Appendix

A

2D 圖形影像處理

A.1 繪圖基礎onDraw

SDK 版本：API level 1、Android 1.0

本章節會介紹繪圖。在本節之中，我們會用 onDraw 的範例，透過二段的程式讓你了解繪圖的技巧，第一段的程式是透過 onDraw 方式，畫出一個圓形。第二段可以讓使用者畫畫。

Android 的 Gallery UI 的範例程式路徑為：TutorialOndraw。

onDraw 元件的功能說明

FrameLayout 主要是讓程式設計人員可以在這一個區域之中，畫出特定圖像。以下的範例，我們會在這一個 FrameLayout 中畫出一個圓形。然後在第二段的程式可以讓用戶自行切換新增和移動圓形，這樣就可以表現出畫畫的效果。

新增一個 Android 應用程式，並打開範例程式中的 main.xml，全部替換成下面的 XML 來增加一個新的 FrameLayout UI，新增一段 XML 之後，所表現出來的情況如右圖所示。

圖 A-1　main.xml 所定義出來的結果

範例 A-1　sample\chA\TutorialOndraw\res\layout\main.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <FrameLayout
3.    xmlns:android="http://schemas.android.com/apk/res/android"
4.    android:id="@+id/main_view"
5.    android:layout_width="fill_parent"
6.    android:layout_height="fill_parent"
7.    android:background="#FF66FF33" />
```

這個畫面設定文件 XML 中的重點為：

- < FrameLayout >…</ FrameLayout > 加入一個 FrameLayout 的 UI 元素。
- android:id="@+id/ main_view" 定義 FrameLayout 控制項控制碼為 main_view。
- android:background="#FF66FF33" 背景的顏色。

onDraw UI 元件的程式範例

您可以看到我們透過 onDraw 的控制項，像是畫布一樣，讓程式設計人員畫出想要的圖像。Android 原始檔案 TutorialGallery.java 的程式碼片段如下所示：

範例 A-2 sample\chA\TutorialOndraw\src\com\example\TutorialOndraw\TutorialGallery.java

```
3. public class TutorialOndraw extends Activity {
4.         @Override
5.         public void onCreate(Bundle savedInstanceState) {
6.             super.onCreate(savedInstanceState);
7.             setContentView(R.layout.main);
8.            FrameLayout main=(FrameLayout) findViewById(R.id.main_view);
9.             main.addView(new Ball(this,50,50,25));
10.         }
11. }
```

程式說明：

- main.addView 在 main 上面加上另一個，然後加上 view。
- new Ball(this,50,50,25); 新宣告一個 Ball class，這一個 class 等一下會說明。

範例 A-3 sample\chA\TutorialOndraw\src\com\example\TutorialOndraw\TutorialGallery.java

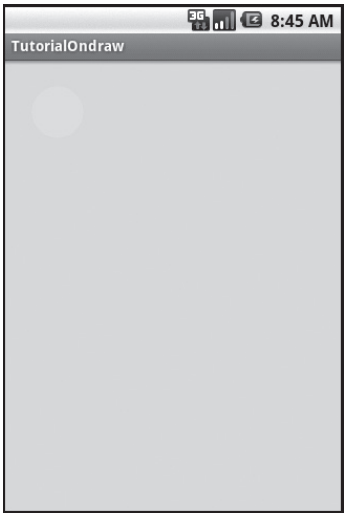
```
1. public class Ball extends View {
2.             private final float x;
3.          private final float y;
4.             rivate final int r;
5.          private final Paint mPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
6.             public Ball(Context context, float x, float y, int r) {
7.                 super(context);
8.                 mPaint.setColor(0xFF00ffff);
9.                 this.x = x;
10.                 this.y = y;
11.                 this.r = r;
12.             }
13.
14.             @Override
15.             protected void onDraw(Canvas canvas) {
16.                 super.onDraw(canvas);
17.                 canvas.drawCircle(x, y, r, mPaint);
18.             }
19. }
```

程式說明：

- public class Ball extends View：設計一個新的 class 名為 Ball 是繼承 View class 下來。

- private .nal .oat x; 私有變數 .oat x r，是用設定畫圓的 x 位置。
- private .nal .oat y; 私有變數 .oat x r，是用設定畫圓的 y 位置。
- private .nal int r; 私有變數 int r，是用設定畫圓的直徑。
- private final Paint mPaint. 變數 Paint mPaint 是設定 Paint(Paint.ANTI_ALIAS_ FLAG);。
- mPaint.setColor(0xFF00ffff); 設定要畫是什麼顏色。
- this.x = x; 把函數進來的參數，放在私有變數 .oat x。
- this.y = y; 把函數進來的參數，放在私有變數 .oat x。
- this.r = r; 把函數進來的參數，放在私有變數 int r r，設定畫圓的直徑。
- protected void onDraw(Canvas canvas){.} 主要的函數，當 Android 的軟體要重新畫 FrameLayout 就會使用這一個函數。
- super.onDraw(canvas); 使用被繼承的 View 中的 onDraw。
- canvas.drawCircle(x, y, r, mPaint); 畫圓，位置在 x,y，圓形的直徑大小在 r，然後畫在 mPaint 的畫布上。

執行結果：



▲圖 A-2　範例程式的執行結果

延伸學習

剛剛那一段程式已經可以畫出一個圓形，然後第二段的程式是讓用戶自行切換新增和移動圓形，這樣就可以表現出畫畫的效果，只要加上用戶觸控畫面的動作就可以了。所以新加上的程式，當用戶觸控畫面，就可以畫出一個新的圓形。

Android 原始檔案 TutorialOnDraw.java 的第二段程式碼片段如下所示：

範例 A-4 sample\chA\TutorialOndraw\src\com\example\TutorialOndraw\TutorialGallery.java

```
1. public void onCreate(Bundle savedInstanceState) {
2.             super.onCreate(savedInstanceState);
3.             setContentView(R.layout.main);
4.             FrameLayout main = (FrameLayout) findViewById(R.id.main_view);
5.             main.addView(new Ball(this,50,50,25));
6.
7.                 main.setOnTouchListener(new View.OnTouchListener() {
8.                      @Override
9.                     public boolean onTouch(View v, MotionEvent e) {
10.                         float x = e.getX();
11.                         float y = e.getY();
12.                        FrameLayout flView = (FrameLayout) v;
13.                         flView.addView(new Ball(getParent(), x,y,25));
14.                     return true;
15.                     }
16.                 });
17. }
```

程式說明：

- main.setOnTouchListener(new View.OnTouchListener() {.}); 加上 FrameLayout 監聽觸控的動作。
- public boolean onTouch(View v, MotionEvent e) {.} 當使用者觸控畫面，Android 就會使用這函數裡面的程式。
- .oat x = e.getX(); 得到使用者觸控畫面的 x 橫軸位置。
- .oat y = e.getY(); 得到使用者觸控畫面的 y 橫軸位置。
- FrameLayout .View = (FrameLayout) v; 把 View v 資料轉換成 FrameLayout .View
- .View.addView(.); 加上一個新的 View。
- new Ball(getParent(), x,y,25)); 新加一個 Ball class。
- getParent(); 用來取得 Context context。

所以當用戶觸控畫面的動作就會透過 main.setOnTouchListener 執行 onTouch() 這一段程式，然後新加一個圓形 .View.addView(new Ball(getParent(), x,y,25));。

範例程式的執行結果如下圖：

▲圖 A-3　範例程式的執行結果

A.2 drawable 各種繪圖的方法

SDK 版本：API level 1、Android 1.0

drawable 的範例會有二段的程式，第一段的程式是透過 drawable 方式，畫出七個圖形。

drawable 元件的範例程式：

Android 的 drawableUI 的範例程式其路徑為：TutorialDrawable。

drawable 元件的功能說明：

FrameLayout 主要是讓程式員可以在這一個區域之中，畫出特定圖像。等一下的範例，我們會在這一個 FrameLayout 中畫出七種圖形，分別是：

- 紅色正方形。
- 綠色的橢圓形。
- 藍色的橢圓形邊的正方形。
- 像是人工手畫出來的正方形。
- 有多種顏色的中空正方形。

- 有藍色條紋的菱形。
- 有開口的圓形。

然後了解整個繪圖程式之後，可以自行畫出不同的圖。

drawable UI 元件的畫面設定資源：

新增一個 Android 應用程式，並打開範例程式中的 main.xml，全部替換成下面的 XML，來增加一個新的 FrameLayout UI。

執行結果：

圖 A-4　main.xml 所定義出來的結果

A.2.1　drawable UI 元件的程式範例 – 畫出一個紅色正方形

在本章節可以看到我們透過 Canvas 的控制項，像是畫布一樣，讓程式開發者畫出想要的圖像，Android 原始檔案 TutorialGallery.java 的程式碼片段如下所示：

範例 A-5　sample\chA\TutorialDrawable\src\com\res\com\powenko\tutorialdrawable\TutorialGallery.java

```
2.   public class TutorialDrawable extends Activity {
3.         @Override
4.         public void onCreate(Bundle savedInstanceState) {
```

```
5.              super.onCreate(savedInstanceState);
6.              setContentView(R.layout.main);
7.              FrameLayout main = (FrameLayout) findViewById(R.id.main_view);
8.              main.addView(new SampleView(this));
9.          }
10. }
```

程式說明：

- main.addView　在 main 上面加上另一個 view。
- new SampleView(this); 新宣告一個 SampleView class，這一個類別待會特別的說明。現在要看重點是如何畫出一個紅色正方形。

範例 A-6 sample\chA\TutorialDrawable\src\com\res\com\powenko\tutorialdrawable\TutorialGallery.java

```
1.    private static class SampleView extends View {
2.         private ShapeDrawable[] mDrawables;
3.         public SampleView(Context context) {
4.             super(context);
5.             setFocusable(true);
6.             mDrawables = new ShapeDrawable[7];
7.             mDrawables[0] = new ShapeDrawable(new RectShape());
8.             mDrawables[0].getPaint().setColor(0xFFFF0000);
9.         }
10.
11.      @Override
12.      protected void onDraw(canvas){
13.             int x = 10;
14.             int y = 10;
15.             int width = 300;
16.             int height = 50;
17.
18.             for (Drawable dr : mDrawables) {
19.                 if (dr!=null)
20.                 {
21.                     dr.setBounds(x, y, x + width, y + height);
22.                     dr.draw(canvas);
23.                 }
24.                 y += height + 5;
25.             }
26. }
```

程式說明：

- public class Ball extends View 設計一個新的 class 名為 Ball 是繼承 View class 下來。
- private .nal .oat x; 私有變數 .oat x r，是用設定畫圓的 x 位置。

- mDrawables[0] = new ShapeDrawable(new RectShape()); 新增一個畫正方形的 class。這樣就畫出一個正方形了。
- mDrawables[0].getPaint().setColor(0xFFFF0000); 設定顏色為紅色。
- protected void drawable(Canvas canvas){.} 主要的函數，當 Android 的軟體要重新畫 FrameLayout 就會使用這一個函數。
- int x = 10; int y = 10; int width = 300; int height = 50; 設定私有變數，等一下可以用到。
- for (Drawable dr : mDrawables) {. } 將 mDrawables[x] 矩陣資料一個一個放在 rawable dr 中。
- dr.setBounds (x, y, x + width, y + height); 畫圓，位置在 x,y，將圖畫在大小為 x + width 的寬度和 y +height 的高度上面。

執行結果：

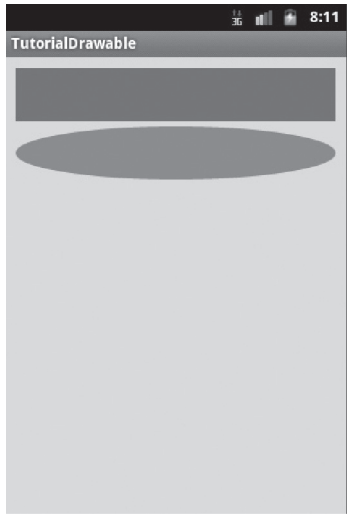


圖 A-5　範例執行結果

A.2.2　drawable UI 元件的程式範例 – 畫出一個綠色的橢圓形

Android 原始檔案 TutorialGallery.java 的程式碼片段如下所示，現在要看重點是如何畫出多種不同的圖形。

範例 A-7 sample\chA\TutorialDrawable\src\com\res\ com\powenko\tutorialdrawable\ TutorialGallery.java

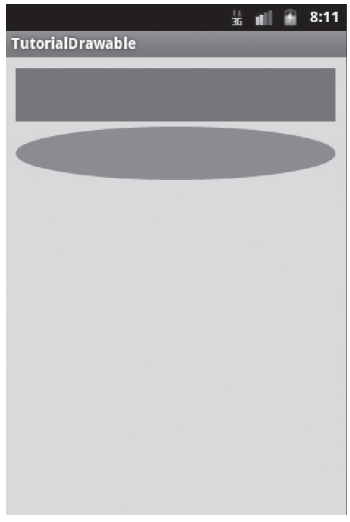
```
49.    private static class SampleView extends View {
50.         private ShapeDrawable[] mDrawables;
51.         public SampleView(Context context) {
52.             super(context);
53.             setFocusable(true);
54.             mDrawables = new ShapeDrawable[7];
55.             mDrawables[0] = new ShapeDrawable(new RectShape());
56.             mDrawables[0].getPaint().setColor(0xFFFF0000);
57.               mDrawables[1] = new ShapeDrawable(new OvalShape());
58.               mDrawables[1].getPaint().setColor(0xFF33aa00);
59.         }
60.      @Override protected void onDraw(Canvas canvas){
61.             int x = 10;
62.             int y = 10;
63.             int width = 300;
64.             int height = 50;
65.             for (Drawable dr : mDrawables) {
66.                 if (dr!=null){
67.                     dr.setBounds(x, y, x + width, y + height);
68.                    dr.draw(canvas);
69.                 }                          畫出指定的效果
70.                 y += height + 5;
74.             }
75. }
```

程式說明：

- public class Ball extends View 設計一個新的 class 名字 Ball 是繼承 View class 下來。
- private .nal .oat x; 私有變數 .oat x r，是用設定畫圓的 x 位置。
- mDrawables[0] = new ShapeDrawable(new RectShape()); 新增一個畫正方形的 class。這樣就畫出一個正方形了。
- mDrawables[0].getPaint().setColor(0xFFFF0000); 設定顏色為紅色。
- mDrawables[1] = new ShapeDrawable(new OvalShape()); 新增一個畫正方形的 class。這樣就畫出一個圓形。
- mDrawables[1].getPaint().setColor(0xFF00FF00); 設定顏色為綠色。
- protected void drawable(Canvas canvas){.} 主要的函數，當 Android 的軟體要重新畫 FrameLayout 就會使用這一個函數。
- int x = 10; int y = 10; int width = 300; int height = 50; 設定私有變數，等一下可以用到。

- for (Drawable dr : mDrawables) {. } 將 mDrawables[x] 矩陣資料一個一個放在 rawable dr 中。
- dr.setBounds (x, y, x + width, y + height); 畫圓，位置在 x,y，將圖畫在大小為 x + width 的寬度和 y +height 的高度上面。

執行結果：



▲圖 A-6　範例執行結果

A.2.3　drawable UI 元件的程式範例 – 畫出一個藍色的橢圓形邊的正方形

延續上一章節，再加一段新的程式，畫出一個藍色的橢圓形邊的正方形，Android 原始檔案 TutorialGallery.java 的程式碼片段如下所示：

範例 A-8　sample\chA\TutorialDrawable\src\com\res\com\powenko\tutorialdrawable\TutorialGallery.java

```
76.             float[] outerR = new float[] { 12, 12, 12, 12, 0, 0, 0, 0 };
77.             mDrawables[2] = new ShapeDrawable(new RoundRectShape(outerR,
                null,null));
78.             mDrawables[2].getPaint().setColor(0xFF0000FF);
```

程式說明：

- mDrawables[2] = new ShapeDrawable(new RoundRectShape(outerR, null,null)); 設定畫出圓弧狀的正方形。

- mDrawables[2].getPaint().setColor(0xFF0000FF); 設定為藍色。

執行結果：

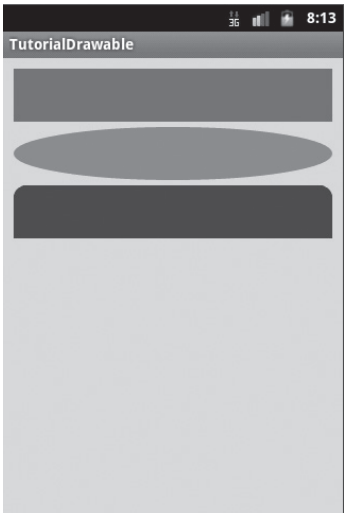


圖 A-7　範例執行結果

A.2.4　drawable UI 元件的程式範例 – 畫出一個多種顏色的橢圓形邊並中空的正方形

延續上一章節，再加一段新的程式，畫出一個藍色的橢圓形邊的正方形。

範例 A-9　sample\chA\TutorialDrawable\src\com\res\com\powenko\tutorialdrawable\TutorialGallery.java

```
49. private static class SampleView extends View {
50.         private ShapeDrawable[] mDrawables;
51.
52.     private static Shader makeSweep() {
53.             return new SweepGradient(150, 25,
54.                new int[] { 0xFFFF0000, 0xFF00FF00, 0xFF0000FF, 0xFFFF0000 },
55.                 null);
56.           }
57.
58.         public SampleView(Context context) {
59.             super(context);
60.             setFocusable(true);
61.             mDrawables = new ShapeDrawable[7];
62.             mDrawables[0] = new ShapeDrawable(new RectShape());
63.             mDrawables[0].getPaint().setColor(0xFFFF0000);
```

多種顏色的設定

```
64.
65.                mDrawables[1] = new ShapeDrawable(new OvalShape());
66.                mDrawables[1].getPaint().setColor(0xFF00FF00);
67.
68.              float[] outerR = new float[] { 12, 12, 12, 12, 0, 0, 0, 0 };
69.              mDrawables[2] = new ShapeDrawable(new RoundRectShape(outerR,
                 null,null));
70.              mDrawables[2].getPaint().setColor(0xFF0000FF);
71.
72.
73.              RectF   inset = new RectF(6, 6, 6, 6);
74.              mDrawables[3] = new ShapeDrawable(new RoundRectShape(outerR,
                 inset,null));
75.              mDrawables[3].getPaint().setShader(makeSweep());
76.              PathEffect pe = new DiscretePathEffect(10, 4);
77.              PathEffect pe2 = new CornerPathEffect(4);
78.              mDrawables[3].getPaint().setPathEffect(new ComposePathEffect
                 (pe2, pe))
```

程式說明：

- mDrawables[3] = new ShapeDrawable(new RoundRectShape(outerR, inset,null)); 在 這裡 outerR 畫出的周邊，而 inset 是定義裡面挖空的圖形大小，你可以用現在這一個 Rect[4] 的方法，或者是 .oat[8] 都一樣。如果設定為 null 的話，就不會畫出內部的弧度了。
- mDrawables[3].getPaint().setShader(makeSweep()); 設定顏色的部分，我們換另外一個方法，用 Shader（著色者）使用 makeSweep() 函數來設定不同的顏色。
- PathEffect pe = new DiscretePathEffect(10, 4); 這是線條彎彎曲曲的函數，運用變數的方式，把要畫出來的線弄成彎彎曲曲的樣子。
- PathEffect pe2 = new CornerPathEffect (4); 角度的變換方式，可以設定並取代之間的線段到指定的半徑圓角中的任何尖角角度。
- mDrawables[3].getPaint().setPathEffect(new ComposePathEffect(pe2, pe)); 畫上並且運用剛剛所設定的效果。
- private static Shader makeSweep() {...} 剛剛用在 setShader 函數，設定漸層顏色。
- SweepGradient(150, 25, new int[] { 0xFFFF0000, 0xFF00FF00, 0xFF0000FF, 0xFFFF0000 }, null); 意思是 (.oat cx, .oat cy, int[] colors, .oat[] positions) 要填的顏色，設定顏色在 cx, cy 的地方，然後真正的顏色 RGB 為 new int[] { 0xFFFF0000 (紅色), 0xFF00FF00（綠色）, 0xFF0000FF（藍色）, 0xFFFF0000（紅色）}。

執行結果：

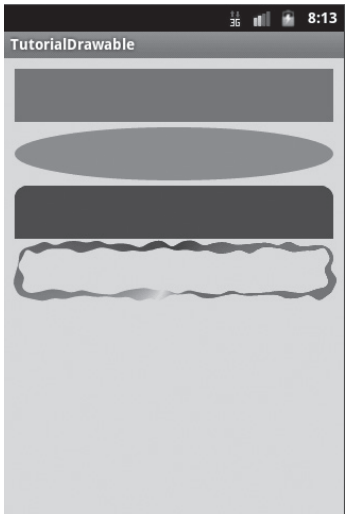


圖 A-8　範例執行結果

A.2.5　drawable UI 元件的程式範例 – 畫出一個多種顏色的橢圓形邊並中空的正方形

本節會用 drawable 的元件，畫出想要的圖像，我們在加一段新的程式，畫出一個多種顏色的橢圓形邊並中空的正方形。

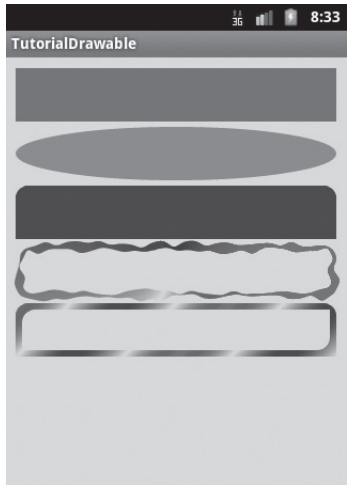
範例 A-10　sample\chA\TutorialDrawable\src\com\res\com\powenko\tutorialdrawable\TutorialGallery.java

```
99.  float[] innerR = new float[] { 12, 12, 0, 0, 12, 12, 0, 0 };
100. mDrawables[4] = new ShapeDrawable(new RoundRectShape(outerR, inset,
     innerR));
101. mDrawables[4].getPaint().setShader(makeLinear());
       ...
128.     private static Shader makeLinear() {
129.               return new LinearGradient(0, 0, 50, 50,
130.                         new int[] { 0xFFFF0000, 0xFF00FF00, 0xFF0000FF },
131.                         null, Shader.TileMode.MIRROR);
132.               }
```

程式說明：

- public class Ball extends View 設計一個新的 class 名為 Ball 是繼承 View class 下來。

執行結果：



▲圖 A-9　範例執行結果

A.2.6　drawable UI 元件的程式範例 – 畫出一個藍色條紋的菱形

本章節可以看到透過 drawable 的控制項，像是畫布一樣，讓程式設計人員畫出想要的圖像，我們在加一段新的程式，畫出一個藍色條紋的菱形。

範例 A-11　sample\chA\TutorialDrawable\src\com\res\com\powenko\tutorialdrawable\TutorialGallery.java

```
28. private static Shader makeTiling() {
29.         int[] pixels = new int[] { 0xFFFF0000, 0xFF00FF00
                                     , 0xFF0000FF, 0};
30.         Bitmap bm = Bitmap.createBitmap(pixels, 2, 2,
31.         Bitmap.Config.RAGB_8888);
32.         return new BitmapShader(bm, Shader.TileMode.REPEAT,
33.         Shader.TileMode.REPEAT);
34. }
     …中間省略
49.    public SampleView(Context context){ ...
50.             Path path = new Path();
51.             path.moveTo(50, 0);
52.             path.lineTo(0, 50);
53.             path.lineTo(50, 100);
54.             path.lineTo(100, 50);
55.             path.close();
56. mDrawables[5] = new ShapeDrawable(new PathShape(path, 100, 100));
```

```
57. mDrawables[5].getPaint().setShader(makeTiling());
    …中間省略
138.  @Override protected void onDraw(Canvas canvas) {
139.             int x = 10;
140.             int y = 10;
141.             int width = 300;
142.             int height = 50;
143.
144.             for (Drawable dr : mDrawables) {
145.                      if (dr!=null){
146.                               dr.setBounds(x, y, x + width, y + height);
147.                               dr.draw(canvas);
148.                      }
149.                 y += height + 5;
150.             }
151. }
```

程式說明：

- private static Shader makeTiling() {..} 自訂函數 makeTiling，和 makeSweep 非常的類似。setShader 函數，用在設定漸層顏色。
- int[] pixels = new int[] { 0xFFFF0000, 0xFF00FF00, 0xFF0000FF, 0};，意思是 createBitmap 要畫的顏色。
- Bitmap bm = Bitmap.createBitmap(pixels, 2, 2, Bitmap.Config.ARGB_8888); 定義一個新的 Bitmap createBitmap(int[] colors, int width, int height, Bitmap.Config config) createBitmap（顏色，寬度，高度，這一個 Bitmap 的 Con.g 設定）;。
- return new BitmapShader(bm, Shader.TileMode.REPEAT,Shader.TileMode.REPEAT); 調用此 BitmapShader 並著色繪製圖。BitmapShader(Bitmap bitmap, Shader.TileMode tileX, Shader.TileMode tileY) TileMode 為重複，tileX 和 tileY 也都是重複畫出 bm 的圖。
- Path path = new Path(); 設定要畫出來的路徑動線。
- path.moveTo(50, 0); 首先，先從 x=50 y=0 的位置開始。
- path.lineTo(0, 50); 畫一條線從 x=50 y=0 的位置開始，到 x=0 y=50。
- path.lineTo(50, 100); 畫一條線從 x=0 y=50 的位置開始，到 x=50 y=100。
- path.lineTo(100, 50); 畫一條線從 x=50 y=100 的位置開始，到 x=100 y=50。
- path.close(); 結束要畫的路徑。
- mDrawables[5] = new ShapeDrawable(new PathShape(path, 100, 100)); 真正的把剛剛的路徑圖畫出來。

- mDrawables[5].getPaint().setShader(makeTiling()); 把顏色依照 makeTiling() 私有函數的方法填入顏色。
- protected void drawable(Canvas canvas){.} 主要的函數，當 Android 的軟體要重新畫 FrameLayout 就會使用這一個函數。
- int x = 10; int y = 10; int width = 300; int height = 50; 設定私有變數，等一下可以用到。
- for (Drawable dr : mDrawables) {. } 將 mDrawables[x] 矩陣資料一個一個放在 rawable dr 中。
- dr.setBounds (x, y, x + width, y + height); 畫圓，位置在 x,y，將圖畫在大小為 x + width 的寬度和 y +height 的高度上面。

執行結果：

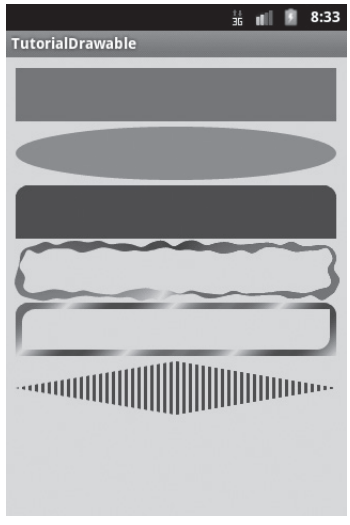


圖 A-10　範例執行結果

A.2.7　drawable UI 元件的程式範例 – 有缺口的圓形

本章節透過 drawable 元件，畫出一個黃色的圓形缺了一個角。

範例 A-12　sample\chA\TutorialDrawable\src\com\res\com\powenko\tutorialdrawable\TutorialGallery.java

```
119.  mDrawables[6] = new MyShapeDrawable(new ArcShape(45, -270));
120.  mDrawables[6].getPaint().setColor(0x88FF8844);
123.  MyShapeDrawable msd = (MyShapeDrawable)mDrawables[6];
124.  msd.getStrokePaint().setStrokeWidth(4);
      ...
```

```
138.  @Override protected void onDraw(Canvas canvas) {
139.             int x = 10;
140.             int y = 10;
141.             int width = 300;
142.             int height = 50;
143.
144.             for (Drawable dr : mDrawables) {
145.                      if (dr!=null){
146.                               dr.setBounds(x, y, x + width, y + height);
147.                               dr.draw(canvas);
148.                      }
149.                 y += height + 5;
150.             }
151. }
     …
157. private static class MyShapeDrawable extends ShapeDrawable {
158. private Paint mStrokePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
159.
160.             public MyShapeDrawable(Shape s) {
161.                 super(s);
162.                 mStrokePaint.setStyle(Paint.Style.STROKE);
163.             }
164.             public Paint getStrokePaint() {
165.                 return mStrokePaint;
166.             }
167.             Override protected void onDraw(Shape s, Canvas c, Paint p) {
168.                 s.draw(c, p);
169.                 s.draw(c, mStrokePaint);170.             }
171.         }
172. public SampleView(Context context) {
```

程式說明：

- mDrawables[6] = new MyShapeDrawable(new ArcShape(45, -270)); 我們從角度 45 開始，用逆時鐘方向畫出 270 度。
- mDrawables[6].getPaint().setColor(0x88FF8844); 設定顏色為黃色。
- MyShapeDrawable msd = (MyShapeDrawable) mDrawables[6]; 把 mDrawables[6]; 的資料型態轉換成 MyShapeDrawable class。
- msd.getStrokePaint().setStrokeWidth(4); 然後透過 setStrokeWidth 加上一個新的功能，邊寬有 4 個點數。
- private static class MyShapeDrawable extends ShapeDrawable {..} 設定一個新的 class 然後繼承 ShapeDrawable。

- private Paint mStrokePaint = new Paint(Paint.ANTI_ALIAS_FLAG); 私有變數。
- public MyShapeDrawable(Shape s) {..} 初始函數。
- super(s);
- mStrokePaint.setStyle(Paint.Style.STROKE); 設定畫筆風格為 STROKE 有邊框的風格。
- public Paint getStrokePaint() { return mStrokePaint; } 取得畫筆的風格。
- @Override protected void onDraw(Shape s, Canvas c, Paint p) {
- s.draw(c, p); s.draw(c, mStrokePaint); } 實際畫出來的函數。

執行結果：

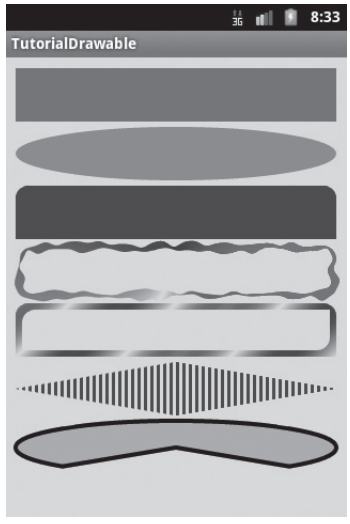


圖 A-11　範例執行結果

執行影片：

範例執行影片 A-2-drawable.mov(http://youtu.be/gG5LysB5Oqw)。

A.3 drawableImageView UI元件的程式範例 – 繪畫板

SDK 版本：API level 1、Android 1.0

在本章節透過 drawable 的控制項和 onTouchEvent，把 Android 當成畫布一樣，讓用戶畫出想要的圖畫。

drawableImageView 的畫面設定資源：

新增一個 Android 應用程式，並打開範例程式中的 main.xml，全部替換成下面的 XML，來增加一個新的 FrameLayout UI。

介面定義：

▲圖 A-12　main.xml 所定義出來的結果

drawableImageView 的程式範例 - 繪畫板：

範例 A-13 sample\chA\TutiroalDrawableImageView\src\com.example.TutiroalDrawableImageView\TutiroalDrawableImageView.java)

```
1. package com.example.TutiroalDrawableImageView;
2. …
3. public class TutiroalDrawableImageView extends Activity {
4.
5.     @Override
6.     public void onCreate(Bundle savedInstanceState) {
7.         LinearLayout mLinearLayout;
```

```
8.          super.onCreate(savedInstanceState);
9.          mLinearLayout = new LinearLayout(this);
10.         Bitmap bitmapOrg = BitmapFactory.decodeResource(getResources(),
            R.drawable.green);
11.         DrawableImageView mDrawableImageView=new DrawableImageView(this,
            bitmapOrg);
12.         mLinearLayout.addView(mDrawableImageView);
13.         setContentView(mLinearLayout);
14.     }
15. }
```

程式說明：

- LinearLayout mLinearLayout; 宣告一個 LinearLayout。
- mLinearLayout = new LinearLayout(this); 然後指派給自己。
- Bitmap bitmapOrg = BitmapFactory.decodeResource(getResources(), R.drawable.green); 畫出一個綠色的背景。
- DrawableImageView mDrawableImageView=new DrawableImageView(this,bitmapOrg); 宣告一個我們寫的 DrawableImageView class
- mLinearLayout.addView(mDrawableImageView); 把它加到 LinearLayout 畫面上。
- setContentView(mLinearLayout); 軟體的畫面設定為剛剛新增的 LinearLayout。

剛剛那一個程式的意思是把軟體的畫面指派給我們自己寫的 DrawableImageView class，而這一個 class 是如何作用的，首先我們先撰寫有關於用戶觸控的動作，請看 Android 原始檔案 DrawableImageView.java 的部分程式碼片段如下所示：

範例 A-14 sample\chA\TutiroalDrawableImageView\src\com.example.TutiroalDrawableImageView\TutiroalDrawableImageView.java

```
1.
2.     float lastX;
3.     float lastY;  // 記錄上回的位置。
4.     @Override public boolean onTouchEvent(MotionEvent event) { // 觸控的動作
5.         mPaint.setStrokeWidth(width/scale); // 設定 mPaint 筆的大小
6.
7.         float curX =  event.getX()/scale;    // 拿到用戶 X 點的位置，並轉換 scale
8.         float curY =  event.getY()/scale;
9.   switch (event.getAction()){               // 使戶觸控的動作方式
10.    case MotionEvent.ACTION_DOWN:{          // 按下去
11.             mCanvas.drawCircle(curX, curY,width/2/scale, mPaint);
12.             break;
13.    }
14.    case MotionEvent.ACTION_MOVE:{          // 移動
```

```
15.     mCanvas.drawLine(lastX, lastY, curX, curY, mPaint);
16.             mCanvas.drawCircle(curX, curY,width/2/scale, mPaint);
17.             break;
18.    }
19.    case MotionEvent.ACTION_CANCEL:                                    //取消
20.    case MotionEvent.ACTION_UP:{
21.     mCanvas.drawLine(lastX, lastY, curX, curY, mPaint);               //畫線
22.             mCanvas.drawCircle(curX, curY,width/2/scale, mPaint); //畫圓
23.             break;
24.    }
25.
26.   }
27.   lastX = curX;
28.   lastY = curY;
29.         invalidate();  //強制畫面更新
30.
31.      return true;
32.     }
```

範例 A-15 sample\chA\TutiroalDrawableImageView\src\com.example.TutiroalDrawableImageView\DrawableImageView.java

```
12. public class DrawableImageView extends View {  //我們設計的類別
13.     private Bitmap mBitmap;
14.     private Bitmap pic;
15.     private Canvas mCanvas;
16.     private final Paint mPaint;
17.     private int a = 255;
18.     private int r = 255;
19.     private int g = 255;
20.     private int b = 255;
21.     private float width = 4;
22.
23.     public DrawableImageView(Context c, Bitmap img) {  // 初始化
24.         super(c);
25.         pic = img;
26.         mPaint = new Paint();
27.         mPaint.setAntiAlias(true);
28.        mPaint.setARGB(a,r,g,b); // 設定畫筆顏色
29.                                          建立 Bitmap 畫板
30.
31.         Bitmap newBitmap = Bitmap.createBitmap(img.getWidth(), img.
            getHeight(), Bitmap.Config.RGB_565);  //設定和畫出背景圖畫
32.         Canvas newCanvas = new Canvas();
33.         newCanvas.setBitmap(newBitmap);
34.         if (img != null) {
```

```
35.         newCanvas.drawBitmap(img, 0, 0, null);
36.         }
37.         mBitmap = newBitmap;
38.         mCanvas = newCanvas;
39.
40.          mCanvas.setBitmap(mBitmap);
41.     }
42.
43.     public DrawableImageView(Context c, Bitmap img, int alpha, int red,
        int green, int blue) {
44.           this(c, img);
45.           setColor(alpha, red, green, blue);
46.     }
47.     public DrawableImageView(Context c, Bitmap img, int alpha, int red,
        int green, int blue, float w) {
48.      this(c, img, alpha, red, green, blue);
49.      width = w;
50.     }
51.
52.     public Bitmap getBitmap() {return mBitmap;}
53.     public void setWidth(float w) {width = w;}
54.     public void setColor(int alpha, int red, int green, int blue) {
55.      a = alpha;
56.      r = red;
57.      g = green;
58.      b = blue;
59.         mPaint.setARGB(a,r,g,b);
60.     }
61.     public void Undo() {
62.   mCanvas.drawBitmap(pic, 0, 0, null);
63.   invalidate();
64.  }
65.
66.     float scaleX;
67.     float scaleY;
68.     float scale;
69.     @Override protected void onSizeChanged(int w, int h, int oldw, int oldh) {
70.      scaleX = (float) w/mBitmap.getWidth();
71.      scaleY = (float) h/mBitmap.getHeight();
72.      scale = scaleX > scaleY ? scaleY : scaleX;
73.     }
74.
75.     @Override protected void onDraw(Canvas canvas) {  // Android 畫面的函數
76.         if (mBitmap != null) {
77.             Matrix matrix = new Matrix();
```

```
78.          matrix.postScale(scale, scale);
79.          canvas.drawBitmap(mBitmap, matrix, null);
80.             //canvas.drawBitmap(mBitmap, 0,0, null);
81.         }
```

執行結果：

▲圖 A-13　範例執行結果

執行影片：

範例執行影片 A-3-TutiroalDrawableImageView.mov(http://youtu.be/HhixFeHfto8)。

A.4 Image Process元件 – 改變圖片顏色

SDK 版本：API level 1、Android 1.0

您可以看到我們透過 Bitmap 的方法，把圖像讀取進來後，等待用戶按下鍵後，產生 onTouchEvent 的情況，再用數學計算處理，把圖像顯像在畫面上。

Android 的 drawableUI 的範例程式其路徑為：TutorialImageProcess。

其中各個文件目錄為：

- AndroidManifest.xml：各個應用描述文件。
- asset/：資產文件。
- res/：資源檔案目錄。

- src/com/example/ TutorialDrawable/src/：原始檔案目錄。

範例程式：

- 原始檔案：src/com/example/ TutorialDrawable/src/ TutorialImageProcess.java。
- 畫面設定資源程式碼：res/layout/main.xml。

新增一個 Android 應用程式，並打開範例程式中的 main.xml，全部替換成下面的 XML，來增加一個新的 FrameLayout UI。

介面定義：

▲ 圖 A-14　main.xml 所定義出來的結果

範例 A-16　sample\chA\TutorialImageProcess\res\layout\main.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3.     android:orientation="vertical"
4.     android:layout_width="fill_parent"
5.     android:layout_height="fill_parent"
6.     >
7.
8. <ImageView
9.     android:id="@+id/ImageView01"
10.     android:src="@drawable/a1"
11.     android:layout_width="wrap_content"
12.     android:layout_height="wrap_content">
```

```
13. </ImageView>
14. </LinearLayout>
```

範例 A-17 sample\chA\TutorialImageProcess\src\com\example\TutorialImageProcess\TutorialImageProcess.java

```
1.
2. public class TutorialImageProcess extends Activity {
3.     private ImageView mIV;
4.     private Bitmap mBitmap;
5.     private int picw = 320
6.     private int pich = 240;
10.     @Override
11.     public void onCreate(Bundle icicle) {
12.         super.onCreate(icicle);
13.         setContentView(R.layout.main);
14.                     mBitmap = BitmapFactory.decodeResource(getResources(),
                        R.drawable.a1);
15.         picw = mBitmap.getWidth();//  .width();
16.         pich = mBitmap.getHeight();//  .height();
17.         mIV=(ImageView)findViewById(R.id.ImageView01);
18.     }
19.
20.
21.     @Override
22.     public boolean onKeyDown(int keyCode, KeyEvent event) {
23.         if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {  // 當按下中間鍵時
24.            // 呼叫自己寫的函數
25.            TintThePicture(20);
26.            // 儲存檔案
27.            SaveThePicture();
28.             return (true);
29.         }
30.         return super.onKeyDown(keyCode, event);
31.     }
32. }
```

範例 A-18 sample\chA\TutorialImageProcess\src\com\example\TutorialImageProcess\TutorialImageProcess.java

```
1.     private void TintThePicture(int deg) {   // 處理顏色的程式函數
2.            int[] pix = new int[picw * pich]; // 設定放顏色的 array 大小
3.            mBitmap.getPixels(pix, 0, picw, 0, 0, picw, pich);
              // 把 mBitmap 的圖像資料放到 array 中
4.
5.            int RY, GY, BY, RYY, GYY, BYY, R, G, B, Y;
6.            double angle = (3.14159d * (double)deg) / 180.0d;
```

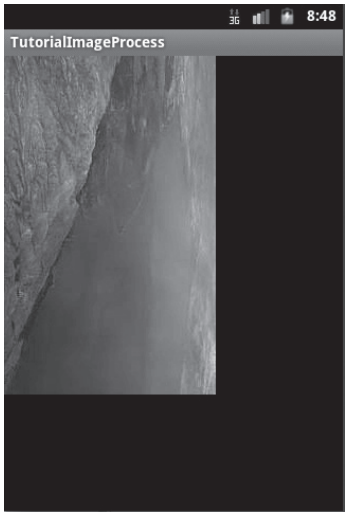
```
7.            int S = (int)(256.0d * Math.sin(angle));
8.            int C = (int)(256.0d * Math.cos(angle));
9.
10.            for (int y = 0; y < pich; y++)
11.            for (int x = 0; x < picw; x++)
12.               {
13.               int index = y * picw + x;
14.               int r = (pix[index] >> 16) & 0xff;
15.               int g = (pix[index] >> 8) & 0xff;
16.               int b = pix[index] & 0xff;
17.               RY = ( 70 * r - 59 * g - 11 * b) / 100;
18.               GY = (-30 * r + 41 * g - 11 * b) / 100;
19.               BY = (-30 * r - 59 * g + 89 * b) / 100;
20.               Y  = ( 30 * r + 59 * g + 11 * b) / 100;
21.               RYY = (S * BY + C * RY) / 256;
22.               BYY = (C * BY - S * RY) / 256;
23.               GYY = (-51 * RYY - 19 * BYY) / 100;
24.               R = Y + RYY;
25.               R = (R < 0) ? 0 : ((R > 255) ? 255 : R);
26.               G = Y + GYY;
27.               G = (G < 0) ? 0 : ((G > 255) ? 255 : G);
28.               B = Y + BYY;
29.               B = (B < 0) ? 0 : ((B > 255) ? 255 : B);
30.               pix[index] = 0xff000000 | (R << 16) | (G << 8) | B;
31.               }
32.
33.            Bitmap bm = Bitmap.createBitmap(picw, pich,Config.ARGB_8888  );
               // 新增加一個 Bitmap
34.            bm.setPixels(pix, 0, picw, 0, 0, picw, pich);
35.            BitmapDrawable tBitmapDrawable = new BitmapDrawable(bm);
36.            mIV.setImageDrawable(tBitmapDrawable); // 顯示在畫面上
37.            mBitmap = bm;
38.            pix = null;
39. }
```

範例 A-19 sample\chA\TutorialImageProcess\src\com\example\TutorialImageProcess\TutorialImageProcess.java

```
1. // 儲存檔案
2.     private void SaveThePicture() {
3.     try {
4.         FileOutputStream fos = super.openFileOutput("output.jpg", MODE_WORLD_
           READABLE);      // 新增加一個檔案 output.jpg
5.         mBitmap.compress(CompressFormat.JPEG, 75, fos);  // 把 mBitmap 的圖，
           用 jpeg 的方法，壓縮到 75% 的容量，並存入 output.jpg 檔案中。
```

```
6.          fos.flush();
7.          fos.close();
8.      } catch (Exception e) {
9.          Log.e("MyLog", e.toString());
10.     }
11. }
```

執行結果：



▲圖 A-15　範例執行結果

執行影片：

範例執行影片 A-4-imageprocess.mov(http://youtu.be/NU-CEe3Cj3k)。