**Управляем цветами**

Вы уже знаете, что цвет задаётся шестнадцатеричным значением в виде #xxxxxxxx. Напишем приложение, где пользователь может вводить нужное число, чтобы увидеть получившийся цвет. В статье используются различные техники, которые могут пригодиться в других приложениях.

Создадим разметку следующего вида.

<RelativeLayout 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: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" >

<TextView

android:id="@+id/tv\_colorpreview"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Color:"

android:textAppearance="?android:attr/textAppearanceLarge" />

<View

android:id="@+id/v\_border"

android:layout\_width="130dp"

android:layout\_height="130dp"

android:layout\_below="@id/tv\_colorpreview"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="12dp"

android:background="#FF000000" />

<TextView

android:id="@+id/tv\_inputcolor"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignLeft="@+id/tv\_colorpreview"

android:layout\_below="@+id/v\_border"

android:layout\_marginTop="15dp"

android:text="Type a color (ARGB, Hex):"

android:textAppearance="?android:attr/textAppearanceMedium" />

<EditText

android:id="@+id/hexcolor"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignLeft="@+id/tv\_inputcolor"

android:layout\_below="@+id/tv\_inputcolor"

android:layout\_marginTop="10dp"

android:ems="8"

android:inputType="textCapCharacters|textNoSuggestions|textVisiblePassword"

android:maxEms="8"

android:maxLength="8"

android:maxLines="1"

android:minLines="1"

android:text="FFFFFFFF" />

<ImageView

android:id="@+id/colorpreview"

android:layout\_width="130dp"

android:layout\_height="130dp"

android:layout\_alignBottom="@+id/v\_border"

android:layout\_alignLeft="@+id/v\_border"

android:layout\_alignRight="@+id/v\_border"

android:layout\_alignTop="@+id/v\_border"

android:layout\_margin="1dp"

android:background="@drawable/checkered\_pattern\_bg"

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

<Button

android:id="@+id/buttonOk"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignBottom="@+id/hexcolor"

android:layout\_alignTop="@+id/hexcolor"

android:layout\_marginLeft="7dp"

android:layout\_toRightOf="@+id/hexcolor"

android:text="OK" />

</RelativeLayout>

Обратите внимание на атрибуты у компонента **EditText**:

android:inputType="textCapCharacters|textNoSuggestions|textVisiblePassword"

android:maxLength="8"

Атрибут **android:inputType** позволяет показать нужный тип клавиатуры. В нашем случае клавиатура будет с символами верхнего регистра (**textCapCharacters**), без подсказок при вводе текста (**textNoSuggestions**) и без автокоррекции (**textVisiblePassword**). Атрибут **android:maxLength**устанавливает максимально допустимое число символов в текстовом поле. Для цвета достаточно восьми символов.

Для просмотра получившегося цвета используется **ImageView**, который содержит белую непрозрачную картинку размером 1×1 пиксель (0xFFFFFFFF) из файла **singlepixelwhite.png**. Картинка растягивается по размеру ImageView. Подобный способ позволит нам просматривать заданный цвет, включая прозрачность, накладывая цвет на белую картинку.

Теперь код для приложения:

package ru.alexanderklimov.hexadecimalcolor;

import android.os.Bundle;

import android.app.Activity;

import android.graphics.PorterDuff;

import android.text.Editable;

import android.text.TextWatcher;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.view.WindowManager;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.Toast;

public class MainActivity extends Activity {

private ImageView ivColorPreview;

private EditText editHexColorInput;

private Button buttonOk;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

ivColorPreview = (ImageView) findViewById(R.id.colorpreview);

editHexColorInput = (EditText) findViewById(R.id.hexcolor);

buttonOk = (Button) findViewById(R.id.buttonOk);

// This prevents the EditText View from automatically receiving focus on

// Activity startup

getWindow().setSoftInputMode(

WindowManager.LayoutParams.SOFT\_INPUT\_STATE\_ALWAYS\_HIDDEN);

// Assign a TextWatcher to the EditText

editHexColorInput.addTextChangedListener(new TextWatcher() {

@Override

public void onTextChanged(CharSequence s, int start, int before,

int count) {

}

@Override

public void beforeTextChanged(CharSequence s, int start, int count,

int after) {

}

// Right after the text is changed

@Override

public void afterTextChanged(Editable s) {

// Store the text on a String

String text = s.toString();

// Get the length of the String

int length = s.length();

/\*

\* If the String length is bigger than zero and it's not

\* composed only by the following characters: A to F and/or 0 to

\* 9

\*/

if (!text.matches("[a-fA-F0-9]+") && length > 0) {

// Delete the last character

s.delete(length - 1, length);

}

}

});

buttonOk.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

// Create and initialize the int 'color' with the white color.

int color = 0xFFFFFFFF;

try {

// Get the text at the EditText field and save it in a

// String

String colorValue = editHexColorInput.getText().toString();

/\* Convert the String into a Color ARGB hexadecimal integer. \*/

color = Integer.parseInt(colorValue.substring(2), 16)

+ (Integer.parseInt(colorValue.substring(0, 2), 16) << 24);

} catch (Exception e) // Something went wrong while parsing the

// String into an Integer

{

// Reset the EditText field to white

editHexColorInput.setText("FFFFFFFF");

// Display a toast message

Toast.makeText(MainActivity.this, "Invalid color!",

Toast.LENGTH\_SHORT).show();

}

/\*

\* Get the Drawable displayed by the Image View (in this case,

\* the 1x1 PNG opaque white image). Clear any ColorFilters

\* previously applied to it

\*/

ivColorPreview.getDrawable().clearColorFilter();

// Multiply the white image with the hexadecimal color passed at

// the EditText field

ivColorPreview.getDrawable().setColorFilter(color,

PorterDuff.Mode.MULTIPLY);

// Invalidate the ImageView, forcing it to be re-rendered.

ivColorPreview.invalidate();

}

});

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

// If the only button at this options menu is pressed

case R.id.action\_settings: {

// Remove any kind of color filter

ivColorPreview.getDrawable().clearColorFilter();

// Render the ImageView again

ivColorPreview.invalidate();

// Reset the color to its initial value (White = 0xFFFFFFFF)

editHexColorInput.setText("FFFFFFFF");

return true;

}

default: {

return super.onOptionsItemSelected(item);

}

}

}

}

Чтобы виртуальная клавиатура не появлялась при запуске автоматически, используется вызов **setSoftInputMode()** с нужным флагом. Это даёт возможность пользователю увидеть программу без лишних деталей, которые могут помешать понять логику приложения.

К текстовому полю подключается интерфейс **TextWatcher**, позволяющий отслеживать ввод символов. В нашем случае нас интересует метод **afterTextChanged()**, реагирующий на событие после ввода символа. Наша задача - не допустить ввод лишних символов по указанному регулярному выражению.

Если строка не содержит символы от A до F (в любом регистре) и/или цифры от 0 до 9, то последний символ, введённый пользователем, удаляется.

При нажатии кнопки текст из текстового поля преобразуется в шестнадцатеричное целое число и затем полученное значение используется для создания нужного цвета.

Преобразование происходит через метод **Integer.ParseInt()**. Во избежание ошибок используется блок **try/catch** с перехватом ошибок. Обратите внимание, что преобразование состоит из двух частей:

color = Integer.parseInt(colorValue.substring(2), 16) +

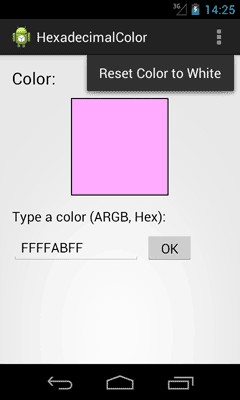
(Integer.parseInt(colorValue.substring(0,2), 16) << 24);

Сначала получаем значение RGB-каналов, а потом значение прозрачности (первые два символа). Альфа-канал сдвигается на 24 бита влево, чтобы они оказались в начале шестнадцатеричного значения.

Представьте себе, что мы задали непрозрачный чёрный цвет значением 0xFF000000. За альфа-канал отвечают символы FF. После преобразования в шестнадцатеричное число мы получим десятичное число 255. Но, вспомните, что у нас был цвет 0xFF000000 (Десятичное число: 4278190080), а не 0x000000FF (Десятичное число: 255). Поэтому мы и сдвигаем символы влево.

Сам цвет создаётся через методы **getDrawable()**, **xxxColorFilter()**. Режим **PorterDuff.Mode** позволяет смешивать два значения и получить различные результаты. Например, **PorterDuff.Mode.MULTIPLY** позволяет получить нужный цвет.

Для быстрого сброса предусмотрен пункт меню, который позволят быстро установить начальный белый цвет.



Источник (с исходниками): [Android: hexadecimal color input using an EditText](http://www.41post.com/5108/programming/android-hexadecimal-color-input-using-an-edittext)