ANDROID LAB 2
SENSORS

OVERVIEW

1. Introduction to “Activity” class
2. Run Apps on your own Android phone
3. Introduction to “Listener”
4. GPS
5. Sensor: Accelerometer

NOTICE

- If you don’t have Android phone, you can borrow one from TA.

- when you see a CHECK POINT title, please call the TA to check your progress.
- Check Point 3, 4 can be demo in next course

INTRODUCTION TO ACTIVITY

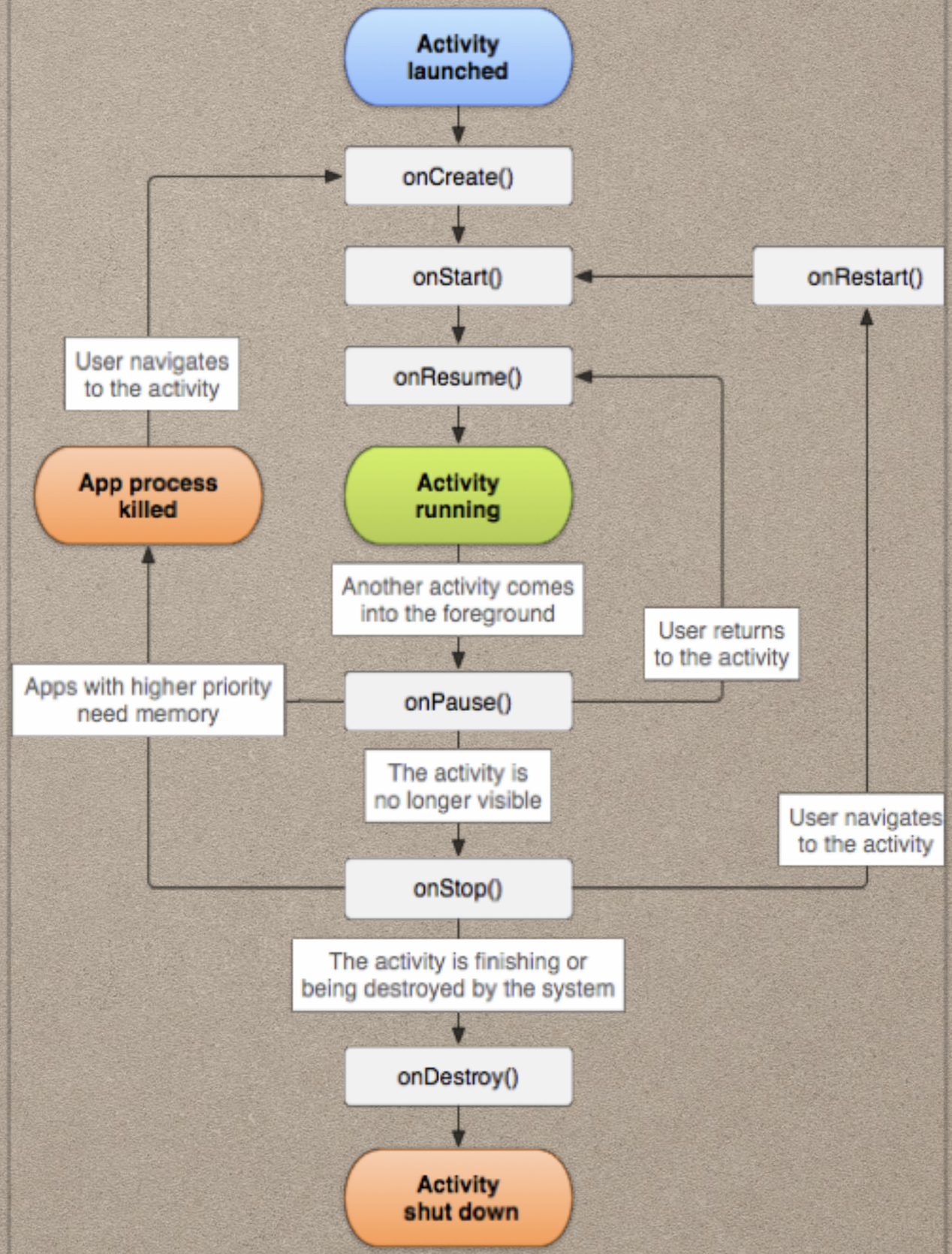
ACTIVITY CLASS

- Activity class creating a window for you in which you can place your UI with setContentView(View).

```
public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

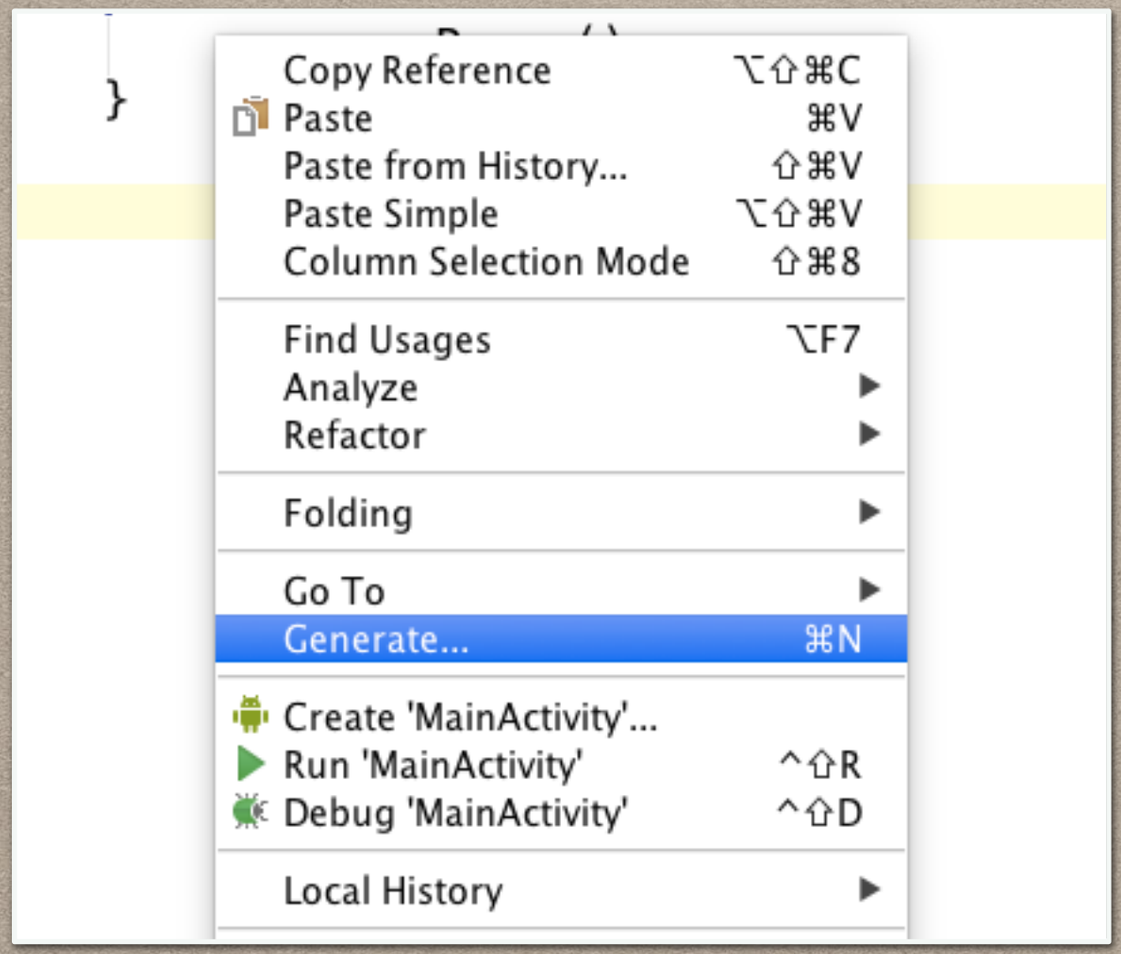
ACTIVITY LIFE CYCLE

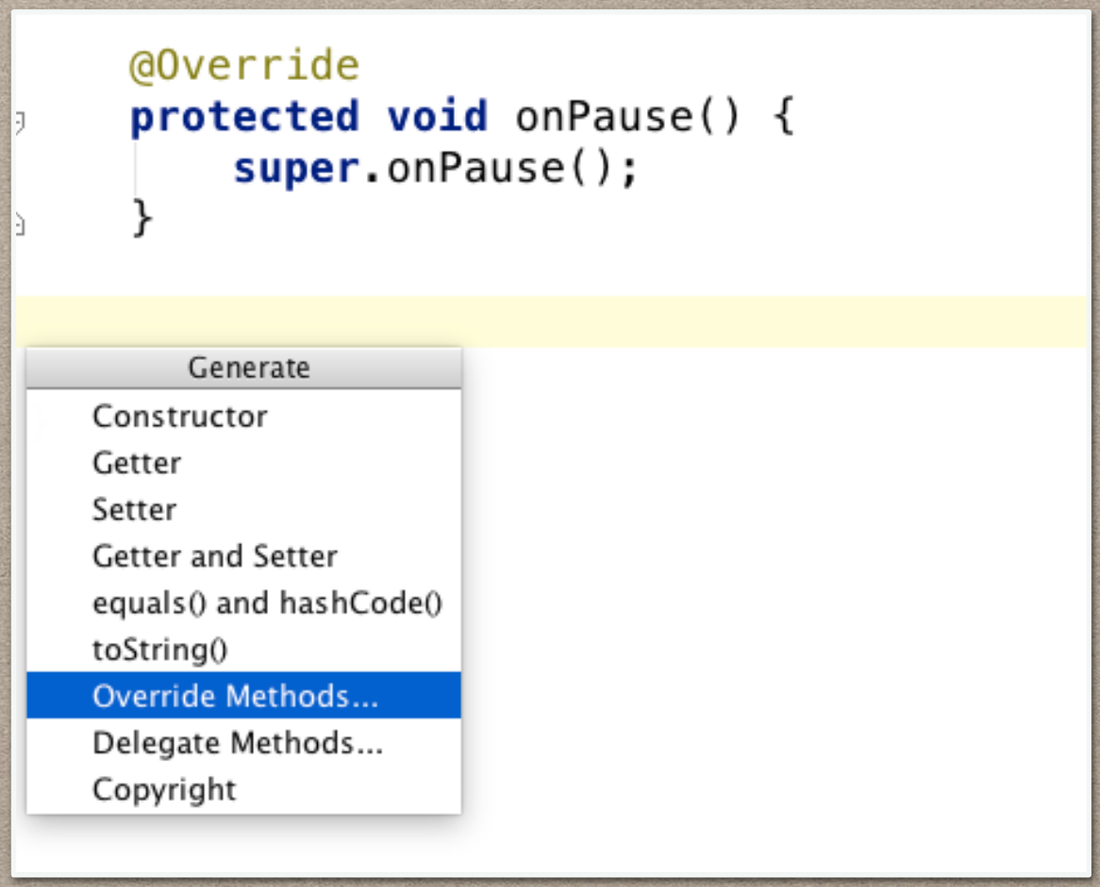
- when you launch your app on the phone, activity will go through
  onCreate()
  onStart()
  onResume()



- There are two methods almost all subclasses of Activity will implement:
  - onCreate(Bundle) is where you initialize your activity. Most importantly, here you will usually call setContentView(int) with a layout resource defining your UI, and using findViewById(int) to retrieve the widgets in that UI that you need to interact with programmatically.
  - onPause() is where you deal with the user leaving your activity. Most importantly, any changes made by the user should at this point be committed (usually to the ContentProvider holding the data).

- You can right click in activity class and add a method.

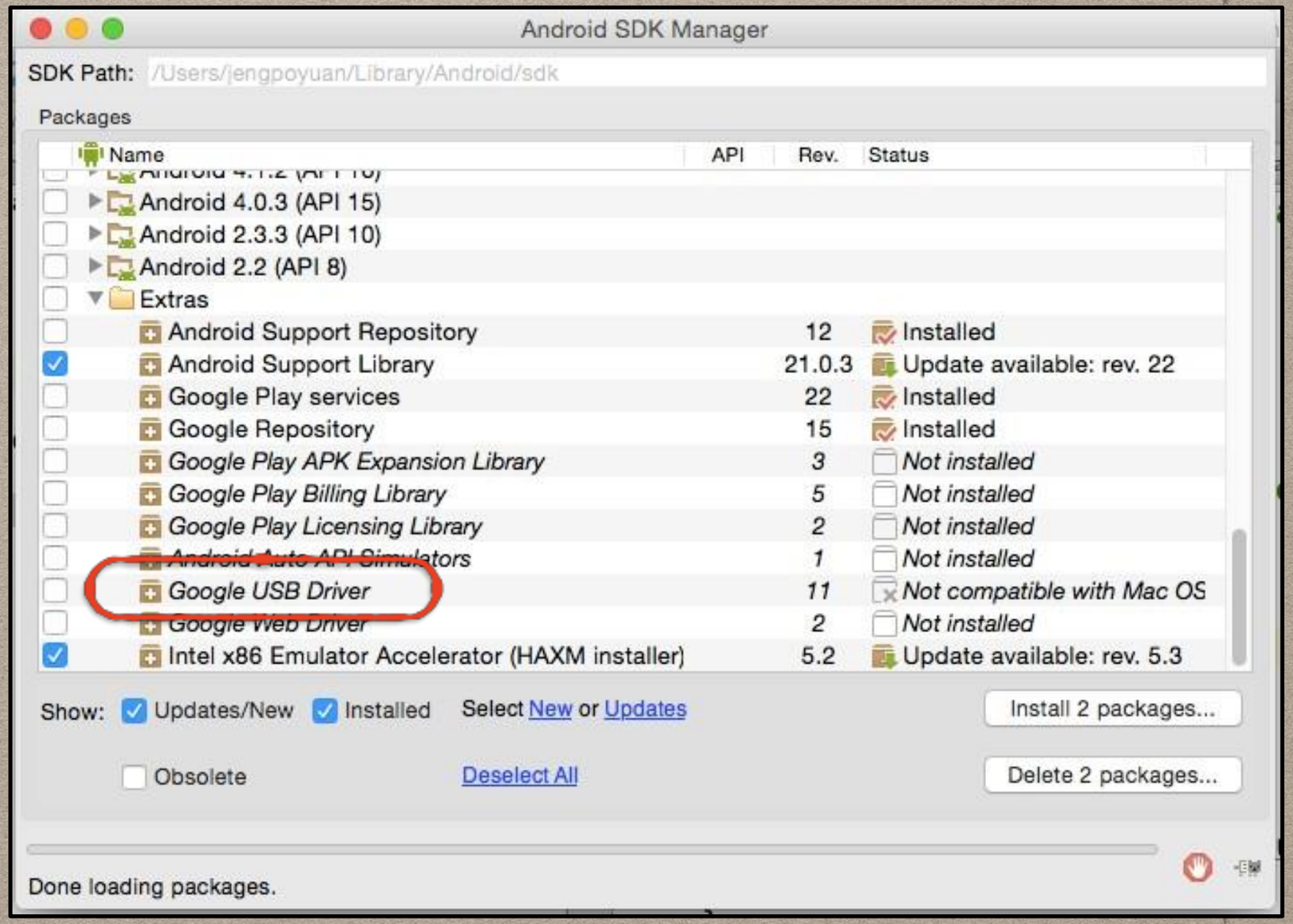




RUN YOUR APP ON YOUR PHONE

INSTALL GOOGLE USB DRIVER

- Install google usb driver through SDK manager

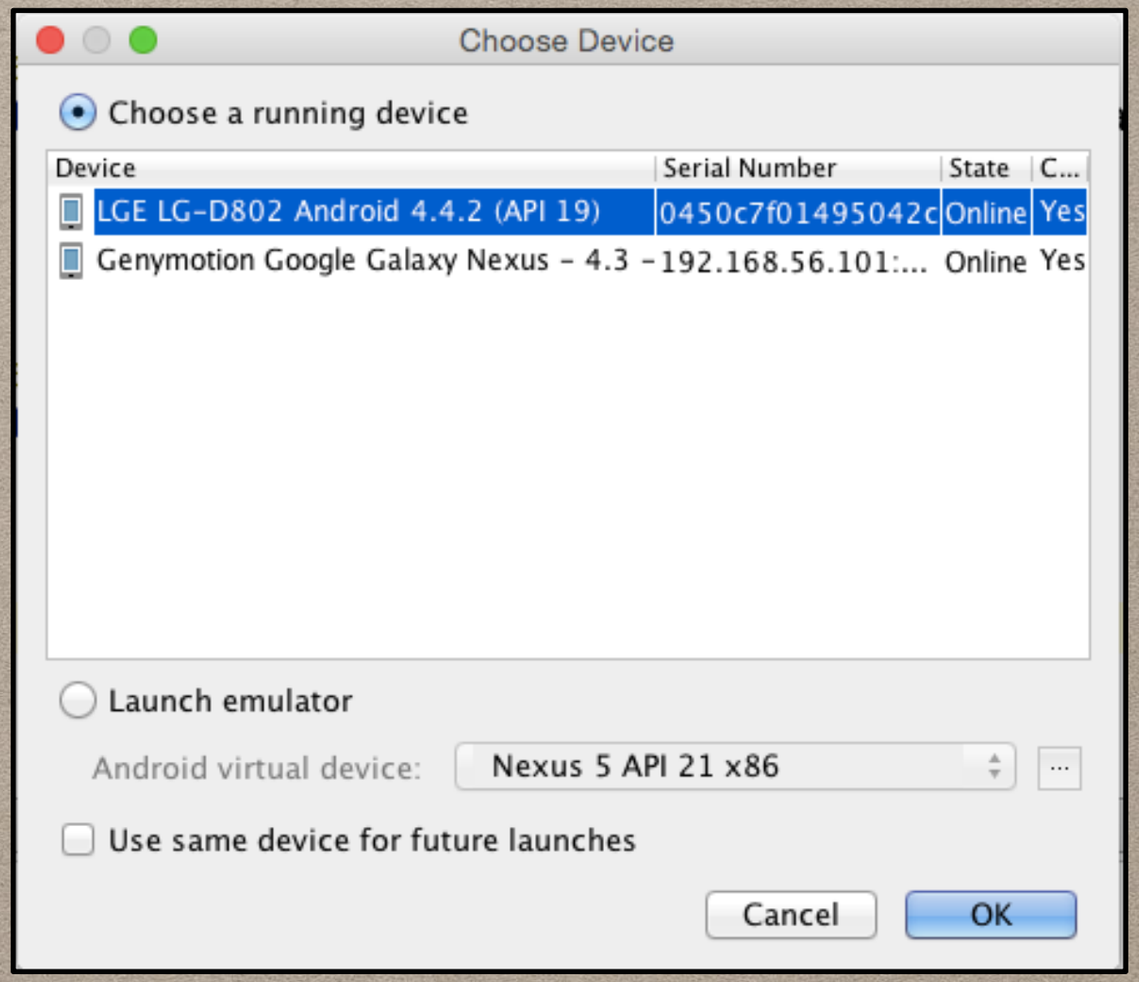


ENABLE USB DEBUGGING ON YOUR DEVICE

- On most devices running Android 3.2 or older, you can find the option under Settings > Applications > Development.
- On Android 4.0 and newer, it's in Settings > Developer options. Note: On Android 4.2 and newer, Developer options is hidden by default. To make it available, go to Settings > About phone >Software information and tap Build number seven times. Return to the previous screen to find Developer options.

RUN YOUR APP ON THE PHONE

- connect your phone to the computer
- press run.
- then you can choose your debug device.



INTRODUCTION TO "LISTENER"

WHATS "LISTENER" FOR?

- You can set an onClicklistener() for a button to monitor whether the user press the button.
- You can set a LocationListener to monitor the GPS information.

EXAMPLE: BUTTON CLICK LISTENER

```
Button btn = (Button) findViewById(R.id.btn) ;

btn.setOnClickListener(new Button.OnClickListener() {
    @Override
    public void onClick(View v) {
        // do whatever you want when button is pressed
    }
}) ;
```

auto fix: alt + Enter

- other event listener:
  - View.OnClickListener：執行點擊事件。
  - View.OnLongClickListener：執行長按事件。
  - View.OnKeyListener：執行實體按鍵操作事件。
  - View.OnTouchListener：執行觸控螢幕操作事件。

- There are other listeners for GPS, Sensors …
  we will use them later in this Lab.

GPS

ADD GPS PERMISSION

- First, we need to add some permission to allow the program access GPS resource.

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

- internet is for accessing internet
  access_fine_location is for GPS
  access_coarse_location is for locating by internet

IMPLEMENT LOCATIONLISTENER

- To monitor GPS location, implements LocationListener

```
public class MainActivity extends AppCompatActivity implements LocationListener {
```

- We can auto generate some methods

```
@Override
public  void onLocationChanged(Location location){    //當地點改變時

}

@Override
public  void onStatusChanged(String provider, int status, Bundle extras){   //當定位狀態改變時

}

@Override
public  void onProviderEnabled(String provider){        //GPS或網路定位功能開啟

}

@Override
public void onProviderDisabled(String provider){        //GPS或網路定位功能關閉

}
```

IMPLEMENT LOCATIONLISTENER

- LocationManager class provides access to the system location services. These services allow applications to obtain periodic updates of the device's geographical location.

Hint:記得將Google map更新至最新版

```
LocationManager locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
LocationListener LL = this;
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, LL);
                                        // provider, update frequency, shortest distance, update listener
Location location = locationManager.getLastKnownLocation(locationManager.NETWORK_PROVIDER);
LL.onLocationChanged(location);
```

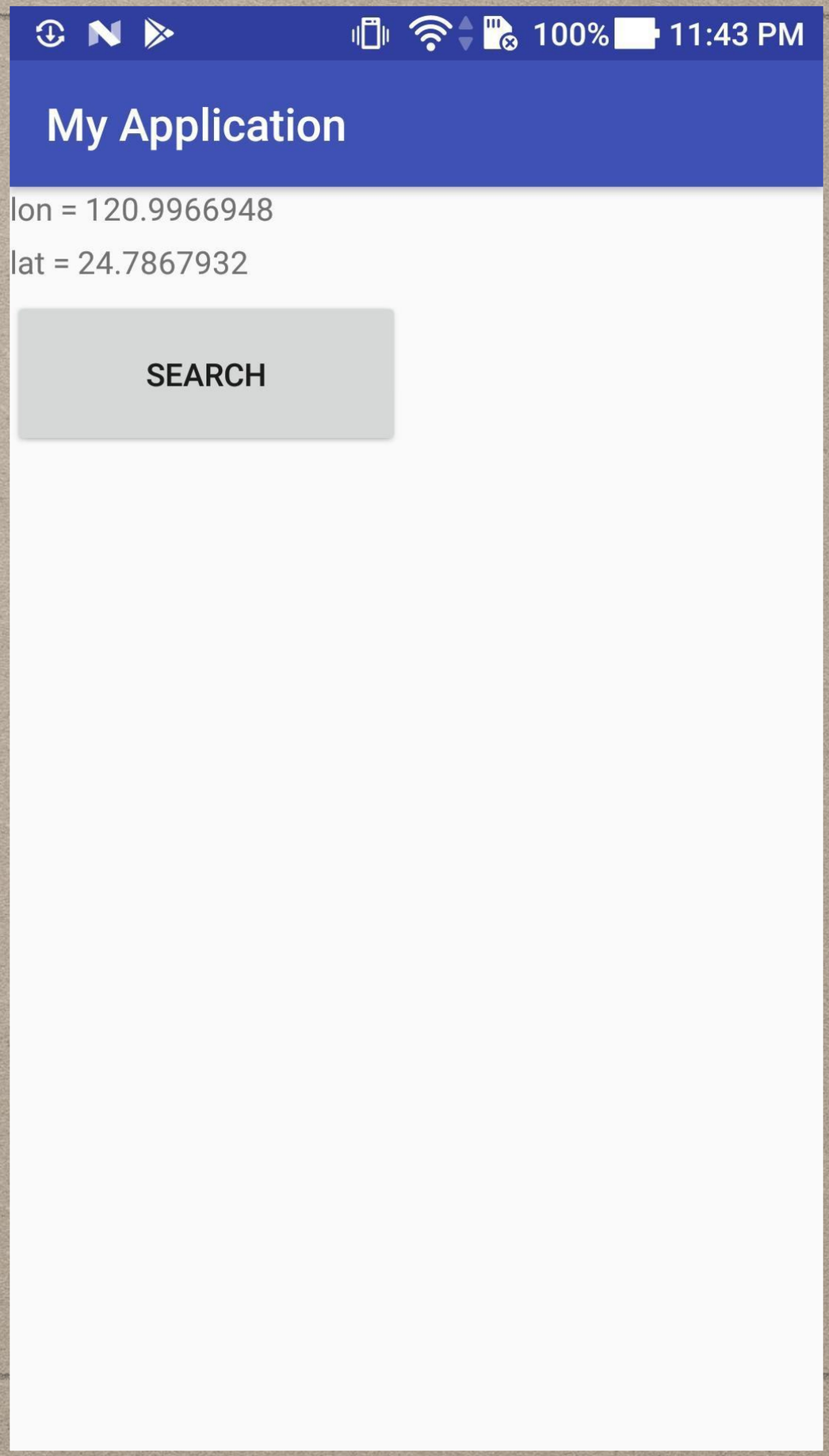
If (checkSelfPermission...)

```
ActivityCompat.requestPermissions(this,
        new String[]{Manifest.permission.ACCESS_FINE_LOCATION,
                Manifest.permission.ACCESS_COARSE_LOCATION},
        1);
```

CHECK POINT-1

- GPS + SEARCH

Hint:location.get…()



可能會遇到的問題

- 如果定位沒抓到，開啟APP時很可能會閃退，這時候可以先開啟手機內建的GoogleMAP先行定位自己的位子，等到大約30秒後再開啟自己的APP

另一個解決方法:

問題點:

getLastKnownLocation() frequently returns null, on all versions of Android

◆getLastKnownLocation()返回null的解決方法

http://fzlihui.iteye.com/blog/713050

SENSOR: ACCELEROMETER

SENSORS

- 加速度感測器(accelerometer)
- 陀螺儀感測器(gyroscope)
- 環境光照感測器(light)
- 磁力感測器(magnetic field)
- 方向感測器(orientation)
- 壓力感測器(pressure)
- 距離感測器(proximity)
- 溫度感測器(temperature)

SENSOR MANAGER

- SensorManager lets you access the device's sensors. Get an instance of this class by calling Context.getSystemService() with the argument SENSOR_SERVICE.

SENSOR EVENT LISTENER

- Used for receiving notifications from the SensorManager when sensor values have changed.

STEPS FOR USING ACCELEROMETER

1. SensorManager

```
SensorManager mSensorManager;
Sensor mAccelerometer;
mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

2. Implement SensorEventListener

```
public class MainActivity extends ActionBarActivity implements SensorEventListener {
```

```
@Override
public void onSensorChanged(SensorEvent event) {

}

@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {

}
```

auto fix: alt + Enter

3. Register listener

SensorManager.SENSOR_DELAY_FASTEST：最快，延遲最小。
SensorManager.SENSOR_DELAY_GAME：適合遊戲的頻率。
SensorManager.SENSOR_DELAY_NORMAL：正常頻率。
SensorManager.SENSOR_DELAY_UI：適合普通用戶界面UI變化的頻率。

```
mSensorManager.registerListener( [redacted] this, mAccelerometer, mSensorManager.SENSOR_DELAY_GAME);
```

4. Add three TextView

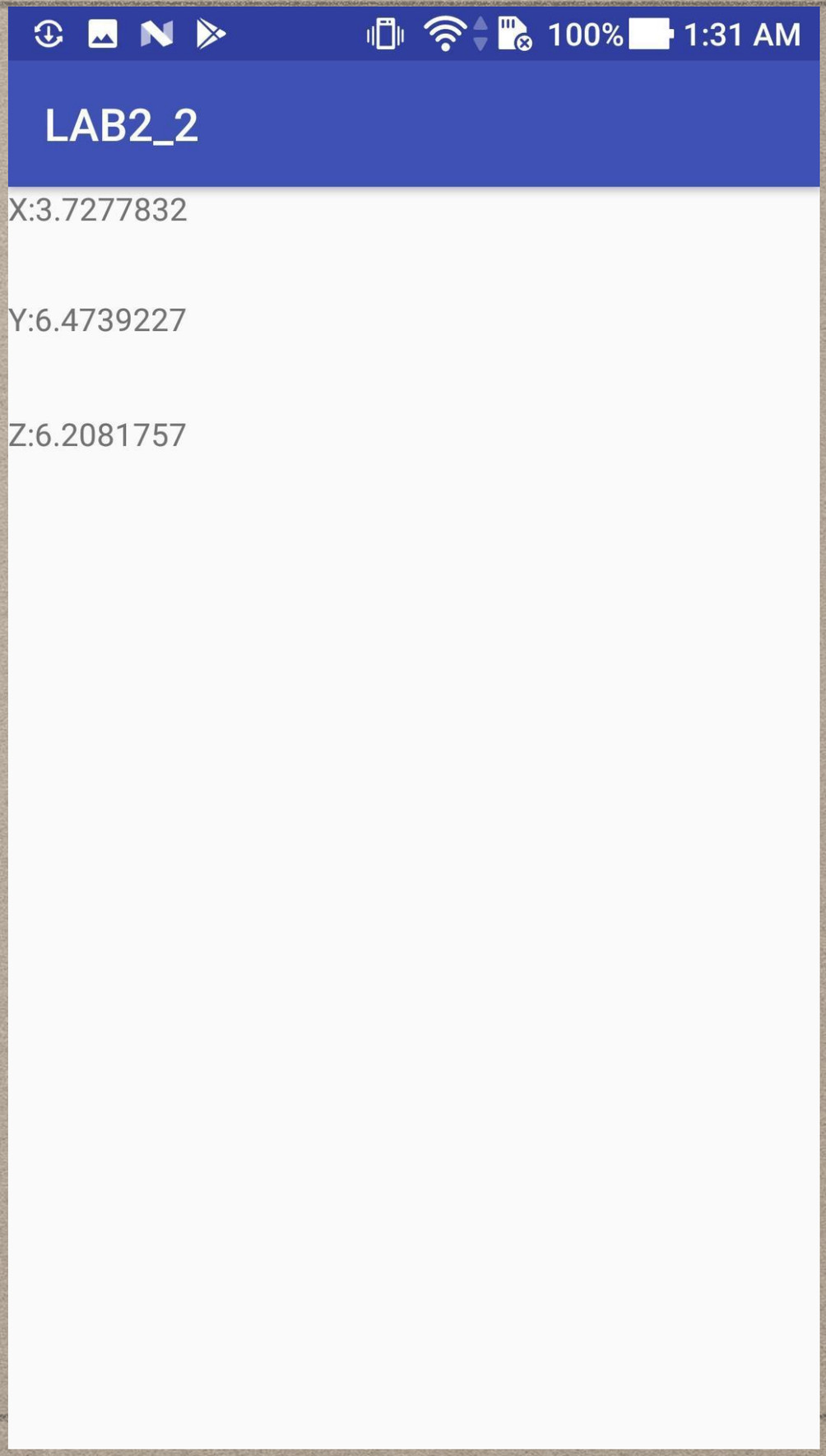
```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Large Text"
    android:id="@+id/x_txt"
    android:layout_alignParentTop="true"
    android:layout_alignParentStart="true" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Large Text"
    android:id="@+id/y_txt"
    android:layout_alignParentTop="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="64dp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Large Text"
    android:id="@+id/z_txt"
    android:layout_alignParentTop="true"
    android:layout_alignParentStart="true"
    android:layout_marginTop="75dp" />
```

5. Print the accelerometer data

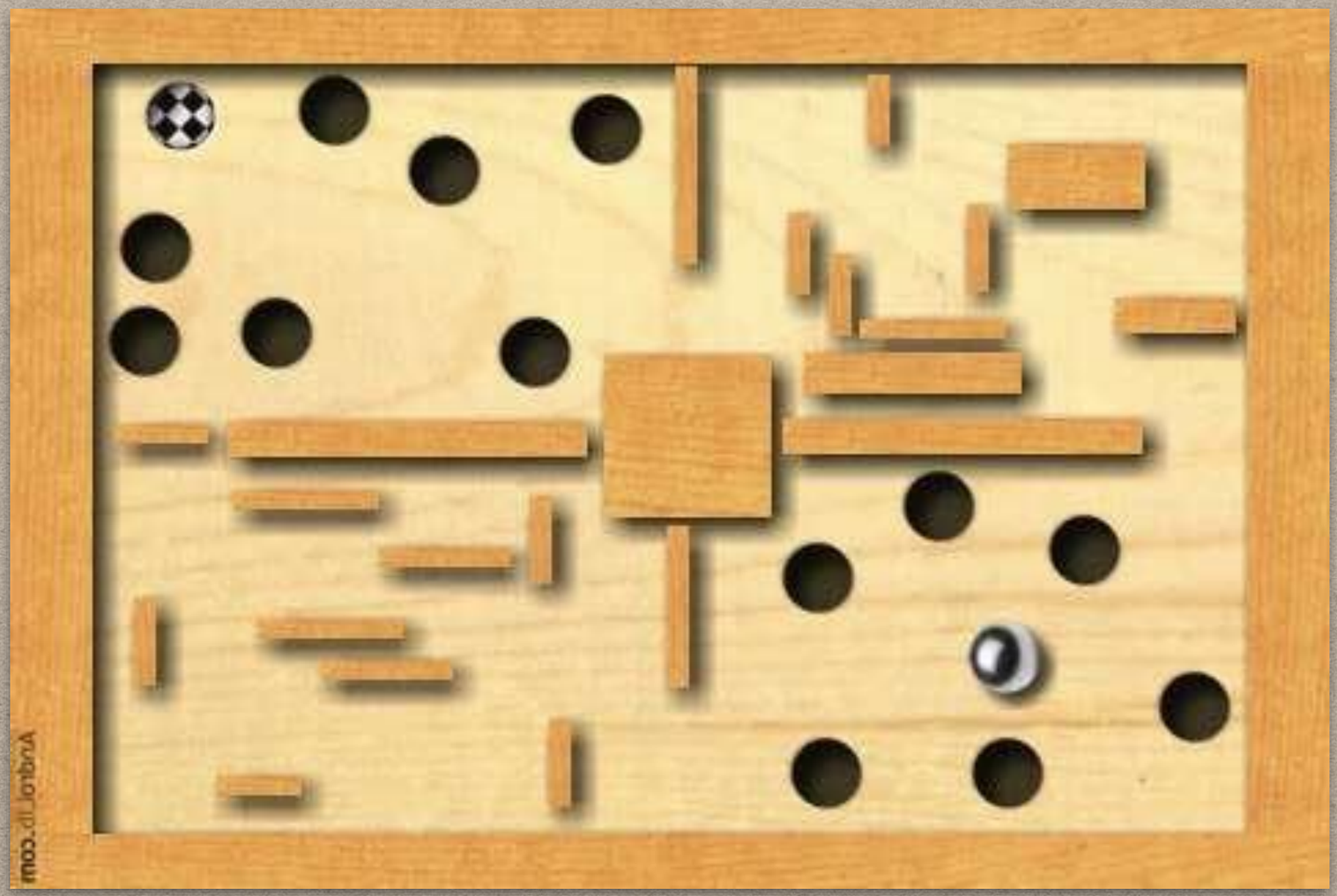
```
public void onSensorChanged(SensorEvent event) {
    x = event.values[0] ;
    y = event.values[1] ;
    z = event.values[2] ;

    tx.setText("X:" + x);
    ty.setText("Y:" + y);
    tz.setText("Z:" + z);

}
```

CHECK POINT-2

- print the data of accelerometer



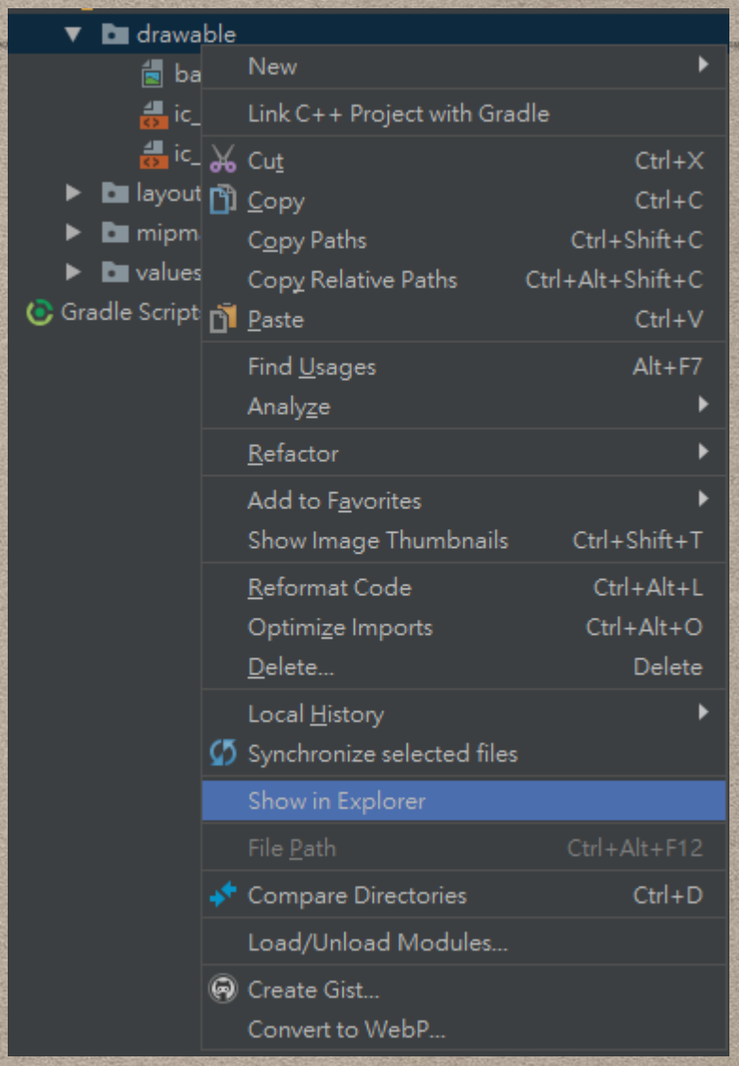
LET'S CONTROL AN IMAGE BY ACCELEROMETER



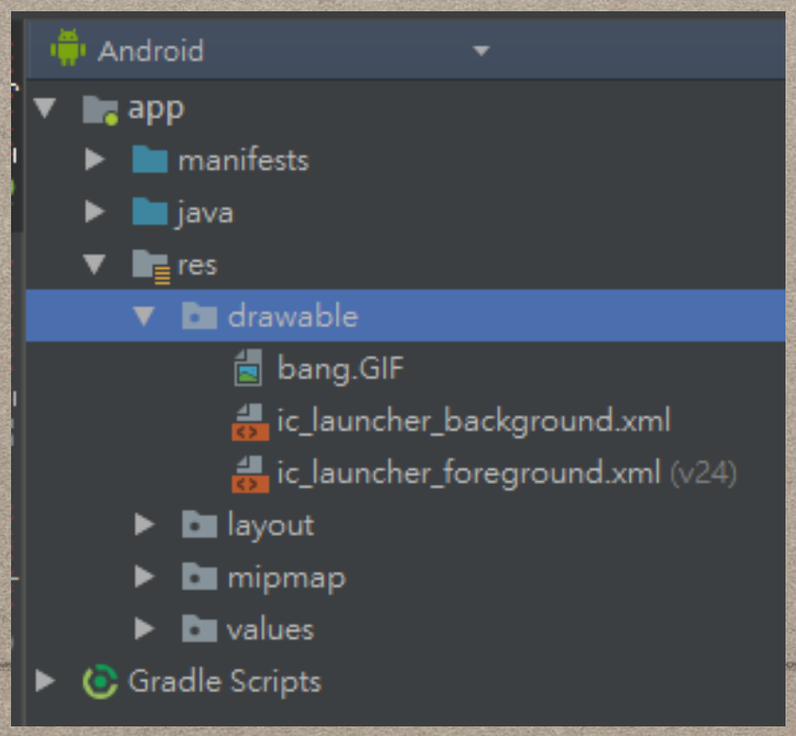
ADD A IMAGE VIEW

1. Add an image to the directory project/res/drawable

(對drawable右鍵->Show in Explorer)



1. add an ImageView in layout




```
<ImageView
    android:id="@+id/iv_bang"
    android:layout_width="141dp"
    android:layout_height="148dp"
    android:layout_alignParentBottom="true"
    android:layout_alignParentStart="true"
    android:scaleType="centerInside"
    app:srcCompat="@drawable/bang" />
```

3. declare an ImageView and LayoutParams

```
ImageView bang;
RelativeLayout.LayoutParams LP;
```

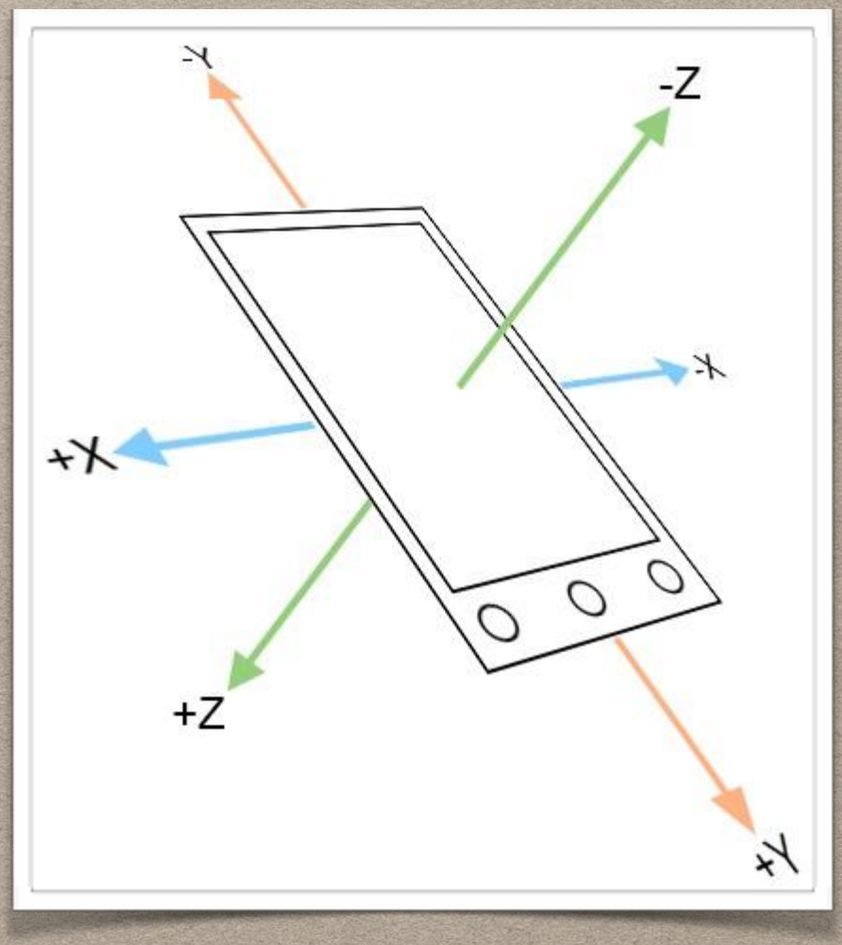
4. back to onCreate(), assign the object

```
bang = (ImageView) findViewById(R.id.iv_bang);
LP = (RelativeLayout.LayoutParams) bang.getLayoutParams();
```

Note:圖片外層要用RelativeLayout包住!!

5. Back to onSensorChanged(), we can control the image by changing the parameters.

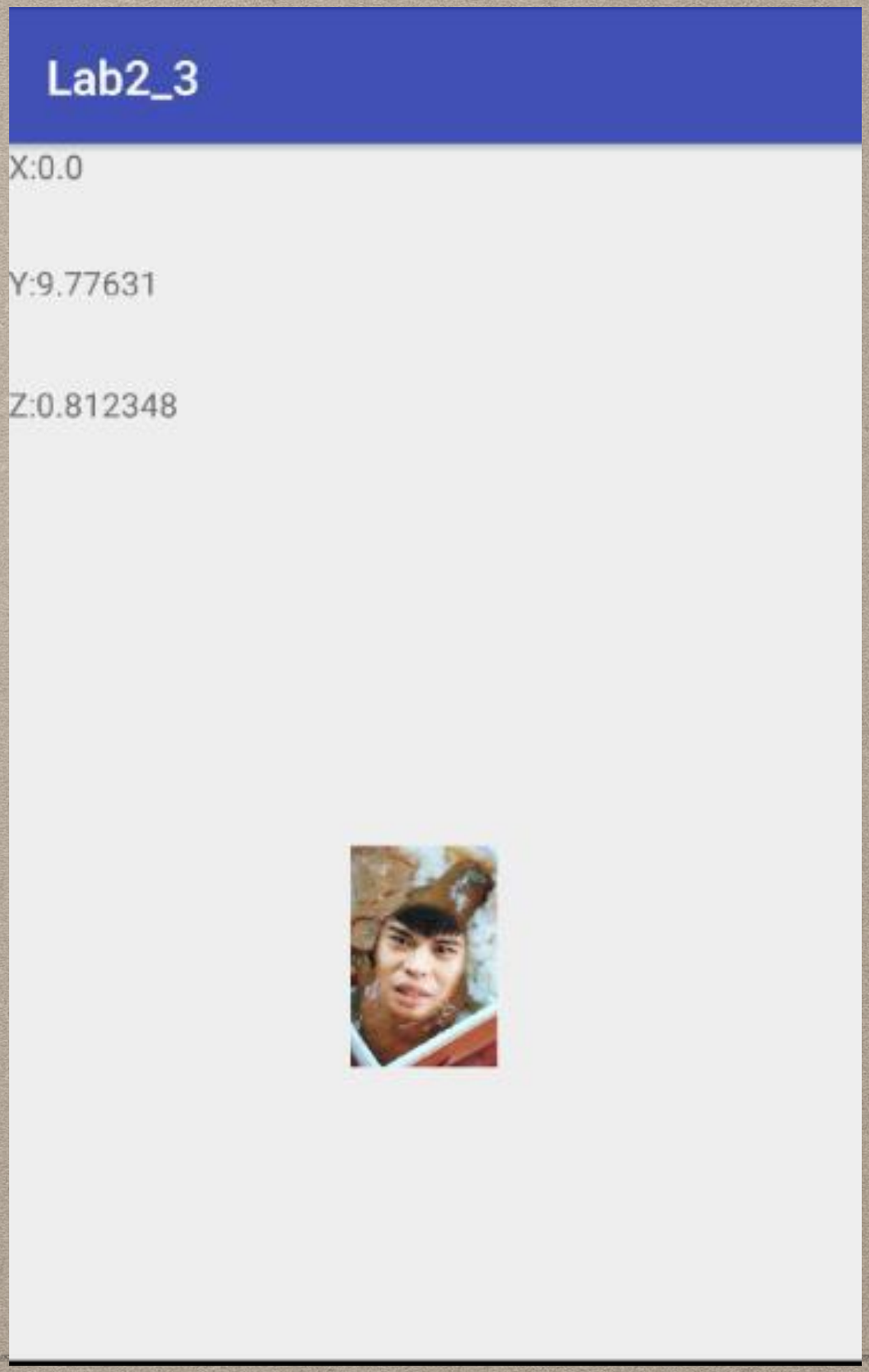
```
LP.leftMargin = ..... ;
LP.topMargin = ..... ;
bang.setLayoutParams(LP);
```



CHECK POINT-3

- A running picture(whatever you download).

  Hint:textView.getleft()…



Appendix

FIND OTHER SENSORS

- There are three categories of sensor
- Motion sensors
  These sensors measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.
- Environmental sensors
  These sensors measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers.
- Position sensors
  These sensors measure the physical position of a device. This category includes orientation sensors and magnetometers.

MONITOR MULTIPLE SENSORS

- we have to change our code to monitor multiple sensors. Change the Sensor name, then new a SensorEventListener when we register.

```
SensorManager mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE) ;
Sensor s1 = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) ;//set acce
mSensorManager.registerListener(new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent event) {
        x = event.values[0] ;
        y = event.values[1] ;
        z = event.values[2] ;

        tx.setText("AccelerometerX:" + x);
        ty.setText("AccelerometerY:" + y);
        tz.setText("AccelerometerZ:" + z);
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }
},s1,mSensorManager.SENSOR_DELAY_GAME) ;
```

```
SensorManager mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE) ;
Sensor s1 = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) ;//set acce
mSensorManager.registerListener(new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent event) {...}

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {...}
},s1,mSensorManager.SENSOR_DELAY_GAME) ;


Sensor s2 = mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE) ;
mSensorManager.registerListener(new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent event) {...}

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {...}
},s2,mSensorManager.SENSOR_DELAY_GAME) ;

Sensor s3 = mSensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY) ;
mSensorManager.registerListener(new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent event) {...}

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {...}
},s3,mSensorManager.SENSOR_DELAY_GAME) ;

Sensor s4 = mSensorManager.getDefaultSensor(Sensor.TYPE_AMBIENT_TEMPERATURE) ;
mSensorManager.registerListener(new SensorEventListener() {
    @Override
```

DEMO


100%
23:03
Sensor
AccelerometerX:-0.041152954
AccelerometerY:2.993454
AccelerometerZ:9.38414
GyroscopeX:0.009017944
GyroscopeY:0.023712158
GyroscopeZ:0.0023345947
Proximity:5.000305
Light:130.0