Creating Mapping Applications for the iPhone

Kiichi Takeuchi[1,2] | kiichi.takeuchi@liu.edu
Patrick J. Kennelly[1] | patrick.kennelly@liu.edu

[1] Department of Earth and Environmental Science
C.W. Post Campus of Long Island University
720 Northern Blvd.
Brookville, NY 11548

[2] ObjectGraph LLC
315 Fifth Avenue, 10th Floor,
New York, NY 10016

INTRODUCTION

Any locational-based service (LBS) should take into account specific requirements associated with the users, locations, contexts, and data (Jiang and Yao 2006). One category of LBS gaining much notoriety and popularity in the past few years are mapping applications that run on smartphones. These small and lightweight mobile computing devices, which often have built-in global positioning systems, have ever-increasing connectivity, storage capacity, computational power, and battery life (Chang and Chen 2005).

Mobile mapping application developers should consider the following aspects of this emerging LBS. First, users are numerous and may be globally distributed. Second, the context within which mobile mapping applications are being used is increasing. Dey (2001) describes the context as "any information that can be used to characterize the situation of...a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." Third, high speed wireless networks and smartphone storage capacity ensure large quantities of data are readily accessible by the user. Such a wealth

of opportunities offers great challenges and rewards to effective mobile application design and development (Kangas and Kinnunen 2005). Beginning such design and development from scratch, however, can be a time-consuming and tedious task. To streamline design, many smartphone application developers follow a process similar to creating mashups on the Internet. Google Maps is widely used for web-based maps, as the *Application Programming Interface* (API) is available for extension. In many cases, point locations are added to the Google Maps basemap (Miller 2006). With more advanced programming, other cartographic researchers have implemented choropleth maps (Peterson 2008) and animated mashups (Roth and Ross 2009) using the Google Maps API.

Similar mapping mashups can be created for smartphones. In this article, we describe how simple mapping applications can be created or modified for the Apple iPhone. We chose the iPhone because it leads the smartphone market in number of applications and because of its ease of distribution via the Apple App Store. Apple also offers a development environment that is available free of charge. A simple iPhone application plots the user's current location on a Google Maps basemap. More interesting applications, however, plot point locations from web databases. As an example, the Mailbox Find app, created by ObjectGraph (**http://www.objectgraph.com/**) and available in the Apple iTunes Store, displays all mailboxes in the United States in proximity to the user. The 8.6 Mb of information, consisting of 178,315 points, included in the database is downloaded and stored locally on the iPhone.

TO HELP DEVELOPERS GET STARTED IN ACCESSING LOCATIONS FROM DATABASES, OBJECTGRAPH HAS LAUNCHED A PROJECT CALLED GEOGEARS, AN OPEN-SOURCE, MAP-BASED IPHONE APP PROJECT

To help developers get started in accessing locations from databases, ObjectGraph (the application development company for which the first author is a founder) has launched a project called *GeoGears*, an open-source, map-based iPhone app project. The goals of GeoGears are twofold. One is to serve as a generic application into which users without programming knowledge can add their customized geographic locations. The second, not explored in this introductory article, is to provide an open source project that can be customized by programmers.

This document is designed as a tutorial for the non-programmer to begin to create or modify iPhone/iPod Touch/iPad applications (*apps*) that include a Google Maps basemap. After listing requirements and options, the first section guides the novice through the processes of installing the iPhone *Software Development Kit* (SDK) and building a simple application with basic Google Maps functionality from scratch. The second section shows potential uses for GeoGears and how to access a project prototype. This template allows users to integrate quickly their own geospatial point data into an app without any programming knowledge. Instructions show users how to substitute the provided sample data with their own delimited text data to create a customized map.

TUTORIAL

Making iPhone Mapping Applications using GeoGears

Kiichi Takeuchi | kiichi.takeuchi@liu.edu
Patrick J. Kennelly | patrick.kennelly@liu.edu

REQUIREMENTS AND OPTIONS

Listed below are the requirements and options needed to create iPhone apps with this tutorial.

REQUIREMENTS

- Mac Laptop or Mac Desktop (a recent Intel-based CPU machine)
- Leopard (10.5) or higher MacOS
- 3GB space on the hard drive
- Apple Developer's account
- The latest iPhone SDK with XCode
- Ability to modify SQLite database (the latest FireFox Internet browser & SQLite Manager add-on)

OPTIONS

- iPhone or iPod Touch device with OS 3.0 or higher
- Geospatial data in a spreadsheet or delimited text format
- $99 for iPhone Developer License Fee

BUILDING A SIMPLE APP FROM SCRATCH

OVERVIEW

With the first five required resources listed above, this section provides step-by-step instructions for creating a simple app from scratch that plots the user's current location atop the Google Maps basemap.

SET UP A MACHINE

Your Mac should be running the Leopard or higher operating system (OS). To check your OS version, Go to the **Apple icon** in the **Menu** and click **About This Mac**. Confirm that the Mac OS X version is 10.5.8 or higher and that it utilizes an Intel processor.

If your Mac is not running a current version of the OS, click on the **Software Update** button. Check all items in the pop-up window and then click the **Install** button.

SET UP A DEVELOPER'S ACCOUNT

The next step is to set up an Apple developer's account so that you can download the iPhone SDK, both of which are free of charge. Open an Internet browser and go to the following URL: **http://developer.apple.com/iphone/**

Click the **Register** link and create your developer's account. If you already have an account with Apple, you can choose to use this information instead of creating a new account.

SET UP XCODE AND IPHONE SDK

Once you log into the *Apple Dev Center* with your user ID and password, click the **Downloads** link and download the *iPhone SDK*. Be sure to select the version that is compatible with the version of the OS you identified in the earlier step. If your MacOS is version 10.5.x, you are running Leopard; if it is version 10.6.x, you are running Snow Leopard.

Download the file on your desktop and, once the download is complete, follow the installation wizard. Install the software using the default settings. As this file is large (more than 2 GB), this process may take some time. Once installed, the application can be found under **Machintosh HD > Developer > Applications**. Double click on **Xcode.app** to start the application. You may want to create a shortcut to this application by dragging and dropping it to your desktop or your dock.

XCode is a software program used to edit program source codes and resources. It helps you to develop software for the Mac or the iPhone. iPhone SDK is an additional package of components that is necessary for iPhone development. The SDK includes a simulator to test iPhone apps on a Mac within a window that resembles an iPhone, allowing developers to see how the application will appear on an iPhone without having to load the app onto the physical device. Apple developed a high-level API called Cocoa, which is used on Mac OS X to construct the *Graphic User Interface* (GUI). Since the iPhone OS is one of a family of Mac OS, Apple integrated Cocoa into iPhone OS, calling it *Cocoa Touch*. As you can imagine from the name, it handles users' behavior on the iPhone based on Apple's touch screen technology. Cocoa Touch is tightly integrated into the SDK and XCode so that developers can handle complicated user touch actions (e.g., pinch), as standardized by Apple. *Interface Builder* (IB) is another important component of the SDK that integrates Cocoa Touch technology on XCode. The IB is designed so that each layout of an iPhone display can be modified. Simple modifications do not require programming knowledge; components simply can be added and positioned by dragging, dropping, and resizing them in the application.

CREATE A SIMPLE MAPPING APP

Open XCode and Create a Cocoa Application using the following steps:

1. Select **File** > **New Project** and choose **iPhone OS** > **Application** on the left selection menu.
2. Click **View-based Application** as your template and click the **Choose** button in the lower right corner.
3. Enter "MyMapApp" as the project name and click **Save**.
4. Now that this project is saved, if you quit XCode you can reopen the project by going to the "MyMapApp" folder and double clicking on the file "MyMapApp.xcodeproj."

You can see that some other files have been generated when completing these steps. You can browse these by looking under **Groups & Files** in the selection menu on the left (Figure 1). Files are organized under "MyMapApp" in various folders. You are now ready to add the MapKit framework to your project. This framework will provide your application with a Google Maps basemap, as well as allow it to utilize locational information and features. Begin by expanding the **Targets** menu in the selection menu on the left. Double-click "MyMapApp" found beneath the **Targets**. This will open a new window with target information for this app.

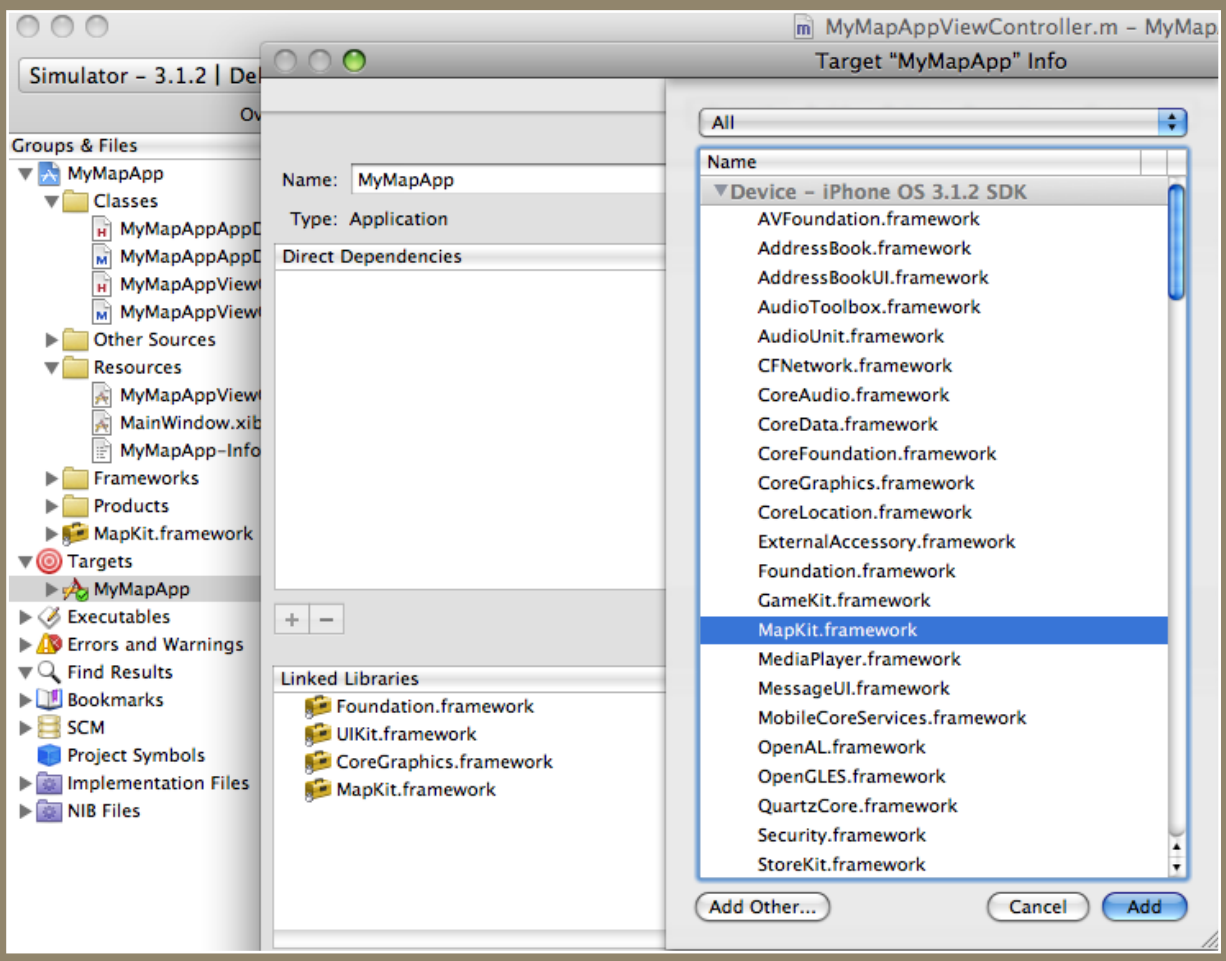

Figure 1. Three menus (see text for detail) which shows procedures to add MapKit framework into your map application.

Apple provides fundamental programming components, called frameworks (e.g., MediaPlayer, GameKit, and MapKit), to speed up the development process. Let's add the MapKit framework to our project. Click the + button in the lower left corner of this window. This provides a list of frameworks which can be incorporated into the app. Scroll down the list until you find the **MapKit.framework**, select it, and click the **Add** button. You should now see the **MapKit.framework** listed in your **Groups & Files** under "MyMapApp." This framework is now available for your developmental use.

You can add a **MapView** in your app with IB. To add the **MapView**, expand the **Resources** folder and double click "MyMapAppViewController.xib." The extension .xib indicates that this is a layout file.

The IB will open additional windows for your use. This tutorial will refer to four of these windows. It is possible, however, that some of these windows may not be visible to you upon first opening the IB. To ensure that all of the necessary windows are visible, look under the **Tools** menu and activate **Library** and **Inspector.** If **Tools > Reveal In Document Window** menu item is available, select it as well. Then double click **View** icon.

If at any point you are unable to find one of these windows, use the **Tools** menu and the instructions above to make the windows visible once again. At this point, you should be able to view all of the windows shown below in Figure 2.

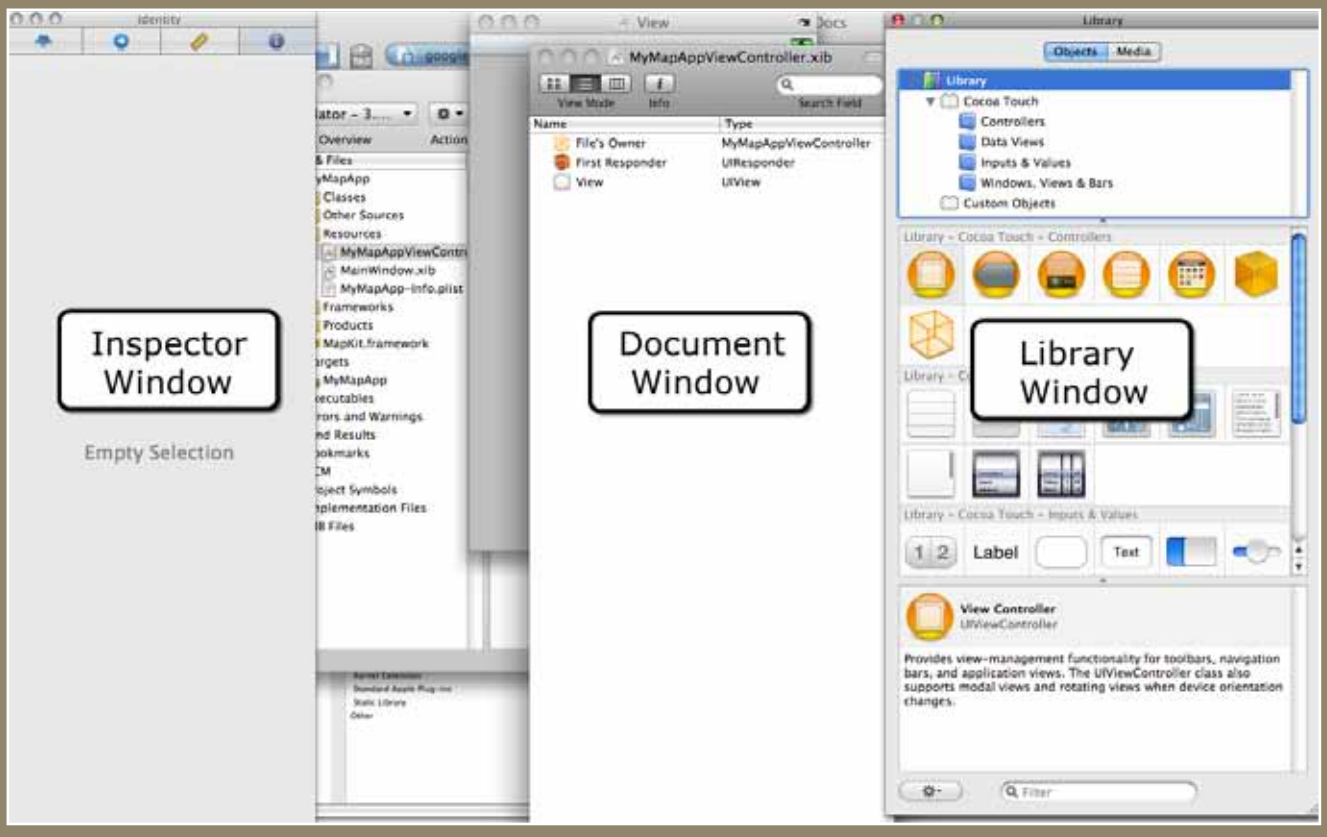

Figure 2.The three windows of Interface Builder (IB).

In the **Library** window (Figure 3), scroll down your list of objects until you see an icon for **Map View**. This is the data view that will allow you to add the Google Maps basemap and basic functionality to your app. Google provides the map data via a network connection and it downloads the images to render them on the screen. In addition, users are able to navigate the map with a multi-touch user interface, such as pinching to zoom in or out and swiping to pan. Drag and drop the **Map View** icon into **View** window. If you do not see the **View** window, go to the **top menu > Tools > Reveal in Document Window >** double-click **View**.

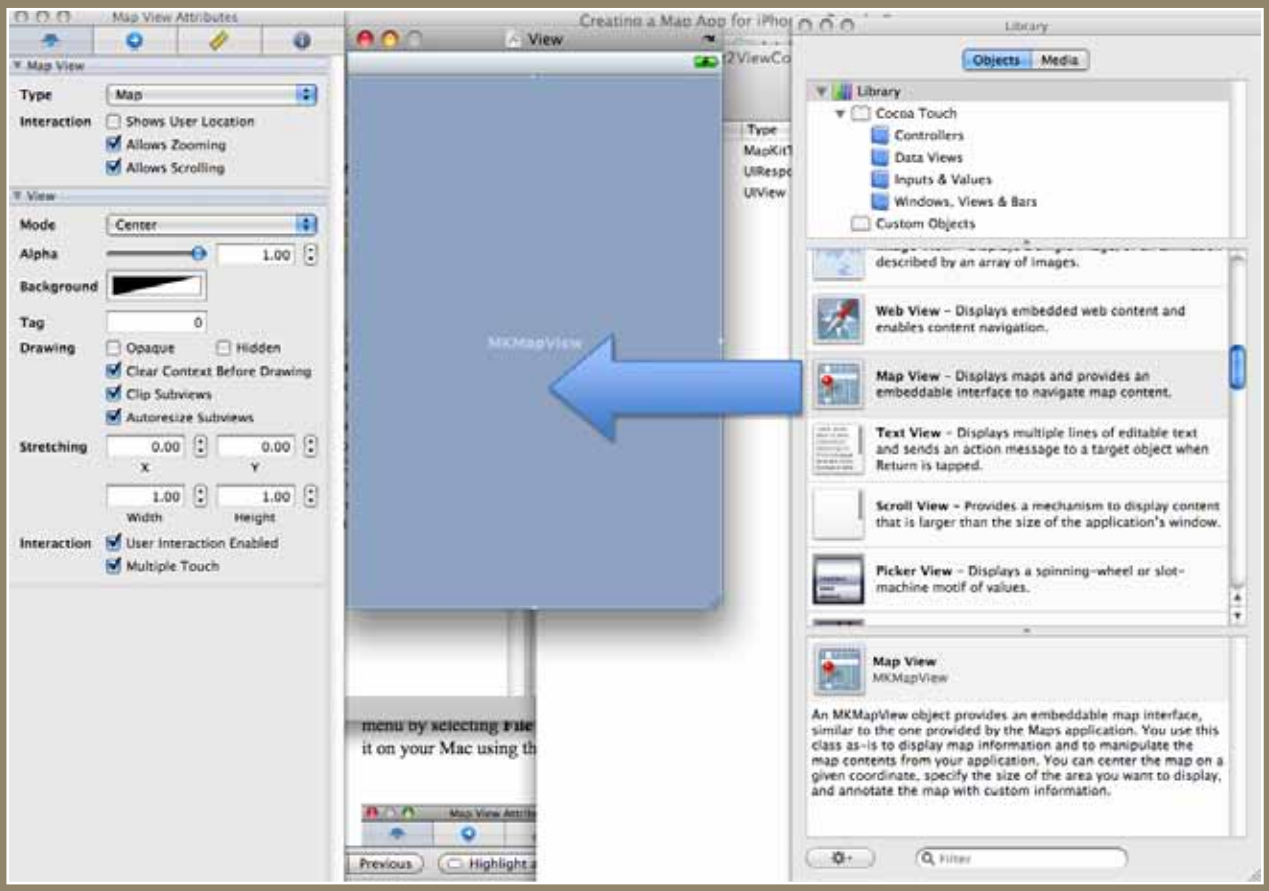

Figure 3. Adding the Map View control to the MyMapApp layout.

There is some limited functionality you can set for the map view in your app by selecting or deselecting different options. In this example, we assume you will want your app to display the current location of the user. To include this functionality, go to the **Inspector** window and click the leftmost tab. You can see the other available functions include zooming and scrolling (i.e., panning) of the map display. Check **Shows User Location** (Figure 4) and save your changes from the top menu by selecting **File > Save**. You are now ready to to build your first iPhone app and demonstrate it on your Mac using the Xcode simulator.

You may keep the IB windows open, but go back to XCode window. The **Targets** should still be set to MyMapApp. Click **Build and Go** at the top center of the XCode window.

This will build your app and initiate the simulator, which looks like an iPhone displayed on your Mac (Figure 5). If successful (it may take a minute to build and initiate), you should see the simulator window with a map and a blue dot as shown below. The blue dot is supposed to indicate your current position, but for apps running on the simulator the default location is Apple's headquarters in California. If you were to deploy your app on an iPhone, it would display your current location, using coordinates from the global positioning system (GPS) built into the iPhone.

Click and drag to scroll around the map display. Double-click to zoom in on a map location and single click (with two fingers) while holding option key to zoom out. Alternatively, you can simulate the "pinch" techniques used for maps on multi-touch devices such as the iPhone to zoom in and out. While running the simulator, hold down the option key while using a finger and thumb to pinch in or out on the touchpad. Gray circles on the map display approximate your pinching motion.

To exit this screen, press the home button at the bottom center of the simulation.

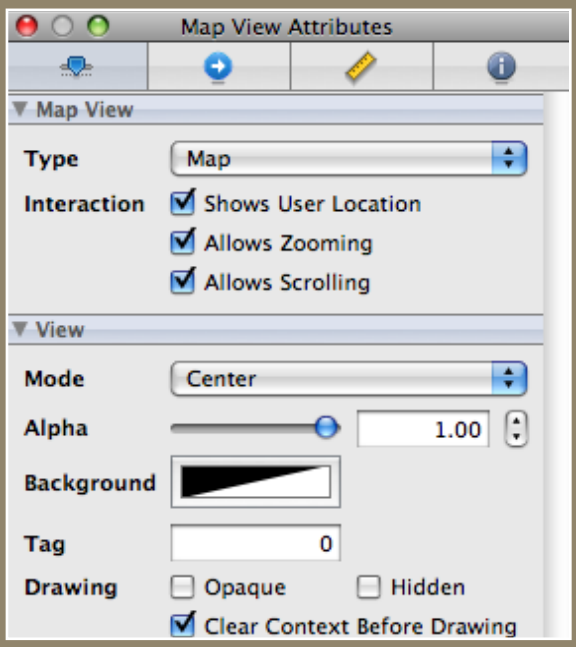

Figure 4. Options for types of interactions with the map, including an option that allows the user's location to be displayed in the MyMapApp.

CUSTOMIZING THE GEOGEARS PROTOTYPE APP

OVERVIEW

GeoGears is a prototype app that accesses spatial data from a database and plots the data on a map. It maps this data on the MapKit basemap, discussed in the previous section. It also has additional functionality built into it, such as options for mapping one of a selection of databases and an option for searching within the databases. This section discusses how to download and open the GeoGears project on a Mac, which will display sample points around the default location. Users are then instructed how to download and install a Firefox plug-in called SQLite to edit the dataset of sample points. Users are guided through steps for editing one record in the database representing one point, and then through steps for copying and pasting their own delimited text data into an empty database for creating custom maps.

DOWNLOAD GEOGEARS PROJECT

Download GeoGears from the following URL: http://www.geogearsapp.com/download/.

Figure 5. The simulator window.

Click on the "GeoGears XCode Project (The Latest Version)" link. After downloading, double click the file "GeoGears.zip" to uncompress the project into a GeoGears folder. In the GeoGears folder, double click the blue icon labeled "GeoGears.xcodeproj" and XCode will launch the project. To test the app, click the **Build and Go** icon referenced in the previous section of this tutorial. You should now see the project in the simulator.

In the search bar, you can enter an address that you would like to search (e.g, your zip code). The spatial search service is from the Google Maps geocoding service. Before proceeding to the next steps of modifying a sample database, press the **home** button and exit the simulator screen.

INSTALL SQLITE MANAGER ON FIREFOX

In this tutorial, you will be given instructions on how to modify a sample database using SQLite Manager, a plug-in for the Firefox web browser. If you do not have Firefox (or its latest version), go to http://getfirefox.com/ and install the latest version of this web browser on your Mac. You can access the SQLite Manager download by first clicking on the **Add-ons** menu on the Firefox website and performing a search for "SQLite Manager."

Find this add-on in the list and click **Add to Firefox** button. You also can find this plug-in at the following Google code website: http://code.google.com/p/sqlite-manager/. If Firefox blocks the popup menu required for installation, click the **Allow** button in the yellow dialog box. Then, click the **Install Now** button after the download is complete. The installation wizard will recommend that you restart Firefox installation; click the **Restart Firefox** button to do so. You are now ready to use SQLite Manager to modify the sample database.

Figure 6. GeoGears with sample data.

MODIFY THE SAMPLE DATABASE

Begin by ensuring the sample database is present. Go to Finder and open GeoGears folder. Navigate to the **GeoGears > data > database > Sample** sub-folder. You should find two files here: "data.sqlite" and "info.html." You will be opening and modifying the "data.sqlite" file. The "info.html" file contains information related to the database itself, including potential information on copyright or ownership. This information can be displayed on the detail page. To see the detail page, click on the **database** button in the lower right corner of the map in the simulator, then click on the blue and white arrow next to Sample listed under **Select database to use**.

Start the Firefox web browser and go to the top menu item **Tools > SQLite Manager**. Click the **Connect Database** icon, which is represented as an open yellow folder. Navigate to the sub-folder as instructed above (**GeoGears > data > database > Sample**) and select the file "data.sqlite." From the left menu, expand **Tables (2)** and select the "locations" table. On the right hand side, click **Browse & Search** tab. The display of the database in SQLite Manager should look like the image shown in Figure 7.

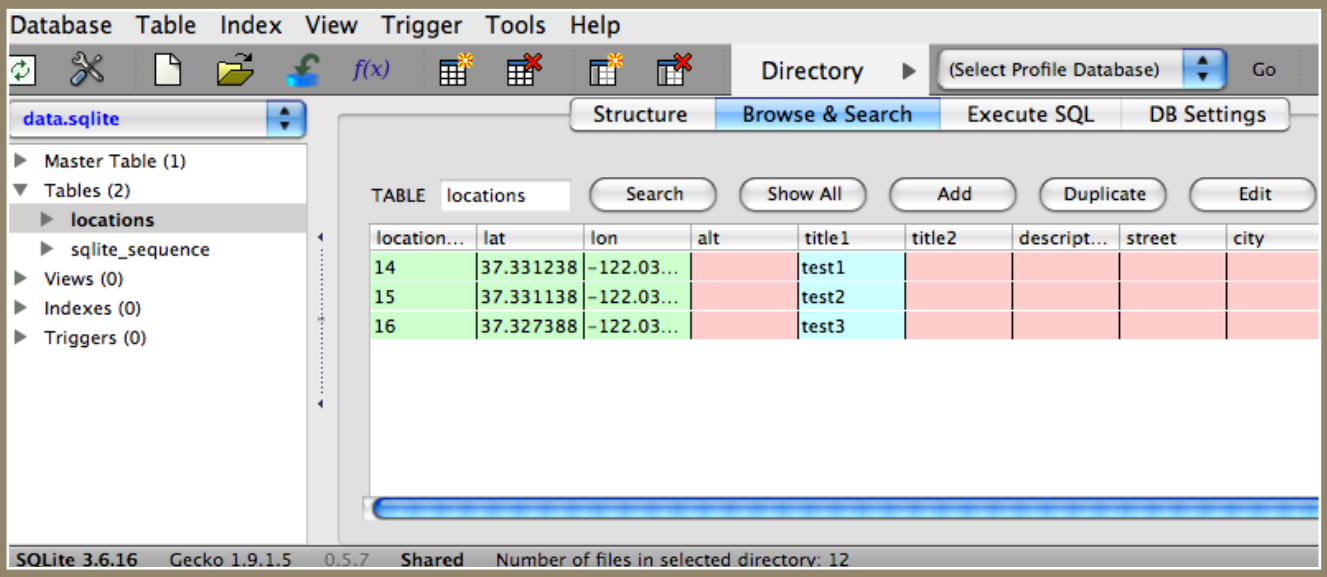

Figure 7. SQLite Manager editing screen.

You are now able to edit this database, and your edits will be displayed in the app simulator. Begin by double clicking the first row, which will open an editing screen. At this time, we will not change locations, but instead the attributes of one of the points. When these points are displayed on the map, a single click of a pin will show a display of **title1** and **title2**. Clicking on this title window will display a separate layout containing all of the other information in the database. Modify **title1**, **title2** and **description**. In the Figure 8 example, we enter **title1** as "My Pin," **title2** as "Hello World," and **description** as "This is my sample pin."

In addition to textual information, you can also change a database value to modify the color of a pin. To change the color of the pin with **title1** equal to "My Pin" from red to green, change the value of **pin_type** to **2**. Click **OK**, and you should see a message that the changes were made successfully. Your modified database display should now look similar to the image shown in Figure 8. After completing your edits, click **Cancel** to exit this display. If users have programming knowledge in Objective-C, they can make additional customizations to the color, symbol, etc.

Figure 8. A mapping app displaying customized points built with GeoGears.

PACKAGE AND INSTALL THE SAMPLE DATABASE

The GeoGears application requires that data is packaged (compressed) for efficient download. You thus are required to compress your "Sample" folder, which should now contain the modified "data.sqlite" file and the unmodified "info.html" file. Go to **Finder** and navigate to **GeoGears > data > database**. First, delete the old "Sample.zip" file, as you are creating a new one. Drag & drop "Sample.zip" from the "database" folder into the trash can. Select the "Sample" folder then right click (press the Control key and click) to display a pop-up menu. Select **Compress "Sample"** in the menu.

Your new package, "Sample.zip", is now ready. Go back to the XCode window. From the top menu, select **Build > Clean All Targets** then click

Clean to confirm. This will ensure that the old database is gone and that it is prepared for a new one to be installed. Click the **Build and Go** icon and the simulator will open in a short time.

You should now see a map of the sample data with two red pins and one green pin. The green pin is the one representing the record in the database that you modified in the steps above. Zoom in by double clicking the map until you are able to click on the green pin. You now should see "My Pin" and "Hello World" displayed in the text fields title1 and title2, which you modified. Now click on the blue and white arrow associated with this text to see the remainder of the database. This should include any modifications you made in the previous steps.

POPULATE A DATABASE WITH CUSTOM DATA

In this section, we provide steps to import a text file in comma-separated value (csv) format into a database that can be mapped by the GeoGears application. You will accomplish this by cutting and pasting your custom data into a .csv file. After completing this process and converting your file into ".sqlite" format, you can package and install the new database in the GeoGears application, as described in the previous section.

We have provided a file called "locations.csv" which can be found in the **GeoGears > data > database folder.** Open this file with Microsoft Excel or another program that can edit comma delimited text files.

The file is empty except for the first row, which is composed of the header names for each column. These names should not be changed, and the format of data you copy into this file must be consistent with the data types of each column listed below:

- location_id - (Required) Leave this column blank.
- lat - (Required) Numeric in geographic decimal degree format.
- lon - (Requied) Numeric in geographic decimal degree format.
- alt - (Optional) Numeric
- title1 - (Optional) Alpha Numeric. *Will appear when pin is clicked.*
- title2 - (Optional) Alpha Numeric. *Will appear when pin is clicked.*
- description - (Optional) Alpha Numeric. HTML format is available.
- street - (Optional) Alpha Numeric
- city - (Optional) Alpha Numeric
- state - (Optional) Alpha Numeric
- zip - (Optional) Alpha Numeric
- country - (Optional) Alpha Numeric
- pin_type - (Required) Numeric. Either 1 (red pin), 2(green pin) or 3(purple pin).

The first column, although required by the project, should be left blank so that GeoGears can assign a unique identifier to each record. You must copy in values of **lat** and **lon** in numeric geographic decimal degree format for pins to be placed properly on the basemap. As in the previous example, **title1**

and **title2** will appear when an individual pin is clicked. The remainder of the database will appear on a separate page when the blue and white arrows next to these titles are clicked.

You are now ready to convert the ".csv" file into a database in ".sqlite" format. Go to your Firefox web browser and access **Tools > SQLite Manager** from the top menu. Click the **Connect Database** icon (yellow open folder icon), as instructed previously, and reconnect to "data.sqlite." Before importing your custom data, you should right-click on the "locations" table and then select **Empty Table**. This will ensure that the data from the Sample folder accessed previously has been removed. After clearing the old sample locations, go to the **Top Menu > Database > Import**. Click the **Select File** button and select the "locations.csv" file into which you have copied your data. Ensure that "locations" is in the text box under **Enter the name of the table in which data will be imported**. Check the **First row contains column names** option.

Click the **OK** button then click **OK** on the confirmation dialog box. You should see a message that the import was successful. Upon clicking the **Browse & Search** tab, you should see your imported data in the SQLite Manager. The "data.sqlite" file in the Sample folder has now been revised.

Now that your data is saved in an ".sqlite" format, GeoGears can use it to create a map. You first will need to delete the old "Sample.zip" file, package your data into a new "Sample.zip" file, and rebuild the application, as described in section 3.5. If your data is not near the default location in the simulator, type the location of your data into the search bar. You also may want to set the search radius to a larger value depending on the extent of your point data. To reset the search radius, click on the **Gear** icon in the lower right corner of the map in the simulator and set the **slider bar** for your desired search radius.

DEPLOY THE APP ON A DEVICE

If you have an iPhone, iPod Touch, and the $99 developer's license, you can deploy this app with your custom dataset on your personal device. The most notable difference in functionality is that now the blue dot representing the current location should indeed represent your present location instead of the Apple headquarters default location used by the simulator.

If you would like to submit your new app for approval by Apple, you may do so by following instructions documented at the iPhone Development Center (http://developer.apple.com/iphone/). Specific instructions can be found in the Program Portal User Guide at the **Program Portal** menu. Approved apps are distributed exclusively through Apple's iTunes Store. In reality, most apps that are approved and frequently downloaded require significant amounts of customization beyond displaying a custom database on a Google Maps base.

CLOSING COMMENTS

The evolution of mobile mapping applications—such as smartphone applications in general and iPhone apps in particular—is likely to continue at a rapid pace. Those interested in developing such apps should consider opportunities that meet at the crossroads of two important considerations. The first consideration is how new technology can best be leveraged, the primary focus of this introductory tutorial. The second consideration is how mapping apps can best meet the geospatial needs of potential users, a critical and very open-ended question.

With tens of millions of individuals carrying smartphones, the market for useful applications that run on these devices is huge and is continuing to grow. Additionally, some distribution models for applications, like the Apple App Store, have built-in functionality for capturing user demand and feedback. In addition to documenting the number of downloads, users are encouraged to rate the app and supply feedback, which is often incorporated into future releases. This integrated system for acquiring valuable information on how users interact with maps should be of great interest to academic and professional cartographers.

A cartographer developing apps for the iPhone user is likely to want to customize numerous aspects of his or her maps. Such customizations are possible for mapping apps using the MapKit components. Our next tutorial will describe some examples of customization possible for experienced programmers, which are designed to begin making a mapping application easier to use and richer in functionality. Additionally, GeoGears is designed as an open source project which we hope programmers will find useful. For example, instead of conforming to the structure of the sample database presented in this tutorial, the code could be modified to conform to any user's database structure.

As a forum for further discussion, we have included the capacity to comment on this application on the GeoGears web page. Please feel free to leave comments or questions at: **http://www.geogearsapp.com/** (click the **# Comment(s)** link).

REFERENCES

ARTICLES

Dey, A. K. 2001. Understanding and using context. *Personal and Ubiquitous Computing* 5:4–7.

Jiang, B. and X. Yao. 2006. Location-based services and GIS in perspective. *Computers, Environment and Urban Systems* 30:712–725.

Kangas, E. and T. Kinnunen. 2005. Applying user-centered design to mobile application development. *Communications of the ACM* 48:55–59.

Miller, C. C. 2006. A beast in the field: The Google Maps mashup as GIS/2. *Cartographica* 41:187–199.

Peterson, M. 2008. Choropleth Google Maps. *Cartographic Perspectives* 60:80–83.

Roth, R. E. and K. S. Ross. 2009. Extending the Google Maps API for event animation mashups. *Cartographic Perspectives* 64:21–40.

Yung-Fu Chang and C. S. Chen. 2005. Smart phone—the choice of client platform for mobile commerce. *Computer Standards & Interfaces* 27:329–336.

WEB RESOURCES

Firefox web browser: **http://getfirefox.com/**

GeoGears Project Home: Instructions, Demo, Download and Blog: **http://geogears.objectgraph.com/**

iPhone Development Center: **http://developer.apple.com/iphone/**

SQLite Manager plug-in: **http://code.google.com/p/sqlite-manager/**