

Lab 09: GUI

You should turn in:

1. A file named Lab09.m
2. A screenshot of you 'purchasing' an item from your vending machine.

Instructions:

First, make a function called lab09. It will not take in any input variables, nor return any output variables. Everything we write, except the callback functions, will be inside this function.

We will need to make a global variable - a variable we can use in multiple functions without having to pass it in manually. You can call it whatever you like, but in this example it will be called 'vendor'. To declare a global variable, all we need to do is write the following line of code:

```
global vendor;
```

After declaring our variable, we can then add information onto it. Our first attribute will be making it keep track of the funds a user has deposited into the vending machine. This will be referred to as 'money' in this example. To add an attribute to a variable, you will write the following:

```
vendor.money = 0.00;
```

This attaches a field to the vendor variable, which is now a structure. We can update vendor.money like any other variable we have encountered in the past.

Now that we've got the basics down, it's time to turn 'vendor' into a user interface.

We will define another field, in this case it is called 'fig'. This will make a pop up window that we can then place buttons and text on.

```
vendor.fig = figure('numbertitle', 'off', 'name', 'Vending Machine');
```

After the window is created, we can add words to it, for readability.

```
vendor.moneyDisplayMessage = uicontrol('style', 'text', 'units', 'normalized', 'position', [.034 .78 .09 .095], 'string', 'Money in machine:', 'horizontalalignment', 'right');
```

This specific line will add the words “Money in machine:” to the upper left hand section of the window. To make this more useful, we will display the dollar amount that the user has in the machine.

```
vendor.moneyDisplay = uicontrol('style', 'text', 'units', 'normalized', 'position',[.15 .78 .09 .05],  
'string', num2str(vendor.money), 'horizontalalignment', 'right');
```

Now, if we run this, it will show the number '0' next to our previously created text - if we change the previous line where we set the dollar amount, then this value will change, but we want the user to be able to change it themselves, so we will need to add more interface elements - a quarter slot, in this case.

```
vendor.quarterSlot = uicontrol('style', 'pushbutton', 'units', 'normalized', 'position',[.034 .007 .14  
.05], 'string', 'Add 25¢', 'callback', {@addMoney,.25});
```

The end of this line of code references a function we have not made yet, addMoney. We will get to it in just a bit, but for now we will continue to add elements to this window, for linearity sake.

Now we will be adding in a product someone may purchase. When making this attribute, we will denote the sub variable with a (1), like we would with an array. This will allow us to add in more later, under the same variable name, using different numbers.

```
vendor.products(1) = uicontrol('style', 'pushbutton', 'units', 'normalized', 'position',[.50 .76 .18  
.05], 'string', 'XXTRA Flamin Hot Cheetos', 'callback', {@vendProduct,1.5,1});
```

Again, this references another function, vendProduct, that we will write shortly. We will add in three more lines about our first product before writing additional functions

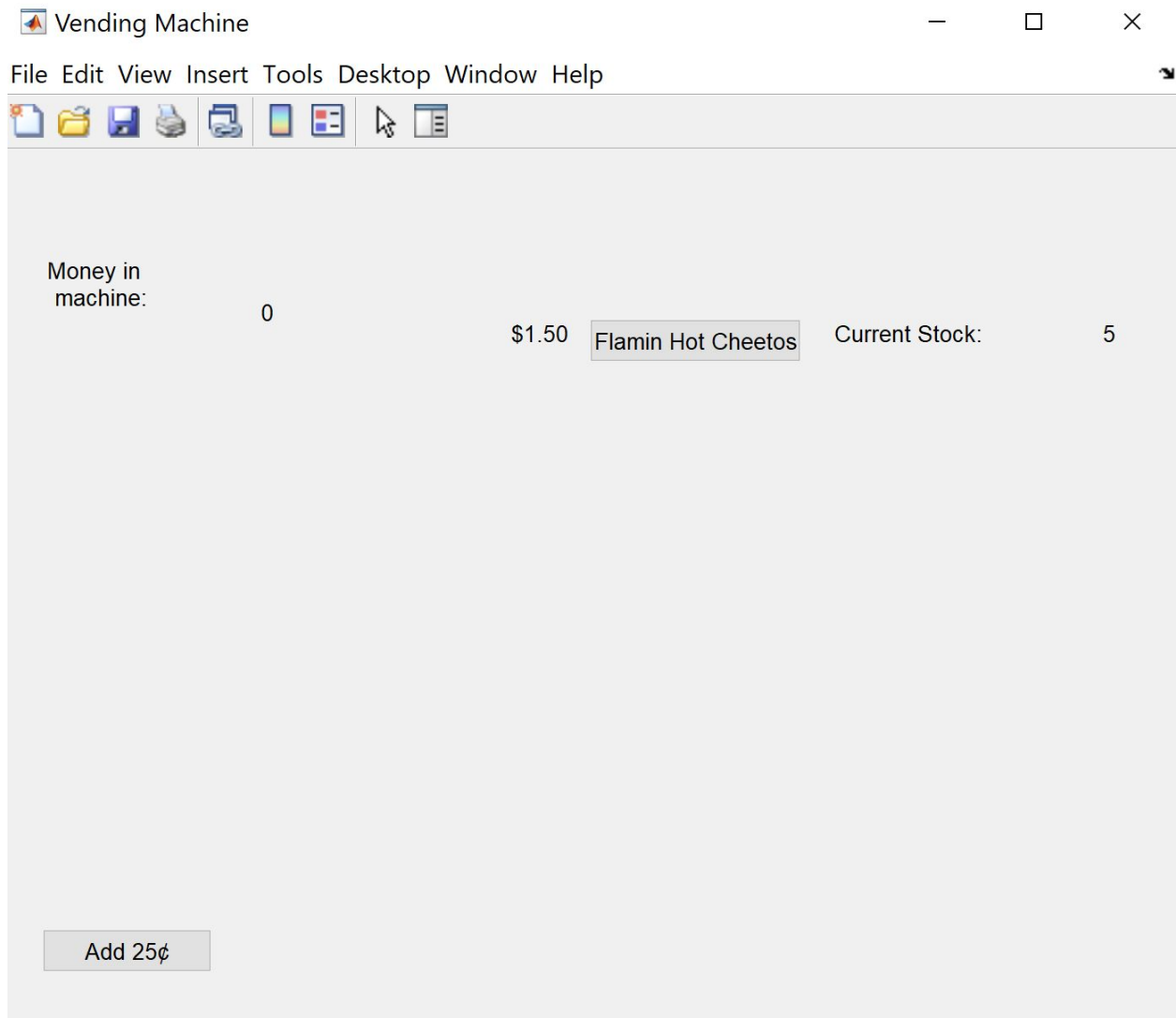
```
vendor.stock(1) = uicontrol('style', 'text', 'units', 'normalized', 'position', [.90 .75 .05 .05], 'string',  
'5', 'horizontalalignment', 'right');
```

This line indicates that we have five of the first product in our vending machine. We also need to show the user the price, and tell them what the above line means:

```
vendor.price(1) = uicontrol('style', 'text', 'units', 'normalized', 'position', [.42 .76 .05 .05], 'string',  
 '$1.50', 'horizontalalignment', 'right');
```

```
vendor.stockText(1) = uicontrol('style', 'text', 'units', 'normalized', 'position', [.70 .76 .15 .05],  
'string', 'Current Stock:', 'horizontalalignment', 'right');
```

With this, your user interface should look like the following:



[Image ID: a popup window labeled “Vending Machine.” Partway down the screen, starting from the left, are the following: text that reads “Money in machine:” followed by the number zero. To the right of that is the text “\$1.50” followed by a button labeled “Flamin Hot Cheetos”, followed by the text “Current Stock:” and the number 5 along the right side of the window. At the bottom left is a button labeled “add 25¢.” End Image ID]

From here, if you click any of the buttons, you will see errors in your command window. To remedy this, we will write our additional functions.

Our first is “addMoney”. This function will take in three elements - source, event, and the amount of money the user is putting into the machine, which will be referred to as ‘cash’, and it will not return any output variables. If you examine the line where we made our vendor.quarterSlot, you will see that the text within the curly brackets has only two elements inside of it. This is because the source is our vendor variable, and the event is a click - these two are automatically sent to

the callback function. We tell it which function to go to, and give it the last bit of information: .25 as our dollar amount to add to their funds.

In this function, you should add the amount of deposited funds to their current total number. This means updating both the vendor.money variable, as well as the vendor.moneyDisplay.String variables. Reference the previous lines of written code to determine how to do so.

The next function should be to vend a product. This function takes in source, event, price, and index as its four input variables. The index for our Cheetos is simply 1, but when we add in more products, the indices will be different.

In order to correctly vend a product, the user must have enough money to purchase the product, and there should be at least one of said product in the machine. If these two conditions are met, you will need to remove money from the user's funds equal to the price of the product, remove one product from its current stock, and tell the user that they were successful. For convenience, the lattermost task is provided for you below:

```
product = ['You receive a ', source.String];  
msgbox(product, 'Vending Machine Slot', 'modal');
```

This will create a popup window telling the user they received their item. Reminder to make sure to update both the actual numbers of money and stock, as well as the strings associated with them.

If the machine is out of the product, that is, if the stock of the item is 0, the code should tell the user that there is insufficient stock, and that they cannot buy the product. That line is the following:

```
msgbox('Insufficient Stock!', 'Vending Machine Error', 'error', 'modal')
```

A similar message box should appear if the user does not have enough money inserted into the machine, telling them they have insufficient funds.

Your final task for this lab will be to add in two more products of your choice, and add in a dollar slot to make adding funds more efficient.

With the additional products, be sure to place them on the screen such that they line up with the first product, but do not overlap. Give them prices that are different than the given product.

With the dollar slot, place it next to the quarter slot.

Once these additional tasks are done, you may turn in the lab.