

Android終端系統匯流應用設計

感測器應用 – 加速度感應器

Brian Wang 王昱景

brian.wang.frontline@gmail.com

感應器介紹

- 感應器 (sensor) 是專門感應外界事物變化，並將其變化轉為數值的一種接收器
- 日常生活中常見的感應器有：溫度計 (感應外界溫度變化)、指北針 (感應南北極磁場)
- 電視遊樂器 Wii，其搖桿內藏加速度感應器，可以讓 Wii 透過該感應器知道搖桿傾斜的狀況來作適當的回應

- 與 Android 感應器有關的函式庫都放在「android.hardware」套件內
- 不是每一台 Android 行動裝置都有感應器，如果沒有對應的感應器，即使寫程式也無法取得對應的資料
- Android 模擬器無法模擬感應器功能，建議直接在實機上測試
- 目前所支援的感應器如下：

感應器	對應的值
加速度感應器	Sensor.TYPE_ACCELEROMETER
重力感應器	Sensor.TYPE_GRAVITY
陀螺儀感應器	Sensor.TYPE_GYROSCOPE
光線感應器	Sensor.TYPE_LIGHT
線性加速度感應器	Sensor.TYPE_LINEAR_ACCELERATION
磁場感應器	Sensor.TYPE_MAGNETIC_FIELD
方位感應器	SensorManager.getOrientation() 已取代 Sensor.TYPE_ORIENTATION
壓力感應器	Sensor.TYPE_PRESSURE
接近感應器	Sensor.TYPE_PROXIMITY
旋轉向量感應器	Sensor.TYPE_ROTATION_VECTOR
溫度感應器	Sensor.TYPE_TEMPERATURE

- 不論是何種感應器，最重要的就是取得其對外界感應後所蒐集到的數值
- 數值是以一個 float 陣列儲存，通常以 `values [i]` 來代表
- `values` 代表該陣列名稱，`i` 代表索引
- 依照不同的感應器，陣列的元素個數也會有所不同

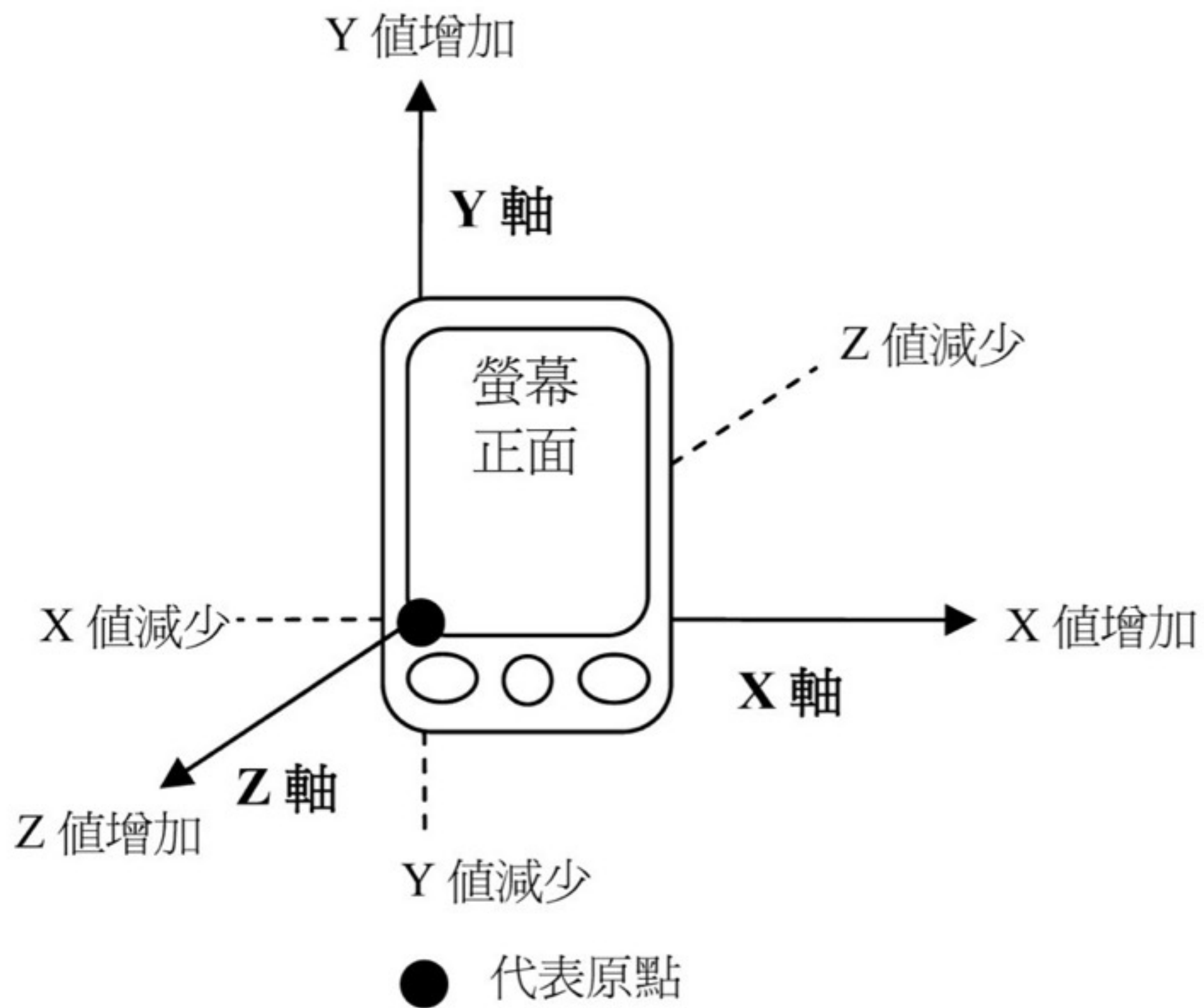
- 例如加速度與方位感應器都有 X 軸、Y 軸、Z 軸觀念，所以有 3 個數值：
`values[0]`、`values[1]`、`values[2]` 以儲存對應資訊
- 而接近感應器只有距離一個數值，所以只使用到 `values[0]`
- 一般 Android 行動裝置大部分都有加速度感應器、方位感應器、接近感應器與光線感應器的功能

加速度感應器

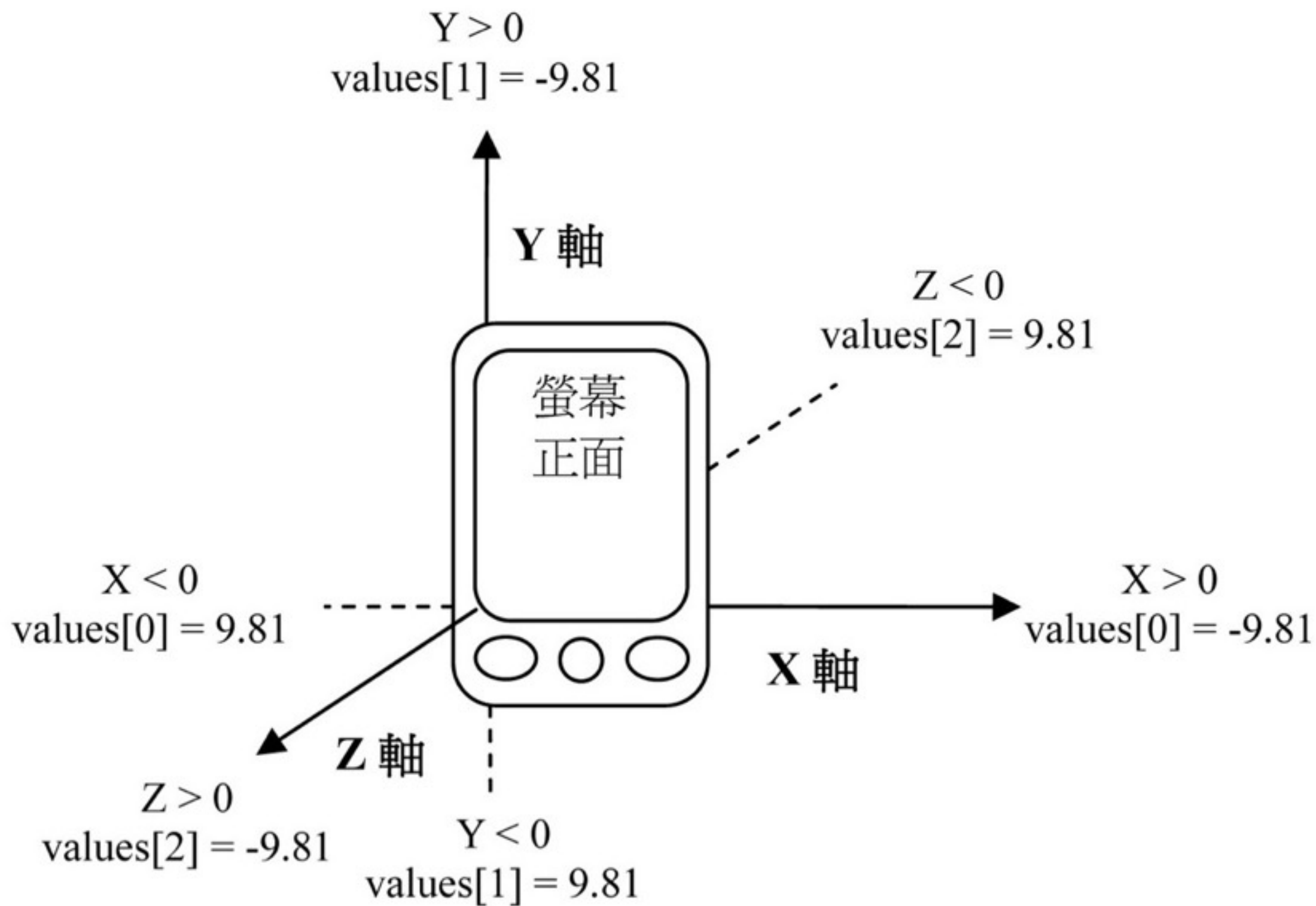
- 在說明加速度感應器之前，先說明 X 軸、Y 軸、Z 軸所代表的位置
- Android 採用 OpenGL ES 的座標系統

- 螢幕左下角頂點為原點 ($x=0, y=0, z=0$)，此與一般 2D 座標系統原點在螢幕左上角不同
- X 軸為左向右的水平方向，所以向右 X 值增加，向左 X 值減少
- Y 軸為下向上的垂直方向，所以向上 Y 值增加，向下 Y 值減少

- Z 軸為後向前的方向，所以向前 Z 值增加，向後 Z 值減少

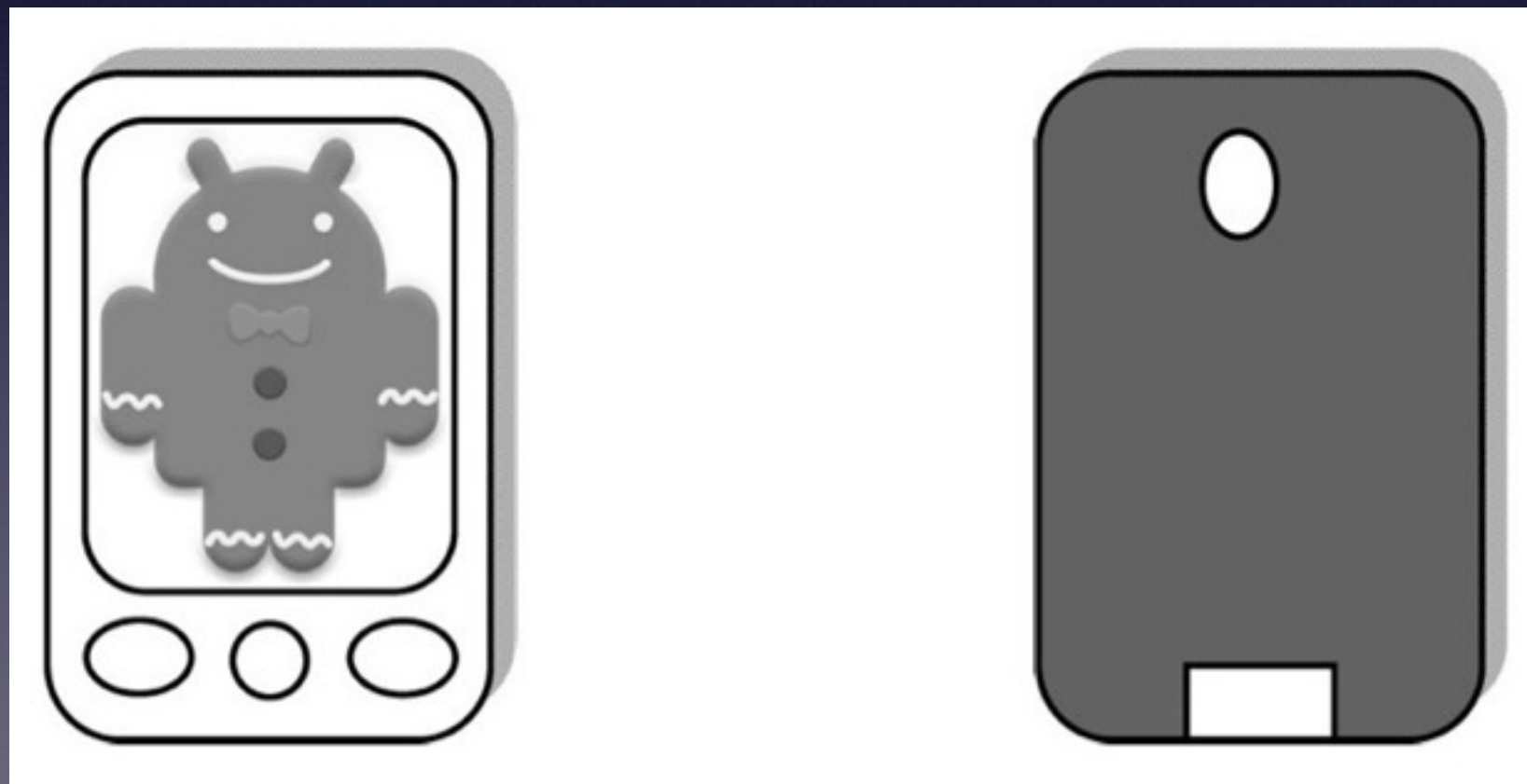


- 加速度的單位是 m/sec^2 (公尺/秒的平方)
- 加速度感應器則是反應 X 軸、Y 軸、Z 軸受到地心引力的影響情形
- 重力方向恰與座標方向相反，所以若符合重力方向與座標方向相反，會得到正的值，反之會得到負的值



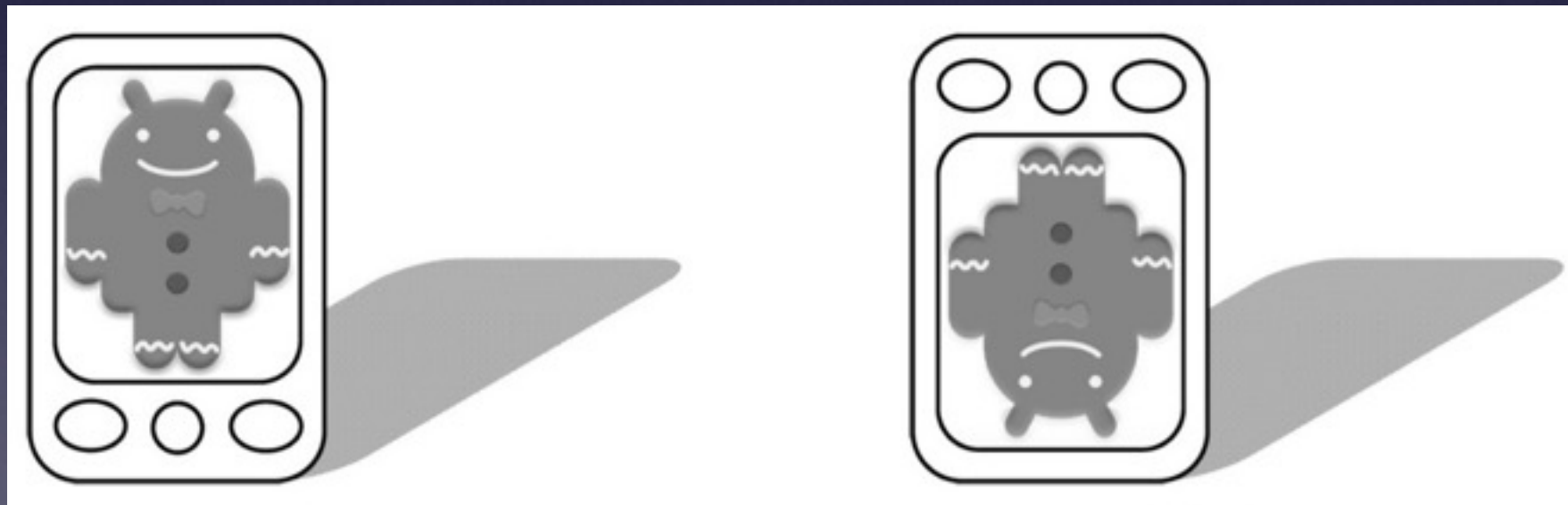
- 行動裝置平躺 (螢幕正面朝上)，此時 Z 軸受重力影響，values 值如下：
- $\text{values}[0] = 0.0$ ，代表 X 軸未受重力影響
- $\text{values}[1] = 0.0$ ，代表 Y 軸未受重力影響
- $\text{values}[2] = 9.81$ ，值為正代表 Z 軸後面方向($Z < 0$)受重力影響

- 若行動裝置平躺但螢幕正面朝下，背蓋朝上，則 $\text{values}[2] = -9.81$ ，代表 Z 軸前面方向受重力影響



- 行動裝置呈現縱向直立狀態 (稱為 portrait)，此時 Y 軸受重力影響，values 值如下：
- $\text{values}[0] = 0.0$ ，代表 X 軸未受重力影響
- $\text{values}[1] = 9.81$ ，值為正代表 Y 軸下面方向 ($Y < 0$) 受重力影響
- $\text{values}[2] = 0.0$ ，代表 Z 軸未受重力影響

- 若行動裝置縱向直立方式上下顛倒，則 $\text{values}[1] = -9.81$ ，代表 Y 軸上面方向受重力影響



- 行動裝置呈現橫向直立狀態 (稱為 landscape)，此時 X 軸受重力影響，values 值如下：
- $\text{values}[0] = 9.81$ ，代表 X 軸左面方向 ($X < 0$) 受重力影響
- $\text{values}[1] = 0.0$ ，代表 Y 軸未受重力影響
- $\text{values}[2] = 0.0$ ，代表 Z 軸未受重力影響

- 若行動裝置橫向直立方式左右顛倒，則 $\text{values}[0] = -9.81$ ，代表 X 軸右面方向受重力影響



AccelerometerDemo


- 開啟和執行 Android 專案
- 建立使用介面的版面配置
- 建立 Activity 活動類別

I.開啟和執行Android專案

- 請啟動 Eclipse IDE
- 建立 Android 專案
 - Project Name: AccelerometerDemo
 - Build Target: Android API 4.4
 - Package Name: tw.edu.uch

New Android Application

Creates a new Android Application



Application Name: ⓘ AccelerometerDemo

Project Name: ⓘ AccelerometerDemo


Package Name: ⓘ tw.edu.uch


Minimum Required SDK: ⓘ API 8: Android 2.2 (Froyo) ▾

Target SDK: ⓘ API 19: Android 4.4 (KitKat) ▾

Compile With: ⓘ API 19: Android 4.4 (KitKat) ▾

Theme: ⓘ Holo Light with Dark Action Bar ▾

 The package name must be a unique identifier for your application. It is typically not shown to users, but it **must** stay the same for the lifetime of your application; it is how multiple versions of the same application are considered the "same app". This is typically the reverse domain name of your organization plus one or more application identifiers, and it must be a valid Java package name.



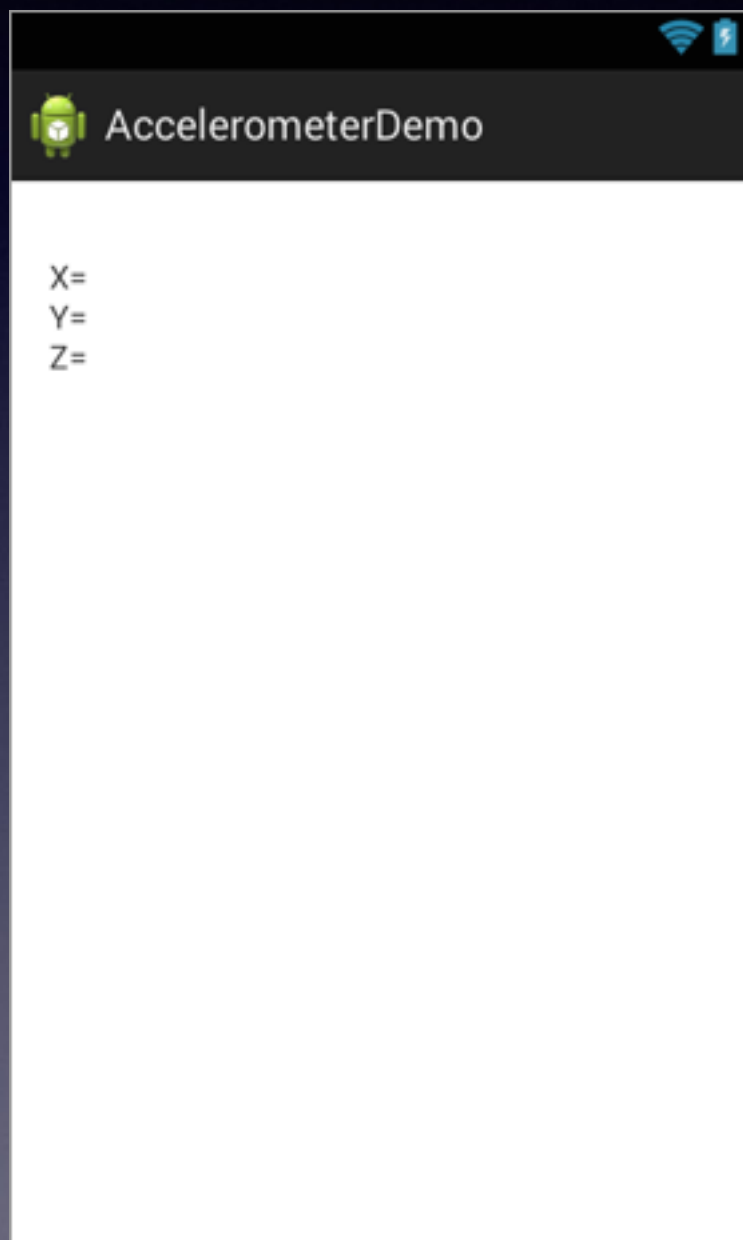
< Back

Next >

Cancel

Finish

2.建立使用介面的版面配置



```
<LinearLayout
    android:layout_width="match_parent" android:layout_height="wrap_content" >
    <TextView android:id="@+id/info"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/blank" />
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent" android:layout_height="wrap_content" >
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/x" />
    <TextView android:id="@+id/textx"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/blank" />
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent" android:layout_height="wrap_content" >
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/y" />
    <TextView android:id="@+id/texty"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/blank" />
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent" android:layout_height="wrap_content" >
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/z" />
    <TextView android:id="@+id/textz"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/blank" />
</LinearLayout>
```

3.建立 Activity 活動類別

- 在活動類別的開頭宣告成員變數

```
private SensorManager sensorManager;  
private boolean accelerometerPresent;  
private Sensor accelerometerSensor;  
private TextView textInfo, textX, textY, textZ;
```

• onCreate() 方法

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

    if (savedInstanceState == null) {
        getSupportFragmentManager().beginTransaction().add(R.id.container, new PlaceholderFragment()).commit();
    }

    textInfo = (TextView) findViewById(R.id.info);
    textX = (TextView) findViewById(R.id.textx);
    textY = (TextView) findViewById(R.id.texty);
    textZ = (TextView) findViewById(R.id.textz);

    sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    List<Sensor> sensorList = sensorManager.getSensorList(Sensor.TYPE_ACCELEROMETER);

    if (sensorList.size() > 0) {
        accelerometerPresent = true;
        accelerometerSensor = sensorList.get(0);

        StringBuilder sb = new StringBuilder();
        sb.append("Name: ").append(accelerometerSensor.getName()).append("\n");
        sb.append("Version: ").append(accelerometerSensor.getVersion()).append("\n");
        sb.append("Vendor: ").append(accelerometerSensor.getVendor()).append("\n");
        sb.append("Type: ").append(accelerometerSensor.getType()).append("\n");
        sb.append("Max: ").append(accelerometerSensor.getMaximumRange()).append("\n");
        sb.append("Resolution: ").append(accelerometerSensor.getResolution()).append("\n");
        sb.append("Power: ").append(accelerometerSensor.getPower()).append("\n");
        sb.append("Class: ").append(accelerometerSensor.getClass().toString());

        textInfo.setText(sb.toString());
    } else {
        accelerometerPresent = false;
    }
}
```


- onResume() 方法

```
@Override
protected void onResume() {
    super.onResume();

    if (accelerometerPresent) {
        sensorManager.registerListener(accelerometerListener, accelerometerSensor, SensorManager.SENSOR_DELAY_NORMAL);
        Toast.makeText(this, "Register accelerometerListener", Toast.LENGTH_LONG).show();
    }
}
```

- onStop() 方法

```
@Override
protected void onStop() {
    super.onStop();

    if (accelerometerPresent) {
        sensorManager.unregisterListener(accelerometerListener);
        Toast.makeText(this, "Unregister accelerometerListener", Toast.LENGTH_LONG).show();
    }
}
```

- SensorEventListener

```
private SensorEventListener accelerometerListener = new SensorEventListener() {  
  
    @Override  
    public void onAccuracyChanged(Sensor arg0, int arg1) {  
    }  
  
    @Override  
    public void onSensorChanged(SensorEvent event) {  
        textX.setText(String.valueOf(event.values[0]));  
        textY.setText(String.valueOf(event.values[1]));  
        textZ.setText(String.valueOf(event.values[2]));  
    }  
  
};
```