**AlphaAnimation и <alpha>**

Тег **alpha** и соответствующий ему класс **AlphaAnimation** служат для анимации прозрачности объекта.

Рассмотрим пример изменения прозрачности. Создадим проект, в котором элемент **ImageView** будет постоянно исчезать и появляться. В папке **res/anim** создадим два файла:

fadein.xml

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

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

android:shareInterpolator="false" >

<alpha

android:duration="1000"

android:fromAlpha="0.0"

android:toAlpha="1.0" >

</alpha>

</set>

fadeout.xml

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

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

android:shareInterpolator="false" >

<alpha

android:duration="1000"

android:fromAlpha="1.0"

android:toAlpha="0.0" >

</alpha>

</set>

Разместите на экране активности 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 mFadeInAnimation, mFadeOutAnimation;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

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

// подключаем файл анимации

mFadeInAnimation = AnimationUtils.loadAnimation(this, R.anim.fadein);

mFadeOutAnimation = AnimationUtils.loadAnimation(this, R.anim.fadeout);

mFadeInAnimation.setAnimationListener(animationFadeInListener);

mFadeOutAnimation.setAnimationListener(animationFadeOutListener);

// при запуске начинаем с анимации исчезновения

mImageView.startAnimation(mFadeOutAnimation);

}

@Override

protected void onPause() {

super.onPause();

mImageView.clearAnimation();

}

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

@Override

public void onAnimationEnd(Animation animation) {

mImageView.startAnimation(mFadeInAnimation);

}

@Override

public void onAnimationRepeat(Animation animation) {

// TODO Auto-generated method stub

}

@Override

public void onAnimationStart(Animation animation) {

// TODO Auto-generated method stub

}

};

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

@Override

public void onAnimationEnd(Animation animation) {

mImageView.startAnimation(mFadeOutAnimation);

}

@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.

**Использование системных ресурсов**

В Android есть пара системных ресурсов, связанных с изменением прозрачности: **android.R.anim.fade\_in** и **android.R.anim.fade\_out**. При их использовании вам не нужно создавать собственные XML-файлы анимации, если стандартная системная анимация устроит вас. Просто замените в предыдущем примере пару строчек:

animationFadeIn = AnimationUtils.loadAnimation(this, android.R.anim.fade\_in);

animationFadeOut = AnimationUtils.loadAnimation(this, android.R.anim.fade\_out);