CM2208 Scientific Computing

MATLAB Audio Player/Organiser

Cardiff University

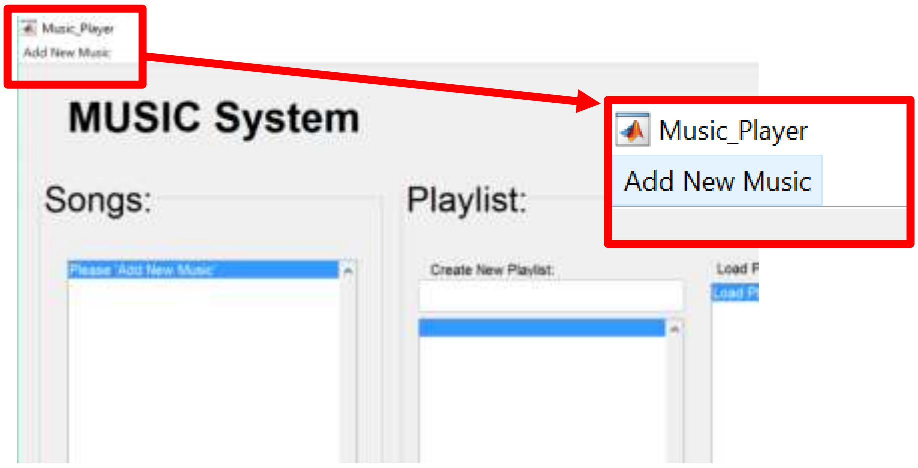
C1535633 Zak-Sum Yeung

Table of Contents

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’.

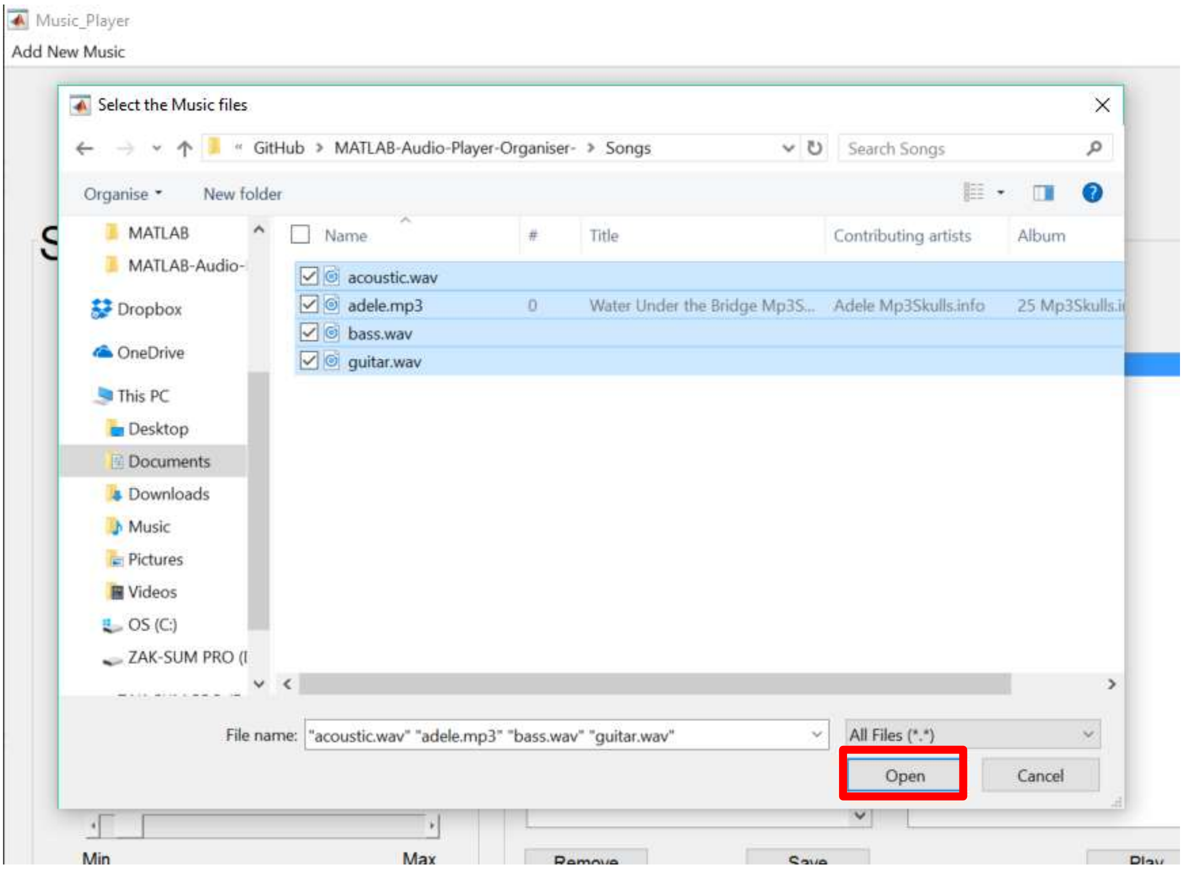


Import and Play Audio

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.

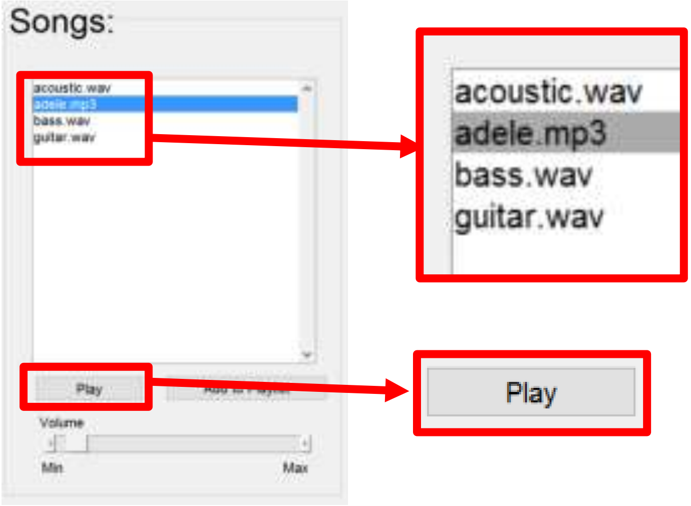


☐ Name	#	Title	Contributing artists	Album
☑ acoustic.wav				
☑ adele.mp3	0	Water Under the Bridge Mp3S...	Adele Mp3Skulls.info	25 Mp3Skulls.i
☑ bass.wav				
☑ guitar.wav				

File name: "acoustic.wav" "adele.mp3" "bass.wav" "guitar.wav"

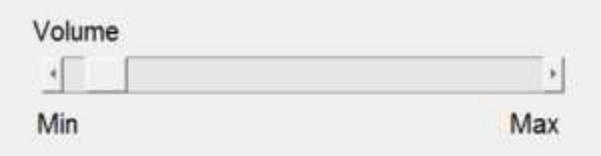
All Files (*.*)

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.



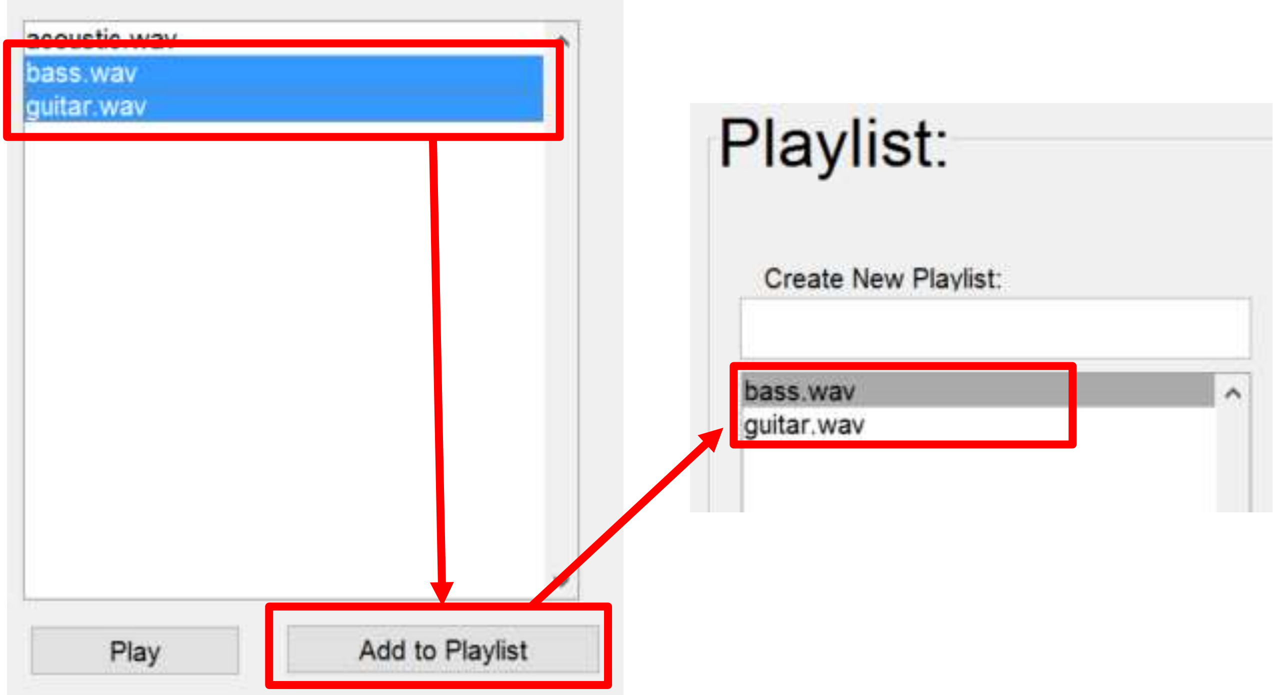
This is done by first setting the values of the slider bar from a minimum value of 0 all the way to the maximum value of 20. Where inside the Callback function, whenever the user slides the slide bar, the system would get the value of the slider object and if the slider equals to the value of the slider,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

Creating a Playlist

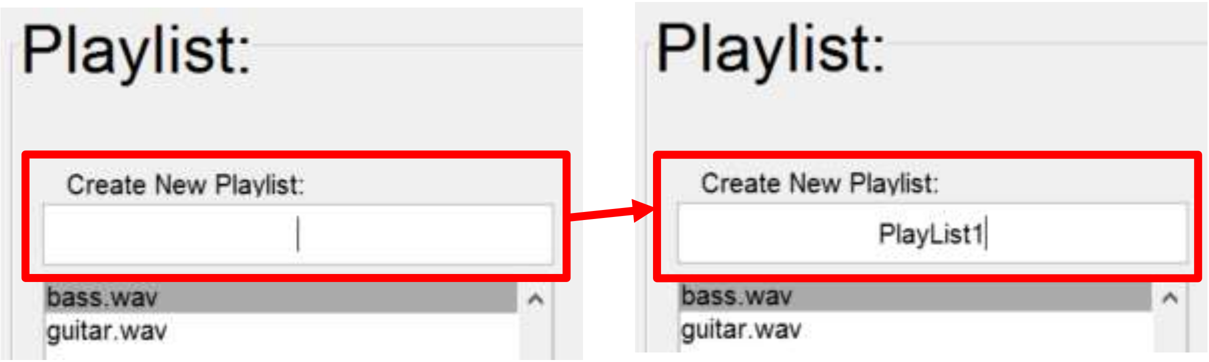
A playlist is a function that exists in a wide variety of music applications. The benefits of having a Playlist allows users to add specific songs and save it to a folder, where when the user selects the folders name, and click ‘Play’, it would Play all the music in that selected folder.

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.

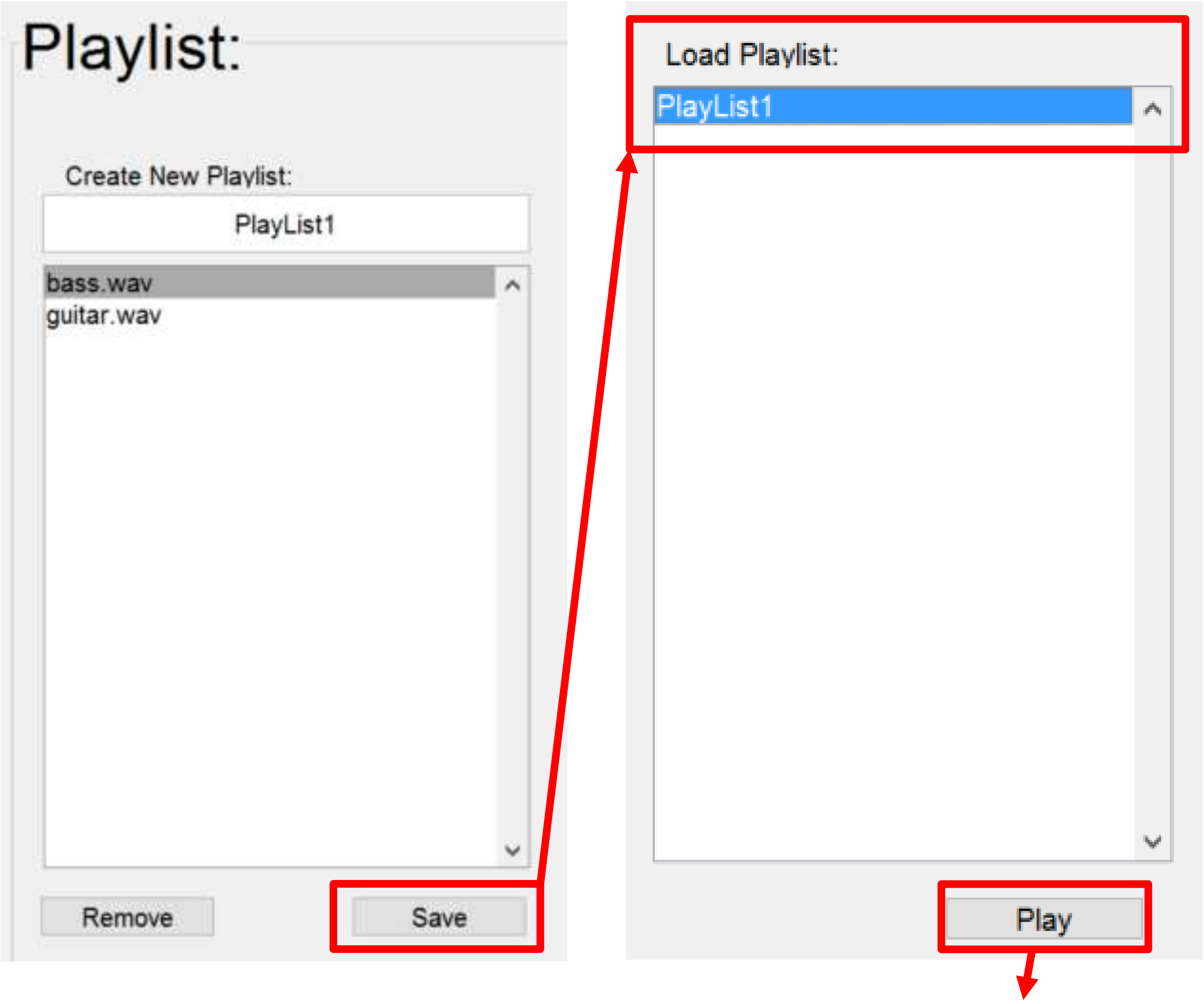
Selecting the song to ‘Add to Playlist’



Naming and Saving Playlist



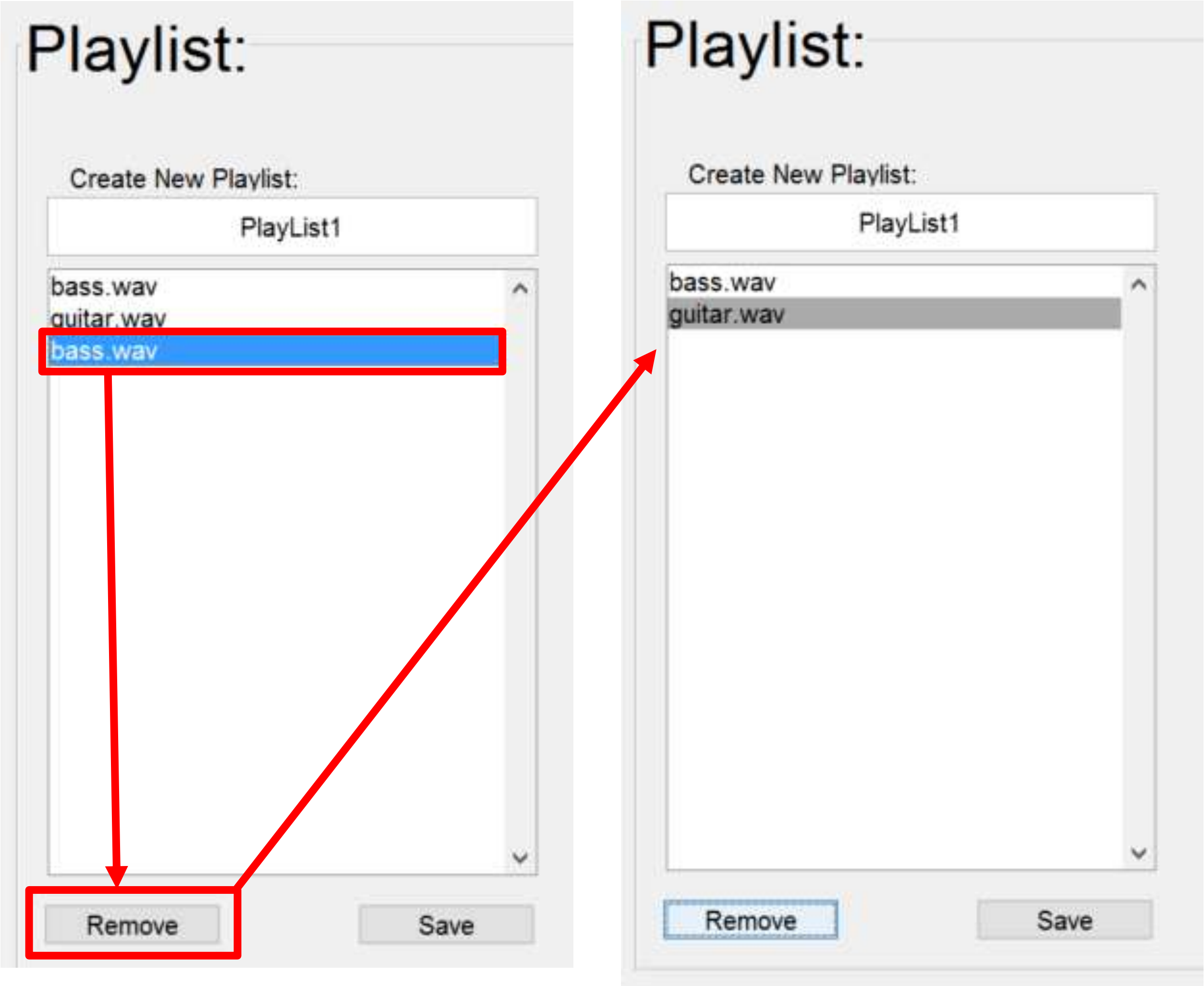
After the user, has added the songs into an empty playlist, the user will then type a name of the playlist into the empty textbox to save the playlist in the cell array.



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:



Create New Playlist: PlayList1

bass.wav
guitar.wav
bass.wav

Create New Playlist: PlayList1

bass.wav
guitar.wav

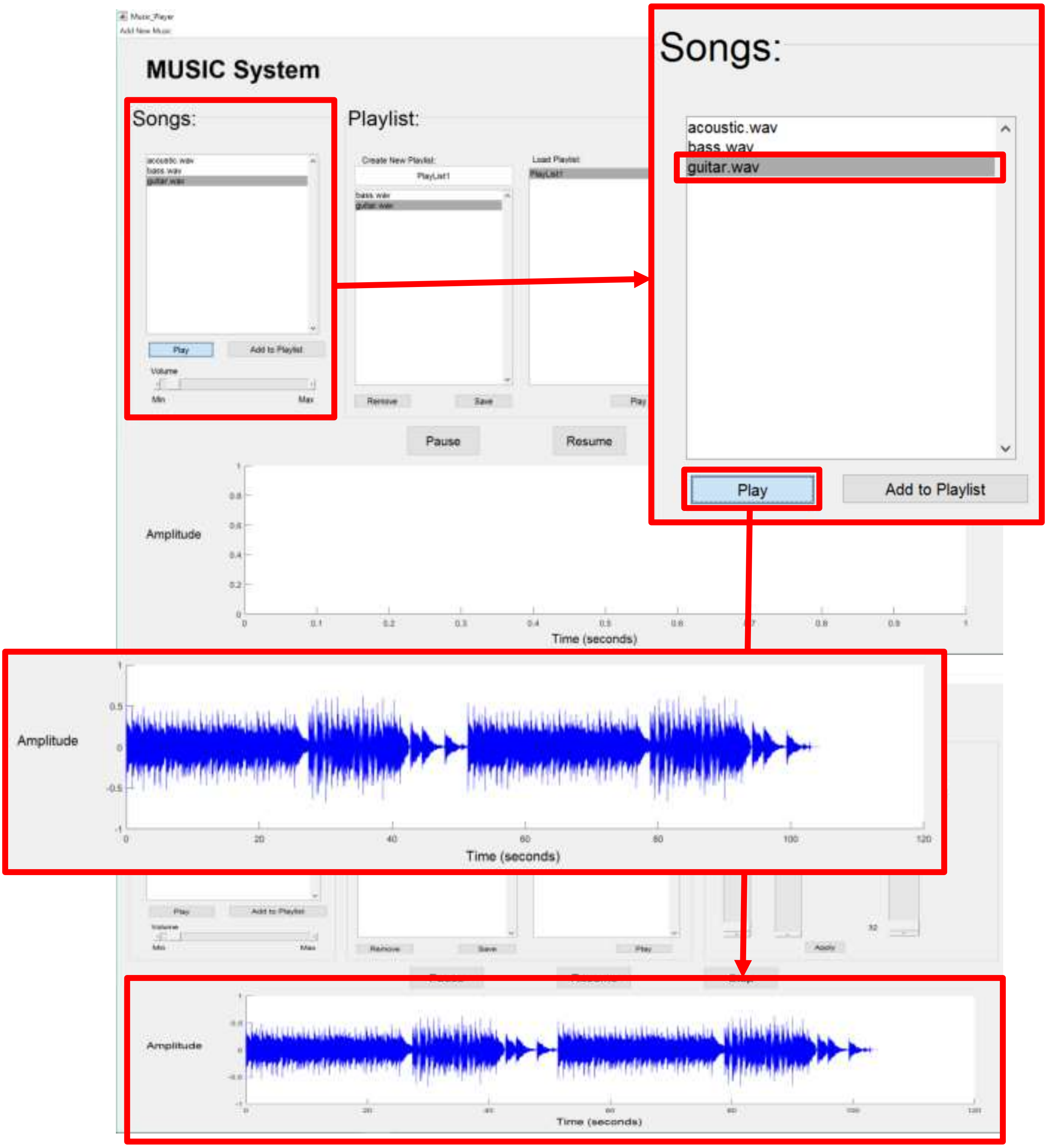
Since 'bass.wav' was a duplicate song in the playlist, we would want to remove it. Therefore, after clicking the 'Remove' button, it removed the value of the index.

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.

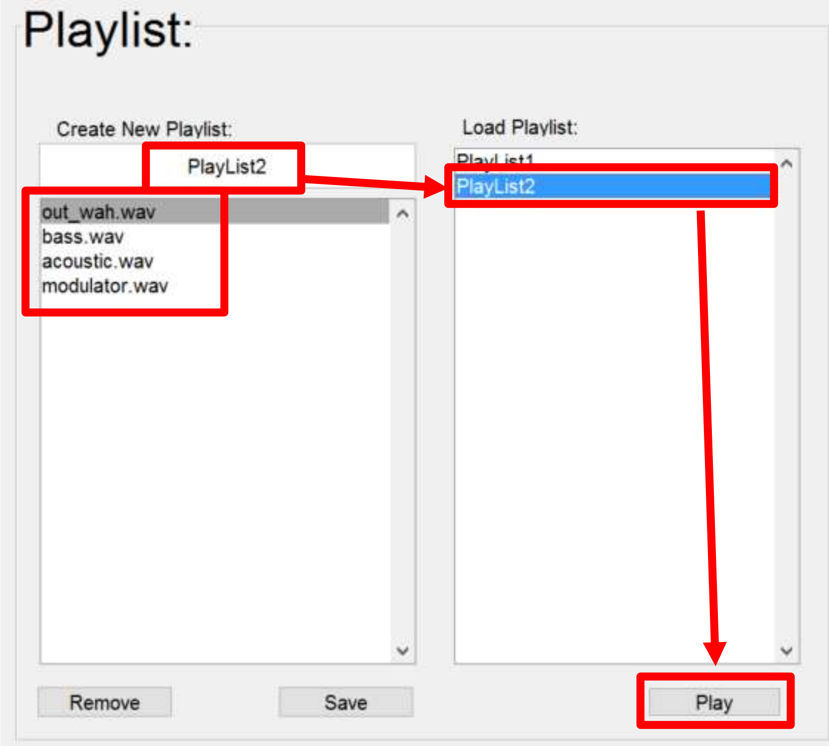
'Play' Button in Songs category:

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

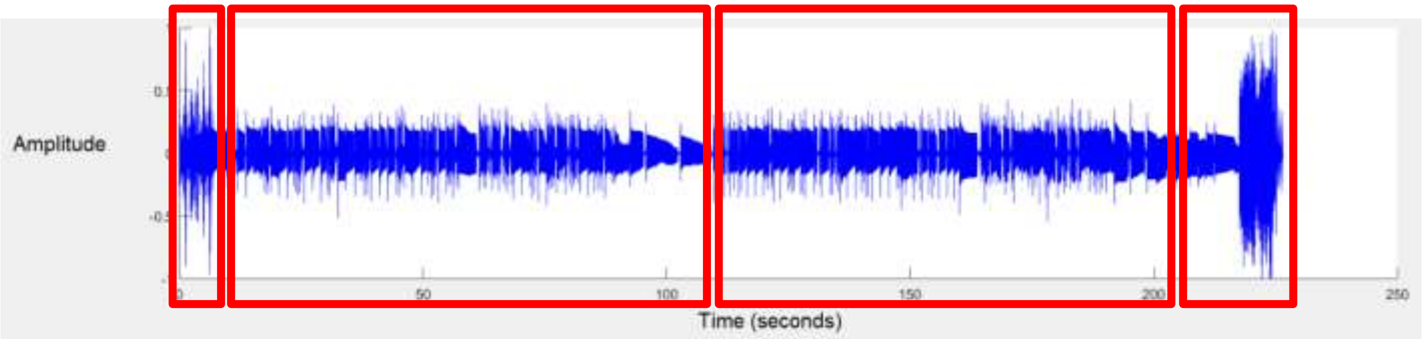


‘Play’ Button in Playlist category:

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’.



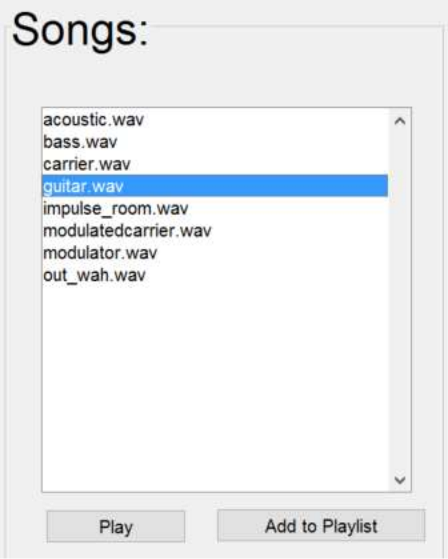
As you can see form above, all the audios are merged into one huge song, and when plotted on the axis, the user can tell the difference in the four songs as they have a different amplitude compared to the other audio files.

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:

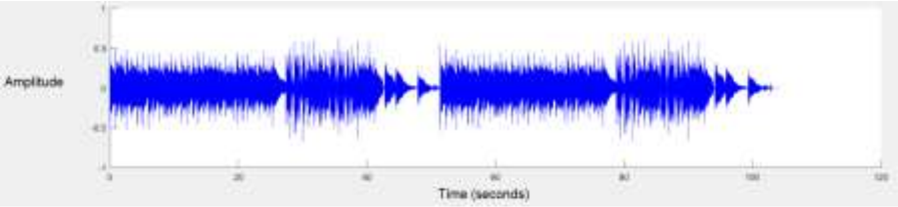
1. Play the song from any list:



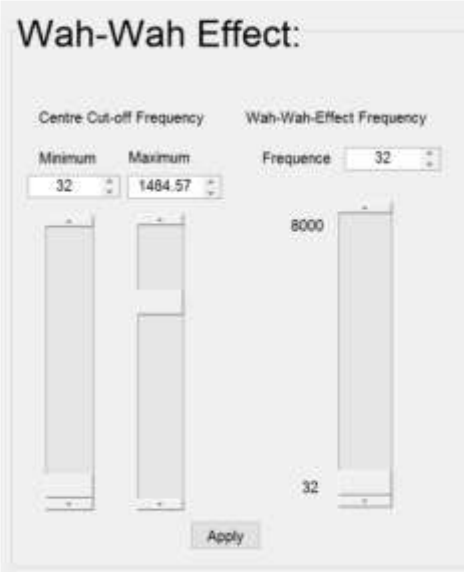
Songs:

acoustic.wav
bass.wav
carrier.wav
guitar.wav
impulse_room.wav
modulatedcarrier.wav
modulator.wav
out_wah.wav

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.



2. Applying the 'Wah-Wah effect':



Wah-Wah Effect:

Centre Cut-off Frequency

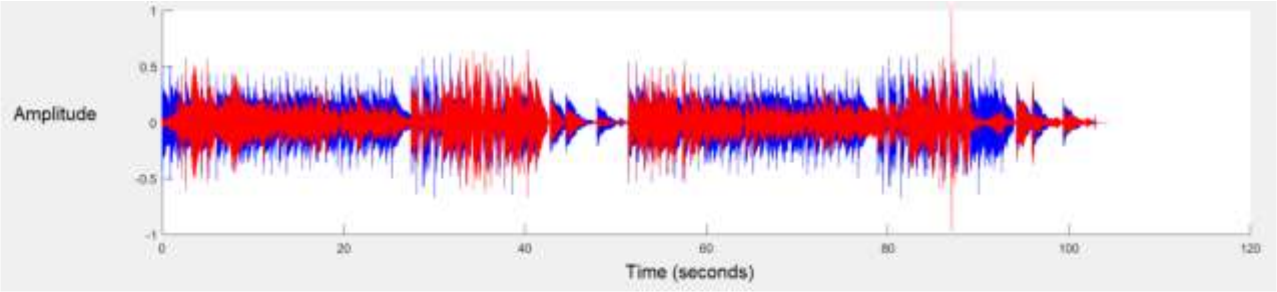
Minimum 32

Maximum 1484.57

Wah-Wah-Effect Frequency

Frequence 32

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
```