ViewEdit User's Guide

preliminary draft 1

• WARNING • This is the first alpha release of ViewEdit. It is untested and not yet beta quality. It should be used with extreme caution: it may crash your system or even erase files from your hard disk. Save your work often and don't use ViewEdit when a system crash would cause you to lose data.

Preface

Welcome to the ViewEdit User's Guide, and to the power of ViewEdit. ViewEdit is a MacApp utility program that allows you to create view hierarchies in a what-you-see-is-what-you-get editing environment, rather than in a compiled resource file.

ViewEdit gives you as much view-editing power as Rez but provides Commando-like dialogs for entering values into each 'view' resource field. This relieves you of having to remember which fields and values are associated with which 'view' types.

ViewEdit also allows you to draw, resize, and move your views using the standard Macintosh interface. It even creates and rearranges your view hierarchies as you go!

Before you read this document, you should understand these concepts:

- **View hierarchies.** These are introduced in the "Architecture" section of Chapter 4 of the *MacApp 2.x Manual (Interim Version).*
- **View classes.** These are described in the MacApp® 2.0B5 Display Architecture Release Notes.
- **View resources.** These are explained in the "Creating View Templates" recipe in Chapter 7, "The CookBook", in the *MacApp 2.x Manual (Interim Version).*

This guide is divided into two parts: a step-by-step tutorial and a command reference. ViewEdit is simple enough to use that you may find you won't need to refer to this manual frequently. However, you should read through this manual at least once, for there are many shortcuts and features hidden in ViewEdit, as well as a few eccentricities.

©1988 Apple Computer

A first look at ViewEdit

This section shows you how to use ViewEdit to edit the view resources in the DemoDialogs sample application. Before you begin this tutorial, you should build the DemoDialogs sample. If you are new to MacApp and the MPW environment, Chapter 6, of the *MacApp 2.x Manual (Interim Version)*, "How to Install and Use MacApp", will show you how to build the sample applications.

After building DemoDialogs, open the ViewEdit application by double-clicking on its icon.

The resource file window

The first thing that you will see after starting ViewEdit is the empty "Untitled-1" window. Close this window and choose the Open command from the File menu. Then select the DemoDialogs application from the Standard File dialog.

The window that appears is reminiscent of the resource file window in ResEdit. There is an icon for each type of resource in DemoDialogs. The icon for 'view' resources is always in the upper-left hand corner (and if you have a color screen, you will see it is also the only colored icon).

Here is the resource file window from DemoDialogs:

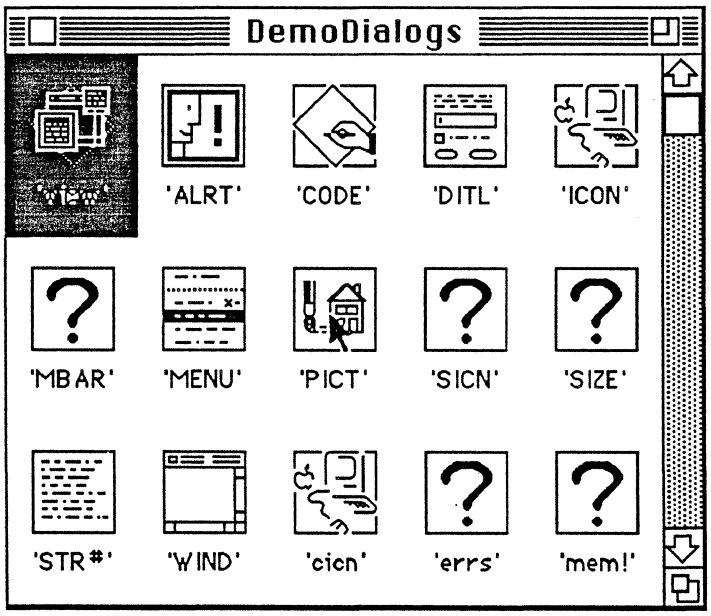


Fig 1.1 DemoDialogs resource file window

Because there is a view icon present, you can tell DemoDialogs already contains view resources. If no view resources existed, you could create an initial view resource at this point by choosing the Create Resource command from the Edit menu.

©1988 Apple Computer

The resource type window

If you double-click on any of the resource type icons in the resource file window, you will open a window that contains a complete list of resources of that type. Alternatively, you can select the resource type with the cursor keys and press the Return or Enter key.

For example, double-clicking on the 'ICON' resource icon will open this window:

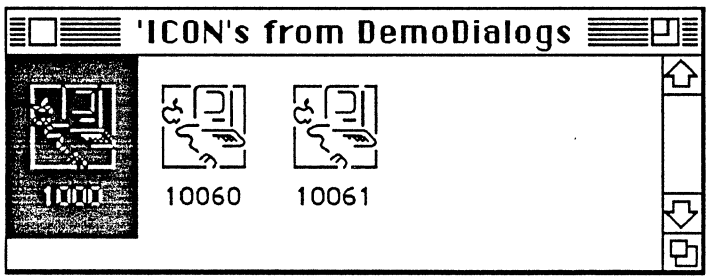


Fig 1.2 'ICON' resource type window

ViewEdit will display the ID's of a variety of resources, but the only resources it allows you to examine and edit are views. If you try to "open" any of these 'ICON' resources, as you might in a complete resource editor, nothing will happen.

©1988 Apple Computer

On the other hand, when you open the 'view' resource type window, which looks like this:

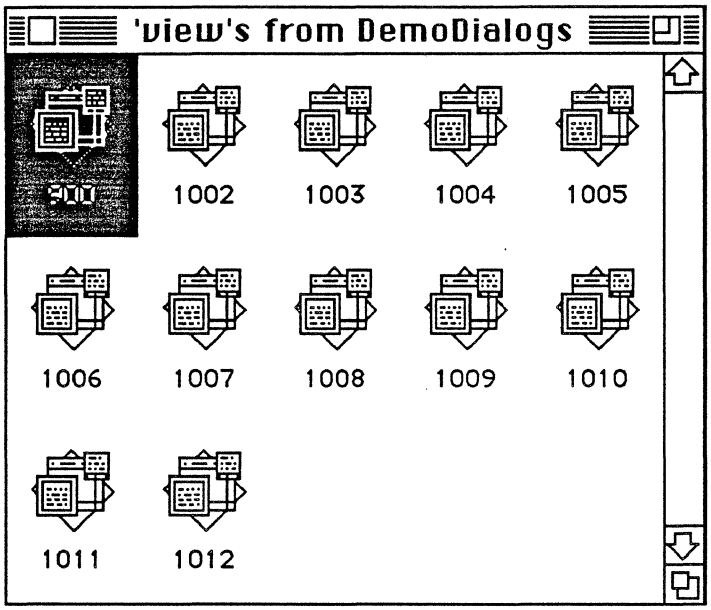


Fig 1.3 'view' resource type window

you have the ability to open and edit each of these resources. At this point you may also want to create a new 'view' resource, by choosing the Create Resource command from the Edit menu.

Double-clicking on any of the 'view' resource icons will result in opening a **view-editing window.** (Again, you can alternatively select a 'view' resource icon with the cursor keys, and then press the Return or Enter key.)

©1988 Apple Computer

The view-editing window

Simply put, the view-editing window allows you to edit your view resources. Let's look at a sample view-editing window. Double-click on the icon representing 'view' resource ID 1008, which will open this view-editing window:

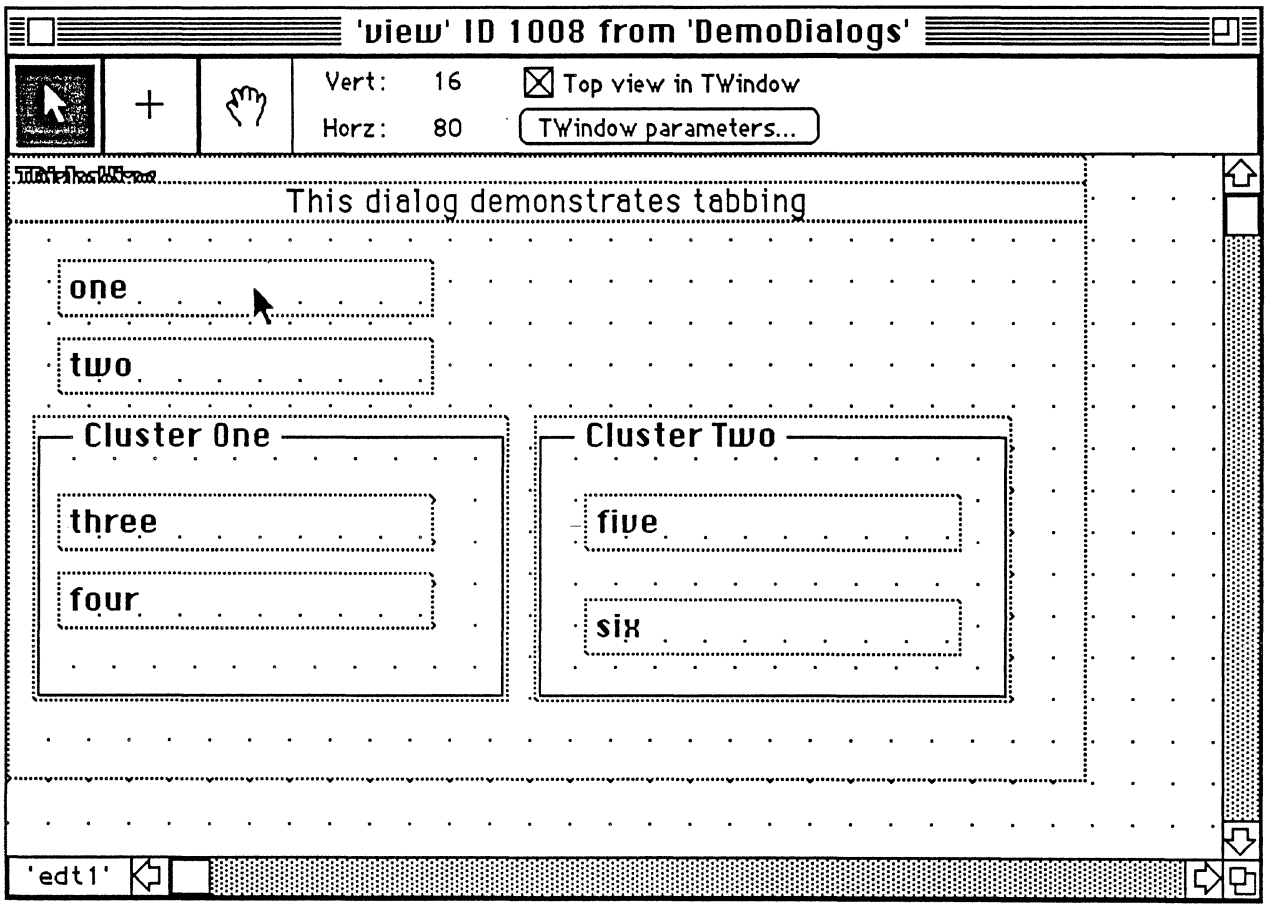


Fig 1.4 'view' ID 1008 view-editing window

©1988 Apple Computer

To refresh your memory, this view resource corresponds to the "Tabbing Test" dialog in DemoDialogs, which looks like this:

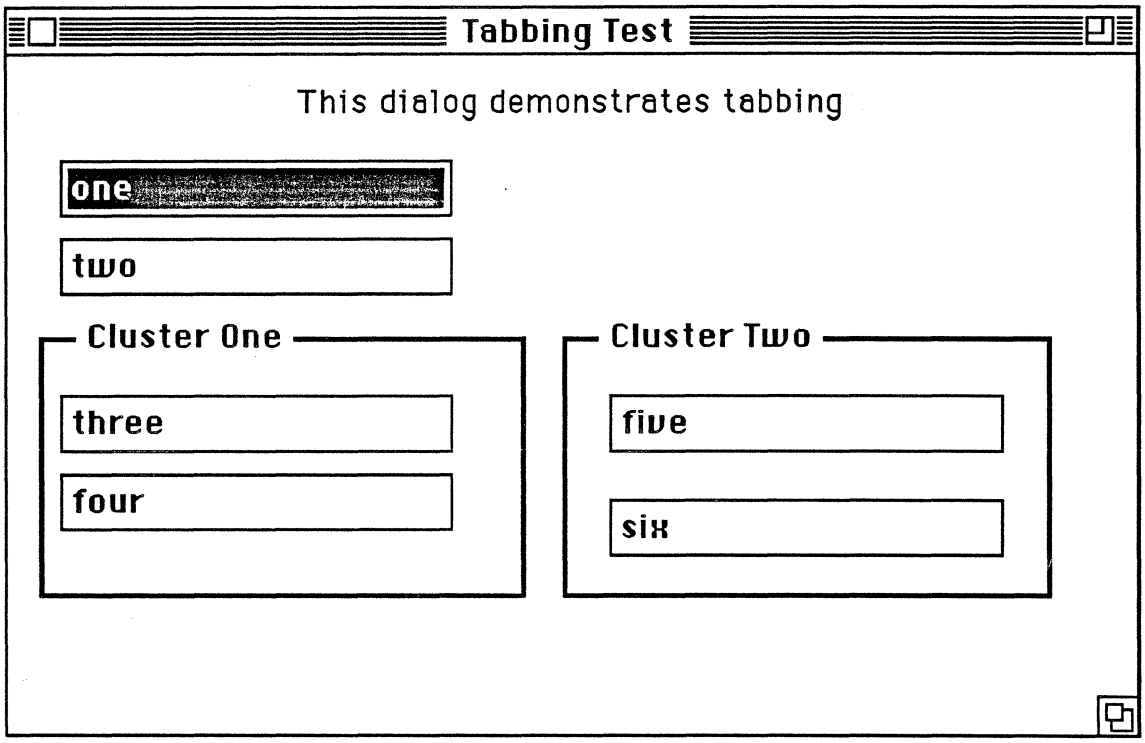


Fig 1.5 "Tabbing Test" dialog from DemoDialogs

As you can see, the view-editing window closely resembles the actual dialog from the application. However there are a few important differences.

First of all, the view-editing window has a palette attached to it. This palette consists of three tools, a vertical and horizontal location indicator, a check box, and a button. The elements of this palette, called the control palette, are described in the following sections.

Under the control palette is the actual representation of the view resource. Each view in the view resource hierarchy is represented here. Each visible view is surrounded by a dotted line, and certain view types have other visual clues. For example, the static and editable text views in this view-editing window are filled with their initial text. Similarly, the cluster views each are drawn with their label and border. This allows the view-editing window to duplicate the actual dialog as closely as possible. However some views, like dialog views, have no visual representation in the final application. Views of this sort are labeled by their type name in their upper left hand corner. In the current view-editing window, you can see the label "TDialogView" (which is partially hidden) in the upper left hand corner of the entire view-editing window. As you see more and different types of views, you'll see exactly how much of a visual clue is associated with each type. You might be surprised how faithful some of these visual cues are.

You might be wondering why the "TDialogView" label is partially hidden. This is due to the implementation of the view hierarchy. In the view-editing window, superviews are placed under their subviews. In this example, the TDialogView view is the superview of the static text view that contains the text "This dialog demonstrates tabbing." Therefore, everything associated with the TDialogView view (of which the "TDialogView" label is the most conspicuous part) is drawn below the uneditable text view. The same holds true for the first two editable text views and the two clusters. Each cluster is the superview of two more editable text views; so once again, the subviews are drawn on top of their superview.

Another aspect of the view-editing window you might notice is that it is covered with dots. These dots represent the current **grid** of the window. This grid, like the grid in MacDraw, constrains the drawing and moving of views.

As you move the pointer around the view-editing window, several items are updated. In the bottom-left-hand corner a message box which displays the identifier of the view the pointer is currently over. In Figure 1.4, the identifier is 'edt1'. Also as you move the pointer around, you will see its current coordinates appear in the Vert and Horz fields in the control palette. The coordinate values displayed in these fields are always relative to the local coordinate system of whichever view the pointer is currently over. For example, in Figure 1.4, the pointer is positioned over the editable text view containing the text "one." The vertical and horizontal coordinates shown are 16 and 80. Since the local coordinate system of this text box starts at (0, 0) in its upper left hand corner, this means that the pointer is currently 16 pixels down and 80 pixels over from the upper left hand corner of this box.

• **NOTE** • Actually, the coordinates shown are "snapped" to the current grid. In other words, they are rounded to the nearest value allowed by the current grid resolution.

Another important point to note is that the grid is relative to the origin of every view. If some subview is not currently aligned to its superview's grid, then its grid will be slightly shifted from its superview's grid. This is the case with every subview in Figure 1.4.

The selecting tool

The first of the three tools in the control palette is the selecting tool, which is the standard "pointer" cursor. It is the default tool when you open a view-editing window. You can choose this tool either by clicking on its icon in the control palette or by choosing the Select Views command in the Mode menu. With this tool you can select, resize, and open a view.

Selecting a view, as you might imagine, requires a single click anywhere in that particular view. Of course, if the view is covered by other views, you will have to find some uncovered piece to be able to select it—otherwise you'd have no place to click.

You can execute editing commands on the selected view, including Cut, Copy, and Clear from the Edit menu. You can also Delete by pressing the Delete key.

©1988 Apple Computer

A selected view looks like this:

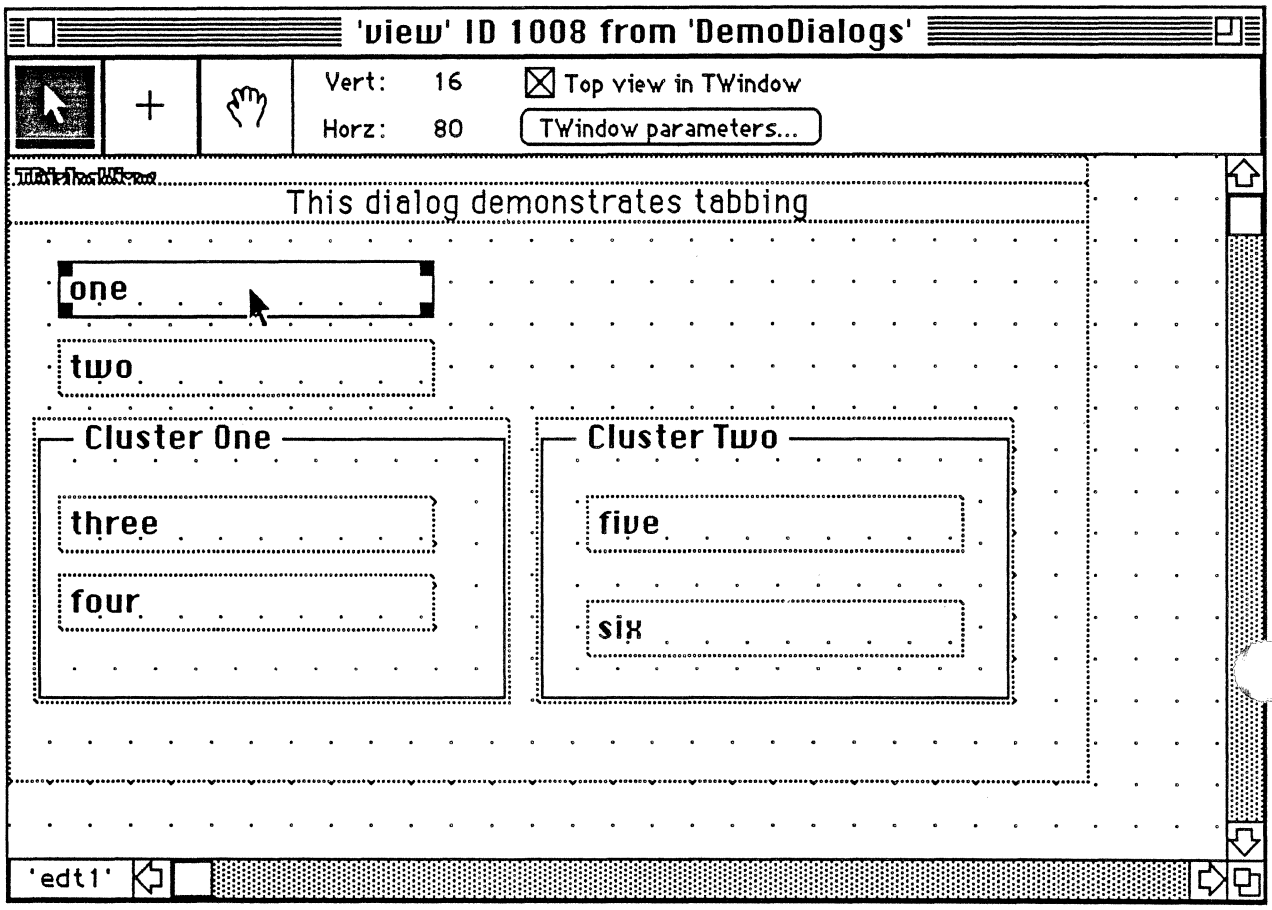


Fig 1.6 A selected editable text view

©1988 Apple Computer

Notice that four **handles** have appeared—one in each corner of the view. These handles allow you to resize the view, by clicking and dragging. For example, if you drag the lower-right handle to the right, you can resize the view accordingly:

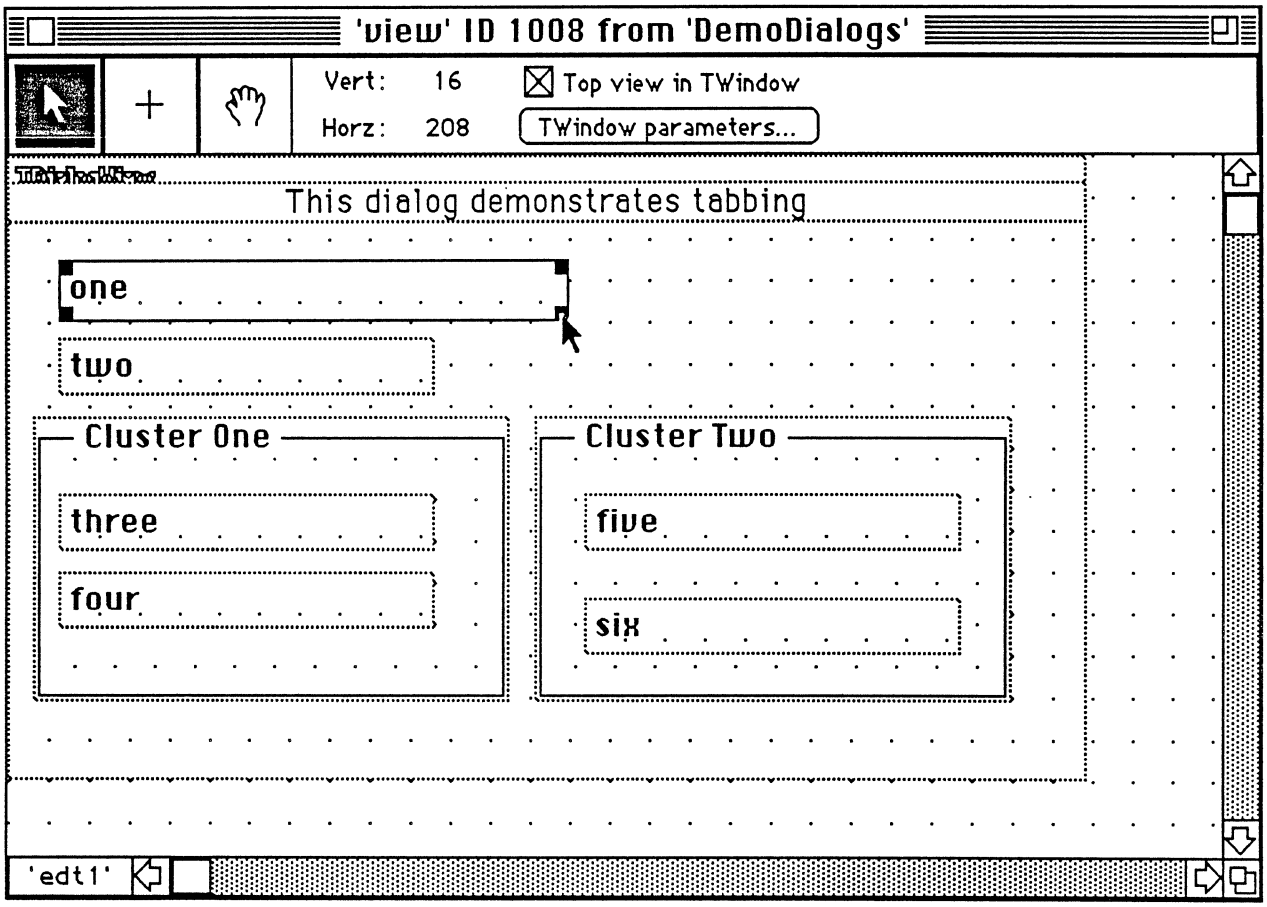


Fig 1.7 Resizing a view

If you're performing each step of this tutorial in ViewEdit, you probably noticed the effect of the grid. Although the editable text view was originally not aligned with its superview's grid, once you grabbed that handle and started to move it, it immediately snapped to the nearest grid point (in the superview's grid). Unless you recalibrate or turn off the grid, that corner will stay aligned to grid points for each subsequent move or resizing.

©1988 Apple Computer

Notice only that corner handle is now completely aligned to the superview's grid. To align the others, you need to select and move them. For example, you can now align the upper right handle by grabbing and dragging it:

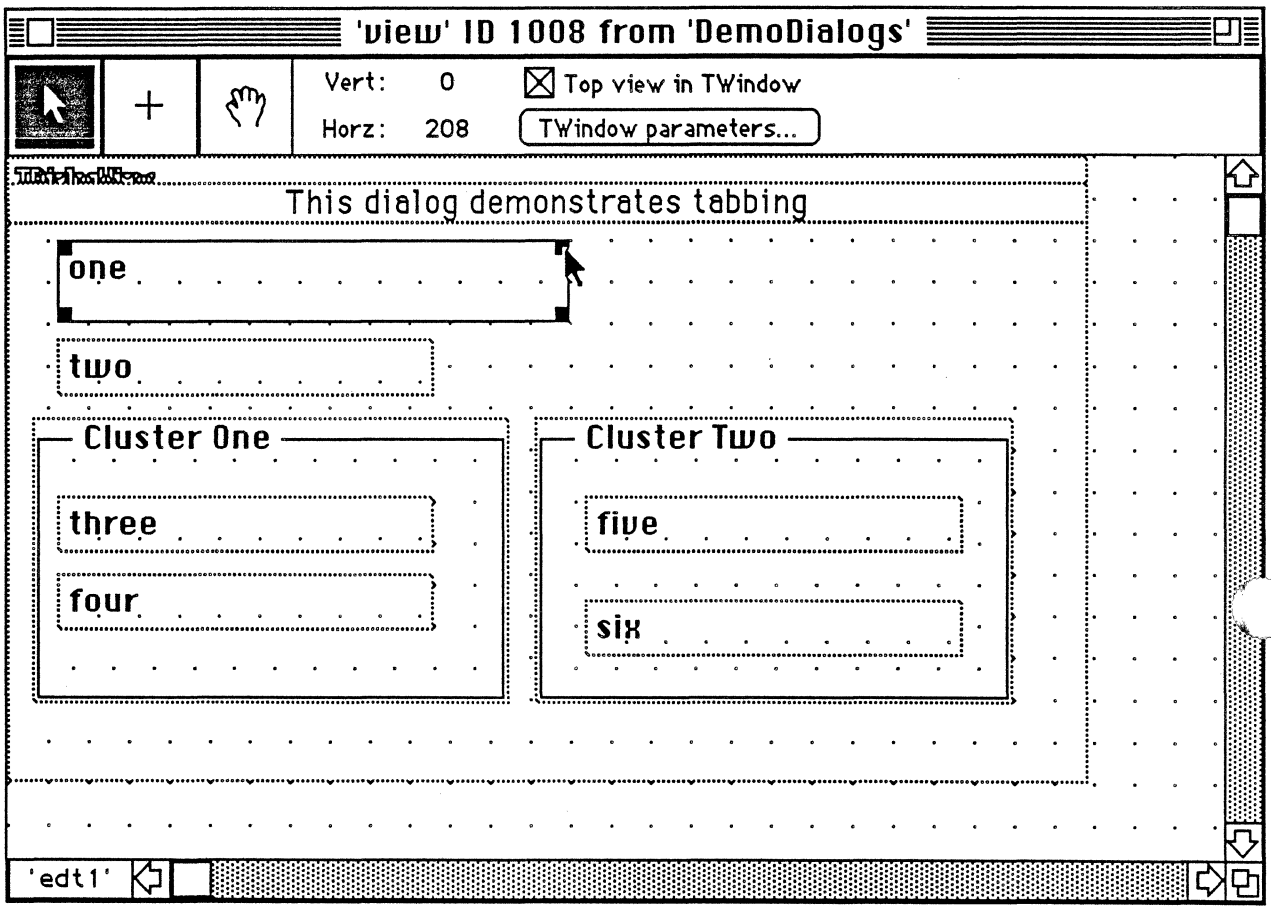


Fig 1.8 More view resizing

Of course, if you only wish to align a view to its superview, you don't have to move each handle. You can simply choose the Snap to Grid command in the Arrange menu.

At this point, you may not like the new position of the editable text view. You can always revert to the last saved version of the entire file (which would be acceptable, since this is the only change so far), or you can resize the view back to its original size. To do this, you need to manipulate the grid.

There are a number of ways that you can manipulate the grid. You can make the grid invisible by selecting the Show Grid command in the Arrange menu (which is a toggle command) but this won't turn the grid off. To do that, you must choose the Grid Values command from the Arrange menu, unhighlight the Use Horizontal Grid and the Use

Vertical Grid check boxes in the Grid Values dialog box. Notice you can also change the grid resolution in this dialog box. (This simultaneously changes the grid in every view of the view editing window.)

When a view is selected, you can automatically resize it to align to the current grid of its superview by choosing the Snap to Grid command in the Arrange menu. You can also rearrange the order of subviews in a superview. (Remember that the order of subviews affects how they appear on the screen. A subview "in front" of another subview will cover the other subview if their boundary rectangles intersect.) The Send Back and Bring Forward commands in the Arrange menu allow you to reposition the selected view.

The sketching tool

The sketching tool (a crosshair pointer) allows you to create new objects by **sketching**—clicking and dragging to define the boundary rectangle of the new view, much as you would draw a rectangle in MacPaint. The view that is created is added to the view hierarchy as a subview of whichever view the initial click occurred in. The type of view that is created is whichever view type is currently selected in the Views menu.

For example, if you select the TPopup view type from the Views menu and sketch the following rectangle:

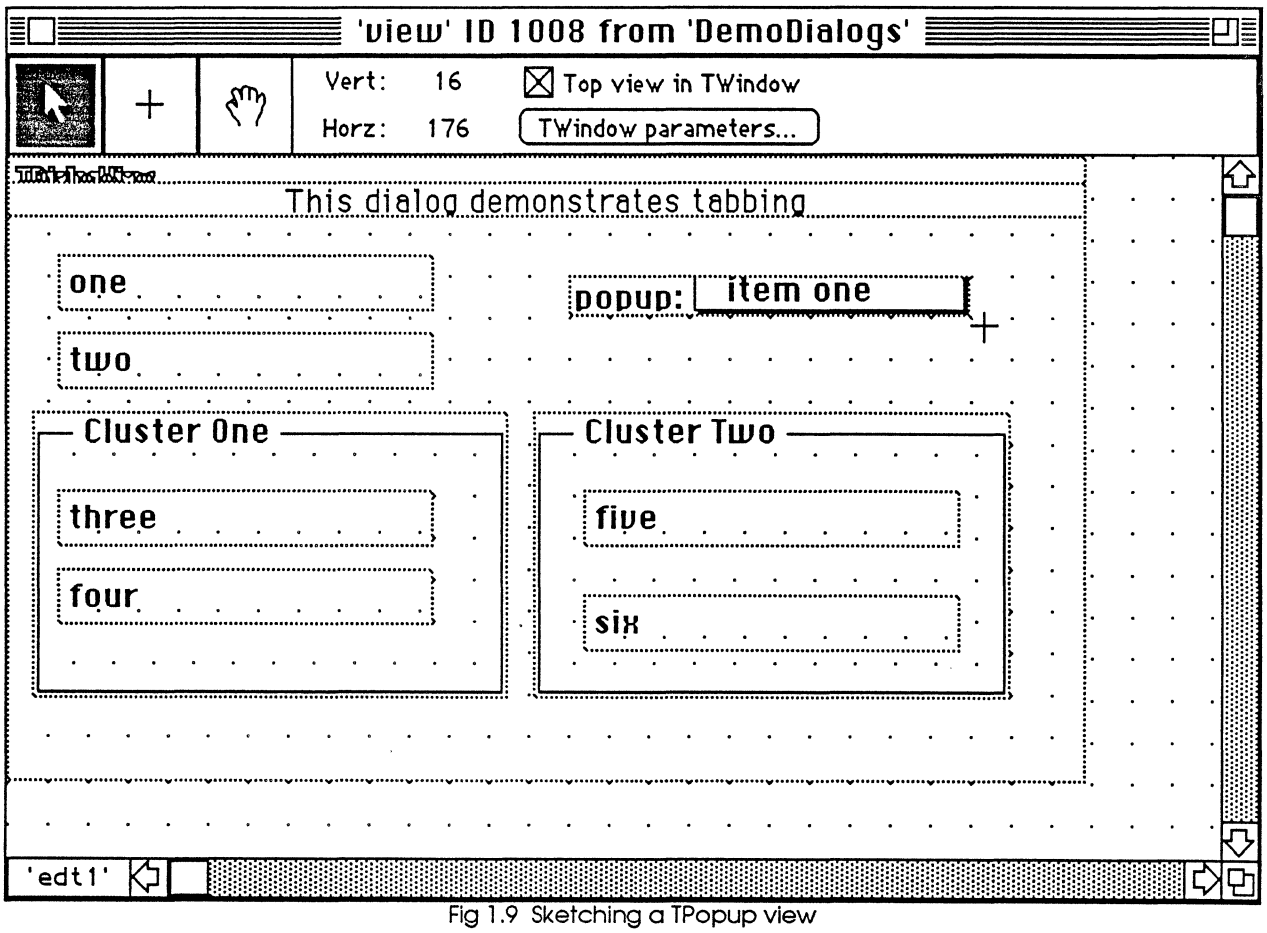


Fig 1.9 Sketching a TPopup view

©1988 Apple Computer

A TPopup view is created as a subview of the TDialogView view. Notice that this sketching, as you might expect, is constrained to the current grid.

There are three ways to select the sketching tool. Like the selecting tool, you can select the crosshair icon from the control palette or you can choose the Draw Views command from the Mode menu. As a third alternative you can hold down the command key. This will change the selecting tool into the sketching tool for as long as you hold the command key down. (If you release the command key while actually doing a sketch, the sketching tool will remain until you release the mouse.)

• **WARNING** • A final important fact about sketching a new view: the new view is not automatically selected. In other words, if the editable text view containing "one" was selected before you sketched the TPopup view, that text view will remain selected even after you've created the new view. This means that if you press the delete key immediately after sketching a new view, thinking that you will delete it, you may actually delete an entirely different view—the last one selected. Fortunately, the Undo command will bring back the lost view.

The moving tool

The moving tool (represented by the "grabbing hand" icon) allows you to grab a view and move it to another location, as well as to another place in the view hierarchy.

As with the sketching tool there are three ways to select the moving tool: selecting the grabbing hand icon from the control palette, choosing the Drag Views command from the Mode menu, and holding down the Option key. This last choice is a shortcut which temporarily changes the selecting tool into the moving tool.

Moving a view has interesting repercussions on your view hierarchy. If you move a view out of the boundaries of its superview, the following dialog box appears:

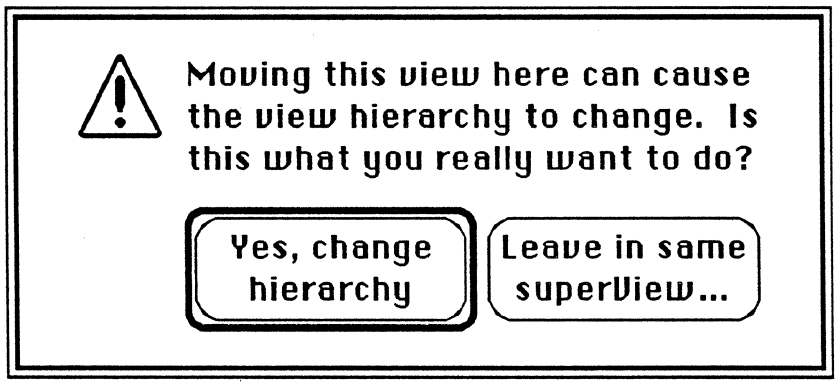


Fig 1.10 Change of hierarchy dialog box

This dialog box is straightforward: you can have the view remain a subview of the same superview as before, despite

the fact that you've moved it, or you can have the view become a subview of whichever view its new location implies.

When you move a view, it is the position of its upper-left corner that decides which view is the "proper" superview. In other words, if the upper-left corner of the view moves out of the boundary rectangle of its superview, and into another, then the dialog box in Figure 1.10 appears, and if you specify that you did intend to change superviews, then it is the location of this upper-left corner that decides which is the new superview.

To avoid this dialog box altogether, you can hold down the Control key (on keyboards that have a Control key) while you are moving the view. This forces the view to change its place in the hierarchy.

As with the sketching tool, the moving tool does not select the item that is being moved—you must select with the selecting tool only. Be careful not to assume that the moved view is now selected and try to delete it by pressing the Delete key!

©1988 Apple Computer

The view description dialog

You've seen how to select a view by clicking on it with the selecting tool. If you double-click on a view, the view "opens" into a **view description dialog**. (Alternatively, you can press return or enter while the view is highlighted.)

A view description dialog, like a Commando dialog, contains fields, pop-up menus, and help information to allow you to control every aspect of the different view types. For example, if you double-click on the selected TDialogView, like this:

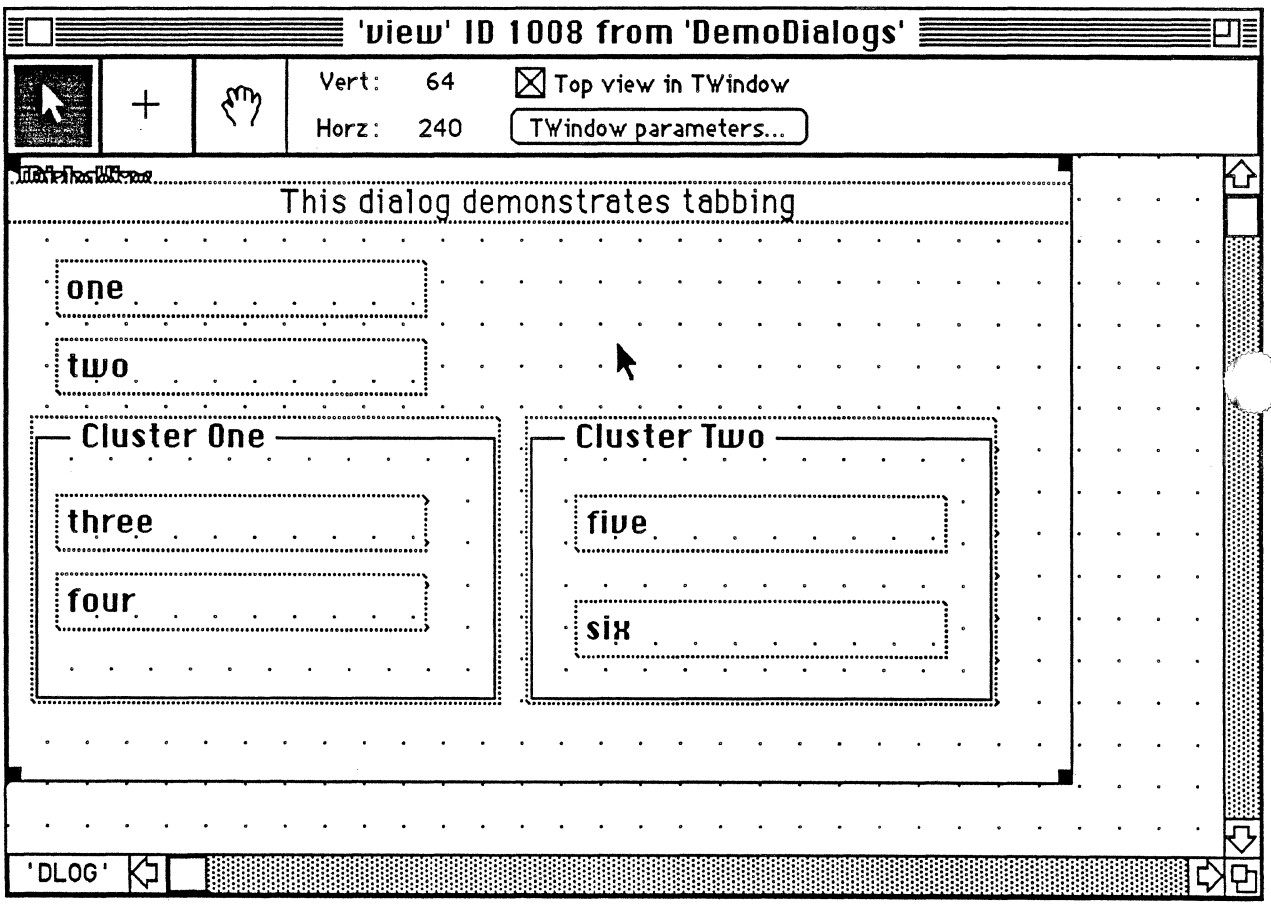


Fig 1.11 Opening the TDialogView view

you will open the following view description dialog:

©1988 Apple Computer

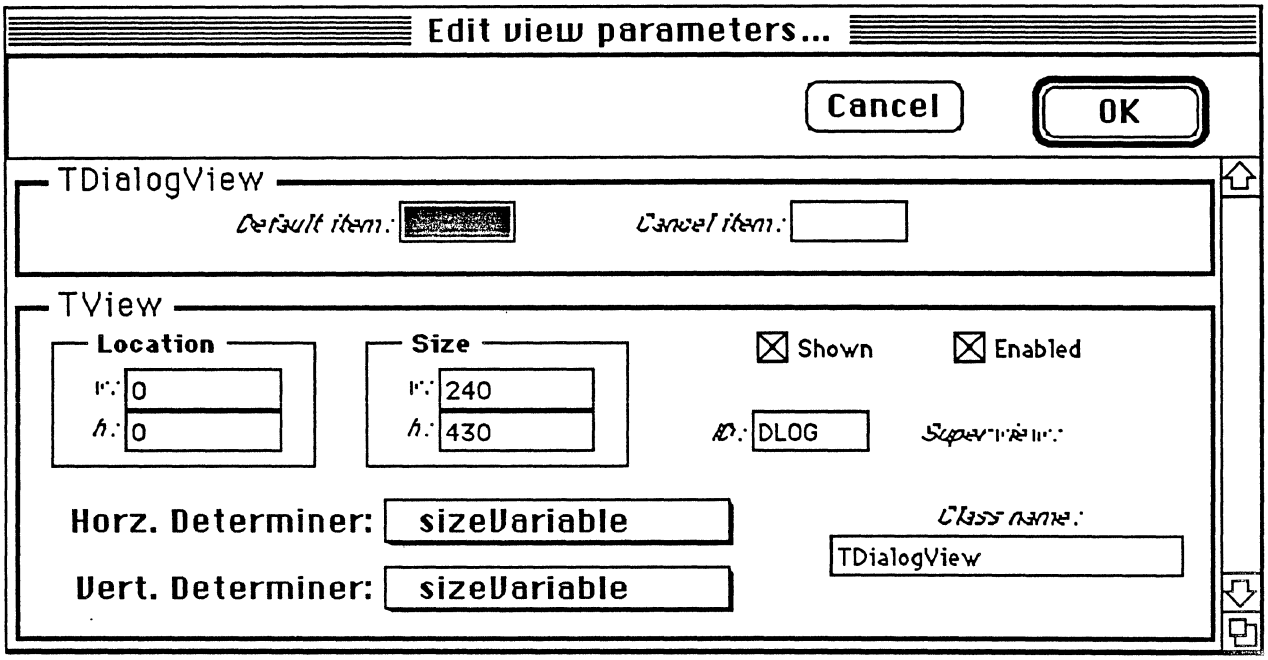


Fig 1.12 A TDialogView view description dialog

This view description dialog consists of two parts, or **view description boxes.** The upper view description box allows you to enter values associated with the fields of a TDialogView. The lower view description box allows you to enter inherited fields—in this case, those belonging to the TView class.

Each view description dialog follows this pattern—each has a number of view description boxes. The uppermost box represents the specific class of the view that was opened. Each successive box represents the ancestor class of the previous box, until the class TView is reached.

©1988 Apple Computer

Let's take another example. If you double-click on the static text box, like this:

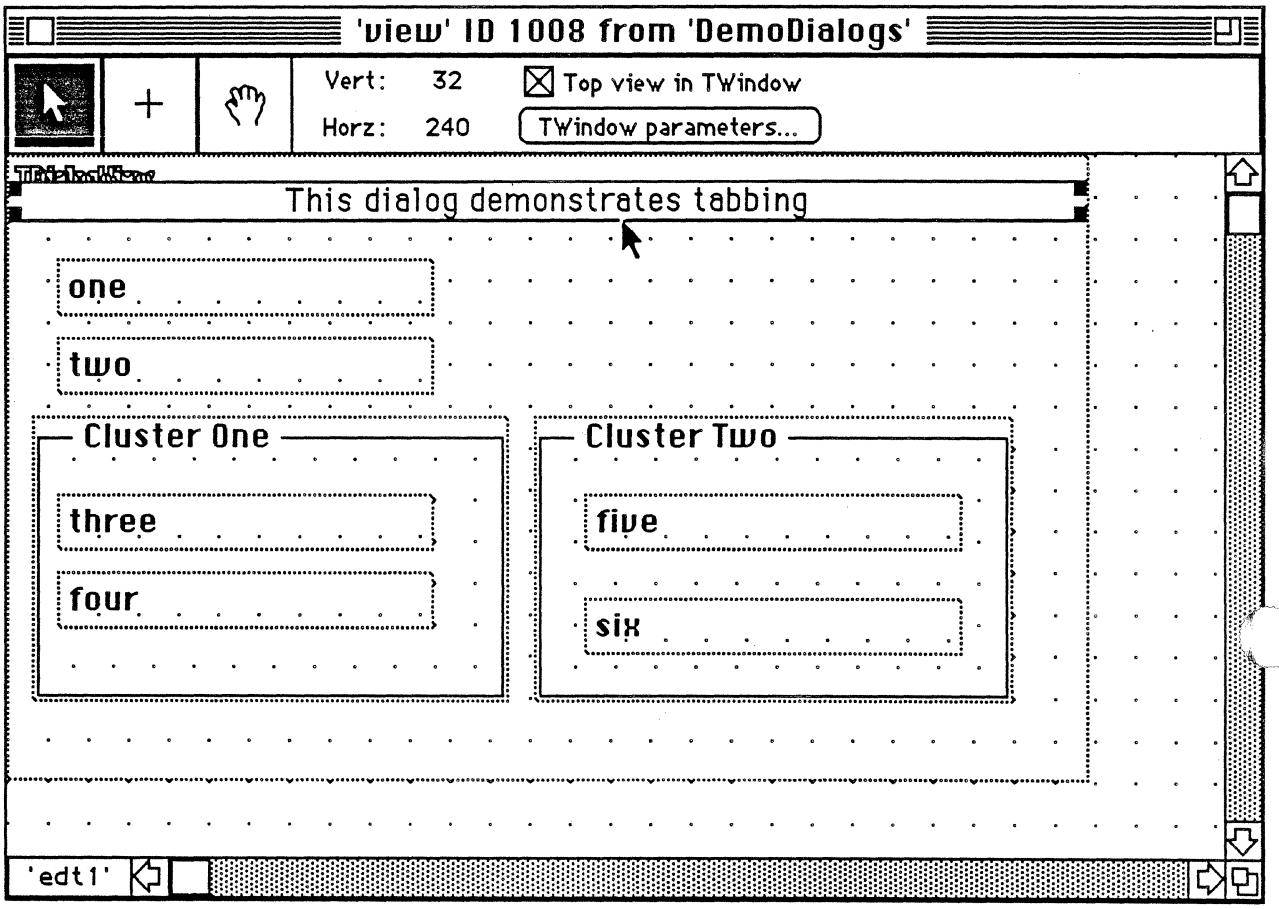


Fig 1.13 Opening a TStaticText view

©1988 Apple Computer

Then you will open the following view description dialog:

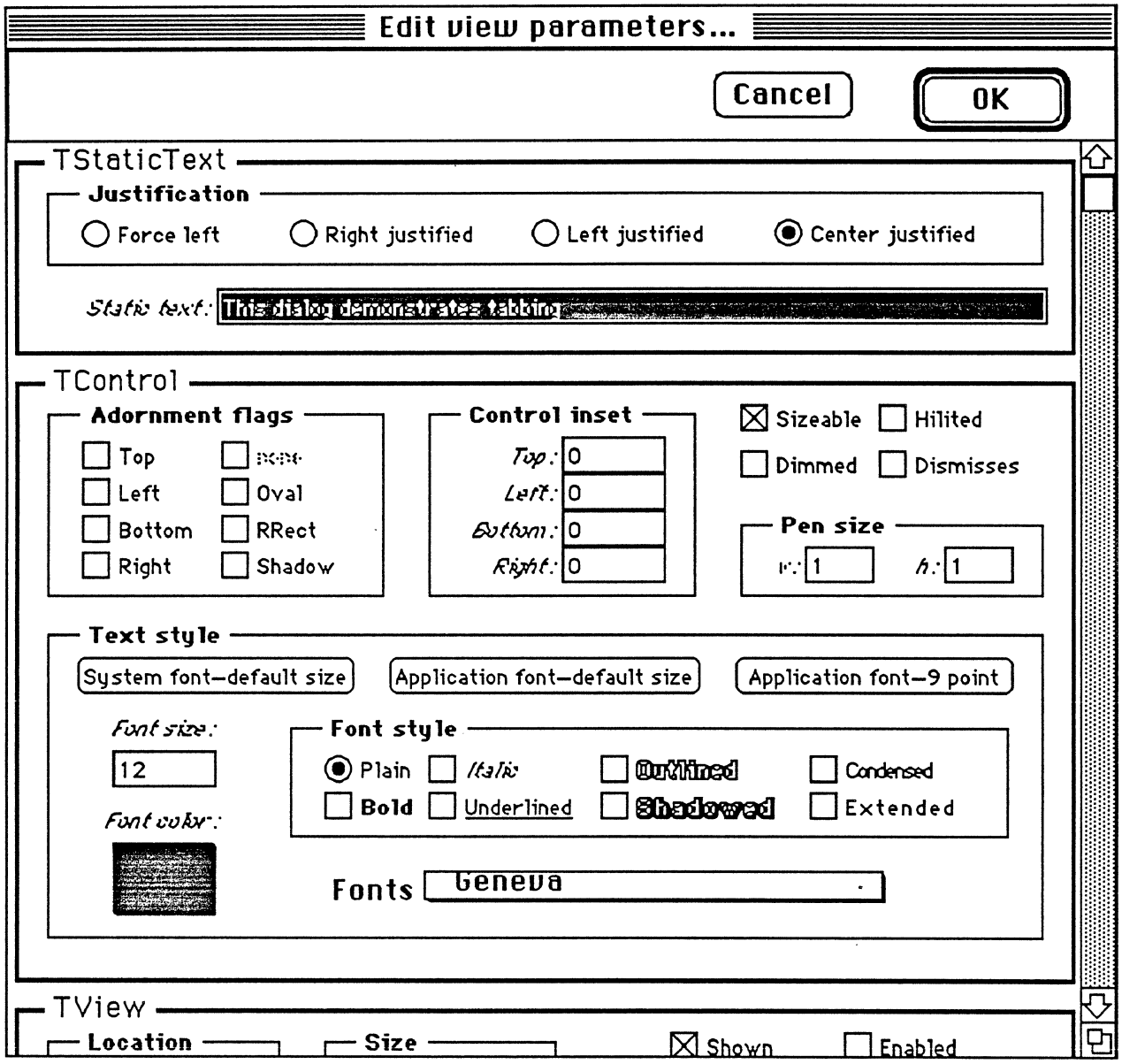


Fig 1.14 A TStaticText view description dialog

As before, you can see a number of view description boxes—one for each class in the view class hierarchy between TStaticText and TView.

©1988 Apple Computer

As a final example, open the view description dialog for a view of the TEditText class:

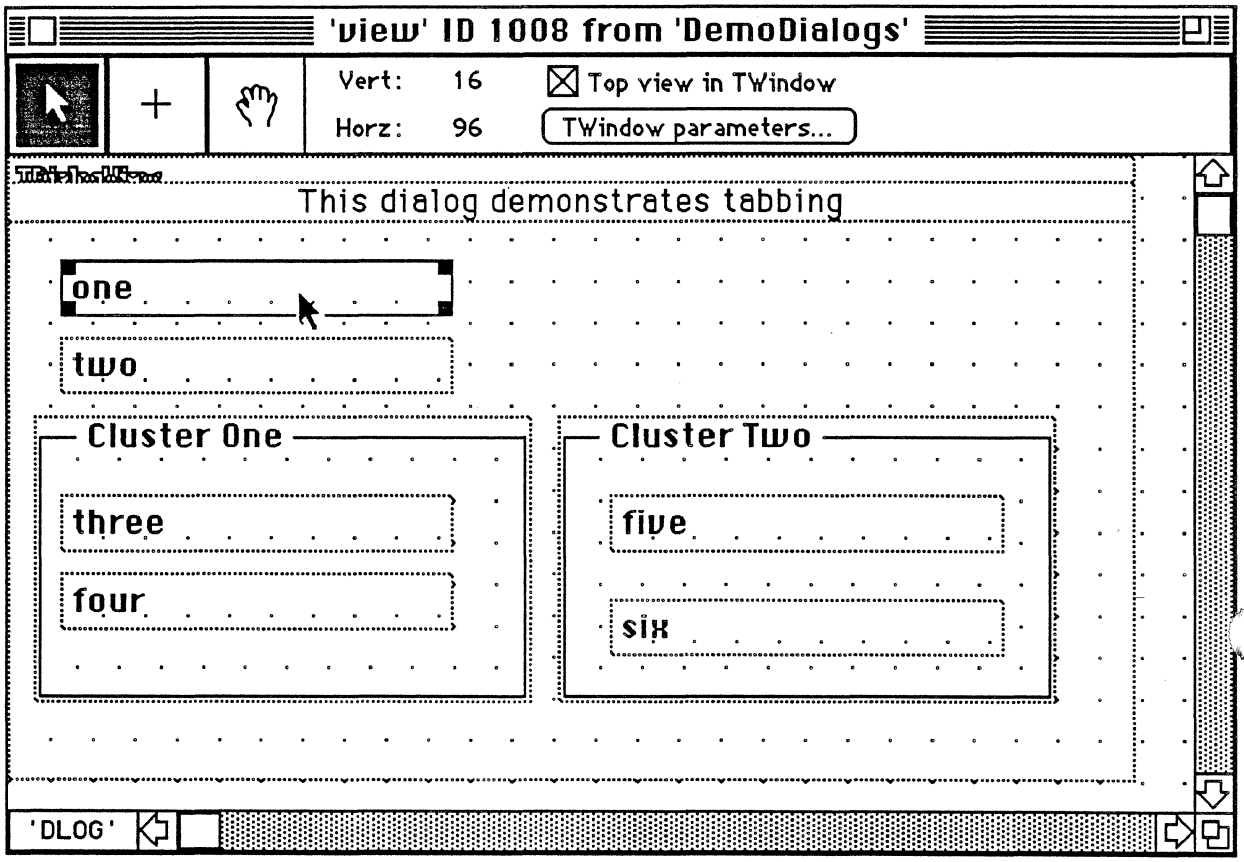


Fig 1.15 Opening a TEditText view

©1988 Apple Computer

This view description dialog should look like this:

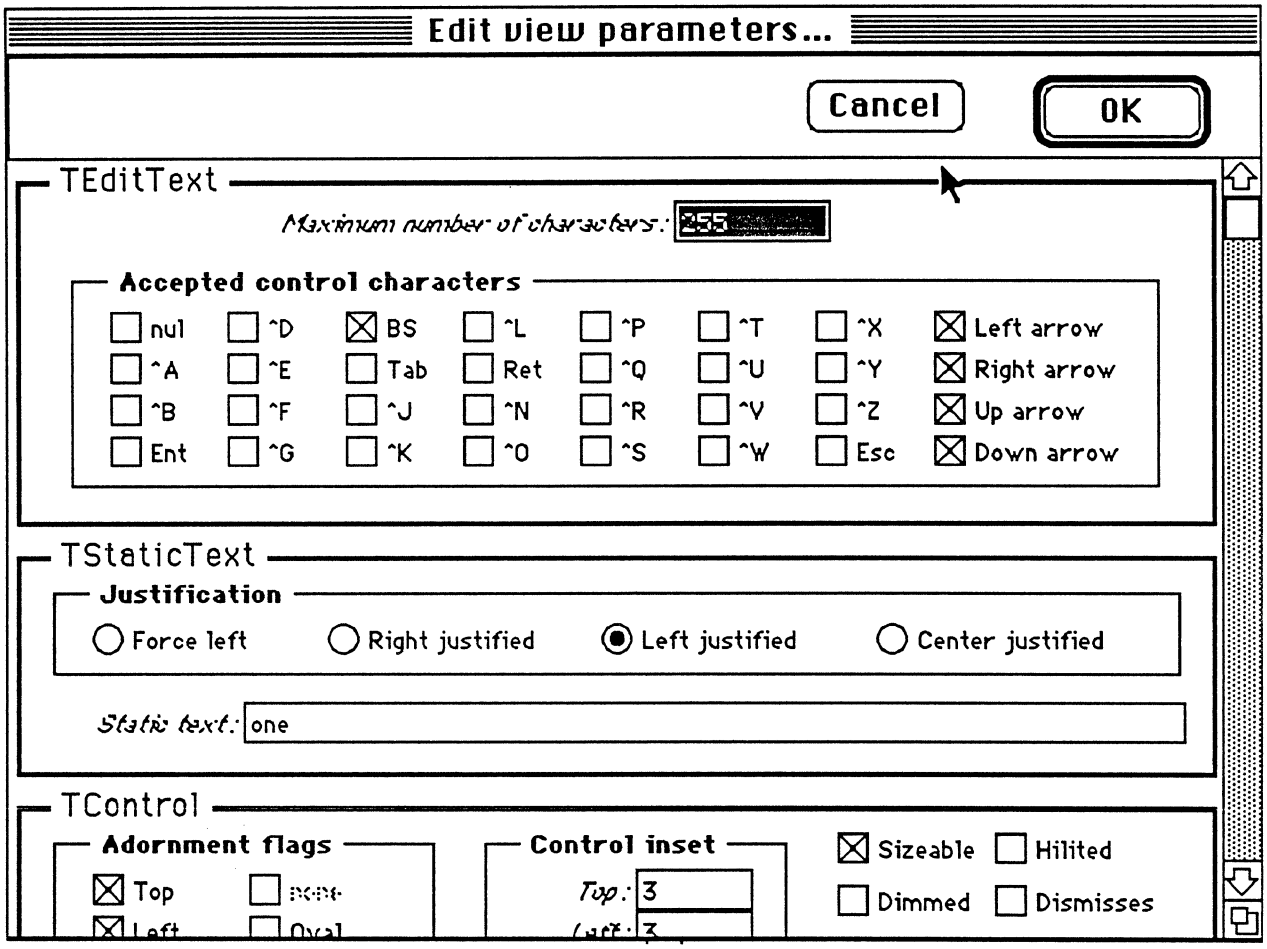


Fig 1.16 A TStaticText view description dialog

Here you can see an interesting change—the scroll bar actually becomes necessary for a dialog!

The different view description boxes allow you to input all of the information that a view resource file does. For further description of the different fields and what they mean, see the "Creating View Templates" recipe in Chapter 7, "The CookBook", in the *MacApp 2.x Manual (Interim Version)*, and the MacApp Release Notes. For an example of each type of view description box, see the "View Description Boxes" section at the end of this guide.

©1988 Apple Computer

The window description dialog

So far, you've seen a way to create and edit every type of view object *except windows*. Window view objects are a special case because they must always be at the top of the view instance hierarchy.

If you want the views displayed in your view-editing window to be subviews of an actual window object, then click on the Top View in TWindow checkbox in the control palette. This tells ViewEdit that the view hierarchy you have created is to be made a subview of a TWindow view object. The topmost views in the view-editing window will be placed in the TWindow view in the same position that they are currently placed in the view-editing window.

To edit the fields of other view objects, you double-click in their boundary rectangle to open their view description dialog. Similarly, to edit the fields of window view objects, you click on the TWindow parameters button in the control palette. This opens the view description dialog for the window superview. For the 'view' ID 1008 from previous examples, this dialog looks like this:

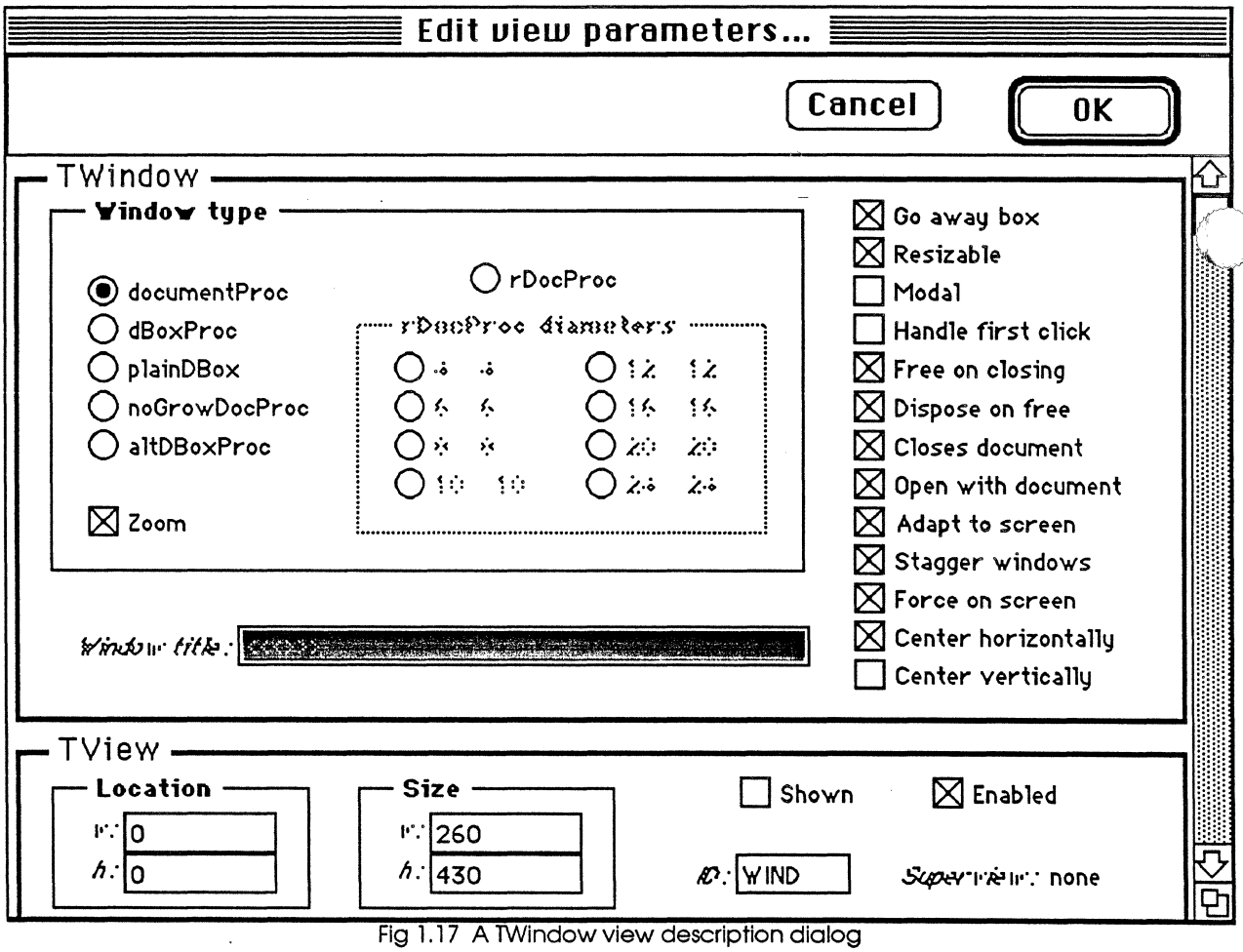


Fig 1.17 A TWindow view description dialog

©1988 Apple Computer

In all other ways, this view description dialog is the same as other view description dialogs.

• **WARNING** • Currently, changes made in view description dialogs *are not undoable*. While the view description dialog is open, the Undo command in the Edit menu will not work—don't try it. When you close the view description dialog, the Undo command does not undo changes made in the dialog, and may be dangerous. Don't use it until you've done some new undoable action!

A few more view examples

So far, you've really only seen a few of the many different types of views. You should probably use ViewEdit to experiment with some of the other views in DemoDialogs and other sample applications. Be sure to make a copy of any important file that you open with ViewEdit!

In your experimenting, you may run across a few issues that haven't been mentioned yet. For example, if a view in a view-editing window is any of the resource-driven views (pop-ups, icons, patterns, pictures), and a resource of the appropriate type and number exists in the resource file, ViewEdit will put a "working" version of that view in the view-editing window. If a corresponding resource does not exist, ViewEdit puts its own version of that type of view in the view-editing window. This feature allows your view-editing window to be faithful to the actual view in your application.

Another hint is to remember that ViewEdit displays what is in actual 'view' resources—sometimes these may look different than actual views you see in an application. For example, certain views are missing, or incorrectly sized. This discrepancy occurs when an application modifies its view resources before displaying them.

 ©1988 Apple Computer

A good example of this is TSScrollbars. Remember that TScroller views programmatically create their corresponding TSScrollbar so you won't see TSScrollbar views in view-editing windows. You will, however, see a *space* for the TSScrollbar view, as in the case of 'view' resource ID 900 from DemoDialogs:

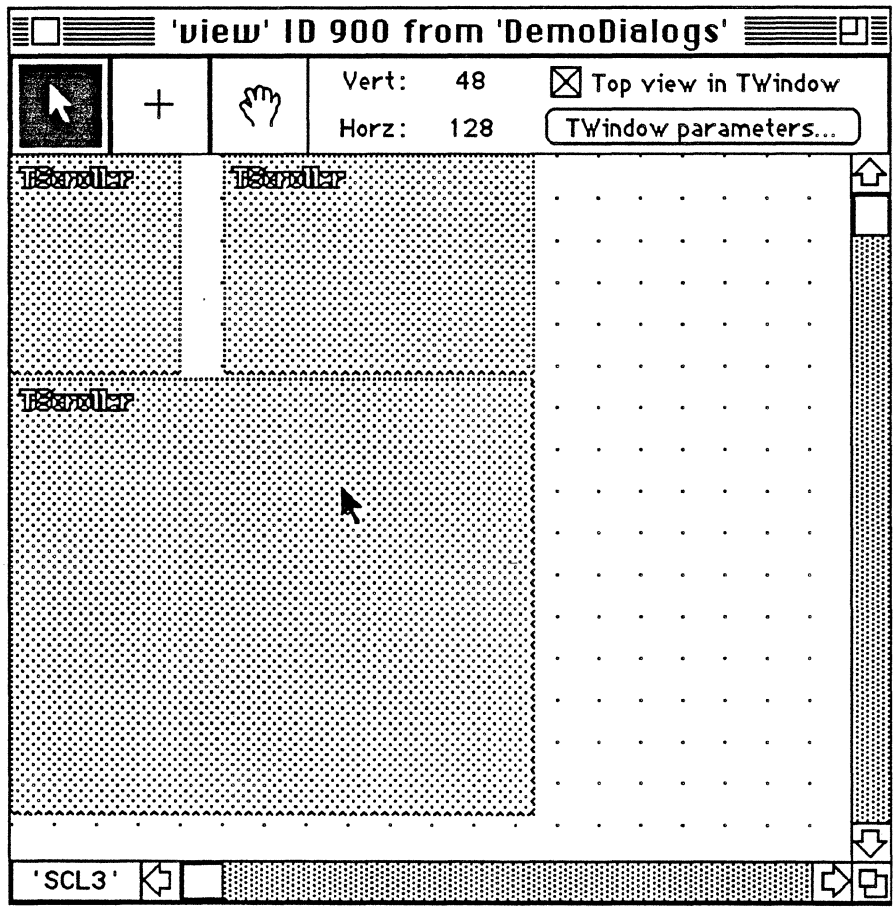


Fig 1.18 TScroller views

©1988 Apple Computer

This view-editing window represents the views in an Inspector window, which looks like this:

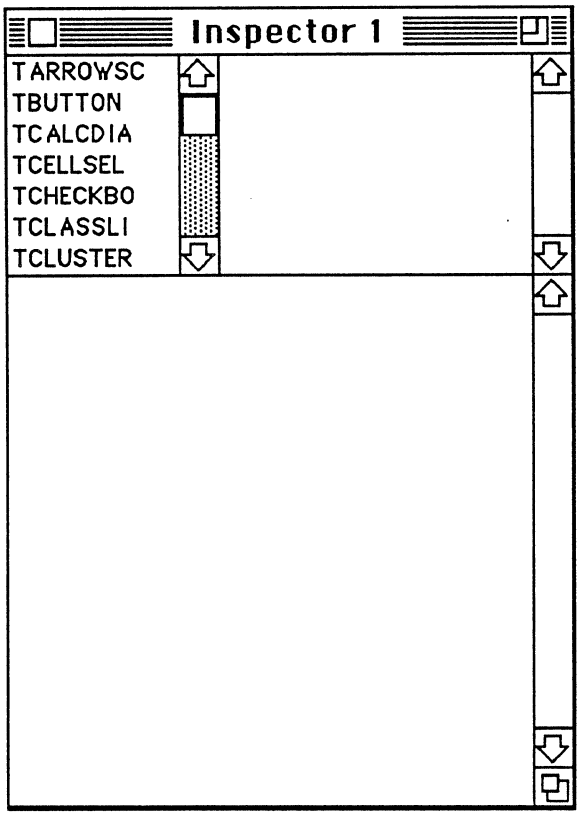


Fig 1.19 An Inspector window

As you can see, space has been left for the TSScrollbar objects, but they are not themselves represented in the view-editing window.

• **NOTE** • Since TScrollers will create up to two TSScrollbar objects for you, the TSScrollbar class is not represented in the Views menu. This means you cannot create TSScrollbar views from within ViewEdit. Use DeRez and Rez to add extra TSScrollbar views to your view hierarchy.

The file commands

When you have finished editing with a view-editing window, you can close that window. If you have made any changes, the 'view' icon for that window will have its identifier highlighted. This is a marker to help you remember which resources have been edited. Of course, none of your changes are saved until you choose the Save command from

 ©1988 Apple Computer

the File menu.

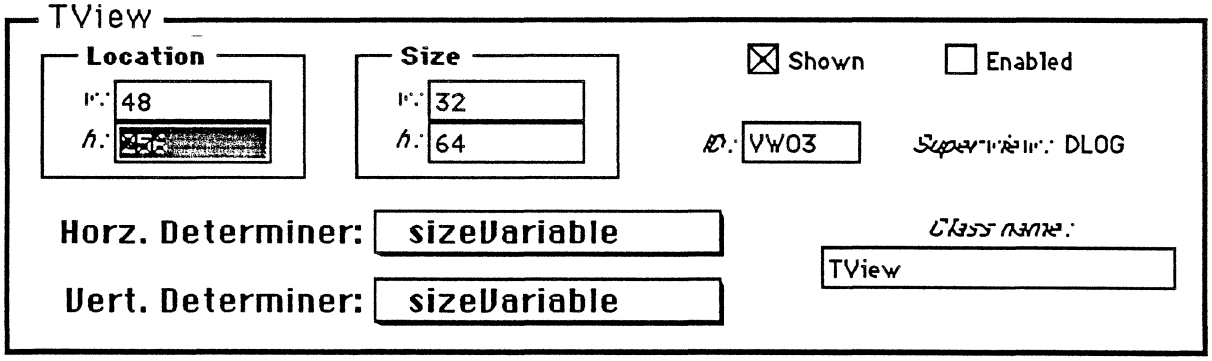
You can also mark items for deletion by selecting them, and then choosing the Delete Resource command from the Edit menu. Again, the resource is not actually deleted until you save the file.

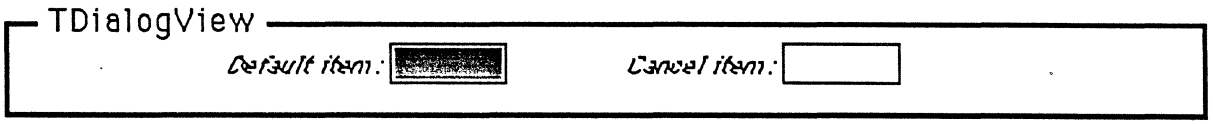
Remember that ViewEdit saves its view hierarchies as resources in a resource file. You can access these through DeRez and Rez, but it should be possible for you to only use ViewEdit, given that you have the patience to use it carefully through this early release.

Good luck and happy view editing!

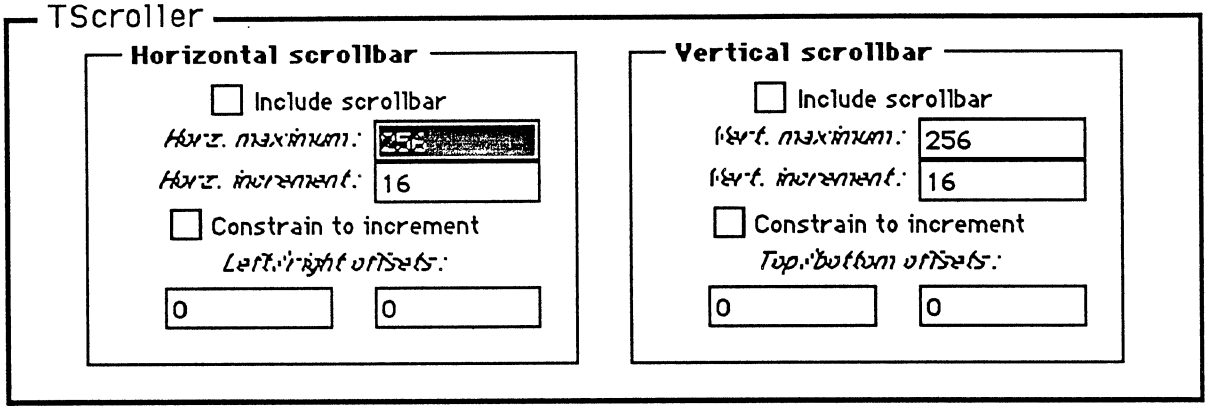
View Description Boxes

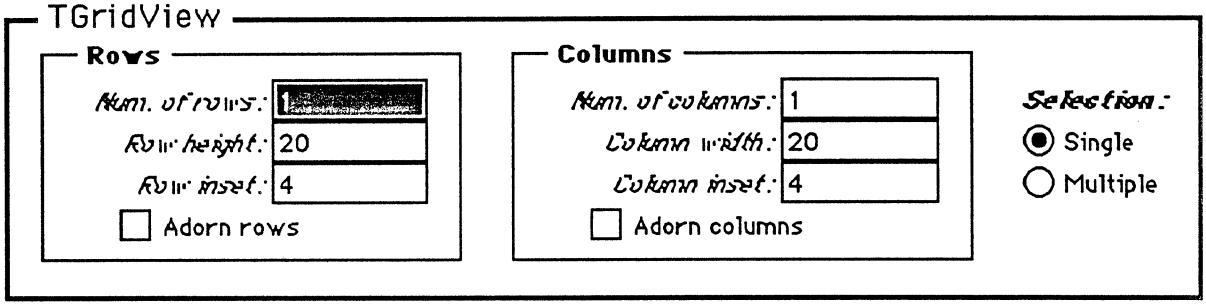
This section contains an example of each of the different types of view description boxes, in the order that they appear in the Views menu. Refer to the "Creating View Templates" recipe in Chapter 7, "The CookBook", in the *MacApp 2.x Manual (Interim Version)*, and the MacApp 2.0 Display Architecture Release Notes for a description of the different fields and possible values.

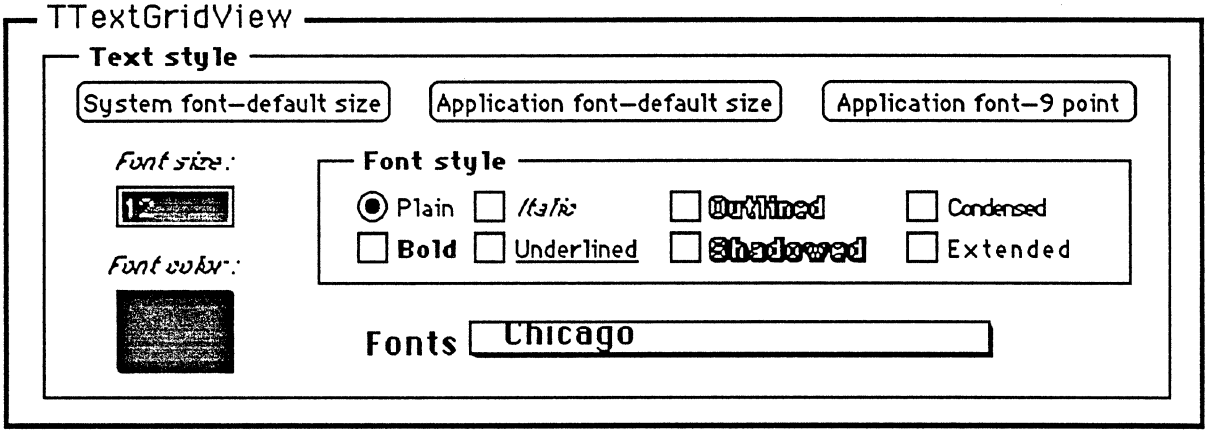




 ©1988 Apple Computer





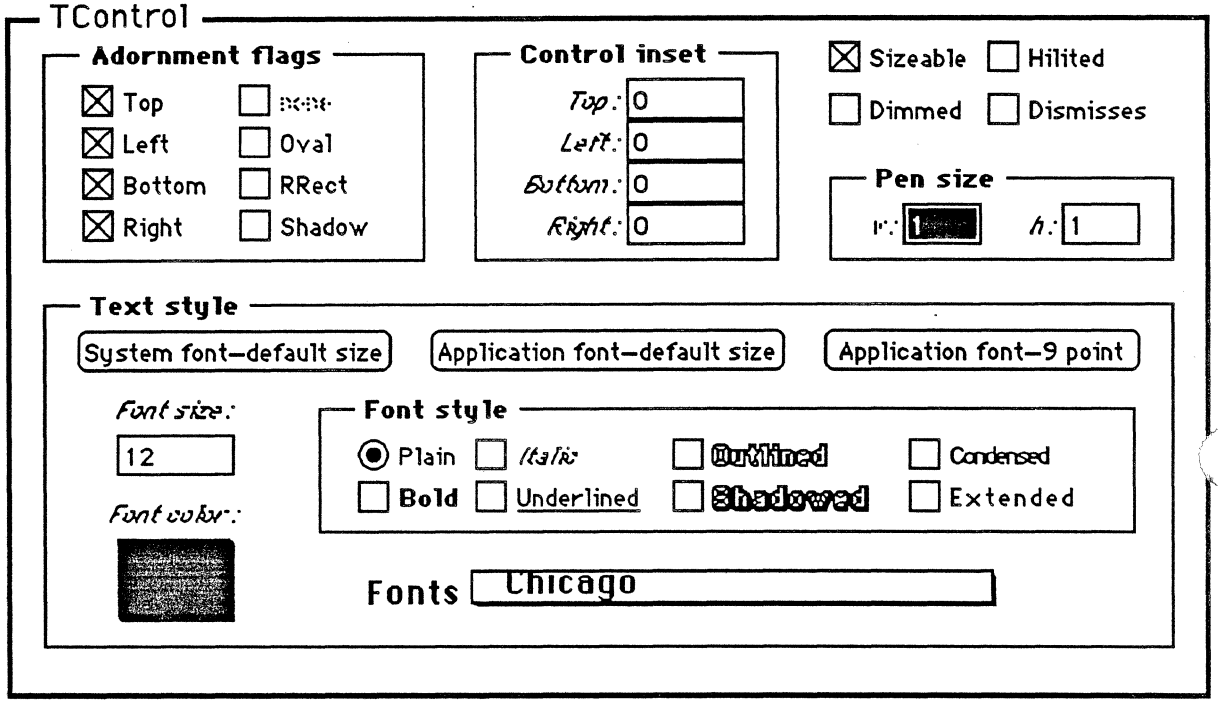


©1988 Apple Computer

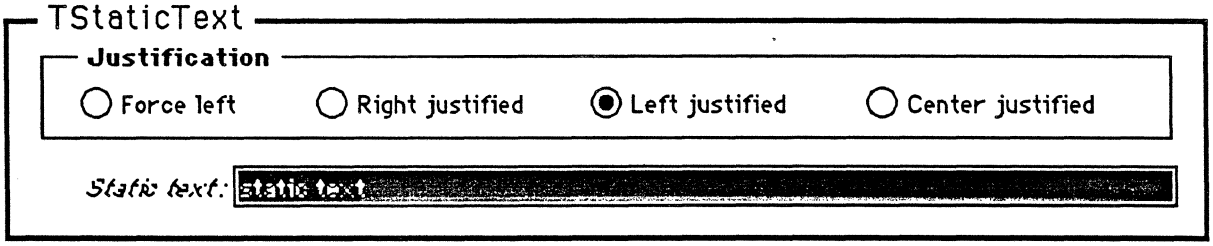
TTextListView

A TTextListView has no additional fields over a TTextGridView. Normally, you'll want to set the number of columns to 1, and the horizontal SizeDeterminer to sizeIsSuperView (???).

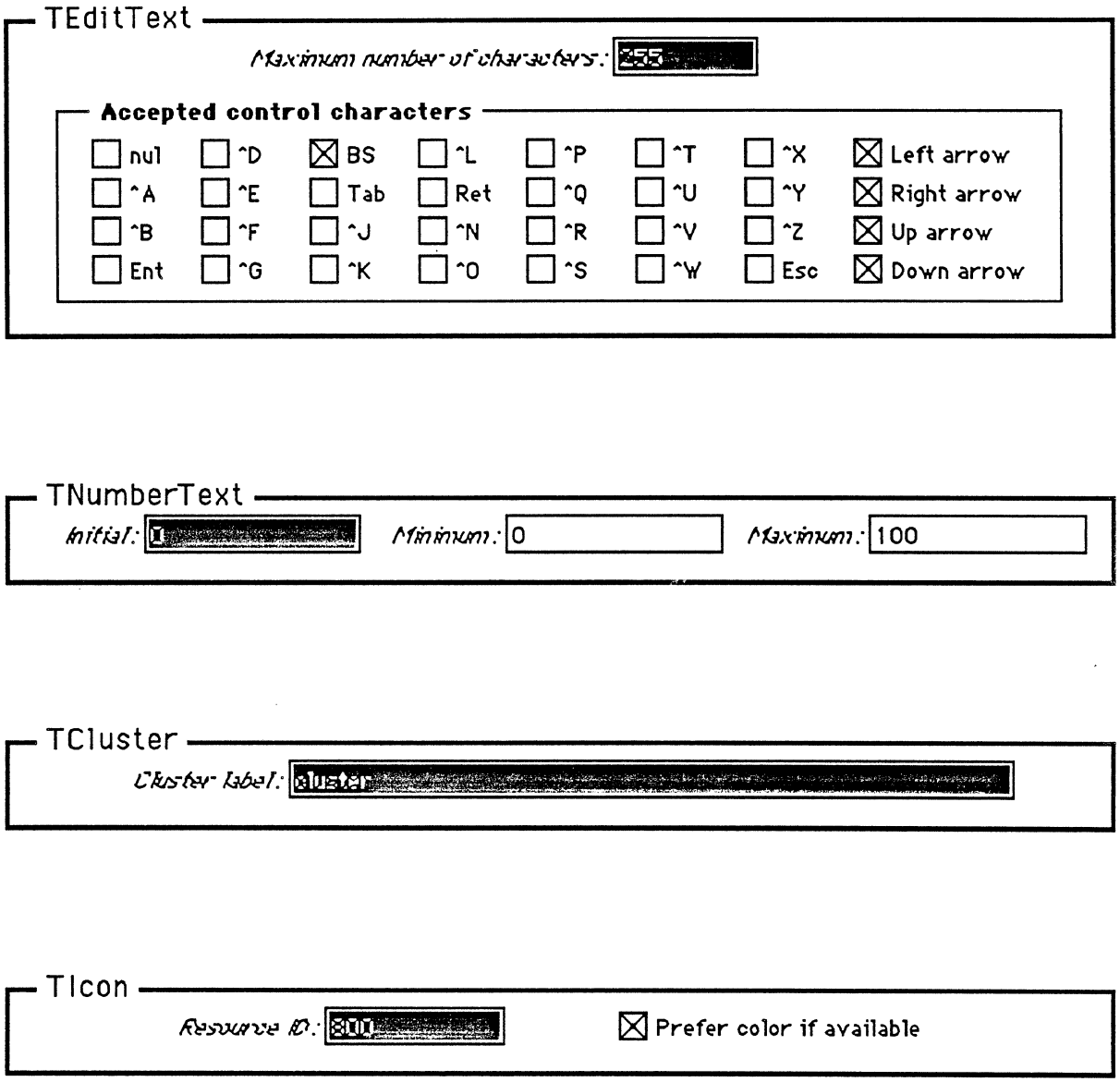
TControl



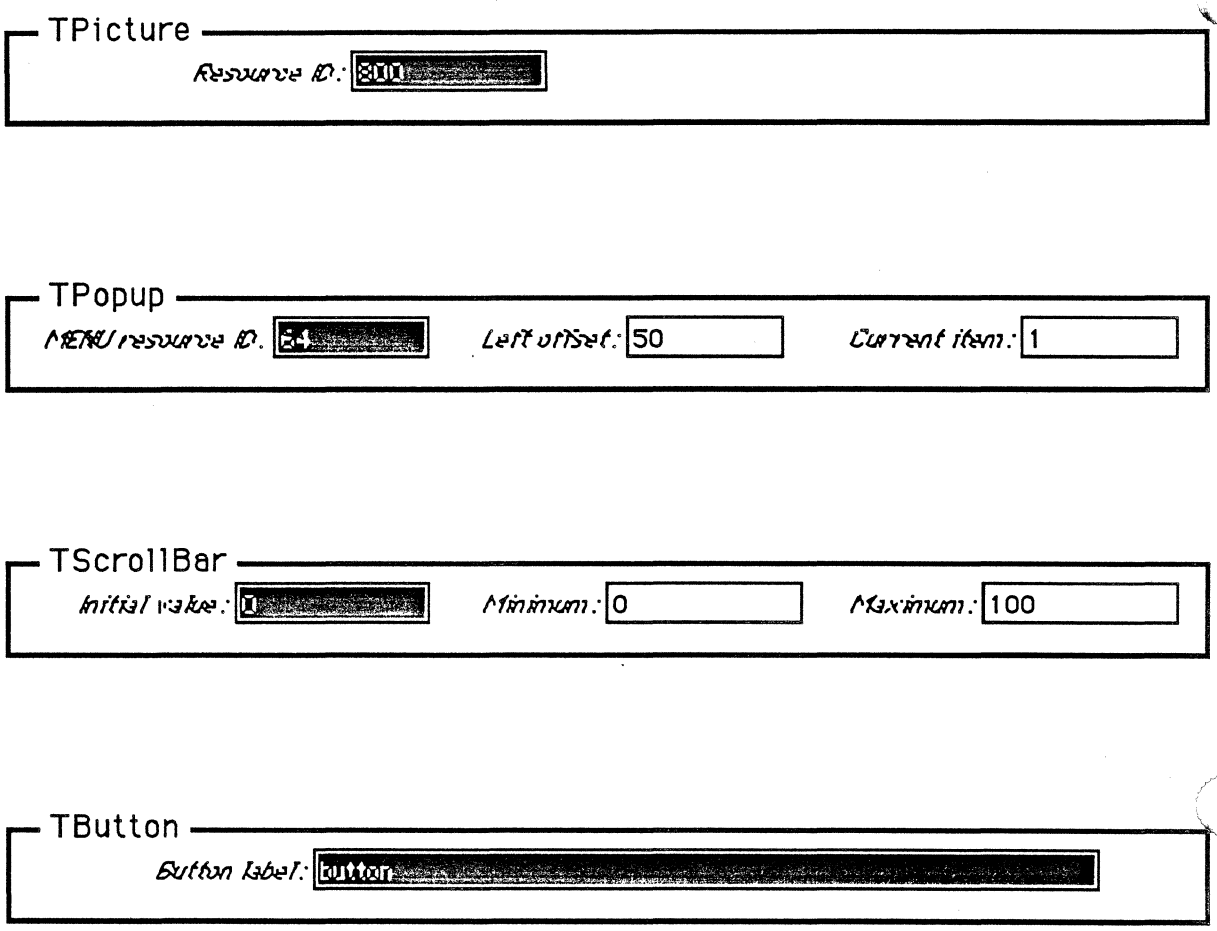
TStaticText



©1988 Apple Computer



©1988 Apple Computer



©1988 Apple Computer