Displaying Sprites with a Sega Saturn and the Sega Game Library

Reinhart and others

2003

1 Changes

- 2003.09.17 : fixes (Vreuzon, Rockin'B, Antime);
- 2003.09.16 : some changes (Vreuzon);
- 2003.09.15 : document creation (Reinhart).

2 Introduction

Here is a little tutorial. It focus on displaying a 24 bits single sprite (more to come). Feedbacks, corrections, rewiews are welcome (my english may be bad).

To use this tutorial, you have to get a running dev environment. You can get all you need at antime's site [1]. You also have to know some C or C++ .

2.1 Colordepth and paletted sprites digression

The color table can either hold 1) *Color bank codes* (VDP2 palettes), allowing framebuffer rotations and scrolls; 2) *RGB data* (no palette). In this tutorial, we focus on 2) for simplicity, but you should be aware of its limitations and how to bypass them.

- Rockin'-B complains about paletted sprites :

  > For sprites it would be much better to use paletted images[2], because of memory usage. I remember Reinhards stating on his site that he tried to get more sprites managed. Paletted sprites is the solution. But no one did this. What I read is that the used pallette is the same that is registered inside VDP2. But there seems to be another way to place multiple palettes of 16 colors into VDP1 RAM.

- "Yes, it is", Antime answers :

  > In order to "place multiple palettes of 16 colors into VDP1 RAM", The "Color Mode" bits in the CMDPMOD command table field must be set to 001. The address of the color table divided by 8 is stored in the CMDCOLR field. Set the MSB (Most Significant Bit) of the color to 1 for it to be interpreted as RGB. If VDP1 is in 8bpp mode (framebuffer rotation or hires) you can only use color bank codes.
  >
  > Using the SGL, you can use paletted data by setting the right face/sprite attribute data: The *Mode* should be `CL16Look`, the lookup table address (divided by 8, I assume) should be given as *Color* and the *Option* should be `UsePalette`.

[1] Antime feeble saturn page : `http://www.helsinki.fi/~ammonton/sega/`

[2] 24 bit depth is not paletted: RVB values are stored

I think you have to manage the CLUTs (Color LookUp Table) yourself, couldn't see any references in the manual.

Also note that color number zero is used as transparent, unless the disable bit is set. If you enable end codes, you lose another color (number 15 with 4bpp data).

Let's go back to our 24 bits sprites for now.

3 Transform the sprite into a .C char

The sprite

1. Size – The width must be a multiple of eight pixels (max 504 pixels) while the height can be given in one-pixel precision (max 255 pixels). We will assume that our sprite is 16x16.
2. Color Depth – convert your sprite to 24bit colordepth and raw format. We will assume that our first sprite is named **ring1.raw**

Now use SSConv to create a `.C` file :

```
\code{ssconv ring1.raw ring1.c true c ring1}
```

The **ring1.c** file should look like that:

```
// Source File: ring1.raw
// sprite : 16x16

unsigned short ring1{[]{[]} = {
  0x0000, 0x0000, 0x0000, 0x0000,
  0x0000, 0x8210, 0x839C, 0x839C,
  (...),
  0x8108, 0x8108, 0x8108, 0x0000, }
  0x0000, 0x0000, 0x0000, 0x0000 };
}
unsigned long ring1Size = 256;
```

TIPS :

- In order to have a transparent background, replace all occurence of the first hexadecimal value by `0x0000`. This first value represent the 'transparent' color.
- If you don't like the command line, the conversion can be completed using `SegaConverter`, a windows GUI app, allowing you to cut and paste sprites from and to other apps. The bitmap depth has to be 32 bits. You will be asked the background (transparent) color when you save to a `.C` file. Rockin-B says : " *Don't give up if it says that the color format is not supported. Create a new file in Converter, select proper resolution and 32K colors. Then open the sprite as BMP file, select all, copy and paste into the new file. Deselect the pasted image and saving will be possible. Save as* `SGL .c`. *Finished!*"

4 Define sprite `attributes` and `data`, define `textures`

4.1 SGL structures

The SGL uses three straightforward structures (`PICTURE`,`TEXTURE` and `SPR_ATTR`), as well as associated creation crystal-clear macros (`PICDEF`, `TEXDEF` and `SPR_ATTRIBUTE`), defined in `sgl.h`):

```
#define SPR_ATTRIBUTE(t,c,g,a,d) \
  {t,(a)|(((d)>>24)&0xc0),c,g,(d)&0x0f3f}

typedef struct {
    Uint16       texno ;
    Uint16       atrb ;
    Uint16       colno ;
    Uint16       gstb ;
    Uint16       dir ;
} SPR_ATTR ;

#define TEXDEF(h,v,presize) \
  {h,v,(cgaddress+(((presize)*4)>>(pal)))/8,(((h)&0x1f8)<<5 | (v))}

typedef struct {
    Uint16       Hsize ;
    Uint16       Vsize ;
    Uint16       CGadr ;
    Uint16       HVsize ;
} TEXTURE ;

#define   PICDEF(texno,cmode,pcsrc) \
   {(Uint16)(texno),(Uint16)(cmode),(void *)(pcsrc)}

typedef struct {
    Uint16       texno ;
    Uint16       cmode ;
    void        *pcsrc ;
} PICTURE ;
```

Nobody will ask you to fully understand the lines above.

Anyway, you now have to create a new file that includes `<sgl.h>` as well as the C file we have created (the **ring1.c** file). This new file will be called **spr_data.c**.

```
#include "sgl.h"   //library needed
#include "ring1.c" //the sprite you want to display
```

4.1.1 Attributes

Add:

```
SPR_ATTR attr[] = {
  SPR_ATTRIBUTE(0,No_Palet,No_Gouraud,CL32KRGB|ECdis,sprNoflip),
};
```

The first parameter is important as it is a reference to the sprite. Our **ring1** is sprite n° 0. Keep parameter 2, 3 and 4 like that. 5th parameter allow to flip sprite(mirroring).

4.1.2 Size

Add:

```
TEXTURE tex_spr[] = {
  //(sprite_width, sprite_height, prev_sprite_width*prev_sprite_height)
  TEXDEF(16,16,0),
};
```

Position of the line TEXDEF(a,b,c) in the structure is important as the first `TEXDEF(a,b,c)` is related with `SPR_ATTRIBUTE(0,...)`.

4.1.3 Sprite attribute and sprite data connexion

```
PICTURE pic_spr[] = {
  //attribute n°0, number of color, which sprite to display
  PICDEF(0,COL_32K,ring1),
}
```

TODO : if COL_32K is the number of color, why is it already defined in the attribute ?

4.2 Optional definitions (usefull, but not related to the SGL)

4.2.1 Initial position of the sprite on the screen

Add:

```
FIXED stat[][XYZS] = {
  //Position of ring1 (X_pos, Y_pos, Z_pos(depth), scale)}
  {toFIXED(50),toFIXED(-130),toFIXED(169),toFIXED(1.0)}
};
```

4.2.2 Rotation angle

```
ANGLE angz[] = {
  DEGtoANG( 0),
};
```

Save this file as **spr_data.c**

5 Load and display the sprite

Now, the **main.c** file. Add:

```
/*---------------------------------------------------------------------*/
/*      Sprite tutorial 1 */
/*---------------------------------------------------------------------*/
#include      "sgl.h"

extern TEXTURE tex_spr[];
extern PICTURE pic_spr[];
extern FIXED stat[][XYZS];
extern SPR_ATTR attr[];
extern ANGLE angz[];

/*---------------------------------------------------------------------*/
/*      Set_sprite : register all sprite define in Spr_data.c */
/*---------------------------------------------------------------------*/
static void set_sprite(PICTURE *pcptr , Uint32 NbPicture) {
    TEXTURE *txptr;

     for(; NbPicture-- > 0; pcptr++){
         txptr = tex_spr + pcptr->texno;
```

```
        slDMACopy((void *)pcptr->pcsrc,
                (void *)(SpriteVRAM + ((txptr->CGadr) << 3)),
                (Uint32)((txptr->Hsize * txptr->Vsize * 4) >> (pcptr->cmode)));
    }
}

/*---------------------------------------------------------------------*/
/*      disp_sprite : display sprite selected */
/*---------------------------------------------------------------------*/
static void disp_sprite(int SpriteNo) {
     slDispSprite((FIXED *)stat[SpriteNo],
                  (SPR_ATTR *)(&attr[SpriteNo].texno), DEGtoANG(0));
}

/*---------------------------------------------------------------------*/
void ss_main(void)
{
    slInitSystem(TV_320x224,tex_spr,1);

    //register the sprite, here we only have 1 sprite
    set_sprite(pic_spr,1);

    while(1) {
         //display our sprite (note the first sprite is n\x{00B0} 0)
         disp_sprite(0);

         slSynch();
    }
}
```

Add the **spr_data.c** in your **OBJECTS** files (containing the objects to be linked). Compile these files and your Saturn will display a sprite.