Appendix B

3D OpenGL ES

B.1 OpenGL ES元件

B.1.1 概論

OpenGLR ES 是免授權費的一個跨平臺、功能完善的 2D 和 3D 圖形應用程式介面 API，當然 Android 平臺也是其中之一，而 OpenGL 和 OpenGL ES 有何不同，OpenGL ES 針對多種嵌入式系統專門設計，包括移動電話、手持裝置、家電設備和汽車。它由精心定義的 OpenGL 核心功能所組成，創造了軟體與圖形加速間靈活強大的底層函數。OpenGL ES 包含浮點運算和定點運算系統描述，以及 EGL 針對設備的視窗系統規範。OpenGL ES 1.X 針對功能固定的硬體所設計並提供加速支援、圖形質量及性能標準。OpenGL ES 2.X 則提供包括材質和貼圖技術在內的全新 3D 圖形演算法。OpenGL ES-SC 專為有高安全性需求的特殊市場精心打造，而目前 Android SDK 1.5、1.6、2.0 只有執行 OpenGL ES 1.0（同 OpenGL 1.3）和部分的 OpenGL ES 1.1 基本的功能。

OpenGL ES 元件的範例程式，設定 OpenGL ES 的基本架構：Android 的 OpenGL 的範例程式，其路徑為：AndroidOpenGLESPowenko。

參考範例程式：

原始檔案：src/com/looptek/AndroidOpenGLESPowenko/TutorialPart001.java。

畫面設定資源程式碼：res/layout/main.xml。

OpenGL ES 元件的功能說明：

OpenGLR ES 主要是用在遊戲、高畫質畫面的動態影像等需要有快速畫面的軟體程式的應用上面。

B.2 OpenGL ES範例一 – OpenGL ES的基本架構，設定一個OpenGL ES View

SDK 版本：API level 1、Android 1.0

OpenGL ES 最基本的是建立一個 View 的基本架構，讓 Android 軟體可以執行 OpenGL ES 的程式，設定 View 為設定一個 OpenGL ES View。

範例 B-1 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart001.java

```
1. package com.looptek.AndroidOpenGLESPowenko;
2.
3. import android.app.Activity;
4. import android.opengl.GLSurfaceView;
5. import android.os.Bundle;
6.
7. public class TutorialPart001 extends Activity {
8.     /** Called when the activity is first created. */
9.     @Override
10.     public void onCreate(Bundle savedInstanceState) {
11.         super.onCreate(savedInstanceState);
12.          GLSurfaceView view = new GLSurfaceView(this);
13.            view.setRenderer(new TutorialPart001OpenGLRenderer());
14.            setContentView(view);
15.  }      }
```

這畫面設定文件 XML 中的重點為：

- GLSurfaceView 設定 OpenGL ES View 是用 TutorialPart001OpenGLRenderer class，GLSurfaceViewGLSurfaceView，它在 Android 1.5 是一個 API class 1.5，幫助你寫 OpenGL ES 應用程式，提供連接系統連結 OpenGL ES 到 View 的系統，並且連接系統讓 OpenGL ES 和 Activity 工作在一起。
- setContentView(view); 設定軟體的畫面是從 GLSurfaceView。

範例 B-2 sample\chB\\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\OpenGLRenderer.java

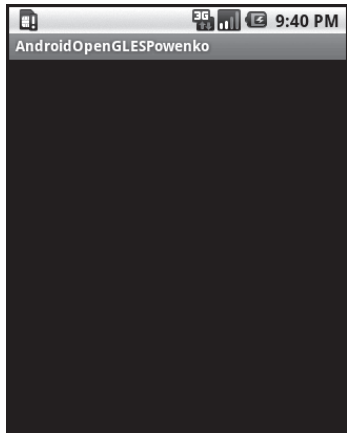
```
1. ...
2. public class TutorialPart001OpenGLRenderer implements Renderer {
3.
4.     public void onSurfaceCreated(GL10 gl, EGLConfig config) {
5.                                                      // 設定背景顏色
6.         gl.glClearColor(0.0f, 0.0f, 0.0f, 0.5f);    // 啟動滑順畫面的功能
8.         gl.glShadeModel(GL10.GL_SMOOTH);// OpenGL docs.
9.                                                      // Depth buffer setup.
10.         gl.glClearDepthf(1.0f);                     // 啟動畫面深度 Z
12.         gl.glEnable(GL10.GL_DEPTH_TEST);            // 啟動深度測試
14.         gl.glDepthFunc(GL10.GL_LEQUAL);             // 啟動深度測試的模式。
16.         gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, // 設定透視計算為最好的
18.          }
19.
20.
```

```
21.     public void onDrawFrame(GL10 gl) {
22.                 …
23.     }
24.
25.     public void onSurfaceChanged(GL10 gl, int width, int height) {
26.                 …
27.     }
28. }
```

```
1.
2.     public void onDrawFrame(GL10 gl) {
3.                     // 清除畫面和深度
4.         gl.glClear(GL10.GL_COLOR_BUFFER_BIT | // 清除畫面和緩衝區
5.                             GL10.GL_DEPTH_BUFFER_BIT);
6. }
```

```
1. public void onSurfaceChanged(GL10 gl, int width, int height) {
   // 當畫面被改變時
2.                 // 設定 opengl es 顯示的大小
3.         gl.glViewport(0, 0, width, height); // 設定 opengles 顯示的大小
5.         gl.glMatrixMode(GL10.GL_PROJECTION);// 選擇投影矩陣
7.         gl.glLoadIdentity();                // 重置投影矩陣
9.         GLU.gluPerspective(gl, 45.0f,  (float) width / (float) height,
    0.1f, 100.0f);
10.                // 設定畫面的長寬比
11.         gl.glMatrixMode(GL10.GL_MODELVIEW);// 選擇模型觀察矩陣
13.         gl.glLoadIdentity();               // 恢復初始位置
14. }
```

執行結果：



▲圖 B-1　範例執行結果

執行影片：

範例執行影片 B-1-background.mov (http://youtu.be/r9CPo-1DtLE)。

B.3 OpenGL ES元件的程式範例二 – 全畫面

SDK 版本：API level 1、Android 1.0

此範例會把畫面設定為全畫面，是 OpenGL ES 最基本的是建立一個 View 的基本架構，讓 Android 軟體可以執行 OpenGL ES 的程式，並且設定 View 為一個 OpenGL ES View。

基本架構：

Android 的 OpenGL 的範例程式，其路徑為：AndroidOpenGLESPowenko。

參考範例程式：

原始檔案：src/com/looptek/AndroidOpenGLESPowenko/TutorialPart002.java。

畫面設定資源程式碼：res/layout/main.xml。

程式範例：

我們在把剛剛的程式改一下變成全畫面。

範例 B-3 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart002.java

```
1. public class TutorialPart002 extends Activity {
2.     @Override
3.     public void onCreate(Bundle savedInstanceState) {
4.           super.onCreate(savedInstanceState);
5.        this.requestWindowFeature(Window.FEATURE_NO_TITLE); // 去除狀態欄
6.        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
7.            WindowManager.LayoutParams.FLAG_FULLSCREEN);    // 設定為全畫面
8.        GLSurfaceView view = new GLSurfaceView(this);
9.       view.setRenderer(new TutorialPart001OpenGLRenderer());
10.       setContentView(view);
11.     }
12. }
```

這個程式中的重點為：

- GLSurfaceView 設定 OpenGL ES View 是用 TutorialPart001OpenGLRenderer class、GLSurfaceView GLSurfaceView，它在 Android 1.5 是一個 API class 1.5，幫助你寫 OpenGL ES 應用程式，提供連接系統連結 OpenGL ES 到 View 的系統，並且連接系統讓 OpenGL ES 和 Activity 工作在一起。
- setContentView(view); 設定軟體的畫面是從 GLSurfaceView。

執行結果：

▲圖 B-2　範例執行結果

執行影片：

範例執行影片 B-2-black.mov(http://youtu. be/1sRTmv2zq7U)。

B.4 OpenGL ES元件的程式範例三 – 畫正方形

SDK 版本：API level 1、Android 1.0

此範例會把畫面設定為全畫面 OpenGL ES，並畫出一個正方形的白色方塊。整個程式最基本的是建立一個 View 的基本架構，讓 Android 軟體可以執行 OpenGL ES 的程式，設定 View 為設定一個 OpenGL ES View，並畫出一個正方形。

如何在 3D 的世界畫出人物，這就像是小時候使用竹子或鐵絲，想做出一個燈籠的道理是很像的。首先我們用竹子把燈籠的外型先架設好，然後再用紙糊上。建立多邊形是 3D Model 的最基本的元件，首先您必須先瞭解 3D models 的建立是從小的元件開始 (vertices, edges, faces, and polygons) ，這樣的動作就像是用竹子把燈籠的外型先架設好。

Vertex 頂點就如下圖所示。

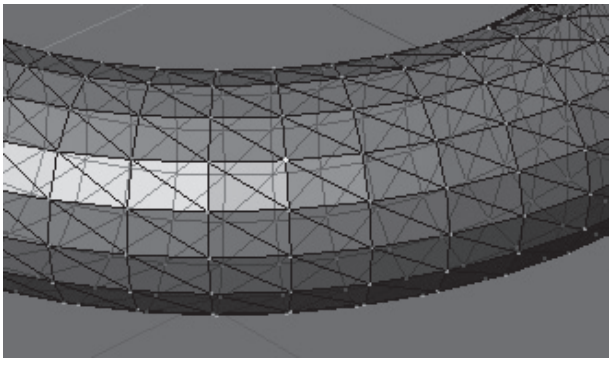


圖 B-3　黃色的點就是 Vertex 頂點

您可以看到圖中黃色的點就是頂點，而您可以看到那一個點被很多條黑色的線一起共用，所有的點可以很多連接成邊 edges、面 paces 和 polygons 多邊形，您可以想成是燈籠的竹子骨架，同樣的，攝影機或燈光都可以用頂點來定義。

2D 的世界表示方法

首先，我們先把這 2D 的四個點放在 3D 的表示方法，設定 Z 為 0，每一個點都是用 X, Y, Z 的方式來定義，而每一個數都是用 Float 來定義：

- V0 的話是 -1.0f, 1.0f, 0.0f, //X, Y, Z。
- V1 的話是 -1.0f, -1.0f, 0.0f, //X, Y, Z。
- V2 的話是 1.0f, -1.0f, 0.0f, //X, Y, Z。
- V3 的話是 1.0f, 1.0f, 0.0f, //X, Y, Z。

所以我們用矩陣的方式，把所有的資料放在一起。

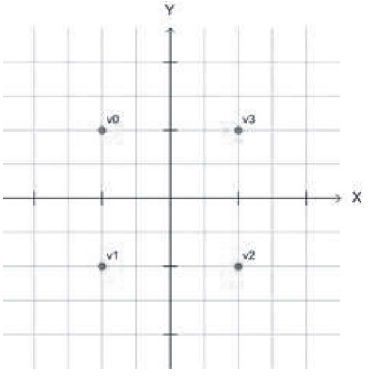


圖 B-4　2D 的世界表示方法

範例 B-4 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart003Square.java

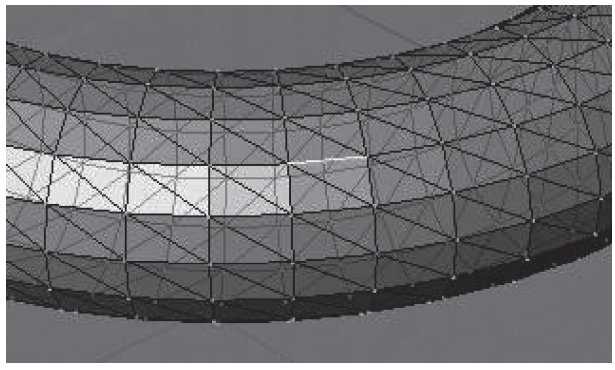
```
private float vertices[] =
{
  -1.0f, 1.0f, 0.0f,                          // 0, Top Left
  -1.0f, -1.0f, 0.0f,                         // 1, Bottom Left
   1.0f, -1.0f, 0.0f,                         // 2, Bottom Right
   1.0f, 1.0f, 0.0f                           // 3, Top Right};
```

再來把 vertices[] 的資料放到 OpenGL ES 之中，叫做 ByteBuffer class。

```
public TutorialPart003Square() {
// 設定位置
ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
vbb.order(ByteOrder.nativeOrder());
FloatBuffer vertexBuffer = vbb.asFloatBuffer();
vertexBuffer.put(vertices);
vertexBuffer.position(0);
…
}
```

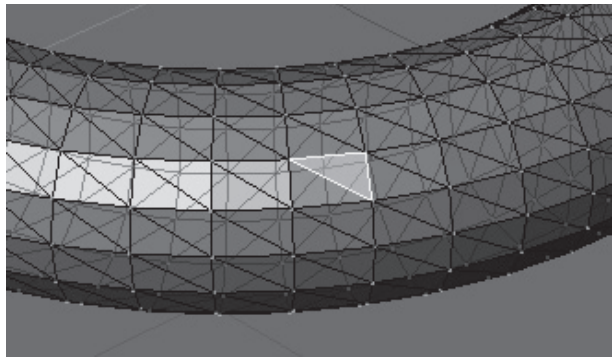
程式說明：

- 你在上面的程式中，一定會看到 ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);，這是因為所有的點都是由一個浮點數是 4 bytes 所組成的。
- FloatBuffer vertexBuffer = vbb.asFloatBuffer(); 設定資料大小。
- vertexBuffer.put(vertices); 把四個點的資料放入。
- vertexBuffer.position(0); 設定這四個點的位置。
- OpenGL ES 有個 pipeline functions，用來跟 render 顯示的部分溝通專用，而大部分的 functions 的預設值都沒有啟動，您要使用時，請記得啟動它們。



▲圖 B-5　線 Edge 邊

線 Edge 邊就是二個點 vertices 連起來的一條線。在 3D model 中線 edge 能夠被分享使用在二個面之間，或者是多邊形 polygons 之間。在 OpenGL ES 中基本定義上是沒有預設邊的 edges，你可以定義使用三個邊 edges 來組成一個面，請看圖中的黃色線 edge。



▲圖 B-6　面 Face

圖中所表現的是一個面 face，是由三個邊 edge 組成，我們用右旋的方式來一個點接一個點組成一個面，而設定 GL_CCW 是用逆時鐘的方式，把每一個點畫出來。

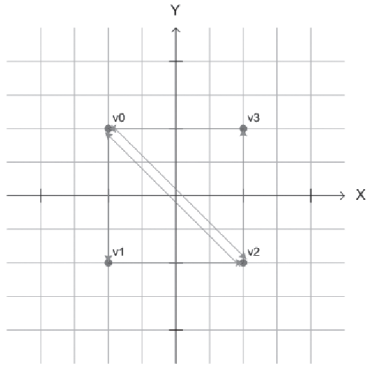
```
gl.glFrontFace(GL10.GL_CCW);
```

我們不需要設定哪一面是前面的面，哪一面是後面的面，因為執行速度效率的關係，我們不希望畫出二個面，因為前面的面會把後面給遮擋住，所以設定：

```
gl.glEnable(GL10.GL_CULL_FACE);
```

當然，如果你想要畫出背面的話，就把三數改成

```
gl.glCullFace(GL10.GL_BACK);
```



▲圖 B-7　畫出多邊形 Polygon

剛剛的面是由三個點所畫出來的，那如果要畫出四點的話要如何處理？

如上圖所表示的方法，透過二個三角形組成，v0 → v1 → v2 → v0 畫出一個三角形的面，然後 v0 → v2 → v3 → v0 畫出另外一個三角形，這樣就可以畫出一個多邊形。

範例 B-5 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart003Square.java

```
1. private short[] indices = { 0, 1, 2,
2.                                                    0, 2, 3 };
3.
4.     // 設定記憶體存放點的內容
5.     private ShortBuffer indexBuffer;
6.     public TutorialPart003Square() {
7.     ...
8.          // short is 2 bytes, therefore we multiply the number if
9.          // vertices with 2.
10.         ByteBuffer ibb = ByteBuffer.allocateDirect(indices.length * 2);
11.         ibb.order(ByteOrder.nativeOrder());
12.         indexBuffer = ibb.asShortBuffer();
13.         indexBuffer.put(indices);
14.         indexBuffer.position(0);15.     }
```

程式說明：

- private ShortBuffer indexBuffer; 要轉到 OpenGL ES 看得懂的格式。
- ByteBuffer ibb = ByteBuffer.allocateDirect(indices.length * 2); 宣告記憶體，short 的大小為 2 bytes。
- indexBuffer = ibb.asShortBuffer();indexBuffer 的格式是 Short。
- indexBuffer.put(indices); 把資料放進入。
- indexBuffer.position(0); 設定現在是第幾筆資料。

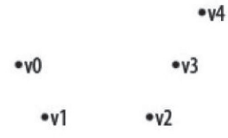
範例 B-6 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart003Square.java

```
51. public void draw(GL10 gl) {
52.           gl.glFrontFace(GL10.GL_CCW);    // 設定為順時鐘的方向來畫圖
53.           gl.glEnable(GL10.GL_CULL_FACE); // 設定要畫出來的是 GL_CULL_FACE 面
54.           gl.glCullFace(GL10.GL_BACK);
              // 撲殺背面，就是背面的話就不畫，以節省 CPU 的時間
55.           gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
              // 開啟 GL_VERTEX_ARRAY 用 Array 的資料來畫出圖
57.           gl.glVertexPointer(3, GL10.GL_FLOAT, 0, vertexBuffer);
              // 指出要出來的點，相關資料在 vertexBuffer 中
59.          gl.glDrawElements(GL10.GL_TRIANGLES, indices.length,
60.                                    GL10.GL_UNSIGNED_SHORT, indexBuffer);
```

```
61.                                        // 開始畫出點，並用 indexBuffer 的方法順序來畫
62.          gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
63.                                        // 停止更新，準備下次資料更新
64.           gl.glDisable(GL10.GL_CULL_FACE);
65. }
```

GL_POINTS

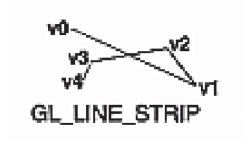
就是把每一個點都畫出來。



▲圖 B-8　GL_POINTS

GL_LINE_STRIP

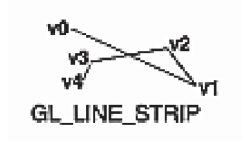
依照順序把點連接在一起。



▲圖 B-9　GL_LINE_STRIP

GL_LINE_LOOP

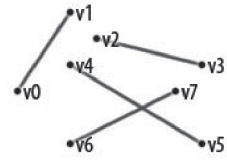
跟 GL_LINE_STRIP 很像，但把最後一個點連到第一個點。



▲圖 B-10　GL_LINE_STRIP

GL_LINES

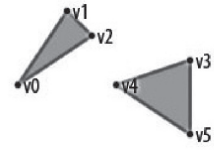
每一對的點畫成一條線。



▲圖 B-11　GL_LINES

GL_TRIANGLES

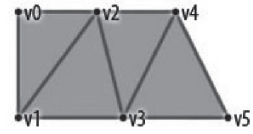
每三個點畫成一個三角形的面。



▲圖 B-12　GL_TRIANGLES

GL_TRIANGLE_STRIP

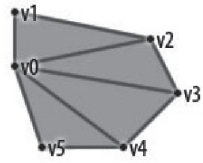
畫連續的三個點成一個三角形，使用點 v0, v1, v2, 和 v2, v1, v3 (請注意順序，因為要逆時鐘和 v2, v3, v4 等等，這樣的順序是為了確認所有畫出來的三角形是用同一個順時鐘或是逆時鐘的方式所畫出來)。



▲圖 B-13　GL_TRIANGLE_STRIP

GL_TRIANGLE_FAN

很像 GL_TRIANGLE_STRIP 的方式，但是畫的點是依照 v0, v1, v2, 和 v0, v2, v3, 和 v0, v3, v4, 等等。



▲圖 B-14　GL_TRIANGLE_FAN

以經驗來說。GL_TRIANGLES 是最簡單的方法，等一下我們的範例就用這個。

程式範例：

Android 的 OpenGL 的範例程式，其路徑為：AndroidOpenGLESPowenko。

原始檔案：src/com/looptek/AndroidOpenGLESPowenko/TutorialPart003.java。

畫面設定資源程式碼：res/layout/main.xml。

基本架構：

我們再把剛剛的程式改一下。

範例 B-7　sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart003.java

```
1. public class TutorialPart003OpenRenderer implements Renderer {
2.     private TutorialPart003Square square;
3.
4.     public TutorialPart003OpenRenderer() {          // 初始化方塊
5.         square = new TutorialPart003Square();
6.     }
7.
8.     public void onSurfaceCreated(GL10 gl, EGLConfig config) {
9.                                                     // 設定背景顏色
10.         gl.glClearColor(0.0f, 0.0f, 0.0f, 0.5f);
11.                                                    // 陰影模糊 smooth
12.         gl.glShadeModel(GL10.GL_SMOOTH);
13.                                                    // 深度暫存器設定
14.         gl.glClearDepthf(1.0f);
15.                                                    //啟動深度測試
16.         gl.glEnable(GL10.GL_DEPTH_TEST);
17.                                                    // 測試要做的深度型態。
18.         gl.glDepthFunc(GL10.GL_LEQUAL);
19.                                                    // 最好的透視 計算
20.         gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, GL10.GL_NICEST);
21.     }
22.     public void onDrawFrame(GL10 gl) {             // 清除屏幕和深度緩存
23.         gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
```

```
24.                                          // 初始化現在的矩陣。
25.         gl.glLoadIdentity();
26. …
27.                                          // 繪製方塊
28.         square.draw(gl) ;                // 畫出
29.     }
32.     public void onSurfaceChanged(GL10 gl, int width, int height) {
33.                                          // 設定現在的視窗大小
34.     ....
46.     }
47. }
```

範例 B-8 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart003Square.java

```
1. import javax.microedition.khronos.opengles.GL10;
2. public class TutorialPart003Square {
3.                                                  // 設定資料的點
4.     private float vertices[] = {
5.               -1.0f,  1.0f, 0.0f,               // 0, 左上方
6.               -1.0f, -1.0f, 0.0f,               // 1, 左下方
7.                1.0f, -1.0f, 0.0f,               // 2, 右下方
8.                1.0f,  1.0f, 0.0f,               // 3, 右上方
9.         };
10.                                                  // 連接的號碼順序
11.     private short[] indices = { 0, 1, 2, 0, 2, 3 };
14.     private FloatBuffer vertexBuffer;
15.     private ShortBuffer indexBuffer;
16.     public TutorialPart003Square() {
17.                                                  // 一個 float 是 4 個 byte
18.         ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
19.         vbb.order(ByteOrder.nativeOrder());
20.         vertexBuffer = vbb.asFloatBuffer();
21.         vertexBuffer.put(vertices);
22.         vertexBuffer.position(0);
23.                                                  // short 是 2 個 bytes,
24.         ByteBuffer ibb = ByteBuffer.allocateDirect(indices.length * 2);
25.         ibb.order(ByteOrder.nativeOrder());
26.         indexBuffer = ibb.asShortBuffer();
27.         indexBuffer.put(indices);
28.         indexBuffer.position(0);
29.     }
30. }
```

把 onDrawFrame 改一下：

```
public void onDrawFrame(GL10 gl) {
…
```

```
gl.glClear(GL10.GL_COLOR_BUFFER_BIT |  GL10.GL_DEPTH_BUFFER_BIT);
square.draw(gl); // ( NEW ) // Draw our square.
}
```

如果你執行這一個應用程式有問題的話，那是因為整個畫面都是黑色的，所以你完全看不到，而 OpenGL ES 的渲染顯示的部分，初始值是 0,0,0。剛好這一個位置也是 view port 的位置，OpenGL ES 無法顯示太靠近的 view port。解決方法很容易，就是把 draw position 移往後面幾個位置就好了。

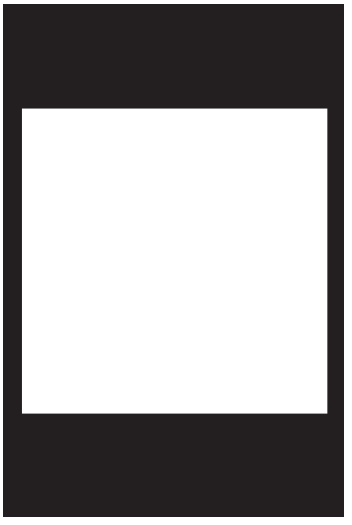
```
// Translates 4 units into the screen.
gl.glTranslatef(0, 0, -4); // OpenGL docs
```

之後會討論有關於物件位置轉換的方法，修改後，你就會看到方塊被畫出，但是那一個方塊會越來越小，因為 OpenGL ES 沒有重新回到要畫的那一個點 drawing point，而是把前一個 frame 的 z=z-4 一直做下去，解決方法就是加上下面這一個程式就好。

```
// Replace the current matrix with the identity matrix
gl.glLoadIdentity(); // OpenGL docs
```

現在，終於看到一個正方形出現了，範例程式的執行結果如下所示。

執行結果：



▲圖 B-15　範例執行結果

執行影片：

範例執行影片 B-3-square.mov(http://youtu.be/ h8gvdiwEK0Q) 。

B.5 OpenGL ES的程式範例四 – 物件移動

SDK 版本：API level 1、Android 1.0

本章節要討論如何移動多邊形的物件，在移動物件座標時有幾個方法，可以用數學演算法，也能夠透過 openGL ES 相對應的函數或是自動計算。而數學原理是透過每一個點 X,Y,Z 位置，使用「矩陣相乘」計算的數學方法就可以做到移動、旋轉、放大和縮小，透過矩陣相乘之後改變所有頂點位置，相乘之後就是物件的新位置結果。OpenGL ES 相對應的函數，我們只要知道如何使用函數就好了，不用拿紙和筆來算數學。

B.5.1 座標系統 (Coordinate System)

OpenGL 使用右手座標系統，簡單的說 Z 數字越大就越靠近螢幕，而 Z 數字越小的話就是離螢幕遠，如下圖所表現的 X 的話，是右邊大，左邊小，Y 的話，是上面大，下面小。

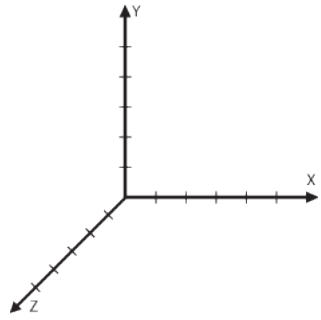


圖 B-16 右手座標的世界

B.5.2 移動 (Translate)

```
public abstract void glTranslatef (.oat x, .oat y, .oat z)
```

這一個函數主要功能就是做移動的動作，不包含旋轉。

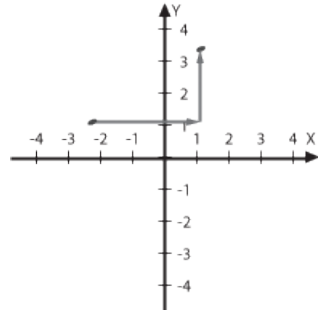


圖 B-17 移動 Translate

2D 的話，開始在 {x:-2, y:1}，最後希望能在停到 {x:1, y:3}，所以需要加上 {x:3, y:2}。

整段來看的話如下：

```
{x:-2, y:1} + {x:3, y:2} = {x:-2 + 3, y:1 + 2} = {x:1, y:3}.
```

3D 的話，開始在 {x:1, y:1,z:0}，最後希望能在停到 {x:1, y:1, z:-3}，所以需要加上 {x:0, y:0, z:-3}。

整段來看的話如下：

```
{x:1, y:1,z:0} +{x:0, y:0, z:-3}= {x:1 + 0, y:1 + 0, z: 0-3 } = {x:1, y:1, z:-3}
```

然後我們會將剛剛的四角形多邊形，從原本的 {x:0, y:0, z:0} 把它移到整體的 view {x:0, y:0, z:-4}。

```
gl.glTranslatef(0, 0, -4); // Translates 4 units into the screen.
```

B.5.3 旋轉（Rotate）

可以使用旋轉的函數

```
public abstract void glRotatef(.oat angle, .oat x, .oat y, .oat z)
```

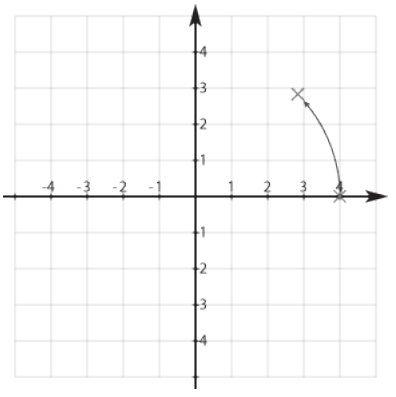


圖 B-18　Rotate 旋轉

以函數做 (x:0,y:0,z:0) 旋轉一定角度，而角度的範圍在 -180,-90,0, 90,180 等等。

注意

OpenGL use degrees 是用角度。

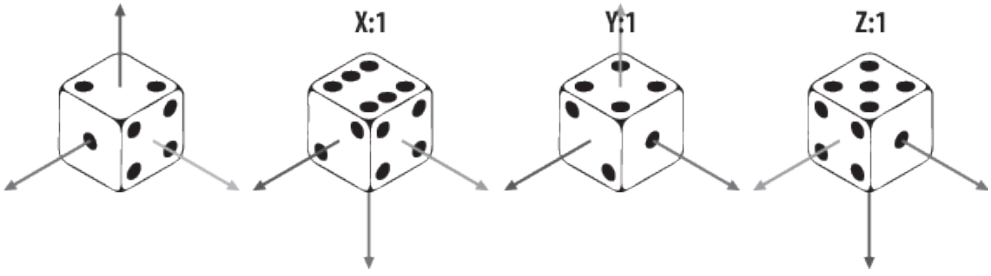
如果多次使用旋轉的函數，情況如下：

- gl.glRotatef(90f, 1.0f, 0.0f, 0.0f);
- gl.glRotatef(90f, 0.0f, 1.0f, 0.0f);
- gl.glRotatef(90f, 0.0f, 0.0f, 1.0f);

x: 藍色

y: 紅色

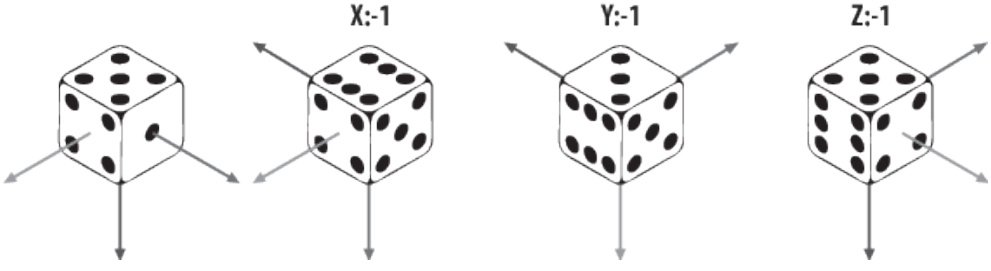
z: 綠色



▲圖 B-19　Rotate 旋轉三次後的結果

然後再用負數轉回來的話：

- gl.glRotatef(90f, -1.0f, 0.0f, 0.0f);
- gl.glRotatef(90f, 0.0f, -1.0f, 0.0f);
- gl.glRotatef(90f, 0.0f, 0.0f, -1.0f);

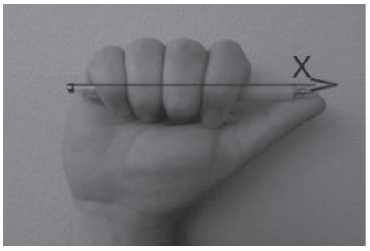


▲圖 B-20　用負數轉 Rotate 旋轉三次後的結果

你會發現，剛剛的回轉無法回來原本的樣子，但是如果依照這樣的順序的話，就沒有問題了。

- gl.glRotatef(90f, 0.0f, 0.0f, -1.0f);
- gl.glRotatef(90f, 0.0f, -1.0f, 0.0f);
- gl.glRotatef(90f, -1.0f, 0.0f, 0.0f);

提供一個經驗談，90 度的方向是逆時鐘轉 90 度，很多時候，你會亂掉轉到哪一個方位，所以我會用右手拿一支筆，這樣手指頭的方向就是逆時鐘。

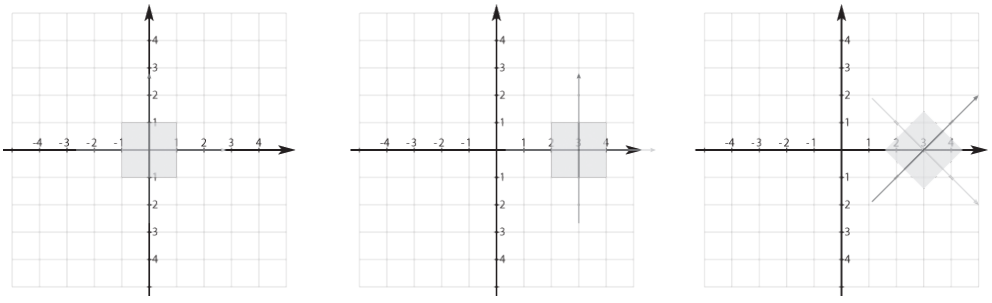


▲圖 B-21　右手是逆時鐘

B.5.4　移動和旋轉 (Translate & Rotate)

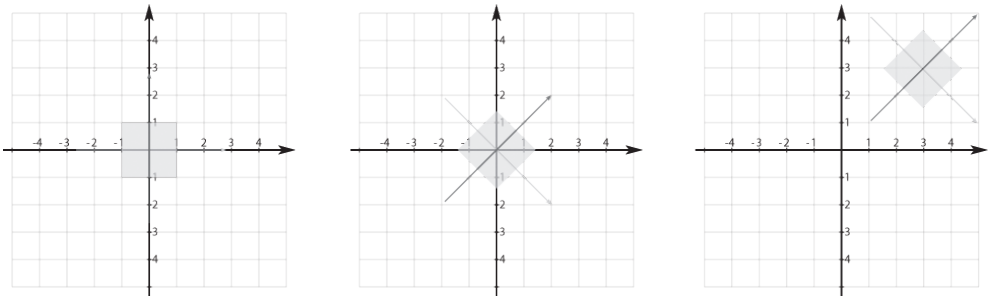
我們剛剛分別介紹移動和旋轉，但是放在一起的時候要注意哪些？

如果先移動，然後再旋轉，就會變成下面的狀況。



▲圖 B-22　先移動，然後再旋轉

但是如果旋轉，然後再移動的話，就會變成下面的情況。



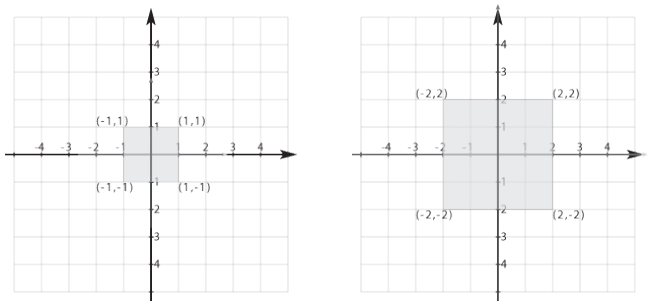
▲圖 B-23　先旋轉，再移動

縮放 (Scale)

public abstract void glScalef (.oat x, .oat y, .oat z)

假如縮放 2f。

```
gl.glScalef(2f, 2f, 2f);
```

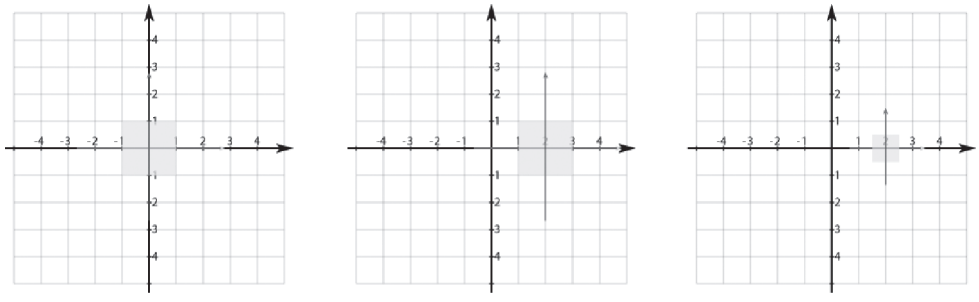


▲圖 B-24　假如縮放 2f

移動和縮放 (Translate & Scale)

移動和縮放 0.5f。

```
gl.glTranslatef(2, 0, 0);
gl.glScalef(0.5f, 0.5f, 0.5f);
```



▲圖 B-25　移動 x=2 和縮放 0.5f

但是如果你先做出縮小的動作，然後做移動的話，出現情況就會不一樣，你會發現它向右移 1 個單位。

```
gl.glScalef(0.5f, 0.5f, 0.5f);
gl.glTranslatef(2, 0, 0);
```

B.5.5 初始矩陣、放入矩陣和取出矩陣（Load Identity, push and pop matrix）

透過剛剛幾個範例，你可發現如果你做一連串的旋轉、移動時，整個結果便會出現意想不到的狀況，所以 OpenGL ES 便提供這三個函數。

- glLoadIdentity
- glPushMatrix
- glPopMatrix

glLoadIdentity

```
public abstract void glLoadIdentity()
```

glLoadIdentity 會取代先現有的矩陣資料，當使用它時，便會把矩陣的資料變成像下的情況：

1 0 0 0

0 1 0 0

0 0 1 0

0 0 0 1

glPushMatrix

```
public abstract void glPushMatrix()
```

把矩陣的資料放在堆疊中。這樣計算完畢後，就可以取回，意思是先把矩陣資料放到堆疊，之後再把資料從堆疊中取出來。

glPopMatrix

```
public abstract void glPopMatrix()
```

把矩陣的資料取回。意思是剛剛已經先把矩陣資料放到堆疊，然後用 glPopMatrix 把資料從堆疊中取出來。

B.5.6 物件轉換（三個方塊移動和旋轉的基本架構）

SDK 版本：API level 1、Android 1.0

Android 的 OpenGL 的範例程式其路徑為：AndroidOpenGLESPowenko。

原始檔案：src/com/looptek/AndroidOpenGLESPowenko/TutorialPart005.java。

程式介紹：

這個範例是用剛剛上一個範例做修改，一個有三個方塊，並做移動和旋轉的計算。現在把所有的程式碼放在一起，並且畫出三個方塊，叫做 A、B、C，我們分別用不同的方式來處理，你就可以看出差別了。

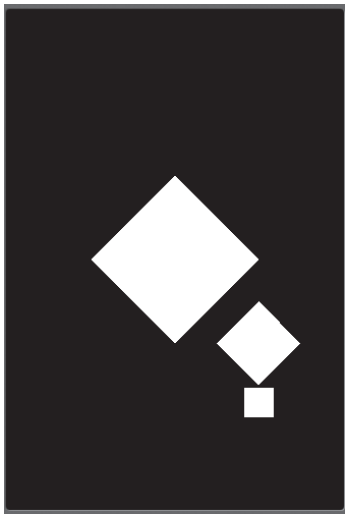
範例 B-9 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart005.java

```
public void onDrawFrame(GL10 gl) {
Angle=0;
gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
                                  // 清空
gl.glLoadIdentity();
gl.glTranslatef(0, 0, -10);       // 移動
gl.glPushMatrix();  // SQUARE A   // 儲存位置
gl.glRotatef(angle, 0, 0, 1);     // 旋轉
square.draw(gl);                  // 畫出 A.
gl.glPopMatrix();                 // 恢復
                                  // SQUARE B
gl.glPushMatrix();                // 儲存位置
gl.glRotatef(-angle, 0, 0, 1);    // 旋轉 A.
gl.glTranslatef(2, 0, 0);         // 移動 B.
gl.glScalef(.5f, .5f, .5f);       // 縮小 50%
square.draw(gl);                  // 畫出
                                  // SQUARE C
gl.glPushMatrix();
                                  // 儲存位置
gl.glRotatef(-angle, 0, 0, 1);    // 旋轉
gl.glTranslatef(2, 0, 0);
gl.glScalef(.5f, .5f, .5f);       // 縮小 50% B
gl.glRotatef(angle*10, 0, 0, 1);  // 旋轉
square.draw(gl);                  // 畫出 C.
gl.glPopMatrix();                 // 資料恢復到 C.
gl.glPopMatrix();                 // 資料恢復到 B.}
```

然後不要忘記 angle 的變數宣告：

```
public class OpenGLRenderer implements Renderer
{
private float angle;
...
}
```

執行結果：



▲圖 B-26　範例執行結果

B.5.7　更多的學習

把這個範例再修改一下。把私有變數 Angle 做增加，這樣原本的三個方塊，會一起做移動和旋轉的計算，看來就像是太陽系，月亮繞著地球轉，地球再繞著太陽轉，然後一直轉個不停。

```
    private float angle;
    public void onDrawFrame(GL10 gl) {
        // Clears the screen and depth buffer.
        …
        // Increse the angle.
        // angle=45;
        angle++;
    }
```

執行結果：

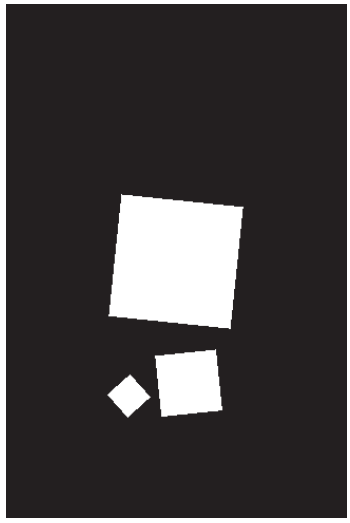


圖 B-27　範例執行結果

執行影片：

範例執行影片 B-4-rotation.mov(http://youtu. be/zUeIG0DEWBE)。

B.6 OpenGL ES的程式範例五 – 物件的畫法 glDrawElements

在前面畫方塊的範例中，您是否還記得 glDrawElements

GL_POINTS

就是把每一個點都畫出來。

圖 B-28　GL_POINTS

GL_LINE_STRIP

依照順序把點連接在一起。

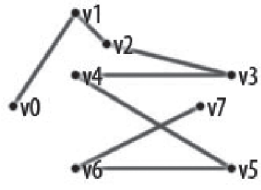
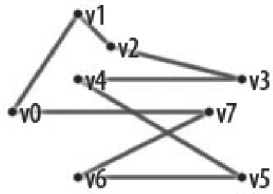


圖 B-29　GL_LINE_STRIP

GL_LINE_LOOP

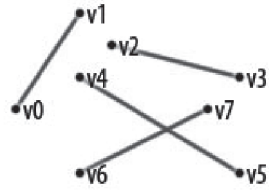
跟 GL_LINE_STRIP 很像，但把最後一個點連到第一個點。



▲圖 B-30　GL_LINE_LOOP

GL_LINES

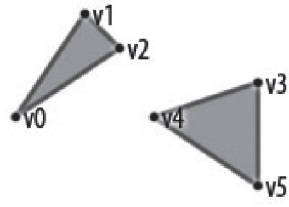
每一對的點畫成一條線。



▲圖 B-31　GL_LINES

GL_TRIANGLES

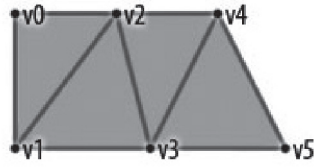
每三個點畫成一個三角形的面。



▲圖 B-32　GL_TRIANGLES

GL_TRIANGLE_STRIP

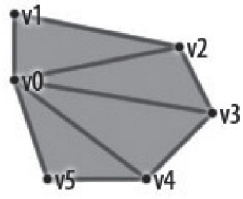
畫連續的三個點成一個三角形，使用點 v0, v1, v2, 和 v2, v1, v3 (請注意順序，因為要逆時鐘和 v2, v3, v4 等等這樣的順序，是為了確認所有畫出來三角形是用同一個順時鐘或是逆時鐘的方式所畫出來)。



▲圖 B-33　GL_TRIANGLE_STRIP

GL_TRIANGLE_FAN

很像 GL_TRIANGLE_STRIP 的方式，但是畫的點是依照 v0, v1, v2 和 v0, v2, v3 和 v0, v3, v4 等等。



▲圖 B-34　GL_TRIANGLE_FAN

基本架構：

Android 的 OpenGL 的範例程式，其路徑為：AndroidOpenGLESPowenko。

原始檔案：src/com/looptek/AndroidOpenGLESPowenko/TutorialPart005.java。

程式範例：

延續範例 src/com/looptek/AndroidOpenGLESPowenko/TutorialPart003.java 來做修改，我們先把方塊產生五個，然後再用不同的 glDrawElements 來畫出來。

Android 原始檔案 TutorialPart005OpenRenderer.java 的程式碼片段修改如下所示：

範例 B-10 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart005OpenRenderer.java

```
1. public void onDrawFrame(GL10 gl) {
2.         // Clears the screen and depth buffer.
3.         gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
4.         // 把位置大小旋轉重置
5.         gl.glLoadIdentity();
6.         gl.glTranslatef(-2.2f, 2.5f, -10);
7.         gl.glScalef(.5f, .5f, .5f);
8.         // 畫出
9.         square.draw(gl,1);
10.
11.         gl.glLoadIdentity();
12.         gl.glTranslatef(0, 2.5f, -10);
13.         gl.glScalef(.5f, .5f, .5f);
14.         // 畫出
15.         square.draw(gl,2);
16.
17.         gl.glLoadIdentity();
18.         gl.glTranslatef(2.2f, 2.5f, -10);
19.         gl.glScalef(.5f, .5f, .5f);
20.         // 畫出
21.         square.draw(gl,3);
22.         gl.glLoadIdentity();
23.         gl.glTranslatef(-2.2f,0, -10);
24.
25.         gl.glScalef(.5f, .5f, .5f);
26.         // 畫出
27.         square.draw(gl,4);
28.
29.         gl.glLoadIdentity();
30.         gl.glTranslatef(0, 0, -10);
31.
32.         gl.glScalef(.5f, .5f, .5f);
33.         // 畫出
34.         square.draw(gl,5);
35.
36. }
```

Android 原始檔案 TutorialPart005Square.java 的程式碼片段修改如下所示：

範例 B-11 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart005Square.java

```
1. public void draw(GL10 gl,int i_action) {
2.
3.          // Counter-clockwise winding.
4.          gl.glFrontFace(GL10.GL_CCW);
5.          // 啟用背面不畫
6.          gl.glEnable(GL10.GL_CULL_FACE);
8.          gl.glCullFace(GL10.GL_BACK);
9.          // 啟用的頂點緩衝器寫入和過程中使用的渲染
11.         gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
15.         gl.glVertexPointer(3, GL10.GL_FLOAT, 0, vertexBuffer)
16.         if (i_action==1){
17.             gl.glDrawElements(GL10.GL_POINTS , indices.length,
18.                 GL10.GL_UNSIGNED_SHORT, indexBuffer);
19.         }else if (i_action==2){
20.             gl.glDrawElements(GL10.GL_LINE_STRIP  , indices.length,
21.                 GL10.GL_UNSIGNED_SHORT, indexBuffer);
22.         }else if (i_action==2){
23.             gl.glDrawElements(GL10.GL_LINE_LOOP  , indices.length,
24.                     GL10.GL_UNSIGNED_SHORT, indexBuffer);
25.         }else if (i_action==3){
26.             gl.glDrawElements(GL10.GL_LINES   , indices.length,
27.                     GL10.GL_UNSIGNED_SHORT, indexBuffer);
28.         }else if (i_action==4){
29.             gl.glDrawElements(GL10.GL_TRIANGLES  , indices.length,
30.                     GL10.GL_UNSIGNED_SHORT, indexBuffer);
31.         }else if (i_action==4){
32.             gl.glDrawElements(GL10.GL_TRIANGLE_STRIP   , indices.length,
33.                     GL10.GL_UNSIGNED_SHORT, indexBuffer);
34.         }else if (i_action==5){
35.             gl.glDrawElements(GL10.GL_TRIANGLE_FAN    , indices.length,
36.                     GL10.GL_UNSIGNED_SHORT, indexBuffer);
37.         }
38.
39.         // Disable the vertices buffer.
40.         gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
41.         // Disable face culling.
42.         gl.glDisable(GL10.GL_CULL_FACE);
43. }
```

這一段程式的意思是

```
gl.glDrawElements(GL10.GL_POINTS , indices.length,
```

- GL10.GL_UNSIGNED_SHORT, indexBuffer); 這就是這一個範例的重點，用不同的方式畫出來這四個點。

執行結果：

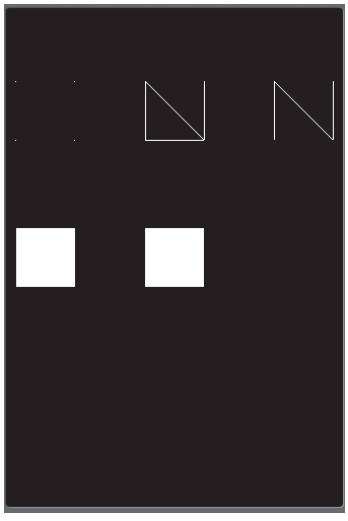


圖 B-35　範例程式的執行結果

執行影片：

範例執行影片 B-5-point.mov(http://youtu.be/ l2NeoYjKCPE)。

B.7 OpenGL ES範例六 – 顏色

SDK 版本：API level 1、Android 1.0

現在我們來加上顏色，3D models 如果沒有顏色，實在很無趣，我們現在來說明如何加上顏色，在 OpenGL ES 通常使用 RGBA (紅 Red, 綠 Green, 藍 Blue and 和透明 Alpha) 來定義顏色，使用如「#FF00FF」的 16 進位元的方式表現，或是用「255, 0, 255」10 進位來表現，設定顏色的方式都是用點，意思是每一個點設定一個顏色，常看的平面顏色是設定一個固定的顏色，給每一個點 vertex，同樣的道理，材質的運用也是一樣。

B.7.1 平面顏色 (Flat coloring)

OpenGL ES 有兩種設定顏色的設定方法，其中平面顏色是一個比較簡單的方法，只要跟 OpenGL ES 說要顯示什麼顏色就好了。

圖 B-36 設定相同的顏色

只要用這一個函數就可以。設定顏色的值 red = 1、green = 1、blue = 1、alpha = 1。

```
public abstract void glColor4f(.oat red, .oat green, .oat blue, .oat alpha)
```

那實際要如何使用？如下面的範例：

```
public void draw(GL10 gl)
{       ...
gl.glColor4f(1.0f,0.0f, 0.0f, 1.0f);
  square.draw(gl);
       ...
}
```

這個程式中的重點為：

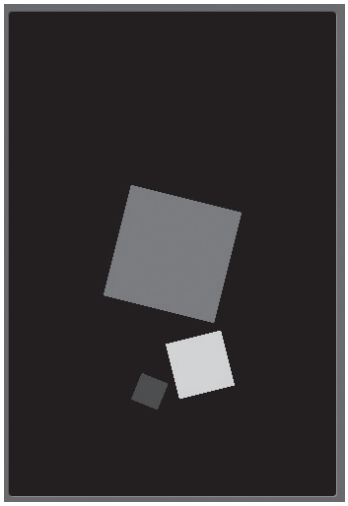
- gl.glColor4f(1.0f, 0.0f, 0.0f, 1.0f); 顏色的意思是紅色 red = 255/255、綠色 green = 0/255、藍色 blue = 0/255 和透明 = 255/255，也就是不透明。或者可以說 0Xff0000ff。

基本架構：

Android 的 OpenGL 的範例程式，其路徑為：AndroidOpenGLESPowenko。

原始檔案：src/com/looptek/AndroidOpenGLESPowenko/TutorialPart006.java。

執行結果：



▲圖 B-37　範例程式的執行結果

執行影片：

範例執行影片 B-6-ani.mov(http://youtu.be/ tPMnxTILBmA)。

B.8 OpenGL ES的程式範例七 – 漸層顏色 (Smooth coloring)

SDK 版本：API level 1、Android 1.0

漸層顏色 (Smooth coloring) 是設定每一個點各擁有的顏色，然後經過設定之後，OpenGL ES 便會自己幫你產生中間漸層的效果。

▲圖 B-38　漸層顏色

為了達到這個目的，在此延續範例 src/com/looptek/AndroidOpenGLESPowenko/TutorialPart003Square.java 來做修改，把它複製之後，再做修改。

我們把 TutorialPart003Square class 定義一個新的 TutorialPart007Square class。

範例 B-12 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart007Square.java

```
1. public class TutorialPart007Square{
4.         float[] colors = {
5.                 1f, 0f, 0f, 1f, // vertex 0 red
6.                 0f, 1f, 0f, 1f, // vertex 1 green
7.                 0f, 0f, 1f, 1f, // vertex 2 blue
8.                 1f, 0f, 1f, 1f, // vertex 3 magenta
9.         };
10. …
11. }
```

這個程式的重點為：

- 意思是紅色 red = 1 → 255/255，綠色 green = 0 → 0/255，藍色 blue =0 → 0/255 和透明 =1 → 255/255，也就是不透明，或者可以說 0Xff0000ff。

然後為了把每一個點和顏色整合在一起，我們需要做以下的修改：

```
    public TutorialPart007Square() {
                …

                        // float has 4 bytes, colors (RGBA) * 4 bytes
            ByteBuffer cbb = ByteBuffer.allocateDirect(colors.length * 4);
            cbb.order(ByteOrder.nativeOrder());
            colorBuffer = cbb.asFloatBuffer();
            colorBuffer.put(colors);
            colorBuffer.position(0);
    }
```

這個程式的重點為：

- ByteBuffer 是設定顏色，並且放在記憶體中。
- ByteBuffer cbb = ByteBuffer.allocateDirect(colors.length * 4); 跟它每一個點的顏色，定義為 Float 4 bytes。
- colorBuffer.put(colors); 顏色放入。
- colorBuffer.position(0); 訂定第一個 0 顏色為開始顏色的位置。

還有別忘了把變數 colorBuffer 做個宣告的動作。

```
private FloatBuffer colorBuffer;
```

如果要使用顏色漸層的功能的話，就需要跟 openGL 的 color buffer 做個啟動的設定。這個程式的重點為：

- 意思是紅色 red = 1 → 255/255，綠色 green = 0 → 0/255，藍色 blue =0 → 0/255 透明 =1 → 255/255，也就是不透明，或者可以說 0Xff0000ff。

範例 B-13 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart007Square.java

```
public void draw(GL10 gl) {
                        gl.glFrontFace(GL10.GL_CCW);
        gl.glEnable(GL10.GL_CULL_FACE);
        gl.glCullFace(GL10.GL_BACK);
        gl.glVertexPointer(3, GL10.GL_FLOAT, 0, vertexBuffer);
        // 使在顯示過程中所使用的彩色陣列的緩衝區。
        gl.glEnableClientState(GL10.GL_COLOR_ARRAY); // 啟動顏色
        // 指定顏色內容的變數
        gl.glColorPointer(4, GL10.GL_FLOAT, 0, colorBuffer);
        gl.glDrawElements(GL10.GL_TRIANGLES, indices.length,
                    GL10.GL_UNSIGNED_SHORT, indexBuffer);
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
        gl.glVertexPointer(3, GL10.GL_FLOAT, 0, vertexBuffer);
        gl.glDrawElements(GL10.GL_TRIANGLES, indices.length,
                  GL10.GL_UNSIGNED_SHORT, indexBuffer);
        gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
        // 關閉點的記憶體，
        gl.glDisable(GL10.GL_CULL_FACE);
        gl.glDisableClientState(GL10.GL_COLOR_ARRAY);
    }
```

還有不要忘了把剛剛上一段的設定，單一顏色的拿掉，gl.glClearColor(0.0f, 0.0f, 0.0f, 0.5f); 如果你忘記的話，表現出來的是混合的顏色。

再把正方形稍稍的移動一下。

```
public void onDrawFrame(GL10 gl) {
...
        // Translate to end up under the flat square.
gl.glTranslatef(0, 0, -10);
// Draw our smooth
square.square.draw(gl);
}
```

程式範例：

Android 的 OpenGL 的範例程式，其路徑為：AndroidOpenGLESPowenko。

原始檔案：src/com/looptek/AndroidOpenGLESPowenko/TutorialPart007.java。

這是一個漸層顏色的正方形。

執行結果：

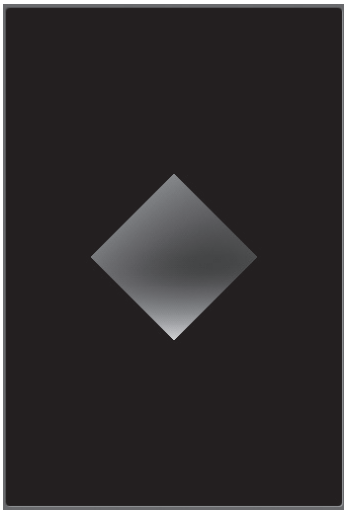


圖 B-39　範例程式的執行結果

執行影片：

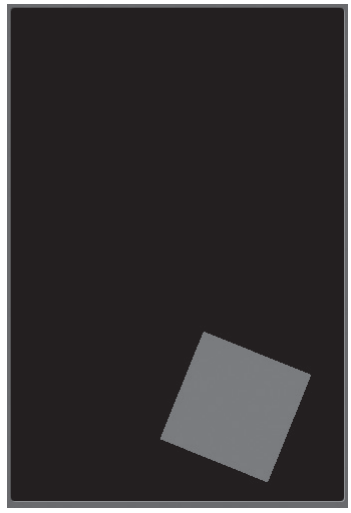
範例執行影片 B-7-colorsmooth.mov(http://youtu.be/3HuyEhvB3WI)。

程式範例

Android 的 OpenGL 的範例程式，其路徑為：AndroidOpenGLESPowenko。

原始檔案：src/com/looptek/AndroidOpenGLESPowenko/TutorialPart008.java。

你可以執行程式，用點選的方法移動正方形，並用按鍵改變正方形的顏色。



▲圖 B-40　範例程式的執行結果

範例 B-14 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart007Square.java

```
26.  @Override
27.     public boolean onTouchEvent(MotionEvent event)
28.     {
29.            int action = event.getAction();
30.            switch (action) {
31.               case (MotionEvent.ACTION_DOWN) : // Touch screen pressed
32.
                     break;
33.               case (MotionEvent.ACTION_UP) : // Touch screen touch ended
34.                  break;
35.               case (MotionEvent.ACTION_MOVE) :
                  // 當使用者移動手指的位置
36.                   float t_x=event.getX();
37.                   float t_y=event.getY();
38.                   float t_x1=((t_x/320)-0.5f)*10.0f;
39.                   float t_y1=-((t_y/480)-0.5f)*10.0f;
40.                   mTutorialPart008OpenRenderer.SetLocation(t_x1,t_y1,-10);
41.                  break;
42.               case (MotionEvent.ACTION_CANCEL) : // Touch event cancelled
43.                  break;
44.            }
45.            return super.onTouchEvent(event);
46.     }
47.     @Override
48.     public boolean onKeyDown(int keyCode,KeyEvent msg){
```

```
49.             switch(keyCode){
50.             case KeyEvent.KEYCODE_DPAD_CENTER:
51.                         mTutorialPart008OpenRenderer.SetColor(1,0,0,1);
52.                    return true;
53.             case KeyEvent.KEYCODE_DPAD_LEFT:
54.                      mTutorialPart008OpenRenderer.SetColor(0,1,0,1);
55.                    return true;
56.             case KeyEvent.KEYCODE_DPAD_RIGHT:
57.                      mTutorialPart008OpenRenderer.SetColor(0,0,1,1);
58.                      return true;
59.             case KeyEvent.KEYCODE_DPAD_UP:
60.                   mTutorialPart008OpenRenderer.SetColor(1,1,0,1);
61.                 return true;
62.             case KeyEvent.KEYCODE_DPAD_DOWN:
63.                      mTutorialPart008OpenRenderer.SetColor(0,1,1,1);
64.                   return true;
65.             }
66.
67.     return false;
68.     }
```

範例 B-15 sample\chB\AndroidOpenGLESPowenko\src\com\looptekAndroidOpenGLESPowenko\TutorialPart008OpenRenderer.java

```
36. private float angle;
37.          private float mx=0;
38.          private float my=0;
39.          private float mz=-10;
40.          private float mColorR=0.5f;
41.          private float mColorG=0.5f;
42.          private float mColorB=0.5f;
43.          private float mColorA=1.0f;
44.
45.          public void onDrawFrame(GL10 gl) {
46.
47.                   // 清除畫面
48.                   gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_
                      BUFFER_BIT);
49.                   // 位置重置
50.                   gl.glLoadIdentity();
51.                   // 移動
52.                   gl.glTranslatef(mx, my, mz);
56.                   gl.glRotatef(angle, 0, 0, 1);
57.                   // 設定顏色
58.                   gl.glColor4f(mColorR,mColorG,mColorB,mColorA);
59.
```

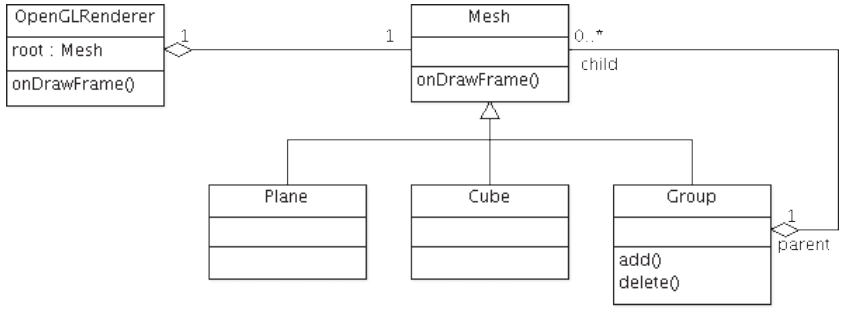
```
60.                     square.draw(gl);
61.
62.                     angle++;
63.             }
64.
65.             public void SetLocation(float i_x,float i_y,float i_z)
66.             {
67.                     mx=i_x;
68.                     my=i_y;
69.                     mz=i_z;
70.             }
71.             public void SetColor(float i_R,float i_G,float i_B,float i_A)
72.             {
73.                     mColorR=i_R;
74.                     mColorG=i_G;
75.                     mColorB=i_B;
76.                     mColorA=i_A;
77.             }
78.
79.
```

執行影片：

範例執行影片 B-80-UI.mov(http://youtu.be/jFBaS4w9wjI)。

B.9 OpenGL Es的程式範例八 – 網格

本章節會介紹 OpenGL framework 的複合模式 composite pattern 的設計。



▲圖 B-41　複合模式

首先建立一個 Mesh：

```
package com.looptek.AndroidOpenGLESPowenko.mesh;
public class Mesh {
 ...
}
```

在之前 TutorialPart004Square.java 例子中，增加新的作用，AndroidOpenGLES Powenko/mesh.java。

範例 B-16 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLES Powenko\Mesh\Mesh.java

```
public class Mesh {
    private FloatBuffer verticesBuffer = null;    // 點的記憶體
    private ShortBuffer indicesBuffer = null;     // 點的先後順序
    private int numOfIndices = -1;                // 目前第幾個點
    private float[] rgba = new float[] { 1.0f, 1.0f, 1.0f, 1.0f };  // 顏色
    private FloatBuffer colorBuffer = null;       // 顏色的記憶體

    public float x = 0;                           // 旋轉參數
    public float y = 0;
    public float z = 0;

    public float rx = 0;                          // Rotate params.
    public float ry = 0;
    public float rz = 0;
    public void draw(GL10 gl) {
        gl.glFrontFace(GL10.GL_CCW);              // 逆時鐘作畫
        gl.glEnable(GL10.GL_CULL_FACE);
        gl.glCullFace(GL10.GL_BACK);
        // 啟用頂點緩衝器寫入和過程中使用的渲染
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
        // 指定頂點的陣列的位置和數據格式，座標繪製時使用
        gl.glVertexPointer(3, GL10.GL_FLOAT, 0, verticesBuffer);
        gl.glColor4f(rgba[0], rgba[1], rgba[2], rgba[3]);  // Set flat color

        if (colorBuffer != null) {                          // 如果有設定漸層顏色
            gl.glEnableClientState(GL10.GL_COLOR_ARRAY) 指定設定顏色為 Array
            gl.glColorPointer(4, GL10.GL_FLOAT, 0, colorBuffer);
        }
        gl.glTranslatef(x, y, z);
        gl.glRotatef(rx, 1, 0, 0);
        gl.glRotatef(ry, 0, 1, 0);
        gl.glRotatef(rz, 0, 0, 1);

        // 點出了其中的顏色緩衝區
```

```
        gl.glDrawElements(GL10.GL_TRIANGLES, numOfIndices,
                GL10.GL_UNSIGNED_SHORT, indicesBuffer);
        // 禁用頂點緩衝區
        gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
        // 禁用面剔除
        gl.glDisable(GL10.GL_CULL_FACE);
    }
}
```

- gl.glColorPointer(4, GL10.GL_FLOAT, 0, colorBuffer); 把顏色資料放進去。設定 Mesh class 的 ByteBuffer 和顏色。

然後增加一些公共變數來設定 x,y,z 位置，和 rx,ry,rx 旋轉的角度。

範例 B-17 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\Mesh\Mesh.java

```
73. protected void setVertices(float[] vertices) {
74.          // a float is 4 bytes, therefore we multiply the number if
75.          // vertices with 4.
76.          ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
77.                   vbb.order(ByteOrder.nativeOrder());
78.                   verticesBuffer = vbb.asFloatBuffer();
79.                   verticesBuffer.put(vertices);
80.                   verticesBuffer.position(0);
81.      }
82.
83.      protected void setIndices(short[] indices) {
84.          // short is 2 bytes, therefore we multiply the number if
85.          // vertices with 2.
86.          ByteBuffer ibb = ByteBuffer.allocateDirect(indices.length * 2);
87.          ibb.order(ByteOrder.nativeOrder());
88.          indicesBuffer = ibb.asShortBuffer();
89.          indicesBuffer.put(indices);
90.          indicesBuffer.position(0);
91.          numOfIndices = indices.length;
92.      }
93.      public void setColor(float red, float green, float blue, float alpha) {
94.          // Setting the flat color.
95.          rgba[0] = red;
96.          rgba[1] = green;
97.          rgba[2] = blue;
98.          rgba[3] = alpha;
99.      }
100.     public void setColors(float[] colors) {
101.          // float has 4 bytes.
102.          ByteBuffer cbb = ByteBuffer.allocateDirect(colors.length * 4);
```

```
103.            cbb.order(ByteOrder.nativeOrder());
104.            colorBuffer = cbb.asFloatBuffer();
105.            colorBuffer.put(colors);
106.            colorBuffer.position(0);
107. }
```

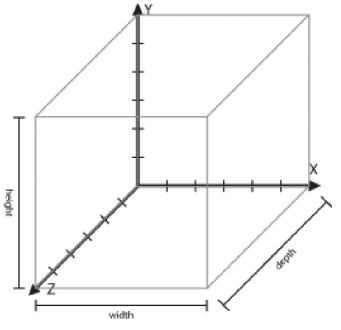
然後增加一些公共變數來設定 x,y,z 位置，和 rx,ry,rx 旋轉的角度。

```
    // 移動變數
    public float x = 0;
    public float y = 0;
    public float z = 0;
    // 旋轉
    public float rx = 0;
    public float ry = 0;
    public float rz = 0;
```

實際的函數，移動並且旋轉物體，可以看到我們 x,y,z, 分開處理和計算。

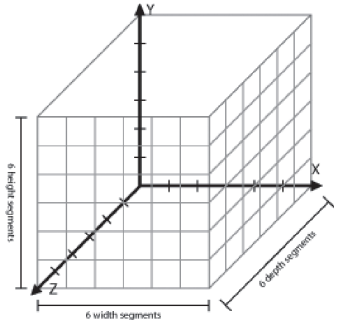
```
gl.glTranslatef(x, y, z);
gl.glRotatef(rx, 1, 0, 0);
gl.glRotatef(ry, 0, 1, 0);
gl.glRotatef(rz, 0, 0, 1);
```

B.9.1 網格 – 方塊 (Meshes)



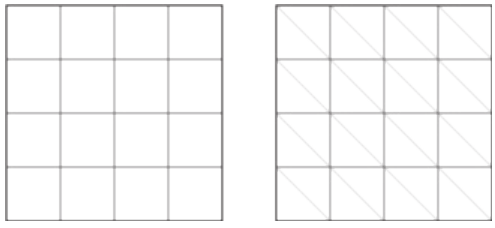
▲圖 B-42 方塊

在之前的章節範例做出來的平面，都是沒有分段，即左到右二個點處理完，但如果要把畫面畫得精緻，我們都會分成多段，將一定的長度中除以部分，如果你需要做表面的部分要有一些功能，就要分成段。所得到的東西可以作為一個遊戲的地平面，呈現出一個不錯的質感。



▲圖 B-43　分段

您可以看到把一個平面切成好幾個方形，我們之前學到，一個方塊需要切成二個三角形來畫。



▲圖 B-44　分成多個三角形

B.9.2　網格 (Meshes) – 平面 (Plane class)

我先設定平面開始 Plane Class：

```
public class Plane extends Mesh {} // 然後繼承 Mesh
```

我們新增一個 Plane.java，加上一段程式碼如下：

範例 B-18　sample\chBAndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\Mesh\Plane.java

```
1. package com.looptek.AndroidOpenGLESPowenko.mesh;
2. public class Plane extends Mesh {
3.     public Plane() {
4.         this(1, 1, 1, 1);
5.     }
6.     public Plane(float width, float height) {
7.         this(width, height, 1, 1);
8.     }
9.     public Plane(float width, float height, int widthSegments, int
    heightSegments) {
```

```
10.     float[] vertices = new float[(widthSegments + 1) * (heightSegments + 1)* 3];
11.     short[] indices = new short[(widthSegments + 1) * (heightSegments + 1)* 6];
12.
13.         float xOffset = width / -2;
14.         float yOffset = height / -2;
15.         float xWidth = width / (widthSegments);
16.         float yHeight = height / (heightSegments);
17.         int currentVertex = 0;
18.         int currentIndex = 0;
19.         short w = (short) (widthSegments + 1);
20.         for (int y = 0; y < heightSegments + 1; y++) {
21.             for (int x = 0; x < widthSegments + 1; x++) {
22.                 vertices[currentVertex] = xOffset + x * xWidth;
23.                 vertices[currentVertex + 1] = yOffset + y * yHeight;
24.                 vertices[currentVertex + 2] = 0;
25.                 currentVertex += 3;
26.
27.                 int n = y * (widthSegments + 1) + x;
28.
29.                 if (y < heightSegments && x < widthSegments) {
30.                     // 單一畫面
31.                     indices[currentIndex] = (short) n;
32.                     indices[currentIndex + 1] = (short) (n + 1);
33.                     indices[currentIndex + 2] = (short) (n + w);
34.                     // 第二個畫面
35.                     indices[currentIndex + 3] = (short) (n + 1);
36.                     indices[currentIndex + 4] = (short) (n + 1 + w);
37.                     indices[currentIndex + 5] = (short) (n + 1 + w - 1);
38.                     currentIndex += 6;
39.                 }
40.             }
41.         }
42.
43.         setIndices(indices);
44.         setVertices(vertices);
45.     }
46. }
```

這個程式的重點為：

- public Plane(.oat width, .oat height) {..} 這個是沒有段的平面。
- public Plane(.oat width, .oat height, int widthSegments, int heightSegments){} 這個是切多段的平面。

B.9.3 網格 (Meshes) – 正方形 (Cube Class)

```
public Cube(.oat width, .oat height, .oat depth) {..} // 然後繼承 Mesh
```

我們新增一個 Cube.java，然後加上一段程式碼如下：

範例 B-19 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\Mesh\Cube.java

```
1. package com.looptek.AndroidOpenGLESPowenko.mesh;
2.
3. public class Cube extends Mesh {
4.     public Cube(float width, float height, float depth) {
5.         width  /= 2;
6.         height /= 2;
7.         depth  /= 2;
8.         float vertices[] = { -width, -height, -depth, // 0
9.                               width, -height, -depth, // 1
10.                              width,  height, -depth, // 2
11.                             -width,  height, -depth, // 3
12.                             -width, -height,  depth, // 4
13.                              width, -height,  depth, // 5
14.                              width,  height,  depth, // 6
15.                             -width,  height,  depth, // 7
16.         };
17.         short indices[] = { 0, 4, 5,
18.                             0, 5, 1,
19.                             1, 5, 6,
20.                             1, 6, 2,
21.                             2, 6, 7,
22.                             2, 7, 3,
23.                             3, 7, 4,
24.                             3, 4, 0,
25.                             4, 7, 6,
26.                             4, 6, 5,
27.                             3, 0, 1,
28.                             3, 1, 2, };
29.
30.         setIndices(indices);
31.         setVertices(vertices);
32.     }
33. }
```

這個程式的重點為：

- 第 8-16 行：vertices 正方形的 8 個點。

- 第 17-28 行：indices 正方形，用 12 個三角形畫出來的點的編號。

整段 Cube.java 的程式如下：

```
public OpenGLRenderer() {
    …
    cube = new Cube(1, 1, 1);
    cube.rx = 45;
    cube.ry = 45;
…
}
public void onDrawFrame(GL10 gl) {
    ...
    cube.draw(gl);
}
```

B.9.4 網格 (Meshes) – Group

本章節將設計一個 Group 的 class，讓程式設計師可以把不同的 mesh 放在一起，這裡用的是 children 繼承的作法，你不用重新撰寫程式，只要修改局部即可。

範例 B-20 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\Mesh\Group.java

```
1. package com.looptek.AndroidOpenGLESPowenko.mesh;
2. import java.util.Vector;
3. import javax.microedition.khronos.opengles.GL10;
4.
5. public class Group extends Mesh {
6.     private Vector<Mesh> children = new Vector<Mesh>();
7.
8.     @Override
9.     public void draw(GL10 gl) {
10.         int size = children.size();
11.         for( int i = 0; i < size; i++)
12.             children.get(i).draw(gl);
13.     }
14.
15.     public void add(int location, Mesh object) {
16.         children.add(location, object);
17.     }
18.
19.     public boolean add(Mesh object) {
20.         return children.add(object);
21.     }
22.
```

```
23.     public void clear() {
24.         children.clear();
25.     }
26.
27.     public Mesh get(int location) {
28.         return children.get(location);
29.     }
30.
31.     public Mesh remove(int location) {
32.         return children.remove(location);
33.     }
34.
35.     public boolean remove(Object object) {
36.         return children.remove(object);
37.     }
38.
39.     public int size() {
40.         return children.size();
41.     }
42. }
```

使用上如下：

```
Group group = new Group();
Cube cube = new Cube(1, 1, 1);
cube.rx = 45;
cube.ry = 45;
group.add(cube);
root = group;
```

畫的方法：

```
public void onDrawFrame(GL10 gl) {
   ...
   // Draw our scene.
   root.draw(gl);
}
```

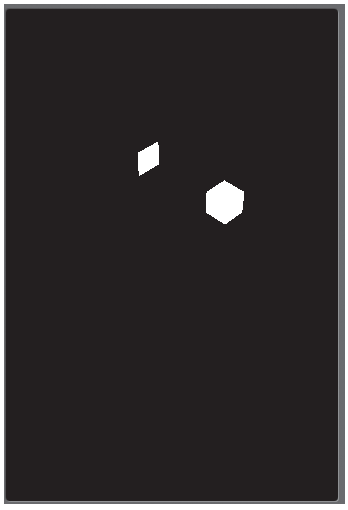


圖 B-45　範例程式的執行結果

執行影片：

範例執行影片 B-9-group.mov(http://youtu.be/w7Dqjgx8ohM)。

B.10 TextureView 貼材質

TextureView 是一種貼上圖片的方法，允許你來顯示圖片或影片，比如想要顯示圖片或一個開放式繪圖介面。雖然與 SurfaceView 相似，但 TextureView 可以顯示得更快，因為它是透過 opengl Es 和硬體處理，而不是創造另一個窗口，所以你可以使用 ViewPropertyAnimator，或調整其透明度 setAlpha()。注意，TextureView 是透過 openGL ES 的才會有作用。

範例 B-21　sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart011\Square.java

```
28. private float texCoords[] = {
29.
30.                 0.0f, 1.0f,
31.                 0.0f, 0.0f,
32.                 1.0f, 0.0f,
33.                 1.0f, 1.0f
34.         };
35.
36.     // 材質的位置先順序
37.     private short[] indices = { 0, 1, 2, 0, 2, 3 };
```

程式說明：

- 第 28-34 行：在使用貼圖之前，程式需要做決定把這一張圖分別對應位置，也就是左上、右上、左下、右下的位置，對應到真實的圖片，寬度的 0 到 1 之間，高度的 0 到 1 之間。

範例 B-22 sample\chB\AndroidOpenGLESPowenko\src\com\looptek\AndroidOpenGLESPowenko\TutorialPart011\Square.java

```
28.
29.     // 頂點緩衝。
30.     private FloatBuffer vertexBuffer;
31.
32.     // 先後順序緩衝
33.     private ShortBuffer indexBuffer;
34.
35.     public TutorialPart011Square(Context context) {
36.
37.         mContext=context;
38.         // 一個 float 是 4 bytes,
39.
40.
41.         ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
42.         vbb.order(ByteOrder.nativeOrder());
43.         vertexBuffer = vbb.asFloatBuffer();
44.         vertexBuffer.put(vertices);
45.         vertexBuffer.position(0);
46.
47.         // 一個 short 是 2 bytes
48.         ByteBuffer ibb = ByteBuffer.allocateDirect(indices.length * 2);
49.         ibb.order(ByteOrder.nativeOrder());
50.         indexBuffer = ibb.asShortBuffer();
51.         indexBuffer.put(indices);
52.         indexBuffer.position(0);
65.     }
66.
67.     public void onSurfaceCreated(GL10 gl, EGLConfig config) {
68.
69.         mCubeTexBuffer = util.createFloatBuffer(texCoords);
70.         loadBitmapTex(gl, R.drawable.icon);
71.
72.
73.
74.         gl.glEnable(GL10.GL_TEXTURE_2D); // 啟動材質貼圖
75.         gl.glDisable(GL10.GL_DITHER);
76.         gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,
```

```
77.                 GL10.GL_FASTEST);
78
79.     }
```

程式說明：

- 第 81,82 行：然後我們產生一個材質庫檔，並且指定 R.drawable.icon 這張圖片，讀取到 opengl 之中。
- 第 86 行：告知 opengl 打開材質貼圖。

```
80.
81.     private FloatBuffer mCubeTexBuffer;
82.     protected Context mContext;
83.     private int mTextureID[];
84.       private void loadBitmapTex(GL10 gl, int res){
85.             Bitmap bmp = util.getTextureFromBitmapResource(mContext, res);
86.
87.             mTextureID = new int[1];
88.             gl.glGenTextures(1, mTextureID, 0);
89.             gl.glBindTexture(GL10.GL_TEXTURE_2D, mTextureID[0]);
90.
91.             // 使用最近的表現。
92.             gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MIN_FILTER,
93.                                GL10.GL_NEAREST);
94.             gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_MAG_FILTER,
95.                                GL10.GL_NEAREST);
96.
97.
98.             gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_WRAP_S,
99.                                 GL10.GL_CLAMP_TO_EDGE);
100.          gl.glTexParameterf(GL10.GL_TEXTURE_2D, GL10.GL_TEXTURE_WRAP_T,
101.                             GL10.GL_CLAMP_TO_EDGE);
102.          gl.glTexEnvf(GL10.GL_TEXTURE_ENV, GL10.GL_TEXTURE_ENV_MODE,
103.                       GL10.GL_REPLACE);
104.          gl.glBindTexture(GL10.GL_TEXTURE_2D, mTextureID[0]);
105.          GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bmp, 0);
106.          bmp.recycle();
107.          return;
108.     }
```

程式說明：

- 第 97 行：讀取資料檔到 Bitmap bmp 中。
- 第 100 行：設定這一個 OpenGL ES 應用程式有多少個材質庫，這裡設定有一個。
- 第 101 行：指定 openGL ES 的材質庫檔 0，做以下的設定。

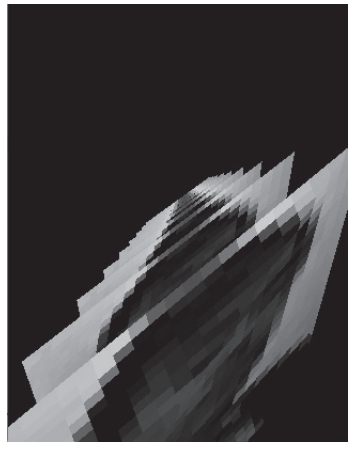
```
111.     public void draw(GL10 gl) {
112.
113.
114.         gl.glFrontFace(GL10.GL_CCW);
115.         gl.glEnable(GL10.GL_CULL_FACE);
116.         gl.glCullFace(GL10.GL_BACK);
117.
118.         gl.glVertexPointer(3, GL10.GL_FLOAT, 0, vertexBuffer);
119.         gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
120.         gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, mCubeTexBuffer);
121.
122.
123.         gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
124.         gl.glVertexPointer(3, GL10.GL_FLOAT, 0, vertexBuffer);
125.
126.         gl.glDrawElements(GL10.GL_TRIANGLES, indices.length,
127.         GL10.GL_UNSIGNED_SHORT, indexBuffer);128.
129.         gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
130.         gl.glDisable(GL10.GL_CULL_FACE);
131.
132.     }
```

程式說明：

- 把材質貼到指定的物體上面。

執行結果：

請加上 21\Android6_cookbook_ch21.odt 的內容。



▲圖 B-46　貼材質的範例執行結果

執行影片：

範例執行影片 B-10-texture.mov(http://youtu. be/3GLTrkSr7as)。