Table of Contents

Overview

This program, formerly known as brandontools, represents a swiss army knife of useful utilities for developing homebrew applications on Nintendo's portable devices for now the GBA, NDS, and 3DS systems. My goal for this program is a simple command line interface that converts images, videos, animated images, and sounds into a format suitable for Nintendo's portable devices.

Note I will not go over how to draw anything here. I only explain the output formats and where does the data need to go to. TONC and GBATEK are good resources to cover everything else.

Setup

Linux (Manual/Automated compile)

You can run sudo ./setup.sh which does the tasks in the list below.

1. Ensure you have a C++ compiler (sudo apt-get install build-essential)

2. Install dependency wxWidgets 3.0 (sudo apt-get install libwxgtk3.0-dev)

3. Install dependency Magick++ (sudo apt-get install libmagick++-dev libmagickcore-dev libmagickwand-dev)

4. Build the program (make)

5. Install the program (sudo make install)

General Usage

The general usage of this program is as follows.

nin10kit -mode=**mode export_filename image_files**

Where:

mode is one of 0, 3, 4, palette, tiles, map, sprites, or lut.

export_filename is the filename to export to or a full path to the file to export to. If a directory is not specified then the files will be exported to the current working directory. This will create the files **export_filename**.c and **export_filename**.h

image_files is a list of paths to image files (either a path to an image file or a URL) to export.

The most common of image formats (bmp, jpeg, gif, png) should be supported. The formats supported by the program will depend on options was given to Magick++ when it was compiled.

TIP the image file paths can also use ImageMagick's read modifiers for a explanation of how to use these please refer to http://www.imagemagick.org/Usage/files/#read_mods

TIP To see a list of image formats this program supports run the following command after installing the program *nin10kit -help=formats*

Mode 3

To enable this mode pass -mode=3 for the mode parameter.

This mode will export an image as a 16 bpp bitmap.

Output

In the C file

> A single one dimensional array of type const unsigned short sized image_width * image_height for each image exported. Each element in the array corresponds to a pixel in the image, and pixels are stored in this array row by row. Each pixel represents an unsigned short laid out in the following bit pattern.

0	B	B	B	B	B	G	G	G	G	G	R	R	R	R	R

In the H file

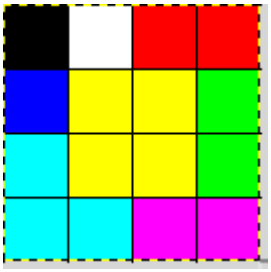
1. define for imagename_SIZE which contains the total size of the image in shorts.
2. define for imagename_WIDTH which contains the width of the image.
3. define for imagename_HEIGHT which contains the height of the image.
4. An extern for the array found in the C file

Example

name = example.png

width = 4

height = 4



sample command: *nin10kit -mode3 example example.png*

In the C file

```
const unsigned short example[16] = {
        0x0000, 0x7fff, 0x001f, 0x001f, // row 0
        0x7c00, 0x03ff, 0x03ff, 0x03e0, // row 1
        0x7fe0, 0x03ff, 0x03ff, 0x03e0, // row 2
        0x7fe0, 0x7fe0, 0x7c1f, 0x7c1f, // row 3
};
```

In the H file

```
extern const unsigned short example[16];
#define EXAMPLE_SIZE 16
#define EXAMPLE_WIDTH 4
#define EXAMPLE_HEIGHT 4
```

To get a pixel from the array given its position (x, y) / (r, c) you want to get you use the formula

y * width_of_image + x

or

r * width_of_image + c

LUTs

To enable this mode pass -mode=lut for the mode parameter.

This mode will export a look up table for a function. Look up tables speed up computation by precomputing values for a function and works like a hash table. For more information of its uses see: http://en.wikipedia.org/wiki/Lookup_table

To define a function to generate a look up table for you must use the –func parameter

--func=function_name,type,start,end,step[,in_degrees]

Where:

1. function_name: Name of the function to generate a lut for. (see Available functions below).
2. type: Input and output type of the look up table (see Specifying Types below).
3. start: Starting value as a real number in the look up table array
4. end: Ending value as a real number in the look up table array
5. step: The step between values in the look up table (should evenly divide (end-start).)
6. in_degrees: Optional and defaults to false. Values given are in degrees.

Available functions

Any function in C++'s cmath header (http://www.cplusplus.com/reference/cmath/) that takes in one input value and returns a value can be used to generate a look up table.

Specifying Types

There are two main types nin10kit supports integral types and fixed point types.

Integral types

nin10kit supports the following integral types:

1. char - an 8 bit value
2. byte - alias for char, an 8 bit value
3. short - a 16 bit value
4. int – a 32 bit value
5. long – a 64 bit value

Fixed point types

To specify a fixed point type the syntax is the following

integral_type.fixed_length

Where:

1. integral type is one of the types listed in Integral types above.
2. fixed_length is the number of bits dedicated to specifying the fractional portion of the value.

Examples

- byte.4 represents an 8 bit fixed point type whose bottom 4 bits are used to represent the fractional portion of the number. Minimum value is -8.9375, Maximum value is 7.9375. Incrementing the value adds 0.0625 to it.
- short.1 represents a 16 bit fixed point type whose least significant bit is used to represent the fractional portion of the number. One can only represent values in increments of 0.5 with this representation.

For more information on fixed point types refer to http://en.wikipedia.org/wiki/Fixed-point_arithmetic

Output

In the C file

> A single array of the appropriate type with (end – start) / step + 1 entries containing the value computed from the function entry 0 corresponds to func(start), entry 1 corresponds to func(start + step), entry array_size corresponds to func(end).

In the H file

1. An extern for the above array
2. #define name_SIZE – The size of the look up table
3. [If a fixed type was given] #define name_FIXED_LENGTH The length of the fractional portion
4. #define name_BEGIN The starting value expressed in the type given
5. #define name_END The ending value expressed in the type given
6. #define name_STEP The step between input values in the array

Mode 4

To enable this mode pass -mode=4 for the mode parameter.

This mode will export an image as a 8 bpp bitmap with a palette.

Requirements

All images exported with this mode must have a width evenly divisible by 2

Output

In the C file

1. An single array of at most size 256 containing the palette. The palette is an array of colors that are used in the image. If multiple images was passed in then you get a palette that can be used to display all of the images exported. The bit pattern for palette entries is the following.

0	B	B	B	B	B	G	G	G	G	G	R	R	R	R	R

2. A single one dimensional array of type const unsigned short sized image_width * image_height / 2 for each image exported. Each pixel in the image corresponds to an 8 bit value, an index into the palette, thus, each element in this array contains 2 pixels. The data in the image is stored row by row. The format for each entry in the array is the following. (*1)

Pixel 1 – 8 bits	Pixel 0 – 8 bits

In the H file

1. A single extern for the palette
2. define for export_filename_PALETTE_SIZE with the number of entries in the palette.
3. define for imagename_SIZE which contains the total size of the image in shorts.
4. define for imagename_WIDTH which contains the width of the image.
5. define for imagename_HEIGHT which contains the height of the image.
6. An extern for the array found in the C file

*(*1) You may be wondering why the first pixel is in the higher order bits its because the GBA is a little endian machine. Little endian stores bytes in memory in reverse of what you expect. In short, don't worry about it. The bits in memory would be the same as if it was a char array with the pixels stored sequentially. For more information on Endianness please refer to http://en.wikipedia.org/wiki/Endianness*

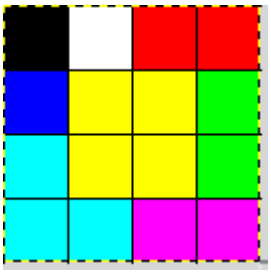
Example

export_filename = images

name = example

width = 4

height = 4



sample command: *nin10kit -mode4 images example.png*

In the C file **1*

```
const unsigned short images_palette[8] = {
     0x0000, 0x001f, 0x03e0, 0x03ff,
     0x7c00, 0x7c1f, 0x7fe0, 0x7fff,
};

const unsigned short example[8] = {
     0x0700, 0x0101, // row 0
     0x0304, 0x0203, // row 1
     0x0306, 0x0203, // row 2
     0x0606, 0x0505  // row 3
};
```

In the H file

```
extern const unsigned short images_palette[8];
#define IMAGES_PALETTE_SIZE 8

extern const unsigned short example[16];
#define EXAMPLE_SIZE 16
#define EXAMPLE_WIDTH 4
#define EXAMPLE_HEIGHT 4
```

Advanced Mode 4

Fine tune color allocation for multiple images.

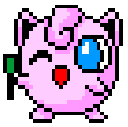
-palette=X Restricts the palette to X colors.

-start=X Offsets the palette by X

When using -start an extra define for name_PALETTE_OFFSET is given with this value. In addition if start is a value other than 0 the transparent color will not be the first entry in the palette.

Using these two options it makes it possible to allocate space for colors used in each image.

For example given these three images:



I can give 156 to dragonite (first image), 8 colors to jigglypuff (second image), and 100 to pikachu (third image) with the following commands

nin10kit -mode=4 -start=0 -palette=92 pikachu pikachu.png

nin10kit -mode=4 -start=92 -palette=8 jigglypuff jigglypuff.gif

nin10kit -mode=4 -start=100 -palette=156 dragonite dragonite.png

From this point you just simply copy the image's palettes one after the other into the palette memory.

Custom palettes

If you pass in an image (or set of images) via -palette_image this image will be treated as a file containing a palette (in the case of multiple images a palette that fits them all will be generated). Any images exported will be exported against the palette generated via palette_image. The resulting c and h files will only contain image arrays without a palette.

In addition to this to just export a palette for a given set of images pass -mode=palette instead of -mode=4 with this you can generate just a palette. Then you can pass -palette_image=files to export your images against your custom palette.

Note If your palette has a special transparent color then you must set -transparent in both your exported palette array and any images you export against the custom palette.

Custom palettes may also be used with tileset and sprite exporting.

Mode 0

To enable this mode pass -mode=0 for the mode parameter and specify bits per pixel with -bpp=4 or 8

This mode will export a map with a tileset and palette. If multiple maps are given you will get a global tileset and palette.

Requirements

All maps exported with this mode must be sized (256,256) (256x512) (512x256) or (512x512)

Among the maps there must not be more than 1024 unique tiles

Output

In the C file

1. An single array of at most size 256 containing the palette. The palette is an array of colors that are used in the tileset. If multiple maps were passed in then you get a palette that can be used to display all of the tiles exported. The bit pattern for entries in the palette is the following:

0	B	B	B	B	B	G	G	G	G	G	R	R	R	R	R

2. A single array of at most size 32768 containing the tileset. The tileset stores tiles whose size is either 8 or 16 shorts depending on bpp. Each tile contains indices into the palette (or palette bank if bpp=4). The bit pattern for entries in the tileset is the following (depending on bpp):

Pixel3 – 4 bits	Pixel2 – 4 bits	Pixel1 – 4 bits	Pixel0 - 4bits
Pixel1 – 8 bits		Pixel0 – 8 bits	

3. A single one dimensional array of type const unsigned short sized image_width * image_height for each map exported. Each element in this array contains a reference to what tile to draw, how to draw it, and what palette bank to use (for bpp = 4). The bit pattern for map entries is the following (PID - Palette bank id) (VF - Vertical flip) (HF - Horizontal flip) (TID - tile id). This array is arranged in sets of 32x32 tiles in row order.

PID	PID	PID	PID	VF	HF	TID	TID	TID	TID	TID	TID	TID	TID	TID	TID

In the H file

1. A single extern for the palette
2. A single extern for the tileset
3. define for export_filename_PALETTE_SIZE with the number of entries in the palette.
4. define for export_filename_PALETTE_TYPE containing the PALETTE TYPE flag which can be directly passed to the Background control register
5. define for export_filename_TILES with the number of tiles in the tileset
6. define for export_filename_TILES_SIZE with the size of the tileset in shorts
7. define for imagename_WIDTH which contains the width of the map in tiles
8. define for imagename_HEIGHT which contains the height of the map in tiles
9. define for imagename_MAP_SIZE which is width * height
10. define for imagename_MAP_TYPE which contains the MAP_SIZE TYPE flag which can be directly passed to the background control register
11. An extern for the map found in the C file

Example (4bpp)

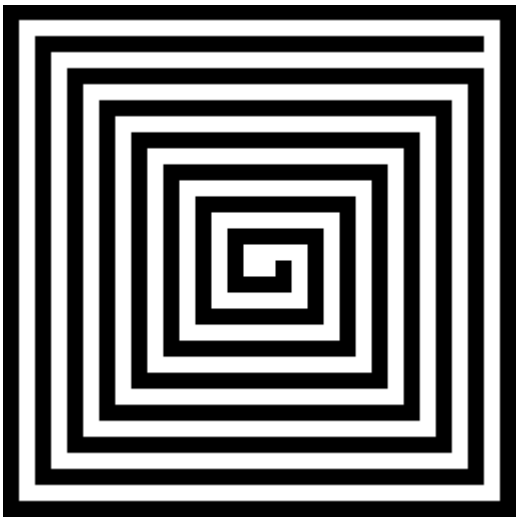
export_filename = maps

name = example

width = 256

height = 256

bpp = 4



sample command: *nin10kit -mode=0 -bpp=4 maps example.png*

In the C file (*1)

```
const unsigned short maps_palette[256] =
{
     0x0000, 0x7fff, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, // Palette bank 1
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, // Palette bank 2
     // continued for Palette banks 3-16.
};
```

*(*1) Palette size for 4bpp mode is always 256.*

```
const unsigned short maps_tiles[32] =
{
     0x0000, 0x0000, // Tile 0 row 0
     0x0000, 0x0000, // Tile 0 row 1
     0x0000, 0x0000, // Tile 0 row 2
     0x0000, 0x0000, // Tile 0 row 3
     0x0000, 0x0000, // Tile 0 row 4
     0x0000, 0x0000, // Tile 0 row 5
     0x0000, 0x0000, // Tile 0 row 6
     0x0000, 0x0000, // Tile 0 row 7
     0x1111, 0x1111, 0x1111, 0x1111, 0x1111, 0x1111, 0x1111,
     0x1111 // Tile 1
};

const unsigned short example[1024] =
{
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, // Row 0
     0x0001, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0001, // Row 1
     // etc.
}
```

In the H file

```
extern const unsigned short maps_palette[256];
#define MAPS_PALETTE_SIZE 256
extern const unsigned short maps_tiles[32];
#define MAPS_PALETTE_TYPE (0 << 7)
#define MAPS_TILES 2
#define MAPS_TILES_SIZE 32
extern const unsigned short example[1024];
#define EXAMPLE_WIDTH 32
#define EXAMPLE_HEIGHT 32
#define EXAMPLE_MAP_SIZE 1024
#define EXAMPLE_MAP_TYPE (0 << 14)
```

Example (8bpp)

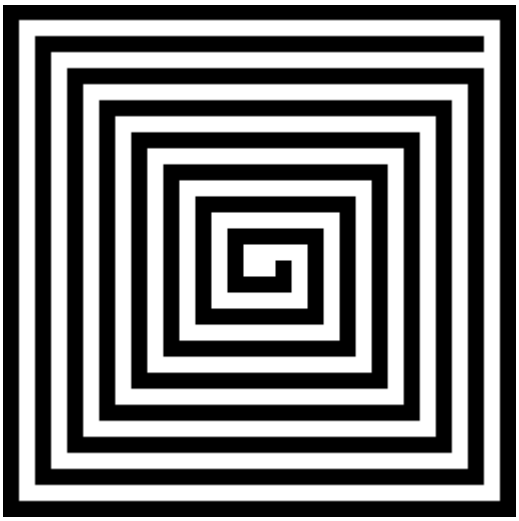
export_filename = maps

name = example.png

width = 256

height = 256

bpp = 8



sample command: *nin10kit -mode=0 -bpp=8 maps example.png*

In the C file

```
const unsigned short maps_palette[2] =
{
     0x0000, 0x7fff
};

const unsigned short maps_tiles[64] =
{
     0x0000, 0x0000, 0x0000, 0x0000, // Tile 0 row 0
     0x0000, 0x0000, 0x0000, 0x0000, // Tile 0 row 1
     0x0000, 0x0000, 0x0000, 0x0000, // Tile 0 row 2
     0x0000, 0x0000, 0x0000, 0x0000, // Tile 0 row 3
     0x0000, 0x0000, 0x0000, 0x0000, // Tile 0 row 4
     0x0000, 0x0000, 0x0000, 0x0000, // Tile 0 row 5
     0x0000, 0x0000, 0x0000, 0x0000, // Tile 0 row 6
     0x0000, 0x0000, 0x0000, 0x0000, // Tile 0 row 7
     0x1111, 0x1111, 0x1111, 0x1111, 0x1111, 0x1111, 0x1111,
     0x1111, 0x1111, 0x1111, 0x1111, 0x1111, 0x1111, 0x1111,
     0x1111, 0x1111 // Tile 1
};

const unsigned short example[1024] =
{
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, // Row 0
     0x0001, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
     0x0000, 0x0000, 0x0000, 0x0001, // Row 1
     // etc.
}
```

In the H file

```
extern const unsigned short maps_palette[2];
#define MAPS_PALETTE_SIZE 2
extern const unsigned short maps_tiles[64];
#define MAPS_PALETTE_TYPE (1 << 7)
#define MAPS_TILES 2
#define MAPS_TILES_SIZE 64
extern const unsigned short example[1024];
#define EXAMPLE_WIDTH 32
#define EXAMPLE_HEIGHT 32
#define EXAMPLE_MAP_SIZE 1024
#define EXAMPLE_MAP_TYPE (0 << 14)
```

Tiles

To enable this mode pass -mode=tiles for the mode parameter and specify bpp with -bpp=4 or 8

This mode will export a tileset and palette only. This is useful for dynamically generating maps as you know exactly which tile_id corresponds to what tile. In addition this should be a faster method than using -mode=0 directly at the expense of including tiles that aren't used in any map taking up space. Also you may use a custom palette to export your tiles see Custom Palette section under Advanced Mode 4.

Requirements

All tilesets exported with this mode must have dimensions divisible by 8.

There must not be more than 1024 unique tiles.

For best results the same tile must not appear in the set of tilesets given to the program

When using -mode=map you must pass in the tilesets in the same order as you have with this mode

Output

In the C file

1. An single array of at most size 256 containing the palette. The palette is an array of colors that are used in the tileset. If multiple tilesets was passed in then you get a palette that can be used to display all of the tiles exported. The bit pattern for entries in the palette is the following:

0	B	B	B	B	B	G	G	G	G	G	R	R	R	R	R

2. A single array of at most size 32768 containing the tileset. The tileset stores tiles whose size is either 8 or 16 shorts depending on bpp. Each tile contains indices into the palette (or palette bank if bpp=4). The bit pattern for entries in the tileset is the following (depending on bpp):

Pixel3 – 4 bits	Pixel2 – 4 bits	Pixel1 – 4 bits	Pixel0 - 4bits
Pixel1 – 8 bits		Pixel0 – 8 bits	

In the H file

1. A single extern for the palette
2. A single extern for the tileset
3. define for export_filename_PALETTE_SIZE with the number of entries in the palette.
4. define for export_filename_PALETTE_TYPE containing the PALETTE TYPE flag which can be directly passed to the Background control register
5. define for export_filename_TILES with the number of tiles in the tileset
6. define for export_filename_TILES_SIZE with the size of the tileset in shorts

Example (4 bpp)

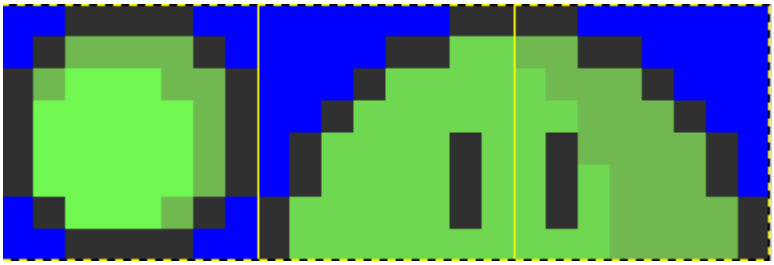
export_filename = mario

name = mario.png

width = 24

height = 8

transparent=0000FF



Sample command: *nin10kit -mode=tiles -bpp=4 -transparent=0000FF mario mario.png*

In the C file *1

```
const unsigned short mario_palette[256] =
{
     0x7c00,0x18a5,0x2aed,0x2bed,0x274d,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb 0
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb 1
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb 2
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb 3
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb 4
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb 5
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb 6
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb 7
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb 8
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb 9
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb A
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb B
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb C
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb D
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb E
     0x7c00,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000  // pb F
};
```

**1 Palette size for 4bpp mode is always 256, if a transparent color is given it will appear first in all palette banks*

```
const unsigned short mario_tiles[64] =
{
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, //t 0
     0x1100,0x0011,0x2210,0x0122,0x3321,0x1223,0x3331,0x1233,
     0x3331,0x1233,0x3331,0x1233,0x3310,0x0123,0x1100,0x0011, //t 1
     0x0011,0x0000,0x1122,0x0000,0x2224,0x0001,0x2244,0x0012,
     0x2214,0x0122,0x2414,0x0122,0x2414,0x1222,0x2444,0x1222, //t 2
     0x0000,0x1100,0x0000,0x4411,0x1000,0x4444,0x4100,0x4444,
     0x4410,0x4144,0x4410,0x4144,0x4441,0x4144,0x4441,0x4444, //t 3
};
```

In the H file

```
     #define MARIO_TRANSPARENT 0x00
     extern const unsigned short mario_palette[256];
     #define MARIO_PALETTE_SIZE 256
     #define MARIO_PALETTE_TYPE (0 << 7)
     extern const unsigned short mario_tiles[64];
     #define MARIO_TILES 4
     #define MARIO_TILES_SIZE 64
```

Example (8 bpp)

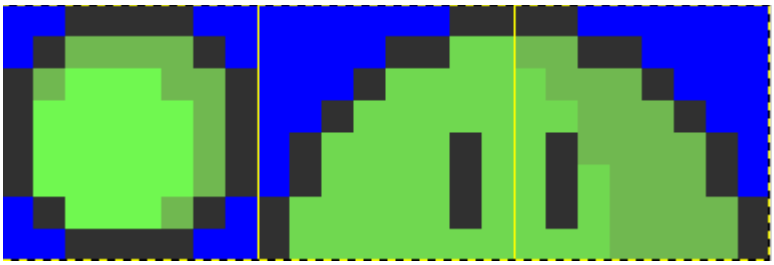
export_filename = mario

name = mario

width = 24

height = 8

transparent=0000FF



Sample command: *nin10kit -mode=tiles -bpp=8 -transparent=0000FF mario mario.png*

In the C file

```
const unsigned short mario_palette[5] =
{
     0x7c00,0x18a5,0x2aed,0x274d,0x2bed
};

const unsigned short mario_tiles[128] =
{
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, //t 0
     0x0000,0x0101,0x0101,0x0000,0x0100,0x0202,0x0202,0x0001,
     0x0201,0x0404,0x0204,0x0102,0x0401,0x0404,0x0404,0x0102,
     0x0401,0x0404,0x0404,0x0102,0x0401,0x0404,0x0404,0x0102,
     0x0100,0x0404,0x0204,0x0001,0x0000,0x0101,0x0101,0x0000, //t 1
     0x0000,0x0000,0x0000,0x0101,0x0000,0x0000,0x0101,0x0303,
     0x0000,0x0100,0x0303,0x0303,0x0000,0x0301,0x0303,0x0303,
     0x0100,0x0303,0x0303,0x0301,0x0100,0x0303,0x0303,0x0301,
     0x0301,0x0303,0x0303,0x0301,0x0301,0x0303,0x0303,0x0303, //t 2
     0x0101,0x0000,0x0000,0x0000,0x0202,0x0101,0x0000,0x0000,
     0x0203,0x0202,0x0001,0x0000,0x0303,0x0202,0x0102,0x0000,
     0x0103,0x0202,0x0202,0x0001,0x0103,0x0203,0x0202,0x0001,
     0x0103,0x0203,0x0202,0x0102,0x0303,0x0203,0x0202,0x0102  //t 3
};
```

In the H file

```
     #define MARIO_TRANSPARENT 0x00
     extern const unsigned short mario_palette[5];
     #define MARIO_PALETTE_SIZE 5
     #define MARIO_PALETTE_TYPE (1 << 7)
     extern const unsigned short mario_tiles[128];
     #define MARIO_TILES 4
     #define MARIO_TILES_SIZE 128
```

Map

To enable this mode pass -mode=map for the mode parameter and specify bpp with -bpp=4 or 8

This mode will export a map only. It is used in combination with the tiles mode to separate out tileset information from map information. When using this mode you must also provide the tilesets the map is using with the tileset parameter. The tilesets must be specified in the same exact order as when they was exported with tiles mode with the same bpp option.

Requirements

All maps exported with this mode must have dimensions equal to one of (256, 256), (512, 256), (256, 512) or (512, 512).

All tiles used in the maps exported must be used in the tileset.

Output

In the C file

1. A single one dimensional array of type const unsigned short sized image_width * image_height for each map exported. Each element in this array contains a reference to what tile to draw, how to draw it, and what palette bank to use (for bpp = 4). The bit pattern for map entries is the following (PID - Palette bank id) (VF - Vertical flip) (HF - Horizontal flip) (TID - tile id). This array is arranged in sets of 32x32 tiles in row order.

PID	PID	PID	PID	VF	HF	TID	TID	TID	TID	TID	TID	TID	TID	TID	TID

In the H file

1. define for imagename_WIDTH which contains the width of the map in tiles
2. define for imagename_HEIGHT which contains the height of the map in tiles
3. define for imagename_MAP_SIZE which is width * height
4. define for imagename_MAP_TYPE which contains the MAP_SIZE TYPE flag which can be directly passed to the background control register
5. An extern for the map found in the C file

Example (4 bpp)

export_filename = map

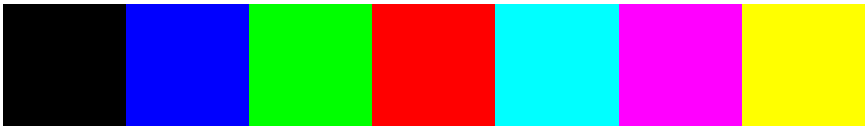
name = map.png

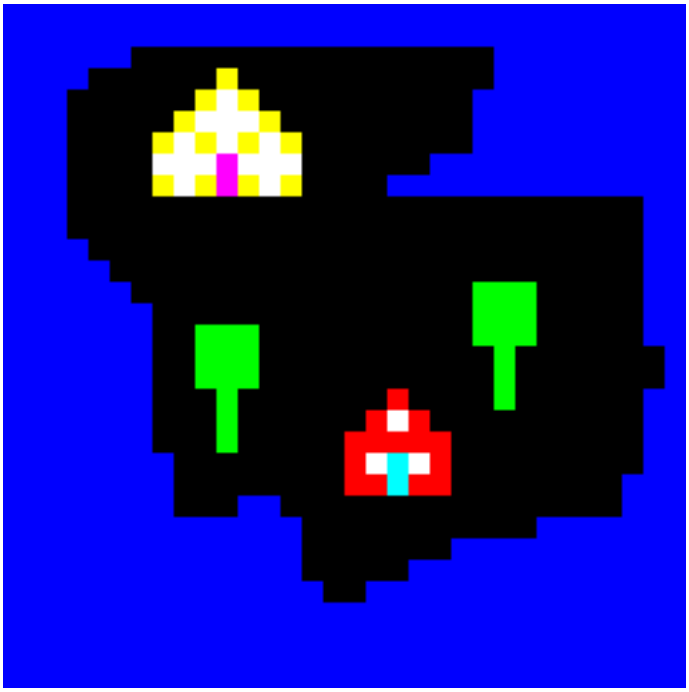
tileset = simple.png

width = 256

height = 256

simple.png





Sample command: *nin10kit -mode=map -bpp=4 -tileset_image=simple.png map map.png*

In the C file

```
const unsigned short map[1024] =
{
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 0
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 1
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 2
	0x0001,0x0001,0x0001,0x0001,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0006,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 3
	0x0001,0x0001,0x0001,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0006,0x0007,0x0006,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 4
	0x0001,0x0001,0x0001,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0006,0x0007,0x0007,0x0007,0x0006,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 5
	0x0001,0x0001,0x0001,0x0000,0x0000,0x0000,0x0000,0x0006,
	0x0007,0x0006,0x0007,0x0006,0x0007,0x0006,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 6
	0x0001,0x0001,0x0001,0x0000,0x0000,0x0000,0x0000,0x0007,
	0x0007,0x0007,0x0005,0x0007,0x0007,0x0007,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 7
	0x0001,0x0001,0x0001,0x0000,0x0000,0x0000,0x0000,0x0006,
	0x0007,0x0006,0x0005,0x0006,0x0007,0x0006,0x0000,0x0000,
	0x0000,0x0000,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 8
	0x0001,0x0001,0x0001,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001, // 9
	0x0001,0x0001,0x0001,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001, // 10
	0x0001,0x0001,0x0001,0x0001,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001, // 11
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0000,0x0000,0x0000,
```

```
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001, // 12
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0002,0x0002,
0x0002,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001, // 13
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0002,0x0002,
0x0002,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001, // 14
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0000,
0x0000,0x0002,0x0002,0x0002,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0002,0x0002,
0x0002,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001, // 15
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0000,
0x0000,0x0002,0x0002,0x0002,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0002,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001, // 16
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0000,
0x0000,0x0002,0x0002,0x0002,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0002,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001, // 17
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0000,
0x0000,0x0000,0x0002,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0004,0x0000,0x0000,0x0000,0x0000,0x0002,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001, // 18
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0000,
0x0000,0x0000,0x0002,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0004,0x0007,0x0004,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001, // 19
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0000,
0x0000,0x0000,0x0002,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0004,0x0004,0x0004,0x0004,0x0004,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001, // 20
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0004,0x0007,0x0003,0x0007,0x0004,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001, // 21
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0004,0x0004,0x0003,0x0004,0x0004,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001,0x0001, // 22
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
0x0000,0x0000,0x0000,0x0001,0x0001,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001,0x0001, // 23
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0000,0x0000,
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
0x0000,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 24
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0000,0x0000,
```

```
	0x0000,0x0000,0x0000,0x0000,0x0000,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 25
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0000,0x0000,
	0x0000,0x0000,0x0000,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 26
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0000,
	0x0000,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 27
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 28
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 29
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001, // 30
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
	0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001  // 31
};
```

In the H file

```
extern const unsigned short map[1024];
#define MAP_WIDTH 32
#define MAP_HEIGHT 32
#define MAP_MAP_SIZE 1024
#define MAP_MAP_TYPE (0 << 14)
```

The output in 8bpp mode is exactly the same given the parameters so I omit it heree.

Sprites

To enable this mode pass -mode=sprites for the mode parameter and specify bpp with -bpp=4 or 8

This mode will export a palette and sprites.

Also you may use a custom palette to export your sprites see Custom Palette section under Advanced Mode 4.

Requirements

All images given to be turned into sprites must have one of these dimensions (8,8) (16,16) (32,32) (64,64) (8,16) (16,8) (8,32) (32,8) (16,32) (32,16) (32,64) (64,32)

With 4 bpp mode the total area of your sprites must not exceed 256x256 or 1024 tiles

With 8 bpp mode the total area of your sprites must not exceed 128x256 or 512 tiles

For the purposes of this readme I will not go into 2D mode as there are no benefits when using it over 1D mode.

Output

In the C file

1. An single array of at most size 256 containing the palette. The palette is an array of colors that are used in the sprite images. The bit pattern for entries in the palette is the following:

0	B	B	B	B	B	G	G	G	G	G	R	R	R	R	R

2. A single array of at most size 16384 containing the sprite graphics. The sprite graphics are tiles whose size is either 8 or 16 shorts depending on bpp. Each tile contains indices into the palette (or palette bank if bpp=4). The bit pattern for entries in the sprite graphics is the following (depending on bpp):

Pixel3 – 4 bits	Pixel2 – 4 bits	Pixel1 – 4 bits	Pixel0 - 4bits
Pixel1 – 8 bits		Pixel0 – 8 bits	

In the H file

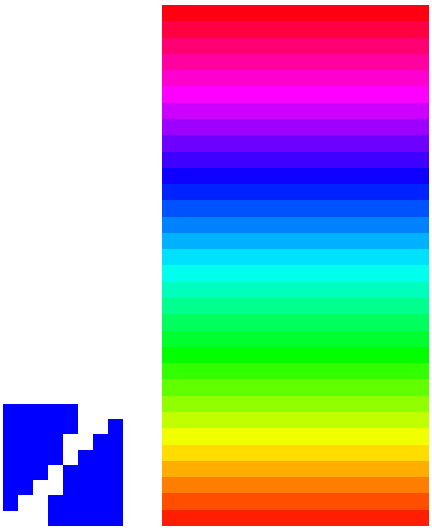
1. a single extern for the palette
2. define for export_filename_PALETTE_SIZE containing the number of entries in the palette
3. a single extern for the sprite graphics array
4. define for export_filename_PALETTE_TYPE which contains the PALETTE_TYPE flag to be passed into OAM attribute 0
5. define for export_filename_DIMENSION_TYPE which contains the DIMENSION_TYPE flag to be passed into the display control register.
6. define for export_filename_SIZE which contains the size of the sprite graphics array
7. define for imagename_PALETTE which contains the bank the sprite is using (bpp = 4 only) This can be passed into OAM attribute 2
8. define for imagename_SPRITE_SHAPE which contains the shape flag of the sprite. This can be passed into OAM attribute 0
9. define for imagename_SPRITE_SIZE which contains the size flag of the sprite. This can be passed into OAM attribute 1
10. define for imagename_ID which contains the proper id into the sprite graphics array where the sprites tiles live. This can be passed into OAM attribute 2

Example (4 bpp)

export_filename = sprites

image1 = tiny1x1 (8, 8)

image2 = rainbow2x4 (16, 32)



Sample command: *nin10kit -mode=sprites -bpp=4 sprites tiny1x1.png rainbow2x4.png*

In the C file

```
const unsigned short sprites_palette[256] =
{
     0x0000,0x7803,0x7c60,0x7c12,0x101f,0x401e,0x701e,0x007e,
     0x7d80,0x017e,0x7b40,0x13c0,0x03c5,0x43c0,0x02fe,0x03f7, // pb 0
     0x0000,0x7c00,0x7fde,0x0000,0x0000,0x0000,0x0000,0x0000,
     0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000, // pb 1
     // palette banks 2 - 15 omitted
}

const unsigned short sprites[144] =
{
     // rainbow2x4
     0x4444,0x4444,0x4444,0x4444,0x5555,0x5555,0x5555,0x5555,
     0x6666,0x6666,0x6666,0x6666,0x6666,0x6666,0x3333,0x3333, //t 0
     0x4444,0x4444,0x4444,0x4444,0x5555,0x5555,0x5555,0x5555,
     0x6666,0x6666,0x6666,0x6666,0x6666,0x6666,0x3333,0x3333, //t 1
     0x3333,0x3333,0x1111,0x1111,0x1111,0x1111,0x2222,0x2222,
     0x8888,0x8888,0x8888,0x8888,0xaaaa,0xaaaa,0xaaaa,0xaaaa, //t 2
     0x3333,0x3333,0x1111,0x1111,0x1111,0x1111,0x2222,0x2222,
     0x8888,0x8888,0x8888,0x8888,0xaaaa,0xaaaa,0xaaaa,0xaaaa, //t 3
     0xaaaa,0xaaaa,0xdddd,0xdddd,0xdddd,0xdddd,0xdddd,0xdddd,
     0xbbbb,0xbbbb,0xcccc,0xcccc,0xcccc,0xcccc,0xcccc,0xcccc, //t 4
     0xaaaa,0xaaaa,0xdddd,0xdddd,0xdddd,0xdddd,0xdddd,0xdddd,
     0xbbbb,0xbbbb,0xcccc,0xcccc,0xcccc,0xcccc,0xcccc,0xcccc, //t 5
     0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xeeee,0xeeee,
     0xeeee,0xeeee,0x9999,0x9999,0x9999,0x9999,0x7777,0x7777, //t 6
     0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xeeee,0xeeee,
     0xeeee,0xeeee,0x9999,0x9999,0x9999,0x9999,0x7777,0x7777, //t 7
     // tiny1x1
     0x1111,0x2221,0x1111,0x1221,0x1111,0x1122,0x1111,0x1112,
     0x2111,0x1111,0x2211,0x1111,0x1221,0x1111,0x1222,0x1111
};
```

In the H file

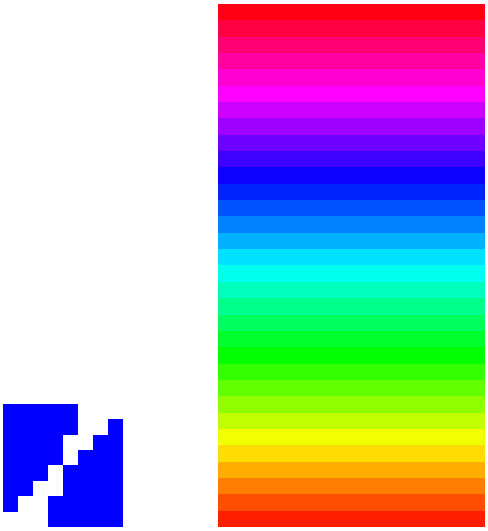
```
#define SPRITES_PALETTE_TYPE (0 << 13)
#define SPRITES_DIMENSION_TYPE (1 << 6)

extern const unsigned short sprites_palette[256];
#define SPRITES_PALETTE_SIZE 256

extern const unsigned short sprites[144];
#define SPRITES_SIZE 144

#define RAINBOW2X4_PALETTE (0 << 12)
#define RAINBOW2X4_SPRITE_SHAPE (2 << 14)
#define RAINBOW2X4_SPRITE_SIZE (2 << 14)
#define RAINBOW2X4_ID 0

#define TINY1X1_PALETTE (1 << 12)
#define TINY1X1_SPRITE_SHAPE (0 << 14)
#define TINY1X1_SPRITE_SIZE (0 << 14)
#define TINY1X1_ID 8
```

Example (8 bpp)

export_filename = sprites

image1 = tiny1x1 (8, 8)

image2 = rainbow2x4 (16, 32)



Sample command: *nin10kit -mode=sprites -bpp=8 sprites tiny1x1.png rainbow2x4.png*

In the C file

```
const unsigned short sprites_palette[35] =
{
     0x0000,0x7800,0x7c00,0x7806,0x7c60,0x780c,0x7c12,0x041e,
     0x201e,0x341e,0x7c18,0x501e,0x641e,0x007e,0x7c1e,0x7d40,
     0x011e,0x7de0,0x01de,0x7ea0,0x02be,0x03c0,0x7b60,0x13c0,
     0x2fc0,0x5bc0,0x03e5,0x47e0,0x77c0,0x03cb,0x035e,0x03d1,
     0x03f7,0x03dd,0x7fde
};
```

```
const unsigned short sprites[288] = {
     // tiny1x1
     0x0202,0x0202,0x2202,0x2222,0x0202,0x0202,0x2202,0x0222,
     0x0202,0x0202,0x2222,0x0202,0x0202,0x0202,0x0222,0x0202,
     0x0202,0x2202,0x0202,0x0202,0x0202,0x2222,0x0202,0x0202,
     0x2202,0x0222,0x0202,0x0202,0x2222,0x0222,0x0202,0x0202,
     // rainbow2x4
     0x0707,0x0707,0x0707,0x0707,0x0808,0x0808,0x0808,0x0808,
     0x0909,0x0909,0x0909,0x0909,0x0b0b,0x0b0b,0x0b0b,0x0b0b,
     0x0c0c,0x0c0c,0x0c0c,0x0c0c,0x0e0e,0x0e0e,0x0e0e,0x0e0e,
     0x0a0a,0x0a0a,0x0a0a,0x0a0a,0x0606,0x0606,0x0606,0x0606,
     0x0707,0x0707,0x0707,0x0707,0x0808,0x0808,0x0808,0x0808,
     0x0909,0x0909,0x0909,0x0909,0x0b0b,0x0b0b,0x0b0b,0x0b0b,
     0x0c0c,0x0c0c,0x0c0c,0x0c0c,0x0e0e,0x0e0e,0x0e0e,0x0e0e,
     0x0a0a,0x0a0a,0x0a0a,0x0a0a,0x0606,0x0606,0x0606,0x0606,
     0x0505,0x0505,0x0505,0x0505,0x0303,0x0303,0x0303,0x0303,
     0x0202,0x0202,0x0202,0x0202,0x0404,0x0404,0x0404,0x0404,
     0x0f0f,0x0f0f,0x0f0f,0x0f0f,0x1111,0x1111,0x1111,0x1111,
     0x1313,0x1313,0x1313,0x1313,0x1616,0x1616,0x1616,0x1616,
     0x0505,0x0505,0x0505,0x0505,0x0303,0x0303,0x0303,0x0303,
     0x0202,0x0202,0x0202,0x0202,0x0404,0x0404,0x0404,0x0404,
     0x0f0f,0x0f0f,0x0f0f,0x0f0f,0x1111,0x1111,0x1111,0x1111,
     0x1313,0x1313,0x1313,0x1313,0x1616,0x1616,0x1616,0x1616,
     0x1c1c,0x1c1c,0x1c1c,0x1c1c,0x1919,0x1919,0x1919,0x1919,
     0x1b1b,0x1b1b,0x1b1b,0x1b1b,0x1818,0x1818,0x1818,0x1818,
     0x1717,0x1717,0x1717,0x1717,0x1515,0x1515,0x1515,0x1515,
     0x1a1a,0x1a1a,0x1a1a,0x1a1a,0x1d1d,0x1d1d,0x1d1d,0x1d1d,
     0x1c1c,0x1c1c,0x1c1c,0x1c1c,0x1919,0x1919,0x1919,0x1919,
     0x1b1b,0x1b1b,0x1b1b,0x1b1b,0x1818,0x1818,0x1818,0x1818,
     0x1717,0x1717,0x1717,0x1717,0x1515,0x1515,0x1515,0x1515,
     0x1a1a,0x1a1a,0x1a1a,0x1a1a,0x1d1d,0x1d1d,0x1d1d,0x1d1d,
     0x1f1f,0x1f1f,0x1f1f,0x1f1f,0x2020,0x2020,0x2020,0x2020,
     0x2121,0x2121,0x2121,0x2121,0x1e1e,0x1e1e,0x1e1e,0x1e1e,
     0x1414,0x1414,0x1414,0x1414,0x1212,0x1212,0x1212,0x1212,
     0x1010,0x1010,0x1010,0x1010,0x0d0d,0x0d0d,0x0d0d,0x0d0d,
     0x1f1f,0x1f1f,0x1f1f,0x1f1f,0x2020,0x2020,0x2020,0x2020,
     0x2121,0x2121,0x2121,0x2121,0x1e1e,0x1e1e,0x1e1e,0x1e1e,
     0x1414,0x1414,0x1414,0x1414,0x1212,0x1212,0x1212,0x1212,
     0x1010,0x1010,0x1010,0x1010,0x0d0d,0x0d0d,0x0d0d,0x0d0d
};
```

In the H file

```
#define SPRITES_PALETTE_TYPE (1 << 13)
#define SPRITES_DIMENSION_TYPE (1 << 6)

extern const unsigned short sprites_palette[35];
#define SPRITES_PALETTE_SIZE 35

extern const unsigned short sprites[288];
#define SPRITES_SIZE 288

#define TINY1X1_SPRITE_SHAPE (0 << 14)
#define TINY1X1_SPRITE_SIZE (0 << 14)
#define TINY1X1_ID 0

#define RAINBOW2X4_SPRITE_SHAPE (2 << 14)
#define RAINBOW2X4_SPRITE_SIZE (2 << 14)
#define RAINBOW2X4_ID 2
```

Animation

If you pass in an image file type that is animated or a video format (examples: gif, pdf, avi) then you will get an additional array of appropriate type with pointers to all of the frames in the image.

In summary the additional things will appear if you pass in an image with multiple frames

In the C file

- An array of pointers (and in the case of sprites sprite IDs)
- Each image will be named <image_name><frame>

In the H file

- Common declarations (such as size, width and height) are declared per image and not per frame.
- Additional extern for animation frames array
- define for imagename_FRAMES with the number of frames found in the image

Sprites & Mode 3

GBA sprites can also be used in mode 3. The only caveat is that you can only use half of the sprite graphics memory here as one of the characterblocks is taken up by the memory used to represent the videobuffer. In addition all sprite IDs must be offset by 512 as you have to use characterblock 6,

To at least offset the sprite IDs and include error checking for exporting too much sprite data you can export your sprites normally with -mode=sprites and pass in the extra flag -for_bitmap=1

So in summary

1. You only get a total area of 256x128 for bpp=4 and 128x128 for bpp=8
2. Sprite IDs must be offset 512
3. Pass in -for_bitmap=1 to offset the sprite IDs and error out if you exported too much data.

Miscellany

Generating individual data in batch mode for mode0 and mode 4

Command line option -split=1 allows you to treat all images as if they were exported one at a time using the program. That is...

In Mode 4 each image will get its own palette of 256 colors.

In Mode 0 each map will get its own palette and own tileset.

In any other mode this option has no effect.

Resizing images

Command line option -resize allows you to resize individual images the format of this is -resize=resize_str1,resize_str2

where resize_str1 is of the format widthxheight ex. 10x12

If specified on the command line it must be given for all images. You can however not specify a string to leave the corresponding image alone

Example

-resize=10x12,,72x42

The first image will be resized to width 10 height 12.

Second image is not resized.

Third image is resized to width 72 height 42.

Renaming image arrays

Command line option -names allows you to rename the symbol names for arrays generated. This is useful if the original image filename can't be represented as a variable in C.

Using it has the same restrictions as -resize above. If you specify -names then a name must be given to ALL images you export; however, unlike -resize you can't skip elements.

Credits

Me (Brandon) for the code
Magick++ developers for Magick++
The authors at http://en.literateprograms.org/Median_cut_algorithm_(C_Plus_Plus)?action=history&offset=20080309133934 for their median cut algorithm which I slightly modified.
http://www.compuphase.com/riemer.htm for dithering algorithm.