Article for March Issue of *Developer's Review*

Overview of NextStep

Over the past twelve months significant advances in hardware technology have caused higher education to reposition their definition of desktop computing. In recent periodicals and news stories the term "personal computer" is quickly being replaced by the term "workstation", which infers a more powerful and integrated computing environment for the academic user. This phenomenon is partially a result of an established base of personal computer users realizing that the current technology and tools they used to succeed in a specific goal, becomes the ground floor for evaluating all new technologies. Another reason for this repositioning is the need for hardware to handle more sophisticated tasks to meet the goals for academic computing in the 1990s.

In EDUCOM's book, Organizing and Managing Information Resources on Campus[1], Kenneth King outlines the need for academic hardware to provide 1) high bandwidth networking, 2) user interfaces which are intuitive, consistent, and standard; making use of high-resolution graphics and sound, 3) development environments of reusable frameworks (or objects), created by experts which will be modular, plugable, readable, connectable, and editable. These needs require horsepower which is not readily available on current generation personal computers.

The NeXT workstation is an example of an academic workstation that addresses these academic needs and has been described as a model of what academic workstations will look like in the next decade. The most important technology that NeXT demonstrates is the close integration of hardware power with a set of object based development tools called NextStep. NextStep allows for rapid development of powerful software applications which models itself after the smalltalk paradigm. This article will give an overview of NextStep and demonstrate some of the exciting capabilities of this development environment.

Introduction to NextStep

NextStep consists of four levels of software that fit between a UNIX[2] operating system and applications. These levels include (from the bottum up) the

[1] Brian L. Hawkins, ed. Organizing and Managing Information Resources on Campus (Academic Computing Publications, Inc., 1989).

[2] UNIX is a trademark of AT&T

PostScript window server, the Application Kit, the Interface Builder and the workspace manager. NextStep currently runs on NeXT hardware, IBM-RT, PS/2 as well and the new IBM RISC architectures.

New Terminology

Previous issues of *Developers Review* covered the essentials of object oriented programming. This included a discussion of Encapsulation, Inheritance and Messaging. It is essential to understand these terms for you to use NextStep effectively. One additional term that is used in NextStep is *run-time binding* or *dynamic binding*. This is the processes of making decisions on the destination of a message when the program is **run** rather than when the program is **compiled**. This means that you can change user interfaces to programs and integrate new objects without ever needing access to the source code of the objects in question. This promotes tool-kits rather than stand-alone programs.

Why is NextStep Revolutionary?

1) Because it is included in every NeXT system sold.

First, this allows every NeXT to become a developer if they choose. This will become more critical as the distinction between a creator of objects and a user of objects begins to blur. Bundling the development environment also allows a much higher level of common exchange between NeXT developers sharing objects. For example suppose your are a physics professor and you need an object that does a semi-log plot. Suppose there is professor down the hall that has already developed a semi-log plotting object. Because all plot objects simply sub-class the "View" class, sharing these objects is usually just a matter of dragging an icons from one directory to another and selecting the "parse" feature of Interface Builder.

2) Because it is the first connection based programming system with a graphic user interface.

Sending a message from one object to another is usually a matter of pointing to the source object and dragging a "rubber-band line" to the destination object. This is easier in some ways then procedural programming. It will perhaps replace "logo" as the an introduction to controlling computers.

3) Because it uses client/server models.

In the example below we will be creating a figure called a "blob". To create it we use a quick interactive editor (for example YAP, DisplayTalk or Preview) to interpret and draw the graphic we are building. It takes approximately 16 milliseconds to re-draw the object after any input is changed. This is one example of using a server (in this case the NeXT PostScript server) to service program requests. Other examples of this are the Mathematica and Sybase SQL server that are also bundled with every next system. In fact every NextStep program has the potential to be turned into a server by using standard system messages.

4) Because it adds run-time binding.

As mentioned earlier in the article, run-time binding moves decision making from the compile stage to the run stage. The benefit of promoting tool-kits of pre-compiled objects is the most important aspect of run-time binding. Another aspect is the ability to customize user interfaces to meet specific user needs or to extract objects from existing applications and to integrate them into new applications. Users find that dynamically linked systems are easier to integrate, personalize and customize.

5) Because it has a rich set of objects needed by most applications.

Although object oriented programming systems have been available on other micro-computer systems for years, there have been few that have ever been used to create commercial applications. This can be attributed to lack of performance and a minimal list of useful features. In contrast, every NextStep application program created that uses the graphical interface uses NextStep. It includes all of the standard objects typical programs need such as objects that change font, open files and send output to the printer.

6) Because it is one of the few graphic user interface system designed from the ground up based the principals of object oriented programming.

Object oriented compilers are currently in the process of being "grafted" into third party software systems that must be purchased in addition to a base computer system. It is often up to the user to then integrate the third party software system into the operating system to access system resources. After you add you new compiler to the system you then need to go shopping for a toolkit to build your applications. With NextStep all of the integration has been done for you. Because the compilers, debuggers and application kit objects are an integral part of the operating system the integration is seamless. Every menu, window, button and sound is already an object so the developer can use them just by pointing and clicking.

7) Because it attackes at the most significant problem facing software developers: the creation and maintenance of graphic user interfaces.

Studies done have shown that up to 90% of the time it takes to create easy to use programs that leverage off of graphic user interfaces is spent creating and maintaining the part of the program that interacts with the user. This is why a great deal of time was spent making the creation of user interfaces as easy as drawing.

8) Because it is very easy to use.

Ease of use can be expanded into several areas. They include easy to learn, easy to extend, and the power to do complex operation is a short amount of time and with a few lines of code. The example program below was created by about two dozen mouse events. The entire user interface was created without having to interact with the UNIX shell. After the template was created there were about a dozen lines of code inserted in to the objects and less then a dozen to create the drawing graphic. The entire program can easily be created in a few minutes.

9) Because it runs on a hardware platforms that will not restrict you.

Every system that runs NextStep also uses virtual memory. That means you don't have to worry about using complex memory management systems that will soon be obsolute. Every system that runs NextStep has a multi-tasking operations system. This means that servers can assumed to be running without leaving your application.

10) Because NextStep works in a distributed environment.

NextStep messages can move between processors on a network as easily as they can between two objects in single program. There are many examples of messaging to Mathematica on remote systems in the current release of NextStep. The most common is sound processing. A gas molecule hitting the edge of a gas chamber can send a message to the digital signal processor telling it to create a sound.

11) Because NextStep is language independant.

The glue that holds all NextStep objects together is the message statement. This message statement is language independant. So if you are using Objective C, LISP, FORTRAN, PostScript, Mathematica or DSP Assembler, all objects can work together.

Example: Building a Small Application

In the last issue of Developers Review we had an example Turbo Pascal program that created an object called a "blob". In this NextStep example we will draw that blob but also be able to rotate and magnify the object. The program is around a doze lines long and takes just a few minutes to create.

By follow instructions in figure 1 for creating the user interface to the blob program on the NeXT system we can see just how easy it is to build custom applications with NextStep. This example shows the power of Interface Builder as well as the power of PostScript. Many drawing environments would require ten to 100 times a many lines of code to do rotation and scaling. With PostScript these operations are just one line each. Another aspect of this program is the ability to print a view and to copy a view created by a progam into a word processor. Adding printing to this program is simply a matter of adding a "print" menu option and connecting it to the "printPScode" method of the view. Copying the the view to a word processor requires sending a message to the pasteboard object.

Future Releases

Although sharing objects is relatively easy in the current release, the ability to dynamicaly add objects to the Interface Builder takes considerable work. This will be a great deal easier in a future next release of NextStep. The ability to dynamically load palettes to the Interface Builder will use a new feature called dynamic loading. This allows you to create new binarys by adding toolkit binaries to existing tools. This will promote third party objects in a way never seen before.

Future articles in this newletter will demonstrate how to write small programs that interact with the Mathematica and Sybase SQL server as well as how to create your own servers.