Bitmap Writer

Generated by Doxygen 1.8.6

Thu Jul 27 2017 11:33:15

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 COLOR Namespace Reference

5.1.1 Detailed Description

Namespace containing different colour definitions:

- Black
- White
- Grey
- Lightgrey
- Darkgrey
- Blue
- Red
- Green
- Magenta
- Cyan
- Yellow

Chapter 6

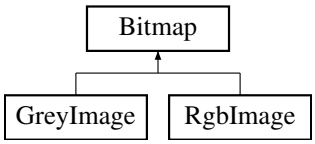
Class Documentation

6.1 Bitmap Class Reference

Abstract Bitmap class.

`#include <Bitmap.h>`

Inheritance diagram for Bitmap:



Classes

- struct BmpFileHeader_t
- struct DibHeader_t

Public Member Functions

- virtual ~Bitmap ()
- void load ()
- void load (const std::string &fname)
- void store () const
- void store (const std::string &fname) const

Protected Member Functions

- Bitmap ()
- Bitmap (int32_t width, int32_t height)
- Bitmap (const std::string &fname)
- void check (int32_t x, int32_t y) const

Protected Attributes

- BmpFileHeader_t m_header
- DibHeader_t m_dib

- std::string m_fname
- int32_t m_width
- int32_t m_height
- int32_t m_rowsize
- uint8_t ** m_data

6.1.1 Detailed Description

Abstract Bitmap class.

This class should not be used directly, as it has no sense of different colour depths. Image data is seen on byte level only.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Bitmap::~Bitmap ( ) [virtual]

Cleans up dynamically genereted bitmap array

6.1.2.2 Bitmap::Bitmap ( ) [protected]

6.1.2.3 Bitmap::Bitmap (int32_t *width*, int32_t *height*) [protected]

6.1.2.4 Bitmap::Bitmap (const std::string & *fname*) [protected]

6.1.3 Member Function Documentation

6.1.3.1 void Bitmap::check (int32_t *x*, int32_t *y*) const [protected]

6.1.3.2 void Bitmap::load ( )

Loads image data from file, if and only if the member field m_fname is set, which happens whens using the string constructor. If m_fname is empty, this function ends.

6.1.3.3 void Bitmap::load (const std::string & *fname*)

Loads image from file with given name. Throws invalid_argument when file could not be opened and runtime_error when file is incomplete. No other checks are done.

Parameters

fname	URL to file

6.1.3.4 void Bitmap::store ( ) const

Stores image data to file, if and only if member field m_fname is set, which happens when using the string constructor. if m_fname is empty, this function ends.

6.1.3.5 void Bitmap::store (const std::string & *fname*) const

Stores image as bmp to file denoted by fname. Existing files will be overwritten

Parameters

fname	URL to file

6.1.4 Member Data Documentation

6.1.4.1 uint8_t** Bitmap::m_data [protected]

6.1.4.2 DibHeader_t Bitmap::m_dib [protected]

6.1.4.3 std::string Bitmap::m_fname [protected]

6.1.4.4 BmpFileHeader_t Bitmap::m_header [protected]

6.1.4.5 int32_t Bitmap::m_height [protected]

6.1.4.6 int32_t Bitmap::m_rowsize [protected]

6.1.4.7 int32_t Bitmap::m_width [protected]

The documentation for this class was generated from the following files:

- Bitmap.h
- Bitmap.cpp

6.2 Bitmap::BmpFileHeader_t Struct Reference

`#include <Bitmap.h>`

Public Attributes

- uint8_t magic [2]
- uint32_t filesz
- uint16_t creator1
- uint16_t creator2
- uint32_t bmp_offset

6.2.1 Detailed Description

magic: Header identifier, should be "BM" for Windows NT Bitmaps

filesz: complete file size in bytes

creator1 and creator2: reserved

bmp_offset: Offset where pixel data array can be found

6.2.2 Member Data Documentation

6.2.2.1 uint32_t Bitmap::BmpFileHeader_t::bmp_offset

6.2.2.2 uint16_t Bitmap::BmpFileHeader_t::creator1

6.2.2.3 uint16_t Bitmap::BmpFileHeader_t::creator2

6.2.2.4 uint32_t Bitmap::BmpFileHeader_t::filesz

6.2.2.5 uint8_t Bitmap::BmpFileHeader_t::magic[2]

The documentation for this struct was generated from the following file:

- Bitmap.h

6.3 Bitmap::DibHeader_t Struct Reference

`#include <Bitmap.h>`

Public Attributes

- uint32_t header_sz
- int32_t width
- int32_t height
- uint16_t nplanes
- uint16_t bitspp
- uint32_t compress_type
- uint32_t bmp_bytesz
- int32_t hres
- int32_t vres
- uint32_t ncolors
- uint32_t nimpcolors

6.3.1 Detailed Description

header_sz: Size of DIB Header in bytes. Use 40bytes for standard Windows NT DIB Header.

width: the bitmap width in pixels (signed integer)

height:the bitmap height in pixels (signed integer)

nplanes: the number of color planes must be 1

bitspp: the number of bits per pixel, which is the color depth of the image. Typical values are 1, 4, 8, 16, 24 and 32.

compress_type: Compression method used, 0 for none

bmp_bytesz: Size of raw bitmap data, can be 0 if no compression is used

hres and vres: horizontal and vertical resolution in pixel per meter

ncolors: the number of colours in the colour palette

nimpcolor: the number of important colours, 0 for every colour is important

6.3.2 Member Data Documentation

6.3.2.1 uint16_t Bitmap::DibHeader_t::bitspp

6.3.2.2 uint32_t Bitmap::DibHeader_t::bmp_bytesz

6.3.2.3 uint32_t Bitmap::DibHeader_t::compress_type

6.3.2.4 uint32_t Bitmap::DibHeader_t::header_sz

6.3.2.5 int32_t Bitmap::DibHeader_t::height

6.3.2.6 int32_t Bitmap::DibHeader_t::hres

6.3.2.7 uint32_t Bitmap::DibHeader_t::ncolors

6.3.2.8 uint32_t Bitmap::DibHeader_t::nimpcolors

6.3.2.9 uint16_t Bitmap::DibHeader_t::nplanes

6.3.2.10 int32_t Bitmap::DibHeader_t::vres

6.3.2.11 int32_t Bitmap::DibHeader_t::width

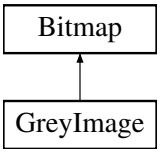
The documentation for this struct was generated from the following file:

- Bitmap.h

6.4 GreyImage Class Reference

`#include <Bitmap.h>`

Inheritance diagram for GreyImage:



Public Member Functions

- GreyImage (int32_t width, int32_t height)
- GreyImage (const std::string &fname)
- GreyPixel getPixel (int32_t x, int32_t y) const
- void setPixel (int32_t x, int32_t y, GreyPixel val)
- void fill (GreyPixel p)

Additional Inherited Members

6.4.1 Constructor & Destructor Documentation

6.4.1.1 GreyImage::GreyImage (int32_t *width*, int32_t *height*)

Creates an instance with given width and height in pixels. The proper amount of memory is allocated, but not filled, so image data is kind of random at this point. This throws an invalid_argument exception if width and/or height is negative. Note that the standard actually supports negative height, but this implementation does not.

Note that also this uses a LUT, there is currently no way to edit the LUT. The LUT is created when the image is stored as a .bmp file and contains 256 entries for the 256 Greyscale colours.

Parameters

width	Desired image width in pixels
height	Desired image height in pixels

6.4.1.2 GreyImage::GreyImage (const std::string & *fname*)

Creates an instance by loading from the file given as parameter. This uses the Bitmap class' method load() to load that file.

If the file contained a LUT, this LUT is not loaded to memory.

Parameters

fname	URL to file

6.4.2 Member Function Documentation

6.4.2.1 void GreyImage::fill (GreyPixel *p*)

This fills the whole image by calling setPixel for every valid (x,y) combination.

Parameters

p	GreyPixel element that should be written

6.4.2.2 GreyPixel GreyImage::getPixel (int32_t *x*, int32_t *y*) const

Gets a pixel from the Bitmap. Throws out_of_range if the desired pixel is not inside the actual Bitmap.

Parameters

x	X position, 0 is left
y	Y position, 0 is up

Returns

Copy of GreyPixel Element at (x,y)

6.4.2.3 void GreyImage::setPixel (int32_t *x*, int32_t *y*, GreyPixel *val*)

Sets the desired pixel to the given value. Throws out_of_range and does not change anything if the desired location is outside the actual image.

Parameters

x	X position, 0 is left
y	Y position, 0 is up
val	GreyPixel element that should be written

The documentation for this class was generated from the following files:

- Bitmap.h
- Bitmap.cpp

6.5 GreyPixel Class Reference

`#include <Bitmap.h>`

Public Member Functions

- GreyPixel (uint8_t value)

Public Attributes

- uint8_t value

6.5.1 Detailed Description

Class holding single byte value denoting grey values ranging from 0 to 255.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 GreyPixel::GreyPixel (uint8_t *value*) [inline]

Constructs GreyPixel and sets internal value to given value

Parameters

value	Grey value ranging from 0 to 255, 0 beeing black

6.5.3 Member Data Documentation

6.5.3.1 uint8_t GreyPixel::value

The documentation for this class was generated from the following file:

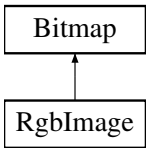
- Bitmap.h

6.6 RgbImage Class Reference

Class handling 24-bit RGB Images.

`#include <Bitmap.h>`

Inheritance diagram for RgbImage:



Public Member Functions

- RgbImage (int32_t width, int32_t height)
- RgbImage (const std::string &fname)

- RgbPixel getPixel (int32_t x, int32_t y) const
- void setPixel (int32_t x, int32_t y, RgbPixel val)
- void fill (RgbPixel p)

Additional Inherited Members

6.6.1 Detailed Description

Class handling 24-bit RGB Images.

This class handles a single Bitmap image with a colour depth of 24 bit, meaning three byte values for red, green and blue channel. An Alpha Channel is not supported.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 RgbImage::RgbImage (int32_t *width*, int32_t *height*)

Creates an instance with given width and height in pixels. The proper amount of memory is allocated, but not filled, so image data is kind of random at this point. This throws an invalid_argument exception if width and/or height is negative. Note that the standard actually supports negative height, but this implementation does not.

Parameters

width	Desired image width in pixels
height	Desired image height in pixels

6.6.2.2 RgbImage::RgbImage (const std::string & *fname*)

Creates an instance by loading from the file given as parameter. This uses the Bitmap class' method load() to load that file.

Parameters

fname	URL to file

6.6.3 Member Function Documentation

6.6.3.1 void RgbImage::fill (RgbPixel *p*)

This fills the whole image by calling setPixel for every valid (x,y) combination.

Parameters

p	RgbPixel element that should be written

6.6.3.2 RgbPixel RgbImage::getPixel (int32_t *x*, int32_t *y*) const

Gets a pixel from the Bitmap. Throws out_of_range if the desired pixel is not inside the actual Bitmap.

Parameters

x	X position, 0 is left

y	Y position, 0 is up

Returns

Copy of RgbPixel Element at (x,y)

6.6.3.3 void RgbImage::setPixel (int32_t *x*, int32_t *y*, RgbPixel *val*)

Sets the desired pixel to the given value. Throws out_of_range and does not change anything if the desired location is outside the actual image.

Parameters

x	X position, 0 is left
y	Y position, 0 is up
val	RgbPixel element that should be written

The documentation for this class was generated from the following files:

- Bitmap.h
- Bitmap.cpp

6.7 RgbPixel Class Reference

`#include <Bitmap.h>`

Public Member Functions

- RgbPixel (const GreyPixel &p)
- RgbPixel (uint8_t blue, uint8_t green, uint8_t red)

Public Attributes

- uint8_t red
- uint8_t green
- uint8_t blue

6.7.1 Detailed Description

Class holding three byte values denoting red, green and blue values for creating a RGB-Value.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 RgbPixel::RgbPixel (const GreyPixel & *p*) [inline]

Conversion from GreyPixel. Sets alls red, green and blue values to grey value from given GreyPixel

Parameters

p	GreyPixel to be copied

6.7.2.2 RgbPixel::RgbPixel (uint8_t *blue*, uint8_t *green*, uint8_t *red*) [inline]

Instantiates a RgbPixel with given red, green and blue values

Parameters

blue	Value of blue chanel from 0 to 255, 0 meaning black
green	Value of green chanel from 0 to 255, 0 meaning black
red	Value of green chanel from 0 to 255, 0 meaning black

6.7.3 Member Data Documentation

6.7.3.1 uint8_t RgbPixel::blue

6.7.3.2 uint8_t RgbPixel::green

6.7.3.3 uint8_t RgbPixel::red

The documentation for this class was generated from the following file:

- Bitmap.h

Chapter 7

File Documentation

7.1 .dep.inc File Reference

7.2 Bitmap.cpp File Reference

```
#include "Bitmap.h"
#include <fstream>
#include <stdexcept>
```

7.3 Bitmap.h File Reference

```
#include <stdint.h>
#include <iostream>
#include <string>
```

Classes

- class Bitmap

  Abstract Bitmap class.
- struct Bitmap::BmpFileHeader_t
- struct Bitmap::DibHeader_t
- class GreyPixel
- class RgbPixel
- class RgbImage

  Class handling 24-bit RGB Images.
- class GreyImage

Namespaces

- COLOR

7.4 main.cpp File Reference

```
#include <cstdlib>
#include <iostream>
#include "Bitmap.h"
#include <fstream>
```

Functions

- int main (int argc, char **argv)

7.4.1 Function Documentation

7.4.1.1 int main (int *argc,* char ** *argv*)

Index