Locations in Android: Some Updates
Location

- My slides: Covered Android.location:
  - As I mentioned, Google would prefer you NOT use this way to access location
  - But used by most books, available code

- Preferred way: Google Location Services API. Can retrieve
  - Geographical location (latitude, longitude)
  - location updates at regular intervals using requestLocationUpdates()

- Can also retrieve location object using fused location provider
  - Contains bearing (direction of horizontal travel), altitude, velocity
Location

- Official Google documentation for Google Location Services API looks good, adequate
  - Overview: https://developer.android.com/training/location
  - Request location permissions: https://developer.android.com/training/location/permissions
  - Get last known location: https://developer.android.com/training/location/retrieve-current
  - Change location settings (e.g. GPS vs WiFi): https://developer.android.com/training/location/change-location-settings
  - Request location updates: https://developer.android.com/training/location/request-updates
  - Access location in background: https://developer.android.com/training/location/background
GeoFencing in Android: Some Updates
GeoFencing: Old Way

- **Old way**: `GeofencingApi` deprecated
- Code sample in Android studio implements old way unfortunately
- `GeofencingApi` typically used in conjunction with a `GoogleApiClient`

```java
new GoogleApiClient.Builder(context)
        .addApi(LocationServices.API)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .build()
```
GeoFencing: New Way

- New way: GeofencingClient
- Create, start monitoring geoFences
  - Need to create instance of `GeofencingClient`
- Specify GeoFences using:
  - GeofencingRequest
  - GeofencingRequestBuilder
- Create broadcast receiver to be notified of geofence transitions
- Add geofences using `GeofencingClient.addGeofences()`
- Remove geofences using `geofencingClient.removeGeofences()`
GeoFencing

- Official Google documentation
  - https://developer.android.com/training/location/geofencing
  - https://developers.google.com/location-context/geofencing

- Good reference articles with good examples, gentle walkthrough:
  - https://techpaliyal.com/android-geofencing/
MediaPlayer in Android: Minor Updates
MediaPlayer

- Main API (MediaPlayer) is same
- Slight changes in some methods. Needs to be updated.
  - E.g Now set audio attributes using mediaPlayer.setAudioAttributes(.. )
  - Also material on WakeLocks (Power savings), etc

- Official Google documentation (looks good), adequate documentation:
  - https://developer.android.com/guide/topics/media/mediaplayer
Using Maps
MapView and MapActivity

- **MapView**: UI widget that displays maps
- **MapActivity**: java class (extends Activity), handles map-related lifecycle and management for displaying maps.
7 Steps for using Google Maps Android API
https://developers.google.com/maps/documentation/android-api/start

1. Install Android SDK (Done!!)
2. Add Google Play services to Android Studio
3. Create a Google Maps project
4. Obtain Google Maps API key
5. Hello Map! Take a look at the code
6. Connect an Android device
7. Build and run your app
Step 2: Add Google Play Services to Android Studio
https://developers.google.com/maps/documentation/android-api/start

- Google Maps API v2 is part of Google Play Services SDK
- Use Android Studio SDK manager to download Google Play services
Step 3: Create new Android Studio Project

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

- Select “Google Maps Activity, click Finish
Step 4: Get Google Maps API key

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

- To access Google Maps servers using Maps API, must add Maps API key to app
- Maps API key is free. E.g.

![Your API key](image)

- Google uses API key to uniquely identify your app, track its resource usage, etc
Step 4a: Fast, Easy way to get Maps API Key
https://developers.google.com/maps/documentation/android-api/start

- Copy link provided in `google_maps_api.xml` of Maps template into browser
- Goes to Google API console, auto-fills form
- Creates API key
Step 4a: Fast, Easy way to get Maps API Key

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

- If successful, Maps API key generated
  - API key created
    - Use this key in your application by passing it with the key=API_KEY parameter.
    - Your API key: AIzaSyCc0_1EEjPl1TlnPkJVsX0YiY7oBa9XsXs
      - Restrict your key to prevent unauthorized use in production.
  - Close or restrict key

- Copy key, put it in <string> element in google_maps_api.xml file

```xml
<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AIzaSyCc0_1EEjPl1TlnPkJVsX0YiY7oBa9XsXs</string>
```
Step 4b: Longer (older) way to API key

- If easy way doesn’t work, older way to obtain a Maps API key
- Follow steps at:
  - See: https://developers.google.com/maps/documentation/android-api/signup
Step 5: Examine Code Generated by Android Studio Maps Template

- XML file that defines layout is in res/layout/activity_maps.xml

```xml
<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"/>
```
Step 5: Examine Code Generated by Android Studio Maps Template

- Default Activity file is `MapActivity.java`

```java
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapActivity extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;

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

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

        // Add a marker in Sydney, Australia, and move the camera.
        LatLng sydney = new LatLng(-34, 151);
        mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }
}
```
Steps 6, 7

- **Step 6:** Connect to an Android device (smartphone)

- **Step 7:** Run the app
  - Should show map with a marker on Sydney Australia

- More code examples at:
  - https://github.com/googlemaps/android-samples
AsyncTask API
AsyncTask API

- For compute intensive tasks, remote or tasks that take a long time, doing it in main activity blocks
- **AsyncTask**: spawn separate thread to offload such task, free up main Activity

One thread = Frustrated user!
What other Android APIs may be useful for Mobile/ubicomp?
Speaking to Android

https://developers.google.com/voice-actions/

- **Speech recognition:**
  - Accept inputs as speech (instead of typing) e.g. dragon dictate app?
  - Note: Requires internet access

- **Two forms**
  1. **Speech-to-text**
     - Convert user’s speech to text. E.g. display voicemails in text
  2. **Voice Actions:** Voice commands to smartphone (e.g. set alarm)
Google Voice Actions
https://developers.google.com/voice-actions/

- E.g. Tell Google to set an alarm
Gestures

https://developer.android.com/training/gestures/index.html

- **Gesture:** Hand-drawn shape on the screen, swipe pattern
- **Example uses:**
  - Search your phone, contacts, etc by handwriting onto screen
  - Speed dial by handwriting first letters of contact’s name
  - Multi-touch, pinching
More MediaPlayer & RenderScript

- MediaRecorder is used to **record** audio
  - Manipulate raw audio from microphone/audio hardware, PCM buffers
    - E.g. if you want to do audio signal processing, speaker recognition, etc
    - **Example**: process user’s speech, detect emotion, nervousness?
  - Can playback recorded audio using MediaPlayer

- **RenderScript**
  - High level language for computationally intensive tasks/GPGPU,
  - Can be used to program phone CPU, GPU in a few lines of code
  - Use Phone’s Graphics Processing Unit (GPU) for computational tasks
  - Useful for heavy duty tasks. E.g. image processing, computational photography, computer vision
Wireless Communication


- **Bluetooth**
  - Discover, connect to nearby bluetooth devices
  - Communicating over Bluetooth
  - Exchange data with other devices
  - Killer app now: COVID contact tracing,
    Too Close for Too Long (< 6 ft for > 15 mins)

- **WiFi**
  - Scan for WiFi hotspots
  - Monitor WiFi connectivity, Signal Strength (RSSI)
  - Do peer-to-peer (mobile device to mobile device) data transfers
Wireless Communication


- **NFC:**
  - Contactless, transfer small amounts of data over short distances
  - **Applications:** Share spotify playlists, Google wallet
  - **Android Pay**
    - Store debit, credit card on phone
    - Pay by tapping terminal
Telephony and SMS


- **Telephony:**
  - Initiate phone calls from within app
  - Access dialer app, etc

- **SMS:**
  - Send/Receive SMS/MMS from app
  - Handle incoming SMS/MMS in app
Google Play Services: Nearby Connections API
https://developers.google.com/nearby/connections/overview

- Peer-to-peer networking API, allows devices communicate over a LAN
- One device serves as host, advertises
- Other devices can discover host, connect, disconnect
- **Use case:** Multiplayer gaming, shared virtual whiteboard
Google Android Samples

- Android Studio comes with many sample programs
- Just need to import them
Google Android Samples

- Can click on any sample, read overview
- Source code available on github
- Tested, already working
- **Note:** Some code may use deprecated APIs
Other 3rd Party Stuff
http://web.cs.wpi.edu/~emmanuel/courses/ubicomp_projects_links.html
https://developer.qualcomm.com/software/trepn-power-profiler

- **MPAndroid**: Add charts to your app

- **Trepn**: Profile power usage and utilization of your app (CPU, GPU, WiFi, etc)
  - By Qualcomm
Other 3rd Party Stuff
http://web.cs.wpi.edu/~emmanuel/courses/ubicomp_projects_links.html

- **Programmable Web APIs:** 3rd party web content (e.g. RESTful APIs) you can pull into your app with few lines of code
  - **Weather:** Weather channel, yahoo weather
  - **Shared interests:** Pinterest
  - **Events:** Evently, Eventful, Events.com
  - **Photos:** flickr, Tumblr
  - **Videos:** Youtube
  - **Traffic info:** Mapquest traffic, Yahoo traffic

- **E.g. National Geographic:** picture of the day
More Android APIs for Mobile Computing

- Depth Sensing: Project Tango (Dead? Delete?)
- MPAndroid: 3\textsuperscript{rd} party charts
- Trepn: Measure resource consumption (power, CPU, GPU, etc)
- Programmable Web APIs: E.g. National Geographic API, new picture in your app daily
- Augmented Reality: ARtoolkit, vuforia, EasyAR
- Mobile Commerce:
  - Android Pay
  - Analytics
  - Advertising: E.g. Adwords, Admobs

- Other Google APIs (that could be used by mobile devices):
  - Google Fit: Health and fitness, nutrition, steps, etc
  - Google Cast: allows screen-sharing
More Android APIs for Mobile Computing

- Mobile Communication:
  - Wireless Communication: Bluetooth, WiFi, NFC, etc
  - Telephone/SMS
  - Nearby Connections API

- Mobile Cloud:
  - Google Drive API, Google cloud, etc

- Mobile computation:
  - Renderscript: Easy computational programming (smartphone GPU, CPU)
Other Mobile Technology

- Mobile programming/development:
  - Kotlin
  - iPhone development
  - 3rd part libraries, app frameworks: Xamarin, flutter, ionic, etc
  - Mobile web programming
  - PhoneGap
  - AppInventor
  - Mobile game development tools: Unity,

- Machine/Deep Learning:
  - Deep Learning/machine learning in Android: Tensorflow, etc
  - Mobile machine/deep learning support in MATLAB
  - Keras support for Android Deep learning
  - Neural Networks API (NNAPI)
References

- John Corpuz, 10 Best Location Aware Apps
- Liane Cassavoy, 21 Awesome GPS and Location-Aware Apps for Android,
- Head First Android
- Android Nerd Ranch, 2nd edition
- Busy Coder’s guide to Android version 6.3
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014