GUI LITE - A REDUCED COMPLEXITY GRAPHICAL USER INTERFACE DEVELOPMENT TOOLBOX IN MATLAB (WITH APPLICATIONS TO DIGITAL SPEECH PROCESSING PROBLEMS)

BY REEMY MARIA D'SOUZA

A thesis submitted to the

Graduate School-New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science
Graduate Program in Electrical and Computer Engineering
written under the direction of

Professor Lawrence R. Rabiner
And approved by

_____

_____

_____

New Brunswick, New Jersey

January, 2011

ABSTRACT OF THE THESIS

GUI Lite - a Reduced Complexity Graphical User Interface Development Toolbox in MATLAB (with Applications to Digital Speech Processing Problems)

by Reemy Maria D'Souza

Thesis director

Dr. Lawrence R. Rabiner

Graphical User Interfaces (GUIs) are used to view and study the capabilities and limitations of a range of speech processing applications. They are invaluable teaching and algorithm implementation aids. Using a GUI to explore the capabilities of a given application greatly increases the utility of the application, particularly in the area of digital speech processing.

Currently there exists a powerful GUI design toolbox, called the GUIDE (Graphical User Interface Development Environment), included with MATLAB. Learning how to use the GUIDE effectively is complicated and time-consuming. Our basic premise about the GUIDE is that a small and manageable subset of the GUIDE's capability could provide sufficient flexibility to implement most speech processing problems of interest.

With this driving principle, we have designed and implemented the GUI Lite Version 1 and Version 2 which enable the user to easily design and create GUIs in MATLAB. GUI Lite Version 1 is a single-pass design tool in which the GUI layout and callback functions (i.e., code associated with the various GUI elements like graphical displays and buttons) are integrated into a single stage solution. The GUI Lite Version 1 User Manual explains how to write code to control and manipulate the various GUI components used in a given implementation of a speech processing algorithm.

GUI Lite Version 2 is a two-pass design tool in which the GUI layout is implemented in the first stage, and the selected GUI element callback functions are implemented in the second stage. GUI Lite Version 2 automates and separates the design and layout of the GUI from the writing of the callback code that controls the various GUI elements. This two stage GUI design and creation tool simplifies the process of creating viable GUIs and improves the user experience significantly. GUI Lite Versions 1 and 2 have undergone a series of user trials to develop GUIs for a range of speech processing algorithms. The trial results indicate that the two GUI Lite tools succeed in making the creation process of GUIs for speech processing algorithms a great deal simpler and more intuitive than MATLAB's GUIDE tool.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Prof. Lawrence Rabiner for his constant guidance and motivation. Inspite of his busy schedule, Prof. Rabiner was always available to discuss my thesis, edit my manuscript and offer encouragement. This thesis would not have been possible without him.

I would also like to thank my parents, and my extended family, Savio, Remya, Cibin, Vanchi, Preeti, Kashyap and Narayanan for their support and motivation while writing my thesis.

Table of Contents

Chapter 1 Introduction

1.1 Motivation

User Interfaces (UIs) are used by researchers, teachers and students alike to demonstrate and study various speech processing algorithms and applications. GUIs [1] [2] [3] [4] [5] make it fast and easy for users to change parameters, optimize performance and get fast results from speech processing applications without delving into the application's code.

A fairly powerful UI toolkit called the GUIDE [6] was created by MathWorks and developed in MATLAB. UIs are implemented using the GUIDE which is an acronym for the MATLAB Graphical User Interface Development Environment. The GUIDE toolbox provides the user with extensive UI development facilities. In order to create a button, a user launches the GUIDE toolbox at the MATLAB command prompt and then selects the style of button that he or she would like to create from a menu bar that is provided after the toolbox is launched. Double clicking the created button will provide the user with approximately 37 properties that the user can either set or leave at their default values. The GUIDE toolbox enables the user to create an extremely sophisticated GUI with extensive UI development features. However, in order to use the GUIDE, the user needs to be able to learn a new environment and develop a level of expertise in it to understand how to use it effectively. GUIDE is complicated and takes a long time to get working, even for small programs. The GUIDE is also not very intuitive and often it takes longer to create the user interface than it does to write the speech processing application itself.

To ameliorate this issue, we have designed and implemented a new toolbox, called the GUI Lite toolbox, which is a lighter, more intuitive, alternative to the GUIDE toolbox. It is based on the driving principle that a small and manageable subset of GUIDE's capability will provide sufficient flexibility to create GUIs for almost all speech processing applications. Though GUI Lite does not provide a plethora of UI development capabilities to the user like the GUIDE, it is more than sufficient to design and implement UI environments that are of low-to-moderate complexity. GUI Lite significantly reduces the time and effort to design and create GUIs for speech processing applications.

1.2 Description of GUI Lite

GUI Lite is a graphical user interface design tool set implemented in MATLAB. It provides an intuitive and relatively simple UI development environment which aims to improve a user's ability and experience while creating UIs. GUI Lite is intended to help the user focus on the implementation of the application rather than the implementation of the UI for the application. The GUI Lite toolbox is intended to be mainly used to create graphical user interfaces for speech processing applications. GUI Lite provides a highly intuitive and straightforward method to create GUIs. The target users are researchers, faculty and students.

GUI Lite attempts to create a trade-off of complexity versus features. It offers reduced complexity while maintaining most of the essential features required by a GUI design toolbox. The MATLAB UI design toolkit, GUIDE, offers far more features than GUI Lite but using the toolkit is quite complicated and not very intuitive.

GUI Lite, the low complexity toolbox for easy UI development was implemented in two different ways which we call Version 1 and Version 2. Version 1 was a single stage design in which GUI features and callback code implementation for the GUI components were integrated into a single package. Version 1 was tested on a range of signal processing problems until it became clear that a 2-stage solution, in which the design of the UI and the code implementation for each of the UI elements was separated into individual code components, was the preferred solution.

Subsequent GUI development showed that the 2-stage solution, dramatically simplified the problem of how to create viable user interfaces. A brief user manual (included in Appendix A for GUI Lite-Version 1 and Appendix B for GUI Lite-Version 2) and a package of files that must be saved onto the user's computer for Version 2 are included in the appendices of this thesis.

1.2.1 Components that can be created using GUI Lite.

A GUI created using the GUI Lite toolkit consists of one or more of the following GUI objects.

1. Panel: A panel is a gray rectangular outline that is used to contain buttons, plots, title boxes or a combination of all three. Panels are also used to group UI objects (buttons or plots) performing similar functions together. Panels improve the layout of the GUI by providing a well-defined, more organized way of arranging the elements of the GUI.
2. Graphic Panel: A graphic panel is a plot window in which plots are drawn. It is different from the panel above. It is a white box and is sometimes enclosed by a 'panel'.

3. Title Box: A title box is a gray rectangular box which is used to add a title for a panel, a graphic panel or a group of buttons. The text to be displayed in the title box is entered while writing callbacks (i.e., code implementations of the desired buttons) for the various GUI objects.
4. Button: A button is an object that is used to input a desired parameter to the GUI, perform a desired function when clicked, or provide options (e.g.: a drop down menu). Figure 1.1 displays a dummy GUI with GUI objects including a panel, a graphic panel, a title box and a button created and labeled. Note that the rectangular outline for the title box is not clearly distinguishable from the background since they are both gray in color.

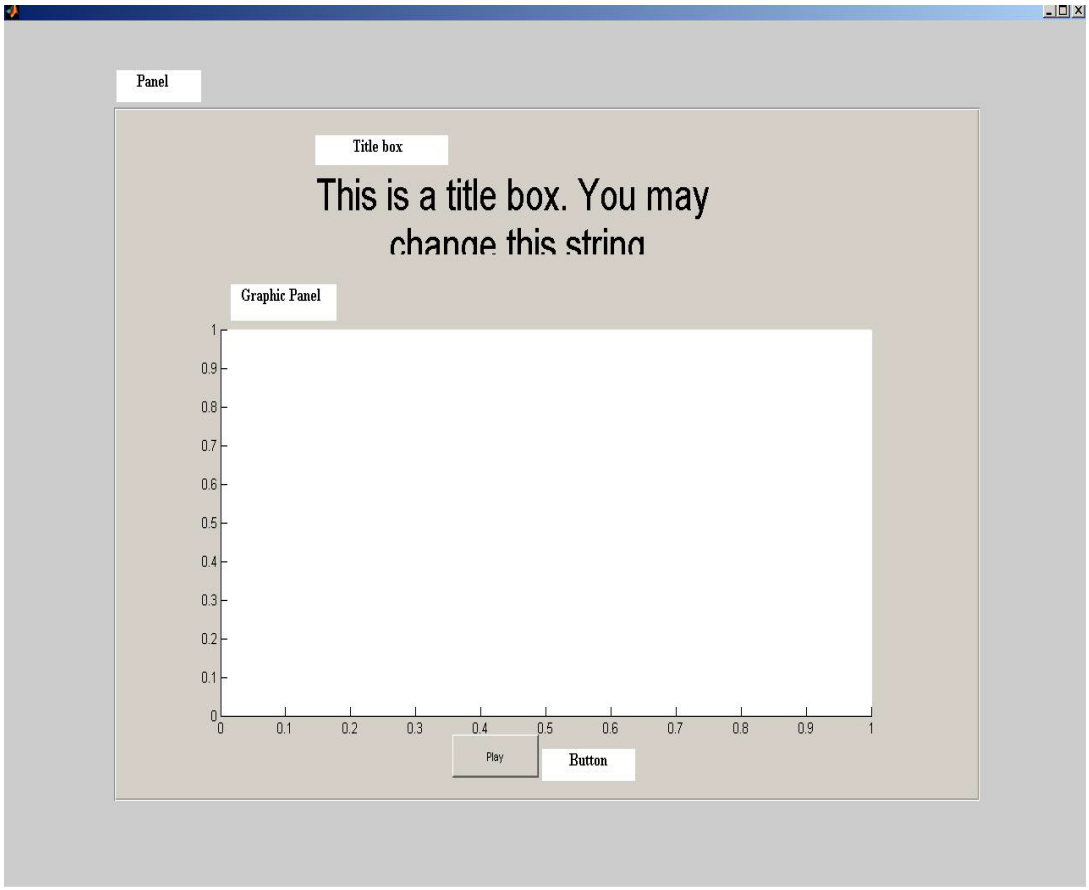

Figure 1.1 A dummy GUI with GUI objects including a panel, a graphic panel, a title box and one button.

The types of buttons that can be created using GUI Lite are as follows:

A) Pushbutton: This button performs a certain function when clicked i.e., it executes code in its callback function when clicked via the mouse. This is the default 'type' of any button created using GUI Lite.

B) Edit box: This kind of button is used to provide input to a GUI in MATLAB (e.g., the value of a variable). To create an edit box, the user must specify the type of button be created as an 'edit' button. The value in

the edit box can be changed by moving the mouse to the edit box and editing the value in the box.

C) Text button: This is a gray box that contains text. It can be used to hold a common title for a group of buttons or the label for a single button. The text in this box cannot be edited. Also, the text for this button is entered while writing the code for the callback functions.

D) Popupmenu button: The popupmenu button is a pull down menu which contains a list of options for user selection (e.g., possible speech files within a directory for analysis). The user must enter the type of button as a 'popupmenu' button while creating the button.

E) Slider button: This button is a horizontal slider button. It can be used for tasks such as volume control. The user must enter the type of button as a 'slider' button while creating this type of button.

5. 'Callback' functions: The 'uicontrol' object/function is used to create buttons for a GUI. The buttons created by the 'uicontrol' function do not perform the desired functions until the callback code for the buttons has been written. The 'callback' function is the backend code of the button that actually performs the function the button of the button. The 'callback' code is the application's code, and is written by the user.

1.2.2 Essential MATLAB functions for GUI development.

MATLAB provides its users with a multitude of functions which they use to write their own applications. Two of the most important functions provided by MATLAB for GUI development are the 'uipanel' and the 'uicontrol' functions. They are explained below.

- ‘uipanel’ function: The ‘uipanel’ [7] function/object is used to create user interface objects such as panels which are used to outline GUI objects such as graphic panels and buttons. The ‘uipanel’ object has attributes like ‘Title’, ‘BackgroundColor’, ‘Position’, etc., which must be specified by the user while creating the ‘panel’. Entering the values for the ‘position’ attribute of the ‘uipanel’ object is tedious, time consuming and difficult to get right. To help demonstrate how tedious entering the ‘position’ attribute of the uipanel’ object can be, a dummy GUI, named the ‘Simple Test GUI’ which contains four panels created and is displayed in Figure 1.2.

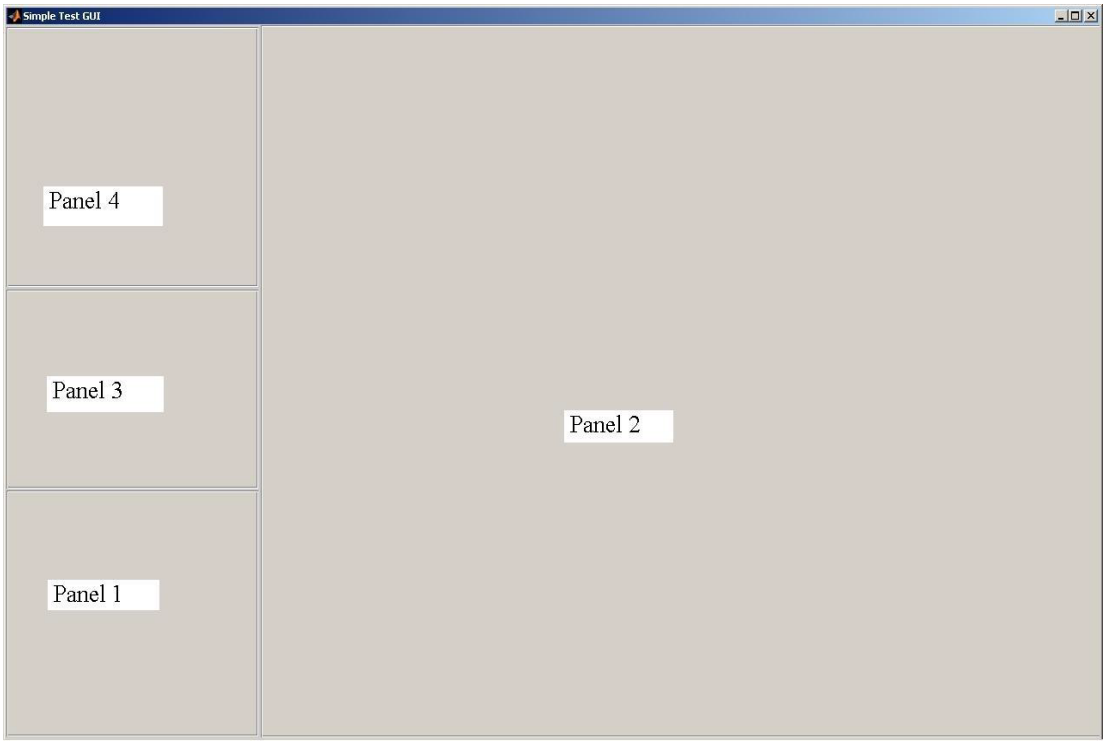

Figure 1.2 The ‘Simple Test GUI’ contains four panels. The panels are labeled as Panel 1, Panel 2, Panel 3 and Panel 4.

Figure 1.3 displays the 'Simple Test GUI' with the co-ordinates and dimensions of the four 'uipanel' objects clearly marked on the GUI.

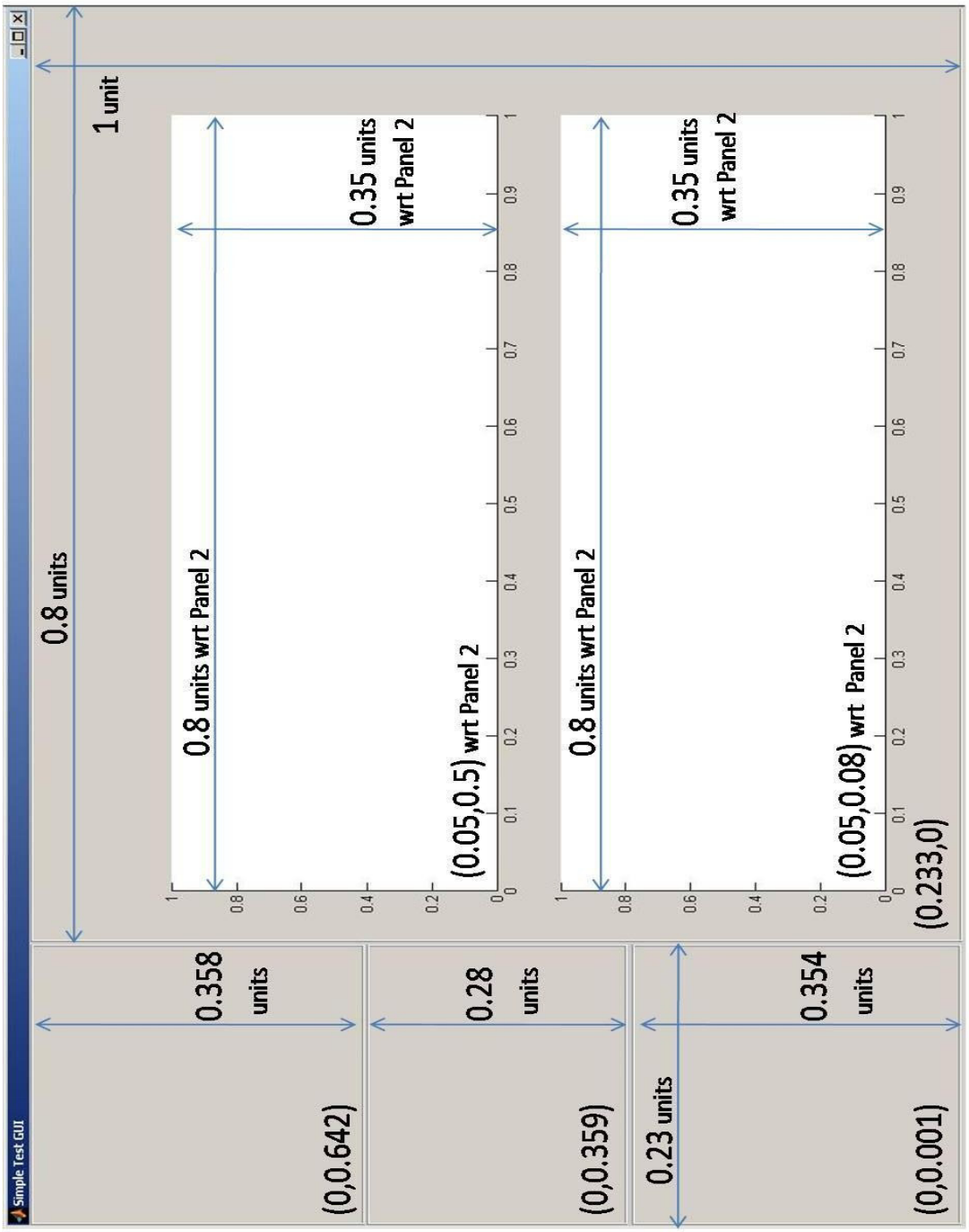

Figure 1.3 The 'Simple Test GUI' with the co-ordinates and the dimensions of the panels and graphic panels clearly marked on it.

- ‘uicontrol’ object: The ‘uicontrol’ [8] object is used to create user interface objects such as buttons. The ‘uicontrol’ object is used to create different types of buttons such as ‘pushbutton’, ‘edit’, ‘popupmenu’, ‘text’ and ‘slider’ buttons. Similar to the ‘uipanel’ object, each button has attributes such as ‘BackgroundColor’, ‘String’, ‘Type’, ‘Position’, etc. Entering the ‘position’ attribute of a ‘uicontrol’ object is very tedious since getting the positions of the buttons to look just right on the GUI window requires meticulously entering the position attribute of the ‘uicontrol’ object, sometimes with accuracy even up to 3 decimal places.

  Figure 1.4 displays the ‘Simple Test GUI’ with the co-ordinates and dimensions of the buttons clearly marked on it.

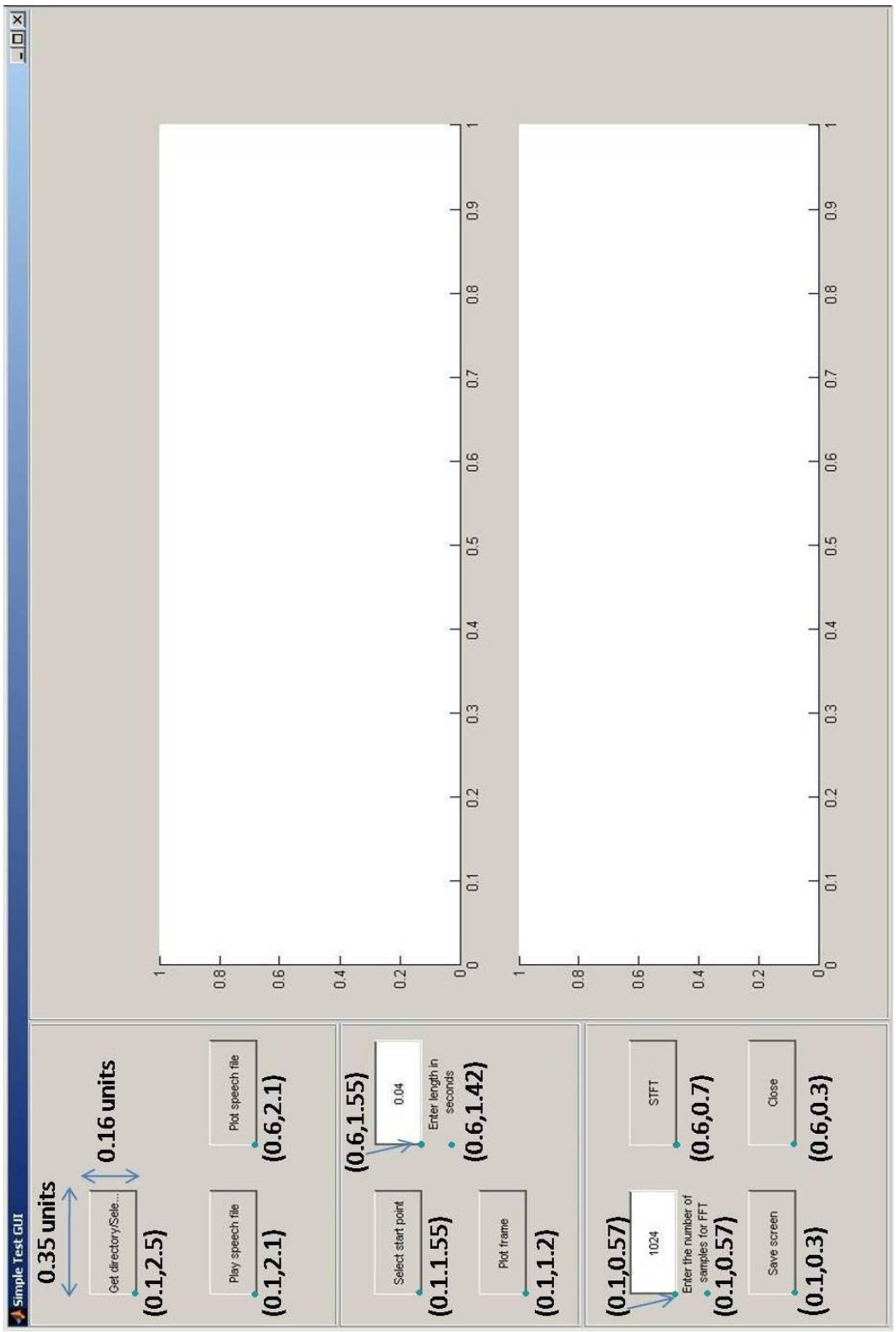

Figure 1.4 The 'Simple Test GUI' with the co-ordinates and dimensions of the buttons marked on it.

From Figures 1.3 and 1.4, it is clear that properly positioning GUI objects like panels and buttons is a tiresome and painstaking task.

The ‘position’ attribute of a ‘uipanel’ or ‘uicontrol’ object needs to be entered in the following format:

[x y length width] where

‘x’ and ‘y’ are the x and y co-ordinates respectively of the bottom left corner of the created panel with respect to the bottom left corner i.e., origin [0, 0] of the GUI window. The ‘length’ and ‘width’ variables are the horizontal length and vertical width of the GUI object whose ‘position’ attribute is being entered. For the GUI object ‘panel1’ displayed in Figures 1.2 and 1.3, the position attribute is [0 0.001 0.23 0.354].

Similarly, the ‘position’ attribute for the ‘Get directory/Select file’ button which is displayed in the top left corner of the ‘Simple Test GUI’ in Figure 1.4 is [0.1 2.5 0.35 0.16].

Chapter 2 Implementation of a GUI using GUI Lite -Version 1

2.1 GUI Lite - Version 1

GUI Lite - Version 1 is a one-pass design tool in which GUI features and callback programs (i.e., code associated with the various panels, graphic panels, title boxes and buttons) are integrated into a single stage solution. GUI Lite - Version 1 (for which a detailed user manual is given in Appendix A) explains how to understand and write code to create GUI objects such as panels, graphic panels, title boxes and buttons. The GUI Lite – Version 1 User's Guide is a self contained user manual that includes detailed instructions, code snippets and clearly marked screenshots from which the user can create a viable user interface. The user's guide for GUI Lite – Version 1 provides an example showing how to create a GUI in MATLAB. The user's guide begins with the layout of the GUI which is created using combinations of 'uicontrol' and 'uipanel' functions. The user's guide explains how to write callbacks for the created GUI objects. GUI Lite - Version 1 was developed with the intention of assisting and teaching the user how to create his or her own GUIs with ease. Using the detailed user's guide, a user should be able to familiarize himself with the various components of a GUI and understand how GUI objects can be manipulated to create much more complicated GUIs than the one given in the user guide.

2.2 Implementation of four Baseline Programs Using GUI Lite Version 1

In this section we explain how to use the GUI Lite – Version 1 to design UI's for four speech processing exercises. The four exercises are ordered to demonstrate increasing complexity GUI solutions. The reader should consult the User's Guide for GUI Lite –

Version 1 (Appendix A) for additional input and explanations of the Version 1 steps in setting up the GUI for each of these four examples. The following four exercises show simple examples of how to build GUIs using the GUI Lite – Version 1 Toolkit. The programs consist of the following signal processing exercises.

- Program 1 - Hello World program.
- Program 2 - Display the waveform of a designated speech file.
- Program 3 - Browse a directory, load the selected speech file, play it and display the waveform.
- Program 4 - Load an existing speech file or record a new speech file. Play the selected speech array and save the recorded speech array in a designated file.

The MATLAB code for all four examples has been included in Appendix C**.**

2.2.1 Program 1 - Hello World program

The GUI Lite - Version 1 toolbox can easily be used to create a GUI which simply displays the text 'Hello World' when a button is pressed. The first step in creating a GUI is to determine what the GUI should look like. To do this the user must draw a sketch of the GUI on paper, as shown in Figure 2.1 for the Hello World program.

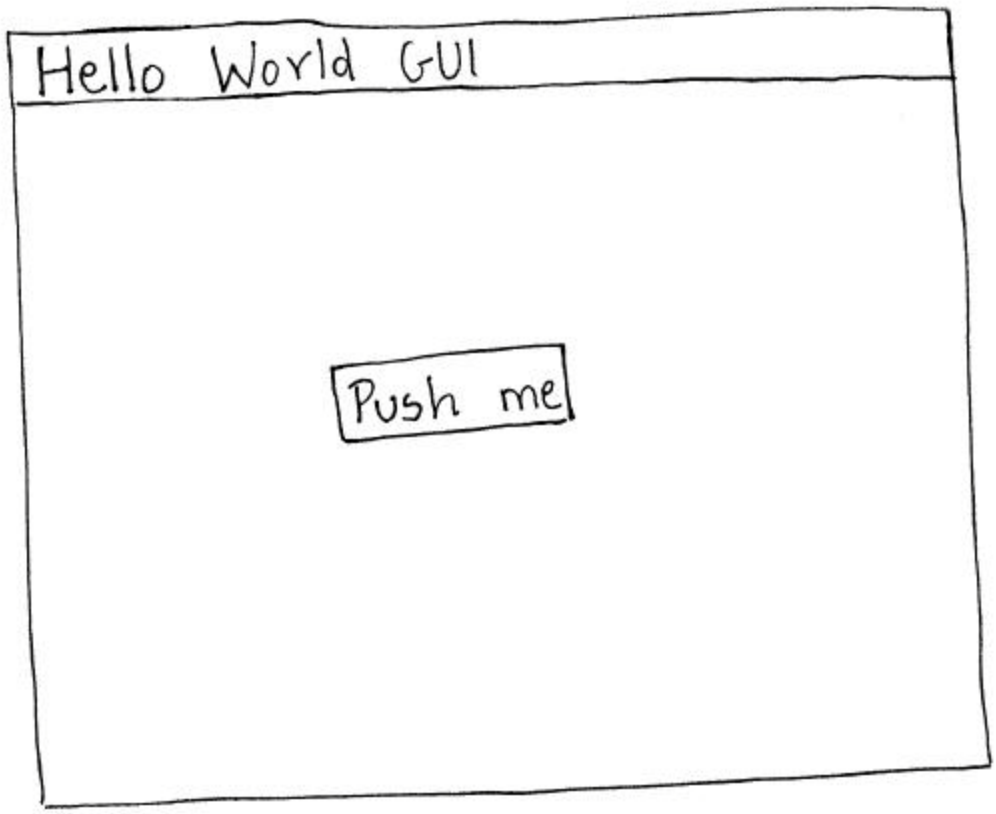

Figure 2.1 A sketch of the tentative layout for the 'Hello World GUI'.

For this simple GUI the user will require a GUI window i.e., a window/screen where the desired buttons will be laid out and a button which, when clicked, will display the message 'Hello World'.

The first step in the in the 'Hello World GUI' development process is to define a function called 'helloWorld'. The 'helloWorld' function represents the GUI being created. The 'helloWorld' function is created using the following code snippet:

```
function helloWorld
%embedded code for the GUI application
end
```

The user should first enter the commands ‘clc; clear all;’ within the function that has just been created (i.e., helloworld). The ‘clc;’ and ‘clear all;’ commands clear all the variables from the workspace before running the GUI. The ‘Hello World GUI’ contains only one button and hence does not require any panels,. The following code snippet should next be inserted within the function ‘helloWorld’ created by the user.

```
clc;clear all;
f = figure('Visible','on',...
            'Units','normalized',...
            'Position',[0,0,1,1],...
            'MenuBar','none',...
            'NumberTitle','off');
% Assign the GUI a name to appear in the window title.
set(f,'Name','Hello World');
```

The above code creates a GUI window named ‘Hello World’ whose units are normalized to the [0 1] range and whose position is set to full screen. The GUI is positioned at coordinate (0, 0) (the left bottom corner) of the user’s computer screen and has length and width of one unit. The user can save and run the ‘helloWorld.m’ file to see the screen shown in Figure 2.2.

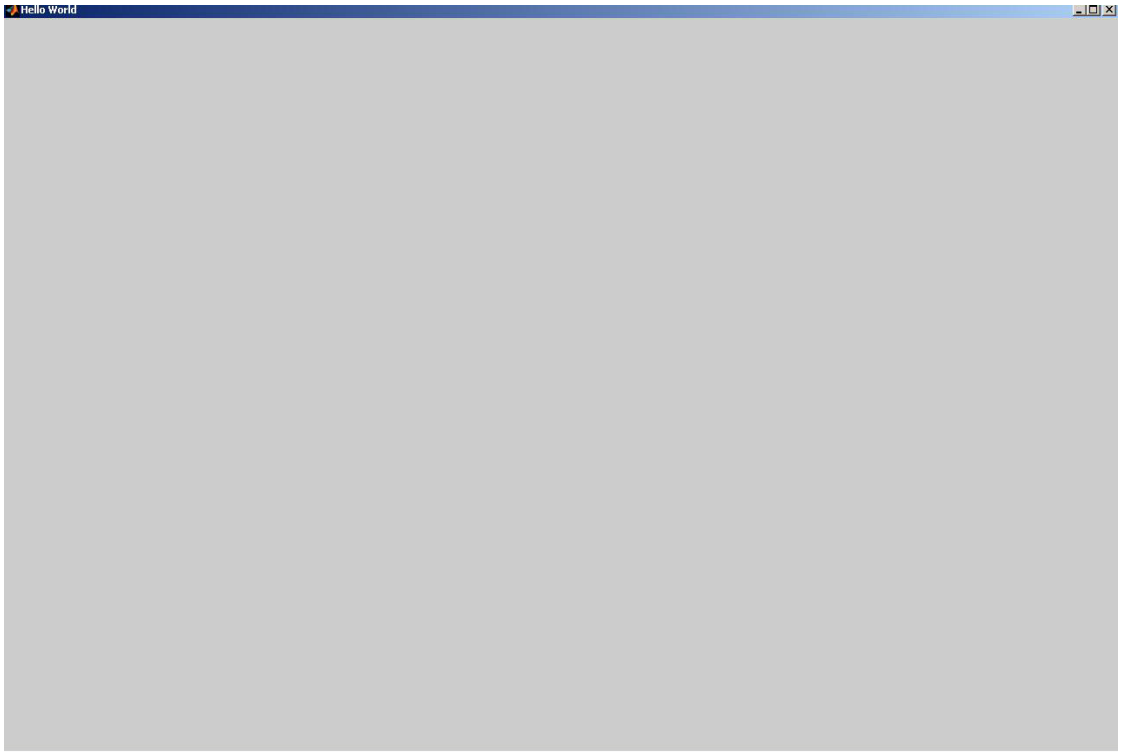

Figure 2.2 The 'Hello World' 'figure' window created using the 'figure' command. The dark blue line at the top of the gray window contains the name of the window.

The next step in the 'Hello World GUI' development process is to create the 'Push me' button using the 'uicontrol' function/object. Since the default style of a 'uicontrol' (user interface object) object is 'pushbutton', if the style attribute of the 'uicontrol' function is not specified, it is assumed to be a pushbutton. The user can create a 'pushbutton' that says 'Push me' using the following code (as given in the user's guide in Appendix A):

```
%BUTTON
% Push me button
pushMebutton=uicontrol('Parent',f,...
            'Units','Normalized',...
            'Position',[0.1 0.3 0.2 0.1],...
            'String', 'Push Me',...
            'Callback',@pushMeCallback);
```

While using the code from the user's guide, the user should remember to change the 'parent' attribute of the 'uicontrol' function to 'f', where 'f' is the default name of the 'figure' window created by the user and within it lies the 'Push me' button. Hence 'f' is known as the 'parent' of the 'Push me' button. Also the 'string', 'position' and 'callback' attributes should be assigned as shown in the above code snippet. The callback function 'pushMeCallback' is called and executed when the 'Push me' button is clicked. Figure 2.3 displays how the GUI window will look to the user after the 'Push me' button has been created.

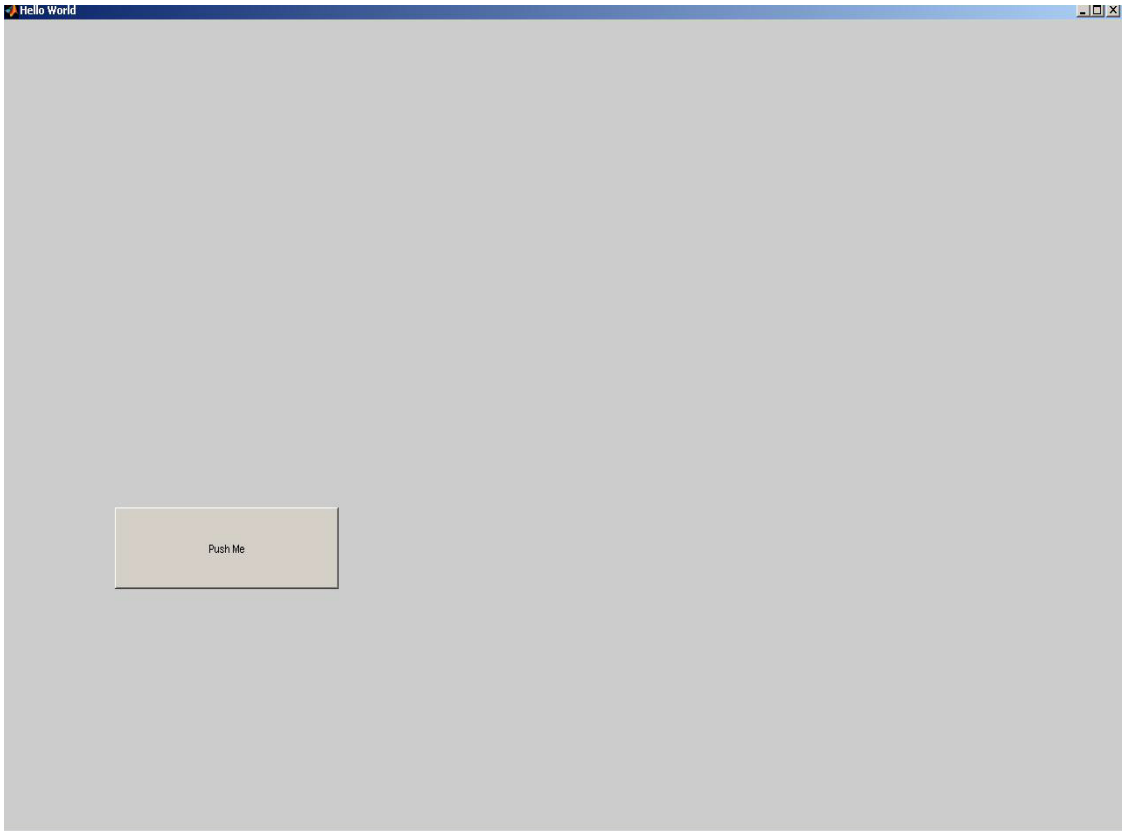

Figure 2.3 The 'Hello World' GUI containing the 'Push me' 'pushbutton' button. This button is not ready to be clicked since the callback for the button has not yet been written. Clicking the button will display an error alert on the MATLAB command window.

The next step is to write a callback for the ‘Push Me’ button. The callback is the code that actually performs the function that the ‘Push Me’ button is supposed to do when the ‘Push me’ button is clicked. The button code that was written using the ‘uicontrol’ function only creates the button. If the user were to push the button at this stage he (or she) would get an error alert on the MATLAB command window as the button does not perform any function. The code for the callback for the ‘Push me’ button is as follows:

```
%callback for the push me button
      function pushMeCallback(h,eventdata)
            msgbox('Hello World!','modal')
      end
```

In the above callback function, the argument ‘h’ represents the handle to the object ‘pushMe’ and the argument ‘eventdata’ is reserved for use in future versions of MATLAB. The ‘h’ and ‘eventdata’ arguments are always passed while writing callback functions. The ‘msgbox’ function is used to create and display a message box that displays the text sequence ‘Hello World!’. The ‘modal’ attribute of the ‘msgbox’ function prevents the user from interacting with other windows in MATLAB before responding to the ‘msgbox’. After all of the above code has been entered into an editor, the completed ‘.m’ file needs to be saved and then can be run. The result of running the ‘helloWorld.m’ file is shown, Figure 2.4 where we see both the ‘Push me’ button and the ‘Hello World!’ Message box.

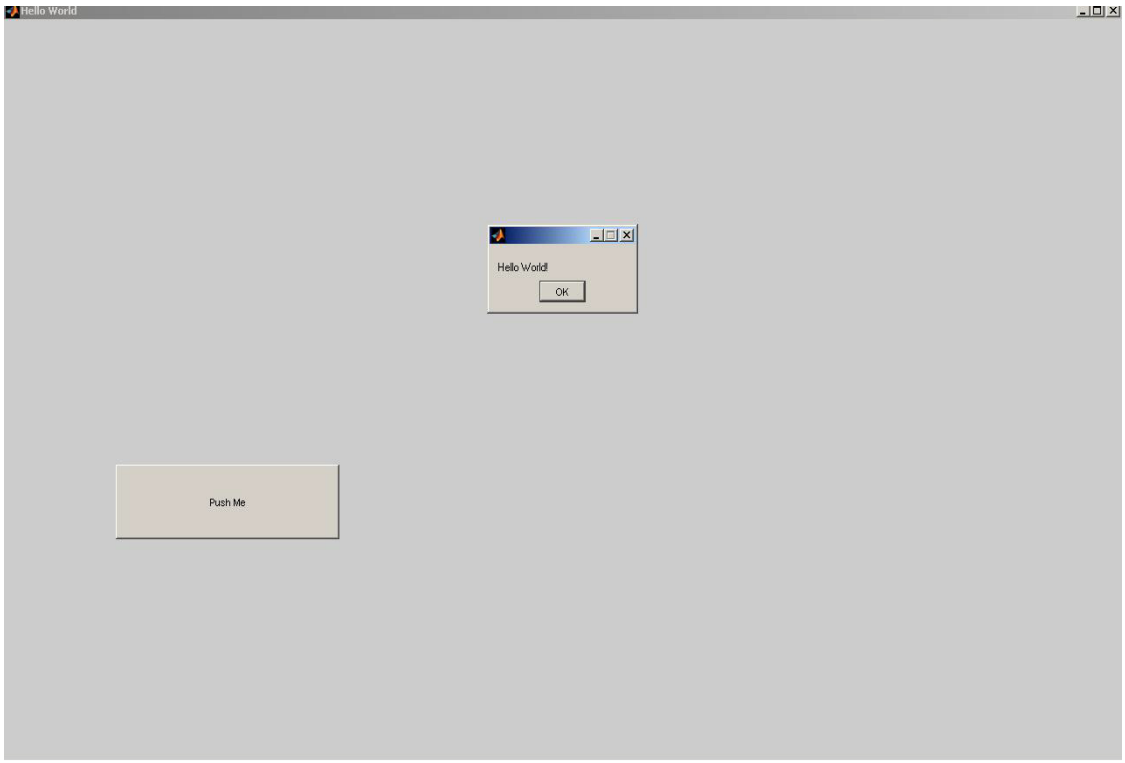

Figure 2.4 The fully functioning 'Hello World' GUI with a message box displaying the message 'Hello World!' when the 'Push me' button is clicked.

2.2.2 Program 2 - Display the waveform of a designated speech file.

In order to display the waveform of a designated speech file, the user must first sketch the desired layout of the GUI on paper. The GUI for Program 2 will require a GUI window , a 'pushbutton' (to initiate the plot) and a 'plot window' (graphic panel) within which the designated speech file is displayed. The user should also have a button which, when clicked, closes this GUI. The user can also draw individual panels around both the buttons and the 'plot window' to make the resulting GUI more visually appealing and to create a visual separation between the button space and the plot space. The 'plot window' is also referred to as a 'graphic panel'. This makes a total button count of two, a panel count of two and a graphic panel count of one for the basic GUI. Figure 2.5 shows how the GUI should look on paper.

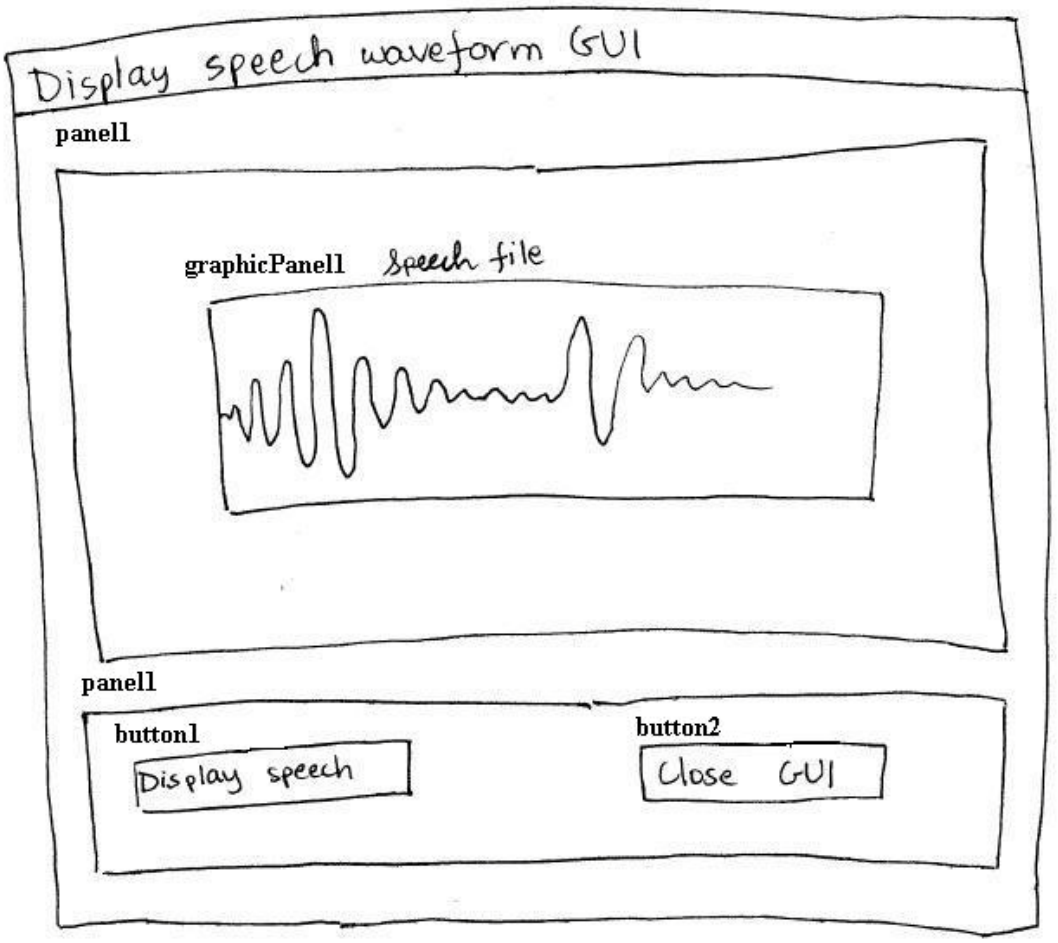

Figure 2.5 A sketch of the tentative layout of the 'Display Speech Waveform GUI'.

The first step in building the GUI for Program 2 is to create a 'displaySpeechWaveform' function, as follows:

```
function displaySpeechWaveform

%embedded code for the GUI application
end
```

Once the 'displaySpeechWaveform' function has been created, the desired GUI code must be embedded into it. The user needs to create a figure window and the two panels that will enclose the buttons and the plot window. The 'plot window' is also referred to

as a ‘graphic panel’. The following code is used to create the GUI window and its panels.

```
clc;clear all;
f = figure('Visible','on',...
            'Units','normalized',...
            'Position',[0,0,1,1],...
            'MenuBar','none',...
            'NumberTitle','off');
% Assign the GUI a name to appear in the window title.
set(f,'Name','Display speech waveform GUI');
%GUI PANELS
%This GUI is divided into two panels, a panel to group the buttons and
a %panel to enclose the graphic panel.
panel1=uipanel('Parent',f,...
            'Units','Normalized',...
            'Position',[0.1 0.05 0.75 0.2]);%button panel

panel2=uipanel('Parent',f,...
            'Units','Normalized',...
            'Position',[0.1 0.3 0.75 0.65]);%plot window
                                        %(graphic panel) panel
```

Creating the panels is extremely tedious since setting the ‘position’ attribute of the ‘uipanel’ (panel object) object requires multiple tries to get the panel positions and dimensions to look just right. Figure 2.6 displays the ‘Display speech waveform GUI’ window with only the panels drawn on it.

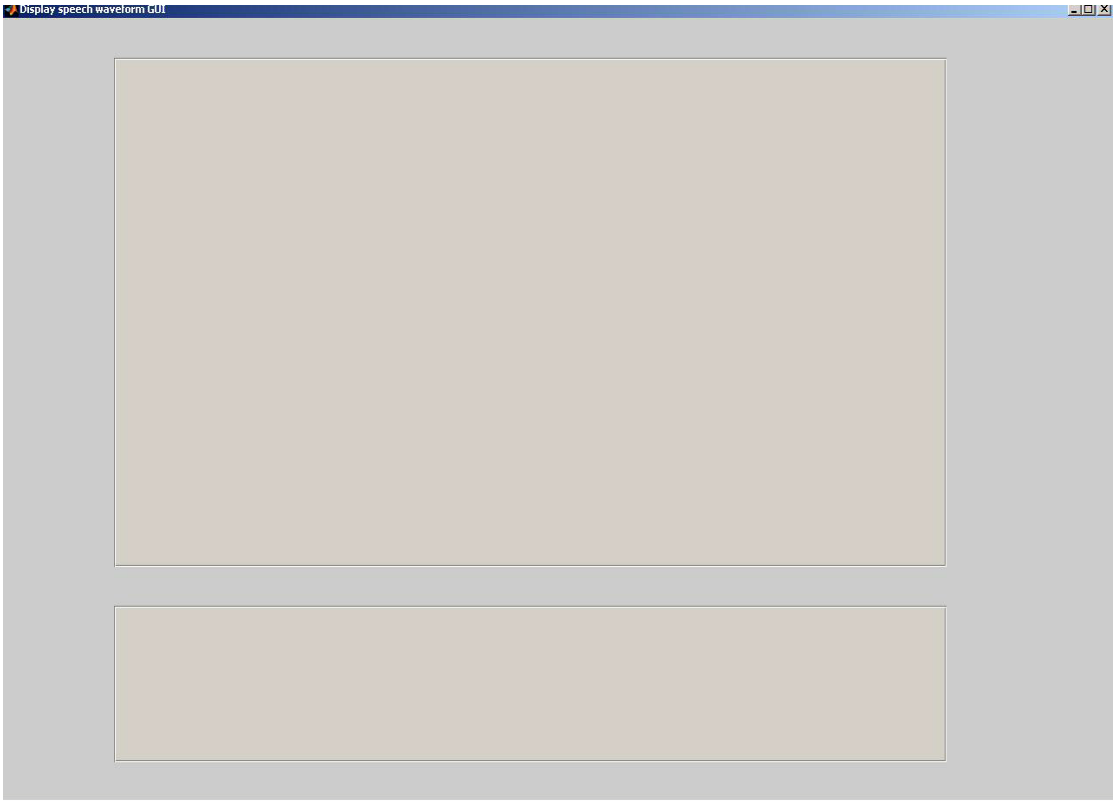

Figure 2.6 The 'Display speech waveform GUI' with two panels created in it. The two large gray rectangles contained within the GUI window are the panels. They are used to visually separate the button space and the plot space.

Now that the two panels have been created, the next step is to create the buttons and the 'graphic panel', as follows:

```
%The speech waveform will be displayed within graphicPanel.
graphicPanel = axes('parent',panel2,...
            'Units','Normalized',...
            'Position',[0.1 0.3 0.8 0.5],...
            'GridLineStyle','--');
```

The user should remember that 'panel2' is the 'parent' of 'graphicPanel' and hence should set the 'parent' attribute of the 'axes' function to 'panel2' as done in the above code snippet.

The next step is to create the buttons for Program 2. The user can use the code used in Program 1 as a template for creating the required pushbuttons.

```
%BUTTONS
% Display speech waveform button
displaySpeechbutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.1 0.3 0.2 0.3],...
            'String', 'Display speech waveform',...
            'Callback',@displaySpeechCallback);

% Close GUI button
closebutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.6 0.3 0.2 0.3],...
            'String', 'Close GUI',...
            'Callback',@closeCallback);
```

Now that the buttons have been created, the display should look like the screen shown in Figure 2.7.

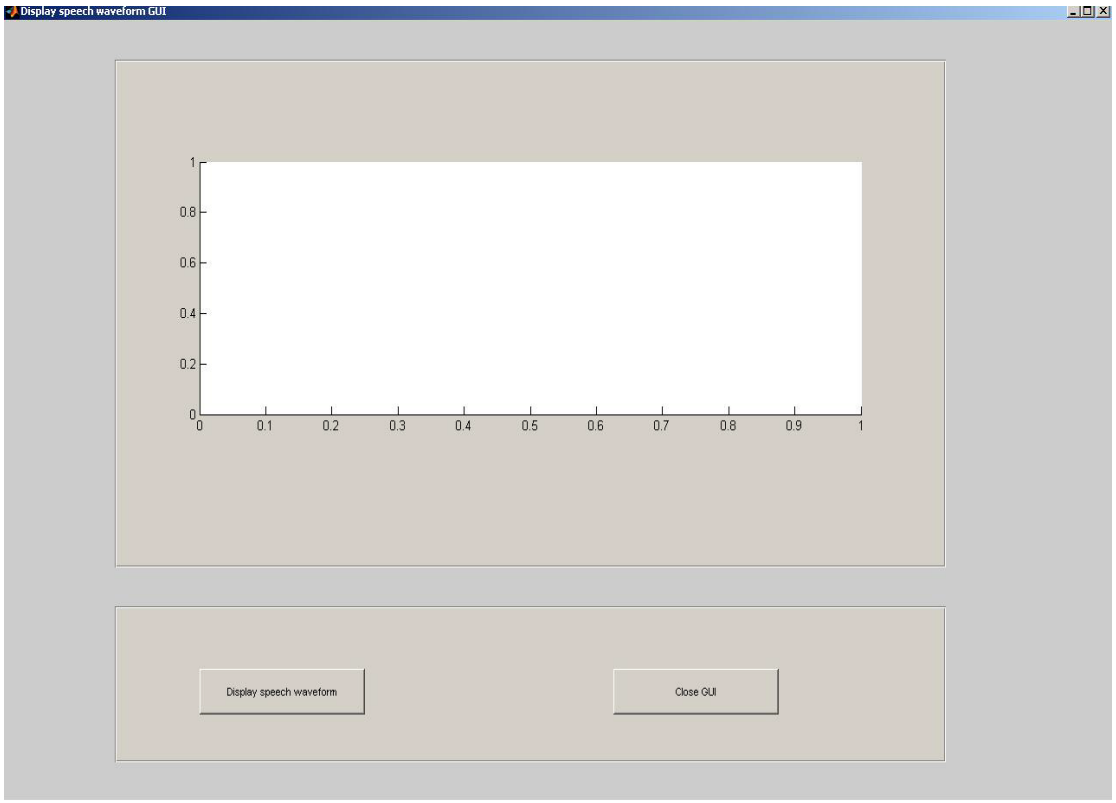

Figure 2.7 The 'Display speech waveform GUI' with two panels, one graphic panel, and the 'Display speech waveform' and 'Close GUI' 'pushbutton' buttons. Callbacks for the two buttons have not been written as yet.

Now that the buttons have been created, the callbacks (i.e., the working code that gets executed on each button press) must be provided, and is of the form:

```
%callback for the display speech waveform button
    function displaySpeechCallback(h,eventdata)
            loadedSpeech=wavread('s1.wav');
            %The speech file is 's1.wav'
            axes(graphicPanel);
            plot(loadedSpeech);
            title('s1.wav');
            xlabel('Time in seconds');
            ylabel('Amplitude');
```

```
    end
```

For the ‘displaySpeechWaveformCallback’ code to work properly, the speech waveform that is going to be displayed (in this case the speech array from file ‘s1.wav’) must be saved in the directory that the code for ‘displaySpeechWaveformCallback.m’ file is saved in. The above callback loads the image ‘s1.wav’ from the current directory and displays it on the ‘graphicPanel’. The image has been titled, ‘s1.wav’.

```
%callback for the close GUI button
    function closeCallback(h,eventdata)
            close(gcf);
    end
```

The ‘closeCallback’ code closes the current GUI window. Figure 2.8 displays the completed GUI. The final Program 2 code is included in Appendix C.

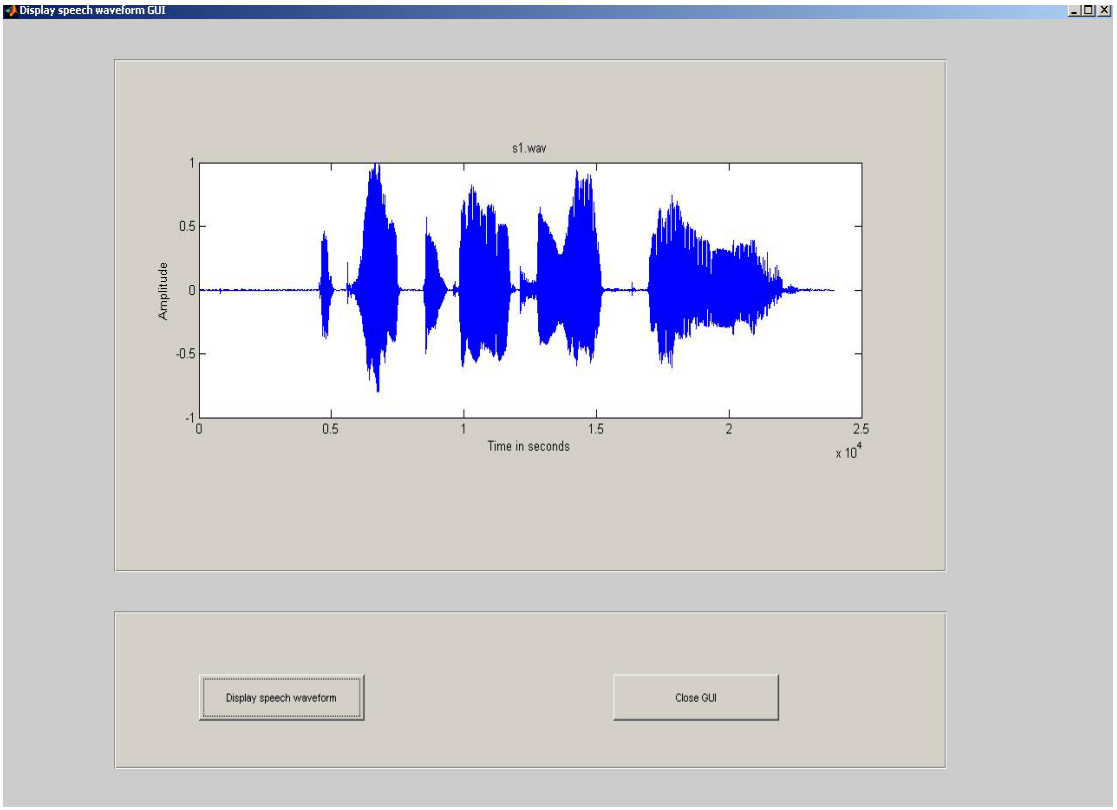

Figure 2.8 The fully functioning and completed 'Display speech waveform GUI'. On clicking the 'Display speech waveform' button, 's1.wav' is displayed.

2.2.3 Program 3 - Load a Speech File, Play it Back and Display the Waveform.

Program 3 shows how to build a GUI to browse a directory in order to find and load a speech file, play out the speech array and finally display the waveform of the speech file. Again the user first needs to sketch the layout of the GUI. For this GUI, we again use two 'panels' to create a well organized GUI. The first panel is a button panel that contains the required buttons, the second panel contains the graphic panel which is used to display the speech waveform. The buttons that are required for this GUI are a 'Get directory' button (which is used to browse the file system), a 'popupmenu' button (which will be populated with the file list from the selected directory), a 'Play' button, a

'Plot' button and a 'Close GUI' button. Figure 2.9 shows a sketch of the layout for the Program 3 GUI.

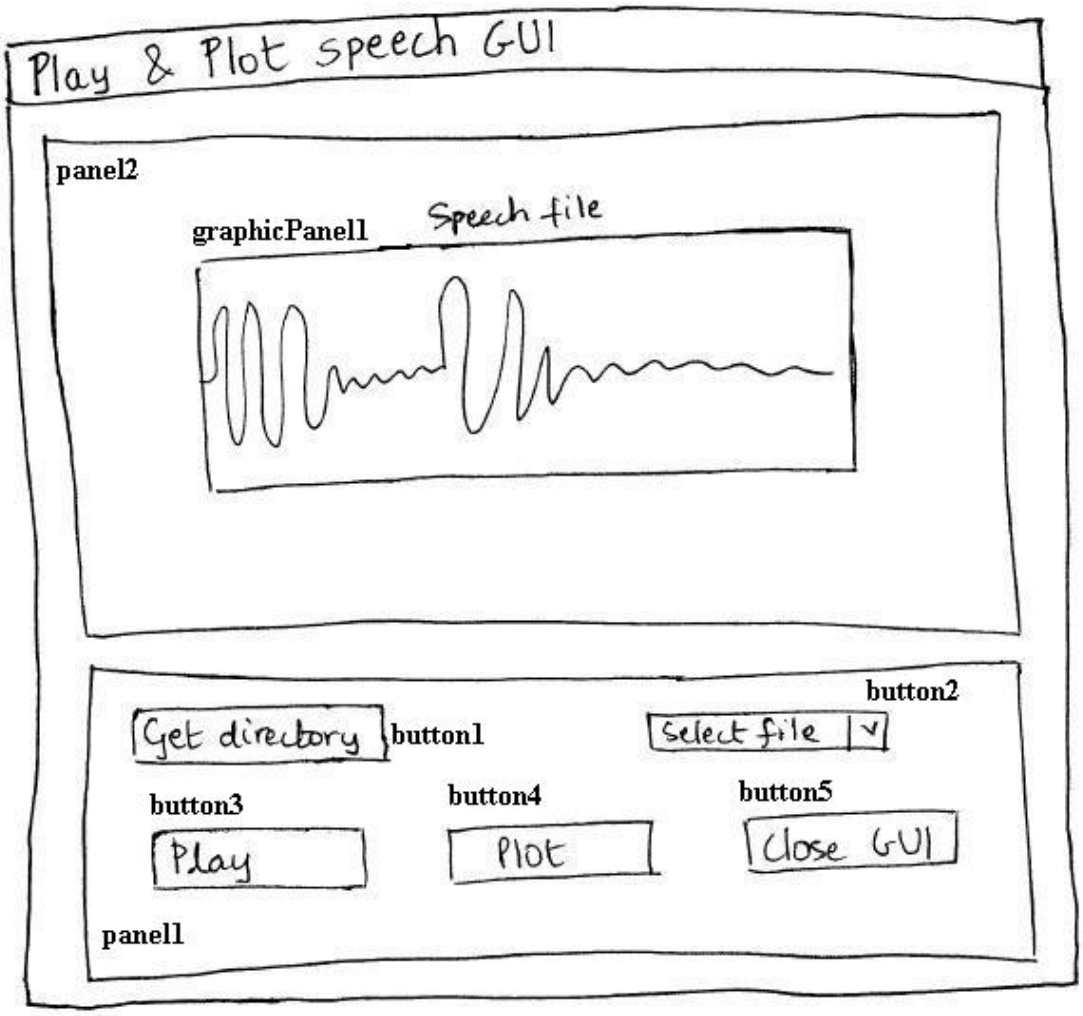

Figure 2.9 Sketch of the layout of the Program 3 GUI.

We begin writing the code for Program 3 by first creating the framework for the different functions, using the code:

```
function playPlotSpeechGUI
%embedded code for the GUI application
end
```

Next the GUI window and the panels are created, as in the previous examples, using the code:

```
clc;clear all;
%variable initialization
%the variables have been initlaized
%to dummy values.
curr_file=1;
fs=1;
directory_name='ABCD';
wav_file_names='ABCD';
file_info_string='ABCD';

f = figure('Visible','on',...
            'Units','normalized',...
            'Position',[0,0,1,1],...
            'MenuBar','none',...
            'NumberTitle','off');
% Assign the GUI a name to appear in the window title.
set(f,'Name','Play and plot speech GUI');
%GUI PANELS
%This GUI is divided into two panels
panel1=uipanel('Parent',f,...
            'Units','Normalized',...
            'Position',[0.1 0.05 0.75 0.35]);%button panel

panel2=uipanel('Parent',f,...
            'Units','Normalized',...
            'Position',[0.1 0.45 0.75 0.5]);%plot window
                                        %(graphic panel) panel
```

The variable initialization segment of the code will be explained during the callback section of this program. Now that the panels which organize the GUI have been

created, the next step is to create the graphic panel within which the plot of the speech file will be displayed.

```
%The image will be displayed within graphicPanel.
graphicPanel = axes('parent',panel2,...
            'Units','Normalized',...
            'Position',[0.1 0.2 0.8 0.7],...
            'GridLineStyle','--');
```

Note that the ‘parent’ attribute of the ‘axes’ function has been set to ‘panel2’ since the ‘graphicPanel’ is found in ‘panel2’.

The next step is to create the buttons which control the GUI. Again, the user can redeploy the code used in Program 2 for creating pushbuttons.

```
%BUTTONS
% Get directory button
getDirectorybutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.25 0.55 0.2 0.25],...
            'String', 'Get directory/Select file',...
            'Callback',@getDirectoryCallback);
```

The ‘Get directory’ button is a ‘pushbutton’. Hence the style attribute of the ‘uicontrol’ object used to create the button need not be specified. The ‘parent’ attribute of the ‘uicontrol’ function has been set to ‘panel1’ since the ‘Get directory’ button is found in ‘panel1’.

The next button that needs to be created is a ‘popupmenu’ button which creates a drop down menu. This drop down menu will be populated with the speech files present in

the directory selected using the ‘Get directory’ button. Not that the ‘style’ attribute of the ‘uicontrol’ object must be changed to ‘popupmenu’ for the ‘Select file’ button.

```
% Select file button
selectFilebutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.55 0.45 0.2 0.25],...
            'style','popupmenu',...
            'BackgroundColor','white',...
            'String', 'Select file',...
            'Callback',@selectFileCallback);
```

The last three buttons i.e., ‘Play’, ‘Plot’ and ‘Close GUI’ are ‘pushbuttons’ and can be created easily using the code from the earlier ‘pushbuttons’ as templates.

```
%Play button
playbutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.1 0.1 0.2 0.25],...
            'String', 'Play',...
            'Callback',@playCallback);

%Plot button
plotbutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.4 0.1 0.2 0.25],...
            'String', 'Plot',...
            'Callback',@plotCallback);

% Close GUI button
closebutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.7 0.1 0.2 0.25],...
            'String', 'Close GUI',...
            'Callback',@closeCallback);
```

Figure 2.10 displays the GUI window visible to the user after the two panels, the graphic panel and the five buttons have been created.

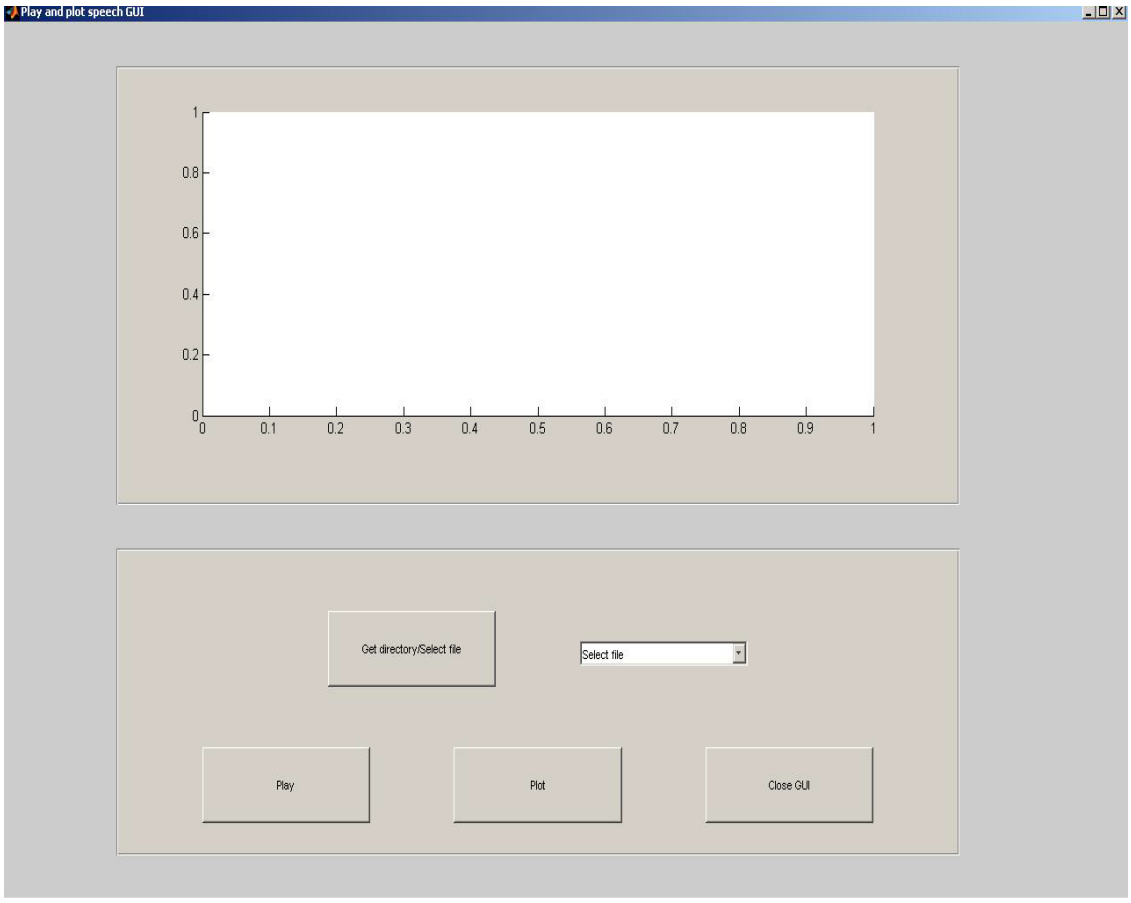

Figure 2.10 Screenshot of the GUI window after the all the GUI elements have been added to

it. Callbacks for the buttons have not yet been written.

The callbacks for the different buttons are described below. The user does not need to write code that is identical to the code displayed in the callbacks below. This code is just an example that the user can use as a guideline to write his (or her) own code. The 'getDirectoryCallback' function is used to browse and load the contents of the selected directory from the user's computer into the 'selectFilebutton' button. The 'loadSelection' function loads the default/initial value of the drop down menu into the

'curr_file' variable which stores the index of the speech file to be played. This function can be ignored or completely re-implemented by the user. Within the 'getDirectoryCallback' function, there are some variables that need to be shared with the other callback functions. These variables need to be initialized at the beginning of the 'playPlotSpeechGUI' code to dummy values so that they can be shared amongst all the callback functions. This initialization of variables has been included in the code just before the GUI window is created using the 'figure' command. The callback code [9] for the various buttons is as follows:

```
%Callbacks
%Get directory callback

function getDirectoryCallback(src,eventdata)
        directory_name = uigetdir('start_path','dialog_title');
        A=strvcat(strcat((directory_name),'\*.wav'));
        struct_filenames=dir(A);
        wav_file_names={struct_filenames.name};
        set(selectFilebutton,'String',wav_file_names);
        %once the popupmenu/drop down menu is created, by default, the
first
        %selection from the popupmenu/drop down menu must be loaded
even if the
        %callback for the popupmenu/drop down menu id not called

        indexOfDrpDwnMenu=1;%by default first option from the
popupmenu/dropdown
                        %menu will be loaded
        [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu);
end
```

This ‘selectFileCallback’ callback gets the value entered into the ‘selectFilebutton’ button and passes the value to the ‘loadSelection’ function. The callback code for the ‘selectFilebutton’ button is as follows:

```
%Select file callback
function selectFileCallback(src,eventdata)
      indexOfDrpDwnMenu=get(selectFilebutton,'val');
      [curr_file,fs]=loadSelection(directory_name,...
      wav_file_names,indexOfDrpDwnMenu);
end
```

The function code for the ‘loadSelection’ function is as follows:

```
%Function--load selection
function [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu)
        fin_path=strcat(directory_name,'\',...
        strvcat(wav_file_names(indexOfDrpDwnMenu)));
            %fin_path is the complete path of the file .wav file that
is
            %selected
            clear curr_file;
      clear fs;
      [curr_file, fs]=wavread(fin_path);
      FS=num2str(fs);
      %Information about the file being played
      file_info_string=strcat('Current file = ',...
      wav_file_names(indexOfDrpDwnMenu),...
      '. Sampling frequency = ',FS,'Hz',...
      '. Number of samples in file = ',...
      num2str(length(curr_file)));
```

The callback code for the ‘playbutton’ is used to play the selected speech file. It also clears the ‘graphicPanel’. This is required because if the user selects, plays and plots

file1, the ‘graphicPanel’ will display the waveform of file1. Then if the user selects and plays file2, the ‘graphicPanel’ will still contain the image of file1 unless it is automatically cleared in the ‘playCallback’. The resulting callback code is thus of the form:

```
%Callback for the playbutton
function  playCallback(h,eventdata)
      sound(curr_file,fs);
      reset(graphicPanel); %clearing the graphic panel
      temp=0;
      plot(temp);
end
```

The callback code for the ‘plotbutton’ is as follows:

```
%callback for the plotbutton
 function plotCallback(h,eventdata)
      hold off;  %It is essential to turn hold off so that the
      hold off;  %earlier contents of the panel are replaced
      %the  two  hold  off's  are  for  the  speech  file  and  the  hamming
window
            grid off;
      reset(graphicPanel);
      axes(graphicPanel);
      l=length(curr_file);
      i=(1:l)/fs;%coverting samples to time
      plot(i,curr_file,'k','LineWidth',2),...
           xlabel('Time in seconds'),...
           ylabel('Amplitude');
      title(file_info_string);
      axis tight;
      grid on;
 end
```

The callback code for the 'closebutton' is as follows:

```
%Callback for close
function closeCallback(h,eventdata)
    close(gcf);
end
```

Once all of the above code has been entered into the MATLAB editor, the user must save and run the program. The final code for Program 3 is shown in Appendix C. Figure 2.11 displays the completed GUI for Program 3.

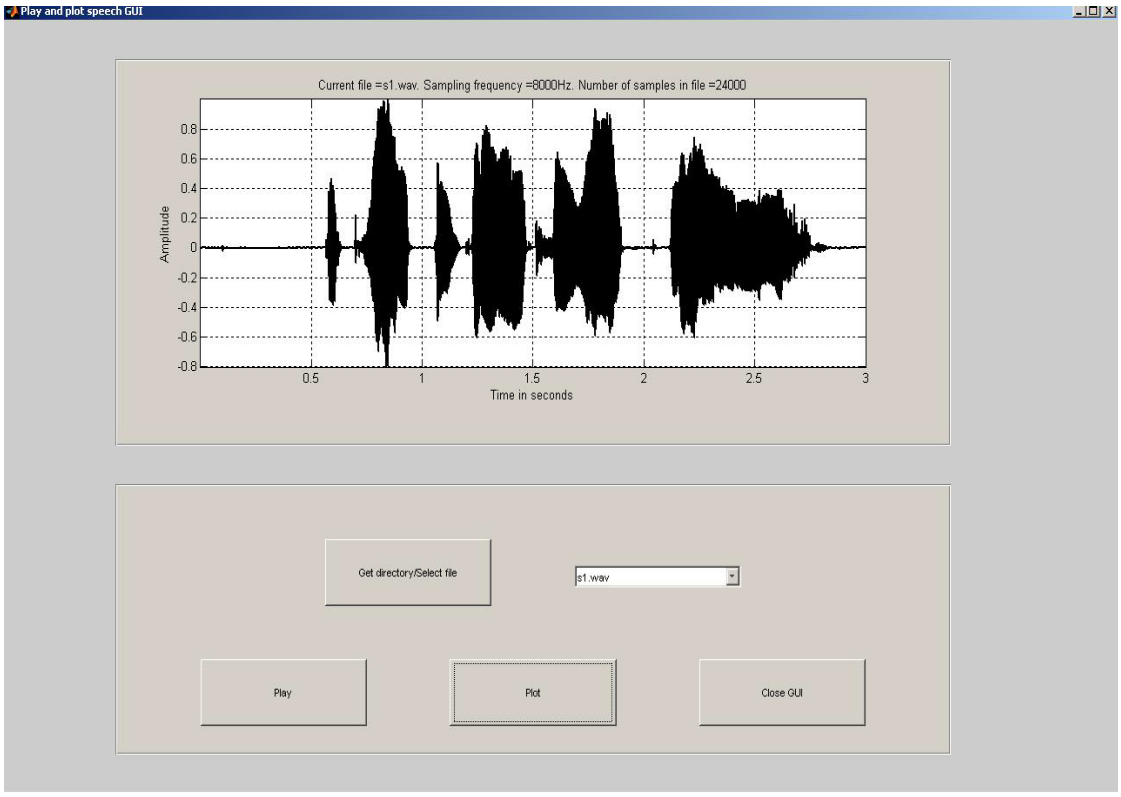

Figure 2.11 The fully functioning and completed GUI for Program 3. On selecting the desired speech file using the 'Get directory' and 'Select file' buttons, the 'Plot' button will display the waveform of the selected speech file in the 'graphicPanel.'

2.2.4 Program 4 - Load an Existing Speech File or Record a New Speech File. Play the File, Display a Waveform of the Speech File and Save the File.

Since the previous three examples have a lot of common steps, in this example, only the steps that are different from the previous ones will be covered in detail. As always, the

user needs to visualize and sketch the panels, graphic panels and buttons of the GUI. The sketch of the GUI for Program 4 is given in Figure 2.12.

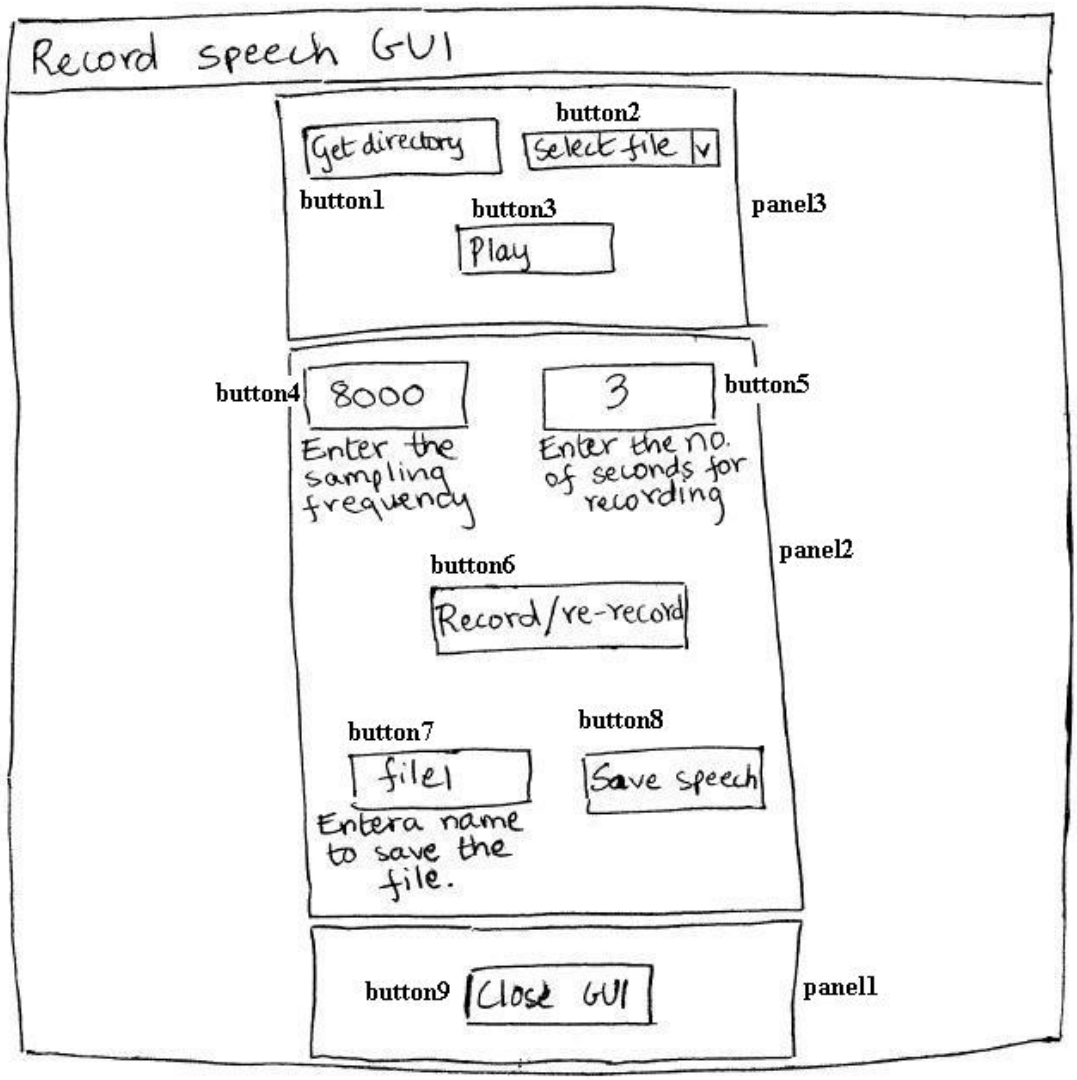

Figure 2.12 A tentative sketch of the Program 4 GUI.

Program 4 again uses panels to organize the GUI. In this program we use multiple panels to organize the GUI, since groups of buttons, related to the same function, can

be grouped together and thus dependencies and the errors caused by clicking buttons not meant to be used consecutively can be minimized. e.g., the ‘Get directory’ and ‘Select file’ buttons in Program 3 which were used to browse a selected directory and populate the drop down menu with the speech files in the current directory.

The buttons that our implementation of Program 4 requires are a ‘Get directory’ and ‘Select file’ button to browse and load the desired speech file. A ‘Play’ button is used to play the selected speech file. To implement the record speech section of Program 4, the user also needs to create a ‘Record/re-record’ button. Two ‘edit’ buttons are used to enter the parameters for recording speech (sampling frequency and number of seconds for recording) should also be created. Next, the user needs to create an ‘edit’ button in which the filename used to save the recorded speech can be entered. Finally the user needs to create two additional ‘pushbuttons’, one to save the recorded speech and the other to close the GUI window. These buttons should also be incorporated into the design for the Program 4 GUI.

In our implementation of Program 4, the buttons are distributed into three panels to organize them more efficiently. The panels that were created are the following:

- ‘Panel1’ contains the ‘Get directory’, ‘Select file’ and ‘Play’ buttons.
- ‘Panel2’ contains the ‘Enter the sampling frequency in Hz’ ‘edit’ button and the ‘text’ button that contains the label for the ‘edit’ button. ‘Panel2’ also contains the ‘Enter the number of seconds for recording’ ‘edit’ button and its label. It also contains the ‘Record/re-record’ button, the ‘Enter a filename to save recorded speech’ ‘edit’ button, its label and the ‘Save speech’ action button.
- ‘Panel3’ contains the ‘Close GUI’ button.

The steps that are followed in the creation of the GUI for Program 4 are as follows:

1. Create the GUI function framework.
2. Create a set of three panels.
3. Create the set of nine buttons.
4. Write the callbacks for all the buttons.

The first step is to enter the GUI function framework into the MATLAB editor. Next the GUI window and the panels that organize the GUI must be created. These two steps can be done as shown in the code below:

```
function recordGUI
%embedded code for the GUI application
clc;clear all;
%INITIALIZATION
%The variables returned from the edit boxes must
%be initialized, else the box will only display
%the value but not actually hold that value
curr_file=1;
directory_name='ABCD';
wav_file_names='ABCD';
y=1;%y is the variable that contains the recorded speech
nsec=3;
fs=8000;
fileName='file1';
%The 1 1 in the position attribute means that the GUI
%is fit to screen
f = figure('Visible','on',...
    'Units','normalized',...
    'Position',[0,0,1,1],...
    'MenuBar','none',...
    'NumberTitle','off');

%GUI PANELS
%This GUI is divided into four panels
```

```
panel1=uipanel('Parent',f,...
    'Units','Normalized',...
    'Position',[0.3 0.65 0.3 0.345]);%top panel
panel2=uipanel('Parent',f,...
    'Units','Normalized',...
    'Position',[0.3 0.245 0.3 0.4]);%center panel
panel3=uipanel('Parent',f,...
    'Units','Normalized',...
    'Position',[0.3 0.04 0.3 0.2]);%bottom panel
% Assign the GUI a name to appear in the window title.
set(f,'Name','Record speech GUI');
%Initialize GUI
set([f,panel1,panel2,panel3],'Units','normalized')
end
```

After creating the three panels, Figure 2.13 will be visible to the user.

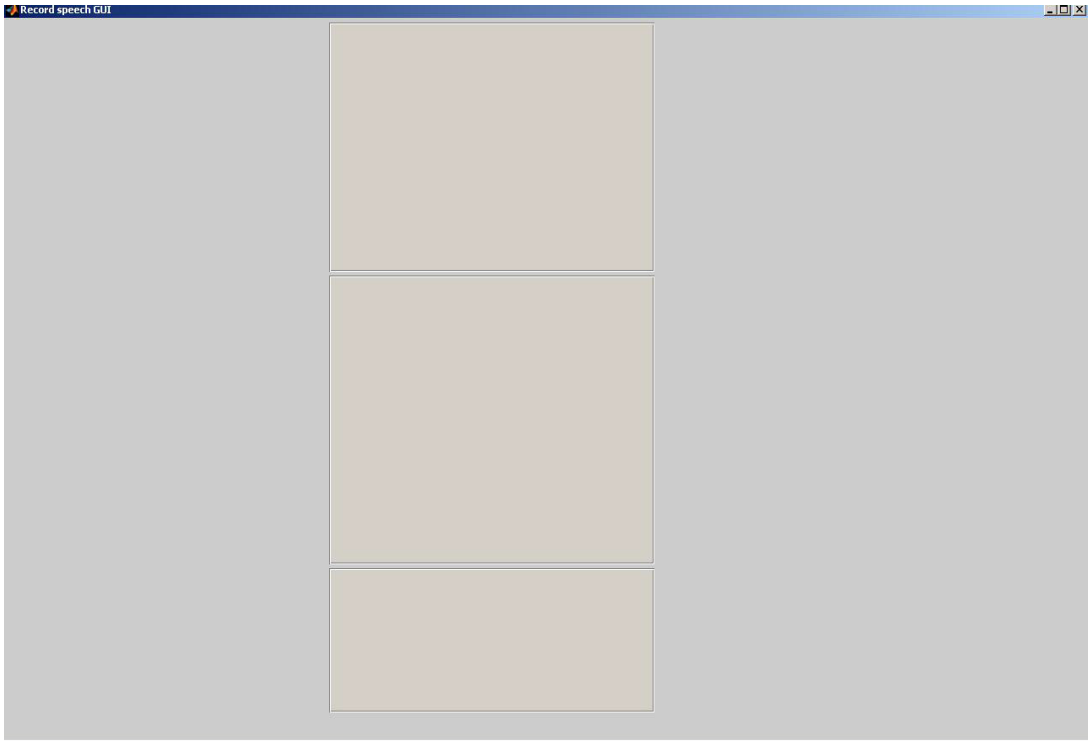

Figure 2.13 This GUI window contains the three panels that have been created for grouping the various buttons of the GUI.

The above code includes code for the initialization of variables. This will be discussed during the button creation and callbacks section of Program 4. The next step is to create the buttons for this GUI. The code for creating the buttons for 'Panel1' is as follows:

```
%BUTTONS
% Get directory button
getDirectorybutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.05 0.6 0.35 0.2],...
            'String', 'Get directory/Select file',...
            'Callback',@getDirectoryCallback);

% Select file button
selectFilebutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.55 0.5 0.35 0.25],...
            'style','popupmenu',...
            'BackgroundColor','white',...
            'String', 'Select file',...
            'Callback',@selectFileCallback);

%Play  button
playbutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.3 0.2 0.35 0.2],...
            'String', 'Play speech',...
            'Callback',@playCallback);
```

In the above code for creating the buttons for Panel 1, the user should note that the 'parent' attribute of the 'Get directory', the 'Select file' and the 'Play' buttons should be set to 'panel1' which is the panel in which they reside.

Setting the ‘position’ attribute of the buttons is an extremely tedious task since it takes numerous adjustments and tweaks to set the position of the buttons to exactly the right position and dimensions.

The code to create the buttons for ‘Panel2’ is shown below. ‘Panel2’ contains a new type of button, the ‘edit’ button and its label which is a ‘text’ button. An ‘edit’ button is created by setting the ‘style’ attribute of the ‘uicontrol’ object to ‘edit’. Another change in the definition of this button is that the ‘backgroundcolor’ attribute is set to ‘white’ so that the button is white in color and not gray. Also the ‘string’ attribute must be set to the user defined default value of the ‘edit’ button. The resulting code is:

```
%Enter sampling frequency for recording in Hz button
fsbutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.05 0.8 0.35 0.15],...
            'style','edit',...
            'String', '8000',...
            'BackgroundColor','white',...
            'Callback',@fsCallback);
```

In the above example, the ‘string’ attribute has been set to ‘8000’ which is the default value for the sampling frequency.

```
%Label for 'Enter sampling frequency for recording in Hz' button
fsLabelbutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.05 0.63 0.35 0.15],...
            'style','text',...
            'String', 'Enter sampling frequency for recording in Hz');

 %Enter the number of seconds for recording button
nsecbutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.55 0.8 0.35 0.15],...
```

```
            'style','edit',...
            'String', '3',...
            'BackgroundColor','white',...
            'Callback',@fsCallback);
```

In the above example, the ‘string’ attribute has been set to ‘3’ which is the default value for the number of seconds for recording.

```
%Label for 'Enter the number of seconds for recording' button
nsecLabelbutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.55 0.63 0.35 0.15],...
            'style','text',...
            'String', 'Enter the number of seconds for recording');
%Record/re-record button
recordbutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.3 0.5 0.35 0.15],...
            'String', 'Record/re-record',...
            'Callback',@recordCallback);

%Enter a file name to save the recorded speech button
fileNamebutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.05 0.2 0.35 0.15],...
            'style','edit',...
            'String', 'file1',...
            'BackgroundColor','white',...
            'Callback',@fileNameCallback);
```

In the above example, the ‘string’ attribute has been set to ‘file1’ which is the default value for the file name that the recorded speech will use if/when it is saved in a new file.

```
%Label for 'Enter a file name to save the recorded speech' button
filenameLabelbutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.05 0.03 0.35 0.15],...
```

```
            'style','text',...
            'String',  'Enter  a  file  name  to  save  the  recorded
speech');
%Save speech  button
savebutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.55 0.2 0.35 0.15],...
            'String', 'Save speech',...
            'Callback',@saveCallback);
% Close GUI button
closebutton=uicontrol('Parent',panel3,...
            'Units','Normalized',...
            'Position',[0.3 0.25 0.35 0.4],...
            'String', 'Close GUI',...
            'Callback',@closeCallback);
```

Figure 2.14 displays the completed ‘recordGUI.m’ GUI.

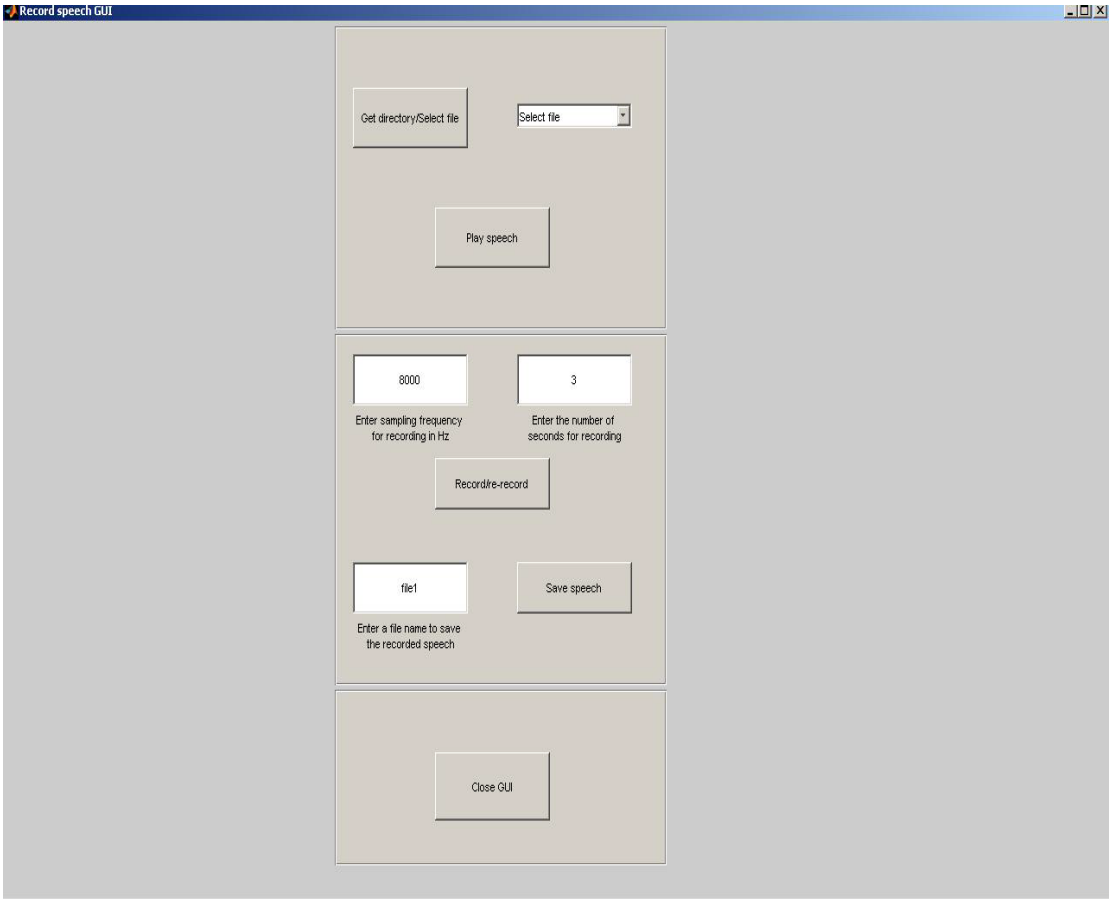

Figure 2.14 The GUI window visible to the user once the 'recordGUI.m' file has been saved and run. The callbacks for the GUI have not yet been written at this point.

The next step is to write the callbacks [9] for the buttons that have been created. The callback codes for the 'getDirectoryCallback', the 'selectFileCallback' and the 'loadSelection' function are identical to the codes used in Program 3 and have been included below. Some variables in these functions need to be shared among other functions and hence need to be initialized at the beginning of the program. The variables that need to be initialized to dummy values are curr_file, directory_names, wav_file_names, y, nsec, fs and filename.

```
%Callbacks
```

```
function getDirectoryCallback(src,eventdata)
        directory_name = uigetdir('start_path','dialog_title');
        A=strvcat(strcat((directory_name),'\*.wav'));
        struct_filenames=dir(A);
        wav_file_names={struct_filenames.name};
        set(selectFilebutton,'String',wav_file_names);
        %once the popupmenu/drop down menu is created, by default, the
first
        %selection from the popupmenu/drop down menu must be loaded
even if the
        %callback for the popupmenu/drop down menu id not called

        indexOfDrpDwnMenu=1;%by default first option from the
popupmenu/dropdown
                        %menu will be loaded
        [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu);
end

function selectFileCallback(src,eventdata)
        indexOfDrpDwnMenu=get(selectFilebutton,'val');
        [curr_file,fs]=loadSelection(directory_name,...
            wav_file_names,indexOfDrpDwnMenu);
end

%Function--load selection
function [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu)
        fin_path=strcat(directory_name,'\',...
            strvcat(wav_file_names(indexOfDrpDwnMenu)));
            %fin_path is the complete path of the file .wav file that
is
            %selected
        clear curr_file;
        clear fs;
        [curr_file, fs]=wavread(fin_path);
        FS=num2str(fs);
        %Information about the file being played
```

```
        file_info_string=strcat('Current file = ',...
            wav_file_names(indexOfDrpDwnMenu),...
            '. Sampling frequency = ',FS,'Hz',...
            '. Number of samples in file = ',...
            num2str(length(curr_file)));
 end
```

The ‘playCallback’ code is similar to the code used in Program 3.

```
%Callback for the playbutton
 function  playCallback(h,eventdata)
        sound(curr_file,fs);
 end
```

The callbacks [9] for the ‘Enter sampling frequency’ button i.e. ‘fsCallback’ and for the ‘Enter the number of seconds for recording’ button i.e. ‘nsecCallback’ are as follows. The value returned from the ‘edit’ buttons is a string and needs to be converted into a numeric variable. The user can do this using the ‘str2num’ function. Also the variables returned from these functions, fs, and, nsec, need to be initialized to their default values at the beginning of the program.

```
%callback for the fs button
function fsCallback(h,eventdata)
       fs=str2num(get(fsbutton,'string'));
end

%callback for the nsec button
function nsecCallback(h,eventdata)
        nsec=str2num(get(nsecbutton,'string'));
end
```

The code shown below is one way of writing a function that records speech.

```
%callback for the record/ re-record button
%record speech file of fixed duration (nsec) and
%given sampling rate(fs)
```

```
function recordCallback(h,eventdata)
      fsCallback(h,eventdata);
       nsecCallback(h,eventdata);
    % yn=speech samples normalized to 1
    % N is the number of samples in each speech file
    % ch is the number of channels in the recording
        N=fs*nsec;
        ch=1;
        y=wavrecord(N,fs,ch,'double');
        ymin=min(y);
        ymax=max(y);
                % calculate dc offset and correct
        offset=sum(y(N-999:N))/1000;
        y=y-offset;
        sound(y,fs);
 end
```

The function ‘fileNameCallback’ is used to obtain the name of the file in which the recorded speech is to be saved. It retrieves the ‘string’ entered in the ‘fileNameButton’.

```
%callback for filename speech
function fileNameCallback(h,eventdata)
        fileName=get(fileNamebutton,'string');
end
```

The function ‘saveCallback’ is used to save the recorded speech in the specified file and is of the form:

```
%callback for save speech
function saveCallback(h,eventdata)
        currentDir=pwd
        currDir=strcat(currentDir,'\',fileName,'.wav')
        wavwrite(y,fs,strvcat(currDir));
        c=wavread(strvcat(currDir));
        soundsc(c,fs)
end
```

The code for ‘closeCallback’ is used to close the GUI and is:

```
%Callback for close
function closeCallback(h,eventdata)
        close(gcf);
end
```

A screenshot of the final GUI for Program 4 is shown in Figure 2.14 and the final code for Program 4 is included in Appendix C.

2.3 GUI Lite - Version 1: Strengths and Weaknesses

The four examples in section 2.2 clearly explain and demonstrate how to use GUI Lite – Version 1 to create GUIs of varying complexity. GUI Lite – Version 1 provides the user with sufficient functionality to create GUIs for speech processing applications. Along with creating a GUI for the user, GUI Lite – Version 1 teaches a user how to use the various GUI development functions provided by MATLAB like 'uicontrol', 'uipanel', 'figure', etc. We also provide an extremely useful guide/ instructional manual to understand how to put together a good looking GUI using the functions and features provided by MATLAB. The downside of using GUI Lite – Version 1 is that the user needs to enter the position attribute for every GUI object that he or she adds to the GUI layout. Entering the position attribute is an extremely tedious and painstaking task. Automating the process of selecting the positions of the various GUI objects like buttons and panels would greatly reduce the time taken to enter the co-ordinates and size of each of the GUI objects. GUI Lite – Version 2 separates and automates this process of positioning the GUI objects like panels and buttons thereby reducing the time taken to create a GUI and improving the user's experience while using GUI Lite.

Chapter 3 Implementation of a GUI using GUI Lite -Version 2

3.1 GUI Lite - Version 2

GUI Lite - Version 2 is intended to improve the user experience while creating viable GUIs by providing an interface that is more intuitive and simpler than GUI Lite – Version 1. The goal for GUI Lite – Version 1 was to enable the user to focus on the actual application itself, rather than the creation of the GUI. However the resulting GUI Lite – Version 1 required the user to focus a lot of attention on handling and optimizing the actual GUI code. Version 1 focused on helping users to write their own code for the GUI's layout and functioning. The user had to decide the position of each of the buttons and meticulously enter the panel and button screen co-ordinates, and repeatedly adjust them to create a good looking GUI. To solve this problem, a two – step procedure for creating GUIs was conceived in which the first stage was essentially a layout of the GUI created using a set of automated tools, and the second stage was implementation of the application code. This two stage process enabled a user to define panels, graphic panels, title boxes (titles for graphic panels or groups of buttons) and buttons using mouse clicks on a GUI screen. The layout and positioning of the various GUI elements were then stored into a '.mat' file and the layout of the GUI was created and displayed. In the second stage, the callbacks for the various GUI objects were written and integrated into the final program. The complete code and user manual for the GUI Lite – Version 2 toolbox is included in Appendix B.

3.1.1 Naming conventions for GUI Lite – Version 2

GUI Lite – Version 2 lets users create panels, graphic panels, title boxes and buttons using a graphic cursor and mouse clicks. Consecutive panels will be referred to as

'panel1', 'panel2', 'panel3', etc. as per the order in which they were created. Similarly, if three graphic panels are created, they will be referred to using the names 'graphicPanel1', 'graphicPanel2' and 'graphicPanel3' respectively when being called in the callbacks. Also if the user creates two title boxes, they will be referred to as 'titlebox1' and 'titlebox2' based on the order in which they were created. Buttons will be referred to as 'button1', 'button2', 'button3', etc depending on the number of buttons created and the order in which they are created. This sequential numbering process greatly simplifies stage 1 of the GUI Lite – Version 2 GUI creation process.

3.2 Implementation of four Baseline Programs Using GUI Lite - Version 2

In the remainder of this section, we illustrate the process of designing simple GUIs for the 4 program examples discussed in Chapter 2 using GUI Lite – Version 2. The resulting code and supporting explanations and screenshots will help illustrate how GUI Lite – Version 2 greatly simplifies the process of designing Lite GUIs for a range of program complexities. Detailed explainations regarding how to use the GUI Lite – Version 2 toolbox and various objects that can be created using it are mentioned in Appendix B.

3.2.1 Program 1 - Hello world program

The first step in designing a GUI for Program 1 is for the user to visualize and sketch the GUI. Once the user has sketched the GUI and decided which panels and buttons to include, GUI development can begin. Figure 3.1 displays an initial sketch of the Program 1 GUI.

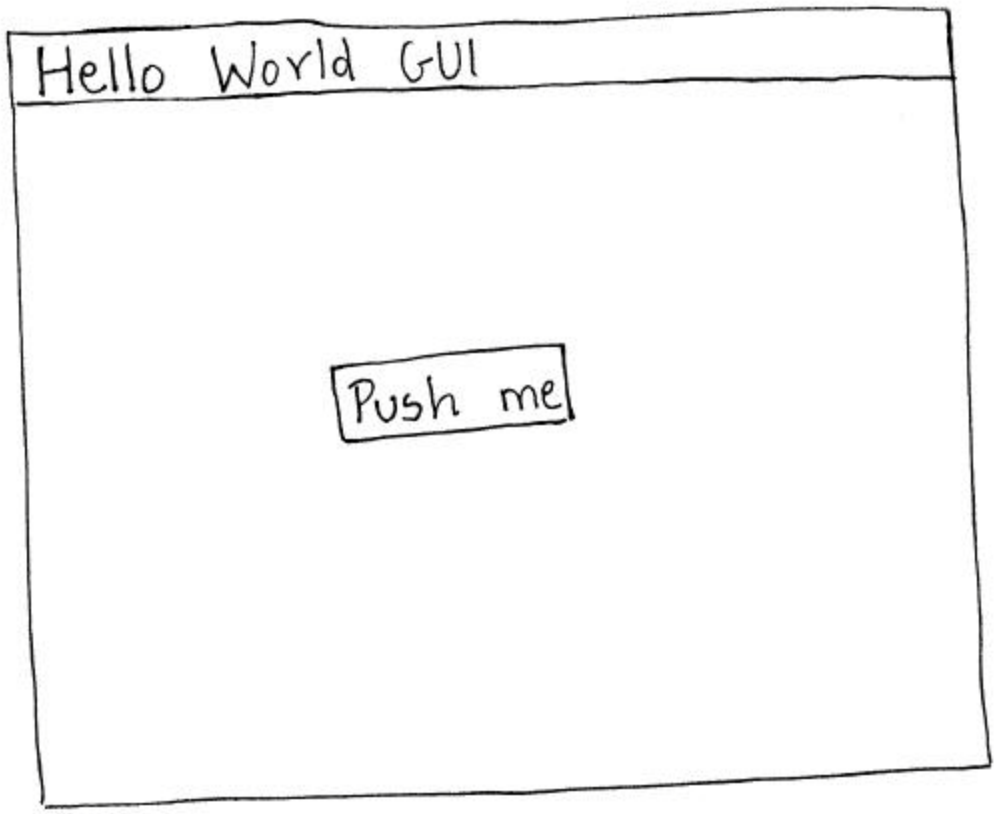

Figure 3.1 A sketch of the layout of the 'Hello World GUI', containing just one button.

The 'Hello World GUI' contains only one button, namely a 'pushbutton' which when clicked displays a message saying 'Hello World'. The GUI also contains a single panel to outline the GUI window and enclose the button. This panel has not been shown in Figure 3.1.

The user first needs to load the GUI Lite folder containing the files 'panelButtonSetup.m', 'runGUI.m' and 'PanelandButtonCallbacks.m' into the MATLAB work folder. After this has been done, the 'panelButtonSetup.m' program should be run. As shown in Figure 3.2, an initial screen, the 'Button/Panel Setup GUI'

is displayed asking the user to enter the total number of panels, graphic panels (plot windows), title boxes and buttons that need to be created. As in GUI Lite – Version 1, panels are used to visually group related GUI elements, graphic panels are used for plotting, title boxes are used to display information regarding a plot or a group of buttons, and buttons are used to perform user defined functions.

Additional buttons are used to define the length and width of a 'standard' size button, a name for the saved GUI code and finally a button to initiate the GUI Lite – Version 2 design process.

For Program 1, the user enters one as the number of panels, zero as the number of graphic panels and title boxes, and one as the number of buttons. Once the user has entered the number of GUI elements, he or she needs to enter a name in with which the file containing the layout of the GUI is saved. For this example, the user chooses the name 'helloworld'. Figure 3.3 displays a screenshot of the 'Button/Panel Setup GUI' window with the user's parameters entered into it. The user can now click the 'Begin drawing panels & buttons' button to begin the GUI Lite – Version 2 design process.

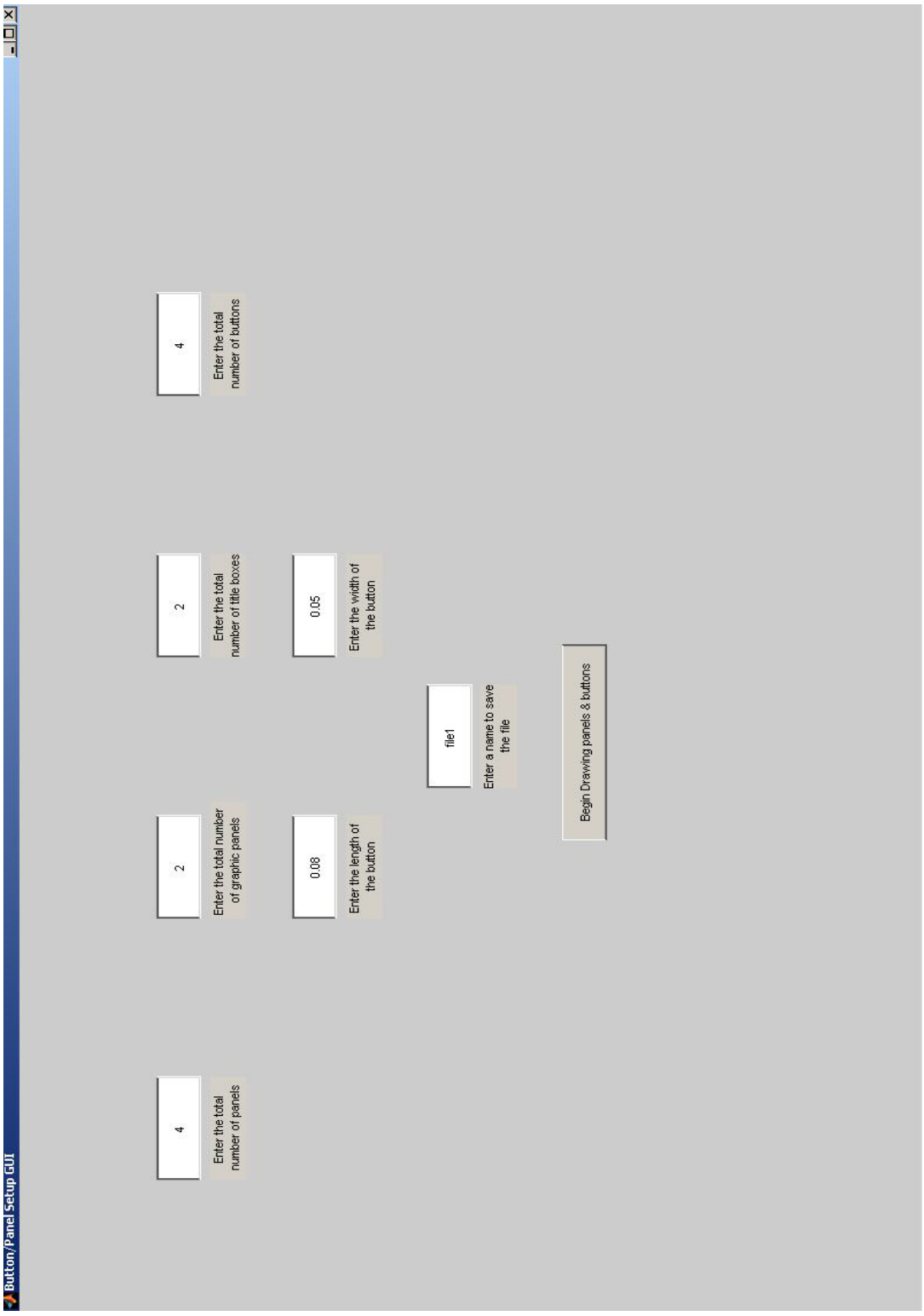

Figure 3.2 The first screen that is visible to the user when the GUI Lite toolbox is launched. It contains default values for the number of panels, graphic panels, title boxes and buttons to be created.

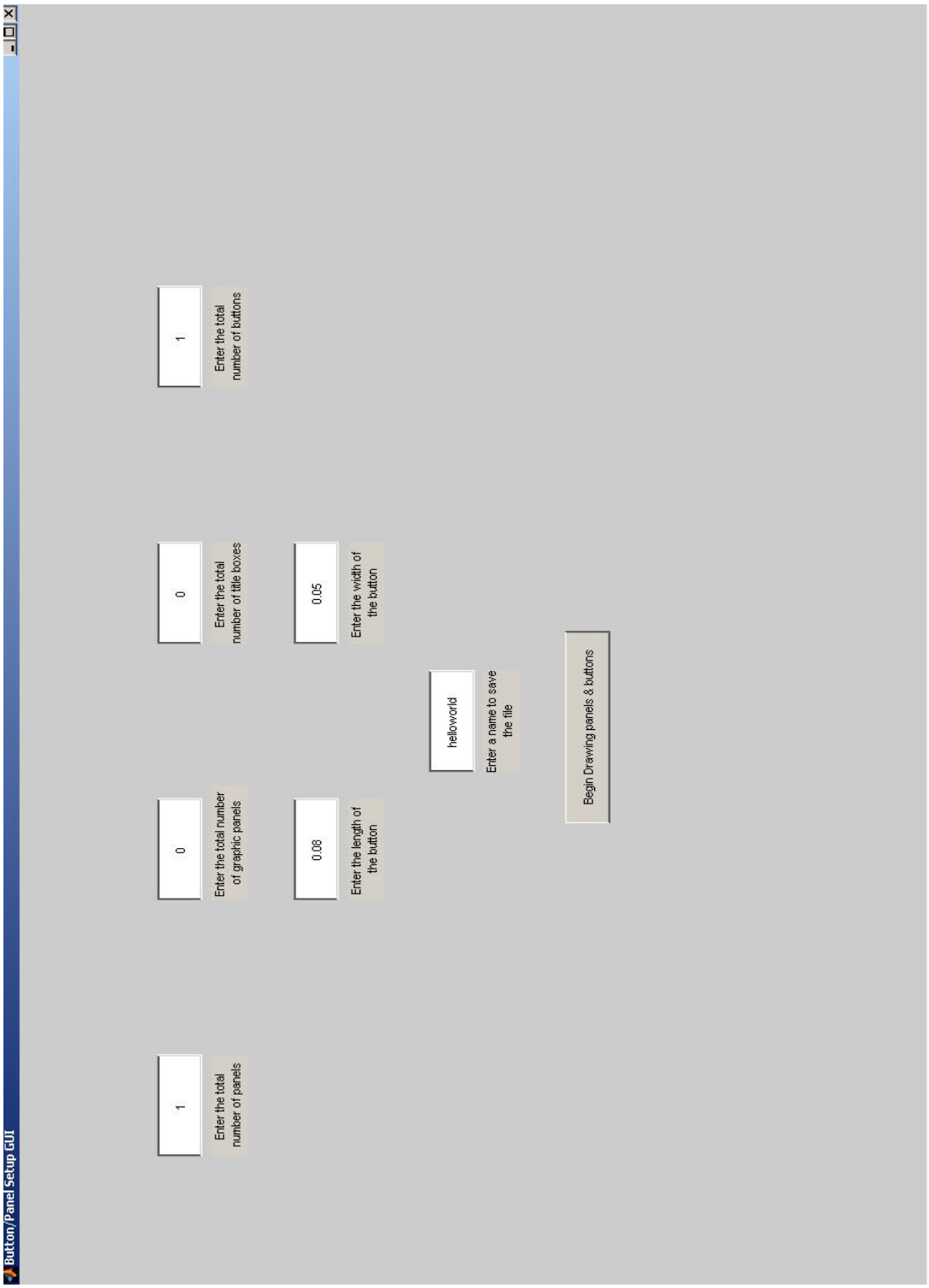

Figure 3.3 The 'Button/Panel Setup GUI' with the user's values for the number of panels, graphic panels, title boxes and buttons entered into it. The filename entered to save the layout is 'helloworld'.

A white screen appears with a grid overlaid over it with horizontal and vertical crosshairs that are visible to the user. Using these crosshairs, the user can select the position and size of the single panel for Program 1. Moving the crosshairs over the desired location and using a mouse click, the initial co-ordinate (the lower left co-ordinate of the panel) is first selected and is the bottom left corner of the panel. The next co-ordinate selected (using the graphic cursor) is the bottom right corner of the panel; the third and final co-ordinate is the top right corner of the panel. Changing the order of selection of co-ordinates will cause an error. The user does not need to enter the fourth co-ordinate of the panel since the GUI Lite toolbox automatically calculates and plots it based on the previous three co-ordinates.

The user can now select the position of the single button. The button requires only one co-ordinate, i.e., the bottom left co-ordinate since a 'standard size' button is assumed. The length and the width of the buttons are entered in the initial 'Button/Panel Setup GUI' shown in Figure 3.2. Once the position of the button has been selected, another GUI window, the 'Button Stats' GUI will appear on the screen. The user selects the type of the button as 'pushbutton' and enters the string to be written on the button as 'Push me'. The 'Push me' button does not require a label explaining its function since the name 'Push me' written over the button is self-explanatory. Also the length and width of the button can be left at their default values and the 'Accept Specs Button' can be clicked. Figure 3.4 displays the 'Button Stats' GUI with the user's parameters for the 'Push me' button.

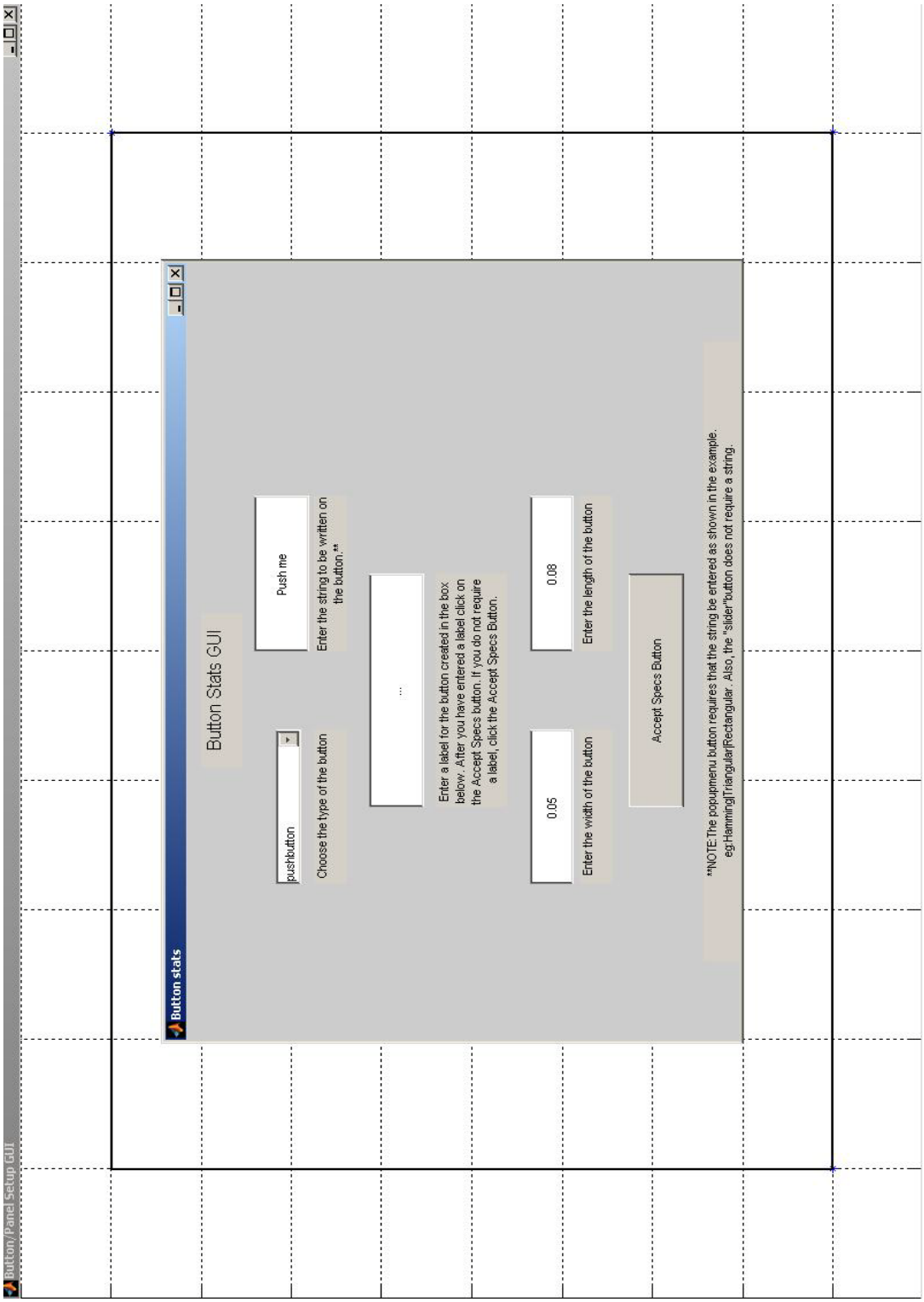

Figure 3.4 The GUI visible to the user after selecting the position of the 'Push me' button. It contains the user parameters for the 'Push me' button.

After clicking the ‘Accept Specs Button’, the user will be prompted to save the selection and can select the ‘Yes’ radio button displayed in Figure 3.5.

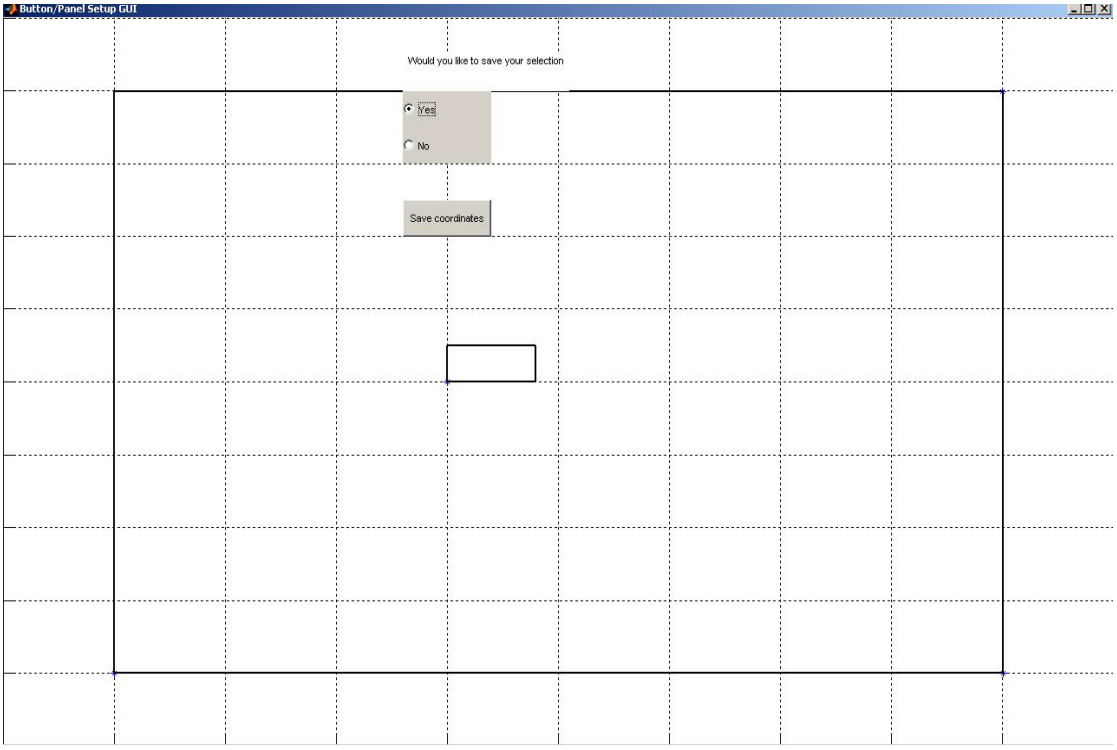

Figure 3.5 The GUI development screen containing, the user defined panel i.e. the large rectangle outlined by a solid black line and the ‘Push me’ button which is the small rectangle in the center of the figure. The user has been prompted to save his selection.

In order to see what the GUI looks like at this stage, the user can edit the name of the ‘.mat’ file in ‘runGUI.m’ to ‘helloworld’, save the ‘runGUI.m’ file and run it. The user will then be able to see the screen displayed in Figure 3.6.

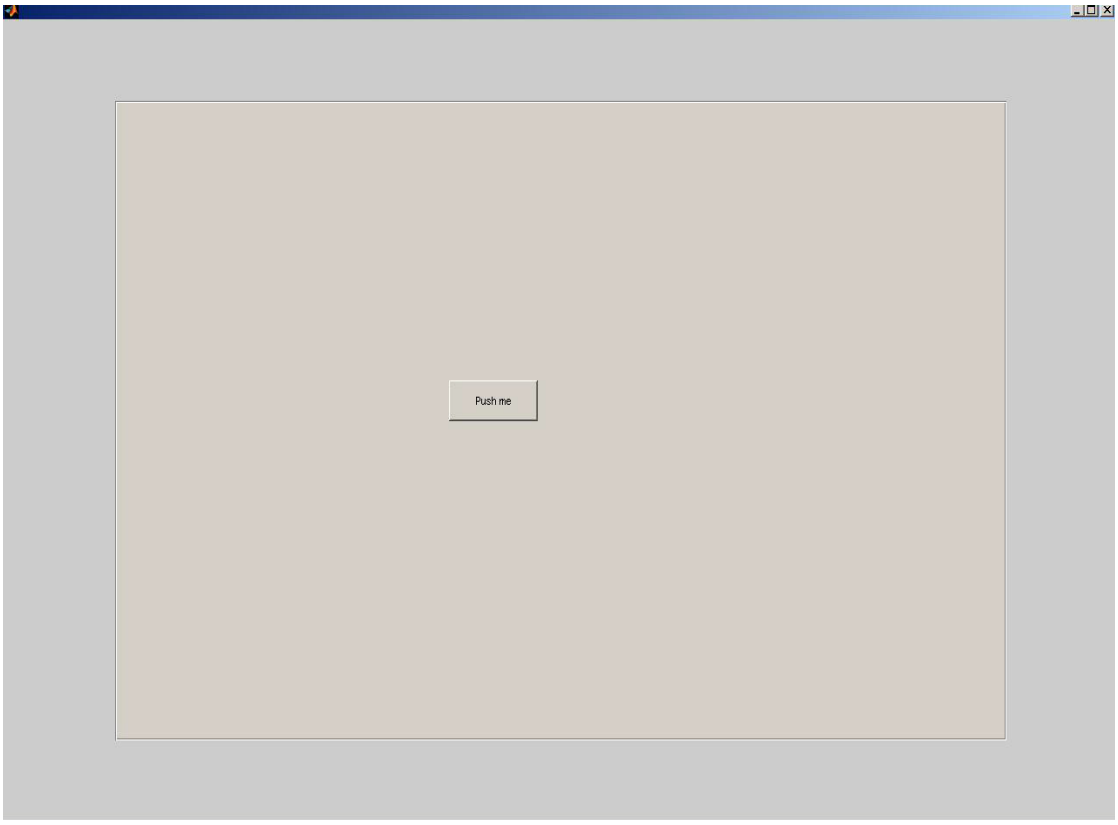

Figure 3.6 The GUI window created by the user. It contains one panel and one button. The callback for the button has not yet been written.

At this stage clicking the 'Push me' button will only result in a MATLAB error since the callback for the button has not yet been written and integrated into the MATLAB code.

To enter the callback code for the 'Push me' button, the user needs to go into the 'PanelandButtonCallbacks.m' file and enter the callback code in the callback framework for button1. The callback framework for button1 is as follows:

```
%Callback for the button1
function  button1Callback(h,eventdata)

%Enter user's callback code
end
```

The framework with the callback code for button 1 is as shown below.

```
function button1Callback(src,eventdata)
        msgbox('Hello World!','modal');
end
```

The GUI created by the user is referred to as ‘f’ as per the naming convention of the GUI Lite toolbox. The user can choose a name for the GUI ‘f’ using the ‘set’ function as shown below. This code snippet should be inserted just above the callback framework for button1 in the ‘PanelandButtonCallbacks.m’ file.

```
%set a name for the GUI
    set(f,'Name','Version 2-Hello World GUI');
```

Figure 3.7 displays the completed, fully functioning GUI. The complete code for this example is included in Appendix D.

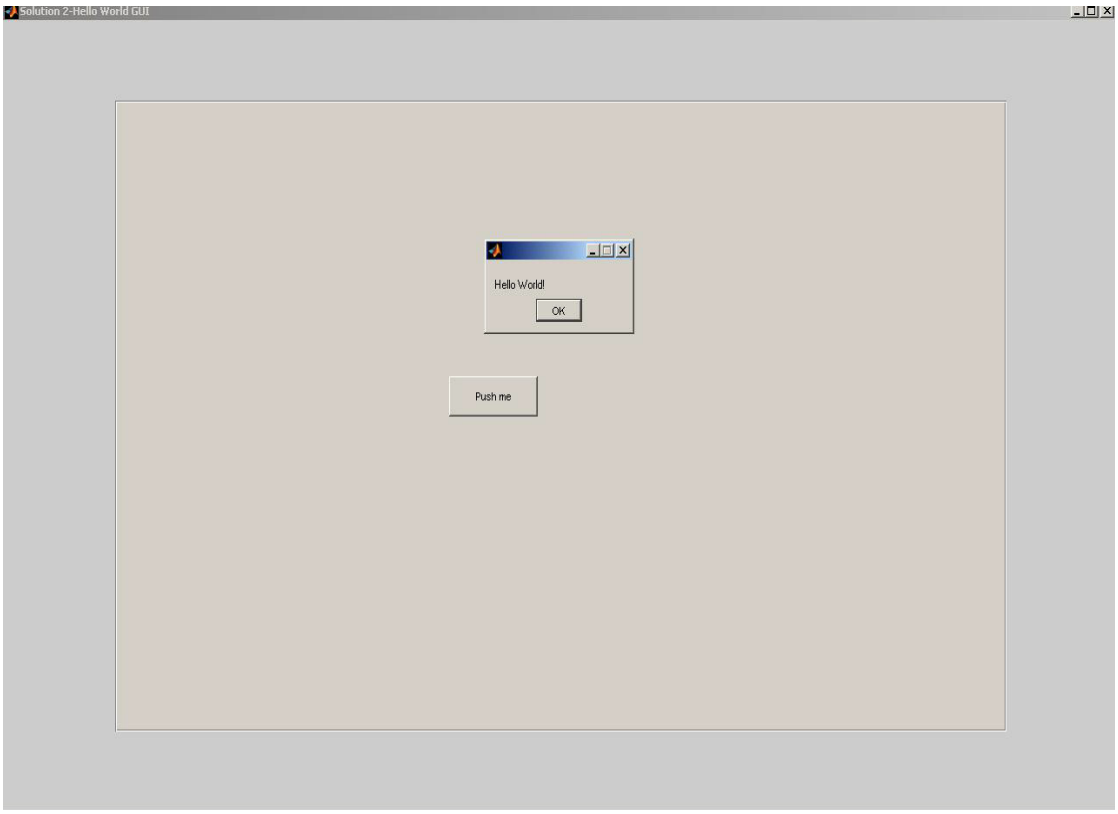

Figure 3.7 The completed and fully functioning 'Hello World GUI'. Clicking the 'Push me' button will display the 'Hello World!' message.

3.2.2 Program 2 - Display the waveform of a designated speech file.

Program 2 implements a GUI in which the click of a button displays the speech waveform of a designated speech file. Using GUI Lite – Version 2, the user should attempt to replicate the layout of the GUI created by GUI Lite – Version 1 for the same problem. The first step is to sketch the GUI and the various elements (panels, graphic panels, title boxes and buttons) included in this GUI. This GUI will have two panels, two buttons and one 'graphic panel' within which the speech waveform will be displayed. The two buttons that the user needs to create are the 'Display speech waveform' the 'Close GUI' 'pushbutton' buttons. Figure 3.8 displays a sketch of a possible layout for the Program 2 GUI.

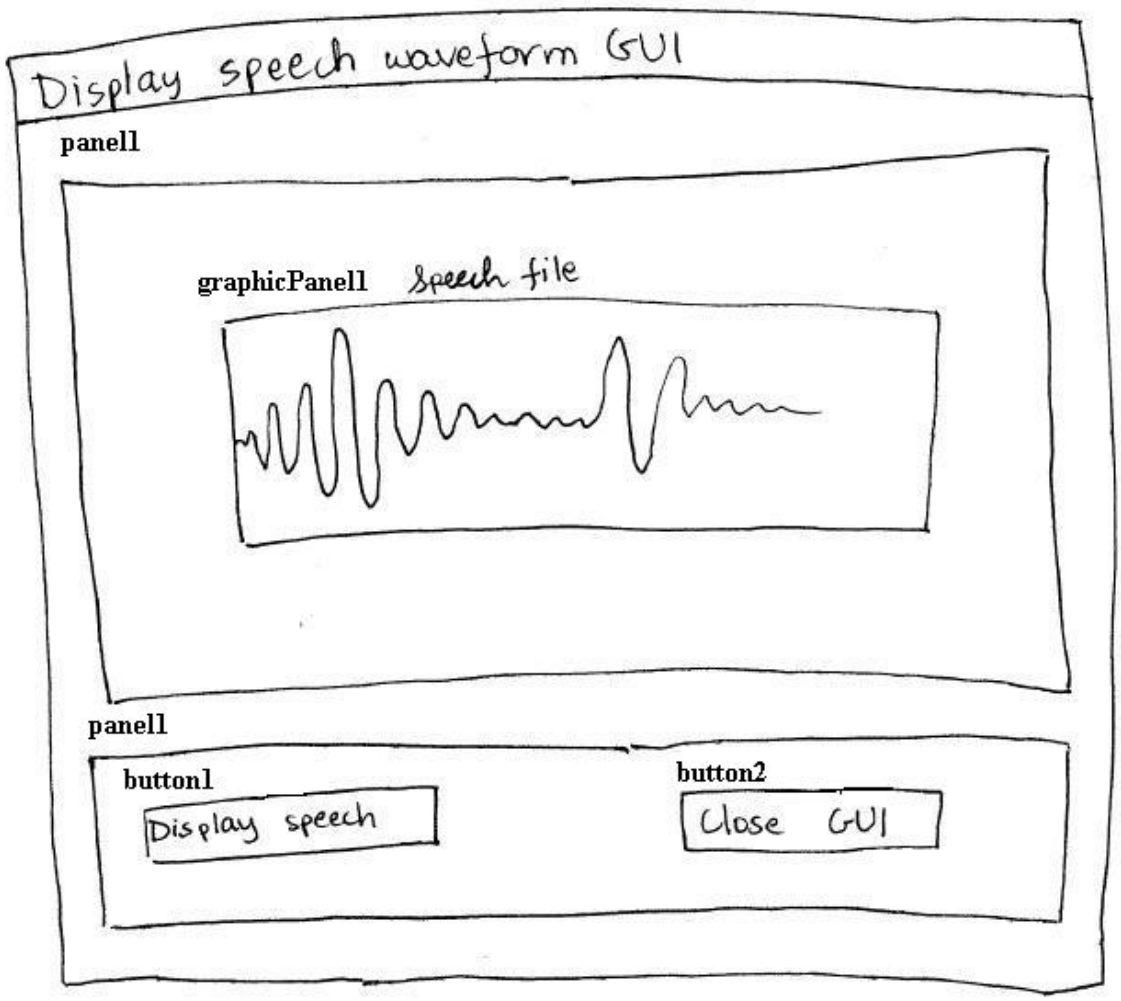

Figure 3.8 A sketch of the Program 2 GUI.

The user should now save the GUI Lite folder into the MATLAB work directory and run the 'panelButtonSetup.m' file. When the 'panelButtonSetup.m' file is run, an initial screen in which the GUI parameters can be entered is displayed. Figure 3.9 displays the GUI window with the parameters for Program 2 entered in it.

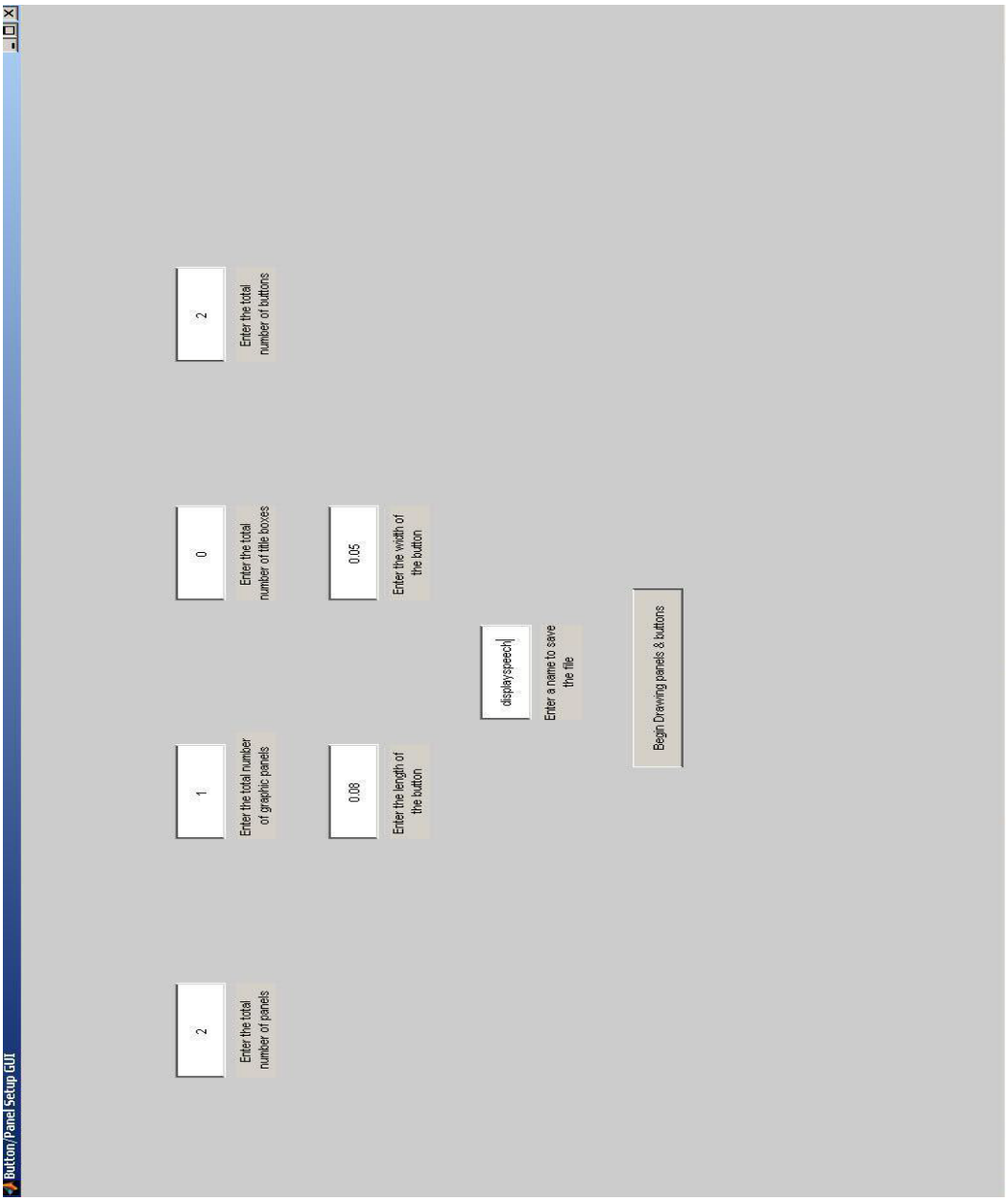

Figure 3.9 The GUI window with the parameters for the GUI elements for Program 2 entered into it.

Once the user has entered the number of GUI elements to create in the 'Button/Panel Setup GUI', the user should enter a filename for the GUI layout to be saved. In this example the name 'displayspeech' can be entered. On clicking the 'Begin Drawing buttons & panels' button, a white screen with crosshairs will be visible to the user. The user can select the co-ordinates of the two panels and a single graphic panel using these

crosshairs. Next, the user will be prompted so select the position of the ‘Display speech waveform’ button using the crosshairs. After selection of the position of the ‘Display speech waveform’ button on the white screen, another GUI, the ‘Button Stats’ GUI will be displayed on the screen. The user can use this GUI to enter the parameters of the ‘Display speech waveform’ ‘pushbutton’ button. The ‘Button Stats’ GUI with the parameters for the ‘Display speech waveform’ button entered into it is shown in Figure 3.10.

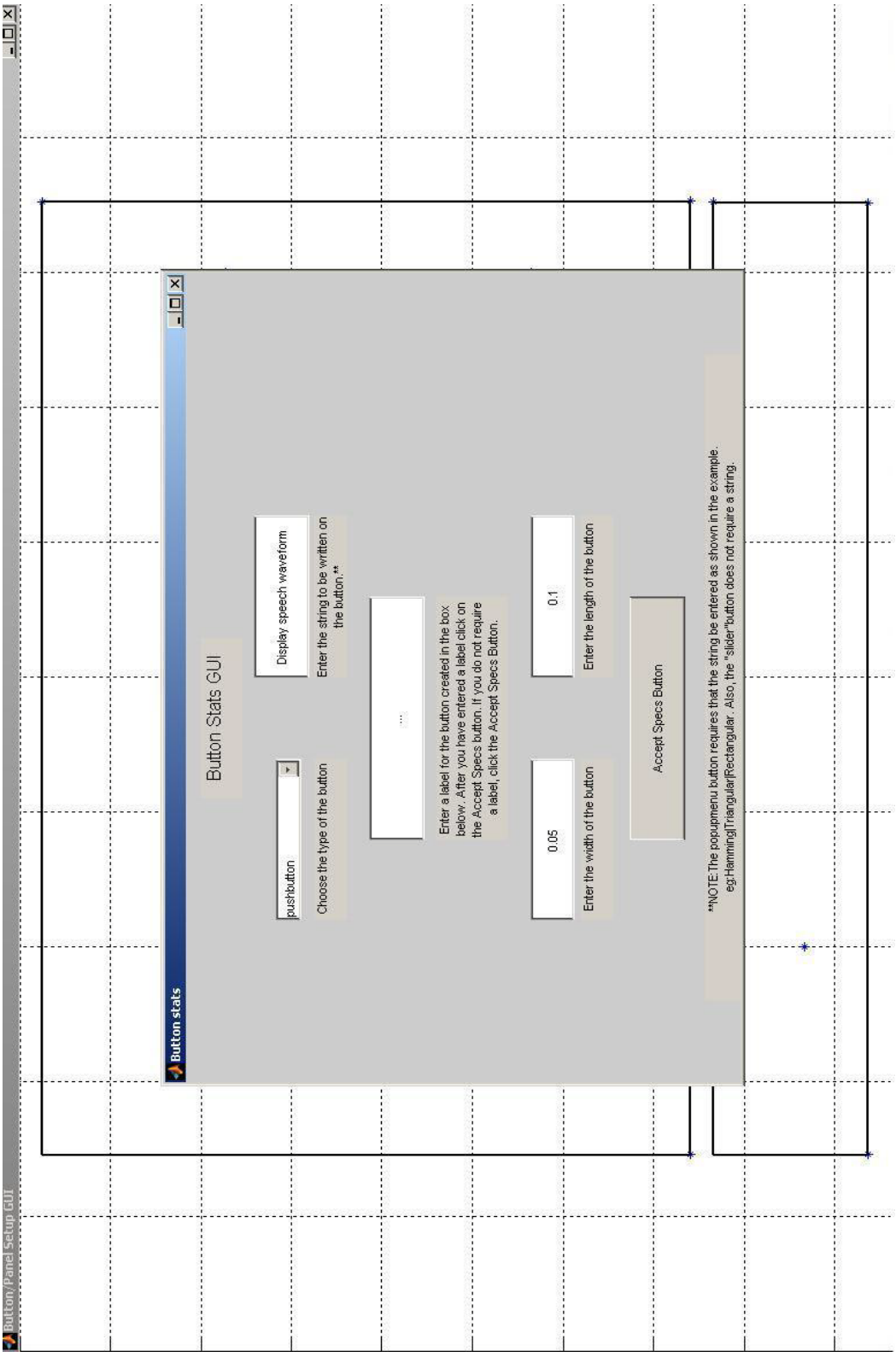

Figure 3.10 The 'Button Stats' GUI with the parameters for the 'Display speech waveform' entered into it.

After entering the parameters for the ‘Display speech waveform’, the user is prompted to select the position of the ‘Close GUI’ button. On selecting the position of the ‘Close GUI’ button, the ‘Button Stats’ GUI is again visible to the user. The user can leave the type of the button at its default value, i.e., ‘pushbutton’, and enter ‘Close GUI’ as the name of the button. The length of the button is increased to 0.1 units and the width is left at its default value. The user can then click the ‘Accept Specs Button’.

Since the positions of all the GUI elements have been selected, the user is now prompted to save the selection of GUI positions. Figure 3.11 displays the screen visible to the user when he is prompted to save his selection.

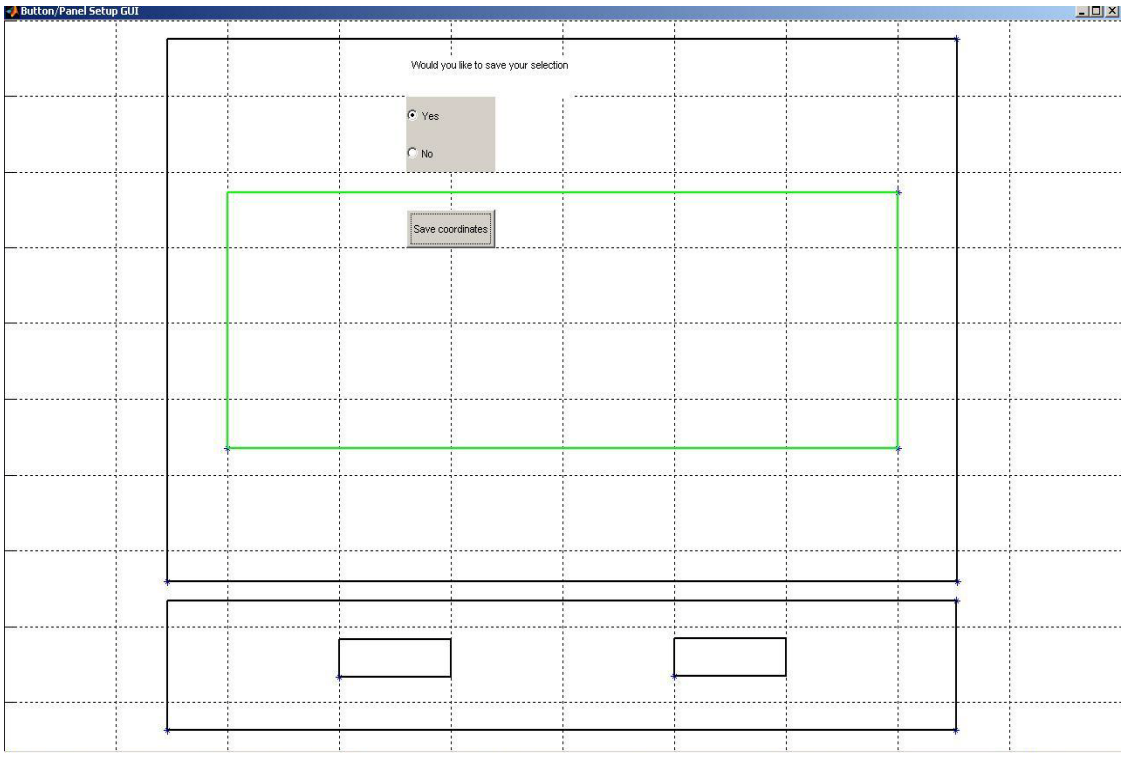

Figure 3.11 The user is prompted to save his selection of panels, graphic panel and button positions for Program 2. The large rectangles stacked over each other and outlined in black are the panels, the green rectangle is the graphic panel and the two small rectangles are the buttons.

In order for the user to see what the GUI looks like at this stage, the user can edit the name of the '.mat' file in the 'runGUI.m' file to 'displayspeech.mat', save the 'runGUI.m' file and run it. Figure 3.12 will be visible to the user on running the newly edited 'runGUI.m' file.

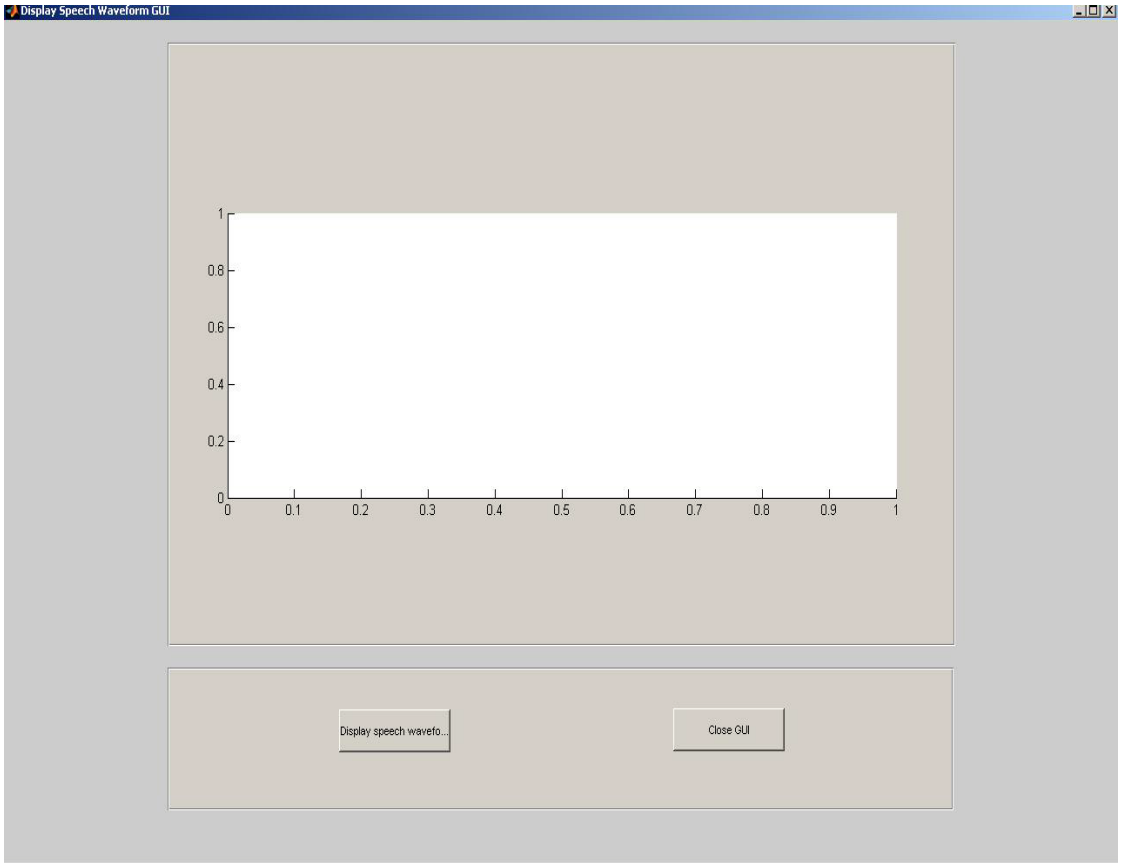

Figure 3.12 The 'Display Speech Waveform' GUI with the user defined panels, graphic panel and button. No callbacks are written for the buttons at this stage.

The next stage in the development of the GUI is writing callbacks for the buttons created by the user. Before writing callbacks for the user defined GUI elements, the user should name the GUI. The code snippet mentioned below is used in the naming of the GUI. This code snippet should be inserted below the comment 'USER CODE FOR THE VARIABLES, CALLBACKS AND INITIALIZATION'.

```
%set the name of the GUI
    set(f,'Name','Display Speech Waveform GUI');
```

To write the callback for the first button, the user enters the callback code for ‘button1’ within the pre-defined button framework for ‘button1’ in the ‘PanelandButtonCallbacks.m’ file. The callback code for ‘button1’ is as follows:

```
%button 1-Display speech callback
    function button1Callback(src,eventdata)
        loadedSpeech=wavread('s1.wav');
            %The speech file is 's1.wav'
            axes(graphicPanel1);
            plot(loadedSpeech);
            title('s1.wav');
            xlabel('Time in seconds');
            ylabel('Amplitude');

    end
```

The ‘button1Callback’ function loads the ‘s1.wav’ file using the ‘wavread’ function and displays it on ‘graphicPanel1’. The waveform is titled ‘s1.wav’.

The callback code for the ‘Close GUI’ button is as follows:

```
%button2-Close callback
    function button2Callback(src,eventdata)
       close(gcf);
    end
```

The ‘button2Callback’ function closes the current GUI window. Figure 3.13 displays the completed and fully functioning ‘Display Speech Waveform’ GUI. The complete code for Program 3 has been included in Appendix D.

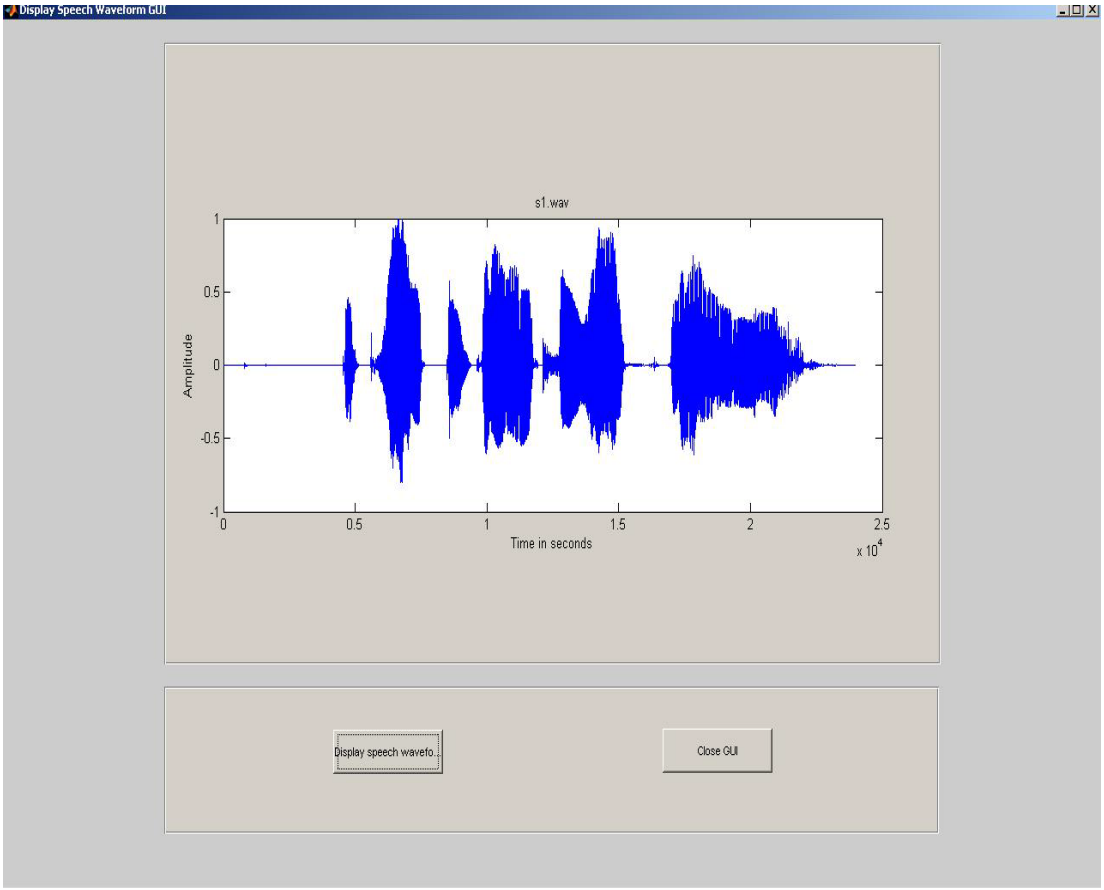

Figure 3.13 The completed and fully functioning 'Display Speech Waveform' GUI with 's1.wav' displayed in 'graphicPanel1'.

3.2.3 Program 3 - Load a Speech File, Play it back and Display the Waveform.

Program 3 is an extension of Program 2. In Program 3, the user is able to browse the file system and select a particular directory. The user then loads a selected '.wav' speech file from the selected directory, and can then play and display a waveform of the selected file.

The first step in the GUI development for Program 3 is to sketch the GUI and the various elements (panels, graphic panels, title boxes and buttons) included in this GUI. This GUI will have two panels, five buttons and one 'graphic panel' within which the

image will be displayed. The five buttons that the user creates are the ‘Get directory’ ‘pushbutton’ button, the ‘Select file’ ‘popupmenu’ button and the ‘Play’, ‘Plot’ and ‘Close GUI’ ‘pushbutton’ buttons. The user can also use a ‘titleBox’ to display the filename of the speech waveform being displayed in the ‘graphicPanel’. Figure 3.14 shows a sketch of the desired GUI.

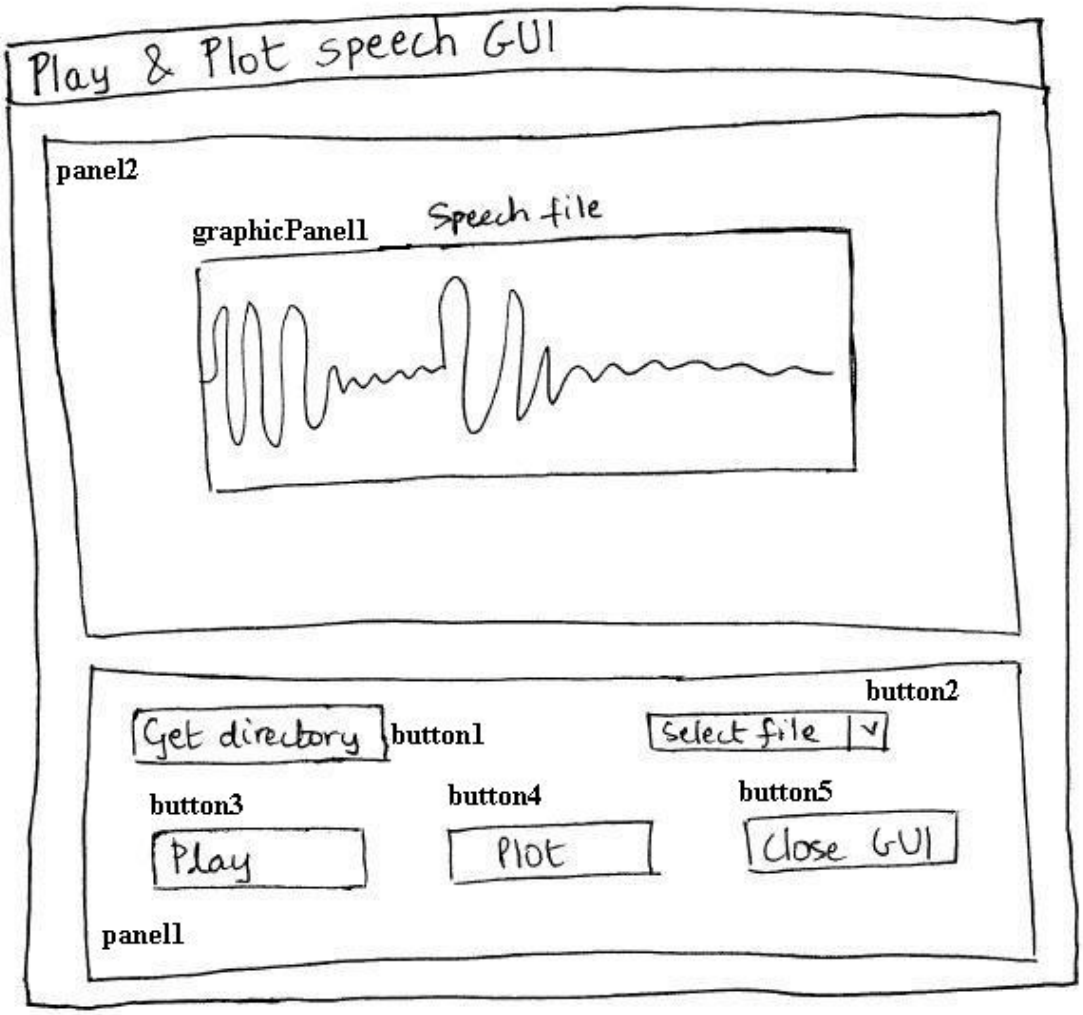

Figure 3.14 The tentative layout of the GUI for Program 3 – Play and Plot Speech GUI on paper.

Once the user has saved the GUI Lite folder into his ‘work’ folder, the GUI development can begin. On running, the ‘panelButtonSetup.m’ file, the user can enter the number of panels as two, the number of graphic panels and title boxes as one and the number of buttons as five. The filename to be entered to save the selection of panel and button positions and dimensions is ‘play&plot’.

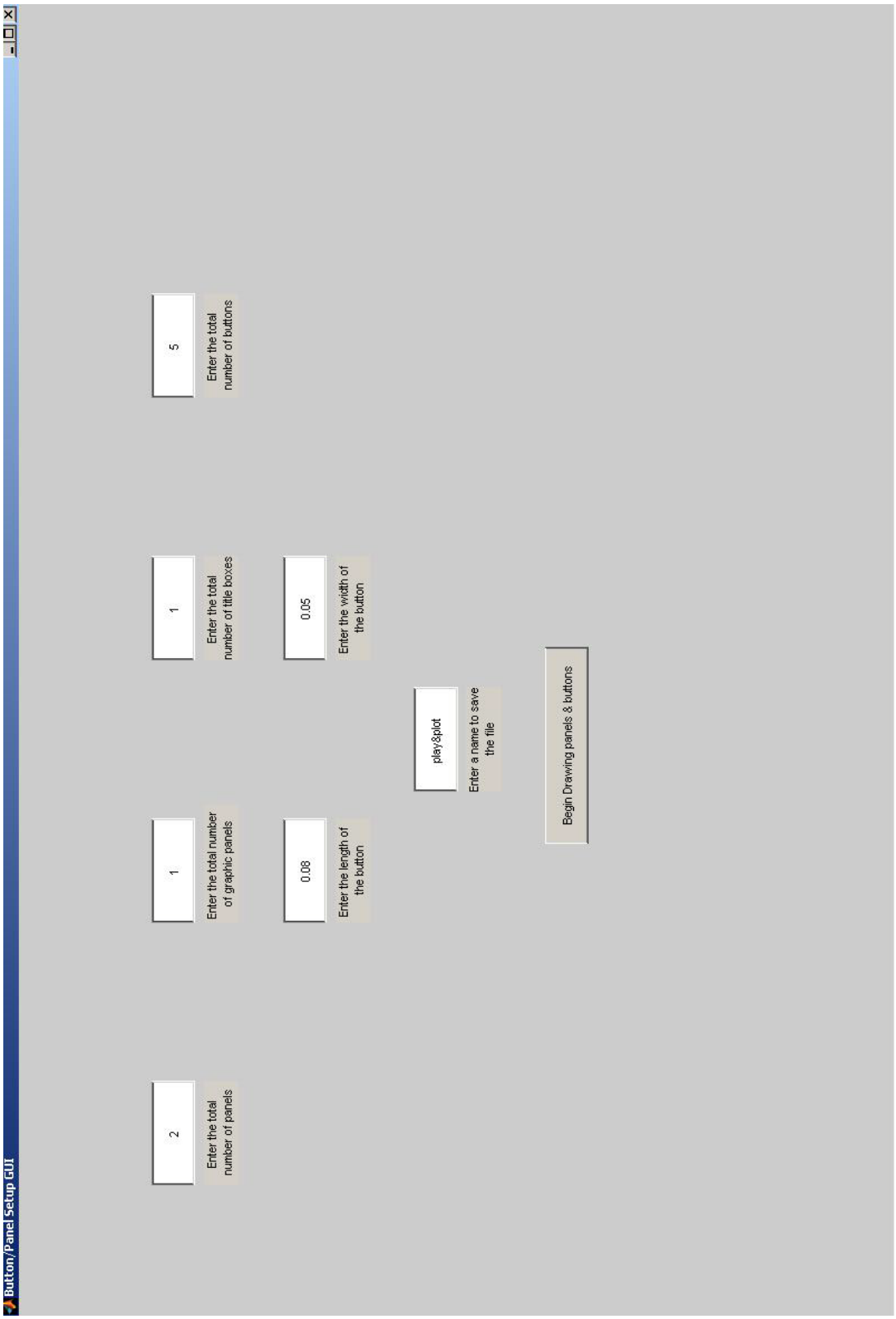

Figure 3.15 The 'Button/Panel Setup GUI' with the user parameters for the various GUI elements.

On clicking the ‘Begin Drawing panels and buttons’ button, crosshairs will be visible on the screen, from which the user can select the endpoints of the panels, the graphic panel and the title box for Program 3. After selecting the position of the first button, the user is prompted to select and enter parameters for that button. The user is then prompted to select the position of the next button and to enter the parameters for that button. The user needs to follow the same procedure while creating the last three buttons for Program 3. For the ‘Get directory’ button, the user should leave the type of button at its default value i.e., ‘pushbutton’, the name of the button can be entered as ‘Get directory/Select file’ and the length of the button should be increased to 0.15 units to make it easier to read the button name. For the ‘Select file’ ‘popupmenu’, the user can select the type of the button as ‘popupmenu’, the name of the button as ‘Select file’ and again the length of the button should be increased to 0.15 units. For the ‘Play’, ‘Plot’ and ‘Close GUI’ buttons, the type of the button should be left at its default value, i.e., ‘pushbutton’ and the name for the buttons should be entered as ‘Play’, ‘Plot’ and ‘Close GUI’ respectively. The lengths of the ‘Play’, ‘Plot’ and ‘Close GUI’ buttons should be increased to 0.15 units. Figures 3.16 and 3.17 show the screens visible to the user while creating the ‘Plot’ and ‘Close GUI’ buttons respectively.

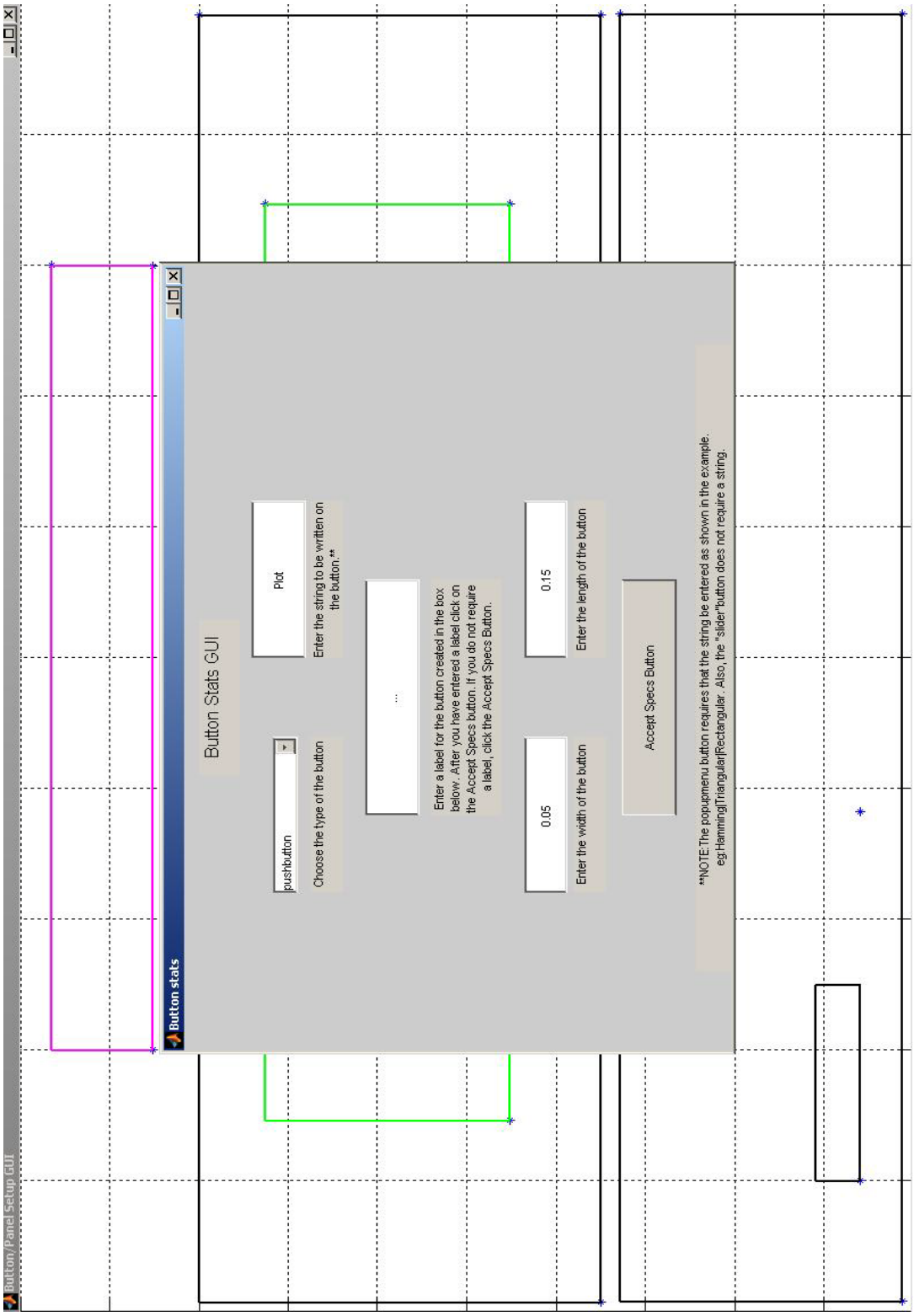

Figure 3.16 This is the 'Button Stats' GUI after the user has entered the parameters for the 'Plot' button. The type of button is 'pushbutton'. The string to be written on the button is 'Plot'. The length of the button has been changed to 0.15units.

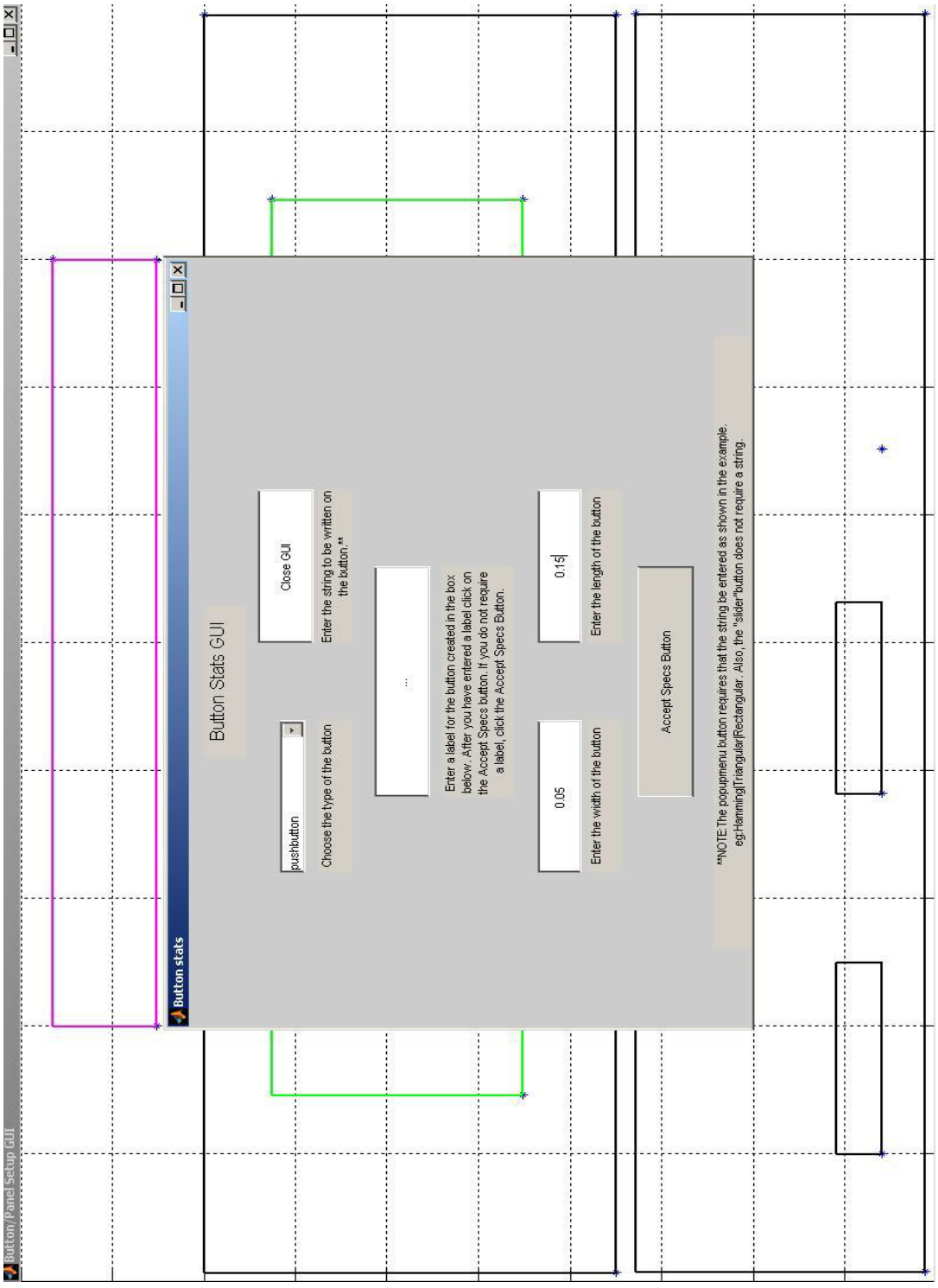

Figure 3.17 This is the 'Button Stats' GUI after the user has entered the parameters for the 'Close GUI' button. The type of button is 'pushbutton'. The string to be written on the button is 'Close GUI'. The length and width have been left at their default values.

After all five buttons for Program 3 have been created, the user is prompted to save the selection. Figure 3.18 displays the GUI window after the user has selected the positions of the panels, the graphic panel, the title box and the buttons.

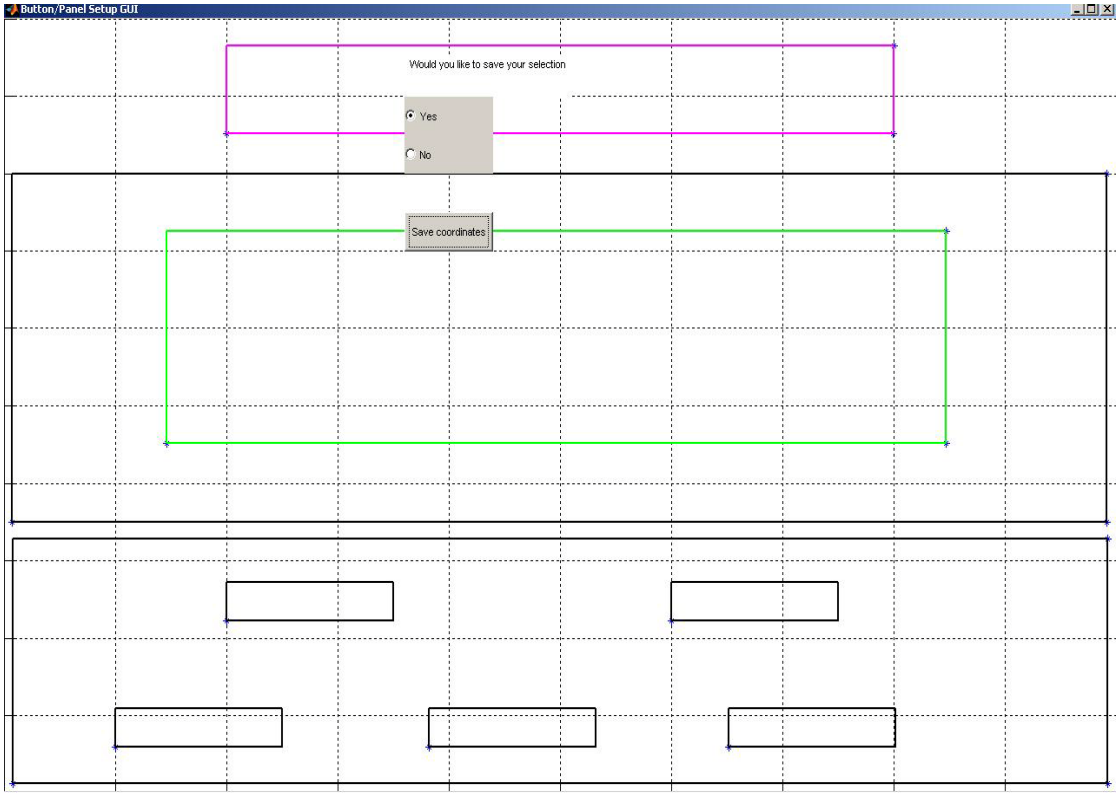

Figure 3.18 This is the screen visible to the user after the panels, graphic panels and buttons have been created. The two large rectangles stacked above each other represent the panels. The green and the pink rectangle represent the graphic panel and the title box. The five small rectangles are the buttons.

To view how the GUI will look at this stage, the user can edit the '.mat' filename from 'runGUI.m' to 'play&plot.mat' and save and run the 'runGUI.m' file. The screen in Figure 3.19 will be visible to the user. At this stage none of the buttons work and clicking on then would cause errors since the callbacks for them have not yet been written.

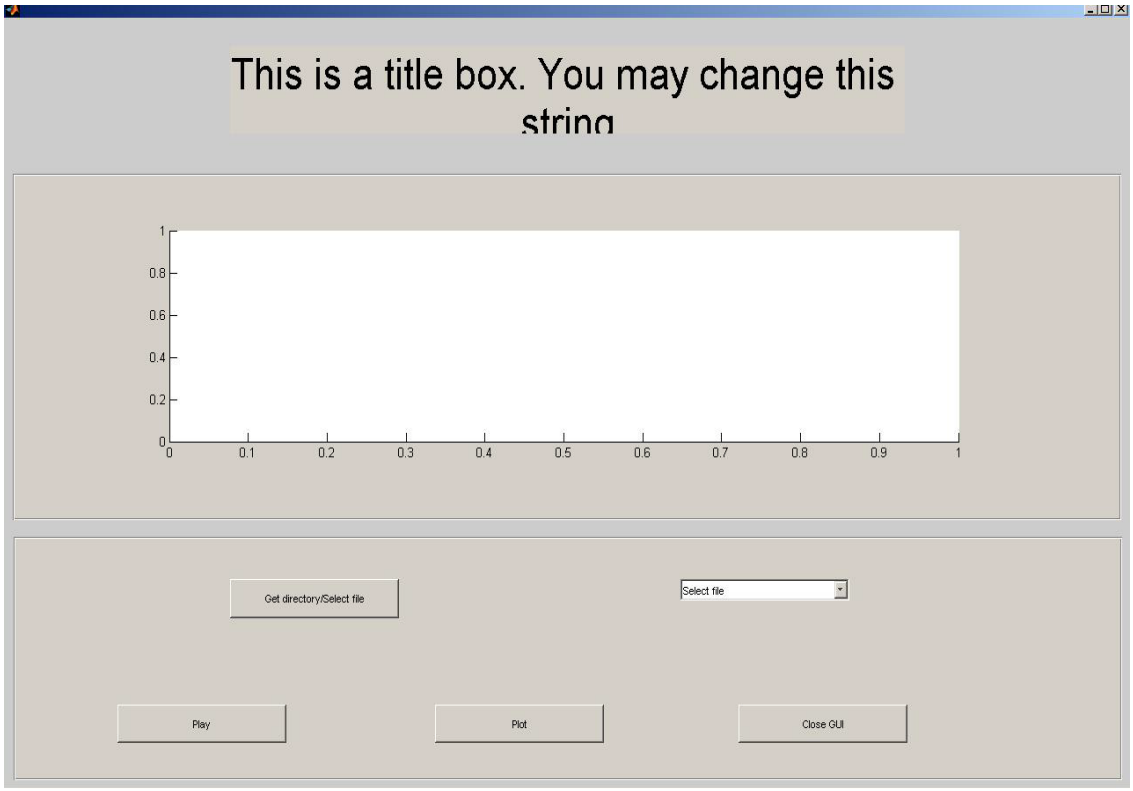

Figure 3.19 This screen displays the user's GUI with the panels, graphic panel, title box and buttons for Program 3. The buttons do not work at this point since the callbacks for the buttons have not been written.

In order to write the callbacks, the user must insert the callback code [9] into the callback frameworks already defined in the 'PanelandButtonCallbacks.m' file.

The user can name the GUI using the 'set' function included in MATLAB. The code to set the name of the GUI is given below:

```
%set the name of the GUI
    set(f,'Name','Play and Plot Speech GUI');
```

The callbacks for the 'Get directory' and 'Select file' button are as follows:

```
     function button1Callback(src,eventdata)

        directory_name = uigetdir('start_path','dialog_title');
```

```
        A=strvcat(strcat((directory_name),'\*.wav'));
        struct_filenames=dir(A);
        wav_file_names={struct_filenames.name};
        set(button2,'String',wav_file_names);
        %once the popupmenu/drop down menu is created, by default, the
first
        %selection from the popupmenu/drop down menu must be loaded
even if the
        %callback for the popupmenu/drop down menu id not called

        indexOfDrpDwnMenu=1;%by default first option from the
popupmenu/dropdown
                          %menu will be loaded
        [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu);


    end
    function button2Callback(src,eventdata)
        indexOfDrpDwnMenu=get(button2,'val');
        [curr_file,fs]=loadSelection(directory_name,...
            wav_file_names,indexOfDrpDwnMenu);
    end
```

The ‘loadSelection’ function is not a part of the GUI Lite toolbox and should be a user defined function. Its purpose is to load the default value of the ‘popupmenu’ as the current speech file to be played once the ‘Select file’ button is populated. The user can use this function or write another one.

```
    %Function--load selection
    function [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu)
        fin_path=strcat(directory_name,'\',...
            strvcat(wav_file_names(indexOfDrpDwnMenu)));
            %fin_path is the complete path of the file .wav file that
is
            %selected
        clear curr_file;
```

```
        clear fs;
        [curr_file, fs]=wavread(fin_path);
        FS=num2str(fs);
        %Information about the file being played
        file_info_string=strcat('Current file = ',...
            wav_file_names(indexOfDrpDwnMenu),...
            '. Sampling frequency = ',FS,'Hz',...
            '. Number of samples in file = ',...
            num2str(length(curr_file)));
        set(titleBox1,'String',file_info_string);
        set(titleBox1,'FontSize',0.3);
    end
```

In the 'loadSelection' function above, the user sets the 'string' attribute of the 'titleBox1' object to the string in the variable 'file_info_string' so that the title box will reflect the contents of the 'file_info_string' variable. The 'file_info_string' variable contains information regarding the file currently being played. The above 'button1Callback', the 'button2Callback' and the 'loadSelection' functions require some variables to be shared amongst them. In GUI Lite, the user can share variables among all the callback functions by initializing the variables before the callback functions are written. They variables to be share are initialized to dummy values just below the comment that says 'USER CODE FOR THE VARIABLES, CALLBACKS AND INITIALIZATION'. The code for initializing the variables is given below:

```
curr_file=1;
fs=1;
directory_name='abcd';
wav_file_names='abcd';
```

The next callback to be written is the callback for button3 i.e., the 'Display speech Waveform' button. This button displays the speech waveform of the selected '.wav' file in the graphic panel. Since there is only one graphic panel created, it is referred to as

'graphicPanel1' as per the naming convention of the GUI Lite toolbox. The callbacks for button3 and button4 are provided below. 'button3Callback' displays 's4.wav' and also creates the title 's4.wav'. 'button4Callback' closes the current GUI window.

```
%Callback for the playbutton
    function  button3Callback(h,eventdata)
        sound(curr_file,fs);
    end
    %callback for the plotbutton
     function button4Callback(h,eventdata)
        hold off;  %It is essential to turn hold off so that the
        hold off;  %earlier contents of the panel are replaced
        %the two hold off's are for the speech file and the hamming
window

        grid off;
        reset(graphicPanel1);
        axes(graphicPanel1);
        l=length(curr_file);
        i=(1:l)/fs;%coverting samples to time
        plot(i,curr_file,'k','LineWidth',2),...
            xlabel('Time in seconds'),...
            ylabel('Amplitude');
        axis tight;
        grid on;
     end
%callback for the close GUI button
     function button5Callback(h,eventdata)
       close(gcf);
    end
```

The completed and fully functioning GUI is displayed in Figure 3.20.

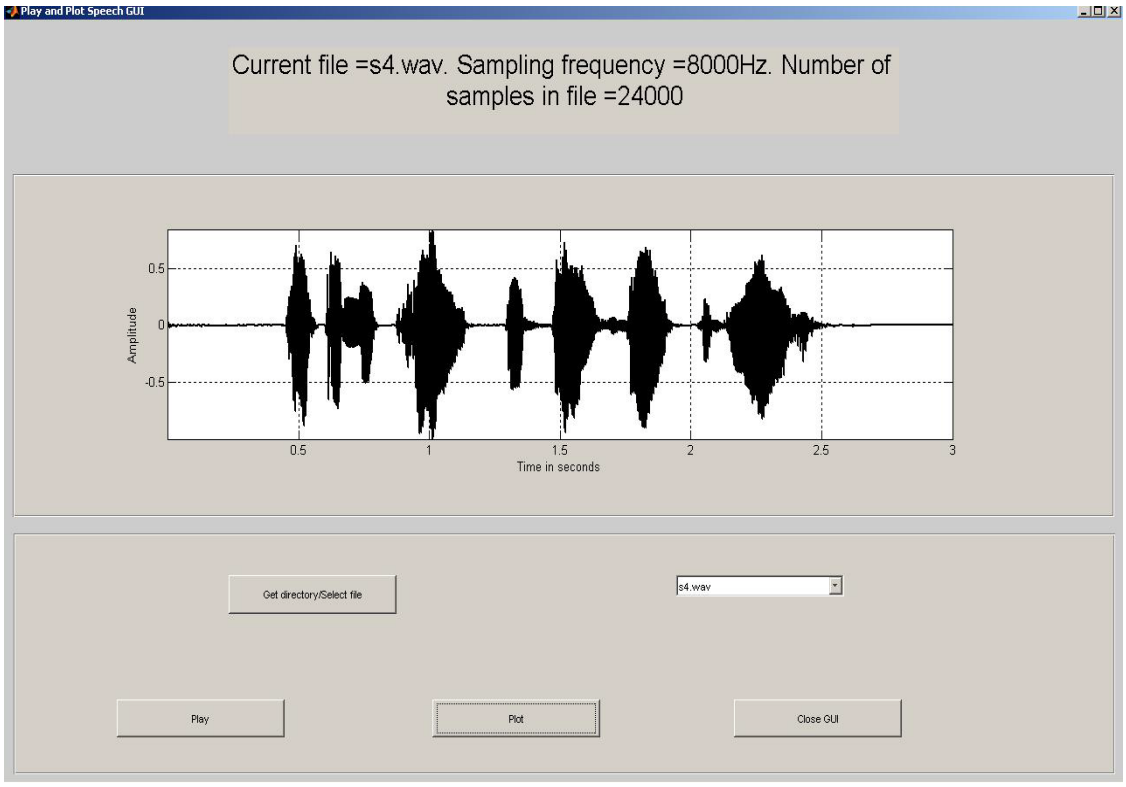

Figure 3.20 This is the completed 'Display Image GU I'. The file to be displayed is selected using the 'Get directory' and 'Select file' buttons. Clicking the 'Display image' button displays the desired speech file in 'graphicPanel1'.

3.2.4 Program 4 - Load an Existing Speech File or Record a New Speech File. Play the File and Save the File.

For this example, we will try to replicate the look of the GUI created by Version 1 for the same problem. As done in the previous examples, the user first needs to visualize and sketch the GUI to determine which GUI elements need to be included. Figure 3.21 displays a sketch of the proposed layout for the 'Record Speech GUI'.

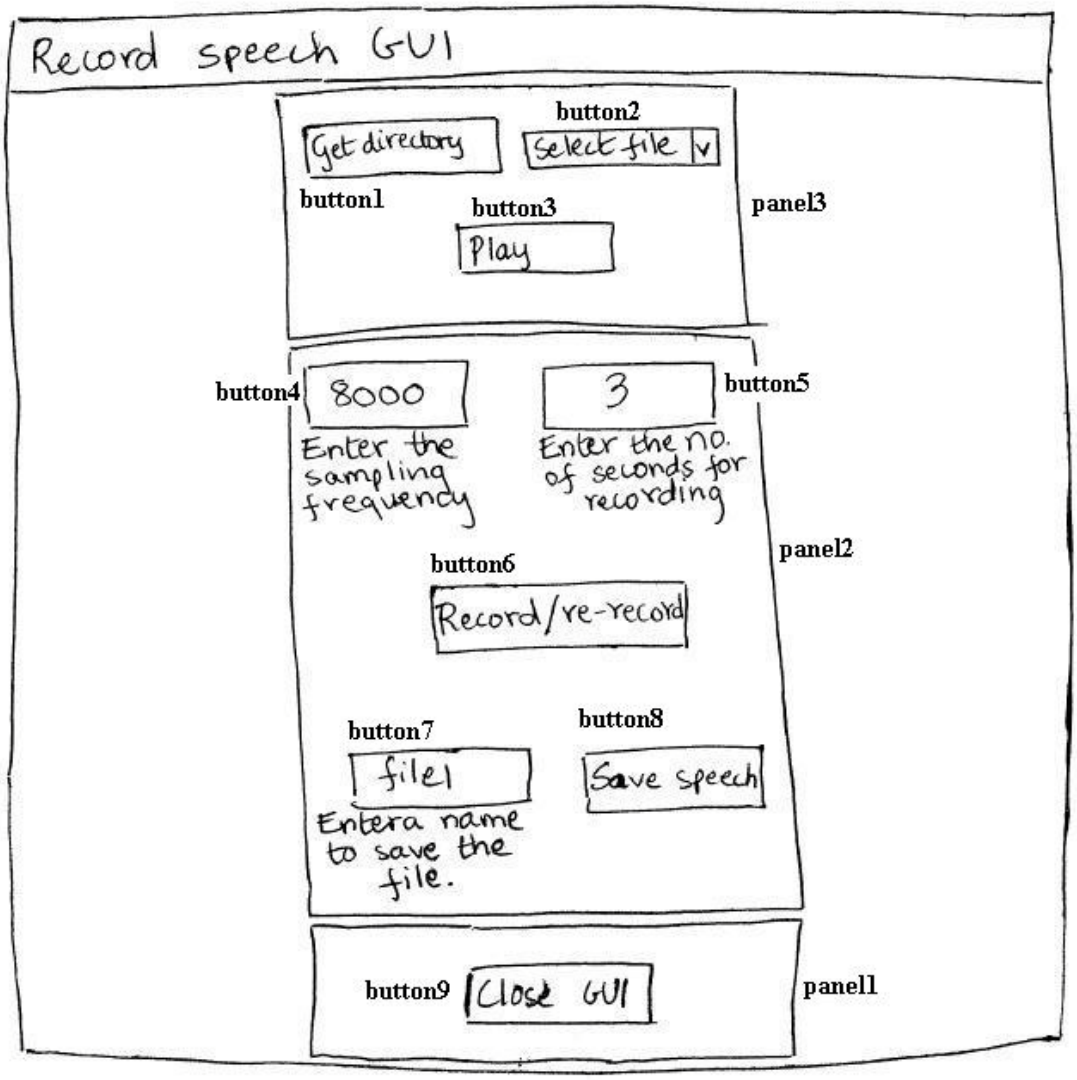

Figure 3.21 A sketch of the layout of the 'Record Speech GUI'.

The 'Get directory' 'pushbutton', the 'Select file' 'popupmenu' and the 'Play' 'pushbutton' buttons are grouped together in the first panel i.e., 'Panel1'. The 'Enter the sampling frequency' and 'Enter the number of seconds for recording' 'edit' buttons along with the 'Record/re-reccord' 'pushbutton', the 'Enter a filename to save the recorded speech' 'edit' button and the 'Save speech' 'pushbutton' are grouped together into the second panel i.e., 'Panel2'. The 'Close GUI' 'pushbutton' is in 'Panel3'. This

grouping of buttons into different panels is done to minimize errors caused by clicking buttons unrelated to each other consecutively. Grouping related buttons together reduces errors caused by inter-button dependencies.

Once a tentative sketch of the GUI is available, the user can save the GUI Lite folder in the ‘work’ folder and run the ‘panelButtonSetup.m’ file. In the initial screen that appears on running ‘panelButtonSetup.m’, the user enters the number of panels as three, the number of graphic panels and title boxes as zero and finally the number of buttons as nine as shown in Figure 3.22. For this example, ‘recordgui’ is entered as the filename that is used to save the GUI layout.

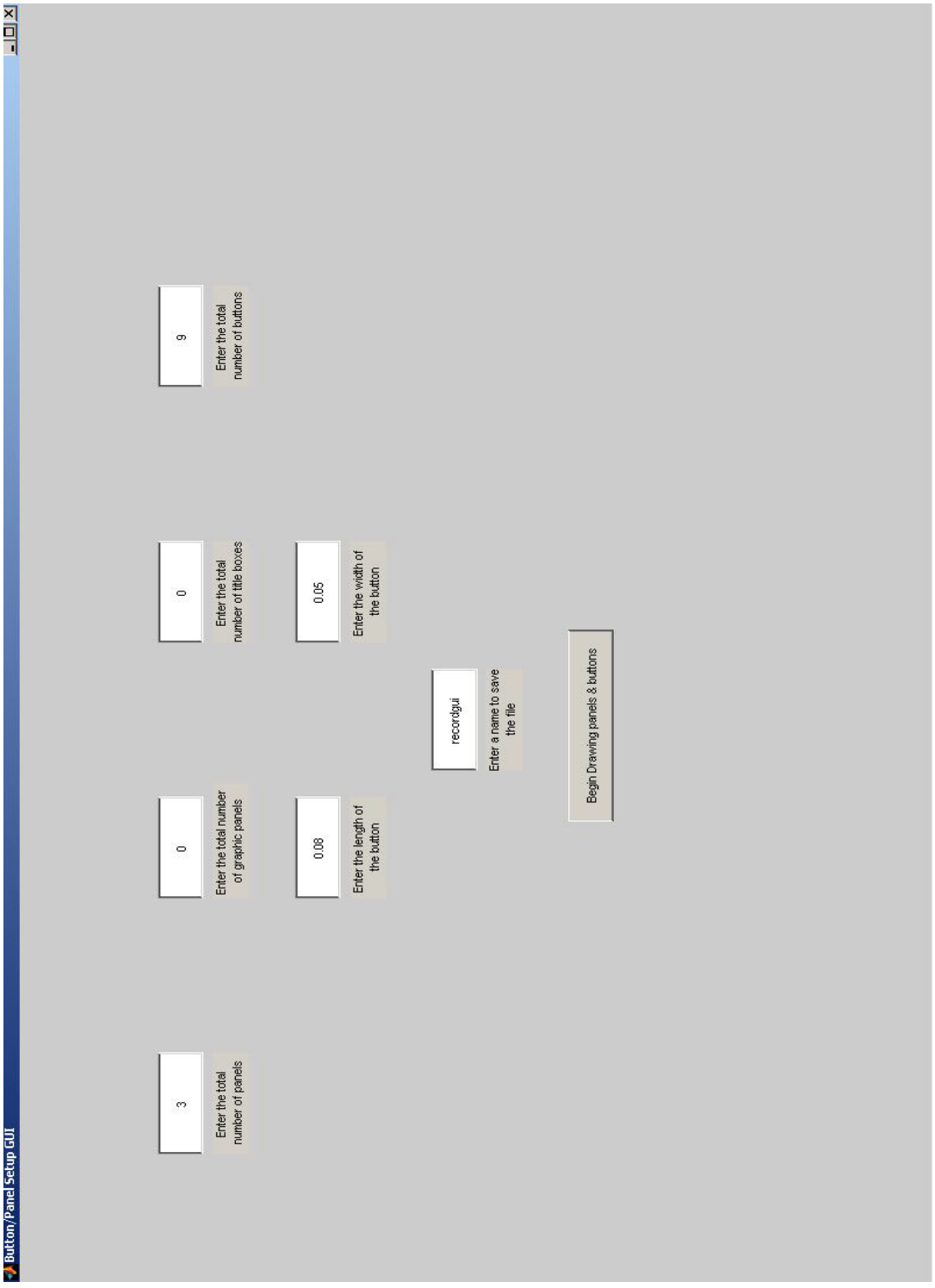

Figure 3.22 The GUI screen that is visible to the user after the user has run the 'panelButtonSetup.m' file and entered all the user parameters into the GUI.

After the user parameters for the GUI are entered and the ‘Begin drawing panels and buttons’ button is clicked, the user will be asked to select the positions of the three panels. Once the panels are defined, the user will be asked to create the buttons by selecting their positions and entering their parameters. Screenshots of some of the buttons being created are included below as Figures 3.23, 3.24, 3.25, 3.26 and 3.27.

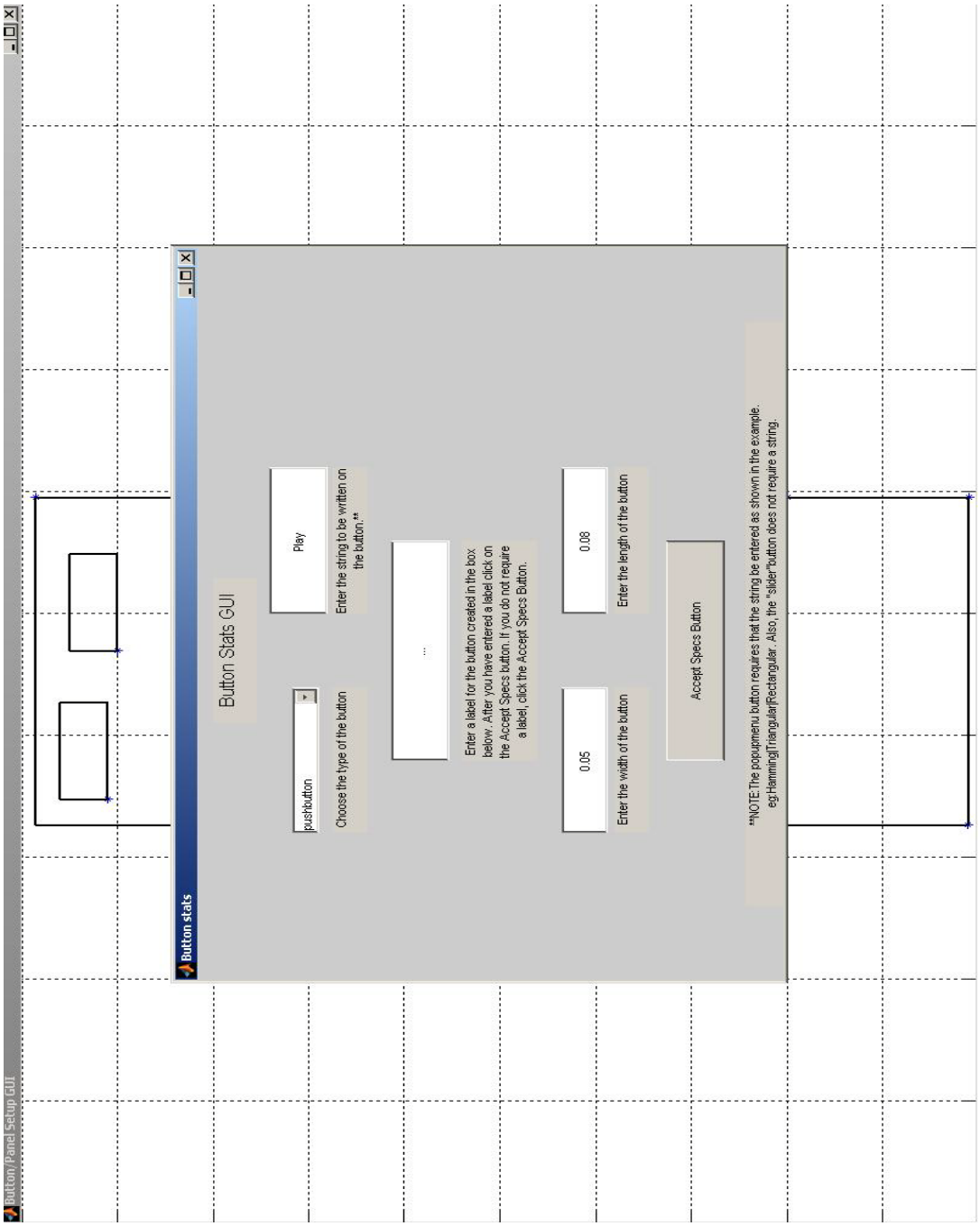

Figure 3.23 The 'Button Stats' GUI after the user has entered the parameters for the 'Play' button. The type of button is 'pushbutton'. The string to be written on the button is 'Play'. The length and width of the button have been left at their default values.

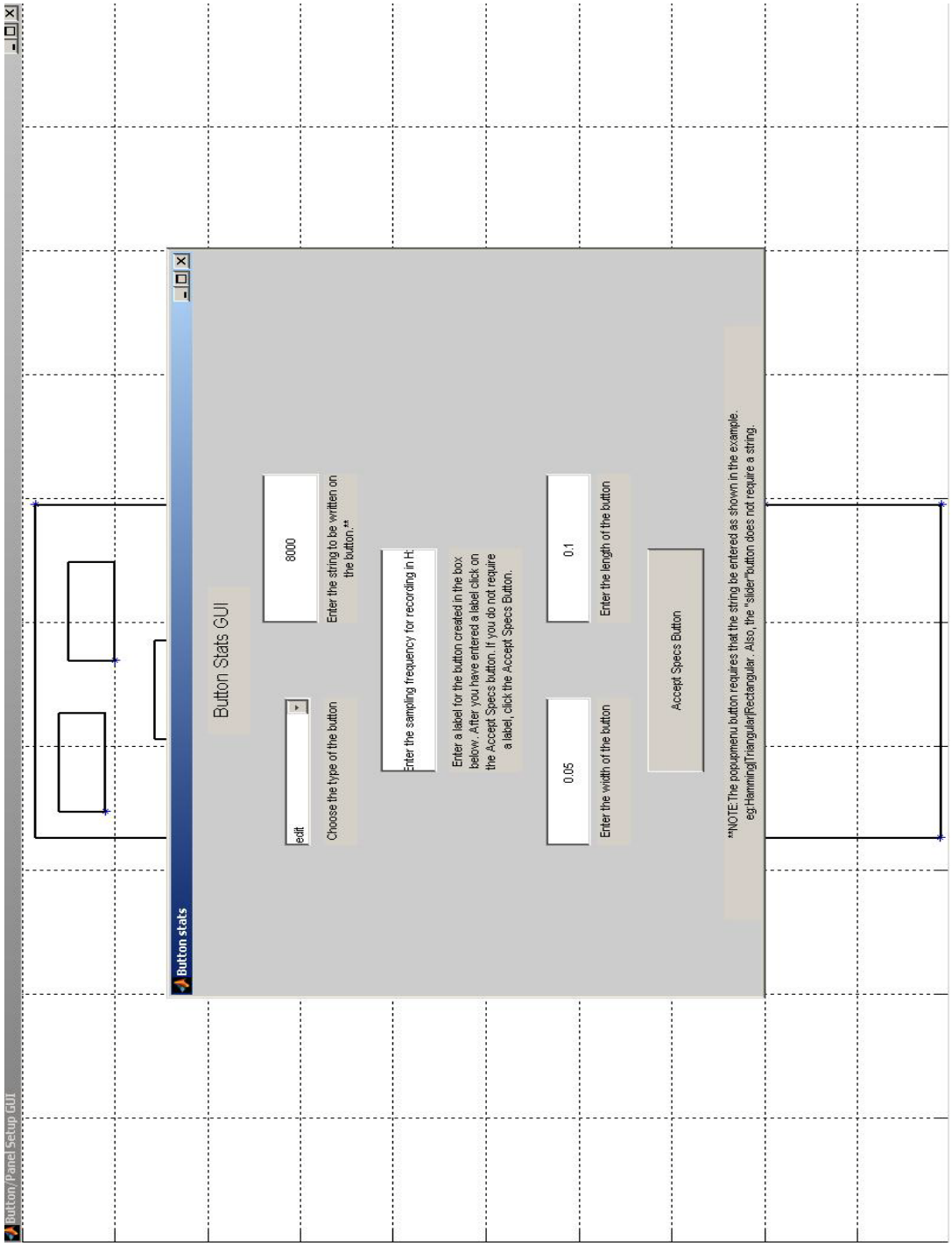

Figure 3.24 The 'Button Stats' GUI after the user has entered the parameters for the 'Enter the sampling frequency in Hz' button. The type of button is 'edit'. The string to be written on the button is 'Enter the sampling frequency in Hz'. The length has been changed to 0.1 units while the width has been left at its default value.

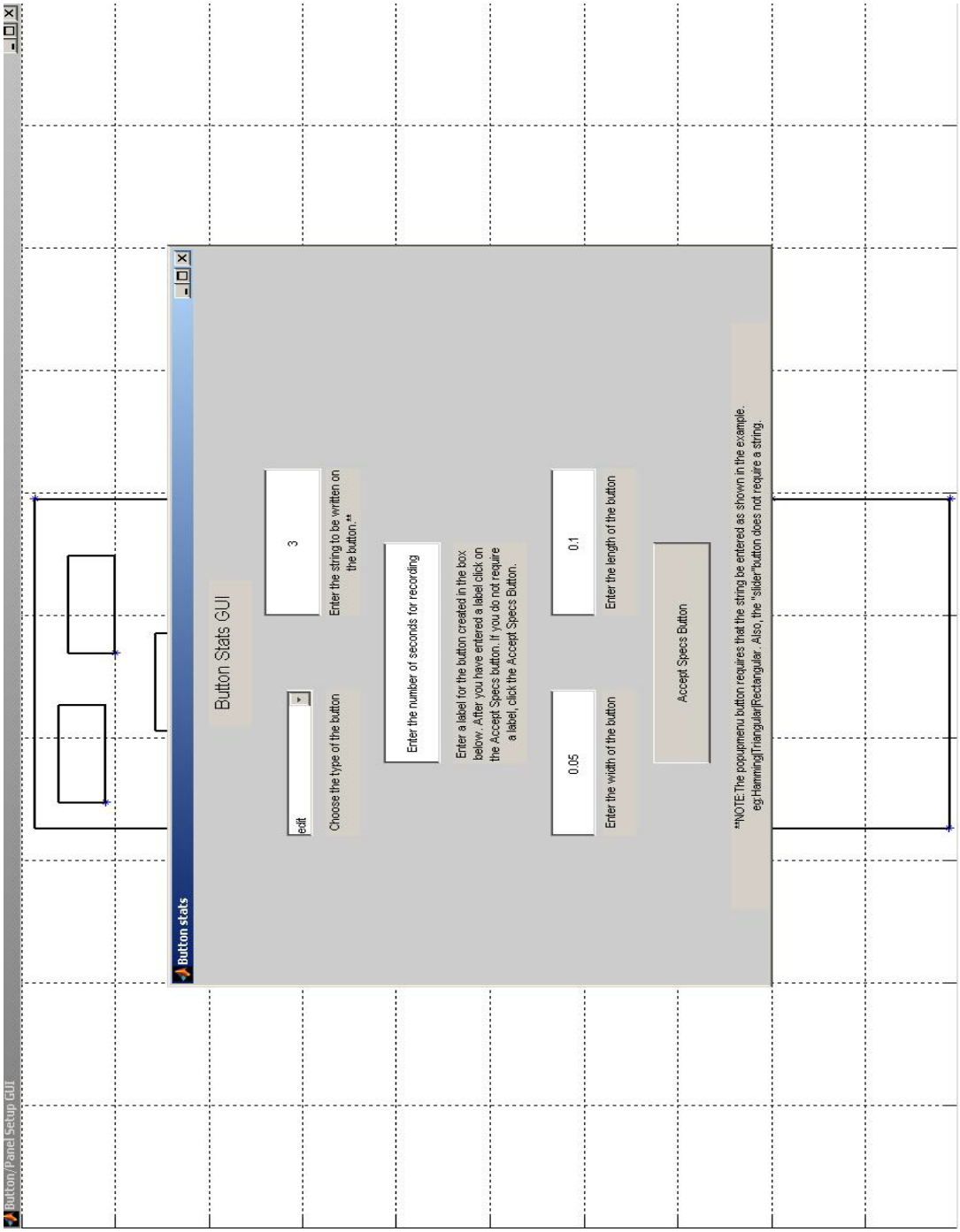

Figure 3.25 The 'Button Stats' GUI after the user has entered the parameters for the 'Enter the number of seconds for recording' button. The type of button is 'edit'. The string to be written on the button is 'Enter the number of seconds for recording'. The length has been changed to 0.1 units while the width has been left at its default value.

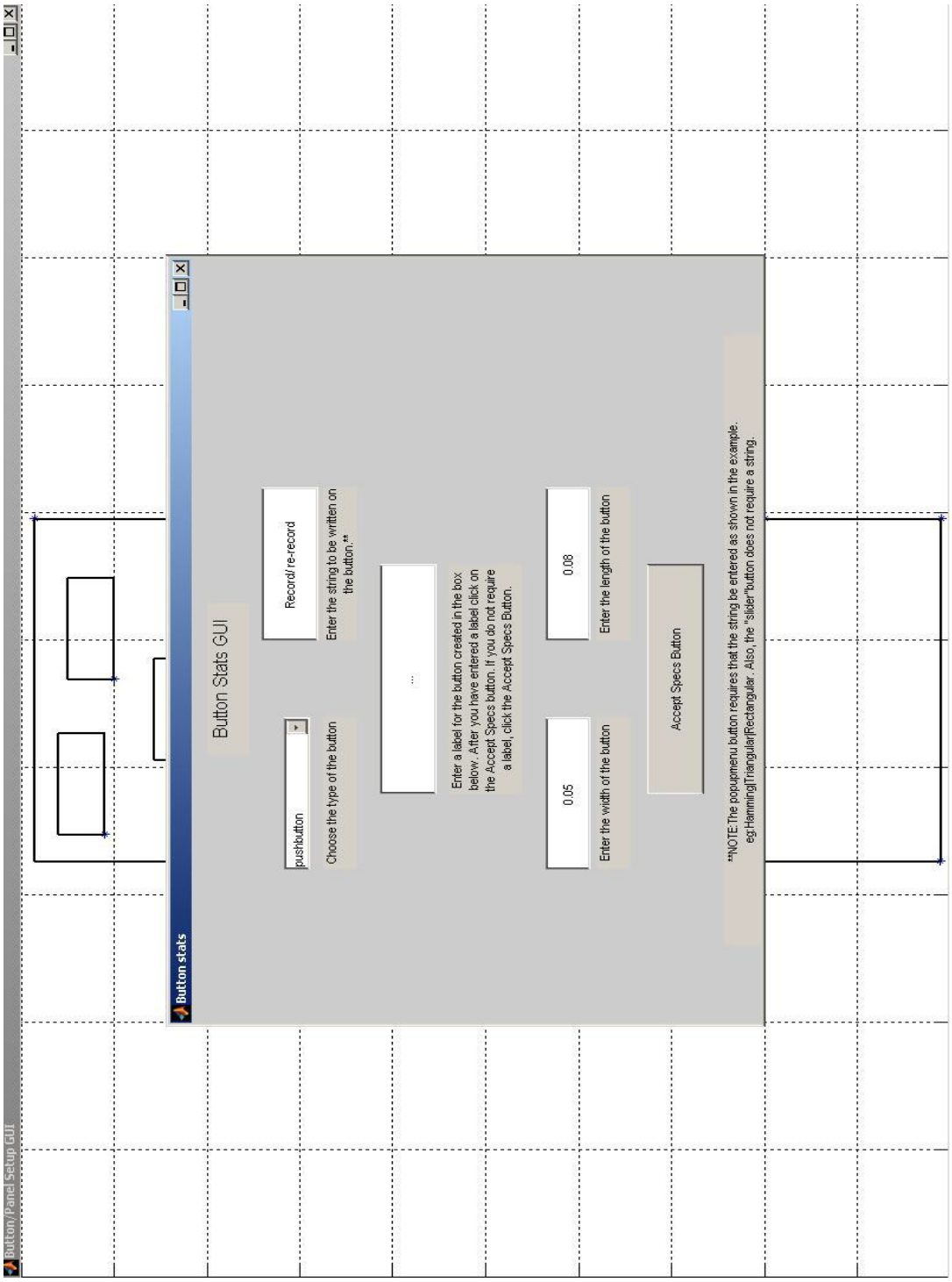

Figure 3.26 The 'Button Stats' GUI after the user has entered the parameters for the 'Record/re-record' button. The type of button is 'pushbutton'. The string to be written on the button is 'Record/re-record'. The length and width have been left at their default values.

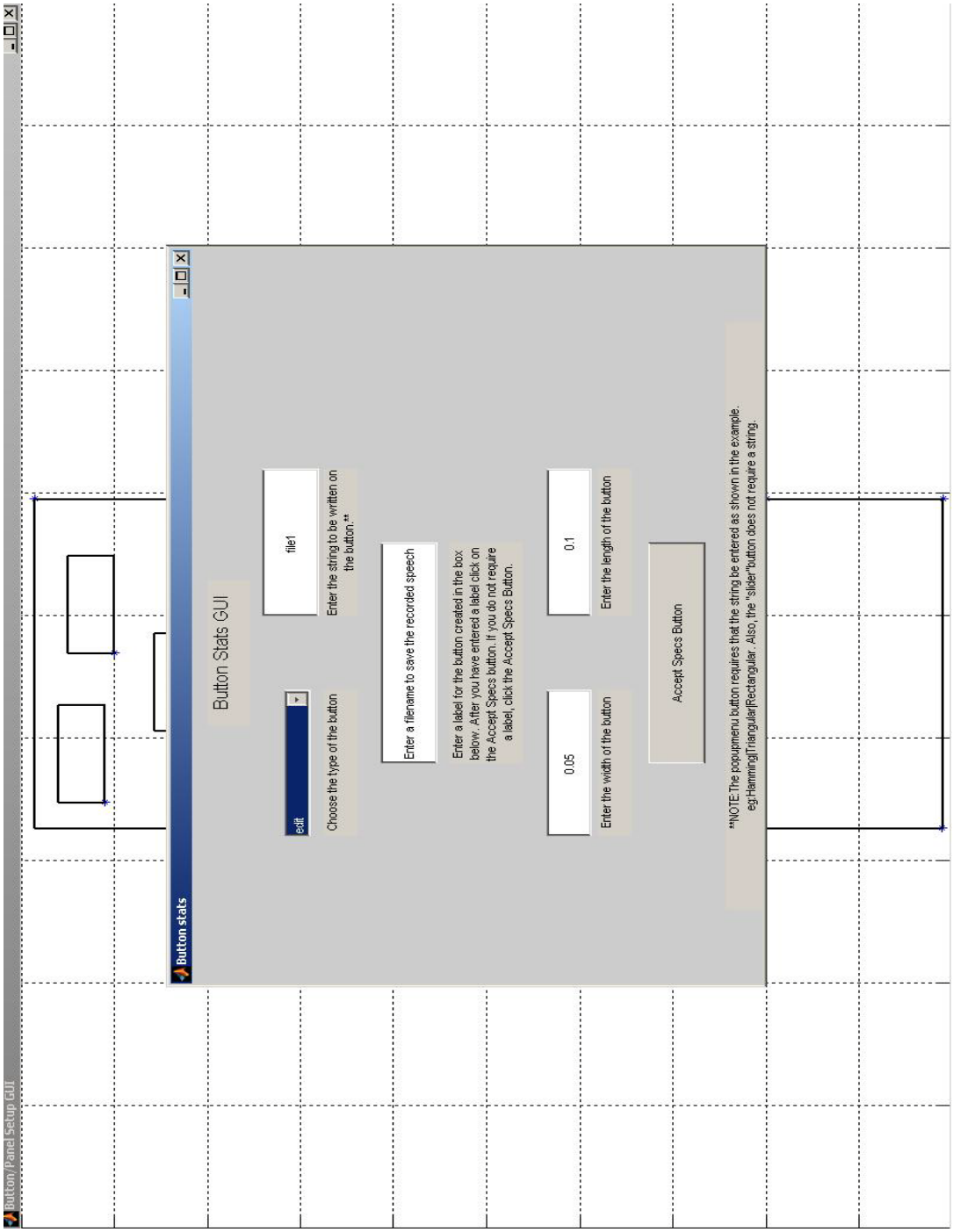

Figure 3.27 The 'Button Stats' GUI after the user has entered the parameters for the 'Enter a filename to save the recorded speech' button. The type of button is 'edit'. The string to be written on the button is 'Enter a filename to save the recorded speech'. The length has been changed to 0.1 units while the width has been left at its default value.

Screenshots of the ‘Get directory’, ‘Select file’ and ‘Close GUI’ have not been included as these buttons are created in exactly the same way as they were in section 3.2.2. Figure 3.28 displays the GUI window visible to the user after the user has saved the selection of positions and dimensions for the created GUI elements.

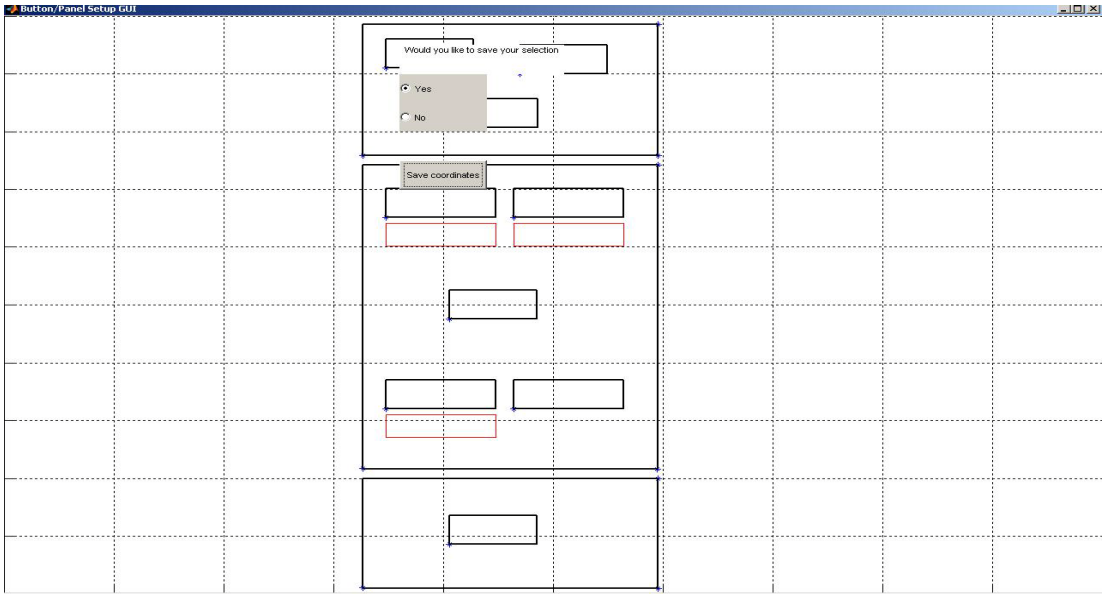

Figure 3.28 The three large vertically stacked rectangles that enclose the smaller rectangles are the three panels. The smaller rectangles are buttons. The red rectangles are the labels for the black rectangular edit boxes directly above them.

In order to see how the GUI looks with all the user parameters loaded into it, the user can edit the name of the ‘.mat’ file in the ‘runGUI.m’ file to ‘recordgui.mat’, and save and run the ‘runGUI.m’ file. On running the ‘runGUI.m’ file, the screen in Figure 3.29 will be visible to the user.

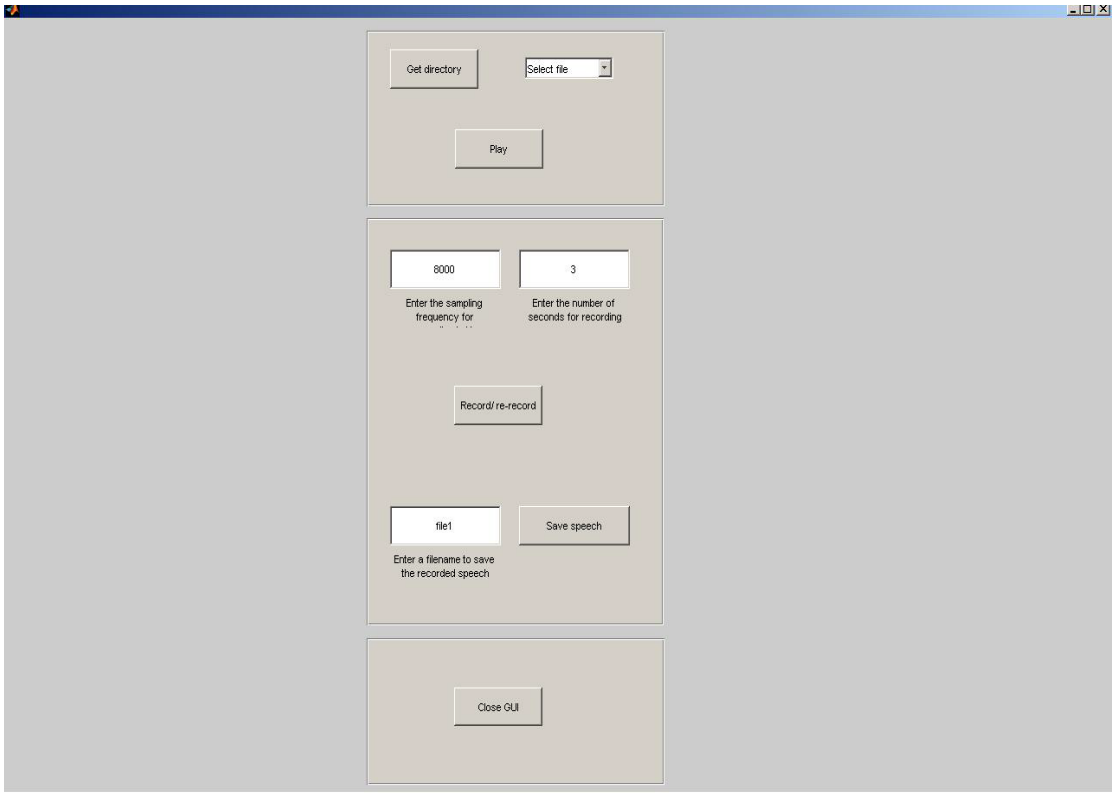

Figure 3.29 The screen visible to the user on running the 'runGUI.m' file with the 'recodgui.mat' file.

At this stage, clicking any of the GUI buttons will cause a MATLAB error alert since the callbacks for the buttons have not yet been written.

The callbacks [9] for the buttons created are mentioned below.

```
%button1-Get directory
    function button1Callback(src,eventdata)
        directory_name = uigetdir('start_path','dialog_title');
        A=strvcat(strcat((directory_name),'\*.wav'));
        struct_filenames=dir(A);
        wav_file_names={struct_filenames.name};
        set(button2,'String',wav_file_names);
        %once the popupmenu/drop down menu is created, by default, the
first
        %selection from the popupmenu/drop down menu must be loaded
even if the
```

```
        %callback for the popupmenu/drop down menu id not called

        indexOfDrpDwnMenu=1;%by   default   first   option   from   the
popupmenu/dropdown
                          %menu will be loaded
        [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu);
    end

    %button2-Select file
    function button2Callback(src,eventdata)
        indexOfDrpDwnMenu=get(button2,'val');
        [curr_file,fs]=loadSelection(directory_name,...
            wav_file_names,indexOfDrpDwnMenu);
    end

    %Function--load selection
    function [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu)
        fin_path=strcat(directory_name,'\',...
            strvcat(wav_file_names(indexOfDrpDwnMenu)));
            %fin_path is the complete path of the file .wav file that
is
            %selected
        clear curr_file;
        clear fs;
        [curr_file, fs]=wavread(fin_path);
        FS=num2str(fs);
        %Information about the file being played
        file_info_string=strcat('Current file = ',...
            wav_file_names(indexOfDrpDwnMenu),...
            '. Sampling frequency = ',FS,'Hz',...
            '. Number of samples in file = ',...
            num2str(length(curr_file)));
    end
```

The callback for button3, the ‘Play’ button plays the selected speech file using the ‘sound’ function of MATLAB.

```
%button3-play button
function button3Callback(src,eventdata)
    sound(curr_file,fs);
end
```

The callback for button4, the ‘Enter the sampling frequency’ button gets the ‘string’ value from the ‘edit’ button and converts it into a ‘numeric’ value. In order for this numeric value ‘fs’ to be shared by the other callbacks, the value ‘fs’ must be initialized to an initial value. Also if an ‘edit’ button contains a default value, the variable returned from the ‘edit’ box, ‘fs’ (in this case), should be initialized to the default value of that box, i.e., ‘8000’. Since the default value for the sampling frequency is 8000 Hz, the value of ‘fs’ has been initialized to 8000. The code for the initialization is included after the code for the ‘Close GUI’ callback.

```
%button4-enter sampling freq
function button4Callback(src,eventdata)
    fs=str2num(get(button4,'string'));
end
```

The callback for button5, performs a function similar to the callback of button4. The variable ‘nsec’ is initialized to the value 3 so that its value can be shared with all the callback functions.

```
%button5-enter no of secs for recording button
function button5Callback(src,eventdata)
    nsec=str2num(get(button5,'string'));
end
```

Callback for the ‘Record/re-record’ button

```
%callback for the record/ re-record button
%record speech file of fixed duration (nsec) and
%given sampling rate(fs)
function button6Callback(h,eventdata)
   button4Callback(h,eventdata);
```

```
   button5Callback(h,eventdata);
% yn=speech samples normalized to 1
% N is the number of samples in each speech file
% ch is the number of channels in the recording
    N=fs*nsec;
    ch=1;
    y=wavrecord(N,fs,ch,'double');
    ymin=min(y);
    ymax=max(y);
    % calculate dc offset and correct
    offset=sum(y(N-999:N))/1000;
    y=y-offset;
    sound(y,fs);
end
```

The callback for button7 performs a function similar to the callback of button4. It gets the 'string' value from 'button7' which is then used by the callback for 'button8'. The variable 'filename' is initialized to the value 'file1' so that it can be shared with all the callbacks.

```
%button7-get filename
 function button7Callback(src,eventdata)
    fileName=get(button7,'string');
 end
```

The callback for button8 is used to save the recorded speech.

```
%button8-save speech
function button8Callback(src,eventdata)
    currentDir=pwd
    currDir=strcat(currentDir,'\',fileName,'.wav')
    wavwrite(y,fs,strvcat(currDir));
    c=wavread(strvcat(currDir));
    soundsc(c,fs)
end
```

The callback for the 'Close GUI' button is used to close the current GUI window.

```
%button9-close gui
function button9Callback(src,eventdata)
    close(gcf);
end
```

Besides the above variables, there are some additional variables that need to be initialized so that their values can be shared among the various callback functions. Also the user can set a name for his GUI using the code:

```
curr_file=1;
directory_name='ABCD';
wav_file_names='ABCD';
y=1;%y is the variable that contains the recorded speech
nsec=3;
fs=8000;
fileName='file1';

%set a name for the GUI
set(f,'Name','Version 2-Record speech GUI');
```

The variable initialization code and the code that sets the name of the GUI are added just below the comment 'USER CODE FOR THE VARIABLES, CALLBACKS AND INITIALIZATION'. The fully functioning GUI at this stage looks like the screen shown in Figure 3.29. The complete code for Program 4 is included in Appendix D.

3.3 GUI Lite – Version 2: Strengths and Weaknesses

GUI Lite – Version 2 automates the processing of selecting the co-ordinates and dimensions of the various objects that are part of a GUI. Version 2 separates the layout process of the various GUI objects including panels, graphic panels, title boxes and buttons from the writing of the callback code that controls and manipulates the created GUI elements. Version 2 simplifies the layout process and saves a considerable amount

of the user's time during GUI development. Version 2 provides the user with predefined callback frameworks for each button. Callback code that performs the functions the buttons are supposed to is written within these frameworks. GUI Lite - Version 2 is scalable and can easily be scaled to accommodate more than the defined number of panels, graphic panels, title boxes and button. GUI Lite – Version 2 simplifies and improves the user experience of creating GUIs a great deal. It provides more than sufficient capabilities for GUI development even though it does not offer the extensive functionality provided by the GUIDE toolbox of MATLAB.

Chapter 4 Testing of GUI Lite - Version 1

4.1 Overview of the testing of GUI Lite - Version 1

The GUI Lite – Version 1's User's Guide was given to three test users for evaluation purposes. The users, after experimenting with GUI Lite – Version 1, were asked to answer the following questions on a scale of 1-10, with 1 being the lowest score and 10 the highest score.

1. Rate your MATLAB proficiency: ____/10.
2. Rate your GUI development proficiency in MATLAB: ___/10.
3. Clarity of GUI Lite - Version 1's User Guide: ____/10.
4. Ease of creating a GUI using GUI Lite - Version 1:____/10.
5. Willingness to use the GUI Lite - Version 1 again: ____/10.
6. Rate your GUI development proficiency after using GUI Lite - Version 1:____/10.
7. Overall rating of the guide: ____/10.

The users were also asked to provide detailed feedback on how to make the User's Guide for Version 1 more user friendly and how to improve its efficiency and clarity.

4.1.1 Feedback for the GUI Lite – Version 1 from user 1

User 1 was a graduate student in engineering who is extremely proficient with MATLAB.

User 1's scores on the above questions are as follows.

1. Rate your MATLAB proficiency: __8__/10.
2. Rate your GUI development proficiency in MATLAB: __1__/10.
3. Clarity of GUI Lite - Version 1's User Guide: __7__/10.

4. Ease of creating a GUI using GUI Lite - Version 1:__7__/10.
5. Willingness to use the GUI Lite - Version 1 again: __8__/10.
6. Rate you GUI development proficiency after using GUI Lite - Version 1:__7__/10.
7. Overall rating of the guide: __7__/10.

Additional feedback from User 1:

User 1 suggested adding more figures and screenshots into the GUI Lite - Version 1's User Guide so that the user could see what the GUI would look like at every stage of its development. User 1 appreciated the detail oriented nature of the User's Guide and the idea that it explained the process of creating a GUI from the most basic level possible. User 1 found editing the position attribute of the various GUI objects very tedious. The user felt that changing the position attribute took as much time as it took to write the entire application's code itself.

4.1.2 Feedback for the GUI Lite – Version 1 from user 2

User 2 was a graduate student in engineering who is moderately proficient with MATLAB.

User 2's feedback after using the GUI Lite – Version 1 is as follows.

1. Rate your MATLAB proficiency: __6__/10.
2. Rate your GUI development proficiency in MATLAB: __1__/10.
3. Clarity of GUI Lite - Version 1's User Guide: __8__/10.
4. Ease of creating a GUI using GUI Lite - Version 1:__7__/10.
5. Willingness to use the GUI Lite - Version 1 again: __8__/10.

6. Rate you GUI development proficiency after using GUI Lite - Version 1:__8__/10.
7. Overall rating of the guide: __8__/10.

Additional feedback from User 2:

User 2 suggested adding more figures into the User's Guide so that the user could see what the GUI being developed would look like at every stage. User 2 found it difficult to position the various GUI objects on the GUI window since the values of the position attribute needed to be entered with a high degree of precision.

4.1.3 Feedback for the GUI Lite – Version 1 from user 3

User 3 was a graduate student in Mathematics who is extremely proficient with MATLAB.

User 3's feedback after using the GUI Lite – Version 1 is as follows.

1. Rate your MATLAB proficiency: __10__/10.
2. Rate your GUI development proficiency in MATLAB: __1__/10.
3. Clarity of GUI Lite - Version 1's User Guide: __9__/10.
4. Ease of creating a GUI using GUI Lite - Version 1:__8__/10.
5. Willingness to use the GUI Lite - Version 1 again: __9__/10.
6. Rate you GUI development proficiency after using GUI Lite - Version 1:__9__/10.
7. Overall rating of the guide: __8__/10.

Additional feedback from User 3:

User 3 too suggested adding a lot more figures into the User's Guide to see step-by-step images of GUI development. User 3 mentioned that the User's Guide for Version 1 was

very detailed, lengthy, and took a lot of time to go through completely. User 3 commended the User's Guide's attention to detail saying that it made understanding how to write a GUI very easy.

4.2 Analysis of the feedback

From the written feedback provided by the users it was evident that the user manual needed to have many more images and screenshots to document all stages of the GUI during GUI development. Based on the feedback, the GUI Lite – Version 1's User's Guide was updated to include a lot more figures and text explaining how the GUI needed to be created. The test users also mentioned that the GUI Lite – Version 1's User Guide was very extensive and took a lot of time to read. The test users also found that entering the 'position' attribute of the various 'uicontrol' objects while creating and positioning the various GUI objects on the GUI window was very time consuming and difficult to get right. The majority of the time allotted to GUI development by a user was taken up by entering the 'position' attribute of the various 'uicontrol' objects. Based on this feedback and the GUI Lite toolbox developer's own experience, we decided to automate and separate the design, positioning and layout of the GUI elements from the writing of the code that controlled them. This idea was the concept behind GUI Lite – Version 2.

Chapter 5 Testing for GUI Lite - Version 2

5.1 Testing for GUI Lite – Version 2

Testing for the GUI Lite - Version 2 was conducted by asking the test subjects to complete a specified GUI design problem using first the GUIDE toolbox and then the GUI Lite – Version 2 toolkit. The GUI created using each toolbox had to be designed and completed within an hour.

The GUI design problem was to create a simple GUI which contained a single button. The button when clicked would display the message 'Hello World'.

The test subjects were asked to use MATLAB's 'help' feature while creating the GUI using MATLAB's GUIDE toolkit. The users were also asked to complete the same GUI design problem using the GUI Lite – Version 2 and its User's Guide. One hour was allotted to completing the problem using each of the toolkits.

The test subjects were asked to fill the questionnaires in sections 5.1.1 and 5.1.2 after completing the problem using each of the toolboxes.

5.1.1 Questionnaire for users using MATLAB's GUIDE toolbox

Rate the following questions on a scale of 1-10, with 1 being the lowest and 10 being the highest.

1. Rate your proficiency in MATLAB? :____/10
2. Rate your proficiency in UI development before using GUIDE? : ____/10
3. The complexity of the problem assigned to you : ____/10
4. How much of your task were you able to complete? : ____/10
5. The level of complexity of the GUIDE toolbox: ____/10

6. The helpfulness of MATLAB's 'help' section for the 'GUIDE' toolbox:

   ____/10

7. Rate your proficiency in UI development after using GUIDE? : ____/10
8. Rate your willingness to use the GUIDE toolbox again? :____/10
9. Mention any other feedback that you would like to provide.

   _____

   _____

   _____

10. Did you use any help other than MATLAB help to complete this task?

    E.g.: Google, other human beings, etc. If yes, list the sources of help.

    _____

    _____

    _____

11. What was the degree of help you required from an external source? : ____/10

5.1.2 Questionnaire for users using the GUI Lite-Version 2 toolbox

Rate the following on a scale of 1-10 with 1 being the lowest and 10 being the highest.

1. Rate your proficiency in MATLAB? :____/10
2. Rate your proficiency in UI development? : ____/10
3. How much of your task were you able to complete? : ____/10
4. The level of complexity of the GUI Lite toolbox: ____/10
5. The helpfulness of the GUI Lite User's Guide for Version 2: ____/10
6. Rate your willingness to use the GUI Lite - Version 2 toolbox again? :____/10
7. Any other feedback that you would like to provide.

_____

_____

_____

8. Did you use any help other than MATLAB help to complete this task?

   E.g.: Google, other human beings, etc. If yes, list the sources of help.

   _____

   _____

   _____

9. What was the degree of help you required from an external source? : ____/10

5.2 Results of the comparative testing between GUIDE and GUI Lite – Version 2

5.2.1 Feedback from User 1 after testing the GUIDE and the GUI Lite toolboxes

User 1 was a graduate student in Engineering.

User 1's feedback after attempting to complete the given problem using MATLAB's GUIDE toolbox is included below.

- Rate the following on a scale of 1-10 with 1 being the lowest and 10 being the highest.

  1. Rate your proficiency in MATLAB? : __5__/10.
  2. Rate your proficiency in UI development before using GUIDE? : __1__/10.
  3. The complexity of the problem assigned to you: __7__/10.
  4. How much of your task were you able to complete? : __10__/10.
  5. The level of complexity of the GUIDE toolbox: __8__/10.

6. The helpfulness of MATLAB's 'help' section for the 'GUIDE' toolbox: __2__/10.
7. Rate your proficiency in UI development after using GUIDE? : __2__/10.
8. Rate your willingness to use the GUIDE toolbox again? : __1__/10.
9. Any other feedback that you would like to provide.

   The 'help' section provided by MATLAB was not very useful. I found better tutorials online that were really helpful to do the given task. The layout provided by GUIDE is formidable and confusing. I decided to just look up an example on the internet because the GUIDE's 'help' section was too confusing.

10. Did you use any help other than MATLAB help to complete this task?

    E.g.: Google, other human beings, etc. If yes, list the sources of help.

    Yes, I used Google. I could not have finished my task without it.

11. What was the degree of help you required from an external source? : __10__/10.

User 1's feedback after using the GUI Lite – Version 2 toolbox is included below.

- Rate the following on a scale of 1-10 with 1 being the lowest and 10 being the highest.

  1. Rate your proficiency in MATLAB? :__5__/10.
  2. Rate your proficiency in UI development? : __2__/10.
  3. How much of your task were you able to complete? : __10__/10.
  4. The level of complexity of the GUI Lite toolbox: __2__/10.

5. The helpfulness of the 'GUI Lite User's Guide for Version 2': __7__/10.
6. Rate your willingness to use the GUI Lite - Version 2 toolbox again?

   :__8__/10.
7. Any other feedback that you would like to provide.

   Using the GUI –Lite toolbox to complete my task was very easy. I just needed to read the manual to know which files to use. Once I found out which files to use the whole process was very intuitive and simple.
8. Did you use any help other than MATLAB help to complete this task?

   E.g.: Google, other human beings, etc. If yes, list the sources of help.

   No.
9. What was the degree of help you required from an external source? :

   __1__/10.

5.2.2 Feedback from User 2 after testing the GUIDE and the GUI Lite toolboxes

User 2 was also a graduate student in Engineering. User 2's feedback after using the GUIDE and the GUI Lite – Version 2 toolkit to complete the given task is mentioned below.

- Rate the following on a scale of 1-10 with 1 being the lowest and 10 being the highest.

  1. Rate your proficiency in MATLAB? :__2__/10
  2. Rate your proficiency in UI development before using GUIDE? : __1__/10
  3. The complexity of the problem assigned to you : __2__/10
  4. How much of your task were you able to complete? : __7__/10
  5. The level of complexity of the GUIDE toolbox: __9__/10

6. The helpfulness of MATLAB's 'help' section for the 'GUIDE' toolbox:

   __2__/10

7. Rate your proficiency in UI development after using GUIDE? : __1__/10

8. Rate your willingness to use the GUIDE toolbox again? :__1__/10

9. Any other feedback that you would like to provide.

   The example demonstrating how to use the GUIDE in MATLAB's help section was a very complicated one. Using the GUIDE within the time limit specified was difficult and I could not complete my task.

10. Did you use any help other than MATLAB help to complete this task?

    E.g.: Google, other human beings, etc. If yes, list the sources of help.

    Yes, I used Google for help. Using Google, I found a simple example to demonstrate how to use the GUIDE toolbox.

11. What was the degree of help you required from an external source? :

    __10__/10.

- User 2's feedback after testing the GUI Lite – Version 2 toolbox is included below. Rate the following on a scale of 1-10 with 1 being the lowest and 10 being the highest.

  1. Rate your proficiency in MATLAB? :__2__/10
  2. Rate your proficiency in UI development? : __1__/10
  3. The helpfulness of the 'GUI Lite User's Guide for Version 2': __9__/10
  4. How much of your task were you able to complete? : __10__/10
  5. The level of complexity of the GUI Lite toolbox: __2__/10

6. Rate your willingness to use the GUI Lite - Version 2 toolbox again?

   :__10__/10

7. Any other feedback that you would like to provide.

   I found the GUI Lite very systematic and easy to follow. Its step-by-step procedure helped me to complete my task easily and well within the time limit.

8. Did you use any help other than MATLAB help to complete this task?

   E.g.: Google, other human beings, etc. If yes, list the sources of help.

   No.

9. What was the degree of help you required from an external source? :

   __0__/10.

5.3 Analysis of the feedback obtained after testing the GUIDE and the GUI Lite toolbox

The feedback from the users who tested the GUIDE and GUI Lite toolboxes for GUI development clearly indicate that using the GUI Lite – Version 2 toolbox is simpler, easier and far more intuitive than using the GUIDE toolbox from MathWorks. The GUI Lite toolbox improves the user's experience while creating a GUI. GUI Lite allows a user to create a GUI of the same complexity level that would have been possible using the GUIDE toolbox. GUI Lite provides a user with sufficient functionality while offering a low complexity user interface for the user to create GUIs in MATLAB. GUI Lite has achieved its objective of providing a low complexity and highly intuitive GUI development solution which users can use in MATLAB to create Graphical User Interfaces with ease.

Appendix A

GUI Lite – Version 1 User's Guide for the simpleGUI

The GUI Lite is a Graphical User Interface design tool. It provides a user with a step-by-step guide to create GUI's to demonstrate and test various speech processing applications with ease.

The current user guide is written using functions provided in MATLAB 7.8.0.347 (R2009a).

This guide is specifically written to create a GUI to display, play and spectrally analyze a subset of the file. It has features to select a frame of a user specified size of the speech file and to plot the log magnitude Fourier Transform of the selected frame. To use this guide you must read the explanations and comments provided with the code. Use the given sections of code in the MATLAB editor to create you own GUI. The complete code is given at the end of this document. The buttons are populated from top-to-bottom and left-to-right.

Overview of the GUI design and the various buttons involved.

The entire GUI figure window is divided into four panels in which buttons and graphics panels i.e. panels in which plots will be drawn will be placed. Figures 1 and 2 provide you with an idea of how the GUI window will look. Panels 1, 3 and 4 are button panels i.e. panels containing only buttons and panel 2 contains two graphic panels.

The types of buttons that will be used for this GUI are as follows:

1. Pushbutton: This is the default style of a button. Hence there is no need to specify ‘style’ in the ‘uicontrol’ function of MATLAB while creating a pushbutton. This button executes code in the callback when clicked via mouse.
2. Edit box: This kind of button is used to provide input to a GUI in MATLAB (eg: The value of a variable). The ‘style’ property of the ‘uicontrol’ function of this button is set to ‘edit’. The value in the edit box can be changed by moving the mouse to the box and editing the value in the box.
3. Popupmenu button: The popupmenu button generates a pull down menu which is populated with a list of options for user selection (eg: Possible speech files within a directory for analysis). The ‘style’ property of the ‘uicontrol’ function of this button is set to ‘popupmenu’.

1. Define the GUI as a function.

```
function simpleGUI
%embedded code for the GUI application
end
```

2. Create the GUI and define the various panels that you require within the created function.

Initially add the line ‘clc; clear all;’ to the GUI code to clear all variables before running. The global variable declarations and variables initializations should preferably be done after the ‘clc; clear all;’ line.

Create a GUI window using the ‘figure’ command. The GUI objects that need to be created are added into this GUI window. The 'figure' function in MATLAB is used to define the GUI window within which all the panels are defined. Normalizing the units of the GUI window to the range [0 0] to [1 1] using the ‘figure’ function lets the position of the various panels defined within the GUI window be fixed even if the GUI is run on another computer.

The panels are the rectangles defined within the main GUI window. Buttons performing various GUI functions are then defined within the panels. The ‘set’ function in MATLAB is used to set the name of the GUI and to initialize the GUI and its associated panels. Here the name of the GUI is 'Simple Test GUI'. Figure 1 displays the output of the figure command which defines the GUI window and divides it into four panels. The code below details how the panels are created.

```
clc;clear all;

%GLOBAL VARIABLE DECLARATIONS
global curr_file;%curr_file is the value of
                 %the .wav file once read into MATLAB
global fs;       %sampling frequency
global x;        %starting point for expansion
                 %selected using ginput function
global frame_dur;%length of the frame in seconds
                 %for which expansion must be done
global W;        %it is the selected frame of curr_file
global nfft;     %nfft is the value of n in the n point stft


%INITIALIZATION
%The variable returned from the edit boxes must
%be initialized, else the box will only display
```

```
%the value but not actually hold that value
frame_dur=0.04;
nfft=1024;

%The 1 1 in the position attribute means that the GUI
%is fit to screen
f = figure('Visible','on',...
    'Units','normalized',...
    'Position',[0,0,1,1],...
    'MenuBar','none',...
    'NumberTitle','off');

%GUI PANELS
%This GUI is divided into four panels
panel1=uipanel('Parent',f,...
    'Units','Normalized',...
    'Position',[0 0.001 0.23 0.354]);%left bottom panel
panel2=uipanel('Parent',f,...
    'Units','Normalized',...
    'Position',[0.233 0 0.8 1]);%right panel
panel3=uipanel('Parent',f,...
    'Units','Normalized',...
    'Position',[0 0.359 0.23 0.28]);% left center panel
panel4=uipanel('Parent',f,...
    'Units','Normalized',...
    'Position',[0 0.642 0.23 0.358]);%left top panel

% Assign the GUI a name to appear in the window title.
set(f,'Name','Simple Test GUI');

%initialize GUI
set([f,panel1,panel2,panel3,panel4],'Units','normalized')
```

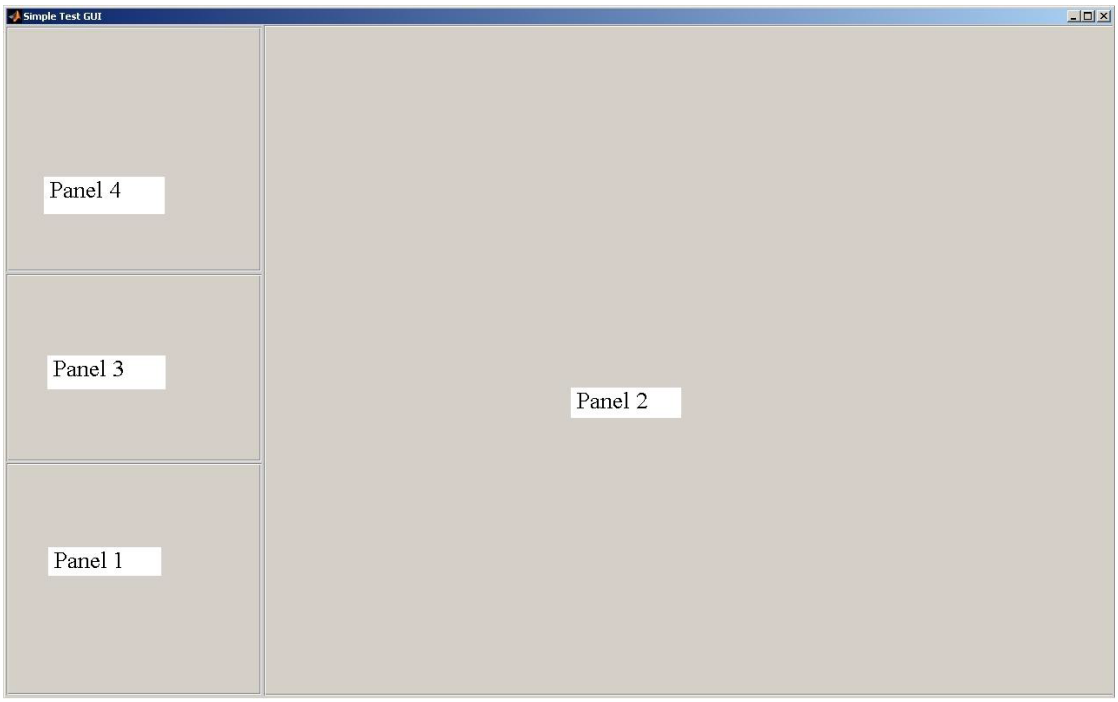

Figure 1: This displays the GUI window along the four panels created within it. Within the panels, buttons controlling the GUI will be defined.

The next step is to create two graphics panels within which the waveform and its log magnitude Fourier Transform can be displayed. Both graphics panels are created within Panel 2 which is the large panel on the right. The graphics panels are created using the 'axes' function. The ‘axes’ function lets the user specify parameters like parent, i.e the panel in which the new graphics panel is going to be placed, position, units, etc. and returns the handle or name of the panel. Figure 2 shows the GUI window with the relevant co-ordinates and size of the panels marked on the figure.

```
%GRAPHICS PANELS
%Create graphics panels for plots in panel2
%handle_top is the handle of the top graphics panel.
%Similarly handle_bott is the handle of the bottom panel.
%The bottom panel plots the waveform of the complete file.
%It is also used to plot one frame of the file.
%The top panel is used to plot the log magnitude
```

```
%spectrum of the window-weighted FFT of the selected frame.
handle_top = axes('parent',panel2,...
    'Units','Normalized',...
    'Position',[0.05 0.5 0.8 0.35],...
    'GridLineStyle','--');
handle_bott = axes('parent',panel2,...
    'Units','Normalized',...
    'Position',[0.05 0.08 0.8 0.35],...
    'GridLineStyle','--');
```

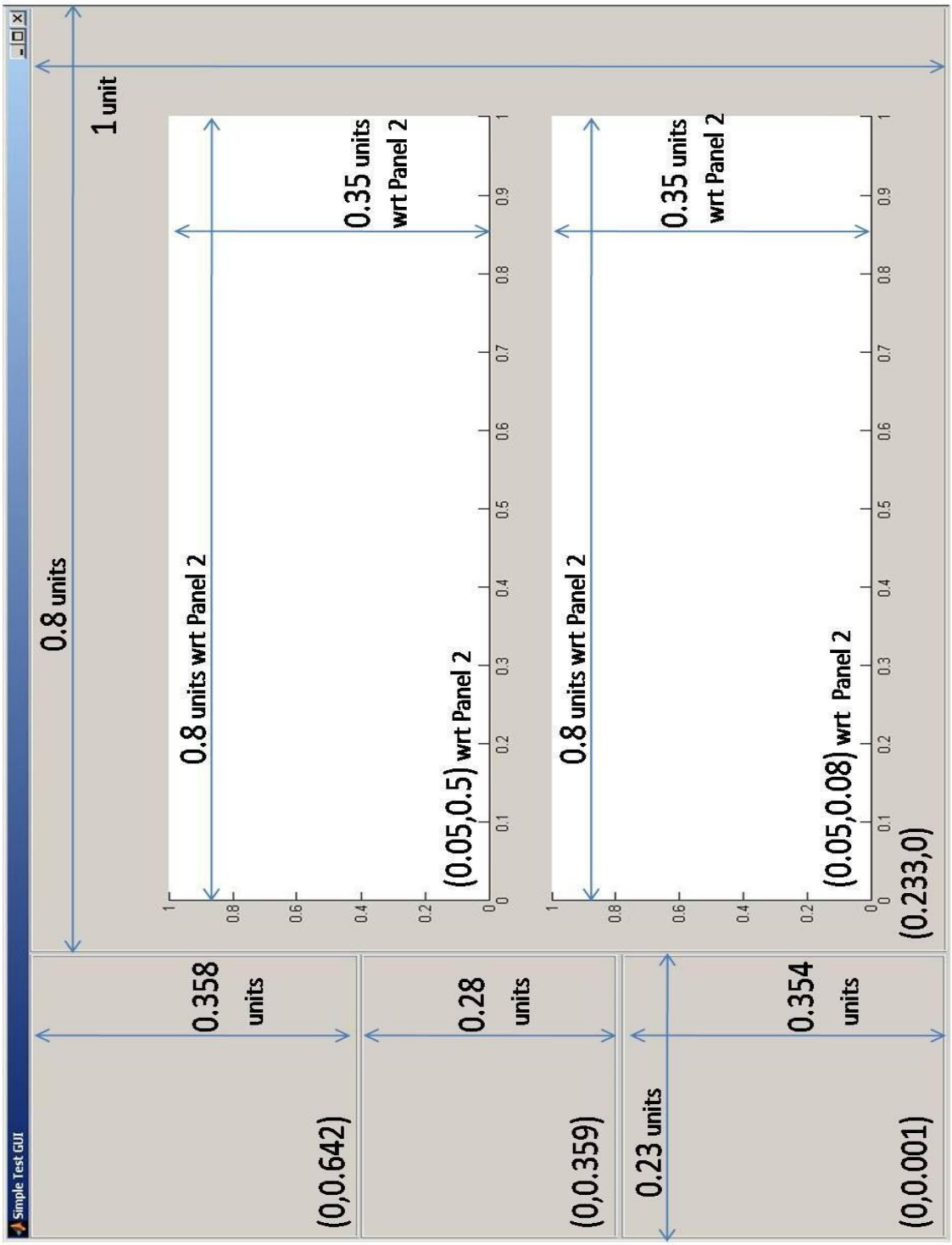

Figure 2: View of the GUI after step 2. The panels on the left are used for the controls of the GUI. The panel on the right containing two graphics panels is used to display plots of the waveform (lower graphics panel) and the log magnitude spectrum (upper graphics panel).

3. Creating the controls for the GUI. This includes all the buttons and menus which are required in order to display the waveform and the spectral log magnitude output on the bottom and upper sub-panels of panel 2.

1. Get directory/Select file button

This button is used to get a list of directories from which the .wav file to be displayed is selected and to select the speech file for display and analysis. This button is defined using the 'uicontrol' function. The 'uicontrol' function is used to specify various attributes of the button.

The callback for the 'Get directory/Select file' button will be explained in the section related to callbacks. The callback is a section of code that makes the buttons actually work. The buttons are not yet ready to be clicked since the callbacks have not yet been defined. Note that the default style of button is a 'pushbutton' , hence it is not specified below.

```
%BUTTONS
% Get directory/Select file button
getDirectorybutton=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.1 2.5 0.35 0.16],...
    'String','Get directory/Select file',...
    'Callback',@getDirectoryCallback);
```

2. Play button

This button is used to play the selected .wav file. It is created in the same way as the 'Get directory/Select file' button.

```
%Play button
playbutton=uicontrol('Parent',panel1,...
```

```
    'Units','Normalized',...
    'Position',[0.1 2.1 0.35 0.16],...
    'String','Play speech file',...
    'Callback',@playbuttonCallback);
```

3. Plot button

This button is used to plot the selected .wav file. It is created in the same way as the 'Get directory/Select file' button.

```
%Plot button
plotbutton=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.6 2.1 0.35 0.16],...
    'String','Plot speech file',...
    'Callback',@plotbuttonCallback);
```

4. Select starting sample in speech file for display and short time Fourier analysis.

This is used to select the starting sample for a frame of speech to be displayed in the bottom graphics panel. This frame of speech is defined from the selected starting sample and is of the duration specified by the frame length (in seconds) button, as described below.

```
%Select starting sample of frame for expansion
startpoint=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.1 1.55 0.35 0.16],...
    'String','Select start point',...
    'CallBack', @startpointCallBack);
```

5. Enter frame length (in seconds) box.

This button is used to input the length of the frame (in seconds) to be displayed. Note that the type of this box is 'Edit' and that it will have a label (‘Enter length in seconds’), explaining the meaning of the input. Frame length is initially set to 40 msecs i.e 0.04 seconds. Hence the variable returned by the frame_lenCallback function is initialized to 0.04 as follows.

```
%INITIALIZATION
frame_dur=0.04;
```

Frame length button code

```
%Enter frame length
frame_len = uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.6 1.55 0.35 0.16],...
    'Style','Edit',...
    'String','0.04',...
    'HorizontalAlignment','center',...
    'BackgroundColor','w',...
    'CallBack', @frame_lenCallBack);
Label2 = uicontrol('parent',panel1,...
    'Units','Normalized',...
    'Position',[0.6 1.42 0.35 0.12],...
    'Style','text',...
    'String','Enter length in seconds');
```

6. Plot frame button.

This button is a push button which will plot the selected frame of speech with a Hamming window superimposed over the duration of the speech frame.

```
%This plots the expanded waveform on the screen
expand=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.1 1.2 0.35 0.16],...
    'String','Plot frame',...
    'HorizontalAlignment','center',...
    'CallBack', @plotFrameCallBack);
```

7. FFT size box.

This button is used to enable the user to enter the size of the FFT analysis (in samples). Note that the style of the button is 'Edit' since users need to enter a value into it. This box will have a label below it explaining what it does or the value that must be entered. Here it is called 'Label1'. By default the FFT size box has the value '1024' in it. In order for an edit box to have an initial value, the string field must have the same initial value. One important point to remember about edit boxes which are initialized to some default value is that just setting the string field to the initial value is not sufficient for correct operation. The variable returned by the FFTsizeCallback function should be initialized to the same initial value at the beginning of the program, or else the callback function will not work properly. The initializations must be done at the beginning of the program below the initial 'clc' and 'clear' functions.

```
%INITIALIZATION
nfft=1024;

%Enter FFT size, box code
%Enter no of samples for FFT
FFTsize=uicontrol('Parent',panel1,...
```

```
    'Units','Normalized',...
    'Position',[0.1 0.7 0.35 0.16],...
    'Style','Edit',...
    'String','1024',...
    'HorizontalAlignment','center',...
    'BackgroundColor','w',...
    'CallBack', @FFTsizeCallBack);
Label1 = uicontrol('parent',panel1,...
    'Units','Normalized',...
    'Position',[0.1 0.57 0.35 0.12],...
    'Style','text',...
    'String','Enter the number of samples for FFT');
```

8. STFT button.

This button is a push button which computes the log magnitude of the short-time Fourier transform (STFT) of the window-weighted speech frame and plots it in the upper graphics panel.

```
%Display FFT
getstft=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.6 0.7 0.35 0.16],...
    'String','STFT',....
    'Callback',@getstftCallback);
```

9. Save screen button.

This button is a push button which allows you to save a screenshot of the GUI graphics display.

```
%Save screen button
savebutton=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.1 0.3 0.35 0.16],...
    'String','Save screen',...
    'Callback',@saveCallback);
```

10. Close GUI button.

This button is a push button which closes the GUI window.

```
%Close
closebutton=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.6 0.3 0.35 0.16],...
    'String','Close',...
    'Callback',@closeCallback);
```

The reader is cautioned that none of these buttons work, as yet, since the callbacks for them have yet to be defined. Callbacks are the code that actually makes the buttons work. The callback code performs the function tha Figure 3 shows how the GUI will look after all the buttons have been added.

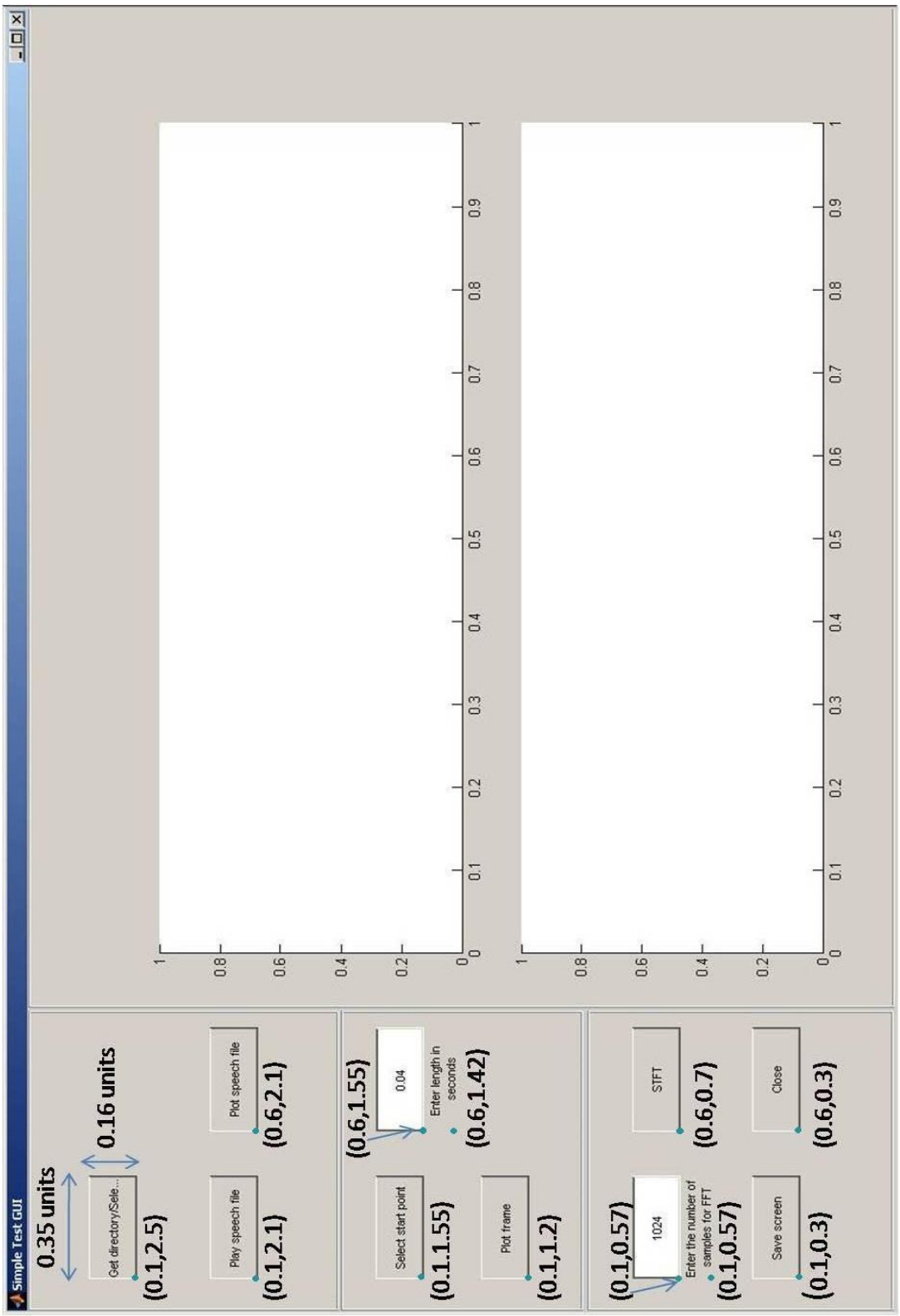

Figure 3: View of the GUI after step 3: Insertion of all required buttons and boxes.

Observe that all buttons related to a particular function, eg: The playing or plotting a speech file, are dependent on the 'Get directory/Select file' button which ultimately specifies the speech file for analysis. Hence those set of buttons are all placed in one panel. Buttons that are dependent on each other are placed in a single panel so that the user can easily identify dependencies and minimize errors.

4. Adding callbacks for each of the buttons created.

1. Callback for the Get directory/Select file button.

The getDirectoryCallback function allows the user to select a folder (directory) which contains speech files stored in .wav format. If the selected folder does contain one or more .wav files, a popup menu is created above the 'Get directory/Select file' button. This allows the user to select a .wav file to work with. The popup menu and its callback are nested within the getDirectoryCallback function because the popup menu depends on the getDirectoryCallback function. Also, as soon as a file is selected, a label, 'file_info_label' will display relevant information about the current file. The information consists of, name of the file being played, sampling rate and the total number of samples in the file.

The 'getDirectoryCallback' callback is implemented by using the 'uigetdir' function which launches a dialog box using which the folder containing the .wav files maybe selected. The value returned from the 'uigetdir' function is then

concatenated with '*.wav' and passed to the 'dir' function. This provides us with a list of .wav files which are displayed using the popup menu.

The popup menu is created using the 'uicontrol' function, specifying the style of the button as 'popupmenu'. The other attributes of the button are specified in the same way as described earlier for the other buttons.

The 'file_info_label' is created the same way you would create a button, however the style is specified as 'text'.

```
%Callbacks
%callback for the Get directory/Select file button
 function getDirectoryCallback(h,eventdata)
    dir_name = uigetdir('start_path','dialog_title');
    A=strvcat(strcat((dir_name),'\*.wav'));
    emp_struct1=dir(A);
    wav_file_names={emp_struct1.name};
    %drop down menu
    drp_dwn_menu = uicontrol('Parent',panel1,...
        'Units','Normalized',...
        'Position',[0.6 2.45 0.35 0.16],...
        'Style','popupmenu',...
        'Visible','on',...
        'String',wav_file_names,...
        'BackgroundColor','w',...
        'CallBack', @drp_dwn_callback);

    function drp_dwn_callback(h,eventdata)
        val1=get(drp_dwn_menu,'val');
        %it is the path of the file .wav file that is selected

fin_path=strcat(dir_name,'\',strvcat(wav_file_names(val1)));
        %RMD
```

```
        clear curr_file;
        clear fs;
        [curr_file, fs]=wavread(fin_path);
        FS=num2str(fs);
        %Information about the file being played
        file_info_strng=strcat('Current file = ',...
            wav_file_names(val1),...
            '. Sampling frequency = ',FS,'Hz',...
            '. Number of samples in file = ',...
            num2str(length(curr_file)));
        file_info_label = uicontrol('parent',panel2,...
            'Units','Normalized',...
            'Position',[0 0.9 0.9 0.05],...
            'Style','text',...
            'FontUnits','Normalized','FontSize',0.5,'String',...
            file_info_strng);
    end
end
```

2. Callback for the Play speech file button.

The sound function in MATLAB is used to play the selected file. The arguments for the sound function are curr_file and fs which are variables present in the getDirectoryCallback function. In order for the playbuttonCallback function to work, these variables need to be declared as global. Use the declarations below at the beginning of the GUI function in Step 1.

```
%GLOBAL VARIABLE DECLARATIONS
global curr_file;%curr_file is the value of
                 %the .wav file once read into MATLAB
global fs;       %sampling frequency
```

The function below is the one that plays the sound file. This snippet should be placed with all the other callbacks.

```
%callback for the playbutton
 function playbuttonCallback(h,eventdata)
    sound(curr_file,fs);
 end
```

3. Callback for the Plot speech file button.

In order to plot a waveform, the panel in which it will be displayed must be selected. Here the name or handle is handle_bott since we are using the bottom graphics panel. The panel is selected by passing the handle of the panel as an argument to the axes function.

```
%callback for the plotbutton
 function plotbuttonCallback(h,eventdata)
    hold off;  %It is essential to turn hold off so that the
               %earlier contents of the panel are replaced
    axes(handle_bott);
    l=length(curr_file);
    i=(1:l)/fs;%coverting samples to time
    plot(i,curr_file,'k','LineWidth',2),...
        xlabel('Time in seconds'),...
        ylabel('Amplitude');
    axis tight;
 end
```

4. Callback to obtain starting sample for frame analysis.

The 'ginput' function uses vertical and horizontal cursor lines to return the x and y co-ordinates of the selected sample.

For this function only the x-coordinate has any relevance. To retain this value for use by any other function, declare it as global as shown. This will ensure that the value of the x-coordinate is shared by all the functions.

```
%GLOBAL VARIABLE DECLARATIONS
global x;        %starting point for expansion
                 %selected using ginput function
```

Starting sample function:

```
%callback for startpoint
 function startpointCallBack(varargin)
    [x,y]=ginput(1);
 end
```

5. Callback to set the frame length.

For this function use the following statement for global variable declaration.

```
%GLOBAL VARIABLE DECLARATIONS
global frame_dur;%length of the frame in seconds
                 %for which expansion must be done
```

Frame length function:

```
%callback for frame_len
 function frame_lenCallBack(varargin)
    frame_dur = str2num(get(frame_len,'String'));
 end
```

6. Callback to plot the selected frame (along with a Hamming window overlap).

This function plots the selected frame. It also creates a label, Label5, giving information regarding the length of the selected frame in samples. This information is displayed above the top graphics panel.

For this function the selected frame array should be declared as global. The getstftCallback requires this array for calculation and display of the STFA of the window-weighted frame. The global declaration is as follows.

```
%GLOBAL VARIABLE DECLARATIONS
global W;        %it is the selected frame of curr_file
```

Function to plot the selected frame:

```
%callback for expand
 function plotFrameCallBack(h,eventdata)
    hold off; %to clear the hold set by the plot command
    X=round(x*fs);
    W=curr_file(X:round(x*fs+frame_dur*fs));
    axes(handle_bott);  %plotting  will  be  done  in  the  bottom
panel
    i=(1:length(W))/fs;%converting samples to time
    plot(i,W,'k','LineWidth',2),...
    xlabel('Time in seconds'),...
    ylabel('Amplitude');
    hold on;
    plot(i,hamming(length(W)),'b');%hamming   window   over   the
drawn plot
    grid on;
    hold on;
```

```
    axis tight;
    frame_length=num2str(length(W));
    frame_info_strng=strcat('Selected           frame           =
',frame_length,'samples');
    Label5 = uicontrol('parent',panel2,...
        'Units','Normalized',...
        'Position',[0 0.85 0.9 0.05],...
        'Style','text',...
        'FontUnits','Normalized',...
        'FontSize',0.5,...
        'String',frame_info_strng);
 end
```

7. Callback to get the size of FFT used in the STFA.

Here the 'get' function is used to get the value of the box with the handle num_FFT. The value entered is in string form and must be converted to a number.

For this function the value is returned in a variable nfft. It should be declared as a global variable. The following statement should be placed with the other global variable declarations.

```
%GLOBAL VARIABLE DECLARATIONS
global nfft;     %nfft is the value of n in the n point stft
```

Function to get size of FFT:

```
%Callback to get the number of samples for which FFT must be
calculated
 function FFTsizeCallBack(varargin)
    nfft = str2num(get(FFTsize,'String'));
 end
```

8. Callback to calculate and display the log magnitude of the STFA.

```
%callback for STFT
 function getstftCallback(h,eventdata)
    if (length(W)>nfft)%Check if aliasing needs to be done
         %Msg box provides a warning
         msgbox('Length of the frame must be less than FFT
length','modal')
    else
        W1=W*hamming(length(W))';
        f=fft(W1,nfft);
    end
    F=20*log10(abs(f));
    tempFFT=F(1:length(F)/2);%displaying fft for only
                             %half the range since it is
periodic
    axes(handle_top);        %displaying fft in the top panel
    %Y is the range from 0 to the number of samples specified by
user
    Y=(1:nfft/2)*(fs/nfft);
    plot(Y,tempFFT,'k','LineWidth',2),...
        xlabel('Frequency in Hz'),...
        ylabel('Log Magnitude in dB'),...
        title('Fourier Transform');
    axis tight;
 end
```

9. Callback for the Save screen button.

This callback saves a screen shot of the GUI in bitmap format in the location specified in the code.

```
%Callback for save screenshot
function saveCallback(h,eventdata)
    currentDir=pwd;
    currDir=strcat(currentDir,'\new');
    print('-dbitmap',currDir)%prints a screenshot and saves it
in
                             %the work folder as new.bmp
```

```
end
```

10. Callback for the Close GUI button.

This callback closes the GUI window.

```
%Callback for close
function closeCallback(h,eventdata)
    close(gcf);
end
```

Figure 4 shows an example of the completed GUI for the file s3.wav.

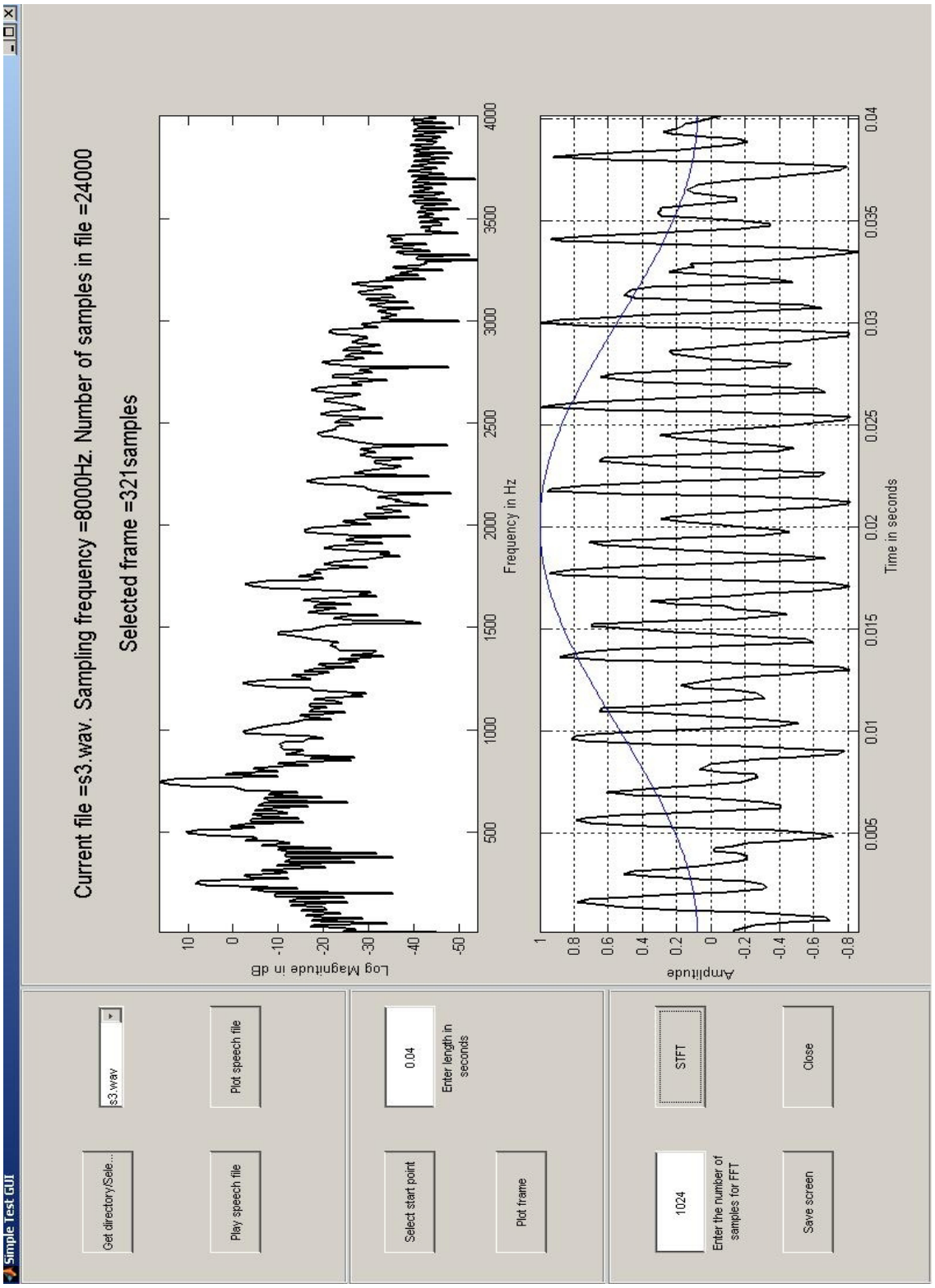

Figure 4: The completed GUI with all the buttons and their associated callbacks working perfectly.

Complete Code:

```
function simpleGUI

clc;clear all;

%GLOBAL VARIABLE DECLARATIONS
global curr_file;%curr_file is the value of
                 %the .wav file once read into MATLAB
global fs;       %sampling frequency
global x;        %starting point for expansion
                 %selected using ginput function
global frame_dur;%length of the frame in seconds
                 %for which expansion must be done
global W;        %it is the selected frame of curr_file
global nfft;     %nfft is the value of n in the n point stft


%INITIALIZATION
%The variable returned from the edit boxes must
%be initialized, else the box will only display
%the value but not actually hold that value
frame_dur=0.04;
nfft=1024;

%The 1 1 in the position attribute means that the GUI
%is fit to screen
f = figure('Visible','on',...
    'Units','normalized',...
    'Position',[0,0,1,1],...
    'MenuBar','none',...
    'NumberTitle','off');

%movegui(f,'center');

%GUI PANELS
%This GUI is divided into four panels
```

```
panel1=uipanel('Parent',f,...
    'Units','Normalized',...
    'Position',[0 0.001 0.23 0.354]);%left bottom panel
panel2=uipanel('Parent',f,...
    'Units','Normalized',...
    'Position',[0.233 0 0.8 1]);%right panel
panel3=uipanel('Parent',f,...
    'Units','Normalized',...
    'Position',[0 0.359 0.23 0.28]);% left center panel
panel4=uipanel('Parent',f,...
    'Units','Normalized',...
    'Position',[0 0.642 0.23 0.358]);%left top panel

% Assign the GUI a name to appear in the window title.
set(f,'Name','Simple Test GUI');

%initialize GUI
set([f,panel1,panel2,panel3,panel4],'Units','normalized')

%GRAPHICS PANELS
%Create graphics panels for plots in panel2
%handle_top is the handle of the top graphics panel.
%Similarly handle_bott is the handle of the bottom panel.
%The bottom panel plots the waveform of the complete file.
%It is also used to plot one frame of the file.
%The top panel is used to plot the log magnitude
%spectrum of the window-weighted FFT of the selected frame.
handle_top = axes('parent',panel2,...
    'Units','Normalized',...
    'Position',[0.05 0.5 0.8 0.35],...
    'GridLineStyle','--');
handle_bott = axes('parent',panel2,...
    'Units','Normalized',...
    'Position',[0.05 0.08 0.8 0.35],...
    'GridLineStyle','--');

%BUTTONS
% Get directory/Select file button
```

```
getDirectorybutton=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.1 2.5 0.35 0.16],...
    'String','Get directory/Select file',...
    'Callback',@getDirectoryCallback);


%Play button
playbutton=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.1 2.1 0.35 0.16],...
    'String','Play speech file',...
    'Callback',@playbuttonCallback);

%Plot button
plotbutton=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.6 2.1 0.35 0.16],...
    'String','Plot speech file',...
    'Callback',@plotbuttonCallback);

%Select starting sample of frame for expansion
startpoint=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.1 1.55 0.35 0.16],...
    'String','Select start point',...
    'CallBack', @startpointCallBack);

%Enter frame length
frame_len = uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.6 1.55 0.35 0.16],...
    'Style','Edit',...
    'String','0.04',...
    'HorizontalAlignment','center',...
    'BackgroundColor','w',...
    'CallBack', @frame_lenCallBack);
Label2 = uicontrol('parent',panel1,...
```

```
    'Units','Normalized',...
    'Position',[0.6 1.42 0.35 0.12],...
    'Style','text',...
    'String','Enter length in seconds');

%This plots the expanded waveform on the screen
expand=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.1 1.2 0.35 0.16],...
    'String','Plot frame',...
    'HorizontalAlignment','center',...
    'CallBack', @plotFrameCallBack);

%Enter no of samples for FFT
FFTsize=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.1 0.7 0.35 0.16],...
    'Style','Edit',...
    'String','1024',...
    'HorizontalAlignment','center',...
    'BackgroundColor','w',...
    'CallBack', @FFTsizeCallBack);
Label1 = uicontrol('parent',panel1,...
    'Units','Normalized',...
    'Position',[0.1 0.57 0.35 0.12],...
    'Style','text',...
    'String','Enter the number of samples for FFT');

%Display FFT
getstft=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.6 0.7 0.35 0.16],...
    'String','STFT',....
    'Callback',@getstftCallback);

%Save screen button
savebutton=uicontrol('Parent',panel1,...
    'Units','Normalized',...
```

```
    'Position',[0.1 0.3 0.35 0.16],...
    'String','Save screen',...
    'Callback',@saveCallback);

%Close
closebutton=uicontrol('Parent',panel1,...
    'Units','Normalized',...
    'Position',[0.6 0.3 0.35 0.16],...
    'String','Close',...
    'Callback',@closeCallback);

%Callbacks
%callback for the Get directory/Select file button
 function getDirectoryCallback(h,eventdata)
    dir_name = uigetdir('start_path','dialog_title');
    A=strvcat(strcat((dir_name),'\*.wav'));
    emp_struct1=dir(A);
    wav_file_names={emp_struct1.name};
    %drop down menu
    drp_dwn_menu = uicontrol('Parent',panel1,...
        'Units','Normalized',...
        'Position',[0.6 2.45 0.35 0.16],...
        'Style','popupmenu',...
        'Visible','on',...
        'String',wav_file_names,...
        'BackgroundColor','w',...
        'CallBack', @drp_dwn_callback);

    function drp_dwn_callback(h,eventdata)
        val1=get(drp_dwn_menu,'val');
        %it is the path of the file .wav file that is selected
        fin_path=strcat(dir_name,'\',strvcat(wav_file_names(val1)));
        %RMD
        clear curr_file;
        clear fs;
        [curr_file, fs]=wavread(fin_path);
        FS=num2str(fs);
        %Information about the file being played
```

```
        file_info_strng=strcat('Current file = ',...
            wav_file_names(val1),...
            '. Sampling frequency = ',FS,'Hz',...
            '. Number of samples in file = ',...
            num2str(length(curr_file)));
        file_info_label = uicontrol('parent',panel2,...
            'Units','Normalized',...
            'Position',[0 0.9 0.9 0.05],...
            'Style','text',...
            'FontUnits','Normalized','FontSize',0.5,'String',...
            file_info_strng);
    end
 end

%callback for the playbutton
 function playbuttonCallback(h,eventdata)
    sound(curr_file,fs);
 end

%callback for the plotbutton
 function plotbuttonCallback(h,eventdata)
    hold off;  %It is essential to turn hold off so that the
               %earlier contents of the panel are replaced
    axes(handle_bott);
    l=length(curr_file);
    i=(1:l)/fs;%coverting samples to time
    plot(i,curr_file,'k','LineWidth',2),...
        xlabel('Time in seconds'),...
        ylabel('Amplitude');
    axis tight;
 end

%callback for startpoint
 function startpointCallBack(varargin)
    [x,y]=ginput(1);
 end

%callback for frame_len
```

```
 function frame_lenCallBack(varargin)
    frame_dur = str2num(get(frame_len,'String'));
 end

%callback for expand
 function plotFrameCallBack(h,eventdata)
    hold off; %to clear the hold set by the plot command
    X=round(x*fs);
    W=curr_file(X:round(x*fs+frame_dur*fs));
    axes(handle_bott); %plotting will be done in the bottom panel
    i=(1:length(W))/fs;%converting samples to time
    plot(i,W,'k','LineWidth',2),...
    xlabel('Time in seconds'),...
    ylabel('Amplitude');
    hold on;
    plot(i,hamming(length(W)),'b');%hamming window over the drawn plot
    grid on;
    hold on;
    axis tight;
    frame_length=num2str(length(W));
    frame_info_strng=strcat('Selected              frame              =
',frame_length,'samples');
    Label5 = uicontrol('parent',panel2,...
        'Units','Normalized',...
        'Position',[0 0.85 0.9 0.05],...
        'Style','text',...
        'FontUnits','Normalized',...
        'FontSize',0.5,...
        'String',frame_info_strng);
 end

%Callback to get the number of samples for which FFT must be
calculated
 function FFTsizeCallBack(varargin)
    nfft = str2num(get(FFTsize,'String'));
 end

%callback for STFT
```

```
 function getstftCallback(h,eventdata)
    if (length(W)>nfft)%Check if aliasing needs to be done
         %Msg box provides a warning
         msgbox('Length of the frame must be less than FFT
length','modal')
    else
        W1=W*hamming(length(W))';
        f=fft(W1,nfft);
    end
    F=20*log10(abs(f));
    tempFFT=F(1:length(F)/2);%displaying fft for only
                             %half the range since it is periodic
    axes(handle_top);        %displaying fft in the top panel
    %Y is the range from 0 to the number of samples specified by user
    Y=(1:nfft/2)*(fs/nfft);
    plot(Y,tempFFT,'k','LineWidth',2),...
        xlabel('Frequency in Hz'),...
        ylabel('Log Magnitude in dB'),...
        title('Fourier Transform');
    axis tight;
 end

%Callback for save screenshot
function saveCallback(h,eventdata)
    currentDir=pwd;
    currDir=strcat(currentDir,'\new');
    print('-dbitmap',currDir)%prints a screenshot and saves it in
                             %the work folder as new.bmp
end

%Callback for close
function closeCallback(h,eventdata)
    close(gcf);
end

end
```

Appendix B

GUI Lite User's Guide for Version 2

GUI Lite is a Graphical User Interface design tool which allows a user to create a graphical user interface for virtually any speech processing application with ease. It uses a two-stage approach in which the creation of the layout of the GUI and the actual writing of the callback code for the created GUI objects is separated into two stages to simplify GUI development. It automates the layout and positioning of various GUI objects and separates them from the writing of the code that controls the created objects. The GUI Lite User's Guide is a user manual that explains the capabilities of the GUI Lite – Version 2. The user's guide explains using step-by-step instructions, how to create a GUI that uses all the different types of buttons that can be created using GUI Lite. The user's guide also demonstrates how to create a GUI to select a speech file, play and plot it using the capabilities of GUI Lite.

The current User's Guide for GUI Lite – Version 2 is written using functions provided in MATLAB 7.8.0.347 (R2009a). GUI Lite – Version 2 is compatible with lower versions of MATLAB also.

Important vocabulary pertaining to GUI Lite

A GUI created using the GUI Lite toolkit consists of one or more of the following GUI objects.

1. Panel: A panel is a gray rectangular outline that is used to group or separate

buttons, plots, title boxes or a combination of all three. Panels are used to group UI objects (buttons or plots) performing similar functions together. Panels improve the layout of the GUI by providing a well-defined, more organized way of arranging the elements of the GUI.

2. Graphic Panel: A graphic panel is a plot window in which plots are drawn. It is different from the panel mentioned above. It is a white box and is sometimes enclosed by a 'panel' to provide better presentation.
3. Title Box: A title box is a gray rectangular box and is used to add a title for a panel, a graphic panel or a group of buttons. The text to be displayed in the title box is entered while writing callbacks (i.e., code implementations of the desired buttons) for the various GUI objects.
4. Button: A button is an object that is used to input a desired parameter to the GUI, perform a desired function when clicked, or provide options (e.g.: a drop down menu). The types of buttons that can be created using GUI Lite are as follows:
    - Pushbutton: This button performs a certain function when clicked i.e., it executes code in its callback function when clicked via the mouse. This is the default 'type' of any button created using GUI Lite.
    - Edit box: This kind of button is used to provide input to a GUI in MATLAB (e.g., the value of a variable). To create an edit box, the user must specify the type of button be created as an 'edit' button. The value in the edit box can be changed by moving the mouse to the edit box and editing the value in the box.

- Text button: This is a gray box that contains text. It can be used to hold a common title for a group of buttons or the label for a single button. The text in this box cannot be edited. Also, the text for this button is entered while writing the code for the callback functions.
- Popupmenu button: The popupmenu button is a pull down menu which contains a list of options for user selection (e.g., possible speech files within a directory for analysis). The user must enter the type of button as a 'popupmenu' button while creating the button.
- Slider button: This button is a horizontal slider button. It can be used for tasks such as volume control. The user must enter the type of button as a 'slider' button while creating this type of button.

5. 'Callback' functions: The 'uicontrol' object/function is used to create buttons for a GUI. The buttons created during the layout stage of GUI Lite do not perform the desired functions until the callback code for the buttons has been written. The 'callback' function is the backend code of the button that actually performs the function the button of the button.

   For example when 'button1' is clicked, the callback for 'button1' i.e. 'button1Callback' is called to execute the code that performs the function that button1 performs when it is clicked. This callback code is the application's code and is written by the user and needs to be entered in the button framework that is provided in the latter part of the 'PanelandButtonCallbacks.m' file. This will be explained in detail in the section on callbacks later in this user's guide.

   Figure 1 displays a screenshot of GUIwith all the GUI objects.

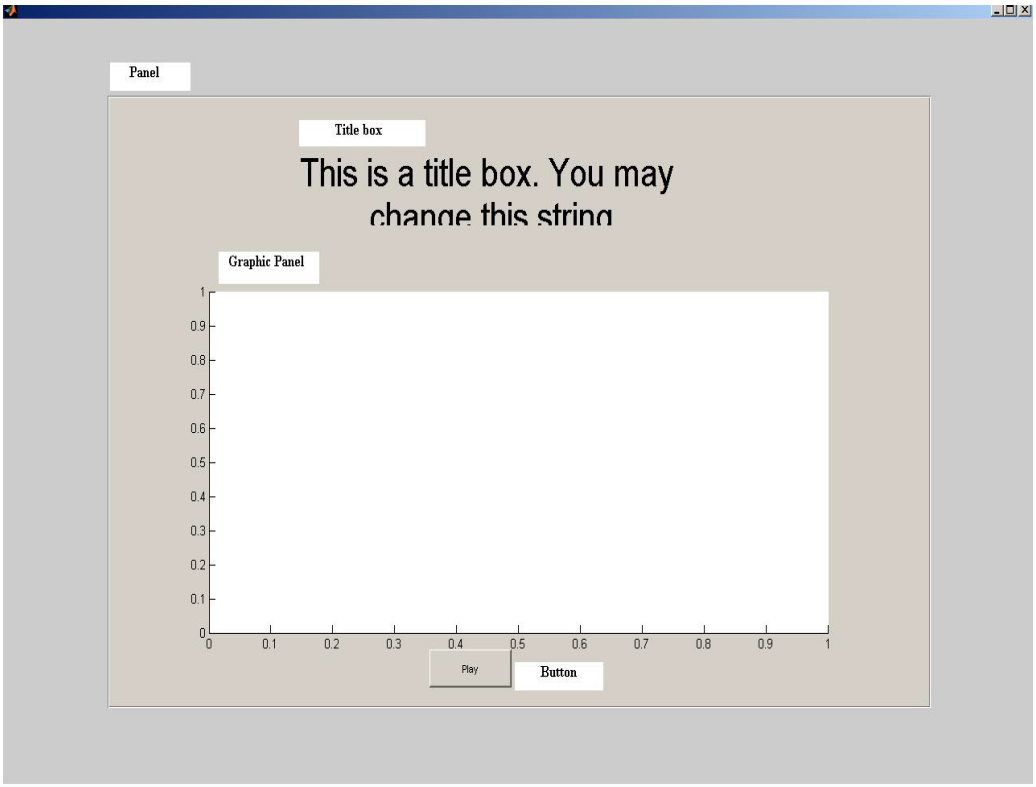

Figure 1: A dummy GUI with GUI objects including a panel, a graphic panel, a title box and one button.

Guidelines to use GUI Lite Solution 2 effectively

1. Before using GUI Lite, draw a sketch of the layout of the GUI on paper.
2. The GUI Lite folder should then be downloaded and saved into the 'work' folder of MATLAB. The GUI Lite folder contains three '.m' files. They are 'panelButtonSetup.m', 'runGUI.m' and 'PanelandButtonCallbacks.m'.
3. The GUI Lite toolbox is then launched by running the 'panelButtonSetup.m' file. On running the 'panelButtonSetup.m' file from the 'GUI Lite', the user is first asked to enter the number of panels that need to be drawn. An important point to remember is that panels do not include graphic panels (i.e. plot windows which are

used for plotting) but are just gray rectangular outlines that are used to group buttons, graphic panels and title boxes. The window that the user views on running the 'panelButtonSetup.m' file is displayed in Figure 2.

4. The user should also enter the number of graphic panels, title boxes and buttons in the order mentioned.
5. Once the user has entered these values, the user should change the name in the box 'Enter a name to save the file'. This box is used to provide a filename to save the co-ordinates obtained for the layout of the GUI. After entering the file name, the user clicks the 'Begin drawing GUI' button.

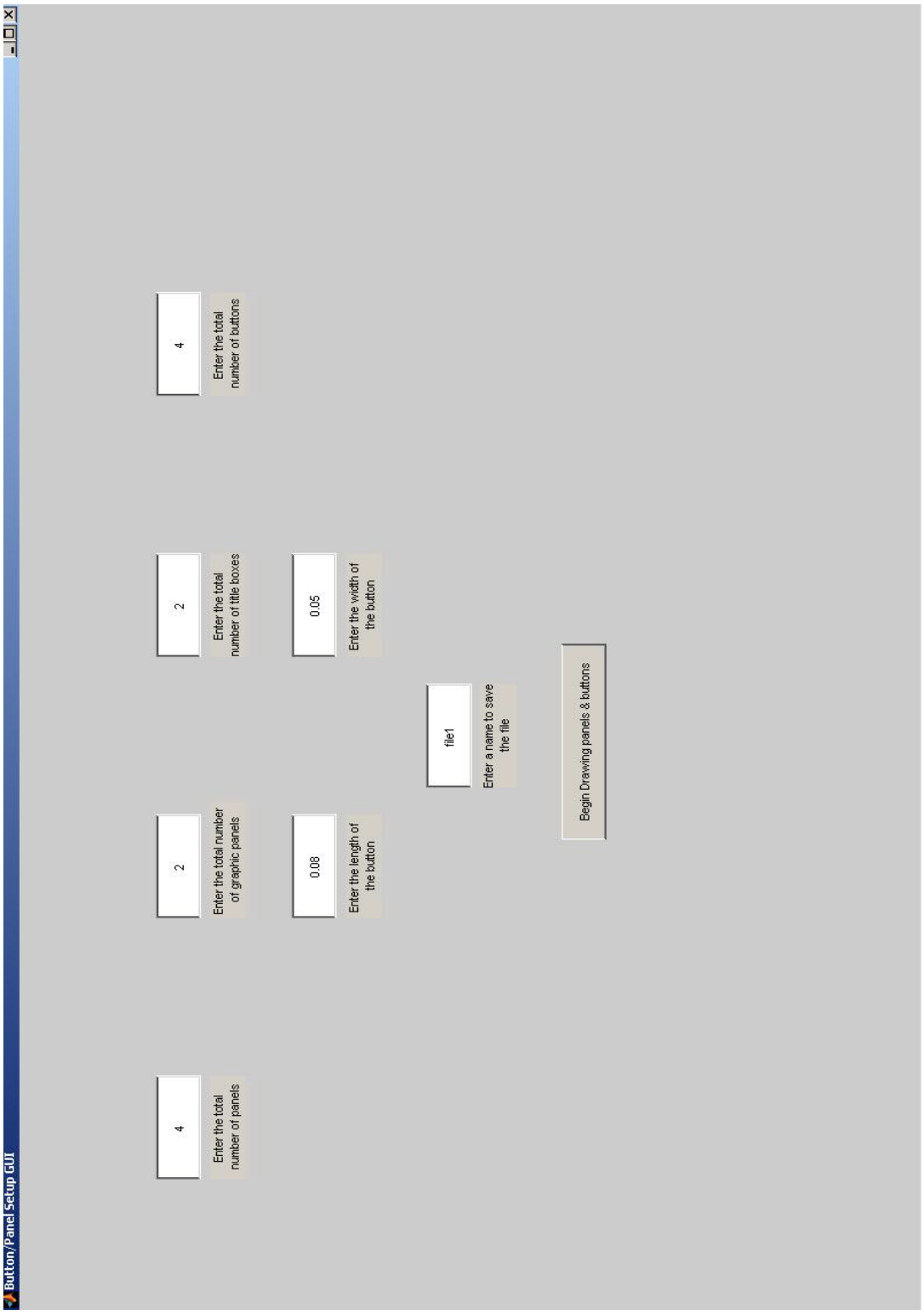

Figure 2: This is the first screen that appears once the GUI Lite toolbox is launched. It allows a user to enter parameters using which it draws the layout of a GUI. This launch screen contains default parameters.

To demonstrate the functionality of the GUI Lite toolkit, the user's guide will first explain:

I. An example to create a GUI which utilizes the different types of buttons that can be created using GUI Lite. The different types of buttons are a 'pushbutton', an 'edit' button, a 'text' button, a 'popupmenu' button and finally a 'slider' button. No callbacks will be written for these buttons. This exercise only demonstrates to the user the various types of buttons that can be created using GUI Lite.

II. The user's guide will also explain using an example, how to create a GUI to select, play and plot a desired speech file.

I. Example to create a GUI containing the five types of buttons that can be created using GUI Lite – Version 2.

After following the first three guidelines from the section on 'Guidelines to using the GUI Lite – Version 2 effectively', the user will be prompted to enter the number of panels, graphic panels, title boxes and buttons to be created. The user can enter zero as the number of panels, graphic panels and title boxes in the screen displayed in Figure 2. The user can enter five as the number of buttons that need be to be created and enter the file name to save the button parameters as 'buttonparameters'. After the GUI parameters have been entered by the user, the user should click on the 'Begin Drawing buttons and panels' button. After clicking the 'Begin drawing buttons and panels' button, a white screen with a grid overlaid on it and crosshairs will appear. Using the vertical and horizontal crosshairs that appear over the grid, the user can now select the

position of the first button after which the screen in Figure 3, i.e. the ‘Button Stats’ GUI will be visible to the user.

For the first button, the user can use the default values already present in the ‘Button Stats’ GUI. The type of button is ‘pushbutton’ , the string to be written on the button is ‘Play’ and the length and width of the button are set to default values. The ‘edit’ box in which a label can be entered for the button is left blank since ‘pushbuttons’ generally do not require labels since the string written on them explains their purpose.

Clicking the ‘Accept Specs Button’ will save the current button parameters and allow the user to select the position of the next button which is an ‘edit’ button. After entering the ‘edit’ button’s parameters the user can click the ‘Accept Specs Button’. Now, the white screen with the grid will reappear and on it a black rectangle which represents the previous ‘pushbutton’ will be drawn. Also horizontal and vertical crosshairs which let the user select the position of the next button will be available.

After selecting the position of second button, the ‘Button Stats’ GUI will appear allowing the user to select and enter the various button parameters. For the second button, the user needs to set the value of the drop down menu button to ‘edit’ to create an ‘edit’ button. Figure 4 displays the ‘Button Stats’ GUI with all the ‘edit’ button parameters added to it. ‘Edit’ buttons generally have labels below them which explain what purpose the ‘edit’ buttons serve. The width of the edit button has been increased to 0.08 units. This gives it a square like appearance. Figure 5 displays the GUI element selection grid (white screen with the grid) with the positions of the ‘pushbutton’ and the ‘edit’ buttons.

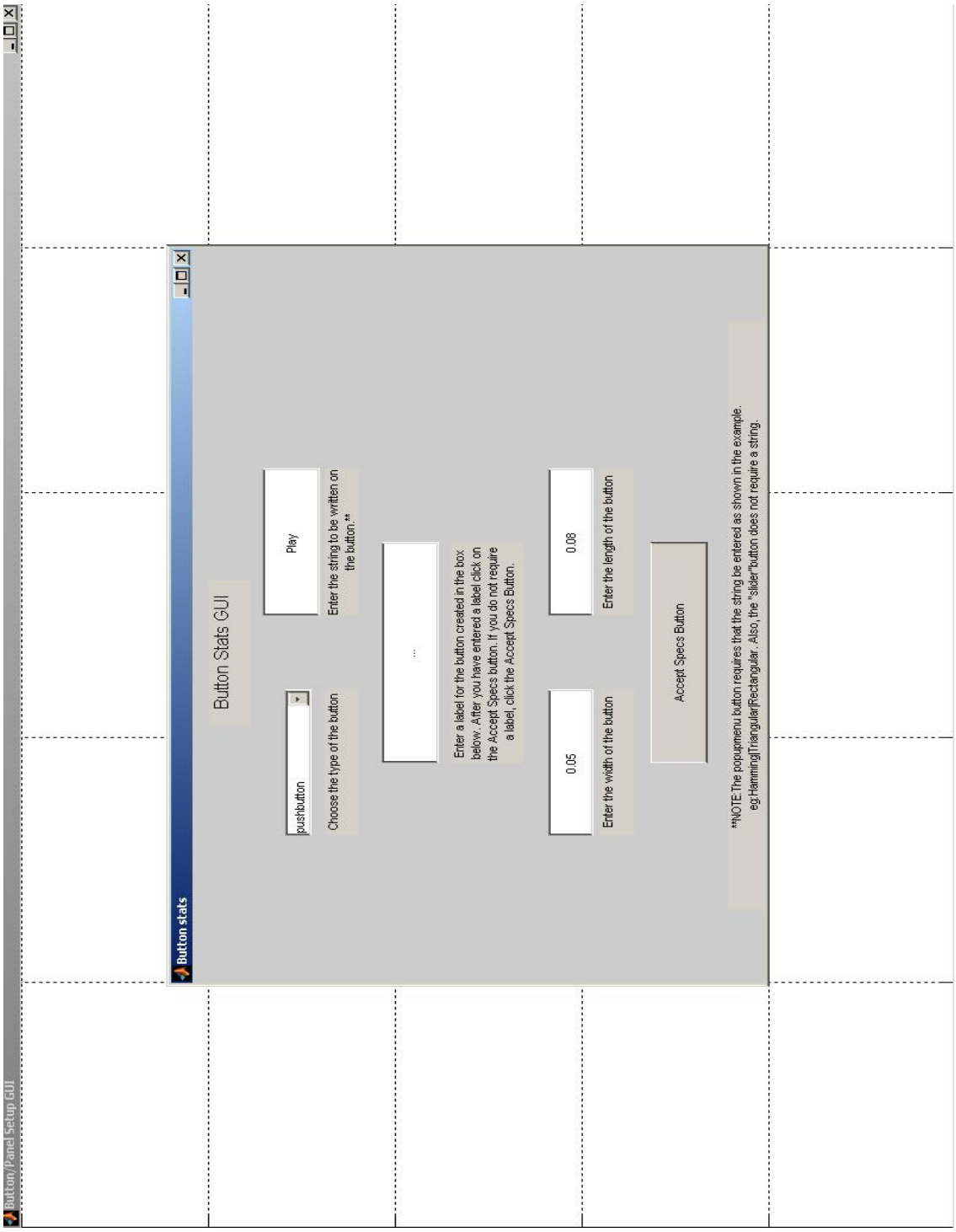

Figure 3: The 'Button Stats' GUI with the parameters of the first button, a 'pushbutton' entered into it.

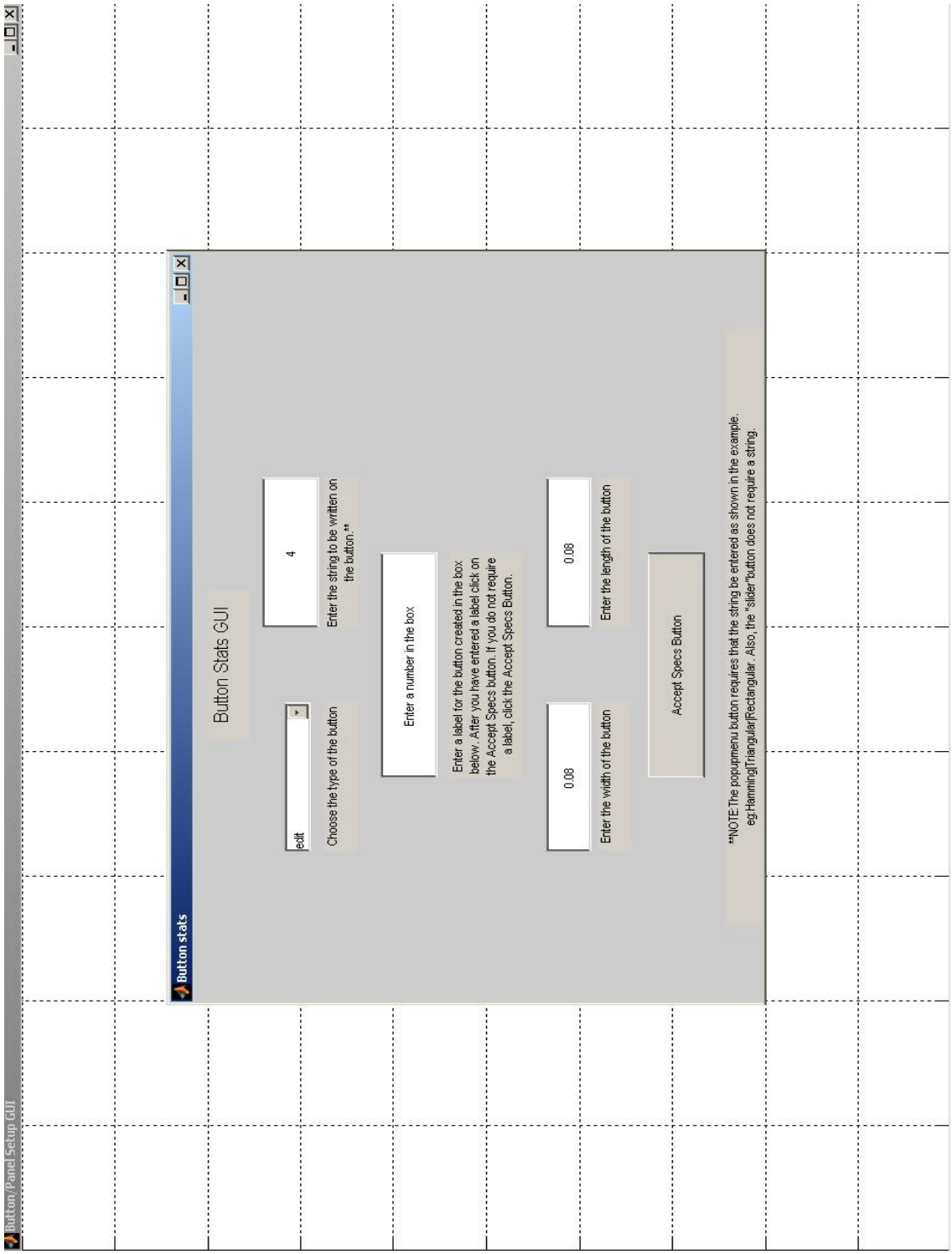

Figure 4: The 'Button Stats' GUI that appears after the position for the 'edit' button has been selected. It has all the parameters for the 'edit' button entered into it.

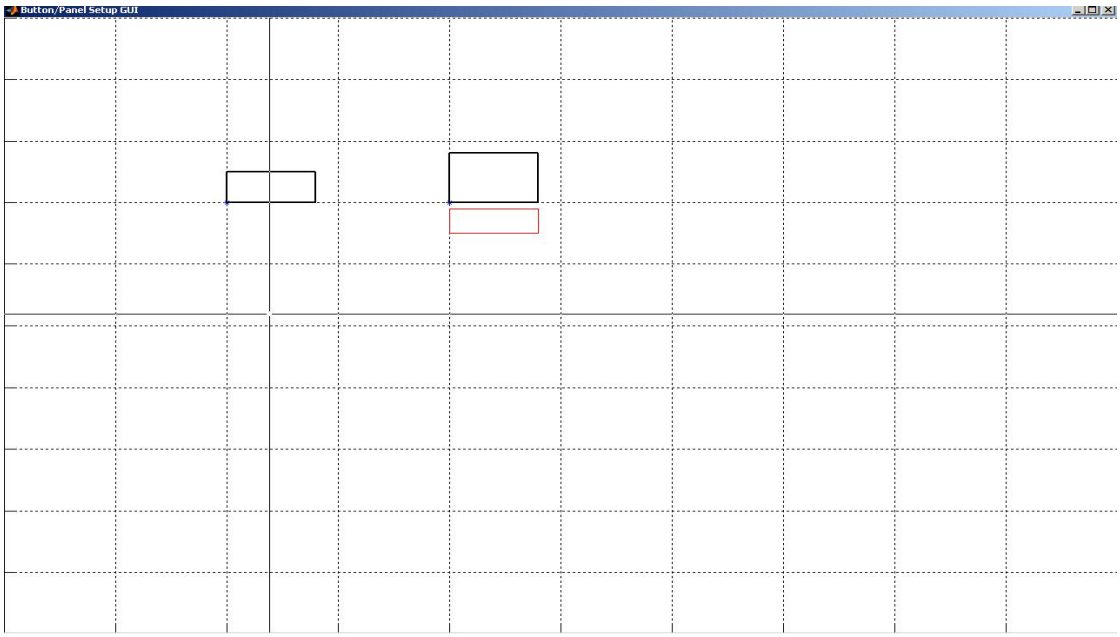

Figure 5: This white screen is visible to the user after the user has selected the positions of the first two buttons. The black rectangle towards the left represents the 'pushbutton' and the black rectangle on the right represents the 'edit' button. The red rectangle below the black rectangle on the right represents the 'label' for the 'edit' button. The vertical line which runs over the 'pushbutton' along with the horizontal line that intersects it are the crosshairs which are used for selecting the positions and dimensions of the GUI elements.

Next, the position of the third button, a 'text' button needs to be selected. After selecting the position of the 'text' button, the user enters the parameters for the 'text' button in the 'Button Stats' GUI. Figure 6 displays the 'Button Stats' GUI.

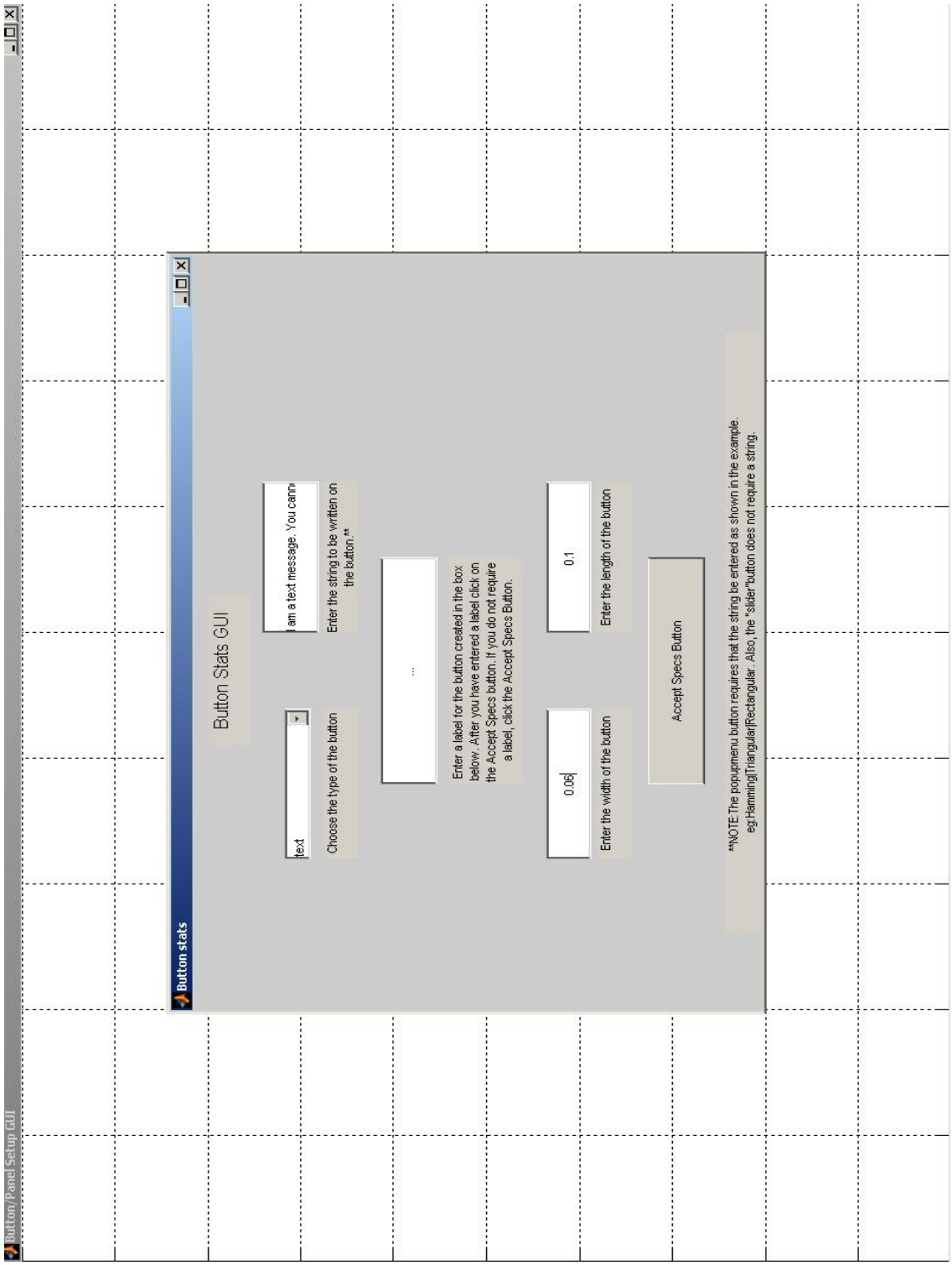

Figure 6: The 'Button Stats' GUI with the parameters of the 'text' button entered into it.

Similarly, after selection of the position of the ‘popupmenu’, which is the fourth button that is going to be created, the screen in Figure 7 will be displayed.

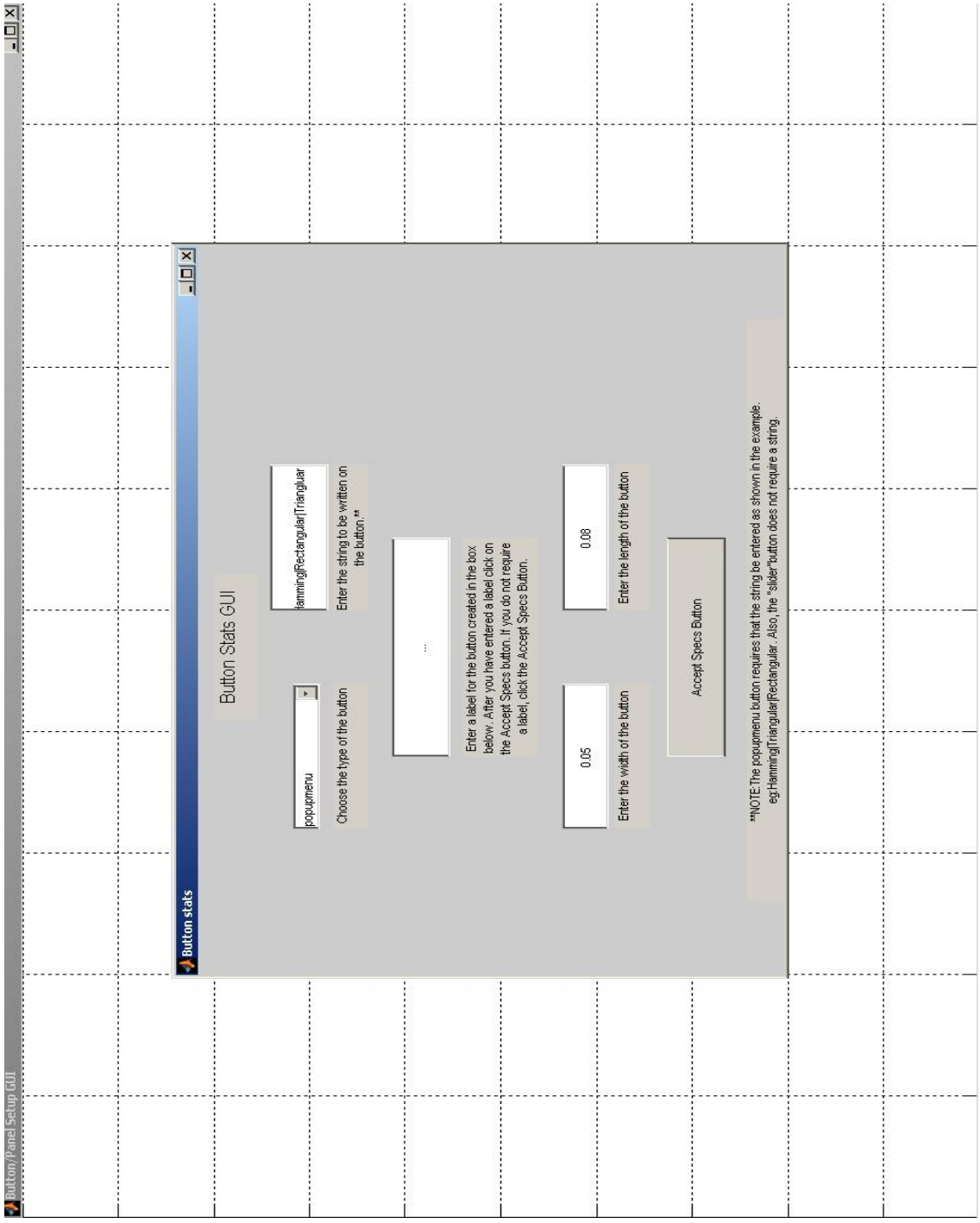

Figure 7: The ‘Button Stats’ GUI with the parameters of the ‘popupmenu’ entered into it.

The last button that needs to be created is the ‘slider’ button. Figure 8 displays the ‘Button Stats’ GUI with the parameters of the ‘slider’ button entered into it. A ‘slider’ button does not allow text to be written over it hence ‘Enter the string to be written on the button’ box in the GUI had been left blank. Also the length and width of the ‘slider’ button has been adjusted to create a longer, slimmer button.

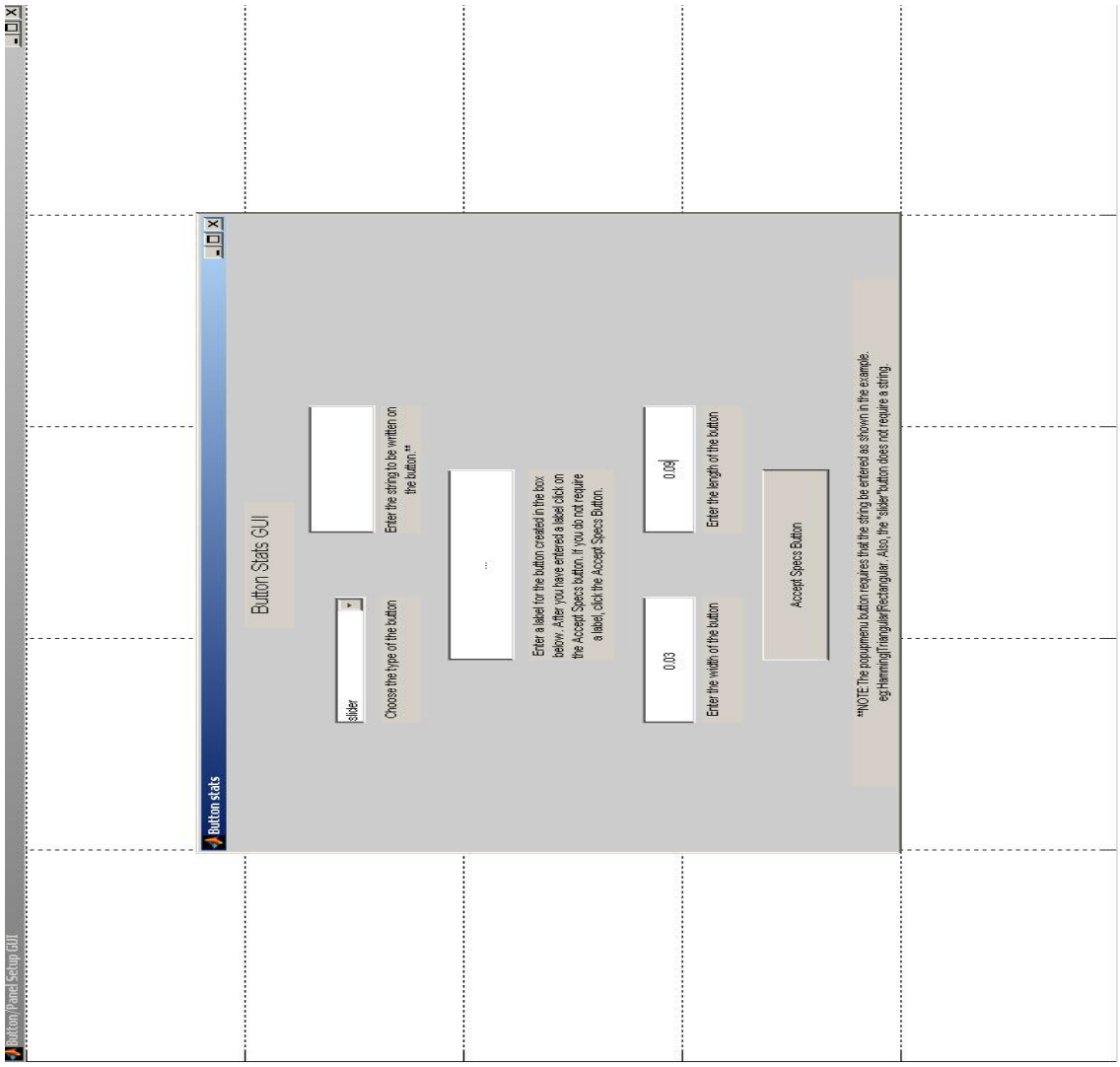

Figure 8: The ‘Button Stats’ GUI with the parameters of the slider button entered into it.

Figure 9 displays the white screen with the grid and the six rectangles that represent the buttons and their label. After all the buttons have been created, the user will be prompted to save the selection of buttons.

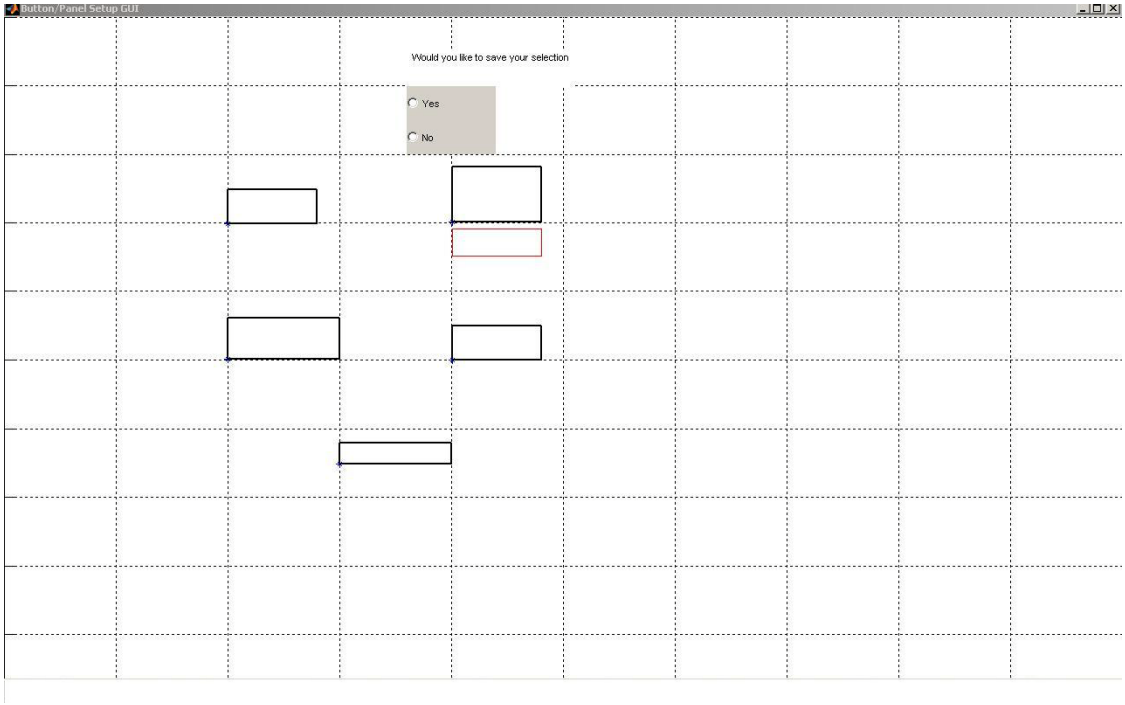

Figure 9: This screen contains the user selections of the button positions and dimensions. The user may either choose to save the selection or discard it and begin again.

On running the 'runGUI.m' file with the 'buttonparameters.mat' file that contains the above button selection, Figure 10 is displayed to the user. To run the 'runGUI.m' file with the 'buttonparameters.mat', the user must edit the 'runGUI.m' file and change the name of the '.mat' file to 'buttonparameters.mat'. After this change is completed, the 'runGUI' file must be saved and run. The code snippet to edit the 'runGUI.m' file is as follows. The user does not need to paste this code into 'runGUI.m' but can use it to understand how to edit the 'runGUI.m' file.

```
%ENTER THE NAME OF THE . mat FILE
    fileData=load('buttonparameters.mat');
```

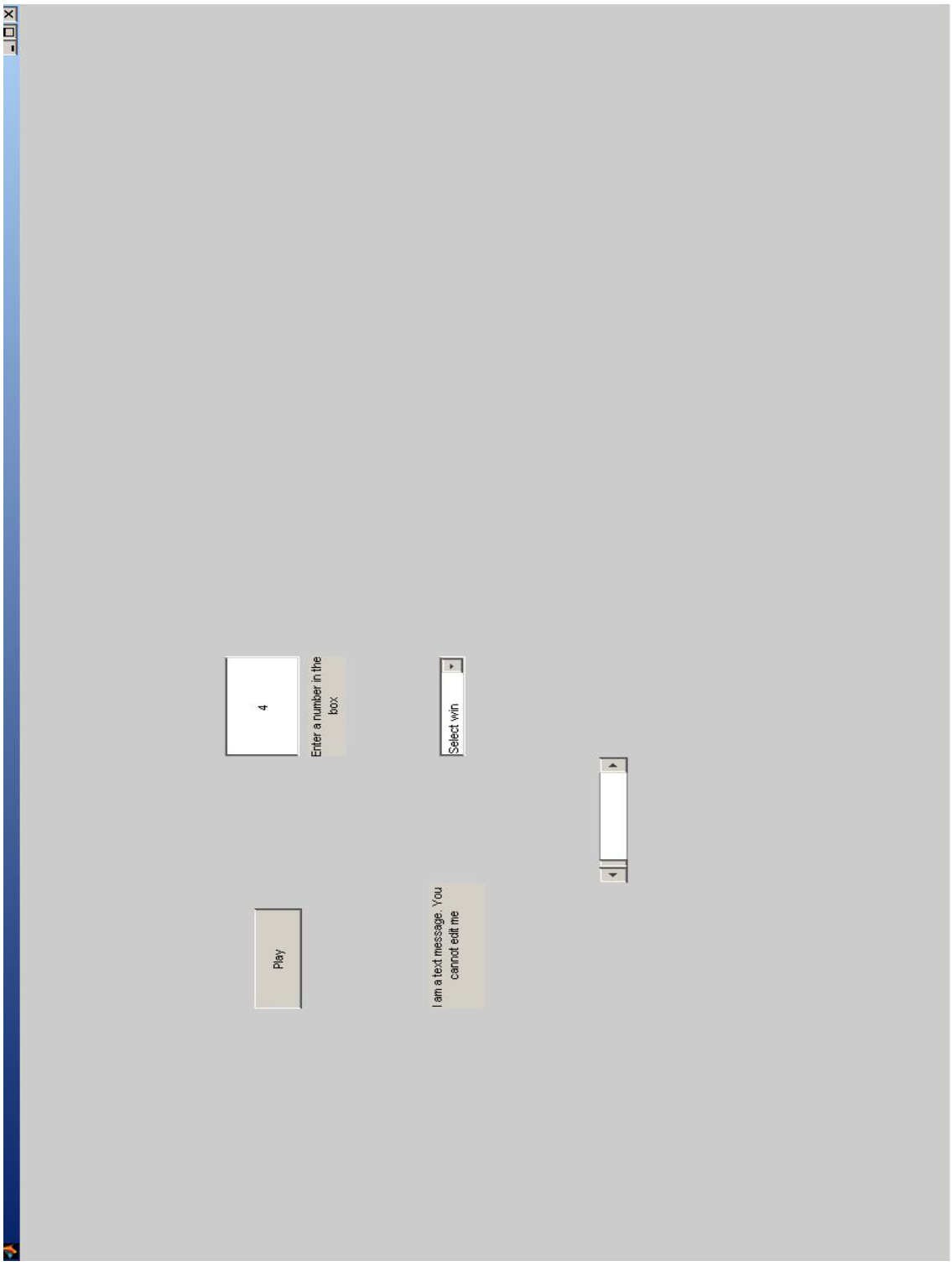

Figure 10: The GUI screen that is visible to the user when the 'runGUI.m' file is run with the current selection of button positions and dimensions.

At this stage none of the buttons work and if clicked will cause errors in MATLAB. In order for the buttons to work, callbacks for the buttons must be written.

II. An example which demonstrates how to create a GUI that plays and plots a selected .wav file using GUI Lite. It includes information on how to write callbacks for buttons created using GUI Lite.

The first step in implementing this example is to draw a sketch of the tentative layout of the GUI. Once this has been completed, the user should save the GUI Lite folder in the MATLAB 'work' folder and the 'panelButtonSetup.m' file must be run. On running the 'panelButtonSetup.m' file, the user will be able to see the screen displayed in Figure 2. For this example, the user can enter the number of panels as two, the number of graphic panels as one, the number of title boxes as one and finally number of buttons as four. The length and width of the buttons can be left at their default values. Here the user can use the name 'simpleGUI' as the name of the file that will contain the coordinates for the various GUI elements. Finally the user needs to click the 'Begin Drawing panels and buttons' button to start drawing the GUI layout. The screen that will be visible to the user at this stage is displayed in Figure 11.

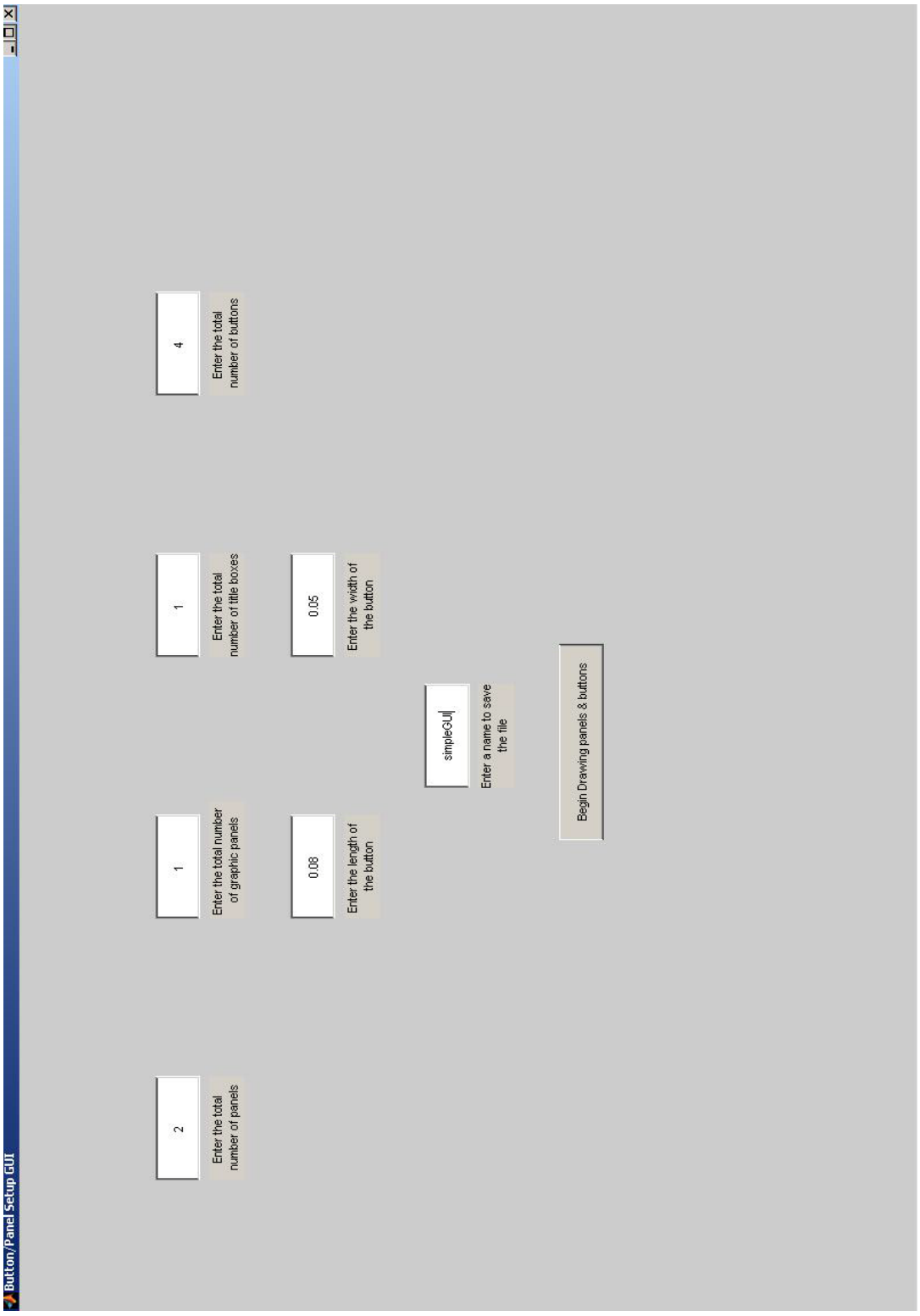

Figure 11: The GUI window containing the user's parameters for GUI development.

Once the ‘Begin Drawing buttons and panels’ button is clicked, a white screen with a grid will appear. Horizontal and vertical crosshairs will also appear over the grid. Using these crosshairs, the users selects the endpoints of the panels, the graphic panels, the title boxes and finally the positions of the buttons. The endpoints of the panels, graphic panels and title boxes need to be selected by first selecting the left bottom co-ordinate then the right bottom co-ordinate and finally the top right co-ordinate of the panels, graphic panels or the title boxes using mouse clicks. The user does not need to select the top left co-ordinate for any of the GUI objects since this is done automatically through GUI Lite. The grid laid out over the GUI layout window is to help select the endpoints of the various GUI objects accurately. Any change in the order of selecting the endpoints will cause a MATLAB error.

To select the position of a button object, the user needs to only to select the left bottom co-ordinate of the button object using the horizontal and vertical cursors and a mouse click. The button object with then be drawn automatically since a standard size button is assumed.

After the positions and properties of all the GUI elements have been selected and entered, the user will be asked to save the selection. If the user does not wish to save the selection, the user may begin selecting the coordinates of the GUI elements again.

While selecting the coordinates of the GUI elements, if the user makes a mistake in selection, there are three things that the user can do to rectify the situation.

1. Stop the program by closing the current window or by pressing the keys ‘Ctrl’ and ‘c’.

2. The user can choose to complete selecting all the coordinates and then click the option ‘No’ when asked to save the selection of coordinates. The user will then be allowed to reselect all the co-ordinates without re-entering the number of panels, graphic panels, title boxes and buttons.
3. The user can complete selecting all the coordinates and save the selection. Then while writing the callback code for the various GUI elements in the ‘panelButtonCallbacks.m’ file the user can reset the coordinates of the created GUI elements that need to be changed using the ‘set’ command.

   For example, if the user needs to change the position attribute of the title box, the user would use code similar to the one used below:

```
set(titleBox1,'Position', [u(1) v(1)+0.01 u(2)-u(1) v(3)-v(2)+0.01]);
```

   Here the u and v represent the x and y co-ordinates of the position of the title box. Each title box that is created has three co-ordinates stored for it into the ‘.mat’ file. u(1),v(1) are the co-ordinates of the first i.e. left bottom point, u(2),v(2) co-ordinates of the second i.e. right bottom point and u(3),v(3) the co-ordinates for the third i.e. right top point. In general, each GUI element like the panel, graphic panel and title box have a set a three points which contain their position and dimension. These three points are the lower left corner, lower right corner and the top right corner. Hence changing the position of any of the GUI elements just involves changing the co-ordinates involved in the ‘position’ attribute of the element that needs to be changed as done in the above code snippet. The code snippet is used to move the title box upwards by 0.01 units.

Similarly, to change the string attribute of the title box, the user would use the following command:

```
set(titleBox1,'String','I am the new title box');
```

The ‘string’ attribute of the title box is reset from its current value to the value ‘I am the new title box’. Here there is only one title box being used hence as per the naming convention followed by GUI Lite, this title box is referred to as ‘titleBox1’.

The screen that appears after the user has chosen to save his selection is displayed in Figure 12.

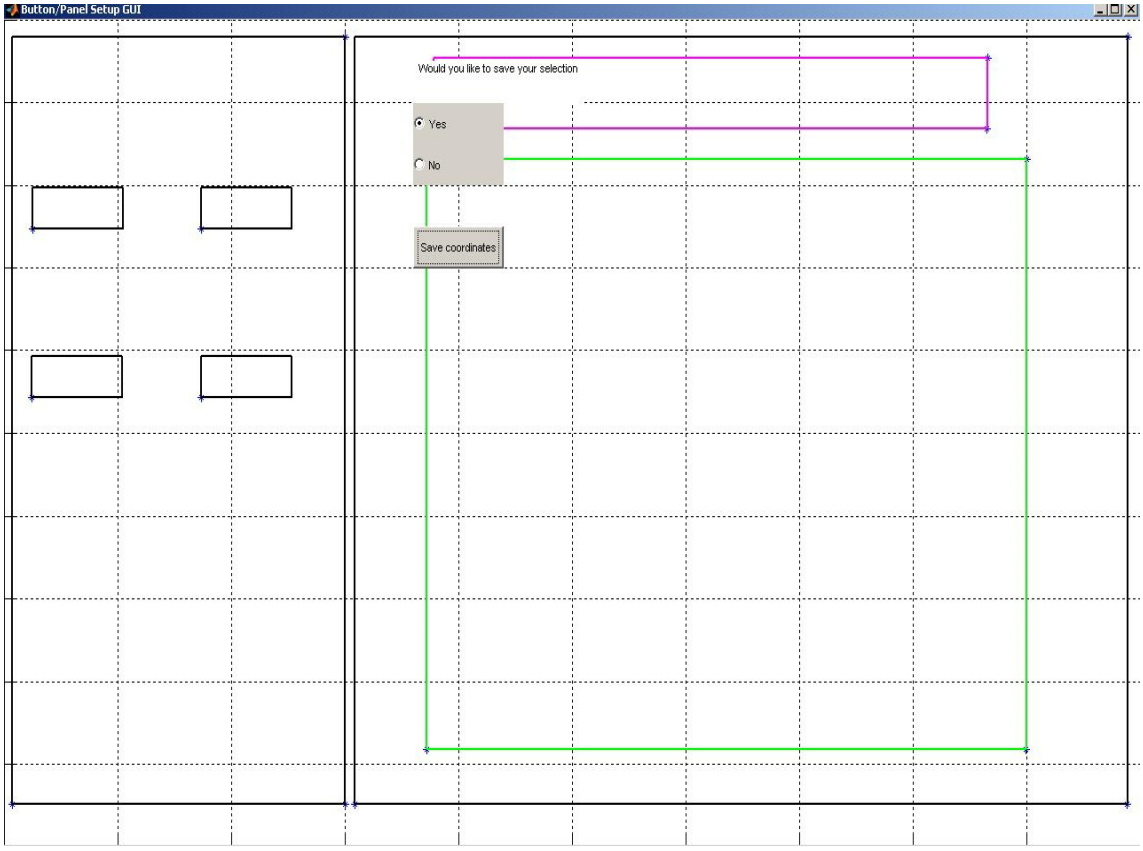

Figure 12: The GUI screen after the user has selected the positions of the panels, graphic panel, title boxes and button. The large black boxes are the panels. The green box is a graphic panel, the pink one is a title box and the small black rectangles are button.

Now that the layout of the GUI has been saved, the user needs to change the change the name of the file which needs to be loaded in the 'runGUI.m' from 'file1.mat' to 'simpleGUI.mat'. 'simpleGUI.mat' contains the layout of the GUI as selected and saved by the user. Once the user had updated the file name change in the 'runGUI.m' file, the 'runGUI.m' file can be saved and run. On running the 'runGUI.m' function, the screen in Figure 13 is displayed.

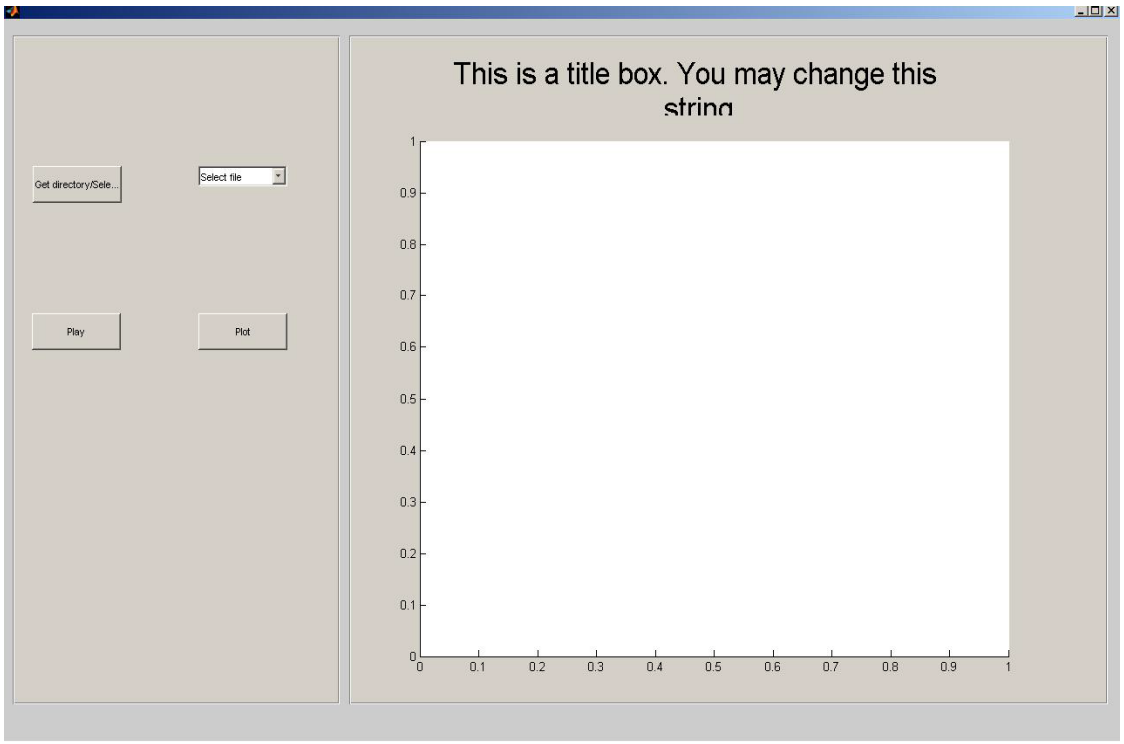

Figure 13: This is the screen that runs once the 'runGUI.m' file is loaded with the user's GUI layout.

If the user were to click any of the buttons in the GUI at this stage (Figure 13), none of them would work since the callbacks for the buttons have not been written as yet. Hence the next step to making the GUI a fully functioning one is to write the callbacks in the 'PanelandButtonCallbacks.m' file.

While writing the code for the callbacks, the user can set a name for the created GUI by using the ‘set’ function. The window that contains the GUI created by the user is referred to as ‘f’ by the GUI Lite toolbox. Using the ‘name’ attribute of the ‘set’ function the user can assign a name to it as follows.

```
set(f,'Name','Solution2-Play and Plot GUI');
```

The above code snippet needs to be inserted in the file ‘PanelandButtonCallbacks.m’ just below the comment that says ‘USER CODE FOR THE VARIABLES, CALLBACKS AND INITIALIZATION’.

The ‘PanelandButtonCallbacks.m’ file contains pre-defined callback frameworks for each of the buttons. For example, to complete writing a callback for a button, the user just needs to enter the callback code within the pre-defined function framework defined below.

```
%Callback for the button1
function  button1Callback(h,eventdata)

%Enter user's callback code

end
```

There are 15 such callback frameworks defined in the ‘PanelandButtonCallbacks.m’ file for buttons. Also as per GUI Lite’s naming convention, if three panels were to be created, they would be called ‘Panel1’, ‘Panel2’ and ‘Panel3’ depending on the order in which they were created. If there were two graphic panels created, they would be called ‘graphicPanel1’, ‘graphicPanel2’, etc. Similarly title boxes would be called titleBox1, titleBox2, etc and buttons would be called ‘button1’, ‘button2’, etc. If the user needs

more than 15 buttons, the GUI Lite toolkit can easily be scaled to accommodate the extra buttons by replicating the callback frameworks.

The callback code for button 1 is mentioned below. Button1 is used to browse the user's file system.

```
%Callback for the Get directory button(button1)
function button1Callback(src,eventdata)

        directory_name = uigetdir('start_path','dialog_title');
        A=strvcat(strcat((directory_name),'\*.wav'));
        struct_filenames=dir(A);
        wav_file_names={struct_filenames.name};
        set(button2,'String',wav_file_names);
        %once the popupmenu/drop down menu is created, by default, the
first
        %selection from the popupmenu/drop down menu must be loaded
even if the
        %callback for the popupmenu/drop down menu id not called

        indexOfDrpDwnMenu=1;%by default first option from the
popupmenu/dropdown
                        %menu will be loaded
        [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu);

end
```

Similarly, the callback code for the remaining buttons have been inserted into their respective callback frameworks. Button2 is a 'drop down menu' which is populated with the names of speech files present in a selected folder. The callback code for button2 is as follows.

```
%Call back for the Select file drop down menu (button2)
function button2Callback(src,eventdata)
```

```
        indexOfDrpDwnMenu=get(button2,'val');
        [curr_file,fs]=loadSelection(directory_name,...
            wav_file_names,indexOfDrpDwnMenu);
end
```

The ‘loadSelection’ function called in the ‘button2Callback’ is used to set the first entry that populates the ‘Select file’ drop down menu/popupmenu button as the value of the ‘curr_file’ variable which holds the speech array to be played. The user may either re-implement this function or use the function mentioned below.

```
%Function--load selection
function [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu)
        fin_path=strcat(directory_name,'\',...
            strvcat(wav_file_names(indexOfDrpDwnMenu)));
            %fin_path is the complete path of the file .wav file that
is
            %selected
        clear curr_file;
        clear fs;
        [curr_file, fs]=wavread(fin_path);
        FS=num2str(fs);
        %Information about the file being played
        file_info_string=strcat('Current file = ',...
            wav_file_names(indexOfDrpDwnMenu),...
            '. Sampling frequency = ',FS,'Hz',...
            '. Number of samples in file = ',...
            num2str(length(curr_file)));
        set(titleBox1,'String',file_info_string);
        set(titleBox1,'FontSize',0.3);
end
```

The callback code for ‘button3’ is mentioned below.

```
%Callback for the play button(button3)
function  button3Callback(h,eventdata)
        sound(curr_file,fs);
```

```
end
```

The callback code for ‘button4’ is mentioned below.

```
%callback for the plot button(button4)
function button4Callback(h,eventdata)
        hold off;  %It is essential to turn hold off so that the
        hold off;  %earlier contents of the panel are replaced
        %the two hold off's are for the speech file and the hamming
window

        grid off;
        reset(graphicPanel1);
        axes(graphicPanel1);
        l=length(curr_file);
        i=(1:l)/fs;%coverting samples to time
        plot(i,curr_file,'k','LineWidth',2),...
            xlabel('Time in seconds'),...
            ylabel('Amplitude');
        axis tight;
        grid on;
end
```

Some variables like the ones mentioned below need to be initialized so that they can be shared among the various callback functions.

```
curr_file=1;
fs=1;
directory_name='abcd';
wav_file_names='abcd';
```

Figure 14 displays what the GUI created by the user when he has clicked the ‘Get directory, ‘Select file’ and ‘Plot’ buttons.

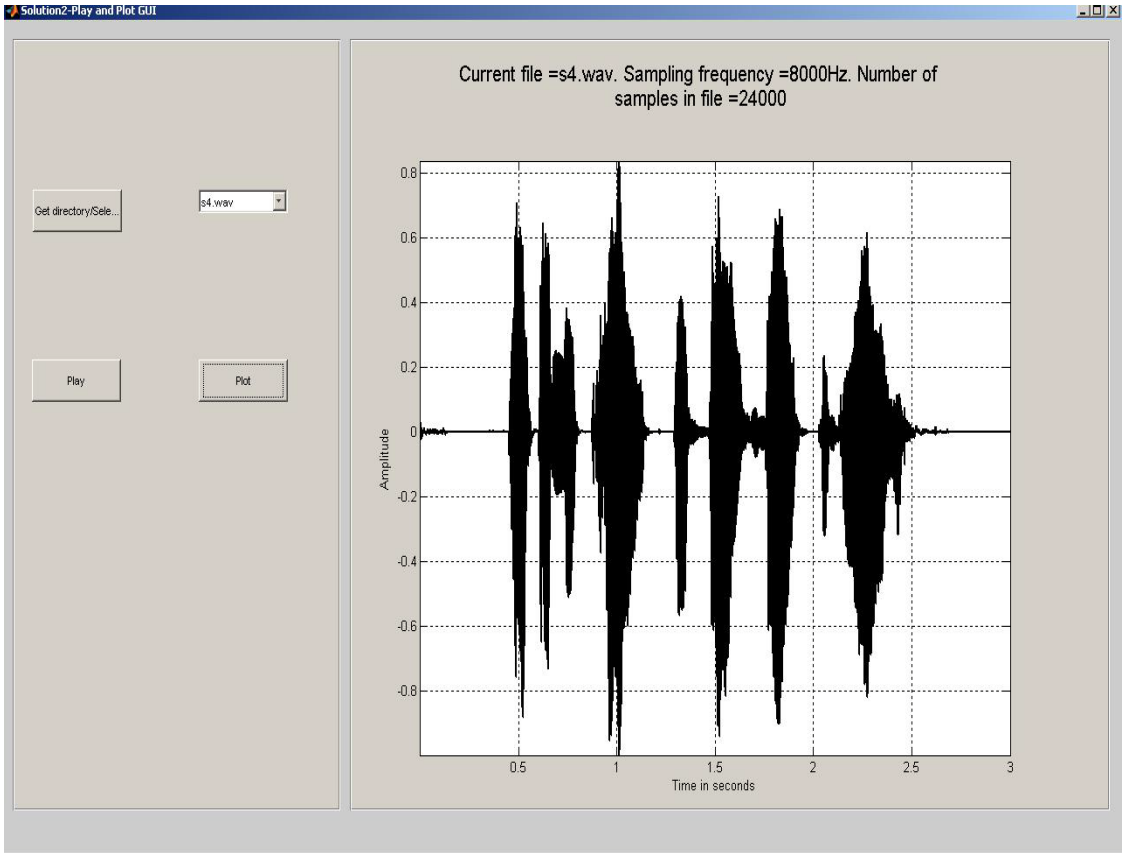

Figure 14: After selecting a '.wav' file using the 'Get directory' and 'Select file' buttons, the user can click the 'Play' and 'Plot' buttons in to hear the file and see it plotted.

Appendix C

Code for the four example GUI programs created using GUI Lite – Version 1

1. Program 1 – Hello World Program

```
function helloWorld
%embedded code for the GUI application
clc;clear all;
f = figure('Visible','on',...
            'Units','normalized',...
            'Position',[0,0,1,1],...
            'MenuBar','none',...
            'NumberTitle','off');
% Assign the GUI a name to appear in the window title.
set(f,'Name','Hello World');
%BUTTONS
% Push me button
pushMebutton=uicontrol('Parent',f,...
        'Units','Normalized',...
            'Position',[0.1 0.3 0.2 0.1],...
            'String','Push Me',...
            'Callback',@pushMeCallback);
%callback for the push me button
    function pushMeCallback(h,eventdata)
            msgbox('Hello World!','modal')
    end


end
```

2. Program 2 – Display the Waveform of a Designated Speech File.

```
function displaySpeechWaveform
%embedded code for the GUI application
clc;clear all;
f = figure('Visible','on',...
            'Units','normalized',...
            'Position',[0,0,1,1],...
```

```
            'MenuBar','none',...
            'NumberTitle','off');

% Assign the GUI a name to appear in the window title.
set(f,'Name','Display speech waveform GUI');

%GUI PANELS
%This GUI is divided into two panels,a panel to group the buttons and
a
%panel to enclose the graphic panel.
panel1=uipanel('Parent',f,...
            'Units','Normalized',...
            'Position',[0.1 0.05 0.75 0.2]);%button panel

panel2=uipanel('Parent',f,...
            'Units','Normalized',...
            'Position',[0.1 0.3 0.75 0.65]);%plot window
%                                           %(graphic panel) panel

%The speech waveform will be displayed within graphicPanel.
graphicPanel = axes('parent',panel2,...
            'Units','Normalized',...
            'Position',[0.1 0.3 0.8 0.5],...
            'GridLineStyle','--');

%BUTTONS
% Display speech waveform button
displaySpeechbutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.1 0.3 0.2 0.3],...
            'String', 'Display speech waveform',...
            'Callback',@displaySpeechCallback);
% Close GUI button
closebutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.6 0.3 0.2 0.3],...
            'String', 'Close GUI',...
```

```
            'Callback',@closeCallback);

%callback for the display image button
    function displaySpeechCallback(h,eventdata)
            loadedSpeech=wavread('s1.wav');
            %The speech file is 's1.wav'
            axes(graphicPanel);
            plot(loadedSpeech);
            title('s1.wav');
            xlabel('Time in seconds');
            ylabel('Amplitude');
    end
%callback for the close GUI button
    function closeCallback(h,eventdata)
            close(gcf);
    end

end
```

3. Program 3 – Load a Speech File, Play it Back and Display the Waveform.

```
function playPlotSpeechGUI
clc;clear all;
curr_file=1;
fs=1;
directory_name='ABCD';
wav_file_names='ABCD';
file_info_string='ABCD';

f = figure('Visible','on',...
            'Units','normalized',...
            'Position',[0,0,1,1],...
            'MenuBar','none',...
            'NumberTitle','off');
% Assign the GUI a name to appear in the window title.
set(f,'Name','Play and plot speech GUI');
%GUI PANELS
%This GUI is divided into two panels
```

```
panel1=uipanel('Parent',f,...
            'Units','Normalized',...
            'Position',[0.1 0.05 0.75 0.35]);%button panel

panel2=uipanel('Parent',f,...
            'Units','Normalized',...
            'Position',[0.1 0.45 0.75 0.5]);%plot window
                                        %(graphic panel) panel

%The image will be displayed within graphicPanel.
graphicPanel = axes('parent',panel2,...
            'Units','Normalized',...
            'Position',[0.1 0.2 0.8 0.7],...
            'GridLineStyle','--');

%BUTTONS
% Get directory button
getDirectorybutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.25 0.55 0.2 0.25],...
            'String', 'Get directory/Select file',...
            'Callback',@getDirectoryCallback);

% Select file button
selectFilebutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.55 0.45 0.2 0.25],...
            'style','popupmenu',...
            'BackgroundColor','white',...
            'String', 'Select file',...
            'Callback',@selectFileCallback);

 %Play  button
playbutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.1 0.1 0.2 0.25],...
            'String', 'Play',...
```

```
            'Callback',@playCallback);

%Plot button
plotbutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.4 0.1 0.2 0.25],...
            'String', 'Plot',...
            'Callback',@plotCallback);

% Close GUI button
closebutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.7 0.1 0.2 0.25],...
            'String', 'Close GUI',...
            'Callback',@closeCallback);

    %Callbacks
    %Get directory callback
    function getDirectoryCallback(src,eventdata)
        directory_name = uigetdir('start_path','dialog_title');
        A=strvcat(strcat((directory_name),'\*.wav'));
        struct_filenames=dir(A);
        wav_file_names={struct_filenames.name};
        set(selectFilebutton,'String',wav_file_names);
        %once the popupmenu/drop down menu is created, by default, the
first
        %selection from the popupmenu/drop down menu must be loaded
even if the
        %callback for the popupmenu/drop down menu id not called

        indexOfDrpDwnMenu=1;%by default first option from the
popupmenu/dropdown
                        %menu will be loaded
        [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu);
    end
```

```
    %Select file callback
    function selectFileCallback(src,eventdata)
        indexOfDrpDwnMenu=get(selectFilebutton,'val');
        [curr_file,fs]=loadSelection(directory_name,...
            wav_file_names,indexOfDrpDwnMenu);
    end

    %Function--load selection
    function [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu)
        fin_path=strcat(directory_name,'\',...
            strvcat(wav_file_names(indexOfDrpDwnMenu)));
            %fin_path is the complete path of the file .wav file that
is
            %selected
        clear curr_file;
        clear fs;
        [curr_file, fs]=wavread(fin_path);
        FS=num2str(fs);
        %Information about the file being played
        file_info_string=strcat('Current file = ',...
            wav_file_names(indexOfDrpDwnMenu),...
            '. Sampling frequency = ',FS,'Hz',...
            '. Number of samples in file = ',...
            num2str(length(curr_file)));
     end

    %Callback for the playbutton
    function  playCallback(h,eventdata)
        sound(curr_file,fs);
        reset(graphicPanel);
        temp=0;
        plot(temp);
        hold off;
        hold off;

    end
```

```
    %callback for the plotbutton
     function plotCallback(h,eventdata)
        hold off;  %It is essential to turn hold off so that the
        hold off;  %earlier contents of the panel are replaced
        %the two hold off's are for the speech file and the hamming
window
        grid off;
        reset(graphicPanel);
        axes(graphicPanel);
        l=length(curr_file);
        i=(1:l)/fs;%coverting samples to time
        plot(i,curr_file,'k','LineWidth',2),...
            xlabel('Time in seconds'),...
            ylabel('Amplitude');
        title(file_info_string);
        axis tight;
        grid on;
     end

    %Callback for close
    function closeCallback(h,eventdata)
        close(gcf);
    end
end
```

4. Program 4 – Load an Existing Speech File or Record a New Speech File. Play the File and Display a Waveform and Save the File.

```
function recordGUI
%embedded code for the GUI application
clc;clear all;

%INITIALIZATION
%The variable returned from the edit boxes must
%be initialized, else the box will only display
%the value but not actually hold that value
```

```
curr_file=1;
directory_name='ABCD';
wav_file_names='ABCD';
y=1;%y is the variable that contains the recorded speech
nsec=3;
fs=8000;
fileName='file1';

%The 1 1 in the position attribute means that the GUI
%is fit to screen
f = figure('Visible','on',...
    'Units','normalized',...
    'Position',[0,0,1,1],...
    'MenuBar','none',...
    'NumberTitle','off');

%GUI PANELS
%This GUI is divided into four panels
panel1=uipanel('Parent',f,...
    'Units','Normalized',...
    'Position',[0.3 0.65 0.3 0.345]);%top panel
panel2=uipanel('Parent',f,...
    'Units','Normalized',...
    'Position',[0.3 0.245 0.3 0.4]);%center panel
panel3=uipanel('Parent',f,...
    'Units','Normalized',...
    'Position',[0.3 0.04 0.3 0.2]);%bottom panel



% Assign the GUI a name to appear in the window title.
set(f,'Name','Record speech GUI');

%Initialize GUI
set([f,panel1,panel2,panel3],'Units','normalized')

%BUTTONS
% Get directory button
```

```
getDirectorybutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.05 0.6 0.35 0.2],...
            'String', 'Get directory/Select file',...
            'Callback',@getDirectoryCallback);

% Select file button
selectFilebutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.55 0.5 0.35 0.25],...
            'style','popupmenu',...
            'BackgroundColor','white',...
            'String', 'Select file',...
            'Callback',@selectFileCallback);

%Play  button
playbutton=uicontrol('Parent',panel1,...
            'Units','Normalized',...
            'Position',[0.3 0.2 0.35 0.2],...
            'String', 'Play speech',...
            'Callback',@playCallback);

%Enter sampling frequency for recording in Hz button
fsbutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.05 0.8 0.35 0.15],...
            'style','edit',...
            'String', '8000',...
            'BackgroundColor','white',...
            'Callback',@fsCallback);

%Label for 'Enter sampling frequency for recording in Hz' button
fsLabelbutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.05 0.63 0.35 0.15],...
            'style','text',...
            'String', 'Enter sampling frequency for recording in Hz');
```

```
 %Enter the number of seconds for recording button
nsecbutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.55 0.8 0.35 0.15],...
            'style','edit',...
            'String', '3',...
            'BackgroundColor','white',...
            'Callback',@fsCallback);

%Label for 'Enter the number of seconds for recording' button
nsecLabelbutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.55 0.63 0.35 0.15],...
            'style','text',...
            'String', 'Enter the number of seconds for recording');

%Record/re-record button
recordbutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.3 0.5 0.35 0.15],...
            'String', 'Record/re-record',...
            'Callback',@recordCallback);

%Enter a file name to save the recorded speech button
fileNamebutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.05 0.2 0.35 0.15],...
            'style','edit',...
            'String', 'file1',...
            'BackgroundColor','white',...
            'Callback',@fileNameCallback);

%Label for 'Enter a file name to save the recorded speech' button
filenameLabelbutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.05 0.03 0.35 0.15],...
```

```
            'style','text',...
            'String',  'Enter  a  file  name  to  save  the  recorded
speech');



%Save speech  button
savebutton=uicontrol('Parent',panel2,...
            'Units','Normalized',...
            'Position',[0.55 0.2 0.35 0.15],...
            'String', 'Save speech',...
            'Callback',@saveCallback);




% Close GUI button
closebutton=uicontrol('Parent',panel3,...
            'Units','Normalized',...
            'Position',[0.3 0.25 0.35 0.4],...
            'String', 'Close GUI',...
            'Callback',@closeCallback);

    %Callbacks

    function getDirectoryCallback(src,eventdata)
        directory_name = uigetdir('start_path','dialog_title');
        A=strvcat(strcat((directory_name),'\*.wav'));
        struct_filenames=dir(A);
        wav_file_names={struct_filenames.name};
        set(selectFilebutton,'String',wav_file_names);
        %once the popupmenu/drop down menu is created, by default, the
first
        %selection  from  the  popupmenu/drop  down  menu  must  be  loaded
even if the
        %callback for the popupmenu/drop down menu id not called
```

```
        indexOfDrpDwnMenu=1;%by   default   first   option   from   the
popupmenu/dropdown
                        %menu will be loaded
        [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu);
    end

    function selectFileCallback(src,eventdata)
        indexOfDrpDwnMenu=get(selectFilebutton,'val');
        [curr_file,fs]=loadSelection(directory_name,...
            wav_file_names,indexOfDrpDwnMenu);
    end
    %Function--load selection
    function [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu)
        fin_path=strcat(directory_name,'\',...
            strvcat(wav_file_names(indexOfDrpDwnMenu)));
            %fin_path is the complete path of the file .wav file that
is
            %selected
        clear curr_file;
        clear fs;
        [curr_file, fs]=wavread(fin_path);
        FS=num2str(fs);
        %Information about the file being played
        file_info_string=strcat('Current file = ',...
            wav_file_names(indexOfDrpDwnMenu),...
            '. Sampling frequency = ',FS,'Hz',...
            '. Number of samples in file = ',...
            num2str(length(curr_file)));
    end

 %Callback for the playbutton
    function  playCallback(h,eventdata)
        sound(curr_file,fs);
    end
```

```
%callback for the fs button
function fsCallback(h,eventdata)
    fs=str2num(get(fsbutton,'string'));
end

%callback for the nsec button
function nsecCallback(h,eventdata)
    nsec=str2num(get(nsecbutton,'string'));
end

%callback for the record/ re-record button
%record speech file of fixed duration (nsec) and
%given sampling rate(fs)

function recordCallback(h,eventdata)
   fsCallback(h,eventdata);
   nsecCallback(h,eventdata);
% yn=speech samples normalized to 1
% N is the number of samples in each speech file
% ch is the number of channels in the recording
    N=fs*nsec;
    ch=1;
    y=wavrecord(N,fs,ch,'double');

    ymin=min(y);
    ymax=max(y);

    % calculate dc offset and correct
    offset=sum(y(N-999:N))/1000;
    y=y-offset;
    sound(y,fs);
end

%callback for filename speech
function fileNameCallback(h,eventdata)
    fileName=get(fileNamebutton,'string');
end
```

```
    %callback for save speech
    function saveCallback(h,eventdata)
        currentDir=pwd
        currDir=strcat(currentDir,'\',fileName,'.wav')
        wavwrite(y,fs,strvcat(currDir));
        c=wavread(strvcat(currDir));
        soundsc(c,fs)
    end

    %Callback for close
    function closeCallback(h,eventdata)
        close(gcf);
    end

end
```

Appendix D

Code for the four example GUI programs created using GUI Lite Version 2

When creating programs using the GUI Lite – Version 2, the user needs to save the GUI Lite folder into the MATLAB work folder. The GUI Lite folder contains three '.m' files.

Their names are as follows:

- panelButtonSetup.m
- PanelandButtonCallbacks.m
- runGUI.m

The panelButtonSetup.m file is not edited by the user and stays the same for any GUI program being created. The '.mat' file that is generated by the 'panelButtonSetup.m' file is not included in this thesis since it is not in human readable format. The user needs to create the '.mat' file in order to test these examples. The code for the 'panelButtonSetup.m' file is included below:

The 'panelButtonSetup.m' file.

```
function panelButtonSetup
clc;
clear all;

%DECLARATIONS
% lButton;%length of a button
% wButton;%width of a button
% x;      %x co ordinates of the panel
% y;      %y co ordinates of the panel
% m;      %left bottom x co ordinate of the button
% n;      %left bottom y co ordinate of the button
global ENTERSTYLEOFBUTTON;%element of a cell array in which the STYLE
                    %of the button is stored
```

```
global ENTERSTRINGOFBUTTON;%element of a cell array in which the
STRING
                    %of the button is stored
global ENTERLABELOFBUTTON;
global FLAGFORPAUSE; %for the pause function after the button stats
GUI appears
                    %it is related to the accept specs button
global YESBUTTON;
global NOBUTTON;
global SAVETEXTBOX;
global SAVEBUTTON;

%INITIALIZATION
move=0.1;%the buttons move by this amount
labelDist=0.05;%distance that the label is below the button
noPanels=4;
noGraphicPanels=2;
noButtons=4;
noTitles=2;
lButton=0.08;

wButton=0.05;
%labelHeight=0.8*wButton;
labelHeight=0.8*0.05;
writefilename='file1';%name of the file in which the name
               %of the file is stored
typeIndex=1;%index for the popupmenu which allows you to choose the
type
           %of button. typeIndex=1=pushbutton,2=edit,etc.
FLAGFORPAUSE=0;%for the pause function after the button stats GUI
              %appears it is related to the accept specs button

f = figure('Visible','on',...
    'Units','normalized',...
    'Position',[0 0 1 1],...
    'MenuBar','none',...
    'NumberTitle','off');
```

```
set(f,'Name','Button/Panel Setup GUI');

%Enter the number of panels
numberOfPanels=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Style','edit',...
    'Position',[0.1 0.9-move 0.08 0.05],...
    'Backgroundcolor','w',...
    'String','4',...
    'Callback',@noPanelsCallback);
Label1=uicontrol('parent',f,...
    'Units','Normalized',...
    'Position',[0.1 0.85-move 0.08 0.04],...
    'Style','text',...
    'String','Enter the total number of panels');

%Enter the number of panels
numberOfGraphicPanels=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Style','edit',...
    'Position',[0.3 0.9-move 0.08 0.05],...
    'Backgroundcolor','w',...
    'String','2',...
    'Callback',@noGraphicPanelsCallback);
Label2=uicontrol('parent',f,...
    'Units','Normalized',...
    'Position',[0.3 0.85-move 0.09 0.04],...
    'Style','text',...
    'String','Enter the total number of graphic panels');

%Enter the number of title boxes
numberOfTitles=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Style','edit',...
    'Position',[0.5 0.9-move 0.08 0.05],...
    'Backgroundcolor','w',...
    'String','2',...
```

```
    'Callback',@noTitlesCallback);
Label4=uicontrol('parent',f,...
    'Units','Normalized',...
    'Position',[0.5 0.85-move 0.08 0.04],...
    'Style','text',...
    'String','Enter the total number of title boxes');

%Enter the number of buttons
numberOfButtons=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Style','edit',...
    'Position',[0.7 0.9-move 0.08 0.05],...
    'Backgroundcolor','w',...
    'String','4',...
    'Callback',@noButtonsCallback);
Label3=uicontrol('parent',f,...
    'Units','Normalized',...
    'Position',[0.7 0.85-move 0.08 0.04],...
    'Style','text',...
    'String','Enter the total number of buttons');

%Enter the length
lengthButton=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Style','edit',...
    'Position',[0.3 0.75-move 0.08 0.05],...
    'Backgroundcolor','w',...
    'String','0.08',...
    'Callback',@lButtonCallback);
Label5=uicontrol('parent',f,...
    'Units','Normalized',...
    'Position',[0.3 0.7-move 0.08 0.04],...
    'Style','text',...
    'String','Enter the length of the button');

%Enter the width
widthButton=uicontrol('Parent',f,...
```

```
    'Units','Normalized',...
    'Style','edit',...
    'Position',[0.5 0.75-move 0.08 0.05],...
    'Backgroundcolor','w',...
    'String','0.05',...
    'Callback',@wButtonCallback);
Label6=uicontrol('parent',f,...
    'Units','Normalized',...
    'Position',[0.5 0.7-move 0.08 0.04],...
    'Style','text',...
    'String','Enter the width of the button');

% Enter a file name for the save option
fileName=uicontrol('parent',f,...
    'Units','normalized',...
    'Position',[0.4 0.6-move 0.08 0.05],...
    'Style','edit',...
    'String','file1',...
    'BackgroundColor','white',...
    'Callback',@fileNameCallback);
Label7=uicontrol('parent',f,...
    'Units','normalized',...
    'Position',[0.4 0.55-move 0.08 0.04],...
    'Style','text',...
    'String','Enter a name to save the file');

%Begin drawing
beginDraw=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[0.36 0.45-move 0.15 0.05],...
    'String','Begin Drawing panels & buttons',...
    'Callback',@beginDrawCallback);



%CALLBACKS
%callback for begin drawing
 function beginDrawCallback(h,eventdata)
```

```
    %turn all the buttons and edit boxes off
    set(numberOfPanels,'Visible','off');
    set(numberOfGraphicPanels,'Visible','off');
    set(numberOfButtons,'Visible','off');
    set(numberOfTitles,'Visible','off');
    set(lengthButton,'Visible','off');
    set(widthButton,'Visible','off');
    set(Label1,'Visible','off');
    set(Label2,'Visible','off');
    set(Label3,'Visible','off');
    set(Label4,'Visible','off');
    set(Label5,'Visible','off');
    set(Label6,'Visible','off');
    set(beginDraw,'Visible','off');
    set(fileName,'Visible','off');
    set(Label7,'Visible','off');

    %A cell i.e. temp, which stores all information from panelSetup is
    %created and will be written into a .mat file
    temp=cell(1,18);
    temp{1,14}=noPanels;
    temp{1,15}=noGraphicPanels;
    temp{1,16}=noButtons;
    temp{1,18}=noTitles;

    %Draw a plot to select points.If a plot(i.e. the white screen)is
not
    %drawn over the figure window using the axes command, the ginput
    %command will not work
    plotHandle = axes('parent',f,...
    'Units','Normalized',...
    'Position',[0 0 1 1]);
%,...
 %   'GridLineStyle','--');
    grid on;
    hold on;
    %PANELS
    %You may now begin selecting the endpoints for each panel
```

```
    %Only three endpoints begining with the lower left point, the
    %lower right point and the upper right point needs to be selected
    x=zeros(1,3*noPanels);
    y=zeros(1,3*noPanels);

    if noPanels~=0
        %Msgbox indicating that the co-ordinates for the panels will
now begin
        %to be selected
        panelString=strcat('You may now begin selecting the ',...
        ' co-ordinates of the panels. Begin by selecting the lower
',...
        ' left corner, then the lower right corner and finish by ',...
        ' selecting the top right corner for each panel.');
        msgbox(panelString,'modal')
        uiwait(gcf);%This stops execution until the user has clicked
ok

        for i=0:noPanels-1
            for j=1:3
                [x(j+4*i) y(j+4*i)]=ginput(1);
                plot(x(j+4*i),y(j+4*i),'marker','*');
                hold on;
                %display('hellooo');
                xlim([0 1]);ylim([0 1]);
            end

            line([x(1+4*i) x(2+4*i) x(3+4*i) x(1+4*i) x(1+4*i)],...
            [y(1+4*i) y(2+4*i) y(3+4*i) y(3+4*i) y(1+4*i)],...
            'Color', 'k','linewidth', 2);
            if i==noPanels-1
                temp{1,1}=x;%x and y are for the button panels
                temp{1,2}=y;
            end
       end
    end
```

```
    %GRAPHICS PANELS
    %You may now begin selecting the endpoints for each graphic panel
    %Only three endpoints begining with the lower left point, the
    %lower right point and the upper right point needs to be selected
    a=zeros(1,3*noGraphicPanels);
    b=zeros(1,3*noGraphicPanels);
    if noGraphicPanels~=0
        %Msgbox indicating that the panels have been drawn and that
the
        %graphic panels may now be drawn
        graphicString=strcat('You may now begin selecting the ',...
            ' co-ordinates of the graphics panels. Begin by ',...
            ' selecting the lower left corner, then the lower ',...
            ' right corner and finish by selecting the top right ',...
            ' corner for each graphic panel.');
        msgbox(graphicString,'modal')
        uiwait(gcf);%This stops execution until the user has clicked
ok

        for i=0:noGraphicPanels-1
            for j=1:3
                [a(j+4*i) b(j+4*i)]=ginput(1);
                plot(a(j+4*i),b(j+4*i),'marker','*');
                hold on;
                %display('hellooo');
                xlim([0 1]);ylim([0 1]);

            end
            line([a(1+4*i) a(2+4*i) a(3+4*i) a(1+4*i) a(1+4*i)],...
            [b(1+4*i) b(2+4*i) b(3+4*i) b(3+4*i) b(1+4*i)],...
            'Color', 'g','linewidth', 2);
            if i==noGraphicPanels-1
                temp{1,3}=a;%a and b are coordiantes for the
                temp{1,4}=b;%graphics panel
            end
        end
    end
```

```
    %TITLE BOXES
    %You may now begin selecting the endpoints for each title box
    %Only three endpoints begining with the lower left point, the
    %lower right point and the upper right point needs to be selected
    u=zeros(1,3*noTitles);
    v=zeros(1,3*noTitles);
    if noTitles~=0
        %Msgbox indicating that the graphic panels have been drawn and
        %that the title boxes may now be drawn
        titleString=strcat('You may now begin selecting the ',...
            '  co-ordinates  of  the  title  boxes.  Begin  by  selecting
',...
            ' the lower left corner, then the lower right corner ',...
            ' and finish by selecting the top right corner for ',...
            ' each title box.');
        msgbox(titleString,'modal')
        uiwait(gcf);%This  stops  execution  until  the  user  has  clicked
ok

        for i=0:noTitles-1
            for j=1:3
                [u(j+4*i) v(j+4*i)]=ginput(1);
                plot(u(j+4*i),v(j+4*i),'marker','*');
                hold on;
                %display('hellooo');
                xlim([0 1]);ylim([0 1]);
            end
            line([u(1+4*i) u(2+4*i) u(3+4*i) u(1+4*i) u(1+4*i)],...
            [v(1+4*i) v(2+4*i) v(3+4*i) v(3+4*i) v(1+4*i)],...
            'Color', 'm','linewidth', 2);
            if i==noTitles-1
                temp{1,5}=u;%u and v are for the title boxes
                temp{1,6}=v;
            end
        end
    end
    %BUTTONS
```

```
    %You may now begin selecting the endpoints for each button
    %Only the lower left point needs to be selected
    m=zeros(1,noButtons);
    n=zeros(1,noButtons);

    %Length and width of a button
    %initialized to 0.08 and 0.05 respectively
    lButton(1:noButtons)=0.08;
    wButton(1:noButtons)=0.05;

    if noButtons~=0
        %Msgbox indicating that the title boxes have been drawn and
        %that the buttons may now be drawn
        buttonString=strcat('You may now begin selecting the ',...
            ' co-ordinates of the buttons. Select only the lower ',...
            ' left corner for each button.');
        msgbox(buttonString,'modal')
        uiwait(gcf);%This stops execution until the user has clicked
ok

        %Initializing for default values
        for i=1:noButtons
            ENTERSTYLEOFBUTTON{1,i}='pushbutton';
            ENTERSTRINGOFBUTTON{1,i}='Play';
            ENTERLABELOFBUTTON{1,i}='';
        end

        for i=1:noButtons
            [m(i) n(i)]=ginput(1);
            plot(m(i),n(i),'marker','*');
            hold on;
            %display('hellooo');
%             xlim([0 1]);ylim([0 1]);
%             line([m(i) m(i)+lButton m(i)+lButton m(i) m(i)],...
%                 [n(i) n(i) n(i)+wButton n(i)+wButton n(i)],...
%                 'Color', 'k');
```

```
            %if u have not yet selected the position of the
button,pause until
            %you have. If a pause is not created, the button stats gui
will
            %appear before the position of the point is selected.
            while m(i)==0
                pause(0.01);
            end
            %move1=-0.05;
            %Button Stats figure to enter the button details
            buttonStats = figure('Visible','on',...
            'Units','normalized',...
            'Position',[0.2 0.2 0.6 0.6],...
            'MenuBar','none',...
            'NumberTitle','off');
            set(buttonStats,'Name','Button stats');

            buttonStatsTitleDisplay=uicontrol('Parent',buttonStats,...
            'Units','Normalized',...
            'Style','text',...
            'Position',[0.35 0.9 0.2 0.075],...
            'FontSize',13,...
            'String','Button Stats GUI');

            %Buttons for Button stats
            ENTERSTYLEOFBUTTONBox=uicontrol('Parent',buttonStats,...
            'Units','Normalized',...
            'Style','popupmenu',...
            'Position',[0.2 0.84-move 0.2 0.1],...
            'Backgroundcolor','w',...
            'String','pushbutton|edit|text|popupmenu|slider',...
            'Callback',{@ENTERSTYLEOFBUTTONCallback,i});
            Label8=uicontrol('parent',buttonStats,...
                'Units','normalized',...
                'Position',[0.2 0.81-move 0.2 0.06],...
                'Style','text',...
                'String','Choose the type of the button');
```

```
            ENTERSTRINGOFBUTTONBox=uicontrol('Parent',buttonStats,...
            'Units','Normalized',...
            'Style','edit',...
            'Position',[0.5 0.88-move 0.2 0.1],...
            'Backgroundcolor','w',...
            'String','Play',...
            'Callback',{@ENTERSTRINGOFBUTTONCallback,i});
            Label9=uicontrol('parent',buttonStats,...
                'Units','normalized',...
                'Position',[0.5 0.81-move 0.2 0.06],...
                'Style','text',...
                'String','Enter the string to be written on the
button.**');


            ENTERLABELOFBUTTONBox=uicontrol('Parent',buttonStats,...
            'Units','Normalized',...
            'Style','edit',...
            'Position',[0.3 0.67-move 0.3 0.1],...
            'Backgroundcolor','w',...
            'String','...',...
            'Callback',{@ENTERLABELOFBUTTONCallback,i});

        labelString=strcat('Enter a label for the button ',...
                ' created in the box below. After you have ',...
                ' entered a label click on the Accept Specs ',...
                ' button. If you do not require a label, ',...
                ' click the Accept Specs Button. ');

            labelInfoBox=uicontrol('Parent',buttonStats,...
            'Units','Normalized',...
            'Style','text',...
            'Position',[0.3 0.52-move 0.3 0.13],...
            'String',labelString);

         %ENTER NEW LENGTH OF BUTTON
```

```
        ENTERLENGTHOFBUTTONBox=uicontrol('Parent',buttonStats,...
        'Units','Normalized',...
        'Style','edit',...
        'Position',[0.5 0.4-move 0.2 0.08],...
        'Backgroundcolor','w',...
        'String','0.08',...
        'Callback',{@ENTERLENGTHOFBUTTONCallback,i});
        lengthLabel=uicontrol('Parent',buttonStats,...
            'Units','Normalized',...
            'Style','text',...
            'Position',[0.5 0.33-move 0.2 0.06],...
            'String','Enter the length of the button');

        %ENTER NEW WIDTH OF BUTTON
        ENTERWIDTHOFBUTTONBox=uicontrol('Parent',buttonStats,...
        'Units','Normalized',...
        'Style','edit',...
        'Position',[0.2 0.4-move 0.2 0.08],...
        'Backgroundcolor','w',...
        'String','0.05',...
        'Callback',{@ENTERWIDTHOFBUTTONCallback,i});
        lengthLabel=uicontrol('Parent',buttonStats,...
            'Units','Normalized',...
            'Style','text',...
            'Position',[0.2 0.33-move 0.2 0.06],...
            'String','Enter the width of the button');

            acceptSpecsBox=uicontrol('Parent',buttonStats,...
            'Units','Normalized',...
            'Position',[0.3 0.2-move 0.3 0.1],...
            'String','Accept Specs Button',...
            'Callback',{@acceptSpecsCallback,i,m,n});

            labelNote=strcat('**NOTE:The  popupmenu  button  requires
that the ',...
                ' string be entered as shown in the example. ',...
```

```
                '    eg:Hamming|Triangular|Rectangular.    Also,    the
"slider" ',...
                'button does not require a string. ');

            labelNoteBox=uicontrol('Parent',buttonStats,...
            'Units','Normalized',...
            'Style','text',...
            'Position',[0.1 0.1-move 0.8 0.065],...
            'String',labelNote);

            %While the FLAGFORPAUSE is not set,pause. Then reset the
FLAGFORPAUSE
            while FLAGFORPAUSE==0
                pause(0.01);
            end
            FLAGFORPAUSE=0;
            pause(0.05);

            xlim([0 1]);ylim([0 1]);
            line([m(i) m(i)+lButton(i) m(i)+lButton(i) m(i) m(i)],...
                [n(i) n(i) n(i)+wButton(i) n(i)+wButton(i) n(i)],...
                'Color', 'k', 'linewidth', 2);

            %Load all button parameters into the cell once all the
buttons are
            %created
            if i==noButtons
                temp{1,7}=m;%m and n are for the buttons
                temp{1,8}=n;
                temp{1,9}=lButton;
                temp{1,10}=wButton;
                temp{1,11}=ENTERSTYLEOFBUTTON;
                temp{1,12}=ENTERSTRINGOFBUTTON;
                temp{1,13}=ENTERLABELOFBUTTON;
                temp{1,17}=labelDist;
            end
        end
```

```
  end
%UICONTROL objects associated with saving the current coordinates
SAVETEXTBOX=uicontrol('Parent',f,...
  'Units','Normalized',...
  'Style','text',...
  'Position',[0.36 0.9 0.15 0.05],...
  'Backgroundcolor','w',...
  'String','Would you like to save your selection',...
  'Callback',@saveCallback);
YESBUTTON=uicontrol('Style','RadioButton',...
    'Units','Normalized',...
    'String','Yes',...
    'Position',[0.36 0.85 0.08 0.05],...
    'Parent',f,...
    'Callback',{@yesCallback,temp});
NOBUTTON=uicontrol('Style','RadioButton',...
     'Units','Normalized',...
    'String','No',...
    'Position',[0.36 0.8 0.08 0.05],...
    'Parent',f,...
    'Callback',@noCallback);


%CALLBACKS within the beginDrawing function
 function ENTERSTYLEOFBUTTONCallback(src,eventdata,i)
      typeIndex=get(ENTERSTYLEOFBUTTONBox,'val');
      if typeIndex==1
          ENTERSTYLEOFBUTTON{1,i}='pushbutton';
      elseif typeIndex==2
           ENTERSTYLEOFBUTTON{1,i}='edit';
      elseif typeIndex==3
           ENTERSTYLEOFBUTTON{1,i}='text';
      elseif typeIndex==4
           ENTERSTYLEOFBUTTON{1,i}='popupmenu';
      elseif typeIndex==5
          ENTERSTYLEOFBUTTON{1,i}='slider';
      end
  end
  function ENTERSTRINGOFBUTTONCallback(src,eventdata,i)
```

```
        ENTERSTRINGOFBUTTON{1,i}=get(ENTERSTRINGOFBUTTONBox,'string');
    end
    function ENTERLABELOFBUTTONCallback(src,eventdata,i)
        ENTERLABELOFBUTTON{1,i}=get(ENTERLABELOFBUTTONBox,'string');
    end
    function ENTERLENGTHOFBUTTONCallback(src,eventdata,i)
        lButton(1,i)=str2num(get(ENTERLENGTHOFBUTTONBox,'string'));
    end
    function ENTERWIDTHOFBUTTONCallback(src,eventdata,i)
        wButton(1,i)=str2num(get(ENTERWIDTHOFBUTTONBox,'string'));
    end
    function acceptSpecsCallback(src,eventdata,i,m,n)
        %If a label is created, then draw the label.
        if strcmp(ENTERLABELOFBUTTON{1,i},'')==0
            figure(f),line([m(i)  m(i)+lButton(i)  m(i)+lButton(i)  m(i)
m(i)],...
            [n(i)-labelDist n(i)-labelDist...
            n(i)-labelDist+labelHeight ...
            n(i)-labelDist+labelHeight n(i)-labelDist],...
            'Color', 'r');
        end
        close(buttonStats);
        FLAGFORPAUSE=1;   %Set FLAGFORPAUSE to release the pause
    end


 end


 function noPanelsCallback(varargin)
    noPanels = str2num(get(numberOfPanels,'string'));
 end
function noGraphicPanelsCallback(varargin)
    noGraphicPanels = str2num(get(numberOfGraphicPanels,'string'));
end
function noTitlesCallback(varargin)
    noTitles = str2num(get(numberOfTitles,'string'));
 end
function noButtonsCallback(varargin)
    noButtons = str2num(get(numberOfButtons,'string'));
```

```
 end
 function lButtonCallback(varargin)
    lButton = str2num(get(lengthButton,'string'));
 end
 function wButtonCallback(varargin)
    wButton = str2num(get(widthButton,'string'));
 end

%callback for filename speech
function fileNameCallback(h,eventdata)
    writefilename=get(fileName,'string');
end



function yesCallback(h, eventdata,temp)
    if get(YESBUTTON,'Value')==1
       %turn off the no radio button
       set(NOBUTTON,'Value',0);

       %Save button
       SAVEBUTTON=uicontrol('Parent',f,...
            'Units','normalized',...
            'Position',[0.36 0.7 0.08 0.05],...
            'String','Save coordinates',...
            'Callback',{@saveCallback,temp});
    else
        display('You selected no');
    end

end

%If you donot want to save your selection, turn the radio buttons and
%save off and call the beginDrawCallback again
function noCallback(h, eventdata)
    if get(NOBUTTON,'Value')==1
      set(SAVETEXTBOX,'Visible','off');
      set(YESBUTTON,'Visible','off');
```

```
      set(NOBUTTON,'Visible','off');
      set(SAVEBUTTON,'Visible','off');
      beginDrawCallback(h,eventdata);
    end

end

%Callback for save speech
%The entire cell, temp is saved into a .mat file
function saveCallback(src,eventdata,temp)
    currentDir=pwd;
    currDir=strcat(currentDir,'\',writefilename,'.mat');
    save(strvcat(currDir),'temp');
end

end
```

The code for the 'PanelandButtonCallbacks.m' file and the 'runGUI.m' file for each of the example programs is mentioned below.

1. Program 1 – Hello World Program

- The 'panelandButtonCallbacks.m' file.

```
function PanelandButtonCallbacks(f,C,labelHeight)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
x=C{1,1};
y=C{1,2};
a=C{1,3};
b=C{1,4};
u=C{1,5};
v=C{1,6};
m=C{1,7};
n=C{1,8};
lButton=C{1,9};
wButton=C{1,10};
enterType=C{1,11};
```

```
enterString=C{1,12};
enterLabel=C{1,13};
noPanels=C{1,14};
noGraphicPanels=C{1,15};
noButtons=C{1,16};
labelDist=C{1,17};%distance that the label is below the button
noTitles=C{1,18};
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%BUTTON PANELS
for j=0:noPanels-1
    uipanel('Parent',f,...
        'Units','Normalized',...
        'Position',[x(1+4*j)   y(1+4*j)   x(2+4*j)-x(1+4*j)   y(3+4*j)-
y(2+4*j)]);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%GRAPHIC PANELS
for i=0:noGraphicPanels-1
  switch (i+1)
    case 1
        graphicPanel1 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 2
        graphicPanel2 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 3
        graphicPanel3 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
```

```
            'GridLineStyle','--');
    case 4
        graphicPanel4 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i) b(1+4*i) a(2+4*i)-a(1+4*i) b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 5
        graphicPanel5 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i) b(1+4*i) a(2+4*i)-a(1+4*i) b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 6
        graphicPanel6 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i) b(1+4*i) a(2+4*i)-a(1+4*i) b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%TITLE BOXES
for k=0:noTitles-1
    %Temporary strings whose value can be changed to reflect the
correct title
string1='This is a title box. You may change this string';
string2='This is a title box. You may change this string';
string3='This is a title box. You may change this string';
string4='This is a title box. You may change this string';
string5='This is a title box. You may change this string';
string6='This is a title box. You may change this string';

switch (k+1)

    case 1
        titleBox1 = uicontrol('parent',f,...
```

```
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string1);
    case 2
        titleBox2 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string2);
    case 3
        titleBox3 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string3);
    case 4
        titleBox4 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string4);
    case 5
        titleBox5 = uicontrol('parent',f,...
            'Units','Normalized',...
```

```
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string5);
    case 6
        titleBox6 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string6);
end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%BUTTONS
for i=1:noButtons
        enterColor='w';
        if                          strcmp(enterType{i},'pushbutton')==1
||strcmp(enterType{i},'text')==1
            enterColor='default';
        end
        if strcmp(enterLabel{1,i},'')==0%i.e. there is a label
            %creating a label for some buttons
            uicontrol('Parent',f,...
            'Units','Normalized',...
            'Position',[m(i)         n(i)-labelDist         lButton(i)
labelHeight],...
            'Style','text',...
            'String',enterLabel{i},...
            'HorizontalAlignment','center');
        end
     switch i
```

```
case 1
button1=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button1Callback);
case 2
button2=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button2Callback);
case 3
button3=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button3Callback);
case 4
button4=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button4Callback);
case 5
button5=uicontrol('Parent',f,...
```

```
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button5Callback);
case 6
button6=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button6Callback);
case 7
button7=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button7Callback);
case 8
button8=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button8Callback);
case 9
button9=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
```

```
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button9Callback);
case 10
button10=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button10Callback);
case 11
button11=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button11Callback);
case 12
button12=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button12Callback);
case 13
button13=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
```

```
            'BackgroundColor',enterColor,...
            'HorizontalAlignment','center',...
            'Callback',@button13Callback);
        case 14
        button14=uicontrol('Parent',f,...
            'Units','Normalized',...
            'Position',[m(i) n(i) lButton(i) wButton(i)],...
            'Style',enterType{i},...
            'String',enterString{i},...
            'BackgroundColor',enterColor,...
            'HorizontalAlignment','center',...
            'Callback',@button14Callback);
        case 15
        button15=uicontrol('Parent',f,...
            'Units','Normalized',...
            'Position',[m(i) n(i) lButton(i) wButton(i)],...
            'Style',enterType{i},...
            'String',enterString{i},...
            'BackgroundColor',enterColor,...
            'HorizontalAlignment','center',...
            'Callback',@button15Callback);
     end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
    %USER CODE FOR THE VARIABLES AND CALLBACKS
    %INITIALIZATION

    %set a name for the GUI
    set(f,'Name','Solution 2-Hello World GUI');

    function button1Callback(src,eventdata)
        msgbox('Hello World!','modal');
    end

end
```

- The ‘runGUI.m’ file.

```
function runGUI
    clc;
    clear all;

    %ENTER THE NAME OF THE FILE
    fileData=load('helloworld.mat');
    temp=fileData(1).temp;

    labelHeight=0.8*0.05;

    %x y are related to the panels
    %m n are related to the buttons

    f = figure('Visible','on',...
        'Units','normalized',...
        'Position',[0,0,1,1],...
        'MenuBar','none',...
        'NumberTitle','off');

    PanelandButtonCallbacks(f,temp,labelHeight);

end
```

2. **Program 2 - Display the waveform of a designated speech file.**

- The ‘panelandButtonCallbacks.m’ file.

```
function PanelandButtonCallbacks(f,C,labelHeight)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
x=C{1,1};
y=C{1,2};
a=C{1,3};
b=C{1,4};
u=C{1,5};
v=C{1,6};
m=C{1,7};
n=C{1,8};
lButton=C{1,9};
```

```
wButton=C{1,10};
enterType=C{1,11};
enterString=C{1,12};
enterLabel=C{1,13};
noPanels=C{1,14};
noGraphicPanels=C{1,15};
noButtons=C{1,16};
labelDist=C{1,17};%distance that the label is below the button
noTitles=C{1,18};
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%BUTTON PANELS
for j=0:noPanels-1
    uipanel('Parent',f,...
        'Units','Normalized',...
        'Position',[x(1+4*j)   y(1+4*j)   x(2+4*j)-x(1+4*j)   y(3+4*j)-
y(2+4*j)]);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%GRAPHIC PANELS
for i=0:noGraphicPanels-1
  switch (i+1)
    case 1
        graphicPanel1 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 2
        graphicPanel2 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 3
        graphicPanel3 = axes('parent',f,...
            'Units','Normalized',...
```

```
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 4
        graphicPanel4 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 5
        graphicPanel5 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 6
        graphicPanel6 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%TITLE BOXES
for k=0:noTitles-1
    %Temporary strings whose value can be changed to reflect the
correct title
string1='This is a title box. You may change this string';
string2='This is a title box. You may change this string';
string3='This is a title box. You may change this string';
string4='This is a title box. You may change this string';
string5='This is a title box. You may change this string';
string6='This is a title box. You may change this string';

switch (k+1)
```

```
    case 1
        titleBox1 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string1);
    case 2
        titleBox2 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string2);
    case 3
        titleBox3 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string3);
    case 4
        titleBox4 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string4);
    case 5
        titleBox5 = uicontrol('parent',f,...
```

```
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string5);
    case 6
        titleBox6 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string6);
end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%BUTTONS
for i=1:noButtons
        enterColor='w';
        if                          strcmp(enterType{i},'pushbutton')==1
||strcmp(enterType{i},'text')==1
            enterColor='default';
        end
        if strcmp(enterLabel{1,i},'')==0%i.e. there is a label
            %creating a label for some buttons
            uicontrol('Parent',f,...
            'Units','Normalized',...
            'Position',[m(i)         n(i)-labelDist         lButton(i)
labelHeight],...
            'Style','text',...
            'String',enterLabel{i},...
            'HorizontalAlignment','center');
        end
```

```
switch i
   case 1
   button1=uicontrol('Parent',f,...
       'Units','Normalized',...
       'Position',[m(i) n(i) lButton(i) wButton(i)],...
       'Style',enterType{i},...
       'String',enterString{i},...
       'BackgroundColor',enterColor,...
       'HorizontalAlignment','center',...
       'Callback',@button1Callback);
   case 2
   button2=uicontrol('Parent',f,...
       'Units','Normalized',...
       'Position',[m(i) n(i) lButton(i) wButton(i)],...
       'Style',enterType{i},...
       'String',enterString{i},...
       'BackgroundColor',enterColor,...
       'HorizontalAlignment','center',...
       'Callback',@button2Callback);
   case 3
   button3=uicontrol('Parent',f,...
       'Units','Normalized',...
       'Position',[m(i) n(i) lButton(i) wButton(i)],...
       'Style',enterType{i},...
       'String',enterString{i},...
       'BackgroundColor',enterColor,...
       'HorizontalAlignment','center',...
       'Callback',@button3Callback);
   case 4
   button4=uicontrol('Parent',f,...
       'Units','Normalized',...
       'Position',[m(i) n(i) lButton(i) wButton(i)],...
       'Style',enterType{i},...
       'String',enterString{i},...
       'BackgroundColor',enterColor,...
       'HorizontalAlignment','center',...
       'Callback',@button4Callback);
   case 5
```

```
button5=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button5Callback);
case 6
button6=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button6Callback);
case 7
button7=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button7Callback);
case 8
button8=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button8Callback);
case 9
button9=uicontrol('Parent',f,...
    'Units','Normalized',...
```

```
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button9Callback);
case 10
button10=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button10Callback);
case 11
button11=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button11Callback);
case 12
button12=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button12Callback);
case 13
button13=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
```

```
            'String',enterString{i},...
            'BackgroundColor',enterColor,...
            'HorizontalAlignment','center',...
            'Callback',@button13Callback);
        case 14
        button14=uicontrol('Parent',f,...
            'Units','Normalized',...
            'Position',[m(i) n(i) lButton(i) wButton(i)],...
            'Style',enterType{i},...
            'String',enterString{i},...
            'BackgroundColor',enterColor,...
            'HorizontalAlignment','center',...
            'Callback',@button14Callback);
        case 15
        button15=uicontrol('Parent',f,...
            'Units','Normalized',...
            'Position',[m(i) n(i) lButton(i) wButton(i)],...
            'Style',enterType{i},...
            'String',enterString{i},...
            'BackgroundColor',enterColor,...
            'HorizontalAlignment','center',...
            'Callback',@button15Callback);
     end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
    %USER CODE FOR THE VARIABLES AND CALLBACKS
    %INITIALIZATION

    %set the name of the GUI
    set(f,'Name','Display Speech Waveform GUI');

    %button 1-Display speech callback
    function button1Callback(src,eventdata)
        loadedSpeech=wavread('s1.wav');
            %The speech file is 's1.wav'
            axes(graphicPanel1);
            plot(loadedSpeech);
```

```
            title('s1.wav');
            xlabel('Time in seconds');
            ylabel('Amplitude');

    end

    %button2-Close callback
    function button2Callback(src,eventdata)
       close(gcf);
    end

end
```

- The ‘runGUI.m’ file.

```
function runGUI
    clc;
    clear all;

    %ENTER THE NAME OF THE FILE
    fileData=load('displayspeech.mat');
    temp=fileData(1).temp;

    labelHeight=0.8*0.05;

    %x y are related to the panels
    %m n are related to the buttons

    f = figure('Visible','on',...
        'Units','normalized',...
        'Position',[0,0,1,1],...
        'MenuBar','none',...
        'NumberTitle','off');

    PanelandButtonCallbacks(f,temp,labelHeight);
```

```
end
```

3. **Program 3 - Load a Speech File, Play it back and Display the Waveform.**

- The 'panelandButtonCallbacks.m' file.

```
function PanelandButtonCallbacks(f,C,labelHeight)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
x=C{1,1};
y=C{1,2};
a=C{1,3};
b=C{1,4};
u=C{1,5};
v=C{1,6};
m=C{1,7};
n=C{1,8};
lButton=C{1,9};
wButton=C{1,10};
enterType=C{1,11};
enterString=C{1,12};
enterLabel=C{1,13};
noPanels=C{1,14};
noGraphicPanels=C{1,15};
noButtons=C{1,16};
labelDist=C{1,17};%distance that the label is below the button
noTitles=C{1,18};
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%BUTTON PANELS
for j=0:noPanels-1
    uipanel('Parent',f,...
        'Units','Normalized',...
        'Position',[x(1+4*j)   y(1+4*j)   x(2+4*j)-x(1+4*j)   y(3+4*j)-
y(2+4*j)]);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%GRAPHIC PANELS
```

```
for i=0:noGraphicPanels-1
  switch (i+1)
    case 1
        graphicPanel1 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 2
        graphicPanel2 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 3
        graphicPanel3 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 4
        graphicPanel4 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 5
        graphicPanel5 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 6
        graphicPanel6 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
```

```
  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%TITLE BOXES
for k=0:noTitles-1
    %Temporary strings whose value can be changed to reflect the
correct title
string1='This is a title box. You may change this string';
string2='This is a title box. You may change this string';
string3='This is a title box. You may change this string';
string4='This is a title box. You may change this string';
string5='This is a title box. You may change this string';
string6='This is a title box. You may change this string';

switch (k+1)

    case 1
        titleBox1 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string1);
    case 2
        titleBox2 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string2);
    case 3
        titleBox3 = uicontrol('parent',f,...
            'Units','Normalized',...
```

```
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string3);
    case 4
        titleBox4 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string4);
    case 5
        titleBox5 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string5);
    case 6
        titleBox6 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string6);
end

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
```

```
%BUTTONS
for i=1:noButtons
        enterColor='w';
        if                          strcmp(enterType{i},'pushbutton')==1
||strcmp(enterType{i},'text')==1
            enterColor='default';
        end
        if strcmp(enterLabel{1,i},'')==0%i.e. there is a label
            %creating a label for some buttons
            uicontrol('Parent',f,...
            'Units','Normalized',...
            'Position',[m(i)          n(i)-labelDist          lButton(i)
labelHeight],...
            'Style','text',...
            'String',enterLabel{i},...
            'HorizontalAlignment','center');
        end
     switch i
        case 1
        button1=uicontrol('Parent',f,...
            'Units','Normalized',...
            'Position',[m(i) n(i) lButton(i) wButton(i)],...
            'Style',enterType{i},...
            'String',enterString{i},...
            'BackgroundColor',enterColor,...
            'HorizontalAlignment','center',...
            'Callback',@button1Callback);
        case 2
        button2=uicontrol('Parent',f,...
            'Units','Normalized',...
            'Position',[m(i) n(i) lButton(i) wButton(i)],...
            'Style',enterType{i},...
            'String',enterString{i},...
            'BackgroundColor',enterColor,...
            'HorizontalAlignment','center',...
            'Callback',@button2Callback);
        case 3
        button3=uicontrol('Parent',f,...
```

```
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button3Callback);
case 4
button4=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button4Callback);
case 5
button5=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button5Callback);
case 6
button6=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button6Callback);
case 7
button7=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
```

```
        'Style',enterType{i},...
        'String',enterString{i},...
        'BackgroundColor',enterColor,...
        'HorizontalAlignment','center',...
        'Callback',@button7Callback);
    case 8
    button8=uicontrol('Parent',f,...
        'Units','Normalized',...
        'Position',[m(i) n(i) lButton(i) wButton(i)],...
        'Style',enterType{i},...
        'String',enterString{i},...
        'BackgroundColor',enterColor,...
        'HorizontalAlignment','center',...
        'Callback',@button8Callback);
    case 9
    button9=uicontrol('Parent',f,...
        'Units','Normalized',...
        'Position',[m(i) n(i) lButton(i) wButton(i)],...
        'Style',enterType{i},...
        'String',enterString{i},...
        'BackgroundColor',enterColor,...
        'HorizontalAlignment','center',...
        'Callback',@button9Callback);
    case 10
    button10=uicontrol('Parent',f,...
        'Units','Normalized',...
        'Position',[m(i) n(i) lButton(i) wButton(i)],...
        'Style',enterType{i},...
        'String',enterString{i},...
        'BackgroundColor',enterColor,...
        'HorizontalAlignment','center',...
        'Callback',@button10Callback);
    case 11
    button11=uicontrol('Parent',f,...
        'Units','Normalized',...
        'Position',[m(i) n(i) lButton(i) wButton(i)],...
        'Style',enterType{i},...
        'String',enterString{i},...
```

```
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button11Callback);
case 12
button12=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button12Callback);
case 13
button13=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button13Callback);
case 14
button14=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button14Callback);
case 15
button15=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
```

```
            'Callback',@button15Callback);
     end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
    %USER CODE FOR THE VARIABLES AND CALLBACKS
    %INITIALIZATION
    curr_file=1;
    fs=1;
    directory_name='abcd';
    wav_file_names='abcd';
%set(titleBox1,'Position', [u(1) v(1)+0.01 u(2)-u(1) v(3)-v(2)+0.01]);



%    set the name of the GUI
    set(f,'Name','Play and Plot Speech GUI');

    function button1Callback(src,eventdata)

        directory_name = uigetdir('start_path','dialog_title');
        A=strvcat(strcat((directory_name),'\*.wav'));
        struct_filenames=dir(A);
        wav_file_names={struct_filenames.name};
        set(button2,'String',wav_file_names);
        %once the popupmenu/drop down menu is created, by default, the
first
        %selection from the popupmenu/drop down menu must be loaded
even if the
        %callback for the popupmenu/drop down menu id not called

        indexOfDrpDwnMenu=1;%by default first option from the
popupmenu/dropdown
                        %menu will be loaded
        [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu);

    end
```

```
    function button2Callback(src,eventdata)
        indexOfDrpDwnMenu=get(button2,'val');
        [curr_file,fs]=loadSelection(directory_name,...
            wav_file_names,indexOfDrpDwnMenu);
    end
    %Function--load selection
    function [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu)
        fin_path=strcat(directory_name,'\',...
            strvcat(wav_file_names(indexOfDrpDwnMenu)));
            %fin_path is the complete path of the file .wav file that
is
            %selected
        clear curr_file;
        clear fs;
        [curr_file, fs]=wavread(fin_path);
        FS=num2str(fs);
        %Information about the file being played
        file_info_string=strcat('Current file = ',...
            wav_file_names(indexOfDrpDwnMenu),...
            '. Sampling frequency = ',FS,'Hz',...
            '. Number of samples in file = ',...
            num2str(length(curr_file)));
        set(titleBox1,'String',file_info_string);
        set(titleBox1,'FontSize',0.3);
    end
    %Callback for the playbutton
    function  button3Callback(h,eventdata)
        sound(curr_file,fs);
    end
    %callback for the plotbutton
     function button4Callback(h,eventdata)
        hold off;  %It is essential to turn hold off so that the
        hold off;  %earlier contents of the panel are replaced
        %the two hold off's are for the speech file and the hamming
window

        grid off;
```

```
        reset(graphicPanel1);
        axes(graphicPanel1);
        l=length(curr_file);
        i=(1:l)/fs;%coverting samples to time
        plot(i,curr_file,'k','LineWidth',2),...
            xlabel('Time in seconds'),...
            ylabel('Amplitude');
        axis tight;
        grid on;
     end
%callback for the close GUI button
     function button5Callback(h,eventdata)
       close(gcf);
    end
end
```

- The 'runGUI.m' file.

```
function runGUI
    clc;
    clear all;

    %ENTER THE NAME OF THE FILE
    fileData=load('play&plot.mat');
    temp=fileData(1).temp;

    %wButton=temp{1,10};
    labelHeight=0.8*0.05;

    %x y are related to the panels
    %m n are related to the buttons
    %panel1pos=[x(1) y(1) x(2)-x(1) y(4)-y(2)];
    f = figure('Visible','on',...
        'Units','normalized',...
        'Position',[0,0,1,1],...
        'MenuBar','none',...
        'NumberTitle','off');
```

```
    PanelandButtonCallbacks(f,temp,labelHeight);

end
```

4. Program 4 - Load an existing speech file or record a new speech file. Play the file, display a waveform and save the file.

The 'panelandButtonCallbacks.m' file.

```
function PanelandButtonCallbacks(f,C,labelHeight)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
x=C{1,1};
y=C{1,2};
a=C{1,3};
b=C{1,4};
u=C{1,5};
v=C{1,6};
m=C{1,7};
n=C{1,8};
lButton=C{1,9};
wButton=C{1,10};
enterType=C{1,11};
enterString=C{1,12};
enterLabel=C{1,13};
noPanels=C{1,14};
noGraphicPanels=C{1,15};
noButtons=C{1,16};
labelDist=C{1,17};%distance that the label is below the button
noTitles=C{1,18};
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%BUTTON PANELS
for j=0:noPanels-1
    uipanel('Parent',f,...
        'Units','Normalized',...
        'Position',[x(1+4*j)   y(1+4*j)   x(2+4*j)-x(1+4*j)   y(3+4*j)-
y(2+4*j)]);
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%GRAPHIC PANELS
for i=0:noGraphicPanels-1
  switch (i+1)
    case 1
        graphicPanel1 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 2
        graphicPanel2 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 3
        graphicPanel3 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 4
        graphicPanel4 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 5
        graphicPanel5 = axes('parent',f,...
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
    case 6
        graphicPanel6 = axes('parent',f,...
```

```
            'Units','Normalized',...
            'Position',[a(1+4*i)  b(1+4*i)  a(2+4*i)-a(1+4*i)  b(3+4*i)-
b(2+4*i)],...
            'GridLineStyle','--');
  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%TITLE BOXES
for k=0:noTitles-1
    %Temporary  strings  whose  value  can  be  changed  to  reflect  the
correct title
string1='This is a title box. You may change this string';
string2='This is a title box. You may change this string';
string3='This is a title box. You may change this string';
string4='This is a title box. You may change this string';
string5='This is a title box. You may change this string';
string6='This is a title box. You may change this string';

switch (k+1)

    case 1
        titleBox1 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string1);
    case 2
        titleBox2 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
```

```
            'String',string2);
    case 3
        titleBox3 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string3);
    case 4
        titleBox4 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string4);
    case 5
        titleBox5 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string5);
    case 6
        titleBox6 = uicontrol('parent',f,...
            'Units','Normalized',...
            'Position',[u(1+4*k)  v(1+4*k)  u(2+4*k)-u(1+4*k)  v(3+4*k)-
v(2+4*k)],...
            'Style','text',...
            'FontUnits','Normalized',...
            'FontSize',0.5,...
            'String',string6);
end
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%BUTTONS
for i=1:noButtons
        enterColor='w';
        if                          strcmp(enterType{i},'pushbutton')==1
||strcmp(enterType{i},'text')==1
            enterColor='default';
        end
        if strcmp(enterLabel{1,i},'')==0%i.e. there is a label
            %creating a label for some buttons
            uicontrol('Parent',f,...
            'Units','Normalized',...
            'Position',[m(i)          n(i)-labelDist          lButton(i)
labelHeight],...
            'Style','text',...
            'String',enterLabel{i},...
            'HorizontalAlignment','center');
        end
     switch i
        case 1
        button1=uicontrol('Parent',f,...
            'Units','Normalized',...
            'Position',[m(i) n(i) lButton(i) wButton(i)],...
            'Style',enterType{i},...
            'String',enterString{i},...
            'BackgroundColor',enterColor,...
            'HorizontalAlignment','center',...
            'Callback',@button1Callback);
        case 2
        button2=uicontrol('Parent',f,...
            'Units','Normalized',...
            'Position',[m(i) n(i) lButton(i) wButton(i)],...
            'Style',enterType{i},...
            'String',enterString{i},...
            'BackgroundColor',enterColor,...
```

```
    'HorizontalAlignment','center',...
    'Callback',@button2Callback);
case 3
button3=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button3Callback);
case 4
button4=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button4Callback);
case 5
button5=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button5Callback);
case 6
button6=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button6Callback);
```

```
case 7
button7=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button7Callback);
case 8
button8=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button8Callback);
case 9
button9=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button9Callback);
case 10
button10=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button10Callback);
case 11
button11=uicontrol('Parent',f,...
```

```
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button11Callback);
case 12
button12=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button12Callback);
case 13
button13=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button13Callback);
case 14
button14=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
    'Style',enterType{i},...
    'String',enterString{i},...
    'BackgroundColor',enterColor,...
    'HorizontalAlignment','center',...
    'Callback',@button14Callback);
case 15
button15=uicontrol('Parent',f,...
    'Units','Normalized',...
    'Position',[m(i) n(i) lButton(i) wButton(i)],...
```

```
            'Style',enterType{i},...
            'String',enterString{i},...
            'BackgroundColor',enterColor,...
            'HorizontalAlignment','center',...
            'Callback',@button15Callback);
     end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
    %USER CODE FOR THE VARIABLES AND CALLBACKS
    %INITIALIZATION

    curr_file=1;
    directory_name='ABCD';
    wav_file_names='ABCD';
    y=1;%y is the variable that contains the recorded speech
    nsec=3;
    fs=8000;
    fileName='file1';

    %set a name for the GUI
    set(f,'Name','Solution 2-Record speech GUI');

    %button1-Get directory
    function button1Callback(src,eventdata)
        directory_name = uigetdir('start_path','dialog_title');
        A=strvcat(strcat((directory_name),'\*.wav'));
        struct_filenames=dir(A);
        wav_file_names={struct_filenames.name};
        set(button2,'String',wav_file_names);
        %once the popupmenu/drop down menu is created, by default, the
first
        %selection from the popupmenu/drop down menu must be loaded
even if the
        %callback for the popupmenu/drop down menu id not called
```

```
        indexOfDrpDwnMenu=1;%by   default   first   option   from   the
popupmenu/dropdown
                            %menu will be loaded
        [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu);
    end

    %button2-Select file
    function button2Callback(src,eventdata)
        indexOfDrpDwnMenu=get(button2,'val');
        [curr_file,fs]=loadSelection(directory_name,...
            wav_file_names,indexOfDrpDwnMenu);
    end

    %Function--load selection
    function [curr_file,fs]=loadSelection(directory_name,...
        wav_file_names,indexOfDrpDwnMenu)
        fin_path=strcat(directory_name,'\',...
            strvcat(wav_file_names(indexOfDrpDwnMenu)));
            %fin_path is the complete path of the file .wav file that
is
            %selected
        clear curr_file;
        clear fs;
        [curr_file, fs]=wavread(fin_path);
        FS=num2str(fs);
        %Information about the file being played
        file_info_string=strcat('Current file = ',...
            wav_file_names(indexOfDrpDwnMenu),...
            '. Sampling frequency = ',FS,'Hz',...
            '. Number of samples in file = ',...
            num2str(length(curr_file)));

    end

 %button3-play button
    function button3Callback(src,eventdata)
```

```
    sound(curr_file,fs);
end

%button4-enter samplign freq
function button4Callback(src,eventdata)
    fs=str2num(get(button4,'string'));
end

%button5-enter no of secs for recording button
function button5Callback(src,eventdata)
    nsec=str2num(get(button5,'string'));
end

%callback for the record/ re-record button
%record speech file of fixed duration (nsec) and
%given sampling rate(fs)

function button6Callback(h,eventdata)
   button4Callback(h,eventdata);
   button5Callback(h,eventdata);
% yn=speech samples normalized to 1
% N is the number of samples in each speech file
% ch is the number of channels in the recording
    N=fs*nsec;
    ch=1;
    y=wavrecord(N,fs,ch,'double');

    ymin=min(y);
    ymax=max(y);

    % calculate dc offset and correct
    offset=sum(y(N-999:N))/1000;
    y=y-offset;
    sound(y,fs);
end

%button7-get filename
```

```
    function button7Callback(src,eventdata)
        fileName=get(button7,'string');
    end

    %button8-save speech
    function button8Callback(src,eventdata)
        currentDir=pwd
        currDir=strcat(currentDir,'\',fileName,'.wav')
        wavwrite(y,fs,strvcat(currDir));
        c=wavread(strvcat(currDir));
        soundsc(c,fs)
    end



    %button9-close gui
    function button9Callback(src,eventdata)
        close(gcf);
    end

end
```

- The 'runGUI.m' file.

```
function runGUI
    clc;
    clear all;

    %ENTER THE NAME OF THE . mat FILE
    fileData=load('recordgui.mat');
    temp=fileData(1).temp;

    labelHeight=0.8*0.05;

    %x y are related to the panels
    %m n are related to the buttons
```

```
    f = figure('Visible','on',...
        'Units','normalized',...
        'Position',[0,0,1,1],...
        'MenuBar','none',...
        'NumberTitle','off');

    PanelandButtonCallbacks(f,temp,labelHeight);
end
```

References

[1] McGrenere, J., Baecker, R., Booth, K. (2002) An Evaluation of a Multiple Interface Design Solution for Bloated Software. In Proceedings of the CHI 2002 Conference on Human Factors in Computing Systems, April 20-25, 2002, Minneapolis, MN, pp. 163-170.

[2] Hsi, I. and Potts, C. (2000). Studying the evolution and enhancement of software features. International Conference on Software Maintenance, 143-151.

[3] Fischer, G. (1993). Shared knowledge in cooperative problem-solving systems - integrating adaptive and adaptable components. In M. Schneider-Hufschmidt, T. Kuhme and U. Malinowski (Eds.), Adaptive user interfaces: Principles and practice (pp. 49-68). North Holland: Elsevier Science Publishers B.V.

[4] Kaufman, L. and Weed, B. (1998). Too much of a good thing? Identifying and resolving bloat in the user interface: A CHI 98 workshop. SIGCHI Bulletin, 30(4), 46-47

[5] Mackay, W. E. (1991). Triggers and barriers to customizing software. CHI'91, 153 – 160.

[6] MathWorks. http://www.mathworks.com/help/techdoc/creating_guis/bqz79mu.html

[7] MathWorks. http://www.mathworks.com/help/techdoc/ref/uipanel.html

[8] MathWorks. http://www.mathworks.com/help/techdoc/ref/uicontrol_props.html

[9] L. R. Rabiner and R. W. Schafer. Theory and Applications of Digital Speech Processing. Prentice-Hall Inc., 2011.