Implementing a Multiwindow Choice List in Version 3

by Lincoln Stoller, Ph.D.
10/7/92

4th Dimension version 3 adds new commands whose full powers are not immediatly obvious. Version 3's most powerful feature is its ability to create multiple processes that enable us to support multiple nonmodal windows, the kind that come to the foreground and become enterable when you click on them. This article shows you how to use some of these new tools to present a list of choices in a second, nonmodal window. The new features we'll use include multiple processes, interprocess variables, the procedures `Call Process`, `Bring to Front` and `Process Attributes` and the functions `Outside call`, `New process` and `Current process`.

A choice list is a display of alternatives that you can use when you're assigning values. In version 2 of 4D you either displayed the choices in a modal window, a window that disables background windows until it's closed, or you displayed the choice list within the entry window itself. With version 3 we can open a second window and the user can work with the windows in any order they please.

I'll begin with the file structure in figure 1 consisting of a Baby file and a Names file, in honor of the new daughter of Thom Hickey, ACIUS's Manager of Technical Support. We'll enter a new record in the Baby file and use the Names file for a list of possible first names.
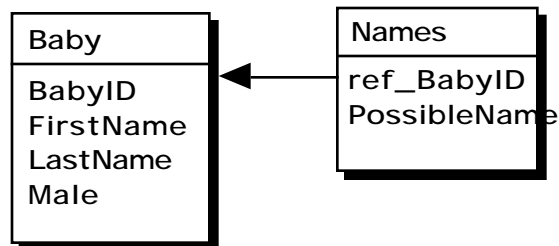
Figure #1: the file structure.

The input layout for a Baby record, shown in figure 2, has fields for first and last names and whether the baby is a boy or girl. In addition to an Accept and Cancel button the third button, labeled Names List, opens another window showing names from the Names file. The user will highlight a name, press a button and the program will copy the name to the FirstName field.

Both windows will be nonmodal and the list window will remain open after the user selects a name. Finally, when the input layout is accepted or canceled both windows will close.

The key to supporting two nonmodal windows is to open each one in a different process. 4D automatically starts a process for the Runtime/User environment and we'll manually start a second process for the list window when the user presses the Names List button. Everything begins with the input layout procedure:

```
                              Baby Record
                First Name  FirstName
                Last Name   LastName
                          ○ Girl      ○ Boy
                 ▶Names List    Accept     Cancel
```

```
  Case of                `Layout Procedure of [Baby];"Baby_Input" layout.
   : (Before)
     BabyPrcesNm=Current process
     SelectRN:=-1  `Assign value not held by any record indicating nothing chosen yet.
    ListPrcesNm:=-1 Assign value not held by any process indicating list isn't displayed.
    ListState:=-1   `Assign a value indicating the process isn't running.

   : (Outside call)
   If ( SelectRN>=0)                 `If a record was selected then assign it.
     GOTO RECORD([Names];SelectRN) `This process has it's own selection of [Names].
      [Baby]FirstName:=[Names]PossibleName
      SelectRN:=-1
    End if

   :(During)
   If ((bAccept=1) | (bCancel=1))
    CALL PROCESS(ListPrcesNm)   `Activate the other process if it's still running.
    End if
  End case
```

**Figure #2: the input layout in Design
environment and its procedure.**

As soon as we enter the input layout in the Before phase we define two interprocess variables, these are variables that can be shared between the two processes and are identified by having ' ' (option-shift-v) as their leading character.   BabyPrcesNm stores the number of the current process and   SelectRN will be used to store a [Names] record number.

The Outside call phase is new to version 3 and is only executed when this process is called from another process. It will be called from the choice list process after that process has been opened.

The During phase triggers the Outside call phase of the list window process, if that process is running, when the user presses the Accept or Cancel button.

After the Before phase runs the user sees the layout shown in figure 1 and can press the Names List button to fire the following script:

```
 ▶Names List 
```
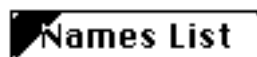
```
  `Script for 'Names List' button of [Baby];"Baby_Input" layout.
  PROCESS ATTRIBUTES(ListPrcesNm;ListName;ListState;ListTime)

  If (ListState<0)  `The process has not been created yet, create it now.
    SelectRN:=-1
```

```
   ListStateNew process("ChoiceList";32000;"NamesList")

 Else               `The list is already displayed in a separate process.
  BRING TO FRONT(ListPrcesNm)
 End if
```

Figure #3: Names List button and button script.

This script first determines if the choice list is already being displayed by testing for the process specified by the ListPrcesNm variable. If the list window has not been opened PROCESS ATTRIBUTES leaves ListState with its initial value of -1 and we go on to start a new process. But if we've already pressed this button and opened the list window then PROCESS ATTRIBUTES assigns a greater-than-zero value to the ListState variable and brings its window to the foreground.
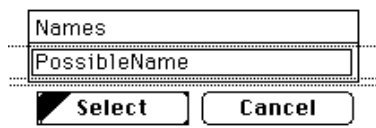
In the first case the New process command starts a new process with the name 'NamesList,' which appears in the process window in the Design environment, and runs the ChoiceList global procedure. When the ChoiceList procedure finishes 4D will close the NamesList process. In order to keep the NamesList process from immediately finishing the ChoiceList procedure enters a Modify Selection loop that waits for user input:

```
  `ChoiceList global procedure.
 OPEN WINDOW(100;200;300;450;4)
 ALL RECORDS([Names])
 MODIFY SELECTION([Names];*)
 CLOSE WINDOW
```

This opens a window showing the contents of the [Names] file using the output layout and layout procedure shown in figure 4:

```
┌─────────────────────────────┐
│ Names                       │
│─────────────────────────────│
│ PossibleName                │
│─────────────────────────────│
│  ◢ Select      │ Cancel │   │
└─────────────────────────────┘
```

```
 Case of               `Layout procedure for [Names];"Name_Ouput"
  : (Before)

  : (Outside call)`The input layout is indicating that this process should cancel itself.
    CANCEL
 End case
```

**Figure #4: output layout in the Design
environment and its procedure.**

The output layout procedure traps two phases, a Before phase and an Outside call phase. The Outside call phase only executes when this process is called from the input layout procedure, running in the first process, using the Call Process command; it is not executed at any other time in this example.

If the user hits the Cancel button they'll exit the MODIFY SELECTION, the list window will close and the new process will disappear. At this point the input layout, running in its own process, will come to the foreground.

If the user highlights a [Names] record and presses 'Select' the Select button script will run:


```
`Script of 'Select' button on layout [Names];"Name_Output".
If (Records in set("UserSet")=1)
 CREATE SET([Names];"CurrentSet")
 USE SET("UserSet")
  SelectRN:Record number([Names])
 USE SET("CurrentSet")
 CLEAR SET("CurrentSet")

 CALL PROCESS( BabyPrcesNm)
 BRING TO FRONT( BabyPrcesNm)

Else
 ALERT("Please highlight a record and press the SELECT button.")
End if
```

Figure #5: the Select button and script.

This script tests the 'UserSet' to see if a record is highlighted. If so it stores the current selection in a temporary set before replacing it with the highlighted record. It then assigns the highlighted record's record number to the interprocess variable SelectRN and restores the current selection.

Unlike all the global variables defined in the NamesList process the SelectRN variable can be referenced by the input layout procedure, however nothing will happen until the input layout procedure is reactivated. The CALL PROCESS command forces the input layout to run its Outside call phase and the BRING TO FRONT command brings the input layout to the foreground.

Referring back to the Baby_Input layout procedure you can see that the Outside call phase tests SelectRN which now contains a non-negative value. It uses this record number to locate the chosen record and assign its value to the FirstName field of the current Child record.

We need to perform this extra step of reestablishing the chosen record because each process has its own current selection. That means that even if we limited the current selection of [Names] to the single record highlighted in the NamesList process, the current selection of [Names] in the input layout procedure would remain unaffected. We can not use the current selection to carry information from one process to another the way we used it to carry information from one procedure to another. Instead we must use some explicit mechanism such as an interprocess variable.

Let's review where we stand. The user opens a list in a second process, makes a selection and returns to the first process. The user's choice is displayed in the FirstName field. The user can click on the list window, bringing it to the foreground, and select a different name. Or they can click in the list window and press Cancel to close the window.

The last interaction between the two processes occurs when the user accepts or cancels the input layout. In this case either the bAccept or bCancel variable is set to 1 and trapped by the 'If' statement in the During phase:

```
:(During)
  If ((bAccept=1) | (bCancel=1))
   CALL PROCESS(ListPrcesNm)  `Activate the other process if it's still running.
  End if
```

If the list window is still open the ListPrcesNm variable will refer to it. The CALL PROCESS(ListPrcesNm) will trigger the Outside call phase of the Name_Output layout which CANCEL's the layout and closes the process. The input layout then finishes processing and returns to whatever code opened it in the first place. This completes the entry of a new Baby record.

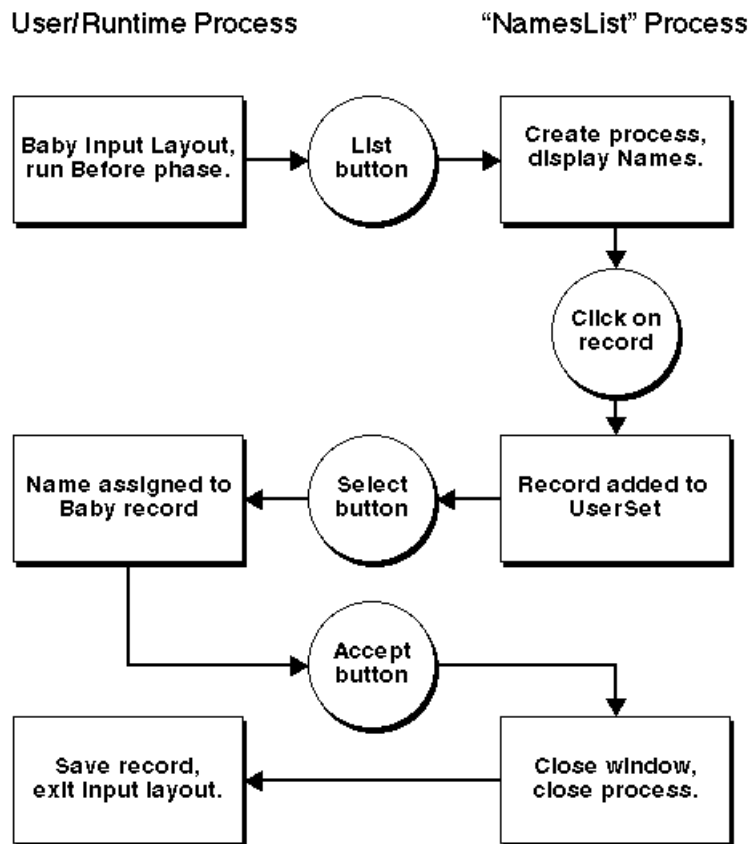The following schematic reviews what we've done.

Figure #6: schematic of multiprocess flow.

4D's new multiprocess tools provide the opportunity to learn powerful new programming skills. I hope this introduction will get you started.