Анимация фрагментов

Android позволяет переключаться между фрагментами с помощью анимационных эффектов. Рассмотрим примеры с применением готовых эффектов, а также создадим свою собственную анимацию.

Создадим новый проект с главной активностью. В активность динамически будут подключаться фрагменты, между которыми будем переключаться с анимационными эффектами при помощи кнопки. Компонент **FrameLayout** послужит контейнером для подключаемых фрагментов.

Разметка для активности:

<RelativeLayout 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" >

<FrameLayout

android:id="@+id/fragment"

android:layout\_width="wrap\_content"

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

<Button

android:id="@+id/button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_toRightOf="@id/fragment"

android:text="Переключиться" />

</RelativeLayout>

Создадим разметки для фрагментов.

fragment1.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"

android:background="#FF0000">

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:text="Fragment 1" />

</LinearLayout>

fragment2.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"

android:background="#00FF00">

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:text="Fragment 2" />

</LinearLayout>

Отличия между фрагментами - фоновый цвет (красный и зелёный) и текст, по которым будет проще различать фрагменты.

Фрагмент должен иметь свой класс. Для примера достаточно простейшего кода, где нужно указать разметку фрагмента.

Fragment1.java

package ru.alexanderklimov.fragmentsanimation;

import android.app.Fragment;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

public class Fragment1 extends Fragment {

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

return inflater.inflate(R.layout.fragment1, null);

}

}

По такому же принципу создайте класс для второго фрагмента самостоятельно.

Теперь перейдём к коду для главной активности:

package ru.alexanderklimov.fragmentsanimation;

import android.os.Bundle;

import android.app.Activity;

import android.app.Fragment;

import android.app.FragmentTransaction;

import android.view.Menu;

import android.view.View;

public class MainActivity extends Activity {

private Fragment fragment1;

private Fragment fragment2;

private FragmentTransaction transaction;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

fragment1 = new Fragment1();

fragment2 = new Fragment2();

transaction = getFragmentManager().beginTransaction();

transaction.replace(R.id.fragment, fragment1);

transaction.addToBackStack(null);

transaction.commit();

}

public void onClick(View v){

transaction = getFragmentManager().beginTransaction();

// transaction.setTransition(FragmentTransaction.TRANSIT\_FRAGMENT\_OPEN);

// transaction.setCustomAnimations(R.animator.slide\_in\_left, R.animator.slide\_in\_right);

if(fragment1.isVisible()){

transaction.replace(R.id.fragment, fragment2);

}else{

transaction.replace(R.id.fragment, fragment1);

}

transaction.commit();

}

}

В коде я закомментировал две строчки для анимации. В таком виде переключение между фрагментами происходит стандартным способом без анимации. Теперь раскомментируйте первую строчку **transaction.setTransition(FragmentTransaction.TRANSIT\_FRAGMENT\_OPEN);**. В классе **FragmentTransaction** есть несколько готовых анимаций. С помощью метода **setTransition(int transit)** мы можем указать нужную анимацию и увидеть её в действии.

Список анимаций

* **TRANSIT\_FRAGMENT\_CLOSE**
* **TRANSIT\_FRAGMENT\_OPEN**
* **TRANSIT\_FRAGMENT\_FADE**
* **TRANSIT\_NONE**

Снова закомментируйте строку и раскомментируйте вторую строку **transaction.setCustomAnimations(R.animator.slide\_in\_left, R.animator.slide\_in\_right);**. Метод **setCustomAnimations()** позволяет указать собственную анимацию. Методу передаются два параметра. Первый параметр описывает анимацию для фрагмента, который появляется, а второй — описывает анимацию для фрагмента, который убирается с экрана устройства. Метод следует вызывать до появления фрагментов, иначе анимация не будет применена.

С методом **setCustomAnimations()** нужно быть осторожным при работе с фрагментами из support-библиотеки. В одной из версий библиотеки разработчики из Гугла всё поломали и код перестал работать. Неизвестно когда починят. Поэтому используйте стандартные фрагменты.

Для анимации нужно создать XML-файлы в папке **res/animator** (её тоже нужно создать вручную).

slide\_in\_left.xml

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

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

<objectAnimator

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

android:duration="1500"

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

android:propertyName="y"

android:valueFrom="-1280"

android:valueTo="0"

android:valueType="floatType" />

</set>

slide\_in\_right.xml

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

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

android:ordering="together">

<objectAnimator

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

android:propertyName="alpha"

android:valueType="floatType"

android:valueTo="0"

android:duration="300"/>

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

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

android:propertyName="x"

android:valueType="floatType"

android:valueTo="1280"

android:valueFrom="0"

android:duration="1500"/>

</set>

Элементы визуальных эффектов задаются в теге **objectAnimator**. У атрибута **propertyName** указывается свойство фрагмента, которое мы будем изменять при анимации, **valueType** указывает тип изменяемого параметра. Атрибуты **valueFrom** и **valueTo** указывают диапазон изменения параметра, указанного в **propertyName**. Если параметр **valueFrom** не указан, то значение берётся равное текущему. В нашем случае **valueFrom** равен -1280, это означает, что движение фрагмента по оси **y** будет начинаться со значения -1280 и перемещение будет происходить пока значение **y** не станет равным 0 для верхнего левого угла нашего фрагмента в течении 1500 миллисекунд (атрибут **duration**).

Тег **set** служит для объединения эффектов либо их разделения. В файле **slide\_in\_right.xml** используется атрибут **ordering** со значением **together**, что означает проигрывать эффекты одновременно, в противовес ему существует значение **sequentially**, которое требует последовательного отображения эффектов в анимации.

Переворачиваем карту

Напишем ещё один пример для закрепления материала.

Найдите в интернете две картинки, например, изображения карты и её рубашки. Подготовим четыре файла в папке **res/animator**.

card\_flip\_left\_enter.xml

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

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

<objectAnimator

android:valueFrom="1.0"

android:valueTo="0.0"

android:propertyName="alpha"

android:duration="0"/>

<objectAnimator

android:valueFrom="-180"

android:valueTo="0"

android:propertyName="rotationY"

android:interpolator="@android:interpolator/accelerate\_decelerate"

android:duration="@integer/card\_flip\_duration\_full"/>

<objectAnimator

android:valueFrom="0.0"

android:valueTo="1.0"

android:propertyName="alpha"

android:startOffset="@integer/card\_flip\_duration\_half"

android:duration="1"/>

</set>

card\_flip\_left\_exit.xml

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

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

<objectAnimator

android:duration="@integer/card\_flip\_duration\_full"

android:interpolator="@android:interpolator/accelerate\_decelerate"

android:propertyName="rotationY"

android:valueFrom="0"

android:valueTo="180"/>

<objectAnimator

android:duration="1"

android:propertyName="alpha"

android:startOffset="@integer/card\_flip\_duration\_half"

android:valueFrom="1.0"

android:valueTo="0.0"/>

</set>

card\_flip\_right\_enter.xml

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

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

<objectAnimator

android:duration="0"

android:propertyName="alpha"

android:valueFrom="1.0"

android:valueTo="0.0"/>

<objectAnimator

android:duration="@integer/card\_flip\_duration\_full"

android:interpolator="@android:interpolator/accelerate\_decelerate"

android:propertyName="rotationY"

android:valueFrom="180"

android:valueTo="0"/>

<objectAnimator

android:duration="1"

android:propertyName="alpha"

android:startOffset="@integer/card\_flip\_duration\_half"

android:valueFrom="0.0"

android:valueTo="1.0"/>

</set>

card\_flip\_right\_exit.xml

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

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

<objectAnimator

android:duration="@integer/card\_flip\_duration\_full"

android:interpolator="@android:interpolator/accelerate\_decelerate"

android:propertyName="rotationY"

android:valueFrom="0"

android:valueTo="-180"/>

<objectAnimator

android:duration="1"

android:propertyName="alpha"

android:startOffset="@integer/card\_flip\_duration\_half"

android:valueFrom="1.0"

android:valueTo="0.0"/>

</set>

В файле **res/values/strings.xml** добавьте пару новых ресурсов.

<integer name="card\_flip\_duration\_full">1000</integer>

<integer name="card\_flip\_duration\_half">500</integer>

Теперь создадим два макета для двух фрагментов в папке **res/layout**. Фрагменты будут содержать по одной картинке.

fragment\_card\_front.xml

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:scaleType="centerCrop"

android:src="@drawable/card\_front"/>

fragment\_card\_back.xml

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

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:scaleType="centerCrop"

android:src="@drawable/card\_back"/>

Создаём два класса для фрагментов, которые загружают свои макеты.

package ru.alexanderklimov.as21;

import android.app.Fragment;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

public class CardFrontFragment extends Fragment {

@Override

public View onCreateView(LayoutInflater inflater,

ViewGroup container, Bundle savedInstanceState) {

return inflater.inflate(

R.layout.fragment\_card\_front, container, false);

}

}

package ru.alexanderklimov.as21;

import android.app.Fragment;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

public class CardBackFragment extends Fragment {

@Override

public View onCreateView(LayoutInflater inflater,

ViewGroup container, Bundle savedInstanceState) {

return inflater.inflate(

R.layout.fragment\_card\_back, container, false);

}

}

Упростим макет активности.

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

<FrameLayout android:id="@+id/container"

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

android:layout\_width="match\_parent"

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

Осталось написать код. Ещё раз напомню, что используйте стандартные фрагменты, а не из библиотеки совместимости.

package ru.alexanderklimov.as21;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.FrameLayout;

public class MainActivity extends AppCompatActivity {

private boolean mShowingBack = false;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

FrameLayout frameLayout = (FrameLayout) findViewById(R.id.container);

assert frameLayout != null;

frameLayout.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

flipCard();

}

});

if (savedInstanceState == null) {

getFragmentManager()

.beginTransaction()

.add(R.id.container, new CardFrontFragment())

.commit();

}

}

private void flipCard() {

if (mShowingBack) {

mShowingBack = false;

getFragmentManager().popBackStack();

} else {

mShowingBack = true;

getFragmentManager()

.beginTransaction()

.setCustomAnimations(

R.animator.card\_flip\_right\_enter,

R.animator.card\_flip\_right\_exit,

R.animator.card\_flip\_left\_enter,

R.animator.card\_flip\_left\_exit)

.replace(R.id.container, new

CardBackFragment())

.addToBackStack(null)

.commit();

}

}

}

Контейнер обрабатывает касание пальца как щелчок и переворачивает фрагмент на другой. Результат в видео.