Unleashing Aurora GT

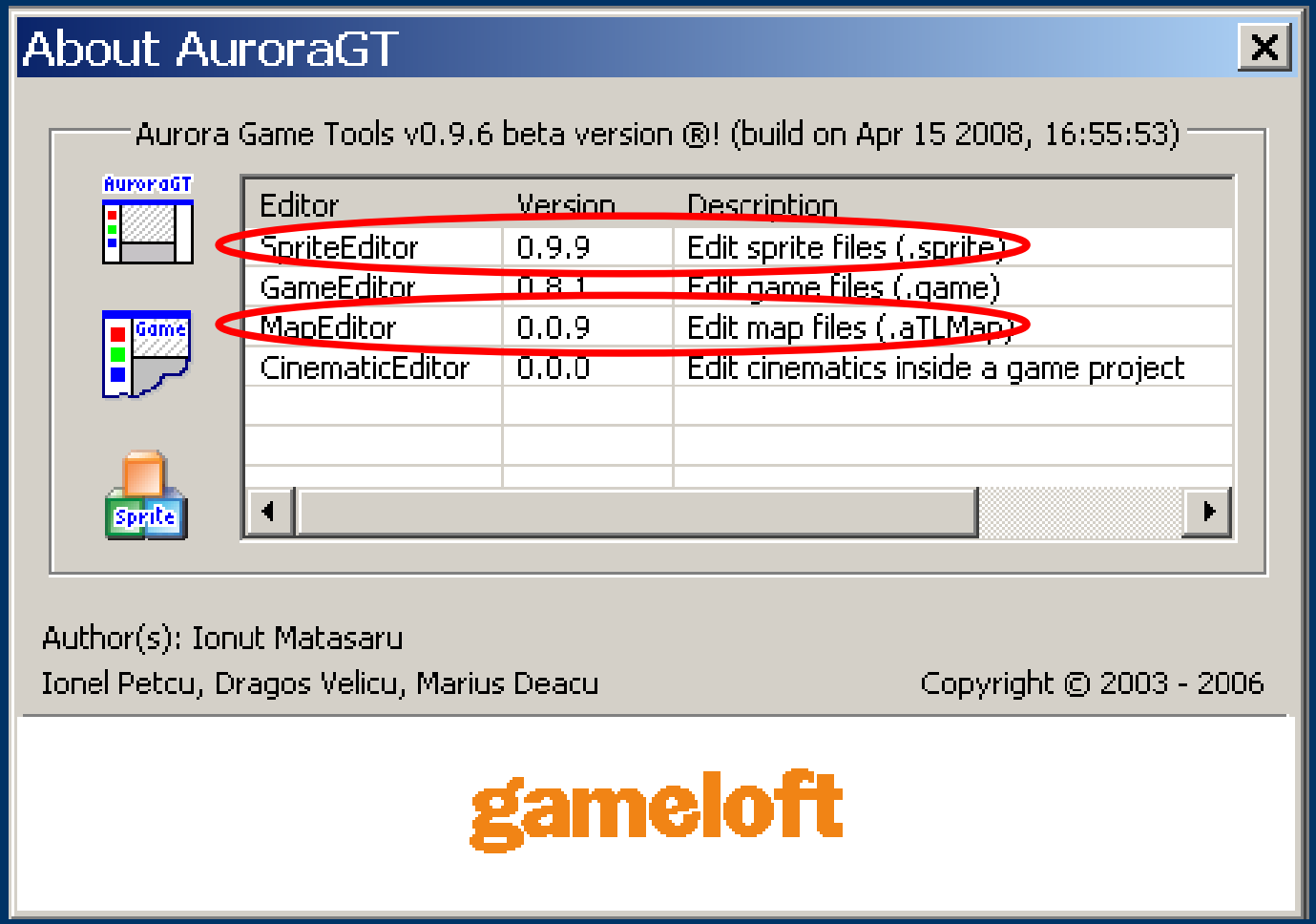
03: Optimizing and Exporting

gameloft

Version

25/01/08	Diego.Mercado@gameloft.com	0.0.1	Initial draft
27/02/08	Diego.Mercado@gameloft.com	0.0.2	Added Tileset editor
29/02/08	Diego.Mercado@gameloft.com	0.0.3	Modified gpl2act args, compound graphic & minor changes
10/03/08	Diego.Mercado@gameloft.com	1.0.0	Reorder some slides & minor changes
10/03/08	Diego.Mercado@gameloft.com	1.0.1	Added mask subdivision, MapEditor including isometric maps (r1006) & some optimizations
17/04/08	Diego.Mercado@gameloft.com	1.0.2	Added preview of an animation, more flags, support for non-indexed images and truecolor bmp & updated to r1093: support for more types (triangles & arcs), new bsprite's chunks, and some minor changes
22/04/08	Diego.Mercado@gameloft.com	1.0.3	Added Content & Contact Us pages
02/06/08	Diego.Mercado@gameloft.com	1.0.4	Fixed some bugs at the exporting sprite section

Reference Version[1]



[1] https://terminus.mdc.gameloft.org/vc/tools/AuroraGT (r1093)

Content

AuroraGT

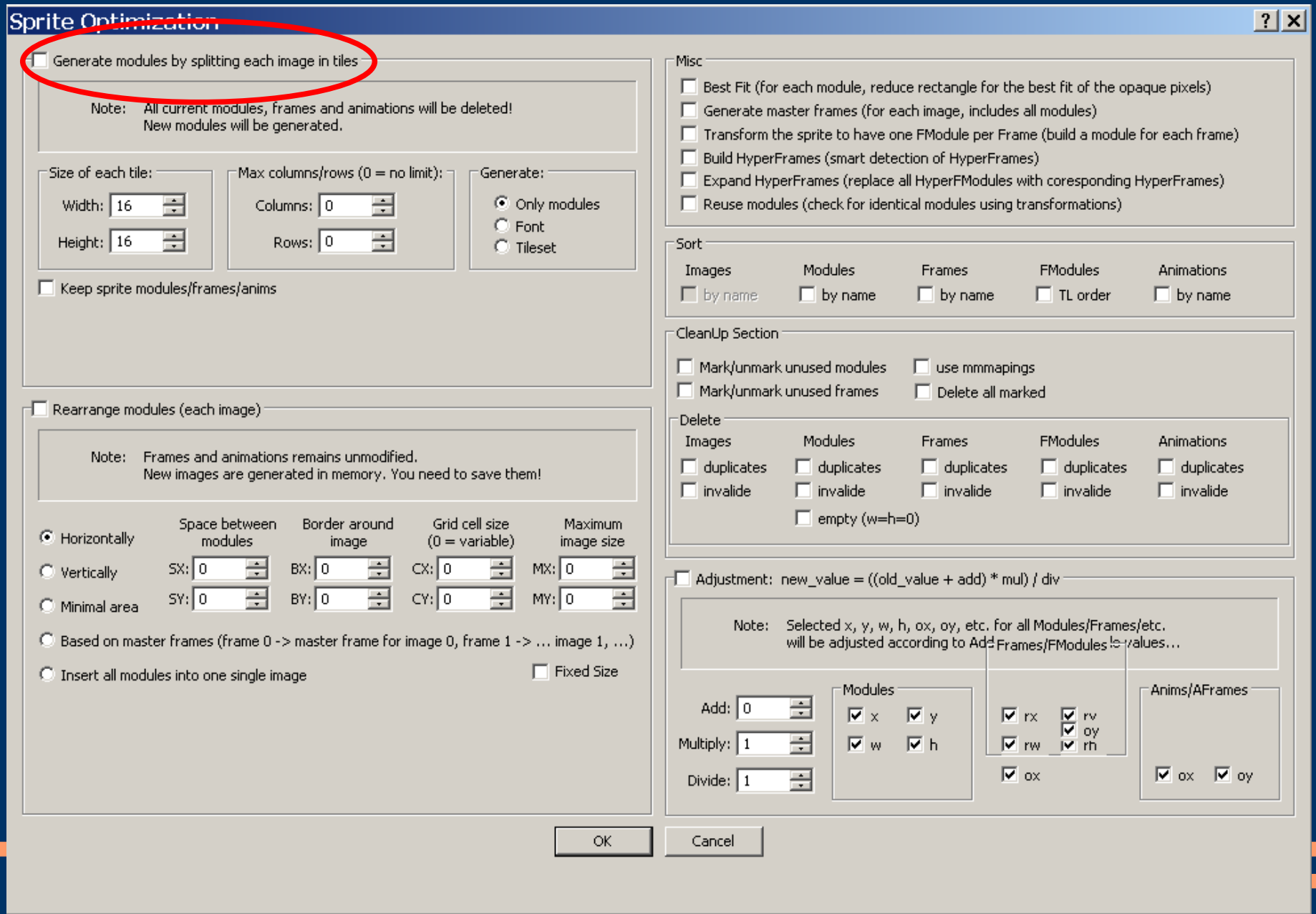
- **AuroraGT** (Aurora Game Tools)
  - Is:
    - A sprite editor
    - A game designing tool
  - It has 3 main different versions:
    - Normal (`AuroraGT.exe`)
    - Home-Edition (`AuroraGT_HE.exe`)
    - Unicode-Edition (`AuroraGT_unicode.exe`)
  - The extensions of its files are:
    - Sprites: `*.sprite`
    - Games: `*.game`
    - Maps: `*.aTLMap`

Sprite
Optimization

- You may want to:
  - Generate modules splitting images
  - Rearrange modules in the image
  - Resize the images / modules
  - Remove duplicated/invalid stuff
  - Sort objects by name
- Go to....
  - Tools -> Optimization...

Sprite

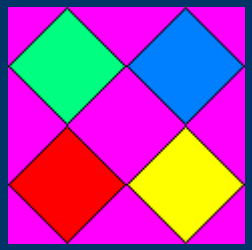
Optimization – Generating modules splitting an image

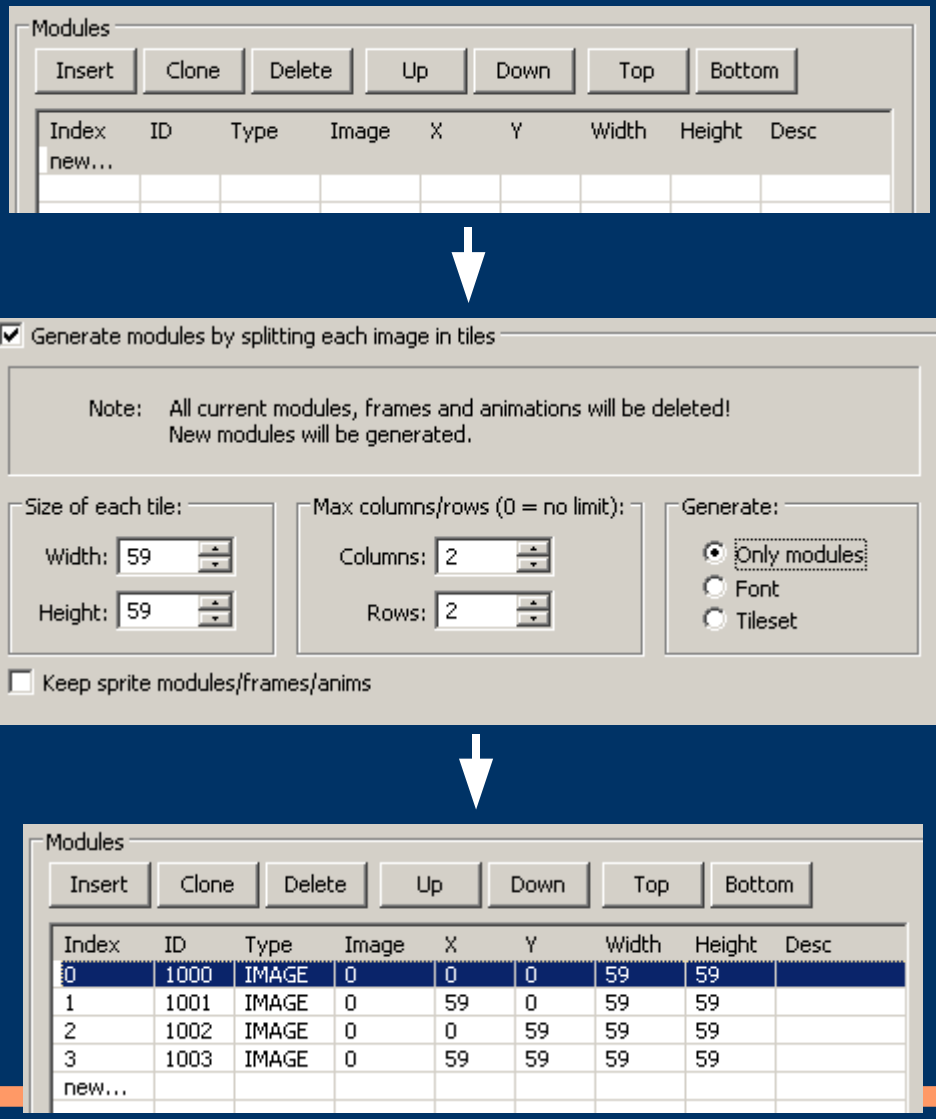


Sprite

Optimization – Generating modules splitting an image

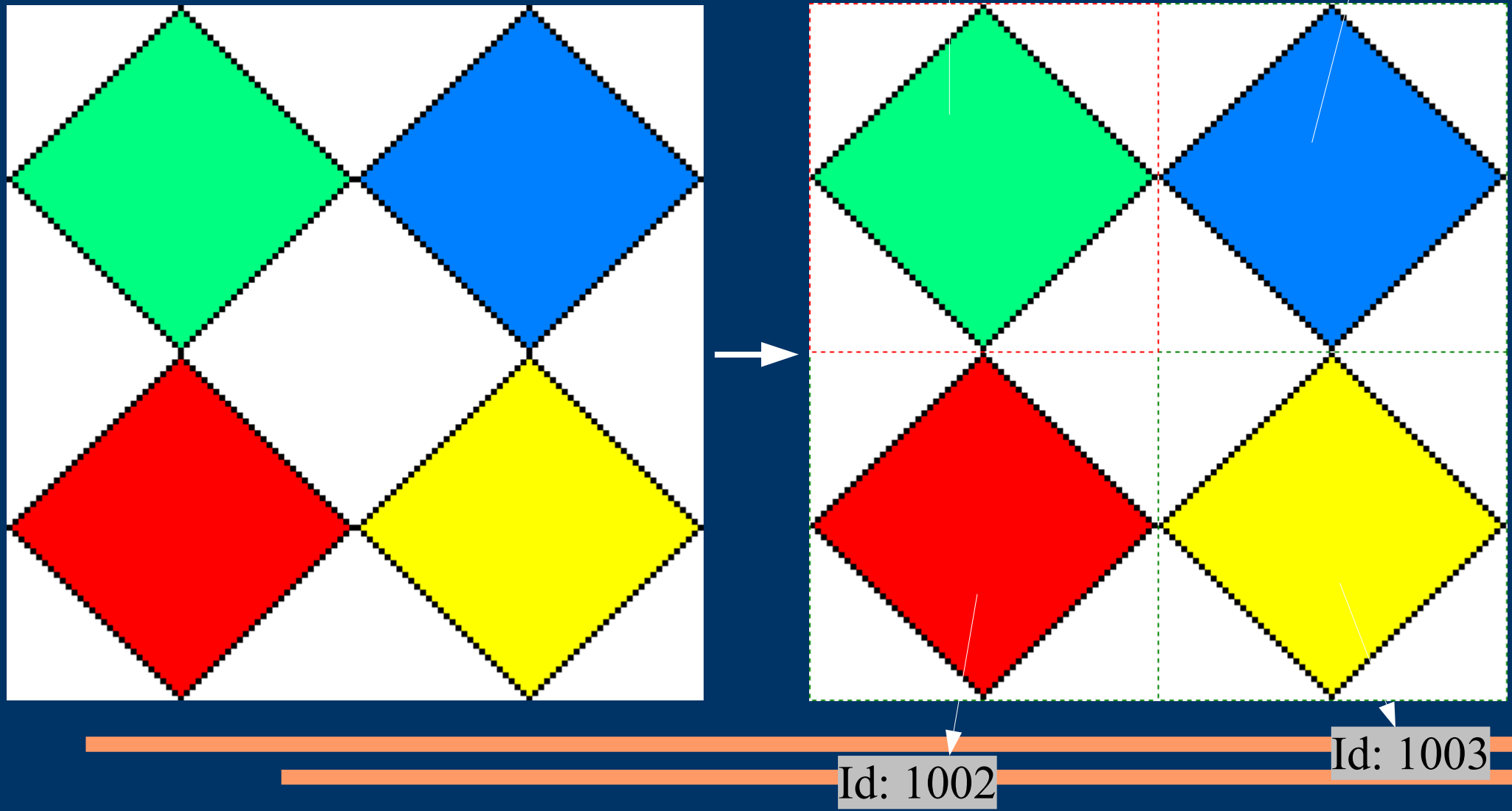
ORIGINAL IMAGE





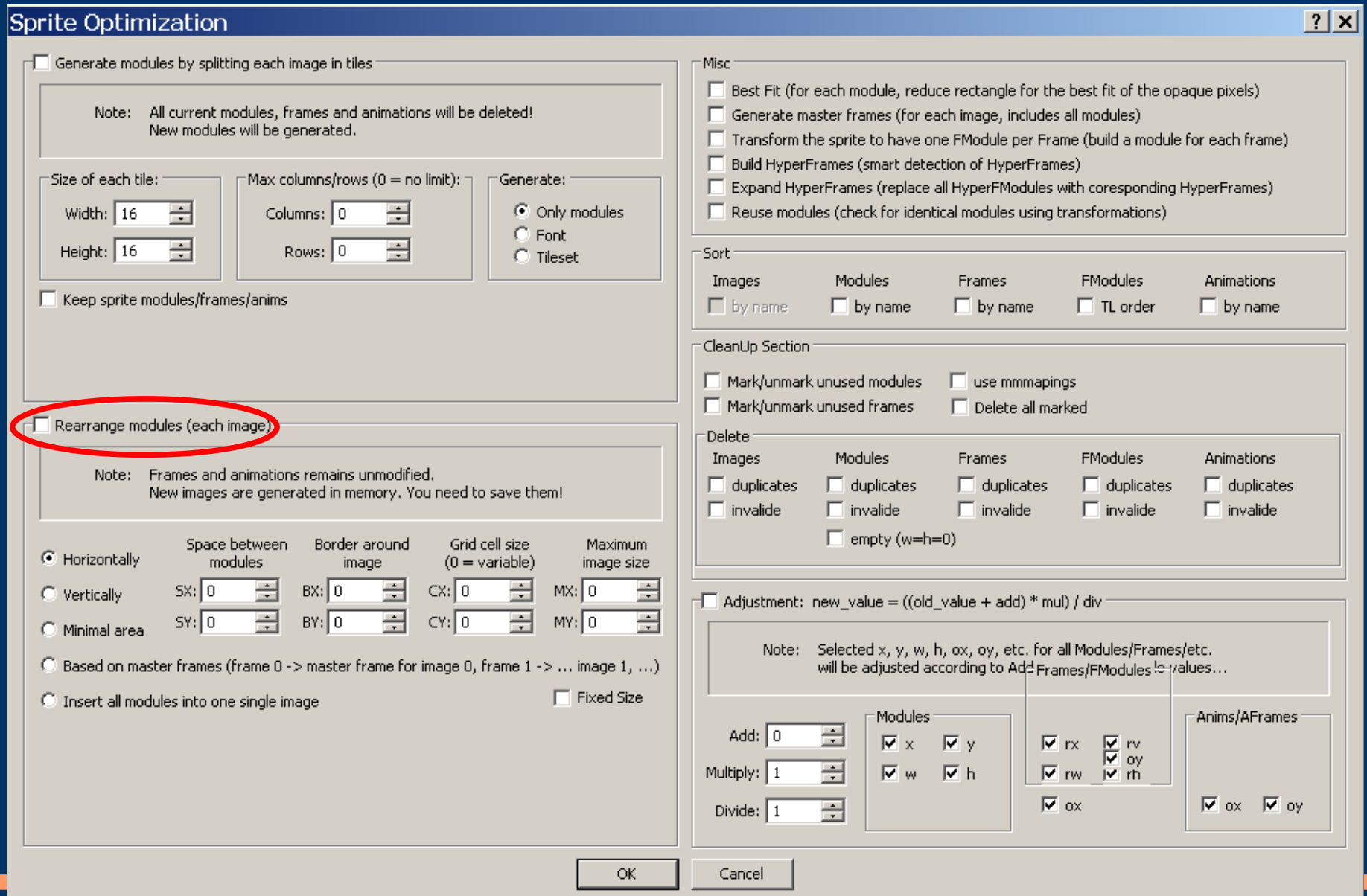
Sprite

Optimization – Generating modules splitting an image



Sprite

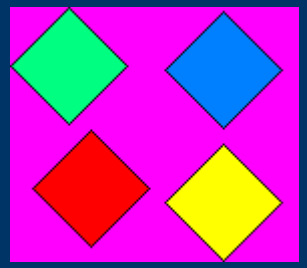
Optimization – Rearrange modules



Sprite

Optimization – Rearrange modules

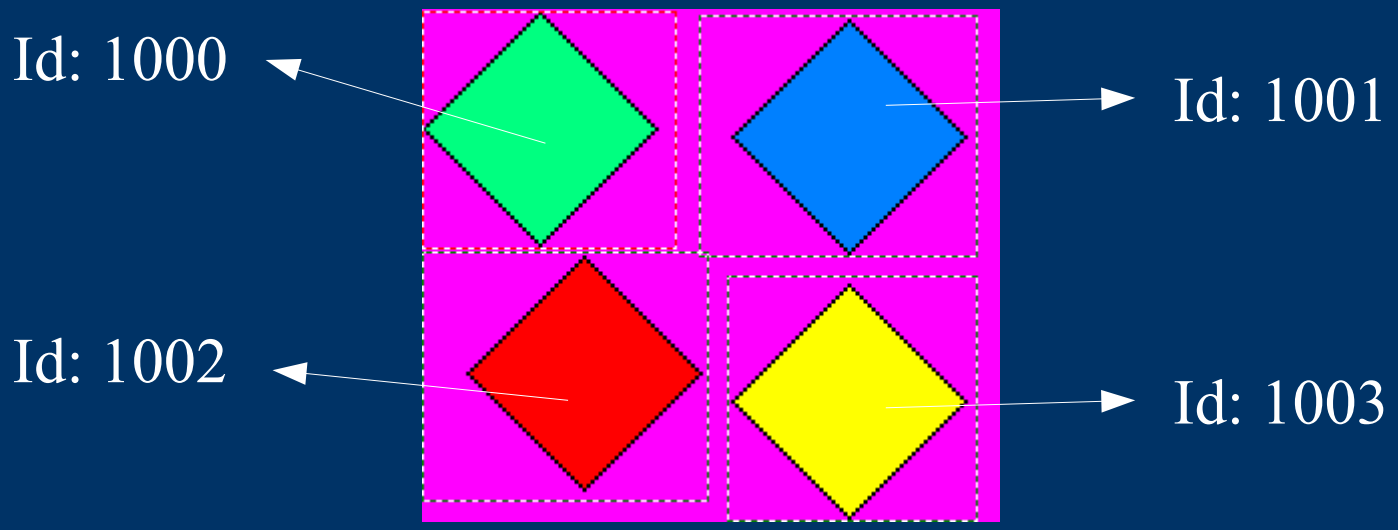
ORIGINAL IMAGE



Modules

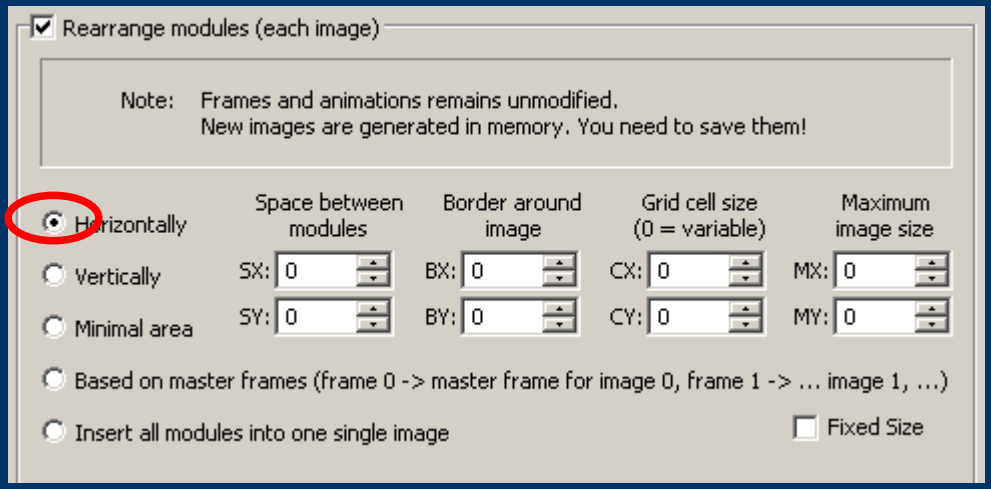
Insert | Clone | Delete | Up | Down | Top | Bottom

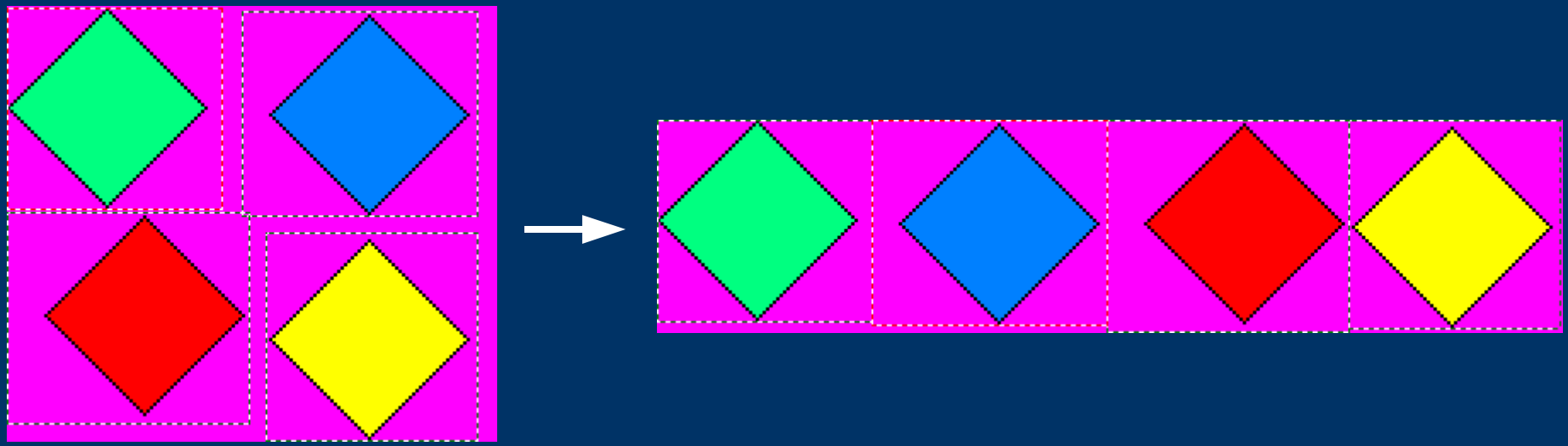
Index	ID	Type	Image	X	Y	Width	Height	Desc
0	1000	IMAGE	0	0	0	63	59	
1	1001	IMAGE	0	69	1	69	60	
2	1002	IMAGE	0	0	60	71	62	
3	1003	IMAGE	0	76	66	62	61	
new...								



Sprite

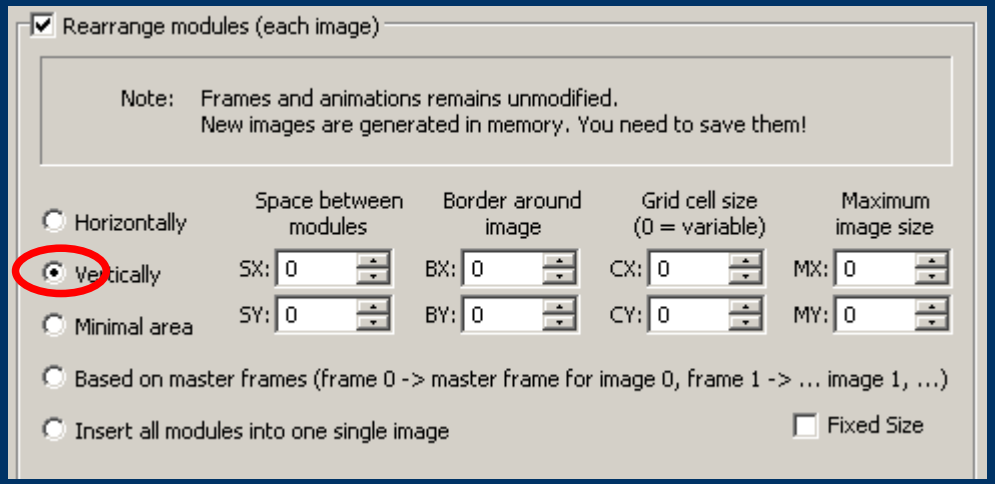
Optimization – Rearrange modules – Horizontally

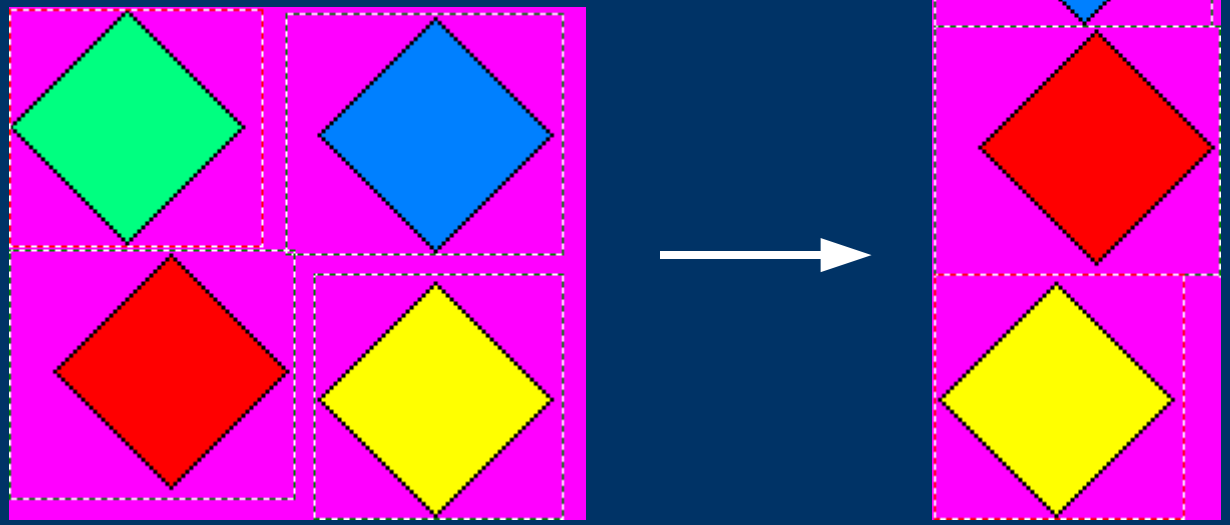




Sprite

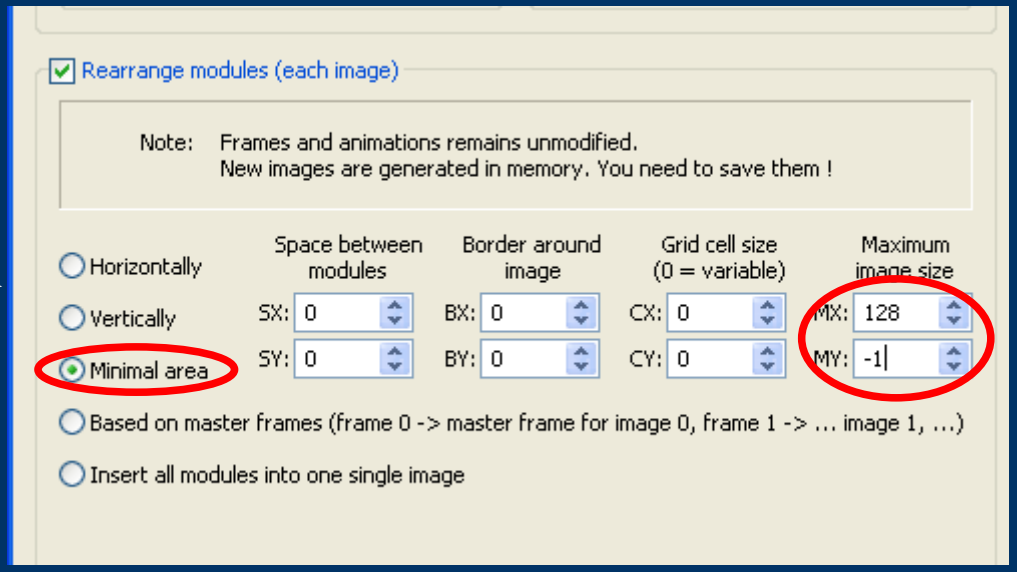
Optimization – Rearrange modules – Vertically





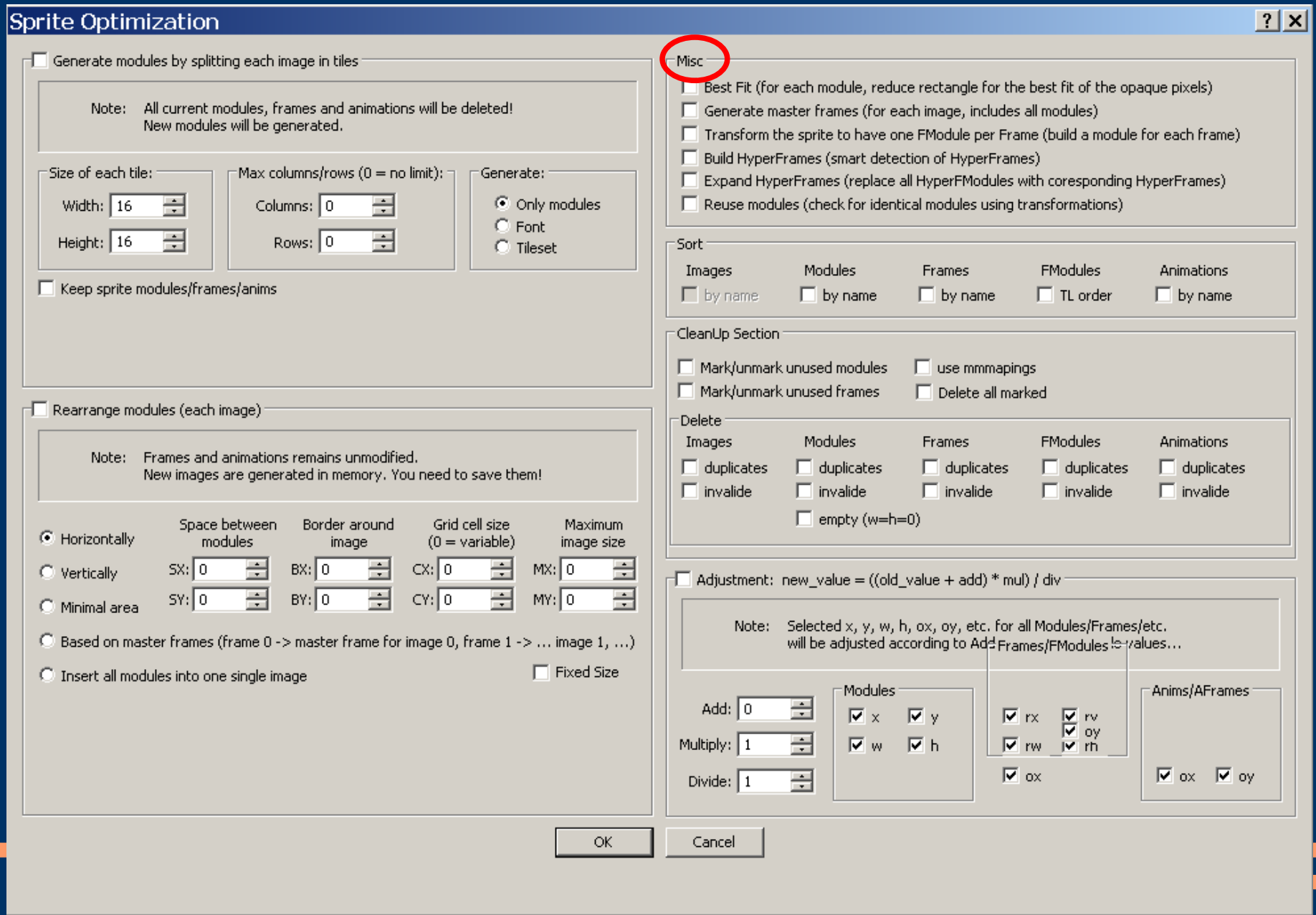
Sprite

Optimization – Rearrange modules – Minimal area



Sprite

Optimization – MISC



Sprite

Optimization – MISC

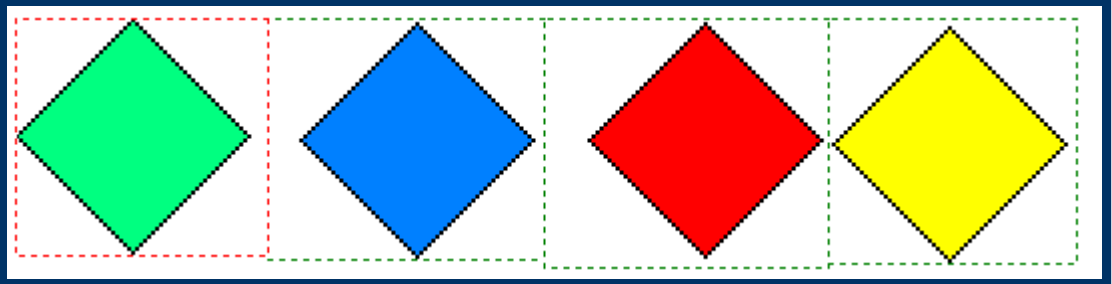
Misc

- [x] Best Fit (for each module, reduce rectangle for the best fit of the opaque pixels)
- [ ] Generate master frames (for each image, includes all modules)
- [ ] Transform the sprite to have one FModule per Frame (build a module for each frame)
- [ ] Build HyperFrames (smart detection of HyperFrames)
- [ ] Expand HyperFrames (replace all HyperFModules with coresponding HyperFrames)
- [ ] Reuse modules (check for identical modules using transformations)

Modules

Insert | Clone | Delete | Up | Down | Top | Bottom

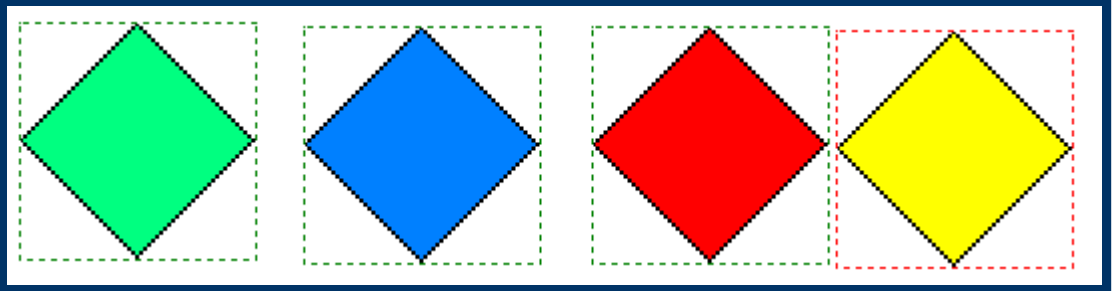
Index	ID	Type	Image	X	Y	Width	Height	Desc
0	1000	IMAGE	0	0	0	63	59	
1	1001	IMAGE	0	63	0	69	60	
2	1002	IMAGE	0	132	0	71	62	
3	1003	IMAGE	0	203	0	62	61	
new...								



Modules

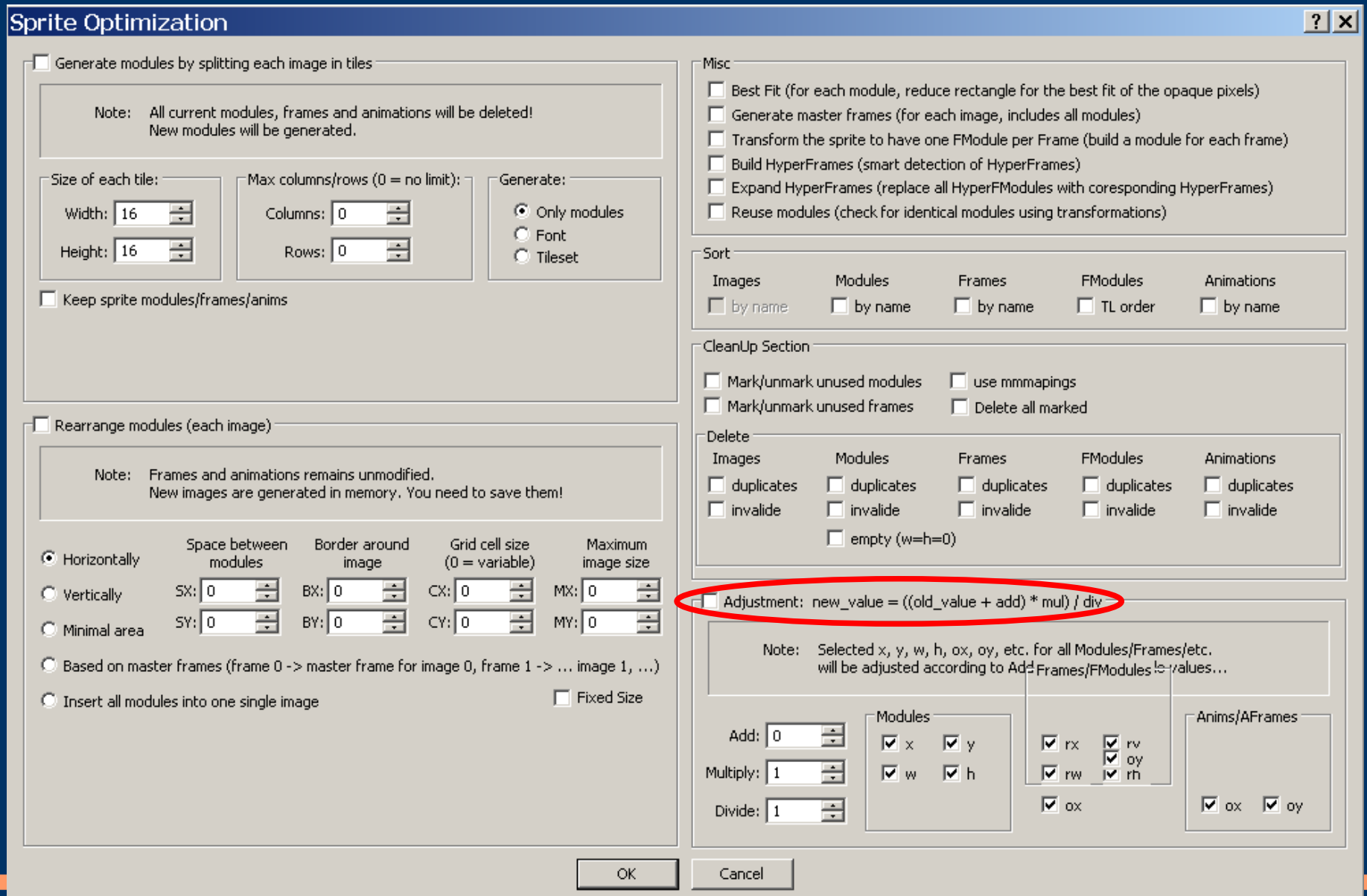
Insert | Clone | Delete | Up | Down | Top | Bottom

Index	ID	Type	Image	X	Y	Width	Height	Desc
0	1000	IMAGE	0	0	0	59	59	
1	1001	IMAGE	0	71	1	59	59	
2	1002	IMAGE	0	143	1	59	59	
3	1003	IMAGE	0	204	2	59	59	
new...								



Sprite

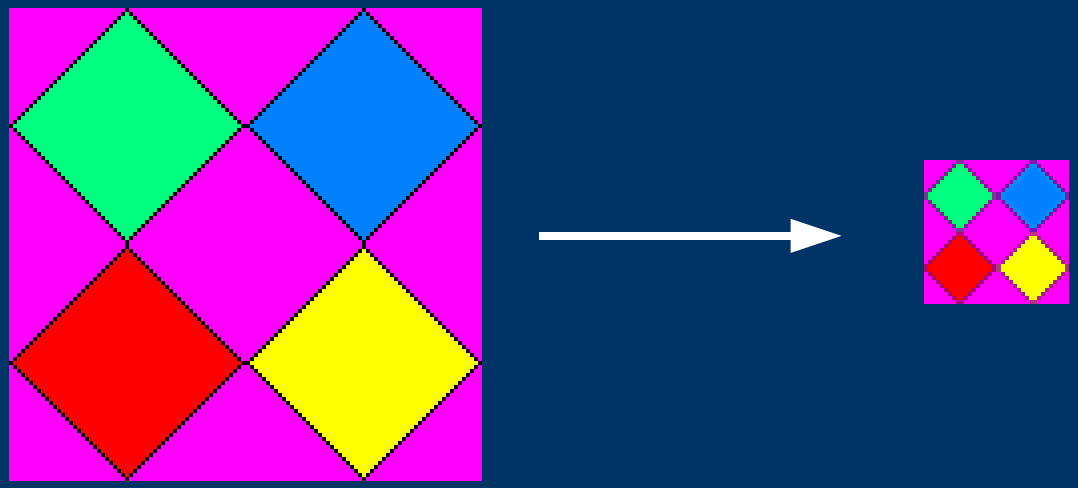
Optimization – Adjustment



Sprite

Optimization – Adjustment

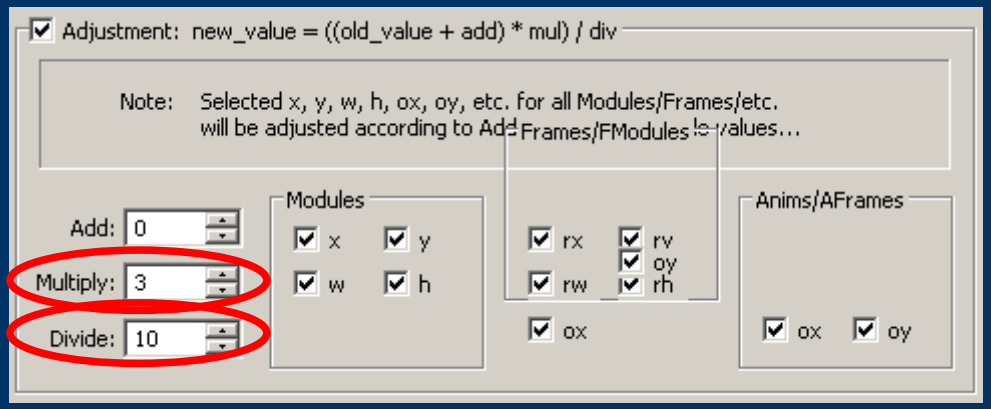
- Imagine that we need to reduce 30% an image:

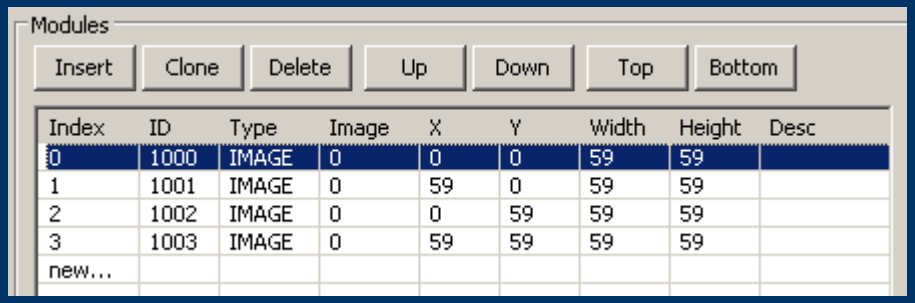


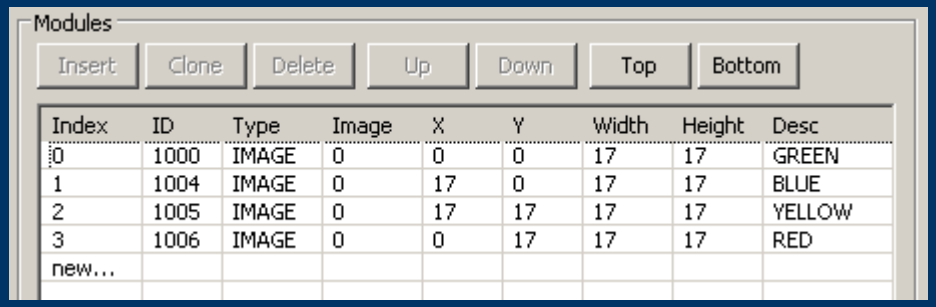
Sprite

Optimization – Adjustment

- We use the adjustment tool, and since is 30% (3/10) we multiply by 3 and divide by 10:







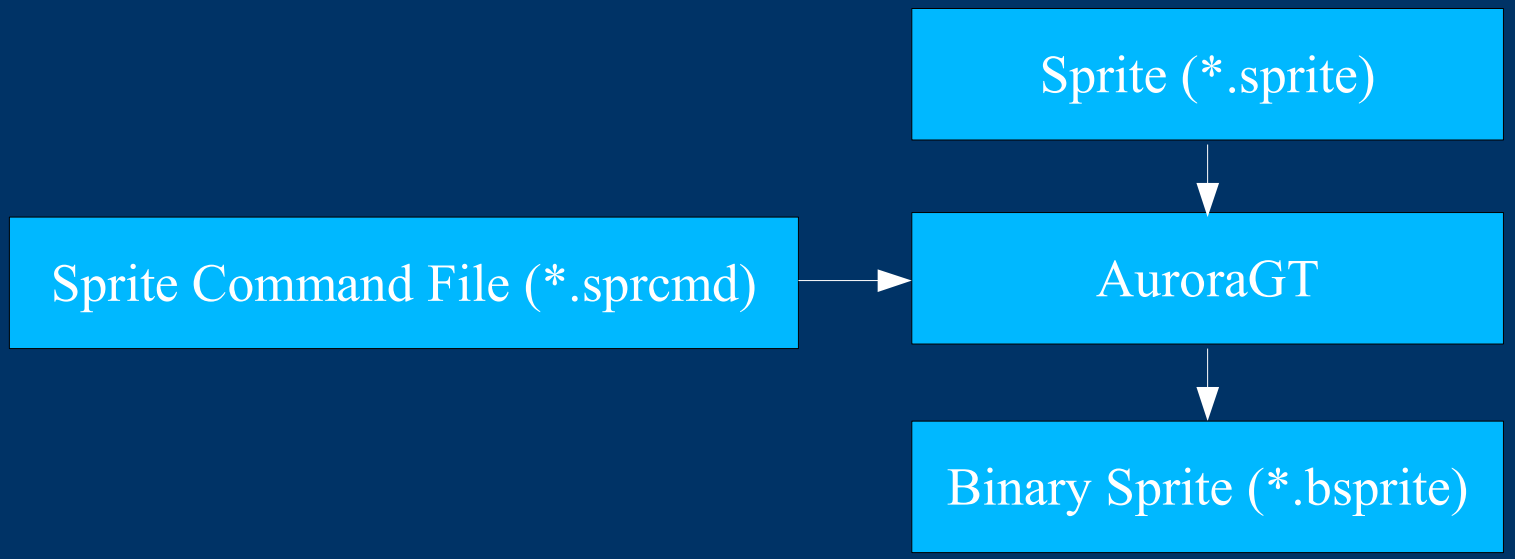
NOTE: obviously you need to resize the image with a graphic tool and then re-load it at the SpriteEditor

Exporting a Sprite

- What can you export from a Sprite
  1) Binary Sprite
     - A packed sprite (`*.bsprite` files) that may have the image data in it decoded then by the ASprite class
  2) Images
     - Save the images from each module, frame or animation
  3) Module Mappings
     - Save 1 or more modules (`*.mmp` files)

Exporting a Sprite
Binary sprite

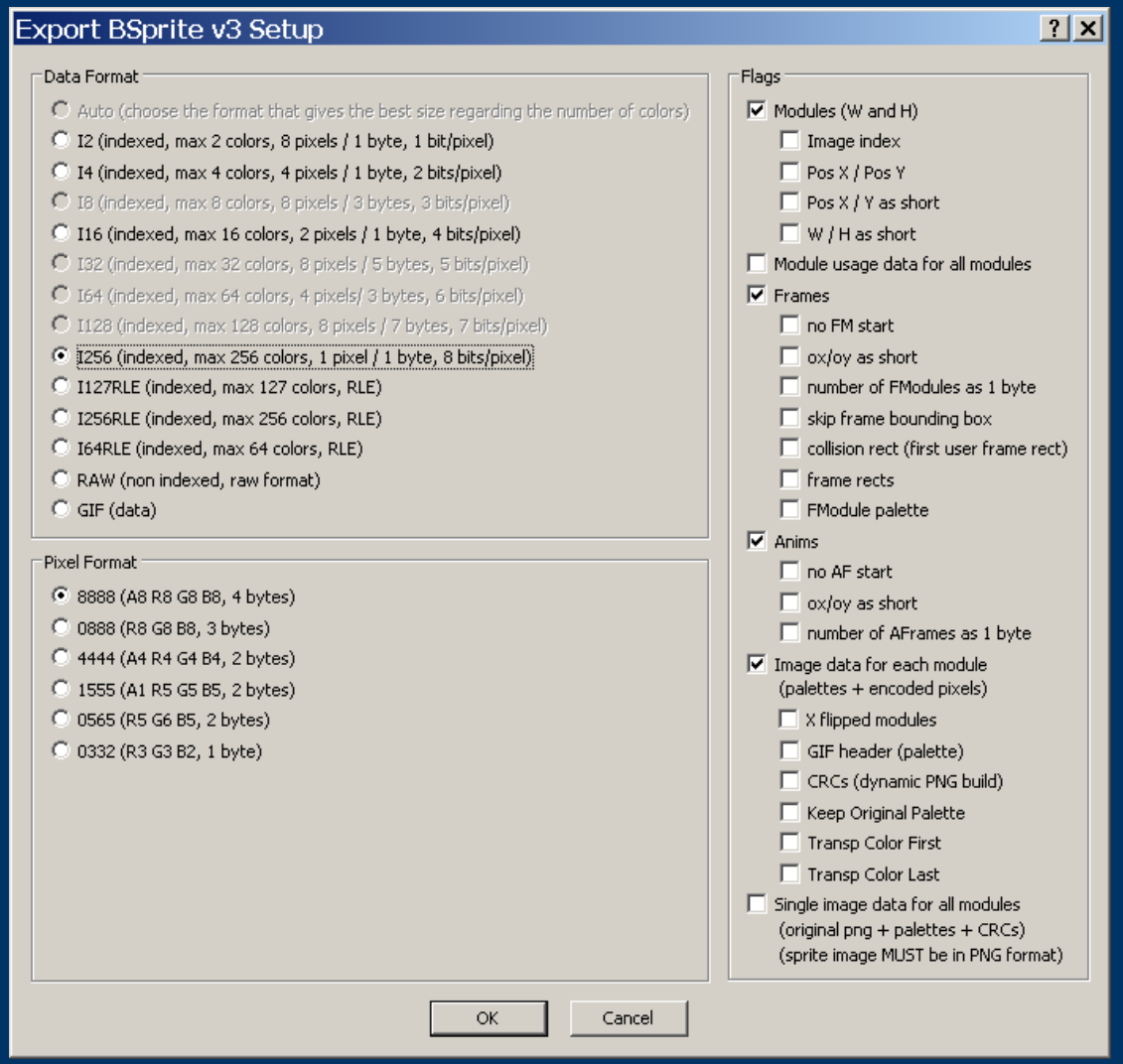
- You have 2 options for exporting a binary sprite:
  - GUI: File -> Export -> Bsprite...
  - Script: Through a SPRCMD file



Exporting a Sprite

Binary sprite

- Through the GUI



Exporting a Sprite
Binary sprite

- Through a script
  - The file must have extension (".sprcmd") and must be specified between quotation marks
    - Use: AuroraGT.exe "file.sprcmd"
  - You can use C++ comment style (//, /* ... */) to comment lines in the script file
  - You can specify one or more images that:
    - can be indexed or not *
    - can be a bmp truecolor (24bpp and 32 bpp)
    - cannot be compressed

Exporting a Sprite

Binary sprite – sprcmd file

- It's specified in:
  - %AURORAGT%/doc/AuroraDocProgrammers/sprmcmd.txt
- You have 45 commands. The most common are:
  - `SetGlobalBSpriteFlags(flags)`
  - `Load("path\in_file.sprite")`
  - `SetPaletteColor(image, palette, color_index, mask, color)`
  - `SetPalette(image, palette, { 0xAARRGGBB ... })`
  - `LoadPalette(image, palette, "file.act" or "IMAGE")`
  - `ExportBSpriteEx("path\out_file.bsprite", flags, encode, pixelformat)`

Exporting a Sprite

Binary sprite – sprcmd file

- For example,

```
Setglobalbspriteflags(BS_MODULES | BS_FRAMES
                      | BS_ANIMS | BS_MODULE_IMAGES)

Load("golfer.sprite")
   LoadPalette(0, 0, "golfer_gold.act")    // DEFAULT
   LoadPalette(0, 1, "golfer_red.act")     // T.WOOD
   ExportBSpriteEx("golfer.bsprite", GLOBAL, I64RLE, _8888)
   ExportBSpriteEx("golfer_4444.bsprite", GLOBAL, I64RLE, _4444)

Load("fonts/fontM.sprite")
   LoadPalette(0, 0, "IMAGE")
   LoadPalette(0, 1, "fontM_yellow.act")
   SetPalette(0, 2, {0x00000000 0xFF0000FF 0xFF000000})
   ExportBSpriteEx("fontM.bsprite", GLOBAL, I16, _8888)
   ExportBSpriteEx("fontM_4444.bsprite", GLOBAL, I16, _4444)
```

export.sprcmd

Exporting a Sprite

Binary sprite – sprcmd file

- Notice that you can set the flags GLOBAL by using the function Setglobalbspriteflags

```
Setglobalbspriteflags(BS_MODULES | BS_FRAMES
                      | BS_ANIMS | BS_MODULE_IMAGES)

Load("golfer.sprite")
    LoadPalette(0, 0, "golfer_gold.act")    // DEFAULT
    LoadPalette(0, 1, "golfer_red.act")     // T.WOOD
    ExportBSpriteEx("golfer.bsprite", GLOBAL, I64RLE, _8888)
    ExportBSpriteEx("golfer_4444.bsprite", GLOBAL, I64RLE, _4444)

Load("fonts/fontM.sprite")
    LoadPalette(0, 0, "IMAGE")
    LoadPalette(0, 1, "fontM_yellow.act")
    SetPalette(0, 2, {0x00000000 0xFF0000FF 0xFF000000})
    ExportBSpriteEx("fontM.bsprite", GLOBAL, I16, _8888)
    ExportBSpriteEx("fontM_4444.bsprite", GLOBAL, I16, _4444)
```

Exporting a Sprite

Binary sprite – sprcmd file

FLAGS

FLAG	DESCRIPTION
BS_AF_OFF_SHORT	export af offsets as shorts
BS_ANIMS	export animations
BS_FM_FREE_ROTATE_SCALE	export also the freerotate/scale parameters for frame modules
BS_FM_OFF_SHORT	export fm offsets as shorts
BS_FM_PALETTE	export palette used by the module
BS_FRAME_COLL_RC	export frame collision rect
BS_FRAME_RECTS	export frame rects
BS_FRAMES	export frames
BS_GIF_HEADER	export gif header instead of palette
BS_IMAGE_SIZE_INT	export the image size as int
BS_KEEP_PAL	keep original palette (do not optimize colors)
BS_MD_5_BYTES	export non MD_IMAGE modules (MD_RECT, MD_FILL_RECT) encoded on 1 + 4 bytes (type + color)
BS_MODULE_IMAGES	export palettes and images
BS_MODULE_IMAGES_FX	export encoded images for each module (flipped horizontally)
BS_MODULE_IMAGES_TC_BMP	export RGB for each pixel loaded from a true color BMp for each module (works with _0888, _8888 and RAW);
BS_MODULE_IMAGES_USED	export encoded images for each module (used combination Fx/Fy/Rot)
BS_MODULES	export modules
BS_MODULES_IMG	export image index for each module
BS_MODULES_USAGE	export for each module which transformations are used in the sprite
BS_MODULES_WH_SHORT	export Width/Height for each module as short
BS_MODULES_XY	export PosX/PosY for each module
BS_MODULES_XY_SHORT	export PosX/PosY for each module as short
BS_MULTIPLE_IMAGES	export sprite that contain multiple images with palette for each image
BS_NAF_1_BYTE	export naf as byte
BS_NFM_1_BYTE	export nfm as byte
BS_NO_AF_START	do not export start of AFrames
BS_NO_FM_START	do not export start of FModules
BS_OPTIMIZE	optimize export (works with BS_SINGLE_IMAGE)
BS_PNG_CRC	export PNG additional info (CRCs) for each module
BS_SINGLE_IMAGE	Export sprite PNG + (PLTE+CRC) + (tRNS+CRC)
BS_SKIP_FRAME_RC	do not export frame rect
BS_TRANSP_FIRST	move transparency as the first color(s)
BS_TRANSP_LAST	move transparency as the last color(s)

Exporting a Sprite

Binary sprite – sprcmd file

- You may use some of these alias:

VALUE	EQUIVALENT TO...
BS_DEFAULT_DOJA	BS_MODULES | BS_FRAMES | BS_ANIMS
BS_DEFAULT_MIDP2	BS_MODULES | BS_FRAMES | BS_ANIMS | BS_MODULE_IMAGES
BS_DEFAULT_NOKIA	BS_DEFAULT_MIDP2
BS_DEFAULT_MIDP1	BS_MODULES | BS_MODULES_XY | BS_FRAMES | BS_ANIMS
BS_DEFAULT_MIDP1b	BS_MODULES | BS_FRAMES | BS_ANIMS | BS_MODULE_IMAGES | BS_PNG_CRC

Exporting a Sprite

Binary sprite – Pixel format

- Pixel/Color Format
  - Specify each color of the palette

Name	Define	No. Bytes	Bits per Color				OBS
			Alpha	Red	Green	Blue	
(ignore)	USE_ORIGINAL_PAL_8888	n/a	0	0	0	0	
_8888	USE_PIXEL_FORMAT_8888	4	8	8	8	8	
_F888		4	8	8	8	8	No alpha
_0888		3	0	8	8	8	
_4444	USE_PIXEL_FORMAT_4444	2	4	4	4	4	
_F444		2	4	4	4	4	No alpha
_1555	USE_PIXEL_FORMAT_1555	2	1	5	5	5	
_F555		2	0	5	6	5	No alpha
_0565	USE_PIXEL_FORMAT_0565	2	0	5	6	5	
_0332		1	0	3	3	2	
NULL		0	0	0	0	0	if BS_MODULE_IMAGES is not used

Exporting a Sprite
Binary sprite – Data format

- Data format
  - I2
  - I4
  - I16
  - I256
  - I64RLE
  - I127RLE
  - I256RLE
  - RAW
  - GIF

Exporting a Sprite

Binary sprite – Data format

- I*n*
  - *I* indicates that is indexed
  - *n* refers to the number of colors
- RLE refers to "*Run-length encoding*"
  - It's a way to compress data
    - AAABCCDDDDDD -> 3A1B2C6D
- RAW
  - Export RGB values for each pixel instead of a palette index -> not indexed
- GIF
  - Encodes GIF data for DOJA

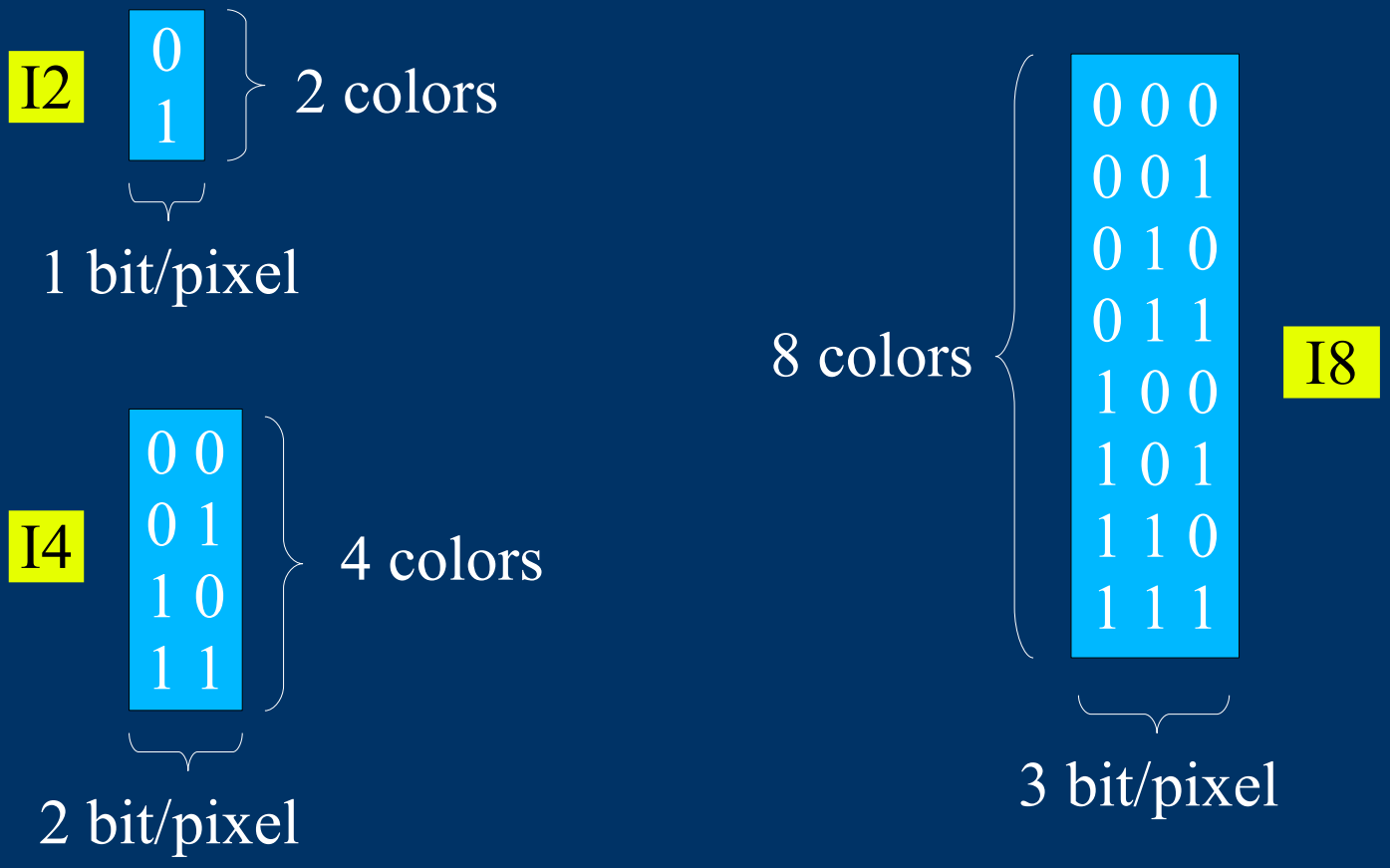
Exporting a Sprite

Binary sprite – Data format

I2	maximum 2 colors (indexed), packed data 8 pixels / 1 byte (1 bit /pixel)
I4	maximum 4 colors (indexed), packed data 4 pixels / 1 byte (2 bits/pixel)
I8	maximum 8 colors (indexed), packed data 8 pixels / 3 bytes (3 bits/pixel)
I16	maximum 16 colors (indexed), packed data 2 pixels / 1 byte (4 bits/pixel)
I32	maximum 32 colors (indexed), packed data 8 pixels / 5 bytes (5 bits/pixel)
I64	maximum 64 colors (indexed), packed data 4 pixels / 3 bytes (6 bits/pixel)
I128	maximum 128 colors (indexed), packed data 8 pixels / 7 bytes (7 bits/pixel)
I256	maximum 256 colors (indexed), raw data 1 pixel / 1 byte (8 bits/pixel)
I64RLE	maximum 64 colors (indexed), compressed data (RLE)
I127RLE	maximum 127 colors (indexed), compressed data (RLE)
I256RLE	maximum 256 colors (indexed), compressed data (RLE)
RAW	non indexed, raw format
GIF	encode gif data for DOJA

Exporting a Sprite

Binary sprite – Data format



Exporting a Sprite

Binary sprite – Example

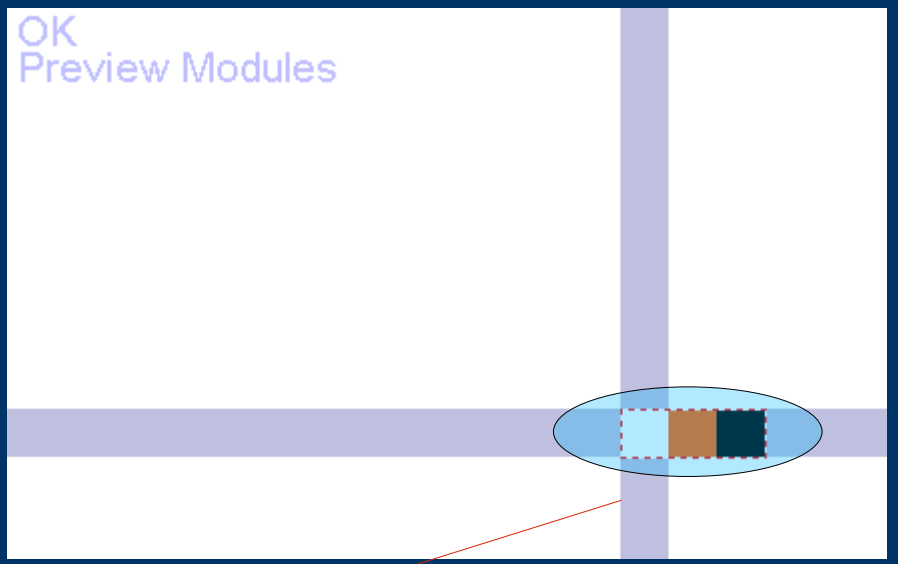
- Let's say that you have this image:

BITMAP
3 colors
3x1 pixels

Exporting a Sprite

Binary sprite – Example

- We create a module with the following characteristics:
  - Type: Image
  - Width: 3
  - Height: 1
  - X pos: 0
  - Y pos: 0



Index	ID	Type	Image	X	Y	Width	Height	Desc
0	1001	IMAGE	0	0	0	3	1	

Exporting a Sprite
Binary sprite – Example

- So, we generate the *bsprite* by using:
  - I4 Data Format
  - 1555 Pixel Format
  - Flags:
    - Modules (W and H)
    - Image data for each module

Exporting a Sprite

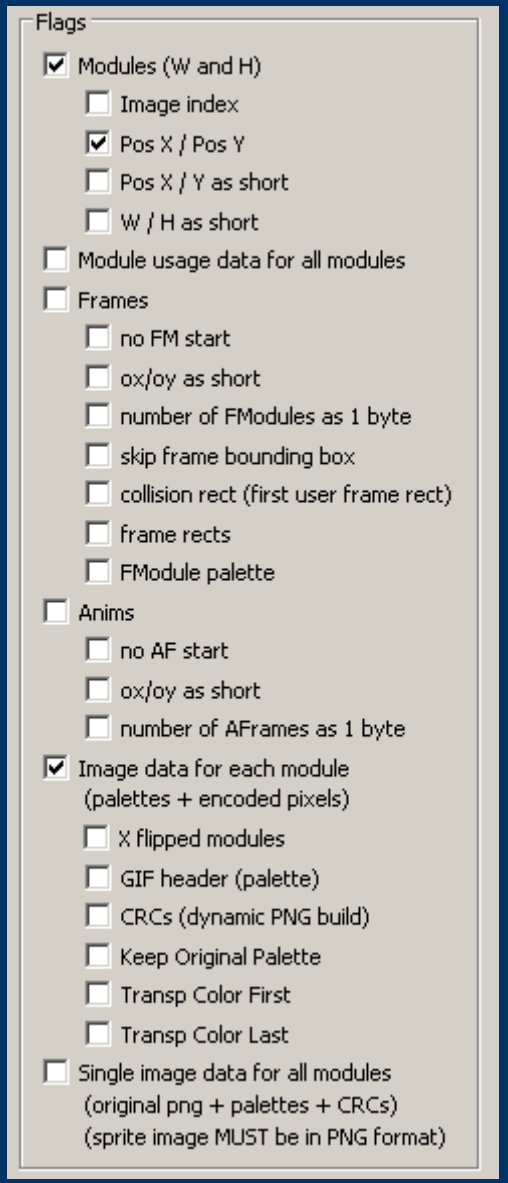
Binary sprite – Example

- 1555 (A1 R5 G5 B5, 2 bytes)
  - Because the image was created using true color (24 bits, 8888) the red color may differ a little: aurora takes an approximation to do it
- I4
  - To specify which color we use. Since we use only 3 colors, it makes sense to use this data format. (I2 will produce an error)

Exporting a Sprite

Binary sprite – Example

- Modules (W and H) & Image data for each module
  - The flags are set at the beginning. This helps the *ASprite* class to know how the *bsprite* file should be parsed
  - The image data is attached at the end



Exporting a Sprite
Binary sprite – Example

- `bsprite_v5.txt` specifies the bsprite's chunks
  - The file is located at `%AURORAGT%/doc/AuroraDocProgrammers/`
  - One important difference between v5 and v4 (`bsprite_v4.txt`) is that v4 add 4 bytes more in the header to specify extraflags (i.e. `BS_FM_FREE_ROTATE_SCALE`)

Exporting a Sprite

Binary sprite – Example

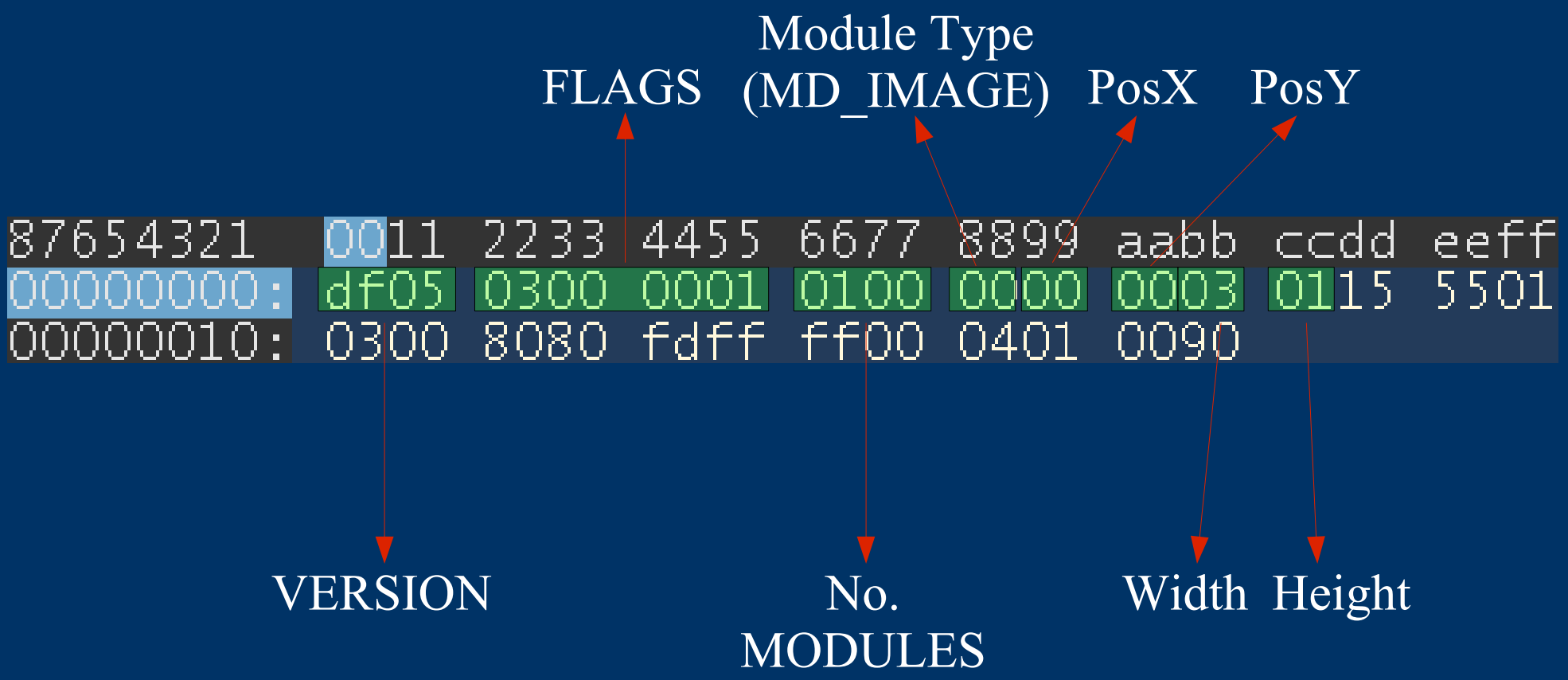
- It's written as pseudocode and the flow control is consider by the use of how the flags are set

```
////////////////////////////////////////////////
// Header...

2 bytes -> file type/version (BSPRITE_v005 = 0x05DF)
4 bytes -> flags (BS_...)

////////////////////////////////////////////////
// Modules...

if (BS_MODULES)
{
    2 bytes -> nm = number of modules
    for each module (nm):
    {
                    if (next byte == 0xFF) // -> MD_RECT
                    {
```

FLAG

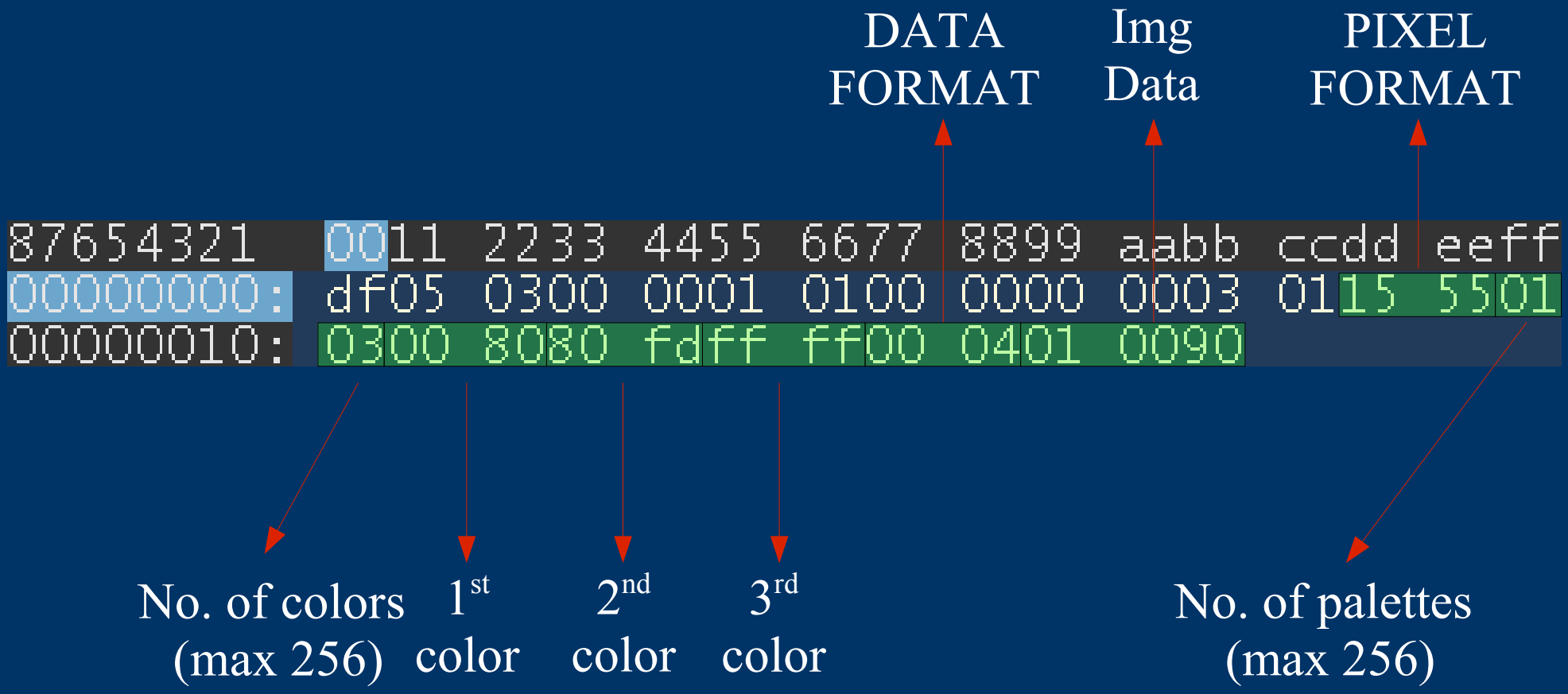
Exporting a Sprite

Binary sprite – Example



Exporting a Sprite

Binary sprite – Example



1555 => np * nc * 2 byte -> each color is 2 byte long

Exporting a Sprite

Binary sprite – Example – ASprite_Load.hxx

- Reading BSPRITE VERSION

```
short bs_version = (short)((file[offset++]&0xFF) +
    ((file[offset++]&0xFF)<<8));
```

```
87654321   0011 2233 4455 6677 8899 aabb ccdd eeff
00000000:  df05 0300 0001 0100 0000 0003 0115 5501
00000010:  0300 8080 fdff ff00 0401 0090
```

Exporting a Sprite

Binary sprite – Example – ASprite_Load.hxx

- Reading FLAGS

```
_bs_flags = ((file[offset++]&0xFF)) +
     ((file[offset++]&0xFF)<< 8) +
     ((file[offset++]&0xFF)<<16) +
     ((file[offset++]&0xFF)<<24);
```

```
87654321   0011 2233 4455 6677 8899 aabb ccdd eeff
00000000:  df05 0300 0001 0100 0000 0003 0115 5501
00000010:  0300 8080 fdff ff00 0401 0090
```

Exporting a Sprite

Binary sprite – Example – ASprite_Load.hxx

- Reading NO. OF MODULES

```
_nModules = (file[offset++]&0xFF) +
     ((file[offset++]&0xFF)<<8);
```

```
87654321   0011 2233 4455 6677 8899 aabb ccdd eeff
00000000:  df05 0300 0001 0100 0000 0003 0115 5501
00000010:  0300 8080 fdff ff00 0401 0090
```

Exporting a Sprite

Binary sprite – Example – ASprite_Load.hxx

- Reading X & Y POSITIONS

```
TEST_BSPRITE_FLAGS(BS_MODULES_XY | BS_MODULES_XY_SHORT) {
	_modules_x = new T_MODULE_XY[_nModules];
	_modules_y = new T_MODULE_XY[_nModules];
}
//...
TEST_BSPRITE_FLAGS(BS_MODULES_XY) {
	_modules_x[i] = file[offset++];
	_modules_y[i] = file[offset++];
}
```

```
87654321   0011 2233 4455 6677 8899 aabb ccdd eeff
00000000:  df05 0300 0001 0100 0000 0003 0115 5501
00000010:  0300 8080 fdff ff00 0401 0090
```

Exporting a Sprite

Binary sprite – Example – ASprite_Load.hxx

- Reading WIDTH and HEIGHT

```
_modules_w = new T_MODULE_WH[_nModules];
_modules_h = new T_MODULE_WH[_nModules];
//...
for (int i = 0; i < _nModules; i++)    {
   //...
   _modules_w[i] = file[offset++];
   _modules_h[i] = file[offset++];
   //...
}
```

```
87654321   0011 2233 4455 6677 8899 aabb ccdd eeff
00000000:  df05 0300 0001 0100 0000 0003 0115 5501
00000010:  0300 8080 fdff ff00 0401 0090
```

Exporting a Sprite

Binary sprite – Example – ASprite_Load.hxx

- Reading PIXEL FORMAT

```
_pixel_format = (short)((file[offset++] & 0xFF) +
    ((file[offset++] & 0xFF) << 8));
```

```
87654321   0011 2233 4455 6677 8899 aabb ccdd eeff
00000000:  df05 0300 0001 0100 0000 0003 0115 5501
00000010:  0300 8080 fdff ff00 0401 0090
```

Exporting a Sprite

Binary sprite – Example – ASprite_Load.hxx

- Reading NO. OF PALETTES & COLORS

```
// Number of palettes...
_palettes = file[offset++] & 0xFF;

// Number of colors...
_colors = file[offset++] & 0xFF;
```

```
87654321   0011 2233 4455 6677 8899 aabb ccdd eeff
00000000:  df05 0300 0001 0100 0000 0003 0115 5501
00000010:  0300 8080 fdff ff00 0401 0090
```

Exporting a Sprite

Binary sprite – Example – ASprite_Load.hxx

- Reading COLORS (from our only palette)

```
_pal = new T_PAL[MAX_SPRITE_PALETTES][];
for (int p = 0; p < _palettes; p++) {
//...
    for (int c = 0; c < _colors; c++) {
        int _1555  = ((file[offset++] & 0xFF));
        _1555 += ((file[offset++] & 0xFF) << 8);
        int a = 0xFF000000;
        if ((_1555 & 0x8000) != 0x8000) {
            a = 0;
            _alpha = true;
        }
        // 1555 -> 8888
        int _8888 = (a |                    // A
             ((_1555 & 0x7C00) << 9) |   // R
             ((_1555 & 0x03E0) << 6) |   // G
             ((_1555 & 0x001F) << 3));   // B

        _pal[p][c] = (T_PAL)_8888;
    }
}
```

Sets _alpha

It converts from 1555 to 8888

```
87654321   0011 2233 4455 6677 8899 aabb ccdd eeff
00000000:  df05 0300 0001 0100 0000 0003 0115 5501
00000010:  0300 8080 fdff ff00 0401 0090
```

Exporting a Sprite

Binary sprite – Example – ASprite_Load.hxx

- Reading DATA FORMAT

  (I4 indexed, max 4 colors, 4 pixels / 1 byte)

```
_data_format = (short)((file[offset++] & 0xFF) +
     ((file[offset++] & 0xFF) << 8));
```

```
87654321   0011 2233 4455 6677 8899 aabb ccdd eeff
00000000:  df05 0300 0001 0100 0000 0003 0115 5501
00000010:  0300 8080 fdff ff00 0401 0090
```

Exporting a Sprite

Binary sprite – Example – ASprite_Load.hxx

- Reading IMAGE DATA

  (setting the length of _modules_data)

```
_modules_data_off = new T_MODULE_DATA_OFF[_nModules];
int len = 0;
int off = offset;

for (int m = 0; m < _nModules; m++)
{
    // Image data for the module...
    int size = (file[off++] & 0xFF) + ((file[off++] & 0xFF) << 8);

    _modules_data_off[m] = (T_MODULE_DATA_OFF)len;
    off += size;
    len += size;
}
```

```
87654321  0011 2233 4455 6677 8899 aabb ccdd eeff
00000000: df05 0300 0001 0100 0000 0003 0115 5501
00000010: 0300 8080 fdff ff00 0401 0090
```

Exporting a Sprite

Binary sprite – Example – ASprite_Load.hxx

- Reading IMAGE DATA
  (filling _modules_data)

```
for (int m = 0; m < _nModules; m++)
{
   // Image data for the module...
   int size = (file[offset++] & 0xFF) +
((file[offset++] & 0xFF) << 8);
   ASPRITE_TRACE_LOAD("frame[" + m + "] size = " +
size);
   System.arraycopy(file, offset, _modules_data,
_modules_data_off[m]/* & 0xFFFF*/, size);
   offset += size;
}
```

```
87654321  0011 2233 4455 6677 8899 aabb ccdd eeff
00000000: df05 0300 0001 0100 0000 0003 0115 5501
00000010: 0300 8080 fdff ff00 0401 0090
```

Exporting a Sprite

FFT File Format Template

- Suppose you need to add some modules to your sprite, but you exceed the max module number
- **What's the solution?**

Specify an FFT file! We can tell Aurora to export module numbers in a bigger data type for example!

FFT is a file with a specific syntaxis.

Available chunks in the next slide...

Exporting a Sprite

FFT File Format Template

- This file enable de DEV to specify every export parameter.
- More powerful than export.sprcmd files
- A little more complex to define
- Used for game exporting as well.
- Image data and sprite info can be splitted

Exporting a Sprite

FFT Files

- What's FFT?
  - It's a list of datatypes.

Exporting a Sprite

FFT Chunks

- FFT Specification here
- FFT Example here

Conclusion

- The Bsprite is a file that contains:
  - Coordinates and positions, number of frames, etc
  - Image data.
- Bsprite is read by Asprite.Load() ,method:
  - 0 Read Bsprite version
  - 1 Read Bsprite flags
  - 2 Read mod/frame/anim info
  - 3 Read image data
- Asprite stores bsprite info into several one-dimensional arrays.
- Take a look at Asprite.java and see it by yourself, it's not so complicated as it seems.

Bibliography

- **AuroraGT official repository**
  https://terminus.mdc.gameloft.org/vc/tools/AuroraGT
- **AuroraGT main wiki**
  https://wiki.gameloft.org/twiki/bin/view/Main/AuroraGT

Contact us

- Please, we look forward for any suggestions or bug found:
  - send us a mail to World-AuroraSuggestions@gameloft.com

Todo