NeXT Users' Journal

Issue 11 - October/November 1990

CONTENTS

Contributers

Erica Liebman, Tom Poiker, David Spitzler, Conrad Geiger, John R. Schutz, Ernest N. Prbhakar, Kazunori Shioya, Mitsuhiro Kishimot, Steven R. Staton, Dick Silbar, William V. Smith, Robert Lin, David Carpenter, Eric P. Scott, Tom Masino, Jeffrey E. Froyd and Brian J. Winkel

Welcome

Erica J. Liebman
erica@kong.gatech.edu

THERE IS an odor of decay in the NeXT community.

Alas. My NeXT is not aging well. And poor Whimsy T. NeXT is not alone in her trials and tribulations. My disk drive is making strange noises. My mouse is barely responding to horizontal motion and keeps clicking and (no kidding) squealing The rubber which bands the keyboard and mouse are starting to decay. I have crashed my system five times in the last two hours. (That is two above average, mind you). I rebuilt my disk no less than four times over the past two months. At least my monitor and OD are behaving well -- unlike most NeXT users, I turn Whimsy off each day after use and dust frequently with tacky-cloth. Several NeXTs of my aquaintance have dimmed beyond vibility. Others have optical drives, broken and destructive to the unwary. I don't know if I am alone in my mouse woes or not.

I ask you, is there suddenly a universal plunge among NeXTs to problems and if so, why?

I have no answers. Merely suspicions. And let me state from the start, that the recent product introduction has nothing to do with it. Whimsy can't read and I haven't told her about the younger, faster, more slender workstations. (Although, I could heartily sympathize with some jealousy on that point - :).) Nor is she in receipt of strange short-waves signalling "obsolete, obsolete, obsolete". (This is, however, a familiar cry from my research advisors. I must bow away from comparison in this case). Still yet, it can't be that my warranty just ran out -- it did that months ago and the original warranty period was for only about three months (if I remember correctly).

I see two fundemental problems. First, that NeXT buys almost everything from third party vendors except the boards which are manufactured so uniquely inhouse. Without their strict standards, these items may suffer from quality control. The troubles of optical disk drives seem legendary. The whole broughaha with the laser printers (a whole lot of them blew their fuses when they were first being used -- apparently an error at Cannon).

Second, NeXT may have planned poorly how much use the machines would accumulate in the field. NeXTs run UNIX and UNIX workstation accumulate hours enormous in comparison to their meek PC cousins. They must be awake when the mail comes in, as the cron ticks and the system waits for ftps. There is no way to turn the monitor off. There is no command-dim. Certainly, these monitors should have been rated at about four or five years of continuous use (with dimming -- I want to reiterate here that Whimsy runs perhaps an average of fifteen hours a week, at one or two long spurts of use and has shown no monitor problems.)

There may be a myth growing here : Is UNIX machine hardware more tempermental than that of PC's? Could it be because the workstation pushes the state of the art back and the PC's are using tried, tested and fundementally limited equipment?

Frankly I don't know. I'd like to invite all of you this month to write in and let me know how old your machine is, how much use it gets and how its been shaping up. Maybe I'll find some patterns. Maybe I won't.

ON OTHER MATTERS, you might have noticed that this issue is in WriteNow format (a bit hard to mistake for any experienced NeXTie of more than two hours experience.) Along with everything else happening to my poor NeXT, my FrameMaker has up and died in conjunction with 2.0. Anyway, most everyone hated the problems with FrameMaker anyway. We'll see what happens when I get the update patches worked through.

I HAVE RECEIVED a copy of the new UUCP tech doc from Alan Marcum. This should be readily available to everyone very soon at the ftp sites if not already. At the last second, I did not include it in the issue because getting it reformatted from NeXT standards into NUJ was becoming near-hell. Anyway, its sharp and slick and I thank Alan for getting it to me.

I HAVE A BACKUP utility from Pat McGee which I had to put off to next month. Right now, this is the only article I have in hand for December. Please, if you promised me an article, get it in. I am hoping that Bruce Henderson will finish another ascerbic and witty "The Way I See It", Arthur Lee will get in that next MUMPS article and we'll be seeing primarily 2.0-focussed articles. Well written How-To with code is (as always) the NeXT Users' Journal domain. Please submit as soon as possible.

FEEDBACK from the TRENCHES
Letters, Hints and Other Fun

(•) Howdy. I am the president of the Phoenix NeXT user group. Phang - PHoenix Area Next user Group. Please change my e-mail address to ggf@jsoft.com.I found your first 8 issues to be very good. They got me going at a point where I was feeling sort of isolated. Not any more. I have the NeXT Users' journal to read now!

You mentioned that BIX has a NeXT special interest group. (ck out p48 in issue 8, which I found by loading the issue and doing a find!!) Dave and BIX do not know of anything special for NeXT user groups. However!!

I am checking it out and will let you know what the story is. I have an article about BIX I will submit as soon as I find out about the user group discount BIX will offer us. I am going to suggest to the moderators of the next conference they keep your newsletter online on BIX. I am sure users will like it. BTW, the moderators are Edward Jung and Bruce Webster.

Thanks again for all the hard work. You must have times when sleep is something you do without - lots! (-: **Gary Frederick,** jsoft!ggf@uunet.UU.NET

(•) Thanks for all your efforts with the newsletter. They are all very informative. Keep up the good work!--**Steve Hayman** Workstation Manager Computer Science Department Indiana U.

(•) Hello Erica, Thanks for publishing my article in Buzz9. The text part of the article came out very well. But, I have to get this this grouch out of my system .. ! The Notebook part of the article came all in ascii without those beautiful cells and the format I originally had..instead, there were some uncanny comments about font,pallettes, colors and what have you..How did they creep in..

I wonder if there is anyway of providing a better source of the algorithm to the interested... through the archives.. Think about it..Thanks again..**Krishnaprasad Kamisetty**, kamisett@enuxha.eas.asu.edu

The uncanny comments clearly were *your illustrations. OOOPS! -- EJL*

(•) A quote that I picked up..."WOW!!! According to an article on the last page of UNIX TODAY 10/1/90, Graham Nash of Crosby, Stills, and Nash is dumping his Sun workstation and buying a NeXT... to be used "as a design tool for a multimedia stage show that will support his planned 1991 solo tour." -- **Conrad Geiger**, Conrad_Geiger@NeXT.com

(•) Hi Erica. here's a little hack that other people might find useful...

The Question: how do you find out what defaults can be set for a program using dwrite?

the Answer: start it under gdb (e.g. 'gdb /NextApps/Terminal') and then give the following string of commands to gdb:

```
break  *0x605d89e
```

```
commands 1
silent
printf "%s: ", *$a2
output {char *}(4+$a2)
echo \n
cont
end
run
```

For example, the output for Terminal is

```
DoLaunchTiming: (char *) 0x0
HomeDirectory: (char *) 0x0
LaunchTime: (char *) 0x6006259 "0 0"
MachLaunch: (char *) 0x0
SavePanelTiming: (char *) 0x0
Uid: (char *) 0x0
UserName: (char *) 0x0
BoldSystemFont: (char *) 0x60062d5 "Helvetica-Bold"
BrowserSpeed: (char *) 0x60062f1 "50"
Printer: (char *) 0x60062fc "Local_Printer"
PrinterHost: (char *) 0x6006316 ""
PrinterResolution: (char *) 0x6006329 "400"
ScrollerButtonDelay: (char *) 0x6006341 "0.5"
ScrollerButtonPeriod: (char *) 0x600635a "0.025"
ScrollerKnobDelay: (char *) 0x6006372 "0.001"
ScrollerKnobCount: (char *) 0x600638a "2"
SystemFont: (char *) 0x6006397 "Helvetica"
UnixExpert: (char *) 0x60063ac "NO"
NXAutoLaunch: (char *) 0x60063ac "NO"
NXCaseSensitiveBrowser: (char *) 0x0
NXHost: (char *) 0x0
NXMargins: (char *) 0x6006445 "72 72 90 90"
NXMenuX: (char *) 0x6006459 "-1.0"
NXMenuY: (char *) 0x6006466 "1000000.0"
NXOpen: (char *) 0x0
NXOpenTemp: (char *) 0x0
NXPSName: (char *) 0x0
NXPaperType: (char *) 0x6006497 "Letter"
NXShowAllWindows: (char *) 0x0
NXShowPS: (char *) 0x0
NXFont: (char *) 0x6006397 "Helvetica"
NXFontSize: (char *) 0x60064ca "12"
NXFixedPitchFont: (char *) 0x60064de "Ohlfs"
NXFixedPitchFontSize: (char *) 0x60064f9 "10"
NXMallocDebug: (char *) 0x600650a "32"
SourceDotLogin: (char *) 0x279f "YES"
LoginProgram: (char *) 0x27b0 "NO"
NXFixedPitchFont: (char *) 0x27c4 "Ohlfs"
NXFixedPitchFontSize: (char *) 0x27df "12.0"
Console: (char *) 0x27b0 "NO"
Shell: (char *) 0x27f2 "DEFAULT"
Columns: (char *) 0x2802 "80"
Lines: (char *) 0x280b "24"
WinLocX: (char *) 0x2816 "100"
WinLocY: (char *) 0x2816 "100"
NXMenuX: (char *) 0x282a "0.0"
```

(note that you get a whole bunch of program-independent defaults at the beginning that are loaded by appkit.) This also starts the program running; you'll probably want to quit it, and quit gdb (by typing 'quit'). By the way, the magic number in the first line is an address in the shared library (NXRegisterDefaults+70), and almost certainly will have to be changed to work under 2.0. .mike. **Mike Dixon**, Xerox PARC

(•) Dear Ms. Liebman, IMPACT is an entrepreneurial software publishing company located in Ithaca, NY. We are currently introducing a software credit card which will allow NeXT users to directly test out demo software on their own machines, and to purchase the fully functional version by calling up IMPACT's 1-800 number and charging the software to their account.

We have information and application form brochures which explain the IMPACT software credit card in more detail, and we would like to make them available to your user group members, if possible...Sincerely, **Mark Lee**, mlee@cs.cornell.edu, IMPACT Software Publishing, Inc. 306 College Avenue, Ithaca, NY 14850

(•) Here are a couple of snippets that I have found useful. The first make the screen background black (or another shade) without having to launch Scene. The second is a simple screen blanker which can be executed from Terminal (or Shell). **Jim Burns**, burns@cc.gatech.edu

```
-clip & save in a executable file - sugg.name: setBackground -
#!/bin/csh -f
# Usage: setBackground color
#  Sets the background color on the NeXT as specified by color
#  James E. Burns, burns@cc.gatech.edu, 11/4/90
pft <<EOF
workspaceWindow
```

```
windowdeviceround
$1 setgray
setexposurecolor
0 0 1120 832 rectfill
EOF
--clip & save in a file - suggested name: BlackForeground ----
% BlackForeground  -- James E. Burns, 11/4/90,
      burns@cc.gatech.edu
% Clear the screen to black
% Usage: pft -f <thisfile>
%  Note: <thisfile> must have full pathname
0 0 1120 832 Nonretained window
windowdeviceround
Above 0 currentwindow
orderwindow
0 setgray
0 0 1120 832 rectfill
```

(•) In NeXT Software Release 2.0, the “recycler” replaces the “black hole.” --**Eric P Scott**

The Practical Side of Bezier Curves.

Tom Poiker
Simon Fraser University

Introduction.

Bezier curves have penetrated computer graphics with a vengeance. Whereas five or so years ago, only a few graphics experts knew about them, everybody is talking about them now and many are using them routinely in their work. But only few understand what they are except maybe that they make very smooth drawings. But there is more to the issue and the average user might profit from knowing more about these elegant constructions. You might draw even better (smoother) pictures by knowing how they work.

In this note, I will describe Bezier curves from a practitioner's standpoint, not from the standpoint of a computer scientist. For the latter, the mathematical elegance of the curves and their functional flexibility are appealing, for the practical designers, it is more their graphic simplicity and ease of manipulation that attracts them.

How Did we Get From There to Here?

Half a decade ago, smooth curves were drawn by creating a large number of straight edges that approximated arcs and curves. It does not matter so much how they were computed, the fact is that sets of edges have some serious shortcomings:

- the edges have to be very short to give the impression of a continuous line,
- even with very short edges, the many start-ups can make the line appear irregular, no matter whether it is drawn by plotter or a matrix printer,
- the graphic becomes scale-dependent, i.e., we cannot change the size of our drawings at will because the length of our edges changes as well.

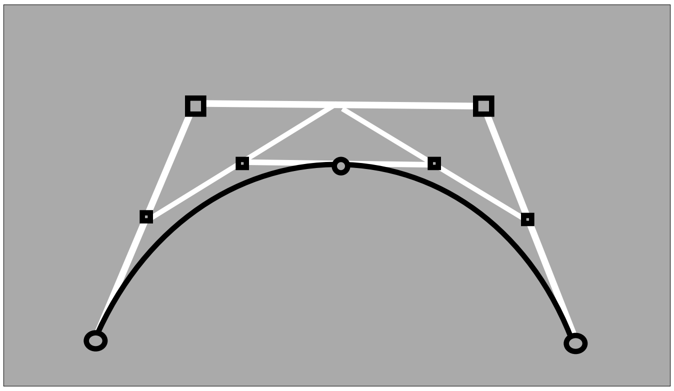
This shortcoming was unavoidable as long as we used pen plotters as our main graphic output devices. The thickness of the pen suppressed most of the irregularities, without giving us the crisp curves that we are used to today, though.

This situation changed radically with the introduction of the laser printer. With 300 to 2500 dark black dots per linear inch, we can compose cool curves that can compete well with manual techniques. We can set the dots so well that the eye, at least the naked eye, cannot tell it from a continuously drawn curve.

A history of computer graphics will tell us that graphic evolution could have gone several ways. One way would have been to develop a class of dot patterns for every possible font size. This is basically the way it was done for QuickDraw on the Macintosh and similar systems. The alternative was the definition of fonts as graphic objects. This is the way PostScript does it:, everything is treated as graphic objects.

PostScript is a major achievement in the graphic community. It brings together the knowledge and experience of computer graphics research and matches it to the prevelant technology of laser printers. Without going into detail, PostScript maintains all graphic objects in a scale-independant form right up to the point when an image is to be printed. The graphic objects can be arrays of pixwels but are more often connected series of curves and striaght lines. All are kept in a display file that can be accessed and manipulate, since it is all in ASCII form. Mind you, PostScript is one of the harder languages to learn: it is stack oriented and threaded, much like FORTH, not everybody's favourite high-level language.

But most people will not have to learn PostScript because there are many graphic packages which produce PostScript code for the printer. In this note, I will discuss three programs that use PostScript, TopDraw by Media Logic Inc., FrameMaker by Frame Technology Corp. and Adobe Illustrator by Adobe Systems Inc. The first two are the the major commercial programs on the NeXT with sophisticated graphic facilities, the last is the graphic design program against which much of PostScript graphics is compared (and which will hopefully be available on the NeXT soon).



Bezier Curves.

One of the corner stones for PostScript is its way to produce curves, called Bezier curves. A Bezier curve is defined by four points (Fig 1), two endpoints, called **anchor points** in Adobe Illustrator (large circles), and two intermediate points, also called **directional points** (large squares). The lines connecting the anchors with their respective directional points are the **tangents** of the curve at the anchors. Here is the code for the creation of two subsequent Bezier curves in PostScript:

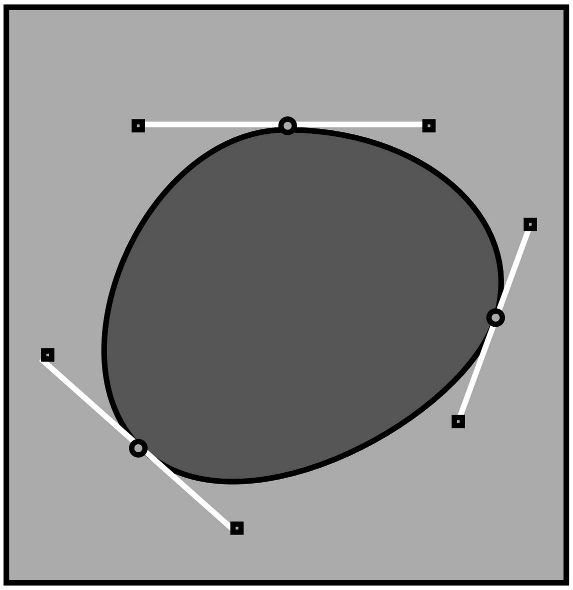
x_1 y_1 `moveto`
x_2 y_2 x_3 y_3 x_4 y_4 `curveto`
x_5 y_5 x_6 y_6 x_7 y_7 `curveto`

The first command sets the first anchor point. The first Bezier curve has points 1 and 4 as anchor points and points 2 and 3 as directional points. The second curve has points 4 and 7 as anchor points and points 5 and 6 as directional points.

The construction of a Bezier curve is graphically simple and computationally fast. Given four points, find the midpoints on all three connecting lines (small squares), link them and find the midpoints again. The midpoint on the final link is the first new point on the Bezier curve (small circle). Now we have two groups of four points and can repeat the process recursively until all pixels along the curve are found.

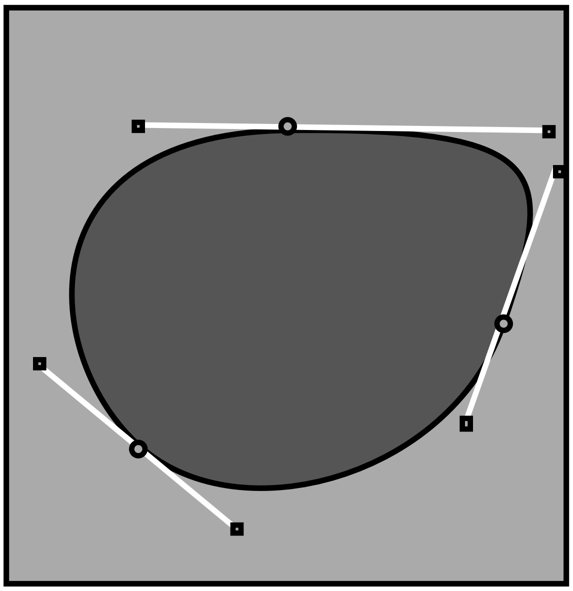
PostScript defines graphic objects in paths which are series of Bezier curves. A path can approximate practically every shape with a small number of curves which results in very compact storage. A path has two properties: **Stroke** and **Fill**. Stroke identifies the character of the line, the thickness, color (graytone), solid or dashed, etc. Fill gives the color (graytone) of the closed object. If the path is not closed, most programs construct a straight line between the endpoints and treat it as closed. "None" is also a color and means transparent. Illustrator only allows graytones and color but FrameMaker and TopDraw provide texture or patterns. For FrameMaker, graytones are a special case of patterns. TopDraw allows the pattern to be given in varying shades of gray. Neat concept.

Illustrator distinguishes between an `Artwork` mode and a `Preview` mode. The former shows the paths in standard one-point line-width, without stroke and fill, the latter presents the graphic as it will look in the final print, but in screen resolution. You work with the former and from time to time check out the result (you can have two windows, one `Artwork` and one `Preview`). The reason for this separation is speed. This way, graphic production is fast, even on a simple Mac Plus. FrameMaker and Topdraw use simple lines without fill while something is being dragged but snaps into visual mode as soon as the mouse button is released. They were never intended for slow machines and therefore adhere to the WYSIWYG philosophy all the way. I have not been able to develop complex graphics on either, therefore cannot say anything about the speed of these programs.



Continuity at an anchor point is maintained by aligning the point with its two directional points along a straight line. In order to maintain continuity at the first derivative, the distances between the two directional points and the anchor points have to be the same (Fig. 2). This is not always desirable. Therefore all three programs allow to lengthen or shorten the individual tangents. Since Adobe Illustrator always shows the curve and its tangents, one only has to drag the directional point to its new desired position. In FrameMaker, one needs to click the anchor point to show its directional points. If one wants to change an old curve, one first has to select `Graphics/Reshape` in the menu. TopDraw takes a few steps more: Select `Edit/Splines/Reshape`, then `Edit/Splines/Unlink Tangents` and then drag the point to its new position. Since `Unlink Tangents` also breaks the continuity of the two neighboring tangents and since at any drag everything except the actual curve disappears from the screen, it is a little difficult to maintain continuity. I usually had to do the dragging in several steps.

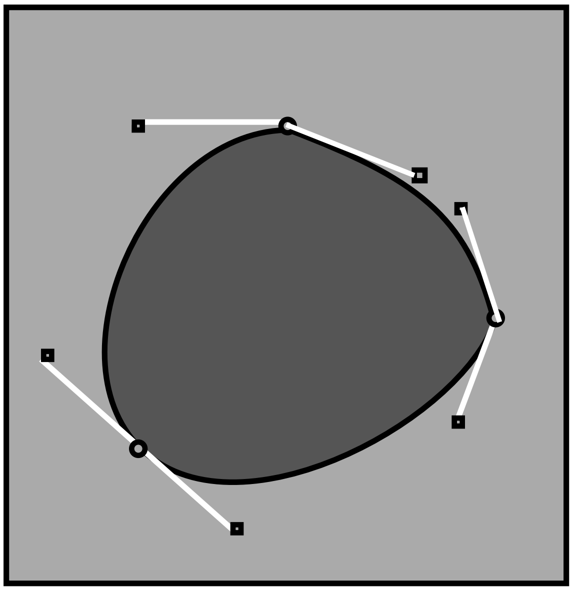
Figure 3 shows the results of such a change.

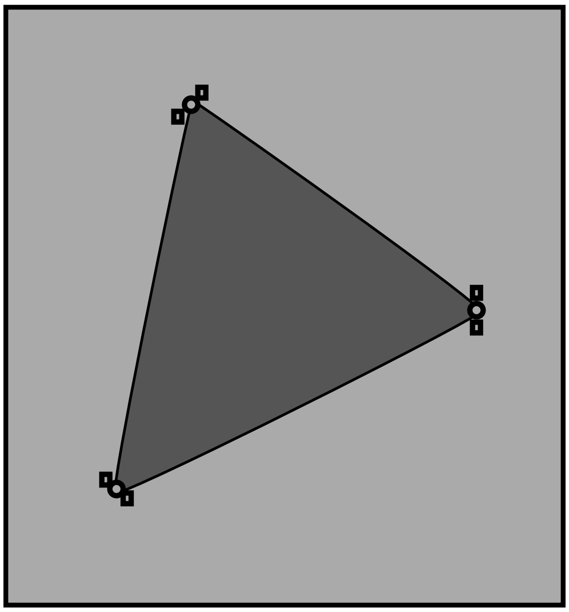


Bezier curves are changed by moving anchor or directional points. All three programs allow this. TopDraw and FrameMaker which construct a rectangle around athe chosen path, allow the move of every handle of this rectaangle, thus changing the x/y proportion of the curve. Illustrator goes a step further by allowing the user to grab a curve at any portion and move it in any direction. I guess Illustrator creates local coordinates for every quadrilateral constructed by the four curve points and changes the coordinate system when a point inside the quadrilateral is changed. This makes a change of a curve very simple. I would highly recommend to the developers of the other two programs to incorporate this feature as well.

Sometimes, we dont want a curve to be smooth at a particular place but have a corner. In practice, this means that we have two tangents, one to each side (Fig 4). Illustrator allows such a break by clicking the anchor point with the option key held down. Framemaker does it with a similar technique and TopDraw uses the approach as above.

It should be clear by this that a Bezier curve with zero length tangents is a straight line. We can also say that the shorter the tangents the sharper the curvature at an anchor point. TopDraw uses this to great advantage with two functions, `Tighten` and `Loosen`. Tighten reduces the length of all tangents of a path by 20%, Loosen does the reverse. FrameMaker has something different: `Smooth` turns a polygon into a Bezier path and `Unsmooth` does the reverse. The polygon sides become almost exactly the two tangents and the anchor points are placed in the middle of the polygon side.





Many people don’t like drawing lines with Bezier curves. Long and complex curves take time and effort and for many, creating curves by dragging tangents is counter-intuitive. They would rather trace lines and have the program convert the traces into sets of Bezier curves. Many programmer think that that is exactly what they should do and have therefore developed the **Freehand** procedure. I have always preferred doing it the hard way because it gives the user more control. Freehand usually produces many more points than the standard procedure because it follows all the tiny detours that a shaky hand can perform with a mouse. And the smoothness of a curve degrades with every point beyond the minimum. One can of course correct the curve but since it is impossible with most programs to delete points, such corrections can become more difficult than doing it the standard way in the first place.

FrameMaker is the exception. First of all, it collects the smallest number of points along a curve. FrameMaker has limited accuracy when in Freehand mode and seems to snap to a grid of appr. .05" but I haven't had time to measure that accurately. Secondly, FrameMaker allows the deletion of points while still attempting to maintain the shape and it does a very good job at that unless one reduces the path to less Bezier curves than there are ondulations in the original trace. But in such a situation nobody could do a good job. I would love to know how FrameMaker does this approximation. It seems to use slight breaks when it runs out of curves (a break is an extra ondulation) but it does it so well that it is hard to recognize the break without looking at the tangents.

NeXT on Campus

NeXT Computer's Quarterly Journal for Higher Education

David Spitzler

The Higher Education Group at NeXT Computer, Inc. is designing the winter 1991 issue of *NeXT on Campus,* and the editors NEED input. "We need leads on great academic projects," says David Spitzler, managing editor of the publication.

What to submit: the most exciting, interesting, mind-blowing, academic projects that are being developed across the country. If you've had a chance to read the fall 1990 issue of *NeXT on Campus*, you know what NeXT is looking for: To highlight projects in a variety of academic disciplines that show NeXT technology is being used for things that can't be done on other platforms. Most projects will be listed with a brief description in the section called "Academic Projects." Some will be the subjects of feature articles.

The drop-dead date to submit descriptions of projects is November 9. *[Ed : OOPS - Please DO submit, however]*

Please send your suggestions to David Spitzler (NOC's editor) and Jeff Wishnie (a campus consultant who is a key writer/designer, as well as guardian of our academic project list):

David_Spitzler@next.com and **Jeff_Wishnie@next.com**

You may send U.S. mail to:

David Spitzler
NeXT on Campus
NeXT Computer, Inc.
900 Chesapeake Drive
Redwood City, CA 94063

Please submit the following information:

1. Name, department, and college/university of the person managing the project
2. Brief description of the project or application
3. Telephone number and e-mail address of the project manager

(For guidelines on what to write, see pages 22–25 of the fall *NeXT on Campus.)*

Thanks for your help!

FYI : What is CD-ROM

Conrad Geiger
NeXT Inc.

FYI,What is CD-ROM: Like our optical disk drive, CD-ROM uses Optical Media to read information. They store over 500MB's of information on a single disk. Those of you who have used an Audio CD already are familiar with CD's. CD-ROM's store data - including text, audio, images and even video. They are only read from and cannot be written to.

What are the advantages of CD-ROM: The fact that you can store >500MB's on a very small disk is impressive. However, what is even more astonishing is that you can now duplicate CD's at about $2.00 per disk. That is as cheap as a floppy which hold less than 4MB's. CD's are very strong, cheap and easily manufactured. In fact, you can turn around CD-ROMs overnight through companies like 3M and PDO. Unlike our optical disk drive, CD's can be stamped out. Each bit does not need to be recorded on the disk like our optical drive.

What are CD-ROM's used for: Many companies are now shipping large data files on CD-ROM. These include complete books, encyclopedias, Professional Photography in 24-bit Color, sound tracks and combinations of the above. Their are children's applications, disk with 150,000 FAX phone numbers listed, Complete Business listings, etc. Access to information is becoming a key strategic advantage. This has become a new business. We intend to participate -- tools like our Digital Librarian are just the start of some of the things we will be able to do.

How do we support CD-ROM: First, you need a player! We introduced and are planning to ship in Q1, our CD-ROM drive for the NeXTcube. This is Sony's most advanced CD-ROM drive. It features <350ms average access time (they were 500msec) and combines a data cache to improve the system throughput. The drive provides auto inject and eject (similar to our floppy and optical drive support).

In the NeXTcube, the CD-ROM will work with a floppy and 105/340MB hard disk. The CD-ROM will be in the bottom bay. CD's are inserted and removed through the Optical slot.

The Sony Drive we are using is Model CDU-541.   Our 3rd Party that supplies our external floppy for the cube also will be selling the drive, but it will be in an external enclosure. This means for customers that want to use CD-ROM with the NeXTstation or NeXTstation Color can still purchase the drive. Both the internal and external drives are SCSI based.

Integrated in Software Release 2.0 is complete support for CD-ROM. We support both the High Sierra and ISO 9660 standards. This means that you can browse disk written for IBM. However, many Macintosh specific disk are not recognized in our system because we do not support the Mac file system. This does not mean that NO Mac CD's will work. Since, 500MB's is a lot of space, many publishers include both Mac and ISO formats on the disk. These are browsable in our system.

We also support other CD-ROM drives. Include are: PLI CD-ROM, Apple CD-SC and the Denon. We also believe many others will work, we just haven't test all of them.

We are planning to explore and utilize this technology to our advantage. We have worked with a company called Young Minds to provide mastering SW. It is available today. Check out the 3rd party guide.--Conrad

The CubeFloppy : My Experiences

John R. Schutz
Programmer/System Administrator
Center for Space Research
University of Texas at Austin
Email and NeXTmail: john@csrnxt1.ae.utexas.edu

Well, I feel stupid. I knew that the new system announcements were coming out soon, but I felt a pressing need. Here at the Center for Space Research we receive vast amounts of information, most of which is stored on MS-DOS/PC-DOS format 720k floppies. Well, my NeXT optical is an ideal place to store all of this data, but after a week of ’ftping’ the data to the NeXT, my fingers got lots of blisters and started bleeding. So I talked my department into purchasing a CubeFloppy 1.4 from Digital Instrumentation Technology, Inc. This floppy drive was advertised as being capable to read MS-DOS, UNIX, or Macintosh disks. Sounds like a good deal, because I can not only transfer my data from MS-DOS, but also fonts from my Mac to my NeXT.

The CubeFloppy came after about two weeks. I eagerly ripped open the box and found what I was looking for. There it was, a nice black color that matched my real Cube beautifully, except for the fact that when I thunk the top of the CF (CubeFloppy) I get a different pitch than when I thunk the NC (NeXT Cube). Well, I eagerly installed it, which was very simple. I just connected it to the SCSI port with the provided cable, and plugged the power cable into my power strip. Easy, quick, and painless (although I have not installed a drive on a NeXT with any other extra external SCSI devices).

Well, to copy their ’translation’ software onto my hard disk, I simply mounted the floppy, and copied the correct software into my LocalApps directory. I then corrected the permissions on it (it must run with root privs). All was well on the home front.

The Transfer Program

I started the transfer program provided and eagerly popped in one of my data diskettes. Everything worked fine. So I chose some files to copy over. That worked fine, except for the fact that it was excruciatingly slow. I am not sure of what the fastest reading rate for a 3 1/2" floppy drive is, but this drive certainly isn’t even near fast, but it worked.

After letting the first floppy finish copying, I ejected it. Then I hid the transfer program, like I do all programs when I might need it later. Well, the CF kept making this annoying clicking sound. Chalk up another one to bad design. The transfer program constantly checks to see if you have just inserted a floppy. So unless you want to restart the program every time you want to use it or leave a floppy in the drive whenever you log in, I hope you can selectively block out sounds. Five hours of clicking gets pretty annoying.

The transfer program is friendly enough (see below for picture of it), and seems to have lots of options. But, it is obvious that DIT rushed the product out the door to beat the new product announcements. Many ’features’ are not implemented. Let’s say you want to strip out control Z’s from all files to be copied over, and you want it done like this always, every day, unless you tell it otherwise. Well, you can choose filters like that, but you cannot save the options. Whenever you start up the application, you must rechoose file and filename filters.

One big disappointment was the fact that Macintosh disks are currently not supported. It will recognize the fact that the disk you just entered was a Mac disk, but you cannot transfer it. So much for fonts for now. Another strange thing is that UNIX format floppies are also recognized, but CF cannot read or write them with this release of the software.

The browser included is, to put it bluntly, terrible. At times, the browser has gotten confused, and when I try to change directories, it just wants to copy the file across. If I rename a file I have just copied over, the browser will jump up to the top of the directory, usually seperating you from the files you want to work with by great distances. The browser is also slow, with no options to display any extra file information.

Another annoyance is that when you are copying files, a status window pops up in the middle of the screen, and it stays in front of everything else on the screen. While I am writing this, I am copying over some files, and whenever I start a new batch, I have to move the status window out of the way so I can see what I am typing.

The Release Notes

The release notes are rather optimistic for the next (NeXT?) release of their software. In the next release of software they hope to include the following enhancements:

o Unix floppy disks may be loaded and formatted in the CubeFloppy transfer window.

o Display of disk and file information.

o Macintosh formatted disk compatibility (1.44 MB only)

As you can see, if you use 800k Mac floppies, it still will not work for you. Also, they mention nothing of improving the browser in the transfer window. They are going to send all registered owners the new release of the software when they complete it. I sure hope that they make many improvements.

IMHO

Well, this product is certainly not something I recommend that you just run out and buy. At about $500.00, it is not a steal of a deal, nor is it even a good deal. Unless you just *MUST* have a floppy drive for your NeXT now, I would advise that you wait and see what NeXT’s floppy drive will be like. In any case, I am, if not happy, at least content with the fact that there are no more blisters on my fingers.

Deck O' Cards

Ernest N. Prbhakar, Caltech
ernest@pundit.cithep.caltech.edu

I was supposed to spend this summer developing a user interface front end on the NeXT. However, using Interface Builder, it only took me an afternoon to get a decent prototype, much to the surprise of my boss. The code for the back end wasn't available yet, so I was able to turn my time to other projects.

One thing I had been interested in doing was creating FlashCards for memorization purposes. It seemed a useful thing to have running in the corner of the screen while waiting for other stuff to happen. I particularly wanted to implement the system my Greek teacher taught me, where you put cards further back in the deck depending on how easy it had been for you to guess them.

The user interface was easy: two scrollviews for the card faces, a Reveal button to toggle view of the back of the card, and three buttons (Easy, Fair, and Hard) to control placement of card back into the deck.

In addition to popping cards in and out, I realized I wanted to be able to scramble the deck, locate specific cards, and sort them in some appropriate manner. After a few days of design, I realized these was the same sort of things I would want to do with a deck of playing cards. So, in a fit of altruism (reinforced by Erica's pleas in the Object Wanted column), I decided to create generic Card and Deck objects that would be suitable for such use, and put them in the Public Domain for other programmers to use in writing their card games.

The importance of this decision is that it changed my goal from just solving a particular problem to creating a generic solution to a class of problems. I called up all the card sharks I knew and asked them what sort of things they wanted to be able to do with decks and cards. I then tried to implement them in the most general way possible. What follows is a brief summary of some of the choices I faced, which I hope will encourage others in writing reusable objects.

The first question was what to use for the superclass of Deck. At first I considered using Storage, and having each card be an element. However, I realized that if I made it a List, I would a) compartmentalize functionality between the Deck and the Card objects, rather than having it all in one place; an b) Deck would be general enough that people could use it with other 'Card-like objects' I hadn't even considered.

Having decided to make Deck a List, I therefore needed to make Cards some type of object. I had already decided each Card should have a front and a back, since I wanted to use them as FlashCards, so I first made it a subclass of Object and stored the information in an array (I was visualizing the information as text or RTF strings). Then I realized I wouldn't have to write archiving or allocation entries if I made it a subclass of Storage. This not only increased the elegance of the design, it expanded the power (arbitrary numbers of faces, element shifting, etc.). The only drawback was bugs in Storage and List that prevented the 'copy' method from working properly in subclasses, but NeXT Technical Services sent a workaround and promised a fix in 2.0 (along with everything else, right?).

Card.h thus looked like this:

```
typedef struct _CardFace {
  char *data;
  int length;
  int tag;
  id object; } CardFace;
static const char *cardDescription =
  "{*ii@}";
@interface Card : Storage ...
```

and Card.m is then:

```
@implementation Card
+ new
{
  return[self newCount:2];
}
+ newCount:(unsigned)numSlots
{
  return [self newCount:numSlots
     elementSize:sizeof(CardFace)
     description:cardDescription];
}
```

Having fixed my two main objects, I then had to decide how they needed to communicate. As a subclass of List, Deck generally doesn't care about what it is manipulating. For shuffling, drawing, discarding, and dealing out it just manipulates the indices. The two operations that it needs to know the contents for are sorting and finding.

In the latest version of Deck and Card, I use the convention defined by StepStone for their **OrderedCollection** class. I hope NeXT endorses and sets up a standard for such calls, so as to increase interoperability of objects. The compare: method returns +1, -1, or 0 depending on whether the passed object is greater than, less then, or equal to the called object. This is the same convention used by the Unix qsort(3) routine, thus allowing me to write a sophisticted sorting routine in only a few lines. (To be honest, this same trick is used in the DrawApp example to alphabetize the Activate menu, so I can't claim originality. For that matter, most of my structure was drawn from that excellent example). All I had to was wra p a function around the qsort(3) calls, pass the data array and that function to qsort, and Unix did the rest. I was very impressed by the breadth of Unix tools available on the NeXT, even apart from the AppKit and Co.

```
@implementation Deck
static int  qcomp(id *a, id *b)
{
   return[*a compare:*b];
}
...
- sort:sender
{
   qsort(dataPtr, numElements,
                  sizeof(id), qcomp);
   return self;
}
```

For searches, I used the findSTR: method for both the Deck and the Card. That is, objects could call other objects recursively to find if they contain a given string. However, I cheated. Although I passed the string, I also called the Unix regular expression system regex(3), so the Card just checked that instead of actually searching for the string.

```
@implementation Deck
BOOL regexString(char *expr)
{
   char *result = re_comp(expr);
   if (result) {
    Notify("Find", result);
    return NO;
   }
   return YES;
}
- findSTR:(const char *) aString
{
...
 if (regexString(aString))
  for (<all data>)
      if ([*data findSTR:aString])
         return *dataIndex;
 return nil;
}

@implementation Card
- findSTR:(STR) aString
{
 if(re_exec(FACE_DATA(activeFace)))
  return self;
 return nil;
}
```

The question now was which of the many fields and faces should be looked at during the search and sort. I decided there should be two flags, activeFace and sortType, that would be set to determine how to proceed. The former determined which element would be looked at, and the latter determined which field inside that should be looked at while sorting (for searches, only the data string was looked at). This information is entirely contained within the Card object. It only needs to be changed if one doesn't like the default behavior (searching on strings of face 0), and that would done by the application.

At first, I had a copy of each variable in each card instance, and I would loop through and change them all each time. Then I realized that they should all be the same, so I made it a class variable. Now the application can either set the default behavior to be whatever it likes, or can modify it dynamically by a Find Panel of some sort.

```
@implementation Card
+ (int)activeFace...
+ setActiveFace:(int)face...
+ (int)sortFlag...
+ setSortFlag:(int)flag...

- (int)compare:aCard
{
...
  switch (sortFlag) {
    case CARDSORT_Tag:...
   case CARDSORT_InvFace:...
   case CARDSORT_Face:
    default:...
    }
}
```

That is most of the interesting techniques, though there are some other things I personally thought were rather clever. If you would like to examine them more, the code is at available as 'DeckofCards.tar.Z' at an archive near you. The objects are bundled with **AppWithDoc** and **Document**, two abstract superclasses that contain the most useful code I was always stealing from Draw, as well as a front-end to the random(3) system call called **Random**. There is also the finished program, ***Flash***,(FlashCards.tar.Z) that does a creditable job of flash cards, and a demo program ***Deck*** (DeckView.tar.Z) that shows off the playing card capabilities of Deck using bitmaps purloined off of Poker.

I would encourage people to use these to develop their own card playing programs, as well as develop analogous objects for other purposes. Objective-C's inventor, Brad Cox, dreamed of a world of interchangeable part objects. To make his dream a reality on the NeXT would be in all of our best interests.

Minutes of First NeXus Meeting

`NeXus-office@etl.go.jp`
Written by Kazunori Shioya
Translated by Mitsuhiro Kishimoto

On July 25, we held the first NeXus meeting in Tokyo under the chairmanship of Mr. Shioya. It started at 7:00 p.m. and ended at 9:30 p.m. Forty people attended. Members came from not only Tokyo area. One came from Kobe and another came from Kyushu!

Mr. Shioya first explained that, at first, Canon Software Inc. tried to organize this user group, but we decided NeXus should be independent from any vendor. It means ownership of a NeXT cube is not a condition for membership and more people who have wider interests can attend. He also emphasized the necessity of volunteers to run NeXus.

The meeting also discussed regulations, activities of NeXus, and how to distribute PDS and newsletters. Distribution using Mac or MS-DOS format floppy disk was requested by several members.

We didn't have enough time for questions and answers or technical discussion, but, we had short demonstrations of Stuart and PS hack by Dr. Tashiro.

The monthly meetings are held on the fourth Wednesday.

GeoKit: An Object System for Map Rendering on the NeXT

Steven R. Staton
blackbox!deltos!sstaton@uunet.UU.NET

Brad Cox, the inventor of Objective-C, proclaims that the conversion of programming in the structure-oriented realm of the 1970's to the object-oriented paradigm of the 1990's is the same transition that manufacturing shops of the 1780's made in their migration from hand-made parts (each unique and thus non-interchangeable) to the machine made parts of the later Industrial Revolution[1]. The "standardized parts" paradigm made possible by machined components stimulated dramatic strides forward in mechanization, manufacturing and product support, because each new machine could be built upon the foundation of standardized parts that were readily available from other manufacturers.

The GeoKit™ is a first-generation product based on this notion: a standardized software part that is used in other software "machines" without the designers knowing how the "software parts" are designed. The machined software component, like its hardware analog, performs a certain function according to specification. How it does this is not known by the users; with black box objects, "Not Invented Here" has no bearing on software design.

GeoKit is a collection of NeXTstep™ objects which facilitate map rendering under NeXTstep on the NeXT™ Computer and the IBM RS/6000 running NeXTstep. The GeoKit is composed of a GeoView class, a GeoFeature class, and a GeoProjection class. In addition to the three main classes, there are also several smaller classes used for data storage and manipulation, like the QuadTree class. All these classes are part of the GeoKit.

GeoView Class

The GeoView class encapsulates a map sheet. A map sheet is best thought of as a sheet of imaginary paper, infinitely large, which can be stretched, rotated or slid underneath the window through which it is viewed, and upon which a map is drawn. The GeoView object is placed in a Panel or Window to display map data in an application.

Since the GeoView class is subclassed from the View class, it carries the features of a View such as having a superview and having its own coordinate system (set to reflect the coordinate system of the map.) the GeoView class is available within Interface Builder in its own palette. Typically, to place a map into an application, the first step is to instantiate a GeoView and place it within the DocView of a Window.

The GeoView object maintains a storage class object that is used to hold GeoFeatures that are displayed in a map. As a result, all data shown on a given map must be passed to the GeoView object in order to be included in the map. There are two NeXT storage classes available to store GeoFeatures in the GeoView: List and B-Tree. There is a third class provided with the GeoKit: QuadTree. The QuadTree class implements a Quad-Tree data storage technique, which is a method of storing graphic objects in such a way that they are quickly indexed by the region they lie in. This makes image generation, panning and zooming very, very fast.

GeoProjection Class

The GeoProjection class encapsulates geographic projections. It is extensible through Mathematica™ and by subclassing. As currently shipped, it supports twenty-six (26) different geodetic projection functions based on software developed by Petroconsultants (CES), Ltd. of Cambridge, UK. Associated with the GeoProjection class is a set of three IB panels for user control of projection objects: the Projection Panel, the Projection Parameters Panel, and the Ellipsoid Panel.

The standard user panel for choosing projections is shown in Figure 1. This panel is provided as part of the IB section of the GeoKit. To effectively utilize this panel, the NXApp must execute the following code as part of its AppDidInit:method of an application delegate:

#define PROJ_MAT 5 // Tag the panel in IB with this value to id it.

```
 - setMyProjectionPanel:anObject
 {
   id projCell, contentView;
   int   i, j, rows, cols;

   myProjectionPanel = anObject;

// Poll the matrix in the projection
// panel for the actual
// working  projections.
// To find the matrix, tag it with
```

---

[1] Cox, Brad. "There Is a Silver Bullet." BYTE, October 1990, pp. 209-18.

```
// the value PROJ_MAT, and
// look for a view with that tag.

contentView = [[myProjectionPanel
    contentView]
    findViewWithTag:PROJ_MAT];
[contentView getNumRows:&rows
    numCols:&cols];
for (i = 0; i < cols; i++)
  for (j = 0; j < rows; j++)
  {
    projCell = [contentView
       cellAt:j:i];
    [projCell setEnabled:
       [self isEnabled:
          [projCell tag]]];
    }
  return self;
}
```

In the above code fragment, the GeoProjection class is polled as to whether or not it can perform a given projection function. Using this technique, the projection panel is dynamically updated at run-time to reflect the projections it supports, on the fly, without having to know what the Projection class is capable of at compile time.

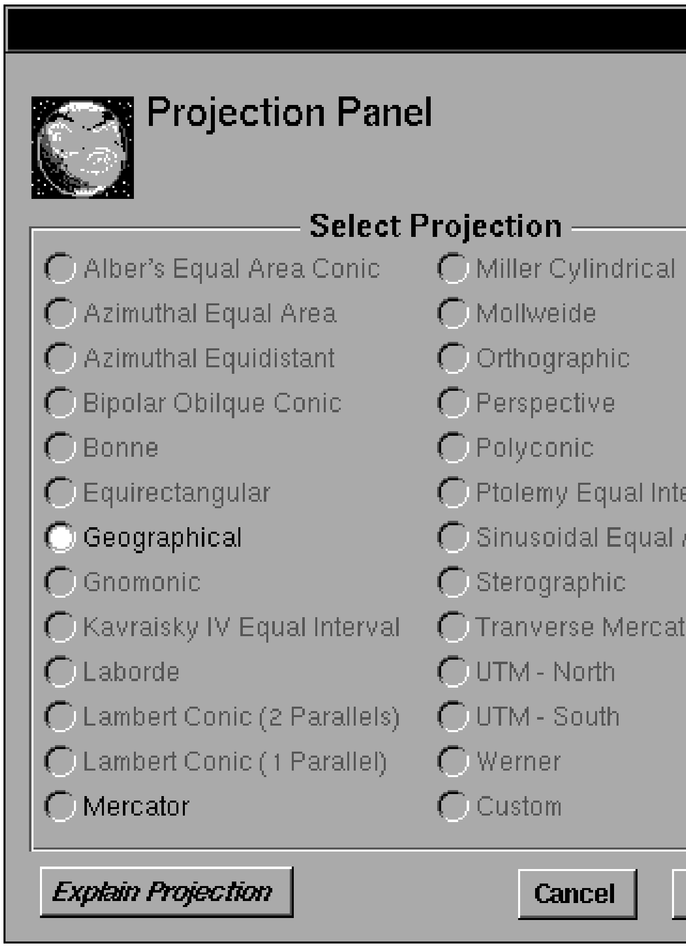


Figure 1. *The Interface Builder standard Projection Panel used with the Projection Class.*

Note that only two projection functions are available in this panel.

Panels for the ellipsoid and projection parameters are also provided as part of the standard GeoKit package (not shown). Since ellipsoids are merely mathematical constants representing the radius of the Earth, the standard Ellipsoid Panel is statically defined and does not need to poll the GeoProjection class with regard to the possible values of ellipsoids. The Projection Parameter panel must dynamically enable and disable fields of projection parameter values depending upon which projection is chosen in the Projection Panel. This is handled by the GeoProjection class itself.

GeoFeature Class

The GeoFeature class encapsulates features commonly found on maps. The fundamental class provides a polymorphism of the Feature concept, while all the subclasses provide the actual support code for each kind of object rendered.

The two primary considerations with subclassing GeoFeature are data storage (i.e. what kind of data is being stored in the map) and rendering (i.e. how to draw the object in PostScript™).

For example, the MundoCart™ data set contains three kinds of features: lines, text and points. Each of these data types is subclassed under GeoFeature and the particulars for storing the data of each feature are encoded in each object implementation. The display: method for each subclass contains the Display PostScript necessary to draw the feature, based on its instance variables and the current projection context.

Another example is the United States Geologic Survey (USGS), which uses a node, line, and area feature set to render maps. Consequently, a GeoNode, GeoLine and GeoArea class is needed to render USGS maps using GeoKit. Each of these subclasses of GeoFeature would store the data particular to its characteristics (a node is a point, a line is a line, and an area is a boundary enclosed by lines connected at the end points by nodes) and must know how to render its own instance data as graphics within a specific projection. The polymorphic nature of the GeoFeature class guarantees that even though the GeoKit knows nothing about what is stored in each of its user defined subclasses, each subclass object will respond to fundamental methods like display:, project:, and clear: in an unambiguous manner. Thus, one can treat any subclassed instance of GeoFeature like any other when using root GeoKit methods (i.e. those associated with projecting and displaying map information in a GeoView).

Summary of GeoKit

GeoKit is a system of objects for rendering maps under NeXTstep. It can be treated as a set of pre-defined parts of an application that provide a view of a map and the underlying data structures needed to store map information and project that information from within that view.

Rendering of PostScript is standard in the GeoKit, and the rendering methods understand how to optimize drawing in the Display PostScript model. They also know when to draw full PostScript for the printer. Bitmaps and the QuadTree data storage technique are used extensively in the GeoView to enhance performance and storage requirements. Printer output is in true PostScript vector format, to a true scale.

Interface Builder .NIB files are provided for key user interface components such as ellipsoid and projection selection. They may be ignored if user control of these things is unwarranted.

Terms and Availability

DFC is offering the GeoKit on a license basis, with a per-units-shipped charge based on volume. Source code is available on a non-exclusive basis. Customization of the GeoKit is available on a consulting basis.

Deltos Fleet Computing's GeoKit is a leader in standardized software component products for the NeXT, and will continue to be one of the most cost effective ways to add cartographic functions to NeXTstep applications.

Δ ***Deltos Fleet* Computing**

5104 Quail Creek Drive • McKinney, Texas 75070
214 • 540 • 2301 voice 214 • 540 • 2629 fax/digital

A NeXTmail Tale

Dick Silbar, Los Alamos National Laboratory
silbar@whistler.lanl.gov

Not long ago it occurred to me to do a "du -s ~/Mailboxes" command to see what was in that directory. It reported 17 MBytes of my disk was tied up in mail! On looking the directory and its descendents over (as described below) I found that most of the stuff in ~/Mailboxes was junk I thought I had once deleted.

Now, an important fact in my NeXT Life is that my 330 MB disk is always at least 87% full (even after removing Shakespeare and Quotations and a bunch of other things I don't use very often). I rapidly decided that I'd be a lot better off recovering most of that disk space. The following tale could be subtitled "Or, How I Learned to Compact my Mail." Maybe it will be useful to BuzzNUG readers who are as naive as I am (was?).

Warning: I was (and am) no mail maven. The following remarks often reflect my mindset about what I thought/think a proper mail utility ought to do. Most of this mindset was derived from using a VAX VMS mail utility for seven years before the NeXT.

Let me see if I can make historical sense of this. I had been out of town a few days and, on return home, decided to catch up on my mail. So I logged onto the Cube from home using my wife's PC. After clearing the Net news, I looked at my mail by doing "mail -f ~/Mailboxes/Active.mbox/mbox" from the c-shell. That showed 852 messages, all marked as U(nread). I.e., everything I ever received since the Cube first started getting mail some 16 months ago.

I may have tried deleting one or two messages of the newer junk messages, which I *now* know was a bad thing to try. [The Mail App was already in a launched state on the Console at the office.] However, I soon decided to get out of there; a hopeless task to try to clean it all up without the mouse. On quitting from Unix mail, I then looked at the size of my mbox file (more than 3 MB!). Golly Gee, so I then brought it into Emacs to look at it. True -- everything I had ever received was there.

This was surprising because I'm fairly religious about deleting junk and outdated mail. The headers in NeXTmail only showed some thirty or forty messages the day before I left on travel. So, I concluded (correctly, it turns out) that NeXTmail does not *really* delete things from the mbox files. I further surmised that maybe a deletion just changes things in the table_of_contents file.

On getting back to my Cube, I immediately unhid NeXTmail. Oops -- all the letters in the read-window were all jumbled up, one on top of another, not properly divided, and nothing showing having any relationship to the header that was highlighted. (The headers in the header-window looked more or less OK). The reason for this was explained to me much later; it was the result of the deletions I tried to do from Unix mail while Next mail was still running.

But I didn't know that at the time, and I had this problem of a jumbled-up read-window. After a while, it finally occurred to me to quit the NeXT mail application and re-launch it. This worked. It cleared up text in the read-window but -- horrors! -- now I had 852 messages in my header-window. (All noted as read, incidently.) All my deletions in months past had gone for naught.

Moreover, also in the Active.mbox directory there was EVERY NeXTmail attachment I had ever received, including ones that went with messages I thought I had deleted and other ones which I thought had been transferred to other mail boxes (folders). It is now worth saying that I tend to move a lot of my keepable mail into specialized mbox's.

[All this "ls -l"ing evidently has to be done from the shell -- unless there is a way to force Workspace Manager to go into such a directory that ends in ".mbox" (or ".wn") that I don't know about.]

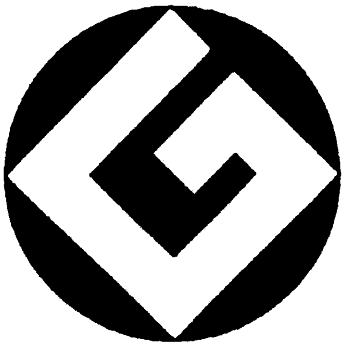
By now I suspected that (since some of these attachments were pretty big directories and files) most of that 17 MB in ~/Mailboxes was tied up there. Particularly after discovering that, say, my TopDraw.mbox directory *also* had **copies** (!) of ALL of my NeXTmail attachments, most of which had nothing to do with TopDraw. (This was also true, I believe, for the other mbox directories which had, somewhere, NeXT attachments.)

What was going on here? At this point I decided I needed some help, so I posted a plea to *comp.sys.next* on the NeT. Lots of folks (including at least two from NeXT.com) told me about a menu item in the Mail App called Utilities which in turn has a submenu item called Compact. The conjecture above was right -- deleted mail is not really deleted until Compact is called. In fact, you can even undelete things (as long as you do so before calling Compact)..

Other news from my respondents explained why the table_of_contents file was re-created from scratch (after I corrupted it so), giving me back *every*thing. And what I had to do in the future -- like QUITting the NeXTmail application on the Console, if I want to read my mail remotely. When you *do* that, in fact, you'll get the "You have mail" message when you log in from afar; otherwise you don't.

Unfortunately, to use Compact, I now had to go through the 852 messages in Active.mbox and re-delete the stuff I didn't want. OK, not more than about 40 minutes of re-living old memories. And then, Compact worked like a charm. It even gets rid of the attachments that are no longer attached. Neat stuff!

The end of this tale is: I now am down to a mere 3.6 MB in my Mailboxes.



James Higa, Manager of Canon Relationship
NeXT Computer Inc

The G-mark is an award given by the Ministry of International Trade and Industry (MITI) in Japan. Since 1957, MITI has awarded this mark to products available in Japan with outstanding appearance, functionality, quality, and safety. There are13 major product categories including Audiovisual, Transportation, and Information Equipment.

Last year for instance, 744 companies submitted 3,787 products. Of these entries1,146 products were awarded the G-mark. The Sony 8mm Handicam camera edged out the Nissan 300ZX for the overall Grand Prize Award as the best designed product of the year.

The NeXT Computer system is the Grand Prize winner for 1990!

We beat out Sony, Honda, and anyone else you care to name. This is the first time that a foreign company has ever won the Grand Prize. Rich Page and I will be in Tokyo on Oct. 1 to accept the award from the Minister of MITI. Dr. Yamaji will accept on behalf of Canon who submitted our application.

Congrats to everyone.

SIMM Installation

William V. Smith
smithw%mathnx.byu.edu

To install 4 MB SIMMS in a cube, assuming they are the sort with chips mounted horizontally (these are somewhat taller than the 1 MB SIMMS as you must know), you must have 4 or 8, or 12 of them (it is rather more work to install 16).

Pull out the motherboard. If you have 16 1 MB SIMMS, you will have to remove the ones starting with socket 15 (sockets are numbered 0 to 15). Socket 15 is the one closest to the end of the board with the external connections on it (ethernet T connector, etc.). If you don't have 16 1 MB SIMMS installed, you should remove enough to use up whatever amount of 4 MB SIMMS you have available up to 12 sockets.

Once you have your SIMMS installed, the board should have 1 MB SIMMS in sockets 0-3 at least (perhaps 0-7, etc.). Now looking at the open back of the cube, you will see two screws at the bottom holding in the drive/power supply case. Remove these two screws and carefully pull the drive case out of the cube. If you have never done this before, it could be a bit stiff but should slide right out after the initial pull.

You probably have some kind of hard drive installed so you will need to remove at least one of the mounting screws for the hard drive (not the optical). If you install 12 of the 4 MB SIMMS, you must remove both screws on the "board side" of the drive case and replace them with flat head screws. You *must* remove one screw in any case. This is the one (facing the back of the cube) on the right side closest to the back (i.e., closest to the back cover of the cube which you have removed to do this!)- be sure to check the other drive mounting screws to see that they are tight.

The remaining three screws will provide sufficient support for the drive (I have a big WREN in my cube and things work fine). Now start the drive case back in to the cube about 1/4 of the way and then start the motherboard back in to the cube, connecting the drive cables back to the motherboard while the motherboard is a few inches further out than the drive case. Now slide the motherboard 1/4 the way into the cube until it is even with the drive case, then SLIDE THEM INTO THE CUBE TOGETHER. The fit is snug but there is a VERY small amount of space between the SIMMS and drive case when every thing is back together.

If you are nervous about having your hard drive mounted with only 3 screws you can go out and buy a flat head screw to replace the standard round head you removed. Press both the motherboard and the drive case firmly into their connectors at the front of the cube.

Put the drive case screws back in. Reconnect the fan cable to the fan and without screwing in the backplate screws, hook up the monitor cable to the motherboard, plug in the power cord and start the machine . It does seem to be somewhat common to get bad 4 MB memory SIMMS so watch the boot on the ROM monitor. If you get error messages about memory, or the monitor doesn't light up, or something else is weird, it is probable that you have one or more bad SIMMS.

Sometimes at boot the ROM monitor will tell you which sockets have bad memory, and sometimes it won't. Bad memory can also lead to strange things happening with apps dying unexpectedly or the windowserver dying, etc. If nothing is wrong with the boot (you may get a message about different size memory being installed in various blocks-this is ok, the ROM will keep track of this and you won't get a second message on another boot). Watch the boot process a couple of times and if every thing is ok, turn off the machine and tighten the screws on the backplate, hook up the rest of your stuff (printer, ethernet, etc.) and you've got more memory. I have 8 1 MB SIMMS and 8 4 MB SIMMS in my cube. Works like a champ, except one of the 4 MB SIMMS went bad once a couple of months ago and the machine would hang all time. Watched a boot and the monitor told me I had a bad SIMM in socket 14. Replaced it and have never had any more trouble. My big jobs don't use much swap space now and things generally work much faster. Also, you can reinstall the same way on your upgrade board if you decide to go that route.

-Bill-

TAO

Robert Lin

Hey, Erica, how's it going? Haven't heard from you for a long while. Meanwhile, here's Tao #6 (hot off the keyboard press). Enjoy!

Tao #6 - Sep 23, 1990

The subscription list is now too large for me to handle. That's why I didn't mail out copies of Tao #5. The people who can get Tao on the Internet should contact me and have their names dropped. It costs me about $1 to mail each newsletter, not to mention the more expensive factor of time. Let's trim down that list so we can continue the public service.

This issue is devoted to analyzing the new line up of machines announced by NeXT on September 18th. I will assume you all know what the new machines are like.

-Robert Lin/rlin@cs.ubc.ca

To Upgrade Or Not To Upgrade

My friend Peter Phillips quipped, "If the 'slab' is termed to mean 'fast', 'lean', and 'slim', then what should we call the 68030 cube, which was slow, squarish, and big? Flab?"

NeXT offers us a way to shave the flab and replace it with 68040 muscle, via a board upgrade which costs $1,300 in hardware, $195 in software (release 2.0), or $1495 altogether. For developers and academics, which means most you readers, we pay $995. We pry all the RAM chips off our 68030, send in the bare board, get a bare 68040 board back, and repopulate it.

However, we will also likely need to buy an external 2.88M floppy. Let's face it, the good old days are gone and the bad older days are here again. With the new announcement, NeXT is effectively burying the optical disk as a method of distribution. In fact, by not offering an external SCSI optical drive, even companies that may have wished to distribute software on OD, will now reconsider.

Though the OD was never a good commercial distribution media, it sure made for excellent backup storage, not to mention the times when I sent out an OD chock full of all back issues of Tao and BuzzNUG. So, okay, we have to get a 2.88M floppy. There are two ways you can do this, either buy from Peripheral Land their $499 external 2.8 floppy, or buy a slab and use its floppy over ethernet.

Sounds outrageous? Perhaps not. I'd propose for people to instead of upgrading their motherboard, to use that money and buy a slab. Just the slab, no monitor, no keyboard. NeXT lists an 8M slab with 105M hd (the 105/8 configuration) for $3,775. The academic/developer discount is 30% off, or $2643.

Let's say the 105M hard disk is worth $800, conservatively. It comes pre-loaded with System 2.0. The 8M RAM we will price at $50 / Meg, or $400. That's about the best price you'd expect if you shopped around and haggled.

Just by counting components, you can see that the slab is compared against a bare 68040 board plus system software ($995), plus 8M of RAM ($400), plus an external 2.88 floppy ($499), plus a 105M hard disk ($800), for a total of -- guess what, $2694. Now compare that to your price for a slab-only machine, $2643, plus $50 cabling, or $2693.

Going beyond component counting, let's look at flexibility. By having two CPUs, you can effectively parallel process much of the system load onto two systems. You see, in the component count, we didn't even figure in the cost of the 68030 you just gave away, or the supporting circuitry that gave you more expandability.

You could use the 68030 cube as your print/fax/file server. You know how much it slows down your system whenever you print. Faxing is about the same way too, except it uses less memory, so doesn't havesuch a big impact (rasterization of PostScript at 200 dpi max vs. 400 dpi max for laser printing). Offloading all file I/O means you can copy a whole OD, and still compute away on your slab at full tilt.

The end result is you have a faster system. Much faster, in fact, than a 68040 alone can be. Yes, you now have "true mainframe throughput" -- your 68030 is your I/O slave!

You also get more expandability. You now have four serial ports, instead of two. I know I sure can use more myself. You can add up to fourty SCSI devices in total now, instead of seven (even though I've never seen anyone use up all seven SCSI device spaces, I just know -- one day -- you'll need it). You get two DSP ports, so one can actually be dedicated full time to, say, Digital Eyes, doing security surveilance.

Notice the slab comes with SCSI-2 and not the DB-25 SCSI-1 connector. Instead of buying a SCSI-2 to SCSI-1 adaptor ($49 from NeXT), you could save that money by putting everything on the cube. Scanners, tape drives, external hard drives, whatever.

If you have people who log in remotely, or have telnet running, you can also offload that task to the 68030 cube.

You also get portability. You can take the whole slab and move it off-site, much easier than taking the cube.

Of course, you can always upgrade the 68030 at a later time. NeXT has put no time limit on its offer, except the one inducement for quick upgrade, a free copy of Improv. However, you'd get that when you buy your slab anyways.

The only down side is the more complicated boot procedure. You must boot the 68030 first with the keyboard and monitor attached (since the power button is routed through the monitor cable), then move the cable over to the slab, and then power that on. The 68030 will run happily without a monitor, though it will not have any audio output, since the monitor also houses the speaker.

If having audio output is supremely important, you should pick up a Sound Box from NeXT, for $88 to developers. Or it may make for an entertaining hardware project, to cable in your own set of speakers.

When some time down the line, you find yourself needing to add a second computer, you will pleasantly find the only cost is an extra MegaPixel and keyboard ($1220 X 70% = $854). Show that to people shopping for X-terminals, and watch them die of envy.

Bits & Pieces

* The new NeXTs use RS-423 instead of RS-422 serial ports. RS-423 uses one line for data, instead of two, saving one pin to be used for hardware flow control. It looks like NeXT listened to the user community and acted to remedy a mistake. Hooray! NeXT has reaffirmed itself, in big ways and in small ways, as a company that listens.

* What's SCSI-2? Why change from SCSI-1? SCSI-2 is a tighter specification that resolves ambiguities from the original SCSI-1 draft; it also gives better ways for devices to intercommunicate. So with the right software, you can for example get your hard disks to talk directly together for block transfers, without interrupting the CPU, once the command is initiated.

* Price on the laser has been dropped, from $2000 to $1396 for academia and developers. The 400 dpi laser is now an even better bargain.

* The light-weight slab design opens new speculations on a portable NeXT. If you look at the inside, you will find a compact power supply, fan, hard disk and floppy, a small motherboard, and lots of space. There's space behind the floppy drive, around the hard disk, behind the power supply. RAM can be laid at an angle to save even more space. A laptop engineer, who has seen the interior of the slab, assured me, "it's a simple matter to squeeze the whole setup into 2/3 of what it takes up now". If we throw the floppy drive away, perhaps even by 1/2.

Basically the only obstacle now is market demand. Until the NeXT is a mass-market phenomenon like Macs and PCs, there won't be enough buyers for portables to make it worth NeXT's while. For now, people can transport the slab between a monitor/keyboard at work, and at home.

* With NeXT selling extended capacity floppies at $8 a shot, many vendors will resort to distributing on 1.44M floppies, which are about $1 a disk to consumers.

* Tilt feet for the keyboard, as predicted in Tao #1, is now bundled.

* Where's Renderman?? According to sources at NeXT, Renderman is being worked on for release in first quarter, 1991. Though originally intended for September 18th release, it was still not ship-shape, so was delayed.

So far, the engineering field has yet to be fully addressed. We still need both renderman and some good CAD packages to round out the NeXT.

* The hottest item at the unveiling was the NeXTdimension board. The i860 (reported in Tao #5) runs PostScript at blinding speed, using its own 8M of standard RAM, expandable up to 32M. When Renderman is released, chances are very good that it, too will run on the i860 for extra speed. Renderman under the NeXTstation Color may be quite slow by comparison.

* With the JPEG real-time compression capability of NeXTdimension, it can now play real-time video directly from hard disk. The power that this opens up is so enormous, it boggles the mind. NeXT is the first consumer computer with JPEG capability. You can now edit movies and mix special effects that were previously the sole domain of Hollywood studios. Imagine filming your own video of two men fighting with stick swords, over blue background. Now run a special f/x program, identify the stick swords from frames 0:00 to 3:45 and substitute with light sabres. Now film an ant hill at high magnification, and overlay it into the frames' blue background. Touch up the feet where they meet the sand with a paint brush. With a finished special effect movie, you can now output to a VCR and record it for distribution.

Imagine wiring up the building with NeXTdimension cubes. Now aim a camera at yourself, fed into the

NeXTdimension's composite video input. Transmit image over ethernet, and you have real-time, video conferencing within the whole building. With real-time compression, even low-bandwidth devices like ethernet, with its piddly 10Mbits per second, can handle video conferencing. To take it behind your building, use ISDN to achieve world-wide video conferencing.

* Heard on the net: EMACS stands for Eight Megabytes And Constantly Swapping.

Slab Users Beware : For first time buyers of NeXT, who are arguably the luckiest people on the planet, the slab will be a natural choice. The price is right,the speed is quick, and software is pre-loaded. But what happens when the disk gets trashed? Boot from 2.88 M floppy? Very unlikely. There may be no good way to rebuild the hard disk, short of installing from scratch off 30 floppies. My idea of a nightmare. Most people won't even have the floppy backup, since NeXT asks for $225 extra for these, and they aren't bundled. The situation calls for an external SCSI drive. For a paltry $1200, one can buy an external 330M SCSI, with case and cable. That should be the boot device, with its secondary image of System 2.0.

The more paranoid (reads: security conscious) users will detach the internal 105M SCSI and put it into a safe. Most of us should simply boot off the external SCSI and use the internal as a /tmp drive. The more daring (and less well-heeled) may want to delete all but the basic bootable System 2.0, to get more storage.

It's important to have a bootable hard disk in case of disaster. Say the boot block gets wiped out, but everything else is basically sound. You can boot from the internal hard disk and recover all your files, instead of throwing everything away, and wasting several hours rotting your arm away with floppy swapping.

RS/6000 the Schizoprenic Chip

The first hint that all was not well came in UNIX World, when the benchmarkers found the RS/6000's multi-user performance quite dismal. Neal Nelson & Associates reported the POWERserver 320 is only "comparable to the best of 80486 PCs we have tested." (Sept 1990, p. 13). In fact, though it was 20% faster than an HP Vectra 486 in I/O operations, it was 20% slower in floating-point and in integercalculations. This, after a glitzy blitz from IBM, with super impressive SPECmark numbers and megaflops.

Then, slowly, other reports start to come in. One developer found the X-windows performance to be equal or below his 386, using a "no-big-deal" 16-bit VGA that is rather common. "The load time on the RS/6000 was faster, but after that the 386 is as just fast, and sometimes faster, than the RS/6000".

Did IBM lie about the SPECmark? To get to the bottom of all this, I talked to an IBM developer, who was closely associated with the design of RIOS. According to him, "the RS/6000 is a schizoprenic chip. Though it excells in floating point operations, it really slows down with bitwise operations".

In addition, AIX is so big that with 8M of memory, there is hardly anything left to programs. You wouldn't want to be running EMACS on a stock RIOS.

What about the benchmark numbers? Were they exaggerated? "No, but when they did the benchmarks, they found some that were very unflattering. Those, they decided to leave out," according to the IBM developer. If you want to run multi-users, you'd best stick to an AT&T 3B2 or a 386. You'll be a lot happier that way.

Now here comes the scary part. It almost sounds as if IBM wants to make sure the RS/6000 does not encroach their big bread & butter, traditional multi-user computer: AS/400. Perhaps the multi-user performance was deliberately sabotaged?

And yet, by endorsing every open standard there is, IBM can now pay lip service to "Open Systems". The low prices they initially advertised turned out to be unavailable here in Canada, and with their proprietary memory chips, they know they'll milk you good before you can get a system that is actually useful.

All this leads me to believe that IBM has done it again. They'll sell plenty of RS/6000's, and they'll make lots of money, and people who buy into RS/6000 will go back for more. When people realized they aren't meant as multiuser servers, IBM will be there with AS/400.

Advice To NeXT

1. Technology

The time is now to establish a multi-CPU architecture. With the 68040 now in place, NeXT must move quickly produce an upgrade path that does not sacrifice binary compatibility. I've reported that there's an 88000 on the drawing board, that prototypes have already been built. But I don't feel that's the way to go, not now and not yet.

The biggest problem is still binary compatibility. A small company like NeXT can ill afford to support two binary systems. They are stretched as it is with the rock-bottom pricing.

What NeXT needs to do now, is to leverage the MACH advantage. Some may argue that there isn't sufficient need for multi-CPU NeXT, but that's just like saying there isn't the need for a 040 upgrade. People can always use speed and more speed. One particular area where multi-CPU architecture wins big, is an Application Server.

The guys who wrote UNIX are now hard at work on Plan 9, a new operating system that employs the Application Server idea. They call it a CPU server, which runs programs but does no display. The output goes into a "Gnot", or an intelligent display terminal, that recognizes an X-windows-like protocol. Such an approach makes a lot of sense for both theoretical and business reasons.

The hackers at AT&T will tell you all about the theoretical ones, where maximum utilization of system resource is made, where linear upgradability of system performance comes cheap, where improved inter-process communications (in memory instead of over network) increasesperformance.

The business reasons are euqally important. A NeXTStep terminal can be built inexpensively, without hard disk and with minimal memory. If NeXT can bring out a terminal for under $1500, it will make for a pursuasive business solution for multi-users.

That pricing point helps NeXT win large government and corporation contracts, which typically require many "seats". A low cost per seat, added to high multi-user performance, spells contracts.

NeXT should not wait for Motorola to unveil their 68050. As usual they will be late. Historically, Sun left Motorola in frustration and turned to SPARC. It turned out to be a wise decision in the long run, since Sun now has control of the CPU architecture, but it takes a company the size of Sun or IBM to underwrite the expensive CPU development cycle. It takes a lot of sales to support a brand new CPU.

NeXT doesn't have the size or the sales volume for their own CPU. NeXT can't afford to take on some other RISC chip, or else they will have support and maintenance logistic problem to plague them for years. NeXT can't afford not to have higher end machines, or they won't ever penetrate the top segment of government contracts, the truly lucrative multi-million deals.

So the answer is clear: a multi-68040 architecture. Bring out a new product line that replaces the old motherboard with scalable power. Add an optional fiber optics ethernet connector, and put FDDI support into the kernel. When you have 1,000 Display Postscript terminals hanging off one 32 CPU server, you better have the network bandwidth to handle it all.

The fiber optic ethernet should be able to run simultaneous with the thinnet, which should be connected to an NFS server. This NFS server holds gigabytes of storage, something like the Epoch-1 InfiniteStorage Server on page 60 of the Fall 1990 catalog.

By separating the NFS server from the terminal network, you reduce congestion on the terminal net. The terminal net and the NFS net both need to burst at the same time; when a user views an SQL table, for example, both the terminal fiber net and the NFS ethernet goes active momentarily.

The primary cost component in a NeXTStep server is the monitor. By offering a 14" monochrome version (using half toning to achieve grey scale effects, the same as Macintosh) at a reduced price, NeXT can easily achieve the $1500 price point, and even go beyond. To give you an idea of the costs involved, look at the pricing for any 386 machine "bare" (without hard disk) using monochrome VGA display, and add an ethernet card. Astute buyers pay $800 or less. A final reason for having a multi-CPU server is upgradability. Large corporations that buy into a line, must consider the scalability factor. The decision makers ask themselves, "If my company's needs grow, will I have to junk my hardware and software investment and go to another company, or can I stay with NeXT and expect their hardware to keep up?".

2. Marketing

NeXT needs to expand their distribution channel to include Value Added Resellers (VARs).

Right now, the biggest hole in the marketing plan is a lack of demand. Not software, not hardware, just plain lack of demand. NeXT needs energetic, motivated people with lots of know-how in particular areas, pushing their box as The Choice.

Sun pursued the same strategy with great success. It didn't matter that Suns could not be purchased from BusinessLand; the people who bought wanted solutions, not hardware pieces. They wanted vertical packages and they wanted after sales support. That's the role VARs fill.

In any successful VAR plan, you've got to have strong leadership from the company, good support to the VARs, and enough exclusivity so that price slashing wars don't break out. NeXT must seek out VARs who have identifiable niches, who have expertise in those niches; the number of VARs in a geographical area isn't an issue, but having well-defined VAR strategy is.

On the issue of support, NeXT needs to contract a field representative to handle repairs. Organizations such as Bell and GE and Xerox, all offer contracts for field repairs. NeXT has to provide the support that corporations demand.

3. Future Directions

NeXT needs to legitimize X as an alternative display system. Right now X exists as unofficial software from MIT. Its flaky status deters government buyers, who knows NeXT is not serious about X. Well, X happens to be a FIPS standard now, one established by the government standards organization.

The US government accounts for more than 10% of each year’s UNIX purchase volume. NeXT has to get serious about supporting X, if it wants to invade that market.

Getting serious means officially supporting X. Distasteful as that may be to some, it’s necessary if NeXT wants a bigger slice of the pie.

By sneaking in NeXTStep through the "back-door", at least there’s a chance for Uncle Sam to also formally adopt NextStep as an alternative standard display system.

View from Educom

David Carpenter
dcarpent@sjuphil.UUCP
St. Joseph’s University
Philadelphia, PA 19131

Educom’s annual conference on information technology in higher education was held this year in Atlanta, GA, hosted by Georgia Tech (home of BuzzNUG). It was a great conference, and NeXT was there in force. They ran both a display of some 12 machines in the vendor exhibit area and anothersix or so machines in their hospitality suite. They had the entire new product line present, including the NeXTDimension color system: beautiful! Steve Jobs gave the closing talk, "Interpersonal computing for the 90s," which turned out to be a version of the demo he gave at the introduction of the new products in San Francisco. An impressive performance, though more of a sustained advertisement for his wares than a keynote address. The audience seemed appreciative, however.

NeXT seemed successful in generating a lot of interest and enthusiasm. As the conference wore on, the new black turtleneck NeXT T-shirts appeared in greater and greater numbers, and people seemed interested in the new NeXT. Jobs’ speech, which was the last event of the conference, was well attended.

A few things I learned, which may or may not come as news to others:

1. OCR software is now available from HSD, the people who make the grey-scale scanner for the Cube. It is added automatically to the "services" menu item so that OCR becomes a kind of standard, system-wide service (although it must be purchased separately).

2. Database kits are on the way, i.e., objects for interfacing with either the Sybase or Oracles database servers. They were being demonstrated at the conference, and I was told that they would be available fairly soon at a low price.

3. NeXT will probably not be supporting SLIP in the foreseeable future. I was told that they are aware of the problem of unnetworked cubes, and that a solution would be made available at some point, but that support of SLIP was not in the works.

4. The upgrades will evidently ship with dust filters that will extend the life of the OD drive by several years. The third-party filters already available tended to raise the temperature inside the cube to unacceptable levels. The NeXT filters do not, and are safe.

5. I got the impression frotalking to some NeXT people that the 2.0 release of the system software had been hardware driven, and was too rushed to include many of the enhancements that NeXT wants to make. So we can look forward to really goodhings (on the level of system software) in 3.0.

All of the above is based on casual conversations with people from NeXT who were at Educom. I make no guarantee for its accuracy, nor do I have any "inside" information.

Other subjective impressions:

- Improv made a big hit, and NeXT was really pushing it.
- To an untrained eye, the NeXTStation color is hard to distinguish from the NeXTDimension color, i.e., the 16 bit color looks really good.
- As usual, there are a lot of "extras" in 2.0, including a nice calendar/scheduler that I may well end up prefering to Calendoscope.
- The whole mood was upbeat, both among the people from NeXT and among those who were seeing the machines for the first time. The NeXT hospitality suite was jammed. I left with a very good feeling about NeXT’s prospects for long-term success. People were talking about buying the machines; many said they already had them on order. In short, things seem to be looking up.

Displaying NeXT Graphics with NCSA Telnet for the Macintosh

Eric P. Scott
eps@toaster.sfsu.edu

San Francisco State University has a diverse, multivendor environment, and a mandate to “make everything talk to everything else.” Thus, we installed a flexible campuswide network, and adopted TCP/IP as the protocol suite of choice. NeXT workstations fit naturally in our plan, since they already have the requisite hardware and software. The bigger problem was thousands of extant microcomputers that weren’t designed with networking in mind.

We “got lucky” with the Apple MacintoshÒApple promoted a standardized Ethernet API, and their machine didn’t have the address space limitations that cripple MS-DOS systems. And we were fortunate that the National Center for Supercomputing Applications (NCSA) at the University of Illinois, Urbana-Champaign had created the premier network application: NCSA Telnet. This let our Macintoshes serve as terminals to serious computing resources. It also let us copy files between Mac floppies and NeXTs, and IBMs, and everything else.

With Apple’s commitment to support TCP/IP through its MacTCP and A/UX products, we have software that integrates the major network services within the Mac interface. Products like the Shiva FastPath 4 and Cayman Gatorbox let us use lower cost AppleTalk networks where performance isn’t critical. Many of our NeXTs serve as focal points for Mac users, e.g. providing e-mail access with SMTP and POP servers, or just being “normal UNIX boxes.” Despite NeXT’s notoriety for its excellent GUI, we’ve found NeXTs popular and cost-effective for traditional timesharing, and the Macintosh one of the best “terminals” for remote use.

Getting here from there

In July, 1989, the NCSA released version 2.3 of NCSA Telnet for the Macintosh, and with it, the capability to display Interactive Color Raster (ICR) graphics. This capability went largely ignored, since there weren’t many applications that took advantage of it. Meanwhile, in the NeXT arena, TIFF files seemed to be everywhere. NeXT Mail wanted pictures of our users, and we could only generate them using an Apple Scanner attached to a Mac II, then transferring the image files to the NeXT with the FTP server built into NCSA Telnet. AppleScan didn’t produce quite what the NeXT wanted, so I wrote a simple program called tifftoeps that converted grayscale TIFF to Encapsulated PostScript. Recently, I modified that same program to convert TIFF to ICR, so images can now travel in the reverse direction: from NeXT to Mac.

I wanted to present a “working demo” that others could use as a starting point for further development, and given the ready supply of TIFF files in Digital Webster, this seemed like an obvious choice. I used the webster(3) routines out of sheer laziness—they’re not powerful enough to implement a “full” Webster application. They’re buggy, too. If the “picture” string in the returned Definition structure is filled-in, it’s supposed to be the pathname of the accompanying illustration. As it turns out, the library routines aren’t as bright one might expect, and a word having multiple definitions usually claims to have more than one picture. Also, some words have EPSF illustrations rather than TIFF, but the “picture” name always seems to have .tiff appended. Finally, some illustrations have captions in Rich Text Format. I’m only concerned with the TIFF files.

Webster’s TIFF files use a compression scheme that NeXT’s standard TIFF routines (NXGetTIFFInfo, NXReadTIFF) won’t process, hence the reinvented wheel. Although my routines will handle most “normal” 1-, 2-, 4-, and 8-bit deep grayscale TIFF files, they’re a far cry from a “general” TIFF-handling package. Thankfully, one already exists—Sam Leffler’s libtiff, and I am indebted to him for the use of his decompression code in my demonstration.

How ICR works

ICR uses a variation of the ANSI Privacy Message sequence to embed graphics data in the output stream. ICR is not terribly sophisticated. Basically, it allows creation and destruction of named graphics windows, and drawing arbitrary horizontal segments in those windows.

New windows start with an 8-bit grayscale palette. Most of the time I’m not displaying 8-bit deep images, so I load a “more reasonable” color pallete containing 16 levels of gray. You’ll see the colors change onscreen as this happens.

ICR drawing is slow! This is primarily due to the relatively large amount of data required to render a complete image, and an encoding method that forces even 2-bit graphics to require two bytes per pixel. Fortunately, ICR allows scan lines to be run-length compressed, and most images benefit greatly from this.

Is it useful?

I don’t know, but it sure impresses Mac users. I fear this will be another one of those “I wrote it as a joke but everyone uses it” programs. Now that I’ve opened a Pandora’s box, someone might find some good in it.

If you don’t have NCSA Telnet yet

NCSA Telnet is in the Public Domain. Internet sites may obtain NCSA Telnet free of charge by anonymous FTP from ftp.ncsa.uiuc.edu in the NCSA_Telnet/Mac directory*. UUNET member sites can obtain it via uucp, and non-member US sites can use UUNET’s 1-900-GOT-SRCS anonymous UUCP service (40¢ per minute). NCSA software and documentation can also be ordered directly from the University of Illinois; request a catalog for current offerings and prices.

NCSA Documentation Orders
152 Computing Applications Building
605 East Springfield Avenue
Champaign, IL 61820
(217) 244-0072

*I actually use the slightly modified version called BYU Telnet, in the NCSA_Telnet/contributions/BYU_TELNET directory.

Sam Leffler’s libtiff is available by anonymous FTP from ucbvax.berkeley.edu in the pub/tiff directory (and from UUNET in the graphics directory).

```
‹/*
 * icrwebster - display webster’s images with NCSA Telnet 2.3
 * Eric P. Scott, San Francisco State University, October 1990
 *
 * Copyright 1990 by Eric P. Scott except as noted below.
 *
 * Credit where credit is due-ware: recognize what I’ve
 * contributed, take responsibility for what you do with/to it.
 * (i.e. don’t pass my work off as your own, or vice-versa.)
 * You are granted a nonexclusive, royalty-free license to use
 * and adapt this software, and you may redistribute it in whole
 * or in part, in source and/or binary forms.
 *
 *
 * This is an exercise in code reusability; several existing
 * code fragments were blended together to produce this
 * monstrosity.  It’s not intended to be elegant, it’s just a
 * little experiment I thought I’d share with the net.community.
 *
 * You need a Mac II with an 8-bit color display to try this--
 * see Chapter 7 in the NCSA Telnet 2.3 documentation.
 */
#include <stdio.h>
#include <text/webster.h>
#ifndef lint
static char sccsid[]="@(#)icrwebster  1.0  (SFSU) 10/20/90";
#endif
ReferenceBook *book;
int thesaurus=0;
char *wname;
main(argc, argv)
int argc;
char *argv[];
{
	char *malloc(), *rindex();
	void lookup();
	extern int optind;
	extern char *optarg;
	register int i;
	char line[1024];

	dictionaryFoldSenses=60;
	while ((i=getopt(argc, argv, "tw:"))!=EOF) switch (i) {
	case ’t’:
		thesaurus++;
		break;
	case ’w’:
		if (*optarg>=’0’&&*optarg<=’9’) {
			dictionaryFoldSenses=atoi(optarg);
			break;
		}
		/*FALL THROUGH*/
	default:
		(void)fprintf(stderr,
			"Usage: %s [options] [word [...]]\n", *argv);
		(void)fputs("\t-t  use Thesaurus\n\t-w# set width\n", stderr);
		exit(0);
		break;
	}
#ifdef __STRICT_BSD__
	(void)setbuffer(stdout, malloc(4096), 4096);
#else
	setvbuf(stdout, (char *)NULL, _IOFBF, 4096);
#endif
	if (optind<argc) do {
		if (wname=rindex(argv[optind], ’/’)) {
```

```
            wname++;
            (void)image(argv[optind]);
        }
        else lookup(argv[optind]);
    } while (++optind<argc);
    else for (;;) {
        (void)fputs("Word: ", stderr); fflush(stderr);
        if (!fgets(line, sizeof line, stdin)||line[0]==’\n’) break;
        line[strlen(line)-1]=’\0’;
        lookup(line);
    }
    if (book) (void)referenceClose(book);
    exit(0);
}

void lookup(word)
char *word;
{
    register int i;
    int n;
    Definition *D[16];      /* how big is big enough? */

    if (!book) {
        if (!(book=referenceOpen(thesaurus ? "Webster-Thesaurus" :
            "Webster-Dictionary"))) {
            fputs("referenceOpen failed!\n", stderr);
            exit(1);
        }
    }
    bzero((char *)D, sizeof D);
    n=getDefinitions(word, book, 1, D, sizeof D/sizeof D[0]-1);
    if (n==0) {
        (void)fputs("No definition for ", stdout);
        (void)fputs(word, stdout);
        (void)fputs(".\n", stdout); (void)fflush(stdout);
        return;
    }
    i=0; do {
        if (i>1) putchar(’\n’);
        if (D[i]->section>0&&!thesaurus) (void) printf("%% %s\n\n",
            websterSectionName[D[i]->section]);
/*
 * The webster routines in libtext.a (1.0/1.0a) are really braindamaged.
 * I can’t make use of font information since there’s no setting for
 * dictionaryOutputFormat that doesn’t trash stuff I need (why isn’t
 * there a W_RAW???) and I don’t feel like picking though the
database
 * by hand.  If you want a decent webster program, check out Steve
 * Hayman’s--his way is faster, too.  (And it does spell checking!)
 */
        (void)putDefinition(D[i], 0);
        putchar(’\n’);
        if (D[i]->picture&&
            D[i]->picture[strlen(D[i]->picture)-1]==’f’) {
            (void)fflush(stdout);
            wname=D[i]->dotted ? D[i]->dotted : D[i]->entry;
            (void)image(D[i]->picture);
        }
    } while (++i<n);
    (void)fflush(stdout);
    freeDefinitions(D);
}

int iwidth, ilen, bits, invert, compression;
long stripoff;
int image(tfile)
char *tfile;
{
    void NULLDecode(), NeXTDecode();
    FILE *tf;
    register int i;
    register long value;
    long sig, ifdoff;
    short ifditems;
    struct ifdent {
        short ifd_tag;
        short ifd_type;
        long ifd_len;
        union {
            unsigned long ifd_long;
            unsigned short ifd_short;
            unsigned char ifd_byte;
        } ifd_off;
    } *ifdp;

    if (!(tf=fopen(tfile, "r"))) {
        perror(tfile);
        return(1);
    }
    if (fread((char *)&sig, sizeof sig, 1, tf)!=1) {
        perror("fread");
    fatal:
```

```
    fclose(tf);
    return(1);
}
if (sig!=0x4d4d002a) { /* big-endian */
    (void)fprintf(stderr, "%s: that’s not a TIFF file!\n", tfile);
    goto fatal;
}
iwidth=0; ilen=0; bits=0; invert=255; compression=1;
stripoff=0L;
if (fread((char *)&ifdoff, sizeof ifdoff, 1, tf)!=1) {
    perror("fread");
    goto fatal;
}
if (fseek(tf, ifdoff, 0)<0L) {
    perror("fseek");
    goto fatal;
}
if (fread((char *)&ifditems, sizeof ifditems, 1, tf)!=1) {
    perror("fread");
    goto fatal;
}
if (!(ifdp=(struct ifdent *)malloc(ifditems*sizeof (struct ifdent)))) {
    perror("malloc");
    goto fatal;
}
if (fread((char *)ifdp, sizeof (struct ifdent), ifditems, tf)!=
    ifditems) {
    perror("fread");
    goto fatal;
}
for (i=0;i<ifditems;i++) {
    switch (ifdp->ifd_type) {
    case 1:
        value=ifdp->ifd_off.ifd_byte;
        break;
    case 3:
        value=ifdp->ifd_off.ifd_short;
        break;
    case 4:
        value=ifdp->ifd_off.ifd_long;
        break;
    default: /* ifdp->ifd_off.ifd_long is offset */
        value=0L;        /* do The Wrong Thing(tm) */
        break;
    }
    switch (ifdp->ifd_tag) {
```

```
        case 256:    /* ImageWidth */
            iwidth=value;
            break;
        case 257:    /* ImageLength */
            ilen=value;
            break;
        case 258:    /* BitsPerSample */
            bits=value;
            break;
        case 259:    /* Compression */
            if (value!=1L&&value!=32766L) {
                (void)fprintf(stderr,
                    "%s: unsupported Compression=%ld\n",
                    tfile, value);
                goto fatal;
            }
            compression=value;
            break;
        case 262:    /* PhotometricInterpretation */
            switch (value) {
            case 0L:
                invert=255;
                break;
            case 1L:
                invert=0;
                break;
            default:
                (void)fprintf(stderr,
"%s: unsupported PhotometricInterpretation=%ld\n", tfile , value);
                switch ((int)value) {
                case 5:    /* grayscale alpha */
                case 6:    /* RGB alpha */
                case 7:    /* CMY or CMYK alpha */
                case 14: /* 8-bit Pcolor alpha */
                    (void)fputs("\tI don’t grok alpha\n",
                        stderr);
                    break;
                case 2:    /* RGB */
                case 3:    /* CMY or CMYK */
                case 10: /* 8-bit Pcolor */
                    (void)fputs("\tNo color support!\n",
                        stderr);
                    break;
                }
                goto fatal;
            }
            break;
```

```
        case 273:       /* StripOffsets */
            stripoff=value;
            break;
        case 277:       /* SamplesPerPixel */
            if (value!=1L) {
                (void)fprintf(stderr,
"%s: unsupported SamplesPerPixel=%ld\n", tfile, value);
                goto fatal;
            }
            break;
#ifdef NOTDEF
        case 284:       /* PlanarConfiguration */
            /* don't care when SamplesPerPixel==1 */
            break;
#endif
        default:
            break;
        }
        ifdp++;
    }
    if (iwidth<=0) {
        (void)fprintf(stderr, "%s: missing ImageWidth\n", tfile);
        goto fatal;
    }
    if (ilen<=0) {
        (void)fprintf(stderr, "%s: missing ImageLength\n", tfile);
        goto fatal;
    }
    if (bits<=0) {
        (void)fprintf(stderr, "%s: missing BitsPerSample\n", tfile);
        goto fatal;
    }
    if (stripoff<=0L) {
        (void)fprintf(stderr, "%s: missing StripOffsets\n", tfile);
        goto fatal;
    }
    if (fseek(tf, stripoff, 0L)<0L) {
        perror("fseek");
        goto fatal;
    }
#ifdef DEBUG
    (void)fprintf(stderr, "imaging %dx%dx%d\n", iwidth, ilen, bits);
#endif
    if (compression==1) NULLDecode(tf);
    else if (compression==32766) {
        if (bits==2) NeXTDecode(tf);
```

```
        else (void)fprintf(stderr,
"%s: NeXT compressions only supported for 2-bit images\n", tfile);
    }
    (void)fclose(tf);
    return(0);
}

void NULLDecode(tf)
FILE *tf;
{
    void output();
    char buf[284];
    int scanline, row;

    scanline = ((long)iwidth*bits+7L)/8L;
    for (row=0; row<ilen; row++) {
        (void)fread(buf, 1, scanline, tf);
        output((unsigned char *)buf, row);
    }
}

/* This table derived from NCSA Public Domain material. */
static unsigned char rgb[768]={
    255, 255, 255, 255, 255, 204, 255, 255, 153,
    255, 255, 102, 255, 255,  51, 255, 255,   0,
    255, 204, 255, 255, 204, 204, 255, 204, 153,
    255, 204, 102, 255, 204,  51, 255, 204,   0,
    255, 153, 255, 255, 153, 204, 255, 153, 153,
    255, 153, 102, 255, 153,  51, 255, 153,   0,
    255, 102, 255, 255, 102, 204, 255, 102, 153,
    255, 102, 102, 255, 102,  51, 255, 102,   0,
    255,  51, 255, 255,  51, 204, 255,  51, 153,
    255,  51, 102, 255,  51,  51, 255,  51,   0,
    255,   0, 255, 255,   0, 204, 255,   0, 153,
    255,   0, 102, 255,   0,  51, 255,   0,   0,
    204, 255, 255, 204, 255, 204, 204, 255, 153,
    204, 255, 102, 204, 255,  51, 204, 255,   0,
    204, 204, 255, 204, 204, 204, 204, 204, 153,
    204, 204, 102, 204, 204,  51, 204, 204,   0,
    204, 153, 255, 204, 153, 204, 204, 153, 153,
    204, 153, 102, 204, 153,  51, 204, 153,   0,
    204, 102, 255, 204, 102, 204, 204, 102, 153,
    204, 102, 102, 204, 102,  51, 204, 102,   0,
    204,  51, 255, 204,  51, 204, 204,  51, 153,
    204,  51, 102, 204,  51,  51, 204,  51,   0,
    204,   0, 255, 204,   0, 204, 204,   0, 153,
```

```
204,   0, 102, 204,   0,  51, 204,   0,   0,
153, 255, 255, 153, 255, 204, 153, 255, 153,
153, 255, 102, 153, 255,  51, 153, 255,   0,
153, 204, 255, 153, 204, 204, 153, 204, 153,
153, 204, 102, 153, 204,  51, 153, 204,   0,
153, 153, 255, 153, 153, 204, 153, 153, 153,
153, 153, 102, 153, 153,  51, 153, 153,   0,
153, 102, 255, 153, 102, 204, 153, 102, 153,
153, 102, 102, 153, 102,  51, 153, 102,   0,
153,  51, 255, 153,  51, 204, 153,  51, 153,
153,  51, 102, 153,  51,  51, 153,  51,   0,
153,   0, 255, 153,   0, 204, 153,   0, 153,
153,   0, 102, 153,   0,  51, 153,   0,   0,
102, 255, 255, 102, 255, 204, 102, 255, 153,
102, 255, 102, 102, 255,  51, 102, 255,   0,
102, 204, 255, 102, 204, 204, 102, 204, 153,
102, 204, 102, 102, 204,  51, 102, 204,   0,
102, 153, 255, 102, 153, 204, 102, 153, 153,
102, 153, 102, 102, 153,  51, 102, 153,   0,
102, 102, 255, 102, 102, 204, 102, 102, 153,
102, 102, 102, 102, 102,  51, 102, 102,   0,
102,  51, 255, 102,  51, 204, 102,  51, 153,
102,  51, 102, 102,  51,  51, 102,  51,   0,
102,   0, 255, 102,   0, 204, 102,   0, 153,
102,   0, 102, 102,   0,  51, 102,   0,   0,
 51, 255, 255,  51, 255, 204,  51, 255, 153,
 51, 255, 102,  51, 255,  51,  51, 255,   0,
 51, 204, 255,  51, 204, 204,  51, 204, 153,
 51, 204, 102,  51, 204,  51,  51, 204,   0,
 51, 153, 255,  51, 153, 204,  51, 153, 153,
 51, 153, 102,  51, 153,  51,  51, 153,   0,
 51, 102, 255,  51, 102, 204,  51, 102, 153,
 51, 102, 102,  51, 102,  51,  51, 102,   0,
 51,  51, 255,  51,  51, 204,  51,  51, 153,
 51,  51, 102,  51,  51,  51,  51,  51,   0,
 51,   0, 255,  51,   0, 204,  51,   0, 153,
 51,   0, 102,  51,   0,  51,  51,   0,   0,
  0, 255, 255,   0, 255, 204,   0, 255, 153,
  0, 255, 102,   0, 255,  51,   0, 255,   0,
  0, 204, 255,   0, 204, 204,   0, 204, 153,
  0, 204, 102,   0, 204,  51,   0, 204,   0,
  0, 153, 255,   0, 153, 204,   0, 153, 153,
  0, 153, 102,   0, 153,  51,   0, 153,   0,
  0, 102, 255,   0, 102, 204,   0, 102, 153,
  0, 102, 102,   0, 102,  51,   0, 102,   0,
  0,  51, 255,   0,  51, 204,   0,  51, 153,
      0,  51, 102,   0,  51,  51,   0,  51,   0,
      0,   0, 255,   0,   0, 204,   0,   0, 153,
      0,   0, 102,   0,   0,  51, 238,   0,   0,
    221,   0,   0, 187,   0,   0, 170,   0,   0,
    136,   0,   0, 119,   0,   0,  85,   0,   0,
     68,   0,   0,  34,   0,   0,  17,   0,   0,
      0, 238,   0,   0, 221,   0,   0, 187,   0,
      0, 170,   0,   0, 136,   0,   0, 119,   0,
      0,  85,   0,   0,  68,   0,   0,  34,   0,
      0,  17,   0,   0,   0, 238,   0,   0, 221,
      0,   0, 187,   0,   0, 170,   0,   0, 136,
      0,   0, 119,   0,   0,  85,   0,   0,  68,
      0,   0,  34,   0,   0,  17, 238, 238, 238,
    221, 221, 221, 187, 187, 187, 170, 170, 170,
    136, 136, 136, 119, 119, 119,  85,  85,  85,
     68,  68,  68,  34,  34,  34,  17,  17,  17,
      0,   0,   0
};

static unsigned char idx1[2]={ 255, 0 }, idx2[4]={ 255, 251, 248, 0 },
    idx4[16]={ 255, 254, 253, 172, 252, 251, 129, 250,
    249, 86, 248, 247, 43, 246, 245, 0 };

int ox, oy;
void output(buf, row)
unsigned char *buf;
int row;
{
    register unsigned char *p, *q;
    register int i, c;
    unsigned char pbuf[1132];
#ifndef NO_RLE
    unsigned char rbuf[1144];
#endif

    if (row==0) {
        (void)printf("\033^W;%d;%d;%d;%d;0;%s^", ox, oy, iwidth,
ilen,
            wname);
        ox+=5; oy+=3;
        if (bits<8) {
            (void)printf("\033^M;0;256;768;%s^", wname);
            p=rgb; q=p+sizeof rgb; do {
                if (*p>31&&*p<123) putchar(*p++);
                else {
                    putchar((*p>>6)+123);
                    putchar((*p++&63)+32);
```

```
            }
        } while (p<q);
    }
    (void)fflush(stdout);
}
p=buf; q=pbuf;
switch (bits) {
case 1:
    for (i=0;i<iwidth;i++) {
        c= *p++^invert;
        *q++ = idx1[(c>>7)&1];
        if (++i>=iwidth) break;
        *q++ = idx1[(c>>6)&1];
        if (++i>=iwidth) break;
        *q++ = idx1[(c>>5)&1];
        if (++i>=iwidth) break;
        *q++ = idx1[(c>>4)&1];
        if (++i>=iwidth) break;
        *q++ = idx1[(c>>3)&1];
        if (++i>=iwidth) break;
        *q++ = idx1[(c>>2)&1];
        if (++i>=iwidth) break;
        *q++ = idx1[(c>>1)&1];
        if (++i>=iwidth) break;
        *q++ = idx1[c&1];
    }
    break;
case 2:
    for (i=0;i<iwidth;i++) {
        c= *p++^invert;
        *q++ = idx2[(c>>6)&3];
        if (++i>=iwidth) break;
        *q++ = idx2[(c>>4)&3];
        if (++i>=iwidth) break;
        *q++ = idx2[(c>>2)&3];
        if (++i>=iwidth) break;
        *q++ = idx2[c&3];
    }
    break;
case 4:
    for (i=0;i<iwidth;i++) {
        c= *p++^invert;
        *q++ = idx4[(c>>4)&15];
        if (++i>=iwidth) break;
        *q++ = idx4[c&15];
    }
        break;
    default:
        for (i=0;i<iwidth;i++) *q++ = *p++^invert;
    }
#ifndef NO_RLE
    i = rleit(pbuf, rbuf, iwidth);
    if (i>0) {
        (void)printf("\033^R;0;%d;1;%d;%s^", row, i, wname);
        p=rbuf; q=p+i;
#else
    if (iwidth>0) {
        (void)printf("\033^P;0;%d;1;%d;%s^", row, iwidth, wname);
        p=pbuf; q=p+iwidth;
#endif
        do {
            if (*p>31&&*p<123) putchar(*p++);
            else {
                putchar((*p>>6)+123);
                putchar((*p++&63)+32);
            }
        } while (p<q);
    }
    (void)fflush(stdout);
}

#ifndef NO_RLE
/*  rleit
*  by Tim Krauskopf
*  National Center for Supercomputing Applications
*  University of Illinois, Urbana-Champaign
*
*  This program is in the public domain.
*
*
*  Compress the data to go out with a simple run-length encoded
scheme.
*
*/

rleit(buf,bufto,len)
    int len;
    unsigned char *buf,*bufto;
    {
    register unsigned char *p,*q,*cfoll,*clead;
    unsigned char *begp;
    int i;
```

```
    p = buf;
    cfoll = bufto;              /* place to copy to */
    clead = cfoll + 1;

    begp = p;
    while (len > 0) {           /* encode stuff until gone */

        q = p + 1;
        i = len-1;
        while (i && *p == *q && i+120 > len) {
            q++;
            i--;
        }

        if (q > p + 2) {        /* three in a row */
            if (p > begp) {
                *cfoll = p - begp;
                cfoll = clead;
            }
            *cfoll++ = 128 | (q-p);    /* len of seq */
            *cfoll++ = *p;      /* char of seq */
            len -= q-p;         /* subtract len of seq */
            p = q;
            clead = cfoll+1;
            begp = p;
        }
        else {
            *clead++ = *p++;    /* copy one char */
            len--;
            if (p > begp + 120) {
                *cfoll = p - begp;
                cfoll = clead++;
                begp = p;
            }
        }

    }
/*
*  fill in last bytecount
*/
    if (p > begp)
        *cfoll = p - begp;
    else
        clead--;            /* don't need count position */
```

```
    return((int)(clead - bufto));       /* how many stored as encoded */
}
#endif


/*
 * The following adapted from tif_next.c 1.8 5/22/90
 * Copyright (c) 1988, 1990 by Sam Leffler.
 * All rights reserved.
 *
 * This file is provided for unrestricted use provided that this
 * legend is included on all tape media and as a part of the
 * software program in whole or part.  Users may copy, modify or
 * distribute this file at will.
 */

/*
 * NeXT 2-bit Gray Scale Compression Algorithm Support
 */

#define LITERALROW     0x00
#define LITERALSPAN    0x40

void
NeXTDecode(tf)
FILE *tf;
{
    register unsigned char *op;
    register int n;
    unsigned char buf[284];
    int scanline, row;

    scanline = ((long)iwidth*2+7L)/8L;

    for (row=0; row<ilen; row++) {
        switch (n = getc(tf)) {
        case LITERALROW:
            /*
             * The entire scanline is given as literal values.
             */
            (void)fread((char *)buf, 1, scanline, tf);
            break;
        case LITERALSPAN: {
            int off;
            /*
             * Each scanline is assumed to start off as all
```

```
         * white (we assume a PhotometricInterpretation
         * of ‘‘min-is-black’’).
         */
        op=buf; n=sizeof buf; do *op++ = 0xff; while (--n>0);
        /*
         * The scanline has a literal span
         * that begins at some offset.
         */
        off = getc(tf)*256;
        off += getc(tf);
        n = getc(tf)*256;
        n += getc(tf);
        (void)fread((char *)buf+off, 1, n, tf);
        break;
    }
    default: {
        register int npixels = 0, gray;

        /*
         * The scanline is composed of a sequence
         * of constant color ‘‘runs’’.  We shift
         * into ‘‘run mode’’ and interpret bytes
         * as codes of the form <color><npixels>
         * until we’ve filled the scanline.
         */
        op = buf;
        for (;;) {
            gray = (n>>6) & 0x3;
            n &= 0x3f;
            while (n-- > 0) switch (npixels++ & 3) {
            case 0:    op[0]  = gray << 6; break;
            case 1:    op[0] |= gray << 4; break;
                case 2:                op[0] |= gray << 2;
break;
                case 3:                *op++ |= gray; break;
                }

                if (npixels >= iwidth)
                    break;
                n = getc(tf);
            }
            break;
        }
        }
        output(buf, row);
    }
}
```

BANG OCTOBER 17 MEETING SUMMARY

Tom Masino Stanford University

BANG, the Bay Area NeXT User Group, held its monthly meeting at Stanford on October 17. Eric Ly stepped down as Executive Director of BANG as Rick Reynolds took over.

Marc Pawliger of IBM presented NeXTstep 1.0 on their RS/6000 workstations and discussed IBM's future plans for NeXTstep. Then, Andrew Stone of Stone Design demonstrated Create!, a "meta-drawing" program, and DataPhile, a database application.

Marc Pawliger from IBM's Palo Alto Advanced Workstations Division demonstrated NeXTstep 1.0 on the IBM RISC-based RS/6000 machine. The RS/6000 runs AIX, IBM's version of UNIX. NeXTstep 2.0 is currently in development, Marc said. Though the demonstration machine had 32 MB of RAM, AIX NeXTStep can run on the base memory level of 8 MB. The screen looks just like the one on a NeXT computer, complete with an icon dock, browser, tear-off menus, etc. The notable exception is the use of the IBM logo on the login box rather than the NeXT logo.

Users can choose between running NeXTstep or X-windows simply by setting either init file to activate upon startup. IBM's HFT (High Function Terminal) allows the user to "hot-key" between different virtual terminals, allowing NeXTStep and X to be run concurrently. Marc mentioned that code written on the NeXT computer needs to be recompiled to run on the RS/6000 with only small modifications to the Makefile. Interface Builder on the RS/6000 will take care of this modification automatically when saving Makefiles originally done on a NeXT computer.

Marc and his group ported NeXTstep largely by writing AIX kernel extensions to emulate the Mach calls that are not resident in the AIX kernel.

Marc briefly discussed InfoExplorer, IBM's functional equivalent of Digital Librarian, which includes graphics, linking, and remote database access capabilities. As of the meeting, no 3rd party applications have been ported.

With regards to speed differences, link time is reduced on the RS/6000 due to the machine's much higher speed than on a NeXT cube. Shared object libraries also considerably reduce executable size and launch times for the RS/6000. Marc reported a 2-3 fold speed increase for demo applications. The only caution Marc offered to programmers wishing to port their apps is that the RS/6000's screen is a bit larger than NeXT's MegaPixel Display, which may lead to ill-placed windows on the IBM screen. Differences between the RS/6000's hardware and the cube, such as a different video interface and lack of a DSP were also mentioned. Also, he cautioned that since the Mach kernel extension only implements the Mach calls needed for the Workspace and the AppKit, code that makes direct calls to the Mach layer will need to be replaced or changed. Other than that, the code which runs on the RS/6000 consists of the entire AppKit as shipped by NeXT, making most code easily portable.

The requisite IBM marketing person in the audience stated that the cost for NeXTstep on the RS/6000 is $500 and that there are currently no plans to market the completed NeXTstep PS/2 port.

Then, Andrew Stone, author of TextArt, took us through his two latest Stone Design's NeXT applications: Create! and DataPhile. Andrew called his Create! application a "meta-drawing" program, which, as those of you who know Andrew can imagine, means that you can create sophisticated and fanciful PostScript (in color if you have the $$) images simply by taking advantage of a large assortment of low and high level drawing tools. While other drawing programs are just beginning to explore this domain in which images result not only from paintbrush- or pencil-like drawing but can also from specifying higher-level graphic operations such as curves, squares, arcs, and so on, Create! takes it a step further by allowing easy control over such things as shadowing with a movable lightsource, gradient shading, animation, alpha-channeling, rotation, and more. Other features include unlimited zoom, an Expert PostScript hacking environment for editing and displaying raw postscript code and then determining bounding box which turns graphics into Encapsulated PostScript. An online library allows users to add and catalog their own images consisting of one or more Create! graphics.

Create! also takes advantage of the new Services remote procedure method so you can select text in any application, call upon Create to "illuminate" it along various paths: for example, straight, circle, lower or upper semicirle. This package must be seen to be appreciated but just to wet your whistle, try imagining a animated color neon bezier curve.

If you can do that you’re half way towards appreciating Create!. Andrew said that the application will be available summer of 1991.

DataPhile is a database program which at first resembles traditional Macintosh database applications such as Filemaker II. However, Stone Design has added a handful of useful features. In addition to text or calculations, fields can be sound or TIFF graphic images. Recursive sorts and finds allow for a subset of your records to be searched or queried. Default layouts are automatically generated for imported data files. Labels follow fields around in layouts. Mulitple views can be seen at the same time. Keyboard entry is available for all commands. Sort orders allow for sorting any number of fields in any order and for saving that index. Several tools assist in report generation. Sound annotation allows adding editable voice messages to any record. Exports can be made in Rich Text Format. DataPhile will ship in the spring of 1991 with a list price of $495.

For inquiries about BANG, please send mail or email to:

BANG P.O. Box 8858 Stanford, CA 94309 BANG-request@meta-x.stanford.edu

Integrated, First-Year Curriculum in Science, Engineering, and Mathematics

Jeffrey E. Froyd and Brian J. Winkel
Rose-Hulman Institute of Technology
Terre Haute, Indiana 47803

Report to the Sigma Xi Workshop on Entry-Level
Undergraduate Courses
Wingspread/Johnson Foundation
Racine, Wisconsin 53401
June 22, 1990

In November of 1986, a group of Rose-Hulman faculty conceived the idea of an integrated curriculum for first-year students that would be designed with two objectives. First, thematic concepts, concepts that span two or more scientific disciplines would be stressed instead of individual topics. Second, emphasis would be shifted from numeric and symbolic manipulation to problem formulation, problem-solving strategies, and solution interpretation. A preliminary syllabus has been developed for the entire first year, NeXT workstations and physics laboratory stations have been purchased, and activities are now being finalized for the sixty (60) students who will begin the new curriculum in August 1990.

Present Curriculum

In general, the present, first-year curriculum in science, engineering, and mathematics at Rose-Hulman Institute of Technology consists of the following courses.

Calculus I, II, and III (15 credits)
Mechanics or Engineering Statics (4 credits)
Electricity and Magnetism (4 credits)
General Chemistry I, II (8 credits)
Graphical Communication (2 credits)
Introduction to Design (2 credits)
Computer Programming I (2 credits)

Together, these courses represent 37 credits. In addition to these courses, students take courses in military science, literature and writing, and electives in humanities and social science.

Overemphasis on Manipulation

Conversations among the faculty revealed two widely perceived weaknesses in the current curriculum. First, there is too much emphasis on numeric and symbolic manipulation, especially the latter. Methods of integration can focus on finding closed-form anti-derivatives for varieties of integrals without developing intuition about the concept of integration. Physics and chemistry courses often allow success with the following problem-solving strategy: find the formula which contains the symbols which match the values given in the problem statement. Students write pages of algebraic manipulation without discerning the nature of the problem or developing an estimate of the solution that is required, or interpreting the result which is obtained. From their perspective, courses require that they memorize many collections of special techniques whose intellectual and scientific content remains obscure. Overemphasis on manipulative skills suppresses student curiosity and fails to develop required problem formulation and solution interpretation skills.

Emphasis on manipulative skills at the expense of concepts and problem solving strategies would be forgivable if technology to perform the manipulative tasks were not available. However, the technology is available, and curricula must address the issues of content, problem formulation, and solution interpretation.

Compartmentalization

The present curriculum presents students with discipline-oriented "containers of knowledge" called "courses." Each course focuses on topics, techniques, and applications which arise in the discipline. Integrating concepts is left entirely to students who are never given formal instruction on how to recognize and apply relationships across the boundaries of different disciplines. Students learn each new topic presented in each different course without developing a framework in which these topics may be integrated to create broad, thematic concepts which are more powerful and more generally applicable. Failure to recognize relationships and to integrate topics produces less efficient instruction and less effective problem solvers at a time when our curriculum is experiencing enormous pressure to add more material.

Now that the problems have been targeted, solutions are required.

Proposal for an Integrated, First-Year Curriculum

Structure

Design of the new curriculum followed four guiding principles

- Interdisciplinary
- Efficient

modern technology

designed coherent redundancy to reinforce and relate concepts common to a number of disciplines

- Adaptable - identify, codify, and introduce fundamentals as science and technology advance
- Visibly relevant and interesting

The resulting structure is a three course sequence (quarter system). Each course is twelve credits.

Concepts

One of the greatest challenges for science, engineering, and mathematics education posed by exponential knowledge growth and rapidly advancing technology is identify a small number of broad, powerful concepts which must be communicated to students. At the risk of adopting an overused word, these concepts are the fundamentals. Today, there are no fundamentals. Or rather, each person has his or her own set of fundamentals and the intersection of the sets of seven of more people is nearly empty. Educators must identify fundamentals and select topics to reinforce these fundamentals instead of arbitrarily deciding which topics will be taught and which topics will not be presented.

To counter compartmentalization, a small number of concepts were identified as the focal points in the curriculum. These concepts were organized into three categories.

1. Basic Building Blocks
2. Thematic Concepts
3. Problem-Solving Strategies

Basic Building Blocks

Basic building blocks are the concepts upon which science, mathematics, and engineering are based. These must be communicated early and reinforced throughout the curriculum. Four basic building blocks have been identified

1. Function
2. Vector
3. Three-Dimensional Visualization
4. Physical Abstraction.

The first three are self-explanatory. The fourth requires some explanation. Physical abstractions are quantities which scientists invent to describe and explain observed behavior. They include length, mass, temperature, energy, entropy, etc. Students need to realize that physical abstractions are not concrete; instead, they have been made up and are used simply because of past success in describing behavior in the physical world around us. Also, students must learn and use the units associated with the physical abstractions.

Thematic Concepts

Thematic concepts are links which span two or more different disciplines. Three have been identified.

1. Rate - The rate at which a quantity is changing, both average and instantaneous, is important. The concept of rate appears in reaction kinetics, velocity, acceleration, the derivative, and in Newton’s Second Law where force is set equal to the rate at which linear momentum is changing.

2. Accumulation - Accumulation is the notion that the value of a quantity can be calculated by summing individual contributions. Further, the accuracy of the value can be increased by summing a greater number of smaller pieces. Areas under curves are calculated by summing areas of rectangles or trapezoids. Total mass, center of mass, and moment of inertia are calculated by summing contributions of infinitesimal pieces of mass. Work is calculated by summing individual pieces of work, and, in the limit, work is calculated by using a line integral. In the limit, rate and accumulation are related by the fundamental theorem of calculus.

3. Conservation - Observations of physical phenomena indicate that there are physical abstractions whose total amount in the universe remains constant with respect to time. Such physical abstractions are said to be conserved. Often, realization that there is a quantity which is conserved generates a new physical abstraction. If an abstraction is conserved, and if in your system the quantity of the abstraction is either increasing or deceasing, then the quantity of the abstraction in the environment must be decreasing or increasing. Conservation laws are invaluable in formulating problem statements which can be solved. Quantities which are conserved are

Amount of elements (in the absence of nuclear reactions) - In a chemical reaction, the amount of hydrogen is constant.

Linear momentum -Two systems exchange linear momentum through an abstraction called force.

Angular momentum - Two systems exchange angular momentum through an abstraction called torque.

Charge

Mass

Energy

(**Entropy** - Even though the total amount of entropy in the universe is not constant, it is non-decreasing. Therefore, it is worthwhile to list this physical abstraction.)

Problem-Solving Strategies

Problem-solving strategies are what we use when we don’t see how to solve the problem. Strategies can be as simple as draw a picture of your problem or consider the units in the problem. They may be more complex: first, identify the goal; second, decide how you plan to reach the goal; and third, implement your plan. First-year mathematics, science, and engineering students have very few problem-solving strategies because they have worked through very few problems in which the solution was not immediately apparent to them. They are ill-equipped to attack multi-step word problems in physics, chemistry, and calculus, not because of their failure to grasp the concepts involved, but because they don’t immediately see how to solve the problem and therefore, they become convinced that they don’t understand the material.

In the new curriculum, teachers will present and model problem-solving strategies for the students. Students will be required to elucidate their problem-solving strategy when they submit their problem solutions.

Implementation

The new curriculum will be offered for the first-time to sixty (60) students during the 1990-91 academic year. Participants will be selected from students who have volunteered to become a part of the new curriculum. To date, over two hundred students from an incoming first-year class of approximately 360 students have indicated they want to participate in the integrated curriculum. Participants will be selected by 15 July 1990.

To shift emphasis from numeric and symbolic manipulations to problem formulation and solution interpretation, Rose-Hulman has purchased seventy (70) NeXT workstations and equipped them with, WingZ (a spreadsheet), and FrameMaker (a document preparation system). Also, each NeXT comes bundled with Mathematica, Interface Builder (a graphical tool for user interface and software development), an Objective-C programming environment (compiler, editor, and debugger), Digital Librarian, Webster’s Dictionary, and WriteNow (an easy-to-use word processor). Also, the Institute has purchased sixteen (16) physics laboratory stations with an air table, sonic ranger, rotational table, and a Zenith 286-LP computer to support data acquisition and analysis.

In the 1991-92 academic year, the integrated curriculum will be taught with 120 students. On the basis of the two years’ experience, the Institute will decide to expand the curriculum to the entire first-year class, offer the curriculum to a portion of the first-year class, or discontinue the integrated curriculum.

Acknowledgments

Six faculty prepared the preliminary syllabus for the integrated, first-year curriculum in the summer of 1988. Their work was supported by The Lilly Endowment, Inc., Grant Number 870643.

Development of the integrated curriculum and the first two years of testing are being supported by the Undergraduate Curriculum Development in Engineering Program of the National Science Foundation, Grant Number 893553.

NeXT Applications

Area - In one window Area shows the graph of a function. In a second window, Area shows the integral of the function. As the user moves a slider, Area highlights the area under the first curve, shows the value on the second graph, shows the x-coordinate for the function, the y-coordinate for the function, and the area under the function. (Author - Coey Minear)

AreaGame - In one window AreaGame shows the graph of a user-selected, but unknown, function. The user can select from one of nine functions. As the user moves a slider, the area under the curve is highlighted. Then, the user must point to the value of the integral of the function at the point which has been selected. After selecting an arbitrary number of points, the user indicates DONE. Then, AreaGame draws the integral of the function and provides a score to the user indicating the accuracy of the values which the user has entered. (Author - Coey Minear)

Calc - Calc is a scientific RPN calculator. It provides trigonometric, logarithmic, exponential, gamma, and erf functions. A more comprehensive calculator is currently being produced. (Author - Jeff Froyd)

Crystal - Crystal is a port of Crystal produced by the Northstar program at Dartmouth College. Users can select a crystal structure, add or delete atoms, or slice the lattice along a specified plane. (Authors - Northstar programming team, Troy Thomas, Jeff Adams)

Curves - Curves is an application to help students visualize the sine, cosine, exponential, and logarithmic functions. For example, in one window a graph with four sliders and the text A sin(Bt + C) +D is shown. With the sliders or their associated text fields, users adjust the values of A, B, C, and D. Curves provides immediate feedback on the effect of changing, for example, the frequency. Then, the graph of the sine function with the selected parameters values is produced. Similar windows are provided for the cosine, exponential, and logarithmic functions. (Authors - Jerry Fine, David Fischer, David Smith)

DeFun - DeFun shows three graphs one above the other. The top graph is the function, the middle graph is the first derivative, and the bottom graph is the second derivative. DeFun reinforces graphical interpretations of the first and second derivative. DeFun messages Mathematica to produce the graphs. (Author - David Fischer)

DialMForMagnet - DialMForMagnet shows a grid for four hundred points. In the grid, the user may place combinations (taken two at a time) of lines of current flowing perpendicular to the grid or loops of current perpendicular to the grid. As the user moves either the lines of current or the loops of current, the program recalculates the magnetic field at each grid point and displays the magnetic field by showing vectors at each grid point. (Author - Matthew Drew)

ElectricCompany - ElectricCompany shows a grid of one hundred points. In the grid, the user may place combinations (taken two at a time) of points charges, rings of charge, or lines of charge. As the user moves the locations of the charges, the program recalculates the electrostatic field at each grid point and displays the electrostatic field by showing vectors at each grid point. (Author - Matthew Drew)

FieldSimulator - FieldSimulator allows students to explore gravitational and electrostatic fields. For gravitational fields, students can enter point masses, rings of mass, hollow spheres of mass, and solid spheres of mass. Then, the program can display the gravitational field or the equal gravitational potential contours. It can also display the value of the gravitational field or gravitational potential at user selected points. For electrostatic fields, students can enter point charges, charge dipoles, rings of charge, hollow spheres of charge, and solid spheres of charge. Then, the program can display the electrostatic field or equal electrostatic potential contours. It can also display the value of the electrostatic field or electrostatic potential at user selected points. Finally, the user can enter a test charge/mass and follow the trajectory of the test charge/mass. (Author - Jeff Adams)

FunctionAndSons - FunctionAndSons shows the graphs one above the other. In random order, the graphs are the function, the first derivative, and the second derivative. The user must correctly identify which is which. FunctionAndSons is a game to reinforce graphical interpretations of the first and second derivative. FunctionAndSons messages Mathematica to produce the graphs. (Author - David Fischer)

GasLaw - GasLaw allows the user to select one of the four variables in the ideal gas law: $PV = nRT$, as the independent variable. It then displays the other three as functions of the independent variable. GasLaw allows users to explore the functional relationships implicit in the ideal gas law. (Author - David Smith)

Nomenclature - Nomenclature is a drill and practice program for (1) element symbols, element names, and ionization states for the element names, (2) compound names and symbols, and (3) ion names and symbols. (Author - David Fischer)

Parametric - Parametric allows the user to specify a wide range of parametric equations. It then animates the trajectory of the particle, the velocity vector, the acceleration vector, and the circle of curvature. (Author - Matthew Drew)

PhysicsWorld - PhysicsWorld allows users to explore the motion of particles with both charge and mass through gravitational, electrostatic, and magnetic fields. Users can specify combinations of fields and up to twenty particles. (Author - Jeff Adams)

RateLaw - RateLaw animates the kinetics of chemical reactions as the reaction moves toward equilibrium. Users can specify amount of reactants, products, and rates of reaction, both forward and reverse. (Author - David Fischer)

Riemann - Riemann allows users to explore the nature and relative accuracy of various numerical integration methods. Methods examined are the left-hand rule, the right-hand rule, the midpoint rule, the trapezoidal method, and Simpson’s rule. Riemann displays the graph of the function, and the areas used by each numerical method to approximate the area under the curve. Numerical values for each estimate are obtained and can be compared graphically. (Author - Matthew Drew)

Secant - Secant allows users to explore the secant as an approximation to the derivative. Users can select the interval over which the secant is calculated. The graph of the slope of the secant versus the x-coordinate of the left-hand endpoint is displayed below the graph of the function. (Author - Matthew Drew)

SigFigQuiz - SigFigQuiz is a drill-and-practice program. The first part helps users to identify the significant figures in a number. The second part drills users on addition, subtraction, multiplication, and division of numbers with significant figures. (Author - David Smith)

Skiing - Skiing shows three graphs of the skier's motion. The skier's x-position versus time, x-velocity versus time, and x-acceleration versus time. Given the three graphs, the user must construct the ski slope on which the skier was skiing to produce the displayed motion. Bunny and expert slopes are available. (Author - Matthew Drew)

Slope - In one window Slope shows the graph of a function. In a second window, Slope shows the derivative of the function. As the user moves a slider, Slope shows the line tangent to the first curve, shows the value on the second graph, shows the x-coordinate for the function, the y-coordinate for the function, and the slope of the function. (Author - Coey Minear)

SlopeGame - In one window SlopeGame shows the graph of a user-selected, but unknown, function. The user can select from one of nine functions. As the user moves a slider, the slope of the curve is shown. Then, the user must point to the value of the derivative of the function at the point which has been selected. After selecting an arbitrary number of points, the user indicates DONE. Then, SlopeGame draws the derivative of the function and provides a score to the user indicating the accuracy of the values which the user has entered. (Author - Coey Minear)

TaylorSeries - TaylorSeries uses Mathematica to compute the Taylor polynomial of user-selected order about a user-selected point. It can then graph (by messaging Mathematica) both the original function and the Taylor polynomial. Then, the user can perform two types of error analysis. First, the user can select a point and evaluate the difference between the original function and the Taylor polynomial. Second, the user can specify the magnitude of the difference between the original function and the Taylor polynomial. Then, the program will compute the distances (left-hand and right-hand) from the point about which the Taylor polynomial was generated for which the difference between the original function and the Taylor polynomial will be within the specified tolerance. Finally, TaylorSeries can display graphs of the original function and multiple Taylor polynomial approximations. (Authors - Michael Allard, Anthony Zamora)

TrigGig I - TrigGig I displays the unit circle and graphs of the sine, cosine, and tangent functions. As the user selects an angle on the unit circle, the corresponding values of the sine, cosine, and tangent functions are indicated on their respective graphs. (Author - David Fischer)

TrigGig II - In TrigGig II, a graph of a random sinusoidal function is displayed. Then, users are prompted for values of the mean, amplitude, radian frequency, and phase angle (in radians). After the user has supplied the values, the sinusoidal function with their selected parameters is displayed along with the original function, and a score is generated by comparing the correct values with the user-entered values. (Author - David Fischer)