Copyright ©1996 by NeXT Software, Inc. All Rights Reserved.

WhenInterfaceBuilderStartsUp;¬When Interface Builder Starts Up

When you open a nib file, Interface Builder displays several windows and panels on your screen.

The palette window Holds all currently loaded palettes of objects. You select a palette by clicking its icon. Then you drag objects from the palette to the appropriate surface.

When InterfaceBuilderStartsUp1.eps ¬

The interface window This window or panel displays the actual interface that you're working on. If this is the first time you've opened a main nib file in Project Builder, an empty window is displayed.

WhenInterfaceBuilderStartsUp3.eps ¬

The nib file window This window contains multiple views that display the contents of the nib file. These views show archived objects, the connections among objects, the current class hierarchy (including any custom classes that you may have created), and the imagesand soundsstored in the nib file. Click a tab to switch the view.

458562_WhenInterfaceBuilderStartsUp2.eps ¬

The Inspector panel This multiform panel displays the attributes, connections, and size of a selected object. It also presents the object's resizing characteristics, its associated help, and which class it inherits from.

WhenInterfaceBuilderStartsUp4.eps ¬

You can control whether the palette window and the Inspector panel appear when Interface Builder starts up by checking the appropriate boxes in the Preferences panel.

28012_TableRule.eps ¬

SavingtheNibFile;¬Saving the Nib File

An UNTITLED nib file window appears for each newly created nib file. After you make changes to an

interface, remember to save the nib file. Choose Save from the Document menu and specify a path and file name in the Save panel. Interface Builder may ask if you want to insert the file into your project; confirm by clicking Yes.

_SavingNibFile.eps ¬

918348_TableRule.eps ¬

What'sinaNibFile;¬What's in a Nib File

Every application has at least one nib file (actually a file package). The main nib file contains the main menu and often a window and other objects. An application can have other nib files as well. Each nib file contains:

Archived Objects Encoded information on OPENSTEP objects, including their size, location, and position in the object hierarchy (for view objects, determined by superview/subview relationship). At the top of the hierarchy of archived objects is the File's Owner object, a proxy object that points to the actual object that owns the nib file. (For a description of File's Owner, see the concept summary on File's Owner, First Responder, and Font Manager in Chapter€4. ;../04_Connections/ConnectionsConcepts.rtfd;StandardObjectsintheInstancesDisplay:File'sOwner,FirstResponder,andFontManager;¬)

Sounds and Images Any sound or image files (TIFF or EPS) that you drag and drop over the nib file window.

Class References Interface Builder can store the details of OPENSTEP objects and objects that you palettize (static palettes), but it does not know how to archive instances of your custom classes since it doesn't have access to the code. For these classes, Interface Builder stores a proxy object to which it attaches class information.

Connection Information Information about how objects within the object hierarchy are interconnected. Connector objects special to Interface Builder store this information. When you save the document, connector objects are archived in the nib file along with the objects they interconnect.

When You Load a Nib File

In your code, you can load a nib file by sending the NSBundle class the message **loadNibNamed:owner:** or **loadNibFile:externalNameTable:withZone:**. When you do this, the following things happen:

- The run-time system unarchives the objects from the object hierarchy, allocating memory for each object and sending it an **initWithCoder:** message.
- It unarchives each proxy object and queries it to determine the identity of the class that the proxy represents. Then it creates an instance of this custom class (**alloc** and **init**) and frees the proxy.
- The system unarchives the connector objects and allows them to establish connections, including connections to File's Owner.
- As the final step, the run-time system sends **awakeFromNib** to all objects that were derived from information in the nib file, signalling that the loading process is complete.

nib_filed.eps ¬

241701_TableRule.eps ¬

SelectingMultipleObjects;¬Selecting Multiple Objects

You can select multiple objects and then move, copy, or do other things with them as a group. There are two ways to select more than one object:

- Hold down the Shift key while you click objects in succession.
- Click in an empty area, then draw a ªrubberbandingº rectangle around all objects you want selected.

After making the selection, press (**don't** momentarily click) the mouse pointer on one of the objects and drag the group to the new location. (Or do another suitable operation, such as copy and paste.)

To deselect an object in a grouped selection, hold down the Shift key and click that object.

You cannot do sizing operations on multiple selected objects.

To select all objects in a window or panel, first select the window or panel, and then choose Select All from the Edit menu. You can also use this command to select all items in the Instances or Classes display of the nib file window. The key equivalent for Select All is Command-a.

323619_TableRule.eps ¬

WherePaletteObjectsGo;¬Where Palette Objects Go

You put items from the Views, TabulationViews, and DataViews palettes anywhere within the bounds of a window or panel. These items include buttons, labels, fields, boxes, text fields, scroll views, browsers, and custom
views.

_WherePaletteObjectsGo1.eps ¬

You can put windows and panels anywhere in the work space. Nothing contains them except the screen.

_WherePaletteObjectsGo2.eps ¬

You drag a menu item from the Menus palette and drop it in the application's menu. When you release the mouse button, Interface Builder inserts the item between the two menu commands underneath it.

_WherePaletteObjectsGo3.eps ¬

You drag a formatter from the Formatters palette and drop it on a text field or a cell in a tableview. That formatter will control the formatting for all of the cells in that column.

_WherePaletteObjectsGo4.eps ¬

Some palettes, like the one for the Enterprise Objects Framework, carry objects that do not appear on an interface. These usually are controller objects that perform management or computational functions. Drop these objects anywhere on the Instances display of the nib file window.

_WherePaletteObjectsGo5.eps ¬

577195_TableRule.eps ¬

TheCoordinateSysteminInterfaceBuilder;¬The Coordinate System in Interface Builder

The Size display of an object's Inspector panel shows that object's precise location and dimensions. The **x** and **y** fields hold the origin point (horizontal and vertical) for the object within the drawing context of its enclosing window, panel, or superview. The **w** and **h** fields hold the width and height. All values are in pixels.

Within a window or panel, the lower left corner is origin 0,0. This is the point of reference for objects within that window or panel.

Therefore, when you move or size objects downward or to the left, the values in the Size display are decreased.

The point of reference for a window or panel (or origin 0,0) is the lower-left corner of the screen. This means that the same relationship applies: if you decrease its **x** value in the Size display, it moves to the left; if you decrease its **y** value, it moves toward the bottom of the screen.

_CoordinateSystemInterfaceBuilder.eps ¬

298408_TableRule.eps ¬

CopyingObjectstoOtherInterfaces;¬Copying Objects to Other Interfaces

To copy objects to different nib files, simply select a group of objects in one nib file and Alternate-drag those objects to the appropriate ªsurfaceº of the other nib file.

You can copy entire windows or panels as well as custom, non-UI objects between interfaces.

The surface you drop objects onto must be compatible:

- Non-UI objects must be dropped over the nib file window.
- View objects are dropped over a window or panel or over the nib file window.
- Windows and panels must be dropped over the nib file window.

The basic technique of Alternate-drag also copies the connections among selected objects. See ªCopying interconnected objectsº in Chapter 4 for details. ;../04_Connections/CopyingInterconnectedObjects.rtfd;;¬

840722_TableRule.eps ¬

NeXT'sBasicUIDesignPhilosophy;¬NeXT's Basic UI Design Philosophy

Composing a user interface involves much more than techniques for placing, sizing, and arranging objects on a window. When you put your application's UI together in Interface Builder, keep in mind the following principles that NeXT has developed, through trial and test, to guide interface designers.

Make It Consistent

When all applications share the same basic interface, each application benefits. Consistency makes each application easier to learn, and so increases the likelihood of acceptance and use. Just as with so many natural ªinterfacesº in life, conventions count for a great deal. Although different applications are designed to accomplish different tasks, they all share, to some degree, a set of common operations such as selecting, editing, scrolling, and setting options. Reliable conventions are possible only when these operations are carried out the same way for all applications.

Make it Feel Natural

Try to make the screen a visual metaphor for the real world, so that the objects in it reflect the way the represented things actually behave. That's what an ªintuitiveº interface is - it behaves as we expect based on our experience with objects in the real world.

Modeled objects don't have to mimic every detail of their real counterparts, but they should behave in similar ways. For example, objects in the real world stay where we put them; they don't disappear and reappear again, unless someone causes them to do so. Users should immediately recognize the objects in your interface and should use them for the sorts of operations that people typically use their real counterparts for.

Put the User in Charge

Users should have the widest freedom of action. If an action makes sense, your application should allow it. In particular, avoid setting up arbitrary modes, periods during which only certain actions are permitted. On occasion, however, modes are a reasonable way of solving a problem, particularly in these forms:

- attention panels
- modal tools

- ªspring-loadedº mode (while mouse or key down)

But these modes should be freely chosen, provide an easy way out, be visually apparent, and keep the user in control.

At the same time, you should try to anticipate what users will do and ease their way, reducing the actions they must perform. Give them freedom, but still act on their behalf without waiting for their instructions. These helping actions should be simple and convenient, like, in the Open panel, preselecting a directory that is probably in the path of the final selection.

Focus on the Mouse

The mouse is the most appropriate instrument for a graphical interface. The keyboard is principally used for entering text, but the mouse is the instrument by which users manipulate the objects of your interface. Your user interface should support three paradigms of mouse action:

- Direct manipulation
- Targeted action
- Modal tool