

纹理压缩

1 目的

- (1) 减少显存占用；
- (2) 降低数据传输的带宽。

2 纹理压缩格式

2.1 S3TC/DXTC

2.1.1 介绍

S3TC 是 S3 公司提出 Texture Compression 算法,也叫做 DXTC(Direct Texture Compression)。基本思想为块压缩,以原始大小存储一些颜色,然后以不同的编码方案存储其他颜色。如图1所示,将未压缩的纹理分解成 4×4 的纹理块,对每个块进行压缩,此处要求被压缩的纹理的维度必须是 4 的倍数。

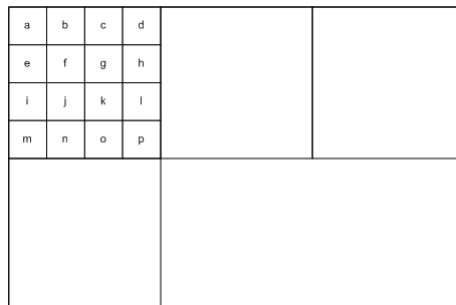


图 1: Block Compress

2.1.2 BC1

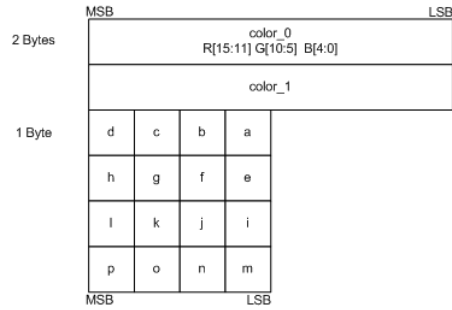


图 2: BC1

如图2所示，BC1 压缩格式对 4×4 的纹理块进行处理，BC1 并不是存储 16 种颜色，而是存储两种颜色（*color_1* 和 *color_2*）以及 16 个 2 位颜色指数，两种颜色都是用 5:6:5（5 位红色、6 位绿色、5 位蓝色）进行编码，将所需内存从 48 字节减少到 8 字节。能表示如下四种颜色：

color_0

color_1

$$color_2 = 2/3 * color_0 + 1/3 * color_2$$

$$color_3 = 1/3 * color_0 + 2/3 * color_2$$

四种分别对应 00、01、10 和 11，纹理块中的每种颜色和四种颜色比较后，存储最接近的颜色在纹理块中。

该算法同样适用于包含 1 位 alpha 的数据，进行如下设置：

color_0

color_1

$$color_2 = 1/2 * color_0 + 1/2 * color_2$$

$$color_3 = 0$$

2.1.3 BC2

相比于 BC1，BC2 更加关注低一致性的 alpha 数据。BC2 将 alpha 存储为单独的 4 位值，将 64 字节压缩到 16 字节，如图3所示。



图 3: BC2

2.1.4 BC3

BC3 用于存储高一致性的颜色数据（对于一致性较低的 alpha 数据使用 BC2）。BC3 的颜色数据格式与 BC1 相同，对于 alpha 数据，BC3 存储两个参考 alpha (*alpha_0* 和 *alpha_1*)，以及 16 个 3 位颜色指数，将 64 字节压缩到 16 字节，如图4所示。

BC3 的 alpha 查找表共有 8 个元素，除了 *alpha_0* 和 *alpha_1*，计算方法如下：

```

1 if( alpha_0 > alpha_1 )
2 {
3     // 6 interpolated alpha values.
4     alpha_2 = 6/7*alpha_0 + 1/7*alpha_1; // bit code 010
5     alpha_3 = 5/7*alpha_0 + 2/7*alpha_1; // bit code 011
6     alpha_4 = 4/7*alpha_0 + 3/7*alpha_1; // bit code 100

```

```

7   alpha_5 = 3/7*alpha_0 + 4/7*alpha_1; // bit code 101
8   alpha_6 = 2/7*alpha_0 + 5/7*alpha_1; // bit code 110
9   alpha_7 = 1/7*alpha_0 + 6/7*alpha_1; // bit code 111
10  }
11  else
12  {
13      // 4 interpolated alpha values.
14      alpha_2 = 4/5*alpha_0 + 1/5*alpha_1; // bit code 010
15      alpha_3 = 3/5*alpha_0 + 2/5*alpha_1; // bit code 011
16      alpha_4 = 2/5*alpha_0 + 3/5*alpha_1; // bit code 100
17      alpha_5 = 1/5*alpha_0 + 4/5*alpha_1; // bit code 101
18      alpha_6 = 0;                          // bit code 110
19      alpha_7 = 255;                         // bit code 111
20  }

```



图 4: BC3

2.1.5 BC4

BC4 用于存储单通道数据，如灰度图。BC4 存储两种参考颜色 (red_0 和 red_1)，以及 16 个 3 位颜色指数，将 16 字节压缩到 8 字节，如图5所

示。

BC4 和 BC3 一样，通过检查两个参考颜色来确定内插颜色值的数量。如果 red_0 大于 red_1 ，BC4 内插 6 个颜色值；否则，内插 4 个颜色值。当 BC4 内插 4 个颜色值，会设置两个额外的颜色值（0.0f 表示完全透明，1.0f 表示完全不透明）。BC4_UNORM 存储 $[0, 1]$ 范围内的浮点数据，BC4_SNORM 存储 $[-1, 1]$ 范围内的浮点数据。

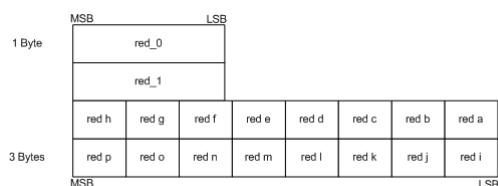


图 5: BC4

2.1.6 BC5

BC5 用于存储两个颜色分量的数据，每个分量存储 2 种参考颜色 (red_0 、 red_1 、 $green_0$ 、 $green_1$) 和 16 个三位颜色指数，将 32 个字节压缩到 16 个字节，如图6所示。每个分量的插值策略与 BC4 相同，BC5_UNORM 存储 $[0, 1]$ 范围内的浮点数据，BC5_SNORM 存储 $[-1, 1]$ 范围内的浮点数据。

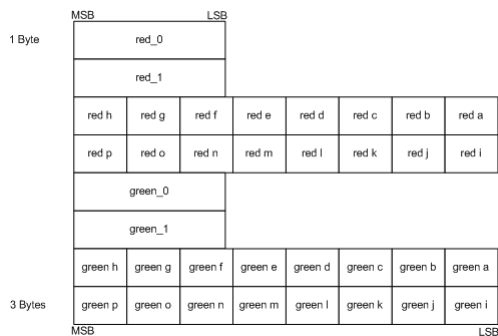


图 6: BC5

2.1.7 BC6H

BC6H 用于存储 HDR 颜色数据，存储三个颜色通道，每个通道存储 16 位值，不支持 alpha 通道。16 位浮点格式分为以下两种形式：

- (1) UF16: 无符号浮点数，包含 5 个指数位和 11 个尾数位；
- (2) SF16: 有符号浮点数，包含 1 个符号位、5 个指数位和 10 个尾数位。

BC6H 处理的对象仍然是 4×4 的纹理块，BC6H 将其压缩成 128bit 的数据，处理后的 BC6H 块包含 mode bits、compressed endpoints、compressed indices 和可选的 partition index。

(1) BC6H 对 4×4 的纹理块的处理模式分为单分区和双分区，单分区只有两个 color，用于插值；双分区各有两个 color；

(2) BC6H 总共有 14 种模式，如图7所示；

(2.1) 模式 1-10 用于双分区，mode bit 可以为 2 或 5，以模式 3 为例，表示 color endpoint 的精度 level 为 11，存储变换后的 endpoint 的 delta value 的精度分别为 5bit (red)、4bit (green)、4bit (blue)；模式 10 没有使用 delta 压缩，直接存储 4 个 color endpoint；

(2.2) 模式 11-14 用于单分区，mode bit 固定为 5，以模式 11 为例，没有使用 delta 压缩，直接存储两个 color endpoint；

Mode	Partition Indices	Partition	Color Endpoints	Mode Bits
1	46 bits	5 bits	75 bits (10.555, 10.555, 10.555)	2 bits (00)
2	46 bits	5 bits	75 bits (7666, 7666, 7666)	2 bits (01)
3	46 bits	5 bits	72 bits (11.555, 11.444, 11.444)	5 bits (00010)
4	46 bits	5 bits	72 bits (11.444, 11.555, 11.444)	5 bits (00110)
5	46 bits	5 bits	72 bits (11.444, 11.444, 11.555)	5 bits (01010)
6	46 bits	5 bits	72 bits (9555, 9555, 9555)	5 bits (01110)
7	46 bits	5 bits	72 bits (8666, 8555, 8555)	5 bits (10010)
8	46 bits	5 bits	72 bits (8555, 8666, 8555)	5 bits (10110)
9	46 bits	5 bits	72 bits (8555, 8555, 8666)	5 bits (11010)
10	46 bits	5 bits	72 bits (6666, 6666, 6666)	5 bits (11110)
11	63 bits	0 bits	60 bits (10.10, 10.10, 10.10)	5 bits (00011)
12	63 bits	0 bits	60 bits (11.9, 11.9, 11.9)	5 bits (00111)
13	63 bits	0 bits	60 bits (12.8, 12.8, 12.8)	5 bits (01011)
14	63 bits	0 bits	60 bits (16.4, 16.4, 16.4)	5 bits (01111)

图 7: BC6H 模式

(3) 对于双分区，BC6H 共有 32 种分割情况，如图8所示；图7中的

Partition 对应选择哪种分割情况，图8中有 1 被添加了下划线和加粗，表示使用了端点对称性的技巧：假设用 3bit 来表示 color1 和 color2 的插值权重，如果权重最高位为 0，那么我们舍弃这个 0，像素的权重位可以节省 1 位，如果最高位为 1，可以交换 color1 和 color2，那么权重最高位会由 1 变成 0，还是可以节省 1 位；

对应双分区，0 分区固定第一个像素节省最高位，1 分区根据分区表，对应像素节省最高位，每个索引占用 3bit，有 $16 \times 3 - 2 = 46$ bit；

对应单分区，每个像素的索引占用 4 位，第一个像素节省最高位，有 $16 \times 4 - 1 = 63$ bit。

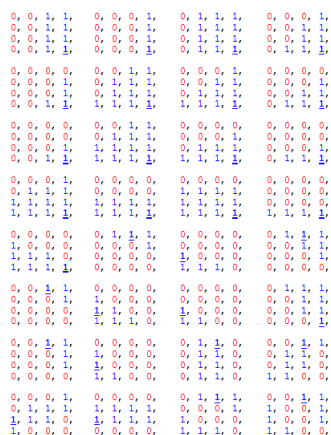


图 8: BC6H 分割情况

2.1.8 BC7

BC7 格式是用于 RGB 和 RGBA 数据的高质量压缩的纹理压缩格式。BC7 对 4×4 的纹理块进行处理，用 16 字节（128 位）的固定块进行表示。

BC7 block contains...	mode bits	rotation bits	index selector bit	partition bits	compressed endpoints	P-bit	compressed indices
color components only	required	N/A	N/A	required	required	optional	required
color + alpha combined	required	N/A	N/A	optional	required	optional	required
color and alpha separated	required	required	optional	N/A	required	N/A	required

图 9: BC7 Block

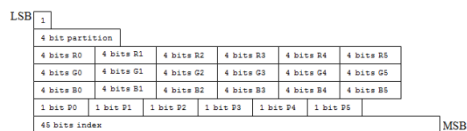
如图9所示，展现了不同情况下 128 位的 BC7 数据结构，分为以下三

(1)不包含 alpha 的 BC7 块,BC7 块由模式位、分区位、压缩的 endpoint、压缩索引和可选的 P 位组成,其中 endpoint 为 RGB 格式,对于源数据中的所有纹素, alpha 值均为 1.0;

(3) alpha 部分和 RGB 部分单独处理的 BC7 块, BC7 块由模式位、旋转位、压缩的 endpoint、压缩索引和可选的索引选择位组成, alpha 部分和 RGB 部分分别进行插值。

下面，简单介绍下 BC7 块的模式 0，其余见BC7 模式：

- 只有颜色分量（无 alpha）
- 每个块有 3 对 endpoint
- 每个 endpoint 的存储格式为 RGBP 4.4.4.1
- 3 位索引
- 4bit 的分区



8

3 参考

Compressed GPU texture formats

Unity 对各平台纹理格式的建议

块压缩 (Direct3D 10)

Direct3D 11 中的纹理块压缩