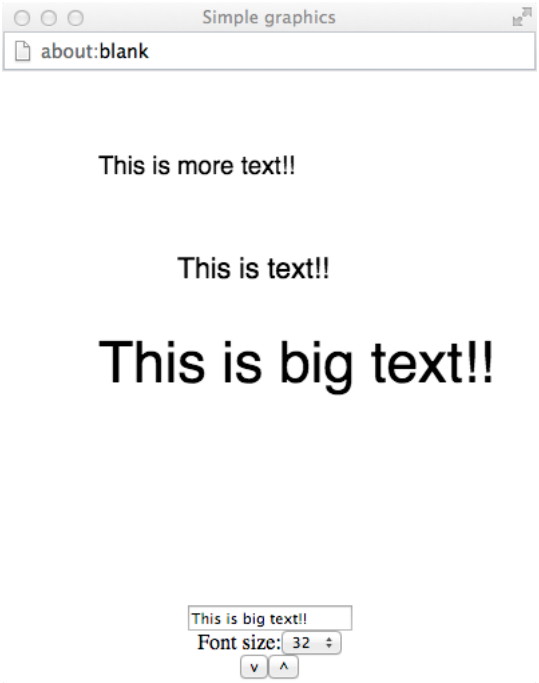
CS 051G Homework Laboratory # 6
GUI Practice

Objective: To gain experience using GUI components and listeners.

—

The Scenario. In this lab, we would like you to get practice in the use of GUI components. As a result we would like you to create a program which includes a TextField, a TextBox, a pop-up menu (Choice), and two Buttons. A picture of the screen can be seen below:



As usual, the center of the screen is the canvas. At the "South" end of the screen is a container that itself holds three containers that are stacked vertically. The top container holds the TextField. where the user can write some text that will be displayed on the canvas wherever the user clicks. Below this is a container that contains two components used to select a font size. On the left is a label (TextBox), while to its right is a pop-up menu that should contain the font sizes "9","12","16","24", and "32". (Note they must be entered as strings.) Below that is a container with two Buttons used to decrease or increase the font size by one for fine tuning the font size. The main container will be set to arrange the subcontainers vertically so that they are arranged on top of each other.

When the user clicks anywhere on the canvas, your program should display the text currently showing in the TextField at the place where the user clicked. When the user adjusts the font size by making a selection from the pop-up menu or clicking on one of the buttons, the last text placed on the canvas should change its sizeaccordingly. Changing the text in the TextField has no impact on items displayed on the canvas.

Start up code I have provided you with a type Sizable:

```
type Sizable = {
  fontSize:=(size: Number) -> Done
  fontSize -> Number
}
```

and an object emptyText:

```
def emptyText = object {
  method fontSize:=(size: Number) -> Done {
  }

  method fontSize -> Number { 0 }
}
```

The object emptyText is to be used as an initial value for the variable that you will use in your program to refer to the last text item put on the screen. It is there to protect against the case that the user changes the font size before clicking on the screen and placing a real text item there. Thus your program should include a variable declaration something like:

```
var currentText: Sizable := emptyText
```

How to Proceed Before beginning be sure that you have the document "Using GUI components in Grace" from the documentation page on the course web site up on your screen. As a quick reminder, remember the basic steps in displaying components:

1. Construct an instance of the component.
2. Add the component to a Container or to the graphic application.

To react to events generated by the user interacting with a GUI component, remember to make a method request to the component (e.g., myMenu.onChangeDo {... } to associate an action with the component. Don't forget to include the event parameter.

Here is a suggestion on how to decompose the development of this program into simpler steps.

1. Include code to create and add a Container at the bottom of the program window. Let's call it southEnd. Instruct the system that SouthEnd should be arranged vertically. Name and create the TextField and then append it to SouthEnd.

   Write the onMousePress method so that if the user clicks anywhere in the canvas then whatever is showing in the TextField is displayed as a Text item on the canvas. Make sure that successive clicks on the canvas insert new Text items on the screen (showing whatever is currently in the TextField).

   You will want the variable currentText associated with the Text item most recently displayed so that you can change its font size later. You will associate this name with the current text item in your onMousePress method.

   Since your program does not react immediately to the user typing in the TextField, it is not necessary to associate an action with this component.

   Now that you can display text items, you are ready to start working on adding the components that will let you modify them.

2. The next step is to create the Choice menu that will allow you to choose the font size in which the Text item is displayed. Because we want there to be a label (TextBox) next to it to help the user figure out what to do, we will create a container to hold those two items. See the instructions on using GUI components to see how create a new container. We'll refer to it here as row2.

Creating a TextBox is easy. See the instructions. Creating a pop-up menu (Choice) is nearly as easy. Don't forget to include all of the possible font sizes (but enter them as strings!) as arguments to the class construction.

You can now add the two items to row2 using the append method. You can then add the row2 to southBox. Test your program to make sure the GUI components show up where you want them to.

Now you can instruct the system to change the font size of the last text item created when the user selects a new font size from the Choice menu. Use the method onChangeDo on the menu to associate an action that will change the font size. Don't forget that the action needs an event parameter! (See the examples from class and the text.)

To change the font size, you can call the method

```
method fontSize:=(size: Number) -> Done
```

of the Text type to change the font size. Notice that this requires a Number, while the selected method of Choice returns a string. Luckily, the String type has a method asNumber that you can use to convert a string to a number as long as the String is the valid representation of a number, like e.g. 47 or 3.14159. Test this to make sure it works.

3. Next create the two buttons to increase and decrease font sizes. They will go in yet another new container, which can itself be appended to SouthBox. Associate actions with each of these to either increase or decrease the font size.

Be careful that your program does not crash if you manipulate the controls before adding the first text item to the canvas.

Extra Credit If you are interested in doing some extra credit work on this project, here is a suggestion.

Add the capability of changing the color of your text. There are two ways of doing this, one fairly simple, while the other is more complicated. We'll explain the simple one first.

1. Add a Choice menu with the names of colors (*e.g.*, black, red, blue, green, yellow). Associate an action to change the color of the text when the user uses the color menu. (You will likely need to add a method to the type Sizable to make this work.)

2. A more flexible way to control the text color would be to use a set of three number fields (with labels).

If you do either of these, please place the controls at the top of the screen as the bottom will be getting too crowded.

Another alternative is to include an extra button to erase the last text item drawn. (Note: after you have erased the last text item, just ignore further presses of this button or the other controls until the user clicks to create a new text item.)

Submitting Your Work When your work is complete you should deposit in the dropoff folder a folder with your grace program. Make sure the folder name is labeled in the form : lab6_lastnamefirstname Also make sure that your file includes a comment containing your name.

Also, before turning in your work, be sure to double check both its logical organization and your style of presentation. Make your code as clear as possible and include appropriate comments describing major sections of code and declarations.

This lab is due as usual on Tuesday night at 11 p.m., though I urge you to get it out of the way earlier so working on it doesn't get in the way of your test programs.

Table 1: Grading Guidelines

Value	Feature
	Code quality (5 pts total)
1 pts.	Descriptive comments
1 pts.	Good names and formatting
1 pts.	Good use of constants
1 pts.	Good use of private and local variables
1 pts.	Good use of private methods
	Correctness (5 pts total)
1 pt.	Displaying appropriate text at click point
2 pt.	Changing the font size using choice
1 pt.	Changing the font size using buttons
1 pt.	Avoiding errors before first text item drawn