第5章 chapter 5

Android 用户界面

Android 用户界面是应用程序开发的重要组成部分，决定了应用程序是否美观、易用。通过本章的学习可以让读者熟悉 Android 用户界面的基本开发方法，了解在 Android 界面开发过程中常见的界面控件、界面布局、菜单和界面事件的使用方法，充分理解手机应用程序与桌面应用程序在用户界面开发上的不同与相同之处。

本章学习目标：

- 了解各种界面控件的使用方法；
- 掌握各种界面布局的特点和使用方法；
- 掌握选项菜单、子菜单和快捷菜单的使用方法；
- 掌握操作栏和 Fragment 的使用方法；
- 掌握按键事件和触摸事件的处理方法。

5.1 用户界面基础

用户界面(User Interface，UI)是系统和用户之间进行信息交换的媒介，实现信息的内部形式与人类可以接收形式之间的转换。最古老的用户界面是各种形式的文字、图形、旗帜和手势等，这些抽象符合作为信息传递的介质，使人类可以理解这些信息所包含的意义或指代的实体。

算盘是由柱子和珠子组成的最早的人机交互界面。在计算机出现的早期，批处理界面(1945—1968 年)和命令行界面(1969—1983 年)得到广泛的使用。目前，流行的用户界面是图形用户界面(Graphical User Interface，GUI)，采用图像的方式与用户进行交互。与早期的交互界面相比，图形用户界面对于用户来说更加简便易用，用户从此不再需要记住大量的命令，取而代之的是通过窗口、菜单和按钮等方式来进行操作。未来的用户界面将更多地运用虚拟现实技术，使用户能够摆脱键盘与鼠标的交互方式，而通过动作、语言，甚至脑电波来控制计算机。当然，对用户界面的深入探讨远超出了本书的涉及范围，感兴趣的读者可以阅读相关书籍。

在手机上进行用户界面设计是一项具有挑战性的工作。首先，手机的界面设计者和程序开发者是独立且并行工作的，这就需要界面设计与程序逻辑完全分离，不仅有利于并行工作，而且在后期修改界面时也可以避免修改程序的逻辑代码；其次，不同型号手机

的屏幕解析度、尺寸和长宽比各不相同，程序界面需要能够根据屏幕信息，自动调整界面控件的位置和尺寸，避免因为屏幕解析度、尺寸或纵横比的变化而出现显示错误；最后，手机屏幕尺寸较小，设计者必须能够合理利用有限的显示空间，构造出符合人机交互规则的用户界面，避免出现凌乱、拥挤的用户界面。

Android 系统已经为使用者解决了界面设计前两个问题。在界面设计与程序逻辑分离方面，Android 系统将用户界面和资源从逻辑代码中分离出来，使用 XML 文件对用户界面进行描述，资源文件独立保存在资源文件夹中。Android 系统的用户界面描述非常灵活，允许模糊定义界面元素的位置和尺寸，通过声明界面元素的相对位置和粗略尺寸，从而使界面元素能够根据屏幕尺寸和屏幕摆放方式动态调整显示方式。

Android 用户界面框架（Android UI Framework）采用 MVC（Model-View-Controller）模型，为用户界面提供了处理用户输入的控制器（Controller）和显示图像的视图（View），模型（Model）是应用程序的核心，数据和代码被保存在模型中。控制器、视图和模型的关系如图 5.1 所示。

MVC 模型中的视图将应用程序的信息反馈给用户，可能的反馈方法包括视觉、听觉或触觉等，但最常用的就是通过屏幕显示反馈信息。Android 系统的界面元素以一种树形结构组织在一起，这种树形结构称为视图树（View Tree），如图 5.2 所示。Android 系统在屏幕上绘制界面元素时，会依据视图树的结构从上至下绘制每一个界面元素。每个元素负责对自身的绘制，如果元素包含子元素，该元素会通知其下所有子元素进行绘制。Android 系统在用户界面绘制上还有一些提高效率的办法，例如，如果父元素能够确定某个区域一定会被其子元素绘制，则父元素会停止绘制该区域，以提高屏幕绘制的效率，缩短绘制时间。

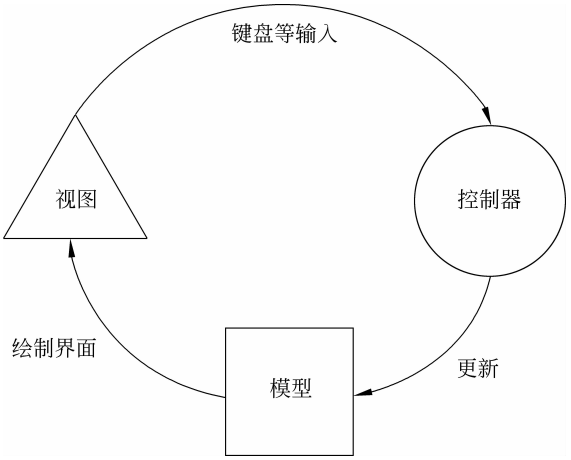

图 5.1　MVC 模型

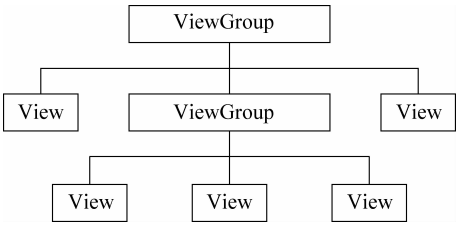

图 5.2　视图树

视图树由 View 和 ViewGroup 构成。View 是界面中最基本的可视单元，存储了屏幕上特定矩形区域内所显示内容的数据结构，并能够实现所占据区域的界面绘制、焦点变化、用户输入和界面事件处理等功能。View 也是一个重要的基类，所有在界面上的可见元素都是 View 的子类。ViewGroup 是一种能够承载多个 View 的显示单元，一般有

两个用途，一个是承载界面布局，另一个是承载具有原子特性的重构模块。

MVC 模型中的控制器能够接收并响应程序的外部动作，如按键动作或触摸屏幕动作等。控制器使用队列处理外部动作，每个外部动作作为一个独立的事件被加入队列中，然后 Android 用户界面框架按照“先进先出”的规则从队列中获取事件，并将这个事件分配给所对应的事件处理函数。

Android 用户界面框架中另一个重要的概念就是单线程用户界面(Single-threaded UI)。在单线程用户界面中，控制器从队列中获取事件，视图在屏幕上绘制用户界面，使用的都是同一个线程。单线程用户界面能够降低应用程序的复杂程度，同时也能降低开发的难度。首先，用户不需要在控制器和视图之间进行同步。其次，所有事件处理完全按照其加入队列的顺序进行，也就是说，在事件处理函数返回前不会处理其他事件，因此，用户界面的事件处理函数具有原子性。但单线程用户界面也有其缺点，如果事件处理函数过于复杂，可能会导致用户界面失去响应。因此，应尽可能在事件处理函数中使用简短的代码，或将复杂的工作交给后台线程处理。

5.2 界面控件

Android 系统的界面控件分为定制控件和系统控件。定制控件是用户独立开发的控件，或通过继承并修改系统控件后所产生的新控件，能够提供特殊的功能和显示需求。系统控件是 Android 系统中已经封装的界面控件，是应用程序开发过程中最常见的功能控件。系统控件更有利于进行快速开发，同时能够使 Android 应用程序的界面保持一定的一致性。

常见的系统控件包括 TextView、EditText、Button、ImageButton、Checkbox、RadioButton、Spinner、ListView 和 TabHost。

5.2.1 TextView 和 EditText

TextView 是一种用于显示字符的控件，EditText 则是用来输入和编辑字符的控件，因为 EditText 继承于 TextView，所以 EditText 是一个具有编辑功能的 TextView 控件。

TextViewDemo 示例如图 5.3 所示，从上至下分别是 TextView01 和 EditText01。在 XML 文件(/res/layout/main.xml)中的代码如下。

```
<TextView android:id="@+id/TextView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="TextView01" >
</TextView>
<EditText android:id="@+id/EditText01"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="EditText01" >
```

```
10 </EditText>
```

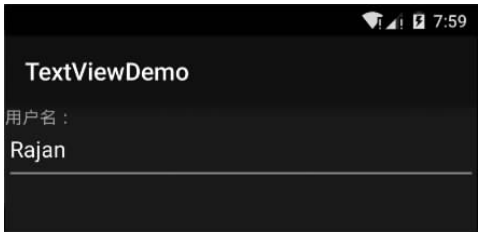

图 5.3 TextViewDemo 示例

第 1 行代码中的 android:id 属性声明了 TextView 的 ID，这个 ID 主要用于在代码中引用 TextView 对象。@＋id/TextView01 表示所设置的 ID 值，其中@表示后面的字符串是 ID 资源；加号(＋)表示需要建立新资源名称，并添加到 R.java 文件中；斜杠后面的字符串(TextView01)表示新资源的名称。如果不是新添加的资源，或属于 Android 框架的资源，则不需要使用加号，但必须添加 Android 包的命名空间，例如 android:id＝"@android:id/empty"。

第 2 行代码中的 android:layout_width 属性用来设置 TextView 的宽度，wrap_content 表示 TextView 的宽度只要能够包含所显示的字符串即可。第 3 行代码中的 android:layout_height 属性用来设置 TextView 的高度。第 4 行代码表示 TextView 所显示的字符串，在后面将通过代码更改 TextView 的显示内容。第 7 行中代码的 match_parent 表示 EditText 的宽度将等于父控件的宽度。

TextViewDemo.java 文件中，引用 XML 文件中建立的 TextView 和 EditText，并更改其显示内容。为了能够使程序正常运行，需要在代码中引入 android.widget.EditText 和 android.widget.TextView。

```
1       TextView textView=(TextView)findViewById(R.id.TextView01);
2       EditText editText=(EditText)findViewById(R.id.EditText01);
3       textView.setText("用户名:");
4       editText.setText("Rajan");
```

第 1 行的 findViewById()函数能够通过 ID 引用界面上的任何控件，只要该控件在 XML 文件中定义过 ID 即可。第 3 行的 setText()函数用来设置 TextView 所显示的内容。

5.2.2 Button 和 ImageButton

Button 是按钮控件，用户点击该控件，能够引发相应的事件处理函数。如果需要在按钮上显示图像，可以使用 Android 系统提供的 ImageButton 控件，图 5.4 是 ButtonDemo 示例，上方是 Button 控件，下方是 ImageButton 控件。

ButtonDemo 示例从上至下分别是 TextView01、Button01 和 ImageButton01。在 XML 文件(/res/layout/main.xml)中的代码如下。

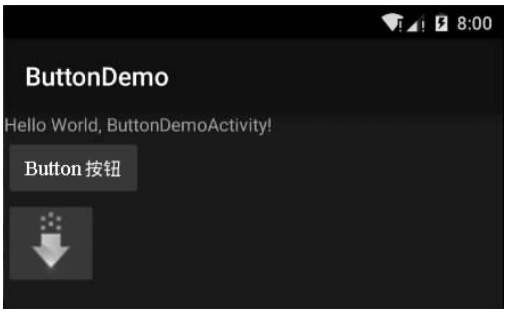

图 5.4 ButtonDemo 示例

```
1  <Button android:id="@+id/Button01"
2      android:layout_width="wrap_content"
3      android:layout_height="wrap_content"
4      android:text="Button01" >
5  </Button>
6  <ImageButton android:id="@+id/ImageButton01"
7      android:layout_width="wrap_content"
8      android:layout_height="wrap_content">
9  </ImageButton>
```

XML 文件中定义了两个按钮的宽度和高度，并定义了 Button 控件所显示的内容，但没有定义 ImageButton 所显示的图像，显示图像内容在后面的代码中进行定义。

Android 系统支持多种图形格式，如 png、ico 等，本例 ImageButton 所使用的是 png 格式。首先在/res 目录下建立 drawable 目录，然后将 download.png 文件复制到/res/drawable 文件夹下。

ButtonDemo.java 文件中，引用 XML 文件建立的 Button 控件和 ImageButton 控件，更改 Button 显示字符内容和 ImageButton 图像内容。为了程序正常运行，需要在代码中引入 android.widget .Button和 android.widget.ImageButton。

```
1  Button button=(Button)findViewById(R.id.Button01);
2  ImageButton imageButton=(ImageButton)findViewById(R.id.ImageButton01);
3  button.setText("Button 按钮");
4  imageButton.setImageResource(R.drawable.download);
```

第 1 行和第 2 行代码用于引用在 XML 文件中定义的 Button 控件和 ImageButton 控件。第 3 行代码将 Button 的显示内容更改为“Button 按钮”。第 4 行代码利用 setImageResource()函数，将新加入的 png 文件 R.drawable.download 传递给 ImageButton。

为了能够使按钮响应点击事件，在 onCreate()函数中为 Button 控件和 ImageButton 控件添加点击事件的监听器，代码如下。

```
1  final TextView textView=(TextView)findViewById(R.id.TextView01);
2  button.setOnClickListener(new View.OnClickListener(){
3      public void onClick(View view){
```

```
4           textView.setText("Button 按钮");
5       }
6   });
7   imageButton.setOnClickListener(new View.OnClickListener(){
8       public void onClick(View view){
9           textView.setText("ImageButton 按钮");
10      }
11  });
```

在第2行代码中，button对象通过调用setOnClickListener()函数，注册一个点击(Click)事件的监听器View.OnClickListener()。第3行代码是点击事件的回调函数。第4行代码将TextView的显示内容更改为“Button按钮”。

View.OnClickListener()是View定义的点击事件的监听器接口，并在接口中仅定义了onClick()函数。当Button从Android界面框架中接收到事件后，首先检查这个事件是否是点击事件，如果是点击事件，同时Button又注册了监听器，则会调用该监听器中的onClick()函数。

每个View仅可以注册一个点击事件的监听器，如果使用setOnClickListener()函数注册第二个点击事件的监听器，之前注册的监听器将被自动注销。为每个按钮注册一个点击事件监听器，每个按钮的事件处理程序都有各自的onClick()函数，这样能够使代码更加清晰、易读，且易于维护。当然，也可以将多个按钮注册到同一个点击事件的监听器上，示例代码如下。

```
1   Button.OnClickListener buttonListener=new Button.OnClickListener(){
2           @Override
3           public void onClick(View v){
4               switch(v.getId()){
5                   case R.id.Button01:
6                       textView.setText("Button 按钮");
7                       return;
8                   case R.id.ImageButton01:
9                       textView.setText("ImageButton 按钮");
10                      return;
11              }
12          }};
13      button.setOnClickListener(buttonListener);
14      imageButton.setOnClickListener(buttonListener);
```

第1～12行代码定义了一个名为buttonListener的点击事件监听器，第13行和第14行代码分别将该监听器注册到Button和ImageButton上。

5.2.3 CheckBox和RadioButton

CheckBox是同时可以选择多个选项的控件，而RadioButton则是仅可以选择一个选

项的控件。RadioGroup 是 RadioButton 的承载体，程序运行时不可见，应用程序中可能包含一个或多个 RadioGroup。RadioGroup 包含多个 RadioButton，在一个 RadioGroup 中，用户仅能够选择其中的一个 RadioButton。在图 5.5 中，RadioGroup 中包含两个 RadioButton，当选择 RadioButton01 后，RadioButton02 自动变为非选择状态。

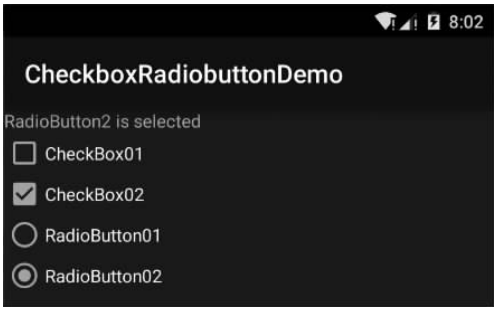

图 5.5 CheckboxRadiobuttonDemo 示例

CheckboxRadiobuttonDemo 示例如图 5.5 所示，从上至下分别是 TextView01、CheckBox01、CheckBox02、RadioButton01 和 RadioButton02。在 XML 文件(/res/layout/main.xml)中的代码如下。

```
<TextView android:id="@+id/TextView01"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"/>
<CheckBox android:id="@+id/CheckBox01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="CheckBox01" >
</CheckBox>
<CheckBox android:id="@+id/CheckBox02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="CheckBox02" >
 </CheckBox>
<RadioGroup android:id="@+id/RadioGroup01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <RadioButton android:id="@+id/RadioButton01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="RadioButton01" >
    </RadioButton>
    <RadioButton android:id="@+id/RadioButton02"
        android:layout_width="wrap_content"
```

```
25          android:layout_height="wrap_content"
26          android:text="RadioButton02" >
27      </RadioButton>
28  </RadioGroup>
```

第15行代码中的<RadioGroup>标签声明了一个RadioGroup，第18行和第23行代码分别声明了两个RadioButton，这两个RadioButton是RadioGroup的子元素。

在代码中引用CheckBox和RadioButton的方法可以参考下面的代码。

```
1  CheckBox checkBox1=(CheckBox)findViewById(R.id.CheckBox01);
2  RadioButton radioButton1=(RadioButton)findViewById(R.id.RadioButton01);
```

CheckBox设置点击事件监听器的方法，与Button中介绍的方法相似，唯一不同在于将Button.OnClickListener换成了CheckBox.OnClickListener，下面给出简要代码。

```
1  CheckBox.OnClickListener checkboxListener=new CheckBox.OnClickListener(){
2      @Override
3      public void onClick(View v){
4        //过程代码
5      }};
6  checkBox1.setOnClickListener(checkboxListener);
7  checkBox2.setOnClickListener(checkboxListener);
```

RadioButton设置点击事件监听器的方法。

```
1  RadioButton.OnClickListener radioButtonListener=new RadioButton
   .OnClickListener(){
2      @Override
3      public void onClick(View v){
4        //过程代码
5      }};
6  radioButton1.setOnClickListener(radioButtonListener);
7  radioButton2.setOnClickListener(radioButtonListener);
```

5.2.4 Spinner

Spinner是从多个选项中选择一个选项的控件，类似于桌面程序的组合框(ComboBox)，但没有组合框的下拉菜单，而是使用浮动菜单为用户提供选择。

SpinnerDemo示例如图5.6所示，从上至下分别是TextView01和Spinner01。在XML文件(/res/layout/main.xml)中的代码如下。

```
1  <TextView  android:id="@+id/TextView01"
2      android:layout_width="match_parent"
3      android:layout_height="wrap_content"
4      android:text="@string/hello"/>
5  <Spinner android:id="@+id/Spinner01"
```

```
6       android:layout_width="300dip"
7       android:layout_height="wrap_content">
8   </Spinner>
```

图 5.6 SpinnerDemo 示例

第 5 行代码使用<Spinner>标签声明了一个 Spinner 控件，并在第 6 行代码中指定该控件的宽度为 300dip。

在 SpinnerDemo.java 文件中，定义一个 ArrayAdapter 适配器，在 ArrayAdapter 中添加在 Spinner 中可以选择的内容。为了使程序能够正常运行，需要在代码中引入 android.widget.ArrayAdapter 和 android.widget.Spinner。

```
1   Spinner spinner=(Spinner)findViewById(R.id.Spinner01);
2   List<String>list =new ArrayList<String>();
3   list .add("Spinner 子项 1");
4   list .add("Spinner 子项 2");
5   list .add("Spinner 子项 3");
6   ArrayAdapter< String > adapter = new ArrayAdapter < String > (this, android. R.
    layout.simple_spinner_item, list);
7   adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
8   spinner.setAdapter(adapter);
```

第 2 行代码建立了一个字符串数组列表(ArrayList)，这种数组列表可以根据需要进行增减，<String>表示数组列表中保存的是字符串类型的数据。在代码的第 3～5 行中，使用 add()函数分别向数组列表中添加 3 个字符串。第 6 行代码建立了一个 ArrayAdapter 的数组适配器，数组适配器能够将界面控件和底层数据绑定在一起。在这里，ArrayAdapter 将 Spinner 和 ArrayList 绑定在一起，所有 ArrayList 中的数据将显示在 Spinner 的浮动菜单中，绑定过程由第 8 行代码实现。第 7 行代码设定了 Spinner 浮动菜单的显示方式，其中，android.R.layout.simple_spinner_dropdown_item 是 Android 系统内置的一种浮动菜单，如图 5.6 所示。如果将其改为 android.R.layout.simple_spinner_item，则显示效果如图 5.7 所示。

为了保证用户界面显示的内容与底层数据一致，应用程序需要监视底层数据的变化，如果底层数据更改了，则用户界面也需要修改显示内容。在使用适配器绑定界面控件和底层数据后，应用程序就不需要再监视底层数据的变化，从而极大地简化了代码的复杂性。

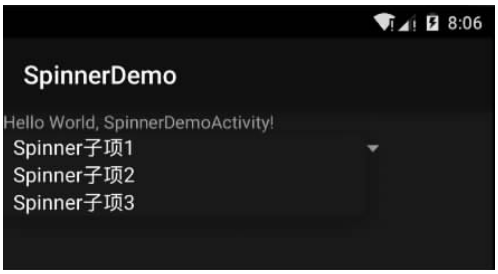

图 5.7　Spinner 的 item 菜单

5.2.5　ListView

ListView 是用于垂直显示的列表控件，如果显示内容过多，则会出现垂直滚动条。ListView 是在界面设计中经常使用的界面控件，其原因是 ListView 能够通过适配器将数据和显示控件绑定，在有限的屏幕上提供大量内容供用户选择；而且它支持点击事件，可以用少量的代码实现复杂的选择功能。

ListViewDemo 示例如图 5.8 所示，从上至下分别是 TextView01 和 ListView01。在 XML 文件(/res/layout/main.xml)中的核心代码如下。

```
1  <TextView android:id="@+id/TextView01"
2      android:layout_width="match_parent"
3      android:layout_height="wrap_content"
4      android:text="@string/hello" />
5  <ListView android:id="@+id/ListView01"
6      android:layout_width="wrap_content"
7      android:layout_height="wrap_content">
8  </ListView>
```

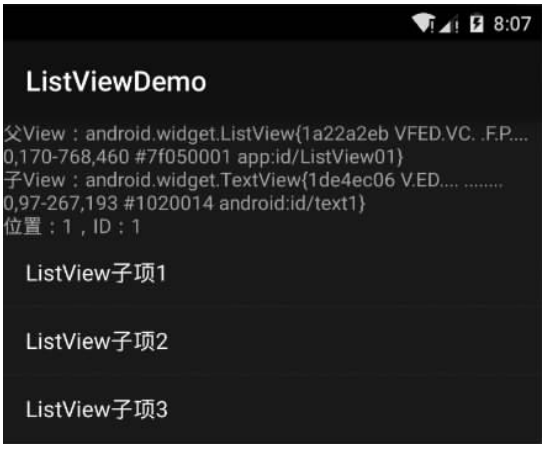

图 5.8　ListViewDemo 示例