Displaying Sprites with the Sega Game Library

Reinhart and others

2003

1 Changes

- 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++ .

3 Transform the sprite into a .C char

The sprite

1. Size – it appears that everything works better if sprite size is multiple of 8 (16x16, 32x32, ...). 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 :

```
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,
```

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

```
  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 24 bits too. You will be asked the background (transparent) color when you save to a `.C` file.

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();
    }
}
```

Compile these files and your Saturn will display a sprite.