gimp-msx-plugins Documentation

Release 0.1

Thies Hecker

Oct 03, 2020

Contents:

CHAPTER 1

Introduction

gimp-msx-plugins is a collection of plug-ins for the GNU Image Manipulation Program (GIMP),which help to create graphics for the MSX (1) home computer. The plug-ins included in the collection are:

- MSX GMII Exporter - exports background images in MSX Graphics Mode II compatible binary files (pattern, name and color tables)
- MSX Sprite Exporter - exports MSX(1) compatible sprites (binary data for sprite pattern and sprite attribute tables)

gimp-msx-plugins is licensed under GPLv3.

The source code can be found at:

https://gitlab.com/thecker/gimp-msx-plugins

CHAPTER 2

Install instructions

2.1 Binary versions

2.1.1 MS Windows

The easiest way to get the plug-ins for MS Windows (64bit) is to use a binary version, which can be found here:

https://www.mediafire.com/folder/mm09nocktntaz/

You will just need to follow the instructions in the README.txt provided in the zip-file.

2.1.2 GNU/linux

There are no binary packages for linux yet. But it should be no issue to compile the plug-ins from source on any recent linux distribution - see instructions below.

Note On some linux distros (like *Arch Linux*) the python support has been dropped in the official *GIMP* package! Some of the plug-ins are written in python and thus might not be working in such case.

For Arch Linux there is fortunately an AUR for *GIMP* with python2 support - see: https://aur.archlinux.org/packages/python2-gimp/

2.2 Building from source

2.2.1 Prerequisites

The GIMP plug-ins are written partly in C and partly in python.

To compile the plug-ins written in C you will need following libraries installed:

- gimp >= 2.2.0 (libgimp2.0)
- cairo

- glib2 (libglib2.0)
- gtk2 (libgtk2.0)

For using the python plug-ins you should have python 2.7.x installed. Linux distros should normally include it by default.

2.2.2 Obtain the source code

The source code for the *gimp-msx-plugins* can be found at:

https://gitlab.com/thecker/gimp-msx-plugins

The best way to get latest the source code is to clone the repo:

```
git clone https://gitlab.com/thecker/gimp-msx-plugins.git
```

or download the source code in form of a gzipped tarball or zip file.

2.2.3 Compiling under GNU/linux

Compiling under linux should be straight forward (autotools). cd into the parent directory and run the configure script:

```
./configure
```

If you add *–prefix=$HOME* option the configure script will try to determine your user's local GIMP folder (does not require root privileges to install) - so below is the recommended way.

```
./configure --prefix=$HOME
```

Note: You should have launched GIMP at least once before for the current user to make sure, that the local GIMP folders are created.

To compile and install type:

```
make && make install
```

Note: If configure was not run with *–prefix=$HOME* you'll probably need to run the install command with root privileges (e.g. *sudo make install*).

2.2.4 Compiling under MS Windows

Since the GIMP libraries are developed with the GNU toolchain, the easiest way to compile the source under MS Windows is probably to install MSYS2.

MSYS2 provides the MinGW (minimalist GNU for Windows - e.g. C compiler) and a suite of POSIX tools. And it also provides a pre-compiled GIMP 2.10 package for the 64-bit version - so make sure, you select the *msys2-x86_64-...* installer!

For perparing your build environment please follow the instruction on the GIMP wiki in the "Setting up a devel environment" section.

(Actually you only need to install the packages *base-devel*, *git* and *mingw-w64-x86_64-toolchain*, the rest will be installed automatically in the next step)

After that you can just install the GIMP 2.10 package:

```
pacman -S mingw-w64-x86_64-gimp
```

Before compiling the *gimp-msx-plugins* you will most likely have to set up some environment variables so that the C-compiler (*gcc*) and *libgimp* is found by the configure script.

```
export PATH=$PATH:/mingw64/bin
export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/mingw64/lib/pkgconfig
```

You can also add these lines to the `.bash_profile` file in your *MSYS2* home directory to make them defined on start-up.

After that you should be able to compile the plug-ins in *MSYS2*:

```
git clone https://gitlab.com/thecker/gimp-msx-plugins.git
cd gimp-msx-plugins
./configure
make
```

You can now find the compiled plug-in files in the *gimp-msx-plugins\src* folder under your MSYS home folder.

The location of the MSYS home folder depends on the your install - if you chose the default install location it will probably be something like:

C:\msys64\home\<your_user_name>\

Copy all .exe and .py files in the *src* folder to the plug-ins folder of your Windows installation of GIMP.

You can find out, where the GIMP folders are located, if you launch GIMP and go to the *Preferences* menu in GIMP:

Edit –> Preferences –> Folders –> Plug-Ins

You will also need to copy the *palettes* to the corresponding folders of your GIMP installation.

The palettes (.gpl files) can be found in *gimp-msx-plugins\data*. Check the location of the palettes folder analogous to the plug-ins folder.

Edit –> Preferences –> Folders –> Palettes

After you have copied the plug-ins and palettes, you should be able to use the *gimp-msx-plugins* after you restart GIMP.

CHAPTER 3

User guide

This section explains the usage of the plug-ins.

All plug-ins can be accessed in GIMP via the main menu:

Filters –> Misc –> MSX...

Below sections describe the usage of the individual plug-ins.

3.1 MSX GMII Exporter

The GMII exporter can convert (background) images to MSX graphics mode II / SCREEN 2 compatible formats. The plug-in can export the data into the (binary) format used by MSX's video display processor - i.e. a representation of the relevant VRAM sections.

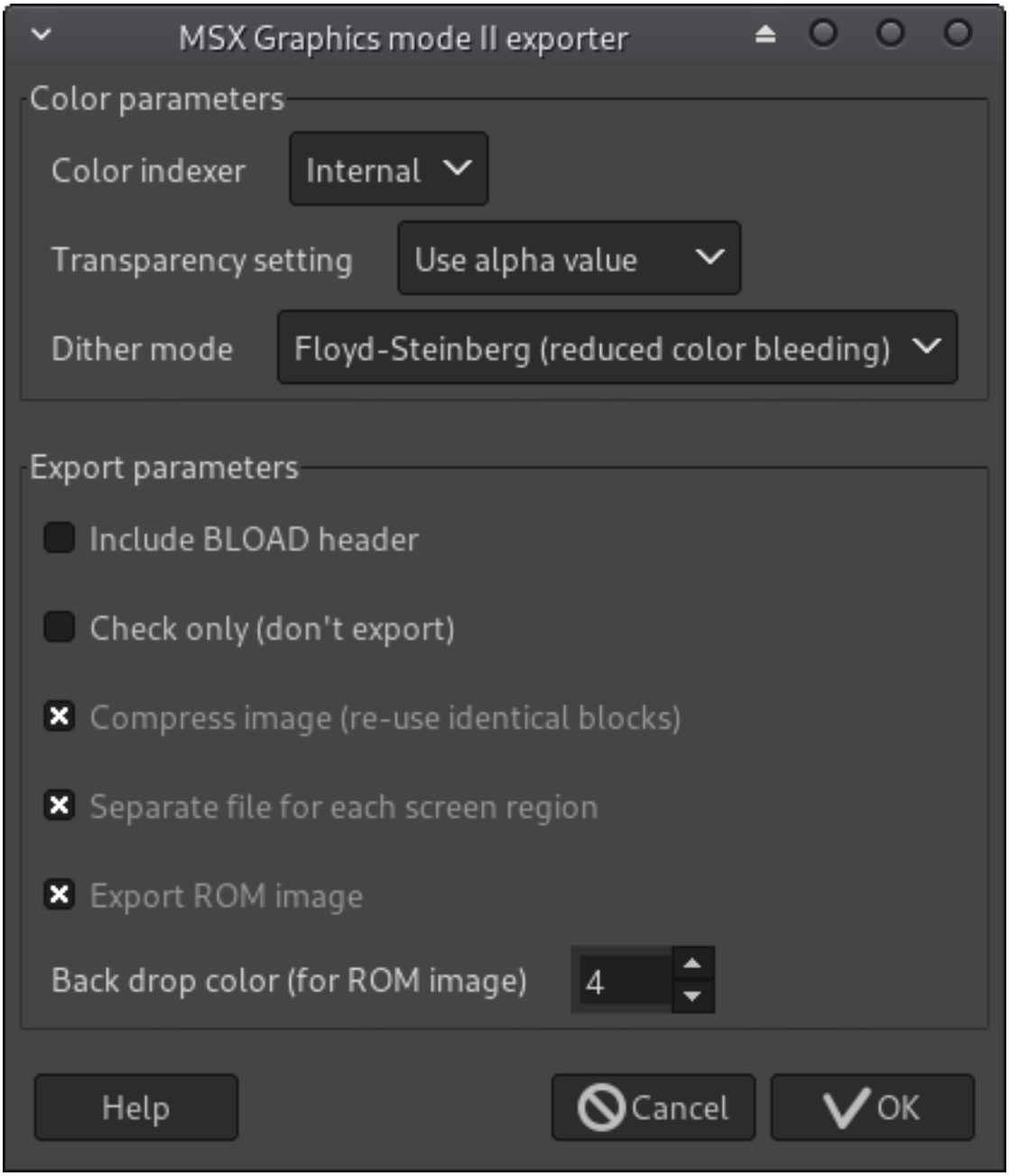
3.1.1 Using the plug-in

The GMII Exporter can be found in the GIMP main menu under:

Filters –> Misc –> MSX GMII Exporter

To use the exporter just load an image and launch the plug-in.

When launching the plug-in an dialog with the exporter options menu will pop-up:



The options are explained below.

To run the export just click "OK". The output files will be stored in the same folder as the image file.

The exporter will automatically index the image[1] to the MSX's color palette and convert pixels, which are not valid in Graphics mode II (i.e. in each row of an 8x8 pixel block only two different colors may be used).

A new layer with the converted image will be added.

Below example illustrates the process on a picture of the GIMP mascot "Wilber" (by Jakub Steiner, CC BY-SA 3.0):

[1] You can also manually do the indexing. If the plug-in is launched on an already indexed image it will not re-index it. But note, that the index number should match the MSX1 color numbers.

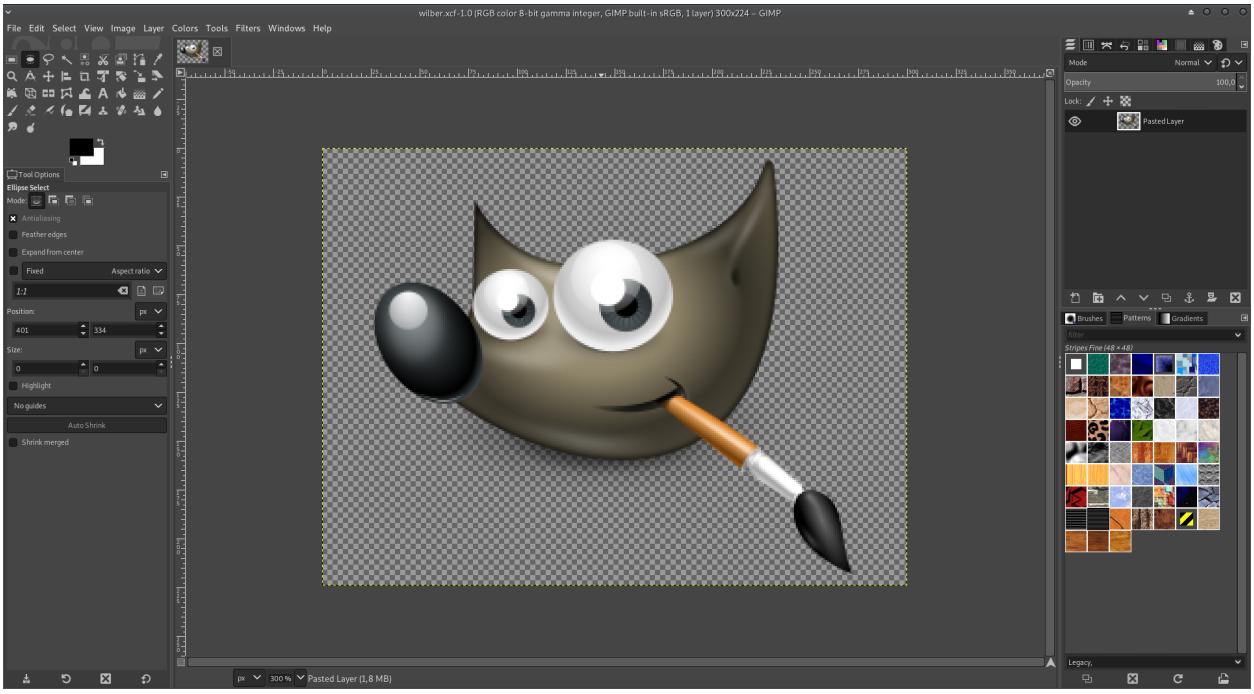


Fig. 1: Original image (RGB)

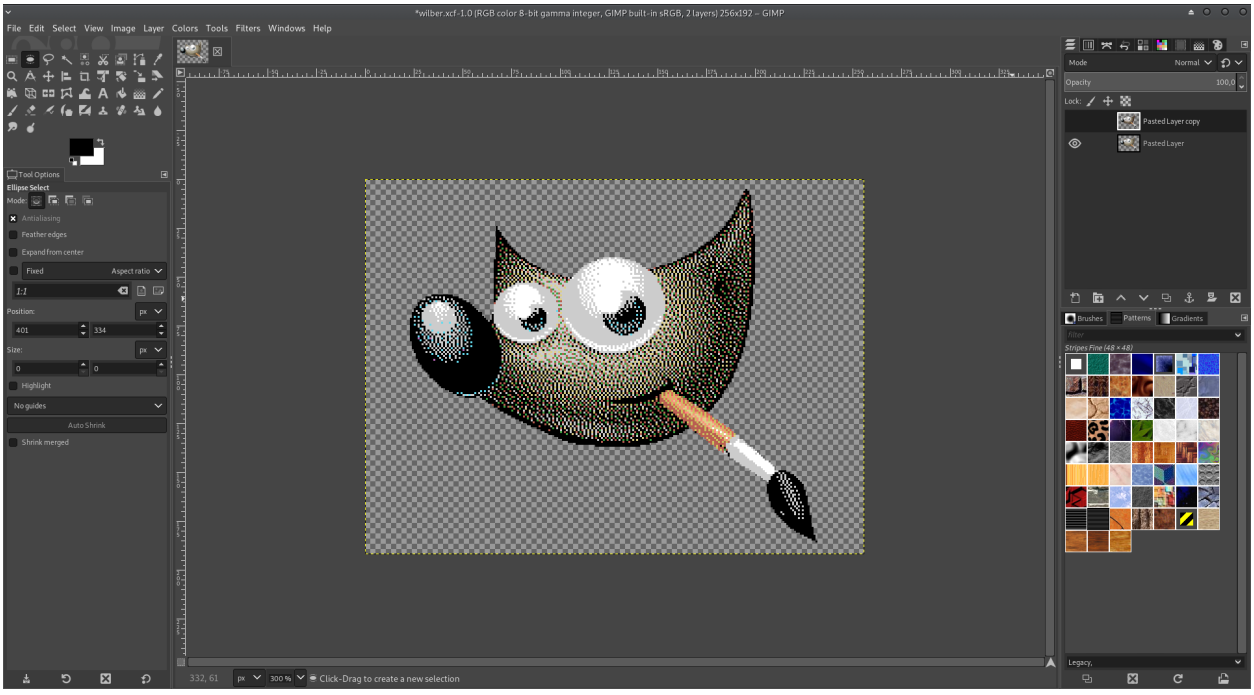


Fig. 2: Image converted to the MSX's color palette (Indexed)

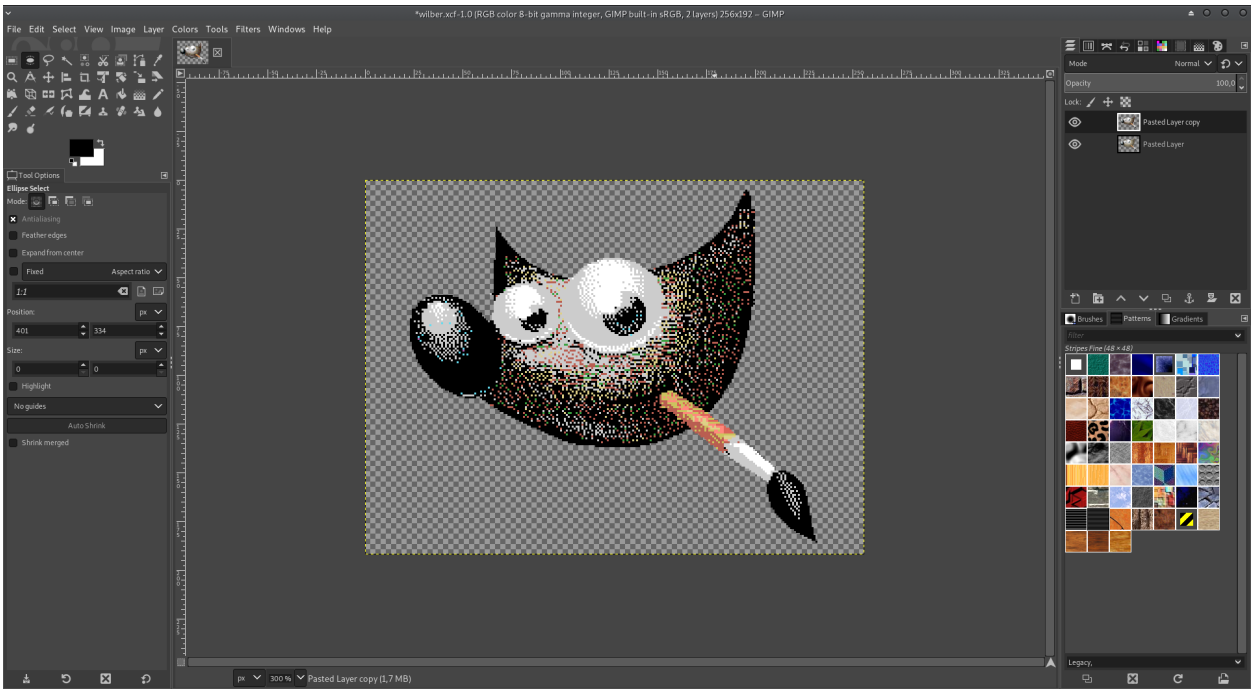


Fig. 3: Image corrected for MSX Graphics mode 2 limitations

3.1.2 Export options

Color indexer You may select either the GIMP's "Internal" indexer (which uses the built-in gimp-image-convert-indexed procedure by Spencer Kimball & Peter Mattis) or the *MSXize plug-in* by Weber Estevan Roder Kai. Note, that you need to install MSXize separately - checkout the MSXize project gitlab page.

Transparency setting Provides different options how transparent pixels (MSX color 0) shall be identified. Possible values are:

Use alpha value Treats all pixels with an alpha value of 0 as transparent

Use color #ff00eb All pixels of RGB color #ff00eb will be treated as transparent

No transparency Image will only be indexed to the 15 real colors of the MSX palette

Note, that pixels of alpha value = 0, will also be considered as transparent, if the other two options are selected.

Dither mode If you are using the internal indexer you can provide the dither type parameter using this checkbox.

Include BLOAD header If selected, a 7 byte header is prepended to the exported binary files. MSX basic's BLOAD can then be used to load the file contents into RAM/VRAM.

Below is an example MSX-basic script to setup the VDP and load the name, pattern and color table to the corresponding VRAM locations.

```
10 'reg 0 - screen 2
20 VDP(0) = &H02
30 '.reg 1 - 16k, 16x16 sprites, mag. on
40 VDP(1) = &HE3
50 '.reg 2 - name table at &h3800
60 VDP(2) = &HE
70 '.reg 3 - color table at &h2000
80 VDP(3) = &HFF
```

(continues on next page)

(continued from previous page)

```
90 '.reg 4 - pattern table at &h0000
100 VDP(4) = &H03
110 '.reg 5 - sprite attributes table at &h3B00
120 VDP(5) = &H76
130 '.reg 6 - sprite pattern table at &h1800
140 VDP(6) = &H03
150 '.reg 7 - backdrop color = green
160 VDP(7) = &H02
170 '.....end of VPD setup...
180 BLOAD"wilber.nt",S,&h3800
190 BLOAD"wilber.pt",S,&h0000
200 BLOAD"wilber.ct",S,&h2000
210 'end of code
```

Check only (don't export) If selected the image will be checked for violations of graphics mode II (e.g. more than two colors in a row of 8x8 pixel block). A new layer will be added to the image with the invalid rows highlighted in red.

Compress image (re-use identical blocks) If selected identical blocks will only be saved once in the pattern and color tables thus reducing the table sizes. The name tables will be generated accordingly.

If not selected each block is mapped to the corresponding position in the name tables (i.e. name tables simply contain all numbers counting from 0 to 255).

Seperate file for each screen region If selected, the exporter will write the tables for each of the 3 screen regions (top, mid, bottom) into seperate files. I.e. using this option the exporter will generate 9 binary files, which have the name of the image file appended by: .pt1, .pt2, .pt3, .ct1, .ct2, .ct3, .nt1, .nt2 and .nt3 (where "pt" files hold the pattern generator table, "ct" the color table and "nt" the name table).

If this option is not selected a single file will be exported for the pattern generator table (.pt), the color table (.ct) and the name table.

Export ROM image If selected a 16k ROM image will be exported. The file will have the same name as the image file appended by ".rom". The ROM contains a simple program, which displays the image. Useful for testing the image output on real hardware or an emulator - like in the screen shot below.

3.2 MSX Sprite exporter

The sprite exporter can export (composited) sprites to the MSX(1) sprite pattern and sprite attribute tables. Your image dimension have to be multiples of the sprite size (i.e. either 8 or 16 px). Although the size and number of colors is not limited by the plug-in, keep in mind, that the MSX's VDP only supports 32 active (monochrome) sprites. And only 4 sprites can be shown per scan line. The plug-in will raise respective warnings.

3.2.1 Usage

If an image is active, the plug-in can be launched from the *Filters* menu (*Filters –> Misc –> MSX sprite exporter*).

A dialog with plug-in option will appear (see explanation of options below).

When you click OK the plug-in will analyze the image and export the sprite data. The filenames either end with .sat (for the sprite attribute table) or .spt (for the sprite pattern table). Depending on the plug-in options and individual sprites in the image one or more .sat and .spt files will be exported.

Below example shows a 32x32 pixel 2-color (black and grey) sprite example.

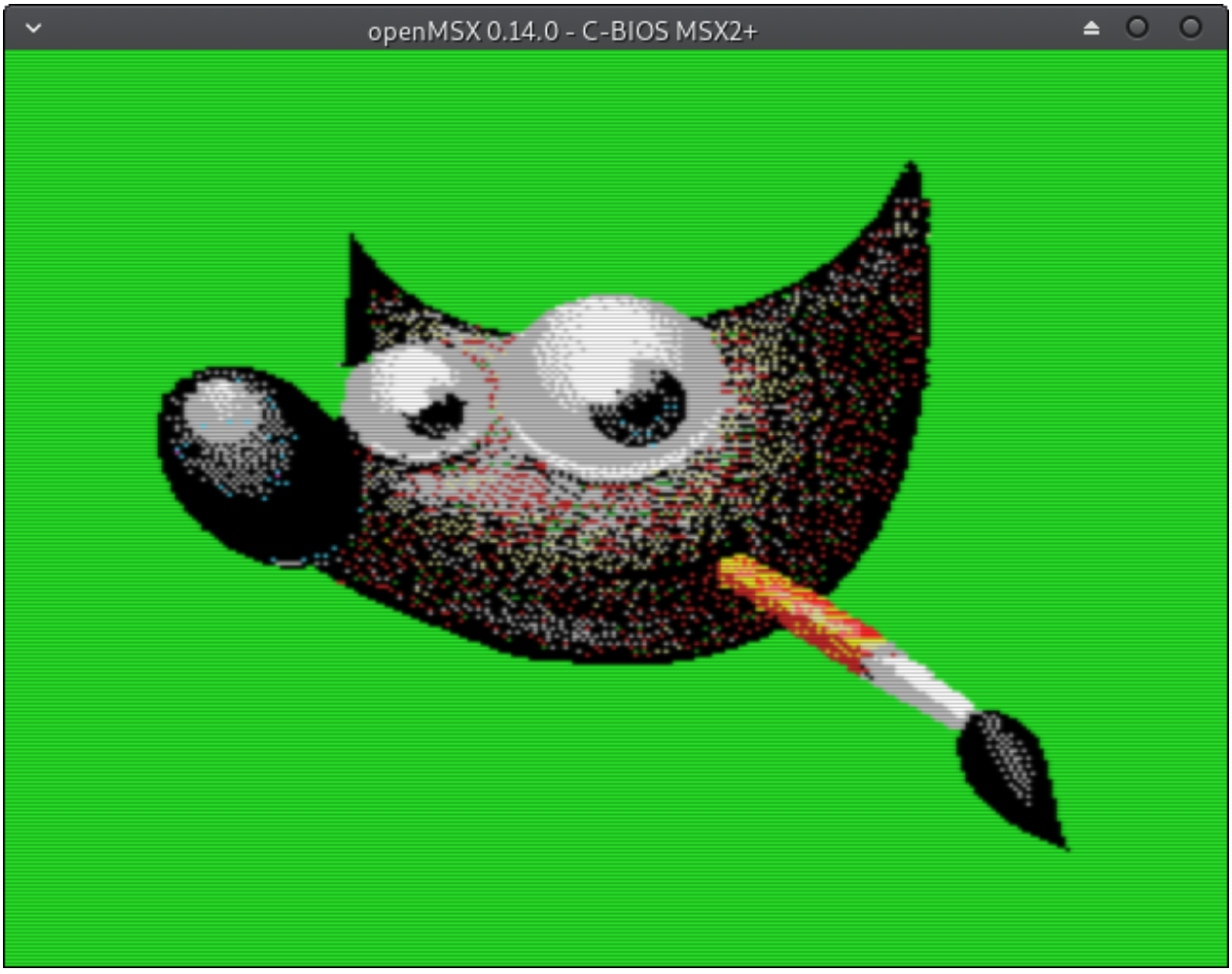


Fig. 4: Screen shot of openMSX running the test ROM image for the example above

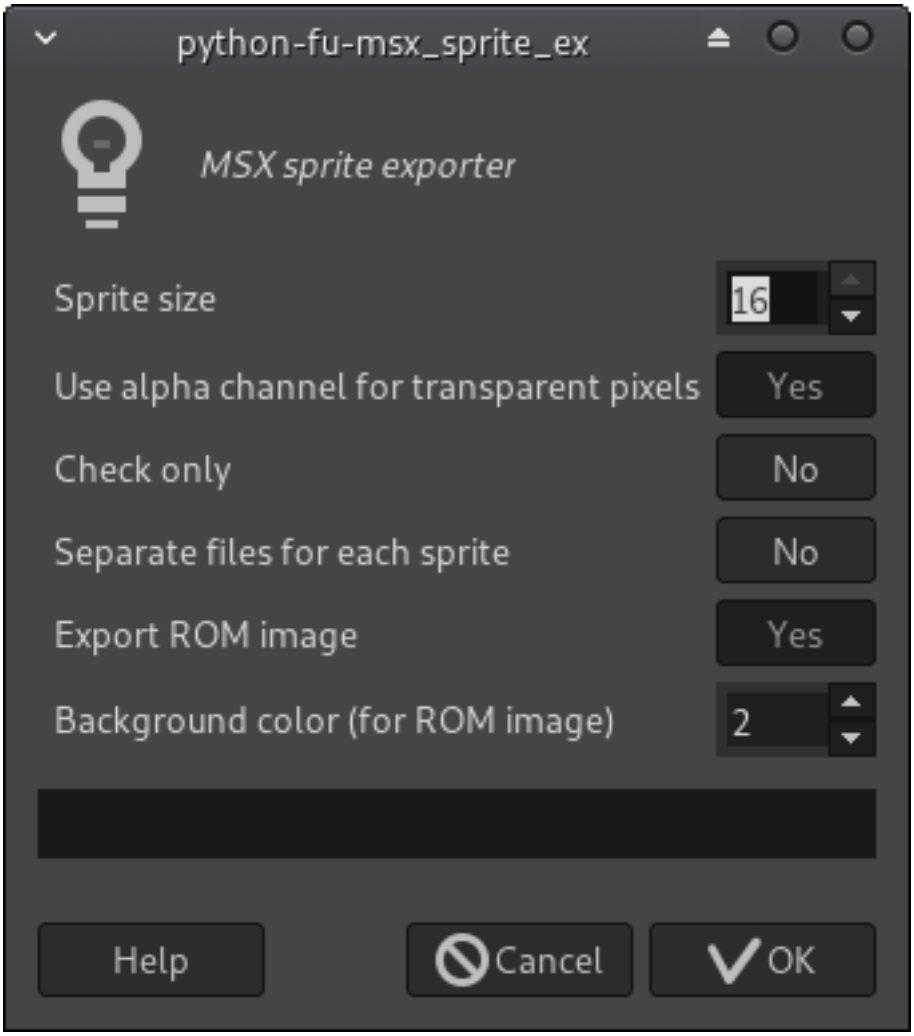


Fig. 5: Sprite exporter options dialog

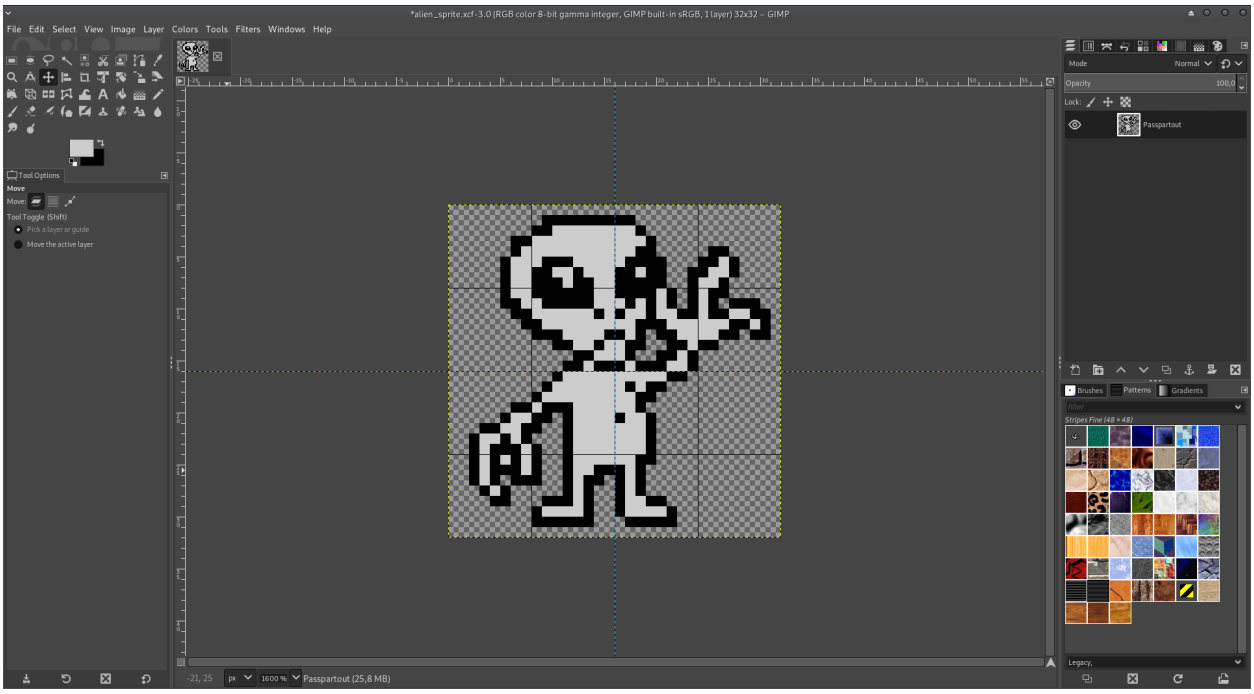


Fig. 6: 32 x 32 px 2-color sprite (with transparent background)

Using the default export options the plug-in will export the image into 8 sprites: 1 sprite for each of the two colors in each of the four 16x16 quarters (max. MSX (single) sprite size). The data will be saved into one .sat and one .spt file containing the attribute and pattern information of all 8 sprites.

3.2.2 Plug-in options

Sprite size Can be set to both possible MSX sprite size values - 8 or 16. I.e. either 8x8 px or 16x16 px sprites will be extracted.

Use alpha channel for transparent pixels If this option is selected, the transparent pixels will be identified by an alpha value of 0 (i.e. transparent like the background in the example above). If this option is not selected, transparent pixels have be indicated by a color value of #ff00eb - like shown in the example below:

Check only If selected, the plug-in does not export any files (just performs the checks)

Separate files for each sprite If selected individual .sat and .spt files are exported for each sprite. The files will be named . . . _01.sat, . . . 02.sat, etc. - otherwise all sprites are stored in one .sat and one .spt file

Export ROM image If selected a ROM image with a test program showing the sprite will be exported (additionally to the .sat and .spt files). This can be used for testing the exported output - below image shows the test program for the sprite example above.

Background color (for ROM image) Defines the background color by one of the 16 MSX color values. (Color 2 - green is used in the example above)

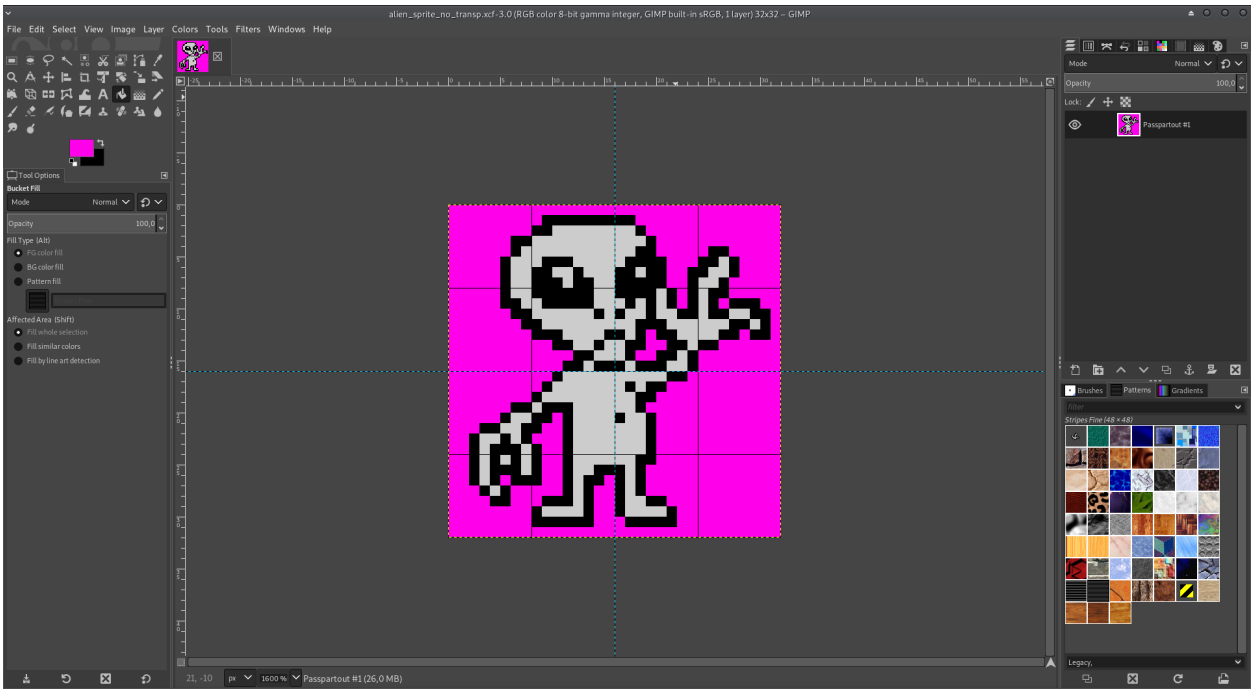


Fig. 7: 32 x 32 px 2-color sprite (with transparent pixels indicated by color #ff00eb)

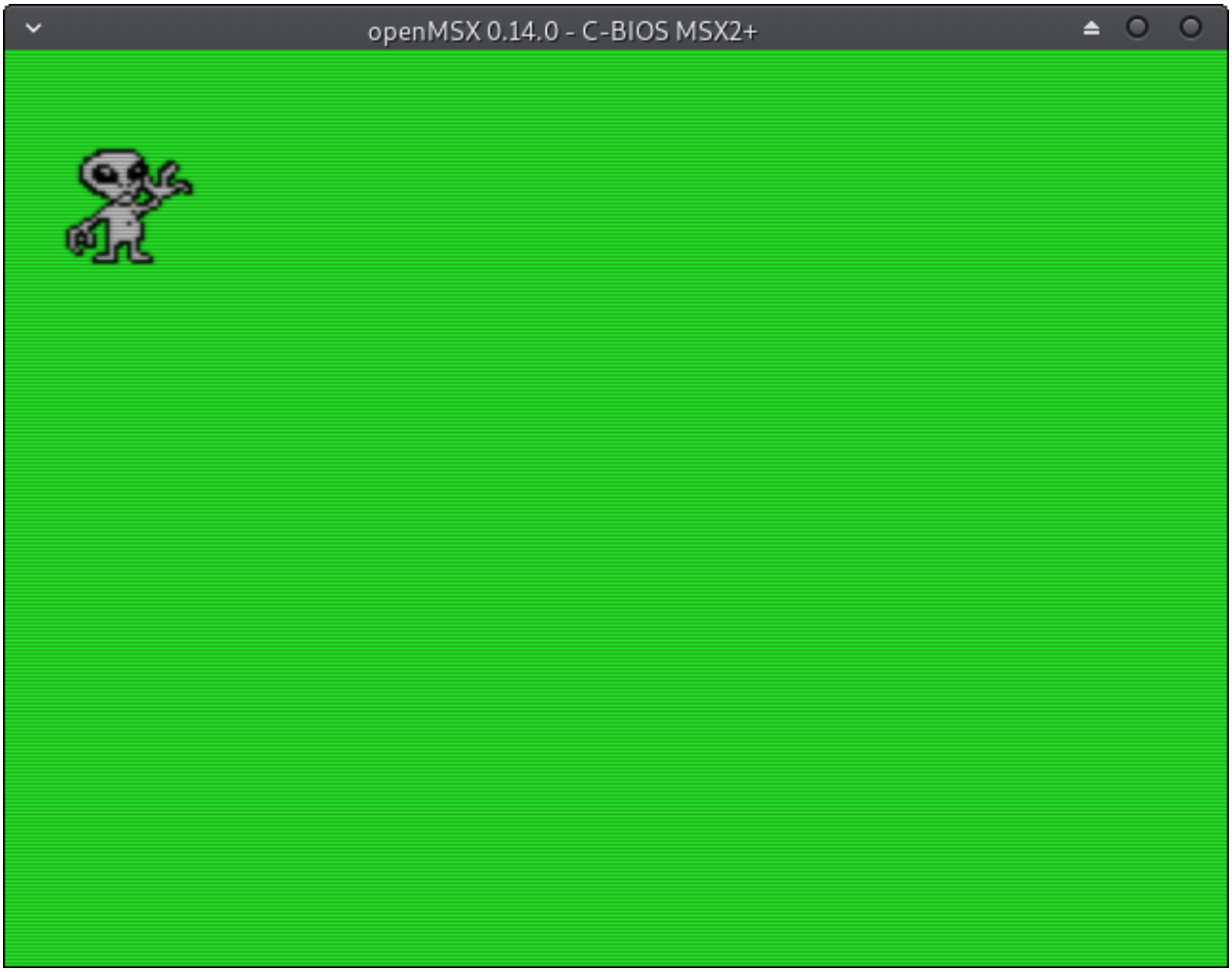


Fig. 8: openMSX showing the 32 x 32 px 2-color sprite example

CHAPTER 4

Developer information

This section shall give a basic overview for everyone, who would like to help improving the *gimp-msx-plugins*.

4.1 Submitting bugs and feature requests

If you observe a bug in the software or if you want to post a feature request, please use the issue tracker:

https://gitlab.com/thecker/gimp-msx-plugins/issues

4.2 The source code

The source code can be found at:

https://gitlab.com/thecker/gimp-msx-plugins

A brief overview of the repository's folders:

- doc/ - documentation sources (uses sphinx)
- **src/ - source code**
  - msx_gm2_exporter (written in C)
  - msx_sprite_exporter (written in python)
  - .asm files for ROM image export function op-code
- data/ - palette files required for indexing to MSX color palette

4.3 API reference

TBD

CHAPTER 5

Change log

5.1 release 0.3.1

Date: 2020-10-03

- Sprite exporter [bugfix]: Disabling "Use alpha for transparent pixels" on images without alpha channel crash fixed
- Sprite exporter [bugfix]: Error messages & warnings were not rendered to screen
- GMII exporter: Gimp 2.8 compatibility fix

5.2 release 0.3.0

Date: 2019-11-23

- GMII exporter: automatic image scaling & resizing added
- GMII exporter: BLOAD headers can now be added using compress & split files option
- GMII exporter: Transparency handling improved (alpha channel now supported)
- GMII exporter: Background color for ROM image can be defined
- palettes adjusted (based on Paul Wratt's HW-MSX palette)

5.3 release 0.2.0

Date: 2019-11-20

- fixed bug #5: sprite exporter raises IOError on Win7
- fixed bug #1/#4: GM II exporter exported wrong colors

- GM II exporter: new color indexing features (select dither type and option to use external converter MSXize by Weber Estevan Roder Kai)

5.4 release 0.1.3

- bugfix in GM II exporter (color identification failed for some blocks)
- sprite exporter added
- GM II exporter some code refactoring

5.5 release 0.1.2

- missing files for configure/make added + system compatibility improved

5.6 release 0.1.1

- fixed bug #2: GM II exporter: compress image does not work properly

CHAPTER 6

Road map

The roadmap has been moved to the project's gitlab site:

https://gitlab.com/thecker/gimp-msx-plugins/-/milestones

Please add requests for enhancements or bug fixes on gitlab as well:

https://gitlab.com/thecker/gimp-msx-plugins/issues

CHAPTER 7

Indices and tables

- genindex
- modindex
- search