_rain

Generated by Doxygen 1.8.14

Contents

Chapter 1

_Rain

BRIEF

What is __rain_

- Granular Synth with some extra features running on Rhaspberry PI
- loop-based Grainplayer
- Master section with filter and bitcrusher
- Control via OSC

Features Grainplayer

- position: start sample of the section
- Loop length: length of the section in samples
- Loop mode - infinite forward - infinite backwards - infinite forward / backward
- Playback speed (pitch)
- Grain length: length of the window

_rain modes

- forward: Grainplayer is running forward
- backward: Grainplayer is running backwards
- forward/backward: Grainplayer is running forward/backwards
- funk (optional): randomizes propagation

texture / choppy

- controls how far the grains jump during playback

color

- Lowpass (+ Bitcrusher): filters the overtones of the master signal with lowpass, distort the filtered signal with bitcrusher

Libraries

The following libraries are used within this project:

- JACK
- libsndfile
- liblo

Usage

The binaray granular_example expects three command line arguments:

- the wav file to read with the flag '-f'
- the grain size in samples '-l'

```
~/rain $ ./rain -f ../../wav/dopesample.wav -l 1000
```

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

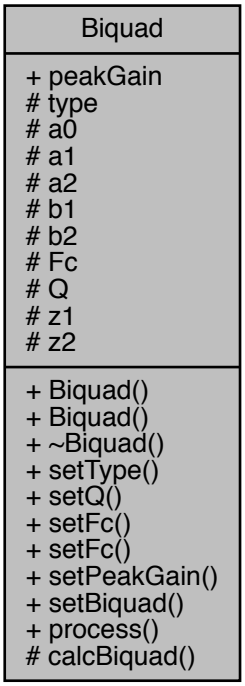
Class Documentation

4.1 Biquad Class Reference

A biquad filter with variable Filter Type, Q, fc, Gain.

`#include <Biquad.h>`

Collaboration diagram for Biquad:



Public Member Functions

- **Biquad** (int type, double Fc, double Q, double peakGainDB)
- void setType (int type)

  sets the filter type
- void setQ (double Q)

  sets the filter quality
- void setFc (double Fc)

  sets the cutoff frequency
- void setFc (double Fc, int fs)

  sets the cutoff frequency in relation to the sample rate
- void setPeakGain (double peakGainDB)

  sets the resonance gain value
- void setBiquad (int type, double Fc, double Q, double peakGain)

  sets member variables necessary for calculation of the filter coefficients
- float process (float in)

  filters input signal

Public Attributes

- double peakGain

  peakGain resonance gain of the biquad filter

Protected Member Functions

- void calcBiquad (void)

  calculates filter coefficients from class member variables

Protected Attributes

- int type

  represents the filter type
- double a0

  represents the filter coefficient a0
- double a1

  represents the filter coefficient a1
- double a2

  represents the filter coefficient a2
- double b1

  represents the filter coefficient b1
- double b2

  represents the filter coefficient b2
- double Fc

  represents filter cutoff frequency
- double Q

  represents filter Q
- double z1

  buffer for previous calculation
- double z2

  buffer for previous calculation

4.1.1 Detailed Description

A biquad filter with variable Filter Type, Q, fc, Gain.

4.1.2 Member Function Documentation

4.1.2.1 process()

```
float Biquad::process (
            float in )  [inline]
```

filters input signal

Parameters

in	is input sample

Returns

filtered output

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

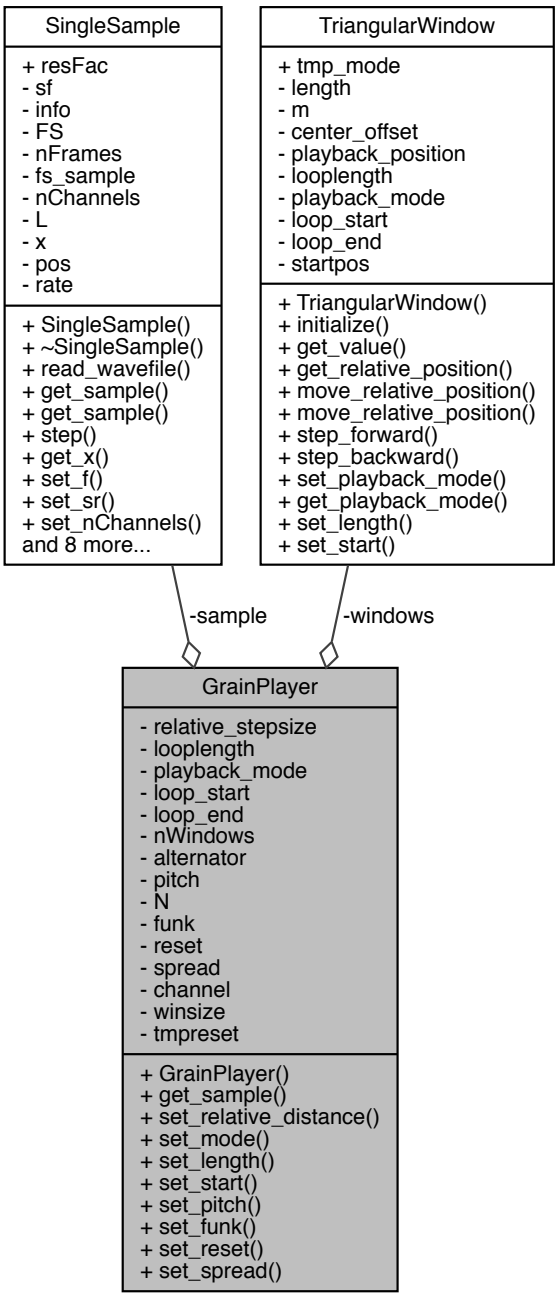
- /Users/marquis/Desktop/_rain/Biquad.h
- /Users/marquis/Desktop/_rain/Biquad.cpp

4.2 GrainPlayer Class Reference

The granular player class, holding a SingleSound and an array of window functions, representing the grains.

```
#include <grainplayer.h>
```

Collaboration diagram for GrainPlayer:



Public Member Functions

- GrainPlayer (std::string filePath, int fs, int win_size, int ch)

  GrainPlayer the constructor.
- double get_sample ()

  get a single sample, using all active grains
- void set_relative_distance (double r)

is the step size of the moving playback pos.

- void set_mode (int m)

  sets the playback mode: forward, backwards or forward-backward played loop
- void set_length (double l)

  sets the looplength
- void set_start (double s)

  sets the start of the loop
- void set_pitch (double p)

  sets the playback speed of the loop
- void set_funk (int f)

  randomized the relative position
- void set_reset (int r)

  randomized the relative position
- void set_spread (double s)

  randomized the relative position

Private Attributes

- SingleSample * sample

  an instance of the single sample class
- double relative_stepsize

  fraction of window size to shift the window position
- TriangularWindow * windows

  pointer to an array of windows
- int looplength

  specifies the Length of the playbackloop
- int playback_mode

  specifies whether the window propagates in a forward or backward loop
- int loop_start

  specifies the start of the playback loop
- int loop_end

  specifies the end of the playback loop
- int nWindows

  specifies how many windows contribute to the Grainplayer
- int alternator

  is an auxiliary variable for forward-backward looping
- double pitch

  is the playback speed
- int N

  is the amount of Samples corresponding to the Length of the loaded Wav File
- int funk

  initiates a random forward movement of the playback window
- int reset

  initalizes the windows of the GrainPlayer
- int spread

  modify the distances between the start positions of the loop in relation to each other
- int channel

  represents left and right channel of the output
- int winsize

  represents window size
- int tmpreset = 1

  auxilary variable for reset function

4.2.1 Detailed Description

The granular player class, holding a SingleSound and an array of window functions, representing the grains.

Author

Henrik von Coler modified by Marquis Fields and Malte Schneider

Date

2019/10/04

4.2.2 Constructor & Destructor Documentation

4.2.2.1 GrainPlayer()

```
GrainPlayer::GrainPlayer (
            std::string filePath,
            int fs,
            int win_size,
            int ch )
```

GrainPlayer the constructor.

Parameters

filePath	path to the wav file to read
fs	sampling rate of the JACK server
win_size	the grain size (size of the window)
nWin	the number of grains to be used

4.2.3 Member Function Documentation

4.2.3.1 get_sample()

```
double GrainPlayer::get_sample ( )
```

get a single sample, using all active grains

Returns

a sound sample for the output

4.2.3.2 set_funk()

```
void GrainPlayer::set_funk (
            int f )
```

randomized the relative position

Parameters

f	funk

4.2.3.3 set_length()

```
void GrainPlayer::set_length (
            double l )
```

sets the looplength

Parameters

l	looplength

4.2.3.4 set_mode()

```
void GrainPlayer::set_mode (
            int m )
```

sets the playback mode: forward, backwards or forward-backward played loop

Parameters

m	playback_mode

4.2.3.5 set_pitch()

```
void GrainPlayer::set_pitch (
            double p )
```

sets the playback speed of the loop

Parameters

p	pitch

4.2.3.6 set_relative_distance()

```
void GrainPlayer::set_relative_distance (
            double r )
```

is the step size of the moving playback pos.

Parameters

r	relative_stepsize

4.2.3.7 set_reset()

```
void GrainPlayer::set_reset (
            int r )
```

randomized the relative position

Parameters

r	reset

4.2.3.8 set_spread()

```
void GrainPlayer::set_spread (
            double s )
```

randomized the relative position

Parameters

s	spread

4.2.3.9 set_start()

```
void GrainPlayer::set_start (
            double s )
```

sets the start of the loop

Parameters

s	loop_start

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

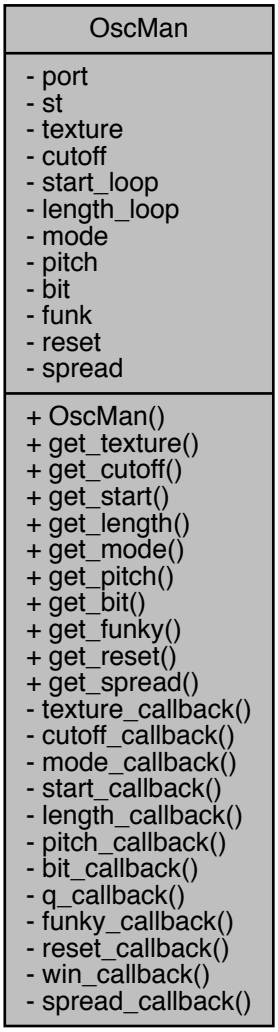
- /Users/marquis/Desktop/_rain/grainplayer.h
- /Users/marquis/Desktop/_rain/grainplayer.cpp

4.3 OscMan Class Reference

Class which parses the incoming OSC messages.

```
#include <oscman.h>
```

Collaboration diagram for OscMan:



Public Member Functions

- OscMan (int p)

  Constructor.
- double get_texture ()

  returns texture input from osc
- double get_cutoff ()

  returns filter cutoff input from osc
- double get_start ()

  returns loopstart input from osc
- double get_length ()

  returns looplength input from osc
- int get_mode ()

returns loopmode input from osc

- double get_pitch ()

  returns pitch input from osc

- int get_bit ()

  returns loopmode input from osc

- int get_funky ()

  returns the randomized relative playback position input from osc

- int get_reset ()

  toggles reset function

- double get_spread ()

  returns the spread input

Static Private Member Functions

- static int texture_callback (const char *path, const char *types, lo_arg **argv, int argc, void *data, void *user_data)

  OscMan::texture_callback.

- static int cutoff_callback (const char *path, const char *types, lo_arg **argv, int argc, void *data, void *user_data)

  OscMan::cutoff_callback.

- static int mode_callback (const char *path, const char *types, lo_arg **argv, int argc, void *data, void *user_data)

  OscMan::mode_callback.

- static int start_callback (const char *path, const char *types, lo_arg **argv, int argc, void *data, void *user_data)

  OscMan::start_callback.

- static int length_callback (const char *path, const char *types, lo_arg **argv, int argc, void *data, void *user_data)

  OscMan::length_callback.

- static int pitch_callback (const char *path, const char *types, lo_arg **argv, int argc, void *data, void *user_data)

  OscMan::pitch_callback.

- static int bit_callback (const char *path, const char *types, lo_arg **argv, int argc, void *data, void *user_data)

  OscMan::bit_callback.

- static int q_callback (const char *path, const char *types, lo_arg **argv, int argc, void *data, void *user_data)

  OscMan::q_callback.

- static int funky_callback (const char *path, const char *types, lo_arg **argv, int argc, void *data, void *user_data)

  OscMan::funky_callback.

- static int reset_callback (const char *path, const char *types, lo_arg **argv, int argc, void *data, void *user_data)

  OscMan::reset_callback.

- static int win_callback (const char *path, const char *types, lo_arg **argv, int argc, void *data, void *user_data)

  OscMan::win_callback.

- static int spread_callback (const char *path, const char *types, lo_arg **argv, int argc, void *data, void *user_data)

  OscMan::spread_callback.

Private Attributes

- int port
  the OSC port to be opened
- lo::ServerThread * st
  OSC server thread
- double texture
  texture Slider
- double cutoff
  cutoff Slider
- double start_loop
  determines the start of the loop
- double length_loop
  determines the end of the loop
- int mode
  mode Slider
- double pitch
  pitch Slider
- int bit
  determines the bit depth of the grainer
- int funk
  funk initiates a random forward/backward movement of the playback window
- int reset
  initalize the windows of the GrainPlayer
- double spread
  modify the distances between the start positions of the loop in relation to each other

4.3.1 Detailed Description

Class which parses the incoming OSC messages.

Author

Henrik von Coler

Date

2019/03/18

Author

Henrik von Coler

Date

2019/03/18$

4.3.2 Constructor & Destructor Documentation

4.3.2.1 OscMan()

```
OscMan::OscMan (
            int p )
```

Constructor.

Parameters

p	OSC port number

4.3.3 Member Function Documentation

4.3.3.1 bit_callback()

```
int OscMan::bit_callback (
            const char * path,
            const char * types,
            lo_arg ** argv,
            int argc,
            void * data,
            void * user_data )  [static], [private]
```

OscMan::bit_callback.

Parameters

path	
types	
argv	
argc	
data	
user_data	

Returns

4.3.3.2 cutoff_callback()

```
int OscMan::cutoff_callback (
            const char * path,
            const char * types,
            lo_arg ** argv,
            int argc,
            void * data,
            void * user_data )  [static], [private]
```

OscMan::cutoff_callback.

Parameters

path	
types	
argv	
argc	
data	
user_data	

Returns

4.3.3.3 funky_callback()

```
int OscMan::funky_callback (
            const char * path,
            const char * types,
            lo_arg ** argv,
            int argc,
            void * data,
            void * user_data )  [static], [private]
```

OscMan::funky_callback.

Parameters

path	
types	
argv	
argc	
data	
user_data	

Returns

4.3.3.4 get_bit()

```
int OscMan::get_bit ( )
```

returns loopmode input from osc

Returns

bit

4.3.3.5 get_cutoff()

```
double OscMan::get_cutoff ( )
```

returns filter cutoff input from osc

Returns

filter cutoff

4.3.3.6 get_funky()

```
int OscMan::get_funky ( )
```

returns the randomized relative playback position input from osc

Returns

funk

4.3.3.7 get_length()

```
double OscMan::get_length ( )
```

returns looplength input from osc

Returns

length_loop

4.3.3.8 get_mode()

```
int OscMan::get_mode ( )
```

returns loopmode input from osc

Returns

returns mode

4.3.3.9 get_pitch()

```
double OscMan::get_pitch ( )
```

returns pitch input from osc

Returns

pitch

4.3.3.10 get_reset()

```
int OscMan::get_reset ( )
```

toggles reset function

Returns

reset

4.3.3.11 get_spread()

```
double OscMan::get_spread ( )
```

returns the spread input

Returns

spread

4.3.3.12 get_start()

```
double OscMan::get_start ( )
```

returns loopstart input from osc

Returns

start_loop

4.3.3.13 get_texture()

```
double OscMan::get_texture ( )
```

returns texture input from osc

Returns

texture

4.3.3.14 length_callback()

```
int OscMan::length_callback (
            const char * path,
            const char * types,
            lo_arg ** argv,
            int argc,
            void * data,
            void * user_data )  [static], [private]
```

OscMan::length_callback.

Parameters

path	
types	
argv	
argc	
data	
user_data	

Returns

4.3.3.15 mode_callback()

```
int OscMan::mode_callback (
            const char * path,
            const char * types,
            lo_arg ** argv,
            int argc,
            void * data,
            void * user_data )  [static], [private]
```

OscMan::mode_callback.

Parameters

path	
types	
argv	
argc	
data	
user_data	

Returns

4.3.3.16 pitch_callback()

```
int OscMan::pitch_callback (
            const char * path,
            const char * types,
            lo_arg ** argv,
            int argc,
            void * data,
            void * user_data )  [static], [private]
```

OscMan::pitch_callback.

Parameters

path	
types	
argv	
argc	
data	
user_data	

Returns

4.3.3.17 q_callback()

```
static int OscMan::q_callback (
            const char * path,
            const char * types,
            lo_arg ** argv,
            int argc,
```

```
            void * data,
            void * user_data )  [static], [private]
```

OscMan::q_callback.

Parameters

path	
types	
argv	
argc	
data	
user_data	

Returns

4.3.3.18 reset_callback()

```
int OscMan::reset_callback (
            const char * path,
            const char * types,
            lo_arg ** argv,
            int argc,
            void * data,
            void * user_data )  [static], [private]
```

OscMan::reset_callback.

Parameters

path	
types	
argv	
argc	
data	
user_data	

Returns

4.3.3.19 spread_callback()

```
int OscMan::spread_callback (
            const char * path,
            const char * types,
            lo_arg ** argv,
            int argc,
```

```
            void * data,
            void * user_data )  [static], [private]
```

OscMan::spread_callback.

Parameters

path	
types	
argv	
argc	
data	
user_data	

Returns

4.3.3.20 start_callback()

```
int OscMan::start_callback (
            const char * path,
            const char * types,
            lo_arg ** argv,
            int argc,
            void * data,
            void * user_data )  [static], [private]
```

OscMan::start_callback.

Parameters

path	
types	
argv	
argc	
data	
user_data	

Returns

4.3.3.21 texture_callback()

```
int OscMan::texture_callback (
            const char * path,
            const char * types,
            lo_arg ** argv,
            int argc,
```

```
            void * data,
            void * user_data )  [static], [private]
```

OscMan::texture_callback.

Parameters

path	
types	
argv	
argc	
data	
user_data	

Returns

4.3.3.22 win_callback()

```
static int OscMan::win_callback (
            const char * path,
            const char * types,
            lo_arg ** argv,
            int argc,
            void * data,
            void * user_data )  [static], [private]
```

OscMan::win_callback.

Parameters

path	
types	
argv	
argc	
data	
user_data	

Returns

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

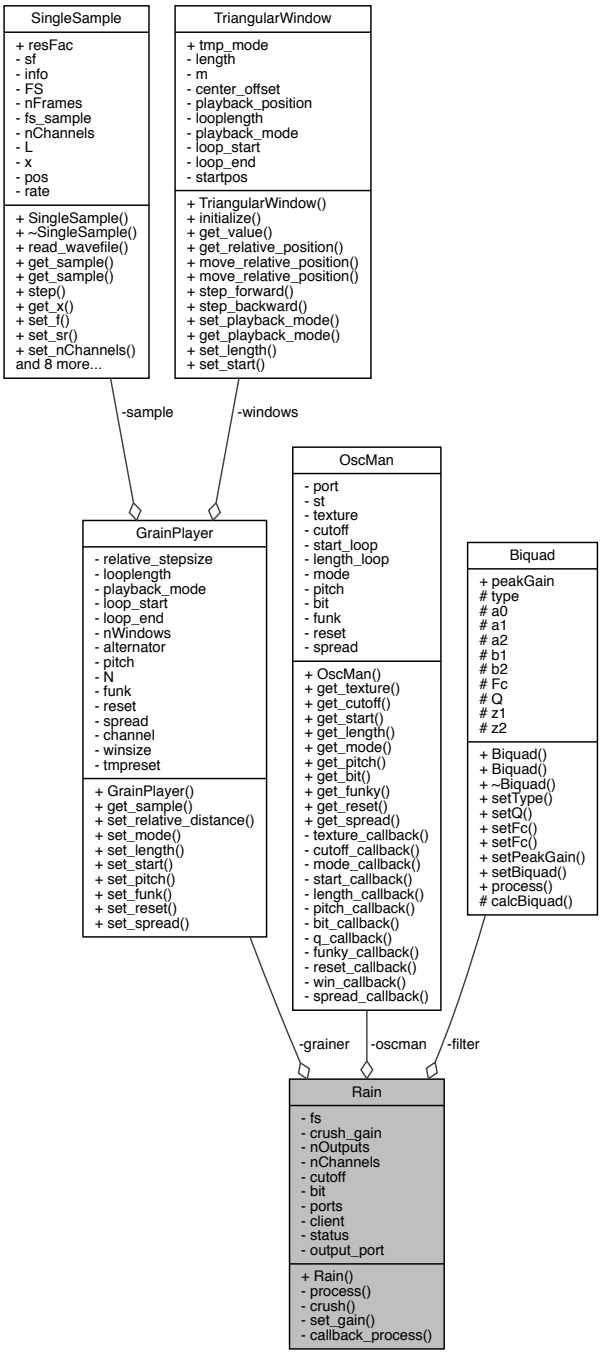
- /Users/marquis/Desktop/_rain/oscman.h
- /Users/marquis/Desktop/_rain/oscman.cpp

4.4 Rain Class Reference

Read a wave file and play it back in a granular style, with modification through OSC.

`#include <rain.h>`

Collaboration diagram for Rain:



Public Member Functions

- Rain (std::string filename, int win_size)

  Constructor.

Private Member Functions

- int process (jack_nframes_t nframes)

  get the next buffer from the sample and playback
- double crush (double sample)

  reduce bit depth of input sample
- void set_gain (double fc)

  sets gain of bitcrushed signale in dependence of filter cutoff frequency

Static Private Member Functions

- static int callback_process (jack_nframes_t x, void ∗object)

Private Attributes

- GrainPlayer ∗ grainer [2]

  creates an GrainPlayer array with all necessary parameters
- Biquad ∗ filter

  filter is a lowpass filter with variable cutoff frequency
- int fs = 0

  sample rate of the jack server
- double crush_gain = 0

  gain of bit crushed output
- int nOutputs = 2

  number off jack output ports
- int nChannels

  number of channels in the sample
- double cutoff

  cutoff represents the cutoff frequency of the lowpass fitler
- int bit

  determines bit depth of bitcrushing
- const char ∗∗ ports

  the jack output ports
- jack_client_t ∗ client

  pointer to Jack client
- jack_status_t status

  Jack status
- jack_port_t ∗∗ output_port

  pointer to Jack audio output port
- OscMan ∗ oscman

  pointer to OSC manager object

4.4.1 Detailed Description

Read a wave file and play it back in a granular style, with modification through OSC.

Author

Henrik von Coler modified by Marquis Fields and Malte Schneider

Date

2019/10/04

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Rain()

```
Rain::Rain (
            std::string filename,
            int win_size )
```

Constructor.

Parameters

filename	path to wav file
win_size	window size of triangular window

4.4.3 Member Function Documentation

4.4.3.1 callback_process()

```
int Rain::callback_process (
            jack_nframes_t x,
            void * object )  [static], [private]
```

Parameters

x	
object	

4.4.3.2 crush()

```
double Rain::crush (
            double sample )  [private]
```

reduce bit depth of input sample

Parameters

sample	current sample to be crushed

Returns

bitcrushed input sample

4.4.3.3 process()

```
int Rain::process (
            jack_nframes_t nframes )  [private]
```

get the next buffer from the sample and playback

Parameters

nframes	buffer size

Returns

output of rain

4.4.3.4 set_gain()

```
void Rain::set_gain (
            double fc )  [private]
```

sets gain of bitcrushed signale in dependence of filter cutoff frequency

Parameters

fc	filter cutoff frequency

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

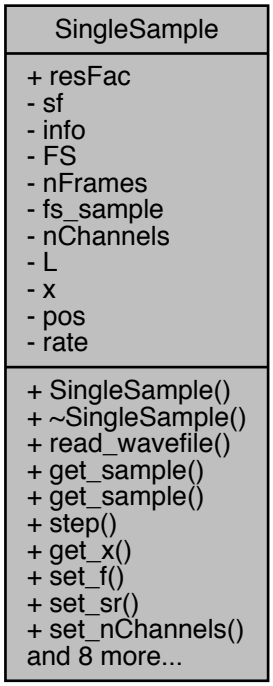
- /Users/marquis/Desktop/_rain/rain.h
- /Users/marquis/Desktop/_rain/rain.cpp

4.5 SingleSample Class Reference

Class which holds the waveform of a sample in an array.

`#include <singlesample.h>`

Collaboration diagram for SingleSample:



Public Member Functions

- SingleSample (std::string filePath, int fs)

  SingleSample the standard constructor.
- virtual ~SingleSample ()

  ~SingleSample ths should be the destructor
- void read_wavefile (std::string filePath)

  read_wavefile
- double get_sample (int chan, int pos)

  get_sample get sample values at exact integer postions
- double get_sample (int chan, double pos)

  get_sample return interpolated value this is an overloaded function!
- void step ()

  step this function sets a new position, regarding the speed, and the resampleFactor
- double ** get_x ()

  get_x

- void **set_f** (int in)
- void **set_sr** (int in)
- void **set_nChannels** (int in)
- int **get_nChannels** ()
- void **set_nFrames** (int in)
- int **get_nFrames** ()
- double **get_rate** ()
- void **set_rate** (double r)
- double **get_pos** ()
- void **set_pos** (double p)
- int **getFS** ()

Public Attributes

- double resFac

  resampleFactor rescales the playback speed according to the sample rate of the wav file and the JACK server

Private Attributes

- SNDFILE * **sf**
- SF_INFO **info**
- int **FS**
- int nFrames

  nFrames The number of frames in the wav file which is the length in samples
- int fs_sample

  fs_sample sampling rate of the wav file
- int nChannels

  nChannels number of channels in this wav
- int L

  L length of the interleaved wav data (number of frames times number of channels)
- double ** x

  x The 2d-array with the sample data, arranged as a matrix.
- double pos

  pos the recent postion within the sample
- double rate

  the playback rate

4.5.1 Detailed Description

Class which holds the waveform of a sample in an array.

This class holds the waveform of a sample in an array and provides all necessary methods for reading and accessing it.

Author

Henrik von Coler

Version

Revision

0.52

Date

2016-11-22

This class holds the waveform of a sample in an array and provides all necessary methods for reading and accessing it. It serves as a state machine by holding a playback speed and position.

Author

Henrik von Coler

Version

Revision

0.527

Date

2016-11-22

4.5.2 Constructor & Destructor Documentation

4.5.2.1 SingleSample()

```
SingleSample::SingleSample (
            std::string filePath,
            int fs )
```

SingleSample the standard constructor.

Parameters

filePath	
fs	the jack sample rate

4.5.3 Member Function Documentation

4.5.3.1 get_sample() [1/2]

```
double SingleSample::get_sample (
            int chan,
            int pos )
```

get_sample get sample values at exact integer postions

Parameters

chan	
pos	

Returns

4.5.3.2 get_sample() [2/2]

```
double SingleSample::get_sample (
            int chan,
            double pos )
```

get_sample return interpolated value this is an overloaded function!

Parameters

chan	the cahnnel to access
pos	the floating point position

Returns

the sample's interpolated value

4.5.3.3 get_x()

```
double ** SingleSample::get_x ( )
```

get_x

Returns

returns a pointer to the sample data arrays

4.5.3.4 read_wavefile()

```
void SingleSample::read_wavefile (
            std::string filePath )
```

read_wavefile

Parameters

filePath	

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

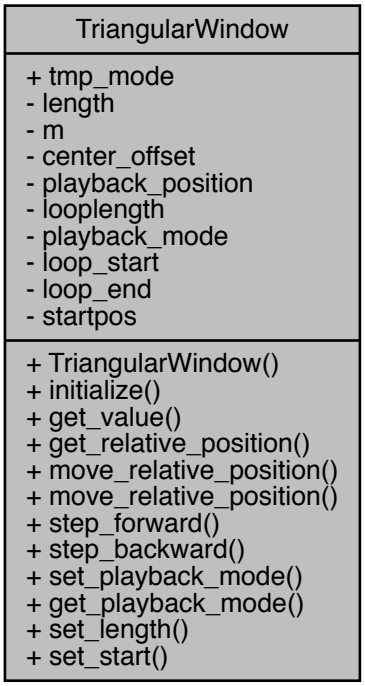
- /Users/marquis/Desktop/_rain/singlesample.h
- /Users/marquis/Desktop/_rain/singlesample.cpp

4.6 TriangularWindow Class Reference

A single window (envelope) for granular synthesis.

```
#include <triangularwindow.h>
```

Collaboration diagram for TriangularWindow:



Public Member Functions

- TriangularWindow ()

  Constructor
- void initialize (int L, int start, int looplength, int direction, int startpos)

  initializes window parameters
- double get_value ()

  get the value of the window at the playback_position
- int get_relative_position ()

  get the relativ playback position of window in wav file
- void move_relative_position (double p, int maxLength)

  move window position in the wav file
- void move_relative_position (double p, int maxLength, int win)

  move window position in the wav file
- int step_forward ()

  increments playback position
- int step_backward ()

  decrements playback position
- void set_playback_mode (int mode)

  sets playback loop mode
- int get_playback_mode ()

  get the playback loop mode
- void set_length (int l, int n)

  sets playback loop length
- void set_start (int s, int n)

  sets playback loop start position

Public Attributes

- int tmp_mode

  auxiliary variable to control the playbackmode

Private Attributes

- int length

  the length of the window
- double m

  the slope of the triangular window
- int center_offset

  the position of the window in the wav file
- int playback_position

  is the playback position within the window
- int looplength

  specifies the Length of the playbackloop
- int playback_mode

  specifies whether the window propagates in a forward or backward loop
- int loop_start

  represets playback loop start position
- int loop_end

  represets playback loop end position
- int startpos

  represets start position of window in the wav file

4.6.1 Detailed Description

A single window (envelope) for granular synthesis.

Author

Henrik von Coler modified by Marquis Fields and Malte Schneider

Date

2019/10/04

4.6.2 Member Function Documentation

4.6.2.1 get_playback_mode()

```
int TriangularWindow::get_playback_mode ( )
```

get the playback loop mode

Returns

playback_mode

4.6.2.2 get_relative_position()

```
int TriangularWindow::get_relative_position ( )
```

get the relativ playback position of window in wav file

Returns

relative positon

4.6.2.3 get_value()

```
double TriangularWindow::get_value ( )
```

get the value of the window at the playback_position

Returns

window value

4.6.2.4 initialize()

```
void TriangularWindow::initialize (
            int L,
            int start,
            int looplength,
            int direction,
            int startpos )
```

initializes window parameters

Parameters

L	window length
start	start position of loop
looplength	length of loop
direction	direction of playback
startpos	relative position of center_offset in wav file

4.6.2.5 move_relative_position() [1/2]

```
void TriangularWindow::move_relative_position (
            double p,
            int maxLength )
```

move window position in the wav file

Parameters

p	the fraction of the window size to shift the position
maxLength	the maximum length of the wav file

4.6.2.6 move_relative_position() [2/2]

```
void TriangularWindow::move_relative_position (
            double p,
            int maxLength,
            int win )
```

move window position in the wav file

Parameters

p	the fraction of the window size to shift the position
maxLength	the maximum length of the wav file
win	window size in samples

4.6.2.7 set_length()

```
void TriangularWindow::set_length (
            int l,
            int n )
```

sets playback loop length

Parameters

l	desired playback loop length
n	length of wav file in samples

4.6.2.8 set_playback_mode()

```
void TriangularWindow::set_playback_mode (
            int mode )
```

sets playback loop mode

Parameters

mode	the desired playback loop mode

4.6.2.9 set_start()

```
void TriangularWindow::set_start (
            int s,
            int n )
```

sets playback loop start position

Parameters

l	desired playback loop start position
n	length of wav file in samples

4.6.2.10 step_backward()

```
int TriangularWindow::step_backward ( )
```

decrements playback position

Returns

returns 3 if end of window is reached

4.6.2.11 step_forward()

```
int TriangularWindow::step_forward ( )
```

increments playback position

Returns

returns 1 if end of window is reached

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

- /Users/marquis/Desktop/_rain/triangularwindow.h
- /Users/marquis/Desktop/_rain/triangularwindow.cpp

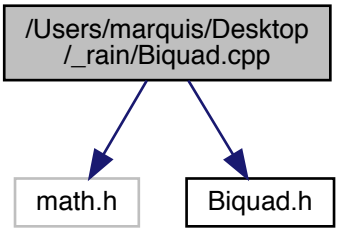
Chapter 5

File Documentation

5.1 /Users/marquis/Desktop/_rain/Biquad.cpp File Reference

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

Include dependency graph for Biquad.cpp:

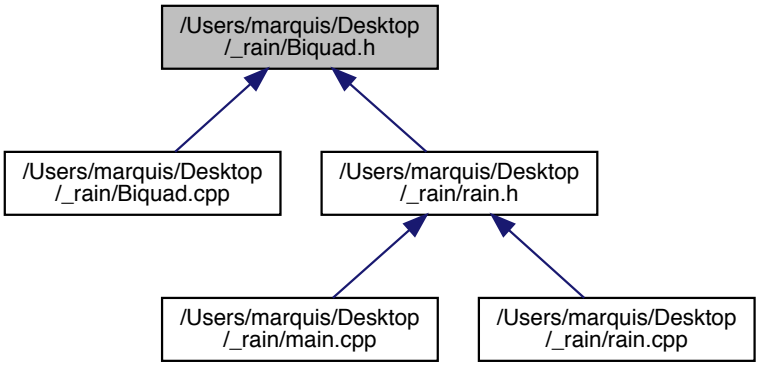


5.1.1 Detailed Description

Henrik von Coler modified by Marquis Fields and Malte Schneider 2019/10/04

5.2 /Users/marquis/Desktop/_rain/Biquad.h File Reference

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



Classes

- class Biquad

  A biquad filter with variable Filter Type, Q, fc, Gain.

Enumerations

- enum {
  bq_type_lowpass = 0, **bq_type_highpass**, **bq_type_bandpass**, **bq_type_notch**,
  bq_type_peak, **bq_type_lowshelf**, **bq_type_highshelf** }

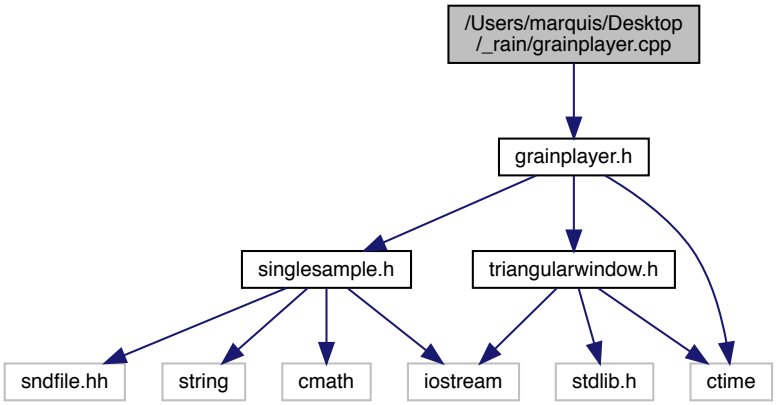
5.2.1 Detailed Description

Henrik von Coler modified by Marquis Fields and Malte Schneider 2019/10/04

5.3 /Users/marquis/Desktop/_rain/grainplayer.cpp File Reference

```
#include "grainplayer.h"
```

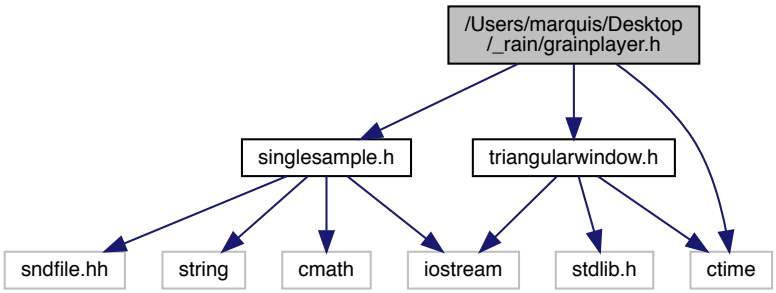
Include dependency graph for grainplayer.cpp:



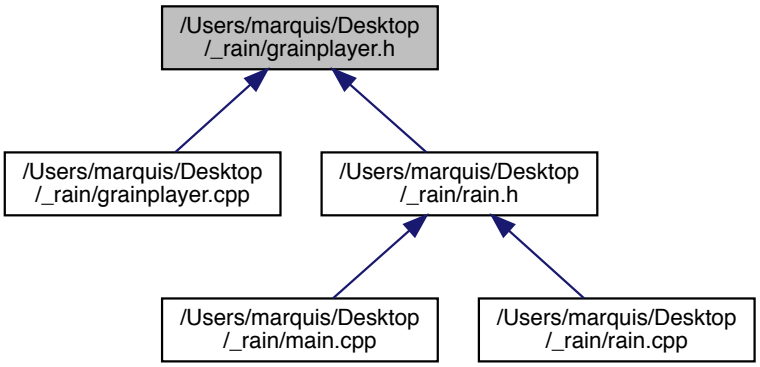
5.4 /Users/marquis/Desktop/_rain/grainplayer.h File Reference

```
#include "singlesample.h"
#include "triangularwindow.h"
#include <ctime>
```

Include dependency graph for grainplayer.h:



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



Classes

- class GrainPlayer

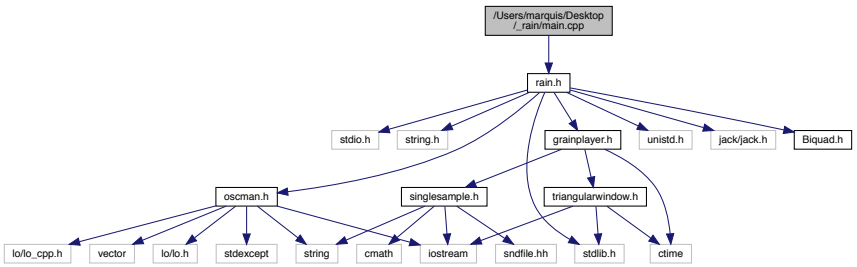
  The granular player class, holding a SingleSound and an array of window functions, representing the grains.

5.5 /Users/marquis/Desktop/_rain/main.cpp File Reference

Granular Synth with some extra features running on Rhaspberry PI.

```
#include "rain.h"
```

Include dependency graph for main.cpp:



Functions

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

5.5.1 Detailed Description

Granular Synth with some extra features running on Rhaspberry PI.

Author

Henrik von Coler modified by Marquis Fields and Malte Schneider

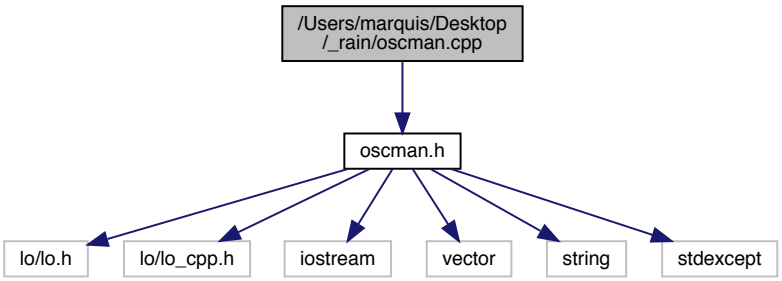
Date

2019/10/04

5.6 /Users/marquis/Desktop/_rain/oscman.cpp File Reference

```
#include "oscman.h"
```

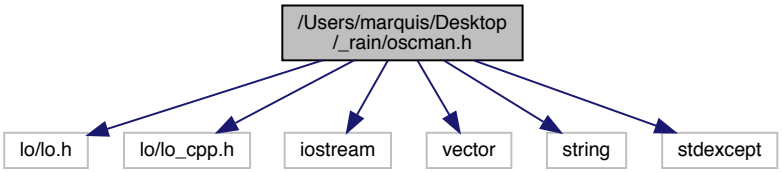
Include dependency graph for oscman.cpp:



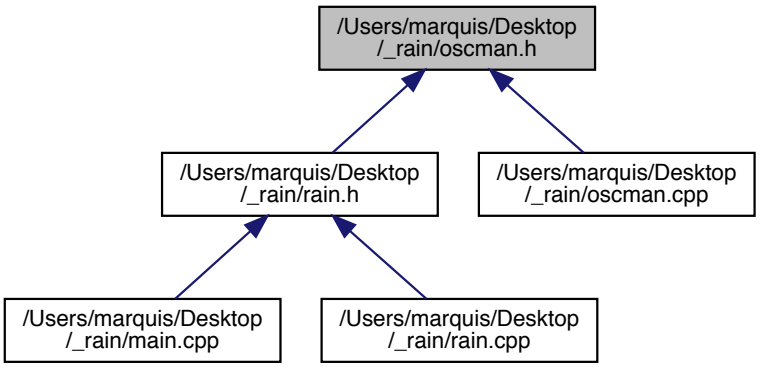
5.7 /Users/marquis/Desktop/_rain/oscman.h File Reference

```
#include <lo/lo.h>
#include <lo/lo_cpp.h>
#include <iostream>
#include <vector>
#include <string>
#include <stdexcept>
```

Include dependency graph for oscman.h:



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



Classes

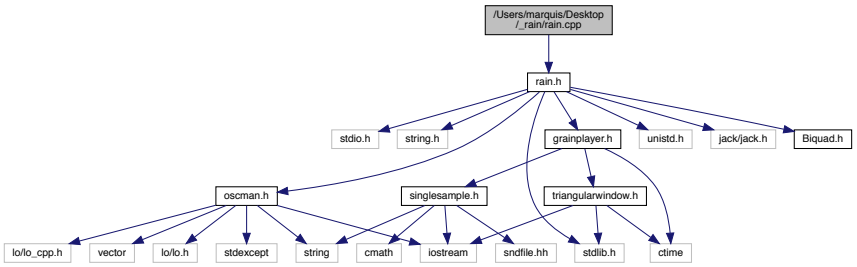
- class OscMan

  Class which parses the incoming OSC messages.

5.8 /Users/marquis/Desktop/_rain/rain.cpp File Reference

```
#include "rain.h"
```

Include dependency graph for rain.cpp:

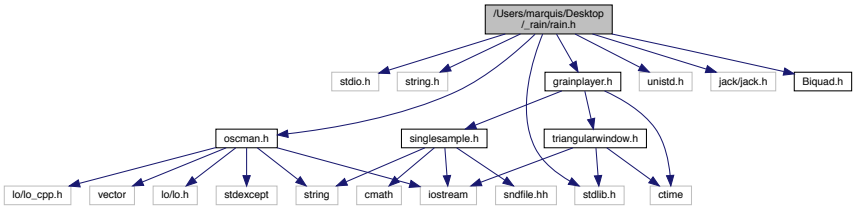


5.9 /Users/marquis/Desktop/_rain/rain.h File Reference

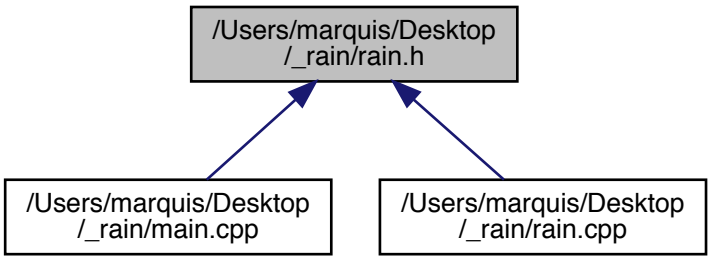
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <jack/jack.h>
#include "oscman.h"
```

```
#include "grainplayer.h"
#include "Biquad.h"
```

Include dependency graph for rain.h:



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



Classes

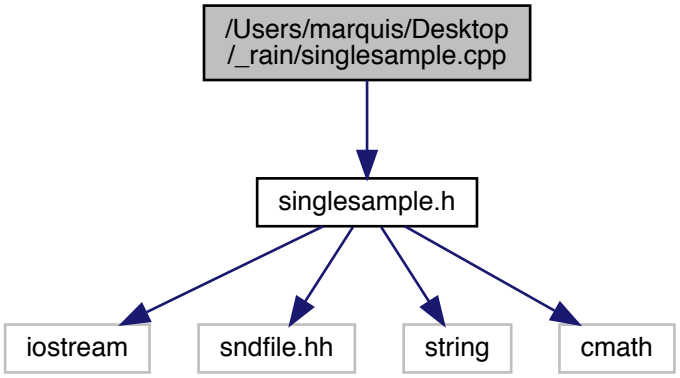
- class Rain

  Read a wave file and play it back in a granular style, with modification through OSC.

5.10 /Users/marquis/Desktop/_rain/singlesample.cpp File Reference

```
#include "singlesample.h"
```

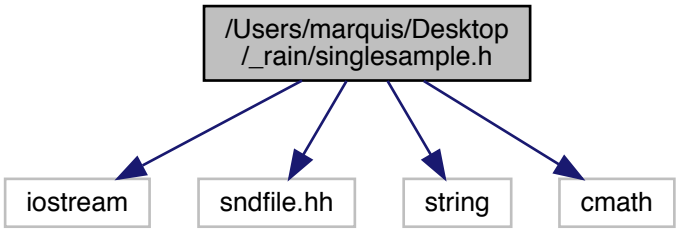
Include dependency graph for singlesample.cpp:



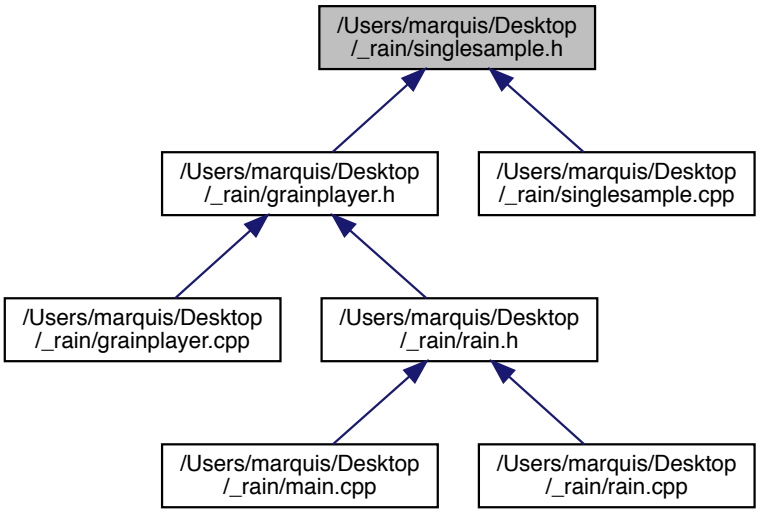
5.11 /Users/marquis/Desktop/_rain/singlesample.h File Reference

```
#include <iostream>
#include <sndfile.hh>
#include <string>
#include <cmath>
```

Include dependency graph for singlesample.h:



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



Classes

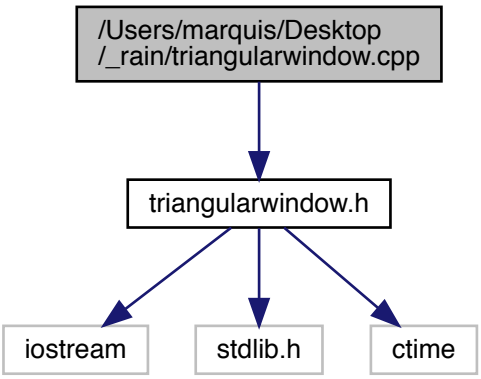
- class SingleSample

  Class which holds the waveform of a sample in an array.

5.12 /Users/marquis/Desktop/_rain/triangularwindow.cpp File Reference

```
#include "triangularwindow.h"
```

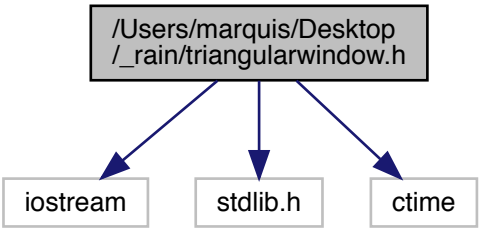
Include dependency graph for triangularwindow.cpp:



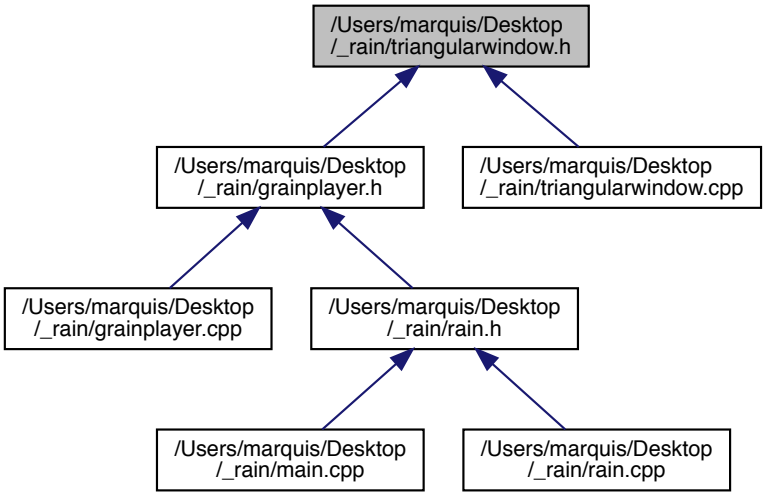
5.13 /Users/marquis/Desktop/_rain/triangularwindow.h File Reference

```
#include <iostream>
#include <stdlib.h>
#include <ctime>
```

Include dependency graph for triangularwindow.h:



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



Classes

- class TriangularWindow

  A single window (envelope) for granular synthesis.

Index