DaisySP

Generated by Doxygen 1.9.1

Chapter 1

Main Page

DaisySP A Powerful, Open Source DSP Library in C++

Applications• Features• Examples• Getting Started• Community• Contributing • License

1.0.1 Applications

- Embedded hardware using the Daisy Audio Platform
- Audio plug-ins (VST, AU, JUCE)
- Mobile apps (iOS, Android)
- VCV Rack modules

1.0.2 Features

- **Synthesis Methods:** Subtractive, Physical Modeling, FM
- **Filters:** Biquad, State-Variable, Modal, Comb
- **Effects Processors:** Reverb, Delay, Decimate, Compressor
- **Utilities:** Math Functions, Signal Conditioning, Aleatoric Generators

1.0.3 Code Example

1.0.4 Getting Started

- Get the source: `git clone` `https://github.com/electro-smith/DaisySP`
- Navigate to the DaisySP repo: `cd DaisySP`
- Build the library: `make`
- Make some noise with the `example programs!`

1.0.5 Community

Connect with other users and developers:

- Join the `Daisy Forum`
- Chat on the `Daisy Slack Workspace`

1.0.6 Contributing

Here are some ways that you can get involved:

- Proof read the `documentation` and suggest improvements
- Test existing functionality and make `issues`
- Make new DSP modules. See issues labeled "feature"
- Port existing DSP modules from other open source projects (MIT). See issues labeled "port"
- Fix problems with existing modules. See issues labeled "bug" and/or "polish"

Before working on code, please check out our `Contribution Guidelines` and `Style Guide.`

1.0.7 License

DaisySP uses the MIT license.

It can be used in both closed source and commercial projects, and does not provide a warranty of any kind.

For the full license, read the `LICENSE` file in the root directory.

Chapter 2

Todo List

Class daisysp::AdEnv

- Add Cycling
- Implement Curve (its only linear for now).
- Maybe make this an ADsr_ that has AD/AR/Asr_ modes.

Class daisysp::Compressor

Add soft/hard knee settings

Class daisysp::NlFilt

make this work on a single sample instead of just on blocks at a time.

Class daisysp::Phasor

Selecting which channels should be initialized/included in the sequence conversion.

Setup a similar start function for an external mux, but that seems outside the scope of this file.

Class daisysp::PitchShifter

- move hash_xs32 and myrand to dsp.h and give appropriate names

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Class Documentation

5.1 daisysp::AdEnv Class Reference

```
#include <adenv.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process ()
- void Trigger ()
- void SetTime (uint8_t seg, float time)
- void SetCurve (float scalar)
- void SetMin (float min)
- void SetMax (float max)
- float GetValue () const
- uint8_t GetCurrentSegment ()
- bool IsRunning () const

5.1.1 Detailed Description

Trigger-able envelope with adjustable min/max, and independent per-segment time control.

Author

shensley

Todo • Add Cycling

- Implement Curve (its only linear for now).

- Maybe make this an ADsr_ that has AD/AR/Asr_ modes.

5.1.2 Member Function Documentation

5.1.2.1 GetCurrentSegment()

```
uint8_t daisysp::AdEnv::GetCurrentSegment ( ) [inline]
```

Returns the segment of the envelope that the phase is currently located in.

5.1.2.2 GetValue()

```
float daisysp::AdEnv::GetValue ( ) const [inline]
```

Returns the current output value without processing the next sample

5.1.2.3 Init()

```
void AdEnv::Init (
             float sample_rate )
```

Initializes the ad envelope.

Defaults:

- current segment = idle
- curve = linear
- phase = 0
- min = 0
- max = 1

Parameters

sample_rate	sample rate of the audio engine being run

5.1.2.4 IsRunning()

```
bool daisysp::AdEnv::IsRunning ( ) const [inline]
```

Returns true if the envelope is currently in any stage apart from idle.

5.1.2.5 Process()

```
float AdEnv::Process ( )
```

Processes the current sample of the envelope. This should be called once per sample period.

Returns

the current envelope value.

5.1.2.6 SetCurve()

```
void daisysp::AdEnv::SetCurve (
             float scalar )  [inline]
```

Sets the amount of curve applied. A positve value will create a log curve. Input range: -100 to 100. (or more)

5.1.2.7 SetMax()

```
void daisysp::AdEnv::SetMax (
             float max )  [inline]
```

Sets the maximum value of the envelope output. Input range: -FLTmax_, to FLTmax_

5.1.2.8 SetMin()

```
void daisysp::AdEnv::SetMin (
             float min )  [inline]
```

Sets the minimum value of the envelope output. Input range: -FLTmax_, to FLTmax_

5.1.2.9 SetTime()

```
void daisysp::AdEnv::SetTime (
             uint8_t seg,
             float time )  [inline]
```

Sets the length of time (in seconds) for a specific segment.

5.1.2.10 Trigger()

```
void daisysp::AdEnv::Trigger ( )  [inline]
```

Starts or retriggers the envelope.

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

- Source/Control/adenv.h
- Source/Control/adenv.cpp

5.2 daisysp::Adsr Class Reference

```
#include <adsr.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (bool gate)
- void SetTime (int seg, float time)
- void SetSustainLevel (float sus_level)
- uint8_t GetCurrentSegment ()
- bool IsRunning () const

5.2.1 Detailed Description

adsr envelope module

Original author(s) : Paul Batchelor

Ported from Soundpipe by Ben Sergentanis, May 2020

5.2.2 Member Function Documentation

5.2.2.1 GetCurrentSegment()

```
uint8_t daisysp::Adsr::GetCurrentSegment ( )  [inline]
```

get the current envelope segment

Returns

the segment of the envelope that the phase is currently located in.

5.2.2.2 Init()

```
void Adsr::Init (
            float sample_rate )
```

Initializes the Adsr module.

Parameters

sample_rate	- The sample rate of the audio engine being run.

5.2.2.3 IsRunning()

```
bool daisysp::Adsr::IsRunning ( ) const  [inline]
```

Tells whether envelope is active

Returns

true if the envelope is currently in any stage apart from idle.

5.2.2.4 Process()

```
float Adsr::Process (
             bool gate )
```

Processes one sample through the filter and returns one sample.

Parameters

gate	- trigger the envelope, hold it to sustain

5.2.2.5 SetSustainLevel()

```
void daisysp::Adsr::SetSustainLevel (
             float sus_level )  [inline]
```

Sustain level

Parameters

sus_level	- sets sustain level

5.2.2.6 SetTime()

```
void daisysp::Adsr::SetTime (
             int seg,
             float time )  [inline]
```

Sets time Set time per segment in seconds

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

- Source/Control/adsr.h
- Source/Control/adsr.cpp

5.3 daisysp::Allpass Class Reference

```
#include <allpass.h>
```

Public Member Functions

- void Init (float sample_rate, float ∗buff, size_t size)
- float Process (float in)
- void SetFreq (float looptime)
- void SetRevTime (float revtime)

5.3.1 Detailed Description

Allpass filter module
Passes all frequencies at their original levels, with a phase shift.
Ported from soundpipe by Ben Sergentanis, May 2020

Author

Barry Vercoe, John ffitch

Date

1991

5.3.2 Member Function Documentation

5.3.2.1 Init()

```
void Allpass::Init (
            float sample_rate,
            float * buff,
            size_t size )
```

```
Initializes the allpass module.
\param sample_rate  The sample rate of the audio engine being run.
```

Parameters

buff	Buffer for allpass to use.
size	Size of buff.

5.3.2.2 Process()

```
float Allpass::Process (
             float in )
```

Parameters

in	Input sample.

Returns

Next floating point sample.

5.3.2.3 SetFreq()

```
void Allpass::SetFreq (
             float looptime )
```

Sets the filter frequency (Implemented by delay time).

Parameters

looptime	Filter looptime in seconds.

5.3.2.4 SetRevTime()

```
void daisysp::Allpass::SetRevTime (
             float revtime )  [inline]
```

Parameters

revtime	Reverb time in seconds.

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

- Source/Filters/allpass.h
- Source/Filters/allpass.cpp

5.4 daisysp::AnalogBassDrum Class Reference

808 bass drum model, revisited.

```
#include <analogbassdrum.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (bool trigger=false)
- void Trig ()
- void SetSustain (bool sustain)
- void SetAccent (float accent)
- void SetFreq (float f0)
- void SetTone (float tone)
- void SetDecay (float decay)
- void SetAttackFmAmount (float attack_fm_amount)
- void SetSelfFmAmount (float self_fm_amount)

5.4.1 Detailed Description

808 bass drum model, revisited.

Author

Ben Sergentanis

Date

Jan 2021 Ported from pichenettes/eurorack/plaits/dsp/drums/analog_bass_drum.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.4.2 Member Function Documentation

5.4.2.1 Init()

```
void AnalogBassDrum::Init (
            float sample_rate )
```

Initialize the module

Parameters

sample_rate	Audio engine sample rate

5.4.2.2 Process()

```
float AnalogBassDrum::Process (
            bool trigger = false )
```

Get the next sample.

Parameters

trigger	True strikes the drum. Defaults to false.

5.4.2.3 SetAccent()

```
void AnalogBassDrum::SetAccent (
             float accent )
```

Set a small accent.

Parameters

accent	Works 0-1

5.4.2.4 SetAttackFmAmount()

```
void AnalogBassDrum::SetAttackFmAmount (
             float attack_fm_amount )
```

Set the amount of fm attack. Works together with self fm.

Parameters

attack_fm_amount	Works best 0-1.

5.4.2.5 SetDecay()

```
void AnalogBassDrum::SetDecay (
             float decay )
```

Set the decay length of the drum.

Parameters

decay	Works best 0-1.

5.4.2.6 SetFreq()

```
void AnalogBassDrum::SetFreq (
             float f0 )
```

Set the drum's root frequency

Parameters

f0	Frequency in Hz

5.4.2.7 SetSelfFmAmount()

```
void AnalogBassDrum::SetSelfFmAmount (
             float self_fm_amount )
```

Set the amount of self fm. Also affects fm attack, and volume decay.

Parameters

self_fm_amount	Works best 0-1.

5.4.2.8 SetSustain()

```
void AnalogBassDrum::SetSustain (
             bool sustain )
```

Set the bassdrum to play infinitely

Parameters

sustain	True = infinite length

5.4.2.9 SetTone()

```
void AnalogBassDrum::SetTone (
             float tone )
```

Set the amount of click.

Parameters

tone	Works 0-1.

5.4.2.10 Trig()

```
void AnalogBassDrum::Trig ( )
```

Strikes the drum.

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

- Source/Drums/analogbassdrum.h
- Source/Drums/analogbassdrum.cpp

5.5 daisysp::AnalogSnareDrum Class Reference

808 snare drum model, revisited.

```
#include <analogsnaredrum.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (bool trigger=false)
- void Trig ()
- void SetSustain (bool sustain)
- void SetAccent (float accent)
- void SetFreq (float f0)
- void SetTone (float tone)
- void SetDecay (float decay)
- void SetSnappy (float snappy)

Static Public Attributes

- static const int **kNumModes** = 5

5.5.1 Detailed Description

808 snare drum model, revisited.

Author

Ben Sergentanis

Date

Jan 2021 Ported from pichenettes/eurorack/plaits/dsp/drums/analog_snare_drum.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.5.2 Member Function Documentation

5.5.2.1 Init()

```
void AnalogSnareDrum::Init (
            float sample_rate )
```

Init the module

Parameters

sample_rate	Audio engine sample rate

5.5.2.2 Process()

```
float AnalogSnareDrum::Process (
            bool trigger = false )
```

Get the next sample

Parameters

trigger	Hit the drum with true. Defaults to false.

5.5.2.3 SetAccent()

```
void AnalogSnareDrum::SetAccent (
            float accent )
```

Set how much accent to use

Parameters

accent	Works 0-1.

5.5.2.4 SetDecay()

```
void AnalogSnareDrum::SetDecay (
            float decay )
```

Set the length of the drum decay

Parameters

decay	Works with positive numbers

5.5.2.5 SetFreq()

```
void AnalogSnareDrum::SetFreq (
             float f0 )
```

Set the drum's root frequency

Parameters

f0	Freq in Hz

5.5.2.6 SetSnappy()

```
void AnalogSnareDrum::SetSnappy (
             float snappy )
```

Sets the mix between snare and drum.

Parameters

snappy	1 = just snare. 0 = just drum.

5.5.2.7 SetSustain()

```
void AnalogSnareDrum::SetSustain (
             bool sustain )
```

Init the module

Parameters

sample_rate	Audio engine sample rate

5.5.2.8 SetTone()

```
void AnalogSnareDrum::SetTone (
             float tone )
```

Set the brightness of the drum tone.

Parameters

tone	Works 0-1. 1 = bright, 0 = dark.

5.5.2.9 Trig()

```
void AnalogSnareDrum::Trig ( )
```

Trigger the drum

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

- Source/Drums/analogsnaredrum.h
- Source/Drums/analogsnaredrum.cpp

5.6 daisysp::ATone Class Reference

```
#include <atone.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float &in)
- void SetFreq (float &freq)
- float GetFreq ()

5.6.1 Detailed Description

A first-order recursive high-pass filter with variable frequency response. Original Author(s): Barry Vercoe, John FFitch, Gabriel Maldonado

Year: 1991

Original Location: Csound – OOps/ugens5.c

Ported from soundpipe by Ben Sergentanis, May 2020

5.6.2 Member Function Documentation

5.6.2.1 GetFreq()

```
float daisysp::ATone::GetFreq ( ) [inline]
```

get current frequency

Returns

the current value for the cutoff frequency or half-way point of the filter.

5.6.2.2 Init()

```
void ATone::Init (
             float sample_rate )
```

Initializes the ATone module.

Parameters

sample_rate	- The sample rate of the audio engine being run.

5.6.2.3 Process()

```
float ATone::Process (
             float & in )
```

Processes one sample through the filter and returns one sample.

Parameters

in	- input signal

5.6.2.4 SetFreq()

```
void daisysp::ATone::SetFreq (
             float & freq ) [inline]
```

Sets the cutoff frequency or half-way point of the filter.

Parameters

freq	- frequency value in Hz. Range: Any positive value.

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

- Source/Filters/atone.h
- Source/Filters/atone.cpp

5.7 daisysp::Autowah Class Reference

```
#include <autowah.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float in)
- void SetWah (float wah)
- void SetDryWet (float drywet)
- void SetLevel (float level)

5.7.1 Detailed Description

Autowah module

Original author(s) :

Ported from soundpipe by Ben Sergentanis, May 2020

5.7.2 Member Function Documentation

5.7.2.1 Init()

```
void Autowah::Init (
            float sample_rate )
```

Initializes the Autowah module.

Parameters

sample_rate	- The sample rate of the audio engine being run.

5.7.2.2 Process()

```
float Autowah::Process (
            float in )
```

Initializes the Autowah module.

Parameters

in	- input signal to be wah'd

5.7.2.3 SetDryWet()

```
void daisysp::Autowah::SetDryWet (
            float drywet ) [inline]
```

sets mix amount

Parameters

drywet	: set effect dry/wet

5.7.2.4 SetLevel()

```
void daisysp::Autowah::SetLevel (
            float level ) [inline]
```

sets wah level

Parameters

level	: set wah level

5.7.2.5 SetWah()

```
void daisysp::Autowah::SetWah (
            float wah ) [inline]
```

sets wah

Parameters

wah	: set wah amount

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

- Source/Effects/autowah.h
- Source/Effects/autowah.cpp

5.8 daisysp::Balance Class Reference

```
#include <balance.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float sig, float comp)
- void SetCutoff (float cutoff)

5.8.1 Detailed Description

Balances two sound sources. Sig is boosted to the level of comp.

Original author(s) : Barry Vercoe, john ffitch, Gabriel Maldonado

Year: 1991

Ported from soundpipe by Ben Sergentanis, May 2020

5.8.2 Member Function Documentation

5.8.2.1 Init()

```
void Balance::Init (
             float sample_rate )
```

Initializes the balance module.

Parameters

sample_rate	- The sample rate of the audio engine being run.

5.8.2.2 Process()

```
float Balance::Process (
             float sig,
             float comp )
```

adjust sig level to level of comp

5.8.2.3 SetCutoff()

```
void daisysp::Balance::SetCutoff (
             float cutoff )  [inline]
```

adjusts the rate at which level compensation happens

Parameters

cutoff	: Sets half power point of special internal cutoff filter.

defaults to 10

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

- Source/Dynamics/balance.h
- Source/Dynamics/balance.cpp

5.9 daisysp::Biquad Class Reference

```
#include <biquad.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float in)
- void SetRes (float res)
- void SetCutoff (float cutoff)

5.9.1 Detailed Description

Two pole recursive filter

Original author(s) : Hans Mikelson

Year: 1998

Ported from soundpipe by Ben Sergentanis, May 2020

5.9.2 Member Function Documentation

5.9.2.1 Init()

```
void Biquad::Init (
            float sample_rate )
```

Initializes the biquad module.

Parameters

sample_rate	- The sample rate of the audio engine being run.

5.9.2.2 Process()

```
float Biquad::Process (
            float in )
```

Filters the input signal

Returns

filtered output

5.9.2.3 SetCutoff()

```
void daisysp::Biquad::SetCutoff (
            float cutoff ) [inline]
```

Sets filter cutoff in Hz

Parameters

cutoff	: Set filter cutoff.

5.9.2.4 SetRes()

```
void daisysp::Biquad::SetRes (
            float res ) [inline]
```

Sets resonance amount

Parameters

res	: Set filter resonance.

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

- Source/Filters/biquad.h
- Source/Filters/biquad.cpp

5.10 daisysp::Bitcrush Class Reference

```
#include <bitcrush.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float in)
- void SetBitDepth (int bitdepth)
- void SetCrushRate (float crushrate)

5.10.1 Detailed Description

bitcrush module

Original author(s) : Paul Batchelor,

Ported from soundpipe by Ben Sergentanis, May 2020

5.10.2 Member Function Documentation

5.10.2.1 Init()

```
void Bitcrush::Init (
            float sample_rate )
```

Initializes the bitcrush module.

Parameters

sample_rate	- The sample rate of the audio engine being run.

5.10.2.2 Process()

```
float Bitcrush::Process (
             float in )
```

bit crushes and downsamples the input

5.10.2.3 SetBitDepth()

```
void daisysp::Bitcrush::SetBitDepth (
             int bitdepth )  [inline]
```

adjusts bitdepth

Parameters

bitdepth	: Sets bit depth.

5.10.2.4 SetCrushRate()

```
void daisysp::Bitcrush::SetCrushRate (
             float crushrate )  [inline]
```

adjusts the downsampling frequency

Parameters

crushrate	: Sets rate to downsample to.

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

- Source/Effects/bitcrush.h
- Source/Effects/bitcrush.cpp

5.11 daisysp::BlOsc Class Reference

```
#include <blosc.h>
```

Public Types

- enum Waveforms { **WAVE_TRIANGLE** , **WAVE_SAW** , **WAVE_SQUARE** , **WAVE_OFF** }

Public Member Functions

- void Init (float sample_rate)
- float Process ()
- void SetFreq (float freq)
- void SetAmp (float amp)
- void SetPw (float pw)
- void SetWaveform (uint8_t waveform)
- void Reset ()

5.11.1 Detailed Description

Band Limited Oscillator

Based on bltriangle, blsaw, blsquare from soundpipe

Original Author(s): Paul Batchelor, saw2 Faust by Julius Smith

Ported by Ben Sergentanis, May 2020

5.11.2 Member Enumeration Documentation

5.11.2.1 Waveforms

```
enum daisysp::BlOsc::Waveforms
```

Bl Waveforms

5.11.3 Member Function Documentation

5.11.3.1 Init()

```
void BlOsc::Init (
             float sample_rate )
```

-Initialize oscillator. -Defaults to: 440Hz, .5 amplitude, .5 pw, Triangle.

5.11.3.2 Process()

```
float BlOsc::Process ( )
```

- Get next floating point oscillator sample.

5.11.3.3 Reset()

```
void BlOsc::Reset ( )
```

- reset the phase of the oscillator.

5.11.3.4 SetAmp()

```
void daisysp::BlOsc::SetAmp (
            float amp )  [inline]
```

- Float amp: Set oscillator amplitude, 0 to 1.

5.11.3.5 SetFreq()

```
void daisysp::BlOsc::SetFreq (
            float freq )  [inline]
```

- Float freq: Set oscillator frequency in Hz.

5.11.3.6 SetPw()

```
void daisysp::BlOsc::SetPw (
            float pw )  [inline]
```

- Float pw: Set square osc pulsewidth, 0 to 1. (no thru 0 at the moment)

5.11.3.7 SetWaveform()

```
void daisysp::BlOsc::SetWaveform (
            uint8_t waveform )  [inline]
```

- uint8_t waveform: select between waveforms from enum above.
- i.e. SetWaveform(BL_WAVEFORM_SAW); to set waveform to saw

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

- Source/Synthesis/blosc.h
- Source/Synthesis/blosc.cpp

5.12 daisysp::Chorus Class Reference

Chorus Effect.

```
#include <chorus.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float in)
- float GetLeft ()
- float GetRight ()
- void SetPan (float panl, float panr)
- void SetPan (float pan)
- void SetLfoDepth (float depthl, float depthr)
- void SetLfoDepth (float depth)
- void SetLfoFreq (float freql, float freqr)
- void SetLfoFreq (float freq)
- void SetDelay (float delayl, float delayr)
- void SetDelay (float delay)
- void SetDelayMs (float msl, float msr)
- void SetDelayMs (float ms)
- void SetFeedback (float feedbackl, float feedbackr)
- void SetFeedback (float feedback)

5.12.1 Detailed Description

Chorus Effect.

Author

Ben Sergentanis

Date

Jan 2021 Based on `https://www.izotope.com/en/learn/understanding-chorus-flangers-and-pha` `html`
and `https://www.researchgate.net/publication/236629475_Implementing_↩` `Professional_Audio_Effects_with_DSPs`

5.12.2 Member Function Documentation

5.12.2.1 GetLeft()

```
float Chorus::GetLeft ( )
```

Get the left channel's last sample

5.12.2.2 GetRight()

```
float Chorus::GetRight ( )
```

Get the right channel's last sample

5.12.2.3 Init()

```
void Chorus::Init (
             float sample_rate )
```

Initialize the module

Parameters

sample_rate	Audio engine sample rate

5.12.2.4 Process()

```
float Chorus::Process (
             float in )
```

Get the net floating point sample. Defaults to left channel.

Parameters

in	Sample to process

5.12.2.5 SetDelay() [1/2]

```
void Chorus::SetDelay (
             float delay )
```

Set both channel delay amounts.

Parameters

delay	Both channel delay amount. Works 0-1.

5.12.2.6 SetDelay() [2/2]

```
void Chorus::SetDelay (
             float delayl,
             float delayr )
```

Set both channel delay amounts individually.

Parameters

delayl	Left channel delay amount. Works 0-1.
delayr	Right channel delay amount.

5.12.2.7 SetDelayMs() [1/2]

```
void Chorus::SetDelayMs (
             float ms )
```

Set both channel delay in ms.

Parameters

ms	Both channel delay amounts in ms.

5.12.2.8 SetDelayMs() [2/2]

```
void Chorus::SetDelayMs (
             float msl,
             float msr )
```

Set both channel delay individually.

Parameters

msl	Left channel delay in ms.
msr	Right channel delay in ms.

5.12.2.9 SetFeedback() [1/2]

```
void Chorus::SetFeedback (
             float feedback )
```

Set both channels feedback.

Parameters

feedback	Both channel feedback. Works 0-1.

5.12.2.10 SetFeedback() [2/2]

```
void Chorus::SetFeedback (
            float feedbackl,
            float feedbackr )
```

Set both channels feedback individually.

Parameters

feedbackl	Left channel feedback. Works 0-1.
feedbackr	Right channel feedback.

5.12.2.11 SetLfoDepth() [1/2]

```
void Chorus::SetLfoDepth (
            float depth )
```

Set both lfo depths.

Parameters

depth	Both channels lfo depth. Works 0-1.

5.12.2.12 SetLfoDepth() [2/2]

```
void Chorus::SetLfoDepth (
            float depthl,
            float depthr )
```

Set both lfo depths individually.

Parameters

depthl	Left channel lfo depth. Works 0-1.
depthr	Right channel lfo depth.

5.12.2.13 SetLfoFreq() [1/2]

```
void Chorus::SetLfoFreq (
             float freq )
```

Set both lfo frequencies.

Parameters

depth	Both channel lfo freqs in Hz.

5.12.2.14 SetLfoFreq() [2/2]

```
void Chorus::SetLfoFreq (
             float freql,
             float freqr )
```

Set both lfo frequencies individually.

Parameters

depthl	Left channel lfo freq in Hz.
depthr	Right channel lfo freq in Hz.

5.12.2.15 SetPan() [1/2]

```
void Chorus::SetPan (
             float pan )
```

Pan both channels.

Parameters

pan	Where to pan both channels to. 0 is left, 1 is right.

5.12.2.16 SetPan() [2/2]

```
void Chorus::SetPan (
             float panl,
             float panr )
```

Pan both channels individually.

Parameters

panl	Pan the left channel. 0 is left, 1 is right.
panr	Pan the right channel.

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

- Source/Effects/chorus.h
- Source/Effects/chorus.cpp

5.13 daisysp::ChorusEngine Class Reference

Single Chorus engine. Used in Chorus.

```
#include <chorus.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float in)
- void SetLfoDepth (float depth)
- void SetLfoFreq (float freq)
- void SetDelay (float delay)
- void SetDelayMs (float ms)
- void SetFeedback (float feedback)

5.13.1 Detailed Description

Single Chorus engine. Used in Chorus.

Author

Ben Sergentanis

5.13.2 Member Function Documentation

5.13.2.1 Init()

```
void ChorusEngine::Init (
            float sample_rate )
```

Initialize the module

Parameters

sample_rate	Audio engine sample rate.

5.13.2.2 Process()

```
float ChorusEngine::Process (
            float in )
```

Get the next sample

Parameters

in	Sample to process

5.13.2.3 SetDelay()

```
void ChorusEngine::SetDelay (
            float delay )
```

Set the internal delay rate.

Parameters

delay	Tuned for 0-1. Maps to .1 to 50 ms.

5.13.2.4 SetDelayMs()

```
void ChorusEngine::SetDelayMs (
            float ms )
```

Set the delay time in ms.

Parameters

ms	Delay time in ms.

5.13.2.5 SetFeedback()

```
void ChorusEngine::SetFeedback (
             float feedback )
```

Set the feedback amount.

Parameters

feedback	Amount from 0-1.

5.13.2.6 SetLfoDepth()

```
void ChorusEngine::SetLfoDepth (
             float depth )
```

How much to modulate the delay by.

Parameters

depth	Works 0-1.

5.13.2.7 SetLfoFreq()

```
void ChorusEngine::SetLfoFreq (
             float freq )
```

Set lfo frequency.

Parameters

freq	Frequency in Hz

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

- Source/Effects/chorus.h
- Source/Effects/chorus.cpp

5.14 daisysp::ClockedNoise Class Reference

```
#include <clockednoise.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process ()
- void SetFreq (float freq)
- void Sync ()

5.14.1 Detailed Description

```
@brief Clocked Noise Module
```

Author

Ported by Ben Sergentanis

Date

Jan 2021 Noise processed by a sample and hold running at a target frequency.

Ported from pichenettes/eurorack/plaits/dsp/noise/clocked_noise.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.14.2 Member Function Documentation

5.14.2.1 Init()

```
void ClockedNoise::Init (
             float sample_rate )
```

Initialize module

Parameters

sample_rate	Audio engine sample rate

5.14.2.2 Process()

```
float ClockedNoise::Process ( )
```

Get the next floating point sample

5.14.2.3 SetFreq()

```
void ClockedNoise::SetFreq (
             float freq )
```

Set the frequency at which the next sample is generated.

Parameters

freq	Frequency in Hz

5.14.2.4 Sync()

```
void ClockedNoise::Sync ( )
```

Calling this forces another random float to be generated

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

- Source/Noise/clockednoise.h
- Source/Noise/clockednoise.cpp

5.15 daisysp::Comb Class Reference

```
#include <comb.h>
```

Public Member Functions

- void Init (float sample_rate, float ∗buff, size_t size)
- float Process (float in)
- void SetPeriod (float looptime)
- void SetFreq (float freq)
- void SetRevTime (float revtime)

5.15.1 Detailed Description

Comb filter module

Original author(s) :

Ported from soundpipe by Ben Sergentanis, May 2020

5.15.2 Member Function Documentation

5.15.2.1 Init()

```
void Comb::Init (
             float sample_rate,
             float ∗ buff,
             size_t size )
```

Initializes the Comb module.

Parameters

sample_rate	- The sample rate of the audio engine being run.
buff	- input buffer, kept in either main() or global space
size	- size of buff

5.15.2.2 Process()

```
float Comb::Process (
            float in )
```

processes the comb filter

5.15.2.3 SetFreq()

```
void daisysp::Comb::SetFreq (
            float freq ) [inline]
```

Sets the frequency of the comb filter in Hz

5.15.2.4 SetPeriod()

```
void Comb::SetPeriod (
            float looptime )
```

Sets the period of the comb filter in seconds

5.15.2.5 SetRevTime()

```
void daisysp::Comb::SetRevTime (
            float revtime ) [inline]
```

Sets the decay time of the comb filter

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

- Source/Filters/comb.h
- Source/Filters/comb.cpp

5.16 daisysp::Compressor Class Reference

```
#include <compressor.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float in)
- float Process (float in, float key)
- float Apply (float in)
- void ProcessBlock (float ∗in, float ∗out, size_t size)
- void ProcessBlock (float ∗in, float ∗out, float ∗key, size_t size)
- void ProcessBlock (float ∗∗in, float ∗∗out, float ∗key, size_t channels, size_t size)
- float GetRatio ()
- void SetRatio (float ratio)
- float GetThreshold ()
- void SetThreshold (float threshold)
- float GetAttack ()
- void SetAttack (float attack)
- float GetRelease ()
- void SetRelease (float release)
- float GetMakeup ()
- void SetMakeup (float gain)
- void AutoMakeup (bool enable)
- float GetGain ()

5.16.1 Detailed Description

dynamics compressor

influenced by compressor in soundpipe (from faust).

Modifications made to do:

- Less calculations during each process loop (coefficients recalculated on parameter change).
- C++-ified
- added sidechain support
- pulled gain apart for monitoring and multichannel support
- improved readability
- improved makeup-gain calculations
- changing controls now costs a lot less
- a lot less expensive

by: shensley, improved upon by AvAars

Todo Add soft/hard knee settings

5.16.2 Member Function Documentation

5.16.2.1 Apply()

```
float daisysp::Compressor::Apply (
            float in )  [inline]
```

Apply compression to the audio signal, based on the previously calculated gain

Parameters

in	audio input signal

5.16.2.2 AutoMakeup()

```
void daisysp::Compressor::AutoMakeup (
             bool enable )  [inline]
```

Enables or disables the automatic makeup gain. Disabling sets the makeup gain to 0.0

Parameters

enable	true to enable, false to disable

5.16.2.3 GetAttack()

```
float daisysp::Compressor::GetAttack ( )  [inline]
```

Gets the envelope time for onset of compression

5.16.2.4 GetGain()

```
float daisysp::Compressor::GetGain ( )  [inline]
```

Gets the gain reduction in dB

5.16.2.5 GetMakeup()

```
float daisysp::Compressor::GetMakeup ( )  [inline]
```

Gets the additional gain to make up for the compression

5.16.2.6 GetRatio()

```
float daisysp::Compressor::GetRatio ( )  [inline]
```

Gets the amount of gain reduction

5.16.2.7 GetRelease()

```
float daisysp::Compressor::GetRelease ( ) [inline]
```

Gets the envelope time for release of compression

5.16.2.8 GetThreshold()

```
float daisysp::Compressor::GetThreshold ( ) [inline]
```

Gets the threshold in dB

5.16.2.9 Init()

```
void Compressor::Init (
            float sample_rate )
```

Initializes compressor

Parameters

sample_rate	rate at which samples will be produced by the audio engine.

5.16.2.10 Process() [1/2]

```
float Compressor::Process (
            float in )
```

Compress the audio input signal, saves the calculated gain

Parameters

in	audio input signal

5.16.2.11 Process() [2/2]

```
float daisysp::Compressor::Process (
            float in,
            float key ) [inline]
```

Compresses the audio input signal, keyed by a secondary input.

Parameters

in	audio input signal (to be compressed)
key	audio input that will be used to side-chain the compressor

5.16.2.12 ProcessBlock() [1/3]

```
void Compressor::ProcessBlock (
            float ** in,
            float ** out,
            float * key,
            size_t channels,
            size_t size )
```

Compresses a block of multiple channels of audio, keyed by a secondary input

Parameters

in	audio input signals (to be compressed)
out	audio output signals
key	audio input that will be used to side-chain the compressor
channels	the number of audio channels
size	the size of the block

5.16.2.13 ProcessBlock() [2/3]

```
void Compressor::ProcessBlock (
            float * in,
            float * out,
            float * key,
            size_t size )
```

Compresses a block of audio, keyed by a secondary input

Parameters

in	audio input signal (to be compressed)
out	audio output signal
key	audio input that will be used to side-chain the compressor
size	the size of the block

5.16.2.14 ProcessBlock() [3/3]

```
void daisysp::Compressor::ProcessBlock (
            float * in,
            float * out,
            size_t size ) [inline]
```

Compresses a block of audio

Parameters

in	audio input signal
out	audio output signal
size	the size of the block

5.16.2.15 SetAttack()

```
void daisysp::Compressor::SetAttack (
            float attack ) [inline]
```

Sets the envelope time for onset of compression for signals above the threshold.

Parameters

attack	Expects 0.001 -> 10

5.16.2.16 SetMakeup()

```
void daisysp::Compressor::SetMakeup (
            float gain ) [inline]
```

Manually sets the additional gain to make up for the compression

Parameters

gain	Expects 0.0 -> 80

5.16.2.17 SetRatio()

```
void daisysp::Compressor::SetRatio (
            float ratio ) [inline]
```

Sets the amount of gain reduction applied to compressed signals

Parameters

ratio	Expects 1.0 -> 40. (untested with values < 1.0)

5.16.2.18 SetRelease()

```
void daisysp::Compressor::SetRelease (
             float release )  [inline]
```

Sets the envelope time for release of compression as input signal falls below threshold.

Parameters

release	Expects 0.001 -> 10

5.16.2.19 SetThreshold()

```
void daisysp::Compressor::SetThreshold (
             float threshold )  [inline]
```

Sets the threshold in dB at which compression will be applied

Parameters

threshold	Expects 0.0 -> -80.

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

- Source/Dynamics/compressor.h
- Source/Dynamics/compressor.cpp

5.17 daisysp::CrossFade Class Reference

```
#include <crossfade.h>
```

Public Member Functions

- void Init (int curve)
- void Init ()
- float Process (float &in1, float &in2)
- void SetPos (float pos)
- void SetCurve (uint8_t curve)
- float GetPos (float pos)
- uint8_t GetCurve (uint8_t curve)

5.17.1 Detailed Description

Performs a CrossFade between two signals

Original author: Paul Batchelor

Ported from Soundpipe by Andrew Ikenberry

added curve option for constant power, etc.

5.17.2 Member Function Documentation

5.17.2.1 GetCurve()

```
uint8_t daisysp::CrossFade::GetCurve (
             uint8_t curve )  [inline]
```

Returns current curve

5.17.2.2 GetPos()

```
float daisysp::CrossFade::GetPos (
             float pos )  [inline]
```

Returns current position

5.17.2.3 Init() [1/2]

```
void daisysp::CrossFade::Init ( )  [inline]
```

Initialize with default linear curve

5.17.2.4 Init() [2/2]

```
void daisysp::CrossFade::Init (
             int curve )  [inline]
```

Initializes CrossFade module Defaults

- current position = .5
- curve = linear

5.17.2.5 Process()

```
float CrossFade::Process (
            float & in1,
            float & in2 )
```

processes CrossFade and returns single sample

5.17.2.6 SetCurve()

```
void daisysp::CrossFade::SetCurve (
            uint8_t curve ) [inline]
```

Sets current curve applied to CrossFade Expected input: See `Curve Options`

5.17.2.7 SetPos()

```
void daisysp::CrossFade::SetPos (
            float pos ) [inline]
```

Sets position of CrossFade between two input signals Input range: 0 to 1

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

- Source/Dynamics/crossfade.h
- Source/Dynamics/crossfade.cpp

5.18 daisysp::DcBlock Class Reference

```
#include <dcblock.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float in)

5.18.1 Detailed Description

Removes DC component of a signal

5.18.2 Member Function Documentation

5.18.2.1 Init()

```
void DcBlock::Init (
             float sample_rate )
```

Initializes DcBlock module

5.18.2.2 Process()

```
float DcBlock::Process (
             float in )
```

performs DcBlock Process

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

- Source/Utility/dcblock.h
- Source/Utility/dcblock.cpp

5.19 daisysp::Decimator Class Reference

```
#include <decimator.h>
```

Public Member Functions

- void Init ()
- float Process (float input)
- void SetDownsampleFactor (float downsample_factor)
- void SetBitcrushFactor (float bitcrush_factor)
- void SetBitsToCrush (const uint8_t &bits)
- float GetDownsampleFactor ()
- float GetBitcrushFactor ()

5.19.1 Detailed Description

Performs downsampling and bitcrush effects

5.19.2 Member Function Documentation

5.19.2.1 GetBitcrushFactor()

```
float daisysp::Decimator::GetBitcrushFactor ( )  [inline]
```

Returns current setting of bitcrush

5.19.2.2 GetDownsampleFactor()

```
float daisysp::Decimator::GetDownsampleFactor ( )  [inline]
```

Returns current setting of downsample

5.19.2.3 Init()

```
void Decimator::Init ( )
```

Initializes downsample module

5.19.2.4 Process()

```
float Decimator::Process (
             float input )
```

Applies downsample and bitcrush effects to input signal.

Returns

one sample. This should be called once per sample period.

5.19.2.5 SetBitcrushFactor()

```
void daisysp::Decimator::SetBitcrushFactor (
             float bitcrush_factor )  [inline]
```

Sets amount of bitcrushing Input range:

5.19.2.6 SetBitsToCrush()

```
void daisysp::Decimator::SetBitsToCrush (
             const uint8_t & bits )  [inline]
```

Sets the exact number of bits to crush 0-16 bits

5.19.2.7 SetDownsampleFactor()

```
void daisysp::Decimator::SetDownsampleFactor (
             float downsample_factor )  [inline]
```

Sets amount of downsample Input range:

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

- Source/Effects/decimator.h
- Source/Effects/decimator.cpp

5.20 daisysp::DelayLine< T, max_size > Class Template Reference

```
#include <delayline.h>
```

Public Member Functions

- void Init ()
- void Reset ()
- void SetDelay (size_t delay)
- void SetDelay (float delay)
- void Write (const T sample)
- const T Read () const
- const T Read (float delay) const
- const T **ReadHermite** (float delay) const
- const T **Allpass** (const T sample, size_t delay, const T coefficient)

5.20.1 Detailed Description

template<typename T, size_t max_size>
class daisysp::DelayLine< T, max_size >

Simple Delay line. November 2019

Converted to Template December 2019

declaration example: (1 second of floats)

DelayLine<float, SAMPLE_RATE> del;

By: shensley

5.20.2 Member Function Documentation

5.20.2.1 Init()

```
template<typename T , size_t max_size>
void daisysp::DelayLine< T, max_size >::Init ( )  [inline]
```

initializes the delay line by clearing the values within, and setting delay to 1 sample.

5.20.2.2 Read() [1/2]

```
template<typename T , size_t max_size>
const T daisysp::DelayLine< T, max_size >::Read ( ) const  [inline]
```

returns the next sample of type T in the delay line, interpolated if necessary.

5.20.2.3 Read() [2/2]

```
template<typename T , size_t max_size>
const T daisysp::DelayLine< T, max_size >::Read (
             float delay ) const  [inline]
```

Read from a set location

5.20.2.4 Reset()

```
template<typename T , size_t max_size>
void daisysp::DelayLine< T, max_size >::Reset ( )  [inline]
```

clears buffer, sets write ptr to 0, and delay to 1 sample.

5.20.2.5 SetDelay() [1/2]

```
template<typename T , size_t max_size>
void daisysp::DelayLine< T, max_size >::SetDelay (
             float delay )  [inline]
```

sets the delay time in samples If a float is passed in, a fractional component will be calculated for interpolating the delay line.

5.20.2.6 SetDelay() [2/2]

```
template<typename T , size_t max_size>
void daisysp::DelayLine< T, max_size >::SetDelay (
             size_t delay )  [inline]
```

sets the delay time in samples If a float is passed in, a fractional component will be calculated for interpolating the delay line.

5.20.2.7 Write()

```
template<typename T , size_t max_size>
void daisysp::DelayLine< T, max_size >::Write (
             const T sample )  [inline]
```

writes the sample of type T to the delay line, and advances the write ptr

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

- Source/Utility/delayline.h

5.21 daisysp::Drip Class Reference

```
#include <drip.h>
```

Public Member Functions

- void Init (float sample_rate, float dettack)
- float Process (bool trig)

5.21.1 Detailed Description

Imitates the sound of dripping water via Physical Modeling Synthesis.
Ported from soundpipe by Ben Sergentanis, May 2020

Author

Perry Cook

Date

2000

5.21.2 Member Function Documentation

5.21.2.1 Init()

```
void Drip::Init (
             float sample_rate,
             float dettack )
```

```
Initializes the Drip module.
\param sample_rate The sample rate of the audio engine being run.
```

Parameters

dettack	The period of time over which all sound is stopped.

5.21.2.2 Process()

```
float Drip::Process (
             bool trig )
```

Process the next floating point sample.

Parameters

trig	If true, begins a new drip.

Returns

Next sample.

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

- Source/PhysicalModeling/drip.h
- Source/PhysicalModeling/drip.cpp

5.22 daisysp::Dust Class Reference

Dust Module.

```
#include <dust.h>
```

Public Member Functions

- void **Init** ()
- float **Process** ()
- void **SetDensity** (float density)

5.22.1 Detailed Description

Dust Module.

Author

Ported by Ben Sergentanis

Date

Jan 2021 Randomly Clocked Samples

Ported from pichenettes/eurorack/plaits/dsp/noise/dust.h
to an independent module.
Original code written by Emilie Gillet in 2016.

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

- Source/Noise/dust.h

5.23 daisysp::Flanger Class Reference

Flanging Audio Effect.

```
#include <flanger.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float in)
- void SetFeedback (float feedback)
- void SetLfoDepth (float depth)
- void SetLfoFreq (float freq)
- void SetDelay (float delay)
- void SetDelayMs (float ms)

5.23.1 Detailed Description

Flanging Audio Effect.

Generates a modulating phase shifted copy of a signal, and recombines with the original to create a 'flanging' sound effect.

5.23.2 Member Function Documentation

5.23.2.1 Init()

```
void Flanger::Init (
            float sample_rate )
```

Initialize the modules

Parameters

sample_rate	Audio engine sample rate.

5.23.2.2 Process()

```
float Flanger::Process (
            float in )
```

Get the next sample

Parameters

in	Sample to process

5.23.2.3 SetDelay()

```
void Flanger::SetDelay (
             float delay )
```

Set the internal delay rate.

Parameters

delay	Tuned for 0-1. Maps to .1 to 7 ms.

5.23.2.4 SetDelayMs()

```
void Flanger::SetDelayMs (
             float ms )
```

Set the delay time in ms.

Parameters

ms	Delay time in ms.

5.23.2.5 SetFeedback()

```
void Flanger::SetFeedback (
             float feedback )
```

How much of the signal to feedback into the delay line.

Parameters

feedback	Works 0-1.

5.23.2.6 SetLfoDepth()

```
void Flanger::SetLfoDepth (
             float depth )
```

How much to modulate the delay by.

Parameters

depth	Works 0-1.

5.23.2.7 SetLfoFreq()

```
void Flanger::SetLfoFreq (
             float freq )
```

Set lfo frequency.

Parameters

freq	Frequency in Hz

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

- Source/Effects/flanger.h
- Source/Effects/flanger.cpp

5.24 daisysp::Fm2 Class Reference

```
#include <fm2.h>
```

Public Member Functions

- void Init (float samplerate)
- float Process ()
- void SetFrequency (float freq)
- void SetRatio (float ratio)
- void SetIndex (float index)
- float GetIndex ()
- void Reset ()

5.24.1 Detailed Description

Simple 2 operator FM synth voice.

Date: November, 2020

Author: Ben Sergentanis

5.24.2 Member Function Documentation

5.24.2.1 GetIndex()

```
float Fm2::GetIndex ( )
```

Returns the current FM index.

5.24.2.2 Init()

```
void Fm2::Init (
             float samplerate )
```

Initializes the FM2 module.

Parameters

samplerate	- The sample rate of the audio engine being run.

5.24.2.3 Process()

```
float Fm2::Process ( )
```

Returns the next sample

5.24.2.4 Reset()

```
void Fm2::Reset ( )
```

Resets both oscillators

5.24.2.5 SetFrequency()

```
void Fm2::SetFrequency (
             float freq )
```

Carrier freq. setter

Parameters

freq	Carrier frequency in Hz

5.24.2.6 SetIndex()

```
void Fm2::SetIndex (
             float index )
```

Index setter

Parameters

FM	depth, 5 = 2PI rads

5.24.2.7 SetRatio()

```
void Fm2::SetRatio (
             float ratio )
```

Set modulator freq. relative to carrier

Parameters

ratio	New modulator freq = carrier freq. ∗ ratio

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

- Source/Synthesis/fm2.h
- Source/Synthesis/fm2.cpp

5.25 daisysp::Fold Class Reference

```
#include <fold.h>
```

Public Member Functions

- void Init ()
- float Process (float in)
- void SetIncrement (float incr)

5.25.1 Detailed Description

fold module

Original author(s) : John FFitch, Gabriel Maldonado

Year : 1998

Ported from soundpipe by Ben Sergentanis, May 2020

5.25.2 Member Function Documentation

5.25.2.1 Init()

```
void Fold::Init ( )
```

Initializes the fold module.

5.25.2.2 Process()

```
float Fold::Process (
            float in )
```

applies foldvoer distortion to input

5.25.2.3 SetIncrement()

```
void daisysp::Fold::SetIncrement (
            float incr )  [inline]
```

Parameters

incr	: set fold increment

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

- Source/Effects/fold.h
- Source/Effects/fold.cpp

5.26 daisysp::FormantOscillator Class Reference

Formant Oscillator Module.

```
#include <formantosc.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process ()
- void SetFormantFreq (float freq)
- void SetCarrierFreq (float freq)
- void SetPhaseShift (float ps)

5.26.1 Detailed Description

Formant Oscillator Module.

Author

Ben Sergentanis

Date

Dec 2020 Sinewave with aliasing-free phase reset.

Ported from pichenettes/eurorack/plaits/dsp/oscillator/formant_oscillator.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.26.2 Member Function Documentation

5.26.2.1 Init()

```
void FormantOscillator::Init (
             float sample_rate )
```

Initializes the FormantOscillator module.

Parameters

sample_rate	- The sample rate of the audio engine being run.

5.26.2.2 Process()

```
float FormantOscillator::Process ( )
```

Get the next sample

5.26.2.3 SetCarrierFreq()

```
void FormantOscillator::SetCarrierFreq (
             float freq )
```

Set the carrier frequency. This is the "main" frequency.

Parameters

freq	Frequency in Hz

5.26.2.4 SetFormantFreq()

```
void FormantOscillator::SetFormantFreq (
            float freq )
```

Set the formant frequency.

Parameters

freq	Frequency in Hz

5.26.2.5 SetPhaseShift()

```
void FormantOscillator::SetPhaseShift (
            float ps )
```

Set the amount of phase shift

Parameters

ps	Typically 0-1. Works with other values though, including negative.

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

- Source/Synthesis/formantosc.h
- Source/Synthesis/formantosc.cpp

5.27 daisysp::FractalRandomGenerator< T, order > Class Template Reference

Fractal Noise, stacks octaves of a noise source.

`#include <fractal_noise.h>`

Public Member Functions

- void Init (float sample_rate)
- float Process ()
- void SetFreq (float freq)
- void SetColor (float color)

5.27.1 Detailed Description

template<typename T, int order>
class daisysp::FractalRandomGenerator< T, order >

Fractal Noise, stacks octaves of a noise source.

Author

Ported by Ben Sergentanis

Date

Jan 2021 T is the noise source to use. T must have SetFreq() and Init(sample_rate) functions.
Order is the number of noise sources to stack.

Ported from pichenettes/eurorack/plaits/dsp/noise/fractal_random_generator.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.27.2 Member Function Documentation

5.27.2.1 Init()

```
template<typename T , int order>
void daisysp::FractalRandomGenerator< T, order >::Init (
             float sample_rate )  [inline]
```

Initialize the module

Parameters

sample_rate	Audio engine sample rate.

5.27.2.2 Process()

```
template<typename T , int order>
float daisysp::FractalRandomGenerator< T, order >::Process ( )  [inline]
```

Get the next sample.

5.27.2.3 SetColor()

```
template<typename T , int order>
void daisysp::FractalRandomGenerator< T, order >::SetColor (
             float color )  [inline]
```

Sets the amount of high frequency noise. ** Works 0-1. 1 is the brightest, and 0 is the darkest.

5.27.2.4 SetFreq()

```
template<typename T , int order>
void daisysp::FractalRandomGenerator< T, order >::SetFreq (
             float freq )  [inline]
```

Set the lowest noise frequency.

Parameters

freq	Frequency of the lowest noise source in Hz.

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

- Source/Noise/fractal_noise.h

5.28 daisysp::GrainletOscillator Class Reference

Granular Oscillator Module.

```
#include <grainlet.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process ()
- void SetFreq (float freq)
- void SetFormantFreq (float freq)
- void SetShape (float shape)
- void SetBleed (float bleed)

5.28.1 Detailed Description

Granular Oscillator Module.

Author

Ben Sergentanis

Date

Dec 2020 A phase-distorted single cycle sine $*$ another continuously running sine,
the whole thing synced to a main oscillator.

Ported from pichenettes/eurorack/plaits/dsp/oscillator/grainlet_oscillator.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.28.2 Member Function Documentation

5.28.2.1 Init()

```
void GrainletOscillator::Init (
            float sample_rate )
```

Initialize the oscillator

Parameters

sample_rate	Sample rate of audio engine

5.28.2.2 Process()

```
float GrainletOscillator::Process ( )
```

Get the next sample

5.28.2.3 SetBleed()

```
void GrainletOscillator::SetBleed (
            float bleed )
```

Sets the amount of formant to bleed through

Parameters

bleed	Works best 0-1

5.28.2.4 SetFormantFreq()

```
void GrainletOscillator::SetFormantFreq (
             float freq )
```

Sets the formant frequency

Parameters

freq	Frequency in Hz

5.28.2.5 SetFreq()

```
void GrainletOscillator::SetFreq (
             float freq )
```

Sets the carrier frequency

Parameters

freq	Frequency in Hz

5.28.2.6 SetShape()

```
void GrainletOscillator::SetShape (
             float shape )
```

Waveshaping

Parameters

shape	Shapes differently from 0-1, 1-2, and $>$ 2.

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

- Source/Noise/grainlet.h
- Source/Noise/grainlet.cpp

5.29 daisysp::HarmonicOscillator< num_harmonics > Class Template Reference

Harmonic Oscillator Module based on Chebyshev polynomials.

```
#include <harmonic_osc.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process ()
- void SetFreq (float freq)
- void SetFirstHarmIdx (int idx)
- void SetAmplitudes (const float ∗amplitudes)
- void SetSingleAmp (const float amp, int idx)

5.29.1 Detailed Description

template<int num_harmonics = 16>
class daisysp::HarmonicOscillator< num_harmonics >

Harmonic Oscillator Module based on Chebyshev polynomials.

Author

Ben Sergentanis

Date

Dec 2020 Harmonic Oscillator Module based on Chebyshev polynomials
Works well for a small number of harmonics. For the higher order harmonics.
We need to reinitialize the recurrence by computing two high harmonics.

Ported from pichenettes/eurorack/plaits/dsp/oscillator/harmonic_oscillator.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.29.2 Member Function Documentation

5.29.2.1 Init()

```
template<int num_harmonics = 16>
void daisysp::HarmonicOscillator< num_harmonics >::Init (
            float sample_rate )  [inline]
```

Initialize harmonic oscillator

Parameters

sample_rate	Audio engine samplerate

5.29.2.2 Process()

```
template<int num_harmonics = 16>
float daisysp::HarmonicOscillator< num_harmonics >::Process ( ) [inline]
```

Get the next floating point sample

5.29.2.3 SetAmplitudes()

```
template<int num_harmonics = 16>
void daisysp::HarmonicOscillator< num_harmonics >::SetAmplitudes (
             const float * amplitudes ) [inline]
```

Set the amplitudes of each harmonic of the root.

Parameters

amplitudes	Amplitudes to set. Sum of all amplitudes must be < 1. The array referenced must be at least as large as num_harmonics.

5.29.2.4 SetFirstHarmIdx()

```
template<int num_harmonics = 16>
void daisysp::HarmonicOscillator< num_harmonics >::SetFirstHarmIdx (
             int idx ) [inline]
```

Offset the set of harmonics. Passing in 3 means "harmonic 0" is the 3rd harm., 1 is the 4th, etc.

Parameters

idx	Default behavior is 1. Values < 0 default to 1.

5.29.2.5 SetFreq()

```
template<int num_harmonics = 16>
void daisysp::HarmonicOscillator< num_harmonics >::SetFreq (
             float freq ) [inline]
```

Set the main frequency

Parameters

freq	Freq to be set in Hz.

5.29.2.6 SetSingleAmp()

```
template<int num_harmonics = 16>
void daisysp::HarmonicOscillator< num_harmonics >::SetSingleAmp (
              const float amp,
              int idx ) [inline]
```

Sets one amplitude. Does nothing if idx out of range.

Parameters

amp	Amplitude to set
idx	Which harmonic to set.

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

- Source/Synthesis/harmonic_osc.h

5.30 daisysp::HiHat< MetallicNoiseSource, VCA, resonance > Class Template Reference

808 HH, with a few extra parameters to push things to the CY territory...

```
#include <hihat.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (bool trigger=false)
- void Trig ()
- void SetSustain (bool sustain)
- void SetAccent (float accent)
- void SetFreq (float f0)
- void SetTone (float tone)
- void SetDecay (float decay)
- void SetNoisiness (float noisiness)

5.30.1 Detailed Description

template<typename MetallicNoiseSource = SquareNoise, typename VCA = LinearVCA, bool resonance = true>
class daisysp::HiHat< MetallicNoiseSource, VCA, resonance >

808 HH, with a few extra parameters to push things to the CY territory...

Author

Ben Sergentanis

Date

Jan 2021 The template parameter MetallicNoiseSource allows another kind of "metallic \n noise" to be used, for results which are more similar to KR-55 or FM hi-hats.

Ported from pichenettes/eurorack/plaits/dsp/drums/hihat.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.30.2 Member Function Documentation

5.30.2.1 Init()

```
template<typename MetallicNoiseSource = SquareNoise, typename VCA = LinearVCA, bool resonance
= true>
void daisysp::HiHat< MetallicNoiseSource, VCA, resonance >::Init (
             float sample_rate )  [inline]
```

Initialize the module

Parameters

sample_rate	Audio engine sample rate

5.30.2.2 Process()

```
template<typename MetallicNoiseSource = SquareNoise, typename VCA = LinearVCA, bool resonance
= true>
float daisysp::HiHat< MetallicNoiseSource, VCA, resonance >::Process (
             bool trigger = false )  [inline]
```

Get the next sample

Parameters

trigger	Hit the hihat with true. Defaults to false.

5.30.2.3 SetAccent()

```
template<typename MetallicNoiseSource = SquareNoise, typename VCA = LinearVCA, bool resonance
= true>
void daisysp::HiHat< MetallicNoiseSource, VCA, resonance >::SetAccent (
             float accent )  [inline]
```

Set how much accent to use

Parameters

accent	Works 0-1.

5.30.2.4 SetDecay()

```
template<typename MetallicNoiseSource = SquareNoise, typename VCA = LinearVCA, bool resonance
= true>
void daisysp::HiHat< MetallicNoiseSource, VCA, resonance >::SetDecay (
             float decay )  [inline]
```

Set the length of the hihat decay

Parameters

decay	Works $>$ 0. Tuned for 0-1.

5.30.2.5 SetFreq()

```
template<typename MetallicNoiseSource = SquareNoise, typename VCA = LinearVCA, bool resonance
= true>
void daisysp::HiHat< MetallicNoiseSource, VCA, resonance >::SetFreq (
             float f0 )  [inline]
```

Set the hihat tone's root frequency

Parameters

f0	Freq in Hz

5.30.2.6 SetNoisiness()

```
template<typename MetallicNoiseSource = SquareNoise, typename VCA = LinearVCA, bool resonance
= true>
```

```
void daisysp::HiHat< MetallicNoiseSource, VCA, resonance >::SetNoisiness (
             float noisiness )  [inline]
```

Sets the mix between tone and noise

Parameters

snappy	1 = just noise. 0 = just tone.

5.30.2.7 SetSustain()

```
template<typename MetallicNoiseSource = SquareNoise, typename VCA = LinearVCA, bool resonance
= true>
void daisysp::HiHat< MetallicNoiseSource, VCA, resonance >::SetSustain (
             bool sustain )  [inline]
```

Make the hihat ring out infinitely.

Parameters

sustain	True = infinite sustain.

5.30.2.8 SetTone()

```
template<typename MetallicNoiseSource = SquareNoise, typename VCA = LinearVCA, bool resonance
= true>
void daisysp::HiHat< MetallicNoiseSource, VCA, resonance >::SetTone (
             float tone )  [inline]
```

Set the overall brightness / darkness of the hihat.

Parameters

tone	Works from 0-1.

5.30.2.9 Trig()

```
template<typename MetallicNoiseSource = SquareNoise, typename VCA = LinearVCA, bool resonance
= true>
void daisysp::HiHat< MetallicNoiseSource, VCA, resonance >::Trig ( )  [inline]
```

Trigger the hihat

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

- Source/Drums/hihat.h

5.31 daisysp::Jitter Class Reference

```
#include <jitter.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process ()
- void SetCpsMin (float cps_min)
- void SetCpsMax (float cps_max)
- void SetAmp (float amp)

5.31.1 Detailed Description

Randomly segmented line generator
Originally extracted from csound by Paul Batchelor.
Ported by Ben Sergentanis, June 2020

Author

Gabriel Maldonado

@year 1998

Location: Opcodes/uggab.c (csound)

5.31.2 Member Function Documentation

5.31.2.1 Init()

```
void Jitter::Init (
            float sample_rate )
```

Initializes Jitter module

Parameters

sample_rate	Audio engine sample rate

5.31.2.2 Process()

```
float Jitter::Process ( )
```

Get next floating point jitter sample

5.31.2.3 SetAmp()

```
void Jitter::SetAmp (
             float amp )
```

Set the amplitude of the jitter. Jitters fall from -amp to +amp

Parameters

amp	Jitter amplitude

5.31.2.4 SetCpsMax()

```
void Jitter::SetCpsMax (
             float cps_max )
```

Set the maximum speed of the jitter engine.

Parameters

cps_max	Maximum number of jitters per second.

5.31.2.5 SetCpsMin()

```
void Jitter::SetCpsMin (
             float cps_min )
```

Set the minimum speed of the jitter engine.

Parameters

cps_min	Number of new jitters per second

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

- Source/Utility/jitter.h
- Source/Utility/jitter.cpp

5.32 daisysp::Limiter Class Reference

```
#include <limiter.h>
```

Public Member Functions

- void Init ()
- void ProcessBlock (float ∗in, size_t size, float pre_gain)

5.32.1 Detailed Description

Simple Peak Limiter

This was extracted from pichenettes/stmlib.

Credit to pichenettes/Mutable Instruments

5.32.2 Member Function Documentation

5.32.2.1 Init()

```
void daisysp::Limiter::Init ( )
```

Initializes the Limiter instance.

5.32.2.2 ProcessBlock()

```
void daisysp::Limiter::ProcessBlock (
             float * in,
             size_t size,
             float pre_gain )
```

Processes a block of audio through the limiter.

Parameters

in	- pointer to a block of audio samples to be processed. The buffer is operated on directly.
size	- size of the buffer "in"
pre_gain	- amount of pre_gain applied to the signal.

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

- Source/Dynamics/limiter.h
- Source/Dynamics/limiter.cpp

5.33 daisysp::Line Class Reference

```
#include <line.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (uint8_t ∗finished)
- void Start (float start, float end, float dur)

5.33.1 Detailed Description

creates a Line segment signal

5.33.2 Member Function Documentation

5.33.2.1 Init()

```
void Line::Init (
             float sample_rate )
```

Initializes Line module.

5.33.2.2 Process()

```
float Line::Process (
             uint8_t * finished )
```

Processes Line segment. Returns one sample. value of finished will be updated to a 1, upon completion of the Line's trajectory.

5.33.2.3 Start()

```
void Line::Start (
             float start,
             float end,
             float dur )
```

Begin creation of Line.

Parameters

start	- beginning value
end	- ending value
dur	- duration in seconds of Line segment

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

- Source/Control/line.h

- Source/Control/line.cpp

5.34 daisysp::LinearVCA Class Reference

Linear type VCA.

```
#include <hihat.h>
```

Public Member Functions

- float **operator()** (float s, float gain)

5.34.1 Detailed Description

Linear type VCA.

Author

Ben Sergentanis

Date

Jan 2021 Ported from pichenettes/eurorack/plaits/dsp/drums/hihat.h
to an independent module.
Original code written by Emilie Gillet in 2016.

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

- Source/Drums/hihat.h

5.35 daisysp::Maytrig Class Reference

```
#include <maytrig.h>
```

Public Member Functions

- float Process (float prob)

5.35.1 Detailed Description

Probabilistic trigger module

Original author(s) : Paul Batchelor

Ported from soundpipe by Ben Sergentanis, May 2020

5.35.2 Member Function Documentation

5.35.2.1 Process()

```
float daisysp::Maytrig::Process (
            float prob )  [inline]
```

probabilistically generates triggers

Parameters

prob	(1 always returns true, 0 always false)

Returns

given a probability 0 to 1, returns true or false.

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

- Source/Utility/maytrig.h

5.36 daisysp::Metro Class Reference

```
#include <metro.h>
```

Public Member Functions

- void Init (float freq, float sample_rate)
- uint8_t Process ()
- void Reset ()
- void SetFreq (float freq)
- float GetFreq ()

5.36.1 Detailed Description

Creates a clock signal at a specific frequency.

5.36.2 Member Function Documentation

5.36.2.1 GetFreq()

```
float daisysp::Metro::GetFreq ( )  [inline]
```

Returns current value for frequency.

5.36.2.2 Init()

```
void Metro::Init (
             float freq,
             float sample_rate )
```

Initializes Metro module. Arguments:

- freq: frequency at which new clock signals will be generated Input Range:
- sample_rate: sample rate of audio engine Input range:

5.36.2.3 Process()

```
uint8_t Metro::Process ( )
```

checks current state of Metro object and updates state if necesary.

5.36.2.4 Reset()

```
void daisysp::Metro::Reset ( )  [inline]
```

resets phase to 0

5.36.2.5 SetFreq()

```
void Metro::SetFreq (
             float freq )
```

Sets frequency at which Metro module will run at.

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

- Source/Utility/metro.h
- Source/Utility/metro.cpp

5.37 daisysp::ModalVoice Class Reference

Simple modal synthesis voice with a mallet exciter: click -> LPF -> resonator.

```
#include <modalvoice.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (bool trigger=false)
- void SetSustain (bool sustain)
- void Trig ()
- void SetFreq (float freq)
- void SetAccent (float accent)
- void SetStructure (float structure)
- void SetBrightness (float brightness)
- void SetDamping (float damping)
- float GetAux ()

5.37.1 Detailed Description

Simple modal synthesis voice with a mallet exciter: click -> LPF -> resonator.

Author

Ben Sergentanis

Date

Jan 2021 The click can be replaced by continuous white noise.

Ported from pichenettes/eurorack/plaits/dsp/physical_modelling/modal_voice.h
and pichenettes/eurorack/plaits/dsp/physical_modelling/modal_voice.cc
to an independent module.
Original code written by Emilie Gillet in 2016.

5.37.2 Member Function Documentation

5.37.2.1 GetAux()

```
float ModalVoice::GetAux ( )
```

Get the raw excitation signal. Must call Process() first.

5.37.2.2 Init()

```
void ModalVoice::Init (
            float sample_rate )
```

Initialize the module

Parameters

sample_rate	Audio engine sample rate

5.37.2.3 Process()

```
float ModalVoice::Process (
             bool trigger = false )
```

Get the next sample

Parameters

trigger	Strike the resonator. Defaults to false.

5.37.2.4 SetAccent()

```
void ModalVoice::SetAccent (
             float accent )
```

Hit the resonator a bit harder.

Parameters

accent	Works 0-1.

5.37.2.5 SetBrightness()

```
void ModalVoice::SetBrightness (
             float brightness )
```

Set the brighness of the resonator, and the noise density.

Parameters

brightness	Works best 0-1

5.37.2.6 SetDamping()

```
void ModalVoice::SetDamping (
             float damping )
```

How long the resonant body takes to decay.

Parameters

damping	Works best 0-1

5.37.2.7 SetFreq()

```
void ModalVoice::SetFreq (
             float freq )
```

Set the resonator root frequency.

Parameters

freq	Frequency in Hz.

5.37.2.8 SetStructure()

```
void ModalVoice::SetStructure (
             float structure )
```

Changes the general charater of the resonator (stiffness, brightness)

Parameters

structure	Works best from 0-1

5.37.2.9 SetSustain()

```
void ModalVoice::SetSustain (
             bool sustain )
```

Continually excite the resonator with noise.

Parameters

sustain	True turns on the noise.

5.37.2.10 Trig()

```
void ModalVoice::Trig ( )
```

Strike the resonator.

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

- Source/PhysicalModeling/modalvoice.h
- Source/PhysicalModeling/modalvoice.cpp

5.38 daisysp::Mode Class Reference

```
#include <mode.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float in)
- void Clear ()
- void SetFreq (float freq)
- void SetQ (float q)

5.38.1 Detailed Description

Resonant Modal Filter

Extracted from soundpipe to work as a Daisy Module,

originally extracted from csound by Paul Batchelor.

Original Author(s): Francois Blanc, Steven Yi

Year: 2001

Location: Opcodes/biquad.c (csound)

5.38.2 Member Function Documentation

5.38.2.1 Clear()

```
void Mode::Clear ( )
```

Clears the filter, returning the output to 0.0

5.38.2.2 Init()

```
void Mode::Init (
            float sample_rate )
```

Initializes the instance of the module. sample_rate: frequency of the audio engine in Hz

5.38.2.3 Process()

```
float Mode::Process (
            float in )
```

Processes one input sample through the filter, and returns the output.

5.38.2.4 SetFreq()

```
void daisysp::Mode::SetFreq (
            float freq ) [inline]
```

Sets the resonant frequency of the modal filter. Range: Any frequency such that sample_rate / freq $<$ PI (about 15.2kHz at 48kHz)

5.38.2.5 SetQ()

```
void daisysp::Mode::SetQ (
            float q ) [inline]
```

Sets the quality factor of the filter. Range: Positive Numbers (Good values range from 70 to 1400)

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

- Source/Filters/mode.h
- Source/Filters/mode.cpp

5.39 daisysp::MoogLadder Class Reference

```
#include <moogladder.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float in)
- void SetFreq (float freq)
- void SetRes (float res)

5.39.1 Detailed Description

Moog ladder filter module

Ported from soundpipe

Original author(s) : Victor Lazzarini, John ffitch (fast tanh), Bob Moog

5.39.2 Member Function Documentation

5.39.2.1 Init()

```
void MoogLadder::Init (
             float sample_rate )
```

Initializes the MoogLadder module. sample_rate - The sample rate of the audio engine being run.

5.39.2.2 Process()

```
float MoogLadder::Process (
             float in )
```

Processes the lowpass filter

5.39.2.3 SetFreq()

```
void daisysp::MoogLadder::SetFreq (
             float freq )  [inline]
```

Sets the cutoff frequency or half-way point of the filter. Arguments

- freq - frequency value in Hz. Range: Any positive value.

5.39.2.4 SetRes()

```
void daisysp::MoogLadder::SetRes (
             float res )  [inline]
```

Sets the resonance of the filter.

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

- Source/Filters/moogladder.h
- Source/Filters/moogladder.cpp

5.40 daisysp::NlFilt Class Reference

```
#include <nlfilt.h>
```

Public Member Functions

- void Init ()
- void ProcessBlock (float ∗in, float ∗out, size_t size)
- void SetCoefficients (float a, float b, float d, float C, float L)
- void SetA (float a)
- void SetB (float b)
- void SetD (float d)
- void SetC (float C)
- void SetL (float L)

5.40.1 Detailed Description

Non-linear filter

port by: Stephen Hensley, December 2019

The four 5-coefficients: a, b, d, C, and L are used to configure different filter types.

Structure for Dobson/Fitch nonlinear filter

Revised Formula from Risto Holopainen 12 Mar 2004

```
Y{n} =tanh(a Y{n-1} + b Y{n-2} + d Y^2{n-L} + X{n} - C)
```

Though traditional filter types can be made, the effect will always respond differently to different input.

This Source is a heavily modified version of the original source from Csound.

Todo make this work on a single sample instead of just on blocks at a time.

5.40.2 Member Function Documentation

5.40.2.1 Init()

```
void NlFilt::Init ( )
```

Initializes the NlFilt object.

5.40.2.2 ProcessBlock()

```
void NlFilt::ProcessBlock (
            float * in,
            float * out,
            size_t size )
```

Process the array pointed to by ∗in and updates the output to ∗out; This works on a block of audio at once, the size of which is set with the size.

5.40.2.3 SetA()

```
void daisysp::NlFilt::SetA (
            float a ) [inline]
```

Set Coefficient a

5.40.2.4 SetB()

```
void daisysp::NlFilt::SetB (
            float b ) [inline]
```

Set Coefficient b

5.40.2.5 SetC()

```
void daisysp::NlFilt::SetC (
            float C ) [inline]
```

Set Coefficient C

5.40.2.6 SetCoefficients()

```
void daisysp::NlFilt::SetCoefficients (
            float a,
            float b,
            float d,
            float C,
            float L ) [inline]
```

inputs these are the five coefficients for the filter.

5.40.2.7 SetD()

```
void daisysp::NlFilt::SetD (
            float d )  [inline]
```

Set Coefficient d

5.40.2.8 SetL()

```
void daisysp::NlFilt::SetL (
            float L )  [inline]
```

Set Coefficient L

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

- Source/Filters/nlfilt.h
- Source/Filters/nlfilt.cpp

5.41 daisysp::Oscillator Class Reference

```
#include <oscillator.h>
```

Public Types

- enum {
  WAVE_SIN , **WAVE_TRI** , **WAVE_SAW** , **WAVE_RAMP** ,
  WAVE_SQUARE , **WAVE_POLYBLEP_TRI** , **WAVE_POLYBLEP_SAW** , **WAVE_POLYBLEP_SQUARE** ,
  WAVE_LAST }

Public Member Functions

- void Init (float sample_rate)
- void SetFreq (const float f)
- void SetAmp (const float a)
- void SetWaveform (const uint8_t wf)
- bool IsEOR ()
- bool IsEOC ()
- bool IsRising ()
- bool IsFalling ()
- float Process ()
- void PhaseAdd (float _phase)
- void Reset (float _phase=0.0f)

5.41.1 Detailed Description

Synthesis of several waveforms, including polyBLEP bandlimited waveforms.

5.41.2 Member Enumeration Documentation

5.41.2.1 anonymous enum

```
anonymous enum
```

Choices for output waveforms, POLYBLEP are appropriately labeled. Others are naive forms.

5.41.3 Member Function Documentation

5.41.3.1 Init()

```
void daisysp::Oscillator::Init (
             float sample_rate )  [inline]
```

Initializes the Oscillator

Parameters

sample_rate	- sample rate of the audio engine being run, and the frequency that the Process function will be called.

Defaults:

- freq_ = 100 Hz
- amp_ = 0.5
- waveform_ = sine wave.

5.41.3.2 IsEOC()

```
bool daisysp::Oscillator::IsEOC ( )  [inline]
```

Returns true if cycle is at end of cycle. Set during call to Process.

5.41.3.3 IsEOR()

```
bool daisysp::Oscillator::IsEOR ( )  [inline]
```

Returns true if cycle is at end of rise. Set during call to Process.

5.41.3.4 IsFalling()

```
bool daisysp::Oscillator::IsFalling ( )  [inline]
```

Returns true if cycle falling.

5.41.3.5 IsRising()

```
bool daisysp::Oscillator::IsRising ( )  [inline]
```

Returns true if cycle rising.

5.41.3.6 PhaseAdd()

```
void daisysp::Oscillator::PhaseAdd (
             float _phase )  [inline]
```

Adds a value 0.0-1.0 (mapped to 0.0-TWO_PI) to the current phase. Useful for PM and "FM" synthesis.

5.41.3.7 Process()

```
float Oscillator::Process ( )
```

Processes the waveform to be generated, returning one sample. This should be called once per sample period.

5.41.3.8 Reset()

```
void daisysp::Oscillator::Reset (
             float _phase = 0.0f )  [inline]
```

Resets the phase to the input argument. If no argumeNt is present, it will reset phase to 0.0;

5.41.3.9 SetAmp()

```
void daisysp::Oscillator::SetAmp (
             const float a )  [inline]
```

Sets the amplitude of the waveform.

5.41.3.10 SetFreq()

```
void daisysp::Oscillator::SetFreq (
             const float f )  [inline]
```

Changes the frequency of the Oscillator, and recalculates phase increment.

5.41.3.11 SetWaveform()

```
void daisysp::Oscillator::SetWaveform (
             const uint8_t wf )  [inline]
```

Sets the waveform to be synthesized by the Process() function.

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

- Source/Synthesis/oscillator.h
- Source/Synthesis/oscillator.cpp

5.42 daisysp::OscillatorBank Class Reference

Oscillator Bank module.

```
#include <oscillatorbank.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process ()
- void SetFreq (float freq)
- void SetAmplitudes (const float ∗amplitudes)
- void SetSingleAmp (float amp, int idx)
- void SetGain (float gain)

5.42.1 Detailed Description

Oscillator Bank module.

Author

Ben Sergentanis

Date

Dec 2020 A mixture of 7 sawtooth and square waveforms in the style of divide-down organs

Ported from pichenettes/eurorack/plaits/dsp/oscillator/string_synth_oscillator.h

to an independent module.
Original code written by Emilie Gillet in 2016.

5.42.2 Member Function Documentation

5.42.2.1 Init()

```
void OscillatorBank::Init (
             float sample_rate )
```

Init string synth module

Parameters

sample_rate	Audio engine sample rate

5.42.2.2 Process()

```
float OscillatorBank::Process ( )
```

Get next floating point sample

5.42.2.3 SetAmplitudes()

```
void OscillatorBank::SetAmplitudes (
            const float * amplitudes )
```

Set amplitudes of 7 oscillators. 0-6 are Saw 8', Square 8', Saw 4', Square 4', Saw 2', Square 2', Saw 1'

Parameters

amplitudes	array of 7 floating point amplitudes. Must sum to 1.

5.42.2.4 SetFreq()

```
void OscillatorBank::SetFreq (
            float freq )
```

Set oscillator frequency (8' oscillator)

Parameters

freq	Frequency in Hz

5.42.2.5 SetGain()

```
void OscillatorBank::SetGain (
            float gain )
```

Set overall gain.

Parameters

gain	Gain to set. 0-1.

5.42.2.6 SetSingleAmp()

```
void OscillatorBank::SetSingleAmp (
             float amp,
             int idx )
```

Set a single amplitude

Parameters

amp	Amplitude to set.
idx	Which wave's amp to set

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

- Source/Synthesis/oscillatorbank.h
- Source/Synthesis/oscillatorbank.cpp

5.43 daisysp::Overdrive Class Reference

Distortion / Overdrive Module.

```
#include <overdrive.h>
```

Public Member Functions

- void Init ()
- float Process (float in)
- void SetDrive (float drive)

5.43.1 Detailed Description

Distortion / Overdrive Module.

Author

Ported by Ben Sergentanis

Date

Jan 2021 Ported from pichenettes/eurorack/plaits/dsp/fx/overdrive.h
to an independent module.
Original code written by Emilie Gillet in 2014.

5.43.2 Member Function Documentation

5.43.2.1 Init()

```
void daisysp::Overdrive::Init ( )
```

Initializes the module with 0 gain

5.43.2.2 Process()

```
float daisysp::Overdrive::Process (
             float in )
```

Get the next sample

Parameters

in	Input to be overdriven

5.43.2.3 SetDrive()

```
void daisysp::Overdrive::SetDrive (
             float drive )
```

Set the amount of drive

Parameters

drive	Works from 0-1

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

- Source/Effects/overdrive.h
- Source/Effects/overdrive.cpp

5.44 daisysp::Particle Class Reference

Random impulse train processed by a resonant filter.

```
#include <particle.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process ()
- float GetNoise ()
- void SetFreq (float frequency)
- void SetResonance (float resonance)
- void SetRandomFreq (float freq)
- void SetDensity (float density)
- void SetGain (float gain)
- void SetSpread (float spread)
- void SetSync (bool sync)

5.44.1 Detailed Description

Random impulse train processed by a resonant filter.

Author

Ported by Ben Sergentanis

Date

Jan 2021 Noise processed by a sample and hold running at a target frequency.

Ported from pichenettes/eurorack/plaits/dsp/noise/particle.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.44.2 Member Function Documentation

5.44.2.1 GetNoise()

```
float Particle::GetNoise ( )
```

Get the raw noise output. Must call Process() first.

5.44.2.2 Init()

```
void Particle::Init (
             float sample_rate )
```

Initialize the module

Parameters

sample_rate	Audio engine sample rate.

5.44.2.3 Process()

```
float Particle::Process ( )
```

Get the next sample

5.44.2.4 SetDensity()

```
void Particle::SetDensity (
             float density )
```

Noise density

Parameters

Works	0-1.

5.44.2.5 SetFreq()

```
void Particle::SetFreq (
             float frequency )
```

Set the resonant filter frequency

Parameters

freq	Frequency in Hz

5.44.2.6 SetGain()

```
void Particle::SetGain (
             float gain )
```

Overall module gain

Parameters

Works	0-1.

5.44.2.7 SetRandomFreq()

```
void Particle::SetRandomFreq (
             float freq )
```

How often to randomize filter frequency

Parameters

freq	Frequency in Hz.

5.44.2.8 SetResonance()

```
void Particle::SetResonance (
             float resonance )
```

Set the filter resonance

Parameters

resonance	Works 0-1

5.44.2.9 SetSpread()

```
void Particle::SetSpread (
             float spread )
```

How much to randomize the set filter frequency.

Parameters

spread	Works over positive numbers.

5.44.2.10 SetSync()

```
void Particle::SetSync (
            bool sync )
```

Force randomize the frequency.

Parameters

sync	True to randomize freq.

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

- Source/Noise/particle.h
- Source/Noise/particle.cpp

5.45 daisysp::Phasor Class Reference

```
#include <phasor.h>
```

Public Member Functions

- void Init (float sample_rate, float freq, float initial_phase)
- void Init (float sample_rate, float freq)
- void Init (float sample_rate)
- float Process ()
- void SetFreq (float freq)
- float GetFreq ()

5.45.1 Detailed Description

Generates a normalized signal moving from 0-1 at the specified frequency.

Todo Selecting which channels should be initialized/included in the sequence conversion.

Setup a similar start function for an external mux, but that seems outside the scope of this file.

5.45.2 Member Function Documentation

5.45.2.1 GetFreq()

```
float daisysp::Phasor::GetFreq ( )  [inline]
```

Returns current frequency value in Hz

5.45.2.2 Init() [1/3]

```
void daisysp::Phasor::Init (
             float sample_rate )  [inline]
```

Initialize phasor with samplerate

5.45.2.3 Init() [2/3]

```
void daisysp::Phasor::Init (
             float sample_rate,
             float freq )  [inline]
```

Initialize phasor with samplerate and freq

5.45.2.4 Init() [3/3]

```
void daisysp::Phasor::Init (
             float sample_rate,
             float freq,
             float initial_phase )  [inline]
```

Initializes the Phasor module sample rate, and freq are in Hz initial phase is in radians Additional Init functions have defaults when arg is not specified:

- phs = 0.0f
- freq = 1.0f

5.45.2.5 Process()

```
float Phasor::Process ( )
```

processes Phasor and returns current value

5.45.2.6 SetFreq()

```
void Phasor::SetFreq (
             float freq )
```

Sets frequency of the Phasor in Hz

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

- Source/Control/phasor.h
- Source/Control/phasor.cpp

5.46 daisysp::PitchShifter Class Reference

```
#include <pitchshifter.h>
```

Public Member Functions

- void Init (float sr)
- float Process (float &in)
- void SetTransposition (const float &transpose)
- void SetDelSize (uint32_t size)
- void SetFun (float f)

5.46.1 Detailed Description

time-domain pitchshifter

Author: shensley

Based on "Pitch Shifting" from ucsd.edu

t = 1 - ((s ∗f) / R)

where: s is the size of the delay f is the frequency of the lfo r is the sample_rate

solving for t = 12.0 f = (12 - 1) ∗ 48000 / SHIFT_BUFFER_SIZE;

Todo
- move hash_xs32 and myrand to dsp.h and give appropriate names

5.46.2 Member Function Documentation

5.46.2.1 Init()

```
void daisysp::PitchShifter::Init (
             float sr )  [inline]
```

Initialize pitch shifter

5.46.2.2 Process()

```
float daisysp::PitchShifter::Process (
             float & in )  [inline]
```

process pitch shifter

5.46.2.3 SetDelSize()

```
void daisysp::PitchShifter::SetDelSize (
             uint32_t size )  [inline]
```

sets delay size changing the timbre of the pitchshifting

5.46.2.4 SetFun()

```
void daisysp::PitchShifter::SetFun (
             float f )  [inline]
```

sets an amount of internal random modulation, kind of sounds like tape-flutter

5.46.2.5 SetTransposition()

```
void daisysp::PitchShifter::SetTransposition (
             const float & transpose )  [inline]
```

sets transposition in semitones

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

- Source/Effects/pitchshifter.h

5.47 daisysp::Pluck Class Reference

```
#include <pluck.h>
```

Public Member Functions

- void Init (float sample_rate, float ∗buf, int32_t npt, int32_t mode)
- float Process (float &trig)
- void SetAmp (float amp)
- void SetFreq (float freq)
- void SetDecay (float decay)
- void SetDamp (float damp)
- void SetMode (int32_t mode)
- float GetAmp ()
- float GetFreq ()
- float GetDecay ()
- float GetDamp ()
- int32_t GetMode ()

5.47.1 Detailed Description

Produces a naturally decaying plucked string or drum sound based on the Karplus-Strong algorithms.

Ported from soundpipe to DaisySP

This code was originally extracted from the Csound opcode "pluck"

Original Author(s): Barry Vercoe, John ffitch Year: 1991

Location: OOps/ugens4.c

5.47.2 Member Function Documentation

5.47.2.1 GetAmp()

```
float daisysp::Pluck::GetAmp ( )  [inline]
```

Returns the current value for amp.

5.47.2.2 GetDamp()

```
float daisysp::Pluck::GetDamp ( )  [inline]
```

Returns the current value for damp.

5.47.2.3 GetDecay()

```
float daisysp::Pluck::GetDecay ( )  [inline]
```

Returns the current value for decay.

5.47.2.4 GetFreq()

```
float daisysp::Pluck::GetFreq ( )  [inline]
```

Returns the current value for freq.

5.47.2.5 GetMode()

```
int32_t daisysp::Pluck::GetMode ( )  [inline]
```

Returns the current value for mode.

5.47.2.6 Init()

```
void Pluck::Init (
             float sample_rate,
             float * buf,
             int32_t npt,
             int32_t mode )
```

Initializes the Pluck module.

```
\param sample_rate: Sample rate of the audio engine being run.
\param buf: buffer used as an impulse when triggering the Pluck algorithm
\param npt: number of elementes in buf.
\param mode: Sets the mode of the algorithm.
```

5.47.2.7 Process()

```
float Pluck::Process (
             float & trig )
```

Processes the waveform to be generated, returning one sample. This should be called once per sample period.

5.47.2.8 SetAmp()

```
void daisysp::Pluck::SetAmp (
             float amp )  [inline]
```

Sets the amplitude of the output signal. Input range: 0-1?

5.47.2.9 SetDamp()

```
void daisysp::Pluck::SetDamp (
             float damp )  [inline]
```

Sets the dampening factor applied by the filter (based on PLUCK_MODE) Input range: 0-1

5.47.2.10 SetDecay()

```
void daisysp::Pluck::SetDecay (
             float decay )  [inline]
```

Sets the time it takes for a triggered note to end in seconds. Input range: 0-1

5.47.2.11 SetFreq()

```
void daisysp::Pluck::SetFreq (
             float freq )  [inline]
```

Sets the frequency of the output signal in Hz. Input range: Any positive value

5.47.2.12 SetMode()

```
void daisysp::Pluck::SetMode (
             int32_t mode )  [inline]
```

Sets the mode of the algorithm.

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

- Source/PhysicalModeling/pluck.h
- Source/PhysicalModeling/pluck.cpp

5.48 daisysp::PolyPluck< num_voices > Class Template Reference

```
#include <PolyPluck.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float &trig, float note)
- void SetDecay (float p)

5.48.1 Detailed Description

template<size_t num_voices>
class daisysp::PolyPluck< num_voices >

Simplified Pseudo-Polyphonic Pluck Voice

Template Based Pluck Voice, with configurable number of voices and simple pseudo-polyphony.

DC Blocking included to prevent biases from causing unwanted saturation distortion.

Author∗∗: shensley

Date Added∗∗: March 2020

5.48.2 Member Function Documentation

5.48.2.1 Init()

```
template<size_t num_voices>
void daisysp::PolyPluck< num_voices >::Init (
             float sample_rate )  [inline]
```

Initializes the PolyPluck instance.

Parameters

sample_rate	rate in Hz that the Process() function will be called.

5.48.2.2 Process()

```
template<size_t num_voices>
float daisysp::PolyPluck< num_voices >::Process (
            float & trig,
            float note ) [inline]
```

Process function, synthesizes and sums the output of all voices, triggering a new voice with frequency of MIDI note number when trig $>$ 0.

Parameters

trig	value by reference of trig. When trig $>$ 0 a the next voice will be triggered, and trig will be set to 0.
note	MIDI note number for the active_voice.

5.48.2.3 SetDecay()

```
template<size_t num_voices>
void daisysp::PolyPluck< num_voices >::SetDecay (
            float p ) [inline]
```

Sets the decay coefficients of the pluck voices.

Parameters

p	expects 0.0-1.0 input.

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

- Source/PhysicalModeling/PolyPluck.h

5.49 daisysp::Port Class Reference

```
#include <port.h>
```

Public Member Functions

- void Init (float sample_rate, float htime)
- float Process (float in)
- void SetHtime (float htime)
- float GetHtime ()

5.49.1 Detailed Description

Applies portamento to an input signal.

At each new step value, the input is low-pass filtered to move towards that value at a rate determined by ihtim. ihtim is the half-time of the function (in seconds), during which the curve will traverse half the distance towards the new value, then half as much again, etc., theoretically never reaching its asymptote.

This code has been ported from Soundpipe to DaisySP by Paul Batchelor.

The Soundpipe module was extracted from the Csound opcode "portk".

Original Author(s): Robbin Whittle, John ffitch

Year: 1995, 1998

Location: Opcodes/biquad.c

5.49.2 Member Function Documentation

5.49.2.1 GetHtime()

```
float daisysp::Port::GetHtime ( )  [inline]
```

returns current value of htime

5.49.2.2 Init()

```
void Port::Init (
             float sample_rate,
             float htime )
```

Initializes Port module

Parameters

sample_rate	sample rate of audio engine
htime	half-time of the function, in seconds.

5.49.2.3 Process()

```
float Port::Process (
             float in )
```

Applies portamento to input signal and returns processed signal.

Returns

slewed output signal

5.49.2.4 SetHtime()

```
void daisysp::Port::SetHtime (
             float htime )  [inline]
```

Sets htime

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

- Source/Utility/port.h
- Source/Utility/port.cpp

5.50 daisysp::Resonator Class Reference

Resonant Body Simulation.

```
#include <resonator.h>
```

Public Member Functions

- void Init (float position, int resolution, float sample_rate)
- float Process (const float in)
- void SetFreq (float freq)
- void SetStructure (float structure)
- void SetBrightness (float brightness)
- void SetDamping (float damping)

5.50.1 Detailed Description

Resonant Body Simulation.

Author

Ported by Ben Sergentanis

Date

Jan 2021 Ported from pichenettes/eurorack/plaits/dsp/physical_modelling/resonator.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.50.2 Member Function Documentation

5.50.2.1 Init()

```
void Resonator::Init (
             float position,
             int resolution,
             float sample_rate )
```

Initialize the module

Parameters

position	Offset the phase of the amplitudes. 0-1
resolution	Quality vs speed scalar
sample_rate	Samplerate of the audio engine being run.

5.50.2.2 Process()

```
float Resonator::Process (
             const float in )
```

Get the next sample_rate

Parameters

in	The signal to excited the resonant body

5.50.2.3 SetBrightness()

```
void Resonator::SetBrightness (
             float brightness )
```

Set the brighness of the resonator

Parameters

brightness	Works best 0-1

5.50.2.4 SetDamping()

```
void Resonator::SetDamping (
             float damping )
```

How long the resonant body takes to decay.

Parameters

damping	Works best 0-1

5.50.2.5 SetFreq()

```
void Resonator::SetFreq (
             float freq )
```

Resonator frequency.

Parameters

freq	Frequency in Hz.

5.50.2.6 SetStructure()

```
void Resonator::SetStructure (
             float structure )
```

Changes the general charater of the resonator (stiffness, brightness)

Parameters

structure	Works best from 0-1

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

- Source/PhysicalModeling/resonator.h
- Source/PhysicalModeling/resonator.cpp

5.51 daisysp::ResonatorSvf< batch_size > Class Template Reference

SVF for use in the Resonator Class
.

```
#include <resonator.h>
```

Public Types

- enum **FilterMode** { **LOW_PASS** , **BAND_PASS** , **BAND_PASS_NORMALIZED** , **HIGH_PASS** }

Public Member Functions

- void **Init** ()
- template<FilterMode mode, bool add>
  void **Process** (const float ∗f, const float ∗q, const float ∗gain, const float in, float ∗out)

5.51.1 Detailed Description

template<int batch_size>
class daisysp::ResonatorSvf< batch_size >

SVF for use in the Resonator Class
.

Author

Ported by Ben Sergentanis

Date

Jan 2021 Ported from pichenettes/eurorack/plaits/dsp/physical_modelling/resonator.h
to an independent module.
Original code written by Emilie Gillet in 2016.

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

- Source/PhysicalModeling/resonator.h

5.52 daisysp::ReverbSc Class Reference

```
#include <reverbsc.h>
```

Public Member Functions

- int Init (float sample_rate)
- int Process (const float &in1, const float &in2, float ∗out1, float ∗out2)
- void SetFeedback (const float &fb)
- void SetLpFreq (const float &freq)

5.52.1 Detailed Description

Stereo Reverb

Reverb SC: Ported from csound/soundpipe

Original author(s): Sean Costello, Istvan Varga

Year: 1999, 2005

Ported to soundpipe by: Paul Batchelor

Ported by: Stephen Hensley

5.52.2 Member Function Documentation

5.52.2.1 Init()

```
int ReverbSc::Init (
             float sample_rate )
```

Initializes the reverb module, and sets the sample_rate at which the Process function will be called. Returns 0 if all good, or 1 if it runs out of delay times exceed maximum allowed.

5.52.2.2 Process()

```
int ReverbSc::Process (
             const float & in1,
             const float & in2,
             float * out1,
             float * out2 )
```

Process the input through the reverb, and updates values of out1, and out2 with the new processed signal.

5.52.2.3 SetFeedback()

```
void daisysp::ReverbSc::SetFeedback (
             const float & fb )  [inline]
```

controls the reverb time. reverb tail becomes infinite when set to 1.0

Parameters

fb	- sets reverb time. range: 0.0 to 1.0

5.52.2.4 SetLpFreq()

```
void daisysp::ReverbSc::SetLpFreq (
             const float & freq )  [inline]
```

controls the internal dampening filter's cutoff frequency.

Parameters

freq	- low pass frequency. range: 0.0 to sample_rate / 2

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

- Source/Effects/reverbsc.h
- Source/Effects/reverbsc.cpp

5.53 daisysp::ReverbScDl Struct Reference

```
#include <reverbsc.h>
```

Public Attributes

- int write_pos
- int buffer_size
- int read_pos
- int read_pos_frac
- int read_pos_frac_inc
- int dummy
- int seed_val
- int rand_line_cnt
- float filter_state
- float ∗ buf

5.53.1 Detailed Description

Delay line for internal reverb use

5.53.2 Member Data Documentation

5.53.2.1 buf

```
float* daisysp::ReverbScDl::buf
```

buffer ptr

5.53.2.2 buffer_size

```
int daisysp::ReverbScDl::buffer_size
```

buffer size

5.53.2.3 dummy

```
int daisysp::ReverbScDl::dummy
```

dummy var

5.53.2.4 filter_state

```
float daisysp::ReverbScDl::filter_state
```

state of filter

5.53.2.5 rand_line_cnt

```
int daisysp::ReverbScDl::rand_line_cnt
```

number of random lines

5.53.2.6 read_pos

```
int daisysp::ReverbScDl::read_pos
```

read position

5.53.2.7 read_pos_frac

```
int daisysp::ReverbScDl::read_pos_frac
```

fractional component of read pos

5.53.2.8 read_pos_frac_inc

```
int daisysp::ReverbScDl::read_pos_frac_inc
```

increment for fractional

5.53.2.9 seed_val

```
int daisysp::ReverbScDl::seed_val
```

randseed

5.53.2.10 write_pos

```
int daisysp::ReverbScDl::write_pos
```

write position

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

- Source/Effects/reverbsc.h

5.54 daisysp::RingModNoise Class Reference

Ring mod style metallic noise generator.

```
#include <hihat.h>
```

Public Member Functions

- void **Init** (float sample_rate)
- float **Process** (float f0)

5.54.1 Detailed Description

Ring mod style metallic noise generator.

Author

Ben Sergentanis

Date

Jan 2021 Ported from pichenettes/eurorack/plaits/dsp/drums/hihat.h
to an independent module.
Original code written by Emilie Gillet in 2016.

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

- Source/Drums/hihat.h
- Source/Drums/hihat.cpp

5.55 daisysp::SampleHold Class Reference

```
#include <samplehold.h>
```

Public Types

- enum **Mode** { **MODE_SAMPLE_HOLD** , **MODE_TRACK_HOLD** , **MODE_LAST** }

Public Member Functions

- float Process (bool trigger, float input, Mode mode=MODE_SAMPLE_HOLD)

5.55.1 Detailed Description

Dual track and hold / Sample and hold module.
Ported from soundpipe by Ben Sergentanis, June 2020.

Author

Paul Batchelor

Date

2015

5.55.2 Member Function Documentation

5.55.2.1 Process()

```
float daisysp::SampleHold::Process (
             bool trigger,
             float input,
             Mode mode = MODE_SAMPLE_HOLD )  [inline]
```

Process the next sample. Both sample and track and hold are run in parallel

Parameters

trigger	Trigger the sample/track and hold
input	Signal to be sampled/tracked and held
mode	Whether to output the tracked or sampled values.

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

- Source/Utility/samplehold.h

5.56 daisysp::SampleRateReducer Class Reference

Sample rate reducer.

`#include <samplerateredducer.h>`

Public Member Functions

- void Init ()
- float Process (float in)
- void SetFreq (float frequency)

5.56.1 Detailed Description

Sample rate reducer.

Author

Ben Sergentanis

Date

Jan 2021 Ported from pichenettes/eurorack/plaits/dsp/fx/sample_rate_reducer.h
to an independent module.
Original code written by Emilie Gillet in 2014.

5.56.2 Member Function Documentation

5.56.2.1 Init()

```
void SampleRateReducer::Init ( )
```

Initialize the module

5.56.2.2 Process()

```
float SampleRateReducer::Process (
             float in )
```

Get the next floating point sample

Parameters

in	Sample to be processed.

5.56.2.3 SetFreq()

```
void SampleRateReducer::SetFreq (
             float frequency )
```

Set the new sample rate.

Parameters

Works	over 0-1. 1 is full quality, .5 is half sample rate, etc.

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

- Source/Effects/sampleratereducer.h
- Source/Effects/sampleratereducer.cpp

5.57 daisysp::SmoothRandomGenerator Class Reference

Smooth random generator for internal modulation.
.

```
#include <smooth_random.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process ()
- void SetFreq (float freq)

5.57.1 Detailed Description

Smooth random generator for internal modulation.
.

Author

Ported by Ben Sergentanis

Date

Jan 2021 Ported from pichenettes/eurorack/plaits/dsp/noise/smooth_random_generator.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.57.2 Member Function Documentation

5.57.2.1 Init()

```
void daisysp::SmoothRandomGenerator::Init (
            float sample_rate )  [inline]
```

Initialize the module

Parameters

sample_rate	Audio engine sample rate.

5.57.2.2 Process()

```
float daisysp::SmoothRandomGenerator::Process ( )  [inline]
```

Get the next float. Ranges from -1 to 1.

5.57.2.3 SetFreq()

```
void daisysp::SmoothRandomGenerator::SetFreq (
             float freq )  [inline]
```

How often to slew to a new random value

Parameters

freq	Rate in Hz

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

- Source/Utility/smooth_random.h

5.58 daisysp::SquareNoise Class Reference

808 style "metallic noise" with 6 square oscillators.

```
#include <hihat.h>
```

Public Member Functions

- void **Init** (float sample_rate)
- float **Process** (float f0)

5.58.1 Detailed Description

808 style "metallic noise" with 6 square oscillators.

Author

Ben Sergentanis

Date

Jan 2021 Ported from pichenettes/eurorack/plaits/dsp/drums/hihat.h
to an independent module.
Original code written by Emilie Gillet in 2016.

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

- Source/Drums/hihat.h
- Source/Drums/hihat.cpp

5.59 daisysp::String Class Reference

Comb filter / KS string.

```
#include <KarplusString.h>
```

Public Member Functions

- void Init (float sample_rate)
- void Reset ()
- float Process (const float in)
- void SetFreq (float freq)
- void SetNonLinearity (float non_linearity_amount)
- void SetBrightness (float brightness)
- void SetDamping (float damping)

5.59.1 Detailed Description

Comb filter / KS string.

Author

Ben Sergentanis

Date

Jan 2021 "Lite" version of the implementation used in Rings

Ported from pichenettes/eurorack/plaits/dsp/oscillator/formant_oscillator.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.59.2 Member Function Documentation

5.59.2.1 Init()

```
void String::Init (
            float sample_rate )
```

Initialize the module.

Parameters

sample_rate	Audio engine sample rate

5.59.2.2 Process()

```
float String::Process (
             const float in )
```

Get the next floating point sample

Parameters

in	Signal to excite the string.

5.59.2.3 Reset()

```
void String::Reset ( )
```

Clear the delay line

5.59.2.4 SetBrightness()

```
void String::SetBrightness (
             float brightness )
```

Set the string's overall brightness

Parameters

Works	0-1.

5.59.2.5 SetDamping()

```
void String::SetDamping (
             float damping )
```

Set the string's decay time.

Parameters

damping	Works 0-1.

5.59.2.6 SetFreq()

```
void String::SetFreq (
            float freq )
```

Set the string frequency.

Parameters

freq	Frequency in Hz

5.59.2.7 SetNonLinearity()

```
void String::SetNonLinearity (
            float non_linearity_amount )
```

Set the string's behavior.

Parameters

-1	to 0 is curved bridge, 0 to 1 is dispersion.

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

- Source/PhysicalModeling/KarplusString.h
- Source/PhysicalModeling/KarplusString.cpp

5.60 daisysp::StringVoice Class Reference

Extended Karplus-Strong, with all the niceties from Rings.

```
#include <stringvoice.h>
```

Public Member Functions

- void Init (float sample_rate)
- void Reset ()
- float Process (bool trigger=false)
- void SetSustain (bool sustain)
- void Trig ()
- void SetFreq (float freq)
- void SetAccent (float accent)
- void SetStructure (float structure)
- void SetBrightness (float brightness)
- void SetDamping (float damping)
- float GetAux ()

5.60.1 Detailed Description

Extended Karplus-Strong, with all the niceties from Rings.

Author

Ben Sergentanis

Date

Jan 2021 Ported from pichenettes/eurorack/plaits/dsp/physical_modelling/string_voice.h
and pichenettes/eurorack/plaits/dsp/physical_modelling/string_voice.cc
to an independent module.
Original code written by Emilie Gillet in 2016.

5.60.2 Member Function Documentation

5.60.2.1 GetAux()

```
float StringVoice::GetAux ( )
```

Get the raw excitation signal. Must call Process() first.

5.60.2.2 Init()

```
void StringVoice::Init (
             float sample_rate )
```

Initialize the module

Parameters

sample_rate	Audio engine sample rate

5.60.2.3 Process()

```
float StringVoice::Process (
            bool trigger = false )
```

Get the next sample

Parameters

trigger	Strike the string. Defaults to false.

5.60.2.4 Reset()

```
void StringVoice::Reset ( )
```

Reset the string oscillator

5.60.2.5 SetAccent()

```
void StringVoice::SetAccent (
            float accent )
```

Hit the string a bit harder. Influences brightness and decay.

Parameters

accent	Works 0-1.

5.60.2.6 SetBrightness()

```
void StringVoice::SetBrightness (
            float brightness )
```

Set the brighness of the string, and the noise density.

Parameters

brightness	Works best 0-1

5.60.2.7 SetDamping()

```
void StringVoice::SetDamping (
             float damping )
```

How long the resonant body takes to decay relative to the accent level.

Parameters

damping	Works best 0-1. Full damp is only achieved with full accent.

5.60.2.8 SetFreq()

```
void StringVoice::SetFreq (
             float freq )
```

Set the string root frequency.

Parameters

freq	Frequency in Hz.

5.60.2.9 SetStructure()

```
void StringVoice::SetStructure (
             float structure )
```

Changes the string's nonlinearity (string type).

Parameters

structure	Works 0-1. 0-.26 is curved bridge, .26-1 is dispersion.

5.60.2.10 SetSustain()

```
void StringVoice::SetSustain (
            bool sustain )
```

Continually excite the string with noise.

Parameters

sustain	True turns on the noise.

5.60.2.11 Trig()

```
void StringVoice::Trig ( )
```

Strike the string.

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

- Source/PhysicalModeling/stringvoice.h
- Source/PhysicalModeling/stringvoice.cpp

5.61 daisysp::Svf Class Reference

```
#include <svf.h>
```

Public Member Functions

- void Init (float sample_rate)
- void Process (float in)
- void SetFreq (float f)
- void SetRes (float r)
- void SetDrive (float d)
- float Low ()
- float High ()
- float Band ()
- float Notch ()
- float Peak ()

5.61.1 Detailed Description

Double Sampled, Stable State Variable Filter

Credit to Andrew Simper from musicdsp.org

This is his "State Variable Filter (Double Sampled, Stable)"

Additional thanks to Laurent de Soras for stability limit, and Stefan Diedrichsen for the correct notch output

Ported by: Stephen Hensley

5.61.2 Member Function Documentation

5.61.2.1 Band()

```
float daisysp::Svf::Band ( )  [inline]
```

bandpass output

Returns

band pass output of the filter

5.61.2.2 High()

```
float daisysp::Svf::High ( )  [inline]
```

highpass output

Returns

high pass output of the filter

5.61.2.3 Init()

```
void Svf::Init (
             float sample_rate )
```

Initializes the filter float sample_rate - sample rate of the audio engine being run, and the frequency that the Process function will be called.

5.61.2.4 Low()

```
float daisysp::Svf::Low ( )  [inline]
```

lowpass output

Returns

low pass output of the filter

5.61.2.5 Notch()

```
float daisysp::Svf::Notch ( )  [inline]
```

notchpass output

Returns

notch pass output of the filter

5.61.2.6 Peak()

```
float daisysp::Svf::Peak ( )  [inline]
```

peak output

Returns

peak output of the filter

5.61.2.7 Process()

```
void Svf::Process (
             float in )
```

Process the input signal, updating all of the outputs.

5.61.2.8 SetDrive()

```
void Svf::SetDrive (
             float d )
```

sets the drive of the filter affects the response of the resonance of the filter

5.61.2.9 SetFreq()

```
void Svf::SetFreq (
             float f )
```

sets the frequency of the cutoff frequency. f must be between 0.0 and sample_rate / 3

5.61.2.10 SetRes()

```
void Svf::SetRes (
             float r )
```

sets the resonance of the filter. Must be between 0.0 and 1.0 to ensure stability.

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

- Source/Filters/svf.h
- Source/Filters/svf.cpp

5.62 daisysp::SwingVCA Class Reference

Swing type VCA.

`#include <hihat.h>`

Public Member Functions

- float **operator()** (float s, float gain)

5.62.1 Detailed Description

Swing type VCA.

Author

Ben Sergentanis

Date

Jan 2021 Ported from pichenettes/eurorack/plaits/dsp/drums/hihat.h
to an independent module.
Original code written by Emilie Gillet in 2016.

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

- Source/Drums/hihat.h

5.63 daisysp::SyntheticBassDrum Class Reference

Naive bass drum model (modulated oscillator with FM + envelope).

`#include <synthbassdrum.h>`

Public Member Functions

- void Init (float sample_rate)
- float DistortedSine (float phase, float phase_noise, float dirtiness)
- float TransistorVCA (float s, float gain)
- float Process (bool trigger=false)
- void Trig ()
- void SetSustain (bool sustain)
- void SetAccent (float accent)
- void SetFreq (float freq)
- void SetTone (float tone)
- void SetDecay (float decay)
- void SetDirtiness (float dirtiness)
- void SetFmEnvelopeAmount (float fm_envelope_amount)
- void SetFmEnvelopeDecay (float fm_envelope_decay)

5.63.1 Detailed Description

Naive bass drum model (modulated oscillator with FM + envelope).

Author

Ben Sergentanis

Date

Jan 2021 Inadvertently 909-ish.

Ported from pichenettes/eurorack/plaits/dsp/drums/synthetic_bass_drum.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.63.2 Member Function Documentation

5.63.2.1 DistortedSine()

```
float SyntheticBassDrum::DistortedSine (
             float phase,
             float phase_noise,
             float dirtiness )  [inline]
```

Generates a distorted sine wave

5.63.2.2 Init()

```
void SyntheticBassDrum::Init (
             float sample_rate )
```

Init the module

Parameters

sample_rate	Audio engine sample rate.

5.63.2.3 Process()

```
float SyntheticBassDrum::Process (
             bool trigger = false )
```

Get the next sample.

Parameters

trigger	True triggers the BD. This is optional.

5.63.2.4 SetAccent()

```
void SyntheticBassDrum::SetAccent (
             float accent )
```

Sets the amount of accent.

Parameters

accent	Works 0-1.

5.63.2.5 SetDecay()

```
void SyntheticBassDrum::SetDecay (
             float decay )
```

Sets how long the drum's volume takes to decay.

Parameters

Works	0-1.

5.63.2.6 SetDirtiness()

```
void SyntheticBassDrum::SetDirtiness (
             float dirtiness )
```

Makes things grimy

Parameters

dirtiness	Works 0-1.

5.63.2.7 SetFmEnvelopeAmount()

```
void SyntheticBassDrum::SetFmEnvelopeAmount (
             float fm_envelope_amount )
```

Sets how much of a pitch sweep the drum experiences when triggered.

Parameters

fm_envelope_amount	Works 0-1.

5.63.2.8 SetFmEnvelopeDecay()

```
void SyntheticBassDrum::SetFmEnvelopeDecay (
             float fm_envelope_decay )
```

Sets how long the initial pitch sweep takes.

Parameters

fm_envelope_decay	Works 0-1.

5.63.2.9 SetFreq()

```
void SyntheticBassDrum::SetFreq (
             float freq )
```

Set the bass drum's root frequency.

Parameters

Frequency	in Hz.

5.63.2.10 SetSustain()

```
void SyntheticBassDrum::SetSustain (
             bool sustain )
```

Allows the drum to play continuously

Parameters

sustain	True sets the drum on infinite sustain.

5.63.2.11 SetTone()

```
void SyntheticBassDrum::SetTone (
             float tone )
```

Sets the overall bright / darkness of the drum.

Parameters

tone	Works 0-1.

5.63.2.12 TransistorVCA()

```
float SyntheticBassDrum::TransistorVCA (
             float s,
             float gain )  [inline]
```

Transistor VCA simulation.

Parameters

s	Input sample.
gain	VCA gain.

5.63.2.13 Trig()

```
void SyntheticBassDrum::Trig ( )
```

Trigger the drum

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

- Source/Drums/synthbassdrum.h
- Source/Drums/synthbassdrum.cpp

5.64 daisysp::SyntheticBassDrumAttackNoise Class Reference

Attack Noise generator for SyntheticBassDrum.

```
#include <synthbassdrum.h>
```

Public Member Functions

- void Init ()
- float Process ()

5.64.1 Detailed Description

Attack Noise generator for SyntheticBassDrum.

Author

Ben Sergentanis

Date

Jan 2021 Ported from pichenettes/eurorack/plaits/dsp/drums/synthetic_bass_drum.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.64.2 Member Function Documentation

5.64.2.1 Init()

```
void SyntheticBassDrumAttackNoise::Init ( )
```

Init the module

5.64.2.2 Process()

```
float SyntheticBassDrumAttackNoise::Process ( )
```

Get the next sample.

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

- Source/Drums/synthbassdrum.h
- Source/Drums/synthbassdrum.cpp

5.65 daisysp::SyntheticBassDrumClick Class Reference

Click noise for SyntheticBassDrum.

```
#include <synthbassdrum.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float in)

5.65.1 Detailed Description

Click noise for SyntheticBassDrum.

Author

Ben Sergentanis

Date

Jan 2021 Ported from pichenettes/eurorack/plaits/dsp/drums/synthetic_bass_drum.h
to an independent module.
Original code written by Emilie Gillet in 2016.

5.65.2 Member Function Documentation

5.65.2.1 Init()

```
void SyntheticBassDrumClick::Init (
            float sample_rate )
```

Init the module

Parameters

sample_rate	Audio engine sample rate.

5.65.2.2 Process()

```
float SyntheticBassDrumClick::Process (
            float in )
```

Get the next sample.

Parameters

in	Trigger the click.

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

- Source/Drums/synthbassdrum.h
- Source/Drums/synthbassdrum.cpp

5.66 daisysp::SyntheticSnareDrum Class Reference

Naive snare drum model (two modulated oscillators + filtered noise).

```
#include <synthsnaredrum.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (bool trigger=false)
- void Trig ()
- void SetSustain (bool sustain)
- void SetAccent (float accent)
- void SetFreq (float f0)
- void SetFmAmount (float fm_amount)
- void SetDecay (float decay)
- void SetSnappy (float snappy)

5.66.1 Detailed Description

Naive snare drum model (two modulated oscillators + filtered noise).

Author

Ben Sergentanis

Date

Jan 2021 Uses a few magic numbers taken from the 909 schematics:

- Ratio between the two modes of the drum set to 1.47.
- Funky coupling between the two modes.
- Noise coloration filters and envelope shapes for the snare.

  Ported from pichenettes/eurorack/plaits/dsp/drums/synthetic_snare_drum.h
  to an independent module.
  Original code written by Emilie Gillet in 2016.

5.66.2 Member Function Documentation

5.66.2.1 Init()

```
void SyntheticSnareDrum::Init (
             float sample_rate )
```

Init the module

Parameters

sample_rate	Audio engine sample rate

5.66.2.2 Process()

```
float SyntheticSnareDrum::Process (
             bool trigger = false )
```

Get the next sample.

Parameters

trigger	True = hit the drum. This argument is optional.

5.66.2.3 SetAccent()

```
void SyntheticSnareDrum::SetAccent (
             float accent )
```

Set how much accent to use

Parameters

accent	Works 0-1.

5.66.2.4 SetDecay()

```
void SyntheticSnareDrum::SetDecay (
             float decay )
```

Set the length of the drum decay

Parameters

decay	Works with positive numbers

5.66.2.5 SetFmAmount()

```
void SyntheticSnareDrum::SetFmAmount (
             float fm_amount )
```

Set the amount of fm sweep.

Parameters

fm_amount	Works from 0 - 1.

5.66.2.6 SetFreq()

```
void SyntheticSnareDrum::SetFreq (
             float f0 )
```

Set the drum's root frequency

Parameters

f0	Freq in Hz

5.66.2.7 SetSnappy()

```
void SyntheticSnareDrum::SetSnappy (
             float snappy )
```

Sets the mix between snare and drum.

Parameters

snappy	1 = just snare. 0 = just drum.

5.66.2.8 SetSustain()

```
void SyntheticSnareDrum::SetSustain (
             bool sustain )
```

Make the drum ring out infinitely.

Parameters

sustain	True = infinite sustain.

5.66.2.9 Trig()

```
void SyntheticSnareDrum::Trig ( )
```

Trigger the drum

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

- Source/Drums/synthsnaredrum.h
- Source/Drums/synthsnaredrum.cpp

5.67 daisysp::Tone Class Reference

```
#include <tone.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float &in)
- void SetFreq (float &freq)
- float GetFreq ()

5.67.1 Detailed Description

A first-order recursive low-pass filter with variable frequency response.

5.67.2 Member Function Documentation

5.67.2.1 GetFreq()

```
float daisysp::Tone::GetFreq ( )  [inline]
```

Returns

the current value for the cutoff frequency or half-way point of the filter.

5.67.2.2 Init()

```
void Tone::Init (
             float sample_rate )
```

Initializes the Tone module. sample_rate - The sample rate of the audio engine being run.

5.67.2.3 Process()

```
float Tone::Process (
             float & in )
```

Processes one sample through the filter and returns one sample. in - input signal

5.67.2.4 SetFreq()

```
void daisysp::Tone::SetFreq (
             float & freq )  [inline]
```

Sets the cutoff frequency or half-way point of the filter.

Parameters

freq	- frequency value in Hz. Range: Any positive value.

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

- Source/Filters/tone.h
- Source/Filters/tone.cpp

5.68 daisysp::Tremolo Class Reference

Tremolo effect.

```
#include <tremolo.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process (float in)
- void SetFreq (float freq)
- void SetWaveform (int waveform)
- void SetDepth (float depth)

5.68.1 Detailed Description

Tremolo effect.

Author

Ben Sergentanis

Date

Jan 2021 Based on `https://christianfloisand.wordpress.com/2012/04/18/coding-some-tremolo`

5.68.2 Member Function Documentation

5.68.2.1 Init()

```
void Tremolo::Init (
            float sample_rate )
```

Initializes the module

Parameters

sample_rate	The sample rate of the audio engine being run.

5.68.2.2 Process()

```
float Tremolo::Process (
             float in )
```

Parameters

in	Input sample.

Returns

Next floating point sample.

5.68.2.3 SetDepth()

```
void Tremolo::SetDepth (
             float depth )
```

How much to modulate your volume.

Parameters

depth	Works 0-1.

5.68.2.4 SetFreq()

```
void Tremolo::SetFreq (
             float freq )
```

Sets the tremolo rate.

Parameters

freq	Tremolo freq in Hz.

5.68.2.5 SetWaveform()

```
void Tremolo::SetWaveform (
             int waveform )
```

Shape of the modulating lfo

Parameters

waveform	Oscillator waveform. Use Oscillator::WAVE_SIN for example.

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

- Source/Effects/tremolo.h
- Source/Effects/tremolo.cpp

5.69 daisysp::VariableSawOscillator Class Reference

Variable Saw Oscillator.

```
#include <variablesawosc.h>
```

Public Member Functions

- void **Init** (float sample_rate)
- float Process ()
- void SetFreq (float frequency)
- void SetPW (float pw)
- void SetWaveshape (float waveshape)

5.69.1 Detailed Description

Variable Saw Oscillator.

Author

Ben Sergentanis

Date

Dec 2020 Saw with variable slope or notch.

Ported from pichenettes/eurorack/plaits/dsp/oscillator/variable_saw_oscillator.h

to an independent module.
Original code written by Emilie Gillet in 2016.

5.69.2 Member Function Documentation

5.69.2.1 Process()

```
float VariableSawOscillator::Process ( )
```

Get the next sample

5.69.2.2 SetFreq()

```
void VariableSawOscillator::SetFreq (
             float frequency )
```

Set master freq.

Parameters

frequency	Freq in Hz.

5.69.2.3 SetPW()

```
void VariableSawOscillator::SetPW (
             float pw )
```

Adjust the wave depending on the shape

Parameters

pw	Notch or slope. Works best -1 to 1.

5.69.2.4 SetWaveshape()

```
void VariableSawOscillator::SetWaveshape (
             float waveshape )
```

Slope or notch

Parameters

waveshape	0 = notch, 1 = slope

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

- Source/Synthesis/variablesawosc.h
- Source/Synthesis/variablesawosc.cpp

5.70 daisysp::VariableShapeOscillator Class Reference

Variable Waveshape Oscillator.

```
#include <variableshapeosc.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process ()
- void SetFreq (float frequency)
- void SetPW (float pw)
- void SetWaveshape (float waveshape)
- void SetSync (bool enable_sync)
- void SetSyncFreq (float frequency)

5.70.1 Detailed Description

Variable Waveshape Oscillator.

Author

Ben Sergentanis

Date

Dec 2020 Continuously variable waveform.

Ported from pichenettes/eurorack/plaits/dsp/oscillator/variable_shape_oscillator.h

to an independent module.
Original code written by Emilie Gillet in 2016.

5.70.2 Member Function Documentation

5.70.2.1 Init()

```
void VariableShapeOscillator::Init (
            float sample_rate )
```

Initialize the oscillator

Parameters

sample_rate	Audio engine sample rate

5.70.2.2 Process()

```
float VariableShapeOscillator::Process ( )
```

Get next sample

5.70.2.3 SetFreq()

```
void VariableShapeOscillator::SetFreq (
             float frequency )
```

Set master freq.

Parameters

frequency	Freq in Hz.

5.70.2.4 SetPW()

```
void VariableShapeOscillator::SetPW (
             float pw )
```

Set pulse width / saw, ramp, tri.

Parameters

pw	PW when shape is square. Saw, ramp, tri otherwise.

5.70.2.5 SetSync()

```
void VariableShapeOscillator::SetSync (
             bool enable_sync )
```

Whether or not to sync to the sync oscillator

Parameters

enable_sync	True to turn sync on.

5.70.2.6 SetSyncFreq()

```
void VariableShapeOscillator::SetSyncFreq (
             float frequency )
```

Set sync oscillator freq.

Parameters

frequency	Freq in Hz.

5.70.2.7 SetWaveshape()

```
void VariableShapeOscillator::SetWaveshape (
             float waveshape )
```

Switch from saw/ramp/tri to square.

Parameters

waveshape	0 is saw/ramp/tri, 1 is square.

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

- Source/Synthesis/variableshapeosc.h
- Source/Synthesis/variableshapeosc.cpp

5.71 daisysp::VosimOscillator Class Reference

Vosim Oscillator Module

.

```
#include <vosim.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process ()
- void SetFreq (float freq)
- void SetForm1Freq (float freq)
- void SetForm2Freq (float freq)
- void SetShape (float shape)

5.71.1 Detailed Description

Vosim Oscillator Module

.

Author

Ben Sergentanis

Date

Dec 2020 Two sinewaves multiplied by and sync'ed to a carrier.

Ported from pichenettes/eurorack/plaits/dsp/oscillator/vosim_oscillator.h

to an independent module.
Original code written by Emilie Gillet in 2016.

5.71.2 Member Function Documentation

5.71.2.1 Init()

```
void VosimOscillator::Init (
             float sample_rate )
```

Initializes the FormantOscillator module.

Parameters

sample_rate	- The sample rate of the audio engine being run.

5.71.2.2 Process()

```
float VosimOscillator::Process ( )
```

Get the next sample

5.71.2.3 SetForm1Freq()

```
void VosimOscillator::SetForm1Freq (
             float freq )
```

Set formant 1 frequency.

Parameters

freq	Frequency in Hz.

5.71.2.4 SetForm2Freq()

```
void VosimOscillator::SetForm2Freq (
             float freq )
```

Set formant 2 frequency.

Parameters

freq	Frequency in Hz.

5.71.2.5 SetFreq()

```
void VosimOscillator::SetFreq (
             float freq )
```

Set carrier frequency.

Parameters

freq	Frequency in Hz.

5.71.2.6 SetShape()

```
void VosimOscillator::SetShape (
             float shape )
```

Waveshaping

Parameters

shape	Shape to set. Works -1 to 1

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

- Source/Synthesis/vosim.h
- Source/Synthesis/vosim.cpp

5.72 daisysp::WhiteNoise Class Reference

```
#include <whitenoise.h>
```

Public Member Functions

- void Init ()
- void SetAmp (float a)
- float Process ()

5.72.1 Detailed Description

fast white noise generator

I think this came from musicdsp.org at some point

5.72.2 Member Function Documentation

5.72.2.1 Init()

```
void daisysp::WhiteNoise::Init ( )  [inline]
```

Initializes the WhiteNoise object

5.72.2.2 Process()

```
float daisysp::WhiteNoise::Process ( )  [inline]
```

returns a new sample of noise in the range of -amp_ to amp_

5.72.2.3 SetAmp()

```
void daisysp::WhiteNoise::SetAmp (
             float a )  [inline]
```

sets the amplitude of the noise output

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

- Source/Noise/whitenoise.h

5.73 daisysp::ZOscillator Class Reference

ZOscillator Module
.

```
#include <zoscillator.h>
```

Public Member Functions

- void Init (float sample_rate)
- float Process ()
- void SetFreq (float freq)
- void SetFormantFreq (float freq)
- void SetShape (float shape)
- void SetMode (float mode)

5.73.1 Detailed Description

ZOscillator Module
.

Author

Ben Sergentanis

Date

Dec 2020 Sinewave multiplied by and sync'ed to a carrier.

Ported from pichenettes/eurorack/plaits/dsp/oscillator/z_oscillator.h

to an independent module.
Original code written by Emilie Gillet in 2016.

5.73.2 Member Function Documentation

5.73.2.1 Init()

```
void ZOscillator::Init (
            float sample_rate )
```

Init ZOscillator module

Parameters

sample_rate	Audio engine sample rate.

5.73.2.2 Process()

```
float ZOscillator::Process ( )
```

Get next sample

5.73.2.3 SetFormantFreq()

```
void ZOscillator::SetFormantFreq (
             float freq )
```

Set the formant osc. freq

Parameters

freq	Frequency in Hz.

5.73.2.4 SetFreq()

```
void ZOscillator::SetFreq (
             float freq )
```

Set the carrier frequency

Parameters

freq	Frequency in Hz.

5.73.2.5 SetMode()

```
void ZOscillator::SetMode (
             float mode )
```

Set the offset amount and phase shift.
$<$ 1/3 is just phase shift, $>$ 2/3 is just offset, and between them is both.

Parameters

mode	Mode to set. Works best -1 to 1

5.73.2.6 SetShape()

```
void ZOscillator::SetShape (
             float shape )
```

Adjust the contour of the waveform.

Parameters

shape	Waveshape to set. Works best 0-1.

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

- Source/Synthesis/zoscillator.h
- Source/Synthesis/zoscillator.cpp

Chapter 6

File Documentation

6.1 Source/Drums/analogbassdrum.h File Reference

```
#include <stdint.h>
#include "Synthesis/oscillator.h"
#include "Filters/svf.h"
```

Classes

- class daisysp::AnalogBassDrum

  808 bass drum model, revisited.

6.2 Source/Drums/analogsnaredrum.h File Reference

```
#include "Filters/svf.h"
#include <stdint.h>
```

Classes

- class daisysp::AnalogSnareDrum

  808 snare drum model, revisited.

6.3 Source/Drums/hihat.h File Reference

```
#include "Filters/svf.h"
#include "Synthesis/oscillator.h"
#include <stdint.h>
#include <stdlib.h>
```

Classes

- class daisysp::SquareNoise

    808 style "metallic noise" with 6 square oscillators.
- class daisysp::RingModNoise

    Ring mod style metallic noise generator.
- class daisysp::SwingVCA

    Swing type VCA.
- class daisysp::LinearVCA

    Linear type VCA.
- class daisysp::HiHat< MetallicNoiseSource, VCA, resonance >

    808 HH, with a few extra parameters to push things to the CY territory...

6.4 Source/Drums/synthbassdrum.h File Reference

```
#include "Filters/svf.h"
#include "Utility/dsp.h"
#include <stdint.h>
```

Classes

- class daisysp::SyntheticBassDrumClick

    Click noise for SyntheticBassDrum.
- class daisysp::SyntheticBassDrumAttackNoise

    Attack Noise generator for SyntheticBassDrum.
- class daisysp::SyntheticBassDrum

    Naive bass drum model (modulated oscillator with FM + envelope).

6.5 Source/Drums/synthsnaredrum.h File Reference

```
#include "Filters/svf.h"
#include <stdint.h>
```

Classes

- class daisysp::SyntheticSnareDrum

    Naive snare drum model (two modulated oscillators + filtered noise).

6.6 Source/Effects/chorus.h File Reference

```
#include <stdint.h>
#include "Utility/delayline.h"
```

Classes

- class daisysp::ChorusEngine

  Single Chorus engine. Used in Chorus.
- class daisysp::Chorus

  Chorus Effect.

6.7 Source/Effects/flanger.h File Reference

```
#include <stdint.h>
#include "Utility/delayline.h"
```

Classes

- class daisysp::Flanger

  Flanging Audio Effect.

6.8 Source/Effects/overdrive.h File Reference

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

Classes

- class daisysp::Overdrive

  Distortion / Overdrive Module.

6.9 Source/Effects/sampleratereducer.h File Reference

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

Classes

- class daisysp::SampleRateReducer

  Sample rate reducer.

6.10 Source/Effects/tremolo.h File Reference

```
#include <stdint.h>
#include <math.h>
#include "Synthesis/oscillator.h"
```

Classes

- class daisysp::Tremolo

  Tremolo effect.

6.11 Source/Filters/allpass.h File Reference

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

Classes

- class daisysp::Allpass

6.12 Source/Noise/clockednoise.h File Reference

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

Classes

- class daisysp::ClockedNoise

6.13 Source/Noise/dust.h File Reference

```
#include <cstdlib>
#include <random>
#include "Utility/dsp.h"
```

Classes

- class daisysp::Dust

  Dust Module.

6.14 Source/Noise/fractal_noise.h File Reference

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

Classes

- class daisysp::FractalRandomGenerator< T, order >

  Fractal Noise, stacks octaves of a noise source.

6.15 Source/Noise/grainlet.h File Reference

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

Classes

- class daisysp::GrainletOscillator

  Granular Oscillator Module.

6.16 Source/Noise/particle.h File Reference

```
#include "Filters/svf.h"
#include <stdint.h>
#include <cstdlib>
```

Classes

- class daisysp::Particle

  Random impulse train processed by a resonant filter.

6.17 Source/PhysicalModeling/drip.h File Reference

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

Classes

- class daisysp::Drip

6.18 Source/PhysicalModeling/KarplusString.h File Reference

```
#include <stdint.h>
#include "Dynamics/crossfade.h"
#include "Utility/dcblock.h"
#include "Utility/delayline.h"
#include "Filters/svf.h"
#include "Filters/tone.h"
```

Classes

- class daisysp::String

  Comb filter / KS string.

6.19 Source/PhysicalModeling/modalvoice.h File Reference

```
#include <stdint.h>
#include "Filters/svf.h"
#include "PhysicalModeling/resonator.h"
#include "Noise/dust.h"
```

Classes

- class daisysp::ModalVoice

  Simple modal synthesis voice with a mallet exciter: click -> LPF -> resonator.

6.20 Source/PhysicalModeling/resonator.h File Reference

```
#include <stdint.h>
#include <stddef.h>
#include "Utility/dsp.h"
```

Classes

- class daisysp::ResonatorSvf< batch_size >

  SVF for use in the Resonator Class

  .
- class daisysp::Resonator

  Resonant Body Simulation.

6.21 Source/PhysicalModeling/stringvoice.h File Reference

```
#include "Filters/svf.h"
#include "PhysicalModeling/KarplusString.h"
#include "Noise/dust.h"
#include <stdint.h>
```

Classes

- class daisysp::StringVoice

  Extended Karplus-Strong, with all the niceties from Rings.

6.22 Source/Synthesis/formantosc.h File Reference

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

Classes

- class daisysp::FormantOscillator

  Formant Oscillator Module.

6.23 Source/Synthesis/harmonic_osc.h File Reference

```
#include <stdint.h>
#include "Utility/dsp.h"
```

Classes

- class daisysp::HarmonicOscillator< num_harmonics >

  Harmonic Oscillator Module based on Chebyshev polynomials.

6.24 Source/Synthesis/oscillatorbank.h File Reference

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

Classes

- class daisysp::OscillatorBank

  Oscillator Bank module.

6.25 Source/Synthesis/variablesawosc.h File Reference

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

Classes

- class daisysp::VariableSawOscillator

  Variable Saw Oscillator.

6.26 Source/Synthesis/variableshapeosc.h File Reference

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

Classes

- class daisysp::VariableShapeOscillator

  Variable Waveshape Oscillator.

6.27 Source/Synthesis/vosim.h File Reference

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

Classes

- class daisysp::VosimOscillator

  Vosim Oscillator Module
  .

6.28 Source/Synthesis/zoscillator.h File Reference

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

Classes

- class daisysp::ZOscillator

  ZOscillator Module
  .

6.29 Source/Utility/smooth_random.h File Reference

```
#include "dsp.h"
#include <stdint.h>
#include <stdlib.h>
```

Classes

- class daisysp::SmoothRandomGenerator

  Smooth random generator for internal modulation.
  .

Index