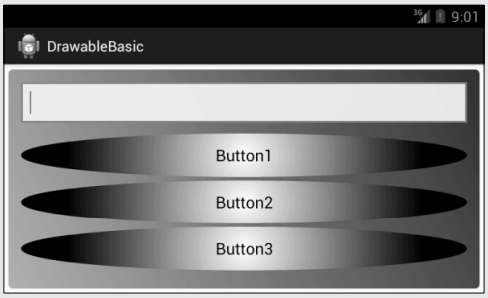
CHAPTER

10 Graphics IIa: Animation

เนื้อหาในบทนี้

- การวาดพื้นหลังของวิวด้วย Shape Drawable



- การวาดพื้นหลังของวิวด้วยภาพ Nine-Patch
- การวาดลงบน Canvas โดยตรง
- การทำ Animation โดยวาดลงบน Canvas
- การทำ Animation ให้กับวิวโดยใช้ XML
- การทำ Animation ให้กับวิวโดยใช้โค้ด Java
- การทำ Animation แบบ Frame-by-Frame
- Property Animation

การวาดพื้นหลังของวิวด้วย Shape Drawable

Drawable คือตัวแทนหรือ abstraction ของสิ่งที่สามารถวาดลงบน UI ได้ เมื่อพูดถึง Drawable เรามักจะนึกถึงรีซอร์สที่เก็บอยู่ในโฟลเดอร์ res\drawable ของโปรเจ็ค เช่น ไฟล์รูปภาพต่างๆ แต่นอกจากไฟล์ภาพแล้ว แอนดรอยด์ยังอนุญาตให้เราใช้ XML กำหนดรูปทรงขึ้นมาเป็น Drawable (เรียกว่า Shape Drawble) แล้วนำไปวาด UI เช่น พื้นหลังของวิว ได้เช่นกัน

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะกำหนด Shape Drawable ขึ้นมา 3 ชุด แล้วนำไปวาดพื้นหลังของวิวต่างๆใน UI

1 สร้างไฟล์ shape01.xml ขึ้นที่โฟลเดอร์ res\drawable (ถ้ายังไม่มีโฟลเดอร์ drawable ภายใต้โฟลเดอร์ res ก็ให้สร้างขึ้นมาก่อน) แล้วพิมพ์โค้ดดังนี้

โปรเจ็ค DrawableBasic, ไฟล์ shape01.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" > ❶

    <stroke ❷
        android:width="6dp"
        android:color="#ffffff" />

    <gradient ❸
        android:type="linear"
        android:startColor="#cc000000"
        android:endColor="#cc2eccfa"
        android:angle="225" />

    <padding ❹
        android:bottom="20dp"
        android:left="20dp"
        android:right="20dp"
        android:top="20dp" />

    <corners ❺
        android:bottomLeftRadius="16dp"
        android:bottomRightRadius="16dp"
        android:topLeftRadius="16dp"
        android:topRightRadius="16dp" />

</shape>
```

❶ กำหนดรูปทรงสี่เหลี่ยม (rectangle)

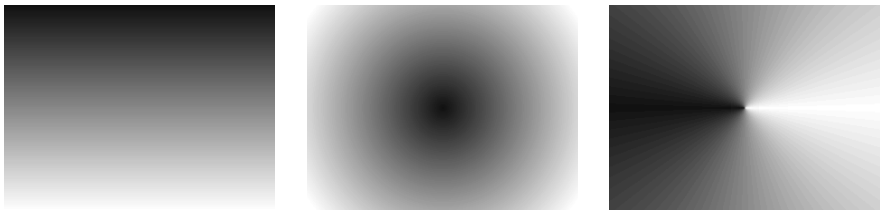
รูปทรงอื่นๆที่กำหนดได้นอกจาก rectangle ก็คือ line (เส้นตรงแนวนอน), oval (วงกลม/วงรี) และ ring (วงแหวน)

line rectangle oval ring

❷ กำหนดรายละเอียดเกี่ยวกับเส้นขอบของรูปทรง ในที่นี้กำหนดเส้นขอบหนา 6 dp สีขาว

❸ กำหนดวิธีการไล่สีภายในรูปทรง ในที่นี้ไล่สีแบบเส้นตรง (linear) โดยเริ่มจากสี #cc000000 ไปยังสี #cc2eccfa ทิศทาง 225 องศา (0 องศาคือซ้ายไปขวา, 90 องศาคือ ล่างขึ้นบน ฯลฯ)

รูปแบบการไล่สีที่กำหนดได้นอกจาก linear ก็คือ radial และ sweep



linear radial sweep

❹ กำหนดระยะ padding (ระยะห่างระหว่างขอบรูปทรงกับเนื้อหาภายในรูปทรง)

❺ กำหนดมุมของรูปทรงให้มีความโค้งมน

2 สร้างไฟล์ shape02.xml ขึ้นที่โฟลเดอร์ res\drawable แล้วพิมพ์โค้ดดังนี้

โปรเจ็ค DrawableBasic, ไฟล์ shape02.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

    <stroke
        android:width="2dp"
        android:color="#ff6600" />

    <solid android:color="#f0ffff99" /> ❶
```

```
    <padding
        android:bottom="10dp"
        android:left="10dp"
        android:right="10dp"
        android:top="10dp" />

</shape>
```

❶ กำหนดสีภายในรูปทรง (กำหนดด้วยสีเดียว แทนที่จะใช้การไล่สีแบบ shape01.xml)

3 สร้างไฟล์ shape03.xml ขึ้นที่โฟลเดอร์ res\drawable แล้วพิมพ์โค้ดดังนี้

โปรเจ็ค DrawableBasic, ไฟล์ shape03.xml

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval" > ❶

    <gradient
        android:type="radial"
        android:startColor="#ffffff"
        android:endColor="#000000"
        android:gradientRadius="300"
        android:useLevel="false" />

</shape>
```

❶ กำหนดรูปทรงวงกลม/วงรี

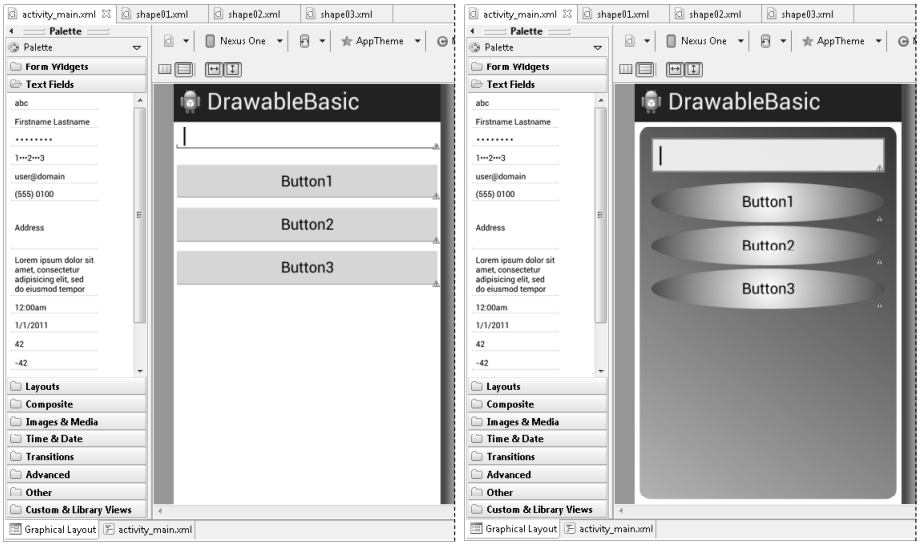
4 กำหนด Layout ดังนี้

โปรเจ็ค DrawableBasic, ไฟล์ activity__main.xml

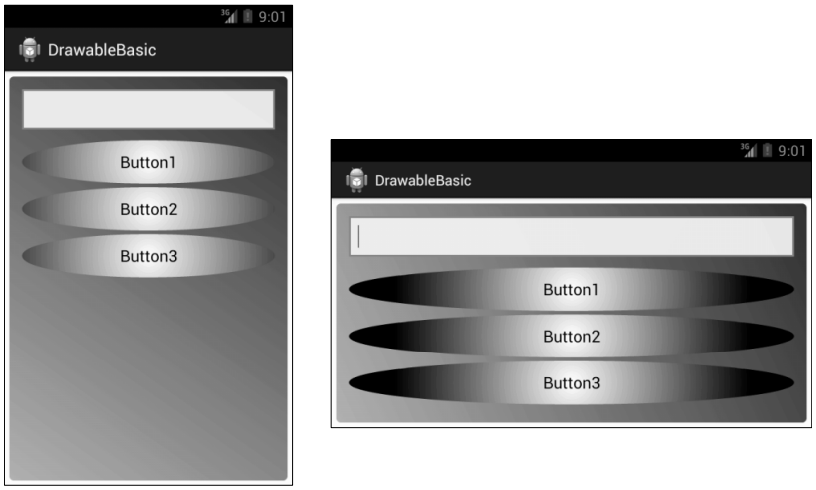
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/shape01"
    android:orientation="vertical" >

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="12dp"
        android:background="@drawable/shape02"
        android:ems="10" >

        <requestFocus />
    </EditText>
```

```
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="4dp"
        android:background="@drawable/shape03"
        android:text="Button1" >
    </Button>

    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="4dp"
        android:background="@drawable/shape03"
        android:text="Button2" >
    </Button>

    <Button
        android:id="@+id/button3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="4dp"
        android:background="@drawable/shape03"
        android:text="Button3" >
    </Button>

</LinearLayout>
```

เรานำ Shape Drawable มาวาดพื้นหลังของ `LinearLayout`, `EditText` และปุ่มทั้งสาม ซึ่งเมื่อแสดง Layout ในมุมมองกราฟิกจะได้ดังรูปขวา ส่วนรูปซ้ายคือหน้าตาปกติของ Layout เมื่อตัดแอตทริบิวต์ `background` ของทุกอิลิเมนต์ออกไป



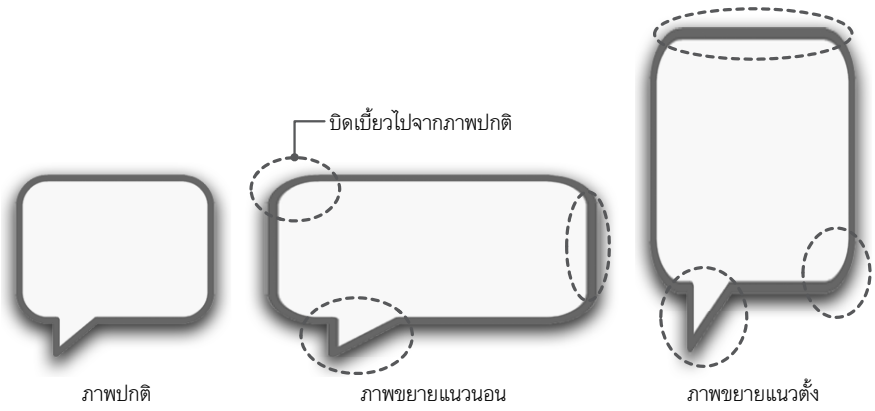
ให้สังเกตว่าในไฟล์ XML ของ Shape Drawable เราไม่ได้ระบุขนาดรูปทรง แอนดรอยด์จึงวาดรูปทรงให้พอดีกับขนาดของวิว ซึ่งเมื่อวิวเปลี่ยนขนาดก็จะทำให้รูปทรงเปลี่ยนขนาดตามไปด้วย เช่น เมื่อหมุนหน้าจอ ดังรูป



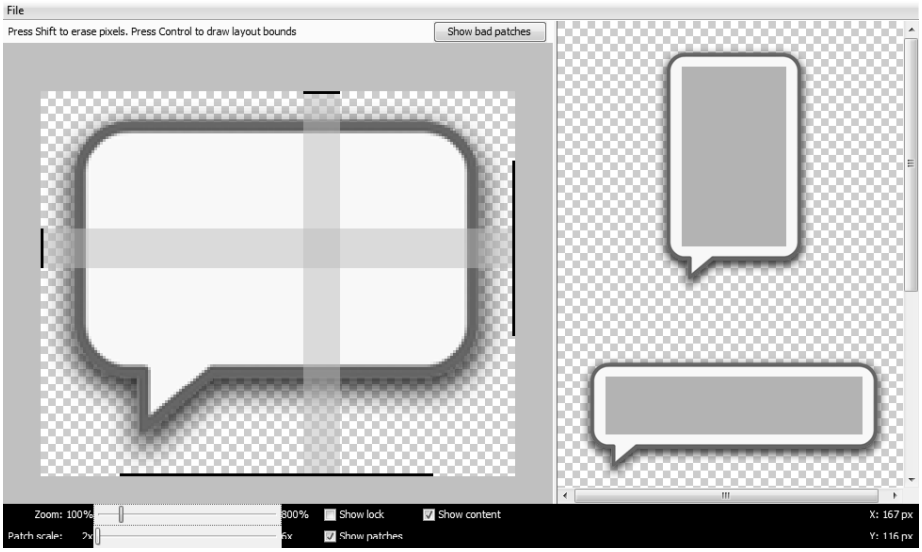
การวาดพื้นหลังของวิวด้วย Shape Drawable ทำให้ได้กราฟิกที่สวยงามลงตัวและสามารถปรับเปลี่ยนขนาดได้ (scalable) ไม่ว่าขนาดของวิว ความละเอียดหน้าจอ หรือความหนาแน่นจุดภาพของหน้าจอจะเป็นเท่าใด

การวาดพื้นหลังของวิวด้วยภาพ Nine-Patch

ข้อจำกัดของ Shape Drawable ในตัวอย่างที่ผ่านมาก็คือมีรูปทรงให้กำหนดได้เพียงไม่กี่แบบ และไม่สามารถใส่ลูกเล่นอะไรได้มากนัก ดังนั้นหากคุณต้องการกราฟิกที่ซับซ้อนยิ่งขึ้นก็จะต้องเตรียมไฟล์ภาพมาเอง แต่ปัญหามีอยู่ว่า แอนดรอยด์สนับสนุนเฉพาะภาพแบบ bitmap (เช่น PNG, JPEG) ซึ่งเมื่อถูกปรับขนาดจะทำให้ภาพบิดเบี้ยวไป ดังรูป



เพื่อแก้ปัญหาดังกล่าว แอนดรอยด์จึงรองรับไฟล์ภาพ PNG แบบพิเศษที่เรียกว่า Nine-patch ซึ่งเราสามารถกำหนดว่าส่วนใดของภาพที่ให้ยืดออกได้หรือไม่ได้ โดยใช้โปรแกรม draw9patch ที่ติดตั้งมาพร้อมกับ Android SDK (ไฟล์ draw9patch.bat ในโฟลเดอร์ *<ADT folder>*\sdk\tools) ดังรูป



เมื่อเปิดไฟล์ PNG ขึ้นมาในโปรแกรม draw9patch โปรแกรมจะเพิ่มขอบขนาด 1 พิกเซล รอบภาพ เราสามารถกำหนดให้จุดภาพโดยรอบเหล่านี้เป็นสีโปร่งใสหรือสีดำอย่างใดอย่างหนึ่ง โดยจุดสีดำที่แต่ละด้านจะมีความหมายดังนี้

ด้านซ้าย	กำหนดพื้นที่ของภาพที่สามารถยืดออกในแนวตั้งได้
ด้านบน	กำหนดพื้นที่ของภาพที่สามารถยืดออกในแนวนอนได้
ด้านขวา	กำหนดพื้นที่เนื้อหา (content area) ในแนวตั้ง ซึ่งเนื้อหาของวิว (เช่น ข้อความบนปุ่ม, ข้อความใน TextView ฯลฯ) จะถูกแสดงภายในพื้นที่นี้
ด้านล่าง	กำหนดพื้นที่เนื้อหา (content area) ในแนวนอน ซึ่งเนื้อหาของวิวจะถูกแสดงภายในพื้นที่นี้

ที่ฝั่งขวาของโปรแกรมจะแสดงตัวอย่างภาพเมื่อถูกยืดออกในแนวต่างๆ โดยสี่เหลี่ยมข้างในก็คือพื้นที่สำหรับแสดงเนื้อหาของวิว ตามที่กำหนดด้วยจุดสีดำทางด้านขวาและด้านล่างของภาพ (ต้องเลือก Show content จึงจะแสดงสี่เหลี่ยมนี้)

หลังจากกำหนดเรียบร้อยแล้ว ให้บันทึกไฟล์ภาพเป็นนามสกุล .9.png เพื่อนำไปใช้ในโปรเจ็คแอนดรอยด์ต่อไป

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะกำหนดภาพ Nine-patch เป็นพื้นหลังของ `TextView`

1. สร้างไฟล์ภาพ Nine-patch ชื่อ text_bubble.9.png แล้วอิมพอร์ตเข้ามาในโฟลเดอร์ res\drawable ของโปรเจ็ค
2. กำหนด Layout ของหน้าจอดังนี้

โปรเจ็ค DrawableNinePatch, ไฟล์ activity__main.xml

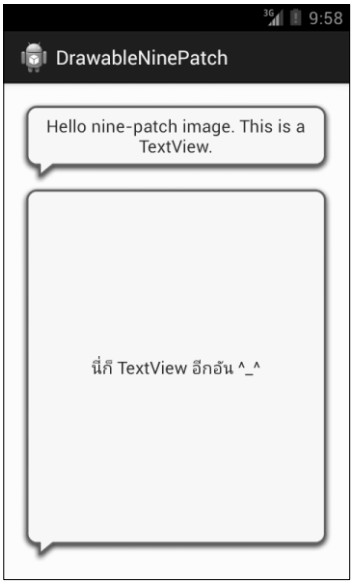
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp" >

    <TextView
        android:layout_width="match_parent" ❷
        android:layout_height="wrap_content"
        android:background="@drawable/text_bubble" ❶
        android:gravity="center"
        android:text="Hello nine-patch image. This is a TextView."
        android:textSize="16sp" />
```

```
    <TextView
        android:layout_width="match_parent" ❷
        android:layout_height="0dp"
        android:layout_weight="1" ❸
        android:background="@drawable/text_bubble" ❶
        android:gravity="center"
        android:text="นี่ก็ TextView อีกอัน ^_^"
        android:textSize="16sp" />
</LinearLayout>
```

เราสร้าง `TextView` 2 อันใน Layout และกำหนดให้วาดพื้นหลังของ `TextView` ทั้งสองด้วยภาพจากไฟล์ text_bubble.9.png ❶

`TextView` ทั้งสองจะมีความกว้างเต็ม Layout ❷ และ `TextView` อันหลังจะใช้ความสูงของ Layout ที่เหลือทั้งหมด ❸

ผลการรัน



จะเห็นว่าภาพ Nine-patch นั้นแม้ถูกยืดออกก็ยังคงดูสวยงาม ไม่บิดเบี้ยว ต่างจากภาพ bitmap ทั่วไป ภาพ Nine-patch จึงเหมาะกับการสร้าง UI ในแบบของเราเอง ซึ่งสามารถปรับเปลี่ยนขนาดได้อย่างลงตัว

การวาดลงบน Canvas โดยตรง

ถ้าหากคุณต้องการสร้าง UI โดยวาดกราฟิกเองทั้งหมด เช่น หน้าจอเกมหมากรุก เกมซูโดกุ หรือเกมจับคู่ภาพ เป็นต้น วิธีที่เหมาะสมที่สุดก็คือการวาดลงบน Canvas โดยตรง

วิวทุกชนิดจะมีเมธอด `onDraw` ซึ่งเราสามารถ Override เพื่อระบุโค้ดที่ทำหน้าที่วาด UI ของวิวนั้นๆได้ โดยการวาดจะใช้เมธอด เช่น `drawLine`, `drawCircle`, `drawBitmap` ของออบเจ็ค `Canvas` ที่แอนดรอยด์ส่งผ่านเป็นพารามิเตอร์มายังเมธอด `onDraw`

NOTE

เมธอด onDraw เป็น Callback Method กล่าวคือ เราไม่ได้เรียกเมธอดนี้เอง แต่แอนดรอยด์จะเรียกให้อัตโนมัติเมื่อจำเป็นต้องวาด UI ของวิวนั้นใหม่ (re-draw) ดังนั้นเราอยากวาดอะไรก็เพียงแต่ใส่โค้ดไว้ในเมธอดนี้

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะสร้าง Custom View (ซับคลาสของคลาส `View`) และวาดกราฟิกลงบนวิวนี้ตรงตำแหน่งที่ผู้ใช้แตะ (หรือคลิกเมาส์ในกรณีของอีมูเลเตอร์)

1 สร้างคลาสใหม่ชื่อ MyView.java แล้วพิมพ์โค้ด ดังนี้

โปรเจ็ค CanvasDraw, ไฟล์ MyView.java

```
package com.example.canvasdraw;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.view.MotionEvent;
import android.view.View;

public class MyView extends View {

    private static final int RADIUS = 90; // ขนาดรัศมีวงกลมที่จะวาด

    private Paint mPaint;   // พู่กันวาดภาพ
    private Bitmap mImage;  // ภาพจากไฟล์ ic_launcher.png
    private float mX, mY;   // ตำแหน่งที่ผู้ใช้แตะ

    public MyView(Context context) {
        super(context);
```

```
        // เตรียมออบเจ็คต่างๆที่จะใช้ในเมธอด onDraw
        mPaint = new Paint();
        mImage = BitmapFactory.decodeResource(getResources(),
                                              R.drawable.ic_launcher);
        // ระบุการทำงานเมื่อแตะหน้าจอในส่วนของวิวนี้
        setOnTouchListener(new View.OnTouchListener() {

            @Override
            public boolean onTouch(View v, MotionEvent event) {
                // อ่านตำแหน่งที่ถูกแตะมาเก็บลงตัวแปร mX และ mY
                mX = event.getX();
                mY = event.getY();
                // ให้วาด UI ของวิวใหม่ ซึ่งจะทำให้แอนดรอยด์เรียกมายังเมธอด onDraw
                invalidate();
                return true;
            }
        });
    }

    @Override
    public void onDraw(Canvas canvas) {
        // วาดวงกลม ให้ศูนย์กลางอยู่ตรงตำแหน่งที่แตะ
        mPaint.setColor(Color.argb(200, 50, 0, 255));
        mPaint.setStyle(Paint.Style.STROKE);
        mPaint.setStrokeWidth(10);
        mPaint.setAntiAlias(true);
        canvas.drawCircle(mX, mY, RADIUS, mPaint); ❶

        // วาดข้อความ ให้ตำแหน่งที่แตะอยู่กึ่งกลางข้อความในแนวนอน
        mPaint.setColor(Color.RED);
        mPaint.setStyle(Paint.Style.FILL);
        mPaint.setTextSize(40);
        mPaint.setTextAlign(Paint.Align.CENTER);
        canvas.drawText("Android", mX, mY - 10, mPaint); ❷

        // วาดภาพจากไฟล์ ic_launcher.png ให้ตำแหน่งที่แตะอยู่กึ่งกลางภาพในแนวนอน
        canvas.drawBitmap(mImage, mX - (mImage.getWidth() / 2), mY,
                          mPaint); ❸
    }
}
```

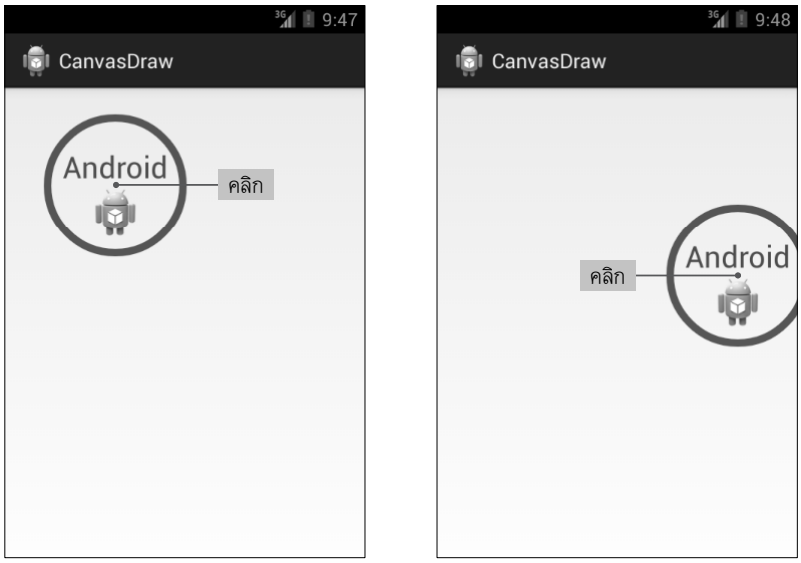
การวาดลงบน Canvas จะต้องเตรียมออบเจ็ค Paint ซึ่งเปรียบเสมือนพู่กันวาดภาพ โดยลักษณะต่างๆของสิ่งที่วาดออกมา เช่น สี ขนาดเส้น ขนาดตัวอักษร ฯลฯ จะกำหนดโดยออบเจ็ค Paint นี้ ให้สังเกตว่าการเรียกเมธอด drawCircle เพื่อวาดวงกลม ❶, drawText เพื่อวาดข้อความ ❷ และ drawBitmap เพื่อวาดภาพจากไฟล์ ❸ จะมีการระบุออบเจ็ค Paint เป็นพารามิเตอร์ตัวสุดท้ายทั้งหมด

2 แก้ไขเมธอด onCreate ของแอคทิวิตี เพื่อนำ MyView มากำหนดเป็น UI ของแอคทิวิตี

โปรเจ็ค CanvasDraw, ไฟล์ MainActivity.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(new MyView(this));
}
```

ผลการรัน



การทำ Animation โดยวาดลงบน Canvas

จากหลักการในหัวข้อที่แล้ว เราสามารถเพิ่มโค้ดเพื่อควบคุมให้มีการวาดภาพลงบน Canvas อย่างต่อเนื่อง โดยภาพที่วาดใหม่แต่ละครั้งจะแตกต่างจากเดิมเล็กน้อย ทำให้เกิดเป็นภาพเคลื่อนไหวหรือแอนิเมชั่น (Animation) ขึ้น ซึ่งหลักการนี้สามารถนำไปประยุกต์สร้างเกมแอนิเมชั่นง่ายๆที่ไม่ต้องการประสิทธิภาพในด้านกราฟิกมากนัก เช่น เกมงู หรือเกม Tetris เป็นต้น

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะวาดวงกลมที่กระเด้งไปมาอยู่ภายในหน้าจอ

1 สร้างคลาสใหม่ชื่อ MyView.java แล้วพิมพ์โค้ดดังนี้

โปรเจ็ค CanvasDrawAnimation, ไฟล์ MyView.java

```
package com.example.canvasdrawanimation;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.os.Handler;
import android.view.View;

public class MyView extends View {

    private static final int RADIUS = 40;      // ขนาดรัศมีวงกลม

    private static final int DISTANCE_X = 5;  // ระยะการเคลื่อนที่ในแนวนอน (x)
    private static final int DISTANCE_Y = 5;  // ระยะการเคลื่อนที่ในแนวตั้ง (y)

    /*************************************************************************
     * Note: ถ้ากำหนดระยะการเคลื่อนที่แนวนอนและแนวตั้งเท่ากัน จะเคลื่อนที่ในทิศทาง 45 องศา
     *************************************************************************/

    private Paint mPaint;  // พู่กันวาดภาพ
    private float mX, mY;  // ตำแหน่งปัจจุบันของจุดศูนย์กลางวงกลม

    // ทิศทางการเคลื่อนที่ในแนวนอน (1 เคลื่อนที่ไปทางขวา, -1 เคลื่อนที่ไปทางซ้าย)
    private int mDirectionX = 1;
    // ทิศทางการเคลื่อนที่ในแนวตั้ง (1 เคลื่อนที่ลง, -1 เคลื่อนที่ขึ้น)
    private int mDirectionY = 1;

    // ออบเจ็ค Handler สำหรับใส่ (post) Runnable ลงใน Message Queue
    Handler mHandler = new Handler();

    // ออบเจ็ค Runnable – โค้ดที่ใช้อัพเดทตำแหน่งวงกลม
```

```
Runnable mTick = new Runnable() { ❶
    public void run() { ❷
        // อัพเดทตำแหน่งของวงกลมในแนวนอน
        mX += (mDirectionX * DISTANCE_X);

        // เมื่อเคลื่อนที่ถึงขอบด้านซ้ายหรือขวาของวิว ให้เคลื่อนที่กลับในทิศตรงข้าม
        if (mX - RADIUS <= 0) {
            mDirectionX = 1;
        } else if (mX + RADIUS >= MyView.this.getWidth()) {
            mDirectionX = -1;
        }

        // อัพเดทตำแหน่งของวงกลมในแนวตั้ง
        mY += (mDirectionY * DISTANCE_Y);

        // เมื่อเคลื่อนที่ถึงขอบด้านบนหรือล่างของวิว ให้เคลื่อนที่กลับในทิศตรงข้าม
        if (mY - RADIUS <= 0) {
            mDirectionY = 1;
        } else if (mY + RADIUS >= MyView.this.getHeight()) {
            mDirectionY = -1;
        }

        // ให้วาด UI ของวิวใหม่ ซึ่งแอนดรอยด์จะเรียกมายังเมธอด onDraw
        invalidate(); ❸

        /* ใส่ (post) Runnable นี้ ลงใน Message Queue ของเธรดเพื่อให้รันซ้ำในอีก 20
           มิลลิวินาทีข้างหน้า */
        mHandler.postDelayed(this, 20); ❹
    }
}; // จบออบเจ็ค Runnable ตรงนี้

// เริ่มแอนิเมชั่นโดยใส่ Runnable ลงใน Message Queue
void startAnimation() {
    mHandler.removeCallbacks(mTick);
    mHandler.post(mTick);
}

// หยุดแอนิเมชั่นโดยลบ Runnable ออกจาก Message Queue
void stopAnimation() {
    mHandler.removeCallbacks(mTick);
}

public MyView(Context context) {
    super(context);
```

```
        /* เตรียมพู่กันที่ใช้วาดวงกลม โดยกำหนดคุณสมบัติต่างๆให้เรียบร้อยตรงนี้เลย เพื่อลดงานใน
           เมธอด onDraw ให้เหลือน้อยที่สุด */
        mPaint = new Paint();
        mPaint.setColor(Color.BLUE);
        mPaint.setStyle(Paint.Style.FILL);
        mPaint.setAntiAlias(true);

        startAnimation();  // เริ่มแอนิเมชั่น
    }

    @Override
    public void onDraw(Canvas canvas) {
        // ใช้พู่กัน mPaint วาดวงกลมขนาดรัศมี RADIUS โดยให้จุดศูนย์กลางอยู่ที่ mX, mY
        canvas.drawCircle(mX, mY, RADIUS, mPaint);
    }
}
```

การทำให้วงกลมที่วาดเคลื่อนไหวได้ จะใช้วิธีสร้างออบเจ็ค Runnable ❶ โดยเตรียมโค้ดที่อัพเดทตำแหน่งจุดศูนย์กลางวงกลม (ตัวแปร mX, mY) ไว้ในเมธอด run ❷ ของออบเจ็คนี้ หลังจากอัพเดทตำแหน่งแล้วจะเรียกเมธอด invalidate ❸ เพื่อขอให้แอนดรอยด์วาด UI ของวิว (MyView) ใหม่ จากนั้นก็จะใส่ Runnable ลงใน Message Queue เพื่อรันซ้ำในอีก 20 มิลลิวินาทีข้างหน้า ❹ ซึ่งการทำงานจะวนซ้ำอย่างนี้ไปเรื่อยๆจนกว่าจะออกจากแอพหรือหยุดแอนิเมชั่นโดยใช้เมธอด stopAnimation (แต่ในตัวอย่างนี้ไม่ได้เรียกเมธอด stopAnimation)

NOTE

Message Queue คือ คิวการประมวลผลของเธรด ซึ่งกรณีของเธรดหลัก (UI Thread) เมื่อเกิดอีเวนต์ขึ้นใน UI โค้ดที่ใช้จัดการอีเวนต์นั้นก็จะถูกใส่ลง Message Queue เพื่อรอการประมวลผลไปตามลำดับ

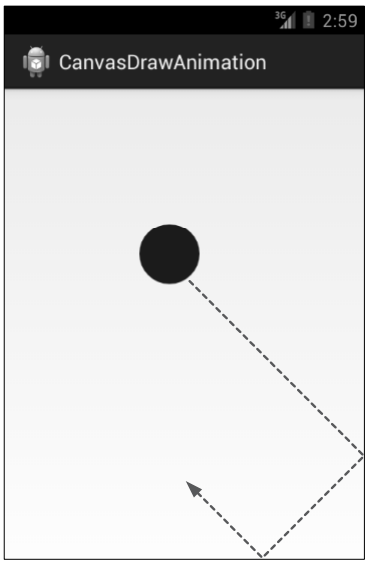
การใช้ Handler ร่วมกับ Runnable เป็นวิธีที่ช่วยให้เราใส่โค้ดที่ต้องการรันลงใน Message Queue เองได้ โดย Runnable ก็คือโค้ดที่เราต้องการรัน (จริงๆโค้ดจะอยู่ในเมธอด run ของ Runnable อีกที) ส่วน Handler ให้คิดง่ายๆว่าคือ "ออบเจ็คตัวช่วย" (Helper Object) สำหรับใส่ Runnable ลงใน Message Queue

2 แก้ไขเมธอด onCreate ของแอคทิวิตี เพื่อนำ MyView มากำหนดเป็น UI ของแอคทิวิตี

โปรเจ็ค CanvasDrawAnimation, ไฟล์ MainActivity.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(new MyView(this));
}
```

ผลการรัน



การทำ Animation ให้กับวิวโดยใช้ XML

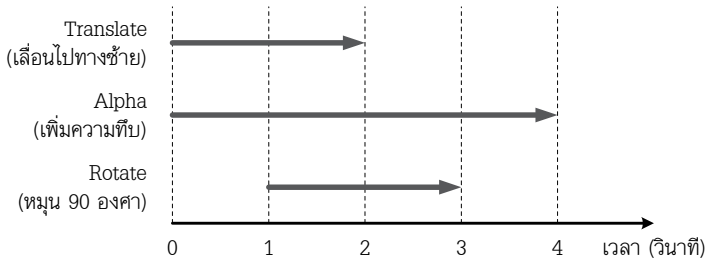
แอนดรอยด์มี API สำหรับสร้างแอนิเมชั่นในรูปแบบที่เรียกว่า Tweened Animation ซึ่งเป็นการทำให้ UI เคลื่อนไหวโดยการเปลี่ยนตำแหน่ง ขนาด ทิศทาง และความโปร่ง/ทึบของมัน ยกตัวอย่างเช่น เราอาจทำให้ `EditText` “สั่น” เมื่อผู้ใช้กรอกข้อมูลไม่ถูกต้อง เป็นต้น

ข้อดีของ Tweened Animation เมื่อเทียบกับการทำแอนิเมชั่นด้วยวิธีอื่นๆ (เช่น การวาดลงบน Canvas เอง) ก็คือ ความง่าย และใช้ทรัพยากรของระบบน้อยกว่า

Tweened Animation สามารถทำได้ทั้งโดยการใช้ XML และโค้ดจาวา ในหัวข้อนี้จะแสดงการใช้ XML ส่วนการใช้โค้ดจาวาจะอยู่ในหัวข้อถัดไป

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะทำแอนิเมชั่นให้กับ `TextView` โดยตอนเริ่มต้น `TextView` จะอยู่ทางขวาของจอและมีความโปร่งจนเกือบมองไม่เห็น เราจะทำให้มันวิ่งมาทางซ้ายพร้อมกับค่อยๆเพิ่มความทึบขึ้นเป็น 100% หลังจากนั้นจะหมุน 90 องศา (การเลื่อนมาทางซ้ายและการเพิ่มความทึบจะเริ่มพร้อมกัน แต่การหมุนจะเริ่มหลังจากนั้น 1 วินาที ซึ่งตอนนั้นการเลื่อนและการเพิ่มความทึบยังไม่สิ้นสุด ดังรูปหน้าถัดไป)



1 สร้างไฟล์ my_anim.xml ขึ้นที่โฟลเดอร์ res\anim (ถ้ายังไม่มีโฟลเดอร์ anim ภายใต้โฟลเดอร์ res ก็ให้สร้างขึ้นมาก่อน) แล้วพิมพ์โค้ดดังนี้

โปรเจ็ค ViewAnimationXML, ไฟล์ res\anim\my__anim.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android" >

    <translate
        android:interpolator="@android:anim/accelerate_interpolator" ❷
        android:fromXDelta="100%p"
        android:toXDelta="0"                                          ❶
        android:duration="2000" />

    <alpha
        android:fromAlpha="0.2"
        android:toAlpha="1.0"                                         ❸
        android:duration="4000" />

    <rotate
        android:interpolator=
            "@android:anim/accelerate_decelerate_interpolator" ❷
        android:startOffset="1000" ❻
        android:fromDegrees="0"
        android:toDegrees="-90"                                       ❹
        android:duration="2000"
        android:pivotX="100%"
        android:pivotY="0%" />                                        ❺
</set>
```

ไฟล์ my_anim.xml ข้างต้นคือ Animation Resource ที่กำหนดรายละเอียดของแอนิเมชั่น

Tweened Animation มี 4 ประเภทคือ

Translate การเลื่อนจากจุดหนึ่งไปยังอีกจุดหนึ่ง; กำหนดด้วยแท็ก `<translate>` ใน XML หรือออบเจ็ค `TranslateAnimation` ในโค้ดจาวา

Scale	การปรับเปลี่ยนขนาด ซึ่งปรับได้ทั้งแนวนอน (x), แนวตั้ง (y) หรือทั้งสองแนวพร้อมกัน; กำหนดด้วยแท็ก `<scale>` ใน XML หรือออบเจ็ค `ScaleAnimation` ในโค้ดจาวา
Rotate	การหมุน; กำหนดด้วยแท็ก `<rotate>` ใน XML หรือออบเจ็ค `RotateAnimation` ในโค้ดจาวา
Alpha	การเปลี่ยนความโปร่ง/ทึบ; กำหนดด้วยแท็ก `<alpha>` ใน XML หรือออบเจ็ค `AlphaAnimation` ในโค้ดจาวา

เราสามารถกำหนดแอนิเมชั่นหลายประเภทพร้อมกันได้โดยใช้แท็ก `<set>` ครอบ ดังเช่นตัวอย่างนี้ ซึ่งแอนิเมชั่นที่อยู่ใน `<set>` จะถูกทำไปพร้อมๆกัน (แต่สามารถระบุแอตทริบิวต์ `startOffset` เพื่อให้บางแอนิเมชั่นเริ่มต้น ณ เวลาที่ต้องการได้)

ความหมายของแอนิเมชั่นในไฟล์ my_anim.xml คือ ให้เลื่อนจากตำแหน่ง 100%p ไปยังตำแหน่ง 0 ในเวลา 2 วินาที (2,000 มิลลิวินาที) ❶ ทั้งนี้ในการระบุค่าตำแหน่ง ถ้าระบุตัวเลขล้วนๆจะหมายถึงพิกเซล, ถ้าระบุ % จะหมายถึงตำแหน่งเมื่อเทียบกับตัวเอง และถ้าระบุ %p จะหมายถึงตำแหน่งเมื่อเทียบกับวิวที่บรรจุวิวนั้นไว้ (parent)

ดังนั้นการเลื่อนจากตำแหน่ง 100%p ไปยังตำแหน่ง 0 จึงมีความหมายว่า ตอนเริ่มให้อยู่ที่ขอบด้านขวาของ parent และตอนจบให้อยู่ที่ตำแหน่ง 0 พิกเซล

สำหรับแอตทริบิวต์ `interpolator` ❷ ใช้กำหนดลักษณะการเคลื่อนไหว/เปลี่ยนแปลง เช่น ค่า accelerate_interpolator หมายถึงให้เคลื่อนไหว/เปลี่ยนแปลงอย่างช้าๆในช่วงแรก แล้วเพิ่มอัตราการเคลื่อนไหว/เปลี่ยนแปลงขึ้นเรื่อยๆเมื่อเวลาผ่านไป

นอกจากนี้เรายังเพิ่มความทึบจาก 0.2 ไปเป็น 1.0 ในเวลา 4 วินาที ❸ ซึ่งค่าความโปร่ง/ทึบที่กำหนดได้คือตั้งแต่ 0.0 (โปร่งใส) ถึง 1.0 (ทึบสนิท)

แอนิเมชั่นประเภทสุดท้ายในตัวอย่างนี้คือการหมุนจากมุม 0 องศา ไปยังมุม -90 องศา ในเวลา 2 วินาที ❹ โดยใช้ตำแหน่งมุมบนขวาของวิวนั้นๆเป็นจุดหมุน ❺ การหมุนจะเริ่มหลัง Translate และ Alpha อยู่ 1 วินาที ❻

2 กำหนด Layout ของหน้าจอ

โปรเจ็ค ViewAnimationXML, ไฟล์ activity__main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
    android:orientation="vertical"
    android:padding="16dp" >

    <Button
        android:id="@+id/animate_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="16dp"
        android:text="Animate Text" />

    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        android:textSize="20sp" />

</LinearLayout>
```

3 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี

โปรเจ็ค ViewAnimationXML, ไฟล์ MainActivity.java

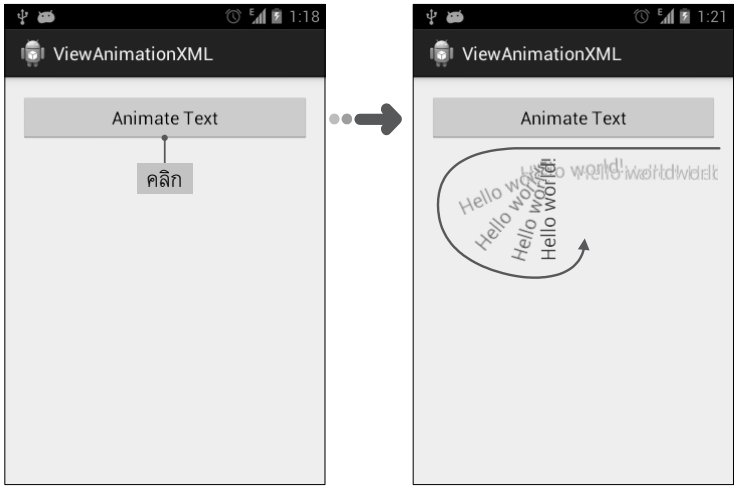
```
final TextView text = (TextView) findViewById(R.id.text);
text.setVisibility(View.INVISIBLE); ❸

// โหลดแอนิเมชั่นจากไฟล์ my_anim.xml และระบุการทำงานตอนแอนิเมชั่นเริ่มต้นและสิ้นสุด
final Animation anim = AnimationUtils
                .loadAnimation(MainActivity.this, R.anim.my_anim); ❶
anim.setAnimationListener(new AnimationListener() {

    @Override
    public void onAnimationStart(Animation animation) {
        // แสดง TextView ตอนแอนิเมชั่นเริ่มต้น
        text.setVisibility(View.VISIBLE); ❹
    }

    @Override
    public void onAnimationEnd(Animation animation) {
        // ซ่อน TextView ตอนแอนิเมชั่นสิ้นสุด
        text.setVisibility(View.INVISIBLE); ❺
    }

    @Override
    public void onAnimationRepeat(Animation animation) {
    }
});
```

```
// เมื่อปุ่มถูกคลิกจะเริ่มต้นแอนิเมชั่น
Button btnAnimate = (Button) findViewById(R.id.animate_button);
btnAnimate.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        text.startAnimation(anim); ❷
    }
});
```

หลังจากกำหนดแอนิเมชั่นด้วยไฟล์ XML แล้ว เมื่อต้องการนำมาใช้ (apply) กับวิวใน UI ก่อนอื่นให้โหลดแอนิเมชั่นจากไฟล์ XML โดยใช้เมธอด `loadAnimation` ❶ ซึ่งเป็น Static Method ของคลาส `AnimationUtils` แล้วจึงสั่งเริ่มแอนิเมชั่นโดยเรียกเมธอด `startAnimation` ❷

Tweened Animation จะมีผลกับวิวเพียงชั่วคราว กล่าวคือหลังจากแอนิเมชั่นสิ้นสุดลง วิวจะกลับมาอยู่ในตำแหน่งและสภาพเดิมก่อนที่เราจะ apply แอนิเมชั่นให้มัน ดังนั้นเพื่อไม่ให้ตัวอย่างนี้ดูแปลกๆ เราจึงซ่อน `TextView` ไว้ตอนที่แอพรันขึ้นมา ❸ จากนั้นเมื่อแอนิเมชั่นเริ่มต้นก็ค่อยแสดง `TextView` ออกมา ❹ และเมื่อแอนิเมชั่นสิ้นสุดก็จะซ่อนไว้ตามเดิม ❺

ผลการรัน

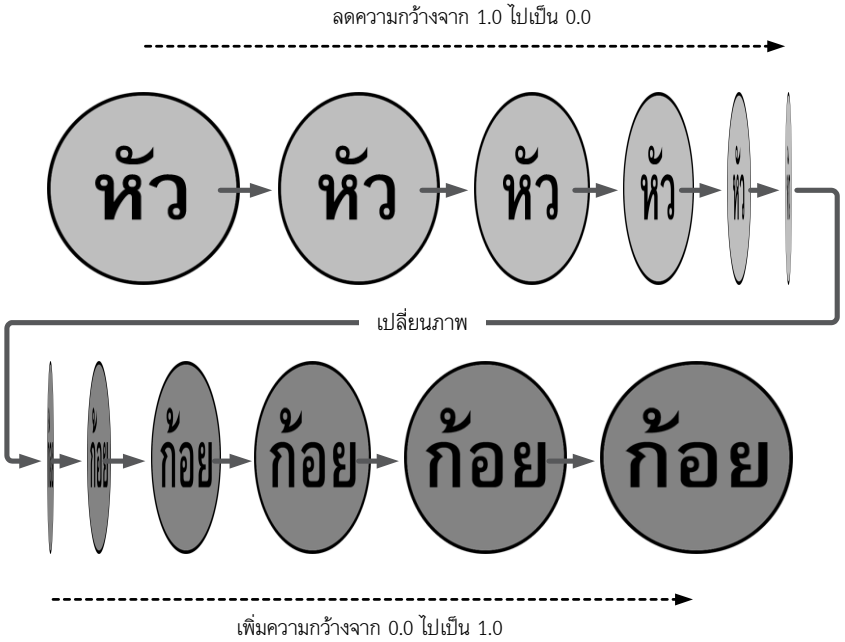


การทำ Animation ให้กับวิวโดยใช้โค้ด Java

ในหัวข้อนี้จะแสดงการทำ Tweened Animation ด้วยโค้ดจาวา

ตัวอย่างและคำอธิบาย

เราจะใช้ Tweened Animation ประเภท Scale สร้างภาพแอนิเมชั่นของเหรียญที่พลิกจากด้านหัวไปด้านก้อยหรือกลับกัน ดังรูป



1 กำหนด Layout ของหน้าจอ

โปรเจ็ค ViewAnimationJava, ไฟล์ activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true" />

</RelativeLayout>
```

2 เพิ่มโค้ดในแอคทิวิตีจนเป็นดังนี้

โปรเจ็ค ViewAnimationJava, ไฟล์ MainActivity.java

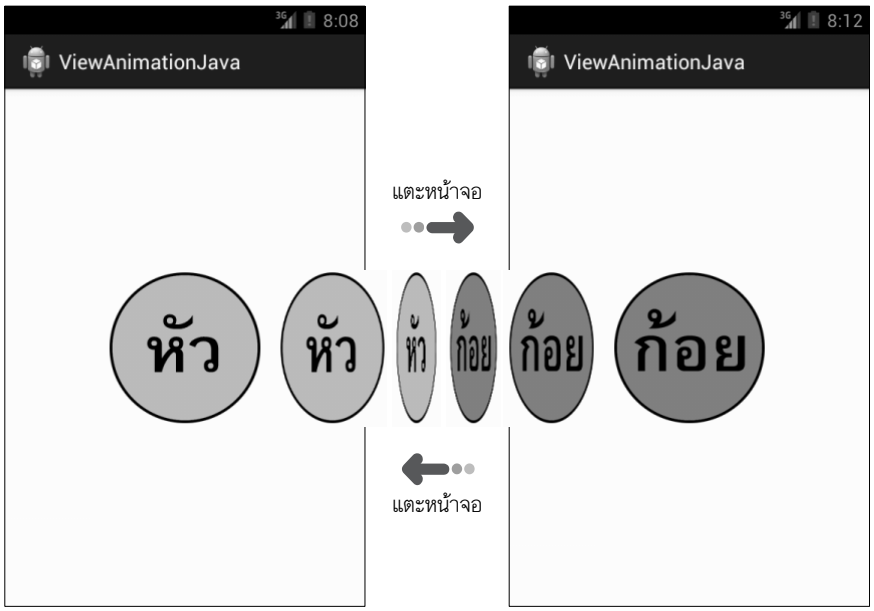
```
package com.example.viewanimationjava;

import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.animation.Animation;
import android.view.animation.ScaleAnimation;
import android.widget.ImageView;

public class MainActivity extends Activity {

    ImageView imgCoin;            // ภาพเหรียญ
    ScaleAnimation shrink, grow; // แอนิเมชั่นที่ใช้ลดความกว้าง, เพิ่มความกว้าง
    boolean isHead;               /* จดจำว่าตอนนั้นแสดงด้านหัวหรือก้อย (เพื่อสลับภาพ
                                     หัวเป็นก้อย และก้อยเป็นหัว) */

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

        // เริ่มต้นจะแสดงภาพเหรียญด้านหัว
        imgCoin = (ImageView) findViewById(R.id.image);
        imgCoin.setImageResource(R.drawable.coin_head);
        isHead = true;

        // กำหนดแอนิเมชั่นที่ใช้ลดความกว้าง ❶
        shrink = new ScaleAnimation(
                        1.0f, // ความกว้างตอนเริ่มต้นแอนิเมชั่น
                        0.0f, // ความกว้างตอนสิ้นสุดแอนิเมชั่น
                        1.0f, // ความสูงตอนเริ่มต้นแอนิเมชั่น
                        1.0f, // ความสูงตอนสิ้นสุดแอนิเมชั่น
                        ScaleAnimation.RELATIVE_TO_SELF,
                        0.5f, // ให้ใช้กึ่งกลางภาพเป็นจุดยึดสำหรับการเปลี่ยนขนาดความกว้าง
                        ScaleAnimation.RELATIVE_TO_SELF,
                        0.5f  // ให้ใช้กึ่งกลางภาพเป็นจุดยึดสำหรับการเปลี่ยนขนาดความสูง
        );
        shrink.setDuration(150); // กำหนดช่วงเวลาของแอนิเมชั่นเป็น 150 มิลลิวินาที

        // กำหนด Listener เพื่อระบุการทำงานตอนสิ้นสุดแอนิเมชั่น
        shrink.setAnimationListener(new Animation.AnimationListener() {
```

```
            @Override
            public void onAnimationStart(Animation animation) {
            }

            @Override
            public void onAnimationRepeat(Animation animation) {
            }

            // เมื่อสิ้นสุดแอนิเมชั่นการลดขนาด จะสลับภาพ และเริ่มแอนิเมชั่นการเพิ่มขนาด
            @Override
            public void onAnimationEnd(Animation animation) {
                if (isHead) {
                    isHead = false;
                    imgCoin.setImageResource(R.drawable.coin_tail);
                } else {
                    isHead = true;
                    imgCoin.setImageResource(R.drawable.coin_head);
                }
                imgCoin.startAnimation(grow); ❺
            }
        });

        // กำหนดแอนิเมชั่นที่ใช้เพิ่มความกว้าง ❷
        grow = new ScaleAnimation(0.0f, 1.0f, 1.0f, 1.0f,
                                  ScaleAnimation.RELATIVE_TO_SELF, 0.5f,
                                  ScaleAnimation.RELATIVE_TO_SELF, 0.5f);
        grow.setDuration(150);
    }

    // เมื่อแตะหน้าจอ จะเริ่มแอนิเมชั่นการลดขนาด
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            imgCoin.startAnimation(shrink); ❸
            return true;
        }
        return super.onTouchEvent(event);
    }
}
```

❹

เราสร้าง `ScaleAnimation` สำหรับลดความกว้างเก็บในตัวแปร `shrink` ❶ และ `ScaleAnimation` สำหรับเพิ่มความกว้างเก็บในตัวแปร `grow` ❷ โดยเมื่อแตะหน้าจอจะเรียก `startAnimation(shrink)` เพื่อเริ่ม `shrink` ก่อน ❸ แล้วพอ `shrink` สิ้นสุดก็จะสลับภาพ ❹ แล้วเรียก `startAnimation(grow)` ต่อ ❺

ผลการรัน

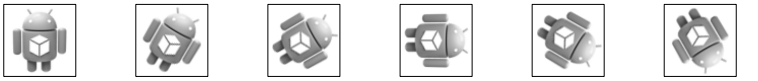


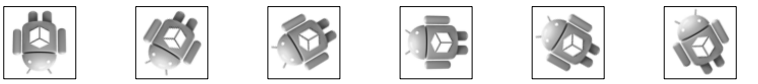
การทำ Animation แบบ Frame-by-Frame

Frame-by-frame Animation (ในเอกสารของแอนดรอยด์เรียกว่า Drawable Animation) คือการแสดงภาพนิ่งต่อเนื่องกันจนดูเป็นภาพเคลื่อนไหว โดยภาพนิ่งเหล่านี้คือออบเจ็ค `Drawable` ซึ่งอาจเป็นไฟล์ภาพ หรือ `Drawable` ชนิดอื่นๆ เช่น Shape Drawable ก็ได้

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะแสดงแอนิเมชั่นเป็นภาพหุ่นแอนดรอยด์หมุน โดยใช้ภาพนิ่งทั้งหมด 12 ภาพมาแสดงต่อเนื่องกัน


frame01.png frame02.png frame03.png frame04.png frame05.png frame06.png


frame07.png frame08.png frame09.png frame10.png frame11.png frame12.png

1 สร้างไฟล์ rotating_android.xml ขึ้นที่โฟลเดอร์ res\drawable แล้วพิมพ์โค้ดดังนี้

โปรเจ็ค AnimationFrameByFrame, ไฟล์ rotating__android.xml

```
<animation-list
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false" >

    <item android:drawable="@drawable/frame01" android:duration="100"/>
    <item android:drawable="@drawable/frame02" android:duration="100"/>
    <item android:drawable="@drawable/frame03" android:duration="100"/>
    <item android:drawable="@drawable/frame04" android:duration="100"/>
    <item android:drawable="@drawable/frame05" android:duration="100"/>
    <item android:drawable="@drawable/frame06" android:duration="100"/>
    <item android:drawable="@drawable/frame07" android:duration="100"/>
    <item android:drawable="@drawable/frame08" android:duration="100"/>
    <item android:drawable="@drawable/frame09" android:duration="100"/>
    <item android:drawable="@drawable/frame10" android:duration="100"/>
    <item android:drawable="@drawable/frame11" android:duration="100"/>
    <item android:drawable="@drawable/frame12" android:duration="100"/>

</animation-list>
```

แท็ก `<animation-list>` ใช้กำหนดภาพและช่วงเวลาของแต่ละเฟรมในแอนิเมชั่น ซึ่งภาพจะถูกแสดงตามลำดับที่กำหนดไว้ภายในแท็กนี้

แอตทริบิวต์ `oneshot` จะกำหนดให้แสดงแอนิเมชั่นครั้งเดียว (ถ้ากำหนดเป็น true) หรือวนซ้ำไปเรื่อยๆ (ถ้ากำหนดเป็น false)

2 กำหนด Layout ของหน้าจอ

โปรเจ็ค AnimationFrameByFrame, ไฟล์ activity__main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true" />

</RelativeLayout>
```

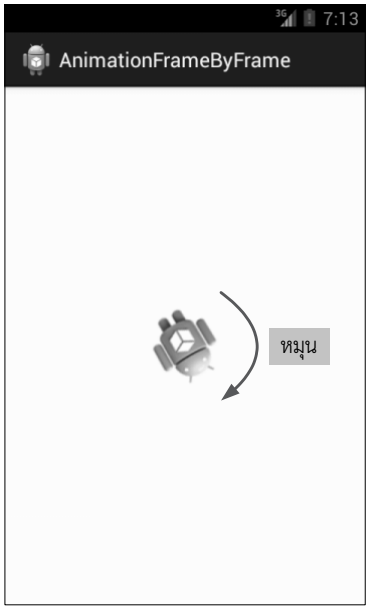
3 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี

โปรเจ็ค AnimationFrameByFrame, ไฟล์ MainActivity.java

```
ImageView image = (ImageView) findViewById(R.id.image);
image.setBackgroundResource(R.drawable.rotating_android); ❶

AnimationDrawable anim = (AnimationDrawable) image.getBackground(); ❷
anim.start(); ❸
```

เรานำไฟล์ rotating_android.xml ที่เก็บข้อกำหนดเกี่ยวกับแอนิเมชั่นมากำหนดเป็นพื้นหลังของ ImageView ❶ จากนั้นเข้าถึงพื้นหลังดังกล่าว ❷ แล้วเรียกเมธอด start เพื่อเริ่มแอนิเมชั่น ❸

ผลการรัน



Property Animation

แอนดรอยด์ 3.0 (API Level 11) ได้เพิ่มแอนิเมชั่นอีกรูปแบบหนึ่งที่เรียกว่า Property Animation ซึ่งเป็นการเปลี่ยนแปลงค่าพร็อพเพอร์ตี้ของออบเจ็คจากค่าหนึ่งไปอีกค่าหนึ่งภายในช่วงเวลาที่กำหนด โดยออบเจ็คที่เราเปลี่ยนแปลงค่าของมันอาจเป็นวิวใน UI หรือออบเจ็คที่ไม่มีรูปร่างหน้าตาก็ได้ ดังนั้น Property Animation จึงไม่ได้จำกัดอยู่เฉพาะการทำให้วัตถุที่มองเห็นได้บนจอเคลื่อนไหวเท่านั้น

การกำหนด Property Animation จะใช้เมธอด `ofFloat`, `ofInt` หรือ `ofObject` ของคลาส `ObjectAnimator` ขึ้นอยู่กับชนิดข้อมูลของพร็อพเพอร์ตี้ที่เราต้องการเปลี่ยนแปลงค่า เช่น

```
ObjectAnimator anim = ObjectAnimator.ofFloat(ออบเจ็ค, ชื่อพร็อพเพอร์ตี้, ค่าเริ่มต้น,
                                                ค่าสิ้นสุด);
```

หรือถ้าต้องการใช้ค่าปัจจุบันเป็นค่าเริ่มต้น ก็ให้กำหนด Property Animation ดังนี้ (เลือกใช้เมธอดให้เหมาะสมกับชนิดข้อมูลด้วย)

```
ObjectAnimator anim = ObjectAnimator.ofFloat(ออบเจ็ค, ชื่อพร็อพเพอร์ตี้, ค่าสิ้นสุด);
```

จากนั้นให้เรียกเมธอด `start` เพื่อเริ่มแอนิเมชั่น

```
anim.start();
```

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะทำแอนิเมชั่นการเพิ่มและลดขนาดตัวอักษรใน `TextView` โดยใช้ Property Animation

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค PropertyAnimation, ไฟล์ activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp" >

    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world"
        android:textSize="12sp" />

</RelativeLayout>
```

2 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี

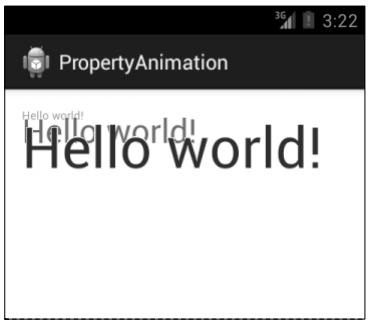
โปรเจ็ค PropertyAnimation, ไฟล์ MainActivity.java

```
TextView text = (TextView) findViewById(R.id.text);

/* กำหนด Property Animation ซึ่งจะเปลี่ยนค่าพร็อพเพอร์ตี้ textSize (ขนาดตัวอักษร) ของ
   TextView จากขนาด 12sp ไปเป็นขนาด 50sp */
ObjectAnimator animResizeText = ObjectAnimator.ofFloat(
                                    text, "textSize", 12f, 50f);
// กำหนดช่วงเวลาของแอนิเมชั่นเป็นครึ่งวินาที
animResizeText.setDuration(500);
// หลังจากสั่ง start แล้วให้รอ 1 วินาทีค่อยเริ่มแอนิเมชั่น
animResizeText.setStartDelay(1000);
// ให้ทำแอนิเมชั่นซ้ำอีก 10 ครั้ง (รวมครั้งแรกด้วยเป็น 11 ครั้ง)
animResizeText.setRepeatCount(10);
/* กำหนดรูปแบบการทำซ้ำเป็นแบบกลับไปกลับมา ในที่นี้คือเมื่อเพิ่มขนาดตัวอักษรจาก 12 เป็น 50 แล้ว
   แอนิเมชั่นครั้งถัดไปจะลดขนาดตัวอักษรจาก 50 ลงมาที่ 12 */
animResizeText.setRepeatMode(ObjectAnimator.REVERSE);
/* กำหนดลักษณะการเปลี่ยนแปลงค่าเป็นแบบ Decelerate Interpolation คือช่วงแรกจะ
   เปลี่ยนแปลงเร็ว หลังจากนั้นอัตราการเปลี่ยนแปลงจะช้าลง */
animResizeText.setInterpolator(new DecelerateInterpolator());
// เริ่มต้นแอนิเมชั่น
animResizeText.start();
```

3 ที่ไฟล์ AndroidManifest.xml ให้แก้ `minSdkVersion` เป็น 11 หรือสูงกว่า

ผลการรัน

เมื่อแอพรันขึ้นมา ข้อความ Hello world! จะใหญ่ขึ้นและเล็กลงสลับกันไป



NOTE

Property Animation จะแก้ไขค่าพร็อพเพอร์ตี้ของออบเจ็คโดยตรง ดังนั้นเมื่อสิ้นสุดแอนิเมชั่น ออบเจ็คจึงมีสถานะตามที่ถูกแก้ไขล่าสุด (ในตัวอย่างนี้คือตัวอักษรจะค้างอยู่ที่ขนาด 50sp) ซึ่งต่างจาก Tweened Animation ที่มีผลแค่ชั่วคราว

สร้าง Animation Set

แอนดรอยด์เตรียมคลาส AnimatorSet ไว้ให้เรากำหนดแอนิเมชั่นหลายๆอย่างที่จะทำไปพร้อมกันหรือต่อเนื่องกัน เราจะเพิ่มแอนิเมชั่นให้กับตัวอย่างเมื่อครู่นี้ โดยหลังจากแอนิเมชั่นเพิ่ม-ลดขนาดตัวอักษรสิ้นสุดแล้ว จะกำหนดให้ข้อความค่อยๆจางหายไป (fade out) จากนั้นเปลี่ยนข้อความเป็นคำว่า “สวัสดี” แล้วค่อยๆแสดงข้อความออกมาอีกครั้ง (fade in)

ให้แก้ไขและเพิ่มเติมโค้ดในเมธอด onCreate ของแอคทิวิตีจนเป็นดังนี้

โปรเจ็ค PropertyAnimation, ไฟล์ MainActivity.java

```
final TextView text = (TextView) findViewById(R.id.text);

ObjectAnimator animResizeText = ObjectAnimator.ofFloat(
                                    text, "textSize", 12f, 50f);
animResizeText.setDuration(500);
animResizeText.setStartDelay(1000);
animResizeText.setRepeatCount(10);
animResizeText.setInterpolator(new DecelerateInterpolator());
animResizeText.setRepeatMode(ObjectAnimator.REVERSE);
//animResizeText.start(); // เปลี่ยนไปเรียกเมธอด start ของ AnimatorSet แทน

/* กำหนด Property Animation ที่เปลี่ยนค่าพร็อพเพอร์ตี้ alpha (ความโปร่ง/ทึบ) ของ TextView
   จาก 1 เป็น 0 ซึ่งจะทำให้ TextView ค่อยๆจางหายไป (fade out) */
ObjectAnimator animFadeOut = ObjectAnimator.ofFloat(text, "alpha", 1f, 0f);
animFadeOut.setDuration(300);
animFadeOut.setRepeatCount(0);
animFadeOut.addListener(new Animator.AnimatorListener() {

    @Override
    public void onAnimationStart(Animator animation) {
    }

    @Override
    public void onAnimationRepeat(Animator animation) {
    }

    // เมื่อสิ้นสุด fade out ให้เปลี่ยนข้อความเป็น “สวัสดี”
    @Override
    public void onAnimationEnd(Animator animation) {
        text.setText("สวัสดี");
    }

    @Override
    public void onAnimationCancel(Animator arg0) {
    }
});
```

```
/* กำหนด Property Animation ที่เปลี่ยนค่าพร็อพเพอร์ตี้ alpha (ความโปร่ง/ทึบ) ของ TextView
   จาก 0 เป็น 1 ซึ่งจะทำให้ TextView ค่อยๆชัดขึ้น (fade in) */
ObjectAnimator animFadeIn = ObjectAnimator.ofFloat(text, "alpha", 0f, 1f);
animFadeIn.setDuration(300);
animFadeIn.setRepeatCount(0);

// กำหนด Animation Set
AnimatorSet animSet = new AnimatorSet();
// ทำ resize text ก่อน fade out
animSet.play(animResizeText).before(animFadeOut);
// ทำ fade in หลัง fade out
animSet.play(animFadeIn).after(animFadeOut);
animSet.start(); // เริ่มต้นแอนิเมชั่น
```