Technical University of Cluj-Napoca

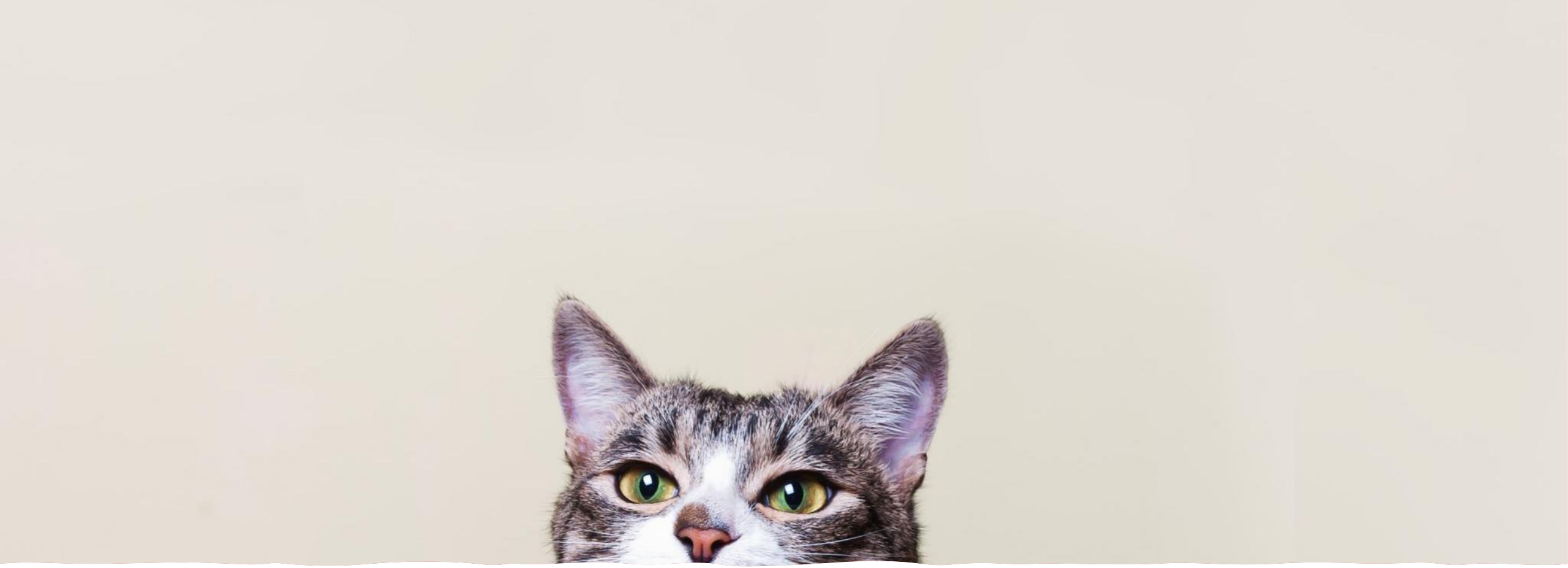
Faculty of Electronics, Telecommunications and Information Technology

MUSIC PLAYER GUI

Student
Razvan Câcu

Coordinators
- Mihaela Cîrlugea
- Paul Farago

2023

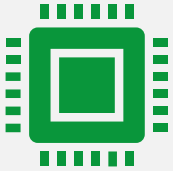


Content

- Introduction
- The Purpose of the project
- Theoretical Presentation
- Exploration Part
- Conclusion
- References
- Appendix

Introduction

MATLAB (short for "Matrix Laboratory") is a programming language and software environment for numerical computation, visualization, and programming. It was developed by MathWorks in the 1970s and is currently used in a wide range of fields, including engineering, finance, and scientific research. One of the main strengths of MATLAB is its ability to work with matrices, which makes it well-suited for linear algebra and other matrix-based computations. It also includes a large library of built-in functions and toolboxes for specific applications, such as signal processing, control systems, and image processing. The first version of MATLAB was released in 1984, and it has been updated and expanded many times since then to include new features and capabilities.



A music player GUI, or graphical user interface, in MATLAB is a program that allows users to play and control audio files within the MATLAB environment.

Such a GUI typically includes controls for playing, pausing, and stopping the audio, as well as adjusting the volume and seeking within the file. It may also include visualizations of the audio data, such as a spectrogram or oscilloscope display.

The Purpose of the project

- The purpose of a music player GUI in Matlab is to provide a convenient and user-friendly interface for playing and controlling audio files within the Matlab environment. The main goal is to make it easy for users to access and play audio files, as well as adjust various settings such as volume, playback position, and equalization.
- Additionally, the purpose of a music player GUI can be to improve the user's experience by providing additional functionalities like, creating playlists, displaying of meta-data such as the song title and artist information, the ability to shuffle songs and repeat a song or a list, etc.
- Furthermore, it may also include visualizations of the audio data, such as a spectrogram or oscilloscope display, which can be helpful for audio analysis and processing, in areas such as speech, audio recognition and analysis.
- In summary, the purpose of a music player GUI in Matlab is to provide a way to play, control and analyze audio files within Matlab environment, in a convenient and user-friendly manner, making it a useful tool for both audio processing and analysis as well as providing a simple audio playback solution.

Theoretical Presentation

- "audioread" function is a built-in function in MATLAB and can be used to read audio files. It can read a wide range of audio file formats including WAV, MP3, AIFF, and others. The function takes the file name or path as an input and returns a matrix of audio samples and a sample rate.
- "audioplayer" is a built-in class in MATLAB that can be used to play audio data. It can take an audio matrix and a sample rate as input and play the corresponding audio. An object of the class audioplayer is created and then it can be used to play the sound by calling the play method on that object.
- "spectrogram" function is a built-in function in MATLAB that can be used to visualize the frequency content of an audio signal over time. The function takes an audio signal and a sample rate as input and returns a matrix of frequency data and a vector of frequencies. It can also be used to display the frequency content in a 2D plot, where the x-axis represents time, and the y-axis represents frequency.

Experimental Part

The GUI contains multiple axes for displaying different information such as a spectrogram, waveform, and sound amplitude.

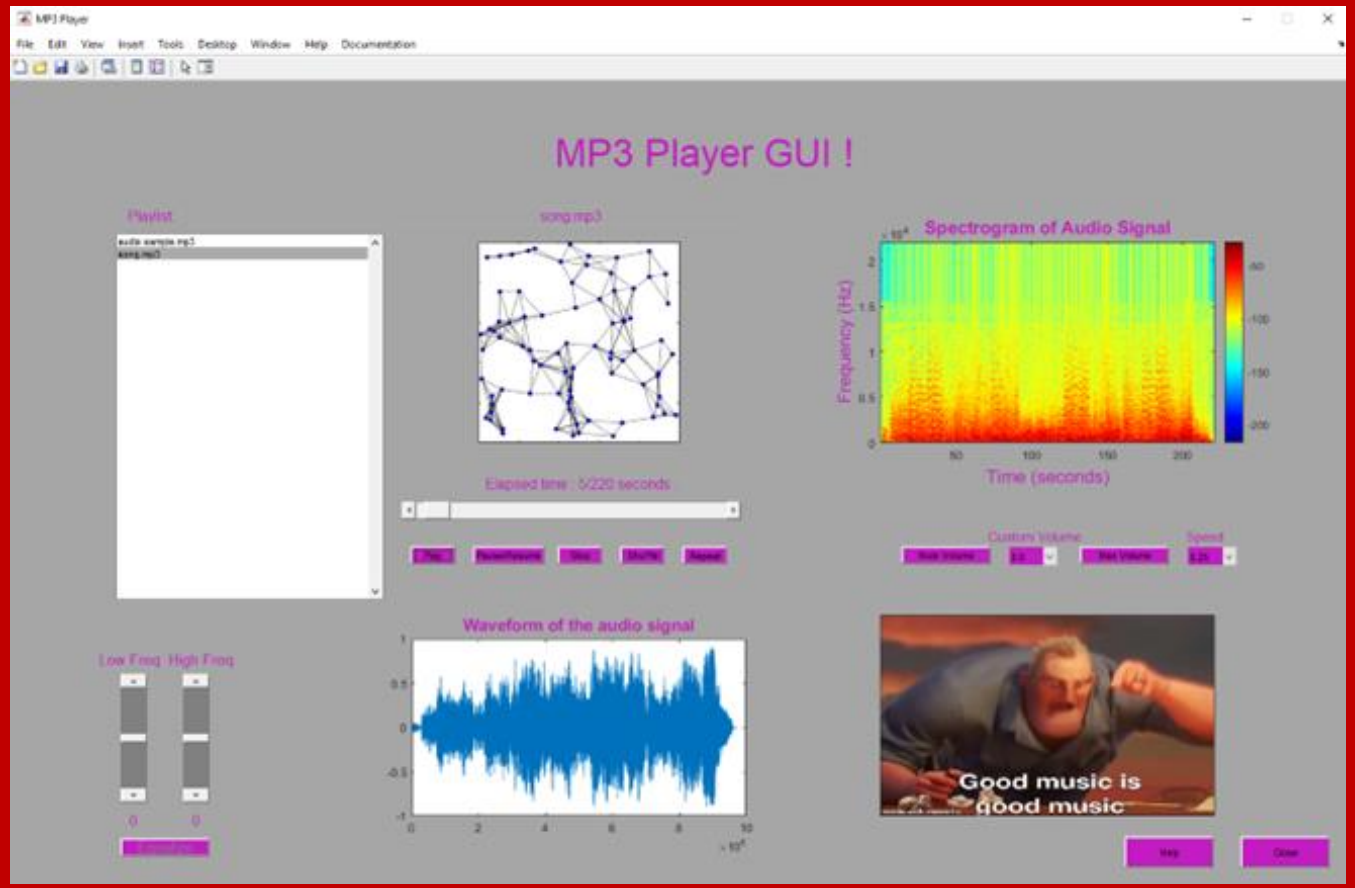
A "Documentation" menu item was added to the GUI.

The GUI contains a listbox for displaying a list of MP3 files in the current folder.

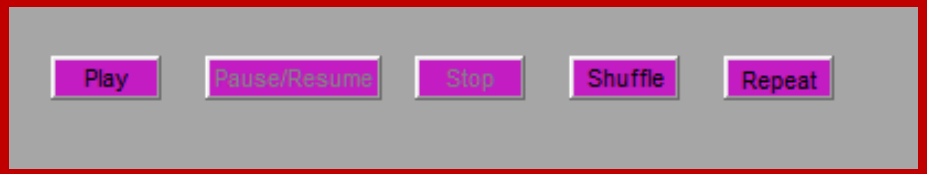
It has several labels and textboxes for displaying information such as elapsed time and the song that is currently playing.

The GUI contains several buttons for controlling audio playback such as 'Play', 'Pause/Resume', 'Stop', and 'shuffle'.

The Interface



Description of the buttons



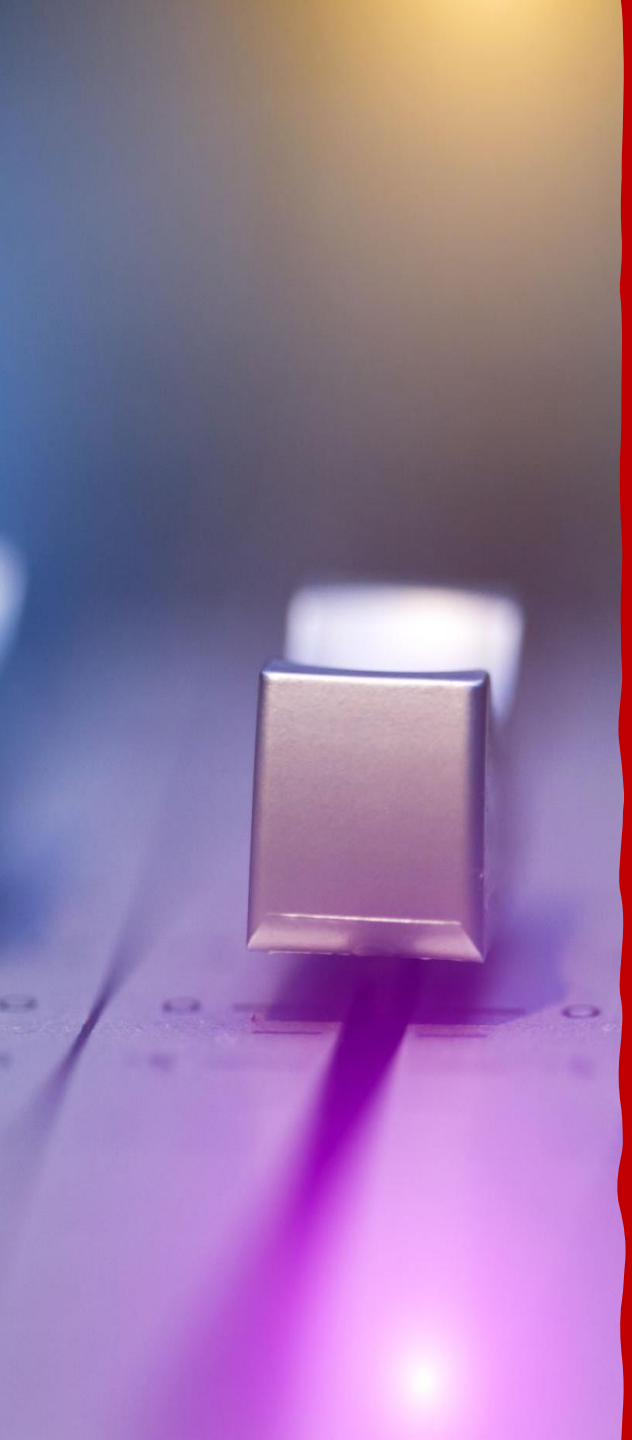
When the "Play" button is pressed, it uses the MATLAB functions 'audioplayer' and 'audioreader' to play the selected song from the playlist. It also activates the "Pause/Resume" and "Stop" buttons. Additionally, when the song is playing, the spectrogram of the song is shown on the top right axes. The top left axes displays random figures using a custom function, the bottom left axes displays the amplitude of the audio signal, and finally on the bottom right axes displays a random photo.

When the "Pause" button is pressed, the audio player is paused and the top left axes stops updating with new figures, also, the bottom right axes displays a specific random photo. Pressing the "Pause" button again will resume the playback and the top left axes will continue to generate new figures while the song continues playing, similar to the functionality of the "Play" button.

Pressing the "Stop" button brings the audio playback to an immediate halt, returning all settings to their initial state. Additionally, the "Pause" button is deactivated, ensuring a clean and fresh start for the next audio selection.

Activating the "Shuffle" button shuffles the order of the songs in the playlist, providing a randomized and diverse listening experience.

Toggling the "Repeat" button allows for the currently playing song to be repeated continuously until it is disabled .



- The "Mute Volume" button allows you to instantly silence the audio playback .
- The "Custom Volume" button allows users to adjust the volume to their desired level by providing a range of selectable volume options.
- Pressing the "Max Volume" button will boost the audio to its highest level, allowing you to fully immerse in your listening experience.
- The "Speed" dropdown menu allows the user to select a desired playback speed from a variety of options, adjusting the playback of the song to the chosen speed, and providing a personalized and dynamic listening experience.

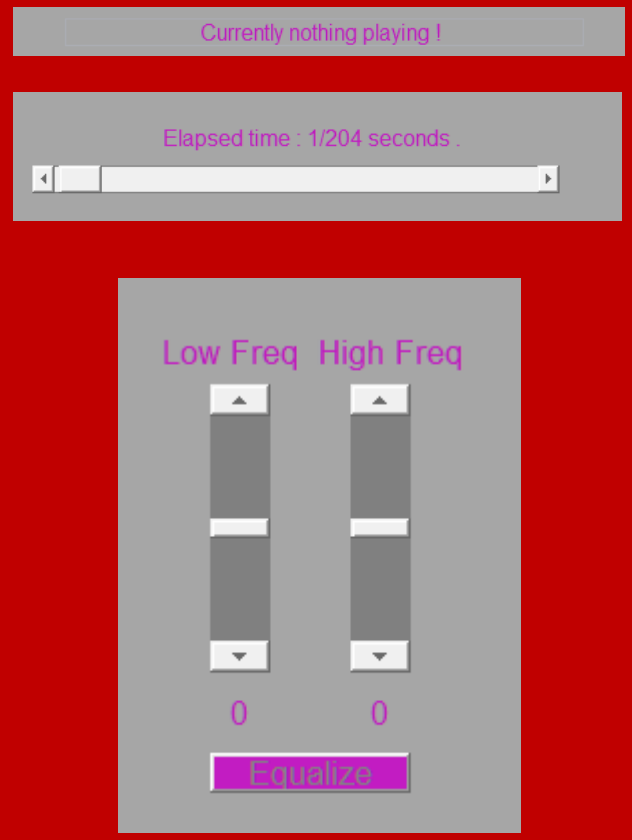
Sliders and textboxes

The "Label" textbox dynamically updates to reflect the current state of the playback, displaying the name of the song when play, "Paused" when on hold, and "Currently nothing playing!" when playback is stopped.

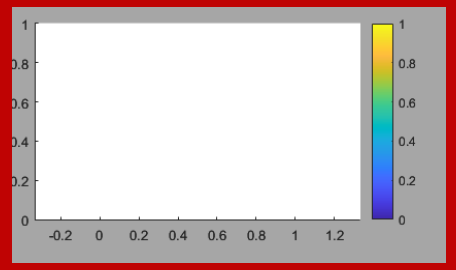
The "Elapsed Time" label displays the current second of the song being played and the total duration, allowing you to keep track of the progress of the audio playback.

The "Low Freq" and "High Freq" sliders let the user control the low and high frequencies of the audio signal respectively and manipulate it to their liking. It enables them to reshape the audio signal .

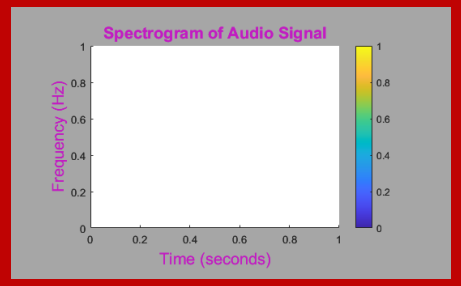
Once the desired low and high frequency gains are selected, the "Equalize" button becomes active. When pressed, it will play the resampled signal, adjusting the audio signal to the selected frequencies and applying the same changes to the visualization as the "Play" button does .



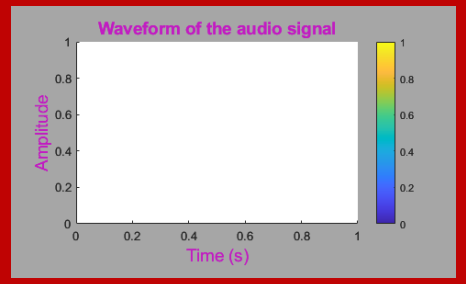
Axes



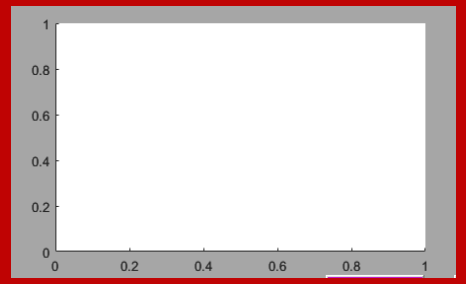
This axes, using the unoriented graphs principle, will dynamically generate unique and random figures while the audio is playing, providing an interesting visual representation of the audio playback.



When the 'spectrogram' function is called, this axes will showcase the audio's frequency content by presenting the spectrogram of the input audio sound.



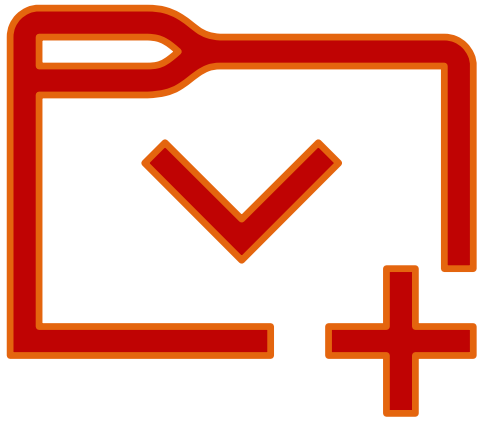
As the audio plays, this axes will display the waveform of the audio signal, providing a visual representation of the audio's amplitude over time

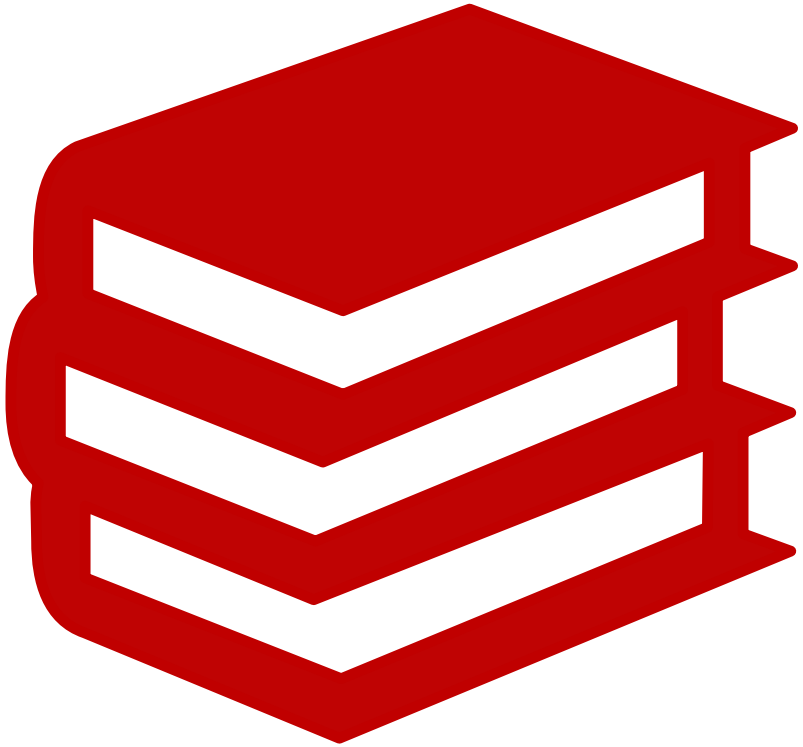


This axes is used to provide a dynamic visual representation during audio playback, displaying a selection of random images in response to user interactions with the buttons.

Conclusion

- In conclusion, the implementation of the music player GUI was successful, I was able to implement all my ideas, such as the spectrogram, waveform, randomly generated photos, volume controls, and playback speed controls, among other features, to create an engaging and comprehensive audio listening experience.





References

https://www.mathworks.com/help/matlab/ref/audioread.html

https://www.mathworks.com/help/signal/ref/spectrogram.html

https://www.mathworks.com/help/matlab/ref/audioplayer.html

Appendix

```
Function mp3_player(audioState,selectionString, player, fs,
lowFreqGain, highFreqGain, leqState, heqState)


SoundVolume(1);
% Create a figure window
f = figure('Name', 'MP3 Player','Position',[100 100 1500 900] ,'Color',
[0.65,0.65,0.65],Resize='off',NumberTitle='off');


% Create an axes object
ax = axes(f, 'Position', [0.3,0.55,0.25,0.25]);
axis equal
colorbar;
waveformaxes= axes(f, 'Position', [0.3,0.086,0.25,0.22]);
xlabel('Time (s)',Color=[ 0.76 , 0.11, 0.76 ],FontSize=14);
ylabel('Amplitude',Color=[ 0.76 , 0.11, 0.76 ],FontSize=14);
title(waveformaxes,'Waveform of the audio signal',Color=[ 0.76 ,
0.11, 0.76 ],FontSize=14);
colorbar;
playAxes = axes(f, 'Position', [0.65,0.086,0.25,0.25]); % adjust the
position of the axes
ax1 = axes(f,'Position', [0.65,0.55,0.25,0.25]);


doc=uimenu(f,"Text",'Documentation');
uimenu(doc,"Text",'Documentation');
colorbar;
xlabel('Time (seconds)',Color=[0.76 , 0.11, 0.76],FontSize=15);
ylabel('Frequency (Hz)',Color=[0.76 , 0.11, 0.76],FontSize=15);
title('Spectrogram of Audio Signal',Color=[0.76 , 0.11,
0.76],FontSize=15);


% Create a listbox
listbox = uicontrol(f, 'Style', 'listbox', 'Position', [120,320,300,410]);


uicontrol(f, 'Style', 'text', 'String', 'Playlist:', 'Position',
[120,740,80,20],'BackgroundColor',[0.65,0.65,0.65],'ForegroundColo
r','[ 0.76 , 0.11, 0.76 ] ',FontSize=12 );


% Add some items to the listbox
% Get a list of all MP3 files in the folder
mp3Files = dir('*.mp3');
%
% Create a cell array of file names
items = {mp3Files.name};


% Set the listbox strings to the file names
set(listbox, 'String', items);
uicontrol(f, 'Style', 'text', 'Position',
[264,770,1033,78], 'String', 'MP3 Player GUI
!','ForegroundColor','[ 0.76 , 0.11, 0.76 ]
','BackgroundColor',[0.65,0.65,0.65],FontSiz
e=32);


elapsedTimeTextbox = uicontrol(f, 'Style',
'text', 'Position', [471,440,341,20], 'String',
'Elapsed time : -/- seconds
.','ForegroundColor','[ 0.76 , 0.11, 0.76 ]
','BackgroundColor',[0.65,0.65,0.65],FontSiz
e=12);


songTextbox = uicontrol(f, 'Style', 'edit',
'Position', [440,740,380,20], 'String',
'Currently nothing playing
!','BackgroundColor',[0.65,0.65,0.65],'Foreg
roundColor','[ 0.76 , 0.11, 0.76 ]
',FontSize=12);


% Create a play button
playButton = uicontrol(f, 'Style',
'pushbutton', 'String',
'Play','BackgroundColor','[ 0.76 , 0.11, 0.76 ]
' ,'Position', [450,360,50,20], 'Callback',
@playButtonCallback,Enable='off');


% Create a pause button
pauseButton = uicontrol(f, 'Style',
'pushbutton', 'String', 'Pause/Resume',
'BackgroundColor','[ 0.76 , 0.11, 0.76 ] '
,'Position', [520,360,80,20], 'Callback',
@pauseButtonCallback,Enable='off');


% Create a stop button
stopButton = uicontrol(f, 'Style',
'pushbutton', 'String',
'Stop','BackgroundColor','[ 0.76 , 0.11,
0.76 ] ' , 'Position' ,[615,360,50,20],
'Callback',
@stopButtonCallback,Enable='off');


% Create a shuffle button
uicontrol(f, 'Style', 'pushbutton', 'String',
'Shuffle','BackgroundColor','[ 0.76 , 0.11,
0.76 ] ' , 'Position', [685,360,50,20],
'Callback',
@shuffleButtonCallback,Enable='on');


% Create a repeat button
uicontrol(f, 'Style', 'togglebutton', 'String',
'Repeat','BackgroundColor','[ 0.76 , 0.11,
0.76 ] ' , 'Position', [755,360,50,20],
'Callback',
@repeatButtonCallback,Enable='on');


%Create a close button
uicontrol(f, 'Style', 'pushbutton', 'String',
'Close','BackgroundColor','[ 0.76 , 0.11,
0.76 ] ' , 'Position', [1380,20,100,35],
'Callback', @closeButtonCallback);
```

```
%Create a random button
uicontrol(f, 'Style', 'pushbutton', 'String',
'Help','BackgroundColor','[ 0.76 , 0.11, 0.76 ] '
,'Position', [1250,20,100,35],'Callback', @rollCallback);


% Create a time slider
timeSlider = uicontrol(f, 'Style', 'slider', 'Min', 0, 'Max',
100, 'Value', 0, 'Position', [440,410,380,20],'Callback',
@timeSliderCallback,Enable='off');


% Create a button for muting the audio
uicontrol('Style','pushbutton','String','Mute
Volume','BackgroundColor','[ 0.76 , 0.11, 0.76 ]
','Position',[1000,360,100,20],'Callback',@muteCallback
);


% Create a button for setting the audio to maximum
volume
uicontrol('Style','pushbutton','String','Max
Volume','BackgroundColor','[ 0.76 , 0.11, 0.76 ]
','Position',[1200,360,100,20],'Callback',@maxVolumeC
allback);


uicontrol('Style','text','Position',[1280,380,120,20],'Strin
g',' Speed ','BackgroundColor',[0.65 0.65
0.65],FontSize=11,ForegroundColor=[0.76 , 0.11, 0.76]);


uicontrol('Style','popup','String',{'0.25','0.5','0.75','1','1.
25','1.5','1.75','2','5'},'BackgroundColor','[ 0.76 , 0.11,
0.76 ]','Position',[1320,360,56,20],
'Callback',@changeSpeed);


uicontrol('Style','text','Position',[1090,380,120,20],'Strin
g',' Custom Volume ','BackgroundColor',[0.65 0.65
0.65],FontSize=11,ForegroundColor=[0.76 , 0.11, 0.76]);


uicontrol('Style','popup','String',{'0.0','0.1','0.2','0.3','0.4','0.
5','0.6','0.7','0.8','0.9','1.0'},'Position',[1120,360,55,20],'Bac
kgroundColor','[ 0.76 , 0.11, 0.76
]','Callback',@volumeCallback);


uicontrol(f, 'Style', 'text', 'String', 'Low
Freq','ForegroundColor','[ 0.76 , 0.11, 0.76 ] ', 'Position',
[100,244,70 ,20],BackgroundColor=[0.65 0.65
0.65],FontSize=13);
uicontrol(f, 'Style', 'text', 'String', 'High
Freq','ForegroundColor','[ 0.76 , 0.11, 0.76 ] ', 'Position',
[170, 244, 90, 20],BackgroundColor=[0.65 0.65
0.65],FontSize=13);
lowFreqSlider = uicontrol(f, 'Style', 'slider', 'Min',-12, 'Max',
12,'Value',0,'BackgroundColor','[ 0.5 , 0.5, 0.5 ] ','Position',
[125,93,30,144],'SliderStep',[0.1
0.1],'Callback',@lowfreqSliderCallback);


highFreqSlider = uicontrol(f, 'Style', 'slider', 'Min',-12, 'Max',
12,'Value',0,'BackgroundColor','[ 0.5 , 0.5, 0.5 ] ','Position',
[195,93,30,144],'SliderStep',[0.1
0.1],'Callback',@highfreqSliderCallback);


lowFreqTextbox = uicontrol(f, 'Style',
'text','BackgroundColor','[ 0.65 , 0.65, 0.65 ]
','ForegroundColor','[ 0.76 , 0.11, 0.76 ] ', 'Position', [125,
63, 30, 20], 'String', '0',FontSize=12);


highFreqTextbox = uicontrol(f, 'Style', 'text',
'BackgroundColor','[ 0.65 , 0.65, 0.65 ]
','ForegroundColor','[ 0.76 , 0.11, 0.76 ] ','Position', [195,
63, 30, 20], 'String', '0',FontSize=12);


equalizeButton = uicontrol(f, 'Style', 'pushbutton', 'String',
'Equalize','BackgroundColor','[ 0.76 , 0.11, 0.76 ] '
,'Position', [125 ,33,100,20], 'Callback',
@equalizeButtonCallback,Enable='off',FontSize=13);


if isempty(items)
% If the listbox is empty, display an error message and
close the figure
errordlg('The playlist is empty. Please add some songs
first.', 'Error', 'modal');
close(f);
return;
else
set(playButton,'Enable','on');
end
```

```
function playButtonCallback(~, ~)
% Get the selected item from the listbox
set(pauseButton,'Enable','on');
set(timeSlider,'Enable','on');
set(stopButton,'Enable','on');
selection = get(listbox, 'Value');
% Get the corresponding string from the list of items
selectionString = items{selection};
% Read the audio data from the file
[y, fs] = audioread(selectionString);
% Average the left and right channels into a single
channel
y = mean(y, 2);
% Play the audio data



plot(waveformaxes,y);
xlabel('Time (s)',Color=[ 0.76 , 0.11, 0.76
],FontSize=14);
ylabel('Amplitude',Color=[ 0.76 , 0.11, 0.76
],FontSize=14);
title(waveformaxes,'Waveform of the audio
signal',Color=[ 0.76 , 0.11, 0.76 ],FontSize=14);
player = audioplayer(y, fs);
play(player);
% Set the maximum value of the time slider to the
length of the song
% (in seconds)
set(timeSlider, 'Max', length(y) / fs);
% Update the time slider as the song plays
audioState = 1;
index =0;


% Call the plotSpectrogram function to plot the
spectrogram
plotSpectrogram(ax1,y, fs);
axes(playAxes);
playDisplayRandomPhoto;
```

```
while audioState == 1
set(songTextbox, 'String', selectionString);
currentTime = player.CurrentSample / fs;
set(timeSlider, 'Value', currentTime);
elapsedTime = floor(player.CurrentSample / fs);
totalDuration = floor(player.TotalSamples / fs);
% Convert to strings
elapsedTimeStr = num2str(elapsedTime);
totalDurationStr = num2str(totalDuration);
% Update elapsed time display
set(elapsedTimeTextbox, 'String', ['Elapsed time : '
elapsedTimeStr '/' totalDurationStr ' seconds .']);
if index == 20
pause(0.3);
visualize(player);
index=0;
else
index=index+1;
end
drawnow;
if player.CurrentSample == player.TotalSamples
stop(player);
audioState = 0;
stopDisplayRandomPhoto;
set(songTextbox, 'String', "Currently nothing playing ! ");
set(timeSlider, 'Value', 0);
end
end
end
```

```
function visualize(~)
axes(ax);
% Set the parameters for the RGG function
nodes = randi([50, 100]); % Random integer between 20 and 200
radius = rand(1) * 0.3 + 0.1; % Random value between 0.1 and 0.7
x = single(rand(nodes,1));
y = single(rand(nodes,1));
A = false(nodes);
for j = 1:nodes
x0 = x(j); y0 = y(j);
distance = sqrt((x - x0).^2 + (y - y0).^2);
neigh = zeros(nodes,1);
neigh(distance <= radius) = true;
A(j,:) = logical(neigh);
A(j,j) = false;
end
switcher = 0;
% Plot the visualizer
plotVisualizer(A, x, y, ax,radius, nodes, switcher);


end
```

```
function pauseButtonCallback(~, ~)
% Check the current status of the audioplayer
object
if strcmp(player.Running, 'on')
% Pause the audio playback
audioState = 0;
% Pause the audioplayer object
pause(player);
axes(playAxes);
stopDisplayRandomPhoto;
set(songTextbox, 'String', "Paused !");
else
axes(playAxes);
% Resume the audio playback
audioState = 1;
% Resume the audioplayer object
resume(player);
playDisplayRandomPhoto;
index=0;
while audioState == 1
set(songTextbox, 'String', selectionString);
set(timeSlider, 'Value', player.CurrentSample / fs);
% Update the visualizer
if index == 20
visualize(player);
index=0;
else
index=index+1;
end
drawnow;
end
end
end
```

```
function stopButtonCallback(~, ~)

set(pauseButton,'Enable','off');
% Stop the audio playback and reset the current position to the beginning
audioState = -1;
set(songTextbox, 'String', "Currently nothing playing ! ");
set(timeSlider, 'Value', 0);
set(elapsedTimeTextbox,'String','Elapsed time : -/- seconds .');
% Stop the audioplayer object and reset the current position to the
beginning
axes(playAxes);
stop(player);
stopDisplayRandomPhoto;
end

function shuffleButtonCallback(~, ~)
% Shuffle the items in the listbox
items = items(randperm(length(items)));
set(listbox, 'String', items);
axes(playAxes);
shuffleDisplayRandomPhoto;
end

function repeatButtonCallback(hObject, ~)
axes(playAxes);
% Check the value of the toggle button
if get(hObject, 'Value') == 1
% Set listbox selection to current song
set(listbox, 'Value', get(listbox, 'Value'));
% Set listbox selection mode to single
set(listbox, 'Max', 1, 'Min', 1);
else
% Set listbox selection mode to multiple
set(listbox, 'Max', inf, 'Min', 0);
end
repeatDisplayRandomPhoto;
end
```

```
function volumeCallback(hObject, ~)
volume = hObject.String{hObject.Value};
volume = str2double(volume);
SoundVolume(volume);
end

function plotSpectrogram(ax1,y, fs)



% Compute the spectrogram
window = hamming(512);
noverlap = 256;
[~,F,T,P] = spectrogram(y,window,noverlap,512,fs);

% Plot the spectrogram
imagesc(T,F,10*log10(P));


colorbar;
xlabel('Time (seconds)',Color=[0.76 , 0.11, 0.76],FontSize=15);
ylabel('Frequency (Hz)',Color=[0.76 , 0.11, 0.76],FontSize=15);
title('Spectrogram of Audio Signal',Color=[0.76 , 0.11, 0.76],FontSize=15);
set(ax1, 'Position', [0.65,0.55,0.25,0.25]);
set(ax1,'YDir','normal');
end

function closeButtonCallback(~, ~)
% Close the figure window
close(f);

% Close any other objects you want to close (e.g. figures, UI objects, etc.)
close all;
end
function muteCallback(~, ~)
% Close the figure window
SoundVolume(0);
end
```

```
function maxVolumeCallback(~, ~)
% Close the figure window
SoundVolume(1);
end


function rollCallback(~, ~)
% Open the web link in the default web browse
web('https://www.youtube.com/watch?v=dQw4w9WgXcQ');
end

function playDisplayRandomPhoto
axes(playAxes);

% Get a list of all image files in the folder
imageFiles = dir('play/*.jpg');

% Choose a random image file
randomIndex = randi(numel(imageFiles));
randomImage = imageFiles(randomIndex);


% Read in the image file
imageData = imread(fullfile('play', randomImage.name));


% Display image in plot
h = image(imageData);
% Set CDataMapping to 'scaled' to scale data values to colormap
set(h, 'CDataMapping', 'scaled');
% Set XData and YData to specify coordinate limits
set(h, 'XData', [1 size(imageData,2)]);
set(h, 'YData', [1 size(imageData,1)]);
% Delete x-axis label
xlabel('');
% Delete y-axis label
ylabel('');
% Delete x-axis tick marks and labels
set(gca, 'XTick', []);
% Delete y-axis tick marks and labels
set(gca, 'YTick', []);
end
```

```
function stopDisplayRandomPhoto
axes(playAxes);


% Get a list of all image files in the folder
imageFiles = dir('stop/*.jpg');


% Choose a random image file
randomIndex = randi(numel(imageFiles));
randomImage = imageFiles(randomIndex);


% Read in the image file
imageData = imread(fullfile('stop', randomImage.name));


% Display image in plot
h = image(imageData);
% Set CDataMapping to 'scaled' to scale data values to colormap
set(h, 'CDataMapping', 'scaled');
% Set XData and YData to specify coordinate limits
set(h, 'XData', [1 size(imageData,2)]);
set(h, 'YData', [1 size(imageData,1)]);
% Delete x-axis label
xlabel('');
% Delete y-axis label
ylabel('');
% Delete x-axis tick marks and labels
set(gca, 'XTick', []);
% Delete y-axis tick marks and labels
set(gca, 'YTick', []);
end
```

```
function shuffleDisplayRandomPhoto
axes(playAxes);


% Get a list of all image files in the folder
imageFiles = dir('shuffle/*.jpg');


% Choose a random image file
randomIndex = randi(numel(imageFiles));
randomImage = imageFiles(randomIndex);


% Read in the image file
imageData = imread(fullfile('shuffle', randomImage.name));


% Display image in plot
h = image(imageData);
% Set CDataMapping to 'scaled' to scale data values to colormap
set(h, 'CDataMapping', 'scaled');
% Set XData and YData to specify coordinate limits
set(h, 'XData', [1 size(imageData,2)]);
set(h, 'YData', [1 size(imageData,1)]);
% Delete x-axis label
xlabel('');
% Delete y-axis label
ylabel('');
% Delete x-axis tick marks and labels
set(gca, 'XTick', []);
% Delete y-axis tick marks and labels
set(gca, 'YTick', []);


end
```

```
function repeatDisplayRandomPhoto
axes(playAxes);


% Get a list of all image files in the folder
imageFiles = dir('repeat/*.jpg');


% Choose a random image file
randomIndex = randi(numel(imageFiles));
randomImage = imageFiles(randomIndex);


% Read in the image file
imageData = imread(fullfile('repeat',
randomImage.name));


% Display image in plot
h = image(imageData);
% Set CDataMapping to 'scaled' to scale
data values to colormap
set(h, 'CDataMapping', 'scaled');
% Set XData and YData to specify
coordinate limits
set(h, 'XData', [1 size(imageData,2)]);
set(h, 'YData', [1 size(imageData,1)]);
% Delete x-axis label
xlabel('');
% Delete y-axis label
ylabel('');
% Delete x-axis tick marks and labels
set(gca, 'XTick', []);
% Delete y-axis tick marks and labels
set(gca, 'YTick', []);
end
```

```
function timeSliderCallback(source, ~)
% Get the current value of the time slider
value = get(source, 'Value');

% Stop the audio player
stop(player);

% Set the audio player to play from the new position to the end
play(player, floor(value)*fs);
playDisplayRandomPhoto;

audioState =1;
index=0;

while audioState == 1
set(songTextbox, 'String', selectionString);
set(timeSlider, 'Value', player.CurrentSample / fs);
elapsedTime = floor(player.CurrentSample / fs);
totalDuration = floor(player.TotalSamples / fs);
elapsedTimeStr = num2str(elapsedTime);
totalDurationStr = num2str(totalDuration);
set(elapsedTimeTextbox, 'String', ['Elapsed time : ' elapsedTimeStr '/'
totalDurationStr ' seconds .']);
if index == 20
visualize(player);
index=0;
else
index=index+1;
end
drawnow;
if player.CurrentSample == player.TotalSamples
stop(player);
audioState = 0;
stopDisplayRandomPhoto;
set(songTextbox, 'String', "Currently nothing playing ! ");
set(timeSlider, 'Value', 0);

end
end

end
```

```
function changeSpeed(source, ~)


selection = get(listbox, 'Value');
selectionString = items{selection};


[y, fs] = audioread(selectionString);
speed = source.String{source.Value};
speed = str2double(speed);
fs_mod = fs * speed;
player = audioplayer(y, fs_mod);
play(player);


set(pauseButton,'Enable','on');
set(timeSlider,'Enable','on');
set(timeSlider,'Enable','on');
set(stopButton,'Enable','on');


playDisplayRandomPhoto;


audioState =1;
index=0;
axes(playAxes);


while audioState == 1


set(songTextbox, 'String', selectionString);
set(timeSlider, 'Value', player.CurrentSample / fs_mod);


elapsedTime = floor(player.CurrentSample / fs_mod);
totalDuration = floor(player.TotalSamples / fs_mod);
elapsedTimeStr = num2str(elapsedTime);
totalDurationStr = num2str(totalDuration);
set(elapsedTimeTextbox, 'String', ['Elapsed time : ' elapsedTimeStr '/'
totalDurationStr ' seconds .']);
```

```
if index == 20
visualize(player);
index=0;
else
index=index+1;
end
drawnow;
if player.CurrentSample ==
player.TotalSamples
stop(player);
audioState = 0;
stopDisplayRandomPhoto;
set(songTextbox, 'String', "Currently
nothing playing ! ");
set(timeSlider, 'Value', 0);


end
end
end
function lowfreqSliderCallback(~,~)
value = get(lowFreqSlider,'Value');
value = round(value ,2 );

lowFreqGain = value ;
set(lowFreqTextbox,'String',lowFreqGain);
leqState = leqState+1 ;

if(leqState>=1&&heqState>=1)
set(equalizeButton,'Enable','on');

end
end
function highfreqSliderCallback(~,~)
value = get(highFreqSlider,'Value');
value = round(value ,2 );

highFreqGain = value ;
set(highFreqTextbox,'String',highFreqGain);
heqState = heqState+1 ;

if(leqState>=1&&heqState>=1)
set(equalizeButton,'Enable','on');

end
end
```

```
function equalizeButtonCallback(~, ~)


set(pauseButton,'Enable','on');
set(timeSlider,'Enable','on');
set(stopButton,'Enable','on');

selection = get(listbox, 'Value');
selectionString = items{selection};
[audio,fs] = audioread(selectionString);

audio(:,1) = audio(:,1) * lowFreqGain;
audio(:,2) = audio(:,2) * highFreqGain;

player = audioplayer(audio,fs);

plot(waveformaxes,audio);

xlabel('Time (s)',Color=[ 0.76 , 0.11, 0.76 ],FontSize=14);
ylabel('Amplitude',Color=[ 0.76 , 0.11, 0.76 ],FontSize=14);

title(waveformaxes,'Waveform of the audio signal',Color=[ 0.76 , 0.11,
0.76 ],FontSize=14);

play(player);

set(songTextbox,'String',['Currently playing: ', selectionString]);
set(playButton,'Enable','off');

axes(playAxes);

playDisplayRandomPhoto;

audioState =1;
index=0;
```

```
while audioState == 1
set(songTextbox, 'String', selectionString);
set(timeSlider, 'Value', player.CurrentSample / fs);
elapsedTime = floor(player.CurrentSample / fs);
totalDuration = floor(player.TotalSamples / fs);
elapsedTimeStr = num2str(elapsedTime);
totalDurationStr = num2str(totalDuration);
set(elapsedTimeTextbox, 'String', ['Elapsed time : ' elapsedTimeStr '/'
totalDurationStr ' seconds .']);


if index == 20
visualize(player);
index=0;
else
index=index+1;
end


drawnow;


if player.CurrentSample == player.TotalSamples
stop(player);
audioState = 0;
stopDisplayRandomPhoto;
set(songTextbox, 'String', "Currently nothing playing ! ");
set(timeSlider, 'Value', 0);


end
end
end


end
```

plotVisualizer.m

```
function h = plotVisualizer(A,x,y,ax,varargin)
%varargin{1} = radius, varargin{2} = nNodes, varargin{3} = switcher


% Plot the graph in the subplot
g = graph(A,'OmitSelfLoops');
h = plot(ax,g,'XData',x,'YData',y,'LineWidth',1);
h.NodeLabel = {};
h.EdgeColor = 'k';
h.MarkerSize = 3;


%Node color according to number of neighbours
colorSaver = jet;
% colorSaver(1:end,:,:) = colorSaver(end:-1:1,:,:); %upside down, better hot = more neighs
neighCounter = sum(A);
% step = round(64/max(neighCounter));
% ind = 1:step(1):63;
ind = round(linspace(1,63,length(neighCounter)));
MColor = colorSaver(ind(neighCounter+1),:);
h.NodeColor = MColor;


%Create Colormap
colormap(colorSaver);
str = sprintf('%d',length(neighCounter));
c.Label.String = 'Number of Neighbours';


axis equal
axis([ 0 1 0 1])
xticklabels('')
yticklabels('')
if length(varargin) == 3
%varargin{1} = radius, varargin{2} = nNodes, varargin{3} = switcher
if varargin{3} ~= 0
hold on
viscircles([x,y],repmat(varargin{1},varargin{2},1),'LineWidth',1,'Color', [0.5 0.5 0.5]);
hold off
end
end
end
```

main.m

```
clear ;
close all ;


audioState =-1;
selectionString =[];
player=[];
fs=44100;


lowFreqGain =0;
highFreqGain =0 ;
leqState =0;
heqState=0;


mp3_player(audioState,selectionString,player,
fs,lowFreqGain,highFreqGain,leqState,heqState)
```