DSP Project Abstracts
CE 101
February 2010

Project Code:

CE101-feb2010-01

Title:

Color Code

Proponents:

Jeremiah Aboganda
Christopher Go
Jonathan Maniago

Abstract:

Color Code is a C++ program developed using wxDev and included wxWidgets. The project aims to develop a encoding and decoding system for colored “bar” codes built on a custom character codec. The program has two major functions namely the Encoder and Decoder. The Encoder produces a colored “bar” code image from user supplied text. The produced bar code image has both vertical and horizontal reference bars, to be used in detection and color tolerance reference, and the data grid area, where the data blocks are located. The Decoder locates the color bar code from images and converts it to its text equivalent using the same codec from the Encoder.

The Decoder function asks for text input from the user. The text input is split and processed by character to produce its binary string equivalent as found in the codec library. The binary equivalent is then converted to image form using a custom layout.

The Encoder function takes a an image and process it for the reference bars of the colored “bar” code. The colored “bar” codes data grid is then processed and converted into binary equivalents using the previous codec library. The resultant binary is then converted into text.

Notes:

Project Code:

CE101-feb2010-02

Title:

Chorus Effect

Proponents:

Francis Gerald D. Bautista
Helen Dominic L. Cabrera
Claire Kathleen K. Yu

Abstract:

The Chorus Effect software was implemented using Dev-C++. The main goal of this software is to produce pitch shifted copies of an audio file with a single voice or instrument and combine these as one; thereby, producing a chorus effect. Audio files in *.wav format can be opened, processed and produced by this software. All interactions between the user and the software occur within the Command Line Interface (CLI).

Two ways have been included to accomplish the chorus effect. First, the user may opt to use the Audiere library, wherein sixteen pitch shifted copies will be produced and combined into a single audio file together with the original audio file. With this option, the pitch shift as well as the number of copies are not defined by the user but instead pre-set within the code.

Second, the user may opt to define their own settings. With this option, the number of copies to be produced as well as the pitch shift for each copy are not pre-set but instead defined by the user. The extraction of data and other properties from the original audio file is accomplished by the libsndfile library. The pitch shifting process is then done by the smbPitchShift class, which contains the smbPitchShift() and the smbFft() functions. The smbPitchShift() function does the pitch shifting and it uses the smbFft() function to do the Fast Fourier Transform (FFT). The pitch shifted copies are combined into one to create the chorus effect. The chorus effect created is played using the Audiere library. Finally, the chorus effect created can be written to a *.wav file using the libsndfile library.

Notes:

Project Code:

CE101-feb2010-03

Title:

Image Enhancement Using Tone-Correction

Proponents:

Kenneth T. Chua
John Patrick Y. Sevilla
Bianca Elinor T. Uy

Abstract:

Image Enhancement Using Tone Correction is a C++ program developed using wxDevC++ compiler and the wxWidgets library. In a two-part process, the program first loads an image and, from this image, an inverted, low-pass filtered monochrome image of the original input is produced to be used as a tone mask. Then, the derived mask is combined with the input image through a non-linear operation using an exponential function that has the mask value as part of its exponent and the pixel values as part of its base.

The group created a user-friendly GUI that automatically copies the loaded input image to another window so that the enhanced image can be compared to the input image before saving. Also, a slider was added to allow users to make further adjustments to the enhanced image. The final image can then be saved as JPEG, BMP or PNG.

Notes:

Project Code:

CE101-feb2010-04

Title:

Echo Simulation

Proponents:

Maria Divina Dalafu
Rachelle Mae Fuentes
Gabrielle Jimenez

Abstract:

A program that simulates echo on WAV audio files is created using C++ programming with the help of wxDev-C++ software.

The implementation is done through the application of LTI Filter algorithm. The program has a Graphical User Interface made through the tools generated by wxDev-C++ software. A WAV audio file is opened through a wxOpenFileDialog button. Samples are extracted from the file using an open source library, libsndfile library. These samples are then gathered and put into a dynamic array to be used in creating the echo effect. While samples are extracted, the libsndfile library also gets information about the audio file, specifically the number of channels, format, sections and if it is "seekable".

The echo delay can be varied in milliseconds. The echo volume can also be varied in terms of percentage of the volume of the original audio. The GUI also has a wxButton with a label "Generate Echo" and wxSliders which determines the value of delay for the echo to be generated and the desired echo volume. When a file is already opened and samples are saved in an array, the user is already allowed to choose the value of the delay in milliseconds by moving the slider to the right to increase, or left to decrease. Likewise, the echo volume can be varied by a separate slider. After the user has chosen the delay and volume values, the user can now click the "Generate Echo" button which has an event handler that implements the LTI Filter algorithm. A Windows Media Player is integrated with the GUI of the program which activates after the echo has been generated. The user can then play the audio file with the echo. The user also has an option to play the original audio file by setting the delay and volume values to zero.

Furthermore, the audio file with echo is saved as a WAV format by the libsndfile library. The user has the option to save the generated audio file with echo by clicking on the save file icon. The user can then specify the filename and file path to where the file is to be saved.

Notes:

Project Code:

CE101-feb2010-05

Title:

Multiple Audio Player

Proponents:

Patrick B. Jawili
Jon Dexter S. Laurin
Michelle Jean R. Sia

Abstract:

Multiple Audio Player is a C++ program developed using DevC++ with the libsndfile library, which aims to play multiple wav files simultaneously by adding them together. It makes use of command line arguments to play the input wav files. Through this Multiple Audio Player, the user is able to play any number of wav files at the same time, without regard to the order in which the wav files are inputted on the screen and their individual lengths.

The program first asks for the names of the wav files to be played. The wav files are decoded into their audio sample equivalent and then stored into a dynamically allocated array. These arrays are added together to form the output values. These array is copied to a vector that increases or decreases its size so that even wav files of different lengths can be added together. The program keeps track of how many wav files are to be added and divides each audio samples by this value to avoid clipping after they are added.

The program plays the audio data by using the Audiere library openBuffer() function. The user then is asked if he would want to save the audio to file.

Notes:

Project Code:

CE101-feb2010-06

Title:

Speech Analysis and Synthesis Using Linear Predictive Coding

Proponents:

Rasia Elaine S. Marcelo
Marlon Roy S. Pelayo
Jeremy Ryan A. Santos

Abstract:

The project is a windows-based application implemented using C++ that attempts to analyze speech signals by extracting the basic parameters of speech through linear predictive analysis and then synthesizing to create the transformed voice data array. In the process, Hamming Window is implemented for spectral analysis and an LPC inverse filter involving Autocorrelation and Levinson-Durbin algorithms is designed. After a signal passes through the inverse filter, the signal left (residual speech signal) is stored and then used later on together with the predictor values (formants) of another voice for synthesizing. The library libsndfile and wxDev widgets are used to provide reading and writing sampled sound functionalities and creation of a GUI, respectively. The application is intended for analyzing and synthesizing 8 kHz wave files.

Notes: