Filter List using the Keyboard

CA Gen Tech Tip - by John Carter

Introduction

In many applications, it is possible to navigate through a large list of items by using the keyboard to indicate which items in the list should be displayed or where the list should be positioned. For example, in a list with text as the key for the filter, pressing a single text character would either reposition the list so that the first row in the list that begins with the selected character would be positioned at the top of the list or only those rows that do begin with that character are shown in the list. This technical tip will detail a method of providing the second functionality for CA Gen list boxes. (Note that this capability works best for dialogs that have one list and no entry fields.)

Method Description

The basic idea is to create a hidden entry field on the window that will be used to attach a keypress event to. This field will always have focus on the window even though it is not displayed to the end user. When a key is pressed, the keypress event is called and the key is copied to a local view which saves the current set of pressed characters. Then, the list is scanned for all rows that match the text in the local view. Any rows that do not match will have their selection indicator set to hidden where all others will have the selection indicator set to spaces. For this particular method, pressing the space key clears the filter and displays all rows.

Implementation Details

The key to the method is the code in the keypress event. It will look something like the following:

```
 1 ┌─ EVENT ACTION navigate_keypress
 2 │  NOTE ************************************************************
 3 │       FILTER OBJECT LIST KEYPRESS EVENT.  When a character is typed
 4 │       in the hidden navigate field, the object list will be modified
 5 │       to show only those items that begin with the input character.
 6 │       If a space is entered, the full list will be displayed.  Non-
 7 │       alphanumeric characters will be ignored.
 8 │       ************************************************************
 9 │  SET export keypress keypress TO "reject"
10 │  SET export keypress character TO upper(export keypress character)
11 │  ┌─ IF verify(export keypress character, "ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789")
12 │  │                          IS NOT EQUAL TO 0
13<──┼──ESCAPE
14 │  └──
15 │  ┌─ IF export keypress character IS EQUAL TO SPACES
16 │  │  ┌─ FOR SUBSCRIPT OF export_group_of_objects  FROM 1  TO LAST OF
17 │  │  │                          export_group_of_objects  BY 1
18 │  │  │  SET export_grp ief_supplied select_char TO " "
19 │  │  └──
20 │  │  SET lcl work filter TO SPACES
21<──┼──ESCAPE
22 │  └──
23 │  SET lcl work filter TO concat(trim(lcl work filter), export keypress character)
24 │  SET export work filter TO lcl work filter
25 │
26 │  NOTE ************************************************************
```

```
27 |       Remove any rows in the object list for which the mnemonic
28 |       does not start with the text given in the navigate field
29 |       ************************************************************
30 |   ┌─ FOR SUBSCRIPT OF import_group_of_objects  FROM 1  TO LAST OF
31 |   |                         import_group_of_objects  BY 1
32 |   |  SET SUBSCRIPT OF export_group_of_objects  TO SUBSCRIPT OF
33 |   |                               import_group_of_objects
34 |   |  ┌─ IF substr(import_grp object short_name, 1, length(trim(upper(export work
35 |   |  |                             filter)))) IS NOT EQUAL TO trim(export work filter)
36 |   |  |  SET export_grp ief_supplied select_char TO "H"
37 |   |  ├─ ELSE
38 |   |  |  SET export_grp ief_supplied select_char TO " "
39 |   |  └──
40 |   └──
41 └──
```

The event starts in line 9 by setting the keypress return code to “reject”. This is done because the event logic will populate the export view explicitly. Next, the character is converted to upper case in order to make comparisons easier. The IF statement that comes next ensures that the selected character is one of the ones that we expect to use for the filter.

The IF statement at line 15 and the code following clears the filter if the entered character is spaces. In line 20, the local filter string is cleared before returning to the user.

In line 23, the local filter string is updated by concatenating the selected character to the string. Then the export version of the filter string is updated with the value of the local filter. Note that the local view is defined so that it is not initialized on entry so that its value is maintained between executions of the procedure step. The export view is mapped to an input view so that it is also saved on the window.

Starting in line 30, the import group view is scanned for matching values. The subscript of the export group is set to be the same value as the subscript of the input group. The IF statement in line 34 compares the data in the list (which, for this application, is already upper-cased) with the value in the filter. If the values do not match, the row is hidden by setting the selection indicator character to “H”, otherwise the selection indicator character is set to a space. CA Gen will use this information to determine which rows are displayed and which rows are hidden.

One other thing that must be done is that whenever a pushbutton or menu item is selected, the cursor focus must be returned to the hidden field. This is done in the events for the menu items or push buttons and also in a window event called Activated. At the end of each event, a statement like the following is added to reset the focus for the hidden field:

```
SET <WINDOW>.<field-name>.Focus TO “True”
```

Note that a similar technique may be used with a visible filter field. Using a visible filter is necessary when the dialog has other entry fields as well as the list box. In this case, the user is responsible for setting the focus on the filter field.

Hopefully this technique will prove useful to you and will help to make your applications easier to use.