**Interpolator**

[Новые интерполяторы из библиотеки поддержки](http://developer.alexanderklimov.ru/android/animation/interpolator.php#support)

Атрибут **android:interpolator** в файле анимации определяет постоянную скорости, которая описывает динамику визуальной деятельности в зависимости от времени. Интерполяторы часто используют в различных библиотеках для создания реалистичных анимаций. Например, любой кот знаком с анимацией падающего мячика, который после каждого отскока поднимается на меньшую высоту и постепенно остаётся лежать неподвижным на земле. Другой пример - вы дёргаете за ручку ящика, который постоянно заедает в письменном столе - сначала вы несколько раз безуспешно дёргаете за ручку, а потом рывком открываете ящик. Очень много подобных видов интерполяторов есть в JavaScript-библиотеке jQuery. В Android встроенных интерполяторов поменьше. Рассмотрим их на примерах.

Список интерполяторов (в скобках указан соответствующий класс)

* accelerate\_interpolator (AccelerateInterpolator)
* decelerate\_interpolator (DecelerateInterpolator)
* accelerate\_decelerate\_interpolator (AccelerateDecelerateInterpolator)
* anticipate\_interpolator (AnticipateInterpolator)
* anticipate\_overshoot\_interpolator (AnticipateOvershootInterpolator)
* overshoot\_interpolator (OvershootInterpolator)
* bounce\_interpolator (BounceInterpolator)
* cycle\_interpolator (CycleInterpolator)
* linear\_interpolator (LinearInterpolator)

Если представить себе переход от состояния "от" до состояния "до" в виде прямой линии, то интерполятор определяет, в какой точке этой линии будет находиться переход в любой момент времени. Самый простой интерполятор - линейный. Он делит прямую на равные отрезки и осуществляет равномерное перемещение по этим отрезкам на протяжении указанного времени. В результате объект движется без ускорения от первого состояния до второго.

По умолчанию, используется интерполятор **accelerate\_decelerate\_interpolator**, в котором используется плавное ускорение, а затем плавное замедление.

Интерполятор **bounce\_interpolator** - это стандартный отскок, когда объект доходит до конечной точки и отскакивает обратно к начальной точке. И такое действие повторяется несколько раз.

Создадим в папке **res/anim** 9 файлов для каждого вида интерполятора.

Показать код (щелкните мышкой)

accelerate.xml

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/accelerate\_interpolator" >

<translate

android:duration="2000"

android:fromYDelta="-100%p"

android:toYDelta="0" />

</set>

decelerate.xml

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/decelerate\_interpolator" >

<translate

android:duration="2000"

android:fromYDelta="-100%p"

android:toYDelta="0" />

</set>

accelerate\_decelerate.xml

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/accelerate\_decelerate\_interpolator" >

<translate

android:duration="2000"

android:fromYDelta="-100%p"

android:toYDelta="0" />

</set>

anticipate.xml

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/anticipate\_interpolator" >

<translate

android:duration="2000"

android:fromYDelta="-80%p"

android:toYDelta="0" />

</set>

anticipate\_overshoot.xml

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/anticipate\_overshoot\_interpolator" >

<translate

android:duration="2000"

android:fromYDelta="-80%p"

android:toYDelta="0" />

</set>

overshoot.xml

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/overshoot\_interpolator" >

<translate

android:duration="2000"

android:fromYDelta="-80%p"

android:toYDelta="0" />

</set>

bounce.xml

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/bounce\_interpolator" >

<translate

android:duration="2000"

android:fromYDelta="-100%p"

android:toYDelta="0" />

</set>

cycle.xml

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/cycle\_interpolator" >

<translate

android:duration="2000"

android:fromYDelta="-50%p"

android:toYDelta="0" />

</set>

linear.xml

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/linear\_interpolator" >

<translate

android:duration="2000"

android:fromYDelta="-100%p"

android:toYDelta="0" />

</set>

Теперь на главном экране приложения разместим кнопки для запуска каждого вида анимации и ImageView, к которому будем применять выбранную анимацию.

main.xml

Показать код (щелкните мышкой)

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="vertical" >

<LinearLayout

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="horizontal" >

<LinearLayout

android:layout\_width="wrap\_content"

android:layout\_height="fill\_parent"

android:layout\_margin="30dp"

android:gravity="bottom"

android:orientation="vertical" >

<ImageView

android:id="@+id/image"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:src="@drawable/ic\_launcher" />

</LinearLayout>

<LinearLayout

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="vertical" >

<Button

android:id="@+id/bounce"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Bounce" />

<Button

android:id="@+id/accelerate"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Accelerate" />

<Button

android:id="@+id/decelerate"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Decelerate" />

<Button

android:id="@+id/acceleratedecelerate"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Accelerate\_Decelerate" />

<Button

android:id="@+id/anticipate"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Anticipate" />

<Button

android:id="@+id/anticipateovershoot"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Anticipate\_Overshoot" />

<Button

android:id="@+id/overshoot"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Overshoot" />

<Button

android:id="@+id/cycle"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Cycle" />

<Button

android:id="@+id/linear"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Linear" />

</LinearLayout>

</LinearLayout>

</LinearLayout>

Запускаем

Пишем код для каждого вида анимаций. Несмотря на большое количество строк, код очень простой. Создаем объект Animation для каждого вида анимации и запускаем созданную анимацию при нажатии соответствующей кнопки.

Показать код (щелкните мышкой)

Для разнообразия у кнопки с анимацией Bounce (любимая анимация котов) я привёл программное использование объектов **TranslateAnimation** и **BounceInterpolator**.



Для закрепления материала создадим ещё один проект. Нам понадобится разметка с текстовыми метками:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:gravity="center\_horizontal"

android:orientation="vertical" >

<TextView

android:id="@+id/firstTextView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:gravity="center\_horizontal"

android:text="Кот"

android:textSize="42sp" />

<TextView

android:id="@+id/secondTextView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:gravity="center\_horizontal"

android:text="Который"

android:textSize="42sp" >

</TextView>

<TextView

android:id="@+id/thirdTextView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:gravity="center\_horizontal"

android:text="Гулял"

android:textSize="42sp" >

</TextView>

<TextView

android:id="@+id/fourthTextView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:gravity="center\_horizontal"

android:text="Сам"

android:textSize="42sp" >

</TextView>

<TextView

android:id="@+id/fifthTextView"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:gravity="center\_horizontal"

android:text="По себе"

android:textSize="42sp" >

</TextView>

</LinearLayout>

Создадим два файла анимации в папке **res/anim**.

scalerotate.xml

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android" >

<rotate

android:interpolator="@android:anim/anticipate\_overshoot\_interpolator"

android:duration="3000"

android:fromDegrees="359"

android:pivotX="50%"

android:pivotY="50%"

android:toDegrees="0" />

<scale

android:interpolator="@android:anim/accelerate\_interpolator"

android:duration="3000"

android:fromXScale="1.0"

android:fromYScale="1.0"

android:toXScale="2.0"

android:toYScale="2.0" >

</scale>

</set>

translatealpha.xml

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/accelerate\_decelerate\_interpolator" >

<translate

android:duration="3000"

android:fromXDelta="-200%"

android:fromYDelta="200%"

android:toXDelta="0%"

android:toYDelta="0%"

android:zAdjustment="top" />

<alpha

android:duration="6000"

android:fromAlpha="0.0"

android:toAlpha="1.0" />

</set>

Обратите внимание, что в первом файле интерполяторы заданы отдельно для каждой анимации, а во втором - для секции **set**.

Напишем отдельный метод, который можно вызывать в методе активности **onCreate()**:

private void RunAnimations() {

Animation a = AnimationUtils.loadAnimation(this, R.anim.translatealpha);

a.reset();

TextView tv = (TextView) findViewById(R.id.firstTextView);

tv.clearAnimation();

tv.startAnimation(a);

a = AnimationUtils.loadAnimation(this, R.anim.translatealpha);

a.reset();

tv = (TextView) findViewById(R.id.secondTextView);

tv.clearAnimation();

tv.startAnimation(a);

a = AnimationUtils.loadAnimation(this, R.anim.scalerotate);

a.reset();

tv = (TextView) findViewById(R.id.thirdTextView);

tv.clearAnimation();

tv.startAnimation(a);

a = AnimationUtils.loadAnimation(this, R.anim.translatealpha);

a.reset();

tv = (TextView) findViewById(R.id.fourthTextView);

tv.clearAnimation();

tv.startAnimation(a);

a = AnimationUtils.loadAnimation(this, R.anim.scalerotate);

a.reset();

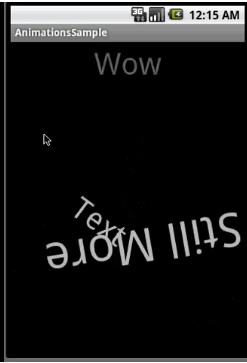
tv = (TextView) findViewById(R.id.fifthTextView);

tv.clearAnimation();

tv.startAnimation(a);

}

На экране вы увидите следующую картинку



**BounceView**

Создадим собственный простейший компонент **BounceView**, который будет выводить текст с анимацией в месте касания экрана.

**BounceView.java**

package ru.alexanderklimov.bounce;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.view.MotionEvent;

import android.view.View;

import android.view.animation.BounceInterpolator;

import android.view.animation.TranslateAnimation;

public class BounceView extends View {

float currentX;

float currentY;

public BounceView(Context context) {

super(context);

}

@Override

protected void onDraw(Canvas canvas) {

if (currentX > 0 && currentY > 0) {

Paint p = new Paint();

p.setColor(Color.RED);

p.setTextSize(20);

canvas.drawText("Туда-сюда", currentX, currentY, p);

}

}

@Override

public boolean onTouchEvent(MotionEvent event) {

currentX = event.getX();

currentY = event.getY();

TranslateAnimation translateanim = new TranslateAnimation(-100, 0, 0, 0);

translateanim.setInterpolator(new BounceInterpolator());

translateanim.setDuration(1000);

startAnimation(translateanim);

invalidate();

return true;

}

}

Подключаем компонент в основной активности:

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

BounceView bounce = new BounceView(this);

setContentView(bounce);

}

Запустите проект и посмотрите на эффект.

**Новые интерполяторы из библиотеки поддержки**

В библиотеке поддержки v4 Support Library появились новые интерполяторы, которые разработаны с учётом Material Design.

* **FastOutLinearInInterpolator** - Быстро ускоряется и сохраняет ускорение до конца анимации. Используется при исчезновении объекта с экрана с полным уходом за границы, иначе произойдёт слишком резкая остановка объекта. Подходит для эффектов "circular reveal", т.е. для появления объектов «из круга». Ближайший аналог **AccelerateInterpolator**.
* **FastOutSlowInInterpolator** - Движение объекта по экрану. Быстро ускоряется и постепенно замедляется. Используется при движении объекта в пределах экрана, когда он виден от начала и до конца движения. Сюда же относится большинство случаев вращения объектов. Ближайший аналог **AccelerateDecelerateInterpolator**
* **LinearOutSlowInInterpolator** - Начинает анимацию с полным ускорением и постепенно замедляется. Используется при появлении объекта из-за пределов экрана и коллапс эффекта "circular", т.е. полная противоположность **FastOutLinearInInterpolator**. Ближайший аналог **DecelerateInterpolator**

Использование интерполяторов делает анимацию более реалистичной. Берите пример с котов!