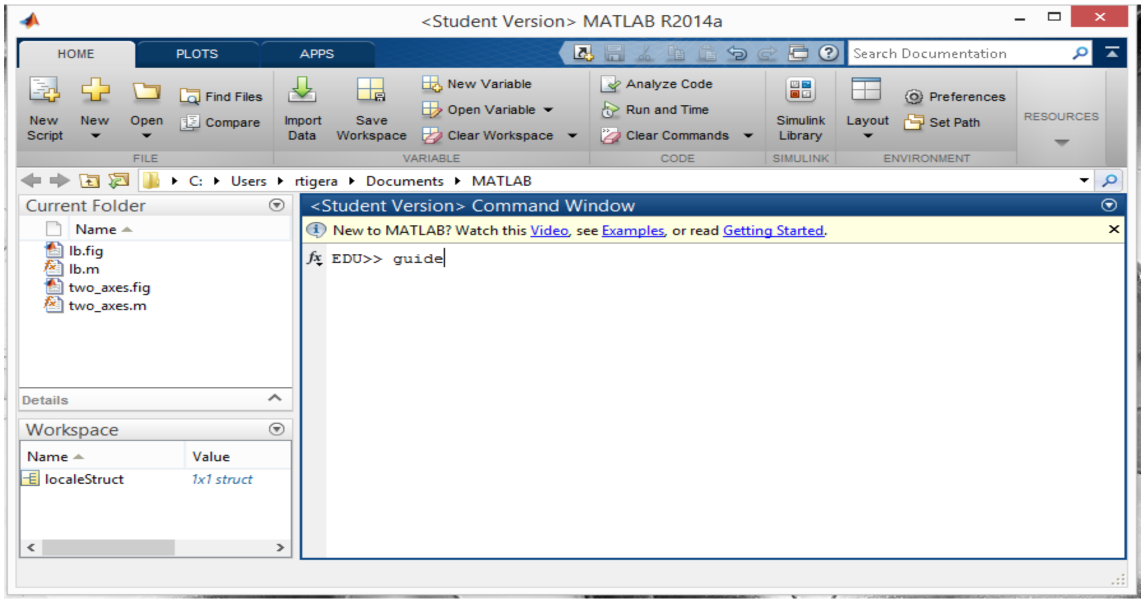
Guide to creating GUI's in MATLAB 2014

Introduction:

There are two ways to create a GUI in Matlab. The first is to use the Guide tool provided by Matlab to design your GUI and, based on your design, Matlab will supply the necessary program structure. This structure can be customized with a Matlab program and commands to engage the user. The second method is to write out the entire program in the Editor screen, save and then run. For the purpose of this manual, the procedure will give step by step instructions using the Guide tool in Matlab.

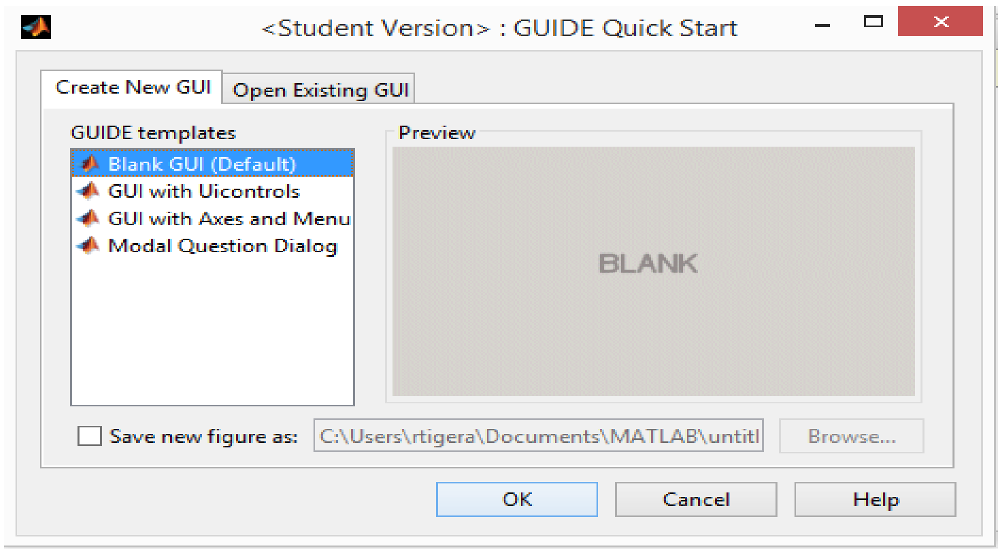
Instructions:

Open Matlab to the Command window and type in "guide" where prompted:

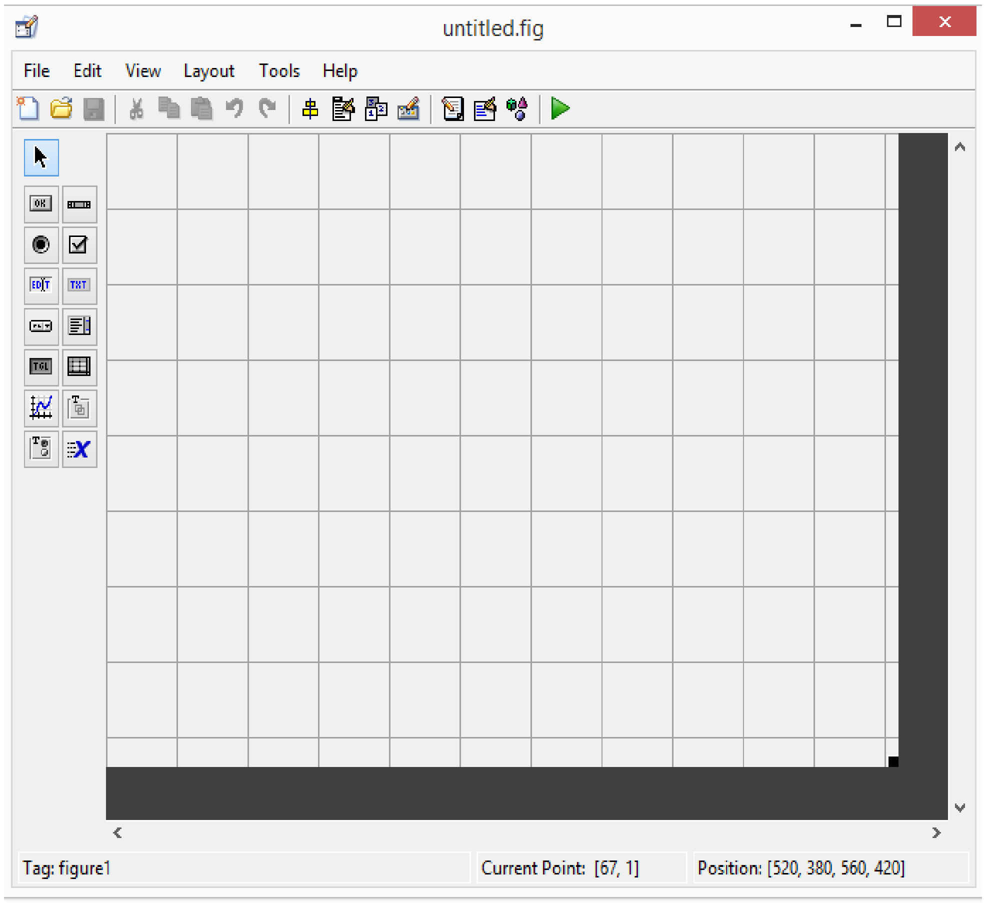


```
EDU>> guide
```

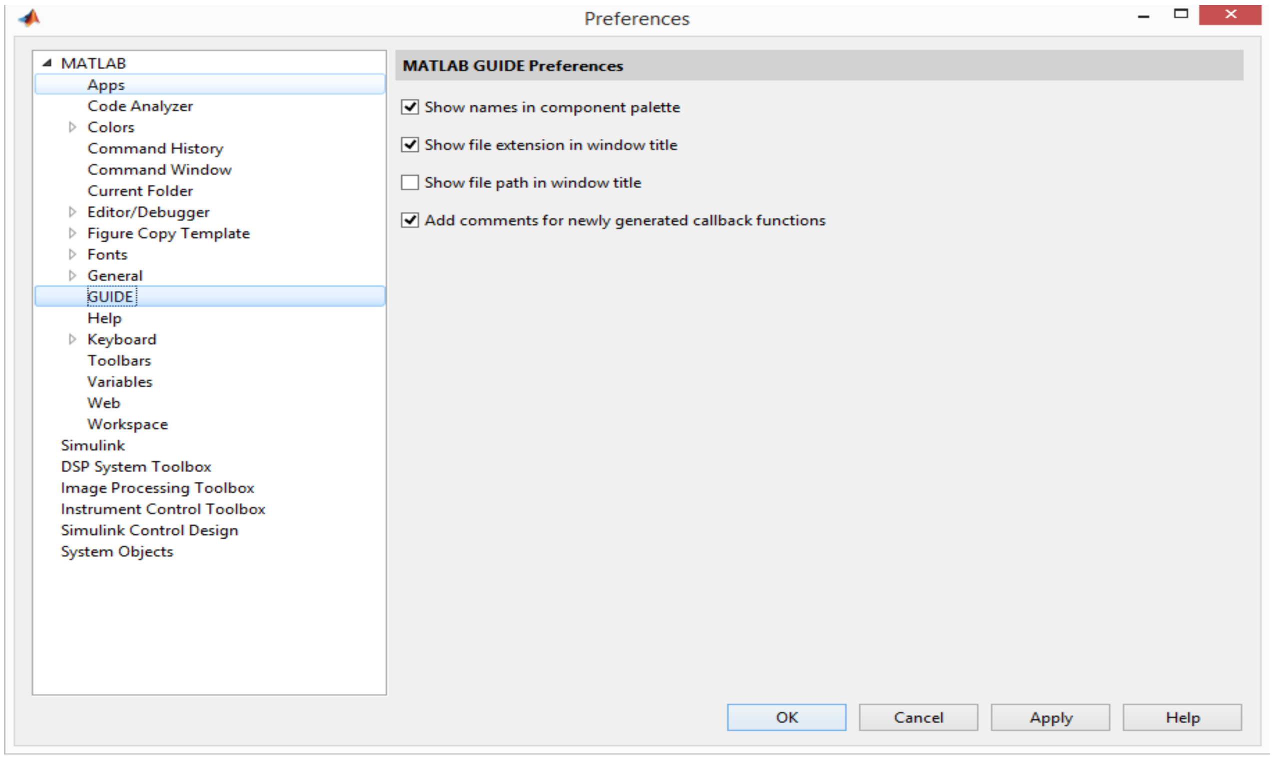
Upon hitting enter, a dialogue box will pop up providing different types of GUI's to be created:



Once this Quick Start guide pops up, any number of choices can be made when creating a new GUI or, if the “Open Existing GUI” is selected, the creator will be able to access and edit a previous GUI. In this case, “Blank GUI (Default)” is chosen. Once selected, click “OK” and this screen will display:



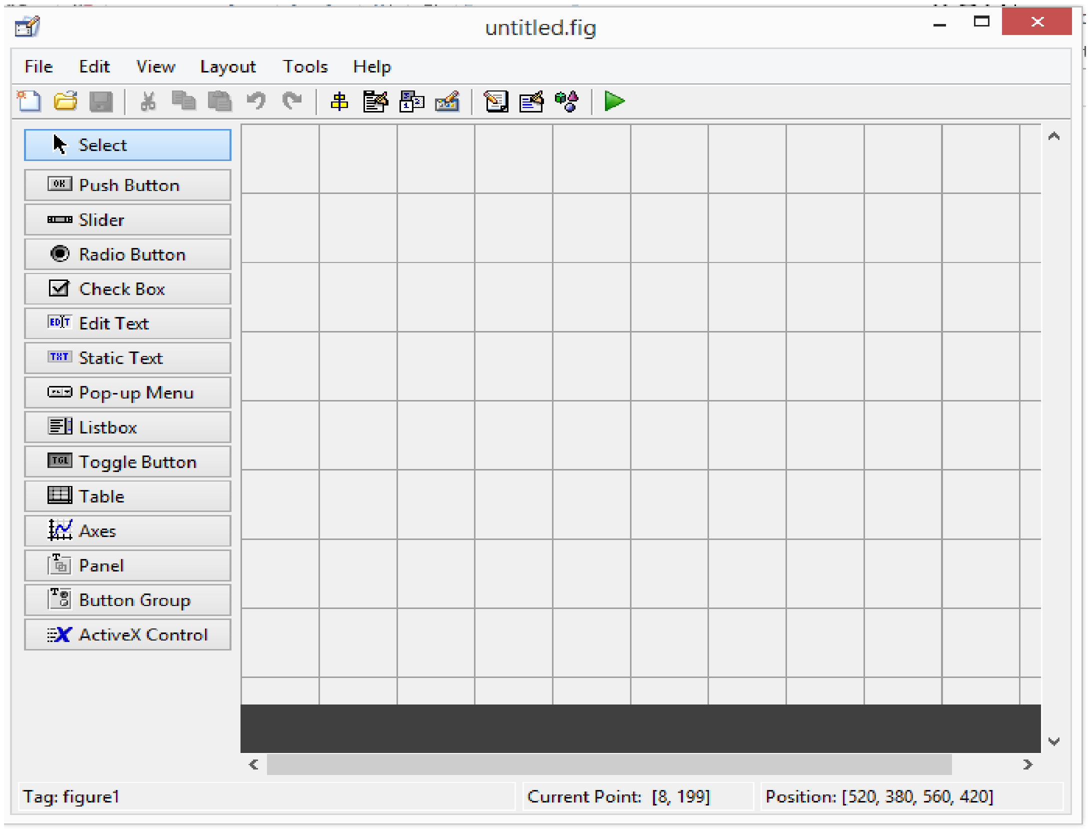
This is the blank screen for the GUI creator. Select the File tab and then choose Preferences. This menu will be displayed:



MATLAB GUIDE Preferences

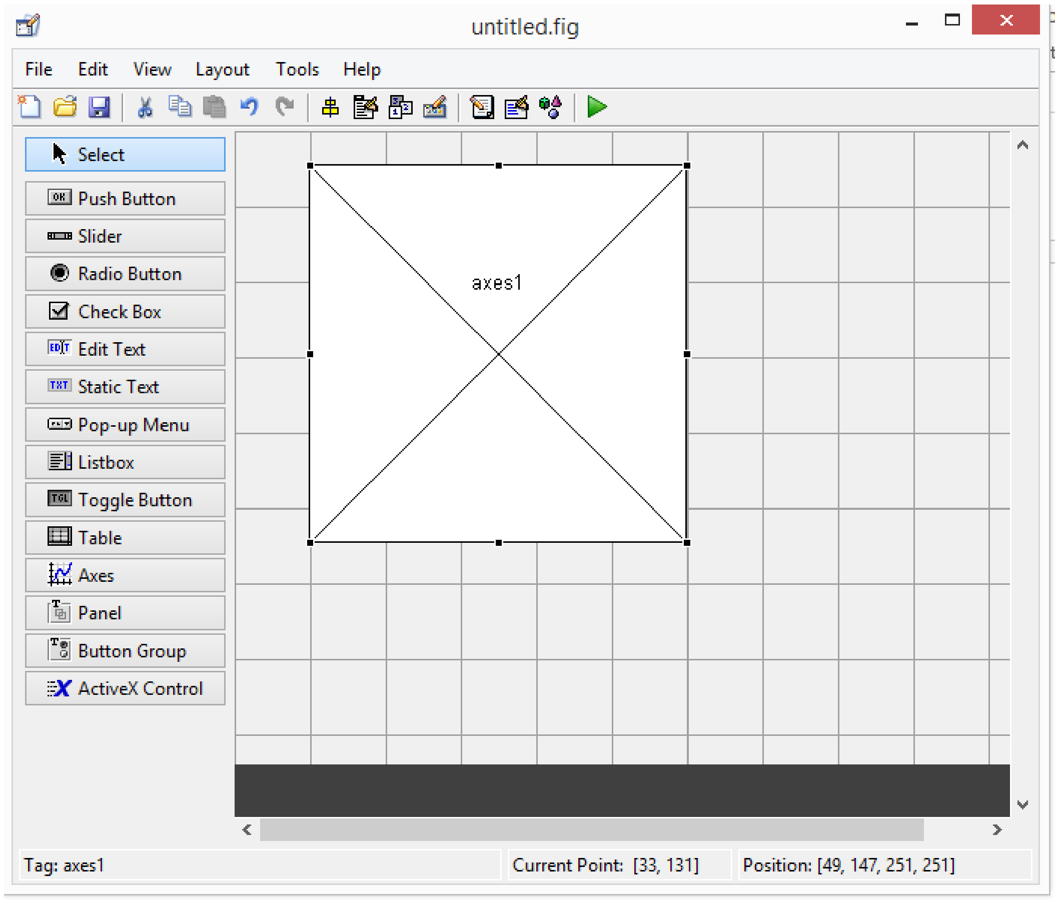
- [✓] Show names in component palette
- [✓] Show file extension in window title
- [ ] Show file path in window title
- [✓] Add comments for newly generated callback functions

The option at the very top says "Show names in component palette". Select this option and click "OK". A new version of the GUI will pop up that is more user friendly:



Tag: figure1 Current Point: [8, 199] Position: [520, 380, 560, 420]

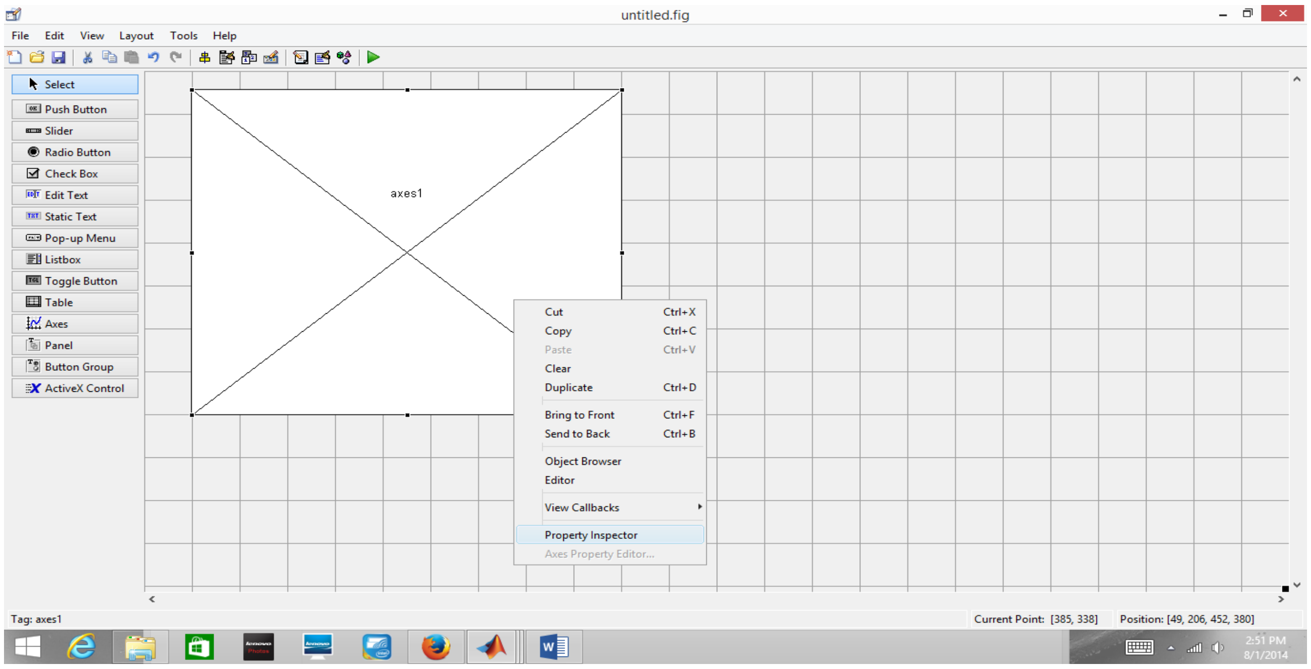
Notice how the tools on the left side of the GUI have a label next to the icon. This will be helpful in order to ensure the right tool is chosen for the GUI. Now you are ready to create. Once a tool is selected, it can be placed on the GUI at the user's discretion. In this next picture, an axes has been chosen and placed on the GUI screen.



Tag: axes1 Current Point: [33, 131] Position: [49, 147, 251, 251]

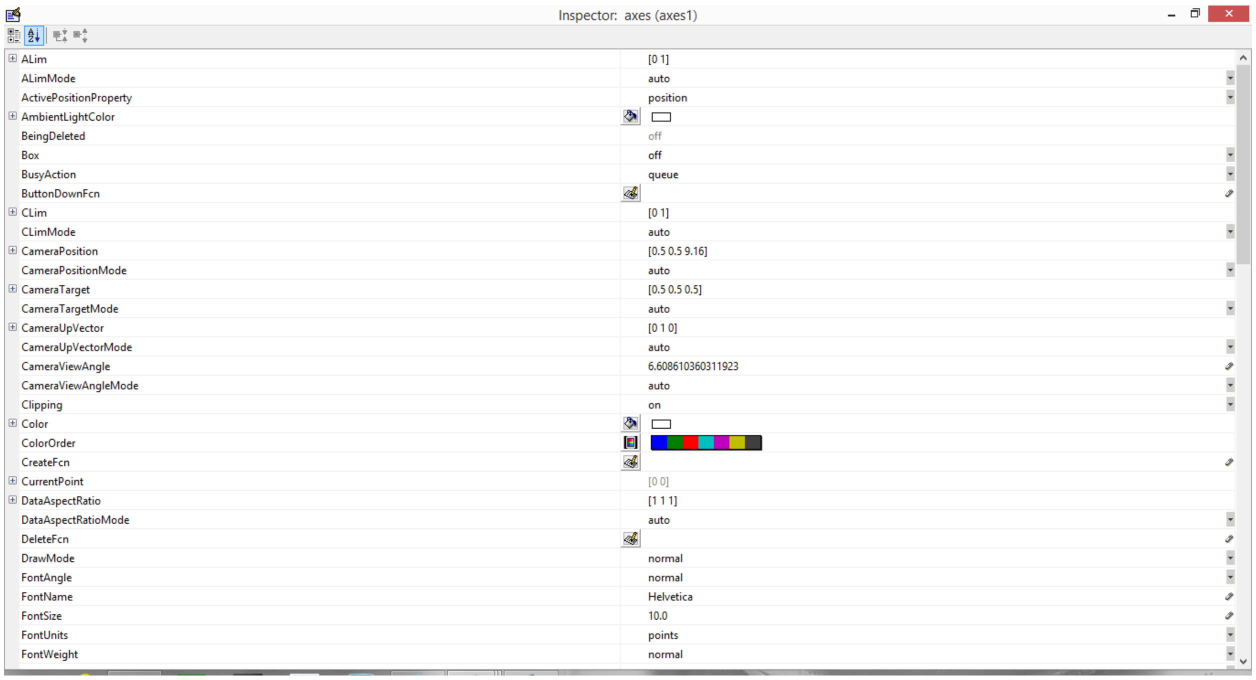
Once the axes have been placed, the size and/or location can be changed by clicking on the axes. If the mouse arrow is hovering in the center of the axes window, it becomes a four pronged star and the axes can be moved. If the mouse arrow hovers over the axes boundary, it will become a two pronged arrow that can increase or decrease the size of the window. This can be done after selecting any of the tools icons and allows the user to customize their GUI as they see fit.

Now we are going to walk through the steps to label a Tools icon for programming. Hover over the axes table and right click. Select Property Inspector, which is the second to last option from the drop down menu.



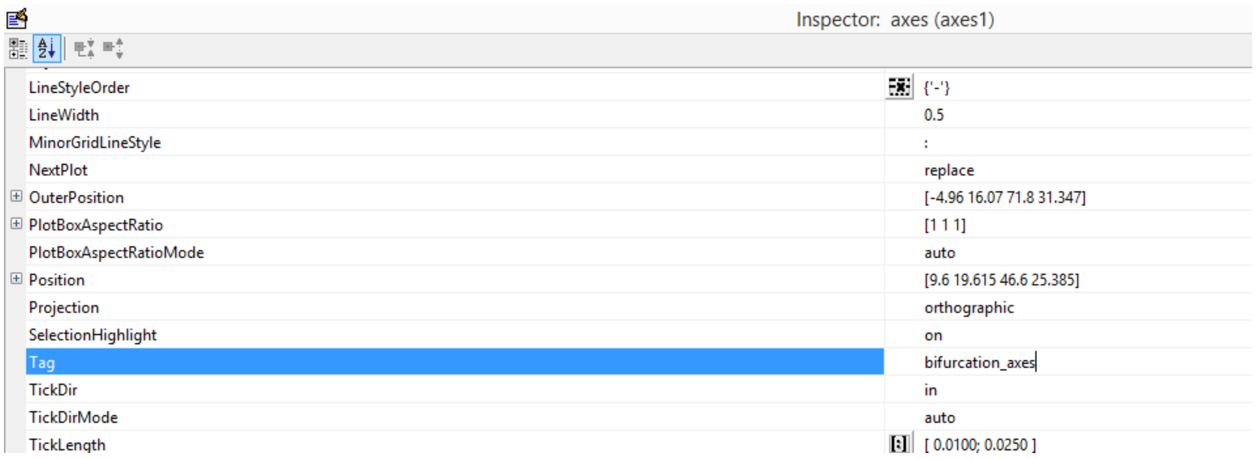
Tag: axes1 Current Point: [385, 338] Position: [49, 206, 452, 380]

Select the Property Inspector and a menu of options will display for the particular icon you have selected. In this case, a menu for the axes will display.



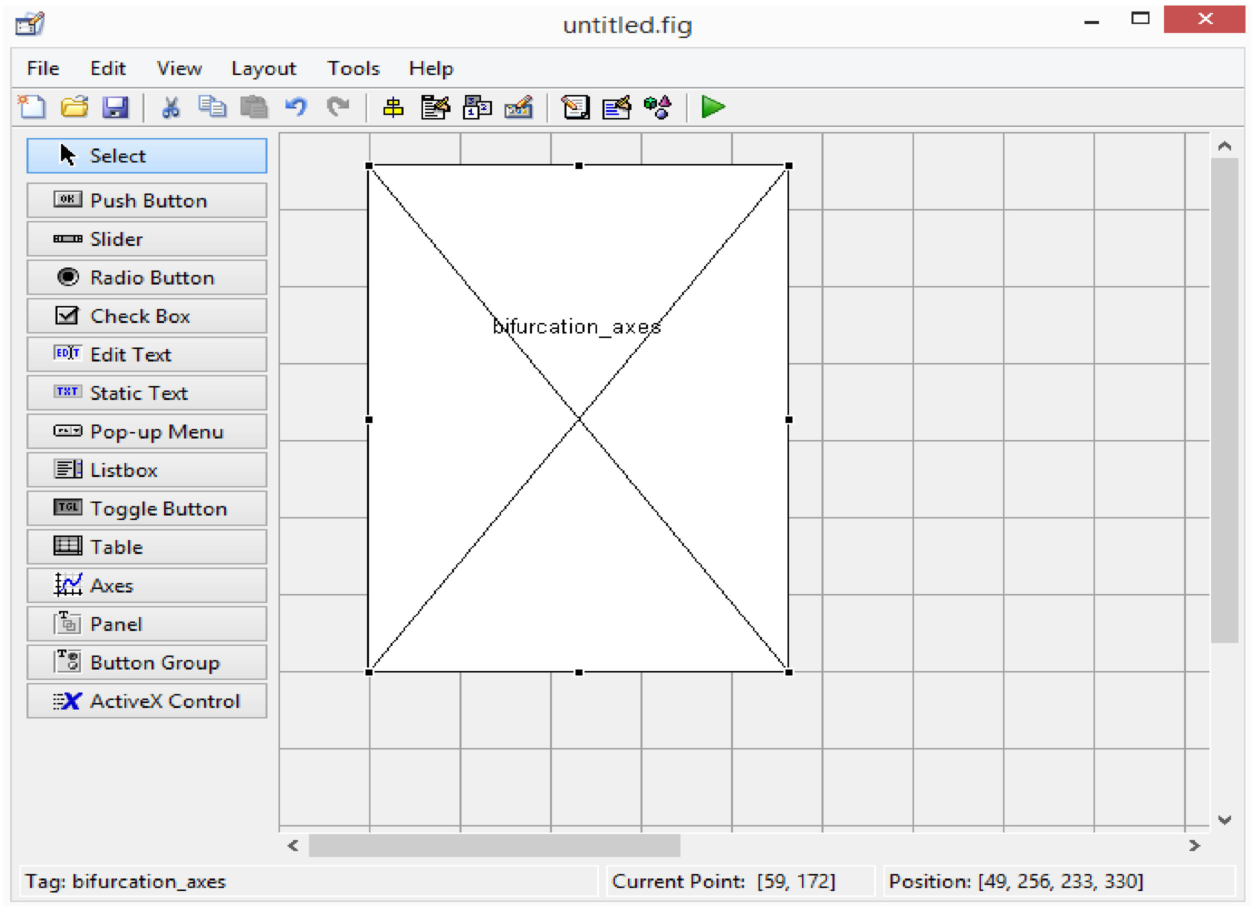
ALim	[0 1]
ALimMode	auto
ActivePositionProperty	position
AmbientLightColor	▭
BeingDeleted	off
Box	off
BusyAction	queue
ButtonDownFcn	
CLim	[0 1]
CLimMode	auto
CameraPosition	[0.5 0.5 9.16]
CameraPositionMode	auto
CameraTarget	[0.5 0.5 0.5]
CameraTargetMode	auto
CameraUpVector	[0 1 0]
CameraUpVectorMode	auto
CameraViewAngle	6.608610360311923
CameraViewAngleMode	auto
Clipping	on
Color	▭
ColorOrder	🟦🟩🟥🟦🟪🟨⬛
CreateFcn	
CurrentPoint	[0 0]
DataAspectRatio	[1 1 1]
DataAspectRatioMode	auto
DeleteFcn	
DrawMode	normal
FontAngle	normal
FontName	Helvetica
FontSize	10.0
FontUnits	points
FontWeight	normal

Scroll down toward the bottom until you see the “Tag” option. Change the name of the axes to something that is easy to recognize amongst lots of code. For this example, the axes tag is changed to “bifurcation_axes”. Make sure to press Enter after the name is changed and press the red “x” to exit the screen.


Inspector: axes (axes1)

LineStyleOrder	{'-'}
LineWidth	0.5
MinorGridLineStyle	:
NextPlot	replace
OuterPosition	[-4.96 16.07 71.8 31.347]
PlotBoxAspectRatio	[1 1 1]
PlotBoxAspectRatioMode	auto
Position	[9.6 19.615 46.6 25.385]
Projection	orthographic
SelectionHighlight	on
Tag	bifurcation_axes
TickDir	in
TickDirMode	auto
TickLength	[0.0100; 0.0250]

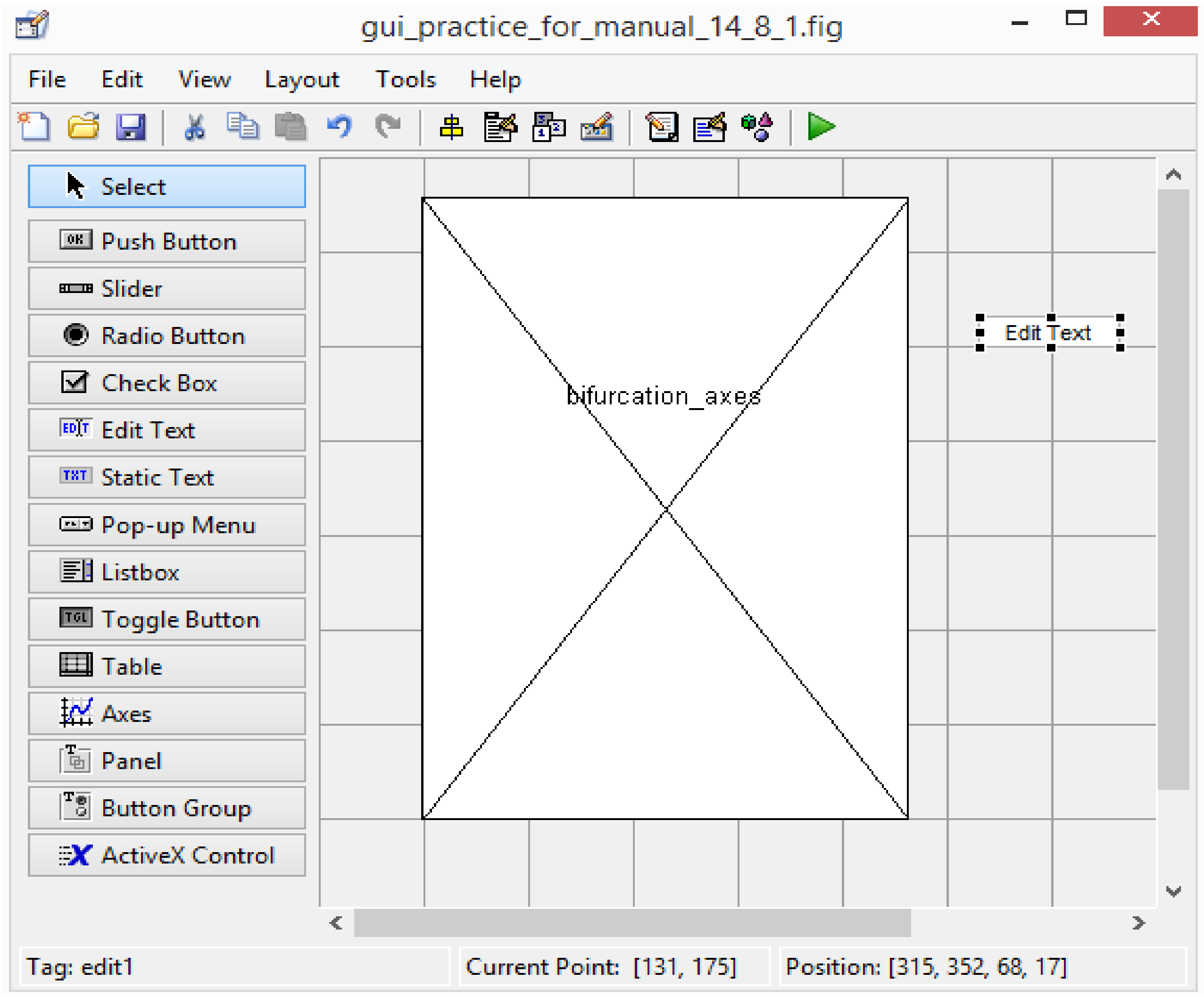
Once the “Tag” name is changed, the new name will be displayed on the axes in the GUI as seen below.



Tag: bifurcation_axes Current Point: [59, 172] Position: [49, 256, 233, 330]

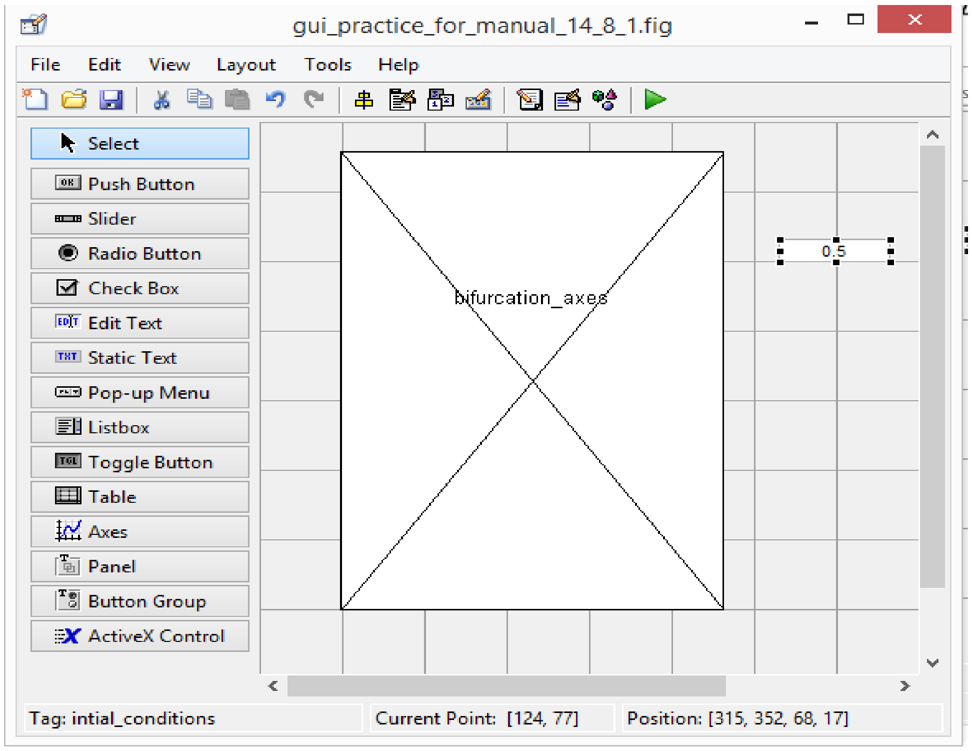
This is an important step that should be completed for each Tool icon added to the GUI because it will help in keeping track of the icons in the code. As shown in the next steps, it can get complicated as multiple options of the same icon are added to the GUI. For our next icon, let’s add an edit text box. This will engage the user to input the data they would like to see the GUI manipulate and plot in the axes window.

Move the mouse over to the component palette and select “Edit Text box”. Using the mouse arrow, insert an “Edit text box”:



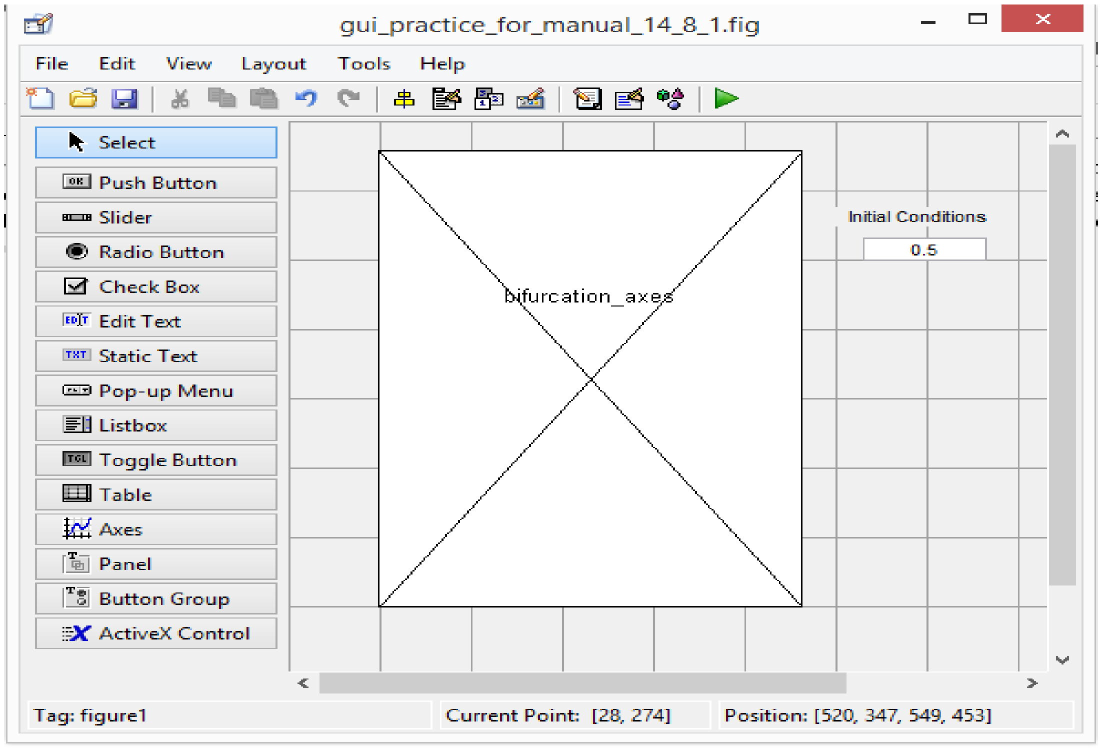
Tag: edit1 Current Point: [131, 175] Position: [315, 352, 68, 17]

Using the same procedure as the axes, select the “Edit Text box” and right click. Choose “Property Inspector” and change the “Tag” to a name that is relevant for the text box. In this case, the “Tag” will be initial conditions. Also, notice the “String” line two lines above the “Tag”. In this line, add the value that will show up as the first value for the GUI to process. In this case, the initial condition will be 0.5.



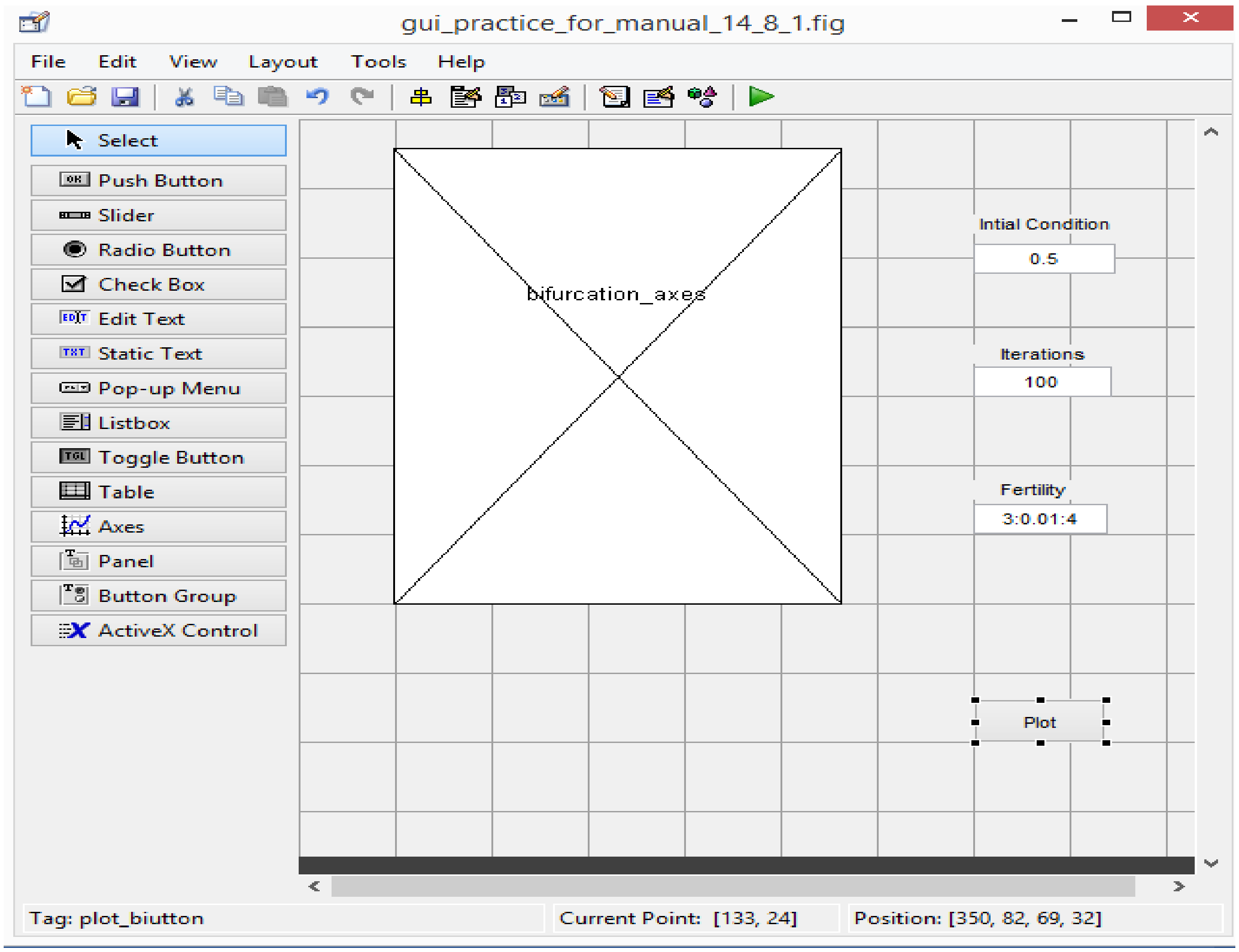
Tag: intial_conditions Current Point: [124, 77] Position: [315, 352, 68, 17]

The next step is to add “Static Text” above the “Edit Text Box” in order to inform the user what type of values need to be entered into this particular “Edit Text Box”. To create a “Static Text”, select the “Static Text” box on insert it above the “Edit Text box”. Select the box, right click, chose Property Inspector and change the “String” line to Initial Conditions. Press Enter and then exit. The phrase “Initial Conditions” will appear above the box.



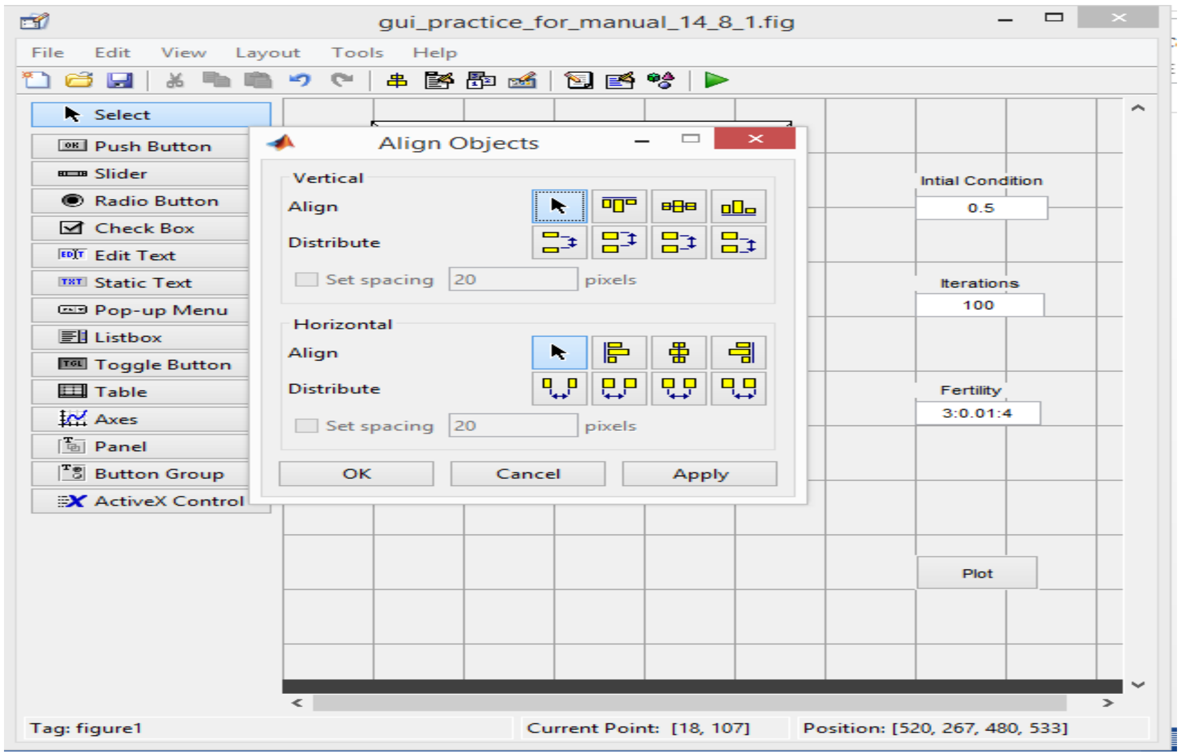
Tag: figure1 Current Point: [28, 274] Position: [520, 347, 549, 453]

The creator can add any number of icons using the same process described above. For this example, two more "Edit Text box" icons and a "Push button" will be added. If you are following along with the example, it should look like the figure below. Note: remember to change the "Tag" and "String" for the "Edit Text Box" and "Push Button", but just the "String" for the "Static Text Box".

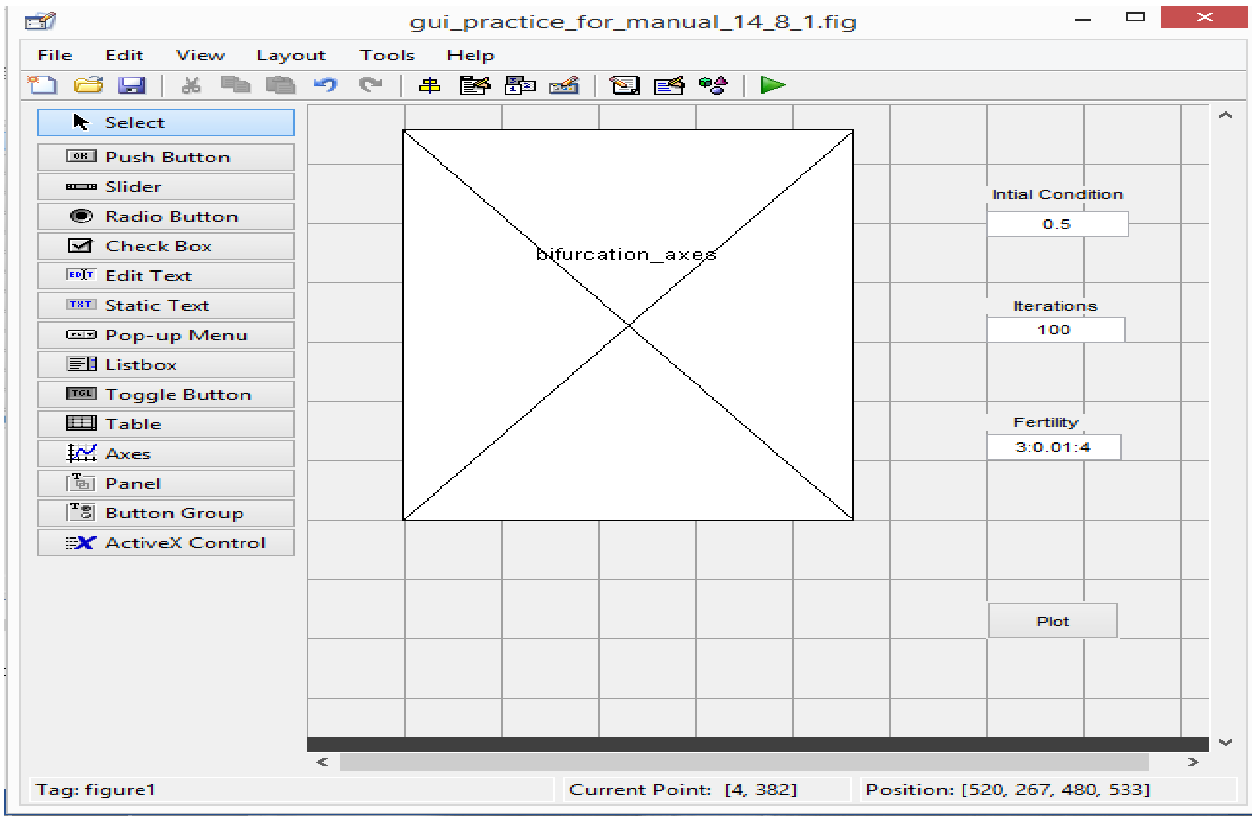


Tag: plot_biutton Current Point: [133, 24] Position: [350, 82, 69, 32]

Once the design is completed, the tools selected for the GUI can be aligned using the "Align Objects" option. Go to the Tools tab and select "Align Objects". A menu should display like the one in the figure below.

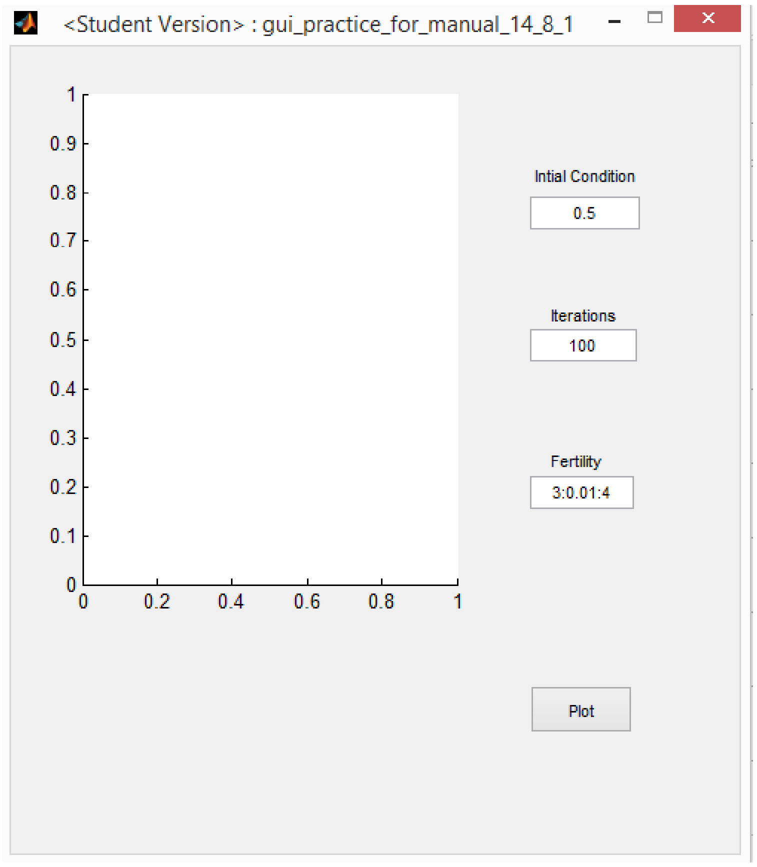


For this example, select the “Align Left” icon, which is the second button to the right on the “Horizontal, Align” line. This will align all the selected tool buttons in a straight line starting from the left. Note: To select all the tools to be aligned, hold down the “Ctrl” button while choosing in order to select all.



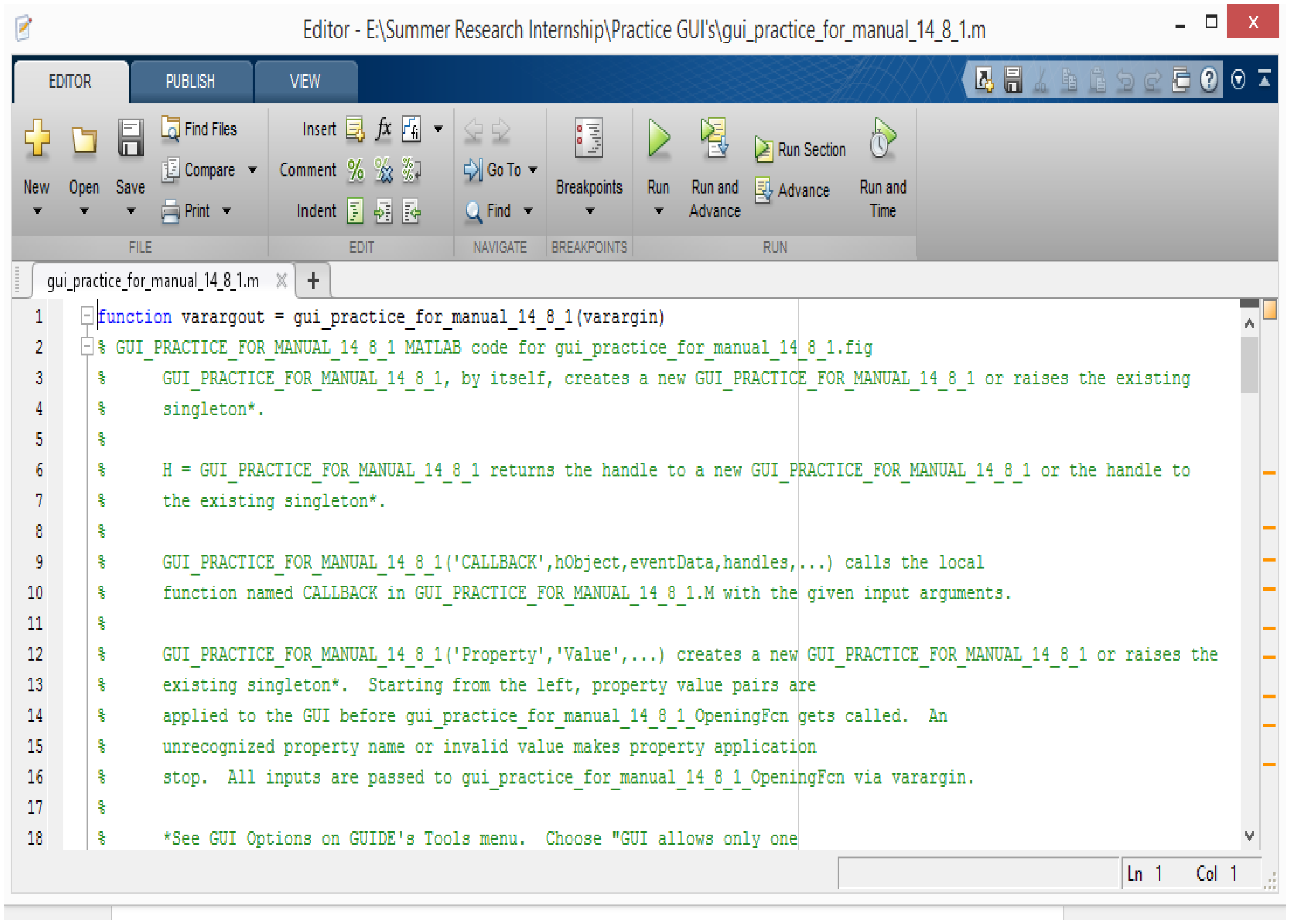
If you haven't been saving it periodically, now it is an important time to save. For this example, the GUI design is completed and now ready to press the green arrow to "Run". By pressing the green arrow, this will generate a program scaffold in order to code the GUI and a GUI screen showing your design.

Here is the GUI after pressing the green arrow:



Notice the information you put in the "String" line when manipulating the Property Inspector appears in the "Edit Text Box" as the first values to be used by the GUI. Also, if you press the plot button nothing will appear in the axes because the GUI has not been customized yet.

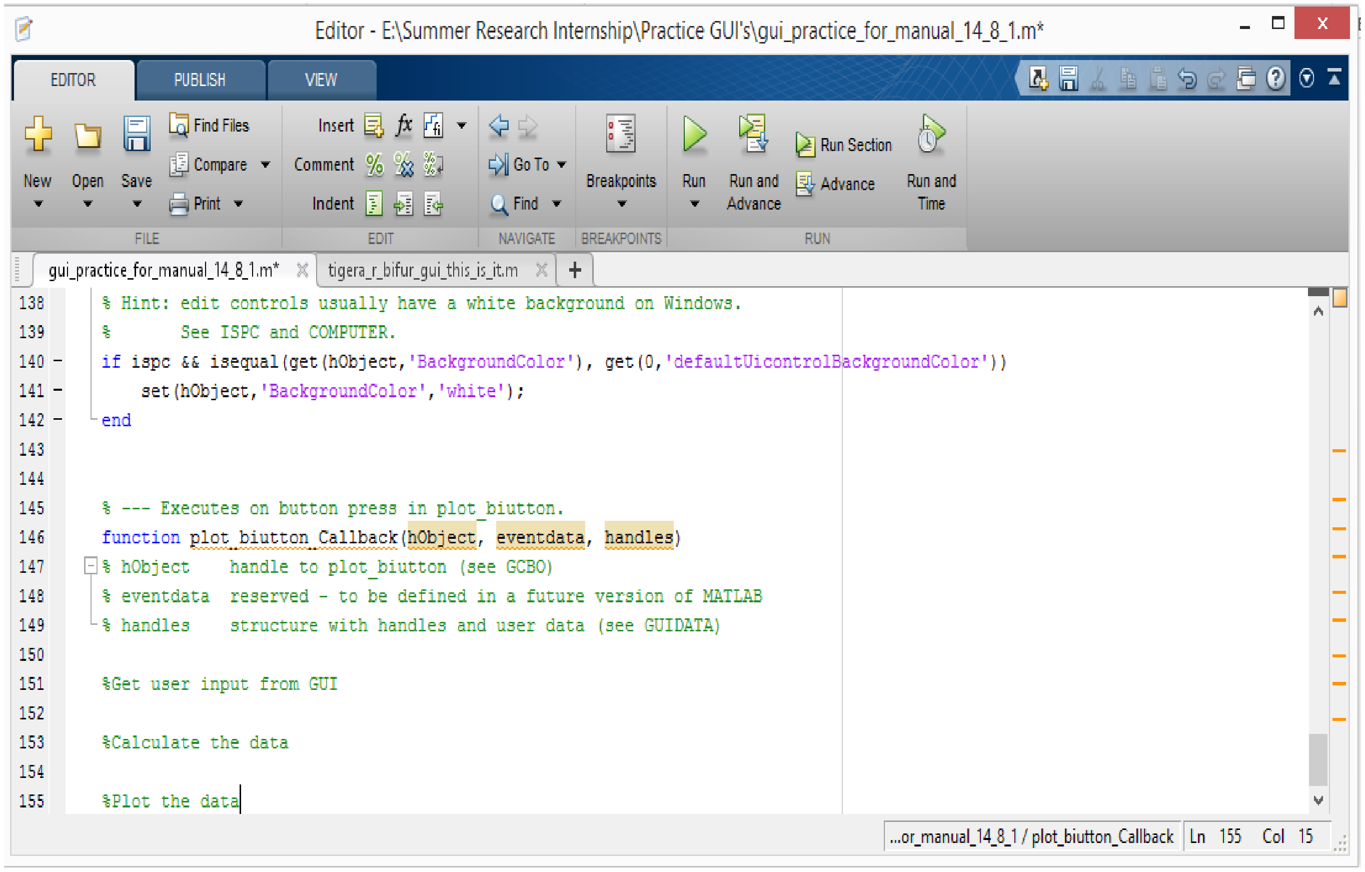
The second screen that will appear is the program scaffold.



```
function varargout = gui_practice_for_manual_14_8_1(varargin)
% GUI_PRACTICE_FOR_MANUAL_14_8_1 MATLAB code for gui_practice_for_manual_14_8_1.fig
%      GUI_PRACTICE_FOR_MANUAL_14_8_1, by itself, creates a new GUI_PRACTICE_FOR_MANUAL_14_8_1 or raises the existing
%      singleton*.
%
%      H = GUI_PRACTICE_FOR_MANUAL_14_8_1 returns the handle to a new GUI_PRACTICE_FOR_MANUAL_14_8_1 or the handle to
%      the existing singleton*.
%
%      GUI_PRACTICE_FOR_MANUAL_14_8_1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI_PRACTICE_FOR_MANUAL_14_8_1.M with the given input arguments.
%
%      GUI_PRACTICE_FOR_MANUAL_14_8_1('Property','Value',...) creates a new GUI_PRACTICE_FOR_MANUAL_14_8_1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before gui_practice_for_manual_14_8_1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to gui_practice_for_manual_14_8_1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
```

From this screen, each button will be programmed in what's called a Callback. A callback references the "Edit Text Box" for each input and inserts them into the calculations. Each one of those boxes will be assigned a variable and that variable will be in the equations used to plot the data. Furthermore, there is no need to erase or remove any of the program that initializes when the green arrow is pushed. The only task now is to add the program you wish to analyze in the GUI.

The part of the program that calculates and plots the data will be under the Plot button. Scroll down to the last line of your program (if following the example, it is line 150), and write in %Get user input from GUI, % Calculate the data, and %Plot the data. Throughout this process, it is good practice to label the different components of the program as it makes it easier to debug if necessary. With regard to labeling the components of the program, it does not have to be the specific labels used in the procedure. Please find a method that works best for you and your program. See the picture below:


Editor - E:\Summer Research Internship\Practice GUI's\gui_practice_for_manual_14_8_1.m*
EDITOR
PUBLISH
VIEW
New
Open
Save
Find Files
Compare
Print
Insert
Comment
Indent
Go To
Find
Breakpoints
Run
Run and Advance
Run Section
Advance
Run and Time
FILE
EDIT
NAVIGATE
BREAKPOINTS
RUN
gui_practice_for_manual_14_8_1.m*
tigera_r_bifur_gui_this_is_it.m
...or_manual_14_8_1 / plot_biutton_Callback
Ln 155 Col 15

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in plot_biutton.
function plot_biutton_Callback(hObject, eventdata, handles)
% hObject    handle to plot_biutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Get user input from GUI

%Calculate the data

%Plot the data
```

The first part of the GUI to address is inserting the values entered into the Edit Text boxes. These boxes can be changed as many times as desired until the Plot button is selected. The values will then be used to create the graph in the axes. For this case, the Initial conditions box and the Iterations box are a single value while the Fertility box is an array.

For single values, there is an order of code that is necessary in order for the program to call upon the value in the text box. Using the bifurcation curve as an example, here is the code for the initial conditions and iterations input:

```
cla reset
%Get user data from GUI
x = str2double(get(handles.initial_condition,'String'));
n = str2double(get(handles.iterations_,'String'));
```

cla reset is the part of the command that resets the graph each time the GUI is told to plot. This allows the user to only see the plot of the values entered in the GUI and removes the previous plot.

str2double is the part of the command that converts the input of the user in the text box into a value that Matlab can use in the programs.

Get is the part of the command that tells Matlab to retrieve the value from the specified location

Handles is the part of the command that tells Matlab that the values it needs is located in the “Tag” line of the Property Inspector. The specific Tag label is the next part of the command.

Initial_conditions/Iterations is the name of the Tag line of the edit text box whose value Matlab will retrieve. In this case, the initial conditions value will be used for x in the equation and the iterations value will be n.

‘String’ is the part of the command that tells Matlab which line in the Property Inspector holds the value of interest for the calculations.

In conclusion, these commands tell Matlab to go to the String line in Property Inspector of each individual edit text box based on their label in the Tag line and use that value as the assigned value (x or n) in the calculations.

The line of code for the fertility is listed below:

```
r = eval(get(handles.fertility_input,'String'));
```

The only difference here is that instead of str2double, the code uses eval as its first term. The eval term tells Matlab to analyze these values as an array rather than a single value. In the String line of the Property Inspector, the value added was 3:0.001:4. This is important for the bifurcation curve because, as seen in the calculations section, the fertility factor needs to have the same vector length as the values of x in order to plot the graph. The values of x will be put into an array within the program while the values of r will be evaluated as a range of numbers within the text box.

Before moving on to the calculations part of the GUI, it is recommended that the creator craft their program in the Matlab Editor window before adding it to the GUI structure. This provides a more precise and succinct venue of programming and, once the program is able to run in the Editor script, it can easily be added to the calculations section of a GUI.

Once the Edit text boxes have been assigned as variables, the actual program that performs the calculations follows next. For an explanation of the program’s logic, please see the previous discussion on the creation of the classic bifurcation curve in Matlab. The program is pasted below:

```
%Calculate the data
hold on
x(1)=x;
for i = 1:length(r)
     for j=2:n
        x(j)= r(i)*x(j-1)*(1-x(j-1));
     end
     y = x(1:end);
```

In terms of separating the different functions of the program, this section creates the values for x based on the Edit Text boxes and stores them in an array assigned the value y.

The plot section of the program is pasted below:

```
%plot the data
 plot(handles.bifurcation_axes,r(i),y)
```

```
end
 set(handles.bifurcation_axes,'XMinorTick','on')
 xlabel('Parameters/Fertility')
 ylabel('Population')
 grid on
 hold off
```

A discussion of the plotting section of the program is in the previous section describing the creation of a classic logistic bifurcation curve in Matlab. The important part when setting up a GUI is to have Matlab plot the data. As seen in this line of code:

```
plot(handles.bifurcation_axes,r(i),y)
```

The line begins with the term “plot”, which tells Matlab to plot the data on the tool window with the title “bifurcation_axes” in the “Tag” line. The independent variable is then set to be the fertility values and the dependent is the array created for the results of the equation assigned to the variable x.

Furthermore the line of code below allows the creator to add different components to the axes. In this case, the minor grid lines are added to the axes window.

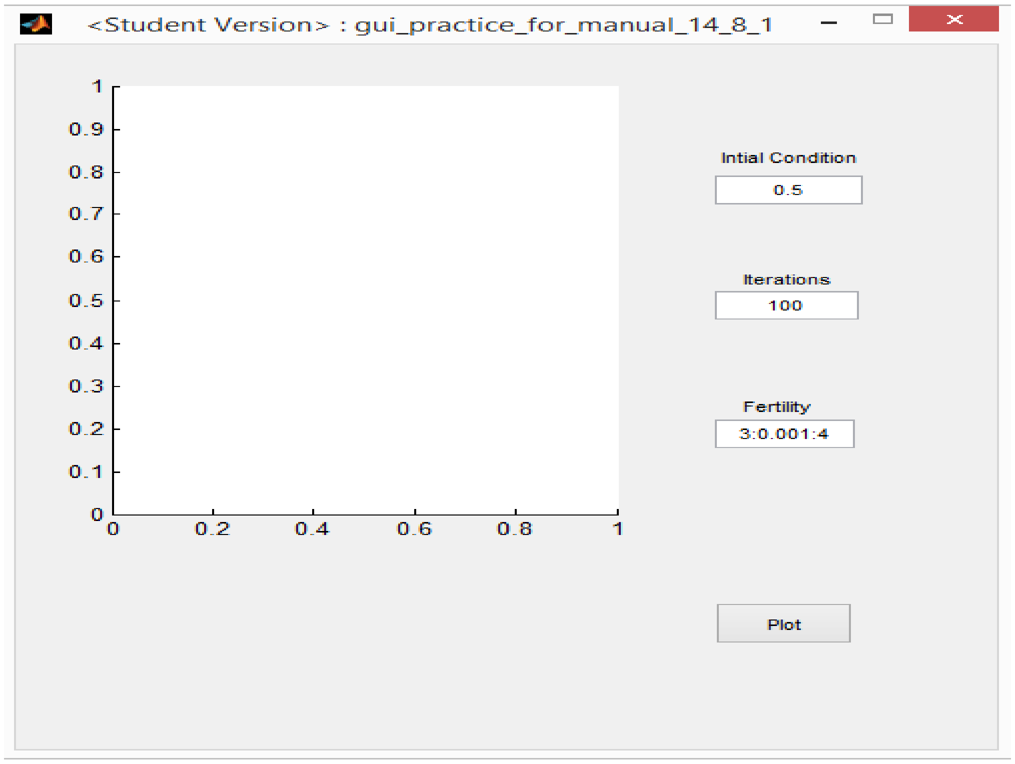
```
set(handles.bifurcation_axes,'XMinorTick','on')
```

The program is now complete. The entire program is added under the plot button section of the GUI structure and looks like this in its entirety:

```
% --- Executes on button press in plot_button.
function plot_button_Callback(hObject, eventdata, handles)
% hObject    handle to plot_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Cla reset

%Get user data from GUI
x = str2double(get(handles.initial_condition,'String'));
n = str2double(get(handles.iterations_input,'String'));
r = eval(get(handles.fertility_input,'String'));

%Calculate the data
hold on
x(1)=x;
for i = 1:length(r)
     for j=2:n
        x(j)= r(i)*x(j-1)*(1-x(j-1));
     end
     y = x(1:end);

%Plot the data
 plot(handles.bifurcation_axes,r(i),y)
end
 set(handles.bifurcation_axes,'XMinorTick','on')
 xlabel('Parameters/Fertility')
 ylabel('Population')
```

```
grid on
hold off
```

During this step by step procedure, the only section manipulated by the creator was under the plot button callback. The rest of the program was generated from customizing the GUI on the GUIDE screen. Pushing the green arrow, the user sees this screen:

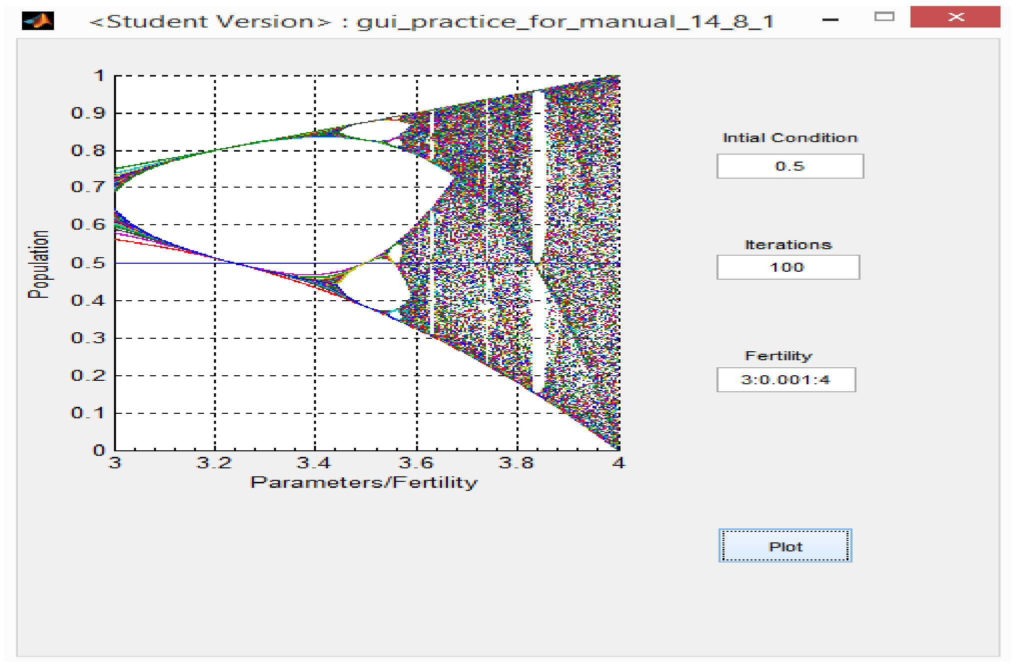


Intial Condition: 0.5

Iterations: 100

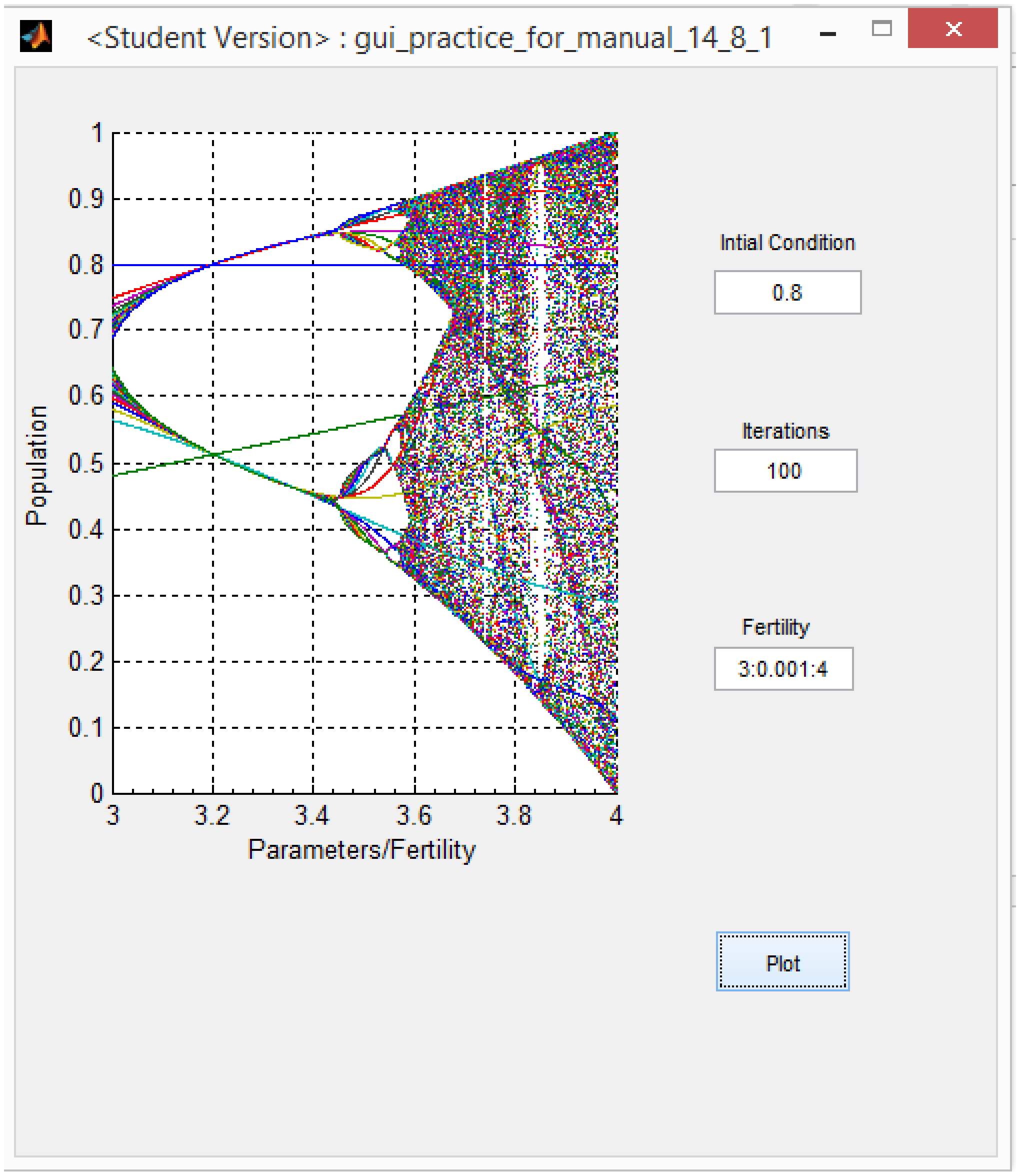
Fertility: 3:0.001:4

From this screen the user can interact with the program. The initial values set in the Property Inspector are visible in the Text boxes. Pressing Plot, the GUI generates this graph:



Intial Condition: 0.5
Iterations: 100
Fertility: 3:0.001:4

This matches the plot generated when the program was just in the Editor script and ran as a common Matlab program. Notice the blue line that runs horizontally from the 0.5 value on the vertical axis. This exists because 0.5 is the initial condition. Now let's check to see if our Callbacks are written correctly. Change the initial condition to see if the blue line that signifies the initial condition moves:

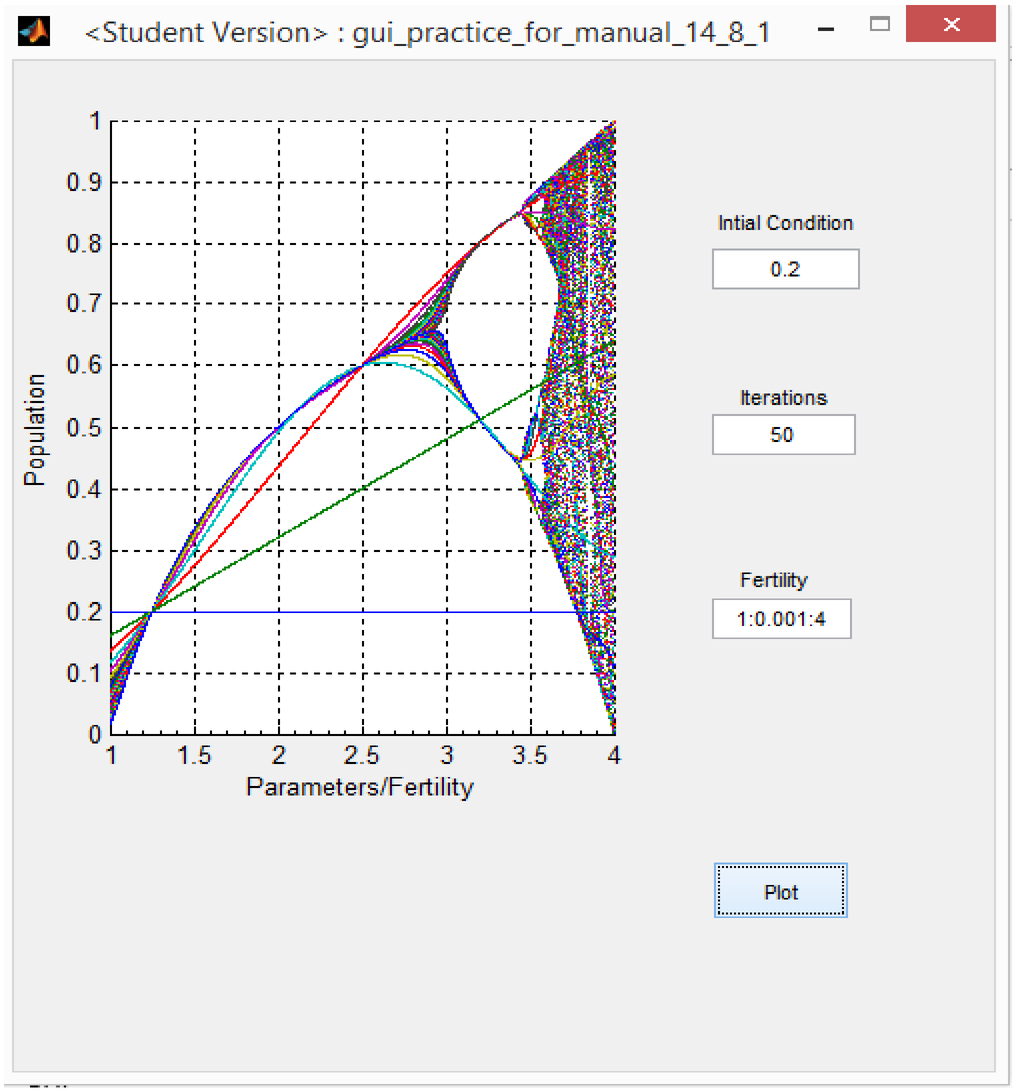


Intial Condition: 0.8

Iterations: 100

Fertility: 3:0.001:4

In comparing the two graphs, the blue line indicating the initial condition of the plot moves from 0.5 to 0.8 in response to the change in the initial conditions text box. Now let's change all three text boxes to check the response of the GUI.



Intial Condition: 0.2
Iterations: 50
Fertility: 1:0.001:4

The plot has responded accordingly to our new values and has removed the previous plots just displaying the current condition. Our GUI is now complete and ready to be used as an interactive tool for learning.