# GIF格式文件结构

一个GIF文件的结构可分为文件头(File Header)、GIF数据流(GIF Data Stream)和文件终结器(Trailer)三个部分。文件头包含GIF文件署名(Signature)和版本号(Version)；GIF数据流由控制标识符、图象块(Image Block)和其他的一些扩展块组成；文件终结器只有一个值为0x3B的字符（';'）表示文件结束。下表显示了一个GIF文件的组成结构：

|  |  |  |
| --- | --- | --- |
| GIF署名 | | 文件头 |
| 版本号 | |
| 逻辑屏幕标识符 | | GIF数据流 |
| 全局颜色列表 | |
| … | |
| 图象标识符 | 图象块 |
| 图象局部颜色列表图 |
| 基于颜色列表的图象数据 |
| … | |
| GIF结尾 | | 文件结尾 |

GIF格式的文件结构整体上分为三部分：文件头、GIF数据流、文件结尾。其中，GIF数据流分为全局配置和图像块。接下来我们将逐一分析GIF格式各部分的作用，并结合Glide的代码，学习如何解析。

# GIF署名(Signature)和版本号(Version)

GIF的前6个字节内容是GIF的署名和版本号。我们可以通过前3个字节判断文件是否为GIF格式，后3个字节判断GIF格式的版本。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | BIT |
| 1 | ‘G’ | | | | | | | | GIF文件标识 |
| 2 | ‘I’ | | | | | | | |
| 3 | ‘F’ | | | | | | | |
| 4 | ‘8’ | | | | | | | | GIF文件版本号:  87a - 1987年5月  89a - 1989年7月 |
| 5 | ‘7’或’9’ | | | | | | | |
| 6 | ‘a’ | | | | | | | |

# 逻辑屏幕标识符(Logical Screen Descripotr)

逻辑屏幕标识符配置了GIF一些全局属性，我们通过读取解析它，获取GIF全局的一些配置

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | BIT |
| 1 | 逻辑屏幕宽度 | | | | | | | | 像素数，定义GIF图象的宽度 |
| 2 |
| 3 | 逻辑屏幕高度 | | | | | | | | 像素数，定义GIF图象的高度 |
| 4 |
| 5 | m | cr | | | s | pixel | | | 具体描述见下… |
| 6 | 背景色 | | | | | | | | 背景颜色（在全局颜色列表中的索引，如果没有全局颜色列表，该值没有意义） |
| 7 | 像素宽高比 | | | | | | | | 像素宽高比(Pixel Aspect Radio) |

屏幕逻辑宽度：定义了GIF图像的像素宽度，大小为2字节;

屏幕逻辑高度：定义了GIF图像的像素高度，大小为2字节;

m - 全局颜色列表标志(Global Color Table Flag)，当置位时表示有全局颜色列表，pixel值有意义;

cr - 颜色深度(Color ResoluTion)，cr+1确定图象的颜色深度;

s - 分类标志(Sort Flag)，如果置位表示全局颜色列表分类排列;

pixel - 全局颜色列表大小，pixel+1确定颜色列表的索引数（2^(pixel+1)）;

背景颜色：背景颜色在全局颜色列表中的索引（PS:是索引而不是RGB值，所以如果没有全局颜色列表时，该值没有意义）;

像素宽高比：全局像素的宽度与高度的比值;

# 全局颜色列表(Global Color Table)

全局颜色列表，在逻辑屏幕标识之后，每个颜色索引由三字节组成，按RGB顺序排列。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | BIT |
| 1 | 索引1的红色值 | | | | | | | |  |
| 2 | 索引1的绿色值 | | | | | | | |  |
| 3 | 索引1的蓝色值 | | | | | | | |  |
| 4 | 索引2的红色值 | | | | | | | |  |
| 5 | 索引2的绿色值 | | | | | | | |  |
| 6 | 索引2的蓝色值 | | | | | | | |  |
| 7 | … | | | | | | | |  |

这里可以说明一下。整个GIF在每一帧的画面数组时，是不会出现RGB值的，画面中所有像素的RGB值，都是通过从全局/局部颜色列表中取得。可以让颜色列表理解为调色板。我需要什么RGB，我不能直接写，而是写我想要RGB对应颜色列表的索引。

这样做的好处，比如我想对GIF进行调色，如果我每一帧画面直接使用了RGB，那我每一帧都需要进行图像处理。有了调色盘，我只需要对调色板进行处理，每帧画面都会改变。

# 图像标识符(Image Descriptor)

一个GIF文件中可以有多个图像块，每个图像块就会有图像标识符，描述了当前帧的一些属性。下面我们来看看图像标识符中包含的一些信息。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | BIT |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 图象标识符开始，固定值为’,’(0x2c) |
| 2 | X方向偏移量 | | | | | | | | 必须限定在逻辑屏幕尺寸范围内 |
| 3 |
| 4 | Y方向偏移量 | | | | | | | |
| 5 |
| 6 | 图象宽度 | | | | | | | |
| 7 |
| 8 | 图象高度 | | | | | | | |
| 9 |
| 10 | m | i | s | r | | pixel | | | m-局部颜色列表标志(Local Color Table Flag) |

图像标识符以','(0x2c)作为开始标志。接着定义了当前帧的偏移量和宽高。

最后5个标志的意义分别为：

m - 局部颜色列表标志(Local Color Table Flag)

置位时标识紧接在图象标识符之后有一个局部颜色列表，供紧跟在它之后的一幅图象使用；值否时使用全局颜色列表，忽略pixel值。

i - 交织标志(Interlace Flag)，置位时图象数据使用交织方式排列，否则使用顺序排列。

s - 分类标志(Sort Flag)，如果置位表示紧跟着的局部颜色列表分类排列.

r - 保留，必须初始化为0.

pixel - 局部颜色列表大小(Size of Local Color Table)，pixel+1就为颜色列表的位数

这一段除了交织标志外，其他的与全局配置类似，比较容易理解。交织标志将在图片的解码时单独解释。

# 局部颜色列表(Local Color Table)

如果上面的局部颜色列表标志置位的话，则需要在这里（紧跟在图象标识符之后）定义一个局部颜色列表以供紧接着它的图象使用，注意使用前应线保存原来的颜色列表，使用结束之后回复原来保存的全局颜色列表。如果一个GIF文件即没有提供全局颜色列表，也没有提供局部颜色列表，可以自己创建一个颜色列表，或使用系统的颜色列表。局部颜色列表的排列方式和全局颜色列表一样：RGBRGB......

# 基于颜色列表的图像数据(Table-Based Image Data)

基于颜色列表的图像数据必须紧跟在图像标识符后面；

由两部分组成：LZW编码长度(LZW Minimum Code Size)和图像数据(Image Data)。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | BIT |
| 1 | LZW编码长度 | | | | | | | | LZW编码初始码表大小的位数，详细描述见LZW编码… |
|  | … | | | | | | | | 图像数据，由一个或几个数据块(Data Sub-blocks)组成 |
|  | 数据块 | | | | | | | |
|  | … | | | | | | | |

GIF文件内部是按块划分的，包括控制块（ Control Block ）和数据块（Data Sub-blocks）两种。控制块是控制数据块行为的，根据不同的控制块包含一些不同的控制参数；数据块只包含一些8-bit的字符流，由它前面的控制块来决定它的功能，每个数据块大小从0到255个字节，数据块的第一个字节指出这个数据块大小（字节数），计算数据块的大小时不包括这个字节，所以一个空的数据块有一个字节，那就是数据块的大小0x00。下表是一个数据块的结构：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | BIT |
| 0 | 块大小 | | | | | | | | Block Size - 块大小，不包括这个字节（不计算块大小自身） |
| 1 |  | | | | | | | | Data Value - 块数据，8-BIT的字符串 |
| 2 |  | | | | | | | |
| … |  | | | | | | | |
|  | | | | | | | |
| 254 |  | | | | | | | |
| 255 |  | | | | | | | |

# 扩展控制块（GIF89a）

## 图形控制扩展(Graphic Control Extension)

在89a版本，GIF添加了图形控制扩展块。放在一个图象块(图象标识符)的前面，用来控制紧跟在它后面的第一个图象的显示。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | BIT |
| 1 | 扩展块标识 | | | | | | | | Extension Introducer – 标识这是一个扩展块，固定值0x21 |
| 2 | 图形控制扩展标签 | | | | | | | | Graphic Control Label – 标识这是一个图形控制扩展块，固定值 0xF9 |
| 3 | 块大小 | | | | | | | | Block Size – 不包括块终结器，固定值4 |
| 4 | 保留 | | 处置方法 | | | | i | t | i-用户输入标志；t-透明色标志。详细描述见下… |
| 5 | 延迟时间 | | | | | | | | Delay Time – 单位1/100秒，如果不为1，表示暂停规定的时间后再继续往下处理数据流 |
| 6 |
| 7 | 透明色索引 | | | | | | | | Transparent Color Index – 透明色索引值 |
| 8 | 块终结器 | | | | | | | | Block Terminator – 标识块终结，固定值 0 |

处置方法(Disposal Method)：指出处置图形的方法，当值为：

0 - 不使用处置方法

1 - 不处置图形，把图形从当前位置移去

2 - 回复到背景色

3 - 回复到先前状态

4-7 - 自定义用户输入标志(Use Input Flag)：指出是否期待用户有输入之后才继续进行下去，置位表示期待，值否表示不期待。

用户输入可以是按回车键、鼠标点击等，可以和延迟时间一起使用，在设置的延迟时间内用户有输入则马上继续进行，或者没有输入直到延迟时间到达而继续。

透明颜色标志(Transparent Color Flag)：置位表示使用透明颜色。

## 注释扩展(Comment Extension)

这一部分是可选的（需要89a版本），可以用来记录图形、版权、描述等任何的非图形和控制的纯文本数据(7-bit ASCII字符)，注释扩展并不影响对图象数据流的处理，解码器完全可以忽略它。存放位置可以是数据流的任何地方，最好不要妨碍控制和数据块，推荐放在数据流的开始或结尾。具体组成：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | BIT |
| 1 | 扩展块标识 | | | | | | | | Extension Introducer – 标识这是一个扩展块，固定值0x21 |
| 2 | 注释块标签 | | | | | | | | Comment Label – 标识这是一个注释块，固定值oxFE |
|  | … | | | | | | | | Comment Data – 一个或多个数据块组成 |
| 注释块 | | | | | | | |
| … | | | | | | | |
|  | 块终结器 | | | | | | | | Block Terminator – 标识注释块结束，固定值0 |

## 图形文本扩展(Plain Text Extension)

这一部分是可选的（需要89a版本），用来绘制一个简单的文本图象，这一部分由用来绘制的纯文本数据（7-bit ASCII字符）和控制绘制的参数等组成。绘制文本借助于一个文本框（Text Grid）来定义边界，在文本框中划分多个单元格，每个字符占用一个单元，绘制时按从左到右、从上到下的顺序依次进行，直到最后一个字符或者占满整个文本框（之后的字符将被忽略，因此定义文本框的大小时应该注意到是否可以容纳整个文本），绘制文本的颜色使用全局颜色列表，没有则可以使用一个已经保存的前一个颜色列表。另外，图形文本扩展块也属于图形块(Graphic Rendering Block)，可以在它前面定义图形控制扩展对它的表现形式进一步修改。图形文本扩展的组成：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE |  |  |  |  |  |  |  |  | BIT |
| 1 | 扩展标识 | | | | | | | | Extension Introducer - 标识这是一个扩展块，固定值0x21 |
| 2 | 图形控制扩展标签 | | | | | | | | Plain Text Label - 标识这是一个图形文本扩展块，固定值0x01 |
| 3 | 块大小 | | | | | | | | Block Size - 块大小，固定值12 |
| 4 | 文本框左边界位置 | | | | | | | | Text Glid Left Position - 像素值，文本框离逻辑屏幕的左边界距离 |
| 5 |
| 6 | 文本框上边界位置 | | | | | | | | Text Glid Top Position - 像素值，文本框离逻辑屏幕的上边界距离 |
| 7 |
| 8 | 文本框宽度 | | | | | | | | Text Glid Width -像素值 |
| 9 |
| 10 | 文本框高度 | | | | | | | | Text Glid Height - 像素值 |
| 11 |
| 12 | 字符单元格宽度 | | | | | | | | Character Cell Width - 像素值，单个单元格宽度 |
| 13 | 字符单元格高度 | | | | | | | | Character Cell Height- 像素值，单个单元格高度 |
| 14 | 文本前景色索引 | | | | | | | | Text Foreground Color Index - 前景色在全局颜色列表中的索引 |
| 15 | 文本背景色索引 | | | | | | | | Text Blackground Color Index - 背景色在全局颜色列表中的索引 |
| N | … | | | | | | | | Plain Text Data - 一个或多个数据块组成，保存要在显示的字符串。 |
| 文本数据块 | | | | | | | |
| … | | | | | | | |
| N+1 | 块终结 | | | | | | | | Block Terminator - 标识注释块结束，固定值0 |

推荐：1.由于文本的字体(Font)和尺寸(Size)没有定义，解码器应该根据情况选择最合适的；

2.如果一个字符的值小于0x20或大于0xF7，则这个字符被推荐显示为一个空格(0x20)；

3.为了兼容性，最好定义字符单元格的大小为8x8或8x16（宽度x高度）。

## 应用程序扩展(Application Extension)

这是提供给应用程序自己使用的（需要89a版本），应用程序可以在这里定义自己的标识、信息等，组成：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | BIT |
| 1 | 扩展块标识 | | | | | | | | Extension Introducer – 标识这是一个扩展块，固定值 0x21 |
| 2 | 图形控制扩展标签 | | | | | | | | Application Extension Label – 标识这是一个应用程序扩展块，固定值 0xFF |
| 3 | 块大小 | | | | | | | | Block Size – 块大小,固定值11 |
| 4 | 应用程序标识符 | | | | | | | | Application Identifier – 用来鉴别应用程序自身的标识（8个连续ASCII字符） |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |
| 11 |
| 12 | 应用程序鉴别码 | | | | | | | | Application Authentication Code – 应用程序定义的特殊标识码（3个连续ASCII字符） |
| 13 |
| 14 |
| N | … | | | | | | | | 应用程序自定义数据块 – 一个或多个数据块组成，保存应用程序自己定义的数据 |
| 应用程序数据 | | | | | | | |
| … | | | | | | | |
| N+1 | 块终结器 | | | | | | | | Lock Terminator - 标识注释块结束，固定值0 |

# 文件终结(Trailer)

这部分只有一个字节，标识一个GIF文件结束，当解析程序读到0x3B时，文件终结。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BYTE | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
| 1 | 文件终结 | | | | | | | | GIF Trailer – 标识GIF文件结束，固定值0x3B |