第 8 次课　　第 4 章　图形与多媒体处理

授课班级：　计算机 21-1

教学方式：　PPT 教学+ 课堂实例教学 + 学生课堂练习 同步方式

教学要点：第一节　绘制几何图形、第二节　触摸屏事件的处理、实验 Android 几何绘图

教学过程：　需注意同学们实践中对环境的理解与掌握，带领学生完成实践过程非常重要，特别需要注意，对长期未从事开发工作的同学的培养，去除其恐惧心理。

高级教学内容：

1）几何绘图，重点是学会 Canvas 对象的应用

2）自定义组件（本书没有专门章节，仅以一个例子来说明）

课外学习内容：

1）涂鸦程序的设计

2）俄罗斯方块程序设计

【例 4-1】绘制几何图形示例。

代码见 PPT

【例 4-2】绘制一个可以在任意指定位置显示的小球。

代码见 PPT

下面这个例子非常重要，基本应该属于高级技巧，“自定义组件”

先看例子

【例 4-3】自定义一个组件，再通过布局界面显示出来。

1. 新建 java 文件，编写 View 子类：TestView.java

```
package com.example.ex4_3;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.View;
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。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.ex4_3.MainActivity">
    <com.example.ex4_3.TestView
        android:id="@+id/testview1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
     />
</RelativeLayout>
```

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);
    }
}
```

下面例子是应用例 3 的进一步使用

【例 4-4】设计一个在屏幕上移动小球的程序。

（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 getXY(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.getXY(x1, y1);
34               setContentView(testView);
35           }
36         if (event.getAction() == MotionEvent.ACTION_DOWN)
37           {
38              x1 = (int) event.getX();
39              y1 = (int) event.getY();
40              testView.getXY(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/android"
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_margin"
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
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 =findViewById(R.id.testview1);
10                  setContentView(R.layout.activity_main);
11          }
12  }
```

下面这个例子是课后练习题：内容比较多，有些旧了，看谁能自己做出来。

【例 4-5】设计一个能在图片上涂鸦的程序。

（1）布局文件

```
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        <com.ex06_04.HandWrite
7              android:id="@+id/handwriteview"
8              android:layout_width="fill_parent"
9              android:layout_height="380dp" />
10       <LinearLayout
11           android:layout_width="fill_parent"
12           android:layout_height="fill_parent"
13           android:orientation="horizontal"
14           android:gravity="center_horizontal" >
15           <Button
16                 android:id="@+id/clear"
17                 android:layout_width="200dp"
18                 android:layout_height="wrap_content"
18                 android:text="清屏" />
20     </LinearLayout>
21 </LinearLayout>
```

导入自定义 view，注意要带包

（2）主控文件 MainActivity.java

```
1   package com.ex4_5;
2   import android.app.Activity;
3   import android.os.Bundle;
4   import android.view.View;
5   import android.view.View.OnClickListener;
6   import android.widget.Button;
7   public class MainActivity extends Activity
8   {
9      private HandWrite handWrite = null;
10     private Button clear = null;
11     @Override
12     public void onCreate(Bundle savedInstanceState)
13     {
14        super.onCreate(savedInstanceState);
15        setContentView(R.layout.main);
16        handWrite = (HandWrite)findViewById(R.id.handwriteview);
17        clear = (Button)findViewById(R.id.clear);
18        clear.setOnClickListener(new mClick());
19     }
20     private class mClick implements OnClickListener
21     {
22        public void onClick(View v)
```

关联 View 组件

```
23          {
24              handWrite.clear();
25          }
26      }
27 }
```

清屏

(3) 记录在屏幕上滑动的轨迹，实现在图片上涂鸦的功能

```
1   package com.ex4_5;
2   import android.content.Context;
3   import android.graphics.*;
4   import android.graphics.Paint.Style;
5   import android.util.AttributeSet;
6   import android.view.MotionEvent;
7   import android.view.View;
8   public class HandWrite extends View
9   {
10      Paint paint = null;                    //定义画笔
11      Bitmap originalBitmap = null;     //存放原始图像
12      Bitmap new1_Bitmap = null;          //存放从原始图像复制的位图图像
13      Bitmap new2_Bitmap = null;          //存放处理后的图像
14      float startX = 0,startY = 0;       //画线的起点坐标
15      float clickX = 0,clickY = 0;       //画线的终点坐标
16      boolean isMove = true;                //设置是否画线的标记
17      boolean isClear = false;              //设置是否清除涂鸦的标记
18      int color = Color.GREEN;              //设置画笔的颜色（绿色）
19      float strokeWidth = 2.0f;             //设置画笔的宽度
20      public HandWrite(Context context, AttributeSet attrs)
21      {
22          super(context, attrs);
23          originalBitmap = BitmapFactory
24                  .decodeResource(getResources(), R.drawable.cy);
25          new1_Bitmap = Bitmap.createBitmap(originalBitmap);
26      }
27      public void clear(){
28          isClear = true;
29          new2_Bitmap = Bitmap.createBitmap(originalBitmap);
30          invalidate();
31      }
32      public void setstyle(float strokeWidth){
33          this.strokeWidth = strokeWidth;
34      }
35      @Override
36      protected void onDraw(Canvas canvas)
```

自定义 View 组件 HandWrite

从资源中获取原始图像

建立原始图像的位图

清除涂鸦

```
37      {
38          super.onDraw(canvas);                                          ← 显示绘图
39          canvas.drawBitmap(HandWriting(new1_Bitmap), 0, 0,null);
40      }
41      public Bitmap HandWriting(Bitmap o_Bitmap)                         ← 记录绘制图形
42      {
43          Canvas canvas = null;                                          ← 定义画布
44          if(isClear)
45           {
46                canvas = new Canvas(new2_Bitmap);                        ← 创建绘制新图形的画布
47           }
48          else{
49                canvas = new Canvas(o_Bitmap);                           ← 创建绘制原图形的画布
50           }
51          paint = new Paint();
52          paint.setStyle(Style.STROKE);
53          paint.setAntiAlias(true);                                      ← 定义画笔
54          paint.setColor(color);
55          paint.setStrokeWidth(strokeWidth);
56      if(isMove)
57      {
58         canvas.drawLine(startX, startY, clickX, clickY, paint);         ← 在画布上画线条
59       }
60       startX = clickX;
61       startY = clickY;
62       if(isClear)
63       {
64            return new2_Bitmap;                                          ← 返回新绘制的图像
65        }
66       return o_Bitmap;                                                 ← 若清屏，则返回原图像
67    }
68    @Override
69    public boolean onTouchEvent(MotionEvent event)                       ← 定义触摸屏事件
70    {
71       clickX = event.getX();                                           ← 获取触摸坐标位置
72       clickY = event.getY();
73       if(event.getAction() == MotionEvent.ACTION_DOWN)                 ← 按下屏幕时无绘图
74       {
75            isMove = false;
76            invalidate();
77            return true;
78       }
79       else if(event.getAction() == MotionEvent.ACTION_MOVE)
80       {                                                                ← 记录在屏幕上划动的轨迹
```

```
81                  isMove = true;
82                  invalidate();
83                  return true;
84              }
85              return super.onTouchEvent(event);
86      }
87  }
```