Nyola Sound Library

1.0

Generated by Doxygen 1.8.3.1

Sun Oct 13 2013 20:39:37

Contents

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

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 (const int size)
- 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 .

Definition at line 30 of file cosft.cpp.

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.

Definition at line 50 of file cosft.cpp.

2.1.1.3 int get_array_lenght (const int *size*)

Definition at line 14 of file cosft.cpp.

2.2 fft.cpp File Reference

```
#include <stdlib.h>
#include <math.h>
#include <algorithm>
#include <complex>
```

Macros

- #define PI 3.141592654
- #define FORWARD 1
- #define BACKWARD -1;

Functions

- int bits_length (int n)
  Calculates the necessary bits to store the array indexes.
- int bitrev (int num, int bitslength)
  The function attenuates only the highest frequencies to a determinated frequency in a wave.
- int next_pow_2 (int size)
  calculates the next power of two .
- void fft (int smpls_read, double ∗buffer, complex< double > ∗transformed)
  The function calculated the fast fourier transform.

2.2.1 Macro Definition Documentation

2.2.1.1 #define BACKWARD -1;

Definition at line 8 of file fft.cpp.

2.2.1.2 #define FORWARD 1

Definition at line 7 of file fft.cpp.

2.2.1.3 #define PI 3.141592654

Definition at line 6 of file fft.cpp.

2.2.2 Function Documentation

2.2.2.1 bitrev (int *num*, int *bitslength*)

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

Parameters

bitslength,:	the number of bits.
num:is	the number you are looking for the bit reverse.

Definition at line 33 of file fft.cpp.

2.2.2.2 bits_length (int *n*)

Calculates the necessary bits to store the array indexes.

Parameters

n,:	is number tells how many bits (binary digits) are required to store that number in decimal

Definition at line 18 of file fft.cpp.

2.2.2.3 fft (int *smpls_read*, double ∗ *buffer*, complex< double > ∗ *transformed*)

The function calculated the fast fourier transform.

FFT requires the array length to be a power of two. The input array contains N complex samples, with real and imaginary part alternating, so the number of samples must be multiplied by two

Parameters

∗buffer,:	is a pointer towards the array whit the decoded song .
smpls_num,:	is the number of total samples read.
∗transformed,:	is a pointer towards the array complex where send the result of the transform.

Definition at line 70 of file fft.cpp.

2.2.2.4 next_pow_2 (int *size*)

calculates the next power of two .

Parameters

size	: is the large of the array .

Definition at line 52 of file fft.cpp.

2.3 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)

2.3.1 Function Documentation

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

Definition at line 9 of file fftwest.cpp.

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

Definition at line 28 of file fftwest.cpp.

2.4 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.

2.4.1 Macro Definition Documentation

2.4.1.1 #define PI 3.141592654

Definition at line 1 of file freqfilters.cpp.

2.4.2 Function Documentation

2.4.2.1 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.

Definition at line 44 of file freqfilters.cpp.

2.4.2.2 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.

Definition at line 26 of file freqfilters.cpp.

2.4.2.3 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.

Definition at line 10 of file freqfilters.cpp.

2.5 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.

2.5.1 Function Documentation

2.5.1.1 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.

Definition at line 12 of file noise.cpp.

2.6 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.

2.6.1 Macro Definition Documentation

2.6.1.1 #define READ 1000000

Definition at line 10 of file ogg_vorbis.cpp.

2.6.2 Function Documentation

2.6.2.1 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 name and direction of the song.
smpls_num	: is the number of total samples and is equal to size of buffer.

Returns

the number of samples read

Definition at line 73 of file ogg_vorbis.cpp.

2.6.2.2 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.

Definition at line 107 of file ogg_vorbis.cpp.

2.6.2.3 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 name and direction of the song.

Returns

The number of samples in total song.

Definition at line 25 of file ogg_vorbis.cpp.

2.6.2.4 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 name and direction of the song.

Returns

The samples for second of the song.

Definition at line 55 of file ogg_vorbis.cpp.

2.6.2.5 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 name and direction of the song.

Returns

The total time in seconds song.

Definition at line 40 of file ogg_vorbis.cpp.

2.7 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.

2.7.1 Macro Definition Documentation

2.7.1.1 #define PI 3.141592654

Definition at line 7 of file pitch_filter.cpp.

2.7.1.2 #define PITCH 1

Definition at line 8 of file pitch_filter.cpp.

2.7.1.3 #define UNPITCH 0

Definition at line 9 of file pitch_filter.cpp.

2.7.2 Function Documentation

2.7.2.1 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.

Definition at line 28 of file pitch_filter.cpp.

2.8 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)

2.8.1 Function Documentation

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

Definition at line 32 of file plot.cpp.

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

Definition at line 14 of file plot.cpp.

2.9 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)

2.9.1 Function Documentation

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

Definition at line 34 of file sinft.cpp.

2.9.1.2 int get_array_lenght (const int *size*)

Definition at line 16 of file sinft.cpp.

2.9.1.3 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.

Definition at line 53 of file sinft.cpp.

Index