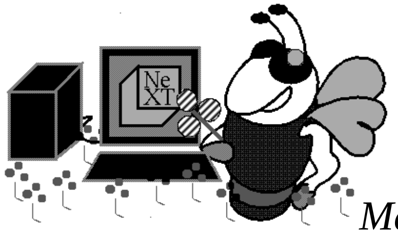
NeXT Extends Warranty on All Products to 1 Year



BuzzNUG *Buzzings*

March 1990 - Issue 4

Contents

The BuzzNUG General Council
Erica J. Liebman, Pres. erica@kong.gatech.edu
David Rosenbaum, V.P. daver@pyr.gatech.edu
David Samuel King, Tres/Sec dsking@pyr.gatech.edu
Phyllis A. Huster, Editorial Consultant, ccuseph@hydra.gatech.edu
Keith Edwards, keith@kong.gatech.edu, NeXT Campus Consultant
Tamora V. Sealey, Accounts tammy@cadnext2.gatech.edu

Contributers : Erica J. Liebman, Andrew Stone, Jeanette Allen, Phyllis Huster, David King, Chuck Fleming, Judy Halchin, Doug Brenner, Roger Rosner, Jonathan Schwartz, Morris Meyer!, Brian Bartholemew, Pat McGee, David Stutz, Doug Kiesler, M. Dixon, Elizabeth Hayes

"Grep Foo, whilst ye may, oh Daemons of the Spring, tho' thy sed be strong,
thy string is long and awk be not everything."

Welcome

My family has a hard-and-fast rule, which I inevitably break. Whenever I get too busy to chat, I am required to call at least once a week and tell my folks that I'm still alive. My sister Barbara called up a few days (weeks?) ago. "Why haven't you called your parents?" she asked. "Huh?" said I. "Didn't I just?" "Yeah," she said. "About two weeks ago." Oops. The *Welcome* this month goes out to all those people who I have ignored this quarter (and last, come to think of it). Sorry all. Here's the letter or phone call I've been meaning to send to you.

`Dear <appropriate-person-who-I-have-neglected>,`

This has been an especially busy quarter at school, although it is finally drawing to a close. I took far too many courses and far too much BuzzNUG for my own good. I know that you keep telling me to relax and enjoy life, but things got busy awfully quick. At least the courses I took were really good. I took a software engineering class on testing and maintenance and stuff which I really liked and that theory class turned out to be pretty cool after I stuck it through that second homework. I worked on my "Ugly Little Teapot that Could" video and took a course in program animation that I got to write an article about in this month's issue. John and I did manage to get that OOPSLA paper out. My fingers are crossed. The Musical is still on hold.

You can tell Dad that I've been spending a lot of time trying to track down a way to get Buzzings out on paper and finally decided to change the name to NeXT User's Journal, starting with the May issue. Mom will be pleased to hear that the apartment got especially clean as my term-project due-dates and the Buzzings deadline drew near.

I just wanted to write and tell you that even though I didn't call/write/return-your-e-mail/speak to you this month, (or even this quarter), I was thinking of you a lot. Hey, give me a call sometime. I may even be finished with my finals soon.

This message goes out to Ethel, Bernie, Sally, Paul, Quimby, Barbara, Grandma Mary, Grandma Rosalind, Unkie, Dwight, All-my-local-cousins-in-Atlanta-who-I-really-should-be-talking-with-more-often, Steve, Amy, Andy, Caro, Mark, David, Sharon, Deebster, Robert, Brian, Mark, Dale, all my friend NeXTies, Lily/Dave/Joe/Cathy/Kathleen/Robert/etc--whom I >really< owe some letters, and everyone I missed.

Buzz HUGS

Special thank you's go out this time to David King, Jeanette Allen and Phyllis Huster for making this issue become real. Thank you to Morris Meyer (and Avie!) for getting that Godzilla stuff to me, even when it became inconvenient for you. Morris, you did terrific. When do I get my Turtleneck? And thank you to Mark Landry, for suggesting (appropriately) this section.

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.
• Please spellcheck.
• Wit is welcome, overwhelming cuteness and obscenity are not.

Preformatting with the following guides will aid us ENORMOUSLY :

• Please use 1" and 7.25" margins, indent at 1.25", tab at 1.5" and each quarter inch thereafter.
• Title : 14 Point Times Bold, Left Justified.
• Author : 12 Point Times Italic, Left Justified.
• Article : 12 Points Times Roman with 12 Points Times Bold headers.
• Left justify bolded headers.
• Code : 10 Point Courier, no paragraph indent.
• Center graphics and use 12 Point Times Italic centered captions.
• Please avoid blank lines between paragraphs, but 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. 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.

Feedback from the Trenches

>NeXTFest, a rat fest for the nineties? Wouldn't it be great if we organized a meeting of the minds? Maybe this summer..
Andrew Stone, stone%hydra.unm.edu@ariel.unm.edu

>I have a question for you about Mathematica (yes i work for Wolfram). The question is: How does the average NeXT user go about getting support for Mma? Do they call NeXT, if so what is the quality of the support? Thanks.
Harold Barker, wri!barker@uunet.uu.net

Anyone had any experience yet with Mathematica problems? Drop Harold & me a line -- thanks.

>BuzzNUG Buzzings sounds like just what we need. Any chance of a free introductory copy and details of how to subscribe?
Nick Hoggard, nick@abblund.se

It's still free -- so far

> Pick up your copy of Buzzings #3 via anonymous ftp>> p.s. Tell me what you liked and disliked!
What I disliked most was not being able to print it out :-(
I only have access Laser Writers and Print Server 20s... Neither one has the %+ Ohlfs fonts. Suggestions?... (should I just switch it to courier?)
Aydin Edguer, edguer@curie.ces.cwru.edu

*I've uploaded a non-ohlfs copy of #3 to j.cc.purdue.edu, which so far can be found in **/pub**. We'll try to keep to Courier, Helvetica and Times in the future.*

> I'd really like to see some articles on the Music Kit. I've played with it a little, but would really like to hear from someone who really works with it. There have been some requests on comp.sys.next for information on how to synthesize instruments, but I haven't seen any responses. Perhaps you could put out a request for some articles on the Music Kit. Now that would be Cool.
Chuck Fleming, cfcgf@mrcnext.cso.uiuc.edu

>BTW, Buzzings #3 was even better than its predecessors (and that's saying a lot)!
Cameron Smith , cameron@wri.com

>How much lead time do you need on written material?
Eric Blankenburg, windsor@symcom.math.uiuc.edu

Try to get articles to me at least one month ahead of issue release. I have included a set of article guidelines in this issue. Check 'em out. We're counting on all of you for articles.

> I'm a modem user, I spent a lot of time in dealing with tip,/etc/remote, communication parameter. I did got help from Mark landry, Steve and Emory book store. But I was disappointed because I couldn't find "How to install a modem" chapter from manual. Tip is not friendly to new-modem user(personal view)...Some complains (expectation)about NeXT(BuzzNUG):

1. The price of Tech Docs shall be nonprofit.(why need $250-280?)
2. Again, There shall be a chapter dealing with modem usage in manual.
3. On-line Documentation is bad for beginner.
4. Some comments about Emacs and Edit. (It seems that I need learn both)
5. Review of Communication software(kermit,umodem,zmodem,Communicae,tip)
6. What are the candidates of the other three empty slots.
7. Tell Webster,some one is expecting his book(v1.0).(when can I get it?)

Golden T.J. Yang, ty@beach.cis.ufl.edu

We've got a mini-review of the Communicae demo in this issue. Readers : anyone willing to get me a review of communication software out there?

> Buzzings 1-3 were wonderful. The pictures... the articles... the introduction... They were just marvelous. We really really really loved it. Of course, we didn't understand a single word, but we love you darling. We're sure it made sense to everyone else.

Mr & Mrs B. Liebman, Erica's parents.

Adding a Custom Object to Interface Builder

Charles G. Fleming and Judy D. Halchin

Abstract

This article discusses customizing Interface Builder for user-defined objects in the Palette Window. Custom objects promote re-use, without time-consuming manual connections of outlets and actions. Sample code is included.

Introduction

You are probably aware that it is possible to use Interface Builder with custom objects in the palette window. We found this to be an efficient way to reuse objects over and over in a number of programs. Each instance of one object required a slider, three textFields, and a subclass of Object, then seven target/action and outlet connections between these. Clearly, handcrafting multiple instances is time-consuming. By adding this set of objects, with their connections, as a single object in IB's palette window, we made it possible to instantiate by dragging from the palette. In this article, we will outline customizing IB and provide source code for such an object.

The Custom Object

Our custom object uses a slider and with three textFields showing the current slider value, the minimum and the maximum values of the slider's range. At run time, the user changes any of these values by editing the textFields, and may set a new value by moving the slider. A controller object coordinates these activities, providing user value error checking of the fields. For example, the user may not enter a maximum value less than the minimum. A delegate is assigned for specialized checking -- an action message will to sent to the designated target whenever the current value is changed. To make this collection (controlling object, textFields, and slider) into a single object, we made subclassed View, which we called MaxMinSlider. This controlling object contains the other components as subviews.

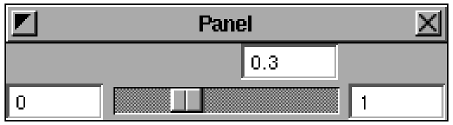
The *1.0 Release Notes for Interface Builder*, pages 4-7 contain a detailed outline of how to make a simple customization of Interface Builder, but our object required an additional technique (many thanks to Bruce Blumberg of NeXT for his assistance). If you carry out the instructions below, you will want to refer to these notes for more detail.

Customizing IB

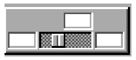
The first step in creating a customized IB is to write the class and header files for a new object. The code for MaxMinSlider.h and MaxMinSlider.m below is discussed later. Make two .nib files containing information for the Palettes window of the customized IB. Do this as follows.

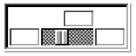
1. Launch IB and choose New Module. Click OK in the New Module panel that appears.
2. Drag in a Panel and put a CustomView in it. Size the CustomView to 255 by 44 or a little larger, and the panel to just large enough to hold the View.
3. In Autosizing, make the CustomView resizable in both directions.
4. Save the module as MaxMinSliderView.
5. Go to the Project inspector and click OK to create a project. In the attributes of the Project, select CustomIB, change the application name to ExtendedIB (or whatever you want your customized IB to be called), and click OK.

6. Now bring MaxMinSlider.h and .m into the current directory, if you haven't already. Bring in Delegate.h also (we'll explain its purpose later).
7. In IB's Classes window, add MaxMinSlider as a subclass of View and parse it.
8. Add the class file MaxMinSlider.m to the project.
9. In the Attributes window of the Inspector, make the CustomView be an instance of the MaxMinSlider class.
10. Now save and close the file.
11. Start a new module, again clicking OK in the New Module panel.
12. Drag in a Panel and size it to exactly the same size as your MaxMinSlider view was in the previous module.
13. Put into the panel a slider and three textFields, as pictured.



14. In the Miscellaneous window of the Inspector, change the names of these objects to minField (the one on the left), maxField, (on the right), valueField, and slider.
15. Save this work as MaxMinSliderContents.nib.
16. In the Project Inspector, add MaxMinSliderContents.nib as a .nib file and MaxMinSliderView.nib as a Palettes file.
17. The last thing to do in IB is to drag two .tiff files into the icons suitcase. These must be named MaxMinSliderView.tiff and MaxMinSliderViewH.tiff. These are the icons that will appear in the top portion of the Palettes window of your new IB. (The one with H in the name is a highlighted version of the other one.) We created these files in Icon. The tools provided there made it fairly easy to create a small picture of our object. You may want to use the ScreenGrab tool and simply grab the pictures shown below.


MaxMinSliderView.tiff


MaxMinSliderViewH.tiff

18. Once you've brought these into IB, save your work and make. The executable produced will be a customized IB.

While running this new IB, you may choose MaxMinSlider in the Palettes window. Drag one or more MaxMinSliders into a window, resize and rearrange their components . Set initial values, maxima, and minima by typing in the textFields. You may connect the MaxMinSlider delegate and target . You may not, however, specify the action method when connecting the target. This may be easily done in code, as we will see below. (Subclassing MaxMinSlider to Control enables this feature, but takes away access to rearranging and typing into fields.) When you look at the Connections panel in the Inspector window, note that the outlets of MaxMinSlider used internally, namely slider, maxField, minField, and valueField, appear unconnected. Ignore this. The connections are made in the code of

MaxMinSlider.m. You need *not* make these connections in ExtendedIB. Furnish voice messages and sounds for illegal user input by importing or recording badValueSound, badMinimumSound, and badMaximumSound.

Look at the newFrame: method in MaxMinSlider.m to see why we needed two .nib files. We want each instance of MaxMinSlider to have the outlets maximumField, minimumField, valueField, and slider connected, the three textFields and slider to be subviews of MaxMinSlider, and the slider and textField targets to be set to MaxMinSlider. "loadNibSection:MaxMinSliderContents.nib owner:self" sent to NXApp reads in information from MaxMinSliderContents.nib. This .nib file contains the slider and textFields. A pointer to the main (in this case, the only) window is returned. Free the panel, since we have no need for it, and use NXFindObjectWithName() to get pointers to each object we do need. Insert these into the view hierarchy as subviews of the MaxMinSlider.

Set the target/actions for each object. An object's newFrame: method is called as soon as it is brought into the application in IB, not at execution time. The object is saved in the .nib file and retrieved at execution time by the write: and read: methods. Reading in the values from the textFields in write:, we assure that any initial user-entered values while running ExtendedIB will be used. To understand more fully, insert printf's in read:, write:, and newFrame: with distinctive messages (or break on read:, write:, and newFrame: in gdb), then recompile and run ExtendedIB from Shell. You'll be able to see just when, in the course of running ExtendedIB, the read:, write:, and newFrame: methods are invoked.

ExtendedIB will not let you set the action message you want MaxMinSlider to send to its target. Fortunately, this is easily done in code. Suppose the target object should receive the action message drawGraph: whenever the value of the MaxMinSlider is changed. You must make the MaxMinSlider an outlet of the target object. If the outlet name is mmSlider, connect the outlet in ExtendedIB and add a setMmSlider: method to your object. Include

[mmSlider setAction:@selector(drawGraph:)];

in the setMmSlider: method to set the MaxMinSlider action. See the *System Reference Manual*, Chapter 3, page 19 for more information on @selector.

Delegates

The delegate of MaxMinSlider can choose to respond to any of the methods

```
  - (BOOL)maximumWillChange:sender
to:(double)possibleMaximum;
  - (BOOL)minimumWillChange:sender
to:(double)possibleMinimum;
  - (BOOL)valueWillChange:sender to:(double)possibleValue;
```

Each of these should return YES if the given value is acceptable, NO if not. However, when you compile MaxMinSlider.m, the compiler will object on the grounds that it has never heard of these methods. We solved the problem by making a file called Delegate.h as follows.

```
#import <objc/Object.h>

@interface Delegate:Object
{
}

- (BOOL)maximumWillChange:sender to:(double)possibleMaximum;
- (BOOL)minimumWillChange:sender to:(double)possibleMinimum;
- (BOOL)valueWillChange:sender to:(double)possibleValue;

@end
```

MaxMinSlider.m imports Delegate.h, so the compiler is kept happy, even if your delegate has not implemented these methods. This may not be a best solution to the problem, but it works and it's fairly simple.

Sounds, Messages and Enhancements

You may import sounds for user errors. These sounds may be set via code. Methods are also provided for other objects to set MaxMinSlider values, maxima, and minima or query for these values. The method setFloatingPointDigits: lets you specify a decimal accuracy for MaxMinSlider. See the comments in the code for further details on these methods. The code provided below should give you a reasonable functional level. Of course, lots of enhancements may be added to make the MaxMinSlider more versatile and more powerful.

Summary

In summary, when writing a new application using a MaxMinSlider object, do the following: move MaxMinSlider.h and .m into the new application directory, add these in ExtendedIB's Project Inspector. Move Delegate.h into the directory. Set the action message for the target via setAction: with the @selector directive. Otherwise, use a MaxMinSlider just as you would any other object in the Palette window.

MaxMinSlider.h

```
#import <appkit/View.h>
@interface MaxMinSlider:View
{
    id  delegate;
    id  slider;
    id  valueField;
    id  minimumField;
    id  maximumField;
    id  target;
    SEL action;
    id  badValueSound;
    id  badMinimumSound;
    id  badMaximumSound;

    BOOL shouldFormat;
    int formatDigits;
    double value;
    double minimum;
    double maximum;
}
+ newFrame:(NXRect *)frameRect;

- setDelegate:anObject;
- setTarget:anObject;
- setAction:(SEL)anAction;
- setBadMaximumSound:aSound;
- setBadMinimumSound:aSound;
- setBadValueSound:aSound;
- (BOOL)setValue:(double)possibleValue;
- (BOOL)setMinimum:(double)possibleMin;
- (BOOL)setMaximum:(double)possibleMax;
- getValueFromField:sender;
- getValueFromSlider:sender;
- getMinimum:sender;
- getMaximum:sender;

- setFloatingPointDigits:(int)digits;
- round:(double *)number;

- (double) value;
- (double) minimum;
- (double) maximum;

- read:(NXTypedStream *)stream;
- write:(NXTypedStream *)stream;
@end
```

MaxMinSlider.m

```
/* Generated by Interface Builder */

#import "MaxMinSlider.h"
#import "Delegate.h"
#import <appkit/appkit.h>
#import <soundkit/Sound.h>

@implementation MaxMinSlider

+ newFrame:(NXRect *)frameRect
{
    id panel;

    self = [super newFrame:frameRect];

    // load the nib file with the subviews
       panel = [NXApp loadNibSection:"MaxMinSliderContents.nib"
                                                    owner:self];
    // get pointers to the subviews and insert them as subviews
       slider = NXGetNamedObject("slider", panel);
       valueField = NXGetNamedObject("valueField", panel);
       minimumField = NXGetNamedObject("minField", panel);
       maximumField = NXGetNamedObject("maxField", panel);
       [self addSubview:slider];
       [self addSubview:valueField];
       [self addSubview:minimumField];
       [self addSubview:maximumField];
       [panel free];
    // set target/action of each subview
       [slider setTarget:self];
       [slider setAction:@selector(getValueFromSlider:)];
       [valueField setTarget:self];
       [valueField setAction:@selector(getValueFromField:)];
       [minimumField setTarget:self];
       [minimumField setAction:@selector(getMinimum:)];
       [maximumField setTarget:self];
       [maximumField setAction:@selector(getMaximum:)];
    // initialize formatting flag
       shouldFormat = NO;
    return self;
}
```



```
- setDelegate:anObject
{
    delegate = anObject;
    return self;
}

- setTarget:anObject
{
    target = anObject;
    return self;
}

- setAction:(SEL)anAction
{
    action = anAction;
    return self;
}

- setBadValueSound:aSound
{
    badValueSound = aSound;
    return self;
}

- setBadMinimumSound:aSound
{
    badMinimumSound = aSound;
    return self;
}

- setBadMaximumSound:aSound
{
    badMaximumSound = aSound;
    return self;
}
```

```
//  This should be sent by other objects who want to set the
//  current value of the slider. If the value is legal, all
//  appropriate actions are taken and YES is returned.  If the
//  value is illegal, the original value remains unchanged and NO
//  is returned.
- (BOOL) setValue:(double)possibleValue
{
    BOOL legal;

   // Format the given number
      [self round:&possibleValue];
   // Give another object the opportunity to check the legality
      if ([delegate respondsTo:@selector(valueWillChange:to:)])
         legal = [delegate valueWillChange:self
                                                  to:possibleValue];
      else
         legal = YES;

   // see if the value is within bounds
      if (legal && (possibleValue <= [slider maxValue])
                        && (possibleValue >= [slider minValue]))
      {
         value = possibleValue;
         [slider setDoubleValue:value];
         [valueField setDoubleValue:value];
         [target perform:action with:self];
         return YES;
      }
      else
         return NO;
}
```



```
//  This should be sent by other objects who want to set the
//  minimum value of the slider. If the value is legal all
//  appropriate actions are taken and YES is returned.  If the
//  value is illegal, the original value remains unchanged and NO
//  is returned.
- (BOOL) setMinimum:(double)possibleMinimum
{
   double oldValue;
   BOOL legal;

   // Format the given number
      [self round:&possibleMinimum];
   // Give another object the opportunity to check the legality
      if ([delegate respondsTo:@selector(minimumWillChange:to:)])
         legal = [delegate minimumWillChange:self
                                                  to:possibleMinimum];
      else
         legal = YES;

   // see if the value is within bounds
      if (legal && (possibleMinimum < [slider maxValue]))
      {
         minimum = possibleMinimum;
         oldValue = [slider doubleValue];
         [slider setMinValue:minimum];
         [minimumField setDoubleValue:minimum];
         if (oldValue != [slider doubleValue])
         {  [valueField setDoubleValue:[slider doubleValue]];
              [target perform:action with:self];
         }
         return YES;
      }
      else
         return NO;
}
```

```
//  This should be sent by other objects who want to set the
//  maximum value of the slider. If the value is legal all
//  appropriate actions are taken and YES is returned.  If the
//  value is illegal, the original value remains unchanged and NO
//   is returned.
- (BOOL) setMaximum:(double)possibleMaximum
{
   double oldValue;
   BOOL legal;

   // Format the given number
      [self round:&possibleMaximum];
   // Give another object the opportunity to check the legality
      if ([delegate respondsTo:@selector(maximumWillChange:to:)])
         legal = [delegate maximumWillChange:self
                                             to:possibleMaximum];
      else
         legal = YES;

   // see if the value is within bounds
      if (legal && (possibleMaximum > [slider minValue]))
      {
         maximum = possibleMaximum;
         oldValue = [slider doubleValue];
         [slider setMaxValue:maximum];
         [maximumField setDoubleValue:maximum];
         if (oldValue != [slider doubleValue])
         {  [valueField setDoubleValue:[slider doubleValue]];
              [target perform:action with:self];
         }
         return YES;
      }
      else
         return NO;
}
```



```
//  When a value is typed into the current value field, this
//  message is received.  If the new value is valid, the
//  MaxMinSlider’s target will be sent an action message. This
//  message should not be sent by other objects.
- getValueFromField:sender
{
    double possibleValue;
    BOOL legal = YES;

    possibleValue = [sender doubleValue];
    [self round:&possibleValue];
    if ([delegate respondsTo:@selector(valueWillChange:to:)])
       legal = [delegate valueWillChange:self to:possibleValue];
    else
       legal = YES;

    if (legal && (possibleValue <= [slider maxValue])
                      && (possibleValue >= [slider minValue]))
    {
       value = possibleValue;
       [slider setDoubleValue:value];
       [target perform:action with:self];
    }
    else
       [badValueSound play];
    [valueField setDoubleValue:value];
    return self;
}

//  When the slider is moved this message is received.  The
//  MaxMinSlider’s target will be sent an action message. This
//  message should not be sent by other objects.
- getValueFromSlider:sender
{
    value = [sender doubleValue];
    [self round:&value];
    [valueField setDoubleValue:value];
    [target perform:action with:self];
    return self;
}
```

```
//  When a value is typed into the minimum value field, this
//  message is received.  If the new value is valid, the
//  MaxMinSlider’s target will be sent an action message. This
//  message should not be send by other objects.
- getMinimum:sender
{
    double possibleMinimum, oldValue;
    BOOL legal = YES;

    possibleMinimum = [sender doubleValue];
    [self round:&possibleMinimum];
    if ([delegate respondsTo:@selector(minimumWillChange:to:)])
       legal = [delegate minimumWillChange:self
                                          to:possibleMinimum];
    else
       legal = YES;

    if (legal && (possibleMinimum < [slider maxValue]))
    {
        minimum = possibleMinimum;
        oldValue = [slider doubleValue];
        [slider setMinValue:minimum];
        if (oldValue != [slider doubleValue])
        {  [valueField setDoubleValue:[slider doubleValue]];
             [target perform:action with:self];
        }
    }
    else
    {
       [badMinimumSound play];
       [minimumField setDoubleValue:minimum];
    }
    return self;
}
```



```
//  When a value is typed into the maximum value field, this
//  message is received.  If the new value is valid, the
//  MaxMinSlider’s target will be sent an action message. This
//  message should not be send by other objects.
- getMaximum:sender
{
    double possibleMaximum, oldValue;
    BOOL legal = YES;

    possibleMaximum = [sender doubleValue];
    [self round:&possibleMaximum];
    if ([delegate respondsTo:@selector(maximumWillChange:to:)])
       legal = [delegate maximumWillChange:self
                                          to:possibleMaximum];
    else
       legal = YES;

    if (legal && (possibleMaximum > [slider minValue]))
    {
       maximum = possibleMaximum;
       oldValue = [slider doubleValue];
       [slider setMaxValue:maximum];
       if (oldValue != [slider doubleValue])
       {  [valueField setDoubleValue:[slider doubleValue]];
          [target perform:action with:self];
       }
    }
    else
    {
       [badMaximumSound play];
       [maximumField setDoubleValue:maximum];
    }

    return self;
}
```

```
//  Send the number of digits that should be kept to the right of
//  the decimal.  Send a -1 to turn off formatting.  All values
//  will be rounded to the specified number of digits.
- setFloatingPointDigits:(int)digits
{
    if ((digits >= 0) && (digits <= 12))
       formatDigits = digits;
    if (formatDigits = -1)
       shouldFormat = NO;
    else
       shouldFormat = YES;
    return self;
}

//   This function is for internal use only.
- round:(double *)number
{
    float power;
    int i;

    if (shouldFormat)
    {  power = 1;
       for (i = 0; i < formatDigits; i++)
          power = power * 10;

       if (fmod(*number, 1/power) >= 0.5)
          *number = floor((*number) * power) / power;
       else
          *number = ceil((*number) * power) / power;
    }
    return self;
}

- (double) value;
{
   return value;
}

- (double) minimum;
{
   return minimum;
}

- (double) maximum;
{
   return maximum;
}
```



```
- read:(NXTypedStream *)stream
{
    [super read:stream];
    slider = NXReadObject(stream);
    valueField = NXReadObject(stream);
    minimumField = NXReadObject(stream);
    maximumField = NXReadObject(stream);
    NXReadTypes(stream, "fffii", &value, &minimum, &maximum,
                                     &shouldFormat, &formatDigits);
    // get the bad entry messages supplied in ExtendedIB
       badValueSound = [Sound findSoundFor:"badValueSound"];
       badMinimumSound = [Sound findSoundFor:"badMinimumSound"];
       badMaximumSound = [Sound findSoundFor:"badMaximumSound"];
    return self;
}

- write:(NXTypedStream *)stream
{
    // get any values the user typed into the fields in Ext'dIB
       value = [valueField doubleValue];
       minimum = [minimumField doubleValue];
       maximum = [maximumField doubleValue];
       [slider setFloatValue:value];
       [slider setMinValue:minimum];
       [slider setMaxValue:maximum];
    [super write:stream];
    NXWriteObject(stream, slider);
    NXWriteObject(stream, valueField);
    NXWriteObject(stream, minimumField);
    NXWriteObject(stream, maximumField);
    NXWriteTypes(stream, "fffii", &value, &minimum, &maximum,
                                       &shouldFormat, &formatDigits);
    return self;
}

@end
```

Finis

NeXT's Great Text Editor

Andrew Stone, Stone Designs

I love the Edit program, thank you Bryan Yamamoto! Features like "Zoom", which shrink the file to just the method names and static declarations is extremely useful. The notion of opening a directory to double click open files is very powerful. [Command-D brings up the open directory dialogue, replete with name expansion]. Other features you may want to take advantage of is the ability to add your own menu commands to the Edit main menu. The **.userdict** file which resides in your home directory holds commands that get added to your "User" menu at runtime. It has a basic table syntax like this:

`<Menu Item Name> TAB <COMMAND to be performed> RETURN`

.userdict:

```
------------- clip here ---------------------

Current Time/Date date
Word Count Selection          wc|
Spell Check Selection   spell -d /LocalLibrary/hlist|
Add to Spelling List          spelling /LocalLibrary/hlist > /tmp/hlist ; mv
/tmp/hlist /LocalLibrary/hlist

File List               ls -l | sort +3 -nr|
Directory               ls |
Calendar                | cal 1988 |
Methods           grep "^[+-]" |
Word Count        wc |
Print                   enscript
Indent            indent -st
Time                    | date
Comment           awk -f /me/bin/comment.awk
DeComment               awk -f /me/bin/decomment.awk
------------- clip here ---------------------
```

These two beauties [by Kris Jensen] handle the way we comment and uncomment the code:

/me/bin/comment.awk:

```
------------- clip here ---------------------
{print "//|" $0}
------------- clip here ---------------------
```


/me/bin/decomment.awk:

```
------------- clip here ---------------------

   BEGIN {FS = "|"}

   {for (i = 2; i <= NF; i = i + 1) printf("%s", $i)
   printf ("\n")}
------------- clip here ---------------------
```

And this is my .edictDict which is where you can create macros that type in huge blocks of code. You simply type the first unique characters in the dictionary key, and then hit <ESC> and Bingo, there is the block. For example, typing "w"<ESC> would insert the code to handle disabling flushWindow, and then matching that with the reenable, flush method. Here are a few handy ones:

.edictDict

```
------------- clip here ---------------------

win     [window disableFlushWindow];\
        \
        [[window reenableFlushWindow]flushWindow];\

        return self;\
}

if      if («cond») {\
        «stmt» \
}

elseif  if («cond») {\
        «stmt» \
else {         \
        «stmt» \
}

-method - «methodname» «parameters» \
{                        \
        «stmt»    \
}                        \
```

```
    +method + «methodname» «parameters» \
    {                         \
            «stmt»    \
    }                         \


    for     for («start»;«end»;«update») {\
            «stmt» \
    }

    while   while («cond») {\
            «stmt» \
    }

    message [«target» : «method» «parameters»]

    printf  printf("«msg»\n");

header      /*********************************************************************
*****\
*\
*                  Object Name:\
*\
*-------------------------------------------------------------------------\
*                  History:\
*\
*                     Date  Initials    Comment\
*-------------------------------------------------------------------------\
*\
*                  Description:\
*\
*-------------------------------------------------------------------------\
*                  Programmer:\
*                  Copyright (c) 1989,1990 Stone Design Corp.  All rights
reserved. \
**************************************************************************/

copyright   /*********************************************************************
****\
*                  Copyright (c) 1989,1990 Stone Design Corp.  All rights
reserved. \
**************************************************************************/
    ------------- clip here ----------------------
```


These things just scratch the surface of Edit. There is the nest/unnest commands. There is the ability to double click on a paren or brace and the matching pair is selected. Many emacs-style commands are available to move the cursor without a mouse.Chapter 18-15 of the NeXT Reference Manual:Concepts goes into more detail of this fine program. Emacs commands are on page 18-18.

© 1990

Finis

The Custom Icon Quest

Doug Brenner, Weeg Computing Center,
University of Iowa

Overview

I've found the Digital Librarian a useful tool for searching the online information NeXT has supplied with their computer. After some reading in chapter 13 of the NeXT *User's Reference Manual*, I learned how to add and search my own targets with the Digital Librarian. (I use the word *target* because it conforms with the terminology used by NeXT.)

With time, however, I grew curious. Did I really have to live with the default folder icon for my targets? Where was the icon information kept and could I supply my own?

Well, the answer is that you *can* supply your own custom icons for Digital Librarian targets, and the entire process is not difficult. (It would be nicer if the Digital Librarian helped you to attach your own icons to targets—perhaps in a future release.)

Creating the icon

The first step is to create your custom icon. For this task I would suggest the Icon application located in **/NextDeveloper/Demos**. There are various techniques for creating icons (I'm fond of ScreenGrab…); I'll simply cover the basic requirements for a Digital Librarian icon.

An icon for the Digital Librarian should be 48 by 48 pixels in size (use Icon's Preferences… command) and should be saved in TIFF format *with* the alpha channel. The other important aspect of a Digital Librarian icon is its background—it shouldn't have one—the Digital Librarian will supply a light gray background on which the icon will appear.

As an example, here is an icon I created for my Macintosh Technical Notes. Graphic 1a is a window captured directly from the Icon application. Notice how the areas around the Macintosh and the Apple shadow appear white. Rather than being white however, this area is actually transparent to allow the Digital Librarian's light gray background to show through.

1a: The icon correctly drawn in the Icon application. | 1b: The icon as seen in the Digital Librarian. | 1c: The icon highlighted in the Digital Librarian.

Figure 1: The icon drawn correctly with a transparent background.

To create this effect you should draw over your icon's background after you choose "Clear" from the popup menu on Icon's Tools palette. To test your work, hold down the command key and press the "1" key. A "granite" background will move under your icon showing any transparent areas. (For more information on creating icons, choose "Creating_an_icon" from the popup menu on Icon's Info… panel.)

If you supply your own background for the icon (see graphic 2a), your icon will appear correctly in the Digital Librarian (graphic 2b) until you select it (graphic 2c).

2a: The icon incorrectly drawn with a light gray background. | 2b: The icon as seen in the Digital Librarian. | 2c: The icon highlighted in the Digital Librarian.

Figure 2: The icon incorrectly drawn with its own background.

Putting the icon in place

Once you have your custom icon created you need to associate it with the proper Digital Librarian target. This can be done in one of two ways.

Method 1: Copy the icon's TIFF file into the target's index directory and name it **icon.tiff**

When you index a target with the Digital Librarian, a directory named **.index** is created within the target directory. (I won't go into detail about the contents of this directory as it primarily contains files to manage the target index.)

As an example of this process, let's say you created a target called **MacTechNotes** in your **~/Library/Documentation** directory. After you index the target, the **.index** directory has been created. Using the Directory Browser you can drag your custom icon (say **MacNotes.tiff**) into **~/Library/Documentation/MacTechNotes/.index** and rename it **icon.tiff**. (In order to see the index directory, you may need to check "Special Files" in the Filter… panel displayed via the Directory Browser's View menu.)

Method 2: Add an "icon:" tag to the **Targets1.0** file in directory **~/.NeXT/targets** and copy the icon into the directory **~/.NeXT/targets**.

If you look at the format of the **Targets1.0** file, you'll find a group of tags and values for each target in the Digital Librarian. The basic format is something like the one below.

```
title: Macintosh TechNotes
position: 4
directory: /LocalLibrary/Documentation/MacTechNotes
```

Each of the above tags may be changed within the Digital Librarian itself and that is how I would suggest you change their values. The only tag that you might wish to use is the one for specifying an icon. (I believe this feature is left over from version 0.9.)

Directly after the "directory:" tag, on a separate line, you can place an additional line that gives the icon file name. The new entry should look something like the following.

title: Macintosh TechNotes
position: 4
directory: /LocalLibrary/Documentation/MacTechNotes
icon: MacNotes.tiff

By default the Digital Librarian looks for the icon in the directory **~/.NeXT/targets**. I would expect you can also specify a full pathname.

Method one is probably the preferred method. It allows you to index something in a public directory (e.g., **/LocalLibrary/Documentation**) and make your custom icon available to anyone who places your target into their Digital Librarian. The second method requires others to update their private **Targets1.0** file. The first method also allows you to distribute a complete Digital Librarian target, i.e., informational files, index directory and icon as a complete package.

Summary

As always Mr. Phelps, if you or any of your friends, find this procedure doesn't work after a new release of the Digital Librarian, I'll deny any knowledge of this entire procedure.

In other words, NeXT has yet to document this information. Any time a vendor doesn't document something you should be prepared for it to change.

References

NeXT *User's Reference Manual*, Chapter 13.
/NextLibrary/Documentation/NeXT/RelNotes/IndexingNotes.wn
/NextLibrary/Documentation/NeXT/RelNotes/IndexingPaper.wn
UNIX manual pages for *ixBuild* and *ixClean*

Additional hints

Should you decide to start your own Digital Librarian quest, here are a few additional hints to get you started. Enjoy and good luck.

*The **.roffArgs** file*

This file, like the file **icon.tiff**, appears in some index directories. If you have information formatted with nroff or troff macros (e.g., UNIX manual pages or the Sybase documentation) this is your way to tell the Digital Librarian what arguments to supply nroff for formatting.

*The directory **/NextApps/Librarian.app/Images***

This directory contains the little icons used in the Digital Librarian summary area. Chapter 13 of the NeXT User's Reference Manual (p. 252) lists many of the icons and what they represent. If you want to investigate further, this might be a starting location.

*The **Targets1.0** tags*

title: Title displayed under the target, by default the directory name. This can be changed within the Digital Librarian by selecting the text under the target, typing a new title, and pressing Return.

position: Position number of the target in the Digital Librarian target area. Targets are numbered from left to right starting with zero. This can be changed within the Digital Librarian by dragging the targets within the target area.

directory: The complete pathname of the indexed directory. This is determined when you drag a target into the Digital Librarian.

selected: If this tag is present, its value is "Yes" indicating that the target is currently selected within the Digital Librarian.

icon: This tag associates a TIFF file with a target. The NeXT supplied targets do not use this tag; instead they use files named **icon.tiff** in the index directories.

—Doug Brenner <dbrenner@umaxc.weeg.uiowa.edu>
Weeg Computing Center, The University of Iowa

Finis

The Best Free Software

Roger Rosner and Jonathan Schwartz

Free software, by any name, is a great thing. Some of the world's most useful programs come without a price tag. While NeXT's choice of the Unix™ operating system has sparked some debate, there is no doubt that doing so provided convenient access to what is quite possibly the world's largest and most interesting collection of freely distributable software.

Although this article references highlights of the free software world and provides pointers for accessing archives, we make no pretense of avoiding shameless self-promotion. This article is being written during the release of a major collection of publicly available software for the NeXT. The collection was compiled by Lighthouse Design in conjunction with Quality/Liebsoft and contains well over 400 megabytes of software, when uncompressed. Although most everything on the disk is available without charge elsewhere, the amount of effort involved in collecting this software was considerable. This is our way of owning up to (and perhaps apologizing for) the disk's $125 price tag.

NeXT-specific Software

The activity on the NeXT software archives has been heating up rather nicely of late. Hardly a day goes by without new contributions. This, in our opinion, is surely one of the strongest indications of the future potential of the NeXT Computer. Some software is a little rough in the user interface department, but a number of programs are quite good nonetheless.

Highlights as of this writing are

- Roy Mongiovi's **Monitor**, an excellent machine performance monitor;
- Bryce Jasmer's **LockScreen**, which prevents others from using a cube when it is unattended—however, LockScreen is shareware;
- Andrew Stone's **CharFind**, a Key Caps for the NeXT;
- Eric Ly's addictive **Tetris** game;
- Jayson Adams' **AltDock**, which provides a windfall of dock real estate;
- and Kevin Steele's **NX_VOID**, a 3-D space video game.

A number of useful information files, copies of the *Buzzings* newsletters, programming documentation, some demos of commercial software, and even a few Objective-C classes are available on the archives as well.

comp.sources

For a number of years, authors of free Unix software have submitted their code moderated Usenet news conferences like comp.sources.unix, comp.sources.misc and comp.sources.games. Some of this software has become integral to the Unix world (e.g., the **compress** utility and **bnews**). Other highlights of this collection are the **rn** and **nn** newsreaders, the **perl** and **xlisp** programming languages, and the **cnews** news processing software. There are literally hundreds more useful utilities here. Fortunately, a comprehensive index, with descriptions, is also available.

The Free Software Foundation and GNU

Richard Stallman, founder of the Free Software Foundation, thinks all software should be free. While not everyone agrees with him, we can all certainly appreciate the fruits of Mr. Stallman's genius. He and the FSF have created some of the world's best freely distributable software.

One of the goals of the FSF is to build a complete Unix-compatible operating system, GNU (for GNU's Not Unix). To this end, the FSF has created a number of programs that, at first glance, seem identical to standard Unix utilities. Closer examination usually reveals broadened functionality and performance improvements.

It is difficult to do justice to the magnitude and quality of FSF software in a few sentences. Suffice it to say that they have created and currently maintain one of the best text editors available (**emacs**), and one of the best C compilers available (**gcc**), in addition to scores of other excellent programs. In fact, NeXT chose to base its Objective-C compiler and debugger on FSF's; they also supply GNU emacs with every cube.

We have heard that a number of companies and sales people discourage the use of FSF software. They claim the software, since it generates no profit, cannot be well maintained or have similar quality to commercial software. These claims are simply unfounded. Though it is true that the FSF has no legal obligation to maintain their software, in practice, they are significantly more responsive to their user communities than most commercial software houses. Furthermore, because the GNU sources are widely available and produced by some of the brightest programmers around—Stallman's feats of programming are legendary in the artificial intelligence community. GNU software is incredibly powerful, complete, and stable.

Much of the GNU distribution has not yet been customized for the NeXT, so installation may take some programming prowess. In our experience, however, a good portion of the programs compile without a hitch.

The Rest of the Net

The amount of software on the net today is staggering, certainly on the order of gigabytes. But there are no complete central repositories or indexes, and finding interesting programs is often by word-of-mouth. One of the best ways to stay informed is to read the comp.archives Usenet newsgroup.

Useful sources available on the net include the following:

* non-AT&T portions of BSD Unix
* networking programs, mail programs
* a great variety of programming languages and utilities (such as Walter Tichy's much lauded Revision Control System—**rcs**, an old version of which comes with the NeXT)
* hundreds of little Unix utilities to do almost anything.

Many of these programs are complex or obscure, but for those with courage, a great many re-inventions of the wheel can be avoided.

The Network Archives

There are literally dozens of sites on the Internet and Usenet that have some form of archive. All the major universities, and many corporations keep archives.

Of the two major archives of NeXT-specific software on the Internet, cs.orst.edu (maintained by Oregon State University) and j.cc.purdue.edu (maintained by Purdue University). cs.orst.edu seems to get the most contributions with things eventually reaching Purdue. Both archives are accessed by using the “ftp” program.

The Purdue archive also maintains a mail server which lets non-Internet hosts access the archives by sending and receiving mail messages. To learn more about the mail server, send a message containing the single word “help” to “archive-server@cc.purdue.edu”.

Unfortunately, there are no NeXT archives currently allowing anonymous uucp access. Contrarily comp.sources and GNU software are available by anonymous ftp from a number of places. Two of the major archive sites are uunet.uu.net and j.cc.purdue.edu, but the primary distribution point for GNU software is prep.ai.mit.edu. Oregon State maintains an anonymous uucp service (host osu-cis). For more information, refer to the comp.archives newsgroup.

A database of archives accessible by mail, ftp, or uucp exists and can be reached by sending an e-mail message with the single word “help” to comp-archives-server@twwells.com. Bear in mind, however, that this database is somewhat incomplete: given the estimated 4.5 million people on the net, being a little behind is understandable.

Please be careful to access these sites during off-peak hours. Site administrators are providing these services out of the kindness of their hearts, and significant loads during work hours could easily cause management to discontinue access.

To make effective use of network archives, you should use the **compress** and **tar** utilities. You will probably need to compile and install any programs you download. For major system programs, like **cnews** or **bnews**, the installation can be harrowing.

Safe Software

We would like to announce that all free software is good software, or at least well intentioned software. However, this world is home to many with less than a kind heart. Some with a nasty streak can even write code (though seldom is it good code). Since the results have been splashed all over the front page, we won’t bore you with a long history.

Suffice it to say that Unix (and thus the NeXT) is not very secure when confronted with a determined attack. Although it seems no new viruses, worms, etc. have been targeted at the cube, there will possibly come a day when some ego-starved prankster writes a few lines of dangerous code, or revives some old Unix virus, and hurts some NeXT users.

The solution is to get your free software from reputable distribution points. The major network archives—cs.orst.edu, j.cc.purdue.edu, or uunet.uu.net—are usually quite safe, as are most commercial software distributors. Beware of people wandering around users group meetings with disks full of the latest neat stuff. You don’t know where those disks have been, or what lurks within. Saturday Night Live said it best (an indisputable truth, but what follows bears no resemblance whatsoever to their quote so don’t get you’re hopes up): “when you share software with someone, you’re sharing your software with everyone else they’ve ever shared their software with.”

Please don’t let these warnings deter you from taking advantage of free software. Taking some minor precautions should sufficiently protect you, and your increased productivity from using free software should more than balance the remainder of the risk.

Most importantly, *always back up important data.* There is no better protection.

Pay Up When Asked

As everyone knows, not all publicly accessible software is free. Please pay for the shareware programs you use—it helps to support further development.

The authors of this article, Roger Rosner, and Jonathan Schwartz, are partners in Lighthouse Design, creators of electrical engineering CAD tools and CASE tools for the NextStep environment, and distributors of an optical disk of freely distributable software.

For more information on the compilation disk mentioned above, send electronic mail to disk@lighthouse.com or ...!uunet!lighthouse!disk, or send physical mail to Lighthouse Design, 6516 Western Avenue, Chevy Chase, MD 20815-3212, or call 800-366-2279.

For more information on other Lighthouse Design products or to contact the authors, send electronic mail to lighthouse@lighthouse.com or ...!uunet!lighthouse!lighthouse, or send physical mail to Lighthouse Design, 6516 Western Avenue, Chevy Chase, MD 20815-3212, or call 800-366-2279.

Finis

The "Godzilla" approach to distributed computation

Morris Meyer, NeXT, Inc.

Abstract.

A "Godzilla unit" is defined as fifty networked NeXT Computers, each running with eight megabytes of physical memory and a local disk. The combined amount of processing power is equivalent to a 250+ MIPS machine with 400 megabytes of real memory, 4 gigabytes of virtual memory, and 10 gigabytes of optical disk storage. A Godzilla network was used recently to settle numerical cases of Fermat's "Last Theorem" to an unprecedented limit.

1. Introduction

In 1989 the Educational Fellow at NeXT, Prof. R.E. Crandall, came to me with the problem of distributing the numerical resolution of special cases of Fermat's "Last Theorem." He and mathematician Prof. J. P. Buhler of Reed College had some new algorithms for settling Fermat's celebrated conjecture: that $x^n + y^n = z^n$ has no positive integer solutions when $n > 2$. They had developed, on single NeXT Computers, algorithms that, given enough processing power, would settle the Fermat dilemma for all n less than one million. The record since 1987 has been $n < 150{,}000$, so this project had a certain allure.

As presented to me, the criteria for the distributed processing were as follows:

1. The program was to involve little or no maintenance.

2. Running the program on the machines in NeXT's software department needed to be virtually invisible to the programmers in the department, and not hinder their productivity in any way. In other words, *Godzilla should be polite*.

3. The program should be written as quickly as possible given the other two constraints. Many features of Mach on the NeXT machine were used, as well as other standard UNIX text processing utilities.

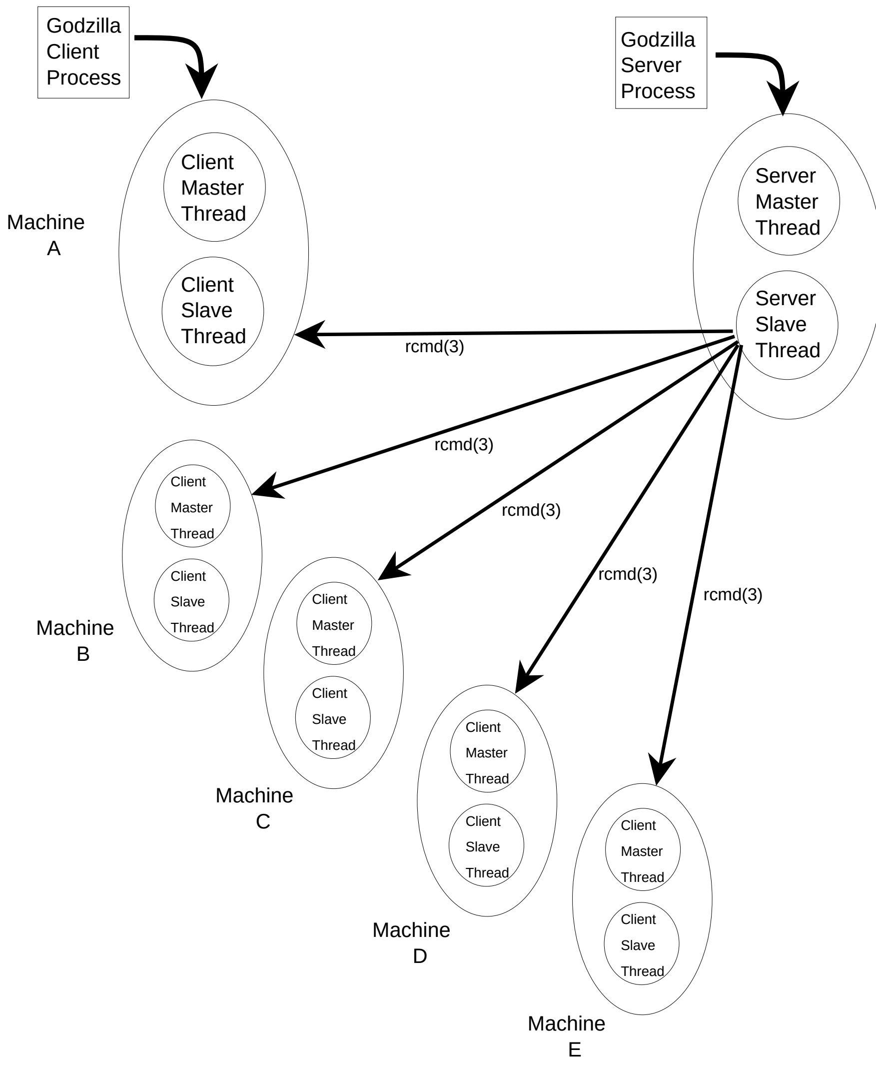
2. Purpose

This paper will show how these three constraints three constraints were achieved in under three days on the NeXT machine. This paper will also lay the framework for future distributed processing projects using scores of NeXT Computers.

3. Godzilla structure

Godzilla has one server process containing a master and a slave thread, and N client processes running on remote machines (figure). Each client process has two threads of execution and is virtually invisible to the programmer working on his or her machine.

During the bootstrap of the Godzilla program, the server creates a slave thread. The slave thread scans through a list of machines contained in the Godzilla server configuration file. Using the UNIX **rcmd(3)** library utility, the slave thread executes Godzilla clients on each of those machines.



Each client program starts up on the remote machines and looks for a port that was allocated by the server using the **netname_check_in()** function described in Chapter 27 of the NeXT System Reference Manual. The server has already allocated a network-wide port that each client uses to communicate with the server.

The client program creates a slave thread to run the computation that searches for counter-examples to Fermat's Last Theorem. The client slave thread loops, giving the server results for each integer exponent searched. The server returns the next integer exponent in the server's list for the client to compute.

The messages that travel between client and server are generated by MiG, the Mach Interface Generator. The MiG specification file for the entire Godzilla distributed service was less than thirty lines (Listing 1).

While the client slave thread works on the computation, the master thread loops, sleeping ten seconds between each loop. The master thread suspends the slave when wakened and the screen is not dimmed . The ten second interval inevitably went unnoticed by the programmer at his or her NeXT machine.

If the screen is dimmed, the Godzilla client master thread resumes the client slave thread if suspended. If the slave thread is active, the client master thread does nothing. Since the Godzilla client only checks whether the screen is dimmed every ten seconds, a programmer returning to his or her machine might notice a sluggish machine for up to ten seconds until the next check is made. Programmers in the software department have found a few seconds of sluggishness to be acceptable.

3. Godzilla conclusions

Godzilla was conceived, created, and debugged in two fourteen-hour programmer days by a programmer first learning about the Mach C-threads package and MiG. The NeXT machine has proven to be an excellent platform for program development and debugging. UNIX tools like **awk(1)**, **grep(1)**, **sed(1)**, **sort(1)**, and **split(1)** worked the output into a form that could be used for scientific analysis. The work done on the output could have been done inside Godzilla, but the added complexity and programmer time was a few times greater than the five minutes a day spent running a **C-Shell** script at the end of each run.

Finis

Listing 1:

```
/*
 * MiG  specification file for Godzilla server
 */
#include <std_types.defs>
import "distribKitTypes.h";
subsystem distribKit 0;
type retval_t = int;
type resultstring_t = (MSG_TYPE_STRING, 8*1024);
WaitTime 50000;

function NotifyClientSuspended (
                                server : port_t;
                                message : resultstring_t) : retval_t;

function NotifyClientResumed (
                                server : port_t;
                                message : resultstring_t) : retval_t;

function ServerMessage (
                                server : port_t;
                                message : resultstring_t) : retval_t;

function GetNextValue (
                                server : port_t;
                                lastresult : resultstring_t;
                                lastvalue : int;
                                out nextvalue : retval_t) : retval_t;

function CheckForTerminate (
                                server : port_t;
                                out shouldterm :retval_t) : retval_t;

function SendClientHostname (
                                server : port_t;
                                hostname : resultstring_t
                                ) : retval_t;
```

My First Week On a NeXT

Brian Bartholomew, University of Florida

A few weeks ago, I discovered that the campus computing center was getting a NeXT machine for two weeks of unlimited demonstration and exploration by students. After following the exploits of NeXT, inc. for months, I could experience a cube. Erica expressed great excitement for an article describing my first week's experiences. Due to a surplus of hot breath, that of the demonstration room operator, on my neck, I could not use the machine meaningfully. Luckily, the <kind> Electrical Engineering department allowed me time on a machine during slack periods. Thus this missive.

Although I started U*IX with two years of running an ASCII terminal off XENIX on a Compaq 386, the past year I've used X-windows (rel 3 & 4) on a color Sun 3/80. Combined with the good connectivity of the Computer Science department, this environment created a powerful baseline against which to form opinions about the NeXT.

There is one important thing which I must mention. While I have read every bit of advertising literature available about the NeXT, I have still not read a page of user or system documentation. Not even the quick start guide. This attitude is motivated by impatience, curiosity--about exactly how "obvious" the machine is to use --and a desire to see what knowledge transfers from X on the Suns.

When finally I logged onto a NeXT, my first impression was of the keyboard. It felt flimsy and cheap. I was expecting a good, solid keyboard that could take a nuclear explosion or 300 pound user with aplomb. Instead it felt like something from a cheap PC clone. The lack of angle-changing legs was an unwelcome omission. The mouse was different from what I was used to. The Sun uses an optical mouse on felt pads, pushed along a special grid plate; the NeXT rodent contains a rubber-coated steel ball. When I tried to point with the mouse, I overshot terribly. Eventually, I realized I was using a proportional mouse - that is, the screen space covered by the mouse is proportional to the rate at which the mouse is pushed and how far it travels. I double clicked on Preferences and found that the mouse was set to the most exaggerated proportional setting. Moving it down to the second setting made it work more like I expected. The video display was the sharpest I had ever seen. The grey levels made everything look extremely snappy, especially compared to the pixel-dithering used on the B/W Sun's.

I wanted to bring up a terminal window to type U*IX commands and get my bearings. Obvious things with the browser in /bin took care of that. With the second window open, I noticed that the csh(1) had command completion on Escape, but not command editing. Also, the keyboard input focus did not follow the mouse pointer, but instead was click-to-type. This is an extreme hassle, as I am used to the focus following the mouse pointer.

Eventually, I moved the mouse to the software I wanted to work with and clicked repeatedly until I got some response. Due to high software quality, I never blew anything up this way, although I still got type-in sent to wrong programs because I did not select them as active. As I tried out various applications and demos, I got a wait cursor more often than expected. (On the Suns, U*IX is expected to be multitasking, with the user doing several things at once). I expected to be able to flit between windows, depositing a few keystrokes to each. Waiting around for a non-multitasking window manager to get its act together was a seriously annoying. Eventually, I found (through sheer impatience)

that the cursor shape is managed by the application which controls the focus, not the window manager itself. This means you can ignore the wait cursors and switch tasks as you please.

In general, the system *felt* different. I have already mentioned how slow the startup is, but window operations are blindingly fast. On the Sun, windows first redraw borders and then the text from top down. On the NeXT, however, windows just pop up with the new contents, without warning. In a few cases, I was not sure whether a window's content had changed since I was used to screen redraws as visual confirmation of some action. Printing slows the system to a *crawl*, but once it starts it *goes*. I wholeheartedly support the design tradeoff NeXT made of sharing printer smarts with the rest of the computer; the modularity and controllability--and low price--are easily worth a few pauses in computing. Perhaps some person familiar with the DPS implementation could add code to the printer spooler, making the page image at about 75 dpi for a serial Epson or other cheap dot-matrix. I'd expect the output on a 24-pin printer would be quite good.

As my hours-in-type grew, I found that there were many subtle human engineering features. Having monitor brightness and volume controls near is unexpectedly handy. Sliding the keyboard under the crook of the monitor stand frees up a large amount of desk space, especially when you can easily roll the monitor back using the knobs on the front of the stand. The ability to wheel what must be a 50 pound monitor with just your fingers lets you willfully rearrange your desk. I immediately noticed quick, tangible benefits from the menu consistency. You can type Command-q to anything, and it will go away. Thank you NeXT for putting all the vertical and horizontal scroll buttons right next to each other, rather than scattering them to the far corners of the window (like a certain sluglike window manager does. In case the reader needs a hint about the OS to which I refer, consider the slash ("/") in the name as a fraction-style division sign. Then, the name says it all; "half an operating system.")

I believe the philosophy of maximizing application throughput is significantly different from that used on the Sun. On our Suns, you open one large application at a time, fairly quickly, and close it when done. This keeps memory usage and swap traffic to a minimum. Closing a window to an icon doesn't reduce its load on the system, it just saves screen space. Seeing thirteen icons at login time made me worry about system load, although eventually I remembered the "3 dots" convention for non-running programs. On the NeXT, a good idea is to start up all the programs for a session, and go drink a recyclable aluminum can of your favorite, caffinated beverage while they start. Once everything settles down and you use them a bit, the various caches fill up and things become reasonably fast. When you finish with an application hide it, don't kill it. This way, you are only penalized for startup once, not each time it opens, as you might if you were using a Sun.

NeXT in Review

Brian's® Critique List

Good Hardware:

• You don't have to the mouse keep square to pad.
• The sound quality is pretty incredible, even from the monitor speaker
• Volume and brightness keys are right at hand -- another "you didn't know you needed this until you try it"feature.

Good Human Engineering

• You get more screen space due to smaller fonts - but still legible.
• The mouse pointer disappears after awhile, when you don't move the mouse. *Nice*.

Bad Software

• The shell/terminal dicotomy is ridiculous.
• You have to initially search for Terminal/Shell, to get any work done
• GNU Emacs doesn't support the mouse (like the X version does).
• Although, the man pages in shell work nicely, man pages through digital librarian messed up by proportional font.
• Startup is *slow*,, but once started everything improves.
• It is hard to figure out how *read* manuals, not *search*, mind you, but *read.*
• The meta key seems not to be defined in emacs.

Good Software/Hardware

• WriteNow gives priority to displaying immediate keyboard input, rather than slavishly formatting after each keystroke - this means non-immediate formatting is ok.
• I liked how menus follow apps, kept lots of useful information on the screen, without drowning me in windows - plus, clicking on a window activates it AND raises it to the top AND opens it AND shows the menu.
• I like everything using mouse button one - so tell me, what can I use the second for?
• I could not find WriteNow documentation in digital librarian - although I didn't look very hard.
• In general, I kept clicking until something happened and didn't have to worry about destroying things this way due to software quality and good prompts.
• Finally, I got used to the mouse. In fact, it is much better than Sun mouse (surprised?). It allows you to move easily -- almost without moving the heel of your hand

Finis

Curses!... No, I mean cursors.

(Duplicating the Interface Builder Text View object)

Pat McGee, Computer Graphics Group

```
#include <long.rambling.introduction.wn>
```

Some people are just gluttons for punishment. Here I am, writing a strip chart recorder object, doing it without using Interface Builder.

Why am I doing this? Does insanity run in my family? Well, I want several views in my object, with a menu that selects these. This is a cinch in IB. But, if I wrote my strip chart recorder with IB for the internals, then users of my object would have to perform connections themselves. I can just see the directions now: "Step 24: connect the X Grid view radio button 1 to the 'SetXGridOff' action in the Annotated Strip Chart instance." (Ugh!)

Life would be so much nicer for documentation writers (me) if objects had lots of other objects hidden away inside them. It would also make life easier for the ultimate user (me again). So, now I need to duplicate anything IB does.

As it turns out, duplicating most IB is easy. If IB can do it in one click, it generally takes only one line of code to duplicate it standalone. Buttons are easy. So are switches, text fields, panels, menus and scrollers. However, a few things that don't work this way. The neat little ScrollView with a Text object inside is one of them. This is a pretty tricky piece of code, at least for a beginner like me.

One view I wanted inside my strip chart recorder was one that would work just like ScrollView. So, I started with a CustomView, subclass ScrollView, and added code. I made two applications, one using IB to a ScrollView into a small window. The other used my custom code. Then I inspected each with gdb. Whenever I found a difference between instance variables, I added some code to my implementation to change that variable.

Most things were pretty easy to figure out. A problem with this approach is that some things don't seem to be stored in instance variables -- the text font, for example. I had all the instance variables the same (well, all the ones that seemed to matter), but the views still looked different. I realized mine was in Times-Roman, and IB's was in Helvetica. So, I added some code to change the font. The font information, I figure, must be stored in the gstate, instead of the Text object.

Then I got to the real problem. IB's window changed the cursor to an I Beam when over that view; mine didn't. In the documentation, I found the instructions for changing the cursor to be less than clear. I sent a quick note to Bruce Blumberg. Lo, and behold, some sample code showed up the next day. (Thanks, Bruce!) Some fiddling with the rectangle parameters got my ScrollText working just like IB's ScrollView with Text.

I have learned a lot about how to use NeXTStep and the appkit while working on this. Using IB teaches you what *can* be done. Doing the same things directly with appkit teaches *how* these things happen.

```
#end  <long.rambling.introduction.wn>
```

The following code duplicates ScrollView functionality with the Text object that IB gives you on the palette. As far as I can determine, IB's ScrollView has no connections or actions. If you want to message the subordinate Text object, you must ask the ScrollView for its docView. In the interests of compatibility, my ScrollView object does the same.

To use this object from IB (why would you want to?), subclass ScrollView, name the new class "ScrollText", and parse. Make a custom view and change its class to ScrollText. (Actually, there *is* another difference. IB's ScrollView object defaults to autosize itself on both X and Y. ScrollText, like any CustomView, defaults to no autosize.)

To use this object from your own code, send

```
obj = [ScrollText newFrame: &rect];
```

with the rectangle you want it to live in. TextObj is a Text object; you can send it any Text message. For instance, to put some text into it :

```
TextObj = [myto docView];
[TextObj setText: "This is \n a string value\n."];
```

Anyway, the code follows.

P.S. You may be curious about my strip chart recorder object. I'm working on an application that will monitor a Learning Classifier System (a kind of AI program), and display about forty or fifty variables continuously. It's not ready yet, but I am planning on writing an article about it when it gets ready. The interface works, but not all the functionality is there yet. I ask that you please hold your letters for now. Thank you.

Pat McGee
Computer Graphics Group
Los Alamos Nat. Lab.
jpm@lanl.gov

Finis

=================== **ScrollText.h** =======================

```
/* ScrollText.h - Duplicate Scrolling Text object from IB default items palette
 * 2 Mar 1990, last date modified
 * written by Pat McGee, jpm@lanl.gov
 */


#import <appkit/ScrollView.h>

@interface ScrollText:ScrollView
{
}

+ newFrame: (const NXRect *) frameRect;
- resetCursorRects;

@end
```


=================== **ScrollText.m** =======================

```
/* ScrollText.m - Duplicate Scrolling Text object from IB default items palette
 * 2 Mar 1990, last date modified
 * written by Pat McGee, jpm@lanl.gov
 */


#import "ScrollText.h"

#import <appkit/Cursor.h>
#import <appkit/Font.h>
#import <appkit/Scroller.h>
#import <appkit/Text.h>

@implementation ScrollText

+ newFrame: (const NXRect *) frameRect;
{
	NXRect		contentRect;		// rectangle to put Text object in
	id		cView;			// the Text object
	id		dFont;			// font object to set Text font
	id		scrollsContentView;	// can't get  ClipView directly,
							// use this
	id		temp;			// for checking status in gdb
	NXSize		tempSize;		// for setting max and min sizes


	self = [super newFrame: frameRect];

	// duplicate attributes of IB object
	[self setVertScrollerRequired: YES];
	[self setHorizScrollerRequired: NO];
	[self setBorderType: NX_BEZEL];

	// do the content view - a Text object, and duplicate IB attributes
	contentRect.origin.x = contentRect.origin.y = 0.0;
	[self getContentSize: &(contentRect.size)];
	cView = [Text newFrame: &contentRect];
	[cView setOpaque: YES];
	[cView setAutosizing: NX_WIDTHSIZABLE];
	tempSize.width = tempSize.height = 1e+30;
	[cView setMaxSize: &tempSize];
	[cView setMinSize: &(contentRect.size)];
	[cView setVertResizable: YES];
	[cView setOverstrikeDiacriticals: NO];
```

```
        dFont = [Font newFont: "Helvetica" size: 12.0];
        temp = [cView setFont: dFont];

        temp = [self setDocView: cView];

        scrollsContentView = [cView superview];
        [scrollsContentView setAutosizing: NX_WIDTHSIZABLE |
                NX_HEIGHTSIZABLE];
        [scrollsContentView setBackgroundGray: 1.0];
        [scrollsContentView setAutoresizeSubviews: YES];

        [self resetCursorRects];

        return self;
} /* end newFrame: */



- resetCursorRects
{
        NXRect
        contentRect;


        contentRect.origin.x = contentRect.origin.y = 0.0;
        [self getContentSize: &(contentRect.size)];
        contentRect.origin.x += NX_SCROLLERWIDTH + 3;// I don't know why the +3
        contentRect.origin.y += 2;              // again, I don't know why, but it works.
        [self addCursorRect:&contentRect cursor:NXIBeam];
        return self;
}

@end
```


Convex Hull Animation

Erica J. Liebman

On a convex hull, any two points may be connected without passing through any third hull point. Convex hulls are used widely in graphics, image processing, graph theory and computational geometry. Since this algorithm is rather simple, I picked it as suitable for helping me better understand program animation.

This program illustrates a number of techniques : lists, handling mouse-down events and drawing to a custom view. Originally, I wanted lists of data points, using **NXPoint**. I quickly found that the **List** object dealt only with objects of type **id**. To handle this, I created a new class *Node* which has one NXPoint instance variable. I was displeased about the space overhead (4 bytes/node object + code), but could not arrive at a better compromise.

Second, I had to deal with mouse-down events. I chose to draw a circle at each mouse-down corresponding to a location for a data point. The simplicity of my mouseDown method is due in part to the minimum of interaction needed. I did no mouse-tracking. The mouse event returned has a **location** member of type NXPoint.

Third, I had to deal with drawing and animating. I created line and circle drawing methods (connect::::: and linScan:::: -- the names derive from their initial function in the linear scan of the hull points and connecting those points during the algorithm). The basic drawing scheme in a **View** subclass is :

```
[self lockFocus];
```

PostScript (PS)-code & supplementary code for drawing;

```
[self unlockFocus];
```

Avoid doing too much in the Postscript section. If system events interrupt, you may be left in a corrupted state. I learned to minimize the time in a locked focus through hard experience. May you learn from my mistakes.

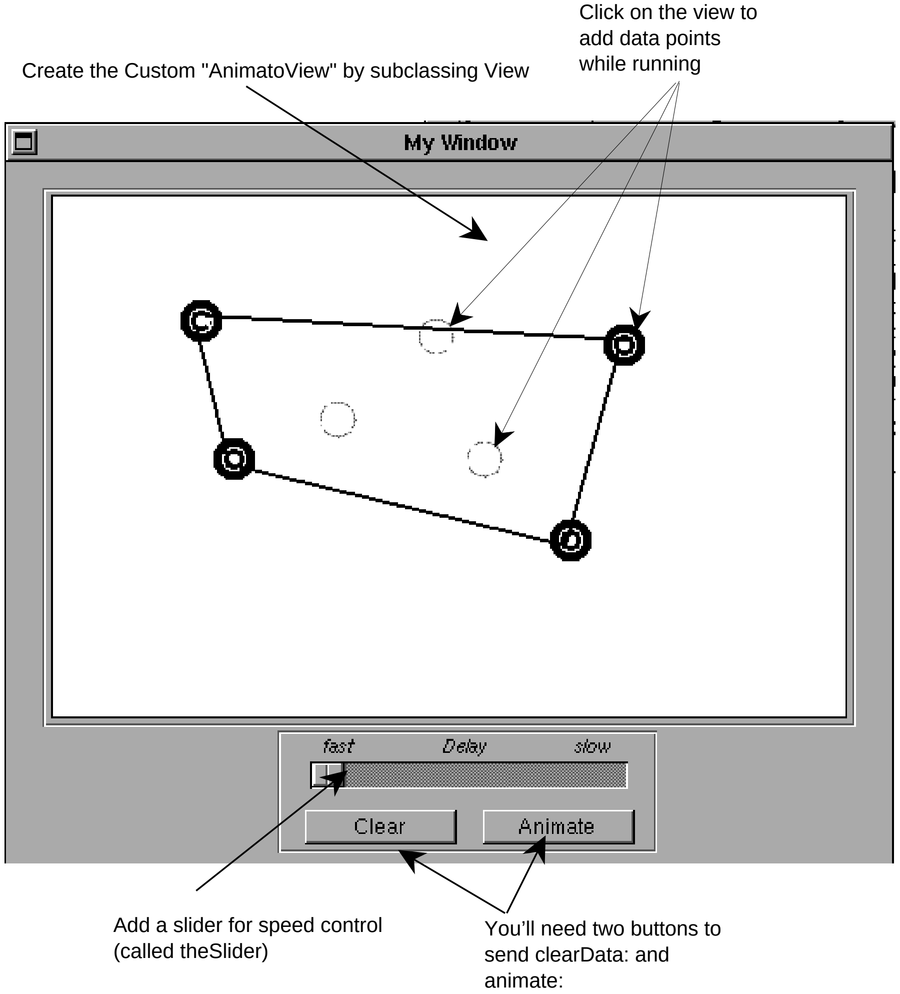
The animation itself went mostly like a breeze. Problems I encountered :

- When drawing the "currently inspected point triple" which is displayed with a centrally drawn gray-point, and two thick examination lines., I found I was inadvertently erasing parts of the "point-circles".
- A second problem is boundary conditions. Of course, the first and final points are members of the hull and should be drawn, but when do we let the user know that they've been included? I decided to set the large black circle of a selected state on the first node before running the algorithm and the last node afterwards.
- A third problem, related to this, was screen-refresh. Although I tried using a fresh full-screen redraw of the points before animating (just in case the user added some extra points before clearing), the time to redisplay was both noticeable and annoying. More annoying, I may add than watching a *significantly* longer animation. I found that the patience of the viewer was directly related to his interest in the ongoing action. Actions perceived as *system overhead* were especially negative.

On the positive side, seeing algorithms run gives a concrete feel for the manipulated representation. Graph theory and computational geometry problems seem especially well suited to this approach. With a minimum of code change, it would be easy to implement a set-diameter algorithm or a max/min searcher. I hope you find the code easy to follow and try.

Node.h
```
#import <appkit/appkit.h>
#import <objc/Object.h>
@interface Node:Object
{
   NXPoint pt;
}

+new:(int)theX:(int)theY;
- (int) x;
- (int) y;
- (NXPoint) Pt;

@end
```

Node.m
```
#import "Node.h"

@implementation Node

+new:(int)theX:(int)theY
{
   self = [super new];
   pt.x = theX;
   pt.y = theY;
   return self;
}

- (int) x
{
   return pt.x;
}

- (int) y
{
   return pt.y;
}

- (NXPoint) Pt
{
   return pt;
}
```

AnimatoView.h

```
#import <appkit/appkit.h>
#import <objc/Object.h>
#import <objc/List.h>
#import "Node.h"

@interface AnimatoView:View
{
      id dataPoints;
      id theSlider;
}
+ newFrame:(const NXRect *)tF;
- drawSelf:(NXRect *)tR:(int)c;
- mouseDown:(NXEvent *)ptr;
- clearData:sender;
- animate:sender;
- setTheSlider:anObject;
- (BOOL) turn:Hull:(int)i;
- connect:(NXPoint)pt1:(NXPoint)pt2:(float)color:(float)offset;
- linScan:(NXPoint)pt:(float)radius:(float)width:(float)color;
@end
```

AnimatoView.m

```
#import "AnimatoView.h"

@implementation AnimatoView

// Frame Creation for custom View
+ newFrame:(const NXRect *)tF
{
   self = [super newFrame:tF];
   dataPoints = [List new];
   return self;
}

// initialize speed slider
- setTheSlider:anObject
{
   theSlider = anObject;
   return self;
}
```



```
// Initialize the Frame View
- drawSelf:(NXRect *)tR:(int)c
{
   NXEraseRect(&bounds);
   PSsetgray(0.0);
   NXFrameRect(&bounds);
   return self;
}

// Deal with User Interaction
- mouseDown:(NXEvent *)ptr
{
   NXPoint mLoc;
   NXEvent *nextEvent;
   id theNode;
   int i;

   [self lockFocus];
   PSsetgray(0.50);
   PSsetlinewidth(1.0);
   mLoc = ptr->location;
   [self convertPoint:&mLoc fromView:nil];
   PSarc(mLoc.x, mLoc.y, 10.0, 0.0, 360.0);
   PSstroke();
   [window flushWindow];
   [self unlockFocus];
   theNode = [Node new:mLoc.x:mLoc.y];
   i = 0;
   // insert new data point into the list, in sorted order.
   while ((i < [dataPoints count]) &&
         ([[dataPoints objectAt:i] x] < mLoc.x)) i++;
   if (([[dataPoints objectAt:i] x] != mLoc.x) ||
      ([[dataPoints objectAt:i] y] != mLoc.y))
      [dataPoints insertObject:theNode at:i];
   return self;
}
```

```
// Clear Data at request of User
- clearData:sender
{
   [dataPoints freeObjects];
   [dataPoints free];
   dataPoints = [List new];
   [self lockFocus];
   NXEraseRect(&bounds);
   NXFrameRect(&bounds);
   [self unlockFocus];
   return self;
}

// Draw the Point during a scan
- linScan:(NXPoint)pt:(float)radius:(float)width:(float)color;
{
   [self lockFocus];
   PSsetgray(color);
   PSsetlinewidth(width);
   PSarc(pt.x, pt.y, radius, 0.0, 360.0);
   PSstroke();
   [window flushWindow];
   [self unlockFocus];
   return self;
}

// Draw a Line connecting two points
- connect:(NXPoint)pt1:(NXPoint)pt2:(float)color:(float)offset;
{
   [self lockFocus];
   PSsetgray(color);
   PSsetlinewidth(2.0);
   PSmoveto(pt1.x+offset, pt1.y+offset);
   PSlineto(pt2.x+offset, pt2.y+offset);
   PSstroke();
   [window flushWindow];
   [self unlockFocus];
   return self;
}
```



```
// Determine Hull Convexity or Concavity
- (BOOL) turn:Hull:(int)i;
{
      NXPoint p1, p2, p3, v1, v2;
      int riz;

      p1 = [[Hull objectAt:(i -1)] Pt];
      p2 = [[Hull objectAt:(i)] Pt];
      p3 = [[Hull objectAt:(i +1)] Pt];

      v1.x = p1.x - p2.x;
      v1.y = p1.y - p2.y;

      v2.x = p3.x - p2.x;
      v2.y = p3.y - p2.y;

      riz = v1.x*v2.y - v1.y*v2.x;

      if  (riz > 0.0) return TRUE; else return FALSE;
}

// The Algorithm and the Animation for Convex Hull algo.
- animate:sender
{
   id curr, max, min;
   int i,j, speed;
   id UHull, LHull;

   if ([dataPoints count] == 0)
   {
      NXRunAlertPanel("Convex Hull Animator", "No Data Points",
      "OK", NULL, NULL);
      return self;
   }

   if ([dataPoints count] == 1)
   {
     NXRunAlertPanel("Convex Hull Animator",
      "Single Data Point \nResult is the Point",
      "OK",NULL,NULL);
     return self;
   }
```

```
if ([dataPoints count] == 2)
   {
      NXRunAlertPanel("Convex Hull Animator",
      "Two Data Points\nResult is Line Segment",
      "OK", NULL, NULL);
      return self;
   }

   // turn off mouse
   [NXWait push];

   // set Speed
   speed = [theSlider intValue];

   curr = [dataPoints objectAt:0];
   UHull = [List new];
   LHull = [List new];
   [UHull addObject:curr];
   [LHull addObject:curr];


   for (i = 1; i < [dataPoints count]; i++)
   {
      if ([[dataPoints objectAt:i] x] > [curr x])
      {
         curr = [dataPoints objectAt:i];
         max = min = curr;
         [UHull addObject:curr];
         [LHull addObject:curr];
      }
      else // strictly ==
      {
         if  ([[dataPoints objectAt:i] y] > [max y])
         {
            [UHull replaceObjectAt:([UHull count] - 1)
               with:[dataPoints objectAt:i]];
            max = [dataPoints objectAt:i];
         }
         if  ([[dataPoints objectAt:i] y] < [min y])
         {
            [LHull replaceObjectAt:([LHull count] - 1)
               with:[dataPoints objectAt:i]];
            min = [dataPoints objectAt:i];
         }
      }
   }
```



```
if (([UHull count] < 3) || ([LHull count] < 3))
{
   NXRunAlertPanel("Convex Hull Animator",
      "Colinear Points Preclude Convex Hull",
      "OK", NULL, NULL);
   return self;
}

[self linScan:[[UHull objectAt:0] Pt]:5.0:3.0:0.0];
i = 1;
while (i < ([UHull count]-1))
{
   for (j = 0; j < speed; j++)
   {
      [self connect:[[UHull objectAt:(i-1)] Pt]:
         [[UHull objectAt:i] Pt]:1.0:3.0];
      [self connect:[[UHull objectAt:(i)] Pt]:
         [[UHull objectAt:(i+1)] Pt]:1.0:3.0];
      [self linScan:[[UHull objectAt:i] Pt]:5.0:3.0:1.0];
      [self linScan:[[UHull objectAt:i] Pt]:5.0:3.0:0.5];
      [self connect:[[UHull objectAt:(i-1)] Pt]:
         [[UHull objectAt:i] Pt]:0.3:3.0];
      [self connect:[[UHull objectAt:(i)] Pt]:
         [[UHull objectAt:(i+1)] Pt]:0.3:3.0];
   }

   if ([self turn:UHull:i])
   {
      [self linScan:[[UHull objectAt:i] Pt]:5.0:3.0:0.0];
      i++;
   }
   else
   {
      [self linScan:[[UHull objectAt:i] Pt]:5.0:3.0:1.0];
      [self connect:[[UHull objectAt:(i-1)] Pt]:
         [[UHull objectAt:i] Pt]:1.0:3.0];
      [self connect:[[UHull objectAt:(i)] Pt]:
         [[UHull objectAt:(i+1)] Pt]:1.0:3.0];
      [self connect:[[UHull objectAt:(i-1)] Pt]:
         [[UHull objectAt:(i+1)] Pt]:0.0:3.0];
      [UHull removeObjectAt:i];
      if (i > 1) i = i - 1;
   }
}
```

```
[self linScan:[[UHull objectAt:([UHull count]-1)] Pt]:5.0:3.0:0.0];

i = 1;
while (i < ([LHull count]-1))
{
   for (j = 0; j < speed; j++)
   {
      [self connect:[[LHull objectAt:(i-1)] Pt]:
      [[LHull objectAt:i] Pt]:1.0:-3.0];
      [self connect:[[LHull objectAt:(i)] Pt]:
         [[LHull objectAt:(i+1)] Pt]:1.0:-3.0];
      [self linScan:[[LHull objectAt:i] Pt]:5.0:3.0:1.0];
      [self linScan:[[LHull objectAt:i] Pt]:5.0:3.0:0.5];
      [self connect:[[LHull objectAt:(i-1)] Pt]:
         [[LHull objectAt:i] Pt]:0.3:-3.0];
      [self connect:[[LHull objectAt:(i)] Pt]:
         [[LHull objectAt:(i+1)] Pt]:0.3:-3.0];
   }

   if (![self turn:LHull:i])
   {
      [self linScan:[[LHull objectAt:i] Pt]:5.0:3.0:0.0];
      i++;
   }
   else
   {
      [self linScan:[[LHull objectAt:i] Pt]:5.0:3.0:1.0];
      [self connect:[[LHull objectAt:(i-1)] Pt]:
         [[LHull objectAt:i] Pt]:1.0:-3.0];
      [self connect:[[LHull objectAt:(i)] Pt]:
         [[LHull objectAt:(i+1)] Pt]:1.0:-3.0];
      [self connect:[[LHull objectAt:(i-1)] Pt]:
         [[LHull objectAt:(i+1)] Pt]:0.0:-3.0];
      [LHull removeObjectAt:i];
      if (i > 1) i = i - 1;
   }
}
```



```
// Just redraw all the points and lines  to look nicely here, in Black!

   [self linScan:[[UHull objectAt:0] Pt]:10.0:5.0:0.0];
   for (i = 1; i < [UHull count]; i++)
   {
      [self linScan:[[UHull objectAt:i] Pt]:10.0:5.0:0.0];
      [self connect:[[UHull objectAt:(i-1)] Pt]:
         [[UHull objectAt:i] Pt]:0.0:3.0];
   }

   for (i = 1; i < [LHull count]; i++)
   {
      [self linScan:[[LHull objectAt:i] Pt]:10.0:5.0:0.0];
      [self connect:[[LHull objectAt:(i-1)] Pt]:
         [[LHull objectAt:i] Pt]:0.0:-3.0];
   }

   // restore cursor
   NXPing();
   [NXWait pop];
   return self;
}

@end
```

Affairs of State: the Interface Builder and Custom Objects

David Stutz

To an inveterate software-is-the-solution person such as myself, one of the most exciting capabilities of the NeXT computer is the ease with which I can build extremely "soft" interfaces. Rather than making final decisions on the "look and feel" of an application at design time, I can defer all of these nits until application buildtime, or even until runtime, depending on the expertise and preferences of the application's final users. The mechanisms that support this totally irresponsible shirking of responsibility are not black magic; runtime message binding and inheritance play the largest part and are easily understood.

Designing the objects which populate your applications to use these basic mechanisms in the same way that NextStep objects do. After all, decisions about the placement and function of onscreen control elements are only the beginning when it comes to application mutability. The cube is a foot square box of objects, any of which can be connected together (within reason) by either interprocess messaging or by the use of the Interface Builder. In the same way that interface elements can be decoupled from the algorithms which they represent on screen and made available for re-use, any arbitrary object that you create, whether or not it draws itself on the screen, can be made available as an independent resource at runtime or in the Interface Builder. By using the same conventions as NeXT does, your own objects can be easily and almost imperceptibly integrated into NextStep. Tremendous gains in productivity maintainability and mental health result.

The base software technologies that come bundled with the cube are all easily extended; Display PostScript, Common Lisp, Sybase, and Mathematica all run as interpreters in which one can define language extensions which take effect immediately! Mach itself is based on an extensible set of "system resources" which are typically available as runtime services which can be added and removed on-the-fly (such as device drivers or virtual memory pagers). Less commonly known is the fact the the Interface Builder is also extendible in its own terms; by using the Interface Builder, anyone can define new palettes, inspectors, custom objects, and initial default objects that will then appear as parts of the IB.

As an immediate and visceral example, the IB is probably the quickest validation of the benefits of a runtime connection-based world. Whip up an application, put the IB in test mode, and try it. Reconfigure and try again. The immediate availability of the constituent appkit components for a given application dramatically improves the traditional compile cycle when it comes to building and testing. The only problem is that the objects doing most of the work, the custom objects containing your algorithms, aren't part of IB. When you throw the test switch, custom objects don't test!

Fortunately, this is easily fixed. Charles and Judy describe a basic method to extend your palettes in another article; what I'll do here is describe why its easy to fix, and how to add inspector capabilities to your custom IB.

The Wonders of Buildtime

Interface Builder is the primary NeXT buildtime tool for our User Interface Management System (UIMS). IB provides an easy-to-use environment in which several applications can be prototyped and tested, independent of their underlying object implementations. Rather than forcing the developer to create new communication protocols for each new object family, IB uses a few simple protocols, expecting to apply them universally.

What is buildtime? Buildtime refers to a bonus time dimension provided to you, a developer of objects, by the UIMS. Having created object descriptions at compile time, you must eventually populate an application with instantiations of these.. Rather than writing code (the old fashioned way), you can instantiate an object collection by pulling them off IB palettes or by creating them via the class browser. Plug together and configure objects using both target/action and notification delegation, and finally, test the aggregate repeatedly. This is buildtime.

Buildtime could not exist were it not for NextStep support for dynamic message lookup in objects and message passing between running processes. The runtime system supports a complete binding spectrum[1] for software component objects. This lets you use interactive tools such as IB to design and build applications. The ability of an object to archive and unarchive itself using read: and write: calls is also used heavily.

What happens when you pull an IB palette object into your application?. This object, including its attributes and sub-objects (remember the Font MenuItem), is archived to memory and then immediately unarchived. This has copies the object content of the palette object into an identical object which is made part of the IB working collection. Given a new object like this, how do you make it function in the overall scheme of the new application? Most commonly, you draw a line from that object (if it's a control) to another object; this sets the object's target to the object pointed to. You select an action from the list of potential action methods, and punch the connect button. Alternatively, you could draw the line from a previously instantiated object to your new object for outlet selection.

Linkage information such as target ids and action selectors is generally applied to objects in the current working collection as it is created. This results in an application testable at any point in the design process. Objects not in the IB are represented by proxy; their link information will be loaded at runtime. And of course, all of this information is pumped into a nibfile when you select the Save menu item.

What does a nibfile really contain? A simple, incomplete answer is instantiated object state information and data needed to recreate linkages between them. Both target/action and notification-delegation links depend on the existence of a *set* method (setTarget:, setDelegate:, or setYourInstanceVariable:) used to initialize links at runtime from the nibfile. These also rely on Objective-C's capability to convert a string or an @selector into a method reference and perform: that method at runtime. (For example, think of how a Window knows to send a windowDidClose: message to a delegate that was created as a custom object in the Interface Builder.) Nibfiles are full of stuff that you'd rather not program manually.

[1] See Jon Hopkins' excellent description of binding in Objective-C in the January/Februrary issue of the Journal of Object-Oriented Programming.

Custom Interface Builders

If you've gone the Digital Librarian route with your technical documentation, you may have come across the following line in the Interface Builder tutorial:

<<The version of Interface Builder included in the 1.0 release doesn't fully support customization. See the Interface Builder release notes for more details.>>

If you were familiar with hypermedia, you immediately followed (manually) this link and located the following in the 1.0 Release Notes:

As you might suspect, Interface Builder's visual interface is itself derived from interface files: Interface Builder is built using Interface Builder. The displays you see in the Palettes window, for example, are unarchived from interface files. In customizing Interface Builder, you create another interface file, which is added to those used to construct the Palettes window. The new interface file defines a new display in the Palettes window.

¥

In the 1.0 implementation, several restrictions concern customization. First, you can only add View objects. If your added View object has instance variables you want written to the interface file, you'll have to write read: and write: methods in its class implementation file. Second, IB won't understand the custom palette specification data that your *customized* Interface Builder version will be able to read and write. A later release of Interface Builder will allow dynamic loading of classes, eliminating this problem. Because of these restrictions, it's best at this time to customize Interface Builder only for demonstration, experimentation, and course work—not for serious development.

The 1.0 documentation continues with step-by-step instructions on creating a custom palette, and is recommended reading. Does this mean that customized Interface Builders don't really work? NO! its quite possible to do real development using a customized Interface Builder, although to do it, you need to know a bit about UNIX and Makefiles. So, in the spirit of experimentation, let's customize the IB and talk about the repercussions of what's going on.

For this example, I will to add a palette to IB containing only one object, the notorious Freeman/Oei blinking TextField. This object is a subclass of TextField that has (basically) one added feature: it can be made to blink with a regular frequency, thanks to a timer that is managed by the Display PostScript server. Here is its header information:

```
@interface BlinkTextField:TextField

{
  float realGray;
  double interval;
  BOOL active;
  DPSTimedEntry teNum;
  int priority ;
  int which;
}
```

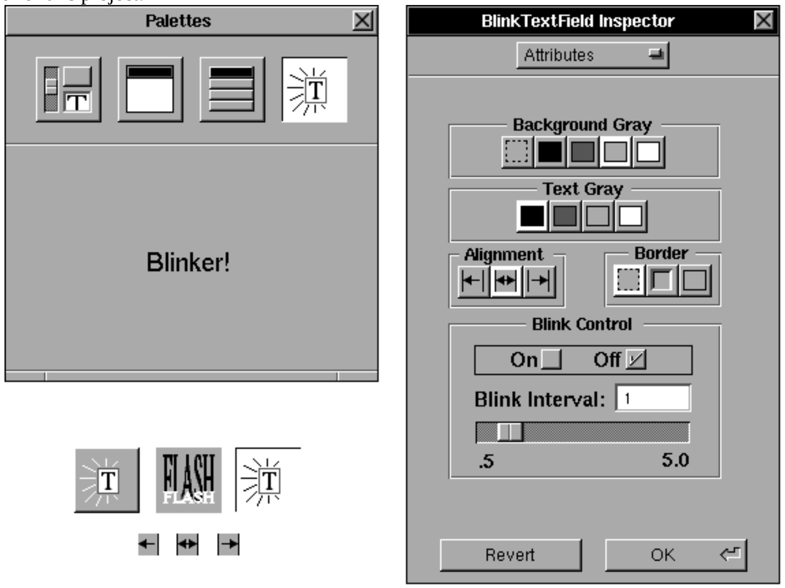


```
+ newFrame:(NXRect*)r;

- setTextGray:(float)g;
- flashGray;
- (float)realGray;

- setEditable:(BOOL)flag;
- setSelectable:(BOOL)flag;
- setBlink:(BOOL)flag;/* YES starts,NO stops */
- deferredSetBlink:(BOOL)flag;
- setBlinkRate:(double)theRate;/* seconds per blink */

- (double)blinkRate;
- (int)isBlinking;
- (double)blinkRate;
- (int)isBlinking;

- read:(NXTypedStream*)s;
- write:(NXTypedStream*)s;
- free;

- (const char*)inspectorName;

@end
```

Why is this a good example? Well, its a subclass, and that's definitely the right way to approach the world. It's also a "live" object, one that depends on external services to work. (In this case the Display PostScript server provides timer services.) This shows that the Interface Builder is not just a screen painter; when the test switch is thrown, the application becomes live. That service might just as well be Mathematica, Sybase, or any server on the net! I'm not going to discuss the actual mechanisms behind the BlinkTextField's blinking behavior, but I'd encourage you to peruse it for a good example of how to use asynchronous system services.

On the downside, this example has an awful lot of methods to write for such a simple functional change, doesn't it? (But wait, there's more...we haven't even looked at the Inspector subclass yet!) Fortunately, the benefits of having your own objects as testable citizens of the Interface Builder will greatly outweigh the small amount of additional coding that is necessary. Also, the additional code is often useful in its own right, since it provides support for typed streams and other slightly esoteric, but very useful software mechanisms that exist in NextStep.

Before I examine a few of the more important methods in the BlinkTextField, with an eye towards why they are necessary for a custom palette or a custom inspector, let's look at what the palette and the inspector will look like, along with some of the TIFF resources used for this project:



Although the details of placing your object in the palette are covered elsewhere in this issue of Buzzings and in the NeXT Release Notes, it's worth reviewing the two critical features of the BlinkTextField that actually make it palettizable. First, it is a subclass of TextField, which itself is descended from View. You can incorporate an object that is not directly related to View into a custom Interface Builder, but only as an instantiable class in the class browser. Because my custom object is descended from View, the factory method newFrame: must be implemented. This is what will get called to create the initial object in the palette. In the case of BlinkTextField, quite a bit of mundane initialization code goes into this method.

The second, critical feature of BlinkTextField is its matched pair of read: and write: methods. These methods contain code to save any state that I'd like to see reappear. They also contain runtime adjustments to the object, in this case a call to setBlink: which registers the object with Display PostScript. There are alternative locations for this kind of code in Objective-C, so you should examine your options before putting logic into your read: and write: methods. Specifically, awake and initialize are sometimes the appropriate methods to use for initialization. Here are these two methods:


```
- read:(NXTypedStream*)s
{
  [super read:s];
  NXReadTypes(s,"fdii", &realGray, &interval, &active, &priority);
  if (active)
    [self setBlink:YES];
  return self;
}

- write:(NXTypedStream*)s
{
  [super write:s];
  NXWriteTypes(s,"fdii", &realGray, &interval, &active,&priority);
  return self;
}
```

Having successfully palettized your objects, you may now want to go one step further and add an inspector for them. Before proceeding, realize that the following procedure is *totally* undocumented. Although this will work with the 1.0 release of NextStep, there is absolutely **no guarantee** that this will work in future releases. I am not acting as a representative of NeXT in publishing this! It is, however, a nifty hack and can be very useful to anyone who is developing applications based on their own custom objects (which should include everyone who writes software on the cube).

Notice the last method in BlinkTextObject, a method called inspectorName. Here is the code for this method:

```
- (const char*)inspectorName
{
  return "BTFInspector";
}
```

It turns out that I have another class description in my custom IB project, a class by the name of BTFInspector. BTFInspector is a subclass of the Inspector class, described in a file called /usr/lib/nib/InterfaceBuilder.h which is the base class for all inspectors used in IB. When using the Interface Builder I decide to inspect an object, the selected object is asked for its inspector's name. This name is then apparently turned into a class name, and the class name is converted into an instance of the appropriate inspector subclass. So, in order to produce an inspector, all I have to do is to define a subclass of inspector for my custom class and provide a matching inspectorName method. The Interface Builder will do the rest.

Of course, there are a few rules to follow when defining an inspector class. These should become clear by examining the BTFInspector class:

```
@interface BTFInspector:Inspector
{
    idintervalSlider;
    idalignMatrix;
    idtextGrayMatrix;
    idborderMatrix;
    idbackGrayMatrix;
    idintervalField;
    idblinkOnOff;
}

+ new;

- setIntervalSlider:anObject;
- setAlignMatrix:anObject;
- setTextGrayMatrix:anObject;
- setBorderMatrix:anObject;
- setBackGrayMatrix:anObject;
- setIntervalField:anObject;
- setBlinkOnOff:anObject;

- takeBlinkRateAndOK:sender;
- ok:sender;
- revert:sender;
```

There are a number of instance variables in this class, all of which refer to controls located inside the inspector panel. These controls must be queried whenever the OK or Cancel buttons are depressed in the inspector, so that the inspected object can remain correctly synchronized. The OK button actually sends the ok: message to the inspector object, while the Cancel button sends revert:. Within the context of an inspector subclass, the currently selected object can be referred to as object, for example
`[object setTextGray:NX_WHITE]`.

The critical methods in the BTFInspector class are new, ok:, and revert:. Here is the code for these three methods:

```
+ new
{
  self = [super new];
  [NXApp loadNibSection:"BTFInspector.nib" owner:self];
  return self;
}
```



```
- ok:sender
{
  int t;

  [window endEditingFor:self];

  if ((t = [[backGrayMatrix selectedCell] tag]) == 0) {
    [object setBackgroundGray:-1.0];
  } else {
    [object setBackgroundGray:(t - 1) / 3.0];
  }

  [object setTextGray:[[textGrayMatrix selectedCell] tag] / 3.0];

  if ((t = [[borderMatrix selectedCell] tag]) == 0) {
    [object setBezeled:NO];
    [object setBordered:NO];
  } else if (t == 1) {
    [object setBezeled:YES];
    [object setBordered:NO];
  } else {
    [object setBezeled:NO];
    [object setBordered:YES];
  }

  if ((t = [[alignMatrix selectedCell] tag]) == 0) {
    [object setAlignment:NX_LEFTALIGNED];
  } else if (t == 1) {
    [object setAlignment:NX_CENTERED];
  } else {
    [object setAlignment:NX_RIGHTALIGNED];
  }

  [object setBlinkRate:[intervalField doubleValue]];
  if ([[blinkOnOff selectedCell] tag] == 0) {
    // we want the field to blink when its read into test mode
    [object deferredSetBlink:YES];
  } else {
    [object deferredSetBlink:NO];
  }

  return [super ok:sender];
}
```

```
- revert:sender
{
  [window endEditingFor:self];

  if ([object backgroundGray] < 0.0) {
    [backGrayMatrix selectCellAt:0 :0];
  } else {
    [backGrayMatrix selectCellAt:0:
         ([object backgroundGray] * 3.0)+1];
  }
  [textGrayMatrix selectCellAt:0 :([object textGray] * 3.0)];

  if ([object isBordered]) {
    [borderMatrix selectCellAt:0 :2];
  }
  else if ([object isBezeled]) {
    [borderMatrix selectCellAt:0 :1];
  }
  else {
    [borderMatrix selectCellAt:0 :0];
  }

  if ([object alignment] == NX_LEFTALIGNED) {
    [alignMatrix selectCellAt:0 :0];
  } else if ([object alignment] == NX_CENTERED) {
    [alignMatrix selectCellAt:0 :1];
  } else if ([object alignment] == NX_RIGHTALIGNED) {
    [alignMatrix selectCellAt:0 :2];
  }

  [intervalField setDoubleValue:[object blinkRate]];
  [intervalSlider setDoubleValue:[object blinkRate]];
  if ([object isBlinking]) {
    [blinkOnOff selectCellAt:0 :0];
  } else {
    [blinkOnOff selectCellAt:0 :1];
  }

  return [super revert:sender];
}
```


The new method loads a nibfile that describes the inspector and hooks all of the controls to their outlet instance variables. This nibfile contains a description for a non-deferred Panel object which should be roughly the size of the "Inspector" Panel in the Interface Builder. The only caveat here is that the nibfile should not include an OK or a Cancel button, because these will be provided by the superclass Inspector. (The contents of the Panel that is described by the nibfile are moved into the Interface Builder's already existing "Inspector" Panel.) The owner of the nibfile must be a BTFInspector object, and that object's window instance variable (inherited from Inspector) must be connected to the Panel object which contains its guts.

Ok: and revert: are straightforward, if long-winded. The important points to note are the ability to refer to "object," thanks to the Inspector superclass, and the return statements which call the superclass' corresponding method. The many lines of code simply set or reset specific attributes of the selected BlinkTextField object.

Interface Builder is NOT a debugger, and while you can use it to poke and prod a stable object, determining its limits, its primary benefit is as an object connection editor. This point is important, and deserves to be discussed in more detail than space and time allow here; IB treats abstract objects in the same way as screen-viewable objects. For example, a Sound objects presents an interface in the IB in the same way as a Slider, through a list of outlets and target/action methods. You may easily extend the default set of abstract objects available in the Class Browser in the same way that we have extended the palettized objects. Connectable objects *do not* have to be subclasses of View!

Finis

Reviews, Rumors & Etc (a myopic eye into the NeXT Market)

Erica J. Liebman

Review Stuff

It's always a delight to see new hardware and software. On a new machine, the question seems to always be "Sure it has great hardware, but what software is out there for it?" I know; you've heard this too. This month, the Buzz team was especially lucky. We had a chance to look at a number of new products including Digital Ears, the Communicae Demo and Spring, an educational package. We hope to have extended reviews of these products in later issues.

Ian Smith, bless him, brought in his entire stereo system to help test out ***Digital Ears.*** He, Keith Edwards and I sat down for several hours of CD-heaven. Hardware installation was a snap. Plug the little black box to the big black box, plug the CD player into the little black box. Since we had a demo version, we only got to run the simplest recorder and o-scope applications. We digitized at CODEC, 22 and 44 Khz. The sound quality was staggeringly good. The swap file size became staggering also. Especially since a 44 Khz/4-minute song takes about 48 megabytes of disk space. We did have a number of fatal crashes. When things got bad, they got real bad.

Each sampling rate was audibly better than the former, even to an audience of "wood ears". I found a number of code snippets for using Digital Ears routines in custom programs. I found this particularly exciting and wished I could find further documentation on these features. I didn't find any specific FFT routines, but am sure they must be supported. If someone has more information about this, please drop me a line. Digital Ears is not cheap, but I know a lot of people -- of artistic and electrical engineering bent -- who will be interested in both the sound and data acquisition features of this product.

Brian Hess of Active Ingredients got me a copy of the ***Communicae*** demo. This is one large, spiffy program, although I found it to be little daunting and confusing at first. The real target audience seems to be people who have an installed base of software requiring access to VT220 and Tektronix emulation -- probably industry, laboratories and government. The demos for each of these modes appeared exhaustive and complete. This is probably not a program for the person who just wants to occasionally use a modem.

I quickly found out what Paul Nevai told me in this letter, the support was >excellent< :

> ***Capsule Review****: Communicae is basically the cube version of the MacVersaTerm Pro. A superset of Shell and Terminal. Can print selectedtext => great feature. Can do emacs, paste, save etc.*
> ***Costs*** *$399 - yuck! They include optical disk but who needs it when the university cost of such a disk is 50 bucks. I' rather pay $100 and FTP it.*
> ***Bottom Line****: Superior Software and superior technical support.*
> *-- Paul Nevai*

Jeanette Allen helped test-drive ***Spring***, a physics tutorial about spring motion. This program seems aimed at the high-school or college physics student. Although the program was slow at times, it really seemed to convey the lesson well. Most importantly, it encouraged experimentation. (What happens when I increase the mass on the spring? The period lengthens. What if I make it a negative mass? The program explains through voice messages that this is not allowed.) A very good feature was the multiple window views and simultaneous graphing.

Jeanette's biggest complaint was the way text-field entry was handled. (Required returns, but no mention made to naive users) which can give the false impression that the program has been left in an inconsistent state. Also, while the Y-axis on the graph is set by a slider, the X-axis was set by a pair of data entry values. It took us a while to figure out how that strange Y-vertical axis worked. I found the erratic speeds of computation to be distracting.

We both thought that this was a wonderful experimentation system. Although I eventually went off to my College physics book to read more, the user really doesn't need any knowledge of the underlying equations to get a feel for the mechanics of spring motion.

Summary

Digital Ears : Utterly cool, software bombed an awful lot, sound quality was superb.

Communicae : A bit confusing and non-NeXTish at first, but powerful as can be, well supported (thanks Brian!) It really seems to do the job.

Spring : Gets the idea of spring-motion physics across well. Easily used by non-NeXTperts. Multiple views of animations is its best feature.

Rumor Stuff & Etc.

• A reliable source tells me that Ashton Tate is finishing a Spreadsheet that will both "sell the cube to the business market" and "beat Wingz into a pulp". The secret to all this? Rather than simple porting, they are apparently re-writing from ground up, using NeXT philosophy with inspectors, etc. Don't look for this any sooner than July though.

• I understand that WordPerfect was so impressed with previews of this beast that it is planning to build it's word-processing entry from scratch. With the high- and low-ends covered in the word-processing realm (WriteNow and FrameMaker), I'm pleased to see a middle-market word-processing company step in.

• A group up in the North are planning to market a number of NeXT-accessories to help clean your cube, but you'll have to wait for more specific information as they asked me to hold off on details until plans for marketing were firmed.

• I know from a VERY reliable source that Lighthouse's Public Domain compilation disk has been finished and handed in to duplication. (OK, I admit it, I had a hand in this compilation. Buy it. Please.)

• Finally, I understand that a California Mail-Order house either is or soon will be selling external floppy drives as an alternative to Dayna. I'm trying to hunt down further information about costs, etc.

Market View

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.

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

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

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"

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"

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"

FrameMaker 2.0 *Frame Technology Corporation* 1-408-433-3311 "powerful, cost-effective workstation publishing software"

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"

TextArt *Stone Design Corporation* 1-505-345-4800 "array of tools that allow immediate creation of outstanding PostScript images"

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

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

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"

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" (I gave them a call in January. It looks like we may have a review of this for the April Issue.)

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"

Digtal 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"

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"

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"

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)"

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

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

User Groups

Here are some pointers to Users Groups that may be in your area. Send info if you don't see your group here.

Maryland/Northern Virginia/DC

•Washington Apple Pi, Hugh V. O'Neill, (301)-328-9510 , Chairperson
POB 39036, Washington, DC 20016

The purpose of the SIG is to exchange information from an end-user's viewpoint on the capabilities (present, planned, and potential) of the NeXT computer/information system. The goals and objectives include the following:

1. Presentation of relevant, timely, accurate and complete information on the performance characteristics, hardware, and software of the entire system.
2. Discussion of applications in various areas including: education, research, medicine, law, decision making/management, policy making/analysis, science, engineering, mathematics, statistics, humanities, arts, business, governmental activity (federal, state, and local), executive information systems, artificial intelligence (AI), operations research/management science, systems analysis, desktop publishing, office use and home use.
3. Cost-effectiveness/benefits/performance evaluation from the viewpoint of the end-user.
4. Special topics such as networks, information system security/integrity, speech recognition, signal recognition, array processing, digital signal processing, and sound/music.

The NeXT SIG usually meets at 7:30 p.m. on the 2nd Wednesday of the month at the National Institutes of Health (NIH) in Bethesda.

In addition, we have special joint meetings with other organization groups and SIGs at mutually acceptable times and places.
[Thanks Keith & Lisa for getting this to me! Hugh also mentioned on the phone that he's going to try to put together a Newsletter focussed on the Applications end-user.]

Georgia

•BuzzNUG is sponsoring local demos and talks. Contact Erica Liebman at erica@kong.gatech.edu. (404)-352-5551 Our next event is a Businessland Open House from 3-5 on 20 March 1989. Call me for directions. Dave Barry is our Businessland contact, and he promises some cubes, a linotronic printer and munchies. I'll be there for a good deal of the time. I am looking hard for a volunteer to help organize our local meetings. Please show up at the reception or give me a call if you think you can help.

Massachusetts

•The Boston Computer Society (BCS) has a NeXT special interest group. Contact Dan Lavin at 1-617-969-6555, or Jan McPeek at 1-617-926-4027. BCS's NeXT SIG has 750 members. Contact Dan at BCS 1 Center Plaza, Boston, Mass 02108. They've been meeting monthly since January 1989 and have a Bulletin Board and NewsLetter aimed at the Business Community. There are member fees. We'll probably have more info next month.

California

• Robert D. Nielsen 1-408-995-5775 coordinates **BANG!** out of San Jose, loosely associated with Stanford & San Jose State Univ. Robert is the new SJSU NeXT Consultant. (Atta-boy!) Definitely call if you live within fifty miles of San Jose. Or are willing to drive further. Or have your own 'copter. A newsletter is planned.
• Paul Lowe (714)787-3883 at the University of California at Riverside is interested in seeing what others are doing with their NeXT Cubes to distribute to the other NeXT Users (six so far) on campus. write to : plowe@ucrac1.ucr.edu

Texas *(yee-hah!)*

• The report goes that there "is some massive new NeXT user group down there". Says Jerry Goode : "Hi there! Just wanted to get back to you with some info about the NeXT user's group down here in Dallas, Texas...

First, we've had a grand total of 1 meeting with about 35 people in attendance. Metaresearch came down to speak and demo Digital Eye & Digital Ears. Pretty excellent stuff if you haven't seen it lately! Second, here is the name of the fellow who is heading up the group down here: Dirk Hardy/Hofbauer Information Systems/5080 Spectrum Drive/Suite 912W (Lock Box 21)/Dallas, Texas 75248/Phone: 214-385-2991

Hofbauer is a NeXT registered developer doing courseware authoring tools, and Dirk is working in conjunction with Dr. Ali from North Texas State U. to get the group going. I encourage you to get in touch with him - he's got a lot of creative ideas! Hope to have an email address for him soon.....

Finally, we plan to meet the third Thursday of every month at 7 pm, somewhere! The logistics are still a bit up in the air as we try to figure out how much this thing will grow. If the response after our first meeting is any indication, we could settle in at around 50 people.

By the way, I have had Buzzings forwarded to me and it's GREAT. Dirk has copies of both issues so far and was really impressed. You may want to talk with him about organizing a Texas contingent contribution as we try to get things rolling here. "

Washington (*the state*)

• University of Washington (with over 110 cubes) has a NeXT User group dating back to March 1989. Contact Corey Satten (corey@cac.washington.edu).

Canada (*O!)*

• Vancouver NeXT User's Group (in BC, CA) is headed up by Lionel Tolan (lionel_tolan@cc.sfu.ca).
• Tom Poiker of the University of British Columbia is starting a new newsletter NeXTVieW (name tentative) with Shirley Chan. (poiker@whistler.sfu.ca) Tom's particular interests are geographic imaging on the NeXT. [*Pretty cool, eh?]*

Ohio?

Gary T. Lang (well!glang@apple.com, mips!apple!well!glang@cis.ohio-state.edu) says he's trying to put together a newsletter aimed at the business market.

Scenes from the NeXT Issue

• We should be hearing more from Baker Corey about AI/X and NextStep.
• Bryce Jasmer got caught again the mad rush, but should get a description of the Oregon State archives in. Maybe Gerrit will have an update too.
• Hopefully we'll hear from Craig Shock and William Shipley.
• Pat McGee has a browser article in and Dick Silbar sent me something that I haven't had a chance to look at yet.
• We should be hearing about Bruce Webster, author of **The NeXT Book**, soon to be out in version 1.0, through an interview and a review. (Want to ask him something? Write soon to erica@kong.gatech.edu).

I haven't really had a chance to weed through letters this month, so expect some pleasant surprise for April.

Buzz's Hint Corner

• Here's a little expansion of a hint that appeared in the last issue. You can automate the process of extracting NeXT-formatted messages that appeared in the last hint corner with the following shell file :

```
#!/bin/csh
#
# extract the NeXT-encoded mail message in /tmp/NeXTmsg
#
cd /tmp
rm .tar*
uudecode NeXTmsg
set tname = .tar*
mv $tname NeXTmsg.Z
chmod 644 NeXTmsg.Z
mkdir d$tname
cd d$tname
zcat ../NeXTmsg.Z | tar xf -
open .
```

which does all the necessary translations, and pops up a browser on a directory containing the original message and enclosures. if you read mail with emacs rmail, you can go one step further and wire this to a single key by adding the following to your .emacs file (assuming the shell file above has been stored as ~/Unix/bin/parse-NeXT-mail).

```
(defun parse-NeXT-mail ()
  (interactive)
  (goto-char (point-min))
  (search-forward "\nbegin ")
  (write-region (- (point) 6) (point-max) "/tmp/NeXTmsg" nil 'quiet)
  (call-process "~/Unix/bin/parse-NeXT-mail"))
```

M. Dixon, Xerox Parc, mdixon@parc.xerox.com

• You can find out if you're on a NeXT via

```
strings /vmunix |grep NeXT > /dev/null
```

This will return an exit status of 0 if NeXT is found and 1 if not.

Elizabeth Hayes, Elizabeth got me this answer via NeXT technical support.

Buzz's Hint Corner, continued...

***Doug Kieslar** of NeXT has given a number of supplementary hints which you'll be seeing over the next few issues. These hints come from the most commonly asked support questions. Thanks Doug!*

Q: How can you add text to the end of a text object?
A: The trick is to make a zero length selection at the end of the text, and replace the text there. Your code might end up looking something like this:

```
docView = [myScroller docView];
length = [docView textLength];
[docView setSel:length :length];
[docView replaceSel:newfont];
[docView display];
```

Q: It seems like the scrollView ignores the frame.origin of its docView. Also, does a ScrollView clip to its update rect(s)?
A: A ScrollView translates the coordinate system of its contentView in response to user movement of the scrollers. Since the frame of the docView is defined in the coordinate system of the contentView (i.e. it's a subview of the contentView), translating the coordinate system of the contentView has the effect of moving the frame, and hence the visible portion of the docView.

When you do a setDocView: the system translates the coordinate system of the contentView so that the contentView's bounds.origin is the same as the frame.origin of the docView. For example: if the frame.origin of your docView is {100.,100.}, the bounds.origin of the contentView will be {100.,100.} as well.

So the ScrollView does pay attention to what the frame.origin of the docView is, but it doesn't really matter what it is, at least initially.

No, the drawing is not clipped to the update rects; however, you can restrict the area being redrawn yourself by drawSelf:: method you can use PSrectclip() or NXRectClip().

Q: I've installed an external disk, and the system won't boot now. What's wrong?
Q: What are the SCSI target numbers of disks NeXT ships from the factory?
Q: How do I change the SCSI target number of a Maxtor drive?
A: If you tell the system to boot from the SCSI disk, it looks for the LOWEST NUMBER SCSI target that understands booting, and attempts to boot from it. You can determine the SCSI target number of the boot device by watching the (early!) boot messages.

As we ship currently, the disk targets are:

330, 660-	*Target 1*
40 -	*Target 6*
CPU -	*Target 7*

Buzz’s Hint Corner, continued...

Through sometime late in 1989, some 330s and 660s were shipped as target 0. Some VERY VERY early systems might have the hard disk set as target 6. As we currently ship, it is now easy to add a disk which is the default boot device (set it as target 0), or to add a disk which does NOT supplant the on-board drive as the boot device (set is as target something-bigger-than-1).

To change the SCSI target of an internal Maxtor disk, you must remove the drive from the machine, and set its target jumpers. Look at the component side of the PC board on the bottom of the disk. If you look carefully about 2/3 of the way toward the connector end, you’ll see three jumpers (three pairs of pins) lined up perfectly in a 3x2 rectangle, and labeled JP35, JP36, and JP37 (or J35, J36, J37, depending on the drive). JP35 represents the low-order bit. To configure the target, use the following table

Target	***JP37***	***JP36***	***JP35***
0	*out*	*out*	*out*
1	*out*	*out*	*IN*
2	*out*	*IN*	*out*
3	*out*	*IN*	*IN*
4	*IN*	*out*	*out*
5	*IN*	*out*	*IN*
6	*IN*	*IN*	*out*
7	*IN*	*IN*	*IN*

(by "IN," I mean that the little blue plastic jumper gizmo is intalled between the appropriate pairs of jumper pins). By the way, if you set a SCSI device to target 7, the system won’t work very well. No, it is not possible to change the CPU’s target.

Note also that the assignment of the device special files (e.g., /dev/sd0*) to the actual drive is done dynamically at boot time. The drive with the lowest numbered SCSI target becomes drive 0, next is drive 1, etc.

Finis!