Newsletter of the NeXT Programmer's Community

NeXT Users' journal

JULY 1990 • ISSUE 8

WELCOME

Erica J. Liebman

Apologies first. A couple of things got bounced from this issue to the next one because of communications problems and/or space. First is Dave Hieb's excellent Mathematica tutorial which will run next month. The Compus Government Watch column also was shifted a month. We don't have a Stone Age this month, but Dan Gessel's 3D Kit article outlines a set of objects for displaying wireframe 3D graphics.

Terrence Talbot should have a review of Diagram for next month if he can get it from Lighthouse and I need to be in touch with Media Logic to see if I can lend him a copy of TopDraw for review without infringing licensing.

I've spent a good part of the month trying to track down "paper" options and getting scared out of my skull by the tax-laws, incorporation laws and so forth involved. So, like any good NeXTie, I immediately began investigating alternative, *electronic,* forms of distribution.

First off, I have been in contact with Portal Communications and CompuServe. I mean to hit MCI Mail , GEnie and the Source over the next few weeks as well. A few of our members already are getting their issues through these channels. This casts expanded service in a positive light.

I've been trying to examine careful (in response to continued complaints/feedback/etc) how best to present the issue for both on-line and off-line readers. I've experimented in this issue with perhaps four or five different column styles. I need you to feed back to me which were most effective.

When talking recently (and continuously) with David Joerg, who by the way has undergone some painful surgery and can use some cheering up, we hit on the topic of including NIB files in the Journal. It was, and is, my feeling that if we could do so in "Icon" format (like in Mail), it would be most effective. What other media would you like to see in Journal distributions?

Now, the funny story. I was talking with Frame recently. You know all the problems we've been going through in terms of incompatibilities with printers/machines and so forth... So I started talking to these Frame types and telling them about my woes. They interrupted me. "You know," they said, "those are technical problems."

"Yes," I agreed. "Who in Frame can I contact to deal with these problems? They aren't reflecting well on you for all those people who are receiving the Journal and might consider buying a copy."

"You'll have to contact your local Businessland rep. They handle all our technical questions."

As always folks, we are looking for articles : how-to and review articles are especially in demand. Say, did Bruce Henderson's latest "The Way I See It" disturb you? Want to rebut? Write a two to four page essay (that is, up to two Frame pages worth) and get your viewpoint heard from.

Have you come across any neat hacks or nifty ideas? Send 'em in to Buzz's Hint Corner.

Are you aware of NeXT Products not listed in Market View? Send as much information in as you can and we'll get a listing going for that product.

Are you a government contractor? Give me a call and let's set up a Government Watch column for an upcoming month.

Take care, all. I'll see you next month after another few hard-working sleepless nights of putting this baby together -- Be Real or Be Integer -- Erica

**Copyright 1990 by BuzzNUG
Individual Articles Copyright 1990
by their Authors. This issue of
NUJ may be freely copied and
distributed but not altered. Authors
are free to resubmit articles for
publication in other media.
This issue of NUJ may not be
sold for profit.**

INDEX

Welcome • 1
Headline News • 2
Editorial Stuff • 2
Feedback from the Trenches • 2
3D Kit • 4
C Threads • 14
Tao • 24
HardCore SIG - Parallelism• 19
Mumps Language for the NeXT • 33
Great Scott/Taming of the Sh (and Csh) • 34
Reviews, Rumors & Stuff (The Myopic Eye) •46
Market View • 48
User Groups • 51
Buzz's Hint Corner • 55
NeXT Support Answers • 56
Next on Campus • 57
The Way I See It • 58

Contributors:

Erica J. Liebman,Ian Smith, Doug Kieslar, Dan Gessel, Bruce Henderson, Scott Hess, David Joerg, Conrad Geiger, Robert Lin, Arthur Lee

•Editorial Matters•
1150 Collier Road NW/L-12
Atlanta, GA
30318

HEADLINE NEWS

• NeXT Announces that 68030 Machines will run OS version 2.0 and 68040 Machines will require 2.0.

• NeXT confirms that 68030 Simms can be used on the 68040 board.

• Ashton Tate and WordPerfect are looking for BetaTest sites for new products.

• Lighthouse is demonstrating its new **Diagram** product.

• NeXT on Campus is looking for writeups of Academic programs for their Fall issue. (They are still giving away free subscriptions!).

•Editorial Stuff

Articles for Buzzings are accepted in various forms, NeXT mail enclosures and Internet .wn.tar.Z forms are preferred but ascii text via the net, IBM and Mac Disks via USMail and (yikes) written text via the same USMail are happily accepted. We can guarantee no return of materials without SASEs, sorry. Our focus is how-to articles, especially with sample code. All articles are subject to editorial review.

We also welcome copies of new (and old) software for review from third party vendors. Again, we can not guarantee material return without SASE or guarantee publication dates, if at all, although we try to be prompt.

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

Mailed subscriptions should start real soon now, with any luck. Please write for information.

Feature Submission Guidelines

If you have NeXT mail, simply drag your icon into next mail and post it off to me. Don't worry about tar'ing and compressing.

•Avoid passive voice. Please.

•Spellcheck.

•Wit is welcome, obscenity is not.

•Preformatting with the following guides will aid us ENORMOUSLY:

•Paragraphs: No tabs, single return at end of paragraph, no hard-returns at ends of lines

•Title on one line, author information on one line.

•Avoid blank lines between paragraphs. Use blank lines to denote sections.

•Start sections on first line after section header.

Editorial matters to:

BuzzNUG c/o EJ Liebman
1150 Collier Rd./NW L-12
Atlanta, GA 30318.
1-404-352-5551.

There is always an answering machine, but please respect relatively normal hours. Long distance phone calls may not be returned by the impoverished student at the other end -- if you leave a message, indicate if it is acceptable to call you collect. Please send any deliveries of items that will not fit within a tiny mailbox care of the Leasing Office. To contact me directly for subscription information, corrections, requests, or just to say hi, write via internet: erica@kong.gatech.edu

This issue of NeXT Users' Journal was prepared in FrameMaker, generously donated by Frame.

Feedback from the Trenches

••Very nice. Style is crystallizing. The little bits of clip art help (emphasis) A LOT. A Graphical User Interface for the Journal. NUJ continues to carry a high concentration of useful information and interesting news, even in a time period that, in retrospect, will seem very dull and boring (compared to September-October-November). Bruce Henderson's "two-page fiasco in print" nearly killed me. I mean it.Understand, I'm sitting here covered in stitches and tape from expensive and painful Surgery, laughing my guts bloody... the best use of italics and boldface since _The_Book_of_The_SubGenius_... and his points werewell-made too.Obscure NeXT Trivia: In which NeXT manual is J. R. "Bob" Dobbs mentioned? (Digital Librarian won't help you, it's not on-line) - *David Joerg*

••I just read your lastest version of Buzzings on the NeXT preview. I am very impressed with the contents. I have difficulty printing out, could you convert it to ascii for me. Preferably I would like to receive a copy of it by mail. Any suggestions ?Thanks for the help -- *Chris Schmechel, Duke University Academic Computing Center,Student Consultant, chris@dukemvs,-chris@dukemvs.ac.duke.edu*

Is it possible to buy ad space in The NeXT User's Journal? We have a product we are announcing and your magazine is about the only NeXT magazine right now. Thanks, --*Paul Morin*

So long as we are distributing on Internet, it is inappropriate to take paid advertising. We are happy to include product listings and reviews in the Market-View and Review sections of the newsletter -- EJL.

••Hi. Just got Buzz7 from cs.orst.edu. Haven't had much chance for reading it yet, but layout- and contentwise (don't you just hate people who use xwise) it looks to be up to your usual good stuff.

HOWEVER, (now comes the zinger) - there were the usual mechanicalproblems. In the Frame version,

page 20 will not print and will not even display - just get a blank page. On the bright side, it neitherkilled Frame nor the cube. The ps version put up a window about 1x3,tried to write all 44 pages on top of each other then died of exhaustion with a cryptic message from PostScript that I didn't try to capture.Just occurred to me that I could have tried to use Edit, but then Imainly wanted to print page 20 in some formatted form.

I hope you don't construe this as negativism on my part. I amimpressed, nay overwhelmed, by the amount of work you must put into this and manage classes as well. But if I don't report problems, you can'tfix them or else tell me what I'm doing wrong. -- *Betty Johnson*

••In your article on Minimum Spanning Trees, I noted a couple of misconceptions (not that I want to preach at you, just to let youknow :-]).

First, List is not List in the sense of linked-lists. List instances actually work with an array malloced somewhere, much like Storage instances do. As such, some methods are not as efficient as in a linked list, such as inserting an element at the front of a longlist. In fact, List could be implemented as a subclass of Storage (and I wish I knew why NeXT didn't do that!)

Second, Storage allocates storage for you. You used:

```
mLoc=(NXPoint *)malloc( sizeof( NXPoint));
*mLoc=ptr->location;
...
[dataPoints addElement:mLoc];
```

unless I greatly misunderstand what you are doing, [dataPoints addElement:&(ptr->location)] should do the trick. This is because Storage copies the data from the pointed-to area intoits internal array. Then, you need not worry about malloc() andfree() (that's why Storage is useful!)

Also, your instantiation of the dataPoints object will not work well for archiving! You used a human-readable form for the description. What you really need is the form documented in the documentation for the Storage class (look on the online stuff if you don't have 1.0 Reference manual). There, you will see that whatyou really want is:

```
dataPoints=[Storage newCount:0 elementSize:sizeof( NXPoint) description:"{dd}"];
```

for a Storage instance holding structures which contain two doubles. Lastly (finally!), to fix some of the problems you mentioned with working with pointers, you should probably subclass Storage to getwhat you want. To save you trouble, here's a version I would use (note that I'm essentially typing this in off the top of my head, so don't expect it to compile first time!):

```
Source: NXPoints.h
#import <dpsclient/event.h>     // For the NXPoint declaration.
#import <objc/Storage.h>

@interface NXPoints : Storage
{
}
+ newCount:(int)count;
- addNXPoint:(NXPoint *)p;
-(NXPoint *)NXPointAt:(unsigned)i;
- replaceNXPoint:(NXPoint *)p at:(unsigned)i;

@end

Source: NXPoints.m
#import "NXPoints.h"

@implementation NXPoints : Storage
{
}
+ newCount:(int)count
{
  return [super newCount:count
              elementSize:sizeof( NXPoint)
              description:"{dd}"];
}
- addNXPoint:(NXPoint *)p
{
  return [super addElement:p];
}
-(NXPoint *)NXPointAt:(unsigned)i
{
  return (NXPoint *)[super elementAt:i];
}
- replaceNXPoint:(NXPoint *)p at:(unsigned)i
{
  return [super replace:p at:i];
}

@end
```

Now, to instantiate:

```
 dataPoints=[NXPoints newCount:0];
```

to get a point:

```
NXPoint p=*[dataPoints NXPointAt:10];
```

and to add a point:

```
[dataPoints addNXPoint:&p];
```

I would have allowed passing of NXPoint structures directly (so that you could do p=[dataPoints NXPointAt:10];), but I have had bad luck with that, though it should work fine.

Anyway, hope this helps you understand the intricacies (sp?) of the Storage class a little better!-- *Scott Hess,* scott@gacvax1.bitnet

When I did not *originally use malloc's I found that my program would crash, by adding the memory allocation, and copying from the event I had no problem. I suspec that it is the way that the event structure is handled that is the problem, but hey, I'm still learning this stuff. Thank you, Scott, as per our continued e-mail conversation. -- EJL.*

I found a copy of your Projection Lab and just love it. I think there is a fee for it, still have to pay. Have you seen our second issue of NeXTVieW and the two issues of Tao (our tabloid) it should be on the Purdue archive and is for sure on the UBC archive (ftp cs.ubc.ca). Love your issue 6. Can I do more without the wait for programs? Do you want to print my Bezier Curve paper? Regards- *Tom Poiker*

If you register for either Projection Lab or Bernie the Duck, I will automatically register you for the other program as well (what a special!). I'll be looking for your Bezier Curve paper! -- EJL.

3DKIT

A set of objects for displaying wireframe 3D graphics.

Daniel Mark Gessel
5 Roylencroft Ln.
Rose Valley, PA 19065
gessel@cs.swarthmore.edu untill Sept. 1990

Introduction

I have set out to create an example of how the AppKit's hierarchical View structure can be adapted for displaying 3D graphics. You will find the code for the classes I have created below. It will be submitted, along with a copy of this article, and an example program created through Interface Builder, to various ftp sites.

Please feel free to contact me at the above addresses about suggestions for additions, modifications, successes and failures you have had with this kit, as well as any bugs that may have crept in. I will submit this code along with an example created in Interface Builder to various archive sites. If interest warrants, I will keep the kit up to date with suggestions and modifications, to the best of my ability.

Core Structure

The file `3D.h` declares the type `vector3D`. It is used throughout the kit as a primitive type. There are three core objects for this kit.

Context3D

This is a subclass of View. It holds unique information that is vital for the presentation of 3D graphics on screen. In this simple case, it holds the distance from the viewer's eye to the picture plane, which the screen represents, in the instance variable *`pictureDistance`*. The conversion of 3D coordinates to 2D coordinates is dependent on this distance. It also holds distance to a z clipping plane. Any drawing that would be closer to the eye than *clippingDistance* is not drawn.

The negative z axis is into the screen, the x axis is toward the right, and the y axis is upward. This makes the coordinate system right handed. The clipping and picture planes are represented as distances, however. This means that the picture plane has the implicit formula 0=-*z*-*`pictureDistance`*. Simalarly for the clipping plane.

View3D

This is a subclass of object. It is not a responder, since this 3D Kit assumes that the 3D objects drawn by this kit are purely graphical. This is not necessarilly how we would like it to be, but, because this kit is slow, and only implements drawing routines for wireframe images, making a Responder3D class seemes unnecessary. All drawing is done by View3D subclasses, using `moveto:`, `lineto:` and `polygon:howMany:` messages sent to `self` or directly to `superview`, where `superview` has a function identical to the superview in the View class. Sending these messages directly to the `superview` will avoid the View3D's own transformation object. This may be useful at times.

Transformation3D

This is also a subclass of object. It implements the methods `operateOn:` and `operateOn:howMany:` such that they do nothing. The methods are to be overwritten (as in the subclass Linear3D) to perform transformations on vector3Ds, pointers to which are passed as arguments to these methods. Linear3D implements many useful transformations, including rotations, translations and combinations of these. There is a `transformation` instance variable in the View3D class. This can be assigned to an instance of a subclass of Transformation3D, which will automatically be sent `operateOn:` messages with the vector3Ds which are passed to the drawing messages mentioned above.

Comments

There is the Linear3D (mentioned above), a Transformation3D subclass, and Cube, a View3D which draws a simple unit cube centered at the origin. The code is commented, with instructions on how to use the classes in the interface files.

The drawing as implemented is slow. There is no attempt at optimization, just to create a simple example of object oriented 3D drawing in a style similar to the AppKit.

The drawing routines are simple, and use an easy perspective projection. To get a zoom View, put the pictureDistance at a large negative z value. Put whatever objects you wish to view at a similar distance. The resulting image will have little perspective. To increase the perspective effects, bring the pictureDistance close to zero.

The Cube class provided is very simple. Since the polygons are declared locally within the render method, the fact that Transformation3D's operateOn: and operateOn:howMany: are destructive (that is, manipulate the data passed at the memory location passed) is meaningless. If you create an object on the fly, or declare variables which hold the definition of the object, you must pass copies to moveto:, lineto: and polygon:howMany:.

Play around with the Linear3D transformation and the Cube, using translation in the negative z direction to get the rendering to show on screen (rather than behind it). Large negative z values will put the objects far off into the distance, so they will be small. If you haven't had linear algebra, play around with concatinating various Linear3D transformations in various orders. To rotate a cube, say, rotate first, then push it back on the z axis. This will rotate the cube around it's origin. If you do the operations in the reverse order, the cube will rotate around the viewer such that a 180 degree rotation would put the cube behind the screen.

Try creating a few simple 3D objects as well, like tetrahedrons, etc. Static objects are easiest, since you can

just modify the Cube directly. Try some heirarchical objects, such as a finger, using subviews of subviews. Modify the code, to make it better or faster. Make it collect up polygons in a list and then use a hidden line algorithm for the actual rendering.

Again this is a simple example, feel free to improve on it, modify it, or rewrite it your own way.

```
/* 3D.h */

/*  This file (3D.h) declares the vector3D type. This is not an
    object so that operations on it and storage etc. can be as flexible
    and direct as possible. */

typedef struct vector3D
    float x,y,z;
 vector3D;

/* Context3D.h */

#import <appkit/View.h>
#import "3D.h"

@interface Context3D:View

    id contentView;
    vector3D currentPoint;
    float pictureDistance,clippingDistance;


    /*
        contentView
            is the top of the View3D drawing hierarchy for this context.
        currentPoint
            is the current drawing point, used by lineto: messages and altered by
            moveto:, lineto: and polygon:howMany: messages
        pictureDistance
            is the distance from the viewers eye to the virtual screen which
            the Context3D represents
        clippingDistance
            is the distance at which all closer drawing will not be drawn (clipped
            out)
    */

+ newFrame:(NXRect *)frameRect;

    /*
        +newFrame: calls [View +newFrame:frameRect] to create the new View
        object, andthen initiates it's instance variables. The following are
        initial Values:
            contentView to a new View3D, which should be freed if changed.
            currentPoint to 0.0,0.0,0.0
            pictureDistance to 1500.0
            clippingDistance to 1.0
    */

- moveto:(vector3D *)where;

    /*  -moveto: set's the currentPoint equal to *where. */

- lineto:(vector3D *)where;

    /*
        -lineto: draws a line on screen from the currentPoint to where after
        projecting the current point and *where to the Context3D's coordinates
        as a View. It leaves currentPoint == *where
    */

- polygon:(vector3D *)vertices howMany:(int)count;

    /*
        vertices must point to an array of count elements type vector3D.
        -polygon: howMany: does the equivalent of a moveto: to the first
        vector3D of vertices, and then lineto: to each succesive vertex
        back to the first one.
    */

- contentView;

    /* -contentView returns the id of the current contentView. */

- setContentView:anObject;

    /*
        -setContentView: sets the id of the current contentView, and will
        send [anObject setSuperView:self].
    */

- setPictureDistance:(float)dist;

    /*
        sets the pictureDistance instance Variable to dist. dist must be
        a postitive number. If succesful, setProjection returns self, otherwise
        it returns nil.
    */

- setClippingDistance:(float)dist;
```

```
    /*
        sets the clippingDistance instance Variable to dist. dist must be
        a postitive number. If succesful, setClipping returns self, otherwise
        it returns nil.
    */

- drawSelf:(const NXRect *)rects:(int)rectCount;

    /* sends render to its contentView */
- free;
@end

/* Context3D.m */

#import "Context3D.h"
#import "View3D.h"
#import <dpsclient/wraps.h>

@implementation Context3D
+ newFrame:(NXRect *)frameRect

    self=[super newFrame:frameRect];
    pictureDistance=1500.0;
    clippingDistance=1.0;
    currentPoint.x=0.0;
    currentPoint.y=0.0;
    currentPoint.z=0.0;
    [self setContentView:[View3D new]];
    return self;


- moveto:(vector3D *)where

    currentPoint=*where;
    return self;


- lineto:(const vector3D *)where

    NXPoint start,end;
    vector3D current,to;
    if(currentPoint.z > -clippingDistance) // we make to be the point
        if(where->z > -clippingDistance) // less than clippingDistance
            currentPoint=*where;      // if there is one.
            return self;                     // if both to and currentPoint
                                                 // are closer than
        to=currentPoint;                 // clippingDistance, we return
        current=*where;                  // out.

    else
        to=*where;
        current=currentPoint;

    if(to.z > -clippingDistance) // if to is closer than
        float temp;                          // clippingDistance, we
        vector3D tempVect;             // know from the above
        temp=current.z+clippingDistance;// steps that current
        temp=temp/(current.z-to.z);// is not.
        tempVect.x=to.x-current.x;// We recalculate to such that
        tempVect.y=to.y-current.y;// it is at the intersection
        tempVect.z=to.z-current.z;// of the line segment and the
        to.x=current.x + tempVect.x*temp;// clipping plane.
        to.y=current.y + tempVect.y*temp;
        to.z=current.z + tempVect.z*temp;


    // here we do a simple projection, since we now know that both current and
    // to are in front of the clipping plane.

    start.x=current.x*(-pictureDistance)/current.z;
    start.y=current.y*(-pictureDistance)/current.z;
    end.x=to.x*(-pictureDistance)/to.z;
    end.y=to.y*(-pictureDistance)/to.z;
    PSmoveto(start.x,start.y);
    PSlineto(end.x,end.y);
    PSstroke();
    currentPoint=*where;
    return self;


- polygon:(vector3D *)vertices howMany:(int)count

    int ctr;
    [self moveto:&vertices[0]];
    for(ctr=1;ctr<count;++ctr)
        [self lineto:&vertices[ctr]];
    [self lineto:&vertices[0]];
    return self;


- contentView

    return contentView;


- setContentView:anObject
```

```
    contentView=anObject;
    [anObject setSuperView:self];
    return self;


- setPictureDistance:(float)dist

    if(dist > 0.0)
        pictureDistance=dist;
        return self;

    return nil;


- setClippingDistance:(float)dist

    if(dist > 0.0)
        clippingDistance=dist;
        return self;

    return nil;


- drawSelf:(const NXRect *) rects:(int)rectCount

    [contentView display];
    return self;

- free
    [contentView free];
    [return [super free]];

@end

/* View3D.h */

#import <objc/Object.h>
#import "3D.h"

@interface View3D:Object

    id transformation;
    id subviews;
    id superview;


    /*
        tranformation
            is a Transformation3D object, used on all drawing that is done by
            this view3D or any of it's subviews.
        subviews
            is a List object which contains all the View3D's subviews
        superview
            should be a View3D or a Context3D. All View3D's must have a superview
            to function properly
    */


+ new;

    /*
        sets up a new View3D object with the instance variables initialized
        to the following:
            transformation is nil
            subviews is an empty list
            superview is nil
    */


- moveto:(vector3D *)where;

    /*
        moveto: sends [transformation operateOn:where] and then
        [superview moveto:where]
    */

- lineto:(vector3D *)where;

    /*
        lineto: sends [transformation operateOn:where] and then
        [superview lineto:where]
    */

- polygon:(vector3D *)vertices howMany:(int)count;

    /*
        polygon:howMany: sends [transformation OperateOn:vertices
        howMany:count]
        and then [superview polygon:vertices:count]
    */

- superview;

    /* Returns the View3D's superview */

- setSuperView:aView3D;
```

```
    /*
        Should only be sent by another View3D or a Context3D to let the receiver
        know
        that its superview has been changed. Any calls to addSubView should
        automatically send this message to the subView being added. aView3D
        should be either a View3D or a Context3D.
        (see addSubView: below)
    */

- addSubView:aView3D;

    /*
        This is the way to add View3Ds into a View3D heirarchy. It Will
        send the setSuperView: message to aView3D with self as an argument,
        and will add aView3D into subviews automatically.
        (see setSuperView: above)
    */

- render;

    /*
        This replaces the DrawSelf method of the View class. It should send
        itself moveto:, lineto: and polygon:howMany: messages. The vector3Ds
        sent along as arguments must be disposable, or assigned their original
        values each time the render message is sent, since they will be modified
        as they are passed up the View3D heirarchy to be displayed by the
        Context3D at the top.
        It will usually be called by it's superview when the
        Context3D at the top of the heirarchy is sent a drawSelf:: message.
        It shouldn't be called directly, only through calls to display.
    */

- display;

    /* This sends itself render then sends display to each subview */

- transformation;

    /* returns the View3D's transformation instance variable */

- setTransformation:anObject;

    /*
        sets transformation = anObject. The object must respond to operateOn
        and operateOn:howMany messages or else an error will be generated when
        the View3D is sent any moveto: lineto: or polygon:howMany: messages
    */

- subviews;

    /* returns the subviews list */
- free;

@end

/* View3D.m */

#import "View3D.h"
#import "Transformation3D.h"
#import "Context3D.h"
#import <objc/List.h>

@implementation View3D

+ new

    self = [super new];
    subviews=[List new];
    superview=nil;
    return self;

- free
    [transformation free];
    [subviews freeObjects];
    [subviews free];
    [return [super free]];



- moveto:(vector3D *)where

    [transformation operateOn:where];
    [superview moveto:where];
    return self;


- lineto:(vector3D *)where

    [transformation operateOn:where];
    [superview lineto:where];
    return self;


- polygon:(vector3D *)vertices howMany:(int)count

    [transformation operateOn:vertices howMany:count];
```

```
    [superview polygon:vertices howMany:count];
    return self;


- superview

    return superview;


- setSuperView:aView3D

    superview=aView3D;
    return self;


- addSubView:aView3D

    [aView3D setSuperView:self];
    [subviews addObjectIfAbsent:aView3D];
    return self;


- render

    return self;


- display

    [self render];
    [subviews makeObjectsPerform:@selector(display)];
    return self;


- transformation

    return transformation;


- setTransformation:anObject

    transformation=anObject;


- subviews

    return subviews;

@end

/* Transformation3D.h */

/*  Transformation3D is an abstract superclass. It should be used for objects
    (such as Linear3D) which implement various 3D transformations that can be
    done to variables of type Vector3D (defined in 3D.h).
    Matrix can handle linear transformations, but subclasses of either
    Transformation 3D or Matrix should be created for
    faster or non-linear operations. */


#import <objc/Object.h>
#import "3D.h"

@interface Transformation3D:Object


- operateOn:(vector3D *)theVect;

/*  - operateOn should be a DESTRUCTIVE routine wich modifies the structure
    which theVect points to in whatever way the subclass desires */

- operateOn:(vector3D *)theVects howMany:(int)count;

/* - operateOn: howMany: should be a way to perform the same operation on
    count Vector3D's. Again it should be destructive, and modify theVects[0]
    through theVects[count - 1] */

@end

/* Transformation3D.m */

#import "Transformation3D.h"

@implementation Transformation3D

- operateOn:(vector3D *)theVect


- operateOn:(vector3D *)theVects howMany:(int)count;


@end
```

```
/* Linear3D.h */

#import "Transformation3D.h"

#define L3D_X_AXIS 0
#define L3D_Y_AXIS 1
#define L3D_Z_AXIS 2

    /*
        these are constants to be used when calling rotate to choose an
        axis.
    */

@interface Linear3D:Transformation3D

    float coeff[4][3];

    /*
        coeff is the coefficients of the transformation matrix
        and the translation.
    */

+new;

    /* initializes coeff such that the transformation is the identity */


- operateOn:(vector3D *)theVect;

    /*
        will perform matrix multiplication with the vector3D pointed to by
        theVect, using the first three rows of the array coeff as a 3D
        matrix, and adding the fourth as a translation vector.
        It will leave the result in the vector3D pointed to by theVect.
    */

- operateOn:(vector3D *)theVects howMany:(int)count;

    /*
        will perform the same operation as operateOn: to the array of vector3D
        pointed to by theVects. It will leave the array itself modified.
    */

- concatBefore:aLinear;
- concatAfter:aLinear;

    /*
        These methods combine the receiver with another Linear3D transformation
        in such a way the the receiver will be modified to perform both
        operations in succession, by doing simple matrix multiplication.
        concatBefore: will make the receiver a
        Linear3D such that it performs the receiver's original operation first,
        and aLinear's second. concatAfter: causes the receiver to perform it's
        original operation after aLinear's.
    */

- rotation:(int)axis:(float)angle;

    /*
        causes the values in coeff to be set such that it represents a rotation
        transformation of angle degrees, around the axis determined by axis
        using the constants L3D_X_AXIS,L3D_Y_AXIS and L3D_Z_AXIS.
    */


- scaling:(float)x:(float)y:(float)z;

    /*
        causes the values in coeff to be set such that it represents a scaling
        transformation,where x,y,z represent the scaling in each direction.
    */

- translation:(float)x:(float)y:(float)z;

    /*
```

```
        causes the values in coeff to be set such that it represents a
        translation transformation.
    */

- (float)coefficient:(int)row column:(int)column;

    /* returns coeff[row][column] */

@end

/* Linear3D.m */

#import "Linear3D.h"
#import <math.h>

@implementation Linear3D

+new

    int ctr1,ctr2;
    self=[super new];
    for(ctr1=0;ctr1<4;++ctr1)
        for(ctr2=0;ctr2<4;++ctr2)
            coeff[ctr1][ctr2]=0.0;


    coeff[0][0]=1.0;
    coeff[1][1]=1.0;
    coeff[2][2]=1.0;
    return self;


- operateOn:(vector3D *)theVect

    vector3D temp;
    temp.x=theVect->x*coeff[0][0]+theVect->y*coeff[1][0]+
        theVect->z*coeff[2][0]+coeff[3][0];
    temp.y=theVect->x*coeff[0][1]+theVect->y*coeff[1][1]+
        theVect->z*coeff[2][1]+coeff[3][1];
    temp.z=theVect->x*coeff[0][2]+theVect->y*coeff[1][2]+
        theVect->z*coeff[2][2]+coeff[3][2];
    *theVect=temp;
    return self;


- operateOn:(vector3D *)theVects howMany:(int)count;

    vector3D temp;
    int ctr;
    for(ctr=0;ctr<count;++ctr)
        temp.x=theVects[ctr].x*coeff[0][0]+theVects[ctr].y*coeff[1][0]+
            theVects[ctr].z*coeff[2][0]+coeff[3][0];
        temp.y=theVects[ctr].x*coeff[0][1]+theVects[ctr].y*coeff[1][1]+
            theVects[ctr].z*coeff[2][1]+coeff[3][1];
        temp.z=theVects[ctr].x*coeff[0][2]+theVects[ctr].y*coeff[1][2]+
            theVects[ctr].z*coeff[2][2]+coeff[3][2];
        theVects[ctr]=temp;

    return self;


- concatBefore:aLinear

    int ctrRow,ctrColumn,ctr;
    float temp[4][3];
    if([aLinear isKindOf:[Linear3D class]])
        for(ctrRow=0;ctrRow<3;++ctrRow)
            for(ctrColumn=0;ctrColumn<3;++ctrColumn)
                temp[ctrRow][ctrColumn]=0.0;
                for(ctr=0;ctr<3;++ctr)
                    temp[ctrRow][ctrColumn] +=
                        coeff[ctrRow][ctr]*
                        [aLinear coefficient:ctr column:ctrColumn];
```

```
        for(ctr=0;ctr<3;++ctr)
            temp[3][ctr]=coeff[3][0]*[aLinear coefficient:0 column:ctr]+
                coeff[3][1]*[aLinear coefficient:1 column:ctr]+
                coeff[3][2]*[aLinear coefficient:2 column:ctr]+
                [aLinear coefficient:3 column:ctr];

        for(ctrRow=0;ctrRow<4;++ctrRow)
            for(ctrColumn=0;ctrColumn<3;++ctrColumn)
                coeff[ctrRow][ctrColumn]=temp[ctrRow][ctrColumn];
        return self;

    return nil;


- concatAfter:aLinear

    int ctrRow,ctrColumn,ctr;
    float temp[4][3];
    if([aLinear isKindOf:[Linear3D class]])
        for(ctrRow=0;ctrRow<3;++ctrRow)
            for(ctrColumn=0;ctrColumn<3;++ctrColumn)
                temp[ctrRow][ctrColumn]=0.0;
                for(ctr=0;ctr<3;++ctr)
                    temp[ctrRow][ctrColumn] +=
                        coeff[ctr][ctrColumn]*
                        [aLinear coefficient:ctrRow column:ctr];


        for(ctr=0;ctr<3;++ctr)
            temp[3][ctr]=coeff[0][ctr]*[aLinear coefficient:3 column:0]+
                coeff[1][ctr]*[aLinear coefficient:3 column:1]+
                coeff[2][ctr]*[aLinear coefficient:3 column:2]+
                coeff[3][ctr];

        for(ctrRow=0;ctrRow<4;++ctrRow)
            for(ctrColumn=0;ctrColumn<3;++ctrColumn)
                coeff[ctrRow][ctrColumn]=temp[ctrRow][ctrColumn];
        return self;

    return nil;


- rotation:(int)axis:(float)angle

    int ctrRow,ctrColumn;

    angle *= 3.14159/180.0;

    for(ctrRow=0;ctrRow<4;++ctrRow)
        for(ctrColumn=0;ctrColumn<3;++ctrColumn)
            coeff[ctrRow][ctrColumn]=0.0;

    coeff[0][0]=1.0;
    coeff[1][1]=1.0;
    coeff[2][2]=1.0;

    switch(axis)
        case L3D_X_AXIS:
            coeff[1][1]=cos(angle);
            coeff[2][2]=cos(angle);
            coeff[1][2]=sin(angle);
            coeff[2][1]=-sin(angle);
            break;
        case L3D_Y_AXIS:
            coeff[0][0]=cos(angle);
            coeff[2][2]=cos(angle);
            coeff[0][2]=-sin(angle);
            coeff[2][0]=sin(angle);
            break;
        case L3D_Z_AXIS:
            coeff[0][0]=cos(angle);
            coeff[1][1]=cos(angle);
            coeff[0][1]=sin(angle);
            coeff[1][0]=-sin(angle);
            break;
```

```
    return self;


- scaling:(float)x:(float)y:(float)z

    int ctrRow,ctrColumn;

    for(ctrRow=0;ctrRow<4;++ctrRow)
        for(ctrColumn=0;ctrColumn<3;++ctrColumn)
            coeff[ctrRow][ctrColumn]=0.0;

    coeff[0][0]=x;
    coeff[1][1]=y;
    coeff[2][2]=z;

    return self;
- translation:(float)x:(float)y:(float)z

    int ctrRow,ctrColumn;

    for(ctrRow=0;ctrRow<4;++ctrRow)
        for(ctrColumn=0;ctrColumn<3;++ctrColumn)
            coeff[ctrRow][ctrColumn]=0.0;

    coeff[0][0]=1.0;
    coeff[1][1]=1.0;
    coeff[2][2]=1.0;
    coeff[3][0]=x;
    coeff[3][1]=y;
    coeff[3][2]=z;

    return self;

- (float)coefficient:(int)row column:(int)column

    if(row > 3 || column > 2 || row < 0 || column < 0) return 0.0;
    return coeff[row][column];

@end

/* Cube.h */

#import "View3D.h"

@interface Cube:View3D

@end

/* Cube.m */

#import "Cube.h"
#import <stdio.h>
#import "Linear3D.m"
#import <dpsclient/wraps.h>

@implementation Cube

- render

    vector3D back[] = .5,.5,.5,.5,-.5,.5,-.5,-.5,.5,-.5,.5,.5,
        front[]=.5,.5,-.5,-.5,.5,-.5,-.5,-.5,-.5,.5,-.5,-.5,
        left[]=-.5,-.5,-.5,-.5,.5,-.5,-.5,.5,.5,-.5,-.5,.5,
        right[]=.5,.5,-.5,.5,-.5,-.5,.5,-.5,.5,.5,.5,.5,
        top[]=.5,.5,.5,-.5,.5,.5,-.5,.5,-.5,.5,.5,-.5,
        bottom[]=.5,-.5,-.5,-.5,-.5,-.5,-.5,-.5,.5,.5,-.5,.5;
    PSsetgray(0.0);
    PSsetlinewidth(3.0);
    [self polygon:back howMany:4];
    [self polygon:front howMany:4];
    [self polygon:left howMany:4];
    [self polygon:right howMany:4];
    [self polygon:top howMany:4];
    [self polygon:bottom howMany:4];
    return self;


@end
```

C Threads: Weaving Your App

by David S. Joerg

of MindShock, Inc.

Focus

The Mach operating system is designed to allow for parallel and distributed processing. C Threads provides an API to Mach's thread-manipulation functions. To quote the C Threads abstract: "The C Threads package allows parallel programming in C under the MACH operating system. The package provides multiple threads of control for parallelism, shared variables, mutual exclusion for critical sections, and condition variables for synchronization of threads."[1]

This article is not an attempt to rewrite or rephrase the existing on-line literature about C Threads (see **Background**). It is not an analysis of parallel processing *per se*, nor a comparison of parallel processing techniques. It totally ignores the other parallel processing solutions available on the NeXT, particularly the Godzilla/MiG approach discussed in the March issue of NUj (then known as Buzzings). Negations aside, this article *is* a brief introduction to the Mach C Threads package, and a discussion of using the package as a *complement* to NextStep programming. We'll go over the caveats, hidden bonuses, and some examples.

Why use C Threads?

Response time for computationally-intensive applications. Some applications need to do computations that will take a noticeable length of time. However, one does not want to ignore the user's actions while one is doing the computation. The typical solution to this problem involves cutting the computation into little pieces and using the timed entry mechanism to execute each sub-computation. Especially if your computation is not easily segmented, C Threads is the way to go.

Elegance. C Threads was not specifically designed to mesh with AppKit programming, so it doesn't feel elegant at first. You can't directly fork a thread into an Objective-C method; you have to use a despised "glue function", a rotten one-liner that calls the Objective-C method for you. After some exposure, however, I think you'll find that C Threads is a well-developed set of ideas. The structural clarity of C Threads is a refreshing break from the tangled world of Responders, Cells, Views, Actions, Speakers, Targets, nib modules, Listeners, Delegates, Windows, and other frustrating but necessary brouhaha.

Speed. For computations that are easily segmented, using multiple threads may lead to large performance gains on a multiple-CPU machine. The multiple-CPU NeXT as of July, 1990, is still everybody's pipe dream, but... I won't tarnish my reputation by discussing baseless rumors. ;-)

A word of caution: C Threads is not part of NextStep; it is part of Mach. Be aware that when you use C Threads in your applications, you are making it that much harder to port your software to an implementation of NextStep that is not based on Mach.

Background

The best online documentation for C Threads is to be found in the *System Reference Manual*, Chapter 16: The Mach Operating System, pages 27-32. You'll find an excellent example program that demonstrates thread forking, mutexes and condition variables. Unfortunately, the example is of the terminal-driven variety, but it does cover the basic concepts.

All the C Threads functions are described at the end of Chapter 27: Mach Functions. A concise listing of the functions is at the end of Appendix I: Summary of Mach Functions.

It is strongly recommended that you at least browse through the above-mentioned sections before reading further. However, this article does not assume that you have done so. Here's an outline of the basic Mach abstractions you'll need to use C Threads:

Machine. Each NeXT is a machine. A machine provides the physical environment in which these fanciful abstractions live and breathe.

Task. A task is a little kingdom. It has its own virtual memory space, its own heap and stack, etc. Each application you run is a separate task, and many system services and utilities run in

their own tasks.

Thread. A thread runs inside a task. Each task has at least one thread, sometimes more. The threads within a particular task are not protected from each other: they all have access to their task's address space, port rights, etc. Since every thread on a machine can potentially be running in parallel, it is very easy for threads to screw each other up, if they're not designed carefully. On single-CPU machines like your NeXT, the CPU deftly jumps from thread to thread, creating the appearance of parallelism.

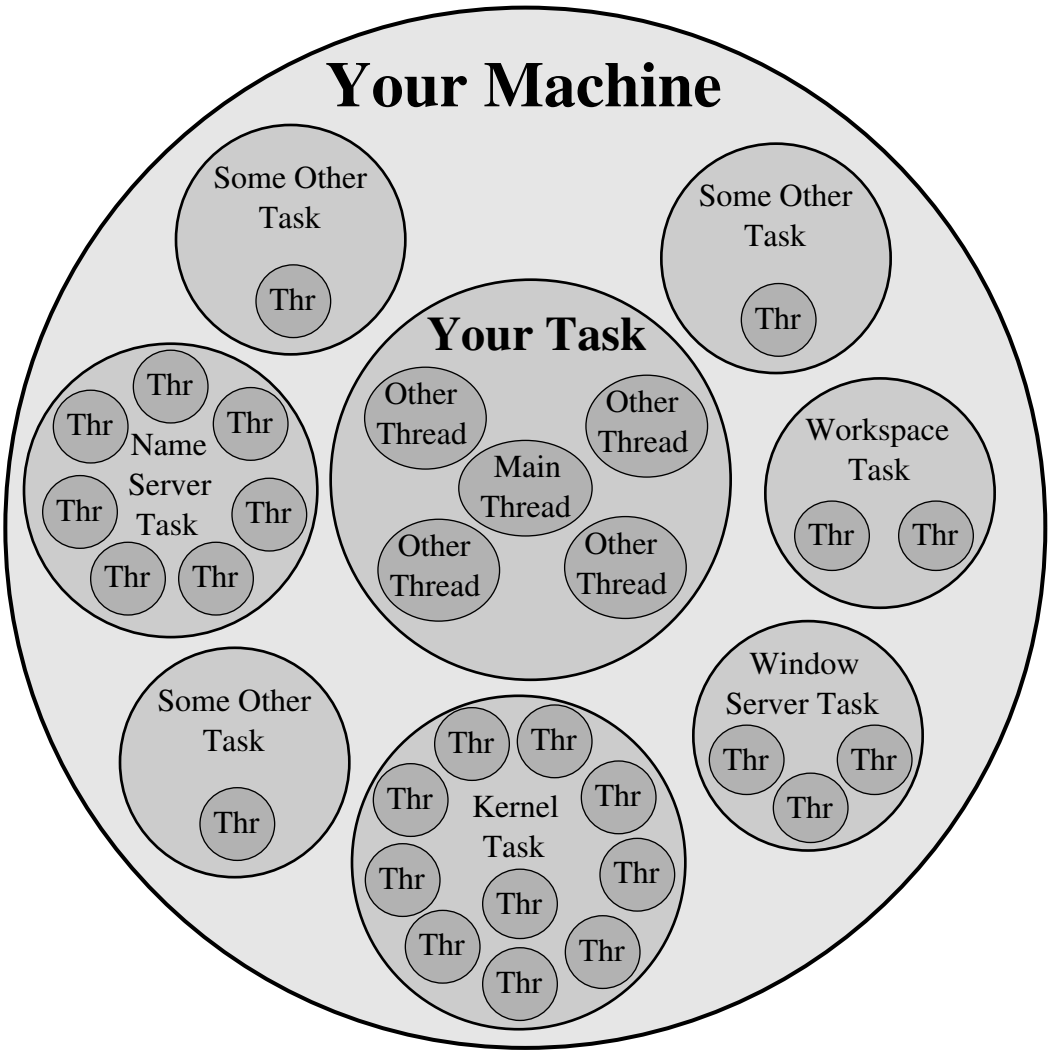


Fig. 1. An illustration of the task/thread concept. Typically, there are many more tasks running on a machine than depicted here. Type `/bin/ps -axgm` for a listing of all the currently running tasks on your machine, and the threads inside them.

The C Threads package introduces some additional abstractions:

Mutex Variable. A lock. You can use it to serialize your thread's access to things: the Window Server, a physical device, or a piece of data, for example. The mutex provides "mutually exclusive access to mutable data", which is necessary to keep your threads from constantly stepping on each other's toes. It does not explicitly lock anything; it works only if you consistently observe an etiquette of lock-usage. If one thread locks a mutex, and then another thread tries to lock that mutex, the latter thread will "sleep" until the former thread unlocks the mutex (signifying it is done using whatever resource the mutex is guarding).

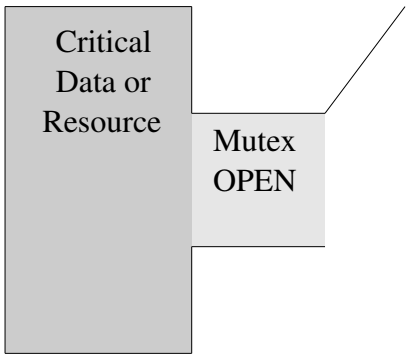


Fig. 2. An illustration of the mutex concept. The mutex, which is guarding some critical data or resource, is open (unlocked).

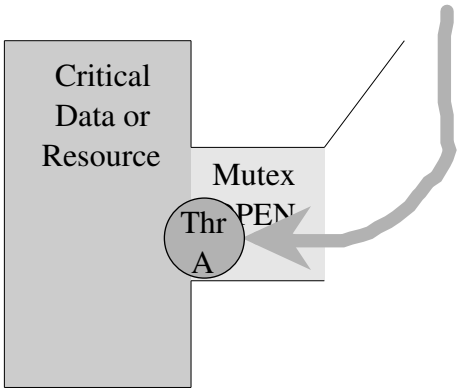


Fig. 3. Thread A needs to use the critical data or resource, so it "enters" the mutex, and "shuts" (locks) it.

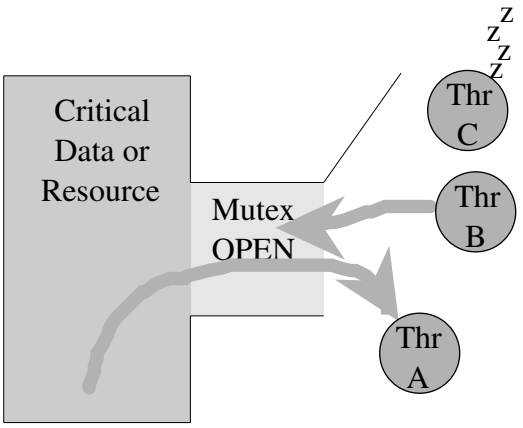


Fig. 4. Thread A freely manipulates and/or reads the critical data or resource. Other threads cannot do so, because when they try to lock the mutex, they "fall asleep". They will slumber until Thread A unlocks the mutex.

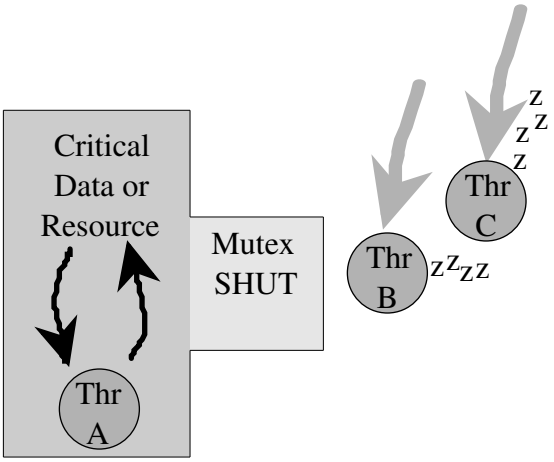


Fig. 5. Thread A is done, it unlocks the mutex, and "leaves". Thread B happens to awaken first, and locks the mutex, so it can begin its access to the data or resource. Thread C will "sleep" until Thread B is done, and unlocks the mutex.

Condition Variable. This is the primary scheduling tool. A thread can do two things with a condition variable: signal it, or "sleep" until another thread signals it. Threads use condition variables to notify other threads when certain events occur. For example, a speech-recognition-shell might have three threads: one that

collects the raw sound data from an input device, another that translates the raw sound into phonemes, and a third that translates the phonemes into a written language. Thread#2, before starting its analysis, would wait for thread#1 to signal a condition variable, signifying that it had collected a certain quantity of sound data. Thread#1 would go on to collect more sound data, while thread#2 worked on phoneme-translation. When thread#2 finished, it would signal another condition variable, waking thread#3. Of course, how a condition variable is used depends entirely on the needs of your application -- some applications don't need them at all, as in the following example.

A Prime Example

Now that we know the basics of C Threads, let's try applying our new knowledge. We will be doing a simple thing: finding prime numbers. Optimally (for us), we'd like to be able to run a repeating loop, that would find new primes, and periodically give results to the user:

```
void findPrimes(void)

   while (done == NO) /* repeat until done */

       /* find some more primes */
       /* give results to user */
```

That's just not good enough. Today's users expect to be in control of an application, or at least have their commands recognized promptly. How can we do this? One solution, a well-trodden path, looks something like this:

```
void findPrimes(DPSTimedEntry te, double now, void *ignored)

   int i;
   /* find GRANULARITY number of primes */
   for (i=0; i<GRANULARITY; i++)

       /* find a new prime */

   /* give results to user */

@implementation SomeAppSubclass
- appDidInit:sender

...
DPSAddTimedEntry(0.0, findPrimes, NULL, NX_BASETHRESHOLD);
...

@end
```

The `DPSAddTimedEntry()` function establishes a "timed entry" that calls the `findPrimes()` function as often as possible, through the application's event-queue. Thus, user input is not ignored, which is good, but... it's still not right. It's a kludge: *using a timed entry for something which is **not** meant to be a periodic occurrence, but rather a continuous background activity.* Here's the right way (IMAO):

```
void findPrimes(int ignored)

   while (YES) /* repeat forever */

       /* find a new prime */
```

```
void updateOutput(DPSTimedEntry te, double now, void *self)

   /* send results to user */

@implementation SomeAppSubclass
- appDidInit:sender

...
cthread_fork(findPrimes, 0);
DPSAddTimedEntry(UPDATE_PERIOD, updateOutput, self, NX_BASETHRESHOLD);
...

@end
```

A few notes on this example:

It is assumed that the list of primes is maintained as global data.

`cthread_fork()` creates a new thread. The new thread executes the function passed as `cthread_fork()`'s first parameter, `findPrimes()`. The second parameter to `cthread_fork()` is, in turn, passed to `findPrimes()`, which happens to ignore it.

`findPrimes()` runs forever. It just finds more and more primes, until the application quits (all threads are instantly slain when the task they are in dies).

`UPDATE_PERIOD` specifies how often, in seconds, `updateOutput()` should be called.

`updateOutput()` is responsible for sending new output to the user, reflecting changes in the list of primes.

OK, we're done outlining concepts. Here's a real app that you can bring to rampaging life on your own NeXT. It's called PrimeThreads, and the "brains" of the app are all in a class cleverly named PrimeThreadsApp. All the files needed to make PrimeThreadsApp can be obtained from the anonymous-ftp archive servers at cs.orst.edu and cc.purdue.edu. If you don't have access to those places, fear not! everything you need to reconstruct the files is right here in this article.

PrimeThreads

1. Create a fresh new directory, start IB, create a new application & save into that new directory. Create a new project into this directory. Put PrimeThreadsApp.[hm] in here.
2. Subclass Application as PrimeThreadsApp. Parse in PrimeThreadsApp. Add the .[hm] files to the project.
3. Change the class of your Application object (the one labeled "File's Owner") to PrimeThreadsApp.
3. Put two TextField's in the window. Make the TextField's wide enough to handle large numbers. Label the TextField's "Number of Primes Found" and "Highest Prime Found". Connect the numPrimesField and highestPrimeField outlets of the PrimeThreadsApp object to the corresponding TextField's.
4. Put a button in the window. Change its title to "Go", and change its alternate title to "Stop". Change its type to "Toggle", and checkmark "Selected". Once you hit the "OK" button in the Button Inspector, the button should read "Stop" (because it's a toggle-type button and it's selected, thus it displays its alt. title). Connect the button to the PrimeThreadsApp object, with an action of toggleGoStop:.
5. Save. Leave IB. Compile through *make*. Run the program. You can use Mathematica to confirm the correctness of PrimeThreads' results. If, for instance, the "Number of Primes Found" field is 2355, type `Prime[2355]` to Mathematica, and "lo and behold!" both Mathematica and PrimeThreads say 20939.
6. Questions, real deep problems, and fan mail to joerg@alliant.mcs.anl.gov.

```
/*********************/
/* PrimeThreadsApp.h */
/*********************/
#import <appkit/Application.h>
#import <cthreads.h>
@interface PrimeThreadsApp:Application

   id             numPrimesField;
   id             highestPrimeField;
   cthread_t primeFinderThread;

- setNumPrimesField:anObject;
- setHighestPrimeField:anObject;
- appDidInit:sender;
- toggleGoStop:sender;
- updateFields;
@end


/*********************/
/* PrimeThreadsApp.m */
/*********************/
#import "PrimeThreadsApp.h"
#import <appkit/Button.h>
#import <appkit/Control.h>
#import <appkit/chunk.h>
#import <dpsclient/dpsNeXT.h>
#import <math.h>

/* the code is much smaller when you take out the comments ;-) */

/*
* thePrimeList is maintained by the NXChunk system.  GROW_BY
* is how much the structure's malloc'ed space will grow, when
* it needs to be expanded.  The chunk is also initialized
* with GROW_BY bytes.
*/
#define GROW_BY 8192
/* every UPDATE_PERIOD seconds, a timed entry calls updateFieldsGlue */
#define UPDATE_PERIOD 0.25

struct primeList
/* see /usr/include/appkit/chunk.h for a description of the NXChunk */
   NXChunk chunk;
/* number of primes in the list */
   int numPrimes;
/* This is just a bogus declaration. There can be any number of primes. */
   int thePrimes[2];
 *thePrimeList;

/* this mutex guards thePrimeList */
```

```
mutex_t primeListLock;

/* this is the function the timed entry calls */
void updateFieldsGlue(DPSTimedEntry te, double now, void *self)
/* we, in turn, call the updateFields method (we're just glue) */
 [(id)self updateFields];

/* findPrimes is executed on its own thread.  it never stops. */
void findPrimes(int ignored)

   int counter, possiblePrime = 5, testLimit;
   BOOL isPrime;

   while (YES) /* repeat forever */

/* isPrime starts at YES.  if an exact divisor is found, it becomes NO */
       isPrime = YES;
/* for any non-prime integer x, one of its factors is less than sqrt(x) */
       testLimit = sqrt(possiblePrime);
     for (counter=1; isPrime && (thePrimeList->thePrimes[counter]<=testLimit);
counter++)
           if (possiblePrime % thePrimeList->thePrimes[counter] == 0)
               isPrime = NO;
       if (isPrime)

/* lock the mutex before changing the list */
         mutex_lock(primeListLock);
/* another prime for the list! */
         thePrimeList->numPrimes++;
/*
* grow the chunk.  usually this does nothing; sometimes more
* space gets allocated.
*/
         thePrimeList =
           (struct primeList *)NXChunkGrow((NXChunk *)thePrimeList,
             thePrimeList->numPrimes * sizeof(int));
/* place our newly-certified prime into the list. */
         thePrimeList->thePrimes[thePrimeList->numPrimes - 1] = possiblePrime;
/* unlock the mutex now that we're done with the list */
         mutex_unlock(primeListLock);

       possiblePrime += 2; /* don't bother with even numbers */
/* tell the scheduler that this is a convenient time to run other threads */
       cthread_yield();



@implementation PrimeThreadsApp

/* IB outlets */
- setNumPrimesField:anObject
 numPrimesField = anObject; return self;
- setHighestPrimeField:anObject
```

```
 highestPrimeField = anObject; return self;

- appDidInit:sender

  /* allocate the mutex */
   primeListLock = mutex_alloc();
  /*
   * initialize thePrimeList with GROW_BY bytes of memory
   * to start with, and make it grow by GROW_BY bytes when
   * it expands.
   */
   thePrimeList = (struct primeList *)NXChunkMalloc(GROW_BY, GROW_BY);
  /* we start the list with 2 primes: 2 and 3. */
   thePrimeList->numPrimes = 2;
   thePrimeList->thePrimes[0] = 2;
   thePrimeList->thePrimes[1] = 3;
  /*
   * start up the primeFinderThread. It will never stop, but we can
   * suspend and resume the thread, which is just as good.
   */
   primeFinderThread = cthread_fork(findPrimes, 0);
  /* since we will never want to join the thread, we detach it. */
   cthread_detach(primeFinderThread);
  /* create the timed entry that will handle updating the fields */
   DPSAddTimedEntry(UPDATE_PERIOD, updateFieldsGlue,
     (void *)self, NX_BASETHRESHOLD);
   return self;


- toggleGoStop:sender

/*
* cthread_thread gets us the actual thread, which is a part of
* the cthread structure.  struct cthread has a lot of other neat
* stuff in it, too, but we don't (explicitly) use any of it.
* thread_resume() and thread_suspend() just want the thread identifier,
* not a cthread, and we give it to 'em.
*/
   if ([sender state] == YES) /* turn it on */
       thread_resume(cthread_thread(primeFinderThread));
   else /* turn it off */
       thread_suspend(cthread_thread(primeFinderThread));
   return self;


- updateFields

/* lock the mutex before reading the list */
   mutex_lock(primeListLock);
   [numPrimesField setIntValue:thePrimeList->numPrimes];
   [highestPrimeField setIntValue:thePrimeList->thePrimes[thePrimeList->num-
Primes - 1]];
```

```
/* unlock the mutex, now that we're done with the list */
   mutex_unlock(primeListLock);
   return self;


@end
```

Full Appkit Integration

PrimeThreads is cute, but it's too simple to shed light upon the major considerations when integrating C Threads and the AppKit. They fall mostly into two categories:

Library Calls. Some library calls make use of global data. Of course, when multiple threads (all sharing the same memory space, as always) are calling such library functions, bizarre and erratic problems will crop up. Chapter 16 of the *System Reference Manual* states, "Unless a library has been designed to work in the presence of reentrancy, the operations provided by the library must be presumed to make unprotected use of shared data. Hence, you must protect against this through the use of a mutex that's locked before every library call (or sequence of library calls) and unlocked afterwards."[2] This isn't as bad as it may sound -- many library functions work fine in a multithreaded environment. You just have to be aware of the potential danger of multiple threads making simultaneous library calls in a single memory space. NeXT is aware of the problem; their RelNotes/LIBCNotes.wn clearly states that they intend to make the LIBC library multithread-friendly.

External Resource Access. Consider this scenario: thread A decides it's time to bring up an alert panel. It does the usual things, calling `NXRunAlertPanel()`, which in turn locks the focus on a particular View, starts drawing things, and so on. In the meantime, thread B decides it's time to draw some text. It locks the focus on a TextField, starts drawing some characters, and so on. *Ouch!* Thread A is still issuing drawing code, because it is completely oblivious to thread B's lockFocus villainry. This sort of problem could also occur with the SoundKit, MusicKit, port access, i/o channels, or any external resource which should be accessed on a one-at-a-time basis. The easy solution is to allocate a mutex that guards the resource. However, this is often not satisfactory. Even though *your* code might obey a mutex-locking etiquette, the AppKit objects you use are not easily coerced into observing any such etiquette. There's no built-in way to tell your AppKit objects, "Hey, every time you call the Window Server, lock this mutex...", unless you want to override every method of every AppKit class (including the secret ones).

I have found it much more effective to restrict resource access to a single thread only, the thread on which your Application object is running. When another thread wants something done, it generates an event of type `NX_APPDEFINED`, which is then picked off the event queue by the Application (running on the main thread), and processed by the Application's (or its delegate's) `applicationDefined:` method. For example:

```
 /* a thread somewhere */
NXEvent theEvent;
...
 theEvent.type = NX_APPDEFINED;
 /* see /usr/include/dpsclient/event.h for the wonderful NXEventData type */
 theEvent.data.compound.subtype = /* whatever subtype you want */
 theEvent.data.compound.misc./*F,L,S,C*/ = /* another 8 bytes of whatever you
want */
 /* lock a global mutex here? */
 DPSPostEvent(&theEvent, FALSE); /* post at end of queue */
 /* unlock a global mutex here? */
...

@implementation someAppSubclass
```

```
- applicationDefined:(NXEvent
*theEvent)
switch (theEvent.data.compound.sub-
type)
  case SUBTYPE_1:
    /* lock focus on a View, perhaps */
     /* draw this & that */
  break;
  /* other event subtypes */
```

I'm not sure if it's necessary to prevent simultaneous access to `DPSPostEvent()`, or if there's already some sort of locking system on the event queue. I've never used a lock for it, and haven't gotten any problems. Then again, I haven't tried it on a multiple-CPU machine.

Don't Try This At Home

Never, *never*, allocate more than a few dozen threads. In my youth, I wrote a nasty little app that just forked threads like mad. After about 300 forks or so, the whole system shuddered in *panic*, and everything crashed. Resource consumption has always been a problem in UNIX-flavored systems, and the solution has always been to have more resources on hand then you'll ever need.

Fear deadlock. It is your enemy. An easy way to achieve this Zen-like "no-mind" state is to lock a mutex, then lock it again, with the same thread. Since you locked the mutex, you're supposed to unlock it when you're done. Of course, you *can't* unlock it, because you're waiting for the lock to become unlocked! It's Ganto's Ax, folks, all over again. Ignore me not, overconfident programmer, who thinks, "Humpf! I'd never be stupid enough to lock the same mutex twice!" It's easier than you think! Imagine, for instance, that you have a mutex that protects the Window Server. Each of your functions that accesses the Window Server, directly or indirectly, locks that mutex before starting its Window Server transactions. If one of these functions happens to call another of these functions, you'll get deadlock when the latter function tries to lock the mutex, which was already locked by the former function. This is another reason why I am in favor of the `NX_APPDEFINED` approach: one thread alone is the task's diplomat to the "outside world".

Masked Avenger

gdb is your best friend. If you didn't know it yet, you know it now. Besides all of gdb's other amazing features, it has some neat commands for debugging multithreaded programs. *thread-list* or simply *tl* will list all the threads in your task. *thread-select* or *ts* will select a thread. The selected thread is the one that's currently being debugged, and is the implicit target of all the other debugging commands like *where*, *step*, and *next*. *tsuspend* will suspend a thread (add 1 to its suspend count). *tresume* will resume a thread (subtract 1 from its suspend count). A suspend count of 0 means the thread is running; a higher suspend count means the thread is suspended. You might want to try these commands on PrimeThreads, especially to see a living example of the suspend count, and how it is affected by the "Stop/Go" button. Don't be surprised if the `findPrimes()` thread's stack seems to be in limbo sometimes; this happens whenever it calls `cthread_yield()`. Warning: never tell gdb to suspend thread #0, because gdb itself will become wedged. Don't ask me why; probably has something to do with ports or whatnot.

Free Neat Stuff

I lied; I told some people I was going to discuss parallel maze-solving stuff in this article. Sorry, it just didn't get debugged in time. However, when it is ready, it will be good, it will be put on the archive servers.

Another "neat thing" that *someone* should do *immediately* is a C Threads/NextStep visually intuitive demonstration of the "Dining Philosophers" or "Philosophical Programmers" or "Dead Philosophers" problem. Let the user choose the number of philosophers, and choose from different scheduling/synchronization strategies. Then watch the philosophers struggle in an all-out battle for food and prestige!

References

1 - Eric C. Cooper, Richard P. Draves. *C Threads.* Department of Computer Science, Carnegie Mellon University. Draft of 20 July, 1987.

2 - *System Reference Manual*. Chapter 16: The Mach Operating System. Version 1.0 [981.00]. NeXT, Inc.

Tao by Robert Lin

rlin@cs.ucb.ca

The NeXT is a machine that galvanizes people. It has only been around for less than two years, but it has already secured a loyal band of enthusiasts. Witness, for example, the number of user groups and newsletters that have been organized. Can you name any other workstation that can boast that many SIGs, with that many newsletters, in such a short time?

I have noticed, however, that most of what gets published is news release from NeXT, and there isn't much in the way of a user's commentary column. Furthermore, most newsletters deal with facts and not rumours.

So, combining my two interests, I decided to put out a tabloid-style newsletter called ***Tao****. It is filled with rumours, gossips, opinions, and speculations. Portions of it tend to be highly technical, and sometimes highly controversial.*

I am currently up to issue #3. Some of the highlights from past issues were:

• The Thin Wallet NeXT, ideas on getting the most NeXT bang for your buck;

• All About External SCSIs, on adding more storage for your cube;

• The Future of NextStep;

• SLIP and the NeXT, building a wide area network over the telephone;

• The NeXT Portable, a theoretical portable NeXT, backed by some facts.

And of course lots of juicy rumours and gossips.

Tao is available for anonymous ftp from j.cc.purdue.edu or cs.ubc.ca (under next/Tao). It is formatted for both WriteNow and ASCII. People who don't have ftp access can e-mail me at rlin@cs.ubc.ca for a copy, or use the Purdue e-mail server. Finally, free paper subscription is now available. For your copy, write to: Robert Lin/1701 West 64th Ave.

Vancouver, BC V6P 2P3/Canada/Please leave your name, address, phone number, and the name of your organization. You can also fax it to me at (604) 261-5324. (See below)

The Future of NextStep

NextStep is the soul of the NeXT computer. It is what makes the machine unique. When Steve Jobs set out to create NeXT, he did not have all the answers. He merely found the people who had the answers, and gave them the freedom to be at their best.

If we look at the NeXT machine in cold objectivity, we find a machine like many other UNIX machines. The Floptical drive is something available to any micro or workstation; the DSP option can be had on Suns, Macs, PCs, and others; SCSI, ethernet, and big monitors are all old news; MACH is funky but offers the users nothing over stock UNIX.

The true genius of NeXT is the author of NextStep. The fluid integration of Display PostScript, Objective C, and the AppKit leap-frogs other me-too UNIX boxes. It is the gains in programmer productivity that sets NeXT apart.

The large number of bundled software is surely significant, but it is only made possible with NextStep as a solid foundation. We may have very few of the bundled software, if NextStep didn't encourage productivity the way it does.

People talk about how application software drives the hardware market. But what drives the application

Free Paper Subscription!

People who'd like to get a paper subscription of Tao should clip out the form below and fill it out. Fax it to (604) 263-5324, or send it to: *OSE Corp./Attn: Robert Lin/1701 West 64th Ave./Vancouver, BC V6P 2P3/Canada.* The other way to get Tao is anonymous ftp from a NeXT archive. I will upload to the UBC (cs.ubc.ca) archive and the Purdue (j.cc.purdue.edu) archive. People without ftp access can send e-mail to archive-server@cc.purdue.edu, with subject of "help", to get information on how they can receive software through e-mail.. For people who don't have NeXT machines, or work off text terminals, I have prepared an ASCII version of the newsletter. I cannot guarantee how many issues you get in a year, or how long I can maintain free subscription, or even if the magazine will be around at all tomorrow. A lot of it depends on support from NeXT and from the readers. If you like Tao, tell your NeXT Rep!

Organization _____

Name_____

Address_____

City_____Province/State_____Postal Code_____

I have a NeXT machine YES / NO; I have access to a NeXT machine that isn't mine YES / NO

Signature_____

software? Some would say it is large market shares. I would agree, but point out also that good software can only come about from good application platforms. Look at the mess in DOS world. Millions of applications, and almost all limited by the 640K memory barrier. In fact, it isn't the quantity of application software that counts, it is rather the quality and coverage of software that matters.

We have seen how NextStep promotes quality software, because so much of the hard work for writing quality software has been done for you already. We have also seen how NextStep promotes coverage. Since it is easy to write programs, more people will write more code on more different areas.

Now comes the market share problem. Why would people write NextStep programs, when only 10,000 cubes have been sold, according to NextWeek? The R/6000 isn't even on the market yet. But what if NextStep was available on every 386? All of a sudden, the picture changes.

There is no technical barrier that stands in the way. An 80386 with a Weitek FPU is more powerful than the 68030 in just about every conceivable way. The new generation 486's are at least twice as fast as their 386 counterparts, and completely compatible. The standard Super-VGA graphics that most 386s support will give you 800 by 600, with 16 colors, and 1:1 aspect ratio. Many VGA boards also give you 1024 X 768, with 256 simultaneous colors. That compares favorably with the MegaPixel display.

IBM has the right and the resources to put NextStep on 386 boxes. This would not be another pretty DOS front-end, since NextStep does demand UNIX as the underlying operating system. UNIX doesn't have to be expensive; Marc Williams Software has a $99 UNIX called Concurrent for PC, and it runs happily within 640K.

It is to both IBM and NeXT's interest to sell NextStep/386 for cheap, bundled with perhaps AIX/386. IBM would instantly create a huge market for NextStep applications, a situation that can only help its new R/6000. NeXT would extract royalty from potentially millions of copies sold. The DOS crowd could even be pleased with a DOS compatibility box, much like that of OS/2 or Windows 3.0.

Such a move would devastate Microsoft's grand plans. Windows 3.0 and OS/2 Presentation Manager can never hope to match NextStep. Even its most ardent supporters would defect, once they find themselves free of the cursed GDI.

It is a pain to deal with Microsoft's GUIs. They only look similar to a user. One needs different API for Windows 3.0, PM, and Motif. Worse yet, the operating systems calls for all three are different. Finally, until Royale font technology is implemented, the programmer has no true device independence. Companies stick by Microsoft right now for one simple reason: the huge PC market, with everyone eager to go GUI. Give them a better way, and they will rally behind NextStep.

There's been a persistent rumor that NextStep has been seen running on a PS/2 Model 70. I believe the current NextStep application base is strong enough to convince any 386 user to dump Windows or OS/2, if given a choice. It is now up to NeXT and IBM to show both the wisdom and the courage to popularize NextStep.

Odd Bits & Pieces

NextStep for PS/2 sighted: at **Uniforum** in Washington D.C., NextStep was seen running on a PS/2 under **AIX** 1.2. Performance was sluggish, according to eye witness. One explanation for this is either the lack of a math coprocessor, or the wimpy **80387** being a bottle neck for a floating point-intensive GUI.

I have just received report from Bob Ellefson at **UCSD** that they are now using NextStep on the PS/2 for an IBM joint study.

Take heart, NeXT programmers. Our code may yet one day appear on every 386 PC desktop!

Hot off the rumor mill: IBM will be offering NextStep for **RS/6000** at $1,500. The entire operating system, TCP/IP, NFS, etc., including NextStep and **Ingres** database, will be $2,500. Sounds like another OS/2 buster.

The word from new RS/6000 users is that AIX is full of bugs. One NeXT developer considering porting software over to RS/6000 confided, "We won't be buying the RS/6000 at least until the next release of AIX. Why should we pay good money to hunt down bugs for IBM?"

Persistent rumors call for a faster **Optical Drive** in the new release of NeXT machines. My question is: what about the people who upgrade? Will they, too, get faster ODs?

NeXT has promised **version 2.0** system software to be shipped soon, perhaps as of 4th quarter 1990. Amongst other things, greater speed is on top of the list. I am told the launch time for applications is substantially decreased, so that even on 68030 machines, applications will launch noticeably faster.

Heard on the Internet:

NeXT will be using the same memory chips for the 68040 as the 68030. When they upgrade your board, all they do is move the memory right over. For those of you who have purchased extra memory, perhaps from third party, that's great news: your investment is secure.

Are you running out of ttys on your NeXT? There are currently two ways to solve the problem: a SCSI device that branches out into a cluster of serial and Centronics parallel ports, and Cisco's ethernet device that gives you a whole bunch more serial ports.

I am currently running a laser off ttya, a modem off ttyb, and I could use about two more serial ports.

Just received **Diagram** from Lighthouse Design Inc., a wonderful little diagramming tool that I will review in the next issue of Tao. If you are a programmer doing documentation, or any other forms of diagrams, you'll want to check this

out.

Lighthouse Design is also working on **Exploder**, a CASE tool for programming with persistent objects. It uses Sybase as the back-end storage for these persistent objects, and generates Objective-C code. Look for a review within the future pages of Tao.

If you are not using **Stuart** from Scott Hess, get it today! Stuart takes the best features of Terminal and Shell and adds other goodies along the way. Download it from any of the popular ftp archive sites, or ask your NeXT representative to get you a copy of this very useful piece of shareware.

After using Stuart and going back to Terminal and Shell, I felt like deleting the pair from my hard disk. I used to maintain three copies of Terminal on my Icon Dock and one copy of Shell. Now only Stuart is evident.

For you NeXT historians: Fortune magazine featured an article on how **Jobs** and **IBM** hooked up, and all that political back-biting between Microsoft and IBM, in the October 9, 1989 issue. It told of how Jobs met IBM Chairman **John Akers** at a party, Steve told John what he thought of OS/2, and John countered, "How would you like to help us?"

If this thing with PS/2 works out, it may be IBM that helps NeXT, not the other way around.

Amongst other things, the article pegged the deal between IBM and NeXT at the neighborhood of $50 million the first year.

BusinessLand finally sets up shop in Tokyo. The store is also funded by other Japanese companies, including Canon Inc., a big NeXT share holder, which reportedly holds 6% interest of BusinessLand Japan.

Four months ago, I saw a prototype of Japanese kanji running on NeXT. I wouldn't be surprised if all this signals a thrust from NeXT into the potentially lucrative Japanese market.

NeXT Inc. is offering its employees option to buy stocks, at $1 a share, or so I've heard. If I were a NeXT employee, I'd buy as much as I am allowed; at $1 a share, it's a bargain. The cynics say those stocks will be worthless when IBM/ DEC/Sun/Apple/what-have-you walks all over NeXT. But you know, I've seen their software and I am willing to bet hard cash on NeXT.

Computex Report

You should have seen it. A computer trade show the size of Comdex, featuring all the latest high tech toys from Taiwan. That's the 1990 Computex, recently held in Taipei.

The show grabber is without a doubt the new line of 80486 motherboards. I saw at least six companies with **486 EISA motherboards**. I played with a 486 box running SCO UNIX. I couldn't believe the speed! It felt more like a DEC 3100 RISC box. Benchmarks show the 486 to be anywhere from two to three times faster than 386.

The 486 has an integrated math coprocessor, which gave it excellent floating point performance. I'd love to see how it fares with AIX and running NextStep.

The Taiwan N.T. dollar is currently very strong against the US dollar. This has made life difficult for Taiwan exporters, especially when it's been accompanied by rising labour cost. Now Taiwan, like Japan did with its cars, is going upstream to high end products, where profit margins are better.

In pushing into the 486 market, traditional Taiwan clone makers find themselves in brand new territories. Suddenly, there isn't anybody to copy anymore. Indeed, the Taiwan computer industry is gradually shifting from Mr. Copy Shabby to innovative designs. I saw demonstrations of bus mastering SCSI controllers, parallel processing transputers, etc. All based on EISA and all using Surface Mount Technology.

Even the clones are becoming more high end. **Twinhead Corp**. of Taipei announced they will make **SparcStation** compatibles, at half their price. That puts Sparc power squarely in PC dollar range. This follows an announcement by **Hyunda**i to make super fast Sparc clones.

I also counted five companies that are now manufacturing high speed, **9600 V.32 MNP 5 modems**. Competition is bound to drive the price of modems down. Street price for a good 9600 bps V.32 modem is around $700 to $800 US right now. In a year, it'll be half that.

Finally, a Taiwan consortium is trying to break Japan's monopoly on **CCD** (Charge Coupled Device), a prime component used for scanners, fax, and digital cameras. By manufacturing their own CCD, consortium members hope to further cut prices on all of the above.

SLIP Into the Future

All NeXT machines come with ethernet built-in. Using TCP/IP and NFS, NeXT works as a harmonious member of UNIX networks. We even have this marvelous ability to run a NextStep application on one machine, but see it on another machine across the network.

When you specify the -NXHost command, a NextStep application automatically invokes NSWS (see your man page about it) to establish a TCP/IP link with another machine on the network. The target machine runs NSWSd, the daemon that intercepts the stream and directs it into the Display Postscript Server.

With this ability, we can use computers in new and wonderful ways. This is the concept behind an Application Server (see issue #1); instead of running the program on your machine, which may have limited memory and storage space, and maybe even slow CPU, why not run it on a big machine next door, then send all the display code to your machine?

With the advent of RS/6000 and the forth coming 68040 NeXT, we will see a proliferation of Application Servers. It won't matter so much that your machine is only a 68030 with no hard disk. The fact is, by off-loading the Display Postscript Server (DPS) stuff from the Application Server, you get better over-all performance. You are effectively

putting two processors to work on one application.

What makes all this possible is a very compact display protocol, Postscript. When your program sends its output to a DPS, it is a tightly encoded binary stream. The division of responsibility between the DPS and the application is designed to minimize communications overhead.

X-windows was designed around the very same idea, except X doesn't have the Postscript component. It turns out that the division of responsibility in X is different than that in NextStep, and the communications burden is higher.

This is all very good and fine if you've got a network, but what about the Rest of Us?

Well, I've got news for you. Everyone's got a network, a Wide Area Network (WAN) in the form of telephone lines.

TCP/IP talks over ethernetlines, but you can replace the ethernet with dial-up phone lines using **SLIP** (Synchronous Line Internet Protocol). With SLIP, you can interconnect computers using standard telephone. As far as your software is concerned, the machine two hundred miles away looks exactly the same as the way next to you.

Of course, the transmission speed is no where near that of ethernet, but with the new modems coming out, it can be quite impressive.

In the old days, when 2400 bps was the standard, SLIP would have been unbearably slow. But now we've got 9600 bps V.32 modems. And better yet, a new generation of V.32 modems now offer V.42bis, an international standard for data compression. With V.42bis, text data travels at at 4 times the speed, or 38,400 bits per second, on the average.

The compression algorithm used is a derivative of the famous **Lempel-Ziv Welch** algorithm, which looks for repeating patterns in a data stream, replacing them with single tokens. Its performance entirely depends on the type of data being sent. English text can typically be compressed to one-fourth size, and I'd expect SLIP packets of DPS stream to average about half to one-third size.

Running a remote application over SLIP would be noticeably slower than over ethernet, but it is entirely feasible. And when ISDN becomes a reality, we will have 64Kbps, which makes all this even more feasible.

This makes possible a whole new class of services. You could, for example, log into a software publisher's computer to "test drive" their new application. The vendor need not make a special crippled "demo" version, and they need not worry about piracy.

SLIP can also be used to query remote databases. Sybase servers communicate with the clients through TCP/IP, so if you run the front end on your machine, you can still access all the remote data easily and transparently from a remote Sybase server, all through the magic of SLIP.

Database packets are typically very compressible. Since all rows are returned one at a time in Sybase, user perceived response would be excellent. In fact, it'd be just as good as "being there".

So the next logical question is, how can I get SLIP? Well, you can't, not at the moment. In their foolishness, NeXT did not put in the supporting hooks into the TCP/IP kernel driver code. NeXT says they may change that in version 2.0. They'd better!

Portable NeXT, Part II

Three months ago, I wrote about a hypothetical portable NeXT. The article is reprinted here (apologies to those of you who've already read it), with new comments at the end.

The NeXT machine, wonderful though it may be, is desk bound. When we NeXT devotees travel, we are forced to put up with mediocrity. It is hard to go back to PCs, Macs, or (shivers) terminals, after being spoiled by the cube. The answer: we need a portable NeXT!

It makes perfect sense. Every NeXT is equipped with an ethernet connector, so the devotees need not fumble with RS232 cables, nor deal with LapLink type software. The portable NeXT should have an internal hard drive (but no optical disk). On the hard drive will be the bare minimum systems software we need to get things done on the road: no Shakespeare, etc. The portable will need plenty of RAM as to minimize or avoid swapping, which chews up quite a bit of disk space.

What about the monster 17 inch screen? Some compromises are inevitable here. But since all the software is PostScript based, there is an immediate advantage: device independence. Granted, we will probably have only six docks left, but at least all the WriteNow documents will look properly spiffy and wysiwyg.

Thanks to all the work already done on VLSI, the component count should be low, especially now that the optical disk controller is no longer needed.

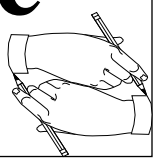
At Computex Taipei, I saw a beautiful electro-luminescent screen with 1024 X 864 resolution, is incredibly sharp, has 16 levels of grey, and has a viewing angle of 120 degrees. Best of all, it is flat. Sounds to me like the ideal laptop screen.

By the way, by laptop I don't mean battery powered. I have never found battery power to be any sort of advantage, except perhaps for the frequent flyer. To me, it's more important to have a sharp monitor than to squint at a MegaPixel LCD for hours at a time.

I'd like to see a minimum of 12 megs of RAM, or 3 banks of 4M SIMMs, on the laptop motherboard. An 8M system just isn't enough. After Postscript Server and Workspace Manager loads, there is hardly any room left, and the disk ends up thrashing all the time.

SIG HardCore

moderated by
Ian Smith
iansmith@warhol.gatech.edu



Parallelism and the NeXT Workstation

by Ian Smith & Mike Gourlay

Welcome to another installment of NeXT Hardcore, the special interest group for systems hackers on the NeXT workstation. First, some housekeeping... If you have something that you think would be interesting to NeXT hardcore readers, please mail it to : next-hardcore@-warhol.gatech.edu , or iansmith@warhol.gatech.edu, or (in the worst case)

Ian Smith/1004 Curran St./Atlanta Ga, 30332.

We would love to have ideas and/or articles from readers, as this allows us to target our audience more effectively.

In this article we will be discussing how to simulate parallel processing on your NeXT computer by using "threads." Threads are so-named because they are used to implement multiple parallel lines of execution through a program. Threads are defined to be processes that execute in the same virtual address space (as opposed to "normal" processes which have their own virtual address space). Threads can be told to share their time on the CPU in a couple of different ways: statically or dynamically. In this article we will motivate threads through the use of an example which generates the mandelbrot set "in parallel." Additionally, we will use a shared queue, to demonstrate how threads may share data space and more effectively communicate.

By using threads, a program is (at least theoretically) being executed in two (or more) places at once. Threads are "lightweight" compared to a unix process (called a "task" in the Mach environment), as a thread is just an execution unit, nothing else. If several threads are executing a program in parallel there is only one copy of the program, one copy of the data with multiple places in the program being executed simultaneously. Thus, thread parallelization is much cheaper than the standard unix fork operation which makes a completely new copy of the program on each fork. (A standard unix process is comparable to a Mach task running one thread.)

On the NeXT computer, of course, there is only one CPU so the threads are not really executing in parallel, but simulating this behavior by clever scheduling. However, this fact should really be ignored as much as possible when thinking about thread behavior and programming, as if you were to port multi-threaded software to a machine that had multiple CPUs, the threads may in fact execute in parallel. (We can only hope that NeXT brings out a board with multiple CPUs to fit in our cubes... then we can really have some fun.) Due to the way Mach is implemented on the NeXT as far as I have been able to determine, the threads are not pre-emptively scheduled ("coroutine implementation" for you Mach afficianados), so you must give the scheduler clues on when to switch between the threads.

As mentioned above, threads share data space, so global data is available for reading and writing to all threads. There is an obvious problem here: What if one thread reads some data, decides to act accordingly, and while the action is in progress another thread modifies the original value to something else? This leads one into a quandry of "who does what first?" This is classicly called a "race condition." To solve this problem, you must be able to implement "mutal exclusion." This is a process by which a thread (or other parallel execution entity) has exclusive access to a data structure. Mach provides you with the means to implement this via the mutex_t data type. Anytime you call mutex_lock() on an initialized mutex_t data item, when the function call returns you are guaranteed to be the only person in the "possession" of that lock. By using mutex locks, multiple threads may lock each other out of "critical sections" of their code which must be executed atomically. In the example we give below all the threads share a queue of output results. It is crucial that when a thread updates the count of items in the queue that he is the only one who is performing that action. We use a call to mutex_lock to insure that this occurs.

It is also frequently necessary for one thread to wait on another thread to perform some task. When we are generating the mandelbrot set, we may have a thread that wants to pop some data off the queue, but there is nothing currently in it. This thread that wishes to remove data from the queue then needs to wait on some other thread to put some data in the fifo for its consumption. This is implemented through calls to condition_wait() and condition_signal(). The thread wishing to wait for a condition uses condition_wait() to wait on a condition, which is asserted by some other thread which executes a condition_signal to inform the waiting thread that the condition has been fulfilled. Condition_wait() takes two parameters, one is a mutex_t described above, and the other one is a condition_t. This data type is useful only to name the signals and waits that the program is executing. In our example we have two condition_t data items "freed" and "data". The freed condition is asserted when a thread pops some data from the stack, and the data signal is asserted when data is pushed onto the back of the stack.

Above I mentioned that you have two choices on how to schedule your threads, dynamically or statically. A static scheduling of the threads results in the work being divided up into n groups and each of n threads is turned loose on his particular portion of the work. When a thread finishes his portion of the work, he simply stops.

By contrast in a dynamically scheduled environment, when a thread finishes his current job, he determines if there is more useful work that he can be doing, and he does that. In this manner, n threads may or may not break the whole job into n pieces, but in fact may use considerably more than n processes. This is useful when different portions of the job take different amounts of time to compute (as in our Mandelbrot example), so that threads which finish quickly may continue doing work that otherwise would have to wait.

The example program listed below puts all the things I have been discussing together. It spawns any number of threads, and then uses the threads to generate the Mandelbrot set. It is broken into two files, 'pmandel.c' and 'queue.c'. The first is the main program, which forks off the threads and does the mathematics necessary for the calculation. The program takes a command line argument to inform it how many work threads to spawn. Other than the work threads (doing the calculations) there are two other threads in use; these are the dynamic scheduler (who assigns jobs to the the work threads) and the main thread, which reads data off the data queue and prints results on standard output. If you wish to see the effects of static scheduling on the program, compile the program with -DSTATIC. Note: This isn't true static scheduling, its really being simulated by the dynamic scheduler, but the effects are still visible.

The most important thing to the successful execution of the program is the shared data structures-- the queues. The code to do the shared fifos is in the file 'queue.c'. We have implemented a general purpose shared fifo for multi-threaded applications. The way we chose to make the fifo general purpose is by using the mtq_fifo_t type. You may declare variables of this type, and pass them to mtq_init. Each structure passed to mtq_init creates a new fifo, upon which the mtq_push and mtq_pop operations will operate. You may have a queue of different data structures operating in the same program (as we do) since all pertinent information is contained in the structure and the fifo can figure out how to operate based on the information. You may consider these structures opaque if you wish, but the idea of mixed data type fifos may be appealing to some. (You could implement a non-wasteful type of union this way.)

Included is a little curses program that will take the output of the main program and display it on a terminal. It displays numbers in ascii that correspond to the intensity value of the 'point' on the display. We would love to have an appkit type front end for the program, but we just did not time to do it. We are currently porting the program to some parallel architectures; we will hopefully have the comparative results of these for the next issue.

```
/* file: pmandel.c
        Multi-threaded Mandelbrot generation
        by Mike Gourlay and Ian Smith

        to compile this program for dynamic scheduling use:
        cc -O -o pmandel pmandel.c
        to compile this program for static scheduling use:
        cc -O -o pmandel -DSTATIC pmandel.c
        usage: pmandel n
        n is the number of work threads you wish to spawn.  Be sure
        not to make n too large (>200) as it may cause your machine to
        go into a system panic
*/
#include <stdio.h>
#include <cthreads.h>
#include <stdlib.h>
#include <math.h>
#include "que.h"
#include "que.c"
/* point_color_t is 2-D coordinate with a dependent value. */
typedef struct {
  double x;
  double y;
  int color;
} point_color_t;
/* window_t defines the upper right and lower left corners on 2_D */
typedef struct {
        double  Xi,     /*first x value */
                Xf,     /*last x value */
                Yi,     /*first y value */
                Yf;     /*last y value */
} window_t;
/* function declarations */
void    mandel(),       /* thread to compute mandelbrot points */
        dynamic_schedule();     /* fills the work que */

/* GLOBALS */
double  mandel_dX,      /* Mandelbrot resolution */
        mandel_dY;
int     NumThreads=10,  /* the number of threads to parallelize over */
        RotatePeriod=40,/* how often each thread yeilds to another thread */
        MaxIter=300,    /* Maximum number of mandelbus iterations */
        Width=79,       /* physical screen width in pixels */
        Height=24,      /* physical screen height in pixels */
        WorkQueSize,    /* depth of the work que */
        MandataQueSize=50,      /* data que depth */
        XInc=2, YInc=2; /* size of job in points */
char    Author[]="Michael J. Gourlay & Ian E. Smith";
/* if the origin is at ULC, YFLIP=1  and YOFFSET=0
   if the origin is at LLC, YFLIP=-1 and YOFFSET=Height */
int YFLIP  =1,
```

```
    YOFFSET=0,
        XFlip=1,
        XOffSet=0;
mtq_fifo_t      work_que,       /* que info structures */
                mandata_que;    /* for work and data ques */
window_t        window; /* total window area */

main(argc,argv)
     int argc;
     char **argv;
{
  int           ThreadNum,              /* current thread */
                TotalArea;              /* area in pixels of window */
  double        AreaPerThread;          /* area of the plane for each thread */
  point_color_t val;                    /* generic point for working */
  window_t      mandel_thread_window;   /* thread window area */
  if (argc<2) {         /* num of threads manditory command line arg */
    fprintf(stderr,"usage: %s threads\n",argv[0]);
    exit(1);
  }
  NumThreads=abs(atoi(argv[1]));/* number of threads to run simultaneously */
  WorkQueSize=NumThreads;       /* depth of work que */
  cthread_init();               /* gotta do this */
  mtq_init(&work_que, sizeof(window_t), WorkQueSize);   /* init fifo */
  mtq_init(&mandata_que, sizeof(point_color_t), MandataQueSize);
  window.Xi=-1.5;       /* region of the complex plane to view */
  window.Xf= 1.5;
  window.Yi=-1.5;
  window.Yf= 1.5;
  mandel_dX=(double) fabs((window.Xf-window.Xi)/Width); /* define resolution */
  mandel_dY=(double) fabs((window.Yf-window.Yi)/Height);
#ifdef STATIC           /* for static instead of dynamic scheduling */
  TotalArea = Height*Width;
  AreaPerThread=(double)TotalArea/(double)NumThreads;
  YInc= (NumThreads>=TotalArea) ?
        1 :
        (NumThreads-1)*(1-Height)/(TotalArea-1)+Height;
  XInc= AreaPerThread / YInc;
  if(XInc < 1) XInc=1;
  fprintf(stderr,"area x y %f %d %d\n", AreaPerThread, XInc, YInc);
#endif
  /* spawn the threads */
  /* administrate the tasks */
  cthread_detach(cthread_fork(dynamic_schedule, &window));
  /* spawn theMandelbus threads */
  for (ThreadNum=0; ThreadNum<NumThreads; ++ThreadNum)
    cthread_detach(cthread_fork(mandel, NULL));
  /* pull the info off of the data que */
  ThreadNum=0;
  while(ThreadNum<NumThreads) {
        mtq_pop(&val, &mandata_que);
        if(val.color==-1) {             /* a thread died */
                ++ThreadNum;
        }
        else                            /* data came in */
                printf("%d %d %d\n",(int)val.x, (int)val.y, val.color);
  }
}

/*
Fill the work que with regions of the complex plane to be computed by
the Mandelbrot routines .
*/
void dynamic_schedule(window_t *window) {
  window_t      mandel_thread_window;
  int           Xindx, Yindx, ThreadNum;
  double        Xtmp, Ytmp;
  /* fill the work que */
  for(Xindx=0;
      (mandel_thread_window.Xi=window->Xi+mandel_dX*(double)Xindx)<=window->Xf;
      Xindx+=XInc)
        for(Yindx=0;
            (mandel_thread_window.Yi=window->Yi+mandel_dY*(double)Yindx)<=window->Yf;
            Yindx+=YInc) {
                mandel_thread_window.Yf =
                        (Ytmp=window->Yi+mandel_dY*((double)(Yindx+YInc-1)+.5))>=window->Yf
                        ? window->Yf            /* don't go out of bounds */
                        : Ytmp;
                mandel_thread_window.Xf =
                        (Xtmp=window->Xi+mandel_dX*((double)(Xindx+XInc-1)+.5))>=window->Xf
                        ? window->Xf
                        : Xtmp;
                /* push the subregion onto the work que for mandel() to use */
                mtq_push(&mandel_thread_window, &work_que);
        }
  /* tell the threads to kill themselves */
  for(ThreadNum=0; ThreadNum<=NumThreads; ThreadNum ++) {
        mandel_thread_window.Xi = 0.0;
        mandel_thread_window.Xf = 0.0;
        mandel_thread_window.Yi = 0.0;
        mandel_thread_window.Yf = 0.0;
        mtq_push(&mandel_thread_window, &work_que);
```

```
  }
}
/*****************************************************************************/
/* Mandelbrot routines */
static int MandelIter ();
static double graphx(), graphy();
/* the Mandelbrot thread */
void mandel() {
  point_color_t MandelData;
  window_t work;/* work que item; has assignments */
  int   YIter;  /* what line this point is on */
  int   XIter;  /* how far along the real plane is this point */
  double        Ytmp;

  for(;;) {
    mtq_pop(&work, &work_que);  /* get the work assignment from the work que*/
    /* no more work, kill the thread */
    if(((work.Xf-work.Xi)==0.0) && (work.Yf-work.Yi)==0.0) {
        MandelData.color=-1;    /* tell the main that this process has died */
        push(&MandelData, &mandata_que);
        return;
    }
    for(YIter=0;        /* along the y's */
       (MandelData.y=mandel_dY*(double)YIter+work.Yi) <= work.Yf;
       YIter++) {
         for(XIter=0;   /* along the x's */
            (MandelData.x=mandel_dX*(double)XIter+work.Xi) <= work.Xf;
            XIter++) {
                MandelData.color=MandelIter(MandelData.x,MandelData.y);
                /* scale the data to screen coordinates */
                MandelData.x = graphx(MandelData.x) + 0.5;
                Ytmp = MandelData.y;
                MandelData.y = graphy(MandelData.y) + 0.5;
                /* push the computed data onto the data que for main to use */
                mtq_push(&MandelData, &mandata_que);
                MandelData.y = Ytmp;
         }
     }
   }
}

/* The Mandelbrot iteration algorithm. */
/* Mandelbrot set as defined by Benoit Mandelbrot, IBM Labs in Zurich */
int MandelIter (double x, double y) {
        register double a=0.0, b=0.0, atmp =0.0;
        register int    ColorIter=0;
        while((ColorIter < MaxIter) && ((a*a + b*b) < 4.0)) {
                atmp = a*a - b*b + x;
                b = 2.0 * a * b + y;
                a = atmp;
                if(ColorIter%RotatePeriod == 0) {
                        /*printf("yielded %d\n",ColorIter);*/
                        cthread_yield();
                }
                ColorIter++;
        }
        return(ColorIter);
}

/* map the complex plane coordinates to screen locations */
double graphx(double x) {
        return(fabs((x+window.Xi)/(window.Xf-window.Xi)) * (double)Width);
}

double graphy(double y) {
        return(fabs((y+window.Yi)/(window.Yf-window.Yi))
                        *(double)Height*(double)YFLIP + (double)YOFFSET);
}
/*****************************************************************************/


/*
        file: queue.c
        by  Michael J. Gourlay
        general purpose multi-threaded queue
*/

static int mtq_next();

/* initialize everything */
void mtq_init(
        mtq_fifo_t      *fifo_info,
        size_t          element_size,
        unsigned int    fifo_size)
{
  fifo_info->last  = 0;
  fifo_info->first = 0;
  fifo_info->lock  = mutex_alloc();
  fifo_info->freed = condition_alloc();
  fifo_info->data  = condition_alloc();
  fifo_info->fifo_depth = (fifo_size>1) ? fifo_size : 2;
  fifo_info->item_size  = element_size;
```

```
  fifo_info->fifo  = malloc(element_size*(size_t) fifo_info->fifo_depth);
}

/* Push a value on the fifo */
void mtq_push(
        void            *datum,
        mtq_fifo_t      *finfo)
{
  char *cp;
  mutex_lock(finfo->lock);  /* lock so that pushes are atomic */
  /* if fifo not full then push */
  while (mtq_next(finfo->last, finfo->fifo_depth)==finfo->first)
        condition_wait(finfo->freed, finfo->lock);
  bcopy((char *) datum,
          (char *) finfo->fifo+(finfo->last*finfo->item_size),
          (int)    finfo->item_size);
  finfo->last=mtq_next(finfo->last, finfo->fifo_depth);
  mutex_unlock(finfo->lock);  /* give someone else a chance */
  condition_signal(finfo->data);  /* tell pop that we just pushed something */
}

/* pop a value */
void mtq_pop(
        void            *datum,
        mtq_fifo_t      *finfo)
{
  mutex_lock(finfo->lock);  /* pop operations are atomic */
  /* is it empty? if so, wait on someone to push */
  while (finfo->last==finfo->first)
        condition_wait(finfo->data,finfo->lock);
  bcopy((char *) finfo->fifo+(finfo->first*finfo->item_size),
          (char *) datum,
          (int)    finfo->item_size);
  finfo->first=mtq_next(finfo->first, finfo->fifo_depth);
  mutex_unlock(finfo->lock);
  condition_signal(finfo->freed); /* tell push that we freed some space */
}

int mtq_next(   int index,
                unsigned int MaxFifoSize) {
        if(index==(MaxFifoSize-1))
                return(0);
        else
                return(index+1);
}

/*
        file: drawit.c
        by Ian Smith
        to compile this program use:
        cc drawit.c -O -o drawit -lcurses -ltermcap

        We have had some problems in using the program from some types
        of terminals, especially dial-ups at slow baud rates. If you
        have such a problem, inserting some type of delay into the
        main loop of the program will help. (Say, computing the sin of
        26 a few hundred times...)
*/

#include <stdio.h>
#include <curses.h>

main(argc,argv)
     int argc;
     char **argv;
{

  int x,y,i;
  char c;

  initscr();
  leaveok(stdscr,TRUE);
  scrollok(stdscr,FALSE);
  clear();
  refresh();
  while (!feof(stdin)) {
    fscanf(stdin,"%d %d %d",&x,&y,&i);
    if (i!=0)  {
      c='0';
      c+=i % 10;
      if (mvaddch((int)y,(int)-x+79,c)==ERR) ;
    }
    refresh();
  }

  endwin();
  sleep(10);
}
```

MUMPS NOW AVAILABLE FOR THE NEXT COMPUTER!

Arthur Lee@Rocky.LabMed.Washington.Edu

For this issue, I've only had time to write a very brief introductory article. Another, more detailed article, should be ready for the next month.

In this article, I will attempt to describe the MUMPS programming language and how one can obtain it for the NeXT. I hope to write a longer article that describe specific MUMPS programming examples in the next issue. This article is simply an introduction.

WHAT IS MUMPS?

MUMPS, the programming language, is now available for the NeXT. MUMPS is a programming language developed in the mid-1960's at the Massachusetts General Hospital in Boston, Massachusetts. MUMPS is an acronym for Massachusetts General Hospital Utility Multi-Programming System. Although associated with the medical community, MUMPS is eing used for virtually every type of scientific, technological, health service and corporate data processing system. It is most often used for on-line, interactive applications where immediate data retrieval and prompt response are important.

MUMPS is a multi-purpose, multi-user, high-level programming language designed for interactive data management applications. Its string-handling features and embedded database capability makes the language suitable for applications where large, complex files must be designed, constructed and maintained. These capabilities of MUMPS makes it an ideal language for developing systems requiring a shared database for real-time transaction processing.

MUMPS is an ANSI language, which means that all implementations of Standard MUMPS use the same commands, functions, operators, and data structure. This enables programs to be transported between different computers running Standard MUMPS. In talking about programming languages, people often have a very difficult time defining what exactly constitutes a "standard" for a given language. Portability of MUMPS programs is hard to beat and other languages would find it difficult to beat this "feature."

MUMPS is also a Federal Information Processing Standard (FIPS). The Institute for Computer Science and Technology develops FIPS for programming languages when there are significant benefits for federal users, and when technically sound specifications exist. As a result of FIPS classification, federal departments and agencies may select MUMPS as their language of choice.

WHY SHOULD YOU CONSIDER MUMPS?

MUMPS is simple to learn yet powerful to use. It gives the novice programmer the tools to develop professional programming applications. Experienced programmers find that the language facilitates the solving of complex data management problems. Its simple structure and interactive nature help the programmer to create and debug programs quickly.

Studies show that MUMPS significantly enhances programmer productivity over other languages. A typical MUMPS program requires fewer lines of code than other languages. As a result, MUMPS programmers spend much less time for system development.

Another important attribute, mentioned previously, is program portability. MUMPS programs can be easily moved between different computer hardware and MUMPS implementations.

UNDER WHAT OPERATING SYSTEMS DOES MUMPS RUN?

MUMPS runs under several operating systems. Among these are the following:

- *CCDOS (Chinese characters)*
- *IBM VM*
- *Unix*
- *DG AOS*
- *Macintosh OS*
- *VAX/VMS*
- *MS-DOS*

WHICH COMPUTERS RUN STANDARD MUMPS?

MUMPS runs on a variety of hardware systems, ranging from microcomputers through large mainframes. Among these are systems made by the following:

Altos	*Apple*	*AT&T*
BBN	*Bull HN*	*Burroughs*
Compaq	*Data General*	*DEC*
Epson	*Fujitsu*	*Gould*
Harris	*Hewlett-Packard*	*Hitachi*
IBM (micros and mainframes)		*Intel*
ITT	*Mitsubishi*	*Motorola*
NCR	*NEC*	*Prime*
Pyramid Technology		*Sun*
Unisys		

WHO PROVIDES MUMPS FOR THE NEXT?

MUMPS has been made available on the NeXT by Plus Five Computer Services of St. Louis, Missouri. Plus Five provides MUMPS for Unix systems and they have just recently ported MUMPS onto the NeXT. They are currently working on taking advantage of all the "features" of the NeXT and building it into their MUMPS implementation.

Plus Five Computer Services
1968 Innerbelt Business Center Drive
St. Louis, MO 63114
(314) 426-3900

NEXT TIME-What is MUMPS, really, and what can it do? Program examples will be included.

Great Scott!

This Month : News from the Hessian Front and The Taming of the Sh (and Csh too!)

Scott Hess

Well, Stuart1.0-b just got finished downloading to cs.orst.edu:pub/next/submissions. It should eventually make its way to j.cc.purdue.edu (gerritt?). Following is the README file from the distribution (well, the top part, at least).

If you have an old version of Stuart, don't wait to get it! This is a fairly substantial improvement over 1.0- and 1.0-a. *I* almost like it.

Why is it that I always end up doing alot of Stuart stuff the days I have to go out of town? Must be a law of nature. I should go out of town more often.

Fixes

OK, this is the realease that should have been 1.0-a. Fixed from 1.0-a are:

1) Character sizing bug. Now fonts other than Ohlfs-10 should work.

2) Scrolling bug. nn manifested a bug when scrolling. This seems to work with nn, now, but WorkPerfect on VMS still manifests a similar bug.

3) Print Cancel. The cancel button on the print panel used to require three cancels to really figure out what you want. This wasn't my fault - appkit bug. But, I used an alternate method, so now that works.

4) Selection is a _little_ bit better. This is touchy, so I didn't modify much. I've tried to make it select the character you're on more than it selects the character to the left, or the line below. Also fixed is the bug where the selection wouldn't find the end of the line in the scrollback buffer, sometimes. Actually, now that I think of it, selection is a _lot_ better :-) Still not very smooth, and not line or word selection.

5) Do to VMS's usage of lf, pasting is now strange. Under VMS, lf means delete previous word. Most copy/cut/paste on the NeXT uses lf to indicate end of line. Stuart copies using lf to indicate eol for compatibility with other programs. When the Translate default is YES (that's the default value), then lf's are translated into cr's on paste. This works fine for unix, I think, but if you have programs which don't like it, turn Translate off. This is only availiable from the command line using dwrite/dread.

6) backgrounded processes no longer hang the window if you tryto exit the shell while they have stdio open.

New Stuff

Well, 1.0-b is not _really_ what 1.0-a should have been, its more. I've taken some of the new, improved Stuart 1.0 stuff and added where I could. New stuff:

If you have an old version of Stuart, don't wait to get it! This is a fairly substantial improvement over 1.0- and 1.0-a. I almost like it.

1) Backwrap and Autowrap defaults. Autowrap is the regular vt100 autowrap feature, and defaults to YES. Backwrap is a feature which Terminal and xterm have, where backspace at the left-hand side of the screen wraps backwards. Backwrap defaults to YES, also. Backwrap is only active when Autowrap is active.

2) Added a Jump to Bottom on the Window Menu. Works like Shell's version.

3) Not really anything added, but by using the Meta default, you can have ESC be a meta-prefix for when you press alternate-key. key must be on the main keyboard for this to work (keypad and alternate have other meanings). Using ESC over -1 (set high bit) works better for me, because, by default, rlogin doesn't pass the high bit, and I need the meta key on other workstations. I'm lazy, too. Using ESC works well with both csh and tcsh. To use:

```
dwrite Stuart Meta 27
```

or use Preferences.

4) The Shell default may now work. I've got it set up to correctly parse the shell command passed, and seperate out the arguments and all. This means that more complex Shell defaults work (such as -Shell "telnet vax1"). Within the Shell default, quoted characters are passed, and quotes can be either "" or ". \ quotes \, ", and ', both inside and outside quotes. ' is passed in "", and " in ".

Lastly, in the "just implemented last night" category, I've done some neat stuff. Due to the lateness at which this was implemented, this stuff is _definitely_ experimental. I've not gotten any crashes due to it, but what does that mean?

Speaker/Listener

1) I've added a Speaker/Listener interface to Stuart.Whee! That was fun (?).The interface essentially allows an outside program to send a bunch of defaults and parameters to Stuart, and Stuart will start up a new window using them.This is certainly _not_ advanced.I only had a couple hours!

I've supplied a program, called soil (if you know where Stuart came from, you know about soil), to demonstrate this usage. This program may be used sort of like a little Stuart. Anything on the command line which it thinks looks like a default is packaged and sent to Stuart.Thus, -DefName value is sent as DefName=value.If the parameter after a -DefName start with a -, then the DefName parameter is sent w/o a value.Watch out!I tried once doing something like -Meta -1, and you guessed it!The -1 looked like a parameter. So, for something like this, do -Meta " -1".

Soil supports the defaults Stuart uses, plus the NXHost defaults and a new Activate default.NXHost specifies the host to contact Stuart at.Activate specifies a flag to pass to [NXApp activateSelf:flag] within Stuart.Specifying just -Activate w/o a value assumes YES for the flag value.If Activate is YES, then Stuart activates himself when after he starts up the new terminal.

.stuartrc

2) Since I had the Speaker/Listen-

er interface, I decided to add the ability to do a .Stuartrc file.This file contains setup for Stuart when he runs.The Stuartrc default specifies the file to use.If the name begins with ~/, it is relative to the user's home directory.The default value is ~/.Stuartrc. In the .Stuartrc file, there are lines of the form:

```
DefName=DefValue# comment.
```

Anything between a # and \n is ignored, unless the # is quoted by "", '', or \.defValue may be multiple words, and is terminated by \n.When either a | character, or the end of file, is encountered, a new window is started with the accumulated defaults.These defaults override the real defaults, and any real defaults not overridden are also used.

After the new window is started, the defaults are reset, and another window may be read.Fun, fun fun.- For instance, the following is what I have in ~/.Stuartrc:

```
Shell=/usr/hosts/mcs-server # log a window onto the server
```

SourceDotLogin=NO # rlogin doesn't like the -.

```
WinLocY=734 # above the local window.
```

| # bong!

This start a session on the host mcs-server, and also one on the local machine.(The | starts the mcs-server window, and the EOF starts the local one.)Since my other defaults are set up to put the windows along the doc, 80 columns, etc, I get one window (mcs-server) on top of the other.

If .Stuartrc doesn't exist, the old Stuart behaviour still works - a single, default window is brought up.If .Stuartrc is empty, the same thing should happen (as the EOF will bring up a new window).

Note that the Speaker/Listener interface follows the exact same model.In fact, I added the ability to soil to read the standard input when no parameters are given, and send that on to Stuart.Also, both the Speaker/Listener interface and the .Stuartrc file do not recognize the WinLocD[xy] parameters, and, in fact, reset the window position to the default position for the next window which is newed

Thank You

Thanks to Robert Lin <rlin@cs.ubc.ca>, author of Tao and iwscript, for jiggling my brains last night. I wouldn't otherwise have thought to add the .Stuartrc stuff.

What the Future May Hold

OK, so what's this you might have heard about that famous grail, Stuart 1.0? I don't know. I've gotten alot done on it, and it even allows me to login, again :-). Actually, it is coming along well, and I plan to have a beta version for my testers sometime next week. w/o lots of the new stuff. But, eventually it will, like at the end of the month of July.

But, I don't think Stuart 1.0 will be substantially different from Stuart1.0-b. I really don't think I'll be able to both get 1.0 up to the 1.0-b capabilities, _and_ add more. 1.0 will mostly be an improvement in scrolling and selection. Possibly a speed increase (no promises, there).

One thing that 1.0 (may) have over 1.0-b is documentation. I have a feeling most of it needs trashed . . .

scott hess

scott@gacvax1.bitnet

LATE BREAKING NEWS

OK. Finally, I think I've got a solid 1.0- Stuart. Who knows? I don't, as everything I can think of testing works. The only solid bug which doesn't is when the Line default is set to 32, and the second window never comes up and the third crashes bug. Also, the reported but not followed up bug with WordPerfect on VMS, and a hard to repeat bug which pulls lines from other windows when a new window is created. All of which were in 1.0-.

Anyhow, the main thing I added to 1.0-c is new documentation. If you don't trust it for anything else, grab it for the documentation. Since it is entirely new, I would recommend that even long-time users read the new docs. They aren't too long, and they are in roff -man format, which I found more concise than using NeXT-style documentation. It also looks really nice when ptroff'ed, and I have grown to intensly dislike WriteNow, anyway.

I suppose I should give instructions for non-hackers. To view the manual pages, you would more than likely want to run the following on each:

```
nroff -man Stuart.1 >Stuart.man
```

replacing Stuart with the appropriate filename. This will give a straight ASCII version of the file. To print them, do:

```
ptroff -man files
```

Don't print the nroff version - the ptroff version looks a _lot_ nicer. Of course, the files could be placed in /usr/man, and reindexed . . .

A problem was reported for 1.0-b where new windows wouldn't come up. That happened here once, too. The problem seems to manifest when you copy a new version of Stuart over the old, and some buffers somewhere don't get updated. Anyhow, it went away when the user logged out and back in. When installing Stuart, it may be best to

power off and back on. This goes for any time you do it, with any version. Alternately, it may be enough to first delete the old version, and then copy the new version in. Also, make sure no executing copies are out there when you install it!

A note to all users who are waiting for 1.0: You're waiting for a chimera. 1.0 is not going to be anything amazing for 1.0-c users. In fact, most users a probably will not be able to tell the difference between 1.0-c and 1.0. That is my aim for 1.0. 1.0-c is solid (sounds familiar), and I would be tempted to say it is 1.0 if I wasn't so picky about my code. It crashes on me less often than earlier versions did. I removed Shell from my Dock sometime during the day. I even forced one of my local users to switch to it (he'd been using a semi-fixed 1.0-), and he even said it looked solid. Anyway, I'm sick of 1.0-. Unless bugs come in which crash things, or are otherwise dangerous, 1.0- has met the end of the line. I've better things to work on. It it now out on cs.orst.edu, coming soon to a NeXT near you.

The Taming of the sh (and the csh, too).

Scott Hess

A short time ago I downloaded the ShellPanel utilities by Christopher Lane (lane@sumex-aim.stanford.edu). These utilities provide access to the SavePanel, OpenPanel, and AlertPanel from the command-line, and appear to be useful additions to a shell programmer's toolkit. Even though I have little use for such tools (I try to avoid shell programming, though it is often quite fun), they did prompt me to think of another interesting command-line tool: Text.

In the same spirit as the ShellPanel utilities, the Text utility combines a NeXTStep(TM) interface with a unix style program. Text takes ascii input, and displays it in a Text object in a window, resulting in a combination of a Text object and the more(1) program.Text is also amazingly flexible, considering its small size.

How to Compile It:

1) Create a new application with IB, saving it as Text/Text.nib. Create a project in that directory from the Project window. This should be done immediately (more later).

2) Place TextApp.[hm] in the Text directory, then make a TextApp as a subclass of Application. Parse in the class, and make sure the file is in the project. Make the file's owner an instance of TextApp.

3) Remove the main window and the info panel (yes, really!) Pull out a Panel, and drag a ScrollView into it from Palettes. Make sure the Panel has a resize bar, close, and miniaturize buttons.

4) Connect the ScrollView to the text outlet of the File's Owner,and the File's Owner to the delegate outlet of the Panel.

5) Cut out the following icon:

with Icon, and save it into a file named MiniWindow.tiff. The icon is 48x48, with alpha values or without make no difference. Drag this icon into the Icons suitcase of InterfaceBuilder.

6) Remove the Main Menu by cutting it. I did not want the menu cluttering up things. Also, due to the way in which the run method of the application works, the main menu will not be useable. If you did not save before removing the Main Menu, then start over. Without a main menu, InterfaceBuilder thinks that the .nib file cannot be the application's main .nib file, so it will not be loaded.

7) Type make from the command line in the Text directory.

Source: TextApp.h

```
#import <appkit/Application.h>

@interface TextApp : Application
{
   id    text;
   int miniWindow;
}

- setText:anObject;

@end
```

Source: TextApp.m

```
#import "TextApp.h"

#import <appkit/ScrollView.h>
#import <appkit/Text.h>
#import <appkit/Panel.h>
#import <appkit/Font.h>
#import <appkit/defaults.h>
```

```
#import <streams/streams.h>
#import <appkit/SavePanel.h>

#import <sys/file.h>
#import <sys/errno.h>
#import <libc.h>
#import <stdlib.h>
#import <ctype.h>
#import <sys/param.h>

                                // a short method of getting defaults.
#define def( name) NXGetDefaultValue( "Text", name)
#define idef( name) atoi( def( name))
#define fdef( name) atof( def( name))

extern int getwd( char *);

@implementation TextApp : Application
{
   id    text;
   int miniWindow;
}

        /*
          This captures output as it comes in, and places it in
          the output Text object.  I had to do the last display
          because otherwise I got a disgusting looking caret at
          the end of the text.  I though selectNull was supposed
          to take care of it?  I think it was a bug, because even
          after I started selecting text, the caret stayed on.
          The explicit redisplay kills it nicely, though.
        */
void handleInput( int fd, struct Text *output)
{
#define buffSize 4096
  static char buffer[ buffSize];
  int ret=read( fd, buffer, buffSize);
  if( ret>0)
    {
     int len=[output textLength];
     [output setSel:len :len];// get to the end of the text,
     [output replaceSel:buffer length:ret];// and output our data.
    }
  else
    {
     DPSRemoveFD( fd);// assume we're done,
     if( ![output textLength] && idef( "Close"))
       [NXApp abortModal];
     else
       {
```

```
            [output selectNull];// get rid of caret,
            [output display];// and force the caret away.
                                                    // If the user can edit,
            if( idef( "Edit") || def( "Save"))
              [output setEditable:YES];// set editable on.
            else
              [output setSelectable:YES];// allow user to select text.
        }
    }
}

        /*
          This is called when the time is up, if the Time default
          was >0.  It just removes itself from the TimedEntry queue,
          and calls the application to abort the modal loop.
        */
void killIt( DPSTimedEntry te, double time, id self)
{
 DPSRemoveTimedEntry( te);
 [self abortModal];
}

        /*
          Since the object which is passed is actually a ScrollView,
          I get the docView for the text outlet.  Also, I set up
          the miniwindow icon, and set the miniWindow number to -1
          to indicate no miniWindow, yes.
        */
- setText:anObject
{
 text = [anObject docView];
 [[text window] setMiniwindowIcon:"MiniWindow"];
 miniWindow=-1;
 return self;
}

        /*
          This routine is overridden so that I can catch command-key
          equivalents, since there is not a main menu in the .nib
          file to catch them for me.  I also need to catch the
          double-clicks in the miniWindow (if miniaturized) because
          the application runs in and modal event loop.  That means
          the regular double-clicks are ignored when not in the
          main window.
        */
-(NXEvent *)getNextEvent:(int)mask waitFor:(double)timeout threshold:(int)level
{
 NXEvent *e;

        /*
```

```
        I find gotos hideous, but I think this is one of those times when
        their use may be justified.  I will have you know that this is
        the _first_ goto I've used in c.  The first time through, I used
        a recursive call, but I thought that looked too awkward.
       */
loop:

                                        // Really go get the event.
 e=[super getNextEvent:mask waitFor:timeout threshold:level];

                                        // if its a command-key event,
 if( e && e->type==NX_KEYDOWN && e->flags&NX_COMMANDMASK)
  switch( e->data.key.charCode)// handle it.
   {
      case 's' :                        // save
       if( ![text isEditable])// if not editable,
         break;                         // makes no sense to save.
      case 'w' :            // close, quit and save are effectively the same thing,
      case 'q' :
       [self stopModal];// just stop the loop
       goto loop;
      case 'a' :                        // select all.
       [text selectAll:self];
       goto loop;
      case 'c' :                        // copy
       [text copy:self];
       goto loop;
      case 'x' :                        // cut
       [text cut:self];
       goto loop;
      case 'v' :                        // paste.
       [text paste:self];
       goto loop;
      case 'p' :                        // print.
       if( [text isSelectable])
         [text printPSCode:self];
       goto loop;
      case 'm' :                        // miniaturize
       if( miniWindow!=-1)// if already mini,
         {                                      // get back the main window.
          [[self findWindow:miniWindow] deminiaturize:self];
          miniWindow=-1;
         }
       else
         [[text window] miniaturize:self];
       goto loop;
      default :
       break;
   }
```

```
                                        // if its a double-click in the miniWindow,
  if( e && e->window==miniWindow && e->type==NX_MOUSEDOWN && e->data.mouse.click>1)
    {                                   // bring up the main window.
      [[self findWindow:miniWindow] deminiaturize:self];
      miniWindow=-1;
    }
  return e;                     // return whatever event we eventually got.
}

static NXDefaultsVector defs=
{
  {"Title", NULL},          // Title of the Panel.
  {"Width", "512"}, {"Height", "256"},// Size of the Panel.
  {"Time", "0"},            // Time to live for Panel.
  {"NXFont", "Helvetica"},// Font to use in Panel,
  {"NXFontSize", "12"},// Font's size.
  {"Command", NULL},// Command to use as input.
  {"File", NULL},           // File to use as input.
  {"Burst", "1"},           // Read input before displaying Panel.
  {"Close", "0"},           // Close Panel if no input.
  {"Edit", "0"},            // Let user edit, and output result.
  {"Save", NULL},           // Where to put the Editted result.
  {"Backup", "1"},          // Make backup of Editted File?
  {"Overwrite", "0"},       // Force overwrite if file exists?
  {NULL, NULL},
};

        /*
          I override this method to load up the defaults, and
          display the output window.  I use runModalFor: to do
          this, because I do not want an application icon, or a
          menu (not that there is a menu in the .nib file . . .)
        */
- run
{
  id font;
  float time;
  const char *command, *file, *save, *title;
  FILE *f=stdin, *s=stdout;// default to stdin for input.

                                        // register our defaults.
  NXRegisterDefaults( "Text", defs);

                                        // If we are to read from a file,
  if( (file=def( "File")) && *file)
    {
      f=fopen( file, "r");// open that file,
      if( !f)                   // and error out if couldn't.
        {
          NXRunAlertPanel( "Unable to open file.",
```

```
                        "The file \"%s\" could not be opened.",
                        "OK", NULL, NULL, file);
        return self;
       }
  }                                 // If really to use a command,
else if( (command=def( "Command")) && *command)
  {
   f=popen( command, "r");// open a file from that command,
   if( !f)                  // and error out if couldn't.
     {
      NXRunAlertPanel( "Unable to execute command.",
                        "The command \"%s\" could not be executed.",
                        "OK", NULL, NULL, command);
        return self;
       }
  }

                                    // If in Burst mode,
if( idef( "Burst"))
  {                                 // open a Stream on the file,
   NXStream *st=NXOpenFile( fileno( f), NX_READONLY);
   [text readText:st];// and read from there.
   NXClose( st);
                                    // If no text and Close, exit.
   if( ![text textLength] && idef( "Close"))
     return self;
   else if( idef( "Edit") || def( "Save"))// If the user can edit,

       [text setEditable:YES];// set editable on.
  }
else
  {
   [text setSelectable:NO];// don't allow selection,
                                          // and set a routine to catch the output.
   DPSAddFD( fileno( f), (DPSFDProc)handleInput, text, NX_MODALRESPTHRESHOLD);
  }

                                    // size the window as indicated in defaults.
[[text window] sizeWindow:idef( "Width") :idef( "Height")];

                                    // get the indicated font.
if( font=[Font newFont:def( "NXFont") size:fdef( "NXFontSize")])
  [text setFont:font];

if( (time=fdef( "Time"))>0)// get time and use it if >0
  DPSAddTimedEntry( time, (DPSTimedEntryProc)killIt,
                    self, NX_MODALRESPTHRESHOLD);

                                    // set the title
if( !(title=def( "title")) || !*title)
```

```
  if( file && *file)
    title=file;
  else if( command && *command)
    title=command;
  else
    title="Text";
[[text window] setTitle:title];

[self activateSelf:YES];// make us active (really)

[self runModalFor:[text window]];// and get into a run loop.

if( idef( "Edit") || def( "Save"))// If the user could edit,
  {
    NXStream *st;                   // a stream for our writing.
    if( save=def( "Save"))// If there is a save default listed,
      {
        if( !*save && file && *file)// if it doesn't specify a file,
          {                                             // but the input was from a file,
            if( idef( "Backup"))// if Backup is specified
                {                                       // make a backup filename.
                  char *t=strcat( strcpy( malloc( strlen( file)+2), file), "~");
                  if( rename( file, t)==-1)// if couldn't rename, ask user.
                    switch(  NXRunAlertPanel( "Cannot make backup.",
                                "Could not make a backup of %s.  Overwrite anyway?",
                                "Cancel", "Overwrite", "Save Panel", file))
                      {
                       case NX_ALERTDEFAULT :
                            case NX_ALERTERROR :
                              return self;
                            case NX_ALERTOTHER :
                              file="";          // This will force a SavePanel.
                            default :
                              break;
                      }
                }
            save=file;                  // let the save name be the file name.
          }
        else
          {                                             // if no save name, but was a command,
            if( !*save && command && *command)
                {                                       // copy the command string.
                  char *t=strcpy( malloc( strlen( command)+1), command);
                  char *p;
                  char brkchars[]=" &|;$";// (may not be complete).
                  for( p=t; *p; p++)// get to the end of the command's name.
                    if( index( brkchars, *p) || iscntrl( *p))
                      {
                            *p=0;
                            break;
```

```
                    }
                  if( rindex( t, '/')) // get the basename.
                    t=rindex( t, '/');
                  save=t;                       // make that the savefile.
                 }

                                                // if that file's there, and not Overwrite mode,
            if( access( save, F_OK)==0 && !idef( "Overwrite"))
              switch(  NXRunAlertPanel( "File exists.",// ask the user.
                                        "The file %s already exists.  Overwrite?",
                                        "Cancel", "Overwrite", "Save Panel", save))
                  {
                    case NX_ALERTDEFAULT :
                    case NX_ALERTERROR :
                      return self;
                    case NX_ALERTOTHER :
                      save="";                  // This will force a SavePanel.
                    default :
                      break;
                  }
             }
            if( !*save)                         // if we don't have a filename yet,
              {
               id savePanel=[SavePanel new];// look at a savePanel.
               char path[ MAXPATHLEN];
               if( getwd( path))        // put it into the current directory.
                 [savePanel setDirectory:path];
               [savePanel runModal];// go for it.
               save=[savePanel filename];// make that the save name.
              }
            if( !save || !*save)            // _still_ no filename!
              return self;                  // just give it up as a bad job.
            else if( !(s=fopen( save, "w")))
              {
               NXRunAlertPanel( "Unable to open save file.",
                                        "Text was unable to open file %s for saving.\n",
                                        "OK", NULL, NULL, save);
               return self;
              }
          }
                                                // Write the text to the output file.
      [text writeText:st=NXOpenFile( fileno( s), NX_WRITEONLY)];
      NXClose( st);
    }

  return self;
}

          /*
            When the window is closed, we need to stop the modal
```

```
        loop.
       */
- windowWillClose:sender
{
 [self stopModal];
 return self;
}

       /*
        When the window miniaturizes, we need to set up the
        miniwindow number so we can catch events.
       */
- windowWillMiniaturize:sender toMiniwindow:mw
{
 miniWindow=[mw windowNum];
 return self;
}

@end
```

Usage

The Text command recognizes the following defaults, either from the command line, or in the defaults database:

Name	Default value	Description
Title	(Text)	Title of the Panel.
Width, Height	(512, 256)	Size of the Panel.
Time	(0)	Time to live for Panel (Time<=0 means forever).
NXFont	(Helvetica)	Font to use in Panel,
NXFontSize	(12)	Font's size.
Command	()	Command to use as input.
File	()	File to use as input.
Burst	(1)	Read input before displaying Panel?
Close	(0)	Close Panel if no input?
Edit	(0)	Allow user to edit text, output when done?
Save	()	See discussion below.
Backup	(1)	Backup editted Files?
Overwrite	(0)	Overwrite if file exists?

In general, Text will be used from the command line to display the output of some command. For instance:

```
ls -l | Text -Title "ls -l of `pwd`"
```

gives a listing of the current directory, with a title which gives the name of the directory and the command being executed on it (ls -l). This effect could also be achieved with:

```
Text -Title "ls -l of `pwd`" -Command "ls -l"
```

Since Text uses popen(3) to run the passed Command, the Command option may contain arbitrarily complex commands, such as:

```
Text -Title "Finger and Fortune" \
    -Command "finger ; echo " ; fortune"
```

(This assumes, of course, that you have fortune installed on your system!) A most useful alias to add to .cshrc (if you use csh) would be:

```
alias Fortune Text -Title Fortune -Command fortune
```

The File default is useful for those cases in which a file already exists. File uses the specified file rather than stdin or a command. This could just as easily be accomplished by cat'ing the file to Text, but using File simply looks better. If Title is not explicitely specified, which-

ever of File or Command that are specified is used

for the Title, or "Text" if neither are specified.

The File and Command defaults can be ambiguous. When either is specified, the standard input is ignored. But if both are specified, which one should be used? I arbitrarily chose File to take precedence, so in:

Text -File /etc/ttys -Command "ls -lR"

the file /etc/ttys is output to the window, and the Command parameter is ignored. Command and File should not be written into the defaults database, because that would cause the Command or File to be used every time the program is run, unless specifically overridden on the command-line. If you do need to override one of them at the command line, just specify the default without a value.

The usage of the Burst and Close defaults may require explanation. Burst is used to tell Text whether to display the file after all input is read, or incrementally. For small files, or quick commands, all at once (-Burst 1) works well. But, if the command producing output is slow, or if the output is long, then the user should get

some indication of what is happening. With Burst off, the window is displayed immediately, and the user can see the output as it occurs. Close is useful along with Burst. Close indicates that the Panel should be closed if the input is empty. This is useful for cases where the user should not see a panel at all, if its an empty panel. Using Close and Burst together means that if the input is empty, the user never sees the window. Using Close with Burst off, though, means that the window will appear for a time, and then disappear.

The Edit default allows a user to edit the displayed text, and outputs the modified text to the standard output when finished. Save is a more powerful form of Edit. Save can be used to specify a file to save the output to. If Save is specified, but without

a filename, then the behaviour is determined by other parameters. In this case, if File is used for input, then the output is saved to File, with a backup being made (by appending ~ to the filename) depending on the Backup flag. If the backup cannot be made, an Alert Panel is displayed. If Command is specified, then a file is saved used which has a name based on the name of the executed command. If the filename resulting from either Command or Save already exists, and Overwrite is non-zero, the file is overwritten without question, otherwise an Alert Panel is displayed.

If no filename is specified, or the user so chooses in one of the possible Alert Panels, a Save Panel will be displayed, allowing the user to specify a destination for the file. With this in mind, the command:

cat /etc/passwd | Text -Save

will allow the user to edit the /etc/passwd file, and then will bring up a SavePanel to prompt for somewhere to save it to. More powerful users (like root) could do:

Text -File /etc/passwd -Save -Backup 1

To edit it in place, with a backup copy created as /etc/passwd~. A simple editor could be specified in your .cshrc file by

alias QEdit Text -Save -Backup 1 -File

with font, fontsize, and window size to taste. With this alias, you could then type

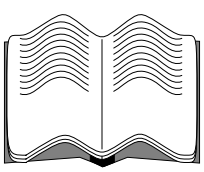
QEdit /etc/passwd -NXHost next-5

to edit the passwd file with the window going to next-5.

Though Text does not have a main menu, some command keys are useable. I've implemented the q, c, x, v, a, w, m, s, and p keys, with the same meanings as most other applications give them. q (Quit), w (Close) and s (Save) all have the same meaning, quitting the application and saving if applicable. The Print panel, unfortunately, comes up in a weird position, but that's OK (I guess), since I cannot change it, anyhow. I believe that this is because I did not use the correct routines when in the run loop of the application - I'm sure that run is supposed to do more than I have it do! There may also be other problems, though I've not run across any, yet.

The Text program can be downloaded from j.cc.purdue.edu and cs.orst.edu, along with full source and this article. If you could, I would recommend retrieving the version which is known to work, and has all connections and the icon intact. I cannot be sure that I described the procedure throughly enough that the results will be the same for everyone. If there are problems . . . Problems, complaints, friendly conversation, money, NeXTs to:

scott hess/NeXT Campus Consultant/Gustavus Adolphus College,St Peter, Mn 56082/scott@gacvax1.bitnet



Reviews, Rumors & Stuff

A Myopic Eye

into the NeXT Community

by
Erica J. Liebman
erica@kong.gatech.edu

Top Draw

Well, I just finished off another month of playing with Top Draw off and on. I discovered some neat things this month. First and most useful is *Arbitrary Rotation.* This is simply terrific. In TopDraw you can rotate any graphical object. Do you need an ellipse with the long axis at an angle? Just draw it normally and then turn it as you need to. Grouped Objects can be rotated. So can imported pictures.

Second is the **Grouping** mechanism. TopDraw allows you to group ala Draw and several other commercial packages -- however, these groups maintain their relationships especially robustly through rotation, translation and scaling. I was very impressed.

Third is **Nice Lettering**. Letters can be shaded, gradiated and so forth. It is so easy to create drop-shadow lettering in about 3 steps (although I wish there had been a "macro" associated with this function because I used it a lot). I created a paen to Whimsy, my NeXT in the space of five minutes. Very nice. I've included the result on the next page. (You may have noticed that I have figured out now how to import topDraw into Frame -- I "exported" from topDraw into encapsulated postscript format and then did a standard import into Frame.)

Word Perfect & Ashton Tate

I got sneak previews of two products from WordPerfect and Ashton Tate. Both look terrific. WordPerfect is coming out with a middle-of-the-road word processor with !outline! capabilities (thank heavens) and most of the other features from the PC version, but in a very NeXT application. Ashton Tate similarly is demoing a simply gorgeous spreadsheet app. This baby is packed with features : inspectors, real-time 3-D graph rotation and so forth. Don't look for split-views (like Excel) yet because they seem to be waiting on the 2.0 public split-view browsers. Both were primarily built from the ground up rather than being ported from other platforms. These are going to be two great products for selling more of the cubes.

Reviews

Terrence Talbot has officially joined the BuzzNUG team as our head reviewer. We will have a three-stage reviewing system. All product announcements and press releases go into Market View and New this Month. I get first shot at new products for "quick looks" within a month of receipt. Products are then shipped off to Terrence for more in-depth coverate to be printed within two to three months of receipt. We are hoping to do comparison reviews as well as single-product ones.

Diagram

I've been beta-testing Diagram for Lighthouse Designs and have been very happy with it. Diagram allows users to create those infamous "wiring" organizational charts, annotate maps or program-structures and seems a perfect aid to the documentation manual writer.

Diagram users perform their work by dragging items from pallettes into an expandable and multi-page drawing space, connecting and annotating them. You can import postscript and tiff pictures into your documents and add them to pallets. Compound objects can be created through grouping and put on your pallet as well. This gives you a lot of power for creating custom "clip-art" sorts of pallets for all types of end-users.

Learning to use the program goes very quickly. Controls are easily found and pretty intuitve. I had a few problems with understanding some options in my version, but these have all been fixed (as I saw at a recent demo) and new users should be able to get started with doing their (real) work within an hour of firing up the program.

Diagram does not attempt to cover a lot of scope : its purpose is to create diagrams and annotation and no more. Within these bounds it does its job well and efficiently.

I would have included some pictures I created, but my Beta version timed out recently (just before a demo, mind you), and I haven't figured out what day to set my system clock back to, to get it to run again. Lighthouse promises me a more recent Beta-version.

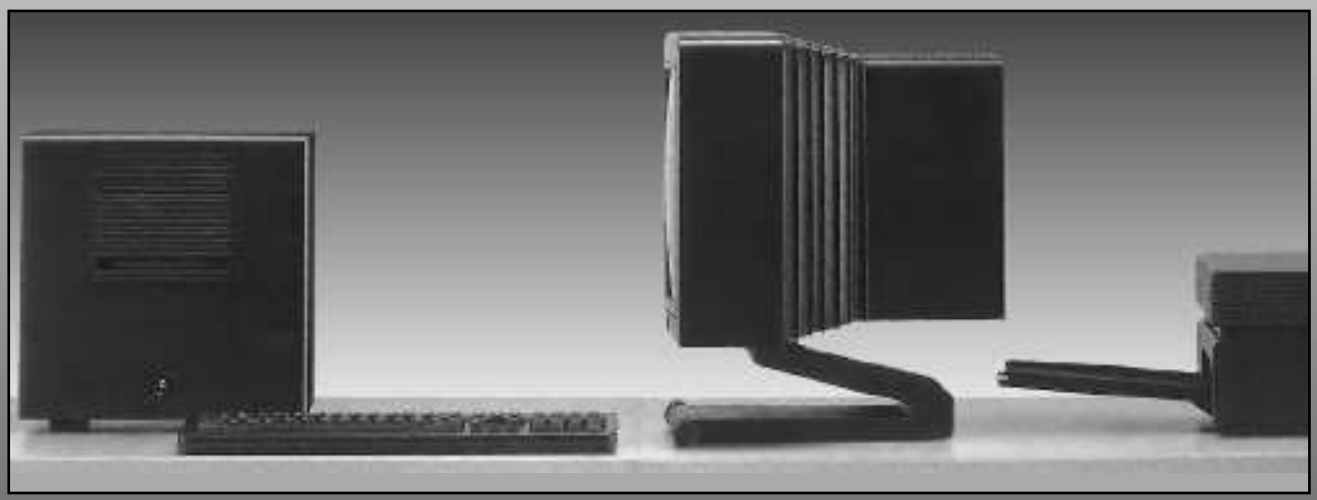
Don't forget. Lighthouse is still selling their (and my) first Compilation Disk. I've got a bit more than 1/3rd of a disk of a second Compilation Disk ready, so look forward to another compilation by around December/January. Lighthouse is also working on Exploder, a CASE tool with support for "persistant objects". More later.

Third Party Catalog & More

The new third-party catalog should be available within a week or two. NeXT sent out an interesting mailing with a pair of developer-support documents (whose titles escape me, I left them at home today. Ooops) and an announcement that while the 68040 machine will **require** version 2.0 system software, the 68030 machine will not, being able to run both 1.0 and 2.0 system software.

One would gather (unsurprisingly) that 2.0 will be co-delivered with the 68040 upgrades.

WHIMSY



a fanciful or fantastic device, object, or creation esp. in writing or art (syn caprice, boutade, conceit, crotchet, fancy, freak, humor, megrim, vagary, whim. rel idea, disposition, inclination, thought; dream, fantasy, vision)

Market View

New This Month

THIS SECTION IS PROVIDED AS A COURTESY TO OUR READERS AND THIRD PARTY FRIENDS. BuzzNUG IS NOT RESPONSIBLE FOR THE CONTENT OF THESE OR ACCURACY OF INFORMATION PRESENTED IN THIS SECTION

Robert Lin of Tao sends us : "Interested in a fax solution for your NeXT? **Objective Software Engineering** is about to release **Nexus Fax**, a 9600 bps Group III fax modem plus a 2400 bps Hayes compatible modem. The software allows one to fax any document that can be printed, as well as receive fax in the background. Availability is expected in fourth quarter of 1990. Please call (604) 261-0186, or fax to (604) 261-5324 for more information."

Portal is offering **mail-only connections** : $15 signup, $15 per month, including 4 hours of connect time per month. Extra hours cost $3.95/hour. All UUCP accounts include a Portal account. Polling for 150% of telephone cost.No usenet.No support.For a limited time only. For more information, please contact: Portal Communications Company, 10385 Cherry Tree Lane, Cupertino, CA 95014,408.973.9111 (voice),408.725.1580 (fax), cs@-cup.portal.com (email)

Baran's Tech Letter/ P.O. Box 876/Sandpoint, ID/ 83864-0876/Nicholas Baran (208) 265-5286. I'll just give you the straight quote on this one : "*In response to growing demand for an independent publication covering the NeXT(tm) Computer, former BYTE West Coast bureau chief, Nicholas Baran, announced his plans today to publish a monthly newsletter called Baran's Tech Letter. The first issue is slated for publication in September, 1990.*

Aimed at both end users and developers, Baran's Tech Letter will provide news and analysis of new products and technological developments affecting the NeXT environment. "Baran's Tech Letter will be a much needed source of information about third party software for our NeXT Computer customers," said NeXT's manager of marketing communications, Karen Sipprell. In addition to third party software, Baran's Tech Letter will cover topics such as:

o Postscript Level 2; how will it affect Display Postscript in future releases of the NeXT system software?

o Motorola's new 96002 digital signal processor; it's the successor to the DSP chip in the NeXT Computer; how is it different and is it likely

o The competition; are other vendors catching up to the unique capabilities of Interface Builder and NeXTStep? Is IBM really committed to NeXTStep?

"The NeXT user community is steadily growing and users need a professional, independent publication that covers the machine," said Baran. "I don't think the trade press has been providing adequate coverage of the NeXT Computer or third party products," he added. As an independent publication, Baran's Tech Letter will carry no advertising. Each issue will be 8 to 16 pages long; the annual subscription price for 12 issues will be $125."

Digital Instrumentation Technology Inc (505)662-1459 announced a planned August release of an upgraded software package for their **Cube Floppy 1.4**. This is a free upgrade : if you already own a drive, it will be sent to you on (unsurprisingly) a floppy disk. Contact Liz Shrum of DIT for further information.

Product Listings

I've received literature from NeXT or Third Party Developers on the following products. No warranty, express or implied, is given. The quotes are mostly the developers' and may not reflect reality. Please send up-to-date info and review copies for new products.

•QuintProcessor - 5 27-MHz DSP56001s on a Board, Ariel Corp, 201-249-2900.

•Objective DB Toolket - 30 classes to links NextStep to Sybase. Professional Software, Inc./Lakeside Office Park,/599 North Avenue - Door 7/Wakefield, MA 01880/ (617)246-2425

•uni-REXX and uni-XEDIT - Unix version of IBM's mainframe/procedural language and general purpose editor./The Workstation Group, wrk/grp/6300 N. River Road/Rosemont, IL 60018/(708) 696-4800

•Digital Instrumentation Technology Inc (505)662-1459 is shipping beta copies of **Cube Floppy 1.4**, a 3.5" floppy disk drive connecting to the SCSI port, reading and writing MS-DOS(720K and 1.44MB disks), UNIX and Macintosh (1.4MB disks) file formats.

• WeDesign Inc (415)-479-1105 sent a flyer on **TheLibrary**., an on-line information system with Objective C references and authoring tools.

• Pacific Micro (415)948-6200 is shipping the **PM1.44** and **PM HDE** external 3.5" floppy drive and hard disk enclosure. Special order line is 1-800-628-DISK.

• BYTE's **BIX** service has a special NeXT interest group. Contact 1-800-227-2983 (BIX Customer Service) if interested. Please mention both BuzzNUG and Dave Andrews (of Byte) as contact names.

•Communicae - Active Systems 1-617-576-2000 "high performance communications package. VT240 emulation" *SHIPPING*

•Wingz Informix Software, Inc. 1-913-599-7100 "graphic spreadsheet featuring advanced charting, desktop presentation capabilities, and HyperScript" *SHIPPING*

•Scan 300/GS Abaton 1-415-683-2226 "300 dpi flatbed scanner with TIFF compatility"

•DM-N Digital Microphone Ariel Corporation 1-201-249-2900 "software-selectable sample rates from 88.2

kHz to 5.5 kHz per channel" *SHIPPING*

• BUG-56 : DSP debugger, Ariel Corporation 1-408-982-0400/1-201-249-2900, *SHIPPING*

•DaynaFILE Dayna Communications, Inc. 1-801-531-0600 "external, SCSI floppy disk drive to write to standard UNIX-formatted diskettes, as well as MS-DOS formats" *SHIPPING*

•Smart Art Emerald City Software, Inc. 1-800-223-0417 "50 text and graphics effects and easily customized in any NeXT word processor, desktop presentation, or page layout program *SHIPPING, -- I GOT A THIRD PARTY LISTING OF THIS FROM NeXT WHICH SAYS THIS IS NOW OFFERED BY ADOBE! (415)962-2045*

•FrameMaker 2.0 Frame Technology Corporation 1-408-433-3311 "powerful, cost-effective workstation publishing software" *SHIPPING, RECOMMENDED FOR DESKTOP PUBLISHING, PAGE LAYOUT FOR IN-HOUSE PUBLICATIONS, BROCHURES.*

•Artisan Media Logic Incorporated 1-213-453-7744 "high-resolution paint and image processing system"

•TopDraw Media Logic Incorporated 1-213-453-7744 "complete and advanced page-based graphics software" *SHIPPING*

• HSD incorporated US Scan-X 1600/600 415-964-1400. Scanners for “line art and grayscale” *SHIPPING*

•TextArt Stone Design Corporation 1-505-345-4800 "array of tools that allow immediate creation of outstanding PostScript images" *SHIPPING, RECOMMENDED FOR “SPLASH” APPEAL, PROGRAMMING WITH ICONS.*

•Encapsulated PostScript ClickArt T/Maker Company 1-415-962-0195 "combines ClickArt EPS portfolios into a collection of high-quality Encapsulated PostScript (EPS) artwork" *SHIPPING*

•Public Domain Disk #1 - Lighthouse Design 1-800-FOOBAR9 "Public Domain Software & More" (I'm in on this one. Buy it. Please!) *SHIPPING*

•Scematic Entry - Lighthouse Design 1-800-FOOBAR9 "CAD Tool for designing electrical circuit schematics"

•Media Station - Imagine Inc 1-313-434-1970 "archival, retrieval and processing of multi-media information" *SHIPPING*

•Fortran 77 -- Absoft 1-313-853-0050 "Objective Fortran-77"

•DisplayTalk - Emerald City Software - 1-800-223-0417 "Complete development environment for Display PostScript programming" *SHIPPING-- I GOT A THIRD PARTY LISTING OF THIS FROM NeXT WHICH SAYS THIS IS NOW OFFERED BY ADOBE! (415)962-2045*

• *SmartArt*, Adobe Systems Inc. 415-962-2045 “Graphics and headline type effects using Display PostScript”.

•Video Monitor and Projector Interfaces Extron Electronics 1-800-633-9876 "offers three video monitor and projector interfaces"

•Digital Ears Metaresearch, Inc. 1-503-238-5728 ""allows entering and recording compact disc-quality sounds" *SHIPPING, REVIEWED IN ISSUE 4*

•Digital Eye Metaresearch, Incorporated 1-503-238-5728 "allows entering and recording NTSC video images"

•NVT High Density Video Drive New Vision Technologies, Inc. 1-415-285-8744 ""video playback device for interactive multi-media applications"

•JETSTREAM Tape Backup System Personal Computer Peripherals Corporation 1-813-884-3092 "high performance tape backup system"

•A/D64x Analog/Digital Interface Singular Solutions 1-818-792-9567 "a low-cost platform for sound recording, experimentation, and analysis"

•Who's Calling Adamation, Inc. 1-415-452-5252 "lets sales & business professionals keep track of phone calls and other client information" *SHIPPING, THEIR BROCHURE IS REALLY COOL, THIS IS ONE I WANT TO REVIEW.*

•GEMS (Generalized Equilibrium Modeling System) Data Transforms, Inc. 1-303-832-1501 "a flexible way to model economic systems"

•InDia (Influence Diagram Processor) Data Transforms, Inc. 1-303-832-1501 "graphical application for representing complex decision-making"

•Knowledge Retrieval System (KRS) KnowledgeSet, Corporation 1-415-968-9888 "rapidly searches and retrieves information from large databases of text and graphics"

•OMEN III Microstat Development Corporation 1-604-228-1612 "stock quotation and financial system"

•TACTICIAN Plus SouthWind Software, Inc. 1-316-636-5100 "multi-user spreadsheet that supports high-level functions and adds built-in presentation graphics" *I’VE SEEN A DEMO VERSION, BASIC SPREADSHEET FUNCTIONALITY -- IT’S IN THERE.*

•Adobe Illustrator Adobe Systems Incorporated 1-415-961-4400 "graphic design and illustration program for generating high-quality artwork"

•Adobe Type Library Adobe Systems Incorporated 1-415-961-4400 "offers more than 500 different typefaces" *I WANT A COPY OF THIS!*

•Flash Graphics Flash Graphics 1-415-331-7700 "extensive charting, illustration, and text functions in a graphics package for screen, slide and paper presentations"

•InterFax 24/96N Abaton 1-415-683-2226 "combines a 9600 bps Group 3 fax modem with a 2400 bps MNP 5, Hayes-compatible data modem"

•GatorBox Cayman Systems, Inc. 1-617-494-1999 "LocalTalk to Ethernet gateway that translates the Network File System (NFS) protocol into Apple Filing Protocol (AFP)" *SHIPPING*

•MacLinkPlus/PC DataViz Inc. 1-203-268-0030 "kit for transferring and translating files between NeXT and

Macintosh environments" *SHIPPING*

•Ethernet PhoneNET, Sound and Interpersonal Communications Farallon Computing, Inc. 1-415-849-2331 "used to build LANs over standard telephone cables"

•Etherport NL Kinetics 1-415-947-0998 "allows the NeXT computer to connect directly to standard twisted-pair Ethernet networks"

•INFORMIX-TURBO Informix Software, Inc. 1-415-926-6300 "database engine for on-line transaction processing (OLTP)"

•INGRES Relational Database Management System Relational Technology, Inc. 1-800-4-INGRES "SQL database engine provides on-line transaction processing (OLTP) in single- or multi- CPU and distributed environments"

•DAN - The Data Analyzer Triakis Inc 1-505-672-3180 "data analysis package for reducing data and generating presentation-quality plots"

•Math++ - Triakis Inc 1-505-672-3180 "C-language math library. Approx 100 math functions"

•Dreams - Innovated Data Design 1-415-680-6818 "Frm the makers of MacDraft, drawing and drafting tools"

•Cross Assember/Simulator Programs - Motorola 1-512-891-2030 "for the 56000 and 96000"

•Fortran, C and Pascal Compilers - OASYS 1-617-890-7889 *SHIPPING*

SIMMS

The SIMM information below was posted by William Smith on the public Comp.Sys.Next newsgroup. It is reproduced without direct permission.

PSI 2005 Hamilton Ave., #220, San Jose, CA,95125 Ph. 408-559-8544,Price: $59.50 per simm ,800-622-1722, Lifetime Guarantee

Third Wave Computing, Inc. 1826-B Kramer Lane Austin, TX 78758, ,Ph. 800-284-0486. They take PO's,Price: ?, 512-832-8282

Memory international,Ph. 714-588-0488. Price: $68.00

Turbo Technologies, Inc.,Ph. 800-542-7466 Lifetime warranty,They take PO's.Price: ?

South Coast Electronics. Ph. 800-289-8801,They take PO's 10920 Wilshire Blvd.Suite 110, LA,calif. 90024,Lifetime warranty,Price: ?

Newer Technology 1117 S. Rock Rd. Suite 4 Wichita, KS 67207,Price: ? Ph. 800-678-3726, 316-685-4904

Micro Electronic Technologies, inc. 35 South St., Hopkinton, MA 01748,Price: ?Ph. 800-766-SIMM, 508-435-9057

Peripheral Outlet 314 S. Broadway Ada, OK 74820, Lifetime guarantee, Ph. 800-332-6581, they take PO's, Price: $62.00, 405-332-6581

ETC 5426 Beaumont Center Blvd. Suite 340 Tampa,Fla 33634, They take PO's, Ph. 800-882-2863, Price: ?, 813-884-2863

Shecom computers 22755 Savi Ranch Pkwy G, Yorba Linda, CA 92686, They take PO's Ph. 800-366-4433, Price: $59.00,3 year warranty, 714-637-4800

Chip Merchant 9285 Chesapeake Dr., Suite L, San Diego, CA 92123, 5 year guarantee Ph. 800-426-6375, no credit cards,Price: $60.00,619-268-4774

Memory Plus, Ph. 800-388-PLUS, Price: $62.00

Megabyte Memory Products 737 Pearl St., Suite #208B La Jolla,CA 92037, 5 year warranty, Ph. 800-748-5766, Price: $62.50

Technology Works 4030 Braker Lane West, Austin, TX 78759,They take PO's, Ph. 800-688-7466, Price: ?, 512-794-8533

Stratum Technologies inc. 12191 Technology Blvd., Austin, TX 78727,Lifetime warranty, Ph. 800-533-1744, 30 day trial period (?),they take PO's, Price: ?

Delta Research Labs 26072 Merit Circle, Suite 119 Laguna Hills, CA 92653,warranty 60ns,70ns,80ns,100ns,Ph. 800-999-1593, Price:?, 714-367-0344

Computer Care, Ph 800-950-2273/612-371-0061, Price : $99, Ford Center Suite 1180, 420 N. Fifth St, Minneapolis, MN 55401.

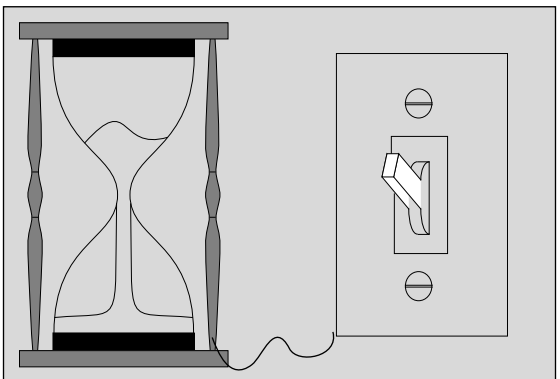
Impediment, Ph. 617-837-8877 warranty 6years

Price: (they will generally beat the best price you can come up with), I think they take PO's

ClearPoint Computers, Ph. 800-253-2778,Price:?

Associated Technologies Marketing Ph. 214-248-0700

Memory Plus sells *4 MB* SIMMs (low profile, 80ns) for $299.00,**Memory Internationa**l sells them for $298.00.



Unix Electronic Timing Circuitry

User Groups

compiled by
Conrad Geiger
Conrad_Geiger@Next.Com

CANADA

British Columbia

Vancouver NeXT Group, Lionel Tolan, chairman, Computing Services, Simon Fraser University, Vancouver, B.C., Canada, V5A 1S6, phone - (604) 291-4702, email - lionel_tolan@cc.sfu.ca

"We meet every last Wednesday of the month, 6:30 pm at the Vancouver NeXT office: 230-1333 Johnston St., Granville, Island, Vancouver, B.C."

Newsletter: NeXTVieW, quarterly, editors Tom Poiker and Shirley Chan, Address: Dr. T.K.Poiker, Department of Geography, Simon FraserUniversity, Burnaby, Canada V5A 1S6, phone - (604)291-4515, email - USERTONI@cc.sfu.ca or poiker@whistler.sfu.ca

NeXT Tabloid: "Tao" (published monthly), editor is Robert Lin, Objective Software, 1701W 64th Ave, Vancouver, B.C., Canada V6P 2P3, phone - (604 261-0186, email - rlin@cs.ubc.ca

Quebec

Montreal Next Section of Club MacIntosh, Robert Paulhus - president, Club MacIntosh of Montreal - NeXT section, 2250 Guy Street room 303, Montreal, Quebec H3H2N2, phone - (514)939-0382, email - paulhus@calvin.cs.mcgill.ca

Ottawa NeXT User Group, Hugo DeRosier - president, 19 du Muguet, Hull, Quebec, Canada J9A 9Z7, email - hugo@csi2.uofo.edu

Toronto

Toronto NeXT User Group (more next issue)

JAPAN

Tokyo

Japan NeXT User Society - Katsuhiro Ohashi, 4-4-2, Ebara, Shinagawa-ku, Tokyo, 142 JAPAN,NeXus-office@etl.go.jp, NIFTY : NAC01417(Kishimoto) or NCB00436(Shioya)

Kon-nichiwa

We are very happy to formally announce about the inauguration of our user group. The following is the latest information about our user group.
We are going to start up the NeXT User Society (NeXus). This is the first NeXT User Group in Japan and Asia. The first meeting will be held in Tokyo on July 25th from 7:00 PM. NeXT users who can attend the meeting, please contact us at the following address.
Liaison: Internet: NeXus-office@etl.go.jp, Nikkei-MIX:-shioya, NIFTY: NAC01417(Kishimoto) or NCB00436(Shioya), Facsimile:+81-03-351-0880 (Shioya, SRA, Inc.), mail:Katsuhiro Ohashi,4-4-2, Ebara, Shinagawa-ku, Tokyo, 142 JAPAN
Meetings: Monthly meetings will be held on the 4th Wednesday of each month, from 7PM to 9PM. The first meeting will be held on July 25th at the showroom of Canon Software Inc (a Japanese distributor of NeXT Computers). We will discuss regulations and organization of NeXus, have a Q&A corner, and review and demonstrate some software.
Newsletter: We are going to publish a monthly NeXus Memo and quarterly Newsletter. The NeXus Memo will be distributed at or after every meeting. The first Newsletter is expected to be published this fall.
Activities: We are also planning to have technical information exchanges, technical support, and distribution of Free Software and Shareware.
Mitsuhiro Kishimoto/Fujitsu Laboratories Ltd./Artificial Intelligence Laboratory/Parallel Processing section/ kiss@flab.fujitsu.co.jp

UNITED STATES

National Groups:

Medical: NeXTMed - National NeXT Medical Users Group, Bill Barker and Jim Brinkley, Biological Structure, SM - 20, University of Washington, Seattle, WA 98195, email - NeXTMed-request@ulnar.biostr.washington.edu

Also : Conrad Geiger, 4130 Bagley Ave. North, Seattle, Washington 98103, phone - (415)780-2771

Music: (more info next issue)

Programmers: NeXT Programmers SIG

To join: email to "next-prog-request@cpac.washington.edu"

United States
States:

Arizona

Phoenix NeXT User Group, Gary Frederick - president, 20826 N 16th Ave, Phoenix, Arizona 85027-3531, phone - (602)869-0316, email - frederic@cimnext.cim.eas.asu.edu

Also : Chet Kapoor, 7150 E. Camelback Road, Suite 300, Scottsdale, Arizona 85251, phone - (602)423-7080

Also: Jim Ames, Arizona State University, CIM Systems Research Center, Tempe, AZ 85287-5106

Tucson NeXT User Group, Robert W. Layhe, CCIT/ User Support, University of Arizona, Tucson, AZ 85721, email address - layhe@rcnext1.rc.arizona.edu

California

BaNG (Bay Area NeXT User Group), P.O. Box 8858, Stanford, CA 94309, BaNG-request@meta-x.stanford.edu, Eric Ly, P.O. Box 12318, Stanford, CA, 94309, phone - (415)780-2877, email - dayglow@-portia.stanford.edu.

Also : Joe Barello, 1505 Grand Avenue Piedmont, California 94611, phone - (415)652-0769, email - joeba@lll-lcc.llnl.gov,

Also : Chris Overton, P.O. Box 2628, Stanford, CA 94309, email - louiex2@portia.stanford.edu

"Meetings are every third Wednesday of the month at 7 PM on the Stanford University campus."

Santa Barbara NeXT User Group, Amir Gharaat - president, email - erone%pumpkin@hub.ucsb.edu

Nuggets (Cal State - Los Angeles), Gary Novak - president, Department of Geology, Cal State, 5151 State University Drive, Los Angeles, Ca. 90032, phone - (213)343-2400

JPL/Caltech NeXT User Group, Leo Blume - president, adr: JPL, MS-510-202, 4800 Oak Grove Dr., Pasadena, Ca. 91109, phone - (818)397-9521, email - leo@emerald.jpl.nasa.gov

UC Riverside (California) NeXT User Group, Paul Lowe, president, phone - (714)787-3883, email - plowe@ucrac1.ucr.edu

SNuG - San Diego NeXT User Group, Nicholas MacConnell, 1135 Stratford Ct., Del Mar, Ca. 92014, phone (619)481-7535 or (619)565-9738

Also: David Rivas, email - david@ece.ucsd.edu

Colorado

rmNUG (Rocky Mountain NeXT Users Group), Dave Hieb, Chairman, 4521 Wellington Rd, Boulder, Colorado 80301, phone - (303)530-2560, email - davehieb@boulder.colorado.edu

Also: Brad Green, Co-chairman, 4600 South Ulster Street, Suite 700, Denver, CO 80237, email - green_-bk@cubldr.colorado.edu

The Rocky Mountain NeXT Users' Group (rmNUG) provides monthly meetings in the Boulder/Denver area for all NeXT enthusiasts/users living in the Front Range region of Colorado.

David Hieb (davehieb@boulder.Colorado.EDU, (303)492-5720) is the leader of rmNUG and is a System Administrator for the University of Colorado, Boulder.

Brad Green (Brad_Green@NeXT.COM, (303)786-9371) is the NeXT Campus Consultant for the University and provides support and leadership for rmNUG as well.

We are successfully providing an environment for the NeXT users in Colorado to share, learn and profit from the collective expertise of the group.

District of Columbia (Northern Virginia, Southern Maryland)

NeXT Special Interest Group (SIG of BCS) - Washington, D.C., Hugh O'Neill, President, P.O. Box 39036, Washington, D.C. 20016,

Also :Joel McClung, email: joel@next.com, (703)938-NeXT, Mailing address for newsletter, NeXT, Inc., 8300 Boone Blvd., Suite 558, Vienna, VA 22182, Attn: Fillipe Fuster

"Meet 2nd Wednesday of month at Building 12A, Room B-51 of NIH Complex in Bethesda, Md. Meetings are at 7:30 and run anywhere from 1 to 3 hours."

Georgia

BUZZNUG (Georgia Tech NeXT User Group), Erica Liebman - editor of NeXT Users Journal, 1150 Collier Road NW Apt L-12, Atlanta, GA 30318, email - erica%kong@gatech.edu, phone -(404)352-5551

Illinois

Chicago NeXT User Group, Mark Henderson, Building 203 Room C-246, Advanced Computing Research Facility, 9700 South Cass Avenue, Argonne National Laboratory , Argonne, Illinois 60439, email - henderson@mcs.anl.gov, phone - (708) 972-5963

NU NeXT Users Group - (Northwestern University),

Bill Parod - president, Northwestern University, 627 Dartmouth Place, Evanston, IL 60208, email - parod@baris.acns.nwu.edu, phone - (708)491-5368

Massachusetts

BCS - Boston Computer Society, Dan Lavin - president, NeXT User Group, Boston Computer Society, One Center Plaza, Boston, Ma. 02108, phone - (617)969-6555

Minnesota

Minnesota NeXT User Group, Mike Tie - President/ Treasurer, Math/CS Department, Carleton College, One North College Street, Northfield, MN 55057, phone -(507)663-4067, email - mtie@carleton.edu

"We meet on the second Tuesday of every month. We hold the meeting at a different location every month, so I send a mailing to all our members with the information about location and the agenda. People who are interesting in attending our meetings should get in touch with me."

Missouri

St. Louis NeXT User Group, John Bartley, Deloitte & Touche, One City Centre, St. Louis, Missouri 63101, phone - (314)343-4996

New Mexico

Albuquerque NeXT User Group, Jeff Jortner, Sandia National Laboratories, Division 1424, PO Box 5800, Albuquerque, NM 87185, email - jnjortn@cs.sandia.gov

Los Alamos NeXT Users Group, Dwight Barrus, Chairman, Group C-6, MS-B272, Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM 87545, phone (505)667-8870, email - dmb@lanl.gov

Also: Joe Kleczka, Co-Chairman, email - jhk@lanl.gov, phone -(505)667-4584 (page number)

New York

New York City NeXT User Group, Tim Reed, 7 Dey St. Suite 711, New York, New York 10007, phone (212)227-6767

Ohio

Columbus NeXT User Group (Ohio), Chuck Dyer, Sr. Systems Analyst, The Ohio State University, 1971 Neil Ave-IRCC, Columbus, OH 43210, phone - (614)292-4843, email - dyer-c@osu-20.ircc.ohio-state.edu

Oregon

OSU - Oregon State University NeXT User Group, Tom Leach - president, Ocean Admin 104, College of Oceanography, Oregon State University, Corvallis, Oregon, email - leach@satchmo.oce.orst.edu

Meet 2nd Thursday of month at 2:30pm in Milne Computer Center - Faculty Development Lab.

Portland NeXT User Group (more next issue)

Pennsylvania

Allegheny College - IMUG (Interface Builder User Group), Dr. Joel Smith, Allegheny College, Allegheny, Pa., email - js01@music.alleg.edu

Texas

DFW NeXT User Group (Dallas/ Ft. Worth), Dirk Hardy, president, Hofbauer Information Systems, 5080 Spectrum Drive, Suite 912W (Lock Box 21), Dallas, TX 75248, phone - (214) 385-2991

Also : Charlie Lindahl, Automation and Robotics Research Institute, University of Texas at Arlington, 7300 Jack Newell Blvd. S., Ft. Worth, TX 76118, phone - (214)284-6122, email - lindahl@evax.arl.utexas.edu

"We meet the third Thursday of every month at theNeXT office in Las Colinas. Our meetings last from 7 until 9 (official hours), and there are usually informal discussions until 10PM or so."

hAng - Houston Area Next Group, John R. Glover - president, E.E. Department, University of Houston, Houston, TX 77204-4793, email - glover@uh.edu, phone - (713)749-1820

"We meet first Wednesday of each month at the Engieering College, UH, from 6 PM until around 8:30 PM."

Washington

University of Washington NeXT User Group,Corey Satten - president, University of Washington, 156D Academic Computer Center, HG-45, 3737 Brooklyn Avenue, NE, Seattle, WA 98105, email - corey@cac.washington.edu, phone - (206)543-5611

"We meet the third Wednesday of each month at 3:30PM in Parrington Hall, room 306 on the University of Washington campus."

Washington State University NeXT User Group, Joe Gerkman - president, N.E. 545 Kamiaken St. #8, Pullman, WA 99163, phone- (509) 334-9594, email -

gerkman@pcc-next.csc.wsu.edu -OR- gerkman@-wsuvm1.bitnet

BANG

BaNG is an organization dedicated to providing an open forum for end-users, corporate developers, academic developers, and NeXT, Inc. in which to exchange ideas about the future of computing.It is interested in serving NeXT users all over the world.

For BaNG to become a genuine success, it needs to become financially self-sufficient.This is where membership in BaNG can help.In the past, BaNG has been able to provide high-quality meetings, and with your support and our increasing experience, BaNG will continue to do improve as our services to you expand.

Meetings will feature demonstrations of significant new applications, many of them premier public presentations.Members will have the opportunity to ask questions and be heard by the decision makers at BaNG's open forums.

Help make BaNG the best user group in the world.

As a registered member of BaNG, your benefits include all of thefollowing:

o The handsome multicolor BaNG T-shirt.

o A one-year subscription to the quarterly BaNG newsletter. One issue each year will include a membership directory, with information about users, their interests and projects. Here's your chance to gain exposure and to contact others with similar interests.

o Access to the BaNG archives, with public-domain software exchanges at our monthly meetings.BaNG will be bringing you the best of the NeXT archive sites around the country, including documents such as the NeXT Users Journal and NextAnswers, as well as software available only through BaNG.This archive will be made available to those without Internet access.

o At meetings, BaNG will be auctioning off a copy or two of an application being demonstrated.Here is your chance to obtain the latest and greatest new products, maybe even below retail!

o Special deals and offers available only to BaNG members. BaNG will get special price breaks on services, literature and software from time to time, and these discounts will be passed on to registered members.Already, BaNG has special, low rates for members wanting to gain network access to the electronic network through Portal Communications.This is only a sample of the possibilities.

o Finally, the knowledge that you are helping to support the West Coast's (and soon to be the world's) largest NeXT user group.

For all these benefits and more, BaNG is asking for a $30 annualmembership fee ($20 for students).

Please contact BaNG to get a Membership application & questionnaire.

rmNUG - Rocky Mountain NeXT User Group

The third meeting of the Rocky Mountain NeXT Users Group (rmNUG) was a great success. We all enjoyed the chance to visit with other NeXT enthusiasts, try out some new hardware/software and hear a great presentation by our NeXT Campus Consultant, Brad Green. Brad's special feature focused on "Music on the NeXT". Brad had just recently attended a national computer music convention and was able to talk with leading computer music developers and pick up some new software for the NeXT. He demonstrated everything from synthesizers to voice trainers and even a complete mixing station. [Thanks to the Denver office of NeXT Inc. for supplying the meeting room!]

I will be presenting Mathematica as the special feature at our next rmNUG meeting which is scheduled for the 18th of July. My presentation will be taken from a entry-level paper that I have been writing for an upcoming BuzzNUG issue. This meeting is for those of you who are interested in Mathematica and are either somewhat intimidated or just haven't taken the time to look at it closely.

I was very pleased to see several new faces at the meeting and I hope that each one of you enjoyed yourself. Make sure to bring your floptical disks to each meeting so that you can get any new software that you are interested in. Also bring any images/prints to our next meeting that you want scanned (Our friends at NeXT will have the new NeXT scanner on hand).

If you have any input, comments or questions about rmNUG, please call me (303)492-4316) or email me (davehieb@boulder.Colorado.EDU). This type of feedback is encouraged and necessary to any successful user group. As always, I am very interested in future special features for rmNUG. Be thinking of how your specialization or project might be of interest to the group. So far we have the following tentatively scheduled:

1)A developer from Mathematica presenting the development of Mathematica.

2)An employee from NeXT Inc. presenting the Inter-

face Builder.

3)A rmNUG member presenting Internet archive access and what is available for the NeXT.

4)A rmNUG consultant giving a Postscript tutorial.

5)A rmNUG businessman presenting Desktop Publishing on the NeXT.

Thanks for making this work!

David Hieb::davehieb@boulder.Colorado.EDU/ Brad Green::Brad_Green@NeXT.COM

Who's on NeXT?

Appologies given, Author Anonymous

"What's NeXT?"

"I don't know, you tell me what's next."
"Its a computer."
"I don't see no computer, just give me some fruit. How about one of those apples."
"That's one too."
"What's one too?"
"Counting, I think...one two three?"
"Why are you talking about spreadsheets?"
"I'm not talking about spreadsheets, I'm talking about NeXT."
"Next what?"
"Yes. They are very energy inefficient."

Buzz's Hint Corner

Coordinated by EJ Liebman

- Did you know that Frame supports Emacs style commands? -- *EJL*
- In Frame, control-click selects a column. You can then use the graphics in the "Draw" palette to shade and frame a column -- *EJL*
- In TopDraw, a Copy/Paste combination on Letters is just right for putting the copy in the right place for a drop-shadowed letter style. -- *EJL*
- In Frame, and several other applications, even though objects are grouped together they can still be modified and highlited individually -- *EJL*
- Erica, here's two tips for you :

Tip #1:

The loginwindow program allows the specification of two programs to run on login or logout. These are specified by the -LoginHook and -LogoutHook parameters respectively. These programs can almost any program which really doesn't need the Workspace Manager to run. They may also be shell scripts. The program is run as root, with the first parameter being the user being logged in or out.

Locally, I have a script which first does cat /etc/motd, and then executes a script in the logging-in user's directory. The script is /etc/WorkspaceLogin, and the user's version is ~/.workspaceLogin. This allows capabilities similar to the those supplied by the /etc/cshrc and ~/.cshrc files. The file in /etc/ may be used to do stuff you wish every user to do, while the file in the user's directory can do personalized things. I have the output from both scripts fed to Text (see article in this issue). Right now mine only does cat /etc/motd in the system file, and fortune in the ~scott/.workspaceLogin file, but other possibilities are to have the system files handle correctly logging the user to the wtmp file, or correctly removing the user from the utmp file when one of the command line programs gets messed up (Shell :-[).

Tip #2:

Been attempting to find a real use for the Draw program from the Demos directory? Use it to draw hierarchy diagrams like the NeXT documentation. Set the grid size (command-r) to 8 pixels, create a rectangle with sides 0 pixels wide, then put some text into it. Center the text with the Text menu, and make the text cell as wide as the rectangle. Group them together with command-g. Now you have a template to create more boxes.

To connect the boxes, use the "corner" tool with linewidth of 4 pixels. (The corner tool looks curvy in the Tools panel, but it isn't when you use it). Use the tool to make corners which are about 3 grid squares on a side. Then connect them to the boxes with lines, or however you want.

This works fairly well, though I've had problems getting things to print out well. It keeps wanting to put the page into Landscape mode :-) It is also a bother to have to ungroup boxes when you wish to edit the text within.

scott hess

scott@gacvax1.bitnet

NeXT Support Answers

Care of Doug Kieslar

Loginwindow PowerOff, Disabled power off power down (QA333)

Q: I set the PowerOffDisabled default of loginwindow to Yes to disable to the power key, but it doesn't seem to workWhen I hit the Power key, the Workspace Manager still gives me the power-off alert, and the machine will still power down if I confirm.

A: You need to change the default value of PowerOffDisabled for root, not yourself. Note that with this done, you will still get the power-off alert from WSM, and if you confirm, you will still be logged off.But, the machine won't power down; you'll instead find yourself at the login window.

Q: Assuming I get PowerOffDisabled working, how do I power down the machine?

A: The PowerOffDisabled default simply changes what the loginwindow does when the user tries to power down the machine by hitting the power key when either the login window or WSM is up.At other times, like when you are in the console or at the NeXT prompt, you can still use the power key to turn the machine off.

Application Icon Edit (QA 480)

Q: I think I've done everything right, but my application shows up with the icon of a text file (Edit's document icon). What's wrong?

A: Assuming the application is in the Workspace Manager's ApplicationPaths, and assuming that execute permission is granted, and assuming that a Find Applications was done, then what's most likely is the icon is either an incorrect size (not 48x48) or was saved without Alpha. The .tiff file should be 1292 bytes; a 48x48 icon without Alpha will be 686 bytes.

TIFF compatibility Draw rasterfile format Icon (QA 426)

Q: I'm trying to convert a TIFF file made from Draw into a rasterfile format (for, e.g., an SGI IRIS). All the TIFF converters I know of complain about an invalid TIFF format. What's going on?

A: NXWriteTIFF() writes out data and alpha as one strip, not two separate strips; alpha is assumed to follow the data. But, the TIFF standard says that in this case (samples per pixel of 2) there should be 2 StripOffsets, not 1. So the file is not considered valid.

A workaround is to load the file into Icon and save it, ideally without Alpha Channel. (Saving with Alpha will lead to many display devices getting confused, because of the photometric interpretation value of 5.

Optical Disk Mounts (QA 380)

Q: How many ODs can I mount at the same time through the Workspace Manager? What happens if I try to mount more?

A: You can have up to eight ODs (od0-od7) mounted at the same time through the Workspace Manager's autodiskmount facility. If you try to mount the ninth, you'll get an Alert Panel that says that /dev/rod8a doesn't exist.

Disk Disktab Front Porch/Back Porch (QA 435)

Q: What are the frontporch (fp) and the backporch (bp) shown in disktab?

A: The front and back porches are for "non-file-system stuff". The front porch has: the disk label and two copies of the boot program. The backporch is currently unused by the SCSI hard disk driver.

Lengthening Printer Cable (QA 433)

Q: Can the printer cable be extended somehow?

A: No. There will be too much signal skew.

Printer Voice Alerts (QA 375)

Q: How can I change the voice alerts for my printer?

A: The voice alerts are in the directory /usr/lib/NextPrinter. There are four of them:

manualfeed.snd/"Your printer is waiting for paper"

nopaper.snd/"Your printer is out of paper"

printeropen.snd/"Your printer cover is open"

paperjam.snd/"Paper is jammed in your printer"

These are ordinary CODEC-based sound files. One can replace them with one's own recording.(The execute bit need not be on.)

NeXT On Campus

Jeff_Wishne@NeXT.com

I am involved in producing the fall issue of the NeXT on Campus Journal. The NeXT on Campus Journal is a quarterly newsletter which showcases institutions and academics that are using the NeXT in innovative ways. In the fall issue, NeXT plans to add a new resource --A listing of academic projects and project managers. If you are currently working on a NeXT and would like to be included in the listing, please fill out the brief form at the bottom of this pageand return it to me via e-mail or U.S. mail. In order for a project to be listed in the fall issue, I must receive a form by Friday, July 27th Thank you for your time,

-Jeff Wishnie/ NeXT on Campus/900 Chesapeake Dr./Redwood City, CA 94063

p.s. If you have not had an opportunity to read the first issue of NeXT on Campus, it is available at the Purdue NeXT archive site, j.cc.purdue.edu.

Project Form

Institution/Organization:

Approximate # of installed systems:

Department: # of systems in department:

Project Manager Name & title:

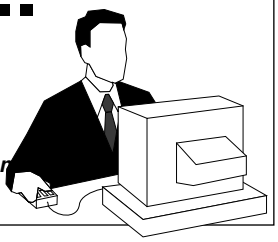
Address:

Phone & E-mail:

Brief Description of Application, Project or Lab:

The way I see it...

by
Bruce Henderson
atncpc!jupiter!bruce@NeXT.com



How many people have bought a NeXT?

I don't mean those of you whose Universities decided to crank out the cash and bought them direct from NeXT, Inc; or you Developers who bought them at the "nice price" ; or even you Students who somehow managed to have enough cash to buy one at your campus store (with some sort of campus blood sucking markup as well, I bet!). No I mean who out there actually went down to you local, authorized NeXT dealer, slapped down the major coins and walked out with your lovely white boxes of stuff.

Chances are, almost none of you.

As someone who is spending his working time, as well as a great deal of leisure time trying to write software for the machine, such statements naturally cause one to question his/her sanity. The truth of the matter is, there are so few NeXTs out there in the hands of people other than developers and schools, that to invest money in creating software for this machine is a sure ticket to economic suicide.

But the NeXT is such a marvelous machine. It should just fly off the shelves! That's what some of us might think. But the fact of the matter is it isn't true. Why not? Because (according to the ever so snobby salesman at one of my local outlets) there is no reason to sell them, because no one in their right mind would by this machine.

The story goes like this. It's early Friday morning. Before I go spend 14 hours working in front of my black monitor, I decide to drive down the road and visit my local NeXT reseller (no names here). I walk in, the store is beautiful, very high tech looking (they recently re-modeled). I wade my way past the Compaqs and Macintoshes to the back corner, where sits a NeXT. The first two things that I notice.

1) No one is playing around with the NeXT! A quick count reveals seven sales-droids performing various mindless acts with gizmos and widgets on other computers.

2) The machine looks unkempt. Almost as if I were witnessing a ritual of neglect that had gone on for weeks prior to my visit.

Seeing as I was one of 3 customers in the store, my presence eventually generated an interrupt in one of the sales-droids. It loped over towards me, in a neatly pressed suit, the hair greased back, wearing one of those Silicon Valley "I'm so upwardly mobile i dare not look down" power ties. It asked "can I answer any questions?" I decided to play it calm and see what the score was.

"Yes, what can you tell me about the NeXT computer?" I asked.

"Well, not much. But I'll try" it replied.

I though to myself, at least it was going to try and earn some of its wages today.

After a moment of staring blankly at the login prompt, it asked me "do you know how to start one of these things up?"

"Don't you just log in?" I replied.

"Well, the guy who knows how to sell this quit last week, I think he was the only one who knew how to start it up."

At which point the Management Object scuttled from it's hidden office, to see what was the matter. "What are you doing?" it asked the sales droid.

The unhappy but always arrogant droid responded that i had some questions about "that computer".

The pompous Management Object then informed me that the machine was broken, and that no one here knew how to fix it. Would I be interested in looking at a nice Macintosh IIfx?

The anger began to build in my guts (or was it the pizza for breakfast?) and I typed at the login window "root" hit return twice and was successfully logged in as god of this machine.

"How did you do that ?" the Management Object asked, clearly infuriated that I had just made a fool of him in front of his sub human minions.

"You know" I began, "I really wonder about the future of the NeXT, If people like you can't give an interested customer a demo. You tried to sell me a Macintosh instead." At that point I was sure it was anger building, and I let them both have it. "I actually depend on the NeXT for my bread and butter. If the machine doesn't sell, my software doesn't sell, then the company cancels the project and I am out on the street. I come in here and you guys can't demo this thing if your like depended on it!"

To which the now clearly infuriated Management Object replied "Well, why should we? We would have a much easier time selling Macs or IBMs or Compaqs! Besides, no one in their right mind would by this machine, the company is only going to be around a couple of more months anyhow. I read in MacWeek that the entire marketing staff resigned."

My eyes rolled. I couldn't believe that this person. A manager of the sole reseller of this awesome machine would actually utter such mindless comments in front of customers and his sales force!

If the gang in Redwood City ever puzzles about their dismal sales numbers, they should perhaps rethink about the dedicated business professionals they have recruited to sell their machine to the masses.

And that's the way I see it.