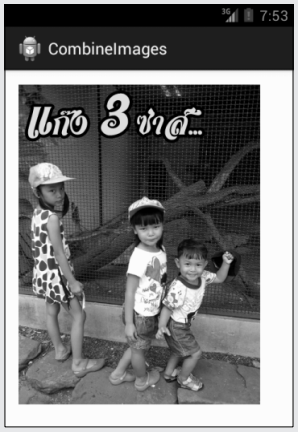
CHAPTER

05 การทำงานกับรูปภาพ

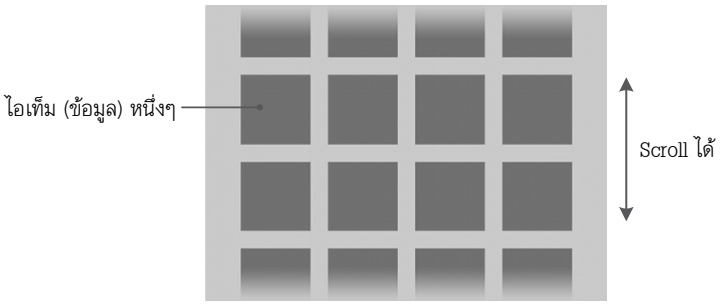
เนื้อหาในบทนี้

- การแสดงรูปภาพด้วย GridView
- การสร้าง Gallery รูปภาพของเราเอง
- การเลือกรูปภาพจาก Gallery ของแอนดรอยด์
- การย่อขนาดรูปภาพ
- การซ้อนรูปภาพ



การแสดงรูปภาพด้วย GridView

GridView คือ วิวที่ใช้แสดงรายการข้อมูลในรูปแบบกริด 2 มิติ โดยผู้ใช้สามารถเลื่อนดูข้อมูลที่แสดงออกมาไม่ครบได้ (scrollable)



จริงๆเราสามารถใช้ GridView แสดงข้อมูลอะไรก็ได้ แต่คุณคงเห็นด้วยว่ารูปแบบของมันเหมาะกับการแสดงรูปภาพเป็นอย่างยิ่ง

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะแสดงรูปภาพต่างๆใน GridView ซึ่งเมื่อคลิกรูปภาพจะแสดง Toast ที่บอกเลขตำแหน่งของรูปภาพนั้นใน GridView

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค GridViewDemo, ไฟล์ res\layout\activity__main.xml

```
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/grid_of_images"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnWidth="100dp" ❶
    android:numColumns="auto_fit" ❷
    android:gravity="center" ❸
    android:stretchMode="columnWidth" ❹ />
```

หน้าจอของแอพนี้จะมี GridView เพียงอย่างเดียวโดยแสดงผลเต็มจอ เรากำหนดความกว้างของแต่ละคอลัมน์ใน GridView เป็น 100dp ❶ และให้มีจำนวนคอลัมน์ตามความเหมาะสม ❷ (ขึ้นอยู่กับขนาดความกว้างของหน้าจอว่าแบ่งได้กี่คอลัมน์) กำหนดให้แสดงข้อมูลตรงกลางช่องของ GridView ❸ และสุดท้ายกำหนดว่าถ้าหน้าจอมีพื้นที่ว่างเหลือ ให้ขยายความกว้างของแต่ละคอลัมน์ออกไปเท่าๆกัน ❹

2 สร้างคลาสใหม่ชื่อ ImageAdapter แล้วพิมพ์โค้ดดังนี้

โปรเจ็ค GridViewDemo, ไฟล์ ImageAdapter.java

```
package com.example.gridviewdemo;

import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;

public class ImageAdapter extends BaseAdapter { ❶

    private Context context;
    private int[] imagesId;

    public ImageAdapter(Context context, int[] imagesId) {
        this.context = context;
        this.imagesId = imagesId;
    }

    @Override
    public int getCount() {
        return imagesId.length;
    }

    @Override
    public Object getItem(int position) {
        return null;
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    @Override
    public View getView(int position, View convertView,
                        ViewGroup parent) {
        ImageView image;

        if (convertView == null) { ❷
            image = new ImageView(context); ❹
```

```
            image.setLayoutParams(new GridView.LayoutParams(150, 150)); ❺
            image.setScaleType(ImageView.ScaleType.CENTER_INSIDE); ❻
            image.setPadding(8, 8, 8, 8); ❼
        } else {
            image = (ImageView) convertView; ❸
        }

        image.setImageResource(imagesId[position]); ❽
        return image; ❾
    }
}
```

คลาส ImageAdapter ข้างต้นคือ Adapter ที่เป็นตัวกลางระหว่าง GridView กับรูปภาพที่เราต้องการแสดงใน GridView เราสร้างคลาสนี้โดยสืบทอดจากคลาส BaseAdapter ❶ ทำให้ต้อง Override เมธอดต่างๆตามที่ BaseAdapter บังคับไว้ ได้แก่ getCount, getItem, getItemId และ getView โดยกติกาก็คือ เมธอด getCount จะต้อง return จำนวนข้อมูลใน Adapter, เมธอด getItem และ getItemId ต้อง return ข้อมูลและ ID ของข้อมูล ตามลำดับ ซึ่งสองเมธอดนี้ไม่ได้ใช้ในตัวอย่างนี้ เราจึง return ค่า null และ 0 ออกไป และสุดท้ายเมธอด getView ต้อง return วิว สำหรับแต่ละข้อมูลที่จะแสดงออกมาใน GridView ดังที่อธิบายในบทที่แล้ว

ภายในเมธอด getView ก่อนอื่นเราตรวจสอบว่ามีวิวที่ใช้ซ้ำได้หรือไม่ ❷ ซึ่งถ้ามี (นั่นคือ convertView ไม่เป็น null) ก็จะใช้งานวิวนั้น ❸ แต่ถ้าไม่มี (convertView เป็น null) ก็จะสร้างออบเจ็ค ImageView ขึ้นใหม่ ❹ จากนั้นกำหนดขนาดเป็น 150x150 พิกเซล ❺, กำหนดวิธีปรับขนาดรูปภาพเป็น CENTER_INSIDE ❻ คือย่อรูปให้แสดงอยู่ภายในและตรงกลาง ImageView และกำหนดระยะห่าง (padding) รอบรูปภาพเป็น 8 พิกเซล ❼ สุดท้ายจึงกำหนดรูปภาพหนึ่งๆให้กับ ImageView ❽ แล้ว return ImageView ออกไป ❾

TIP

ขนาดของ ImageView และระยะ padding ที่กำหนดในโค้ดจาวาข้างต้นจะเป็นหน่วยพิกเซล (Pixel) ซึ่งทำให้ขนาดที่ผู้ใช้มองเห็นแตกต่างกันไปในแต่ละอุปกรณ์ ขึ้นอยู่กับความหนาแน่นจุดภาพ (Pixel Density) ของหน้าจออุปกรณ์นั้นๆ เราสามารถแก้ปัญหานี้ได้โดยตั้งค่าในหน่วย dp ขึ้นมา จากนั้นแปลง dp เป็นพิกเซล แล้วจึงนำค่าพิกเซลที่ได้ไประบุ ก็จะทำให้ได้ขนาดหรือระยะเท่ากันบนทุกอุปกรณ์ ไม่ว่าความหนาแน่นจุดภาพของหน้าจอจะเป็นเท่าใด

การแปลง dp เป็นพิกเซลจะเขียนโค้ดดังนี้

```
int value_in_dp = 8;  // ต้องการกี่ dp ให้ระบุที่บรรทัดนี้
final float scale = getResources().getDisplayMetrics().density;
int value_in_px = (int) (value_in_dp * scale + 0.5f);
// นำค่า value_in_px ไปใช้ ...
```

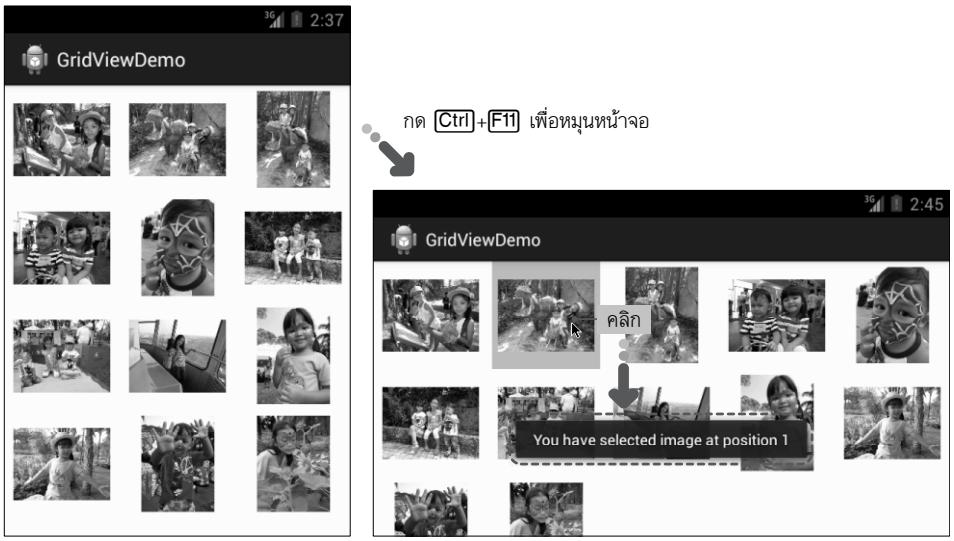
3 ที่แอคทิวิตี ให้เพิ่มการประกาศอาร์เรย์ imagesId และเพิ่มโค้ดในเมธอด onCreate ดังนี้

โปรเจ็ค GridViewDemo, ไฟล์ MainActivity.java

```
public class MainActivity extends Activity {

    // ID ของไฟล์รูปภาพทั้งหมด
    private int[] imagesId = {
            R.drawable.photo01, R.drawable.photo02, R.drawable.photo03,
            R.drawable.photo04, R.drawable.photo05, R.drawable.photo06,
            R.drawable.photo07, R.drawable.photo08, R.drawable.photo09,
            R.drawable.photo10, R.drawable.photo11, R.drawable.photo12 };

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

        GridView grid = (GridView) findViewById(R.id.grid_of_images);

        // สร้าง Adapter (Custom Adapter) แล้วผูกเข้ากับ GridView
        ImageAdapter adapter = new ImageAdapter(this, imagesId);
        grid.setAdapter(adapter);

        // ระบุการทำงานเมื่อไอเท็มใน GridView ถูกคลิก
        grid.setOnItemClickListener(new AdapterView.OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> parent, View v,
                                    int position, long id) {
                String msg = "You have selected image at position ";
                msg += String.valueOf(position);

                Toast.makeText(MainActivity.this, msg, Toast.LENGTH_SHORT)
                     .show();
            }
        });
    }
}
```

ผู้เขียนเตรียมไฟล์รูปภาพ photo01.jpg ถึง photo12.jpg ไว้ในโฟลเดอร์ res\drawable-hdpi ของโปรเจ็ค

ผลการรัน



แสดงรูปภาพที่ถูกคลิกในไดอะล็อก

เราจะทำให้ตัวอย่างนี้ดูน่าสนใจขึ้นอีกนิด โดยแสดงรูปภาพที่ถูกคลิกออกมาในไดอะล็อก แทนที่จะแสดง Toast บอกตำแหน่งของรูปภาพ

ที่เมธอด `onCreate` ของแอคทิวิตี ให้แก้ไขโค้ดที่ระบุการทำงานเมื่อไอเท็มใน `GridView` ถูกคลิกเป็นดังนี้ (ในซอร์สโค้ด ผู้เขียนแยกออกมาเป็นโปรเจ็คใหม่ชื่อ GridViewDemo2)

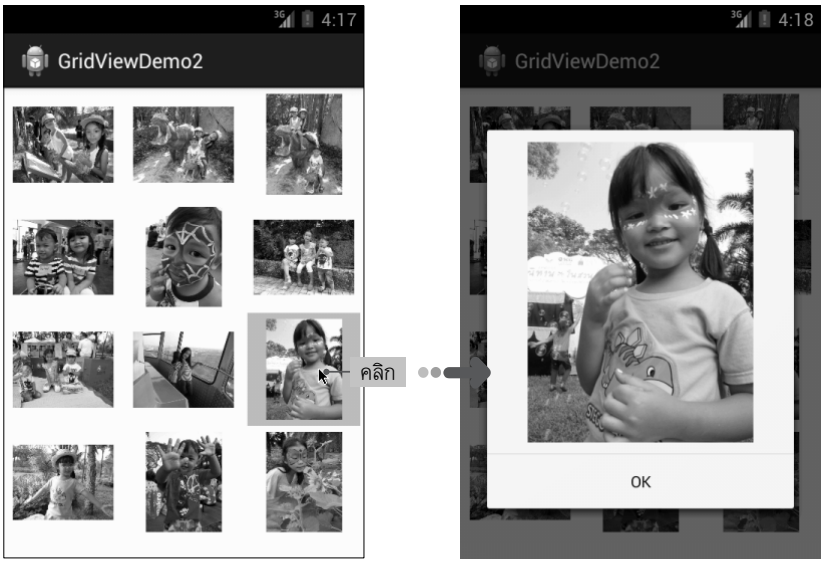
โปรเจ็ค GridViewDemo2, ไฟล์ MainActivity.java, เมธอด onCreate

```
grid.setOnItemClickListener(new AdapterView.OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> parent, View v, int position,
                            long id) {
        ImageView image = new ImageView(MainActivity.this); ❶
        image.setPadding(16, 16, 16, 16);
        image.setImageResource(imagesId[position]); ❷

        new AlertDialog.Builder(MainActivity.this) ❸
           .setView(image) ❹
           .setPositiveButton("OK", new DialogInterface.OnClickListener() {

               @Override
               public void onClick(DialogInterface dialog, int which) {
                   return;
```

```
                }
            })
            .show();
    }
});
```

จากโค้ด เราสร้าง ImageView ขึ้นมาใหม่ ❶ จากนั้นนำค่าของพารามิเตอร์ position ไปใช้เป็น Index เพื่อเข้าถึง ID ของรูปภาพหนึ่งๆในอาร์เรย์ imagesId (รูปภาพที่ถูกคลิกใน GridView) แล้วกำหนดรูปภาพนั้นให้แก่ ImageView ❷

ถัดไปก็สร้างไดอะล็อกขึ้นมา ❸ โดยกำหนด ImageView เป็น Custom Layout ของไดอะล็อก ❹ เพื่อแสดงรูปภาพของ ImageView นั้นออกมาในไดอะล็อก



การสร้าง Gallery รูปภาพของเราเอง

ตัวอย่างนี้เป็นการสร้าง Gallery รูปภาพ (Images Gallery) ที่แสดงภาพขนาดย่อและภาพขนาดเต็มไว้ในหน้าเดียวกัน เพื่อให้คลิกดูรูปภาพต่างๆได้อย่างสะดวก ไม่ต้องคอยปิดไดอะล็อกเหมือนตัวอย่างที่แล้ว

ภาพขนาดย่อทั้งหมดจะแสดงอยู่ใน HorizontalScrollView ที่ด้านบนของหน้าจอ ซึ่งสามารถเลื่อนซ้าย-ขวาเพื่อดูภาพขนาดย่อทั้งหมด และเมื่อคลิกที่ภาพขนาดย่อก็จะแสดงภาพขนาดเต็มออกมาในพื้นที่ว่างด้านล่าง (ลองพลิกไปดูผลการรันก่อน จะเข้าใจโค้ดง่ายขึ้น)

ตัวอย่างและคำอธิบาย

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค ImagesGallery, ไฟล์ res\layout\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" >

    <HorizontalScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#660066"
        android:padding="8dp" >

        <LinearLayout ❶
            android:id="@+id/images_gallery"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:orientation="horizontal" />

    </HorizontalScrollView>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="center" >

        <ImageView
            android:id="@+id/full_size_image"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

    </LinearLayout>

</LinearLayout>
```

แอนดรอยด์กำหนดว่า `HorizontalScrollView` จะมี Child (อิลิเมนต์ลูก) ได้เพียงอิลิเมนต์เดียวเท่านั้น โดยอิลิเมนต์ต่างๆที่ต้องการแสดงใน `HorizontalScrollView` เพื่อให้เลื่อนดูซ้าย-ขวาได้จะต้องบรรจุอยู่ภายใน Child ดังกล่าวอีกที

ในที่นี้สร้าง `LinearLayout` แนวนอน ❶ เป็น Child ของ `HorizontalScrollView` จากนั้นจะใช้โค้ดจาวาเพิ่ม `ImageView` สำหรับแต่ละรูปภาพเข้าไปใน `LinearLayout` ในช่วง runtime เพื่อแสดงรูปภาพเหล่านั้นใน `HorizontalScrollView`

2 เพิ่มการประกาศอาร์เรย์ imagesId และตัวแปรออบเจ็ค imgOldSelected ในแอคทิวิตี

โปรเจ็ค ImagesGallery, ไฟล์ MainActivity.java

```
// ID ของไฟล์รูปภาพทั้งหมด
private int[] imagesId = {
        R.drawable.photo01, R.drawable.photo02, R.drawable.photo03,
        R.drawable.photo04, R.drawable.photo05, R.drawable.photo06,
        R.drawable.photo07, R.drawable.photo08, R.drawable.photo09,
        R.drawable.photo10, R.drawable.photo11, R.drawable.photo12 };

// ภาพขนาดย่อที่ถูกคลิกก่อนหน้า
private ImageView imgOldSelected;
```

แต่ละครั้งที่ภาพขนาดย่อถูกคลิกเลือก เราจะแสดงไฮไลท์รอบภาพนั้น และยกเลิกไฮไลท์ของภาพขนาดย่อที่ถูกคลิกก่อนหน้า ซึ่งเราจะเก็บ reference ของภาพขนาดย่อที่ถูกคลิกก่อนหน้าไว้ในตัวแปร imgOldSelected นี้

3 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี

โปรเจ็ค ImagesGallery, ไฟล์ MainActivity.java

```
// LinearLayout ที่เป็น Child ของ HorizontalScrollView
LinearLayout gallery = (LinearLayout) findViewById(R.id.images_gallery);
// ImageView สำหรับแสดงภาพขนาดเต็ม
final ImageView imgFullSize =
        (ImageView) findViewById(R.id.full_size_image);

for (int id : imagesId) {
    ImageView image = new ImageView(this); ❶ // ภาพขนาดย่อ
    image.setLayoutParams(new LinearLayout.LayoutParams(150, 150));
    image.setScaleType(ImageView.ScaleType.CENTER_INSIDE);
    image.setPadding(8, 8, 8, 8);
    image.setImageResource(id);                                    } ❷

    // ระบุการทำงานเมื่อภาพขนาดย่อถูกคลิก
    image.setOnClickListener(new View.OnClickListener() { ❸

        @Override
        public void onClick(View v) {
            ImageView imgSelected = (ImageView) v; // ภาพขนาดย่อที่ถูกคลิก
            Bitmap bitmap = ((BitmapDrawable) imgSelected.getDrawable())
                            .getBitmap(); ❺
            imgFullSize.setImageBitmap(bitmap); ❻

            if (imgOldSelected != null) {
                imgOldSelected.setBackgroundColor(Color.TRANSPARENT); ❼
            }
```

```
            imgSelected.setBackgroundColor(Color.YELLOW); ❽
            imgOldSelected = imgSelected;
        }
    });

    gallery.addView(image); ❹
}
```

จากโค้ด ซึ่งจะทำงานตอนแอคทิวิตีถูกสร้างขึ้นมานั้น เราสร้าง `ImageView` สำหรับแสดงภาพขนาดย่อของแต่ละภาพที่มี ID อยู่ในอาร์เรย์ `imagesId` ❶ แล้วเพิ่ม `ImageView` เหล่านั้นเข้าไปใน `HorizontalScrollView` ❹ (จริงๆคือเพิ่มเข้าไปใน `LinearLayout` ที่เป็น Child ของ `HorizontalScrollView`) ซึ่ง `ImageView` แต่ละอันจะถูกกำหนดพร็อพเพอร์ตี้ต่างๆ ❷ รวมถึงการกำหนด Click Listener ที่ระบุการทำงานเมื่อมันถูกคลิก ❸

การทำงานของเมธอด `onClick` ใน Click Listener ก็คือ เราจะอ่านข้อมูลภาพ (Bitmap Data) ของภาพขนาดย่อที่ถูกคลิก ❺ แล้วกำหนดให้แก่ `ImageView` ที่ใช้แสดงภาพขนาดเต็ม ❻ จากนั้นยกเลิกไฮไลท์รอบภาพขนาดย่อที่ถูกคลิกในครั้งก่อนหน้า ❼ และแสดงไฮไลท์รอบภาพขนาดย่อที่เพิ่งถูกคลิก ❽

ทั้งนี้การแสดงไฮไลท์รอบภาพจะใช้วิธีกำหนดพื้นหลังเป็นสีเหลือง ซึ่งการมี padding รอบรูปจะทำให้เกิดเป็นไฮไลท์ดังกล่าว ส่วนการยกเลิกไฮไลท์จะทำโดยกำหนดพื้นหลังเป็นสีโปร่งใส

ผลการรัน

การเลือกรูปภาพจาก Gallery ของแอนดรอยด์

แอนดรอยด์มีแอพ Gallery เอาไว้ให้ผู้ใช้ดูรูปภาพต่างๆที่เก็บอยู่ใน SD card ของเครื่อง ข้อดีของแอพนี้สำหรับนักพัฒนาก็คือ ถ้าหากเราต้องการให้ผู้ใช้เลือกรูปภาพจาก SD card เข้ามาในแอพของเรา เราไม่จำเป็นต้องสร้างหน้าจอสำหรับเลือกรูปภาพขึ้นเอง แต่สามารถเรียกใช้ความสามารถของแอพ Gallery ได้เลย

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะแสดงรูปภาพและชื่อพาธของไฟล์ภาพที่ผู้ใช้เลือกจาก Gallery

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค PickImage, ไฟล์ res\layout\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/pick_image_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Pick Image" />

    <TextView
        android:id="@+id/text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:gravity="center"
        android:text="@string/hello_world" />

    <ImageView
        android:id="@+id/image"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:layout_gravity="center"
        android:layout_marginTop="16dp"
        android:scaleType="centerInside"
        android:src="@drawable/ic_launcher" />

</LinearLayout>
```

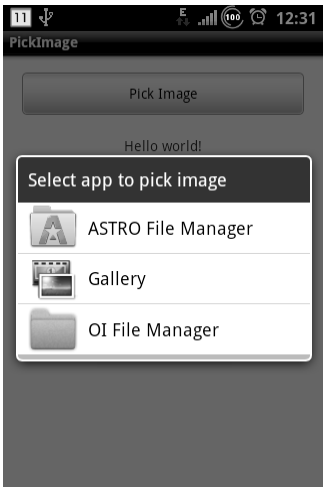
2 ที่แอคทิวิตี ให้เพิ่มตัวแปรระดับคลาส (ฟิลด์ของคลาส) และเพิ่มโค้ดในเมธอด onCreate ดังนี้

โปรเจ็ค PickImage, ไฟล์ MainActivity.java

```
// กำหนดเป็นค่าอะไรก็ได้ (ดูรายละเอียดในหัวข้อ "การส่งข้อมูลกลับไปให้แอคทิวิตีต้นทาง" ในบทที่ 1)
private static final int PICK_IMAGE = 1;

private TextView text;
private ImageView image;

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

    Button button = (Button) findViewById(R.id.pick_image_button);
    text = (TextView) findViewById(R.id.text);
    image = (ImageView) findViewById(R.id.image);

    // ระบุการทำงานเมื่อปุ่มถูกคลิก
    button.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            Intent intent = new Intent();
            intent.setAction(Intent.ACTION_GET_CONTENT);
            intent.setType("image/*");
            startActivityForResult(Intent.createChooser(intent,
                    "Select app to pick image"), PICK_IMAGE);
        }
    });
}
```

เมื่อปุ่มถูกคลิก เราจะเรียกไปยังแอคทิวิตีที่ใช้เลือกรูปภาพซึ่งเป็นส่วนหนึ่งของแอพ Gallery (หรือแอพอื่นๆที่ลงทะเบียนกับระบบว่าสามารถจัดการเรื่องการเลือกรูปภาพได้) โดยการเรียกแอคทิวิตีดังกล่าวต้องใช้เมธอด startActivityForResult เนื่องจากจะมีการส่งข้อมูลเกี่ยวกับรูปภาพที่ถูกเลือกกลับมาให้

เมธอด createChooser จะสร้างหน้าจอที่ให้ผู้ใช้เลือกแอพ (แอคทิวิตี) ที่จัดการเรื่องการเลือกรูปภาพ กรณีมีแอพอื่นๆนอกเหนือจาก Gallery ที่สามารถจัดการงานดังกล่าวได้ ดังรูป



3 เพิ่มเมธอด onActivityResult ในแอคทิวิตี เพื่อรับข้อมูลจากแอคทิวิตีที่ใช้เลือกรูปภาพ

โปรเจ็ค PickImage, ไฟล์ MainActivity.java

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
                                Intent returnedIntent) {
    super.onActivityResult(requestCode, resultCode, returnedIntent);

    switch (requestCode) {
    case PICK_IMAGE:
        if (resultCode == RESULT_OK) {
            Uri imageUri = returnedIntent.getData(); ❶
            String msg = "URI: " + imageUri + "\n";

            String imagePath = findPath(imageUri); ❷
            msg += "Path: " + imagePath;

            text.setText(msg);

            Bitmap imageData = BitmapFactory.decodeFile(imagePath); ❸
            image.setImageBitmap(imageData); ❹
        }
    }
}
```

แอคทิวิตีที่ใช้เลือกรูปภาพจะส่งข้อมูลกลับมาให้ทางพารามิเตอร์ตัวที่สาม (`returnedIntent`) ซึ่งเป็นอินเทนต์ โดยเมธอด `getData` ของอินเทนต์นี้จะให้ผลลัพธ์เป็น URI ของไฟล์ภาพที่ถูกเลือก ❶ เราจะหาชื่อพาธของไฟล์ภาพจาก URI นี้โดยใช้เมธอด `findPath` ที่เราเขียนขึ้นเอง ❷ (รายละเอียดของเมธอดนี้จะอธิบายถัดไป) แล้วอ่านข้อมูลภาพ ❸ มาแสดงใน `ImageView` ที่เตรียมไว้ ❹

4 เพิ่มเมธอด `findPath` ในแอคทิวิตี

โปรเจ็ค PickImage, ไฟล์ MainActivity.java

```
private String findPath(Uri uri) {
    String imagePath;

    String[] columns = { MediaStore.Images.Media.DATA };
    Cursor cursor = getContentResolver().query(uri, columns, null,
                                                null, null); ❶
    if (cursor != null) { // กรณีของแอพ Gallery
        cursor.moveToFirst();
        int columnIndex = cursor.getColumnIndexOrThrow(
                                MediaStore.Images.Media.DATA);
        imagePath = cursor.getString(columnIndex); ❷
    }
    else { // กรณีของแอพอื่นๆ เช่น OI File Manager, ASTRO File Manager
        imagePath = uri.getPath(); ❸
    }

    return imagePath;
}
```

การหาพาธของไฟล์ภาพจาก URI จะแยกเป็น 2 กรณี คือกรณีที่ใช้แอพ Gallery เลือกรูปภาพกับกรณีใช้แอพอื่นๆ (ถ้ามีแอพเหล่านั้นอยู่ในเครื่อง) เลือกรูปภาพ เนื่องจาก 2 กรณีนี้จะให้ค่า URI ของไฟล์ภาพที่มีรูปแบบแตกต่างกัน

กรณีของแอพ Gallery เราจะหาชื่อพาธของไฟล์ภาพโดยการคิวรี Content Provider ❶ ซึ่งชื่อพาธเก็บอยู่ในคอลัมน์ `MediaStore.Images.Media.DATA` ของ Provider เราจึงคิวรีคอลัมน์นี้มาเพียงคอลัมน์เดียว (ถ้าคุณต้องการข้อมูลอื่นๆเกี่ยวกับไฟล์ภาพ เช่น สถานที่หรือวันเวลาที่ภาพถูกถ่ายก็สามารถระบุคอลัมน์อื่นๆเพิ่มได้ แต่ในที่นี้จะสนใจเฉพาะชื่อพาธ) แล้วอ่านข้อมูล (ชื่อพาธ) จากคอลัมน์นั้นมาเก็บลงตัวแปร `imagePath` ❷

กรณีของแอพอื่นๆ เช่น OI File Manager และ ASTRO File Manager เมื่อคิวรี Content Provider จะได้ผลลัพธ์เป็น `null` กลับมา (ตัวแปร `cursor` มีค่า `null`) และเราจะหาชื่อพาธของไฟล์ภาพโดยเรียกเมธอด `getPath` บน URI แทน ❸

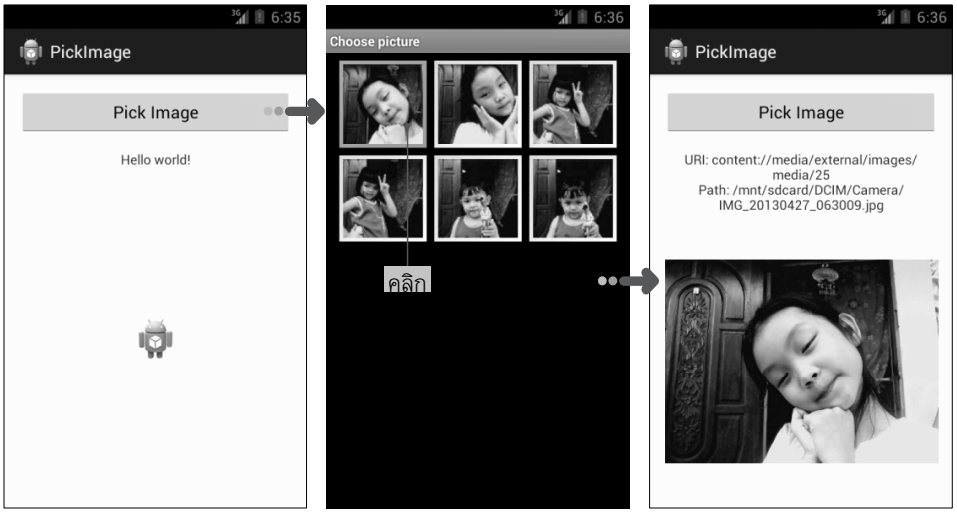
TIP

จริงๆถ้าไม่ต้องการรู้พาธของไฟล์ภาพ เราสามารถอ่านข้อมูลภาพจาก URI ได้เลย โดยเขียนโค้ดดังนี้ (เท่าที่ผู้เขียนทดสอบ โค้ดนี้ใช้ได้ไม่ว่าจะเลือกรูปภาพผ่านแอพใดก็ตาม)

```
InputStream imageStream;
try {
    imageStream = getContentResolver().openInputStream(imageUri); ❶
    Bitmap imageData = BitmapFactory.decodeStream(imageStream); ❷
    image.setImageBitmap(imageData); ❸
} catch (FileNotFoundException e) {
    e.printStackTrace();
}
```

เมธอด openInputStream จะรับ URI ของไฟล์ภาพเข้ามาเป็นพารามิเตอร์ แล้วส่งคืนออบเจ็ค InputStream ออกไป ❶ จากนั้นจึงใช้เมธอด decodeStream แปลง InputStream ไปเป็นออบเจ็ค Bitmap ❷ แล้วแสดงผลใน ImageView ❸

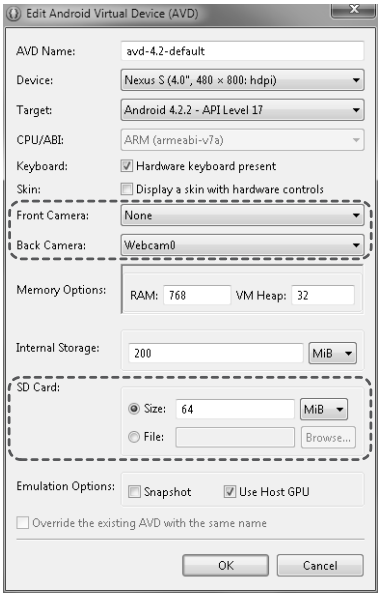
ผลการรัน



การทดสอบแอพนี้บนอีมูเลเตอร์ คุณจะต้องจำลอง SD Card แล้วก๊อปปี้ไฟล์ภาพจำนวนหนึ่งไปเก็บไว้ หรือหากเครื่องพีซีของคุณมีกล้องเว็บแคม ก็อาจจำลองกล้องหน้าหรือกล้องหลังให้กับอีมูเลเตอร์โดยใช้กล้องเว็บแคมของพีซี แล้วใช้แอพ Camera ในอีมูเลเตอร์ถ่ายภาพเก็บลง SD Card เพื่อจะได้มีรูปภาพใน SD Card ให้เลือกได้

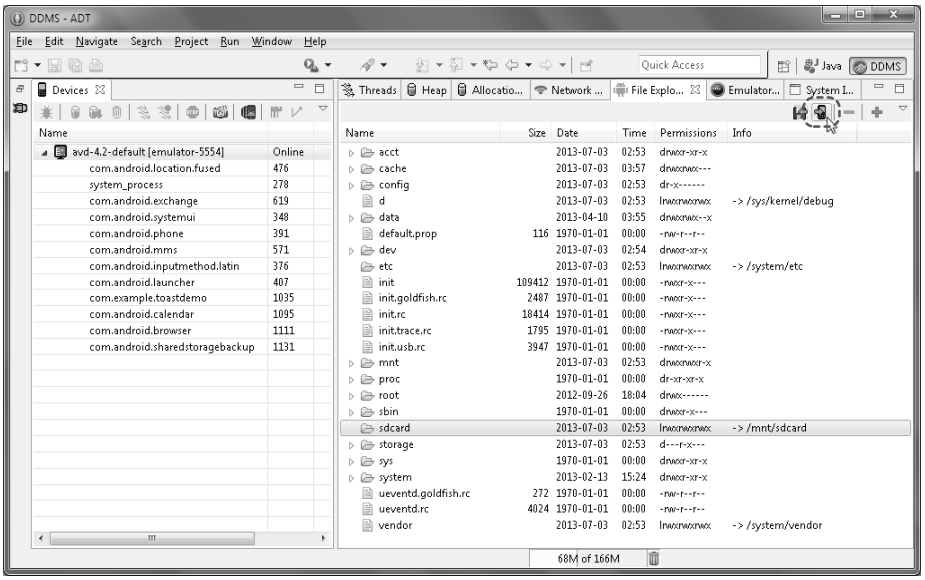
การจำลอง SD Card และกล้องให้กับอีมูเลเตอร์

ให้เปิดคุณสมบัติของอีมูเลเตอร์ขึ้นมาแก้ไขตามรูป จากนั้นให้ปิดแล้วรันอีมูเลเตอร์ใหม่



การก๊อปปี้ไฟล์ภาพไปยัง SD Card ของอีมูเลเตอร์

ไปที่แท็บ File Explorer ใน DDMS Perspective, คลิกไดเร็คทอรี sdcard, คลิกปุ่ม Push a file onto the device บนทูลบาร์ แล้วเลือกไฟล์จากเครื่องพีซีที่ต้องการก๊อปปี้ลงอีมูเลเตอร์



การย่อขนาดรูปภาพ

เดี๋ยวนี้กล้องในมือถือแอนดรอยด์รุ่นใหม่ๆมีความละเอียดสูงมาก ทำให้ภาพถ่ายมีขนาดไฟล์ใหญ่ตามไปด้วย ดังนั้นการให้ผู้ใช้เลือกไฟล์ภาพจาก SD Card เข้ามาในแอพจึงอาจเกิดปัญหาหน่วยความจำไม่พอได้ ซึ่งเราสามารถแก้ปัญหานี้โดยการย่อขนาดของภาพก่อนนำมาใช้งาน เช่น ก่อนแสดงออกมาใน ImageView เป็นต้น

NOTE

การกำหนดแอตทริบิวต์ scaleType ของ ImageView เพื่อย่อภาพขนาดใหญ่ให้แสดงภายใน ImageView ได้นั้น ไม่ได้ทำให้ข้อมูลภาพ (ข้อมูลพิกเซล) มีปริมาณน้อยลง ดังนั้นจึงไม่ได้ลดการใช้หน่วยความจำของเครื่อง

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะอ่านไฟล์ภาพ family.jpg จาก SD Card ของเครื่องมาแสดงใน ImageView จากนั้นเมื่อคลิกปุ่มก็จะย่อภาพ แล้วแสดงใน ImageView อีกอันหนึ่ง

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค ResizeImage, ไฟล์ res\layout\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" >

    <ImageView
        android:id="@+id/source_image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" />

    <Button
        android:id="@+id/resize_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginTop="8dp"
        android:text="Resize" />

    <ImageView
        android:id="@+id/resized_image"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" />

</LinearLayout>
```

2 ที่แอคทิวิตี ให้เพิ่มตัวแปรระดับคลาส และเพิ่มโค้ดในเมธอด onCreate ดังนี้

โปรเจ็ค ResizeImage, ไฟล์ MainActivity.java

```
ImageView sourceImage;    // แสดงภาพต้นฉบับ
ImageView resizedImage;   // แสดงภาพที่ย่อขนาดแล้ว
private File sdCardRoot;  // รูทไดเร็คทอรีของ SD Card
private File file;        // ไฟล์ภาพต้นฉบับ

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

    sourceImage = (ImageView) findViewById(R.id.source_image);
    sdCardRoot = Environment.getExternalStorageDirectory(); ❶
    file = new File(sdCardRoot, "family.jpg"); ❷

    Bitmap sourceBitmap = BitmapFactory
                          .decodeFile(file.getAbsolutePath()); ❸
    sourceImage.setImageBitmap(sourceBitmap); ❹

    resizedImage = (ImageView) findViewById(R.id.resized_image);
    Button button = (Button) findViewById(R.id.resize_button);
    button.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            Bitmap resizedBitmap = decodeAndResize(file); ❺
            resizedImage.setImageBitmap(resizedBitmap); ❻
        }
    });
}
```

เมื่อแอคทิวิตีรันขึ้นมา เราจะหารูทไดเร็คทอรีของ SD Card ❶ จากนั้นสร้างออบเจ็ค File ที่เป็นตัวแทนของไฟล์ family.jpg ในไดเร็คทอรีดังกล่าว ❷ แล้วอ่านข้อมูลภาพจากไฟล์ ❸ มาแสดงใน ImageView ❹

เมื่อปุ่มถูกคลิก เราจะเรียกเมธอด decodeAndResize ที่เราเขียนขึ้นเอง (รายละเอียดของเมธอดนี้จะอธิบายถัดไป) เพื่ออ่านข้อมูลภาพจากไฟล์อีกครั้งพร้อมทั้งย่อขนาดภาพลงด้วย ❺ แล้วแสดงภาพใน ImageView อีกอันหนึ่ง ❻

3 เพิ่มเมธอด decodeAndResize ในแอคทิวิตี

โปรเจ็ค ResizeImage, ไฟล์ MainActivity.java

```
private Bitmap decodeAndResize(File file) {
    Bitmap bitmap = null;
    try {
        BitmapFactory.Options option = new BitmapFactory.Options(); ❶
        option.inSampleSize = 2; ❷
        FileInputStream fis = new FileInputStream(file);
        bitmap = BitmapFactory.decodeStream(fis, null, option); ❸
        fis.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return bitmap;
}
```

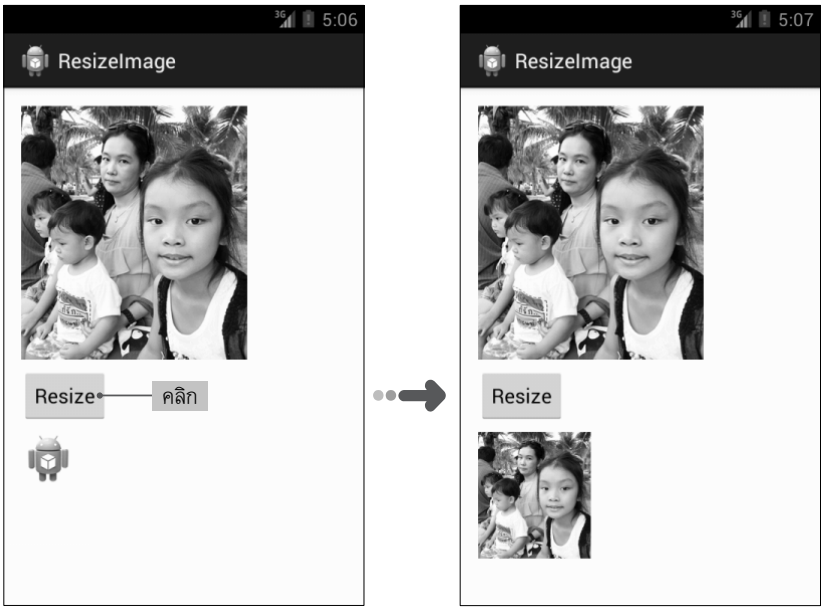
เราออกแบบเมธอดนี้ให้รับพารามิเตอร์เป็นออบเจ็ค File ของไฟล์ภาพต้นฉบับเข้ามา แล้ว return ออบเจ็ค Bitmap ซึ่งเป็นข้อมูลภาพที่ย่อขนาดแล้วกลับออกไป

วิธีย่อขนาดภาพที่ใช้ในตัวอย่างนี้ เป็นการย่อขณะแปลงสตรีมข้อมูลของไฟล์ไปเป็นข้อมูลภาพ ซึ่งทำได้โดยสร้างออบเจ็ค BitmapFactory.Options ❶ แล้วกำหนดอัตราส่วนในการย่อภาพที่แอตทริบิวต์ inSampleSize ของออบเจ็คดังกล่าว ❷ ในที่นี้กำหนดค่า 2 หมายถึงให้ลดขนาดความกว้างและความสูงของภาพเหลือครึ่งหนึ่งของขนาดเดิม (นำขนาดเดิมมาหารด้วย 2) จากนั้นจึงนำออบเจ็ค BitmapFactory.Options ไประบุเป็นพารามิเตอร์ของเมธอด decodeStream ที่ใช้แปลงสตรีมข้อมูลของไฟล์ไปเป็นข้อมูลภาพ ❸ ก็จะได้ข้อมูลภาพที่มีความกว้างและความสูงลดลงตามต้องการ

ผลการรัน

ก่อนรันแอพนี้ ให้คุณหาไฟล์ภาพ JPEG มาไฟล์หนึ่ง เปลี่ยนชื่อเป็น family.jpg (หรือแก้ชื่อไฟล์ในโค้ดให้ตรงกับชื่อไฟล์ JPEG นั้นก็ได้) แล้วใส่ลงในไดเร็คทอรี /mnt/sdcard ของอีมูเลเตอร์ ตามวิธีที่อธิบายในตัวอย่างที่แล้ว

แนะนำว่าไฟล์ภาพที่ใช้ทดสอบไม่ควรมีขนาดใหญ่เกินไป เพราะภาพอาจจะแสดงผลล้นจอจนทำให้ภาพที่ย่อขนาดแล้วและอาจรวมถึงปุ่ม Resize ตกหน้าจอไป (ในรูปข้างล่างนี้ผู้เขียนกำหนดหน้าจออีมูเลเตอร์เป็นขนาด 480x800 พิกเซล และใช้ไฟล์ภาพขนาด 300x338 พิกเซล)



บันทึกรูปภาพที่ย่อขนาดแล้วลง SD Card

หลังจากย่อภาพแล้ว คุณอาจต้องการบันทึกภาพนั้นเก็บเป็นไฟล์ใน SD Card ของเครื่อง ซึ่งทำได้โดยเพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตีดังนี้

โปรเจ็ค ResizeImage, ไฟล์ MainActivity.java

```
try {
    File outFile = new File(sdCardRoot, "family_resized.jpg");
    FileOutputStream fos = new FileOutputStream(outFile);
    resizedBitmap.compress(Bitmap.CompressFormat.JPEG, 100, fos);
    fos.close();

    String msg = "File saved to SD card.";
    Toast.makeText(MainActivity.this, msg, Toast.LENGTH_SHORT).show();
} catch (IOException e) {
    e.printStackTrace();
}
```

โค้ดข้างต้นจะบันทึกข้อมูลภาพที่ย่อขนาดแล้วเป็นไฟล์ชื่อ family_resized.jpg ในรูทไดเร็คทอรีของ SD Card หลังจากนั้นจะแสดง Toast บอกให้รู้ว่าบันทึกไฟล์แล้ว

การเขียนไฟล์ลงใน SD Card จะต้องขอสิทธิ์ WRITE_EXTERNAL_STORAGE จากแอนดรอยด์ก่อน มิฉะนั้นจะเกิดข้อผิดพลาดตอนรันแอพ ให้คุณเพิ่มบรรทัดนี้ในไฟล์ AndroidManifest.xml โดยใส่ไว้ก่อนแท็ก <application>

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

การซ้อนรูปภาพ

คุณผู้อ่านคงรู้จักแอพติกเกอร์ (Ticker) แอพที่ให้เราใส่ข้อความกวนๆลงในรูปภาพแล้วแชร์ไปยังโซเชียลเน็ตเวิร์กได้ แน่นอนว่าการพัฒนาแอพนี้ต้องใช้เทคนิคการซ้อนภาพอย่างไม่ต้องสงสัย ซึ่งถ้าคุณอยากรู้ว่าการซ้อนภาพทำได้อย่างไร หัวข้อนี้มีคำตอบครับ

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะอ่านไฟล์ภาพ main.jpg และ sticker.png จาก SD Card จากนั้นนำภาพทั้งสองมาซ้อนกันโดยให้ภาพจาก sticker.png อยู่ข้างบน แล้วแสดงภาพผลลัพธ์ออกมาใน ImageView

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค CombineImages, ไฟล์ res\layout\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" >

    <ImageView
        android:id="@+id/result_image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

2 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี

โปรเจ็ค CombineImages, ไฟล์ MainActivity.java

```
// หารูทไดเร็คทอรีของ SD Card
File sdCardRoot = Environment.getExternalStorageDirectory();

// อ่านข้อมูลภาพจากไฟล์ main.jpg แล้วสร้างสำเนาของข้อมูลภาพนั้นขึ้นมา (เพื่อให้แก้ไขได้)
File file = new File(sdCardRoot, "main.jpg");
Bitmap mainBitmap = BitmapFactory.decodeFile(file.getAbsolutePath())
                    .copy(Bitmap.Config.ARGB_8888, true);
```

```
// อ่านข้อมูลภาพจากไฟล์ sticker.png
file = new File(sdCardRoot, "sticker.png");
Bitmap stickerBitmap = BitmapFactory.decodeFile(file.getAbsolutePath());

// สร้าง Canvas ขึ้นมาครอบ mainBitmap
Canvas canvas = new Canvas(mainBitmap);
// วาด stickerBitmap ลงไปบน mainBitmap โดยผ่านทาง Canvas
canvas.drawBitmap(stickerBitmap, 0f, 0f, null);

// แสดง mainBitmap ใน ImageView
ImageView image = (ImageView) findViewById(R.id.result_image);
image.setImageBitmap(mainBitmap);
```

หลักการซ้อนภาพในตัวอย่างนี้คือ เราจะอ่านข้อมูลภาพจากไฟล์ทั้งสอง (ไฟล์ภาพหลัก และไฟล์ภาพสติกเกอร์) มาเก็บในออบเจ็ค Bitmap จากนั้นสร้าง Canvas ขึ้นมาครอบภาพหลัก แล้ววาดภาพสติกเกอร์ลงไปบนภาพหลักโดยผ่านทาง Canvas นั้น

สำหรับไฟล์ภาพหลักเมื่อใช้เมธอด decodeFile อ่านข้อมูลภาพแล้ว จะต้องใช้เมธอด copy สร้างสำเนาของข้อมูลนั้นขึ้นมาด้วยจึงจะทำให้เราสามารถแก้ไขภาพได้ อย่างไรก็ตาม การแก้ไขภาพในที่นี้คือการแก้ไขข้อมูลภาพในหน่วยความจำ ซึ่งไม่มีผลกับไฟล์ภาพที่เก็บอยู่ใน SD Card

3 ถ้าหากต้องการจัดเก็บภาพผลลัพธ์เป็นไฟล์ใน SD Card ให้เพิ่มโค้ดใน onCreate ดังนี้

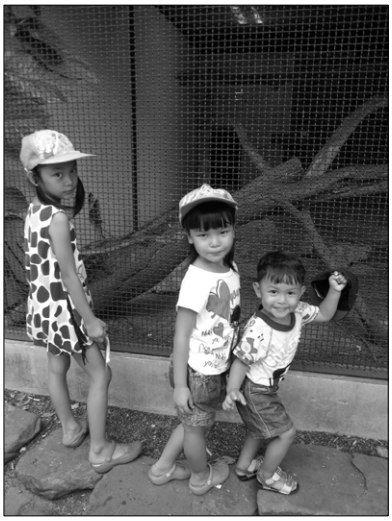
โปรเจ็ค CombineImages, ไฟล์ MainActivity.java

```
try {
    File outFile = new File(sdCardRoot, "result.jpg");
    FileOutputStream fos = new FileOutputStream(outFile);
    mainBitmap.compress(Bitmap.CompressFormat.JPEG, 100, fos);
    fos.close();

    String msg = "File saved to SD card.";
    Toast.makeText(MainActivity.this, msg, Toast.LENGTH_SHORT).show();
} catch (IOException e) {
    e.printStackTrace();
}
```

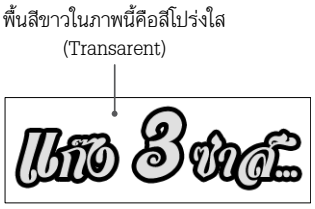
และอย่าลืมเพิ่มการขอสิทธิ์ในไฟล์ AndroidManifest.xml ด้วย (ดูหัวข้อที่แล้ว)

ผลการรัน

ก่อนรันแอพ ให้คุณเตรียมไฟล์ภาพ main.jpg และ sticker.png ในไดเร็คทอรี /mnt/sdcard ของอีมูเลเตอร์ ในที่นี้ผู้เขียนเตรียมไฟล์ภาพดังรูป

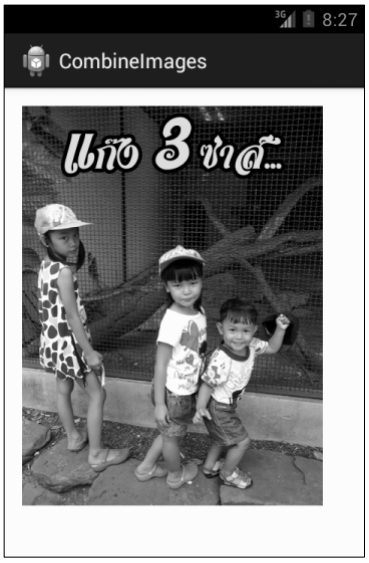


main.jpg



sticker.png

เมื่อรันแอพจะได้ผลลัพธ์ดังรูปถัดไปทางซ้าย



ระบุตำแหน่งในการซ้อนภาพ

ตอนเรียกเมธอด drawBitmap ของ Canvas เพื่อวาดภาพสติกเกอร์ลงบนภาพหลัก เราสามารถระบุตำแหน่งในการวาดได้โดยกำหนดพารามิเตอร์ตัวที่ 2 และ 3 ของ drawBitmap

```
canvas.drawBitmap(stickerBitmap, x, y, null);
```

ยกตัวอย่างเช่น ถ้าต้องการวาดภาพสติกเกอร์ให้อยู่กึ่งกลางในแนวนอนของภาพหลัก จะเขียนโค้ดดังนี้

```
int mainWidth = mainBitmap.getWidth();          // หาความกว้างของภาพหลัก
int stickerWidth = stickerBitmap.getWidth();    // หาความกว้างของภาพสติกเกอร์

// หาตำแหน่งการวาดในแนวนอน (x) ที่จะทำให้ภาพสติกเกอร์อยู่ตรงกลางภาพหลักพอดี
float xPosition = (mainWidth - stickerWidth) / 2;

canvas.drawBitmap(stickerBitmap, xPosition, 0f, null);
```

เมื่อรันก็จะได้ผลลัพธ์ดังรูปบนทางขวา