# Chapter 11.

# Sensor & Location Service

**2023-2024**
**COMP7506 Smart Phone Apps Development**
**Dr. T.W. Chim (E-mail: twchim@cs.hku.hk)**
**Department of Computer Science, The University of Hong Kong**

# Agenda

- Overview of Sensors
- Motion Sensors
- Position Sensors
- Environmental Sensors
- Location Service
- Google Map

## Overview of Sensors

## Sensors

- Device components sensing various environment factors
  - Main difference from stationary devices (desktop, laptop)
- Useful for many applications
  - Game, weather, travel, …
- Some sensors are common, some are not
  - Your app needs to check the device sensors before using them!
    - Availability
    - Capability
- Emulators cannot emulate sensors!
  - Need to test your app on a real device

## Supplement: Testing using Real Android Devices

For older Android versions:

- In the "AndroidStudioProjects\<Project Name>\app\build\outputs\apk" folder, you should find the apk file "app-debug.apk"
- Copy the apk file into your Android phone for installation.

For newer Android phones:

- Cannot install from apk files directly.
- Access Developer Options and enable USB Debugging on Android device (https://www.howtogeek.com/129728/how-to-access-the-developer-options-menu-and-enable-usb-debugging-on-android-4.2/).
- E.g. For Samsung phones, "Settings" → "About Phone" → "Software Information" → tap "Internal Version Number / Build Number" 7 times, you should see "Developer options" when going back to "Settings".
- Connect your computer and the Android device using USB cable.
- When playing the application, choose to open in real device (you should see information about your Android device).

## Types

- Motion sensors
  - Linear / angular device velocity / acceleration
    - e.g., accelerometer, gravity, linear acceleration, gyroscope, rotational vector
  - Footsteps
- Position sensors
  - Physical position / orientation of device (relative to earth or user)
    - e.g., magnetometer, orientation, proximity
- Environment sensors
  - Environmental parameters: temperature, pressure, light, humidity
    - e.g., barometer, photometer, thermometer

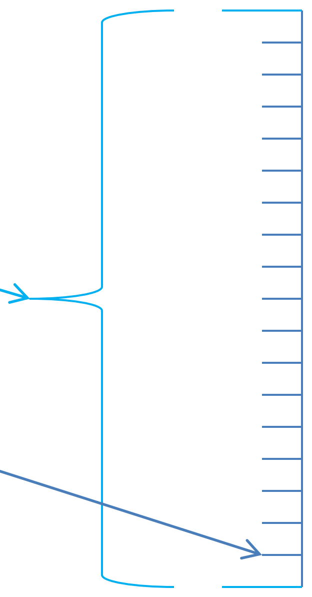
## Implementation

- Hardware sensor
  - Built-in physical components
  - Derive data directly from measurements
  - Examples: accelerometer, gyroscope
- Software sensor
  - Derive data from hardware sensor(s)
  - Examples: linear acceleration, gravity
  - Facilitate app coding and development
  - Also called
    - Virtual sensor
    - Synthetic sensor

Example: An accelerometer can measure the static gravitation field of earth + linear acceleration. The force of gravity can be isolated with a low-pass filter and then the gravity can be subtracted from the raw signal. This has the same effect as creating a high-pass filter that isolates the linear acceleration.

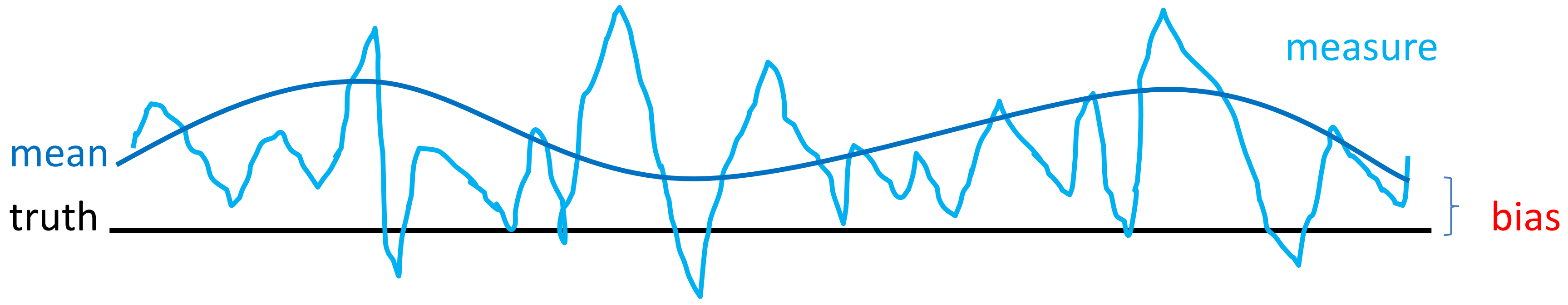
## Characteristics

- Availability
  - Do not assume every device has every sensor!
- Capabilities
  - Maximum range
  - Resolution
  - Update rate
  - Manufacturer
  - Power consumption (very important)



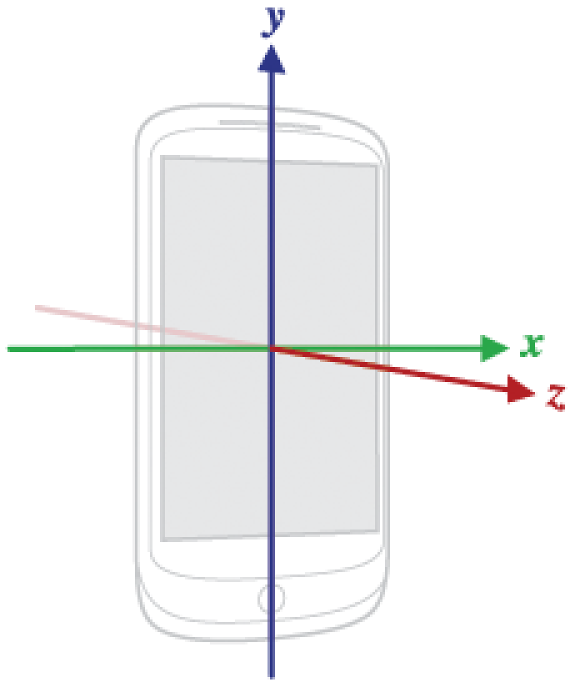
# Noise & Bias

- Two very important concepts about physical quantities
- Noise
  - Variations from ground truth
- Bias
  - Systematic deviation from ground truth
- Most physical sensors contain noise and bias

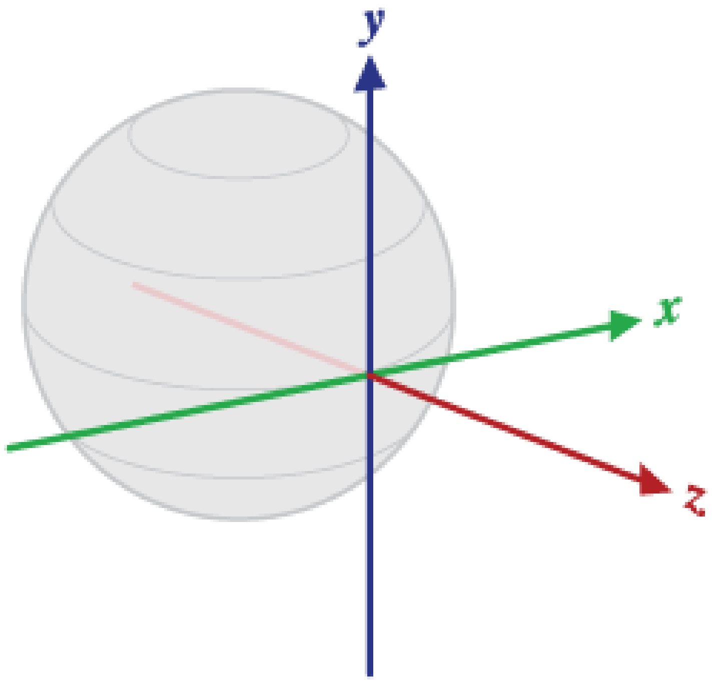


## Coordinate System

- Device coordinate system, local space



- World coordinate system, world space



Important for motion and position sensors

# App Consideration

- Requirement
  - Mandatory or optional?
  - AndroidManifest.xml

```
<uses-feature android:name="android.hardware.sensor.accelerometer" android:required="false" />
```

*Note:*

android:required="true“: You are specifying that the application cannot function, or is not designed to function, when the specified feature is not present on the device.

android:required="false“: It means that the application prefers to use the feature if present on the device, but that it is designed to function without the specified feature, if necessary.

- Resource consumption
  - Sensors can be power hogs
  - Mobile apps need to be very power aware
- Trade offs
  - App should work on as many devices as possible
  - App should perform more and consume less on each device

## Availability

- Depends on device and OS version
- Few devices have every type of sensor
  - Can have 0, 1, or multiple sensors per type
- Common
  - Accelerometer, gyroscope, proximity sensor
- Less common
  - Thermometer
- Sensors available since Android 4.0:
  - Accelerometer, gravity, gyroscope, linear acceleration, magnetic field, orientation, rotation vector, pressure, proximity, humidity, light, device temperature, ambient temperature
- Sensors available since iPhone X:
  - Accelerometer, gyroscope, barometer, proximity sensor, ambient light sensor, digital compass. iPhone 12 and 12 Mini also include the Face ID sensor for biometric authentication.

## Android

## Sensor Programming Framework

- Sensor
  - Instance of sensor, sensor capability
- SensorManager
  - Listing + accessing sensors
  - Registering / unregistering event listeners
- SensorEvent
  - Raw sensor data, sensor type, accuracy, timestamp
- SensorEventListener
  - Sensor value change
  - Sensor accuracy change

## Identify Sensor and Capabilities

```
SensorManager sensor_manager = (SensorManager) getSystemService(SENSOR_SERVICE);

List<Sensor> sensors = sensor_manager.getSensorList(Sensor.TYPE_ALL);

for (int i = 0; i < sensors.size(); i++)
{
        Sensor sensor = sensors.get(i);
        String name = sensor.getName();
        float resolution = sensor.getResolution();
        float range = sensor.getMaximumRange();
        float power = sensor.getPower();   // The power in mA used by this sensor while in use.
}
```

## Identify Specific Sensor

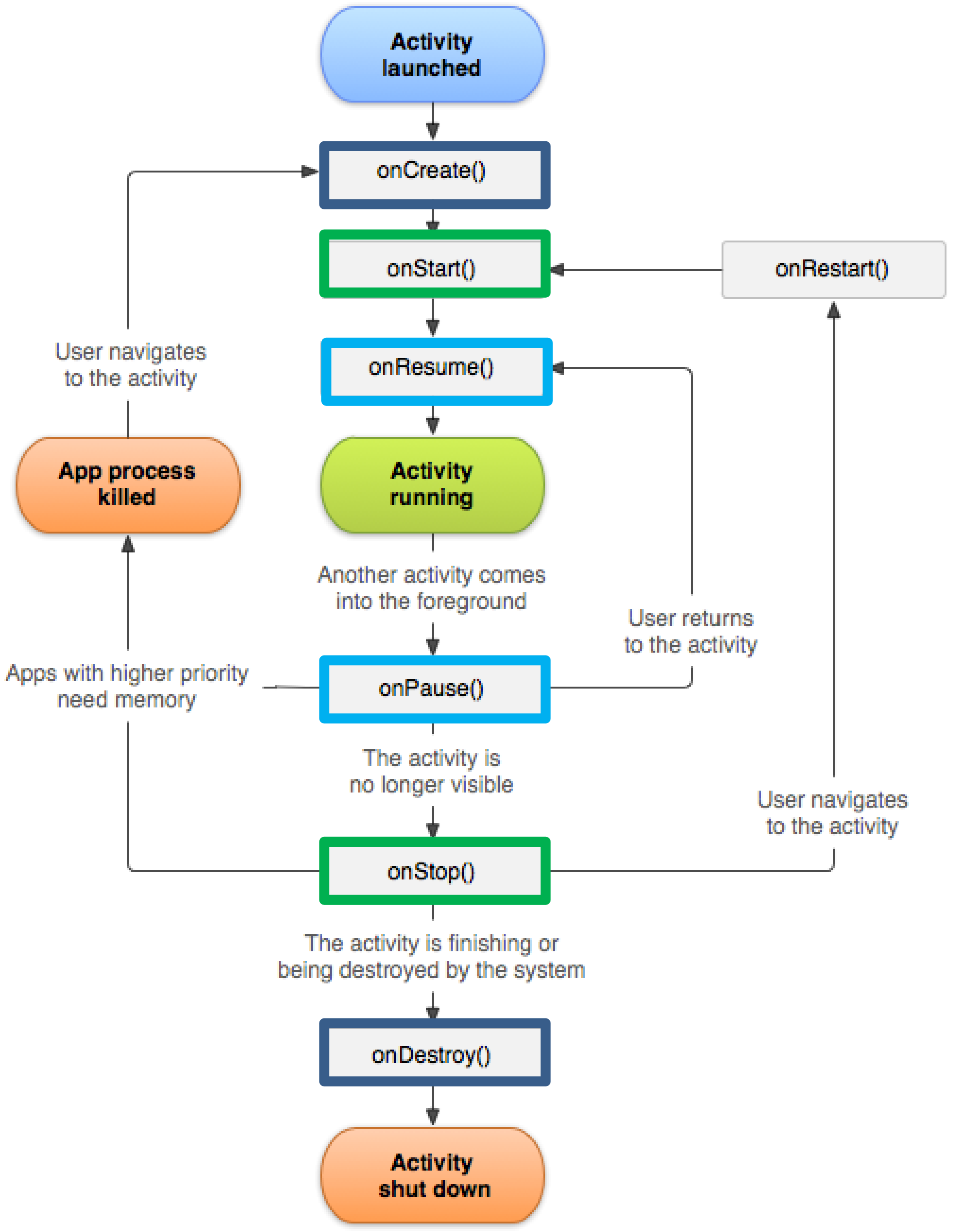
```
Sensor accelerometer_sensor =
sensor_manager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

if (accelerometer_sensor == null) {
		// sensor not available
}
else {
		// sensor available
}
```

## Monitor Sensor Events

```
SensorEventListener listener = new SensorEventListener() {
        @Override
        public void onSensorChanged(SensorEvent event) {
                float[] values = event.values;   // different contents for different types of sensors
                …
        }
        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {
                // Called when the accuracy of a sensor has changed.
                // Sensor: The sensor you instantiated + registered
                // Accuracy:
                // - SENSOR_STATUS_ACCURACY_LOW
                // - SENSOR_STATUS_ACCURACY_MEDIUM
                // - SENSOR_STATUS_ACCURACY_HIGH
                // - SENSOR_STATUS_UNRELIABLE
        }
};
```

## Register and Unregister Listener

- Use onResume() and onPause(), not onCreate()
- Power consumption issue!



Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

## Register and Unregister Listener

```
public void onResume() {
        if (listener == null) {
                sensor_manager.registerListener(listener,
                accelerometer_sensor,
                SensorManager.SENSOR_DELAY_NORMAL);
        }
}
public void onPause() {
        if (listener != null) {
                sensor_manager.unregisterListener(listener);
        }
}
```

## Best Practice

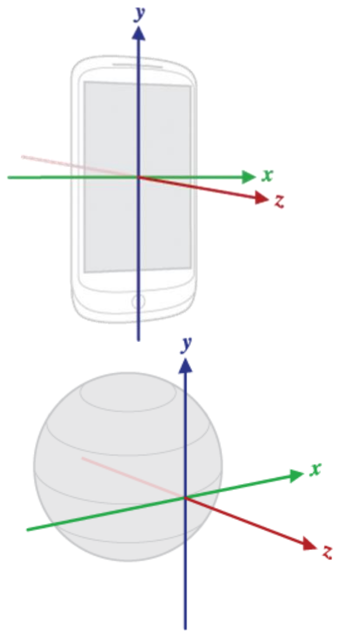
- Register / unregister event listeners at onResume() / onPause()
  - By default, sensors continue to run even your app is paused (thus power drain).
- Test code on real device, not emulator
- Don’t block onSensorChanged()
- Avoid deprecated sensor types or methods
- Verify sensors before using them
  - Availability & capability

# Motion Sensors

| Sensors | Type Name | Hardware / Software | SensorEvent.values |
|---|---|---|---|
| Accelerometer | TYPE_ACCELEROMETER | Hardware | [0, 1, 2]: Acceleration along x, y, z axis in $m/s^2$ |
| Gravity | TYPE_GRAVITY | Software | [0, 1, 2]: Gravitational acceleration along x, y, z axis in $m/s^2$ |
| Linear Acceleration | TYPE_LINEAR_ ACCELERATION | Software | [0, 1, 2]: Non-gravitational acceleration along x, y, z axis in $m/s^2$ |
| Gyroscope | TYPE_GYROSCOPE | Hardware | [0, 1, 2]: Rotation speed along x, y, z axis in radian/s |
| Rotation Vector | TYPE_ROTATION_ VECTOR | Software | [0, 1, 2]: Rotation vector along x, y, z axis: [x, y, z] * sin(angle / 2) [3]: Scalar component of rotation vector: cos(angle/2) (optional) (unit-less) |

## Applications of Motion Sensors

- Device movements
  - Tilt, shake, rotation, swing
  - Relative to user
    - e.g. moving device by hand
  - Relative to environment
    - e.g. inside a car the user is driving
- Applications
  - Sports (e.g. Wii-like controller)
  - Simulation (e.g. steering wheel)
  - Gaming



# Position Sensors

| Sensors | Type Name | Hardware / Software | SensorEvent.values |
|---|---|---|---|
| Proximity | TYPE_PROXIMITY | Hardware | [0]: Distance between device and the nearest object in cm |
| Magnetic Field | TYPE_MAGNETIC_FIELD | Hardware | [0, 1, 2]: Magnetic strength along x, y, z axis in micro tesla |

## Applications of Position Sensors

- Proximity sensor
  - Tune volume according to user distance
  - Turn off the monitor when you place the phone close to your ear
  - Smart phone games
- Geomagnetic field sensor
  - Orientation
  - Compass

# Environmental Sensors

| Sensors | Type Name | Hardware / Software | SensorEvent.values |
|---|---|---|---|
| Ambient Temperature | TYPE_AMBIENT_ TEMPERATURE | Hardware | [0]: Temperature in °C |
| Device Temperature | TYPE_TEMPERATURE (Deprecated in Android 4.0) | Hardware | [0]: Device temperature in °C |
| Light | TYPE_LIGHT | Hardware | [0]: Illumination in lux |
| Air Pressure | TYPE_PRESSURE | Hardware | [0]: Pressure in hPa or mbar |
| Humidity | TYPE_HUMIDITY | Hardware | [0]: Relative humidity in % |

## Applications of Environmental Sensors

- Device management
  - Light for adjusting screen brightness
  - Temperature for ventilation?
- Simulation
  - Plant growth
- Travel
  - Weather, altitude, compass

## Reference

- You may check the following sites for a full list of sensors available in an Android device:
  - https://developer.android.com/guide/topics/sensors/sensors_motion.html
  - https://developer.android.com/guide/topics/sensors/sensors_position.html
  - https://developer.android.com/guide/topics/sensors/sensors_environment.html

## iOS

## Core Motion

- Reports motion- and environment-related data from the onboard hardware of iOS devices
- Accesses hardware-generated data for using in app
- Examples: Accelerometers, gyroscopes, pedometer (counts each step a person takes by detecting the motion of the person’s hands or hips), magnetometer, and barometer.
- Sample usage: A game might use accelerometer and gyroscope data to control onscreen game behavior.

## CMMotionManager : NSObject

- For receiving 4 types of motion data:
  - Accelerometer data, indicating the instantaneous acceleration of the device in 3-dimensional space.
  - Gyroscope data, indicating the instantaneous rotation around the device's three primary axes.
  - Magnetometer data, indicating the device's orientation relative to Earth's magnetic field.
  - Device-motion data, indicating key motion-related attributes such as the device's user-initiated acceleration, its attitude, rotation rates, orientation relative to calibrated magnetic fields, and orientation relative to gravity. This data is provided by Core Motion’s sensor fusion algorithms.

# Accelerometer & Gyroscope Examples

```
let motion = CMMotionManager()

func startAccelerometers() {
   // Make sure the accelerometer hardware is available.
   if self.motion.isAccelerometerAvailable {
      self.motion.accelerometerUpdateInterval = 1.0 / 60.0  // 60 Hz
      self.motion.startAccelerometerUpdates()

      // Configure a timer to fetch the data.
      self.timer = Timer(fire: Date(), interval: (1.0/60.0),
            repeats: true, block: { (timer) in
         // Get the accelerometer data.
         if let data = self.motion.accelerometerData {
            let x = data.acceleration.x
            let y = data.acceleration.y
            let z = data.acceleration.z

            // Use the accelerometer data in your app.
         }
      })

      // Add the timer to the current run loop.
      RunLoop.current.add(self.timer!, forMode: .defaultRunLoopMode)
   }
}
```

```
func startGyros() {
   if motion.isGyroAvailable {
      self.motion.gyroUpdateInterval = 1.0 / 60.0
      self.motion.startGyroUpdates()

      // Configure a timer to fetch the accelerometer data.
      self.timer = Timer(fire: Date(), interval: (1.0/60.0),
             repeats: true, block: { (timer) in
         // Get the gyro data.
         if let data = self.motion.gyroData {
            let x = data.rotationRate.x
            let y = data.rotationRate.y
            let z = data.rotationRate.z

            // Use the gyroscope data in your app.
         }
      })

      // Add the timer to the current run loop.
      RunLoop.current.add(self.timer!, forMode: .defaultRunLoopMode)
   }
}

func stopGyros() {
   if self.timer != nil {
      self.timer?.invalidate()
      self.timer = nil

      self.motion.stopGyroUpdates()
   }
}
```

## UIDevice : NSObject

- For getting information about the device such as assigned name, device model, and operating-system name and version.
- For detecting changes in the device’s characteristics, such as physical orientation.
- For obtaining information and notifications about changes to the battery’s charge state and charge level (batteryLevel property).
- For providing access to the proximity sensor state (proximityState property). The proximity sensor detects whether the user is holding the device close to their face.
- Note: Enable battery monitoring or proximity sensing only when you need it.

## Some Examples

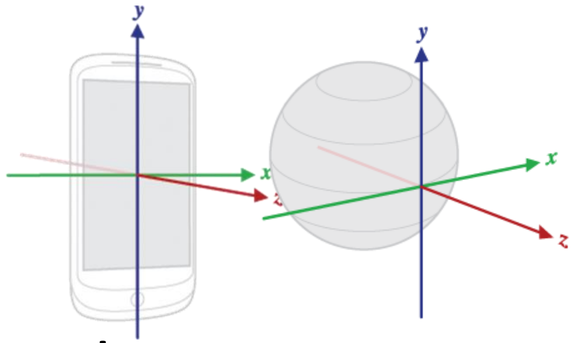
- Tracking the Device Orientation:
  - var orientation: UIDeviceOrientation
    - The physical orientation of the device.
  - enum UIDeviceOrientation
    - Constants that describe the physical orientation of the device.
- Determining the Current Orientation
  - var isPortrait: Bool
    - A Boolean value that indicates whether the device is in a portrait orientation.
  - var isLandscape: Bool
    - A Boolean value that indicates whether the device is in a landscape orientation.
- Using the Proximity Sensor
  - var isProximityMonitoringEnabled: Bool
    - A Boolean value that indicates whether proximity monitoring is enabled.
  - var proximityState: Bool
    - A Boolean value that indicates whether the proximity sensor is close to the user.

## Reference

- You may check the following sites for more information:
  - https://developer.apple.com/documentation/coremotion
  - https://developer.apple.com/documentation/uikit/uidevice
  - https://developer.apple.com/documentation/sensorkit

## Location Service

## Location



- Sensors
  - Cannot tell the global location of the device
    - O Local linear / angular acceleration / velocity relative to device frame
    - O Global orientation relative to world frame
    - X Global location relative to world frame
- Need “extra help” to determine global position
  - Something outside the mobile device

## Network and GPS Locations

- Network location
  - Cell tower
  - Wi-fi

  - X Less spatial accuracy
  - O Less power consumption
  - O More frequent / faster update
  - O Works both in / outdoor

- GPS location
  - Satellites

  - O More spatial accuracy
  - X More power consumption
  - X Less frequent / slower update
  - X Works only outdoor

## Usages

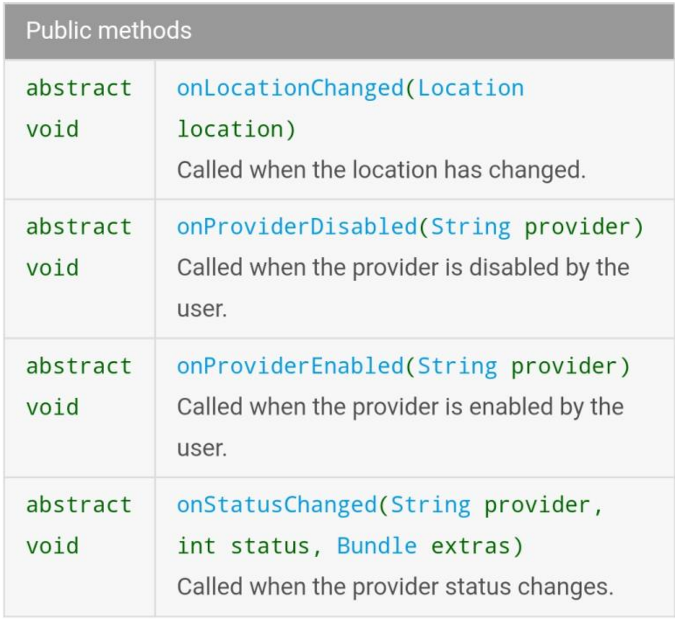
- Location is a very useful feature for mobile devices
  - In contrast to "stationary" devices
- Periodic location updates
  - Tracking user's whereabouts
  - Tagging user content
    - Photos, tweets, …
  - Providing services relevant to location
    - Directions & routes
    - Maps
    - Recommendation for restaurants/services
- Proximity alert
  - Big sale in a shop nearby

## Android Location Framework

- LocationManager
  - Manager for all location services
    - Event listener for periodic updates
    - Intent for proximity events
- LocationListener
  - Event listener
- Location
  - The location data structure returned by listener

# Basic Example

```
// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

// Define a listener that responds to location updates
LocationListener locationListener = new LocationListener() {
        public void onLocationChanged(Location location) {
                // Called when a new location is found
                // by the network location provider.
                makeUseOfNewLocation(location);
        }

        public void onStatusChanged(String provider, int status, Bundle extras) {}

        public void onProviderEnabled(String provider) {}

        public void onProviderDisabled(String provider) {}
};

//Register the listener with the Location Manager to receive location updates
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, min_time, min_dist, locationListener);
```



| Public methods | |
|---|---|
| abstract void | onLocationChanged(Location location) Called when the location has changed. |
| abstract void | onProviderDisabled(String provider) Called when the provider is disabled by the user. |
| abstract void | onProviderEnabled(String provider) Called when the provider is enabled by the user. |
| abstract void | onStatusChanged(String provider, int status, Bundle extras) Called when the provider status changes. |

## Permission

- Android manifest file
- GPS
  - `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />`
- Network (cell-id, wi-fi)
  - `<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />`
- Emulator
  - `<uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />`
- FINE covers COARSE

# iOS Core Location Framework

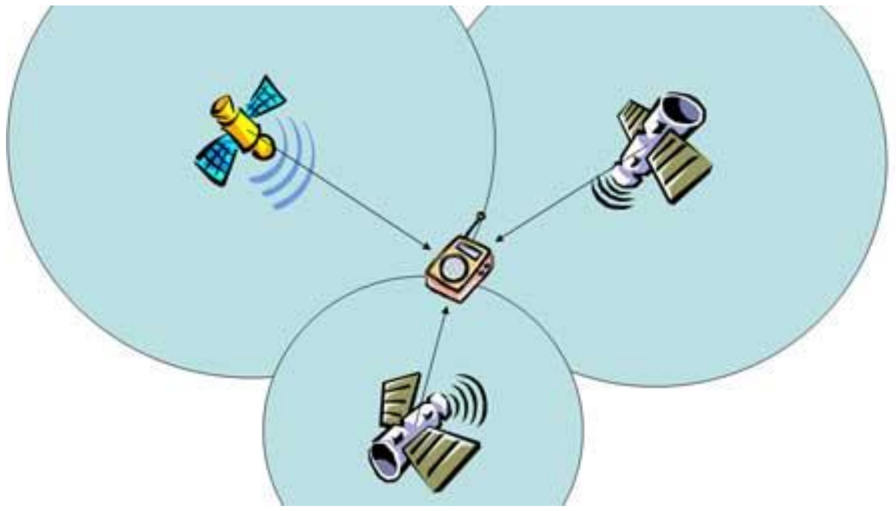
- Provides services that determine a device’s geographic location, altitude, and orientation, or its position relative to a nearby iBeacon device. The framework gathers data using all available components on the device, including the Wi-Fi, GPS, Bluetooth, magnetometer, barometer, and cellular hardware.
- You use instances of the CLLocationManager class to configure, start, and stop the Core Location services. A location manager object supports the following location-related activities:
  - Standard and significant location updates. Track large or small changes in the user’s current location with a configurable degree of accuracy.
  - Region monitoring. Monitor distinct regions of interest and generate location events when the user enters or leaves those regions.
  - Beacon ranging. Detect and locate nearby beacons.
  - Compass headings. Report heading changes from the onboard compass.
- To use location services, your app requests authorization and the system prompts the user to grant or deny the request.
- Details:
  - https://developer.apple.com/documentation/corelocation/
  - https://developer.apple.com/documentation/corelocation/ranging_for_beacons

## Challenges

- Multiple location sources
  - Different cost / benefit tradeoffs for GPS, Cell ID, and Wi-Fi
  - Which one(s) to use, and how to combine them?
- User movements
  - Totally unpredictable
- Measurement inaccuracies
  - Spatial / temporal noise / bias

## How to determine the location?

- Distance to each "anchor" is computed
  - Anchor: can be cell tower or satellite
- Voting from multiple anchors
  - Triangulation

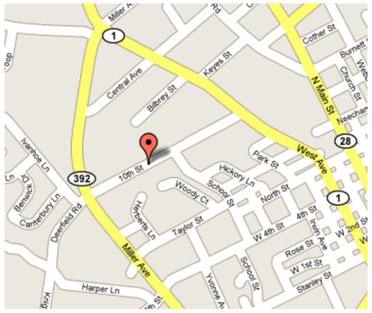


# Proximity Alert

- Register alerts with the system for proximity events
  - Triggers predetermined actions when the device enters / exits the vicinity of specific locations
  - Alleviates the need for constant location monitoring in your app
- Applications
  - Games (e.g., proximity bombs / traps)
  - Location based services (e.g., friends check-in to nearby locations)
- Example in Android:
  - void addProximityAlert (double latitude, double longitude, float radius, long expiration, PendingIntent intent)

## Google Map

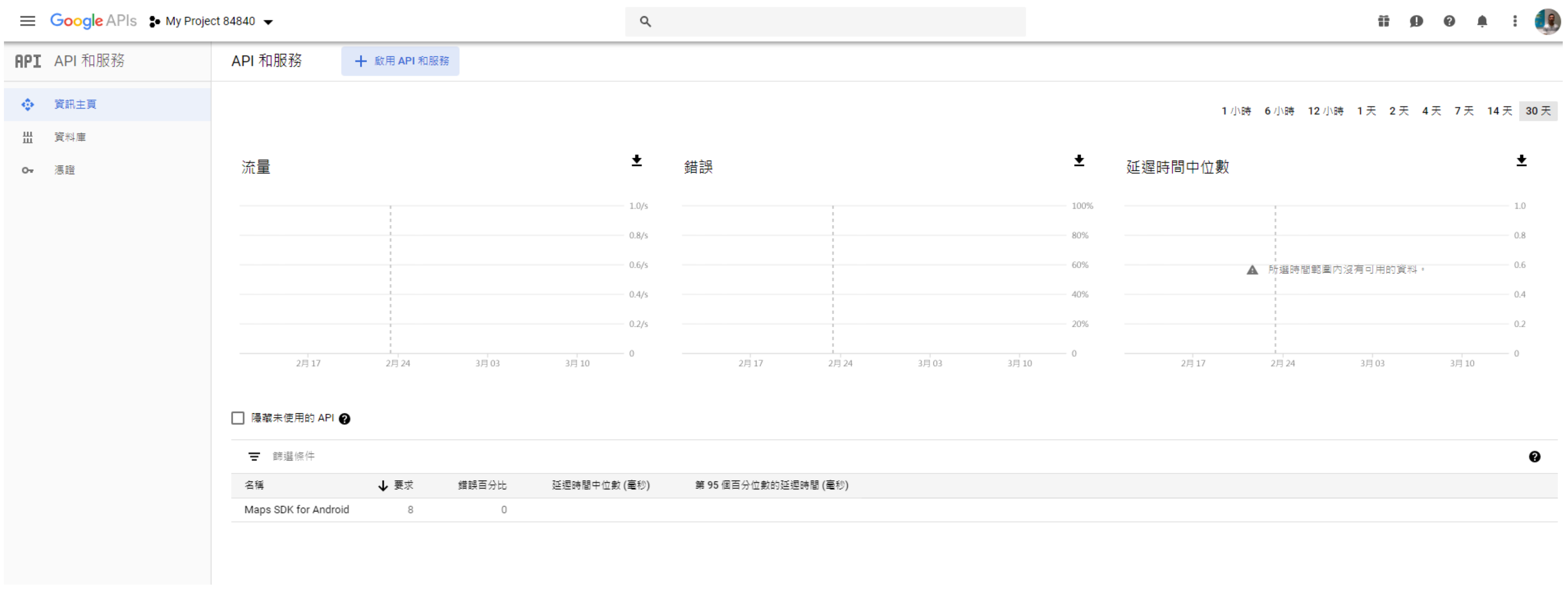
## Google Map



- Useful for applications involving maps
  - Directions
  - Traffic
  - Location-based services and games
- Usage
  - map = f(location)

# Application for Google Map APIs

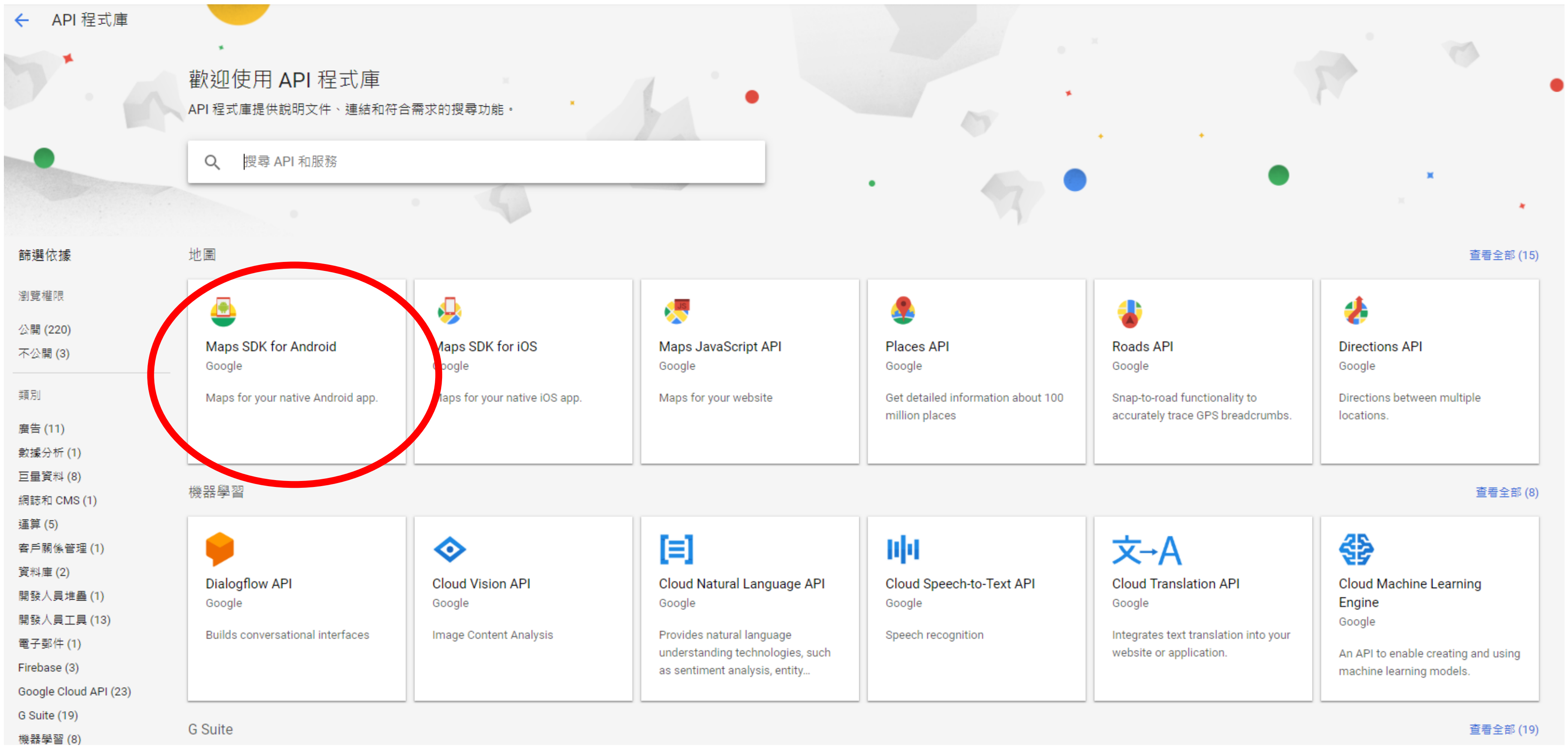
- Use any browser to open the following page: [https://console.developers.google.com](https://console.developers.google.com)
- Login using your Google account
- You should see the following page. Click “Database”.



| 名稱 | ↓ 要求 | 錯誤百分比 | 延遲時間中位數 (毫秒) | 第 95 個百分位數的延遲時間 (毫秒) |
|---|---|---|---|---|
| Maps SDK for Android | 8 | 0 | | |

# Application for Google Map APIs

- Choose “Maps SDK for Android” or “Maps SDK for iOS”



API 程式庫

歡迎使用 API 程式庫

API 程式庫提供說明文件、連結和符合需求的搜尋功能。

搜尋 API 和服務

篩選依據

瀏覽權限

公開 (220)
不公開 (3)

類別

廣告 (11)
數據分析 (1)
巨量資料 (8)
網誌和 CMS (1)
運算 (5)
客戶關係管理 (1)
資料庫 (2)
開發人員堆疊 (1)
開發人員工具 (13)
電子郵件 (1)
Firebase (3)
Google Cloud API (23)
G Suite (19)
機器學習 (8)

地圖 — 查看全部 (15)

Maps SDK for Android
Google
Maps for your native Android app.

Maps SDK for iOS
Google
Maps for your native iOS app.

Maps JavaScript API
Google
Maps for your website

Places API
Google
Get detailed information about 100 million places

Roads API
Google
Snap-to-road functionality to accurately trace GPS breadcrumbs.

Directions API
Google
Directions between multiple locations.

機器學習 — 查看全部 (8)

Dialogflow API
Google
Builds conversational interfaces

Cloud Vision API
Google
Image Content Analysis

Cloud Natural Language API
Google
Provides natural language understanding technologies, such as sentiment analysis, entity...

Cloud Speech-to-Text API
Google
Speech recognition

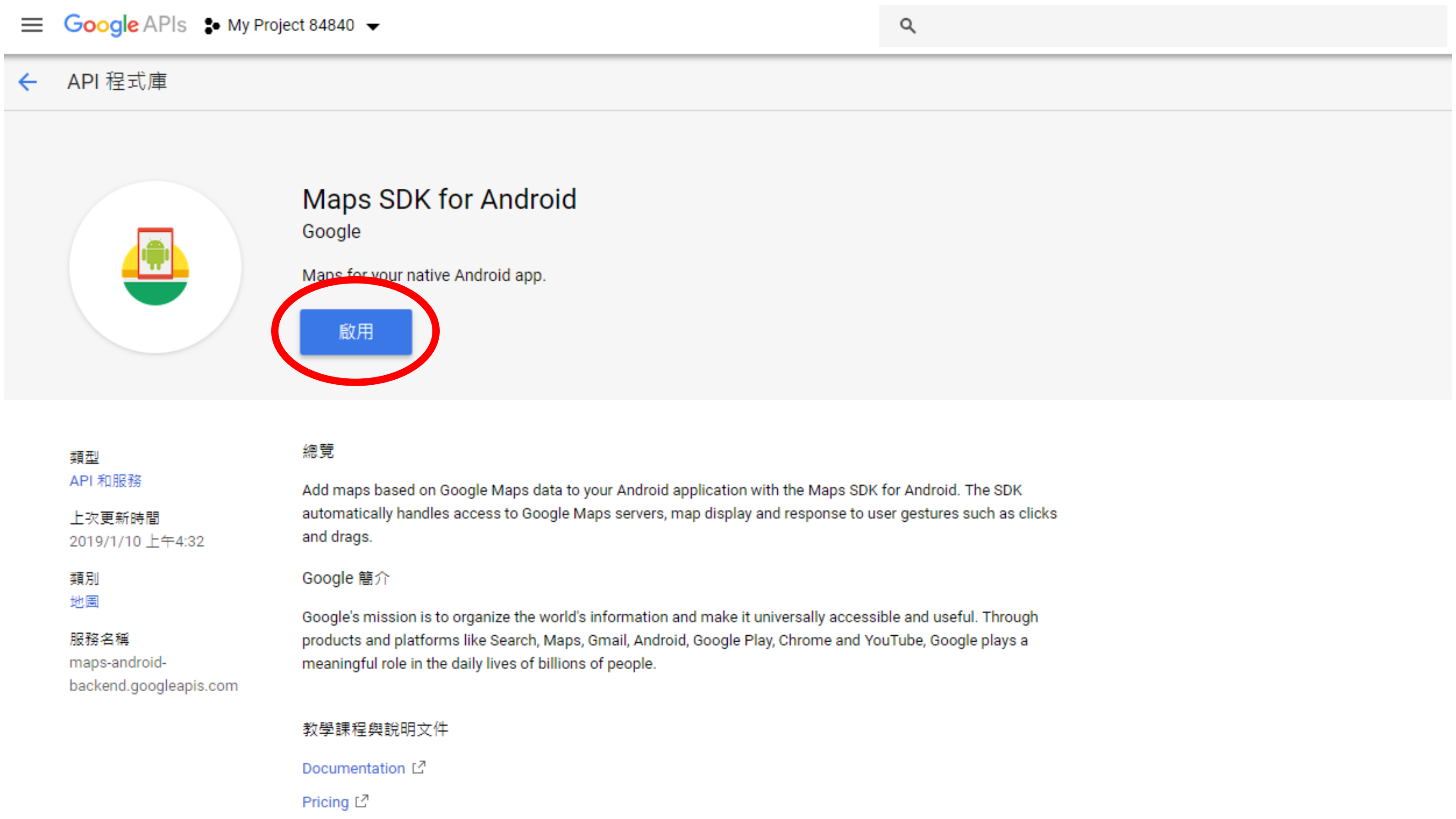
Cloud Translation API
Google
Integrates text translation into your website or application.

Cloud Machine Learning Engine
Google
An API to enable creating and using machine learning models.

G Suite — 查看全部 (19)

## Application for Google Map APIs

- Click “Activate” to activate your API.



Maps SDK for Android
Google
Maps for your native Android app.

類型
API 和服務

上次更新時間
2019/1/10 上午4:32

類別
地圖

服務名稱
maps-android-backend.googleapis.com

總覽

Add maps based on Google Maps data to your Android application with the Maps SDK for Android. The SDK automatically handles access to Google Maps servers, map display and response to user gestures such as clicks and drags.

Google 簡介

Google's mission is to organize the world's information and make it universally accessible and useful. Through products and platforms like Search, Maps, Gmail, Android, Google Play, Chrome and YouTube, Google plays a meaningful role in the daily lives of billions of people.

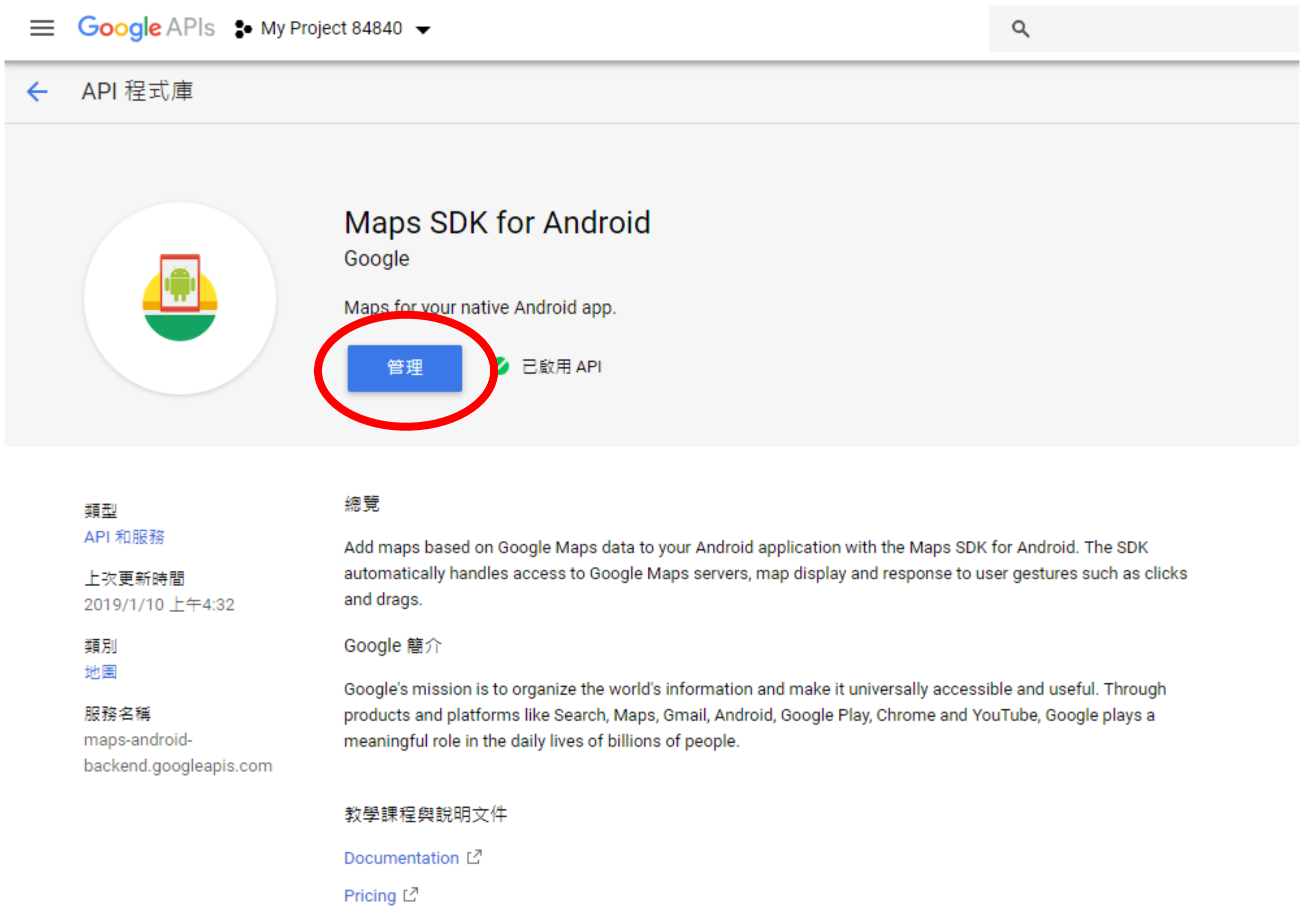
教學課程與說明文件

Documentation

Pricing

## Application for Google Map APIs

- Click “Manage”.



Maps SDK for Android
Google
Maps for your native Android app.

類型
API 和服務
上次更新時間
2019/1/10 上午4:32
類別
地圖
服務名稱
maps-android-backend.googleapis.com

總覽

Add maps based on Google Maps data to your Android application with the Maps SDK for Android. The SDK automatically handles access to Google Maps servers, map display and response to user gestures such as clicks and drags.

Google 簡介

Google's mission is to organize the world's information and make it universally accessible and useful. Through products and platforms like Search, Maps, Gmail, Android, Google Play, Chrome and YouTube, Google plays a meaningful role in the daily lives of billions of people.

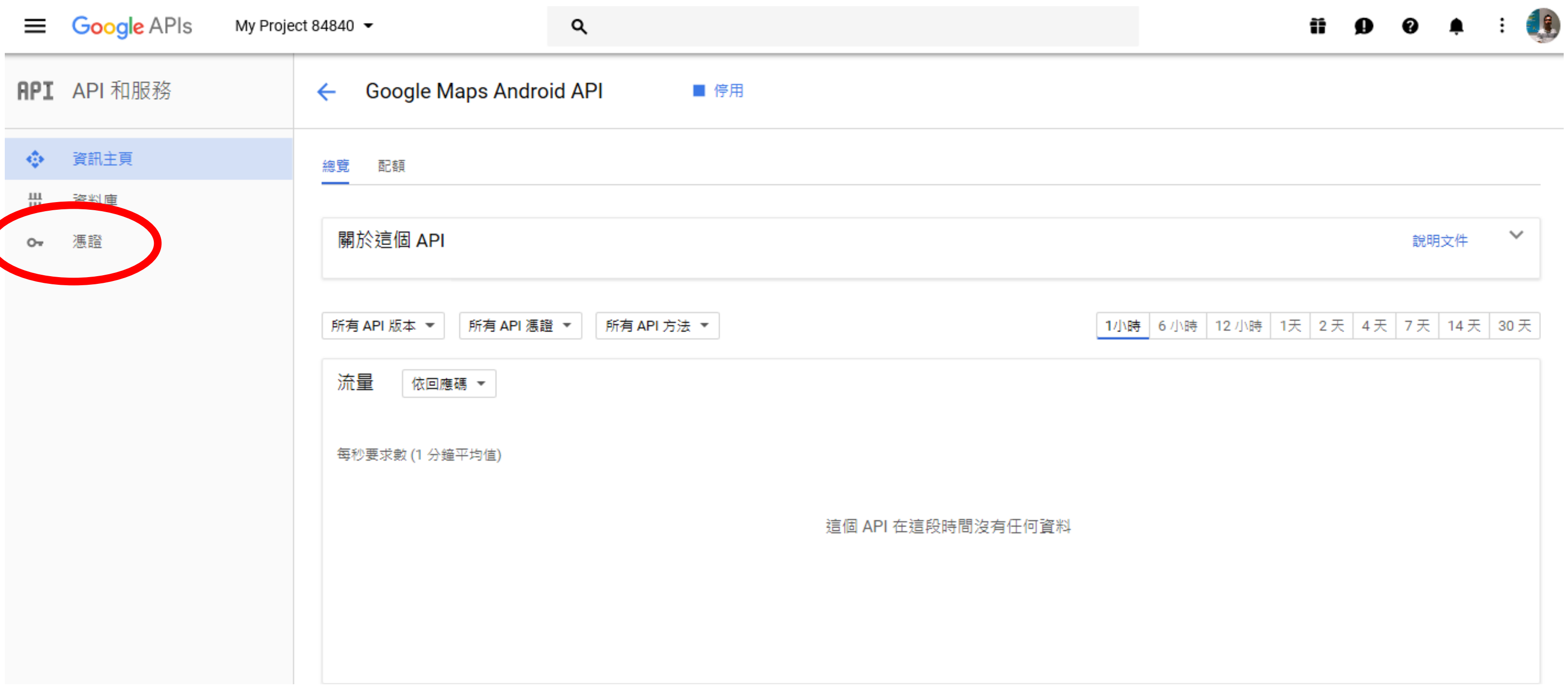
教學課程與說明文件

Documentation

Pricing

# Application for Google Map APIs

- Click the “Verify” option



Google APIs My Project 84840

API 和服務

資訊主頁

資料庫

憑證

Google Maps Android API 停用

總覽 配額

關於這個 API 說明文件

所有 API 版本 | 所有 API 憑證 | 所有 API 方法

1小時 | 6 小時 | 12 小時 | 1天 | 2 天 | 4 天 | 7 天 | 14 天 | 30 天

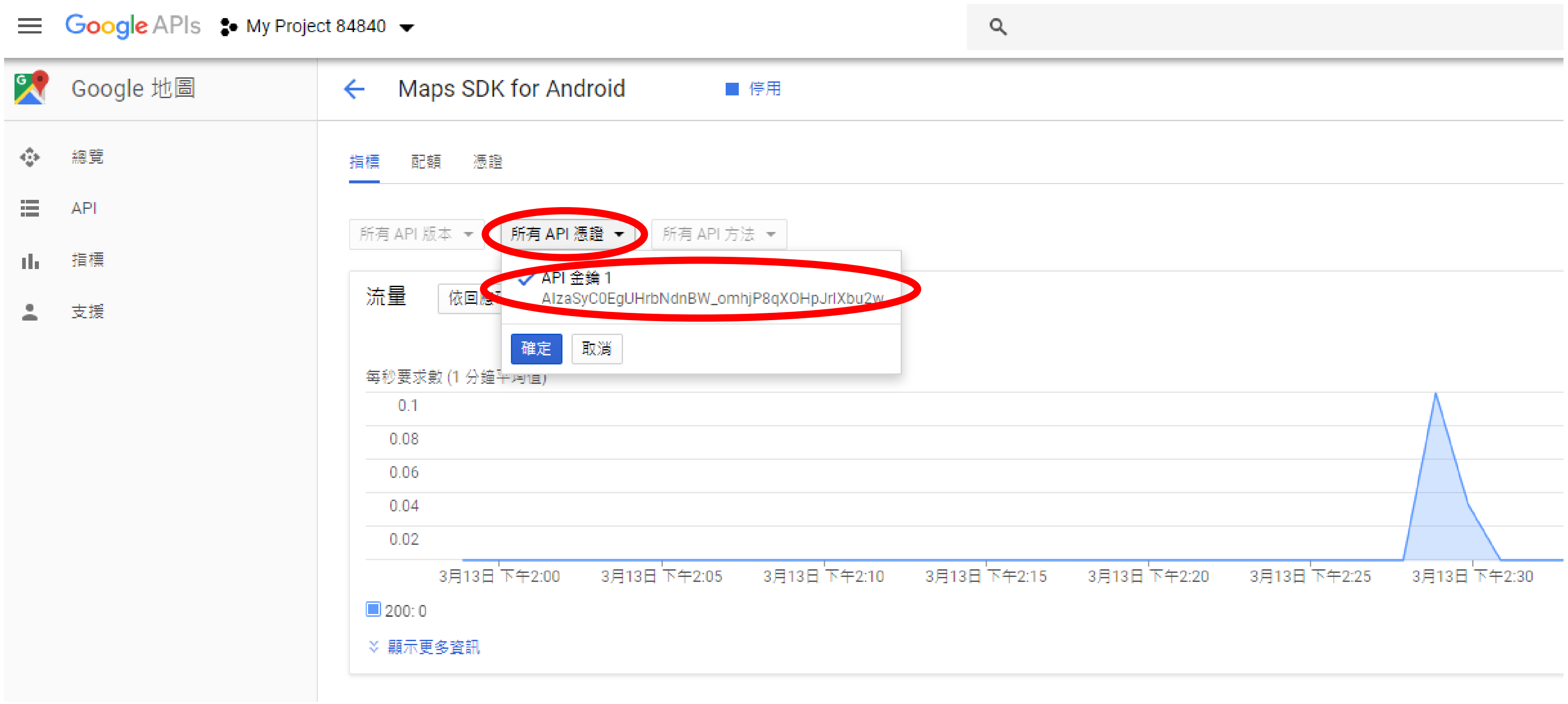
流量 依回應碼

每秒要求數 (1 分鐘平均值)

這個 API 在這段時間沒有任何資料

# Application for Google Map APIs

- You should see a key for your API application. You need this key in your Android Studio project.



API 金鑰 1

AIzaSyC0EgUHrbNdnBW_omhjP8qXOHpJrlXbu2w

# Code in Android Studio

- The following permission statements should be added to AndroidManifest.xml:

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

- The following should be added to the application:

```
<application
        …
        <meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyC0EgUHrbNdnBW_omhjP8qXOHpJrlXbu2w" />
        <activity
        …
```

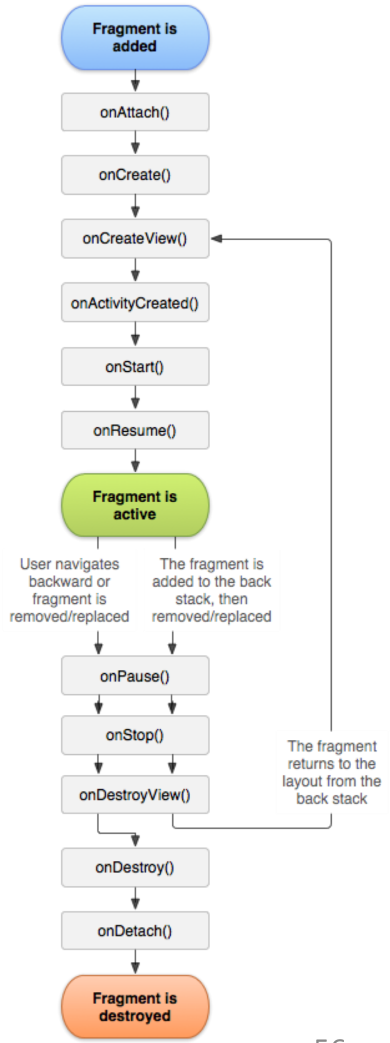
## Code in Android Studio

- To add a map to your activity, the following fragment should be added to the layout file:

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

## Supplementary Information on Android Fragment

- A Fragment represents a behavior or a portion of user interface in a FragmentActivity.
- You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.
- A fragment is like a modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running (like a "sub-activity" that you can reuse in different activities).



## Supplementary Information on Android Fragment: Basic Steps

- Modify the layout to include one or more regions (e.g., sub-layouts) for the fragments.
- For each fragment,
  - create a layout.
  - create a class which extents Fragment and implement essential methods.
- In the main activity, at where you want to add the fragment, include the following code:

```
FragmentManager manager = getSupportFragmentManager();
FragmentTransaction ft = manager.beginTransaction();
ft.replace(<ID of region in main layout>, <class name of fragment>(), <a number
tag to represent this fragment>);
ft.commitAllowingStateLoss();
```

- At where you want to add a fragment to back stack, call the following:

```
ft.addToBackStack(<number tag representing the fragment>);
```

## Supplementary Information on Android Fragment: Example

- In the main layout file, reserve a region to show the fragment.

```
<FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/frame_container"
        android:layout_margin="15dp">
</FrameLayout>
```

- Suppose there are two fragments and their layout files are "fragment_fragment_one.xml" and "fragment_fragment_two.xml" respectively.

## Supplementary Information on Android Fragment: Example

- The program of the first fragment should look like.

```
public class FragmentOne extends Fragment {
        public FragmentOne() {
                // Required empty public constructor
        }
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
        }
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                Bundle savedInstanceState) {
                // Inflate the layout for this fragment
                return inflater.inflate(R.layout.fragment_fragment_one, container, false);
        }
}
```

- The program of the second fragment should look similar.

## Supplementary Information on Android Fragment: Example

- Since you may need to load different fragments in your main program and for each fragment, you may need to load it multiple times, it’s better to include the fragment management logic into a method.

```
public void addFragment(Fragment fragment, boolean addToBackStack, String tag) {
        FragmentManager manager = getSupportFragmentManager();
        FragmentTransaction ft = manager.beginTransaction();
        if (addToBackStack) {
                ft.addToBackStack(tag);
        }
        ft.replace(R.id.frame_container, fragment, tag);
        ft.commitAllowingStateLoss();
}
```

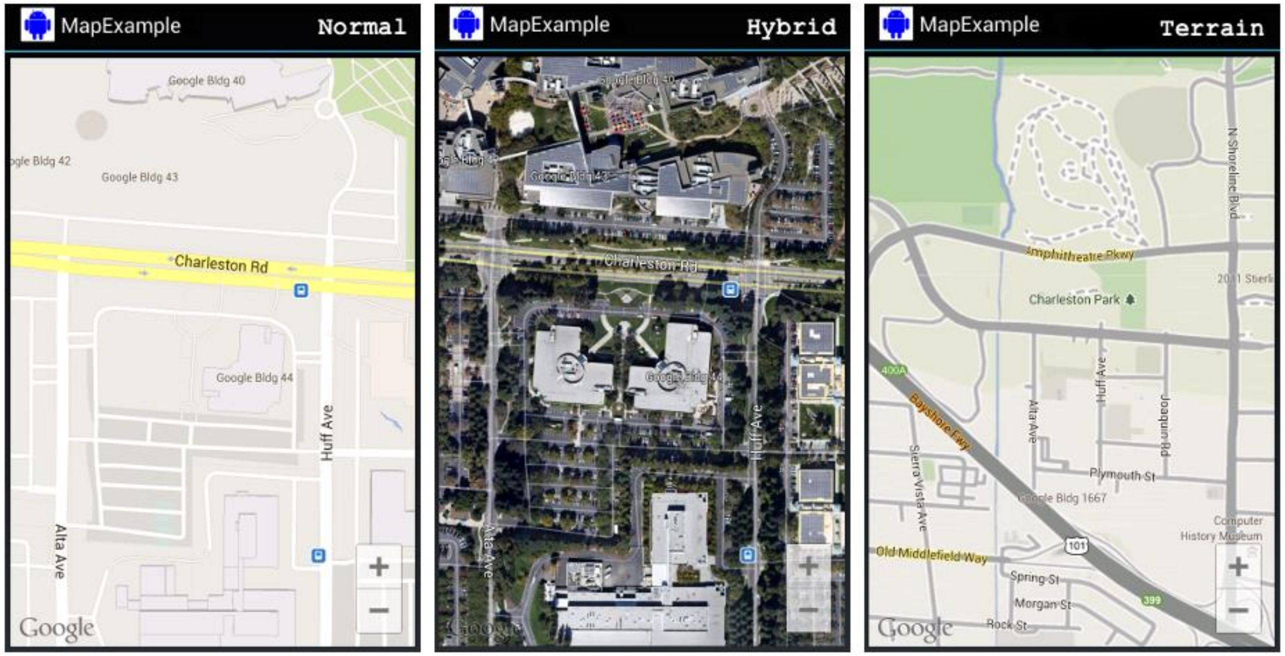
By calling addToBackStack(), the replace transaction is saved to the back stack so the user can reverse the transaction and bring back the previous fragment by pressing the Back button.

- To load the first fragment, the main program should contain the following logic.

```
addFragment(new FragmentOne(), false, "one");
// FragmentOne is the class name of the first fragment
```

## Types of Maps

- There are four types of maps available within the Google Maps API.
  - Roadmap / Normal: displays the default road map view. This is the default map type.
  - Satellite: displays Google Earth satellite images
  - Hybrid: displays a mixture of normal and satellite views
  - Terrain: displays a physical map based on terrain information.



## Code in Android Studio

- Some common settings can be done using the statements below. These statements should be self-explanatory!

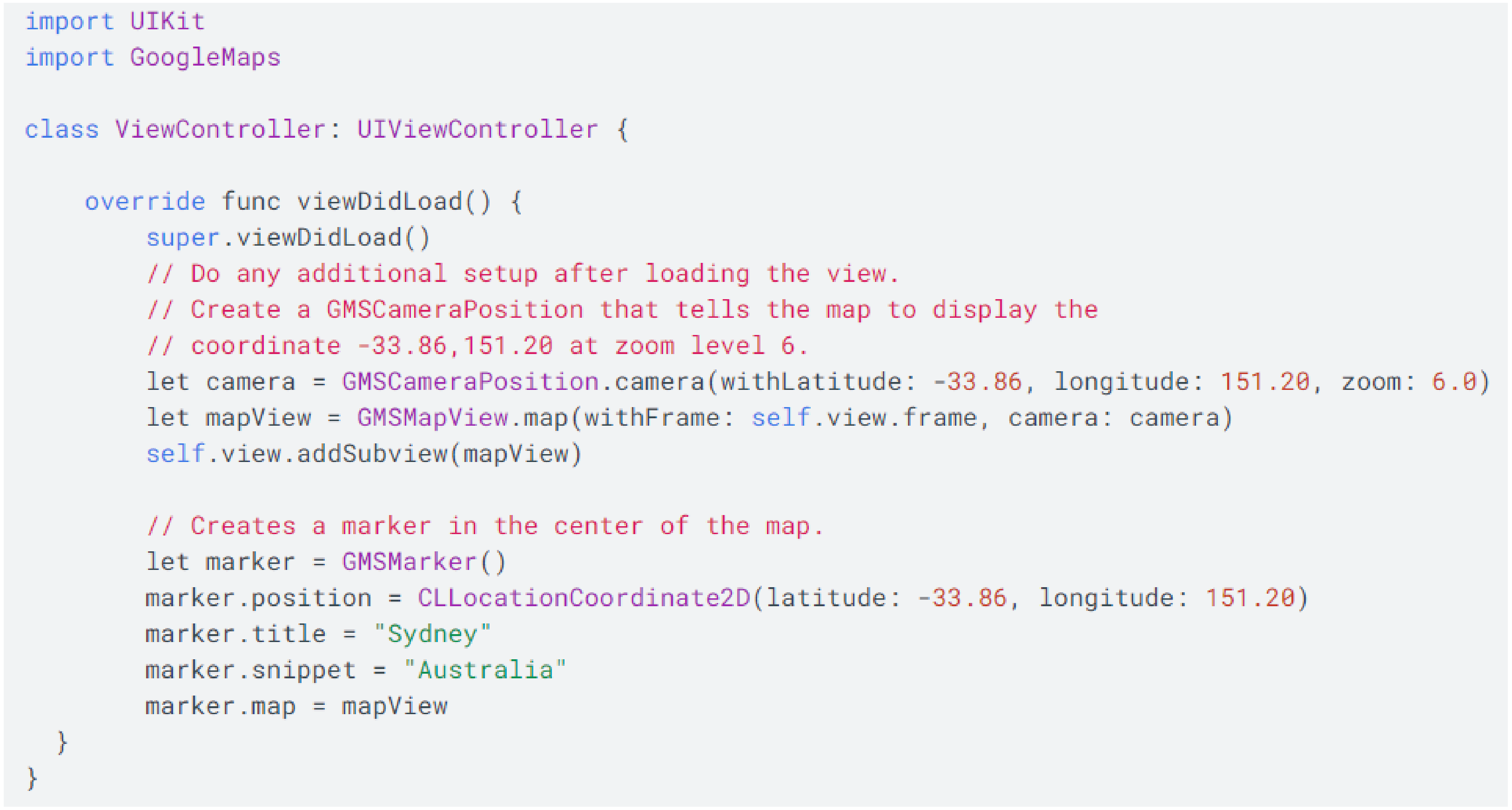
```
GoogleMap map;
map.setMapType(GoogleMap.MAP_TYPE_HYBRID);
map.setTrafficEnabled(true);
map.setMyLocationEnabled(true);
LatLng hku = new LatLng(22.2804, 114.131);
Marker marker_hku = map.addMarker(new MarkerOptions()
            .position(hku)
            .title(“The University of Hong Kong”)
            .snippet(“The best university in Hong Kong”)
            .icon(BitmapDescriptorFactory.fromRource(R.drawable.pin)));
```

## Sample Code

- Please refer to samples provided by official GitHub: https://developers.google.com/maps/documentation/android-sdk/code-samples.

# Using Maps SDK for iOS

- Please refer to the site below for details: https://developers.google.com/maps/documentation/ios-sdk/start



```
import UIKit
import GoogleMaps

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        // Create a GMSCameraPosition that tells the map to display the
        // coordinate -33.86,151.20 at zoom level 6.
        let camera = GMSCameraPosition.camera(withLatitude: -33.86, longitude: 151.20, zoom: 6.0)
        let mapView = GMSMapView.map(withFrame: self.view.frame, camera: camera)
        self.view.addSubview(mapView)

        // Creates a marker in the center of the map.
        let marker = GMSMarker()
        marker.position = CLLocationCoordinate2D(latitude: -33.86, longitude: 151.20)
        marker.title = "Sydney"
        marker.snippet = "Australia"
        marker.map = mapView
  }
}
```

## Collecting Data from Wearable Device

- Besides built-in sensors, mobile phone can also collect data from wearable device.
- Most of the smart watches nowadays can detect step-count and heart-beat rate.
- Some smart watches can also measure:
  - Blood pressure
    - Use an internal Photoplethysmogram sensor to measure both systolic and diastolic pressures. Calibration using traditional blood pressure monitor is required for accurate readings.
  - Blood oxygen level
    - Red LED is emitted onto skin surface and a sensor is used to detect the reflected red LED. Since Hemoglobin with and without oxygen have different rate of red-light absorption, blood oxygen level can be estimated from the reflected LED light.
  - Blood sugar / glucose level
    - Ions under our skin move together with glucose elements to form glucose oxidase. Blood glucose level can be estimated by measuring the degree of movement of glucose oxidase. Calibration using traditional blood glucose meter is required.
- Mobile app developers can think about how to utilize these health data!

## Final Remark

- In terms of programming, sensors, location service and maps are straight-forward and standard.
- But how to utilize them to form innovative applications is far more important. E.g.,
  - Can sensor data improve your app to make it more useful or interesting?
  - Can location service and map bring more convenience to your users?
- Try to think about how your app can behave better than other similar apps in the market with these new sensed  data!

# Chapter 11.

End

2023-2024
COMP7506 Smart Phone Apps Development
Dr. T.W. Chim (E-mail: twchim@cs.hku.hk)
Department of Computer Science, The University of Hong Kong