HOTSPOT SS18 Wie klingt ein Bild?

6

Erzeugt von Doxygen 1.8.14

Inhaltsverzeichnis

Kapitel 1

Verzeichnis der Namensbereiche

1.1 Liste aller Namensbereiche

Liste aller Namensbereiche mit Kurzbeschreibung:

Kapitel 2

Klassen-Verzeichnis

2.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

Kapitel 3

Datei-Verzeichnis

3.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

Kapitel 4

Dokumentation der Namensbereiche

4.1 bitmap-Namensbereichsreferenz

Verwaltungsinformationen zu Bitmap Dateien.

Klassen

- struct BitmapFileHeader

  Dateiheader fuer .bmp.
- struct BitmapInfoHeader

  Bitmapheader. Enthaelt Informationen ueber die Eigenschaften des Bildes.

4.1.1 Ausführliche Beschreibung

Verwaltungsinformationen zu Bitmap Dateien.

4.2 Processing-Namensbereichsreferenz

Beeinhaltet Datenstrukturen zur Umwandlung von Bildinformationen in Toene.

Klassen

- struct AdditionalSettings

  Zusaetzliche Einstellungen spezifisch fuer die verschienden Algorithmen.
- class Processing

  Klasse zur Umwandlung von Bilddaten in Tondaten.
- struct ProcessingSettings

  Grundeinstellungen fuer die Tonerzeugung.

Aufzählungen

- enum ProcessingAlgorithm {
  ProcessingAlgorithm::LR_SCAN = 0, ProcessingAlgorithm::LR_SCAN_NO_THRESHOLD, ProcessingAlgorithm::UD_SCAN,
  ProcessingAlgorithm::UD_SCAN_NO_THRESHOLD,
  ProcessingAlgorithm::TRIPLET, ProcessingAlgorithm::TRIPLET_JMP, ProcessingAlgorithm::UD_LR_SCAN
  }

  Verfuegbare Algorithmen.

4.2.1 Ausführliche Beschreibung

Beeinhaltet Datenstrukturen zur Umwandlung von Bildinformationen in Toene.

4.2.2 Dokumentation der Aufzählungstypen

4.2.2.1 ProcessingAlgorithm

```
enum Processing::ProcessingAlgorithm  [strong]
```

Verfuegbare Algorithmen.

Aufzählungswerte

LR_SCAN	
LR_SCAN_NO_THRESHOLD	
UD_SCAN	
UD_SCAN_NO_THRESHOLD	
TRIPLET	
TRIPLET_JMP	
UD_LR_SCAN	

Kapitel 5

Klassen-Dokumentation

5.1 Processing::AdditionalSettings Strukturreferenz

Zusaetzliche Einstellungen spezifisch fuer die verschienden Algorithmen.

`#include <Processing.h>`

Öffentliche Attribute

- uint64_t settings1 = 0

  Zusaetzliche Einstellung spezifisch fuer die verschiedenen Algorithmen
  LR_SCAN: Maximale Frequenz
  LR_SCAN_NO_THRESHOLD: Maximale Frequenz
  UD_SCAN: Maximale Frequenz
  UD_SCAN_NO_THRESHOLD: Maximale Frequenz
  TRIPLET: -
  TRIPLET_JMP: -
  UD_LR_SCAN: Maximale Frequenz.
- uint64_t settings2 = 0

  Zusaetzliche Einstellunge spezifisch fuer die verschiedenen Algorithmen
  LR_SCAN: Aktivierungsschwelle fuer Pixel
  LR_SCAN_NO_THRESHOLD: -
  UD_SCAN: Aktivierungsschwelle fuer Pixel
  UD_SCAN_NO_THRESHOLD: -
  TRIPLET: Tondauer abhaengig von dem Blauanteil (0-255)
  TRIPLET_JMP: Tondauer abhaengig von der Pixelintensitaet
  UD_LR_SCAN: Tondauer abhaengig vom Blauanteil.
- uint64_t settings3 = 0

  Zusaetzliche Einstellunge spezifisch fuer die verschiedenen Algorithmen
  LR_SCAN: -
  LR_SCAN_NO_THRESHOLD: -
  UD_SCAN: -
  UD_SCAN_NO_THRESHOLD: -
  TRIPLET: Anzahl der Triplets die man pro Frequenz ueberspringen soll
  TRIPLET_JMP: Anzahl der Zyklen durch die Bilddaten (z.B. 128)
  UD_LR_SCAN: -.
- uint64_t settings4 = 0

  Zusaetzliche Einstellunge spezifisch fuer die verschiedenen Algorithmen
  LR_SCAN: Pixel invertieren
  LR_SCAN_NO_THRESHOLD: Pixel invertieren
  UD_SCAN: Pixel invertieren
  UD_SCAN_NO_THRESHOLD: Pixel invertieren
  TRIPLET: Pixel invertieren
  TRIPLET_JMP: Pixel invertieren
  UD_LR_SCAN: Pixel invertieren.

5.1.1 Ausführliche Beschreibung

Zusaetzliche Einstellungen spezifisch fuer die verschienden Algorithmen.

5.1.2 Dokumentation der Datenelemente

5.1.2.1 settings1

```
uint64_t Processing::AdditionalSettings::settings1 = 0
```

Zusaetzliche Einstellung spezifisch fuer die verschiedenen Algorithmen
LR_SCAN: Maximale Frequenz
LR_SCAN_NO_THRESHOLD: Maximale Frequenz
UD_SCAN: Maximale Frequenz
UD_SCAN_NO_THRESHOLD: Maximale Frequenz
TRIPLET: -
TRIPLET_JMP: -
UD_LR_SCAN: Maximale Frequenz.

5.1.2.2 settings2

```
uint64_t Processing::AdditionalSettings::settings2 = 0
```

Zusaetzliche Einstellunge spezifisch fuer die verschiedenen Algorithmen
LR_SCAN: Aktivierungsschwelle fuer Pixel
LR_SCAN_NO_THRESHOLD: -
UD_SCAN: Aktivierungsschwelle fuer Pixel
UD_SCAN_NO_THRESHOLD: -
TRIPLET: Tondauer abhaengig von dem Blauanteil (0-255)
TRIPLET_JMP: Tondauer abhaengig von der Pixelintensitaet
UD_LR_SCAN: Tondauer abhaengig vom Blauanteil.

5.1.2.3 settings3

```
uint64_t Processing::AdditionalSettings::settings3 = 0
```

Zusaetzliche Einstellunge spezifisch fuer die verschiedenen Algorithmen
LR_SCAN: -
LR_SCAN_NO_THRESHOLD: -
UD_SCAN: -
UD_SCAN_NO_THRESHOLD: -
TRIPLET: Anzahl der Triplets die man pro Frequenz ueberspringen soll
TRIPLET_JMP: Anzahl der Zyklen durch die Bilddaten (z.B. 128)
UD_LR_SCAN: -.

5.1.2.4 settings4

```
uint64_t Processing::AdditionalSettings::settings4 = 0
```

Zusaetzliche Einstellunge spezifisch fuer die verschiedenen Algorithmen
LR_SCAN: Pixel invertieren
LR_SCAN_NO_THRESHOLD: Pixel invertieren
UD_SCAN: Pixel invertieren
UD_SCAN_NO_THRESHOLD: Pixel invertieren
TRIPLET: Pixel invertieren
TRIPLET_JMP: Pixel invertieren
UD_LR_SCAN: Pixel invertieren.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- Source/Processing.h

5.2 bitmap::BitmapFileHeader Strukturreferenz

Dateiheader fuer .bmp.

```
#include <Image.h>
```

Öffentliche Attribute

- uint16_t bfType

  ASCII Zeichen "BM" notwendig.
- uint32_t bfFileSize

  Groesse der .bmp Datei in bytes.
- uint16_t bfRes1

  Muss 0 sein.
- uint16_t bfRes2

  Muss 0 sein.
- uint32_t bfStartAdress

  Offset zu den Bilddaten von Beginn der Datei.

5.2.1 Ausführliche Beschreibung

Dateiheader fuer .bmp.

5.2.2 Dokumentation der Datenelemente

5.2.2.1 bfFileSize

```
uint32_t bitmap::BitmapFileHeader::bfFileSize
```

Groesse der .bmp Datei in bytes.

5.2.2.2 bfRes1

```
uint16_t bitmap::BitmapFileHeader::bfRes1
```

Muss 0 sein.

5.2.2.3 bfRes2

```
uint16_t bitmap::BitmapFileHeader::bfRes2
```

Muss 0 sein.

5.2.2.4 bfStartAdress

```
uint32_t bitmap::BitmapFileHeader::bfStartAdress
```

Offset zu den Bilddaten von Beginn der Datei.

5.2.2.5 bfType

```
uint16_t bitmap::BitmapFileHeader::bfType
```

ASCII Zeichen "BM" notwendig.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- Source/Image.h

5.3 bitmap::BitmapInfoHeader Strukturreferenz

Bitmapheader. Enthaelt Informationen ueber die Eigenschaften des Bildes.

```
#include <Image.h>
```

Öffentliche Attribute

- uint32_t biSize

  Groesse dieses Headers in Byte.
- uint32_t biWidth

  Breite des Bildes in Pixel.
- uint32_t biHeight

  Hoehe des Bildes in Pixel.
- uint16_t biColorPlanes

  In .bmp nicht verwendet.
- uint16_t biBitsPerPixel

  Farbtiefe in bpp (8)
- uint32_t biCompression

  Nicht beachtet.
- uint32_t biImageSize

  Groesse der Bilddaten.
- uint32_t biHorizontalResolution

  Nicht beachtet.
- uint32_t biVerticalResolution

  Nicht beachtet.
- uint32_t biColorsPerColorPalette

  Nicht beachtet.
- uint32_t biImportantColorsUsed

  Nicht beachtet.

5.3.1 Ausführliche Beschreibung

Bitmapheader. Enthaelt Informationen ueber die Eigenschaften des Bildes.

5.3.2 Dokumentation der Datenelemente

5.3.2.1 biBitsPerPixel

```
uint16_t bitmap::BitmapInfoHeader::biBitsPerPixel
```

Farbtiefe in bpp (8)

5.3.2.2 biColorPlanes

```
uint16_t bitmap::BitmapInfoHeader::biColorPlanes
```

In .bmp nicht verwendet.

5.3.2.3 biColorsPerColorPalette

```
uint32_t bitmap::BitmapInfoHeader::biColorsPerColorPalette
```

Nicht beachtet.

5.3.2.4 biCompression

```
uint32_t bitmap::BitmapInfoHeader::biCompression
```

Nicht beachtet.

5.3.2.5 biHeight

```
uint32_t bitmap::BitmapInfoHeader::biHeight
```

Hoehe des Bildes in Pixel.

5.3.2.6 biHorizontalResolution

```
uint32_t bitmap::BitmapInfoHeader::biHorizontalResolution
```

Nicht beachtet.

5.3.2.7 biImageSize

```
uint32_t bitmap::BitmapInfoHeader::biImageSize
```

Groesse der Bilddaten.

5.3.2.8 biImportantColorsUsed

```
uint32_t bitmap::BitmapInfoHeader::biImportantColorsUsed
```

Nicht beachtet.

5.3.2.9 biSize

```
uint32_t bitmap::BitmapInfoHeader::biSize
```

Groesse dieses Headers in Byte.

5.3.2.10 biVerticalResolution

```
uint32_t bitmap::BitmapInfoHeader::biVerticalResolution
```

Nicht beachtet.

5.3.2.11 biWidth

```
uint32_t bitmap::BitmapInfoHeader::biWidth
```

Breite des Bildes in Pixel.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- Source/Image.h

5.4 Image Klassenreferenz

Klasse zum oeffnen und auslesen von Bitmap Bilddateien.

```
#include <Image.h>
```

Öffentliche Methoden

- Image ()=delete
  Standardkonstruktor.
- Image (Image &&other)=default
  Standard Movekonstruktor.
- Image (std::string fileName)
  Konstruktor. Oeffnet .bmp Datei mit dem angegebenen Namen.
- Image & operator= (const Image &other)=delete
  Zuweisungen fuer Image nicht verfuegbar.
- Image & operator= (Image &&other)=delete
  Zuweisungen fuer Image nicht verfuegbar.
- const Pixel & getPixel (int y, int x) const
  Pixel an der Stelle (y,x) auslesen.
- uint32_t getHeight () const
  Hoehe des Bildes erhalten.
- uint32_t getWidth () const
  Breite des Bildes erhalten.
- ~Image ()

Private Methoden

- void readHeader ()

  Header der .bmp Datei auslesen, um Verwaltungsinformationen wie z.B. Hoehe und Breite zu erhalten.
- void readImageData ()

  Bilddaten aus der .bmp Datei auslesen und in diesem Objekt speichern.

Private Attribute

- bitmap::BitmapFileHeader bitmapFileHeader

  Informationen ueber die Datei (Datentyp,Bytes,Startadresse der Bilddaten..)
- bitmap::BitmapInfoHeader bitmapInfoHeader

  Informationen ueber das Bild (Breite, Hoehe, Farben..)
- std::vector< std::vector< Pixel > > bitmapData

  2D Vector mit den Pixeldaten des Bildes
- std::ifstream file

  Inputfilestream fuer die zu oeffnende Bilddatei.

5.4.1 Ausführliche Beschreibung

Klasse zum oeffnen und auslesen von Bitmap Bilddateien.

5.4.2 Beschreibung der Konstruktoren und Destruktoren

5.4.2.1 Image() [1/3]

```
Image::Image ( )  [delete]
```

Standardkonstruktor.

5.4.2.2 Image() [2/3]

```
Image::Image (
            Image && other )  [default]
```

Standard Movekonstruktor.

Parameter

other	Image rvalue

5.4.2.3 Image() [3/3]

```
Image::Image (
            std::string fileName )
```

Konstruktor. Oeffnet .bmp Datei mit dem angegebenen Namen.

Parameter

fileName	Name der .bmp Datei als String

5.4.2.4 ~Image()

```
Image::~Image ( )
```

5.4.3 Dokumentation der Elementfunktionen

5.4.3.1 getHeight()

```
uint32_t Image::getHeight ( ) const  [inline]
```

Hoehe des Bildes erhalten.

Rückgabe

Hoehe als uint32_t

5.4.3.2 getPixel()

```
const Pixel& Image::getPixel (
            int y,
            int x ) const  [inline]
```

Pixel an der Stelle (y,x) auslesen.

Parameter

y	Koordinate Y
x	Koordinate X

Rückgabe

Referenz auf ein konstanten Pixel der sich an Stelle (y,x) befindet.

5.4.3.3 getWidth()

```
uint32_t Image::getWidth ( ) const  [inline]
```

Breite des Bildes erhalten.

Rückgabe

Breite als uint32_t

5.4.3.4 operator=() [1/2]

```
Image& Image::operator= (
            const Image & other )  [delete]
```

Zuweisungen fuer Image nicht verfuegbar.

5.4.3.5 operator=() [2/2]

```
Image& Image::operator= (
            Image && other )  [delete]
```

Zuweisungen fuer Image nicht verfuegbar.

5.4.3.6 readHeader()

```
void Image::readHeader ( )  [private]
```

Header der .bmp Datei auslesen, um Verwaltungsinformationen wie z.B. Hoehe und Breite zu erhalten.

5.4.3.7 readImageData()

```
void Image::readImageData ( )  [private]
```

Bilddaten aus der .bmp Datei auslesen und in diesem Objekt speichern.

5.4.4 Dokumentation der Datenelemente

5.4.4.1 bitmapData

```
std::vector<std::vector<Pixel> > Image::bitmapData  [private]
```

2D Vector mit den Pixeldaten des Bildes

5.4.4.2 bitmapFileHeader

```
bitmap::BitmapFileHeader Image::bitmapFileHeader  [private]
```

Informationen ueber die Datei (Datentyp,Bytes,Startadresse der Bilddaten..)

5.4.4.3 bitmapInfoHeader

```
bitmap::BitmapInfoHeader Image::bitmapInfoHeader  [private]
```

Informationen ueber das Bild (Breite, Hoehe, Farben..)

5.4.4.4 file

```
std::ifstream Image::file  [private]
```

Inputfilestream fuer die zu oeffnende Bilddatei.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Source/Image.h
- Source/Image.cpp

5.5 Pixel Strukturreferenz

Pixel 8 Bit RGB Werte.

```
#include <Image.h>
```

Öffentliche Attribute

- uint8_t r
  Rotanteil.
- uint8_t g
  Gruenanteil.
- uint8_t b
  Blauanteil.

5.5.1 Ausführliche Beschreibung

Pixel 8 Bit RGB Werte.

5.5.2 Dokumentation der Datenelemente

5.5.2.1 b

```
uint8_t Pixel::b
```

Blauanteil.

5.5.2.2 g

```
uint8_t Pixel::g
```

Gruenanteil.

5.5.2.3 r

```
uint8_t Pixel::r
```

Rotanteil.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- Source/Image.h

5.6 Processing::Processing Klassenreferenz

Klasse zur Umwandlung von Bilddaten in Tondaten.

`#include <Processing.h>`

Öffentliche Methoden

- void start (const std::string &imageFile, const std::string &outputFile, ProcessingAlgorithm algorithm, const ProcessingSettings &settings, const AdditionalSettings &additionalSettings)

  Starte die Umwandlung eines Bildes in eine Audiodatei, anhand eines vorgegebenen Algorithmus.
- Processing ()=default

  Standardkonstruktor.
- Processing (const Processing &other)=delete

  Kopieren nicht moeglich.
- Processing & operator= (const Processing &other)=delete

  Zuweisung nicht moeglich.
- Processing & operator= (Processing &&other)=delete

  Movezuweisung nicht moeglich.

Private Methoden

- void lrScan (const AdditionalSettings &additionalSettings)

  Bild wird von Links nach Rechts spaltenweise durchgescannt. Jede Zeile steht fuer eine Frequenz. Je intensiver RGB vom Pixel desto hoeher die Amplitude der jeweiligen Frequenz. Pixel werden nur betrachtet, wenn der Durschnitt von RGB > Aktivierungsschwelle ist.
- void lrScan_no_threshold (const AdditionalSettings &additionalSettings)

  Bild wird von Links nach Rechts spaltenweise durchgescannt. Jede Zeile steht fuer eine Frequenz. Je intensiver RGB vom Pixel desto hoeher die Amplitude der jeweiligen Frequenz. Keine Aktivierungsgrenze.
- void udScan (const AdditionalSettings &additionalSettings)

  Bild wird von Oben nach Unten zweilenweise durchgescannt. Jede Spalte steht fuer eine Frequenz. Je intensiver RGB vom Pixel desto hoeher die Amplitude der jeweiligen Frequenz. Pixel werden nur betrachtet, wenn der Durschnitt von RGB > Aktivierungsschwelle ist.
- void udScan_no_threshold (const AdditionalSettings &additionalSettings)

  Bild wird von Oben nach Unten zweilenweise durchgescannt. Jede Spalte steht fuer eine Frequenz. Je intensiver RGB vom Pixel desto hoeher die Amplitude der jeweiligen Frequenz. Keine Aktivierungsschwelle.
- void triplet (const AdditionalSettings &additionalSettings)

  Bild wird immer um 3 Byte (Triplet) oder Vielfache durchgesprungen. Jede Farbe hat eine Eigenschaft bei der Tonerzeugung Rot: Lautstaerke ; Gruen: Tonhoehe; Blau: Tondauer.
- void triplet_jmp (const AdditionalSettings &additionalSettings)

  Bild wird mit einem Offset abhaengig von der Farbe durchgesprungen Rot: Lautstarke, Gruen: Tonhoehe; Blau: keine Funktion.
- void ud_lr_scan (const AdditionalSettings &additionalSettings)

  Das Bild wird von oben nach unten und von links nach rechts gleichzeitig gescannt. Die Senkrechte zur X-Achse und die Parallele erzeugen beide jeweils einen Ton.
  Jeder Ton wird abhaengig von jeweiligen Farbanteil erzeugt.
  Rot bestimmt die Frequenz, Gruen bestimmt die Lautstaerke und der gemeinsame Blauanteil die Dauer der beiden Frequenzen.

Private Attribute

- std::unique_ptr< Image > image
- std::unique_ptr< Sound > sound
- std::unique_ptr< Wave > wave

5.6.1 Ausführliche Beschreibung

Klasse zur Umwandlung von Bilddaten in Tondaten.

5.6.2 Beschreibung der Konstruktoren und Destruktoren

5.6.2.1 Processing() [1/2]

```
Processing::Processing::Processing ( )  [default]
```

Standardkonstruktor.

5.6.2.2 Processing() [2/2]

```
Processing::Processing::Processing (
            const Processing & other )  [delete]
```

Kopieren nicht moeglich.

5.6.3 Dokumentation der Elementfunktionen

5.6.3.1 lrScan()

```
void Processing::Processing::lrScan (
            const AdditionalSettings & additionalSettings )  [private]
```

Bild wird von Links nach Rechts spaltenweise durchgescannt. Jede Zeile steht fuer eine Frequenz. Je intensiver RGB vom Pixel desto hoeher die Amplitude der jeweiligen Frequenz. Pixel werden nur betrachtet, wenn der Durschnitt von RGB > Aktivierungsschwelle ist.

Parameter

additionalSettings	

5.6.3.2 lrScan_no_threshold()

```
void Processing::Processing::lrScan_no_threshold (
            const AdditionalSettings & additionalSettings )  [private]
```

Bild wird von Links nach Rechts spaltenweise durchgescannt. Jede Zeile steht fuer eine Frequenz. Je intensiver RGB vom Pixel desto hoeher die Amplitude der jeweiligen Frequenz. Keine Aktivierungsgrenze.

Parameter

additionalSettings	

5.6.3.3 operator=() [1/2]

```
Processing& Processing::Processing::operator= (
            const Processing & other )  [delete]
```

Zuweisung nicht moeglich.

5.6.3.4 operator=() [2/2]

```
Processing& Processing::Processing::operator= (
            Processing && other )  [delete]
```

Movezuweisung nicht moeglich.

5.6.3.5 start()

```
void Processing::Processing::start (
            const std::string & imageFile,
            const std::string & outputFile,
            ProcessingAlgorithm algorithm,
            const ProcessingSettings & settings,
            const AdditionalSettings & additionalSettings )
```

Starte die Umwandlung eines Bildes in eine Audiodatei, anhand eines vorgegebenen Algorithmus.

Parameter

imageFile	Dateiname vom Bild
outputFile	Dateiname der zu erstellenden Audiodatei
algorithm	Algorithmus der fuer die Umwandlung genutzt wird
settings	Allgemeine Einstellungen fuer die Tonerzeugung
additionalSettings	Spezifische Einstellungen fuer den jeweiligen Algorithmus

5.6.3.6 triplet()

```
void Processing::Processing::triplet (
            const AdditionalSettings & additionalSettings )  [private]
```

Bild wird immer um 3 Byte (Triplet) oder Vielfache durchgesprungen. Jede Farbe hat eine Eigenschaft bei der Tonerzeugung Rot: Lautstaerke ; Gruen: Tonhoehe; Blau: Tondauer.

Parameter

additionalSettings	

5.6.3.7 triplet_jmp()

```
void Processing::Processing::triplet_jmp (
            const AdditionalSettings & additionalSettings )  [private]
```

Bild wird mit einem Offset abhaengig von der Farbe durchgesprungen Rot: Lautstarke, Gruen: Tonhoehe; Blau: keine Funktion.

Parameter

additionalSettings	

5.6.3.8 ud_lr_scan()

```
void Processing::Processing::ud_lr_scan (
            const AdditionalSettings & additionalSettings )  [private]
```

Das Bild wird von oben nach unten und von links nach rechts gleichzeitig gescannt. Die Senkrechte zur X-Achse und die Parallele erzeugen beide jeweils einen Ton.
Jeder Ton wird abhaengig von jeweiligen Farbanteil erzeugt.
Rot bestimmt die Frequenz, Gruen bestimmt die Lautstaerke und der gemeinsame Blauanteil die Dauer der beiden Frequenzen.

Parameter

additionalSettings	

5.6.3.9 udScan()

```
void Processing::Processing::udScan (
            const AdditionalSettings & additionalSettings )  [private]
```

Bild wird von Oben nach Unten zweilenweise durchgescannt. Jede Spalte steht fuer eine Frequenz. Je intensiver RGB vom Pixel desto hoeher die Amplitude der jeweiligen Frequenz. Pixel werden nur betrachtet, wenn der Durschnitt von RGB > Aktivierungsschwelle ist.

Parameter

additionalSettings	

5.6.3.10 udScan_no_threshold()

```
void Processing::Processing::udScan_no_threshold (
            const AdditionalSettings & additionalSettings )  [private]
```

Bild wird von Oben nach Unten zweilenweise durchgescannt. Jede Spalte steht fuer eine Frequenz. Je intensiver RGB vom Pixel desto hoeher die Amplitude der jeweiligen Frequenz. Keine Aktivierungsschwelle.

Parameter

additionalSettings	

5.6.4 Dokumentation der Datenelemente

5.6.4.1 image

```
std::unique_ptr<Image> Processing::Processing::image  [private]
```

5.6.4.2 sound

```
std::unique_ptr<Sound> Processing::Processing::sound  [private]
```

5.6.4.3 wave

```
std::unique_ptr<Wave> Processing::Processing::wave  [private]
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Source/Processing.h
- Source/Processing.cpp

5.7 Processing::ProcessingSettings Strukturreferenz

Grundeinstellungen fuer die Tonerzeugung.

```
#include <Processing.h>
```

Öffentliche Attribute

- double audioVolume = 50
  Grundlautstaerke.
- double samplingFrequency = 44100
  Abtastrate.
- size_t samples = 128
  Anzahl der Samples pro Frequenzgenerierung, erhoeht die Tondauer&Rechenaufwand.

5.7.1 Ausführliche Beschreibung

Grundeinstellungen fuer die Tonerzeugung.

5.7.2 Dokumentation der Datenelemente

5.7.2.1 audioVolume

```
double Processing::ProcessingSettings::audioVolume = 50
```

Grundlautstaerke.

5.7.2.2 samples

```
size_t Processing::ProcessingSettings::samples = 128
```

Anzahl der Samples pro Frequenzgenerierung, erhoeht die Tondauer&Rechenaufwand.

5.7.2.3 samplingFrequency

```
double Processing::ProcessingSettings::samplingFrequency = 44100
```

Abtastrate.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- Source/Processing.h

5.8 Sound Klassenreferenz

Klasse zur Erzeugung von Toenen.

```
#include <Sound.h>
```

Öffentliche Methoden

- Sound ()
  Standardkonstruktor.
- Sound (double samplingFrequency, double audioVolume, size_t bufferSize)
  Konstruktor.
- void addFrequency (double volume, double frequency)
  Einen Ton mit einer bestimmter Frequenz&Amplitude aufaddieren.
- const std::vector< double > & getBuffer () const
  Den zuvor beschriebenen Puffer auslesen.
- void resetBuffer ()
  Puffer loeschen/zuruecksetzen.
- ~Sound ()=default

Private Attribute

- std::vector< double > buffer
  Puffer fuer die Audiodaten.
- const double samplingFrequency { DEFAULT_SAMPLINGRATE }
  Abtastfrequenz welche zur Tonerzeugung genutzt wird.
- const double audioVolume { DEFAULT_AUDIOVOLUME }
  Grundlautstaerke.
- double lastPos { 0 }
  Letzte Position der Sinuswelle bei verlassen der Funktion addFrequency.
- const size_t bufferSize { DEFAULT_BUFFERSIZE }
  Groesse des Puffers.

5.8.1 Ausführliche Beschreibung

Klasse zur Erzeugung von Toenen.

5.8.2 Beschreibung der Konstruktoren und Destruktoren

5.8.2.1 Sound() [1/2]

```
Sound::Sound ( )
```

Standardkonstruktor.

5.8.2.2 Sound() [2/2]

```
Sound::Sound (
            double samplingFrequency,
            double audioVolume,
            size_t bufferSize )
```

Konstruktor.

Parameter

samplingFrequency	Abtastrate fuer Tonerzeugung
audioVolume	Grundlautstaerke
bufferSize	Puffergroesse. Je groesser desto groesser/laenger ein Sample und Rechenaufwand

5.8.2.3 ~Sound()

```
Sound::~Sound ( )  [default]
```

5.8.3 Dokumentation der Elementfunktionen

5.8.3.1 addFrequency()

```
void Sound::addFrequency (
            double volume,
            double frequency )
```

Einen Ton mit einer bestimmter Frequenz&Amplitude aufaddieren.

Parameter

volume	Amplitude (Lautstarke) der Frequenz
frequency	Frequenz in Hz

5.8.3.2 getBuffer()

```
const std::vector< double > & Sound::getBuffer ( ) const
```

Den zuvor beschriebenen Puffer auslesen.

Rückgabe

Konstante Referenz auf den Puffer

5.8.3.3 resetBuffer()

```
void Sound::resetBuffer ( )
```

Puffer loeschen/zuruecksetzen.

5.8.4 Dokumentation der Datenelemente

5.8.4.1 audioVolume

```
const double Sound::audioVolume { DEFAULT_AUDIOVOLUME }  [private]
```

Grundlautstaerke.

5.8.4.2 buffer

```
std::vector<double> Sound::buffer  [private]
```

Puffer fuer die Audiodaten.

5.8.4.3 bufferSize

```
const size_t Sound::bufferSize { DEFAULT_BUFFERSIZE }  [private]
```

Groesse des Puffers.

5.8.4.4 lastPos

```
double Sound::lastPos { 0 }  [private]
```

Letzte Position der Sinuswelle bei verlassen der Funktion addFrequency.

5.8.4.5 samplingFrequency

```
const double Sound::samplingFrequency { DEFAULT_SAMPLINGRATE }  [private]
```

Abtastfrequenz welche zur Tonerzeugung genutzt wird.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Source/Sound.h
- Source/Sound.cpp

5.9 Wave Klassenreferenz

Klasse zur Erzeugung von .wav Audiodateien.

```
#include <Wave.h>
```

Öffentliche Methoden

- Wave (std::string fileName, uint32_t sampleRate)
  Konstruktor.
- Wave ()
  Standardkonstruktor.
- template<typename T >
  void writeSamples (T data)
  Schreibt Daten in Audiodatei.
- void finishHeader ()
  Header fertig machen. Erst nutzen wenn Audiodateien fertig beschrieben wurde.
- void closeFile ()
  Audiodatei schliesen.
- ~Wave ()

Private Methoden

- void prepareHeader ()

  Bereitet Header fur Audiodatei vor.

Private Attribute

- std::ofstream file {"WieKlingtEinBild.wav", std::ios::binary}

  Outputfilestream. Genutzt um Audiodaten in Datei zu schreiben.
- struct {
  const uint32_t subChunk1Size {16}
  Keine zusaetzlichen Daten.
  const uint16_t audioFormat {1}
  Keine Kompression.
  const uint16_t numChannels {1}
  Anzahl der Channel (hier: 1)
  uint32_t sampleRate {44100}
  Abtastrate.
  const uint32_t byteRate {static_cast<uint32_t>(sampleRate * numChannels * DEFAULT_BYTES_PER_SAMPLE)}
  Abtastrate Frame Groesse.*
  const uint16_t blockAlign {static_cast<uint16_t>(numChannels * DEFAULT_BYTES_PER_SAMPLE)}
  *Allignment. numChannels * DEFAULT_BYTES_PER_SAMPLE.*
  const uint16_t bitsPerSample {DEFAULT_BITS_PER_SAMPLE}
  Bits pro Samplewert pro Kanal.
  } waveHeader

  Header der .wav Datei.

5.9.1 Ausführliche Beschreibung

Klasse zur Erzeugung von .wav Audiodateien.

5.9.2 Beschreibung der Konstruktoren und Destruktoren

5.9.2.1 Wave() [1/2]

```
Wave::Wave (
            std::string fileName,
            uint32_t sampleRate )
```

Konstruktor.

Parameter

fileName	Dateiname der zu erzeugenden Audiodatei (.wav)
sampleRate	Abtastrate der Audiodatei

5.9.2.2 Wave() [2/2]

```
Wave::Wave ( )
```

Standardkonstruktor.

5.9.2.3 ∼Wave()

```
Wave::∼Wave ( )
```

5.9.3 Dokumentation der Elementfunktionen

5.9.3.1 closeFile()

```
void Wave::closeFile ( )
```

Audiodatei schliesen.

5.9.3.2 finishHeader()

```
void Wave::finishHeader ( )
```

Header fertig machen. Erst nutzen wenn Audiodateien fertig beschrieben wurde.

5.9.3.3 prepareHeader()

```
void Wave::prepareHeader ( )  [private]
```

Bereitet Header fur Audiodatei vor.

5.9.3.4 writeSamples()

```
template<typename T >
void Wave::writeSamples (
            T data )  [inline]
```

Schreibt Daten in Audiodatei.

Template Parameters

T	Typ von Daten z.B. uint16_t

Parameter

data	Daten die geschrieben werden sollen

5.9.4 Dokumentation der Datenelemente

5.9.4.1 audioFormat

```
const uint16_t Wave::audioFormat {1}
```

Keine Kompression.

5.9.4.2 bitsPerSample

```
const uint16_t Wave::bitsPerSample {DEFAULT_BITS_PER_SAMPLE}
```

Bits pro Samplewert pro Kanal.

5.9.4.3 blockAlign

```
const uint16_t Wave::blockAlign {static_cast<uint16_t>(numChannels * DEFAULT_BYTES_PER_SAMP↩
LE)}
```

Allignment. numChannels ∗ DEFAULT_BYTES_PER_SAMPLE.

5.9.4.4 byteRate

```
const uint32_t Wave::byteRate {static_cast<uint32_t>(sampleRate * numChannels * DEFAULT_BYTE↩
S_PER_SAMPLE)}
```

Abtastrate∗ Frame Groesse.

5.9.4.5 file

```
std::ofstream Wave::file {"WieKlingtEinBild.wav", std::ios::binary}  [private]
```

Outputfilestream. Genutzt um Audiodaten in Datei zu schreiben.

5.9.4.6 numChannels

```
const uint16_t Wave::numChannels {1}
```

Anzahl der Channel (hier: 1)

5.9.4.7 sampleRate

```
uint32_t Wave::sampleRate {44100}
```

Abtastrate.

5.9.4.8 subChunk1Size

```
const uint32_t Wave::subChunk1Size {16}
```

Keine zusaetzlichen Daten.

5.9.4.9 waveHeader

```
struct { ... } Wave::waveHeader  [private]
```

Header der .wav Datei.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- Source/Wave.h
- Source/Wave.cpp

Kapitel 6

Datei-Dokumentation

6.1 Source/Image.cpp-Dateireferenz

```
#include "Image.h"
#include <iostream>
```

6.2 Source/Image.h-Dateireferenz

```
#include <fstream>
#include <string>
#include <vector>
```

Klassen

- struct bitmap::BitmapFileHeader

  Dateiheader fuer .bmp.
- struct bitmap::BitmapInfoHeader

  Bitmapheader. Enthaelt Informationen ueber die Eigenschaften des Bildes.
- struct Pixel

  Pixel 8 Bit RGB Werte.
- class Image

  Klasse zum oeffnen und auslesen von Bitmap Bilddateien.

Namensbereiche

- bitmap

  Verwaltungsinformationen zu Bitmap Dateien.

6.3 Source/Processing.cpp-Dateireferenz

```
#include "Processing.h"
#include <iostream>
```

Makrodefinitionen

- #define PRINT_PROGRESS

Funktionen

- void printProgress (int x, int xMax)

6.3.1 Makro-Dokumentation

6.3.1.1 PRINT_PROGRESS

```
#define PRINT_PROGRESS
```

6.3.2 Dokumentation der Funktionen

6.3.2.1 printProgress()

```
void printProgress (
            int x,
            int xMax )
```

6.4 Source/Processing.h-Dateireferenz

```
#include "Image.h"
#include "Sound.h"
#include "Wave.h"
#include <memory>
```

Klassen

- struct Processing::ProcessingSettings

  Grundeinstellungen fuer die Tonerzeugung.
- struct Processing::AdditionalSettings

  Zusaetzliche Einstellungen spezifisch fuer die verschienden Algorithmen.
- class Processing::Processing

  Klasse zur Umwandlung von Bilddaten in Tondaten.

Namensbereiche

- Processing

  Beeinhaltet Datenstrukturen zur Umwandlung von Bildinformationen in Toene.

Aufzählungen

- enum Processing::ProcessingAlgorithm {
  Processing::ProcessingAlgorithm::LR_SCAN = 0, Processing::ProcessingAlgorithm::LR_SCAN_NO_THRESHOLD,
  Processing::ProcessingAlgorithm::UD_SCAN, Processing::ProcessingAlgorithm::UD_SCAN_NO_THRESHOLD,
  Processing::ProcessingAlgorithm::TRIPLET, Processing::ProcessingAlgorithm::TRIPLET_JMP, Processing::ProcessingAlgorith
  }

  Verfuegbare Algorithmen.

6.5 Source/Quelle.cpp-Dateireferenz

```
#include <iostream>
#include "Processing.h"
```

Funktionen

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

6.5.1 Dokumentation der Funktionen

6.5.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

6.6 Source/Sound.cpp-Dateireferenz

```
#include "Sound.h"
#include <cmath>
```

Makrodefinitionen

- #define _USE_MATH_DEFINES

6.6.1 Makro-Dokumentation

6.6.1.1 _USE_MATH_DEFINES

```
#define _USE_MATH_DEFINES
```

6.7 Source/Sound.h-Dateireferenz

```
#include <vector>
```

Klassen

- class Sound

  Klasse zur Erzeugung von Toenen.

6.8 Source/Wave.cpp-Dateireferenz

```
#include "Wave.h"
```

6.9 Source/Wave.h-Dateireferenz

```
#include <fstream>
```

Klassen

- class Wave

  Klasse zur Erzeugung von .wav Audiodateien.

Index