**Table of contents**

**Contents Page no**

Acknowledgements ii

List of figures iii

Introduction 1 Design and Flow of routine 2

Format and Style of routine 5

Results and analysis 7

Discussion 9

Conclusion 10

References 11

Appendix 12

**List of Figures:-**

**Figure**   **Page No**

Figure 1 :- Notes…………………….……….…………………………..1

Figure 2:- Main function…………………….…………………………..3

Figure 3:- Different functions……………..…………………………....4

Figure 4:- Main Gui…………..……………………………………........7

Figure 5:- D# note…………………………………………………….…8

Figure 6:- Recording output…………………………..………………....8

**INTRODUCTION**

Music is as vital as any other activity in a person's daily life; for some, it relaxes, while for others, it heals and even helps someone become more focused and energetic about their work. Music is such a broad topic of study that it has spawned several subfields of study in philosophy and engineering. Music is described by the Oxford dictionary as "the art of blending vocal or instrumental sounds (or both) to create shape, harmony, and emotional expression." Which is correct, but this is beneficial for a general knowledge of music; yet, whether there is a definition of music in engineering or science remains a point of contention; however, the most renowned and thorough explanation of music is Music is a type of art and a cultural activity in which sound is ordered in time. Pitch (which determines melody and harmony), rhythm (and its linked ideas of tempo, metre, and articulation), and dynamics are all common aspects of music (loudness and softness).

Diagram

Description automatically generated

Figure 1:- Notes.

MATLAB, a high-performance programming language for technical computer applications, is utilized in the assignment. It's easy to use since problems and solutions are written in a mathematical format that most people are already familiar with. GUIDE, APP designer, and Simulink are all options for creating a graphical user interface in MATLAB. APP Designer is the tool we'll be utilizing for this project. MATLAB App Designer, a feature of the MATLAB software, may be used to design interactive apps. As a result of the drag-and-drop functionality and auto functions provided by App Designer, constructing a GUI is simplified for the user, who now just needs to implement the logic of their software rather than separate GUI code.

**DESIGN AND FLOW OF ROUTINES**

An algorithm was developed based on the requirements of the program; the algorithm was then implemented to flowchart. The flow chart shows the overall process of the MATLAB.

Figure 2 and figure 3 shows the flow chart of the overall process, figure 2 shows that the main function and properties and objects declaration. The objects in the properties are initialized to have an initial state of the values necessary for the operation. There are several callback functions included in the program for selection of different notes, for play button, for duration operation, for record operation, for clear button, for save button and for loudness selection. These functions all respond to the main function which is named as musical function. Variables required for the musical operation were initialized. sample time was initialized from initial value (0) to the time value selected by the user with a sample rate set as the periodic intervals. Sound Vector variable is the which stores the information related to the notes generated.

. Based on the variables assigned amplitude variable is calculated, fouriersignal is applied on the amplitude and then a fouriershift is applied on the signal. This operation was performed to plot different notes on the graph. Plot function was used to plot the generated amplitude and generated notes on the two different axes. Sound function was used to generate the sound based on the soundvector. A record variable which holds the data for the record button is used to check whether the current button will be concatenate with the previous notes, and based on this the another variable which store the notes is played once the user press the play button.

Diagram

Description automatically generated

Figure 2:- Main function.

Diagram

Description automatically generated

Figure 3:- Different functions.

**FORMAT AND STYLE OF ROUTINES.**

Following the completion of the flowchart, the procedure was carried out in MATLAB to its conclusion. In order to create the software, the Graphical User Interface (GUI) was built using the App designer, and a number of functional components were employed. Plotting the produced sound and the note was accomplished with the help of two Axes components. To include all of the notes, a list box was utilised, and a radio button was used to set the duration and loudness of the operation. Several buttons were utilised for switch or latching type operations such as recording, playing, pausing, and saving as well as for other functions as well.

properties (Access = public)

Property % Description

samplerate=1/20000;

Soundvar = 0;

TimeValue=0.4;

loudness = 2;

Amplitude=0;

end

The sample rate variable is a double type that is used to sample the frequency, which is defined as the number of audio samples carried per second. The sample rate is calculated using the formula (1/t), where t is the sampling period. In this program, the sample rate variable is initialized to 1/20000. The TimeValue variable is used for time sampling, which is like frequency sampling except that it is time related. The amplitude variable is used to create a graph of frequency vs amplitude for each note specified by the user.

sampletime = 0:app.samplerate:app.TimeValue;

soundVector = sin(2\*pi\*frequency\*sampletime);%sine wave selection for the sound

fs=500;%frequency shift

t = 0:1/fs:1-1/fs;%time plot values

app.Amplitude=(sin(2\*pi\*frequency\*t))\* app.loudness;

fouriersignal=fft(app.Amplitude);

fouriersignal=fftshift(fouriersignal);

frequency=fs/2\*linspace(-1,1,fs);

plot(app.UIAxes2,frequency,abs(fouriersignal))

plot(app.UIAxes,t,app.Amplitude)%plotting the selected

The SampleRate and TimeValue variables are only used to access the values of the primary samplerate and TimeValue variables, respectively, from the properties. The TimeSample variable is an array variable that is used for time sampling, and it makes use of the SampleRate and TimeValue variables to sample the time period. The frequency shift fs variable is used to create a default variable of 4000, which is used to determine the time interval in the graph. The variable t is then used to create an array of time samples using the frequency value, which is then used in the initialization of the Amplitude variable, and finally the plot function is used to create the graph.

sound(soundVector, 1/app.samplerate)

CheckRecord = app.RECORDButton.Value;

%check if the record is stopped or not

if CheckRecord == 1

%intitializing every variable

SOUNDCOMPOSE=app.Soundvar;

if SOUNDCOMPOSE == 0

SOUNDCOMPOSE=soundVector;

else

SOUNDCOMPOSE=cat(2,SOUNDCOMPOSE,soundVector);

end

app.Soundvar=SOUNDCOMPOSE;

end

This program makes use of the inbuilt sound () function to generate sound, which basically converts a matrix of signal data into sound. While sound() can take up to three parameters, only two are passed to the sound function in this program: the first is an array of signal data, and the second is the sample rate of the signal data.

For the record function, the program first checks whether the user has selected the record button using the if else case, and if the program receives a value of 1, it initializes the SoundVector array, which holds the current note selected by the user, to the SOUNDCOMPOSE variable, and then, using the cat function, which is an inbuilt function in MATLAB, it concatenates or merges the array passing 2 as the dimension size of the array and SOUNDCOMPOSE having the past notes.

**RESULTS AND ANALYSIS**

Graphical user interface

Description automatically generated

Figure 4:-Main GUI.

Figure 4 depicts the graphical user interface (GUI) for the constructed application, which includes all the essential parameters.

Graphical user interface

Description automatically generated

Figure 5:- D# note.

Figure 5 shows the app with displayed graph on both of the axis.

Graphical user interface

Description automatically generated

Figure 7: Recording.

Figure 7 shows the app when the record button is pushed, and different notes were

selected.

**DISCUSSION**

Other music composition applications have been used to inspire the design of the GUI, which has been made more user-friendly and realistic by incorporating elements from them. As a result, the MATLAB help library is considered to be one of the most significant and powerful tools available in the MATLAB environment. The application of several Matlab GUI components proved to be beneficial in a variety of scenarios involving the usage of distinct functions or methods. Initially, the function record proved to be a bit hard due to the fact that it was called from inside its callback, but this was eventually resolved by applying the logic within the play or main function rather than calling it from within its callback.

Some of the built-in functions of the MATLAB demonstrate the adaptability and relevance of the MATLAB in the engineering environment, such as plot and sound, for example. In this programme, plot is one of the most useful elements of MATLAB, and it has been utilised for two distinct graph implementations. When you first think about it, exporting a file to an audio format seems like a difficult operation to do. However, with the aid of the function audiowrite, it turns out to be a simple one-line procedure.

**Conclusion**

In conclusion, it can be stated that the assignment's requirements have been satisfied in terms of the specifications specified. The assignment makes use of App designer for the building of the graphical user interface, and a list box was developed for the selection of the notes. In order to allow the user to input the name of the recorded file, an editfield text component was added to the application. The axis was utilized to depict the specifics of the current notes, and the record button was built to record and play back the music that was made. The findings and analysis section, in addition, reveal that the designed system generated the values in the way that had been predicted. In order to make it easier for other users to comprehend, comments have been placed in the application as well.

**REFRENCES**

|  |  |  |
| --- | --- | --- |
| **Resources** | **In-Text Citation** | **End-Text Referencing (Reference List)** |
| **Article:**  **Four.**  **Author** | MATLAB App Designer is a feature that allows MATLAB code to be packaged into an interactive software. Harun, N. H., Hambali, H. A., Hassan, M. G., & Karim, K. N. (2017). Learn by example: MATLAB app design. | Harun, N. H., Hambali, H. A., Hassan, M. G., & Karim, K. N. (2017). Learn by example: MATLAB app design. |
| **Article:**  **Two.**  **Author** | The study of Fourier series .Petersen, M. R. (2004). Musical analysis and synthesis in Matlab. *College Mathematics Journal*, 396-401. | Petersen, M. R. (2004). Musical analysis and synthesis in Matlab. *College Mathematics Journal*, 396-401. |

**APPENDIX**

classdef app1 < matlab.apps.AppBase

% Properties that correspond to app components

properties (Access = public)

UIFigure matlab.ui.Figure

NOTEListBoxLabel matlab.ui.control.Label

NOTEListBox matlab.ui.control.ListBox

UIAxes matlab.ui.control.UIAxes

UIAxes2 matlab.ui.control.UIAxes

PLAYButton matlab.ui.control.Button

CLEARButton matlab.ui.control.Button

SAVEASButton matlab.ui.control.Button

FILENAMEEditFieldLabel matlab.ui.control.Label

FILENAMEEditField matlab.ui.control.EditField

DURATIONButtonGroup matlab.ui.container.ButtonGroup

SHORTButton matlab.ui.control.RadioButton

MEDIUMButton matlab.ui.control.RadioButton

LONGButton matlab.ui.control.RadioButton

LOUDNESSButtonGroup matlab.ui.container.ButtonGroup

SOFTButton matlab.ui.control.RadioButton

MEDIUMButton\_2 matlab.ui.control.RadioButton

LOUDButton matlab.ui.control.RadioButton

RECORDButton matlab.ui.control.StateButton

SONGMAKERLabel matlab.ui.control.Label

end

properties (Access = public)

Property % Description

samplerate=1/20000;

Soundvar = 0;

TimeValue=0.4;

loudness = 2;

Amplitude=0;

end

methods (Access = public)

function musical(app, frequency)

sampletime = 0:app.samplerate:app.TimeValue;

soundVector = sin(2\*pi\*frequency\*sampletime);%sine wave selection for the sound

fs=500;%frequency shift

t = 0:1/fs:1-1/fs;%time plot values

app.Amplitude=(sin(2\*pi\*frequency\*t))\* app.loudness;

fouriersignal=fft(app.Amplitude);

fouriersignal=fftshift(fouriersignal);

frequency=fs/2\*linspace(-1,1,fs);

plot(app.UIAxes2,frequency,abs(fouriersignal))

plot(app.UIAxes,t,app.Amplitude)%plotting the selected sound.

sound(soundVector, 1/app.samplerate)

CheckRecord = app.RECORDButton.Value;

%check if the record is stopped or not

if CheckRecord == 1

%intitializing every variable

SOUNDCOMPOSE=app.Soundvar;

if SOUNDCOMPOSE == 0

SOUNDCOMPOSE=soundVector;

else

SOUNDCOMPOSE=cat(2,SOUNDCOMPOSE,soundVector);

end

app.Soundvar=SOUNDCOMPOSE;

end

end

end

% Callbacks that handle component events

methods (Access = private)

% Value changed function: NOTEListBox

function NOTEListBoxValueChanged(app, event)

value = app.NOTEListBox.Value;

switch value

case 'C'

frequency = 130.82

case 'C#'

frequency = 138.59

case 'D'

frequency = 146.83

case 'D#'

frequency = 155.56

case 'E'

frequency = 164.81

case 'F'

frequency = 174.61

case 'F#'

frequency = 185

case 'G'

frequency = 196

case 'G#'

frequency = 207.65

case 'A'

frequency = 220

case 'A#'

frequency = 233.08

case 'B'

frequency = 246.94

end

musical(app, frequency)

end

% Button pushed function: PLAYButton

function PLAYButtonPushed(app, event)

soundvar=app.Soundvar;

samplerateplay=app.samplerate;

sound(soundvar,1/samplerateplay);

end

% Selection changed function: DURATIONButtonGroup

function DURATIONButtonGroupSelectionChanged(app, event)

selectedButton = app.DURATIONButtonGroup.SelectedObject.Text;

app.TimeValue = 0.4;

switch(selectedButton)

case 'SHORT'

app.TimeValue=app.TimeValue;

case 'MEDIUM'

app.TimeValue=app.TimeValue\*2;

case 'LONG'

app.TimeValue=app.TimeValue\*4;

end

end

% Value changed function: RECORDButton

function RECORDButtonValueChanged(app, event)

value = app.RECORDButton.Value;

end

% Button pushed function: CLEARButton

function CLEARButtonPushed(app, event)

app.Soundvar = 0

end

% Button pushed function: SAVEASButton

function SAVEASButtonPushed(app, event)

if app.Soundvar==0

disp("THERE IS NO NOTE RECORDED YET OR MUST HAVE BEEN RESET")

else

AUDIOFILE=app.SONGNAMEFIELD.Value;

EXT='.wav';

FILENAME=strcat(AUDIOFILE,EXT);

DIR='G:\ASSIHNMENT 4th sem\MATLAB\ori\';

FINALNAME=strcat(DIR,FILENAME);

audiowrite(FINALNAME,app.Soundvar,1/app.samplerate,'BitsPerSample',32);

end

end

% Selection changed function: LOUDNESSButtonGroup

function LOUDNESSButtonGroupSelectionChanged(app, event)

selectedButton = app.LOUDNESSButtonGroup.SelectedObject;

switch(selectedButton)

case 'SOFT'

app.loudness = app.loudness / 4;

case 'MEDIUM'

app.loudness = app.loudness;

case 'LONG'

app.loudness = app.TimeValue\*4;

end

end

end

% Component initialization

methods (Access = private)

% Create UIFigure and components

function createComponents(app)

% Create UIFigure and hide until all components are created

app.UIFigure = uifigure('Visible', 'off');

app.UIFigure.Position = [100 100 753 545];

app.UIFigure.Name = 'MATLAB App';

% Create NOTEListBoxLabel

app.NOTEListBoxLabel = uilabel(app.UIFigure);

app.NOTEListBoxLabel.HorizontalAlignment = 'right';

app.NOTEListBoxLabel.VerticalAlignment = 'top';

app.NOTEListBoxLabel.Position = [33 510 42 22];

app.NOTEListBoxLabel.Text = 'NOTE ';

% Create NOTEListBox

app.NOTEListBox = uilistbox(app.UIFigure);

app.NOTEListBox.Items = {'C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B'};

app.NOTEListBox.ValueChangedFcn = createCallbackFcn(app, @NOTEListBoxValueChanged, true);

app.NOTEListBox.Position = [11 276 109 227];

app.NOTEListBox.Value = 'C';

% Create UIAxes

app.UIAxes = uiaxes(app.UIFigure);

title(app.UIAxes, 'FREQUENCY')

xlabel(app.UIAxes, 'frequency')

ylabel(app.UIAxes, 'Amplitude')

app.UIAxes.Position = [1 10 381 230];

% Create UIAxes2

app.UIAxes2 = uiaxes(app.UIFigure);

title(app.UIAxes2, 'NOTES')

xlabel(app.UIAxes2, 'Sample')

ylabel(app.UIAxes2, 'notes')

app.UIAxes2.Position = [381 1 373 239];

% Create PLAYButton

app.PLAYButton = uibutton(app.UIFigure, 'push');

app.PLAYButton.ButtonPushedFcn = createCallbackFcn(app, @PLAYButtonPushed, true);

app.PLAYButton.Position = [282 292 100 30];

app.PLAYButton.Text = 'PLAY';

% Create CLEARButton

app.CLEARButton = uibutton(app.UIFigure, 'push');

app.CLEARButton.ButtonPushedFcn = createCallbackFcn(app, @CLEARButtonPushed, true);

app.CLEARButton.Position = [405 292 100 30];

app.CLEARButton.Text = 'CLEAR';

% Create SAVEASButton

app.SAVEASButton = uibutton(app.UIFigure, 'push');

app.SAVEASButton.ButtonPushedFcn = createCallbackFcn(app, @SAVEASButtonPushed, true);

app.SAVEASButton.Position = [640 292 76 30];

app.SAVEASButton.Text = 'SAVE AS';

% Create FILENAMEEditFieldLabel

app.FILENAMEEditFieldLabel = uilabel(app.UIFigure);

app.FILENAMEEditFieldLabel.HorizontalAlignment = 'right';

app.FILENAMEEditFieldLabel.Position = [556 344 69 22];

app.FILENAMEEditFieldLabel.Text = 'FILE NAME';

% Create FILENAMEEditField

app.FILENAMEEditField = uieditfield(app.UIFigure, 'text');

app.FILENAMEEditField.Position = [640 344 100 22];

% Create DURATIONButtonGroup

app.DURATIONButtonGroup = uibuttongroup(app.UIFigure);

app.DURATIONButtonGroup.SelectionChangedFcn = createCallbackFcn(app, @DURATIONButtonGroupSelectionChanged, true);

app.DURATIONButtonGroup.Title = 'DURATION';

app.DURATIONButtonGroup.Position = [154 365 123 106];

% Create SHORTButton

app.SHORTButton = uiradiobutton(app.DURATIONButtonGroup);

app.SHORTButton.Text = 'SHORT';

app.SHORTButton.Position = [11 60 63 22];

app.SHORTButton.Value = true;

% Create MEDIUMButton

app.MEDIUMButton = uiradiobutton(app.DURATIONButtonGroup);

app.MEDIUMButton.Text = 'MEDIUM';

app.MEDIUMButton.Position = [11 38 71 22];

% Create LONGButton

app.LONGButton = uiradiobutton(app.DURATIONButtonGroup);

app.LONGButton.Text = 'LONG';

app.LONGButton.Position = [11 16 65 22];

% Create LOUDNESSButtonGroup

app.LOUDNESSButtonGroup = uibuttongroup(app.UIFigure);

app.LOUDNESSButtonGroup.SelectionChangedFcn = createCallbackFcn(app, @LOUDNESSButtonGroupSelectionChanged, true);

app.LOUDNESSButtonGroup.Title = 'LOUDNESS';

app.LOUDNESSButtonGroup.Position = [381 365 123 106];

% Create SOFTButton

app.SOFTButton = uiradiobutton(app.LOUDNESSButtonGroup);

app.SOFTButton.Text = 'SOFT';

app.SOFTButton.Position = [11 60 58 22];

% Create MEDIUMButton\_2

app.MEDIUMButton\_2 = uiradiobutton(app.LOUDNESSButtonGroup);

app.MEDIUMButton\_2.Text = 'MEDIUM';

app.MEDIUMButton\_2.Position = [11 38 71 22];

app.MEDIUMButton\_2.Value = true;

% Create LOUDButton

app.LOUDButton = uiradiobutton(app.LOUDNESSButtonGroup);

app.LOUDButton.Text = 'LOUD';

app.LOUDButton.Position = [11 16 65 22];

% Create RECORDButton

app.RECORDButton = uibutton(app.UIFigure, 'state');

app.RECORDButton.ValueChangedFcn = createCallbackFcn(app, @RECORDButtonValueChanged, true);

app.RECORDButton.Text = 'RECORD';

app.RECORDButton.Position = [154 292 100 30];

% Create SONGMAKERLabel

app.SONGMAKERLabel = uilabel(app.UIFigure);

app.SONGMAKERLabel.HorizontalAlignment = 'center';

app.SONGMAKERLabel.FontSize = 40;

app.SONGMAKERLabel.FontWeight = 'bold';

app.SONGMAKERLabel.Position = [203 482 422 50];

app.SONGMAKERLabel.Text = 'SONG MAKER';

% Show the figure after all components are created

app.UIFigure.Visible = 'on';

end

end

% App creation and deletion

methods (Access = public)

% Construct app

function app = app1

% Create UIFigure and components

createComponents(app)

% Register the app with App Designer

registerApp(app, app.UIFigure)

if nargout == 0

clear app

end

end

% Code that executes before app deletion

function delete(app)

% Delete UIFigure when app is deleted

delete(app.UIFigure)

end

end

end