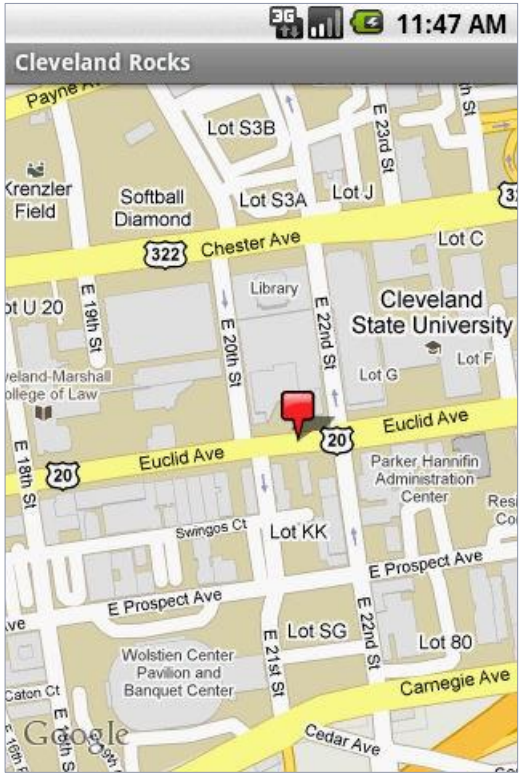
Maps API

Google Maps External Library

- Android uses the *Google Maps External Library* to add mapping capabilities to your applications.

- *Google Maps External Library* includes the **com.google.android.gms.maps package**. The classes of this package offer built-in *downloading*, *rendering*, and *caching* of Maps tiles, as well as a variety of *display option*s and *controls*.

- The key class is the **GoogleMap** class
- The map will be represented either by a **MapFragment** or a **MapView** object



Road View

https://developers.google.com/maps/documentation/android-api/map

The Map Object

GoogleMap automatically handles the following

- Connecting to Google Maps service
- Downloading map tiles
- Displaying tiles on the device screen
- Displaying controls (e.g. pan, zoom)
- Responding to gestures on the map

GoogleMap allows creation and control of map objects

Example Additions: Set marker icons, add overlays Example Controls: Clicks, swipes

The container object for the map and the **GoogleMap** object is the MapFragment

Adding a MapFragment

XML Defined:

```
<?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"/>
```

Code Defined:

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

Adding Map Code

To work with the map inside the application, OnMapReady callback must be implemented

Use getMapAsync() to set the callback on the fragment

The onMapReady(GoogleMap) callback returns a handle to the GoogleMap object

MapViews

MapView – subclass of View class
Allows pacement of a map in a View to act as container for the **GoogleMap** object

When using the API in "fully interactive mode" users of the MapView class must forward callbacks for:

- onCreate()
- onStart()
- onResume()
- onPause()
- onStop()
- onDestroy()
- onSaveInstanceState()
- onLowMemory()

Map Styles

The GoogleMap API allows four+1 styles of maps:

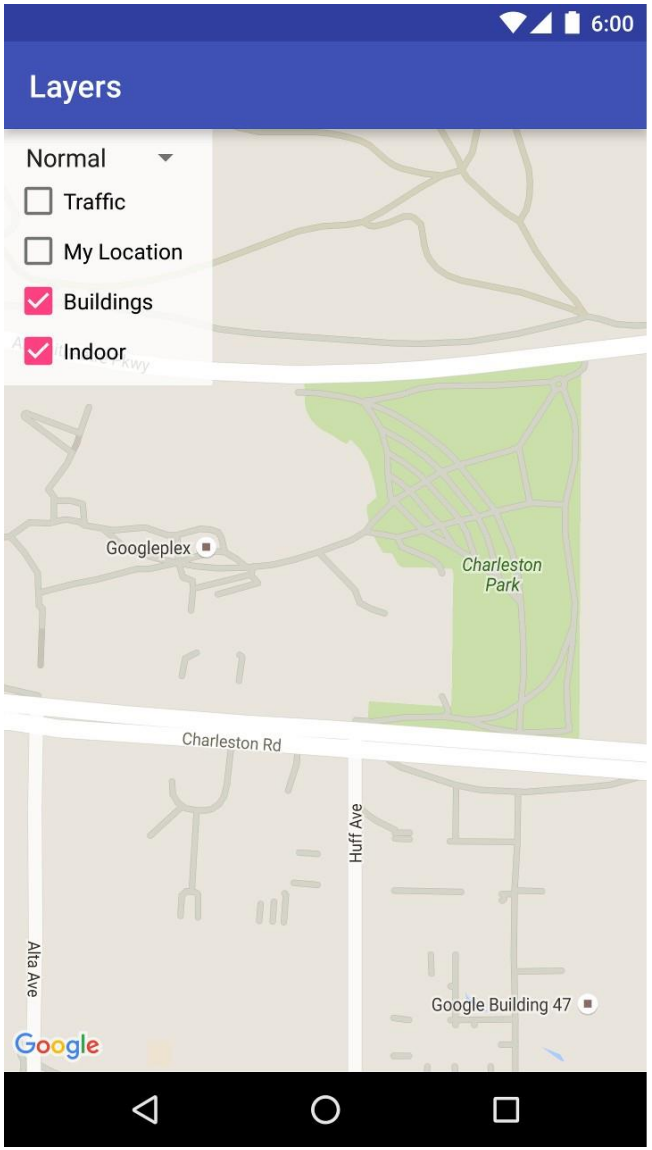
- Normal
- Hybrid
- Satellite
- Terrain
- None

Can choose by setting MapType

e.g. map.setMapType(GoogleMap.MAP TYPE HYBRID);

Map Style - Normal

Implemented as a road map
Shows roads, some features, as well as labels for roads and features



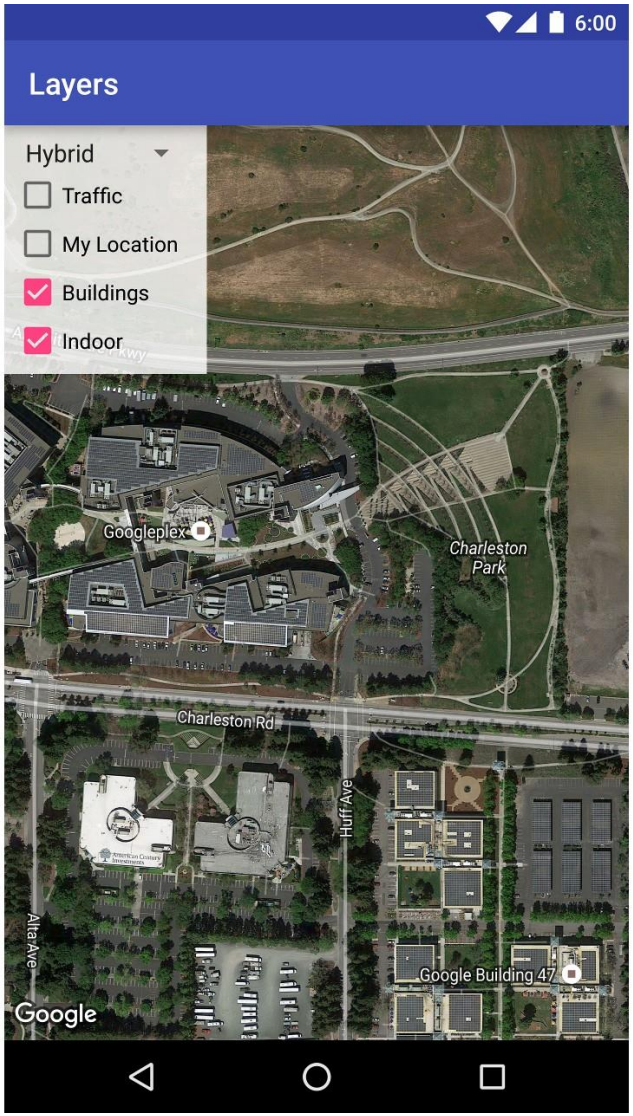
Map Style - Sattelite

Satellite photograph data
Road and feature labels are not visible.

Map Style - Hybrid

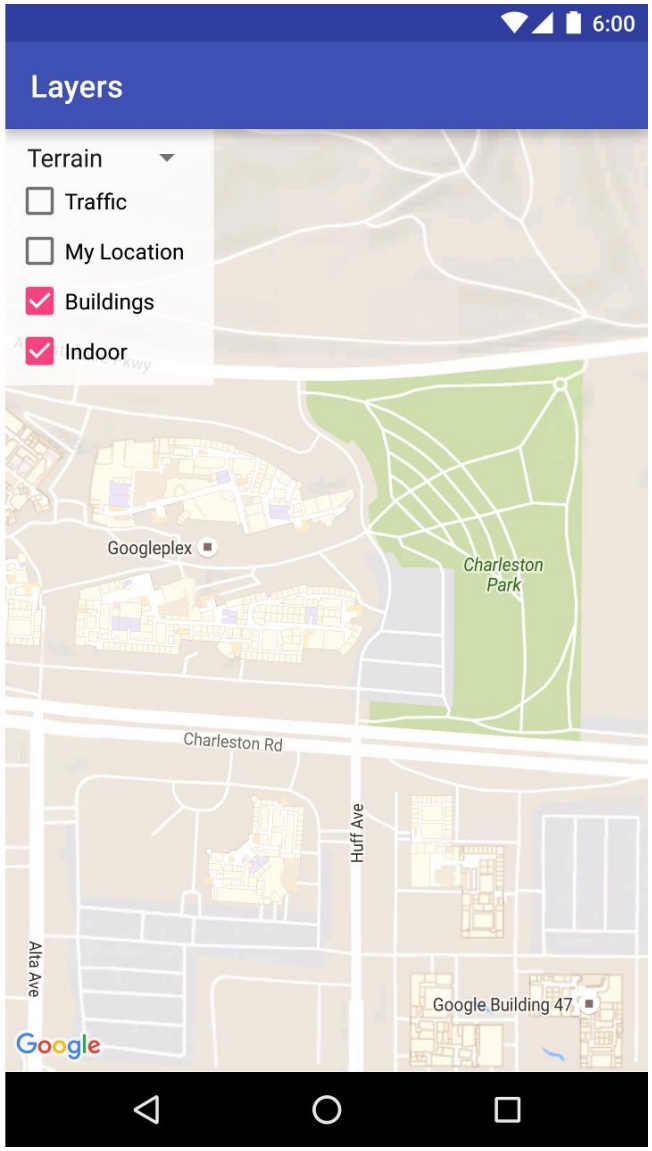
Satellite photograph data with road maps added

Road and feature labels are also visible



Map Style - Terrain

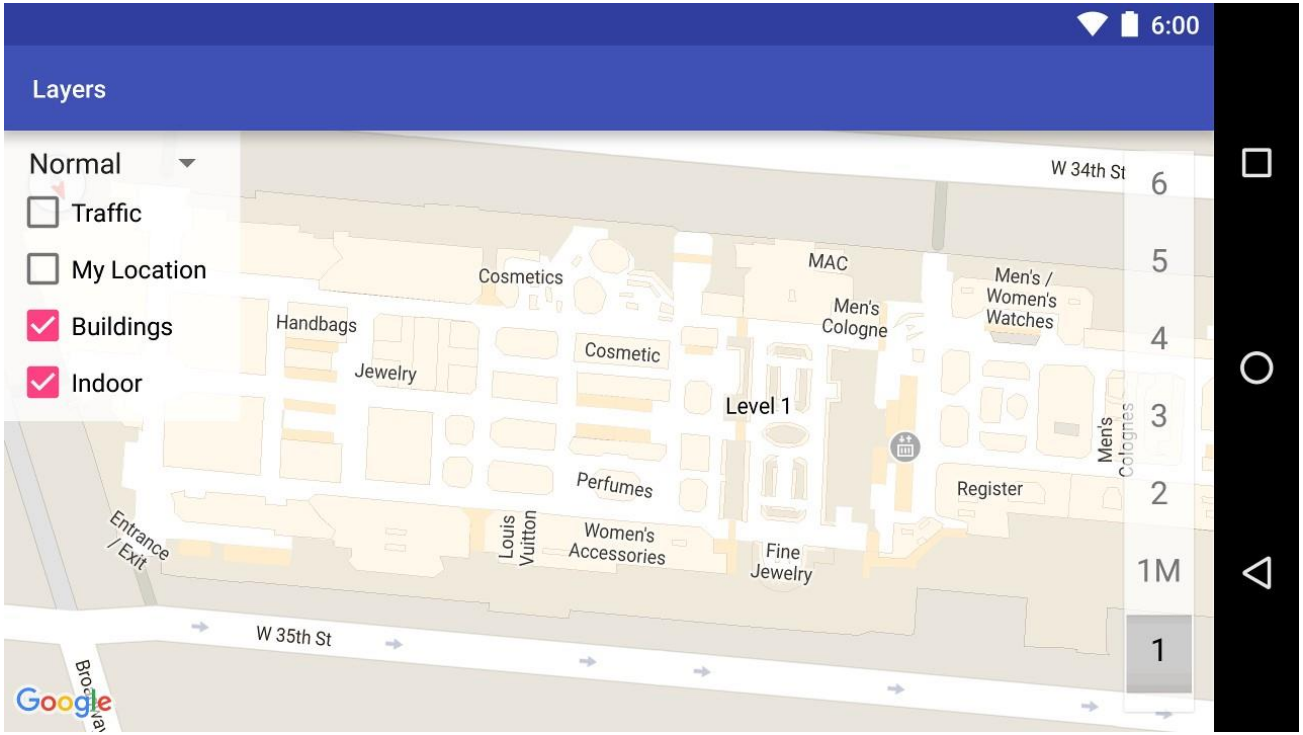
Topographic data
The map includes colors, contour lines and labels, and perspective shading
Some roads and labels are also visible



Indoor Maps

At high zoom levels, the Normal map type will show plans for indoor spaces

Indoor maps become automatically enabled when the user zooms in



Indoor Maps API

Can disable Indoor maps

e.g. map.setIndoorEnabled(false);

Indoor maps can be enabled only on one map at a time

- The first map added to the app by default

Must disable indoor on first map and enable on second map

- The Floor Picker can also be disabled
  GoogleMap.getUiSettings().setIndoorLevelPickerEnabled(false);

- Callback OnIndoorStateChangeListener allows event driven response to buildings coming into focus GoogleMap.getFocusedBuilding() gives the building that is currently in focus

Adding Indoor Maps

How can I add indoor maps to my application?

Two ways:

1. Add floor plans to Google Maps directly[1]
2. Display a floor plan as a ground or tile overlay on the map

[1] https://support.google.com/maps/answer/2803784

Initial State

The Maps API allows configuring the initial state of the map
You can specify the following:

- The Camera position including:
  - Zoom
  - Bearing
  - Tilt
- The Map type
- Whether zoom/compass appear on screen
- What gestures can be used to manipulate the camera
- Whether lite mode is enabled

Configuring Initial State

Two ways to configure the Map:

- Using custom XML attributes for MapFragment or MapView
- Programmatically with the Maps API

Configuring using XML

Before you can configure the attributes, you must add a namespace declaration:

xmlns:namespace="http://schemas.android.com/apk/res-auto"

```
<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"/>
```

Configuring Programmatically

GoogleMapOptions object[3] – Container object for all options

Get instance:

GoogleMapOptions myOptions = new GoogleMapOptions();

Then configure with attributes

e.g. myOptions.compassEnabled(false)

Finally, add Fragment or View and pass the options object as a parameter

e.g. MapFragment.newInstance(myOptions);

[3]https://developers.google.com/android/reference/com/google/android/gms/maps/G

Map Padding

The MapFragment and MapView objects are designed to use the entire region of the parent container object

This affects the behavior of the map elements:

- The camera center will be at the center of the container
- The Map Controls will be positioned to the edges
- Legal information is provided along the bottom

Often, you may wish to add elements on top of the map which allow control, e.g. Some sidebar with buttons

Map padding enables this behavior

Map Padding

- Map Padding enables adding space (padding) to the edges of a map
- Changes the behavior of map elements

- GoogleMap.setPadding(left,top,right,bottom);
- This method takes four integers for pixels to add to each of the four sides of the map

Styling the Map

Custom styling can be applied to the Normal map type
(Does not affect indoor maps)

Applied using a GoogleMapsObject and passing a JSON dictionary describing the style

Can play with options and create a style at
https://mapstyle.withgoogle.com/

Drawing on the Map

Markers

Markers – Indicate single locations on the map

Stylizing Markers:

- Can change the default color
- Can replace with a custom image
- Can apply info windows (especially during an onClick)

Add a Marker

Pass a MarkerOptions object to the addMarker function of the GoogleMap object

e.g. map.addMarker(new MarkerOptions().position(new LatLng(36.0822,94.172)).title("San Jose"));

Associating Data with a Marker

Aside from the default options, the Marker class can associate with an arbitrary object

Use Marker.setTag() & Marker.getTag() to set and get the object respectively

Getting Marker Click Events

By default, markers enable their onClick callback

Override the onMarkerClick(Marker) method

```
/** Called when the user clicks a marker. */
@Override
public boolean onMarkerClick(final Marker marker) {

    // Retrieve the data from the marker.
    Integer clickCount = (Integer) marker.getTag();

    // Check if a click count was set, then display the click count.
    if (clickCount != null) {
        clickCount = clickCount + 1;
        marker.setTag(clickCount);
        Toast.makeText(this,
                       marker.getTitle() +
                       " has been clicked " + clickCount + " times.",
                       Toast.LENGTH_SHORT).show();
    }
    return false;
}
```

Markers Options

There are many ways that Markers can be stylized

Find these at:

https://developers.google.com/android/reference/com/google/android/gms/maps/model/Marker

and

https://developers.google.com/maps/documentation/android-api/marker

Google API Key

To use the Google Map service an API key is needed. The API key can obtained as follows for **development/debugging** application:

1)Get **debug.keystore** file. In Android Studio, the debug keystore and certificate in **$HOME/.android/debug.keystore**, (On Windows: **C:\Users*user*\.android** and on MAC: **~/.android**/)

2)Use **keytool** tool to generate Certificate fingerprint (MD5). Use following command on command prompt

- keytool -list -alias androiddebugkey -keystore <keystore_location>.keystore -storepass android -keypass android

3)Go to 'Sign Up for the Android Maps API' page. Put your Certificate fingerprint (MD5) And get your API key for android GMap application.

4)Replace "**YOUR KEY HERE**" in Manifest with your API key.

To Do

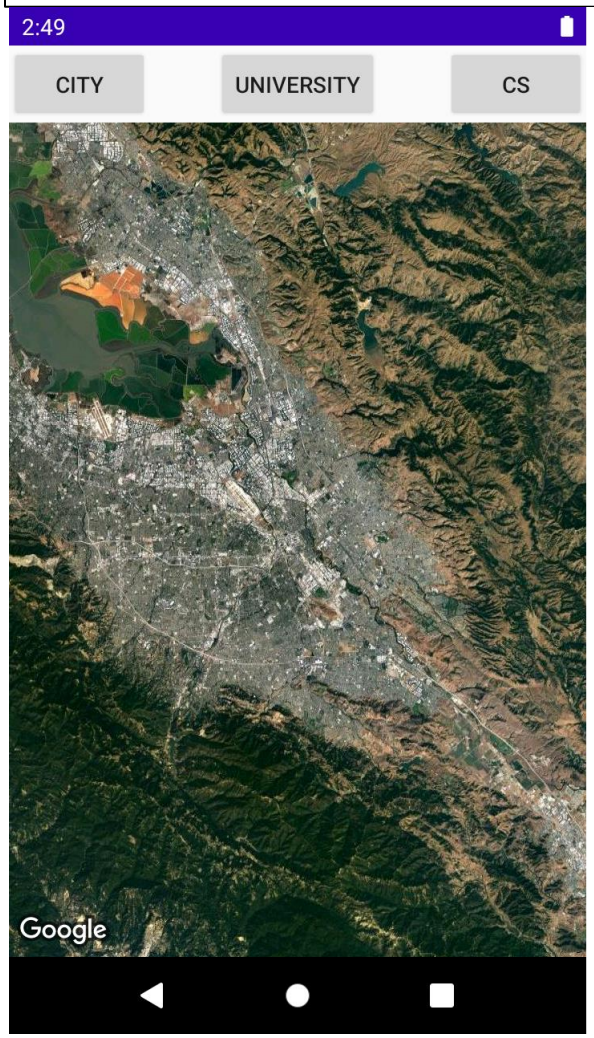
- Example 2: My Locations on Google Map

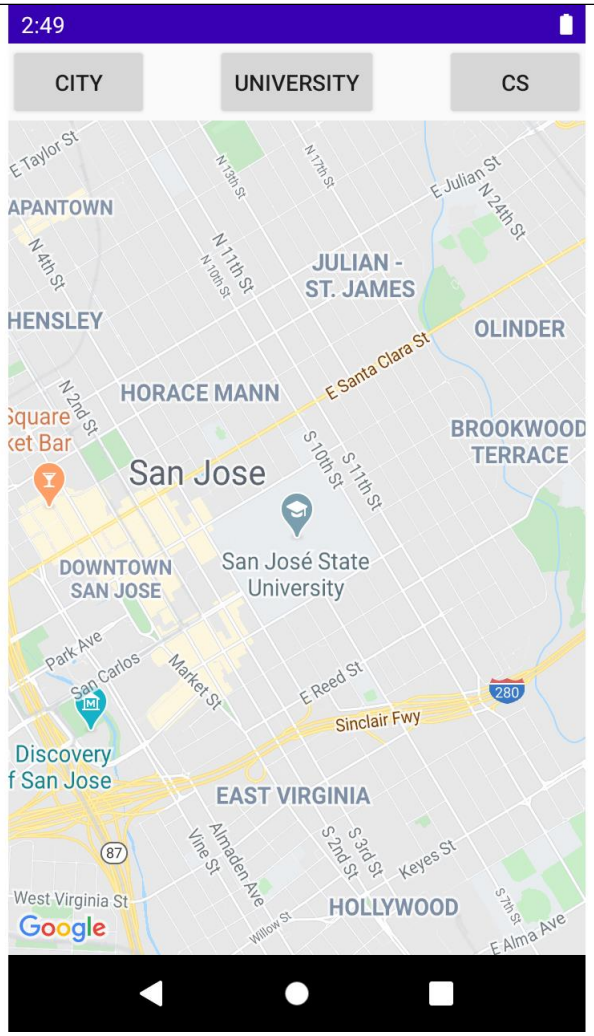
Get your API key by following this documentation:

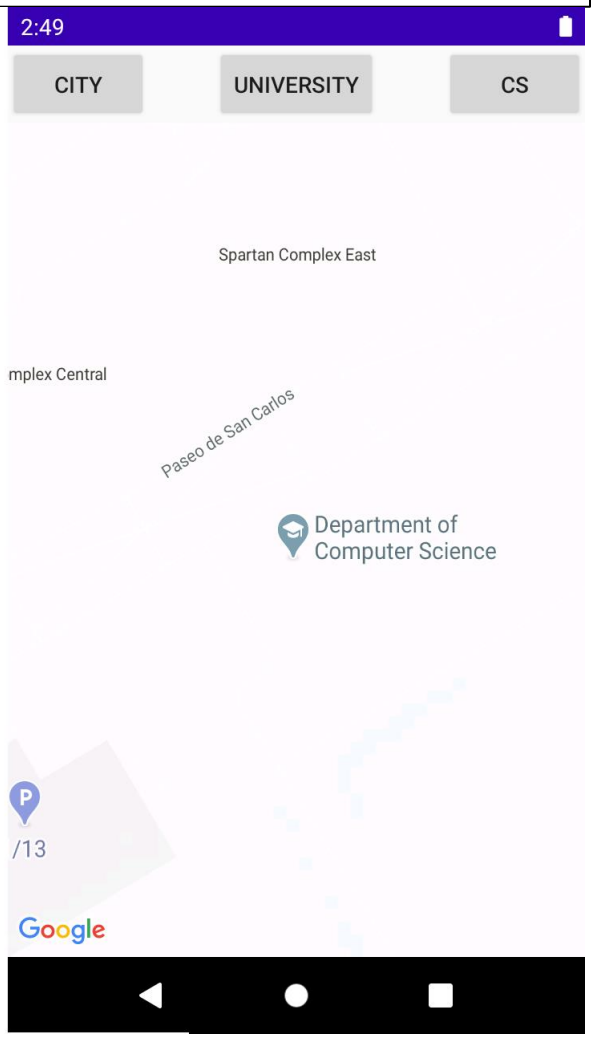
https://developers.google.com/maps/documentation/android/start#getting_the_google_maps_android_api_v2

Example

Example 2: My Locations on Google Map
Layout







Example 2: My Locations on Google Map

Layout

```
    <Button
        android:id="@+id/btnCS"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignEnd="@+id/map"
        android:onClick="onClick_CS"
        android:text="CS"/>

    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_below="@+id/btnCS"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <Button
        android:id="@+id/btnUniv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/btnCS"
        android:layout_centerHorizontal="true"
        android:onClick="onClick_Univ"
        android:text="University"/>

    <Button
        android:id="@+id/btnCity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/btnUniv"
        android:onClick="onClick_City"
        android:text="City"/>
</RelativeLayout>
```

Example 2: My Locations on Google Map

Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.googlemaptutorial"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="11"
        android:targetSdkVersion="19" />
    <permission
        android:name="com.example.googlemaptutorial.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />
    <uses-permission android:name="com.example.googlemaptutorial.permission.MAPS_RECEIVE"/
    <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"/>
    <!-- The following two permissions are not required to use
         Google Maps Android API v2, but are recommended. -->
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true"/>
```

Example 2: My Locations on Google Map

Manifest

```
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <meta-data
            android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="YOUR KEY HERE"/>
    </application>

</manifest>
```

Example 2: My Locations on Google Map

MapsActivity

```
import androidx.fragment.app.FragmentActivity;

import android.os.Bundle;
import android.view.View;

import com.google.android.gms.maps.*;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

    private final LatLng LOCATION_UNIV = new LatLng(37.335371, -121.881050);
    private final LatLng LOCATION_CS = new LatLng(37.333714, -121.881860);
    private GoogleMap map;

    @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) {
        map = googleMap;
        map.addMarker(new MarkerOptions().position(LOCATION_CS).title("Find Me Here!"));
    }
```

Example 2: My Locations on Google Map

MainActivity

```
public void onClick_CS(View v) {
    map.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
    CameraUpdate update = CameraUpdateFactory.newLatLngZoom(LOCATION_CS, 18);
    map.animateCamera(update);
}

public void onClick_Univ(View v) {
    map.setMapType(GoogleMap.MAP_TYPE_NORMAL);
    CameraUpdate update = CameraUpdateFactory.newLatLngZoom(LOCATION_UNIV, 14);
    map.animateCamera(update);
}

public void onClick_City(View v) {
    map.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
    CameraUpdate update = CameraUpdateFactory.newLatLngZoom(LOCATION_UNIV, 10);
    map.animateCamera(update);
}
```