CS371m - Mobile Computing

Maps
Using Google Maps

• Content on using Google Maps inside your app

• Alternatives Exist:
  – Open Street Maps

• If you simply want to display a "standard Google map" from your app
String uri = String.format(Locale.ENGLISH, "geo:%f,%f", latitude, longitude);
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(uri));
context.startActivity(intent);
Setting Zoom Level and Label

• Specify Zoom Level for standard Map
• Zoom Levels in Google Maps, 0 - 21
  – 0 is the whole earth
  – 21 individual buildings

// with label and given zoom
String label = "GDC";
String uriBegin = "geo:" + latitude + ""," + longitude;
String query = latitude + ""," + longitude + "(" + label + ")";
String encodedQuery = Uri.encode(query);
String uriString = uriBegin + "?q=" + encodedQuery + "&z=18";
Uri uri = Uri.parse(uriString);
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
this.startActivity(intent);
Using Google Maps

• Requires an API key from Google

• [https://developers.google.com/maps/documentation/android/](https://developers.google.com/maps/documentation/android/)

• required to use MapView class or MapFragments

• Must:
  – Register the SHA-1 fingerprint of the certificate used to sign the application.
  – Adding a reference to the Maps API Key in each MapView (xml or code)
INCORPORATING MAPS IN YOU APP
Using Google Maps API v2

• Google Maps API v2 part of the *Google Play Services* sdk
• Download via SDK Manager
Include Google Play Services in Manifest

- To make use of Google Play Services add data to manifest
- Google Play Services has a host of non standard android tools
  - "simple location API"
  - "activity recognition"

```xml
<meta-data
  android:name="com.google.android.gms.version"
  android:value="@integer/google_play_services_version" />
```
Obtaining an API Key

• Most web APIs require a key to use
  – a few do not such as the Yahoo finance api
• Same with Google Maps API
• New way of obtaining keys via Android Studio and Google Developers console is mostly painless
• Old way of obtaining the key required some knowledge regarding how apps are published and painful
New Way to Get Maps API Key

- Create a Google Maps Activity in Android Studio
New Way to Get Maps API Key

• Look at google_maps_api.xml file in new project

```xml
<:--
TODO: Before you run your application, you need a Google Maps API key.
To get one, follow this link, follow the directions and press "Create" at
https://console.developers.google.com/flows/enableapi?apiid=maps_android
```
Go to Developer Console

Register your application for Google Maps Android API in Google Developers Console

Google Developers Console allows you to manage your application and monitor API usage.

Select a project where your application will be registered
You can use one project to manage all of your applications, or you can create a different project for each application.

Create a new project

Continue
Create Key

API Manager

Credentials

Create Android API key

Name

| Android key 1 |

Restrict usage to your Android apps (Optional)

Android devices send API requests directly to Google. Google verifies that each request comes from an Android app that matches a package name and SHA-1 signing-fingerprint name that you provide. Get the package name from your AndroidManifest.xml file. Use the following command to get the fingerprint. [Learn more](#)

```
keytool -list -v -keystore mystore.keystore
```

API key

Here is your API key

AIzaSyDWtuMQOMPTWWAajQz27BaUs

OK
Last Step

• Copy API Key into google_maps_api.xml file

Once you have your key (it starts with "AIza"), replace the string in this file.

```xml
<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">YOUR_KEY_HERE</string>
```
OLD PAINFUL WAY
Signing Apps

• deploying apps on the Google Play requires signing the app with a certificate

• development and debugging uses an automatic key creation process
  – invisible to us

• In release mode you must create your own private key to sign apps
  – use of keytool program from Java SDK

Signing Apps

• A Java Keystore is a file (or files) that stores security certificates
• Included in the JDK (Java Development Kit) is the keytool program
• Used to create manipulate the keystore
Signing Apps via Android Studio

- Android Studio provides a GUI to run keytool for you
- Build -> Generate Signed APK
Obtaining an API Key

• For Google Maps API v2
• One key tied to one signing certificate
• Same key used for all instances of app
• Normally sign apps with different certificates
• If so different API keys required if two different apps use maps and signed with different certificates
Obtaining an API Key

• To obtain a Google Maps API key we need the SHA-1 fingerprint of the signing certificate
• A short form of the certificate based on the SHA-1 hashing algorithm
• run keytool from the command line to pull out fingerprint of certificate
fingerprint via keytool

- prompt>keytool -v -keystore <keystore_file_name> -alias <certificate_name> -storepass <keystore_password> -keypass <certificate_password>

```
C:\Users\scottm>keytool -list -v -keystore android_keystore.jks -alias sample_certificate
Alias name: sample_certificate
Creation date: Feb 27, 2015
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Michael Scott, OU=Computer Science Department, O=University of
Issuer: CN=Michael Scott, OU=Computer Science Department, O=University of
Serial number: 6e5e9bb2
Certificate fingerprints:
Signature algorithm name: SHA256withRSA
Version: 3
```
debug certificate

• In development you are using a pre-generated debug keystore to sign apps
• Happens behind the scenes
• Security settings on device
debug certificate

- possible to obtain API key tied to your debug keystore
- works in development
- would need to change manifest with certificate used to sign the app
Obtaining the API key

• The SHA-1 fingerprint is a 20 digit hexadecimal number

• Use Google APIs console to obtain key for Maps
  – requires Google account

• ... and must agree to the terms of service.
Google Maps Terms of Service

• Some Highlights

Android Maps APIs Terms of Service

Last Updated: October 13, 2008

Thanks for your interest in the Android Maps APIs. The Android Maps APIs are a collection of services (including, but not limited to, the "com.google.android.maps.MapView" and "android.location.Geocoder" classes) that allow you to include maps, geocoding, and other content from Google and its content providers in your Android applications. The Android Maps APIs explicitly do not include any driving directions data or local search data that may be owned or licensed by Google.

– may include ads in future
– Google may limit number of transactions
– Cannot use for turn-by-turn directions or autonomous driving
– suppose to show license info in the app!
Using Maps API Key in App

• Must add key to manifest

```xml
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="API_KEY"/>
```

• KEY POINT: Replace API_KEY in the second line only (android:value) with the API key you obtained in previous steps
DISPLAYING A MAP INSIDE YOUR APP
Permissions

• Recommended Permissions for manifest when using Google Maps inside your app

```xml
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```
Display Simple Map in App

• Hello Map

• Like Hello World, but layout file becomes:

```xml
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name="com.google.android.gms.maps.MapFragment"/>
```
Hello Map Activity

• Lots of new classes!!

```java
import android.app.Activity;
import android.os.Bundle;

import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
```
Specifying Locations

• Latitude and Longitude

```java
public class SimpleMapActivity extends Activity {

    static final LatLng AUSTIN = new LatLng(30.287, -97.737);
    static final LatLng ARLINGTON = new LatLng(32.751, -97.083);
    private GoogleMap map;
```
onCreate

```java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_simple_map);
    map = ((MapFragment) getSupportFragmentManager().findFragmentById(R.id.map)).getMap();
}
```

• More on Fragments later ...
  – mini activities
adding Markers in onCreate

• first Marker uses default, pin and has a title, "Austin"

```java
if (map != null) {
    Marker austin
        = map.addMarker(new MarkerOptions().position(AUSTIN)
            .title("Austin"));

    Marker arlington = map.addMarker(new MarkerOptions() 
        .position(ARLINGTON) 
        .title("Arlington") 
        .snippet("Play Ball!!") 
        .icon(BitmapDescriptorFactory 
            .fromResource(R.drawable.ic_launcher)));
```

• second Marker uses a different icon and adds text after the title
Center and Zoom

• Running app as is produces this:
• Centered where???
• Zoomed out
• Zoom levels 0 to 21
```javascript
CameraUpdate center = CameraUpdateFactory.newLatLng(AUSTIN);
CameraUpdate zoom = CameraUpdateFactory.zoomTo(5);

map.moveCamera(center);
map.animateCamera(zoom);
```
Debug Key

- Portion of debug.keystore
Getting MD5 Fingerprint

• use keytool program
• keytool part of Java SDK
• keytool -list -alias androiddebugkey -keystore <path_to_debug_keystore>.keystore -storepass android -keypass android
• gives MD5 fingerprint of the debug certificate
• keytool of Java 1.7 gives SHA1 by default — use -v after keytool, before -list
Debug API Key

Thank you for signing up for an Android Maps API key!

Your key is:

0ce0zsn6afbLp8R8ZR1_9rizJTjrJIoa4w_VkCQ

This key is good for all apps signed with your certificate whose fingerprint is:

DF:6E:BD 0C:98:B3

Here is an example xml layout to get you started on your way to mapping glory:

```xml
<com.google.android.mapsMapView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:apiKey="0ce0zsn6afbLp8R8ZR1_9rizJTjrJIoa4w_VkCQ"
/>
```
Hello MapView

• Build Target - Google, not Android

• MapView not a standard Android class
  – part of Google Maps Library
  – add to manifest

```xml
<uses-library android:name="com.google.android.maps" />
```

• must also include INTERNET permission and LOCATION permission
## Aside - Permissions


<table>
<thead>
<tr>
<th>Permission Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS_CHECKIN_PROPERTIES</td>
<td>Allows read/write access to the &quot;properties&quot; table in the checkin database</td>
</tr>
<tr>
<td>ACCESS_COARSE_LOCATION</td>
<td>Allows an application to access coarse (e.g., Cell-ID, WiFi) location</td>
</tr>
<tr>
<td>ACCESS_FINE_LOCATION</td>
<td>Allows an application to access fine (e.g., GPS) location</td>
</tr>
<tr>
<td>ACCESS_LOCATION_EXTRA_COMMANDS</td>
<td>Allows an application to access extra location provider commands</td>
</tr>
<tr>
<td>ACCESS_MOCK_LOCATION</td>
<td>Allows an application to create mock location providers for testing</td>
</tr>
<tr>
<td>ACCESS_NETWORK_STATE</td>
<td>Allows applications to access information about networks</td>
</tr>
<tr>
<td>ACCESS_SURFACE_FLINGER</td>
<td>Allows an application to use SurfaceFlinger's low level features</td>
</tr>
<tr>
<td>ACCESS_WIFI_STATE</td>
<td>Allows applications to access information about Wi-Fi networks</td>
</tr>
<tr>
<td>ACCOUNT_MANAGER</td>
<td>Allows applications to call into AccountAuthenticators.</td>
</tr>
<tr>
<td>ADD_VOICEMAIL</td>
<td>Allows an application to add voicemails into the system.</td>
</tr>
<tr>
<td>AUTHENTICATE_ACCOUNTS</td>
<td>Allows an application to act as an AccountAuthenticator for the AccountManager</td>
</tr>
</tbody>
</table>
MapView

• A type of view for layout file

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mainlayout"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <com.google.android.maps.MapView
        android:id="@+id/mapview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:clickable="true"
        android:apiKey="Your Maps API Key"
    />

</RelativeLayout>
```
MapActivity

• Create class that extends MapActivity instead of Activity
• import com.google.android.maps.MapActivity;
• must implement isRouteDisplayed method

```java
@Override
protected boolean isRouteDisplayed() {
    return false;
}
```

• must return true if any kind of route (to be followed) is displayed, per terms of use
Instance Vars and onCreate

- Add instance variables and initialize in onCreate method

```java
private LinearLayout linearLayout;
private MapView mapView;

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mapView = (MapView) findViewById(R.id.mapview);
    mapView.setBuiltInZoomControls(true);
}
```
HelloMapView

- Run app
- Displays map and allows panning and zooming
Customizing Map

• Easy to display map and allow interaction
• Customize with markers and overlays
• Overlays
  – used to display information on top of map
  – simple choice: ItemizedOverlay class
public class HelloItemizedOverlay extends ItemizedOverlay {

    private ArrayList<OverlayItem> mOverlays;

    public HelloItemizedOverlay(Drawable defaultMarker) {
        super(boundCenterBottom(defaultMarker));
        mOverlays = new ArrayList<>(0);
    }

    public void addOverlay(OverlayItem overlay) {
        mOverlays.add(overlay);
        // inherited method to prepare overlays to be drawn populate();
    }
}
ItemizedOverlay

- populate method will call createItem
- define createItem and return value from the ArrayList instance var
- define size method that returns number of overlay items

```
@Override
protected OverlayItem createItem(int i) {
    return mOverlays.get(i);
}

@Override
public int size() {
    return mOverlays.size();
}
```
Adding Overlays

• In MapActivity create OverlayItem
• add to HelloItemizedOverlay
• add to MapView
• Need a drawable for the marker
  – res/drawable
  – issues display gif format images on some devices
Changes to HelloMapView

```java
private LinearLayout linearLayout;
private MapView mapView;
private List<Overlay> mapOverlays;
private Drawable drawable;
private HelloItemizedOverlay itemizedOverlay;

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);
    mapView = (MapView) findViewById(R.id.mapview);
    mapView.setBuiltInZoomControls(true);

    mapOverlays = mapView.getOverlays();
    drawable = this.getResources().getDrawable(R.drawable.Longhorn);
    itemizedOverlay = new HelloItemizedOverlay(drawable);
}
```
Add Overlay Items

• Create GeoPoint and use these to create OverlayItems
• GeoPoint based on microdegrees
  – lat and long times 1,000,000
• Build OverlayItems out of GeoPoints and include strings for title and snippet text to display when drawable clicked
private void addOverlays() {
    int[] parlin = {(int) (30.284882 * 1e6), (int) (-97.740127 * 1e6)};
    int[] mcdonald = {(int) (30.671581 * 1e6), (int) (-104.022431 * 1e6)};
    int[] marine = {(int) (27.835926 * 1e6), (int) (-97.050372 * 1e6)};
    int[][][] points = {parlin, mcdonald, marine};

    String[][][] titlesAndSnippets = {{"UT",
        "Parlin Hall at The University of Texas at Austin"},
        {"McDonald",
            "McDonald Observatory - University of Texas - West Texas"},
        {"Marine Biology",
            "University of Texas Marine Science Institute - Port Aransas"}};

    for(int i = 0; i < points.length; i++) {
        GeoPoint g = new GeoPoint(points[i][0], points[i][1]);
        OverlayItem oi = new OverlayItem(g,
            titlesAndSnippets[i][0], titlesAndSnippets[i][1]);
        itemizedOverlay.addOverlay(oi);
    }

    mapOverlays.add(itemizedOverlay);
}
Result

- one overlay with multiple items
- based on locations we added
Display Information

• To display information (title and snippet) of overlay override the onTap method in the ItemizedOverlay class

```java
@Override
protected boolean onTap(int index) {
    OverlayItem item = mOverlays.get(index);
    AlertDialog.Builder dialog = new AlertDialog.Builder(mContext);
    dialog.setTitle(item.getTitle());
    dialog.setMessage(item.getSnippet());
    dialog.show();
    return true;
}
```
Results of Clicking Longhorn

Marine Biology
University of Texas Marine Science Institute - Port Aransas

UT
Parlin Hall at The University of Texas at Austin
Reverse Geocoding

• Find addresses from longitude/latitude
• Geocoder uses a backend that is NOT included in the core android framework
• use isPresent method to check for service

```java
location = locationManager getLastKnownLocation(LocationManager.GPS_PROVIDER);
double lat = location.getLatitude();
double lng = location.getLongitude();

Geocoder gc = new Geocoder(this, Locale.getDefault());
List<Address> addresses = null;
try {
    addresses = gc.getFromLocation(lat, lng, 5); // maxResults
} catch (IOException e) {} 
```
Forward Geocoding

- Find longitude/latitude (and more) from address or airport code

```java
Geocoder gc = new Geocoder(this, Locale.US);
List<Address> addresses = null;
try {
    addresses = gc.getFromLocationName("713 N. Duchese, St., Missouri", 5);
} catch (IOException e) {}

double lat = addresses.get(0).getLatitude();
double lng = addresses.get(0).getLongitude();
String zip = addresses.get(0).getPostalCode();
```

HelloMapView Geocode is present: true
HelloMapView lat: 38.7991079, long: -90.494416, zip: 63301
Recent Changes

• Google Maps API version 2
  – somewhat new, Released December 2012 as part of Google Play Services SDK

• features:
  – indoor maps
  – simplified location services
Maps Example

• Route Tracker using Locations, MapActivity, MapView, and Google Maps
  – from Deitel AFP-AADA

• Similar to Map My Ride
  – popular app among cyclists and runners
RouteTracker App

Distance and Average Speed

Distance: 0.1KM 0.1MI
Average speed: 2.9KPH 1.8MPH

OK
RouteTracker App

• using FrameLayout to stack components with the most recently added component on top
• ToggleButton at bottom to start and stop route tracking
• MapView added to FrameLayout
• route is an overlay to map with points and lines connecting points
RouteTracker Classes

**RouteTracker**
Starting Activity
deals with LocationProvider

**BearingFrameLayout**
Displays MapView
rotates based on bearing from location

**RouteOverlay**
Overlay with location points (every 10th) and lines connecting.
Converts locations to GeoPoints.
Overloads draw
Criteria Class

• Set criteria for selecting a LocationProvider

```java
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_FINE);
criteria.setBearingRequired(true);
criteria.setCostAllowed(true);
criteria.setPowerRequirement(Criteria.POWER_LOW);
criteria.setAltitudeRequired(false);

locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);

// get the best provider based on our Criteria
String provider = locationManager.getBestProvider(criteria, true);
```
GpsStatus.Listener

- Responds to changes in GPS status
- Are we receiving GPS fixes?
- App does not track unless this is true

```java
// determine whether we have GPS fix
GpsStatus.Listener gpsStatusListener = new GpsStatus.Listener() {
    public void onGpsStatusChanged(int event) {
        if (event == GpsStatus.GPS_EVENT_FIRST_FIX) {
            gpsFix = true;
            Toast results = Toast.makeText(RouteTracker.this,
                getResources().getString(R.string.toast_signal_acquired),
                Toast.LENGTH_SHORT);

            results.setGravity(Gravity.CENTER,
                results.getXOffset() / 2, results.getYOffset() / 2);
            results.show();
        }
    }
};
```
Simulating GPS Data

• to simulate changes in location in emulator

• GPS data in a file
  – GPS Exchange Format (GPX)
Creating GPX Files

• Many apps and programs
• One option for Android devices
• GPSLogger
• gpsbabel to convert between various GPS formats
  – gpx has different versions
Running GPX files in App

- DDMS
- Emulator Control Tab
- GPX Tab
- Load
Running GPX
Geocoding

• Finding addresses from lat / long and vice versa
• Reverse geocoding: find address from lat and long
• Forward geocoding: find lat and long from address