Матрица

2-й курс/Закрытая зона

[Метод preScale()](http://developer.alexanderklimov.ru/android/catshop/matrix.php#prescale)  
[Метод setScale()](http://developer.alexanderklimov.ru/android/catshop/matrix.php#setscale)  
[Метод postConcat()](http://developer.alexanderklimov.ru/android/catshop/matrix.php#postconcat)  
[Метод postScale()](http://developer.alexanderklimov.ru/android/catshop/matrix.php#postscale)  
[Метод postRotate()](http://developer.alexanderklimov.ru/android/catshop/matrix.php#postrotate)  
[Метод setRotate()](http://developer.alexanderklimov.ru/android/catshop/matrix.php#setrotate)  
[Метод postSkew()](http://developer.alexanderklimov.ru/android/catshop/matrix.php#postskew)  
[Метод setTranslate()](http://developer.alexanderklimov.ru/android/catshop/matrix.php#settranslate)

Класс **android.graphics.Matrix** позволяет производить различные преобразования с изображением. Мы рассмотрим несколько примеров из множества методов данного класса.

Матрица использует массив из девяти чисел 3х3. По математическим формулам производится расчёт и данные обновляются. Существуют готовые стандартные методы, в которых зашиты преобразования матрицы. Также можно создавать матрицы вручную.

Сначала немного теории.

Каждое число в матрице относится к одному из трёх координат (x, y, z) для каждой точки изображения. Например, имеется матрица:

1 0 0

0 1 0

0 0 1

Первый ряд (1, 0, 0) определяет, что координаты X будут преобразованы по формуле: x = 1x + 0y + 0z.

По этой формуле получается, что будут меняться значения X, а остальные значения Y и Z останутся нетронутыми.

Второй ряд (0, 1, 0) означает, что координаты Y будут преобразованы по формуле: y = 0x + 1y + 0z.

Соответственно, третий ряд (0, 0, 1) означает, что координаты Z вычисляются по формуле z = 0x + 0y + 1z.

Создадим объект **Matrix** и установим значения через метод **setValues()**.

Matrix matrix = new Matrix();

matrix.setValues(new float[]{

1, 0, 0,

0, 1, 0,

0, 0, 1

});

canvas.drawBitmap(bitmap, matrix, paint);

Данный пример создаст копию изображения, так как все значения останутся прежними.

Заменим первое число 1 на 0.5.

0.5 0 0

0 1 0

0 0 1

Если вы вручную посчитаете некоторые координаты точек, то увидите, что они будут смещены в сторону. Например, координата (10, 0) после применения матрицы станет (5, 0). По сути, картинка сжимается в два раза по горизонтали.

В коде это будет выглядеть следующим образом.

Matrix matrix = new Matrix();

matrix.setValues(new float[] {

.5f, 0, 0,

0, 1, 0,

0, 0, 1

});

canvas.drawBitmap(bitmap, matrix, paint);

Изменим второе число для эксперимента.

Matrix matrix = new Matrix();

matrix.setValues(new float[] {

1, .5f, 0,

0, 1, 0,

0, 0, 1

});

canvas.drawBitmap(bitmap, matrix, paint);

На этот раз вы увидите эффект сдвига.

Также можно использовать отрицательные значение. Во время манипуляций размеры изображения меняются и может так получиться, что новое изображение будет обрезано, если не позаботиться заранее о создании нового объекта **Bitmap** с подходящими размерами.

alteredBitmap = Bitmap.createBitmap(bitmap.getWidth() \* 2, bitmap.getHeight(), bitmap.getConfig());

Вариант, когда изменённое изображение подгоняется под нужный размер:

Matrix matrix = new Matrix();

matrix.setRotate(15, bitmap.getWidth() / 2, bitmap.getHeight() / 2);

alteredBitmap = Bitmap.createBitmap(bitmap, 0, 0, bitmap.getWidth(), bitmap.getHeight(),

matrix, false);

alteredImageView.setImageBitmap(alteredBitmap);

Но, вручную составлением матрицы не занимаются. Самые распространённые формулы зафиксированы и вы можете использовать готовые методы с нужными эффектами.

У класса **Matrix** есть несколько методов с приставками **pre** и **post**. Они позволяют выполнять трансформацию изображения в нужном порядке. Например, можно вызвать **preScale()**, а затем **setRotate()** или **setRotate()**, а затем **postRotate()**. Результат будет разным.

Для опытов нам понадобится **ImageView** с картинкой, а также различные элементы управления: **Button**, **Spinner**, **SeekBar** и т.д.

Метод preScale()

Любой уважающий себя графический редактор обязан уметь отражать картинку по вертикали или горизонтали.

Для преобразований изображений существует класс **Matrix** с различными методами. Для нашего случая достаточно вызова метода **preScale(float sx, float sy)**. Все, что нам требуется - менять на отрицательные значения первый или второй параметр. Всего возможно четыре варианта:

Без изменений: [ x = x, y = y ]  
Для вертикального отражения: [ x = x, y = y \* -1 ]  
Для горизонтального отражения: [ x = x \* -1, y = y ]  
Отражение в двух направлениях: [ x = x \* -1, y = y \* -1 ]

Чтобы увидеть все варианты отражения, нам хватит двух кнопок, так как мы будем брать текущее изображение и применять к нему операцию отражения.

Разместим на экране элемент **ImageView** с каким-нибудь изображением и две кнопки:

<LinearLayout 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"

android:orientation="vertical"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginBottom="10dp"

android:orientation="horizontal" >

<Button

android:id="@+id/buttonHorizontal"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:text="По горизонтали" />

<Button

android:id="@+id/buttonVertical"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_weight="2"

android:text="По вертикали" />

</LinearLayout>

<ImageView

android:id="@+id/imageView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:src="@drawable/sleepbedcat"

android:scaleType="centerCrop" />

</LinearLayout>

Осталось написать код:

// Если этот код работает, его написал Александр Климов,

// а если нет, то не знаю, кто его писал.

package ru.alexanderklimov.testapplication;

import android.graphics.Bitmap;

import android.graphics.Matrix;

import android.graphics.drawable.BitmapDrawable;

import android.os.Bundle;

import android.support.v7.app.ActionBarActivity;

import android.view.View;

import android.widget.Button;

import android.widget.ImageView;

public class MainActivity extends ActionBarActivity {

private ImageView mImageView;

public static final int FLIP\_VERTICAL = 1;

public static final int FLIP\_HORIZONTAL = 2;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Button horizontalFlipButton = (Button) findViewById(R.id.buttonHorizontal);

Button verticalFlipButton = (Button) findViewById(R.id.buttonVertical);

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

horizontalFlipButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

mImageView.setImageBitmap(flip(mImageView, FLIP\_HORIZONTAL));

}

});

verticalFlipButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

mImageView.setImageBitmap(flip(mImageView, FLIP\_VERTICAL));

}

});

}

private Bitmap flip(ImageView imageView, int type) {

// Получаем изображение с указанного ImageView

BitmapDrawable bitmapDrawable = (BitmapDrawable) imageView.getDrawable();

Bitmap bitmap = bitmapDrawable.getBitmap();

int bmpWidth = bitmap.getWidth();

int bmpHeight = bitmap.getHeight();

Matrix matrix = new Matrix();

Bitmap mirrorBitmap = null;

if (type == FLIP\_HORIZONTAL) {

matrix.preScale(-1.0f, 1.0f);

mirrorBitmap = Bitmap.createBitmap(bitmap, 0, 0, bmpWidth, bmpHeight,

matrix, true);

} else if (type == FLIP\_VERTICAL) {

matrix.preScale(1.0f, -1.0f);

mirrorBitmap = Bitmap.createBitmap(bitmap, 0, 0, bmpWidth, bmpHeight,

matrix, true);

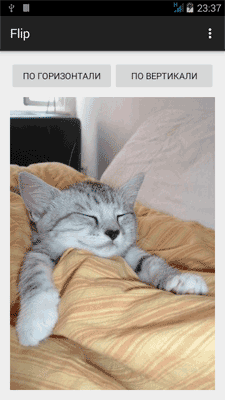
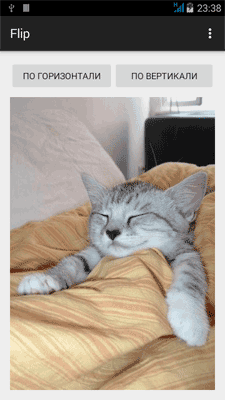
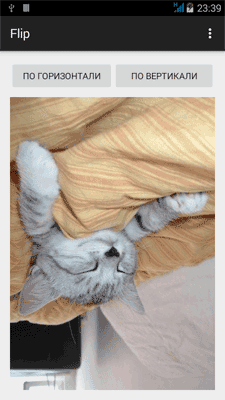
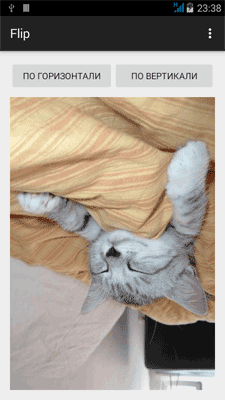
}

return mirrorBitmap;

}

}

Запускаем проект и щелкаем на кнопки по нескольку раз, чтобы получить все варианты отражения:

Метод setScale()

В методе указываются масштаб по двум осям. Увеличим текущее изображение в полтора раза по горизонтали. Код поместил в обработчик щелчка первой кнопки.

horizontalFlipButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// Получаем изображение с указанного ImageView

BitmapDrawable bitmapDrawable = (BitmapDrawable) mImageView.getDrawable();

Bitmap bitmap = bitmapDrawable.getBitmap();

int bmpWidth = bitmap.getWidth();

int bmpHeight = bitmap.getHeight();

Matrix matrix = new Matrix();

Bitmap alteredBitmap;

matrix.setScale(1.5f, 1);

alteredBitmap = Bitmap.createBitmap(bitmap, 0, 0, bmpWidth, bmpHeight,

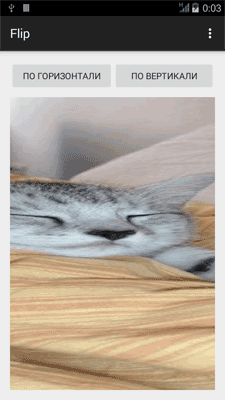
matrix, true);

mImageView.setImageBitmap(alteredBitmap);

}

});

После нескольких щелчков кнопки, кота "разнесло". Жрать надо меньше.



Метод postConcat()

Уже само название метода, содержащее слово *cat*, говорит о том, что им нужно пользоваться только для изображений с котами. Например, можно зеркально отразить картинку.

Чтобы отразить по оси Y нужно применить матрицу:

-1, 0, 0,

0, 1, 0,

0, 0, 1

Также можно отразить по оси X или сразу по двум осям. Напишем пример для всех случаев.

Показать код (щелкните мышкой)

Выше был показан более удобный способ зеркального отражения при помощи метода **preScale()**.

Также отражение можно добиться методами:

matrix.setScale(-1, 1);

matrix.postTranslate(bitmap.getWidth(),0);

Отражение сверху вниз:

matrix.setScale(1, -1);

matrix.postTranslate(0, bitmap.getHeight());

Метод postScale()

Метод **postScale()** позволяет изменять размеры изображения. Добавим к проекту компонент **Spinner**, чтобы выбирать масштаб изменения размера. И напишем код:

package ru.alexanderklimov.matrix;

import android.graphics.Bitmap;

import android.graphics.Matrix;

import android.graphics.drawable.BitmapDrawable;

import android.os.Bundle;

import android.support.v7.app.ActionBarActivity;

import android.view.Menu;

import android.view.View;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.ImageView;

import android.widget.Spinner;

public class MainActivity extends ActionBarActivity {

private ImageView mImageView;

private Spinner mScaleSpinner;

private Bitmap mBitmap;

private static final String[] sScales = { "0.2x", "0.5x", "1.0x", "2.0x" };

private static final float[] sFloatScales = { 0.2F, 0.5F, 1F, 2F };

private final int DEFAULT\_SCALE = 2;

private float mCurScale;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

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

mScaleSpinner = (Spinner) findViewById(R.id.spinner);

ArrayAdapter<String> adapter = new ArrayAdapter<>(this,

android.R.layout.simple\_spinner\_item, sScales);

adapter

.setDropDownViewResource(android.R.layout.simple\_spinner\_dropdown\_item);

mScaleSpinner.setAdapter(adapter);

mScaleSpinner.setSelection(DEFAULT\_SCALE);

mCurScale = sFloatScales[DEFAULT\_SCALE];

BitmapDrawable bitmapDrawable = (BitmapDrawable) mImageView.getDrawable();

mBitmap = bitmapDrawable.getBitmap();

mScaleSpinner

.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

public void onItemSelected(AdapterView<?> parent,

View itemSelected, int selectedItemPosition,

long selectedId) {

mCurScale = sFloatScales[selectedItemPosition];

mBitmap = resizeImage(mBitmap);

mImageView.setImageBitmap(mBitmap);

}

public void onNothingSelected(AdapterView<?> parent) {

mScaleSpinner.setSelection(DEFAULT\_SCALE);

mCurScale = sFloatScales[DEFAULT\_SCALE];

}

});

}

private Bitmap resizeImage(Bitmap src) {

Matrix matrix = new Matrix();

matrix.postScale(mCurScale, mCurScale);

Bitmap resizedBitmap;

resizedBitmap = Bitmap.createBitmap(mBitmap, 0, 0,

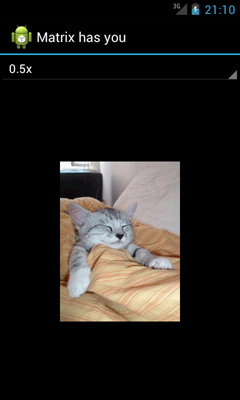
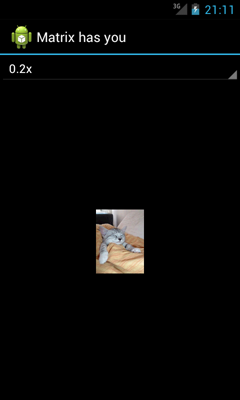
src.getWidth(), src.getHeight(), matrix, true);

return resizedBitmap;

}

}

С помощью **Spinner** мы можем уменьшить картинку в два раза (значение 0.5x) или в 5 раз (значение 0.2x), а также увеличить в два раза (2.0x).

Метод postRotate()

Метод **postRotate()** позволяет вращать изображение на заданный угол. Добавим к предыдущему примеру кнопку и напишем код:

private Bitmap rotateImage(Bitmap src) {

Matrix matrix = new Matrix();

matrix.postRotate(30f); // поворачиваем на 30 градусов

Bitmap resultBitmap = Bitmap.createBitmap(bitmap, 0, 0, src.getWidth(),

src.getHeight(), matrix, true);

return resultBitmap;

}

// щелчок кнопки

public void onClick(View v) {

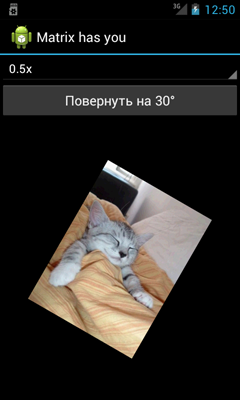
bitmap = rotateImage(bitmap);

myImageView.setImageBitmap(bitmap);

myImageView.setScaleType(ScaleType.CENTER);

}

Каждый новый щелчок по кнопке поворачивает изображение ещё на 30 градусов (а через пять-шесть щелчков у меня приложение вылетает с ошибкой, пичалька).



Метод setRotate()

Аналогичный метод.

Matrix matrix = new Matrix();

matrix.setRotate(15);

canvas.drawBitmap(bitmap, matrix, paint);

Сам поворот происходит через точку с координатами 0, 0. Вы можете вызвать другую версию метода, чтобы задать свою точку поворота, например, в центре.

matrix.setRotate(15, bitmap.getWidth() / 2, bitmap.getHeight() / 2);

Метод postSkew()

Метод **postSkew()** позволяет наклонять изображение. Пример без проверок, возможны сбои.

<LinearLayout 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"

android:orientation="vertical"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<LinearLayout

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_margin="5dp"

android:layout\_weight="1"

android:orientation="vertical">

<TextView

android:id="@+id/textViewSkewX"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Skew-X: 0"

android:textSize="12sp" />

<SeekBar

android:id="@+id/seekBarSkewX"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="200"

android:progress="100" />

</LinearLayout>

<LinearLayout

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_margin="5dp"

android:layout\_weight="1"

android:orientation="vertical">

<TextView

android:id="@+id/textViewSkewY"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Skew-Y: 0"

android:textSize="12sp" />

<SeekBar

android:id="@+id/seekBarSkewY"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="200"

android:progress="100" />

</LinearLayout>

</LinearLayout>

<ImageView

android:id="@+id/imageView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:src="@drawable/sleepbedcat" />

</LinearLayout>

Теперь код:

// Если этот код работает, его написал Александр Климов,

// а если нет, то не знаю, кто его писал.

package ru.alexanderklimov.testapplication;

import android.graphics.Bitmap;

import android.graphics.Matrix;

import android.graphics.drawable.BitmapDrawable;

import android.os.Bundle;

import android.support.v7.app.ActionBarActivity;

import android.widget.ImageView;

import android.widget.SeekBar;

import android.widget.TextView;

public class MainActivity extends ActionBarActivity {

private ImageView mImageView;

private Bitmap mBitmap;

TextView mSkewXTextView, mSkewYTextView;

private float mCurSkewX = 0F;

private float mCurSkewY = 0F;

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

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

SeekBar skewXSeekBar = (SeekBar) findViewById(R.id.seekBarSkewX);

SeekBar skewYSeekBar = (SeekBar) findViewById(R.id.seekBarSkewY);

mSkewXTextView = (TextView) findViewById(R.id.textViewSkewX);

mSkewYTextView = (TextView) findViewById(R.id.textViewSkewY);

BitmapDrawable bitmapDrawable = (BitmapDrawable) mImageView.getDrawable();

mBitmap = bitmapDrawable.getBitmap();

skewXSeekBar

.setOnSeekBarChangeListener(seekbarSkewXSeekBarChangeListener);

skewYSeekBar

.setOnSeekBarChangeListener(seekbarSkewYSeekBarChangeListener);

}

private Bitmap skewImage(Bitmap src){

Matrix matrix = new Matrix();

matrix.postSkew(mCurSkewX, mCurSkewY);

Bitmap resultBitmap;

resultBitmap = Bitmap.createBitmap(mBitmap, 0, 0, src.getWidth(),

src.getHeight(), matrix, true);

return resultBitmap;

}

private SeekBar.OnSeekBarChangeListener seekbarSkewYSeekBarChangeListener = new SeekBar.OnSeekBarChangeListener() {

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

}

@Override

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

mCurSkewY = (float) (progress - 100) / 100;

mSkewYTextView.setText("Skew-Y: " + String.valueOf(mCurSkewY));

mBitmap = skewImage(mBitmap);

mImageView.setImageBitmap(mBitmap);

}

};

private SeekBar.OnSeekBarChangeListener seekbarSkewXSeekBarChangeListener = new SeekBar.OnSeekBarChangeListener() {

@Override

public void onProgressChanged(SeekBar seekBar, int progress,

boolean fromUser) {

mCurSkewX = (float) (progress - 100) / 100;

mSkewXTextView.setText("Skew-X: " + String.valueOf(mCurSkewX));

mBitmap = skewImage(mBitmap);

mImageView.setImageBitmap(mBitmap);

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

}

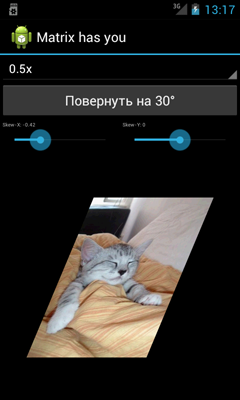
@Override

public void onStopTrackingTouch(SeekBar seekBar) {

}

};

}



Метод setTranslate()

Метод просто перемещает картинку на заданные значения. Первый аргумент сдвигает по оси X, второй - по оси Y.

setTranslate(1.5f, -10);

Реклама