第4章　图形与多媒体处理技术

2024年4月

4.1 绘制几何图形

4.1.1几何图形绘制类

- 在Android系统中绘制几何图形，需要用到一些绘图工具，这些绘图工具都在android.graphics包中。

1、画布Canvas

方 法	功 能
Canvas()	创建一个空的画布，可以使用setBitmap()方法来设置绘制具体的画布。
Canvas(Bitmap bitmap)	以bitmap对象创建一个画布，则将内容都绘制在bitmap上，bitmap不得为null。
drawColor()	设置Canvas的背景颜色。
setBitmap()	设置具体画布。
clipRect()	设置显示区域，即设置裁剪区。
rotate()	旋转画布
skew()	设置偏移量。
drawLine(float x1, float y1, float x2, float y2)	从点(x1, y1)到点(x2, y2)的直线。
drawCircle(float x, float y, float radius, Paint paint)	以(x, y)为圆心， radius为半径画圆。
drawRect(float x1, float y1, float x2, float y2, Paint paint)	从左上角(x1, y1)到右下角(x2, y2)的矩形。
drawText(String text, float x, float y ,Paint paint)	写文字。
drawPath(Path path, Paint paint)	从一点到另一点的连接路径线段。

2、画笔Paint

- 画笔Paint用来描述所绘制图形的颜色、和风格，如线条宽度、颜色等信息。

方 法	功 能
Paint()	构造方法，创建一个辅助画笔对象。
setColor(int color)	设置颜色。
setStrokeWidth(**float width**)	设置画笔宽度。
setTextSize(float textSize)	设置文字尺寸。
setAlpha(int a)	设置透明度alpha值。
setAntiAlias(boolean b)	除去边缘锯齿，取true值。
paint.setStyle(Paint.Style style)	设置图形为空心(Paint.Style.STROKE)或实心(Paint.Style.FILL)。

3、点到点的连线路径Path

- 当绘制由一些线段组成的图形（如：三角形、四边形等），需要用Path类来描述线段路径。

方 法	功 能
lineTo(**float x, float y**)	从当前点到指定点画连线。
moveTo(**float x, float y**)	移动到指定点。
close()	关闭绘制连线路径

4.1.2几何图形绘制过程

在Android中绘制几何图形的一般过程为：

（1）创建一个View的子类，并重写View类的onDraw()方法；

（2）在View的子类视图中使用画布对象Canvas绘制各种图形；

（3）使用invalidate()方法刷新画面。

这个invalidate()方法，其实是向操作系统发送一条消息WM_PAINT,这个消息自动会调用OnPaint()方法，OnPaint再调用onDraw()方法。所以我们重写OnDraw()方法来实现重绘功能。

【例4-1】绘制几何图形示例。

- 本例继承自Android.view.View的 TestView类，重写 View类的onDraw（）方法，在onDraw（）方法中运用Paint对象（绘笔）的不同设置值，在Cavas(画布)上绘制图形，分别绘制了矩形、圆形、三角形和文字。

【例4-1】绘制几何图形示例。

```
1  package com.ex04_01;
2  import android.app.Activity;
3  import android.content.Context;
4  import android.graphics.Canvas;
5  import android.graphics.Color;
6  import android.graphics.Paint;
7  import android.graphics.Path;
8  import android.os.Bundle;
9  import android.view.View;
10 public class MainActivity extends Activity
11 {
12   @Override
13   public void onCreate(Bundle
savedInstanceState)
14   {
15       super.onCreate(savedInstanceState);
16       TestView tView=new TestView(this);
17       setContentView(tView);
18   }
19   private class TestView extends View
20   {
21     public TestView(Context context)
22     {
23          super(context);
24          }
25          /*重写onDraw（）*/
26     @Override
27     protected void onDraw(Canvas canvas)
28     {
29       /*设置背景为青色*/
30       canvas.drawColor(Color.CYAN);
31       Paint paint=new Paint();
32       /*设置画笔宽度*/
33  paint.setStrokeWidth(3);
34       /*设置画空心图形*/
35       paint.setStyle(Paint.Style.STROKE);
36       /*去锯齿*/
37       paint.setAntiAlias(true);
38       /*画空心矩形（正方形）*/
39       canvas.drawRect(10,10,70,70,paint);
40       /*设置画实心图形*/
41       paint.setStyle(Paint.Style.FILL);
42       /*画实心矩形（正方形）*/
43       canvas.drawRect(100,10,170,70,paint);
44       /*设置画笔颜色为蓝色*/
45       paint.setColor(Color.BLUE);
46       /*画圆心为（100，120），半径为30的实心圆*/
47      canvas.drawCircle(100,120,30,paint);
48      /*在上面的实心圆上画一个小白点*/
49      paint.setColor(Color.WHITE);
50      canvas.drawCircle(91,111,6,paint);
```

```
51        /*设置画笔颜色为红色*/
52        paint.setColor(Color.RED);
53        /*画三角形*/
54        Path path=new Path();
55        path.moveTo(100, 170);
56        path.lineTo(70, 230);
57        path.lineTo(130,230);
58        path.close();
59        canvas.drawPath(path,paint);
60        /*文字*/
61        paint.setTextSize(28);
62        paint.setColor(Color.BLUE);
63
canvas.drawText(getResources().getString(R.string.
hello_world),
64                        30,270,paint);
65      }
66    }
67 }
```

【例4-2】绘制一个可以在任意指定位置显示的小球。

- 设计思想：Android系统应用程序的设计模式是采用MVC模式，即把应用程序分为表现层（View）、控制层（Control）、业务模型层（Model）。在本示例中，按照这种模式，图形界面布局为表现层，Activity控制程序为控制层，实现几何作图的绘制过程属于业务模型层。在业务模型层，将圆心坐标设为（x, y），则圆的位置随控制层任意输入的坐标值而改变。


6:00
ex4_2
输入位置：
120
确定

（1）表现层的图形界面布局程序activity_main.xml

```
1  <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/
android"
2    android:layout_width="fill_parent"
3    android:layout_height="fill_parent"
4    android:orientation="vertical">
5   <LinearLayout
6     android:layout_width="wrap_content"
7     android:layout_height="wrap_content">
8     <TextView
9         android:id="@+id/ textView1 "
10        android:layout_width="wrap_content "
11        android:layout_height="wrap_content"
12        android:text="输入位置： "
13        />
14    <EditText
15        android:id="@+id/ editText1 "
16        android:layout_width="120dp"
17
android:layout_height="wrap_content"
18        />
19    <Button
20        android:id="@+id/button1"
21        android:layout_width="wrap_content
"
22
android:layout_height="wrap_content"
23        android:text="确定"
24        />
25    </LinearLayout>
26    <com.example.ex4_2.TestView
27        android:id="@+id/testView1 "
28        android:layout_width="match_parent
"
29
android:layout_height="match_parent "
```

（2）控制层的主控程序MainActivity.java

```
1  package com.example.ex4_2;
2  import android.os.Bundle;
3  import android.view.View;
4  import android.view.View.OnClickListener;
5  import android.widget.Button;
6  import android.widget.EditText;
7  import android.app.Activity;
8  public class MainActivity extends Activity
9  {
10          int x1=150,y1=50;
11   TestView testView;
12   Button btn;
13   EditText  edit_y;
14   @Override
15   public void onCreate(Bundle savedInstanceState)
16   {
17       super.onCreate(savedInstanceState);
18       setContentView(R.layout.activity_main);
19
testView=(TestView)findViewById(R.id.testView1);
20       testView.setXY(x1, y1);
21       btn=(Button)findViewById(R.id.button1);
22       edit_y=(EditText)findViewById(R.id.editText1);
23       btn.setOnClickListener(new mClick());
24   }
25   class mClick implements OnClickListener
26   {
27          @Override
28          public void onClick(View arg0)
29          {
30                  y1 =
Integer.parseInt(edit_y.getText().toString());
31                  testView.setXY(x1, y1);
32                  testView.invalidate();
33                 }
34   }
35 }
```

[说明]：程序第30行

y1 = Integer.parseInt(edit_y.getText().toString());

方法Integer.parseInt(String)为将字符串String转换为整型数据。

（3）业务逻辑层的绘制小球程序TestView.java

```
1  package com.example.ex4_2;
2  import android.util.AttributeSet;
3  import android.view.View;
4  import android.content.Context;
5  import android.graphics.Canvas;
6  import android.graphics.Color;
7  import android.graphics.Paint;
8
9   public class TestView extends View
10  {
11     int x, y;
12     public TestView(Context context, AttributeSet
attrs)
13     {
14        super(context, attrs);
15     }
16     void setXY(int _x, int _y)
17     {
18           x = _x;
19           y = _y;
20     }
21       @Override
22     protected void onDraw(Canvas canvas)
23     {
24           super.onDraw(canvas);
25       /*设置背景为青色*/
26       canvas.drawColor(Color.CYAN);
27       Paint paint=new Paint();
28       /*去锯齿*/
29       paint.setAntiAlias(true);
30       /*设置paint的颜色*/
31       paint.setColor(Color.BLACK);
32       /*画一个实心圆*/
33       canvas.drawCircle(x, y, 15, paint);
34       /*画一个实心圆上的小白点*/
35       paint.setColor(Color.WHITE);
36       canvas.drawCircle(x-6, y-6, 3, paint);
37     }
38  }
```

4.1.3 自定义组件

- 在Android中，可以通过view类的子类自定义组件，然后再添加到布局界面中。

【例4-3】自定义一个组件，再通过布局界面显示出来。

主要设计步骤如下：

（1）编写View子类：TestView.java

（2）把TestView添加到布局界面中

（3）在主程序MainActivity.java中建立TestView对象与布局文件的关联。

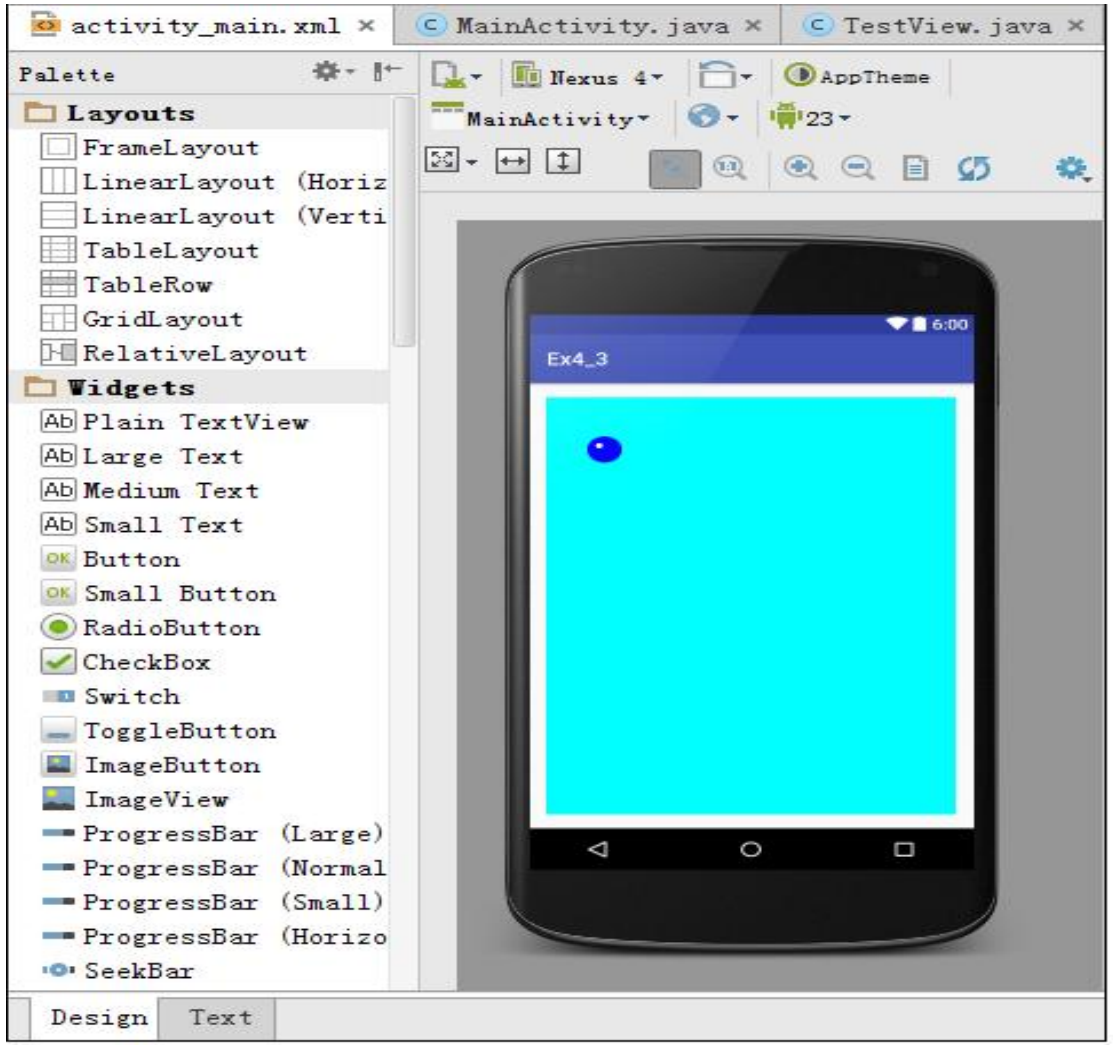
1. 新建java文件，编写View子类：TestView.java

```
class TestView extends View
 {
    public TestView(Context context, AttributeSet attrs)
     {   super(context, attrs);  }
    /* 重写View的抽象方法 onDraw()方法  */
    protected void onDraw(Canvas canvas)
    {
     canvas.drawColor(Color.CYAN);   //设置组件的背景颜色为青色
     Paint paint=new Paint();          //定义画笔
     paint.setStyle(Paint.Style.FILL); //设置画实心图形
     paint.setAntiAlias(true);       //去锯齿
     paint.setColor(Color.BLUE); /*设置画笔颜色为蓝色*/
     canvas.drawCircle(100,120,30,paint); /*圆心为 (100, 120), 半径为30的实心圆*/
     paint.setColor(Color.WHITE); /*在上面的实心圆上画一个小白点*/
     canvas.drawCircle(91,111,6,paint);
    }
}
```

2. 在表现层布局文件activity_main.xml中，添加所设计的组件TestView类

```
<com.example.ex4_3.TestView
        android:id="@+id/testview1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
 />
```

这时，可以在编辑器的预览中看到所自定义的组件.



3. 在主程序MainActivity.java中建立TestView对象与布局文件的关联

```
package com.example.ex4_3;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends Activity {
    TestView tView = null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        tView =(TestView)findViewById(R.id.testview1);
        setContentView(R.layout.activity_main);
    }
}
```

4.2 触摸屏事件处理

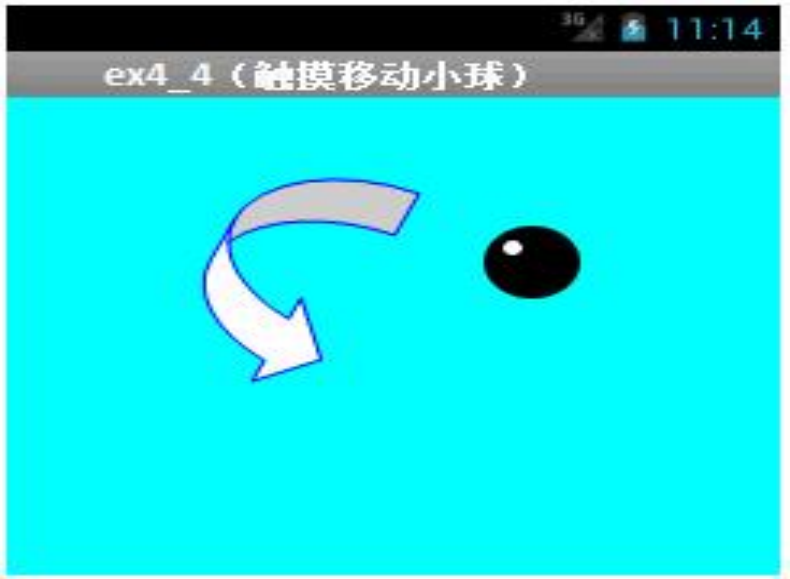
4.2.1 简单触摸屏事件

- 简单触摸屏事件指的是在触摸屏按下、抬起、移动事件（模拟器中为鼠标事件）。在Android系统中，通过OnTouchListener监听接口来处理屏幕事件，当在View的范围内触摸按下、抬起或滑动等动作时都会触发该事件。

- 在设计简单触摸屏事件程序时，要实现android.view.View.OnTouchListener接口，并重写该接口的监听方法onTouch(View v, MotionEvent event)。
- 在监听方法onTouch(View v, MotionEvent event)中，参数v为事件源对象；参数event为事件对象，事件对象为下列常数之一：
  - MotionEvent.ACTION_DOWN　按下；
  - MotionEvent.ACTION_UP　抬起；
  - MotionEvent.ACTION_MOVE　移动。

【例4-4】设计一个在屏幕上移动小球的程序。

- 设计一个继承于Android.view.View的图形绘制视图 TestView，在该视图中绘制一个小球。再设计一个实现OnTouchListener监听接口的类，重写该接口的监听方法onTouch(View v, MotionEvent event)，该方法监听并获取触摸屏幕的坐标位置，并把坐标值传递给图形绘制类TestView，由TestView在该位置重绘小球。



（1）图形绘制类TestView.java（自定义组件）

```
1  package com.example.ex4_4;
2  import   (略)
3  class TestView extends View
4  {
5     int x=150,y=50;
6     public TestView(Context context, AttributeSet attrs)
7               {  super(context, attrs);  }
8     void setXY(int _x, int _y)
9     {
10              x = _x;
11              y = _y;
12     }
13     @Override
14    protected void onDraw(Canvas canvas)
15    {
16        super.onDraw(canvas);
17        canvas.drawColor(Color.CYAN); /*设置背景为青色*/
18        Paint paint=new Paint();
19        paint.setAntiAlias(true); /*去锯齿*/
20        paint.setColor(Color.BLACK); /*设置paint的颜色*/
21        canvas.drawCircle(x, y, 30, paint); /*画一个实心圆*/
22        paint.setColor(Color.WHITE);/*画一个实心圆上的小
白点*/
23        canvas.drawCircle(x-9, y-9, 6, paint);
24     }
```

```
25   private class mOnTouch implements OnTouchListener
26{
27              public boolean onTouch(View v, MotionEvent event
28              {
29                 if (event.getAction() ==
MotionEvent.ACTION_MOVE)
30                    {
31                       x1 = (int) event.getX();
32                       y1 = (int) event.getY();
33                            testView.setXY(x1, y1);
34                             setContentView(testView);//按新从标绘图
35                    }
36                 if (event.getAction() ==
MotionEvent.ACTION_DOWN)
37                    {
38                     x1 = (int) event.getX();
39                      y1 = (int) event.getY();

40                      testView.setXY(x1, y1);
41                          setContentView(testView);
42                  }
43                 return true;
44              }
45   }
```

（2）把自定义组件添加到布局文件activity_main.xml中

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/andro
id"
3    xmlns:tools="http://schemas.android.com/tools"
4    android:layout_width="match_parent"
5    android:layout_height="match_parent"
6
   android:paddingBottom="@dimen/activity_vertical_margi
   n“
7
android:paddingLeft="@dimen/activity_horizontal_margin"
8
android:paddingRight="@dimen/activity_horizontal_margin"
9    android:paddingTop="@dimen/activity_vertical_margin"
10   tools:context="com.example.ex4_4.MainActivity">
11 <com.example.ex4_4.TestView                %添加自定义
   组件
12     android:id="@+id/testview1"
13      android:layout_width="wrap_content"
14     android:layout_height="wrap_content“
15    />
16 </RelativeLayout>
```

（3）主程序MainActivity.java

```
1  package com.example.ex4_4;
2  import
android.support.v7.app.AppCompatActivity;
3  import android.os.Bundle;
4  public class MainActivity extends Activity {
5      TestView tView = null;
6      @Override
7      public void onCreate(Bundle
savedInstanceState) {
8          super.onCreate(savedInstanceState);
9          tView
=(TestView)findViewById(R.id.testview1);
10          setContentView(R.layout.activity_main);
11     }
12  }
```

【例4-5】设计一个能在图片上涂鸦的程序。

（原图）

（涂鸦）

4.2.2 手势识别

- 所谓手势识别，就是识别手指（或鼠标）在屏幕上划动时的轨迹。在Android系统中，android.gesture是用于创建、识别和保存触摸屏手势功能的包。
- 在实现OnGesturePerformedListener接口时，需要复盖其方法：

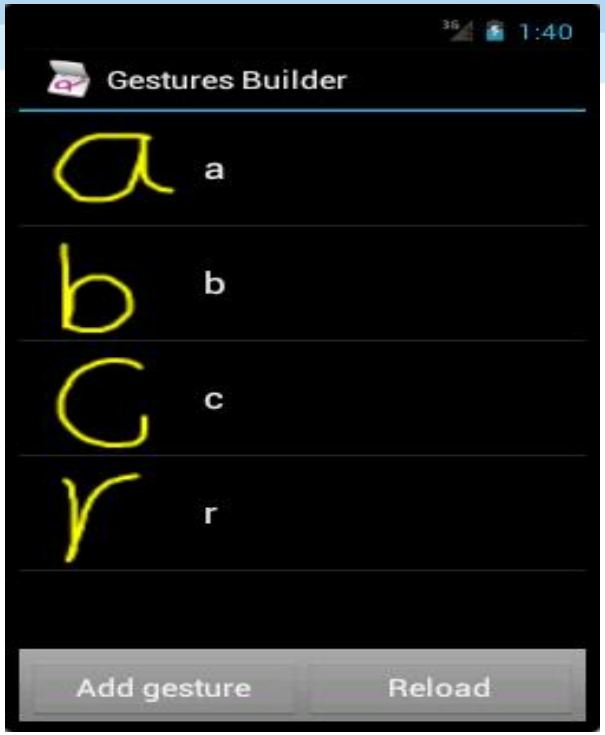
onGesturePerformed(GestureOverlayView overlay, Gesture gesture)

类及接口	功 能
Gesture	触摸屏的手势类
GestureLibraries	手势库
GestureLibrary	手势库
GestureOverlayView	可输入手势的视图
Prediction	手势的预显示类
GestureStroke	记录在触摸屏上手势动作的开始与结束类
OnGestureListener	手势动作的监听接口
OnGesturePerformedListener	可输入手势视图GestureOverlayView的监听接口

【例4-6】设计一个手写字体识别程序。

- 要编写一个手写字体识别程序，必须先建立一个存放手写字体的数据库。
- 在手机模拟器中已经预装了创建手写字体数据库的应用程序Gestures Builder，其图标如图所示。

- 创建手势库如图4.7所示。由手势创建的手写字体将被保存到/sdcard/gestures中，把文件gestures复制到项目/res/raw下，就可以在应用程序里面使用这些手势了。
-



1）界面布局文件activity_main.xml

在界面布局文件activity_main.xml中设置android.gesture.GestureOverlayView组件。其代码如下：

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3    android:layout_width="fill_parent"
4    android:layout_height="fill_parent"
5    android:orientation="vertical" >
6    <TextView
7      android:id="@+id/textView1"
8      android:layout_width="fill_parent"
9      android:layout_height="wrap_content"
10     android:text="@string/text1"
11     android:textSize="24sp"/>
12    <!-- 绘制手势的GestureOverlayView   -->
13  <android.gesture.GestureOverlayView
14     android:id="@+id/gestures1"
15     android:layout_width="fill_parent"
16     android:layout_height="fill_parent"
17     android:gestureStrokeType="multiple"
18     android:eventsInterceptionEnabled="false"
19     android:orientation="vertical"/>
20 </LinearLayout>
```

（2）控制程序MainActivity.java

```
1  package com.ex4_6;
2  import java.util.ArrayList;
3  import android.app.Activity;
4  import android.gesture.Gesture;
5  import android.gesture.GestureLibraries;
6  import android.gesture.GestureLibrary;
7  import android.gesture.GestureOverlayView;
8  import android.gesture.Prediction;
9  import
   android.gesture.GestureOverlayView.OnGesturePe
   rformedListener;
10 import android.os.Bundle;
11 import android.widget.TextView;
12 import android.widget.Toast;
13 public class MainActivity extends Activity
13 implements OnGesturePerformedListener
14 {
15          GestureLibrary mLibrary;
16  GestureOverlayView gesturesView;
17   TextView txt;
18   @Override
19  public void onCreate(Bundle
savedInstanceState)
20  {
21   super.onCreate(savedInstanceState);
22   setContentView(R.layout.main);
23
gesturesView=(GestureOverlayView)findViewById(R.i
d.gestures);
24
gesturesView.addOnGesturePerformedListener(this)
;
25   txt = (TextView)findViewById(R.id.textView1);
26   mLibrary =
GestureLibraries.fromRawResource(this,
27                                  R.raw.gestures);
28   if(!mLibrary.load())
29    {
30      finish();
31    }                                  //加载手势库

32  }
```

```
33/*  根据在GestureOverlayView上画的手势来识别是否匹配手势库里的手势   */
34@Override
35public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture)
36{
37 ArrayList predictions=mLibrary.recognize(gesture);
38   if(predictions.size()>0)
39 {
40   Prediction prediction = (Prediction)predictions.get(0);
41   if(prediction.score > 1.0)
42    {
43      Toast.makeText(this,prediction.name,Toast.LENGTH_SHORT).show();
44      txt.append(prediction.name);
45            }
46    }
47 }
48  }
```

4.3 音频播放

4.3.1 多媒体处理包

android.media包中的主要类

类名或接口名	说 明
MediaPlayer	支持流媒体，用于播放音频和视频
MediaRecorder	用于录制音频和视频
Ringtone	用于播放可用作铃声和提示音的短声音片段
AudioManager	负责控制音量
AudioRecord	用于记录从音频输入设备产生的数据
JetPlayer	用于存储JET内容的回放和控制
RingtoneManager	用于访问响铃、通知和其他类型的声音
Ringtone	快速播放响铃、通知或其他相同类型的声音
SoundPool	用于管理和播放应用程序的音频资源

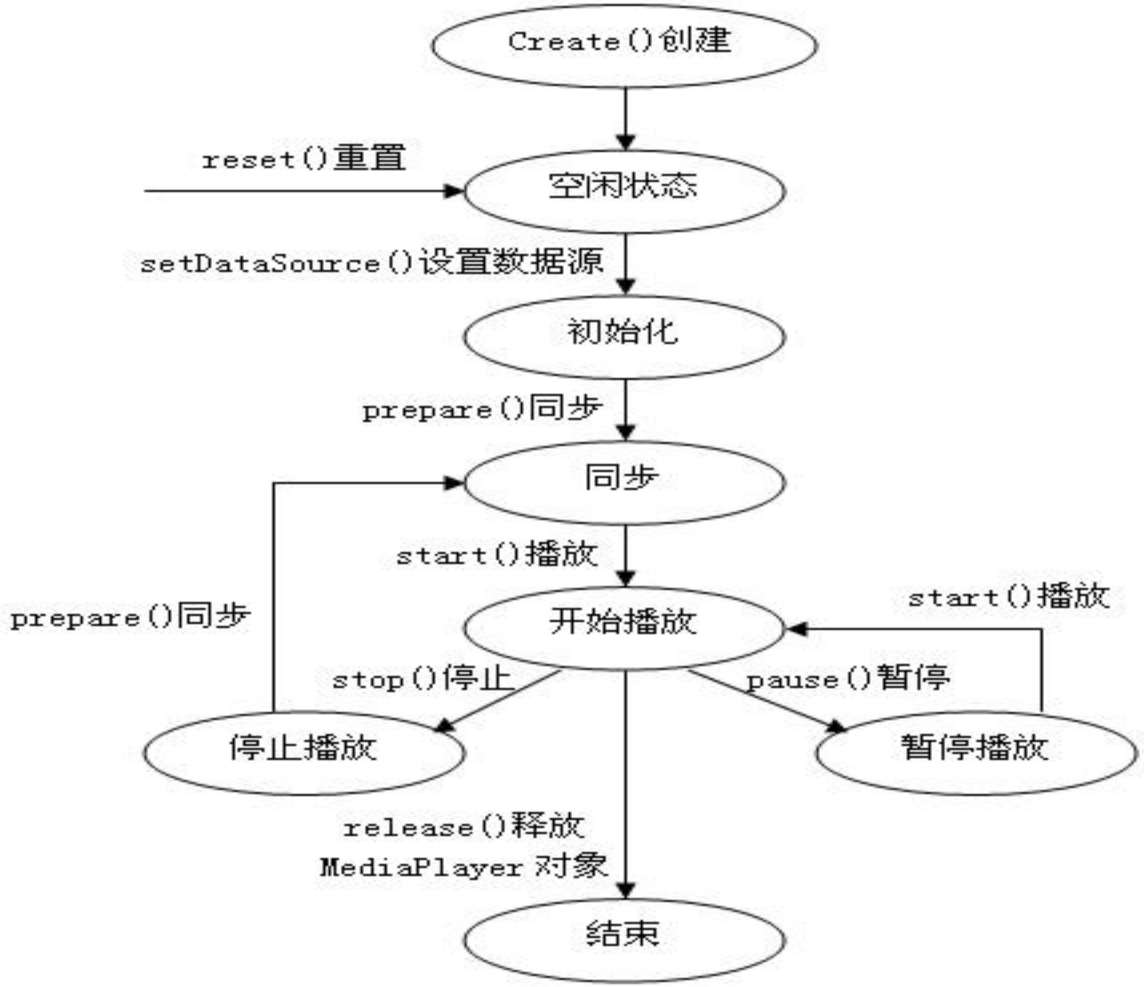
4.3.2　多媒体处理播放器MediaPlayer

- 1、MediaPlayer类的常用方法

方 法	说 明
create( )	创建多媒体播放器。
getCurrentPosition()	获得当前播放位置
getDuration()	获得播放文件的时间
getVideoHeight()	播放视频高度
getVideoWidth()	播放视频宽度
isLooping()	是否循环播放
isPlaying()	是否正在播放
Pause()	暂停
Prepare()	准备播放文件，进行同步处理
prepareAsync()	准备播放文件，进行异步处理
release()	释放MediaPlayer对象
reset()	重置MediaPlayer对象
seekTo()	指定播放文件的播放位置
setDataSource()	设置多媒体数据来源
setVolume()	设置音量
setOnCompletionListener()	监听播放文件播放完毕
start()	开始播放
stop()	停止播放

2、MediaPlayer对象的生命周期

- 通常把一个对象从创建、使用、直到释放该对象的过程称为该对象的生命周期。



4.3.3　播放音频文件

- 通过媒体处理器MediaPlayer提供的方法不仅可以播放存放在SD卡上音乐文件而且还能播放资源中的音乐文件。这二者之间在设计方法上稍有不同。

1、构建MediaPlayer对象

（1）使用new的方式创建MediaPlayer对象

对于播放SD卡上的音乐文件需要使用new方式来创建MediaPlayer对象：

```
MediaPlayer mplayer = new MediaPlayer();
```

（2）使用create方法的方式创建MediaPlayer对象

对于播放资源中的音乐需要使用create方法的方式来创建MediaPlayer对象，如：

```
MediaPlayer mplayer = MediaPlayer.create(this, R.raw.test);
```

2、设置播放文件

MediaPlayer要播放的文件主要包括3个来源：

（1）存储在SD卡或其他文件路径下的媒体文件

对于存储在SD卡或其他文件路径下的媒体文件，需要调用setDataSource（）方法，例如：

```
mplayer.setDataSource("/sdcard/test.mp3");
```

(2）在编写应用程序时事先存放在res资源中的音乐文件

播放事先存放在资源目录res\raw中的音乐文件，需要在使用create（）方法创建MediaPlayer对象时，就指定资源路径和文件名称（不要带扩展名）。由于create（）方法的源代码中已经封装了调用setDataSource（）方法，因此不必重复使用setDataSource（）方法。

(3) 网络上的媒体文件

播放网络上的音乐文件，需要调用setDataSource（）方法，例如：

```
mplayer.setDataSource("http://www.citynorth.cn/music/confucius.mp3");
```

3、对播放器进行同步控制

使用prepare()方法设置对播放器的同步控制，例如：

```
mplayer.prepare();
```

- 如果MediaPlayer对象是由create方法创建的，由于create（）方法的源代码中已经封装了调用prepare（）方法，因此可省略此步骤。

4、播放音频文件

start()是真正启动音频文件播放的方法，如：

```
mplayer.start();
```

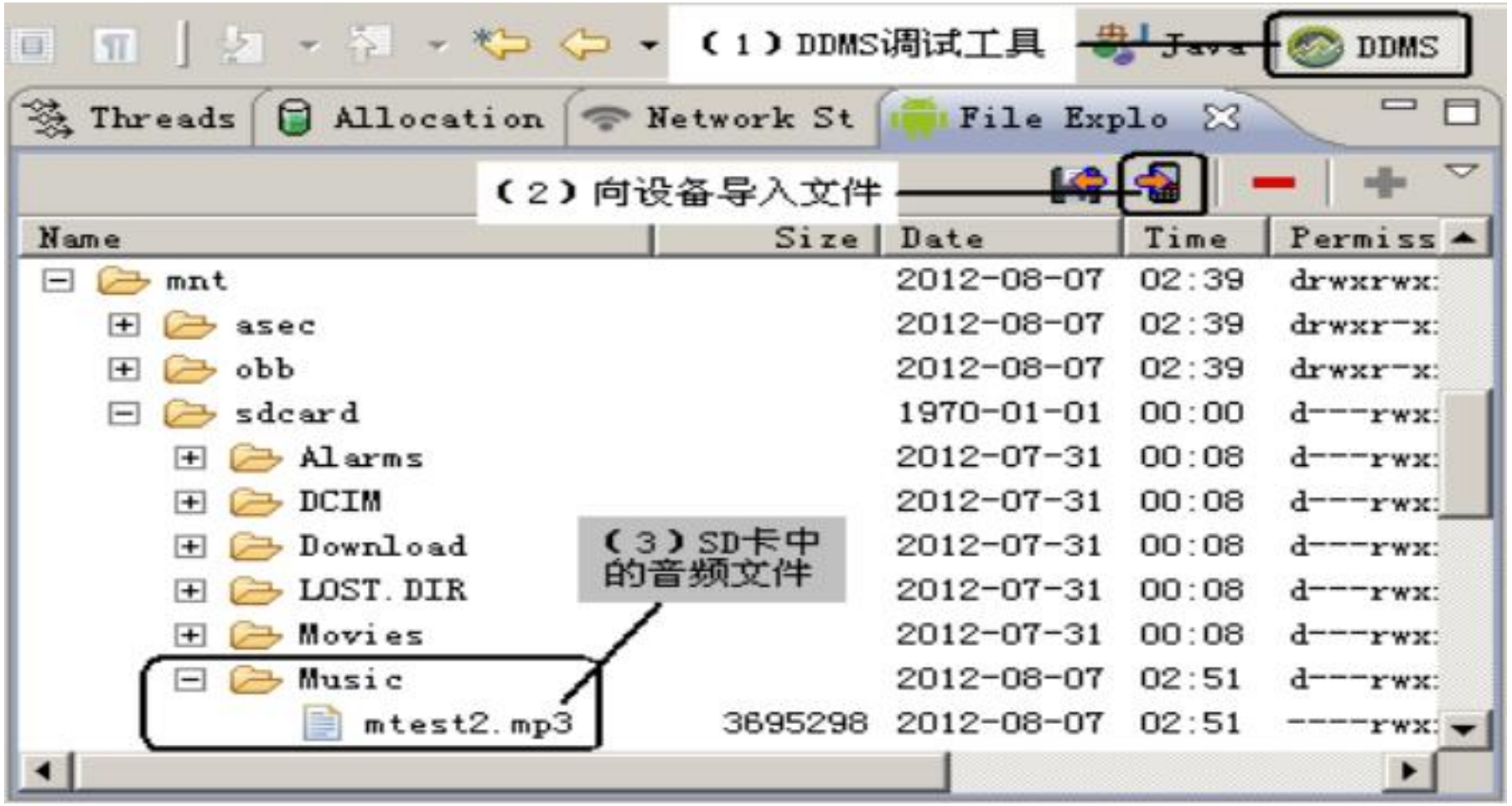
如要暂停播放或停止播放，则调用 pause()和stop()方法。

5、释放占用资源

- 音频文件播放结束应该调用release()释放播放器占用的系统资源。
- 如要重新播放音频文件，需要调用reset()返回到空闲状态，再从第2步开始重复其他各步骤。

【例4-7】设计一个音乐播放器。

- 在本示例中，将分别播放存放在项目资源中的音乐文件和SD卡中的音频文件，因此需要事先将准备好的音频文件保存在指定路径下。
- （1）将测试的音频文件mtest1.mp3复制到新建项目的“res\raw”目录下；
- （2）将音频文件mtest2.mp3复制到SD卡中，（在模拟器中使用SD卡，可以在Eclipse集成环境中，选择“DDMS”调试工具，单击“向设备导入文件”按键，将音频文件复制到模拟器的“/mnt/sdcard/Music”目录下，如图6.9所示）。



将音频文件存放到模拟器的SD卡内

- (代码详见教材)

4.4 视频播放

- 在Android系统中，设计播放视频的应用程序有两种不同方式：
- 一种方式是应用媒体播放器MediaPlayer组件播放视频；
- 另一种方式是应用视频视图VideoView组件播放视频。

4.4.1 应用媒体播放器（MediaPlayer）播放视频

- 媒体播放器MediaPlayer不仅可以播放音频文件，而且还可以播放格式为.3gp的视频文件。
- 与播放音频不同之处为，用于视频播放的播放承载体必须是实现了表面视图处理接口（surfaceHolder）的视图组件，即需要使用SurfaceView组件来显示播放的视频图像。

【例4-8】应用媒体播放器MediaPlayer设计一个视频播放器。

- 事先准备好视频文件sample.3gp，并将其复制到模拟器的SD卡sdcard/zsm目录下。

- （代码详见教材）

4.4.2 应用视频视图（VideoView）播放视频

- 在Android系统中，经常使用android.widget包中的视频视图类VideoView播放视频文件。 VideoView 类可以从不同的来源（例如资源文件或内容提供器） 读取图像，计算和维护视频的画面尺寸以使其适用于任何布局管理器， 并提供一些诸如缩放、着色之类的显示选项。

android.widget.VideoView类的常用方法

方 法	说 明
VideoView(Context context)	创建一个默认属性的VideoView实例。
boolean canPause()	判断是否能够暂停播放视频
int getBufferPercentage()	获得缓冲区的百分比
int getCurrentPosition()	获得当前的位置
int getDuration()	获得所播放视频的总时间
boolean isPlaying()	判断是否正在播放视频
boolean onTouchEvent (MotionEvent ev)	实现该方法来处理触屏事件
seekTo (int msec)	设置播放位置
setMediaController(MediaController controller)	设置媒体控制器
setOnCompletionListener(MediaPlayer.OnCompletionListener l)	注册在媒体文件播放完毕时调用的回调函数
setOnPreparedListener(MediaPlayer.OnPreparedListener l)	注册在媒体文件加载完毕，可以播放时调用的回调函数
setVideoPath(String path)	设置视频文件的路径名
setVideoURI(Uri uri)	设置视频文件的统一资源标识符
start()	开始播放视频文件
stopPlayback()	停止回放视频文件

【例4-9】应用视频视图VideoView组件设计一个视频播放器。

- （代码详见教材）

4.5 录音与拍照

4.5.1 用于录音录像的MediaRecorder类

方法	说明
MediaRecorder()	创建录制媒体对象
setAudioSource(int audio_source)	设置音频源
setAudioEncoder(int audio_encoder)	设置音频编码格式
setVideoSource(int video_source)	设置视频源
setVideoEncoder(int video_encoder)	设置视频编码格式
setVideoFrameRate(int rate)	设置视频帧速率
setVideoSize(int width, int height)	设置视频录制画面大小
setOutputFormat(int output_format)	设置输出格式
setOutputFile(path)	设置输出文件路径
prepare()	录制准备
start()	开始录制
stop()	停止录制
reset()	重置
release()	释放播放器有关资源

4.5.2 录音示例

应用MediaRecorder进行录音频，其主要步骤如下。

（1）创建录音对象

```
MediaRecorder mRecorder = new MediaRecorder();
```

（2）录音对象的设置

设置音频源

```
mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
```

设置输出格式 `mRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);`

设置编码格式 `mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);`

设置输出文件路径

```
mRecorder.setOutputFile(path);
```

(3) 录制准备

mRecorder.prepare();

(4) 开始录制

mRecorder.start();

(5) 录制结束

停止录制

mRecorder.stop();

重置

mRecorder.reset();

释放录音占用的有关资源

mRecorder.release();

【例4-10】设计一个简易录音机。

在配置文件AndroidManifest.xml中要增加音频捕获权限的语句。

音频捕获权限：

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

SD卡的写操作权限：

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

（代码详见教材,真实手机上才能实现录音功能）

4.5.3 拍照

- 使用android.hardware包中的Camera类可以获取当前设备中的照相机服务接口，从而实现照相机的拍照功能。

1、照片服务Camera类

方 法	说 明
open()	创建一个照相机对象
getParameters()	创建设置照相机参数的Camera.Parameters对象
setParameters(Camera.Parameters params)	设置照相机参数
setPreviewDisplay(SurfaceHolder holder)	设置取景预览
startPreview()	启动照片取景预览
stopPreview()	停止照片取景预览
release()	断开与照相机设备的连接，并释放资源
takePicture(Camera.ShutterCallback shutter, Camera.PictureCallback raw, Camera.PictureCallback jpeg)	进行照片拍摄

takePicture方法有3个参数：

- 第1个参数shutter是关闭快门事件的回调接口；
- 第2个参数raw是获取照片事件的回调接口；
- 第3个参数jpeg也是获取照片事件的回调接口。

第2个参数与第3个参数的区别在于回调函数中传回的数据内容。第2个参数指定的回调函数中传回的数据内容是照片的原数据，而第3个参数指定的回调函数中传回的数据内容是已经按照JPEG格式进行编码的数据。

2、实现拍照服务的主要步骤

（1）创建照相机对象

- 通过Camera类的open()方法创建一个照相机对象

```
Camera camera=Camera.open();
```

（2）设置参数

- 创建设置照相机参数的Parameters对象，并设置相关参数

```
parameters = mCamera.getParameters();
```

(3) 对照片预览

- 通过照相机对象的startPreview()方法和stopPreview()方法启动或停止对照片的预览。

(4) 照片拍摄

- 使用照相机接口的takePicture()方法可以异步地进行照片拍摄。
- 通过照片事件的回调接口PictureCallback，可以获取照相机所得到的图片数据，从而可以进行下一步的行动，例如保存到本地存储、进行数据压缩、通过可视组件显示。

(5) 停止照相

- 通过照相机对象的release()方法可以断开与照相机设备的连接，并释放与该照相机接口有关的资源。

```
camera.release();
camera=null;
```

【例4-11】设计一个简易照相机。

设计照相机，为了取景，需要应用SurfaceView组件来显示摄像头所能拍照的景物，再使用回调接口SurfaceHolder.Callback监控取景视图，Callback接口有3个方法需要实现：

- surfaceCreated(SurfaceHolder holder)方法，用于初始化；
- surfaceChanged(SurfaceHolder holder, int format, int width,
- int height)方法，当景物发生变化时触发；
- surfaceDestroyed(SurfaceHolder holder)方法，释放对象时触发。

在配置文件AndroidManifest.xml增加允许操作SD卡和使用摄像头设备的语句：

```
<uses-permission
    android:name="android.permission.CAMERA"/>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-feature
     android:name="android.hardwre.camera"/>
    <uses-feature
     android:name="android.hardwre.camera.autofocus"/>
```

（代码详见教材,要在真实手机上运行）

4.6动画技术

4.6.1 动画组件类

1. 动画组件Animations概述

Animations是一个实现android UI界面动画效果的API，Animations提供了一系列的动画效果，可以进行旋转、缩放、淡入淡出等，这些效果可以应用在绝大多数的控件中。

2. 动画组件Animations的分类

（1）补间动画（Tweened Animations）

补间动画Tweened Animations就是只需指定开始、结束的“关键帧“，而变化中的其他帧由系统来计算，不必一帧一帧的去定义。

（2）逐帧动画（Frame Animations）

逐帧动画Frame Animations可以创建一个Drawable序列，这些Drawable可以按照指定的时间间歇一个一个的显示。

（3）属性动画（Property Animation）

属性动画Property Animation是在Android 3.0版本以后才引进的，它可以直接更改对象的属性。

4.6.2补间动画Tweened Animations

1. 补间动画效果的种类及对应着的子类

   补间动画Tweened Animations共有4种动画效果及对应着的子类：

   （1）Alpha：淡入淡出效果，其对应子类为AlphaAnimation；

   （2）Scale：缩放效果，其对应子类为ScaleAnimation；

   （3）Rotate：旋转效果，其对应子类为RotateAnimation

   （4）Translate：移动效果，其对应子类为TranslateAnimation

表 4-10　　动画效果对应子类的构造方法

对应子类的构造方法	参数说明
AlphaAnimation（ float fromAlpha, float toAlpha ）	参数 1 fromAlpha：起始透明度 参数 2 toAlpha：　　终止透明度 （取值 0.0～1.0 的数值，1.0 为完全不透明，0.0 为完全透明）
ScaleAnimation(float fromX, float toX, float fromY, float toY, int pivotXType, float pivotXValue, int pivotYType, float pivotYValue ）	参数 1 fromX：x 轴的初始值 参数 2 toX：x 轴收缩后的值 参数 3 fromY：y 轴的初始值 参数 4 toY：y 轴收缩后的值 参数 5 pivotXType：确定 x 轴坐标的类型 参数 6 pivotXValue：x 轴的值，0.5f 表明是以自身这个控件的一半长度为 x 轴 参数 7 pivotYType：确定 y 轴坐标的类型 参数 8 pivotYValue：y 轴的值，0.5f 表明是以自身这个控件的一半长度为 x 轴

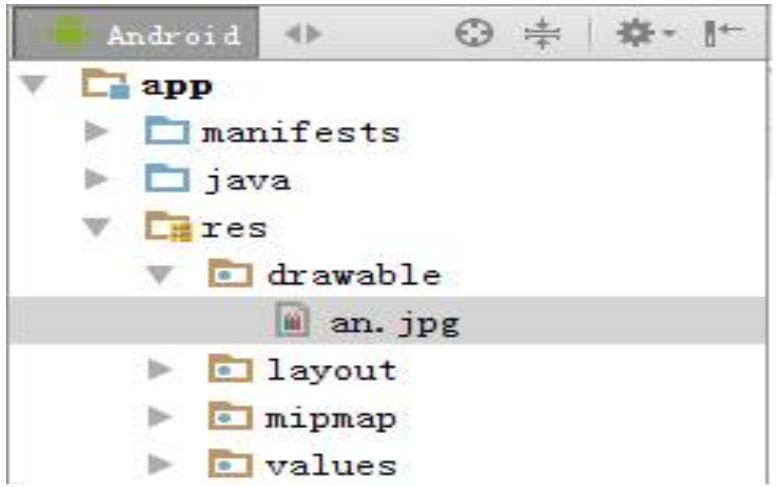
RotateAnimation(float fromDegrees, float toDegrees, int pivotXType, float pivotXValue, int pivotYType, float pivotYValue)	参数1 fromDegrees：从哪个旋转角度开始 参数2 toDegrees：转到什么角度 后面4个参数用于设置围绕着旋转的圆的圆心位置 参数3 pivotXType：确定x轴坐标的类型，有ABSOLUT绝对坐标、RELATIVE_TO_SELF相对于自身坐标、RELATIVE_TO_PARENT相对于父控件的坐标 参数4 pivotXValue：x轴的值，0.5f表明是以自身这个控件的一半长度为x轴 参数5 pivotYType：确定y轴坐标的类型 参数6 pivotYValue：y轴的值，0.5f表明是以自身这个控件的一半长度为x轴
TranslateAnimation(float fromXDelta, float toXDelta, float fromYDelta, float toYDelta)	参数1 fromXDelta：x轴的开始位置 参数2 toXDelta：x轴的结束位置 参数3 fromYDelta：y轴的开始位置 参数4 toYDelta：x轴的结束位置

2. AnimationSet类

AnimationSet是Animation的子类，用于设置Animation的属性

【例4-12】编写一个可以旋转、缩放、淡入淡出、移动的补间动画程序。

（1）新建工程ex4_12，并将事先准备的图片an.jpg复制到项目的res\drawable目录下，如图4.16所示。



(2) 编写界面布局xml程序

在界面布局xml程序中，按垂直线性布局，并放置4个按钮组件Button和一个图像显示组件ImageView。

(3) 编写控制层java程序

4.6.3 属性动画Property Animation

1. 属性动画的核心类

属性动画，就是通过控制对象中的属性值产生的动画。属性动画主要的核心类有ValueAnimator和ObjectAnimator。

（1）ValueAnimator类

ValueAnimator是整个属性动画机制当中最核心的一个类。属性动画的运行机制是通过不断地对值进行操作来实现的，而初始值和结束值之间的动画过渡就是由ValueAnimator这个类来负责计算的。它的内部使用一种时间循环的机制来计算值与值之间的动画过渡，我们只需要将初始值和结束值提供给ValueAnimator，并且告诉它动画所需运行的时长，那么ValueAnimator就会自动帮我们完成从初始值平滑地过渡到结束值这样的效果。除此之外，ValueAnimator还负责管理动画的播放次数、播放模式、以及对动画设置监听器等。

(2) ObjectAnimator 类

ObjectAnimator 是 ValueAnimator 的子类，他本身就已经包含了时间引擎和值计算，所以它拥有为对象的某个属性设置动画的功能。这使得为任何对象设置动画更加的容易。

ObjectAnimator 类是设计动画中最常使用的类，因为 ValueAnimator 只不过是对值进行了一个平滑的动画过渡，但实际使用到这种功能的场景好像并不多。而 ObjectAnimator 则就不同了，它是可以直接对任意对象的任意属性进行动画操作的，比如说 View 的 alpha 属性。

构造 ObjectAnimator 对象的方法为 ofFloat()，其方法原型为：

```
public static ObjectAnimator ofFloat(
        Object target,
        String propertyName,
        float... values
        );
```

- 第一个参数用于指定动画对象要操作的控件；
- 第二个参数用于指定动画对象所要操作控件的属性；
- 第三个参数是可变长参数，设置动画的起点和终点位置。

2. ObjectAnimator类实现动画示例

【例4-13】编写一个可以旋转、缩放、淡入淡出的属性动画程序。

（1）新建工程ex4_13，并将事先准备的图片an.jpg复制到项目的res\drawable目录下。

（2）编写界面布局xml程序

在界面布局xml程序中，按垂直线性布局，并放置按钮组件Button和图像显示组件ImageView。

(3) 编写控制层java程序

习题四

1、设计一个可以移动的小球，当小球被拖到一个小矩形块中时，则退出程序。

2、设计一个手绘图形的画板。

3、建立一个手写字体识别的字体库。

4、设计一个具有选歌功能的音频播放器。

5、为例6-7的视频播放器添加停止播放的功能。

6、编写一个具有飞入文字功能的程序。

Q&A

谢谢大家！