**ObjectAnimator**

[Анимация перемещения - translationX](http://developer.alexanderklimov.ru/android/animation/objectanimator.php#translate)  
[Сделаем кнопку прозрачной - alpha](http://developer.alexanderklimov.ru/android/animation/objectanimator.php#alpha)  
[Переворачиваем ListView - rotationY](http://developer.alexanderklimov.ru/android/animation/objectanimator.php#rotationy)  
[Переворачиваемся снова - rotationX](http://developer.alexanderklimov.ru/android/animation/objectanimator.php#rotationx)  
[С ног на голову - rotation](http://developer.alexanderklimov.ru/android/animation/objectanimator.php#rotation)  
[Изменяем фон - backgroundColor](http://developer.alexanderklimov.ru/android/animation/objectanimator.php#background)  
[Набор анимаций AnimatorSet](http://developer.alexanderklimov.ru/android/animation/objectanimator.php#animatorset)

В Android 3.0 появился новый класс **ObjectAnimator**.

Для анимации используются строковые значения, например, **"translationX"**. В Android 4.0 появилась возможность использовать константы класса **View**, которые работают быстрее, так как не используют рефлексию.

Вы можете использовать следующие значения **ALPHA, ROTATION, ROTATION\_X, ROTATION\_Y, SCALE\_X, SCALE\_Y, TRANSLATION\_X, TRANSLATION\_Y, TRANSLATION\_Z, X, Y, Z**.

Класс также содержит методы для управления длительностью анимацией и повторением.

ObjectAnimator scaleAnimator = ObjectAnimator.ofFloat(tvLabel, "scaleX", 1.0f, 2.0f);

scaleAnimator.**setDuration(3000)**; // 3 секунды

scaleAnimator.**setRepeatCount(ValueAnimator.INFINITE)**; // бесконечно

scaleAnimator.**setRepeatMode(ValueAnimator.REVERSE)**; // повторить анимацию в обратном порядке

scaleAnimator.start();

Поддерживаются интерполяторы:

animator.**setInterpolator(new BounceInterpolator())**;

Слушатель **AnimatorListenerAdapter** позволяет отслеживать состояния анимации с помощью методов **onAnimationStart()** и **onAnimationEnd()**.

Группа анимаций управляется через **AnimatorSet**.

Анимацию также можно задавать через XML. Обратите внимание, что для **ObjectAnimator** папка анимации называется **res/animator** вместо **res/anim**. Пример использования можно найти в статье [Управляй мечтой!](http://developer.alexanderklimov.ru/android/animation/steeryourdream.php) и ниже в этой статье.

Анимация перемещения - translationX

Класс **ObjectAnimator** позволяет работать с объектом без потери его функциональности. Покажу на примере анимации двух кнопок. Заставим первую кнопку переместиться на экране с помощью нового класса **ObjectAnimator**, а вторую кнопку переместим с помощью класса **TranslateAnimation**.

activity\_main.xml

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical" >

<Button

android:id="@+id/animatorbutton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="onClick"

android:text="Первая кнопка" />

<Button

android:id="@+id/animationbutton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="onClick"

android:text="Вторая кнопка" />

</LinearLayout>

Код не сложный.

package ru.alexanderklimov.testapp\_1098;

import android.animation.ObjectAnimator;

import android.app.Activity;

import android.os.Bundle;

import android.view.View;

import android.view.animation.Animation;

import android.view.animation.AnimationSet;

import android.view.animation.TranslateAnimation;

import android.widget.Button;

import android.widget.Toast;

public class TestAppActivity extends Activity {

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

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

Button animatorButton = (Button) findViewById(R.id.animatorbutton);

Button animationButton = (Button) findViewById(R.id.animationbutton);

// анимация для первой кнопки

ObjectAnimator objectAnimatorButton = ObjectAnimator.ofFloat(

animatorButton, "translationX", 0f, 400f);

objectAnimatorButton.setDuration(1000);

objectAnimatorButton.start();

// анимация для второй кнопки

AnimationSet animSetAnimationButton = new AnimationSet(true);

TranslateAnimation translateAnimAnimationButton = new TranslateAnimation(

Animation.RELATIVE\_TO\_SELF, 0, Animation.RELATIVE\_TO\_SELF, 1f,

Animation.RELATIVE\_TO\_SELF, 0, Animation.RELATIVE\_TO\_SELF, 0);

animSetAnimationButton.addAnimation(translateAnimAnimationButton);

animSetAnimationButton.setDuration(500);

animSetAnimationButton.setFillAfter(true);

animationButton.setAnimation(animSetAnimationButton);

}

public void onClick(View v) {

switch (v.getId()) {

case R.id.animatorbutton:

Toast.makeText(getApplicationContext(),

"Вы щелкнули на первой кнопке (Animator)",

Toast.LENGTH\_SHORT).show();

break;

case R.id.animationbutton:

Toast.makeText(getApplicationContext(),

"Вы щелкнули на второй кнопке (Animation)",

Toast.LENGTH\_SHORT).show();

break;

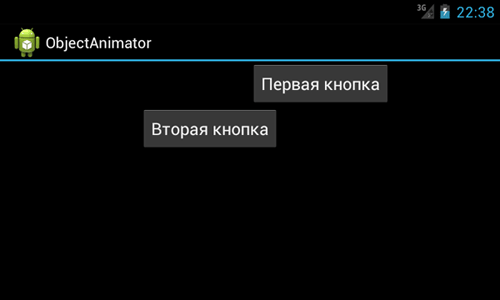
}

}

}

Когда вы запустите пример, то обе кнопки переместятся вправо, используя различные способы анимации. Теперь попробуйте щёлкнуть на этих кнопках. С первой кнопкой проблем не будет - при нажатии появится всплывающее сообщение. А вот вторая кнопка реагировать на нажатия не будет. Теперь попробуйте щелкнуть левее от второй кнопки на черном экране приложения - появится сообщение, что вы щёлкнули на второй кнопке! Магия! Получается, что система "видит" кнопку в том месте, которая была определена свойствами кнопки и никак не учитывает ёё перемещение в результате анимации **translate**. Таким образом вы наглядно видите разницу между двумя анимациями.

В примере использовалась программная реализация анимации **ObjectAnimator**, но также существует возможность использования XML-файла. Читайте документацию.



Для анимации мы использовали метод **ObjectAnimator.ofFloat()** и указали тип анимации в строковом параметре **translationX**, в последних параметрах мы указываем начальное и конечное значение для анимации.

Также есть родственный метод **ofInt()**, а типы анимаций могут быть самыми разными.

Сделаем кнопку прозрачной - alpha

Например, мы хотим сделать кнопку прозрачной:

public void onClick(View view) {

ObjectAnimator animator = ObjectAnimator.ofFloat(buttonSend, "alpha", 0);

animator.addListener(new AnimatorListenerAdapter() {

public void onAnimationEnd(Animator animation) {

buttonSend.setAlpha(1);

buttonSend.setVisibility(View.INVISIBLE);

}

});

animator.start();

}

// вернем видимость

public void onClick2(View view) {

buttonSend.setAlpha(0);

buttonSend.setVisibility(View.VISIBLE);

buttonSend.animate().alpha(1);

}

Можно обойтись без слушателя **AnimatorListenerAdapter**. Сделаем компонент **ImageView** наполовину прозрачным, используя метод **ofInt()**:

ObjectAnimator animator = ObjectAnimator.ofInt(mImageView, "alpha", 255, 128);

animator.setDuration(2000);

animator.start();

Можно указывать и промежуточные значения. Укажем три значения.

ObjectAnimator animator = ObjectAnimator.ofInt(mImageView, "alpha", 255, 128, 255);

Теперь анимация дойдёт до половины прозрачности и вернётся к полной непрозрачности. Таким образом можно создавать пульсирующие изображения.

Напишем ещё один метод для исчезновения Чеширского кота.

public void hideCat(View view) {

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

ObjectAnimator animator = ObjectAnimator.ofFloat(imageView, "alpha", 1.0f,

0.25f, 0.75f, 0.15f, 0.5f, 0.0f);

animator.setDuration(5000);

animator.start();

}

Мы указали тип анимации **alpha**, но при этом создали массив значений для переходных состояний. Поэтому картинка будет исчезать не плавно, а пульсируя.

К анимации можно применить [интерполятор](http://developer.alexanderklimov.ru/android/animation/interpolator.php) через метод **setInterpolator()**. По умолчанию используется **AccelerateDecelerateInterpolator**. Зададим свой:

animator.setInterpolator(new LinearInterpolator());

В Android 4.0 появилась более быстрая анимация через использование свойств компонента.

ObjectAnimator fadeAltAnimator = ObjectAnimator.ofFloat(image, **View.ALPHA**, 0, 1);

fadeAltAnimator.start();

Flip ListView - rotationY

В приложении API Demos есть есть пример анимации двух **ListView**, когда один список выводится на экран, а второй список невидим. При нажатии на кнопку происходит анимация переключения между двумя списками. Демонстрацию можно увидеть в эмуляторе в разделе **Animation/View Flip**.

Разметка

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

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

android:orientation="vertical"

android:layout\_width="match\_parent"

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

<Button

android:id="@+id/button"

android:text="Flip"

android:layout\_width="wrap\_content"

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

<ListView

android:id="@+id/list\_en"

android:layout\_width="match\_parent"

android:layout\_weight="1.0"

android:layout\_height="0dip"/>

<ListView

android:id="@+id/list\_fr"

android:layout\_width="match\_parent"

android:layout\_weight="1.0"

android:layout\_height="0dip"

android:visibility="gone"/>

</LinearLayout>

Код

package example.FlipViewExample;

import android.animation.Animator;

import android.animation.AnimatorListenerAdapter;

import android.animation.ObjectAnimator;

import android.app.Activity;

import android.os.Bundle;

import android.view.View;

import android.view.animation.AccelerateInterpolator;

import android.view.animation.DecelerateInterpolator;

import android.view.animation.Interpolator;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.ListView;

public class FlipViewExamplebrush extends Activity {

private static final String[] LIST\_STRINGS\_EN = new String[] {

"One",

"Two",

"Three",

"Four",

"Five",

"Six"

};

private static final String[] LIST\_STRINGS\_FR = new String[] {

"Un",

"Deux",

"Trois",

"Quatre",

"Le Five",

"Six"

};

ListView mEnglishList;

ListView mFrenchList;

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

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

mEnglishList = (ListView) findViewById(R.id.list\_en);

mFrenchList = (ListView) findViewById(R.id.list\_fr);

// Prepare the ListView

final ArrayAdapter<String> adapterEn = new ArrayAdapter<String>(this,

android.R.layout.simple\_list\_item\_1, LIST\_STRINGS\_EN);

// Prepare the ListView

final ArrayAdapter<String> adapterFr = new ArrayAdapter<String>(this,

android.R.layout.simple\_list\_item\_1, LIST\_STRINGS\_FR);

mEnglishList.setAdapter(adapterEn);

mFrenchList.setAdapter(adapterFr);

mFrenchList.setRotationY(-90f);

Button starter = (Button) findViewById(R.id.button);

starter.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

flipit();

}

});

}

private Interpolator accelerator = new AccelerateInterpolator();

private Interpolator decelerator = new DecelerateInterpolator();

private void flipit() {

final ListView visibleList;

final ListView invisibleList;

if (mEnglishList.getVisibility() == View.GONE) {

visibleList = mFrenchList;

invisibleList = mEnglishList;

} else {

invisibleList = mFrenchList;

visibleList = mEnglishList;

}

ObjectAnimator visToInvis = ObjectAnimator.ofFloat(visibleList, "rotationY", 0f, 90f);

visToInvis.setDuration(500);

visToInvis.setInterpolator(accelerator);

final ObjectAnimator invisToVis = ObjectAnimator.ofFloat(invisibleList, "rotationY",

-90f, 0f);

invisToVis.setDuration(500);

invisToVis.setInterpolator(decelerator);

visToInvis.addListener(new AnimatorListenerAdapter() {

@Override

public void onAnimationEnd(Animator anim) {

visibleList.setVisibility(View.GONE);

invisToVis.start();

invisibleList.setVisibility(View.VISIBLE);

}

});

visToInvis.start();

}

}

Упрощённый вариант с переворачиванием покажу на примере обычной кнопки. Не будем прибегать к вызовам анимации, а просто напишем одну строчку кода:

public void onClick(View v) {

final Button rotatingButton = (Button) findViewById(R.id.rotatingButton);

ObjectAnimator.ofFloat(rotatingButton, "rotationY", 0, 360).setDuration(3000).start();

}

Здесь мы также увидим эффект флипа без настройки анимационных эффектов, будет простая анимация переворачивания, вполне пригодная для использования.

Можно создать универсальный метод для применения анимации у нужного компонента.

public void flipOnVertical(View view) {

View image = findViewById(R.id.imageView);

ObjectAnimator animator = ObjectAnimator.ofFloat(image, "rotationY", 0.0f,

360.0f);

animator.setDuration(500);

animator.start();

}

Статичный метод **ofFloat()** возвращает объект **ObjectAnimator**. В первом параметре метода нужно указать объект, к которому будет применена анимация. Во втором указывается характер анимации. В данном случае указан поворот вокруг оси Y. Последние два параметра задают начальное и конечные значения анимации. Наша картинка сделает полный оборот и вернётся в исходное состояние.

Орёл или решка (новая версия)

В статье про [ScaleAnimation](http://developer.alexanderklimov.ru/android/animation/scaleanimation.php) был показан пример переворачивания монеты. Перепишем код с использованием **ObjectAnimator**. Разметку можно оставить прежнюю:

<?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>

Код будет следующим:

package ru.alexanderklimov.flipcoins;

import ...

public class MainActivity extends Activity {

boolean isHeads;

ObjectAnimator flipper, flipper2;

Bitmap headsImage, tailsImage;

ImageView flipImage;

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

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_test);

setTitle("ScaleAnimation");

headsImage = BitmapFactory.decodeResource(getResources(),

R.drawable.moneycat1);

tailsImage = BitmapFactory.decodeResource(getResources(),

R.drawable.moneycat3);

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

flipImage.setImageBitmap(headsImage);

isHeads = true;

flipper = ObjectAnimator.ofFloat(flipImage, "rotationY", 0f, 180f);

flipper.setDuration(3000);

flipper.addUpdateListener(new AnimatorUpdateListener() {

@Override

public void onAnimationUpdate(ValueAnimator animation) {

if (animation.getAnimatedFraction() >= 0.50f && isHeads) {

flipImage.setImageBitmap(tailsImage);

isHeads = false;

}

}

});

flipper2 = ObjectAnimator.ofFloat(flipImage, "rotationY", -180f, 0f);

flipper2.setDuration(3000);

flipper2.addUpdateListener(new AnimatorUpdateListener() {

@Override

public void onAnimationUpdate(ValueAnimator animation) {

if (animation.getAnimatedFraction() >= 0.50f && !isHeads) {

flipImage.setImageBitmap(headsImage);

isHeads = true;

}

}

});

}

public void onClick(View v) {

if(isHeads){

flipper.start();

}

else{

flipper2.start();

}

}

}

Я создал две отдельные анимации вращения на 180 градусов. Единственное, что мне пришлось изменить - поменять изображение обратной стороны монеты на зеркальную копию, так как во время работы приложения ImageView переворачивается и картинка оказывается перевёрнутой. Поэтому на экране мы увидим картинку в нормальном виде. А также показал реализацию слушателя при анимации.

Переворачиваемся снова - rotationX

Точно также можно переворачивать объекты по оси X. Добавим на экран несколько элементов и каждому присвоим обработку щелчка.

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

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/linearLayout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

android:orientation="vertical"

tools:context=".MainActivity">

<TextView

android:id="@+id/textView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Коты всегда приземляются на лапы"

android:textSize="28dp"

android:textStyle="bold" />

<Button

android:id="@+id/button"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Перевернуть"

android:layout\_marginTop="16dp" />

<ImageView

android:id="@+id/imageView"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

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

android:layout\_gravity="center\_horizontal"

android:layout\_marginTop="16dp" />

</LinearLayout>

Код для класса активности.

// Если этот код работает, его написал Александр Климов,

// а если нет, то не знаю, кто его писал.

package ru.alexanderklimov.testapplication;

import android.animation.ObjectAnimator;

import android.os.Bundle;

import android.support.v7.app.ActionBarActivity;

import android.view.View;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.LinearLayout;

import android.widget.TextView;

public class MainActivity extends ActionBarActivity {

private LinearLayout mLinearLayout;

private TextView mTextView;

private Button mButton;

private ImageView mImageView;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mLinearLayout = (LinearLayout) findViewById(R.id.linearLayout);

mLinearLayout.setOnClickListener(onClickListener);

mTextView = (TextView) findViewById(R.id.textView);

mTextView.setOnClickListener(onClickListener);

mButton = (Button) findViewById(R.id.button);

mButton.setOnClickListener(onClickListener);

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

mImageView.setOnClickListener(onClickListener);

}

View.OnClickListener onClickListener = new View.OnClickListener() {

@Override

public void onClick(View view) {

ObjectAnimator flipAnimator = ObjectAnimator.ofFloat(view, "rotationX", 0f, 360f);

flipAnimator.setDuration(3000);

flipAnimator.start();

}

};

}

С ног на голову - rotation

Если использовать свойство **rotation**, то вращение будет вокруг своей оси. В предыдущем пример достаточно убрать один символ. Но я ещё изменил поворот на 180 градусов, чтобы оправдать заголовок.

View.OnClickListener onClickListener = new View.OnClickListener() {

@Override

public void onClick(View view) {

ObjectAnimator rotateAnimator = ObjectAnimator.ofFloat(view, "rotation", 0f, 180f);

rotateAnimator.setDuration(3000);

rotateAnimator.start();

}

};

Изменяем фон - backgroundColor

Можно менять фон. Добавим **ImageView** и будем менять его фон динамически через свойство **backgroundColor**.

public void onClick(View v) {

int colorStart = 0xFFFF8080;

int colorEnd = 0xFF4080FF;

ValueAnimator animator = ObjectAnimator.ofInt(

mImageView, "backgroundColor", colorStart, colorEnd);

animator.setDuration(3000);

animator.setEvaluator(new ArgbEvaluator());

animator.setRepeatCount(ValueAnimator.INFINITE);

animator.setRepeatMode(ValueAnimator.REVERSE);

animator.start();

}

Нажав на кнопку, вы увидите, как фон за изображением будет постоянно меняться между двумя определёнными значениями. Класс **ValueAnimator**является подмножеством **ObjectAnimator**.

Этот же эффект через XML. Создадим файл **res/animator/background\_animator.xml**:

<set>

<objectAnimator

android:propertyName="backgroundColor"

android:duration="3000"

android:valueFrom="#FFFF8080"

android:valueTo="#FF4080FF"

android:repeatCount="-1"

android:repeatMode="reverse" />

</set>

Код:

AnimatorSet set = (AnimatorSet) AnimatorInflater.loadAnimator(this,

R.anim.background\_animator);

set.setTarget(mImageView);

set.start();

Набор анимаций AnimatorSet

Мы можем управлять сложной анимацией, которая должна выполняться одновременно или последовательно при помощи класса **AnimatorSet.Builder** с использованием **AnimatorSet**.

Создадим пример движения **ImageView** вдоль воображаемого треугольника. Простейшая разметка.

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity" >

<ImageView

android:id="@+id/imageView"

android:layout\_width="100dp"

android:layout\_height="100dp"

android:layout\_marginLeft="20dp"

android:layout\_marginTop="20dp"

android:cropToPadding="true"

android:onClick="onClick"

android:src="@drawable/ic\_android\_cat" />

</FrameLayout>

Мы задали размеры **ImageView** и разместили его в левом верхнем углу. Для определения щелчка определили атрибут **onClick**. Допустим, мы хотим сдвинуть компонент вниз из точки с y-координатой 20dp в точку 200dp, затем переместить вправо из точки с x-координатой 20dp в точку 200dp, и, наконец, вернуть компонент обратно в исходную точку, изменив обе координаты одновременно. Последнее движение будет идти по диагонали.

**ObjectAnimator** может работать одновременно только с одним свойством, поэтому для последнего движения по диагонали нам придётся создать две анимационных настройки. Таким образом нам придётся создать в общей сложности четыре объекта **ObjectAnimator** и запустить их через метод **startAnimation()**.

package ru.alexanderklimov.test;

// Если этот код работает, его написал Александр Климов,

// а если нет, то не знаю, кто его писал.

import android.animation.AnimatorSet;

import android.animation.ObjectAnimator;

import android.app.Activity;

import android.os.Bundle;

import android.view.View;

import android.view.animation.AccelerateDecelerateInterpolator;

import android.widget.ImageView;

public class MainActivity extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

public void onClick(View v) {

float scale = getResources().getDisplayMetrics().density;

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

// Двигаемся вниз

ObjectAnimator objectAnimator1 = ObjectAnimator.ofFloat(catImageView, "y",

20.0f \* scale, 220.0f \* scale);

// Двигаемся вперед

ObjectAnimator objectAnimator2 = ObjectAnimator.ofFloat(catImageView, "x",

20.0f \* scale, 220.0f \* scale);

// Возвращаемся обратно

ObjectAnimator objectAnimator3 = ObjectAnimator.ofFloat(catImageView, "x",

220.0f \* scale, 20.0f \* scale);

ObjectAnimator objectAnimator4 = ObjectAnimator.ofFloat(catImageView, "y",

220.0f \* scale, 20.0f \* scale);

AnimatorSet animatorSet = new AnimatorSet();

animatorSet.play(objectAnimator1).before(objectAnimator2);

animatorSet.play(objectAnimator3).after(objectAnimator2);

animatorSet.play(objectAnimator3).with(objectAnimator4);

animatorSet.play(objectAnimator1).after(500);

animatorSet.setDuration(1000);

animatorSet.setInterpolator(new AccelerateDecelerateInterpolator());

animatorSet.start();

}

}

Вначале мы определяем плотность экрана и затем создаём четыре объекта **ObjectAnimator**, задействуя свойство **x** или **y**.

Далее следует объединить созданные объекты в один набор и указать последовательность. Вызывая метод **animatorSet.play()**, мы создаём объект **Builder**, указывая продолжительность и очерёдность.

Первый вызов **animatorSet.play(objectAnimator1)** создаёт объект **Builder** и добавляем анимацию к набору. Затем метод **before(objectAnimator2)**добавляет второй объект анимации и указывает, что первый объект должен проиграть свою анимацию перед вторым объектом. Также есть похожий метод **after()**, который позволяет проиграть анимацию после указанного объекта.

Чтобы две анимации выполнились одновременно, нужно вызвать метод **with()**, который позволил в нашем примере осуществить движение по диагонали.

Строка **animatorSet.play(objectAnimator1).after(500);** задаёт задержку для первого объекта анимации.

Код можно упростить, объединив вызовы методов в цепочку.

animatorSet.play(objectAnimator1).before(objectAnimator2).after(500);

animatorSet.play(objectAnimator3).after(objectAnimator2).with(objectAnimator4);

Подготовив набор, мы задаём продолжительность анимации (одна секунда) для каждого элемента набора (в итоге общая анимация составит три секунды) и запускаем.

Также вы можете указать продолжительность и интерполятор для каждого элемента набора.

objectAnimator1.setDuration(500);

objectAnimator2.setDuration(500);

objectAnimator1.setInterpolator(new AccelerateInterpolator());

objectAnimator2.setInterpolator(new DecelerateInterpolator());

// и т.д.

Стоит отметить интересный эффект, если использовать разные интерполяторы для одновременных анимаций. Например, при движении по диагонали картинка будет двигаться не прямой, по изогнутой кривой. Попробуйте сами.

...

AnimatorSet animatorSet = new AnimatorSet();

animatorSet.play(objectAnimator1).before(objectAnimator2).after(500);

animatorSet.play(objectAnimator3).after(objectAnimator2).with(objectAnimator4);

objectAnimator1.setDuration(500);

objectAnimator2.setDuration(500);

objectAnimator1.setInterpolator(new AccelerateInterpolator());

objectAnimator2.setInterpolator(new DecelerateInterpolator());

objectAnimator3.setDuration(1000);

objectAnimator4.setDuration(1000);

objectAnimator3.setInterpolator(new LinearInterpolator());

objectAnimator4.setInterpolator(new AccelerateDecelerateInterpolator());

animatorSet.start();

Объединять можно анимации и для разных объектов. Допустим, у нас есть два **ImageView** и нам нужно синхронно или последовательно воздействовать на них.

public void hideCats() {

ImageView cat1 = (ImageView) findViewById(R.id.imageView);

ImageView cat2 = (ImageView) findViewById(R.id.ImageView2);

ObjectAnimator animator1 = ObjectAnimator.ofFloat(cat1, "alpha", 1.0f,

1.0f, 0.25f, 0.75f, 0.15f, 0.5f, 0.0f);

ObjectAnimator animator2 = ObjectAnimator.ofFloat(cat2, "alpha", 0.0f,

1.0f, 0.0f, 0.75f, 0.0f, 0.5f, 0.0f);

AnimatorSet animatorSet = new AnimatorSet();

animatorSet.playTogether(animator1, animator2);

animatorSet.setDuration(5000);

animatorSet.setInterpolator(new LinearInterpolator());

animatorSet.start();

}

Анимация для двух изображений запускается одновременно, но у каждого компонента своя индивидуальная анимация.

Дополнительное чтение

[ObjectAnimator.Path.Lollipop](http://developer.alexanderklimov.ru/android/animation/objectanimator_lollipop.php) - в Android 5.0 добавился новый конструктор.

[Squash & Stretch](http://graphics-geek.blogspot.ru/2013/08/devbytes-squash-stretch.html) - кнопка падает, сплющивается, восстанавливает свою форму и возвращается обратно. Отличный пример создания реалистичной анимации. Там же можно скачать исходник.

[Cartoon Animation Techniques](http://graphics-geek.blogspot.ru/2013/08/devbytes-cartoon-animation-techniques.html) - продолжение предыдущего примера с дополнительными фишками. Там же есть исходники и видео.

На сайте Samsung я нашёл красивое приложение с настройками эффектов для ObjectAnimator. Первое текстовое поле поддерживает автодополнение и вы можете написать нужные имена эффектов и сразу же просмотреть анимацию. Кроме **ObjectAnimator**, программа позволяет переключиться в другие виды анимаций и поиграться с настройками.

[Animation Effects](http://developer.samsung.com/android/technical-docs/Animation-Effects). Там же можно скачать исходник.

