Messages

Now we're cookin' with Crisco, but let's press on and see what other things we can do to improve this. You may have noticed by now that the only way to quit the program is to close the window using the window manager's "close window" option, or to just kill the program outright. We can do better than that. Let's add a message handler for the `FXButton` such that when you click the button, it causes the program to exit:

```
require 'fox16'

include Fox

theApp = FXApp.new

theMainWindow = FXMainWindow.new(theApp, "Hello")
theButton = FXButton.new(theMainWindow, "Hello, World!")
theButton.connect(SEL_COMMAND) do |sender, selector, data|
  exit
end
theApp.create

theMainWindow.show

theApp.run
```

Most FOX objects send out messages (also known as *events*) when something interesting happens. FOX messages have four important elements:

1. The message *sender* is the object that sends the message. In this case, the `FXButton` instance is the sender.

2. The message *type* is a predefined integer constant that indicates what kind of event has occurred (i.e. why this message is being sent). In this case, the message type is `SEL_COMMAND`, which indicates that the command associated with this widget should be invoked.

3. The message *identifier* is another integer constant that is used to distinguish between different messages of the same type. For example, the message that tells a FOX window to make itself visible is a `SEL_COMMAND` message with the identifier `FXWindow::ID_SHOW` (where `ID_SHOW` is a constant defined in the `FXWindow` class). A different message identifier, `FXWindow::ID_HIDE`, tells an `FXWindow` instance to make itself invisible.

4. The message *data* is an object containing message-specific information. For this case (the `FXButton`'s `SEL_COMMAND` message, there is no interesting message data, but we'll see other

kinds of messages where the message data is useful.

For historical reasons, the message type and identifier are usually packed into a single 32-bit unsigned integer known as the *selector*, and this is the value that is passed into the message handler block. Since we don't actually need to use the *sender*, *selector* or *data* arguments for this particular message handler, we can just ignore them and shorten the code to:

```
theButton.connect(SEL_COMMAND) { exit }
```

Re-run the program and push the button to convince yourself that it works.

Prev	Up	Next
Better living through buttons	Home	Adding a tool tip