Open Source Alternatives: Thriving with (Free) UNIX on the Engineering Desktop

Gary E. Rafe, Douglas W. Fraser
University of Pittsburgh / Lucent Technologies

Abstract

The wide-spread use of commercial UNIX operating systems to run desktop workstations and large-scale time-sharing and Internet server systems is well known, especially within the engineering-related academic community. In part due to the "one-size fits all" philosophy of the predominant operating system for desktop personal computers and the recent availability of relatively low-cost alternatives, an increasing amount of interest is being given to UNIX on the PC desktop. We offer some of our observations on the suitability of freely-available UNIX (and UNIX-like) systems for a variety of activities commonly associated with the personal computer domain, with particular emphasis on inter-operability across various system platforms. Included in our discussion are the increasingly-popular Linux, which can be used on many different types of computer hardware, FreeBSD, and Sun Microsystems' Solaris. In addition, we discuss AT&T's U/WIN system, which provides robust traditional UNIX services, and facilitates the use of many so-called open-source applications, on personal computers running Microsoft Windows.

Introduction

Our goal for this paper is to share some of our experiences and observations in the use of free UNIX and UNIX-like operating systems on desktop personal computers in technical computing environments, with particular emphasis in the area of engineering education. Perhaps due to the graphical nature of the user interface provided by the X Window System and the near ubiquitousness of audio hardware on desktop (and portable) computers, we find that these systems are well suited to multi-media applications. For the purposes of our discussion, we consider several classes of systems here, including: (1) systems that can be obtained without cost, such as *FreeBSD*,[1] derived from Berkeley Software Distribution (BSD) UNIX released by the Computer Systems Research Group at the University of California at Berkeley, and *Linux*,[2] a UNIX-like operating system based on a kernel developed by L. Torvalds; (2) commercial UNIX systems that are available at media costs to individuals for personal, non-commercial use, such as Sun Microsystems' *Solaris* (on SPARC- and Intel-based systems) and SCO's *UnixWare* (on Intel-based systems); (3) packages that run on existing Microsoft Windows systems such as AT&T's *U/WIN*[3] and Cygnus Solution's *Cygwin*,[4] also available at no cost to individuals for personal use in educational and research environments. We refer to these systems collectively in the remainder of our discussion as *UNIX*, except where we need to refer to a specific system or environment.

Our presentation is not intended to be a tutorial in the use of any one particular desktop UNIX environment. Rather, we are interested in the ability to accomplish our work on a variety of

systems with (more-or-less) the same paradigm, with the added benefit that the product of our work can be shared easily with others through the use of portable code and architecture-independent file formats.

Background

We bring to this presentation a long familiarity of UNIX systems, having nearly 30 years combined experience with them between us. Our first exposures to the UNIX system began in the mid-1980's when one of us (DWF) went to work for AT&T Bell Laboratories where AT&T UNIX Version 7 ran on a timesharing DEC PDP 11/70; access to this system was via 1200 baud dial-up modems and character display terminals. Such humble beginnings taught us the importance of treating bandwidth frugally, a lesson that continues with us to this day in our preference for mostly simple, character-based interfaces. In the next several years, the PDP was replaced by a VAX 750 running UNIX System V.2 with switched 9600 baud connections, which, in turn, was replaced by a workgroup of AT&T 3B2/1000 servers running UNIX System V.3. During this time, the other's (GER) department obtained a small multi-user AT&T 3B2/300 desktop system for use in FORTRAN programming courses running UNIX System V.3. Such was our affinity for the UNIX paradigm that our home personal computers at the time were the exotic AT&T UNIX PC7300 (Figure 1), which ran a variant of BSD UNIX.

Figure 1 AT&T UNIX PC7300

Since that time, we have used a wide range of commercial UNIX systems, including HP servers and workstations running HP/UX, IBM servers running AIX, generic desktop personal computers (PCs) running Microport *System V/386*, Sun i386 workstations running SunOS, Sun SPARC servers and workstations running SunOS 4.1 and Solaris 2, and SGI workstations running IRIX.

We conclude this summary by describing our various current UNIX desktops. At the University of Pittsburgh, we use a Sun Ultra 1 Creator3D workstation running the Solaris 7 operating system. This operating system is available to students, faculty members, and researchers for personal use for the cost of media and shipping from Sun Microsystems.[5] To build new binary

applications and programs from source code, we use a pre-compiled version of the GNU C/C++ compiler, **gcc**/**g++**.[6] At Lucent Technologies, our most recent (true) UNIX desktop was a Sun Ultra 30 workstation running Solaris 2.5.1; at present, we use Intel-based PCs running Microsoft Windows NT for our software development activities. The transition from the UNIX to the Windows paradigm was not painless. The recent availability of environments like AT&T's U/WIN package,[7] which includes the Korn shell (ksh93)[8] and over 200 standard UNIX utilities, has eased this transition greatly.

When not at our respective workplaces, one of us (GER) prefers the FreeBSD system[9] running on various notebook computers, while the other (DWF) has used SCO UnixWare 2.13, Solaris 2.5 x86, and Caldera OpenLinux 2.3 on a desktop system with varying degrees of success. An important aspect of the FreeBSD and UnixWare systems are their ability to run programs compiled for Linux systems through compatibility packages; this allows us to use the growing number of commercial applications being released for Linux (*e.g.*, Netscape Communicator and Navigator, Corel WordPerfect, RealPlayer, *etc.*) on our preferred systems.

In the remaining sections, we survey a broad range of applications, tools, and utilities that allow us consistent and flexible user interfaces across the various systems with which we work.

Cross-Platform Applications

User Interfaces. One of our primary interests is the ability to share and use information across multiple platforms. This includes sharing our user interfaces across systems. The X Window System (X), which is now standard with all desktop UNIX systems, provides a consistent base across systems. The actual *look-and-feel* of the X desktop (*e.g.*, menus, window controls, and frames) is given by a separate window manager program. We have used the window manager **fvwm**[10] on all of our systems for several years (version 2.2, **fvwm2**, was released in May 1999). Figure 2 illustrates a `fvwm2` desktop running on our SPARC-based Solaris 7 system.

We find this X window manager appealing because of its relatively low memory usage, the simplicity and flexibility of its configuration, and its ability to manage multiple virtual desktops. Other popular open-source X window managers are described by Chapmann.[11]

While not typically part of the user interface, the X desktop background is, in fact, a window (*i.e.*, the *root* window). Rather than leaving it blank or filled with a static image, we typically run the **xearth** program by K.L. Johnson.[12] This little program displays a customizable view of Earth from space based on the current time-of-day and location (timezone) in the current display's root window, and is updated periodically. Thus, `xearth` not only helps reduce premature monitor failure (since it changes frequently and displays mostly dark colors on a black deep-space background), it also offers free daily remainders of Earth's geography! We chose not to include it in the screendumps presented here because its required black background is not printer-friendly.

Before we leave the topic of standard, efficient user interfaces, we note that our preferred UNIX shell is the Korn shell (ksh). The standard shell on Solaris systems is `ksh`, as is the case with the U/WIN package; on our FreeBSD systems, we use a public domain implementation of ksh88 from the Memorial University of Newfoundland.[13] Thus, we have both consistent and effective command-line and graphical user interfaces on all of our various systems.

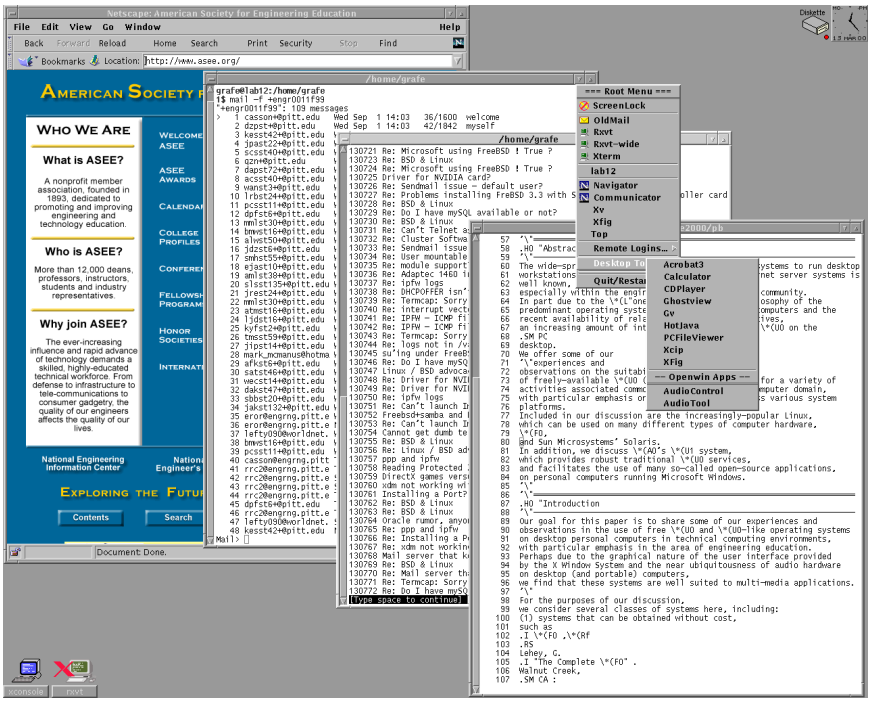

Figure 2 Fvwm2 desktop and X applications

Text and Word Processing. In most cases, we prepare our documents using the plain-text editor **`vi`**, and generate preview and final documents using the GNU **`groff`** package.[14] This package implements most of AT&T's classic *Documenter's Workbench*®(DWB), including the tools **`pic`** (for describing simple line drawings and diagrams), **`eqn`** (for describing mathematical equations), **`tbl`** (for describing tabular information), device-independent **`troff`** (the typesetter), and postprocessors for converting `troff`'s output to an appropriate output device or file format (*e.g.*, PostScript). We have written `troff`-compatible macro packages for formatting our papers, theses, viewgraphs, and other documents into PostScript and Hypertext Markup Language (HTML) documents, which run identically on our SPARC-based Solaris and Intel-based FreeBSD systems. In addition, we use T. Faber's implementation of **`grap`**[15] for generating graphs in our `troff` documents. While we have not used it extensively, the popular *TEX* system and its extensions are generally available for the UNIX systems described here.

We occasionally receive electronic documents that arrive in a common PC word processing format (*e.g.*, Microsoft Word or WordPerfect) rather than an application independent format such as Adobe's PostScript or Portable Document Format (PDF). On such occasions, we make use of Corel's *WordPerfect 8 for Linux*[16] on our Linux-compatible Intel-based systems (Figure 3), which can read and write a wide range of wordprocessing file formats. While not an open-source application, a free personal edition of *WordPerfect 8 for Linux* is available for non-commercial use by individuals from several Internet sites.

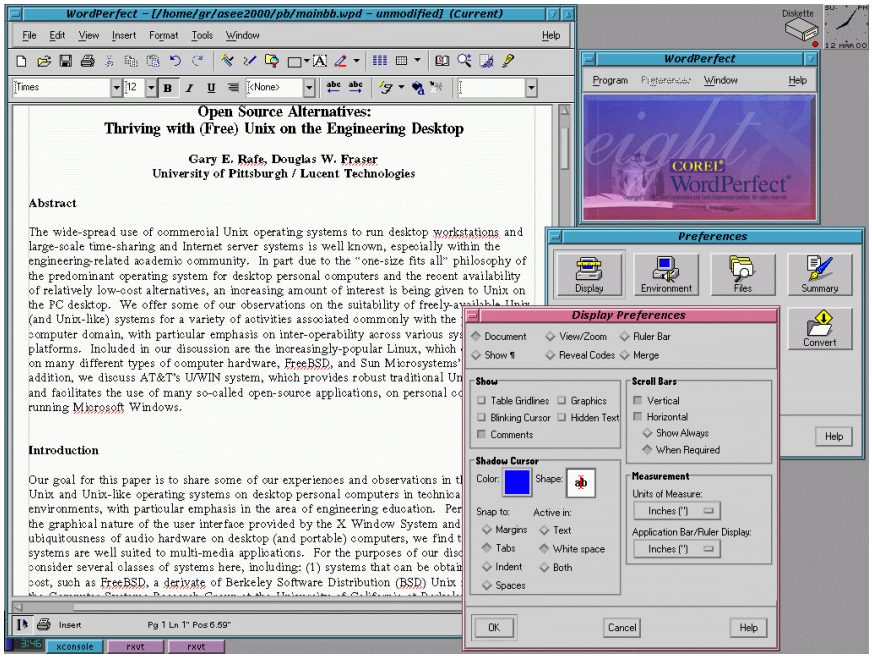

Figure 3 Corel *WordPerfect 8 for Linux*

Since the printing resources to which we have access are generally PostScript-ready, we make frequent use of various open-source PostScript-related tools. These include a set of PostScript utilities by A.J.C. Duggan,[17] which includes a useful program for arranging multiple *logical* pages on a single *physical* page (**psnup**); C. Southeren's **nenscript**[18] program for formatting plain-text (*e.g.*, source code listings and electronic mail messages) as PostScript output for printing; and the popular *Ghostscript* program **gs** from the University of Wisconsin.[19] The gs program is a PostScript language interpreter that produces a output for a wide range of devices and file formats. We use this program most often to preview PostScript documents (using the X front-end programs **ghostview** and **gv**[20]) and to generate architecture-neutral PDF files (*e.g.*, the final form of this paper). We also use gs to print various PostScript jobs on non-PostScript printers, as needed.

We described earlier our desire to share information without regard to the choice of system on either end. In many cases, simple, plain text documents are sufficient to accomplish this. Documents composed in HTML are useful when more complicated formatting and the inclusion of bitmapped images are required. In cases where strict control of page layout is desired, we advocate the use of Adobe's PDF. As we note above, the freely-available Ghostscript program can output PDF from corresponding PostScript input. Of course, the wide-spread availability of PDF interpreters and readers (*e.g.*, Adobe Acrobat Reader[21] and Ghostscript itself) makes this possible.

Electronic Mail. Many diverse configurations are possible for exchanging and managing electronic mail messages. We limit our description to our present University arrangement. After many years, we continue to use the standard **mailx** command on our University desktop, typically running in an X-based terminal window. Because mailx uses a simple text interface, it

is well suited for use of slow dial-up or network connections from remote locations. Since `mailx` (and other electronic mail interfaces) pre-dates the proposed Multipurpose Internet Mail Extension (MIME) standard, we have N.S. Borenstein's *metamail* package[22] installed on our University desktop system. This package includes command-line tools for composing, sending, reading, and processing electronic mail messages with MIME attachments. In addition to having electronic mail messages delivered directly to our desktop system, we maintain an IMAP mailbox with the University's central electronic post office service. To retrieve messages sent to this service, we use E.S. Raymond's **`fetchmail`** program,[23] which can be configured to retrieve mail messages from multiple remote mail services using a variety of messaging protocols. The standard UNIX **`cron`** facility is used to run the `fetchmail` utility at regular intervals throughout the day without our direct intervention.

We conclude these remarks by mentioning the MIME-enabled, character-based **`pine`**[24] electronic mail client from the University of Washington, often standard on many university time-sharing services, may be configured to suit many different operating environments. Also, the Netscape Communicator program, described below, supports both POP and IMAP remote messaging protocols in its Messenger utility.

Web Browsing. The ever-expanding World-Wide Web continues to play an important role in many aspects of engineering education. Our own research at the University of Pittsburgh makes use of several web-related technologies. Documents formatted in standard HTML are available to a large audience, due primarily to the availability of robust web browsers across many different platforms. The **`lynx`** text browser,[25] while not terribly sexy, offers good performance over character-based displays (or in X-based terminal windows). The Netscape Navigator and Communicator[26] browsers are generally released simultaneously for the major UNIX platforms (*e.g.*, Intel Linux, SPARC and Intel Solaris, SGI IRIX, *etc.*). We appreciate having access to both versions of Netscape's browser (*i.e.*, the stand-alone Navigator and the full Communicator) on each of our systems, so each may be called as needed. Since we generally read news groups with the text-based **`trn`** news reader, read electronic mail with the bundled Solaris `mailx` program, and compose HTML with our own `troff` macros, we rarely run the much larger Communicator browser, though it is available on all our systems. The stand-alone Navigator browser running on our Solaris system can be seen in Figure 2.

Multimedia Applications. We next describe a small collection of capable utilities that we use regularly to work with various electronic media. Perhaps the most useful of these tools is the the well-known **`xv`** program by J. Bradley.[27] This X program is used for interactive image display, editing, and conversion from and to a wide variety of bit-mapped image formats; Figure 4 illustrates `xv` running on our Solaris system. We note also that the screen images presented here were captured with the `xv` program running on the respective system.

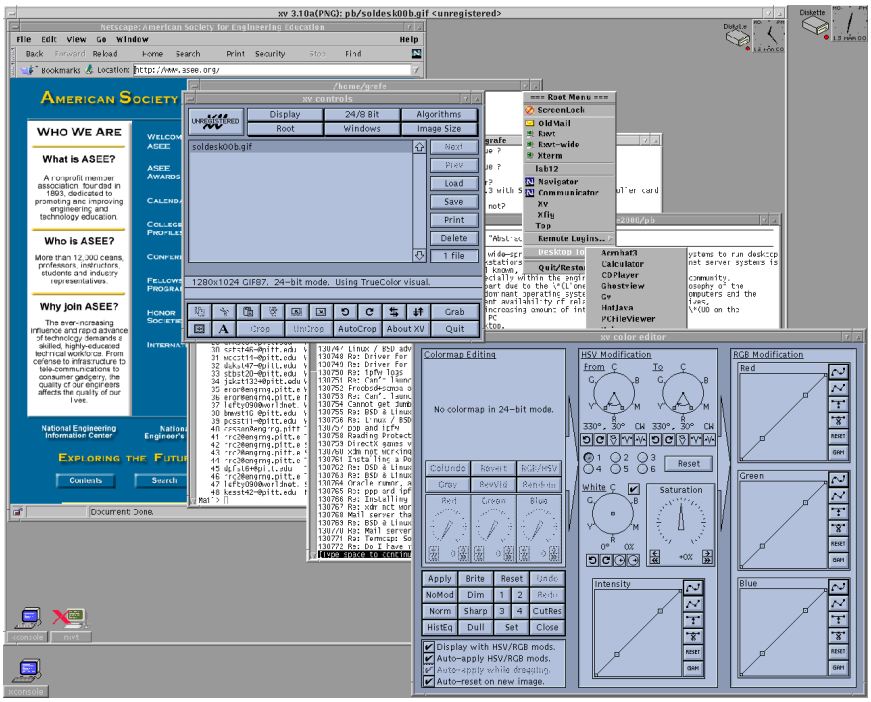

Figure 4 Image display, editing, and conversion with ***xv***

For more complex interactive image processing, we use the GNU Image Processing Program (GIMP).[28] Figure 5 illustrates the `gimp` running on our Intel-based FreeBSD system.

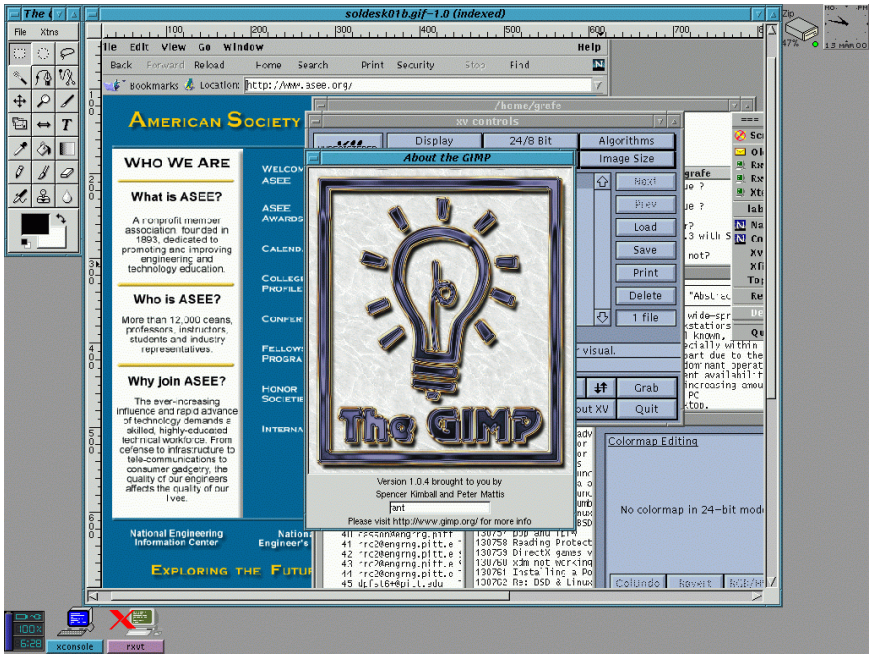

Figure 5 The GNU Image Manipulation Program (GIMP)

For command-line image file processing and conversion, we use the so-called *netpbm*[29] package, which is based on J. Poskanzer's original *Pbmplus* software. The utilities in this package are most often applied in combination with other tools in shell scripts.

We create original photographic images from digital cameras using the small command-line utility **`photopc`** written by E. Crosser.[30] We have used this utility with Epson and Olympus digital cameras connected to both our FreeBSD and Solaris systems. Other original digital images have been created using scanners attached to networked PCs. While researching this paper, we discovered the open-source *Scanner Access Now Easy* (SANE)[31] project. This effort coordinates the development of back-end drivers for various SCSI scanners, and several front-end applications. We hope to investigate the use of the SANE drivers and utilities on our portable FreeBSD systems as time permits.

Finally, we consider the playback of various animation, audio, and video formats on our UNIX systems. M. Podlipec's **`xanim`** program[32] supports a wide range of audio and video codec file formats. For the playback of streaming audio and video on our Intel-based systems, we use Real Networks' *RealPlayer G2 for Linux*,[33] while on our SPARC-based Solaris systems, we use the RealPlayer (v5.0).[34]

Computational Tools. During our graduate studies in industrial engineering at the University of Pittsburgh, we have used an assortment of open-source computational tools and applications, in addition to the normal set of shell tools that comprise UNIX system (*e.g.*, `awk`, `sed`, `grep`, *etc.*), often in place of the typical PC-based solutions suggested by course instructors. These have included M. Berkelaar's **`lp_solve`**[35] for solving mixed integer linear programming problems in an operations research course; J.W. Eaton's GNU Octave[36] application for numerical computations in an applied linear regression course; and the Stuttgart Neural Network Simulator (SNNS)[37] in an applied neural networks course. We note here that as the Linux system continues to gain popularity, more commercial applications are being ported to it. With significance to engineering education, we find that the popular *Mathematica*[38] and MATLAB[39] each have low-cost student versions available for Linux systems. Other applications are sure to follow.

Programming Environments. A rich set of programming environments and toolsets that span architectures and platforms are readily available to both academic and professional software developers in addition to the well-known GNU suite of open-source compilers and related utilities. We consider two such environments here, and a third toolset that facilitates a distributed software development process: Java, the Practical Extraction and Report Language (**`perl`**), and the Concurrent Versions System (CVS).

The Java programming language is having a profound effect on how applications are developed and delivered. In the current context of engineering education, we believe that the Java programming language can be applied effectively throughout and across the curriculum. Current releases of Sun Microsystems' Java Development Kit (JDK) are available for our Solaris systems without cost.[40] Similarly, free and fully-functional JDKs are available for FreeBSD[41] and Linux[42] systems. Additionally, the educational and research communities have free access to the open-source **`jikes`** Compiler Project from IBM Research.[43] This fast Java compiler is available either as source code or pre-compiled binaries for IBM AIX, Intel-based Linux, Sun SPARC-based Solaris, and Microsoft Windows 95/NT systems.

In the Spring 1999 term, one of us (GER) worked in a graduate-level distributed database systems course at the University. All of the assigned work in this course implemented various Java applets, culminating in a small three-tier database application using Remote Method Invocation (RMI) and Java Database Connectivity (JDBC). Since all of the students in the course chose to develop their applets and RMI servers on Microsoft Windows 95 PCs, on-site testing of their finished work was necessary. We accomplished this by bringing our FreeBSD notebook to the appropriate laboratory, connecting to the laboratory's LAN, and running each of the Java applets against Netscape Navigator. To the surprise of many in the course, some of whom were troubled by the need to consider the issue of architecture independence, all of the applets ran correctly on the FreeBSD notebook.

While it may not receive the same level of attention as the Java language, `perl` deserves mention here because of its ubiquitousness and its usefulness. As the the language's name implies, `perl` is well suited to many tasks involving plain-text files. One of us (DWF) has found the language to be useful for the rapid development of relatively small utilities that do useful tasks in a production environment. An example of one such utility generates an editor-specific project file from an existing source code directory tree. This project file subsequently enables the editor to highlight language syntax and browse symbols throughout the source tree, avoiding the odious task of walking the directory hierarchy through the editor's mouse driven, menu popping, file dragging quagmire. The output of a `perl` program need not be simply plain text for it to be useful, either. In our current research at the University, we make frequent use of the open-source FreeWRL,[44] a Virtual Reality Modeling Language (VRML) browser written in `perl`, which uses OpenGL to render three-dimensional graphics. We find this `perl` application useful to test Java interface applets on our Solaris and FreeBSD systems prior to their deployment on a Windows NT PC; Figure 6 depicts a small Java applet driving a simple VRML scene rendered by the FreeWRL browser running on our University Sun workstation.

We next consider an open-source solution to the problem of managing large, distributed software development projects. At Lucent Technologies, we work in a mixed Windows/UNIX environment. Our Windows NT desktops are used for all of our daily tasks (*e.g.*, software development, documentation, e-mail, *etc.*). These desktops are not backed up centrally, so each employee is also provided with a public share on our UNIX server, a Sun SPARCstation 20 running Solaris 2.5.1. It is then the individual's responsibility to copy critical files to the public share or to his/her UNIX account and to maintain proper access permissions there. This may work fine for personal files and such, but is a hindrance to shared development on a common target, as is the case in our project environment. Some people use home spun tools to interface to the Source Code Control System (SCCS) on UNIX while others use Microsoft's Visual SourceSafe version control system product. Both of these solutions have their own quirks and limitations. An alternative solution to the problem of software version control across development teams has been offered by the open-source community, which has settled on the Concurrent Versions System (CVS)[45] almost exclusively. Binary ports of this tool are available for most platforms, as is the source code.[46]

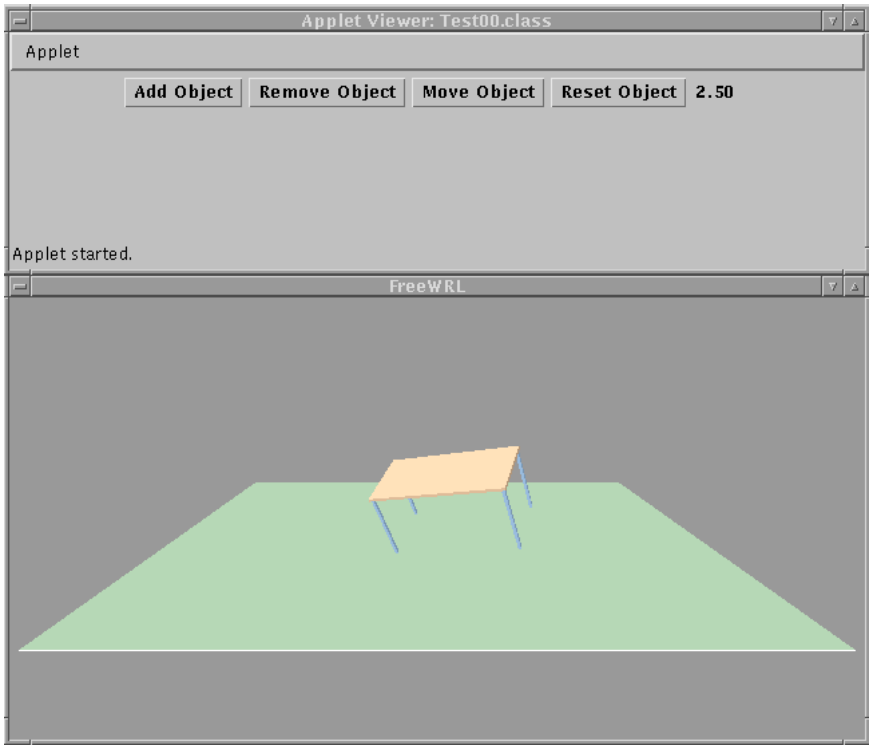

Figure 6 Java Applet and FreeWRL VRML browser

We use CVS in a client/server environment with the server running on our UNIX host. Our Windows clients use standard CVS methods to manage the source code stored on the server. To date, this tool has proven to be robust and very simple to use. Administration on the UNIX host is accomplished through plain text files that are themselves maintained through CVS. The tool is flexible, which means that all development practices must be established locally. Although that means setting the ground rules for the process, it also means that teams have the freedom to use the process with which they are most comfortable. The tool does not enforce a process, it simply enables it. Much like `perl`, CVS is another example of a free tool that has origins in the UNIX environment that has found its way to the Windows world to enable useful activity on the Windows platform.

Security. We comment here briefly on the topic of Internet security, primarily because our University Solaris desktop is connected to a open network, as are our FreeBSD systems when they are connected to the same network, either by modem or Ethernet. We take basic precautions to discourage uninvited access to our systems by persons outside the University's IP domain. The *tcp wrappers*[47] developed by W.Z. Venema are used to control access to various Internet services (*e.g.*, `telnet` and `ftp` servers) when are our systems are connected to the Internet. We also use the encrypted *Secure Shell* (SSH)[48] to communicate (typically via remote login sessions and file transfers) among our various UNIX systems and the University's public UNIX services.

UNIX on Windows

The choice of computer platform often is dependent on the *major* applications and tools required for specific projects, or other organizational criteria. In our cases, we each have the need to access applications that run solely on Microsoft Windows NT PCs regularly. Having strong preferences for the UNIX environment, we investigated several approaches that attempt to give Microsoft Windows PCs a more UNIX-like console interface (and its accompanying extensibility through the use of shell scripts, shell functions, and aliases). The recently released U/WIN package from AT&T appears to provide this functionality. We have U/WIN installed on our Windows NT office desktop at Lucent Technologies and on select project PCs in our department at the University, and find that it provides a reasonable `ksh` console window and most of the common UNIX commands. In addition, U/WIN includes a dynamic library (DLL) that allows us to compile most UNIX utilities using a port of the GNU C compiler `gcc` for Windows[49] with a minimum of modification. At Lucent Technologies, we also make extensive use of the open-source *Cygwin*[50] port of the GNU development toolset for the Windows platform. In this instance, the Cygwin package provides the foundation for a commercial cross-platform development environment we use for our own embedded system software projects.

Summary and Conclusion

We believe that the UNIX operating system model lends itself to the development of efficient problem-solving techniques, particularly in engineering-related disciplines. Our presentation here considered a broad range of software applications available to the engineering education community that operate within the UNIX system model, and in many cases, are not limited to UNIX systems exclusively. Indeed, the UNIX applications and operation systems and environments we considered here are available at no (or low) cost to the educational and research community.

The consistency of user interfaces and portable tools across operating systems and hardware architectures allows us to fashion solutions to problems that run with no (or minimal) changes on our various systems quickly. More importantly, the application of common, architecture-neutral file formats allows us to share information easily and effectively with others. In conclusion, we find that the free UNIX desktop can be a powerful teaching and research tool for the engineering educator.

Bibliography

1. Lehey, G. *The Complete FreeBSD*. Walnut Creek, CA: Walnut Creek CDROM (1997).
2. Weigand, J. The cooperative development of Linux. In *Proceedings of the 1993 IEEE International Professional Communication Conference.* New York: IEEE (1993).
3. Korn, D.G. Porting UNIX to Windows NT. In *Proceedings of the USENIX 1997 Annual Technical Conference.* Berkeley, CA: USENIX Assoc. (1997).
4. Noer, G.J. Cygwin: A Free Win32 Porting Layer for UNIX Applications. In *1998 USENIX Windows NT Workshop Proceedings.* Berkeley, CA: USENIX Assoc. (1998).
5. `http://www.sun.com/edu/solaris/individual.html`

6. http://sunfreeware.com/
7. http://www.research.att.com/sw/tools/uwin/
8. Bolsky, M.I., & D.G. Korn. *New KornShell Command and Programming Language, 2nd Ed.* New York: Prentice Hall PTR (1995).
9. http://www.freebsd.org/
10. http://www.fvwm.org/
11. http://www.plig.org/xwinman/
12. http://www.cs.colorado.edu/~tuna/xearth/
13. http://www.cs.mun.ca/~michael/pdksh/
14. http://www.gnu.org/software/groff/groff.html
15. http://www.lunabase.org/~faber/Vault/software/grap/
16. http://linux.corel.com/products/linuxproducts_wp8.htm
17. ftp://ftp.dcs.ed.ec.uk/ajcd/psutils-p17.ta.gz
18. http://www.im.lcs.mit.edu/~magnus/nenscript/
19. http://www.cs.wisc.edu/~ghost/
20. http://wwwthep.physik.uni-mainz.de/~plass/gv/
21. http://www.adobe.com/products/acrobat/alternate.html
22. ftp://thumper.bellcore.com/pub/nsb/README
23. http://www.tuxedo.org/~esr/fetchmail/
24. http://www.washington.edu/pine/
25. http://lynx.browser.org/
26. http://home.netscape.com/download/selectplatform_1_702.html
27. http://www.trilon.com/xv/xv.html
28. http://www.gimp.org/
29. ftp://metalab.unc.edu/pub/Linux/apps/graphics/convert/netpbm-8.0.tar.gz
30. http://www.average.org/digicam/
31. http://www.mostang.com/sane/
32. http://xanim.va.pubnix.com/
33. http://proforma.real.com/real/player/linuxplayer.html
34. http://proforma.real.com/real/player/blackjack.html
35. ftp://ftp.es.ele.tue.nl/pub/lp_solve/README
36. http://www.che.wisc.edu/octave/
37. http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/snns.html
38. http://www.wolfram.com/products/student/mathforstudents/
39. http://www.mathworks.com/products/studentversion/
40. http://www.sun.com/solaris/java/download.html
41. http://www.freebsd.org/java/
42. http://www.blackdown.org/
43. http://www.research.ibm.com/jikes/
44. http://www.crc.ca/FreeWRL/
45. Fogel, K.F. *Open Source Development with CVS.* Scottdale, AZ: The Coriolis Group (1999).
46. http://www.sourcegear.com/CVS/
47. ftp://ftp.porcupine.org/pub/security/index.html
48. http://www.ssh.org/
49. ftp://ftp.xraylith.wisc.edu/pub/khan/gnu-win32/uwin/
50. http://sourceware.cygnus.com/cygwin/

GARY E. RAFE
Gary Rafe is a Ph.D. candidate in the Industrial Engineering Department at the University of Pittsburgh. He received the B.S.I.E. degree from Alfred University, and the M.S. in Manufacturing Systems Engineering from the University of Pittsburgh. Previously, Mr. Rafe was on the faculty of the State University of New York's College of Technology at Alfred, teaching courses in computer and industrial control programming, computer-aided-design and

manufacturing, and mechanical engineering technology, and managed the College's workstation laboratory network. In addition to his desire to eschew the hegemonic personal computer operating system, his research interests include the application of information system technology in manufacturing enterprises, automating the product design-to-manufacturing process, CAD/CAM integration, and the use of telecommunication technology for training and education.

DOUGLAS W. FRASER

Douglas Fraser has worked as a developer for AT&T and Lucent Technologies for sixteen years. He learned to develop embedded systems using C in the UNIX environment there and has continued to practice that craft with brief forays into UNIX application development. Most of his career has concentrated on remote distributed telephony test systems using both intrusive digital tests and non-intrusive voice band tests over digital T-carrier systems. Mr. Fraser's recent work has included infrastructure development for an eighty channel DWDM optical line system for Lucent's Optical Networking Group. He is currently part of a team developing reference designs for DSL broadband modems at the Microelectronics division of Lucent Technologies.