23
Help Facilities

This chapter addresses the general-usage Help facilities that apply to all XView packages. The help mechanism is available for the application programmer to register help information in response to user requests.

OPEN LOOK describes a help mechanism that enables the user to get help from anywhere in the user interface. The user requests help using the Help key on the keyboard, or, if one does not exist, the `F1` function key. When this key is used, the window under the pointer receives the `ACTION_HELP` event. In such cases, a *Help window* is displayed as shown in Figure 23-1.

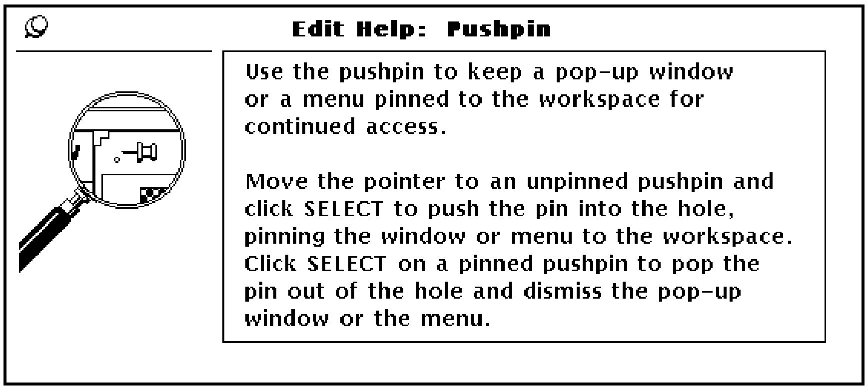

Figure 23-1. A sample Help window

Note that when you request help on a menu or a menu item, this action will dismiss the menu. This dismissal is required, since the Menu package must relinquish its pointer and keyboard grabs in order for the help request to be processed.

23.1 Using XV_HELP_DATA

The attribute XV_HELP_DATA is used to provide XView with information about where to find the text for the help frame. The attribute's value is a string in the form of:

```
"filename:token"
```

The *filename* is found from a list of directories set in the $HELPPATH environment variable. The actual filename has the *.info* suffix appended to its base name. Thus, the complete path to the filename that contains the help information is:

```
$HELPPATH/filename.info
```

If $HELPPATH is not set, the directory */usr/lib/help* is used. However, if $HELPPATH *is* set, then that path is searched exclusively and /usr/lib/help is *not* searched unless it is explicitly listed in the variable's setting. Therefore, users should always have the default path in their environment setting. Since $HELPPATH may be set to any number of directories, a possible setting would be (for the C shell):

```
setenv HELPPATH /usr/lib/help:/path/to/other/help/directories:.
```

Or, for the Bourne shell:

```
HELPPATH=/usr/lib/help:/path/to/other/help/directories:.
export HELPPATH
```

The trailing "." indicates that the current directory is included.

Once the filename is found, the *token* is searched for within the file. The file must contain the string ":*token*" at the beginning of the line. All the text following the token, starting on the next line, is displayed in the scrolling text subwindow within the help frame. A maximum of 10 lines of text, each terminated by a newline, is visible in the help text subwindow. If the help text exceeds 10 lines, a scrollbar is provided. Each line of help text may not exceed 50 characters. Scrollbars do not appear when a line is too long vertically to fit in the help window. You need to manually type a return in the help text to trim help text lines to fit within the 50-character limit. The text ends when a line is found that starts with a "#" or a ":".

23.2 HELP Key Binding

Servers that know about the XK_HELP keysym because they use a key labeled *Help*, as on Type-4 Sun keyboards, provide help messages when the HELP key is selected. On servers that don't know about XK_HELP, the user cannot get help unless some key is mapped to XK_HELP (OPEN LOOK requires that *some* key be mapped to provide help messages). The user, or system administrator, may define a help key by mapping the resource OpenWindows.KeyboardCommand.Help to a key such as F1.

OPEN LOOK also describes a method to access additional on-line information when a "More" button is placed in the help window. XView allows the More button to be selected with the mouse, or used with an accelerator (see Table 23-1). Selecting the More button allows the application to provide additional help according to the current help context.

Help for a text subwindow is also provided. Table 23-1 shows the default bindings for the various types of help functions provided by OPEN LOOK.

Table 23-1. Modified Help Keystrokes

Key	Type of Help
<Help>	Spot Help
Shift-Help	More Spot Help (an accelerator)
Ctrl-Help	Help on Text
Shift-Ctrl-Help	More Help on Text (an accelerator)

23.2.1 Attaching Help Data

Let's say you have written a program called `my_program`. In this program, there is a panel with a button labeled "Save." The user wants to know exactly what this button will do if selected. The user can select the Help key over the panel item in order to obtain help associated with the item. To support this action, you must attach help data to the panel items in the following manner:

```
 . . .
extern Frame frame;
extern void save_it();
Panel panel;

panel = (Panel)xv_create(frame, PANEL, NULL);

xv_create(panel, PANEL_BUTTON,
    XV_HELP_DATA,        "my_program:save",
    PANEL_LABEL_STRING,  "Save",
    PANEL_NOTIFY_PROC,   save_it,
    NULL);
xv_create(panel, PANEL_TEXT,
    XV_HELP_DATA,        "my_program:target",
    PANEL_LABEL_STRING,  "Filename: ",
    NULL);
 . . .
```

The value for `XV_HELP_DATA` contains the name of the program as the "filename" to search for and the token is `:save`. The filename was chosen based on the name of the application for easier administration of the help files. The next step is to create the file *my_program.info* and add the help text. The file might look like this:

```
:save
When selecting this panel item, all changes made
to the current document will be saved in the target
file.  To change the target file, type in a new
name at the "Filename:" text item.
#

:target
This filename represents the name of the file to use
to load a new file or to store editing changes.  Full
pathnames should be specified to insure that you get
the correct filename.
#
```

The “#” used to terminate help text makes the file more readable. Otherwise, it could be omitted, in which case the text would terminate at the next "*:token*" or the end of the file—whichever comes first.

23.2.2 More Help

The Spot Help window displays a More Help button when “More Help” is available. Alternatively, when Shift-Help is depressed, More Help is requested on the graphical object under the pointer, but the Spot Help window is not displayed. The help mechanism provides More Help when a field is added to the *.info* file, and More help is requested. The More Help field specifies a shell command that is executed to provide additional help information.

```
:spot_help_token:more_help_shell_command
```

The `more_help_shell_command` includes everything from the character after the second colon (:), to the character before the newline (\n). The More Help field should contain a string that is used as an argument for an invocation with the `system` library routine. This executes a command that provides help messages, or executes a help viewer (More Help). For example, an entry in a `.info` file containing a token and description for the topic `back`, and providing more help with the `helpopen` command, contains two fields as well as the Spot Help text.

```
:Back:$OPENWINHOME/bin/helpopen  handbooks/workspace.handbook

Choose Back to send the selected window to the
back of the window stack.
#
```

Prior to executing the system command, if the `-display` switch was specified on the command line, the help mechanism calls `putenv()` on the `DISPLAY` environment variable. This ensures that the help viewer, for example, `helpopen` above, is invoked on the display on which the More Help request was issued.

23.2.3 Text Help

Help information for a selected text string is presented using XView's text help feature and the HELP_STRING_FILENAME attribute. Text help allows XView to search for a string of text and to display help information that is stored in a help file associated with the string of text.

The Ctrl-HELP sequence produces an ACTION_TEXT_HELP event which begins the text help search. The window under the pointer will get an ACTION_HELP event and XView then requests the contents of the primary selection, a text string. XView opens the file associated with the window's HELP_STRING_FILENAME attribute and searches for a matching text string. If a match is found, the text's associated filename and target, which are stored in a second field in the HELP_STRING_FILENAME file, provide Spot Help, and/or More Help. If any condition is not met, then a notice is displayed.

The sequence Shift-Ctrl-Help produces the semantic action ACTION_MORE_TEXT_HELP for More help on a text string.

The HELP_STRING_FILENAME attribute contains the name of a file with a list of string pairs. The file is searched for in the directories listed in the environment variable HELPPATH. Each line in the file contains two words: the first word, which must be less than 128 characters, is the string on which help is available. The second word, which must be less than 64 characters, is of the form "file:target," which XView uses to find the Spot Help text.

HELP_STRING_FILENAME is set on the paint window, or on any of its owners where the help strings are painted.

23.2.4 Displaying Help Manually

If the window corresponds to an XView package that handles its own events, such as Panels or Menus, then XView displays the help frame automatically. All you need to provide is the data for the help message by using the attribute XV_HELP_DATA.

If you are handling events (such as a canvas) in the window, then you are responsible for displaying the help frame as well as providing the help data. The function to accomplish this is xv_help_show(). It takes the following form:

```
xv_help_show(window, help_data, event)
    Xv_Window       window;
    char           *help_data; /* "file:key" */
    Event          *event;
```

The window parameter is an XView window associated with an XView object. The help_data parameter is a string identical to the value of XV_HELP_DATA discussed in the previous section. The event parameter represents the event that caused the need for the help window to be displayed. This event structure may be modified upon return, so it should not be referenced after use.

Help Facilities

An event handler for a canvas would have to track help-key events, display the help frame, and provide the text to display in the frame. The following code fragments show how this might be done.

```
...
canvas = xv_create(frame, CANVAS,
    ...
    WIN_CONSUME_EVENTS, ..., ACTION_HELP, ..., NULL,
    WIN_EVENT_PROC, my_event_handler,
    NULL);
...

my_event_handler(window, event)
Xv_Window window;
Event *event;
{
    if (event_action(event) == ACTION_HELP) {
        xv_help_show(window, "canvas:help_info", event);
        return;
    }
    ...
}
```

The meaning of "`canvas:help_info`" is the same as the help data described earlier.

23.2.5 Help File Installation

Once the help file has been written, you should install it in */usr/lib/help* on your system. If you don't, then the user must set the `$HELPPATH` environment variable correctly to point to the path where the file actually resides. Otherwise, the user's request for help will result in a notice that help being posted cannot be found. Further, the file and the path to the file (including directories and links) must be readable and searchable.

If circumstances prevent you from installing the help file in the designated area, it is not reasonable to expect the user to know where the help file is. That is, do not expect that the user has set the `$HELPPATH` variable correctly. You should set the environment for the user. The path must be set to include at least two pathnames: */usr/lib/help* and the path to your help file. Both are needed because the user might request help from XView objects that provide their own help; these help files reside in the default directory and may be needed.

The following code fragment shows how `$HELPPATH` should be initialized to locate help files that do not reside in */usr/lib/help*.

```
#include <stdio.h> /* for BUFSIZ */

#define HELPPATHNAME "/help/directory" /* set this yourself */

main(argc, argv)
int argc;
char *argv[ ];
{
    extern char *getenv();
    char *helppath, buf[BUFSIZ];
```

```
    xv_init(XV_INIT_ARGC_PTR_ARGV, &argc, argv, NULL);

    /* ... */
    sprintf(buf, "HELPPATH=/usr/lib/help:%s:", HELPPATHNAME);
    if (helppath = getenv("HELPPATH"))
        strcat(buf, helppath);
    putenv(buf);

    /* ... */

    xv_main_loop(...);
}
```

Notice that we are setting the value of `$HELPPATH` regardless of its previous value. We don't know if the previous value of `$HELPPATH` existed; if it did, we don't know if the necessary pathnames were already in it. It is not worth the effort writing code to parse the string to see if the path exists. Even if it does, we want to be sure it is at the beginning of the variable. This ensures that the path is searched first. Since the same path can be set more than once, no harm is done by prefixing the variable with the desired pathnames.

Further, the use of `putenv()` requires that the `char *` buffer passed be valid throughout the life of the variable (e.g., until it is unset or reset later). In most cases, we would have to use `malloc()` and pass in new memory or use a static variable. However, since this `putenv()` is called from main, the variable space used by `buf` will not be corrupted until `main()` exits. So we don't bother with allocating memory.

Lastly, it should be mentioned that the effects of `putenv()` are temporary; the new path setting affects this process and application only. Each process has its own version of `HELPPATH`, so setting it explicitly for our application does not affect other programs.

23.2.5.1 HELPPATH usage with internationalization

The help mechanism checks the value of the attribute `XV_USE_LOCALE`. If this attribute is set to `TRUE`, the help files are searched for in locale specific directories, according to the setting of the attribute `XV_LC_DISPLAY_LANG`. If `XV_USE_LOCALE` is `TRUE`, `XV_LC_DISPLAY_LANG` is "C," and HELPATH is set to $APPHOME, then the application searches in the specified directory, as well as in the language specific directory for its *.info* files. In this case, the search path includes two directories:

```
$APPHOME/C/help
$APPHOME
```

23.2.5.2 Setting the application name

The attribute `XV_APP_NAME` sets the string to be used by XView as the application's name. (Currently this is only for the help package). The application name can be localized using `gettext` around the string that is set with `XV_APP_NAME`. XView will use the localized string set with `XV_APP_NAME` in the header of spot help windows to indicate which application generated the help window.

Help Facilities

This attribute allows for more than one help file while still displaying the same name in the spot help header window.

23.3 Help Package Summary

Table 23-2 lists the attributes and procedures for the HELP package. This information is described fully in the *XView Reference Manual*.

Table 23-2. Help Attributes and Procedures

Attributes	Procedures
XV_HELP_DATA HELP_STRING_FILENAME XV_APP_NAME	xv_help_show()