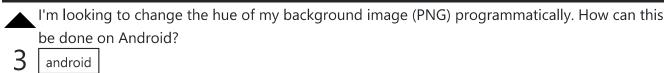
Android: How to change hue of an Image?

Asked 10 years, 5 months agoModified 2 years, 3 months agoViewed 8k times





ShareFollow



asked Apr 20, 2012 at 23:03

Check this post stackoverflow.com/questions/4354939/... – JRaymond Apr 20, 2012 at 23:07 🖍

Sorted by:

3 Answers Highest score (default)

I tested the accepted answer, unfortunately it returns a wrong result. I found and modified this code from here which works fine:

```
6
    // hue-range: [0, 360] -> Default = 0
```

public static Bitmap hue(Bitmap bitmap, float hue) { Bitmap newBitmap = bitmap.copy(bitmap.getConfig(), true); final int width = newBitmap.getWidth(); final int height = newBitmap.getHeight(); float [] hsv = new float[3]; for(int y = 0; y < height; y++){ for(int x = 0; x < width; x++){

int pixel = newBitmap.getPixel(x,y);

Color.colorToHSV(pixel,hsv); hsv[0] = hue;newBitmap.setPixel(x,y,Color.HSVToColor(Color.alpha(pixel),hsv));} } bitmap.recycle(); bitmap = null;return newBitmap;

ShareFollow

answered Sep 14, 2014 at 14:08 user1922137

This code works but is very slow. Any idea to optimize? - Sujay U N Feb 25, 2018 at 18:31

The linked post has some good ideas, but the matrix math used for ColorFilter may be (a) complex overkill, and (b) introduce perceptible shifts in the resulting colors.



Modifying the solution given by janin here - https://stackoverflow.com/a/6222023/1303595 I've based this version on Photoshop's 'Color' blend mode. It seems to avoid the image-darkening caused by PorterDuff.Mode.Multiply, and works very well for color-tinting desaturated/artificial-Black & White images without losing much contrast.

```
* Going for perceptual intent, rather than strict hue-only change.
* This variant based on Photoshop's 'Color' blending mode should look
* better for tinting greyscale images and applying an all-over color
* without tweaking the contrast (much)
* Final color = Target.Hue, Target.Saturation, Source.Luma
* Drawback is that the back-and-forth color conversion introduces some
* error each time.
*/
public void changeHue (Bitmap bitmap, int hue, int width, int height) {
  if (bitmap == null) { return; }
  if ((hue < 0) || (hue > 360)) { return; }
  int size = width * height;
  int[] all_pixels = new int [size];
  int top = 0;
  int left = 0;
  int offset = 0;
  int stride = width;
   bitmap.getPixels (all_pixels, offset, stride, top, left, width, height);
  int pixeI = 0;
  int alpha = 0;
  float[] hsv = new float[3];
  for (int i=0; i < size; i++) {
      pixel = all_pixels [i];
      alpha = Color.alpha (pixel);
      Color.colorToHSV (pixel, hsv);
     // You could specify target color including Saturation for
     // more precise results
      hsv [0] = hue;
      hsv[1] = 1.0f;
      all_pixels [i] = Color.HSVToColor (alpha, hsv);
  }
   bitmap.setPixels (all_pixels, offset, stride, top, left, width, height);
```

ShareFollow







▲ If you wrap your Bitmap in an ImageView there is a very simple way:

nageView circle = new ImageView(this); circle.setImageBitmap(yourBitmap);



My guess is this will be faster than modifying each pixel individually. ShareFollow

answered Jun 17, 2020 at 19:14



10.3k 5