ColoredWaveform 1.0.1

by zplane.development

(c) 2018 zplane.development GmbH & Co. KG

February 13, 2018

Contents

1 ColoredWaveform Documentation

1.1 Introduction

ColoredWaveform is an audio waveform display SDK. It aids developers at displaying audio waveforms in different styles and resolutions. The SDK enables realtime display on screen as well as image generation, e.g. for use in a web environment. The SDK currently provides two types of waveforms: a standard monochrome waveform and a multicolored waveform that colors the waveform along the horizontal axis based on the momentary frequency content. Various envelope types provide control over the way audio samples are summarized when displaying the waveform.

The project contains the required libraries for the operating system ColoredWaveform was licensed for with the appropriate header files. An example application illustrates the functionality of this SDK.

This document is structured as follows: The first part contains the API documentation of the ColoredWaveform SDK. The API documentation contains naming conventions and function descriptions of the C++-API. In the second part, a detailed explanation of the example application is provided.

1.2 API Documentation

The ColoredWaveformIf class provides the functionality to extract a low-resolution representation from the audio signal (i.e. a thumbnail) in the form of a WaveformContainer instance. From the WaveformContainer, audio waveforms at arbitrary starting positions and lengths can be derived. The WaveformContainer cannot provide data at image resolutions that are higher than the resolution specified at instance creation of the ColoredWaveformIf class. If higher resolutions are required, this has to be done with a new instance of ColoredWaveformIf.

When creating an instance of ColoredWaveformIf, two enums can be used to control the appearance of the audio waveform:

ColoredWaveformIf::WaveformType lets you select between a monochrome waveform and multicolored waveform. For the multicolored waveform, audible frequencies are mapped to colors: low frequencies are mapped to red, mid frequencies to green and high frequencies to blue. Most sounds will consist of a combination of low, mid and high frequencies and will thus be a combination of the three colors.

The ColoredWaveformIf::EnvelopeType affects the way amplitudes of the audio signal are summarized to create the waveform display. A waveform display consist of multiple, consecutive vertical lines along a horizontal axis - one line per horizontal pixel position. Each line summarizes a block of audio samples. A line is characterized by an amplitude range and a color. The amplitude range summarizes the audio samples of the block as illustrated in the figure below. When selecting ColoredWaveformIf::absMax as the EnvelopeType, the range is defined by the absolute maximum sample amplitude, when ColoredWaveformIf::minMax is selected, the range will be displayed between the minimum and maximum sample amplitude and

ColoredWaveformIf::rms will display the range as the root mean square of all amplitudes within that block.

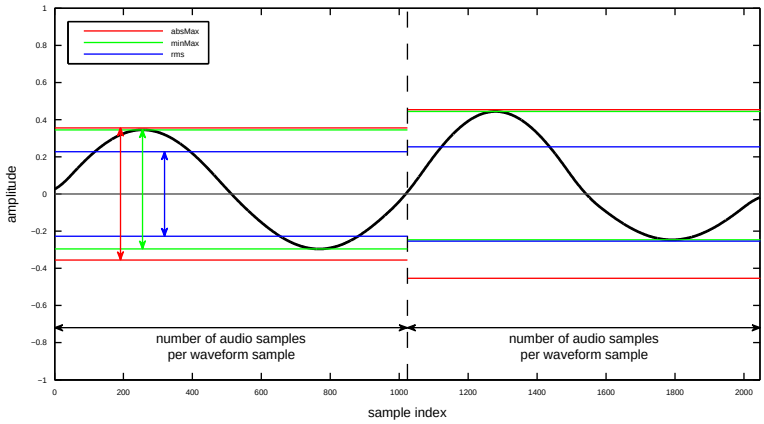

Figure 1: Envelope types

The processing is based on the push principle: successive blocks of input audio frames are pushed into the ColoredWaveformIf::process() function. For each complete block of audio frames, the process function will compute a WaveformContainer::Sample for each audio channel and write/append those to the WaveformContainer. The WaveformContainer itself can be obtained by calling ColoredWaveformIf::getWaveformContainer(). Access to waveforms at different starting positions and screen resolutions is provided by the WaveformContainer::Iterator class. An instance of this class can be obtained by calling WaveformContainer::getIterator() with the desired image resolution. The Iterator can be used to step through each horizontal pixel position in order to draw the waveform at that position. Writing to the WaveformContainer (by calling ColoredWaveformIf::process()) and reading from the WaveformContainer (by calling WaveformContainer::Iterator::next() and WaveformContainer::Iterator::getSamplesForAllChannels()) can be done interchangeably. However, these operations are not thread-safe, i.e. the calling application has to make sure that reading and writing does not happen concurrently.

1.2.1 Memory Allocation

The ColoredWaveform SDK does not allocate any buffers handled by the calling application. The input buffer has to be allocated/created by the calling application. If the WaveformContainer should persist after deletion of the ColoredWaveformIf instance, a copy has to be made.

1.2.2 Instance Control Functions

- **static ErrorType ColoredWaveformIf::createInstance (ColoredWaveform↩ If∗& pColoredWaveform, float fSampleRate, int iNumberOfChannels, int iNumberOfAudioSamplesPerWaveformSample, WaveformType extractor, Envelope↩ Type envelopeType)**
  Creates a new instance of ColoredWaveformIf. The handle to the new instance

is returned in parameter pColoredWaveform. fSampleRate and iNumberOf↩ Channels denote the input samplerate and number of channels, respectively. The parameter iNumberOfAudioSamplesPerWaveformSample specifies the resolution at which data is stored in the WaveformContainer which is equal to the maximum obtainable waveform resolution. The parameter extractor specifies the type of waveform to extract and envelopeType controls the appearance of the waveform.
The function returns an error code (see ColoredWaveformIf::ErrorType). A call to this function is mandatory.

- **static void ColoredWaveformIf::destroyInstance (ColoredWaveformIf*& p↩ ColoredWaveform)**
  Destroys the ColoredWaveformIf instance provided by pColoredWaveform. A call to this function is mandatory.

1.2.3 Processing Function

- **ErrorType ColoredWaveformIf::process (float const *const *const ppfInput↩ Buffer, int iNumberOfFrames)**
  This function extracts the low-resolution representation of the audio waveform. The parameter ppfInputBuffer contains the audio samples for each channel and iNumberOfFrames specifies how many audio frames (i.e. samples in each channel) are provided.

- **ErrorType ColoredWaveformIf::finishProcessing()**
  A call to this functions ends the processing and computes the final WaveformContainer::Sample. No more calls to the process function are possible after a call to this function.

1.2.4 Result Retrieving Function

- **ErrorType ColoredWaveformIf::getWaveformContainer ()**
  This function returns a handle to the internal WaveformContainer. In order to prevent its destruction upon destruction of the ColoredWaveformIf instance, a copy has to be created before destroying the ColoredWaveformIf instance. Note that you can read and write to the WaveformContainer interchangeably, but not concurrently.

- **Iterator WaveformContainer::getIterator (float fStartTimeInSeconds, float fSecondsPerWaveformSample) const**
  Returns an Iterator that enables stepping through the audio waveform and displaying one pixel at a time. Parameters fStartTimeInSeconds and fSecondsPer↩ WaveformSample can be used to obtain waveform data at different resolutions.

- **bool WaveformContainer::Iterator::next()**
  Computes the waveform data for the next pixel to display. It returns true if the data could be computed and otherwise false.

- **std::vector<WaveformContainer::Sample> WaveformContainer::Iterator↩ ::getCurrentFrame()**
  This function returns the waveform data for the current pixel position. It returns a std::vector that contains the waveform data in the form of a WaveformContainer::Sample object per audio channel.

1.3 Command Line Usage Example

The command line example can be executed by the following command

```
ColoredWaveformCl -i <inputFile> -o <outputSVGFile> -w <width> -h <height>
```

The complete code can be found in the example source file ColoredWaveformCl↩ Main.cpp.
In the first step, we declare a ColoredWaveformIf pointer and create an instance of the ColoredWaveformIf class:

```
ColoredWaveformIf* pInstanceHandle = 0;

eError = ColoredWaveformIf::createInstance(pInstanceHandle,
                                           inputFile.GetSampleRate(),
                                           inputFile.GetNumOfChannels(),
                                           512,
                                           ColoredWaveformIf::multicolored
  ,
                                           ColoredWaveformIf::minMax);
```

We then read chunks of data from our input file,

```
while (bReadNextFrame)
{
    // read audio data
    int iNumFramesRead (inputFile.Read (ppfInput, kBlockSize));
```

And push each chunk into our process() function.

```
    eError = pInstanceHandle->process (ppfInput, iNumFramesRead);
    }
```

After the entire file has been read and pushed into the ColoredWaveform instance, we call finishProcessing

```
pInstanceHandle->finishProcessing();
```

and create a copy of the internal WaveformContainer.

```
WaveformContainer waveformContainer (pInstanceHandle->
  getWaveformContainer());
```

Finally we destroy the ColoredWaveformIf instance

```
ColoredWaveformIf::destroyInstance (pInstanceHandle);
```

We then write the extracted waveform to an SVG file by calling

```
SVGWaveformWriter::write (waveformContainer, iWidth, iHeight, backgroundColor,
  outputPath);
```

with the specified width and height.
Within the SVGWaveformWriter, an Iterator is obtained from the waveform container with the given start time and resolution.

```
    // get iterator
    WaveformContainer::Iterator iterator (container.getIterator (
timeLimitsInSeconds.getStart(), timeLimitsInSeconds.getLength() / (1.f * iWidthInPixels)));
```

The subsequent while loop steps through each pixel position,

```
    while (iterator.next() && (iHorizontalPosition < iWidthInPixels))
    {
```

obtains the current waveform frame, i.e. the data for drawing the waveform at this horizontal position

```
        std::vector<WaveformContainer::Sample> currentFrame (iterator.getSamplesForAllChannels());
```

and writes the data for each channel to the SVG file

```
        for (int iChannel (0); iChannel < (int) currentFrame.size(); ++iChannel)
        {
            FloatRange amplitudeRange (currentFrame[iChannel].getLowerAmplitudeBound(),
currentFrame[iChannel].getUpperAmplitudeBound());
            FloatRange verticalImageLimits (iChannel * fHeightPerChannel, (iChannel+1) *
fHeightPerChannel);
            FloatRange verticalRectangleRange (amplitudeRangeToImageRange (amplitudeRange,
amplitudeLimits, verticalImageLimits));
            RGBColor color (currentFrame[iChannel].getColor());
            if (map != NULL)
                color = map (color);

            outputFile << createSVGRectangleTag (verticalRectangleRange,
FloatRange ((float) iHorizontalPosition, iHorizontalPosition + 1.f), color);
            outputFile << std::endl;
        }
        ++iHorizontalPosition;
    }
```

RGBColors can be mapped to a different colorspace by implementing a colorMappingFunction.

The above code snippets demonstrated the basic functionality of the ColoredWaveform library.

1.4 Support

Support for the source code is - within the limits of the agreement - available from:

zplane.development GmbH & Co KG
grunewaldstr. 83
d-10823 berlin
germany

fon: +49.30.854 09 15.0
fax: +49.30.854 09 15.5

@: info@zplane.de

2 Class Index

2.1 Class List

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

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 ColoredWaveformIf Class Reference

Collaboration diagram for ColoredWaveformIf:

Public Types

- enum ErrorType {
  noError, memError, invalidFunctionParamError, invalidFunctionCallError,
  unknownError, numErrors }

- enum VersionType { major, minor, patch, revision }
- enum WaveformType { monochrome, multicolored }
- enum EnvelopeType { absMax, minMax, rms }

Public Member Functions

- virtual ErrorType process (float const *const *const ppfInputBuffer, int iNumber↩ OfFrames)=0
- virtual ErrorType finishProcessing ()=0
- virtual const WaveformContainer & getWaveformContainer ()=0

Static Public Member Functions

- static const int getVersion (const VersionType eVersionIdx)
- static const char * getBuildDate ()
- static ErrorType createInstance (ColoredWaveformIf *&pColoredWaveform, float fSampleRate, int iNumberOfChannels, int iNumberOfAudioSamplesPerWaveform↩ Sample, WaveformType extractor, EnvelopeType envelopeType)
- static void destroyInstance (ColoredWaveformIf *&pColoredWaveform)

Protected Member Functions

- virtual ~ColoredWaveformIf ()

4.1.1 Member Enumeration Documentation

EnvelopeType `enum ColoredWaveformIf::EnvelopeType`

Envelope type, specifying how a block of audio samples should be summarized

Enumerator

absMax	absolute maximum of the block, displayed symmetrically
minMax	minimum and maximum of the block
rms	root mean square of all samples in the block, displayed symmetrically

ErrorType `enum ColoredWaveformIf::ErrorType`

Error codes

Enumerator

noError	no error occurred
memError	memory allocation failed
invalidFunctionParamError	one or more function parameters are not valid
invalidFunctionCallError	function call not allowed

Enumerator

unknownError	unknown error occurred
numErrors	

VersionType `enum ColoredWaveformIf::VersionType`

Version number

Enumerator

major	major version number
minor	minor version number
patch	patch version number
revision	revision number

WaveformType `enum ColoredWaveformIf::WaveformType`

Type of waveform to display

Enumerator

monochrome	monochrome waveform
multicolored	multicolored waveform displaying the frequency content

4.1.2 Constructor & Destructor Documentation

~ColoredWaveformIf() `virtual ColoredWaveformIf::~ColoredWaveformIf ( ) [inline], [protected], [virtual]`

4.1.3 Member Function Documentation

createInstance()

```
static ErrorType ColoredWaveformIf::createInstance (
        ColoredWaveformIf *& pColoredWaveform,
        float fSampleRate,
        int iNumberOfChannels,
        int iNumberOfAudioSamplesPerWaveformSample,
        WaveformType extractor,
        EnvelopeType envelopeType ) [static]
```

Creates an instance of ColoredWaveform.

Parameters

pColoredWaveform	reference to ColoredWaveform instance pointer
fSampleRate	sample rate of input audio
iNumberOfChannels	number of input audio channels
iNumberOfAudioSamplesPer↩ WaveformSample	the resulting WaveformContainer will store a low-resolution version of the audio data and hence summarize a number of audio samples. This parameter determines how many audio samples will be summarized. From this low-resolution version only waveforms with even lower horizontal resolutions can be obtained (see documentation of WaveformContainer) and hence a trade-off has to be made between the required maximum display resolution and memory consumption.

Returns

an error code, or 0 if no error occurred

destroyInstance() `static void ColoredWaveformIf::destroyInstance ( ColoredWaveformIf *& pColoredWaveform ) [static]`

Destroys an instance of ColoredWaveform.

Parameters

pColoredWaveform	reference to ColoredWaveform instance pointer

finishProcessing() `virtual ErrorType ColoredWaveformIf::finishProcessing ( ) [pure virtual]`

Compute final waveform frame from remaining audio frames and finish processing.

Only call this function after all audio data has been provided. After a call to this function, the 'process' function will return an invalidFunctionCallError.

getBuildDate() `static const char* ColoredWaveformIf::getBuildDate ( ) [static]`

Returns the build date string.

getVersion() `static const int ColoredWaveformIf::getVersion ( const VersionType eVersionIdx ) [static]`

Returns major version, minor version, patch and build number of this Kort version.

Parameters

eVersionIdx	requested version number part

Returns

version number part

getWaveformContainer() `virtual const WaveformContainer& ColoredWaveformIf::getWaveformContainer ( ) [pure virtual]`

Returns the WaveformContainer from which waveforms at various horizontal resolutions can be derived.

This function returns a handle to the internal WaveformContainer. In order to prevent its destruction upon destruction of the ColoredWaveform instance, create a copy before destroying the ColoredWaveform instance.

This function is not thread-safe, i.e. it should not be called concurrently with the Process function. The data of the resulting WaveformContainer can and will change when the Process function is called.

process()
```
virtual ErrorType ColoredWaveformIf::process (
        float const *const *const ppfInputBuffer,
        int iNumberOfFrames ) [pure virtual]
```

Process a block of audio.

This function can be called multiple times in order to provide successive chunks of the input audio signal.

Parameters

ppfInputBuffer	pointer to the input data chunk. ppfInputBuffer[i] points to the i-th audio channel. The input data can be in any number range; the resulting WaveformContainer will summarize the data as it is, i.e. it will not truncate or normalize this data during processing to produce the resulting WaveformContainer::WaveformSamples.
iNumberOfFrames	The number of audio samples in each audio channel of the provided input data chunk.

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

- ColoredWaveformIf.h

4.2 FloatRange Class Reference

Collaboration diagram for FloatRange:

Public Member Functions

- FloatRange ()
- FloatRange (float start, float end)
- virtual ~FloatRange ()
- float getStart () const
- float getEnd () const
- float getLength () const
- FloatRange operator+ (float rhs) const
- FloatRange operator* (float rhs) const
- float clip (float value) const
- FloatRange clippedBy (const FloatRange &other)

4.2.1 Constructor & Destructor Documentation

FloatRange() [1/2] `FloatRange::FloatRange ( ) [inline]`

Creates a range with start and end equal to 0.

FloatRange() [2/2]
```
FloatRange::FloatRange (
        float start,
        float end ) [inline]
```

Creates a range with given start and end.

~FloatRange() `virtual FloatRange::~FloatRange ( ) [inline], [virtual]`

Destructor.

4.2.2 Member Function Documentation

clip()
```
float FloatRange::clip (
        float value ) const [inline]
```

Clip a value to the current range. If the value is out of bounds, it will be set to the closest bound.

clippedBy()
```
FloatRange FloatRange::clippedBy (
        const FloatRange & other ) [inline]
```

Return a range that has start and end clipped to another range.

getEnd() `float FloatRange::getEnd ( ) const [inline]`

Returns the end.

getLength() `float FloatRange::getLength ( ) const [inline]`

Returns the length

getStart() `float FloatRange::getStart ( ) const [inline]`
Returns the start.

operator∗() `FloatRange FloatRange::operator* ( float rhs ) const [inline]`
Returns a range with both start and end scaled by rhs.

operator+() `FloatRange FloatRange::operator+ ( float rhs ) const [inline]`
Returns a range with rhs added to both start and end.
The documentation for this class was generated from the following file:

- FloatRange.h

4.3 RGBColor Class Reference

Collaboration diagram for RGBColor:

Public Member Functions

- RGBColor ()
- RGBColor (float fRed, float fGreen, float fBlue)
- float getRed () const
- float getGreen () const
- float getBlue () const
- std::vector< float > getAsVector () const
- void setRed (float iValue)
- void setGreen (float iValue)
- void setBlue (float iValue)
- bool operator== (const RGBColor &rhs)

4.3.1 Detailed Description

Color class.

4.3.2 Constructor & Destructor Documentation

RGBColor() [1/2] `RGBColor::RGBColor ( ) [inline]`
Creates a black color.

RGBColor() [2/2] `RGBColor::RGBColor ( float fRed, float fGreen, float fBlue ) [inline]`
Creates a color with the given RGB values. Values are clipped to the range [0...1].

4.3.3 Member Function Documentation

getAsVector() `std::vector<float> RGBColor::getAsVector ( ) const [inline]`
Returns a three-element vector with values for red, green and blue (in that order).

getBlue() `float RGBColor::getBlue ( ) const [inline]`
Returns the blue color component.

getGreen() `float RGBColor::getGreen ( ) const [inline]`
Returns the green color component.

getRed() `float RGBColor::getRed ( ) const [inline]`
Returns the red color component.

operator==() `bool RGBColor::operator== ( const RGBColor & rhs ) [inline]`
Returns true if two RGBColors are equal.

setBlue() `void RGBColor::setBlue ( float iValue ) [inline]`
Sets the blue color component.

setGreen() `void RGBColor::setGreen ( float iValue ) [inline]`
Sets the green color component.

setRed() `void RGBColor::setRed ( float iValue ) [inline]`
Sets the red color component.
The documentation for this class was generated from the following file:

- RGBColor.h

4.4 SVGWaveformWriter Class Reference

Collaboration diagram for SVGWaveformWriter:

Public Types

- typedef RGBColor(∗ colorMappingFunction) (const RGBColor &)

Static Public Member Functions

- static RGBColor exampleColorMappingFct (const RGBColor &color)
- static bool write (const WaveformContainer &container, int iWidthInPixels, int iHeightInPixels, const RGBColor &backgroundColor, const std::string &outputFilePath, colorMappingFunction map=NULL)
- static bool write (const WaveformContainer &container, const FloatRange &amplitudeLimits, const FloatRange &timeLimitsInSeconds, int iWidthInPixels, int iHeightInPixels, const RGBColor &backgroundColor, const std::string &outputFilePath, colorMappingFunction map=NULL)

4.4.1 Member Typedef Documentation

colorMappingFunction `typedef RGBColor(* SVGWaveformWriter::colorMappingFunction) (const RGBColor &)`

Function pointer for mapping colors to different color space

4.4.2 Member Function Documentation

exampleColorMappingFct() `static RGBColor SVGWaveformWriter::exampleColorMappingFct ( const RGBColor & color ) [inline], [static]`

Example for a color mapping function that scales and shifts each color channel differently.

write() `[1/2]`

```
static bool SVGWaveformWriter::write (
    const WaveformContainer & container,
    int iWidthInPixels,
    int iHeightInPixels,
    const RGBColor & backgroundColor,
    const std::string & outputFilePath,
    colorMappingFunction map = NULL ) [inline], [static]
```

Write a complete waveform to file.

Parameters

container	WaveformContainer holding the waveform data
iWidthInPixels	width of resulting image in pixels
iHeightInPixels	height of resulting image in pixels
backgroundColor	background color of image
outputFilePath	path to output file
map	function pointer to map colors to a different color space

Returns

true if the waveform was written successfully, other wise false

write() [2/2] `static bool SVGWaveformWriter::write (`
`const WaveformContainer & container,`
`const FloatRange & amplitudeLimits,`
`const FloatRange & timeLimitsInSeconds,`
`int iWidthInPixels,`
`int iHeightInPixels,`
`const RGBColor & backgroundColor,`
`const std::string & outputFilePath,`
`colorMappingFunction map = NULL ) [inline], [static]`

Write a waveform section to the file.

Parameters

container	WaveformContainer holding the waveform data
amplitudeLimits	amplitude range to display
timeLimitsInSeconds	time range to display (in seconds)
iWidthInPixels	width of resulting image in pixels
iHeightInPixels	height of resulting image in pixels
backgroundColor	background color of image
outputFilePath	path to output file
map	function pointer to map colors to a different color space

Returns

true if the waveform was written successfully, other wise false

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

- SVGWaveformWriter.h

4.5 WaveformContainer Class Reference

Collaboration diagram for WaveformContainer:

Classes

- class Iterator
- class Sample

Public Member Functions

- WaveformContainer (Impl ∗pImpl)
- WaveformContainer (const WaveformContainer &other)

- ~WaveformContainer ()
- WaveformContainer & operator= (const WaveformContainer &other)
- float getSampleRate () const
- int getNumberOfChannels () const
- int getNumberOfAudioSamplesPerWaveformSample () const
- float getLengthInSeconds () const
- int getNumberOfSamples () const
- float getMaximumResolutionInSeconds () const
- Iterator getIterator (float fStartTimeInSeconds, float fSecondsPerWaveform↩ Sample) const

4.5.1 Detailed Description

Container storing a low-resolution version of the audio waveform. This class allows you to derive scaled down versions of the waveform.

The waveform data is stored as a sequence of WaveformSamples for each audio channel. Each WaveformSample consists of an amplitude range that summarizes a block of audio and that can be used to draw the waveform at single horizontal pixel position.

In order to obtain the waveform at different screen resolutions, an Iterator can be obtained that enables you to step through and display the individual WaveformSamples.

It is possible to read from and write to the WaveformContainer in any order. However, reading and writing is not thread safe, i.e. you must not read and write simultaneously.

4.5.2 Constructor & Destructor Documentation

WaveformContainer() [1/2] `WaveformContainer::WaveformContainer ( Impl * pImpl )`

WaveformContainer() [2/2] `WaveformContainer::WaveformContainer ( const WaveformContainer & other )`

Copy constructor. Creates a deep copy of a WaveformContainer object.

~WaveformContainer() `WaveformContainer::~WaveformContainer ( )`

Destructor.

4.5.3 Member Function Documentation

getIterator() `Iterator WaveformContainer::getIterator (`
`float fStartTimeInSeconds,`
`float fSecondsPerWaveformSample ) const`

Returns an iterator for a given position and resolution.

The Iterator can be used to step through each horizontal position (pixel) in order to draw the audio waveform.

Parameters

fStartTimeInSeconds	audio start time in seconds
fSecondsPerWaveformSample	audio length seconds that is summarized by each WaveformSample

Returns

an Iterator with the given specifications

getLengthInSeconds() `float WaveformContainer::getLengthInSeconds ( ) const`
Returns the length of the audio in seconds.

getMaximumResolutionInSeconds() `float WaveformContainer::getMaximumResolution↩InSeconds ( ) const`
Returns the maximum resolution (i.e. the maximum time distance) between consecutive waveform samples in seconds.

getNumberOfAudioSamplesPerWaveformSample() `int WaveformContainer::get↩NumberOfAudioSamplesPerWaveformSample ( ) const`
Returns the number of source samples per waveform sample.

getNumberOfChannels() `int WaveformContainer::getNumberOfChannels ( ) const`
Returns the number of channels.

getNumberOfSamples() `int WaveformContainer::getNumberOfSamples ( ) const`
Returns the number samples stored in the container.

getSampleRate() `float WaveformContainer::getSampleRate ( ) const`
Returns the sample rate.

operator=() `WaveformContainer& WaveformContainer::operator= ( const WaveformContainer & other )`
Assignment operator.
The documentation for this class was generated from the following file:

- WaveformContainerIf.h

4.6 WaveformContainer::Iterator Class Reference

Collaboration diagram for WaveformContainer::Iterator:

Public Member Functions

- Iterator (Impl *pImpl)
- Iterator (const Iterator &other)
- ~Iterator ()
- Iterator & operator= (const Iterator &other)
- bool next ()
- std::vector< Sample > getSamplesForAllChannels () const
- bool isValid () const

4.6.1 Detailed Description

Iterator class for accessing data of a WaveformContainer at different screen resolutions.

The Iterator internally stores a reference to the WaveformContainer, so the WaveformContainer must always be present during the lifetime of the Iterator instance.

4.6.2 Constructor & Destructor Documentation

Iterator() [1/2] `WaveformContainer::Iterator::Iterator ( Impl * pImpl )`

Iterator() [2/2] `WaveformContainer::Iterator::Iterator ( const Iterator & other )`

Copy constructor. Creates a deep copy of an Iterator object.

~Iterator() `WaveformContainer::Iterator::~Iterator ( )`

Destructor.

4.6.3 Member Function Documentation

getSamplesForAllChannels() `std::vector<Sample> WaveformContainer::Iterator::getSamplesForAllChannels ( ) const`

Returns the current frame, i.e. the sample for all channels at the current sample position.

isValid() `bool WaveformContainer::Iterator::isValid ( ) const`

next() `bool WaveformContainer::Iterator::next ( )`

Computes the next sample for each channel. Returns false if the last sample position is reached or if the Iterator is invalid, otherwise true.

operator=() `Iterator& WaveformContainer::Iterator::operator= ( const Iterator & other )`

Assignment operator.

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

- WaveformContainerIf.h

4.7 WaveformContainer::Sample Class Reference

Collaboration diagram for WaveformContainer::Sample:

Public Member Functions

- Sample (Impl *pImpl)
- Sample (const Sample &other)
- ~Sample ()
- Sample & operator= (const Sample &other)
- float getLowerAmplitudeBound () const
- float getUpperAmplitudeBound () const
- RGBColor getColor () const

4.7.1 Detailed Description

Data container for drawing the audio waveform at a single horizontal position.

4.7.2 Constructor & Destructor Documentation

Sample() [1/2] `WaveformContainer::Sample::Sample ( Impl * pImpl )`

Sample() [2/2] `WaveformContainer::Sample::Sample ( const Sample & other )`

Copy constructor.

~Sample() `WaveformContainer::Sample::~Sample ( )`

Destructor.

4.7.3 Member Function Documentation

getColor() `RGBColor WaveformContainer::Sample::getColor ( ) const`

Return the normalized color.

getLowerAmplitudeBound() `float WaveformContainer::Sample::getLowerAmplitude↩ Bound ( ) const`

Returns the lower amplitude bound of the waveform.

getUpperAmplitudeBound() `float WaveformContainer::Sample::getUpperAmplitude↩Bound ( ) const`

Returns the upper amplitude bound of the waveform.

operator=() `Sample& WaveformContainer::Sample::operator= ( const Sample & other )`

Assignment operator.

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

- WaveformContainerIf.h

5 File Documentation

5.1 ColoredWaveformIf.h File Reference

interface of the ColoredWaveformIf class.

This graph shows which files directly or indirectly include this file:

5.2 docugen.txt File Reference

5.2.1 Detailed Description

source documentation main file

5.3 FloatRange.h File Reference

interface of the FloatRange class.

This graph shows which files directly or indirectly include this file:

Classes

- class FloatRange

5.3.1 Detailed Description

:

5.4 RGBColor.h File Reference

interface of the RGBColor class.

Include dependency graph for RGBColor.h: This graph shows which files directly or indirectly include this file:

Classes

- class RGBColor

5.4.1 Detailed Description

:

5.5 SVGWaveformWriter.h File Reference

interface of the SVGWaveformWriter class.

Include dependency graph for SVGWaveformWriter.h:

Classes

- class SVGWaveformWriter

5.5.1 Detailed Description

:

5.6 WaveformContainerIf.h File Reference

interface of the WaveformContainer class.

Include dependency graph for WaveformContainerIf.h: This graph shows which files directly or indirectly include this file:

Classes

- class WaveformContainer
- class WaveformContainer::Sample
- class WaveformContainer::Iterator

5.6.1 Detailed Description

: