**PNG文件结构分析之一(了解PNG文件存储格式)[多图]**

网站编辑： 出处：http://www.hackhome.com/ 发表时间：2007-6-8 16:04:18 [发表评论](http://www.hackhome.com/InfoView/111967_full.html#comment)   
字体大小：【[小](javascript:void(0)) | [中](javascript:void(0)) | [大](javascript:void(0))】 相关tags: [文件](http://www.hackhome.com/InfoTags/%E6%96%87%E4%BB%B6.html)

**二维码扫描查看**：

**前言**

我们都知道，在进行J2ME的手机应用程序开发的时候，在图片的使用上，我们可以使用PNG格式的图片（甚至于在有的手机上，我们只可以使用PNG格式的图片），尽管使用图片可以为我们的应用程序增加不少亮点，然而，只支持PNG格式的图片却又限制了我们进一步发挥的可能性（其实，应该说是由于手机平台上的处理能力有限）。 在MIDP2中，或者某些厂商（如NOKIA）提供的API中，提供了drawPixels/getPixels的方法，这些方法进一步提高了开发者处理图片的灵活性，然而，在MIDP2还未完全普及的今天，我们需要在MIDP1 .0中实现这类方法还属于异想天开，因此，为了实现更高级的应用，我们必须充分挖掘PNG的潜力。

**PNG的文件结构**

对于一个PNG文件来说，其文件头总是由位固定的字节来描述的：

十进制数 137 80 78 71 13 10 26 10 十六进制数 89 50 4E 47 0D 0A 1A 0A

其中第一个字节0x89超出了ASCII字符的范围，这是为了避免某些软件将PNG文件当做文本文件来处理。文件中剩余的部分由3个以上的PNG的数据块（Chunk）按照特定的顺序组成，因此，一个标准的PNG文件结构应该如下：

PNG文件标志 PNG数据块 …… PNG数据块

**PNG数据块（Chunk）**

PNG定义了两种类型的数据块，一种是称为关键数据块(critical chunk)，这是标准的数据块，另一种叫做辅助数据块(ancillary chunks)，这是可选的数据块。关键数据块定义了4个标准数据块，每个PNG文件都必须包含它们，PNG读写软件也都必须要支持这些数据块。虽然PNG文件规范没有要求PNG编译码器对可选数据块进行编码和译码，但规范提倡支持可选数据块。

下表就是PNG中数据块的类别，其中，关键数据块部分我们使用深色背景加以区分。

**PNG文件格式中的数据块**

**数据块符号**

**数据块名称**

**多数据块**

**可选否**

**位置限制**

IHDR 文件头数据块 否 否 第一块 cHRM 基色和白色点数据块 否 是 在PLTE和IDAT之前 gAMA 图像γ数据块 否 是 在PLTE和IDAT之前 sBIT 样本有效位数据块 否 是 在PLTE和IDAT之前 PLTE 调色板数据块 否 是 在IDAT之前 bKGD 背景颜色数据块 否 是 在PLTE之后IDAT之前 hIST 图像直方图数据块 否 是 在PLTE之后IDAT之前 tRNS 图像透明数据块 否 是 在PLTE之后IDAT之前 oFFs (专用公共数据块) 否 是 在IDAT之前 pHYs 物理像素尺寸数据块 否 是 在IDAT之前 sCAL (专用公共数据块) 否 是 在IDAT之前 IDAT 图像数据块 是 否 与其他IDAT连续 tIME 图像最后修改时间数据块 否 是 无限制 tEXt 文本信息数据块 是 是 无限制 zTXt 压缩文本数据块 是 是 无限制 fRAc (专用公共数据块) 是 是 无限制 gIFg (专用公共数据块) 是 是 无限制 gIFt (专用公共数据块) 是 是 无限制 gIFx (专用公共数据块) 是 是 无限制 IEND 图像结束数据 否 否 最后一个数据块

为了简单起见，我们假设在我们使用的PNG文件中，这4个数据块按以上先后顺序进行存储，并且都只出现一次。

**数据块结构**

PNG文件中，每个数据块由4个部分组成，如下：

**名称 字节数 说明**Length (长度) 4字节 指定数据块中数据域的长度，其长度不超过(231－1)字节 Chunk Type Code (数据块类型码) 4字节 数据块类型码由ASCII字母(A-Z和a-z)组成 Chunk Data (数据块数据) 可变长度 存储按照Chunk Type Code指定的数据 CRC (循环冗余检测) 4字节 存储用来检测是否有错误的循环冗余码

CRC(cyclic redundancy check)域中的值是对Chunk Type Code域和Chunk Data域中的数据进行计算得到的。CRC具体算法定义在ISO 3309和ITU-T V.42中，其值按下面的CRC码生成多项式进行计算：

x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x+1

下面，我们依次来了解一下各个关键数据块的结构吧。

**IHDR**

文件头数据块IHDR(header chunk)：它包含有PNG文件中存储的图像数据的基本信息，并要作为第一个数据块出现在PNG数据流中，而且一个PNG数据流中只能有一个文件头数据块。

文件头数据块由13字节组成，它的格式如下表所示。

**域的名称**

**字节数**

**说明**

Width 4 bytes 图像宽度，以像素为单位 Height 4 bytes 图像高度，以像素为单位 Bit depth 1 byte 图像深度：   
索引彩色图像：1，2，4或8   
灰度图像：1，2，4，8或16   
真彩色图像：8或16 ColorType 1 byte 颜色类型：  
0：灰度图像, 1，2，4，8或16   
2：真彩色图像，8或16   
3：索引彩色图像，1，2，4或8   
4：带α通道数据的灰度图像，8或16   
6：带α通道数据的真彩色图像，8或16 Compression method 1 byte 压缩方法(LZ77派生算法) Filter method 1 byt 滤波器方法 Interlace method 1 byte 隔行扫描方法：  
0：非隔行扫描   
1： Adam7(由Adam M. Costello开发的7遍隔行扫描方法)

由于我们研究的是手机上的PNG，因此，首先我们看看MIDP1.0对所使用PNG图片的要求吧：

* 在MIDP1.0中，我们只可以使用1.0版本的PNG图片。并且，所以的PNG关键数据块都有特别要求：  
  IHDR
* 文件大小：MIDP支持任意大小的PNG图片，然而，实际上，如果一个图片过大，会由于内存耗尽而无法读取。
* 颜色类型：所有颜色类型都有被支持，虽然这些颜色的显示依赖于实际设备的显示能力。同时，MIDP也能支持alpha通道，但是，所有的alpha通道信息都会被忽略并且当作不透明的颜色对待。
* 色深：所有的色深都能被支持。
* 压缩方法：仅支持压缩方式0（deflate压缩方式），这和jar文件的压缩方式完全相同，所以，PNG图片数据的解压和jar文件的解压可以使用相同的代码。（其实这也就是为什么J2ME能很好的支持PNG图像的原因：））
* 滤波器方法：尽管在PNG的白皮书中仅定义了方法0，然而所有的5种方法都被支持！
* 隔行扫描：虽然MIDP支持0、1两种方式，然而，当使用隔行扫描时，MIDP却不会真正的使用隔行扫描方式来显示。
* PLTE chunk：支持
* IDAT chunk：图像信息必须使用5种过滤方式中的方式0 (None, Sub, Up, Average, Paeth)
* IEND chunk：当IEND数据块被找到时，这个PNG图像才认为是合法的PNG图像。
* 可选数据块：MIDP可以支持下列辅助数据块，然而，这却不是必须的。

bKGD cHRM gAMA hIST iCCP iTXt pHYs  
sBIT sPLT sRGB tEXt tIME tRNS zTXt

关于更多的信息，可以参考http://www.w3.org/TR/REC-png.html

**PLTE**

调色板数据块PLTE(palette chunk)包含有与索引彩色图像(indexed-color image)相关的彩色变换数据，它仅与索引彩色图像有关，而且要放在图像数据块(image data chunk)之前。

PLTE数据块是定义图像的调色板信息，PLTE可以包含1~256个调色板信息，每一个调色板信息由3个字节组成：

**颜色**

**字节**

**意义**

Red

1 byte

0 = 黑色, 255 = 红

Green

1 byte

0 = 黑色, 255 = 绿色

Blue

1 byte

0 = 黑色, 255 = 蓝色

因此，调色板的长度应该是3的倍数，否则，这将是一个非法的调色板。

对于索引图像，调色板信息是必须的，调色板的颜色索引从0开始编号，然后是1、2……，调色板的颜色数不能超过色深中规定的颜色数（如图像色深为4的时候，调色板中的颜色数不可以超过2^4=16），否则，这将导致PNG图像不合法。

真彩色图像和带α通道数据的真彩色图像也可以有调色板数据块，目的是便于非真彩色显示程序用它来量化图像数据，从而显示该图像。

**IDAT**

图像数据块IDAT(image data chunk)：它存储实际的数据，在数据流中可包含多个连续顺序的图像数据块。

IDAT存放着图像真正的数据信息，因此，如果能够了解IDAT的结构，我们就可以很方便的生成PNG图像。

**IEND**

图像结束数据IEND(image trailer chunk)：它用来标记PNG文件或者数据流已经结束，并且必须要放在文件的尾部。

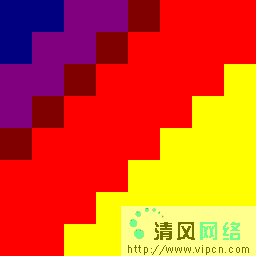
如果我们仔细观察PNG文件，我们会发现，文件的结尾12个字符看起来总应该是这样的：

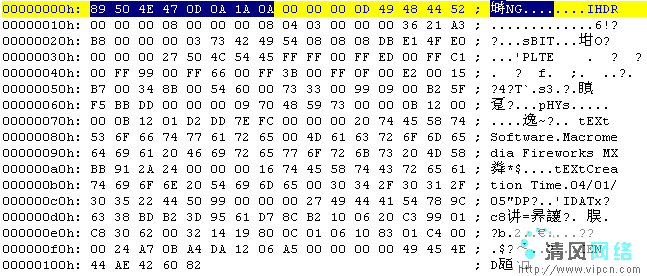
00 00 00 00 49 45 4E 44 AE 42 60 82

不难明白，由于数据块结构的定义，IEND数据块的长度总是0（00 00 00 00，除非人为加入信息），数据标识总是IEND（49 45 4E 44），因此，CRC码也总是AE 42 60 82。

**实例研究PNG**

以下是由Fireworks生成的一幅图像，图像大小为8\*8，PNG文件结构分析之一(了解PNG文件存储格式)[多图]图片1为了方便大家观看，我们将图像放大：

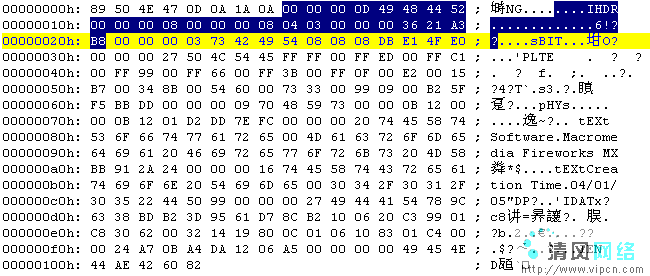
  
  
使用UltraEdit32打开该文件，如下：  
**00000000~00000007：**

 点击查看大图

可以看到，选中的头8个字节即为PNG文件的标识。

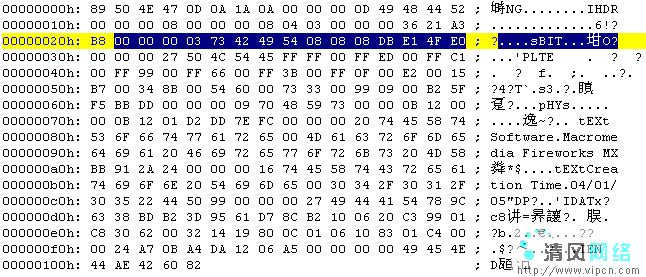
接下来的地方就是IHDR数据块了：

**00000008~00000020：**

 点击查看大图

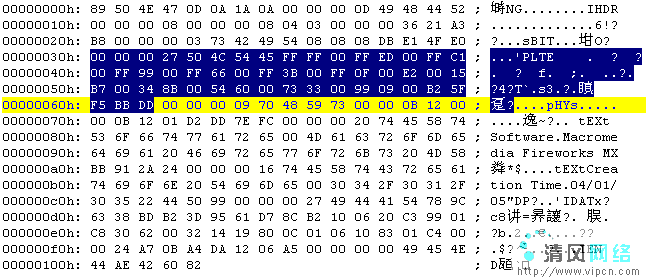
* 00 00 00 0D 说明IHDR头块长为13
* 49 48 44 52 IHDR标识
* 00 00 00 08 图像的宽，8像素
* 00 00 00 08 图像的高，8像素
* 04 色深，2^4=16，即这是一个16色的图像（也有可能颜色数不超过16，当然，如果颜色数不超过8，用03表示更合适）
* 03 颜色类型，索引图像
* 00 PNG Spec规定此处总为0（非0值为将来使用更好的压缩方法预留），表示使压缩方法(LZ77派生算法)
* 00 同上
* 00 非隔行扫描
* 36 21 A3 B8 CRC校验

**00000021~0000002F：**

 点击查看大图

可选数据块sBIT，颜色采样率，RGB都是256（2^8=256）

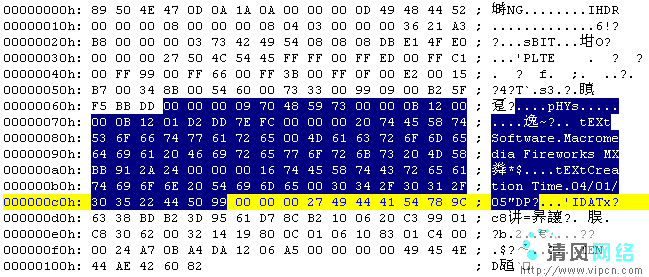
**00000030~00000062：**

 点击查看大图

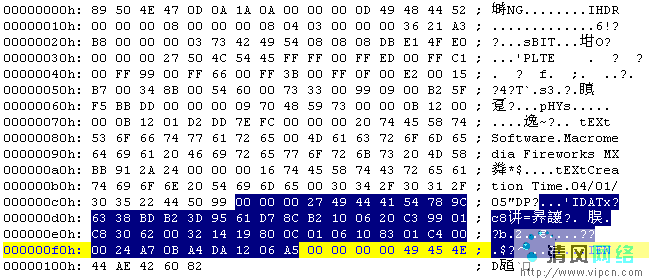
这里是调色板信息

* 00 00 00 27 说明调色板数据长为39字节，既13个颜色数
* 50 4C 54 45 PLTE标识
* FF FF 00 颜色0
* FF ED 00 颜色1
* …… ……
* 09 00 B2 最后一个颜色，12
* 5F F5 BB DD CRC校验

**00000063~000000C5：**

 点击查看大图

这部分包含了pHYs、tExt两种类型的数据块共3块，由于并不太重要，因此也不再详细描述了。  
  
**000000C0~000000F8：**

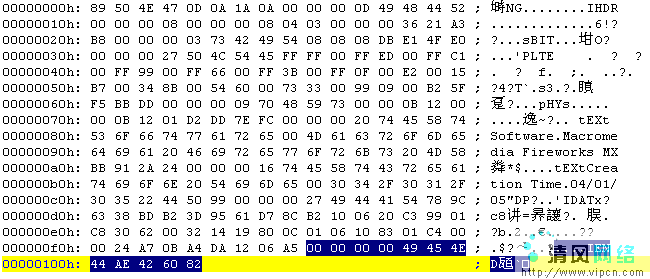
 点击查看大图

以上选中部分是IDAT数据块

* 00 00 00 27 数据长为39字节
* 49 44 41 54 IDAT标识
* 78 9C…… 压缩的数据，LZ77派生压缩方法
* DA 12 06 A5 CRC校验

IDAT中压缩数据部分在后面会有详细的介绍。

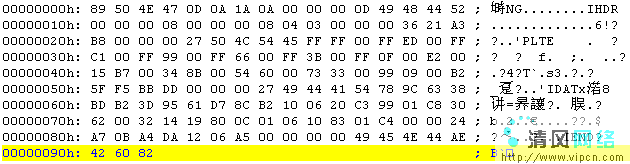
**000000F9~00000104：**

 点击查看大图

IEND数据块，这部分正如上所说，通常都应该是

00 00 00 00 49 45 4E 44 AE 42 60 82

至此，我们已经能够从一个PNG文件中识别出各个数据块了。由于PNG中规定除关键数据块外，其它的辅助数据块都为可选部分，因此，有了这个标准后，我们可以通过删除所有的辅助数据块来减少PNG文件的大小。（当然，需要注意的是，PNG格式可以保存图像中的层、文字等信息，一旦删除了这些辅助数据块后，图像将失去原来的可编辑性。）

 点击查看大图

删除了辅助数据块后的PNG文件，现在文件大小为147字节，原文件大小为261字节，文件大小减少后，并不影响图像的内容。

其实，我们可以通过改变调色板的色值来完成一些又趣的事情，比如说实现云彩/水波的流动效果，实现图像的淡入淡出效果等等，在此，给出一个链接给大家看也许更直接：http://blog.csdn.net/flyingGhost/archive/2005/01/13/251110.aspx，我写此文也就是受此文的启发的。

如上说过，IDAT数据块是使用了LZ77压缩算法生成的，由于受限于手机处理器的能力，因此，如果我们在生成IDAT数据块时仍然使用LZ77压缩算法，将会使效率大打折扣，因此，为了效率，只能使用无压缩的LZ77算法，关于LZ77算法的具体实现，此文不打算深究，如果你对LZ77算法的Java实现有兴趣，可以参考以下两个站点：

* http://jazzlib.sourceforge.net/
* http://www.jcraft.com/jzlib/index.html

**参考资料：**

PNG文件格式白皮书：http://www.w3.org/TR/REC-png.html  
为数不多的中文PNG格式说明：http://dev.gameres.com/Program/Visual/Other/PNGFormat.htm  
RFC-1950(ZLIB Compressed Data Format Specification)：FTP://ds.internic.net/rfc/rfc1950.txt  
RFC-1950(DEFLATE Compressed Data Format Specification)：ftp://ds.internic.net/rfc/rfc1951.txt  
LZ77算法的JAVA实现：http://jazzlib.sourceforge.net/  
LZ77算法的JAVA实现，包括J2ME版本：http://www.jcraft.com/jzlib/index.html