ODK Tables Maps
Christopher Gelon
Department of Computer Science and Engineering
University of Washington, Seattle, WA [USA]
cgelon@cse.uw.edu

ABSTRACT
Open Data Kit (ODK) [1] is a set of open source tools intended for use in the developing world. One of these tools, ODK Tables [7], is a data management application on Android [2] with the ability to be customized by users with minimal overhead. A new enhancement, ODK Tables Maps, adds a virtual map. With this, users can see areas that need help, plan routes, and visualize data in entirely new ways. This paper will discuss the different ways data can be displayed, both on a phone and on a tablet, how fragments will help ODK Tables in the future, and the differences between two available map APIs: Open Street Map [3] and Google Maps [5].

Keywords
Open Data Kit, mobile phones, mobile database, maps, Google Maps, Open Street Map

1. INTRODUCTION
Mobile tools that simplify mobile data collection, such as Open Data Kit (ODK), helped make smartphones a suitable platform for ICTD applications. In addition to collection, organizations are looking for new ways to view this data, including map visualizations.

This paper focuses on providing a simplistic way to display data on a map. By using data collected through ODK Collect [8], ODK Tables enables users to view this data through spreadsheets, lists, and graphs. ODK Tables Maps adds a new way to view the data: on a map.

ODK Tables Maps gives users several options to view their data, including color-coded markers, an easy interface to view additional data, and the ability to easily add data points at any location.

ODK Tables Maps helps organizations in a multitude of ways. For example, if a group is collecting data on vaccines delivered to hospitals throughout Benin, using the map could help determine which areas need focus. Or, if a hospital is low on vaccines, which hospitals in the area could help. Another example is mapping a route to a remote location in Kisumu: by taking data points of landmarks in the area, the group can then create a map for others to follow.

In this paper I will discuss the process of building the mapping tool, including the decisions made along the way. After that, I will explain the final solution, and how well it performs with large datasets.

2. REQUIREMENTS
The requirements of this project evolved from a set of use cases for ICTD Projects in the developing world. The primary dataset used in testing contains information about different medical facilities in Benin. Each facility has a latitude and longitude, in addition to other information. From this dataset, the following requirements were derived to help visualize the data in a new way:

- **Data collected in ODK Tables displays on a map.** Data in the spreadsheet view of ODK Tables is displayed through rows. Each row corresponds to several columns. ODK Tables Maps should display each row as a marker on the map.
- **Quick access to detailed views of each data point from the map.** In addition to marker placement, there should be a way to easily access all the other information contained in a row. Without this, ODK Tables Maps becomes significantly less useful, as users will have to find this information in the spreadsheet themselves from specific coordinates.
- **Color rules from spreadsheet view are displayed on the map.** In the spreadsheet view of ODK Tables, customized rules determine the color of certain cells. These rules are set through three different methods: table rules, status column rules, and column rules. Table rules apply to an entire row, status column rules act like table rules, but the colors are only displayed in an always visible column on the left side of the view, and column rules apply to a singular column. These color rules should affect the color of each marker in some way.
- **Data addition through the map interface.** By using a map, users can pinpoint exact locations. Adding data through the map helps alleviate trying to find the exact coordinates for a point when they can visually see it on the map.

3. IMPLEMENTATION
This section describes the steps taken in implementing ODK Tables Maps (hereafter referred to simply as “Maps”). There were several aspects that were needed to implement the system. This included how the data displays on the map, the choice between Map APIs, how fragments hook into ODK Tables, how to leverage the already established List View, and taking advantage of different screen sizes.

![Figure 1. Demonstration of ODK Tables Maps.](image-url)
3.1. Data on Maps

In Maps, each data row is displayed as a marker, their location defined by latitude and longitude columns. These columns are automatically initialized to columns with names that contain either latitude or longitude, but they can be changed in the settings page. Color rules also apply to the markers, allowing for a clearer visualization of the data. Through the settings page again, the user determines which of the three different methods – table rules, status column rules, or column rules – to use for the markers, or none at all.

Data addition is simple. By long clicking on the map, ODK Collect will launch with the latitude and longitude fields already filled in, allowing the user to easily add data specific locations. If the user wants to add a row at their current location, they should press the + button in the ActionBar [9], which will open a blank ODK Collect form. Through ODK Collect, there is a way to select the user’s location.

![Figure 2. Data addition, from Maps to Collect.](image)

3.2. Map API

The old Maps implemented a barebones system with a map and markers. An ItemizedOverlay displayed the markers, and the markers themselves needed to subclass ItemizedItem. Even though the work was already there, the code was messy and uncommented. Google Maps also needed to use a MapActivity for it to work, which led to some poor design decisions that are discussed in 3.3. When redesigning Maps, starting in November 2012, there was a choice between two map APIs: Google Maps and Open Street Map. A fleshed out demo using Google Maps was created from the skeleton, but it felt buggy and slow. Loading markers was a pain, and there were several hacks to get the pop-ups from markers to work. After this, focus shifted towards other solutions, namely Open Street Map.

3.2.1. Open Street Map

Open Street Map is a free, open source worldwide map, created through crowd-sourcing. osmdroid [4] is a library built to interact with Open Street Map on Android by replacing Androids MapView class. The benefits for using the osmdroid library were:

- Open source and free
- Cached/downloadable tiles for offline use

However, the downsides were:

- Little documentation
- Creating markers on the map used the ItemizedOverlay approach

Maps needed to be iterated quickly, and with osmdroid, the documentation just was not there. ODK Tables is still in alpha, so the need for open source, and a completely free service (instead of the semi-free Google Maps) were nice, but not entirely needed.

3.2.2. Google Maps Android API V2

In December, 2012, Google released Google Maps Android API V2. This improved version of Google Maps added a host of options:

- Map fragments
- Built in markers
- Easy to follow documentation

There were still a few kinks, however:

- API key needed (free for limited use)
- No offline caching

The biggest positive was documentation: easy and simple to follow. Using the new API, the old Maps demo was recreated in a matter of hours, with cleaner code and faster performance.

3.2.3. Decision

Maps implements Google Maps Android API V2 instead of osmdroid for a few reasons. The documentation for Google Maps is readily available, while Open Street Map documentation is hard to find. This allowed Maps to quickly prototype and refine. In addition, the creation of markers, the backbone of the project, are easy and lightweight to use.

Google is still iterating on the new API, introducing several improvements in the recent months. One enhancement, introduced in May 2013, is the ability to change the icon of a marker without having to create a new one.

The new API also offers a way to have an alternate tile provider for the map. This means using Open Street Map, or use offline cached tiles, with Google Maps is possible.

3.2.4. Marker Problems

The new Google Maps Android API V2 is not without problems. The biggest one involve markers. There are two ways to receive a reference to a marker: when it is created or clicked. There is no way to query the map for any marker. If a developer needs a list of all the markers, they have to store it themselves, which is fine until the screen orientation switches.

In Android, whenever the orientation changes, the Activity is completed destroyed and recreated. This enables the developer to create different views for vertical and horizontal orientations. When an Activity is destroyed, it saves important state information in a Bundle. Bundles allow only primitives, some collections (like ArrayLists), and Parcelables.

Parcelables are objects that are able to be serialized, which means broken down into a condensed form to be re-instantiated later. Markers themselves are not Parcelables. However, MarkerOptions, the data behind Markers, are. When the map gets
destroyed, it saves these MarkerOptions, and upon initiation, it recreates all of the Markers.

This is effective, but the only way to get references to these new marker is if a user clicks on them. The old reference map the developer had now causes memory leaks, and does not refer to the correct markers anymore.

The way Maps gets around this is by recreating all the markers on the map when the screen changes orientation. Unfortunately, with the way Google Maps works now, this was the only solution. Fortunately, with the changes Google has already made to the API, the future looks bright towards more access to these markers.

### 3.3. Fragments

ODK Tables uses an interesting design philosophy based on previous limitations. Google MapView needed to be part of a MapActivity in order for it to work. Instead of having each view subclass MapActivity, ODK Tables defines a Controller class, which controls everything. This leads to weird intricacies, such as passing the View to the Controller at creation (onCreate), and then asking for it back when the View is being created (onCreateView).

Maps begins to take this design and simplify it. All of the Controller functionality is contained within a TableActivity, which subclasses a FragmentActivity. A FragmentActivity is an activity that contains Fragments. Instead of every view being an Activity, every view is now a Fragment.

By reverting back to the design initially planned from the start, which was foiled by the old Maps, ODK Tables gets several improvements. First, it is an easier design model to understand. Secondly, transferring all of the activities to fragments allows for multiple views to be displayed at once. For example, the spreadsheet view can be displayed next to the list view. This also allows speed increases when switching views, as fragments are merely suspended when switched.

### 3.4. Leveraging List View

ODK Tables currently contains a List View, which is defined by an easily customizable HTML file. The main advantages of using HTML files are the ability to use JavaScript and the ability to change the file without needing to recompile. Maps leverages the existing List View to help display additional data. By using the List View, snippets of data can be displayed for markers without having to view a full detailed view.

The List View is customized depending on the size of the screen. When the user is on a larger screen, it shows all of the visible markers in the view, sorted by distance from the center of the map. This is very useful for when several markers are clustered together.

If the user is on a small screen, such as a phone, then selecting a marker will show a List View as the only item. When the user touches the marker again, or any other part of the map, the List View will disappear. This gives more screen real-estate to view the map.

Communication between the List View and the Map is also possible. Defined through the HTML, clicking on the title of an item selects that item and centers the map screen on it. Clicking on the arrow leads to a detailed view of the data.

### 3.5. Designing for Different Sizes

Android is both a platform for phones and tablets. Considering the ease of using fragments and layouts, Maps displays data differently on different screen sizes. Maps automatically detects if the user is using a small, or large, screen. The check is done by checking the screen layout size of the Android device, which allows for different logic depending on the screen size.

#### 3.5.1. Phone (Small Screen)

If the screen is small, the map view gets full real estate. When the user selects a marker, the list view will appear, and will have detail only about the selected data. Touching the marker again, or touching anywhere on the map, will hide the list view. This allows users to get both a larger screen real-estate to look around, but also the info they need for markers.

![Figure 3. Maps on a small screen.](Sakabans.png)
4. PERFORMANCE

Once loaded, ODK Tables Maps acts fast and fluid. Performance of Maps is dependent on how much data is displayed. Maps add the markers when opened, which causes overhead. Minimal slowdown occurs once Maps loads.

Speed testing was performed on both the phone and tablet to find if Maps is usable for large datasets. In addition to a different amount of data points, the tests also assess if different amounts of color rules affect performance.

The tests are of the following, averaged over five trials: 10, 100, 1,000, and 10,000 rows of data, with 0, 1, and 2 color rules. The timings are for the time it took to initialize all of the markers (which was the limiting factor when loading Maps).

4.1. HTC Nexus One (Phone)

The HTC Nexus One is an older Android 2.2 phone, the lowest SDK that ODK Tables supports. Sporting a screen ranging 3.7 inches and a 480x800 pixel resolution, this device is presumingly the lowest end that will be used with ODK Tables.

Table 1. Performance Testing for HTC Nexus One.

<table>
<thead>
<tr>
<th># of rows</th>
<th>0 color rules</th>
<th>1 color rule</th>
<th>2 color rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>25 ms</td>
<td>36 ms</td>
<td>35 ms</td>
</tr>
<tr>
<td>100</td>
<td>285 ms</td>
<td>269 ms</td>
<td>246 ms</td>
</tr>
<tr>
<td>1,000</td>
<td>2,390 ms</td>
<td>2,492 ms</td>
<td>2,522 ms</td>
</tr>
<tr>
<td>10,000</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

4.2. Acer Iconia Tab (Tablet)

The Acer Iconia Tab is a newer tablet, feature Android 4.1 and a 7 inch 1024x600 display. This tablet gives a look at how maps runs on a newer Android device.

4.3. Discussion

On the phone, the addition of data points scaled linearly. Up to 1,000 data points are loaded in at a reasonable rate: around 2.5 seconds. Seemingly, color rules had little to no effect on timing; while time increased slightly with 10 and 1,000 rows, for 100 rows, adding color rules decreased the time. This means those differences were most likely not from the addition of color rules. When trying to load 10,000 data points, the phone ran into memory issues, which also occurred when trying to load the Spreadsheet View.

The results for the tablet were very similar. Just like the phone, the addition of data points scaled linearly, and at 10,000 data points, it took over 50 seconds, which is far too long for reasonable use. The color rules, yet again, did not factor into performance.
The timings for the phone and tablet were almost identical. There was no notable improvements for loading markers when using the newer device. This means that older devices will support the same amount of data as newer ones.

Once loaded, the Iconia Tab did run noticeably smoother than the Nexus One, so newer devices do provide a better viewing experience, in addition to a potentially larger, and higher pixel density, screen.

Figure 6. Run times (ms) for marker initiation tests on the Acer Iconia Tab.

Once loaded, the Iconia Tab did run noticeably smoother than the Nexus One, so newer devices do provide a better viewing experience, in addition to a potentially larger, and higher pixel density, screen.

Figure 6. Run times (ms) for marker initiation tests on the Acer Iconia Tab.

5. FURTHER WORK
There are a large number of enhancements that can be made to make Maps an even better experience. With time, and improvements to the Android SDK, the following enhancements are possible.

5.1. Clustering
Clustering allows for multiple markers near each other to be combined into one. This helps drastically reduce the number of markers on screen at once, as well as help unclutter crowded areas. Introducing clustering would allow for a quicker and cleaner experience.

However, algorithms to use clustering may perform slowly, there is the potential to confuse users if the markers are not clearly labeled, and there needs to be a sufficient way to display information through the List View.

5.2. Open Street Maps with Google Maps
As stated in the implementation section, Open Street Maps is still a viable option for use in Google Maps Android API V2. By using Open Street Maps instead of Google Maps, developers could implement a free map with no API key restrictions, in addition to potentially having more detailed areas in certain areas. Offline tiles and caching are also a reality.

5.3. Directions
Android devices allow users to locate where they are. If a user needs to get to a new location, they should be able to get directions from their current position. This would be a simple extension: by using the Google Directions API [6], these directions could be displayed directly on the map. This addition could then extend to planning routes between multiple data points.

The biggest problem for this enhancement is figuring out an intuitive way to display everything on screen.

5.4. Polygons
Unlike other views, Maps allows for a complete visualization of the data. This visualization could also extend to drawing polygons on the map. These polygons could be used to signify a data area (an area instead of a point), or to search for points inside of the polygon.

6. CONCLUSION
ODK Tables is already a flexible information services solution for a variety of use cases, but there was a need for better ways to visualize the data collected through the service. Maps provides users with a useful data visualization tool, along with other improvements, such as adding points easily at specific locations.

By using Google Maps API V2, developers have the ability to quickly expand their ideas, and even change their tile provider to Open Street Map by implementing an easy extension.

Fragments allow for more data to be displayed at a time, and simplifies the entire design of ODK Tables. Leveraging the List View in Maps is just one example of using fragments to provide a different experience.

The experimentation with different screen sizes will lead to new design choices that will provide users with different interfaces for different devices.
The consistent performance across multiple types of devices ensures that no matter the Android platform, Maps will provide a similar experience.

REFERENCES