




< 2016年6月 >						
日	一	二	三	四	五	六
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

公告


昵称：Xiangism
园龄：7年3个月
粉丝：57
关注：11
[+加关注](#)

搜索

找找看
 谷歌搜索

- 随笔分类(54)
- .NET(3)
 - C/C++(10)
 - IT杂项(5)
 - Linux(3)
 - Maths(2)
 - MFC(4)
 - MyApp(9)
 - Python(1)
 - 纯粹的算法(6)
 - 图形/图像(7)
 - 效率(4)

- 随笔档案(52)
- 2016年3月 (2)
 - 2016年2月 (1)
 - 2015年10月 (1)
 - 2015年7月 (5)
 - 2015年6月 (3)
 - 2015年5月 (1)
 - 2014年7月 (1)
 - 2012年11月 (9)
 - 2012年10月 (8)
 - 2012年8月 (1)
 - 2012年7月 (2)
 - 2012年6月 (2)
 - 2011年8月 (1)
 - 2011年7月 (3)
 - 2011年3月 (1)
 - 2010年5月 (1)
 - 2010年4月 (1)
 - 2010年3月 (1)
 - 2009年12月 (5)
 - 2009年9月 (3)

最新评论

1. Re:MFC,C++ 截屏

常见图片格式详解

Posted on 2016-03-31 09:40 Xiangism 阅读(755) 评论(3) 编辑 收藏

做了几年有关图形、图像的工作，对图片格式算是小有经验，在此写成一篇文章总结下。虽然一开始并不想讲很理论的东西，但写完理论，细想一下关于图片格式的知识本身就是理论的东西，囧~~那就力求用最简单的方式将这些“理论”讲清楚吧。

常见的图片格式有bmp, jpg(jpeg), png, gif, webp等。

图像基本数据结构

要讲图片格式还先得从图像的基本数据结构说起。在计算机中, 图像是由一个个像素点组成, 像素点就是颜色点, 而颜色最简单的或RGBA表示, 如图所示

RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB

(图1)

RGBA	RGBA	RGBA	RGBA	RGBA
RGBA	RGBA	RGBA	RGBA	RGBA
RGBA	RGBA	RGBA	RGBA	RGBA
RGBA	RGBA	RGBA	RGBA	RGBA
RGBA	RGBA	RGBA	RGBA	RGBA

(图2)

如果有A通道就表明这个图像可以有透明效果。

R,G,B每个分量一般是用一个字节(8位)来表示，所以图(1)中每个像素大小就是3*8=24位图, 而图(2)中每个像素大小是4*8=32位。

这里有三点需要说明:

一、图像y方向正立或倒立

图像是二维数据，数据在内存中只能一维存储，二维转一维有不同的对应方式。比较常见的只有两种方式: 按像素“行排列”从上到下。

1	2	3
4	5	6
7	8	9

@LubinLew 你再看下我的这篇 好好学下图像吧，不要想着抄代码，关键是思想...

--Xiangism

2. Re:具有编译功能支持无限大数计算器的实现

对，我改了一下，没有改出来，

--大象_无形

3. Re:具有编译功能支持无限大数计算器的实现

@大象213谢谢反馈。直接死循环了，应该是一直卡在小数点的精确度上了...

--Xiangism

4. Re:具有编译功能支持无限大数计算器的实现

好像计算pow(0.5,1.55)有问题，请处理一下。

--大象213

5. Re:介绍四款windows下的神器

@Xiangism可以在工具-选项中设置是否自动包含移动设备...

--maanshancss

阅读排行榜

1. MFC,C++ 截屏(5939)
2. 介绍四款windows下的神器(3534)
3. 当Visual Studio中的注释也会生产代码时.....(3241)
4. Android App请求获取Root权限(2815)
5. 自动扫雷——前言(2735)

推荐排行榜

1. 介绍四款windows下的神器(13)
2. 打磨程序员的专属利器——文本(9)
3. 当Visual Studio中的注释也会生产代码时.....(8)
4. 支持无限精度无限大数的类BigInteger实现(6)
5. 常见图片格式详解(5)

如图所示的图像有9个像素点，如果从上往下排列成一维数据是(123456789)，如果是从下往上排列则为(789456123)。只所以会有这种区别是因为，前一种是以计算机图形学的屏幕坐标系为参考(右上为原点,y轴向下)，而另一种是以标准的数学坐标系为参考(左下为原点,y轴向上)。这两个坐标系只是y值不一样，互相转换的公式为：

$$y_2 = \text{height} - 1 - y_1$$

y_1, y_2 分别为像素在两个坐标系中的y坐标，height为图像的高度。

不过好像只有bmp图片格式以及windows下的GDI，GDI+是从下往上排列，其它比如DirectX,OpenGL,Cocoa(NSImage, Ullma)都是从上往下排列。

二、RGB排列顺序

不同图形库中每个像素点中RGBA的排序顺序可能不一样。上面说过像素一般会有RGB,或RGBA四个分量，那么在内存中RGB的排序，如下：

- RGB
- RBG
- GRB
- GBR
- BGR
- BRG

RGBA的排列有24种情况，这里就不全部列出来了。

不过一般只有RGB,BGR, RGBA, RGB, BGRA这几种排列据。绝大多数图形库或环境是BGR/BGRA排列，cocoa中的NSImage RGBA排列。

三、像素32位对齐

如果是RGB24位图，会存在一个32位对齐的问题——

在x86体系下，cpu一次处理32整数倍的数据会更快，图像处理中经常会按行为单位来处理像素。24位图，宽度不是4的倍数时，是32整数倍。这时可以采取在行尾添加冗余数据的方式，使其行字节数为32的倍数。

比如，如果图像宽为5像素，不做32位对齐的话，其行位数为 $24 \times 5 = 120$ ，120不是32的倍数。是32整数倍并且刚好比120大的数就要在其行尾添加1字节(8位)的冗余数据即可。(一个以空间换时间的例子)

有个公式可以轻松计算出32位对齐后每行应该占的字节数

$$\text{byteNum} = ((\text{width} \times 24 + 31) \& \sim 31) \gg 3;$$

注意结果是字节数，如果想知道位数，还得 $\times 8$

图片格式的必要性

如果将图像原始格式直接存储到文件中将会非常大，比如一个 5000×5000 24位图，所占文件大小为 $5000 \times 5000 \times 3 \text{字节} = 71.5 \text{MB}$ ，可观。

如果用zip或rar之类的通用算法来压缩像素数据，得到的压缩比例通常不会太高，因为这些压缩算法没有针对图像数据结构进行特于是就有了jpeg,png等格式，同样是图像压缩算法jpeg和png也有不同的适用场景，具体在下文再阐述。

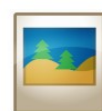
内存中的图像数据

RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB	RGB

硬盘(网络传输)中的文

encode (write)

decode (read)



bmp(未压缩)



jpeg(已压缩)



png(已压缩)

所以可以总结如下: jpeg,png文件之于图像，就相当于zip,rar格式之于普通文件(用zip,rar格式对普通文件进行压缩)。

BMP格式

bmp格式没有压缩像素格式，存储在文件中时先有文件头、再图像头、后面就都是像素数据了，上下颠倒存储。

用windows自带的mspaint工具保存bmp格式时，可以发现四种bmp可供选择：

单色：一个像素只占一位，要么是0，要么是1，所以只能存储黑白信息

16色位图：一个像素4位，有16种颜色可选

256色位图：一个像素8位，有256种颜色可选

24位位图：就是图(1)所示的位图，颜色可有 2^{24} 种可选，对于人眼来说完全足够了。

这里为了简单起见，只详细讨论最常见的24位图的bmp格式。

现在来看其文件头和图片格式头的结构:

文件头信息		
字段	大小(字节)	描述
bfType	2	一定为19778，其转化为十六进制为0x4d42，对应的字符串为BM
bfSize	4	文件大小
bfReserved1	2	一般为0
bfReserved2	2	一般为0
bfOffBits	4	从文件开始处到像素数据的偏移，也就是这两个结构体大小之和

bmp图片结构头		
字段	大小(字节)	描述
biSize	4	此结构体的大小
biWidth	4	图像的宽
biHeight	4	图像的高
biPlanes	2	图像的帧数，一般为1
biBitCount	2	一像素所占的位数，一般是24
biCompression	4	一般为0
biSizeImage	4	像素数据所占大小，即上面结构体中文件大小减去偏移(bfSize-bfOffBits)
biXPelsPerMeter	4	一般为0
biYPelsPerMeter	4	一般为0
biClrUsed	4	一般为0
biClrImportant	4	一般为0

本来在windows平台下wingdi.h文件中已经有这些结构的定义，不过为了不依赖与windows，实现为跨平台，本人将wingdi.h中结构“偷用”出来了。代码如下：

Bmp结构体

由于bmp格式比较简单，本人已实现了一份简单的c++代码，具有读取、保存bmp图片的功能，只支持24位的bmp格式。
代码在 <http://git.oschina.net/xiangism/blogData> 的“常见图片格式详解/ImageDemo/BmpDemo”文件夹中。
虽然这里只建立了vs2008项目，但代码在linux、mac平台下都可以编译通过。

需要说明的是为了统一处理，将bmp读取到LBitmap::m_pixel中时就将其转化为32位从上往下排列的图像格式了。并且会有y坐标所以在读取的时候会有一个temp_line先存储文件中的24位数据，再转化为32位数据。在保存时也是先将32位数据转化到temp_line上，然后再写入文件。(如果仅仅是处理bmp，那么这么多的一个A通道是冗余数据，但后面处理png图片时就会用到这个A通道)
如果用上面的代码来读取如图所示的图片(放大8倍后的显示图):



右上角像素为RGB(255, 128, 0)

```
1 In::LBitmap bmp;  
2 bmp.ReadBmp(L"one.bmp");  
3 unsigned char *p = bmp.Pixel(0, 0);  
4 printf("%d, %d, %d\n", p[0], p[1], p[2]); //显示左上角的像素值  
5 bmp.WriteBmp(L"out.bmp"); //保存到文件，可以测试是否能正确读取和保存bmp
```

运行的结果为: 0,128,255
可以看出像素分布为BGR

ps:

- bmp格式也是可以压缩.
- bmp格式也可以有颜色板。颜色板就是一个颜色的索引，上面说过bmp格式一个像素可以只有2个,16个或256个取值图来说明，默认为0对应RGB(0,0,0) 1,对应RGB(255, 255, 255)
如果颜色板这样定义:
0对应 RGB(255,0, 0)红

1对应 RGB(0, 255, 0)绿
这样黑白图就成了红绿图

JPEG格式

1. jpeg是有损压缩格式, 将像素信息用jpeg保存成文件再读取出来, 其中某些像素值会有少许变化。在保存时有个量化表 [0,100]之间选择, 参数越大图片就越保真, 但图片的体积也就越大。一般情况下选择70或80就足够了。
2. jpeg没有透明信息。
3. **jpeg比较适合用来存储相机拍出来的照片**, 这类图像用jpeg压缩后的体积比较小。其使用的具体算法核心是离散余弦变换、Huffman编码、算术编码等技术, 有兴趣的同学可以在网上找一大堆资料, 本文就不详细介绍了。

接下来要介绍一个有关jpeg非常实用的技术——

jpeg格式支持不完全读取整张图片, 即可以选择读取原图、1/2、1/4、1/8大小的图片

比如5000*5000的一张大图, 可以只读取将其缩小成1/8后即625*625大小的图片。 这样比先完全读取5000*5000的图像, 再用算625*625大小不知快多少倍。

如果应用需求只需要一张小图时, 这种读取方式就可以大显身手了。

在c代码中读取jpeg一般是使用libjpeg, 这个库提供了不完全读取图片的功能。

给In::LBitmap添加有关jpeg的接口, 如下ReadJpeg()第三个参数fraction可取值为1,2,4,8, 分别对应1/1,1/2,1/4,1/8

+	JpegAPI
---	---------

具体的实现在JpegDemo

用上面的函数进行jpeg的读取和保存的测试

```
...
In::LBitmap bmp;
bmp.ReadBmp(L"one.bmp");
unsigned char *p = bmp.Pixel(0, 0);
printf("%d, %d, %d\n", p[0], p[1], p[2]);
bmp.WriteJpeg(L"one.jpg", 90);
...
```

读取one.bmp图片, 然后保存成jpeg格式, one.jpg放大后显示如下



发现左上角的颜色发生了变化, 并且也影响到周围的像素, 就算将上面WriteJpeg()第二个参数换成100, 也还是这种效果, 这是JPG固有的问题

但如果读取一张风景照, 再保存成Jpeg, 就几乎看不出有什么差别了。

android平台下实现jpeg预读

```
BitmapFactory.Options opt = new BitmapFactory.Options();
opt.inJustDecodeBounds = true;
BitmapFactory.decodeFile(info.fullPath, opt); //这里仅仅只读取jpeg的大小
opt.inJustDecodeBounds = false;
if (opt.outWidth > opt.outHeight) {
    opt.inSampleSize = opt.outWidth / phSize; //hpSize是允许的图片宽高的最大值
} else {
    opt.inSampleSize = opt.outHeight / phSize;
}
Bitmap b = BitmapFactory.decodeFile(info.fullPath, opt);
```

将BitmapFactory.Options的inJustDecodeBounds 设置为true后, 就只会读取Jpeg的大小, 而不会去解析像素数据。然后再设后, 就可以根据这个值来读取适当大小的图片, 研究android的源码后可以发现底层也是调用的libjpeg库来实现。

ios,mac

本人还没有在ios/mac中发现如何预读jpeg的官方API。Apple对图形、图像、多媒体领域提供了丰富接口, 如果这个功能真没实现了! 不过Objective-C完全兼容C, 可以调用libjpeg库来实现这个功能。

.NET下仅读取jpeg的大小

下面是用c#仅仅读取jpeg宽高(没有解析像素数据), 直接用C#读取1/2,1/4,1/8还不知道如何实现

```
FileStream stream = new FileStream(path, FileMode.Open);
Image img = Image.FromStream(stream, false, false); //关键是将第三个参数设置为false
Console.WriteLine("size: {0},{1}", img.Width, img.Height);
```

jpeg批量转化工具

用相机拍出来的原始jpeg图片是高保真质量,所占文件体积非常大,本人写了一个批量转化的工具,可以将jpeg的质量都转化成80. 这时人眼几乎看不出有什么差别,但其体积只有原来的1/3. 如果有大量的照片需要保存时,节约的空间就很客观了. 实现原理很简单, 用c#实现的, 代码量非常少, 在此贴出全部源码

```
JpegBatchConvert
```

Exif信息

另外jpeg文件一般有一个附属的exif信息, 这个信息中有图像大小, 拍摄时间, 拍摄的相关参数, 照片方向, 图像缩略图等信息。

用相机拍出来的jpeg都会有这个信息。如果照片方向不是正立的话, 在读取到像素取后, 还得按exif所指明的方向将图像旋转下。没有做过这个处理, 有些图片用picasa查看和用mspaint查看方向就不一样。当然为了简单起见, 上面的LBitmap中也自动忽略了exif拍摄时的方向。

如果不用读取1/2,1/4,1/8的方法, 也可以从exif中来读取缩略图, 但这个缩略图一般很小。

说到exif, 不得不说一款用perl实现的命令行工具:exiftool。几乎所有的多媒体文件(图像、音乐、视频)都可以用这个工具来查看其信息。如果不是jpeg文件就是指广义上的"exif"。在git中有已经编译好可执行文件exiftool.exe。使用方法是把这个文件放到系统路径下的文件路径下执行 exiftool filename

在实现BatchJpeg工具时如果仅仅用上面实现的LBitmap来读取,保存, 将会失去exif信息, 而相片的拍摄时间等信息又很重要, 所以改用exiv2来读取写入exif。如果用c#, 用上面的代码exif信息会自动保留下来。默默地向c#致敬。

intelJpeg库

如果在win32环境下对jpeg IO速度有很高的要求, 可以使用intelJpeg库, 不开源, 但提供有*.h,*.lib文件。这个库可以大大提高速度。

当时分别用c#和c实现了jpeg批量转化工具, 在处理大量图片时发现c#用时居然只有c的一半。太奇怪了, 按理说, c的速度比c#快, 而事实是c慢了这么多。 最后发现问题就在libjpeg上, 用了intelJpeg后速度就和c#差不多了(猜想.NET内部也是用intelJpeg来处理)

PNG格式

- 1. png是一种无损压缩格式, 压缩大概是用行程编码算法。
- 2. png可以有透明效果。
- 3. png比较适合适量图,几何图。 比如本文中出现的这些图都是用png保存, 比用jpeg保存体积要小。

再强调一下: jpeg比较适合存储色彩“杂乱”的拍摄图片, png比较适合存储几何特征比较强的矢量图。

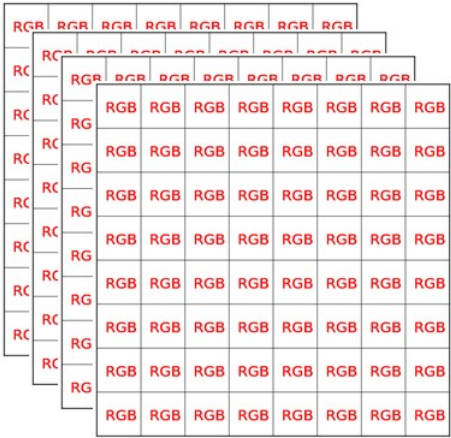
png可能有24位图和32位图之分。32位图就是带有alpha通道的图片。
将图片a绘制到另一幅图片b上, 如果图片a没有alpha通道, 那么就会完全将b图片的像素给替换掉。而如果有alpha通道, 那么最将是 $c = a * \alpha + b * (1 - \alpha)$
再对LBitmap添加png的支持。
添加接口如下:

```
static bool ReadPngSize(const wchar_t *path, int *width, int *height);
static bool IsPngFile(const wchar_t *filename);
bool ReadPng(const wchar_t *filename);
bool WritePng(const wchar_t *filename);
```

具体实现在PngDemo中。有调用libpng库, 并且libpng库依赖zlib库(由此可以看出png算法有用到常规的压缩算法)。

GIF格式

上面提到的bmp, jpeg, png图片都只有一帧, 而gif可以保存多帧图像, 如图所示




libgif库可以用来读取gif图片。gif中有个参数可以控制图片变化的快慢。在程序中使用这个参数, 也可以自己定义一个参数, 图片, 在不同程序中查看时其变化速度不一样。

google开发的一种有损、透明图片格式，相当于jpeg和png的合体，google声称其可以把图片大小减少40%。

一个强大的格式库,CxImage

CxImage几乎可以读取任何图片格式


下面是其头文件中的宏定义:



```
#define CXIMAGE_SUPPORT_WINDOWS 1
#define CXIMAGE_SUPPORT_EXIF 1
#define CXIMAGE_SUPPORT_BMP 1
#define CXIMAGE_SUPPORT_GIF 1
#define CXIMAGE_SUPPORT_JPG 1
#define CXIMAGE_SUPPORT_PNG 1
#define CXIMAGE_SUPPORT_ICO 1
#define CXIMAGE_SUPPORT_TIF 1
#define CXIMAGE_SUPPORT_TGA 1
#define CXIMAGE_SUPPORT_PCX 1
#define CXIMAGE_SUPPORT_WBMP 1
#define CXIMAGE_SUPPORT_WMF 1

#define CXIMAGE_SUPPORT_JP2 1
#define CXIMAGE_SUPPORT_JPC 1
#define CXIMAGE_SUPPORT_PGX 1
#define CXIMAGE_SUPPORT_PNM 1
#define CXIMAGE_SUPPORT_RAS 1

#define CXIMAGE_SUPPORT_MNG 1
#define CXIMAGE_SUPPORT_SKA 1
#define CXIMAGE_SUPPORT_RAW 1
#define CXIMAGE_SUPPORT_PSD 1
```



CxImage在针对特定格式时，也是调用了其它图片库(比如libjpeg, libpng, libtiff)。由于CxImage太过庞大，如果不想使用其全部从中“偷取”特定图片格式的读取、保存代码。

版权声明：本文来自xiangism的博客。仅供参考、学习使用。若需转载或引用本文中的方法，请标明原作者信息
<http://www.cnblogs.com/xiangism>。商业用途请联系博主

分类: 图形/图像

好文要顶 关注我 收藏该文



Xiangism

关注 - 11

粉丝 - 57

+加关注

« 上一篇 : 改写《python基础教程》中的一个例子

Feedback

#1楼
2016-03-31 11:37 by AlanWang
写的很不错啊
#2楼
2016-03-31 13:41 by 路上的脚印
其行字节数为24*5=120，应该是位数=120？
#3楼[楼主]
2016-03-31 13:46 by Xiangism
@ 路上的脚印 额，笔误，已改

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

- 【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】融云即时通讯云 - 豆果美食、Faceu等亿级APP都在用
- 【推荐】报表开发别头大！类Excel 复杂报表开发实例，即学即用
- 【推荐】福利Time，讯飞开放平台注册即送好礼！
- 【推荐】阿里云万网域名：.xin .com将推出重磅优惠



- 最新IT新闻:
- Apple Watch运动表带展示方式获得设计专利
 - 3名宇航员在太空生活186天后安全返回地球
 - 吴文辉上演“王子复仇记”后，阅文集团为何还起内乱
 - 微软测试Windows 10新还原工具
 - 上线24天完成A轮融资 分答快速圈钱背后：盈利模式前景难料
- » 更多新闻...

 **消息推送领导品牌全面升级**



- 最新知识库文章:
- 学习如何学习
 - 一个32岁入门的70后程序员给我的启示
 - 技术发展瓶颈的突破
 - 高效编程之道：好好休息
 - 快速学习者的高效学习策略
- » 更多知识库文章...