GUI4GUI User's Guide

Introduction

Frequently, a computer program requires input parameters to define a specific application prior to running it. For codes that require few input parameters, the usual method to define these parameters is to store them in a file or through commandline arguments. Upon reading these parameters, the computer code then proceed to perform computations or other types of operations. For codes that require more input parameters -- especially under less straightforward conditions -- a Graphical User Interface (GUI) may be preferable to query the code runner for input parameters at runtime. However, writing a GUI can often be time-consuming and the code developer may not be readily familiar with the knowledge necessary to develop a GUI.

With this in mind, the GUI4GUI package (GUI4GUI.zip) is developed to build GUIs automatically based on users providing data that describe the details of the GUI components, such as menus, and their associated actions. The programmer needs no knowledge of MATLAB GUI development fundamentals or usages of GUIDE, the MATLAB GUI development environment.

The GUI4GUI package consists of two tiers of GUIs: the main GUI and an optional secondary GUI. We will discuss the main GUI first and defer the discussion of the secondary GUI until later.

The main GUI has eight menu tasks with pre-determined names. The main GUI builder creates each of these tasks exactly the same way. The contents and functionalities of each menu task, however, are completely determined by the user through a menu definition data file. Currently, each menu task can accomodate up to 3 levels of menu items. The first level consists of up to 9 menu items. Each of these items can in turn cascades into the next, or second, level of up to 9 menu items. Further, the second level menu items can each spawns into a third level of 9 menu items. A menu item can take on one of these three actions:

- it causes a document (in html, pdf, or ppt format) to be displayed;
- it runs a program, such as an m-file, a mex-file, or other executables acceptable by MATLAB. A secondary GUI may need to be launched to query for runtime input to run the job. More on this later . . .
- it cascades into the next level. If it is already on the third level, it can only take on one of the two preceding actions.

The eight fixed menu tasks alluded to above are:

1. File - contains just an "Exit" menu item, upon selecting it causes the GUI to close, along with any children figures.
2. Model - describes the different models of the program package
3. Articles - provides previously published conference papers, refereed journal articles, or departmental reports.
4. Tutorial - a tutorial on how to use the package
5. Examples - examples are good way to learn the operational details of the package.
6. Run - for users who want to use it to run applications, this is the center of action. To run jobs, it may require run time input data. This may be facilitated by spawning a second GUI that querries the program runner for data needed to run the job.
7. Code - it consists of just one menu item. A click on this item displays a 3-frame html page that lists the name of all the m-files in the user package. Selecting a file on this list displays this particular file.
8. Help - it consists of 3 menu items: Contact, Credit, and License.

GUI4GUI Installation

The installation is quite simple:

1. Unzip GUI4GUI
   It is recommend that you unzip it in the directory of your source file. But anywhere else will do too.

2. Starts MATLAB (>>)
3. In the MATLAB window, include GUI4GUI directory in the search path with "File/Set Path".
4. Unzip mfc2html
   This is needed for the "Code" menu task, see "GUI4GUI m-file list" section for an example.
5. Follow the mfc2html readme file for usage. (You will need an installed Perl)

You will also need to install the mfc2html.zip that is included in the GUI4GUI.zip. Please follow the instructions in the readme file for using it. This is needed to provide a list of all your source files for the "code" menu task described above.

GUI4GUI m-file List

A complete list of m-files of the GUI4GUI package is available here. This is done with the use of the mfc2html package, included in the GUI4GUI.zip.

main_inputfile.m

The contents and operations of the eight menu tasks are defined in the file `main_inputfile.m`. This can be achieved manually by editing or creating the file and follow some rules. This can also be prepared by running the `gui4gui` GUI which will prompt you for the data. To help with preparing menu data, it is recommended that you copy a ready-made template called `main_inputfile.m.DONOTDELETE` into `main_inputfile.m` and then proceed to modify it according to your specifications. The template also serves as a sample input file for viewing and understanding the rules of preparations.

Enclosed below is a representative segment of the main_inputfile.m for menu task 5, "Examples". Data for each task is stored as a structure. The field name, "name", is the menu task name. Each menu item requires 4 parameters. They should be entered on each row. These four parameters are:

- Index -- a 3-digit number. The left-most digit is reserved for the 1st level menu. the middle digit indicates the second level menu and the third level menu corresponds to the rght-most of the three digits. A non-zero integer denotes the menu item for that menu level. A "0" is used as a place holder (i.e., all 3 digits must be in place). For example, "100" means menu item 1 of level 1. Similarly, "300" is menu item 3 of level 1. On the other hand, "110" means menu item 1 of level 1 and menu item 1 of level 2 (middle digit). "130" means menu item 3 of level 2 cascaded from menu item 1 of level 1.
- Label -- the name you want the menu item to appear.
- Type -- the type of action; use 'doc1' if you want to open a html, pdf, or ppt file if the application user chooses that label, use 'run1' for a pre-defined secondary GUI (more later). 'run2' is to exit the GUI. This is used in the 'File' menu task. There is another pre-defined secondary GUI that collects data for an array as used in Complement Coding. This is classified as 'run3'. If 'run1' and 'run3' secondary GUI do not fit your requirement, you will have to create, say, 'run4' type. If Type does not apply for a menu item, enter '' (i.e., NULL).
- Ops -- Many of the menu items' type is 'doc1', i.e., to display a html or pdf file. For this, the Ops (or operand) would be the name of the html or pdf file. For 'run1', 'run3', the Ops (operation) would be the name of the MATLAB function, along with any necessary input and output arguments, such as 'rand(n, m)'. If Ops doesn't apply for the specific menu item, enter ''.

These four parameters are associated with each menu item. If you plan to have a combined N menu items for the three levels, then you need to enter a total of N*4 entries. For ease in data entry, the parameters are stored in a linear array, Task (5).string. You do not need to specify the total number of menu items. You can enter the menu items in any order you find most convenient or easy to do. Here is an example for Task(5), 'Examples':

```
   Task(5).name = 'Examples';     % name of the menu task is the string 'Examples'
   Task(5).string = { ...         % this task has 3 levels of menu items
      100 'Low coherence (0%)'  ''  '' ...   % level 1 (leading of 3 digits)
menu item 1
      110 'Input'  'doc1'  'lowCoh_input.html' ...   % level 2 (2nd of 3
digits) menu item 1
      120 'Non-directional transient cells'  'doc1'
'lowCoh_non-directional_transient_cells.html' ...
```

```
      .  .  .  .  .  .  .  .  .  .  .  .  .
      170 'Decision cells and decision gating'  ''  '' ...
      171 'Reaction time task'  'doc1'  'lowCoh_RT_task.html' ...
      172 'Fixed duration task'  'doc1'  'lowCoh_FD_task.html' ...
      .  .  .  .  .  .  .  .  .  .  .  .  .
      .  .  .  .  .  .  .  .  .  .  .  .  .
      300 'High coherence (51.2%)'  ''  '' ...      % 00 is place holder for
levels 2 and 3
      310 'Input'  'doc1'  'highCoh_input.html' ...
      .  .  .  .  .  .  .  .  .  .  .  .  .
      .  .  .  .  .  .  .  .  .  .  .  .  .
      372 'Fixed duration task'  'doc1'  'highCoh_FD_task.html' ...
   };
```

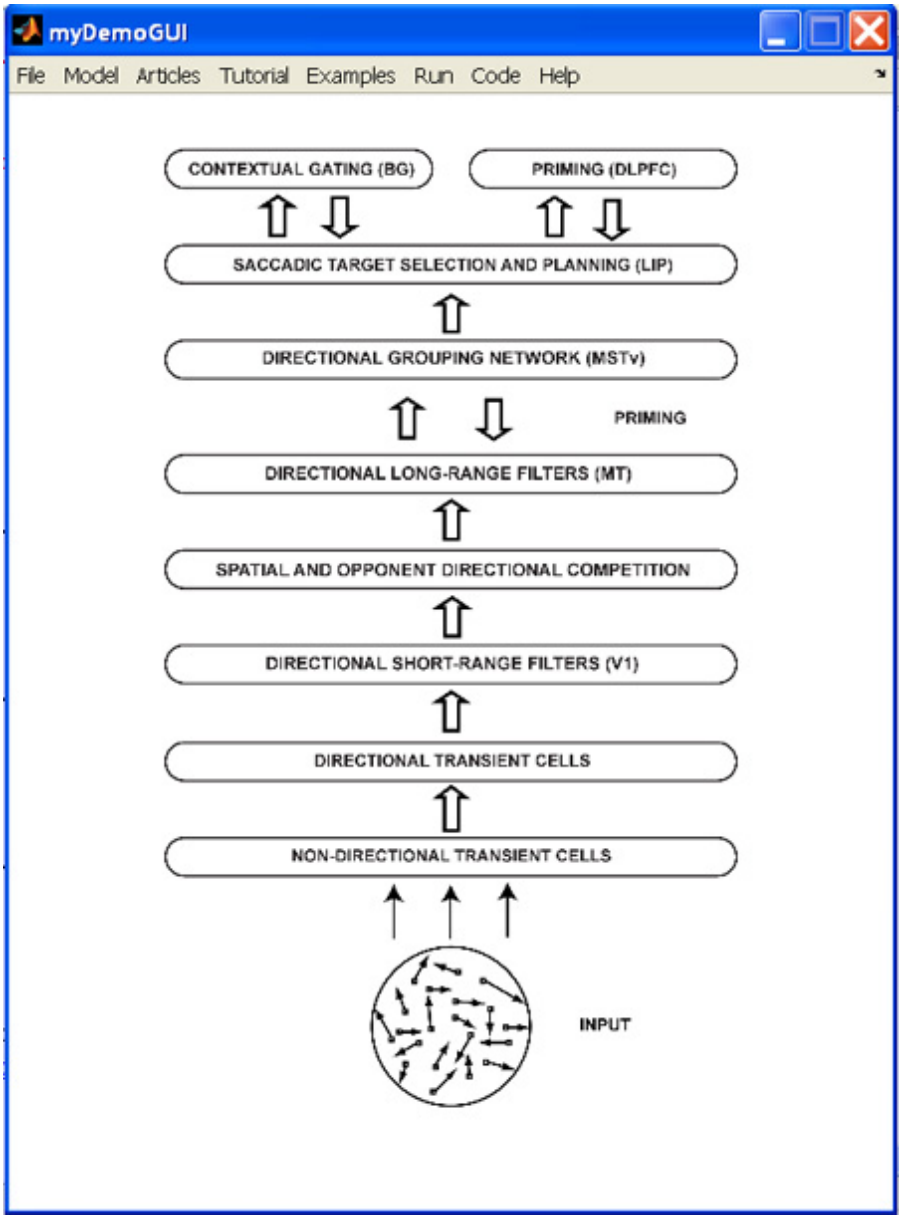
There are other data that are needed to control the appearance of the GUI, such as font size, font weight, or background color. The suggestion is to start with the settings in the `main_inputfile.m` template. If you find the pre-defined setting agreeable, just keep them. Otherwise, change them as you see fit.

```
    GID.guiName = 'myDemoGUI';            % name of your GUI
    GID.Nmenus = Nmenus;                  % number of menu tasks (8)
    GID.frontImage = 'MODE_new.jpg';      % the image file to be display on the GUI
    GID.bgcolor = 'white';                % background color
    GID.fontsize = 10;                    % base font size
    GID.fontweight = 'bold';              % font weight
    GID.position = [0.10 0.10 0.60 0.60]; % size of the window (normalized to 1)
    GID.resize = 'off';                   % do not resize
    GID.task = task;                      % all the tasks defined are absolved
into struct GID
```

buildMainGUI.m

This is a utility m-file that takes menu data definitions from a file provided by the user and generates a programmatic GUI (as opposed to one generated by MATLAB's GUIDE GUI generator). Functionally, a programmatic GUI and GUIDE GUI are identical. However, a GUIDE generated GUI has the advantage that a user can readily change the pre-generated GUI's layout easily. However, such a GUI is developed for a specific application. Making it general for dissimilar applications is very difficult, if not impossible. On the other hand, a programmatic GUI can be generated automatically given the proper information (i.e., data) for a much wider applications.

Using the template sample input file, a `myDemoGUI` will be generated. This is the GUI for the CNS MODE code.



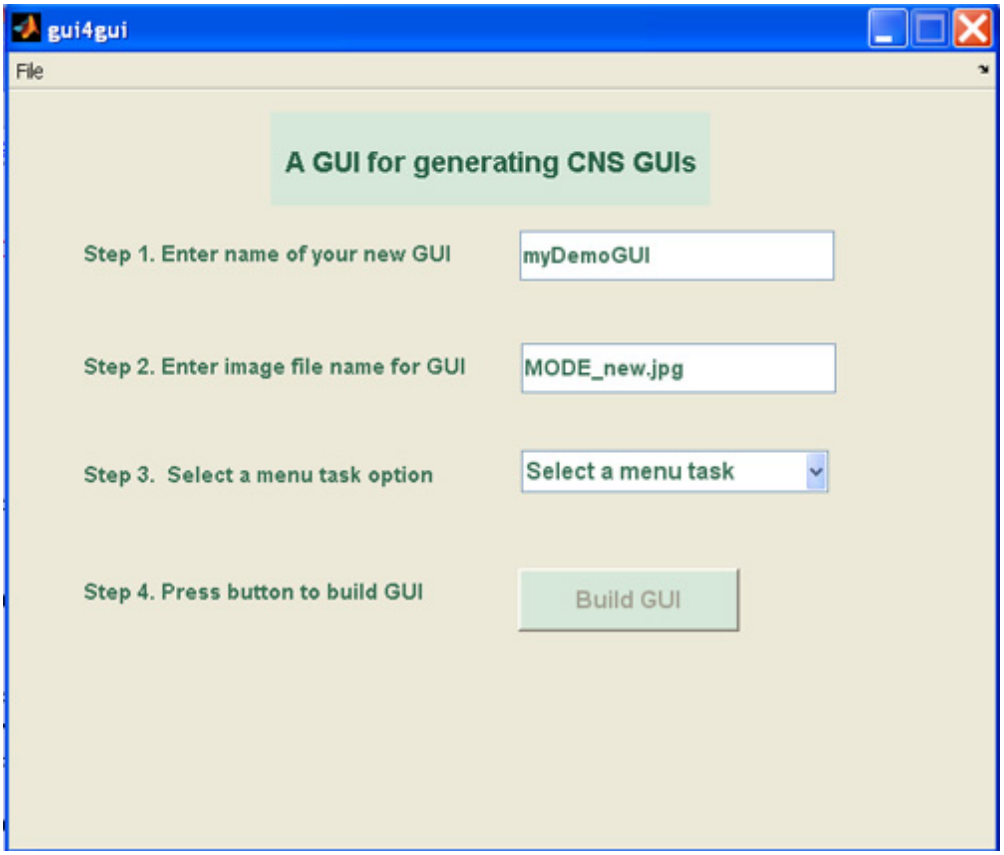
Operational Instructions For Building The Main GUI

There are two methods for building the main GUI:

- Manual Method
  1. `>> copyfile main_inputfile.m.DONOTDELETE main_inputfile.m      % copy file`
  2. `>> edit main_inputfile.m      % modify content for your code`
  3. `>> GID = main_inputfile     % GID is the menu data struct`
  4. `>> buildMainGUI(GID)      % build GUI with GID`
- GUI Method
  1. `>> copyfile main_inputfile.m.DONODELETE main_inputfile.m`
  2. `>> gui4gui    % it overwrites main_inputfile.m; then runs buildMainGUI`

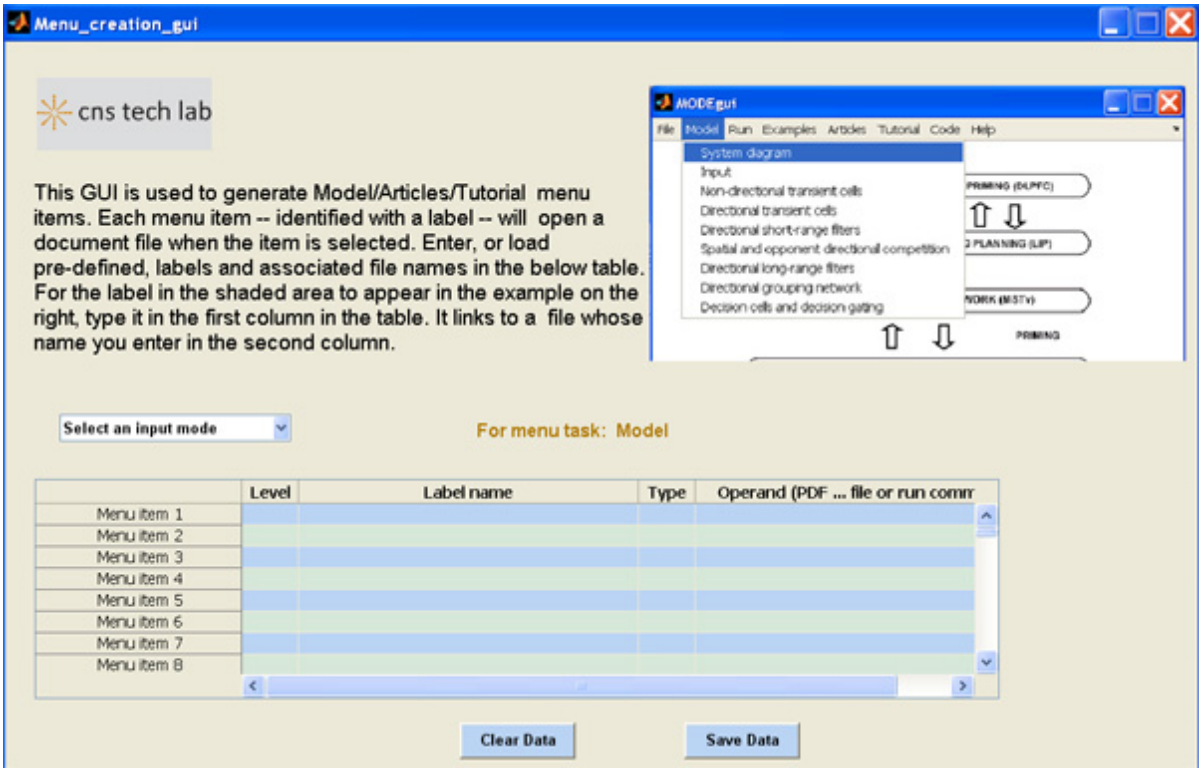
`Data file main_inputfile.m.DONOTDELETE comes with the GUI4GUI.zip.`

>> gui4gui



You have the options of doing one menu task at a time or enter menu data for all eight tasks at once. Whether you choose the former or latter method, each time a task is completed, it will be saved to a temporary file tempout.mat. Keep an eye on the MATLAB command window. The gui4gui will update you on the status of the progress. For example, each time a task is done, it will indicate which tasks are defined and which ones remain undefined. Note that the intelligence of gui4gui is fairly limited. If you come back to the same task more than once, it would not be aware of that but it would not prevent you from doing so either.

Once a task is selected, a GUi will prompt you for data. You must choose whether to enter data from scratch or load the pre-defined menu task data from the main_inputfile.m. Once completed, don't forget to press "Save" to save data and close the window to return to gui4gui.



Secondary GUI -- buildRun1GUI.m

As mentioned above, selection of a menu item that causes an m-file or other executables to run may require another GUI to prop up to query for runtime input data needed by the m-file or executable. Unlike the main GUI which has been standardized, the second level GUI must cater to the need of specific codes. As such, it is not efficient, if not impossible, to design a general GUI template that could accomodate wide-ranging needs of these codes. It is expected that a small number of differing secondary GUI will be needed to address this variant. Usually, this type of GUIs requires few or no menu items. But it will generally reauire a small number of data to be collected.

Instructions for building the 'run1' GUI:

Manual Method

1. `>> copyfile run1_inputfile.m.DONOTDELETE run1_inputfile.m  % copy file`
2. `>> edit run1_inputfile.m  % modify content for your code`
3. `>> GID = run1_inputfile  % GID is the menu data struct`
4. `>> buildRun1GUI(GID(1))  % build GUI with GID(1)`
5. `>> buildRun1GUI(GID(2))  % build GUI with GID(2)`

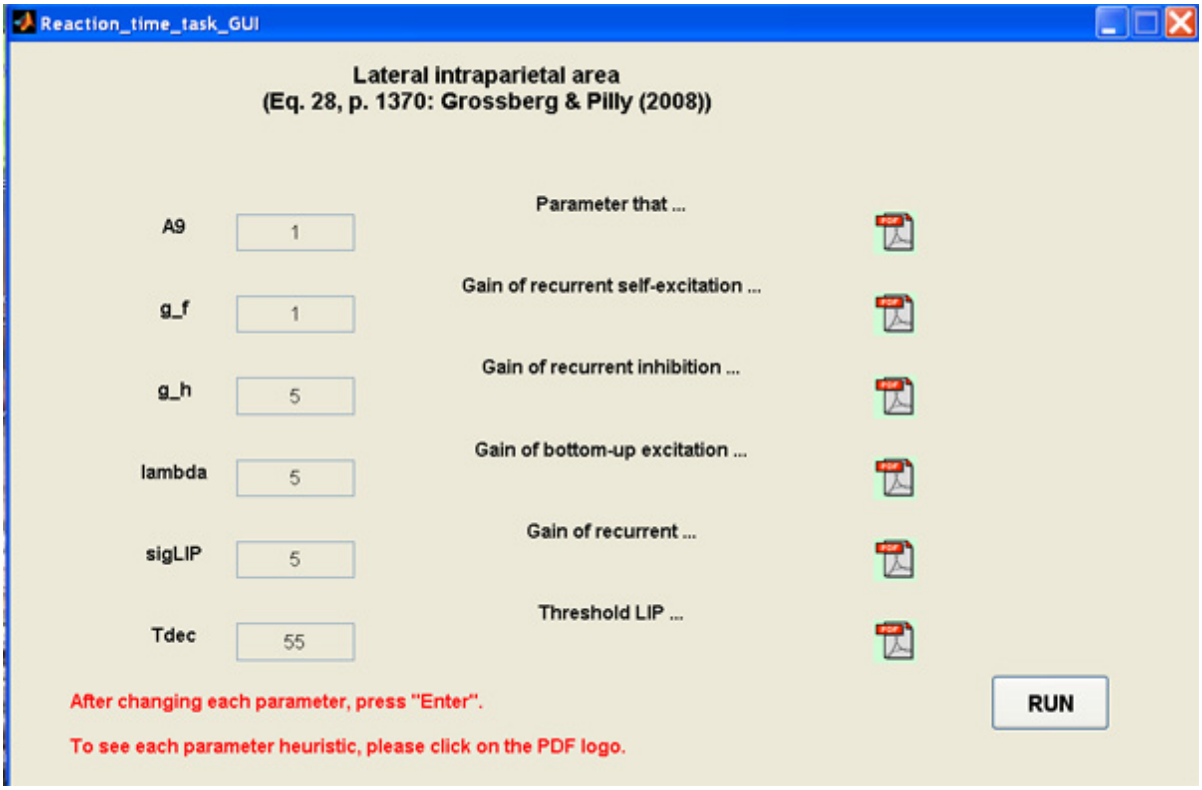
Data file run1_inputfile.m.DONOTDELETE comes with the GUI4GUI.zip.

run1_inputfile.m

As can be seen, this inputfile is considerably simpler than its counter-part, main_inputfile.m. The data layout though is very similar. The only significant departure is GID is a vector, instead of a scalar because there are multiple applications of the 'run1' type.

```
`
    GID(1).guiName = 'Reaction_time_task_GUI'; % name of your GUI
    GID(1).bgcolor = 'white';  % background color
    GID(1).fontsize = 10;      % font size
    GID(1).fontweight = 'bold';% font type: 'bold', 'normal'
    GID(1).position = [0.1 0.1 0.6 0.6];  % [lower-left, lower-bottom,
width, height] range (0,1)
    GID(1).resize = 'off';  % whether window is resizable
    GID(1).title = ['Lateral intraparietal area!(Eq. 28, p. 1370: Grossberg &
Pilly (2008))'];
    GID(1).fn1 = 'After changing each parameter, press "Enter".';  % footnote 1
    GID(1).fn2 = 'To see each parameter heuristic, please click on the PDF
logo.';  % footnote 2
    GID(1).run = 'RTtasknew(A9,g_f,g_h,lambda,sigLIP,Tdec)';  % what to do
when "run" is pressed
    % I could do without the argument list as it can readily be derived
    % from the string list. Good or bad ?
    GID(1).string(1:6,1:4) = { '' };
    % input parameter, default value, description, further reading material
    GID(1).string(1,1:4) = { 'A9' '1' 'Parameter that ...' 'A9_heur.pdf' };
    GID(1).string(2,1:4) = { 'g_f' '1' 'Gain of recurrent self-
excitation ...' 'g_f_heur.pdf' };
    GID(1).string(3,1:4) = { 'g_h' '5' 'Gain of recurrent inhibition ...'
'g_h_heur.pdf' };
    GID(1).string(4,1:4) = { 'lambda' '5' 'Gain of bottom-up
excitation ...' 'lambda_heur.pdf' };
    GID(1).string(5,1:4) = { 'sigLIP' '5' 'Gain of recurrent ...'
'sigLIP_heur.pdf' };
    GID(1).string(6,1:4) = { 'Tdec' '55' 'Threshold LIP ...' 'Tdec_heur.pdf' };
```

If you examine Task(6) ('Run') of main_inputfile.m.DONOTDELETE, you would find that the name 'Reaction_time_task_GUI' was used. If you run the GUI generated by buildMainGUI using that inputfile and the right menu item was selected, the action would spawn the Reaction_time_task_GUI which prompts the user to enter data (A9, g_f, g_h, lambda, sigLIP, and Tdec) to run the application RTtasknew as defined for GID(1).run.




```
For this 'run1' GUI template, up to six editboxes can be used. If you are
familiar with GUIDE, you could also use it to generate your own secondary
GUI without using having to prepare the run1_inputfile or run the buildRun1GUI.
```

Contact info: Kadin Tseng, 617-353-8294, kadin@bu.edu

Date created: June 30, 2009