

didgmo

0.3.2

Generated by Doxygen 1.7.4

Tue May 10 2011 22:09:17

Contents

1	Module Index	1
1.1	Modules	1
2	Module Documentation	3
2.1	didgmo	3
2.1.1	Detailed Description	4
2.2	modules	4
2.2.1	Detailed Description	6
2.3	fft	6
2.3.1	Detailed Description	7
2.3.2	Function Documentation	8
2.3.2.1	fft_new	8
2.3.2.2	fft_free	8
2.3.2.3	fft_wave	9
2.3.2.4	fft_dB	9
2.3.2.5	fft_save	9
2.3.2.6	fft_load	9
2.3.2.7	fft_peak	10
2.4	geo	10
2.4.1	Detailed Description	11
2.4.2	Function Documentation	11
2.4.2.1	seg_new	11
2.4.2.2	seg_update	11
2.4.2.3	seg_copy	11
2.4.2.4	seg_free	12
2.4.2.5	geo_new	12

2.4.2.6	geo_free	12
2.4.2.7	geo_length_get	12
2.4.2.8	geo_load	12
2.4.2.9	geo_copy	13
2.4.2.10	geo_fft	13
2.5	peak	13
2.5.1	Detailed Description	14
2.5.2	Function Documentation	14
2.5.2.1	peak_new	14
2.5.2.2	peak_free	14
2.5.2.3	peak_save	15
2.5.2.4	peak_load	15
2.5.2.5	peak_tone	15
2.5.2.6	peak_freq	15
2.6	plugin	15
2.6.1	Detailed Description	16
2.6.2	Function Documentation	16
2.6.2.1	plugin_new	16
2.6.2.2	plugin_free	17
2.6.2.3	plugin_parse	17
2.6.2.4	plugin_get_info	17
2.6.2.5	plugin_init	17
2.6.2.6	plugin_run	18
2.6.2.7	plugin_info_print	18
2.6.2.8	plugin_return_new	18
2.6.2.9	plugin_return_free	18
2.7	wave	18
2.7.1	Detailed Description	19
2.7.2	Function Documentation	19
2.7.2.1	wave_new	19
2.7.2.2	wave_free	20
2.7.2.3	wave_load	20
2.7.2.4	wave_save	20
2.7.2.5	wave_fft	20

2.7.2.6	wave_play	20
2.8	cleidaki	21
2.9	ell2geo	21
2.10	ell2profile	22
2.11	freq2tone	22
2.12	geo2circ	23
2.13	geo2ell	23
2.14	geo2fft	24
2.15	geo2peak	24
2.16	geo2profile	25
2.17	geo2rattan	25
2.18	geo2wave	26
2.19	gnuplot	26
2.20	help	27
2.21	list	27
2.22	lua	27
2.23	rank_gradual	28
2.24	rank_harmonic	28
2.25	rank_lsrs	29
2.26	rank_lua	29
2.27	rank_lua_fft	30
2.28	rank_num_sum	31
2.29	scan_evolution	31
2.30	scan_evolution_cleidaki	32
2.31	sensitivity	33
2.32	tone2freq	33
2.33	wave2fft	34

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

didgmo	3
fft	6
geo	10
peak	13
plugin	15
wave	18
modules	4
cleidaki	21
ell2geo	21
ell2profile	22
freq2tone	22
geo2circ	23
geo2ell	23
geo2fft	24
geo2peak	24
geo2profile	25
geo2rattan	25
geo2wave	26
gnuplot	26
help	27
list	27
lua	27
rank_gradual	28
rank_harmonic	28
rank_lsrs	29
rank_lua	29
rank_lua_fft	30
rank_num_sum	31
scan_evolution	31

scan_evolution_cleidaki	32
sensitivity	33
tone2freq	33
wave2fft	34

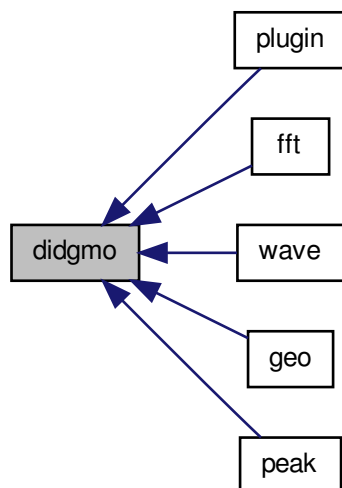
Chapter 2

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 programmed 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.

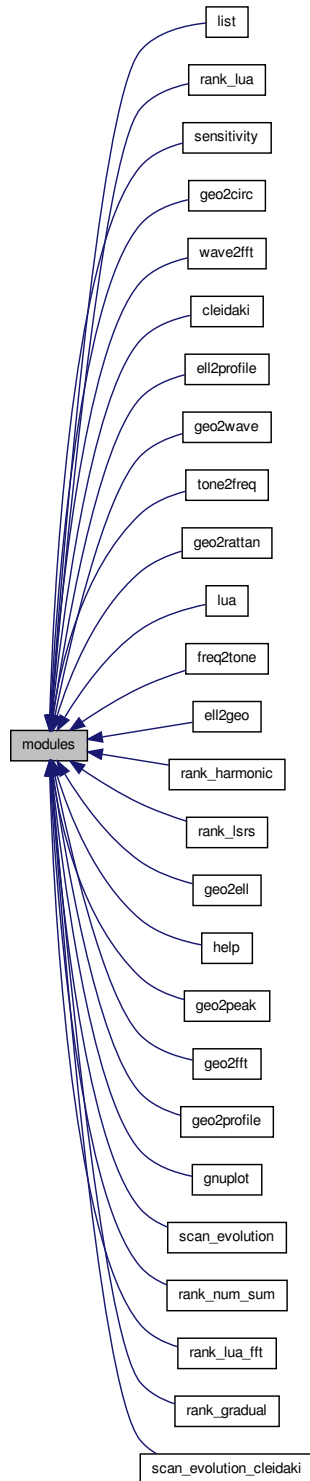
this information is intended for programmers, who like to write their own plugins.

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 being 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](#)

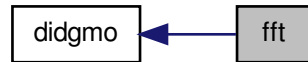
2.2.1 Detailed Description

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>i_size</i>	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>i_fft</i>	fft object
--------------	------------

2.3.2.3 wave* fft_wave (fft * *i_fft*)

creates wave object with reverse fourier transformation.

Parameters

<i>i_fft</i>	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>i_fft</i>	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>i_fft</i>	fft object
<i>i_path</i>	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>i_path</i>	path to load from
<i>i_size</i>	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>i_fft</i>	fft object
<i>i_id</i>	extract peaks from impedance spectrum (0), sound spectrum from ground tone (1) or sound spectrum from first overblow (2)

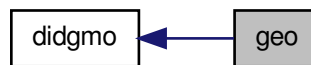
Returns

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

<code>i_L</code>	length of segment
<code>i_d0</code>	diameter at beginning of segment
<code>i_d1</code>	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

<code>i_seg</code>	segment to be updated
--------------------	-----------------------

2.4.2.3 `seg* seg_copy (seg * i_seg)`

copies a segment.

Parameters

<code>i_seg</code>	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>i_seg</i>	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>i_geo</i>	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>i_geo</i>	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>i_path</i>	file path
---------------	-----------

Returns

geometry

2.4.2.9 Eina_List* geo_copy (Eina_List * *i_geo*)

copy a geometry.

Parameters

<i>i_geo</i>	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>i_geo</i>	geometry
<i>i_max</i>	frequency range from 0 to <i>i_max</i>
<i>i_offset</i>	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 frequency 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>i_min_size</i>	number of 'valleys'
<i>i_max_size</i>	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>i_peak</i>	peak object
---------------	-------------

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

saves peak object to file

Parameters

<code>i_peak</code>	peak object
<code>i_path</code>	file path

2.5.2.4 `peak*` `peak_load` (`char * i_path`)

load peak object from file

Parameters

<code>i_path</code>	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

<code>i_freq</code>	frequency
<code>o_cents</code>	cents
<code>o_octave</code>	octave

Returns

name of tone

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

gives frequency 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>i_name</i>	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>i_plugin</i>	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>i_plugin_info</i>	plugin info object
<i>i_argc</i>	number of arguments
<i>i_argv</i>	command line arguments

2.6.2.4 `plugin_info* plugin_get_info (plugin * i_plugin)`

runs plugin info

Parameters

<i>i_plugin</i>	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>i_plugin</i>	plugin instance
<i>i_argc</i>	number of arguments
<i>i_argv</i>	array of arguments

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

runs plugin main routine.

Parameters

<i>i_plugin</i>	plugin instance
<i>i_data</i>	user data

2.6.2.7 void plugin_info_print (plugin_info * *i_plugin_info*)

print plugin information to stdout

Parameters

<i>i_plugin_info</i>	plugin_info instance
----------------------	----------------------

2.6.2.8 plugin_return* plugin_return_new (int *i_param_number*)

create new plugin_return object

Parameters

<i>i_param_number</i>	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>i_plugin_return</i>	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

<code>i_size</code>	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>i_wave</i>	wave object
---------------	-------------

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

loads given wave file.

Parameters

<i>i_path</i>	file path
<i>i_sec</i>	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>i_wave</i>	wave object
<i>i_path</i>	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>i_wave</i>	wave object
<i>i_freq</i>	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>i_wave</i>	wave object
---------------	-------------

2.8 cleidaki

Collaboration diagram for cleidaki:

**Parameters**

<i>name</i>	name of modelling project
<i>dec</i>	decrease of length used in algorithm

Returns

NULL

2.9 ell2geo

Collaboration diagram for ell2geo:

**Parameters**

<i>name</i>	name of modelling project
-------------	---------------------------

Returns

NULL

2.10 ell2profile

Collaboration diagram for ell2profile:



Parameters

<i>name</i>	name of modelling project
<i>step</i>	step size

Returns

NULL

2.11 freq2tone

Collaboration diagram for freq2tone:



Parameters

<i>frequency</i>	frequency of tone
------------------	-------------------

Returns

NULL

2.12 geo2circ

Collaboration diagram for geo2circ:



Parameters

<i>name</i>	name of modelling project
-------------	---------------------------

Returns

NULL

2.13 geo2ell

Collaboration diagram for geo2ell:



Parameters

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

Returns

NULL

2.14 geo2fft

Collaboration diagram for geo2fft:



Parameters

<i>name</i>	name of modelling project
<i>freq_max</i>	maximal frequency [Hz] to simulate spectrum up to

Returns

NULL

2.15 geo2peak

Collaboration diagram for geo2peak:



Parameters

<i>name</i>	name of modelling project
<i>freq_max</i>	maximal frequency [Hz] to simulate spectrum up to

Returns

NULL

2.16 geo2profile

Collaboration diagram for geo2profile:



Parameters

<i>name</i>	name of modelling project
<i>step</i>	step size

Returns

NULL

2.17 geo2rattan

Collaboration diagram for geo2rattan:



Parameters

<i>name</i>	name of modelling project
<i>thickness</i>	wall thickness of didgeridoo
<i>width</i>	width of rattan string

Returns

NULL

2.18 geo2wave

Collaboration diagram for geo2wave:



Parameters

<i>name</i>	name of modelling project
<i>freq_max</i>	maximal frequency [Hz] to simulate spectrum up to

2.19 gnuplot

Collaboration diagram for gnuplot:



Parameters

<i>name</i>	name of modelling project
<i>template</i>	name of gnuplot template
<i>terminal</i>	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

<i>path</i>	path to lua script file
-------------	-------------------------

<i>num_params</i>	number of script parameters
-------------------	-----------------------------

2.23 rank_gradual

Collaboration diagram for rank_gradual:



Parameters

<i>freq_low</i>	lower limit of frequencies
<i>freq_high</i>	higher limit of frequencies
<i>num_peak</i>	number of peaks that shall be optimized

Returns

the actual ranking

2.24 rank_harmonic

Collaboration diagram for rank_harmonic:



Parameters

<i>num</i>	number of frequency peaks to optimize for
------------	---

Returns

the actual ranking

2.25 rank_lsrs

Collaboration diagram for rank_lsrs:

**Parameters**

<i>name</i>	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**

<i>path</i>	path to lua description file
-------------	------------------------------

Returns

the actual ranking

2.27 rank_lua_fft

Collaboration diagram for rank_lua_fft:



Parameters

<i>path</i>	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 character
  -- 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

<i>freq_low</i>	lower limit of frequencies
<i>freq_high</i>	higher limit of frequencies

Returns

the actual ranking

2.29 scan_evolution

Collaboration diagram for scan_evolution:



Parameters

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

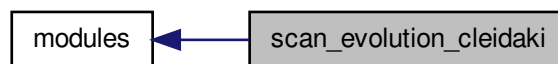
<i>s0</i>	minimal segment length
<i>si</i>	segment increment
<i>dm</i>	minimal diameter
<i>di</i>	diameter increment
<i>fb</i>	flag bell end variation
<i>rank_mod</i>	ranking module

Returns

NULL

2.30 scan_evolution_cleidaki

Collaboration diagram for scan_evolution_cleidaki:

**Parameters**

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

2.32 tone2freq

Collaboration diagram for tone2freq:



Parameters

<i>tone</i>	name of tone
<i>cents</i>	cents of variation
<i>octave</i>	octave of tone

Returns

NULL

2.33 wave2fft

Collaboration diagram for wave2fft:



Parameters

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

Returns

NULL

Index

- cleidaki, [22](#)
- didgmo, [3](#)
- ell2geo, [23](#)
- ell2profile, [23](#)
- eog2geo, [24](#)
- eog_load
 - geo, [14](#)
- fft, [6](#)
 - fft_dB, [9](#)
 - fft_free, [8](#)
 - fft_load, [9](#)
 - fft_new, [8](#)
 - fft_peak, [9](#)
 - fft_save, [9](#)
 - fft_wave, [8](#)
- fft_dB
 - fft, [9](#)
- fft_free
 - fft, [8](#)
- fft_load
 - fft, [9](#)
- fft_new
 - fft, [8](#)
- fft_peak
 - fft, [9](#)
- fft_save
 - fft, [9](#)
- fft_wave
 - fft, [8](#)
- Freq2tone, [24](#)
- geo, [10](#)
 - eog_load, [14](#)
 - geo_copy, [14](#)
 - geo_fft, [15](#)
 - geo_free, [12](#)
 - geo_get_length, [13](#)
 - geo_get_nth, [14](#)
 - geo_length_get, [13](#)
 - geo_load, [14](#)
 - geo_new, [12](#)
 - geo_pop_head, [13](#)
 - geo_pop_tail, [13](#)
 - geo_push_head, [12](#)
 - geo_push_tail, [12](#)
 - geo_save, [14](#)
 - seg_copy, [11](#)
 - seg_free, [12](#)
 - seg_new, [11](#)
 - seg_update, [11](#)
- geo2circ, [25](#)
- geo2ell, [25](#)
- geo2fft, [26](#)
- geo2peak, [26](#)
- geo2profile, [27](#)
- geo2rattan, [27](#)
- geo2wave, [28](#)
- geo_copy
 - geo, [14](#)
- geo_fft
 - geo, [15](#)
- geo_free
 - geo, [12](#)
- geo_get_length
 - geo, [13](#)
- geo_get_nth
 - geo, [14](#)
- geo_length_get
 - geo, [13](#)
- geo_load
 - geo, [14](#)
- geo_new
 - geo, [12](#)
- geo_pop_head
 - geo, [13](#)
- geo_pop_tail
 - geo, [13](#)
- geo_push_head
 - geo, [12](#)
- geo_push_tail

- geo, [12](#)
- geo_save
 - geo, [14](#)
- gnuplot, [28](#)
- gui, [28](#)
- help, [29](#)
- list, [29](#)
- lua, [30](#)
- modules, [4](#)
- peak, [15](#)
 - peak_free, [16](#)
 - peak_freq, [17](#)
 - peak_load, [16](#)
 - peak_new, [16](#)
 - peak_save, [16](#)
 - peak_tone, [17](#)
- peak_free
 - peak, [16](#)
- peak_freq
 - peak, [17](#)
- peak_load
 - peak, [16](#)
- peak_new
 - peak, [16](#)
- peak_save
 - peak, [16](#)
- peak_tone
 - peak, [17](#)
- plugin, [17](#)
 - plugin_free, [18](#)
 - plugin_get_info, [19](#)
 - plugin_info_print, [19](#)
 - plugin_init, [19](#)
 - plugin_new, [18](#)
 - plugin_parse, [19](#)
 - plugin_return_free, [20](#)
 - plugin_return_new, [20](#)
 - plugin_run, [19](#)
- plugin_free
 - plugin, [18](#)
- plugin_get_info
 - plugin, [19](#)
- plugin_info_print
 - plugin, [19](#)
- plugin_init
 - plugin, [19](#)
- plugin_new
 - plugin, [18](#)
- plugin_parse
 - plugin, [19](#)
- plugin_return_free
 - plugin, [20](#)
- plugin_return_new
 - plugin, [20](#)
- plugin_run
 - plugin, [19](#)
- rank_gradual, [30](#)
- rank_lsrs, [31](#)
- rank_lua, [31](#)
- rank_lua_fft, [32](#)
- rank_num_sum, [33](#)
- rank_rel_sum, [33](#)
- scan_conical, [34](#)
- scan_evolution, [34](#)
- scan_intelligent, [36](#)
- scan_segmental, [36](#)
- seg_copy
 - geo, [11](#)
- seg_free
 - geo, [12](#)
- seg_new
 - geo, [11](#)
- seg_update
 - geo, [11](#)
- sensitivity, [37](#)
- Tone2freq, [37](#)
- wave, [20](#)
 - wave_fft, [22](#)
 - wave_free, [21](#)
 - wave_load, [21](#)
 - wave_new, [21](#)
 - wave_play, [22](#)
 - wave_save, [21](#)
- wave2fft, [38](#)
- wave_fft
 - wave, [22](#)
- wave_free
 - wave, [21](#)
- wave_load
 - wave, [21](#)
- wave_new
 - wave, [21](#)

wave_play
 wave, [22](#)
wave_save
 wave, [21](#)