课 外 拓 展

**实 验 报 告**

姓名 X X X 同组实验者

实验名称 补间动画的实现 日期 2023年xx月xx日

**一.实验目的**

（1）补间动画的实现

**二．实验设备**

1．计算机一台

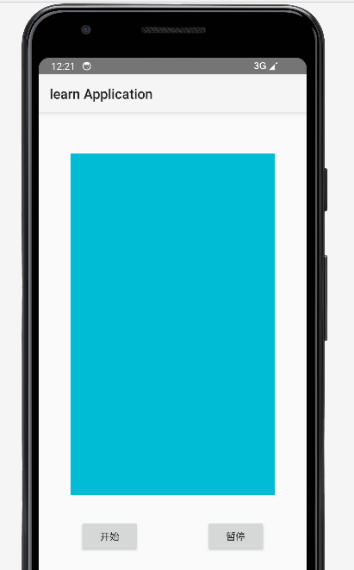
2.Android Studio软件

**三.实验步骤**

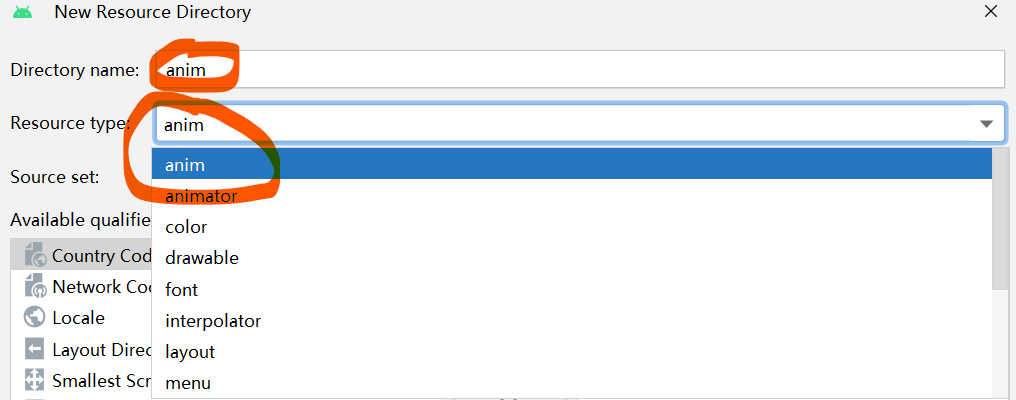
补间动画：

对于我个人理解来说，就是只对一张图实现缩放、曲线运动等操作实现动画。这里我们会用到一个方法 Animation ，

第一步：在lanyout里创建.xml文件做一些简单界面，如下：



接着是在res文件夹下创建一个文件夹叫anim,类型也选择anima，



如果你有就不用创建，然后该文件夹下创建ani\_frame\_1.xml文件，然后在该文件下写：

*<?***xml version="1.0" encoding="utf-8"***?>*<**set xmlns:android="http://schemas.android.com/apk/res/android"**>

</**set**>

第二步，写图片缩放代码，一般我们会用到移动和缩放，我们先写移动的，这会用到一个属性translate

代码如下：

*<?***xml version="1.0" encoding="utf-8"***?>*<**set xmlns:android="http://schemas.android.com/apk/res/android"**>  
 <**translate  
 android:duration="200"  
 android:toXDelta="-100%"** />  
 <**translate  
 android:duration="400"  
 android:startOffset="200"  
 android:toXDelta="100%"** />  
  
</**set**>

第一个translate是左移动200ms，第二个translate是右移动400ms，而且它是等200ms后在开始执行，这里为我们是写400呢，其实实际他移动的时间是200ms，因为它多等了200ms，那么我要是不写400而写200会怎么样呢？你运行后会发现右移动的速度回变快。

* translate 表示移动动作
* duration 是这个动作的执行时长（毫秒）
* toXDelta 表示横向移动，如果X改为Y就是纵向
* startOffset 表示动画开始执行多久后再执行这个动作

第三步，写java代码，用java代码实现该移动动作，我们以之前的帧动画代码为基础，详细可以看我上一个实验的代码，在这里我们在开始按键的监听函数里用Animation写相关启动代码，如下：

**package** com.example.learnapplication;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.annotation.SuppressLint;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.view.animation.Animation;  
**import** android.view.animation.AnimationUtils;  
**import** android.widget.Button;  
**import** android.widget.ImageView;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 **private** ImageView **img**;  
 **private** Button **btn\_start**,**btn\_stop**;  
 **private** Animation **animation**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 **btn\_start** = findViewById(R.id.*btn\_start*);  
 **btn\_stop** = findViewById(R.id.*btn\_stop*);  
 **img** = findViewById(R.id.*iv1*);  
  
 }  
*// 启动* @SuppressLint(**"ResourceType"**)  
 **public void** onClick\_start(View view) {  
 **animation** = AnimationUtils.*loadAnimation*(view.getContext(),R.anim.*ani\_frame\_1*);  
 view.startAnimation(**animation**);  
 }  
  
  
 *// 暂停* **public void** onClick\_stop (View view) {  
 }  
}

我们先定义一个Animation 的类animation，然后在开始监听函数onClick\_start（）使用AnimationUtils.*loadAnimation*(view.getContext(),R.anim.*ani\_frame\_1*)并把这个结果赋给类animation，然后用view.startAnimation(**animation**)启动，R.anim.*ani\_frame\_1*这个就是我们之前写移动属性的.xml文件。

到这里的到这里代码就算写完了，当你点击开始这个按键的时候它就会先往左移动再右移动。

往后的补间动画我们都用这个代码，只需要在*ani\_frame\_1*这里面改一下属性即可。

**四、实验总结**

1.注意设置补间动画动作属性的.xml文件要写在anim文件夹下，别的地方可能会报错，另外使用的是<set></set>;

2.想实现案件控制别的图片缩放，即我不想点击谁谁才缩放，而是间接通过按键控制别的图片缩放，我们只需要在开始的按键监听器函数里加如下代码即可：

**public void** onClick\_start(View view) {  
 view = findViewById(R.id.*iv1*);//这是新添加的代码  
 **animation** = AnimationUtils.*loadAnimation*(view.getContext(),R.anim.*ani\_frame\_1*);  
 view.startAnimation(**animation**);  
}

view = findViewById(R.id.*iv1*)意思就是将要控制的图面id传给view，这之后使用的view.getContext()才知道你传过来的是一个图片视图的id，而不是当前点击的按键视图，不加这一句的话，缩放效果只对按键有效。

**五、拓展**

如果对其他的补间动画属性不了解可以接着往下看。

5.1缩放效果属性设置：

<scale

android:duration="600"

android:fromXScale="1"

android:fromYScale="1"

android:pivotX="50%"

android:pivotY="50%"

android:repeatCount="1"

android:repeatMode="reverse"

android:toXScale="1.35"

android:toYScale="1.35" />

* fromXScale 动画运动的第一帧
* toXScale 动画运动的最后一帧
* android:pivotX 在对象缩放时要保持X 坐标的位置按一定比例变化，如果写0%就是缩放时X轴不变的 ； pivotY 同理。
* android:repeatCount 动画的重复次数。设为 "-1" 表示无限次重复，也可设为正整数。例如，值 "1" 表示动画在初次播放后重复播放一次，因此动画总共播放两次。默认值为 "0"，表示不重复。android:repeatCount 必须设置为正整数或 "-1"，该属性才有效。
* android:repeatMode 设置动画从头到尾的行为。设为 "reverse" 可让动画在每次迭代时反向播放，即再次播放时会先方向走回原来的位置，这个设置看着会连贯一些；设置为 "repeat" 则可让动画每次从头开始循环播放。