Project Title: A Basic Equalizer Made via MATLAB

EE 302 Digital Signal Processing

Instructor: Serap Kırbız
Institution: MEF University

Designers

Mustafa Ali İnan - ID: 041902014
Naci Gökberk Tandoğan - ID: 041802049
Samet Refik Derbentli - ID: 041902001

Abstract and Introduction

An equalizer is tried to be designed via MATLAB. The equalizer filters the signal according to given specifications. The filtration process of the equalizer is conducted by FIR bandpass, lowpass and highpass filters. The equalizer amplifies and compresses amplitudes of different frequency components in compliance with the user's choice. There is a user interface and there are 4 prepared options (Pop, Rock, Bass and Classical).

Problem Specification

Music tracks intensify at different tones and listening to these intensified tones is usually demanded by music lovers. So that, an equalizer can be entitled as "focuser or focalizer". To begin with, a basic equalizer has 10 different frequency intervals which are demonstrated in Fig 1.. Each frequency of the music signal within these intervals should be filtered. And the amplitudes of the frequency components in these 10 intervals should be set by the user. To do that, there should be a user interface that provides user to adjust amplitudes of frequency intervals.

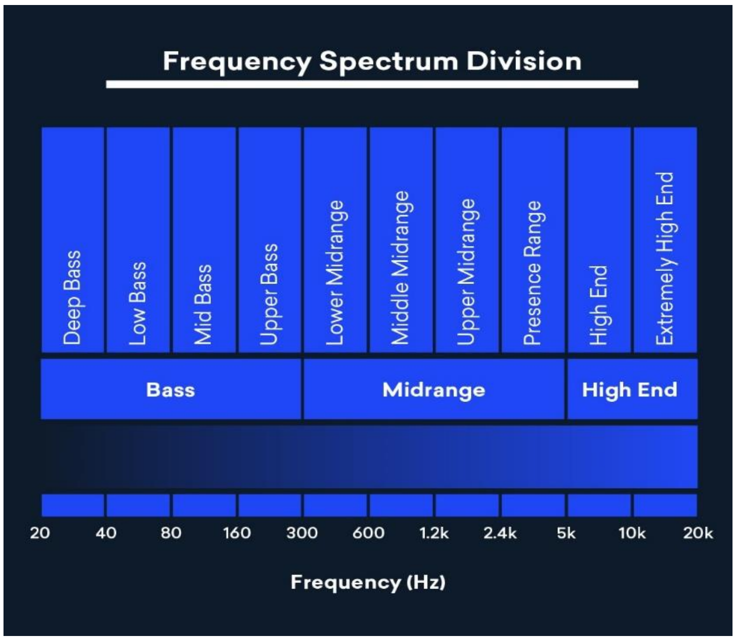


Figure 1: *Frequency Spectrum Division (10 frequency intervals of music tones) [1]*

Data

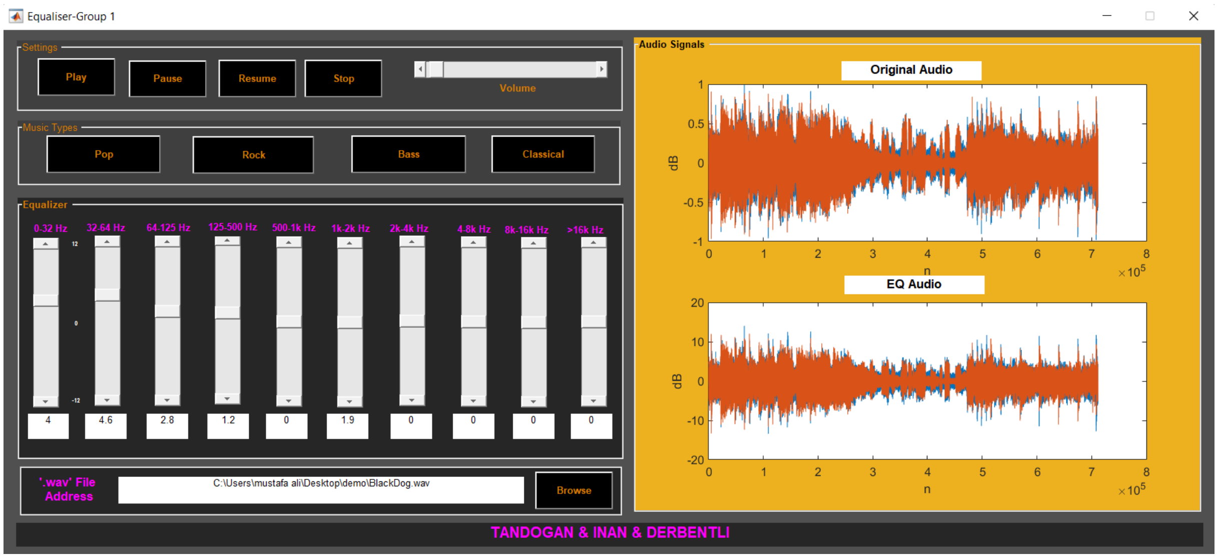
There are three music files which are tested within the equalizer. They are sections from tracks which are of acoustic, bass and rock genres.

Evaluation Criteria

Performance of the equalizer is measured against successfully made online equalizers. Some of other open source equalizers are made more sensitively. Quantitatively, their filters are more pointed and have more prepared specifications. And some of the equalizers are more detailted. They have many more frequency intervals, that provides more focused tones. Example: FX Sound

Approach

The UI (User Interface) is made by GUIDE. GUI is synchronized with the code and figures, sliders, browse pop-up are designed by GUIDE and its code is automatically written. Equalizer UI is designed in another MATLAB figure with the same name as the code file. UI is represented in Fig 2..



0-32 Hz	32-64 Hz	64-125 Hz	125-500 Hz	500-1k Hz	1k-2k Hz	2k-4k Hz	4-8k Hz	8k-16k Hz	>16k Hz
4	4.6	2.8	1.2	0	1.9	0	0	0	0

'.wav' File Address: C:\Users\mustafa ali\Desktop\demo\BlackDog.wav

Figure 2: *UI of the equalizer*

Firstly, music signal should be read then filtered by the equalizer. As it is stated in problem specification, In a basic equalizer, it is predicated on 10 different frequency intervals and amplitude of these frequency components should be adjusted according to desired configuration. Amplitude adjustment is shown in Fig 3..

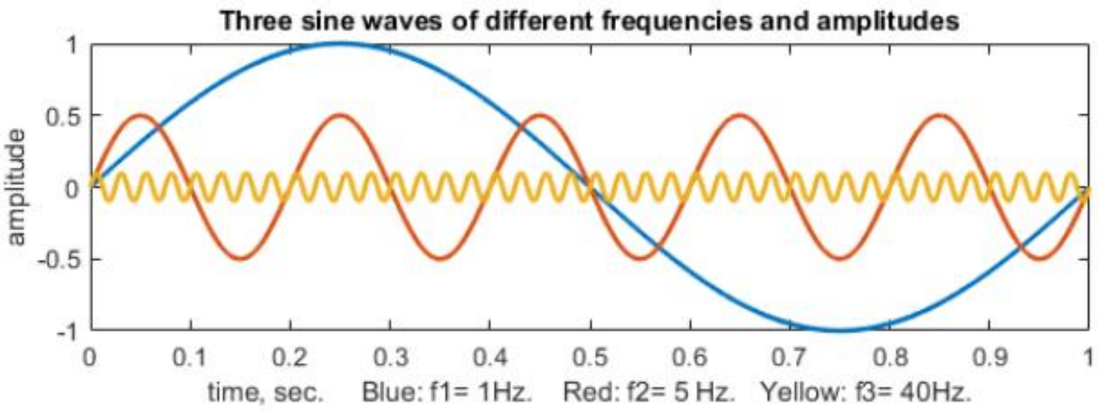


Figure 3: *Changing amplitudes of different frequency components.*

Accordingly, there is a need of 10 FIR filters to design this basic equalizer. There are 1 lowpass, 8 bandpass, 1 highpass filters. Order of the filters (n) are determined as 16. This order is the one of the best-fit order for equalizing the music signal. This value is found by trying many times. If order is too high some music frequency components get lost. This case is demonstrated in Fig 4,5,6,7.

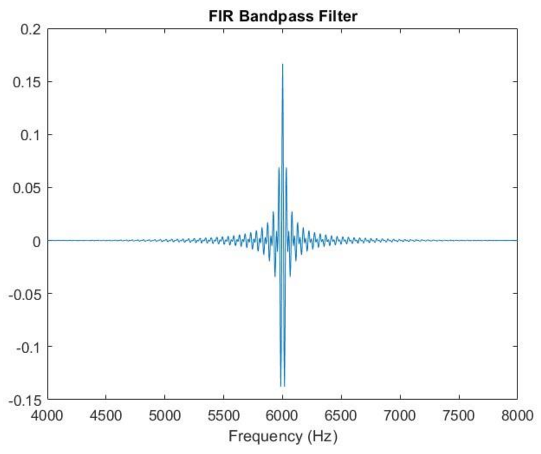


Figure 4: *n = 1000*

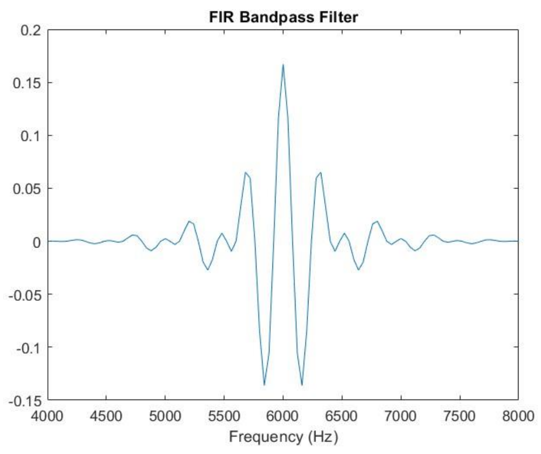


Figure 5: *n = 100*

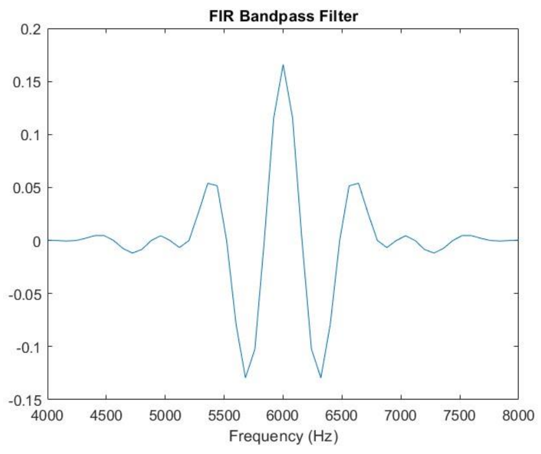


Figure 6: *n = 50*

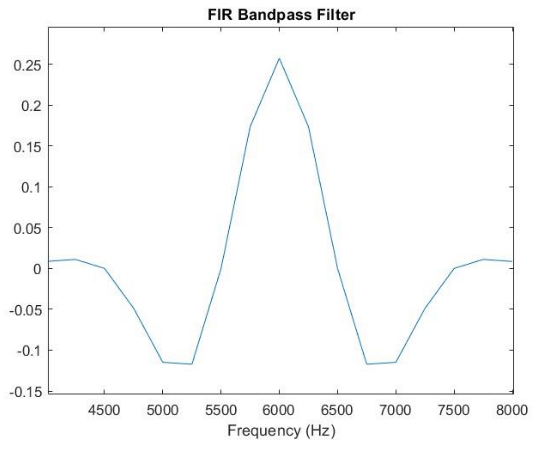


Figure 7: *n=16*

These filters are of the frequency interval 4001-8000 Hz.

After designing filters, filters are respectively applied on the music signal. In other words, music signal is filtered respectively of each frequency interval. Finally, adjustment of these frequency components will be done in amplitude (dB) by the user with sliders or the user can choose one of the prepared configurations.

Results and Analysis

When the tracks are listened as filtered and non-filtered, it can be observed that the music is filtered and equalized well. There is a drawback. If volume is increased by the user, sound becomes hoarse a little bit.

Development

- Filters may be made mor sensitively. For instance, butterworth FIR filters may be used with according ripple values. So that, filters would become more sensitive and equalization will be more steady.
- More specifications could be added.

Conclusions

By designing this equalizer, Designing FIR and butterworth filters, using MATLAB GUI, working procedure of an equalizer are learnt.

Source Code

```
function varargout = equalizer(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @equalizer_OpeningFcn, ...
                   'gui_OutputFcn',  @equalizer_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 equalizer is made visible.
function equalizer_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 equalizer (see VARARGIN)

% Choose default command line output for the program
vol = 1;
set(handles.slider15,'value',vol);
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes equalizer wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = equalizer_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;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename pathname] = uigetfile({'*.wav'},'File Selector');
handles.fullpathname = strcat(pathname, filename);

set(handles.text3, 'String',handles.fullpathname) %showing fullpathname
guidata(hObject,handles)

function play_equalizer(hObject, handles)
global player;
[handles.y,handles.Fs] = audioread(handles.fullpathname);
handles.Volume=get(handles.slider15,'value');
%handles.y=handles.y(NewStart:end,:);
handles.s1=get(handles.slider3,'value');
handles.s2=get(handles.slider4,'value');
handles.s3=get(handles.slider5,'value');
handles.s4=get(handles.slider7,'value');
handles.s5=get(handles.slider8,'value');
 handles.s6=get(handles.slider9,'value');
 handles.s7=get(handles.slider10,'value');
 handles.s8=get(handles.slider11,'value');
 handles.s9=get(handles.slider6,'value');
handles.s10=get(handles.slider12,'value');
set(handles.text16, 'String',handles.s1);
set(handles.text19, 'String',handles.s2);
set(handles.text20, 'String',handles.s3);
set(handles.text21, 'String',handles.s4);
set(handles.text22, 'String',handles.s5);
set(handles.text23, 'String',handles.s6);
set(handles.text24, 'String',handles.s7);
set(handles.text25, 'String',handles.s8);
set(handles.text26, 'String',handles.s9);
set(handles.text27, 'String',handles.s10);

cut_off=31; %cut off low pass dalama Hz
orde=16;
a=fir1(orde,cut_off/(handles.Fs/2),'low');
y1=handles.s1*filter(a,1,handles.y);

% %bandpass1
f1=32;
```

```
f2=64;
b1=fir1(orde,[f1/(handles.Fs/2) f2/(handles.Fs/2)],'bandpass');
y2=handles.s2*filter(b1,1,handles.y);
%
% %bandpass2
f3=65;
f4=125;
b2=fir1(orde,[f3/(handles.Fs/2) f4/(handles.Fs/2)],'bandpass');
y3=handles.s3*filter(b2,1,handles.y);
%
% %bandpass3
 f4=126;
f5=500;
 b3=fir1(orde,[f4/(handles.Fs/2) f5/(handles.Fs/2)],'bandpass');
 y4=handles.s4*filter(b3,1,handles.y);
%
% %bandpass4
 f5=501;
f6=1000;
 b4=fir1(orde,[f5/(handles.Fs/2) f6/(handles.Fs/2)],'bandpass');
 y5=handles.s5*filter(b4,1,handles.y);
%
% %bandpass5
  f7=1001;
f8=2000;
  b5=fir1(orde,[f7/(handles.Fs/2) f8/(handles.Fs/2)],'bandpass');
  y6=handles.s6*filter(b5,1,handles.y);
%
% %bandpass6
  f9=2001;
f10=4000;
  b6=fir1(orde,[f9/(handles.Fs/2) f10/(handles.Fs/2)],'bandpass');
  y7=handles.s7*filter(b6,1,handles.y);
%
% %bandpass7
  f11=4001;
f12=8000;
  b7=fir1(orde,[f11/(handles.Fs/2) f12/(handles.Fs/2)],'bandpass');
  y8=handles.s8*filter(b7,1,handles.y);
%
 % %bandpass8
  f13=8001;
f14=16000;
  b8=fir1(orde,[f13/(handles.Fs/2) f14/(handles.Fs/2)],'bandpass');
  y9=handles.s9*filter(b8,1,handles.y);
%
 %highpass
cut_off2=16001;
c=fir1(orde,cut_off2/(handles.Fs/2),'high');
y10=handles.s10*filter(c,1,handles.y);
%handles.yT=y1+y2+y3+y4+y5+y6+y7;
 handles.yT=y1+y2+y3+y4+y5+y6+y7+y8+y9+y10;
player = audioplayer(handles.Volume*handles.yT, handles.Fs);
 subplot(2,1,1);
 plot(handles.y);xlabel('n');ylabel('dB');
 subplot(2,1,2);
 plot(handles.yT);xlabel('n');ylabel('dB');

guidata(hObject,handles)
```

```matlab
%[y, Fs] = audioread(fullpathname);
% = audioplayer(y, Fs);
%play(player);
%play(player);
%save suara;

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (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 edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (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 pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
global player;
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%equalizer_play();
play_equalizer(hObject, handles);
play(player);
guidata(hObject,handles)

%t=0:1/handles.Fs:(length(handles.player)-1)/handles.Fs;
%plot(handles.yT,handles.axes2);
%set(handles.axes2);
%handles.yT(handles.axes2);


% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (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


% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (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 slider3_Callback(hObject, eventdata, handles)
% hObject    handle to slider3 (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


% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider3 (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',[.3 .3 .3]);
end


% --- Executes on slider movement.
function slider4_Callback(hObject, eventdata, handles)
% hObject    handle to slider4 (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


% --- Executes during object creation, after setting all properties.
function slider4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider4 (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',[.3 .3 .3]);
end


% --- Executes on slider movement.
function slider5_Callback(hObject, eventdata, handles)
% hObject    handle to slider5 (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


% --- Executes during object creation, after setting all properties.
function slider5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider5 (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',[.3 .3 .3]);
end


% --- Executes on slider movement.
function slider6_Callback(hObject, eventdata, handles)
% hObject    handle to slider6 (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


% --- Executes during object creation, after setting all properties.
function slider6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider6 (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',[.3 .3 .3]);
end


% --- Executes on slider movement.
function slider7_Callback(hObject, eventdata, handles)
% hObject    handle to slider7 (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


% --- Executes during object creation, after setting all properties.
function slider7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```matlab
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.3 .3 .3]);
end


% --- Executes on slider movement.
function slider8_Callback(hObject, eventdata, handles)
% hObject    handle to slider8 (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


% --- Executes during object creation, after setting all properties.
function slider8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider8 (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',[.3 .3 .3]);
end


% --- Executes on slider movement.
function slider9_Callback(hObject, eventdata, handles)
% hObject    handle to slider9 (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


% --- Executes during object creation, after setting all properties.
function slider9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider9 (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',[.3 .3 .3]);
end


% --- Executes on slider movement.
function slider10_Callback(hObject, eventdata, handles)
% hObject    handle to slider10 (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


% --- Executes during object creation, after setting all properties.
function slider10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider10 (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',[.3 .3 .3]);
end


% --- Executes on slider movement.
function slider11_Callback(hObject, eventdata, handles)
% hObject    handle to slider11 (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


% --- Executes during object creation, after setting all properties.
function slider11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider11 (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 pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global player;
play_equalizer(hObject, handles);
pause(player);
guidata(hObject,handles)


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global player;
play_equalizer(hObject, handles);
stop(player);
```

```
guidata(hObject,handles)



% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global player;
play_equalizer(hObject, handles);
resume(player);
guidata(hObject,handles)



% --- Executes on slider movement.
function slider12_Callback(hObject, eventdata, handles)
% hObject    handle to slider12 (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


% --- Executes during object creation, after setting all properties.
function slider12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider12 (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',[.3 .3 .3]);
end

% --- Executes on slider movement.
function slider15_Callback(hObject, eventdata, handles)
% hObject    handle to slider15 (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


% --- Executes during object creation, after setting all properties.
function slider15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider15 (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',[.3 .3 .3]);
end
```

```matlab
% --- Executes on button press in Pop.
function Pop_Callback(hObject, eventdata, handles)
% hObject    handle to Pop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s1 = -1.9;
s2 = -1.2;
s3 = 0;
s4 = 1.2;
s5 =  3;
s6 = 3;
s7 = 2.4;
s8= 0.6;
s9 = -0.6;
s10 = -1.2;
set(handles.slider3,'value',s1);
set(handles.slider4,'value',s2);
set(handles.slider5,'value',s3);
set(handles.slider7,'value',s4);
set(handles.slider8,'value',s5);
set(handles.slider9,'value',s6);
set(handles.slider10,'value',s7);
set(handles.slider11,'value',s8);
set(handles.slider6,'value',s9);
set(handles.slider12,'value',s10);
set(handles.text16, 'String',s1);
set(handles.text19, 'String',s2);
set(handles.text20, 'String',s3);
set(handles.text21, 'String',s4);
set(handles.text22, 'String',s5);
set(handles.text23, 'String',s6);
set(handles.text24, 'String',s7);
set(handles.text25, 'String',s8);
set(handles.text26, 'String',s9);
set(handles.text27, 'String',s10);


% --- Executes on button press in Rock.
function Rock_Callback(hObject, eventdata, handles)
% hObject    handle to Rock (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s1 = 4.6;
s2 = 4.3;
s3 = 3.5;
s4 = 3;
s5 = -0.6;
s6 = -1.2;
s7 = 0.6;
s8= 1.9;
s9 = 3.5;
s10 = 4.1;
set(handles.slider3,'value',s1);
set(handles.slider4,'value',s2);
set(handles.slider5,'value',s3);
set(handles.slider7,'value',s4);
set(handles.slider8,'value',s5);
set(handles.slider9,'value',s6);
set(handles.slider10,'value',s7);
set(handles.slider11,'value',s8);
set(handles.slider6,'value',s9);
set(handles.slider12,'value',s10);

set(handles.text16, 'String',s1);
set(handles.text19, 'String',s2);
set(handles.text20, 'String',s3);
set(handles.text21, 'String',s4);
set(handles.text22, 'String',s5);
set(handles.text23, 'String',f6);
set(handles.text24, 'String',s7);
set(handles.text25, 'String',s8);
set(handles.text26, 'String',s9);
set(handles.text27, 'String',s10);


% --- Executes on button press in Bass.
function Party_Callback(hObject, eventdata, handles)
% hObject    handle to Party (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s1 = 4;
s2 =4.6;
s3 = 2.8;
s4 = 1.2;
s5 =  0;
s6 = 1.9;
s7 = 0;
s8= 0;
s9 = 0;
s10 = 0;
set(handles.slider3,'value',s1);
set(handles.slider4,'value',s2);
set(handles.slider5,'value',s3);
set(handles.slider7,'value',s4);
set(handles.slider8,'value',s5);
set(handles.slider9,'value',s6);
set(handles.slider10,'value',s7);
set(handles.slider11,'value',s8);
set(handles.slider6,'value',s9);
set(handles.slider12,'value',s10);

set(handles.text16, 'String',s1);
set(handles.text19, 'String',s2);
set(handles.text20, 'String',s3);
set(handles.text21, 'String',s4);
set(handles.text22, 'String',s5);
set(handles.text23, 'String',s6);
set(handles.text24, 'String',s7);
set(handles.text25, 'String',s8);
set(handles.text26, 'String',s9);
set(handles.text27, 'String',s10);


% --- Executes on button press in Classical.
function Classical_Callback(hObject, eventdata, handles)
% hObject    handle to Classical (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
s1 = 3.6;
s2 = 3.5;
s3 = 3.4;
s4 = 3;
s5 =  -1.2;
s6 = -1.2;
s7 = 1.2;
s8= 2.5;
s9 = 3.8;
s10 = 4;
set(handles.slider3,'value',s1);
set(handles.slider4,'value',s2);
set(handles.slider5,'value',s3);
set(handles.slider7,'value',s4);
set(handles.slider8,'value',s5);
set(handles.slider9,'value',s6);
set(handles.slider10,'value',s7);
set(handles.slider11,'value',s8);
set(handles.slider6,'value',s9);
set(handles.slider12,'value',s10);

set(handles.text16, 'String',s1);
set(handles.text19, 'String',s2);
set(handles.text20, 'String',s3);
set(handles.text21, 'String',s4);
set(handles.text22, 'String',s5);
set(handles.text23, 'String',s6);
set(handles.text24, 'String',s7);
set(handles.text25, 'String',s8);
set(handles.text26, 'String',s9);
set(handles.text27, 'String',s10);
```

References

[1] "Sound Frequency: How to Use the Frequency Spectrum For Better EQ," *LANDR Blog*, Nov. 26, 2019. https://blog.landr.com/sound-frequency-eq/ (accessed Jun. 05, 2022).

[2] L. Soundz, "Understanding Frequency: A Guide To Better EQ In Mixing," *4SoundEngineers*, Aug. 10, 2015. https://www.4soundengineers.com/understanding-frequency-a-guide-to-better-eq-in-mixing/ (accessed Jun. 05, 2022).

[3] J. Langelaar, A. Mattsson, and F. Natvig, "Development of real time audio equalizer application using MATLAB App Designer." Accessed: Jun. 05, 2022. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:1334188/FULLTEXT02

[4]"Designing Lowpass FIR Filters," *Mathworks.com*, 2022. https://www.mathworks.com/help/dsp/ug/designing-low-pass-fir-filters.html (accessed Jun. 05, 2022).