Android 圖像處理變簡單

在 Android 中處理圖像可能會很棘手——位圖會佔用大量記憶體，旋轉會影響方向，裁剪需要精確度。這就是 `BitmapUtils` 和 `Crop` 這類工具類的用武之地。在這篇文章中，我將帶你了解來自 `com.lzw.flower.utils` 包的這對強大的類。我們將探索它們的代碼，分解每個方法的功能，並展示如何在你的項目中使用它們。讓我們開始吧！

BitmapUtils：您的圖像操作工具包 `BitmapUtils` 類是一組用於操作 `Bitmap` 對象的靜態方法。以下是完整的代碼，後面是詳細說明：

```
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;
  }
}
```

內容簡介

- **灰度轉換**：
  - `convertGreyImg`：使用像素陣列批量處理位圖轉換為灰度。
  - `toGreyImg`：逐像素處理可變副本，提供另一種方法。兩者都使用亮度公式（`0.3R + 0.59G + 0.11B`）來生成自然的灰度。
- **文件操作**：
  - `saveBitmapToPath`：將位圖保存為 PNG，必要時創建目錄。
- **轉換**：
  - `rotateBitmap`：使用 `Matrix` 旋轉圖像——簡單有效。
- **加載和取樣**：
  - `getBitmapByUri` 和 `getResourceUri`：從 URI 或資源加載圖像。
  - `decodeSampledBitmapFromPath` 和 `decodeFileByHeight`：根據寬度或高度高效縮放大圖像，避免記憶體問題。

Crop：使用 Android 原生工具進行精確裁剪 `Crop` 類利用 Android 的內建裁剪意圖。以下是代碼：

```
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);
    }
  }
}
```

這裡發生了什麼？

- `startPhotoCrop`：啟動系統裁剪活動，指定 `Uri`、簡化後的長寬比（使用 GCD）和輸出路徑。假設 `App.drawWidth` 和 `App.drawHeight` 在其他地方定義（例如在基礎 `App` 類中）。
- `gcd`：一個遞歸方法來計算最大公因數，確保長寬比簡化。

將它們結合起來：使用示例 這是你可能在 Android 應用中使用這些工具的方式：

```
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);

        // 將圖像轉換為灰度並保存它
        Bitmap original = BitmapFactory.decodeResource(getResources(), R.drawable.sample);
        Bitmap grey = BitmapUtils.convertGreyImg(original);
        BitmapUtils.saveBitmapToPath(grey, "/sdcard/DCIM/grey_image.png");

        // 高效加載和縮放圖像
        Bitmap scaled = BitmapUtils.decodeSampledBitmapFromPath("/sdcard/DCIM/photo.jpg", 200);

        // 啟動裁剪
        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) {
            // 裁剪後的圖像保存在 "/sdcard/DCIM/cropped.png"
        }
    }
}
```

備註：- 確保在 `AndroidManifest.xml` 中有適當的存儲權限，並在運行時檢查文件操作。- `App` 類（在 `Crop` 中引用）應該定義 `drawWidth` 和 `drawHeight`。

為什麼這些工具很棒

1. **高效**：取樣方法可以防止處理大圖像時出現 `OutOfMemoryError`。
2. **靈活性**：灰度、旋轉和裁剪涵蓋了廣泛的用例。
3. **簡單**：靜態方法使集成變得輕鬆——無需實例化。

結語 `BitmapUtils` 和 `Crop` 類是任何需要圖像操作的 Android 應用的絕佳起點。無論你是構建照片編輯器、優化相冊縮略圖，還是添加用戶驅動的裁剪，這段代碼都能滿足你的需求。試試看，根據你的需求進行調整，並告訴我它的效果如何！

你在 Android 中遇到過哪些圖像處理挑戰？在下面分享你的想法！