**ScaleAnimation и <scale>**

[Тег <scale>](http://developer.alexanderklimov.ru/android/animation/scaleanimation.php#scale)  
[Класс ScaleAnimation](http://developer.alexanderklimov.ru/android/animation/scaleanimation.php#scaleanimation)

Тег **scale** и соответствующий ему класс **ScaleAnimation** служат для анимации размеров объекта.

Тег <scale>

Используйте атрибуты **scaleHeight** и **scaleWidth** внутри тега **<scale>**, чтобы описать высоту и ширину относительно границ оригинального объекта **Drawable**. Используйте атрибут **scaleGravity** для изменения опорной точки масштабированного изображения.

Рассмотрим пример масштабирования. Создадим проект, в котором элемент **ImageView** будет пульсировать - постоянно увеличиваться и уменьшаться в размерах. В папке **res/anim** создадим два файла:

res/anim/enlarge.xml

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

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

<scale

android:duration="1000"

android:fromXScale="0.1"

android:fromYScale="0.1"

android:pivotX="50%"

android:pivotY="50%"

android:startOffset="100"

android:toXScale="3.0"

android:toYScale="3.0" />

</set>

shrink.xml

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

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

<scale

android:duration="1000"

android:fromXScale="3.0"

android:fromYScale="3.0"

android:pivotX="50%"

android:pivotY="50%"

android:startOffset="100"

android:toXScale="0.1"

android:toYScale="0.1" />

</set>

Разместите на экране элемент **ImageView** и добавьте код:

package ru.alexanderklimov.testapp;

import android.app.Activity;

import android.os.Bundle;

import android.view.animation.Animation;

import android.view.animation.Animation.AnimationListener;

import android.view.animation.AnimationUtils;

import android.widget.ImageView;

public class MainActivity extends ActionBarActivity {

private ImageView mImageView;

private Animation mEnlargeAnimation, mShrinkAnimation;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mImageView = (ImageView) findViewById(R.id.imageView);

// подключаем файлы анимации

mEnlargeAnimation = AnimationUtils.loadAnimation(this, R.anim.enlarge);

mShrinkAnimation = AnimationUtils.loadAnimation(this, R.anim.shrink);

mEnlargeAnimation.setAnimationListener(animationEnlargeListener);

mShrinkAnimation.setAnimationListener(animationShrinkListener);

// при запуске начинаем с эффекта увеличения

mImageView.startAnimation(mEnlargeAnimation);

}

@Override

protected void onPause() {

super.onPause();

mImageView.clearAnimation();

}

Animation.AnimationListener animationEnlargeListener = new Animation.AnimationListener() {

@Override

public void onAnimationEnd(Animation animation) {

// когда анимация увеличения заканчивается,

// то запускаем анимацию уменьшения

mImageView.startAnimation(mShrinkAnimation);

}

@Override

public void onAnimationRepeat(Animation animation) {

// TODO Auto-generated method stub

}

@Override

public void onAnimationStart(Animation animation) {

// TODO Auto-generated method stub

}

};

Animation.AnimationListener animationShrinkListener = new Animation.AnimationListener() {

@Override

public void onAnimationEnd(Animation animation) {

// когда анимация уменьшения заканчивается,

// то начинаем анимацию увеличения

mImageView.startAnimation(mEnlargeAnimation);

}

@Override

public void onAnimationRepeat(Animation animation) {

// TODO Auto-generated method stub

}

@Override

public void onAnimationStart(Animation animation) {

// TODO Auto-generated method stub

}

};

}

Запускайте проект и любуйтесь эффектом.

**Упрощаем код**

На самом деле мы написали очень много лишнего кода. Если посмотреть внимательно на созданные нами файлы анимации, то увидим, что они содержат одинаковые параметры, которые меняются в обратном порядке - размеры увеличиваются или уменьшаются. Для подобных случаев, когда используются одинаковые значения для анимации, предусмотрены атрибуты **repeatMode** и **repeatCount**. Первый атрибут позволяет сменить направление анимации, а второй - задаёт число анимаций (**infinite** для бесконечности или любое число больше нуля).

Упростим код предыдущего примера. Удалим файл **shrink.xml**. В файле **enlarge.xml** добавим пару новых атрибутов:

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

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

<scale

android:duration="1000"

android:fromXScale="0.1"

android:fromYScale="0.1"

android:pivotX="50%"

android:pivotY="50%"

android:repeatCount="infinite"

android:repeatMode="reverse"

android:startOffset="100"

android:toXScale="3.0"

android:toYScale="3.0" />

</set>

В коде нам уже не понадобятся слушатели анимаций и загрузка второго файла анимации. Код станет намного короче. Судите сами:

package ru.alexanderklimov.testapp;

import android.app.Activity;

import android.os.Bundle;

import android.view.animation.Animation;

import android.view.animation.AnimationUtils;

import android.widget.ImageView;

public class MainActivity extends ActionBarActivity {

private ImageView mImageView;

private Animation mEnlargeAnimation;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mImageView = (ImageView) findViewById(R.id.imageView);

// подключаем файл анимации

mEnlargeAnimation = AnimationUtils.loadAnimation(this, R.anim.enlarge);

// при запуске начинаем с эффекта увеличения

mImageView.startAnimation(mEnlargeAnimation);

}

@Override

protected void onPause() {

super.onPause();

mImageView.clearAnimation();

}

}

**Атрибуты fillBefore, fillAfter и fillEnabled**

Возможно, в некоторых примерах вам будут попадаться атрибуты анимации **fillBefore**, **fillAfter** **fillEnabled** или их программные аналоги - методы **setFillAfter()**, **setFillBefore()**, **setFillEnabled()**.

С помощью данных атрибутов можно указать, когда следует проводить трансформацию. Так **fillAfter** показывает, что финальная трансформация происходит после окончания анимации, а **fillBefore** показывает, что начальная трансформация происходит до начала анимации.

Это может быть полезным в некоторых случаях. Как это работает, лучше смотреть на живых примерах.

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

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

<scale

android:duration="1000"

android:fromXScale="0.1"

android:fromYScale="0.1"

android:pivotX="50%"

android:pivotY="50%"

android:repeatCount="infinite"

android:repeatMode="reverse"

android:startOffset="100"

android:toXScale="3.0"

android:toYScale="3.0"

**android:fillAfter="true"**

**android:fillBefore="false"**

**android:fillEnabled="true"/>**

</set>

Программный способ.

ScaleAnimation scaleAnimation = new ScaleAnimation(0.0f, 1.0f, 0.0f, 1.0f);

scaleAnimation.setFillAfter(true);

scaleAnimation.setFillBefore(false);

scaleAnimation.setFillEnabled(true);

Также применимо к набору анимаций:

<set

android:shareInterpolator="true"

**android:fillBefore="true"**

**android:fillAfter="false">**

<translate

**android:fillEnabled="true"**

android:duration="300"

android:fromXDelta="-100"

android:fromYDelta="0"

android:toXDelta="100"

android:toYDelta="0" />

<translate

android:duration="300"

android:startOffset="300"

android:fromXDelta="0"

android:fromYDelta="-100"

android:toXDelta="0"

android:toYDelta="100" />

</set>

**Класс ScaleAnimation**

Класс **ScaleAnimation** наследуется от класса **Animation** и отвечает за анимацию изменения размеров.

**Орёл или решка**

Напишем пример анимации **flip** - переворачивания изображения на другую сторону.

Добавим на экран **ImageView**, который будет содержать одну из сторон монеты.

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

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

android:layout\_width="fill\_parent"

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

<ImageView

android:id="@+id/flip\_image"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:onClick="onClick" />

</RelativeLayout>

Напишем код переворачивания монетки:

package ru.alexanderklimov.coins;

import ...

public class TestActivity extends Activity {

boolean isHeads;

ScaleAnimation shrink, grow;

ImageView flipImage;

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

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_test);

flipImage = (ImageView) findViewById(R.id.flip\_image);

flipImage.setImageResource(R.drawable.moneycat1);

isHeads = true;

// подключаем анимацию сжатия

shrink = new ScaleAnimation(1.0f, 0.0f, 1.0f, 1.0f,

ScaleAnimation.RELATIVE\_TO\_SELF, 0.5f,

ScaleAnimation.RELATIVE\_TO\_SELF, 0.5f);

shrink.setDuration(500);

shrink.setAnimationListener(new Animation.AnimationListener() {

@Override

public void onAnimationStart(Animation animation) {

}

@Override

public void onAnimationRepeat(Animation animation) {

}

// когда анимация сжатия завершится

@Override

public void onAnimationEnd(Animation animation) {

if (isHeads) {

isHeads = false;

flipImage.setImageResource(R.drawable.moneycat1);

} else {

isHeads = true;

flipImage.setImageResource(R.drawable.monecat2);

}

// запускаем анимацию расширения

flipImage.startAnimation(grow);

}

});

// подключаем анимацию расширения

grow = new ScaleAnimation(0.0f, 1.0f, 1.0f, 1.0f,

ScaleAnimation.RELATIVE\_TO\_SELF, 0.5f,

ScaleAnimation.RELATIVE\_TO\_SELF, 0.5f);

grow.setDuration(500);

}

// касаемся ImageView

public void onClick(View v) {

flipImage.startAnimation(shrink);

}

}

Сжатие происходит с полного размера до 0 от краёв к центру. Расширение использует обратный процесс.

Запустив проект, мы можем наблюдать распространённый эффект вращения монеты (flip), когда видим обратную сторону. Для круглых изображений эффект выглядит достаточно приемлемо. Для прямоугольных изображений лучше использовать анимацию при помощи [ObjectAnimator](http://developer.alexanderklimov.ru/android/animation/objectanimator.php), который использует перспективу для более реалистичного эффекта.

**Через XML**

Чтобы воспроизвести этот же эффект через XML, создадим два файла в папке **res/anim**:

shrink.xml

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

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

<scale

android:duration="500"

android:fromXScale="1.0"

android:fromYScale="1.0"

android:pivotX="50%"

android:pivotY="50%"

android:toXScale="0.0"

android:toYScale="1.0" />

<translate

android:duration="500"

android:fromXDelta="0%"

android:fromYDelta="0%"

android:toXDelta="0%"

android:toYDelta="50%" />

</set>

grow.xml

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

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

<scale

android:duration="150"

android:fromXScale="0.0"

android:fromYScale="1.0"

android:pivotX="50%"

android:pivotY="50%"

android:toXScale="1.0"

android:toYScale="1.0" />

<translate

android:duration="150"

android:fromXDelta="0%"

android:fromYDelta="50%"

android:toXDelta="0%"

android:toYDelta="0%" />

</set>

Разметку оставим прежней, только напишем другой код:

public class FlipCoins extends Activity {

boolean isHeads;

Animation shrink, grow;

ImageView flipImage;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

flipImage = (ImageView)findViewById(R.id.flip\_image);

flipImage.setImageResource(R.drawable.heads);

isHeads = true;

shrink = AnimationUtils.loadAnimation(this, R.anim.shrink);

shrink.setAnimationListener(new Animation.AnimationListener() {

@Override

public void onAnimationStart(Animation animation) {}

@Override

public void onAnimationRepeat(Animation animation) {}

@Override

public void onAnimationEnd(Animation animation) {

if(isHeads) {

isHeads = false;

flipImage.setImageResource(R.drawable.tails);

} else {

isHeads = true;

flipImage.setImageResource(R.drawable.heads);

}

flipImage.startAnimation(grow);

}

});

grow = AnimationUtils.loadAnimation(this, R.anim.grow);

}

public void onClick(View v) {

flipImage.startAnimation(shrink);

}

}

На скриншоте представлены две стороны монеты. Если у вас есть такая монета, то свяжитесь со мной. Я хочу такую монету.

 