ANDROID

Maps and location

16.12.2015

MOBILE APPLICATIONS PROGRAMMING

Krzysztof Pawłowski

Polsko-Japońska Akademia Technik Komputerowych

GOOGLE MAPS API

Google Maps Android API provides:

- access to Google Maps servers
- data downloading
- map display
- response to map gestures
- enables to add markers, polygons, overlays etc.

GET A GOOGLE MAPS API KEY

- The easiest way:
  - add Google Maps Activity in Android Studio
  - open url from generated google_maps_api.xml file and follow instructions to generate keys
  - paste generated key to google_maps_api.xml file

ADDING A MAP TO ACTIVITY

In activity's layout xml file:

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

or (in onCreate method):

```
mMapFragment = MapFragment.newInstance();
FragmentTransaction fragmentTransaction =
        getFragmentManager().beginTransaction();
fragmentTransaction.add(R.id.my_container, mMapFragment);
fragmentTransaction.commit();
```

MAP READY CALLBACK

To be able to implement callback for map ready we need to implement OnMapReadyInterface in out activity:

```
public class MainActivity extends FragmentActivity
    implements OnMapReadyCallback {
...
}
```

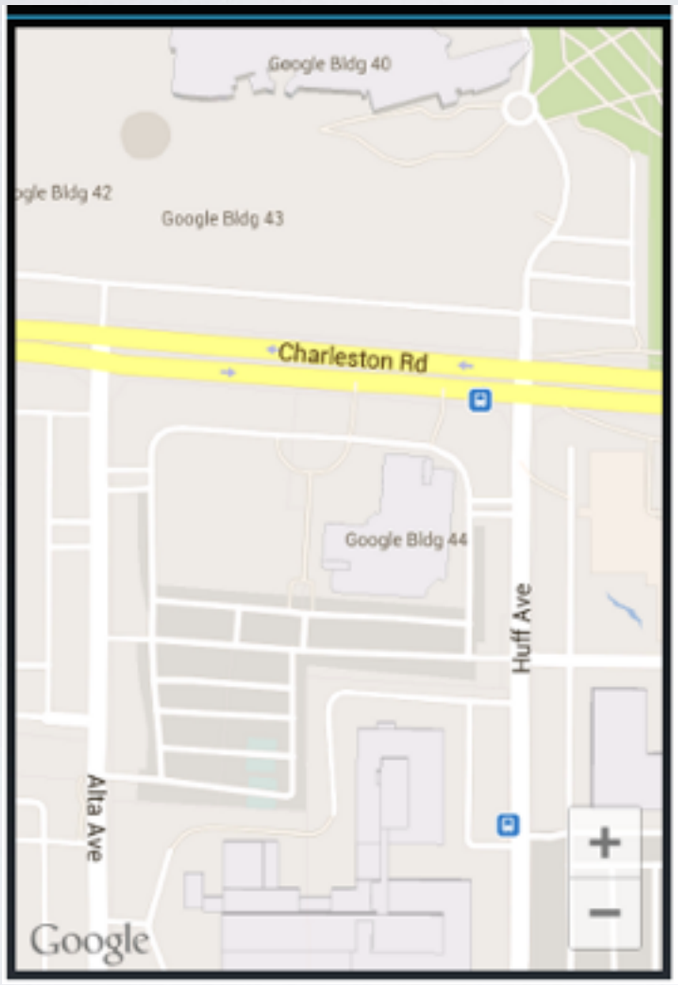
We set our activity as callback class (casting to OnMapReadyCallbackType) by calling getMapAsync method:

```
MapFragment mapFragment = (MapFragment) getFragmentManager()
    .findFragmentById(R.id.map);
mapFragment.getMapAsync(this);
```

We implement the callback for map ready in the activity:

```
@Override
public void onMapReady(GoogleMap map) {
    map.addMarker(new MarkerOptions()
        .position(new LatLng(0, 0))
        .title("Marker"));
}
```

TYPES OF A MAP



Normal

Hybrid

Terrain

TYPES OF A MAP

Setting the type of a map:

```
GoogleMap map;
...
// Sets the map type to be "hybrid"
map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

INDOOR MAPS

- Are displayed for normal and satellite types when zoomed in
- Enabled by default
- To disable indoor maps:

```
GoogleMap.setIndoorEnabled(false)
```

- To disable default level picker:

```
GoogleMap.getUiSettings().setIndoor
LevelPickerEnabled(false)
```

INITIAL STATE OF A MAP

In xml file:

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

INITIAL STATE OF A MAP

Programatically:

```
GoogleMapOptions options = new GoogleMapOptions();

options.mapType(GoogleMap.MAP_TYPE_SATELLITE)
     .compassEnabled(false)
     .rotateGesturesEnabled(false)
     .tiltGesturesEnabled(false);
```

use in:

```
MapFragment.newInstance(GoogleMapOptions options)
```

MAPS LITE MODE

- Is a bitmap image of a map at a specified location and zoom level

- supports all of the map types (normal, hybrid, satellite, terrain) and a subset of the functionality supplied by the full API

MAPS LITE MODE

```
<fragment xmlns:android="http://schemas.android.com/apk/
res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    android:name="com.google.android.gms.maps.MapFragment"
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    map:cameraZoom="13"
    map:mapType="normal"
    map:liteMode=“true"/>
```

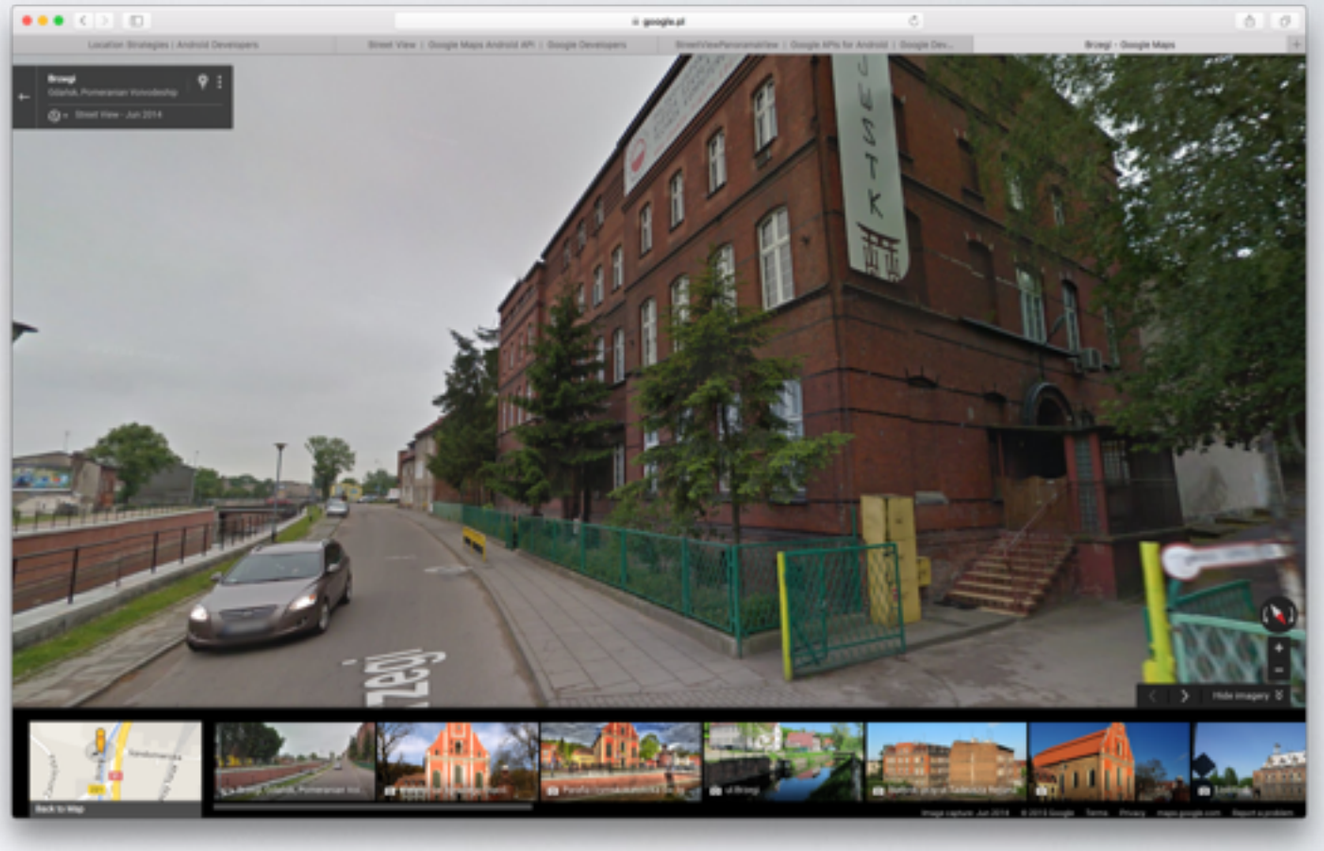
or

```
GoogleMapOptions options = new
GoogleMapOptions().liteMode(true);
```

STREET VIEW

- StreetViewPanorama or StreetViewFragment

- Each Street View panorama set of images with 360-degree view from a single location.



ADDING STREETVIEW TO ACTIVITY

Define fragment in xml layout:

```
<fragment
    android:name="com.google.android.gms.maps.StreetViewPanoramaFragment"
    android:id="@+id/streetviewpanorama"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

ImplementStreetView callback in the activity:

```
public class MainActivity extends FragmentActivity
    implements OnStreetViewPanoramaReadyCallback {
...
}
```

Pass this (activity) for callbacks:

```
StreetViewPanoramaFragment streetViewPanoramaFragment =
    (StreetViewPanoramaFragment) getFragmentManager()
        .findFragmentById(R.id.streetviewpanorama);
streetViewPanoramaFragment.getStreetViewPanoramaAsync(this);
```

INITIAL STATE OF STREETVIEW

- Not possible to configure via XML.
- Pass StreetViewPanoramaOptions object to:
  - `StreetViewPanoramaFragment.newInstance(StreetViewPanoramaOptions options)` or
  - `StreetViewPanoramaView(Context, StreetViewPanoramaOptions)`
- Method on `StreetViewPanorama` object:
  - `setPanningGesturesEnabled()`
  - `setUserNavigationEnabled()`
  - `setZoomGesturesEnabled()`
  - `setStreetNamesEnabled()`

LOCATION DATA

- Mobile phones are location aware
- Wifi, network based location
- GPS based location

LOCATION PERMISSIONS

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp" >
  ...
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
  ...
</manifest>
```

LOCATION PERMISSIONS

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp" >
  ...
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
  ...
</manifest>
```

LOCATION PERMISSIONS - REQUEST IN RUNTIME

```
if (ContextCompat.checkSelfPermission(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)
                    == PackageManager.PERMISSION_GRANTED) {
    mMap.setMyLocationEnabled(true);
} else {
        ActivityCompat.requestPermissions(thisActivity,
                new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                MY_PERMISSIONS_REQUEST_FINE_LOCATION);
    }
}
```

LOCATION PERMISSIONS - REQUEST IN RUNTIME

```
@Override
public void onRequestPermissionsResult(int requestCode, String[]
permissions, int[] grantResults) {
    if (requestCode == MY_LOCATION_REQUEST_CODE) {
      if (permissions.length == 1 &&
          permissions[0] == Manifest.permission.ACCESS_FINE_LOCATION &&
          grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        mMap.setMyLocationEnabled(true);
    } else {
      // Permission was denied. Display an error message.
    }
}
```

GETTING CURRENT LOCATION

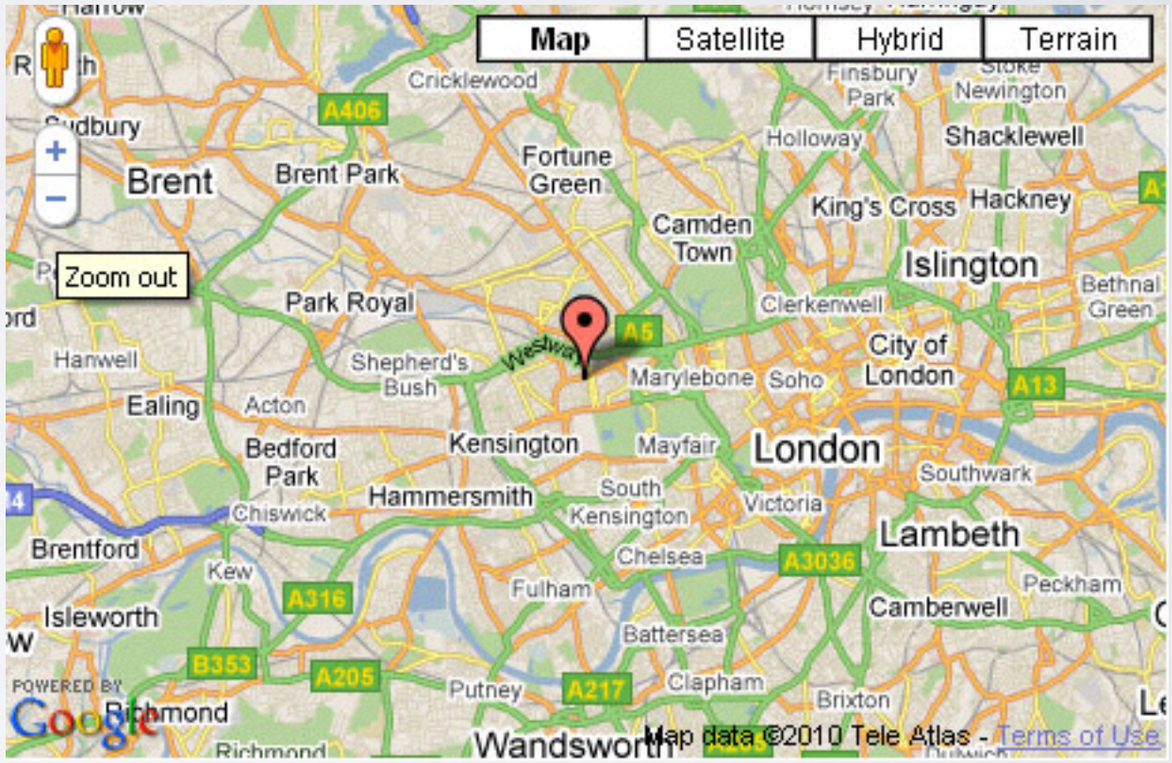
```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback,
LocationListener,
        GoogleApiClient.ConnectionCallbacks, GoogleApiClient.OnConnectionFailedListener {
…
@Override
protected void onCreate(Bundle savedInstanceState) {
    …
    // Create an instance of GoogleAPIClient.
 if (mGoogleApiClient == null) {
     mGoogleApiClient = new GoogleApiClient.Builder(this)
             .addConnectionCallbacks(this)
             .addOnConnectionFailedListener(this)
             .addApi(LocationServices.API)
             .build();
 }

protected void onStart() {
    mGoogleApiClient.connect();
    super.onStart();
}

protected void onStop() {
    mGoogleApiClient.disconnect();
    super.onStop();
}
```

GETTING CURRENT LOCATION

```
@Override
public void onLocationChanged(Location location) { … }

@Override
public void onConnected(Bundle bundle) {
    createLocationRequest();
    startLocationUpdates();
}

  protected void startLocationUpdates() {
      LocationServices.FusedLocationApi.requestLocationUpdates(
              mGoogleApiClient, mLocationRequest, this);
  }
  @Override
  public void onConnectionSuspended(int i) { … }

  @Override
  public void onConnectionFailed(ConnectionResult connectionResult) { … }

  protected void createLocationRequest() {
      mLocationRequest = new LocationRequest();
      mLocationRequest.setInterval(10000);
      mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
  }
}
```

MARKERS

- Indicate single location on the map
- Can be customised:
  - color
  - icon



ADDING A MARKER

```
@Override
public void onMapReady(GoogleMap map) {
    map.addMarker(new MarkerOptions()
        .position(new LatLng(10, 10)));
}
```

MAKING A MARKER DRAGGABLE

```
static final LatLng PERTH = new LatLng(-31.90, 115.86);
Marker perth = mMap.addMarker(new MarkerOptions()
                                .position(PERTH)
                                .draggable(true));
```

CUSTOMISE MARKER COLOUR

```
static final LatLng MELBOURNE = new LatLng(-37.813, 144.962);
Marker melbourne = mMap.addMarker(new MarkerOptions()
   .position(MELBOURNE)
   .icon(BitmapDescriptorFactory.defaultMarker(
      BitmapDescriptorFactory.HUE_AZURE)));
```

CUSTOMISE MARKER OPACITY

```
static final LatLng MELBOURNE = new LatLng(-37.813, 144.962);
Marker melbourne = mMap.addMarker(new MarkerOptions()
                                  .position(MELBOURNE)
                                  .alpha(0.7f));
```

CUSTOMISE MARKER ICON

```
private static final LatLng MELBOURNE = new LatLng(-37.813, 144.962);
private Marker melbourne = mMap.addMarker(new MarkerOptions()
                                .position(MELBOURNE)
                                .icon(BitmapDescriptorFactory
                                    .fromResource(R.drawable.arrow)));
```

FLATTEN A MARKER

```
static final LatLng PERTH = new LatLng(-31.90, 115.86);
Marker perth = mMap.addMarker(new MarkerOptions()
                              .position(PERTH)
                              .flat(true));
```

ROTATE A MARKER

```
static final LatLng PERTH = new LatLng(-31.90, 115.86);
Marker perth = mMap.addMarker(new MarkerOptions()
                                .position(PERTH)
                                .anchor(0.5,0.5)
                                .rotation(90.0));
```

MARKER EVENTS

- GoogleMap.OnMarkerClickListener
  - onMarkerClick()
- GoogleMap.OnMarkerDragListener
  - onMarkerDrag(Marker marker)
  - onMarkerDragStart(Marker marker)
  - onMarkerDragEnd(Marker marker)

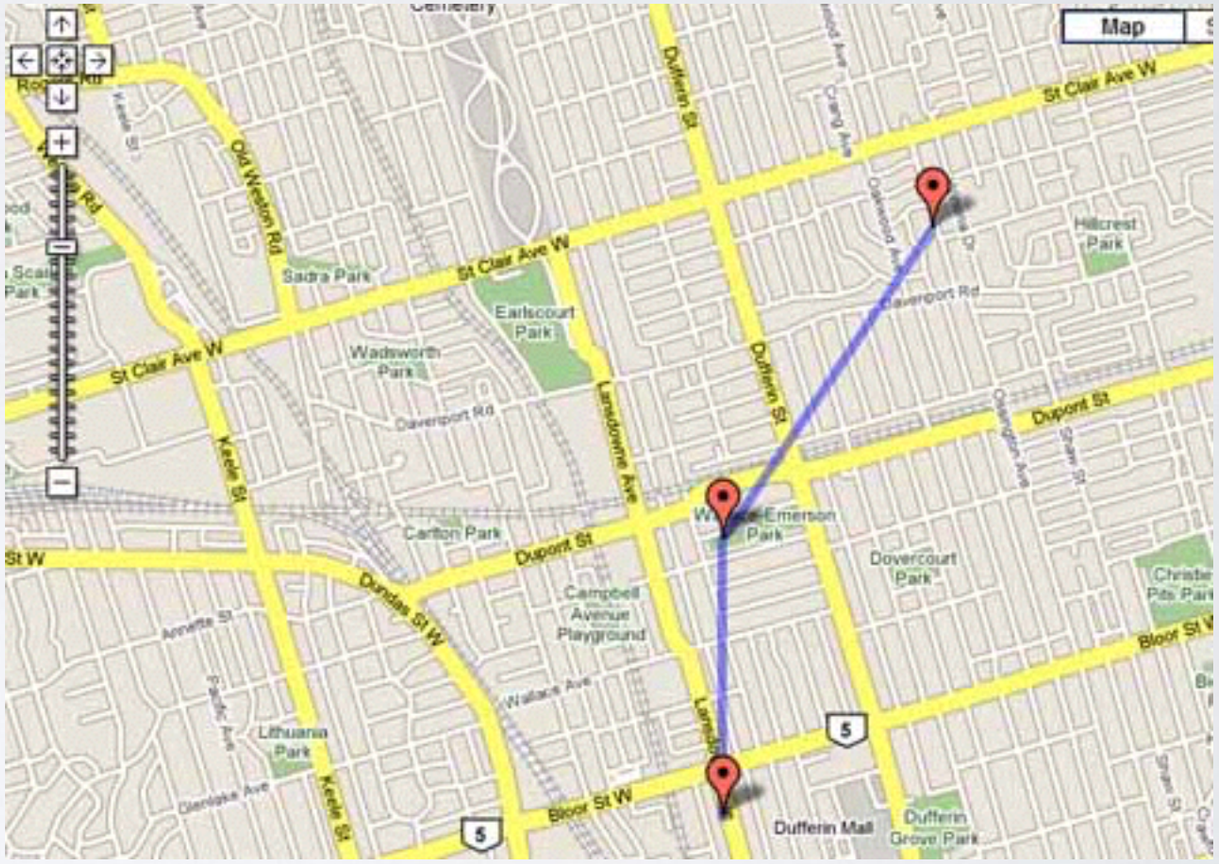
INFO WINDOWS

INFO WINDOWS

```
static final LatLng MELBOURNE = new LatLng(-37.81319, 144.96298);
Marker melbourne = mMap.addMarker(new MarkerOptions()
                                  .position(MELBOURNE)
                                  .title("Melbourne")
                                  .snippet("Population: 4,137,400”));
melbourne.showInfoWindow();
```

POLYLINE

- Defines a set of connected line segments on the map
- Consists of a list of LanLgt locations

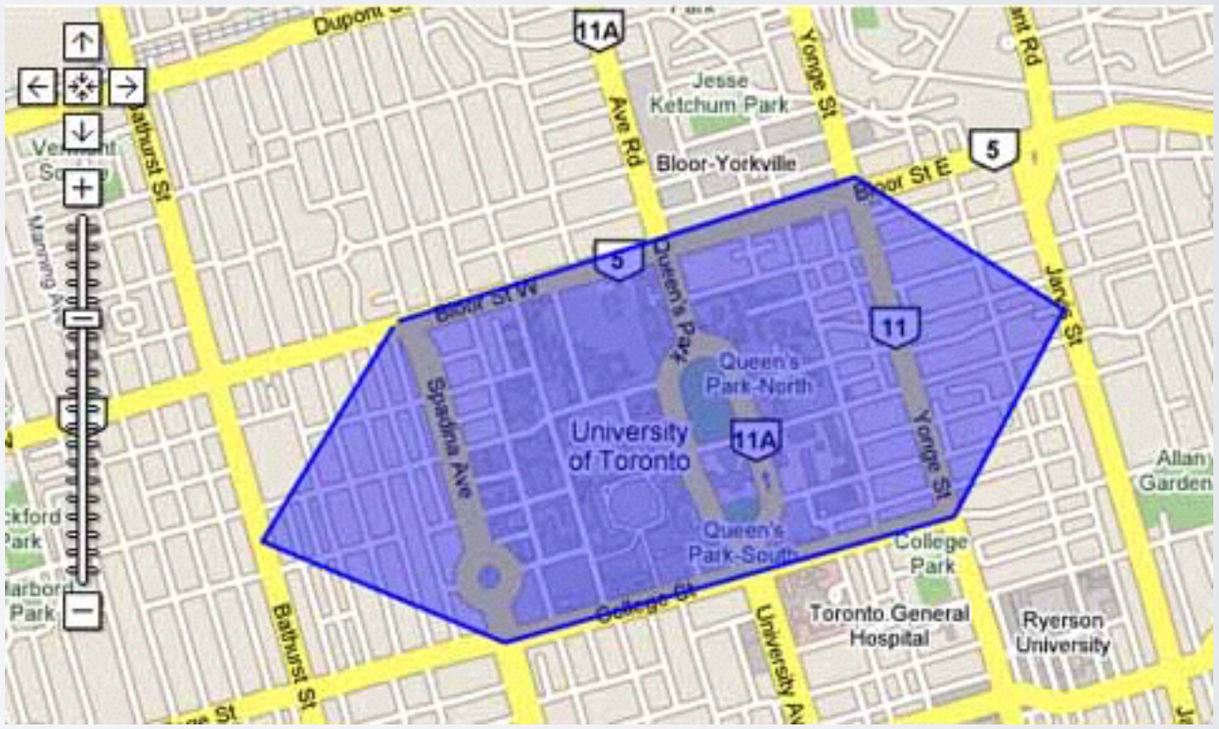


POLYLINE

```
// Instantiates a new Polyline object and adds points to define a
// rectangle
PolylineOptions rectOptions = new PolylineOptions()
        .add(new LatLng(37.35, -122.0))
        .add(new LatLng(37.45, -122.0))
        .add(new LatLng(37.45, -122.2))
        .add(new LatLng(37.35, -122.2))
        .add(new LatLng(37.35, -122.0));

// Get back the mutable Polyline
Polyline polyline = myMap.addPolyline(rectOptions);
```

POLYGON

- Defines an area on the map
- Consists of a list of LanLgt locations

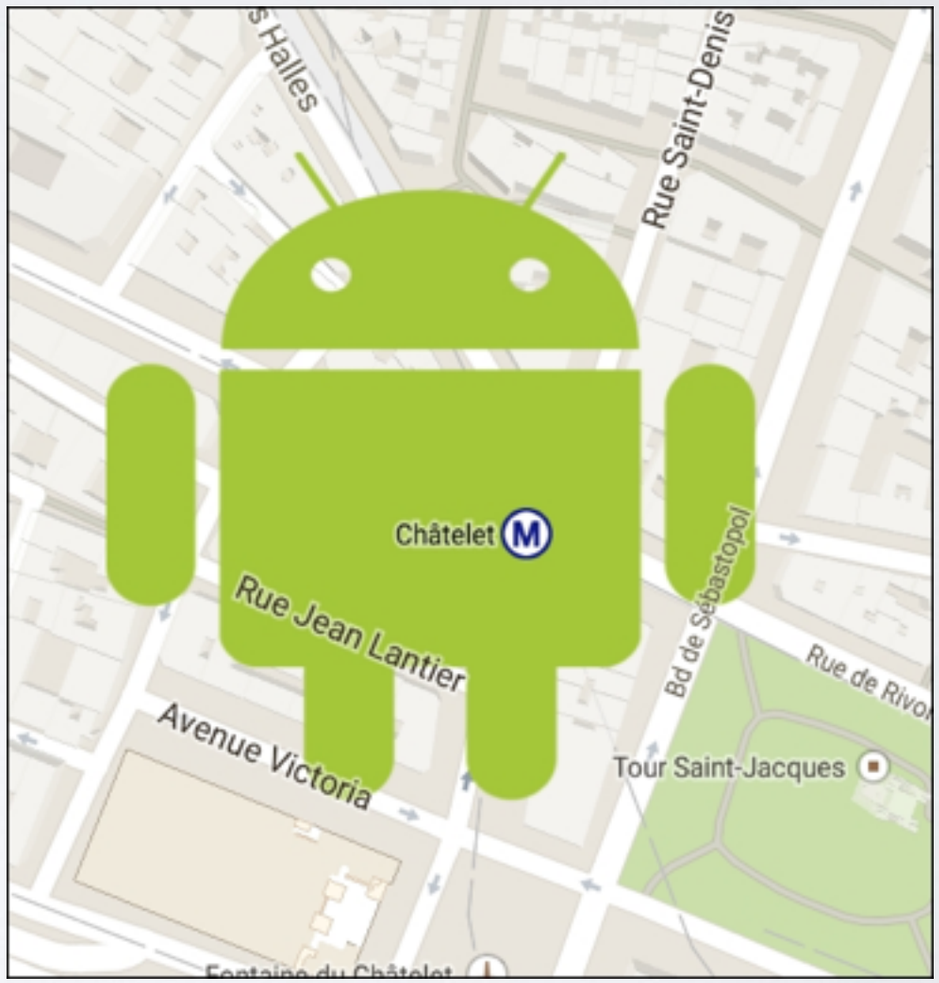


POLYGON

```
// Instantiates a new Polygon object and adds points to
// define a rectangle
PolygonOptions rectOptions = new PolygonOptions()
                .add(new LatLng(37.35, -122.0),
                     new LatLng(37.45, -122.0),
                     new LatLng(37.45, -122.2),
                     new LatLng(37.35, -122.2),
                     new LatLng(37.35, -122.0));

// Get back the mutable Polygon
Polygon polygon = myMap.addPolygon(rectOptions);
```

GROUND OVERLAY

- Image that is fixed to a map
- Oriented against the Earth's surface - rotating, tilting or zooming the map will change the orientation of the image
- Used for single image

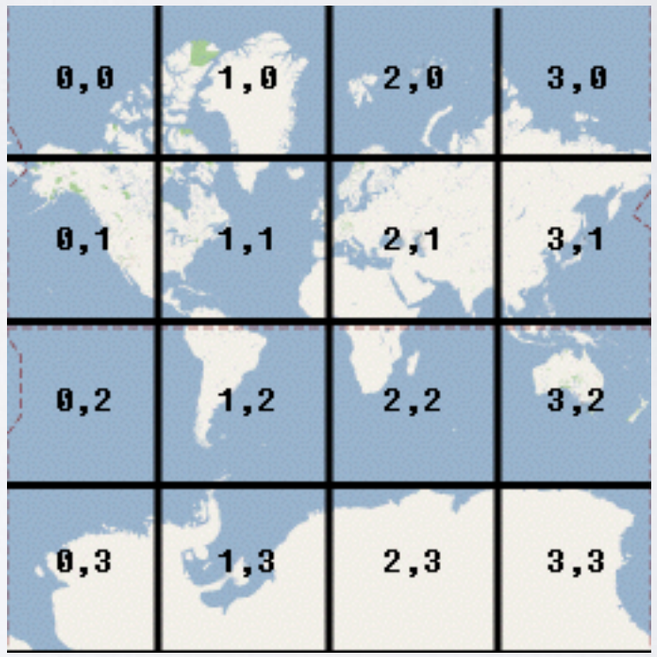


GROUND OVERLAY

```
LatLng NEWARK = new LatLng(40.714086, -74.228697);

GroundOverlayOptions newarkMap = new GroundOverlayOptions()
      .image(BitmapDescriptorFactory.fromResource(R.drawable.newark_nj_1922))
      .position(NEWARK, 8600f, 6500f);

// Add an overlay to the map, retaining a handle to the GroundOverlay
// object.
GroundOverlay imageOverlay = map.addGroundOverlay(newarkMap);
```

TILE OVERLAY

- Defines a set of images that are added on top of the base map tiles.
- Google Maps API breaks up the imagery at each zoom level into a set of square map tiles arranged in a grid.
- At zoom level the map will be rendered as a 2x2 grid of tiles, at zoom level 2, it's a 4x4 grid, at zoom level 3, it's an 8x8 grid, and so on.



TILE OVERLAY

```
private GoogleMap mMap;

TileProvider tileProvider = new UrlTileProvider(256, 256) {
  @Override
  public URL getTileUrl(int x, int y, int zoom) {

    /* Define the URL pattern for the tile images */
  String s = String.format("http://my.image.server/images/%d/%d/%d.png",
      zoom, x, y);

  if (!checkTileExists(x, y, zoom)) {
    return null;
  }

  try {
    return new URL(s);
  } catch (MalformedURLException e) {
      throw new AssertionError(e);
  }
}
```

TILE OVERLAY

```
/*
 * Check that the tile server supports the requested x, y and zoom.
 * Complete this stub according to the tile range you support.
 * If you support a limited range of tiles at different zoom levels, then you
 * need to define the supported x, y range at each zoom level.
 */
private boolean checkTileExists(int x, int y, int zoom) {
    int minZoom = 12;
    int maxZoom = 16;

    if ((zoom < minZoom || zoom > maxZoom)) {
      return false;
    }

    return true;
  }
};

TileOverlay tileOverlay = mMap.addTileOverlay(new TileOverlayOptions()
    .tileProvider(tileProvider));
```

CREDITS

All the code snippets and pictures come from:

https://developer.android.com/guide/topics/ui/index.html