PowerBuilder™

vs.

NeXTSTEP™ Development Environment

This document gives an overview of the *PowerBuilder* product from Powersoft Corporation. Specifically, it compares and contrasts it to the NeXTSTEP development environment (Release 3.0). It should be noted that the information contained within this document was obtained from a *PowerBuilder Product Overview*, published in April of 1991, based on Version 1.0 of *PowerBuilder*. There may be more current versions of *PowerBuilder* available on the market today, with enhanced functionality. However, based on conversations with potential customers, it appears that the basic structure of the product has not changed, and can thus be assumed that the functionality has not significantly changed.

What is PowerBuilder?

In their own words, PowerBuilder is a PC-based, graphical, client/server application development environment. It currently runs under Windows 3.0 and 3.1. Like InterfaceBuilder, PowerBuilder comes with a graphical environment to allow the developer to paint application objects such as windows, menus, and command buttons, which respond to events such as "clicked" or "modified" by executing scripts written in a high-level scripting language called PowerScript™. PowerBuilder supports several relational database management systems, including:

- SQL Server from Microsoft and Sybase Corporation
- SQLBase from Gupta Technologies, Inc.
- ORACLE Server from Oracle Corporation

The PowerBuilder application is much like InterfaceBuilder of 2.0, in that it includes several components of application development besides GUI construction. For example, PowerBuilder includes:

- The PowerScript language for writing application functions
- A full-function text editor for creating PowerScript routines
- A graphical PowerScript debugger with the same type of capabilities as the NeXTSTEP 3.0 GNU Debugger-Edit combination

- A Library Manager for managing the current project, and "objects" from all applications developed
- Database creation and maintenance facilities
- Automatic Application Reports
- Access to Windows Help Facility

Most of these items will be described in more detail below. And of course PowerBuilder supports the test phase during development. From all indications, you compile the scripts first, and then run the program. However, you can continue in the same development environment throughout the lifecycle of the project like you can with InterfaceBuilder. Unlike HyperScript in HyperCard, the end application is bundled into a ".exe" file and can be run as a standalone application outside of PowerBuilder, needing only the PowerBuilder runtime libraries installed on the machine.

Driving PowerBuilder

The main interface to PowerBuilder is the *PowerPanel* pictured below:

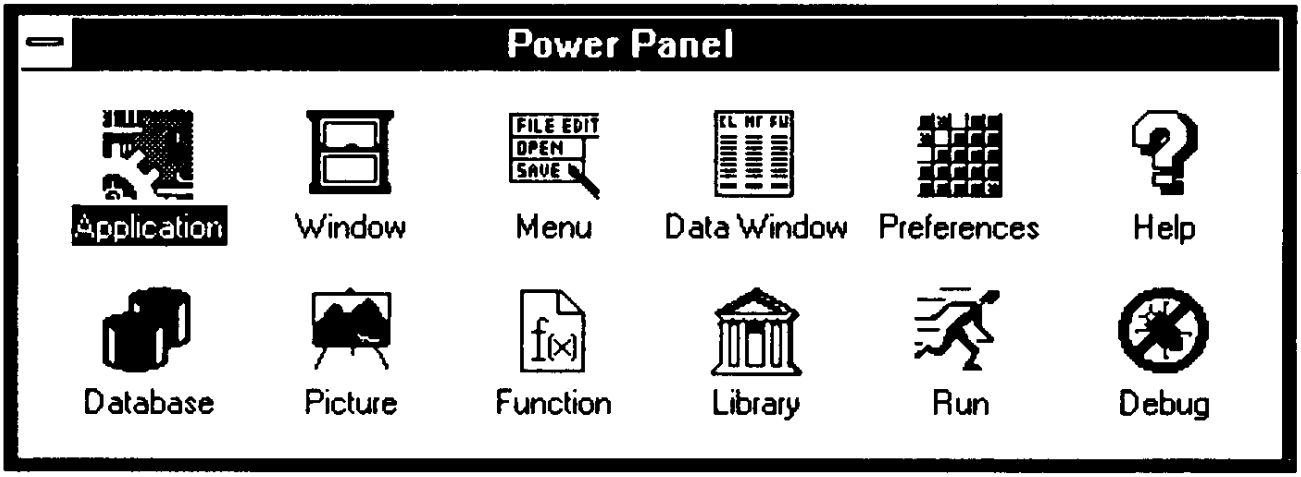

***Figure 1*. The Power Panel painter selection window.**

The PowerPanel contains icons for each of the PowerBuilder *painters*, used in the application development process. The user simply double-clicks on one of these icons to enter that particular painter. For example, double-clicking on the *Run* icon will cause your application to run.

Similarly, *Debug* will put you into the debugger environment. Complete interactive help and reference information for the application and all of its tools is available through the *Help* icon.

PowerBuilder Application Objects

There are two major types of objects in a PowerBuilder application: **windows** and **menus**. Windows are created using the Window Painter. The objects that can be placed inside of a window are any of the objects supported by Microsoft Windows, along with some additional objects for things like database access and updates. Examples of objects that can be used inside of a window and their NeXTSTEP counterparts are:

- CheckBox - toggle button
- CommandButton - button object
- DataWindow - PowerBuilder window for SQL database display
- DropDownListBox - like a pull-down menu
- GroupBox - like the NeXTSTEP Box object
- Horizontal and Vertical scrollers
- Line - a simple graphical line element
- ListBox - like a single column browser, allowing for multiple selection
- MultiLineEdit - small subset of a ScrollView object
- Oval - simple graphical element, outline or filled
- Picture - like a DBImageView, **only** displays .BMP formatted bitmaps
- RadioButton
- Rectangle - yet another simple graphical element
- RoundRectangle - for those who prefer a little more zip in the interface
- SingleLineEdit - a TextField
- StaticText - a Title TextField

The way PowerBuilder works is that you assign a particular PowerScript routine to an object/event combination. These so-called *objects* do not talk to each other, or other objects, they simply invoke application scripts. This capability is much more similar to something like Sun's DevGuide product, where a user paints pictures on the screen, and then assigns "callback" functions for each widget based on a particular event. For example, a *CommandButton* object has a *Clicked* event, and thus a script can be attached to that combination.

As far as menu objects are concerned, they behave just as

CommandButtons, and thus invoke PowerScript routines when clicked upon. The menus are hierarchical, as in NeXTSTEP, and can have shortcut keys assigned to them. The difference is that menus are attached to a particular window, as in the Windows envrionment. Thus, each window has its own menus assigned to it.

The user also has the ability to modify certain attributes of each object, such as size and location, etc.

PowerScript Language

PowerSoft loves to refer to PowerScript as object-oriented, yet it contains none of the features of an object-oriented language. It is a rather simple but thorough scripting language, with many conditional, looping, branching, case selection, etc. features similar to BASIC. Along with basic assignments and calculations, the language provides facilities for processing character strings, converting data types, working with date and time values, performing DOS text file I/O, sending output to a Windows printer, and support of DDE. In order to support database access, PowerScript provides for embedded SQL statements.

In PowerScript, data variables are available at three levels. *Local* variables are known only within the script which they are declared; *shared* and *instance* variables can be accessed by any object within the current window; and *global* variables are known throughout the entire application. Whatever happened to the concepts behind object-oriented design and the encapsulation of data?

The NeXTSTEP advantages are fairly obvious here. Although the PowerScript language is an easier environment to learn for non-programmers, real developers will quickly run into limitations of the language and its environment. For example, there is support for drawing bitmaps out of PowerScript, but not doing live drawing as NeXTSTEP can with PostScript and Interactive Renderman. There is no support for more sophisticated multi-media capabilities such as sound and video. The language does not allow easy access to operating system operations, as you can do in a "C" environment. And most important of all, PowerScript has not brought the developer any closer to an object-oriented environment than X windows and X toolkits have in the UNIX world!

PowerBuilder Painters

Database Painter - This tool provides interactive facilities for the creation and maintenance of SQL databases. Essentially, this tool is a superset of DBModeler. This tool attaches to the database and displays the tables currently defined in the database. The user can then view the columns of any selected table, as well as any indices defined for the table. The user can also do functions such as:

- Create new tables in the database
- Create a database view, joining tables together by clicking on foreign and primary key column names to indicate the relationships between tables
- Specify grouping and sorting for each view
- Create users in the DBMS and assign access privileges
- Define intial values for columns
- Define input validation rules for each column
- Define output display formats for each column

The figure below shows a sample screen of view definition in the Database Painter, showing 3 joined tables.

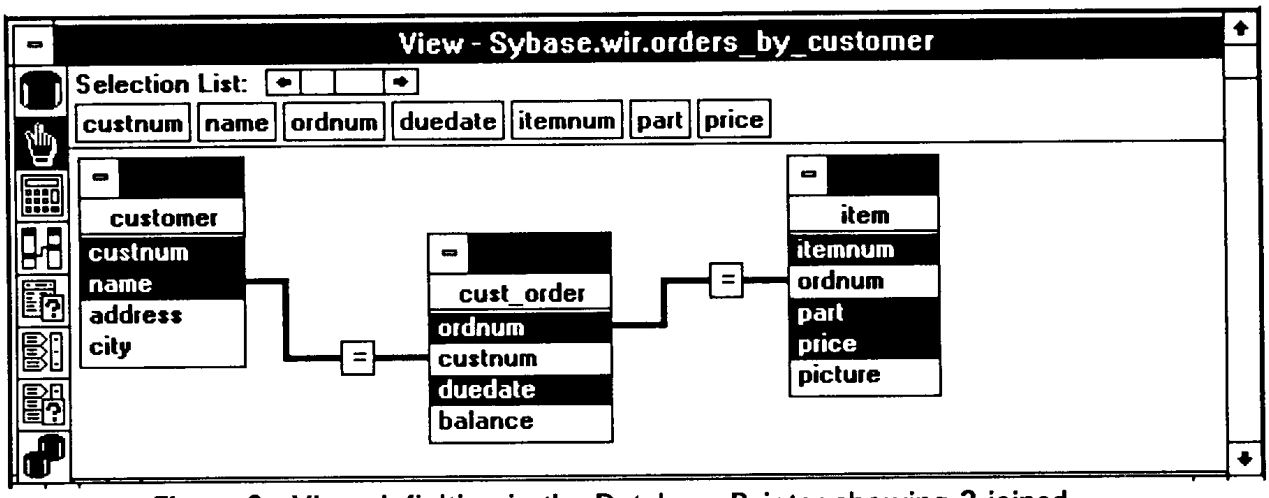

Figure 2. **View definition in the Database Painter showing 3 joined tables.**

Essentially, DBModeler allows the same capabilities to create relationships in a drag-and-drop fashion, and to create logical table and column names. In NeXTSTEP, many of the functions listed above are done at the

application level, or inside InterfaceBuilder. For example, the Database Painter allows you to specify which columns should be visible in a particular database view. In NeXTSTEP, this would be defined in the user-interface of your application, when deciding which columns should show up in something like a DBTableView object. Another example is the ability to specify initial values, input validation, and output formatting. This can all be accomplished in InterfaceBuilder with a combination of the appkit's objects, and something like SmartFields from Objective Technologies. As far as database creation and maintenance is concerned, one could argue this should be a feature best separated from the development environment. Typically the Database Administrator is not the same individual as the corporate developer(s), and does not want to give that type of access to the developer staff. In this case, a separate application such as DBCommander is a much better solution. (Note: DBCommander is a 3rd party NeXTSTEP tool for the creation and maintenance of databases, specifically Sybase.)

Application Painter - This painter allows you to define such things as your application name, application icon, default type styles, etc. You also define which library in PowerBuilder your application and its objects should be stored in, as well as which libraries should be in your search path for accessing objects from other libraries.

The application you develop can also have an *open* and *close* script, similar to appDidInit, and appWillTerminate. These scripts are created in this painter.

Window Painter - This is the tool for allowing the developer to create the windows needed by the application, and creating the objects that go inside of the window. You can also create *open* and *close* scripts for each window. For each window you can set attributes such as size, placement, title, window controls, etc. Below is an example of a window being defined in the Window Painter:

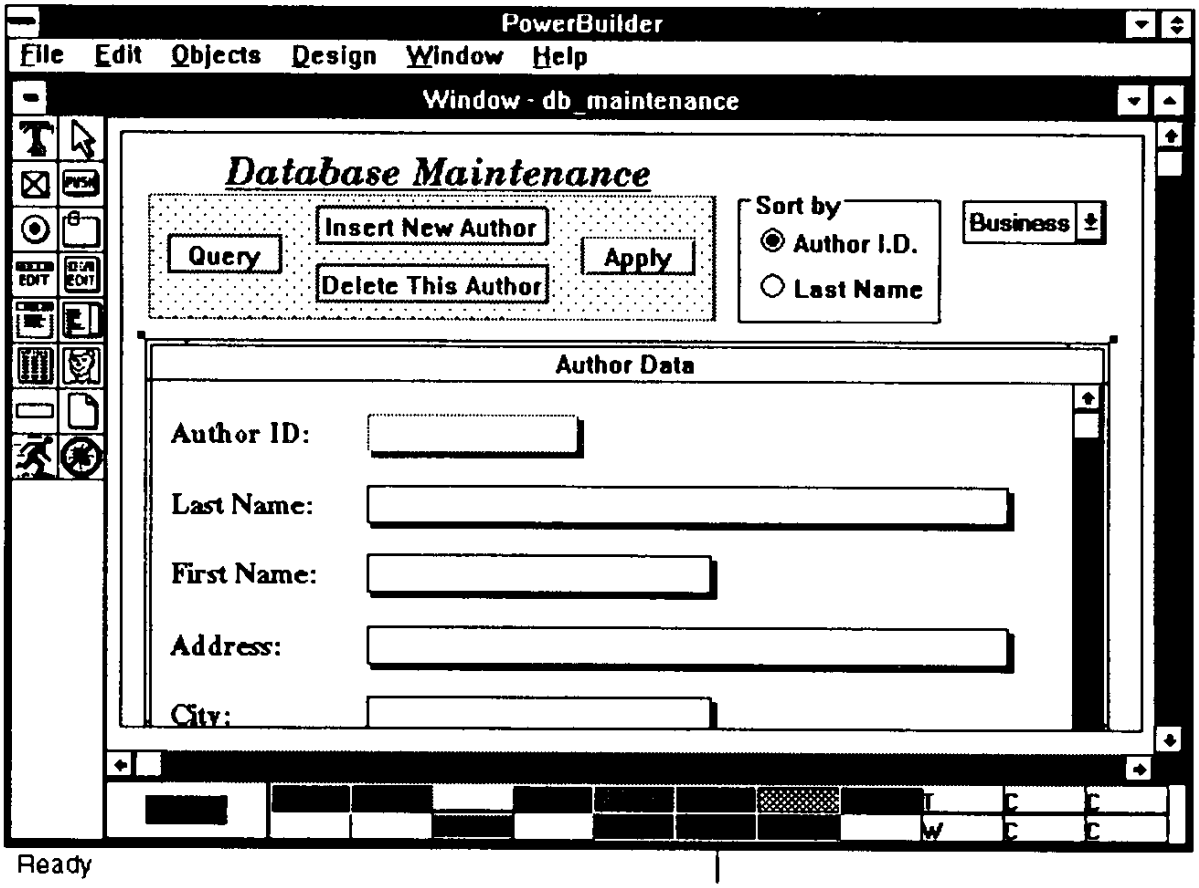

Figure 4. **Window definition in the Window Painter. The window being created includes a DataWindow for displaying author data, several CommandButtons, RadioButtons, StaticText, and a DropDownListBox.**

Menu Painter - This is the tool for creating menus and assigning PowerScript routines to them.

DataWindow Painter - This tool allows for the creation of the DataWindow object, specific to PowerBuilder. It can be thought of as a simple combination of a DBModule and DBTableView. These objects can display data from a database in tabular or free-form layout, just as NeXTSTEP

allows you to display data in things like DBTableViews, or a free-form layout of different TextFields and other objects. The DataWindow object handles requests to retrieve information, insert, update and delete. DataWindows respond to PowerScript functions that your routines invoke, such as:

- **Retrieve** - retrieve data from Database
- **Print** - print the contents of the DataWindow to the printer
- **Update** - save all changes made in the DataWindow, including inserts and deletes
- **Sort** - sort the contents (this would be a nice feature for a DBTableView to have, *hint - hint*)
- **Save as** - instructs the DataWindow to save its contents to disk in one of several formats, including 1-2-3 format, Excel format, and dBASE format.
- **Scroll** functions

The figure below illustrates a DataWindow inside of a window object.

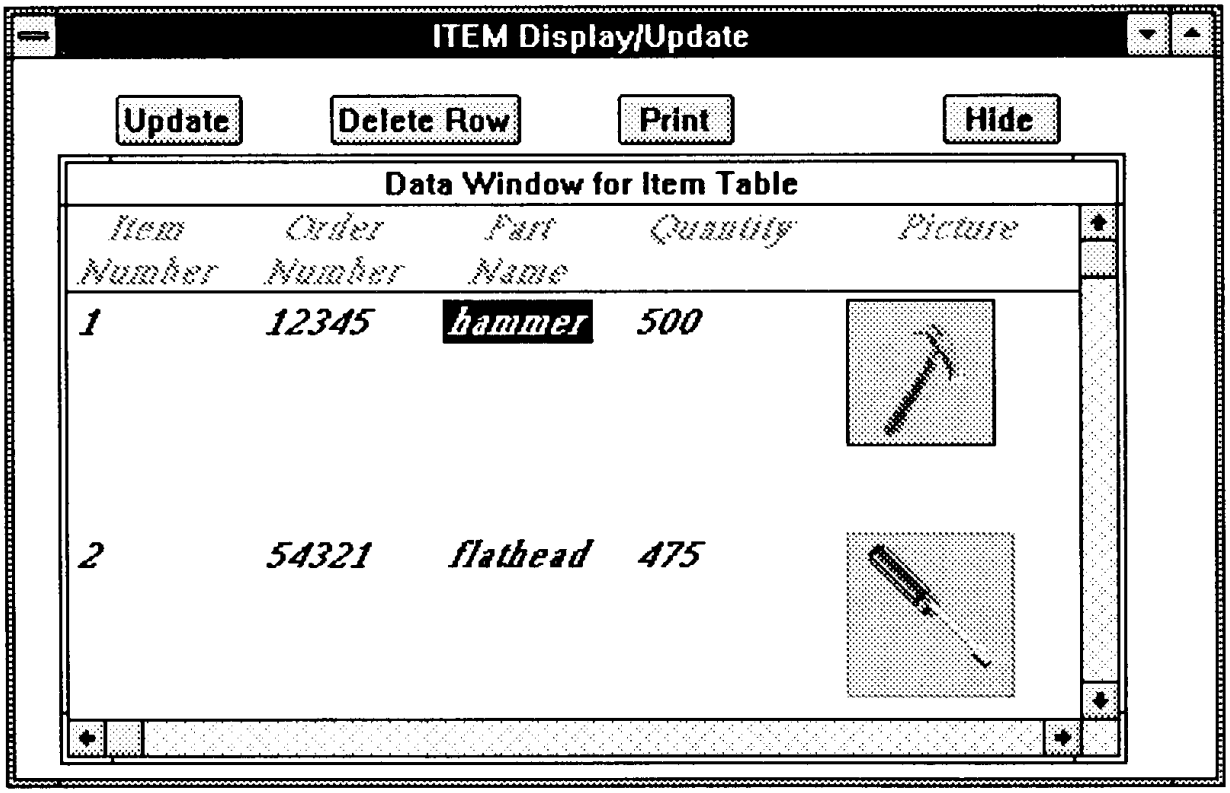

Figure 5. **Transaction window providing retrieval, scrolling, updating, deleting, and printing of data in the *Item* table. The *Hide* button lets the user remove the DataWindow from the screen. The entire transaction requires only 5 PowerScript commands.**

To create the DataWindow, you graphically choose the tables, columns and selection criteria that determine the relation to the data in the database. You must choose the results to be displayed in either tabular format or free-form. You can reposition the data, as you can in a DBTableView object where you can change the order of columns. You can also define sort rules for the data, as well as input validation, and add calculated fields. Data retrieved from a database can be displayed in a single- or multi-line edit box, a drop-down list box, a check box, or a group of radio buttons. For any column, you can supply a translation list that will convert the data from what is stored in the database to what is displayed in the application.

Picture Painter - This tool allows you to create bitmap images to be used in your application. They must be of .BMP format. You can set a drawing tool as the default in PowerBuilder's preferences, such that any time you want to edit a picture, PowerBuilder will launch that tool with the picture to

allow you to begin editing. Picture Painter also allows you to draw freehand pictures, or scan in images.

Function Painter - This tool allows you to package PowerScript calls into user-defined functions. These functions can then be called from any PowerScript routine. A powerful extension to this capability is that Function Painter also allows you to declare as an External Function any function that resides in a Windows Dynamic Link Library (DLL). You can then call these external functions from any PowerScript routine. This opens up the PowerBuilder environment to more functionality.

Script Painter - This is a full-function text editor integrated into PowerBuilder for the creation of the PowerScript routines. It includes context-sensitive help on any PowerScript function, and the browsing of all functions from a selection list. It also performs the compilation of these scripts, and highlights errors detected during this process as does NeXTSTEP's 3.0 ProjectBuilder facility.

Preference Painter - Allows the user to set their preferences for the PowerBuilder application.

Library Management

A nice feature of PowerBuilder is the ability to store objects and scripts into different libraries, which are managed by the **Library Painter**. During your development, you can specify a library search path and order. For example, during your development you could be using some objects from a QA library, some from a departmental test library, and some from your personal development library. The Library Painter allows you to browse libraries, copy objects from one library to another, erase and rename objects.

When you need to document your application, you execute the Library Painter's *Print* command. You can choose which objects should be included in the report, and the level of detail to be reported on. It then creates the report.

Conclusion

PowerBuilder provides a well integrated environment for simple data

retrieval and update capabilities, allowing access to SQL databases through graphical connections and embedded SQL statements. Most retrieval and editing operations can be done graphically with a mouse. However, when this facility falls short, the user is placed into PowerScript and embedded SQL. This may be an easier environment to learn for non-programmers, but as in other high-level scripting languages, the functionality demanded by corporate users typically outstrips the capabilities provided by the language. The emphasis in PowerBuilder is definitely applications needing database access. The emphasis in NeXTSTEP's development environment is a complete integrated solution to application developer's needs, no matter what the focus of the application.

PowerBuilder falls woefully short in the number and type of objects accessible to the application developer. There don't appear to be any complex objects available like there are in NeXTSTEP (e.g. Font Panel, Open/Save panel, Color Panel, SoundKit, VideoKit, 3DKit, Print/Fax object, etc.). PowerBuilder is not extensible to the developer, such as InterfaceBuilder is with the ability to add new object palettes. PowerBuilder does not allow the user to choose the language they want to develop in. PowerBuilder depends on DDE, and therefore does not support network wide communication between applications (i.e. remote objects).

The most important differentiating feature between PowerBuilder and NeXTSTEP's development environment is the lack of a true object-oriented environment in PowerBuilder. The corporate developer does not gain the benefits that OOPs provides by using this tool. It is no easier to share objects in this environment as it is to share subroutines in the old "modular" code days. Data is not encapsulated within objects, but rather scripts have access to data at different levels just like the standard "C" language. There are no facilities for taking advantage of work done before, in the form of subclassing. It is not necessary to go into any detail on these issues, and the others associated with this topic, because these are the same arguments we have been using against almost all other competing environments, from Windows to UNIX. This is definitely a nice product for what it does, but it is not object-oriented, and it is not a complete development environment. It simply adds graphical database access to a scripting environment. Don't fear it, challenge it!