Sprite in Star Trek: Armada

Many aspects of the look and feel of *Star Trek: Armada* are created by so-called *sprites*. Sprites essentially consist of two aspects:

1. A graphics file, containing some sort of bitmap graphic, that is used entirely or in part, the texture.
2. A text file referencing this graphics file and defining how to use its contents for the sprite.

Sprite Files

The text files of sprites are located in the folder with the same name in the Armada installation. They end with the extension *.spr*, e.g. *weapon.spr*. These are the so-called sprite files. There is a number of different types of sprites, defined in their corresponding sprite files.

- *gui_global.spr*: Used for the looks of buttons and wire frames.
- *tex_anim.spr*: Defines the way weapon sprites are displayed and what kind of animations they represent (texture animations or moving textures).
- *weapon.spr*: Used for optical weapon effects.

Lines in sprite files consist basically of two types of definitions.

1. The definition of a section handling a certain type of sprite
2. The definition of specific sprites within these sections.

Sprite Sections/Block Definitions

Block definitions are kind of an introductory header, marking which kind of content is to follow, until another such definition occurs. They at least consist of a line of the form `@reference=`<*number*>. <*number*> is a number of 2 to some exponent, e.g. 64, 128 of 256. This defines the kind of sprite sizes to be expected in this block. E.g. `@reference=256` means, the graphics file is 256 pixels x 256 pixels in size. The graphics are always square shaped.

You may also use `@tmaterial=additive` as an additional information, making the sprites in this section be displayed using *additive blending*. This means, the colors of background and sprite are added, basically making the sprite semi-transparent. Usually this is what is wanted, making a black background of the texture used for the sprite not have any effect on the way it is displayed. (Otherwise it would appear as a black rectangle with some content inside it.) So for a weapons animation this is very useful, whereas for a button this is not a desired effect.

Any new use of *@reference=* opens up a new section in the file. This has the side-effect, that they don't need to be in any particular order and even sections of the same definition may occur in the same file.

Sprite Definitions

A single sprite definition defines *one* sprite, but uses aspects of the block definition that came before it. It may look like this, for example:

```
# Accelerator Cannon
cAccelCannon    wAccelCannon     0    0    256    64    @anim=tex1x4
```

While the first line is a simple comment (comments start if a # sign, and end and the end of the line), the second line is the actual sprite definition. The name of this particular sprite is *cAccelCannon*. This is what the game uses to refer to this specific sprite wherever it is needed in the game. The definition references the file *wAccelCannon.tga*. As you can see, the *.tga* extension is omitted. The next four values are the Sprite Coordinates. The sprite definition is finished with *@anim=tex1x4*, the animation type for this sprite. It tells the game that this is a 1 x 4 animation, meaning that it is 1 frame wide and 4 frames high. This already hints at the concept of animation. An animated texture sprite consists of one texture, that is interpreted in sections that are to be used as frames, just like in a video animation. So the game will display the single frames one after the other, depending on the situation looping over them multiple times.

The meta-format of sprite definition lines is the following:

<Sprite Name> <Graphics File without .tga> <Starting X> <Starting Y> <Width> <Height> <optional Animation>

The sprite name **must** be unique, while the same TGA file might be used by different sprites. However, the texture file for a specific sprite must fit its block definition (e.g. *@reference=64* requires the TGA files used by sprites in that section to be 64 pixels x 64 pixels big). *<Starting X>* to *<Height>* are the sprite coordinates. The *<optional Animation>* is the definition for the Sprite Animations.

Sprite Coordinates

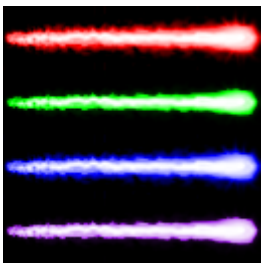
In the previous section the parts *Starting X*, *Starting Y*, *Width* and *Height* were not explained very detailed. They are the actual sprite coordinates. As was stated, an animated sprite may consist of multiple frames. Each frame is taken from the same animation, meaning, one texture contains all frames of the animation. TGA does not support animations, such as GIF does. Instead of that, specific sections of the given graphic are interpreted as being one single frame of the entire animation.

In order to address the sections properly, the game needs to have some information about where to find the frames within the picture. The *<Starting X>* value defines the coordinate from left to right, starting with 0. The *<Starting Y>* value defines the coordinate from top to bottom, starting with 0. In the given example above, the starting coordinates are (0/0), the upper left corner. This is the beginning of the area intended for the sprite. The *<Width>* and *<Height>* correspondingly mean the x-size and the y-size of that area. In the above example the end-coordinates are (256/64), as implied by the size of one frame being 256 pixels wide and 64 high.

This may allow you to define multiple different sprites by use of the very same TGA file, e.g different phaser beams in the same file. Simply use different sections of the same file, only taking a part by

means of the described partitioning definition. As an example the following graphics may be used under a section *@reference=128*. To pick the third element of it (the blue beam) the corresponding sprite definition would look like this:

```
@reference=128
@tmaterial=additive
superpul ppulse 64 128 128 32
```



It could be addressed by its name *superpul*.

But this method of addressing sections of the graphic also allows to partition a texture for use in animations.

Sprite Animations

The given section (0/0) to (256/64) of the described sprite defines only one frame, while the initial animation stated something along the lines of being it a 1 x 4 animation. 1 column (x-direction) and 4 rows (y-direction). This means, there are to be 4 frames found in this TGA file, each one stacked below its predecessor. Meaning, the definition of (0/0) to (256/64) is only the starting frame, with three more coming below it, having the same dimensions. (This also shows, why sprite coordinates are not starting-point + end-point but starting-point + dimensions. The dimensions are re-used for animations and locating the other animation frames.)

So all frames together are 256 pixels wide and 256 pixels high. Of course the TGA file has to have at least this size. While the example uses a 1 x 4 animation, others are also possible, like a 4 x 4 animation (like the ones used for certain photon torpedoes). For a 256 x 256 TGA file that would give each frame a 64 x 64 size, but 16 of them. They are displayed in the order left to right and then top to bottom (just when reading a book).

[**Modding**] [Tools] [ODF Files] [ODF Directives] [Class Labels] [Tech Tree Files] [SOD Files] [Buttons] [Wire Frames] [Sprites] [AI Scripts] [Model Hierarchy] [Node Names] [Emitter Names] [Texture Animation Names] [Sprite Names]

[Back to Modding]