Assignment #3b – Widgets!

Due: Friday April 26, 2002 by 5:00pm

In this assignment you will put the finishing touch on your WindowSystem. Now that you've got your windows showing up, applications can draw into them, and your windowmanager allows the user to move windows around and close them, you might ask what more does the system need? Consider the problem from an application developer's point of view: There's one more thing that modern developers have come to expect, and that is a usable widget set with which to create GUI elements easily.

Simple User Interface Toolkit (S-UITK)

The basic idea here is to create a set of widgets that can be used by a developer to easily include interface elements into their application's display. You will want to leverage inheritance as much as possible, so that each widget borrows as much as possible from its parent. You will need to implement the following widgets:

- Label
- Button

We recommend something like the following hierarchy for your S-UITK widgets:

- SWidget (abstract class)
  - SLabel
    - SButton

SWidget can be an abstract base class for your widget set. Thinking about the features that each of these have, the SLabel is simple – it features the ability to display some text with a colored background. The SButton extends this functionality with some basic event-handling. Your SButton class should allow other objects to register as event listeners, and whenever it is clicked, it should notify each registered listener. The canonical way to set up a listener-notification relationship is by creating an interface that all listeners must adhere to:

```
// SButtonClickListener.java
public interface SButtonClickListener {
      public void buttonClicked( … parameters you may need … );
}
```

Your SButton can keep a collection of SButtonClickListener objects, and notify each one when the time comes. The way that you let the SButton know about relevant mouse events is up to you. (i.e. notifying the button itself could be the job of the WindowManager, the Window, or the WindowSystem, depending on how you implement your widgets)

In order that you can render text on a SLabel or SButton, we have added the following methods to the GraphicsEventSystem:

- public final void drawString(String str, int x, int y)
- public final Font getFont()
- public final void setFont(Font font)

These work just as you would expect: drawString draws the given string at the given x and y coordinates using the current font and color. getFont and setFont will be useful to you in querying and assigning the system's font. You should download the new archive and use it with your project to gain access to these functions.

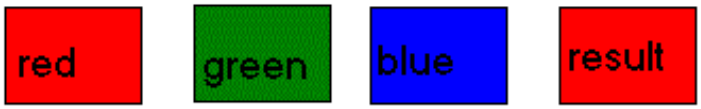
Widget Layout
We don't ask that you do any special layout managing for your widgets. A minimal layout scheme would be to add each widget to the right or below the previous one. This would just require the window to keep track of where the next free space is, which would be updated each time a new widget is placed. To make things even simpler, you could even require the application to inform the window exactly where a widget is to be placed. Either way is fine – or if you want to implement some other scheme, check it with us and we'll probably give it the go-ahead.

Extra Notes
You may have done this already in implementing your window manager, but it will probably come in handy to have a function in your WindowSystem which maps screen coordinates to windows, something like:

- public XWindow getWindowAt(int x, int y)

The logic for this function will depend on how you are keeping track of your windows.

Demo Application
To show off your widget set, please create a simple application that opens a window, and inserts four widgets into it. The application should be a color-chooser application, and it should look something like this:



The rightmost widget should be a label, and the leftmost should be buttons. Clicking on any of the leftmost 3 buttons should have the effect of changing the background color of the rightmost button to the color of the clicked-on button. This simple application will allow us to see that your widget set allows an application to create labels and buttons, and bind correct behavior to them.

Submission: As you did last time, make a directory in the class submission directory with your username (or your usernames, hyphenated together if you are working in a group). The submission directory for this assignment is at:

/afs/ir/class/cs377a/submit/assn3/

You can then FTP your files into that directory. Please archive your CodeWarrior project directory (use the DropStuff application on any public cluster macintosh) and submit the archive to us.