第 5 章　界面布局

本章主要内容：

- Android UI 的 5 种布局。
- Android UI 控件。
- 用户界面设计的原则以及核心概念。
- 菜单以及对话框的简要介绍。
- 滚动处理。

本章讲述了界面布局的相关概念，通过本章内容的学习，读者可以利用 Android 平台提供一套图形用户界面的编程接口，快速掌握图形用户界面的开发。

5.1　Android UI 布局

UI（User Interface）即用户界面，指用户能看到、能操作的部分。UI 设计则是指对软件的人机交互、操作逻辑、界面美观的整体设计。好的 UI 设计不仅使软件变得有个性、有品味，还使软件的操作变得舒适、简单、自由、充分体现软件的定位和特点。

5.1.1　线性布局

这种布局方式是最常见的一种，顾名思义就是每个组件是按照从前到后或者从上到下的方向逐个排列的，这种布局适合上下、左右对齐的相同数量组件以表格方式布局的情况。

线性布局（LinearLayout）根据 orientation 属性值，将包含的所有控件或布局对象排列在同一个方向：水平（horizontal）（图 5-1）或垂直（vertical）（图 5-2）。

图 5-1　水平（horizontal）

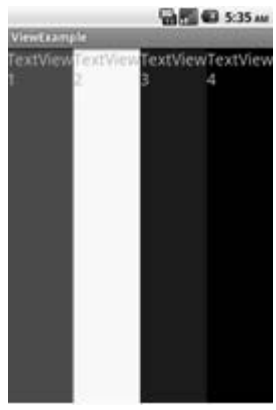

图 5-2　垂直（vertical）

在 LinearLayout 中有几个重要的属性值，如表 5-1 所示。

表 5-1　LinearLayout 重要的属性值

属性值	说明
android:orientation	设置控件或者容器存放的方式
android:id	设置控件 id，方便在使用时找到其引用
android:layout_width	容器的宽度，该值必须设置
android:layout_height	容器的高度，该值必须设置
android:layout_weight	该属性针对其内的子控件，存放在 LinearLayout 中的控件都有这个属性，用来设置该控件或者容器占父控件或者容器的比例

需要注意的，LinearLayout 有两个非常相似的属性：

```
android:gravity
android:layout_gravity
```

它们的区别在于：

（1）“android:gravity”属性是对该 view 中内容的限定。用于设置 view 中内容相对于 view 组件的对齐方式。例如，一个 button 上面的 text，可以设置该 text 相对于 view 的靠左、靠右等位置。

（2）android:layout_gravity 是用来设置该 view 相对于父 view 的位置。用于设置 view 组件相对于 container 的对齐方式。例如，一个 button 在 LinearLayout 里，若要想把该 button 放在 LinearLayout 里靠左、靠右等位置就可以通过该属性设置。

可以通过设置 android:gravity="center"来让 EditText 中的文字在 EditText 组件中居中显示，同时设置 EditText 的 android:layout_gravity="right"来让 EditText 组件在 LinearLayout 中居右显示。

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<EditText
    android:layout_width="wrap_content"
    android:gravity="center"
    android:layout_height="wrap_content"
    android:text="one"
    android:layout_gravity="right"/>
</LinearLayout>
```

5.1.2　帧布局

帧布局（FrameLayout）是指该容器内放置的控件或者容器没有上下左右的关系，只有

层叠前后的关系。放置在容器内的控件按放置的前后顺序逐一层叠摆放，自然地后面摆放的控件就将前面摆放的控件覆盖了，叠在它的上面了。

FrameLayout 被定制为用户屏幕上的一个空白备用区域，之后用户可以在其中填充一个单一对象。例如，一张用户要发布的图片。所有的子元素将会固定在屏幕的左上角；用户不能为 FrameLayout 中的一个子元素指定一个位置。后一个子元素将会直接在前一个子元素之上进行覆盖填充，把它们部分或全部挡住（除非后一个子元素是透明的）。

里面可以放多个控件，不过控件的位置都是相对位置。可以通过设置属性“android：bringToFront="true|false"”将前面放置的控件提到最前面可见。

5.1.3　相对布局

相对布局（Relative Layout）是指利用控件之间的相对位置关系来对布局进行放置。换句话说，在该容器中的控件与其他任何一个控件或者容器（包括父控件）有相对关系。因此，可以以右对齐，或上下，或置于屏幕中央的形式来排列两个元素。元素按顺序排列，因此如果第一个元素在屏幕的中央，那么相对于这个元素的其他元素将以屏幕中央的相对位置来排列。如果使用 XML 来指定这个 layout，在定义它之前，被关联的元素必须定义。

这个布局是最灵活的布局，因此复杂的布局多用这个布局。

LayoutParams 中特殊的参数如下。

（1）属性值为 true 或 false，如表 5-2 所示。

表 5-2　属性值为 true 或 false

属性名称	说　明
android:layout_centerHrizontal	水平居中
android:layout_centerVertical	垂直居中
android:layout_centerInparent	相对于父元素完全居中
android:layout_alignParentBottom	贴紧父元素的下边缘
android:layout_alignParentLeft	贴紧父元素的左边缘
android:layout_alignParentRight	贴紧父元素的右边缘
android:layout_alignParentTop	贴紧父元素的上边缘
android:layout_alignWithParentIfMissing	若找不到兄弟元素以父元素做参照物

（2）属性值必须为 id 的引用名（@id/id-name），如表 5-3 所示。

表 5-3　属性值必须为 id 的引用名（@id/id-name）

属性名称	说　明
android:layout_below	在某元素的下方
android:layout_above	在某元素的上方
android:layout_toLeftOf	在某元素的左边
android:layout_toRightOf	在某元素的右边
android:layout_alignBaseLine	该控件的 baseline 和给定 ID 的控件的 Baseline 对齐
android:layout_alignTop	本元素的上边缘和某元素的上边缘对齐

（续表）

属性名称	说　明
android:layout_alignLeft	本元素的左边缘和某元素的左边缘对齐
android:layout_alignBottom	本元素的下边缘和某元素的下边缘对齐
android:layout_alignRight	本元素的右边缘和某元素的右边缘对齐

5.1.4　表格布局

表格布局（TableLayout）指该容器是一个表格，放置控件时，控件的位置坐落在表格的某个位置上。其中 TableRow 是配合 TableLayout 使用的，目的是为了让 TableLayout 生成多个列，否则 TableLayout 中就只能存在一列元素，但可以有多行。

TableLayout 的直接父类是 LinearLayout，所以其具有 LinearLayout 的属性，TableLayout 中的每一行用 TableRow 表示，每一列就是 TableRow 中的个数指定的。TableRow 的直接父类是 LinearLayout，但是其放置的方式只能水平放置。TableLayout 的重要属性如表 5-4 所示。

表 5-4　TableLayout 的重要属性

属性名称	说　明
android:stretchColumns	伸展的列的索引
android:shrinkColumns	收缩的列的索引
android:collapseColumns	折叠的列的索引

5.1.5　绝对布局

绝对布局（Absolute Layout）是指以屏幕左上角为坐标原点（0,0），控件在容器中的位置以坐标的形式存在，可以随意指定控件的坐标位置，非常灵活。

此种布局在开发过程中很少使用，原因是屏幕兼容性不好，不便控制两个控件之间的位置。AbsoluteLayout 属性如表 5-5 所示。

表 5-5　AbsoluteLayout 属性

属性名称	说　明
android:layout_x	x 方向的坐标
android:layout_y	y 方向的坐标

5.2　Android UI 控件

AndroidSDK 包含一个名为 android.widget 的 Java 包。当提及控件时，通常指该包中的某个类。控件涵盖 Android SDK 中几乎所有可绘制到屏幕上的东西，包括 ImageView、FrameLayout、EditText 和 Button 对象。通常所有的控件都是从 View 类继承而来。

UI 控件就是为用户界面提供服务的视图对象，它是所有具有事件处理控件的父类。

UI 控件的三要素：绘制、数据、控制。

首先，展现在人们视线里的是可见的，那就是绘制，每一个控件都有自己的样子，就跟人的相貌一样，如 TableView 是一张数据表，又如 datePicker 是一个时间选择器，它们的样子都是不一样的。

然后是数据，控件也需要自己的数据，如 label，需要显示文字的数据，如 imageView，需要显示图片的数据，如果没有数据这些控件的使用将会变得没有意义。

最后一个就是控制了，最典型的就是 button 了，这是用户与界面交互的关键，还有其他的控件，如 scrollView，可以滑动加载数据，这是控制。

Android 提供的 UI 控件分别包括了几种 Layout 和多种组件（widget），如 Button、TextView、EditText 等。

5.2.1　UI 事件捕获与处理

事件在图形界面（UI）的开发中，有两个非常重要的内容：一个是控件的布局，另一个就是控件的事件处理。其中，Android 在事件处理过程中主要涉及以下 3 个概念。

（1）事件（Event）。事件表示用户在图形界面的操作的描述，通常是封装成各种类，如键盘事件操作相关的类为 KeyEvent、触摸屏相关的移动事件类为 MotionEvent 等。它可以分为触摸事件、晃动事件、远程控制事件三类。

（2）事件源。事件源是指事件发生的场所，通常是指各个控件，如 Button、EditText 等。

（3）事件处理者。事件处理者是指接收事件对象并对其进行处理的对象，事件处理一般是一个实现某些特定接口类创建的对象。

当 UI 控件被添加到应用程序用户界面后，一部分控件需要对用户的操作事件进行捕获和响应处理，只有实现了对事件的处理才能算是与用户进行了交互。此过程中就包括了响应和处理两个过程，其中响应过程就涉及 Android 对 UI 事件提供的一系列事件响应函数和回调函数，而处理过程则是这些函数中的具体实现代码。控件捕获用户操作事件的方式有以下两种。

（1）定义一个事件监听器并将其绑定到相应的控件。用于监听用户事件，view 类包含了一系列命名类似于 On<Action﹣name>Listener 的接口，而每一个接口都提供了一个命名类似于 On<Action﹣name>()的回调方法。例如，响应视图单击事件的接口和方法分别是 View.OnClickListener 和 onClick()方法，所以如果某控件需要在它被单击时获得通知就需要实现 OnClickListener 接口并定义其 onClick 回调方法，然后通过控件的 setOnClickListener() 方法进行注册绑定。

（2）重写回调方法。这种方式主要用于自主实现的控件类，这种方式允许为自主实现的控件上接收到的每个事件定义默认的处理行为，并决定是否需要将事件传递给其他的子视图。

5.2.2　TextView

TextView（标签文本）用来显示文本信息，它包含了一段提示文字，作为另一个控件的搭配说明。

通常可以在 XML 文件中设置其相应的属性。XML 文件中 TextView 属性如表 5-6 所示。

表 5-6　XML 文件中 TextView 属性

属性名称	说　明
android：layout_height	该控件显示时的高度
android：layout_width	该控件显示时的宽度
android:id="@+id/textView1"	该控件的 id，在布局文件中或者代码中被引用
android:textStyle="bold"	TextView 里面的字加粗显示
android:layout_height="wrap_content"	该控件的高度为其包含内容的高度
android:layout_width="wrap_content"	该控件的宽度为其包含内容的宽度
android:text="@string/signin"	显示的内容，这里表示存放在 string.xml 文件中 name=signin 的文本
android:layout_height="40dip"	设置具体的高度
android:textColor="#7089c0"	设置文本的颜色
android:textSize="18sp"	设置文本的大小
android:gravity="center_vertical"	设置文本纵向居中
android:paddingLeft="5dip"	设置内边距
android:layout_marginTop="5dip"	设置外边距

5.2.3　Button

Button（按钮）是最常用的控件，不管发生什么改变，都需要通过单击按钮来触发。Button 是相应单击事件，可以将 Button 理解为可以单击的 TextView，其使用方法与 TextView 设置一样，区别在于 Button 可以有按键的效果和事件的监听。

常用方法：super.findViewById(id)得到在 layout 中声明的 Button 的引用，setOnClickListener（View.OnClickListener）添加监听。然后在 View.OnClickListener 监听器中使用 v.equals(View) 方法判断哪个按钮被按下，进行分别处理。

关于 android:onClick="onLoginClick"，该属性需要在源代码中设置一个 onLoginClick 方法，作为该 Button 的单击监听方法：

```
    public void onLoginClick(View v)
    {
       if(TextUtils.isEmpty(name.getText().toString())){
name.setError(getString(R.string.no_empyt_name));
          return;
       }
//省略...
    }
```

这样编写的好处在于可以直接完成按键监听，不必通过调用 findViewById(int id)找到该 Button，然后再为其设置单击监听器 setOnClickListener(OnClickListener)。

示例：

```
<Button
android:layout_height="40dip" android:layout_width="wrap_content"
android:minWidth="100dip"
```

```
android:layout_marginLeft="0dip"
android:layout_marginRight="2dip"
android:layout_marginTop="5dip"
android:layout_marginBottom="10dip"
android:background="@drawable/button"
android:text="@string/login"
android:textColor="#fff"
android:textSize="18sp"
android:id="@+id/login"
android:layout_alignRight="@+id/password"
android:layout_below="@+id/password"
android:onClick="onLoginClick"
/>
```

5.2.4　EditText

EditText（文本输入框）用来编辑输入文本信息，接收用户的输入。

EditText 属性的大部分设置与 TextView 是一样的，这里仅介绍 EditText 与 TextView 不同的属性，如表 5-7 所示。

表 5-7　XML 文件中 EditText 属性

属性名称	说　明
android:hint="@string/name"	输入之前的提示，当 EditText 获得输入焦点，并输入文字时，该文本自动消失，起提示的作用
android:singleLine="true"	该文本输入框不可换行输入，只能在一行内输入文本
android:password="true"	该文本输入框是用来输入密码的，输入的文本会自动转换为“ •”，起到隐藏用户密码的作用

5.2.5　CheckBox 与 RadioGroup

在实际的应用程序中，往往会接触到在几个选项中选择一个或者多个选项的操作需要，如批量删除名片、设置情景模式的操作，这时就需要用到复选框（CheckBox）或者单选组框（RadioGroup）。

其中，RadioGroup 需要和 RadioButton（单选按钮）一起使用，RadioButton 要声明在 RadioGroup 中，RadioGroup 是线性布局 android.widget.LinearLayout 的子类。一个 RadioGroup 默认带有 3 个 RadioButton，可以根据实际情况添加或者删除，需要在.xml 的代码文件里面进行操作。

CheckBox、RadioButton 与普通按钮不同的是，它多了一个可选中的功能。因此都可给其额外指定一个“Android：checked”属性，该属性指定 CheckBox、RadioButton 初始是否被选中。

5.2.6　Spinner

Spinner（下拉列表）用来显示列表项，类似于一组单选框 RadioButton。下拉列表也是

一种很常用的 UI 控件，例如很多网页都会使用下拉列表，根据选择下拉列表选项来设置信息，在 Android 中也提供了下拉列表的实现。

除了在程序中对下拉列表赋值之外，还可以将要赋值的内容先写在资源文件 strings.xml 里面，然后再通过适配器赋值。这里接触到了 xml 文件的另外一种用法，即用于存放一组数据，如字符串、数组等，Android 会将这些映射关系自动生成到 R.java 文件中，之后在 Java 代码中就可以通过相应的 id 和 name 来访问这些数据了。在 strings.xml 里是通过类似如下的代码来声明字符串数组的：

```
<resources>
<string-array name="cities">
    <item>北京</item>
    <item>上海</item>
    <item>成都</item>
</string-array>
</resources>
```

然后在代码中利用 Android 通过 ArrayAdapter.createFromResource()方法来获取这个字符串数组，再将这个数组资源的 Adapter 通过 setAdapter()方法与 Spinner 建立关联，Spinner 就能够使用这个数组来初始化下拉列表的内容了。

5.2.7 AutoCompleteTextView

AutoBompleteTextView（自动补全文本框）是一个可编辑的文本视图，能显示用户输入的相关信息。建议列表显示一个下拉菜单，用户可以从中选择一项，以完成输入。建议列表是从一个数据适配器获取的数据。它有以下 3 个重要的方法。

（1）clearListSelection()：清除选中的列表项。

（2）dismissDropDown()：如果存在关闭下拉菜单。

（3）getAdapter()：获取适配器。

对于 AutoCompleteTextView 比较值得注意的一个属性就是 Threshold（阈值），即从多少位开始自动补全，可以根据需求进行设置，若不进行设置则默认阈值为 2，即只有当输入位数达到两位及以上才会触发自动补全的功能。

5.2.8 ProgressBar

ProgressBar（进度条）是改善用户体验的一种重要的控件，进度条的存在可以防止应用程序处于“假死”的状态，避免用户漫无目的的等待，也能够实时地反映出程序运行的状态。在 Android 系统中有两种进度条：一种是圆形进度条；另一种是水平进度条。但通常 Android 中进度条默认为圆圈形式，要显示水平进度条，可以在 xml 文件中设置进度条的 style 属性，其方法如下：

```
style="?android:attr/progressBarStyleHorizontal"   水平进度条
style="?android:attr/progressBarStyleLarge"        较大的进度条
```

```
style="?android:attr/progressBarStyleSmallTitle     标题大小的进度条
```

5.2.9　ListView

ListView（列表）是 Android 中非常重要也相对复杂的一个控件，它将需要显示的内容以列表的形式展示出来，并且能够根据数据的长度适当地调节显示，如名片夹的显示、列表菜单的显示、音乐播放器中的歌曲名列表等，都用到了列表这个组件。

值得注意的是，ListView 的选项可以设置单击监听和选择监听，就是当单击或选择 ListView 的某一个 Item 时，可以分别做出不同的响应。

一个 ListView 要显示其相关内容，需要满足三个条件：需要 ListView 显示的数据、与 ListView 相关联的适配器（Adapter）、一个 ListView 对象来显示内容。

ListView 可用的适配器又有以下三种。

（1）ArrayAdapter，可称为数组适配器，是 ListView 中最简单的一种适配器，它将一个数组和 ListView 之间建立连接，可以将数组里定义的内容一一对应的显示在 ListView 中，每一项一般只有一个 TextView，即一行只能显示一个数组 Item 调用 toString()方法生成的一行字符串。

（2）SimpleAdapter，是扩展性最好的一种适配器，通过这个适配器，可以让 ListView 中的每一项内容可以自定义出各种效果，可以将 ListView 中某一项的布局信息直接写在一个单独的 xml 文件中，通过 R.layout.layout_name 来获得这个布局（layout_name 是 xml 布局文件的名字）。

（3）SimpleCursorAdapter，是 SimpleAdapter 与数据库的简单结合，通过这个适配器，ListView 能方便地显示对应的数据库中的内容。

5.2.10　Window

在开发程序时经常会需要软件全屏显示、自定义标题（使用按钮等控件）和其他的需求，这就需要控制 Android 应用程序的窗体显示。图 5-3～图 5-6 所示为几种不同的窗体风格。

图 5-3　自定义标题栏

图 5- 4　隐藏标题栏

图 5-5　标题栏左端显示图标

图 5-6　隐藏状态栏

Android 应用程序的窗体显示，主要使用 requestWindow Feature(featrueId)方法设置，它的功能是启用窗体的扩展特性，其参数是 Window 类中定义的枚举常量。另外，还可以通过 Window 类的 setFlags()方法来隐藏系统的状态栏。当一个 Activity 设置了同时隐藏标题栏和状态栏时，就是全屏显示的状态了。

部分 Window 类中定义的枚举常量如表 5-8 所示。

表 5-8　部分 Window 类中定义的枚举常量

常　量	说　明
DEFAULT_FEATURES	系统默认的状态
FEATURE_CONTEXT_MENU	启动 ContextMenu，默认启动该项
FEATURE_CUSTOM_TITLE	当需要自定义标题的时候指定
FEATURE_INDETERMINATE_PROGRESS	在标题栏上不确定的进度
FEATURE_LEFT_ICON	标题栏左侧显示图标
FEATURE_NO_TITLE	无标题栏
FEATURE_OPTION_PANEL	启动选项面板功能，默认启动
FEATURE_PROGRESS	进度指示器功能
FEATURE_RIGHT_ICON	标题栏右侧显示图标

5.2.11　其他 UI 控件概览

前几小节中，学习了一部分最常用的 UI 控件，基本的用法都是将其放置在其所隶属的 Layout 上，对控件所具有的属性值进行预设或者在代码中对各属性进行需要的更改即可，这些属性可以在 SDK 文档中查阅到。本小节将通过简单列举的方法来了解其他的 UI 控件。

（1）WebView（网络视图），如图 5-7 所示。

（2）GridView（网格视图），如图 5-8 所示。

（3）Gallery（画廊视图），如图 5-9 所示。初始状态如图 5-10 所示。

图 5-7　网络视图

图 5-8　网格视图

图 5-9　画廊视图

图 5-10　初始状态

（4）DatePicker&TimePicker（日期和时间），如图 5-11 和图 5-12 所示。

图 5-11　设置日期

图 5-12　设置时间

（5）ExpandableListView（可展开、收缩列表），如图 5-13 和图 5-14 所示。

（6）RatingBar（评分条），如图 5-15 所示。

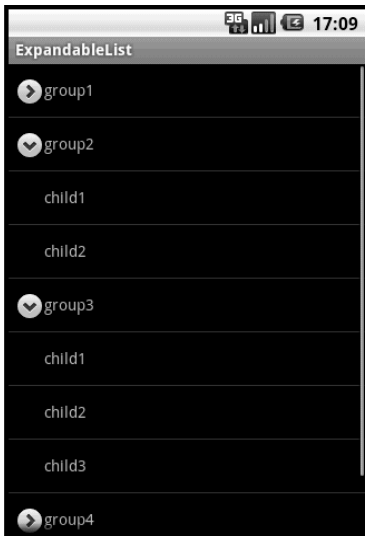

图 5-13　展开列表

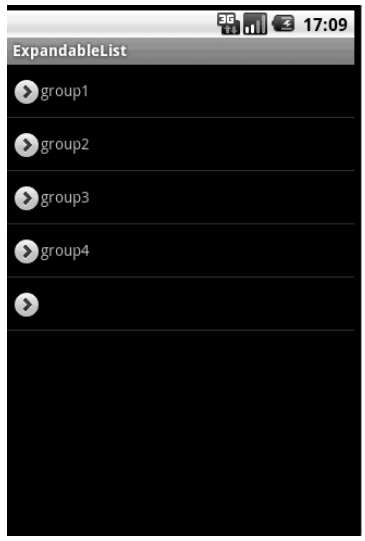

图 5-14 收缩列表

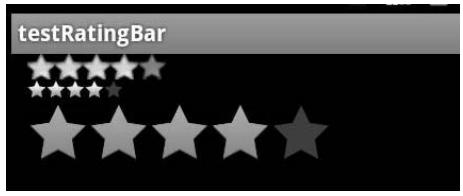

图 5-15　评分条

（7）SlidingDrawer（滑动式抽屉），如图 5-16 和图 5-17 所示。

图 5-16　隐藏状态

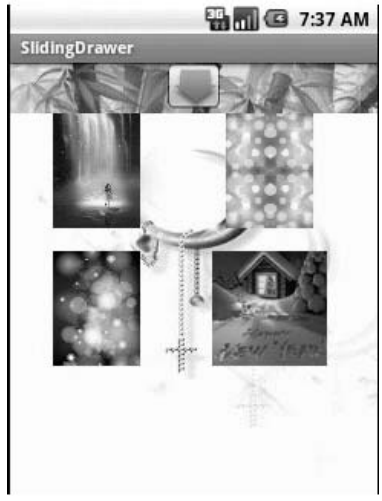

图 5-17 正在滑动

（8）ZoomControls（缩放控件），如图 5-18～图 5-20 所示。

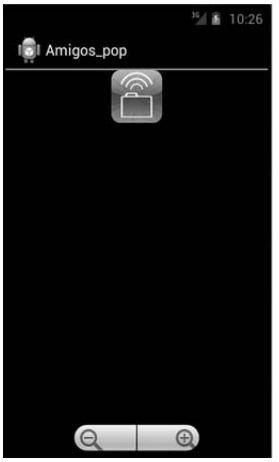

图 5-18　初始状态

图 5-19　放大

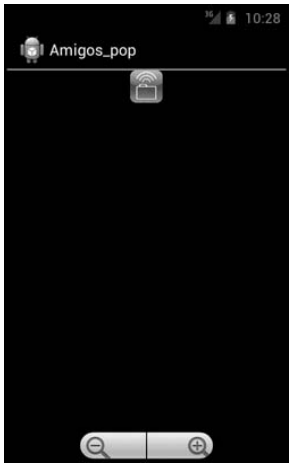

图 5-20　缩小

5.3　用户界面设计原则

在人和机器的互动过程（Human Machine Interaction）中，有一个层面，即人们所说的界面（Interface）。从心理学意义来分，界面可分为感觉（视觉、触觉、听觉等）和情感两个层次。用户界面设计是屏幕产品的重要组成部分。用户界面设计的三大原则是：置界面于用户的控制之下；减少用户的记忆负担；保持界面的一致性。下面将向读者简要介绍其设计规范。

5.3.1　一致性

坚持以用户体验为中心设计原则，界面直观、简洁，操作方便快捷，用户接触软件后对界面上对应的功能一目了然，不需要太多培训就可以方便使用本应用系统。

1. 字体

（1）保持字体及颜色一致，避免一套主题出现多个字体。

（2）不可修改的字段，统一用灰色文字显示。

2. 对齐

保持页面内元素对齐方式的一致，如无特殊情况应避免同一页面出现多种数据对齐方式。

3. 表单录入

（1）在包含必须与选填的页面中，必须在必填项旁边给出醒目标识（*）。

（2）各类型数据输入需限制文本类型，并做格式校验如电话号码输入只允许输入数字、邮箱地址需要包含“@”等，在用户输入有误时给出明确提示。

4. 鼠标手势

可单击的按钮、链接需要切换鼠标手势至手形。

5. 保持功能及内容描述一致

避免同一功能描述使用多个词汇，如编辑和修改、新增和增加、删除和清除混用等。建议在项目开发阶段建立一个产品词典，包括产品中常用术语及描述，设计或开发人员严格按照产品词典中的术语词汇来显示文字信息。

5.3.2 准确性

（1）使用一致的标记、标准缩写和颜色，显示信息的含义应该非常明确，用户不必再参考其他信息源。

（2）显示有意义的出错信息，而不是单纯的程序错误代码。

（3）避免使用文本输入框来放置不可编辑的文字内容，不要将文本输入框当成标签使用。

（4）使用缩进和文本来辅助理解。

（5）使用用户语言词汇，而不是单纯的专业计算机术语。

（6）高效地使用显示器的显示空间，但要避免空间过于拥挤。

（7）保持语言的一致性，如“确定”对应“取消”、“是”对应“否”。

5.3.3 布局合理化

在进行 UI 设计时需要充分考虑布局的合理化问题，遵循用户从上而下，自左向右浏览、操作习惯，避免常用业务功能按键排列过于分散，以造成用户鼠标移动距离过长的弊端。多做“减法”运算，将不常用的功能区块隐藏，以保持界面的简洁，使用户专注于主要业务操作流程，有利于提高软件的易用性及可用性。

1. 菜单

（1）保持菜单简洁性及分类的准确性，避免菜单深度超过 3 层。

（2）菜单中功能是需要打开一个新页面来完成的，需要在菜单名字后面加上“…”。

2. 按钮

确认操作按钮放置左边，取消或关闭按钮放置于右边。

3. 功能

未完成功能必须隐藏处理，不要置于页面内容中，以免引起误会。

4. 排版

所有文字内容排版避免贴边显示（页面边缘），尽量保持 10～20 像素的间距并在垂直方向上居中对齐；各控件元素间也保持至少 10 像素以上的间距，并确保控件元素不紧贴于页面边沿。

5. 表格数据列表

字符型数据保持左对齐，数值型数据右对齐（方便阅读对比），并根据字段要求，统一

显示小数位位数。

6. 滚动条

页面布局设计时应避免出现横向滚动条。

7. 信息提示窗口

信息提示窗口应位于当前页面的居中位置，并适当弱化背景层以减少信息干扰，让用户把注意力集中在当前的信息提示窗口。一般做法是在信息提示窗口的背面加一个半透明颜色填充的遮罩层。

5.3.4　操作合理性

（1）尽量确保用户在不使用鼠标（只使用键盘）的情况下也可以流畅地完成一些常用的业务操作，各控件间可以通过 Tab 键进行切换，并将可编辑的文本全选处理。

（2）查询检索类页面，在查询条件输入框内按回车键应该自动触发查询操作。

（3）在进行一些不可逆或者删除操作时应该有信息提示用户，并让用户确认是否继续操作，必要时应该把操作造成的后果也告诉用户。

5.3.5　响应时间

系统响应时间应该适中，响应时间过长，用户就会感到不安和沮丧，而响应时间过快也会影响到用户的操作节奏，并可能导致错误。因此在系统响应时间上坚持如下原则。

（1）2～5 秒窗口显示处理信息提示，避免用户误认为没响应而重复操作。

（2）5 秒以上显示处理窗口，或显示进度条。

（3）一个长时间的处理完成时应给予完成警告信息。

5.4　用户界面设计核心概念

5.4.1　android.view.View 类

Android 应用中的每个界面都是一个 Activity，Activity 上展现的都是 Android 系统中的可视化组件，这些组件都是从 android.view.View 继承。

5.4.2　View 类的继承关系

View 类有很多子类，分为以下三种。

（1）布局类(Layout)：继承于 ViewGroup 类。

（2）视图容器类(View Container)：继承于 ViewGroup 类。

（3）视图类：继承于 View 类。

创建 View 有两种方式：使用 XML 创建 View 与使用代码创建 View。

1. 使用 XML 创建 View

Android 图形用户界面上的组件可以使用 XML 文件创建，其中 XML 文件中使用属性指定组件的属性，如 id 等；XML 文件放置在 res/layout 下。

View 类的继承关系如图 5-21 所示。

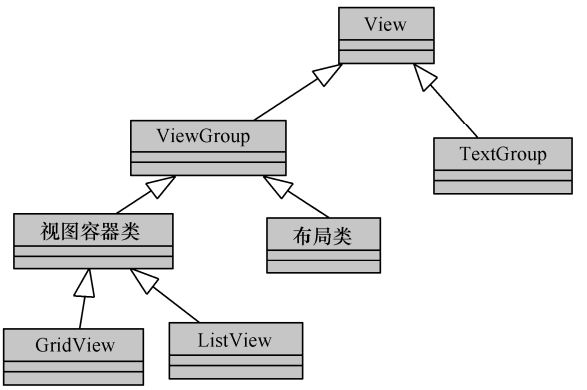

图 5-21　View 类的继承关系

2. 使用代码创建 View

每一个视图组件都是一个 View 类型的对象，可以在代码中使用视图类的构造方法创建 View 对象，并且可以调用 View 对象的 setXXX 方法，设置其属性。

5.5　菜　　单

Android 中菜单（Menu）和对话框（Dialog）的设计对于人机的交换是非常人性化的。菜单提供了不同功能分组展示的能力，菜单包括选项菜单（Option Menu）、上下文菜单（Context Menu）、子菜单（Sub Menu）。

菜单在 Android 的应用程序中相对用得较多，有时为了界面的美观，可以把一些按钮适当换成菜单。上面提到的菜单有 3 种，创建和处理这 3 种菜单都要覆盖相应的事件。

5.5.1　选项菜单

不管在模拟器还是真机上面都有一个 Menu 键，单击该键后就会弹出一个菜单，此菜单就是选项菜单。选项菜单的菜单项最多只能有 6 个，如果超过 6 个，系统会自动将最后一个菜单项显示为“更多”，单击“更多”按钮时会展开隐藏的菜单。下面来看一个系统自带的选项菜单，当单击“Menu”键时就会出现如图 5-22 所示的菜单。

在开发程序时，也经常会用到选项菜单，下面通过一个实例来说明如何创建自己的选项菜单。在玩游戏时，对于声音的控制可以采用选项菜单，如图 5-23 所示。

图 5-22　系统自带菜单

图 5-23　选项菜单

创建菜单的步骤如下。

（1）覆盖方法 onCreateOptionsMenu，通过 Menu 中的一个 add 方法新建菜单并且添加菜单项。

```
/**
* 覆盖该方法添加菜单
*/
@Override
public boolean onCreateOptionsMenu(Menu menu) {
//添加菜单项
menu.add(0, 0, 0, "声音：关");
menu.add(0,1,0,"声音：开");
return super.onCreateOptionsMenu(menu);
}
```

（2）单击每一个菜单项，可以进行相应的操作，需要覆盖方法 onOptionsItemSelected，根据菜单的每个 ID 进行判断。

```
/**
*覆盖方法，对选项菜单进行操作
*/
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {//根据菜单的 ID
  case 0:
  Toast.makeText(OptionMenuDemoActivity.this, "声音已经关闭！！",
  Toast.LENGTH_SHORT).show();
  break;
  case 1:
  Toast.makeText(OptionMenuDemoActivity.this, "声音已经打开！！",
  Toast.LENGTH_SHORT).show();
  break;
  }
  return super.onOptionsItemSelected(item);
}
```

选项菜单的创建很简单，用得也非常多，它的创建和对其进行操作只要覆盖两个方法：

onCreateOptionsMenu 和 onOptionsItemSelected。

5.5.2 上下文菜单

图 5-24 上下文菜单

长按界面，就会弹出一个菜单。这个菜单就称为上下文菜单。下面来看看 Android 中的上下文菜单，如图 5-24 所示，在界面的文字上长按时，会弹出一个上下文菜单，可以单击每一项对字体进行颜色设置。

下面来看看具体实现的代码。

（1）布局文件。这里的界面设计得很简单，详细代码不再具体给出。

（2）覆盖方法 onCreateContextMenu。在这个方法中可以添加相应的菜单项。

```
//覆盖方法，创建上下文菜单
@Override
Public  void  onCreateContextMenu(ContextMenu  menu, View v,
ContextMenuInfo menuInfo) {
super.onCreateContextMenu(menu, v, menuInfo);
//创建菜单项
menu.add(0, RED, 0, "红色");
menu.add(0, BLUE, 0, "蓝色");
menu.add(0, YELLOW, 0, "黄色");
}
```

（3）覆盖方法 onContextItemSelected。对每一个菜单项进行相应的处理，改变字体的颜色。

```
//覆盖方法，对每一个菜单项进行事件监听
@Override
public boolean onContextItemSelected(MenuItem item) {
  switch (item.getItemId()) {
  case RED:
  textView.setTextColor(Color.RED);
  break;
  case YELLOW:
  textView.setTextColor(Color.YELLOW);
  break;
  case BLUE:
  textView.setTextColor(Color.BLUE);
  break;
  default:
  break;
```