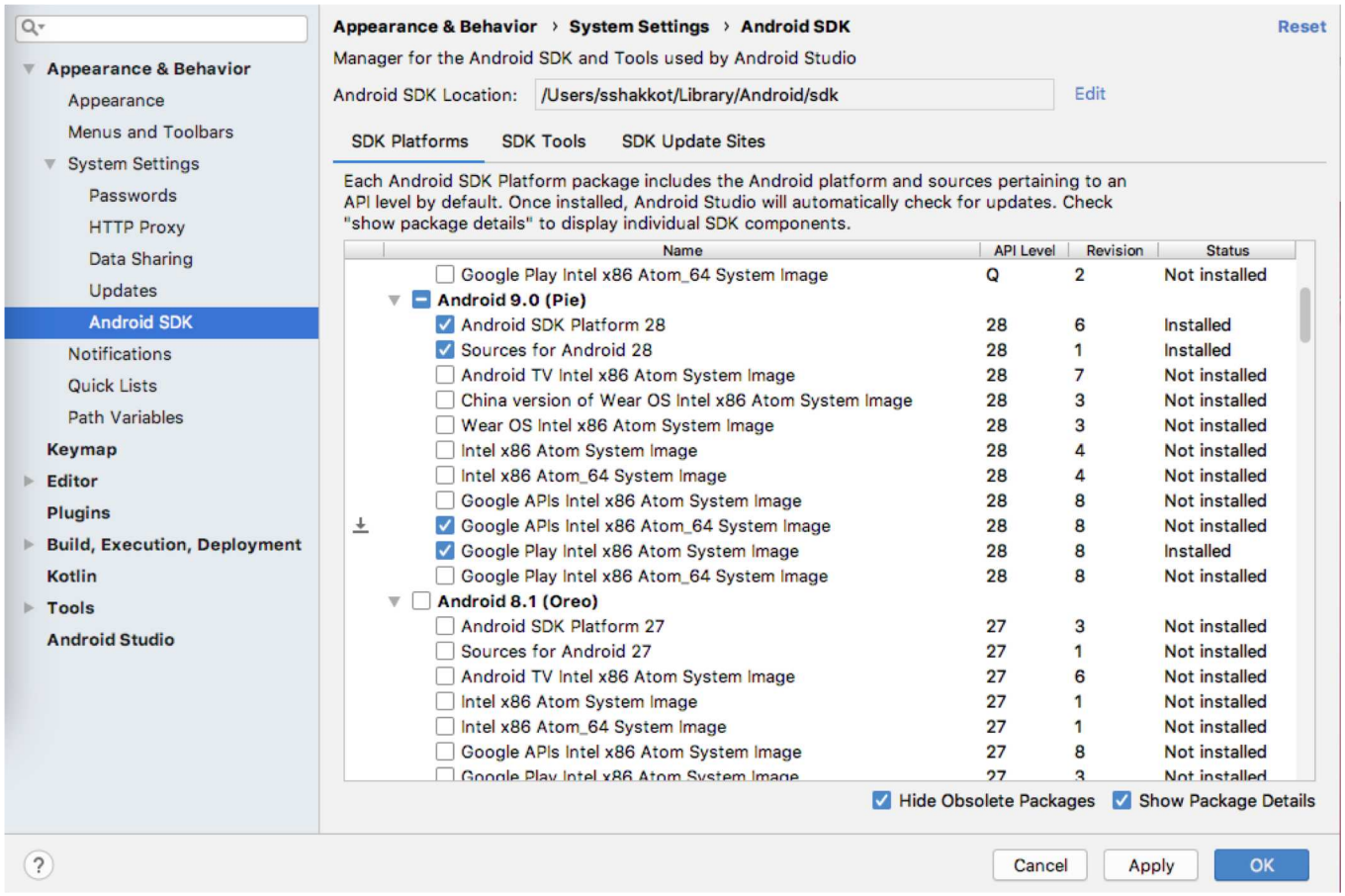
22. Maps and GPS

Installing Google Play services

- need to install **Google Play** services and Google API
  - SDK Manager
  - click Install packages...then create an AVD

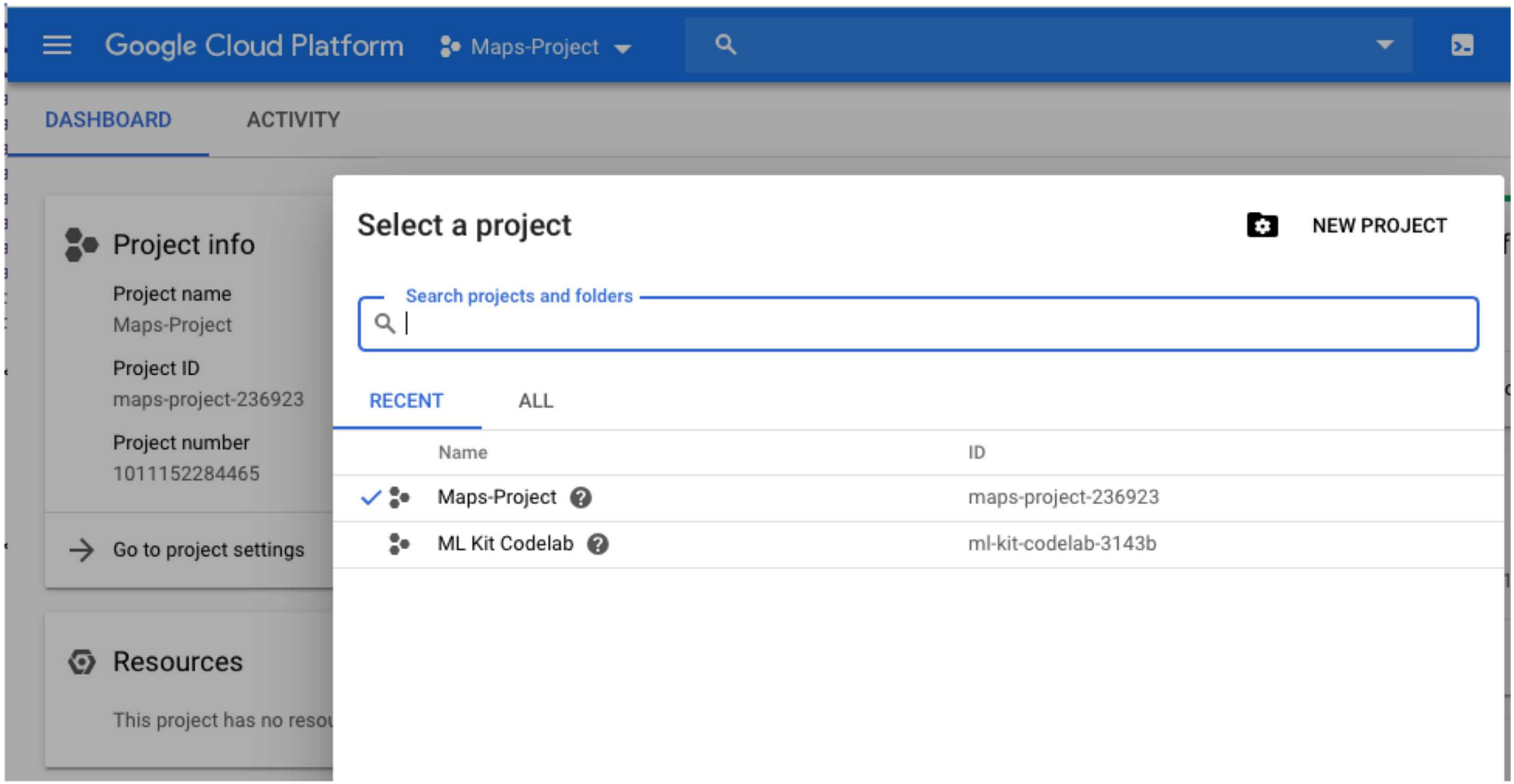


Procedure for Enabling Maps

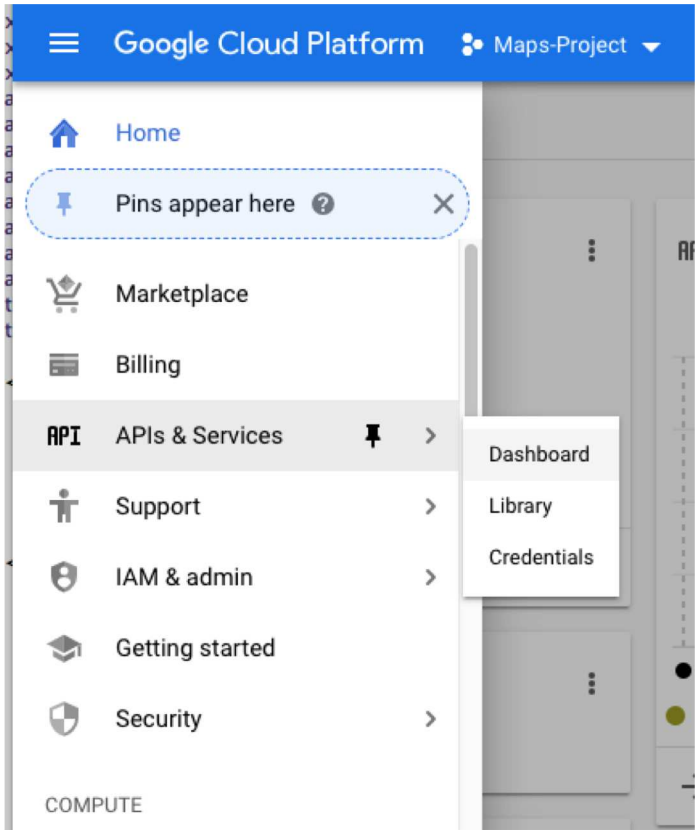
Google Tutorial:
https://developers.google.com/maps/documentation/android-sdk/start

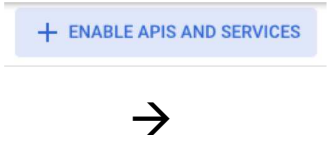
1. Create a Google Cloud Project and enable the Maps API in it (Good to do this first)
2. Create a Google Maps Activity
3. Get a Google Maps API key

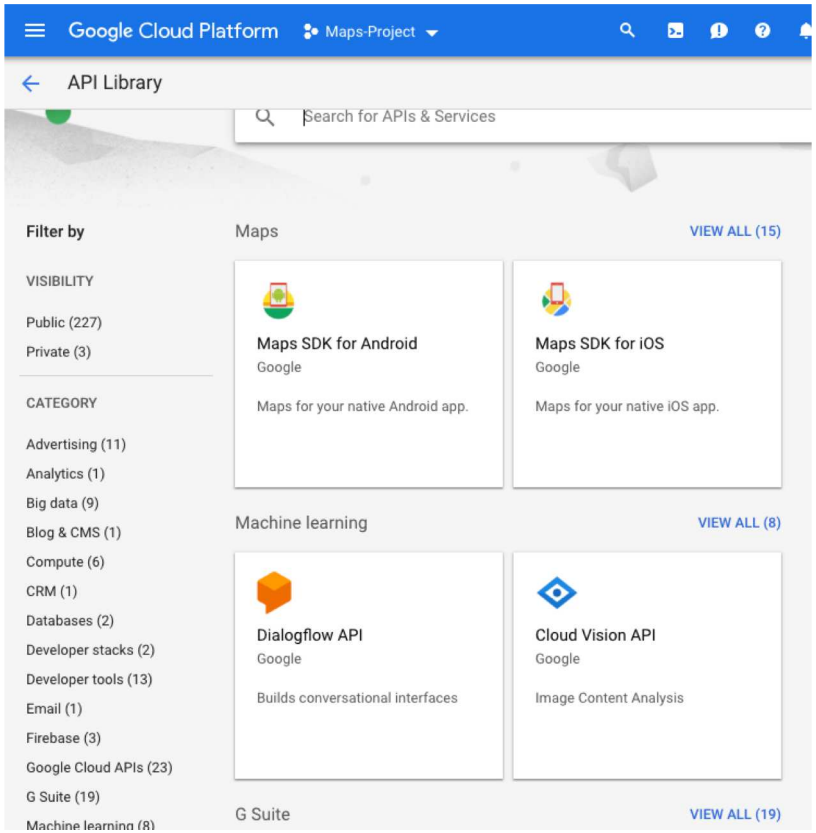
1. Google Cloud: Create Project



1. Google Cloud: Add API (if desired)







2. Create Maps Activity in Android

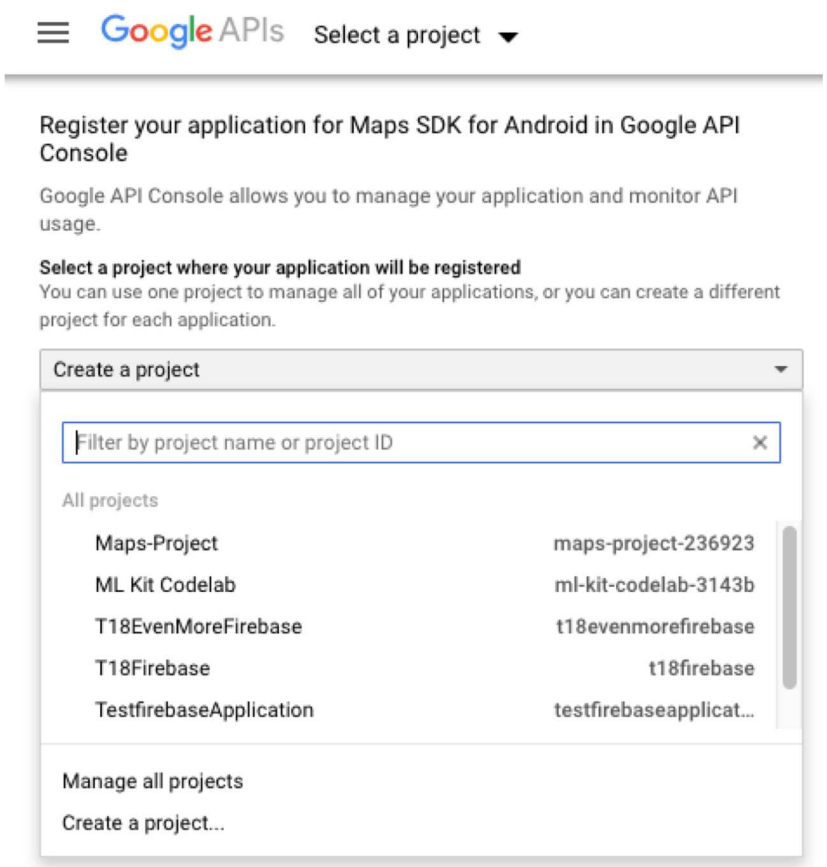
google_maps_api.xml

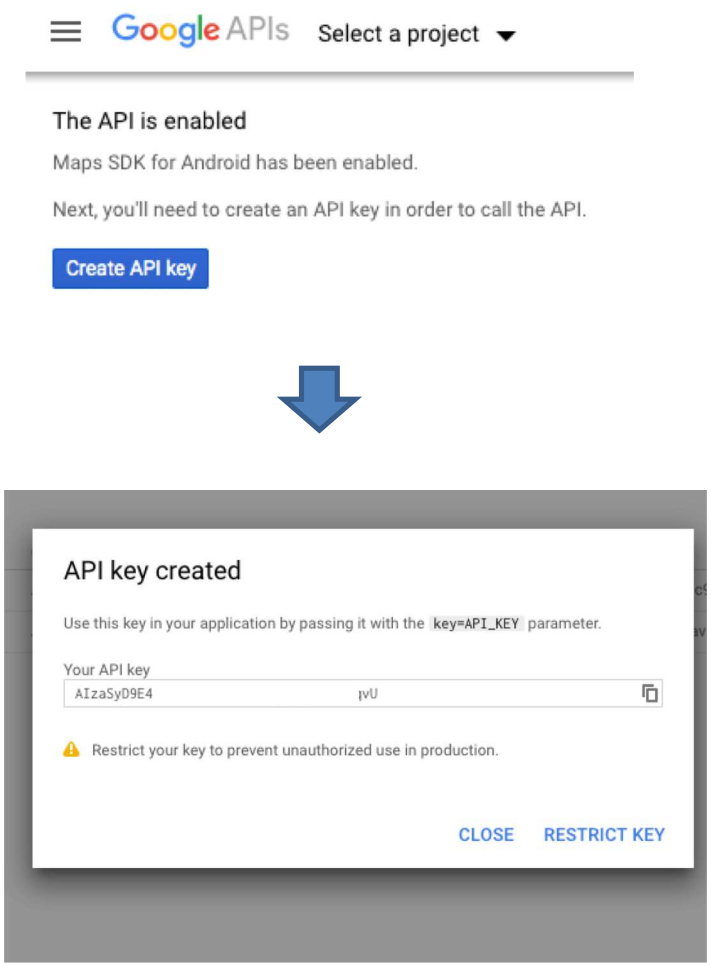
<resources>
<!--
TODO: Before you run your application, you need a Google Maps API key.
…
To get one, follow this link, follow the directions and press "Create" at the end:

https://console.developers.google.com/flows/enableapi...
…
Alternatively, follow the directions here:
https://developers.google.com/maps/documentation/android/start#get-key

Once you have your key (it starts with "AIza"), replace the "google_maps_key" string in this file.
-->
<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">YOUR_KEY_HERE</string>
</resources>

3. Google Cloud: Create Credentials





MapFragment

- Google Maps API provides a fragment class named `MapFragment` for displaying a map within an activity.

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/map"
        tools:context=".MapsActivity"
        android:name="com.google.android.gms.maps.SupportMapFragment" />
```

Using the Map

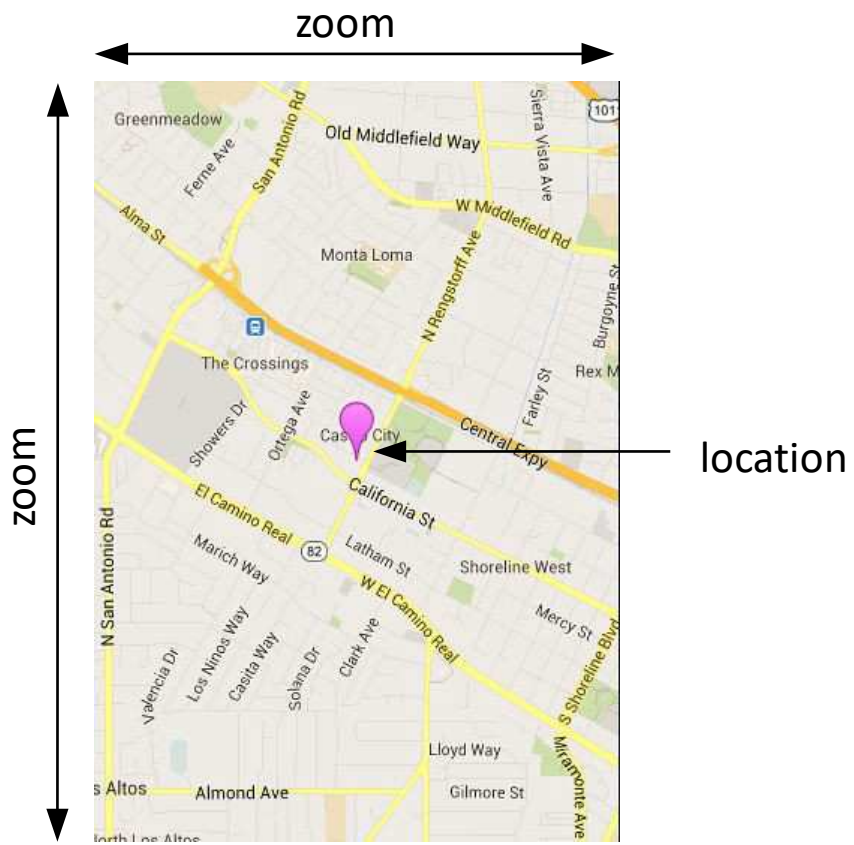
```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // Add a marker in College Station, and move the camera.
        LatLng CLL = new LatLng(30.5910, -96.3628);
        mMap.addMarker(new MarkerOptions().position(CLL).title("Marker in CLL"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(CLL));
    }
}
```

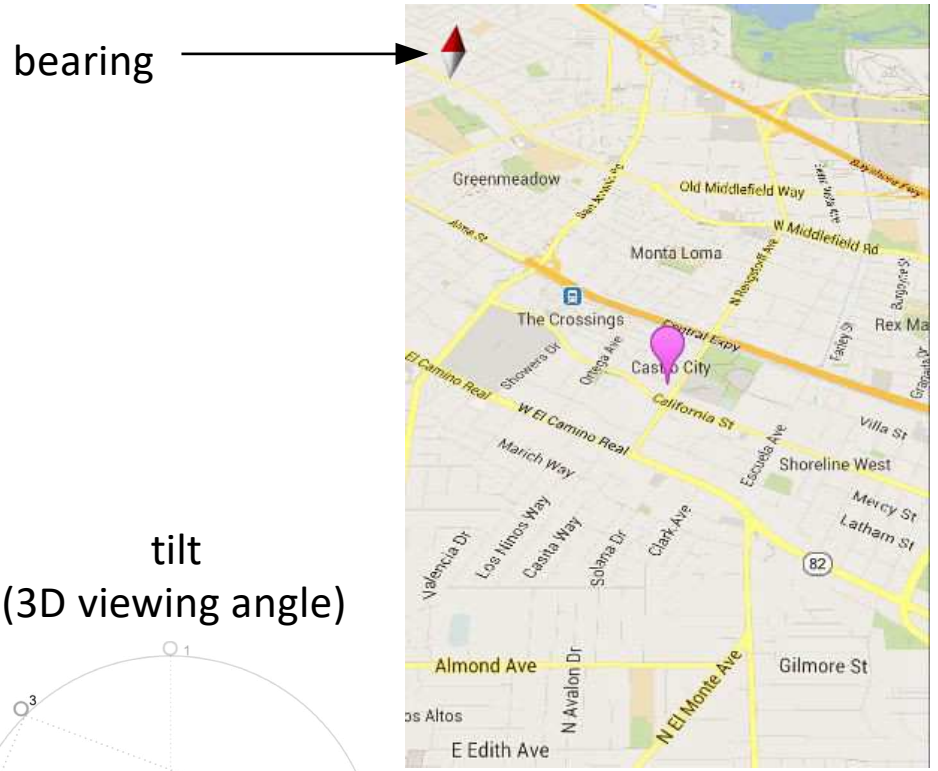
GoogleMap methods

- placing items on the map:
  - addCircle, addGroundOverlay, **addMarker**, addPolygon, **addPolyline**, addTileOverlay
  - **clear** - Removes all markers, polylines/polygons, overlays
- manipulating the camera:
  - getCameraPosition, **moveCamera**, **animateCamera**, stopAnimation
- map settings and appearance:
  - setBuildingsEnabled, setIndoorEnabled, setMapType, setPadding, setTrafficEnabled
- snapshot - take a screen shot of the map as a bitmap
- event listeners:
  - setOnCameraChangeListener, **setOnMapClickListener**, setOnMapLoadedCallback, setOnMapLongClickListener, **setOnMarkerClickListener**, setOnMarkerDragListener, setOnMyLocationChangeListener

The map's camera

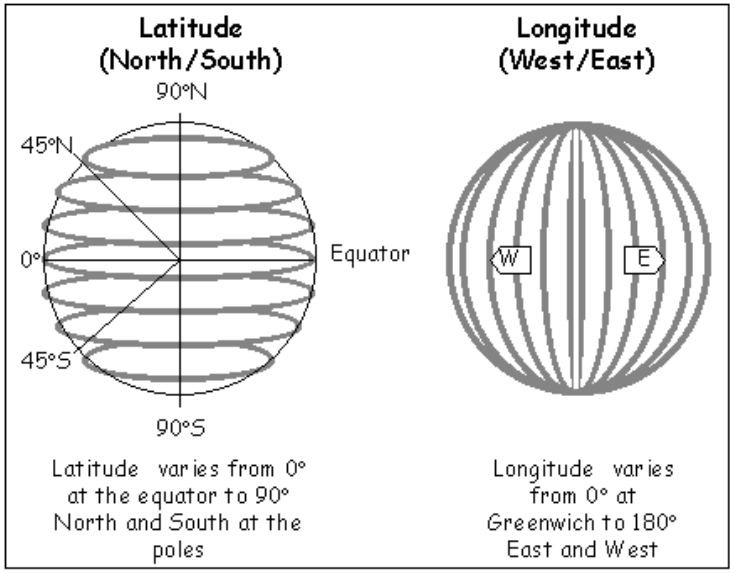
- The current viewing window of a map's camera is defined by:
  - **target** location (latitude/longitude), **zoom** (2.0 - 21.0),
  - **bearing** (orientation/rotation), and **tilt** (degrees)

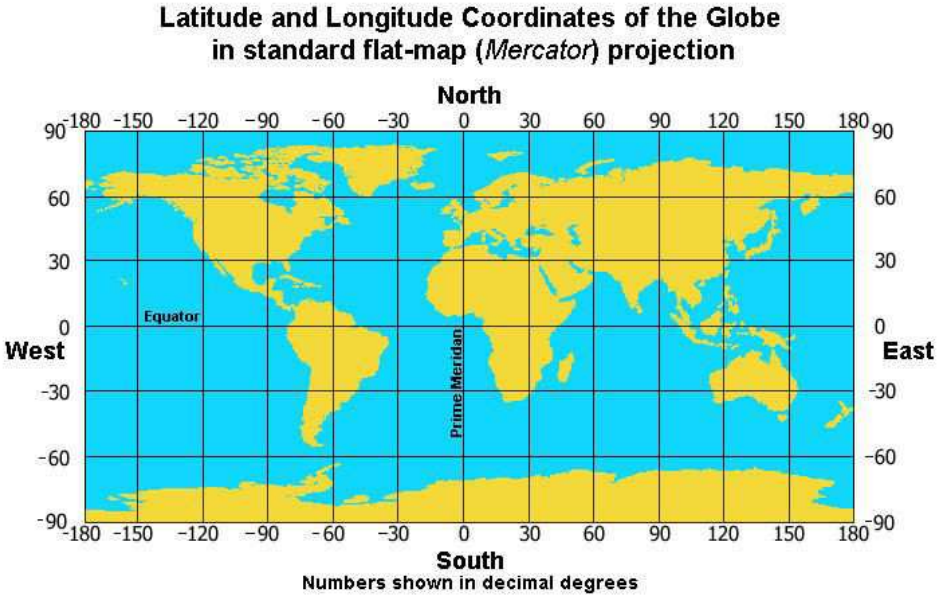




Latitude and longitude

- **latitude**: N/S angle relative to the equator
  - North pole = +90; South pole = -90
- **longitude**: E/W angle relative to prime meridian
  - West = 0 → -180; East = 0 → 180
  - *find lat/lng of a place on Google Maps in URL address bar*

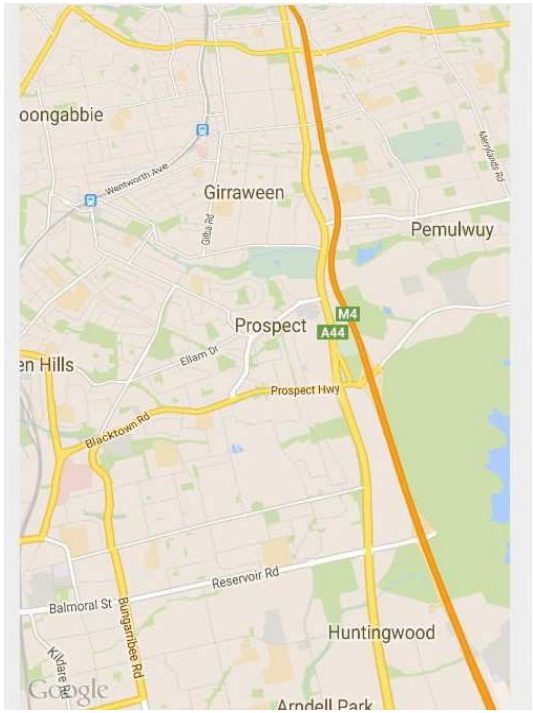




Set camera in XML

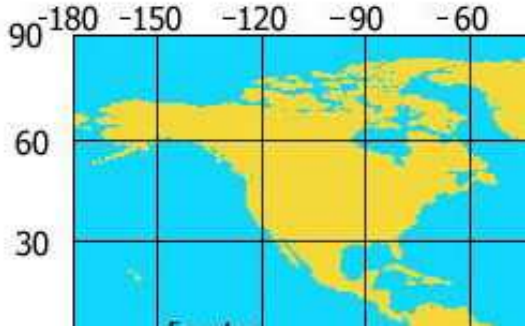
- Set initial map settings and camera position in the layout XML:
  - see here (link) for full list of attributes available

```
<fragment ...
     android:name="com.google.android.gms.maps.MapFragment"
     android:id="@+id/ID"
     map:cameraBearing="112.5"
     map:cameraTargetLat="-33.796923"
     map:cameraTargetLng="150.922433"
     map:cameraTilt="30"
     map:cameraZoom="13"
     map:mapType="normal"
     map:uiCompass="false"
     map:uiRotateGestures="true"
     map:uiScrollGestures="false"
     map:uiTiltGestures="true"
     map:uiZoomControls="false"
     map:uiZoomGestures="true" />
```



Set camera in Java code

- CameraUpdateFactory methods:
  - newLatLng(new LatLng(***lat***, ***lng***))
  - newLatLngBounds(new LatLngBounds(***SW***, ***NE***), ***padding***)
  - newLatLngZoom(new LatLng(***lat***, ***lng***), ***zoom***)
  - newCameraPosition(***CameraPosition***)
  - others:

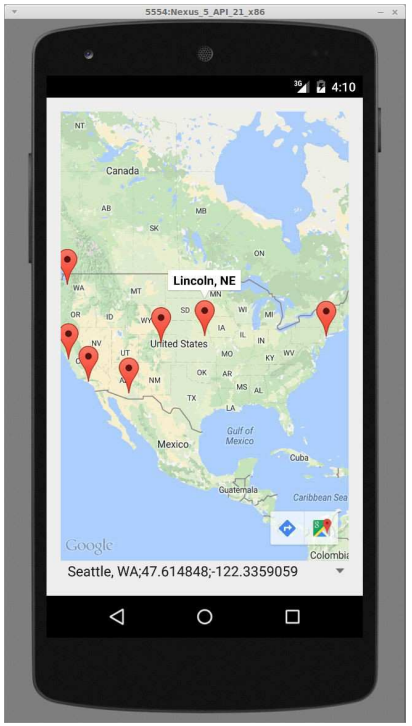



```
// example; show roughly the entire USA
LatLngBounds bounds = newLatLngBounds(
    new LatLng(20, -130.0),     // SW
    new LatLng(55,  -70.0));    // NE

map.moveCamera(CameraUpdateFactory.newLatLngBounds(bounds,  50));

// try also: map.animateCamera
```

Placing markers

- A GoogleMap object has an addMarker method that can let you put "push pin" markers at locations on the map.
  - The marker's methods return the marker, so you can chain them.
  - options: alpha, draggable, icon, position, rotation, title, visible, ...

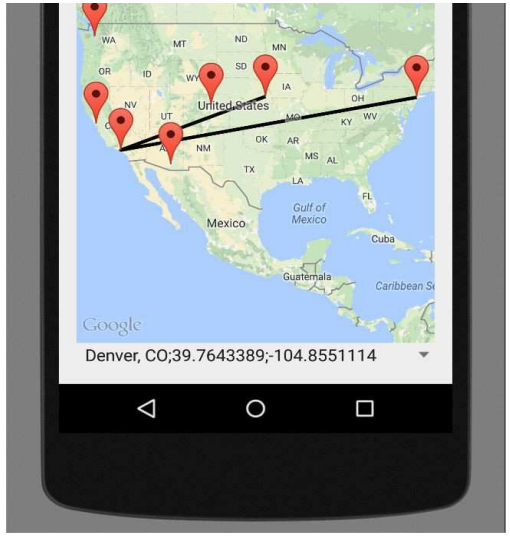
```
map.addMarker(new MarkerOptions()
    .position(new LatLng(40.801, -96.691))
    .title("Lincoln, NE")
);

// to modify/remove the marker later
Marker mark = map.addMarker(new MarkerOptions()
    ...);
mark.remove();
```



Lines and paths

- A GoogleMap object has an addPolyline method that can let you put lines between locations on the map.
  - options: color, visible, width, zIndex, ...

```
map.addPolyline(new PolylineOptions()
    .add(new LatLng(40.801, -96.691))    // Lincoln, NE
    .add(new LatLng(34.020, -118.412))   // Los Angeles, CA
    .add(new LatLng(40.703, -73.980))    // New York, NY
);

// to modify/remove the line later
Polyline polly = map.addPolyline(...);
polly.remove();
```



Accessing Location (Permissions!)

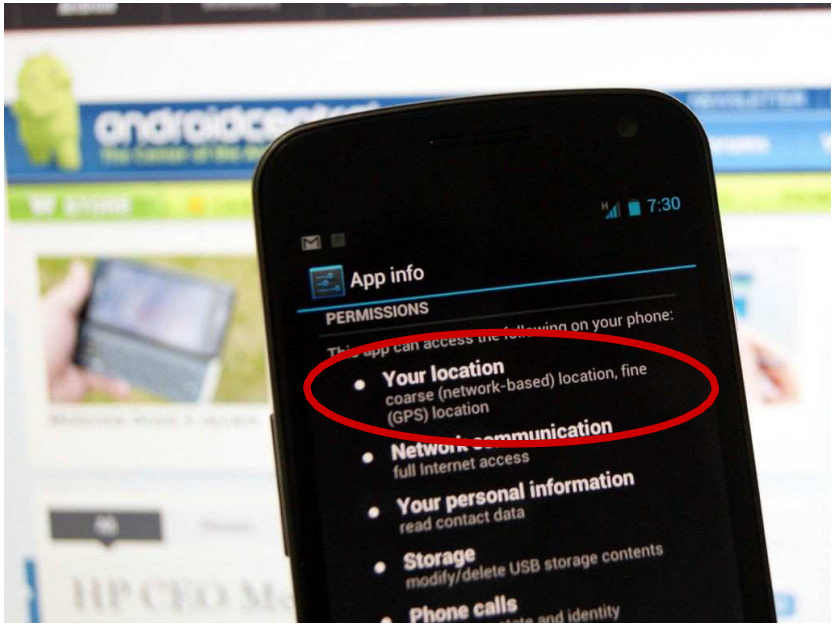
- Android `LocationManager` gives you the phone's position:
  - GPS provider provides highest accuracy
  - Network provider is a fallback in case GPS is disabled / not present

```
LocationManager locationManager = (LocationManager)
    getSystemService(Context.LOCATION_SERVICE);
Location loc = locationManager.getLastKnownLocation(
                    LocationManager.GPS_PROVIDER);
if (loc == null) {
    // fall back to network if GPS is not available
    loc = locationManager.getLastKnownLocation(
                    LocationManager.NETWORK_PROVIDER);
}
if (loc != null) {
    double myLat = loc.getLatitude();
    double myLng = loc.getLongitude();
    ...
    // other methods: getAltitude, getSpeed, getBearing, ...
```

1. AndroidManifest.xml permissions

- Because of privacy issues, to access phone's current location, must request permission in `AndroidManifest.xml`:

```
<manifest ...>
    <uses-permission
        android:name="android.permission.ACCESS_COARSE_LOCATION"  />
    <uses-permission
        android:name="android.permission.ACCESS_FINE_LOCATION"  />

    <application ...>
        ...
    </application>
</manifest>
```



2. Permissions needed in runtime

```
if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

   ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1);
   return;
}
else {
        //You have permissions and may run your code now

}
```

Location update events

- Track user's movement by listening for location update events.

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

locationManager.requestLocationUpdates(
    LocationManager.GPS_PROVIDER, 0, 0,   // provider, min time/distance
    new LocationListener() {
        public void onLocationChanged(Location location) {
            // code to run when user's location changes

        }
        public void onStatusChanged(String prov, int stat, Bundle b){}
        public void onProviderEnabled(String provider) {}
        public void onProviderDisabled(String provider) {}
    }
);
```