Problem set 4: MATLAB Graphics User Interface

Handed out: Friday, November 11, 2016
Due: 23:55pm, Thursday, November 24, 2016

You must hand in two files:

- One file, named `your_name_E4.pdf`, with the solution to the exercise in this set, following the template available in the virtual campus. You must explain how you solved it, the necessary MATLAB code, the results obtained (run transcripts or plots), and some final comments (if any). Please, include the **version of MATLAB** you are working with.

- The second file, named `your_name_E4.zip`, must contain all the MATLAB code you used to solve the exercise, organized in one or more text (.m) files and (.fig) files to allow the teacher to check your solution.

Please, make sure that the code in the pdf file is exactly the same as in the zip file. Remember that all the MATLAB code is supposed to be entirely yours. Otherwise, you must clearly specify which parts are not, and properly attribute them to the source (names of collaborators, links to webpages, book references, etc.). Read the Course Information document for more details on this subject

Exercise 1. DTMF Signal Generator and GUI.

The goal of the exercise is to implement a phone dialer based on Dual-Tone Multi-Frequency signaling (DTMF). The DTMF system uses a set of eight audio frequencies transmitted in pairs to represent 16 signals corresponding to the ten digits (`0`, `1`, `2`, ..., `9`), the letters `A` to `D`, and the symbols # and $\star$.[1]

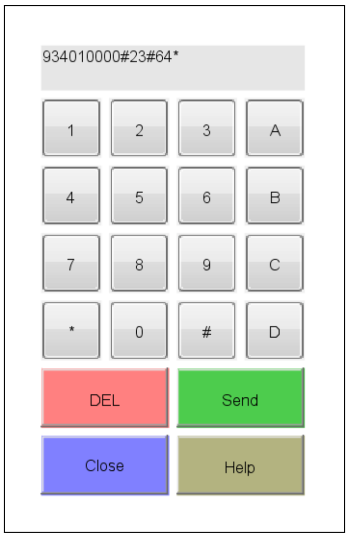
The following table shows the combination of frequencies corresponding to each key. Each row represents the low frequency component and each column represents the high frequency component of the DTMF signal. Pressing a key sends a combination of the row and column frequencies. For example, the key `1` produces a combination of tones of 697 Hz and 1209 Hz.

DTMF keypad frequencies

	1209 Hz	**1336 Hz**	**1477 Hz**	**1633 Hz**
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	$\star$	0	#	D

GUI description. Your work is to create a GUI containing a DTMF telephone keypad (a 4×4 matrix of push buttons, each one corresponding to a key). On top of the keypad there must be a display (a "text"-style **`UIControl`**) showing the sequence of keys entered by the user. As you can see in the figure below, four additional push buttons are also provided in the keypad. Namely,

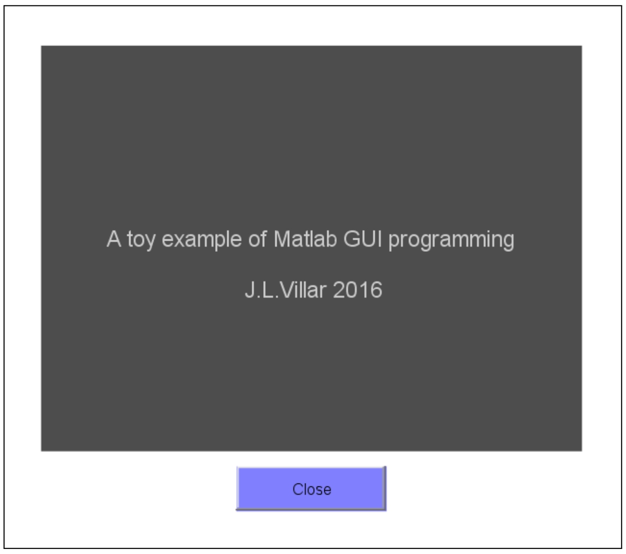
[1]See https://en.wikipedia.org/wiki/Dual-tone_multi-frequency_signaling.



- a `DEL` button that deletes the last key in the sequence,
- a `Close` button that exits the GUI (then closing all the active windows),
- a `Help` button that opens a window containing some help information,
- and a `Send` button that sends the entered sequence of keys.

The GUI must have three different windows (figures):

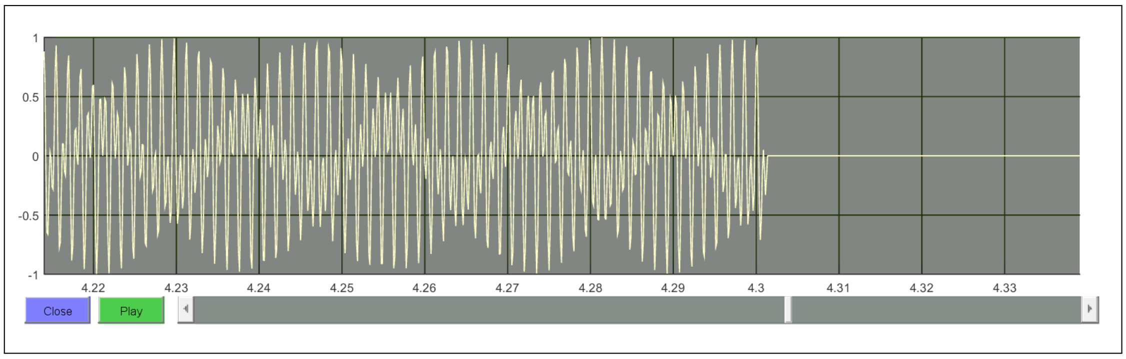
- The keypad window described above.
- The help window activated by the `Help` button in the keypad. This window contains a "text"-style **`UIControl`** showing the help text and a `Close` button that closes the help window, as shown below.



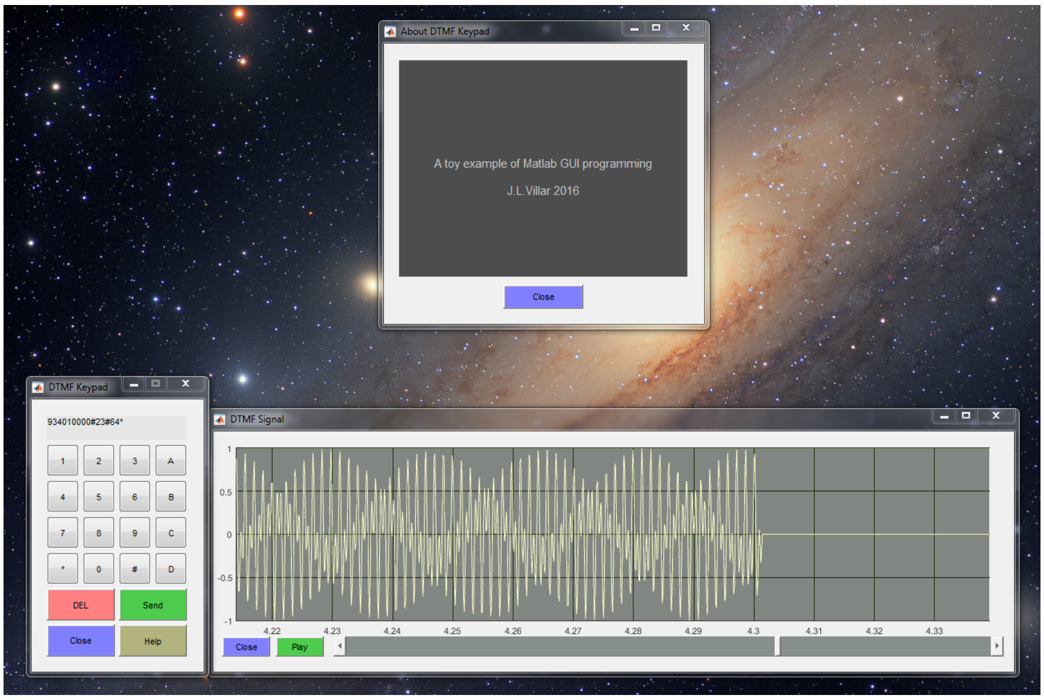
A toy example of Matlab GUI programming

J.L.Villar 2016

- An additional window, which we call the display window, that shows the signal generated from the sequence of keys. This window contains two push buttons: a `Close` button and a `Play` button (see figure below). In addition, a portion of the signal is plotted (using **`plot`** or **`line`**), which can be selected by using a sliding bar (a "slider"-style **`UIControl`**) located below the signal plot.



You can make all the windows not resizable (property **`'Resize'`** of the figure object), which saves you the effort of redistributing the controls when a window changes its size. You can also fix the position of the windows in the screen. See below as an example a screen capture. You can also customize the sizes, colors and distribution of the controls in the figures as you want.



GUI functionality. In the following, we describe the required functionality of the whole GUI. Try to implement as much of the features described below as possible. You are not required to achieve the whole functionality, but a reasonable and coherent subset of it. In particular, you could remove the sliding bar feature and then plot the entire signal. However, the more complete your implementation is, the higher mark you will get in the exercise.

Keypad window

- Key buttons (4×4 array of buttons): When pushing any of the buttons the corresponding character is appended to the sequence and the text display is updated accordingly. The sequence must have a limited length (e.g. 20 characters). Once the limit is reached, the keys must be disabled (using the **`'Enable'`** property and the **`set`** function). Make sure that the maximum length is never exceeded.
- `DEL` button: If this button is pushed then the last character of the sequence (if any) is deleted. This button must be disabled whenever the character sequence is empty (e.g.

when the program starts). Since this button reduces the length of the sequence, the state of the 16 key buttons must be updated (e.g. after pushing the `DEL` button they must be always enabled, because the length of the sequence does not reach its limit).

- `Help` button: Once pushed, the help window is created (if it does not exist). The help window must be unique, so check that never occurs that two different help windows are present.
- `Send` button: It creates the display window (if it does not exist) or updates it (if it already existed). This button must be disabled when the sequence of keys is empty (exactly as the `DEL` button). Once the display window is created, the key sequence is deleted (and then both `DEL` and `Send` buttons are disabled again). The user is then free to introduce a new sequence, as if the program started again.
- `Close` button: It closes all the active windows. Namely, if the help window and/or the display window exist then they are also closed. Perhaps you would need to include the handles of these windows in the **`handles`** structure. Remember that any changes to the **`handles`** structure must be synchronized with **`guidata(hObject,handles)`** to make them visible to the other functions in the GUI.[2]

Observe that a window can be closed in many different ways (and not only with the push button). Ideally, closing the keypad window in any way must also close the other active windows. This can be done by means of the **`CloseFcn`** callback function.

Display window

- Every time the window is created or updated by pushing the `Send` button in the keypad window, the signal is computed in the following way. For each key in the sequence entered by the user, generate and add the two tones using the frequencies given in the DTMF table. Use for instance a duration of 300 ms per key, and insert a silence (zeroes) between every two keys of 100 ms. Use a sampling rate of 8000 Hz. Store the entire signal into a vector **`X`**. This vector should be added to the **`handles`** structure in order to make it available in further function calls.
- The previously computed signal **`X`** is displayed using **`plot`** or **`line`** in the appropriate axis (you must need to customize the axis object using **`set`** and some properties like **`'Position'`**, **`'Units'`**, **`'Fontsize'`**, etc.). Use the **`axis`** command to select only a portion of the signal. When creating the window, display only the first 1000 samples of the signal (corresponding to a window of 125 ms at the given sampling rate). Initialize the position of the sliding bar cursor (the "slider"-style **`UIControl`**) to the leftmost one, using the **`'Value'`** property and the **`set`** function.
- Every time the sliding bar is moved, the plotted portion of signal must be updated (use the callback function **`CallbackFcn`** of the sliding bar to perform the update). Instead of re-plotting the whole signal, the easiest way to update the plot is just calling **`axis`** with the new axis limits. Use the property **`'Value'`** of the sliding bar to get the current position and then compute the new axis limits accordingly, corresponding to a viewing window of 1000 samples, as before.
- When the `Play` button is pushed, use **`soundsc`** to play the signal **`X`** (that you have previously stored into the **`handles`** structure).

[2]Recall that MATLAB does not work with references or pointers to objects but only with copies of them. Therefore, any modification made to a copy of the object has no effect until the working copy replaces the original object.

- The `Close` button just closes the display window (but not the others).

If the user does not close the window and a new sequence of keys is introduced, then pushing the `Send` button must result into refreshing the display window, thus recomputing the signal vector **`X`** (perhaps with a different length) and modifying the signal plot accordingly.

Help window

- Every time the `Help` button in the keypad window is pushed, the help window is created (if needed) and shown. You simply need to display some help information in a "text"-style **`UIControl`**.
- The `Close` button just closes the help window (but not the others).

Additional features

- [QUITE ADVANCED] A second sliding bar could be used in the display window to control the width of the plotted portion of the signal. It could be used to select different widths ranging from 100 samples to the whole signal.
- [ADVANCED] In addition to the push buttons in the DTMF keypad (which are activated with the mouse) you could also provide support to the keyboard. Then the numeric keys together with the keys '`a`', '`b`', '`c`', '`d`', '`*`' and '`#`' would produce the same effect than the corresponding push buttons. You could also use the BACKSPACE key for `DEL`, the ENTER key for `Send`, the '`?`' key for `Help` and the ESC key for `Close`. This can be done with the callback function **`KeyPressedFcn`**.
- [REALLY ADVANCED, ONLY FOR EXPERTS] When running multiple instances of the application several keypad and display windows are generated. However, it only make sense having a single help window. Therefore, the `Help` button creates a help window only if no such window exists on the screen (even when a help window was generated by another keypad window instance). This behaviour can be implemented with a **`persistent`** variable in a function containing a handle to the help window. Notice that, it is the latest closing keypad window which must be responsible to close the (possibly existing) help window.

GUI testing. Consider the following suggested tests to check the performance of the GUI:

- √ Run the main function. The keypad window appears showing the empty key sequence and all buttons are enabled except `DEL` and `SEND`.
- √ Push `Close`. The window closes.
- √ Run again the main function.
- √ Push the button `3`. A 3 appears in the display and now all buttons are enabled.
- √ Push `DEL`. The display shows the empty sequence and `DEL` and `SEND` are disabled again.
- √ Push more buttons until the sequence reaches the maximum length 20. Then all 4×4 buttons are disabled.
- √ Push `DEL` again. Now all buttons are enabled.
- √ Push `Help`. The help window is created.
- √ Push `Help` again. No new help window is created.
- √ Push `Close` on the help window. The help window closes.

√ Push `Help` again. The help window is created again.

√ Push `Send`. The display window is created and the keypad window goes back to the initial state: it displays the empty sequence and all buttons are enabled except `DEL` and `SEND`. The first 125 ms od the signal is shown in the display window and the sliding bar cursor is in its leftmost position.

√ Move the sliding bar cursor to a middle position. The signal display is updated accordingly.

√ Move the sliding bar cursor to its rightmost position. The signal display is updated accordingly. Check that the rightmost time showed corresponds to the duration of the whole signal (the sum of the tones and the silences corresponding to the sequence).

√ Push `Play`. The DTMF signal is reproduced.

√ Push `Close` on the display window. The display window is closed.

√ Introduce the sequence `123456789` in the keypad and push `Send`. The same behavior as before, but now the signal is shorter (use the sliding bar to check this). Play again to check that the sound is different.

√ Without closing the display window, introduce a new sequence in the keypad (e.g. `6421#AC`) and push `Send` again. The display window is updated correctly.

√ Push `Help` again. Since the help window already existed, no new window is created.

√ Push `Close` on the keypad window. The three windows are properly closed.