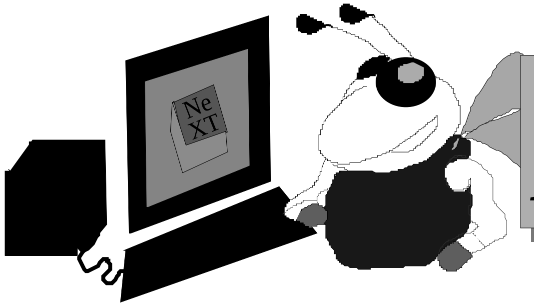
Have you asked your local NeXT rep about 3-News yet? See page 31



BuzzNUG *Buzzings*

January 1990 - Issue 2

Contents

Welcome..2
How I Learned to Stop Worrying & Focus my Monitor....................3
Put Your Program under the GNU General Public License....................4
New Goodies in the NeXT Archive at Purdue....................7
Developing an Intuitive Interface for Creating PostScript Effects 8
Using the NeXT as a Digital Scope....................16
Simple Text Browser....................25
TextArt : A First Look....................28
Feedback from the Trenches....................29
Object Bee-Line....................30
Market View....................30
User Groups....................31
Scenes From NeXT Issue....................32
Buzz's Hint Corner....................32

The BuzzNUG General Council
Erica J. Liebman, Pres. erica@kong.gatech.edu
David Rosenbaum, V.P. daver@pyr.gatech.edu
David Samuel King, Tres/Sec dsking@pyr.gatech.edu
Tamora V. Sealey, Accounts tammy@cadnext2.gatech.edu

Contributers : Erica J. Liebman, David Rosenbaum, Jacob Gore, Gerrit Huizenga., Andrew Stone, Raj B. Apte & the entire BuzzNUG membership.

All rights reserved. Entirety is copyright 1990 by BuzzNUG. Individual Articles are copyright 1990 by their authors. Authors are free to resubmit articles for publication in other media. Each issue of BuzzNUG Buzzings may be freely copied and otherwise reproduced but not altered. BuzzNUG Buzzings may not be sold for profit.

The newspaper is of necessity something of a monopoly, and its first duty is to shun the temptations of monopoly. Its primary office is the gathering of news. At the peril of its soul it must see that the supply is not tainted. Neither in what it gives, nor in what it does not give, nor in the mode of presentation, must the unclouded face of truth suffer wrong. Comment is free but facts are sacred.
C.P. SCOTT (1846—1932) Manchester Guardian, 6 May 1926

WELCOME

Erica J. Liebman

All references to Anne Tyler aside, I've been calling this one the Accidental Issue. It wasn't meant to be. I was firmly, strictly set on a bi-monthly publication. Some things are meant to be. Welcome to the 2nd edition of Buzzings. A month early. (And I thought I had problems with deadlines).

I was both delighted and surprised (pleasantly) the warm response to the first issue. People are signing up for articles for the February and March issues. (You too can get in on the fun, start writing your articles soon). The newsletter is being distributed to non-internet sources through all sorts of creative distribution strategies. A special thank you is owed to Roger and the Guys at Lighthouse (the ones making the EE-CAD package as well as that cool asteroids game, etc) for physically mailing out copies to several uunet sites who couldn't ftp.

We'll take articles for BuzzNUG in all sorts of forms. Internet .wn.tar.Z form is preferred, but ascii text via the net, IBM and Mac Disks via USMail and (yikes) written Text via the same USMail will be happily accepted. (Send to the editorial address below). I can not guarantee return of disks (except through SASE's), sorry. Fortunately non-optical disk prices are really low. Our focus is on how-to articles, especially with sample code. All articles are subject to editorial review.

We're also welcoming review copies of new software from 3rd party vendors. Again, we can not guarantee material return without SASE or guarantee a publication date for the review (if any) although we'll try to be prompt.

"Feedback from the Trenches" is open for comments/letters of limited lengths from all readers. Please write and tell us what you liked and disliked.

And finally, here's an offer you can't refuse (although many of you, I suppose, will) : Get your own free BuzzNUG Membership ID card. Features include : a picture of Buzz the Bee, a membership number of your very own (status symbol city here) and the mentioned membership card. Just send your full name, internet & real address, additional contact information (if you don't mind), and a list of topics that you'd like to see be covered here. If you are not at an internet/uucp site (that is, if you are writing directly by mail), please send a SASE for your Buzz picture and ID card. You won't get a silly message welcoming you from the founders, because here it is now : Welcome.

Editorial Matters to :
BuzzNUG c/o EJ Liebman
1150 Collier Rd NW/L-12
Atlanta, GA 30318

Note : Please send deliveries of items that will not fit in a tiny mailbox care of the Leasing Office. To contact me directly for subscription information, corrections, requests, or just correspondence, write via internet : erica@kong.gatech.edu

Dr. Strainlove
Or, How I Learned to Stop Worrying and Focus my Monitor
David Rosenbaum

This is the story of a problem with a very simple solution. When I first got my NeXT, I thought it was a wonderful machine in all respects except one: the display. I found myself getting tremendous eyestrain from using my NeXT for even short periods. The display seemed blurry, especially around the edges. I remember thinking that it was ridiculous for NeXT to make 10-point Ohlfs the standard font in Terminal when it was virtually impossible to read.

My efforts to solve the problem with my monitor were rather extensive. In retrospect, I'm somewhat embarrassed by how much trouble I took to solve a problem that was so simple to fix. In the end, I learned that my monitor was simply out of focus.

Before I go on to describe the solution, I should mention that NeXT recommends that all adjustments to the monitor be done by an authorized NeXT service representative. Unlike some more draconian companies, however, they don't insist on this; adjusting your own monitor will not void your warranty (unless you break it in the process). But I would only recommend adjusting the focus; if your monitor's problem is worse than simply being out of focus, it would probably be a good idea to take it to a service person.

With that in mind, how does one adjust the focus? First, turn off the system and disconnect all cables from the back of the monitor. Now take the NeXT-supplied tool (I frankly don't know what to call it) and remove the four screws on the monitor's back plate. This allows you to take off the monitor's plastic back casing. Removing the casing will expose the adjustment screws. There are several screws, but the only one to worry about is aptly marked FOCUS. Reconnect the cables to the back of the monitor and bring the machine up. Using a television adjustment tool, which is a long, thin slot-head screwdriver that can be purchased at Radio Shack, adjust the FOCUS screw until you are happy with what you see. Now turn off the machine, disconnect the cables from the monitor, put the plastic case back on, tighten its screws, and finally reconnect the cables.

When properly adjusted, the MegaPixel display is truly excellent. I now have no trouble using 10-point Ohlfs in Terminal; I prefer it over 12-point Ohlfs and a few other fonts I have tried due to its compactness. If you find yourself thinking that your display looks bad, I encourage you to try adjusting the focus before giving up on it.

[Always be sure to check the details of your maintenance contract before opening a machine. These instructions are directed to the capable and competent user. System defaults including fonts may be viewed and changed using the dread *and* dwrite *commands. Try* dread -l *from the Mach shell to see your current defaults. Please use caution in changing these values. One of the reasons these are kept in the "invisible" .NeXT directory is to keep them away from the casual user. Many of the defaults may be changed from the Preferences application. Other files kept here are the parameters for the current dock and the targets from the digital librarian. Knowing how to access and change the system defaults without creating temporary files to clutter the system is a powerful tool for the programmer.--EJL]*

An Easier Way to Put Your Program under the GNU General Public License
Jacob Gore <jacob@gore.com>

The 1.0 version of the NeXT system comes with several programs that were done by the GNU Project of the Free Software Foundation (FSF). The most famous ones are the C compiler, the editor Emacs and the engine of the Chess game.

The GNU General Public License

The goal of FSF is to make software sharing more commonplace. The reader is referred to FSF documents for details of the philosophy behind FSF and the strategy behind the GNU Project. These documents are normally found in the etc subdirectory of the Emacs library directory (/usr/lib/emacs/etc on the NeXT), but they are missing from the 1.0 distribution. They can also be obtained from a number of ftp archives that store GNU software, including FSF's primary ftp archive on the machine prep.ai.mit.edu.

FSF achieves its goal by placing a copyright on its software, and then providing a license that explains how that software may be distributed. This is known as "the GNU General Public License" (GPL).

The major provisions of GPL are:

•If somebody distributed (gave or sold) to you a program covered by the GPL, you are entitled to copy it and distribute it further under the same conditions;

•The source code to the software must be included in the distribution or offered either at no charge or for the cost of media, shipping and handling;

•If you modify the software, you must have a notice in the modified copy that you have modified it;

•Software that is derived from GPL-licensed software is also subject to GPL.

Every program that is covered by GPL must identify itself as such, so that the people to whom it is distributed know their rights under the license. For example, when you start up Emacs, it tells you to hit *Control-h Control-c* for this information.

GNU software in the NeXT 1.0 distribution

Hitting *Control-h Control-c* in Emacs, actually, displays the contents of the file /usr/lib/emacs/etc/COPYING, which was also omitted from the 1.0 distribution. Hopefully, this was an oversight, since this is a blatant violation of the license. Another violation was not telling us how to obtain the source code for Emacs.

The distribution of the Chess game was handled better than that of Emacs: the **Info** panel of Chess displays the GPL. So with Chess, at least, you can tell that NeXT was supposed to tell you how to obtain the source code… but they didn't. Hopefully, this also is an oversight that will be remedied in the next release.

The most important GPL-licensed software in the 1.0 distribution is the Objective–C compiler. It was derived by NeXT from the GNU C compiler, and is thus covered by GPL. This is a very commendable choice by NeXT, and it is the first large-scale software contribution to the GNU Project by a commercial company. NeXT has submitted the sources for the compiler to FSF back in October ’89, and papers that signed the copyright over to FSF were to follow. At the time of this writing (end of December ’89), the papers have still not been received by FSF.

An Easy NextStepish way to use GPL

Some of the NeXT ftp archives (including cs.orst.edu and j.cc.purdue.edu) provide a package called *GNULicense*. The package consists of:

- •GNULicense.wn, a WriteNow document containing GPL;
- •A NIB (NextStep Interface Builder) file containing a sample **Info** panel;
- •A NIB file and source code for the GNULicense and GNULicensePanel objects;
- •instructions for using the above.

The package was designed so that it could be incorporated into an application from within the Interface Builder, without having to write Objective–C code.

Installing the GPL document

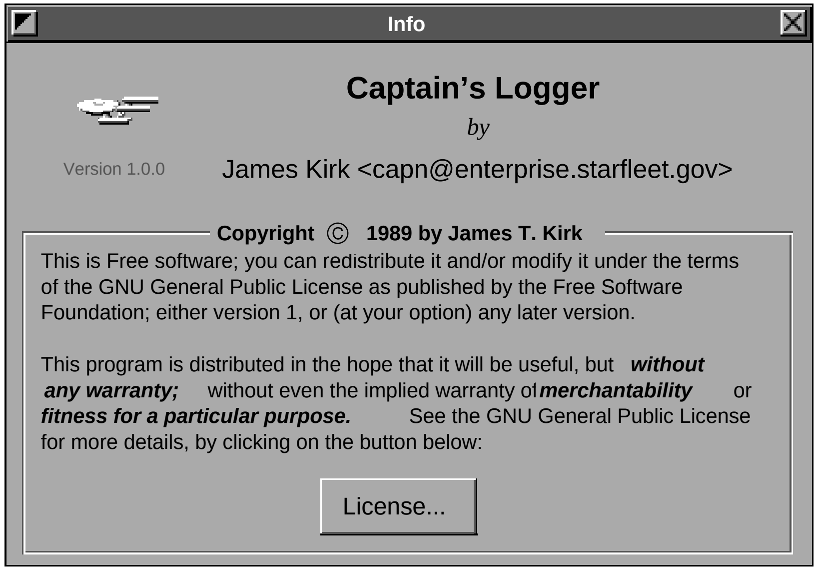
The file GNULicense.wn should be installed in either the directory /LocalLibrary/GNU or the directory ~/Library/GNU (“~” stands for your home directory). This way, all GPL-licensed programs can share it.

*The sample **Info** Panel*

While in the Interface Builder, the module SampleInfoPanel.nib can be opened. It contains the panel below. This panel can be used to replace the **Info** Panel in the application’s main module. Alternately, the Copyright box can be copied from it and pasted into the main module’s **Info** Panel.

A major goal of the *GNULicense* package was to be configurable entirely from within the standard (*i.e*., not requiring modifications) Interface Builder. Thus, only objects that are available in the standard Interface Builder were used. This accounts for some font alignment problems. Only a Text can be multifont, but the Interface Builder only provides Text within a vertical ScrollView. Thus, the text within the Copyright box was done as a TextField in Helvetica Medium font, and overlaid with TextFields in Helvetica Bold Oblique font. A similar trick was done to get the “©” symbol into the title of the box (that symbol is in the Symbol font, and Box titles must be monofont). The font changes look better in the application, on the MegaPixel screen, than they do in this document.

Objective–C is a trademark of Stepstone Corporation



The GNULicense object and the GNULicense Panel

The module GNULicense.nib was designed to be linked in with the rest of the application and loaded on demand (when a user clicks on the “**License…**” button). A GNULicense object serves as the connection between the main module and GNULicense.nib. The details of how the actual connections are made are in file README of the *GNULicense* package.

The license panel (too large to be included here) displays the copyright notice for the license itself in its header and shows the license as scrolling text. It is resizable and includes a “**Print…**” button. Printing is done by messaging WriteNow to open the previously installed GNULicense.wn document. Unfortunately, the user then has to explicitly tell WriteNow to print it (this may not be necessary in future versions of WriteNow).

Study GPL before using it

The GNU General Public License is a nontrivial legal document. The summary of it that was provided in this article is not a legal document, so before you commit your software to GPL, read “the real thing.” If your software uses GPL-licensed software, you may be obligated to put it under GPL—the conditions for this are also listed in the license.

There are ongoing discussions about the advantages and liabilities of using GPL in the *GNUsenet* group gnu.misc.discuss. Questions concerning certain points of GPL can be posted there or mailed to FSF.

What NeXT can do to help

NeXT could distribute the *GNULicense* package. It could put GNULicense.wn and an ASCII version of the license into /NextLibrary/GNU, and then all GPL-covered software could use it. This would make it easier for distributors of such software, *including NeXT*, not to violate the license. Each program would simply point to the license in /NextLibrary/GNU, and add, if it was distributed without source code, a notice about how the source code can be obtained.

Jacob Gore
December 25, 1989

[Further GNU licensing information including NIB files may be downloaded via anonymous ftp from j.cc.purdue.edu. When researching this, I did not find GNULicense.tar.Z on either Oregon or Maryland's file servers but this may change by the time you read this.]

New Goodies in the NeXT Archive at Purdue

Gerrit Huizenga

This has been a very busy period for the NeXT Archives. As many of you may know, in addition to the ftp archive on **cc.purdue.edu**, I am now offering access to the archives via email. Send a message to **archive-server@cc.purdue.edu** with a subject of **help**. Of course, if you already have access to the ftp archives, you won't find anything here that is new, but if you didn't have ftp access, the joys of public domain software are now yours to share.

Of course, this article is really about all the goodies in those archives, and there are quite a few new entries. I'm very happy to report that several of the submissions are from people at NeXT - some of those people acting in an official capacity, some sending in their weekend fun-projects. It is very good to see such participation from NeXT! Also, there is a new directory just for BuzzNUG - you can find both the current and previous Buzzings newsletters on the archive both in WriteNow and PostScript formats.

In the **binaries** directory, there is a monthly planner/appointment scheduler called Cassandra. I only played with it for a little while, but it looks fairly complete.

The only new Class definition is for a browser. I had hoped to see more Class definitions among those files submitted (and I would hope that they had a nice complete spec sheet :-), but perhaps the machine needs to mature just a bit more before we see a lot of those.

NeXT has sent out a new document describing how to *really* install Sybase. They have asked that any additions or corrections to the document be sent in to NeXT via the normal bug reporting channels. The official Chapter 14 of the SysRefMan is now on the archive server - 14_ProgDSP.wn which is the "real" 1.0 documentation on Programming the

DSP. These are both in the **docs** directory.

I have created a new directory called **lore** which will be used to store items of general information in what will hopefully be a nice, concise form. The first file describes how to get the real CMU documents on MACH. The second provides some info for people putting 3rd party disks on the NeXT - in this case, specifically the Wren V. If someone has a collection of disktab files for various hard drives, send them in - someone else can benefit from your work!

Another NeXT submission is the code to the four labs used at the Developer's camp. This are very useful to the beginner who is trying to learn Objective C and the application kit. A nifty program for people who rlogin to their NeXT is **define** which allows you to look up words in webster without the NeXT interface. The code is a good example of using the **libtext** library. NeXT has also provided a fixed version of the **mixsounds** Example program. The on-line version has a few bugs which have been cleaned up. People with serial lines may also be interested in **ZMODEM** and **UMODEM** which have shown up to complement the version of kermit already available. And, one of my personal favorites is **popi**, the Digital Darkroom. Popi was actually written in Australia for a wide range of machines. Joe Freeman at NeXT created an output driver for the NeXT. All of these are in the **source** directory.

I'm sure that isn't everything that I've received since the last Buzzings, but I think I've hit the highlights. If you have something that will raise the quantity and/or quality of publicly available software, send it to **next-archive@cc.purdue.edu**.
NeXT Attachments are preferred when possible, but I'll take them however I can get them! :-)

-gerrit huizenga
Moderator, NeXT Archives
next-archive@cc.purdue.edu

[I admit it. My favorite utility is Eyecon. Based on the X-windows program, two little eyes follow your mouse movements around the screen. I'm hooked. BuzzNUG will eventually (vaporware alert) compile an OD of the best of Pub. Domain for sneakernet transfer to those in the Georgia area who aren't on the net. Real soon now. -- EJL]

Developing an Intuitive Interface for Creating PostScript Effects on the NeXT

Andrew C. Stone

Abstract:

New interface techniques and usability testing guided the development of TextArt. NeXT software development has been an exciting and rewarding. The NeXTStep application kit greatly improves productivity, although dealing with a new OS has its challenges.

I.Introduction

Last spring when NeXT brought its Dog and Pony show to town, I convinced my colleague Kris Jensen to invite these NeXT guests to our user's group. Everyone was drooling, so the topic turned to "how do I get one now?" The fastest route, explained the helpful sales representatives, was to "Become a NeXT developer". Having written software for the Macintosh and UNIX platforms, we found the NeXT to be a happy marriage between all we loved of both worlds: a grep and click dreamland. Stone Design [*Andrew's company - EJL]* submitted a proposal to create "Acme Gizmos", a package of add-on interface objects with even higher abstractions than offered by the NeXT development environment, the Appkit. We planned to provide objects such as Decks of cards, special interface objects like circular sliders and virtual spheres, and other objects which would aid non-programmers in creating useful applications. NeXT accepted us as developers, and last May, we were off to camp.

Developer Camp was definitely worth the $750. The only drawback was they made us quit hacking at 8pm each night. After four intense days, the 30 or so developers returned home, with manuals and cubes in hand, hierarchies and "views" in head.

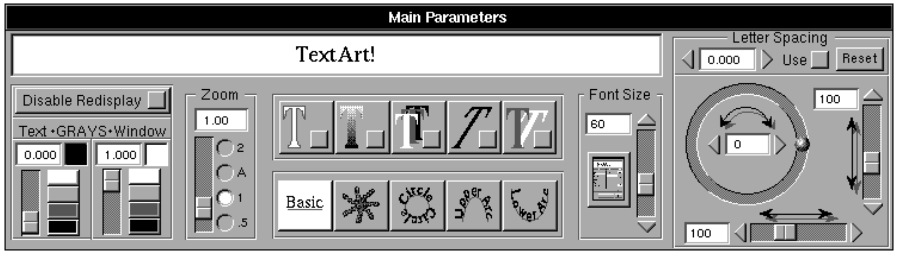
After two weeks playing with the Appkit, I had found a better tool to work on: Here was this amazing machine with Display PostScript, and no way to brainlessly create interesting PostScript images. To make a long intro shorter, TextArt was born. Given NeXT's desire to position the cube as the desktop publishing workstation of choice, a package which allowed anyone to create powerful images instantly seemed like a winner. Six months later, after usability tests and much interface tweaking, version 1.0 was released. We will explore some of the custom controllers and special interface techniques developed for this product, and show how usability testing throughout the development cycle aided us in making design decisions.

II. The Mental Model

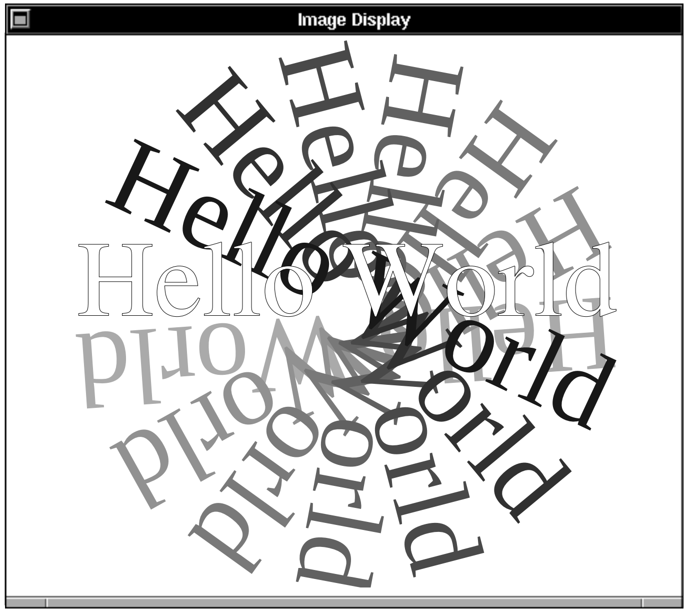
TextArt maps between the primitives of the PostScript language and graphical user interface objects. By clicking buttons and dragging sliders, the naive user has at his fingertips all the power of this language. Since the program was designed for people with no experience with PostScript, we attempted to provide the main options explicitly represented on the screen to let users "play".

Our goals were

- to provide an environment which invites and rewards exploration. We believe that this adventure somehow unleashes creativity.
- to provide consistency throughout the interface.



The main parameters panel provides access to the functions which apply to all modes and options: rotation, scaling, kerning and fonts; as well as the Image Window functions: zooming, gray levels and an inhibit display switch. The result of the execution of the generated postscript code is displayed in the Image Display window.



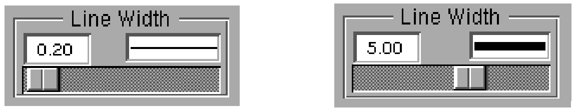
The Image Display Window

The Image Display can be saved as an Encapsulated PostScript (.eps) file, transferred via the Pasteboard [the NeXT's shared buffer] to other NeXT programs or placed in a TextArt layout board for combination with other graphics. The layout board section is a complete draw package. A snapshot of an image can be very compactly represented as the state of each controller. Packed into a byte stream, files are less than 200 bytes.

Options and mode controls form the heart of the Main Parameters panel. Each option brings up a related window with more controllers specific to that function. Later we will explore the evolution of this panel.

III. New Controllers: Direct Manipulation by Proxy

Although the NeXT with its 68030 CPU is quite fast, the elegant 4 gray level bitmaps and Display PostScript window server do take their toll. We derived a mental model we call "Direct Manipulation by Proxy". Since the time to reimage a complex image can be several seconds, each controller has local feedback. We created special "Views" which echoed in real time the effect of the control. For example, the line view shows the line width:

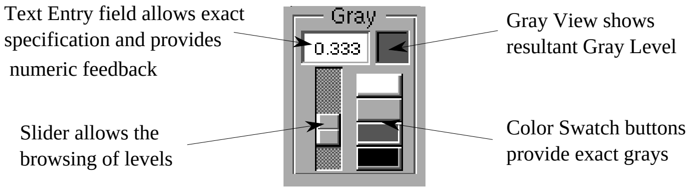


Line Views reflect the current line Width

Similarly, we developed a multi-purpose Gray Scale Controller. We wanted to incorporate these features:

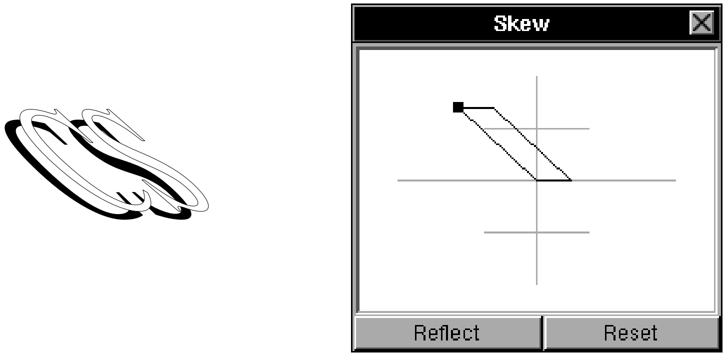
- Be able to specify the four exact gray levels instantly
- Be able to specify an arbitrary gray level exactly
- Be able to “walk the values” in order to select a level

The PostScript language represents gray levels as a real number from 1.0 white to 0.0 black. Our gray scale controller combines a text entry field for typed specification with numeric feedback when the slider is dragged and an exact color is specified via the color swatch buttons:

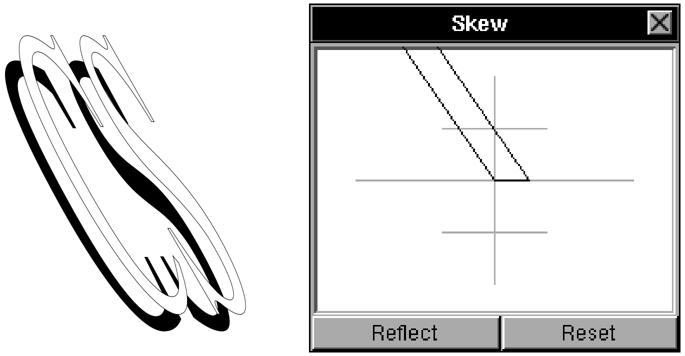


Two Dimensional Controllers:

Several options, such as multiple, skew and shadow, are best specified by coordinates. We used a custom controller that allowed mouse input to represent the x and y values. Users can predict the outcome of the operation.

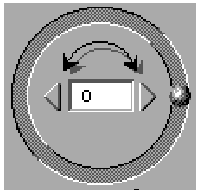


One really useful side effect of entering a mouse down loop while tracking the mouse is that the window server stills sends mouse location events from beyond the perimeter of the current “key” window. This allows an arbitrarily large specification.



Rotational Sliders

A slider is a controller which allows a range of input values. In an effort to produce a mapping between the purpose of a slider to indicate the rotation of text and its visual appearance, we created a “Circular slider”. This device, pictured below, performs this mapping nicely. As an ergonomic feature, “crawling”, that is, moving by one degree at a time, is done by clicking on the right (positive) or left (negative) arrow.

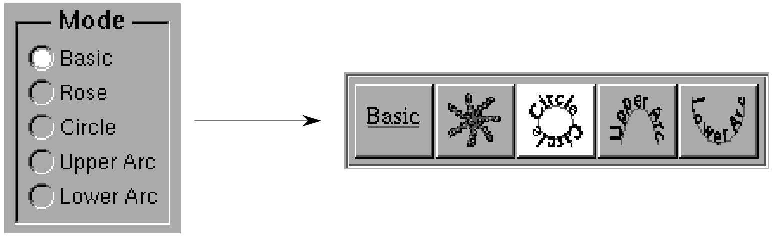


Rotational Sliders produce a mapping between Text Rotation and User Input

IV. What We Learned From Usability Testing

Many of the elements of the interface underwent radical change because of problems recognized during usability testing. Throughout the course of development, a cross section of potential users were asked to test the program. Informal protocol studies were made. Perhaps the biggest problem with usability testing is that it shows that there exist problems, but not how to fix them. Only iterative testing can determine how successful a re-implemented interface is.

The interface moved towards "direct mappings" to help unclutter the general appearance by replacing text descriptions with icons. During the evolution of the interface, we realized that users do not share a common vocabulary in the description of effects. Previously the user had to learn the text description of an option, such as fill or outline. By representing the function with an icon showing the effect, the cognitive load placed on the user is reduced. The next figure shows the before and after of the modes. In **TextArt**, the mode refers to the manner in which the text is placed: in a line, a circle, etc. Since this is a mutually exclusive choice, radio buttons seemed like a natural way to represent the function.



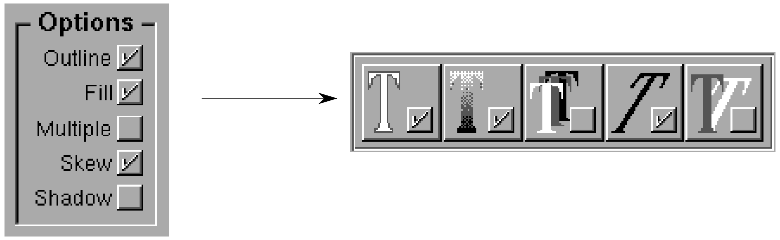
The text representations didn't tell the user enough. We iconically combined the the text with the result of the choice using the layout of a set of old fashioned car radio buttons. Now the user can "see the text path" and select the correct one without first translating the text to an image and then deciding if that is the correct choice.

The option selection area likewise required translation of text into image, so was another candidate for representing pictorially. In order to show that the choices are combinatorial as opposed to mutually exclusive, we borrowed the checkbox metaphor already present in the NeXT interface.

Standard NeXT CheckBoxes:

No Yes

We integrated the checkbox switch with pictorial representations of the effects:



Users reacted very positively to this change, and it helped to satisfy our goal of consistency and explicitness.

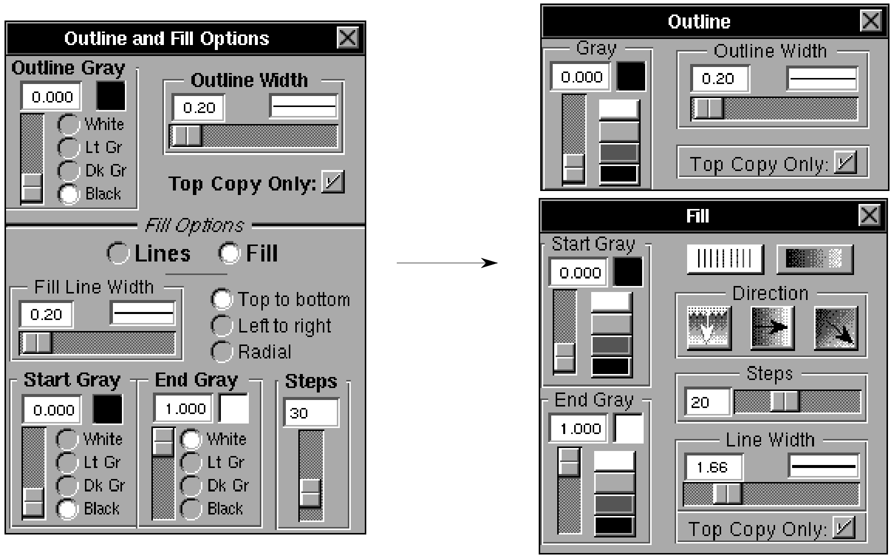
Outline and Fill Options Dilemma

Another component of the interface which gave us trouble was the way to present the various options for fill and outline. In the PostScript language, you can "clip" to an arbitrary path, and then fill or outline this clipped region. Thus there is a fundamental algorithmic connection between fill and outline. This originally led us to present the user with these options on one panel (*see next page*).

First, note an earlier implementation of the Gray Controller using standard NeXT radio buttons instead of color swatch buttons. Secondly, note that we have barely met George Miller's "seven plus or minus two" rule, there are nine functions in the original panel. There has been talk that this rule should be restated "seven minus two"*, given recent studies! Usability tests showed that there was too much going on. Although there is an

[In studying the limitations of human capability, research indicates that people can retain at most seven plus or minus two facts, numbers or other concepts at a time. It is generally recommended that designers should know both the capacity and limitations of their audience in order to provide the most effective working environment tailored to the user. -- EJL]

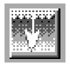
underlying programmatic reason for having the two options together, it is more consistent with the rest of the interface to have them separate: each check box on the main parameters panel brings up a separate options panel. The decision was made to break the panel into two, the only drawback being that the “Top Copy Only” switch needed to be replicated on the second panel to always insure access to this option which relates to the underlying clipping and not to either style in particular. Subsequent usability testing showed us that this division of 3 and 7 items was much more understandable to naive users.



There are several other issues in the redesign of these panels that warrant discussion: layout and organization, and iconification of ambiguous text. Following our desire for consistency we altered the layout so that all the color controls occupied the left half of the panel. This allows the user at a glance to establish the purpose of that portion of the panel, and leaves just a linear arrangement of less obvious controls. This separation of purpose is further emphasized by having all the gray control sliders move vertically and all the other options that use sliders use horizontal ones.

The final changes were the iconification of the two sets of radio buttons present in the original panel. The “Lines-Fill” pair caused users undo problems, after all, what does fill with fill mean?

The direction set of options represented a challenge to iconify. Note that two pieces of information are included in each switch: the direction of the fill and where the start is. This second piece of information provides the mapping for using the “Start Gray” and “End Gray” gray scale controllers.

For example, “Top to bottom” was translated into  which contains both pieces of information.

V. Conclusion

Software is a process, not a product. Although managers do not want to hear this, it is true. Usability testing throughout product development will elucidate problem areas which can be iteratively refined. However, the success of the design decisions lie in whether real end users can effectively and enjoyably use the program. Thus the real world provides the next source of usability testing: a careful analysis of the problems which are discussed with the product support group will effectively show the weaknesses in the interface. Although this is no excuse for releasing Beta software, I contend that software is never perfect, and the most meaningful evaluation comes from the end user.

Using the NeXT as a Digital Scope

Raj B. Apte, Stanford University

Abstract

The NeXT may be used as a digital scope and matched filter for signal processing applications. This paper will discuss hardware and software I/O aspects, emphasizing the sndriver and custom DSP code. Example code follows the article.

Background

All electrical measurement systems require some mechanism of system calibration. The best example is in the field of microwave network analysers, which must be calibrated immediately before each use. These machines have traditionally used general purpose processors (often the Motorola 68000) with ROM codes. The success of this architecture has led one manufacturer to publish it’s bus protocols in order to allow third parties to develop data acquisition systems without having to redesign the display/calibration electronics. The processor and it’s support electronics comprise the ‘back end’ of the system; the actual measurement system, e.g. an analog to digital converter, is called the ‘front end.’ In this type of system the front end designer must write the processing/calibration code and test it on a second processor/simulator. In addition, the designer is limited to the display options of the back-end. While neither of these limitations is fundamental, using the Next machine is a less expensive and more flexible solution to the problem.

Alternatives to the Next

Both SUN and Apple make systems into which third party DSPs can be included. Neither system can be used without a DSP since in both cases the main CPU will be busy with the user interface and display functions. Neither of these systems comes with a fast DPS server. The cost of a fast DPS server and a DSP board make these options less cost effective.

Communication between the front and back ends

In most applications the analog to digital conversion will take place external to the Next machine, so that some protocol must be developed for digital communications from the front to the back end. Both the DSP56000 serial synchronous (SSI) and the serial communications interfaces (SCI) are available at the D-15 connector on the back of the Next. The SCI is set up for easy implementation of several 8 and 11 bit serial formats. The SSI is a more general interface that has a number of different timing options for 8, 16, and 24 bit data. Our front end generates 24 bit data, so we chose to use the SSI. At least two pins are needed to synchronize the data: SCLK and a frame sync, either SC1 or SC2, depending on the mode. Both of these may be internally generated; in our case both are generated externally from the clock on the front end. Although the SSI is a bidirectional port, our system only uses the front to back direction.

Methods of DSP/host communication.

After the data are received by the SSI and processed by the DSP they must be sent to the CPU for display by the DPS server. This step in the communications has the most options since several methods are possible for both the host and the DSP. The host may use the AP library, the sndriver, or the 56000 host interface registers directly. The AP library has a number of useful routines for moving data from the DSP memory to the host memory and requires the least development time. The only drawback to this method is that the host must steal DSP cycles to determine the status of the DSP. Both the sndriver and the memory-mapped interfaces allow low-overhead communications. Of these two, only the sndriver allows the use of interrupts. The DSP may use the AP monitor or the host interface registers. Since the AP monitor takes a certain amount of memory overhead I chose to access the host interface directly from the DSP.

One problem with choosing to perform the DSP interface at a lower level than the host interface is that the DSP interface code must conform to the handshaking specifications of the sndriver. Chapter 14 (SysRefMan) and Chapter 16 (SysRefNotes/16_Mach/Drivers/SoundDriver) are quite useful in understanding the specifications.

Digital Signal Processing Example

With the problem of IO resolved, the designer can examine the requirements of system equalization. In our system a full 1024 point, 24bit stereo bitstream is received in a burst mode (ie, at 5 MBit). Both channels have 1024 point FIR (circular) filters. When the input channels are received the first channel is stored in x external RAM (x-XRAM) and the second in y-XRAM. (The external memory of the DSP can be organized in two ways, as a linear space of 8k 24bit words or as two parallel banks of 4k 24bit words. In this example we use the latter organization.) This signal is then considered as one, complex signal, z=x+iy. This complex signal is transformed by a fixed point, normally ordered output fast fourier transform (FFT) routine composed at Motorola and available from their BBS. The benchmark reported is 3.4ms for the fft. Since the discrete fourier transform (DFT) is linear, the DFTs of the two real input sequences can be separated and encoded into hartley transforms. In terms of data storage, a data set of 1k real points occupies 1k complex

points (or 2k real points) in the fourier domain. The hartley transform encodes the 1k complex points as 1k real points.

The convolution theorem is applied in the hartley domain, and the results are transformed back into the time domain and sent by the host interface to the CPU.
The CPU forms a bitmap image of the data and sends it to the DPS server for display. The display is updated at slightly over a Hertz.

Conclusion

The DSP56001 in the Next machine is fully integrated with the CPU and system software. If native DSP code is written, highly efficient and flexible communication between host and DSP make the Next a useful instrument for real time data acquisition and processing.

CPU Code

```
/*
 * DSP matched filter and scope code.  R. B. Apte.
 * (415) 723-0294
 *
 * This code opens, takes ownership of,
 * loads, and boots the DSP.  It then opens
 * sndriver buffer for the DSP to fill.
 * When two complete waveforms are received,
 * the code then fills a bitmap and displays the
 * waveforms.
 */

#import <sound/sound.h>
#import <sound/sounddriver.h>
#import <mach.h>
#import <stdlib.h>
#import <stdio.h>
#import <appkit/appkit.h>
#import <dpsclient/dpsclient.h>
#import <dpsclient/dpsNeXT.h>
#define READ_TAG 1

#define DATA_SIZE (2*1024)

static port_t cmd_port;
static int done;
static unsigned  int *read_data;
static int read_count;
```

```
static int int_data[2048];
NXRect rect1,rect2,rect3,rect4;

// This routine opens a window in the
// default context set up by [Application New]
// and displays it.
defineps openWindow()
    0 300 1024 512 Buffered window
    windowdeviceround
    Above 0 currentwindow  orderwindow
endps

// fill a bitmap with integer-scaled data and display.
static void drawBitmap(int *data, NXRect *rect_ptr)
{
     static  unsigned int  i,map[16384];

     for (i=0;i<16384;i++) { map[i]=0;}
     for (i=0; i<1024;i ++) { map[ 16 -i /16 + data[i]*64]  |=
0x3 << ((i % 16) *2);}
     NXImageBitmap(rect_ptr,1024,256,2,1,NX_PLANAR,NULL,
          (unsigned char *)map,NULL,NULL,NULL,NULL);
         PSflushgraphics();
}

 static void read_completed(void *arg,
                       int tag,void *p,int nbytes)
 // This interrupt handler gets called when the entire
 // result array has been read from the DSP.
{
    if (tag == READ_TAG) {
     read_data = ( unsigned int *)p;       /* give the data
                                 buffer to the user program*/
     done++;
    }
}
```



```
main (int argc, char *argv[])
{
    int i,j, s_err, size, protocol;
    kern_return_t k_err;
    port_t dev_port, owner_port,
         temp_port, reply_port, read_port, write_port;
    SNDSoundStruct *dspStruct;
    snddriver_handlers_t handlers =
         { 0, 0, 0, 0, 0, 0, 0, 0, read_completed};
    msg_header_t *reply_msg;
    int low_water = 48*1024;
    int high_water = 64*1024;
    int read_width = 4;
    int read_buf_size = vm_page_size/read_width;


    read_count=DATA_SIZE;

// get the dsp resource and own it
    netname_look_up(name_server_port,"","sound", &dev_port);
    port_allocate(task_self(), &owner_port);
    temp_port = owner_port;
    snddriver_set_dsp_owner_port(dev_port,
                               owner_port,&temp_port);
    snddriver_get_dsp_cmd_port(dev_port,
                              owner_port,&cmd_port);
    protocol = 0;

// create DSP to host buffer
    snddriver_stream_setup(dev_port, owner_port,
             SNDDRIVER_STREAM_FROM_DSP,
             read_buf_size, read_width,
             low_water, high_water,
             &protocol, &read_port);
    snddriver_dsp_protocol(dev_port, owner_port, protocol);
    port_allocate(task_self(),&reply_port);

// load and boot the dsp; set up DPS parameters
    SNDReadDSPfile("dsp.lod", &dspStruct, NULL);
    [Application new];                      /* This sets up
```

```
the connection to DPS*/
    openWindow();
    PSsetgray(1.0);
    PSsetalpha(1.0);
    NXSetRect(&rect1,0.0, 0.0,1024.0,256.0);
    NXSetRect(&rect2,0.0,256.0,1024.0,256.0);
    NXSetRect(&rect3,0.0,256.0,256.0,256.0);
    NXSetRect(&rect4,256.0,256.0,256.0,256.0);
    reply_msg = (msg_header_t *)malloc(MSG_SIZE_MAX);
    SNDBootDSP(dev_port, owner_port, dspStruct);
 // receive and display 1000 waveforms before exiting
   for (j=0;j<1000;j++) {
 // this command sends a flag to the DSP
 // to begin receiving data
    snddriver_dsp_write(cmd_port,&read_buf_size,1,4,
      SNDDRIVER_MED_PRIORITY);
 // open the host interface buffer for receiving
   snddriver_stream_start_reading(read_port,0,
            read_count,READ_TAG,
              0,1,0,0,0,0, reply_port);
    done = 0;
 // wait for the DSP to receive
 // and process the data.  done is set by the
 // interrupt handler
   while (!done) {
       reply_msg->msg_size = MSG_SIZE_MAX;
       reply_msg->msg_local_port = reply_port;
       msg_receive(reply_msg, MSG_OPTION_NONE, 0);
       snddriver_reply_handler(reply_msg,&handlers);
       }

      for (i=0;i<2048;i++) {
          int_data[i] =  (128   + (unsigned)read_data[i]
          /0x100000) & 0xff;
      }
             drawBitmap(&int_data[0],&rect1);
      drawBitmap(&int_data[1024],&rect2);
      NXPing();
 }
vm_deallocate(task_self(),(pointer_t)read_data,read_count);
    exit(0);}
```


DSP Code

```
 ; -----------------------Includes
 include 'ioequ.asm'     ;standard equates file
                         ;for memory mapped io registers
 include 'sincos.asm'    ; motorola sinwave generator macro
 include 'fftr2en.asm'   ; motorola 1024 point fft macro
 include 'htofourier.asm'   ; hartley to fourier transform
 include 'ftohartley.asm'   ; fourier to hartley transform
 include 'hconvolve.asm'; hartley domain convolution
 include 'filter.asm'    ; macro for matched filter taps
 ;------------------------- Equates
points  equ  $400        ; 1024  data points
xram0        equ  $a000          ; 1024 point complex buffer
xram1        equ xram0+points
 ; 1024 point complex buffer
xram2        equ xram1+points
 ; 1024 point complex buffer
xram3        equ xram2+points
 ; 512 point complex buffer
xram4        equ  xram3+points/2
 ; 512 point complex buffer
dm_r_reqequ  $050001           ; message -> host to request
dma

streamflags  equ  xram4         ; dspstream flags
dma_doneequ  0                 ; indicates that dma is
complete

data    equ  xram0             ;input data buffer
coef         equ xram2            ;matched filter tap buffer
twiddle equ  xram3             ;fft twiddle factor buffer
output  equ  xram1             ;location of output waveforms
fftpoints    equ  points

 sincos fftpoints,twiddle
 ;invoke twiddle factor macro
 filter coef                      ;invoke filter  macro

 orgp:$0000                    ; boot  location
 jmpreset

 orgp:$0024                    ; DMA-host fast interrupt
;                            (see DSP56000 User's Manual)
 bset   #dma_done,x:streamflags
```

```
 orgp:$40
;
; send two 1024 bit data streams to host.
; based on Next's putHost.
; this code conforms to sndriver's handshake protocol.
;
newPutHost
_beginBuf
 jclr   #m_htde,x:m_hsr,_beginBuf
 movep  #dm_r_req,x:m_htx
 ; send dsp_r_req to host
_ackBegin
 jclr   #m_hf1,x:m_hsr,_ackBegin
 ; wait for HF1 to go high
 move   #fftpoints,b
 move   #output,r2
 move   #fftpoints/2,n2
 move   #$ffff,m2
 ; linear addressing
 do b,_realend
_sendreal
 jclr   #m_htde,x:m_hsr,_sendreal
 movep  x:(r2)+,x:m_htx
 ; send real values
_realend
 move   #output,r2
 do b,_imagend
_sendimag
 jclr   #m_htde,x:m_hsr,_sendimag
 movep y:(r2)+,x:m_htx
_imagend
_prodDMA
 btst   #dma_done,x:streamflags
 jcs_endDMA
 jclr   #m_htde,x:m_hsr,_prodDMA
 movep  #0,x:m_htx             ;send zeros until noticed
 jmp_prodDMA
_endDMA
 bclr   #dma_done,x:streamflags

_ackEnd
 jset   #m_hf1,x:m_hsr,_ackEnd   ;wait for HF1 to go low
_exit
 rts

;  go here first.  set the addressing modes,
```



```
;  hardware configuration, interrupt modes
;
reset
 reset                      ; reset the hardware.
 move   #$ffff,m5              ; set addressing mode
 move   #$ffff,m6
 bset        #0,x:m_pbc        ; host port
 bset   #3,x:m_pcddr        ; pc3 is an output with value
_reset
 jclr   #m_hrdf,x:m_hsr,_reset
 movep #>$000000,x:m_bcr;no wait states on the external sram
 movep  #>$000400,x:m_ipr   ;intr levels: SSI=-, SCI=-, HOST=0
 movep  #>$e0,x:m_pcc
 movep  #>$6000,x:m_cra
 movep  #>$2300,x:m_crb
 move   #>xram0,r5
 clra
 move   a,x:streamflags        ;clear flags
 bset        #m_hcie,x:m_hcr   ;host command interrupts
 move   #0,sr               ;enable interrupts
 do #$400,_readLoop     ;receive the data
_loopa
     jclr        #m_rdf,x:m_sr,_loopa
     move    x:m_rx,a
     do          #14,_scalea
     asr     a                 ;crudely scale the data
_scalea
     move a,y:(r5)
_loopb
     jclr #m_rdf,x:m_sr,_loopb
     move x:m_rx,a
     do #14,_scaleb
     asr a
_scaleb
     move a,x:(r5)+
_readLoop
;jsr newPutHost             ;uncomment to print input
waveforms
 fftr2en    data,twiddle,xram1  ; time -> fourier
 ftohartley points,xram1,xram0 ; fourier -> hartley
 hconvolve points,xram0,coef,xram1; apply hartley convolution
 htofourier points,xram1,xram0 ; hartley -> fourier
 fftr2en    xram0,twiddle,xram1
 ; fourier-> time reversed
 jsr newPutHost                ; send to host
 jmp reset
```

Simple Text Browser

Erica J. Liebman

I decided over winter break to attack my ongoing project of helping to port a full-featured text analysis tool. Between bouts of bronchitis, I ended up having to get back to basics, throw away existing programs and start re-coding from scratch. Along the way, I became a lot more familiar with the Appkit's scrollView and open-panel objects. A little utility I wrote along the way is this simple text browser. I think it demonstrates use of the ScrollView, Open Panel and "wait" cursor in a very straightforward way.

The key to this browser is the "docView" that the Application Kit automatically includes with every instance of a scrollView that you drag into your window in the Interface Builder. The docView is of type Text, accepting the message readText from a file stream. I was a bit surprised to see that NeXT had reimplemented file streams, (I usually use FILE *filename streams in C), but decided to go with the flow.

By including headers from the appkit, I was able to create a temporary OpenPanel object with a call to new. The runModalForDirectory method will return a zero if the user presses cancel, so I checked for this and did a conditional return out of the function. I am generally willing to use goto's (and this really does count as one) for error conditions, avoiding unsightly indentation problems. True C hackers may combine the following two lines by nesting at will.

```
my_open = [OpenPanel new];
if (![my_open runModalForDirectory:"." file:NULL])
         return self;
```

I found that I had to use the totally non-mnemonic NXMapFile to associate the stream with a file name. (An alternative is to use the open() command to get a file descriptor from Unix, er, I mean Mach, and then attach it to a stream using the NXOpenFile command) NXMapFile opens a memory stream and reads in the file contents. Typical use is :

```
NXStream *stream;
stream = NXMapFile("aPathname", NX_READWRITE);
       // or NX_READONLY
....
NXSaveToFile(stream, "aPathname");
NXCloseMemory(stream, NX_FREEBUFFER);
```

If you want to use the file descriptor from the open() command :

```
NXStream *stream;
stream = NXOpenFile(fd, NX_READONLY);
```

More information on the use of NeXT specific streams is available in Chapter 10 of the reference manual "Support". This is available online.

A good habit is letting your user know when operations will take some time. A better habit is letting them know how long it will take. I compromised here by changing the cursor to the "wait" cursor just before reading in the text. Since the entire file is read in at once (to

avoid checking for currently opened files, etc), this can take some time for longer files. (It took about seven minutes for a one-hundred-kbyte file. Your text files will probably be somewhat shorter. Using the wait cursor generally goes along these lines :

```
[NXWait push];
....
NXPing();
[NXWait pop];
```

As always remember to free storage pointed to by temporary object identifiers. Good memory management is important. Here are the steps to getting this running in interface builder.

1. Start Interface Builder, create a new application "TextBrowser" and save it to a fresh new directory. Also create a new project.

2. In the Mach Shell, copy the following files "SimpleBrowser.h" and "SimpleBrowser.m" to this directory.

3. Call up the classes browser by double-clicking on the ".h" suitcase. Create a subclass of Object called "SimpleBrowser". Add the SimpleBrowser.[mh] files to the project. Parse these files in the Classes Browser. In about thirty seconds, you'll be asked if you wish to replace the existing object. Agree. Instantiate the Simple Browser.

4. Drag a scrollview to your "My Window" and make it nice and big. In the Inspector, get rid of the close box and resizability. Get an "Open" bar into your menu. Set up your Info Panel.

5. Double click on the instantiated Simple Browser Instance. Select the Connections Inspector in the Inspector. Connect the myScrollView outlet to the scrollView. (Use control-alt-drag) Connect the Open menu item to the Simple Browser Instance and choose the action performOpen. Save everything.

6. Leave IB. Go to the Mach Shell and perform Make in the appropriate directory. (I know that you can do this out of the IB program, but I've had problems with it...).

7. Run the program. Bugs or real bad problems - contact me at erica@kong.gatech.edu.

```
/* Simple Text Browser Header File : SimpleBrowser.h*/
#import <appkit/appkit.h>
#import <objc/Object.h>

@interface SimpleBrowser:Object
{
    id myScrollView;
}
- setMyScrollView:anObject;
- performOpen:sender;
@end
```

```
/* Simple Text Browser Object File :   SimpleBrowser.m */

#import "SimpleBrowser.h"

@implementation SimpleBrowser

- setMyScrollView:anObject
{
    myScrollView = anObject;
    return self;
}

- performOpen:sender
{
     id my_open;
     NXStream *my_stream;
     char *s;

      /* Create and run appkit open panel.
         Exit if user presses cancel */
     my_open = [OpenPanel new];
     if (![my_open runModalForDirectory:"." file:NULL])
           return self;

     /* Use the system "Wait" cursor in case
        the file is especially large */
     [NXWait push];

     /* open stream using MapFile call */
     my_stream = NXMapFile([my_open filename], NX_READONLY);

     /* Place into the docView provided by the ScrollView */
    [ [myScrollView docView] readText:my_stream];

     /* close things nicely, free the open panel,
         return cursor to normal */
     NXCloseMemory(my_stream, NX_FREEBUFFER);
     [my_open free];
     NXPing();
     [NXWait pop];

     return self;
}

@end
```


TextArt : A First Look
Erica J. Liebman

TextArt, simply, is a good program with the potential to become a great program. Postscript special effects are easily generated for text objects for high-quality logos, labels and headlines. Controls are easy to use and intuitive. A lot of care has been put into the help system and passive presentation of dialogs. A number of key features that will make this program a best-seller (potentially) are missing from the current package. Overall, a solid, easy-to-use package with room to grow.

What is TextArt? Text Art is a graphics program that manipulates Postscript text objects along with a value-added version of the Draw program found in the NextDevelopers directory. The user can take a word or phrase and bend it, shadow it, rosette it or angle it to heart's content. I found it easy to start working without consulting the manual. Although the control panel is initially daunting, controls were placed sensibly and soon I was rosetting with the best of them.

TextArt has a main "easel"-like Image Display and many individual draw windows, similar to "Draw" windows. Individual text items are worked on in the Image Display and then transferred to the draw window or other programs. Double clicking on a text object in a draw window moves it into the easel window. However, as the images I worked with got more complicated, the switching time between windows kept increasing.

I feel strongly that Stone Designs should get rid of the Image Display and work only within the draw windows. Additionally, I really wanted to select a group of text items, apply a single transformation to all of them (upper arc and shadow, or rosette) with one operation. I couldn't. Going between windows and cutting and pasting took over a quarter of an hour to set up the entries for a schedule for my classes for this year. This operation should have taken under a minute. Another flaw in the Image Display is that there is no "undo" operation to recover from loading in another image accidentally (which, of course, I did).

Text Art handles several advanced drawing operations. I had no problem (once I contacted Andrew Stone -- he replied promptly) kerning. Text may be aligned on the draw window, and snapped to grid. There is, however, no distribute feature yet.

Postscript Effects : the Big Missing Piece : The biggest part of the puzzle that is missing from all this is the ability to perform these effects on general postscript objects. For example, shadowing a freehand scribble object. Or taking a square with the word NeXT in it and rotating it 27.5 (or so) degrees. Skewing, rosetting, all the features that are available for text right now in TextArt. The program is a definite **value added** draw program. I'd like to see Stone Designs take this further.

Out of the Box (so to speak) : TextArt installed smoothly in my Apps directory. A call to find applications loaded in the correct Icon without problem. I double-clicked and was on my way. I had a few obligatory crashes (any program using the NeXT appkit Font calls is sure to experience a bomb or two) but overall the program behaved well. The only problem I had was trying to transfer text objects from the Draw section of TextArt into WriteNow. In some mode, (and I don't know how I got into that mode) the darn things just wouldn't image right and kept trying to overwrite text. However, once I opened up a new WriteNow window, the stuff started transferred correctly and well. I can't figure out if the fault belonged to WriteNow or TextArt. Its happened several times now. I've had no problems printing, saving, cutting or pasting from within the program.

TextArt comes with a handy, easy-to-read manual, which frankly you never need to use. The program is virtually self documenting. It's nice to browse through and get ideas. The program also comes with a suite of sample and tutorial files that are fun to browse through.

Target Audience : Who is going to use this sort of program? Right now, people who want to add a bit of splash to their desktop publishing, presentations, letters, drawings, etc. If Stone Designs keep taking this program to its logical end, this is going to be a top flight drawing program. I can only see the target audience growing.

Feedback from the Trenches

The entire BuzzNUG Membership

Many of us are waiting with baited breath for the monitor adjustment article. Any chance of a special issue **David Farber**

BTW, just finished (Tues. pm, actually) reading the BuzzNUG newsletter -- congrats, looks exceedingly nice. One suggestion -- you might want to go to a smaller font, with 2 columns, for the printed version (or maybe even onscreen?) ... I prefer that format to the bigger type, one column, in general.... **Mark ^Zimmerman** *[Unfortunately Writenow can't handle the functionality we really need for this Newsletter. Hopefully someone nice out there at Frame will see this and take pity on me and send a license/review copy of FrameMaker for BuzzNUG.]*

I do NOT think that this magazine should spend valuable space espousing how wonderful the NeXT is compared to anything else in the known universe. First, most of the readers are convinced of that already, and second, discussion of that nature is already too prevalent in comp.sys.next. Articles I would like to see:

- A fixed terminal, w/ scrollbar, w/ keyboard stuff
- A reprint of the Wren V /etc/disktab from c.s.n (? I think)
- The canonical answer to 3rd party RAM expansion - Mac RAM chips? 4

Mbit? Page/Nybble mode? Allowable memory configurations? Sources/prices/bulk deals?
- The canonical answer to the source code issue - is a Sun or BSD license necessary? Prices/places to order from
- A list pointers to machines with Postscript stuff. I might even be able to handle this one myself. Also, a list of good books, and which ones to start reading first, to learn the language. This particular point tripped me up quite nicely.
- A do-it-yourself article from someone who has actually done the following: The best answer *I* know to the "I want a cheap floppy" complaint. Comparatively expensive, until you see the prices for the Dynafile drives. You get a cheap PC clone. You get an ethernet card. You get the public-domain ftp from the NSF. You can stick any kind of drives you want in there - 5.25", 3.5", 1600 bpi 8-track ape, PC hard drive, you name it. It's even PC-compatible!
- A review (with pictures! There should be many more neat pictures when you deal with such a visually-oriented machine!) of Lighthouse Design's EE CAD software (I can hardly wait). **Brian Bartholomew**

Thanks for helping spread the word about NeXT, and doing a fantastic job! **Julie Welch**, Sales Programs Manager, NeXT

Object Bee-Line

Wanted: "ScreenGrabber" object or application. I've been told that it can easily be done by making a transparent window the size of the screen, telling it to update itself (which it will do with the bits underneath itself), then moving the window offscreen, where the bits can be easily read. Contact Pat McGee, jpm@lanl.gov *[Try using /NextDeveloper/demos/scene -- EJL]*

Wanted: "Performance Monitor" object or application. This should duplicate the functionality of Suns perfmeter. I find this much more informative than the load average displayed by NLoad. Contact Pat McGee, jpm@lanl.gov *[Check out the Monitor program available on the public ftp servers--EJL].*

Wanted: Deck of Cards object that will handle sorting, dealing, hands, and iconic representations (both face and flip side) for a card deck. Contact Erica Liebman, erica@kong.gatech.edu.

Market View

I've received literature from 3rd Party Developers on the following Products. No warranty, express or implied, is given. The quotes are the developers' and may not reflect reality.

Communicae *Active Systems* 1-617-576-2000 "high-performance communication package"

TextArt *Stone Design Corp* 1-505-345-4800 "ideal for creating letterheads, icons, logos or other special effects"

Displaytalk NeXT *Emerald City Products* 1-800-233-0417 "200% jump in development productivity over YAP and Preview"

DaynaFile *Dayna* 1-801-531-0203 "external SCSI floppy disk drive for the NeXT"

MediaStation *Imagine Inc* 1-313-487-7117 "Software for capturing, storing and processing high-resolution images".

ADDITIONAL INFORMATION : Contact your local NeXT rep and ask for a copy (the non-confidential parts) of 3-News. This is supposed to be a bi-weekly update of timely third-party information, distributed to NeXT's sales folks. The issue I scanned mentions that T/Maker's ClickArt is shipping with good sales, Informix's Wingz is in Beta test & Sybase is working on a "commercial release" of version 4.0 SQL server.

User Groups

Unfortunately, not all of you out there can live in Georgia. Here are some pointers to Users Groups that may be in your area. (BuzzNUG is an equal-opportunity User Group. Belong to both. Belong to them all. :).)

Maryland/Northern Virginia/DC

•Washington Apple Pi reportedly has a special interest group for NeXT users.

•Ed Klein, the NeXT Consultant of UMD says he's started a user group : NUTS. Contact him at eklein@umd5.umd.edu.

Georgia

•BuzzNUG is sponsoring local demos and talks. Contact Erica Liebman at erica@kong.gatech.edu.

Massachusetts

•The Boston Computer Society (BCS) has a NeXT special interest group. Contact Dan Lavin at 1-617-969-6555, or Jan McPeek at 1-617-926-4027

California

• Robert D. Nielson 1-408-995-5775 and Jeff Wishnie 1-415-324-9567/1-415-780-2753 (voice mail) have started **BANG!** out of San Jose, loosely associated with Stanford. Jeff is apparently the Stanford NeXT Consultant. These guys are clearly in the best location possible for a NeXT user group and I was drooling at their speaker list. Definitely contact them if you live within fifty miles of San Jose. Or are willing to drive further. Or have your own 'copter. Robert has promised me Vietnamese Food if I make it to the Valley some day. No similar promises of food are made to any potential members. Sorry.

Scenes from the NeXT Issue

We've got a really great issue lined up for NeXT month. Here are some highlights.
• I'll do a quick write-up about using icons and a more-detailed article about using the DSP for image processing filtering operations.
• Dave Rosenbaum promises to review the best of the Public Domain
• Edward Jung of the Deep Thought Group is doing an article on "An Objective-C Runtime Class Browser." It's cool. I know, because I saw the first draft. Ed, by the way, is a co-moderator on NeXT-related topics for BIX.
• Dick Silbar promises an article on "Working through the NeXT Developer's Camp Labs"
• Morris Meyer of NeXT says he'll write about "Godzilla" & distributed processing.
• Roger & the Great Guys at Lighthouse said that they'd TRY (honestly) to get an article in on "Object Persistence", but that they're sort of trying to get their electrical engineering package out the door. *Either way, we'll be winners.*
• Ian Smith may get an article in on connecting CD players to the NeXT.
Hopefully, there will be more reviews and product announcements over the coming month(s). And now, what you've been waiting for all month :

Buzz's hint corner

• Try Shift-Clicking on attachments in mail. You can see where they are stored. Neat, huh?

• Ever want to see which applications are active but hidden? Look for the 3-dot ellipsis on the dock icons. If they are present, the application is not active.

• If it involves selecting an item and clicking on a default style button, try double clicking on the item **--** it usually has the same effect. In **Interface Builder** select source and double click destination **--** as good as clicking Connect. **--***Bill Bumgarner*

• In **Interface Builder**, you can add new method names and outlets to a class by selecting the object and typing Command-5 [Class] Command-1 [Attributes] **--** *Andrew Stone*

•In **Objective C** to add a Miniwindow Icon (when miniaturizing) don't use *setMiniWindowIcon:(const char *)anIcon*, because it's a typo in the manual. Instead, use (with **no** capital W)
- *setMiniwindowIcon:(const char *)anIcon(page 21-29 of Reference)*
Here is a typical call, in a +newFrame: method of a custom view.
[window setMiniwindowIcon:[Bitmap
    newFromMachO:"myCoolIcon.tiff"]]; **--** *Andrew Stone*