SMSQESprites

SMSQESprites is a program to convert (some types of) PC images to SMSQ/E (or traditional QL) sprites, and vice-versa, including some small image modifications. The basic mode of operation is that you load either an existing sprite or a PC image, possibly apply some small modification to it and then save it either as a sprite or a PC image (yes, you can load a sprite and re-save it as a sprite).

This program presumes some familiarity with sprites, colour patterns, and masks. There is a succinct explanation of these concepts at the end of this document.

Table of Contents

1 Sprites and images handled

SMSQESprites can handle a variety of sprites and PC images:

1.1 Sprites, images and “pictures”

To avoid confusing vocabulary, in this text:

- “sprites” are traditional QL, or enhanced GD2, sprites;
- “images” are normal PC style images, bmp, or jpeg etc.;
- “pictures” are what SMSQESprites displays in the picture window, which technically can be either a sprite or an image. It is used to avoid having to write “sprite or image” whenever something applies to both.

1.2 Sprites

SMSQESprites should be able to handle any valid traditional QL, or enhanced GD2 sprite, of the following modes:

- QL colour mode 4
- GD2 sprite modes 16 and up.

Any handled sprite should be validly shown in the picture window as soon as it is loaded. There are some restrictions, though:

Chained sprites, multiple sprites and sprites with an options block are handled in such a way that only the very first sprite is shown.

System sprites are not handled at all, neither are QL colour mode 8 sprites.

Sprites that use an EOR mask are displayed in full. For sprites with these masks, SMSQ/E (and the normal pointer environment) will EOR the sprite with the background. This means that the final result shown on the screen will depend on the background. SMSQESprites will show these sprites as if they had a full mask.

1.3 Images

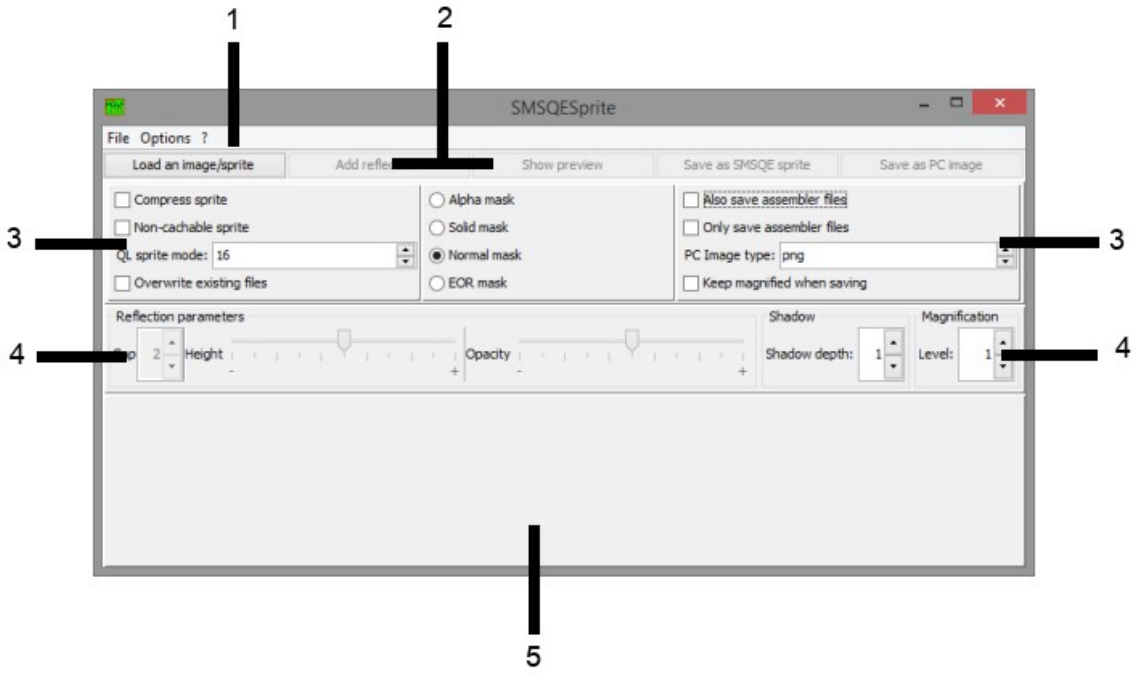
SMSQESprites should be able to handle many valid PC images and show them correctly on the screen. The image formats that it can handle are : bmp, gif, tif(f), png, jp(e)g.

1.4 Alpha masks

When loading sprites or images that have alpha channels, these are kept.

2 Using the program

When you start up the program, it will open its main window which looks something like this, with 5 distinct parts:



1. Menu bar, with some options/commands.
2. Fast action buttons, for fast loading/saving of pictures.
3. Option panels, allowing you to set options for the picture to save.
4. Picture manipulation panel, permitting some very limited image processing.
5. Picture window, showing the currently loaded sprite/image as a “picture”. Use CTRL + mouse wheel to magnify the picture.

These are explained in more detail below. You can generally hover the mouse over a feature and a tooltip will give some indication of what it will do.

3 The Menu bar

This is a typical menu bar, it has a “Files” menu, an “Options” menu and a “?” (About….) menu.

3.1 The Files menu

In the “Files” menu, you can save and load images or sprites, and also start off processing a whole directory.

3.1.1 Process entire directory

As the title suggests, this allows you to process a whole directory. Since this may be a somewhat complicated process, is is treated in more detail below.

3.1.2 Load a picture

Here you may load a sprite or an image. Select the corresponding file and it will be loaded and displayed. Once a picture is loaded, other options (such as saving) become available and the program title changes to reflect the name of the picture.

3.1.3 Save as SMSQ/E sprite

You will be requested for a filename to save to, and the picture will be saved as a sprite, with all the options currently set and obeying the rules of overwriting files as set out here. However, the options to save (additionally or exclusively) as assembler files are not honoured here, the file will ONLY be saved as a sprite, NOT as an assembler file

3.1.4 Save as PC image

You will be requested for a filename to save to, and the picture will be saved as a sprite, with all the options currently set and obeying the rules of overwriting files as set out here. However, the options to save (additionally or exclusively) as assembler files are not honoured here, the file will ONLY be saved as an image, NOT as an assembler file

3.1.5 Save as assembler file

You will be requested for a filename to save to, and the picture will be saved only as a sprite assembler file.

3.2 The Options menu

In the Options menu you can set some options.

3.2.1 Language

Here you can choose the language used by SMSQESprites. There are 4 languages to choose from.

3.2.2 Appearance

You may choose between several looks and feels for the window. On some systems, you might be able to choose a theme called “Nimbus Dark”, which is the same as “Nimbus”, but with darker colours, more suitable for night-time.

3.2.3 Options when loading a new picture

When loading a (new) picture, SMSQESprites will normally reset all the options (shadow, magnification etc.) to the default configuration. Here you can change that momentarily when loading a sprite, without affecting the general configuration.

The loading options (a new menu that opens up) allow you to keep any or all of these options as they are at the time you load the picture.

3.2.4 Save current configuration as default

Once you have configured SMSQESprites to your liking, you can save the current configuration. This will be the new default configuration whenever the program starts up. The configuration information is stored in a file called “SMSQESprites.ini” in your home directory and is created and maintained by the program automatically.

3.3 The “?” menu

This lets you check the version of SMSQESprites.

4 The “fast action” buttons

These allow some fast access to several features, like loading and saving sprites or images. Note that all of these buttons, except for the “load sprite/image” and “reflection” button are unavailable at first – they become available as soon as a picture is loaded.

4.1 Load a picture

This has the same action as that of the corresponding “Files” menu item.

4.2 Add reflection

Lets you set the reflection parameters for the picture. See below for more detail.

4.3 Show preview

After loading a picture, you can have a preview of what a QL sprite would look like in the sprite mode you have chosen. The preview sprite is shown, just like the loaded picture, in the Picture window and to the right of the original picture. Changing the mode in the general options panel will immediately be reflected in the preview. Remember that traditional QL sprites cannot have a reflection.

4.4 Save as SMSQ/E sprite

The picture will be saved as a sprite with the name of the file under which it was loaded, obeying the rules of overwriting files as set out here, and with all the options currently set. This includes possibly saving the corresponding assembler file as well. If you checked the “Only save assembler files” option, the file will ONLY be saved as an assembler file, NOT as a sprite.

4.5 Save as PC image

You will be requested for a filename to save to, and the picture will be saved as a PC image, of the type determined in the option panels, obeying the rules of overwriting files as set out here and with all the options currently set. This includes possibly saving the corresponding assembler file as well. If you checked the “Only save assembler files” option, the file will ONLY be saved as an assembler file, NOT as an image.

5 Option panels

The options panels allow you to set several options concerning the sprite. There are three panels, grouping several options (somewhat haphazardly). From left to right they are: the general options panel, the mask panel and the assembler options panel.

5.1 The general options panel

This panel contains 4 options:

5.1.1 Compression

Sprite data (mask and colour pattern) may be compressed. If this item is selected, they will be. In theory, the mask and the colour pattern may be compressed independently of each other. SMSQESprites does not do that – if you tick this box, both the mask and the colour pattern will be compressed.

Please note, however, that on some occasions, SMSQESprites may refuse to compress a mask and/or colour pattern, even though this item is selected: SMSQESprites compares the compressed and uncompressed sizes of the mask and the colour pattern, and if an uncompressed size is smaller, it uses that.

5.1.2 Non-cacheable sprites

SMSQ/E provides an automatic mechanism whereby sprites are kept in a cache for a certain time. This makes sense notably when a sprite to be displayed is, originally, not of the colour mode of the screen. SMSQ/E converts such a sprite into one that displays correctly in the current screen mode. As it would take much time to do this conversion every time the sprite is redrawn, the converted sprite is kept in the cache for some time. You may, however, set a special parameter which means that a sprite will not be cached at all. This is what this item does, if selected.

5.1.3 Sprite mode

Here you can select the mode which you wish the sprite to have once you save it. Mode 4 is the traditional QL sprite, mode 4. Modes 16, 32, 33 and 64 are the corresponding GD2 sprite modes. SMSQESprites doesn't handle traditional QL sprites in QL colour mode 8. This might be forthcoming in a future version should the need arise (i.e. if somebody asks for it…).

5.1.4 Overwrite existing

If this option is selected, existing files with the same name are simply overwritten. This may have unintended consequences when treating whole directory(ies), see below.

If this option is left unticked and there already is an existing file with the same name, the program will add a number after the name of the file to be saved: So a file called “example.spr”, if re-saved as a sprite, would be saved as “example1.spr”. If a file with this new name also pre-existed, it would be saved as “example2.spr”, or “example3.spr” etc... Please note that the names of sprite files and image files are independent: Suppose you have a file called “example.spr” as in the example above. Saving it as a sprite will result in a file called “example1.spr”, as per the example above. Saving it then as PC image file, would result in a file called “example.png” (unless a file called “example.png” already existed).

Note : if the extension delimiters differ there will not be any problem. Suppose you have “example_spr” and “example.png”. Conversion will be “example_spr” → “example_png” and “example.png“ → example.spr”. There is no filename overlap.

5.2 The mask panel

You can select which of the 4 mask types the sprite is to have. If you do not know what masks, or these types, are, there is a small explanation below.

5.2.1 Masks and reflection

When you use a reflection, the mask for the sprite will ALWAYS be an alpha mask, even if you chose another one. This is why the mask choices become unavailable when you choose to add a reflection. Moreover, you cannot add a reflection to a traditional QL colour mode 4 sprite.

5.2.2 Masks and sprite mode 4

For compatibility reasons, sprites in traditional QL colour mode 4 will not be allowed to have an alpha mask. If you have a picture displaying with an alpha mask and set the sprite mode to 4 then the alpha mask will be changed automatically to a solid mask. **It may not be changed back automatically if you re-select another sprite mode.**

5.3 The assembler panel

This also contains 4 options (only the first two of which actually concern assembler files).

5.3.1 Also save assembler files

If this is selected, it allows you to save an assembler file each time you save the picture, either as a sprite or a PC image. The assembler files have the same name as the picture file, but with the extension “asm”. The extension separator (“.” or “_”) will be kept. **Existing assembler files of the same name are always overwritten**. The sprite name in the assembler file will be the name of the file into which it is saved, stripped of the directory part of the filename. So, if you save a file into
`/myuser/mydirectory/mysprite.png`,
the corresponding assembler file will be
`/myuser/mydirectory/mysprite.asm`
and the sprite will be called `mysprite`.

5.3.2 Only save assembler files

If you select this, only assembler files files will be saved when you save the image as a sprite or PC image. The sprite or image will not be saved.

5.3.3 PC Image type

This determines how an image will be saved back as a PC image. You have the choice between jpg, png and bmp. The corresponding extension will be added to the filename, after removing a possible existing extension.

5.3.4 Keep magnification when saving

If you select this, the picture is saved (whether as PC image or as sprite) with the magnification at which it is currently shown in the preview window (see Magnification, below). If you leave this unchecked, the image is saved at its original size.

6 The picture manipulation panel

This panel allows some very limited image processing. You may

- add a reflection to the picture;
- add a shadow to the picture;
- magnify the displayed picture.

6.1 Reflection

The reflection is a fading mirror image set beneath the original picture. You may set:

- The gap, in pixels, between the original picture and the reflection.
- The reflection height, which is a number between 0 and 100. This, roughly is the percentage of the original image that will be shown in the reflection. If this is set to 0, there is no reflection, if it is set to 100, the reflection will be about as high as the original picture. Remember, though, that the reflection is always fading.
- The opacity of the first line of the reflection, a number between 0 and 100. The higher the number, the more of the reflection will be seen, as the fading uses the first line as a baseline.

Note that the reflection makes use of “alpha masks” to achieve the fading effect. **So a reflected image will always be saved with an alpha mask**.
As the traditional QL sprites do not have alpha masks**, there is no reflection possible for this type of sprite**.

6.2 Shadow

Not implemented yet.

6.3 Magnification

As the name implies, this makes the picture appear bigger in the picture window. This is obtained by simply multiplying the pixels in every dimension by the magnification factor. A factor of one means that there is no magnification, a factor of two means that each pixel is doubled in both dimensions, a factor of three means that each pixel is tripled in both dimensions, etc.

You can set the magnification factor either by clicking in the “Factor” field, or more simply by turning the mouse scroll wheel with the CTRL key kept pressed.

When saving a picture SMSQESprites will normally save the picture at the original size, not the magnified one. You can change that by using the “Keep magnification when saving” item.

7 Processing an entire directory

You may choose to treat a whole source directory from the “Files” menu in the Menu bar. Only one directory may be treated with this option, if you want to process several source directories at once, you must use the command line.

The program will either convert all sprites in the directory to PC images, or all PC images to sprites - or make both conversions at the same time.

To process an entire directory, proceed as follows:

First, in the main window, set the parameters to determine

- whether sprites saved should be compressed;
- whether sprites saved should never be cached;
- the sprite mode all saved sprites should have;
- whether existing files should be overwritten;
- the sprite mask;
- whether assembler files should also, or exclusively, be saved;
- the PC image type all saved images should have;
- the height, gap and opacity of the reflection, if any (height = 0 means no reflection);
- whether a shadow should be added to sprites;
- the possible magnification and whether the magnification should be kept

Once these choices are made in the main window, click on the “Files” menu and then the item to “process an entire directory”. A new window opens, the “process directory” window. There you may set:

- The source directory. This will be the directory from which all files to be converted will be taken. If you leave this empty, nothing will be converted. There can only be one source directory. It is possible to treat several source directories in one go, but only via the command line (see below).

  You can also set whether you want to process any sub-directory of your source directory. If this is selected, any sub-directory (an its sub-directories et) will also be scanned for files to convert.

- An optional destination directory. All converted files will be put there. If you leave this empty, the converted files will be put into the source directory. Please note that if you decide to process also sub-directories, the destination directory should not be in any sub-(sub- et)directory of the source directory.

- What is to be converted. You can select whether you want to convert PC images to QL sprites and/or QL sprites to PC images. If you select both, each will be converted to the other format.

After you have filled in what needs to be done, hit the "Go!" button, and the conversion will start. You may also leave this window at any time with ESC before hitting the "Go!" button, no conversion will then take place.

When converting, SMSQESprites expects sprites to have the extension '.spr' or '_spr' and PC image files one of ".gif", ".png", ".bmp", ".jpg", ".jpeg", ".tif" or ".tiff". The case of the extension does not matter.

To save sprites into the other format after conversion, SMSQESprites uses the same name as the original file but changes the extension. Assembler files also get the same name but with the extension "asm". The extension delimiter ("_" or ".") is kept.

So, supposing you choose to save PC files as png:
example1_spr will be converted to example1_png and the assembler file will be example1_asm.
example2.spr will be converted to example2.png and the assembler file will be example2.asm.
example3.png will be converted to example3.spr and the assembler file will be example2.asm.
The name of the sprite within the assembler file will be sprite_xxxx, where xxxx is an increasing number.

Note that SMSQESprites does not try to reconvert a newly converted image or sprite. If your source directory contains a file called "qlsprite.spr", it will be converted into qlsprite.png – but the program does not, then, try to reconvert "qlsprite.png" into "qlsprite.spr". The only case in which that could happen would be if your destination directory lies in a sub-directory of the source directory and you wished that SMSQESprites also processed sub-directories.

ATTENTION: SMSQESprites abides by the "overwrite existing files" directive as set in the main window and explained here. Likewise for assembler files. **However**: the order in which the assembler files will be saved is not guaranteed. Remember that assembler files will always overwrite any other file with the same name.

The time taken to process an image is with relative to the number of image modifications that must be made : applying a reflection, magnification, or shadow, takes time.
For example, if you wanted to have only assembler files, only for all QL sprites, you would check "convert QL sprites" and "save as assembler only". What will happen then is that this software:

- reads in a sprite,
- checks whether any modification must be made
  - if so, it converts the sprite to an RGB image,
  - possibly applies magnification/reflection/shadow to this image,
  - converts this image back to a sprite
- and then saves that as assembler.

8 Command line processing

You may pass a command line to the program. The command line consists of one or several switches, which are always preceded by a hyphen (“-”) and separated from the next switch by a space. Some switches are followed by parameters, others are not. If you do not set a switch, the corresponding default value will be used. NOTE : parameters MUST NOT contain a hyphen (“-”) or minus sign.

The command line switches are set out in the table below. You must set at least the -p or -q switch and either the -i or -s switch, else nothing is converted. Attention : the default values for command line switches are determined as follows:

8.1 Default values

Each command line switch has a default value – if the switch isn’t present in the command line, the default value is used. Normally, the default value to be used will be taken from the configuration file (see “Save current configuration as default”). However, sometimes you might not want to use this configuration file. If that is the case, use the command line switch “-n” (which MUST then be the first switch on the command line). Then and only then, the defaults used are then those as set out in the table.

8.2 Command line switches

	Effect	Default *
-a	Also save assembler files, in addition to images or sprites. Note that -a and -j are contradictory - the last one passed will be taken into account.	Don’t save assembler files
-b	Use a normal Bitmap mask. Note that this conflicts with the ‘k’ and ‘e’ options. In case of conflict, the last option set will be the one used.	Use alpha mask
-c	Continue if recoverable errors. If SMSQESprites encounters a problem in the command line, it will normally not process anything. If the error is recoverable, it may be possible to continue.	Stop on any error
-d	Destination directory of converted files. If this is absent, the source directory will be used (attention files with the same name will be overwritten). If by error you use this option several times in the command line, only the last one will be used. See note (1) below!	(nothing)
-e	EOR : use Eor mask for sprites. Note that this conflicts with the ‘b’ and ‘k’ options. In case of conflict, the last option set will be the one used.	Use alpha mask
-f	Forced extension: if this is present and followed by one of the following list : “bmp, png, tif, tiff, jpg, jpeg, gif”, only PC images with this extension will be converted. Note that you may give this parameter several times, all extensions (from that list) thus set will be taken into account. This has no influence on the extension necessary for sprite (“spr”).	No forced extension (all image files are treated)
-g	Gap between reflection and source image. Is only taken into account if the -h switch is also present (the order in which they are used doesn’t matter). Negative numbers are ignored.	2
-h	Height of reflection. This switches reflection on or off: if -h is absent or 0, no reflection will be added. Must be between 0 (=off) and 100 and is a (rough) percentage of the height of the reflection with respect to the height of the original image. Any exceeding value will be set to the nearest limit.	0 (=no reflection)

-i	Input file: name of the input file. This is ignored if used if the -s switch is also used. If no destination directory is set, the destination directory is that of the input file. See note (1) below!	None
-j	Just save assembler: only save assembler files, not images or sprites. Note that -a and -j are contradictory - the last one passed will be taken into account.	Don't save only assembler files
-k	use solid masK for the sprite. Note that this conflicts with the 'b' and 'e' options. In case of conflict, the last option set will be the one used.	Use alpha mask
-l	Larger image: magnify the picture. This is followed by a number between 1 (no magnification) and 100 (maximum magnification). The image or sprite is saved with this enlargement.	1 (=no magnification
-m	Mode of resulting QL sprite: 4, 16, 32, 33 or 64.	32
-n	No config file: do not use the program's automatically generated configuration file. If present, this command line switch MUST be the very first switch present in the command line, else it will simply be ignored!	Use config file
-o	Opacity of reflection. Is only taken into account if the -h switch is also present (the order in which they are used doesn't matter). Must be between 0 (reflection is practically entirely transparent) and 10 (reflection is opaque). Any exceeding value will be set to the nearest limit.	5
-p	PC images found should be converted to QL sprites.	No
-pp	PC images found should be converted to PC images (!). **See note (2) below.**	No
-q	QL sprites found should be converted to PC images (NB: SMSQESprites will only "find" sprite files if they have the extension ".spr" or "_spr").	No
-qq	QL sprites found should be converted to QL sprites (NB: SMSQESprites will only "find" sprite files if they have the extension ".spr" or "_spr"). **See note (2) below.**	
-r	RLE encoding is switched off.	ON
-s	Source directory of files to convert. You may give this parameter several times, then all of theses source directories will be processed. See note (1) below!	(nothing)
-t	Treat sub-directories: also process files in any (sub-) sub-directory of every source directory.	Don't treat sub-directories.
-u	Uncachable sprite : the cache for this sprite is switched off.	Switched ON
-v	Verbose: print out info on what is converted.	Not verbose
-w	Write over existing files. As this will replace existing file, this is dangerous. See note (3) below.	Don't overwrite
-x	Extension of saved PC images, this determines format of saved PC images. May be bmp, jp(e)g or png. If this parameter is used several times, the last one is taken into account. See note (1) below!	png
-y	Colour for pixels that become opaque when a normal mask becomes a solid one. Choose one of four values : 0 = black, 2=red, 4=green, 7= white.	0
-z	Shadow depth, followed by a number between 0 and 10 (0 = no shadow) **NOT YET IMPLEMENTED**	0

* Note that these defaults will only be used if either no configuration file exists, or if you used the "-n" switch.

(1) **REMEMBER** : extension, file or directory names MUST NOT contain a hyphen ("-").
(2) Normally, SMSQESprites will not convert a picture of a certain type into the same type of picture: a sprite will not be loaded and then converted into the same sprite. However, that's exactly what the -pp and -qq switches do – you might, under some circumstances want to save a loaded picture into the same type of picture, after treatment. You might, for example, convert all normal sprite masks into solid masks, so that the sprites takes less space, or all mode 64 sprites into mode 32 sprites etc. If you set the overwrite switch, the original sprites/images will then be replaced.

(3) When saving pictures as PC images, SMSQESprites ALWAYS uses the extension you have set for saved PC images (option -x, default = “png”). So, an image called “example.bmp” will be saved as “example.png” and thus not overwrite the original bmp file. However, any existing “example.png” file will be overwritten.

8.3 Command line error treatment

Some errors may be ignored, if the -c switch is set.

Unless otherwise stated,

- if a same command line switch is given several times, only the last one will be taken into account. This is not considered to be an error;
- if contradictory command line switches are used, only the last one will be taken into account. This is not considered to be an error.

8.3.1 Recoverable errors are:

-d with no destination directory. In this case, either the first source directory, or the directory of the input file, will be used.
-f with no, or wrong, extension. Ignored.
-g with wrong parameter. Default (or previously set) is used.
-h with wrong parameter. Default (or previously set) is used.
-i is set with no parameter.
-i and -s are both used at the same time. The -i parameter is then ignored.
-l with a higher value than 100. Will be reduced to 100.
-o with wrong parameter. Default (or previously set) is used.
-x with wrong type of parameter. Default (or previously set) is used.

8.3.2 Irrecoverable errors are:

- Neither -p nor -q were set – there is nothing to do.
- There is no source directory and no input file. Nothing is to be done.
- The destination directory given is not a directory.
- There is no destination directory. No destination directory was set and none could be determined. SMSQESprites tries to determine the destination directory by, first, using the one set with the ‘-d’ switch. If that wasn’t set, it tries to use the first source directory. If that doesn’t exist, the directory of the input file is used. If that doesn’t exist either, the error would already have been caught (see just above).
- A wrong mode value is given for QL sprites.

9 Sprites, patterns and masks

QL sprites have a “pattern” and a “mask”. Each of them describes one aspect of each pixel in the sprite. They may also be compressed.

9.1 Pattern

The **pattern** describes the colour of the pixels in the sprite. Depending on the colour mode, each pixel will take some part of a byte, or one or several bytes. Thus, there is one element in the pattern for each pixel in the sprite. Patterns must always be padded to the nearest long word, and the padding byte(s) used MUST be 0.

9.2 Mask

The **mask** described whether (and if so, how) any pixel is actually displayed.

9.2.1 Traditional bitmap mask: normal mask

At its most basic and traditional level (true for the original QL colour sprites), the mask is a bitmap, each element in the mask is either -1 or 0 and thus either lets the corresponding pixel of the sprite be displayed on the screen (the sprite is opaque at that pixel), or it lets the existing background colour shine through (the sprite is translucent at that pixel).

Masks use the same size element for a pixel as the pattern: If the sprite is a mode 32 sprite, where each pixel is denoted by a word (two bytes), then the mask also uses a word for the corresponding element (there is an exception: see “alpha channel” below). Thus, traditional masks are also padded to the nearest long word.

9.2.2 Special masks

There are three types of special mask:

9.2.2.1 EOR mask

This is like a normal mask, but each and every byte in the mask is set to 0. In this case, the sprite will still be displayed, but it will be EORed into the screen. Thus, the sprite colour will change according to the background. This is the case, for example, for the standard F1-F10 system sprites.

9.2.2.2 Solid mask

There may also be a “**solid**” mask. This is indicated by the fact that, in the sprite header, the pointer to the mask (NOT the content of the mask itself) is 0. It simply means that every pixel of the sprite is opaque and thus must be displayed. The effect is the same as if the sprite had a normal mask and each pixel in that normal mask were set to opaque. Solid masks exist because they take no data space at all in the sprite definition, so the sprite uses less memory. If you have a sprite where every pixel is always shown, this is the way to go.

Note: there seems to be a slight niggle in current versions of SMSQ/E when displaying sprites with

a solid mask in mode 4, or mode 4 sprites with a solid mask in mode 16 or 32 (8/16 bit modes). SMSQ/E will then sometimes display black pixels where there shouldn’t be – this happens when your sprite width is not a multiple of 16.

9.2.2.3 Alpha mask

Last but not least, more modern versions of SMSQ/E have introduced an “**alpha**” mask, available only in higher colour modes (i.e. not in traditional QL 4 or 8 colour modes). Here, each element of the mask determines, for each pixel, not -whether- but -how much- of the background pixel shines through (semi translucency). Each pixel in the sprite takes exactly one byte in an alpha mask. Alpha masks are not padded.

When the “`pto.alph`” flag is set in the sprite control byte, the mask is considered to be an alpha channel. An alpha channel allows gradual mixes between the background and the sprite pattern. Every pixel is represented by exactly one byte which can have all values from 0 to 255. 0 means the pixel is completely transparent, 255 means the pixel is completely opaque. Values in between determine the degree of mixing of background and foreground. Alpha channel information is not padded at the end of each line. There's one byte for every pixel and nothing more. Alpha channels make no sense on traditional QL style sprites and need higher colours.

9.3 RLE compression

Both pattern and mask/alpha channel can be compressed using a simple RLE (run length encoding) algorithm. This is useful with data that is largely homogeneous, which is often the case with masks. Compressed data must be signalled in the sprite control byte (pto.pcmp, pto.mcmp) and starts with the bytes 'RLEx', with 'x' being either 1, 2 or 4. This is the item size (number of bytes) the algorithm is working with. 8 bit RLE compression of 32 bit data wouldn't yield in good results, therefore the algorithm can also work on 16 bit or 32 bit data. After the ID there's one long word containing the size of the data in uncompressed form. After that the compressed data itself is following.

The compressed data always consists of one byte and one or more items. If the leading byte is in the range of 0<=x<128 then x+1 uncompressed items are following. Otherwise only one item is following, which represents 257-x times the same item in the uncompressed data.

10 Copyright and thanks

I would like to thank Giorgio Garabello for beta testing and his bug-hunting.