didgmo

0.3.2

Generated by Doxygen 1.7.4

Tue May 10 2011 22:09:17

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

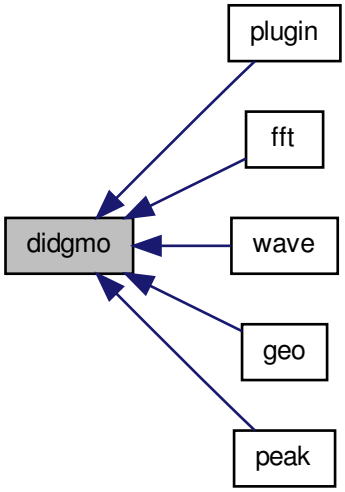
Chapter 2

Module Documentation

2.1 didgmo

this is the source code documentation of didgmo.

Collaboration diagram for didgmo:



Modules

- fft

Routines for interaction with fft objects.

- geo

  Routines for defining, loading, saving of the didgeridoo geometry.

- peak

  Routines for interaction with peak objects.

- plugin

  Routines for interacting with the plugin system.

- wave

  Routines for interaction with wave files.

2.1.1 Detailed Description

this is the source code documentation of didgmo. didgmo is programed in plain ansi C in an object oriented way. the first implementation of didgmo had a scripting support for Ruby, but then didgmo was redesigned for a better speed performance. to allow the user to extend the program with specific routines, didgmo makes use of plugins. the plugins are written in plain ansi C, too. in order to implement your own routines, you need to know about the structure of the program.

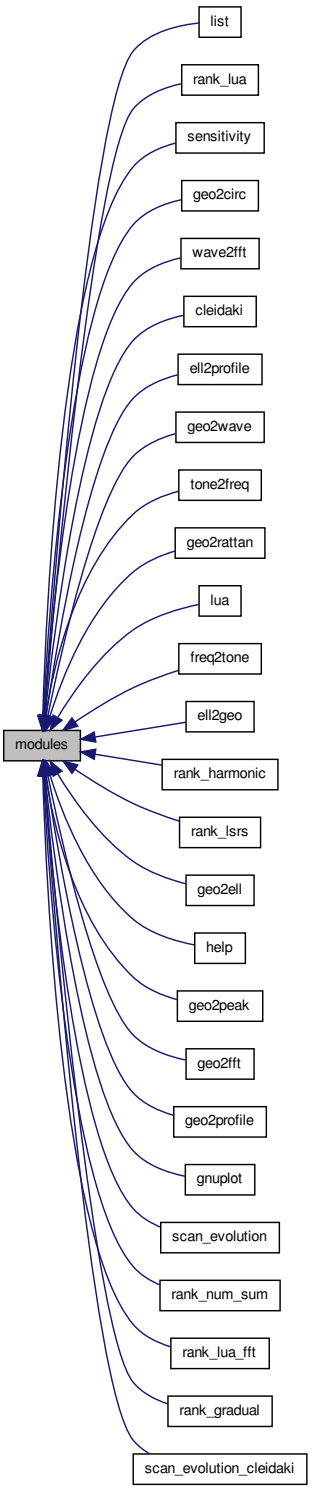
this information is intended for programmers, who like to write their own plugins.

you actually only have to know about three classes: geo, fft, wave. the geo class stores the didgeridoo geometry and includes routines for loading and saving. the fft class stores the impedance and sound spectra and includes routines for loading and saving. the wave object stores the information of a sound file and includes routines for loading, saving and playback. each class has additional routines, which allow for conversion between the different objects, the most prominent beeing the function geo_fft, which calculates the impedance and sound spectra for the given geometry and returns them in an object of the class fft.

2.2 modules

TODO.

Collaboration diagram for modules:



Modules

- cleidaki
- ell2geo
- ell2profile
- freq2tone
- geo2circ
- geo2ell
- geo2fft
- geo2peak
- geo2profile
- geo2rattan
- geo2wave
- gnuplot
- help
- list
- lua
- rank_gradual
- rank_harmonic
- rank_lsrs
- rank_lua
- rank_lua_fft
- rank_num_sum
- scan_evolution
- scan_evolution_cleidaki
- sensitivity

  starts a graphical user interface for interactive sensitivity analysis
- tone2freq
- wave2fft

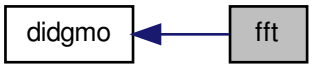
2.2.1 Detailed Description

TODO.

2.3 fft

Routines for interaction with fft objects.

Collaboration diagram for fft:



Functions

- fft ∗ fft_new (int i_size)

  creates fft object with frequency range from 0 to i_size.
- void fft_free (fft ∗i_fft)

  frees fft object.
- wave ∗ fft_wave (fft ∗i_fft)

  creates wave object with reverse fourier transformation.
- fft ∗ fft_dB (fft ∗i_fft)

  creates fft object with sound spectra of ground tone and first overblow expressed in dB.
- void fft_save (fft ∗i_fft, char ∗i_path)

  save fft object to file.
- fft ∗ fft_load (char ∗i_path, int i_size)

  load fft object from file.
- peak ∗ fft_peak (fft ∗i_fft, int i_id)

  extract peak information from fft object.

2.3.1 Detailed Description

Routines for interaction with fft objects. create initial fft object for a range of 800Hz

```
const int fft_size = 800;
fft *fft_obj = fft_new (fft_size);
```

save your fft object

```
const char *fft_path = "my_obj.fft";
fft_save (fft_obj, fft_path)
```

free your fft object, when you don't need it anymore

```
fft_free (fft_obj);
```

load an fft object from a file

```
fft_obj = fft_load (path);
```

convert the amplitudes to dB

```
fft *fft_obj_dB = fft_dB (fft_obj);
fft_free (fft_obj_dB);
```

convert fft object to wave object via inverse fast fourier transformation

```
wave *wave_obj = fft_wave (fft_obj);
wave_free (wave_obj);
```

extract peaks of impedance (0), base (1) and overblow (2) sound spectra from fft object

```
peak *peak_obj_imp = fft_peak (fft_obj, 0);
peak *peak_obj_bas = fft_peak (fft_obj, 1);
peak *peak_obj_ovr = fft_peak (fft_obj, 2);
peak_free (peak_obj_imp);
peak_free (peak_obj_bas);
peak_free (peak_obj_ovr);
```

free your fft object, when you don't need it anymore

```
fft_free (fft_obj);
```

2.3.2 Function Documentation

2.3.2.1 fft* fft_new (int *i_size*)

creates fft object with frequency range from 0 to i_size.

Parameters

i_size	frequency range.

Returns

fft object (free it with fft_free)

2.3.2.2 void fft_free (fft * *i_fft*)

frees fft object.

Parameters

i_fft	fft object

2.3.2.3 wave* fft_wave (fft * *i_fft*)

creates wave object with reverse fourier transformation.

Parameters

i_fft	fft object

Returns

wave object (free it with wave_free)

2.3.2.4 fft* fft_dB (fft * *i_fft*)

creates fft object with sound spectra of ground tone and first overblow expressed in dB.

Parameters

i_fft	fft object

Returns

fft object (free it with fft_free);

2.3.2.5 void fft_save (fft * *i_fft,* char * *i_path*)

save fft object to file.

Parameters

i_fft	fft object
i_path	file path to save to

2.3.2.6 fft* fft_load (char * *i_path,* int *i_size*)

load fft object from file.

Parameters

i_path	path to load from
i_size	frequency range to load

Returns

fft object (free it with fft_free)

2.3.2.7 peak* fft_peak (fft * *i_fft,* int *i_id*)

extract peak information from fft object.

Parameters

i_fft	fft object
i_id	extract peaks from impedance spectrum (0), sound spectrum from ground tone (1) or sound spectrum from first overblow (2)

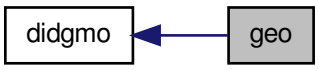
Returns

peak object (free it with peak_free)

2.4 geo

Routines for defining, loading, saving of the didgeridoo geometry.

Collaboration diagram for geo:



Functions

- seg * seg_new (long double i_L, long double i_d0, long double i_d1)
  creates new segment.
- void seg_update (seg *i_seg)
  updates private data of a segment.
- seg * seg_copy (seg *i_seg)
  copies a segment.
- void seg_free (seg *i_seg)
  frees segment created with seg_new.
- Eina_List * geo_new ()
  creates new geometry structure.
- void geo_free (Eina_List *i_geo)
  frees geometry created with geo_new.
- long double geo_length_get (Eina_List *i_geo)
  get length of final didgeridoo represented by geometry.

- Eina_List ∗ geo_load (char ∗i_path)

  load geometry (∗.geo) from file.
- Eina_List ∗ geo_copy (Eina_List ∗i_geo)

  copy a geometry.
- fft ∗ geo_fft (Eina_List ∗i_geo, int i_max, int i_offset)

  calculate whole impedance, sound and overblow spectra of given geometry.

2.4.1 Detailed Description

Routines for defining, loading, saving of the didgeridoo geometry.

2.4.2 Function Documentation

2.4.2.1 seg∗ seg_new (long double *i_L*, long double *i_d0*, long double *i_d1*)

creates new segment.

Parameters

i_L	length of segment
i_d0	diameter at beginning of segment
i_d1	diameter at end of segment

Returns

a new segment (free it with seg_free)

2.4.2.2 void seg_update (seg ∗ *i_seg*)

updates private data of a segment.

Parameters

i_seg	segment to be updated

2.4.2.3 seg∗ seg_copy (seg ∗ *i_seg*)

copies a segment.

Parameters

i_seg	segment to be copied

Returns

a new segment (free it with seg_free)

2.4.2.4 void seg_free (seg * *i_seg*)

frees segment created with seg_new.

Parameters

i_seg	segment to be freed

2.4.2.5 Eina_List* geo_new ( )

creates new geometry structure.

Returns

geometry (free it with geo_free)

2.4.2.6 void geo_free (Eina_List * *i_geo*)

frees geometry created with geo_new.

Parameters

i_geo	geometry to be freed

routine frees automatically all associated segments.

2.4.2.7 long double geo_length_get (Eina_List * *i_geo*)

get length of final didgeridoo represented by geometry.

Parameters

i_geo	geometry

Returns

sum of length of all segments

2.4.2.8 Eina_List* geo_load (char * *i_path*)

load geometry (*.geo) from file.

Parameters

i_path	file path

Returns

geometry

2.4.2.9 Eina_List* geo_copy (Eina_List * *i_geo*)

copy a geometry.

Parameters

i_geo	geometry to copy

Returns

new geometry (free with geo_free)

2.4.2.10 fft* geo_fft (Eina_List * *i_geo,* int *i_max,* int *i_offset*)

calculate whole impedance, sound and overblow spectra of given geometry.

Parameters

i_geo	geometry
i_max	frequency range from 0 to i_max
i_offset	frequency offset of ground tone and first overblow

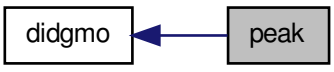
Returns

fft spectra

2.5 peak

Routines for interaction with peak objects.

Collaboration diagram for peak:



Functions

- peak ∗ peak_new (int i_min_size, int i_max_size)

  creates new peak object.
- void peak_free (peak ∗i_peak)

  frees peak object.
- void peak_save (peak ∗i_peak, char ∗i_path)

  saves peak object to file
- peak ∗ peak_load (char ∗i_path)

  load peak object from file
- char ∗ peak_tone (double i_freq, int ∗o_cents, int ∗o_octave)

  gives name of tone corresponding to frequency
- double peak_freq (const char ∗i_tone, int i_cents, int i_octave)

  gives freqency of given tone and cents

2.5.1 Detailed Description

Routines for interaction with peak objects.

2.5.2 Function Documentation

2.5.2.1 peak∗ peak_new (int *i_min_size*, int *i_max_size*)

creates new peak object.

Parameters

i_min_size	number of 'valleys'
i_max_size	number of 'peaks'

Returns

peak object (free with peak_free)

2.5.2.2 void peak_free (peak ∗ *i_peak*)

frees peak object.

Parameters

i_peak	peak object

2.5.2.3 void peak_save (peak * *i_peak*, char * *i_path*)

saves peak object to file

Parameters

i_peak	peak object
i_path	file path

2.5.2.4 peak* peak_load (char * *i_path*)

load peak object from file

Parameters

i_path	file path

Returns

peak object (free with peak_free)

2.5.2.5 char* peak_tone (double *i_freq*, int * *o_cents*, int * *o_octave*)

gives name of tone corresponding to frequency

Parameters

i_freq	frequency
o_cents	cents
o_octave	octave

Returns

name of tone

2.5.2.6 double peak_freq (const char * *i_tone*, int *i_cents*, int *i_octave*)

gives freqency of given tone and cents

@ param i_tone character string of tone @ param i_cents number of cents

@ return frequency of tone

2.6 plugin

Routines for interacting with the plugin system.

Collaboration diagram for plugin:

Functions

- plugin ∗ plugin_new (char ∗i_name)
  creates new plugin instance.
- void plugin_free (plugin ∗i_plugin)
  frees plugin instance.
- void plugin_parse (plugin_info ∗i_plugin_info, int i_argc, char ∗∗i_argv)
  parses command line arguments
- plugin_info ∗ plugin_get_info (plugin ∗i_plugin)
  runs plugin info
- void plugin_init (plugin ∗i_plugin, int i_argc, char ∗∗i_argv)
  runs plugin init routine.
- void plugin_deinit (plugin ∗i_plugin)
  runs plugin deinit routine.
- void ∗ plugin_run (plugin ∗i_plugin, void ∗i_data)
  runs plugin main routine.
- void plugin_info_print (plugin_info ∗i_plugin_info)
  print plugin information to stdout
- plugin_return ∗ plugin_return_new (int i_param_number)
  create new plugin_return object
- void plugin_return_free (plugin_return ∗i_plugin_return)
  free plugin_return object

2.6.1 Detailed Description

Routines for interacting with the plugin system.

2.6.2 Function Documentation

2.6.2.1 plugin∗ plugin_new (char ∗ *i_name*)

creates new plugin instance.

Parameters

i_name	name of plugin

Returns

plugin instance (free with plugin_free)

2.6.2.2 void plugin_free (plugin * *i_plugin*)

frees plugin instance.

Parameters

i_plugin	plugin instance

2.6.2.3 void plugin_parse (plugin_info * *i_plugin_info*, int *i_argc*, char ** *i_argv*)

parses command line arguments

Parameters

i_plugin_info	plugin info object
i_argc	number of arguments
i_argv	command line arguments

2.6.2.4 plugin_info* plugin_get_info (plugin * *i_plugin*)

runs plugin info

Parameters

i_plugin	plugin instance

Returns

plugin info structure

2.6.2.5 void plugin_init (plugin * *i_plugin*, int *i_argc*, char ** *i_argv*)

runs plugin init routine.

Parameters

i_plugin	plugin instance
i_argc	number of arguments
i_argv	array of argmuments

2.6.2.6 void* plugin_run (plugin * *i_plugin,* void * *i_data*)

runs plugin main routine.

Parameters

i_plugin	plugin instance
i_data	user data

2.6.2.7 void plugin_info_print (plugin_info * *i_plugin_info*)

print plugin information to stdout

Parameters

i_plugin_info	plugin_info instance

2.6.2.8 plugin_return* plugin_return_new (int *i_param_number*)

create new plugin_return object

Parameters

i_param_number	number of parameters

free object with plugin_return _free

2.6.2.9 void plugin_return_free (plugin_return * *i_plugin_return*)

free plugin_return object

Parameters

i_plugin_return	plugin_return object

2.7 wave

Routines for interaction with wave files.

Collaboration diagram for wave:

Functions

- wave ∗ wave_new (int i_size)

  creates new wave object with size i_size.
- void wave_free (wave ∗i_wave)

  frees wave object.
- wave ∗ wave_load (char ∗i_path, int i_sec)

  loads given wave file.
- void wave_save (wave ∗i_wave, char ∗i_path)

  saves given wave file.
- fft ∗ wave_fft (wave ∗i_wave, int i_freq)

  fast fourier transformation of wave object to a fft spectrum.
- void wave_play (wave ∗i_wave)

  play back wave object on sound card.

2.7.1 Detailed Description

Routines for interaction with wave files.

2.7.2 Function Documentation

2.7.2.1 wave∗ wave_new (int *i_size*)

creates new wave object with size i_size.

Parameters

i_size	size of wave array (bitrate x duration)

Returns

wave object (free with wave_free)

2.7.2.2 void wave_free (wave * *i_wave*)

frees wave object.

Parameters

i_wave	wave object

2.7.2.3 wave* wave_load (char * *i_path,* int *i_sec*)

loads given wave file.

Parameters

i_path	file path
i_sec	duration to read from wave file

Returns

wave object

2.7.2.4 void wave_save (wave * *i_wave,* char * *i_path*)

saves given wave file.

Parameters

i_wave	wave object
i_path	file path

2.7.2.5 fft* wave_fft (wave * *i_wave,* int *i_freq*)

fast fourier transformation of wave object to a fft spectrum.

Parameters

i_wave	wave object
i_freq	frequency range

Returns

fft spectrum (only sound spectrum is defined)

2.7.2.6 void wave_play (wave * *i_wave*)

play back wave object on sound card.

Parameters

i_wave	wave object

2.8 cleidaki

Collaboration diagram for cleidaki:

Parameters

name	name of modelling project
dec	decrease of length used in algorithm

Returns

NULL

2.9 ell2geo

Collaboration diagram for ell2geo:

Parameters

name	name of modelling project

Returns

NULL

2.10 ell2profile

Collaboration diagram for ell2profile:

Parameters

name	name of modelling project
step	step size

Returns

NULL

2.11 freq2tone

Collaboration diagram for freq2tone:

Parameters

frequency	frequency of tone

Returns

NULL

2.12 geo2circ

Collaboration diagram for geo2circ:

Parameters

name	name of modelling project

Returns

NULL

2.13 geo2ell

Collaboration diagram for geo2ell:

Parameters

name	name of modelling project
type	(1) constant shorter diameter (2) constant fraction
value	value according to type

Returns

NULL

2.14 geo2fft

Collaboration diagram for geo2fft:

Parameters

name	name of modelling project
freq_max	maximal frequency [Hz] to simulate spectrum up to

Returns

NULL

2.15 geo2peak

Collaboration diagram for geo2peak:

Parameters

name	name of modelling project
freq_max	maximal frequency [Hz] to simulate spectrum up to

Returns

NULL

2.16 geo2profile

Collaboration diagram for geo2profile:

Parameters

name	name of modelling project
step	step size

Returns

NULL

2.17 geo2rattan

Collaboration diagram for geo2rattan:

Parameters

name	name of modelling project
thickness	wall thickness of didgeridoo
width	width of rattan string

Returns

NULL

2.18 geo2wave

Collaboration diagram for geo2wave:

Parameters

name	name of modelling project
freq_max	maximal frequency [Hz] to simulate spectrum up to

2.19 gnuplot

Collaboration diagram for gnuplot:

Parameters

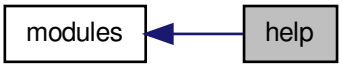
name	name of modelling project
template	name of gnuplot template
terminal	name of gnuplot terminal

Returns

NULL

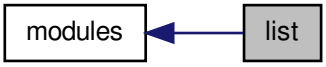
2.20 help

Collaboration diagram for help:



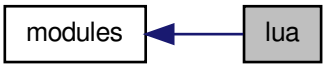
2.21 list

Collaboration diagram for list:



2.22 lua

Collaboration diagram for lua:



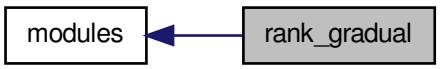
Parameters

path	path to lua script file

num_params	number of script parameters

2.23 rank_gradual

Collaboration diagram for rank_gradual:



Parameters

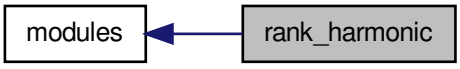
freq_low	lower limit of frequencies
freq_high	higher limit of frequencies
num_peak	number of peaks that shall be optimized

Returns

the actual ranking

2.24 rank_harmonic

Collaboration diagram for rank_harmonic:



Parameters

num	number of frequency peaks to optimize for

Returns

the actual ranking

2.25 rank_lsrs

Collaboration diagram for rank_lsrs:

Parameters

name	path to file containing fft specturm to fit the didg geometry to

Returns

the actual ranking

2.26 rank_lua

Collaboration diagram for rank_lua:

Parameters

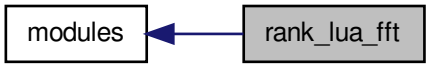
path	path to lua description file

Returns

the actual ranking

2.27 rank_lua_fft

Collaboration diagram for rank_lua_fft:



Parameters

path	path to lua description file

Returns

the actual ranking

This module can be used with all the scan_∗ modules. The rank function here is written in Lua and provided to this module by the file name of the Lua script. In the Lua script, there is a function rank, which takes as argument the fft spectrum of a given didg geometry as a table and returns just one number, the rank. Command line arguments can be given to the Lua script, they are accessible by the table 'arg' in the script.

Example invocation on the cli:

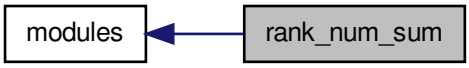
```
didgmo scan_evolution [scan_options] rank_lua_fft {my_script.lua} [rank_options]
```

Example Lua script:

```
-- The options given to the cli as the 'rank_options' are accessible in the table 'arg'
-- Here we read two command line arguments
-- The example invocation on the cli: didgmo scan_evolution [scan_options] rank_lua_fft my_s

argument1 = arg[1] -- 11
argument2 = arg[2] -- 22

function rank (fft)
  local rank

  -- do your ranking here based on the provided command line arguments and the provided fft
  -- and finally return the rank of this fft spectrum.
  -- the higher the rank value, the better the geometry reflects your desired sound characte
  -- avoid giving  binary ranks, e.g. good (1) and bad (0) based on conditionals, use real n

  return rank
end
```

2.28 rank_num_sum

Collaboration diagram for rank_num_sum:



Parameters

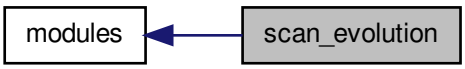
freq_low	lower limit of frequencies
freq_high	higher limit of frequencies

Returns

the actual ranking

2.29 scan_evolution

Collaboration diagram for scan_evolution:



Parameters

name	name of modelling project
l0	minimal length of didg
l1	maximal length of didg
ns	number of segments
np	size of pool
ni	number of iterations
d0	mouth diameter
d1	bell diameter
mf	mutation frequency

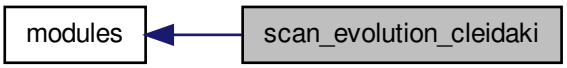
s0	minimal segment length
si	segment increment
dm	minimal diameter
di	diameter increment
fb	flag bell end variation
rank_mod	ranking module

Returns

NULL

2.30 scan_evolution_cleidaki

Collaboration diagram for scan_evolution_cleidaki:



Parameters

name	name of modelling project
l0	minimal length of didg
l1	maximal length of didg
ns	number of segments
np	size of pool
ni	number of iterations
d0	mouth diameter
d1	bell diameter
mf	mutation frequency
s0	minimal segment length
si	segment increment
dm	minimal diameter
di	diameter increment
fb	flag bell end variation
dec	decrease of length used in algorithm
rank_mod	ranking module

Returns

NULL

2.31 sensitivity

starts a graphical user interface for interactive sensitivity analysis

Collaboration diagram for sensitivity:

starts a graphical user interface for interactive sensitivity analysis

Parameters

i_name	project name

2.32 tone2freq

Collaboration diagram for tone2freq:

Parameters

tone	name of tone
cents	cents of variation
octave	octave of tone

Returns

NULL

2.33 wave2fft

Collaboration diagram for wave2fft:

Parameters

name	name of sound file
freq	maximal frequency [Hz] to output
dur	duration to scan through [s]

Returns

NULL

Index