palutils user manual

Title	palutils (Image and palette utilities for embedded systems design).
Author	Nikolaos Kavvadias 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014
Contact	nikos@nkavvadias.com
Website	http://www.nkavvadias.com
Release Date	24 February 2014
Version	1.0.0
Rev. history	
v1.0.0	2014-02-24 Converted documentation to RestructuredText format. Cleaned up directory organization and removed generated files. Upgraded to version 1.0.0.
v0.0.6	2009-03-19 Added "-manhattan" and "-chessboard" options for calculating the Manhattan and chessboard distances (non- Euclidean geometries) of greyscale images. Considered experimental.
v0.0.5	2009-03-11 Added "-epx" option for "Eric's Pixel eXpansion" image scaling filter.
v0.0.4	2008-12-13 Added "-c27" option (27-color image format).
v0.0.3	2008-08-27 Documentation (README) fixes.
v0.0.2	2008-03-10 First public release.

1. Introduction

`palutils` (palette and image utilities) is a collection of handy tools for processing of image files (mainly 8-bit indexed palette bitmaps). Its main focus is on the usage of image files in small-scale embedded systems.

Currently, the `palutils` collection consists of the following programs:

bmp2any translates a BMP file to various formats such as a hex dump or an ANSI C unsigned int array.

memgen generates a VHDL file of a memory initialized to the BMP image data. The VHDL memory can be mapped to a block RAM (automatically inferred in most modern Xilinx FPGA devices).

genpltenc generates all 256-color palette encodings that can be represented using only the index encodings relations (of the image data). The number of r (red), g (green) and b (blue) levels of color are associated by the equation: rlev x glev x blev = 256. Acceptable values for the number of levels are: {2, 4, 8, 16, 32, 64}. There exist 21 unique subsets of this form.

genpalette generates palettes from a predefined set of selections.

palutils the homonymous program provides several generation, conversion, translation and dumping facilities for 8-bit indexed palette and 256-level greyscale bitmap (BMP) image files.

2. File listing

The `palutils` distribution includes the following files:

/palutils	Top-level directory
/doc	Documentation directory
AUTHORS	List of `palutils` authors.
BUGS	Bug list.
ChangeLog	A log for code changes.
COPYING	The GPL, version 2, which governs `palutils`
NEWS	Development news for the `palutils` project.
README	This file.
README.html	HTML version of README.
README.pdf	PDF version of README.
rst2docs.sh	Bash script for generating the HTML and PDF versions.
THANKS	Acknowledgements.
TODO	A list of future enhancements.
VERSION	Current version of the project sources.
/scripts	Scripts directory
binopgen.sh	Generate greyscale images using the generating function: `f(x,y) = x binop y`
bmpembed.sh	Initialize a VHDL ROM/RAM memory from a BMP file.
chaosgen.sh	Generate pseudo-chaotic black-and-white images based on a trick function by Donald E. Knuth (TAOCP, Vol. 4, Bitwise Tricks and Techniques).
dump2bmp.sh	Convert an image dump file to a Windows bitmap (BMP).
dumptrad2bmp.sh	Convert a "traditional" image dump file (using either the C8 or the C27 format to a BMP file.

fractalgen.sh	Generate fractal images.
imgprocess.sh	Generate pseudocolor images for 256-level greyscale images.
lenseffect.sh	Produce "lens-distorted" image.
make_char_rom.sh	Create a font character ROM.
negeffect.sh	Produce "negative" image.
roteffect.sh	Generate pseudocolor images by applying a rotation effect.
rwalkgen.sh	Generate random walk images.
sepiaeffect.sh	Apply the "Sepia" filtering effect on 256-color palette images.
/src	Source code directory
bmp2any.c	Bitmap file converter to a number of output formats.
genpalette.c	Generates palettes from a predefined selection set. The generated palettes are compatible with GIMP and mtpaint.
genpltenc.c	Generates all 256-color palette encodings that can be represented using only the index encodings relations (of the image data).
memgen.c	Program/data memory generator (Block-RAM based).
palutils.c	Image and palette utilities for embedded systems design.
/tests	Main test files directory
/tests/bitmaps	Included bitmap (BMP) files for testing
`*.bmp`	Various BMP images.
/tests/bitmaps	Included bitmap (BMP) files for testing
`*.bmp`	Various BMP images.
/tests/fonts	Font collection as BMP files
`*.bmp`	The OEM, S8x16 and XL fonts as BMP files.
/tests/hexdumps	Hex dumps generated from the font BMP files
`*.txt`	Hexadecimal dumps for the OEM, S8x16 and XL fonts.
/tests/imgdumps	Image ASCII dump files
`*.txt`	ASCII dump files from raster images, mostly in the C8 and C27 custom formats.
/tests/palettes	Palette files
`*.txt`	Sample GPL files for GIMP and mtpaint.

3. Palutils usage

The palutils program can be invoked with several options (see complete options listing below). The usual tasks that can be accomplished with palutils are:

- change a palette to a given BMP file (e.g. for the purpose of pseudocolorization).

- normalize BMP image data to a given or a predefined palette.
- export a BMP file from a VHDL memory dump (a listing of consecutive memory entries encoded either with characters or plain numbers).

`palutils` can be invoked as:

```
$./palutils [options]
```

The complete `palutils` options listing:

-h Print this help.

-d Enable debug output.

-q Enable quiet run. (unimplemented)

-r Replace an imported or BMP file palette.

-k Keep the existing palette of an imported BMP file.

-noplt Donnot use palette; encode instead an RGB value (R:3-bits,G:3-bits,B:2-bits).

-<char><num> Convert bitmap data to use a palette. Valid values for char: c (color), g (grey). Valid values for num: {8,256}.

-lev-<chroma> <num> Indicates that the chromatic components are to be subsampled (when creating the palette) by a power-of-2 factor. All three factors myst be specified, otherwise the defaults {4,8,8} are used. The product of the three factors must yield 256. Valid values of chroma: {r,g,b}. Valid integer factors: {1,2,3,4,5,6}.

-balance Balance the values of the chromatic components so that min (0x00) and max (0xFF) are always included.

-colorize Colorize a greyscale image with colors from a specified palette.

-vhdl Dump generated palette to VHDL ROM.

-inst-<vendor> Instantiate vendor-specific resources for implementing the palette ROM. Choices: {xilinx}. (unimplemented)

-u <file> Import VHDL image data dump.

-v<mode> Read image data dump values as in <mode>. Value options: <mode>={hex (default),int}.

-trad Accept traditional format of 8/27-color VHDL image data dump.

-x <num> Read X-dimension size of input image.

-y <num> Read Y-dimension size of input image.

-s <num> Factor for scaling the output image.

-i <file> Import palette data from file.

-o <file> Export palette data to file.

-b <file> Read bitmap from file.

-e <file> Write bitmap to file.

-cmap <file> Write the colormap (as in NetPBM parlance) to a BMP. The colormap depicts the palette of a 256-color bitmap.

Here follow some simple usage examples of `palutils`.

1. Generate a 256-level greyscale BMP file from a VHDL memory dump (containing hex values in the range 0-255).

```
$ ./palutils -g256 -x 320 -y 400 -u dmem.txt -e dmem.bmp
```

2. Colorization of a greyscale image (monochrome or 256-level) by a given input palette (random assignment of colors). The resulting image is exported as a BMP file as well.

```
$ ./palutils -colorize -x 320 -y 200 -i ${colormap}.gpl
-b ${img}.bmp -e \
" recolored_${img}_${cmappix}.bmp"
```

3. Change the palette of a 256-color bitmap file with a user-supplied (or automatically generated) palette. The image data of the bitmap file are altered accordingly.

```
$ ./palutils -<char><num> -x <x-dim> -y <y-dim> -b
${imgin}.bmp -e ${imgout}.bmp
```

where for example:

- `<char><num> = g256`
- `<x-dim> = 320`
- `<y-dim> = 200`

4. Change the palette of a 256-color bitmap file with a user-supplied (or automatically generated) palette. The image data of the bitmap file are altered accordingly. In addition, generate a VHDL file for palette (with the name: ${imgin}.bmp.vhd).

```
$ ./palutils -<char><num> -vhdl -b ${imgin}.bmp -e
${imgout}.bmp
```

4. bmp2any usage

The `bmp2any` utility extracts the image data from a bitmap (BMP) file and dumps them to the specified format. `bmp2any` can be invoked as follows:

```
$ ./bmp2any [options] -i <infile> -o <outfile>
```

with the following options:

-d Generate an image data dump for the BMP data.

-x Generate a hex dump for the BMP data.

-c Generate a C array for the BMP data (default).

-a Generate assembly directives for the .data segment.

Usually, `bmp2any` is used for generating a hex dump of the image data contents of a BMP file. This can be done with the following command:

```
$ ./bmp2any -x -i ${img}.bmp -o ${img}.data
```

5. memgen usage

The `memgen` utility converts a hex dump to an initialized block RAM. It can be used for initializing block RAMs with BMP image data. *memgen'* is inspired but does not share any code with the `ram_image` utility of the Plasma (`mlite`) processor soft-core distribution at the OpenCores website.

`memgen` can be invoked as follows:

```
$ ./memgen [options] <in_data.txt> <out.vhd>
```

with the following options:

-h Print this help.

-s <num> Aggregate size of memory storage.

-l <num> Program/data word size.

-pmem Generate memory for program storage.

-dmem Generate memory for data storage.

-p Generate VHDL package for monitoring RAM memory signals (one for each memory block).

-m Generate multiple entities in VHDL dump.

-u Generate a single block RAM.

-b Generate VHDL for byte-wide access memory.

-w Generate VHDL for word-wide access memory.

The following example generates a VHDL package containing the signal declarations and initializations for the data that are stored in the block RAM(s).

```
$ ./memgen -s 131072 -p -m -b ${img}.data ${img}_data.vhd
```

6. genpalette usage

The `genpalette` utility generates palettes from a predefined set of possible selections. The generated palettes are compatible with GIMP and mtPaint.

`genpalette` can be invoked as follows:

```
$ ./genpalette [options] -o <outfile.gpl>
```

with the following options:

-h Print this help and exit program.

-g256 Generate a 256-level greyscale palette.

-segm Generate a customized 256-level color palette for visualizing segmentations.

-bw Generate a black-and-white palette.

The following examples generates a 256-level greyscale palette.

```
$ ./genpalette -g256 -o greyscale.gpl
```

7. Advanced usage examples

- TODO, please refer to "8. Example scripts" for the time being.

8. Example scripts

The palutils distribution includes a number of bash scripts that make use of the utilities in the package. The usage of these scripts is explained below:

8.1. bmpembed.sh

Convert an 256-color (or 256-level greyscale) BMP image file to the corresponding VHDL source code for inclusion in the data memory module of the "kaviMIPS" processor.

Usage: `$ ./bmpembed.sh <input-base> <output-base> -[c8|c256|g256] <size>`

Example: `$ ./bmpembed.sh scomet2_8bpp_g scomet2 -c256 8192`

8.2. dump2bmp.sh

Convert an image dump data file (generated by a VHDL simulation) to a 256-color bitmap file.

Usage: `$ ./dump2bmp.sh <base-name> <x-size> <y-size>`

Example: `$ ./dump2bmp.sh dmem 64 64`

8.3. dumptrad2bmp.sh

Convert an image dump data file (generated by a VHDL simulation, using the traditional format of 8 or 27-color information) to a 256-color bitmap file.

Usage: `$ ./dump2bmp.sh <base-name> <x-size> <y-size>`

Example: `$ ./dump2bmp.sh dmem 64 64`

8.4. imgprocess.sh

Apply image processing procedures to BMP files. As of current, adding pseudocolors to greyscale BMP images is the only feature supported.

Usage: `$ ./imgprocess.sh`

8.5. make_char_rom.sh

A script that creates a font character VHDL ROM from a hex data file. The ROM data (text file) are expected in the following form:

```
<hex-byte>
...
<hex-byte>
```

A maximum of 2048 lines (for Spartan-3/3E block RAMs) can be assigned. Output of this process is a VHDL synchronous-read, synchronous-write ROM that can be implemented with a block RAM.

Usage: `$ ./make_char_rom.sh <font-name>`

9. Image formats in palutils

The supported image formats in palutils are:

- An image data dump format for describing 8-color (RGB-3) pixels. Each color is represented by a single character. The following table provides the color encodings and their character assignments:

R	G	B	Char	Description
0	0	0	N	Black (Noir)
0	0	1	B	Blue
0	1	0	G	Green
0	1	1	C	Cyan
1	0	0	R	Red
1	0	1	M	Magenta
1	1	0	Y	Yellow
1	1	1	W	White

- An image data dump format for describing 27-color (3-levels: low-, mid-, high per R,G,B) pixels. Each color is represented by a single character. The following table provides the color encodings and their character assignments:

R	**G**	**B**	**Char**	**Description**
00	00	00	N	Black (Noir)
00	00	01	n	Navy Blue
00	00	10	B	Blue
00	01	00	A	Avocado
00	01	01	s	Teal
00	01	10	a	Azure
00	10	00	G	Green
00	10	01	E	Spring Green
00	10	10	C	Cyan
01	00	00	b	Maroon (Brown)
01	00	01	P	Medium Purple
01	00	10	I	Indigo
01	01	00	K	Olive (Khaki)
01	01	01	g	Gray
01	01	10	i	Indigo Gray
01	10	00	L	Chartreuse
01	10	01	r	Light Green
01	10	10	c	Light Cyan
10	00	00	R	Red
10	00	01	H	Rose (Hot Pink)
10	00	10	M	Magenta
10	01	00	O	Orange
10	01	01	S	Skintone
10	01	10	m	Light Magenta
10	10	00	Y	Yellow
10	10	01	y	Light Yellow
10	10	10	W	White

- 8-bit paletted color bitmaps
- 8-bit greyscale bitmaps

10. Installation

There exists a quite portable Makefile (Makefile in `/src` directory). Running `make` should compile all files: palutils, bmp2any, memgen and genpltenc.

11. Prerequisities

- [mandatory for building] Standard UNIX-based tools
  - gcc (tested with gcc-3.3.3, 3.4.3 and 3.4.4+ on cygwin/x86)
  - make