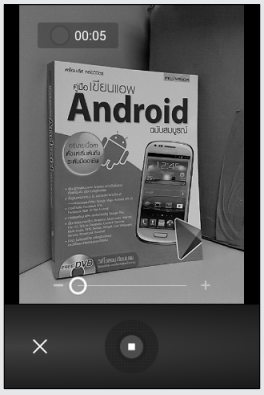
CHAPTER

09 เสียงและวิดีโอ

เนื้อหาในบทนี้

- การเล่นเสียง
- การใช้ MediaController ควบคุมการเล่นเสียง
- การเล่นวิดีโอ
- การบันทึกเสียง
- การถ่ายวิดีโอโดยใช้อินเทนต์



- การถ่ายวิดีโอโดยใช้ MediaRecorder

การเล่นเสียง

แอนดรอยด์มีคลาส MediaPlayer สำหรับเล่นเสียงและวิดีโอ โดยเสียงหรือวิดีโอนั้นอาจเป็นไฟล์รีซอร์สในแอพของเราเอง, เป็นไฟล์ในระบบไฟล์ (เช่น ใน SD card), เป็นข้อมูลจาก Content Provider หรือเป็นสตรีมข้อมูลจากเน็ตเวิร์กก็ได้

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะแสดงการเล่นเสียงจากไฟล์ MP3 ที่เป็นรีซอร์ส

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค SoundPlay, 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:padding="16dp" >

    <Button
        android:id="@+id/play_stop_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Play" />

</LinearLayout>
```

หน้าจอมีปุ่มเพียงปุ่มเดียว เริ่มต้นจะแสดงคำว่า Play ซึ่งเมื่อคลิกจะเริ่มเล่นเสียงและข้อความบนปุ่มจะเปลี่ยนเป็นคำว่า Stop และหากคลิกอีกทีจะหยุดเล่นเสียงและข้อความจะเปลี่ยนกลับเป็น Play

2 เพิ่มเติมโค้ดในแอคทิวิตีจนเป็นดังนี้

โปรเจ็ค SoundPlay, ไฟล์ MainActivity.java

```
package com.example.soundplay;

import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity
                        implements MediaPlayer.OnCompletionListener { ❸

    private MediaPlayer mPlayer;
    Button button;

    @Override
```

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

        button = (Button) findViewById(R.id.play_stop_button);
        button.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // ถ้ายังไม่ได้เล่นเสียงก็ให้เล่นเสียง แต่ถ้าเล่นเสียงอยู่ก็ให้หยุดเล่น
                if (mPlayer == null) {
                    mPlayer = MediaPlayer.create(MainActivity.this,
                                                    R.raw.sound); ❶
                    mPlayer.setOnCompletionListener(MainActivity.this); ❷
                    mPlayer.start(); ❻
                    button.setText("Stop");
                } else {
                    mPlayer.stop(); ❼
                    mPlayer.release(); ❽
                    mPlayer = null;
                    button.setText("Play");
                }
            }
        });
    }

    // ถูกเรียกเมื่อเล่นเสียงจบ
    @Override
    public void onCompletion(MediaPlayer mp) { ❹
        mPlayer.release(); ❺
        mPlayer = null;
        button.setText("Play");
    }

    // ถูกเรียกก่อนที่แอคทิวิตีจะถูกทำลาย
    @Override
    public void onDestroy() {
        super.onDestroy();
        if (mPlayer != null) {
            mPlayer.release(); ❾
            mPlayer = null;
        }
    }
}
```

ผู้เขียนเตรียมไฟล์เสียงชื่อ sound.mp3 ไว้ในโฟลเดอร์ res\raw ของโปรเจ็ค (ปกติโฟลเดอร์นี้จะไม่มี ให้สร้างขึ้นมาเอง)

การเล่นเสียงนั้น อันดับแรกต้องสร้างออบเจ็ค `MediaPlayer` และกำหนดแหล่งข้อมูลของเสียงที่จะเล่นให้เรียบร้อย ซึ่งทำได้โดยใช้เมธอด `create` (เป็น Static Method) ของคลาส `MediaPlayer` โดยระบุคอนเท็กซ์เป็นพารามิเตอร์ตัวแรก และระบุแหล่งข้อมูลเสียงเป็นพารามิเตอร์ตัวที่สอง ❶ ถัดไปกำหนด OnCompletion Listener ให้แก่ `MediaPlayer` เพื่อระบุการทำงานเมื่อเล่นเสียงจบ ในที่นี้ให้เรียกมายังแอคทิวิตีปัจจุบัน ❷ ดังนั้นแอคทิวิตีของเราจึงต้อง Implement อินเทอร์เฟส `OnCompletionListener` ❸ และ Override เมธอด `onCompletion` ❹ ซึ่งแอนดรอยด์จะเรียกมายังเมธอดนี้เมื่อเสียงถูกเล่นจบ เราจึงใส่โค้ดที่จะปลดปล่อยทรัพยากรต่างๆที่ `MediaPlayer` ใช้อยู่กลับคืนให้ระบบ ❺ จากนั้นเริ่มเล่นเสียงโดยเรียกเมธอด `start` ❻ ถ้าหากคลิกปุ่มขณะเล่นเสียงอยู่ เราจะหยุดเล่นเสียง ❼ และปลดปล่อยทรัพยากรเช่นเดียวกับตอนที่เล่นเสียงจบ ❽

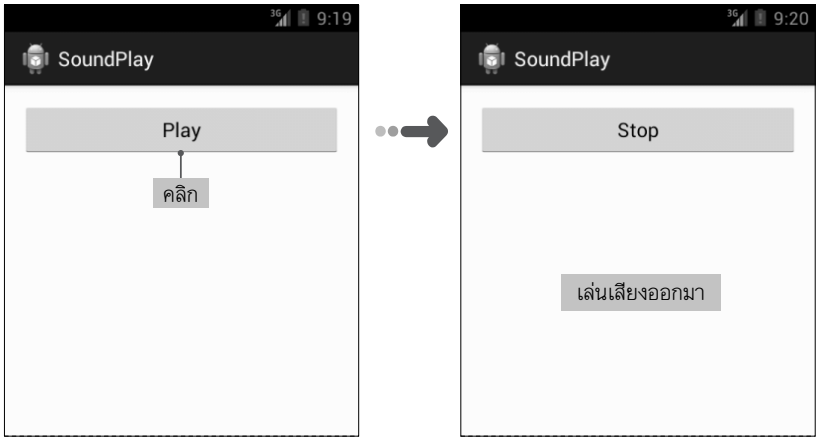
นอกจากนี้เมื่อแอคทิวิตีกำลังจะถูกทำลาย เช่นผู้ใช้กด Back หรือ Home เพื่อออกจากแอพ เราก็จะปลดปล่อยทรัพยากรที่ `MediaPlayer` ใช้งานอยู่เช่นเดียวกัน ❾

NOTE

แอนดรอยด์อนุญาตให้สร้างออบเจ็ค `MediaPlayer` ขึ้นพร้อมกันได้จำนวนหนึ่งเท่านั้น โดยจะเกิดข้อผิดพลาดหากสร้างขึ้นพร้อมกันมากเกินไป (ในเอกสารของแอนดรอยด์ไม่ได้บอกว่าให้สร้างได้สูงสุดเท่าใด แต่บอกแค่ว่า “too many MediaPlayer instances will result in an exception”)

ดังนั้นเมื่อไม่ต้องการเล่นเสียงแล้ว หรือเมื่อจะเรียกเมธอด `create` เพื่อสร้างออบเจ็ค `MediaPlayer` สำหรับเล่นเสียงใหม่ ก็ควรเรียกเมธอด `release` เพื่อปลดปล่อยทรัพยากรของ `MediaPlayer` เดิมก่อนทุกครั้ง

ผลการรัน



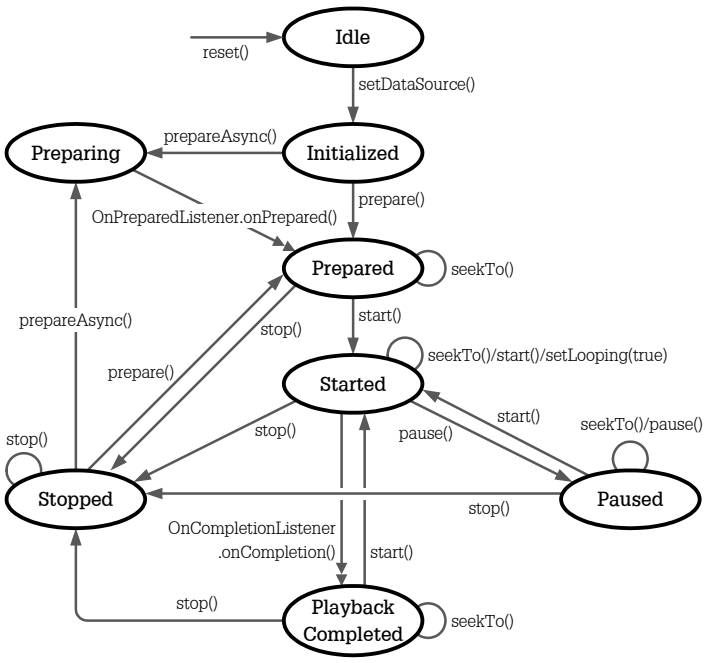
รู้จัก MediaPlayer ให้มากยิ่งขึ้น

`MediaPlayer` มีการทำงานแบบ State Machine กล่าวคือในขณะหนึ่งๆ `MediaPlayer` จะอยู่ในสถานะใดสถานะหนึ่ง และสถานะจะเปลี่ยนไปเมื่อเราเรียกเมธอดต่างๆของมัน ตัวอย่างเช่น ถ้าเรียกเมธอด `stop` ในขณะที่กำลังเล่นเสียงอยู่ จะทำให้ `MediaPlayer` เปลี่ยนสถานะจาก Started ไปเป็น Stopped เป็นต้น

การทำงานกับ `MediaPlayer` ประกอบด้วยขั้นตอนหลักๆ ดังนี้

1. ทำการเริ่มต้น (Initialize) Media Player โดยระบุเสียง/วิดีโอที่จะเล่น โดยใช้เมธอด `setDataSource`
2. เตรียม (Prepare) Media Player สำหรับการเล่นเสียง/วิดีโอ โดยใช้เมธอด `prepare` หรือ `prepareAsync`
3. เริ่ม (Start) เล่นเสียง/วิดีโอ โดยใช้เมธอด `start`
4. หยุดพัก (Pause) หรือหยุด (Stop) เล่นเสียง/วิดีโอก่อนเล่นจบ โดยใช้เมธอด `pause` หรือ `stop`
5. การเล่นเสียง/วิดีโอจบลงอย่างสมบูรณ์

ขั้นตอนต่างๆ ดังกล่าวจะทำให้เกิดการเปลี่ยนสถานะของ MediaPlayer ดังรูป



แผนภาพสถานะ (State Diagram) ของออบเจ็ค `MediaPlayer` และเมธอดต่าง ๆที่เกี่ยวข้อง

- เมื่อกำหนดแหล่งข้อมูลของเสียง/วิดีโอด้วยเมธอด setDataSource จะต้องเรียกเมธอด prepare (หรือ prepareAsync) ก่อนจึงจะสามารถเริ่มเล่นด้วยเมธอด start ได้ เช่น

```
MediaPlayer player = new MediaPlayer();
player.setDataSource("http://www.promlert.com/temp/sound.mp3");
player.prepare();
player.start();
```

อย่างไรก็ตาม ถ้าหากสร้างออบเจ็ค MediaPlayer ด้วยเมธอด create ดังตัวอย่างก่อนหน้านี้ เราไม่ต้องเรียกเมธอด prepare เองเพราะว่าเมธอด create เรียก prepare ให้แล้ว

- ให้ใช้เมธอด prepareAsync สำหรับข้อมูลเสียง/วิดีโอที่สตรีมมาจากเน็ตเวิร์ก เมธอดนี้จะ return ทันที ทำให้แอพของเราไม่ถูกบล็อคไว้ในขณะที่กำลังสตรีมข้อมูล จากนั้นเมื่อข้อมูลพร้อมสำหรับการเล่นแล้วแอนดรอยด์จะเรียกมายังเมธอด onPrepared ในอินเทอร์เฟส OnPreparedListener ที่เราเตรียมไว้ ดังตัวอย่าง

```
MediaPlayer player = new MediaPlayer();
player.setDataSource("http://www.promlert.com/temp/sound.mp3");

player.setOnPreparedListener(new MediaPlayer.OnPreparedListener() {

    @Override
    public void onPrepared(MediaPlayer mp) {
        /* แอนดรอยด์จะส่งออบเจ็ค MediaPlayer มาเป็นพารามิเตอร์ (ตัวแปร mp)
           ดังนั้นให้เรียกเมธอด start บนพารามิเตอร์นี้ (แต่กรณีนี้สามารถเรียกเมธอด
           start บนตัวแปร player ได้เช่นกัน) */
        mp.start();
    }
});

player.prepareAsync();
// อย่าเรียก start ตรงนี้ เพราะจะทำให้เกิดข้อผิดพลาด
```

- เมธอดอื่นๆของ MediaPlayer ที่น่าสนใจ เช่น เมธอด isPlaying บอกให้รู้ว่าขณะนั้นกำลังเล่นเสียง/วิดีโออยู่หรือไม่, เมธอด getDuration ใช้หาความยาวของเสียง/วิดีโอ (หน่วยเป็นมิลลิวินาที), เมธอด getCurrentPosition ใช้หาว่าตอนนั้นเล่นอยู่ที่ตำแหน่งเวลาเท่าใด (หน่วยเป็นมิลลิวินาที) และเมธอด seekTo สำหรับไปยังตำแหน่งเวลาที่ต้องการ (หน่วยเป็นมิลลิวินาทีเช่นกัน)

การใช้ MediaController ควบคุมการเล่นเสียง

เราสามารถสร้างปุ่ม Play, Pause, Stop เอาไว้ให้ผู้ใช้ควบคุมการเล่นเสียงได้ แต่วิธีที่สะดวกกว่าคือการใช้ `MediaController` ที่แอนดรอยด์เตรียมมาให้

ตัวอย่างและคำอธิบาย

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค SoundPlayWithController, ไฟล์ activity__main.xml

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

Layout ของแอพนี้มี `LinearLayout` ว่างๆอยู่อันเดียวซึ่งมี ID เป็น main_layout ส่วน `MediaController` นั้นเราจะสร้างขึ้นด้วยโค้ดจาวา

2 เพิ่มโค้ดในแอคทิวิตีจนเป็นดังนี้

โปรเจ็ค SoundPlayWithController, ไฟล์ MainActivity.java

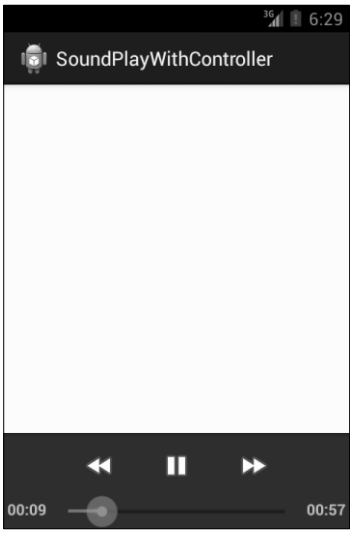
```
public class MainActivity extends Activity
                        implements MediaController.MediaPlayerControl {

    private MediaPlayer mPlayer;
    private MediaController mController;
    private Handler mHandler = new Handler();

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

        // สร้าง MediaController
        mController = new MediaController(this);
        /* ระบุการทำงานของ MediaController ให้เรียกมายังคลาสปัจจุบัน (เช่นเมื่อผู้ใช้คลิกปุ่ม
           Pause ก็จะเรียกมายังเมธอด pause ในแอคทิวิตีนี้) */
        mController.setMediaPlayer(this);
        /* ยึด MediaController ไว้กับ Root Layout ใน Layout ของเรา ซึ่งจะทำให้
           MediaController ปรากฏขึ้นมาด้านล่างจอ */
        mController.setAnchorView(findViewById(R.id.main_layout));
        mController.setEnabled(true);
```

```
        // สร้าง MediaPlayer และระบุแหล่งข้อมูลเสียงที่จะเล่น
        mPlayer = MediaPlayer.create(MainActivity.this, R.raw.sound);

        // ให้แสดง MediaController และเริ่มเล่นเสียงตอนที่เตรียมข้อมูลพร้อมสำหรับการเล่นแล้ว
        mPlayer.setOnPreparedListener(new OnPreparedListener() {
            @Override
            public void onPrepared(MediaPlayer mp) {
                /* ไม่รันโค้ดส่วนนี้ทันที แต่จะสร้างเป็น Runnable แล้วใช้เมธอด Handler.post
                   ใส่ลงใน Message Queue เพื่อรันตอนที่แอคทิวิตีแสดงผลออกมาอย่างสมบูรณ์
                   (มิฉะนั้นการเรียก mController.show จะเกิดข้อผิดพลาด) */
                mHandler.post(new Runnable() {
                    public void run() {
                        // แสดง 10 วินาที แล้วซ่อนอัตโนมัติ (ดีฟอลต์จะแสดงแค่ 3 วินาที)
                        mController.show(10000);
                        mPlayer.start();
                    }
                });
            }
        });
    }

    // MediaController จะซ่อนตัวเองอัตโนมัติ ดังนั้นเมื่อแตะหน้าจอให้แสดงออกมาใหม่
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        mController.show(10000);
        return false;
    }

    /* ซ่อน MediaController, หยุดการเล่นเสียง และปลดปล่อย MediaPlayer ช่วงที่แอคทิวิตี
       กำลังจะถูกทำลาย */
    @Override
    protected void onDestroy() {
        super.onDestroy();
        mController.hide();
        mPlayer.stop();
        mPlayer.release();
        mPlayer = null;
    }

    /**********************************************************************
     *** เมธอดต่างๆที่อินเทอร์เฟส MediaController.MediaPlayerControl กำหนดไว้ ***
     **********************************************************************/
    public void start() {
        mPlayer.start();
```

```
    }

    public void pause() {
        mPlayer.pause();
    }

    public int getDuration() {
        return mPlayer.getDuration();
    }

    public int getCurrentPosition() {
        return mPlayer.getCurrentPosition();
    }

    public void seekTo(int i) {
        mPlayer.seekTo(i);
    }

    public boolean isPlaying() {
        return mPlayer.isPlaying();
    }

    public int getBufferPercentage() {
        return 0;
    }

    public boolean canPause() {
        return true;
    }

    public boolean canSeekBackward() {
        return true;
    }

    public boolean canSeekForward() {
        return true;
    }
}
```

การเพิ่ม `MediaController` เข้ามานั้นจะต้องเตรียมคลาสที่ Implement อินเทอร์เฟส `MediaController.MediaPlayerControl` และทำการ Override เมธอดต่างๆตามที่อินเทอร์เฟสกำหนดไว้ เช่น เมธอด `start`, `pause`, `seekTo`, `getDuration`, `isPlaying` เป็นต้น ในที่นี้กำหนดให้แอคทิวิตี (คลาส `MainActivity`) Implement อินเทอร์เฟสดังกล่าว เพื่อที่เราจะ Override เมธอดเหล่านั้นในแอคทิวิตีเลย ซึ่งการทำงานของเมธอดส่วนใหญ่จะเรียกไปยังเมธอดของออบเจ็ค `MediaPlayer` อีกทีหนึ่ง

ผลการรัน



การเล่นวิดีโอ

การเล่นวิดีโอจะใช้ MediaPlayer เช่นเดียวกับการเล่นเสียง เพียงแต่ต้องเตรียม Surface สำหรับแสดงภาพวิดีโอด้วย วิธีที่ง่ายที่สุดในการเล่นวิดีโอก็คือการใช้ VideoView ซึ่งเป็นวิวที่รวมเอา Surface และ MediaPlayer ที่ใช้เล่นวิดีโอมาให้ในตัวเลย

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะแสดงการเล่นวิดีโอด้วย VideoView โดยมี MediaController ไว้ให้ผู้ใช้ควบคุมการเล่นวิดีโอได้

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค VideoPlay, 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" >

    <VideoView
        android:id="@+id/video"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

2 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี

โปรเจ็ค VideoPlay, ไฟล์ MainActivity.java

```
VideoView video = (VideoView) findViewById(R.id.video);
video.setKeepScreenOn(true); // ไม่ให้หรี่แสงหน้าจอขณะเล่นวิดีโอ
video.setVideoURI(Uri.parse("http://www.promlert.com/temp/test.3gp"));

// สร้าง MediaController แล้วผูกเข้ากับ VideoView
MediaController controller = new MediaController(this);
video.setMediaController(controller);

video.start();
```

ตัวอย่างนี้กำหนดให้เล่นไฟล์วิดีโอจากอินเทอร์เน็ตโดยใช้เมธอด setVideoURI แต่นอกจากนี้ยังสามารถเล่นไฟล์วิดีโอที่เก็บอยู่บนระบบไฟล์ในเครื่องได้โดยใช้เมธอด setVideoPath เช่น ถ้าต้องการเล่นไฟล์ test.3gp ที่อยู่ในรูทไดเร็คทอรีของ SD card จะเขียนโค้ดว่า

```
video.setVideoPath(Environment.getExternalStorageDirectory() + "/test.3gp");
```

การเพิ่ม MediaController เข้ามาในตัวอย่างนี้ไม่ยุ่งยากเหมือนตัวอย่างก่อน ทั้งนี้เพราะ VideoView เตรียมรายละเอียดการทำงานทั้งหมดไว้ให้แล้ว เราเพียงแต่สร้าง MediaController แล้วผูกเข้ากับ VideoView ก็ใช้ได้เลย

3 เพิ่มบรรทัดต่อไปนี้ในไฟล์ AndroidManifest.xml โดยใส่ไว้ก่อนแท็ก <application> เพื่อขอสิทธิ์ในการเข้าถึงอินเทอร์เน็ต (เพราะตัวอย่างนี้เล่นไฟล์วิดีโอจากอินเทอร์เน็ต)

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

ผลการรัน

การบันทึกเสียง

อุปกรณ์แอนดรอยด์ส่วนใหญ่มีไมโครโฟนในตัว จึงสามารถบันทึกเสียงจากภายนอกเข้าสู่เครื่องได้ การบันทึกเสียงจะใช้คลาส `MediaRecorder` ซึ่งง่ายพอๆกับการเล่นเสียงด้วย `MediaPlayer` โดยมีขั้นตอนคร่าวๆคือ

1. สร้างออบเจ็คของคลาส `MediaRecorder`
2. กำหนดแหล่งข้อมูลเสียงด้วยเมธอด `setAudioSource` ซึ่งโดยทั่วไปจะกำหนดเป็น `MediaRecorder.AudioSource.MIC` คือให้รับเสียงจากไมโครโฟนของเครื่อง
3. กำหนดฟอร์แมตของไฟล์เสียง (ไฟล์ผลลัพธ์) ด้วยเมธอด `setOutputFormat`
4. กำหนด Encoder ที่ใช้เข้ารหัสเสียงด้วยเมธอด `setAudioEncoder`
5. กำหนดชื่อพาธของไฟล์เสียงด้วยเมธอด `setOutputFile`
6. เรียกเมธอด `prepare` เพื่อเตรียมการสำหรับการบันทึกเสียง
7. เรียกเมธอด `start` เพื่อเริ่มบันทึกเสียง
8. เรียกเมธอด `stop` เพื่อหยุดบันทึกเสียง
9. หลังจากใช้งาน `MediaRecoder` เสร็จแล้ว ให้เรียกเมธอด `release` เพื่อปลดปล่อยทรัพยากรที่ `MediaRecorder` ใช้งานอยู่คืนให้ระบบ

ตัวอย่างและคำอธิบาย

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค SoundRecord, 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/start_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Start Recording" />

    <Button
        android:id="@+id/stop_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:enabled="false"
        android:text="Stop Recording" />

</LinearLayout>
```

หน้าจอของแอพนี้มีปุ่ม 2 ปุ่มคือ Start Recording กับ Stop Recording โดยเริ่มต้นจะกำหนดให้ปุ่ม Stop Recording ใช้งานไม่ได้

2 เพิ่มเติมโค้ดในแอคทิวิตีจนเป็นดังนี้

โปรเจ็ค SoundRecord, ไฟล์ MainActivity.java

```
package com.example.soundrecord;

import java.io.File;

import android.app.Activity;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {

    private Button mStartButton, mStopButton;
    private MediaRecorder mRecorder;
```

```
private File mOutputFile;

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

    // กำหนด Listener ให้กับปุ่มทั้งสอง
    mStartButton = (Button) findViewById(R.id.start_button);
    mStartButton.setOnClickListener(startListener);
    mStopButton = (Button) findViewById(R.id.stop_button);
    mStopButton.setOnClickListener(stopListener);

    /* สร้างออบเจ็ค MediaRecorder และสร้างไฟล์ผลลัพธ์เตรียมไว้ (เราจะกำหนดให้
       MediaRecorder บันทึกข้อมูลเสียงลงในไฟล์นี้) */
    mRecorder = new MediaRecorder();
    mOutputFile = new File(Environment.getExternalStorageDirectory(),
                           "sound.3gp");
}

// ปลดปล่อย MediaRecorder ในช่วงที่แอคทิวิตีกำลังจะถูกทำลาย
@Override
public void onDestroy() {
    super.onDestroy();
    if (mRecorder != null) {
        mRecorder.release();
        mRecorder = null;
    }
}

// Listener สำหรับปุ่ม Start Recording
private View.OnClickListener startListener =
                              new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        /* กำหนดแหล่งข้อมูลเสียงที่จะบันทึก, ฟอร์แมตของไฟล์ผลลัพธ์, ตัวเข้ารหัสเสียง
           (Encoder) และพาธของไฟล์ผลลัพธ์ */
        mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        mRecorder.setOutputFormat(MediaRecorder.OutputFormat
                                  .THREE_GPP);
        mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
        mRecorder.setOutputFile(mOutputFile.getAbsolutePath());
```

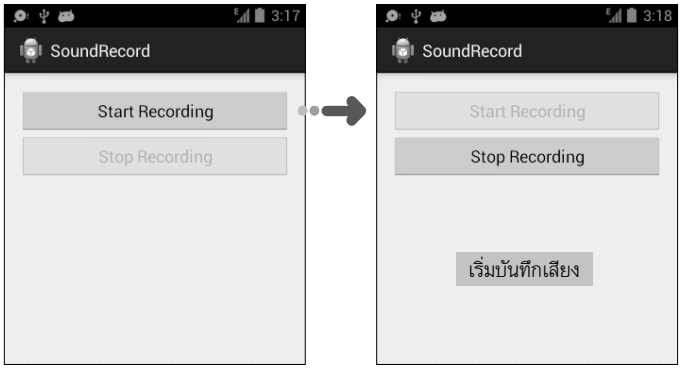
```
            try {
                mRecorder.prepare();          // เตรียมบันทึกเสียง
            } catch (Exception e) {
                e.printStackTrace();
            }

            mRecorder.start();                // เริ่มบันทึกเสียง
            mStartButton.setEnabled(false); // ปิดการใช้งานปุ่ม Start Recording
            mStopButton.setEnabled(true);   // เปิดการใช้งานปุ่ม Stop Recording
        }
    };

    // Listener สำหรับปุ่ม Stop Recording
    private View.OnClickListener stopListener =
                                new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            mRecorder.stop();                 // หยุดบันทึกเสียง
            mStartButton.setEnabled(true);  // เปิดการใช้งานปุ่ม Start Recording
            mStopButton.setEnabled(false);  // ปิดการใช้งานปุ่ม Stop Recording
        }
    };
}
```

3 เพิ่มบรรทัดต่อไปนี้ในไฟล์ AndroidManifest.xml โดยใส่ไว้ก่อนแท็ก `<application>` เพื่อขอสิทธิ์ในการบันทึกเสียงและเขียนไฟล์ลง SD card

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

ผลการรัน

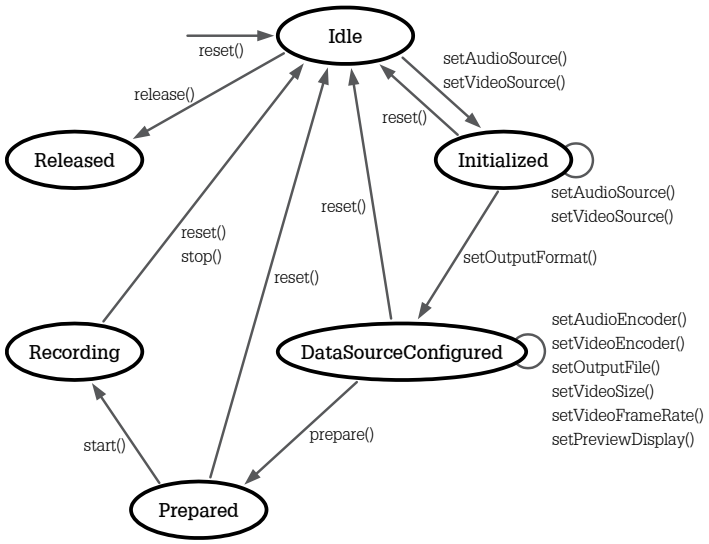
ตัวอย่างนี้ต้องรันบนเครื่องจริง เนื่องจากอีมูเลเตอร์ไม่สามารถบันทึกเสียงได้



หลังจากบันทึกเสียงเสร็จให้คลิก Stop Recording จะมีไฟล์ sound.3gp เพิ่มขึ้นมาที่ SD card ของเครื่อง ให้เปิดไฟล์นี้ด้วยแอพประเภท File Manager หรือ Media Player ในเครื่องแอนดรอยด์นั้น หรืออาจเพิ่มโค้ดจากตัวอย่างแรกในบทนี้เข้ามา เพื่อเล่นไฟล์เสียงดังกล่าวหลังคลิกปุ่ม Stop Recording

รู้จัก MediaRecorder ให้มากยิ่งขึ้น

`MediaRecorder` มีการทำงานแบบ State Machine เช่นเดียวกับ `MediaPlayer` รูปต่อไปนี้แสดง State Diagram และเมธอดต่างๆที่ทำให้สถานะของ `MediaRecorder` เปลี่ยนไป



การถ่ายวิดีโอโดยใช้อินเทนต์

การบันทึกวิดีโอในแอนดรอยด์ทำได้ 2 วิธี วิธีแรกคือใช้อินเทนต์เรียกแอพกล้องวิดีโอขึ้นมาทำหน้าที่บันทึกวิดีโอ และอีกวิธีหนึ่งคือใช้ `MediaRecorder` ซึ่งยุ่งยากกว่า แต่ทำให้แอพของเรามีความสามารถในการบันทึกวิดีโอได้ด้วยตัวเอง ในหัวข้อนี้จะอธิบายวิธีแรกก่อน

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะมีปุ่มซึ่งเมื่อคลิกจะเรียกแอพกล้องวิดีโอขึ้นมา และหลังจากถ่ายวิดีโอเรียบร้อยแล้วจะเล่นวิดีโอนั้นใน `VideoView` ทันที

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค VideoRecordIntent, 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/record_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Record Video" />

    <VideoView
        android:id="@+id/video"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="8dp" />

</LinearLayout>
```

2 ที่แอคทิวิตี ให้เพิ่มการประกาศค่าคงที่ RECORD_VIDEO ที่ส่วนประกาศของคลาส, เพิ่มโค้ดในเมธอด onCreate และเพิ่มเมธอด onActivityResult

โปรเจ็ค VideoRecordIntent, ไฟล์ MainActivity.java

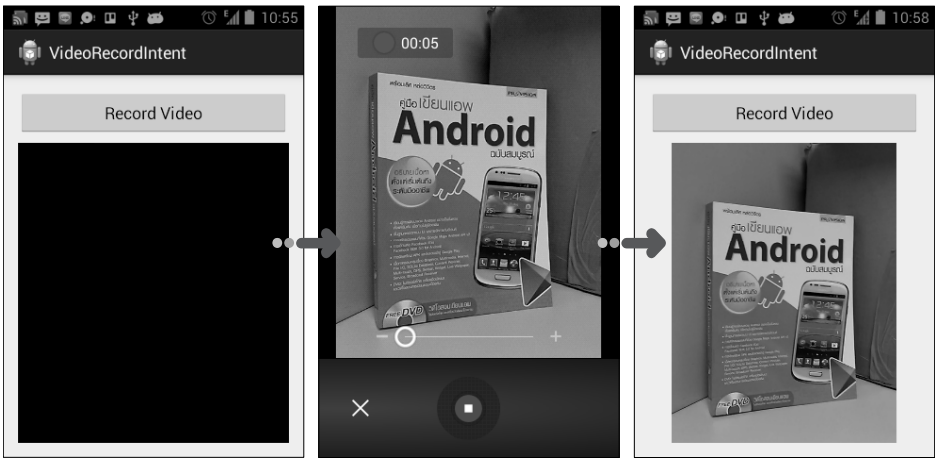
```
package com.example.videorecordintent;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import android.widget.Button;
import android.widget.VideoView;

public class MainActivity extends Activity {

    private static final int RECORD_VIDEO = 200;

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

```
        Button button = (Button) findViewById(R.id.record_button);
        button.setOnClickListener(new View.OnClickListener() {

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

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
                                    Intent data) {
        if (requestCode == RECORD_VIDEO && resultCode == RESULT_OK) {
            VideoView video = (VideoView) findViewById(R.id.video);
            video.setVideoURI(data.getData()); ❸
            video.start(); ❹
        }
    }
}
```

เมื่อปุ่มถูกคลิก เราจะสร้างอินเทนต์ที่เป็นแอคชั่น `MediaStore.ACTION_VIDEO_CAPTURE` ❶ แล้วรันแอคทิวิตีที่สามารถจัดการอินเทนต์นี้ได้ ❷ ซึ่งปกติก็คือแอพกล้องวิดีโอ

และเช่นเคย การรันแอคทิวิตีด้วยเมธอด `startActivityForResult` นั้นจะต้อง Override เมธอด `onActivityResult` เพื่อรับผลลัพธ์ที่แอคทิวิตีปลายทาง (แอพกล้องวิดีโอ) ส่งกลับมาให้ โดยหลังจากผู้ใช้ถ่ายวิดีโอเสร็จและตกลงว่าจะเก็บวิดีโอนั้นไว้แล้ว แอนดรอยด์จะเก็บวิดีโอลงใน Media Store แล้วส่ง URI ของวิดีโอกลับมาให้ทางอินเทนต์ (พารามิเตอร์ `data`) เราจะนำ URI นี้มากำหนดให้ `VideoView` โดยใช้เมธอด `setVideoURI` ❸ แล้วเรียกเมธอด `start` ของ `VideoView` เพื่อเล่นวิดีโอออกมา ❹

ผลการรัน



ตัวเลือกในการถ่ายวิดีโอ

อินเทนต์ที่ใช้ถ่ายวิดีโอยังมีตัวเลือกให้กำหนดได้ดังนี้

- `MediaStore.EXTRA_OUTPUT` ปกติวิดีโอที่ถูกถ่ายด้วยอินเทนต์ `ACTION_VIDEO_CAPTURE` จะถูกเก็บลงใน Media Store (ตำแหน่งที่เป็นดีฟอลต์ของเครื่อง โดยใช้ชื่อไฟล์ที่เป็นดีฟอลต์) แต่หากต้องการให้เก็บที่อื่นก็สามารถระบุ URI ที่ตัวเลือกนี้ได้ เช่นตัวอย่างนี้จะกำหนดให้บันทึกวิดีโอเป็นไฟล์ชื่อ video.mp4 ในรูทไดเร็คทอรีของ SD card

```
Intent intent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);

// สร้างไฟล์ผลลัพธ์เตรียมไว้ เพื่อให้แอพกล้องวิดีโอบันทึกข้อมูลลงในไฟล์
outputFile = new File(Environment.getExternalStorageDirectory(),
                      "video.mp4");
// ส่ง URI ของไฟล์ไปทางอินเทนต์
intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(outputFile));

startActivityForResult(intent, RECORD_VIDEO);
```

- `MediaStore.EXTRA_VIDEO_QUALITY` กำหนดคุณภาพของวิดีโอ โดย 0 หมายถึงวิดีโอคุณภาพต่ำ (สำหรับ MMS) และ 1 หมายถึงวิดีโอคุณภาพสูง ค่าดีฟอลต์คือวิดีโอคุณภาพสูง
- `MediaStore.EXTRA_DURATION_LIMIT` จำกัดความยาวสูงสุดของวิดีโอในหน่วยวินาที

การถ่ายวิดีโอโดยใช้ MediaRecorder

การถ่ายวิดีโอในหัวข้อนี้จะไม่ง้อแอพกล้องวิดีโอ เพราะว่าแอพของเราจะทำงานเป็นแอพกล้องวิดีโอเสียเอง โดยใช้ความสามารถของ `MediaRecorder` ที่ใช้บันทึกเสียงก่อนหน้านี้ ร่วมกับ `SurfaceView`

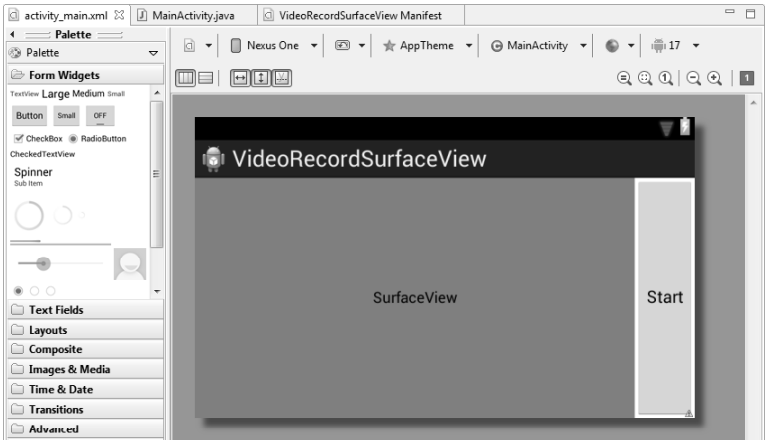
ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะถ่ายวิดีโอเก็บลงไฟล์ video.mp4 ในรูทไดเร็คทอรีของ SD card โดยขั้นตอนการเขียนโค้ดในภาพรวมจะคล้ายกับตัวอย่างการบันทึกเสียง แต่จะเพิ่ม `SurfaceView` สำหรับแสดงภาพพรีวิวจากกล้อง และโค้ดที่ใช้กำหนดรายละเอียดเกี่ยวกับวิดีโอเข้ามา

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค VideoRecordSurfaceView, ไฟล์ activity_main.xml

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

    <SurfaceView
        android:id="@+id/camera_preview"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1" />

    <Button
        android:id="@+id/start_stop_button"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:text="Start" />

</LinearLayout>
```



ในตัวอย่างการบันทึกเสียงเราสร้างปุ่ม Start และ Stop แยกคนละปุ่ม แต่ตัวอย่างนี้จะลดเหลือปุ่มเดียวเพื่อประหยัดพื้นที่หน้าจอ (จะได้เหลือพื้นที่ให้ `SurfaceView` มากหน่อย) โดยทำหน้าที่เป็นปุ่ม Start และ Stop สลับกันไป

2 เพิ่มเติมโค้ดในแอคทิวิตีจนเป็นดังนี้

โปรเจ็ค VideoRecordSurfaceView, ไฟล์ MainActivity.java

```
package com.example.videorecordsurfaceview;

import java.io.File;

import android.app.Activity;
import android.media.CamcorderProfile;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.os.Environment;
import android.view.SurfaceView;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {

    private Button mButton;
    private MediaRecorder mRecorder;
    private File mOutputFile;
    private boolean mIsRecording = false;  // สถานะที่บอกว่าขณะนั้นถ่ายวิดีโออยู่หรือไม่

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

        // กำหนด Listener ให้กับปุ่ม
        mButton = (Button) findViewById(R.id.start_stop_button);
        mButton.setOnClickListener(startListener);

        // สร้างออบเจ็ค MediaRecorder และกำหนดให้แสดงภาพพรีวิวจากกล้องที่ SurfaceView
        mRecorder = new MediaRecorder();
        SurfaceView preview = (SurfaceView)
                              findViewById(R.id.camera_preview);
        mRecorder.setPreviewDisplay(preview.getHolder().getSurface());
```

```
    // สร้างไฟล์ผลลัพธ์เตรียมไว้ (เราจะกำหนดให้ MediaRecorder บันทึกข้อมูลวิดีโอลงในไฟล์นี้)
    mOutputFile = new File(Environment.getExternalStorageDirectory(),
                           "video.mp4");
}

// ปลดปล่อย MediaRecorder ในช่วงที่แอคทิวิตีกำลังจะถูกทำลาย
@Override
public void onDestroy() {
    super.onDestroy();
    if (mRecorder != null) {
        mRecorder.release();
        mRecorder = null;
    }
}

// Listener สำหรับปุ่ม Start/Stop (ปุ่มเดียวทำหน้าที่ทั้ง Start และ Stop)
private View.OnClickListener startListener =
                              new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        /* ถ้าถ่ายวิดีโออยู่จะหยุดถ่ายและกำหนดข้อความบนปุ่มเป็น Start แต่ถ้าไม่ได้ถ่ายวิดีโออยู่
           ก็จะเริ่มถ่ายและกำหนดข้อความบนปุ่มเป็น Stop */
        if (mIsRecording == true) {
            mRecorder.stop();
            mButton.setText("Start");
            mIsRecording = false;
        } else {
            setupRecorder();           // ตั้งค่าการถ่ายวิดีโอ

            try {
                mRecorder.prepare();  // เตรียมถ่ายวิดีโอ
            } catch (Exception e) {
                e.printStackTrace();
            }

            mRecorder.start();         // เริ่มถ่ายวิดีโอ
            mButton.setText("Stop");
            mIsRecording = true;
        }
    }
};
```

```
    // เมธอดสำหรับตั้งค่าการถ่ายวิดีโอ
    private void setupRecorder() {
        // กำหนดแหล่งข้อมูลของเสียงและภาพ โดยรับเสียงจากไมโครโฟน และรับภาพจากกล้อง
        mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        mRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);

        /* กำหนดคุณภาพของวิดีโอ โดยตรวจสอบว่าสนับสนุนคุณภาพแบบใดแล้วเลือกแบบนั้น ไล่จาก
           คุณภาพสูงไปต่ำ (โค้ดส่วนนี้ใช้กับแอนดรอยด์ 3.0 หรือ API Level 11 ขึ้นไป) */
        CamcorderProfile profile = null;

        if (CamcorderProfile.hasProfile(CamcorderProfile.QUALITY_1080P))
            profile = CamcorderProfile.get(CamcorderProfile.QUALITY_1080P);
        else if (CamcorderProfile.hasProfile(CamcorderProfile.QUALITY_720P))
            profile = CamcorderProfile.get(CamcorderProfile.QUALITY_720P);
        else if (CamcorderProfile.hasProfile(CamcorderProfile.QUALITY_480P))
            profile = CamcorderProfile.get(CamcorderProfile.QUALITY_480P);
        else if (CamcorderProfile.hasProfile(CamcorderProfile.QUALITY_HIGH))
            profile = CamcorderProfile.get(CamcorderProfile.QUALITY_HIGH);

        if (profile != null)
            mRecorder.setProfile(profile);

        // กำหนดพาธของไฟล์ผลลัพธ์
        mRecorder.setOutputFile(mOutputFile.getAbsolutePath());
    }
}
```

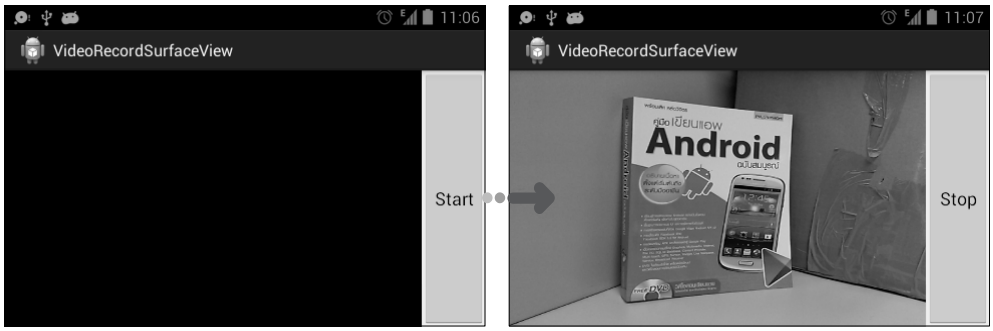
3 ที่ไฟล์ AndroidManifest.xml ให้กำหนด `minSdkVersion` เป็น 11 หรือสูงกว่า, เพิ่มการขอสิทธิ์ต่างๆที่จำเป็น และกำหนดทิศทางของแอคทิวิตีเป็นแนวนอนตายตัว

โปรเจ็ค VideoRecordSurfaceView, ไฟล์ AndroidManifest.xml

```
...
<uses-sdk
    android:minSdkVersion="11"
    android:targetSdkVersion="17" />

<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.RECORD_VIDEO" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

<application
    ... >
    <activity
        android:name="com.example.videorecordsurfaceview.MainActivity"
```

```
        android:label="@string/app_name"
        android:screenOrientation="landscape" >
        ...
    </activity>
</application>
...
```

ผลการรัน



หลังจาก Stop แล้วจะได้ไฟล์ video.mp4 เพิ่มขึ้นมาที่ SD card ดังรูป