**Кадровая анимация - AnimationDrawable   
(<animation-list>)**

[Пример с анимацией](http://developer.alexanderklimov.ru/android/animation/frameanimation.php#sample)  
[Изменяем прозрачность](http://developer.alexanderklimov.ru/android/animation/frameanimation.php#alpha)  
[Анимация при загрузке приложения](http://developer.alexanderklimov.ru/android/animation/frameanimation.php#startup)

Кадровая (фреймовая) анимация — традиционная анимация при помощи быстрой смены последовательных изображений, как на киноплёнке. Данный вид анимации использует XML-файлы в каталоге **res/anim/**.

XML-файл состоит из корневого элемента <animation-list> и дочерних узлов <item>, каждый из которых определяет кадр, имеющий две составляющие:

* графический ресурс для кадра;
* продолжительность кадра;

<animation-list xmlns:android="http://schemas.android.com/apk/res/android"

android:oneshot="true">

<item android:drawable="@drawable/filel" android:duration="200"

<item android:drawable="@drawable/file2" android:duration="200"

<item android:drawable="@drawable/file3" android:duration="200"

</animation-list>

Данная анимация будет выполняться только для трёх кадров. При установке атрибута **android:oneshot** в **true** анимация повторится только один раз и после остановки будет содержать последний кадр. Если же атрибут установить в **false**, то анимация будет циклической. Данный XML-файл, сохраненный в каталоге **res/anim** проекта, можно добавить как фоновое изображение для компонента и затем запустить анимацию.

**Пример с анимацией**

Создадим новый проект и добавим на форму две кнопки для управления анимацией.

<LinearLayout 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"

android:orientation="vertical"

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

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

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

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

tools:context=".MainActivity">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="10dp"

android:orientation="horizontal" >

<Button

android:id="@+id/buttonStart"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="Пуск" />

<Button

android:id="@+id/buttonStop"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="Стоп" />

</LinearLayout>

<ImageView

android:id="@+id/imageView"

android:layout\_width="wrap\_content"

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

</LinearLayout>

Для анимации подготовим заранее отобранные файлы, которые необходимо поместить в каталог **res/drawable**. Наша анимация будет состоять из восьми кадров. Время показа каждого кадра установим в 250 миллисекунд. Запишем наши настройки в XML-файл **catanimation.xml** в каталоге **res/drawable**.

<animation-list

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

android:oneshot="false">

<item android:drawable="@drawable/fitness1" android:duration="250"/>

<item android:drawable="@drawable/fitness2" android:duration="250"/>

<item android:drawable="@drawable/fitness3" android:duration="250"/>

<item android:drawable="@drawable/fitness4" android:duration="250"/>

<item android:drawable="@drawable/fitness5" android:duration="250"/>

<item android:drawable="@drawable/fitness6" android:duration="250"/>

<item android:drawable="@drawable/fitness7" android:duration="250"/>

<item android:drawable="@drawable/fitness8" android:duration="250"/>

</animation-list>

Получить объект **AnimationDrawable** в коде программы можно так:

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

image.setBackgroundResource(R.drawable.catanimation);

AnimationDrawable animation (AnimationDrawable)image.getBackground();

Управлять объектом **AnimationDrawable** можно через методы **start()** и **stop()**.

package ru.alexanderklimov.testapplication;

import android.graphics.drawable.AnimationDrawable;

import android.os.Bundle;

import android.support.v7.app.ActionBarActivity;

import android.view.View;

import android.widget.Button;

import android.widget.ImageView;

public class MainActivity extends ActionBarActivity {

private AnimationDrawable mAnimationDrawable;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

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

imageView.setBackgroundResource(R.drawable.catanimation);

mAnimationDrawable = (AnimationDrawable) imageView.getBackground();

final Button btnStart = (Button) findViewById(R.id.buttonStart);

btnStart.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

mAnimationDrawable.start();

}

});

final Button btnStop = (Button) findViewById(R.id.buttonStop);

btnStop.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

mAnimationDrawable.stop();

}

});

}

}

Запустите приложение и посмотрите на анимацию. Надеюсь, вид кота, делающего упражнения, заставит вас с понедельника начать делать зарядку.



Создание анимации в коде программы

Также можно создавать анимацию в коде — достаточно загрузить последовательно ресурсы кадров и определить время показа для каждого кадра. В качестве примера создадим новый проект, взяв за основу предыдущее упражнение.

В основном классе определим два внутренних метода **startFrameAnimation()** и **stopFrameAnimation()**, которые будем вызывать при нажатии кнопок "Пуск" и "Стоп".

В методе **startFrameAnimation()** реализуем создание анимации. Надо получить кадры анимации в виде набора объектов **Drawable**, загрузив изображения из ресурсов. Для каждого кадра создается отдельный объект **Drawable**:

BitmapDrawable frame1 =

(BitmapDrawable)getResources().getDrawable(R.drawable.fitness1);

BitmapDrawable frame2 =

(BitmapDrawable)getResources().getDrawable(R.drawable.fitness2);

BitmapDrawable frame3

(BitmapDrawable)getResources().getDrawable(R.drawable.fitness3);

// ... и так далее

Созданные объекты **BitmapDrawable** необходимо добавить в объект **AnimationDrawable** методом **addFrame()**, который принимает два параметра: кадр анимации (объект **Drawable**) и продолжительность показа в миллисекундах.

AnimationDrawable mAnimation = new AnimationDrawable();

// устанавливаем циклическое повторение анимации

mAnimation.setOneShot(false);

mAnimation.addFrame(framel, 250);

mAnimation.addFrame(frame2, 250);

mAnimation.addFrame(frame3, 250);

Полный листинг будет следующим:

package ru.alexanderklimov.testapplication;

import android.graphics.drawable.AnimationDrawable;

import android.graphics.drawable.BitmapDrawable;

import android.os.Bundle;

import android.support.v7.app.ActionBarActivity;

import android.view.Menu;

import android.view.View;

import android.widget.Button;

import android.widget.ImageView;

public class MainActivity extends ActionBarActivity {

private AnimationDrawable mAnimationDrawable = null;

private final static int DURATION = 250;

private ImageView mImageView;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

setTitle("AnimationDrawable");

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

final Button btnStart = (Button) findViewById(R.id.buttonStart);

btnStart.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

startFrameAnimation();

}

});

final Button btnStop = (Button) findViewById(R.id.buttonStop);

btnStop.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

stopFrameAnimation();

}

});

}

private void startFrameAnimation() {

BitmapDrawable frame1 = (BitmapDrawable) getResources().getDrawable(

R.drawable.fitness1);

BitmapDrawable frame2 = (BitmapDrawable) getResources().getDrawable(

R.drawable.fitness2);

BitmapDrawable frame3 = (BitmapDrawable) getResources().getDrawable(

R.drawable.fitness3);

BitmapDrawable frame4 = (BitmapDrawable) getResources().getDrawable(

R.drawable.fitness4);

BitmapDrawable frame5 = (BitmapDrawable) getResources().getDrawable(

R.drawable.fitness5);

BitmapDrawable frame6 = (BitmapDrawable) getResources().getDrawable(

R.drawable.fitness6);

BitmapDrawable frame7 = (BitmapDrawable) getResources().getDrawable(

R.drawable.fitness7);

BitmapDrawable frame8 = (BitmapDrawable) getResources().getDrawable(

R.drawable.fitness8);

mAnimationDrawable = new AnimationDrawable();

mAnimationDrawable.setOneShot(false);

mAnimationDrawable.addFrame(frame1, DURATION);

mAnimationDrawable.addFrame(frame2, DURATION);

mAnimationDrawable.addFrame(frame3, DURATION);

mAnimationDrawable.addFrame(frame4, DURATION);

mAnimationDrawable.addFrame(frame5, DURATION);

mAnimationDrawable.addFrame(frame6, DURATION);

mAnimationDrawable.addFrame(frame7, DURATION);

mAnimationDrawable.addFrame(frame8, DURATION);

mImageView.setBackground(mAnimationDrawable);

if (!mAnimationDrawable.isRunning()) {

mAnimationDrawable.setVisible(true, true);

mAnimationDrawable.start();

}

}

private void stopFrameAnimation() {

if (mAnimationDrawable.isRunning()) {

mAnimationDrawable.stop();

mAnimationDrawable.setVisible(false, false);

}

}

}

В методах я также использовал проверку на работающую анимацию (**isRunning()**), хотя и без этой проверки всё работало. Но мало ли что.

Результат будет тот же самый (теоретически). У меня в эмуляторе почему-то работали только первые три кадра. На форумах многие пишут, что у них выводится только первый кадр. Отсюда можно сделать вывод, что не нужно использовать программную анимацию, а использовать анимацию через XML. Статья писалась в 2012 году, возможно сейчас на эмуляторе таких проблем нет.

Изменяем прозрачность

Класс **AnimationDrawable** имеет метод **setAlpha()**, позволяющий изменять прозрачность кадров от 255 до 0. Модифицируем предыдущий пример. Добавим на форму метку и ползунок:

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Set Alpha" />

<SeekBar

android:id="@+id/seekBarAlpha"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="255"

android:progress="255" />

В коде программы добавим слушатель изменения ползунка в методе **onCreate()**:

SeekBar alphaSeekBar = (SeekBar) findViewById(R.id.seekBarAlpha);

alphaSeekBar

.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

@Override

public void onProgressChanged(SeekBar seekBar,

int progress, boolean fromUser) {

mAnimationDrawable.setAlpha(progress);

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

// TODO Auto-generated method stub

}

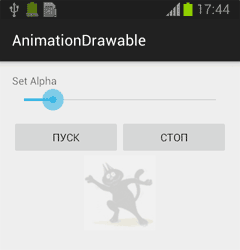
@Override

public void onStopTrackingTouch(SeekBar seekBar) {

// TODO Auto-generated method stub

}

});



Анимация при загрузке приложения

В документации по кадровой анимации имеется предупреждение, что при запуске программы анимация не запустится, так как не все необходимые классы подгружаются в момент старта приложения. В нашем примере мы использовали нажатия кнопок для запуска и остановки анимации, поэтому нам удалось избежать этой проблемы. А что же делать, если нам необходимо запустить анимацию сразу?

В этом случае придется усложнить код. Добавим новый класс:

class Starter implements Runnable {

public void run() {

mFrameAnimation.start();

}

}

А в основном классе из первого примера добавим строчку кода:

image.post(new Starter());

Теперь при запуске приложения анимация автоматически начнет проигрываться.

Также можно попробовать вызвать метод [onWindowFocusChanged()](http://developer.alexanderklimov.ru/android/activity-theory.php" \l "onwindowfocuschanged).

Копирование кадров и показ в обратном порядке

Мы можем получить отдельные кадры анимации и выстроить новую цепочку. Добавим на экран второй компонент **ImageView** и запустим в нём анимацию в обратном порядке, скопировав все кадры из первой анимации.

package ru.alexanderklimov.testapplication;

import ...

public class MainActivity extends ActionBarActivity {

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

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

imageView.setBackgroundResource(R.drawable.catanimation);

ImageView secondImageView = (ImageView) findViewById(R.id.imageView2);

final AnimationDrawable sourceAnimationDrawable

= (AnimationDrawable) imageView.getBackground();

// Копируем кадры в обратном порядке

final AnimationDrawable reversedAnimationDrawable = copyReversedAnimDrawable(sourceAnimationDrawable);

secondImageView.setImageDrawable(reversedAnimationDrawable);

final Button startButton = (Button) findViewById(R.id.buttonStart);

startButton.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

if (!reversedAnimationDrawable.isRunning()) {

sourceAnimationDrawable.start();

reversedAnimationDrawable.start();

}

}

});

final Button stopButton = (Button) findViewById(R.id.buttonStop);

stopButton.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

if (reversedAnimationDrawable.isRunning()) {

sourceAnimationDrawable.stop();

reversedAnimationDrawable.stop();

}

}

});

}

private AnimationDrawable copyReversedAnimDrawable(AnimationDrawable src) {

AnimationDrawable newAnimationDrawable = new AnimationDrawable();

int numberOfFrame = src.getNumberOfFrames();

for (int i = 0; i < numberOfFrame; i++) {

newAnimationDrawable.addFrame(

src.getFrame(numberOfFrame - i - 1),

src.getDuration(numberOfFrame - i - 1));

}

newAnimationDrawable.setOneShot(false);

return newAnimationDrawable;

}

}

На всякий случай предупрежу, что во всех примерах мы использовали фон: **setBackgroundResource()**, **getBackground()** и т.д. Если вы будете использовать атрибут **android:src**, то соответственно используйте **getDrawable()** и другие родственные методы.

У **AnimationDrawable** нет слушателей, позволяющих отслеживать конец анимации. Как вариант, рассчитайте самостоятельно количество кадров и их продолжительность и реализуйте свой способ.

Данный вид анимации подходит для небольших изображений с ограниченным количеством кадров. Если изображения будут большими и кадров слишком много, то наступит переполнение памяти. Помните об этом.