CSP1150/CSP5110: Programming Principles

# Workshop 9: GUI Programming

**Task 1 – Main Window Settings**

We looked into setting options for widgets in the lecture to define things such as size, colour, position, padding and so on. What we didn’t really examine was the *main window* of the program.

The “Tk” class creates this, and it is the first line in the constructor when creating a GUI class:

|  |  |
| --- | --- |
| **import tkinter** | **Python** |
| **class ProgramGUI:** # define a ProgramGUI class  **def init (self):**  **self.main = tkinter.Tk()** # create the main window  **tkinter.mainloop()**  **gui = ProgramGUI()** # create a ProgramGUI object | |

While the “Tk” class creates the main window of an application, additional windows can be created using the “TopLevel” widget/class. The main window is a special type of “TopLevel”, and hence most of what you can do to configure a “TopLevel” can be applied to the main window.

So how do you do things such as setting the window title and size? That’s for you to find out! Do some research and see if you can figure out (and test) how to do the following:

* Set the main window title. *(Hint: See the lecture examples)*
* Set the main window size. *(Hint: Use the “geometry()” method)*
* Prevent the main window from being resized/maximised. *(Hint: Use the “resizable()” method)*
* Change the icon in the top left of the main window. *(Hint: Use the “iconbitmap()” method)*
  + *Find and download an icon file (.ico)* [*here,*](http://www.iconarchive.com/) *and place it in the same directory as your code.*

So far, all of these settings involve calling a method on our main window object (self.main). Hopefully your research informed you of the parameters you need to pass to the methods.

Some other aspects of the main window can be configured by using its “configure()” method, which works similar to setting options for widgets. Find out how to do the following:

* Set the background colour of the main window.
* Set the cursor to be used when the mouse is over the main window.

There’s another way of configuring the main window which isn’t properly implemented yet. It involves using the main window’s “attributes()” method. Since it isn’t properly implemented the settings must be set differently, e.g. instead of “attributes(alpha=0.5)” you must do “attributes('-alpha', 0.5)”. See the list of settings [here](http://effbot.org/tkinterbook/wm.htm#Tkinter.Wm.attributes-method) or [here](http://www.tcl.tk/man/tcl8.6/TkCmd/wm.htm#M8) and try to:

* Set the alpha of the main window to make it semi-transparent.
* Set the main window to be always on top of other windows.
* Set the main window to be full-screen. *(Press Alt+F4 to close it)*

# Task 2 – Working With Images

So far, none of our GUI programs have included images. Before we display an image in our program, find a “.gif” or “.png” image file and copy it into the same directory as your code. While it is possible to show images in other formats, more work is needed.

To display the image in your program, you first need to create a “PhotoImage” object (part of the “tkinter” module) and set the image’s filename as the “file” parameter in the constructor:

**self.logo = tkinter.PhotoImage(file='logo1.png')**

You can then refer to this object in other widgets that can show or otherwise use images. The simplest is a “Label”, which we have only used to display text until now. Instead of specifying the “text” setting when you create a label, specify the “image” setting and reference the object.



Using this knowledge and what we have covered in the lecture, create a program that shows an

image and a Button with “Toggle Image” written on it. Every click of the button toggles the image between two different image files. Here are some hints to help you:

* Change the “file” setting of the “PhotoImage” object whenever the button is clicked.
  + *This will cause the “Label” object to show the new image.*
* To read the value of a widget setting, use the “cget()” method on the widget object. This can be used to determine which file the “PhotoImage” object is displaying.
* Use the “configure()” method on the “PhotoImage” object to change the file setting.

# Task 3 – Radio Buttons and Check Buttons

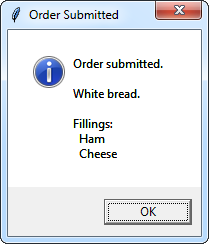
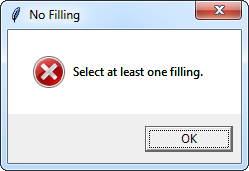
Radio buttons and checkboxes are quite commonly seen in a GUI. Radio buttons allow the user to select one option from a group, while checkboxes allow the user to select as many as they want. The “tkinter” module implements these via the “Radiobutton” and “Checkbutton” classes.

Section 13.8 of the textbook covers the usage of radio buttons and checkboxes, but you can also find tutorials and references online. Some of the key things to keep in mind regarding them are:

* The “variable” setting links a “Radiobutton” or a “Checkbutton” to a Tk variable.
  + *Commonly this is an “IntVar”, but it can be a “StringVar” and so on. Every “Radiobutton” in a group must link to the same variable, while each “Checkbutton” should link to a different variable.*
  + *You can “set()” the variable’s value to match the value of a “Radiobutton” or “Checkbutton” in order to specify a default radio button selection or checkboxes that start out already checked.*
* The “value” setting of a “Radiobutton” defines what the variable is set to when you select that option. Every button in the group should have a different “value”.
* The “offvalue” and “onvalue” settings of a “Checkbutton” define what the variable is set to when the box is unchecked or checked. The “onvalue” is usually the relevant one.
* To determine the value of a “Radiobutton” or if a “Checkbutton” is checked, read the value of the variable linked to it using the variable’s “get()” method.

Once you feel you have a grasp of how to use “Radiobutton” and “Checkbutton”, try to implement the program depicted and described below:

This program uses two labels, three radio buttons, three checkboxes and a button to create a GUI for specifying the details of a toasted sandwich. Some frames are also used for the layout. When the Submit Order button is pressed, one of two windows can appear:



The code in the event handler for the button is not particularly complicated, but be sure to spend some time thinking about it and planning it before you jump in and start trying to code it!

*That’s all for this workshop. If you haven’t completed the workshop or readings, find time to do so before next week’s class.*