



# QOI 图像格式

Specification Version 1.0, 2022.01.05 - qoifformat.org - Dominic Szablewski  
Translation to Chinese by Smgdream, Proof-read by Wenqi. Version: 0.2.2 2025.01.14

一个 QOI 文件包含一个 14-byte 的头, 随后跟着任意数量的 data "chunks" (数据"区块"), 之后是一个 8-byte 的结束标识。

```
qoi_header {
    char    magic[4];        // 幻数字节"qoif"
    uint32_t width;          // 图像宽度像素(大端)
    uint32_t height;         // 图像高度像素(大端)
    uint8_t channels;        // 3 = RGB, 4 = RGBA
    uint8_t colorspace;      // 0 = sRGB 附带线性的 alpha 通道
                                // 1 = 所有通道皆是线性
};
```

colorspace 和 channels 字段仅提供信息。它们不会改变 data chunks 的编码方式。

图像编码方式为: 逐行渐进、从左到右、自上而下。解码器和编码器开始时将 { r: 0, g: 0, b: 0, a: 255 } 作为第一个像素的前一个像素的值。一个完整的图像其像素应当覆盖由 width \* height 所规定的所有像素点。像素的编码方式如下:

- 像素行程编码
- 之前遇见的像素组成的数组的索引
- 当前 r,g,b 与上一个像素的差值
- 完整的 r,g,b 和 r,g,b,a 数值

各色彩通道假设为未预乘 alpha 通道("un-permultiplied alpha")。

编码器和解码器维护一个内容可变数组 `array[64]` (全 0 初始化), 其元素为先前遇见的像素值。这些被编码器和解码器所见的像素将会存进该数组中, 其位置根据 hash 函数通过颜色值计算而得。在编码器中, 如果在索引中的像素值匹配到当前像素, 这个索引位置将会作为 `QOI_OP_INDEX` 写入到 stream 中。获取索引值的 hash 函数如下:

```
index_position = (r * 3 + g * 5 + b * 7 + a * 11) % 64;
```

每个 chunk 起始于一个 2 或 8 bit 的 tag (标签), 随后跟着数个 data bits。Chunks 的 bit 长度可被 8 整除——即: 所有 chunks 都字节对齐。所有从 data bits 编码而得的值有最高有效位在左边。8-bit tag 优先于 2-bit tag。一个解码器必须首先检查 tag 是否为 8-bit tag。

字节流结尾的标识为 7 字节的 0x00 再跟一字节的 0x01。

可能的区块种类如下:

QOI_OP_RGB											
Byte[0]								Byte[1]	Byte[2]	Byte[3]	
7	6	5	4	3	2	1	0	7 .. 0	7 .. 0	7 .. 0	
1	1	1	1	1	1	1	0	red	green	blue	

8-bit tag b1111110  
8-bit red channel value  
8-bit green channel value  
8-bit blue channel value

余下的 alpha 值与上一个像素的 alpha 相比没有改变。

QOI_OP_RGBA												
Byte[0]								Byte[1]	Byte[2]	Byte[3]	Byte[4]	
7	6	5	4	3	2	1	0	7 .. 0	7 .. 0	7 .. 0	7 .. 0	
1	1	1	1	1	1	1	1	red	green	blue	alpha	

8-bit tag b1111111  
8-bit red channel value  
8-bit green channel value  
8-bit blue channel value  
8-bit alpha channel value

QOI_OP_INDEX							
Byte[0]							
7	6	5	4	3	2	1	0
0	0	index					

2-bit tag b00  
6-bit 索引, 索引到颜色数组中: [0, 63]

一个有效的编码器不应生成 2 个及以上连续的指向同一个索引的 QOI\_OP\_INDEX 区块。而应该用 QOI\_OP\_RUN 替代。

QOI_OP_DIFF											
Byte[0]								Byte[1]			
7	6	5	4	3	2	1	0				
0	1	dr		dg		db					

2-bit tag b01  
2-bit 红色通道与上一个像素的差值 [-2, 1]  
2-bit 绿色通道与上一个像素的差值 [-2, 1]  
2-bit 蓝色通道与上一个像素的差值 [-2, 1]

当前通道差值计算使用(无符号)溢出操作, 所以 1 - 2 得 255, 255 + 1 得 0。

值存储为无符号整数且偏移量为 2, 例如 -2 被存储为 0(b00), 1 被存为 3(b11)。

余下的 alpha 值与上一个像素的 alpha 相比没有改变。

QOI_OP_LUMA															
Byte[0]								Byte[1]							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	diff green						dr - dg				db - dg			

2-bit tag b10  
6-bit 绿色通道与上一个像素差值 [-32, 31]  
4-bit 红色通道的差值减绿色通道的差值 [-8, 7]  
4-bit 蓝色通道的差值减绿色通道的差值 [-8, 7]

绿色通道用于指示常规的改变方向(即像素整体的改变趋势)并被编码为 6-bit 的数据。红色和蓝色通道(dr 和 db)基于它们与绿色通道差值的差异。即:

```
dr_dg = (cur_px.r - prev_px.r) - (cur_px.g - prev_px.g)
db_dg = (cur_px.b - prev_px.b) - (cur_px.g - prev_px.g)
```

当前通道的差值计算使用(无符号)溢出操作, 所以 10 - 13 得 253, 而 250 + 7 得 1。

值存储为无符号整型, 绿色通道偏移量为 32, 红色和蓝色通道的偏移量为 8。

余下的 alpha 值与上一个像素的 alpha 相比没有改变。

QOI_OP_RUN							
Byte[0]							
7	6	5	4	3	2	1	0
1	1	run					

2-bit tag b11  
6-bit 行程编码重复之前的像素 [1, 63]

行程编码所存储的值偏移量为 -1。注意: 行程编码为 63 和 64(b11111110 和 b11111111)是非法的, 因为它们占用了 QOI\_OP\_RGB 和 QOI\_OP\_RGBA 的标签。