# How to Make View Flipper Animation

# On Android

Student ID: S3342133

Name: Phan Thanh San

Student ID: S3296761

Name: Dinh Tuan Vu

COSC2543 Mobile Application Development

Lecturer: George Nguyen

# Introduction

This document purposes is to provide some aspect and ways of making a view flipper animation on an android application.

The following step will indicate the progress in making a view flipper animation.

# Advantages and Disadvantages

Advantage: help the application looks more professional, attractive and easy use for user.

Disadvantage: sometimes it will hard to understand the method and tricky to flip over to another layout. Bug might occurs if the others layout doesn’t accept OnTouchListener

# Layout

It is necessary to have more than one layout to switch between them while using the android app.

While writing your layout, put this line to cover up your whole layout code.

For example:

<ViewFlipper

anroid:id=”@+id/layoutswitcher”

android:layout\_width=”fill\_parent”

android:layout\_height=”fill\_parent”

android:layout\_margin=”6dip”>

Here, a n id called “layoutswitcher” is used to mark the view flipper for further use.

It is possible to put 2 layouts in an xml files, for example:

<LinearLayout>

Your code here for the 1st layout

</LinearLayout>

<LinearLayout>

Your code here for the 2nd layout

</LinearLayout>

# Coding

As known, coding is a most important part of building an application. Below is the step which takes to build a complete view flipper animation for the android application.

## Preparation:

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

LinearLayout layout = (LinearLayout) findViewById(R.id.*layout\_main*);

layout.setOnTouchListener((OnTouchListener) **this**);

}

This code used to call the layout in order to set the OnTouchListener for that layout.

## Animation

To scroll to the next page here is the code:

The inFromRight() method used to load the page from right of the screen

**public** **static** Animation inFromLeft() {

Animation inFromRight = **new** TranslateAnimation(

Animation.*RELATIVE\_TO\_PARENT*, +1.0f,

Animation.*RELATIVE\_TO\_PARENT*, 0.0f,

Animation.*RELATIVE\_TO\_PARENT*, 0.0f,

Animation.*RELATIVE\_TO\_PARENT*, 0.0f);

inFromRight.setDuration(350);

inFromRight.setInterpolator(**new** AccelerateInterpolator());

**return** inFromRight;

}

This outToLeft() method used to make the page go out to the left of the screen

**public** **static** Animation outToRight() {

Animation outtoLeft = **new** TranslateAnimation(

Animation.*RELATIVE\_TO\_PARENT*, 0.0f,

Animation.*RELATIVE\_TO\_PARENT*, -1.0f,

Animation.*RELATIVE\_TO\_PARENT*, 0.0f,

Animation.*RELATIVE\_TO\_PARENT*, 0.0f);

outtoLeft.setDuration(350);

outtoLeft.setInterpolator(**new** AccelerateInterpolator());

**return** outtoLeft;

}

To scroll back to the previous page:

The inFromLeft() method used to load the page from the left side of the screen

**public** **static** Animation inFromLeft() {

Animation inFromLeft = **new** TranslateAnimation(

Animation.*RELATIVE\_TO\_PARENT*, -1.0f,

Animation.*RELATIVE\_TO\_PARENT*, 0.0f,

Animation.*RELATIVE\_TO\_PARENT*, 0.0f,

Animation.*RELATIVE\_TO\_PARENT*, 0.0f);

inFromLeft.setDuration(350);

inFromLeft.setInterpolator(**new** AccelerateInterpolator());

**return** inFromLeft;

}

The outToRight() methos udes to make the page go out to the right of the screen

**public** **static** Animation outToRight() {

Animation outtoRight = **new** TranslateAnimation(

Animation.*RELATIVE\_TO\_PARENT*, 0.0f,

Animation.*RELATIVE\_TO\_PARENT*, +1.0f,

Animation.*RELATIVE\_TO\_PARENT*, 0.0f,

Animation.*RELATIVE\_TO\_PARENT*, 0.0f);

outtoRight.setDuration(350);

outtoRight.setInterpolator(**new** AccelerateInterpolator());

**return** outtoRight;

}

## On touch

This is the key to determine which method call from above will be used for the suitable action

@Override

**public** **boolean** onTouch(View arg0, MotionEvent arg1) {

// Get the action that was done on this touch event

**switch** (arg1.getAction()) {

**case** MotionEvent.*ACTION\_DOWN*: {

// store the X value when the user's finger was pressed down

downXValue = arg1.getX();

**break**;

}

**case** MotionEvent.*ACTION\_UP*: {

// Get the X value when the user released his/her finger

**float** currentX = arg1.getX();

// going backwards: pushing stuff to the right

**if** (downXValue < currentX) {

// Get a reference to the ViewFlipper

ViewFlipper vf = (ViewFlipper) findViewById(R.id.*layoutswitcher*);

vf.setInAnimation(*inFromLeft*());

vf.setOutAnimation(*outToRight*());

vf.showPrevious();

}

// going forwards: pushing stuff to the left

**if** (downXValue > currentX) {

// Get a reference to the ViewFlipper

ViewFlipper vf = (ViewFlipper) findViewById(R.id.*layoutswitcher*);

vf.setInAnimation(*inFromRight*());

vf.setOutAnimation(*outToLeft*());

vf.showNext();

}

**break**;

}

}

// if you return false, these actions will not be recorded

**return** **true**;

}

# Conclusion

The view flipper animation is commonly used by many android applications, which has created a trend for the android industry and a way that customer view about android application in general.

# References

Developers, Android 4.1 r1, 14 Aug 2012, viewed 1st September 2012, URL<<http://developer.android.com/reference/android/widget/ViewFlipper.html>>

Francesco Lo Truglio, Web Developer and Project Manager, Android: View Flipper animation, 30 Mar 2010, viewed 1st September 2012, URL<<http://www.codeproject.com/Articles/69008/Android-ViewFlipper-Touch-Animation-like-News-Weat>>