项目3

智能计算器

【教学导航】

学习目标	(1) 理解 Android 的 UI 的含义。 (2) 掌握 Android 的几种常见布局。 (3) 熟悉 Android 系统中的样式和主题。 (4) 培养简单案例的设计开发能力。
教学方法	任务驱动法、理论实践一体化、探究学习法、分组讨论法
课时建议	6 课时

3.1 Android UI 常用布局

一个丰富的界面总是要由很多控件组成的，如何才能让各个控件都有条不紊地摆放在界面上呢？这就需要借助布局来实现了。布局是一种可用于放置很多控件的容器，它可以按照一定的规律调整内部控件的位置，从而编写出精美的界面。当然，布局的内部除了放置控件外，也可以放置布局，通过多层布局的嵌套，就能够完成一些比较复杂的界面。本节详细讲解 Android 中最基本的 3 种布局：LinearLayout(线性布局)、RelativeLayout(相对布局)和 FrameLayout(帧布局)。


微课视频

3.1.1 UI 简介

UI 即 User Interface(用户界面)的简称，是人和设备之间交互的工具，小到手机端的应用，大到计算机上的桌面程序，用户所触摸到的屏幕就是用户界面。

1. UI 界面

用户界面并不是一个新鲜的概念，早在中国古代使用的算盘，它是由珠子组成的最早的人机交互界面。后来，MS DOS 操作系统走进了大众视野，但它呈现出来的是黑色屏幕和

白色代码的命令窗口，界面既不美观也不友好。后来图形化用户界面兴起，这种界面上有很多图标，计算机端用鼠标就可以进行简单的操作，手机端应用手指触摸便可响应。

UI 设计就是设计界面美观、舒适的人机交互界面，让软件开发变得有品位、有个性，让用户操作变得简单、舒适。

2. UI 设计相关的几个概念

在 Android 中，进行界面设计时，经常会用到 View、ViewGroup、Padding、Margins 等概念，对于初学 Android 的人来说，一般不好理解。下面对这几个概念进行详细的介绍。

1）View

View 在 Android 中可以理解为视图，它占据屏幕上的一块矩形区域，负责提供组件绘制和事件处理的方法，如图 3.1 所示。View 相当于窗户上的玻璃，设计窗户时，可以设计自己所需要的玻璃数，可以是 4 块玻璃，也可以是 6 块玻璃。也就是说，在手机屏幕中可以有很多个 View，在 Android 手机中，使用 View 是通过 View 类来实现的，View 类就是所有控件的基类，某一个控件如 TextView 是 View 的子类。

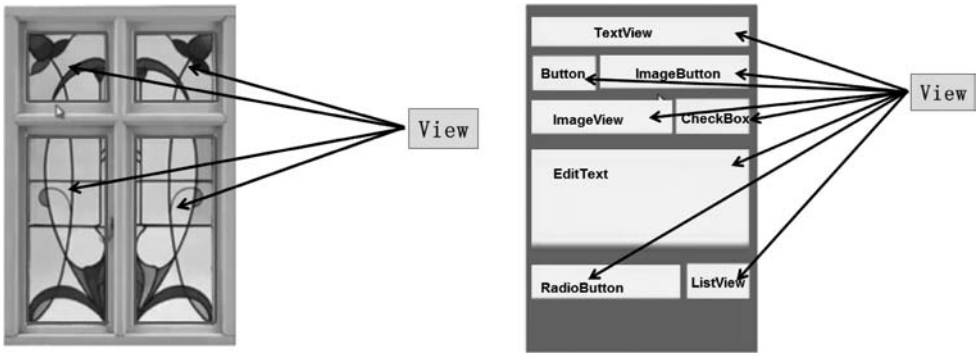

图 3.1 UI 界面上的 View

2）ViewGroup

ViewGroup 在 Android 中可以理解为容器。ViewGroup 类继承自 View 类，它是 View 类的扩展，是用来容纳其他组件的容器，但是由于 ViewGroup 是一个抽象类，所以在实际应用中通常总是使用 ViewGroup 的子类来作为容器的，如图 3.2 所示。

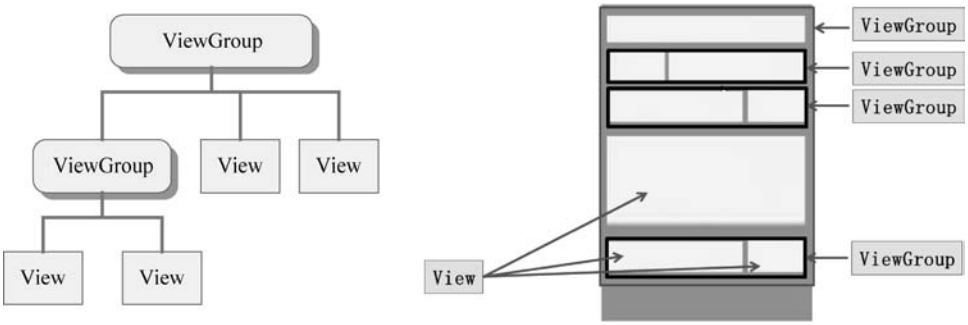

图 3.2 UI 界面上的 ViewGroup

3）Padding 和 Margins

Padding 表示在控件的顶部、底部、左侧和右侧的填充像素，也称为内边距。它设置的是控件内容到控件边缘的距离。Padding 将占据控件的宽度和高度。设置指定的内边距后，内容将偏离控件边缘指定的距离。

Margins 表示控件的顶部、底部、左侧和右侧的空白区域，称为外边距。它设置的是控件与其父容器的距离。Margins 不占据控件的宽度和高度。为控件设置外边距后，该控件将远离父容器指定的距离，如果还有相邻组件，那么也将远离其相邻组件指定距离。

内边距和外边距示意图如图 3.3 所示。

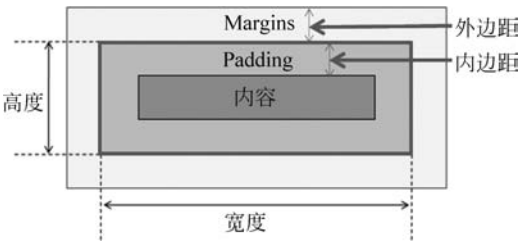

图 3.3　内边距和外边距

4）dp 和 sp

dp 是宽度、高度的单位。例如：按钮的宽度为 100dp，按钮的高度为 60dp。

sp 是文字大小的单位，例如：文字大小为 32sp 等。

3.1.2　LinearLayout 布局

LinearLayout 又称作线性布局，是一种最常用的布局，正如它的名字一样，这个布局会将它所包含的控件在线性方向上依次排列。

1. LinearLayout 常用属性

LinearLayout 属性较多，下面列举一些常用属性，如表 3.1 所示。

表 3.1　LinearLayout 常用属性

属　　性	功能描述	实　　例
android:orientation	指定控件排列方向	android:orientation="vertical"内部控件在垂直方向上排列
		android:orientation="horizontal"内部控件在水平方向上排列
android:layout_gravity	指定控件在布局中的对齐方式	android:layout_gravity="top" 顶端对齐(方向：水平) android:layout_gravity="center_vertical" 垂直居中 android:layout_gravity="bottom" 底端对齐
		android:layout_gravity="left" 左对齐(方向：垂直) android:layout_gravity="center_horizontal" 水平居中 android:layout_gravity="right "右对齐
android:layout_weight	允许控件使用比例方式指定控件的大小	android:layout_weight="1"

1）orientation 属性

orientation 属性可以用来控制线性布局中控件的排列方向。新建一个工程 LinearLayoutTest，打开 activity_main.xml 布局界面，输入代码如下。

```
<?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:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 1"          />
    < Button
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 2"          />
    < Button
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 3"          />
</LinearLayout >
```

上述代码中，界面布局采用 LinearLayout。在 LinearLayout 中添加了 3 个 Button，每个 Button 的高度和宽度都是 wrap_content，并通过 android:orientation 属性指定了排列方向是 vertical，现在运行程序，效果如图 3.4 所示。

然后修改 LinearLayout 的排列方向，将 android:orientation 属性的值修改为 horizontal，这就意味着要让 LinearLayout 中的控件在水平方向上排列。重新运行程序，效果如图 3.5 所示。

2）layout_gravity 属性

该属性与 android:gravity 属性看起来有些相似，它们的区别在于 android:gravity 用于指定文字在控件中的对齐方式，而 android:layout_gravity 用于指定控件在布局中的对齐方式，当 LinearLayout 的排列方向为 horizontal 时，只有垂直方向上的对齐方式才会生效，修改 activity_main.xml 中的代码如下。

```
<?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 = "horizontal"
    tools:context = ".MainActivity">
```

```
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮 1"
        android:layout_gravity="top"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮 2"
        android:layout_gravity="center_vertical"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="按钮 3"
        android:layout_gravity="bottom"/>
</LinearLayout>
```

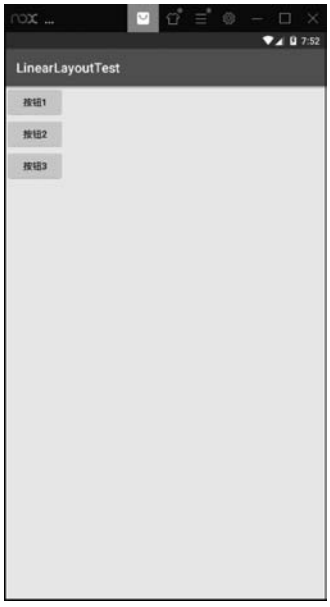

图 3.4　LinearLayout 垂直排列

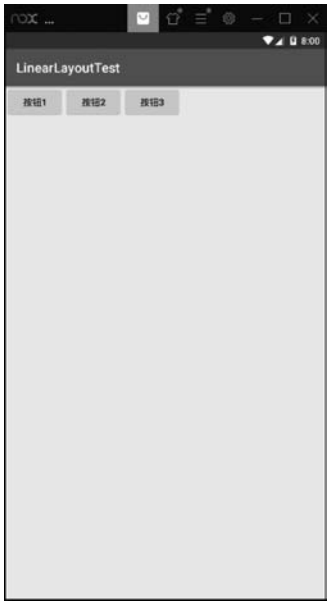

图 3.5　LinearLayout 水平排列

由于目前 LinearLayout 的排列方向是 horizontal，每添加一个控件，水平方向上的长度都会改变，因而无法指定控件左右对齐，只能指定垂直方向上的排列方向。将第一个按钮对齐方式指定为 top，第二个按钮对齐方式指定为 center_vertical，第三个按钮对齐方式指定为 bottom。重新运行程序，效果如图 3.6 所示。

这里需要注意，如果 LinearLayout 的排列方向是 horizontal，内部的控件就绝对不能将宽度指定为 match_parent，因为这样的话，单独一个控件就会将整个水平方向占满，其他的控件就没有可放置的位置了。同样的道理，如果 LinearLayout 的排列方向为 vertical，内部的控件就不能将高度指定为 match_parent。而且排列方向为 vertical 时，只有水平方向上的对齐方式才会生效，修改 activity_main.xml 中的代码如下。

```
<?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:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 1"
        android:layout_gravity = "left"/>
    < Button
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 2"
        android:layout_gravity = "center_horizontal"/>
    < Button
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 3"
        android:layout_gravity = "right"/>
</LinearLayout >
```

再次运行程序,效果如图 3.7 所示。

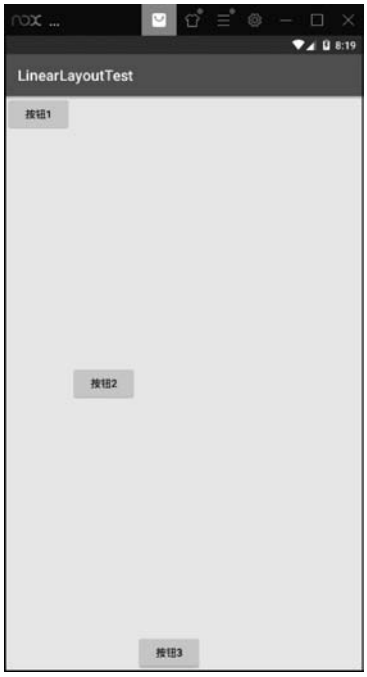

图 3.6 horizontal 方向时 layout_gravity

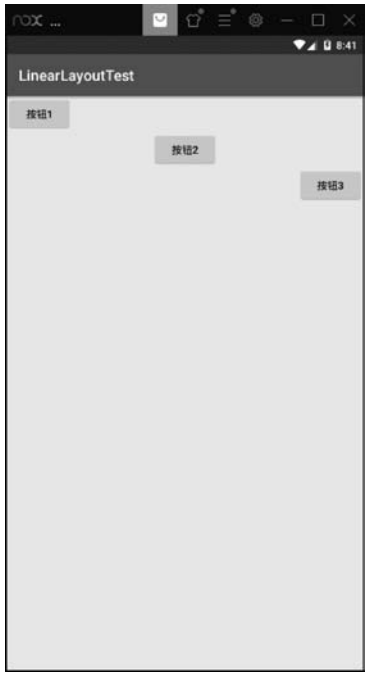

图 3.7 vertical 方向时 layout_gravity

3）layout_weight 属性

这个属性允许使用比例的方式指定控件的大小，它在手机适配性方面可以起到非常重要的作用。例如，编写一个消息发送界面，需要一个文本编辑框和一个“发送”按钮，修改 activity_main.xml 中的代码如下。

```
<?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 = "horizontal"
    tools:context = ".MainActivity">
    < EditText
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:hint = "请输入内容"
        android:layout_weight = "1"/>
     < Button
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "发送"
        android:layout_weight = "1"/>
</LinearLayout >
```

上述代码中，在 EditText 和 Button 里都将 android:layout_weight 属性指定为 1，表示 EditText 和 Button 将在水平方向上平分宽度，运行效果如图 3.8 所示。如果删除 Button 控件里的 android:layout_weight＝"1"这行代码，表示 EditText 会占满屏幕的所有剩余空间，效果如图 3.9 所示。

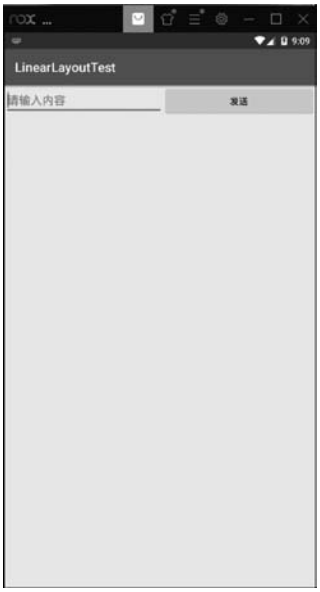

图 3.8　指定 layout_weight 平分宽度

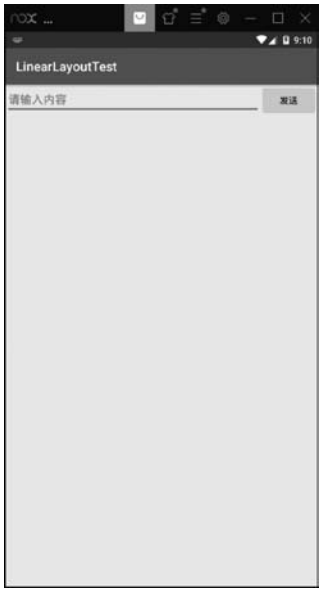

图 3.9　指定 layout_weight 效果

2. LinearLayout 布局案例

1）案例分析

（1）界面分析。本案例共有一个布局界面，自上而下的垂直方向，用户名、密码、按钮这三行是嵌套的水平线性布局，自左向右的水平方向，两个按钮控件权重为 1。

（2）设计思路。布局界面中通过 android:orientation 属性设定线性布局的方向达到用户需求，通过线性布局的嵌套实现用户名、密码和按钮这三行控件的添加，通过 layout_weight 属性设定控件所占的宽度比例，达到用户的需求。

2）实现步骤

（1）创建一个新的工程，工程名为 ZSLinearLayout。

（2）切换工程的 Project 项目结构。选择该模式下方的 app，依次展开，便看到工程的布局界面和工程的类文件，其中，activity_main.xml 是布局界面，MainActivity.java 为类文件。

（3）准备一张图片，图片名为 background.png，将其粘贴到 app 目录结构中 res 下方的 drawable 文件夹下，这张图片作为页面的背景图片。

（4）设计布局界面。双击 layout 文件夹下的 activity_main.xml 文件，便可打开布局编辑器，修改布局类型为 LinearLayout，添加方向属性为 vertical。代码如下。

```
<?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"
    android:background = "@drawable/background"
    tools:context = ".MainActivity">
    < TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "用户登录"
        android:textSize = "40sp"
        android:textColor = "#000000"
        android:textStyle = "bold"
        android:layout_marginTop = "100dp"
        android:layout_marginBottom = "100dp"
        android:layout_gravity = "center"/>
    < LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal"
        android:layout_marginTop = "20dp"
        android:layout_marginLeft = "10dp"
        android:layout_marginRight = "10dp"
```

```
        android:gravity = "center">
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "用户名:"
            android:textSize = "30sp"
            android:textColor = "#000000" />
        < EditText
            android:id = "@ + id/edit_inputname"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:hint = "请输入用户名"
            android:textSize = "25sp" />
    </LinearLayout >
    < LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal"
        android:layout_marginTop = "20dp"
        android:layout_marginLeft = "10dp"
        android:layout_marginRight = "10dp"
        android:gravity = "center">
        < TextView
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "密      码:"
            android:textSize = "30sp"
            android:textColor = "#000000"     />
        < EditText
            android:id = "@ + id/edit_inputpwd"
            android:layout_width = "match_parent"
            android:layout_height = "wrap_content"
            android:hint = "请输入密码"
            android:inputType = "textPassword"
            android:textSize = "25sp" />
    </LinearLayout >
    < CheckBox
        android:id = "@ + id/check_remeber"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "记住密码"
        android:textStyle = "bold"
        android:textSize = "20sp"
        android:layout_gravity = "right"
        android:layout_margin = "10dp"/>

    < LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
```

```
            android:orientation = "horizontal"
            android:layout_marginTop = "20dp"
            android:layout_marginLeft = "10dp"
            android:layout_marginRight = "10dp"
            android:gravity = "center">
            < Button
                android:id = "@ + id/button_no"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "取消"
                android:textSize = "30sp"
                android:textColor = "#050505"
                android:layout_weight = "1"/>
            < Button
                android:id = "@ + id/button_yes"
                android:layout_width = "wrap_content"
                android:layout_height = "wrap_content"
                android:text = "登录"
                android:textSize = "30sp"
                android:textColor = "#050505"
                android:layout_weight = "1"     />
        </LinearLayout >
    </LinearLayout >
```

从上述代码中可以看出,布局的内部除了放置控件外,也可以放置布局,通过多层布局的嵌套,完成了比较复杂的界面。最外层布局界面是 LinearLayout 线性布局,使用 android:orientation 属性指定布局的方向,属性值为 vertical 表示布局内的控件自上而下垂直排列。用户名所在行的两个控件是水平排列的,为了实现效果,采用了嵌套线性布局,将 android:orientation 属性取值 horizontal 表示自左向右的水平方向。将显示用户名的 TextView 控件和输入用户名的 EditText 控件放置于嵌套的线性布局的内部。下方又是一个水平方向的线性布局,里面放置了密码行的两个控件。密码行下方又是一个水平方向的线性布局,两个按钮控件放到布局的内部,为了让两个按钮控件宽度相等,添加了 layout_weight 属性,属性值为 1 表示等分宽度。运行程序,效果如图 3.10 所示。

3.1.3 RelativeLayout 布局

RelativeLayout 又称作相对布局,也是一种常用的布局,该布局和 LinearLayout 的排列规则不同,RelativeLayout 显得更加随意一些,它可以通过相对定位的方式让控件出现在布局中的任何位置。也正因为如此,RelativeLayout 中的属性非常多。

1. RelativeLayout 常用属性

RelativeLayout 的属性非常多,不过这些属性都是有规律可循的,并不难理解和记忆。控件相对于父容器进行定位时。常用属性如表 3.2 所示。

图 3.10　程序运行效果

表 3.2　RelativeLayout 常用属性——控件相对于父容器进行定位

属　　性	功能描述	实　　例
android:layout_alignParentTop	其属性值为 boolean 值，用于指定控件是否与布局管理器顶端对齐	android: layout _ alignParentTop = "true"
android:layout_alignParentBottom	其属性值为 boolean 值，用于指定控件是否与布局管理器底端对齐	android: layout _ alignParentBottom = "true"
android:layout_alignParentLeft	其属性值为 boolean 值，用于指定控件是否与布局管理器左边对齐	android: layout _ alignParentLeft = "true"
android:layout_alignParentRight	其属性值为 boolean 值，用于指定控件是否与布局管理器右边对齐	android:layout _ alignParentRight = "true"
android:layout_centerHorizontal	其属性值为 boolean 值，用于指定控件是否位于布局管理器水平居中的位置	android: layout _ centerVertical = "true"
android:layout_centerVertical	其属性值为 boolean 值，用于指定控件是否位于布局管理器垂直居中的位置	android: layout _ centerHorizontal = "true"
android:layout_centerInParent	其属性值为 boolean 值，用于指定控件是否位于布局管理器的中央位置	android: layout _ centerInParent = "true"

相对布局定位有两种方式，第一种是控件相对于父容器进行定位，第二种是控件相对于控件进行定位。表 3.2 中的属性值是第一种定位方式常用的几个属性，当定位方式为控件相对于控件进行定位时，常用属性如表 3.3 所示。

表 3.3 RelativeLayout 常用属性——控件相对于控件进行定位

属　性	功能描述	实　例
android:layout_above	其属性值为其他 UI 组件的 ID 属性，用于指定该组件位于哪个组件的上方	android:layout_above="@id/button_5"
android:layout_below	其属性值为其他 UI 组件的 ID 属性，用于指定该组件位于哪个组件的下方	android:layout_below="@id/button_5"
android:layout_toLeftOf	其属性值为其他 UI 组件的 ID 属性，用于指定该组件位于哪个组件的左侧	android:layout_toLeftOf="@id/button_5
android:layout_toRightOf	其属性值为其他 UI 组件的 ID 属性，用于指定该组件位于哪个组件的右侧	android:layout_toRightOf="@id/button_5"
android:layout_alignTop	其属性值为其他 UI 组件的 ID 属性，用于指定该组件与哪个组件的上边界对齐	android:layout_alignTop="@id/button_5"
android:layout_alignBottom	其属性值为其他 UI 组件的 ID 属性，用于指定该组件与哪个组件的下边界对齐	android:layout_alignBottom="@id/button_5"
android:layout_alignLeft	其属性值为其他 UI 组件的 ID 属性，用于指定该组件与哪个组件的左边界对齐	android:layout_alignLeft="@id/button_5"
android:layout_alignRight	其属性值为其他 UI 组件的 ID 属性，用于指定该组件与哪个组件的右边界对齐	android:layout_alignRight="@id/button_5"

2. RelativeLayout 布局案例

1）案例分析

（1）界面分析。本工程中一共有两个布局界面，第一个布局界面中有 5 个 Button 控件，其中 4 个按钮位于布局的 4 个角，第 5 个按钮位于布局的正中心。第二个布局界面中有 1 个图片和 7 个 Button 控件，其中一个按钮位于布局的正中心，4 个按钮位于中心按钮的上下左右，另外两个按钮位于中心按钮的左下角和右上角处。

（2）设计思路。布局界面一通过控件相对于父容器进行定位时的 RelativeLayout 常用属性达到用户需求，布局界面二通过控件相对于控件进行定位时的属性达到用户的需求。

2）实现步骤

（1）创建一个新的工程，工程名为 ZSRelativeLayout。

（2）切换工程的 Project 项目结构。选择该模式下方的 app，依次展开，便看到工程的布局界面和工程的类文件，其中，activity_main.xml 是布局界面，MainActivity.java 为类

文件。

(3) 在工程中添加一个新的页面。右击 com.example.zsintent 包→New→Activity→Empty Activity,会弹出一个创建活动的对话框,将活动命名为 SecondActivity,默认勾选 Generate Layout File 关联布局界面,布局界面名称为 activity_second 但不要勾选 Launcher Activity。单击 Finish 按钮,便可在工程中完成第二个页面的添加。

(4) 准备 6 张图片,将其粘贴到 app 目录结构中 res 下方的 drawable 文件夹下。第二个布局界面中,playbg.png 是页面上方的一张游戏图片,start.jpg 图片位于布局界面的中心位置,up.jpg、down.jpg、left.jpg、rigth.jpg 是周围的四张图片。

(5) 设计第一个 Activity 的布局界面。双击 layout 文件夹下的 activity_main.xml 文件,便可打开布局编辑器,修改布局类型为 RelativeLayout,代码如下。

```
<?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">
    < Button
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 1"
        android:textSize = "40sp"
        android:layout_alignParentTop = "true"
        android:layout_alignParentLeft = "true" />
    < Button
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 2"
        android:textSize = "40sp"
        android:layout_alignParentBottom = "true"
        android:layout_alignParentLeft = "true" />
    < Button
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 3"
        android:textSize = "40sp"
        android:layout_alignParentTop = "true"
        android:layout_alignParentRight = "true"/>
    < Button
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 4"
        android:textSize = "40sp"
        android:layout_alignParentBottom = "true"
        android:layout_alignParentRight = "true"/>
    < Button
```

```
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "按钮 5"
        android:textSize = "40sp"
        android:layout_centerInParent = "true"/>
</RelativeLayout >
```

上述代码中，Button1 和父布局的左上角对齐，Button2 和父布局的左下角对齐，Button3 和父布局的右上角对齐，Button4 和父布局的右下角对齐，Button5 位于父布局中心位置。运行程序，效果如图 3.11 所示。其中每个控件都是相对于父布局进行定位的，控件还可以相对于控件进行定位。

图 3.11 相对于父布局定位的效果

（6）设计第二个 Activity 的布局界面。双击 layout 文件夹下的 activity_second.xml 文件，便可打开布局编辑器，修改布局类型为 RelativeLayout，依次添加所需控件，代码如下。

```
<?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 = ".SecondActivity">
    < ImageView
        android:layout_width = "wrap_content"
        android:layout_height = "210dp"
        android:src = "@drawable/background"          />
```

```
        <Button
            android:id="@+id/button_5"
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:layout_centerInParent="true"
            android:background="@drawable/start" />
        <Button
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:background="@drawable/up"
            android:layout_above="@+id/button_5"
            android:layout_alignLeft="@+id/button_5"  />
        <Button
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:background="@drawable/down"
            android:layout_below="@id/button_5"
            android:layout_alignLeft="@id/button_5"         />
        <Button
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:background="@drawable/left"
            android:layout_toLeftOf="@id/button_5"
            android:layout_alignTop="@id/button_5"  />
        <Button
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:background="@drawable/right"
            android:layout_toRightOf="@id/button_5"
            android:layout_alignTop="@id/button_5"  />
        <Button
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:background="#ff0000"
            android:layout_toRightOf="@id/button_5"
            android:layout_above="@id/button_5"  />
        <Button
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:background="#0000ff"
            android:layout_toLeftOf="@id/button_5"
            android:layout_below="@id/button_5"  />
    </RelativeLayout>
```

这个代码稍微复杂一点儿，不过还是有规律可循的。android:layout_above 属性可以让一个控件位于另一个控件的上方，需要为这个属性指定相对控件 ID 的引用，这里填入了@id/button_5，表示让该控件位于 ID 号为 5 的按钮的上方。其他属性也是类似的。android:layout_below 表示让一个控件位于另一个控件的下方，android:layout_toLeftOf 表示让一个控件位于另一个控件的左侧，android:layout_toRightOf 表示让一个控件位于另

一个控件的右侧。android:layout_alignTop 表示让一个控件与另一个控件顶部对齐。注意,当一个控件去引用另一个控件的 ID 时,该控件一定要定义在引用控件的后面,否则会出现找不到 ID 的情况。

(7) 修改页面的启动顺序。依次展开 app 项目结构,双击下方的 Android 配置文件 AndroidManifest.xml,配置文件代码中,< activity android:name=".MainActivity">是第一个页面的开始节点,下方的</activity>是第一个页面的结束节点,中间的四行代码就是决定页面启动顺序的关键代码,将这四行代码剪切、粘贴到第二个页面开始节点< activity android:name=".SecondActivity">与结束节点</activity>的中间位置。修改后代码如下。

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".SecondActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".MainActivity">
    </activity>
</application>
```

(8) 重新运行程序,效果如图 3.12 所示。

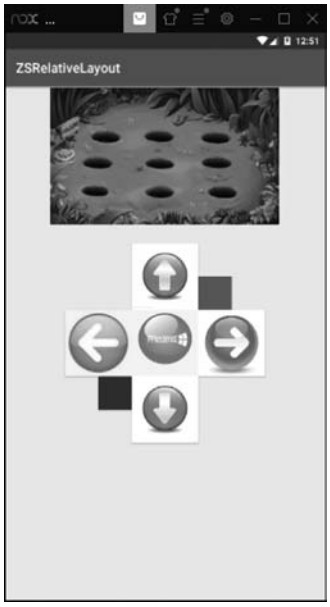

图 3.12 相对于控件定位的效果


微课视频

3.1.4 FrameLayout 布局

FrameLayout 又称为帧布局，是 Android 中最为简单的一种布局。采用帧布局设计界面时，所有控件都默认显示在屏幕的左上角，并按照先后放入的顺序重叠摆放，先放入的控件显示在最底层，后放入的控件显示在最顶层。

1. FrameLayout 案例

1）案例分析

（1）界面分析。布局界面中总体是 LinearLayout，自上而下的垂直方向，界面上方嵌套一个 FrameLayout 帧布局，头像和名字在帧布局的内部，下方的几个控件线性排列。

（2）设计思路。整个布局界面可以通过布局嵌套达到用户需求，上方嵌套的帧布局中，人物图片和用户名默认在左上角重叠摆放，通过 android:layout_gravity 属性设置了对齐方式，通过 android:layout_margin 属性设定了控件周围的空白距离，达到用户的需求。

2）实现步骤

（1）创建一个新的工程，工程名为 ZSFrameLayout。

（2）切换工程的 Project 项目结构。选择该模式下方的 app，依次展开，便看到工程的布局界面和工程的类文件，其中，activity_main.xml 是布局界面，MainActivity.java 为类文件。

（3）修改布局界面。双击 layout 文件夹下的 activity_main.xml 文件，便可打开布局编辑器，修改布局类型为 LinearLayout，添加方向属性为垂直。依次添加嵌套的 FrameLayout 帧布局和其他控件，代码如下。

```
<?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">
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/myinfo_login_bg"       >
        <ImageView
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:src="@drawable/default_icon"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="50dp"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="admin"
```

```
            android:textSize = "30sp"
            android:textColor = "#ffffff"
            android:layout_gravity = "center_horizontal"
            android:layout_marginTop = "130dp"/>
    </FrameLayout>
    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "个人中心"
        android:textSize = "30sp"
        android:textStyle = "bold"
        android:layout_gravity = "center"
        android:layout_marginBottom = "20dp"/>
    <View
        android:layout_width = "match_parent"
        android:layout_height = "2dp"
        android:background = "#E91A1A"/>
    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "设置"
        android:textSize = "25sp"
        android:padding = "10dp"          />
    <View
        android:layout_width = "match_parent"
        android:layout_height = "1dp"
        android:background = "#C5C2C2"/>
    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "修改密码"
        android:textSize = "25sp"
        android:padding = "10dp"          />
    <View
        android:layout_width = "match_parent"
        android:layout_height = "1dp"
        android:background = "#C5C2C2"/>
    <TextView
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android:text = "作品详情"
        android:textSize = "25sp"
        android:padding = "10dp"          />
    <View
        android:layout_width = "match_parent"
        android:layout_height = "1dp"
        android:background = "#C5C2C2"/>
    <TextView
        android:layout_width = "wrap_content"
```

```
            android:layout_height = "wrap_content"
            android:text = "注销"
            android:textSize = "25sp"
            android:padding = "10dp"          />
        < View
            android:layout_width = "match_parent"
            android:layout_height = "1dp"
            android:background = " # C5C2C2" />
    </LinearLayout >
```

上述代码中,线性布局内嵌套了帧布局。在帧布局中,使用 android:layout_gravity 属性让图片水平居中,使用 android:layout_marginTop 确定了用户头像图片的位置。采用同样的方法显示用户名,从而实现了多个控件重叠摆放的问题。另外,画线时使用的控件为 View,该控件的高度用来决定画线的粗细,同时使用 android:background 属性设置画线的颜色。

(4) 重新运行程序,效果如图 3.13 所示。

图 3.13 FrameLayout 布局效果图

3.2 Android 开发中的样式设计

Android 系统中包含很多样式(Style),这些样式用于定义界面上的布局风格,Style 是针对窗体元素级别的,改变指定控件或者 Layout 的样式。例如,TextView 或 Button 等控件,通过样式的设定,让布局合理而且美观,提升用户体验。


微课视频

3.2.1 自定义控件样式

样式是包含一种或多种控件的属性集合，可以指定控件的高度、宽度、字体大小及颜色等。Android 中的样式类似于网页中的 CSS 样式，可以让设计与内容分离。样式文件在 XML 资源文件中定义，并且可以继承、复用等，方便统一管理并减少代码量。

(1) 新建一个工程，工程名为 StyleTest。

(2) 切换工程的 Project 项目结构。选择该模式下方的 app，依次展开，便看到工程的布局界面和工程的类文件，其中，activity_main.xml 是布局界面，MainActivity.java 为类文件。

(3) 创建样式。在 app 目录结构下找到 res/values/style 目录下的 styles.xml 文件，双击打开，看到< resource >根标签和定义样式的< style >标签，它包含多个< item >来声明样式名称和属性，在其中编写两种 TextView 控件样式，代码如下。

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
    <style name="textStyle_one">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#ff00ac</item>
        <item name="android:textSize">35sp</item>
    </style>
    <style name="textStyle_two" parent="@style/textStyle_one">
        <item name="android:textSize">25sp</item>
    </style>
</resources>
```

上述代码中，第 1 个< style >标签中的代码是系统自带的样式，其中，name 属性是样式名称，parent 属性表示继承某个样式，并且通过< item >标签以键值对的形式定义属性和属性值。textStyle_one 是自定义的样式，设置了控件的高、宽、字体颜色、字体大小四个属性。textStyle_two 样式继承了 textStyle_one，并在该属性中重新定义了 android:textSize 的属性。

(4) 设计布局界面。双击 layout 文件夹下的 activity_main.xml 文件，便可打开布局编辑器，修改布局类型为 LinearLayout，添加方向属性为垂直。依次添加两个 TextView 控件，代码如下。

```
<?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">
    < TextView
        style = "@style/textStyle_one"
        android:text = "TextView 样式一"          />
    < TextView
        style = "@style/textStyle_two"
        android:text = "TextView 样式二"          />
</LinearLayout >
```

上述代码中，两个 TextView 只需要以 style="@style/xxx"这种方式就可以引用自定义样式中的所有属性，其中属性值对应自定义样式名称，运行程序，预览效果如图 3.14 所示。

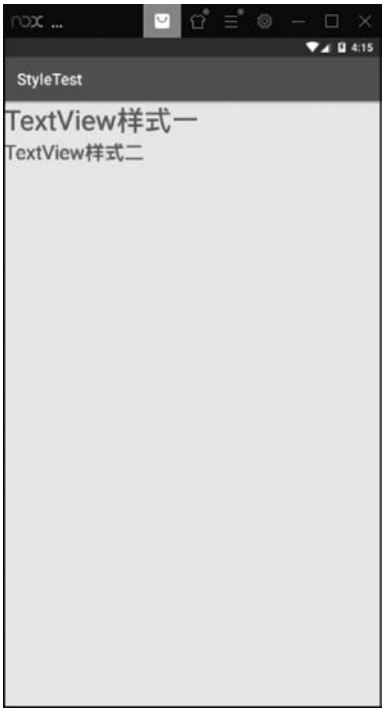

图 3.14　TextView 样式


微课视频

3.2.2　自定义背景样式

在设计开发项目时，也可以自己创建样式文件，通过定义 XML 文件来实现不同的样式。下面以定义一个具有边框色和填充色的输入框 EditText 控件的样式为例，继续在 StyleTest 工程中完善内容。

(1) 边框和填充样式文件创建。展开自定义样式1创建的工程文件StyleTest,在app目录结构下找到res/drawable,右击drawable文件夹,选择New→Drawable resource file,输入样式名称和节点类型,单击OK按钮,如图3.15所示。

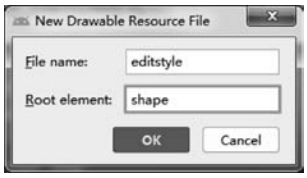

图3.15 样式文件定义

(2) 在editstyle.xml样式文件中输入如下代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< shape xmlns:android = "http://schemas.android.com/apk/res/android">
    < stroke android:width = "3dp" android:color = " # ff0000"/>
    < solid android:color = " # FFEB3B"/>
</shape >
```

上述代码中,stroke节点用来定义边框,使用android:width定义边框宽度,使用android:color定义边框颜色。solid节点用来定义填充,android:color表示填充的颜色。

(3) 设计布局界面。双击layout文件夹下的activity_main.xml文件,便可打开布局编辑器,添加EditText控件,引用定义的样式文件,代码如下。

```
<?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">
    < TextView
        style = "@style/textStyle_one"
        android:text = "TextView 样式一"          />
    < TextView
        style = "@style/textStyle_two"
        android:text = "TextView 样式二"          />
    < EditText
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:hint = "请输入用户名"
        android:textSize = "40sp"
        android:background = "@drawable/editstyle"
        android:layout_margin = "10dp"/>
</LinearLayout >
```

上述代码中,使用android:background给EditText控件设定背景,其值为@drawable/editstyle表示引用drawable文件夹下的样式文件。运行程序,预览效果如图3.16所示。

按钮具有弹起和按下两种状态,使用背景选择器可以让按钮在正常情况下显示为紫色填充和黑色边框,按下时显示为蓝色填充和红色边框。继续在StyleTest工程中完善内容。

图 3.16　EditText 控件样式

（1）创建按钮正常情况下的样式。在 app 目录结构下找到 res/drawable，右击 drawable 文件夹，选择 New→Drawable resource file，输入样式名称和节点类型，单击 OK 按钮，如图 3.17 所示。

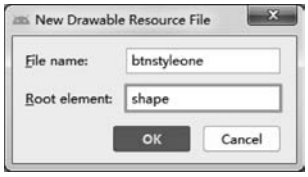

图 3.17　样式文件定义

（2）在 btnstyleone.xml 样式文件中输入如下代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< shape xmlns:android = "http://schemas.android.com/apk/res/android">
    < stroke android:width = "8dp" android:color = " # 090808"/>
    < solid android:color = " # C70AF0"/>
    < corners android:radius = "50dp"/>
</shape >
```

上述代码中，stroke 节点用来定义边框，solid 节点用来定义填充，corners 节点用来定义圆角半径。

（3）创建按钮按下时的样式。方法同上，新建样式名称为 btnstyletwo.xml，节点类型为 shape，在样式文件 btnstyletwo.xml 中输入如下代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< shape xmlns:android = "http://schemas.android.com/apk/res/android">
    < stroke android:width = "8dp" android:color = " # E91E4D"/>
    < solid android:color = " # 172CD5"/>
    < corners android:radius = "50dp"/>
</shape >
```

(4) 创建按钮选择器样式文件。在 app 目录结构下找到 res/drawable，右击 drawable 文件夹，选择 New→Drawable resource file，输入选择器名称和节点类型，单击 OK 按钮，如图 3.18 所示。

图 3.18 按钮选择器样式

(5) 在 btnselector.xml 选择器样式文件中输入如下代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< selector xmlns:android = "http://schemas.android.com/apk/res/android">
    < item android:drawable = "@drawable/btnstyleone" android:state_pressed = "false"/>
    < item android:drawable = "@drawable/btnstyletwo" android:state_pressed = "true"/>
</selector >
```

上述代码中，selector 标签可以添加一个或多个 item 子标签，而相应的状态是在 item 标签中定义的。item 必须指定 android:drawable 属性，drawable 的属性值可以是一张图片，也可以是一个样式文件。android:state_pressed 设置是否按压状态，默认为 false 表示未单击按钮，属性值为 true 时表示单击按钮。

第一行 item 表示按钮正常情况下默认时的背景图片，第二行 item 表示按钮单击时的背景图片。

(6) 设计布局界面。双击 layout 文件夹下的 activity_main.xml 文件，便可打开布局编辑器，添加 Button 控件，引用定义的按钮选择器样式文件，代码如下。

```
<?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">
    < TextView
        style = "@style/textStyle_one"
        android:text = "TextView 样式一" />
    < TextView
        style = "@style/textStyle_two"
        android:text = "TextView 样式二" />
    < EditText
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:hint = "请输入用户名"
        android:textSize = "40sp"
        android:background = "@drawable/editstyle"
        android:layout_margin = "10dp"/>
    < Button
        android:layout_width = "100dp"
        android:layout_height = "100dp"
```

```
            android:text = "美"
            android:textSize = "50sp"
            android:textColor = "#ffffff"
            android:padding = "5dp"
            android:background = "@drawable/btnselector" />
    </LinearLayout >
```

上述代码中，使用 android:background 给 Button 控件设定背景，其值为@drawable/btnselector 表示引用 drawable 文件夹下按钮选择器的样式文件。运行程序，预览效果如图 3.19 所示。

图 3.19 Button 控件选择器样式

3.3 Android 开发中的主题设计

主题(Theme)是应用到整个 Activity 和 Application 的样式，而不是只应用到单个视图，当设置好主题后，Activity 或整个程序中的视图都将使用主题中的属性。当主题和样式中的属性发生冲突时，样式的优先级要高于主题。

主题与样式在代码结构上是一样的，不同之处在于引用方式上，主题要在 AndroidManifest.xml 配置文件中引用，下面通过一个实例说明。

(1) 新建一个工程，工程名为 ThemeTest。

(2) 切换工程的 Project 项目结构。选择该模式下方的 app，依次展开，便看到工程的布局界面和工程的类文件，其中，activity_main.xml 是布局界面，MainActivity.java 为类

文件。

(3) 创建主题样式。在 app 目录结构下找到 res/values/style 目录下 styles.xml 文件，双击打开，在其内部编写主题样式代码，具体代码如下。

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
    <style name="grayTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="android:background">#999999</item>
    </style>
</resources>
```

上述代码中，定义了一个灰色的主题背景。需要注意的是，在定义主题时，需要用到 parent 属性去继承 Theme.AppCompat.Light.DarkActionBar 来保证它的兼容性，否则运行时会出现异常。接下来打开 AndroidManifest.xml 文件引用这个主题，在<activity>标签中添加 android:theme="@style/grayTheme"属性，具体代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.themetest">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity"
            android:theme="@style/grayTheme">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

在上述代码中，大家会发现在<application>标签中同样存在 android:theme 属性，此处是整个应用程序主题的样式，而<activity>标签中是改变当前界面的主题样式，这里要注意区分清楚。运行效果如图 3.20 所示。

图 3.20　主题样式运行结果


微课视频

3.4　实战案例——智能计算器

智能计算器开发主要涉及 LinearLayout、Style、EditText、Button 的使用，为 Android 入门基础内容。本节课将利用项目 3 所学过的知识，引导学生设计并实现智能计算器，系统具有良好的界面、必要的交互信息、简约的美观效果，使用人员能够快捷简单地进行操作，及时准确地获取需要的计算结果，充分降低了数字计算的难度，节约了时间成本。

图 3.21　智能计算器界面效果图

3.4.1　界面分析

本案例中共有一个界面，布局总体为 LinearLayout，自上而下的垂直方向 vertical，通过线性布局的嵌套，实现水平方向按钮的摆放，通过自定义 Style 样式文件，让 EditText 和 Button 呈现较美的外观，界面运行效果如图 3.21 所示。

3.4.2　实现思路

上方的 EditText 控件用来显示参与运算的数据，同时也显示运算的结果。单击数字按钮接受 0～9 组合起来的数字显示到上方的 EditText 控件中，单击加、减、

乘、除运算符按钮时标志着第一个运算数输入结束。此时获取 EditText 中的数值便得到了参与运算的第一个数。继续单击数字按钮 0～9 组合起来的数字，数字依然会显示到上方的 EditText 控件中，当单击等号时意味着第二个运算数输入结束，此时获取 EditText 中的数值便得到了参与运算的第二个数。根据单击的运算符号，对获得的两个运算数进行运算，便可得到结果。最后将运算结果显示到 EditText 编辑框中。

3.4.3 任务实施

【任务 3-1】 自定义 style 样式文件

(1) 创建一个新的工程，工程名为 ZSCalculator。

(2) 切换工程的 Project 项目结构。选择该模式下方的 app，依次展开，便看到工程的布局界面和工程的类文件，其中，activity_main.xml 是布局界面，MainActivity.java 为类文件。

(3) 准备一张图片，图片名为 calculatorbg.jpg，将其粘贴到 app 目录结构中 res 下方的 drawable 文件夹下，图片作为页面布局的背景。

(4) EditText 样式文件创建。在 app 目录结构下找到 res/drawable，右击 drawable 文件夹，选择 New→Drawable resource file，输入样式名称 editstyle1 和节点类型 shape，单击 OK 按钮，在 editstyle1.xml 样式文件中输入如下代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< shape xmlns:android = "http://schemas.android.com/apk/res/android">
    < solid android:color = " # CDDC39"/>
    < stroke android:color = " # D6C9C9" android:width = "3dp"/>
</shape >
```

(5) Button 按钮未单击时样式文件创建。在 app 目录下找到 res/drawable，右击 drawable 文件夹，选择 New→Drawable resource file，输入样式名称 btnstyle3 和节点类型 shape，输入如下代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< shape xmlns:android = "http://schemas.android.com/apk/res/android">
    < solid android:color = " # D024FF"/>
    < stroke android:color = " # 000000" android:width = "5dp"/>
    < corners android:radius = "1000dp"/>
</shape >
```

(6) Button 按钮被单击时样式文件创建。输入样式名称 btnstyle4 和节点类型 shape，输入如下代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< shape xmlns:android = "http://schemas.android.com/apk/res/android">
    < solid android:color = " # FF24AF"/>
    < stroke android:color = " # 28B474" android:width = "5dp"/>
    < corners android:radius = "1000dp"/>
</shape >
```

（7）Button 按钮选择器样式文件创建。输入选择器名称 btn34 和节点类型 selector，输入如下代码。

```
<?xml version = "1.0" encoding = "utf - 8"?>
< selector xmlns:android = "http://schemas.android.com/apk/res/android">
< item android:drawable = "@drawable/btnstyle3" android:state_pressed = "false"/>
< item android:drawable = "@drawable/btnstyle4" android:state_pressed = "true"/>
</selector >
```

【任务 3-2】 计算器 UI 界面设计

设计布局界面。双击 layout 文件夹下的 activity_main.xml 文件，便可打开布局编辑器，修改布局界面类型为 LinearLayout，方向为 vertical，依次添加所需控件，并引用刚刚创建的样式文件，代码如下。

```
<?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"
    android:background = "@drawable/calculatorbg"
    tools:context = ".MainActivity">
    < EditText
        android:id = "@ + id/txt_result"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:hint = "0"
        android:gravity = "right"
        android:textSize = "40sp"
        android:textColor = "#000000"
        android:padding = "10dp"
        android:layout_margin = "10dp"
        android:background = "@drawable/editstyle1" />
    < LinearLayout
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content"
        android:orientation = "horizontal"
        android:weightSum = "4">
        < Button
            android:id = "@ + id/btn_7"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "7"
            android:textSize = "60sp"
            android:textColor = "#ffffff"
            android:textStyle = "bold"
            android:background = "@drawable/btn34"
```

```
            android:padding = "5dp"
            android:layout_margin = "10dp"
            android:layout_weight = "1" />
        < Button
            android:id = "@ + id/btn_8"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "8"
            android:textSize = "60sp"
            android:textColor = " #ffffff"
            android:textStyle = "bold"
            android:background = "@drawable/btn34"
            android:padding = "5dp"
            android:layout_margin = "10dp"
            android:layout_weight = "1" />
        < Button
            android:id = "@ + id/btn_9"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "9"
            android:textSize = "60sp"
            android:textColor = " #ffffff"
            android:textStyle = "bold"
            android:background = "@drawable/btn34"
            android:padding = "5dp"
            android:layout_margin = "10dp"
            android:layout_weight = "1" />
        < Button
            android:id = "@ + id/btn_jia"
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = " + "
            android:textSize = "60sp"
            android:textColor = " #ffffff"
            android:textStyle = "bold"
            android:background = "@drawable/btn34"
            android:padding = "5dp"
            android:layout_margin = "10dp"
            android:layout_weight = "1" />
    </LinearLayout >

    ...

</LinearLayout >
```

上述代码中，第一行 EditText 控件用来存放输入的运算数及运算结果。下方依次是四个线性布局，按自左向右的水平方向放置。

第一行水平线性布局放置数字 7、8、9、+，其 ID 分别为 btn_7、btn_8、btn_9、btn_jia；代码如上，因下方三个水平线性布局与第一行线性布局代码雷同，请自行复制将代码补充

完整。

第二行水平线性布局放置数字 4、5、6、－，其 ID 分别为 btn_4、btn_5、btn_6、btn_jian。

第三行水平线性布局放置数字 1、2、3、＊，其 ID 分别为 btn_1、btn_2、btn_3、btn_cheng。

最后一行水平线性布局放置数字 0、C、＝、/，其 ID 分别为 btn_0、btn_qing、btn_deng、btn_chu。

【任务 3-3】 计算器功能设计

实现 Activity 页面的功能。打开 MainActivity.java 类文件，修改 MainActivity 的代码，如下。

```
public class MainActivity extends AppCompatActivity {
    //定义对象
  private TextView txt_result;
    private Button btn_7;
    private Button btn_8;
    private Button btn_9;
    private Button btn_jia;
    private Button btn_4;
    private Button btn_5;
    private Button btn_6;
    private Button btn_jian;
    private Button btn_1;
    private Button btn_2;
    private Button btn_3;
    private Button btn_cheng;
    private Button btn_0;
    private Button btn_qing;
    private Button btn_deng;
    private Button btn_chu;
    private double num1 = 0,num2 = 0;            //声明两个参数,接收数据
    private double result = 0;                   //运算结果
    private Boolean isClickdeng = false;         //判断是否单击了 =
    private String op = " % ";                   //操作符 + - * /

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main );
//绑定控件
        txt_result = findViewById(R.id.txt_result );
        btn_7 = findViewById(R.id.btn_7);
        btn_8 = findViewById(R.id.btn_8);
        btn_9 = (Button) findViewById(R.id.btn_9);
        btn_jia = (Button) findViewById(R.id.btn_jia );
        btn_4 = (Button) findViewById(R.id.btn_4);
        btn_5 = (Button) findViewById(R.id.btn_5);
        btn_6 = (Button) findViewById(R.id.btn_6);
```

```
        btn_jian = (Button) findViewById(R.id.btn_jian );
        btn_1 = (Button) findViewById(R.id.btn_1);
        btn_2 = (Button) findViewById(R.id.btn_2);
        btn_3 = (Button) findViewById(R.id.btn_3);
        btn_cheng = (Button) findViewById(R.id.btn_cheng );
        btn_0 = (Button) findViewById(R.id.btn_0);
        btn_qing = (Button) findViewById(R.id.btn_qing );
        btn_deng = (Button) findViewById(R.id.btn_deng );
        btn_chu = (Button) findViewById(R.id.btn_chu);

//按钮的单击事件
//数字 0～9 按钮代码
        btn_7.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //按钮单击逻辑
                if(isClickdeng){//说明刚单击了 =,上一个运算刚结束
                    txt_result.setText(""); //重新计算,文本框清空
                    isClickdeng = false; //更改 = 按钮的状态
                }
                txt_result.setText(txt_result.getText().toString() + "7"); //第一种情况,
单击 7 直接显示在文本控件里.第二种情况:先单击别的数字,再单击 7
            }
        });
        btn_8.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //按钮单击逻辑
                if(isClickdeng){//说明刚单击了 =,上一个运算刚结束
                    txt_result.setText(""); //重新计算,文本框清空
                    isClickdeng = false; //更改 = 按钮的状态
                }
                txt_result.setText(txt_result.getText().toString() + "8"); //第一种情况,
单击 8 直接显示在文本控件里.第二种情况:先单击别的数字,再单击 8

            }
        });
        btn_9.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //按钮单击逻辑
                if(isClickdeng){//说明刚单击了 =,上一个运算刚结束
                    txt_result.setText(""); //重新计算,文本框清空
                    isClickdeng = false; //更改 = 按钮的状态
                }
                txt_result.setText(txt_result.getText().toString() + "9"); //第一种情况,
单击 9 直接显示在文本控件里.第二种情况:先单击别的数字,再单击 9

            }
```

```
        });


            …    //此处省略了 4,5,6,按钮代码,请自行补充完整
            …    //此处省略了 1,2,3 按钮代码,请自行补充完整
            …    //此处省略了 0 按钮代码,请自行补充完整
//运算符 + 、- 、* 、/ 按钮单击事件
//运算符 + 按钮代码
        btn_jia.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String st1 = txt_result.getText().toString(); //获取单击加号之前字符串类
                                                              //型的数据
                if(st1.equals("")){//判断获取的数据是否为空
                    return; //返回,什么都不做
                }
      num1 = Double.parseDouble(st1); //将获取的字符串类型的数据转换为 Double 小数类型,这
                                      //是第一个数
                txt_result.setText(""); //清空文本控件中的第一个数
                op = " + "; //表示进行加法计算
                isClickdeng = false; //单击加号按钮时,等号按钮不起作用
            }
        });

            …//此处省略了运算符 - 按钮代码,请自行补充完整
            …//此处省略了运算符 * 按钮代码,请自行补充完整
            …//此处省略了运算符 / 按钮代码,请自行补充完整

//清除 c 按钮代码
        btn_qing.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                txt_result.setText(""); //清除文本框中的内容
            }
        });

//= 按钮代码
        btn_deng.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String str2 = txt_result.getText().toString(); //获取文本框中的数据
                if(str2.equals("")){  //判断获取的数据是否为空
                    return; //返回,什么都不做
                }
     num2 = Double.parseDouble(str2); //将获取的字符串类型的数据转换为 Double 小数类型,这
                                      //是第二个数据
                txt_result.setText(""); //清空文本控件中的第二个数
                switch(op){  //此时判断进行何种操作
                    case " + ":result = num1 + num2;break; //op = +   加法
```

```
                    case "-":result = num1 - num2;break; //op = -   减法
                    case "*":result = num1 * num2;break; //op = *   乘法
                    case "/":result = num1/num2;break; //op = /   除法
                    case "%":result = num2;break; //op = %   不进行计算
                    default:result = 0.0;break;
                }
                txt_result.setText(result + ""); //将 +- */运算的运算结果转换为字符串显
                                                 //示到结果文本框中
                op = "%";
                isClickdeng = true;
            }
        });


    }
}
```

上述代码中，0～9 数字按钮代码雷同。以数字 7 按钮为例，单击数字 7 时，首先要判断等号的状态，如果 isClickdeng 状态为 true，说明上一个运算结果刚结束，此时要清空 EditText 中的运算结果，同时将等号的状态 isClickdeng 置为 false，这样就可以开始接收下一个运算数。如果直接单击数字 7，则将数字 7 显示到 EditText 控件中，代码可表示为：txt_result.setText("7")。如果先单击别的数字之后又单击了数字 7，则需要将先单击数字与 7 连接到一起，形成一个新的数字，显示到 EditText 控件中，因此便出现了此行代码：txt_result.setText(txt_result.getText().toString()＋"7")。如果单击的是数字 8 按钮，那么此行代码就需要更改为：txt_result.setText(txt_result.getText().toString()＋"8")。0～9 这 10 个数字按钮单击事件内部代码相似，请根据已有代码自行补充完整。

＋、－、*、/运算符号代码雷同，以运算符＋为例，单击运算符＋按钮时，表示第一个运算数输入结束，此时获取 EditText 控件的内容便得到了第一个运算数。判断所获得运算数，如果为空，说明用户并没有输入数据，return 表示程序不做任何运算，直接返回空。如果获得第一个运算数不为空，则将获取的字符串类型的数据转换为 double 类型，同时将 EditText 清空，操作运算符置为＋，等号状态 isClickdeng 置为 false，开始准备接收第二个运算数。如果单击的是－按钮，则操作运算符 op＝"－"；单击乘号 * 按钮时将 op＝"*"；单击/按钮时将 op＝"/"。这四个运算符按钮单击事件内部代码相似，请根据已有代码自行补充完整。

“清除”按钮代码较为简单，即清除 EditText 中的输入运算数或者是清空计算结果。

＝按钮代码较为复杂，首先单击＝时，表示第二个运算符输入结束，此时获取 EditText 控件的内容便得到了第二个运算数。同理，判断所获得运算数是否为空，若为空，什么都不做，直接返回。如果获得第二个运算数不为空，则将获取的字符串类型的数据转换为 double 类型，然后判断操作运算符，当操作运算符为＋时，将两个运算数进行加法操作；当操作运算符为－时，将两个运算数进行减法操作；当操作运算符为 * 时，将两个运算数进行乘法操作；当操作运算符为/时，将两个运算数进行除法操作，并将运算结果显示到 EditText 中。

项目小结

本项目内容紧紧围绕 Android 当中的 UI 布局和 Android 开发中的 Style 样式设计的内容来展开，通过三种常用布局的介绍，开阔学生眼界，提升学生设计布局界面的能力。通过 Style 的设计，对布局界面的复杂性和观赏性提出了更高的要求。最后通过智能计算器的案例，培养学生移动端 App 的设计与开发能力。

习题

1. 下面选项中，(　　)布局是线性布局。

A. RelativeLayout　B. LinearLayout　C. FrameLayout　D. TableLayout

2. 线性布局只有两种方向：水平和垂直。(　　)

A. 正确　　B. 错误

3. 在定义输入文本框的样式时，选择的节点类型为(　　)。

A. menu　　B. selector　　C. shape　　D. anim

4. 在定义控件的样式时，(　　)属性表示描边。

A. <solid android:color="#dddddd"/>

B. <stroke android:color="#dfcdae" android:width="3dp"/>

C. <corners android:radius="10dp"/>

D. <padding android:top="20dp"/>

5. 下面关于相对布局，说法错误的是(　　)。

A. 控件相对于父容器进行定位

B. 控件相对于控件进行定位

C. 所有的控件都从左上角开始定位

D. 一个控件可以位于另一个控件的上方、下方、左侧、右侧

6. 下面属性中，(　　)属性表示线性布局的方向是垂直的。

A. android:orientation="vertical"　　B. android:orientation="horizontal"

C. android:layout_gravity="center"　　D. android:layout_weight="1"

7. 所有的 View 都会放在左上角，并且后添加进去的 View 会覆盖之前放进去的 View。具有这种特点的布局是帧布局。(　　)

A. 正确　　B. 错误

8. 在定义控件的样式时，(　　)属性表示圆角半径。

A. <solid android:color="#dddddd"/>

B. <stroke android:color="#dfcdae" android:width="3dp"/>

C. <corners android:radius="10dp"/>

D. <padding android:top="20dp"/>

9. 线性布局中可以嵌套帧布局。(　　)

A. 正确　　B. 错误