/\*

\* Copyright 2012 The Android Open Source Project

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

package com.example.android.animationsdemo;

import android.app.Activity;

import android.app.Fragment;

import android.app.FragmentManager;

import android.content.Intent;

import android.os.Bundle;

import android.os.Handler;

import android.support.v4.app.NavUtils;

import android.view.LayoutInflater;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewGroup;

import android.widget.TextView;

/\*\*

\* Demonstrates a "card-flip" animation using custom fragment transactions ({@link

\* android.app.FragmentTransaction#setCustomAnimations(int, int)}).

\*

\* <p>This sample shows an "info" action bar button that shows the back of a "card", rotating the

\* front of the card out and the back of the card in. The reverse animation is played when the user

\* presses the system Back button or the "photo" action bar button.</p>

\*/

public class CardFlipActivity extends Activity

implements FragmentManager.OnBackStackChangedListener {

/\*\*

\* A handler object, used for deferring UI operations.

\*/

private Handler mHandler = new Handler();

/\*\*

\* Whether or not we're showing the back of the card (otherwise showing the front).

\*/

private boolean mShowingBack = false;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_card\_flip);

if (savedInstanceState == null) {

// If there is no saved instance state, add a fragment representing the

// front of the card to this activity. If there is saved instance state,

// this fragment will have already been added to the activity.

getFragmentManager()

.beginTransaction()

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

.commit();

} else {

mShowingBack = (getFragmentManager().getBackStackEntryCount() > 0);

}

// Monitor back stack changes to ensure the action bar shows the appropriate

// button (either "photo" or "info").

getFragmentManager().addOnBackStackChangedListener(this);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

super.onCreateOptionsMenu(menu);

// Add either a "photo" or "finish" button to the action bar, depending on which page

// is currently selected.

MenuItem item = menu.add(Menu.NONE, R.id.action\_flip, Menu.NONE,

mShowingBack

? R.string.action\_photo

: R.string.action\_info);

item.setIcon(mShowingBack

? R.drawable.ic\_action\_photo

: R.drawable.ic\_action\_info);

item.setShowAsAction(MenuItem.SHOW\_AS\_ACTION\_IF\_ROOM);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case android.R.id.home:

// Navigate "up" the demo structure to the launchpad activity.

// See http://developer.android.com/design/patterns/navigation.html for more.

NavUtils.navigateUpTo(this, new Intent(this, MainActivity.class));

return true;

case R.id.action\_flip:

flipCard();

return true;

}

return super.onOptionsItemSelected(item);

}

private void flipCard() {

if (mShowingBack) {

getFragmentManager().popBackStack();

return;

}

// Flip to the back.

mShowingBack = true;

// Create and commit a new fragment transaction that adds the fragment for the back of

// the card, uses custom animations, and is part of the fragment manager's back stack.

getFragmentManager()

.beginTransaction()

// Replace the default fragment animations with animator resources representing

// rotations when switching to the back of the card, as well as animator

// resources representing rotations when flipping back to the front (e.g. when

// the system Back button is pressed).

.setCustomAnimations(

R.animator.card\_flip\_right\_in, R.animator.card\_flip\_right\_out,

R.animator.card\_flip\_left\_in, R.animator.card\_flip\_left\_out)

// Replace any fragments currently in the container view with a fragment

// representing the next page (indicated by the just-incremented currentPage

// variable).

.replace(R.id.container, new CardBackFragment())

// Add this transaction to the back stack, allowing users to press Back

// to get to the front of the card.

.addToBackStack(null)

// Commit the transaction.

.commit();

// Defer an invalidation of the options menu (on modern devices, the action bar). This

// can't be done immediately because the transaction may not yet be committed. Commits

// are asynchronous in that they are posted to the main thread's message loop.

mHandler.post(new Runnable() {

@Override

public void run() {

invalidateOptionsMenu();

}

});

}

@Override

public void onBackStackChanged() {

mShowingBack = (getFragmentManager().getBackStackEntryCount() > 0);

// When the back stack changes, invalidate the options menu (action bar).

invalidateOptionsMenu();

}

/\*\*

\* A fragment representing the front of the card.

\*/

public static class CardFrontFragment extends Fragment {

public CardFrontFragment() {

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

return inflater.inflate(R.layout.fragment\_card\_front, container, false);

}

}

/\*\*

\* A fragment representing the back of the card.

\*/

public static class CardBackFragment extends Fragment {

public CardBackFragment() {

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

return inflater.inflate(R.layout.fragment\_card\_back, container, false);

}

}

}