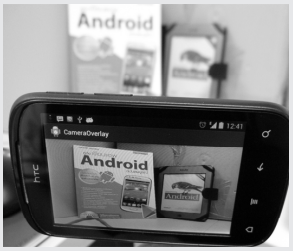
CHAPTER

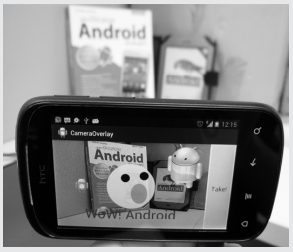
08 การทำงานกับกล้อง

เนื้อหาในบทนี้

- การถ่ายภาพ
- การถ่ายภาพแล้วบันทึกลงไฟล์
- การแสดงภาพพรีวิวจากกล้องออกมาในแอพของเรา



- การวาดกราฟิกซ้อนบนภาพพรีวิวจากกล้อง



การถ่ายภาพ

อุปกรณ์แอนดรอยด์ส่วนใหญ่มีกล้องอยู่ในตัว เราสามารถเรียกใช้ความสามารถของกล้องเพื่อนำภาพถ่ายเข้ามาในแอพของเราได้

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะมีปุ่มซึ่งเมื่อคลิกจะเรียกแอพกล้องขึ้นมา และหลังจากกดถ่ายภาพแล้วจะแสดงภาพที่ได้ใน ImageView

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค CameraImage, 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/capture_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Take a Picture" />

    <ImageView
        android:id="@+id/image"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="center" />

</LinearLayout>
```

2 ที่แอคทิวิตี ให้เพิ่มการประกาศค่าคงที่ TAKE_PICTURE ที่ส่วนประกาศของคลาส, เพิ่มโค้ดในเมธอด onCreate และเพิ่มเมธอด onActivityResult

โปรเจ็ค CameraImage, ไฟล์ MainActivity.java

```
package com.example.cameraimage;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
```

```
import android.widget.Button;
import android.widget.ImageView;

public class MainActivity extends Activity {

    private static final int TAKE_PICTURE = 100;

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

        Button btnCapture = (Button) findViewById(R.id.capture_button);
        btnCapture.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                Intent intent = new Intent(
                        MediaStore.ACTION_IMAGE_CAPTURE); ❶
                startActivityForResult(intent, TAKE_PICTURE); ❷
            }
        });
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
                                    Intent data) {
        ImageView image = (ImageView) findViewById(R.id.image);

        if (requestCode == TAKE_PICTURE && resultCode == RESULT_OK) {
            Bitmap capturedImage = (Bitmap)
                                   data.getExtras().get("data");   } ❸
            image.setImageBitmap(capturedImage);
        }
    }
}
```

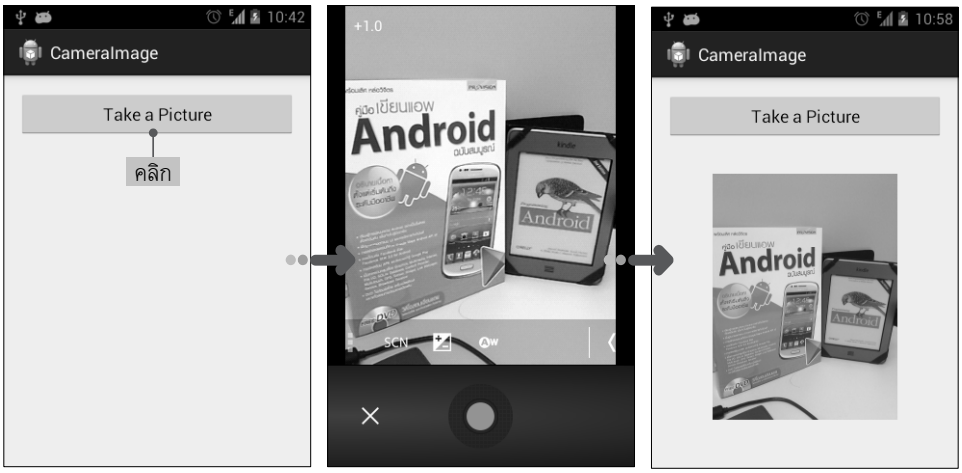
เมื่อปุ่มถูกคลิก เราจะสร้างอินเทนต์ที่เป็นแอคชั่น MediaStore.ACTION_IMAGE_CAPTURE ❶ แล้วรันแอคทิวิตีที่สามารถจัดการอินเทนต์นี้ได้ ❷ ซึ่งโดยทั่วไปก็คือแอพกล้องทั้งหลาย (ถ้ามีแอพกล้องมากกว่า 1 แอพในเครื่อง แอนดรอยด์จะให้ผู้ใช้เลือกแอพที่ต้องการ)

และเช่นเคย การรันแอคทิวิตีด้วยเมธอด `startActivityForResult` นั้น จะต้อง Override เมธอด `onActivityResult` เพื่อรับผลลัพธ์ที่แอคทิวิตีปลายทาง (แอพกล้อง) ส่งกลับมาให้ ในที่นี้ผลลัพธ์ก็คือ ข้อมูลภาพถ่ายซึ่งจะถูกเก็บอยู่ในฟิลด์ data ของข้อมูลเพิ่มเติมภายในอินเทนต์ที่ส่งกลับมา เราจะนำข้อมูลภาพนี้มาแสดงใน `ImageView` โดยใช้เมธอด `setImageBitmap` ❸

ผลการรัน

การรันแอพนี้บนอีมูเลเตอร์ ก่อนอื่นต้องจำลองกล้องหน้าและ/หรือกล้องหลัง โดยเข้าไปยังหน้าจอที่ใช้แก้ไขคุณสมบัติของอีมูเลเตอร์ แล้วกำหนดตัวเลือกที่หัวข้อ Front Camera (กล้องหน้า) และ/หรือ Back Camera (กล้องหลัง) โดยหากเครื่องพีซีของคุณมีกล้องเว็บแคมก็ให้กำหนดหัวข้อใดหัวข้อหนึ่งหรือทั้งสองหัวข้อเป็น Webcam เพื่อใช้ภาพจากกล้องเว็บแคมนั้น แต่หากไม่มีกล้องเว็บแคมก็ให้กำหนดเป็น Emulated ซึ่งอีมูเลเตอร์จะจำลองภาพขึ้นมาเอง หรือไม่ก็รันแอพบนเครื่องจริงที่มีกล้องไปเลย

รูปต่อไปนี้แสดงผลการรันบนเครื่องจริง



การถ่ายภาพแล้วบันทึกลงไฟล์

การถ่ายภาพในตัวอย่างที่ผ่านมา ข้อมูลภาพที่แอพกล้องส่งกลับมาให้ทางอินเทนต์จะเป็นภาพขนาดย่อเท่านั้น แต่หากคุณต้องการภาพขนาดเต็มจะต้องสั่งแอพกล้องให้บันทึกภาพลงไฟล์แทน แล้วค่อยอ่านข้อมูลจากไฟล์มาใช้งานอีกที

ตัวอย่างและคำอธิบาย

ให้ทำต่อจากตัวอย่างที่แล้ว

1 เพิ่มปุ่มใน Layout อีกปุ่มหนึ่งถัดจากปุ่มที่สร้างไว้แล้ว

โปรเจ็ค CameraImage, ไฟล์ activity__main.xml

```
<Button
    android:id="@+id/capture_and_save_button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Take a Picture and Save to File" />
```

2 ที่แอคทิวิตี ให้เพิ่มการประกาศค่าคงที่และตัวแปรที่ส่วนประกาศของคลาส

โปรเจ็ค CameraImage, ไฟล์ MainActivity.java

```
private static final int TAKE_PICTURE_SAVE = 101;
private File imageFile;
```

3 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี เพื่อกำหนด Listener ให้กับปุ่มที่สร้างขึ้นใหม่

โปรเจ็ค CameraImage, ไฟล์ MainActivity.java

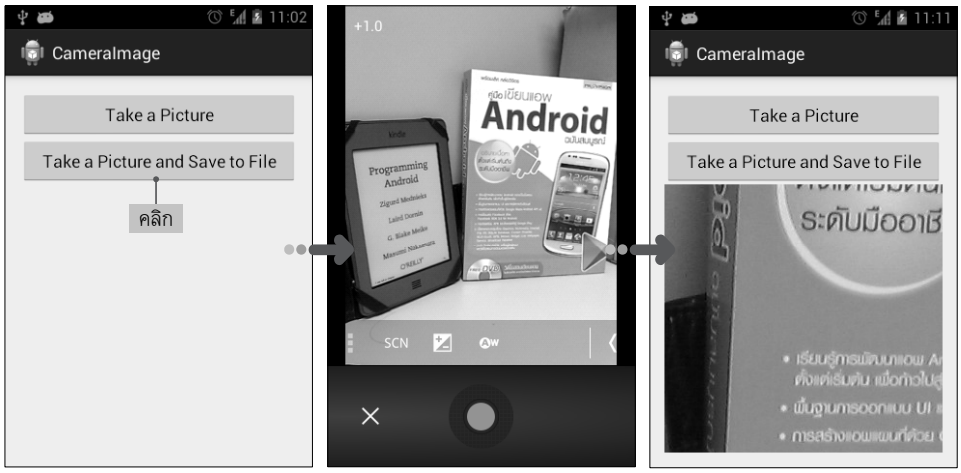
```
Button btnCaptureSave = (Button) findViewById(
                                    R.id.capture_and_save_button);
btnCaptureSave.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        // สร้างไฟล์ my_image.jpg ขึ้นที่ SD Card
        imageFile = new File(Environment.getExternalStorageDirectory(),
                             "my_image.jpg");
        // ส่ง URI ของไฟล์ my_image.jpg ไปให้แอพกล้องทางอินเทนต์
        intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(imageFile));
        startActivityForResult(intent, TAKE_PICTURE_SAVE);
    }
});
```

4 ในเมธอด onActivityResult ให้เพิ่ม else if ต่อท้ายคำสั่ง if เดิม เพื่อระบุการทำงานหลังจากถ่ายภาพแล้ว

โปรเจ็ค CameraImage, ไฟล์ MainActivity.java

```
if (requestCode == TAKE_PICTURE && resultCode == RESULT_OK) {
    Bitmap capturedImage = (Bitmap) data.getExtras().get("data");
    image.setImageBitmap(capturedImage);
} else if (requestCode == TAKE_PICTURE_SAVE && resultCode == RESULT_OK) {
    try {
        // สร้าง FileInputStream ขึ้นจากไฟล์ภาพ
        FileInputStream fis = new FileInputStream(imageFile);
        // แปลงเป็น Bitmap
        Bitmap fullSizeImage = BitmapFactory.decodeStream(fis);
        // แสดงออกมาที่ ImageView
        image.setImageBitmap(fullSizeImage);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
}
```

ผลการรัน



จะเห็นว่า ImageView แสดงได้เพียงแค่บางส่วนของภาพ เนื่องจากภาพมีขนาดใหญ่มาก และเรากำหนด scaleType ของ ImageView ไว้เป็น center หมายถึงให้แสดงภาพตรงกลาง ImageView โดยไม่ต้องมีการย่อหรือขยายภาพ (ถ้ากำหนดเป็น centerInside ภาพจะถูกย่อให้แสดงอยู่ภายใน ImageView แต่เป็นการย่อตอนแสดงผล ไม่ได้ย่อข้อมูลภาพที่อ่านมาจากไฟล์)

TIP

คุณสามารถย่อภาพเพื่อให้ข้อมูลภาพมีขนาดเล็กลงก่อนแสดงออกมาใน ImageView ได้ โดยแก้โค้ดในส่วน else if เป็นดังนี้

```
else if (requestCode == TAKE_PICTURE_SAVE && resultCode == RESULT_OK) {
    try {
        FileInputStream fis = new FileInputStream(imageFile);

        BitmapFactory.Options options = new BitmapFactory.Options();
        // ย่อขนาดภาพลง 8 เท่าทั้งด้านกว้างและด้านสูง (จำนวนพิกเซลน้อยลง 64 เท่า)
        options.inSampleSize = 8;
        Bitmap fullSizeImage = BitmapFactory.decodeStream(fis, null,
                                                          options);
        image.setImageBitmap(fullSizeImage);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
}
```

การแสดงภาพพรีวิวจากกล้องออกมาในแอพของเรา

ใน 2 หัวข้อที่ผ่านมาเป็นการถ่ายภาพโดยใช้อินเทนต์เรียกแอพกล้องขึ้นมา แต่นอกจากนี้แอนดรอยด์ยังเตรียม Camera API ไว้ให้เราเข้าถึงกล้องโดยตรง ทำให้เราสามารถควบคุมการพรีวิวภาพ การถ่ายภาพ และการตั้งค่าที่เกี่ยวกับกล้องได้

ตัวอย่างและคำอธิบาย

การแสดงภาพพรีวิวจากกล้องออกมาในแอพของเราเอง จะทำได้โดยใช้ SurfaceView ซึ่งเป็นวิวที่ถูกสร้างมาให้เป็นพื้นผิวสำหรับวาดภาพ (Drawing Surface) โดยเฉพาะ และเนื่องจากมันเป็นวิวชนิดหนึ่ง เราจึงสามารถนำไปใช้เป็นส่วนหนึ่งใน Layout ได้เช่นเดียวกับ TextView, Button และวิวชนิดอื่นๆ

SurfaceView จะทำงานร่วมกับออบเจ็คอีก 2 ชนิดคือ SurfaceHolder และ Surface โดย Surface คือออบเจ็คที่เก็บข้อมูลพิกเซลของภาพที่ SurfaceView แสดงออกมา ส่วน SurfaceHolder ให้คิดง่ายๆว่าเป็นออบเจ็คที่เชื่อมโยงระหว่าง SurfaceView กับ Surface ดังรูป (ออบเจ็ค SurfaceHolder กับ Surface นี้เราไม่ต้องสร้างเอง แต่ SurfaceView จะจัดการให้อัตโนมัติ เนื่องจากมันเป็นกลไกการทำงานที่อยู่เบื้องหลัง SurfaceView)

Surface นั้นมี Lifecycle กล่าวคือ มันจะถูกสร้างขึ้นเมื่อ SurfaceView แสดงผลบนหน้าจอ และจะถูกทำลายเมื่อ SurfaceView ไม่ได้แสดงผลออกมา ดังนั้นเมื่อจะวาดภาพใน SurfaceView เราต้องแน่ใจว่าขณะนั้น Surface มีตัวตนอยู่ มิฉะนั้นจะเกิดข้อผิดพลาด

SurfaceView แตกต่างจากวิวชนิดอื่นตรงที่มันไม่ได้วาด UI ของตัวเอง แต่จะเปิดโอกาสให้ออบเจ็คอื่นมาวาด ซึ่งออบเจ็คที่จะวาด UI ของ SurfaceView ในตัวอย่างนี้ก็คือออบเจ็ค Camera นั่นเอง (Camera นำภาพพรีวิวที่ได้จากเลนส์กล้องมาวาดลงบน SurfaceView ทำให้ปรากฏภาพพรีวิวออกมาที่ SurfaceView)

ตัวอย่างนี้จะสร้างซับคลาสของ SurfaceView (อาจเรียกว่า Custom SurfaceView) ในลักษณะเดียวกับคลาส WebImageView ที่ใช้ดาวน์โหลดรูปภาพจากเซิร์ฟเวอร์ในบทที่แล้ว โดยเราจะรวบรวมโค้ดที่ใช้แสดงภาพพรีวิวจากกล้องไว้ภายในคลาสนี้ทั้งหมด หลังจากนั้นเพียงแค่นำ Custom SurfaceView นี้ไประบุใน Layout File แอคทิวิตีของเราก็จะแสดงภาพพรีวิวจากกล้องได้ทันที โดยเราไม่ต้องเขียนโค้ดในแอคทิวิตีเลย

1 สร้างคลาสใหม่ชื่อ MySurfaceView แล้วพิมพ์โค้ดดังนี้

โปรเจ็ค CameraOverlay, ไฟล์ MySurfaceView.java

```
package com.example.cameraoverlay;

import java.util.List;

import android.content.Context;
import android.hardware.Camera;
import android.util.AttributeSet;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class MySurfaceView extends SurfaceView
                           implements SurfaceHolder.Callback { ❶

    private static Context mContext;
    private static Camera mCamera;

    // คอนสตรัคเตอร์
    public MySurfaceView(Context context) {
        this(context, null);
    }

    // คอนสตรัคเตอร์
    public MySurfaceView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
```

```
}

@SuppressWarnings("deprecation") // ไม่ให้แสดงคำเตือน (warning) ที่บรรทัด setType
// คอนสตรัคเตอร์
public MySurfaceView(Context context, AttributeSet attrs,
                     int defaultStyle) {
    super(context, attrs, defaultStyle);
    mContext = context;

    SurfaceHolder holder = getHolder();  // เข้าถึง SurfaceHolder
    holder.addCallback(this); ❷ // กำหนด Callback เมื่อเกิดอีเวนต์ของ Surface

    /* เมธอด setType และค่าคงที่ SURFACE_TYPE_PUSH_BUFFERS ถูกยกเลิก
       (deprecated) ตั้งแต่แอนดรอยด์ 3.0 เป็นต้นไป เนื่องจากแอนดรอยด์จะกำหนดให้
       อัตโนมัติอยู่แล้ว (ถ้าจำเป็น) อย่างไรก็ตาม การเรียกเมธอดนี้ยังจำเป็นสำหรับแอนดรอยด์
       ก่อน 3.0 */
    holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
}

@Override
public void surfaceCreated(SurfaceHolder holder) {
    try {
        mCamera = Camera.open(); ❸
        mCamera.setPreviewDisplay(holder); ❹
    } catch (Exception e) {
        e.printStackTrace();
    }
}

@Override
public void surfaceChanged(SurfaceHolder holder, int format,
                           int width, int height) {
    // เข้าถึงพารามิเตอร์ของกล้อง
    Camera.Parameters params = mCamera.getParameters();
    // หาขนาดภาพพรีวิวทั้งหมดที่มีให้เลือก
    List<Camera.Size> sizes = params.getSupportedPreviewSizes();
    // เลือกขนาดภาพพรีวิวขนาดแรก (ปกติจะเป็นขนาดใหญ่ที่สุด เพราะเรียงจากใหญ่ไปเล็ก)
    Camera.Size selected = sizes.get(0);
    // กำหนดขนาดภาพพรีวิวที่เลือกลงในพารามิเตอร์
    params.setPreviewSize(selected.width, selected.height);
    // กำหนดพารามิเตอร์ให้กับออบเจ็ค Camera
    mCamera.setParameters(params);

    mCamera.startPreview(); ❻ // เริ่มแสดงภาพพรีวิวจากกล้อง
```

❺

```
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        mCamera.stopPreview(); ❼  // หยุดแสดงภาพพรีวิว
        mCamera.release(); ❽      // ยกเลิกการใช้งานกล้อง
    }
}
```

NOTE

คลาส Camera ที่ใช้ในตัวอย่างนี้คือ android.hardware.Camera ไม่ใช่ android.graphics.Camera ดังนั้นขอให้แน่ใจว่าคุณอิมพอร์ตคลาสที่ถูกต้อง

เราสร้างคอนสตรัคเตอร์ของ MySurfaceView ขึ้น 3 ชุด (Overloaded Constructors) เพื่อให้เป็นมาตรฐานเดียวกับวิวชนิดอื่นๆ รายละเอียดการทำงานจะอยู่ในคอนสตรัคเตอร์ชุดที่ 3 โดยคอนสตรัคเตอร์อีก 2 ชุดจะเรียกมายังคอนสตรัคเตอร์นี้

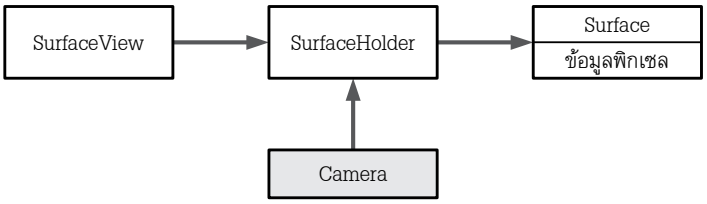
อย่างที่บอกก่อนหน้านี้ว่า ออบเจ็ค Surface ที่เป็นตัวเก็บข้อมูลพิกเซลของ SurfaceView นั้นมี Lifecycle เราสามารถระบุการทำงานในแต่ละอีเวนต์ของ Surface ได้โดยการ Implement อินเทอร์เฟส SurfaceHolder.Callback แล้ว Override เมธอดต่างๆ ได้แก่ surfaceCreated, surfaceChanged และ surfaceDestroyed

ในที่นี้กำหนดให้คลาส MySurfaceView ทำการ Implement อินเทอร์เฟสดังกล่าว ❶ เพื่อจะได้สร้างเมธอดทั้งสามไว้ภายในคลาสนี้เลย จากนั้นในคอนสตรัคเตอร์ เราเรียกเมธอด addCallback ของ SurfaceHolder ❷ เพื่อบอกให้ SurfaceHolder เรียกมายังเมธอดทั้งสามในคลาสนี้เมื่อเกิดอีเวนต์ของ Surface (Surface ถูกสร้าง, มีการเปลี่ยนแปลง หรือถูกทำลาย)

รายละเอียดการทำงานของเมธอดทั้งสาม

- **เมธอด surfaceCreated** เมื่อ Surface ถูกสร้างขึ้นมา เราจะเปิดการทำงานของกล้องโดยเรียกเมธอด open ของคลาส Camera ❸ เมธอดนี้จะส่งคืนออบเจ็ค Camera กลับมาให้ซึ่งเราจะควบคุมและตั้งค่าการถ่ายภาพโดยใช้ออบเจ็คนี้

  ถัดไปเป็นการบอกออบเจ็ค Camera ว่าให้ใช้ Surface นี้ในการแสดงภาพพรีวิวจากกล้อง ❹ (แต่ภาพพรีวิวยังไม่ถูกแสดงออกมาจนกว่าจะเรียกเมธอด startPreview ของ Camera) โดย Camera จะเข้าถึง Surface ผ่านทาง SurfaceHolder ดังรูปหน้าถัดไป



- **เมธอด surfaceChanged** เมธอดนี้จะถูกเรียกเมื่อรูปแบบหรือขนาดของ Surface มีการเปลี่ยนแปลง รวมถึงตอนที่เพิ่งสร้าง Surface ขึ้นมาด้วย ซึ่งเราจะเลือกขนาดภาพพรีวิวที่จะแสดงออกมา (โดยทั่วไปฮาร์ดแวร์ของกล้องจะมีภาพพรีวิวหลายขนาดให้เราเลือกใช้ได้ตามความเหมาะสม) แต่เพื่อความง่าย ในที่นี้จะเลือกภาพพรีวิวขนาดใหญ่ที่สุด ❺ จากนั้นจึงเริ่มการแสดงภาพพรีวิว (Camera เริ่มวาดภาพพรีวิวลงบน Surface) ❻
- **เมธอด surfaceDestroyed** เมื่อ Surface ถูกทำลาย เราจะหยุดการแสดงภาพพรีวิว ❼ และยกเลิกการใช้กล้อง ❽

NOTE

หลังจากใช้งานกล้องเสร็จแล้วต้องยกเลิกการใช้งานทุกครั้ง โดยเรียกเมธอด release ดังเช่นในตัวอย่างนี้ มิฉะนั้นกล้องจะถูกล็อคไว้ ทำให้แอพอื่นไม่สามารถใช้งานได้

2 กำหนด Layout ดังนี้

โปรเจ็ค CameraOverlay, ไฟล์ activity__main.xml

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

    <com.example.cameraoverlay.MySurfaceView
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

จะเห็นว่าเรานำคลาส MySurfaceView ที่สร้างเมื่อครู่นี้ มากำหนดเป็นอิลิเมนต์ใน Layout (ซึ่งทำได้เพราะ MySurfaceView เป็นซับคลาสของ SurfaceView ดังนั้นจึงเป็นวิวชนิดหนึ่ง) โดยให้แสดงผลเต็มหน้าจอ

3 ที่ไฟล์ AndroidManifest.xml ให้ขอสิทธิ์การใช้งานกล้อง และกำหนดให้แอคทิวิตีแสดงผลแนวนอน (landscape) เสมอ

โปรเจ็ค CameraOverlay, ไฟล์ AndroidManifest.xml

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

<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
        android:name="com.example.cameraoverlay.MainActivity"
        android:label="@string/app_name"
        android:screenOrientation="landscape" >
        ...
    </activity>
</application>
...
```

ใน 2 ตัวอย่างที่ผ่านมาเราไม่ต้องขอสิทธิ์เข้าถึงกล้อง เพราะเป็นการใช้อินเทนต์เรียกแอพกล้องขึ้นมา (แอพกล้องต้องเป็นผู้ขอสิทธิ์นี้) แต่ในตัวอย่างนี้เราเข้าถึงกล้องเองโดยตรงจึงต้องขอสิทธิ์เอง

การสร้างแอพถ่ายภาพนั้นปกติจะกำหนดทิศทางของแอคทิวิตีเป็นแนวนอนหรือแนวตั้งอย่างใดอย่างหนึ่งตายตัว เพื่อวางปุ่มชัตเตอร์และตัวเลือกการตั้งค่ากล้องไว้ที่ตำแหน่งเดิมเสมอไม่ว่าผู้ใช้จะถ่ายภาพในแนวตั้งหรือแนวนอนก็ตาม (ถ้าไม่กำหนดตายตัว พอผู้ใช้หมุนจอ แอคทิวิตีและวิวต่างๆใน Layout ก็จะหมุนตาม นอกจากนี้ยังทำให้การแสดงภาพพรีวิวสะดุด เพราะแอคทิวิตีจะถูกทำลายและสร้างขึ้นใหม่ ส่งผลให้ SurfaceView และออบเจ็คอื่นๆที่เกี่ยวข้องถูกทำลายและสร้างขึ้นใหม่เช่นกัน)

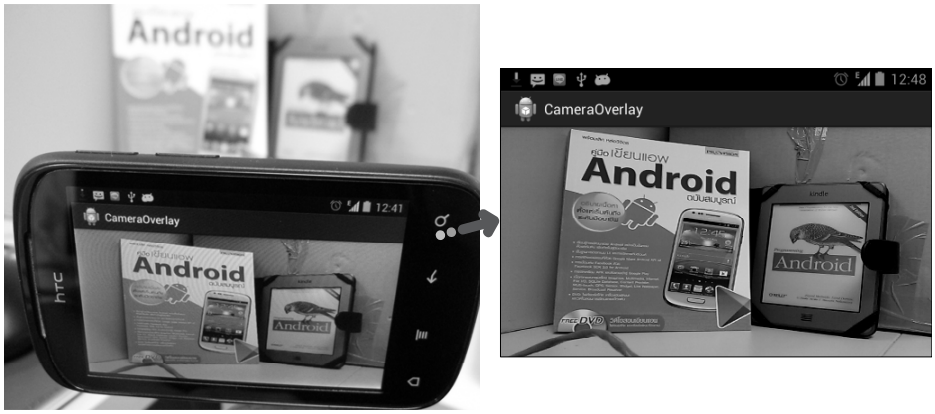
TIP

ถ้าหากคุณจะส่งแอพขึ้น Google Play และต้องการให้เฉพาะเครื่องที่มีกล้องเท่านั้นที่สามารถติดตั้งแอพของคุณได้ ให้เพิ่มแท็กในไฟล์ AndroidManifest.xml ดังนี้ โดยใส่ไว้ก่อนแท็ก <application>

```
<uses-feature android:name="android.hardware.camera" />
```

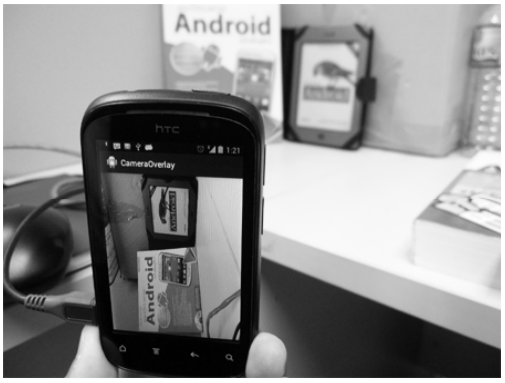
แท็กข้างต้นทำหน้าที่เป็นตัวกรองหรือ Filter โดยอุปกรณ์แอนดรอยด์ที่ไม่มีกล้องเมื่อเข้าไปยัง Google Play จะไม่ปรากฏแอพของคุณออกมาให้เลือกติดตั้งได้

ผลการรัน



แก้ปัญหาภาพหมุน

เมื่อรันแอพนี้บนอุปกรณ์แอนดรอยด์บางรุ่น โดยเฉพาะเมื่อกำหนดแอคทิวิตีไว้เป็นแนวตั้ง (portrait) ตายตัว จะทำให้ภาพพรีวิวที่ปรากฏออกมามีทิศทางผิดเพี้ยนไป เนื่องจากทิศทางของกล้องไม่สอดคล้องกับทิศทางของแอคทิวิตี ดังรูป (นอกจากนี้การรันบนอีมูเลเตอร์ก็จะได้ภาพกลับหัว แต่ผู้เขียนไม่แน่ใจว่าเครื่องอื่นๆจะเป็นหรือไม่ เพราะปัญหาอาจเกิดจากเว็บแคมในเครื่องพีซีของผู้เขียนก็ได้)



เราสามารถแก้ปัญหานี้โดยกำหนดทิศทางของกล้องให้สอดคล้องกับทิศทางของแอคทิวิตีดังนี้

1 เพิ่มเมธอด `setCameraDisplayOrientation` ในคลาส `MySurfaceView`

โปรเจ็ค CameraOverlay, ไฟล์ MySerfaceView.java

```
public void setCameraDisplayOrientation() {
    // ขอข้อมูลเกี่ยวกับกล้อง
    android.hardware.Camera.CameraInfo info =
```

```
            new android.hardware.Camera.CameraInfo();
    android.hardware.Camera.getCameraInfo(0, info);

    // หาทิศทางของแอคทิวิตี
    int rotation = ((WindowManager) mContext
            .getSystemService(Context.WINDOW_SERVICE)).getDefaultDisplay()
            .getRotation();

    int degrees = 0;
    switch (rotation) {
    case Surface.ROTATION_0:
        degrees = 0;
        break;
    case Surface.ROTATION_90:
        degrees = 90;
        break;
    case Surface.ROTATION_180:
        degrees = 180;
        break;
    case Surface.ROTATION_270:
        degrees = 270;
        break;
    }

    // ปรับทิศทางของกล้องให้สอดคล้องกับแอคทิวิตี
    int result;
    if (info.facing == Camera.CameraInfo.CAMERA_FACING_FRONT) {
        result = (info.orientation + degrees) % 360;
        result = (360 - result) % 360;
    } else {
        result = (info.orientation - degrees + 360) % 360;
    }
    mCamera.setDisplayOrientation(result);
}
```

2 ที่เมธอด surfaceChanged ให้เรียกไปยังเมธอด setCameraDisplayOrientation ที่เพิ่งสร้างขึ้นก่อนจะเริ่มแสดงภาพพรีวิว

โปรเจ็ค CameraOverlay, ไฟล์ MySerfaceView.java

```
@Override
public void surfaceChanged(SurfaceHolder holder, int format,
                           int width, int height) {
    Camera.Parameters params = mCamera.getParameters();
    List<Camera.Size> sizes = params.getSupportedPreviewSizes();
```

```
    Camera.Size selected = sizes.get(0);
    params.setPreviewSize(selected.width, selected.height);
    mCamera.setParameters(params);

    setCameraDisplayOrientation();
    mCamera.startPreview();
}
```

3 ที่ไฟล์ AndroidManifest.xml ให้กำหนด `minSdkVersion` เป็น 9 หรือสูงกว่า เนื่องจากคลาส `CameraInfo` มีให้ใช้ในแอนดรอยด์ 2.3 (API Level 9) ขึ้นไป

โปรเจ็ค CameraOverlay, ไฟล์ AndroidManifest.xml

```
<uses-sdk
    android:minSdkVersion="9"
    android:targetSdkVersion="17" />
```

เมื่อรันแอพก็จะได้ภาพพรีวิวที่มีทิศทางถูกต้อง ไม่ว่าจะกำหนดแอคทิวิตีเป็นแนวตั้งตายตัว แนวนอนตายตัว หรือให้หมุนแอคทิวิตีไปตามทิศทางของตัวเครื่องก็ตาม

การวาดกราฟิกซ้อนบนภาพพรีวิวจากกล้อง

การวาดกราฟิกซ้อนลงบนภาพพรีวิวจะทำให้เราสร้างแอพที่น่าสนใจได้ ยกตัวอย่างเช่น เราอาจแสดงภาพหรือข้อความบนภาพพรีวิวโดยใช้ข้อมูลจาก GPS และเซ็นเซอร์ต่างๆของเครื่อง เพื่อสร้างสิ่งที่เรียกว่า Augmented Reality เป็นต้น

ตัวอย่างและคำอธิบาย

เมื่อแสดงภาพพรีวิวจากกล้องได้แล้ว การซ้อนภาพหรือข้อความลงบนภาพพรีวิวก็ไม่ใช่เรื่องยาก ซึ่งทำได้ 2 วิธีคือ

- สร้างวิวต่างๆขึ้นใน Layout และให้วางอยู่บน `SurfaceView` (ระบุแท็กของวิวเหล่านั้นหลังจากแท็กของ `SurfaceView`) โดยใช้ `RelativeLayout` หรือ `FrameLayout` เป็นตัวครอบ `SurfaceView` และวิวเหล่านั้น
- ถ้าหากต้องการวาดข้อความหรือกราฟิกซ้อนลงบนภาพพรีวิว ให้ Override เมธอด `onDraw` ของ `SurfaceView` แต่ทั้งนี้จะต้องเรียกเมธอด `setWillNotDraw` ของ `SurfaceView` โดยกำหนดเป็นค่า `false` ด้วย (ความหมายของเมธอดนี้จะอธิบายต่อไป)

ในที่นี้จะทำทั้งสองวิธีไปพร้อมกัน โดยทำต่อจากหัวข้อที่แล้ว

1 เพิ่มเติม Layout เป็นดังนี้

โปรเจ็ค CameraOverlay, ไฟล์ activity__main.xml

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

    <RelativeLayout ❶
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1" >

        <com.example.cameraoverlay.MySurfaceView ❷
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

        <ImageView ❸
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_alignParentTop="true"
            android:src="@drawable/android3d" />
    </RelativeLayout>

    <Button ❹
        android:id="@+id/shutter_button"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:text="Take!" />

</LinearLayout>
```

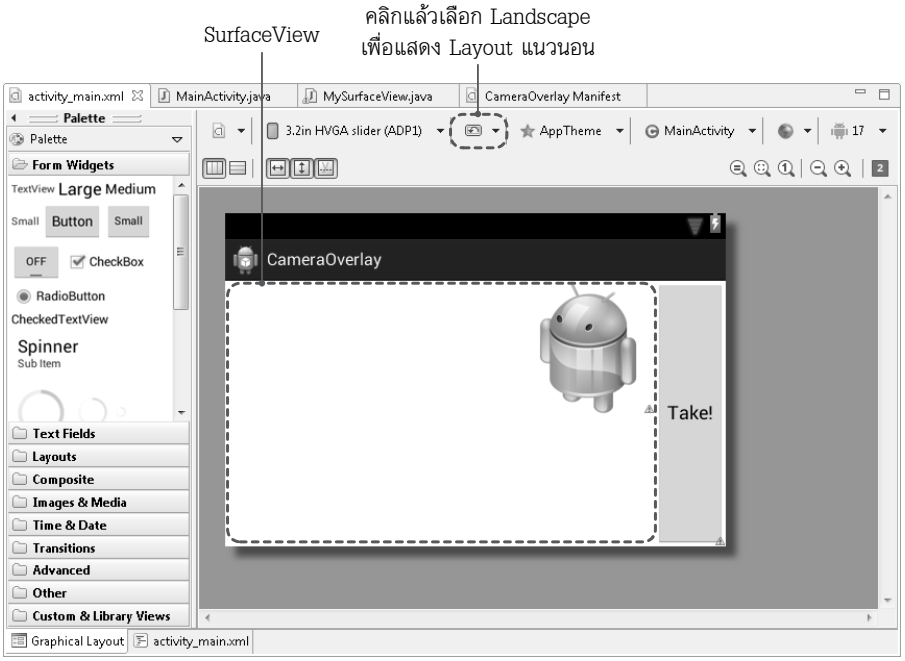
android3d.png
(พื้นหลังสีขาวคือส่วนที่โปร่งใส)

เราสร้าง RelativeLayout ❶ ขึ้นมาครอบ SurfaceView ❷ แล้วเพิ่ม ImageView ❸ ที่แสดงรูปหุ่นแอนดรอยด์ 3 มิติ (ไฟล์ android3d.png) เข้าไปใน RelativeLayout โดยวางชิดมุมขวาบนของ RelativeLayout

การที่เราใส่แท็กของ ImageView ไว้หลังแท็กของ SurfaceView จะทำให้รูปหุ่นแอนดรอยด์ถูกแสดงอยู่เหนือ SurfaceView

สำหรับปุ่มที่เพิ่มเข้ามาคือปุ่มชัตเตอร์ ❹ ปุ่มนี้ไม่ได้อยู่ใน RelativeLayout แต่อยู่ระดับเดียวกันภายใต้ LinearLayout ชั้นนอกสุด เราจะเขียนโค้ดให้กับปุ่มนี้ในหัวข้อถัดไป

2 สลับ Layout File ไปที่มุมมองกราฟิก (โดยคลิกแท็บ Graphical Layout ข้างล่าง) แล้วกำหนดให้แสดง Layout ในแนวนอน ตอนนี้ Layout ควรมีหน้าตาดังรูป



ถัดไปจะเพิ่มโค้ดที่วาดกราฟิกและข้อความลงบน `SurfaceView`

3 ที่คลาส `MySurfaceView` ให้เพิ่มตัวแปรระดับคลาส (ฟิลด์) อีก 5 ตัวแปร

โปรเจ็ค CameraOverlay, ไฟล์ MySerfaceView.java

```
private Paint mPaint;              // พู่กันวาดภาพ
private Bitmap mImage;             // รูปไอคอนแอนดรอยด์ (ic_launcher.png)
private int mCenterX, mCenterY;    // จุดกึ่งกลางของ SurfaceView
private RectF mOvalBound;          // สี่เหลี่ยมที่ใช้เป็นขอบเขตในการวาดวงรี (รูปปาก)
```

4 เพิ่มโค้ดในคอนสตรัคเตอร์ เพื่อสร้างออบเจ็ค `Paint` ซึ่งเป็นพู่กันวาดภาพ (อ้างอิงด้วยตัวแปร `mPaint`) และอ่านข้อมูลพิกเซลภาพจากไฟล์ ic_launcher.png ที่เป็นรีซอร์ส มาเก็บลงในออบเจ็ค `Bitmap` (อ้างอิงด้วยตัวแปร `mImage`)

โปรเจ็ค CameraOverlay, ไฟล์ MySerfaceView.java

```
@SuppressWarnings("deprecation")
public MySurfaceView(Context context, AttributeSet attrs,
                     int defaultStyle) {
    super(context, attrs, defaultStyle);
```

```
    mContext = context;

    SurfaceHolder holder = getHolder();
    holder.addCallback(this);
    holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

    mPaint = new Paint();
    mImage = BitmapFactory.decodeResource(getResources(),
                                          R.drawable.ic_launcher);
}
```

5 เพิ่มเมธอด onSizeChanged

โปรเจ็ค CameraOverlay, ไฟล์ MySerfaceView.java

```
@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
    super.onSizeChanged(w, h, oldw, oldh);

    mCenterX = getWidth() / 2;
    mCenterY = getHeight() / 2;
    mOvalBound = new RectF(mCenterX, mCenterY - 10, mCenterX + 45,
                           mCenterY + 50);
}
```

เมธอดนี้จะถูกเรียกเมื่อขนาดของ SurfaceView เปลี่ยนไป รวมถึงตอนที่ SurfaceView เพิ่งถูกสร้างขึ้นมาด้วย ซึ่งเราจะคำนวณหาจุดกึ่งกลางของ SurfaceView และกำหนดขอบเขตการวาดวงรีในเมธอดนี้ (เราไม่สามารถหาจุดกึ่งกลางของ SurfaceView ในคอนสตรัคเตอร์ได้ เพราะตอนที่คอนสตรัคเตอร์ทำงานจะยังไม่รู้ขนาดของ SurfaceView)

6 เพิ่มการเรียกเมธอด setWillNotDraw ในเมธอด surfaceCreated

โปรเจ็ค CameraOverlay, ไฟล์ MySerfaceView.java

```
@Override
public void surfaceCreated(SurfaceHolder holder) {
    try {
        mCamera = Camera.open();
        setWillNotDraw(false);
        mCamera.setPreviewDisplay(holder);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

เมธอด setWillNotDraw คือเมธอดที่ถูกกำหนดไว้ในคลาส View ซึ่งเป็นซูเปอร์คลาสของวิวทุกชนิด ถ้าหากเรียกเมธอดนี้โดยระบุค่า true จะเป็นการบอกแอนดรอยด์ว่าวิวนั้นไม่วาด UI ของตัวเอง แอนดรอยด์ก็จะ optimize การแสดงผลของวิวให้มีประสิทธิภาพยิ่งขึ้น ดังเช่นกรณีของ ViewGroup ต่างๆ เช่น LinearLayout, RelativeLayout ฯลฯ

กรณีของ SurfaceView ก็เช่นเดียวกัน อย่างที่บอกแล้วว่าปกติ SurfaceView ไม่ได้วาด UI ของตัวเอง มันจึงเรียกเมธอด setWillNotDraw โดยระบุค่า true เป็นดีฟอลต์มาให้ ผลก็คือถ้าเรา Override เมธอด onDraw เพื่อวาดกราฟิกลงบน SurfaceView (ซึ่งถือเป็นการวาดจากภายในตัว SurfaceView เอง) แอนดรอยด์ก็จะไม่สนใจ และกราฟิกที่เราวาดจะไม่ปรากฏออกมา แต่ในที่นี้ต้องการวาดกราฟิกลงบน SurfaceView เพื่อซ้อนภาพลงบนภาพพรีวิวที่มาจากกล้อง ดังนั้นจึงต้องเรียก setWillNotDraw โดยระบุค่า false (ความหมายคือ ต้องการวาด UI ของตัวเองด้วย)

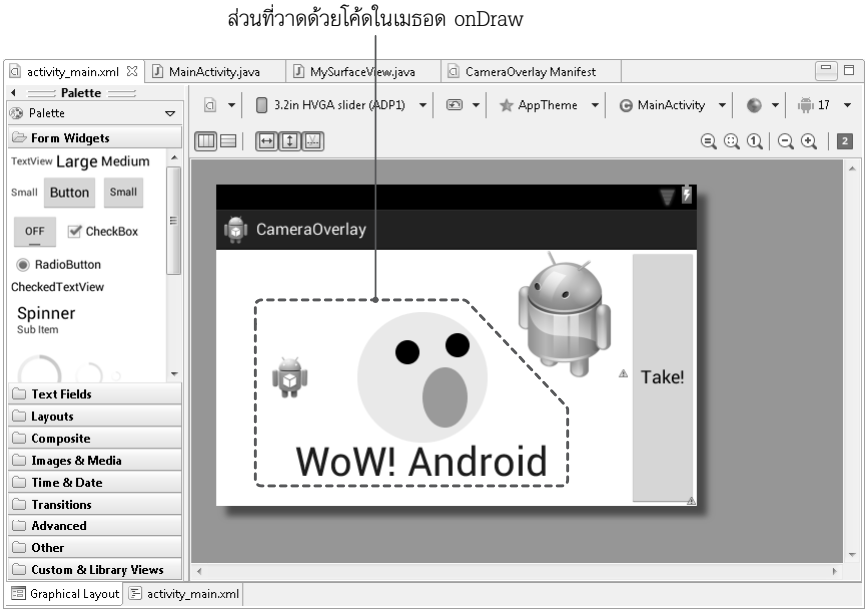
7 เพิ่มเมธอด onDraw

โปรเจ็ค CameraOverlay, ไฟล์ MySerfaceView.java

```
@Override
protected void onDraw(Canvas canvas) {
    mPaint.setFlags(Paint.ANTI_ALIAS_FLAG);

    // วาดหน้ากลมสีเหลือง ค่าโปร่งใส 200 (ค่ายิ่งมากยิ่งโปร่งใส ทำให้มองทะลุไปยังภาพข้างหลังมากขึ้น)
    mPaint.setStyle(Paint.Style.FILL);
    mPaint.setColor(Color.argb(200, 255, 255, 0)); // โปร่งใส, แดง, เขียว, น้ำเงิน
    // ศูนย์กลางอยู่ที่ (mCenterX, mCenterY) รัศมีขนาด 65
    canvas.drawCircle(mCenterX, mCenterY, 65, mPaint);

    // วาดตากลมสีดำซ้าย
    mPaint.setStyle(Paint.Style.FILL);
    mPaint.setColor(Color.BLACK);
    canvas.drawCircle(mCenterX - 15, mCenterY - 25, 12, mPaint);

    // วาดตากลมสีดำขวา
    mPaint.setStyle(Paint.Style.FILL);
    mPaint.setColor(Color.BLACK);
    canvas.drawCircle(mCenterX + 35, mCenterY - 32, 12, mPaint);

    // วาดปากรูปวงรีสีแดง ค่าโปร่งใส 128
    mPaint.setStyle(Paint.Style.FILL);
    mPaint.setColor(Color.argb(128, 255, 0, 0));
    canvas.drawOval(mOvalBound, mPaint);

    // วาดข้อความ "WoW! Android"
    mPaint.setStyle(Paint.Style.FILL);
```

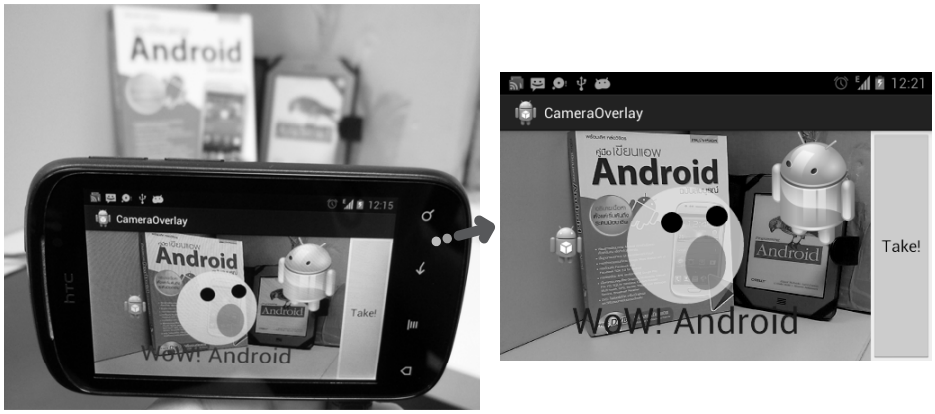
```
        mPaint.setColor(Color.BLUE);
        mPaint.setTextSize(40);
        mPaint.setTextAlign(Paint.Align.CENTER);
        canvas.drawText("WoW! Android", mCenterX, getHeight() - 30, mPaint);

        // วาดภาพจากไฟล์ ic_launcher.png
        canvas.drawBitmap(mImage, 50, (getHeight() - mImage.getHeight()) / 2,
                          null);
    }
```

8 สลับ Layout File ไปที่มุมมองกราฟิก จะได้ดังรูป



จะเห็นว่าเมื่อใส่โค้ดที่ใช้วาดภาพไว้ในเมธอด **onDraw** ของวิว แล้วนำวิวมาระบุใน Layout ก็ทำให้ ADT สามารถแสดงภาพออกมาให้เราเห็นได้ตั้งแต่ช่วงเขียนแอพเลย

ผลการรัน



ระบุการทำงานของปุ่มชัตเตอร์

การเข้าถึงกล้องโดยตรงนั้นเราจะต้องเขียนโค้ดเพื่อถ่ายภาพเอง ในตัวอย่างนี้จะกำหนดให้ปุ่ม Take! เป็นปุ่มชัตเตอร์สำหรับถ่ายภาพ ซึ่งเมื่อคลิกจะบันทึกภาพพรีวิวที่แสดงอยู่ตอนนั้นลงไฟล์

NOTE

ภาพถ่ายที่บันทึกลงไฟล์จะไม่รวมข้อความและกราฟิกต่างๆที่ซ้อนอยู่บนภาพพรีวิว ถ้าคุณต้องการให้ข้อความและกราฟิกเหล่านั้นปรากฏในภาพถ่ายด้วย จะต้องวาดลงบนภาพถ่ายเองดังที่จะอธิบายในตอนท้าย ซึ่งความยุ่งยากอยู่ที่ต้องคำนวณขนาดและตำแหน่งของกราฟิกให้สอดคล้องกับภาพบนจอ เนื่องจากตอนซ้อนภาพพรีวิวเราอ้างอิงจากขนาดของ SurfaceView แต่เมื่อวาดลงบนไฟล์ภาพจะต้องอ้างอิงจากขนาดของภาพถ่าย

การถ่ายภาพ ให้เรียกเมธอด `takePicture` บนออบเจ็ค `Camera` โดยระบุ callback เป็นพารามิเตอร์ ซึ่ง callback นี้คือออบเจ็คของคลาสที่ Implement อินเทอร์เฟส `Camera.PictureCallback` ซึ่งเราจะใส่โค้ดที่ใช้บันทึกข้อมูลภาพลงไฟล์ไว้ในเมธอด `onPictureTaken` ภายในคลาสนี้

1 เพิ่มคลาส `TakePictureCallback` ไว้ภายในคลาส `MySurfaceView` (`TakePictureCallback` จะเป็น Inner Class ของ `MySurfaceView`)

โปรเจ็ค CameraOverlay, ไฟล์ MySerfaceView.java

```
private static class TakePictureCallback
                     implements Camera.PictureCallback {

    // ข้อมูลพิกเซลของภาพถ่ายจะถูกส่งมาทางพารามิเตอร์ data
    @Override
    public void onPictureTaken(byte[] data, Camera camera) {
```

```
        try {
            // บันทึกลงไฟล์ picture.jpg ใน SD Card
            File imageFile = new File(
                Environment.getExternalStorageDirectory(), "picture.jpg");
            FileOutputStream fos = new FileOutputStream(imageFile);
            fos.write(data);
            fos.flush();
            fos.close();

            // แสดง Toast
            Toast.makeText(mContext, "Picture saved.",
                    Toast.LENGTH_SHORT).show();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        /* หลังจากถ่ายภาพแล้วต้องสั่งให้แสดงภาพพรีวิวใหม่ มิฉะนั้น SurfaceView จะแสดง
           ภาพถ่ายค้างไว้ */
        camera.startPreview();
    }
}
```

2 เพิ่มเมธอด takePicture

โปรเจ็ค CameraOverlay, ไฟล์ MySerfaceView.java

```
public static void takePicture() {
    // เลือกภาพถ่ายขนาดใหญ่ที่สุดที่มีให้เลือก
    Camera.Parameters params = mCamera.getParameters();
    List<Camera.Size> sizes = params.getSupportedPictureSizes();
    Camera.Size selected = sizes.get(0); // ขนาดแรกใหญ่สุด เพราะเรียงจากใหญ่ไปเล็ก
    params.setPictureSize(selected.width, selected.height);
    mCamera.setParameters(params);

    /* เรียกเมธอด takePicture พร้อมทั้งกำหนด callback (ออบเจ็คของคลาส
       TakePictureCallback) */
    TakePictureCallback callback = new TakePictureCallback();
    mCamera.takePicture(null, null, null, callback);
}
```

3 ที่แอคทิวิตี ให้เพิ่มโค้ดในเมธอด onCreate เพื่อระบุการทำงานของปุ่ม Take! ให้เรียกไปยังเมธอด takePicture ใน MySurfaceView

โปรเจ็ค CameraOverlay, ไฟล์ MainActivity.java

```
Button shutterButton = (Button) findViewById(R.id.shutter_button);
shutterButton.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        MySurfaceView.takePicture();
    }
});
```

4 เพิ่มบรรทัดต่อไปนี้ในไฟล์ AndroidManifest.xml โดยใส่ไว้ก่อนแท็ก <application> เพื่อขอสิทธิ์ในการเขียนข้อมูลลง SD Card

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

5 รันแอพ แล้วกดปุ่ม Take! เพื่อถ่ายภาพ จะได้ไฟล์ picture.jpg ที่รูทไดเร็คทอรีของ SD Card

แนวทางการวาดกราฟิกลงบนภาพถ่าย

ได้อธิบายแล้วว่าภาพกราฟิกที่เราวาดซ้อนบนภาพพรีวิวจะไม่ปรากฏในภาพถ่าย แต่เราต้องวาดลงบนภาพถ่ายเองอีกที ผู้เขียนจึงคิดว่าน่าจะแสดงโค้ดตัวอย่างสักเล็กน้อย เพื่อให้คุณผู้อ่านนำไปดัดแปลงต่อยอดได้โดยง่าย

ที่เมธอด onPictureTaken ในคลาส TakePictureCallback ให้แก้ไขโค้ดที่ใช้บันทึกไฟล์ picture.jpg (โค้ดในบล็อค try) เป็นดังนี้

โปรเจ็ค CameraOverlay, ไฟล์ MySerfaceView.java

```
// เปิดไฟล์ picture.jpg ที่ SD Card เตรียมไว้สำหรับเขียนข้อมูล
File imageFile = new File(Environment.getExternalStorageDirectory(),
                          "picture.jpg");
FileOutputStream fos = new FileOutputStream(imageFile);

/* แปลงข้อมูลภาพถ่ายที่ส่งมาทางพารามิเตอร์ data (อาร์เรย์ของ byte) ไปเป็นออบเจ็ค Bitmap
   แล้วสร้างสำเนาของออบเจ็ค Bitmap นั้นขึ้นมาเพื่อให้แก้ไขได้ (Bitmap เป็น Immutable Object
   ซึ่งปกติไม่อนุญาตให้แก้ไขข้อมูลได้) */
Bitmap bitmap = BitmapFactory.decodeByteArray(data, 0, data.length)
                             .copy(Bitmap.Config.ARGB_8888, true);
Canvas canvas = new Canvas(bitmap);  // สร้าง Canvas ครอบ Bitmap
```

```
/*************************************************************************
 *** เรียก drawing method เพื่อวาดกราฟิกต่างๆลงบน Bitmap โดยผ่านทาง Canvas *****
 *** ในตัวอย่างนี้วาดวงกลมรูปเดียวที่กึ่งกลางภาพ ********************************
 *************************************************************************/
Paint paint = new Paint();

paint.setFlags(Paint.ANTI_ALIAS_FLAG);
paint.setStyle(Paint.Style.FILL);
paint.setColor(Color.argb(200, 255, 255, 0));  // สีเหลือง ค่าโปร่งใส 200

int pictureWidth = bitmap.getWidth();     // หาความกว้างของภาพถ่าย
int pictureHeight = bitmap.getHeight();  // หาความสูงของภาพถ่าย
canvas.drawCircle(pictureWidth / 2, pictureHeight / 2, pictureHeight / 4,
                  paint);
/*************************************************************************/

// บันทึกข้อมูลจากออบเจ็ค Bitmap ลงไฟล์ ในที่นี้บันทึกเป็นฟอร์แมต JPEG คุณภาพ 80%
bitmap.compress(Bitmap.CompressFormat.JPEG, 80, fos);
```

ไฟล์ภาพ picture.jpg ที่ได้