#88 Custom Dialogs

Written by Dave Dell'Aquila Published Date April 12, 1988

Customized Message dialogs using **DISPLAY RECORD**

4th Dimension provides the MESSAGE command as an easy to use mechanism to display messages to the user. When called by itself, 4th Dimension will open a window centered horizontally and proportional in size to the window that is already open. Used in this fashion, it will default to using a Geneva 12 point font.

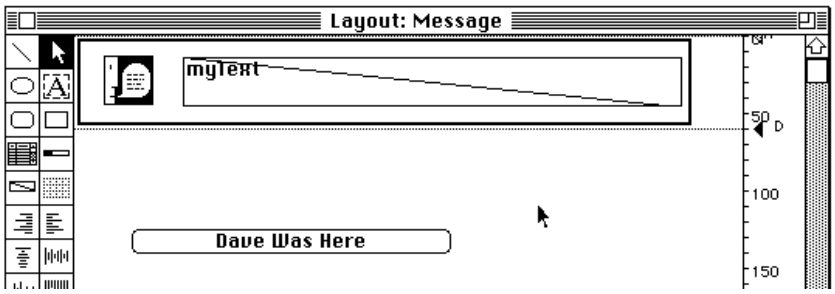
If you call OPEN WINDOW prior to calling MESSAGE, your message text will be displayed in Monaco 9 point. This font and size can be changed by modifying the CUST resource with ResEdit (see Tech Note 16). This Tech Note describes how to create your own custom message dialogs using any Macintosh window type and font you like.

Quick explanation

Open a window and display a record from another file with the DISPLAY RECORD command in a layout which contains your message text. When you're done displaying your message, close the window.

Detailed explanation

To use this technique, you will need to have a record available for displaying (preferably in a read only state) at any time. You will also need to create a layout which looks similar to the following sample layout.



Notice that this layout contains 4 elements:

- Non-enterable variable called 'myText'. Used for displaying the message text.
- An icon (a normal bit-map) for aesthetics. This could also have been a variable which would then make it easy to switch the icon displayed in the message dialog.
- An optional 2 pixel rectangle (explained below).
- A button-button (the button text is optional).

Opening the window

Here is a sample procedure called *myDIALOG* which will open a message window:

```
` Global Procedure: MyDialog
LOAD RECORD([Constants])                    `ensure existance of constants record
L:=(Screen width/2)-($1/2)
T:=110
OPEN WINDOW(L;T;L+$1;T+$2;2)
```

You would call it as follows:

myDIALOG(400;60)

A Word about Windows

4th Dimension allows you open to seven types of Macintosh windows. MultiFinder will allow a user to switch from one application to another by clicking in the target application's widow, except from a modal window. In other words, MultiFinder will not let a user switch applications from a window of type 1. In the above example, it appears to the user that they are looking at a window of type 1, the type most commonly used to display messages on the Macintosh. Often times, it's nice to be able to switch out of a 4th Dimension database that is involved in some long process to work on other applications, but this is not allowed with window type 1. The solution is to create the illusion of a type 1 window by using a type 2 window, and 'faking' the 2 pixel border which appears inside the 1 pixel window border. Since type 2 windows are modeless, the user can switch out of the database into another application.

Displaying the message

To make the use of this technique as much like 4th DIMENSION's built-in method of displaying messages, you would create a procedure to display any text you pass to it. This procedure takes care of assigning the text passed to it to the variable which will be displayed in the layout "Message", it is called *myMESSAGE*:

```
` Global Procedure: MyMESSAGE
INPUT LAYOUT([Constants];"Message")
myText:=$1
DISPLAY RECORD([Constants])
```

Closing the window

When you're done displaying your message, be sure to close your message dialog with CLOSE WINDOW. If this step is omitted, all following actions will occur in the window you opened for the message.

Bringing it all together

Here's some sample code to illustrate how this technique is normally used. In the example, I'm applying a global procedure called *myUpdate* to the current selection of records.

myDIALOG (400;60)

```
myMESSAGE ("Updating records...")
APPLY TO SELECTION([myFile];myUpdate)
myMESSAGE ("Doing something else...")
APPLY TO SELECTION([myFile];myUpdate)
CLOSE WINDOW
```