Procesamiento de Imágenes en Android Hecho Fácil

Manejar imágenes en Android puede ser complicado: los bitmaps consumen memoria, las rotaciones afectan la orientación y el recorte requiere precisión. Es ahí donde entran en juego las clases de utilidad como `BitmapUtils` y `Crop`. En esta entrada, te guiaré a través de un par de clases poderosas del paquete `com.lzw.flower.utils`. Exploraremos su código, desglosaremos qué hace cada método y mostraremos cómo usarlas en tus proyectos. ¡Vamos a ello!

BitmapUtils: Tu Kit de Herramientas de Manipulación de Imágenes La clase `BitmapUtils` es una colección de métodos estáticos para manipular objetos `Bitmap`. Aquí tienes el código completo, seguido de un desglose:

```
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;
  }
}
```

¿Qué Hay Dentro?

- **Conversión a Escala de Grises**:
  - `convertGreyImg`: Usa un array de píxeles para procesar por lotes el bitmap en escala de grises.
  - `toGreyImg`: Trabaja píxel por píxel en una copia mutable, ofreciendo un enfoque alternativo. Ambos usan la fórmula de luminosidad (`0.3R + 0.59G + 0.11B`) para una escala de grises natural.
- **Operaciones de Archivo**:
  - `saveBitmapToPath`: Guarda un bitmap como PNG, creando directorios según sea necesario.
- **Transformaciones**:
  - `rotateBitmap`: Rota una imagen usando una `Matrix`—simple pero efectiva.
- **Carga y Muestreo**:
  - `getBitmapByUri` y `getResourceUri`: Cargan imágenes desde URIs o recursos.

- `decodeSampledBitmapFromPath` y `decodeFileByHeight`: Escalan eficientemente imágenes grandes por ancho o altura, evitando problemas de memoria.

Crop: Recorte Preciso con las Herramientas Nativas de Android La clase `Crop` aprovecha la intención de recorte integrada de Android. Aquí tienes el código:

```
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);
    }
  }
}
```

¿Qué Está Pasando Aquí?

- `startPhotoCrop`: Inicia la actividad de recorte del sistema con un `Uri` especificado, relación de aspecto (simplificada usando GCD) y ruta de salida. Asume que `App.drawWidth` y `App.drawHeight` están definidos en otro lugar (por ejemplo, en una clase base `App`).
- `gcd`: Un método recursivo para calcular el máximo común divisor, asegurando que la relación de aspecto esté en su forma más simple.

Juntándolo Todo: Ejemplos de Uso Aquí tienes cómo podrías usar estas utilidades en una aplicación Android:

```
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);

        // Convertir una imagen a escala de grises y guardarla
```

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

        // Cargar y escalar una imagen de manera eficiente
        Bitmap scaled = BitmapUtils.decodeSampledBitmapFromPath("/sdcard/DCIM/photo.jpg", 200);

        // Iniciar el recorte
        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) {
            // La imagen recortada se guarda en "/sdcard/DCIM/cropped.png"
        }
    }
}
```

Notas: - Asegúrate de tener los permisos de almacenamiento adecuados en tu `AndroidManifest.xml` y las comprobaciones en tiempo de ejecución para las operaciones de archivos. - La clase `App` (referenciada en `Crop`) debe definir `drawWidth` y `drawHeight`.

¿Por Qué Estas Utilidades Son Geniales?

1. **Eficiencia**: Los métodos de muestreo previenen `OutOfMemoryError` al tratar con imágenes grandes.
2. **Flexibilidad**: La conversión a escala de grises, rotación y recorte cubren una amplia gama de casos de uso.
3. **Simplicidad**: Los métodos estáticos hacen que la integración sea sencilla—no se requiere instanciación.

Pensamientos Finales Las clases `BitmapUtils` y `Crop` son un excelente punto de partida para cualquier aplicación Android que necesite manipulación de imágenes. Ya estés construyendo un editor de fotos, optimizando miniaturas de galería o añadiendo recorte dirigido por el usuario, este código te tiene cubierto. ¡Pruébalo, ajústalo a tus necesidades y házmelo saber!