今天介绍4种动画插值器

1.OvershootInterpolator： “Overshoot” means 冲过头了，模拟了冲过了头回滚一点的效果

2.AnticipateInterpolator：“Anticipate” means 抢先，模拟了出发前先后退一步再前冲的动画效果

3.AnticipateOvershootInterpolator：以上两种的结合

4.BounceInterpolator：“Bounce” means 弹跳, 就是模拟了自由落地后回弹的效果

**OvershootInterpolation**

--------------------------------------------------------------------------------

先来看Overshoot的实现的数学公式,mTension默认值为2.0f,有一个构造函数可以进行设置。

public float getInterpolation(float t) {

t -= 1.0f;

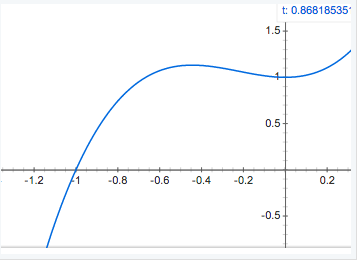
return t \* t \* ((mTension + 1) \* t + mTension) + 1.0f;

}

如果带入默认值，简化下就是

y=3t^3+2t^2+1 // t:[-1~0]

然后去google搜索一下"3t3+2t2+1”,就能看到下面这个曲线图



我们改下参数mTension,就能发现mTension越大，冲过的距离越大，效果越明显!

**AnticipateInterpolation**

--------------------------------------------------------------------------------

再看下Anticipate的实现数学公式，mTension同样默认是2.0f：

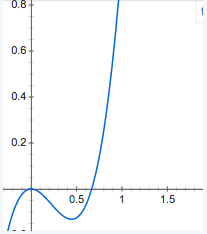
public float getInterpolation(float t) {

return t \* t \* ((mTension + 1) \* t - mTension);

}

于是带上实际参数2.0f，得到

y=3t^3-2t^2 //t:[0~1.0]



同样的结论：mTension越大，抢先动画约长，效果越明显！

**BounceInterpolator**

--------------------------------------------------------------------------------

从源码我们能发现，实现方式是根据时间t，模拟了一次自由落体运动，所以Google这次帮不了我们了。

private static float bounce(float t) {

return t \* t \* 8.0f;

}

public float getInterpolation(float t) {

t \*= 1.1226f;

if (t < 0.3535f) return bounce(t);

else if (t < 0.7408f) return bounce(t - 0.54719f) + 0.7f;

else if (t < 0.9644f) return bounce(t - 0.8526f) + 0.9f;

else return bounce(t - 1.0435f) + 0.95f;

}