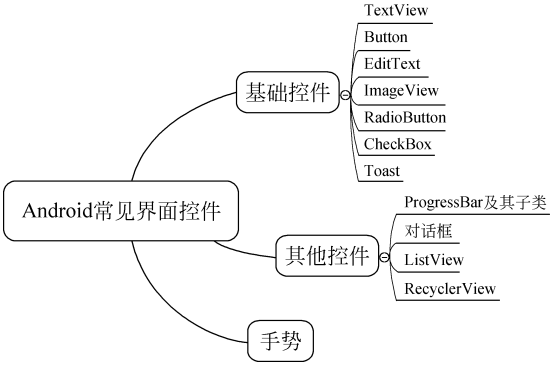
第3章 Android常见界面控件

本章导图



主要内容

- 基础控件的使用。
- ProgressBar 及其子类。
- 对话框的使用。
- ListView 的使用。
- RecyclerView 的使用。
- 手势的使用。

难点

- ListView 的使用。
- 数据适配器的使用方法。
- RecyclerView 的使用。

Android 系统提供了丰富的 UI 组件用于程序设计。在 Android Studio 中一般可以通过拖曳的方式对组件进行布局。各种组件都有一系列的属性和方法，通过这些属性和方法可以方便地操纵组件。对于具有事件触发的组件而言，开发人员可以设置事件监听器进行响应。本章将针对 Android 常见的界面控件进行讲解。

3.1 基础控件的使用

用户界面是系统与用户之间进行信息交互的接口，Android 借用了 Java 中的界面设计思想及事件响应机制。Android 系统为程序员提供了丰富的用户界面组件，包括菜单、对话框、按钮、文本框、下拉列表等。Android 支持控件拖放、XML 源码设计和程序代码操作 3 种设计形式。

观看视频

3.1.1 TextView

TextView 控件用于显示文本信息。其常用方法如表 3.1 所示。

表 3.1 TextView 控件常用方法

方　法	说　明
getText()	用于获取控件中显示的文本
setText(text)	将 text 设置为控件中要显示的文本
setTextColor()	设置文本颜色
setTextSize()	设置文本字体大小

1. 创建程序

创建一个名为 TextViewAPP3_1 的应用程序，指定包名为 com.example.TextViewAPP3_1。

2. 放置界面控件

在 res/layout 文件夹的 activity_main.xml 文件中放置一个 TextView 控件，用于显示文本信息。activitymain.xml 文件的具体代码如【文件 3_1】所示。

【文件 3_1】

```
<?xml version = "1.0" encoding = "utf - 8"?>
< RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
      xmlns:app = "http://schemas.android.com/apk/res - auto"
      xmlns:tools = "http://schemas.android.com/tools"
      android:layout_width = "match_parent"
      android:layout_height = "match_parent"
      tools:context = ".MainActivity">

      < TextView
            android:id = "@ + id/textView"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_centerInParent = "true"
            android:text = "Hello everyone!"
            android:textSize = "34sp" />
</RelativeLayout >
```

观看视频

3.1.2 Button

Button(按钮)控件主要用于响应用户点击并引发点击事件。Button 控件表示按钮，它继承自 TextView 控件，既可以显示文本，又可以显示图片，同时也允许用户通过点击来执

行操作。当 Button 控件被点击时，被按下与弹起的背景会有一个动态的切换效果，这个效果就是点击效果。

通常情况下，所有控件都可以设置点击事件，Button 控件也不例外，Button 控件最重要的作用就是响应用户的一系列点击事件。

为 Button 控件设置点击事件的方式主要有以下三种。

(1) 在布局文件中指定 onClick 属性。可以在布局文件中指定 onClick 属性的值来设置 Button 控件的点击事件，示例代码如下。

```
<Button
      android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center_horizontal"
      android:onClick="click"
      android:text="方式一" />
```

上述代码中，Button 控件指定了 onClick 属性，可以在 Activity 中定义专门的方法来实现 Button 控件的点击事件。需要注意的是，在 Activity 中定义实现点击事件的方法名必须与 onClick 属性的值保持一致。

(2) 使用匿名内部类。在 Activity 中，可以使用匿名内部类为 Button 控件设置点击事件，示例代码如下。

```
button2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        button2.setText("您选择了方式二点击按钮");
    }
});
```

上述代码中，通过 Button 控件设置 setOnClickListener()方法实现对 Button 控件点击事件的监听。SetOnClickListener()方法中传递的参数是一个匿名内部类。如果监听到按钮被点击，那么就会调用匿名内部类中的 onClick()方法实现对 Button 控件的点击事件。

(3) 用 Activity 实现 OnClickListener 接口。用当前 Activity 实现 View.OnClickListener 接口，同样可以为 Button 控件设置点击事件，示例代码如下。

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener{

    private Button button3;

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

    button3 = findViewById(R.id.button3);
    button3.setOnClickListener(this);
}
```

```
    @Override
        public void onClick(View v){
            button3.setText("您选择了方式三点击按钮");
        }
}
```

按钮被点击之前如图 3.1 所示，三个按钮都被点击之后的运行结果如图 3.2 所示。

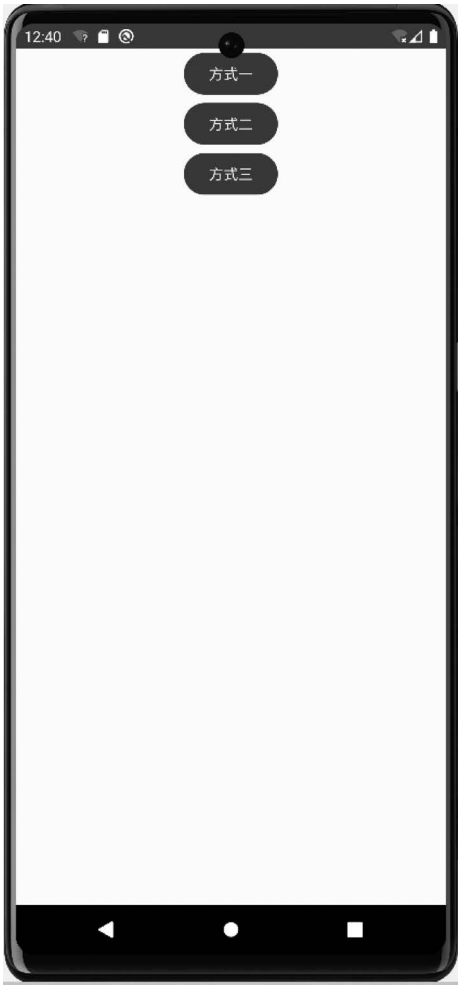


图 3.1　运行前

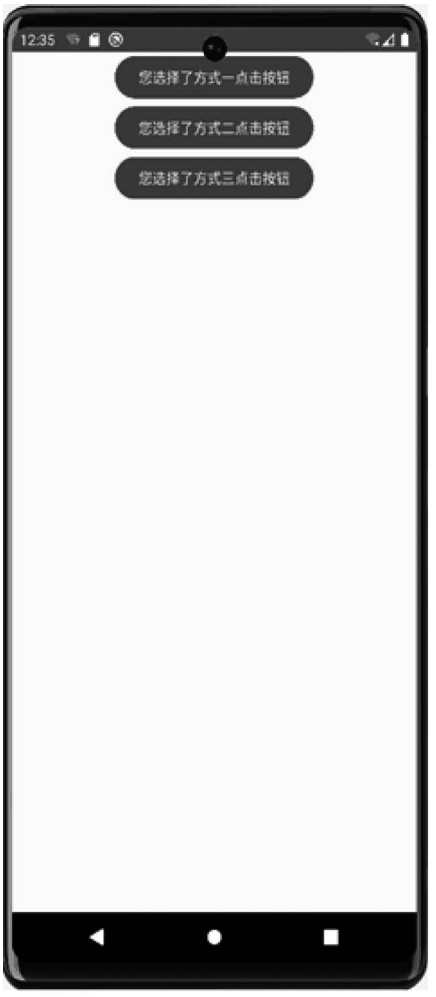


图 3.2　运行结果

3.1.3　EditText

EditText(编辑框)控件可用于输入、显示、编辑字符串。其常用方法如表 3.2 所示。

表 3.2　EditText 控件常用方法

方　法	说　明
getText()	用于获取控件中显示的文本
setText(text)	将 text 设置为控件中要显示的文本

续表

方　法	说　明
setTextColor()	设置文本颜色
setHintTextColor()	设置提示信息文本颜色

以下是编辑框的 XML 标签定义示例：

```
<EditText
      android:id="@+id/editTextText"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@id/textView"
      android:layout_centerHorizontal="true"
      android:ems="10"
      android:inputType="text"
      android:text="Name" />
```

3.1.4 ImageView

ImageView(图像视图)控件用于显示图片信息。其常用方法如表 3.3 所示。

表 3.3 ImageView 控件常用方法

方　法	说　明
setImageURI(Uri uri)	设置 ImageView 所显示内容为指定 Uri
setMaxHeight(int h)	设置控件最大高度
setMaxWeight(int w)	设置控件最大宽度

以下是 ImageView 控件的 XML 标签定义示例：

```
<ImageView
      android:id="@+id/imageView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_above="@id/textView"
      android:layout_centerHorizontal="true"
      tools:srcCompat="@tools:sample/avatars" />
```

3.1.5 RadioButton


观看视频

RadioButton(单选按钮)控件一般以按钮组的形式存在，只能在给定的系列选项组中选中一项，在设计时使用 RadioGroup(单选按钮组)控件对其进行分组。用户选中某个选项时，控件也将产生点击事件。如果单选按钮控件以按钮组的形式存在，单选按钮组控件将产生 onUncheckedChangeListener 事件。

以下是 RadioButton 和 RadioGroup 控件的 XML 标签定义示例：

```
<RadioGroup
     android:id="@+id/radiogroup"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
```

```
        android:orientation = "vertical">
        < RadioButton
              android:id = "@ + id/rbutton1"
              android:layout_width = "wrap_content"
              android:layout_height = "wrap_content"
              android:textSize = "30dp"
              android:text = "对"/>
        < RadioButton
              android:id = "@ + id/rbutton2"
              android:layout_width = "wrap_content"
              android:layout_height = "wrap_content"
              android:textSize = "30dp"
              android:text = "错"/>
    </RadioGroup >
```

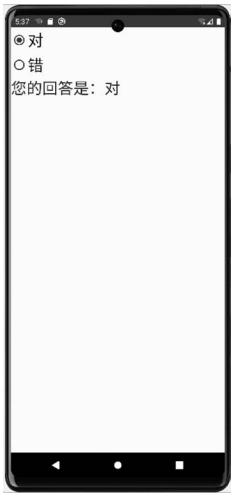


图 3.3　按钮组中的按钮被选中后的结果

RadioGroup 控件常常通过 OnCheckedChangeListener() 方法来响应按钮组中的某个按钮被选中，使用方法如下所示，图 3.3 是按钮被选中后的运行结果。

```
radioGroup.setOnCheckedChangeListener(new RadioGroup.
OnCheckedChangeListener() {
            int checkedId) {
          //判断点击的是哪个 RadioButton
          if (checkedId == R.id.rbutton1) {
              textView.setText("您的回答是:对");
          } else {
              textView.setText("您的回答是:错");
          }
      }
});
```

3.1.6　CheckBox

CheckBox(复选框)控件是可以在给定的一系列选项中选中多项的控件。其常用方法如表 3.4 所示。

表 3.4　CheckBox 控件常用方法

方　　法	说　　明
isChecked()	判断复选框是否被选中
setChecked()	设置复选框状态
setOnClickListener()	设置复选框的点击事件监听器
setOnCheckedChangeListener()	设置复选框状态改变监听器
toggle()	改变复选框当前选中的状态

CheckBox 的 XML 标签定义示例如下所示：

```
< CheckBox
      android:id = "@ + id/like_red"
      android:layout_width = "wrap_content"
      android:layout_height = "wrap_content"
```

```
        android:text = "红"
        android:textSize = "18sp"/>
    < CheckBox
        android:id = "@ + id/like_yellow"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "黄"
        android:textSize = "18sp"/>
```

CheckBox 控件的复选事件是由 OnCheckedChangeListener()方法来监听并做出响应的,运行结果如图 3.4 所示。

3.1.7 Toast


观看视频

Toast 比较适合向用户显示系统运行中的状态消息,这类消息的重要性级别一般比较低,不太需要用户过多关注,如通知用户下载已完成。消息的显示过程中,它不会将焦点从 Activity 上移开,经过一段时间后,Toast 提示框会自动消失。由于无法保证用户会完全注意 Toast 消息提示,因此关键信息不能使用 Toast 消息提示。

用户可以调用 Toast 类的方法进行 Toast 消息提示,以下是调用 makeText()方法来显示短消息的,运行结果如图 3.5 所示。

图 3.4 CheckBox 控件运行结果

图 3.5 Toast 控件运行结果

```
    Toast.makeText(MainActivity.this," 电量不足 20 %,请充电",
Toast.LENGTH_SHORT).show();
```

3.2 ProgressBar 及其子类

ProgressBar(进度条)控件用于指示操作进度的用户界面元素,它以非中断方式向用户显示进度条。一般在应用的用户界面或通知中显示进度条,而不是在对话框中显示。

3.2.1 ProgressBar 的功能和用法

ProgressBar 支持两种表示进度的模式：不确定和确定。当不确定操作需要多长时间时,使用进度条的不确定模式。不确定模式是进度条的默认模式,并显示没有指定特定进度的循环动画。

如果要显示总量和已完成量确定的进度,则使用进度条的确定模式。例如,正在检索的文件的剩余百分比,写入数据库的批处理中的记录数量,或正在播放的音频文件的剩余百分比。其常用方法如表 3.5 所示。

表 3.5 ProgressBar 控件常用方法

方　法	说　明
setProgress(int value)	更新当前的进度
setMax(int max)	设置最大进度值
setIndeterminate(boolean indeterminate)	设置进度的模式：true 为不确定模式；false 为确定模式

以下演示了一个通过按下按钮触发进度条开始更新进度的例子,为了模拟进度条滚动的效果,启动了新线程,并在其中实现了进度的更新。以下是布局文件。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
      xmlns:app = "http://schemas.android.com/apk/res - auto"
      xmlns:tools = "http://schemas.android.com/tools"
      android:layout_width = "match_parent"
      android:layout_height = "match_parent"
      android:orientation = "vertical"
      tools:context = ".MainActivity" >

      < Button
            android:id = "@ + id/button"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_gravity = "center"
            android:onClick = "startProgram"
            android:text = "开始" />

      < ProgressBar
            android:id = "@ + id/progressBar"
            style = "?android:attr/progressBarStyleHorizontal"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:progress = "5"/>
</LinearLayout >
```

以下是实现的代码。

```
public class MainActivity extends AppCompatActivity {

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

    public void startProgram(View view){
        final ProgressBar progressBar = (ProgressBar)findViewById(R.id.progressBar);
        final Thread t = new Thread(){
            @Override
            public void run() {
                super.run();
                int i = 0;
                try {
                    while(i<=100){
                        progressBar.setProgress(i);
                        i+=5;
                        sleep(1000);
                    }
                }catch (Exception e){
                    Log.e("ProgressBar", e.toString());
                }
            }
};
            t.start();
        }
}
```

图 3.6 展示了以上例子的运行结果。

3.2.2 SeekBar 的功能和用法

SeekBar(拖动条)和进度条非常相似,只是进度条采用颜色填充来表明进度完成的程度,而拖动条则通过滑块的位置来标识数值——而且拖动条允许用户拖动滑块来改变值,因此拖动条通常用于对系统的某种数值进行调节,如调节音量等。

由于 SeekBar 继承了 ProgressBar,因此 ProgressBar 所支持的 XML 属性和方法完全适用于 SeekBar。

SeekBar 允许用户改变拖动条的滑块外观,改变滑块外观通过如下属性来指定。

- android:thumb:指定一个 Drawable 对象,该对象将作为自定义滑块。
- android:tickMark:指定一个 Drawable 对象,该对象将作为自定义刻度图标。

为了让程序能响应拖动条滑块位置的改变,可以考虑为它绑定一个 OnSeekBarChangeListener 监听器。

SeekBar 控件的 XML 标签定义示例如下所示:

图 3.6　ProgressBar 控件运行结果

```
<SeekBar
        android:id="@+id/seekBar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:max="100"
        android:progress="1"
        android:thumb="@mipmap/ic_launcher" />
```

图 3.7 显示了 SeekBar 控件的运行结果。

3.2.3　RatingBar 的功能和用法

RatingBar(星级评分条)与 SeekBar 有相同的父类：AbsSeekBar，因此它们十分相似。它们都允许用户通过拖动来改变进度，最大的区别是 RatingBar 通过星星来表示进度。以下是 RatingBar 控件的 XML 标签定义示例，图 3.8 展示了 Ratingbar 控件的运行结果。

图 3.7 SeekBar 控件运行结果

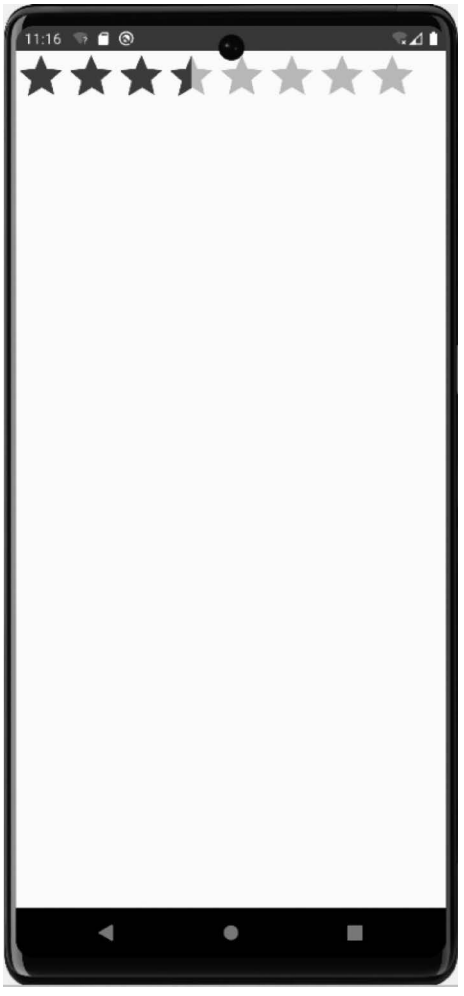


图 3.8 RatingBar 控件运行结果

```
<RatingBar
    android:id="@+id/ratingBar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:numStars="8"
    android:stepSize="0.5"
    android:progress="1"
    android:max="8"/>
```

3.3 对话框的使用

在 Android 应用程序中，AlertDialog 对话框用于提示一些重要信息或者显示一些需要用户额外交互的内容。它一般以小窗口的形式展示在界面上。

3.3.1 使用 AlertDialog 建立对话框

使用 AlertDialog 创建的对话框一般包含标题、内容和按钮三个区域。一般情况下，创建 AlertDialog 对话框的步骤大致分为以下几步。

（1）调用 AlertDialog 静态内部类 Builder，创建 AlertDialog. Builder 对象。

（2）调用 AlertDialog. Builder 对象的 setTitle()和 setIcon()方法分别设置 AlertDialog 对话框的标题名称和图标。

（3）调用 AlertDialog. Builder 对象的 setMessage()、setSingleChoiceItems()方法或 setMultiChoiceItems()方法设置 AlertDialog 对话框的内容为简答文本、单选列表或者多选列表。

（4）调用 AlertDialog. Builder 对象的 setPositiveButton()和 setNegativeButton()方法设置 AlertDialog 对话框的“确定”和“取消”按钮。

（5）调用 AlertDialog. Builder 对象的 create()方法创建 AlertDialog 对象。

（6）调用 AlertDialog 对象的 show()方法显示该对话框。

（7）调用 AlertDialog 对象的 dismiss()方法取消该对话框。

以下代码通过点击手机上的后退键弹出一个对话框，运行结果如图 3.9 所示。

```
@Override
public void onBackPressed() {
      //声明对象
     AlertDialog dialog;
     AlertDialog.Builder builder = new AlertDialog.Builder(this)
               .setTitle("普通对话框")                    //设置对话框的标题
               .setIcon(R.mipmap.ic_launcher)            //设置标题图标
               .setMessage("是否确定退出应用?")           //设置对话框的提示信息
               //添加"确定"按钮
               .setPositiveButton("确定", new DialogInterface.OnClickListener() {
                   @Override
                   public void onClick(DialogInterface dialog, int which) {
                       dialog.dismiss();                 //关闭对话框
                       MainActivity.this.finish();       //关闭 MainActivity
                   }
               })
               //添加"取消"按钮
               .setNegativeButton("取消", new DialogInterface.OnClickListener() {
                   @Override
                   public void onClick(DialogInterface dialog, int which) {
                       dialog.dismiss();
                   }
               });
        dialog = builder.create();
        dialog.show();
}
```

3.3.2 创建单选和多选对话框

单选对话框的内容区域显示为单选列表。只需要调用 AlertDialog. Builder 对象的

图 3.9 普通对话框运行结果

setSingleChoiceItems()方法即可创建带单选列表的对话框。示例代码如下所示。

```
AlertDialog.Builder builder = new AlertDialog.Builder(this)
        .setTitle("设置字体大小")                    //设置标题
        .setIcon(R.mipmap.ic_launcher)
        .setSingleChoiceItems(new String[]{"小号", "默认", "中号", "大号",
                "超大"}, textSize,new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                textSize = which;
            }
        })
        .setPositiveButton("确定", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                //为 TextView 设置在单选对话框中选择的字体大小
                textView.setTextSize(textSizeArr[textSize]);
                dialog.dismiss();                    //关闭对话框
            }
```

```
            })//添加"确定"按钮
            .setNegativeButton("取消", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.dismiss();
                }
            });
dialog = builder.create();
dialog.show();
```

图 3.10 是单选对话框示例的运行结果。

图 3.10　单选对话框示例的运行结果

多选对话框的内容区域显示为多选列表。

以下是多选对话框的示例代码，图 3.11 是多选对话框示例的运行结果。

```
AlertDialog.Builder builder = new AlertDialog.Builder(this)
        .setTitle("请添加兴趣爱好!")
        .setIcon(R.mipmap.ic_launcher)
        .setMultiChoiceItems(items, checkedItems,
                new DialogInterface.OnMultiChoiceClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which, boolean
                            isChecked){
                        checkedItems[which] = isChecked;
```

```
                    }
                })
        .setPositiveButton("确定", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                StringBuffer stringBuffer = new StringBuffer();
                for (int i = 0; i <= checkedItems.length - 1; i++) {
                    if (checkedItems[i]) {
                        stringBuffer.append(items[i]).append(" ");
                    }
                }
                if (stringBuffer != null) {
                    Toast.makeText(MainActivity.this, "" + stringBuffer,
                            Toast.LENGTH_SHORT).show();
                }
                dialog.dismiss();
            }
        })
        .setNegativeButton("取消", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        });
dialog = builder.create();
dialog.show();
```

图 3.11 多选对话框示例的运行结果

3.3.3 创建 DatePickerDialog 和 TimePickerDialog 对话框

DatePickerDialog 与 TimePickerDialog 的用法相似，只需要如下两步：

（1）通过构造器创建 DatePickerDialog、TimePickerDialog 实例，调用它们的 show()方法即可将日期选择对话框、时间选择对话框显示出来。

（2）为 DatePickerDialog、TimePickerDialog 绑定监听器，这样可以保证用户通过 DatePickerDialog、TimePickerDialog 选择日期、时间触发监听器，从而通过监听器来获取用户所选择的日期、时间。

3.4 ListView 的使用

ListView 组件主要用于以列表的形式显示数据并响应用户的选择点击事件。

创建 ListView 组件一般需要以下三个元素。

（1）ListView 中每一行的 View。View 可以是系统存在的布局 XML 文件，也可以是自定义的 XML 布局文件。

（2）需要展示的数据。数据既可以来自数组资源，又可以在程序中生成。

（3）连接数据与 ListView 的适配器。

3.4.1 ListView 控件的简单使用

ListView 控件常用方法如表 3.6 所示。

表 3.6 ListView 控件常用方法

方　法	说　明
setAdapter()	为视图设置数据适配器
getAdapter()	获得当前视图适配器
setDivider()	设置元素间的分隔符
setDividerHeight()	设置分隔符的高度
setSelection(int position)	设置视图中的选中项
getMaxScrollAmount()	获得视图的最大滚动数量
setOnItemClickListener(AdapterView. OnItemClickListener listener)	设置 ListView 数据项点击监听器
setOnItemSelectedListener(AdapterView. OnItemSelectedListener listener)	设置数据项选定时的监听器

在 XML 文件的 LinearLayout 中添加 ListView 控件的示例代码如下。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
      android:layout_width = "match_parent"
      android:layout_height = "match_parent"
      android:orientation = "vertical">
          < ListView
          android:id = "@ + id/lv"
          android:layout_width = "match_parent"
```

```
        android:layout_height = "wrap_content"/>
</LinearLayout >
```

3.4.2 常用数据适配器

适配器主要用于提供数据转换功能，将源数据转换为目标组件需要的数据格式。Android 可以提供多种数据源，而组件能够识别的数据格式却很单一。因此，需要使用适配器实现数据源与组件之间的数据转换。

Android 针对不同的数据源提供了多种适配器，如 ArrayAdapter、SimpleAdapter、CursorAdapter、BaseAdapter 等。其中，ArrayAdapter 最为简单，它用来绑定一个数组，支持泛型操作。SimpleAdapter 有最好的扩充性，可以自定义各种效果，例如，可以组合 ImageView、Button、CheckBox 等多种组件展示一项数据。CursorAdapter 用来绑定游标得到的数据，它可以看作 SimpleAdapter 与数据库的简单结合，可以方便地把数据库的内容以列表的形式展示出来。BaseAdapter 是一种基础数据适配器，BaseAdapter 类是一个抽象类，用户需要继承该类并实现相应的方法，从而对适配器进行更灵活的操作。

在创建适配器后，可以通过 ListView 对象的 setAdapter() 方法添加适配器，如将继承 BaseAdapter 的 MyBaseAdapter 实例添加到 ListView 中，示例代码如下。

```
//初始化 ListView 控件
mListView = (ListView) findViewById(R.id.lv);
//创建一个 Adapter 的实例
MyBaseAdapter mAdapter = new MyBaseAdapter();
//设置 Adapter
mListView.setAdapter(mAdapter);
```

3.4.3 自定义 ListItem

每个 ListView 控件的列表都是由若干 Item 组成的，例如以下示例，每个 Item 上都显示商品的图片、名称及价格，界面效果如图 3.12 所示。

图 3.12 ListItem 界面效果

3.5 RecyclerView 的使用

在 Android 5.0 之后，Google 提供了用于在有限的窗口范围内显示大量数据的控件 RecyclerView。与 ListView 控件相似，RecyclerView 控件同样是以列表的形式展示数据的，并且数据都是通过适配器加载的。但是 RecyclerView 的功能更加强大，接下来从以下几方面来分析。

(1) 展示效果：RecyclerView 控件可以通过 LayoutManager 类实现横向或竖向的列表效果、瀑布流效果和 GridView 效果，而 ListView 控件只能实现竖直的列表效果。

（2）适配器：RecyclerView 控件使用的是 RecyclerView. Adapter 适配器，该适配器将 BaseAdapter 中的 getView()方法拆分为 onCreateViewHolder()方法和 onBindViewHolder()方法，强制使用 ViewHolder 类，使代码编写规范化，避免了初学者写的代码性能不佳。

（3）复用效果：ListView 控件复用 Item 对象的工作由该控件自己实现，而 ListView 控件复用 Item 对象的工作需要开发者通过 convertView 的 setTag()方法和 getTag()方法进行操作。

（4）动画效果：RecyclerView 控件可以通过 setItemAnimator()方法为 Item 添加动画效果，而 ListView 控件不可以通过该方法为 Item 添加动画效果。

以下逻辑代码是实现 RecyclerView 控件的一个示例，运行结果如图 3.13 所示。

```
public class MainActivity extends AppCompatActivity {
    private RecyclerView mRecyclerView;
    private HomeAdapter mAdapter;
    private String[] names = { "小猫", "哈士奇", "小黄鸭","小鹿","老虎"};
    private int[] icons = { R.drawable.cat,R.drawable.siberiankusky,
            R.drawable.yellowduck,R.drawable.fawn, R.drawable.tiger};
    private String[] introduces = {
            "猫,属于猫科动物,分家猫、野猫,是全世界家庭中较为广泛的宠物。",
            "西伯利亚雪橇犬,常见别名哈士奇,昵称为二哈。",
            "鸭的体型相对较小,颈短,一些属的嘴要大些。腿位于身体后方,因而步态蹒跚。",
            "鹿科是哺乳纲偶蹄目下的一科动物。体型大小不等,为有角的反刍类。",
            "虎,大型猫科动物;毛色浅黄或棕黄色,满身黑色横纹;头圆、耳短,耳背面黑色,中
央有一白斑甚显著;四肢健壮有力;尾粗长,具黑色环纹,尾端黑色。"
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mRecyclerView = (RecyclerView) findViewById(R.id.id_recyclerview);
        mRecyclerView.setLayoutManager(new LinearLayoutManager(this));
        mAdapter = new HomeAdapter();
        mRecyclerView.setAdapter(mAdapter);
    }
    class HomeAdapter extends RecyclerView.Adapter<HomeAdapter.MyViewHolder>{
        @Override
        public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
            MyViewHolder holder = new MyViewHolder(LayoutInflater.from(MainActivity.
                    this).inflate(R.layout.recycler_item, parent, false));
            return holder;
        }
        @Override
        public void onBindViewHolder(MyViewHolder holder, int position) {
            holder.name.setText(names[position]);
            holder.iv.setImageResource(icons[position]);
            holder.introduce.setText(introduces[position]);
        }
        @Override
        public int getItemCount() {
            return names.length;
        }
```

```
        class MyViewHolder extends RecyclerView.ViewHolder {
            TextView name;
            ImageView iv;
            TextView introduce;
            public MyViewHolder(View view) {
                super(view);
                name = (TextView) view.findViewById(R.id.name);
                iv = (ImageView) view.findViewById(R.id.iv);
                introduce = (TextView) view.findViewById(R.id.introduce);
            }
        }
    }
}
```

图 3.13 RecyclerView 控件运行结果

3.6 手势

Android 开发中，几乎所有的事件都会和用户进行交互，而最多的交互形式就是手势。目前有多款手机已经支持手写输入，其原理是根据用户输入的内容在预先定义的词库中查找最佳匹配项供用户选择。

手势是指用户手指或触摸笔在触摸屏幕上连续碰撞的行为。当用户触摸屏幕时，会产生许多手势，如按下、滑动、弹起等。Android SDK 提供了一个 GestureDetector 类，通过该

类可以识别很多复杂的手势。

Android 系统对两种手势提供了支持：一种是在屏幕上从上到下绘制一条线条的简单手势，Android 提供了检测此种手势的监听器；另一种是在屏幕上绘制一个不规则的图形的复杂手势，Android 允许开发商添加此种手势，并提供了相应的 API 识别用户手势。

3.6.1 手势检测

Android 系统提供的 GestureDetector 类用于检测用户的触摸手势，该类内部定义了三个监听接口和一个类，分别是 OnGestureListener 接口、OnDoubleTapListener 接口、OnContextClickListener 接口以及 SimpleOnGestureListener 类。以下代码展示了通过在 raw 文件夹中添加手势文件进行匹配。

```
public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     //在 raw 文件夹中加载手势文件
     library = GestureLibraries.fromRawResource(this, R.raw.gestures);
     GestureOverlayView gesture = (GestureOverlayView) findViewById(R.id.
            gestures);
     gesture.addOnGesturePerformedListener(this); //增加事件监听器
}

@Override
public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
     loadStatus = library.load(); //加载手势库
     if (loadStatus) { //如果手势库加载成功
          //识别绘制的手势，Prediction 是一个相似度对象，集合中的相似度是从高到低进行排列
          ArrayList<Prediction> pres = library.recognize(gesture);
          if (!pres.isEmpty()) {
                Prediction pre = pres.get(0);          //获取相似度最高的对象
                //用整型的数表示百分比，如> 40 %
                if (pre.score > 4) { //如果手势的相似度分数大于 40 %，则匹配成功
                     Toast.makeText(this, pre.name, Toast.LENGTH_LONG).show();
                } else {
                     Toast.makeText(MainActivity.this, "手势匹配不成功",
                            Toast.LENGTH_LONG).show();
                }
          } else {
                Toast.makeText(MainActivity.this, "手势库加载失败",
                       Toast.LENGTH_LONG).show();
          }
     }
}
```

3.6.2 增加手势

Android 系统除了提供手势检测之外，还允许应用程序将用户手势添加到指定文件中，便于后续用户再次绘制该手势时，系统可识别该手势。Android 系统使用 GestureLibrary 来代替手势库，并提供了 GestureLibraries 工具类来创建手势库。

3.7 应用实例：图片浏览器

通过本章学习的内容可以实现一个图片浏览器的 Android 应用程序。此程序主要实现两个功能：其一是通过单选按钮切换界面背景图片；其二是通过“上一张”“下一张”两个按钮从图片队列里选择顺序或者逆序展示图片，或者用手指左滑、右滑页面来切换图片。由于需要用到背景图片和需要展示的若干图片，在创建项目之后，需要将所需要的图片导入 res 文件夹的 drawable 文件夹，如图 3.14 所示。

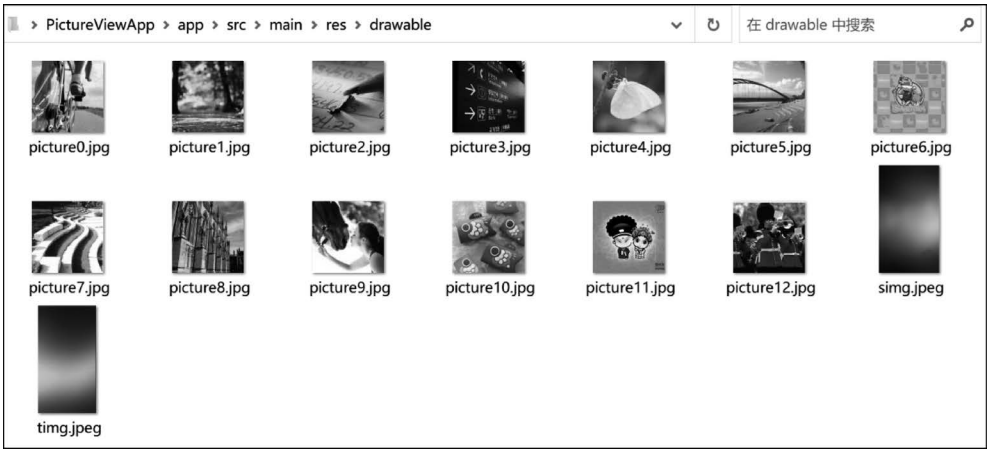


图 3.14 drawable 文件夹

该程序的 XML 界面代码如下所示。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< RelativeLayout xmlns:android = "http://schemas.android.com/apk/res/android"
      xmlns:app = "http://schemas.android.com/apk/res - auto"
      xmlns:tools = "http://schemas.android.com/tools"
      android:id = "@ + id/activity_main"
      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.bob50.pictureviewapp.MainActivity"
      android:background = "@drawable/timg">

      < ImageView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            app:srcCompat = "@drawable/picture0"
            android:layout_marginTop = "57dp"
            android:id = "@ + id/imageView"
```

```
            android:layout_alignParentTop = "true"
            android:layout_centerHorizontal = "true" />

        < Button
            android:text = "上一张"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_below = "@ + id/imageView"
            android:layout_alignParentStart = "true"
            android:layout_marginStart = "49dp"
            android:layout_marginTop = "94dp"
            android:id = "@ + id/button1" />

        < Button
            android:text = "下一张"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_alignTop = "@ + id/button1"
            android:layout_alignParentEnd = "true"
            android:layout_marginEnd = "46dp"
            android:id = "@ + id/button2" />

        < RadioGroup
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:layout_alignParentBottom = "true"
            android:layout_centerHorizontal = "true"
            android:layout_marginBottom = "56dp"
            android:id = "@ + id/radioGroup1" >

            < RadioButton
                android:text = "背景 1"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:id = "@ + id/radioButton1"
                android:layout_weight = "1" />

            < RadioButton
                android:text = "背景 2"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:id = "@ + id/radioButton2"
                android:layout_weight = "1" />

        </RadioGroup >
    </RelativeLayout >
```

该程序的 XML 界面布局如图 3.15 所示。

该程序的逻辑代码如下所示，运行结果如图 3.16 和图 3.17 所示。

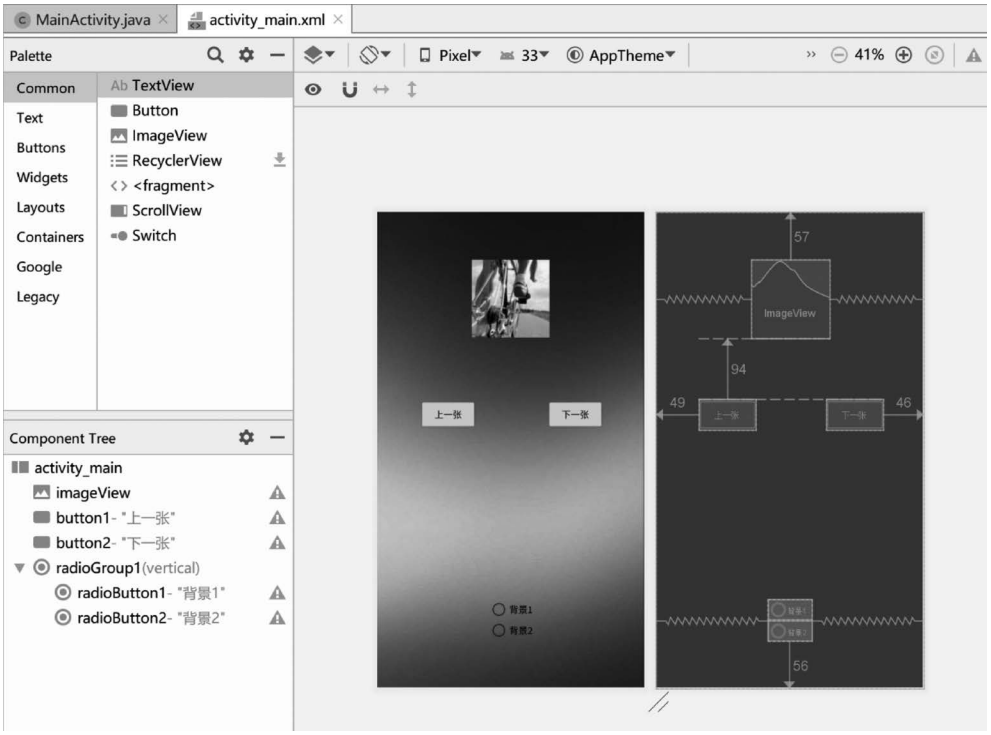


图 3.15　程序的界面布局

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener
{

    private ImageView imageView;
    private Button button1;
    private Button button2;
    private RadioGroup radioGroup1;
    private RelativeLayout relativeLayout;
    //private ArrayList imageList;
    private int[] imageArray;
    private int index = 0;

    private GestureDetector mGestureDetector;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        relativeLayout = (RelativeLayout)findViewById(R.id.activity_main);
        imageView = (ImageView) this.findViewById(R.id.imageView);
        button1 = (Button) this.findViewById(R.id.button1);
        button2 = (Button) this.findViewById(R.id.button2);
        button1.setOnClickListener(this);
        button2.setOnClickListener(this);
        radioGroup1 = (RadioGroup)findViewById(R.id.radioGroup1);
        radioGroup1.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup radioGroup, int i) {
```

```
                    if(i == R.id.radioButton1)
                    {
                        relativeLayout.setBackgroundResource(R.drawable.timg);
                    }
                    if(i == R.id.radioButton2)
                    {
                        relativeLayout.setBackgroundResource(R.drawable.simg);
                    }
                }
            });
            //imageList = new ArrayList < Integer >();
    imageArray = new int[ ]{R.drawable.picture0, R.drawable.picture0, R.drawable.picture1, R.
    drawable.picture2,
    R.drawable.picture3, R.drawable.picture4, R.drawable.picture5, R.drawable.picture6, R.
    drawable.picture7,
    R.drawable.picture8, R.drawable.picture9, R.drawable.picture10, R.drawable.picture11, R.
    drawable.picture12};

                      mGestureDetector = new GestureDetector(this, new GestureDetector.
    SimpleOnGestureListener(){
                    @Override
                     public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
    float velocityY) {
                        //向右滑动表示进入下一页
                        if((e1.getRawX() - e2.getRawX()) > 200)
                        {
                            index++;
                            index = index % 13;
                            Log.d("index:"," " + index);
                            imageView.setImageResource(imageArray[index]);
                            return true;
                        }
                        //向左滑动表示进入上一页
                        if((e2.getRawX() - e1.getRawX()) > 200)
                        {
                            index--;
                            index = (index + 13) % 13;
                            Log.d("index:"," " + index);
                            imageView.setImageResource(imageArray[index]);
                            return true;
                        }
                        return super.onFling(e1, e2, velocityX, velocityY);
                    }
                });
        }

        @Override
        public void onClick(View v) {
            switch (v.getId())
            {
                case R.id.button1:
                    index--;
                    index = (index + 13) % 13;
```

```
                    Log.d("index:"," " + index);
                    this.imageView.setImageResource(this.imageArray[index]);
                    break;
                case R.id.button2:
                    index++;
                    index = index % 13;
                    Log.d("index:"," " + index);
                    this.imageView.setImageResource(this.imageArray[index]);
                    break;
            }
        }

        @Override
        public boolean onTouchEvent(MotionEvent event) {
            return mGestureDetector.onTouchEvent(event);
        }
    }
```

图 3.16 运行结果 1

图 3.17 运行结果 2

3.8 小结

本章主要讲解了 Android 中控件的相关知识，包括简单控件、AlertDialog 对话框、ListView 控件和 RecyclerView 控件以及 ProgressBar 控件等。通过本章的学习，希望初学者能够掌握 Android 控件的基本使用，因为无论创建任何 Android 程序，都可能用到这些控件。

习题 3

1. 简述实现 Button 按钮的点击事件都有哪些方法。
2. 简述 AlertDialog 对话框的创建过程。
3. 简述 ListView 和 RecylerView 的区别。
4. 设计一款移动端计算器的界面。