Android-Bildverarbeitung einfach gemacht

Bearbeitung von Bildern in Android kann knifflig sein –Bitmaps verbrauchen Speicher, Drehungen verzerren die Ausrichtung und Zuschneiden erfordert Präzision. Hier kommen Hilfsklassen wie `BitmapUtils` und `Crop` ins Spiel. In diesem Beitrag werde ich Sie durch ein mächtiges Paar von Klassen aus dem `com.lzw.flower.utils`-Paket führen. Wir werden ihren Code durchgehen, jede Methode analysieren und zeigen, wie man sie in Ihren Projekten verwendet. Lassen Sie uns eintauchen!

BitmapUtils: Ihr Allround-Toolkit für Bildmanipulationen Die `BitmapUtils`-Klasse ist eine Sammlung statischer Methoden zur Manipulation von `Bitmap`-Objekten. Hier ist der vollständige Code, gefolgt von einer Analyse:

```
package com.lzw.flower.utils;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Matrix;
import android.net.Uri;
import android.provider.MediaStore;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

public class BitmapUtils {
  public static Bitmap convertGreyImg(Bitmap img) {
    int width = img.getWidth();
    int height = img.getHeight();
    int[] pixels = new int[width * height];
    img.getPixels(pixels, 0, width, 0, 0, width, height);
    int alpha = 0xFF << 24;
    for (int i = 0; i < height; i++) {
      for (int j = 0; j < width; j++) {
        int grey = pixels[width * i + j];
        int red = ((grey & 0x00FF0000) >> 16);
        int green = ((grey & 0x0000FF00) >> 8);
        int blue = (grey & 0x000000FF);
```

```
      grey = (int) ((float) red * 0.3 + (float) green * 0.59 + (float) blue * 0.11);
      grey = alpha | (grey << 16) | (grey << 8) | grey;
      pixels[width * i + j] = grey;
    }
  }
  Bitmap result = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
  result.setPixels(pixels, 0, width, 0, 0, width, height);
  return result;
}

public static Bitmap toGreyImg(Bitmap bitmapOrg) {
  Bitmap bitmapNew = bitmapOrg.copy(Bitmap.Config.ARGB_8888, true);
  if (bitmapNew == null) {
    return null;
  }
  for (int i = 0; i < bitmapNew.getWidth(); i++) {
    for (int j = 0; j < bitmapNew.getHeight(); j++) {
      int col = bitmapNew.getPixel(i, j);
      int alpha = col & 0xFF000000;
      int red = (col & 0x00FF0000) >> 16;
      int green = (col & 0x0000FF00) >> 8;
      int blue = (col & 0x000000FF);
      int gray = (int) ((float) red * 0.3 + (float) green * 0.59 + (float) blue * 0.11);
      int newColor = alpha | (gray << 16) | (gray << 8) | gray;
      bitmapNew.setPixel(i, j, newColor);
    }
  }
  return bitmapNew;
}

public static void saveBitmapToPath(Bitmap bitmap, String imagePath) {
  FileOutputStream out = null;
  File file = new File(imagePath);
  if (file.getParentFile().exists() == false) {
    file.getParentFile().mkdirs();
  }
  try {
    out = new FileOutputStream(imagePath);
    bitmap.compress(Bitmap.CompressFormat.PNG, 100, out);
    out.flush();
```

```
    } catch (Exception e) {
      e.printStackTrace();
    } finally {
      try {
        if (out != null) out.close();
      } catch (IOException e) {
        e.printStackTrace();
      }
    }
  }

  public static Bitmap rotateBitmap(Bitmap source, float angle) {
    Matrix matrix = new Matrix();
    matrix.postRotate(angle);
    return Bitmap.createBitmap(source, 0, 0, source.getWidth(), source.getHeight(), matrix, true);
  }

  public static Uri getResourceUri(int resId) {
    return Uri.parse("android.resource://com.lzw.flower/" + resId);
  }

  public static Bitmap getBitmapByUri(Context cxt, Uri uri) throws IOException {
    return MediaStore.Images.Media.getBitmap(cxt.getContentResolver(), uri);
  }

  public static int calInSampleSize(BitmapFactory.Options options, int reqWidth) {
    int w = options.outWidth;
    int h = options.outHeight;
    int inSampleSize = 1;
    if (w > reqWidth && reqWidth > 0) {
      inSampleSize = Math.round(w / reqWidth);
    }
    return inSampleSize;
  }

  public static Bitmap decodeSampledBitmapFromPath(String path, int reqWidth) {
    BitmapFactory.Options options = new BitmapFactory.Options();
    options.inJustDecodeBounds = true;
    BitmapFactory.decodeFile(path, options);
    int inSampleSize = calInSampleSize(options, reqWidth);
```

```
    options.inJustDecodeBounds = false;
    options.inSampleSize = inSampleSize;
    return BitmapFactory.decodeFile(path, options);
  }

  public static Bitmap decodeFileByHeight(String path, int reqH) {
    BitmapFactory.Options opt = new BitmapFactory.Options();
    opt.inJustDecodeBounds = true;
    BitmapFactory.decodeFile(path, opt);
    int scale = calInSampleSizeByHeight(opt, reqH);
    opt.inSampleSize = scale;
    opt.inJustDecodeBounds = false;
    Bitmap bm = BitmapFactory.decodeFile(path, opt);
    return bm;
  }

  public static int calInSampleSizeByHeight(BitmapFactory.Options options, int reqHeight) {
    int h = options.outHeight;
    int inSampleSize = 1;
    if (h > reqHeight) {
      inSampleSize = Math.round(h * 1.0f / reqHeight);
    }
    return inSampleSize;
  }
}
```

Was ist drin?

- **Grayscale-Umwandlung**:
  - `convertGreyImg`: Verwendet ein Pixel-Array, um das Bitmap in Graustufen zu verarbeiten.
  - `toGreyImg`: Bearbeitet Pixel für Pixel eine veränderbare Kopie und bietet eine alternative Methode. Beide verwenden die Luminanzformel (`0.3R + 0.59G + 0.11B`) für natürliche Graustufen.
- **Dateioperationen**:
  - `saveBitmapToPath`: Speichert ein Bitmap als PNG und erstellt bei Bedarf Verzeichnisse.
- **Transformationen**:
  - `rotateBitmap`: Dreht ein Bild mit einer `Matrix` –einfach, aber effektiv.
- **Laden und Abtasten**:
  - `getBitmapByUri` und `getResourceUri`: Laden Sie Bilder aus URIs oder Ressourcen.

- `decodeSampledBitmapFromPath` und `decodeFileByHeight`: Skalieren Sie große Bilder effizient nach Breite oder Höhe, um Speicherprobleme zu vermeiden.

Crop: Präzises Zuschneiden mit Androids eingebauten Tools Die `Crop`-Klasse nutzt die eingebaute Zuschneideabsicht von Android. Hier ist der Code:

```
package com.lzw.flower.utils;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.provider.MediaStore;
import com.lzw.flower.base.App;

import java.io.File;

public class Crop {
  public static void startPhotoCrop(Activity cxt, Uri uri, String outputPath, int resultCode) {
    Intent intent = new Intent("com.android.camera.action.CROP");
    intent.setDataAndType(uri, "image/*");
    int w = App.drawWidth;
    int h = App.drawHeight;
    int factor = gcd(w, h);
    int w1 = w / factor;
    int h1 = h / factor;
    intent.putExtra("crop", "true")
        .putExtra("aspectX", w1)
        .putExtra("aspectY", h1)
        .putExtra("scale", true)
        .putExtra("outputX", w)
        .putExtra("outputY", h)
        .putExtra("outputFormat", Bitmap.CompressFormat.PNG.toString());
    intent.putExtra("noFaceDetection", true);
    intent.putExtra("return-data", false);
    Uri uri1 = Uri.fromFile(new File(outputPath));
    intent.putExtra(MediaStore.EXTRA_OUTPUT, uri1);
    cxt.startActivityForResult(intent, resultCode);
```

```
  }

  static int gcd(int a, int b) {
    if (b == 0) {
      return a;
    } else {
      return gcd(b, a % b);
    }
  }
}
```

Was passiert hier?

- `startPhotoCrop`: Startet die Systemzuschneideaktivität mit einem angegebenen `Uri`, Seitenverhältnis (vereinfacht mit GCD) und Ausgabepfad. Es wird angenommen, dass `App.drawWidth` und `App.drawHeight` anderswo definiert sind (z.B. in einer Basis-`App`-Klasse).
- `gcd`: Eine rekursive Methode zur Berechnung des größten gemeinsamen Teilers, um sicherzustellen, dass das Seitenverhältnis in seiner einfachsten Form ist.

Alles zusammen: Verwendungsbeispiele Hier ist, wie Sie diese Hilfsprogramme in einer Android-App verwenden könnten:

```
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import com.lzw.flower.utils.BitmapUtils;
import com.lzw.flower.utils.Crop;

public class MainActivity extends AppCompatActivity {
    private static final int REQUEST_CROP = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Konvertieren Sie ein Bild in Graustufen und speichern Sie es
```

```
        Bitmap original = BitmapFactory.decodeResource(getResources(), R.drawable.sample);
        Bitmap grey = BitmapUtils.convertGreyImg(original);
        BitmapUtils.saveBitmapToPath(grey, "/sdcard/DCIM/grey_image.png");

        // Laden und Skalieren eines Bildes effizient
        Bitmap scaled = BitmapUtils.decodeSampledBitmapFromPath("/sdcard/DCIM/photo.jpg", 200);

        // Starten Sie das Zuschneiden
        Uri imageUri = Uri.fromFile(new File("/sdcard/DCIM/photo.jpg"));
        Crop.startPhotoCrop(this, imageUri, "/sdcard/DCIM/cropped.png", REQUEST_CROP);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST_CROP && resultCode == RESULT_OK) {
            // Das zugeschnittene Bild wird unter "/sdcard/DCIM/cropped.png" gespeichert
        }
    }
}
```

Hinweise: - Stellen Sie sicher, dass Sie die entsprechenden Speicherberechtigungen in Ihrer `AndroidManifest.xml` und Laufzeitprüfungen für Dateioperationen haben. - Die `App`-Klasse (in `Crop` verlinkt) sollte `drawWidth` und `drawHeight` definieren.

Warum diese Hilfsprogramme rocken

1. **Effizienz**: Abtastmethoden verhindern `OutOfMemoryError` bei der Arbeit mit großen Bildern.
2. **Flexibilität**: Graustufen, Drehung und Zuschneiden decken eine breite Palette von Anwendungsfällen ab.
3. **Einfachheit**: Statische Methoden machen die Integration zum Kinderspiel –keine Instanziierung erforderlich.

Schlussgedanken Die `BitmapUtils` und `Crop`-Klassen sind ein fantastischer Ausgangspunkt für jede Android-App, die Bildbearbeitung benötigt. Ob Sie einen Fotoeditor, optimierte Galerien oder benutzerdefiniertes Zuschneiden erstellen, dieser Code hat Sie abgedeckt. Probieren Sie es aus, passen Sie es Ihren Bedürfnissen an und lassen Sie mich wissen, wie es für Sie funktioniert!

Welche Bildverarbeitungsherausforderungen haben Sie in Android erlebt? Teilen Sie Ihre Gedanken unten!