Android Programming
Lecture 11

10/7/2011
ItemizedOverlay

• Each new overlay leads to a new overlay “level” (a stack of overlays)

• Often useful to have a list of items to be presented at one “level”
  – All bus stops (WFU Bus App)
  – All job sites (City App)

• Such overlays often look the same and react similarly

• Android supports this with ItemizedOverlay
ItemizedOverlay

• ItemizedOverlay levels also manages for you:
  – Managing image resources
  – GeoPoint to Pixel conversion
  – Auto shadow drawing
  – Determining which marker in the layer was tapped

• Reasonable to use for a 1-item overlay if desire
OverlayItem

• First key idea: *OverlayItem*
  – An OverlayItem is the “data” of what is a marker
  – Constructor requires:
    • GeoPoint location – Coordinates for marker
    • String title – High level indication of marker purpose
    • String snippet – Longer description of marker purpose

  – You would want to create one of these for each different marker you wanted in this layer
    • Byrum Center, Reynolda Hall, Manchester Hall, etc.
ItemizedOverlay

- To use an ItemizedOverlay, write your own class that:
  - Extends `ItemizedOverlay<OverlayItem>` instead of `Overlay`
    - An ItemizedOverlay holds multiple `OverlayItems`
  - Define Constructor:
    - Takes a `Drawable` component as a parameter to be used as the marker image for all items
    - If using `Toasts`, also takes a `Context` variable as a parameter
    - Chooses whether to have item slightly above target point or centered on target point (`super(boundCenter(...))` or `super(boundCenterBottom(...))`)
    - Creates and adds `OverlayItems` to a list of items to be shown in the layer
    - Calls `populate()` to process all items in the list
  - Implement `public int size()` so that it returns accurate number of markers to appear in the overlay
  - Implement `OverlayItem createItem(int i)` to return a reference to the ith `OverlayItem` out of the list
  - Implement `boolean onTap(int i)` to respond to a tap on the ith item
Drawable Resources

- Under the project `res/` folder, create a new folder called `drawable`
- Enter the `drawable/` folder and ask to Import from `General, File System`
- In `Import` dialog, navigate to folder containing images to use
- Choose images of interest

- Image types: PNG (preferred), JPG, GIF
  - Turned into bitmaps before drawn
Itemized Overlay: Synthesis

- **Goal:**
  - Place star markers on Welcome Center, Library, and Benson
  - Respond to tap by presenting building name and short description
ItemizedOverlay: Synthesis

- In Activity that deals with mapping:

```java
// ITEMIZED OVERLAY CODE
// load the image of interest to use as the marker into a Drawable object
Resources resourceInformation = getResources();

// I already imported the image into my project into the resources/drawable folder
Drawable starMarker = resourceInformation.getDrawable(R.drawable.star);

// Create the overlay layer, passing the star image and the Activity context
WakeForestBuildingOverlay buildingOverlay = new WakeForestBuildingOverlay(starMarker, this);

// add this overlay layer as one of my layers
mapOverlays.add(buildingOverlay);
```
public class WakeForestBuildingOverlay extends ItemizedOverlay<OverlayItem> {
    ArrayList<OverlayItem> buildingItems;
    Context context;

    public WakeForestBuildingOverlay(Drawable defaultMarker, Context contextInput) {
        // call my parent class, and indicate I want the marker to be centered on GeoPoint for each location
        super(boundCenter(defaultMarker));

        // store context received
        context = contextInput;

        // create list of items
        buildingItems = new ArrayList<OverlayItem>(); // construct list

        // construct GPS info and overlay item for each building and add to list of items
        GeoPoint welcomeCenterLocation = new GeoPoint(36131378, -80283661);
        OverlayItem welcomeCenter = new OverlayItem(welcomeCenterLocation, "Byrum Welcome Center", "Visitor's Center/Admissions");
        buildingItems.add(welcomeCenter);

        GeoPoint bensonLocation = new GeoPoint(36132900, -80277900);
        OverlayItem bensonCenter = new OverlayItem(bensonLocation, "Benson Center", "Student Center/Dining");
        buildingItems.add(bensonCenter);

        GeoPoint libraryLocation = new GeoPoint(36132290, -80278500);
        OverlayItem library = new OverlayItem(libraryLocation, "ZSR Library", "Library/Starbucks");
        buildingItems.add(library);

        // populate the overlay
        populate();
    }
}
ItemizedOverlay: Synthesis

Remaining WakeForestBuildingOverlay functions

```java
// return the ith (requested) overlay from the list
protected OverlayItem createItem(int i) {
    return buildingItems.get(i);
}

// return number of items in the overlay (size of the list)
public int size() {
    return buildingItems.size();
}

// handle a tap on the ith item in the list
public boolean onTap(int i) {
    OverlayItem building = buildingItems.get(i);
    Toast toastToShow = Toast.makeText(context,
            building.getTitle() + "\n" + building.getSnippet(),
            Toast.LENGTH_LONG);
    toastToShow.show();
    return true;
}
```
MyLocationOverlay

• There is a built in overlay, MyLocationOverlay, which will automatically source your location from the phone and plot it on the map
  – Also supports compass direction
  – Supports re-centering of map if you move off-screen

Need GPS support first
(GPS permissions, etc) – we’ll come back to this!

– If you own an Android phone, think of the Navigation app
Location Sensors

• Most modern smartphones allow the capability of providing information about the location of the device
  – Multiple ways to obtain information
    • Coarse information: from cellular service
    • Fine information: from GPS satellite system
    • Others?
• To provide one integrated interface to these systems, Android uses a Manager-Provider design
Location Sensors

• Location Sensing is gaining personal information about the user

• Requires explicit request of permissions to access location information

• In AndroidManifest.xml, request permission for either ACCESS_FINE_LOCATION or ACCESS_COARSE_LOCATION
  – FINE implicitly includes COARSE
  – What do you really need for your app?
Location Providers

- Almost every phone supports the following two providers
  - GPS: 
    LocationManager.GPS_PROVIDER
  - Cell Network: 
    LocationManager.NETWORK_PROVIDER

- Right now, assume we will use GPS
  - We will come back to a way to search for a provider
Location Information

• There are a couple of ways we can get at Location Information

  – Request only the last known location
    • May be (very) out of date

  – Request to listen to location updates
    Can limit the rate at which these arrive:
    Only every X amount of time
    Only every Y distance from current location
Location Information

What type of location information do you get?

A `Location` object:

<table>
<thead>
<tr>
<th>Type</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td><code>getAccuracy()</code></td>
<td>Returns the accuracy of the fix in meters.</td>
</tr>
<tr>
<td>double</td>
<td><code>getAltitude()</code></td>
<td>Returns the altitude of this fix.</td>
</tr>
<tr>
<td>float</td>
<td><code>getBearing()</code></td>
<td>Returns the direction of travel in degrees East of true North.</td>
</tr>
<tr>
<td>Bundle</td>
<td><code>getExtras()</code></td>
<td>Returns additional provider-specific information about the location fix as a Bundle.</td>
</tr>
<tr>
<td>double</td>
<td><code>getLatitude()</code></td>
<td>Returns the latitude of this fix.</td>
</tr>
<tr>
<td>double</td>
<td><code>getLongitude()</code></td>
<td>Returns the longitude of this fix.</td>
</tr>
<tr>
<td>String</td>
<td><code>getProvider()</code></td>
<td>Returns the name of the provider that generated this fix, or null if it is not associated with a provider.</td>
</tr>
<tr>
<td>float</td>
<td><code>getSpeed()</code></td>
<td>Returns the speed of the device over ground in meters/second.</td>
</tr>
<tr>
<td>long</td>
<td><code>getTime()</code></td>
<td>Returns the UTC time of this fix, in milliseconds since January 1, 1970.</td>
</tr>
</tbody>
</table>

http://developer.android.com/reference/android/location/Location.html
Getting Location Information

• Getting a reference to the LocationManager:
  ```java
  LocationManager locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
  ```

• Getting last known location:
  ```java
  Location location = locationManager.getLastKnownLocation(String providerStringGoesHere)
  ```

For our examples (asking directly for GPS)
  ```java
  Location location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
  ```
Last Known Location Example

LocationManager locationManager;

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    lastKnownButton = (Button) findViewById(R.id.last_known_button);
    lastKnownButton.setOnClickListener(this);
    textView = (TextView) findViewById(R.id.text_view);
    mapView = (MapView) findViewById(R.id.map_view);
    mapView.setSatellite(true);
    mapView.setBuiltInZoomControls(true);
    mapController = mapView.getController();

    locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
}

public void onClick(View view) {
    if (view.getId() == R.id.last_known_button) {
        Location location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
        if (location != null) {
            textView.setText("" + location.getLatitude() + "," + location.getLongitude());
            GeoPoint targetPoint = convertLatAndLongToGeoPoint(location.getLatitude(), location.getLongitude());
            mapController.setCenter(targetPoint);
            mapController.setZoom(18);
        } else Log.v("WakeLocator", "NULL LOCATION!");
    }

public GeoPoint convertLatAndLongToGeoPoint(double latitude, double longitude) {
    return new GeoPoint(new Double(latitude*1E6).intValue(), new Double(longitude*1E6).intValue());
}
Setting Locations in the Emulator

- Open the DDMS Perspective
- Provide a manual GPS update
- If just opened Emulator, open the Google Maps app on the Emulator so it “gets a first location fix” [a last update]
- Test your app

Emulator doesn’t support actual GPS or cell network access, so have to fake it.
Listening for Location Updates

• Your Activity needs to implement the `LocationListener` interface

• Four functions to complete:
  – `public void onLocationChanged(Location location)`
    Handle a position update
  – `public void onProviderDisabled(String provider)`
    Handle a loss of location source (physically turned off)
  – `public void onProviderEnabled(String provider)`
    Handle a gain of location source (physically turned on)
  – `public void onStatusChanged(String provider, int status, Bundle extras)`
    Handle the location source being turned off or on in a temporary way (i.e. going through a tunnel)
Listening for Location Updates

• The status received in `onStatusChanged` could be:
  – `LocationProvider.OUT_OF_SERVICE` - lost and expected to stay off
  – `LocationProvider.TEMPORARILY_UNAVAILABLE` - lost but expected to come back soon
  – `LocationProvider.AVAILABLE` - available to use

• One needs to register that the Activity is a listener for Location events
  – `locationManagerHere.requestLocationUpdates(String provider, int timeInMillisecondsBetweenUpdates, int distanceInMetersBetweenUpdates, listenerGoesHere);`
Listening for Location Updates Example

- Requesting to listen to updates (in `onCreate`)

```java
locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 5000, 5, this);
```

- Implementing needed `LocationListener` functions

```java
public void onLocationChanged(Location location) {
    textView.setText("Update: "+location.getLatitude() + "," + location.getLongitude());
    GeoPoint targetPoint = convertLatAndLongToGeoPoint(location.getLatitude(), location.getLongitude());
    mapController.setCenter(targetPoint);
    mapController.setZoom(19);
}
public void onProviderDisabled(String provider) {
    textView.setText("Location source unavailable...");
}
public void onProviderEnabled(String provider) {
    textView.setText("Location source available..., wait for update");
}
public void onStatusChanged(String provider, int status, Bundle extras) {
    if (status == LocationProvider.OUT_OF_SERVICE)
        textView.setText("Location source unavailable...");
    else if (status == LocationProvider.TEMPORARILY_UNAVAILABLE)
        textView.setText("Location source temporarily unavailable...");
    else if (status == LocationProvider.AVAILABLE)
        textView.setText("Location source available..., wait for update");
}
Location Update Costs

- If want consistent updates, can have 0 time and 0 distance set in `requestLocationUpdates` function.
- Location updating is very expensive in terms of power, especially GPS, so consider setting those variables to reasonable values.
- Can also choose location provider source based on cost and capability criteria.
MyLocationOverlay

- There is a built-in overlay, `MyLocationOverlay`, which will automatically source your location from the phone and plot it on the map
  - Also supports compass direction
  - Supports re-centering of map if you move off-screen

- Easy to use:

  ```java
  MyLocationOverlay myLocationOverlay = new MyLocationOverlay(Context context, MapView mapView)
  myLocationOverlay.enableCompass();
  myLocationOverlay.enableMyLocation();
  // above enables tracking off screen
  overlays.add(myLocationOverlay); // overlays from map (we saw // how to get this overlays list already)```
MyLocationOverlay Example

```java
locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);

List<Overlay> overlays = mapView.getOverlays();
MyLocationOverlay myLocationOverlay = new MyLocationOverlay(this, mapView);
myLocationOverlay.enableCompass();
myLocationOverlay.enableMyLocation();
overlays.add(myLocationOverlay);
```

The Overlay listens for location updates as well, automatically, moving the blue dot and compass bearing.

My listener is updating the center of the map and the TextView (Update: latitude, longitude)