CALIFORNIA PATH PROGRAM
INSTITUTE OF TRANSPORTATION STUDIES
UNIVERSITY OF CALIFORNIA, BERKELEY

Custom Interface Builder Palettes for Advanced Driver Interface Rapid Prototyping

David W. Moore

UCB-ITS-PWP-93-19

This work was performed as part of the California PATH Program of the University of California, in cooperation with the State of California Business, transportation, and Housing Agency, Department of Transportation; and the United States Department Transportation, Federal Highway Administration.

NOVEMBER 1993

ISSN 1055-1417

Custom Interface Builder Palettes for Advanced Driver Interface Rapid Prototyping

A User's Manual

prepared by David W. Moore at the

Systems Integration Laboratory

Industrial Engineering and Operations Research

University of California, Berkeley Ca

Table of Contents

1.0 Introduction

This manual provides a brief introduction to four classes of software objects developed at the Systems Integration Laboratory in the department of Industrial Engineering and Operations Research at the University of California at Berkeley under a grant provided by Caltrans through the California PATH Program. The Knob, Dial, Alert Light, and Map object classes presented here represent the beginnings of a software library of virtual interface objects designed to support a rapid prototyping environment for advanced interfaces as described in "Rapid Prototyping of Advanced Driver Interface Systems", by Mendel et al (PATH Research Report). The compiled code that accompanies this manual may be loaded into NeXT Computer's Interface Builder application (NextStep version 2.0 running on a NeXT Computer is required.) as four separate "palettes". Once loaded into the Interface Builder environment, the objects composing the four palettes may be graphically manipulated and connected to other application kit software objects to produce working interfaces supporting various users tasks. These software objects may help decrease interface development time and cost by allowing new interface prototypes to be quickly programmed and evaluated in a computer graphics environment before actual physical prototypes are constructed.

The remainder of this manual briefly describes Interface Builder, the four object classes, and the ways by which a user may create and test interfaces using these objects in the Interface Builder application environment.

1.1 Interface Builder

Interface Builder (IB) is a creation of Next Computer designed to reduce the development time required by software engineers to produce custom applications. The IB application kit includes a hierarchy of existing software objects written in Objective-C. These objects are available for graphical manipulation and combination by the user on four distinct palettes provided with Interface Builder. Moreover, the functionality built into these objects may be utilized by developers creating custom software objects and applications by making the custom software objects subclasses of appropriate application kit objects.

Like the application kit palettes, the palettes created in the Systems Integration Lab may be loaded into Interface builder and graphically manipulated or used as superclasses when developing new custom objects. The techniques available to developers for subclassing custom software objects from existing object classes in Objective-C are described in the Interface Builder documentation or in any Objective-C book, and will not be further discussed in this manual. See **Object Oriented Design With Applications** by Grady Booch for a general discussion of object-oriented programming techniques, and **Objective - C, Object Oriented Programming Techniques** by Pinson and Wiener for an Objective - C - specific discussion of techniques.

In order to load the custom palettes into Interface Builder, select the "tools" category of the IB menu and then the "load palettes" category of the resulting submenu. The custom palettes are located on diskette in SILab/Palettes and may be selected for loading from there.

2.0 Palettes

Once the custom palettes have been loaded they will appear in the "palettes" window of the IB application. By selecting a particular palette's icon in the palette window, the objects in that palette appear in the lower portion of the palettes window (the "palette tray") and may be selected via the mouse and dragged into a new "application" (or "module") window. Once in a new application window, an object may be resized or otherwise graphically manipulated using the mouse/keyboard combinations described in the IB documentation. The inspector window provides a means of altering various characteristics of selected objects when such characteristics cannot be easily changed through a graphical means.

The "palette" and "inspector" windows of the IB application will be referred to throughout this manual - the palette window when a new instance of some object class is needed, and the inspector window when specifying connections between objects or when changing particular

object attributes. Objects selected from the palette tray will always be dragged into the application window and all future graphical manipulation of or connections to such objects will be initiated within the application window.

2.1 Knob Palette

The knob palette consists of three object types including a trackball, a knob, and a thumbwheel. Both vertical and horizontal thumbwheels appear in the palette window when the Knob palette is selected, though conceptually these objects are the same. In fact, by using the inspector the user may inspect the "attributes" of a thumbwheel and set the thumbwheel's orientation to either vertical or horizontal (Note that a thumbwheel must be graphically reshaped in the application window once its orientation has been modified in the attributes inspector window.). Several other characteristics of the thumbwheel may be set in the "attributes" mode of the inspector panel including the knurling pattern, speed, and speed limit of the currently selected thumbwheel. The best way to learn how changes in these parameters affect a thumbwheel or a trackball is to make the changes and then test the new configuration.

Open a "new application" window by selecting the "files" category of the IB menu and then selecting the appropriate submenu. Select the application kit palette that contains the "text" cell and use the mouse to drag a text cell object into the application window. Now copy the text cell and paste two more copies of it into the application window, or drag two more text cell objects into the application window from the palette tray. Select the Knob palette icon in the palette window and drag one of each type of knob into the application window (Choose only one of the thumbwheels, either the vertical or the horizontal one, since one orientation can easily be changed to the other via the attributes inspector.) Control-drag a connection from each type of knob to a distinct text cell and "connect" to the "takeFloatValueFrom" action of the selected text cell using the "connections" inspector panel. Now go to the "file" category of the IB menu and select "test interface".

The thumbwheel, knob, and trackball each respond to mouse inputs as the corresponding physical objects would respond to analogous inputs translated via physical contact. For instance, by placing the mouse arrow over the knob and pressing the right mouse button (the mouse button must remain depressed for the knob to remain selected), the knob may be turned from left to right or right to left as a physical knob could similarly be turned by human fingers. Try selecting the knob and turning it with the mouse. Notice that once the knob has been selected, the mouse arrow may be moved away from the knob while still maintaining control of the knob. In fact, by increasing the radius (the distance between the center of the selected knob and the mouse arrow) finer and finer increments of knob output (displayed in the distinct text cell that the knob is currently connected to) may be obtained from corresponding mouse movements along the circle centered at the knob's center and having the current radius. The incremental output change per degree change in knob rotational position may be set in the attributes inspector in the "speed" field. For a given setting of the speed field, the variable radius feature described above provides a means of fine tuning the knob's response to mouse inputs.

The knobs, like all the other palette objects described in this manual, may be connected to remote devices via serial port or bus by creating the proper software objects required to handle the operating system level process connections. The knobs may also be connected to any other software object containing a method analogous to "takeFloatValueFrom", such as the Dial objects described below.

2.2 Dial Palette

The Dial palette consists of a compass object, an artificial horizon object, and several gauge object types including fuel level, temperature, etc. Several characteristics of the interface of any gauge may be changed in the attributes inspector **(A** software tool for changing software object defaults, provided by Interface Builder), including the gauge's shape, background colors, title, and scale. **As** with the Knob palette objects, the best way to become familiar with the available

attribute configurations for the Dial palette objects is to inspect and change the attributes and then play with the newly configured objects in the test interface mode.

Select the Dial palette icon in the palette window, and then select a compass object, an artificial horizon object, and a dial object (only one type of dial needs to be selected, since the attributes inspector can then be used to change the configuration of the dial to the other available types).

Delete all of the knobs and text cells currently in the application window except for the thumbwheel (create two additional copies of the thumbwheel). Control drag a connection from each of the three thumbwheels now present in the application window to a distinct dial in the application window (e.g., connect one thumbwheel each to the compass, artificial horizon, and dial). Make each connection to the "takeFloatValueFrom" method in each dial type (use the connections inspector of Interface Builder to select this method), and then enter the "test interface" mode. Now manipulate the thumbwheels and observe how the various dial types respond. Try changing various attributes of each dial type and testing the result. You may also connect a thumbwheel to the "takePitchValue" method of the artificial horizon and observe the results.

All of the dial objects in the dial palette accept connections from other types of software objects, such as the sliders provided with the application kit. The dial objects may be connected to external devices via serial port or bus outlets by creating the necessary software objects required to handle the operating system calls, just as the objects in the Knob or Alert Light palettes may.

2.3 Alert Light Palette

Clear the application window of all objects, and select the alert light palette icon in the palettes window. The palette tray will display five round and five square alert lights, each of the five (round or square) lights is a different color. Select one of the lights and drag it into the application window. Now select a "button" from the appropriate application kit palette and drag

it into the application window as well.

Control drag a connection from the button to the alert light's "toggleBlink" method and proceed to the test interface mode. Clicking the button with the mouse will start the light blinking at a fixed rate (which can be modified in the attributes inspector), and clicking the button again will terminate the blinking. The alert light class contains methods allowing the blink frequency to be set by some other object (besides the attributes inspector), though these methods have not been implemented in the current version of the code. Try changing various attributes of the light via the inspector panel and retesting the interface to become familiar with the available configurations.

The alert lights may be connected to software objects designed to detect particular events or conditions in the physical world or the operating environment, which in turn generate blink messages to signal the user. The size of the lights can be easily changed using the standard mouse commands for resizing IB palette objects.

2.4 SmartMap Palette

Clear the application window of all objects and select the smartmap icon in the palettes window. Select a map from the palette tray and drag it into the application window. Notice that the palette tray contains a browser object as well, though the code for this object is still under development. This poses no problem, however, because the application kit includes a general browser object that is compatible with the smartmap.

Select the application kit palette icon corresponding to the browser class (the icon looks like a large text cell) and drag a browser object from the palette tray into the application window. Use the "alternate" button on the keyboard in conjunction with the mouse (see IB documentation) to extend the browser horizontally to the right until a second browser column appears. Now control drag a connection from the map to the browser, and select the word "browser" under "outlets of

source" in the connections inspector. With this connection established, select the map object in the application window and switch from the connections inspector to the attributes inspector. In the "File" field of the attributes inspector, enter the path SourceCode/MapPalette/default.map (This path corresponds to the location of the map called "default" in the directory structure included on your diskette. Several other map databases have been converted to the smartmap format, though these other maps are not provided on the diskette). Now enter the test interface mode.

The browser should now display an alphanumeric list in its left-most column. Selecting a particular letter or number in this browser column with the mouse causes the next column to the right in the browser to list all of the streets of the current map that begin with the selected letter or number. Choose a specific street name, and a third browser column appears containing all of the block numbers available for the selected map and street. Double clicking on a block number causes the map to display the actual map data for the selected block and surrounding area.

The mouse may now be used to manipulate the zoom factor and rotational orientation of the map. By placing the mouse arrow on a particular point of the map and double clicking the left mouse button, the zoom method is called and that point on the map becomes larger and more detailed. Double clicking the right mouse button reverses this zoom process. Placing the mouse arrow on some point of the map and holding the left mouse button down allows the map to be moved up and down or from side to side within its frame. **A** similar operation using the right mouse button allows the map to be rotated around its axis within its frame.

Exit the test interface mode and select the dial palette icon in the palettes window. Select a compass object from the palette tray and drag it into the application window. Control drag a connection from the map to the compass, and select the word "compass" beneath the "outlets of source" column of the connection inspector. Enter the test interface mode again, and use the mouse (with the right-most mouse button) to rotate the map as before. Notice that the compass now rotates with the map, displaying the current direction that corresponds to the top of the map.

The attributes inspector may be used to load different maps, set zoom factors, and enable the zoom and rotation methods. Try changing various attributes and retesting the interface to become familiar with the available options.

3.0 Remarks

These palettes represent the beginnings of a library of virtual interface objects that will be developed and used in the Advanced Interface Rapid Prototyping project at the Systems Integration Lab. The dial and knob objects provide specific modes of input and output that may be displayed in any number of ways on the screen. The current postscript code enables these palette objects to be displayed and manipulated in the forms discovered above, but new postscript code may be added at any time to the appropriate palette to add another interface to the general knob or dial functionality. The smartmap object has the potential to be accessed via voice commands, and the SoundKit objects provided by Next provide a basic set of objects to help accomplish this. New Hardware and low level software are currently being installed in the SIL to create a flexible interface for new hardware devices with the computing environment. Soon, interface development for several additional classes of hardware will be possible in the SIL, including cnc machines, robots, VCRs, and perhaps automobiles.