# Graphical User Interfaces (GUIs)

# EP 271 Team Lab

## 1. Introduction

Throughout this semester we have worked on solving practical engineering problems using Matlab and Maple. Up until now, though, we have removed the need for user input during a program's runtime. This has been useful because it has allowed us to treat multi-step problems as one continuous problem. Now, it is useful to show how Matlab can be used to develop programs that can be used by other people, allowing them to insert their own input. Further, the use of graphics will be used to allow this process to be more intuitive to end users.

A graphical user interface (GUI), sometimes pronounced "gooey", is probably what many of you thought a computer program was before taking this course. Your desktop screen filled with all its clickable folders and files is a good example of a GUI, as is the touchscreen device you probably have in your pocket or next to your computer. The reason GUIs are useful for users is because they remove the need for them to understand how a program works. Many of you have probably seen or heard of toddlers that now have access to iPads. Whether you love or hate this idea, the fact is that GUIs are now an integral part of your daily life… so let us begin…

## 2. GUI Basics in Matlab

Before stepping into the two GUIs used for this lab, it is necessary to go over common Matlab commands used for GUI construction. The first two are ones you should already be well acquainted with: "figure" and "axes". Although not a focus of previous lectures, both these commands accept many parameters to specify various properties associated with them. For GUI development, a very common property is 'units', which specifies how position/location/size values are interpreted (the specific nomenclature depends on what is using the 'units').

To start, we will create a figure covering the whole right side of your screen with an axes spanning the top-half of the figure.  This is done with the following two commands:

```
  curFig  = figure( 'units' , 'normalized' , 'position' , [ 0.5 0 0.5 1 ] ) ;
  curAxes =   axes( 'units' , 'normalized' , 'position' , [ 0 0.5 1 0.5 ] ) ;
```

Next, we will add a panel to the bottom-half of this figure, which will eventually hold buttons. The reason a “uipanel” is useful is because it allows easier placement with normal units.  You can think of the uipanel as a subfigure, which works with normalized units. To add one, type:

```
  curPanel = uipanel( 'units' , 'normalized' , 'position' , [ 0 0 1 0.5 ] ) ;
```

Now that we have a figure that is completely filled, we will add several types of buttons.  Each button is added using “uicontrol” by specifying its “style.”  Currently in Matlab, the available styles are: checkboxes, editable text fields, list boxes, popup menus, pushbuttons, radio buttons, sliders, text boxes, and toggle buttons.  For this lab, we will limit ourselves to push buttons and sliders.

For this example, we will have one pushbutton plot a line in the axes and one pushbutton clear them.  The slider will be used to change a plotted line’s angle.  To place these uicontrols inside the curPanel, we will use the “parent” property.  This can be done with the following code:

```
  hold all , axis manual   %  needed for plot consistency

  plotButton  = uicontrol( 'style' , 'pushbutton' , 'parent' , curPanel , ...
    'units' , 'normalized' , 'position', [ 0.00 .3 .5 .6 ] , 'string' , ...
    'Plot' , 'Callback' , 'plot( [ 0 1 ] , [ 0 get(angleSlider,''value'') ] )' ) ;

  clearButton = uicontrol( 'style' , 'pushbutton' , 'parent' , curPanel , ...
    'units' , 'normalized' , 'position', [ 0.5 .3 .5 .6 ] , 'string' , ...
    'Clear', 'Callback' , 'cla' ) ;

  angleSlider = uicontrol( 'style' , 'slider'     , 'parent' , curPanel , ...
    'units' , 'normalized' , 'position', [ 0 0.1 1 .1 ] , 'value' , 0.5 ) ;
```