# [Android] 通过Menu实现图片怀旧、浮雕、模糊、光照和素描效果

原创 Eastmount 最后发布于2014-11-02 19:55:48 阅读数 5373 ☆ 收藏 展开

**由于随手拍项目想做成类似于美图秀秀那种底部有一排Menu实现不同效果的功能,这里先简单介绍如何通过Menu实现打开相册中的图片、怀旧效果、浮雕效果、光照效果和素描效果.后面可能会讲述如何通过PopupWindow实现自定义的Menu效果.**

**希望文章对大家有所帮助,如果有错误或不足之处请海涵~**

## 一. Menu效果展示

**Android手机上有个Menu按键,点击他会弹出一个菜单,通常在屏幕底部或右上角,在选项菜单OptionsMenu中最多显示2排每排3个菜单项,可以包含自定义的图片和文字.如果Menu菜单项多于6项时,第6项(Expanded Menus,扩展菜单)会变成More,点击它会显示后面所隐藏的所有选项.**

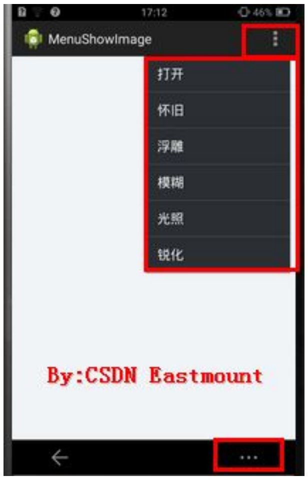
**下面讲述如何在Android 4.0项目中实现简单的Menu功能.添加如下代码：**

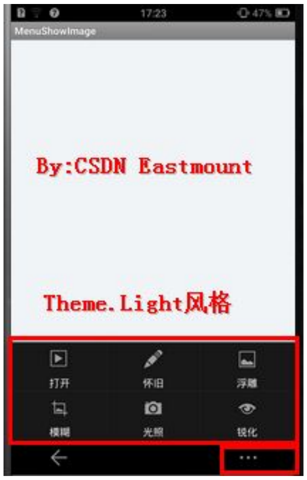
```
@Override
public boolean onCreateOptionsMenu(Menu menu) { //创建Menu
        //自定义menu  添加图标(使用自带图标)
        menu.add(Menu.NONE, Menu.FIRST + 1 , 1, "打开").
                setIcon(android.R.drawable.ic_menu_slideshow);
        menu.add(Menu.NONE, Menu.FIRST + 2 , 2, "怀旧").
                setIcon(android.R.drawable.ic_menu_edit);
        menu.add(Menu.NONE, Menu.FIRST + 3 , 3, "浮雕").
                setIcon(android.R.drawable.ic_menu_gallery);
        menu.add(Menu.NONE, Menu.FIRST + 4 , 4, "模糊").
                setIcon(android.R.drawable.ic_menu_crop);
        menu.add(Menu.NONE, Menu.FIRST + 5 , 5, "光照").
                setIcon(android.R.drawable.ic_menu_camera);
        menu.add(Menu.NONE, Menu.FIRST + 6  , 6, "锐化").
                setIcon(android.R.drawable.ic_menu_view);
    return true;
}
```

**由于Android 4.0系统缺省UI风格有所变化,所以需要设置Activity的theme为Theme.Light.同时也可以在res/menu/main.xml设置菜单项.参考"恺风"博主关于Menu的介绍,非常不错.http://blog.csdn.net/flowingflying/article/details/11967301**

```
<activity
    android:name="com.example.menushowimage.MainActivity"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Light" >
```

**下图是设置前面的显示Menu不同效果,同时我调用的图标都是Android自带的图片,用户也可以自定义.(android默认图标列表)**





**同时设置XML格式显示图片:**

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.touchimagetest.MainActivity"
    tools:ignore="MergeRootFrame" >
    <!-- 顶部添加文字 -->
    <RelativeLayout
        android:id="@+id/Layout_top"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="25dp"
        android:layout_alignParentTop="true"
        android:gravity="center">
         <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:text="请点击menu处理图片" />
        </RelativeLayout>
    <!-- 底部显示图片 -->
    <RelativeLayout
        android:id="@+id/Layout_bottom"
        android:orientation="horizontal"
        android:layout_below="@id/Layout_top"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#EFDFDF"
        android:gravity="center">
        <ImageView
                android:id="@+id/imageView1"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_gravity="center_horizontal" />
    </RelativeLayout>
</RelativeLayout>
```

## 二. Menu实现打开图片

**然后通过onOptionsItemSelected(MenuItem item)实现选择图片,通过调用自定义函数实现各种功能.**

```
@Override
          public boolean onOptionsItemSelected(MenuItem item) { //选择Menu
        //选择id 对应Menu.add的参数Menu.FIRST+i
    int id = item.getItemId();
    switch(id) {
    case Menu.FIRST+1:
        Toast.makeText(this, "打开图片", Toast.LENGTH_SHORT).show();
        OpenImage();
        break;
    case Menu.FIRST+2:
        Toast.makeText(this, "图片怀旧效果", Toast.LENGTH_SHORT).show();
        OldRemeberImage();
        break;
    case Menu.FIRST+3:
        Toast.makeText(this, "图片浮雕效果", Toast.LENGTH_SHORT).show();
        ReliefImage();
        break;
    case Menu.FIRST+4:
        Toast.makeText(this, "图片模糊效果", Toast.LENGTH_SHORT).show();
        FuzzyImage();
        break;
    case Menu.FIRST+5:
        Toast.makeText(this, "图片光照效果", Toast.LENGTH_SHORT).show();
        SunshineImage();
        break;
    case Menu.FIRST+6:
        Toast.makeText(this, "图片锐化效果", Toast.LENGTH_SHORT).show();
        SharpenImage();
        break;
    }

    return super.onOptionsItemSelected(item);
}
```

**其中打开图片函数实现方法如下,而上面的很多自定义函数都将在第三部分介绍,你此处可以注释掉只验证"打开图片".首先添加自定义变量和获取ImageView布局.**

```
//自定义变量
private ImageView imageShow;           //显示图片
private Bitmap bmp;                    //原始图片
private final int IMAGE_OPEN = 0;      //打开图片
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    imageShow = (ImageView) findViewById(R.id.imageView1);
    if (savedInstanceState == null) {
        getFragmentManager().beginTransaction()
                .add(R.id.container, new PlaceholderFragment())
                .commit();
    }
}
```

**然后通过自定义函数OpenImage打开函数,与前面文章介绍的方法一样.**

```
//自定义函数 打开图片
public void OpenImage()
{
        Intent intent = new Intent(Intent.ACTION_PICK,
            android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    startActivityForResult(intent, IMAGE_OPEN); }
//显示打开图片
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(resultCode==RESULT_OK && requestCode==IMAGE_OPEN) {
       Uri imageFileUri = data.getData();
       DisplayMetrics dm = new DisplayMetrics();
       getWindowManager().getDefaultDisplay().getMetrics(dm);
       int width = dm.widthPixels;    //手机屏幕水平分辨率
       int height = dm.heightPixels;  //手机屏幕垂直分辨率
       try {
           //载入图片尺寸大小没载入图片本身 true
           BitmapFactory.Options bmpFactoryOptions = new BitmapFactory.Options();
           bmpFactoryOptions.inJustDecodeBounds = true;
           bmp = BitmapFactory.decodeStream(getContentResolver().openInputStream(imageFileUri), null,
bmpFactoryOptions);
           int heightRatio = (int)Math.ceil(bmpFactoryOptions.outHeight/(float)height);
           int widthRatio = (int)Math.ceil(bmpFactoryOptions.outWidth/(float)width);
           //inSampleSize表示图片占原图比例 1表示原图              if(heightRatio>1&&widthRatio>1) {
               if(heightRatio>widthRatio) {
                   bmpFactoryOptions.inSampleSize = heightRatio;
               }
               else {
                   bmpFactoryOptions.inSampleSize = widthRatio;
               }
           }
           //图像真正解码 false
           bmpFactoryOptions.inJustDecodeBounds = false;
           bmp = BitmapFactory.decodeStream(getContentResolver().openInputStream(imageFileUri), null,
bmpFactoryOptions);
           imageShow.setImageBitmap(bmp);          }
       catch(FileNotFoundException e) {
           e.printStackTrace();
       }
   }  //end if
}
```

**下面讲讲使用Options Menu的函数:**

**onCreateOptionsMenu(Menu menu)创建options menu,这个函数只会在menu第一次显示时调用.**

**onOptionsItemSelected(MenuItem item)处理选中的菜单项.**

**在通过menu.add函数实现添加菜单项,如menu.add(Menu.NONE,Menu.FIRST+1,1,"打开"),第一个参数表示组别;第二个参数menu标志编号与onOptionsItemSelected函数中值对应;第三个参数是在菜单中出现的顺序,顺序由小到大,由左至右;第四个参数是显示的文字,同时setIcon可以设置图标.**

## 三. 图像各种效果实现

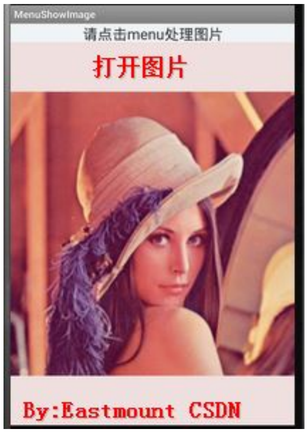
**最后讲讲各个效果实现过程,通过不同自定义函数实现.其中各个效果主要参照《Android图像处理总结》那篇文章和eoeAndroid社区亚瑟的文章.**

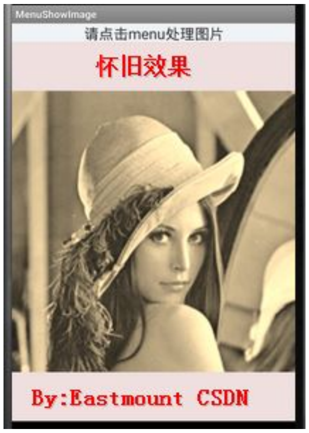
**书籍下载地址:**

### 1.图片怀旧效果

```
//图片怀旧处理
private void OldRemeberImage()
{
        /*
         * 怀旧处理算法即设置新的RGB
         * R=0.393r+0.769g+0.189b           * G=0.349r+0.686g+0.168b
         * B=0.272r+0.534g+0.131b
         */
        int width = bmp.getWidth();
        int height = bmp.getHeight();
        Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.RGB_565);
        int pixColor = 0;
        int pixR = 0;
        int pixG = 0;
        int pixB = 0;
        int newR = 0;
        int newG = 0;
        int newB = 0;
        int[] pixels = new int[width * height];
        bmp.getPixels(pixels, 0, width, 0, 0, width, height);
        for (int i = 0; i < height; i++)
        {
                for (int k = 0; k < width; k++)
                {
                        pixColor = pixels[width * i + k];
                        pixR = Color.red(pixColor);
                        pixG = Color.green(pixColor);
                        pixB = Color.blue(pixColor);
                        newR = (int) (0.393 * pixR + 0.769 * pixG + 0.189 * pixB);
                        newG = (int) (0.349 * pixR + 0.686 * pixG + 0.168 * pixB);
                        newB = (int) (0.272 * pixR + 0.534 * pixG + 0.131 * pixB);
                        int newColor = Color.argb(255, newR > 255 ? 255 : newR, newG > 255 ? 255 : newG, newB
> 255 ? 255 : newB);
                        pixels[width * i + k] = newColor;                  }
        }
        bitmap.setPixels(pixels, 0, width, 0, 0, width, height);
        imageShow.setImageBitmap(bitmap);
}
```

**显示效果如下图所示：**

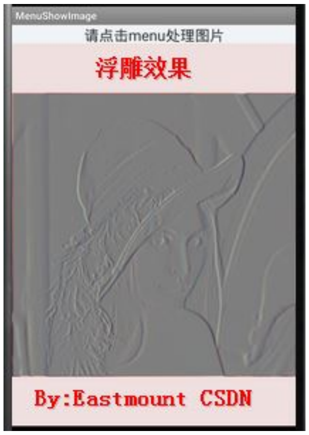




## 2.图片浮雕效果

```
//图片浮雕处理
//底片效果也非常简单:将当前像素点的RGB值分别与255之差后的值作为当前点的RGB
//灰度图像:通常使用的方法是gray=0.3*pixR+0.59*pixG+0.11*pixB
private void ReliefImage()
{
        /*
         * 算法原理：(前一个像素点RGB-当前像素点RGB+127)作为当前像素点RGB值
         * 在ABC中计算B点浮雕效果(RGB值在0~255)
         * B.r = C.r - B.r + 127
         * B.g = C.g - B.g + 127                          * B.b = C.b - B.b + 127
         */
        int width = bmp.getWidth();
        int height = bmp.getHeight();
        Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.RGB_565);
        int pixColor = 0;
        int pixR = 0;
        int pixG = 0;
        int pixB = 0;
        int newR = 0;
        int newG = 0;
        int newB = 0;
        int[] pixels = new int[width * height];
        bmp.getPixels(pixels, 0, width, 0, 0, width, height);
        for (int i = 1; i < height-1; i++)
        {
                for (int k = 1; k < width-1; k++)
                {
                        //获取前一个像素颜色
                        pixColor = pixels[width * i + k];
                        pixR = Color.red(pixColor);
                        pixG = Color.green(pixColor);
                        pixB = Color.blue(pixColor);
                        //获取当前像素
                        pixColor = pixels[(width * i + k) + 1];
                        newR = Color.red(pixColor) - pixR +127;
                        newG = Color.green(pixColor) - pixG +127;
                        newB = Color.blue(pixColor) - pixB +127;
                        newR = Math.min(255, Math.max(0, newR));
                        newG = Math.min(255, Math.max(0, newG));
                        newB = Math.min(255, Math.max(0, newB));
                        pixels[width * i + k] = Color.argb(255, newR, newG, newB);
                }
        }
        bitmap.setPixels(pixels, 0, width, 0, 0, width, height);
        imageShow.setImageBitmap(bitmap);
}
```

**显示效果如下图所示：**



## 3.图像模糊效果

```
//图像模糊处理
private void FuzzyImage()
{
        /*
```

```
         * 算法原理:           * 简单算法将像素周围八个点包括自身共九个点RGB值分别相加后平均,当前像素点的RGB值
         * 复杂算法采用高斯模糊
         * 高斯矩阵 int[] gauss = new int[] { 1, 2, 1, 2, 4, 2, 1, 2, 1 };
         * 将九个点的RGB值分别与高斯矩阵中的对应项相乘的和,再除以一个相应的值作为当前像素点的RGB
         */
    int[] gauss = new int[] { 1, 2, 1, 2, 4, 2, 1, 2, 1 };  // 高斯矩阵
    int delta = 16; // 除以值 值越小图片会越亮,越大则越暗
        int width = bmp.getWidth();
        int height = bmp.getHeight();
        Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.RGB_565);
        int pixColor = 0;
        int pixR = 0;
    int pixG = 0;
    int pixB = 0;
        int newR, newG, newB;
        int pos = 0;    //位置
        int[] pixels = new int[width * height];
    bmp.getPixels(pixels, 0, width, 0, 0, width, height);
        //循环赋值
        for (int i = 1; i < height-1; i++)
        {
                for (int k = 1; k < width-1; k++)
                {
                        pos = 0;
                        newR = 0;
            newG = 0;
            newB = 0;
                        for (int m = -1; m <= 1; m++)  //宽不变
            {
                for (int n = -1; n <= 1; n++) //高先变
                {
                    pixColor = pixels[(i + m) * width + k + n];
                    pixR = Color.red(pixColor);
                    pixG = Color.green(pixColor);
                    pixB = Color.blue(pixColor);
                    //3*3像素相加
                    newR = newR + (int) (pixR * gauss[pos]);
                    newG = newG + (int) (pixG * gauss[pos]);
                    newB = newB + (int) (pixB * gauss[pos]);
                    pos++;
                }
            }
                        newR /= delta;
            newG /= delta;
            newB /= delta;
            newR = Math.min(255, Math.max(0, newR));
            newG = Math.min(255, Math.max(0, newG));
            newB = Math.min(255, Math.max(0, newB));
            pixels[i * width + k] = Color.argb(255, newR, newG, newB);
                }
        }
        bitmap.setPixels(pixels, 0, width, 0, 0, width, height);
        imageShow.setImageBitmap(bitmap);
}
```

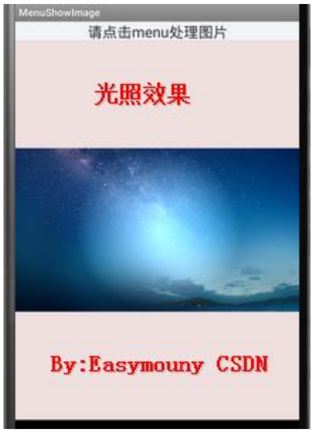
**该图显示效果不是很理想,对高斯模糊理解还不够,建议大家看我收藏合集里面讲述模糊的超链接.**

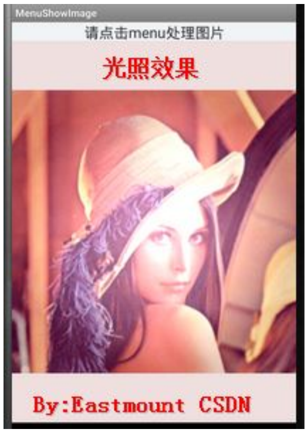
## 4.图像光照效果

```
//图片光照效果
private void SunshineImage()
```

```
{
          /*
         * 算法原理：(前一个像素点RGB-当前像素点RGB+127)作为当前像素点RGB值
         * 在ABC中计算B点浮雕效果(RGB值在0~255)
         * B.r = C.r - B.r + 127
         * B.g = C.g - B.g + 127
         * B.b = C.b - B.b + 127
         * 光照中心取长宽较小值为半径,也可以自定义从左上角射过来
         */
        int width = bmp.getWidth();
        int height = bmp.getHeight();
        Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.RGB_565);
        int pixColor = 0;
        int pixR = 0;
        int pixG = 0;
        int pixB = 0;
        int newR = 0;
        int newG = 0;
        int newB = 0;
        //围绕圆形光照
        int centerX = width / 2;
        int centerY = height / 2;
        int radius = Math.min(centerX, centerY);
        float strength = 150F;  //光照强度100-150
        int[] pixels = new int[width * height];
        bmp.getPixels(pixels, 0, width, 0, 0, width, height);
        for (int i = 1; i < height-1; i++)
        {
                for (int k = 1; k < width-1; k++)
                {
                        //获取前一个像素颜色
                        pixColor = pixels[width * i + k];
                        pixR = Color.red(pixColor);
                        pixG = Color.green(pixColor);
                        pixB = Color.blue(pixColor);
                        newR = pixR;
                        newG = pixG;
                        newB = pixB;
                        //计算当前点到光照中心的距离,平面坐标系中两点之间的距离
                        int distance = (int) (Math.pow((centerY-i), 2) + Math.pow((centerX-k), 2));
                        if(distance < radius*radius)
                        {
                                //按照距离大小计算增强的光照值
                                int result = (int)(strength*( 1.0-Math.sqrt(distance) / radius ));
                                newR = pixR + result;
                                newG = newG + result;
                                newB = pixB + result;
                        }
                        newR = Math.min(255, Math.max(0, newR));
                        newG = Math.min(255, Math.max(0, newG));
                        newB = Math.min(255, Math.max(0, newB));
                        pixels[width * i + k] = Color.argb(255, newR, newG, newB);
                }
        }
        bitmap.setPixels(pixels, 0, width, 0, 0, width, height);
        imageShow.setImageBitmap(bitmap);
}
```

**显示效果如下图所示**



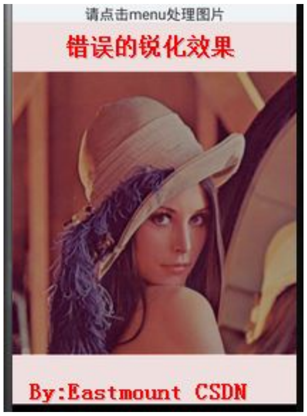


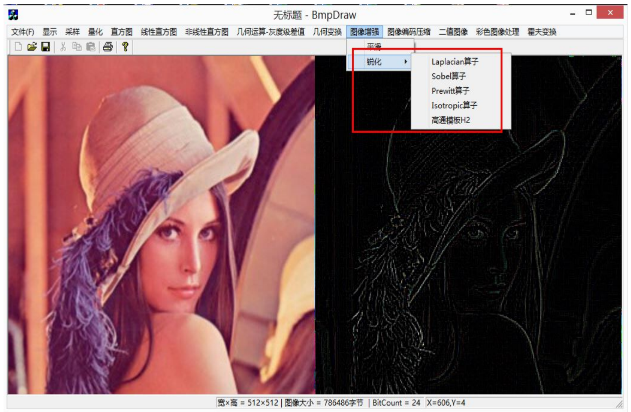
## 5.图片锐化效果

**本打算采用拉普拉斯算子或Sobel算子对图像进行锐化,在使用C++对24位bmp图像处理时能非常好的显示图像的轮廓,但是Android总是效果不是很好啊,而且有虚线!网上一些锐化效果完全没有实现显示图像轮廓,与原图区别不大,感觉是错误的方法.研究ing**

```
//图像锐化处理 拉普拉斯算子处理
private void SharpenImage()
{
        /*
         * 锐化基本思想是加强图像中景物的边缘和轮廓,使图像变得清晰
         * 而图像平滑是使图像中边界和轮廓变得模糊
         *
         * 拉普拉斯算子图像锐化
         * 获取周围9个点的矩阵乘以模板9个的矩阵 卷积
         */
        //拉普拉斯算子模板 { 0, -1, 0, -1, -5, -1, 0, -1, 0 } { -1, -1, -1, -1, 9, -1, -1, -1, -1 }
        int[] laplacian = new int[] {  -1, -1, -1, -1, 9, -1, -1, -1, -1 };
        int width = bmp.getWidth();
    int height = bmp.getHeight();
    Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.RGB_565);
    int pixR = 0;
    int pixG = 0;
    int pixB = 0;
    int pixColor = 0;
    int newR = 0;
    int newG = 0;
    int newB = 0;
    int idx = 0;
    float alpha = 0.3F;  //图片透明度
    int[] pixels = new int[width * height];
    bmp.getPixels(pixels, 0, width, 0, 0, width, height);
    //图像处理
    for (int i = 1; i < height - 1; i++)
    {
        for (int k = 1; k < width - 1; k++)
        {
                idx = 0;
                newR = 0;
            newG = 0;
            newB = 0;
            for (int n = -1; n <= 1; n++)   //取出图像3*3领域像素
            {
                for (int m = -1; m <= 1; m++)  //n行数不变 m列变换
                {
                    pixColor = pixels[(i + n) * width + k + m];  //当前点(i,k)
                    pixR = Color.red(pixColor);
                    pixG = Color.green(pixColor);
                    pixB = Color.blue(pixColor);
                    //图像像素与对应摸板相乘
                    newR = newR + (int) (pixR * laplacian[idx] * alpha);
                    newG = newG + (int) (pixG * laplacian[idx] * alpha);
                    newB = newB + (int) (pixB * laplacian[idx] * alpha);
                    idx++;
                }
            }
            newR = Math.min(255, Math.max(0, newR));
            newG = Math.min(255, Math.max(0, newG));
            newB = Math.min(255, Math.max(0, newB));
            //赋值
            pixels[i * width + k] = Color.argb(255, newR, newG, newB);
        }
    }
    bitmap.setPixels(pixels, 0, width, 0, 0, width, height);
    imageShow.setImageBitmap(bitmap);
}
```

**作图是其显示效果,而右图是我以前《数字图像处理》课用C++写的不同模版的锐化效果.**





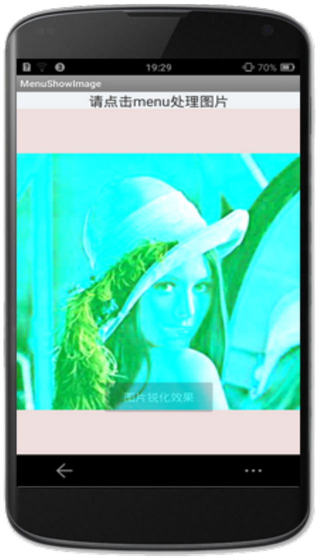
宽×高 = 512×512 | 图像大小 = 786486字节 | BitCount = 24 | X=606,Y=4

**下面再介绍些效果,下面这个效果是参考亚瑟BOY的冰冻效果.**

**源代码地址:http://www.eoeandroid.com/thread-176490-1-1.html**

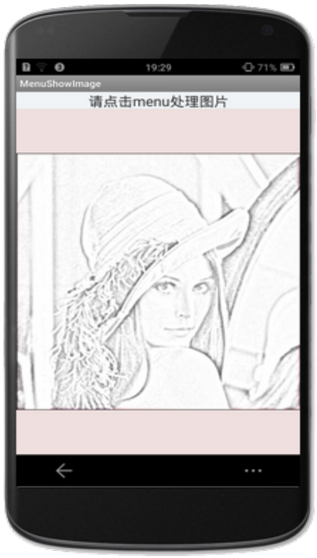
```
//图片冰冻效果
private void IceImage()
{
        int width = bmp.getWidth();
        int height = bmp.getHeight();
        Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.RGB_565);
        int pixColor = 0;
        int pixR = 0;
        int pixG = 0;
        int pixB = 0;
        int newColor = 0;
        int newR = 0;
        int newG = 0;
        int newB =0;
        int[] pixels = new int[width * height];
        bmp.getPixels(pixels, 0, width, 0, 0, width, height);
        for (int i = 0; i < height; i++)
        {
                for (int k = 0; k < width; k++)
                {
                        //获取前一个像素颜色
                        pixColor = pixels[width * i + k];
                        pixR = Color.red(pixColor);
                        pixG = Color.green(pixColor);
                        pixB = Color.blue(pixColor);
                        //红色
                        newColor = pixR - pixG - pixB;
                        newColor = newColor * 3 / 2;
                        if(newColor < 0) {
                                newColor = -newColor;
                        }
                        if(newColor >255) {
                                newColor = 255;
                        }
                        newR = newColor;
                        //绿色
                        newColor = pixG - pixB - pixR;
                        newColor = newColor * 3 / 2;
                        if(newColor < 0) {
                                newColor = -newColor;
                        }
                        if(newColor >255) {
                                newColor = 255;
                        }
                        newG = newColor;
                        //蓝色
                        newColor = pixB - pixG - pixR;
                        newColor = newColor * 3 / 2;
                        if(newColor < 0) {
                                newColor = -newColor;
                        }
                        if(newColor >255) {
                                newColor = 255;
                        }
                        newB = newColor;
                        pixels[width * i + k] = Color.argb(255, newR, newG, newB);
                }
        }
        bitmap.setPixels(pixels, 0, width, 0, 0, width, height);
        imageShow.setImageBitmap(bitmap);
}
```

**下面这个代码是CSDN的xu_fu博主的素描处理,对我软件有用.**
**源代码地址：http://blog.csdn.net/xu_fu/article/details/21485461**
**效果显示如下图所示,在Menu选择中调用函数IceImage或SuMiaoImage即可实现.**


MenuShowImage

请点击menu处理图片

图片锐化效果


MenuShowImage

请点击menu处理图片

```
//素描效果
private void SuMiaoImage()
{
        //创建新Bitmap
        int width = bmp.getWidth();
    int height = bmp.getHeight();
    int[] pixels = new int[width * height];    //存储变换图像
    int[] linpix = new int[width * height];     //存储灰度图像
    Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.RGB_565);
    bmp.getPixels(pixels, 0, width, 0, 0, width, height);
    int pixColor = 0;
        int pixR = 0;
        int pixG = 0;
        int pixB = 0;
    int newR = 0;
    int newG = 0;
    int newB = 0;
    //灰度图像
    for (int i = 1; i < width - 1; i++)
    {
        for (int j = 1; j < height - 1; j++)   //拉普拉斯算子模板 { 0, -1, 0, -1, -5, -1, 0, -1, 0
        {
                //获取前一个像素颜色
                        pixColor = pixels[width * i + j];
                        pixR = Color.red(pixColor);
                        pixG = Color.green(pixColor);
                        pixB = Color.blue(pixColor);
                //灰度图像
                int gray=(int)(0.3*pixR+0.59*pixG+0.11*pixB);
                linpix[width * i + j] = Color.argb(255, gray, gray, gray);
                //图像反向
                gray=255-gray;
                pixels[width * i + j] = Color.argb(255, gray, gray, gray);
        }
    }
    int radius = Math.min(width/2, height/2);
    int[] copixels = gaussBlur(pixels, width, height, 10, 10/3);   //高斯模糊 采用半径10
    int[] result = colorDodge(linpix, copixels);   //素描图像 颜色减淡
    bitmap.setPixels(result, 0, width, 0, 0, width, height);
    imageShow.setImageBitmap(bitmap);
}
```

```
//高斯模糊
          public static int[] gaussBlur(int[] data, int width, int height, int radius,
        float sigma) {

    float pa = (float) (1 / (Math.sqrt(2 * Math.PI) * sigma));
    float pb = -1.0f / (2 * sigma * sigma);

    // generate the Gauss Matrix
    float[] gaussMatrix = new float[radius * 2 + 1];
    float gaussSum = 0f;
    for (int i = 0, x = -radius; x <= radius; ++x, ++i) {
        float g = (float) (pa * Math.exp(pb * x * x));
        gaussMatrix[i] = g;
        gaussSum += g;
    }

    for (int i = 0, length = gaussMatrix.length; i < length; ++i) {
        gaussMatrix[i] /= gaussSum;
    }

    // x direction
    for (int y = 0; y < height; ++y) {
        for (int x = 0; x < width; ++x) {
            float r = 0, g = 0, b = 0;
            gaussSum = 0;
            for (int j = -radius; j <= radius; ++j) {
                int k = x + j;
                if (k >= 0 && k < width) {
                    int index = y * width + k;
                    int color = data[index];
                    int cr = (color & 0x00ff0000) >> 16;
                    int cg = (color & 0x0000ff00) >> 8;
                    int cb = (color & 0x000000ff);

                    r += cr * gaussMatrix[j + radius];
                    g += cg * gaussMatrix[j + radius];
                    b += cb * gaussMatrix[j + radius];

                    gaussSum += gaussMatrix[j + radius];
                }
            }

            int index = y * width + x;
            int cr = (int) (r / gaussSum);
            int cg = (int) (g / gaussSum);
            int cb = (int) (b / gaussSum);

            data[index] = cr << 16 | cg << 8 | cb | 0xff000000;
        }
    }

    // y direction
    for (int x = 0; x < width; ++x) {
        for (int y = 0; y < height; ++y) {
            float r = 0, g = 0, b = 0;
            gaussSum = 0;
            for (int j = -radius; j <= radius; ++j) {
                int k = y + j;
                if (k >= 0 && k < height) {
                    int index = k * width + x;
                    int color = data[index];
                    int cr = (color & 0x00ff0000) >> 16;
```

```
                        int cg = (color & 0x0000ff00) >> 8;
                        int cb = (color & 0x000000ff);

                        r += cr * gaussMatrix[j + radius];
                        g += cg * gaussMatrix[j + radius];
                        b += cb * gaussMatrix[j + radius];

                        gaussSum += gaussMatrix[j + radius];
                    }
                }

                int index = y * width + x;
                int cr = (int) (r / gaussSum);
                int cg = (int) (g / gaussSum);
                int cb = (int) (b / gaussSum);
                data[index] = cr << 16 | cg << 8 | cb | 0xff000000;
            }
        }

        return data;
    }

    //颜色减淡
    public static int[] colorDodge(int[] baseColor, int[] mixColor) {

        for (int i = 0, length = baseColor.length; i < length; ++i) {
            int bColor = baseColor[i];
            int br = (bColor & 0x00ff0000) >> 16;
            int bg = (bColor & 0x0000ff00) >> 8;
            int bb = (bColor & 0x000000ff);

            int mColor = mixColor[i];
            int mr = (mColor & 0x00ff0000) >> 16;
            int mg = (mColor & 0x0000ff00) >> 8;
            int mb = (mColor & 0x000000ff);

            int nr = colorDodgeFormular(br, mr);
            int ng = colorDodgeFormular(bg, mg);
            int nb = colorDodgeFormular(bb, mb);

            baseColor[i] = nr << 16 | ng << 8 | nb | 0xff000000;
        }
        return baseColor;
    }

    private static int colorDodgeFormular(int base, int mix) {

        int result = base + (base * mix) / (255 - mix);
        result = result > 255 ? 255 : result;
        return result;

    }
```

**最后希望文章对大家有所帮助,感谢上面提到的作者,同时可能还有些如LOMO等效果可参考下面的文章,它是图像处理的一个集合超链接.后面会写PopupWindows实现美图秀秀的效果和对人脸进行处理.**

**源代码下载:**

**(By:Eastmount 2014-11-2 晚8点 http://blog.csdn.net/eastmount/)**

点赞 5 ☆ 收藏 分享 ···

Eastmount 博客专家

发布了445 篇原创文章 · 获赞 5981 · 访问量 487万+

他的留言板 关注