Name

paletteconverter — Converts colour palettes (look-up tables) between different formats.

Synopsis

`paletteconverter [(--input | -i) <PATH>[;<PATH>]* [(--fromformat | -f) <FORMAT>][--recurse](--output | -o) <FILENAME>[(--toformat | -t) <FORMAT>][(--cell-width | -w) <INTEGER>][(--cell-height | -h) <INTEGER>][--columns <INTEGER>][--css.name <FORMAT_STRING>][--css.attribute <STRING>][--js.name <FORMAT_STRING>][--js.type (hex | triplet)][(--verbose | -v)][--print-files][--print-files.prefix <STRING>][--print-files.divider <STRING>][--print-summary][--print-set-options]]`

`paletteconverter [--help]`

`paletteconverter [--version]`

DESCRIPTION

PaletteConverter is a command line utility for converting palettes between different formats. It currently supports the following formats:

- GIMP palette (*.gpl)
- Adobe Photoshop palette (*.aco)
- Fractint palette (*.map)
- Microsoft palette (*.pal)
- A hexadecimal color list (*.hextxt)

Each of these formats can be used as input and the read palette can be written in one of these formats.

The following formats can be exported only, but not read:

- HTML (as a table) (*.html)
- CSS color list (*.css)
- JavaScript list (*.js)

Additionally, PaletteConverter can render palettes to .gif-files.

OPTIONS

Input Options:

--input *<PATH>[;<PATH>]**, --i *<PATH>[;<PATH>]**	Defines the files to read; paths are accepted as well.. *Mandatory, type:file- or pathname, default: none*
--fromformat *<FORMAT>*, --f *<FORMAT>*	Defines the input format. *Optional (if the input's extension is unambiguous), type:enum ("gpl", "aco", "map", "pal", "hextxt"), default: none*
--recurse	If given, PaletteConverter will also look for matching files in subfolders. *Optional, type:bool, default: false*

Output Options:

--output <*PATH*>[;<*PATH*>]*, --o <*PATH*>[;<*PATH*>]*	Defines the name of the output file. *Optional (see "USAGE"), type:filename, default: none*
--toformat <*FORMAT*>, -t <*FORMAT*>	Defines the output format. *Mandatory if multiple files are processed, otherwise determined from --output option, type:enum ("gpl", "aco", "map", "pal", "hextxt", "html", "css", "js", "gif"), default: none*

Image Output Options:

--cell-width <*INTEGER*>, --w <*INTEGER*>	Defines the width of a colour cell in pixels. *Optional, type:integer, default: 1*
--cell-height <*INTEGER*>, --h <*INTEGER*>	Defines the height of a colour cell in pixels. *Optional, type:integer, default: 1*
--columns <*INTEGER*>	Defines after how columns the exported image shall have. *Optional, type:integer, default: unset (all colors in one line)*

CSS Export Options:

--css.name <*FORMAT_STRING*>	A format string that defines how the names of the css-entries are generated. *Optional, type:string, default: ".pal_%n_%i"*
--css.attribute <*NAME*>	The name of the attribute to assign the color to. *Optional, type:string, default: "background-color"*

JavaScript (.js) Export Options:

--js.name <*FORMAT_STRING*>	A format string that defines how the name of the JavaScript-variable is generated. *Optional, type:string, default: "%n"*
--js.type <*TYPE*>	Defines how the entries shall be rendered, one of "hex" or "triplet". *Optional, type:enum ("hex", "triplet"), default: "hex"*

Reporting Options:

--verbose, -v	Enables verbose reporting. *Optional, type:bool, default: false*
--print-files	Prints the files that are processed. *Optional, type:bool, default: false*
--print-files.prefix <*STRING*>	Defines the prefix to be used. *Optional, type:string, default: "#"*
--print-files.divider <*STRING*>	IDefines the divider between files. *Optional, type:string, default: ";"*
--print-summary	Prints a summary after processing. *Optional, type:bool, default: false*
--print-set-options	Prints option values before processing. *Optional, type:bool, default: false*

Meta Options:

--version	Prints the application version and stops. *Optional, type:bool, default: false*
--help, -?	Prints the help screen and stops. *Optional, type:bool, default: false*

USAGE

The file to read is given to PaletteConverter using the command line option `--input <PATH>[,<PATH>]*` (or `-i <PATH>[,<PATH>]*` for short). More than one path can be given with ',' being the separator. *<PATH>* may be either the name of a palette file itself, including wildcards, or the path to a folder. In the second case, you should additionally set the option `--recurse` to walk down the folder tree.

The output path is defined using the option `--output <PATH>` (or `-o <PATH>` for short). If only one file is converted, the option should contain the name of the file to generate. You may omit to define the output format, if this file's extension is known to PaletteConverter.

If no output path is given, the name of the file(s) to generate is constructed by appending the output format's extension to the name of the converted input file without extension. Please note that the output format must be given in this case. If the output path is given and more than one file shall be converted, each output file name is constructed as:

OP := *<INPUT_PATH><OUTPUT_PATH><TO_FORMAT_EXTENSION>*

Where *OP is the* name of the generated file; *<INPUT_PATH>* is the path to the file to convert, without the file's name; *<OUTPUT_PATH>* is the value of the output option; *<TO_FORMAT_EXTENSION>* is the extension, as defined using `--toformat <FORMAT>.`

PaletteConverter recognizes the file format of input and output files by extension. It knows:

- GIMP palette (*.gpl)
- Adobe Photoshop palette (*.aco)
- Fractint palette (*.map)
- Microsoft palette (*.pal)
- A list of hexadecimal color entries, one by line (*.hextxt)

Additionally, the following formats are recognized as valid output formats:

- HTML (as a table) (*.html)
- CSS color list (*.css)
- JavaScript list (*.js)
- .gif image (*.gif)

In the case your files have a different or none extension, you have to let PaletteConverter know in what format the files to convert are stored. This is done using the option `--fromformat <FORMAT>` (or `-f <FORMAT>` for short). The same counts for output files. If no known extension is given for the output file, you have to specify it using the `--toformat <FORMAT>` (or `-t <FORMAT>` for short) option.

Some formats have additional parameters.

When writing image files, PaletteConverter writes all colours as single pixels into one line per default. The parameter `--columns <INTEGER>` defines after how many columns, a new line shall be started. The height of a colour cell can be changed using `--cell-height <INTEGER>` (`-h <INTEGER>` for short). Its width can be accordingly changed using `--cell-width <INTEGER>` (`-w <INTEGER>` for short). Both integers are interpreted as numbers of pixels. Please note that only palettes of no more than 256 entries can be used for image generation.

The rendering of .css-tables can be altered as well. The parameter `--css.attribute <NAME>` sets the name of the written attribute, e.g. "font-color". The default is "`background-color`". The parameter `--css.name`

<FORMAT_STRING> defines how the name of the entry is rendered. The parameter is a format string as described in detail below. The default is "`.pal_%n_%i`" - %n is replaced by the palette's name, &i is replaced by the entry's index.

When exporting to JavaScript, the name of the variable can be changed using `--js.name` *<FORMAT_STRING>*. The parameter `--js.type` allows to change how the entries are rendered. The default behaviour is to render them as hex-strings ("#000000" for black). Setting `--js.type triplet` will render them as RGB-triplets ("[0,0,0]" for black).

You may find the obligatory `--verbose` (or `-v`) option which shows some more output. You can force PaletteConverter to write the files it processes using `--print-files` and if you do, you can define the prefix to use using `--print-files.prefix` *<STRING>* and the divider to use using `--print-files.divider` *<STRING>*. You can also print the set options using `--print-set-options`.

Formatted String

At some places, a FORMAT_STRING is used. The string may contain placeholders for current values, e.g. the color table entry's index. The following placeholders are supported:

- `%f`: the filename without extension
- `%n`: the name of the palette if given in the file, otherwise the file name (%f)
- `%i`: the index of the palette entry

Examples

paletteconverter.exe -i myOrigPalette.aco -o myNewPalette.gpl

Reads a palette from the file `myOrigPalette.aco` assuming it is a Adobe Photoshop palette because of knowing the extension and stores it as a GIMP palette into `myNewPalette.gpl`.

paletteconverter.exe -i palette.map -o palette.gif

Reads a palette from the file `palette.map` assuming it is a Fractint palette and generates an image of the palette storing it into the file `palete.gif`.

CAVEATS

Nothing is known, yet.

DIAGNOSTICS

PaletteConverter returns 0 if the given options were valid and the file(s) could be processed.

If a needed option is missing, 1 is returned. If an option was given, but the value cannot be used (e.g. an unknown format was given), PaletteConverter will return a 2. On any other error, PaletteConverter returns 3 and the error is printed.

BUGS

Nothing is known, yet. Please report bugs to daniel@krajzewicz.de.

SEE ALSO

Internet pages at http://www.krajzewicz.de/paletteconverter

COPYRIGHT

http://www.krajzewicz.de/paletteconverter