**ViewPropertyAnimator**

[Скрываем и показываем кнопку через анимацию](http://developer.alexanderklimov.ru/android/animation/viewpropertyanimator.php#hideshow)  
[Класс Fade](http://developer.alexanderklimov.ru/android/animation/viewpropertyanimator.php#customfade)  
[Crossfade: Эффект растворения](http://developer.alexanderklimov.ru/android/animation/viewpropertyanimator.php#crossfade)

В статье про [ObjectAnimator](http://developer.alexanderklimov.ru/android/animation/objectanimator.php) мы рассказывали о новом классе, который появился в Android 3.0. А в Android 3.1 (API 12) делать анимацию для компонентов стало ещё проще! Теперь подключать нужный вид анимации стало намного удобнее.

Для сравнения вспомним код с использованием **ObjectAnimator** для кнопки

ObjectAnimator objectAnimator = ObjectAnimator.ofFloat(

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

objectAnimator.setDuration(1000);

objectAnimator.start();

А теперь посмотрите на новый код для кнопки:

myButton.animate().x(500).y(500);

У кнопки есть методы **animate()**, **x()**, **y()**, которые можно выстроить в цепочку и данный код заставит кнопку переместиться по оси X и Y. Удобно, не так ли?

Для анимации прозрачности достаточно одной  строчки кода:

myButton.animate().alpha(0);

Синтаксис кажется понятным, коротким и легко запоминаемым. Обратите внимание, что мы не нуждаемся также в команде **start()**, так как это работает по умолчанию. Также по умолчанию используется своя продолжительность анимации, но можно задать свою:

myButton.animate().setDuration(2000);

myButton.animate().rotationYBy(720);

myButton.animate().x(100).y(200);

В этом примере задаётся продолжительность в 2 секунды, вращение на 720 градусов (3D-эффект вращения!) и перемещение.

Сами разработчики говорят, что новый вид анимации не только удобен в плане написания кода, но и требует меньше ресурсов. Если в вашем проекте требуется анимация для компонентов, то всячески рекомендую использовать новый класс.

Можете самостоятельно изучить документацию по этому вопросу, там не так много методов.

Видео от разработчиков

**Скрываем и показываем кнопку через** **анимацию**

Разместите на экране три кнопки. Первая кнопка **buttonSend** будет подопытным кроликом, а остальные две будут её скрывать и показывать.

public void onClick(View view) {

buttonSend.animate().alpha(0f).setListener(new AnimatorListenerAdapter() {

public void onAnimationEnd(Animator animation) {

buttonSend.setVisibility(View.INVISIBLE); // или GONE

buttonSend.setAlpha(1f);

buttonSend.animate().setListener(null);

}

});

}

public void onClick2(View view) {

buttonSend.setAlpha(0);

buttonSend.setVisibility(View.VISIBLE);

buttonSend.animate().alpha(1);

}

Надеюсь, вы поняли идею. Вот ещё пример манипуляции с компонентом **View**:

View viewToAnimate;

public void onClick(View v) {

if (viewToAnimate.getAlpha() > 0f) {

// If the view is visible already, slide it out to the right

viewToAnimate.animate().alpha(0f).translationX(1000f);

} else {

// If the view is hidden, do a fade-in in place

// Property Animations actually modify the view, so

// we have to reset the view's location first

viewToAnimate.setTranslationX(0f);

viewToAnimate.animate().alpha(1f);

}

}

**Класс Fade**

Разработчик [CHET HAASE](http://graphics-geek.blogspot.ru/) написал небольшой класс **Fade**, который удобно подключить к своему проекту и использовать для анимации кнопок и других элементов.

Скачать класс с примером можно [отсюда](https://sites.google.com/site/androidcontentfromchet/downloads/Fader.zip).

Используется просто. Добавляете класс в свой проект. Далее в своей активности пишите:

Fade fade = new Fade();

public void onClick(View view) {

fade.hide(buttonTest, View.INVISIBLE);

}

При нажатии на кнопку вторая кнопка станет невидимой. Можно также использовать константу **View.GONE**, чтобы кнопка освободила место на экране.

Чтобы вывести кнопку обратно:

fade.show(buttonSend);

Также у класса есть методы **add()** (два вараианта) и **remove()**, которые позволяет добавить новый View на экран.

На всякий случай, если ссылка станет не рабочей, полный код класса:

Показать код (щелкните мышкой)

**Crossfade: Эффект растворения**

В документации к Android приводится ещё один пример эффекта растворения, когда один элемент постепенно исчезает, а вместо него проступает другой компонент. Я переделал пример под себя. Добавим на экран две текстовые метки:

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

android:layout\_width="match\_parent"

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

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

android:id="@+id/content"

android:layout\_width="match\_parent"

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

<TextView

style="?android:textAppearanceMedium"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:lineSpacingMultiplier="1.2"

android:padding="16dp"

android:text="Здесь очень длинный текст"

android:textIsSelectable="true" />

</ScrollView>

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Здесь очень длинный текст"

android:textAppearance="?android:attr/textAppearanceLarge" />

</FrameLayout>

Переключаться будем через пункт меню

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

<item

android:id="@+id/action\_toggle"

android:showAsAction="ifRoom|withText"

android:title="Toggle"/>

</menu>

В комментариях даны пояснения к коду

package ru.alexanderklimov.test;

import android.animation.Animator;

import android.animation.AnimatorListenerAdapter;

import android.app.Activity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

public class TestActivity extends Activity {

// флаг, определяющий, какой компонент нужно выводить на экран

private boolean mContentLoaded;

private View mContentView;

private View mContentView2;

/\*\*

\* The system "short" animation time duration, in milliseconds. This

\* duration is ideal for subtle animations or animations that occur very

\* frequently.

\*/

private int mShortAnimationDuration;

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

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_test);

mContentView = findViewById(R.id.content);

mContentView2 = findViewById(R.id.textView1);

// прячем первый компонент при загрузке

mContentView.setVisibility(View.GONE);

// Retrieve and cache the system's default "short" animation time.

mShortAnimationDuration = getResources().getInteger(

android.R.integer.config\_shortAnimTime);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

super.onCreateOptionsMenu(menu);

getMenuInflater().inflate(R.menu.test, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.action\_toggle:

// Toggle whether content is loaded.

mContentLoaded = !mContentLoaded;

showContentOrLoadingIndicator(mContentLoaded);

return true;

}

return super.onOptionsItemSelected(item);

}

/\*\*

\* Cross-fades between two Views

\*/

private void showContentOrLoadingIndicator(boolean contentLoaded) {

// Decide which view to hide and which to show.

final View showView = contentLoaded ? mContentView : mContentView2;

final View hideView = contentLoaded ? mContentView2 : mContentView;

// Set the "show" view to 0% opacity but visible, so that it is visible

// (but fully transparent) during the animation.

showView.setAlpha(0f);

showView.setVisibility(View.VISIBLE);

// Animate the "show" view to 100% opacity, and clear any animation

// listener set on

// the view. Remember that listeners are not limited to the specific

// animation

// describes in the chained method calls. Listeners are set on the

// ViewPropertyAnimator object for the view, which persists across

// several

// animations.

showView.animate().alpha(1f).setDuration(mShortAnimationDuration)

.setListener(null);

// Animate the "hide" view to 0% opacity. After the animation ends, set

// its visibility

// to GONE as an optimization step (it won't participate in layout

// passes, etc.)

hideView.animate().alpha(0f).setDuration(mShortAnimationDuration)

.setListener(new AnimatorListenerAdapter() {

@Override

public void onAnimationEnd(Animator animation) {

hideView.setVisibility(View.GONE);

}

});

}

}

Для старых устройств используется библиотека поддержки, которая позволяет также использовать данный класс анимации.

// Gradle. Используйте последнюю версию

compile 'com.android.support:support-v4:23.1.0'

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

**ViewCompat**.animate(button).alpha(0.2f).xBy(-100).yBy(100);