Nyoka Sound Library

alfa 1

Generated by Doxygen 1.8.3.1

Mon Oct 7 2013 13:07:18

Contents

1 File Index **1**

1.1 File List 1

2 File Documentation **3**

2.1 cosft.cpp File Reference 3

2.1.1 Function Documentation 3

2.1.1.1 complete_arr 3

2.1.1.2 cosft 4

2.1.1.3 get_array_lenght 4

2.2 fftwest.cpp File Reference 4

2.2.1 Function Documentation 4

2.2.1.1 fft_west 4

2.2.1.2 ifft_west 5

2.3 freqfilters.cpp File Reference 5

2.3.1 Macro Definition Documentation 5

2.3.1.1 PI 5

2.3.2 Function Documentation 5

2.3.2.1 bandpass 5

2.3.2.2 highpass 5

2.3.2.3 lowpass 5

2.4 noise.cpp File Reference 6

2.4.1 Function Documentation 6

2.4.1.1 whitenoise 6

2.5 ogg_vorbis.cpp File Reference 6

2.5.1 Macro Definition Documentation 7

2.5.1.1 READ 7

2.5.2 Function Documentation 7

2.5.2.1 decode 7

2.5.2.2 encoder 7

2.5.2.3 samples_number 8

2.5.2.4 samples_rate 8

2.5.2.5 total_time 8
2.6 pitch_filter.cpp File Reference 8
2.6.1 Macro Definition Documentation 9
2.6.1.1 PI 9
2.6.1.2 PITCH 9
2.6.1.3 UNPITCH 9
2.6.2 Function Documentation 9
2.6.2.1 pitch_filter 9
2.7 plot.cpp File Reference 9
2.7.1 Function Documentation 9
2.7.1.1 plot_freq 10
2.7.1.2 plot_time 10
2.8 sinft.cpp File Reference 10
2.8.1 Function Documentation 10
2.8.1.1 complete_arr 10
2.8.1.2 get_array_lenght 10
2.8.1.3 sinft 10

Index **10**

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

cosft.cpp 3
fftwest.cpp 4
freqfilters.cpp 5
noise.cpp 6
ogg_vorbis.cpp 6
pitch_filter.cpp 8
plot.cpp 9
sinft.cpp 10

Chapter 2

File Documentation

2.1 cosft.cpp File Reference

```
#include <stdlib.h>
#include <iostream>
#include <math.h>
#include <complex>
#include <fftwest.hh>
```

Functions

- int get_array_lenght (double ∗arr, const int size)

  The function found the number power of two, highest and closest to the size of the array.
- void complete_arr (double ∗arr_in, double ∗arr_out, const int size)

  This function fill an array power of two with other smaller or equal array and with zeros in extra spaces.
- void cosft (double ∗y, complex< double > ∗rbuffer, const int size)

  :Is the function in charge to realize the cosine transform the one that you can obtain with equation $f_j = {k=0}^{n-1} x_k [{}{n} j (k+{1}{2})]$

2.1.1 Function Documentation

2.1.1.1 complete_arr (double ∗ *arr_in,* double ∗ *arr_out,* const int *size*)

This function fill an array power of two with other smaller or equal array and with zeros in extra spaces.

Parameters

:	"∗arr_in" is a pointer towards the smallest array with which will fill the biggest array.
:	"∗arr_out" is a pointer towards the biggest array where is goint to get out the smaller array one time that is complete with the zeros.
:	"size" is the size of the smaller array .

It is equivalent to the imaginary parts of a DFT of roughly twice the length, is a linear and invertible function

Parameters

:	"∗arr_in" is a pointer towards the smallest array with which will fill the biggest array.
:	"∗arr_out" is a pointer towards the biggest array where is goint to get out the smaller array one time that is complete with the zeros.
:	"size" is the size of the smaller array .

2.1.1.2 cosft (double $*$ *y*, complex$<$ double $>$ $*$ *rbuffer*, const int *size*)

:Is the function in charge to realize the cosine transform the one that you can obtain with equation $f_j = {k=0}^{n-1} x_k [{}{n} j (k+{1}{2})]$

: this funcion is variation of the fast fourier transform,this give the sum of cosine functions oscillating at different frequencies from lossy compression of audio and images

Parameters

:	"size" is the size of the array.
:	"$*$y" is a pointer towards an array that has to be of a size power of two.
:	"$*$rbuffer" is a pointer towards an array where is going to return the cosine transform into other array of complex numbers.

2.1.1.3 get_array_lenght (double $*$ *arr*, const int *size*)

The function found the number power of two, highest and closest to the size of the array.

The function found the highest closest number to the size of the array.

Parameters

size	is the size of the array

Returns

number result: Is the number power of two closest and bigger than size .

Parameters

size	is the size of the array

Returns

number result: Is the number power of two closest and bigger than size .

2.2 fftwest.cpp File Reference

```
#include <stdlib.h>
#include <iostream>
#include <complex>
#include <fftw3.h>
```

Functions

- void fft_west (int smpls_num, double $*$buffer, complex$<$ double $>$ $*$retbuffer)
- void ifft_west (int smpls_num, complex$<$ double $>$ $*$buffer, double $*$retbuffer)

2.2.1 Function Documentation

2.2.1.1 void fft_west (int *smpls_num*, double $*$ *buffer*, complex$<$ double $>$ $*$ *retbuffer*)

2.2.1.2 void ifft_west (int *smpls_num,* complex< double > * *buffer,* double * *retbuffer*)

2.3 freqfilters.cpp File Reference

Macros

- #define PI 3.141592654

Functions

- void lowpass (int smpls_num, int smpls_rate, double *buffer, double *lowbuff, double freq)

    The function attenuates only the highest frequencies to a determinated frequency in a wave.
- void highpass (int smpls_num, int smpls_rate, double *buffer, double *highbuff, double freq)

    The function attenuates only the lowest frequencies to a determinated frequency in a wave.
- void bandpass (int smpls_num, int smpls_rate, double *buffer, double *bandbuff, double lowfreq, double highfreq)

    The function in charge of attenuated frequecies outside of a given rack in a wave.

2.3.1 Macro Definition Documentation

2.3.1.1 #define PI 3.141592654

2.3.2 Function Documentation

2.3.2.1 bandpass (int *smpls_num,* int *smpls_rate,* double * *buffer,* double * *bandbuff,* double *lowfreq,* double *highfreq*)

The function in charge of attenuated frequecies outside of a given rack in a wave.

Parameters

smpls_num	is the number of total samples.
smpls_rate	is the number of samples per second.
**buffer*	is a pointer towards the array whit the decoded song .
**bandbuff*	is a pointer towards an array where is going to send the attenuated frequencies.
lowfreq	is the is the minimun unattenuated frequency.
highfreq	is the is the maximun unattenuated frequency.

2.3.2.2 highpass (int *smpls_num,* int *smpls_rate,* double * *buffer,* double * *highbuff,* double *freq*)

The function attenuates only the lowest frequencies to a determinated frequency in a wave.

Parameters

:	*buffer is a pointer towards the array whit the decoded song .
:	*highbuff is a pointer towards an array where is going to send the attenuated frequencies.
:	freq is the is the minimun unattenuated frequency.
:	smpls_num es el tamaño del array buffer
:	smpls_rate is the number of samples per second.

2.3.2.3 lowpass (int *smpls_num,* int *smpls_rate,* double * *buffer,* double * *lowbuff,* double *freq*)

The function attenuates only the highest frequencies to a determinated frequency in a wave.

Parameters

:	∗buffer is a pointer towards the array whit the decoded song .
:	∗lowbuff is a pointer towards an array where is going to send the attenuated frequencies.
:	freq is the is the maximun unattenuated frequency.
:	smpls_num is the number of total samples.
:	smpls_rate is the number of samples per second.

2.4 noise.cpp File Reference

```
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <iostream>
```

Functions

- void whitenoise (int smpls_num, double ∗buffer)

  The function fill a pointer with aleatoriis random numbers which when they join the song they can be seen like noisy.

2.4.1 Function Documentation

2.4.1.1 whitenoise (int *smpls_num,* double ∗ *buffer*)

The function fill a pointer with aleatoriis random numbers which when they join the song they can be seen like noisy.

Parameters

:	∗buffer is a pointer towards an array where is going to return the ramdon numbers.
:	smpls_num is the number of total samples and is equal to size of buffer.

2.5 ogg_vorbis.cpp File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <iostream>
#include <vorbis/vorbisenc.h>
#include <vorbis/vorbisfile.h>
```

Macros

- #define READ 1000000

Functions

- int samples_number (const char ∗fname)

  The function receives a pointer towards to the name of the archive thar has to be .ogg and return the number of tatal samples.

- double total_time (const char ∗fname)

  The function receives a pointer towards to the name of the archive thar has to be .ogg and return the time of the song in seconds.
- double samples_rate (const char ∗fname)

  The function receives a pointer towards to the name of the archive thar has to be .ogg and return the samples for second.
- long decode (const char ∗fname, double ∗buffer, int smpls_num)

  The functiondecodes the song in a real array and send it to buffer this need the file are monochannel sound and be a .ogg.
- void encoder (const char ∗outfilename, double ∗inbuffer, int smpls_num, double smpls_rate)

  The function is the inverce to decoder this convert a array in a file .ogg using pointers.

2.5.1 Macro Definition Documentation

2.5.1.1 #define READ 1000000

2.5.2 Function Documentation

2.5.2.1 decode (const char ∗ *fname,* double ∗ *buffer,* int *smpls_num*)

The functiondecodes the song in a real array and send it to buffer this need the file are monochannel sound and be a .ogg.

Parameters

:	∗buffer is a pointer towards array where de decoder song are sent.
:	∗fname is a pointer towards the fist letter to the archive song.
:	smpls_num is the number of total samples and is equal to size of buffer.

Returns

the number of samples read

2.5.2.2 encoder (const char ∗ *outfilename,* double ∗ *inbuffer,* int *smpls_num,* double *smpls_rate*)

The function is the inverce to decoder this convert a array in a file .ogg using pointers.

Parameters

:	∗outfilename is pointer toward the first letter to file where send the array once encoder
:	∗inbuffer is a pointer to the array whit the song decoder or the file you want to convert.
:	∗fname is a pointer towards the fist letter to the archive song.
:	smpls_num is the number of total samples and is equal to size of buffer.
:	smpls_rate is the number of samples per second.

The function the function choose a centroid and does window as a bell of Hann,whit this one values all the points ariund the centroid and does the point if greatest amplitude pitch and the point of less aplitude unpitc,with this made groups around,start to calculate the news centroids using f2 and f3,repeat the proces until the centroids aren't moving or until 100 iterations, chages the size of the window exponentially dependin of the increment. is chosen the size of the window with more distance between the centroids

Parameters

:	∗buffer is a pointer towards array where de decoder song is saves.
:	percent is percent od samples per second.

:	increment is the increment and is given exponentially.
:	smpls_num is the number of total samples and is equal to size of buffer.
:	smpls_rate is the number of samples per second.

2.5.2.3 samples_number (const char * *fname*)

The function receives a pointer towards to the name of the archive thar has to be .ogg and return the number of tatal samples.

Parameters

:**fname*	is a pointer towards the fist letter to the archive song.

Returns

The number of samples in total song.

2.5.2.4 samples_rate (const char * *fname*)

The function receives a pointer towards to the name of the archive thar has to be .ogg and return the samples for second.

Parameters

:**fname*	is a pointer towards the fist letter to the archive song.

Returns

The samples for second of the song.

2.5.2.5 total_time (const char * *fname*)

The function receives a pointer towards to the name of the archive thar has to be .ogg and return the time of the song in seconds.

Parameters

:**fname*	is a pointer towards the fist letter to the archive song.

Returns

The total time in seconds song.

2.6 pitch_filter.cpp File Reference

```
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <cmath>
#include <iostream>
```

Macros

- #define PI 3.141592654
- #define PITCH 1
- #define UNPITCH 0

Functions

- void pitch_filter (int smpls_num, int smpls_rate, double ∗inbuffer, double percent, double increment)

  The function receives a pointer to real array and with their generates a graphyc respect to time.

2.6.1 Macro Definition Documentation

2.6.1.1 #define PI 3.141592654

2.6.1.2 #define PITCH 1

2.6.1.3 #define UNPITCH 0

2.6.2 Function Documentation

2.6.2.1 pitch_filter (int *smpls_num*, int *smpls_rate*, double ∗ *inbuffer*, double *percent*, double *increment*)

The function receives a pointer to real array and with their generates a graphyc respect to time.

The function receives a pointer towards an array of complex numbers and with their generates a graphhyc respect to the frequency.

Parameters

:	∗buffer is a pointer towards the array whit the decoded song .
:	smpls_read is the number of samples read .
:	smpls_rate is the number of samples per second.
∗buffer	is a pointer towards the array whit the decoded song .
:	smpls_read is the number of samples read .
:	smpls_rate is the number of samples per second.

2.7 plot.cpp File Reference

```
#include <iostream>
#include <stdlib.h>
#include <fstream>
#include <complex>
```

Functions

- void plot_time (double ∗buffer, int smpls_read, double smpls_rate)
- void plot_freq (complex< double > ∗buffer, int smpls_read, double smpls_rate)

2.7.1 Function Documentation

2.7.1.1 void plot_freq (complex< double > * *buffer*, int *smpls_read*, double *smpls_rate*)

2.7.1.2 void plot_time (double * *buffer*, int *smpls_read*, double *smpls_rate*)

2.8 sinft.cpp File Reference

```
#include <stdlib.h>
#include <iostream>
#include <math.h>
#include <complex>
#include <fftwest.hh>
```

Functions

- int get_array_lenght (const int size)
- void complete_arr (double *arr_in, double *arr_out, const int size)
- void sinft (double *y, complex< double > *rbuffer, const int size)

  : Is the function in charge to realize the sine transform the one that you can obtain with equatio F_k = {j=1}^{N-1}f_j sin(j k/N)

2.8.1 Function Documentation

2.8.1.1 void complete_arr (double * *arr_in*, double * *arr_out*, const int *size*)

2.8.1.2 int get_array_lenght (const int *size*)

2.8.1.3 sinft (double * *y*, complex< double > * *rbuffer*, const int *size*)

: Is the function in charge to realize the sine transform the one that you can obtain with equatio F_k = {j=1}^{N-1}f_j sin(j k/N)

:

Parameters

:	"size" is the size of the array.
:	"*y" is a pointer towards an array that has to be of a size power of two.
:	"*rbuffer" is a pointer towards an array where is going to return the cosine transform into other array of complex numbers.

Index

bandpass
  freqfilters.cpp, 5

complete_arr
  cosft.cpp, 3
  sinft.cpp, 10
cosft
  cosft.cpp, 4
cosft.cpp, 3
  complete_arr, 3
  cosft, 4
  get_array_lenght, 4

decode
  ogg_vorbis.cpp, 7

encoder
  ogg_vorbis.cpp, 7

fft_west
  fftwest.cpp, 4
fftwest.cpp, 4
  fft_west, 4
  ifft_west, 4
freqfilters.cpp, 5
  bandpass, 5
  highpass, 5
  lowpass, 5
  PI, 5

get_array_lenght
  cosft.cpp, 4
  sinft.cpp, 10

highpass
  freqfilters.cpp, 5

ifft_west
  fftwest.cpp, 4

lowpass
  freqfilters.cpp, 5

noise.cpp, 6
  whitenoise, 6

ogg_vorbis.cpp, 6
  decode, 7
  encoder, 7
  READ, 7
  samples_number, 8
  samples_rate, 8
  total_time, 8

PI
  freqfilters.cpp, 5
  pitch_filter.cpp, 9
PITCH
  pitch_filter.cpp, 9
pitch_filter
  pitch_filter.cpp, 9
pitch_filter.cpp, 8
  PI, 9
  PITCH, 9
  pitch_filter, 9
  UNPITCH, 9
plot.cpp, 9
  plot_freq, 9
  plot_time, 10
plot_freq
  plot.cpp, 9
plot_time
  plot.cpp, 10

READ
  ogg_vorbis.cpp, 7

samples_number
  ogg_vorbis.cpp, 8
samples_rate
  ogg_vorbis.cpp, 8
sinft
  sinft.cpp, 10
sinft.cpp, 10
  complete_arr, 10
  get_array_lenght, 10
  sinft, 10

total_time
  ogg_vorbis.cpp, 8

UNPITCH
  pitch_filter.cpp, 9

whitenoise
  noise.cpp, 6