VIB dialogs

In addition to creating program interfaces, VIB can create dialog boxes. Here is a dialog for picking a color:

Use VIB as usual to create the interface and then click on the lower right corner to bring up the interface properties. Set the procedure name (color_dialog) and select "dialog window".
VIB dialogs, continued

Here is a simple program to exercise the dialog: (dlg1)

By default, a black circle is drawn in response to a click in the region. Pressing the Color button brings up the color selection dialog to specify the color of subsequent circles.
VIB dialogs, continued

The color dialog is handled entirely in the callback for the Color button:

```lisp
procedure color_cb(vidget, value)
    attribs := table()
    attribs["lightness"] := "medium"
    attribs["hue1"] := "black"

    if color_dialog(&window, attribs) == "Okay" then
        Fg(dwin, attribs["lightness"] || " " ||
            attribs["hue1"] || " " ||
            attribs["hue2"])
    return
end
```

The dialog is brought up by calling `color_dialog`, the procedure name specified in the interface properties, as shown on slide 95.

Two arguments are passed to `color_dialog`: the window to associate the dialog with, and a table specifying initial values for the widgets in the dialog.

When the dialog is dismissed, `color_dialog` returns the label of the button used to dismiss it.

In the above example, if the user pressed the Okay button, the color for `dwin` (a cloned window for the region) is set based on values in `attribs`, which in turn were set to the vidget values present when the dialog was dismissed.
VIB dialogs, continued

There can only be one VIB-generated interface in a source file, so each dialog must be in its own source file.

The color dialog was created with 'vib cdialog.icn'. Here is cdialog.icn, minus comments:

```
link dsetup
###<<<vib:begin>>>=== modify using vib;
procedure color_dialog(win, deftbl)
static dstate
initial dstate := dsetup(win,
    "color_dialog:Sizer::1:0,0,243,120:" [,],
    ["button2:Button:regular::152,86,49,20:Cancel" [,],
     ["hue1:Text::14:87,23,150,20:Hue #1\=" [,],
     ["hue2:Text::14:89,53,150,20:Hue #2\=" [,],
     ["lightness:Choice::5:8,9,72,110:" [,],
      ["pale","light","medium","dark","deep"]]},
    return dpopup(win, deftbl, dstate)
end
###<<<vib:end>>>===
```

The program was built with the command

```
icont dlgl.icn cdialog.icn
```

Note that VIB assumes that non-toggling buttons dismiss the dialog. If a button is specified (via VIB) as "dialog default", then hitting <Enter> will simulate a button press on it.
Utility dialogs

There are several other utility dialogs, such as Notice(). One is TextDialog, which presents a set of labeled text widgets.

This program:

```pascal
global dialog_value
procedure main() # dlg2
  rslt := TextDialog("What is your birthday?", 
             ["Month", "Day", "Year"], 
             [1,1,2000], 
             [2,2,4],  # field widths 
             ["Done","Cancel"])

  write(Image([rslt, dialog_value],3))
end
```

Produces this dialog:

When dismissed, the global dialog_value is set to the values of the fields and the label of the button pressed is returned.

The above Image might produce this:

```
["Done", ["7","4","1976"]]
```
Utility dialogs, continued

Other utility dialogs are:

- **SelectDialog**—displays a set of radio buttons

- **ToggleDialog**—displays a set of toggle buttons

- **ColorDialog**—allows selection of a color using the RGB or HSV color models

- **OpenDialog, SaveDialog**—simple front-ends to **TextDialog** to prompt for file names

Chapter 14 in the graphics text describes the utility dialogs in detail.