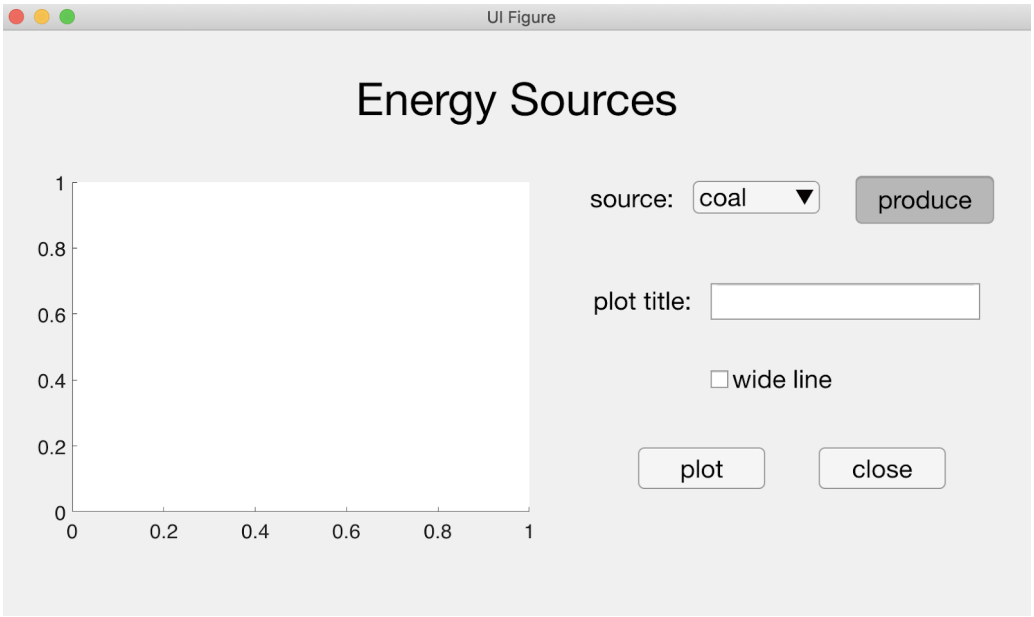
Video #1: Introduction to creation of interactive programs with MATLAB's App Designer

Until now our MATLAB programs included scripts and functions that were executed with minimal input from the user. We're now going to learn how to create more interactive programs that put the user in the driver's seat, enabling them to control what the program does through interactions with a graphical display that we refer to as a graphical user interface, or GUI for short. We'll develop these programs using a MATLAB tool called App Designer.

To illustrate the basic idea, and how to build a program of this sort from scratch, I'm going to use a simple example that draws on your experience with data about energy sources that were produced and consumed in the US, from an early assignment. The program is called energyApp, and I'll run it from my MATLAB Command Window.

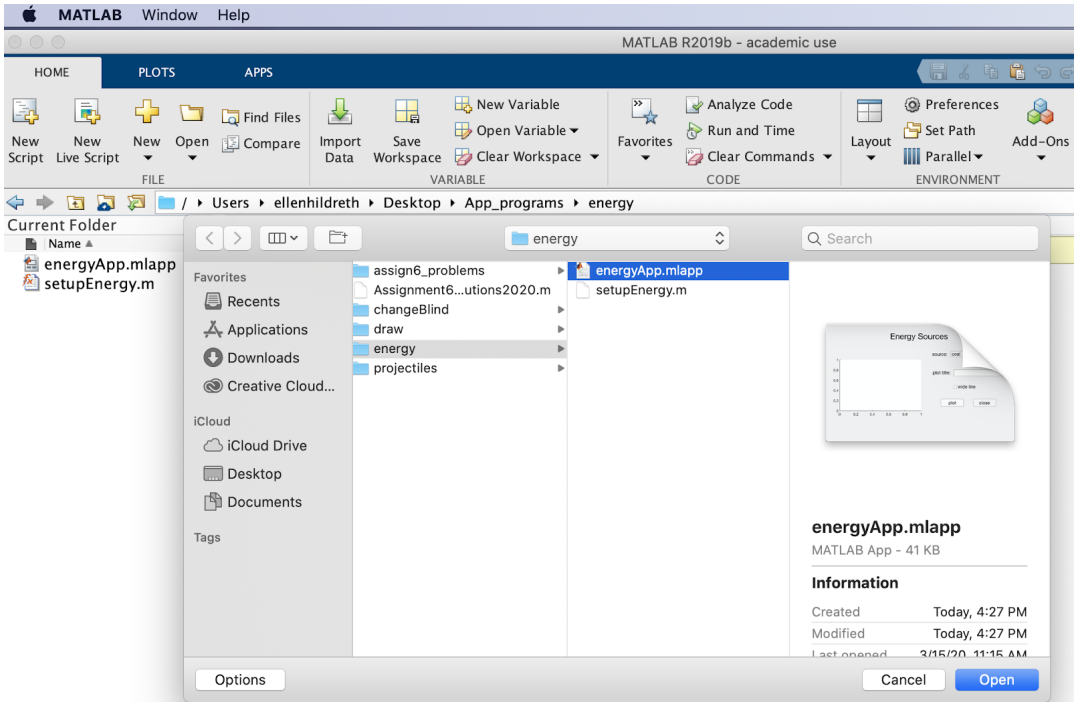
>> energyApp



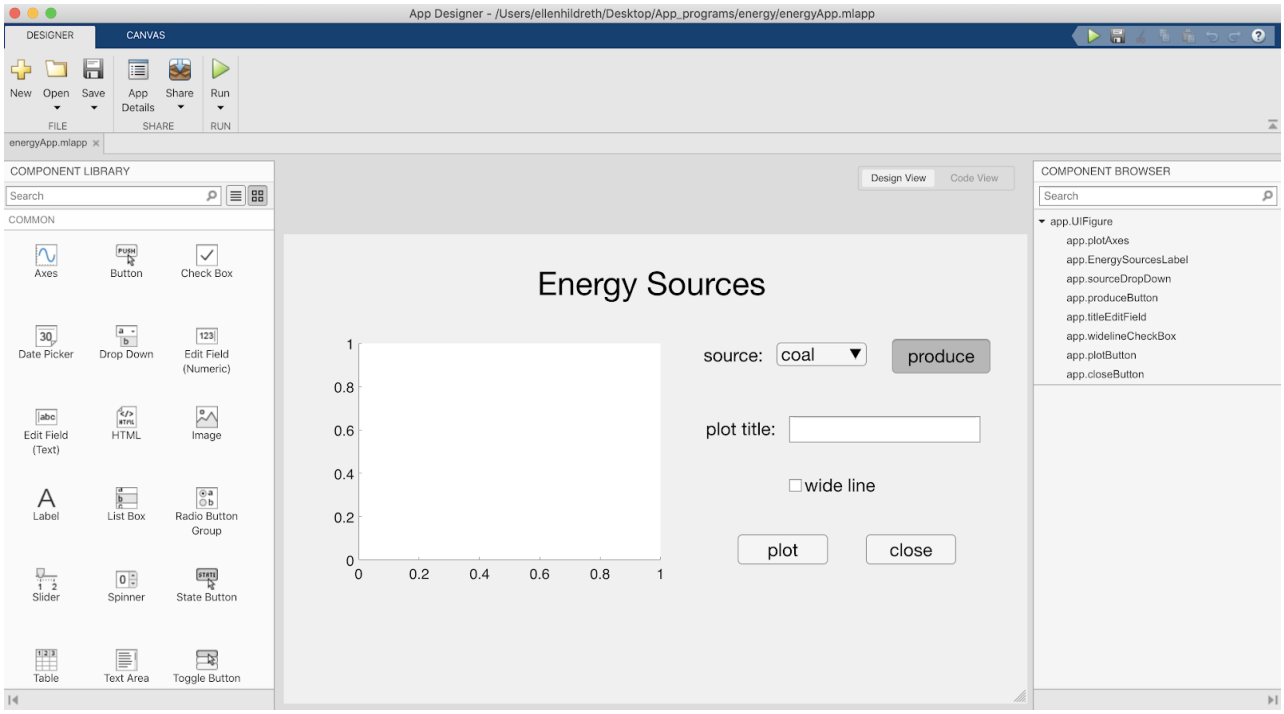
The program allows me to select what energy sources to view through simple components that I can interact with. There's a Drop Down menu where I can select an energy source from a set of options. I can use this button in the upper right to choose between production or consumption data. There's a box where I can enter a string that will be displayed as the title of my plot, and a checkbox that allows me to choose whether to plot the data with a wide line. When I've made my selections, I can click on the plot button to see a plot of the data I selected. I can change my choices and plot my new selection.

All this time, my program has been running - right now, it's just waiting for me to interact with the graphical display, and when I do, the program performs certain actions in response to my interactions, for example, it changes the text on the produce/consume button, or displays a new plot. When I'm done interacting with the program, I can click on the close button to terminate the program.

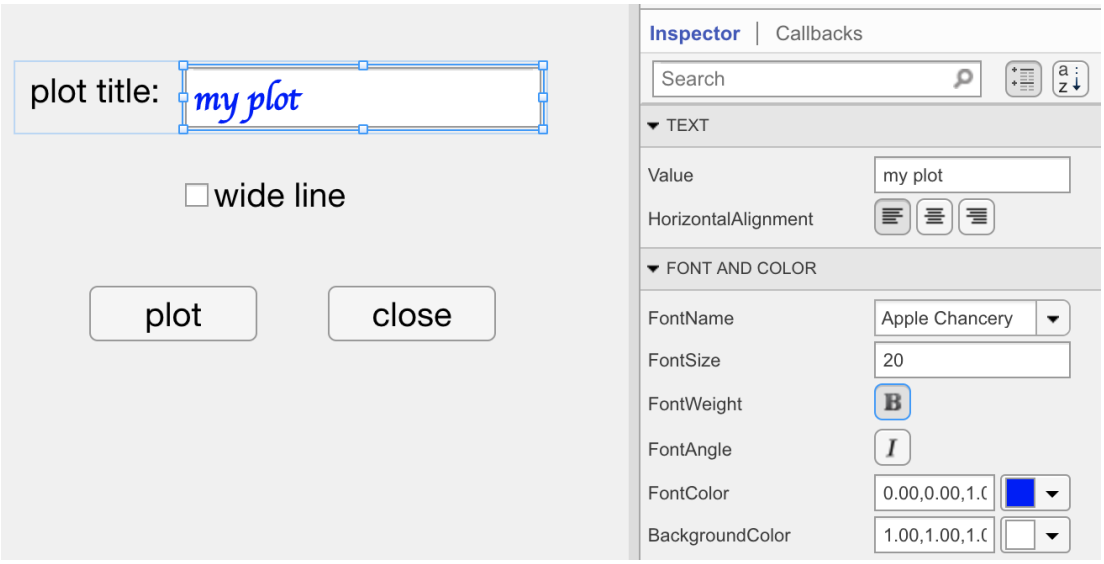
So how did I create this program? I used a MATLAB tool called App Designer. In the next two videos, I'll walk you through the process of creating this program from scratch, but here I'd just like to show you what the end product looks like in App Designer. Programs created with this tool are called "Apps" and they have a file name extension of .mlapp, as you can see in my Current Folder. To see the finished product, I'll go to the Open icon in my toolbar and navigate to the energyApp.mlapp file, and open it in App Designer.



The program has two parts that we can view, called the Design View (shown here) and the Code View. The Design View shows the visual layout of the various GUI components:



There's a palette of possible components on the left, and I added components to the graphical interface by dragging these icons to particular places on the canvas in the middle. So if I want to add a Slider, I can drag the Slider icon to a place on the canvas, and adjust its size. Each component has its own name that I can refer to in my program. For example, if I want to know what is the string that the user entered into this text box, I can refer to this component by its name, and access its value, which is the text the user entered. The names of all my components are shown in the COMPONENT BROWSER on the right. You can see from the compound structure of the name that the components are all stored in a structure named app. If I click on the name of one of the components, such as app.titleEditField, we see two tabs below the list of components, Inspector and Callbacks. Right now we're looking at the property inspector, or Inspector for short. With the Inspector, we can change the appearance of the GUI components. For example, suppose I want my textbox to have a string at the outset that will be my default title. I can enter this string for the Value property (e.g. my plot). Maybe I want to use a fancy font (Apple Chancery) with blue text in bold - I can modify these other properties:



I can save my program and then run it from App Designer by clicking on the green Run triangle in my toolbar - let's see how it looks with these changes I made.

Back to App Designer, the other view is the Code View. App Designer has its own editor, which is separate from the MATLAB editor you're familiar with. One thing that pops out immediately is that there's some code with gray background and other code with white background. The code with gray background is code that MATLAB generated automatically based on the visual layout that I created. If we peruse some parts of the code, we see a list of the names of all the GUI components at the top of the file, like app.titleEditField. Later, we see these components again with all the properties that affect their appearance, like font names, colors, the items in my menu of energy sources, and so on.

```
properties (Access = public)
    UIFigure                matlab.ui.Figure
    UIAxes                  matlab.ui.control.UIAxes
    sourcesLabel            matlab.ui.control.Label
    sourceDropDownLabel     matlab.ui.control.Label
    sourceDropDown          matlab.ui.control.DropDown
    widelineCheckBox        matlab.ui.control.CheckBox
    plotButton              matlab.ui.control.Button
    plottitleEditFieldLabel matlab.ui.control.Label
    plottitleEditField      matlab.ui.control.EditField
    closeButton             matlab.ui.control.Button
    produceButton           matlab.ui.control.StateButton
end
```

```
% Create sourceDropDown
app.sourceDropDown = uidropdown(app.UIFigure);
app.sourceDropDown.Items = {'coal', 'gas', 'oil', 'nuclear', 'renewable', ''};
app.sourceDropDown.FontSize = 20;
app.sourceDropDown.Position = [439 371 146 26];
app.sourceDropDown.Value = 'coal';
```

I added code to this file to implement actions that are unique to this program - this added code is shown with a white background. Let's go back to the running program for a moment. When the user interacts with components on the GUI, there are some actions that MATLAB handles automatically. For example, when I click on the Drop Down menu, MATLAB automatically shows me all the items, and if I make a different selection, MATLAB shows the new selection in the box. When I click on the checkbox, MATLAB turns the check mark on and off, and when I press my produce/consume button here, MATLAB changes the background color to light or dark. But there are other actions, like changing the word on this button or creating my plot, that are special to my program, so I need to write code for these actions.

Where did I write this code? Let's go back to Code View. We place new code inside functions that are referred to as callback functions. Each of these callback functions is associated with a particular GUI component, and gets called automatically when the user interacts with that component. In a later video, we'll see how to create callback functions from scratch, but here, let's just do some more perusing to look at a couple of them.

Whenever I clicked on the button that toggled between produce and consume, this callback function was executed automatically - it has an informative name produceButtonValueChanged, and the code inside captures the idea that depending on the state of the button, we'll change the text on the button to "produce" or "consume". See that the code refers to this button by name, app.produceButton, and this last piece of the name, Value, captures the current state of the button, whether it's on or off, and Text refers to the string of text printed on the button.

```
% Value changed function: produceButton
function produceButtonValueChanged(app, event)
    % change text on toggle button when user presses the button
    if app.produceButton.Value
        app.produceButton.Text = 'produce';
    else
        app.produceButton.Text = 'consume';
    end
end
```

Let's take a quick peak at the callback function for the plot button, named plotButtonPushed - this function is executed whenever the user clicks on the plot button. It determines whether the user wants to show production or consumption data by accessing the produce button, then sets up the right dataSource, then figures out which energy source the user selected in the Drop Down menu, and whether they want the plot to be shown with a wide line. The function then plots the data, adds labels to the plot, and finally copies the string I entered for the title, to the title of the plot.

```
% Button pushed function: plotButton
function plotButtonPushed(app, event)
    % set dataSource to production or consumption data, depending
    % on whether the user selects produce or consume
    if app.produceButton.Value
        dataSource = app.data.produce;
    else
        dataSource = app.data.consume;
    end
    % logical vector indicating which item in the menu is selected
    sourceIndex = find(strcmp(app.sourceDropDown.Items, app.sourceDropDown.Value));
    % use state of checkbox to determine line width
    if app.widelineCheckBox.Value
        linewidth = 4;
    else
        linewidth = 1;
    end
    % plot the data corresponding to requested properties
    plot(app.plotAxes, app.data.years, dataSource(sourceIndex, :), 'LineWidth', linewidth)
    % plot labels can also be created in advance in the Design View
    app.plotAxes.XLabel.String = 'years';
    app.plotAxes.YLabel.String = 'quadrillion btu';
    app.plotAxes.Title.String = app.titleEditField.Value;
end
```

We'll dig more deeply into this code in a later video, but first we'll take a closer look at how we create the visual layout of the graphical user interface in the Design View, in the next video.