13. WHAT ARE MENUS?

While Medley provides a number of menus of its own (see Chapter 3), this section addresses the menus you wish to create. You will learn how to create a menu, display a menu, and define functions that make your menu useful. Menus are instances of records (see Chapter 24). There are 27 fields that determine the composition of every menu. Because Medley provides default values for most of these descriptive fields, you need to familiarize yourself with only a few that we describe in this section.

Two of these fields, the TITLE of your menu, and the ITEMS you wish it to contain, can be typed into the executive window as shown below:

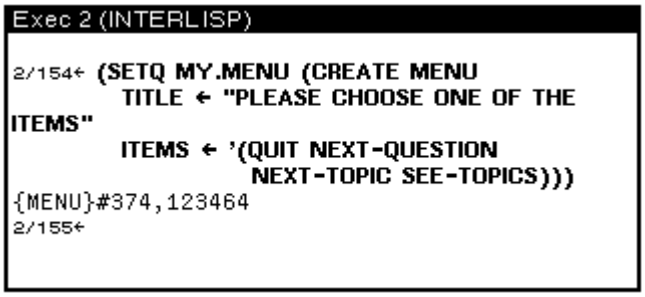

Figure 13-1. Creating a menu

Note that creating a menu does not display it. MY.MENU is set to an instance of a menu record that specifies how the menu will look, but the menu is not displayed.

Displaying Menus

Typing either the MENU or ADDNENU functions will display your menu on the screen. MENU implements pop-up menus, like the Background Menu or the Window Menu. ADDMENU puts menus into a semi-permanent window on the screen, and lets you select items from it.

(MENU MENU POSITION) pops up a menu at a particular position on the screen.

Type:

```
(MENU MY.MENU NIL)
```

to position the menu at the end of the mouse cursor. Note that the POSITION argument is NIL. In order to go on, you must either choose an item, or move outside the menu window and press a mouse button. When you do either, the menu will disappear. If you choose an item, then want to choose another, the menu must be redisplayed.

(ADDMENU *menu window position*) positions a permanent menu on the screen, or in an existing window.

Type:

```
(ADDMENU MY.MENU)
```

to display the menu as shown in Figure 13-2. This menu will remain active, (will stay on the screen) without stopping all the other processes. Because ADDMENU can display a menu without stopping all other processes, it is very popular in users programs.

If window is specified, the menu is displayed in that window. If window is not specified, a window the correct size for the menu is created, and the menu is displayed in that window.

If position is not specified, the menu appears at the current position of the mouse cursor.

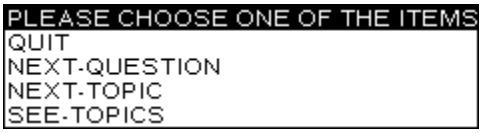

Figure 13-2. Simple MenuDisplayed with ADDMENU

Getting Menus to Do Stuff

One way to make a menu do things is to specify more about the menu items. Instead of items simply being the strings or atoms that will appear in the menu, items can be lists, each list with three elements (see Figure 13-3). The first element of each list is what will appear in the menu; the second expression is what is evaluated, and the results of the evaluation returned, when the item is selected; and the third expression is the expression that should be printed in the Prompt window when a mouse button is held down while the mouse is pointing to that menu item. This third item should be thought of as help text for the user. If the third element of the list is NIL, the system responds with **Will select this item when you release the button**.

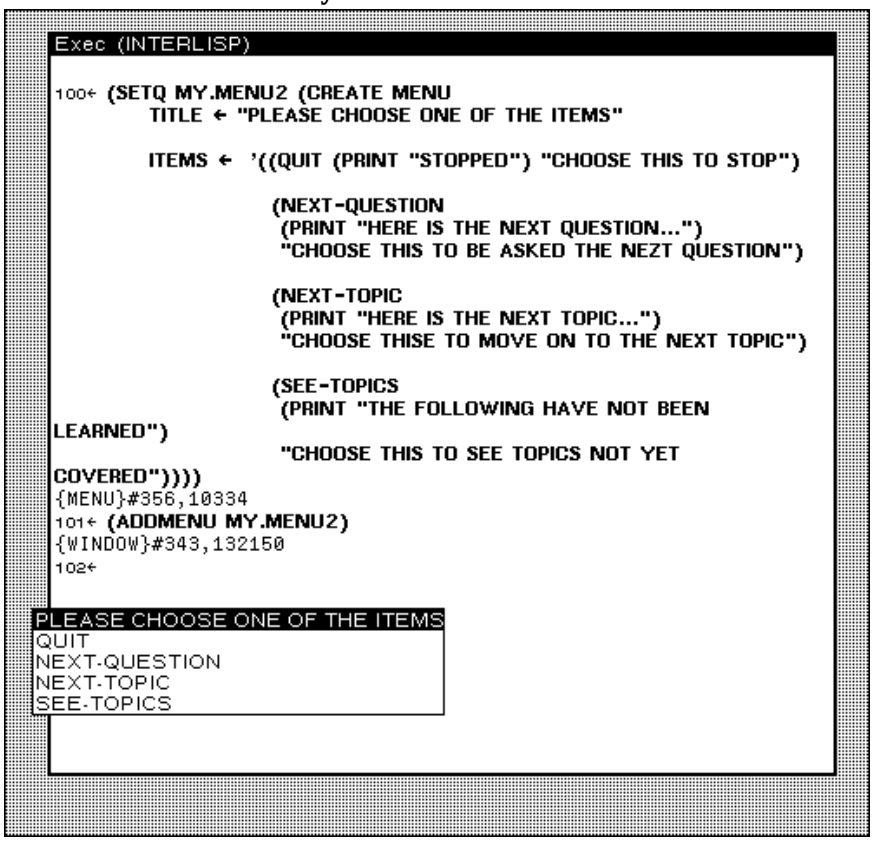

Figure 13-3. Creating a Menu to do Things, then displaying it with the function ADDMENU

Now when an item is selected from MY.MENU2, something will happen. When a mouse button is held down, the expression typed as the third element in the item's specification will be printed in the Prompt window. (See Figure 13-4.)

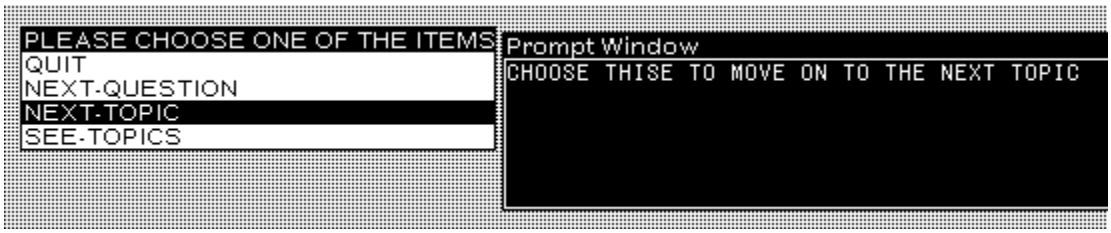

Figure 13-4. Mouse Button Held Down While Mouse Cursor SeIects NEXT.QUESTION

When the mouse button is released (i.e., the item is selected) the expression that was typed as the second element of the item's specification will be run. (See Figure 13-5.)

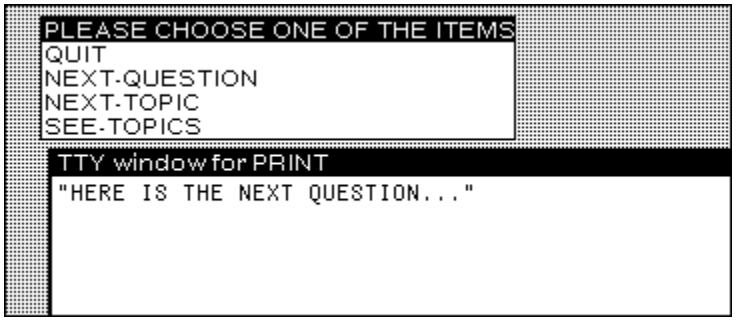

Figure 13-5. NEXT-QUESTION Selected

WHENHELDFN and WHENSELECTEDFN Fields of a Menu

Another way to get a menu to do things is to define functions, and make them the values of the menu's WHENHELDFN and WHENSELECTEDFN fields. As the value of the WHENHELDFN field of a menu, the function you defined will be executed when you press and hold a mouse button inside the menu. As the value of the WHENSELECTEDFN field of a menu, the function you defined will be executed when you choose a menu item. This example has the same functionality as the previous example, where each menu item was entered as a list of three items.

As an example, type in these two functions so that they can be executed when the menu is created and displayed:

```
(DEFINEQ (MY.MENU3.WHENHELD (ITEM.SELECTED MENU.FROM
BUTTON.PRESSED)
(SELECTQ ITEM.SELECTED
      (QUIT (PROMPTPRINT "CHOOSE THIS TO STOP")
      (NEXT-QUESTION (PROMPTPRINT "CHOOSE THIS TO BE ASKED THE

            NEXT QUESTION"))
      (NEXT-TOPIC (PROMPTPRINT "CHOOSE THIS TO MOVE ON TO THE
            NEXT SUBJECT"))
      (SEE-TOPICS (PROMPTPRINT "CHOOSE THIS TO SEE THE TOPICS
            NOT YET LEARNED"))
      (ERROR (PROMPTPRINT "NO MATCH FOUND"))))


(DEFINEQ (MY.MENU3.WHENSELECTED (ITEM.SELECTED MENU.FROM
BUTTON.PRESSED)
(SELECTQ ITEM.SELECTED
      (QUIT (PRINT "STOPPED")
      (NEXT-QUESTION (PRINT "HERE IS THE NEXT QUESTION"))
      (NEXT-TOPIC (PRINT "HERE IS THE NEXT SUBJECT"))
      (SEE-TOPICS (PRINT "THE FOLLOWING HAVE NOT BEEN
            LEARNED . . ."))
      (ERROR (PROMPTPRINT "NO MATCH FOUND"))))
```

Now, to create the menu, type:

```
(SETQ MY.MENU3 (CREATE MENU
    TITLE ← "PLEASE CHOOSE ONE OF THE ITEMS"
    ITEMS ← '(QUIT NEXT-QUESTION NEXT-TOPIC SEE-TOPICS)
    WHENHELDFN ← (FUNCTION MY.MENU3.WHENHELD)
    WHENSELECTEDFN ← (FUNCTION MY.MENU3.WHENSELECTED)))
```

To see your menu work, type

```
(ADDMENU MY.MENU3)
```

Now, due to executing the WHENHELDFN function, holding down any mouse button while pointing to a menu item will display an explanation of the item in the prompt window. The screen will once again look like Figure 13-4 when the mouse button is held when the mouse cursor is pointing to the item NEXT-TOPIC.

Now due to executing the WHENSELECTEDFN function, releasing the mouse button to select an item will cause the proper actions for that item to be taken. The screen will once again look like Figure 13-5 when the item NEXT-TOPIC is selected. The crucial thing to note is that the functions you defined for WHENHELDFN and WHENSELECTEDFN are automatically given the following arguments:

1. The item that was sølected, ITEM.SELECTED
2. The menu it was selected from, MENU.FROM
3. The mousø button that was pressed BUTTON PRESSED

These functions, MY.MENU3.WHENHELD and MY.MENU3.WHENSELECTED, were quoted using FUNCTION instead of QUOTE both for program readability and so that the compiler can produce faster code when the program is compiled. It is good style to quote functions in Lisp by using the function FUNCTION instead of QUOTE.

Looking at a Menu's Fields

INSPECT is a function that displays a list of the fields of a menu, and their values. Figure 13-6 shows the various fields of MY.MENU3 when the function (INSPECT MY.MENU3) was called. Notice the values that were assigned by the examples, and all the defaults.

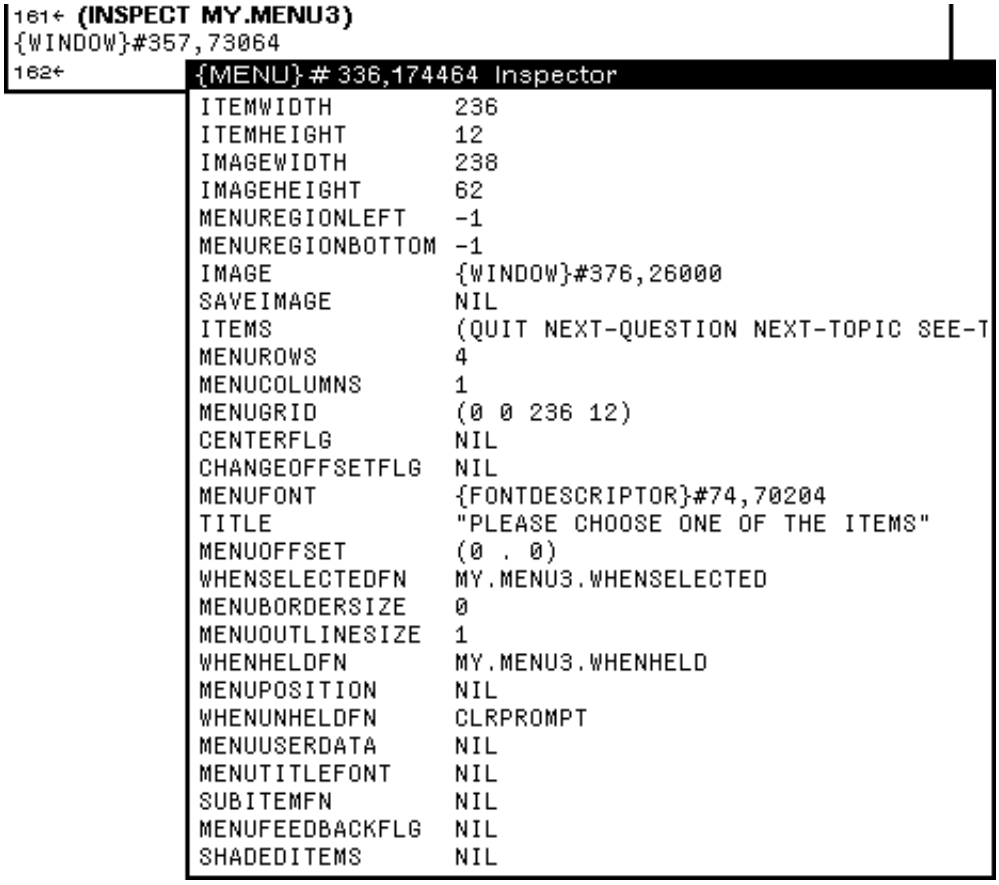

Figure 13-6. MY.MENU3 Fields