|  |
| --- |
| CM2208 Scientific Computing |
| MATLAB Audio Player/Organiser |
| Cardiff University |

|  |
| --- |
| C1535633 Zak-Sum Yeung |

Table of Contents

[Introduction 2](#_Toc481458850)

[Import and Play Audio 3](#_Toc481458851)

[Creating a Playlist 4](#_Toc481458852)

[Display the Amplitude and Time of the Audio file 7](#_Toc481458853)

[Wah-Wah Effect 9](#_Toc481458854)

[Source Code Implementation 10](#_Toc481458855)

[Start.m 10](#_Toc481458856)

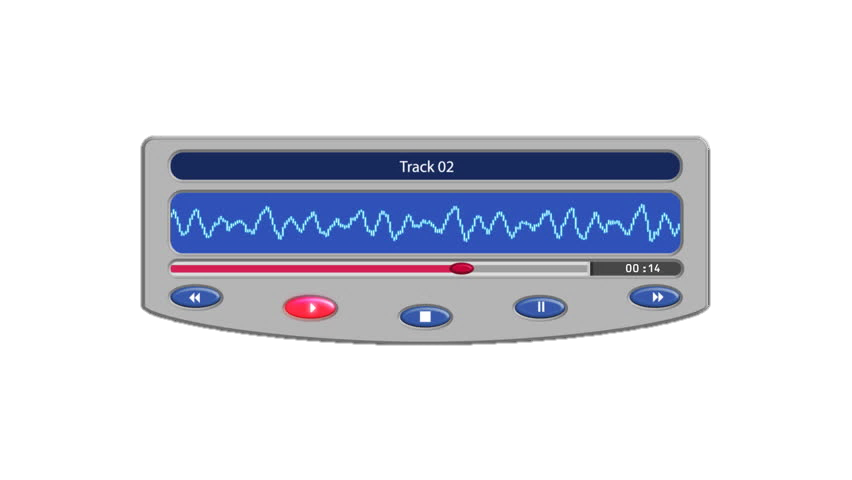
[Music\_Player.m 10](#_Toc481458857)

[wahwah\_effect.m 20](#_Toc481458858)

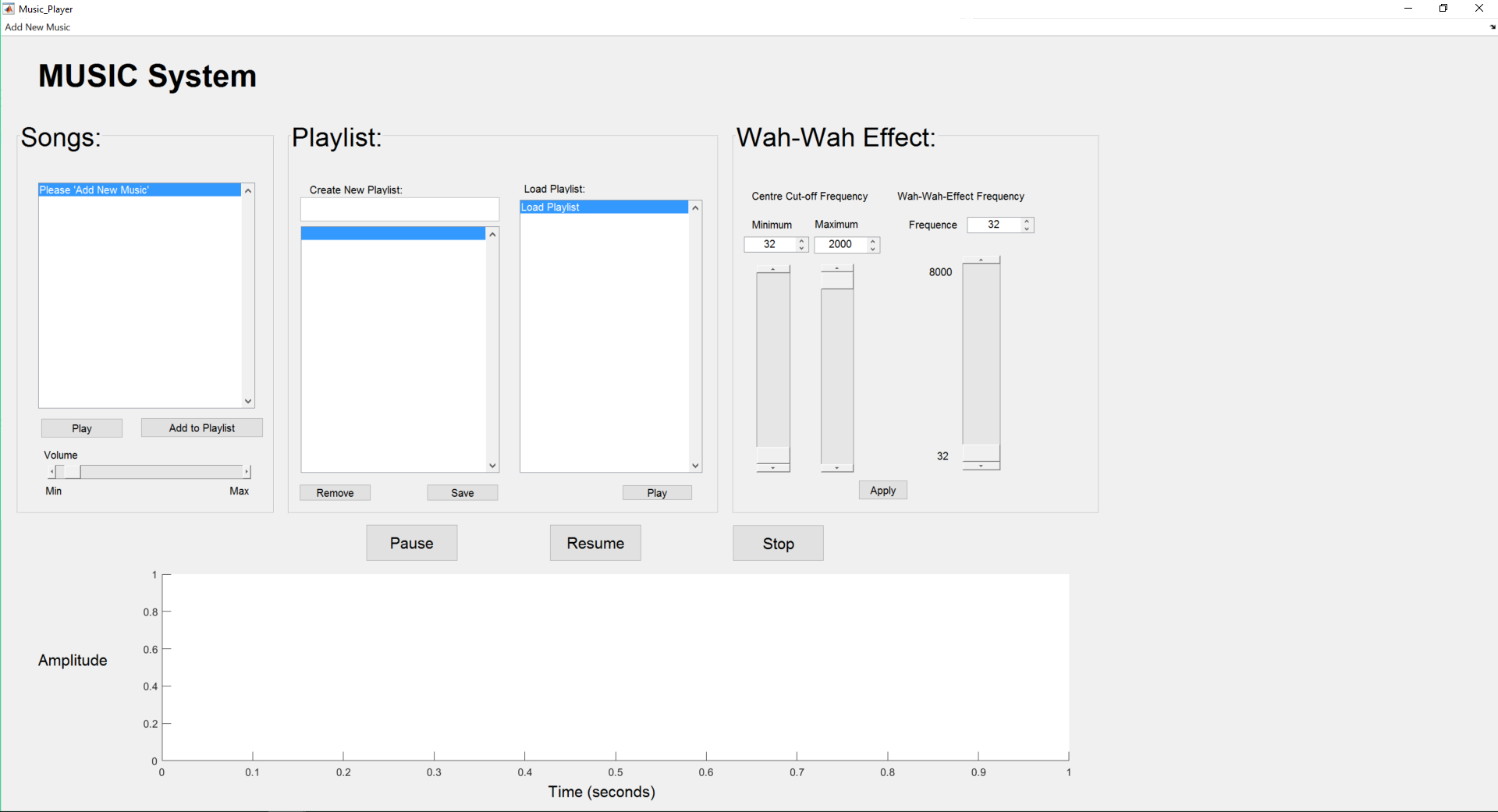
[plotting.m 21](#_Toc481458859)

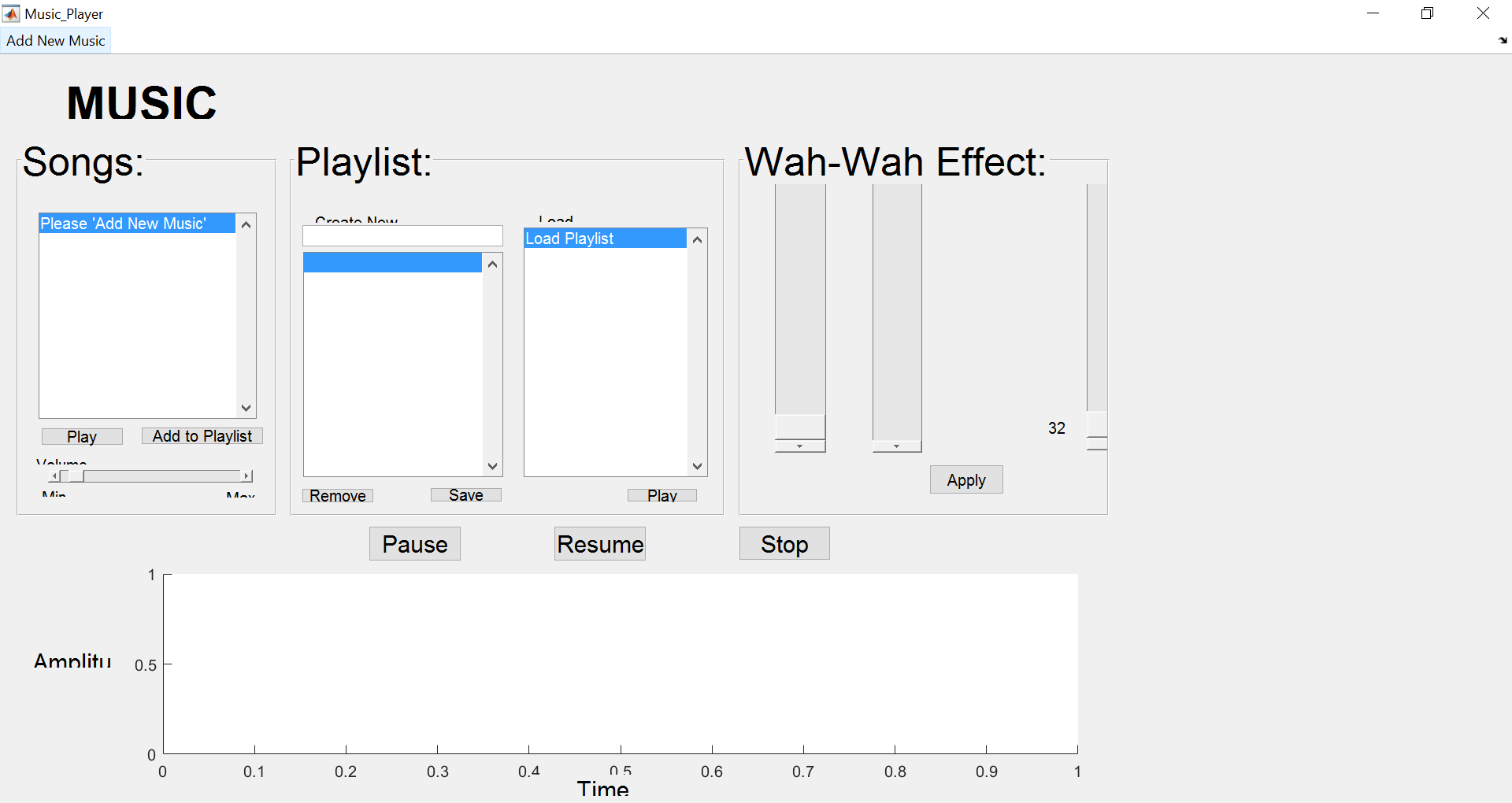
# Introduction

The aim of this report is to be able to build a simple audio player/organiser using MATLAB. With the functionalities of being able to list the audio files, provide some simple means of audio playback such as a ‘Play’, ‘Pause’, ‘Stop’ buttons. This application should also be able to load and play audio in a few formats such as ‘.wav’, ‘.aiff’, ‘.mp3’, ‘.mp4’, ‘.acc’ and ‘.ogg’. But also, aim to allow users to alter the sound frequency of the audio currently being played. An example image is shown below of what the application is meant to look like:



For the users to play their selected audio files, I have created a simple Graphical User Interface (GUI) which allows the users to upload the specific file formats as listed above by clicking the tab button ‘Add New Music’.

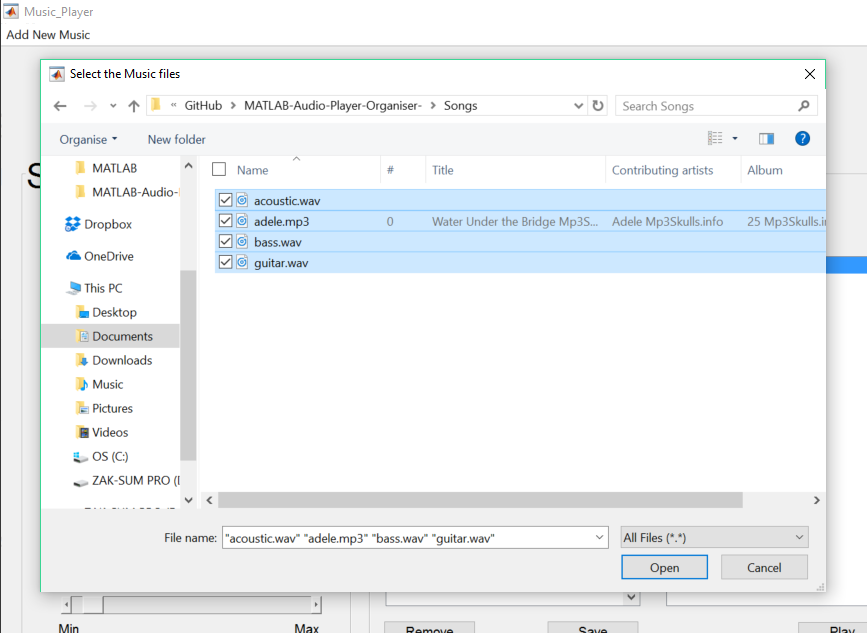


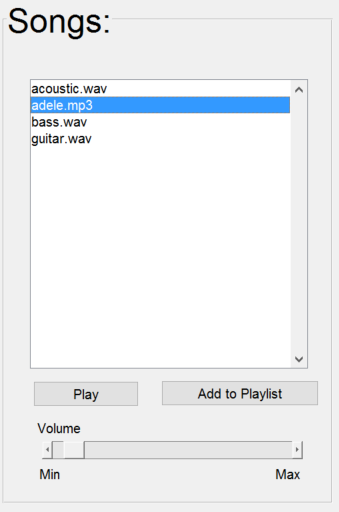


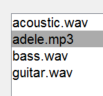
# Import and Play Audio

My GUI design is a simple design where it allows users to upload a multiple selection of the supported audio files, which gets uploaded to a simple List box, showing the users the file names. Once the names of the audio files are shown, the user would then be able to select an audio and press the button ‘Play’ which plays the song. An example is shown below:

Selecting Audio

From the image below, the user can select multiple file types.

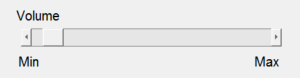
Once the user has selected their songs and click ‘Open’, the song file names will be shown in the List Box, and all left to do is to click the button ‘Play’:





Volume Adjustment

Having pressed the ‘Play’ button, the user would be able to adjust the Volume of the audio. In the GUI, I have added a slider bar from the ‘Minimum’ value to the ‘Maximum’ value.



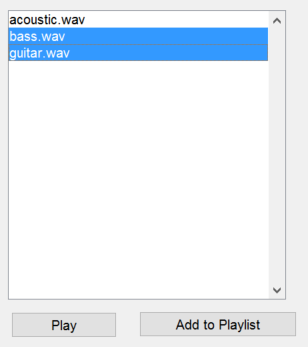
This is done by first setting the values of the slider bar from a minimum value of 0 all the way to the maximum value of 20. Where inside the Callback function, whenever the user slides the slide bar, the system would get the value of the slider object and if the slider equals to the value of the slider, then multiply the values of the audio which are stored in a matrix.

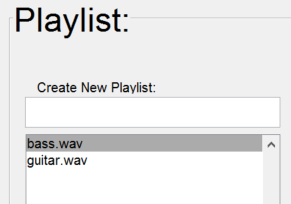
# Creating a Playlist

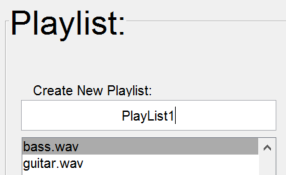
A playlist is a function that exists in a wide variety of music applications. The benefits of having a Playlist allows users to add specific songs and save it to a folder, where when the user selects the folders name, and click ‘Play’, it would Play all the music in that selected folder.

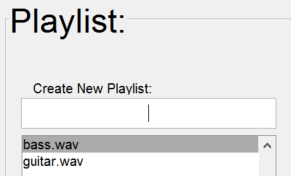
For my system, I decided to store all the audio files in a cell array with the name of the playlist assigned on the first column (cell array {1.1}) and all the audio files on the second column of the same row (cell array {1.2}). Therefore, whenever the user selects the name of the Playlist, the system would get that objects selection, and merge all the songs together into one huge song and play when the user selects the ‘Play’ button.

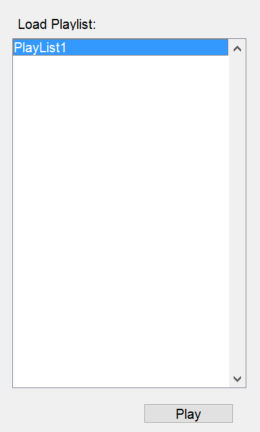
Selecting the song to ‘Add to Playlist’

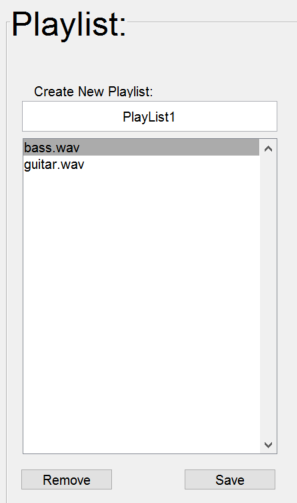




Naming and Saving Playlist

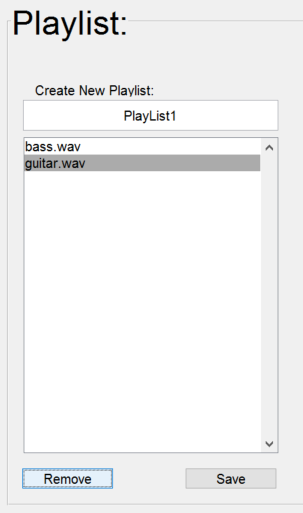


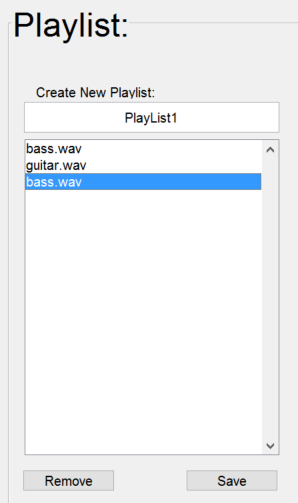
After the user, has added the songs into an empty playlist, the user will then type a name of the playlist into the empty textbox to save the playlist in the cell array.



When the user has created enough playlists, the user would need to select on one of the many playlists created and press the button ‘Play’ to play the entire list of audio files of that playlist.

Removing Songs when added to a new playlist

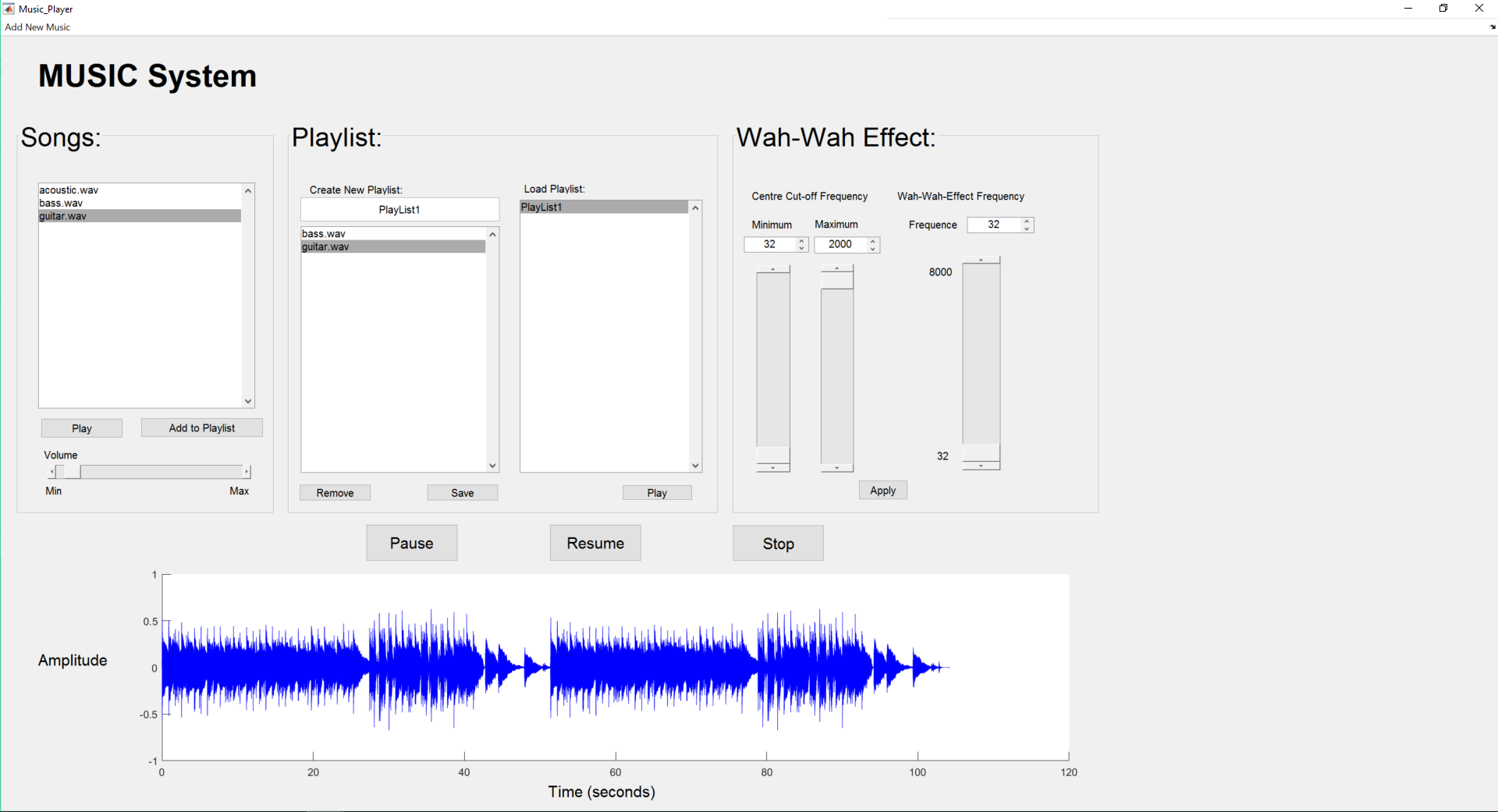
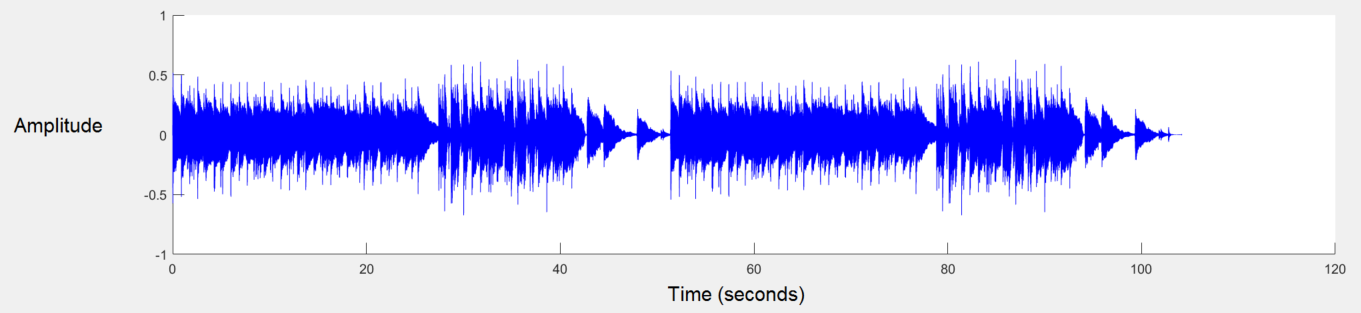
I have decided to include this feature where it gives the users an opportunity to remove the songs before saving the playlist. This is a crucial feature to include because it provides user control, rather than having to restart the program, which is a very inefficient way. Below is an example of the ‘Remove’ button:

Since ‘bass.wav’ was a duplicate song in the playlist, we would want to remove it. Therefore, after clicking the ‘Remove’ button, it removed the value of the index.

# Display the Amplitude and Time of the Audio file

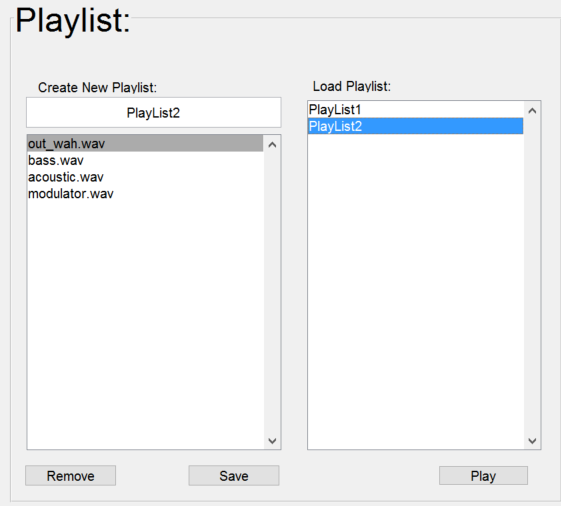
I have created an axis where it would be able to display the audios time in seconds and amplitude. This would give the user an in-depth view of the audio files being played. Below are examples of what happens to the axis once the ‘Play’ buttons are clicked.

‘Play’ Button in Songs category:

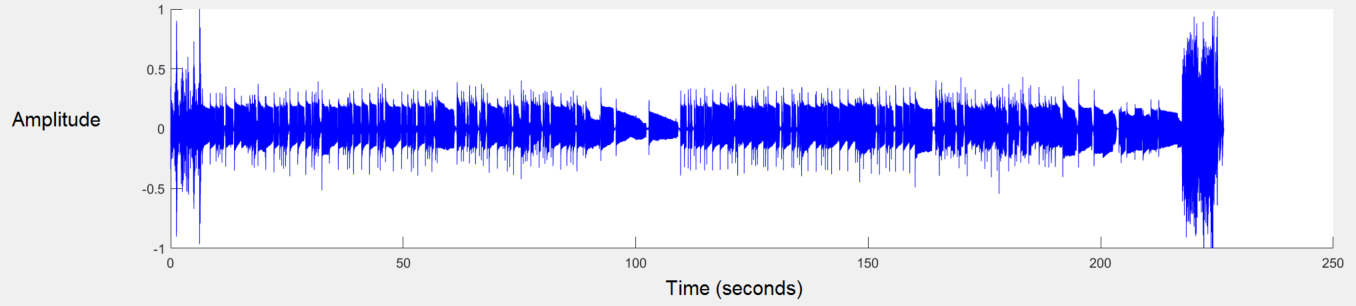
Once the user has imported their songs to the list box, the user selects a song and click on the button ‘Play’:

‘Play’ Button in Playlist category:

With the user, having already created a playlist. When the user clicks on the selected Playlist and clicks the button ‘Play’, the axes would show how long in seconds the whole playlist is, including all the different songs amplitude, therefore it will be visible to the user to find the difference of when it is a different song due to having different amplitudes and different sound frequency.



In this playlist, I have added 4 different audio files and created a playlist called, ‘Playlist2’.



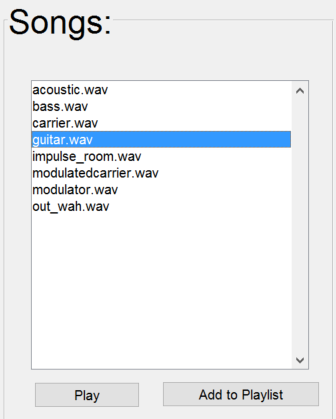
As you can see form above, all the audios are merged into one huge song, and when plotted on the axis, the user can tell the difference in the four songs as they have a different amplitude compared to the other audio files.

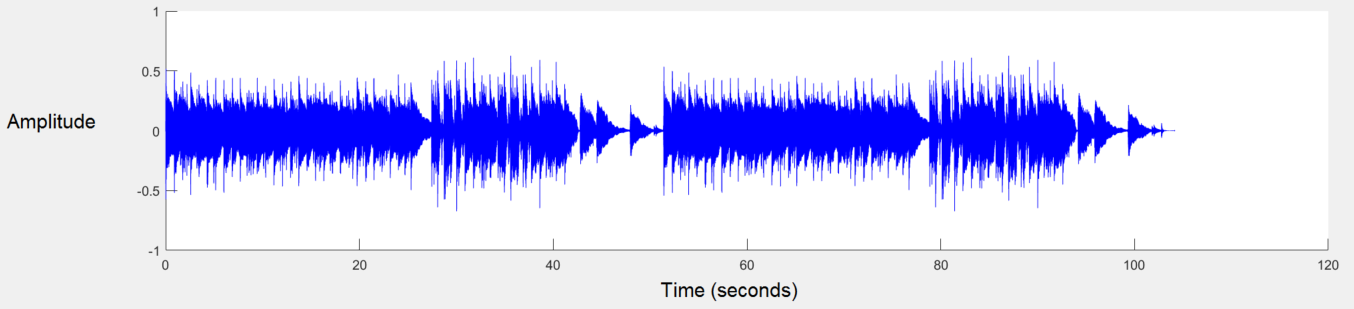
# Wah-Wah Effect

I have adapted the use of a ‘Wah-Wah effect’ to the system allowing the users to give their audio files a ‘wah-wah’ sound depending on the centre cut off frequency of the audio as well as the frequency of the ‘wah-wah effect’.

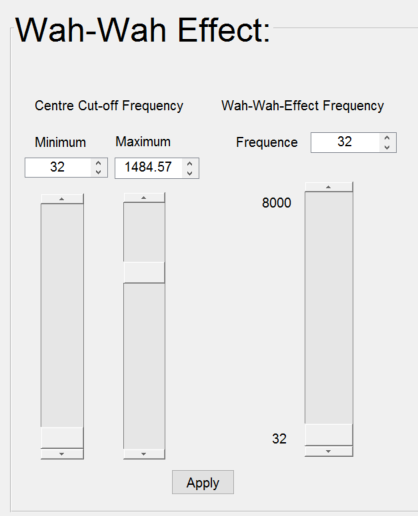
Below is an example of how it can be applied to an audio file after uploading it by the user:

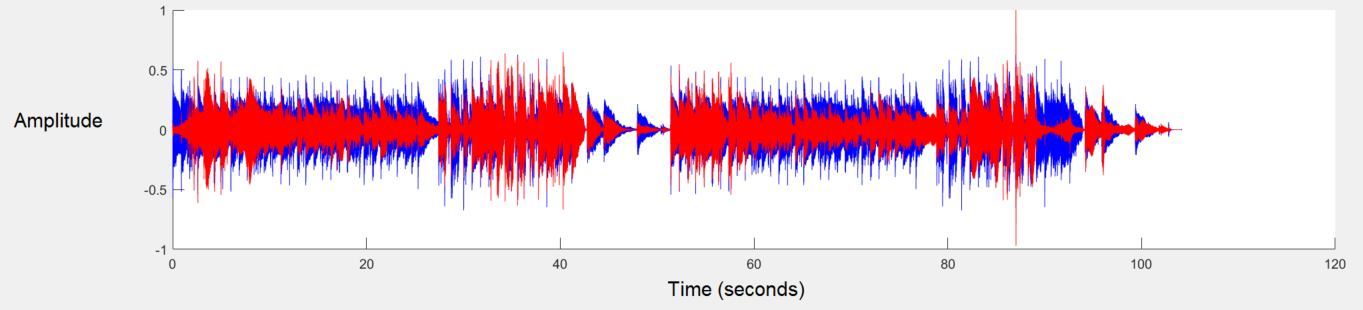
1. Play the song from any list:



As you can see from the axes below, the user has uploaded their songs and decided to play the audio file, ‘guitar.wav’. The audio is then displayed on the axes.

1. Applying the ‘Wah-Wah effect’:



I have decided to set the maximum cut-off time to ‘1484.57’, and leave the minimum cut off time and the wah-wah effect frequency to a default value of ‘32’. Once the values are set, I pressed the ‘Apply’ button where it would show on the axes that the ‘blue’ coloured amplitude is the original audio and the ‘red’ amplitude is the ‘wah-wah effect’ being applied. Allowing the users to see a difference.

The implementation of Wah-Wah Effect references “MATLAB Wah-wah Implementation” by Professor Marshall.

Idea From:

<http://users.cs.cf.ac.uk/Dave.Marshall/CM2208/LECTURES/CM2208_DSP_03_Filters.pdf>

Source Code From:

<http://www.cs.cf.ac.uk/Dave/CM0268/PDF/10_CM0268_Audio_FX.pdf>

# Source Code Implementation

## Start.m

close all;

clear;

clc;

Music\_Player

## Music\_Player.m

function varargout = Music\_Player(varargin)

% MUSIC\_PLAYER MATLAB code for Music\_Player.fig

% MUSIC\_PLAYER, by itself, creates a new MUSIC\_PLAYER or raises the existing

% singleton\*.

%

% H = MUSIC\_PLAYER returns the handle to a new MUSIC\_PLAYER or the handle to

% the existing singleton\*.

%

% MUSIC\_PLAYER('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in MUSIC\_PLAYER.M with the given input arguments.

%

% MUSIC\_PLAYER('Property','Value',...) creates a new MUSIC\_PLAYER or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before Music\_Player\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to Music\_Player\_OpeningFcn via varargin.

%

% \*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one

% instance to run (singleton)".

%

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Music\_Player

% <<<<<<< HEAD

% % Last Modified by GUIDE v2.5 01-May-2017 00:16:47

% =======

% % Last Modified by GUIDE v2.5 30-Apr-2017 18:27:12

% >>>>>>> origin/master

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @Music\_Player\_OpeningFcn, ...

'gui\_OutputFcn', @Music\_Player\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before Music\_Player is made visible.

function Music\_Player\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to Music\_Player (see VARARGIN)

% Choose default command line output for Music\_Player

handles.output = hObject;

% Update handles structure

cellMaster = {};

Songs = {};

PlayList = {};

cellMaster{1} = Songs;

cellMaster{2} = PlayList;

handles.data = cellMaster;

guidata(hObject, handles);

% UIWAIT makes Music\_Player wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = Music\_Player\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

% --------------------------------------------------------------------

function Add\_Callback(hObject, eventdata, handles)

% hObject handle to Add (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

FilterSpec = {'\*.wav'; '\*.aiff'; '\*.mp3'; '\*.mp4'; '\*.acc'; '\*.ogg'};

[FileName,PathName,FilterIndex] = uigetfile(FilterSpec, 'Select the Music files', 'MultiSelect', 'on')

%call list object

lst\_Music = findobj('Tag', 'lst\_Music');

set(lst\_Music,'String',FileName)%display the filename as a string in the Listbox

Songs = handles.data{1};

if (iscell(FileName ))

for i = 1:max(size(FileName))

Songs{i,1} = FileName(i);

Songs{i,2} = PathName;

end

else

Songs{1,1} = {FileName};

Songs{1,2} = PathName;

end

Songs

handles.data{1} = Songs;

guidata(hObject,handles);

% --- Executes on selection change in lst\_Music.

function lst\_Music\_Callback(hObject, eventdata, handles)

% hObject handle to lst\_Music (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns lst\_Music contents as cell array

% contents{get(hObject,'Value')} returns selected item from lst\_Music

% --- Executes during object creation, after setting all properties.

function lst\_Music\_CreateFcn(hObject, eventdata, handles)

% hObject handle to listbox2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on selection change in lst\_NewPlaylist.

function lst\_NewPlaylist\_Callback(hObject, eventdata, handles)

% hObject handle to lst\_NewPlaylist (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns lst\_NewPlaylist contents as cell array

% contents{get(hObject,'Value')} returns selected item from lst\_NewPlaylist

% --- Executes during object creation, after setting all properties.

function lst\_NewPlaylist\_CreateFcn(hObject, eventdata, handles)

% hObject handle to lst\_NewPlaylist (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on button press in btn\_Add.

function btn\_Add\_Callback(hObject, eventdata, handles)

% hObject handle to btn\_Add (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

list\_entry = cellstr(get(handles.lst\_Music,'String'));

index\_selected = get(handles.lst\_Music,'Value'); %changed line

choice\_lst\_Music = list\_entry(index\_selected);

update\_lst\_NewPlaylist = cellstr(get(handles.lst\_NewPlaylist, 'String'));

newmenu = [update\_lst\_NewPlaylist;choice\_lst\_Music];

set(handles.lst\_NewPlaylist,'String', newmenu);

% --- Executes on button press in btn\_Save.

function btn\_Save\_Callback(hObject, eventdata, handles)

% hObject handle to btn\_Save (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

PlayList = handles.data{2};

edtxtNewPlayList = findobj('Tag', 'txt\_Name');%'findobj' to get the whole object

listName = get(edtxtNewPlayList,'String');%get the data from the object

lbNewPlaylist = findobj('Tag', 'lst\_NewPlaylist');

listSong = get(lbNewPlaylist, 'String');

ind = size(PlayList,1) + 1;

PlayList{ind,1} = listName;

PlayList{ind,2} = listSong;

handles.data{2} = PlayList; %update handles.data storage

guidata(hObject,handles); % put the handles back to master

PlayList

%Copy Names of playlist into Listbox

names = PlayList(:,1); %Gets all the first colum data on the cell array Playlist

list\_listNames = findobj('Tag', 'Playlist');%'findobj' to get the whole object

set(list\_listNames,'String',names);%Display Playlist names in the Listbox

function txt\_Name\_Callback(hObject, eventdata, handles)

% hObject handle to txt\_Name (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of txt\_Name as text

% str2double(get(hObject,'String')) returns contents of txt\_Name as a double

% --- Executes during object creation, after setting all properties.

function txt\_Name\_CreateFcn(hObject, eventdata, handles)

% hObject handle to txt\_Name (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on selection change in Playlist.

function Playlist\_Callback(hObject, eventdata, handles)

% hObject handle to Playlist (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns Playlist contents as cell array

% contents{get(hObject,'Value')} returns selected item from Playlist

% --- Executes during object creation, after setting all properties.

function Playlist\_CreateFcn(hObject, eventdata, handles)

% hObject handle to Playlist (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on button press in btn\_Remove.

function btn\_Remove\_Callback(hObject, eventdata, handles)

% hObject handle to btn\_Remove (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

rmvList = handles.lst\_NewPlaylist;

rmvindexed = get(rmvList,'value');

newPlace = rmvindexed(1)-1;

if (newPlace <=0) newPlace = 1; end

rmvnames = get(rmvList,'String');

if ~isempty(rmvnames)

rmvnames(rmvindexed) = [];

set(rmvList,'String',rmvnames,'value', newPlace);

end

% --- Executes on button press in btnPlay.

function btnPlay\_Callback(hObject, eventdata, handles)

% hObject handle to btnPlay (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% allstrings = cellstr( get(handles.lst\_Music, 'String') );

% curvalue = get(handles.lst\_Music, 'Value');

% thisstring = allstrings{curvalue};

% [q, Fs] = audioread(thisstring);

SongList = findobj('Tag', 'lst\_Music');%'findobj' to get the whole object

SongIndex = get(SongList,'Value');%get the data from the object

Songs = handles.data{1};

name = Songs{SongIndex,1};

path = Songs{SongIndex,2};

pathname = char(strcat(path,name));

handles.pathname = pathname;

guidata(hObject,handles)

pathname

[y,Fs] = audioread(pathname);

% sound(y,Fs);

global y\_matrix;

y\_matrix = y;

global Fs\_matrix;

Fs\_matrix = Fs;

global audio;

audio = audioplayer (y\_matrix, Fs\_matrix);

cla reset

plotting(y, Fs, 'b', handles);

handles.pathname = 0;

guidata(hObject,handles)

play(audio);

% --- Executes on button press in btnStop.

function btnStop\_Callback(hObject, eventdata, handles)

% hObject handle to btnStop (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global audio;

stop(audio)

cla reset

handles.pathname = 0;

guidata(hObject,handles)

% --- Executes on button press in btnPause.

function btnPause\_Callback(hObject, eventdata, handles)

% hObject handle to btnPause (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global audio;

if handles.pathname == 0;

pause(audio);

else

handles.pathname == 1;

end

handles.pathname = 1;

guidata(hObject,handles)

% --- Executes on button press in btnResume.

function btnResume\_Callback(hObject, eventdata, handles)

% hObject handle to btnResume (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global audio;

if handles.pathname == 1

resume(audio);

end

handles.pathname = 0;

guidata(hObject,handles)

% --- Executes on slider movement.

function sliderVolume\_Callback(hObject, eventdata, handles)

% hObject handle to sliderVolume (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider

% get(hObject,'Min') and get(hObject,'Max') to determine range of slider

global y\_matrix;

global Fs\_matrix;

global audio;

slider = get(hObject,'value');

if slider == 10

x = y\_matrix; % Same volume

audio = audioplayer(x, Fs\_matrix);

play(audio,[(get(audio, 'SampleRate')\*1)]);

else

x = y\_matrix\*slider; % Max volume

audio = audioplayer(x, Fs\_matrix);

play(audio,[(get(audio,'SampleRate')\*1)]);

end

guidata(hObject,handles)

% --- Executes during object creation, after setting all properties.

function sliderVolume\_CreateFcn(hObject, eventdata, handles)

% hObject handle to sliderVolume (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor',[.9 .9 .9]);

end

set(hObject, 'min', 0);

set(hObject, 'max', 20);

set(hObject, 'value', 1);

function edtWahFs\_Callback(hObject, eventdata, handles)

% hObject handle to edtWahFs (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edtWahFs as text

% str2double(get(hObject,'String')) returns contents of edtWahFs as a double

set(handles.sliderWah\_freq, 'Value', str2double(get(hObject,'String')));

% --- Executes during object creation, after setting all properties.

function edtWahFs\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edtWahFs (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edtWahMax\_Callback(hObject, eventdata, handles)

% hObject handle to edtWahMax (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edtWahMax as text

% str2double(get(hObject,'String')) returns contents of edtWahMax as a double

set(handles.sliderWah\_max, 'Value', str2double(get(hObject,'String')));

% --- Executes during object creation, after setting all properties.

function edtWahMax\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edtWahMax (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edtWahMin\_Callback(hObject, eventdata, handles)

% hObject handle to edtWahMin (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edtWahMin as text

% str2double(get(hObject,'String')) returns contents of edtWahMin as a double

set(handles.sliderWah\_min, 'Value', str2double(get(hObject,'String')));

% --- Executes during object creation, after setting all properties.

function edtWahMin\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edtWahMin (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on button press in btnWaheffect.

function btnWaheffect\_Callback(hObject, eventdata, handles)

% hObject handle to btnWaheffect (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% % handles structureSXZ with handles and user data (see GUIDATA)

global y\_matrix;

global Fs\_matrix;

global audio;

global new\_y\_matrix;

min = get(handles.sliderWah\_min, 'Value');

max = get(handles.sliderWah\_max, 'Value');

Fw = get(handles.sliderWah\_freq, 'Value');

new\_y\_matrix = wahwah\_effect(y\_matrix, Fs\_matrix, 0.05, min, max, Fw);

% [yb, input\_fs] = new\_y\_matrix

% y\_matrix = yb;

% Fs\_matrix = input\_fs;

stop(audio);

audio = audioplayer (new\_y\_matrix, Fs\_matrix);

cla reset

play(audio);

plotting(y\_matrix, Fs\_matrix, 'b', handles);

plotting(new\_y\_matrix, Fs\_matrix, 'r', handles);

% --- Executes on slider movement.

function sliderWah\_freq\_Callback(hObject, eventdata, handles)

% hObject handle to sliderWah\_freq (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider

% get(hObject,'Min') and get(hObject,'Max') to determine range of slider

set(handles.edtWahFs, 'String', get(hObject,'value'));

% --- Executes during object creation, after setting all properties.

function sliderWah\_freq\_CreateFcn(hObject, eventdata, handles)

% hObject handle to sliderWah\_freq (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor',[.9 .9 .9]);

end

% --- Executes on slider movement.

function sliderWah\_max\_Callback(hObject, eventdata, handles)

% hObject handle to sliderWah\_max (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider

% get(hObject,'Min') and get(hObject,'Max') to determine range of slider

set(handles.edtWahMax, 'String', get(hObject,'value'));

% --- Executes during object creation, after setting all properties.

function sliderWah\_max\_CreateFcn(hObject, eventdata, handles)

% hObject handle to sliderWah\_max (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor',[.9 .9 .9]);

end

% --- Executes on slider movement.

function sliderWah\_min\_Callback(hObject, eventdata, handles)

% hObject handle to sliderWah\_min (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider

% get(hObject,'Min') and get(hObject,'Max') to determine range of slider

set(handles.edtWahMin, 'String', get(hObject,'value'));

% --- Executes during object creation, after setting all properties.

function sliderWah\_min\_CreateFcn(hObject, eventdata, handles)

% hObject handle to sliderWah\_min (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor',[.9 .9 .9]);

end

% --- Executes on button press in btnPlayList.

function btnPlayList\_Callback(hObject, eventdata, handles)

% hObject handle to btnPlayList (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

SongsPath = handles.data{1}; %song name, path

PlayList = handles.data{2}; %list names

list\_listNames = findobj('Tag', 'Playlist');%'findobj' to get the whole object

selectedListIndex = get(list\_listNames,'Value');

songs = PlayList{selectedListIndex,2};

newY = [];

%merge all the songs in one song

for i = 1:size(songs,1)

for j = 1:size(SongsPath(:,1), 1)

songs(i)

SongsPath{j, 1}

strcmp(songs(i),SongsPath{j, 1})

if strcmp(songs(i),SongsPath{j, 1}) == 1

name = SongsPath{j,1};

path = SongsPath{j,2};

pathname = char(strcat(path,name));

[y, fs] = audioread(pathname);

newY = [newY;y];

end

end

end

global audio;

global y\_matrix;

global Fs\_matrix;

y\_matrix = newY;

Fs\_matrix = fs;

stop(audio);

audio = audioplayer (y\_matrix, Fs\_matrix);

cla reset

play(audio);

plotting(y\_matrix, Fs\_matrix, 'b', handles);

## wahwah\_effect.m

function [ yb, input\_fs ] = wahwah\_effect( input\_x, input\_fs, damp, minf, maxf, Fw)

%UNTITLED2 Summary of this function goes here

% Detailed explanation goes here

%%%%%%% EFFECT COEFFICIENTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% damping factor

% lower the damping factor the smaller the pass band

%damp = 0.05;

% min and max centre cutoff frequency of variable bandpass filter

%minf=500;

%maxf=3000;

% wah frequency, how many Hz per second are cycled through

%Fw = 2000;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% change in centre frequency per sample (Hz)

%delta=0.1;

delta = Fw/input\_fs;

%0.1 => at 44100 samples per second should mean 4.41kHz Fc shift per sec

% create triangle wave of centre frequency values

Fc=minf:delta:maxf;

while(length(Fc) < length(input\_x) )

Fc= [ Fc (maxf:-delta:minf) ];

Fc= [ Fc (minf:delta:maxf) ];

end

% trim tri wave to size of input

Fc = Fc(1:length(input\_x));

% difference equation coefficients

F1 = 2\*sin((pi\*Fc(1))/input\_fs); % must be recalculated each time Fc changes

Q1 = 2\*damp; % this dictates size of the pass bands

yh=zeros(size(input\_x)); % create emptly out vectors

yb=zeros(size(input\_x));

yl=zeros(size(input\_x));

% first sample, to avoid referencing of negative signals

yh(1) = input\_x(1);

yb(1) = F1\*yh(1);

yl(1) = F1\*yb(1);

% apply difference equation to the sample

for n=2:length(input\_x),

yh(n) = input\_x(n) - yl(n-1) - Q1\*yb(n-1);

yb(n) = F1\*yh(n) + yb(n-1);

yl(n) = F1\*yb(n) + yl(n-1);

F1 = 2\*sin((pi\*Fc(n))/input\_fs);

end

%normaliseaudio

maxyb = max(abs(yb));

yb = yb/maxyb;

end

## plotting.m

function plotting( y, Fs, color, handles)

axes(handles.axes1);

hold on;

N = length(y);

t = linspace(0, N/Fs, N);

plot(t, y, color)

hold off;

end