**Цветовые фильтры PorterDuffColorFilter, LightingColorFilter, ColorMatrixColorFilter**

[PorterDuffColorFilter](http://developer.alexanderklimov.ru/android/catshop/colorfilters.php" \l "porterduffcolorfilter)  
[LightingColorFilter](http://developer.alexanderklimov.ru/android/catshop/colorfilters.php#lightingcolorfilter)  
[ColorMatrixColorFilter](http://developer.alexanderklimov.ru/android/catshop/colorfilters.php#colormatrixcolorfilter)

В составе пакета **android.graphics** входит несколько готовых цветовых фильтров, которые можно применить к графическим объектам. В отличие от фильтров для масок, которые преобразуют альфа-канал объекта Paint, цветовые фильтры затрагивают каждый из каналов RGB. Все потомки класса **ColorFilter** игнорируют альфа-канал во время преобразований

Android содержит три цветовых фильтра:

* **ColorMatrixColorFilter**. Позволяет задать для объекта **Paint** матрицу **ColorMatrix** размером 4x5. Объекты **ColorMatrix**, как правило, используются при программной обработке изображений, могут пригодиться также для последовательных преобразований с применением умножения матриц.
* **LightingColorFilter**. Умножает каналы RGB первого цвета, прежде чем добавить второй. Результат каждого преобразования варьируется от 0 до 255.
* **PorterDuffColorFilter**. Предлагает воспользоваться одним из шестнадцати режимов смешивания цифровых изображений Портера-Даффа, чтобы применить заданный цвет к объекту Paint.

Использовать цветовые фильтры можно, задействовав для этого метод **setColorFilter()**:

myPaint.setColorFilter(new LightingColorFilter(Color.BLUE, Color.RED));

**PorterDuffColorFilter**

Однажды мы применяли этот фильтр (посмотрите урок [Обои главного экрана](http://developer.alexanderklimov.ru/android/wallpaper.php). Давайте ещё раз его рассмотрим фильтр отдельно.

Имеется 18 различных констант **PorterDuff.Mode**, позволяющих добиться различных эффектов: SCREEN, OVERLAY, LIGHTEN, DARKEN, MULTIPLY, ADD и др.

Добавим на экран текстовую метку для отображения имени фильтра, кнопку для запуска эффекта и ImageView для картинки. В качестве изображения возьмём уже знакомую картинку спящего котёнка.

package ru.alexanderklimov.colofilters;

import android.os.Bundle;

import android.app.Activity;

import android.graphics.Color;

import android.graphics.PorterDuff;

import android.graphics.PorterDuffColorFilter;

import android.graphics.drawable.BitmapDrawable;

import android.view.Menu;

import android.view.View;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.Toast;

public class MainActivity extends Activity {

TextView tvFilterName;

ImageView ivCat;

final static private int[] mColors = { Color.BLUE, Color.GREEN, Color.RED,

Color.LTGRAY, Color.MAGENTA, Color.CYAN, Color.YELLOW, Color.WHITE };

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

tvFilterName = (TextView) findViewById(R.id.tvFilterName);

ivCat = (ImageView) findViewById(R.id.ivCat);

}

public void onClick(View v) {

tvFilterName.setText("PorterDuffColorFilter");

BitmapDrawable mydrawable = (BitmapDrawable) ivCat.getDrawable();

int mColor = (int) Math.floor(Math.random() \* mColors.length);

PorterDuffColorFilter porterduffcolorfilter = new PorterDuffColorFilter(

mColors[mColor], PorterDuff.Mode.MULTIPLY);

mydrawable.setColorFilter(porterduffcolorfilter);

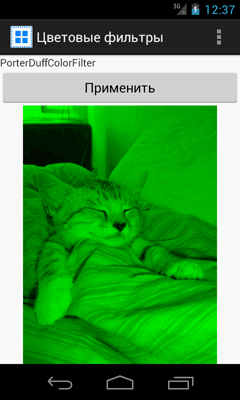
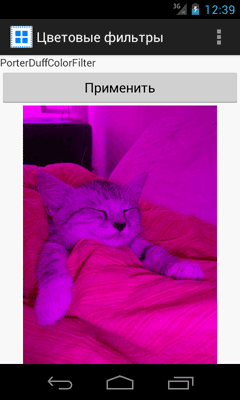
ivCat.setImageDrawable(mydrawable);

ivCat.invalidate();

}

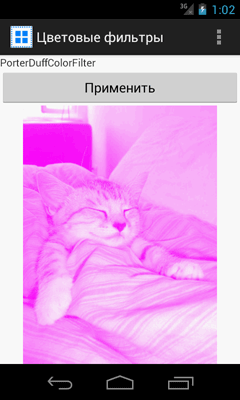
}

У фильтра **PorterDuffColorFilter** во втором параметре есть много вариантов. Для примера используется вариант **PorterDuff.Mode.MULTIPLY**, который даёт интересные эффекты с выбранным цветом, который выбирается случайным образом из массива.

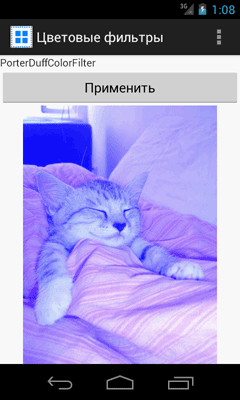
Формула для **MULTIPLY** - [ Sa\*Da, Sc\*Dc ], где a - цвет канала, c - цвет, S - Source, D - Destination.

Если использовать параметр **PorterDuff.Mode.ADD**, то получим эффект Saturate (насыщенность)



Формула добавления насыщенности: Saturate(S+D).

Вариант **PorterDuff.Mode.LIGHTEN** может дать следующий вариант:



**LIGHTEN** работает по формуле [ Sa+Da - Sa\*Da, Sc\*(1-Da) + Dc\*(1-Sa) + max(Sc, Dc) ].

Константа **DARKEN** затемняет итоговую картинку по формуле [ Sa+Da - Sa\*Da, Sc\*(1-Da) + Dc\*(1-Sa) + min(Sc, Dc) ].

Константа **SCREEN** работает по формуле [ Sa+Da - Sa\*Da, Sc+Dc - Sc\*Dc ].

Остальные варианты попробуйте самостоятельно.

Посмотрите также пример [Android-er: PorterDuffColorFilter demonstration](http://android-er.blogspot.ru/2012/09/porterduffcolorfilter-demonstration.html#1098)

Фильтр можно применить не только к изображению, но и кнопке и другим компонентам:

button1.getBackground().setColorFilter(filter1);

**LightingColorFilter**

Рассмотрим другой фильтр **LightingColorFilter**, который умножает RGB-каналы первого цвета, прежде чем добавить второй. Результат каждого преобразования варьируется от 0 до 255. Заменим код из предыдущего примера для обработчика щелчка кнопки:

public void onClick(View v) {

tvFilterName.setText("LightingColorFilter");

BitmapDrawable mydrawable = (BitmapDrawable) ivCat.getDrawable();

int mul = 0xFFFFFF00; // умножаем синий цвет

int add = 0x0000FF00; // устанавливаем зеленый

LightingColorFilter lightingColorFilter = new LightingColorFilter(mul,

add);

mydrawable.setColorFilter(lightingColorFilter);

ivCat.setImageDrawable(mydrawable);

ivCat.invalidate();

}



Также можете посмотреть пример [Adjust lightness and darkness of ImageView, using LightingColorFilter](http://android-er.blogspot.ru/2012/09/adjust-lightness-and-darkness-of.html) с множеством настроек.

**ColorMatrixColorFilter**

Примеры с **ColorMatrixColorFilter** представлены в статье [ColorMatrix](http://developer.alexanderklimov.ru/android/catshop/colormatrix.php).