**Цветовая модель HSV**

Закрытая зона/3-й месяц

О том, что такое цветовая модель HSV можно прочитать в [Википедии](http://ru.wikipedia.org/wiki/HSV_(%D1%86%D0%B2%D0%B5%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C)). Мы же рассмотрим примеры.

Цветовая модель "тон-насыщенность-значение" (hue-saturation-value - HSV) представляет собой альтернативу модели RGB (красный-зеленый-синий) для указания определённого цвета. Цвет в модели HSV представляет собой цветовой круг и его можно задать числом в диапазоне от 0.0 до 1.0.

В Android есть специальные методы **colorToHSV()** и **HSVToColor()**, позволяющие работать с HSV. Напишем пример, загружающий картинку с сайта и преобразующий его в модель HSV.

<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" >

<ImageView

android:id="@+id/imgsource"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

<ImageView

android:id="@+id/imgtarget"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

</LinearLayout>

Код, загружающий картинку в отдельном потоке:

package ru.alexanderklimov.test;

import java.io.IOException;

import java.net.MalformedURLException;

import java.net.URL;

import android.os.AsyncTask;

import android.os.Bundle;

import android.app.Activity;

import android.graphics.Bitmap;

import android.graphics.Bitmap.Config;

import android.graphics.BitmapFactory;

import android.graphics.Color;

import android.widget.ImageView;

public class TestActivity extends Activity {

ImageView imgSource, imgTarget;

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_test);

imgSource = (ImageView) findViewById(R.id.imgsource);

imgTarget = (ImageView) findViewById(R.id.imgtarget);

// Load bitmap from internet

String onLineImgSource = "http://developer.alexanderklimov.ru/images/fruitcat.jpg";

URL urlImgSource;

try {

urlImgSource = new URL(onLineImgSource);

new MyNetworkTask(imgSource, imgTarget).execute(urlImgSource);

} catch (MalformedURLException e) {

e.printStackTrace();

}

}

private class MyNetworkTask extends AsyncTask<URL, Void, Bitmap> {

ImageView ivSource, ivTarget;

public MyNetworkTask(ImageView iSource, ImageView iTarget) {

ivSource = iSource;

ivTarget = iTarget;

}

@Override

protected Bitmap doInBackground(URL... urls) {

Bitmap networkBitmap = null;

URL networkUrl = urls[0]; // Load the first element

try {

networkBitmap = BitmapFactory.decodeStream(networkUrl

.openConnection().getInputStream());

} catch (IOException e) {

e.printStackTrace();

}

return networkBitmap;

}

@Override

protected void onPostExecute(Bitmap result) {

ivSource.setImageBitmap(result);

ivTarget.setImageBitmap(convertColorHSVColor(result));

}

}

// Convert Bitmap from Color to HSV, then HSV to Color

private Bitmap convertColorHSVColor(Bitmap src) {

int w = src.getWidth();

int h = src.getHeight();

int[] mapSrcColor = new int[w \* h];

int[] mapDestColor = new int[w \* h];

float[] pixelHSV = new float[3];

/\*

\* pixelHSV[0] : Hue (0 .. 360) pixelHSV[1] : Saturation (0...1)

\* pixelHSV[2] : Value (0...1)

\*/

src.getPixels(mapSrcColor, 0, w, 0, 0, w, h);

/\*

\* getPixels (int[] pixels, int offset, int stride, int x, int y, int

\* width, int height) - Returns in pixels[] a copy of the data in the

\* bitmap. Each value is a packed int representing a Color.

\*

\* pixels: The array to receive the bitmap's colors offset: The first

\* index to write into pixels[] stride: The number of entries in

\* pixels[] to skip between rows (must be >= bitmap's width). Can be

\* negative. x: The x coordinate of the first pixel to read from the

\* bitmap y: The y coordinate of the first pixel to read from the bitmap

\* width: The number of pixels to read from each row height: The number

\* of rows to read

\*/

int index = 0;

for (int y = 0; y < h; ++y) {

for (int x = 0; x < w; ++x) {

// Convert from Color to HSV

Color.colorToHSV(mapSrcColor[index], pixelHSV);

// Convert back from HSV to Color

mapDestColor[index] = Color.HSVToColor(pixelHSV);

index++;

}

}

Config destConfig = src.getConfig();

/\*

\* If the bitmap's internal config is in one of the public formats,

\* return that config, otherwise return null.

\*/

if (destConfig == null) {

destConfig = Config.RGB\_565;

}

Bitmap newBitmap = Bitmap.createBitmap(mapDestColor, w, h, destConfig);

return newBitmap;

}

}

Не забываем про разрешение **android.permission.INTERNET** в манифесте.

Это был базовый пример. Теперь попробуем поиграть с параметрами HSV - Hue (тон), Saturation (насышенность), Value (значение).

<HorizontalScrollView 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" >

<LinearLayout

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:orientation="horizontal" >

<ImageView

android:id="@+id/imgsource"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

<ImageView

android:id="@+id/imgtarget"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

<ImageView

android:id="@+id/imghue"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

<ImageView

android:id="@+id/imgsat"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

<ImageView

android:id="@+id/imgval"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

</LinearLayout>

</HorizontalScrollView>

Внесём изменения в код:

package ru.alexanderklimov.test;

import java.io.IOException;

import java.net.MalformedURLException;

import java.net.URL;

import android.os.AsyncTask;

import android.os.Bundle;

import android.app.Activity;

import android.graphics.Bitmap;

import android.graphics.Bitmap.Config;

import android.graphics.BitmapFactory;

import android.graphics.Color;

import android.widget.ImageView;

public class TestActivity extends Activity {

ImageView imgSource;

ImageView imgTarget;

ImageView imgHue, imgSat, imgVal;

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_test);

imgSource = (ImageView) findViewById(R.id.imgsource);

imgTarget = (ImageView) findViewById(R.id.imgtarget);

imgHue = (ImageView) findViewById(R.id.imghue);

imgSat = (ImageView) findViewById(R.id.imgsat);

imgVal = (ImageView) findViewById(R.id.imgval);

// Load bitmap from internet

String onLineImgSource = "http://developer.alexanderklimov.ru/images/fruitcat.jpg";

URL urlImgSource;

try {

urlImgSource = new URL(onLineImgSource);

new MyNetworkTask(imgSource, imgTarget, imgHue, imgSat, imgVal)

.execute(urlImgSource);

} catch (MalformedURLException e) {

e.printStackTrace();

}

}

private class MyNetworkTask extends AsyncTask<URL, Void, Bitmap> {

ImageView ivSource, ivTarget;

ImageView ivHue, ivSat, ivVal;

public MyNetworkTask(ImageView iSource, ImageView iTarget,

ImageView iHue, ImageView iSat, ImageView iVal) {

ivSource = iSource;

ivTarget = iTarget;

ivHue = iHue;

ivSat = iSat;

ivVal = iVal;

}

@Override

protected Bitmap doInBackground(URL... urls) {

Bitmap networkBitmap = null;

URL networkUrl = urls[0]; // Load the first element

try {

networkBitmap = BitmapFactory.decodeStream(networkUrl

.openConnection().getInputStream());

} catch (IOException e) {

e.printStackTrace();

}

return networkBitmap;

}

@Override

protected void onPostExecute(Bitmap result) {

ivSource.setImageBitmap(result);

GroupBitmap groupBMResult = convertColorHSVColor(result);

ivTarget.setImageBitmap(groupBMResult.bitmapDest);

ivHue.setImageBitmap(groupBMResult.bitmapHue);

ivSat.setImageBitmap(groupBMResult.bitmapSat);

ivVal.setImageBitmap(groupBMResult.bitmapVal);

}

}

class GroupBitmap {

Bitmap bitmapHue;

Bitmap bitmapSat;

Bitmap bitmapVal;

Bitmap bitmapDest;

};

// Convert Bitmap from Color to HSV, then HSV to Color

private GroupBitmap convertColorHSVColor(Bitmap src) {

GroupBitmap convertedGroupBitmap = new GroupBitmap();

int w = src.getWidth();

int h = src.getHeight();

int[] mapSrcColor = new int[w \* h];

int[] mapDestColor = new int[w \* h];

int[] mapHue = new int[w \* h];

int[] mapSat = new int[w \* h];

int[] mapVal = new int[w \* h];

float[] pixelHSV = new float[3];

/\*

\* pixelHSV[0] : Hue (0 .. 360) pixelHSV[1] : Saturation (0...1)

\* pixelHSV[2] : Value (0...1)

\*/

src.getPixels(mapSrcColor, 0, w, 0, 0, w, h);

/\*

\* getPixels (int[] pixels, int offset, int stride, int x, int y, int

\* width, int height) - Returns in pixels[] a copy of the data in the

\* bitmap. Each value is a packed int representing a Color.

\*

\* pixels: The array to receive the bitmap's colors offset: The first

\* index to write into pixels[] stride: The number of entries in

\* pixels[] to skip between rows (must be >= bitmap's width). Can be

\* negative. x: The x coordinate of the first pixel to read from the

\* bitmap y: The y coordinate of the first pixel to read from the bitmap

\* width: The number of pixels to read from each row height: The number

\* of rows to read

\*/

int index = 0;

for (int y = 0; y < h; ++y) {

for (int x = 0; x < w; ++x) {

// Convert from Color to HSV

Color.colorToHSV(mapSrcColor[index], pixelHSV);

/\*

\* Represent Hue, Saturation and Value in separated color of R,

\* G, B.

\*/

mapHue[index] = Color

.rgb((int) (pixelHSV[0] \* 255 / 360), 0, 0);

mapSat[index] = Color.rgb(0, (int) (pixelHSV[1] \* 255), 0);

mapVal[index] = Color.rgb(0, 0, (int) (pixelHSV[2] \* 255));

// Convert back from HSV to Color

mapDestColor[index] = Color.HSVToColor(pixelHSV);

index++;

}

}

Config destConfig = src.getConfig();

/\*

\* If the bitmap's internal config is in one of the public formats,

\* return that config, otherwise return null.

\*/

if (destConfig == null) {

destConfig = Config.RGB\_565;

}

convertedGroupBitmap.bitmapHue = Bitmap.createBitmap(mapHue, w, h,

Config.RGB\_565);

convertedGroupBitmap.bitmapSat = Bitmap.createBitmap(mapSat, w, h,

Config.RGB\_565);

convertedGroupBitmap.bitmapVal = Bitmap.createBitmap(mapVal, w, h,

Config.RGB\_565);

convertedGroupBitmap.bitmapDest = Bitmap.createBitmap(mapDestColor, w,

h, destConfig);

return convertedGroupBitmap;

}

}

Если вас интересует динамическое изменение параметров, то ниже последняя ссылка ведёт на подобный пример.

Источники: [Android-er: Example of using colorToHSV() and HSVToColor(), convert between argb color and HSV components.](http://android-er.blogspot.ru/2012/10/example-of-using-colortohsv-and.html)  
[Android-er: What's HSV (hue, saturation, value)?](http://android-er.blogspot.ru/2012/10/whats-hsv-hue-saturation-value.html)  
[Android-er: Adjust hue, saturation, and brightness by changing of HSV](http://android-er.blogspot.ru/2012/10/adjust-hue-saturation-and-brightness-by.html)

Также посмотрите пример [Adjust HSV(Hue, Saturation and Value) of bitmap](http://android-er.blogspot.ru/2013/09/adjust-hsvhue-saturation-and-value-of.html) для динамического управления параметрами.

