

gvSIGDROID

An Open Source Gis Solution for the Android Platform

Cristian Martín-Reinhold, Joaquín Huerta and Carlos Granell
Centre for Intreactive Visualization, Department of Information Systems
Universitat Jaume I, Avda Vicent Sos Baynat s/n, Castellón de la Plana, Spain
christian.reinhold@gmail.com, {huerta, carlos.granell}@uji.es

Keywords: Mobile GIS Applications, Open Source, gvSIG, Android, Web Services, OGC.

Abstract: Mobile GIS applications are gaining attention due to a wide range of potential target applications such as e-commerce, tourism, education, agriculture, and field research. This requires easy-to-use geospatial applications operating on mobile devices to enable both visualization and editing of widely-used geospatial data and formats. This paper introduces gvSIGDroid, an open source geospatial mobile application that attempts to reach the new potential users emerged from the Android platform. The prototype's core functionalities rely on gvSIGMobile modules together with an especially designed user interface for the Android platform, allowing mobile users to retrieve, visualize, navigate and modify both local and remote geospatial layers. Future extensions can be deployed so as to provide missing functionalities such as Location Based Services and data sharing.

1 INTRODUCTION

Mobile devices, mostly in the form of PDAs (Personal Digitalized Assistant) and mobile phones, have nowadays changed access to information and services from desktop devices to ubiquitous computing (Goodchild, Johnston, Maguire and Noronha, 2004). This new environment has led companies to migrate and readapt their traditional desktop software applications in order to obtain the best mobile customers' experience on a user-centred scenario (Boll, Breunig, Jensen, König-Ries, Malaka et al., 2004), by providing solutions based either on open source software or commercial packages.

In the Geographical Information Systems (GIS) context, mobile GIS applications can be generically classified into field-based GIS solutions and Location Based Services (LBS) applications (Schiller and Voisard, 2004). The former have a long history since first GIS users already used maps to collect and visualize geographic information (Brown, 1949). On the other hand, LBS applications aim to offer location-based functionality, such as navigation, street and service finding, and route tracking. These services are normally developed as web applications like the Open Route Service (Neis and Zipf, 2008; <http://openrouteservice.org>)

This paper describes the first results of

gvSIGDroid, an ongoing prototype fully developed with open source tools and components on top of the recently emerged Android mobile platform (<http://www.android.com/>). Although it is still an initial release, gvSIGDroid provides a complete GIS mobile application with common GIS functionalities that allows users to manage locally geospatial data, either in vector (in the form of points, lines, polygons) or raster formats (any kind of image format). Besides, the application permits to connect to remote services like web mapping services compliant with the Open Geospatial Consortium (OGC, <http://www.opengeospatial.org/>) service specification Web Mapping Service (WMS) (de la Beaujardiere, 2006). Essentially, the WMS specification describes a set of HTTP-based service interfaces so that any client application like gvSIGDroid may connect, query and retrieve maps from any remote WMS-compliant mapping service elsewhere.

The rest of the paper is organized as follows. Next section describes the conceptual architecture that supports gvSIGDroid. Section 3 shows the first results achieved by the gvSIGDroid application. Finally, in section 4 we provide our conclusions and suggest future developments and research lines.

2 ARCHITECTURE

gvSIGDroid application comprises various open source components following a conceptual architecture based on the Model-View-Controller paradigm (MVC), as illustrated in Figure 1. Wrapping classes that extend the built-in classes were created to complement and simplify from the developer's perspective the interaction with the Android platform's components.

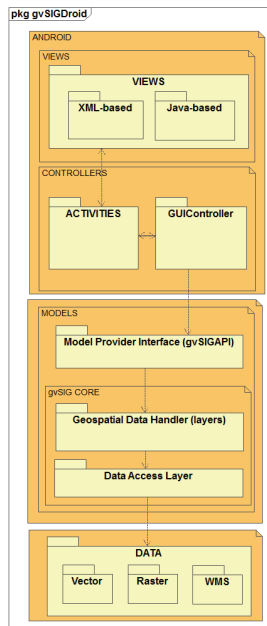


Figure 1: gvSIGDroid's architecture.

As shown in Figure 1, different modules have been implemented to offer users a logical way of interacting with the application. So, each module contains Java classes of all three MVC layers: a set of classes used represents the Presentation or View layer, the *Controllers* and *Activities* corresponds to the Controller layer, whereas the whole gvSIG core along with the *Model Provider Interface* API class constitute the Model layer.

2.1 View Layer

The View or Presentation layer contains all the components that exploit the Android's graphic utilities and capabilities to develop the application user interface. Accordingly to the Android architecture (Android, 2008), the Android platform admits two mechanisms for creating user interfaces, one based on java code and other in XML declarations (similar to Flex user interfaces developments). In our application, we offer a *View*

class that encapsulates both Android's mechanisms as a base class for creating and instantiating all of the graphical objects used. In essence, once instantiated a graphical object, it delegates to native Android methods for input and output operations such as input user and drawing capabilities.

2.2 Controller Layer

The aim of the Controller or Business Logic layer is to constantly catch any user request via the *View* objects of the Presentation layer, and also to delegate these requests to the most appropriate component. The first task is handled by the *Activities*, which contain the main graphical user interface and Android *Controllers* from the Android application framework layer.

Within the *Activities*, these Android *Controllers* are in charge of handling events triggered on user interface objects, like *onClick()* for images and buttons or *onListItemLongClick()* for items of a *ListView*. In addition *Activities* handle system events, like the *onPause()* event fired when another application of the system is going to gain the focus (an incoming call for example). All these scenarios must be correctly handled for every module.

Apart from the events aforementioned, there are several actions that need to be handled on a *View* as well, which are:

- At Business Logic level: Calling other activities. For example, showing a new screen implies the task of calling to a new activity, which will be handled by the *GUIController* class.
- At Model level: To provide the information needed from the internal gvSIG data model, which should not require any graphical control, a different controller that implements a *Model Provider Interface* was used.

2.3 Model Layer

The main goal of the Model layer is to enable data persistence. This is achieved using the data model of gvSIGMobile, so that our application exploits all the geospatial data capabilities already supported in gvSIGMobile. However, other components were created to facilitate the access and retrieval of datasets from remote services.

The bottom of Figure 1 shows the components involved in the Model layer. From top to bottom, we have the *ModelProviderController* component. It provides an internal model that abstracts the core gvSIGMobile data model (*Geospatial Data Handler*). The *ModelProviderController* delegates

the tasks of retrieval and storing of geospatial layers to the gvSIGMobile module, permitting thus a loosely coupled design.

Besides the layer management capability, the *ModelProviderController* also links to the core module of gvSIG (*Data Access Layer*). This module contains the business logic necessary to manage vector and raster data formats as well as handling remote geospatial layers from WMS services into gvSIG-compliant layer objects. Furthermore, the *ModelProviderController* component allow us to proxy all the data handler and access capabilities to gvSIG and gvSIGMobile modules, which have proven to be successful in diverse production applications.

3 RESULTS

The main purpose of gvSIGDroid is to exploit Android SDK to provide a user-friendly application, which results more in keeping with the user's expectation. As gvSIGMobile has been recently released, some of their potential functionalities were not available on the first prototype of gvSIGDroid, like for instance advanced edition capabilities and Global Positioning System (GPS) support.

3.1 Layers Module

Figure 2 illustrates the use cases for adding both local and remote layers into gvSIGDroid. Figure 2(a) guides the use in adding a local layer. After a filter for the file browser is selected (shp, kml, gml, etc.), all available layers are listed. Then, users just need to choose the preferred layer to visualize it.

The second row of images in Figure 2 lets users connect to WMS services to retrieve remote layers. This option is only possible when an Internet connection is available. As shown in Figure 2(b) a server can be selected based on a given list and a dialogue window appears, letting the user select the layers of interest to be loaded. Here, the gvSIGDroid application makes use of the *Data Access Layer* module (See Figure 1) in order to connect and interact with WMS services according to its specification. Once the dialogue is dismissed by pressing *OK*, some of the layer's properties like format (*Format* textbox) and reference system (*SRS* textbox) can also be adjusted. Finally, when pressing again the *OK* button the selected layers will be loaded into the project as in the case of adding local layers.



Figure 2: Adding a local (a) and a remote (b) layer.

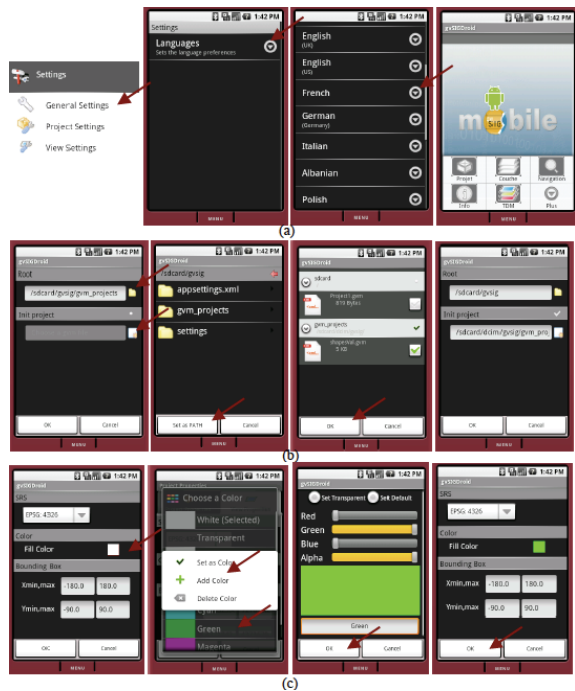


Figure 3: General (a), project (b) and view (c) settings.

3.2 Navigation and Settings Modules

Users also need to navigate through the layer, being able to perform actions like zooming in, zooming out, re-centering the layer by point, measuring paths and areas, and so on. For managing this actions, no views were needed, just a context dialogue was implemented to let the user select between these different commands from a corner of the map view panel.

Figure 3a shows the general settings (language, locale, etc.). Figure 3b illustrates the options supported for project configuration: setting the default root from which file browsers will start, and setting a *gvm* file to be initialized each time the application re-launches. Finally, Figure 3c shows some configuration screens for view settings.

4 CONCLUSIONS AND FUTURE WORK

This paper has described a first attempt in migrating gvSIGMobile software into the Android platform. After some difficulties integrating both libraries, the first prototype implementation, with the aid of the full graphical user interface in Android, led to a clean, nice and user-friendly interface.

gvSIGDroid allows to GIS mobile users to manage almost any kind of geographic data resource. Consequently, users can load, access, edit, and visualize geospatial data about their surroundings from the field, facilitating on-the-fly decision-making. Most importantly, this application allows user to connect and access to remote web map services in compliance with the OGC WMS specification. This means that gvSIGDroid is in the way to becoming a mobile SDI client for Spatial Data Infrastructure. Future work in this direction will be focused on supporting other standard geospatial services, like Web Feature Services (Vretanos, 2005) for retrieving remote vector data (e.g. GML files).

Nevertheless, the first release contains several inconveniences. For example, the loading and drawing of the layers take much longer than in gvSIGMobile. This is due to the fact that gvSIGDroid first uses the core objects as they were designed in gvSIGMobile, before adapting them to its unique Android objects for drawing, consuming substantial time and resources. In future implementations, this limitation may be addressed by transforming the core module that contains gvSIGMobile objects into native Android core objects. Another inconvenience for field-based researchers when using gvSIGDroid is that, within the program, there is still no possibility of taking advantage of the GPS hardware of the device, since the GPS module has not been yet implemented.

Some future work should be done in order to enhance gvSIGDroid with a wide set of GIS functionalities. On the one hand, it is important to explore new ways of loading big shapefiles more efficiently. As said, the core module should be re-

implemented to provide Android shapes' inner classes at earlier stages and merge both Android and gvSIG data shape models for drawing.

ACKNOWLEDGEMENTS

This work has been partially supported by the EC Erasmus Mundus Programme M. Sc. in Geospatial Technologies. We would also like to thank the Prodevelop staff for their invaluable collaboration and help in the use of gvSIGMobile.

REFERENCES

- Android, (2008). Android Architecture. Retrieved October 24, 2009 from <http://developer.android.com/guide/basics/what-is-android.html>
- de la Beaujardiere, J (ed.) (2006). OpenGIS Web Map Service (WMS) Implementation Specification, version 1.3.0. Open Geospatial Consortium, Implementation Specification.
- Boll, S., Breunig, M., Jensen, C. S., König-Ries, B., Malaka, R., Matthes, F., Panayiotou, C., Saltinis, S. & Schwarz, T. (2004). Working Group - Towards a Handbook for User-Centered Mobile Application Design In *Proc. of Mobile Information Management*. Dagstuhl: IBFI.
- Brown, L. B. (1949) *The Story of Maps*. New York: Bonanza Books.
- Goodchild, M. F., Johnston, D. M., Maguire, D. J., & Noronha V. T. (2004). In R. B. McMaster & E. L. Usery, (Ed.), *A Research Agenda for Geographic Information Science. Distributed and mobile computing.*, (pp. 257-286). Boca Raton: CRC Press.
- Haklay, M. & Weber, P. (2008). OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Computing*, 7(4): 12-18.
- Neis, P. and Zipf, A. (2008): Extending the OGC OpenLS Route Service to 3D for an interoperable realisation of 3D focus maps with landmarks. *International Journal of Location Based Services* 2(2): 153-174.
- Schiller, J. & Voisard, A, (eds) (2004). *Location-Based Services*. Morgan Kaufmann.
- Vretanos, P. A. (ed.) (2005). Web Feature Service (WFS) Implementation Specification, version 1.1.0. Open Geospatial Consortium, Implementation Specification.