PlayStation显存字库扩容

时间  2011-01-28

标签 [游戏](http://www.voidcn.com/tag/%E6%B8%B8%E6%88%8F) [c](http://www.voidcn.com/tag/c)

原文   [http://blog.csdn.net/yusjoel/article/details/6167958](javascript:void())

       这个东西不是什么新鲜玩意儿了，Agemo很早就使用过，并且在他的主页上写过一个标题《字库扩容2 字库4色变2色，仅用于显存字库》，但状态一直是写作中。Koro也用这个方法扩容过字库，只是都没有具体的说明文章。

       这次手头碰到一个显存字库的游戏需要扩容，自己硬想总算是想出来了。这里写一下大约的思路。

       首先说一下PS的机能。显存字库一般是游戏在一开始loading的时候从光盘载入到显存中，载入之后就无法直接对显存进行操作了，比如解压之类的，以后就是使用原语指令将这个字库显示到主画面中。

       一般使用Sprite指令，如

sprite (77, 191)\*(10, 10) clut(272, 480), UV(160, 50) RGB(68, 68, 80)

其中UV是材质页地址，字库一般都是当成材质页来看待的。

这条指令的含义大略是从材质页中(160, 50)的位置取10 \* 10大小的区域，复制到(77, 191)这个位置。

材质页的定义如下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | | | | | | | TP | | ABR | | TY | TX | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| TX | 0x0-0xF X×64 材质页X坐标 |
| TY | 0       0     材质页Y坐标 1       256 |
| ABR | 0       0.5×B＋0.5×F 半透明模式 1       1.0×B＋1.0×F 2       1.0×B－1.0×F 3       1.0×B＋0.25×F |
| TP | 0       4位调色板 1       8位调色板 2       15位 |

可以看到材质页只能设置成4bpp，8bpp和16bpp（无调色板），从材质页复制数据到指定区域时，会自动将4bpp/8bpp根据调色板转成16bpp。

扩容的一般想法是将4bpp转成1bpp，那么通过降低质量来换空间，可问题是系统并没有指令可以将1bpp的图像转成16bpp，所以只能用另一种曲线的方式来实现。

主要的想法是1bpp的字库并不是连续存放，而是固定在4bits中的某1位上。

假如说原字库数据如下：

AB CD    (这是2bytes数据，共4个点，显示的时候从左到右也是按照ABCD的顺序)

A1 A2 A3 A4 B1 B2 B3B4C1 C2 C3 C4 D1 D2 D3 D4

现在要存放1bpp的字库，如果也是4个点，通常的做法就是存放在A1 A2 A3 A4，但现在是存放在A1 B1 C1 D1上，这有点“通道”的概念，不妨把1位称作1通道，同理，其他位分别是2-4通道。那么要如何显示1通道的字库呢，只要做一个调色板，将8-15号颜色设成黑色，0-7号颜色设成白色即可。其实就是序号为1xxx的设置成字体颜色，0xxx的设置成底色。可以想象，使用了这个调色板之后，就会无视2-4通道的里面的数据，只根据1通道显示。那么其他通道也类似，2通道将序号x1xx的设置为黑色，x0xx的设置为白色；3通道将xx1x的设置为黑色，xx0x的设置为白色。.

解释:

假如调色板中,字体颜色是蓝色,值为双字节aaaa,背景色为透明色,值为双字节0000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 通道 | 字体色 | 背景色 | 调色板 | 调色板值 |
| 1 | 1xxx | 0xxx | 0-7为蓝色,其它为透明色 | aa aa aa aa aa aa aa aa  00 00 00 00 00 00 00 00 |
| 2 | x1xx | x0xx | 4-7,12-15为蓝色, 其它为透明色 | 00 00 00 00 aa aa aa aa  00 00 00 00 aa aa aa aa |
| 3 | xx1x | xx0x | 2,3,6,7,10,11,14,15为蓝色, 其它为透明色 | 00 00 aa aa 00 00 aa aa  00 00 aa aa 00 00 aa aa |
| 4 | xxx1 | xxx0 | 奇数位为蓝色,其它为透明色 | 00 aa 00 aa 00 aa 00 aa  00 aa 00 aa 00 aa 00 aa |

这个方法不一定非要转成1bpp，如果码表空间允许还可以4bpp转成2bpp，方法也类似。甚至还可以字库4bpp/2bpp/1bpp混合，只要设置相应的调色板即可。当然，这样代码修改起来就复杂多了。

代码修改的话主要是修正CLUT\_ID（调色板ID），定义如下：

指定调色板数据的位置

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Y坐标0－511 | | | | | | | | | | X坐标X/16 | | | | | |
| F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

要注意《Everything》那篇文章中说CLUT\_ID的Y坐标占用的是8-F是错误的，这样也到不了511。应该是占用6-F，这样取值范围虽然到1023，但是有效值只有0-511。

调色板原来只有一个，现在变成了2个以上，如果没有什么地方放的话，就放在字库里面好了。CLUT在显存中是16\*1的一个长条，注意X坐标一定要是16的倍数。一般来说，4个CLUT的话，只要放一个64\*1的长条即可，基本都放的下。指定CLUT时，把通道号（基于0的）加到基准的CLUT\_ID上就行。(注:对每一个字符都要能指定clut,比较困难,需要ASM HACK)

下面这张图可以清晰地看到同一块显存，在CLUT不同的情况下，显示出了不同的字库。

使用agemo的vram viewer，模式9，4bpp+CLUT

