**Анимация преобразований**

[alpha](http://developer.alexanderklimov.ru/android/tweenanimation.php#alphpa)  
[scale](http://developer.alexanderklimov.ru/android/tweenanimation.php#scale)  
[translate](http://developer.alexanderklimov.ru/android/tweenanimation.php#translate)  
[rotate](http://developer.alexanderklimov.ru/android/tweenanimation.php#rotate)  
[AnimationListener](http://developer.alexanderklimov.ru/android/tweenanimation.php#animationlistener)  
[Примеры анимаций](http://developer.alexanderklimov.ru/android/tweenanimation.php#samples)

Анимация может выполняться в виде ряда простых преобразований — изменение позиции, размера, угла вращения и уровня прозрачности на поверхности объекта View. Например, у объекта TextView можно перемещать, вращать, уменьшать или увеличивать текст. Если объект TextView имеет фоновое изображение, оно будет преобразовано наряду с текстом. Пакет **android.view.animation** содержит необходимые классы для анимации с промежуточными кадрами.

Основные классы анимации и их соответствия в XML:

* **AnimationSet** (элемент <set>) — класс, представляющий группу анимаций, которые должны запускаться вместе. Если класс AnimationSet устанавливает какие-либо свойства, эти свойства наследуют и объекты, входящие в группу;
* **AlphaAnimation** (элемент <alpha>) — управляет прозрачностью объекта;
* **RotateAnimation** (элемент <rotate>) — управляет вращением объекта;
* **ScaleAnimation** (элемент <scale>) — управляет масштабированием объекта, т.е. изменениями размеров;
* **TranslateAnimation** (элемент <translate>) — управляет позиционированием объекта (перемещение)

Команды анимации определяют преобразования, которые необходимо произвести над объектом. Преобразования могут быть последовательными или одновременными. Каждое преобразование принимает набор параметров, определённых для этого преобразования (начальный размер, конечный размер при изменении размера, стартовый угол и конечный угол при вращении объекта и т. д.), а также набор общих параметров (например, начального времени и продолжительности). Если требуется сделать несколько преобразований одновременно, им задают одинаковое начальное время. Если требуется сделать последовательные преобразования, задается их время старта плюс продолжительность предыдущего преобразования.

Последовательность команд анимации определяется в XML-файле (предпочтительно) или в программном коде.

**Создание анимации в XML-файле**

XML-файл анимации располагают в каталоге **res/anim/** вашего проекта. Файл должен иметь единственный корневой элемент: это будет любой из элементов <alpha>, <scale>, <translate>, <rotate> или элемент <set>, который является контейнером для этих четырех компонентов (и может включать в себя другой контейнер <set>).

По умолчанию все элементы применяются одновременно. Чтобы запускать элементы последовательно, необходимо определить атрибут **startOffset** и указать значение в миллисекундах, например:

android:startOffset="3000"

**Атрибуты анимации**

У элементов <alpha>, <scale>, <translate>, <rotate> и <set> поддерживаются общие атрибуты, унаследованные от базового класса Animation:

* **duration** — продолжительность эффекта в миллисекундах;
* **startOffset** — начальное время смещения для этого эффекта, в миллисекундах;
* **fillBefore** — когда установлен в *true*, то преобразование анимации применяется перед началом анимации;
* **fillAfter** — когда установлен в *true*, то преобразование применяется после конца анимации;
* **repeatCount** — определяет число повторений анимации, можно использовать значение **infinity** (бесконечность);
* **repeatMode** — определяет поведение анимации при ее окончании. Возможные варианты: restart (перезапустить без изменений) или reverse (изменить анимацию в обратном направлении);
* **zAdjustment** — определяет режим упорядочения оси Z, чтобы использовать при выполнении анимации (нормаль, вершина или основание);
* **interpolator** — определяет постоянную скорости, которая описывает динамику визуальной деятельности в зависимости от времени или, говоря простым языком, определяет скорость изменения анимации. Можно использовать любой из элементов подкласса интерполятора, определенных в R.styleable, например: android:interpolator="@android:anim/decelerate\_interpolator"

**Элемент <set>**

Элемент <set> — контейнер, который может содержать другие элементы. Представляет класс **AnimationSet**. Поддерживает атрибут **shareInterpolator**, который при значении *true* указывает на возможность совместного использования этого интерполятора для всех дочерних элементов.

**Элемент <alpha>**

Постепенно изменяющаяся анимация прозрачности при помощи AlphaAnimation. Поддерживает следующие атрибуты:

* **fromAlpha** — начальное значение прозрачности объекта;
* **toAlpha** — конечное значение прозрачности объекта;

Атрибуты содержат значение прозрачности от 0 до 1 типа Float, где 0 означает полную прозрачность объекта.

**Элемент <scale>**

Элемент <scale> управляет анимацией изменения размеров объекта и представляет класс **ScaleAnimation**. Вы можете определить центральную точку изображения (закрепить центр анимации изображения), относительно которой будет изменяться масштабирование объекта. Элемент <scale> поддерживает следующие атрибуты:

* **fromxScale** — начальный масштаб по X. Допустимы значения от 0 до 1 типа Float;
* **toxScale** — конечный масштаб по X. Допустимы значения от 0 до 1 типа Float;
* **fromYScale** — начальный масштаб по Y. Допустимы значения от 0 до 1 типа Float;
* **toYScale** — конечный масштаб по Y. Допустимы значения от 0 до 1 типа Float;
* **pivotX** — Х-координата закрепленного центра. Описывает центральную точку масштабирования в процентах от ширины от 0% до 100%;
* **pivotY** — Y-координата закрепленного центра. Описывает центральную точку масштабирования в процентах от высоты от 0% до 100%;

**Элемент <translate>**

Элемент <translate> создает вертикальную или горизонтальную анимацию движения. Представляет класс TranslateAnimation и поддерживает следующие атрибуты:

* **fromXDelta** — начальное положение по X;
* **toXDelta** — конечное положение по X;
* **fromYDelta** — начальное положение по Y;
* **toYDelta** — конечное положение по Y

Атрибуты должны быть в любом из следующих трёх форматов:

* Абсолютное значение
* Значения в процентах от -100% до 100%
* Значения в процентах от -100%p до 100%p, где p указывает процент относительно его родителя.

**Слайд-шоу**

Например, для эффекта слайд-шоу, когда один элемент выталкивает полностью другой элемент, реализуется двумя анимациями [справо-налево или слева-направо](http://developer.alexanderklimov.ru/android/animation/slideshow.php)

**Элемент <rotate>**

Элемент <rotate> предназначен для анимации вращения и представляет класс **RotateAnimation**. Поддерживает следующие атрибуты:

* **fromDegrees** — начальный угол вращения в градусах;
* **toDegrees** — конечный угол вращения в градусах;
* **pivotX** — координата X центра вращения в пикселах;
* **pivotY** — координата Y центра вращения в пикселах;

**Примеры с анимацией преобразований**

Можно создать несколько файлов анимации и применять их для одного объекта. Рассмотрим работу анимации на примере прямоугольника.

Создайте новый проект и в каталоге **res/anim/** проекта создайте пять файлов с XML-анимацией: alpha.xml, rotate.xml, scale.xml, translate.xml и файл combination.xml, в котором мы будем использовать комбинацию различных объектов для создания смешанной анимации.

**alpha.xml**

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

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

android:shareInterpolator="false" >

<alpha

android:duration="5000"

android:fromAlpha="0.0"

android:startOffset="0"

android:toAlpha="1.0" >

</alpha>

</set>

**rotate.xml**

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

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

android:shareInterpolator="false" >

<rotate

android:duration="5000"

android:fromDegrees="0"

android:pivotX="50%"

android:pivotY="50%"

android:toDegrees="360" />

</set>

**scale.xml**

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

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

android:shareInterpolator="false" >

<scale

android:duration="2500"

android:fromXScale="1.0"

android:fromYScale="1.0"

android:pivotX="50%"

android:pivotY="50%"

android:toXScale="2.0"

android:toYScale="2.0" />

<scale

android:duration="2500"

android:fromXScale="1.0"

android:fromYScale="1.0"

android:pivotX="50%"

android:pivotY="50%"

android:startOffset="2500"

android:toXScale="0.5"

android:toYScale="0.5" />

</set>

**translate.xml**

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

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

android:shareInterpolator="false" >

<translate

android:duration="2500"

android:fillAfter="true"

android:toYDelta="-100" />

<translate

android:duration="2500"

android:fillAfter="true"

android:startOffset="2500"

android:toYDelta="100" />

</set>

**combination.xml**

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

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

android:shareInterpolator="false" >

<scale

android:duration="2500"

android:fromXScale="1.0"

android:fromYScale="1.0"

android:pivotX="50%"

android:pivotY="50%"

android:toXScale="2.0"

android:toYScale="2.0" />

<rotate

android:duration="5000"

android:fromDegrees="0"

android:pivotX="50%"

android:pivotY="50%"

android:toDegrees="360" />

<translate

android:duration="2500"

android:fillAfter="false"

android:toYDelta="-100" />

<scale

android:duration="2500"

android:fromXScale="1.0"

android:fromYScale="1.0"

android:pivotX="50%"

android:pivotY="50%"

android:startOffset="2500"

android:toXScale="0.5"

android:toYScale="0.5" />

<translate

android:duration="2500"

android:fillAfter="false"

android:startOffset="2500"

android:toYDelta="100" />

</set>

**shape.xml**

Фигуру прямоугольника для анимации определим в файле shape.xml, который будет находиться в каталоге **res/drawable/**.

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

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

android:shape="rectangle" >

<solid android:color="#F00" />

</shape>

**Разметка**

Перейдем к разметке основной формы приложения (main.xml). Поместим элемент ImageView:

<?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:layout\_gravity="center\_vertical|center\_horizontal"

android:orientation="horizontal" >

<ImageView

android:id="@+id/image"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_margin="115px"

android:minHeight="90px"

android:minWidth="90px" />

</LinearLayout>

**Анимация**

Анимация запускается в коде следующим образом: надо создать объект **Animation** через вызов метода **AnimationUtils.loadAnimation()** и передать ему в качестве параметра контекст активности и ссылку на XML-файл анимации. Затем запускается анимация через метод View.startAnimation(), передавая в него объект Animation:

ImageView image = (ImageView) findViewById(R.id.image);

Animation animation = AnimationUtils.loadAnimation(this, R.anim.alpha);

image.startAnimation(animation);

**AnimationListener**

В классе Animation есть вложенный интерфейс **AnimationListener**. Интерфейс AnimationListener позволяет создать обработчик событий, который срабатывает в начале или при завершении анимации. Используя его, вы можете совершать какие-либо операции, прежде чем (или после того как) анимация закончит работу. Это может быть изменение содержимого представления или последовательный показ нескольких анимаций.

У интерфейса AnimationListener есть три метода обратного вызова:

* onAnimationEnd()
* onAnimationRepeat()
* onAnimationStart()

В этих методах можно реализовать код обработки события запуска, окончания и перезапуска анимации. Например, при окончании анимации можно сделать объект анимации невидимым, а при запуске снова отобразить на экране:

Вызовите метод **setAnimationListener()** из объекта Animation и передайте ему в качестве параметра реализацию интерфейса AnimationListener, при необходимости переопределив методы onAnimationEnd(), onAnimationStart() и onAnimationRepeat().

@Override

public void onAnimationStart(Animation animation) {

mImage.setVisibility(View.VISIBLE);

}

@Override

public void onAnimationEnd(Animation animation) {

mImage.setVisibility(View.INVISIBLE);

}

@Override

public void onAnimationRepeat(Animation animation) {

mImage.setVisibility(View.VISIBLE);

}

В основном классе создадим меню из пяти пунктов, соответствующих каждому типу запускаемой анимации: Alpha, Scale, Translate, Rotate и Комбинация. В качестве идентификаторов пунктов меню используем идентификаторы ресурсов XML-файлов анимации, упростив тем самым структуру метода onOptionsitemSeiected(), вызываемого при выборе пункта меню.

package ru.alexanderklimov.tweenanimation;

import android.app.Activity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.view.animation.Animation;

import android.view.animation.Animation.AnimationListener;

import android.view.animation.AnimationUtils;

import android.widget.ImageView;

public class TweenAnimationActivity extends Activity implements AnimationListener{

private ImageView mImage;

Animation animation = null;

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

mImage = (ImageView)findViewById(R.id.image);

mImage.setImageResource(R.drawable.shape);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

menu.add(Menu.NONE, R.anim.alpha, Menu.NONE, "Alpha")

.setAlphabeticShortcut('a');

menu.add(Menu.NONE, R.anim.scale, Menu.NONE, "Scale")

.setAlphabeticShortcut('s');

menu.add(Menu.NONE, R.anim.translate, Menu.NONE, "Translate")

.setAlphabeticShortcut('t');

menu.add(Menu.NONE, R.anim.rotate, Menu.NONE, "Rotate")

.setAlphabeticShortcut('r');

menu.add(Menu.NONE, R.anim.combination, Menu.NONE, "Комбинация")

.setAlphabeticShortcut('o');

return(super.onCreateOptionsMenu(menu));

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

animation = AnimationUtils.loadAnimation(this, item.getItemId());

animation.setAnimationListener(this);

mImage.startAnimation(animation);

return true;

}

@Override

public void onAnimationEnd(Animation animation) {

mImage.setVisibility(View.VISIBLE);

}

@Override

public void onAnimationRepeat(Animation animation) {

mImage.setVisibility(View.VISIBLE);

}

@Override

public void onAnimationStart(Animation animation) {

mImage.setVisibility(View.VISIBLE);

}

}

**Анимация графических файлов**

Анимация для графических файлов ничем особым не отличается от анимации для графических фигур. Рассмотрим на примере анимацию графического объекта, отображаемого в ImageView. Создайте новый проект и найдите какой-нибудь графический файл с изображением.

В XML-файле анимации создадим следующую структуру: используем элемент <scale> для растягивания изображения и вложенный контейнер <set>, в котором определим два дочерних элемента <scale> и <rotate> для одновременного вращения и изменения размеров объекта. Данный файл необходимо сохраним в каталоге **res/anim/** под любым именем, например, crazycat.xml

**crazycat.xml**

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

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

android:shareInterpolator="false" >

<scale

android:duration="3000"

android:fillAfter="false"

android:fromXScale="1.0"

android:fromYScale="1.0"

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

android:pivotX="50%"

android:pivotY="50%"

android:toXScale="1.4"

android:toYScale="0.6" />

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

<scale

android:duration="2000"

android:fillBefore="false"

android:fromXScale="1.4"

android:fromYScale="0.6"

android:pivotX="50%"

android:pivotY="50%"

android:startOffset="3000"

android:toXScale="0.0"

android:toYScale="0.0" />

<rotate

android:duration="2000"

android:fromDegrees="0"

android:pivotX="50%"

android:pivotY="50%"

android:startOffset="3000"

android:toDegrees="-45"

android:toYScale="0.0" />

</set>

</set>

В файле разметки поместим кнопку для запуска анимации и один элемент ImageView для нашего изображения.

<?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="fill\_parent">

<LinearLayout

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_width="fill\_parent">

<Button

android:id="@+id/btn\_start"

android:layout\_height="wrap\_content"

android:text="Поехали!"

android:layout\_width="fill\_parent"

android:layout\_weight="1"/>

</LinearLayout>

<LinearLayout

android:id="@+id/layout\_anim"

android:layout\_height="fill\_parent"

android:gravity="center"

android:layout\_width="fill\_parent">

<ImageView

android:id="@+id/image"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:minHeight="100px"

android:minWidth="100px"

android:layout\_margin="100px"

android:src="@drawable/android3d"/>

</LinearLayout>

</LinearLayout>

Осталось написать программный код:

package ru.alexanderklimov.crazycat;

import android.app.Activity;

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 CrazyCatActivity extends Activity {

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

final ImageView image = (ImageView) findViewById(R.id.image);

final Button btnStart = (Button) findViewById(R.id.btn\_start);

btnStart.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

Animation anim = AnimationUtils.loadAnimation(

getApplicationContext(), R.anim.crazycat);

image.startAnimation(anim);

}

});

}

}

При нажатии кнопки изображение кота сначала плавно растянется по горизонтали, затем одновременно повернется и уменьшится в размерах, после чего вернется в исходное состояние.



Независимо оттого, как анимация изменяет размеры объекта или перемещает его на плоскости, границы элемента View, в который загружено изображение, останутся неизменными: ваша анимация не будет автоматически корректировать размеры представления для размещения объекта. Если анимация выйдет за границы родительского представления, произойдет отсечение объекта анимации.

**Анимация группы представлений**

Анимацию можно сделать и для нескольких представлений, объединив их в группу. Например, поместив представления в контейнер LinearLayout, причем можно использовать не только графические, но и текстовые представления.

Принцип анимации останется тем же. В файле разметки приложения разместите дочерний контейнер LinearLayout, в котором разместите виджет ImageView с изображением и текстовое поле TextView с надписью. Для дочернего контейнера обязательно присвойте идентификатор, по которому вы сможете загрузить его в код программы и использовать для анимации.

<LinearLayout

android:id="@+id/layout\_anim"

android:layout\_height="fill\_parent"

android:gravity="center"

android:layout\_width="fill\_parent">

<ImageView

android:id="@+id/image"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:src="@drawable/crazycat"/>

<TextView

android:id="@+id/text"

android:text="@string/hello"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"/>

</LinearLayout>

Код класса практически не будет отличаться от предыдущего примера, за исключением того, что мы работаем с анимацией не отдельного представления, а с анимацией группы представлений:

layout (LinearLayout)findViewById(R.id.layout\_anim);

animation AnimationUtils.loadAnimation(this, R.anim.combination);