LinearLayout 视图组，所有的窗体在一个单一的方向对齐，垂直或水平。所以只会有一个垂直列表，每行一个窗体，不管他们有多宽，水平列表将仅一行高（最高的孩子的高度，再加上填充）

[**Android的几种布局方式**](http://jakielong.iteye.com/blog/734863)

**博客分类：**

* [Android](http://jakielong.iteye.com/category/118129)

[Android](http://www.iteye.com/blogs/tag/Android)[WinForm](http://www.iteye.com/blogs/tag/WinForm)[360](http://www.iteye.com/blogs/tag/360)[XML](http://www.iteye.com/blogs/tag/XML)[OS](http://www.iteye.com/blogs/tag/OS)

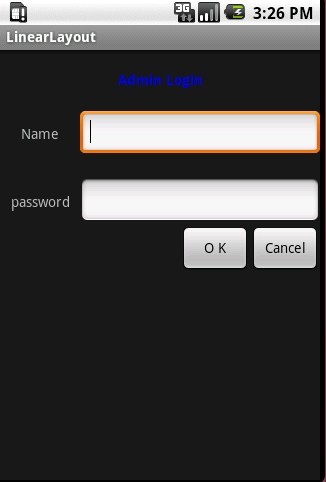
通常我们开发的应用程序都是需要具有友好的用户界面，那么Android中提供了哪些布局方式呢?当我们构思好了所需要的各种控件的时候，怎样才能将它们放到Android设备屏幕上正确的位置呢?在Android中，Layout是负责管理控件在屏幕的位置的类，并且提供了几个简单的布局模型，开发人员通过将这几种布局模型的组合可以构建出我们想要的复杂的用户界面.



（主界面程序列出五种基本的布局方式，选择各种布局可以查看效果）

**1、** **线性布局**

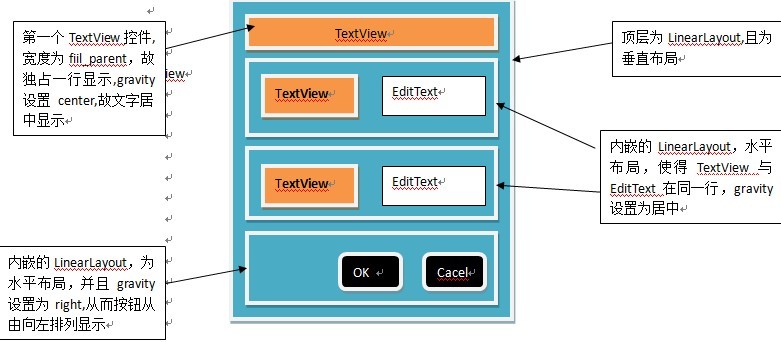
线性布局方式是我们应用程序中最常用的布局方式，主要提供控件水平或者垂直排列的模型，在主界面中点击LinearLayout选项，将会进入线性布局效果显示界面，如下图：



查看该布局界面文件，如下所示：

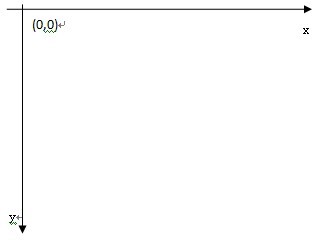
<?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_height="wrap\_content" android:orientation="vertical" android:layout\_width="fill\_parent"> <TextView android:text="Admin Login" android:paddingTop="20dip" android:paddingBottom="20dip" android:textColor="@color/blue" android:id="@+id/TextView01" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content" android:gravity="center" android:soundEffectsEnabled="true" android:password="false" android:textStyle="bold"> </TextView> <LinearLayout android:id="@+id/layout1" android:gravity="center" android:paddingBottom="20dip" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"> <TextView android:id="@+id/txtName" android:text="Name" android:minWidth="80dip" android:gravity="center" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"> </TextView> <EditText android:id="@+id/edtName" android:minWidth="120dip" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content" > </EditText> </LinearLayout> <LinearLayout android:id="@+id/layout2" android:gravity="center" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"> <TextView android:id="@+id/txtPsw" android:minWidth="80dip" android:gravity="center" android:text="password" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"> </TextView> <EditText android:id="@+id/edtPassword" android:minWidth="120dip" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content" > </EditText> </LinearLayout> <LinearLayout android:id="@+id/layout3" android:gravity="right" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content" > <Button android:id="@+id/btnOk" android:minWidth="70dip" android:text="O K" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" > </Button> <Button android:id="@+id/btnCancel" android:minWidth="70dip" android:text="Cancel" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" > </Button> </LinearLayout> </LinearLayout>

从上述我们可以看出，通过灵活的组合LinearLayout布局方式，可以很容易的设计出复杂的一些界面。如下图所示：



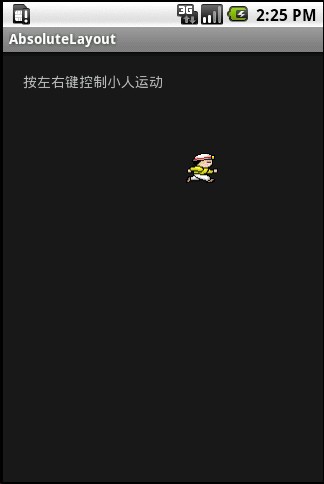
**2、 坐标布局**

坐标布局对于有过.Net Winform开发经验的人员来说应该比较熟悉了，即控件的在父容器上所处的位置主要是有其横纵坐标决定的~在坐标布局中，坐标系如下所示：



对于该布局的示例效果，笔者主要是通过按左右键来控制小人左右移动来讲解的，如下图：





界面布局内容如下：

<?xml version="1.0" encoding="utf-8"?> <AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android" android:id="@+id/AbsoluteLayout01" android:layout\_width="fill\_parent" android:layout\_height="fill\_parent" > <TextView android:id="@+id/txtIntro" android:text="按左右键控制小人运动" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content" android:layout\_x="20dip" android:layout\_y="20dip"> </TextView> <ImageView android:id="@+id/imgMan" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:layout\_x="100dip" android:layout\_y="100dip"> </ImageView> </AbsoluteLayout>

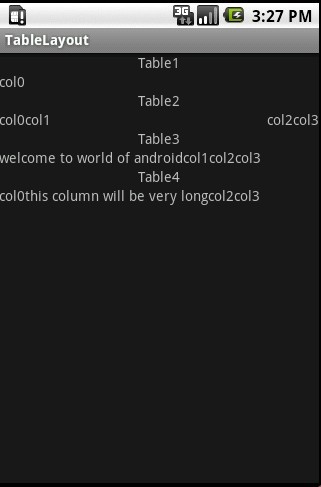
实现控制小人左右走动的动画代码在AbsoluteLayoutActivity.java文件中，内容如下：

package cnblogs.com.layout; import android.app.Activity; import android.os.Bundle; import android.util.DisplayMetrics; import android.view.KeyEvent; import android.widget.ImageView; import cnblogs.com.R; public class AbsoluteLayoutActivity extends Activity{ //人物的View private ImageView man; //人状态的枚举 public enum ManState{ Stand, LeftRun, LeftStop, RightRun, RightStop }; //人当前的状态 private ManState curState=ManState.Stand; private int dx; @Override protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); this.setContentView(R.layout.absolute); man=(ImageView)this.findViewById(R.id.imgMan); man.setBackgroundResource(R.drawable.manstand); } @Override public boolean onKeyDown(int keyCode, KeyEvent event) { boolean ret=super.onKeyDown(keyCode, event); switch(keyCode){ case KeyEvent.KEYCODE\_DPAD\_LEFT: if(curState==ManState.LeftRun){ curState=ManState.LeftStop; man.setBackgroundResource(R.drawable.manleftstop); } else if(curState==ManState.LeftStop){ curState=ManState.LeftRun; man.setBackgroundResource(R.drawable.manleftrun); } else{ curState=ManState.LeftRun; man.setBackgroundResource(R.drawable.manleftrun); } //碰撞检测 dx=man.getLeft()-1>0?man.getLeft()-1:0; man.layout(dx, man.getTop(), dx+Math.abs(man.getWidth()), man.getTop()+Math.abs(man.getHeight())); break; case KeyEvent.KEYCODE\_DPAD\_RIGHT: if(curState==ManState.RightRun){ curState=ManState.RightStop; man.setBackgroundResource(R.drawable.manrightstop); } else if(curState==ManState.RightStop){ curState=ManState.RightRun; man.setBackgroundResource(R.drawable.manrightrun); } else{ curState=ManState.RightRun; man.setBackgroundResource(R.drawable.manrightrun); } //碰撞检测 dx=man.getLeft()+1<360-man.getWidth()?man.getLeft()+1:new DisplayMetrics().widthPixels-man.getWidth(); man.layout(dx, man.getTop(), dx+Math.abs(man.getWidth()), man.getTop()+Math.abs(man.getHeight())); break; default: break; } return ret; } @Override public boolean onKeyUp(int keyCode, KeyEvent event) { boolean ret=super.onKeyUp(keyCode, event); return ret; } }

从这个例子可以看出，实现帧动画的一种方式就是通过不断的改变ImageView控件的横纵坐标，同时更换背景图片即可。

**3、 表格布局**

表格布局主要以行列的形式来管理子控件，其中每一行即一个TableRow对象，每个TableRow对象可以添加子控件，并且每加入一个空间即相当于添加了一列。本文中的示例效果如下所示：



表格界面布局文件内容如下：

<?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" android:orientation="vertical" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content"> <!-- 隐藏第2,3,4列 --> <TableLayout android:id="@+id/table1" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content" android:collapseColumns="1,2,3" > <TextView android:id="@+id/txt1" android:gravity="center" android:text="Table1" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content" /> <TableRow> <TextView android:text="col0"></TextView> <TextView android:text="col1"></TextView> <TextView android:text="col2"></TextView> <TextView android:text="col3"></TextView> </TableRow> </TableLayout> <!-- 设置第二列可伸展,过长可将后面的列挤出可使区 --> <TableLayout android:id="@+id/table2" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content" android:stretchColumns="1" > <TextView android:id="@+id/txt2" android:gravity="center" android:text="Table2" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content" > </TextView> <TableRow> <TextView android:text="col0"></TextView> <TextView android:text="col1"></TextView> <TextView android:text="col2"></TextView> <TextView android:text="col3"></TextView> </TableRow> </TableLayout> <!-- 设置第一列可以收缩 --> <TableLayout android:id="@+id/table3" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content" android:shrinkColumns="0" > <TextView android:id="@+id/txt3" android:gravity="center" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content" android:text="Table3" > </TextView> <TableRow> <TextView android:text="welcome to world of android"></TextView> <TextView android:text="col1"></TextView> <TextView android:text="col2"></TextView> <TextView android:text="col3"></TextView> </TableRow> </TableLayout> <!-- 对各列未作设置,即不可伸缩 --> <TableLayout android:id="@+id/table4" android:gravity="center" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content" > <TextView android:id="@+id/txt3" android:gravity="center" android:layout\_width="fill\_parent" android:layout\_height="wrap\_content" android:text="Table4" > </TextView> <TableRow> <TextView android:text="col0"></TextView> <TextView android:text="this column will be very long"></TextView> <TextView android:text="col2"></TextView> <TextView android:text="col3"></TextView> </TableRow> </TableLayout> </LinearLayout>

展开上述布局内容，最外层是一个垂直布局的LinearLayout的，在其中有四个TableLayout布局，分别展示了四种不同类型的TableLayout布局方式。

**4、 相对布局**

**5、 帧布局**

点击FrameLayout选项，将会进入帧布局效果显示界面，如下图：



上图看上去是不是有点不自然，这是因为上图是通过两个ImageView图片显示控件将两张图片叠在一起显示的。先看下布局界面是怎么编写的：

<?xml version="1.0" encoding="utf-8"?> <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android" android:id="@+id/FrameLayout01" android:layout\_width="fill\_parent" android:layout\_height="fill\_parent" > <ImageView android:id="@+id/img1" android:src="@drawable/shirt" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"> </ImageView> <ImageView android:id="@+id/img2" android:paddingLeft="100dip" android:paddingTop="18dip" android:src="@drawable/head1" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"> </ImageView> </FrameLayout>

可见第一个ImageView控件显示的是T-shirt图片，而第二个ImageView控件则是显示我爱罗的头像。需要注意的是，在帧布局中，先添加的图片会被后添加的图片覆盖。