Lecture
Using Maps in Android

DT228/3

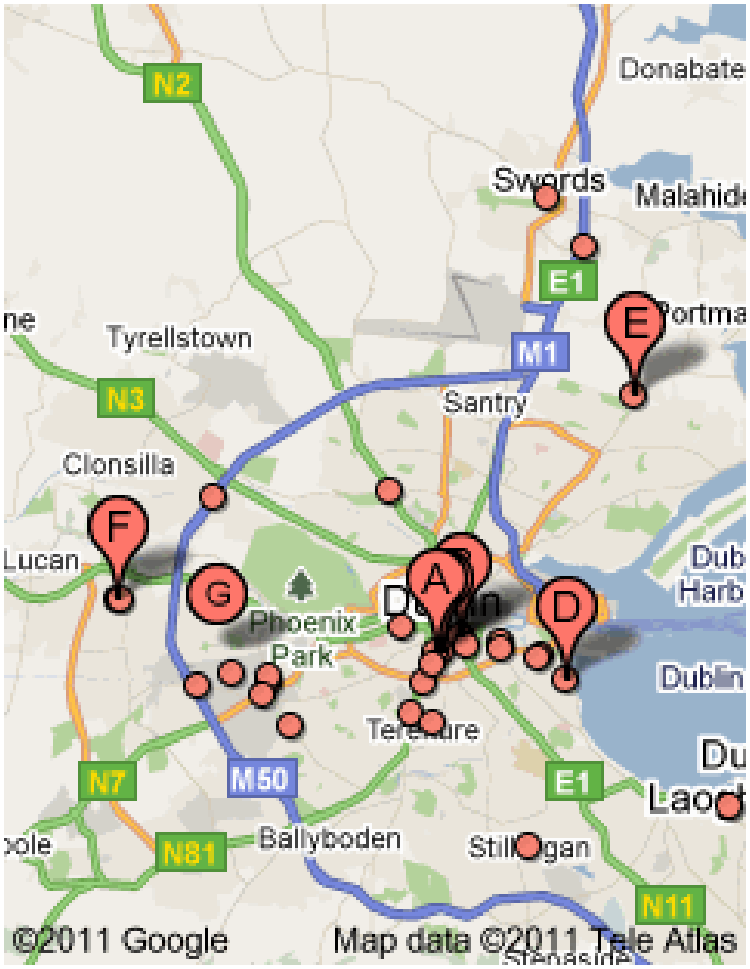
Dr. Susan McKeever

Maps

It’s easy to integrate “maps” into your app

Goes hand in hand with location tracking

Examples:



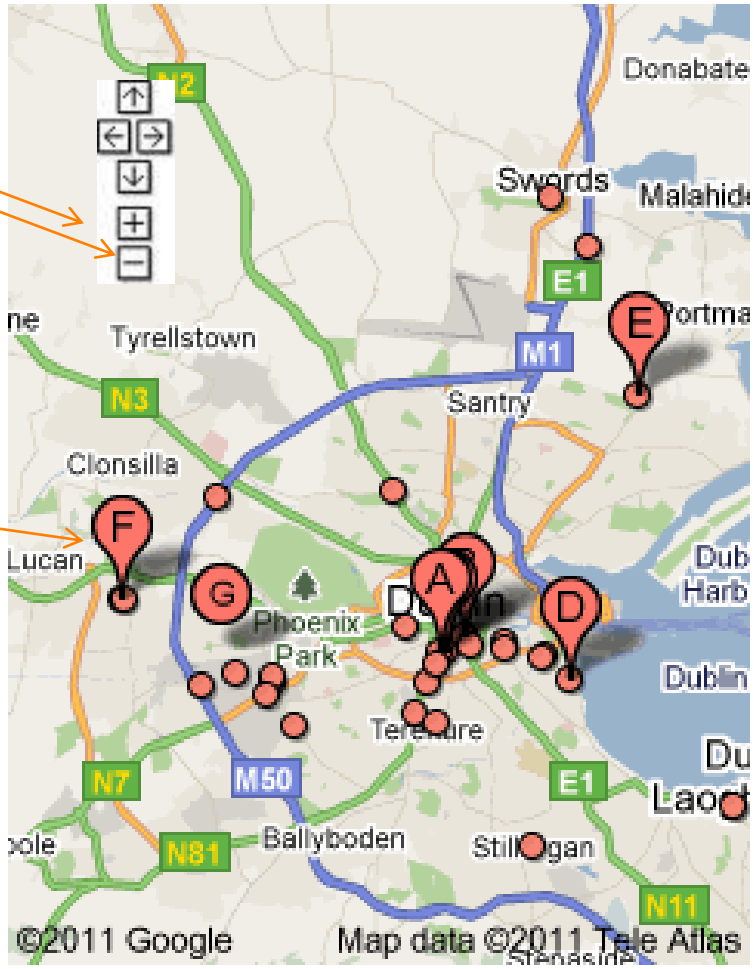
Typical things to do with a map..

Use the map controller to zoom in and out depending on what detail you want

Move around (pan)

Put an overlap on of your own stuff (i.e. pins etc)

Map a typed up street name To a physical map



Be careful…

- Using Maps in an Android app requires the GoogleMaps Android API V2
- Google Maps Android API V1.1 officially deprecated as of Dec 2012
- No longer possible to develop an new app using GoogleMaps Android API V1.1
- We will use Google Maps Android **API V1.2**
- Coding and API Key generation are completely different between V1.1 and V1.2!

In Android .. The “old” map way :

(For reference, in GoogleMaps android API V1.1 USEd.

- **MapActivity** – handles the basics of bringing in a map/
  - Extend “MapActivity” in your activity

- **XML** – a special widget is need to display a map -
  - You have to use the full path name of the MapView widget
  - `<com.google.android.maps.MapView>`)

Beware: Plenty of map code samples use this

In Android.. The “old” map way..

For reference, in Google Maps androidAPI V1 – we used..

- **MapActivity** handles the basics of bringing in a map
  - Extend “MapActivity” in your activity
- **XML** – a special widgets need to display a map -
  - You have to use the full path name of the MapView widget
  - `<com.google.android.maps.MapView>`

In Android..

In Google Maps androidAPI V2 onwards – we use..

- **XML Layout** – a special widget is need to display a map –
  - <Fragment>…is the tag..
  - `A class attribute of com.google.android.gms.maps.MapFragment>`

- **An Activity (or FragmentActivity – see "Quirks")** –
  - To simply display the map – extend Activity
  - And just render the XMl layout

- **A bunch of *very specific* set up steps**: API Key, Google play services, Manifest file changes.

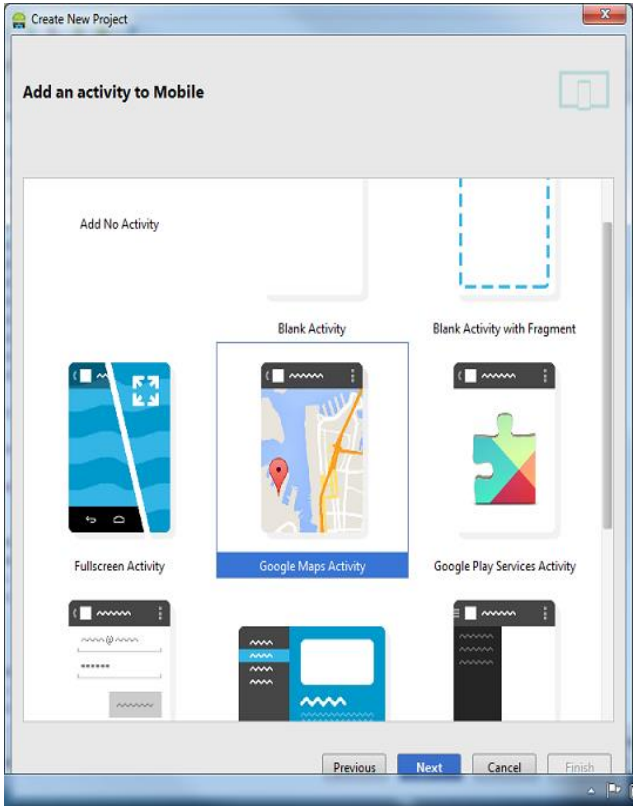
Android Studio

At last....

- **Have automated several of the steps for map set up**

- **Pick “ Google Maps Activity”** when setting up your new activity

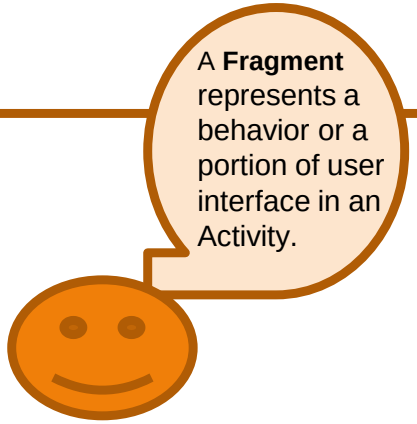
- This automates alot of the Manifest file/ XML/ Key set up...



XMl for displaying the map -

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android=http://schemas.android.com/apk/res/android
 android:layout_width="fill_parent"
 android:layout_height="fill_parent">
  <fragment
    class="com.google.android.gms.maps.MapFragment"
    android:layout_width="match_parent"
    android:id="@+id/map"
    android:layout_height="match_parent"/>
</RelativeLayout>
```

Specific class name required to set the fragment to a google map



Automated in Studio

Activity for displaying the map

```
public class MyMapActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Very simple… just sets the layout to the one with the map fragment

Automated in Studio

Resulting in..

Provided all the set up steps have been done!

Set up steps to implement a map

Step (1) Google Play Services (mostly automated in Studio)

- **Install Google Play Services** –
  - See separate fully documented steps in Web courses.
- **Add Googe Play Services to your workspace**–
  - See separate fully documented steps in Web courses.
- **Link your Android project to Google Play Services Lib.**
  - See separate fully documented steps in Web courses!
  - Note: Poorly documented online..!

Set up steps to implement a map

Step (2) Get a Google API Key

- **Retrieve your SHA1 key for your Android environment**

  Automated in Studio

  - Go to your java installation “bin” directory..
  - Run the “keytool” to get your key from the keystore
  - Note down the generated SHA1 key code
  - See separate fully documented steps in Web courses.
- **On the online “Google API Console”, generate an API key for your project ***
  - See separate fully documented steps in Web courses.

* As of July 2018, Google only give out API keys linked to a billing account – it’s free but still requires a credit card

Set up steps to implement a map

Step (2) Get a Google API V2 Key

- **Add the API key generated to your Android Manifest file in your project**
  - Put this in before </application>
  - Replace API_Key with your key from the Google console..
  - See separate fully documented steps in Web courses.

```
<meta-data
android:name="com.google.android.maps.v2.API_KEY"
android:value="Your Google Maps API V2 Key" />
```

Set up steps to implement a map

Step (3) Set Permissions in the Manifest file

- **An app that displays maps needs to track location, access the internet, write to external storage, network state access..and others: Requires user agreement**
- Put in your project's package name

```
<permission android:name="packagename.permission.MAPS_RECEIVE"
                    android:protectionLevel="signature"/>
<uses-permission android:name="packagename.permission.MAPS_RECEIVE"/>
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Automated in Studio

Set up steps to implement a map

Step (4) Enable your app use to use OPENGL

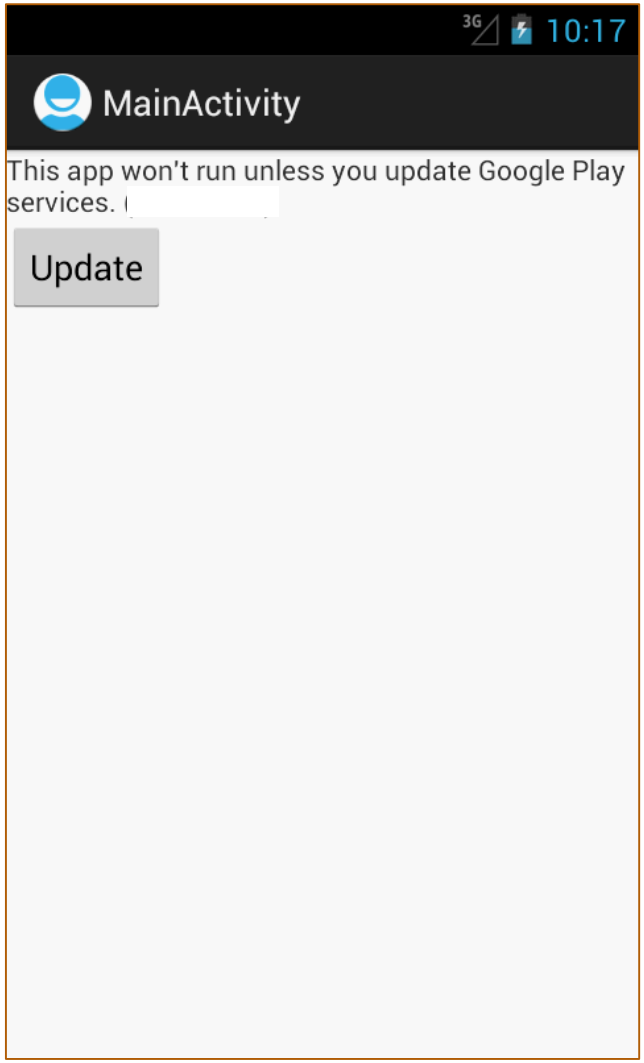
- **OpenGL is needed to render a map**
- It's an API for 2D and 3D graphics rendering
- In the Manifest:

```
<uses-feature
android:glEsVersion="0x00020000"
android:required="true"/>
```

Automated in Studio?

Emulator.. It doesn't run maps! Need A real device..

- Officially..
- You Have to use a real device (i.e. phone or tablet) to test a map
- Maps don't currently display on the Android emulator (but GenyMotion supposedly does..)
- But a few workarounds online…
  - Install two particular APKs using ADB install.. **Go and google!**



A bit more on actual Map coding..

So far...Have just seen how to display a static map. To DO something with the map, need to “connect” to it..

`GoogleMap` is the main map class which contains most of the methods for using maps. The MapFragment has the `getMap()` method to access this class.

```
private GoogleMap myMap;

myMap =
((MapFragment)getFragmentManager().findFragmentById(R.id.map)).getMap();
```

Adding markers to maps

```
private GoogleMap myMap;

myMap =
((MapFragment)getFragmentManager().findFragmentById(R.id.map)).
getMap();

myMap.addMarker(new MarkerOptions()
        .position(new LatLng(0, 0))
        .title("My First Marker"));
```

A default marker is Automated in Studio

Marking specific places..


```
static final LatLng DUBLIN = new LatLng(53.343036,-
6.254654);
Marker dublin = myMap.addMarker(new MarkerOptions()
                                .position(DUBLIN));
```

Note use of LatLng class.. Stores latitude, longitude as coordinates

Making things happen when you click on the markers..

On the GoogleMap you can register a listener for the markers in your map via the `setOnMarkerClickListener(OnMarkerClickListener)` method.

The OnMarkerClickListener class defines the `onMarkerClicked(Marker)` method which is called if a marker is clicked.

Quirks

- Sometimes you'll **see** `SupportMapFragment` used instead of `MapFragment` in the XML
  - It's related to version of Android SDK… Using SupportMapFragment for device running API Level below 12.
  - Mapfragment requried minSDK = 11 in manifest file
  - Your activity to display will extend activity instead of FragmentActivity
  - Functionality of the map will be the same..

Quirks

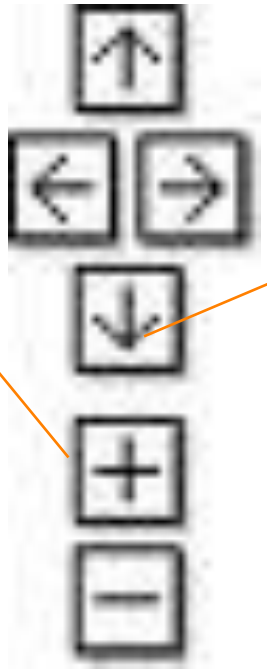
Problems running Keytool when finding the SHA1 key??

- Make sure you are looking in the right place..
  - <java install directory/JREx/bin/keytool
  - Give it the right parameters e.g.

```
c:\Program Files\Java\jdk1.6.0_24\bin>keytool -list -alias
androiddebugkey -keystore
"C:\Users\y\.android\debug.keystore" -storepass android -
keypass android
```

Zoom levels on maps – 21 levels

Level 1 = Whole world



Level 21 = Close up street level

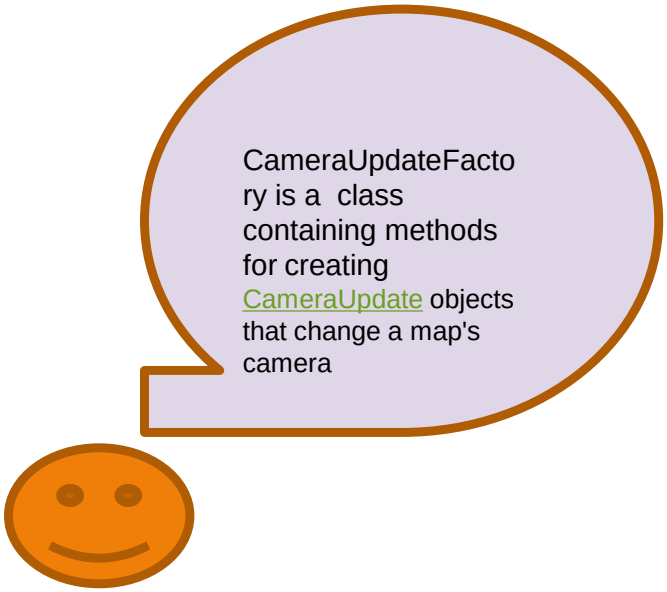
Enabled by default

```
myMap.getUiSettings().setZoomGesturesEnabled(false);
```

to

Sample Code

- DisplayMapActivity.java
- What's it doing?



Other maps things..

- Other aspects
  - **Compass**
    - Enable/disable
  - **GeoCodes**
    - Turning addresses into GPS
    - Reverse Geocoding
      - GPS → Address