安卓属性动画的使用

Duration: 你可以定义一个duration它的默认值是300ms

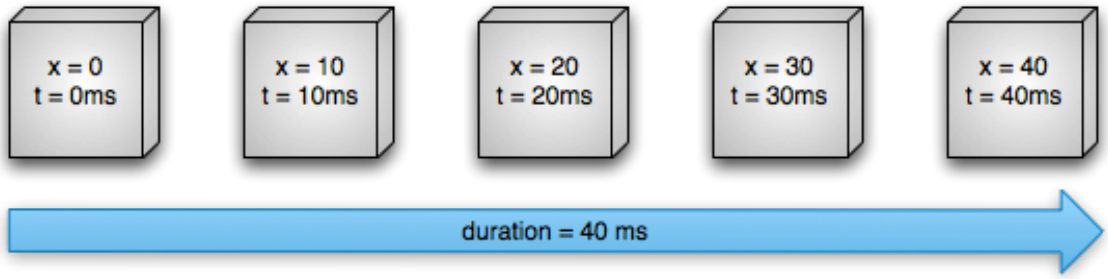
Time interpolation(插入)：你可以指定一个属性的值 用它来处理动画的时间函数

重复的次数以及行为：当一次循环还没有结束时 你可指定或者不指定用多长的时间去重复这个动画 ，你还可以指定你想要播放的动画 设置反复动画 向前然后 向后 直刀到达你所配置的次数

动画设置：可以对动画进行分组 一起玩 或者按顺序 或者按照指定的延迟

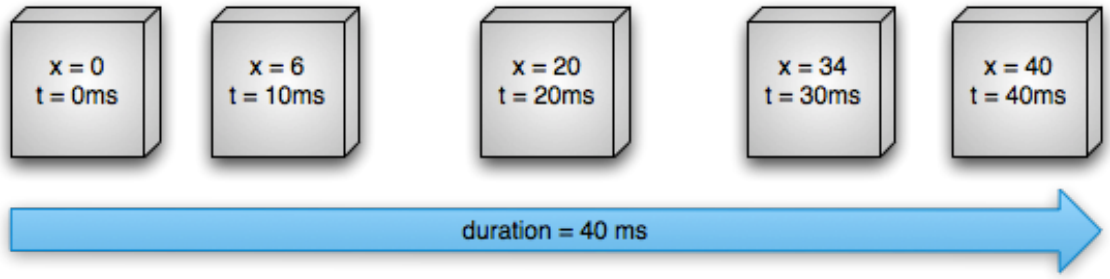
fps 按帧数刷新延迟：你可以指定一个帧数 或者延迟来刷新你的动画 默认是每10秒刷新一次 但是最终如何刷新还是取决于你的系统的多线程繁忙程度 以及系统底层的计时器

属性动画是如何工作的



首先，我们来举个例子来看看动画是怎么工作的 图1：描述了一个虚拟的对象和他的x属性 代表了它在屏幕的水平位置上 动画的设置时间为40ms亿剂在屏幕上移动的距离是40px,每十秒都会按照默认的速度在横向上移动10px

在40ms之后 动画移动了40ms 对象也移动了40ms 这是一个关于线性动画的例子，意味着这个对象是匀速的。



你可以设置动画以一个非匀速的方式移动，图二描述了一个从一开始就加速的动画 然后在东海要结束的时候减速 这个对象依旧在40ms粒移动了40px 这个动画从开始加速一直到中点 然后减速直到结束 图二表示了动画距离的变化 开始和结束

是小于中间的

让我们来详细的看一下属性动画重要的部分 系统会计算动画 就像上面一样 图3描述了 main classe是如何工作的

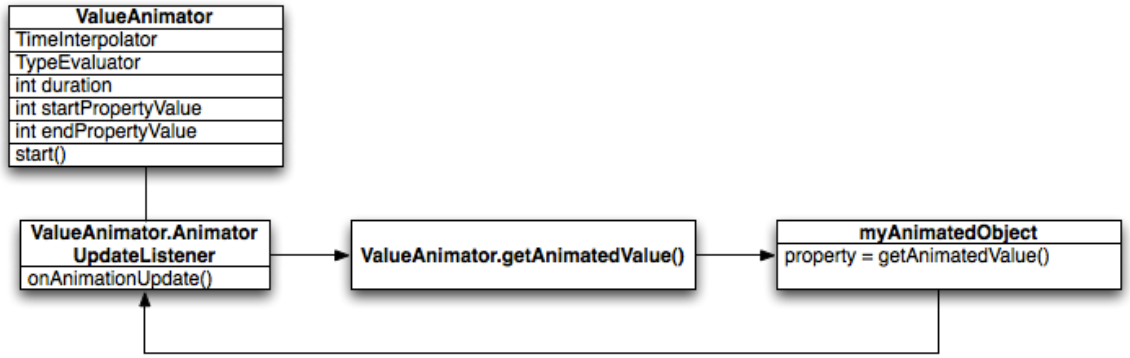


图3 动画是如何计算的

ValueAnimator 用来编辑动画的时机，比如动画运行多长时间 ，并且这些属性都是有生命的 ValueAnimator包含了TimeInterpolator (时间插植器) 定义了动画的插入以及一个 TypeEvaluator(类型鉴别器) 他定义了如何根据动画区计算属性值 举个例子在图二中TimeInterpolator用起来就像AccelerateDecelerateInterpolator(加速减速插入器) and the TypeEvaluator would be IntEvaluator(int鉴别器).

开始一个动画 创建一个ValueAnimator 并且给他一个开始和结束的参数for the property that you want to animate 随着动画的持续时间 当你调用strat() the animation begins

在整个动画的过程中 ValueAnimator计算了一个分数 在0~！之间 随着动画的持续时间 和流逝的时间 流失的时间就代表着动画的完成状况 0代表0% 1代表100% 如图1所示

运行了10ms 就代表着25% because 天河total duration is t=40ms.

当这个ValueAnimator准备好计算 流逝分数时 就会调用TimeInterpolator with current(当前的) set 去计算一个插入分数(*fraction*)

```
mFloatBtn= (FloatingActionButton) findViewById(R.id.float_btn);
        //可以在规定时间实现透明度的渐变 先加速后减速
        ObjectAnimator objAnimatorX=ObjectAnimator.ofFloat(mFloatBtn,"translationX",mFloatBtn.getTranslationX(),+500);
        ObjectAnimator objAnimatorY=ObjectAnimator.ofFloat(mFloatBtn,"translationY",mFloatBtn.getTranslationY(),+500);
```

```
        AnimatorSet animSet = new AnimatorSet();
        //with表示同时进行  after表示按顺序执行
        animSet.play(objAnimatorX).with(objAnimatorY);
        animSet.setDuration(5000);
        animSet.start();
        //实现数值的渐变  先加速后减速
//        ValueAnimator valueAnimator=ValueAnimator.ofFloat(5f,1f,3f,8f);
//        valueAnimator.setDuration(5000);
//        valueAnimator.addUpdateListener(new
ValueAnimator.AnimatorUpdateListener() {
//            @Override
//            public void onAnimationUpdate(ValueAnimator animation) {
//                //拿到当前所对应的值
////                animation.getAnimatedValue()
//                //拿到当前位置所占的百分比
////                getAnimatedFraction
//                Log.e("ValueAnimator",""+animation.getAnimatedFraction()
+Thread.currentThread().getName());
//            }
//        });
//        valueAnimator.start();


//可以在程序内部根据时间的变化计算  两个object内部的计算
    ValueAnimator valueAnimator=ValueAnimator.ofObject(new MyEvaluator(),
10,20,30);
    valueAnimator.setDuration(3000);
    valueAnimator.start();
    valueAnimator.addUpdateListener(this);
}
public class MyEvaluator implements TypeEvaluator<Integer> {


    @Override
    public Integer evaluate(float fraction, Integer startValue, Integer endValue) {
        Log.e("ceshi","数据fraction="+fraction);
        return startValue-endValue;
    }
}
```

layoutAnimation 用于为一个layout中的控件 以及viewGroup里面的控件谁设置动画
http://blog.csdn.net/imdxt1986/article/details/6952943

自定义插植器 可以定义如何计算时间的函数 比如你可以指定整个动画为线性

```
      valueAnimator.setInterpolator(new MyInterpolator());
        valueAnimator.start();
        valueAnimator.addUpdateListener(this);
//        ObjectAnimator
```

```
objectAnimator=ObjectAnimator.ofFloat(runningView,"translationY",runningView.getTranslationX()-500,0);
//        objectAnimator.setDuration(4000);
//        objectAnimator.start();
    }
    public class MyInterpolator implements Interpolator{


//匀速 线性
        @Override
        public float getInterpolation(float input) {
            return input;
        }
//加速减速
//public float getInterpolation(float input) {
//    return (float)(Math.cos((input + 1) * Math.PI) / 2.0f) + 0.5f;
//}

    }
```

下表返回了不同的插值器时间的分布　　左边为线性　　右边为加速减速

ms elapsed	Elapsed fraction/Interpolated fraction (Linear)	Interpolated fraction (Accelerate/Decelerate)
0	0	0
200	.2	.1
400	.4	.345
600	.6	.8
800	.8	.9
1000	1	1

通过keyFrame 键值对 去进行动画的处理

```
Keyframe fram0=Keyframe.ofFloat(0f,0);
Keyframe fram1=Keyframe.ofFloat(0.3f,120);
Keyframe fram2=Keyframe.ofFloat(0.6f,360);
Keyframe fram3=Keyframe.ofFloat(1,0);
PropertyValuesHolder valuesHolder=PropertyValuesHolder.ofKeyframe("rotation",fram0,fram1,fram2,fram3);
ObjectAnimator rotationAnim = ObjectAnimator.ofPropertyValuesHolder(runningView, valuesHolder);
rotationAnim.setDuration(5000);
rotationAnim.start();
```

设置不同进度所对应的不同状态 For a more complete example on how to use keyframes, see the MultiPropertyAnimation sample in APIDemos.

部分属性的介绍

- translationX and translationY: These properties control where the View is located as a delta from its left and top coordinates which are set by its layout container.
- rotation, rotationX, and rotationY: These properties control the rotation in 2D (rotationproperty) and 3D around the pivot point.
- scaleX and scaleY: These properties control the 2D scaling of a View around its pivot point.
- pivotX and pivotY: These properties control the location of the pivot point, around which the rotation and scaling transforms occur. By default, the pivot point is located at the center of the object.
- x and y: These are simple utility properties to describe the final location of the View in its container, as a sum of the left and top values and translationX and translationY values.
- alpha: Represents the alpha transparency on the View. This value is 1 (opaque) by default, with a value of 0 representing full transparency (not visible).

改变view的属性

```
runningView.animate().x(50f).y(100f).start();
```

在xml中使用属性动画

```
<set android:ordering="sequentially">
    <set>
        <objectAnimator
            android:propertyName="x"
            android:duration="500"
            android:valueTo="400"
            android:valueType="intType"/>
        <objectAnimator
            android:propertyName="y"
            android:duration="500"
            android:valueTo="300"
            android:valueType="intType"/>
    </set>
    <objectAnimator
        android:propertyName="alpha"
        android:duration="500"
        android:valueTo="1f"/>
</set>

AnimatorSet set = (AnimatorSet) AnimatorInflater.loadAnimator(myContext,
    R.anim.property_animator);
set.setTarget(myObject);
set.start();
```

ajkndjhanmdn环信