Chow-Chow

0.1

Generated by Doxygen 1.8.20

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 ChowChow::Amp Class Reference

Public Types

- using **amp_t** = __int128
- using **amp_ut** = unsigned __int128
- using **amp_down_t** = int64_t

Public Member Functions

- constexpr **Amp** (amp_t n)
- constexpr **Amp** (long long n)
- constexpr **Amp** (long n)
- constexpr **Amp** (int n)
- constexpr **Amp** (short n)
- constexpr **Amp** (unsigned long long n)
- constexpr **Amp** (unsigned long n)
- constexpr **Amp** (unsigned int n)
- constexpr **Amp** (unsigned short n)
- constexpr **Amp** (long double n)
- constexpr **Amp** (double n)
- constexpr auto **amp** () const
- constexpr void **amp** (const Amp &m)
- std::string **to_s** () const
- constexpr Amp & **operator+=** (const Amp &m)
- constexpr Amp & **operator-=** (const Amp &m)
- constexpr Amp & **operator∗=** (const Amp &m)

Static Public Attributes

- static const int **AMP_SIZE** = sizeof(amp_t)∗CHAR_BIT
- static const amp_t **AMP_MAX**
- static const amp_t **AMP_MIN** = -AMP_MAX - 1
- static const amp_down_t **AMP_DOWN_MAX**
- static constexpr amp_down_t **AMP_DOWN_MAX_P10**
- static constexpr amp_t **SCALE_DOWN**
- static constexpr int **DECIMALS** = 18
- static constexpr amp_t **PI** = 3141592653589793238

2.1.1 Detailed Description

Definition at line 10 of file amp.hpp.

2.1.2 Member Data Documentation

2.1.2.1 AMP_DOWN_MAX

```
const amp_down_t ChowChow::Amp::AMP_DOWN_MAX  [static]
```

Initial value:

```
=
            static_cast<amp_down_t>(static_cast<amp_ut>(AMP_MAX) » AMP_SIZE/2)
```

Definition at line 22 of file amp.hpp.

2.1.2.2 AMP_DOWN_MAX_P10

```
constexpr amp_down_t ChowChow::Amp::AMP_DOWN_MAX_P10  [static], [constexpr]
```

Initial value:

```
= []{
            amp_down_t max = AMP_DOWN_MAX/10;
            amp_down_t tens = 1;
            while (max) {
                max /= 10;
                tens *= 10;
            }
            return tens;
        }()
```

Definition at line 24 of file amp.hpp.

2.1.2.3 AMP_MAX

```
const amp_t ChowChow::Amp::AMP_MAX  [static]
```

Initial value:

```
=
            static_cast<amp_t>((static_cast<amp_ut>(1) « (AMP_SIZE - 1)) - 1)
```

Definition at line 19 of file amp.hpp.

2.1.2.4 SCALE_DOWN

```
constexpr amp_t ChowChow::Amp::SCALE_DOWN  [static], [constexpr]
```

Initial value:

```
=
            static_cast<amp_t>(AMP_DOWN_MAX_P10)
```

Definition at line 36 of file amp.hpp.

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

- /home/zoe/ganymede/code/cpp/chow-chow/include/chow-chow/amp.hpp

2.2 ChowChow::Frequency Class Reference

Public Types

- using **freq_t** = uint16_t

Public Member Functions

- **Frequency** (freq_t integ, freq_t frct)
- **Frequency** (double f)
- constexpr freq_t **intg** () const
- constexpr freq_t **frac** () const
- double **freq** () const
- double **frac_f** () const
- void **intg** (freq_t integ)
- void **frac** (freq_t frct)
- void **freq** (double f)

Static Public Attributes

- static constexpr freq_t **FREQ_MAX** = UINT16_MAX
- static constexpr freq_t **FREQ_MIN** = 0

2.2.1 Detailed Description

Definition at line 7 of file frequency.hpp.

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

- /home/zoe/ganymede/code/cpp/chow-chow/include/chow-chow/frequency.hpp
- /home/zoe/ganymede/code/cpp/chow-chow/src/frequency.cpp

2.3 ChowChow::Operator Class Reference

An FM operator.

`#include <operator.hpp>`

Public Member Functions

- double wave (double time) const

  The current state of the wave, taking into account the ratio, offset, and vibrato.
- double vibrato (double time) const

  The current state of the vibrato wave.
- double sig (double time, double mod=0.) const

  The current state of the output signal.
- void freq (double rate)

  Sets the oscillation frequency.
- void freq_offset (double factor)

  Adjusts the frequency by a small offset.
- void vibrato_freq (double rate)

  The rate of vibrato.
- void vibrato_amp (double amp)

  The strength of vibrato.
- void ratio (double n)

  The modulation ratio.
- void index (double n)

  The modulation index.

2.3.1 Detailed Description

An FM operator.

An FM operator (technically a phase modulation operator). Generates a sinusoid by itself, but can also be modulated by another signal.

Definition at line 12 of file operator.hpp.

2.3.2 Member Function Documentation

2.3.2.1 freq()

```
void Operator::freq (
            double rate )
```

Sets the oscillation frequency.

Parameters

rate	in Hz.

Definition at line 29 of file operator.cpp.

2.3.2.2 freq_offset()

```
void Operator::freq_offset (
            double factor )
```

Adjusts the frequency by a small offset.

This can be useful for chorus-like effects.

Parameters

factor	in the range of -1.0–1.0 or so. 0 is neutral.

Definition at line 41 of file operator.cpp.

2.3.2.3 index()

```
void Operator::index (
            double n )
```

The modulation index.

A coefficient of the output signal. Increases the strength and quantity of partials in the carrier if this operator is used as a modulator.

Parameters

factor	ordinarily in the range of 0.25–20. Higher values may cause aliasing if the operator isn't oversampled sufficiently.

Definition at line 62 of file operator.cpp.

2.3.2.4 ratio()

```
void Operator::ratio (
            double n )
```

The modulation ratio.

A coefficient of the frequency. Increases the spacing of the partials in the carrier if this operator is used as a modulator.

Parameters

factor	ordinarily in the range of 0.25–20. Higher values may cause aliasing if the operator isn't oversampled sufficiently.

Definition at line 46 of file operator.cpp.

2.3.2.5 sig()

```
double Operator::sig (
            double time,
            double mod = 0.  )  const
```

The current state of the output signal.

Parameters

time	in seconds.
mod	optionally, the modulating signal.

Definition at line 24 of file operator.cpp.

2.3.2.6 vibrato()

```
double Operator::vibrato (
            double time ) const
```

The current state of the vibrato wave.

Parameters

time	in seconds.

Definition at line 15 of file operator.cpp.

2.3.2.7 vibrato_amp()

```
void Operator::vibrato_amp (
            double amp )
```

The strength of vibrato.

Parameters

amp	in the range of 1.0–0.1 or so. 0 turns the vibrato off.

Definition at line 57 of file operator.cpp.

2.3.2.8 vibrato_freq()

```
void Operator::vibrato_freq (
            double rate )
```

The rate of vibrato.

Parameters

rate	in Hz; a range of 0.1–19 or so.

Definition at line 48 of file operator.cpp.

2.3.2.9 wave()

```
double Operator::wave (
            double time ) const
```

The current state of the wave, taking into account the ratio, offset, and vibrato.

Parameters

time	in seconds.

Definition at line 8 of file operator.cpp.

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

- /home/zoe/ganymede/code/cpp/chow-chow/include/chow-chow/operator.hpp
- /home/zoe/ganymede/code/cpp/chow-chow/src/operator.cpp

2.4 ChowChow::Operators< N > Class Template Reference

Public Member Functions

- Operator & operator[ ] (std::size_t pos)

Subscripting.

- double connection (std::size_t from, std::size_t to) const

  Get the strength of the connection between two operators.
- bool is_output (std::size_t op) const

  Check if an operator will be included in the output signal.
- std::vector< std::size_t > & order ()

  Get the order the operators will be measured in.
- double check_sig (std::size_t op) const

  Check the current output signal of an operator.
- void connect (std::size_t from, std::size_t to, double amp=1.)

  Connect a modulator to a carrier.
- void output (std::size_t op, bool connect=true)

  Include or exclude an operator from the output signal.
- void reorder ()

  Recompute the unravelled order the operators will be measured in.
- double sig (double time)

  Compute the output signal.

2.4.1 Detailed Description

template<std::size_t N>
class ChowChow::Operators< N >

Definition at line 12 of file operators.hpp.

2.4.2 Member Function Documentation

2.4.2.1 check_sig()

```
template<std::size_t N>
double ChowChow::Operators< N >::check_sig (
            std::size_t op ) const  [inline]
```

Check the current output signal of an operator.

Parameters

op	1-indexed.

Definition at line 62 of file operators.hpp.

2.4.2.2 connect()

```
template<std::size_t N>
```

```
void ChowChow::Operators< N >::connect (
            std::size_t from,
            std::size_t to,
            double amp = 1.  )  [inline]
```

Connect a modulator to a carrier.

Connect the output of one of the operators to the input of another. If the number of the carrier is equal to or higher than the modulator, a feedback loop is implied.

Parameters

from	the number of the operator to use as a modulator, 1-indexed.
to	the number of the operator to use as a carrier, 1-indexed.
amp	optionally, a factor to scale the modulator signal by.

Definition at line 84 of file operators.hpp.

2.4.2.3 connection()

```
template<std::size_t N>
double ChowChow::Operators< N >::connection (
            std::size_t from,
            std::size_t to ) const  [inline]
```

Get the strength of the connection between two operators.

Parameters

from	the number of the modulator, 1-indexed.
to	the number of the carrier, 1-indexed.

Definition at line 32 of file operators.hpp.

2.4.2.4 is_output()

```
template<std::size_t N>
bool ChowChow::Operators< N >::is_output (
            std::size_t op ) const  [inline]
```

Check if an operator will be included in the output signal.

Parameters

op	1-indexed.

Definition at line 43 of file operators.hpp.

2.4.2.5 operator[]()

```
template<std::size_t N>
Operator& ChowChow::Operators< N >::operator[] (
            std::size_t pos )  [inline]
```

Subscripting.

Parameters

pos	1-indexed (the lowest operator is 1), in deference to how FM "algorithms" are routinely diagrammed.

Definition at line 21 of file operators.hpp.

2.4.2.6 output()

```
template<std::size_t N>
void ChowChow::Operators< N >::output (
            std::size_t op,
            bool connect = true )  [inline]
```

Include or exclude an operator from the output signal.

Parameters

op	the number of the operator, 1-indexed.

Definition at line 94 of file operators.hpp.

2.4.2.7 sig()

```
template<std::size_t N>
double ChowChow::Operators< N >::sig (
            double time )  [inline]
```

Compute the output signal.

Parameters

time	in seconds.

Definition at line 132 of file operators.hpp.

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

- /home/zoe/ganymede/code/cpp/chow-chow/include/chow-chow/operators.hpp

2.5 ChowChow::Phase Class Reference

Public Types

- using **phase_t** = uint64_t

Public Member Functions

- constexpr **Phase** (phase_t init)
- constexpr **Phase** (unsigned int init)
- constexpr **Phase** (unsigned short init)
- constexpr **Phase** (long init)
- constexpr **Phase** (int init)
- constexpr **Phase** (short init)
- constexpr **Phase** (long double init)
- constexpr **Phase** (double init)
- constexpr auto **phase** () const
- constexpr long double **phase_d** () const
- constexpr long double **phase_piscale** () const
- constexpr Amp **amp** () const
- void **phase** (phase_t pha)
- Phase & **operator+=** (Phase w)
- Phase & **operator-=** (Phase w)

Static Public Attributes

- static constexpr phase_t **PI** = UINT64_MAX
- static constexpr phase_t **ZERO** = UINT64_MAX/2
- static constexpr phase_t **PI_N** = 0
- static constexpr phase_t **PI_D2** = (PI/4)∗3
- static constexpr phase_t **PI_ND2** = PI/4
- static constexpr long double **LONG_D_PI**

2.5.1 Detailed Description

Definition at line 10 of file phase.hpp.

2.5.2 Member Data Documentation

2.5.2.1 LONG_D_PI

```
constexpr long double ChowChow::Phase::LONG_D_PI  [static], [constexpr]
```

Initial value:

```
=
            3.14159265358979323846264338327950288419716939937510L
```

Definition at line 52 of file phase.hpp.

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

- /home/zoe/ganymede/code/cpp/chow-chow/include/chow-chow/phase.hpp

2.6 ChowChow::Sample Class Reference

Public Types

- using **amp_t** = unsigned short
- using **smp_t** = short

Public Member Functions

- constexpr **Sample** (smp_t n)
- constexpr **Sample** (long double n)
- constexpr **Sample** (double n)
- constexpr **Sample** (int n)
- constexpr **Sample** (long n)
- constexpr **Sample** (long long n)
- constexpr amp_t **amp** () const
- constexpr long double **amp_f** () const
- constexpr smp_t **val** () const
- constexpr long double **val_f** () const
- void **amp** (amp_t n)
- void **val** (Sample s)
- void **amp** (long double n)
- void **amp** (double n)

Static Public Attributes

- static constexpr amp_t **AMP_MAX** = INT16_MIN ∗ -1
- static constexpr smp_t **SMP_MAX_POS** = INT16_MAX
- static constexpr int **SMP_MAX_NEG** = INT16_MIN ∗ -1

2.6.1 Detailed Description

Definition at line 8 of file sample.hpp.

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

- /home/zoe/ganymede/code/cpp/chow-chow/include/chow-chow/sample.hpp
- /home/zoe/ganymede/code/cpp/chow-chow/src/sample.cpp

2.7 ChowChow::Setting Class Reference

Public Types

- using **setting_t** = uint32_t

Public Member Functions

- **Setting** (double max)
- constexpr auto **max** () const
- constexpr setting_t **val_raw** () const
- constexpr double **val** () const
- void **max** (double f)
- void **val** (setting_t n)
- void **val** (double f)

Static Public Attributes

- static constexpr setting_t **SETTING_MAX** = UINT32_MAX

2.7.1 Detailed Description

Definition at line 8 of file setting.hpp.

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

- /home/zoe/ganymede/code/cpp/chow-chow/include/chow-chow/setting.hpp
- /home/zoe/ganymede/code/cpp/chow-chow/src/setting.cpp

2.8 ChowChow::WAVFile Class Reference

Represents audio in RIFF WAVE format.

`#include <wav_file.hpp>`

Public Member Functions

- WAVFile (const std::vector< long double > &samples, unsigned int sample_rate)
- WAVFile (std::vector< long double > &&samples, unsigned int sample_rate)
- std::string header () const

  The RIFF WAVE header.
- std::string data () const

  The formatted sound data.
- void write (std::filesystem::path path) const

  Writes the whole WAVE file to disk.

Static Public Attributes

- static constexpr uint32_t **NUM_CHANNELS** = 2
- static constexpr uint32_t **BITS_PER_SAMPLE** = 24
- static constexpr uint32_t **BYTES_PER_SAMPLE** = BITS_PER_SAMPLE / CHAR_BIT

2.8.1 Detailed Description

Represents audio in RIFF WAVE format.

Represents audio as a 24-bit stereo RIFF WAVE file. Supports writing it to disk for convenience.

Definition at line 16 of file wav_file.hpp.

2.8.2 Constructor & Destructor Documentation

2.8.2.1 WAVFile() [1/2]

```
WAVFile::WAVFile (
            const std::vector< long double > & samples,
            unsigned int sample_rate )
```

Parameters

samples	a stereo-interleaved collection of samples.
sample_rate	in Hz.

Definition at line 12 of file wav_file.cpp.

2.8.2.2 WAVFile() [2/2]

```
WAVFile::WAVFile (
            std::vector< long double > && samples,
            unsigned int sample_rate )
```

Parameters

samples	a stereo-interleaved collection of samples.
sample_rate	in Hz.

Definition at line 18 of file wav_file.cpp.

2.8.3 Member Function Documentation

2.8.3.1 write()

```
void WAVFile::write (
            std::filesystem::path path ) const
```

Writes the whole WAVE file to disk.

Parameters

path	the location to write to (including the filename).

Definition at line 101 of file wav_file.cpp.

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

- /home/zoe/ganymede/code/cpp/chow-chow/include/chow-chow/wav_file.hpp
- /home/zoe/ganymede/code/cpp/chow-chow/src/wav_file.cpp

Index