für die GUI wird generell Swing verwendet (Views, Menü, Widgets, Layout, Events).

Die Darstellung der einzelnen grafischen Objekte auf der Karte übernimmt bisher die Skia-Engine.

Die Skia-Engine ist eine Bibliothek für 2D Grafiken (Texte, Geometrien und Bilder). Sie ist in C++ programmiert und Teil des Android Betriebssystems. Im Projekt Mapsforge (bzw. AdvancedMapViewer) wird Skia nicht direkt verwendet, sondern wurde in die Android-Bibliothek eingebaut. Spezifisch wird Skia im Paket android.graphics benutzt. Das Paket „graph“ enthält nicht alle Skia-Klassen. Einige Skia-Klassen werden vernachlässigt.

Die Skia-Bibliothek kann fast komplett durch das Java-AWT portiert werden. Für alle Klassen, die so nicht umgesetzt werden, muss eine externe Lösung gefunden werden oder die Klassen werden selbst geschrieben, je nach Aufwandsgröße.

Um den Umfang der Portierung abzuschätzen, werden alle betroffenen Klassen untersucht und in einer Liste gesammelt. Damit kann die Anzahl an betroffenen Code-Zeilen ermittelt und so der Portierungsaufwand abgeschätzt werden.

Hier sind es 8 Klassen AdvancedMapViewer + 32 in Mapsforge (map) + ca. 85 Klassen aus der Android Bibliothek, die bei der Portierung betrachtet werden müssen. Diese Zahlen entsprechen insgesamt > 10.000 Codezeilen (Kommentare wurden nicht ausgeblendet).

Sind alle Bibliothekklassen erfolgreich portiert, müssen wahrscheinlich die Klassen aus Mapsforge (map) nicht mehr angepasst werden.

Einige Klassen werden, aufgrund eingeschränkter Funktionalitäten des PCs, nicht umgesetzt (z.B. Standortbestimmung) oder stark umgeformt (z.B. Zoomen).

##### Zeitplan

Für den Zeitplan wurden folgende Einteilungen vorgenommen:

Einarbeitung (3-4 Wochen):

Erlernen und eigenes Beschreiben der Komponentenklassen und ihrer Abhängigkeiten.

Portierung (6-7 Wochen):

Sukzessive Implementierung der zu portierenden Klassen.

Die Portierung wird hier noch folgendermaßen aufgeteilt:

(Bernd): Die Klassen, die für die GUI (Views, Widgets, Events) zuständig sind.

(Martinus): Die Klassen, die für die Map und die Datenstruktur zuständig sind.

(Beide): Resource Management.

Dokumentation API (5 Wochen):

API der portierten Klassen.

Test und Debugging (6-7 Wochen):

Sofortige Tests und Debugging neuer Klassen.

Dokumentation Gesamt (4 Wochen):

Zusammenfassung und Ausführung des gesamten Projekts Arbeit.

Die folgende Tabelle stellt die Arbeitsschritte schematisch dar.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Kalender Woche | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| Bernd Bieber |  |  | Portierung (GUI und Events) | | | | | |  |  |
|  |  | Dokumentation API | | | | |  |  |  |
|  |  |  | Test und Debugging | | | | | |  |
| Einarbeitung | | | |  |  | Dokumentation Gesamt | | | |
| Martinus Dipobagio |  | Portierung (Karte) | | | | | | |  |  |
|  |  | Dokumentation API | | | | |  |  |  |
|  |  | Test und Debugging | | | | | | |  |
| Einarbeitung | | |  |  |  | Dokumentation Gesamt | | | |

##### Verwendete Technologien

Als Entwicklungsumgebung wird Eclipse in Verbindung mit dem Java SDK benutzt.

Zum Verstehen des bisherigen AMV wird das Android SDK incl. Skia-Engine installiert und in Eclipse integriert.

Für die Versionsverwaltung wird Subclipse (SVN) als Eclipse-Plugin benutzt.