《移动终端开发技术》

电子教案

第一单元　常用的布局类型

所属专业（教研室）：计算机软件技术

制定人：陈媛媛

合作人：

制定时间：2018 年 2 月

日照职业技术学院

<table>
<tr><td>单元标题</td><td colspan="2">常用的布局类型</td><td>单元教学学时</td><td>4 课时</td></tr>
<tr><td></td><td colspan="2"></td><td>在整体设计中的位置</td><td>第 5 次</td></tr>
<tr><td>授课班级</td><td colspan="2"></td><td>上课地点</td><td>一体化教室</td></tr>
<tr><td>上课时间</td><td colspan="4">周　　月　　日第　　节</td></tr>
<tr><td rowspan="2">教学
目标</td><td>能力目标</td><td>知识目标</td><td colspan="2">素质目标</td></tr>
<tr><td>能够熟练创建布局文件并应用常用的布局类型。</td><td>1、掌握创建布局文件的方法；
2、掌握相对布局、线性布局的用法；
3、了解表格布局、网格布局、帧布局的用法。</td><td colspan="2">1、养成积极主动学习意识；
2、养成勤于动手的习惯。</td></tr>
<tr><td>教学重点
难点</td><td colspan="4">教学重点：相对布局、线性布局
教学难点：表格布局、网格布局</td></tr>
<tr><td>教学
方法</td><td colspan="4">采用反转课堂教学模式，课前学生学习微课了解知识点，课上采用教师引导、演示，学生分组练习、讨论等教学方法。
运用多媒体、AndroidStudio 开发环境、实训助手、教学平台等辅助授课。</td></tr>
<tr><td>课前需掌握的知识点</td><td colspan="4">常用属性：
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"　　垂直
android:orientation="horizontal"　　水平</td></tr>
<tr><td>教学任务分解</td><td colspan="4">任务一 使用相对布局实现梅花布局效果
任务二 使用线性布局实现按钮的排列
任务三 丰富的表格布局
任务四 使用网格布局实现计算器界面
任务五 使用帧布局实现霓虹灯效果</td></tr>
<tr><td>教学
总结</td><td colspan="4"></td></tr>
</table>

教学过程

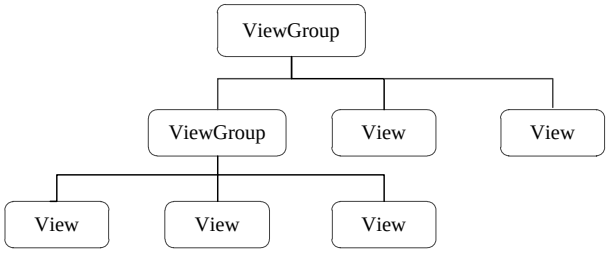
一、情景导入

1、引入 UI 设计

大家有没有发现，Android 应用或者游戏界面做的都非常美观，而且特别吸引人，让大家很有兴趣使用，例如 QQ 界面、微信界面、网易新闻等等。这个就是界面的 UI 效果。

Android 程序开发最重要的一个环节就是界面处理，界面的美观度直接影响用户的第一印象，因此，开发一个整齐、美观的界面是至关重要的。在学习 Android UI 开发之前，首先要了解 UI 这个概念。所谓的 UI（User Interface），它是人与手机之间数据传递、交互信息的重要媒介和对话接口，是 Android 系统的重要组成部分。一个应用程序，除了其功能强大之外，最重要的就是完美的 UI 设计。解了什么是 UI，接下来我们就演示一下 Android 中如何创建布局，并讲述常用的布局类型。

然后引出，一个 Android 应用的界面是由 View 和 ViewGroup 对象构建的。分析 View 与 ViewGroup 的继承关系。



2、引出本单元相关学习内容

本节课我们将一起学习相对布局、线性布局、表格布局、网格布局、帧布局的用法。

二、课前检查

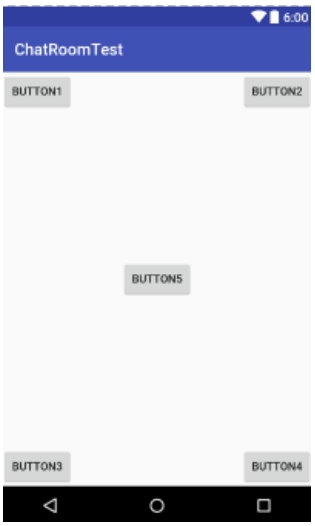
创建一个布局文件的步骤。

三、知识讲解

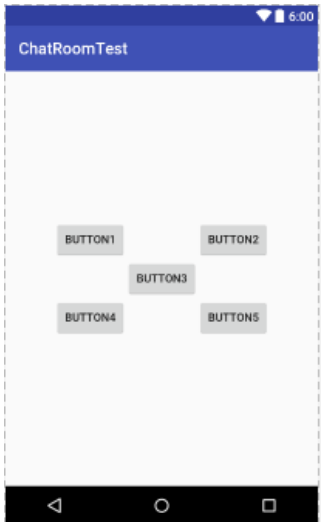
Android 中的常用的布局有五种，分别是相对布局、线性布局、表格布局、网格布局、帧布局本节课先为大家讲解相对布局。

1、相对布局

讲解相对布局时，先要介绍一些常用属性，然后给出相对布局的图，进而给出相对布局的代码。



案例 1 效果图



案例 2 效果图

案例 1、按钮的相对布局

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:text="Button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
       />
    <Button
        android:text="Button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button2"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        />
    <Button
        android:text="Button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:id="@+id/button3" />
    <Button
        android:text="Button4"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:id="@+id/button4" />
    <Button
        android:text="Button5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
       android:layout_centerInParent="true"
       android:id="@+id/button5" />
</RelativeLayout>
```

案例 2、按钮的相对布局

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:text="Button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@id/button3"
        android:layout_toLeftOf="@id/button3"
        android:id="@+id/button1" />
    <Button
        android:text="Button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@id/button3"
        android:layout_toRightOf="@id/button3"
        android:id="@+id/button2" />
    <Button
        android:text="Button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
       android:layout_centerInParent="true"
        android:id="@+id/button3" />
    <Button
        android:text="Button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/button3"
        android:layout_toLeftOf="@id/button3"
        android:id="@+id/button4" />
```

```
    <Button
        android:text="Button5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/button3"
        android:layout_toRightOf="@id/button3"
        android:id="@+id/button5" />
```

2、线性布局

老师引导，下面我们讲解第 2 种布局——线性布局，线性布局是 Android 中较为常用的布局方式，它使用<LinearLayout>标签表示。

介绍线性布局主要有两种形式，一种是水平线性布局，一种是垂直线性布局。

案例 1、水平与垂直排列

android:orientation="vertical"　　垂直

android:orientation="horizontal"　水平

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:text="Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button" />
    <Button
        android:text="Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button2" />
    <Button
        android:text="Button"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:id="@+id/button3" />
</LinearLayout>
```

案例 2、layout_gravity 属性

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:text="Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="top"
        android:id="@+id/button" />
    <Button
        android:text="Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:id="@+id/button2" />
    <Button
        android:text="Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:id="@+id/button3" />
</LinearLayout>
```

案例 3、layout_weigth 属性

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <EditText
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:id="@+id/input_message"
        android:hint="input somthing"/>
    <Button
        android:id="@+id/send"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
```

```
        android:text="send"/>
</LinearLayout>
```

Layout_weight=”1”表示平分宽度，要是一个占 3/5，一个占 2/5 呢？

Layout_weight=”3” Layout_weight=”2”

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <EditText
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:id="@+id/input_message"
        android:hint="input somthing"/>
    <Button
        android:id="@+id/send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="send"/>
</LinearLayout>
```

表示 Button 的宽度按照 wrap_content 计算，而 EditText 则会占满屏幕剩余的空间。

3、表格布局

老师引导，表格布局相对于前两种布局差异较大，表格布局是让控件以表格的形式来排列组件的，只要将组件或信息放在单元格中，组件就可以整齐的排列。

在 TableLayout 中，行数由 TableRow 对象控制的，

在表格布局管理容器中，可以为单元格设置如下 3 种行为方式：

Shrinkable：该列所有单元格的宽度可以被收缩，以保证表格适应父容器的宽度。

Stretchable：该列所有单元格的宽度被拉伸，以保证组件能完全填满表格剩余空间。

Collapsed：该列的所有单元格会被隐藏。

案例 1、丰富的表格布局

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TableLayout
        android:id="@+id/tablelayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:shrinkColumns="1"
        android:stretchColumns="2">
        <!--直接添加按钮，它自己会占一行-->
        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="独占一行的按钮"/>
        <!--添加一行-->
        <TableRow>
            <!--添加三个按钮-->
            <Button
                android:id="@+id/button2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="普通按钮"/>
            <Button
                android:id="@+id/button3"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="收缩的按钮"
                />
            <Button
                android:id="@+id/button4"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="拉伸的按钮"/>
        </TableRow>
    </TableLayout>
    <TableLayout
        android:id="@+id/tablelayout2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:collapseColumns="1">
        <!--直接添加按钮，它自己会占一行-->
        <Button
            android:id="@+id/button5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="独占一行的按钮"/>
        <!--添加一行-->
        <TableRow>
            <!--添加三个按钮-->
            <Button
```

```
                android:id="@+id/button6"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="普通按钮"/>
            <Button
                android:id="@+id/button7"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="收缩的按钮"
                />
            <Button
                android:id="@+id/button8"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="拉伸的按钮"/>
        </TableRow>
    </TableLayout>
    <TableLayout
        android:id="@+id/tablelayout3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:stretchColumns="1,2">
        <!--直接添加按钮，它自己会占一行-->
        <Button
            android:id="@+id/button9"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="独占一行的按钮"/>
        <!--添加一行-->
        <TableRow>
            <!--添加三个按钮-->
            <Button
                android:id="@+id/button10"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="普通按钮"/>
            <Button
                android:id="@+id/button11"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="拉伸的按钮"
                />
            <Button
                android:id="@+id/button12"
```

```
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="拉伸的按钮"/>
        </TableRow>
    </TableLayout>
</LinearLayout>
```

案例 表格效果图

案例 网格布局效果图

4、网格布局

老师引导，网格布局与表格布局有些类似，网格布局用一组无限细的直线将绘图区域分成行、列和单元，并指定控件的显示区域和控件在该区域的显示方式。它实现了控件的交错显示，能够避免因布局嵌套对设备性能的影响，更利于自由布局的开发。

GridLayout 的作用类似 html 中的 table 标签，它把整个容器划分为 rows 和 columns 个网格，每个网格可以放置一个组件。除此之外，也可以设置一个组件横跨多少列、一个组件纵跨多少行。

GridLayout 提供了 setRowCount(int)和 setColumnCount(int)方法来控制该网格的行数量和列数量。属性 rowCount 和 columnCount 控制网格的行数和列数。

需要注意的是，网格布局是 Android4.0 新增的布局，如果在项目中使用，需要把 SDK 的最低版本指定为 Android4.0（API14）以上。

表 1　RelativeLayout 的 XML 属性及相关方法说明

XML 属性	相关方法	说明
android:alignmentMode	setAlignmentMode(int)	设置该布局管理器采用的对齐方式
android:columnCount	setColumnCount(int)	设置该网格的列数量

android:columnOrderPreserved	setColumnOrderPreserved(boolean)	设置该网格容器是否保留列序号
android:rowCount	setRowCount(int)	设置该网格的行数量
android:rowOrderPreserved	setRowOrderPreserved(boolean)	设置该网格容器是否保留行序号
android:useDefaultMargins	setUseDefaultMargins(boolean)	设置该布局管理器是否使用默认的页边距

为了控制 GridLayout 布局容器中各子组件的布局分析，GridLayout 提供了一个内部类，GridLayout.LayoutParams，该类提供了大量的 XML 属性来控制 GridLayout 布局容器中子组件的布局分析。

表 2 显示了 GridLayout.LayoutParams 常用的 XML 属性及相关方法。

表 2 GridLayout.LayoutParams 的 XML 属性及相关方法说明

XML 属性	相关方法	说明
android:layout_column		设置该子组件在 GridLayout 的第几列
android:layout_columnSpan		设置该子组件在 GridLayout 横向上跨几列
android:layout_gravity	setGrvity(int)	设置该子组件采用何种方式占据该网格的空间
android:layout_row		设置该子组件在 GridLayout 的第几行
android:layout_rowSpan		设置该子组件在 GridLayout 纵向上跨几行

案例 1、计算器界面

为了实现计算器界面，可以考虑将该界面分解成一个 6*4 的网格，其中第一个文本框横跨 4 列，第二个按钮横跨 4 列，后面每个按钮各占一格。

为了实现该界面，考虑按如下步骤来实现该界面：

①在布局管理器中定义一个 GridLayout，并在该 GridLayout 中一次定义文本框、按钮，该文本框、按钮各行横跨 4 列。

②在 Java 代码中循环 16 次，依次添加 16 个按钮。

Activity_main.xml 中代码

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="6"
    android:columnCount="4"
    android:id="@+id/root">
    <!--定义一个横跨 4 列的文本框，并设置该文本框的前景色、背景色等属性-->
```

```
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_columnSpan="4"
        android:textSize="50sp"
        android:layout_marginLeft="2pt"
        android:layout_marginRight="2pt"
        android:padding="3pt"
        android:layout_gravity="right"
        android:background="#eee"
        android:textColor="#000"
        android:text="0"/>
    <!--定义一个横跨 4 列的按钮-->
    <Button
      android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_columnSpan="4"
        android:text="清除"/>
</GridLayout>
```

MainActivity.java 代码

```
public class MainActivity extends AppCompatActivity {
GridLayout gridLayout;
    String[] chars=new String[]{"7","8","9","/",
                                "4","5","6","*",
                                "1","2","3","-",
                                ".", "0","=","+"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.gridlayout);
       gridLayout=(GridLayout)findViewById(R.id.root);
        for(int i=0;i<chars.length;i++){
            Button bn=new Button(this);
            bn.setText(chars[i]);
            bn.setTextSize(40);//设置字体大小
            bn.setPadding(5,35,5,35);//设置按钮四周的空白区域
            GridLayout.Spec rowSpec=GridLayout.spec(i/4+2);//设置该组件所在的行从第二行开始放，GridLayout.Spec  rowSpec= GridLayout.spec(4)把一个控件添加到第 5 行（0 是第一行），并且只占一列的大小
            GridLayout.Spec columnSpec=GridLayout.spec(i%4);//设置该组件所在的列
            GridLayout.LayoutParams params=new
GridLayout.LayoutParams(rowSpec,columnSpec);//定义 Layout 的参数
            params.setGravity(Gravity.FILL);//设置该组件占满父容器
            gridLayout.addView(bn,params);
```

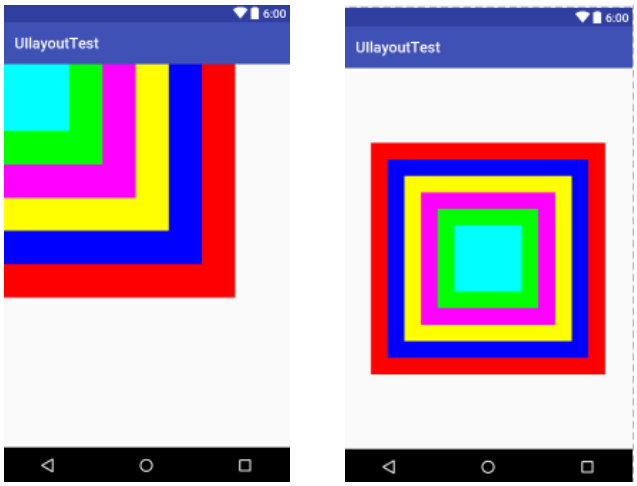
```
        }
    }
```

程序运行结果

5、帧布局

老师提问引导，大家有没有在手机上玩过刮刮卡，你们知道这种界面是怎么实现的吗？同学猜测性的回答，老师说明，其实这种功能就是通过帧布局实现的，一个刮刮卡就是两个重叠在一起的两张图片，通过手指的移动将上一张图片变成透明，然后显示刮奖的效果。

帧布局是 Android 布局中最简单的一种，帧布局为每个加入其中的控件创建一个空白区域（称为一帧，每个控件占据一帧）。采用帧布局方式设计界面时，只能在屏幕左上角显示一个控件，如果添加多个控件，这些控件会按照顺序在屏幕的左上角重叠显示。加上 layout_gravity="center" 可以居中显示。



案例 1、漂亮的帧布局

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/t1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:width="140pt"
        android:height="140pt"
        android:background="#f00"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:width="120pt"
        android:height="120pt"
        android:background="#00f"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:width="100pt"
        android:height="100pt"
        android:background="#ff0"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:width="80pt"
        android:height="80pt"
        android:background="#f0f"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:width="60pt"
        android:height="60pt"
        android:background="#0f0"/>
    <TextView
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:width="40pt"
        android:height="40pt"
        android:background="#0ff"/>
</FrameLayout>
```

四、知识巩固

1、总结知识点，使用教学平台中的随堂练习题巩固本所学知识。

2、使用教学平台中的测试题给学生布置作业。

拓展作业	1、学习微课《常用的控件》； 2、课程平台拓展作业。
教学后记	