Purple Grain

Generated by Doxygen 1.8.17

Chapter 1

Real Time Audio Programming in C

1.0.1 Granular Synth

Nikita Kretschmar - 459160 Adrian Philipp - 459173 Michael Strobl - 367103 Tim Wennemann - 462830

Chapter 2

Todo List

Global SAMPLERATE
make samplerate adjustable

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

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

Chapter 5

Data Structure Documentation

5.1 c_granular_synth Struct Reference

The Purde Data struct of the c_granular_synth~ object.

`#include <c_granular_synth.h>`

Collaboration diagram for c_granular_synth:

Data Fields

- t_word * **soundfile**
- int **soundfile_length**
- int **current_start_pos**
- int **current_grain_index**
- int **current_adsr_stage_index**
- int **grain_size_ms**
- int **grain_size_samples**
- int **num_grains**
- int **midi_pitch**
- int **midi_velo**
- t_int **playback_position**
- bool **reverse_playback**
- float * **soundfile_table**
- t_float **output_buffer**
- t_float **time_stretch_factor**
- t_float **sr**
- grain * **grains_table**
- envelope * **adsr_env**

5.1.1 Detailed Description

The Purde Data struct of the c_granular_synth~ object.

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

- c_granular_synth.h

5.2 envelope Struct Reference

Data Fields

- t_object **x_obj**
- t_int **attack**
- t_int **decay**
- t_float **sustain**
- t_int **release**
- t_int **duration**
- t_int **attack_samples**
- t_int **decay_samples**
- t_int **release_samples**
- t_sample * **envelope_samples_table**
- enum adsr_stage **adsr**

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

- envelope.h

5.3 grain Struct Reference

Collaboration diagram for grain:

Data Fields

- struct grain * **next_grain**
- t_int **grain_size_samples**
- t_int **grain_index**
- t_float **start**
- t_float **end**
- t_float **time_stretch_factor**
- t_float **current_sample_pos**
- t_float **next_sample_pos**
- bool **grain_active**

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

- grain.h

5.4 pd_granular_synth_tilde Struct Reference

Collaboration diagram for pd_granular_synth_tilde:

Data Fields

- t_object **x_obj**
- t_float **f**
- t_float **sr**
- c_granular_synth * **synth**
- t_int **grain_size**
- t_int **start_pos**
- t_int **midi_pitch**
- t_int **midi_velo**
- t_int **attack**
- t_int **decay**
- t_int **release**
- t_float **sustain**
- t_float **time_stretch_factor**
- t_word * **soundfile**
- t_symbol * **soundfile_arrayname**
- int **soundfile_length**
- float **soundfile_length_ms**
- t_word * **envelopeTable**
- t_inlet * **in_grain_size**
- t_inlet * **in_start_pos**
- t_inlet * **in_time_stretch_factor**
- t_inlet * **in_midi_pitch**
- t_inlet * **in_midi_velo**
- t_inlet * **in_attack**
- t_inlet * **in_decay**
- t_inlet * **in_sustain**
- t_inlet * **in_release**
- t_outlet * **out**

Related Functions

(Note that these are not member functions.)

- void * pd_granular_synth_tilde_new (t_symbol *soundfile_arrayname)

  Creates a new pd_granular_synth_tilde object.
  For more information please refer to the `Pure Data Docs`

- t_int * **pd_granular_synth_tilde_perform** (t_int *w)
- void pd_granular_synth_tilde_free (t_pd_granular_synth_tilde *x)

  Frees our object.

- void pd_granular_synth_tilde_dsp (t_pd_granular_synth_tilde *x, t_signal **sp)

  Adds pd_granular_synth_tilde to the signal chain.

- void pd_granular_synth_tilde_setup (void)

  Setup of pd_granular_synth_tilde
  For more information please refer to the `Pure Data Docs`

5.4.1 Friends And Related Function Documentation

5.4.1.1 pd_granular_synth_tilde_free()

```
void pd_granular_synth_tilde_free (
            t_pd_granular_synth_tilde * x )  [related]
```

Frees our object.

Parameters

x	A pointer the pd_granular_synth_tilde object For more information please refer to the Pure Data Docs

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

- pd_granular_synth~.c

5.5 window Struct Reference

Data Fields

- t_object **x_obj**
- t_int **q_factor**
- t_sample * **window_samples_table**

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

- envelope.h

Chapter 6

File Documentation

6.1 c_granular_synth.c File Reference

The C Part of the synthesizer's implementation.

```
#include "c_granular_synth.h"
#include "envelope.h"
#include "grain.h"
#include "purple_utils.h"
```

Include dependency graph for c_granular_synth.c:

6.2 c_granular_synth.h File Reference

Main file header

Main file header.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "math.h"
#include "grain.h"
#include "envelope.h"
#include "m_pd.h"
```

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

Data Structures

- struct c_granular_synth

  The Purde Data struct of the c_granular_synth~ object.

Macros

- #define **NUMELEMENTS**(x) (sizeof(x) / sizeof((x)[0]))

Typedefs

- typedef struct c_granular_synth **c_granular_synth**

Functions

- void **c_granular_synth_free** (c_granular_synth ∗x)
- c_granular_synth ∗ c_granular_synth_new (t_word ∗soundfile, int soundfile_length, int grain_size_ms, int start_pos, float time_stretch_factor, int attack, int decay, float sustain, int release)

  initial setup of soundfile and adjustment silder related variables
- void **c_granular_synth_generate_window_function** (c_granular_synth ∗x)
- void **c_granular_synth_process_alt** (c_granular_synth ∗x, float ∗in, float ∗out, int vector_size)
- void c_granular_synth_process (c_granular_synth ∗x, float ∗in, float ∗out, int vector_size)

  refresh plaback positions, opens grain scheduleing, writes gaus value, writes into output
- void **c_granular_synth_noteOn** (c_granular_synth ∗x, float frequency, float velocity)
- void c_granular_synth_set_num_grains (c_granular_synth ∗x)

  sets number of grains sets number of grains according to soundfile_length and grain_size_samples
- void c_granular_synth_adjust_current_grain_index (c_granular_synth ∗x)

  adjusts current grain index adjusts current grain index according to currents_start_pos and grain_size_samples
- void c_granular_synth_populate_grain_table (c_granular_synth ∗x)

  generates a grain table generates a grain table according to current_grain_index for negative time_stretch_factor values samples are read in backwards direction
- void grain_internal_scheduling (grain ∗g, c_granular_synth ∗synth)

  scheduling of grain playback
- void c_granular_synth_properties_update (c_granular_synth ∗x, int grain_size_ms, int start_pos, float time↩ _stretch_factor, int midi_pitch, int midi_velo, int attack, int decay, float sustain, int release)

  checks on current input states e.g. slider positions and updates correspondent values
- float calculate_adsr_value (c_granular_synth ∗x)

  calculates ADSR value calculates single atm ADSR value according to current state

Variables

- t_float **SAMPLERATE**

6.2.1 Detailed Description

Main file header

Main file header.

Author

Nikita Kretschmar, Adrian Philipp, Micha Strobl, Tim Wennemann
Audiocommunication Group, Technische Universität Berlin

6.2.2 Function Documentation

6.2.2.1 c_granular_synth_adjust_current_grain_index()

```
void c_granular_synth_adjust_current_grain_index (
            c_granular_synth * x )
```

adjusts current grain index adjusts current grain index according to *currents_start_pos* and *grain_size_samples*

Parameters

x	input pointer of c_granular_synth_adjust_current_grain_index object

6.2.2.2 c_granular_synth_new()

```
c_granular_synth* c_granular_synth_new (
            t_word * soundfile,
            int soundfile_length,
            int grain_size_ms,
            int start_pos,
            float time_stretch_factor,
            int attack,
            int decay,
            float sustain,
            int release )
```

initial setup of soundfile and adjustment silder related variables

Parameters

soundfile	contains the soundfile which can be read in via inlet
soundfile_length	lenght of the soundfile as integer variable
grain_size_ms	size of a grain in milliseconds, adjustable through slider
start_pos	position within the soundfile, adjustable through slider
time_stretch_factor	resizes sample length within a grain, adjustable through slider
attack	attack time in the range of 0 - 4000ms, adjustable through slider
decay	decay time in the range of 0 - 4000ms, adjustable through slider
sustain	sustain time in the range of 0 - 1, adjustable through slider
release	release time in the range of 0 - 10000ms, adjustable through slider

Returns

c_granular_synth∗

6.2.2.3 c_granular_synth_populate_grain_table()

```
void c_granular_synth_populate_grain_table (
            c_granular_synth * x )
```

generates a grain table generates a grain table according to *current_grain_index* for negative *time_stretch_factor* values samples are read in backwards direction

Parameters

x	input pointer of c_granular_synth_populate_grain_table object

6.2.2.4 c_granular_synth_process()

```
void c_granular_synth_process (
            c_granular_synth * x,
            float * in,
            float * out,
            int vector_size )
```

refresh plaback positions, opens grain scheduleing, writes gaus value, writes into output

Parameters

x	input pointer of c_granular_synth_process object
in	input
out	output
vector_size	vectoral size of

6.2.2.5 c_granular_synth_properties_update()

```
void c_granular_synth_properties_update (
            c_granular_synth * x,
            int grain_size_ms,
            int start_pos,
            float time_stretch_factor,
            int midi_velo,
            int midi_pitch,
            int attack,
            int decay,
            float sustain,
            int release )
```

checks on current input states e.g. slider positions and updates correspondent values

Parameters

x	input pointer of c_granular_synth_properties_update object
midi_velo	MIDI input velocity value
midi_pitch	MIDI input pitch/key value
grain_size_ms	size of a grain in milliseconds, adjustable through slider
start_pos	position within the soundfile, adjustable through slider
time_stretch_factor	resizes sample length within a grain, adjustable through slider
attack	attack time in the range of 0 - 4000ms, adjustable through slider
decay	decay time in the range of 0 - 4000ms, adjustable through slider
sustain	sustain time in the range of 0 - 1, adjustable through slider
release	release time in the range of 0 - 10000ms, adjustable through slider

6.2.2.6 c_granular_synth_set_num_grains()

```
void c_granular_synth_set_num_grains (
            c_granular_synth * x )
```

sets number of grains sets number of grains according to *soundfile_length* and *grain_size_samples*

Parameters

x	input pointer of c_granular_synth_set_num_grains object

6.2.2.7 calculate_adsr_value()

```
float calculate_adsr_value (
            c_granular_synth * x )
```

calculates ADSR value calculates single atm ADSR value according to current state

Parameters

x	input pointer of calculate_adsr_value object

Returns

float ADSR value

6.2.2.8 grain_internal_scheduling()

```
void grain_internal_scheduling (
            grain * g,
            c_granular_synth * synth )
```

scheduling of grain playback

sheduling of grain playback

Parameters

g	grain
synth	synthesized output of c_granular_synth object

<

<

<

<

<

6.3 envelope.c File Reference

handles envelope generation generates ADSR envelope according to adjustable attack, decay, sustain and release parameters

```
#include "envelope.h"
#include "grain.h"
#include "vas_mem.h"
#include "purple_utils.h"
#include "m_pd.h"
#include "c_granular_synth.h"
```

Include dependency graph for envelope.c:

Functions

- float calculate_adsr_value (c_granular_synth *x)

  calculates ADSR value calculates single atm ADSR value according to current state
- envelope * envelope_new (int attack, int decay, float sustain, int release)

  generates new ADSR envelope
- float gauss (grain x, int grainindex)

  calculates gauss value calculates gauss value according to
- void envelope_free (envelope *x)

  frees envelope

6.3.1 Detailed Description

handles envelope generation generates ADSR envelope according to adjustable attack, decay, sustain and release parameters

Author

Nikita Kretschmar

Adrian Philipp

Micha Strobl

Tim Wennemann

Version

0.1

Date

2021-09-27

Copyright

Copyright (c) 2021

6.3.2 Function Documentation

6.3.2.1 calculate_adsr_value()

```
float calculate_adsr_value (
            c_granular_synth * x )
```

calculates ADSR value calculates single atm ADSR value according to current state

Parameters

x	input pointer of calculate_adsr_value object

Returns

float ADSR value

6.3.2.2 envelope_free()

```
void envelope_free (
            envelope * x )
```

frees envelope

Parameters

x	input pointer of envelope_free object

6.3.2.3 envelope_new()

```
envelope* envelope_new (
            int attack,
            int decay,
            float sustain,
            int release )
```

generates new ADSR envelope

Parameters

attack	attack time in the range of 0 - 4000ms, adjustable through slider
decay	decay time in the range of 0 - 4000ms, adjustable through slider
sustain	sustain time in the range of 0 - 1, adjustable through slider
release	release time in the range of 0 - 10000ms, adjustable through slider

Returns

envelope∗

6.3.2.4 gauss()

```
float gauss (
            grain x,
            int grainindex )
```

calculates gauss value calculates gauss value according to

Parameters

grainindex	
x	input pointer of gauss object
grainindex	index of grain

Returns

float gauss value

6.4 envelope.h File Reference

Envelope file header

Envelope file header.

```
#include "m_pd.h"
#include "grain.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

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

Data Structures

- struct envelope
- struct window

Typedefs

- typedef struct envelope **envelope**
- typedef struct window **window**

Enumerations

- enum **adsr_stage** {
  ATTACK, **DECAY**, **SUSTAIN**, **RELEASE**,
  SILENT }

Functions

- int **getsamples_from_ms** (int ms, float sr)
- envelope ∗ envelope_new (int attack, int decay, float sustain, int release)

  generates new ADSR envelope
- float gauss (grain x, int sample)

  calculates gauss value calculates gauss value according to
- void envelope_free (envelope ∗x)

  frees envelope

6.4.1 Detailed Description

Envelope file header

Envelope file header.

Author

Nikita Kretschmar, Adrian Philipp, Micha Strobl, Tim Wennemann
Audiocommunication Group, Technische Universität Berlin

6.4.2 Function Documentation

6.4.2.1 envelope_free()

```
void envelope_free (
            envelope * x )
```

frees envelope

Parameters

x	input pointer of envelope_free object

6.4.2.2 envelope_new()

```
envelope* envelope_new (
```

```
            int attack,
            int decay,
            float sustain,
            int release )
```

generates new ADSR envelope

Parameters

attack	attack time in the range of 0 - 4000ms, adjustable through slider
decay	decay time in the range of 0 - 4000ms, adjustable through slider
sustain	sustain time in the range of 0 - 1, adjustable through slider
release	release time in the range of 0 - 10000ms, adjustable through slider

Returns

envelope∗

6.4.2.3 gauss()

```
float gauss (
            grain x,
            int grainindex )
```

calculates gauss value calculates gauss value according to

Parameters

grainindex	
x	input pointer of gauss object
grainindex	index of grain

Returns

float gauss value

6.5 grain.c File Reference

handles grain creation

```
#include "grain.h"
#include "c_granular_synth.h"
#include "envelope.h"
#include "purple_utils.h"
#include "vas_mem.h"
```

Include dependency graph for grain.c:

Macros

- #define SAMPLERATE 44100
  set samplerate to 44100

Functions

- grain grain_new (int grain_size_samples, int soundfile_size, int grain_index, float time_stretch_factor)
  generates new grain
- void grain_internal_scheduling (grain *g, c_granular_synth *synth)
  scheduling of grain playback
- void grain_free (grain *x)
  frees grain

6.5.1 Detailed Description

handles grain creation

Author

Nikita Kretschmar
Adrian Philipp
Micha Strobl
Tim Wennemann Audiocommunication Group, Technische Universität Berlin

Version

0.1

Date

2021-09-27

Copyright

6.5.2 Macro Definition Documentation

6.5.2.1 SAMPLERATE

```
#define SAMPLERATE 44100
```

set samplerate to 44100

Todo make samplerate adjustable

6.5.3 Function Documentation

6.5.3.1 grain_free()

```
void grain_free (
            grain * x )
```

frees grain

frees grain

Parameters

x	input pointer of grain_fre object

6.5.3.2 grain_internal_scheduling()

```
void grain_internal_scheduling (
            grain * g,
            c_granular_synth * synth )
```

scheduling of grain playback

sheduling of grain playback

Parameters

g	grain
synth	synthesized output of c_granular_synth object

<

<

<

<

<

6.5.3.3 grain_new()

```
grain grain_new (
            int grain_size_samples,
            int soundfile_size,
```

```
int grain_index,
float time_stretch_factor )
```

generates new grain

generates new grain depending on *grain_size_samples*, *soundfile_size* and *grain_index*

Parameters

grain_size_samples	size of samples contained in a grain
soundfile_size	size of the soundfile which can be read in via inlet
grain_index	corresponding index of a grain
time_stretch_factor	resizes sample length within a grain, adjustable through slider

Returns

grain

6.6 grain.h File Reference

Grain file header

Grain file header.

```
#include "m_pd.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <stdbool.h>
```

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

Data Structures

- struct grain

Typedefs

- typedef struct grain **grain**

Functions

- grain grain_new (int grain_size_samples, int soundfile_size, int grain_index, float time_stretch_factor)

  generates new grain
- void grain_free (grain *x)

  frees grain

6.6.1 Detailed Description

Grain file header

Grain file header.

Author

Nikita Kretschmar, Adrian Philipp, Micha Strobl, Tim Wennemann
Audiocommunication Group, Technische Universität Berlin

6.6.2 Function Documentation

6.6.2.1 grain_free()

```
void grain_free (
            grain * x )
```

frees grain

frees grain

Parameters

x	input pointer of grain_fre object

6.6.2.2 grain_new()

```
grain grain_new (
            int grain_size_samples,
            int soundfile_size,
            int grain_index,
            float time_stretch_factor )
```

generates new grain

generates new grain depending on *grain_size_samples*, *soundfile_size* and *grain_index*

Parameters

grain_size_samples	size of samples contained in a grain
soundfile_size	size of the soundfile which can be read in via inlet
grain_index	corresponding index of a grain
time_stretch_factor	resizes sample length within a grain, adjustable through slider

Returns

grain

6.7 purple_utils.c File Reference

useful utilities for value conversion and manipulation useful utilities for value conversion and manipulation outsourced into own .c file for better code readability

```
#include <stdio.h>
#include <math.h>
#include "m_pd.h"
#include "purple_utils.h"
```

Include dependency graph for purple_utils.c:

Functions

- int get_samples_from_ms (int ms, float sr)

  calculates number of samples from ms and sr
- float get_ms_from_samples (int num_samples, float sr)

  calculates sample time in ms from num_samples and sr
- float get_interpolated_sample_value (float sample_left, float sample_right, float frac)

  calculates interpolated sample value calculates interpolated sample value between sample_left and sample_right
- void switch_float_values (float *a, float *b)

  swaps to values swaps to values of float type

6.7.1 Detailed Description

useful utilities for value conversion and manipulation useful utilities for value conversion and manipulation outsourced into own .c file for better code readability

Author

Nikita Kretschmar

Adrian Philipp

Micha Strobl

Tim Wennemann

Version

0.1

Date

2021-09-27

Copyright

6.7.2 Function Documentation

6.7.2.1 get_interpolated_sample_value()

```
float get_interpolated_sample_value (
            float sample_left,
            float sample_right,
            float frac )
```

calculates interpolated sample value calculates interpolated sample value between *sample_left* and *sample_right*

Parameters

sample_left	value at the beginning of sample
sample_right	value at the end of sample
frac	position after decimal point

Returns

float interpolated sample value

6.7.2.2 get_ms_from_samples()

```
float get_ms_from_samples (
            int num_samples,
            float sr )
```

calculates sample time in ms from *num_samples* and *sr*

Parameters

num_samples	number of samples
sr	defined samplerate

Returns

float sample time

6.7.2.3 get_samples_from_ms()

```
int get_samples_from_ms (
            int ms,
            float sr )
```

calculates number of samples from *ms* and *sr*

Parameters

ms	sample time in ms
sr	defined sample rate

Returns

int number of samples

6.7.2.4 switch_float_values()

```
void switch_float_values (
             float * a,
             float * b )
```

swaps to values swaps to values of float type

Parameters

a	first value to swapped with second
b	second value to be swappend with first