Massachusetts Institute of Technology
Fall Semester, 2011

MIT 6.S083: Building Mobile Applications

Assignment 3, Due: Monday, September 26

Maps

The previous assignments gave you practice building apps that run isolated on the phone, and also with apps that use Web services. In this assignment, you'll see how apps can use Web services to display maps - something that could be useful for projects in the coming weeks.

Part 0: Thinking about semester projects

This week, you'll learn how to create location-aware apps, and next week you'll learn about Web-based tables. At that point, you'll have enough background to begin making prototypes for significant projects. To demonstrate this, you'll do an individual project and you'll present it on October 13 as a midterm checkpoint for the course.

> *Sign up for a meeting with Hal this week or next to discuss your plans for a midterm checkpoint project and presentation. Sign up here.*

Also this week, you should register on the MAS 665 Barter system. Start meeting other people in the Thursday class, so you can be part of a team. Or listen to some of the pitches in MAS class and introduce yourself to a team where you think there could be a role for a mobile app. In any case, you should have joined a "first try" team by October 6.

> *If you are taking Founder's Journey, sign up for a meeting with Hal, just as everyone else, and we can discuss the midterm presentation and also project options.*

Part 1: Showing maps with the activity starter

As you progress through the semester, you may want to create apps that include more complex features than can be built with only blocks. One of the most straightforward ways to do this is to use an *ActivityStarter* component, which lets you take any app that's installed on the phone and run it from App Inventor as a subroutine. When you start the foreign app with ActivityStarter, it takes over the phone screen and runs. Pressing the back key returns to your App Inventor app.

The phone has a web browser. So one way to show a map is use an activity starter to open the

phone's browser to Google Maps at http://maps.google.com. This URL can take extra parameters to control the appearance and behavior of the map. You can find the details at http://mapki.com/wiki/Google_Map_Parameters. You can use your laptop browser to play with some of the capabilities documented there.

1.1 The Map Tour tutorial uses activity starter to show maps with locations in Paris. Go through the tutorial. Do only part 1, and observe how the activity starter is used. You can do this tutorial without building anything new at all, because the tutorial contains links both to the source code and to a packaged version of the app. But at least read through the explanation, and try the app, to understand how it works.

1.2 Use the app to show the location of Notre Dame Cathedral. Depending on your phone, you may get a result that you don't expect. If this happens, can you explain what went wrong? This is an instructive bug with a moral: If you relying on APIs and third-party apps, you run the risk of their behavior changing. In this case, there has been an upgrade(?) to the mapping service since the tutorial was written a year ago, which can affect newer phones. *Hint:* To understand the unexpected behavior, take a careful look at the Google Map Parameters documentation.

Part 2: Using the Static Maps API[1]

One drawback of using the activity starter is that the browser launched is totally separate from your App. It would be convenient is to have something that is a real part of your app, for example, an area that is part of your app's user interface, like an image or a canvas, that holds a web browser. This is what the WebViewer component does: Place a WebVewier on the screen, set it to visit the URL for a Web page, it will show the page. If you like, you could rebuild the Map Tour example with a WebViewer rather than an ActivityStarter.

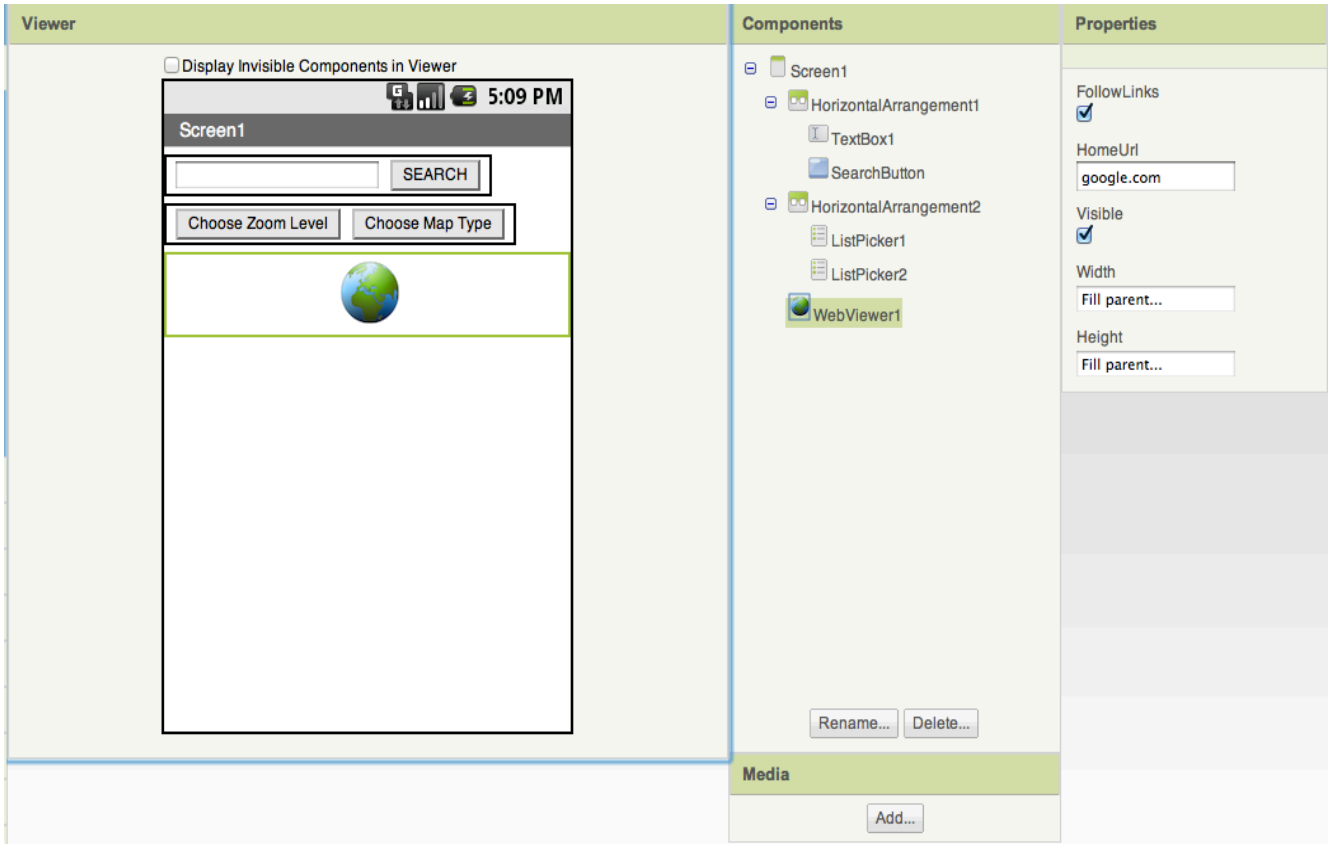
Instead, try something a different, just to see another way to handle maps. Rather than the standard Google maps page, you'll use the Google Static Maps API. This API is designed to return fixed images, rather than active, dynamic maps that expose new areas as you scroll them. But static images can be useful for lots of applications.

Here's how to build an app that displays (static) maps.

Start by making the screen layout in the designer, as shown below. The layout should include two horizontal arrangements, placed above a WebViewer component. The WebViewer component needs a home page. You can just set it to http://google.com, but the home page won't be used once you start viewing maps. The first horizontal arrangement should contain a textbox for entering the maps query, and a search button. You'll use the app by entering a

[1]This example was created by Tim Donegan <donegan@mit.edu> and Carolyn Zhang <carolynz@mit.edu>

query into the textbox and pressing the search button. The second horizontal arrangement should contain two list pickers for selecting the map Zoom Level and the Map Type.

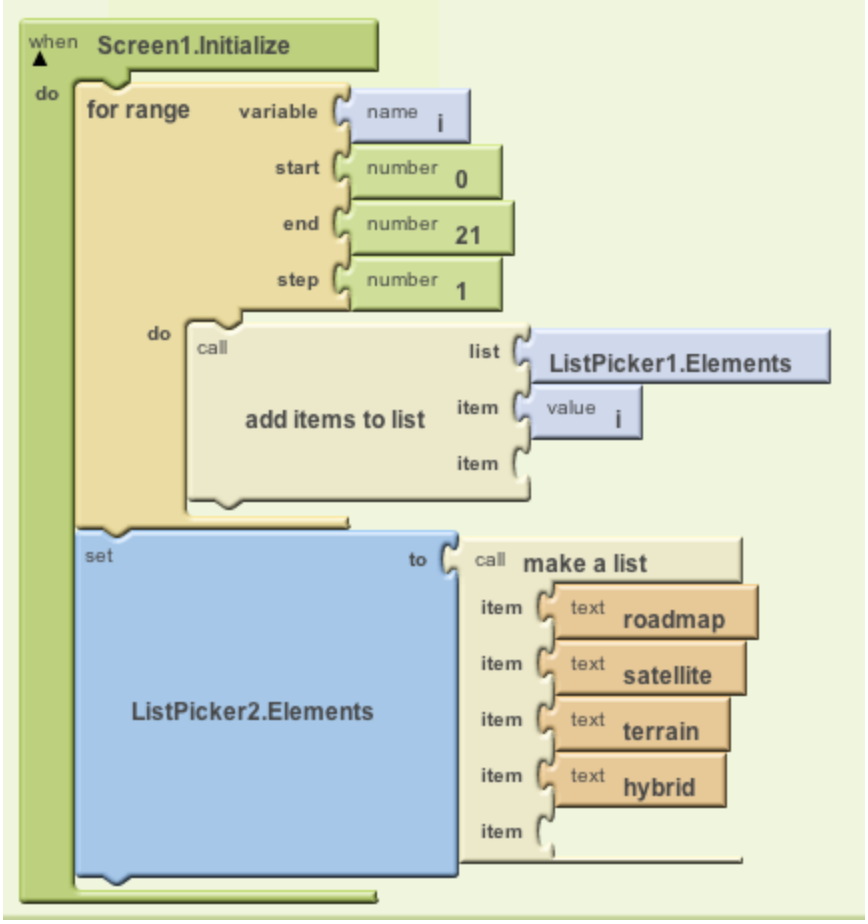


Now build the blocks:

There are three variables necessary for the app: baseURL, zoomLevel and mapType. The baseURL is the start of all static maps URLs: ***http://maps.googleapis.com/maps/api/staticmap?***. The other two variables will keep track of user's selections for map type and zoom level. Their default values should be roadmap and 10.

In the screen1.initialize event handler, initialize the list picker contents. The zoom choices will be

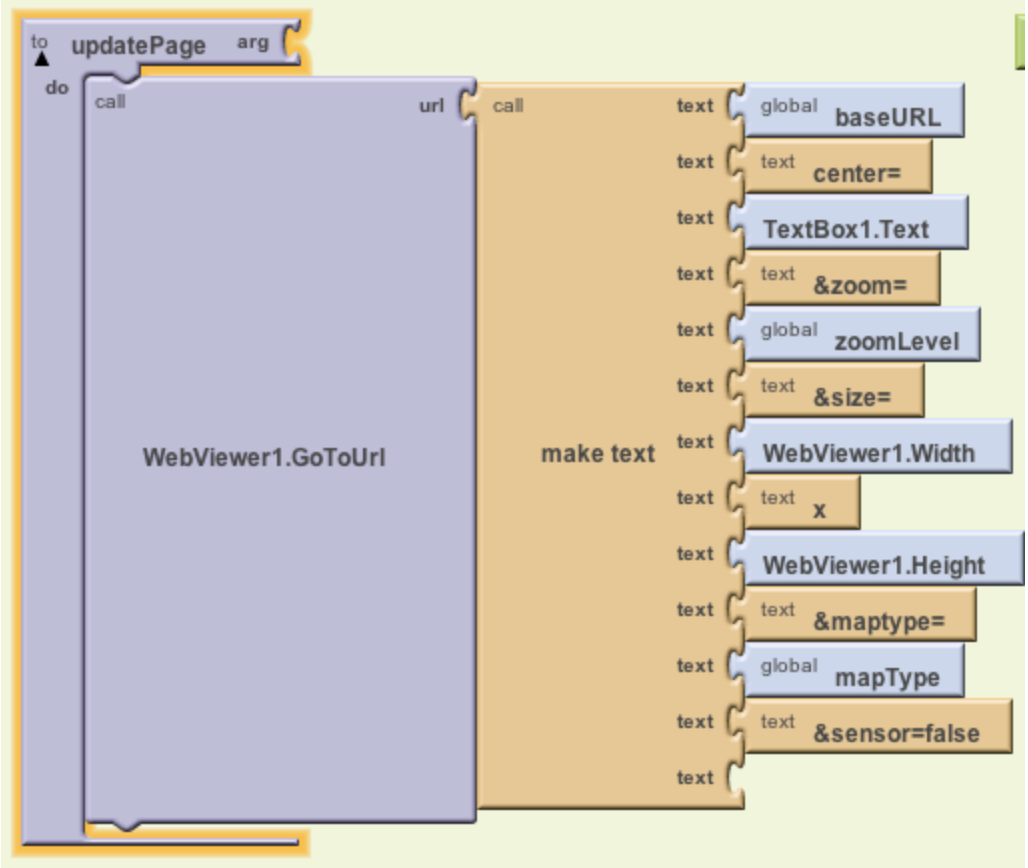
0-21, and the map types are roadmap, terrain, satellite, and hybrid.



Next, write a procedure called updatePage that constructs a new URL and sends the WebViewer to that page. The constructed URL should have the following form:

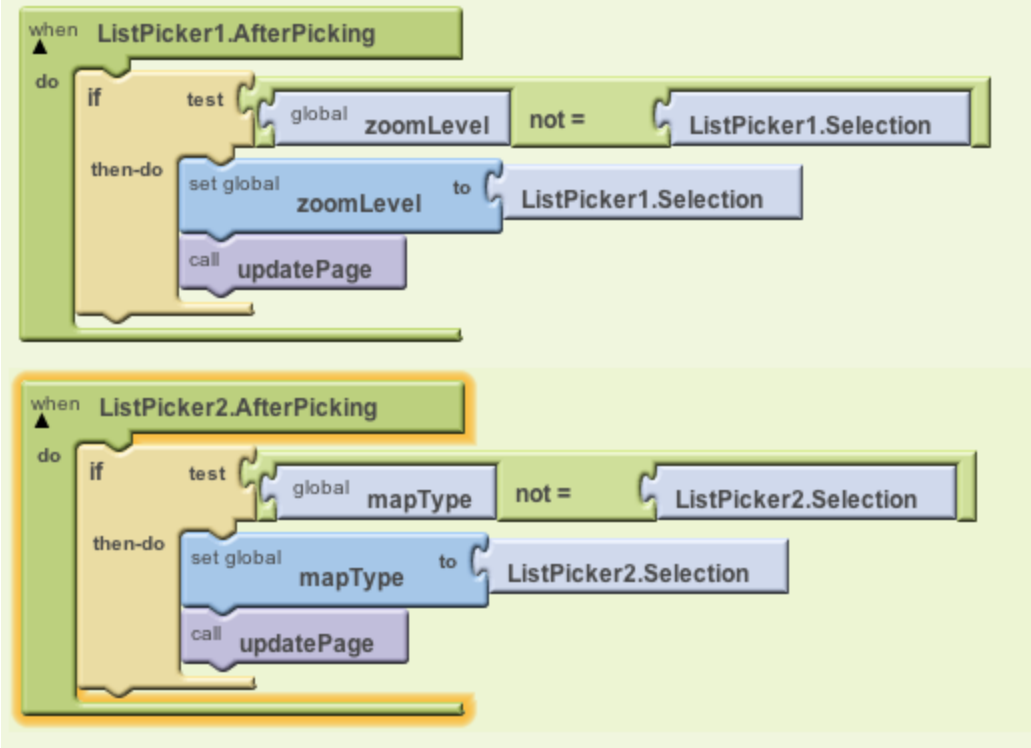
> *baseURL***&center=***textbox1.text***&zoom=***zoomLevel***&size=***WebViewer1.Width***x***WebViewer1.Height***&maptype=***mapType***&sensor=false***

The parts in bold are are literal values, the rest is stuff computed in the app. For more information on constructing these URLs, see the static maps API documentation at http://code.google.com/apis/maps/documentation/staticmaps/.



Finish the app by setting up the controls. The event handler SearchButton.click should simply call the procedure updatePage, as it assumes the user just typed something new into the textbox.

The two listpicker.afterselection event handlers should update their respective variables with the user's selection, and then call updatePage.



2.1 Implement the app as above, and try it to make sure it works. Try the query "MIT" to view MIT at various scales and with various types of maps (e.g., satellite or hybird).

2.2 Use the Static Map API http://code.google.com/apis/maps/documentation/staticmaps/ to experiment more map features. Two useful features are adding markers to maps and drawing paths between locations.

2.3 Make an original app that uses maps in some way. You've now learned enough about App Inventor that you should be able to do something interesting. As always, strive for simplicity and elegance.

Turning in your work

- Prepare a web page that shows off an interesting map you created with Static Map API in 2.2.
- Prepare a brief presentation (two web pages at most) that described the app you crated in 2.3, Include a screen shot of the app, and a brief explanation of what it does and what is interesting about it. As always, keep the explanations very short and use a large enough font so that we can display them in class. Make sure there's an obvious link to this from the main page you use for your 6.083 assignments.

Please have this page posted by Monday, September 26. We'll be viewing these pages late Monday and on Tuesday morning to select examples to show in lab Tuesday evening.