LMTK

Generated by Doxygen 1.9.2

1 LMTK

1.1 Light Multimedia ToolKit

Multimedia processing tool for images, audio and video. Currently in early development for images. Written in C++. A student/hobby project.

Made by marinarasub 2021 - 2022

Currently, this file contains developement progress and features. Keep in mind that this project is under development and there may be uncompleted, buggy or unusable features. I will probably move a lot of this to seperate files/ locations once it gets long.

Note: every feature is WIP, even if not stated there probably is not full support (i.e. metadata, bit depths etc.)

1.2 Latest Release

Coming soon. Currently developing for Windows 32 and 64 bit.

1.3 Features

Here are some of the main features . Demos are available see DEMO.md

1.3.1 Images

See images/ for more. Images used are rolyalty free. Contact me for copyright or takedown, or source (author, url etc.) request. Mostly taken from https://unsplash.com/

1.3.1.1 Supported Formats

- PNG [read/write]
- JPEG [read/write]

1.3.1.2 Tools

- Image convolution
- Resizing TODO bicubic and bilinear
- Mirror TODO angle
- Gradient TODO more options lol
- Alpha blending
- TODO conversion between formats

1.3.1.3 Effects

- Blurs & Sharpen
  - Box/fast blur
  - Gaussian blur
  - Bokeh/lens blur
- Adjustment/Correction
  - HSL (Hue/Sat/Lum) Adjust
- Other
  - Grayscale
  - Chroma key

1.3.2 Audio / DSP

1.3.2.1 Supported Formats

- FLAC [read only]
- NEXT PLANNED: MP3 (MPEG Audio Layer III)

1.3.2.2 Tools/fx

- Audio player to output device
  - Live gain
- Track Blending (just add them like A + B!)
- Wave Generators:
  - Sine
  - Saw
  - Square
- TODO low pass, high pass, band pass
- TODO peak detection

1.3.3 Video

1.3.3.1 Supported Formats

- none lol

1.3.4 Other

- DFT + FFT [WIP] ∗might choose to use fftw or other lib instead
  - Power/frequency spectrum
- Some math stuff

1.4 Compiling / Building

Currently, the project is not generalized for build. I will make it easier to build in the future, perhaps with CMake or something.

I use Visual C++ (Visual Studio).

1.5 Performance

I'll update this once the project is more mature.

1.6 License

MIT license. see LICENSE.

I have not published any third party source code or binaries so far as of January 2022, but info on licenses and links are below.

1.7 Documentation / API

Right now comments are formatted for `Doxygen` mostly using Javadoc styles. Subject to (very much) change. See docs/.

Will update in the future.

1.8 Development

Although it is not intended for community contribution right now, anyone is welcome to contribute, although you should probably wait until I clean up the code and ensure platform & compiler compatibility.

1.8.1 Required Libraries

The following libraries are currently used in the project, subject to change.

I try to use the least libraries with less restrictive open source licenses.

1.8.1.1 Images

- `libpng` - PNG Reference Library License
- zlib (use link above) - zlib license
- `libjpeg-turbo` - Multiple licenses
  - OR `libjpeg` - IJG license

1.8.1.2 Audio

- PortAudio - PortAudio license
- FLAC (libFLAC++) - BSD

1.8.1.3 Other

- Eigen - MPL 2.0
  - will include license if binaries/source code distributed

1.8.2 Testing

I am using Google Test to unit test.

Utility funcitons are tested "normally" using assertions and such.

For media, testing only covers basic things so far or does not perform rigorous checks. Often, minor tweaks are common and it is easier just to look/listen to the result instead of writing tests that break every minor change.

As project features mature, expect more test coverage.

1.8.3 Future / Roadmap / Random TODOs

- Text/font support
- Progress callbacks for image stuff (for loading bar etc.)
- Multithreaded processing
- I plan to make a command line interface and GUI, maybe using WxWidgets or something for editing images, audio etc.
- More formats support.
- Speed optimization
- Probably use ffmpeg in the future.
- Video editor, maybe.
- Writing this file takes a while so I will continue to update as I think of stuff.
- Linux, Mac OS
- Mobile???# Contains all source code

2 Contributing Guidelines

2.1 Issues

TODO

2.2 Commits & Pull Requests

TODO

2.3 Code Style

The code style attempts to be consistent and readable, and is subject to change, but here are the styling I use so far:

For the most part, I follow a similar format to Google C++ Style Guide

2.3.1 General

- blank lines between *everything* for readability
  - ok seriously, use blank lines when better readability
- two newlines at after include, header guards, macros before content
  - header guards follow google C++ conventions
  - include c++ library, then external library headers before user defined

```
#ifndef _SILLYPROJECT_HEADER_H_
#define _SILLYPROJECT_HEADER_H_
#include <stdlib.h>
#include <externallib.h>
#include userfile.h"
class Content
{
    ...
}
#endif // _SILLYPROJECT_HEADER_H_
```

2.3.2 Comments

- Follow Doxygen Javadoc styling for formal specifications

```
/**
* Brief description.
*
# @tags like this
*/
code...
```

- Use // for temporary comments or code likely to change

```
// TODO fix this later (which means never lol)
messy code...
```

- TODO will add banners for each header file once more mature

```
/*****************************************************************************//**
 *
 * THIS THING AT THE TOP
 *
 * SOME IMPORTANT INFO...
 *
 *****************************************************************************/
```

2.3.3 Naming

- UPPERCASE for constants, macros etc.
```
#define MACRO
const char* CONSTANT
```
- PascalCase for classes
```
class HelloWorld
```
- camelCase for functions
```
void myFunc()
```
- snake_case for fields/members and structs (more hidden/low level stuff)
```
double me_is_snake_case
struct my_struct
```
- snake_case for namespaces
```
namespace not_std
```

2.3.4 Brackets/Code Blocks

- curly on next line for functions, classes and branch logic
```
class Class
{
    void func()
    {
        ...
    }
}
"""
- curly on same line for namespaces and try/catch blocks
```
namespace hello {

try {

} catch (...) {

} }
```
- branch logic (not as strict) followed by space for condition and curly next line
```
if (condition) { ... } else { ... }
```
- function calls are no space (is there any other sane way?)
```
hello.function() ```
```

# 3 Namespace Index

## 3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:
```

4 Hierarchical Index

4.1 Class Hierarchy

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

5 Class Index

5.1 Class List

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

6 File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

7 Namespace Documentation

7.1 color Namespace Reference

Contains color representations, functions and conversion between them.

Classes

- struct hsl_color
- struct rgb_color

Functions

- hsl_color **rgb_to_hsl** (double r, double g, double b)

  Translated from `https://www.rapidtables.com/convert/color/hsl-to-rgb.html`.
- hsl_color **rgb_to_hsl** (const rgb_color &clr)
- rgb_color **hsl_to_rgb** (double h, double s, double l)
- rgb_color **hsl_to_rgb** (const hsl_color &clr)

7.1.1 Detailed Description

Contains color representations, functions and conversion between them.

7.2 image::hslcolor Namespace Reference

Contains common colors as constants for convinience.

7.2.1 Detailed Description

Contains common colors as constants for convinience.

See also

HSLAPixel

7.3 image::rgbcolor Namespace Reference

Contains common colors as constants for convinience.

7.3.1 Detailed Description

Contains common colors as constants for convinience.

See also

RGBAPixel

7.4 utils::math Namespace Reference

contains math utility functions.

Typedefs

- typedef std::complex< double > **complex**
  Represents a complex number using doubles.
- typedef std::complex< float > **complexf**
  Represents a complex number using floats.

Functions

- complex real_to_complex (double val)

  Convert a number to a complex type.
- complexf real_to_complexf (float val)

  Convert a number to a complex type, using float.
- double sqr (double val)

  Returns the square of a number.
- float sqrf (float val)

  Returns the square of a number, for float.
- complex sqr (complex val)

  Squares a complex number.
- complexf sqrf (complexf val)

  Squares a complex number, for float.
- double nsum (double arr[ ], size_t n)

  Sums the number in an array of size n.
- float nsumf (float arr[ ], size_t n)

  Sums the number in an array of size n, for float32.
- double dist (double x1, double y1, double x2, double y2)

  Return 2D distance between two points.
- double gaussian (double mean, double sigma, double x)

  Compute the gaussian value of a variable x with mean and standard deviation.
- double gaussian2d (double mean_x, double mean_y, double sigma_x, double sigma_y, double x, double y)

  Compute the 2D gaussian value of a variable x with means and standard deviations.

Variables

- constexpr complex **IMAGINARY_UNIT** = complex(0, 1)

  The imaginary unit, also known as i, sqrt(-1).
- constexpr complexf **IMAGINARY_UNIT_F** = complexf(0, 1)

  The imaginary unit, also known as i, sqrt(-1) using float32.
- const double **EULER** = std::exp(1)

  Euler's number.
- constexpr double **PI** = 3.14159265358979238463

  Pi to a decent precision.

7.4.1 Detailed Description

contains math utility functions.

In most cases, the functions use cartesian coordinates and conventions in euclidian space x, y, z are used to represent major axes.

7.4.2 Function Documentation

7.4.2.1 dist()

```
double utils::math::dist (
        double x1,
        double y1,
        double x2,
        double y2 )
```

Return 2D distance between two points.

Uses euclidian distance

Parameters

x1	X-position of first coordinate
y1	Y-position of first coordinate
x2	X-position of second coordinate
y2	Y-position of second coordinate

Returns

Euclidian distance between (x1, x2) and (y2, y1)

7.4.2.2 gaussian()

```
double utils::math::gaussian (
        double mean,
        double sigma,
        double x )
```

Compute the gaussian value of a variable x with mean and standard deviation.

Gaussian is normalized so integral is 1.

Parameters

mean	Mean value of gaussian
sigma	Standard deviation of gaussian
x	Value of apply gaussian to

Returns

Gaussian of x using N(mean, sigma)

7.4.2.3 gaussian2d()

```
double utils::math::gaussian2d (
        double mean_x,
        double mean_y,
        double sigma_x,
        double sigma_y,
        double x,
        double y )
```

Compute the 2D gaussian value of a variable x with means and standard deviations.

Gaussian is normalized so integral is 1.

Parameters

mean_x	Mean x value of gaussian

Parameters

mean_y	Mean y of gaussian
sigma_x	Standard deviation in x of gaussian
sigma_y	Standard deviation in y of gaussian
x	X location of number
y	Y location of number

Returns

Gaussian at (x, y)

7.4.2.4 nsum() `double utils::math::nsum ( double arr[], size_t n )`

Sums the number in an array of size n.

Parameters

arr	Array of numbers
n	Size of array

Returns

Sum of all elements up to index n-1

7.4.2.5 nsumf() `float utils::math::nsumf ( float arr[], size_t n )`

Sums the number in an array of size n, for float32.

Parameters

arr	Array of numbers
n	Size of array

Returns

Sum of all elements up to index n-1

7.4.2.6 real_to_complex() complex utils::math::real_to_complex (double *val*)

Convert a number to a complex type.

Parameters

val	Number to convert

Returns

Converted complex number

7.4.2.7 real_to_complexf() complexf utils::math::real_to_complexf (float *val*)

Convert a number to a complex type, using float.

Parameters

val	Number to convert

Returns

Converted complex number

7.4.2.8 sqr() [1/2] complex utils::math::sqr (complex *val*)

Squares a complex number.

Parameters

val	Complex number to square

Returns

Squared complex number

7.4.2.9 sqr() [2/2] double utils::math::sqr (double *val*)

Returns the square of a number.

Parameters

val	Number to square

Returns

Square of number, n ∗ n

7.4.2.10 sqrf() [1/2] complexf utils::math::sqrf (complexf *val*)

Squares a complex number, for float.

Parameters

val	Complex number to square

Returns

Squared complex number

7.4.2.11 sqrf() [2/2] float utils::math::sqrf (float *val*)

Returns the square of a number, for float.

Parameters

val	Number to square

Returns

Square of number, n ∗ n as float

8 Class Documentation

8.1 audio::audio_sample Struct Reference

a 2 channel (stereo) audio sample of values [-1, 1]

`#include "audiotrack.h"`

Public Member Functions

- audio_sample operator+ (const audio_sample &other) const

  operator+: Adds two audio samples together.
- void **clamp** ()

  Clamp each channel to [-1, 1].

Public Attributes

- float **left**

  Left channel sample.
- float **right**

  Right channel sample.

8.1.1 Detailed Description

a 2 channel (stereo) audio sample of values [-1, 1]

8.1.2 Member Function Documentation

8.1.2.1 operator+() `audio_sample audio::audio_sample::operator+ ( const audio_sample & other ) const [inline]`

operator+: Adds two audio samples together.

Both channels are added linearly

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

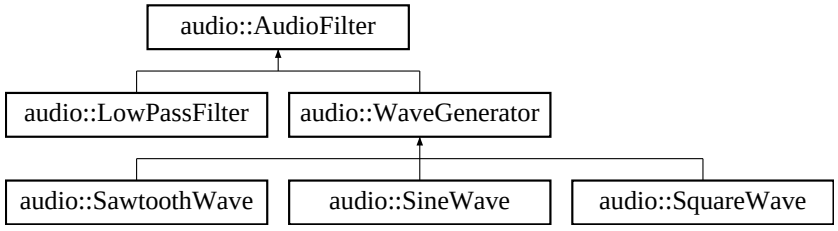
- src/libLMTKaudio/audiotrack.h
- src/libLMTKaudio/audiotrack.cpp

8.2 audio::AudioFilter Class Reference

Abstract audio filter class.

`#include "audiofilter.h"`

Inheritance diagram for audio::AudioFilter:



Public Member Functions

- virtual ~**AudioFilter** ()

  Virtual destructor.
- virtual void operator() (AudioTrack &track)=0

  Abstract operator to an audio track.

8.2.1 Detailed Description

Abstract audio filter class.

8.2.2 Member Function Documentation

8.2.2.1 operator()() `virtual void audio::AudioFilter::operator() ( AudioTrack & track ) [pure virtual]`

Abstract operator to an audio track.

Parameters

track	Audio track to filter

Implemented in audio::LowPassFilter, audio::SawtoothWave, audio::SineWave, audio::SquareWave, and audio::WaveGenerator.

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

- src/libLMTKaudio/audiofilter.h

8.3 audio::AudioPlayer Class Reference

An audio player for audio tracks to device output Uses portaudio for audio device IO.

`#include "audioplayer.h"`

Public Member Functions

- AudioPlayer ()

  Create an audio player.
- ~**AudioPlayer** ()

  Closes audio stream and terminates audio player.
- virtual void operator() (AudioTrack &track)

  Audio track operator: play the selected track.
- void play (AudioTrack &track)

Play the selected track to default audio device.

- void stop ()

  Stops playing the audio track.

- void **close** ()

  Close the audio stream, if open.

- AudioTrack * track ()
- void setGain (float db)

  Sets the gain of the output sound.

8.3.1 Detailed Description

An audio player for audio tracks to device output Uses portaudio for audio device IO.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 AudioPlayer() `audio::AudioPlayer::AudioPlayer ( )`

Create an audio player.

Opens an audio stream for playing

8.3.3 Member Function Documentation

8.3.3.1 operator()() `void audio::AudioPlayer::operator() ( AudioTrack & track ) [virtual]`

Audio track operator: play the selected track.

See also

play()

Parameters

track	Audio track to play

8.3.3.2 play() `void audio::AudioPlayer::play ( AudioTrack & track )`

Play the selected track to default audio device.

Parameters

track	Audio track to play

8.3.3.3 setGain() `void audio::AudioPlayer::setGain ( float db )`

Sets the gain of the output sound.

Parameters

db	Gain in decibels

8.3.3.4 stop() `void audio::AudioPlayer::stop ( )`

Stops playing the audio track.

AbortStream() for immediate stop (forceq)

8.3.3.5 track() `AudioTrack * audio::AudioPlayer::track ( )`

Returns

The audio track associated with the audio player

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

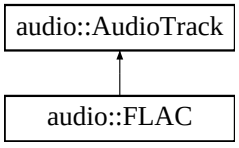
- src/libLMTKaudio/audioplayer.h
- src/libLMTKaudio/audioplayer.cpp

8.4 audio::AudioTrack Class Reference

Class for an AudioTrack containing audio samples which can be played, edited or written to a writeable format.

`#include "audiotrack.h"`

Inheritance diagram for audio::AudioTrack:



Public Member Functions

- **AudioTrack** ()

  Create an empty audio track of length 0.
- AudioTrack (unsigned long length)

  Create and audio track with n samples at default sample rate.
- AudioTrack (unsigned long length, unsigned long sample_rate)

  Create and audio track with n samples and sample rate.
- AudioTrack (const AudioTrack &other)

  Copy constructor.
- ~**AudioTrack** ()

  Delete all sample data.
- AudioTrack operator= (const AudioTrack &other)

  Assignment operator: sets this track equal to given one.
- AudioTrack operator+ (const AudioTrack &other) const

  operator+: Adds two audio tracks.
- unsigned int sampleRate () const
- unsigned long sampleLength () const
- byte channels () const
- void **resetPosition** ()

  Resets the cursor to START_POS.
- unsigned long currentPos () const
- bool ended () const
- audio_sample nextSample ()

  Using the track cursor, get the next sample and move the cursor by 1.
- audio_sample getSample (unsigned long idx) const

  Get the sample at given position in track.
- void setSample (unsigned long idx, audio_sample data)

  Set the sample at given index with an audio sample.
- void resample (unsigned long new_sample_rate)

  Resample the audio data to a new sample rate with minimal alteration to sound.
- void merge (AudioTrack &track)

  Adds given track to ∗this one.

Protected Member Functions

- void copy (const AudioTrack &other)

  Copies an audio track to ∗this, including audio data and track info.
- void **clear** ()

  Clear audio sample data.

Protected Attributes

- audio_sample ∗ **audio**

  Array of audio samples of the track.

8.4.1 Detailed Description

Class for an AudioTrack containing audio samples which can be played, edited or written to a writeable format.

See also

AudioPlayer

8.4.2 Constructor & Destructor Documentation

8.4.2.1 AudioTrack() [1/3] audio::AudioTrack::AudioTrack (unsigned long *length*)

Create and audio track with n samples at default sample rate.

Parameters

length	Length in samples

8.4.2.2 AudioTrack() [2/3] audio::AudioTrack::AudioTrack (unsigned long *length*, unsigned long *sample_rate*)

Create and audio track with n samples and sample rate.

Parameters

length	Length in samples
sample_rate	Sample rate in hz

8.4.2.3 AudioTrack() [3/3] audio::AudioTrack::AudioTrack (const AudioTrack & *other*)

Copy constructor.

Clear current data and copy all sample and track data to ∗this

Parameters

other	Track to copy

8.4.3 Member Function Documentation

8.4.3.1 channels() `byte audio::AudioTrack::channels ( ) const`

Returns

The number of audio channels

8.4.3.2 copy() `void audio::AudioTrack::copy ( const AudioTrack & other ) [protected]`

Copies an audio track to ∗this, including audio data and track info.

Parameters

other	Track to copy

8.4.3.3 currentPos() `unsigned long audio::AudioTrack::currentPos ( ) const`

Returns

Current cursor position in the track

8.4.3.4 ended() `bool audio::AudioTrack::ended ( ) const`

Returns

If the cursor has reached the end of the track

8.4.3.5 getSample() `audio_sample audio::AudioTrack::getSample ( unsigned long idx ) const`

Get the sample at given position in track.

Parameters

idx	Index of sample

8.4.3.6 merge() void audio::AudioTrack::merge (AudioTrack & *track*)

Adds given track to ∗this one.

See also

operator+

Parameters

track	Track to merge with

8.4.3.7 nextSample() audio_sample audio::AudioTrack::nextSample ( )

Using the track cursor, get the next sample and move the cursor by 1.

Returns

The next audio sample at the cursor

8.4.3.8 operator+() AudioTrack audio::AudioTrack::operator+ (const AudioTrack & *other*) const

operator+: Adds two audio tracks.

Adds by adding each sample data together. Also known as track mixing.

Parameters

other	Track to add

Returns

Result of mixing two tracks

8.4.3.9 operator=() AudioTrack audio::AudioTrack::operator= (const AudioTrack & *other*)

Assignment operator: sets this track equal to given one.

Parameters

other	Track to copy

Returns

Copied audio track

8.4.3.10 resample() `void audio::AudioTrack::resample ( unsigned long new_sample_rate )`

Resample the audio data to a new sample rate with minimal alteration to sound.

Parameters

new_sample_rate	New sample rate

8.4.3.11 sampleLength() `unsigned long audio::AudioTrack::sampleLength ( ) const`

Returns

The length of track in samples

8.4.3.12 sampleRate() `unsigned int audio::AudioTrack::sampleRate ( ) const`

Returns

The sample rate

8.4.3.13 setSample() `void audio::AudioTrack::setSample ( unsigned long idx, audio_sample data )`

Set the sample at given index with an audio sample.

Parameters

idx	Index of sample to set
data	Audio sample to set to

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

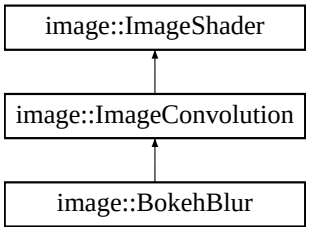
- src/libLMTKaudio/audiotrack.h
- src/libLMTKaudio/audiotrack.cpp

8.5 image::BokehBlur Class Reference

Class for a bokeh/lens blur using a disk/circular convolution.

`#include "bokehblur.h"`

Inheritance diagram for image::BokehBlur:



Public Member Functions

- BokehBlur (float radius)

  Constructs a bokeh blur functor with blur radius.
- BokehBlur (float radiusX, float radiusY)

  Constructs a bokeh blur functor with horizontal and vertial blur radius.

Private Member Functions

- virtual void computeKernel () override

  Computes a circular kernel using radii x, y.

Additional Inherited Members

8.5.1 Detailed Description

Class for a bokeh/lens blur using a disk/circular convolution.

See also

ImageConvolution

8.5.2 Constructor & Destructor Documentation

8.5.2.1 BokehBlur() [1/2] `image::BokehBlur::BokehBlur ( float radius )`

Constructs a bokeh blur functor with blur radius.

Parameters

radius	Radius of blur

8.5.2.2 BokehBlur() [2/2] image::BokehBlur::BokehBlur (float *radiusX*, float *radiusY*)

Constructs a bokeh blur functor with horizontal and vertial blur radius.

Parameters

radiusX	Horizontal radius of blur
radiusY	Vertical radius of blur

8.5.3 Member Function Documentation

8.5.3.1 computeKernel() void image::BokehBlur::computeKernel ( ) [override], [private], [virtual]

Computes a circular kernel using radii x, y.

The edges are antialiased.

See also

ImageConvolution::computeKernel()

Implements image::ImageConvolution.

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

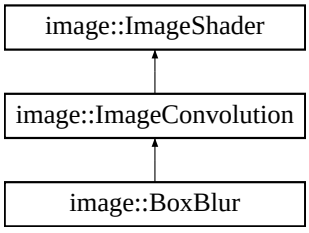
- src/libLMTKimage/bokehblur.h
- src/libLMTKimage/bokehblur.cpp

8.6 image::BoxBlur Class Reference

Class for box blur.

`#include "boxblur.h"`

Inheritance diagram for image::BoxBlur:



Public Member Functions

- **BoxBlur** ()

  Creates a box blur with a default 3x3 convolution.
- BoxBlur (size_t radius)

  Creates a box blur given radius.
- BoxBlur (size_t radiusX, size_t radiusY)

  Creates a box blur given horizontal and vertial radius.

Private Member Functions

- virtual void computeKernel () override

  Sets the kernel to all 1, normalized.

Additional Inherited Members

8.6.1 Detailed Description

Class for box blur.

A box blur convolution is all 1's, smoothing the image with equally weighted surrounding pixels. Repeated iterations / multiple passes approximates the Gaussian Blur.

See also

GaussianBlur

8.6.2 Constructor & Destructor Documentation

8.6.2.1 BoxBlur() [1/2] image::BoxBlur::BoxBlur (size_t *radius*)

Creates a box blur given radius.

Parameters

radius	The matrix radius

Note

"radius" loosely refers to how many pixels the matrix extends from the center horizontally and vertically

See also

ImageConvolution()

8.6.2.2 BoxBlur() [2/2] image::BoxBlur::BoxBlur (

```
size_t radiusX,
size_t radiusY )
```

Creates a box blur given horizontal and vertial radius.

Parameters

radiusX	The horizontal matrix radius
radiusY	The vertical matrix radius

Note

"radius" loosely refers to how many pixels the matrix extends from the center horizontally and vertically

See also

ImageConvolution()

8.6.3 Member Function Documentation

8.6.3.1 computeKernel() void image::BoxBlur::computeKernel ( ) [override], [private], [virtual]

Sets the kernel to all 1, normalized.

Implements image::ImageConvolution.

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

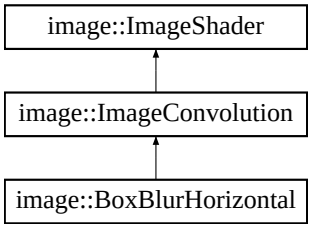
- src/libLMTKimage/boxblur.h
- src/libLMTKimage/boxblur.cpp

8.7 image::BoxBlurHorizontal Class Reference

Class for box blur's seperable horizontal component.

`#include "boxblur.h"`

Inheritance diagram for image::BoxBlurHorizontal:



Public Member Functions

- **BoxBlurHorizontal** ()

  Creates a box blur with default 3x1 convolution.
- BoxBlurHorizontal (size_t radius)

  Creates a box blur with a mx1 convolution.

Private Member Functions

- virtual void computeKernel () override

  Sets the kernel to all 1, normalized.

Additional Inherited Members

8.7.1 Detailed Description

Class for box blur's seperable horizontal component.

The horizontal convolution of box blur, which is a seperable kernel and may be done with a horizontal pass followed by a vertical pass (or vice versa), reducing time complexity from O(n $*$ m) to O(n + m)

See also

BoxBlur

BoxBlurVertical

8.7.2 Constructor & Destructor Documentation

8.7.2.1 BoxBlurHorizontal() `image::BoxBlurHorizontal::BoxBlurHorizontal ( size_t radius )`

Creates a box blur with a mx1 convolution.

Parameters

radius	Horizontal radius

8.7.3 Member Function Documentation

8.7.3.1 computeKernel() `void image::BoxBlurHorizontal::computeKernel ( ) [override], [private], [virtual]`

Sets the kernel to all 1, normalized.

Implements image::ImageConvolution.

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

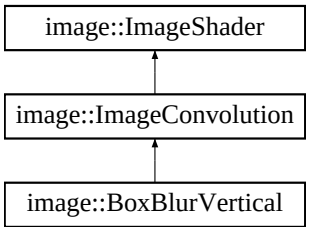
- src/libLMTKimage/boxblur.h
- src/libLMTKimage/boxblur.cpp

8.8 image::BoxBlurVertical Class Reference

Class for box blur's seperable vertical component.

`#include "boxblur.h"`

Inheritance diagram for image::BoxBlurVertical:



Public Member Functions

- **BoxBlurVertical** ()
  Creates a box blur with default 1x3 convolution.
- BoxBlurVertical (size_t radius)
  Creates a box blur with a 1xn convolution.

Private Member Functions

- virtual void computeKernel () override
  Sets the kernel to all 1, normalized.

Additional Inherited Members

8.8.1 Detailed Description

Class for box blur's seperable vertical component.

The vertical convolution of box blur, which is a seperable kernel and may be done with a horizontal pass followed by a vertical pass (or vice versa), reducing time complexity from O(n $*$ m) to O(n + m)

See also

BoxBlur

BoxBlurHorizontal

8.8.2 Constructor & Destructor Documentation

8.8.2.1 BoxBlurVertical() `image::BoxBlurVertical::BoxBlurVertical ( size_t radius )`

Creates a box blur with a 1xn convolution.

Parameters

radius	Vertical radius

8.8.3 Member Function Documentation

8.8.3.1 computeKernel() `void image::BoxBlurVertical::computeKernel ( ) [override], [private], [virtual]`

Sets the kernel to all 1, normalized.

Implements image::ImageConvolution.

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

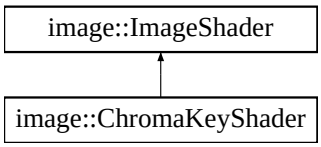
- src/libLMTKimage/boxblur.h
- src/libLMTKimage/boxblur.cpp

8.9 image::ChromaKeyShader Class Reference

Class for a simple chroma keyer (aka green screen).

`#include "chromakeyer.h"`

Inheritance diagram for image::ChromaKeyShader:



Public Member Functions

- ChromaKeyShader (RGBAPixel target)
  Creates a chroma keyer functor with default tolerance.
- ChromaKeyShader (RGBAPixel target, float tol)
  Creates a chroma keyer functor with tolerance.
- virtual void operator() (Image &img) override
  Removes the target color from the image.
- virtual RGBAPixel operator() (const Image &img, size_t x, size_t y)
  Returns a transparent pixel if the original pixel is within tolerance of target.

Static Public Attributes

- static constexpr float **DEFAULT_TOLERANCE** = 0.1

  The default color distance threshold.

Additional Inherited Members

8.9.1 Detailed Description

Class for a simple chroma keyer (aka green screen).

For every pixel in a given image, the operator sets the alpha to 0 if the color distance is within a defined threshold of the target color (default 0, which means the colors must be equal)

8.9.2 Constructor & Destructor Documentation

8.9.2.1 ChromaKeyShader() [1/2]

```
image::ChromaKeyShader::ChromaKeyShader (
        RGBAPixel target )
```

Creates a chroma keyer functor with default tolerance.

Sets the color threshold to default

Parameters

target	Target color to be removed.

8.9.2.2 ChromaKeyShader() [2/2]

```
image::ChromaKeyShader::ChromaKeyShader (
        RGBAPixel target,
        float tol )
```

Creates a chroma keyer functor with tolerance.

Parameters

target	Target color to be removed.
tol	Minimum threshold distance from the target color to be removed.

8.9.3 Member Function Documentation

8.9.3.1 operator()() [1/2] RGBAPixel image::ChromaKeyShader::operator() (

```
const Image & img,
size_t x,
size_t y )  [virtual]
```

Returns a transparent pixel if the original pixel is within tolerance of target.

Does not change the R, G, B channels in case alpha channel is ignored

Parameters

img	Image to operate chroma keyer on
x	X location of target pixel
y	Y location of target pixel

Returns

Transparent if within threshold, else the original pixel

Implements image::ImageShader.

8.9.3.2 operator()() [2/2] void image::ChromaKeyShader::operator() (

```
Image & img )  [override], [virtual]
```

Removes the target color from the image.

For each pixel in the image, removes the pixel if it is within threshold of the target color. It removes a color by setting its alpha value to 0. Otherwise, the pixel is left alone.

Parameters

img	Image to operate chroma keyer on

Reimplemented from image::ImageShader.

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

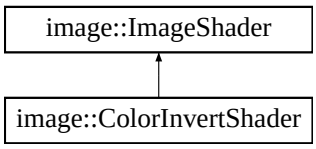
- src/libLMTKimage/chromakeyer.h
- src/libLMTKimage/chromakeyer.cpp

8.10 image::ColorInvertShader Class Reference

Class for a color inversion.

```
#include "colorinvert.h"
```

Inheritance diagram for image::ColorInvertShader:



Public Member Functions

- virtual void operator() (Image &img) override

  Creates a color inversion image functor.
- virtual RGBAPixel operator() (const Image &img, size_t x, size_t y) override

  Creates a color inversion pixel functor.

Additional Inherited Members

8.10.1 Detailed Description

Class for a color inversion.

For every pixel in a given image, the R, G, B channels are inverted and the alpha is left as is.

See also

ImageShader

8.10.2 Member Function Documentation

8.10.2.1 operator()() [1/2]

```
RGBAPixel image::ColorInvertShader::operator() (
        const Image & img,
        size_t x,
        size_t y )  [override], [virtual]
```

Creates a color inversion pixel functor.

Inverts every pixel in the image by calling invert()

See also

invert()

Parameters

img	Image to sample from
x	X location of target
y	Y location of target

Implements image::ImageShader.

8.10.2.2 operator()() [2/2] void image::ColorInvertShader::operator() (Image & *img*) [override], [virtual]

Creates a color inversion image functor.

Inverts every pixel in the image by calling invert()

See also

invert()

Parameters

img	Image to invert

Reimplemented from image::ImageShader.

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

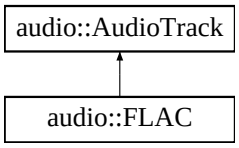
- src/libLMTKimage/colorinvert.h
- src/libLMTKimage/colorinvert.cpp

8.11 audio::FLAC Class Reference

Class for a FLAC file audio track.

`#include "flac.h"`

Inheritance diagram for audio::FLAC:



Public Member Functions

- FLAC (unsigned long length)
  Creates a FLAC of length in samples, at default sample rate.
- FLAC (unsigned long length, unsigned long sample_rate)
  Creates a FLAC of length in samples, at given sample rate.
- FLAC (std::string path)
  Creates FLAC from file on disk.
- bool readFile (std::string path)
  Reads a FLAC file from disk into memory.
- bool **writeFile** (std::string path)

Additional Inherited Members

8.11.1 Detailed Description

Class for a FLAC file audio track.

See also

AudioTrack

8.11.2 Constructor & Destructor Documentation

8.11.2.1 FLAC() [1/3] audio::FLAC::FLAC (unsigned long *length*)

Creates a FLAC of length in samples, at default sample rate.

Parameters

length	Length in audio samples

8.11.2.2 FLAC() [2/3] audio::FLAC::FLAC (unsigned long *length*, unsigned long *sample_rate*)

Creates a FLAC of length in samples, at given sample rate.

Parameters

length	Length in audio samples
sample_rate	Sample rate in hz

8.11.2.3 FLAC() [3/3] audio::FLAC::FLAC (std::string *path*)

Creates FLAC from file on disk.

See also

readFile()

Parameters

path	Path to file

8.11.3 Member Function Documentation

8.11.3.1 readFile() `bool audio::FLAC::readFile ( std::string path )`

Reads a FLAC file from disk into memory.

Parameters

path	Path to file

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

- src/libLMTKaudio/flac.h
- src/libLMTKaudio/flac.cpp

8.12 lmtkimage::fx_info Struct Reference

Public Attributes

- std::function< image::ImageShader *(std::vector< std::string >)> **function**
- std::string **description**
- std::string **usage**

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

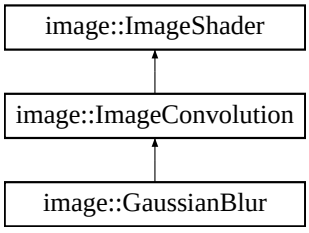
- src/app/tools/imagefx.hpp

8.13 image::GaussianBlur Class Reference

Class for gaussian blur kernel filtering of an image.

`#include "gaussianblur.h"`

Inheritance diagram for image::GaussianBlur:



Public Member Functions

- GaussianBlur (float stdev)

  Creates a gaussian blur with a given standard deviation.
- GaussianBlur (float stdevX, float stdevY)

  Creates gaussian blur with horizontal and vertical standard deviation.
- GaussianBlur (float stdevX, float stdevY, float toleranceX, float toleranceY)

  Creates gaussian blur with standard deviations and tolerances.

Private Member Functions

- virtual void computeKernel () override

  Computes the kernel according to the gaussian function.

Additional Inherited Members

8.13.1 Detailed Description

Class for gaussian blur kernel filtering of an image.

Blurs the image, taking neigboring pixel weighted according to the gaussian/normal distribution function. The result is a smooth, but rather boring looking blur.

See also

utils::math::gaussian()

8.13.2 Constructor & Destructor Documentation

8.13.2.1 GaussianBlur() [1/3]

```
image::GaussianBlur::GaussianBlur (
        float stdev )
```

Creates a gaussian blur with a given standard deviation.

Standard deviation is in pixels. May be fractional.

Parameters

stdev	Standard deviation

8.13.2.2 GaussianBlur() [2/3]

```
image::GaussianBlur::GaussianBlur (
        float stdevX,
        float stdevY )
```

Creates gaussian blur with horizontal and vertical standard deviation.

Standard deviation is in pixels. May be fractional.

Parameters

stdevX	Horizontal standard deviation
stdevY	Vertial standard deviation

8.13.2.3 GaussianBlur() [3/3] image::GaussianBlur::GaussianBlur (float *stdevX*, float *stdevY*, float *toleranceX*, float *toleranceY*)

Creates gaussian blur with standard deviations and tolerances.

Standard deviation is in pixels. May be fractional. Tolerance refers to the value for which any weights smaller than it will not be included in the kernel. A low tolerance will be more accurate, but a cost of a larger kernel (with diminishing return)

Note

Default tolerance is 3 standard deviations (∼99.7%)

Parameters

stdevX	Horizontal standard deviation
stdevY	Vertial standard deviation
toleranceX	Tolerance for kernel columns
toleranceY	Tolerance for kernel rows

8.13.3 Member Function Documentation

8.13.3.1 computeKernel() void image::GaussianBlur::computeKernel ( ) [override], [private], [virtual]

Computes the kernel according to the gaussian function.

See also

utils::math::gaussian2d()

Implements image::ImageConvolution.

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

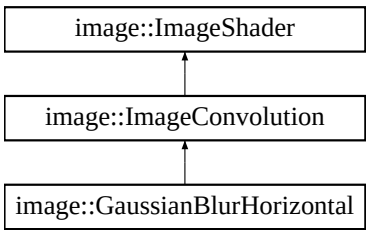
- src/libLMTKimage/gaussianblur.h
- src/libLMTKimage/gaussianblur.cpp

8.14 image::GaussianBlurHorizontal Class Reference

Class for seperable horizontal component of gaussian blur.

`#include "gaussianblur.h"`

Inheritance diagram for image::GaussianBlurHorizontal:



Public Member Functions

- GaussianBlurHorizontal (float stdev)
  Creates a horizontal gaussian blur with a standard deviation.
- GaussianBlurHorizontal (float stdev, float tol)
  Create gaussian blur with horizontal standard deviation and tolerance.

Private Member Functions

- virtual void computeKernel () override
  Computes the kernel according to the gaussian function.

Additional Inherited Members

8.14.1 Detailed Description

Class for seperable horizontal component of gaussian blur.

Blurs the image according to the gaussian horizontally

See also

GaussianBlur

8.14.2 Constructor & Destructor Documentation

8.14.2.1 GaussianBlurHorizontal() [1/2] `image::GaussianBlurHorizontal::GaussianBlurHorizontal ( float stdev )`

Creates a horizontal gaussian blur with a standard deviation.

Standard deviation is in pixels. May be fractional.

Parameters

stdev	Horizontal standard deviation

8.14.2.2 GaussianBlurHorizontal() [2/2]

```
image::GaussianBlurHorizontal::GaussianBlurHorizontal (
        float stdev,
        float tol )
```

Create gaussian blur with horizontal standard deviation and tolerance.

Standard deviation is in pixels. May be fractional. Tolerance refers to the value for which any weights smaller than it will not be included in the kernel.

Parameters

stdev	Horizontal standard deviation
tol	Tolerance for kernel weights

8.14.3 Member Function Documentation

8.14.3.1 computeKernel()

```
void image::GaussianBlurHorizontal::computeKernel ( ) [override], [private], [virtual]
```

Computes the kernel according to the gaussian function.

See also

utils::math::gaussian()

Implements image::ImageConvolution.

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

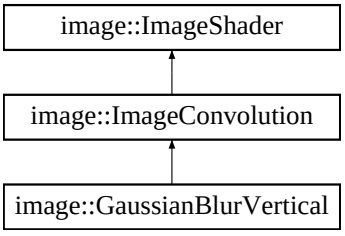
- src/libLMTKimage/gaussianblur.h
- src/libLMTKimage/gaussianblur.cpp

8.15 image::GaussianBlurVertical Class Reference

Class for seperable vertical component of gaussian blur.

`#include "gaussianblur.h"`

Inheritance diagram for image::GaussianBlurVertical:



Public Member Functions

- GaussianBlurVertical (float stdev)

  Creates vertical gaussian blur with a standard deviation.
- GaussianBlurVertical (float stdev, float tol)

  Create gaussian blur with vertical standard deviation and tolerance.

Private Member Functions

- virtual void computeKernel () override

  Computes the kernel according to the gaussian function.

Additional Inherited Members

8.15.1 Detailed Description

Class for seperable vertical component of gaussian blur.

Blurs the image according to the gaussian vertically

See also

GaussianBlur

8.15.2 Constructor & Destructor Documentation

8.15.2.1 GaussianBlurVertical() [1/2] image::GaussianBlurVertical::GaussianBlurVertical (float *stdev*)

Creates vertical gaussian blur with a standard deviation.

Standard deviation is in pixels. May be fractional.

Parameters

stdev	Vertical standard deviation

8.15.2.2 GaussianBlurVertical() [2/2] image::GaussianBlurVertical::GaussianBlurVertical (float *stdev*, float *tol*)

Create gaussian blur with vertical standard deviation and tolerance.

Standard deviation is in pixels. May be fractional. Tolerance refers to the value for which any weights smaller than it will not be included in the kernel.

Parameters

stdev	Vertical standard deviation
tol	Tolerance for kernel weights

8.15.3 Member Function Documentation

8.15.3.1 computeKernel() `void image::GaussianBlurVertical::computeKernel ( ) [override], [private], [virtual]`

Computes the kernel according to the gaussian function.

See also

utils::math::gaussian()

Implements image::ImageConvolution.

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

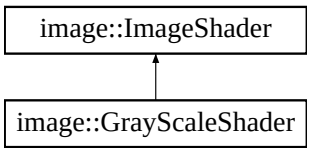
- src/libLMTKimage/gaussianblur.h
- src/libLMTKimage/gaussianblur.cpp

8.16 image::GrayScaleShader Class Reference

Class for a converting any image to a grayscale version.

`#include "grayscale.h"`

Inheritance diagram for image::GrayScaleShader:



Public Member Functions

- virtual void operator() (Image &img) override

  Grayscale each pixel in the image.
- virtual RGBAPixel operator() (const Image &img, size_t x, size_t y) override

  Grayscale a pixel.

Additional Inherited Members

8.16.1 Detailed Description

Class for a converting any image to a grayscale version.

In a grayscale image each pixel is a shade of gray, with only varying intensity. Gray is defined where the RGB components all have the same value. This means if desired, only one color channel would be required.

8.16.2 Member Function Documentation

8.16.2.1 operator()() [1/2]

```
RGBAPixel image::GrayScaleShader::operator() (
        const Image & img,
        size_t x,
        size_t y )  [override], [virtual]
```

Grayscale a pixel.

Parameters

img	Image to select pixel from
x	X location of target pixel
y	Y location of target pixel

Returns

Grayscaled pixel

Implements image::ImageShader.

8.16.2.2 operator()() [2/2]

```
void image::GrayScaleShader::operator() (
        Image & img )  [override], [virtual]
```

Grayscale each pixel in the image.

Parameters

img	Image to grayscale

Reimplemented from image::ImageShader.

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

- src/libLMTKimage/grayscale.h
- src/libLMTKimage/grayscale.cpp

8.17 color::hsl_color Struct Reference

Public Attributes

- double **h**
- double **s**
- double **l**

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

- src/libLMTKimage/color.h

8.18 image::HSLAPixel Class Reference

Class for representing a pixel's color using the HSL color model.

```
#include "hslapixel.h"
```

Public Member Functions

- HSLAPixel ()

  Creates default HSLA Pixel.
- HSLAPixel (float h, float s, float l, float a)

  Creates HSLA Pixel given hue, saturation, lumimance, and alpha.
- HSLAPixel (const RGBAPixel &rgb)

  Converts RGBA Pixel to HSLA.
- void operator= (const HSLAPixel &other)

  Assignment operator: copies other's h, s, l, a values to ∗this.
- bool operator== (const HSLAPixel &other) const

  Equality operator: checks if two HSLA Pixels are equal.
- bool operator!= (const HSLAPixel &other) const

  Inequality operator: check if two HSLA Pixels are not equal.
- void set (float h, float s, float l)

  Sets the H, S, L values.
- void set (float h, float s, float l, float a)

  Sets the H, S, L and A values.
- void set (const HSLAPixel &other)

  Sets all channel values to that of another pixel.
- void set (const RGBAPixel &other)

  Sets HSL values converted from RGB, and alpha.
- void setAlpha (float a)

  Set the alpha value.
- void setTransparency (float transparency)

  Sets the transparency.
- std::string to_string () const

  Returns a string representation of the HSLA.

Public Attributes

- float h

  Represents the amount of hue in degrees [0, 360)
- float s

  Represents the amount of saturation [0, 1].
- float l

  Represents the amount of lightness [0, 1].
- float a

  Represents the alpha [0, 1].

8.18.1 Detailed Description

Class for representing a pixel's color using the HSL color model.

HSL: Hue, Saturation and Luminance. H is the hue, which can be represented by an angle. The resulting "color wheel" is a rainbow spectrum. This implementation uses degrees [0, 360). S is the saturation, which is the "amount of color" or vibrance of the color. A lower value means grey, and a higher value is very bright. L is the luminance, which is the intensity of the light. A lower value is close to black, a higher value is close to white. In addition, there is an alpha layer which is the opacity of the image and can be used to overlay different images when suppported.

See also

color

8.18.2 Constructor & Destructor Documentation

8.18.2.1 HSLAPixel() [1/3] image::HSLAPixel::HSLAPixel ( )

Creates default HSLA Pixel.

Default value is 0 for each channel

8.18.2.2 HSLAPixel() [2/3] image::HSLAPixel::HSLAPixel (

```
float h,
float s,
float l,
float a )
```

Creates HSLA Pixel given hue, saturation, lumimance, and alpha.

Clamps the values.

See also

clampHSLA()

Parameters

h	Hue in degrees (can be negative)
s	Saturation [0, 1]
l	Luminance [0, 1]
a	Alpha [0, 1]

8.18.2.3 HSLAPixel() [3/3] image::HSLAPixel::HSLAPixel (const RGBAPixel & *rgb*)

Converts RGBA Pixel to HSLA.

Parameters

rgb	RGBA Pixel

8.18.3 Member Function Documentation

8.18.3.1 operator"!=() bool image::HSLAPixel::operator!= (const HSLAPixel & *other*) const

Inequality operator: check if two HSLA Pixels are not equal.

See also

operator==()

Parameters

other	Other pixel to check

Returns

True if the two are not equal, false otherwise

8.18.3.2 operator=() void image::HSLAPixel::operator= (const HSLAPixel & *other*)

Assignment operator: copies other's h, s, l, a values to ∗this.

Parameters

other	HSLA Pixel to copy

8.18.3.3 operator==() `bool image::HSLAPixel::operator== ( const HSLAPixel & other ) const`

Equality operator: checks if two HSLA Pixels are equal.

Each channel H, S, L, A must be equal.

Parameters

other	Other pixel to check

Returns

True if the two are equal, false otherwise

8.18.3.4 set() [1/4] `void image::HSLAPixel::set ( const HSLAPixel & other )`

Sets all channel values to that of another pixel.

Equivalent to operator=

Parameters

other	Color to set to

8.18.3.5 set() [2/4] `void image::HSLAPixel::set ( const RGBAPixel & other )`

Sets HSL values converted from RGB, and alpha.

See also

color

Parameters

other	Color to set to

8.18.3.6 set() [3/4] void image::HSLAPixel::set (

```
float h,
float s,
float l )
```

Sets the H, S, L values.

Will clamp values

See also

HSLAClamp()

Parameters

h	Hue value
s	Saturation value
l	Luminance value

8.18.3.7 set() [4/4] void image::HSLAPixel::set (

```
float h,
float s,
float l,
float a )
```

Sets the H, S, L and A values.

Will clamp values

See also

HSLAClamp()

Parameters

h	Hue value
s	Saturation value
l	Luminance value
a	Alpha

8.18.3.8 setAlpha() void image::HSLAPixel::setAlpha (

```
float a )
```

Set the alpha value.

Parameters

a	Alpha

8.18.3.9 setTransparency() `void image::HSLAPixel::setTransparency ( float transparency )`

Sets the transparency.

Transparency is 1.0 - alpha

Parameters

a	Alpha value

8.18.3.10 to_string() `std::string image::HSLAPixel::to_string ( ) const`

Returns a string representation of the HSLA.

Returns

The string representation

8.18.4 Member Data Documentation

8.18.4.1 a `float image::HSLAPixel::a`

Represents the alpha [0, 1].

Warning

Modifying this directly does not clamp the value. Not recommended.

8.18.4.2 h `float image::HSLAPixel::h`

Represents the amount of hue in degrees [0, 360)

Warning

Modifying this directly does not clamp the value. Not recommended.

8.18.4.3 l `float image::HSLAPixel::l`

Represents the amount of lightness [0, 1].

Warning

Modifying this directly does not clamp the value. Not recommended.

8.18.4.4 s `float image::HSLAPixel::s`

Represents the amount of saturation [0, 1].

Warning

Modifying this directly does not clamp the value. Not recommended.

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

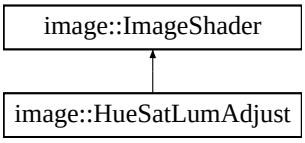
- src/libLMTKimage/hslapixel.h
- src/libLMTKimage/hslapixel.cpp

8.19 image::HueSatLumAdjust Class Reference

Class for adjustin the hue, saturation, and luminance of an image.

`#include "huesatlum.h"`

Inheritance diagram for image::HueSatLumAdjust:



Public Member Functions

- HueSatLumAdjust (float hue, float sat, float lum)

  Creates a HSL Adjust functor with hue shift, saturation and luminance multiplier.
- virtual void operator() (Image &img) override

  Adjust the HSL of an image.
- virtual RGBAPixel operator() (const Image &img, size_t x, size_t y) override

  Return the HSL adjusted color.

Additional Inherited Members

8.19.1 Detailed Description

Class for adjustin the hue, saturation, and luminance of an image.

See also

HSLAPixel

8.19.2 Constructor & Destructor Documentation

8.19.2.1 HueSatLumAdjust() `image::HueSatLumAdjust::HueSatLumAdjust (`

```
float hue,
float sat,
float lum )
```

Creates a HSL Adjust functor with hue shift, saturation and luminance multiplier.

Parameters

hue	Hue shift in degrees
sat	Saturation multiplier
lum	Luminance multiplier

8.19.3 Member Function Documentation

8.19.3.1 operator()() `[1/2]` `RGBAPixel image::HueSatLumAdjust::operator() (`

```
const Image & img,
size_t x,
size_t y )  [override], [virtual]
```

Return the HSL adjusted color.

Parameters

img	Image to select pixel from
x	X location of target pixel
y	Y location of target pixel

Returns

HSL Adjusted color

Implements image::ImageShader.

8.19.3.2 operator()() [2/2] void image::HueSatLumAdjust::operator() (Image & *img*) [override], [virtual]

Adjust the HSL of an image.

Parameters

img	Image to adjust

Reimplemented from image::ImageShader.

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

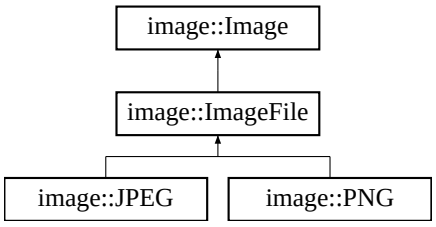
- src/libLMTKimage/huesatlum.h
- src/libLMTKimage/huesatlum.cpp

8.20 image::Image Class Reference

Represents an rectangular, colored image.

`#include <image.h>`

Inheritance diagram for image::Image:



Public Member Functions

- Image ()

  Creates an empty image of size 0 x 0.
- Image (unsigned int w, unsigned int h)

  Creates an image of size width ∗ height.
- Image (const Image &other)

  Creates an image by copying another image's data.
- virtual ∼Image ()

  Deallocates dynamic memory used by image.
- virtual Image operator= (const Image &other)

  Assignment operator: sets this image to another image.
- virtual bool operator== (const Image &other) const

Equality operator: checks if two images are equal.

- virtual bool operator!= (const Image &other) const

  Inequality operator: checks if two images are NOT equal.
- unsigned int width () const

  Get the width of the image in pixels.
- unsigned int height () const

  Get the height of the image in pixels.
- unsigned int size () const

  Get the total size of the image in pixels.
- void fillSolidColor (const RGBAPixel &color)

  Fills the image with the given color.
- void fillGradient (const RGBAPixel &color1, const RGBAPixel &color2)

  Fills the a gradient from color1 to color2.
- void blendNormal (const Image &other)

  Blends the given image on top of this one, including alpha.
- void resize (unsigned int w, unsigned int h)

  Resizes the current image to specified width and height, in pixels.
- void **resize** (unsigned int w, unsigned int h, ResampleMethod method)
- void scale (float scale_x, float scale_y)

  Scales the width and height of the image by a specified amount.
- void **mirror** ()

  Flips the image across an axis.
- RGBAPixel * getRGBAPixel (int x, int y) const

  Returns a pointer to the pixel at (x, y).
- RGBAPixel getRGBAPixel (int x, int y, EdgeHandlingMethod edge_method) const

  Returns pixel at (x, y) if exists, else depends on edge handling.
- RGBAPixel getRGBAPixel (float x, float y, ResampleMethod method) const

  Returns pixel at resampled pixel at (x, y)
- bool setRGBAPixel (const RGBAPixel &color, int x, int y)

  Attempts to set the pixel at (x, y) to the given color.
- bool setHSLAPixel (const HSLAPixel &color, int x, int y)

  Attempts to set the pixel at (x, y) to the given color.
- std::string to_string ()

  Returns a string representation of the image.

Protected Member Functions

- RGBAPixel * **getRow** (int row_num) const

  return pointer to start or row.
- bool **setRow** (RGBAPixel row_pixels[ ], size_t row_width, int row_num)
- RGBAPixel * pixelAt (int x, int y) const

  Returns a pointer to the pixel at (x, y), if exists.
- RGBAPixel **bilinearResample** (float x, float y) const

  Helper for resampling of color at x, y using bilinear resample.
- RGBAPixel **bicubicResample** (float x, float y) const
- void setInitialSize (int w, int h)

  Attempts to set the size of the image and allocates memory for pixel data.
- virtual void copy (const Image &)

  Copies the current image to match the given image.
- virtual void clear ()

  Frees the pixel data, if it is not NULL.

Protected Attributes

- pixel_array_struct **pixel_data**

  The image pixel data.

8.20.1 Detailed Description

Represents an rectangular, colored image.

Image is of size width ∗ height (OR x ∗ y OR #columns ∗ #rows). Stores color/ pixel data in RGBA format and provides basic image creation and modification functions.

Note

The image class does not provide I/O (read/write) functions as it is just a respresentation in memory. For such, use a class for a well defined image file format.

See also

RGBAPixel

8.20.2 Constructor & Destructor Documentation

8.20.2.1 Image() [1/3] image::Image::Image ( )

Creates an empty image of size 0 x 0.

Note

The pixel array is not initialized in this case. Must use setInitialSize() to use image

8.20.2.2 Image() [2/3] image::Image::Image (

```
unsigned int w,
unsigned int h )
```

Creates an image of size width ∗ height.

Each pixel is initialized to the default color with fillSolidColor() The default is transparent.

See also

fillSolidColor()

Parameters

w	Width
h	Height

8.20.2.3 Image() [3/3] image::Image::Image (const Image & other)

Creates an image by copying another image's data.

Parameters

other	Image to copy

8.20.2.4 ~Image() image::Image::~Image ( ) [virtual]

Deallocates dynamic memory used by image.

Frees pixel data, calls clear()

See also

clear()

8.20.3 Member Function Documentation

8.20.3.1 blendNormal() void image::Image::blendNormal (const Image & other)

Blends the given image on top of this one, including alpha.

The current (this) image is modified by blending every pixel with the corresponding color in the other image, proportional to the alpha of each. This is done until the other image has no remaining pixels. If the other image is larger, any remnant pixels are ignored.

Note

If the blending pixel's alpha is 1.0, the current will be replaced

Parameters

color1	The starting color
color2	The end color

8.20.3.2 clear() `void image::Image::clear ( ) [protected], [virtual]`

Frees the pixel data, if it is not NULL.

Sets the pixel data pointer to NULL after completion.

8.20.3.3 copy() `void image::Image::copy ( const Image & other ) [protected], [virtual]`

Copies the current image to match the given image.

Old image data is freed with clear() before allocating space for the size of the image to copy, Then calls setInitialSize()

See also

clear()

setInitialSize()

8.20.3.4 fillGradient() `void image::Image::fillGradient ( const RGBAPixel & color1, const RGBAPixel & color2 )`

Fills the a gradient from color1 to color2.

Uses a linear gradient for each channel indepently with the difference color2 - color1 in each channel R, G, B, A. The channel values may either increase or decrease from color1, depending on if color2's value is larger or smaller.

Parameters

color1	The starting color
color2	The end color

8.20.3.5 fillSolidColor() `void image::Image::fillSolidColor ( const RGBAPixel & color )`

Fills the image with the given color.

Every pixel in the image is set to the given pixel, in RGBA

Parameters

color	The new color

8.20.3.6 getRGBAPixel() [1/3] RGBAPixel image::Image::getRGBAPixel (

```
float x,
float y,
ResampleMethod method ) const
```

Returns pixel at resampled pixel at (x, y)

See also

getRGBAPixel()

Parameters

x	X position of pixel
y	Y position of pixel
method	The resampling method

See also

ResampleMethod

Returns

The sampled pixel at (x, y) using a resample method

8.20.3.7 getRGBAPixel() [2/3] RGBAPixel * image::Image::getRGBAPixel (

```
int x,
int y ) const
```

Returns a pointer to the pixel at (x, y).

See also

pixelAt()

Parameters

x	X position of pixel
y	Y position of pixel

Returns

A pointer to the pixel at (x, y)

Exceptions

an	exception if the pixel does not exist

8.20.3.8 getRGBAPixel() [3/3]

```
RGBAPixel image::Image::getRGBAPixel (
        int x,
        int y,
        EdgeHandlingMethod edge_method ) const
```

Returns pixel at (x, y) if exists, else depends on edge handling.

See also

getRGBAPixel()

Parameters

x	X position of pixel
y	Y position of pixel
edge_method	The edge handling method

See also

EdgeHandlingMethod

Returns

The pixel at (x, y) if exists, else use edge handling method

8.20.3.9 height()

```
unsigned int image::Image::height ( ) const
```

Get the height of the image in pixels.

Returns

Height of the image

8.20.3.10 operator"!=()

```
bool image::Image::operator!= (
        const Image & other ) const  [virtual]
```

Inequality operator: checks if two images are NOT equal.

See also

operator==

Parameters

other	Image to be checked

8.20.3.11 operator=() `Image image::Image::operator= ( const Image & other ) [virtual]`

Assignment operator: sets this image to another image.

If the two images are not the same already, calls copy()

Note

can be overriden to check additional parameters

See also

copy()

8.20.3.12 operator==() `bool image::Image::operator== ( const Image & other ) const [virtual]`

Equality operator: checks if two images are equal.

Two images are equal if they are the same width and height, and if every single pixel is equal.

Note

can be overriden to check additional parameters

Parameters

other	Image to be checked

8.20.3.13 pixelAt() `RGBAPixel * image::Image::pixelAt ( int x, int y ) const [protected]`

Returns a pointer to the pixel at (x, y), if exists.

Parameters

x	X position
y	Y position

Returns

Pointer to pixel at (x, y) or nullptr if out of bounds

8.20.3.14 resize() `void image::Image::resize (`

```
unsigned int w,
unsigned int h )
```

Resizes the current image to specified width and height, in pixels.

Resizing is done using nearest neighbor resampling

Note

Should not be resized to $0 * 0$

Parameters

w	New width
h	New height

8.20.3.15 scale() `void image::Image::scale (`

```
float scale_x,
float scale_y )
```

Scales the width and height of the image by a specified amount.

Calculates the new width and height and calls resize()

See also

resize()

Note

Should not scale by 0

Parameters

scale_x	Width scale
scale_y	Height scale

8.20.3.16 setHSLAPixel() `bool image::Image::setHSLAPixel (`

```
const HSLAPixel & color,
int x,
int y )
```

Attempts to set the pixel at (x, y) to the given color.

See also

pixelAt()

Parameters

color	HSLA Color to set the pixel to
x	X position of pixel
y	Y position of pixel

Returns

True if the color of the pixel was changed, false otherwise

8.20.3.17 setInitialSize() `void image::Image::setInitialSize (`

```
int w,
int h )  [protected]
```

Attempts to set the size of the image and allocates memory for pixel data.

Warning

Should only be used if necessary, and call clear() before using or else will memory leak.

Exceptions

an	exception!!! if the width or height given is 0
an	exception!!! if malloc returns NULL

8.20.3.18 setRGBAPixel() `bool image::Image::setRGBAPixel (`

```
const RGBAPixel & color,
int x,
int y )
```

Attempts to set the pixel at (x, y) to the given color.

See also

pixelAt()

Parameters

color	RGBA Color to set the pixel to
x	X position of pixel
y	Y position of pixel

Returns

True if the color of the pixel was changed, false otherwise

8.20.3.19 size() `unsigned int image::Image::size ( ) const`

Get the total size of the image in pixels.

The size of the image is width $*$ height

Returns

Size of the image

8.20.3.20 to_string() `std::string image::Image::to_string ( )`

Returns a string representation of the image.

Note

This function is subject to change.

Returns

The string representation

8.20.3.21 width() `unsigned int image::Image::width ( ) const`

Get the width of the image in pixels.

Returns

Width of the image

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

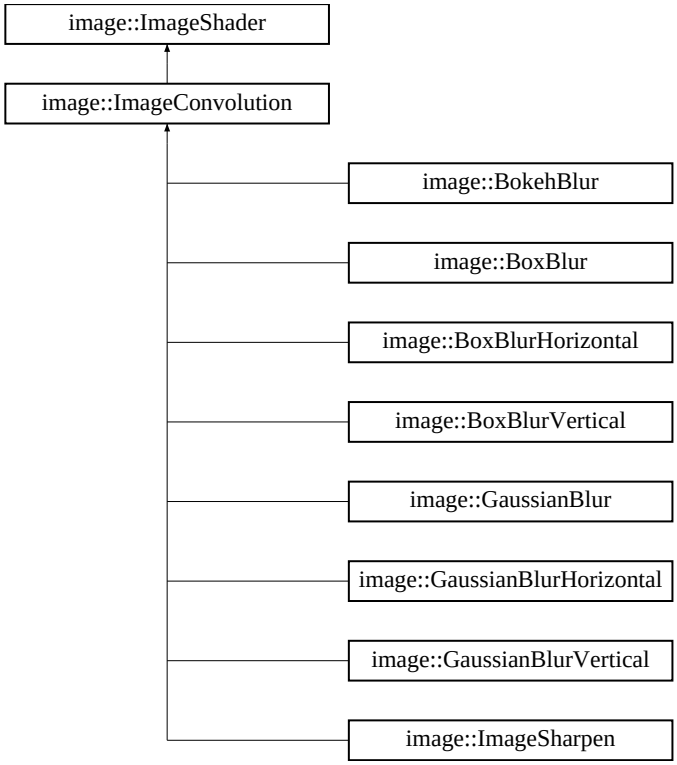
- src/libLMTKimage/image.h
- src/libLMTKimage/image.cpp

8.21 image::ImageConvolution Class Reference

Base class for all image filters using convolution/kernel filtering.

`#include "convolution.h"`

Inheritance diagram for image::ImageConvolution:



Public Member Functions

- ImageConvolution ()

  Creates the default image convolution.
- ImageConvolution (size_t center)

  Creates a image convolution with kernel center at (center, center).
- ImageConvolution (size_t centerX, size_t centerY)

  Creates an image convolution with kernel center at (centerX, centerY).
- ImageConvolution (size_t centerX, size_t centerY, EdgeHandlingMethod edge_method)

  Creates an image convolution with kernel center and edge handling method for boundary pixels convolving outside the image.
- virtual void operator() (Image &img) override

  Convolves the entire image.
- virtual RGBAPixel operator() (const Image &img, size_t x, size_t y) override

  Virtual convolution operator returns the result pixel at x, y.

Protected Member Functions

- void setKernelSize (size_t centerX, size_t centerY)

  Sets the kernel size using a given integral center row and column.
- virtual void computeKernel ()=0

  Abstract function for computing the kernel for derived classes to use.
- virtual RGBAPixel convolution (const Image &img, unsigned int x, unsigned int y)

  Performs convoution multiplication of the kernel and masked pixel matrix around x, y.
- void normalize ()

  Normalized the sum of all weights to 1.

Protected Attributes

- Eigen::MatrixXf **kernel**

  The kernel matrix.

Additional Inherited Members

8.21.1 Detailed Description

Base class for all image filters using convolution/kernel filtering.

Image convolution uses a kernel, which is a m $*$ n matrix of weights. For each pixel in a image, a new pixel is calculated using weights at the corresponding (row, col) in the kernel, where the original pixel is the center.

This implementation guarentees an odd n and m for a well defined center.

Note

Time complexity of one pixel is O(n $*$ m)

8.21.2 Constructor & Destructor Documentation

8.21.2.1 ImageConvolution() [1/4] `image::ImageConvolution::ImageConvolution ( )`

Creates the default image convolution.

A 1 $*$ 1 matrix

8.21.2.2 ImageConvolution() [2/4] `image::ImageConvolution::ImageConvolution ( size_t center )`

Creates a image convolution with kernel center at (center, center).

See also

setKernelSize()

Parameters

center	The center row and column

8.21.2.3 ImageConvolution() [3/4] image::ImageConvolution::ImageConvolution (

```
size_t centerX,
size_t centerY )
```

Creates an image convolution with kernel center at (centerX, centerY).

See also

setKernelSize()

Parameters

centerX	The center column
centerY	The center row

8.21.2.4 ImageConvolution() [4/4] image::ImageConvolution::ImageConvolution (

```
size_t centerX,
size_t centerY,
EdgeHandlingMethod edge_method )
```

Creates an image convolution with kernel center and edge handling method for boundary pixels convolving outside the image.

See also

EdgeHandlingMethod

Parameters

centerX	The center column
centerY	The center row
edge_method	The edge handling method

8.21.3 Member Function Documentation

8.21.3.1 computeKernel() void image::ImageConvolution::computeKernel ( ) [protected], [pure virtual]

Abstract function for computing the kernel for derived classes to use.

Implemented in image::BokehBlur, image::BoxBlur, image::BoxBlurHorizontal, image::BoxBlurVertical, image::GaussianBlur, image::GaussianBlurHorizontal, image::GaussianBlurVertical, and image::ImageSharpen.

8.21.3.2 convolution()

```
RGBAPixel image::ImageConvolution::convolution (
        const Image & img,
        unsigned int x,
        unsigned int y )  [protected], [virtual]
```

Performs convoution multiplication of the kernel and masked pixel matrix around x, y.

Convolution is done by multiplying each kernel entry (row, col) with the corresponding image pixel at (row, col) formed at center (x, y) on the given image. This is done for each channel R, G, B, A

Note

Out of bound pixels needed are handled with edge_method

The resulting color may clip.

8.21.3.3 normalize()

```
void image::ImageConvolution::normalize ( )  [protected]
```

Normalized the sum of all weights to 1.

This guarentees no "loss" or "gain" of color. Optional to use. Some convolutions may not use a normalized kernel.

8.21.3.4 operator()() [1/2]

```
RGBAPixel image::ImageConvolution::operator() (
        const Image & img,
        size_t x,
        size_t y )  [override], [virtual]
```

Virtual convolution operator returns the result pixel at x, y.

Performs convolution using the kernel at x, y on given image with surrounding pixels.

See also

convolution()

Parameters

img	Image to operate over
x	X location of target pixel
y	Y location of target pixel

Returns

Resulting convolved pixel

Implements image::ImageShader.

8.21.3.5 operator()() [2/2]

```
void image::ImageConvolution::operator() (
        Image & img )  [override], [virtual]
```

Convolves the entire image.

Uses the RGBAPixel operator() on each pixel in the given image.

Parameters

img	Image to operate on

Reimplemented from image::ImageShader.

8.21.3.6 setKernelSize()

```
void image::ImageConvolution::setKernelSize (
        size_t centerX,
        size_t centerY )  [protected]
```

Sets the kernel size using a given integral center row and column.

This guarentees and odd number of rows and columns

Parameters

centerX	The index of the central column
centerY	The index of the central row

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

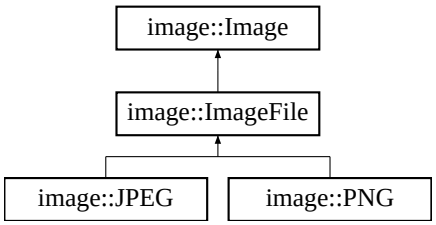
- src/libLMTKimage/convolution.h
- src/libLMTKimage/convolution.cpp

8.22 image::ImageFile Class Reference

class for read/writeable image file.

```
#include "imagefile.h"
```

Inheritance diagram for image::ImageFile:



Public Member Functions

- virtual ~ImageFile ()
  Virtual destructor.
- virtual void readFile (std::string path)=0
  Abstract function for reading an image into memory.
- virtual void writeFile (std::string path)=0
  Abstract function for writing an image to disk.

Protected Member Functions

- ImageFile ()
  Constructs and ImageFile.
- ImageFile (unsigned int w, unsigned int h)
  Creates an image file of size w, h.
- ImageFile (const Image &other)
  Converts an image to to a image file.

Additional Inherited Members

8.22.1 Detailed Description

class for read/writeable image file.

Essentially a interface containing abstract functions for image IO

8.22.2 Constructor & Destructor Documentation

8.22.2.1 ~ImageFile() `virtual image::ImageFile::~ImageFile ( ) [inline], [virtual]`

Virtual destructor.

empty.

8.22.2.2 ImageFile() [1/3] image::ImageFile::ImageFile ( ) [inline], [protected]

Constructs and ImageFile.

Calls base Image constructor

See also

Image

8.22.2.3 ImageFile() [2/3] image::ImageFile::ImageFile (unsigned int *w*, unsigned int *h*) [inline], [protected]

Creates an image file of size w, h.

Call base Image constructor

See also

Image

Parameters

w	Width
h	Height

8.22.2.4 ImageFile() [3/3] image::ImageFile::ImageFile (const Image & *other*) [inline], [protected]

Converts an image to to a image file.

Parameters

other	Image to convert to image file

8.22.3 Member Function Documentation

8.22.3.1 readFile() virtual void image::ImageFile::readFile (std::string *path*) [pure virtual]

Abstract function for reading an image into memory.

Parameters

path	Path to file

Implemented in image::JPEG, and image::PNG.

8.22.3.2 writeFile() `virtual void image::ImageFile::writeFile ( std::string path ) [pure virtual]`

Abstract function for writing an image to disk.

Parameters

path	Path to file

Implemented in image::JPEG, and image::PNG.

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

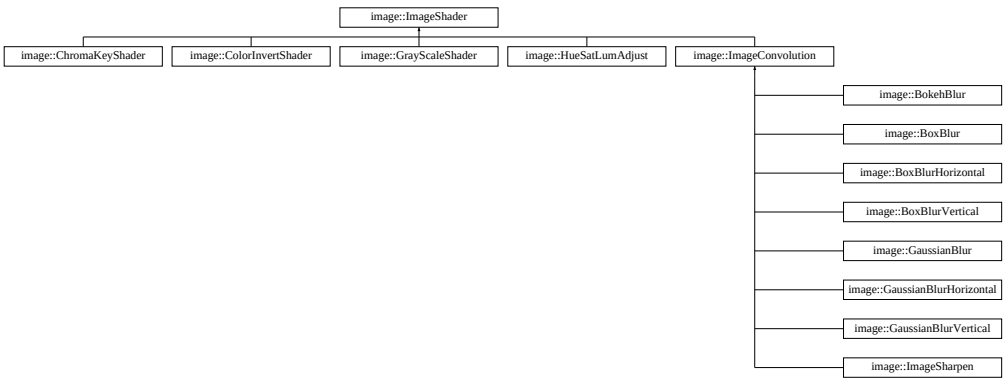
- src/libLMTKimage/imagefile.h

8.23 image::ImageShader Class Reference

Abstract class for an image shader.

`#include "imageshader.h"`

Inheritance diagram for image::ImageShader:



Public Member Functions

- **ImageShader** ()

  Create an image shader with the default edge handling method.
- **ImageShader** (EdgeHandlingMethod method)

  Created an image shader with specified edge handling method.
- virtual ~**ImageShader** ()

  Virtual destructor for an image shader functor.
- virtual void operator() (Image &img)

  Image operator, applies the pixel shader on the whole image.
- virtual RGBAPixel operator() (const Image &img, size_t x, size_t y)=0

  Abstract pixel shader.

Static Public Attributes

- static const EdgeHandlingMethod **DEFAULT_EDGE_METHOD** = EdgeHandlingMethod::EXTEND

  < *Default edge handling method*

Protected Attributes

- EdgeHandlingMethod edge_method

  Edge handling method.

8.23.1 Detailed Description

Abstract class for an image shader.

An image shader operates on a given image with any possible function(s) on any desired pixel and the image is modified per such function(s).

8.23.2 Member Function Documentation

8.23.2.1 operator()() [1/2]

```
virtual RGBAPixel image::ImageShader::operator() (
        const Image & img,
        size_t x,
        size_t y )  [pure virtual]
```

Abstract pixel shader.

Returns a single pixel given a location. Can be overriden do anything you want with the image.

Implemented in image::ChromaKeyShader, image::ColorInvertShader, image::ImageConvolution, image::GrayScaleShader, and image::HueSatLumAdjust.

8.23.2.2 operator()() [2/2] void image::ImageShader::operator() (Image & *img*) [virtual]

Image operator, applies the pixel shader on the whole image.

Can be overriden, but provides basic one-pass shade without extra memory.

Reimplemented in image::ChromaKeyShader, image::ColorInvertShader, image::ImageConvolution, image::GrayScaleShader, and image::HueSatLumAdjust.

8.23.3 Member Data Documentation

8.23.3.1 edge_method EdgeHandlingMethod image::ImageShader::edge_method [protected]

Edge handling method.

See also

EdgeHandlingMethod

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

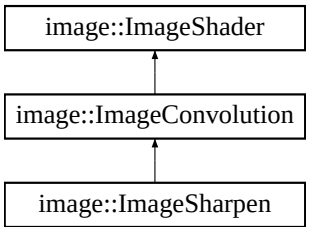
- src/libLMTKimage/imageshader.h
- src/libLMTKimage/imageshader.cpp

8.24 image::ImageSharpen Class Reference

class for sharpening an image

`#include "sharpen.h"`

Inheritance diagram for image::ImageSharpen:



Public Member Functions

- **ImageSharpen** ()

  Create a sharpening functor with default sharpnening amount.
- ImageSharpen (float amt)

  Create a sharpening functor with sharpening amount.

Static Public Attributes

- static const size_t **SHARP_CENTER** = 1

Private Member Functions

- virtual void computeKernel () override

  Computes the kernel using laplcian filter plus identity.

Additional Inherited Members

8.24.1 Detailed Description

class for sharpening an image

Image sharpening is done by emphasizing color differences (edges) This implementation uses a lapacian filter + identity

8.24.2 Constructor & Destructor Documentation

8.24.2.1 ImageSharpen() `image::ImageSharpen::ImageSharpen ( float amt )`

Create a sharpening functor with sharpening amount.

Parameters

amt	Amount to sharpen

8.24.3 Member Function Documentation

8.24.3.1 computeKernel() `void image::ImageSharpen::computeKernel ( ) [override], [private], [virtual]`

Computes the kernel using laplcian filter plus identity.

Implements image::ImageConvolution.

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

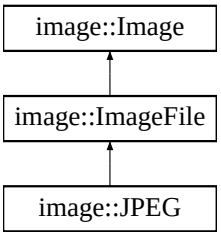
- src/libLMTKimage/sharpen.h
- src/libLMTKimage/sharpen.cpp

8.25 image::JPEG Class Reference

JPEG image class.

`#include <jpeg.h>`

Inheritance diagram for image::JPEG:



Public Member Functions

- JPEG (unsigned int w, unsigned int h)
  Creates a JPEG image of size width ∗ height.
- JPEG (std::string path)
  Creates a JPEG in memory from a jpeg/jpg file stored on disk.
- **JPEG** (const Image &img)
  Converts an image to a writable JPEG.
- ∼**JPEG** ()
  Does nothing right now.
- void readFile (std::string path) override
  Read JPEG file from disk.
- void writeFile (std::string path) override
  Writes image to new or exisiting JPEG file on disk.

Additional Inherited Members

8.25.1 Detailed Description

JPEG image class.

The JPEG class has all image functionality plus read/write capability, and additional JPEG specific tools.

See also

Image

8.25.2 Constructor & Destructor Documentation

8.25.2.1 JPEG() [1/2]

```
image::JPEG::JPEG (
        unsigned int w,
        unsigned int h )
```

Creates a JPEG image of size width ∗ height.

Calls base class constructor

See also

Image

Parameters

w	Width
h	Height

8.25.2.2 JPEG() [2/2] image::JPEG::JPEG (std::string *path*)

Creates a JPEG in memory from a jpeg/jpg file stored on disk.

See also

readFile()

Parameters

path	Path to file

Exceptions

Exception	if could not read file

8.25.3 Member Function Documentation

8.25.3.1 readFile() void image::JPEG::readFile (std::string *path*) [override], [virtual]

Read JPEG file from disk.

Attempt to open file for read, and replaces existing data, if any. Converts byte data from libjpeg to RGBAPixels.

Parameters

path	Path to file to read

Exceptions

An	exception if failed to read (i.e. file does not exist)

Implements image::ImageFile.

8.25.3.2 writeFile() `void image::JPEG::writeFile ( std::string path ) [override], [virtual]`

Writes image to new or exisiting JPEG file on disk.

Directory must exist. Converts RGBAPixel data to jpeg byte data for libjpeg.

Parameters

Path	to write file

Exceptions

An	exception if failed to write

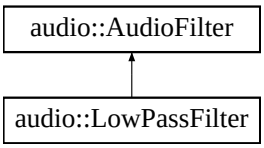
Implements image::ImageFile.

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

- src/libLMTKimage/jpeg.h
- src/libLMTKimage/jpeg.cpp

8.26 audio::LowPassFilter Class Reference

Inheritance diagram for audio::LowPassFilter:



Public Member Functions

- **LowPassFilter** (float cutoff)
- void operator() (AudioTrack &track)

  Abstract operator to an audio track.

8.26.1 Member Function Documentation

8.26.1.1 operator()() `void audio::LowPassFilter::operator() ( AudioTrack & track ) [virtual]`

Abstract operator to an audio track.

Parameters

track	Audio track to filter

Implements audio::AudioFilter.

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

- src/libLMTKaudio/lowpassfilter.h
- src/libLMTKaudio/lowpassfilter.cpp

8.27 image::PluginChain Class Reference

Class for applying multiple effects to an image, in order.

Public Member Functions

- **PluginChain** ()
  Creates an empty plugin chain.
- ~**PluginChain** ()
  Frees memory from each plugin after complete.
- virtual void operator() (Image &img)
  Applies every effect on an image, in order.
- ImageShader * operator[ ] (size_t i)
  Access the ith effect in the plugin chain.
- void addEffect (ImageShader *fx, size_t passes)
  Add an effect to the plugin chain, n times.
- void addEffect (ImageShader *fx)
  Add an effect to plugin chain.
- size_t size ()
  Returns the number of effects in the plugin chain.
- bool empty ()
  Check if the plugin chain is empty.

8.27.1 Detailed Description

Class for applying multiple effects to an image, in order.

Maintains a ordered list of image effects to apply to an image. Individual effects can be accessed or you can apply them all at once.

See also

ImageShader

8.27.2 Member Function Documentation

8.27.2.1 addEffect() [1/2] void image::PluginChain::addEffect (ImageShader * *fx*)

Add an effect to plugin chain.

Adds the effect once.

Parameters

fx	Effect to add

8.27.2.2 addEffect() [2/2] void image::PluginChain::addEffect (ImageShader * fx, size_t passes)

Add an effect to the plugin chain, n times.

Parameters

fx	Effect to add
passes	Number of times to apply the effect in sequence

8.27.2.3 empty() bool image::PluginChain::empty ( )

Check if the plugin chain is empty.

Returns

True if it is empty, false otherwise

8.27.2.4 operator()() void image::PluginChain::operator() (Image & img) [virtual]

Applies every effect on an image, in order.

Parameters

img	Image to apply effects to

8.27.2.5 operator[]() ImageShader * image::PluginChain::operator[] (size_t i)

Access the ith effect in the plugin chain.

Parameters

i	Index of effect to access

8.27.2.6 size() `size_t image::PluginChain::size ( )`

Returns the number of effects in the plugin chain.

Returns

The size of the effect chain

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

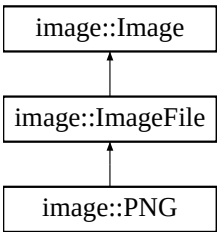
- src/libLMTKimage/pluginchain.h
- src/libLMTKimage/pluginchain.cpp

8.28 image::PNG Class Reference

PNG image class.

`#include <png.h>`

Inheritance diagram for image::PNG:



Public Member Functions

- PNG (unsigned int w, unsigned int h)

  Creates a PNG image of size width ∗ height.
- PNG (std::string path)

  Creates a PNG from a file.
- **PNG** (const Image &other)
- ∼**PNG** () override

  Right now it does nothing.
- void readFile (std::string path) override

  Read PNG data from a file from disk.
- void writeFile (std::string path) override

  Writes the PNG to a file.

Additional Inherited Members

8.28.1 Detailed Description

PNG image class.

The PNG class has all image functionality plus read/write capability, and additional PNG specific tools.

See also

Image

8.28.2 Constructor & Destructor Documentation

8.28.2.1 PNG() [1/2]

```
image::PNG::PNG (
        unsigned int w,
        unsigned int h )
```

Creates a PNG image of size width ∗ height.

public interface functions

Calls base class constructor

See also

Image

Parameters

w	Width
h	Height

8.28.2.2 PNG() [2/2]

```
image::PNG::PNG (
        std::string path )
```

Creates a PNG from a file.

Creates a PNG file and converts all image data to the representation in the Image and PNG classes.

See also

Image

readFile();

Parameters

path	Path of file to open

Exceptions

Exception	if readFile() throws an exception

8.28.3 Member Function Documentation

8.28.3.1 readFile() `void image::PNG::readFile ( std::string path ) [override], [virtual]`

Read PNG data from a file from disk.

Attempts to open and replace existing PNG data if the file given is a valid PNG. Then converts all byte data from the libpng to RGBA pixels.

See also

isFilePNG()

RGBAPixel

Parameters

path	Path of file to open

Exceptions

Invalid	argument exception if the file is not a PNG or does not exist
An	exception if failed to read // TODO

Implements image::ImageFile.

8.28.3.2 writeFile() `void image::PNG::writeFile ( std::string path ) [override], [virtual]`

Writes the PNG to a file.

Attempts to create/open a file if the directory exists, then reverse pixel data into byte data for libpng to use.

See also

RGBAPixel

Note

libpng handles compression

Parameters

path	Path of file to open

Exceptions

Invalid	argument exception if the file is not a PNG // TODO
An	exception filed to write // TODO

Implements image::ImageFile.

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

- src/libLMTKimage/png.h
- src/libLMTKimage/png.cpp

8.29 color::rgb_color Struct Reference

Public Attributes

- double **r**
- double **g**
- double **b**

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

- src/libLMTKimage/color.h

8.30 image::RGBAPixel Class Reference

Represents pixels with color channels red, green, blue and alpha.

```
#include <rgbapixel.h>
```

Public Member Functions

- RGBAPixel ()

  Constructs a default RGBA Pixel.
- RGBAPixel (float r, float g, float b)

  Constructs a RGBA pixels given float values for each channel.
- RGBAPixel (float r, float g, float b, float a)

  Constructs a RGBA pixels given float values for each channel.
- RGBAPixel (uint32_t r, uint32_t g, uint32_t b, uint32_t a, uint8_t bit_depth)

  Constructs a RGBA pixel given unsigned integer values.
- RGBAPixel (const HSLAPixel &hsl)

  Converts a HSLA Pixel to a RGBA Pixel.
- void operator= (const RGBAPixel &other)

  Assignment operator: sets the current RGBA to the other pixel's values.
- bool operator== (const RGBAPixel &other) const

  Equality operator: returns true if two colors are equal.
- bool operator!= (const RGBAPixel &other) const

  Equality operator: returns true if two colors are NOT equal.
- void set (float r, float g, float b)

  Sets the color channels values, RGB.
- void set (float r, float g, float b, float a)

  Sets all channel values, RGBA.
- void set (const RGBAPixel &other)

  Sets all channel values to that of another pixel.
- void set (const HSLAPixel &other)

  Sets the RGB values converted from a HSL, and alpha.
- void setAlpha (float a)

  Sets the alpha value.
- void setTransparency (float transparency)

  Sets the transparency.
- float dist (const RGBAPixel &other) const

  Returns the euclidian distance between two colors.
- std::string to_string () const

  Returns a string representation of the RGBA.

Static Public Member Functions

- static RGBAPixel average (RGBAPixel p1, RGBAPixel p2)

  Computes average RGBA values from 2 pixels.
- static RGBAPixel naverage (RGBAPixel pixels[ ], size_t count)

  Computes average RGBA values from n pixels.
- static RGBAPixel weighted_average (RGBAPixel p1, RGBAPixel p2, float w1, float w2)

  Computes weighted average RGBA values from 2 pixels.
- static RGBAPixel weighted_naverage (RGBAPixel pixels[ ], float weights[ ], size_t count)

  Computes weighted average RGBA values from n pixels.

Public Attributes

- float r

  Represents the amount of red [0, 1].
- float g

  Represents the amount of green [0, 1].
- float b

  Represents the amount of blue [0, 1].
- float a

  Represents the alpha [0, 1].

8.30.1 Detailed Description

Represents pixels with color channels red, green, blue and alpha.

The RGBA color model is an additive color model, where red, green and blue are the primary colors which can be combined to create a wide variety of other colors. White is created when each of the three colors are at maximum intensity, and black is when all three colors have 0 intensity. An additonal channel, alpha represents the opacity of the color which allows the color to be blended with another (known as alpha blending) to create a transparency effect.

See also

color

8.30.2 Constructor & Destructor Documentation

8.30.2.1 RGBAPixel() [1/5] image::RGBAPixel::RGBAPixel ( )

Constructs a default RGBA Pixel.

The default is 0 for each channel.

8.30.2.2 RGBAPixel() [2/5] image::RGBAPixel::RGBAPixel (

```
float r,
float g,
float b )
```

Constructs a RGBA pixels given float values for each channel.

Sets the alpha to 1.0 Each channel is restricted to [0, 1]

See also

clampRGBA()

Parameters

r	Red value
g	Green value
b	Blue value

8.30.2.3 RGBAPixel() [3/5] image::RGBAPixel::RGBAPixel (

```
float r,
float g,
float b,
float a )
```

Constructs a RGBA pixels given float values for each channel.

Each channel is restricted to [0, 1]

See also

clampRGBA()

Parameters

r	Red value
g	Green value
b	Blue value
a	Alpha value

8.30.2.4 RGBAPixel() [4/5] image::RGBAPixel::RGBAPixel (

```
uint32_t r,
uint32_t g,
uint32_t b,
uint32_t a,
uint8_t bit_depth )
```

Constructs a RGBA pixel given unsigned integer values.

RGBA is traditonally defined with an unsigned integer [0, 255] with 8 bit channels, but can accept other bit depths as well.

Note

the maximum bit depth accepted is 32 (integer/dword)

Parameters

r	Red value
g	Green value
b	Blue value
a	Alpha value
bit_depth	Bit depth of all the channels

Note

All channels are assumed to have the same bit depth

8.30.2.5 RGBAPixel() [5/5] image::RGBAPixel::RGBAPixel (const HSLAPixel & hsl)

Converts a HSLA Pixel to a RGBA Pixel.

See also

color

Parameters

hsl	HSLA Pixel to convert

8.30.3 Member Function Documentation

8.30.3.1 average() RGBAPixel image::RGBAPixel::average (RGBAPixel p1, RGBAPixel p2) [static]

Computes average RGBA values from 2 pixels.

Parameters

p1	First pixel
p2	Second Pixel

8.30.3.2 dist() float image::RGBAPixel::dist (const RGBAPixel & other) const

Returns the euclidian distance between two colors.

The eulician distance is the pythagorean sum of the three colored channels (not including alpha)
dist = sqrt(dRED^2 + dGREEN^2 + dBLUE^2)

Note

The max distance is therefore sqrt(3)

Parameters

other	Pixel to compare to

Returns

Distance between the colors

8.30.3.3 naverage() `RGBAPixel image::RGBAPixel::naverage ( RGBAPixel pixels[], size_t count ) [static]`

Computes average RGBA values from n pixels.

Parameters

pixels	Pixels to average
count	Number of pixels in array

8.30.3.4 operator"!=() `bool image::RGBAPixel::operator!= ( const RGBAPixel & other ) const`

Equality operator: returns true if two colors are NOT equal.

See also

operator==()

Parameters

other	Other RGBA Pixel to be checked

Returns

True if the colors are not equal, false if they are equal

8.30.3.5 operator=() `void image::RGBAPixel::operator= ( const RGBAPixel & other )`

Assignment operator: sets the current RGBA to the other pixel's values.

Parameters

other	RGBA Pixel to copy

8.30.3.6 operator==() `bool image::RGBAPixel::operator== ( const RGBAPixel & other ) const`

Equality operator: returns true if two colors are equal.

Each channel must have the same value, including alpha

Parameters

other	Other RGBA Pixel to be checked

Returns

True if the colors are equal, false otherwise

8.30.3.7 set() [1/4] `void image::RGBAPixel::set ( const HSLAPixel & other )`

Sets the RGB values converted from a HSL, and alpha.

See also

color

8.30.3.8 set() [2/4] `void image::RGBAPixel::set ( const RGBAPixel & other )`

Sets all channel values to that of another pixel.

Equivalent to operator=

Parameters

other	Color to set to

8.30.3.9 set() [3/4] `void image::RGBAPixel::set (`

```
            float r,
            float g,
            float b )
```

Sets the color channels values, RGB.

Using this function will also clamp the values if not [0, 1].

See also

clamp()

Parameters

r	Red value
g	Green value
b	Blue value

8.30.3.10 set() [4/4] void image::RGBAPixel::set (

```
            float r,
            float g,
            float b,
            float a )
```

Sets all channel values, RGBA.

Using this function will also clamp the values if not [0, 1].

See also

clamp()

Parameters

r	Red value
g	Green value
b	Blue value
a	Alpha value

8.30.3.11 setAlpha() void image::RGBAPixel::setAlpha (

```
            float a )
```

Sets the alpha value.

Parameters

a	Alpha value

8.30.3.12 setTransparency() `void image::RGBAPixel::setTransparency ( float transparency )`

Sets the transparency.

Transparency is 1.0 - alpha

Parameters

a	Alpha value

8.30.3.13 to_string() `std::string image::RGBAPixel::to_string ( ) const`

Returns a string representation of the RGBA.

Returns

The string representation

8.30.3.14 weighted_average() `RGBAPixel image::RGBAPixel::weighted_average ( RGBAPixel p1, RGBAPixel p2, float w1, float w2 ) [static]`

Computes weighted average RGBA values from 2 pixels.

Parameters

p1	First pixel
p2	Second Pixel
w1	Weight for p1
w2	Weight for p2

8.30.3.15 weighted_naverage() `RGBAPixel image::RGBAPixel::weighted_naverage ( RGBAPixel pixels[], float weights[], size_t count ) [static]`

Computes weighted average RGBA values from n pixels.

Parameters

pixels	Pixels to average
weights	Weights corresponding to pixel[i]
count	Number of pixels in array

8.30.4 Member Data Documentation

8.30.4.1 a `float image::RGBAPixel::a`

Represents the alpha [0, 1].

Warning

Modifying this directly does not clamp the value. Not recommended.

8.30.4.2 b `float image::RGBAPixel::b`

Represents the amount of blue [0, 1].

Warning

Modifying this directly does not clamp the value. Not recommended.

8.30.4.3 g `float image::RGBAPixel::g`

Represents the amount of green [0, 1].

Warning

Modifying this directly does not clamp the value. Not recommended.

8.30.4.4 r `float image::RGBAPixel::r`

Represents the amount of red [0, 1].

Warning

Modifying this directly does not clamp the value. Not recommended.

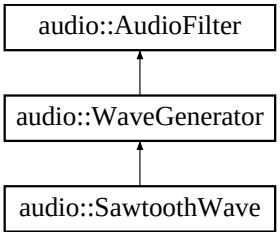
The documentation for this class was generated from the following files:

- src/libLMTKimage/rgbapixel.h
- src/libLMTKimage/rgbapixel.cpp

8.31 audio::SawtoothWave Class Reference

Class for generating a sawtooth wave.

Inheritance diagram for audio::SawtoothWave:



Public Member Functions

- SawtoothWave (float freql, float freqr, float amp)

  Create a sawtooth wave given frequency and amplitude.
- SawtoothWave (float freql, float freqr, float amp, float offset)

  Create a sawtooth wave given frequency, amplitude and offset.
- void operator() (AudioTrack &track)

  Sawtooth operator.

Additional Inherited Members

8.31.1 Detailed Description

Class for generating a sawtooth wave.

A sawtooth wave is a triangular wave with a linear rise, and sharp fall.

See also

WaveGenerator

8.31.2 Constructor & Destructor Documentation

8.31.2.1 SawtoothWave() [1/2]

```
audio::SawtoothWave::SawtoothWave (
        float freql,
        float freqr,
        float amp )
```

Create a sawtooth wave given frequency and amplitude.

Parameters

freql	Left channel frequency
freqr	Right channel frequency
amp	Amplitude

8.31.2.2 SawtoothWave() [2/2] audio::SawtoothWave::SawtoothWave (

```
float freql,
float freqr,
float amp,
float offset )
```

Create a sawtooth wave given frequency, amplitude and offset.

Parameters

freql	Left channel frequency
freqr	Right channel frequency
amp	Amplitude
offset	Offset from 0

8.31.3 Member Function Documentation

8.31.3.1 operator()() void audio::SawtoothWave::operator() (

```
AudioTrack & track ) [virtual]
```

Sawtooth operator.

Creates a sawtooth wave over the track, overwriting existing data

Implements audio::WaveGenerator.

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

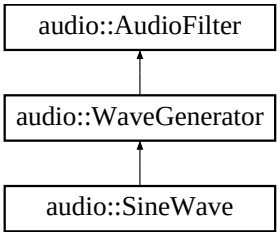
- src/libLMTKaudio/sawtooth.h
- src/libLMTKaudio/sawtooth.cpp

8.32 audio::SineWave Class Reference

Class for generating a sine wave.

`#include "sinewave.h"`

Inheritance diagram for audio::SineWave:



Public Member Functions

- SineWave (float freql, float freqr, float amp)

  Create a sine wave given frequency and amplitude.
- SineWave (float freql, float freqr, float amp, float offset)

  Create a sine wave given frequency, amplitude and offset.
- void operator() (AudioTrack &track)

  Sine wave operator.

Additional Inherited Members

8.32.1 Detailed Description

Class for generating a sine wave.

See also

WaveGenerator

8.32.2 Constructor & Destructor Documentation

8.32.2.1 SineWave() [1/2]

```
audio::SineWave::SineWave (
        float freql,
        float freqr,
        float amp )
```

Create a sine wave given frequency and amplitude.

Parameters

freql	Left channel frequency
freqr	Right channel frequency
amp	Amplitude

8.32.2.2 SineWave() [2/2]

```
audio::SineWave::SineWave (
        float freql,
        float freqr,
        float amp,
        float offset )
```

Create a sine wave given frequency, amplitude and offset.

Parameters

freql	Left channel frequency

Parameters

freqr	Right channel frequency
amp	Amplitude
offset	Offset from 0

8.32.3 Member Function Documentation

8.32.3.1 operator()() `void audio::SineWave::operator() ( AudioTrack & track ) [virtual]`

Sine wave operator.

Creates a sine wave over the track, overwriting existing data

Implements audio::WaveGenerator.

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

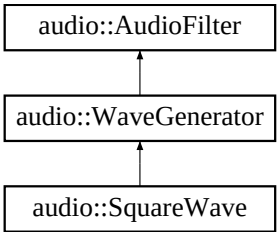
- src/libLMTKaudio/sinewave.h
- src/libLMTKaudio/sinewave.cpp

8.33 audio::SquareWave Class Reference

Class for generating a square wave.

`#include "squarewave.h"`

Inheritance diagram for audio::SquareWave:



Public Member Functions

- SquareWave (float freql, float freqr, float amp)
  Create a square wave given frequency and amplitude.
- SquareWave (float freql, float freqr, float amp, float offset)
  Create a square wave given frequency, amplitude and offset.
- void operator() (AudioTrack &track)
  Square wave operator.

Additional Inherited Members

8.33.1 Detailed Description

Class for generating a square wave.

See also

WaveGenerator

8.33.2 Constructor & Destructor Documentation

8.33.2.1 SquareWave() [1/2]

```
audio::SquareWave::SquareWave (
        float freql,
        float freqr,
        float amp )
```

Create a square wave given frequency and amplitude.

Parameters

freql	Left channel frequency
freqr	Right channel frequency
amp	Amplitude

8.33.2.2 SquareWave() [2/2]

```
audio::SquareWave::SquareWave (
        float freql,
        float freqr,
        float amp,
        float offset )
```

Create a square wave given frequency, amplitude and offset.

Parameters

freql	Left channel frequency
freqr	Right channel frequency
amp	Amplitude
offset	Offset from 0

8.33.3 Member Function Documentation

8.33.3.1 operator()() `void audio::SquareWave::operator() ( AudioTrack & track ) [virtual]`

Square wave operator.

Creates a square wave over the track, overwriting existing data

Implements audio::WaveGenerator.

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

- src/libLMTKaudio/squarewave.h
- src/libLMTKaudio/squarewave.cpp

8.34 threadpool::task Struct Reference

A task for a thread which stores a void calback.

`#include "threadpool.h"`

Public Attributes

- std::function< void()> **callback**

8.34.1 Detailed Description

A task for a thread which stores a void calback.

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

- src/utils/threadpool.h

8.35 threadpool::TaskQueue Class Reference

A FIFO queue of callback tasks that is thread-safe.

`#include "threadpool.h"`

Public Member Functions

- **TaskQueue** ()
  Creates an empty task queue.
- void enqueue (const task &t)
  Pushes a task onto the task queue.
- task dequeue ()
  Pop the oldest task on the queue.
- bool empty () const
  Checks if the task queue is empty.
- size_t size () const
  Get the size of the task queue.

8.35.1 Detailed Description

A FIFO queue of callback tasks that is thread-safe.

8.35.2 Member Function Documentation

8.35.2.1 dequeue() `task threadpool::TaskQueue::dequeue ( )`

Pop the oldest task on the queue.

Removes the task from the queue

Returns

Oldest task on the queue.

8.35.2.2 empty() `bool threadpool::TaskQueue::empty ( ) const`

Checks if the task queue is empty.

Returns

True if empty, false otherwise

8.35.2.3 enqueue() `void threadpool::TaskQueue::enqueue ( const task & t )`

Pushes a task onto the task queue.

Parameters

t	Task to push onto queue

8.35.2.4 size() `size_t threadpool::TaskQueue::size ( ) const`

Get the size of the task queue.

Returns

Size of the task queue

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

- src/utils/threadpool.h
- src/utils/threadpool.cpp

8.36 threadpool::ThreadPool Class Reference

Class for a pool of flexible worker threads which can be given miscellaneous tasks to complete.

```
#include "threadpool.h"
```

Public Member Functions

- ThreadPool ()
  Creates a threadpool with the default number of threads.
- ThreadPool (size_t num_threads)
  Creates a threadpool with n threads.
- ~ThreadPool ()
  Deletes all threads.
- void start ()
  Starts the threads into worker thread main loop.
- void join ()
  Join the threadpool after the tasks are completed.
- void stop ()
  Requests threads to stop execution, regardless of tasks remaining.
- void detach ()
  Detach all threads in the pool, allowing threads to execute independent of main thread.
- template<typename ... Args>
  void addTask (std::function< void(Args...)> callback, Args... args)
  Adds a task for the threadpool given callback and arguments.
- template<typename ... Args>
  void addTask (void(*callback)(Args...), Args... args)
  Adds a task for the threadpool given callback and arguments.
- size_t tasks_remaining () const
  Get number of tasks remaining, when called.
- size_t tasks_completed () const
  Returns the total number of tasks completed by the threadpool when called.

8.36.1 Detailed Description

Class for a pool of flexible worker threads which can be given miscellaneous tasks to complete.

Any sized pool of worker threads can be created, but it is not recommended to create more than hardware support unless tasks require lots of waiting, such as memory or disk read/writes.

8.36.2 Constructor & Destructor Documentation

8.36.2.1 ThreadPool() [1/2] threadpool::ThreadPool::ThreadPool ( )

Creates a threadpool with the default number of threads.

Checks hardware concurrency for number of threads supported.

8.36.2.2 ThreadPool() [2/2] threadpool::ThreadPool::ThreadPool (size_t *num_threads*)

Creates a threadpool with n threads.

Parameters

num_threads	Number of threads to add to pool

8.36.2.3 ~ThreadPool() threadpool::ThreadPool::~ThreadPool ( )

Deletes all threads.

Warning

Should not be called before join if the pool has already been started.

8.36.3 Member Function Documentation

8.36.3.1 addTask() [1/2] template<typename ... Args> void threadpool::ThreadPool::addTask (std::function< void(Args...)> *callback*, Args... *args*) [inline]

Adds a task for the threadpool given callback and arguments.

Add using std::function

Template Parameters

Args	Variadic arguments

Parameters

callback	Callback function for threads to execute
args	Variadic arguments to call callback with Basically, you can call with any arguments you want // !!! I don't think it handles references very well

8.36.3.2 addTask() [2/2] template<typename ... Args>

```
void threadpool::ThreadPool::addTask (
            void(*)(Args...)  callback,
            Args...  args )  [inline]
```

Adds a task for the threadpool given callback and arguments.

Add using c-style callback (function pointer)

Template Parameters

Args	Variadic arguments

Parameters

callback	Callback function for threads to execute
args	Variadic arguments to call callback with Basically, you can call with any arguments you want // !!! I don't think it handles references very well

8.36.3.3 detach() void threadpool::ThreadPool::detach ( )

Detach all threads in the pool, allowing threads to execute independent of main thread.

Threads will no longer be joinable.

8.36.3.4 join() void threadpool::ThreadPool::join ( )

Join the threadpool after the tasks are completed.

Will join each thread as long as joinable (i.e. not detached) Threads will exit when there are no more tasks remaining.

Note

tasks can no longer be added from the thread will calls join as the thread will be blocked.

8.36.3.5 start() `void threadpool::ThreadPool::start ( )`

Starts the threads into worker thread main loop.

The threads will run all tasks are completed, unless stopped

See also

stop()

8.36.3.6 stop() `void threadpool::ThreadPool::stop ( )`

Requests threads to stop execution, regardless of tasks remaining.

Threads will exit when finished their current task, if any. Then joins the threadpool

Note

Not a force terminate, will not save infinite loops, deadlocks or other indefinite execution.

8.36.3.7 tasks_completed() `size_t threadpool::ThreadPool::tasks_completed ( ) const`

Returns the total number of tasks completed by the threadpool when called.

Returns

Cumulative number of tasks completed

8.36.3.8 tasks_remaining() `size_t threadpool::ThreadPool::tasks_remaining ( ) const`

Get number of tasks remaining, when called.

Returns

Number of tasks remaining on queue

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

- src/utils/threadpool.h
- src/utils/threadpool.cpp

8.37 utils::Timer Class Reference

Represents a timer.

`#include "timer.h"`

Public Member Functions

- **Timer** ()
  Creates a timer object.
- void start ()
  Starts the timer.
- double stop ()
  Stops the timer, returning the interval from start time.

8.37.1 Detailed Description

Represents a timer.

8.37.2 Member Function Documentation

8.37.2.1 start() `void utils::Timer::start ( )`

Starts the timer.

Marks the start time.

8.37.2.2 stop() `double utils::Timer::stop ( )`

Stops the timer, returning the interval from start time.

Returns

Interval from last call to start on ∗this

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

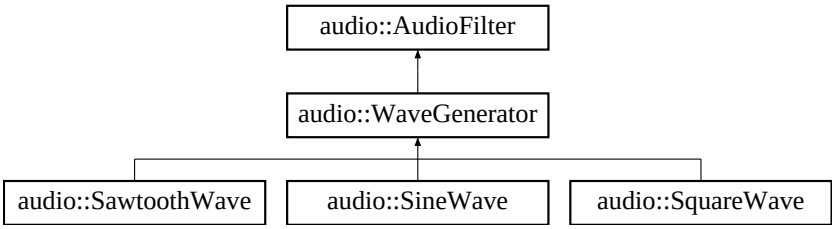
- src/utils/timer.h
- src/utils/timer.cpp

8.38 audio::WaveGenerator Class Reference

Abstract wave generator class.

`#include "wavegenerator.h"`

Inheritance diagram for audio::WaveGenerator:



Public Member Functions

- virtual ~**WaveGenerator** ()
  Virtual destructor.
- virtual void operator() (AudioTrack &track)=0
  Abstract function for wave generator.

Protected Member Functions

- WaveGenerator (float freql, float freqr, float amp, float offset)
  Creates a wave given frequencies, amplitude and offset.

Protected Attributes

- float **left_freq**
  Left channel frequency.
- float **right_freq**
  Right channel frequency.
- float **amplitude**
  Wave amplitude.
- float **offset**
  Offset from 0.

8.38.1 Detailed Description

Abstract wave generator class.

Waves generated will be discrete and applied to an audio track

See also

AudioTrack

8.38.2 Constructor & Destructor Documentation

8.38.2.1 WaveGenerator() `audio::WaveGenerator::WaveGenerator (`

```
float freql,
float freqr,
float amp,
float offset )  [inline], [protected]
```

Creates a wave given frequencies, amplitude and offset.

Parameters

freql	Left channel frequency
freqr	Right channel frequency
amp	Amplitude
offset	Offset from 0

8.38.3 Member Function Documentation

8.38.3.1 operator()() `virtual void audio::WaveGenerator::operator() (`

```
AudioTrack & track )  [pure virtual]
```

Abstract function for wave generator.

Parameters

track	Audio track to generate over

Implements audio::AudioFilter.

Implemented in audio::SawtoothWave, audio::SineWave, and audio::SquareWave.

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

- src/libLMTKaudio/wavegenerator.h

9 File Documentation

9.1 config.h

```
1 // cmake config
2
3 #define LMTK_VERSION  "0.0.0"
4 #define LMTK_VERSION_MAJOR "0"
5 #define LMTK_VERSION_MINOR "0"
6 #define LMTK_VERSION_PATCH "0"
```

9.2 src/app/tools/imagefx.hpp File Reference

Copyright (c) 2022 marinarasub.

```
#include <string>
#include <vector>
#include <unordered_map>
#include <functional>
#include "../../libLMTKimage/pluginchain.h"
#include "../../libLMTKimage/filters.h"
```

Classes

- struct lmtkimage::fx_info

Functions

- BokehBlur * **lmtkimage::make_bokeh_blur** (std::vector< std::string > args)
- BoxBlur * **lmtkimage::make_box_blur** (std::vector< std::string > args)
- BoxBlur * **lmtkimage::make_fast_blur** (std::vector< std::string > args)
- ChromaKeyShader * **lmtkimage::make_chroma_keyer** (std::vector< std::string > args)
- ColorInvertShader * **lmtkimage::make_color_invert** (std::vector< std::string > args)
- GaussianBlur * **lmtkimage::make_gaussian_blur** (std::vector< std::string > args)
- GrayScaleShader * **lmtkimage::make_grayscale** (std::vector< std::string > args)
- HueSatLumAdjust * **lmtkimage::make_hsl_adjust** (std::vector< std::string > args)
- ImageSharpen * **lmtkimage::make_image_sharpen** (std::vector< std::string > args)
- std::string **lmtkimage::help_fx** ()
- ImageShader * **lmtkimage::get_effect** (std::string effect, std::vector< std::string > args)

9.2.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in the LMTKimage app component of LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Header file for string arg wrappers for LMTK image shaders library

imageshader.h

9.3 imagefx.hpp

Go to the documentation of this file.

```
1 /*****************************************************************************/
20 #ifndef _LMTKIMAGE_IMAGEFX_H_
21 #define _LMTKIMAGE_IMAGEFX_H_
22 #pragma once
23
24
25 #include <string>
26 #include <vector>
27 #include <unordered_map> // TODO ordered map by alphabetical??? or nah
28 #include <functional>
29 #include "../../libLMTKimage/pluginchain.h"
30 #include "../../libLMTKimage/filters.h"
31
32
33 namespace lmtkimage
34 {
35     using namespace image;
36
37     // TODO make except. use map usage field
38     // TODO count passes since removed from image shaders TODO!!!!!!
39
40     BokehBlur* make_bokeh_blur(std::vector<std::string> args)
41     {
42         if (args.size() == 0) throw std::invalid_argument("bokeh blur needs 1 or 2 args (radius)");
43         float radx = std::stof(args[0]); // extension from center, EX 1 is 3x3, 2 is 5x5
44         if (args.size() == 1) return new BokehBlur(radx);
45         else if (args.size() == 2)
46         {
47             float rady = std::stof(args[1]);
48             return new BokehBlur(radx, rady);
49         }
50         if (args.size() > 2) throw std::invalid_argument("too many args for bokeh blur");
51     }
52
53     BoxBlur* make_box_blur(std::vector<std::string> args)
54     {
55         if (args.size() == 0) return new BoxBlur();
56         size_t size = std::stol(args[0]); // extension from center, EX 1 is 3x3, 2 is 5x5
57         if (args.size() == 1) return new BoxBlur(size);
58         if (args.size() > 1) throw std::invalid_argument("too many args for box blur");
59     }
60
61     BoxBlur* make_fast_blur(std::vector<std::string> args)
62     {
63         if (args.size() > 0) throw std::invalid_argument("too many args for fast blur, did you mean box
       blur?");
64         return new BoxBlur();
65     }
66
67     ChromaKeyShader* make_chroma_keyer(std::vector<std::string> args)
68     {
69         if (args.size() < 3) throw std::invalid_argument("chroma key needs 3, 4 or 5 args (r, g, b, <a>,
       <threshold>)");
70         float r = std::stof(args[0]);
71         float g = std::stof(args[1]);
72         float b = std::stof(args[2]);
73         RGBAPixel p(r, g, b);
74         float a, tol = ChromaKeyShader::DEFAULT_TOLERANCE;
75         if (args.size() > 3)
76         {
77             a = std::stof(args[3]);
78             p.setAlpha(a);
79         }
80         if (args.size() > 4)
81         {
82             tol = std::stof(args[4]);
83         }
84         if (args.size() > 5) throw std::invalid_argument("too many args for chroma key");
85         return new ChromaKeyShader(p, tol);
86     }
87
88     ColorInvertShader* make_color_invert(std::vector<std::string> args)
89     {
90         if (args.size() > 0) throw std::invalid_argument("too many args for invert");
91         return new ColorInvertShader();
92     }
93
94     GaussianBlur* make_gaussian_blur(std::vector<std::string> args)
95     {
96         if (args.size() < 1) throw std::invalid_argument("needs stdev");
97         if (args.size() > 2) throw std::invalid_argument("too many args for gaussian blur");
98         float stdevx = std::stof(args[0]);
99         if (args.size() == 1) return new GaussianBlur(stdevx, stdevx);
```

```
100         else
101         {
102             float stdevy = std::stof(args[1]);
103             return new GaussianBlur(stdevx, stdevy);
104         }
105     }
106
107     GrayScaleShader* make_grayscale(std::vector<std::string> args)
108     {
109         if (args.size() > 0) throw std::invalid_argument("too many args for grayscale");
110         return new GrayScaleShader();
111     }
112
113     HueSatLumAdjust* make_hsl_adjust(std::vector<std::string> args)
114     {
115         if (args.size() != 3) throw std::invalid_argument("hsl needs 2 args (hue shift degrees, sat
    multiplier, lum multiplier)");
116         float hue = std::stof(args[0]);
117         float sat = std::stof(args[1]);
118         float lum = std::stof(args[2]);
119         return new HueSatLumAdjust(hue, sat, lum);
120     }
121
122
123     ImageSharpen* make_image_sharpen(std::vector<std::string> args)
124     {
125         if (args.size() > 0) throw std::invalid_argument("too many args for sharpen");
126         std::cout « "Sharpen" « '\n';
127         return new ImageSharpen();
128     }
129
130     struct fx_info
131     {
132         std::function<image::ImageShader* (std::vector<std::string>)> function;
133         std::string description;
134         std::string usage;
135     };
136
137     // map for fx string to instantiation wrapper
138     static std::unordered_map<std::string, fx_info> fx_map =
139     {
140         {"bokeh-blur", {make_bokeh_blur, "bokeh/lens blur", "(radius-x, radius-y)"}},
141         {"box-blur", {make_box_blur, "box convolution blur, default is equivalent to fast blur",
    "(radius*)"}},
142         {"fast-blur", {make_fast_blur, "fast blur, approximates gaussian blur", "none"}},
143         {"chroma-key", {make_chroma_keyer, "chroma keyer, removes all color within threshold of target",
    "(r, g, b, a*, threshold*)"}},
144         {"color-invert", {make_color_invert, "inverts color", "none"}},
145         {"gaussian-blur", {make_gaussian_blur, "blurs the image with surrounding pixels with to gaussian
    function", "(stdev-x, stdev-y, TODO*)"}},
146         {"sharpen", {make_image_sharpen, "sharpens the image by emphasizing edges", "none"}},
147         {"hsl-adjust", {make_hsl_adjust, "adjust the hue, saturation and luminance of the image", "(hue
    shift in degrees, saturation multiplier, luminance multiplier)"}},
148         {"grayscale", {make_grayscale, "convert the image to grayscale (black & white)", "none"}}
149     };
150
151
152     // help msg for usage of fx
153     std::string help_fx()
154     {
155         std::string all_fx;
156         all_fx.append("Use effects with --fx \"effect_name(arg1, arg2,...)\" ...\n");
157         all_fx.append("Valid effects (* is optional argument):");
158
159         // find size to resize to
160         size_t max_name = 0, max_desc = 0, max_use = 0;
161         for (auto kv : fx_map)
162         {
163             size_t name_len = kv.first.length();
164             size_t desc_len = kv.second.description.length();
165             size_t use_len = kv.second.usage.length();
166             max_name = name_len > max_name ? name_len : max_name;
167             max_desc = desc_len > max_name ? desc_len : max_desc;
168             max_use = use_len > max_use ? use_len : max_use;
169         }
170         // format
171         for (auto kv : fx_map)
172         {
173             std::string name = kv.first;
174             std::string desc = kv.second.description;
175             std::string use = kv.second.usage;
176             name.resize(max_name, ' ');
177             desc.resize(max_desc, ' ');
178             use.resize(max_use, ' ');
179             all_fx.append("\n  " + name);
180             all_fx.append("    " + desc);
181             all_fx.append("    args: " + use);
```

```
182         }
183         return all_fx;
184     }
185 
186     // returns null if not found
187     ImageShader* get_effect(std::string effect, std::vector<std::string> args)
188     {
189         auto iter = fx_map.find(effect);
190         if (iter != fx_map.end()) return fx_map[effect].function(args);
191         throw std::invalid_argument("unknown effect: " + effect);
192     }
193 
194 
195 }
196 
197 
198 #endif // _LMTKIMAGE_IMAGEFX_H_
```

9.4 multithreadedrenderer.h

```
1 #ifndef __MULTITHREADEDRENDERER_H_
2 #define __MULTITHREADEDRENDERER_H_
3 
4 #pragma once
5 
6 // multithreaded image renderer
7 
8 
9 #endif // __MULTITHREADEDRENDERER_H_
```

9.5 progressbar.h

```
1 #ifndef __PROGRESSBAR_H_
2 #define __PROGRESSBAR_H_
3 
4 #include <stdlib.h>
5 #include <iostream>
6 #include <math.h>
7 #include <string>
8 #include <stdexcept>
9 
10 #if defined(_WIN32) || defined(_WIN64)
11 #define _WINDOWS
12 #include <windows.h>
13     #ifndef ENABLE_VIRTUAL_TERMINAL_PROCESSING
14     #define ENABLE_VIRTUAL_TERMINAL_PROCESSING 0x0004
15     #endif
16 
17 #define _WINDOWS_ANSI()                                                 \
18 do {                                                                    \
19     HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);                      \
20     DWORD dw;                                                           \
21     GetConsoleMode(hout, &dw);                                          \
22     if (!SetConsoleMode(                                                \
23         hout,                                                           \
24         dw | ENABLE_VIRTUAL_TERMINAL_PROCESSING)) exit(-1);             \
25 } while(0)
26 #endif
27 
28 namespace progressbar {
29 
30     const std::string CLEAR = "\x1B[0J";
31     const std::string CLEARSAVED = "\x1B[3J";
32     const std::string CLEARLINE = "\x1B[2K";
33     const std::string YELLOW = "\x1B[33m";
34     const std::string GREEN = "\x1B[32m";
35     const std::string RED  = "\x1B[31m";
36     const std::string RESET = "\x1B[0m";
37     const std::string BLINK = "\x1B[5m";
38     const std::string CURSORRESET = "\x1B[0G";
39 
40     std::string getProgressBar(double percent, bool failed);
41 
42 }
43 
44 #endif // __PROGRESSBAR_H_
```

9.6 src/libLMTKaudio/audiofilter.h File Reference

Copyright (c) 2022 marinarasub.

```
#include "audiotrack.h"
```

Classes

- class audio::AudioFilter
  Abstract audio filter class.

9.6.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKaudio, the audio library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Contains abstract audio filter class.

9.7 audiofilter.h

Go to the documentation of this file.

```
1 /*************************************************************************/
18 #ifndef _LIBLMTKAUDIO_AUDIOFILTER_H_
19 #define _LIBLMTKAUDIO_AUDIOFILTER_H_
20 #pragma once
21
22 #include "audiotrack.h"
23
24
25 namespace audio {
26
31     class AudioFilter
32     {
33     public:
34
38         virtual ~AudioFilter() {};
39
45         virtual void operator()(AudioTrack& track) = 0;
46     };
47 }
48
49 #endif // _LIBLMTKAUDIO_AUDIOFILTER_H_
```

9.8 src/libLMTKaudio/audioplayer.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <portaudio.h>
#include "audiotrack.h"
```

Classes

- class audio::AudioPlayer

  An audio player for audio tracks to device output Uses portaudio for audio device IO.

9.8.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKaudio, the audio library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Contains class for playing an audio track

9.9 audioplayer.h

Go to the documentation of this file.

```
1 /*************************************************************************/
18 #ifndef _LIBLMTKAUDIO_AUDIOPLAYER_H_
19 #define _LIBLMTKAUDIO_AUDIOPLAYER_H_
20 #pragma once
21
22 #include <portaudio.h>
23 #include "audiotrack.h"
24
25
26 namespace audio {
27
33     class AudioPlayer
34     {
35     public:
36
42         AudioPlayer();
43
47         ~AudioPlayer();
48
55         virtual void operator()(AudioTrack& track);
56
62         void play(AudioTrack& track);
63
64         //void record(AudioTrack& track);
65
```

```
69          void stop();
70
74          void close();
75
79          AudioTrack* track();
80
86          void setGain(float db); // in db (10log_10(out/in))
87
88      private:
89
93          static int playCallback(
94              const void* inputBuffer,
95              void* outputBuffer,
96              unsigned long framesPerBuffer,
97              const PaStreamCallbackTimeInfo* timeInfo,
98              PaStreamCallbackFlags statusFlags,
99              void* userData);
100
106         bool handle_pa_error(PaError err);
107
108         PaStream* stream;
110         AudioTrack* audio_track;
112         float amplifier;
114     };
115
116 }
117
118 #endif // _LIBLMTKAUDIO_AUDIOPLAYER_H_
```

9.10 src/libLMTKaudio/audiotrack.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <stdlib.h>
#include <stdio.h>
#include <stdexcept>
#include <math.h>
```

Classes

- struct audio::audio_sample

  a 2 channel (stereo) audio sample of values [-1, 1]
- class audio::AudioTrack

  Class for an AudioTrack containing audio samples which can be played, edited or written to a writeable format.

Typedefs

- typedef unsigned char **audio::byte**

9.10.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKaudio, the audio library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Contains class for a AudioTrack of audio samples which can be played

9.11 audiotrack.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _LIBLMTKAUDIO_AUDIOTRACK_H_
19 #define _LIBLMTKAUDIO_AUDIOTRACK_H_
20 #pragma once
21
22 #include <stdlib.h>
23 #include <stdio.h>
24 #include <stdexcept>
25 #include <math.h>
26
27
28 namespace audio {
29
30     typedef unsigned char byte;
31
36     struct audio_sample
37     {
38         float left;
39         float right;
46         audio_sample operator+(const audio_sample& other) const
47         {
48             audio_sample s{};
49             s.left = this->left + other.left;
50             s.right = this->right + other.right;
51             s.clamp();
52             return s;
53         }
54
58         void clamp();
59     };
60
67     class AudioTrack
68     {
69         // const for number of channels
70         static constexpr byte MONO = 1;
71         // const for number of channels
72         static constexpr byte STEREO = 2;
73
75         static constexpr unsigned int DEFAULT_SAMPLE_RATE = 44100;
76         // 44.1, 48 and multiples/factors of these.
77
78     public:
79
80         // TODO create with time in seconds?
81
85         AudioTrack();
86
92         AudioTrack(unsigned long length);
93
100         AudioTrack(unsigned long length, unsigned long sample_rate); // in samples
101
108         AudioTrack(const AudioTrack& other);
109
113         ~AudioTrack();
114
121         AudioTrack operator=(const AudioTrack& other);
122
130         AudioTrack operator+(const AudioTrack& other) const;
131
132
136         unsigned int sampleRate() const;
137
141         unsigned long sampleLength() const;
142
146         byte channels() const;
147
151         void resetPosition();
152
156         unsigned long currentPos() const;
157
161         bool ended() const;
162
168         audio_sample nextSample();
169
175         audio_sample getSample(unsigned long idx) const;
176
183         void setSample(unsigned long idx, audio_sample data);
184
185         // TODO resample method
192         void resample(unsigned long new_sample_rate);
193
200         void merge(AudioTrack& track);
201
```

```
202     protected:
203
209         void copy(const AudioTrack& other);
210
214         void clear();
215
216         //audio_sample clampSample(audio_sample& sample) const;
217
218
219         audio_sample* audio;
221     private:
222
228         void allocSamples(size_t length);
229
230         unsigned int sample_rate;
232         unsigned long current_pos;
234         unsigned long length;
236         // TODO only support stereo currently
237         byte num_tracks;
239     };
240 }
241
242 #endif // _LIBLMTKAUDIO_AUDIOTRACK_H_
```

9.12 src/libLMTKaudio/flac.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <stdlib.h>
#include <string>
#include <FLAC++/all.h>
#include "audiotrack.h"
#include "../utils/utils.h"
```

Classes

- class audio::FLAC

  Class for a FLAC file audio track.

9.12.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKaudio, the audio library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Contains class for read/writing and using a FLAC file as an audio track

9.13 flac.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
19 #ifndef _LIBLMTKAUDIO_FLAC_H_
20 #define _LIBLMTKAUDIO_FLAC_H_
21 #pragma once
22
23 #include <stdlib.h>
24 #include <string>
25 #include <FLAC++/all.h>
26 #include "audiotrack.h"
27 #include "../utils/utils.h"
28
29
30 namespace FLACAPI = FLAC; // to avoid confusion between libFLAC API
31
32 namespace audio {
33
39     class FLAC : public AudioTrack
40     {
41
46         class FLACDecoder : public FLACAPI::Decoder::File
47         {
48         public:
49             FLACDecoder() : FLACAPI::Decoder::File()
50             {
51                 out = NULL;
52             }
53
54             ~FLACDecoder()
55             {
56                 if (out != NULL) delete out;
57             }
58
59             // return constructed flac. call after read done.
60             FLAC get();
61
62         private:
63
64             virtual FLAC__StreamDecoderWriteStatus write_callback(
65                 const FLAC__Frame* frame,
66                 const FLAC__int32* const* buffer) override;
67
68             virtual void metadata_callback(
69                 const FLAC__StreamMetadata* metadata) override;
70
71             virtual void error_callback(
72                 FLAC__StreamDecoderErrorStatus status) override;
73
74             FLAC* out;
75
76             bool finished = false;
77
78             unsigned int bit_depth = 0; // of samples
79         };
80
81     public:
82
88         FLAC(unsigned long length);
89
96         FLAC(unsigned long length, unsigned long sample_rate);
97
104         FLAC(std::string path);
105
111         bool readFile(std::string path);
112
113         // TODO !!! NOT IMPLEMENTED
114         bool writeFile(std::string path);
115
116     };
117 }
118
119 #endif //_LIBLMTKAUDIO_FLAC_H_
```

9.14 src/libLMTKaudio/lowpassfilter.h File Reference

Copyright (c) 2022 marinarasub.

```
#include "audiofilter.h"
#include <Eigen/Dense>
```

Classes

- class audio::LowPassFilter

9.14.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKaudio, the audio library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Contians handy constants and conversions from musical notes (letters) to frequencies.

9.15 lowpassfilter.h

Go to the documentation of this file.

```
1 /*************************************************************************/
19 // !!! NOT YET IMPLEMENTED !!! //
20 #ifndef _AUDIO_LOWPASSFILTER_H_
21 #define _AUDIO_LOWPASSFILTER_H_
22 #pragma once
23
24 #include "audiofilter.h"
25 #include <Eigen/Dense>
26
27
28 namespace audio {
29
30     class LowPassFilter : public AudioFilter
31     {
32     public:
33         LowPassFilter(float cutoff) : cutoff_freq(cutoff) {};
34
35         void operator()(AudioTrack& track);
36
37     private:
38         float cutoff_freq;
39     };
40
41 }
42
43 #endif // _AUDIO_LOWPASSFILTER_H_
44
```

9.16 src/libLMTKaudio/musicnote.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <math.h>
```

Typedefs

- typedef int **audio::MusicalNote::Note**

Functions

- double audio::MusicalNote::getFrequency (Note note)
- double audio::MusicalNote::getFrequency (double A4Freq, Note note)

Variables

- constexpr double **audio::MusicalNote::DEFAULT_A4_HZ** = 440.0
- constexpr Note **audio::MusicalNote::C_4** = -9
- constexpr Note **audio::MusicalNote::C_MIDDLE** = -9
- constexpr Note **audio::MusicalNote::C_SHARP_4** = -8
- constexpr Note **audio::MusicalNote::D_FLAT_4** = -8
- constexpr Note **audio::MusicalNote::D_4** = -7
- constexpr Note **audio::MusicalNote::D_SHARP_4** = -6
- constexpr Note **audio::MusicalNote::E_FLAT_4** = -6
- constexpr Note **audio::MusicalNote::E_4** = -5
- constexpr Note **audio::MusicalNote::F_4** = -4
- constexpr Note **audio::MusicalNote::F_SHARP_4** = -3
- constexpr Note **audio::MusicalNote::G_FLAT_4** = -3
- constexpr Note **audio::MusicalNote::G_4** = -2
- constexpr Note **audio::MusicalNote::G_SHARP_4** = -1
- constexpr Note **audio::MusicalNote::A_FLAT_4** = -1
- constexpr Note **audio::MusicalNote::A_4** = 0
- constexpr Note **audio::MusicalNote::A_SHARP_4** = 1
- constexpr Note **audio::MusicalNote::B_FLAT_4** = 1
- constexpr Note **audio::MusicalNote::B_4** = 2
- constexpr Note **audio::MusicalNote::C_5** = 3
- constexpr Note **audio::MusicalNote::C_SHARP_5** = 4
- constexpr Note **audio::MusicalNote::D_FLAT_5** = 4
- constexpr Note **audio::MusicalNote::D_5** = 5
- constexpr Note **audio::MusicalNote::D_SHARP_5** = 6
- constexpr Note **audio::MusicalNote::E_FLAT_5** = 6
- constexpr Note **audio::MusicalNote::E_5** = 7
- constexpr Note **audio::MusicalNote::F_5** = 8
- constexpr Note **audio::MusicalNote::F_SHARP_5** = 9
- constexpr Note **audio::MusicalNote::G_FLAT_5** = 9
- constexpr Note **audio::MusicalNote::G_5** = 10
- constexpr Note **audio::MusicalNote::G_SHARP_5** = 11
- constexpr Note **audio::MusicalNote::A_FLAT_5** = 11
- constexpr Note **audio::MusicalNote::A_5** = 12
- constexpr Note **audio::MusicalNote::A_SHARP_5** = 13
- constexpr Note **audio::MusicalNote::B_FLAT_5** = 13
- constexpr Note **audio::MusicalNote::B_5** = 14
- constexpr Note **audio::MusicalNote::C_6** = 15

9.16.1 Detailed Description

LMTK / Light Multimedia ToolKit

Author

marinarasub

Date

January 2022

Contains handy constants and conversions from musical notes (letters) to frequencies.

9.16.2 Function Documentation

9.16.2.1 getFrequency() [1/2]

```
double audio::MusicalNote::getFrequency (
        double A4Freq,
        Note note )
```

Parameters

A4Freq	The A4 frequency to tune to
note	The musical note

Returns

The frequency of a note n semitones from custom A4 frequency

9.16.2.2 getFrequency() [2/2]

```
double audio::MusicalNote::getFrequency (
        Note note )
```

Parameters

note	The musical note

Returns

The frequency of a note n semitones from A440

9.17 musicnote.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
19 #ifndef _LIBLMTKAUDIO_MUSICNOTE_H_
20 #define _LIBLMTKAUDIO_MUSICNOTE_H_
21 #pragma once
22
23 #include <math.h>
24
25
26 namespace audio {
27
28     namespace MusicalNote
29     {
30         // TODO use enum instead?
31
32         typedef int Note; // in delta semitones from A4 below:
33         constexpr double DEFAULT_A4_HZ = 440.0;
34
35         /* ALL NOTES from 1-8 octave */ // TODO finish
36         constexpr Note C_4 = -9, C_MIDDLE = -9;
37         constexpr Note C_SHARP_4 = -8, D_FLAT_4 = -8;
38         constexpr Note D_4 = -7;
39         constexpr Note D_SHARP_4 = -6, E_FLAT_4 = -6;
40         constexpr Note E_4 = -5;
41         constexpr Note F_4 = -4;
42         constexpr Note F_SHARP_4 = -3, G_FLAT_4 = -3;
43         constexpr Note G_4 = -2;
44         constexpr Note G_SHARP_4 = -1, A_FLAT_4 = -1;
45         constexpr Note A_4 = 0;
46         constexpr Note A_SHARP_4 = 1, B_FLAT_4 = 1;
47         constexpr Note B_4 = 2;
48
49         constexpr Note C_5 = 3;
50         constexpr Note C_SHARP_5 = 4, D_FLAT_5 = 4;
51         constexpr Note D_5 = 5;
52         constexpr Note D_SHARP_5 = 6, E_FLAT_5 = 6;
53         constexpr Note E_5 = 7;
54         constexpr Note F_5 = 8;
55         constexpr Note F_SHARP_5 = 9, G_FLAT_5 = 9;
56         constexpr Note G_5 = 10;
57         constexpr Note G_SHARP_5 = 11, A_FLAT_5 = 11;
58         constexpr Note A_5 = 12;
59         constexpr Note A_SHARP_5 = 13, B_FLAT_5 = 13;
60         constexpr Note B_5 = 14;
61
62         constexpr Note C_6 = 15;
63
68         double getFrequency(Note note) {
69             return DEFAULT_A4_HZ * pow(2, 1.0 / 12.0 * note);
70         }
71
77         double getFrequency(double A4Freq, Note note) {
78             return A4Freq * pow(2, 1.0 / 12.0 * note);
79         }
80     }
81 }
82
83 #endif // _LIBLMTKAUDIO_MUSICNOTE_H_
```

9.18 src/libLMTKaudio/sawtooth.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <math.h>
#include "wavegenerator.h"
```

Classes

- class audio::SawtoothWave

  Class for generating a sawtooth wave.

9.18.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKaudio, the audio library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

File containing sawtooth wave generator

9.19 sawtooth.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _LIBLMTKAUDIO_SAWTOOTH_H_
19 #define _LIBLMTKAUDIO_SAWTOOTH_H_
20 #pragma once
21
22 #include <math.h>
23 #include "wavegenerator.h"
24
25
26 namespace audio {
27
35     class SawtoothWave : public WaveGenerator
36     {
37     public:
38
46         SawtoothWave(float freql, float freqr, float amp);
47
56         SawtoothWave(float freql, float freqr, float amp, float offset);
57
63         void operator()(AudioTrack& track);
64
65     };
66 }
67
68 #endif // _LIBLMTKAUDIO_SAWTOOTH_H_
69
```

9.20 src/libLMTKaudio/sinewave.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <math.h>
#include "wavegenerator.h"
```

Classes

- class audio::SineWave

  Class for generating a sine wave.

9.20.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKaudio, the audio library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

File containing sine wave generator

9.21 sinewave.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _LIBLMTKAUDIO_SINEWAVE_H_
19 #define _LIBLMTKAUDIO_SINEWAVE_H_
20 #pragma once
21
22 #include <math.h>
23 #include "wavegenerator.h"
24
25
26 namespace audio {
27
33     class SineWave : public WaveGenerator
34     {
35     public:
36
44         SineWave(float freql, float freqr, float amp);
45
54         SineWave(float freql, float freqr, float amp, float offset);
55
61         void operator()(AudioTrack& track);
62
63     };
64 }
65
66 #endif // _LIBLMTKAUDIO_SINEWAVE_H_
```

9.22 src/libLMTKaudio/squarewave.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <math.h>
#include "wavegenerator.h"
```

Classes

- class audio::SquareWave

  Class for generating a square wave.

9.22.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKaudio, the audio library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

File containing square wave generator

9.23 squarewave.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _LIBLMTKAUDIO_SQUAREWAVE_H_
19 #define _LIBLMTKAUDIO_SQUAREWAVE_H_
20 #pragma once
21
22 #include <math.h>
23 #include "wavegenerator.h"
24
25
26 namespace audio {
27
33     class SquareWave : public WaveGenerator
34     {
35     public:
36
44         SquareWave(float freql, float freqr, float amp);
45
54         SquareWave(float freql, float freqr, float amp, float offset);
55
61         void operator()(AudioTrack& track);
62
63
64     };
65 }
66
67 #endif // _LIBLMTKAUDIO_SQUAREWAVE_H_
68
```

9.24 src/libLMTKaudio/wavegenerator.h File Reference

Copyright (c) 2022 marinarasub.

```
#include "audiofilter.h"
#include "../utils/utilsmath.h"
```

Classes

- class audio::WaveGenerator

  Abstract wave generator class.

Variables

- constexpr double **audio::PI** = utils::math::PI

9.24.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKaudio, the audio library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Contains abstract class for wave singal generation

9.25 wavegenerator.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _LIBLMTKAUDIO_WAVEGENERATOR_H_
19 #define _LIBLMTKAUDIO_WAVEGENERATOR_H_
20 #pragma once
21
22 #include "audiofilter.h"
23 #include "../utils/utilsmath.h"
24
25
26 namespace audio {
27
28     constexpr double PI = utils::math::PI;
29
37     class WaveGenerator : public AudioFilter
38     {
39     public:
40
44         virtual ~WaveGenerator() {};
45
51         virtual void operator()(AudioTrack& track) = 0;
52
53     protected:
54
63         WaveGenerator(float freql, float freqr, float amp, float offset)
64             : left_freq(freql), right_freq(freqr), amplitude(amp), offset(offset)
65         {}
66
67         float left_freq;
68         float right_freq;
69         // Note: will be clamped to [-1, 1]
70         float amplitude;
71         float offset;
72     };
73 }
74
75 #endif // _LIBLMTKAUDIO_WAVEGENERATOR_H_
```

9.26 src/libLMTKimage/bokehblur.h File Reference

Copyright (c) 2022 marinarasub.

```
#include "convolution.h"
#include "../utils/utilsmath.h"
```

Classes

- class image::BokehBlur

  Class for a bokeh/lens blur using a disk/circular convolution.

9.26.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

This file contains a functor for bokeh blur

9.27 bokehblur.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _LIBLMTKIMAGE_BOKEHBLUR_H_
19 #define _LIBLMTKIMAGE_BOKEHBLUR_H_
20 #pragma once
21
22 #include "convolution.h"
23 #include "../utils/utilsmath.h"
24
25
26 namespace image {
27
34     class BokehBlur : public ImageConvolution
35     {
36     public:
37
38         // Note: may want to correct lighting / highlights for artistic effect
39
45         BokehBlur(float radius);
46
53         BokehBlur(float radiusX, float radiusY);
54
55     private:
56
63         virtual void computeKernel() override;
64
65         float radiusX;
66         float radiusY;
67     };
68
69 }
70
71 #endif // _LIBLMTKIMAGE_BOKEHBLUR_H_
```

9.28 src/libLMTKimage/boxblur.h File Reference


```
#include "convolution.h"
```

Classes

- class image::BoxBlur

  Class for box blur.
- class image::BoxBlurHorizontal

  Class for box blur's seperable horizontal component.
- class image::BoxBlurVertical

  Class for box blur's seperable vertical component.

9.28.1 Detailed Description

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

File containing classes for box blur

Contains both full version and seperable versions (horizontal/vertical)

9.29 boxblur.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
21 #ifndef _LIBLMTKIMAGE_BOXBLUR_H_
22 #define _LIBLMTKIMAGE_BOXBLUR_H_
23 #pragma once
24
25 #include "convolution.h"
26
27 namespace image {
28
30     static const size_t DEFAULT_BOXBLUR_RADIUS = 1; // 3x3
31
41     class BoxBlur : public ImageConvolution
42     {
43     public:
44
48         BoxBlur();
49
58         BoxBlur(size_t radius);
59
69         BoxBlur(size_t radiusX, size_t radiusY);
70
71     private:
72
76         virtual void computeKernel() override;
77 ;
78     };
79
80
91     class BoxBlurHorizontal : public ImageConvolution
92     {
93     public:
94
98         BoxBlurHorizontal();
99
105         BoxBlurHorizontal(size_t radius);
106
107     private:
108
112         virtual void computeKernel() override;
113
114     };
115
116
127     class BoxBlurVertical : public ImageConvolution
128     {
129     public:
130
134         BoxBlurVertical();
135
141         BoxBlurVertical(size_t radius);
142
143     private:
144
148         virtual void computeKernel() override;
149
150     };
151
152 }
153
154 #endif // _LIBLMTKIMAGE_BOXBLUR_H_
```

9.30 src/libLMTKimage/chromakeyer.h File Reference

Copyright (c) 2022 marinarasub.

```
#include "imageshader.h"
```

Classes

- class image::ChromaKeyShader

  Class for a simple chroma keyer (aka green screen).

9.30.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

December 2021

Header file for chroma keyer functor

9.31 chromakeyer.h

Go to the documentation of this file.

```
1 /*************************************************************************/
19 #ifndef _LIBLMTKIMAGE_CHROMAKEYER_H_
20 #define _LIBLMTKIMAGE_CHROMAKEYER_H_
21 #pragma once
22
23 #include "imageshader.h"
24
25
26 namespace image {
27
36     class ChromaKeyShader : public ImageShader
37     {
38     public:
39
43         static constexpr float DEFAULT_TOLERANCE = 0.1;
44
51         ChromaKeyShader(RGBAPixel target);
59         ChromaKeyShader(RGBAPixel target, float tol);
60
70         virtual void operator()(Image& img) override;
71
82         virtual RGBAPixel operator()(const Image& img, size_t x, size_t y);
83
84     private:
85         RGBAPixel target;
86         float threshold = 0;
87     };
88
89 }
90
91 #endif // _LIBLMTKIMAGE_CHROMAKEYER_H_
```

9.32 src/libLMTKimage/color.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <math.h>
```

Classes

- struct color::rgb_color
- struct color::hsl_color

Namespaces

- namespace color

  Contains color representations, functions and conversion between them.

Functions

- hsl_color **color::rgb_to_hsl** (double r, double g, double b)

  Translated from https://www.rapidtables.com/convert/color/hsl-to-rgb.html.
- hsl_color **color::rgb_to_hsl** (const rgb_color &clr)
- rgb_color **color::hsl_to_rgb** (double h, double s, double l)
- rgb_color **color::hsl_to_rgb** (const hsl_color &clr)

9.32.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

This file contains functions for representing & converting color types

9.33 color.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
19 #ifndef _LIBLMTKIMAGE_COLOR_H_
20 #define _LIBLMTKIMAGE_COLOR_H_
21 #pragma once
22
23 #include <math.h>
24
29 namespace color {
30
31     // RGB color model
32     // see https://en.wikipedia.org/wiki/RGB_color_model
33     typedef struct {
34         double r, g, b; // [0, 1]
35     } rgb_color;
36
37     // HSL color model
38     // see https://en.wikipedia.org/wiki/HSL_and_HSV
39     typedef struct {
40         double h;  // degrees, [0, 360)
41         double s;  // [0, 1]
42         double l;  // [0, 1]
43     } hsl_color;
44
45     // convert explicit rgb to hsl
46     hsl_color rgb_to_hsl(double r, double g, double b);
47
48     // convert rgb struct to rsl
49     hsl_color rgb_to_hsl(const rgb_color& clr);
50
51     // convert explicit hsl to rgb
52     rgb_color hsl_to_rgb(double h, double s, double l);
53
54     // convert hsl struct to rgb
55     rgb_color hsl_to_rgb(const hsl_color& clr);
56
57 }
58
59 #endif //_LIBLMTKIMAGE_COLOR_H_
```

9.34 src/libLMTKimage/colorinvert.h File Reference

Copyright (c) 2022 marinarasub.

```
#include "imageshader.h"
```

Classes

- class image::ColorInvertShader

  Class for a color inversion.

9.34.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

December 2021

Header file for color invert functor

9.35 colorinvert.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
19 #ifndef _LIBLMTKIMAGE_COLORINVERT_H_
20 #define _LIBLMTKIMAGE_COLORINVERT_H_
21 #pragma once
22
23 #include "imageshader.h"
24
25
26 namespace image {
27
36     class ColorInvertShader : public ImageShader
37     {
38     public:
39
47         virtual void operator()(Image& img) override;
48
58         virtual RGBAPixel operator()(const Image& img, size_t x, size_t y) override;
59
60     private:
67         RGBAPixel invert(RGBAPixel p);
68     };
69
70 }
71
72 #endif // _LIBLMTKIMAGE_COLORINVERT_H_
73
```

9.36 src/libLMTKimage/convolution.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <Eigen/Dense>
#include "imageshader.h"
```

Classes

- class image::ImageConvolution

  Base class for all image filters using convolution/kernel filtering.

9.36.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

December 2021

File containing the base class for convolution filtering

See also

imageshader.h

9.37 convolution.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
22 #ifndef _LIBLMTKIMAGE_CONVOLUTION_H_
23 #define _LIBLMTKIMAGE_CONVOLUTION_H_
24 #pragma once
25
26 #include <Eigen/Dense>
27 #include "imageshader.h"
28
29 namespace image {
30
31     // TODO seperate kernels for each channel
32
33     /*
34     * Convolution Example:
35     *
36     * Ex Identity   0 0 0   1 2 3
37     *               0 1 0 * 4 5 6 = 5
38     *               0 0 0   7 8 9
39     *
40     * Ex Box        1 1 1   1 2 3
41     *               1 1 1 * 4 5 6 = 45
42     *               1 1 1   7 8 9
43     */
44
57     class ImageConvolution : public ImageShader
58     {
59     public:
60
66         ImageConvolution();
67
74         ImageConvolution(size_t center);
75
83         ImageConvolution(size_t centerX, size_t centerY);
84
94         ImageConvolution(size_t centerX, size_t centerY,
95             EdgeHandlingMethod edge_method);
96
103         virtual void operator()(Image& img) override;
104
117         virtual RGBAPixel operator()(const Image& img, size_t x, size_t y)
118             override;
119
120     protected:
121
122         /*
123         * Sizing Example:
124         *
125         * Ex center = 0, 0  0
126         *
127         * Ex center = 1, 1  0 0 0
128         *                   0 c 0
129         *                   0 0 0
130         *
131         * Ex center = 2, 3  0 0 0 0 0
132         *                   0 0 0 0 0
133         *                   0 0 0 0 0
134         *                   0 0 c 0 0
135         *                   0 0 0 0 0
136         *                   0 0 0 0 0
137         *                   0 0 0 0 0
138         */
139
148         void setKernelSize(size_t centerX, size_t centerY);
149
154         virtual void computeKernel() = 0;
155
168         virtual RGBAPixel convolution(const Image& img, unsigned int x, unsigned int y);
169
176         void normalize();
177
178         Eigen::MatrixXf kernel;
179     };
180
181 }
182
183 #endif //_LIBLMTKIMAGE_CONVOLUTION_H_
```

9.38 src/libLMTKimage/filters.h File Reference

Copyright (c) 2022 marinarasub.

```
#include "bokehblur.h"
#include "boxblur.h"
#include "chromakeyer.h"
#include "colorinvert.h"
#include "gaussianblur.h"
#include "grayscale.h"
#include "huesatlum.h"
#include "sharpen.h"
```

9.38.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTK, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Header file for including all basic image filter headers

9.39 filters.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _LIBLMTKIMAGE_FILTERS_H_
19 #define _LIBLMTKIMAGE_FILTERS_H_
20 #pragma once
21
22 // TODO move to public headers
23 #include "bokehblur.h"
24 #include "boxblur.h"
25 #include "chromakeyer.h"
26 #include "colorinvert.h"
27 #include "gaussianblur.h"
28 #include "grayscale.h"
29 #include "huesatlum.h"
30 #include "sharpen.h"
31
32 #endif // _LIBLMTKIMAGE_FILTERS_H_
```

9.40 src/libLMTKimage/gaussianblur.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <math.h>
#include "convolution.h"
#include "../utils/utilsmath.h"
```

Classes

- class image::GaussianBlur

  Class for gaussian blur kernel filtering of an image.
- class image::GaussianBlurHorizontal

  Class for seperable horizontal component of gaussian blur.
- class image::GaussianBlurVertical

  Class for seperable vertical component of gaussian blur.

Functions

- float image::sigmaTolerance (float zscore)

  Helper function for computing gaussian() from given z-score.

9.40.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

December 2021

Header file for gaussian blur shader

9.40.2 Function Documentation

9.40.2.1 sigmaTolerance() `static float image::sigmaTolerance ( float zscore )`

Helper function for computing gaussian() from given z-score.

Parameters

zscore	Z-score, or standard deviations from mean

Returns

gaussian function result

9.41 gaussianblur.h

Go to the documentation of this file.

```
1 /*************************************************************************/
19 #ifndef _LIBLMTKIMAGE_GAUSSIANBLUR_H_
20 #define _LIBLMTKIMAGE_GAUSSIANBLUR_H_
21 #pragma once
22
23 #include <math.h>
24 #include "convolution.h"
25 #include "../utils/utilsmath.h"
26
27
28 namespace image {
29
36     static float sigmaTolerance(float zscore);
37
47     class GaussianBlur : public ImageConvolution
48     {
49     public:
50
57         GaussianBlur(float stdev);
58
66         GaussianBlur(float stdevX, float stdevY);
67
81         GaussianBlur(float stdevX, float stdevY, float toleranceX, float toleranceY);
82
83     private:
84
90         virtual void computeKernel() override;
91
92         float sigmaX;
93         float sigmaY;
95     };
96
97
105     class GaussianBlurHorizontal : public ImageConvolution
106     {
107     public:
108
115         GaussianBlurHorizontal(float stdev);
116
126         GaussianBlurHorizontal(float stdev, float tol);
127
128     private:
129
135         virtual void computeKernel() override;
136
137         float sigma;
139     };
140
141
149     class GaussianBlurVertical : public ImageConvolution
150     {
151     public:
152
159         GaussianBlurVertical(float stdev);
160
170         GaussianBlurVertical(float stdev, float tol);
171
172     private:
173
179         virtual void computeKernel() override;
180
181         float sigma;
183     };
184
185
186 }
187
188 #endif // _LIBLMTKIMAGE_GAUSSIANBLUR_H_
189
```

9.42 src/libLMTKimage/grayscale.h File Reference

Copyright (c) 2022 marinarasub.

```
#include "imageshader.h"
```

Classes

- class image::GrayScaleShader

  Class for a converting any image to a grayscale version.

9.42.1 Detailed Description

LMTK / Light Multimedia ToolKit

Author

marinarasub

Date

December 2021

Header file for a converting an image to a grayscale image.

9.43 grayscale.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
19 #ifndef _LIBLMTKIMAGE_GRAYSCALE_H_
20 #define _LIBLMTKIMAGE_GRAYSCALE_H_
21 #pragma once
22
23 #include "imageshader.h"
24
25
26 namespace image {
27
37     class GrayScaleShader : public ImageShader
38     {
39     public:
40
46         virtual void operator()(Image& img) override;
47
56         virtual RGBAPixel operator()(const Image& img, size_t x, size_t y) override;
57
58
59     private:
60
68         RGBAPixel greyScale(RGBAPixel p);
69     };
70
71 }
72
73 #endif // _LIBLMTKIMAGE_GRAYSCALE_H_
```

9.44 src/libLMTKimage/hslapixel.h File Reference


```
#include <iostream>
#include <string>
#include "color.h"
#include "rgbapixel.h"
```

Classes

- class image::HSLAPixel

  Class for representing a pixel's color using the HSL color model.

Functions

- std::ostream & image::operator<< (std::ostream &out, const HSLAPixel &pixel)

  Add string representation of pixel to stream.

9.44.1 Detailed Description

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

This file contains the class for representing color using the HSLA color model.

9.44.2 Function Documentation

9.44.2.1 operator<<() `std::ostream & image::operator<< (`

```
    std::ostream & out,
    const HSLAPixel & pixel )
```

Add string representation of pixel to stream.

Parameters

out	Output stream
pixel	Pixel to output to stream

9.45 hslapixel.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
19 #ifndef _LIBLMTKIMAGE_HSLAPIXEL_H_
20 #define _LIBLMTKIMAGE_HSLAPIXEL_H_
21 #pragma once
22
23 #include <iostream>
24 #include <string>
25 #include "color.h"
26 #include "rgbapixel.h"
27
28
29 namespace image {
30
31     class RGBAPixel;
32
33
49     class HSLAPixel
50     {
51
52     public:
53
54         // TODO avg functions
55
61         HSLAPixel();
62
73         HSLAPixel(float h, float s, float l, float a);
74
80         HSLAPixel(const RGBAPixel& rgb);
81
87         void operator=(const HSLAPixel& other);
88
96         bool operator==(const HSLAPixel& other) const;
97
105         bool operator!=(const HSLAPixel& other) const;
106
116         void set(float h, float s, float l);
117
128         void set(float h, float s, float l, float a);
129
136         void set(const HSLAPixel& other);
137
144         void set(const RGBAPixel& other);
145
151         void setAlpha(float a);
152
159         void setTransparency(float transparency);
160
161         // !!! This function is subject to change.
167         std::string to_string() const; // todo stream operator instead??
168
171         float h;
174         float s;
177         float l;
180         float a;
182     private:
183
190         void clampHSLA();
191
192
193     };
194
201     std::ostream& operator<<(std::ostream& out, const HSLAPixel& pixel);
202 }
203
204 #endif // _LIBLMTKIMAGE_HSLAPIXEL_H_
```

9.46 src/libLMTKimage/hslcolor.h File Reference

Copyright (c) 2022 marinarasub.

```
#include "hslapixel.h"
```

Namespaces

- namespace image::hslcolor

  Contains common colors as constants for convinience.

9.46.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

This file contains useful constants and such for common HSL colors.

9.47 hslcolor.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
20 #ifndef _LIBLMTKIMAGE_HSLCOLOR_H_
21 #define _LIBLMTKIMAGE_HSLCOLOR_H_
22 #pragma once
23
24 #include "hslapixel.h"
25
26
27 namespace image {
28
34     namespace hslcolor {
35
36         static const HSLAPixel TRANSPARENT(0.0, 0.0, 0.0, 0.0);
38         // TODO add more colors
39     }
40
41 }
42
43 #endif // _LIBLMTKIMAGE_HSLCOLOR_H_
```

9.48 src/libLMTKimage/huesatlum.h File Reference

Copyright (c) 2022 marinarasub.

```
#include "imageshader.h"
```

Classes

- class image::HueSatLumAdjust

  Class for adjustin the hue, saturation, and luminance of an image.

9.48.1 Detailed Description

LMTK / Light Multimedia ToolKit

Author

marinarasub

Date

January 2022

Header file for a adjusting HSL of an image

9.49 huesatlum.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _LIBLMTKIMAGE_HUESATLUM_H_
19 #define _LIBLMTKIMAGE_HUESATLUM_H_
20 #pragma once
21
22 #include "imageshader.h"
23
24
25 namespace image {
26
32     class HueSatLumAdjust : public ImageShader
33     {
34     public:
35
44         HueSatLumAdjust(float hue, float sat, float lum);
45
51         virtual void operator()(Image& img) override;
52
61         virtual RGBAPixel operator()(const Image& img, size_t x, size_t y) override;
62
63     private:
64
65         float hue_shift;
66         float sat_multiplier;
67         float lum_multiplier;
68     };
69
70 }
71
72 #endif // _LIBLMTKIMAGE_HUESATLUM_H_
```

9.50 src/libLMTKimage/image.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <stdlib.h>
#include <stdexcept>
#include "rgbcolor.h"
#include "hslcolor.h"
```

Classes

- class image::Image

  Represents an rectangular, colored image.

Typedefs

- typedef unsigned char **image::byte**

Enumerations

- enum class **EdgeHandlingMethod** {
  NO_HANDLE , **EXTEND** , **WRAP** , **TILE** = WRAP ,
  MIRROR , **DEFAULT** = NO_HANDLE }
- enum class image::ResampleMethod { **NEIGHBOR** , **BILINEAR** , **BICUBIC** , **DEFAULT** = BILINEAR }

  Enum for resizing methods: nearest neighbor, linear, bicubic.

9.50.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

December 2021

Class for representing an colored image in memory

The image class represents a 2D canvas with color, and provides basic image creation and modification.

9.51 image.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
22 #ifndef _LIBLMTKIMAGE_IMAGE_H_
23 #define _LIBLMTKIMAGE_IMAGE_H_
24 #pragma once
25
26 #include <stdlib.h>
27 #include <stdexcept>
28 #include "rgbcolor.h"
29 #include "hslcolor.h"
30
31
32 namespace image {
33
34 //#define SUCCESS 0
35 //#define ERROR -1 // TODO don't use macros, use exceptions instead
36
37     typedef unsigned char byte;
38
39     /*
40     * Enum for picking pixels outside of image dimensions
41     *
42     * DEFAULT is throw exception
43     */
44     enum class EdgeHandlingMethod
45     {
46         NO_HANDLE,
47         EXTEND,
48         WRAP,
49         TILE = WRAP,
50         MIRROR,
51         DEFAULT = NO_HANDLE
52     };
53
54
58     enum class ResampleMethod {
59         NEIGHBOR,
60         BILINEAR,
61         BICUBIC,
62         DEFAULT = BILINEAR
63     };
64
65     // TODO image transformations
66
80     class Image
81     {
88         struct pixel_array_struct
89         {
90             RGBAPixel* pixels;
91             int width;
92             int height;
93         };
94
95
96     public:
103          Image();
104
114          Image(unsigned int w, unsigned int h);
115
121          Image(const Image& other);
122
129          virtual ~Image();
130
138          virtual Image operator=(const Image& other);
139
148          virtual bool operator==(const Image& other) const;
149
156          virtual bool operator!=(const Image& other) const;
157
163          unsigned int width() const;
164
170          unsigned int height() const;
171
178          unsigned int size() const;
179
186          void fillSolidColor(const RGBAPixel& color);
187
188          // TODO add angles and raidal modes. Right now it is only vertical grad.
189          // TODO arbitrary number of "color points", given location to start each.
200          void fillGradient(const RGBAPixel& color1, const RGBAPixel& color2);
201
202          // TODO specify location to place
214          void blendNormal(const Image& other);
215
```

```
216         // TODO bicubic, linear resample
225         void resize(unsigned int w, unsigned int h); // calls default
226         void resize(unsigned int w, unsigned int h, ResampleMethod method); // will call scale resize
227         // calls getPixel with a interp. method (accepts floats)
228         // TODO downsampling
229
239         void scale(float scale_x, float scale_y);
240
241         // TODO crop and size
242         // void crop(); upper left and lower right
243         //
244
245         // TODO flip horizontal
249         void mirror(); // flip across one axis
250
251         // TODO rotate (more complicated, need matrix math and resampling)
252         //
253
254         // TODO may return color instead of ptr in future
264         RGBAPixel* getRGBAPixel(int x, int y) const;
265
276         RGBAPixel getRGBAPixel(int x, int y, EdgeHandlingMethod edge_method) const;
277
288         RGBAPixel getRGBAPixel(float x, float y, ResampleMethod method) const;
289
290         // TODO get HSLA?
291
301         bool setRGBAPixel(const RGBAPixel& color, int x, int y);
302
312         bool setHSLAPixel(const HSLAPixel& color, int x, int y);
313
320         std::string to_string();
321
322
323     protected:
324
325         // lower level functions for read/write
329         RGBAPixel* getRow(int row_num) const;
330
331         // caller's responsibility to make sure row_width is correct
332         // if row_wdith > width(), ignore. < then stop at row_width
333         bool setRow(RGBAPixel row_pixels[], size_t row_width, int row_num);
334
342         RGBAPixel* pixelAt(int x, int y) const;
343
347         RGBAPixel bilinearResample(float x, float y) const;
348
349         // TODO
350         RGBAPixel bicubicResample(float x, float y) const;
351
352         // TODO mipmap, box (downsampling)
353
362         void setInitialSize(int w, int h);
363
372         virtual void copy(const Image&);
373
379         virtual void clear();
380
381         pixel_array_struct pixel_data;
382     };
383
384 }
385
386 #endif // _LIBLMTKIMAGE_IMAGE_H_
```

9.52 src/libLMTKimage/imagefile.h File Reference

Copyright (c) 2022 marinarasub.

Classes

- class image::ImageFile

  class for read/writeable image file.

9.52.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

File containing base class for write/readable image file

9.53 imagefile.h

Go to the documentation of this file.

```
1 /*/*************************************************************************/
19 #ifndef _LIBLMTKIMAGE_IMAGEFILE_H_
20 #define _LIBLMTKIMAGE_IMAGEFILE_H_
21 #pragma once
22
23 #include <string>
24 #include "image.h"
25
26
27 namespace image {
28
35     class ImageFile : public Image
36     {
37
38     public:
39
43         virtual ~ImageFile() {}
44
50         virtual void readFile(std::string path) = 0;
51
57         virtual void writeFile(std::string path) = 0;
58
59     protected:
60
67         ImageFile() : Image() {}
68
77         ImageFile(unsigned int w, unsigned int h) : Image(w, h) {}
78
84         ImageFile(const Image& other) : Image(other) {}
85
86         //ImageFile(const ImageFile &other) : Image(other) {}
87
88     };
89
90 }
91
92 #endif // _LIBLMTKIMAGE_IMAGEFILE_H_
```

9.54 src/libLMTKimage/imageshader.h File Reference

Copyright (c) 2022 marinarasub.

```
#include "image.h"
#include "rgbapixel.h"
#include "hslapixel.h"
```

Classes

- class image::ImageShader

  Abstract class for an image shader.

9.54.1 Detailed Description

LMTK / Light Multimedia ToolKit

Author

marinarasub

Date

December 2021

Header file containing base class for image and pixel shaders

9.55 imageshader.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
19 #ifndef _LIBLMTKIMAGE_IMAGESHADER_H_
20 #define _LIBLMTKIMAGE_IMAGESHADER_H_
21 #pragma once
22
23 #include "image.h"
24 #include "rgbapixel.h"
25 #include "hslapixel.h"
26
27
28 namespace image {
29
37     class ImageShader
38     {
39     public:
40
42         static const EdgeHandlingMethod DEFAULT_EDGE_METHOD
43             = EdgeHandlingMethod::EXTEND;
44
48         ImageShader() : ImageShader(DEFAULT_EDGE_METHOD) {}
49
53         ImageShader(EdgeHandlingMethod method) : edge_method(method) {}
54
58         virtual ~ImageShader() {};
59
66         virtual void operator()(Image& img);
67
74         virtual RGBAPixel operator()(const Image& img, size_t x, size_t y) = 0;
75
76
77     protected:
78
79         EdgeHandlingMethod edge_method;
81     };
82
83 }
84
85 #endif // _LIBLMTKIMAGE_IMAGESHADER_H_
```

9.56 src/libLMTKimage/jpeg.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <stdio.h>
#include <cstring>
#include <stdexcept>
#include <jpeglib.h>
#include "imagefile.h"
#include "../utils/utils.h"
```

Classes

- class image::JPEG

  JPEG image class.

9.56.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

December 2021

Contains the JPEG image class.

The JPG class uses the libjpeg interface for file I/O and compression, and is a high level representation of a JPEG image.

Note

file extensions can be .jpg or .jpeg

9.57 jpeg.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
23 #ifndef _LIBLMTKIMAGE_JPEG_H_
24 #define _LIBLMTKIMAGE_JPEG_H_
25 #pragma once
26
27 #include <stdio.h>
28 #include <cstring>
29 #include <stdexcept>
30 #include <jpeglib.h>
31 #include "imagefile.h"
32 #include "../utils/utils.h"
33
34
35 namespace image {
36
45     class JPEG : public ImageFile
46     {
47     public:
48
57         JPEG(unsigned int w, unsigned int h);
58
66         JPEG(std::string path);
67
71         JPEG(const Image& img);
72
76         ~JPEG();
77
86         void readFile(std::string path) override;
87
96         void writeFile(std::string path) override;
97
98         // write a copy. if name is same as original append " (n)" for n copies
99         // int writeFileCopy(std::string path);
100
101          // hash()
102
103          // !=, == and = copy
104
105     private:
106
114         void row_ptrs_to_pixels(const JSAMPARRAY& row_ptrs);
115
121         void pixels_to_row_ptrs(JSAMPARRAY& row_ptrs);
122     };
123
124 }
125
126 #endif // _LIBLMTKIMAGE_JPEG_H_
127
```

9.58 src/libLMTKimage/pluginchain.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <vector>
#include "imageshader.h"
```

Classes

- class image::PluginChain

  Class for applying multiple effects to an image, in order.

9.58.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Header file for a ordered list of effects to apply to an image.

9.59 pluginchain.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _LIBLMTKIMAGE_PLUGINCHAIN_H_
19 #define _LIBLMTKIMAGE_PLUGINCHAIN_H_
20 #pragma once
21
22 #include <vector>
23 #include "imageshader.h"
24
25
26 namespace image {
27
36     class PluginChain
37     {
38     public:
39
43         PluginChain() {}
44
48         ~PluginChain();
49
55         virtual void operator()(Image &img);
56
57         // TODO throw? or let vector throw
63         ImageShader* operator[](size_t i);
64
71         void addEffect(ImageShader* fx, size_t passes);
72
79         void addEffect(ImageShader *fx);
80
86         size_t size();
87
93         bool empty();
94
95     private:
96
97         std::vector<ImageShader*> chain;
99         //size_t completed;
100
101     };
102
103     // TODO ostream, tostring
104
105 }
106
107 #endif // _LIBLMTKIMAGE_PLUGINCHAIN_H_
```

9.60 src/libLMTKimage/png.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <string>
#include <vector>
#include <png.h>
#include "imagefile.h"
#include "../utils/utils.h"
```

Classes

- class image::PNG

  PNG image class.

Variables

- constexpr size_t **PNG_HEADER_BYTES** = 8

  @constexpr PNG_HEADER_BYTES The number of header bytes read when comparing the file header

9.60.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

December 2021

Contains the PNG image class.

The PNG class uses the libpng interface for file I/O and compression, and is a high level representation of a PNG image.

9.61 png.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
22 #ifndef _LIBLMTKIMAGE_PNG_H_
23 #define _LIBLMTKIMAGE_PNG_H_
24 #pragma once
25
26 //#define _CRT_SECURE_NO_DEPRECATE
27
28 #include <string>
29 #include <vector>
30 #include <png.h>
31 #include "imagefile.h"
32 #include "../utils/utils.h"
33
38 constexpr size_t PNG_HEADER_BYTES = 8;
39
40 namespace image {
41
42     // TODO right now if only writes 8bit channel RGBA, allow more options
43     // TODO PNG info (date author etc) struct or class or hashmap.
52     class PNG : public ImageFile
53     {
54     private:
55         // FILE* file;
56         // image info
57     public:
58         // TODO instantiate PNG info as well.
67         PNG(unsigned int w, unsigned int h);
68
69         // TODO instantiate PNG info as well; exception
80         PNG(std::string path);
81
82         // TODO spec
83         // copy image to saveable png
84         PNG(const Image& other);
85
86         // TODO it is here for future use
90         ~PNG() override;
91
92         // TODO == != operator etc.
93
94         // TODO return bool, use exceptions instead of return codes
106         void readFile(std::string path) override;
107
108         // TODO append .png if path is not already .png end
120         void writeFile(std::string path) override;
121
122         // TOD write a copy to same path as read. if name is same as original append
123         // " (n)" for nth copy
124         // int writeFileCopy(std::string path);
125
126         // TODO hash()
127
128         // TODO !=, == and = copy override
129
130
131     private:
132         // TODO do not return int codes, either use enum or exceptions + bool return
133
144         bool isFilePNG(FILE* fp);
145
156         bool create_png_read_structs(png_structp& png_p, png_infop& info_p, png_infop& end_info_p);
157
166         bool create_png_write_structs(png_structp& png_p, png_infop& info_p);
167
168         // TODO process png info and bit depth etc.
178         int process_png_info(png_structp& png_p, png_infop& info_p);
179
190         int write_png_info(png_structp& png_p, png_infop& info_p);
191
201         int process_png_pixels(png_structp& png_p, png_infop& info_p);
202
212         int write_png_pixels(png_structp& png_p, png_infop& info_p);
213
221         void row_ptrs_to_pixels(const png_bytepp& row_ptrs);
222
230         void pixels_to_row_ptrs(png_bytepp& row_ptrs);
231
232     };
233
234 }
235
236 #endif // _LIBLMTKIMAGE_PNG_H_
```

9.62 src/libLMTKimage/rgbapixel.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <iostream>
#include <cstdint>
#include <string>
#include <math.h>
#include "color.h"
#include "hslapixel.h"
#include "../utils/utils.h"
#include "../utils/utilsmath.h"
```

Classes

- class image::RGBAPixel

  Represents pixels with color channels red, green, blue and alpha.

Functions

- std::ostream & image::operator<< (std::ostream &out, const RGBAPixel &pixel)

  Add string representation of pixel to stream.

9.62.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

December 2021

This file contains the class for representing color using the RGBA color model.

9.62.2 Function Documentation

9.62.2.1 operator<<() `std::ostream & image::operator<< (`

```
    std::ostream & out,
    const RGBAPixel & pixel )
```

Add string representation of pixel to stream.

Parameters

out	Output stream
pixel	Pixel to output to stream

9.63 rgbapixel.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
20 #ifndef _LIBLMTKIMAGE_RGBAPIXEL_H_
21 #define _LIBLMTKIMAGE_RGBAPIXEL_H_
22 #pragma once
23
24 #include <iostream>
25 #include <cstdint>
26 #include <string>
27 #include <math.h>
28 #include "color.h"
29 #include "hslapixel.h"
30 #include "../utils/utils.h"
31 #include "../utils/utilsmath.h"
32
33
34 namespace image {
35
36     class HSLAPixel;
37
38
39     // TODO overload addidion/multiplication for colors
53     class RGBAPixel
54     {
55
56         //typedef char byte_t;
57
58     public:
59
66         static RGBAPixel average(RGBAPixel p1, RGBAPixel p2);
67
74         static RGBAPixel naverage(RGBAPixel pixels[], size_t count);
75
84         static RGBAPixel weighted_average(RGBAPixel p1, RGBAPixel p2, float w1, float w2);
85
93         static RGBAPixel weighted_naverage(RGBAPixel pixels[], float weights[], size_t count);
94
95
101         RGBAPixel();
102
113         RGBAPixel(float r, float g, float b);
114
125         RGBAPixel(float r, float g, float b, float a);
126
127
141         RGBAPixel(
142             uint32_t r,
143             uint32_t g,
144             uint32_t b,
145             uint32_t a,
146             uint8_t bit_depth
147         );
148
149
156         RGBAPixel(const HSLAPixel& hsl);
157
163         void operator=(const RGBAPixel& other);
164
172         bool operator==(const RGBAPixel& other) const;
173
181         bool operator!=(const RGBAPixel& other) const;
182
192         void set(float r, float g, float b);
193
204         void set(float r, float g, float b, float a);
205
212         void set(const RGBAPixel& other);
213
219         void set(const HSLAPixel& other);
220
226         void setAlpha(float a);
```

```
227
234         void setTransparency(float transparency);
235
236         // TODO conversion to HSLA/HSV/CMYK
237         // TODO conversion to rgb8, rgb16 etc or store as rgb8 to save memory
238         //
239
240         // compares within float tolerance, unlike operator==
241         //bool equals(const RGBAPixel& other);
242
253         float dist(const RGBAPixel& other) const;
254
255         // !!! This function is subject to change.
261         std::string to_string() const; // todo stream operator instead??
262
265         float r;
268         float g;
271         float b;
274         float a;
276     private:
283         void clampRGBA();
284
285     };
286
293     std::ostream& operator<<(std::ostream& out, const RGBAPixel& pixel);
294
295 }
296
297 #endif // _LIBLMTKIMAGE_RGBAPIXEL_H_
```

9.64 src/libLMTKimage/rgbcolor.h File Reference

Copyright (c) 2022 marinarasub.

```
#include "rgbapixel.h"
```

Namespaces

- namespace image::rgbcolor

  Contains common colors as constants for convinience.

9.64.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

December 2021

This file contains useful constants and such for common RGB colors.

9.65 rgbcolor.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
19 #ifndef _LIBLMTKIMAGE_RGBCOLOR_H_
20 #define _LIBLMTKIMAGE_RGBCOLOR_H_
21 #pragma once
22
23 #include "rgbapixel.h"
24
25
26 namespace image {
27
33     namespace rgbcolor {
34
35         static const RGBAPixel TRANSPARENT(0.0, 0.0, 0.0, 0.0);
37         static const RGBAPixel WHITE(1.0, 1.0, 1.0, 1.0);
39         static const RGBAPixel GREY(0.5, 0.5, 0.5, 1.0);
41         static const RGBAPixel GRAY = GREY;
43         static const RGBAPixel BLACK(0.0, 0.0, 0.0, 1.0);
45         static const RGBAPixel RED(1.0, 0.0, 0.0, 1.0);
47         static const RGBAPixel GREEN(0.0, 1.0, 0.0, 1.0);
49         static const RGBAPixel BLUE(0.0, 0.0, 1.0, 1.0);
51         static const RGBAPixel YELLOW(1.0, 1.0, 0.0, 1.0);
53     }
54
55 }
56
57 #endif // _LIBLMTKIMAGE_RGBCOLOR_H_
```

9.66 src/libLMTKimage/sharpen.h File Reference

Copyright (c) 2022 marinarasub.

```
#include "convolution.h"
```

Classes

- class image::ImageSharpen
  class for sharpening an image

9.66.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in libLMTKimage, the image library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Class for sharpening an image

9.67 sharpen.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
19 #ifndef _LIBLMTKIMAGE_SHARPEN_H_
20 #define _LIBLMTKIMAGE_SHARPEN_H_
21 #pragma once
22
23 #include "convolution.h"
24
25 namespace image {
26
34     class ImageSharpen : public ImageConvolution
35     {
36     public:
37
38         // sharpen uses 1x1
39         static const size_t SHARP_CENTER = 1;
40
44         ImageSharpen();
45
51         ImageSharpen(float amt);
52
53
54     private:
55
59         virtual void computeKernel() override;
60
61         float amount;
62     };
63 }
64
65 #endif // _LIBLMTKIMAGE_SHARPEN_H_
```

9.68 src/utils/fft.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <stdlib.h>
#include <vector>
#include "utils.h"
#include "utilsmath.h"
```

Functions

- std::vector< math::complex > * **utils::fft::dft** (std::vector< float > &input)

  *Returns discrete fourier transform of 0..n (where n is max freq.) fhat (fourier coeff) = SUM j=0 to n-1 (input * e^(-i2Pljk/n))*
- std::vector< float > * **utils::fft::idft** (std::vector< math::complex > &coeffs)

  *Returns discrete fourier transform of 0..n (where n is max freq.) j (data) = SUM j=0 to n-1 (fhat * e^(i2Pljk/n))*
- std::vector< math::complex > * **utils::fft::fft** (std::vector< float > &input)
- std::vector< float > * **utils::fft::ifft** (std::vector< math::complex > &coeffs)
- std::vector< float > * **utils::fft::power_spec** (std::vector< math::complex > &input)
- math::complex **utils::fft::eitheta** (double theta)
- math::complex **utils::fft::cis** (double theta)
- math::complex **utils::fft::eitheta_inv** (double theta)
- math::complex **utils::fft::cis_inv** (double theta)

9.68.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in the utils library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Utility class for fourier transform

9.69 fft.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 // !!! TODO WIP !!! //
19 #ifndef _UTILS_FFT_H_
20 #define _UTILS_FFT_H_
21 #pragma once
22
23 #include <stdlib.h>
24 #include <vector>
25 #include "utils.h"
26 #include "utilsmath.h"
27
28
29 namespace utils {
30
31     namespace fft {
32
33         std::vector<math::complex>* dft(std::vector<float> &input);
34
35         std::vector<float>* idft(std::vector<math::complex> &coeffs);
36
37         std::vector<math::complex>* fft(std::vector<float>& input);
38
39         std::vector<float>* ifft(std::vector<math::complex>& coeffs);
40
41         std::vector<float>* power_spec(std::vector<math::complex>& input);
42
43         math::complex eitheta(double theta); // e^itheta
44         math::complex cis(double theta); // e^itheta
45         math::complex eitheta_inv(double theta); // e^itheta
46         math::complex cis_inv(double theta); // e^itheta
47     }
48 }
49
50 #endif // _UTILS_FFT_H_
```

9.70 src/utils/file.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <string>
```

Functions

- std::string **utils::file_extension** (std::string path)

9.70.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in the utils library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Utility class for file handling

9.71 file.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _UTILS_FILE_H_
19 #define _UTILS_FILE_H_
20 #pragma once
21
22 #include <string>
23
24
25
26 namespace utils {
27
28     // WIP
29     // gets file extension without dot
30     // no regard for validity of path or extension
31     std::string file_extension(std::string path);
32
33 }
34
35
36 #endif // _UTILS_FILE_H_
```

9.72 src/utils/threadpool.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <thread>
#include <mutex>
#include <condition_variable>
#include <functional>
#include <queue>
#include <iostream>
#include <stdexcept>
#include <stdarg.h>
#include <tuple>
```

Classes

- struct threadpool::task

  A task for a thread which stores a void calback.
- class threadpool::TaskQueue

  A FIFO queue of callback tasks that is thread-safe.
- class threadpool::ThreadPool

  Class for a pool of flexible worker threads which can be given miscellaneous tasks to complete.

9.72.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in the utils library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Contains handy string utilities.

9.73 threadpool.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _UTILS_THREADPOOL_H_
19 #define _UTILS_THREADPOOL_H_
20
21 #include <thread>
22 #include <mutex>
23 #include <condition_variable>
24 #include <functional>
25 #include <queue>
26 #include <iostream>
27 // #include <chrono>
28 // #include <random> // just for testing
29 #include <stdexcept>
30 #include <stdarg.h>
31 #include <tuple>
32
33
34 namespace threadpool {
35
40     struct task
41     {
42         std::function<void()> callback;
43     };
44
45
46     // TODO priority queue
51     class TaskQueue
52     {
53     public:
54
58         TaskQueue();
59
65         void enqueue(const task &t);
66
```

```
73          task dequeue();
74
80          bool empty() const;
81
87          size_t size() const;
88
89      private:
90
91          std::queue<task> task_queue;
93          std::mutex queue_mtx;
94      };
95
96
106     class ThreadPool
107     {
108     public:
109
115         ThreadPool();
116
122         ThreadPool(size_t num_threads);
123
130         ~ThreadPool();
131
138         void start();
139
148         void join();
149
158         void stop();
159
166         void detach();
167
178         template<typename ... Args>
179         void addTask(std::function<void(Args...)> callback, Args... args)
180         {
181             std::function<void()> store = [=]{
182                 callback(args...);
183             };
184             tasks.enqueue(task{store});
185             recheck.notify_one();
186         }
187
198         template<typename ... Args>
199         void addTask(void(*callback)(Args...), Args... args)
200         {
201             std::function<void()> store = [=]{
202                 callback(args...);
203             };
204             tasks.enqueue(task{store});
205             recheck.notify_one();
206         }
207
213         size_t tasks_remaining() const;
214
221         size_t tasks_completed() const;
222
223     private:
224
228         void join_all_threads();
229
233         void detach_all_threads();
234
238         void destroy();
239
249         static void worker_thread(ThreadPool *tpool);
250
256         static void execute_task(const task &t);
257
264         bool should_terminate();
265
272         bool should_wait();
273
280         bool should_run_task();
281
282         TaskQueue tasks;
284         std::vector<std::thread*> workers;
286         size_t idle_threads;
288         std::mutex mtx;
290         // this DOES NOT guarantee all tasks added will be completed
291         bool stop_request;
293         // this guarantees all tasks added will be completed
294         bool finish_up;
296         std::condition_variable complete;
299         std::condition_variable recheck;
302         size_t total_tasks_completed;
304     };
305
306 }
```

```
307 #endif // _UTILS_THREADPOOL_H_
```

9.74 src/utils/timer.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <chrono>
```

Classes

- class utils::Timer

  Represents a timer.

Typedefs

- using **utils::hrc** = high_resolution_clock

9.74.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in the utils library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Contains a basic timer wrapper for standard library chrono

9.75 timer.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _UTILS_TIMER_H_
19 #define _UTILS_TIMER_H_
20
21 #include <chrono>
22
23
24 namespace utils {
25
26     using namespace std::chrono;
27     using hrc = high_resolution_clock;
28
33     class Timer
34     {
35     public:
36
40         Timer();
41
47         void start();
48
54         double stop(); // return time in s.
55
56         // TODO to string.
57
58         // TODO pause/lap
59
60     private:
61
62         hrc::time_point start_point;
64         //time_p
65     };
66 }
67
68
69 #endif // _UTILS_TIMER_H_
```

9.76 src/utils/utils.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <limits>
#include <cstdint>
#include <string>
```

Functions

- uint32_t utils::uint_max_val (uint8_t bit_depth)

  Returns the maximum value for an unsigned int of a given bit depth.
- uint64_t utils::uint64_next_pwr2 (uint64_t val)

  Determines the tightest power of 2 that is equal or greater than a number.
- std::string **utils::get_size_string** (size_t byte_size)

9.76.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in the utils library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

File containing various utility functions

9.76.2 Function Documentation

9.76.2.1 uint64_next_pwr2() `uint64_t utils::uint64_next_pwr2 ( uint64_t val )`

Determines the tightest power of 2 that is equal or greater than a number.

Parameters

val	Number to check

9.76.2.2 uint_max_val() `uint32_t utils::uint_max_val ( uint8_t bit_depth )`

Returns the maximum value for an unsigned int of a given bit depth.

The maximum value is where bit_depth number of bits are set.

Parameters

bit_depth	The bit depth, in bits

Returns

The maximum unsigned integer value of a bit depth

Note

is the bit_depth given is $>$ 32, use UINT32_MAX

9.77 utils.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _UTILS_UTILS_H_
19 #define _UTILS_UTILS_H_
20 #pragma once
21
22 #include <limits>
23 #include <cstdint>
24 #include <string>
25
26 namespace utils
27 {
36     uint32_t uint_max_val(uint8_t bit_depth);
37
38
44     uint64_t uint64_next_pwr2(uint64_t val);
45
46     // WIP, given number of bytes return string rep. in readeable format
47     // (kb, mb, gb etc)
48     std::string get_size_string(size_t num_bytes);
49
50 }
51
52 #endif // _UTILS_UTILS_H_
```

9.78 src/utils/utilsmath.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <math.h>
#include <complex>
```

Namespaces

- namespace utils::math

  contains math utility functions.

Typedefs

- typedef std::complex< double > **utils::math::complex**

  Represents a complex number using doubles.
- typedef std::complex< float > **utils::math::complexf**

  Represents a complex number using floats.

Functions

- complex utils::math::real_to_complex (double val)

  Convert a number to a complex type.
- complexf utils::math::real_to_complexf (float val)

  Convert a number to a complex type, using float.
- double utils::math::sqr (double val)

  Returns the square of a number.
- float utils::math::sqrf (float val)

  Returns the square of a number, for float.
- complex utils::math::sqr (complex val)

  Squares a complex number.
- complexf utils::math::sqrf (complexf val)

  Squares a complex number, for float.

- double utils::math::nsum (double arr[ ], size_t n)

  Sums the number in an array of size n.
- float utils::math::nsumf (float arr[ ], size_t n)

  Sums the number in an array of size n, for float32.
- double utils::math::dist (double x1, double y1, double x2, double y2)

  Return 2D distance between two points.
- double utils::math::gaussian (double mean, double sigma, double x)

  Compute the gaussian value of a variable x with mean and standard deviation.
- double utils::math::gaussian2d (double mean_x, double mean_y, double sigma_x, double sigma_y, double x, double y)

  Compute the 2D gaussian value of a variable x with means and standard deviations.

Variables

- constexpr complex **utils::math::IMAGINARY_UNIT** = complex(0, 1)

  The imaginary unit, also known as i, sqrt(-1).
- constexpr complexf **utils::math::IMAGINARY_UNIT_F** = complexf(0, 1)

  The imaginary unit, also known as i, sqrt(-1) using float32.
- const double **utils::math::EULER** = std::exp(1)

  Euler's number.
- constexpr double **utils::math::PI** = 3.141592653589793238463

  Pi to a decent precision.

9.78.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in the utils library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Contains various handy math functions

9.79 utilsmath.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _UTILS_UTILSMATH_H_
19 #define _UTILS_UTILSMATH_H_
20 #pragma once
21
22 #include <math.h>
23 #include <complex>
24
25
26 namespace utils {
27
36     namespace math {
37
41         typedef std::complex<double> complex;
42
46         typedef std::complex<float> complexf;
47
51         constexpr complex IMAGINARY_UNIT = complex(0, 1);
52
56         constexpr complexf IMAGINARY_UNIT_F = complexf(0, 1);
57
61         const double EULER = std::exp(1);
62
66         constexpr double PI = 3.14159265358979238463;
67
74         complex real_to_complex(double val);
75
82         complexf real_to_complexf(float val);
83
90         double sqr(double val);
91
98         float sqrf(float val);
99
106          complex sqr(complex val);
107
114          complexf sqrf(complexf val);
115
123          double nsum(double arr[], size_t n);
124
132          float nsumf(float arr[], size_t n);
133
134          //double stdev();
135          //double mean();
136          //double median();
137
148          double dist(double x1, double y1, double x2, double y2);
149
160          double gaussian(double mean, double sigma, double x);
161
175          double gaussian2d(
176              double mean_x,
177              double mean_y,
178              double sigma_x,
179              double sigma_y,
180              double x,
181              double y);
182     }
183 }
184
185 #endif // _UTILS_UTILSMATH_H_
```

9.80 src/utils/utilsstring.h File Reference

Copyright (c) 2022 marinarasub.

```
#include <string>
```

Functions

- std::string **utils::remove_spaces** (std::string in)
- size_t **utils::count_char** (std::string in, char c)

9.80.1 Detailed Description

Copyright (c) 2022 marinarasub.

LMTK / Light Multimedia ToolKit

This file is included in the utils library for LMTK LMTK and its subprojects are available under the MIT license.

Author

marinarasub

Date

January 2022

Contains handy string utilities.

9.81 utilsstring.h

Go to the documentation of this file.

```
1 /*****************************************************************************/
18 #ifndef _UTILS_UTILSSTRING_H_
19 #define _UTILS_UTILSSTRING_H_
20 #pragma once
21
22 #include <string>
23
24
25 namespace utils {
26
27     // removes all spaces in string
28     std::string remove_spaces(std::string in);
29
30     // counts number of given char in string
31     size_t count_char(std::string in, char c);
32
33 }
34
35
36 #endif // _UTILS_UTILSSTRING_H_
```

Index