

Engineer's Handbook

General Stuff

May 5, 1994

Contents

- Pointers to Engineering Resources at SGI
- SGI Frequently Asked Questions
- Ptools Class notes
- SGI Bug System Class Notes
- Big Book of Names
- Manuals Section of Price Book
- Using X
- Customizing X
- Open GL & X
- Graphics Library Quick Reference Guide

Engineer's Handbook

General Stuff

May 5, 1994

How to access the handbook command

- install the information tools software (only need to do this once or when the tools change)

```
% su
# inst -f dist.wpd:/sgi/infotools
# exit
% rehash
% echo "make sure /usr/local/bin is in path"
```

- handbook always accesses most current version of the data

- to find out what chapters exist

```
% handbook
```

- to find out what chapters exist

```
% handbook <chapter_name>
```

- to print a chapter on your default printer

```
% handbook -o - <chapter_name> | lp
```

- to get a copy of the actual data

```
% handbook -o <file_name> <chapter_name>
```

- see the man page for further handbook options

Engineer's Handbook General Stuff

May 5, 1994

Pointers to Engineering Resources at SGI

probable data location: `dist.wpd:/sgi/doc/swdev/SGIintro.ps`
access via: handbook SGIintro

Silicon Graphics Inc Pointers to Engr Resources

NOTES

Publication Date: 5/7/93

Overview of FAQ

- What is FAQ

A collection of Frequently Asked Questions both originated internally and externally.

faq.c	C language Frequently Asked Questions
faq.dso	DSO Frequently Asked Questions
faq.gnu-emacs	GNU Emacs Frequently Asked Questions
faq.gnu-emacs.sgi	SGI Localizations to GNU Emacs
faq.graphics	Graphics Frequently Asked Questions
faq.gutils	Graphics Utilities Frequently Asked Questions
faq.jpeg	JPEG Compression Method Frequently Asked Questions
faq.motif	Motif Frequently Asked Questions
faq.perl	perl Frequently Asked Questions
faq.sgi	SGI Frequently Asked Questions
faq.x	X Window System Frequently Asked Questions

- What kinds of info is in faq.sgi

- System Software
- Mailers, Newsreaders and the network
- Useful Programs
- File mount issues
- SGI Software Development Environment
- Unix Hints
- Other Stuff

- Where to obtain faqs?

```
% su
# inst -f dist.wpd:/sgi/faq
Inst> go
Inst> exit
# exit
% /usr/local/bin/faq
```

Software Releases

- miniroot & inst (new inst just here independent of miniroot)
 - new alphas from miniroot, else just inst as root
 - see 'faq faq.sgi 4' and 'faq faq.sgi 5'
- numbering system for releases (4.0.5)
 - 4 is major release number - compatibility break
 - 0 is minor release number - feature release
 - 5 is bug fix release
- sources and distribution machines
 - dist.wpd the official distribution machine
 - distribution machines local to domains/building
 - dist.wpd building 9
 - stress.asd building 7
 - mars.esd building 2
 - atlas.corp all other buildings

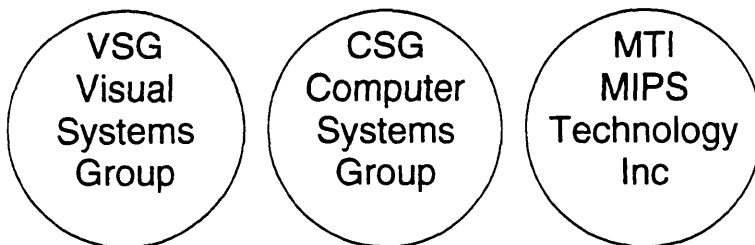
Organizational Evolution

- Silicon Graphics in 1982-1986 (Undifferentiated, Primal Ooze)
- 1987--Divisionalization

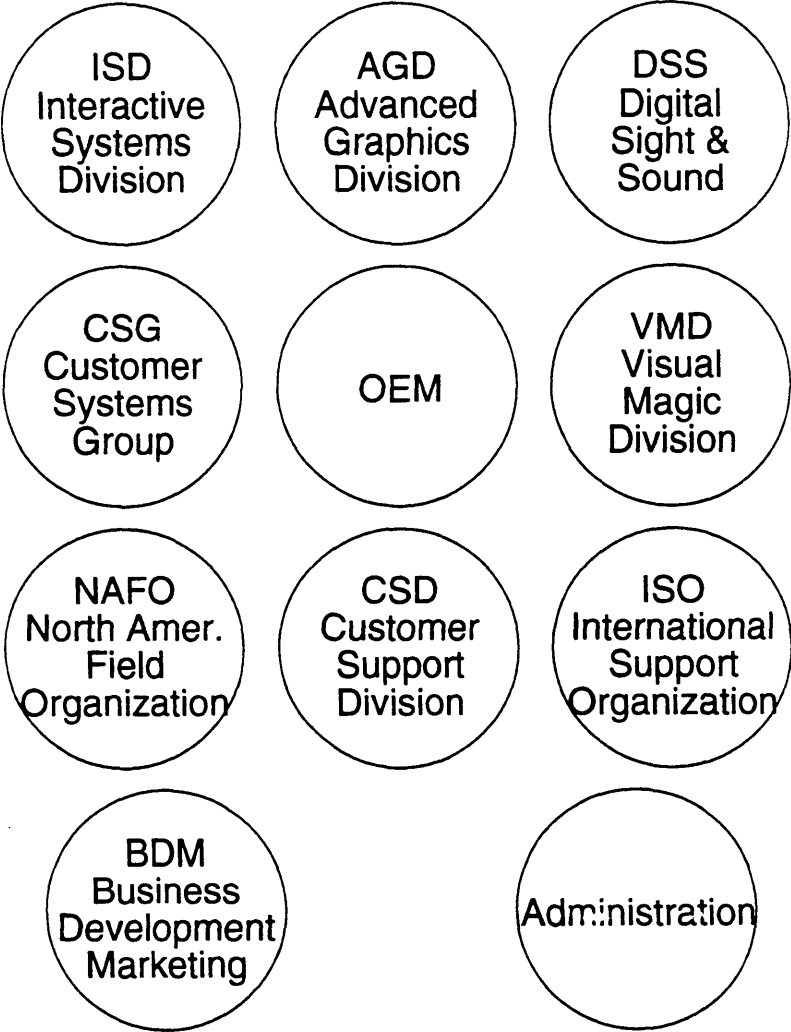


- 1986-1992--sporadic changes
For example, WPD->SSD->SSTC

- 1992--MIPS/SGI merger and reorganization



Organizational Evolution current (yeah right)



Product History

- Corporate Nicknames and Tee-shirts
- Hardware Names==Flowers in the Early Years

1984: IRIS 1000 Series

Graphics Terminals and Workstations

Geometry Engine™

Geometry Pipeline™

Proprietary VLSI

68010 CPU

1985-6: IRIS 2000, 2000 Turbo, and 3000 Series

Better Graphics

68020 CPU for 2000 Turbo and 3000 Series (“Juniper”)

1987: IRIS 4D (60-R2300, 70-ip4, 80-ip4, 100-ip5) (“Clover”)

First MIPS CPU product

First Twin-Tower, Plastic Covered Chassis

More Product History

- Hardware Evolution

1987: IRIS GT ("Clover 1G")

Graphics Upgrade to IRIS 4D

1988: Personal IRIS (ip6) ("Eclipse")

"Eclipse Sun"--Get it?

First product of ESD

Mid-size chassis (could be lifted by one person)

Basis for Personal IRIS ("PI") Series (4D/20)

1988: IRIS GTX ("Clover 2") and S

ASD Products--The Power Series (CPU upgrades)

First Server Product

Multi-Processor machines ("Terminator")

Numbering System

4D/210-ip9, 220-ip7, 240-ip7, 280-ip7

hundreds digit==clock speed of processor (25MHz, 33Mhz, etc.)

tens digit==number of processors

8 processor machine is rack-mounted ("Predator")

Yet More Product History

- Hardware Evolution

1989-90: Personal IRIS CPU & Graphics upgrades

4D/25-ip10(hinv:ip6), 4D/30-ip12, 4D/35-ip12

4D/35 is 36 MHz CPU ("Magnum PI")

(really an Indigo CPU in PI guise, just need microphone
and 8 ohm speaker)

Also TG--Turbo Graphics upgrades

Personal IRIS Server product

Never a PI multi-processor product

1990: IRIS VGX or PowerVision ("Stapuft")

ASD Product

Graphics Upgrade to GTX

1,000,000 Polygons per second

1990: Single Tower Chassis available ("Die Hard")

1991: Skywriter

ASD Product aimed at Visual Simulation

Graphics Upgrade

also available as VGXT

Two Graphics Pipelines

Recent Product History

- Hardware Evolution

1991: IRIS Indigo (ip12) (“Hollywood”)

ESD Product

New Desktop Chassis

Very little “Graphics” Hardware

1992: IRIS Crimson (ip17)

ASD Chassis with ESD Graphics options (see below)

First R4000 CPU Product

1992: XS, XS/24, and Elan Graphics options (“Express”)

Graphics Upgrades to Indigo and Crimson products

1992: The Reality Engine “Venice”

Future Graphics Upgrade

1992: Indigo R4K “Blackjack” (ip20)

R4000 Indigo

1993: Indigo² “Full House” (ip22)

R4000 with EISA

1993(August): “Guinness” “Sapphire” (ip22)

R4000 lower cost than indigo

single board including graphics

Recent Product History (Software)

- Software Name==Trees
- physically located on bonnie (aka jake)
 - Aspen (3.2.x)/Birch (3.3.x) combined release (bonnie:/aspen)
 - Cypress (4.0.{0,1}) (bonnie:/cypress)
 - Lonestar (4.0.{2,3,4,5}) (bonnie:/lonestar)
 - Dogwood (was to be 4.1 or 4.0.6) (bonnie:/dogwood)
 - Blackjack/Venice limited release (bonnie:/bvd)
 - Sherwood (5.0) (last major release)

SGI Software Tools

- **WorkSpace**
 - graphical interface to working environment
 - file manipulation, execution
- **iconsmith**
 - make geometric icons for WorkSpace views
- **Showcase**
 - drawing and presentation package
- **picmail**
 - mail notifier with pictures of mail senders
 - pictures located on `newmedia.esd:/usr/local/lib/faces`
- **zip**
 - a text editor written by Tom Davis
- **CASEVision**
 - a suite of CASE tools
 - static analyzer (`cvstatic`), debugger (`cvd`), performance tool (`cvspeed`)
- look in `/usr/sbin` (`ipaste`, `notepad`, `showmap`, `twilight`)
- mailers and newsreaders discussed later

SGI Toolchests

- upper left icons on screen
- System - Various system managers and workspace
- Windows - window management
- Tools - fork a new window and several useful tools
- Demos - useful demos to see what your system can do
- Overview - very useful overviews

Getting Started

Working with Windows

Exploring WorkSpace

Customizing Your Environment

Tutorial resources

- handbook command

```
% su
# inst -f dist.wpd:/sgi/handbook
Inst> go
Inst> exit
# exit
% /usr/local/bin/handbook
```

1) makeconv	SGI Makefile Conventions
2) commonrules	commonrules/commondefs Class notes
3) ptools	Ptools Class notes
4) faq.sgi	SGI Frequently Asked Questions
5) sw4inst	Engineer's Guide to Packaging Software for Inst
6) using_x	Using X
7) config_x	Configuring X
8) prjmgmt	SGI Software Project Management Class notes
9) glrefcard	Graphics Library Quick Reference Guide

- ~4Dgifts

```
dev.sw.giftsbasic
dev.sw.giftsfull
c++.opt.gifts
eoel.sw.audio
eoe2.sw.demos
il.sw.gifts
imgtools.gifts.images
netvis_data.sw.gifts
pwrc.opt.gifts
```

- SGI Manuals

- 'faq x 2' for X booklist, especially Doug Young books
- 'faq motif 19' for Motif booklist
- other engineers

SGI Software

- X Window System
- OSF/Motif Widget Set
for user interfaces (buttons, menus, slider bars, dialog boxes)
- IRIS Graphics Library (GL)
API for creating interactive, 3-D, color graphics programs
- OpenGL
“Industry Standard” API to supercede IRIS Graphics Library
- IRIS Inventor (“Scenario”)
object-oriented 3-D graphics toolkit for writing programs
built on top of GL (eventually OpenGL) API
- ImageVision Library
object-oriented toolkit for writing image processing programs
- IRIS Explorer
scientific visualization environment
extensible (end-user can create/program new applications)
uses IRIS Inventor, ImageVision Library, and OSF/Motif

SGI Conventions

- Internal classes exist for each
- Makefile Conventions
 - see 'handbook makeconv' and 'handbook commoninclude'
- ISM (Independent Software Module)
 - see 'handbook ism'
- SGI Software Lifecycles
 - see 'handbook prjmgmt' for overview from Project Manager's viewpoint
- ptools - SGI internal configuration management tools
 - see 'handbook ptools' for usage info.
 - see 'faq faq.sgi 42' for installation info.
- SGI bug system
 - submit bugs via mailer or pv
 - view bugs via newsreader, pv(gui), or pvquery (line oriented)
 - 'inst dist.wpd:/sgi/pv/new' for pv and pvquery
 - see 'man pvintro' for overview and info on using news

Interoffice yakking Communication among ourselves

- e-mail and MacMail
- “locate”
- news readers and bulletin board groups
- courtesy and privacy

e-mail

- sgi.com and sgate.sgi.com are internet gateways for SGI (sgate.sgi.com is fastest for ftping , see 'faq faq.sgi 18')
- Mail to sysrequest@corp to get name added to sgi.com aliases
- need to run GETmail
periodically rerun GETmail to get correct view of SGI network
- mail
standard-issue AT&T UNIX mail (on default system)
- Mail
Berkeley Mail with enhancements (on default system)
- MH
Rand mail (not shipped with any SGI products)
available internally
- xmh
point-and-click-with-windows interface to MH
included in the eoe2.sw.Xapps subsystem
must have MH installed on your system (tough for customers!)
- Zmail
- elm
HP interactive screen-oriented mailer

MacMail

- can write to people who use MacIntoshes
- e-mail address is Their_Name@msmail.corp

(People at mips.com may need to use
Their_Name@msmail.corp.sgi.com)

“locate”

- on-line phone and e-mail directory
- installation

```
% su
# rcp guest@odin.corp.sgi.com:/usr/local/bin/locate /usr/local/bin
# exit
% rehash
```

- Usage

```
locate <name>
```

where <name> is a loginID@, or portion of a person's first or last name, or phone extension

(Actually you can view the first parameter to locate to be like a grep pattern with an implied -i option)

- Your /etc/passwd or NIS passwd name must EXACTLY match the locate database if the e-mail address is to be found

Might try “ypchpass” to see NIS name

see ‘faq faq.sgi 36’

News Readers

- see faq faq.sgi 15
- nn--a menu based netnews reader

```
% su
# rcp guest@anchor.esd:/usr/local/install/install.anchor /usr/tmp
# /usr/tmp/install.anchor nn
# exit
% rehash
```

- rn, rrn--an NNTP news reader

```
% su
# rcp -rv guest@odin.corp:/usr/src/news/rrnkit /usr/tmp
# cd /usr/tmp/rrnkit
# more README-SGI
# ./sgi-install-rrn      <-- It will ask you for your news
                        server which is odin.corp.sgi.com

# cd ..
# rm -rf rrnkit
# exit
% rehash
```

- trn--a menu based news reader

based on rn, but as easy to use as nn

```
% su
# rcp guest@lunch.wpd:/usr/local/install/install.pdc /usr/tmp
# /usr/tmp/install.pdc trn-2.0
# exit
% rehash
```

- xrn--an X-based news reader with a graphical interface

```
% su
# rcp guest@newmedia.esd:/usr/local/bin/install.ui /usr/local/bin
# /usr/local/bin/install.ui xrn
# exit
% rehash
```

Global News Groups of Interest

- USENET--a global group of people who exchange articles
- interesting global USENET groups
 - news.announce.newusers--explanatory postings for new users
 - comp.sys.mips--systems based on MIPS chips
 - comp.sys.sgi--SGI workstations and software
- prefixes for global news groups
 - alt.xxx, rec.xxx, soc.xxx, talk.xxx--social and recreational
 - aus.xxx, ba.xxx, ca.xxx, de.xxx, fj.xxx--regional
Australian, Bay Area, California, Deutsch, Japan UNIX
 - bit.xxx, gnu.xxx, ieee.xxx--special interest
 - clari.xxx--wire services for current events (stock prices!)

Local News Groups of Interest

- All local groups prefixed with `sgi.xxx`
- General interest groups
 - `sgi.announce.important`
 - `sgi.general`
 - `sgi.engr.all`
- Work-related special interest groups (or prefixes)
 - `sgi.engr.xxx` and `sgi.engr.x.xxx`
 - `sgi.bugs.xxx` (list on next page)
 - `sgi.hr.xxx`
- Flame-retardant groups
 - (implied snarl/smiley with each posting)
 - `sgi.bad-attitude`
 - `sgi.bizarre` (offspring of `mips.bizarre`)
- Social news groups
 - `sgi.rec.xxx`
 - `sgi.talk.xxx` (such as lavender, politics, ratical, koolkats)

SGI Bug Newsgroups

- bugs is alias to bug group du jour (sgi.bugs.lonestar)
- by trees
 - sgi.bugs.dogwood
 - sgi.bugs.sherwood
 - sgi.bugs.lonestar
 - sgi.bugs.bvd
- by project
 - sgi.bugs.autoplan
 - sgi.bugs.calypso
 - sgi.bugs.compilers
 - sgi.bugs.dmedia
 - sgi.bugs.explorer
 - sgi.bugs.fddi
 - sgi.bugs.frame
 - sgi.bugs.ibm
 - sgi.bugs.illustrator
 - sgi.bugs.imagevision
 - sgi.bugs.ivr
 - sgi.bugs.netls
 - sgi.bugs.netvis
 - sgi.bugs.networker
 - sgi.bugs.newport
 - sgi.bugs.oasis
 - sgi.bugs.online
 - sgi.bugs.opengl
 - sgi.bugs.osi
 - sgi.bugs.printware
 - sgi.bugs.projmgmt
 - sgi.bugs.ptools
 - sgi.bugs.pv
 - sgi.bugs.scenario
 - sgi.bugs.showcase
 - sgi.bugs.swexpress
 - sgi.bugs.tinyui
 - sgi.bugs.video
 - sgi.bugs.viskit
 - sgi.bugs.volmgr
 - sgi.engr.case.bugs

Sensible Rules of News Posting

- You should use broadcasting (news) only when you have some information that needs to be broadcast to members of a group, or when you need some information and you don't know which member of the group has that information.
- If you need additional information about a broadcast article, please send an e-mail message to the person who posted that article.
- Please do not post replies to posted questions. Send e-mail to the person who posted the questions. That person can post a summary of all answers if those answers may interest other members of the group. If you are the only person likely to respond or the subject matter expert for that question, then feel free to respond directly to the newsgroup.
- If you are interested in the answers to posted questions, send e-mail to the person who posted the questions.
- See news.announce.newusers for more elaborate discussion of these issues

“What is Usenet?”

“A Primer on How to Work with the Usenet Community”

Privacy

- The “guest” account is a mechanism for people to have access to your system. Please get permission from owner before accessing (except for public machines).
- Put a password on “guest” to close the mechanism, if desired.
- Otherwise, it’s a no-no to try to pry into directories, if you don’t have owner’s permission.
- By all means, put a password on “root.” (Although many still DON’T!)

Other Resources

- Technical Education
training courses
- Customer Support Engineering (CSE)
formerly Technical Assistance Center (TAC)
Phone Support
Call them if you have hardware problems
An SSE will be dispatched
- Phone number for both Tech Education and CSE is
1-800-800-4SGI
- Help Desk x4357
MacIntosh support
Network/Telecomm
Software Express (formerly Move Over Mac)

Engineer's Handbook General Stuff

May 5, 1994

SGI Frequently Asked Questions

probable data location: `access via: handbook faq.sgi`
`dist.wpd:/sgi/doc/FAQ/faq.sgi`

94/05/03
11:18:58

faq.sgi

These questions and answers are the product of many people based on both a living and moving target. Your corrections and comments are needed for accurate stalking of this target. Send comments, corrections and requests to Tom Murphy (murphy@sgi.com). I would especially like to hear from you if you are a source for the content of all or part of a question.

This file is: dist.wpd:/sgi/doc/FAQ/faq.sgi
created: Tue May 3 11:15:56 PDT 1994

changes bars based on previous version: Sun May 1 00:01:29 PDT 1994

Table of Contents

A "[" preceding a question means the contents of the answer has changed since the last posting. A "*" preceding a question means this a new question. The same symbols are used at the end of a line within a question to show the lines that have been changed or are new.

Other Sources of Information

- Q1: Is there a way to view Frequently Asked Questions online?
- Q2: What other collections of Frequently Asked Questions (FAQs) are available?
- Q3: How frequently are these questions updated and by whom?
- Q4: How do I add or modify material in this FAQ?
- Q5: How do I report errors and new questions for the FAQ?
- Q6: What other sources of information are there?
- Q7: How do I get a copy of the Engineer's Handbook?
- Q8: How do I get a copy of Uncle Art's Big Book of IRIX (Cypress only edition)?
- Q9: How do I get the Mountain View CoreDump Newsletter?

System Software

- Q10: How do I find out what SGI software is installed on my workstation?
- Q11: How do I get SGI software releases for my machine?
- Q12: How do I find out, in advance, the pitfalls bleeding edge campus alphas?
- Q13: How do I use inst to install software on my workstation?
- Q14: What if inst does not install everything I thought it would?
- Q15: How do I load software using the miniroot?
- Q16: How do I do a live install thus avoiding a mini-root install?
- Q17: How do I get hardware and software support for my workstation?
- Q18: What is a maintenance software release?
- Q19: How do I interpret the ten digit version number of a release?
- Q20: How do I set the version number for an inst image I am making?
- Q21: How do I get copies of SGI manuals?
- Q22: How do I get the InSight viewer and the bookshelves to view?
- Q23: How do I find the source for SGI software?
- Q24: How do I find where source is located within a source tree?
- Q25: How do I find out about a specific release using the whatswhere command?
- + Q26: How do I find out what version of Irix in which a particular bug first appears?
- Q27: Is there a place from which I can NFS SGI reference material?

Mailers, Newsreaders and the network

- Q28: What mail programs are available and how do I install them?

- Q29: What do I do if I'm having problems sending mail to people at SGI?
- Q30: What's "metamail" and why can't elm read it?
- Q31: What are the different types of newsreaders?
- Q32: What news readers are available and how do I install them?
- Q33: How do I get the rn that allows for more newsgroups in the in-ware?
- Q34: What are the mail aliases that I can use to post to newsgroups?
- Q35: What other useful campus aliases exist?
- Q36: Do we archive any of the SGI news groups?
- Q37: What do I need to know to use our connections to the Internet?
- Q38: How do I find out where or what ftp material is available?
- Q39: How do I get files via anonymous FTP from Internet sites?
- Q40: Which local servers have network documents, such as RFCs, on line?
- Q41: Do the Request For Comment files (RFCs) really contain a complete history of the Internet?
- Q42: What is ISDN?
- Q43: Who provides ISDN in the Santa Clara valley?
- Q44: How to get ISDN at home?
- Q45: What equipment is required to use ISDN?
- Q46: What about ISDN order forms and other documentation?
- Q47: What is mosaic?

Useful Programs

- Q48: Where is a good place to look for (and deposit) software local to SGI?
- Q49: How do I get my picture added to the face server?
- Q50: How do I set up my system so that I can click on my face to log in?
- Q51: How do I get picmail, the program that shows the faces of those who email me?
- Q52: Is there a tool to view the entire organization including people's faces?
- Q53: How can I see the face for someone?
- Q54: How do I get my picture on the banner page of my laser printer output?
- Q55: What are infotools and how do I load them?
- Q56: How do I get ghostview, the working PostScript previewer.
- Q57: How do I get a copy of Emacs?
- Q58: What is Showcase and how do I get a copy?
- Q59: How do I get a copy of my favorite shell?
- Q60: How do I get Perl, that program thats supposed to replace awk, sed and sh?
- Q61: How do I use rfind, that speedy new replacement for the find command?
- Q62: How do I get that program that "describes the hardware characteristics of my machine?"
- Q63: Will "Power Bloat Vision" (tm) really help me visualize creeping bloat?
- Q64: Is there a vacation program to automatically respond to email that I am gone?
- Q65: How do I get a bingo gamecard that I saw used in a high level meeting?
- Q66: How do I get webster (and xwebster), the online dictionary/spell checker?
- Q67: How do I get that program enhance to spiffify my rgb images?
- Q68: How do I get a copy of VCal, the reasonable electronic daytime program?
- Q69: How do I get less which is more than more?
- Q70: How do I load Framemaker?
- Q71: How do I get a copy of locate, the online phone directory program?
- Q72: Why isn't my login IP in the locate database?
- Q73: How do I change the information about me in the locate database?

File maint issues

94/05/03
11:18:58

faq.sgi

- Q74: How do I decide which machine to access, given several choices?
Q75: What directories should be NFS-mounted on my system?
Q76: What directories should I automount on my system and how do I do it?
Q77: How do I let others automount (or mount) my files?

SGI Software Development Environment

- Q78: How do I set up a software development environment on my workstation?
Q79: How do I get a copy of ptools, and how do I find out about new releases?
Q80: How do I get the document on SGI Makefile conventions?
Q81: What bug tracking system does SGI use?
Q82: What are the newgroups to which I should post bugs?
Q83: How do I install bug tracking software on my workstation?
Q84: How do I get color schemes to work with pv?
Q85: Where can I find documentation on the SGI Bug system?
Q86: How do I find out about known bugs?
Q87: How do I remove a source file from the source tree?
Q88: How do I manage all the environment variables I have to set?
Q89: What is an ISM?
Q90: If I got an ISM, how do I produce it?
Q91: What do I need to get started as an ISM?
Q92: What are the ISM targets I need to support?
Q93: How do I obtain the latest ROOT and TOOLROOTs?
Q94: How do I use the ISM template directory?
Q95: What things do I need to do to maintain my ISM (nightly builds etc)?
Q96: How do I generate version numbers for my ISM?
Q97: How do I construct an exitop for inst?
Q98: How do I test an inst image that I've generated?
Q99: How do I select which products to build?

Releasing S/W Products

- Q100: Who needs to know about my product release?
Q101: Do I need to talk to each group individually?
Q102: When do I need to tell them this info?
Q103: How do I get the Release Engineering group to build my product?
Q104: When do I need to start an ECO?

IndigoMagic Environment Stuff

- Q105: How can I run my own background program with IndigoMagic (perhaps a nightsky with shooting stars)?

Unix Hints

- Q106: How do I bypass the password prompt when switching to root?
Q107: How do I arrange for file name completion from the c shell?
Q108: How can I let someone be root on my system, without giving out my password?
Q109: When should I turn ypserve on (using the chkconfig command)?

Other Stuff

- Q110: How do I set up a dial back connection from work to home?

- Q111: How do I set use the dial-back connection from an SGI machine?
Q112: How do I get blank 1/4" tapes?
Q113: How do I get a blank Exabyte tape?
Q114: Can I electronically extract the serial number of my machine?
Q115: What is the difference among the r4 processors and what does it mean to me?
Q116: How can I learn more about X and Motif?
Q117: Is it possible to scan a picture into an SGI image file?
Q118: What do I do if I suspect I have Carpel Tunnel Syndrome?
Q119: How can I link up with a carpool or some other commute alternative?
Q120: What does the name IRIS mean?
Q121: Who are SGI's Stock Brokers?
Q122: Are there any notary publics at SGI?

- Q1: Is there a way to view Frequently Asked Questions online?

I'm glad you asked that question. You can use the faq command (see Q55 to obtain it). Coming soon will be an Insight viewable form of faq.sgi, the file you are currently viewing.

- Q2: What other collections of Frequently Asked Questions (FAQs) are available?

There are a number of collections of frequently asked questions that are periodically posted to the net. The newsgroup news.answers contains many of them. See Q12 for availability of news readers.

There is a codevision collection of frequently asked questions that is periodically posted to the newsgroup sgi.engr.case.

There are other FAQs that are available in dist.wpi:/sgi/doc/FAQ. The contents of the README file from that directory is duplicated below:

FAQ name/ posted to	maintainer/ provider	contents
faq.comp.lang.c	davea	C Language Questions
faq.clarity	philc@corp	Clarify ClearSupport Call Management System
faq.dso	gischer	Dynamic Shared Objects
faq.dwarf format	davea	Binary debugging information
faq.environment	ddp@esd	Frequently Asked Questions about Recycling (at SGI)
faq.gnu-emacs.0 comp.emacs	jackr	Questions about GNU Emacs: Intro and Table of Contents
faq.gnu-emacs.1 comp.emac	jackr	Notation, Sources (1-29)
faq.gnu-emacs.2 comp.emacs	jackr	Environments, Keys (30-69)

94/05/03
11:18:58

faq.sgi

faq.gnu-emacs.3 comp.emacs	jackr	Building, Weirdnesses, Configuring, ... (70-126)
faq.gnu-emacs.sgi <none>	jackr	Localizations for the copy on Jeeves
faq.graphics comp.graphics	ib	Questions about computer graphics.
faq.gutils alt.graphics.pixutils	ib	Questions about computer graphics utilities.
faq.jpeg comp.graphics	ib	Questions about a data compression method specified by the Joint Photographic Experts Group.
faq.motif comp.windows.x.motif	joel	Questions about Motif
faq.netls	kchin	Network License Server
FAQ.perl comp.lang.perl	murphy	Questions about the programming language perl
faq.sgi sgi.engr.all sgi.general	murphy	Questions local to the SGI software development community.
faq.sherwood	paulm	Info on Sherwood release (5.0)
faq.x	ib	Questions about the X Window System

Q3: How frequently are these questions updated and by whom?
(Author is murphy)

Hold on there! That's two questions. faq.sgi is rebuilt each night at midnight if there any changes to its question directory. There is a monthly posting (the first of the month) to sgi.engr.all showing the table of contents (with change bars reflecting the changes made during the previous month).

The maintenance of the file is currently basically by Tom Murphy, although things are in transition. Feel free to update data that you own (See Q4). The game plan is to have the owners of the information be able to use ptools to update their own information as opposed to the current situation where Tom is a bit of an information bottleneck (1 year and a half went by with no updates and the backlog of updates are currently being worked through). A future version of the faq viewer (see Q1) will optionally report the authors and the date of the information.

Q4: How do I add or modify material in this FAQ?
(Author is murphy)

It is the best of methods. It is the worst of methods.

The file bonnie.wpd:/depot/cgi_info/FAQ/README spells out all the gory details. Here is a quick summary. Each question is a separate file in bonnie.wpd:/depot/cgi_info/FAQ/faq.sgi/questions

You will need to do a ptools workarea into bonnie.wpd:/depot and p update \$WORKAREA/cgi_info/FAQ/faq.sgi.

There are some formatting conventions in place so that you can just edit the data. Lines will be made full and change bars added by the FAQ generation software. The generation software also handles cross references. The formatting conventions are column sensitive, i.e. the first column that data starts on a line determines how that line will be handled.

What does all this mean to you? Modify data carefully keeping to the same columns already used (unless you feel brave.) Some will be free format and some will be fixed (look for the !format line to know which is which).

The easiest way to create a new question by copying an existing one that is similar in format to the data you want to present. You will also need

bonnie.wpd:/depot/cgi_info/FAQ/faq.sgi/questions/TOC
which defines the order in which the questions and answers appear.

Ask Tom Murphy for any help in decoding these methods. The system is in the process of being migrated from one of his personal use to that of a more general use. Q3 indicates that Tom is still doing the majority of the maintenance. Help is always appreciated.

Q5: How do I report errors and new questions for the FAQ?
(Author is murphy)

Currently, the easiest thing to do this is to send email to Tom Murphy (murphy@wpd).

If you are feeling adventuresome, then create a ptools workarea and move to directory bonnie.wpd:/cgi_info/FAQ/faq.sgi/questions. Each question is in a separate file. The file TOC is the table of contents file and sequences the various question files. Update an existing question via ptools. To add a new question, just create a new file, and add a line of the form

```
##file name##
```

at an appropriate place in the file TOC. The adventuresome part is following the formatting and cross reference conventions outlined in the file bonnie.wpd:/cgi_info/FAQ/README.

Q6: What other sources of information are there?
(Authors are murphy,, and scott)

There are a variety of information tools that Tom Murphy offers (see Q5b). There is the Insight command for viewing online manual pages. (There is also a CSD version of this viewer complete with a specialized CSD bookshelf.) There are hardcopy SGI manuals (see Q21 for ordering information).

There is the atlas database of data oriented towards the field. You can load it by typing:

```
% tcp -r quest@atlas.corp:/cgi_field/atlas .
```

To run it (it will also reload a more current version if necessary),

94/05/03
11:18:58

type:

```
% cd atlas
% atlas.START.HERE
```

There is the SGI Info Server, an automated email server. To access it, send email addressed to:

```
info-info@relay.sgi.com
```

and it will send you email with how to get further info subjects. The instructions currently say to email to "info-@sgi.com", but that doesn't work from all domains, the address "info-@relay.sgi.com" should.

Q7: How do I get a copy of the Engineer's Handbook?
(Author is murphy)

The Engineer's Handbook is a collection of disparate descriptions of various parts of the SGI engineering environment. You can view it online via the handbook command. See Q55 to obtain the handbook command.

You can get a by definition at-least-mildly out of date printed version of the handbook command by FAXing your request to the copy center at 408 732-4463. Say you want Tom Murphy's Yellow and Green Engineer's Handbook. Give your department and division number, how many copies you want and your mailstop. You should have your manuals within two days.

Q8: How do I get a copy of Uncle Art's Big Book of IRIX (Cypress only edition)?
(Author is murphy)

You can view it online via the handbook command. Just type:

```
handbook -s dist:/sgi/doc/swdev/BigIrixBook
```

See Q55 to obtain the handbook command.

You can get a hardcopy by calling the copy center at 961-4014. Tell them you want Tom Murphy's Purple Engineer's Handbook. Give them your department and division number, how many copies you want and your mailstop. You will have your manuals within two days.

Q9: How do I get the Mountain View Coredump Newsletter?
(Author is murphy)

The Mountain View Coredump newsletter, the newsletter written by and for the denizens of the Mountain View campus, is readily available via a handy, dandy coredump viewer, conveniently called, coredump. You can obtain the coredump viewer by typing (make sure /usr/local/bin is in your search path):

```
% su
# inst f dist.wpd:/sgi/coredump
Inst> install coredump
Inst> go
Inst> exit
# exit
```

faq.sgi

```
% refresh
```

There is actually one of three versions of the newsletter you will pick up depending on which version of showcase you have (3 or not 3) or whether you want an ascii version. Of course, all versions are the same (modulo media).

The coredump viewer will copy the latest version to your workstation, if needed, and display it for you. The location of the three versions is dist.wpd:/sgi/doc. The file names are coredump2, coredump3 and coredump.a.

You can expect to see a new version of coredump out each month, whether this actually will happen or not, only skipky knows. Questions and comments to coredump@ocumi.corp. Interest in more details to the man page for coredump or the text of the coredump script.

Q10: How do I find out what SGI software is installed on my workstation?

To find out what base version of IRIX you are running:

```
% uname -r
```

(or use the describe command for greater and more useable info (See p62 for info on how to get it))

Use the 'versions' command to find out what software is on your system:

```
% versions -Inv
```

Knowing some terminology helps to interpret the output you get. SGI's software is packaged into "products", which are further subdivided into "images", which are further subdivided into "subsystems". Subsystems are collections of files and are the smallest installable unit (i.e. you can't install individual files by name). When you look at the output from versions, some lines are for products (such as the "nfs" line below), some are for images (such as the "nfs.sw" line), and some are for subsystems ("nfs.sw.nfs").

```
I nfs                c13922652  4D1 3.3 Network File System
I nfs.sw             1005000064  Network File System software
I nfs.sw.nfs         1005000064  NFS Support
```

For each SGI Software Product installed on your machine (and maybe some local software, too) there is a product line. (There is one exception to this: the basic software that comes with every workstation is broken into two "products", "eoe1" and "eoe2".) The release and alpha number for eoe1 represent the version of the OS you have installed. Look at the product line to get the release number (4D1 3.3 in the example above) and name of the product (Network File System), and at the last three digits of the Version column for the images and subsystems to get the alpha number of the product (64 in this case). The first four digits represent the version of the product, typically same as the version of the OS (1005 in this case i.e. OS 3.3). The middle three are the maintenance release level, typically zero. You might also see products with names like "maint1". These product names are used for maintenance releases and contain files that replace their counterparts in the major releases of the products.

Known IRIX ALPHA numbers:

```
OS          Alpha
```

94/05/03
11:18:58

4.0.1	406
4.0.2	526
4.0.3	584
4.0.4	588
4.0.5	729
4.0.5A	731
4.0.5MM	1190
4.0.5D	2872
4.0.5E	2885
4.0.5F	2902
4.0.5G	3075
4.0.5H	3353
5.0	303
5.0.1	451
5.1	68 (may change)

To determine the kernel build time (mmddhhmm = month, day, hour, minute):

```
% uname -v
```

Known kernel build times:

OS	mmddhhmm
4.0	08281003
4.0.1	11150233
4.0.2	02211459
4.0.3	03091138
4.0.4	03241739
4.0.5	06151813
4.0.5C	06151813
4.0.5MM	06240105
4.0.5E	08280217
4.0.5F	08280217
4.0.5G	12171207
4.0.5H	03051335
5.0	03172037 (may change)
5.1	07221513 (may change)

Q11: How do I get SGI software releases for my machine?

First off you need to decide which software release machine to access. (Note that dist.wpd will typically be the most complete and definitive source of both released and yet-to-be released software.) See Q74 to help decide this. Typically, it's

dist.wpd	CSG
stress.asd	ASD
mars.esd	ESD
atlas.corp	CORP
release.csd	CSD

You will also need to know how to use 'inst', our software installation tool (see Q13).

Software releases that have been MR'ed appear in the directory:

```
dist.wpd:/released
```

Released images may also be available on other systems, such as

faq.sgi

```
mars.esd:/4D  
atlas.corp:/4D  
release.csd:/4D  
stress.asd:/4DTEST
```

Yet-to-be-released alphas of the operating system can be found in these distribution directories (this service is provided by the release group):

```
dist.wpd:/test  
  
alpoint:/4DTEST  
atlas.corp:/4DTEST  
babylon.wpd:/4DTEST/5.1  
bennie.asd:/dist  
mars.esd:/4D  
mom.media.corp:/4DTEST  
release.csd:/4DTEST  
sinead.wpd:/4DTEST  
stress.asd:/4DTEST  
summit.wpd:/4DTEST  
viper.esd:/4D
```

The current list of these alpha machines can be found in the file dist.wpd:/usr/dist/tools/data/alpha.db which is maintained by Ken Eubey.

Q12: How do I find out, in advance, the pitfalls/bleeding edge/campus alphas?
(Author: is murphy)

Look to the newsgroup for the flavor of alpha you are about to install to learn insights culled by others. sgi.alpha.sherwood is the one for the irix52 alphas. sgi.alpha.redwood is the one for the redwood alphas. Archives for these newsgroups exist (See Q16).

Talking in the halls is also useful, particularly, if it is with other engineers who can light the path ahead of you.

Q13: How do I use inst to install software on my workstation?

inst is used to install both externally released software (at all stages of development) and software intended for internal distribution only. Some of this software must be installed from the mini-root when the system is idle (or by doing a live install, see Q16). This is typically kernel software. See Q15 for help installing from mini-root. When running inst you will see the question

```
"Are you installing software from a remote CD ROM drive? [n, y]"
```

The answer is currently "n" when loading from all of the standard places.

You can see what software is available for installation from a <machine><directory> by typing

```
% su  
# inst -l <machine><directory>  
inst> list
```

94/05/03
11:18:58

faq.sgi

A typical entry is:

```
i X picmail.man.picmail * 0 0 Man pages for picmail
```

The 'i' indicates that installation of new software is requested. A 'k' would have appeared if you already had the current loaded (or if you had indicated you did not want it loaded). An 'r' would appear if you had requested that the software should be removed. More on 'k' and 'r' later.

The 'X' indicates an older version of the software is already installed. Other possibilities are an 'I' if the software is already installed or an 'N' if a newer version has already been installed. You will not typically see the 'N' since it indicates the software you are considering to load is already outdated.

'picmail.man.picmail' is the name of this software. The name is of the form product.image.subsystem. SGI software is packaged into products, which can be referred to within inst by just the product name. In this case, 'picmail' refers to all picmail software and man pages. A product is subdivided into images, which can be referred to by the product and the image. In this case, 'picmail.man' refers to all man pages associated with the picmail product. The smallest installable unit is the subsystem, which is referred to by the full name. In this case, 'picmail.man.picmail' refers to the man pages for just the picmail program. A subsystem may be composed of arbitrarily many files.

The '*' indicates that this software will be loaded unless you specify otherwise. A '+' could also appear here which indicates that the software is required for basic system functionality. An '@' could also appear here indicating that this software must be installed from the mini-root.

The next two numbers indicate how many additional blocks are required in the root and /usr partitions respectively. The final entry is the description of this software.

You can unselect software for installation by typing one of following

```
Inst> keep <product.image.subsystem>
Inst> keep <product.image>
Inst> keep <product>
```

You can unselect all software for installation by typing

```
Inst> keep *
```

Conversely, you can select software for installation by using the keyword 'install' instead of 'keep'. You can remove software by using the keyword 'remove' instead of 'keep'. An advantage of using inst to maintain your software is that files from any older version are correctly replaced by the files of the latest version being installed.

If any of the software to be installed is marked as requiring the mini-root, then you must exit inst and load the software using the mini-root. See Q15 to load the miniroot.

inst has an interactive help system available at any point by typing

```
Inst> help
```

When you are done selecting software to install, you can begin loading software by typing

```
Inst> go
```

When installation completes, you can complete installation and exit inst by typing

```
Inst> exit
```

At this point, there may be some final commands executed to complete installation of your software.

Q14: What if inst does not install everything I thought it would?

If you did not use the -a option to inst (to automatically install software), then you either asked for the the wrong product/image/subsystem to be installed or the one you asked for doesn't contain what you thought it did.

If you did use the -a option, then either the software you want is not marked for default installation or inst was asked to remove it at some previous time (it remembers not to automatically reinstall it). In any event, you will to manually install. For instance, suppose you want to install infotools.sw.picmail. Lets do a list to see everything that is available and then do the install.

```
% su
# inst ... (remember not to include the -a option)
Inst> list
Inst> install infotools.sw.picmail
Inst> go
Inst> exit
# exit
%
```

Q15: How do I load software using the miniroot?

Well, you can begin reading from the next paragraph to learn how. You might first ask yourself, "why?". It is possible to avoid doing a miniroot install (most of the time). See Q16 for more details.

If you can get by installing software from UNIX command mode, do it. See Q13 to learn how to make this determination. The typical time you must load from the miniroot is when you load a new version of the OS.

First step is to bring your system down to the monitor. Type:

```
% su
# init 0
```

The system will now spend a few moments and screens shutting itself down. When you see the message "Press any key to restart", then press a key (and be prepared to press the escape key). You will now see "To perform system maintenance instead press ESC", which you do.

You should now be on the "System Maintenance Menu". Press 5 to enter the command mode. Type:

```
>> setenv notape 1
>> exit
```

You are now back on the system maintenance menu. Press 2 to install

94/05/03
11:18:58

faq.sgi

system software. You should see the message "Are you using remote tape? (y/n)". If you do not then call the geometry hotline (see-Q17). Request a PROM upgrade. You can read the "IRIS Software Installation Guide" or ask a friend to help you load software while waiting for the upgrade.

Back to the normal case. Answer "n" to the remote tape question. You will now see "Enter name of machine which contains the installation software". Enter the appropriate software source (see Q11), for instance, "dist.wpd:/test/cypress".

You will have a wait from a few moments to a few minutes till you see the response of a series of dots trailing across the screen. The miniroot is loading across the network which can take from several minutes to more than a hour depending on network load. (Midday is not necessarily the best time to load, nor the moments after the announcement of a new alpha version of the OS).

The miniroot initializes and automatically loads inst. You can now continue with the use of inst as outlined in Q13.

Q16: How do I do a live install thus avoiding a mini-root install?

With some caution. It is an undocumented feature (since it is for the campus and not customers). You can safely use it to install a new alpha. You may not be able to use it between major (or even minor releases). For instance, you cannot use it to pass from 4.0.5 to 5.0.1, since this involves directory reorganization.

Make sure you have installed noship.sw.inst. All then have to do is invoke inst with the -L option and all those subsystems which formerly required miniroot (such as eoe1 and eoe2) can be installed via the command line.

What is behind all the smoke and mirrors (you might ask)? Well, all those files that must not be installed while the system is running, aren't. Instead they are saved off to the side and are installed at reboot time via an /etc/rc script. Note that this requires extra disk space. In general, count on needing an extra 30 megabytes in /usr, and 5 in /. If you don't have this space, inst will fill up the disk, and fail horribly.

Q17: How do I get hardware and software support for my workstation?

RTFM is an oft heard response to this question. The manuals are a good source of information, albeit several feet of goodness to pore through.

You can call the Geometry Hotline (800-800 4744) for support of both SGI hardware and SGI software. This is the CSD hotline that customers call. Support contracts exist for our workstations thus we are customers. You will need to have at hand the serial number of your workstation. You can get help over the phone or arrange to have a technician visit your office to work on your machine. Office visits usually happen by the next day.

Another source of information is the I/S help desk at extension HELP (3 4357). This is solely internal support. The help desk provides software support for SGI machines, but not hardware support for SGI machines. It provides both hardware and software support for VAXen and MACs.

The easiest source of software support is to question and answer among

ourselves via netnews or email. Face to face contacts can be more fruitful, but also can divert us from our individual directions. When appropriate, exhaust the above sources before your colleagues. You are the best judge whether you have a standard CSD type question or one only the developer will know. Q24 will help you find out where software lives on the source tree. You can run the command rlog in that directory or p rlog in the corresponding directory of your workarea. This will give you the names of the SGI engineers who have worked on the code and are thus good candidates to ask questions of.

Remember this faq, i.e. if you come up with a question and track down the answer yourself, pass both to Tom Murphy for inclusion in the faq.

Q18: What is a maintenance software release?

A maintenance release is an SGI software release who's number has two periods in it (i.e. 4D1-3.3.2). It has a number of special characteristics:

- o It contains a set of files designed to fix problems, add features or support new hardware for a particular "base" or one period release of system software (i.e. 4D1-3.3) and all of the optional SGI software products that are compatible with that system software release.
- o Maintenance releases are cumulative, meaning that all of the fixes in 4D1 3.3.1 are also contained in 4D1-3.3.2. So, you can install 4D1 3.3.2 on a 4D1-3.3 system that has not had 4D1 3.3.1 installed and still get all of the 4D1-3.3.1 fixes.
- o There is a special algorithm for naming maintenance subsystems: if the original subsystem is "<product>.<image>.<subsystem>", the maintenance release of that subsystem will be called "maint<n>.<product>.<image>.<subsystem>". <n> is an integer that can vary from maintenance release to maintenance release.
- o You should never use inst or versions to remove maintenance releases since you will lose important files. For instance, if a new version of the 'ls' command is included in 4D1 3.3.2 and you remove 4D1-3.3.2, then you will no longer have any copy of 'ls' on your system.
- o The best way to install maintenance releases is to use automatic installation. Inst will automatically give you maintenance subsystems for all of the matching base subsystems that you have installed which is exactly what you want.

Q19: How do I interpret the ten digit version number of a release?
(Author is murphy)

Just use the showversion command to do this. Note that it also attempts to decode version numbers from older numbering schemes and not just the one we recently began using.

You can get the command (as well as the mkversion command and other build tools), by typing the commands:

```
# su
# inst -af dist.wpd:/test/<release name>/nontoto/noship
# exit
```

94/05/03
11:18:58

& rehash

The man page for mkversionnum details the components of a version number, namely:

Case 1: regular release - MMMAAAAABT

MMM - Major number - top 3 digits (101-213)
(It starts with 101 to preserve monotonicity from the preceding version numbering scheme. It can at most be 213 since we must have a bit unsigned version number) This will represent the first two parts of a release number (e.g. 5.1). It replaces the major and minor numbers of the old version numbering scheme. The build group will typically assign this, setting the macro RELEASE_NUM in \$(ROOT)/usr/include/make/releasedefs.

AAAAA - Alpha number - next 5 digits
This represents the last 3 or 4 parts of a release number (e.g. the 2 part of 5.1.2 or the 2.3 part of 5.1.2.3). It replaces both the old maintenance number of the old MR version numbering scheme as well as the alpha number of the developer version numbering scheme. It is automatically generated and is the number of hours since midnight Jan 1, 1993 GMT.

B - Builder - 1 digit
0 if developer built.
1 if project built.
2 if build group built.

For instance, Jack Repenning might build a Tracker image and he would use a 0. Linda Kvarda might build Tracker as part of a build of all the CASE tools and she would use a 1. Aimee Manosa would build Tracker as part of an entire release and she would use a 2.

T - Tree Identification - 1 digit
This digit enables a version number to distinguish among parallel OS efforts. It is normally zero. It will typically mark an OS development tree targeted for a particular new hardware platform. The build group will typically assign this, setting the macro TREE_ID in \$(ROOT)/usr/include/make/releasedefs.

Case 2: patch - RRRRRRRRPP

RRRRRRRR - Release number - top 8 digits
This is the major and alpha numbers of a previous MR'd release.

PP - Patch number - last 2 digits (30-99)
Starts at 30 to distinguish a patch version number from a regular alpha. Defined by the PATCH_NUM macro. A value of 1 (signifying patch 1) will generate a 30, and so on.

Note that we start from 30 to distinguish from a regular release which would have a value from 00 to 29, representing the builder and tree identification.

Q20: How do I set the version number for an inst image I am making?
(Author is murphy)

Well all you have to do is be at the top of your ISM and type the command "make startversion". The version number thus generated will be the one used in the images you create. You probably will want to get in the habit of generating your version number just before you begin generating

faq.sgi

your executables, since we will shortly be able to embed the version number. as an ident string, within your executable.

What if the word ISM above rings no bell with you? Use the handbook command to read the chapter on "ism". This will help you know the process SGI developers use to structure their products into both an inst'able image and easily hand off their efforts to the build group. See Q55 for information on getting the handbook command.

You will need a current public ROOT and TOOLROOT to get the mechanism necessary for the startversion target to work. Alternatively, you can get current versions by installing the buildtools root and toolroot images in your local ROOT/TOOLROOT. They are available from babylon.wpd:/sherwood/LATEST/root/build.root and babylon.wpd:/sherwood/LATEST/toolroot/build_toolroot, respectively.

Q21: How do I get copies of SGI manuals?
(Author is murphy)

I would be remiss, if I did not first suggest you consider Insight. More and more documentation is available online and it is a powerful viewer. See Q22 for more information on Insight.

Type 'handbook manuals' (see Q55 to get the handbook command) to view the SGI manual section of the price book (or borrow a copy of the price book from your admin). Note that the internal price of the manuals is less (or significantly less) than the external price shown.

Fill out an MTR with the manuals you need, get your manager's signature and give it to your admin. The MTR will end up in building 4 where your order will be filled. Jose Sanchez is a good contact should it seem that your order is moving too slowly. Be aware that slowness at end of quarter is quite natural as customers preempt internal orders.

There are only a few MTR fields that need to be filled out:

- lower left hand corner box contains info about you
- get part number and description from the price book
- U/Z is EA (for each, the unit of measure)
- QTY REQ is typically 1
- upper right hand box
- Dept (better match what you already said in lower left corner)
- ACCT is typically 69501
- FINANCIAL APPROVAL is your manager's signature

Questions about this manual ordering process for engineers can be directed to Tom Murphy (murphy@wpd).

Q22: How do I get the InSight viewer and the bookshelves to view?

There are two releases of the IRIS InSight Viewer available. The first release is a Cypress-based product and called InSight 2.0. The second is a Sherwood-based product and is called InSight 2.2.

CYPRESS

You can load the Cypress product from the standard OS distribution location like dist:/released/4.0.5/411 or you can load strictly the

94/05/03
11:18:58

InSight product family images from dist:/released/4.0.5/InSight.

To load the viewer the following images should be installed:

- insight
- insight_style

Load the shared glossary book, if you load any bookshelves (i.e. if no NFS or CD):

- insight_gloss

Books are in the following images:

- insight_eee End User/Admin books
- ido Developer books

The CSD Support bookshelf is in the images:

- insight_lib
- insight_lib_spare

SHERWOOD

The InSight 2.2 viewer is available in the complete OS package, so load the InSight product, i.e. begin installation of your friendly neighborhood OS and at the inst prompt just say:

```
Inst> install insight
```

In Sherwood the books are located with the products they document. You can see what books are available by typing:

```
Inst> list *.books
```

The InSight viewer is using the new SGI Help system so the sgihelp product should also be installed:

- sgihelp.sw.ooo part of 5.1 nontoto/toto
- sgi.books.ViewerHelp Help System's Help

End of SHERWOOD

There are different books available with each version of the OS. The books available with 4.0.5 are not all available with 5.1, but more are being added every release.

The CSD Support library is currently documenting Cypress and not Sherwood. If you install the Support bookshelf along with the Sherwood books, the links between the two collections might not work.

See the release notes for InSight to learn how to NFS mount the books or run them from a CD. There are plans to have NFS mountable versions of all released books (at some point where the insanity slows to just mild bedlam).

See Q11 for info on locating friendly neighborhood software.

faq.sgi

Q23: How do I find the source for SGI software?

Source for SGI software is stored on "trees". Each tree is a more or less complete copy of the UNIX source hierarchy with additional source we've acquired from elsewhere or developed internally. Software for future releases of SGI software products will be created from these trees. "Spec files" in usr/src/cmd/inst/spec and "IDB tag" information in Makefiles define what actually goes into a release from a compiled tree. The source for historical SGI software releases is contained in RCS files on each tree, but the version used to create a particular release is generally not marked.

Q24 will help you locate specific files on a source tree. Q25 will help you find information about a specific product.

The tree nicknames, physical locations and purposes are:

Aspen/Birch
clyde:/aspen/att/usr/src

The aspen/birch tree is for future maintenance releases for 401 3.4.

Birchbark
rouge.esd:/d2/bark

The software release for Magnum, the 40/35, was built from this tree.

Cypress
jake:/cypress/att/usr/src

The Cypress tree is the source tree for the Cypress software release (401 4.0).

Crimson
lone.wpd:/lonestar/att/usr/src

The Lonestar tree is the source tree for the Crimson hardware release, and later became the general 4.0.5 tree.

BVD
jake:/bvd/att/usr/src

Source for the "BVD" release, the last 4.0.5 release.

Sherwood
jake:/proj/irix5.0/isms
jake:/proj/irix5.2/isms
jake:/proj/irix5.0.1/isms
jake:/proj/irix5.1/isms
jake:/proj/irix5.1.1/isms

This is the current "state of the art" in SGI's source trees. ISM stands for Independent Software Module, and is intended to split up the tree under logical boundaries, instead of a huge, monolithic tree under usr/src. Core Irix code is under isms/irix, isms/x is the X tree, etc. These trees are managed rather carefully, and it is often required that you "tag along" any source checkins to these trees. Please make sure you know what you are doing.

Tag alongs are an aspect of our source code management. See Q79 for more info.

94/05/03
11:18:58

faq.sgi

Jake
jake:/jake

The jake tree has historically been the place for source code changes that will be part of the "future" release, where a future release is cloned from jake when the appropriate time comes. That is no longer the case. The "future" release now is carefully defined, and changed with time. At the time of this writing, jake:/proj/redwood/isms is the "future" tree. That will undoubtedly change.

The source used to create historical SGI software releases is in tape archives administered by the Software Release Group in SSD.

Q24: How do I find where source is located within a source tree?

Suppose you have the source machine nfs mounted at /net/<source_machine> and you have rfind installed (See Q61 to install rfind). This also assumes that the rfind server is running on the source machine. See Q61 for a list of current servers. Type

```
% rfind <source_machine> <file_name>
```

Suppose you want to find the files associated with a <product.image.subsystem> that is installed on your workstation. Given the file names you can use rfind to locate them on the source tree. Type:

```
% versions -s <product.image.subsystem>
```

Suppose you want to find the files associated with a <product.image.subsystem> that are not installed on your workstation. Type:

```
% rlogin <distribution_machine> -l guest  
% cd <distribution_directory>  
% grep <product.image.subsystem> *.idb
```

This will produce one line for each file found composed of blank delimited fields. Field 7 is the product.image.subsystem of the file. Field 6 gives the relative location on the source tree. It is usually relative to /<release_name>/att/usr/src.

Q25: How do I find out about a specific release using the whatswere command?
(Authors are beths, and murphy)

Type the following lines to get whatswere on your machine:

```
% su  
# rcp guest@quixote.wpd:/usr/local/bin/whatswere /usr/local/bin  
# exit  
% rehash
```

The release group, (release@wpd), maintains the data accessed by this command. There is information on a variety of subjects. This list was generated by executing the command whatswere with no parameters. The whatswere database is updated frequently.

3270_4.0	dmdev	picmail
4.0.2	dmedia_1.1	ptools
4.0.3	e+ 2.0	pv
4.0.4	explorer 2.0	rsvp
4.0.5	faq	sherwood
4.0.5A	ftam	showcase 3.0
4.0.5E	handbook	snaview 1.0
4.0.5F	imagevision_2.0	svideo
4.0.5G	impressario 1.0	tinyui
4.0.5H	impressario_1.1	tli
4.0.5MM	insight2.0	token 1.0
4DDN_3.0	insight 2.0	token_2.0
4DLT	inventor_1.0.1	vtramer_2.0.1
5080_2.0	inventor 1.0	volmgr_1.0
5080_2.1	locate	whatsubsys
5080_3.0	netvis 2.0	whatswhere
Avalanche 1.0	newdirorg	x.400
FDDIXPress_3.0.1	news	x25
FDDIXPress_3.0	nn	xlocate
Public_root	osi	
Tracker_1.0.1	performer_1.0	

For instance, to find out about 4.0.5H you would type:

```
whatswhere 4.0.5H
```

and you would see:

```
Information on 4.0.5H
```

```
-----  
Release:                4D1 4.0.5H  
Source tree:            bonnie.wpd:/bvd/att/  
MR date:                3/05/93  
Build Frequency:       Monday's and Thursday's
```

```
-----  
***** Build Status
```

```
-----  
Alpha:                 314      sa, eoe1, eoe2, and maint ql x_dev
```

```
Build Date:            2 19 93
```

```
Build Machine:        godzilla.wpd  
WORKAREA:             /lv0/4.0.5H/  
ROOT:                 /lv0/4.0.5H/  
TOOLROOT:             godzilla.wpd:/syda/4.0.5G/TOOLROOT 2.0
```

```
Build Log:            godzilla.wpd:/lv0/4.0.5H/logs/
```

```
Images:               dist.wpd:/test/4.0.5H/          (Newest alpha)  
mars.esd:/4D/fullhouse/test/ (Newest alpha) mars.esd:/4D/fullhouse/  
                    (Stable alpha)
```

```
-----  
***** Software Status Information
```

```
-----  
IRIX 4.0.5H (Fullhouse)
```

Business Team Mtg:	Mondays: 2:00 7L Visual Simulation Room
Project Manager:	Jamie Piotto/Dave Olson
Product Manager:	Jay McCauley
CSD Representative:	Mike Feng/Lynne McDonald
Tech Subs Production:	

94/05/03
11:18:58

Tech Pubs Writer: Arthur Evans
Doc Control: Rod Armer
Build Meister: Arsenio Briones

Q26: How do I find out what version of Irix in which a particular bug first appears?
(Author is murphy)

Unfortunately, there is not a direct answer, but luckily, the indirect answer is pretty good.

If you look up the bug in pv (see Q83 to get pv), and the bug is closed, you just need to look in the TAKE message to see in which source tree the fix was put. In most cases, the source tree name will be the same as the release name. (Current exceptions are sherwood (which is 5.3) and redwood (which is 6.0). You can do a 'whatswhere tree_name' which should unlock the tree-release name mystery as well. (See Q55 to learn how to get the whatswhere command).

If all else fails, email Bob Green or Mary Ann Gallagher who know much of these matters.

Q27: Is there a place from which I can NFS SGI reference material?
(Author is msc)

As a matter of fact, there is and it involves use of the /usr/share filesystem.

Introduction

Shortly before IRIX 5.2 MR'ed, I discovered that it was impossible to install the insight client software when your /usr/share is mounted from another host. The priority 1 bug I filed was made an exception with the comments that a) the release notes should say that we do not yet fully support /usr/share and b) we should publish some guidelines about /usr/share for the benefit of our internal developers.

Since I don't think anyone else is writing any guidelines and since I have a bit of time right now, I decided to do it.

I welcome comments and additions to these guidelines.

Purpose Of /usr/share

/usr/share is intended as a repository for machine independent executables and data files which can be shared between hosts by cross mounting the entire directory. The main goal in inventing it was to reduce the amount of disk spaced required on the local host. It meets this goal principally by making it easy to nfs mount this huge amount of data.

Prime candidates for inclusion in /usr/share are things like the Insight books, the man pages, and the huge files of clip data for applications such as Inventor, Explorer, Showcase and the digital media tools.

It is important to note that /usr/share will typically be mounted read-only.

faq.sgi

Guidelines

Given the purpose, three prime requirements become immediately obvious.

1. A software product planning to install files in /usr/share must put those files in a separate subsystem and at least the description of that subsystem must say that it installs in /usr/share.
2. A product must never have hard prerequisites on something in /usr/share because, on a machine with a mounted /usr/share, inst will not be able to tell that the prerequisite software is present.
3. A product must never use /usr/share for any file intended to be written to in the normal course of the product's operation.

If these requirements are met, a product will be fully /usr/share friendly.

Number 3 raises the issue of where to put items that are to be available for collaborative update. Examples of these items are annotations for the Insight books, adapted templates for speech recognition and collaborative databases such as edman's music catalog. Since many sites want to mount the entire /usr/partition read only these items should not be put anywhere in /usr. Such items should be placed in /var/public. Each product should have its own subdirectory (e.g. /var/public/Insight).

Q28: What mail programs are available and how do I install them?
(Authors are jackr, rck, pdc, and davidp)

The 'mail', 'Mail', MH, 'xmh', 'zmail', and elm mail systems are all in use at SGI and described below.

In order to successfully send mail with any of the mail systems, you will need to run a local program called 'GETmail' to configure your workstation for SGI's mail delivery environment. You will need to periodically rerun GETmail to refresh your workstation with a correct view of the SGI mail network. To get a copy of GETmail and run it (substitute your domain name for <your domain>):

```
% su
# rcp guest@relay.<your domain>.sgi.com:/etc/hosts /etc
# rcp guest@relay.<your domain>.sgi.com:/usr/sbin/GETmail /tmp
# /tmp/GETmail -h relay.<your domain>.sgi.com
```

(Author of this section is unknown) mail

This is standard issue AT&T UNIX mail and is part of the software shipped with every workstation.

Mail

This is Berkeley Mail with enhancements. It is part of the software shipped with every workstation.

MH

The MH mail system, also known as Band mail, is not shipped with any SGI products. It is available internally (including man pages):

94/05/03
11:18:58

```
% su
# rcp guest@anchor.esd:/usr/local/install/install.anchor \
  /usr/local/bin
# /usr/local/bin/install.anchor mh
```

xmh

xmh is a point-and-click with-windows interface to MH. It is shipped standard with every workstation. It is not installed by default, but is included in the eoe2.sw.Xapps subsystem. Prerequisites for running 'xmh' are that you have to be running the X Window System and that you have MH installed on your system (see above).

Zmail (Author - of this section - is davidp)

Zmail is mail system with three user interfaces, the most significant being an X based graphical one.

You can install zmail and its manual pages from either of these distribution machines using the standard installation tool inst (see Q28):

```
atlas.corp:/usr/dist/images/zmail
mars.esd:/40/sgi/zmail
```

The MOM (Move over Mac) installation set-up is the default. The original system.zmailrc file is included in /usr/lib/Zmail under system.zmailrc.orig, just move this to system.zmailrc and you will have ZMail as it comes off the tape.

Elm (Authors - of this section - are rck, and pdc)

Elm is an interactive screen-oriented mailer program developed at Hewlett-Packard. Elm 2.4 at patchlevel 21 is available for internal use.

To install elm and its manual pages, type the following:

```
% su
# inst -f dist.wpd:/sgi/hacks/elm
# exit
% rehash
```

Elm documentation (PostScript files) are included in the inst images as part of elm.man.elm, and get installed into usr/local/lib/elm. It contains compressed copies of the Elm Mail System manuals. To print them on a PostScript printer just type:

```
% uncompress < Alias.PS.Z | lp
% uncompress < Config.PS.Z | lp
% uncompress < Cover.PS.Z | lp
% uncompress < Filter.PS.Z | lp
% uncompress < Form.PS.Z | lp
% uncompress < Ref.PS.Z | lp
% uncompress < Users.PS.Z | lp
```

Questions on elm can be sent to Robert Keller (rck@asd) or Paul Close (pdc@asd).

GNU Emacs (Author - of this section - is jackr)

GNU Emacs can serve as a mail program. It provides 'rmail', mh-mode, and xmh-mode (usable with or without xmh). See Q57 for more

faq.sgi

information about GNU Emacs.

Q29: What do I do if I'm having problems sending mail to people at SGI?

As a rule of thumb, if you have been successful sending and receiving mail in the past, but now your mail is getting returned by the MAILER DAFEMON or you are having any other type of delivery problem, the first thing to do is to run GETmail. It will solve your problem much of the time.

```
% su
# /usr/sbin/GETmail
```

If you are sending mail to someone within SGI, but in another domain and it is not properly addressed ("properly addressed" varies from domain to domain), it can take a week before it is returned to you as undeliverable. In this case, adding the person's domain name (i.e. 'mail so and so@esd') or sending the mail through the machine sgi (i.e., 'mail so and so@sgi.com') might do the trick.

If you're stuck, system administrators can often help sort out mail delivery problems. You can reach them by posting to sysrequest which will get to the network administrator for your domain. You can post to sysrequest@otherdomain to mail to a network administrator in another domain. For instance, mail to sysrequest@wpd to get to Ian, network and espresso guru for SSTC. As a last resort you can post to the newsgroup sgi.mail.admin.

Q30: What's "metamail" and why can't elm read it?
(Author is philv)

Look in /usr/lib/Zmail/bin. Metamail is a part of zmail/mediamail that allows elm to detach and/or read attachments. Elm can't determine all types of attachments with metamail, but it reads most. Put a copy of metamail in /usr/local/bin and elm will find it.

Q31: What are the different types of newsreaders?
(Author is olson)

There are two main methods of reading news (ignoring user interface). See Q32 for info on specific newsreaders.

One method is to NFS mount the news spool area from a server (typically sgi.sgi.com). Posting new articles results in the article being mailed to a special alias (often invisibly to the user). This was the traditional method at SGI, until the company grew too large. The readnews and vnews newsreaders require this method.

The other method is using a network protocol called NNTP (Network News Transport Protocol). Originally designed for transmitting news from one system to another, it was quickly picked up for newsreaders. In general, this requires no NFS mounts (but see the information about nn below), but instead opens a network connection to a particular news machine. At this time, there are a number of NNTP news servers. You should use the one closest to you (network wise). To select a server, you typically change a file containing the server name (i.e. /usr/local/lib/nn/nntp.server), set an environment variable (usually NNTPSERVER), or change a resource file. Typical NNTP newsreaders are nn, gnus, xrn, rrr (often distributed as just rn); there are many others.

94/05/03
11:18:58

faq.sgi

Examples of NNTP news servers are: odin.corp.sgi.com (company wide, on the backbone), zuni.esd.sgi.com (esd only, located in bldg 2) zola.esd.sgi.com (esd only, located in bldg 3), fido.asd.sgi.com (in building 7), and twilight.wpd.sgi.com (in building 9). There are other servers also available, and the list will probably continue to grow.

With either type of newsreader, it is painful to change from one server to another, as different servers may have different sets of groups, and certainly have different article numbers for the same articles. The easiest way to change servers is to catch up on all the groups you care about before changing servers, change to the new server, edit your ~/.newsrc file to remove all article numbers (everything after the ':'), use your newsreader option to catch up on all unread news (everything current), and then start using the new server to read any new articles. You may miss a few articles in the transition, but some newsreaders provide a way to say "mark all articles less than N day(s) old as unread", so you can then make sure you don't miss anything, at the minor cost of being offered some articles that you have already seen.

Q32: What news readers are available and how do I install them?
(Authors are scotth, weissman, mikey, olson, and pdc)

For information on the two different styles of newsreaders see Q31

nn (Author - of this section - is olson)

nn is a menu based netnews reader. To install it:

```
% su
# rcp guest@anchor.esd:/usr/local/install/install.anchor /usr/tmp
# /usr/tmp/install.anchor nn
# exit
% rehash
```

All the files go into /usr/man/man1 or /usr/local, except for /README.nn.sgi, and /init.nn.sample. One of the exit ops will mount the nn database from newmedia.esd. It will also set your default news server to be odin.corp.sgi.com (in /usr/local/lib/nn/nntp.server).

All newsservers, other than odin, have an nn database on the same machine. These newsservers are listed in Q31. If you change your newserver you will need to change both the NFS mount point and the nntp.server file.

Not to confuse things, but another source of nn (using a different database format from odin), is available via:

```
inst -f atlas:/usr/dist/net_services/nn
```

rn (Author - of this section - is scotth)

rn is an NNTP news reader (sometimes called just rn). To install it:

```
% su
# inst -af quest@atlas.corp.sgi.com:/usr/dist/net_services/rn
# exit
% rehash
```

This version fixes the "too many lines in .newsrc" problem.

The default newserver is "news", which if your domain does not have one will not resolve. You can override this by either setting the environment variable "NEWSSERVER" or editing /usr/local/lib/rn/nntpserver with the name of your news server.

tin (Author - of this section - is pdc)

tin is a screen oriented, menu based news reader. It allows one to select news articles from a menu sorted by subject or date, organized into logical threads, or conversations. It is based on rn, so all the rn mechanisms are preserved. Users who currently use rn should definitely consider switching to tin! tin also can be run 'as' rn for backwards compatibility. tin currently gets much more frequent bug fixes than rn, too.

To install it:

```
% su
# inst -f dist.wpd:/sgi/hacks/tin
# exit
% rehash
```

This will create a /usr/local/lib/tin, and put tin, inews, inews, and Rmail in /usr/local/bin. The man page is installed in /usr/catman/local/man1. The default NNTP server is fido.asd.sgi.com. This version of tin requires an NNTP expansion for transport of threads, or INN overview files. Otherwise, tin will revert to rn style behavior.

xrn (Author - of this section - is mikey)

xrn is an X based NNTP news reader. It is not a front end to rn but rather a rn like graphic interface to net news. To install it:

```
% su
# rcp guest@joves.wpd:/usr/local/bin/install.mikey
# /usr/local/bin
```

If you're running Cypress:

```
# /usr/local/bin/install.mikey xrn
```

Otherwise, if you're running Sherwood:

```
# /usr/local/bin/install.mikey xrn sherwood
```

Then (whether Cypress or Sherwood):

```
# exit
% rehash
```

If you don't have the file "/usr/local/lib/rn/nntpserver", you'll need to add another line to your .Xdefaults file saying:

```
*nntpServer: odin.corp.sgi.com
```

Also installed is xrn motif, a Motif version of xrn.

gnus (Author - of this section - is scotth)

gnus is a GNU emacs interface to news via NNTP. It is standard in

94/05/03
11:18:58

faq.sgi

the emacs19 inst package (see the sgi Emacs faq entry).

vnews (Author - of this section - is olson)

vnews is available, but its use is strongly discouraged because of its impact on the network.

turn (Author - of this section - is weissman)

Turn ("Tiny UI News Reader") is a multi-threaded, threading newsreader. (That is, it has multiple light weight processes, and it groups articles together in trees based on the References line.) It is X-based.

One of the main goals of turn is to make the user wait as little as possible for the newsreader.

Turn also has features for displaying pictures or sounds, posted as uuencoded files split across many articles.

Turn requires a news server that supports either the XTHREAD or XOVER extension. The only news servers known to work with turn are zola.esd, fido.asd, and news.mti. (Actually, it will work with any news server, but will be very very slow if it doesn't have the XTHREAD or XOVER extension.) If you need to change news servers, the recommended way is to catch up on all the groups (using your old newsserver), remove all the numbers from your .newsrc, change news servers, start turn, and catch up on all the newsgroups (using the Alt-c command).

To install turn, do:

```
% su
# cd /usr/local/bin
# rcp guest@pianoforte.asd:/usr/local/bin/install.weissman .
# /usr/local/bin/install.weissman turn
# /usr/local/bin/install.weissman pictutils
# exit
% rehash
```

Q33: How do I get the rn that allows for more newsgroups in the .newsrc?
(Authors are lear, and scotth)

```
inst -f atlas.corp:/usr/dist/net-services/rn
```

Q34: What are the mail aliases that I can use to post to newsgroups?
(Author is roberts)

These aliases are brought to you courtesy of sendmail.cf. Your mail is routed to sgi.sgi.com for posting to the appropriate newsgroup. Old versions of sendmail.cf may not have the third alias listed below. This can be remedied by getting a new sendmail.cf. Type:

```
% su
# GETmail
# exit
```

Mail alias	Corresponding Newsgroup
------------	-------------------------

sgi.<name>	sgi.<name>
mail.<name>	mail.<name>
news.<newsgroup_name>	<newsgroup_name>

For instance, to post to comp.sys.sgi you can send mail to news.comp.sys.sgi.

Q35: What other useful campus aliases exist?
(Author is cwilson)

There are building aliases that exist. For instance, to alert the denizens of building 7 there is a car parked behind it with lights ablazing, just address your mail to 'b7'. These aliases are automatically generated and available in each domain from the locate database. Thus if someone is not on the alias there is a larger problem. Namely, "How do I get my email address to show up in the locate database?". For this you need only peruse Q72.

One caution, remember that this quickly alerts a whole lot of people. Think a bit before mailing.

Q36: Do we archive any of the SGI news groups?
(Authors are cwilson, and rck)

As a matter of fact, we do. Take a look in babylon.engr:/usenet. In it you will find archives of the most important campus newsgroups. Namely:

- sgi.bad-attitude
- sgi.engr.all
- sgi.engr.se
- sgi.general
- sgi.alpha.sherwood
- sgi.alpha.redwood

You will also find most of the comp.sys.sgi.* newsgroups.

You can also look to fido.asd:/usenet for additional newsgroup archives. Such as:

- alt.fan.john-palmer
- clari.feature.dilbert
- comp.graphics.explorer
- comp.graphics.opengl
- mail.apps.explorer
- sgi.engr.opengl
- sgi.engr.opengl.arb
- sgi.engr.opengl.nt
- sgi.engr.teton

Q37: What do I need to know to use our connections to the Internet?
(Author is vjs)

Silicon Graphics has two Internet connections, one via sggate.sgi.com and the other via sgi.sgi.com (also known as sgi.com outside the SGI network).

The connection to sggate.sgi.com is commercial and the machine is paid

94/05/03
11:18:58

faq.sgi

for and maintained by the organization in charge of the internal Silicon Graphics network. The connection to sgi.sgi.com is by contract only for "non-commercial research, development, and educational purposes," and is paid for and maintained by the R&D organization within Silicon Graphics.

Because sgigate.sgi.com is designed and intended for "operational" uses, it should be used for most commercial Silicon Graphics activities. Operating system releases, patches by anonymous FTP, and so forth should, whenever possible, be distributed from sgigate.sgi.com.

Information Services has purchased hardware and software that makes it possible for SGI employees to contact the SGI network through sgigate.sgi.com without causing security problems. Similar facilities are not available on sgi.sgi.com.

The public areas on both sgi.sgi.com and sqigate.sgi.com must not be treated as private datastorage. They are regularly cleaned because both machines have limited space, because people rarely clean up after themselves, and because of security problems caused by people outside SGI. Beware that efforts to subvert the mechanisms that clean the common areas are considered attacks on the machines, as "cracking".

Files of enduring, but primarily non commercial value can be made available from sgi.sgi.com by contacting postmaster@sgi.sgi.com. Please provide a pointer to a directory containing compressed data files and a suitable README files. Be patient, because two (2) weeks are often required. Remember that a failure to plan ahead before announcing the availability of files is rarely considered a crisis by anyone except the person who erred.

Q38: How do I find out where or what ftp material is available?

Reading the net will often answer both questions. There is also a net search program, archie, that will quickly locate stuff available from over 600 ftp archive sites throughout the universe. There are two main search commands (each taking a regular expression as an argument). "whereis" behaves like Unix whereis except it reports on ftp files. "prog" tells you at what ftp sites you can find the ftp files. You are probably going to want to tailor some variables. Likely candidates are maxhits (limits number of matches to display), search (whether to exact match instead of default regex search or one of other possibilities), sortby (default is none but filesize, time of modification, hostname and others are available). The help command will unlock this info and more. For instance to search for up to 1000 files with the substring kermit somewhere in its filename:

```
% rlogin sgi.sgi.com -l quest
SGI quest% telnet archie.sura.net      <- or quiche.cs.mcgill.ca
login: archie
archie> prog kermit
```

See Q39 for information on how to use anonymous FTP to obtain these files.

Q39: How do I get files via anonymous FTP from Internet sites?

Login as guest to either of SGI's Internet gateway machines, sgi.sgi.com or sgigate.sgi.com:

```
rsh guest@sgigate.sgi.com
```

or

```
rsh quest@sgi.sgi.com
```

This will create (if needed) and place you in a temporary directory based on your login ID. (On sgi, it's /usr/tmp/<login id>, on sgigate, it's /f/tmp/<login id>.)

Your files should only be stored temporarily on the machine. If you transfer large files, be sure to delete them as soon as possible because these machines do not have a lot of disk space. If you want a permanent copy of the files you get via anonymous ftp, transfer them to your machine.

You will need a hostname or Internet address for the ftp site you want to connect to, such as:

```
a.cs.uic.edu
18.24.0.12
```

To connect to an ftp site, enter the command:

```
ftp site name
```

When you are prompted for a user name (user id), try entering:

```
anonymous
```

You should get a message such as:

```
Guest login ok, type your name as password.
```

Some sites want to know who is using their ftp service. If you don't want to enter your e-mail address, you can enter any string of characters, or press the Enter key. Some very paranoid sites won't let you login without a valid email address, though.

If that does not work, try entering:

```
ftp
```

when you are prompted for a user name.

If your login is rejected, you will see something like:

```
User quest access denied.
Login failed.
```

Enter the command "quit" at the ftp> prompt to return to a UNIX prompt.

If your login is accepted, you should see something like

```
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

You can now enter an ftp command.

If you are not familiar with ftp, enter:

94/05/03
11:18:58

faq.sgi

or:

```
help
```

That will give you a list of ftp commands.

The most frequently used commands are:

```
ls
cd directory-name
ascii
binary
get file-name
mget file-name-with-wildcards (e.g., *.Z)
quit
```

The command 'ls' gives you a list of files in the current directory.

The command 'cd' is used to change the current directory.

The ftp command tries to determine if the remote system is UNIX so it can use the more efficient binary mode for all transfers. If you see this message before the first ftp> prompt

```
Remote system type is UNIX.
Using binary mode to transfer files.
```

then you don't have to use the 'ascii' or 'binary' commands.

The command 'ascii' is used to transfer readable ASCII files when the remote host is not a UNIX system. Files called 'README' or 'Index' are usually readable ASCII files.

Most files available via anonymous ftp are compressed binary files. Before transferring such files, you should enter the command 'binary.' If you transfer a binary file in 'ascii' mode, you will probably not be able to uncompress it or read it. It's OK to transfer an ASCII file in 'binary' mode if the remote host is a UNIX system.

The 'get' command is used to get a single file.

The 'mget' command is used to get multiple files. For example, to get all compressed files in the current directory, enter:

```
mget *.Z
```

You will probably get a prompt for each file asking you if you want to transfer that file. Enter 'y' to transfer that file. Enter 'n' to skip it. The specified files will be transferred into the directory where you entered the command 'ftp.'

FTP sites usually have a README file describing the files that are available for copying. If you want to view the file but save a copy in your directory, use this ftp command

```
get README |more
```

to display the file using the 'more' pager.

If you transfer several files and want to look at some of them while the remaining transfers finish, login to sgi/sqigate from another window on your workstation, and go to the directory to which the files are being transferred. Now you can read in the new window the contents of the

transferred ASCII files, and use the window in which you entered the 'ftp' command to continue entering ftp commands.

When you have transferred all the files you need, enter:

```
quit
```

You can reach the SGI ftp services from within SGI, by entering the commands

```
ftp sgi.sgi.com
```

or

```
ftp sqigate.sgi.com
```

Q40: Which local servers have network documents, such as RFCs, on-line?

odin.com:/usr/doc/internet has a complete list of Internet RFCs and Drafts. There are also some CCITT documents in that directory tree.

You can find the net documents shown in the table below as subdirectories of sgi.sgi.com:/usr/netdocs. For instance, Internet RFCs can be found in sgi.sgi.com:/usr/netdocs/rfc.

Directory	FTP site or source	remote location
dec/rfc	grabbaq.osf.org	cd dec/rfc
dec/src	gatekeeper.dec.com	cd pub/DEC/SRC/\nresearch reports
ietf	nssc.nsl.net	cd ietf
internet-drafts	ftp.nisc.sri.com	cd internet-drafts
iol-unh	public.iol.unh.edu	cd pub
isdn-atm	see isdn-atm/README	
netinfo-FTP.NISC.SRI.COM	ftp.nisc.sri.com	cd netinfo
netinfo-NIC	ftp.internic.net	cd netinfo
mbone	venera.isi.edu	cd mbone
osi	osi.mcs1.nist.gov	cd pub
rfc	ftp.nisc.sri.com	cd rfc
usenet	items seen in comp.doc newsgroup	

Q41: Do the Request For Comment files (RFCs) really contain a complete history of the Internet?
(Author is lear)

Yes. Read on.

What's an Internet RFC?

RFC stands for Request for Comments. Internet RFCs are texts that document problems, test results, protocols, jokes, and just about every aspect of the Internet, since its inception. Starting with rfc3.txt this collection details a technical (and to some extent political) history of what has become a global communication infrastructure, currently present in over thirty countries in all continents.

94/05/03
11:18:58

faq.sgi

Why Would I want to read one?

Aside from their historical significance, RFCs provide the current set of open standards used on the Internet, today. To both ensure interoperability with other products and prevent re-invention of wheels, developers of network applications will find it useful to be familiar with, and to the extent possible conform to, applicable standards. Examples of available standards are the Simple Network Monitoring Protocol (SNMP), Internet Protocol (IP), File Transfer Protocol (FTP), and Simple Message Transfer Protocol (SMTP), all of which are supported by SGI. Thus, if you were to write a network management tool, since numerous network devices use SNMP, you could manage each one by reading about its interface to SNMP, which is likely documented in an RFC.

Protocol specifications explain both the theory of operation and the proper transmission and interpretation of information from one computer to at least one other. Usually they will include examples of how a protocol and its functionality will be used.

There are currently several types of RFCs:

- o Standards
 - o Experimental protocols
 - o Informational texts
 - o Historical documents
 - o Jokes

Standards are further delineated as being proposed, draft, or standard. They are also categorized as being required, recommended, not recommended, or historical. Standards go through a rigorous peer review cycle, requiring deployment and testing of independent implementations. The actual standardization process itself is an RFC.

Experimental protocols are those that have not gone through the rigorous review, and are not meant to be widely deployed.

Informational texts are used to discuss current events or problems relating to the Internet. An informational text may outline concerns over a particular practice or protocol, or it may simply be a note documenting some sort of operational experience with the Internet or an internet application (note the lower case 'i').

Historical documents are any of the above types of RFCs deprecated for one reason or another. For example, at the time of this writing, the current standard for the Internet Protocol is RFC-791. As time goes by it may be necessary to replace IP with a better version. Each RFC may have a notation at the beginning, listing it as either obsoleting an older document, or being obsoleted by a newer document.

It is traditional that on April 1 of every year a humorous and fictional RFC may be issued. For example, one year someone wrote an option to the Telnet protocol called the Subliminal Message Option. Poetry of various luminaries also can also be reviewed.

How do I locate an RFC that might be related to my work?

Each RFC is named rfcN.txt, where N is the number (starting with 3, going to 1539, currently). Some RFCs are not present. These include many of the earlier documents, which may have been simple E Mails between Network Working Group members. In addition a small number of RFCs have been

assigned numbers, but have not been released.

Several indices are provided. See the file rfc-index.txt. One can search in an editor based on the expanded name or author of a protocol. In addition, most RFCs contain bibliographical information.

RFCs are almost always ASCII text documents that can be read with an editor such as Jot or vi. Some RFCs are written in Postscript, and may be read with tools such as xpsview. See Q40 for where you can find these gems on the SGI network.

Q42: What is ISDN?
(Author is vmuir)

BRI ISDN (Integrated Systems Digital Network) is digital service on a pair of wire, providing two 'B' (bearer) channels and one 'D' (data) channel. Each channel provides 64Kbps bandwidth and the data channel provides 16Kbps bandwidth. Generally the 'B' channels are used for either voice or data traffic and the 'D' channel is used for call setup, signalling and other administrative activities. ISDN facilities are modeled after more familiar voice services, making data and voice connections as simple as dialing a number much like Switched 56K service used throughout the US. The advantages are cost over the more conventional dedicated leased line and flexibility. Like voice, the circuit(s) are established and used only for short periods of time, usage charges generally are less than the typical dedicated monthly rates. An additional advantage is the availability of a second channel w/o the added cost until its used. ISDN may also be provided as a BRI line which consists of 23 'B' channels and a 64Kbps 'D' channel.

Since the majority of telecommunications offices are linked with older Switched 56Kbps technology the actual 'B' channel bandwidth is 56Kbps. However, communications techniques such as inverse multiplexing allows aggregating the two 'B' channels into one 112Kbps channel. This is very attractive for telecommuting and other remote applications.

SGI has made significant progress in making available ISDN to home users with the ISDN pilot project sponsored by IS. A limited group of engineers and IS participants have proven the feasibility of providing cost-effective ISDN service for employees at home, allowing "corporate" like connectivity through 112Kbps connections on demand.

If you'd like more information see Q43, Q44 and Q45 or send e-mail to vmuir@corp.sgi.com

Q43: Who provides ISDN in the Santa Clara valley?
(Author is vmuir)

In the Santa Clara Valley, ISDN is available from Pacific Bell (PacBell). Its provided at a nominal charge in one of two forms: SDSIS or CENTREX-IS. The most common is SDSIS since its more accessible for residential users. The Centrex-IS service is convenient for calls or communication within a limited distance of a PacBell office and where most of the calls are "internal", that is, not beyond the same PacBell office. Calls outside the office are charged usage just like SDSIS.

Unfortunately not all PacBell offices have SDSIS. San Jose, Milpitas, San Francisco, and other remote (rural) areas don't yet have the necessary hardware/software to provide residential SDSIS. From experience the cities that have SDSIS are Palo Alto, Cupertino, Menlo

94/05/03
11:18:58

faq.sgi

Park, Mt View, Northern parts of San Jose, Fremont, Campbell, Redwood City and Sunnyvale. This list continues to grow. Its expected that by the end of '95 most cities in the Valley will have SDSIS capability.

To find out if you have ISDN availability contact Robert Rodriguez or Henry Sandigo (SGI PacBell account Representatives) @408-369-3347 and 3328. They'll need only your home phone to check the office. Be aware that many new residential areas are served by underground phone lines and if all lines are used into the house then either trenching or other added expense will need to be considered.

Q44: How to get ISDN at home?

(Author is vmuir)

To get SGI sponsored ISDN into your home, a few things are required. First is management approval. Below is a list of expected costs associated with ISDN installation:

PACBELL INSTALLATION = \$158 includes time and material
PACBELL Montly Charge = \$27.50 this is just for having ISDN
PACBELL Usage = \$50 - \$100/month, varies by usage.
PACBELL waived Charge = \$150 this charge is prorate over 2 years. If the line is disconnected a proportion will have to be paid.
ADDITIONAL Charge = Be aware that if additional lines are required to be added to the premise, trenching may be necessary where underground services are used.

All charges are billed back to the users department.

Once a ISDN-Signup form is completed, a order is placed with PacBell by the IS department to install ISDN. PacBell will do an availability check to see if the residential office has ISDN capability. Once this is determined a site survey is done to determine if additional lines are required. Once this is done, an appointment is made with the home owner to add the line. In the PacBell tradition, they only commit to AM or PM, no specific times are given. However they've been known to call an hour in advance to warn of their arrival.

To get an ISDN Signup form send e-mail to vmuir@corp.sgi.com or copy the form to your system using the following command:

```
rcp guest@wyldman.corp:/usr/people/guest/isdn/isdn_signup .  
rcp guest@wyldman.corp:/usr/people/guest/isdn/isdn_signup_multiuser .
```

This will deposit in your current directory the framemaker files of two isdn signup forms. Print the related form, complete it and send to MS 15L-730 attn. Vince Muir.

Q45: What equipment is required to use ISDN?

(Author is vmuir)

The ISDN connection into SGI goes through an ISDN-hub. This allows central point of control and access. The hub bought from Net Express supports Combinet ISDN/Ethernet bridges. It allows 'B' channel multiplexing (112Kbps) and requires authentication for security.

The security is done in band, transparent to the user. Combinets are provided by the Network Technology group in IS. The cost of the Combinets depends on the model and use...if its meant for a single user

then the CB150 model is appropriate for \$860 (charged back to the department with a MTR). If there are multiple hosts passing through the same Combinet then the CB200 is necessary and costs \$1995 (either paid with a CAR by the department or Asset xfer'd).

In addition to the Combinet a NTL is necessary, serving as a demarcation point for PacBell. Its a pair of units, one for power and one to synchronize with the ISDN network. It has data cord connections to the wall jack and back to the Combinet. This unit is required and costs \$210 (charged back to the department with a MTR).

Equipment not provided is the workstation with an ethernet network interface.

To get an illustration of a typical ISDN home connection, send e-mail to vmuir@corp.sgi.com or copy the showcase file to your system using the following command:

```
rcp guest@wyldman.corp:/usr/people/guest/isdn/isdn-dialup .
```

This will deposit in your current directory the showcase file of the ISDN dial-up network layout.

Q46: What about ISDN order forms and other documentation?

(Author is vmuir)

ISDN forms and other documentation can be copied using the following command:

```
rcp -r guest@wyldman.corp:/usr/people/guest/isdn .
```

This will create an isdn directory and deposit all the related files (some in Framemaker others in Showcase) for your use.

If you'd prefer hardcopies please send e-mail to vmuir@corp.sgi.com with either your fax # or mail-stop and you'll receive these documents within 5 days.

Q47: What is mosaic?

(Author is murphy)

Mosaic is a hyperlink document system that ties together its own flavor of documents, as well as FTP, gopher, WAIS and archie sites. Those last four nouns do not refer to a florist, a rodent, a misspelling of a place of great girth and a comic book character. No, they are both four very different methods of accessing and four different styles of repository of information on the net. Mosaic actually access more flavors of data, I was just not able to conveniently word play them. The others of which I know are: NNTP (the Internet news protocol), telnet, raw ascii, hynetnet, hyper-g, techinfo, texinfo and anything in the form of man pages.

Mosaic unifies and simplifies access through its GUI. A hyperlink in mosaic is words in blue or a picture framed in blue (or purple, if you have traversed it once). When you click, you are propelled into another document that could actually be located anywhere in the world. The nice thing is that you don't need to have any worry as to where the anywhere is. It just happens behind the scene.

Q48: Where is a good place to look for (and deposit) software local to

94/05/03
11:18:58

SGI?

There are two, marginally overlapping, techniques. You can browse likely candidates using the `whatswhere` command. This works only if the author/maintainer has added an entry to the `whatswhere` database (run `'whatswhere whatswhere'` for a pointer to the database and how to add/update entries). Running the command `'whatswhere'` will list commands and release names for which you can obtain more information. Feel free to use this mechanism to document your additions to the SGI bag of tricks. See Q55 to obtain the `whatswhere` command.

The other technique is to look in the directory `dist.wpd:/sgi/hacks` where there are a plethora of `inst'able` images of stuff. See the `README` file in that directory for a description of commands such as:

```
elm
ghost
ispell
jpeg
pbmplus
sc
stocks
trn
uemacs
xv
```

To learn about creating `inst'able` software run the command `'handbook ism'` or `'handbook sw4inst'`. See Q55 to obtain the `handbook` command.

Q49: How do I get my picture added to the face server?
(Author is murphy)

To get your picture added to the face server, `newmedia.esd:/usr/local/lib/faces`, you'll need to do these steps:

1. Stop by Tom Murphy's office in 90 (across from the bathroom) any time you catch him, Tues through Thursday. He'll take a couple of pictures of you with a digital camera.
2. He will process your pictures and deposit the best in `newmedia.esd:/usr/local/lib/faces/-your_email_name>` (You are encouraged to be the one to determine how to define the word `best`.)

An alternative is to arrange with Tom for him to travel to your workarea to take pictures of a batch of people (presumably SGI folk wishing to have their faces in the face server).

Another alternative is to have a photo of your own, use a scanner to produce an RGB image, scale it to be less than 100x100 and deposit it in `newmedia` yourself. (See Q117 for more information on scanners.)

See Q51, Q50 and Q54 for ways of putting your face to use. It makes sense to have your picture taken if you send mail to people on IRISs. They can then see your face when using the `picmail` program.

The face server is backed up automatically and faces are restored automatically as well. This is done to avoid the loss of pictures that periodically afflicts `newmedia`. Both are done via `cron` at the midnight hour. A byproduct of this is that you cannot just delete your face from the `faceserver`. You must also leave an empty file behind (unless you want to witness the file being automatically restored at midnight). The next day the empty file will be removed by the `archive` tool.

faq.sgi

Q50: How do I set up my system so that I can click on my face to log in?

To set up your machine so that you can click on icons of users to login, you'll need to do these things:

1. Make sure that `usr2.sw.vadmin` is installed on your system. See Q11 if you need information on where to get it from.
2. Make sure that you have `newmedia.esd:/usr/local/lib/faces` mounted on your system as `/usr/lib/faces`. Q75 tells you how to do this.
3. Turn on `pandora`:

```
% su
# /etc/chkconfig visuallogin on
# exit
```

`Pandora` will be run at the beginning of every session on the graphics console and you'll see the faces of the users who have home directories on your system.

Q51: How do I get `picmail`, the program that shows the faces of those who email me?
(Author is murphy)

`Picmail` is an IRIS based program that displays the faces of people who send you mail. It updates the screen as new mail comes in and as your mailer finishes with old mail. You can `inst` it by typing:

```
% su
# inst -f dist.wpd:/sgi/picmail
Inst> install picmail
Inst> go
Inst> exit
# exit
% rehash
% /usr/local/bin/picmail
```

This will get you a 3.2 based version. (You can contact Nelson Kolyard to rep an equivalently functioned 4.0.5 based version). You will also want to add the line `'/usr/local/bin/picmail'` to your `$HOME/.sgisession` file so `picmail` is restarted every time you reset your window manager. You will also need to mount the face server `newmedia.esd:/usr/local/lib/faces` as shown in Q75. See Q49 to learn how to get your face added to the face server.

Q52: Is there a tool to view the entire organization including people's faces?
(Author is lauram)

Yes and the tool is called `orgview`. It helps you find job and location information on SGI employees from just ONE place. In addition to showing employee job and location information, `orgview` also displays the photos of employees who have pictures in the `faces` database. It also displays `orgchart` information for the employee which can then be saved as a PostScript file.

94/05/03
11:18:58

faq.sgi

orgview lets you perform a global search on all fields, including: employee name, manager, division, title, e mail address, and extension. This data will be updated weekly so that orgview displays the most current employee and organizational information.

There will soon be a feature which will enable you to send updates to your personal information from within orgview. Some of the fields you will be able to update are: a personalized job title, a personal information field, e-mail address, extension, and mailstop.

Due to it being so close to quarter end, please do not access these tools over over the WAN (networks outside of Mountain View). A solution and instructions for WAN use will appear shortly.

To install orgview:

```
% su
# inst -a -f atlas:/usr/dist/net-services/orgview
# exit
% echo "make sure /usr/sbin is in your search path"
% rehash
```

The prototype for orgview, called emp, was developed by Kurt Akeley and is also available for installation and use. To install emp:

```
% su
# rcp guest@atlas.corp:/usr/dist/net-services/emp/emp /usr/local/bin
# rcp guest@atlas.corp:/usr/dist/net-services/emp/Emp
  /usr/lib/X11/app defaults
# exit
% echo "make sure /usr/local/bin is in your search path"
% rehash
```

Installation Notes:

Orgview and emp expect automount to be running. If you do not wish to have automount, please see the excerpt from the orgview man pages below. If you do not have automount running and wish to do so:

```
% su
# chkconfig automount on
# /usr/etc/automount 'cat /etc/config/automount/options'
```

If you are currently running old versions of orgview or emp, please reinstall in order to be able to access the most current employee data.

Check sgi.faq for information on how to place your photo in the faces database.

Some of the future enhancements to orgview:

- Display of employee photos from the Security database (another good reason to get one of the new ID badges!)
- Ability to edit a personal job title field and an employee information field
- Ability to select employees by navigating through the orgchart
- Adding International employee data
- Ability to capture TBH's
- Ability to search by one or more specific fields (versus alpha version which performs a global search)

Both emp's and orgview's source code and data files are available. The data file is an excellent source for those of you trying to build

distribution lists. The data file is located in:

```
/hosts/orgview/usr/local/lib/orgview/employees.orgview
```

Please refer to the orgview man pages for additional information about the orgview utility including: how to customize the application, source file locations, known bugs, resources, and future enhancements.

If you have any questions or suggestions, you can post to sgi.is.orgview. If you would like to report bugs about the alpha version of orgview, you can post to sgi.bugs.orgview.

Updates/enhancements to orgview will be posted to sgi.general, sgi.hr and sgi.is.orgview.

Q53: How can I see the face for someone?
(Author is murphy)

You can use the locate command (See Q71 to get locate) to find out the building a person is in, walk there, ask where she/he sits, introduce yourself (and thereby accomplish glimpsing the face).

An alternative is to use the following script for the whois command:

```
#!/bin/csh -f
if "$1" == "" then
    echo syntax: 'basename 00' login name
else
    if -f /usr/local/lib/faces/$1 then
        pushd /usr/local/lib/faces >& /dev/null
        ipaste $1 -o 500 500
        popd >& /dev/null
    else if -f /usr/lib/faces/$1 then
        pushd /usr/lib/faces >& /dev/null
        ipaste $1 -o 500 500
        popd >& /dev/null
    else
        echo "no picture of $1 in faces library"
    endif
endif
```

In the long run, the first technique may be preferable, but the second is useful as a temporary workaround.

If the individual does not appear in the faces database, then you can encourage them to do so by following the directions in Q49.

Q54: How do I get my picture on the banner page of my laser printer output?

Laser printers can be set up to print banner pages (on everyone's output, not just yours) that incorporate the pictures of people who have icons in the face server, newmedia.esd:/usr/local/lib/faces. There are three parts to getting the laser printer set up and your picture on your banner pages:

1. Make sure your picture is available in newmedia.esd:/usr/local/lib/faces for printing. See Q49 for how to do this.

94/05/03
11:18:58

2. Make sure that the system that your laser printer is actually connected to has the face server (newmedia.esd:/usr/local/lib/faces) mounted. See Q75 for how to do this.
3. Install the 'facebanner' program on the system that the printer is actually connected to:

```
% su
# rcp guest@newmedia.esd:/usr/local/bin/install.ui /usr/local/bin
# /usr/local/bin/install.ui facebanner
```

Q55: What are infotools and how do I load them?
(Author is murphy)

These are the infotools:

faq - This viewer displays textual answers to Frequently Asked Questions (FAQs). These may be FAQs generated internally as well as from Internet newgroup postings. The FAQ, sgi.faq provides information needed by software development engineers that can be conveyed in a paragraph or two.

handbook - This command is used to present chapters of online handbooks. Currently, there is only one handbook, The Engineer's Handbook, which is displayed by default by the handbook command. It differs from the faq command since it typically displays information of a more extensive nature, often involving graphics.

whatswhere - This command provides a brief description about the location of SGI software releases and other software products. Release engineering keeps the release information current as an automatic part of their build process.

whatsubsys - Given a file on your system that has been placed there via the inst command, whatsubsys will tell you the inst subsystem to which it belongs.

rsvp - This command allows you to learn of and register for internal SGI classes.

newdirorg - This command supports the automatic setting of environment variables as you cd around your file system. A typical use is have the environment variables ROOT, TOOLROOT, and WORKAREA set when you enter a ptools workarea (or one of its subdirectories.)

To get the info tools, just type:

```
% su
# inst -f dist.wpd:/sgi/infotools
# exit
% rehash
% echo "make sure /usr/local/bin is in your search path"
```

Q56: How do I get ghostview, the working PostScript previewer.

Ghostscript, combined with ghostview, allow you to preview PostScript documents in an X window, without the need for Display PostScript (DPS). This allows you to preview postscript on any device that supports X protocol (like X terminals, Suns, etc...) Once installed, you can try it out by running:

faq.sgi

```
/usr/bin/X11/ghostview /usr/lib/X11/ghost/1tiger.ps
```

inst'able images containing binaries for the latest release of ghostscript (2.6.1) and ghostview (1.4.1) (and man pages) are available by typing:

```
% su
# inst -f dist.wpd:/sgi/hacks/ghost
# exit
% rehash
% echo "make sure /usr/bin/X11 is in your search path"
```

Q57: How do I get a copy of Emacs?
(Author is jackr)

There are at least four versions of Emacs in use at SGI, Unipress Emacs, GNU Emacs v18, GNU Emacs v19, and MicroEmacs. SGI sells the Unipress version.

You can install Unipress Emacs using the standard SGI inst tool from the standard distribution machines (see Q11), for instance:

```
% su
# inst -f dist.wpd:/released/5.0/Emacs/emacs
inst> go
inst> exit
# exit
% rehash
```

You can install GNU Emacs v18 or v19 for Irix4 like this:

```
% su
# inst -f atlas.corp:/usr/dist/net-services/gnu
```

or from it's mirror site:

```
# inst -f dist.wpd:/sgi/IS-services/gnu
```

Select one base version from these:

epoch	X11 only, separate window for each file, based on
emacs v18	emacs v18
gnuemacs	emacs 18, in X11 or terminal window, one window for
	all files
nemacs	Like gnuemacs, but also supports Japanese, also
	based upon emacs 18
emacs19	emacs 19 release - multiple X11 windows, or "frames"
	per process, also terminal window mode

You can install GNU Emacs v19 for Irix5.1 or later like this:

```
% su
# inst -f atlas.corp:/usr/dist/net-services/Irix5
```

or from one of the mirror sites:

```
# inst -f dist.wpd:/sgi/IS-services/Irix5
# inst -f dist.wpd:/sgi/hacks
```

Note that all versions of GNU Emacs install into /usr/gnu, and will follow symlinks if made before installation. Symlinks to the binaries are installed into /usr/local/bin.

94/05/03
11:18:58

faq.sgi

You should probably read the faq "gnu emacs.sgi" before installing, as there are some options you'll want to understand.

You can install uemacs (MicroEmacs) v3.12 like this:

```
% su # inst -f dist.wpd:/sgi/hacks/uemacs
```

The manual for Unipress Emacs is part number M4-EMCS-3.3 and can be ordered through the Manual Ordering Process for Engineers (see Q21).

To get the manual for GNU Emacs, "The GNU Emacs Manual", send \$15 for one manual or \$60 for six manuals to

Free Software Foundation
675 Massachusetts Ave.
Cambridge, MA 02139
(617) 876-3296

Q58: What is Showcase and how do I get a copy?

Showcase is a software tool that provides a showcase environment for software on the IRIS. Currently, it provides basic drawing and presentation capabilities allowing users to incorporate text, 2D drawings, 3D models and raster images on pages or slides.

Showcase 3.1 is available as a standard part of the 5.1 release. You can install the latest version using the standard SGI inst tool from the standard distribution machines (see Q11), for instance:

```
% su  
# inst -f dist.wpd:/test/5.1a070/nontoto/showcase  
inst> go  
inst> exit  
# exit  
% rehash
```

An extensive online help system is installed with the software. The 3.0 manual, M4-SHOWUTIL-2.0, can be ordered through the manual ordering process for engineers (see Q21). Email bug reports to sgi.bugs.showcase.

Q59: How do I get a copy of my favorite shell?
(Author is pdc)

The Bourne shell (sh) and the C shell have been part of the standard release since time immemorial or thereabouts.

Ksh is an advanced UNIX shell designed by David Korn at Bell Laboratories. We have version ksh88e. To install ksh on your workstation and get a formatted manual page, do nothing. It is now shipped as a standard part of 4.0 and is automatically installed as part of `eoel.sw.unix`.

Tcsh is like the C shell except with superior, programmable file name completion, command line editing (vi and emacs flavors available), and fewer bugs. To install it:

5.0 or later:
Do nothing. It is a standard part of 5.x.

4.x:

```
% su  
# rcp guest@lunch.asd:/usr/local/bin/install.pdc /usr/tmp  
# /usr/tmp/install.pdc tcsh  
# exit  
% rehash
```

Q60: How do I get Perl, that program that's supposed to replace awk, sed and sh?
(Author is scotth)

If you're running 5.1 or later, install `eoel.sw.perl`.

Otherwise, execute these commands:

```
% su  
# inst -f atlas.corp:/usr/dist/net_services/perl  
# exit  
% rehash
```

If you also want man pages for perl, `a2p` (awk to perl translator), `s2p` (sed to perl translator), and `h2ph`, then do the following line in `inst` (prior to starting the install).

```
inst> install perl.man.perl
```

Q61: How do I use rfind, that speedy new replacement for the find command?
(Author is pj)

"rfind" is a fast, client-server adaption of the `find(1)` command, with easier to use syntax and dramatic speed improvements. You can use rfind to search our main source trees (cypress and such) for a particular file in just seconds.

If you are using IRIX 4.*, you may inst it (as root) with:

```
% su  
# inst -f sam.wpd:/usr/rfind/rfind  
Inst> go  
Inst> exit
```

This will install, by default, `rfind.man.rfind` and `rfind.sw.client`.

If you are using IRIX 5.*, rfind is a default command in the basic `eoel.sw.unix` subsystem.

This version is compatible with all versions of the rfind client command. It is used by pools to speed up inquiries on the server source machine.

Known aliases in `</usr/lib/rfind.aliases>` (with rfind servers currently running) are:

```
root      localhost:/  
usr       localhost:/usr  
aspen     bonnie.wpd:/aspen  
aspen     bonnie.wpd:/d/d2/aspen-emplis  
att       babylon.wpd:/usr/ref
```


94/05/03
11:18:58

```
bonsai      bonnie.wpd:/bonsai
bugs        rains.wpd:/dl
clonestar  orange.wpd:/clonestar
cmplrs     bonnie.wpd:/dl/cmplrs.src
cteam      koko.wpd:/Cteam
cypress    bonnie.wpd:/cypress
dirge      bonnie.wpd:/dirge
dogwood    bonnie.wpd:/dogwood
elwood     bonnie.wpd:/jake/elwood
irix5      babylon.wpd:/usr/work
jake       bonnie.wpd:/jake
lonestar   lone.wpd:/lonestar
osf        maddog.wpd:/osf
view       jeeves.wpd:/VIEW
parts      dist.wpd:/all_parts
test       dist.wpd:/test
```

Known aliases in NIS (YP) rfind.aliases are:

```
root        localhost:/
elwood     bonnie.wpd:/jake/elwood
dogwood    bonnie.wpd:/disks/dogwood
aspen      clyde.wpd:/aspen
cypress    bonnie.wpd:/cypress
motif      bambi.esd:/motif
dce        maddog.wpd:/usr/ref/osf/dce_1.0/snap4
net2       maddog.wpd:/usr/ref/bsd/net-2
nfs4.1     maddog.wpd:/usr/ref/sun/NFS4.1
reno       maddog.wpd:/usr/ref/bsd/reno
svr4       maddog.wpd:/usr/ref/att
svr4.1_C2  maddog.wpd:/usr/ref/att/svr4.1_C2
tahoe      maddog.wpd:/usr/ref/bsd/tahoe
mips       maddog.wpd:/usr/ref/mips
irix5.1    bonnie.wpd:/disks/bits/5.1isms
.
usr        localhost:/usr
jake       bonnie.wpd:/jake
cteam      midas.wpd:/repo/Cteam
lonestar   bonnie.wpd:/lonestar
bugs       midas.wpd:/repo/Cteam/bugs
4.3       maddog.wpd.sgi.com:/usr/ref/bsd/4.3
net1       maddog.wpd:/usr/ref/bsd/net-1
osf        maddog.wpd:/usr/ref/osf/osf1.1
att        maddog.wpd:/usr/ref/att
svr3.1     maddog.wpd:/usr/ref/att/svr3.1
svr4.1     maddog.wpd:/usr/ref/att/svr4.1
svr4.1_B2  maddog.wpd:/usr/ref/att/svr4.1_B2
ref        maddog.wpd:/usr/ref
gnu        bambi.esd:/gnu
parts      dist.wpd:/all_parts
```

You may also install the server side of rfind on your local machine, if you so choose, and use it to quickly (like in under 1 second) perform finds over your local system. Speed is addicting. See the fsdump(1M) manual entry for more details on the server side. Note that rfind only sees files that can be read by the world.

On IRIX 4.*, to install the server type:

```
% su
# inst -f sam.wpd:/usr/rfind/rfind
Inst> install rfind.sw.server
Inst> go
Inst> exit
```

faq.sgi

versions changed

If the versions command shows rfindd.N then merge it into /usr/spool/cron/crontabs/rfindd.

On IRIX 5.*, you can find the server side in eoe2.sw.rfind.

See the man page fsdump(1M) for how to administer the server side.

Q62: How do I get that program that describes the hardware characteristics of my machine?

You need the program describe. It describes your machines hardware characteristics in easy to understand SGI lingo. To get it, just type:

```
% su
# rep quest@mars.esd:/usr/local/bin/describe /usr/local/bin
# exit
% rehash
```

To get the latest version, you need only type:

```
% su
# /usr/local/bin/describe .
```

(*describe .*) might fail if you have an old version that tries to update from mycool.esd. Just reinstall via rep and all will be well, at least with your version of describe.)

Q63: Will "Lower Bloat Vision" (tm) really help me visualize creeping bloat? (Author is roqerc)

Yes.

In fact you can Visualize (tm):

- Physical Memory Breakdown
- Resident Size of Processes
- Total Size of Processes

To get bloatview just type:

```
% su
# inst -f dist.wpd:/released/5.2/noship/noship
# Inst - keep *
# Inst - install noship.sw.bloatview
# Inst - go
# Inst - exit
# exit
% rehash
% /usr/local/bin/bloatview
```

It has a man page that explains what it does, and it also has on-line help. Please send roqerc@sgi.com mail if you have questions, comments, or problems.

Nota Bene: Willso's memusage program offers a similar view of the world.

94/05/03
11:18:58

Nota Multa Bene: This particular tool is of course for internal consumption only. In other words, don't put it on sgi.com for anonymous ftp!

Q64: Is there a vacation program to automatically respond to email that I am gone?
(Author is murphy)

Yup. And the example section of vacation command man page is very helpful in setting things up. It will help you set up three things:

vacation message - This is the email message that will be sent to people to let them know you are away. Make sure you keep the "Precedence: bulk" bulk line. This will help tools like pv to discard such automatic messages and not enshrine them within the bug database or clutter other such places.

.forward file - This is the file that actually turns on the vacation program. It suffices to just rename this file to something like ".forward.vacation" when you do not want your vacation program operating.

ndm vacation database - This is used to track to whom the forward message was sent and to ensure that they don't get the message more often than the time you select. By default, it is once every seven days.

Q65: How do I get a bingo gamecard that I saw used in a high level meeting?
(Author is davis)

The executable and source are bedlam.asd:-davis/junk/(bingo,binqo.c).

This alpha release impacts your opportunity to synergize any proactive meeting you guesstimate you are attending. Of course, you might incent the key player's hot button as you blow them away (and the meeting.) The author refuses to indemnify such a critical path, leading edge 90% solution to standardizing treatment of meeting boredom. However, the author is open to one on one off line discussions to work the issues of revolutionary proactive code improvements.

Q66: How do I get webster (and xwebster), the online dictionary/spell-checker?
(Author is karlton)

webster and xwebster give you access to an on-line dictionary on the main SGI campus. You can inst it by

```
% su
# swmgr -f /hosts/dist.wpd/sgi/hacks/webster Hit the "Go" button.
Exit swmgr when it gives you the chance.
# exit
% rehash
% webster
```

(Note that you can also use inst and a variant of these instructions to accomplish the installation as well) You may also want to add the following command to your ~/.login file which will ensure webster looks

faq.sgi

to the correct place for its database:

```
setenv WEBSTER fudge.asd.sgi.com
```

With the merging of all the engineering domains into engr.sgi.com, the logical location of the data base may change. If the aliasing mechanisms of named don't work or aren't used, this will necessitate a change to the above command.

Q67: How do I get that program enhance to spiffify my rgb images?
(Author is paul)

enhance is an image processing program that makes it easy to explore many different transformations simply and easily. If you want to get the program type:

```
% su
# rcp guest@mackel.asd:/usr/sbin/enhance /usr/local/bin
# exit
% rehash
% enhance <in.rgb> <out.rgb>
```

Q68: How do I get a copy of VCal, the reasonable electronic daytime program?
(Author is mikey)

VCal will help you remember all those meetings that are legion at SGI. It supports repeating entries, alarms, etc. It will do most everything you will want.

If you are running Irix 5.2 MR or later, you can load it by typing:

```
% su
# rcp guest@jeeves.wpd:/usr/local/bin/install.mikey /usr/tmp
# /usr/tmp/install.mikey vcal
# exit
% rehash
```

If you are running Irix 4.* or an older version of Irix 5.*, you can load an older version of VCal by typing:

```
% su
# rcp guest@jeeves.wpd:/usr/local/bin/install.mikey /usr/tmp
# /usr/tmp/install.mikey vcal.old
# exit
% rehash
```

Use the "On Window" menu item under the "Help" menu to see the help text describing VCal and its interface. Send bugs, comments, and suggestions to mikey@sgi.com.

Q69: How do I get less which is more than more?

less can be used to replace the program more with the side effect that you get some extra features. You can scroll backwards as well as forwards and you can use vi style searches for keywords.

To get it, type:

94/05/03
11:18:58

faq.sgi

```
% su
# rcp guest@anchor.esd:/usr/local/install/install.anchor /usr/tmp
# /usr/tmp/install.anchor less
# exit
% rehash
```

To use it as your pager with man then add the following line to your .login file:

```
setenv PAGER 'col|less -ik'
```

Q70: How do I load Framemaker?

You can load the software by adding /usr/frame/bin to your search path and then typing:

```
% su
# inst f mom-media.corp:/usr/dist/apps/frame
# exit
% rehash
% maker
```

One fly in the ointment is that you either need a frame license or access to a machine with a frame license server. You can contact the help desk (3-help) to order the frame license. If you know a license server to use, then include the following line in your .login file to have your copy of maker know about it:

```
setenv FRAMEUSERSD_HOST <license server>
```

We (SGI) are no longer supporting Framemaker directly on campus. Frame is directly taking over support. Bugs and comments to comments@frame.com.

Q71: How do I get a copy of locate, the online phone directory program?

You can install a copy of locate with these commands:

```
% su
# rcp guest@odin.corp.sgi.com:/usr/local/bin/locate /usr/local/bin
# exit
% rehash
```

To use it, give the command 'locate <name>', where <name> is a login ID or any portion of a person's first or last name. Note that mosttemps and contractors are not in the phone book. See Q72 for information on changing the information that locate displays.

Q72: Why isn't my login ID in the locate database?

There are two possible reasons why you don't see your login ID when you look yourself up with locate. The first is that the nightly build of the database may have failed for your domain. In that case, nobody in your domain will have a login ID, but the next nightly build should fix the problem.

The second reason why your login ID might not appear is that your name in the YP database does not exactly match your name in the SGI phone book database (aside from case). Your name in the SGI phone book database is

what you see when you do 'locate <yourself>'. Your name in the YP database is what you see in the "Full Name" field when you do 'ypchpass'. One way to find your entry is to type 'locate <your_telephone_extension>', given of course you have a telephone extension. If you change your full name ypchpass to match what's in the phone book, then you'll see your login ID with locate. While you're using ypchpass, you might want to update all of the information listed so that the finger command will display your telephone extension, etc. If it is your locate entry that contains invalid information, then you can contact telcom through the help desk at X3 4357 and give them the right stuff. (Automaton Imaguard will tell you that pressing 1 will get you telephone or voicemail help. Press 1 it gets you telcom.) See Q71 to get the locate program. See Q73 to learn how to change the remaining fields in the locate database.

Q73: How do I change the information about me in the locate database?

See Q72 for information about changing your email address and name. Email mail stop changes to mailstop changes@msmail.corp. The average mortal cannot change the department name. If a wrong department number appears for you, then call the help desk (3-4357) to get it fixed. To change the name become a manager (hopefully of that department) and call in the name change to the help desk (which will result in changes to two other arcane related databases).

Q74: How do I decide which machine to access, given several choices?

Typically it will be the machine network closest to you. The command 'ping -c 1 -R hostname' will reveal how many hops there are between yourself and another host. For instance it shows 2 hops from quixote.wpd to mars.esd(relay.esd).

```
PING mars.esd.sgi.com (192.26.58.1): 56 data bytes
64 bytes from 192.26.58.1: icmp_seq=0 ttl=254 time=30 ms
RR:   gate-whizzer.esd.sgi.com (192.26.58.97)
      relay.esd.sgi.com (192.26.58.1)
      whizzer.wpd.sgi.com (192.26.61.24)
      quixote.wpd.sgi.com (192.26.61.8)
```

The speed of the connection on a machine is a consideration as well. For instance, mars.esd has a slow onp board while atlas.corp has a fast ot connection to the backbone.

SGI software releases are located on several systems. The exact machines and locations are fairly stable, but can change as new releases are added, the systems with software releases run out of disk space, etc. See Q11 to learn where to get software on the distribution machines. So, with that said, the three machines that store software releases are:

```
dist.wpd      CSG, bldg 9
stress.asd    ASD, R&D, bldg 7
mars.esd      ESD, bldg 2
atlas.corp    CORE, all other bldgs
release.csd   CSD, bldg 12
```

Q75: What directories should be NFS mounted on my system? (Author is dana)

94/05/03
11:18:58

The things you probably want NFS mounted on your system are:

the SGI face server place for 100x100 rgb pictures of SGI
 employees used by pandora, picmail and on
 laser printer banner pages

Sample entries in your /etc/fstab for mounting these things are given below. The angle brackets, <>, indicate portions of path names that can be customized for your machine. The names inside the brackets are good choices. Note that you can use the automounter to mount these directories instead (see Q76).

```
# SGI face server
newmedia.esd:/usr/local/lib/faces /usr/lib/faces nfs ro,by,intr,hard 0 0
```

You'll need to edit your /etc/fstab, create each of the directories listed as mount points (second column) and mount them. A sample command sequence is:

```
% su
# vi /etc/fstab            (add above entries with your mount points)
# mkdir /usr/lib/faces
# mount -a
# exit
```

To use the public \$ROOTs to build software on your workstation, you'll need to set \$ROOT and \$TOOLROOT. You'll need to set \$WORKAREA to point to your p_tools workarea (see Q79). See Q88 for some info on managing the setting of your environment variables.

There is a twist to the ROOT and TOOLROOT information mentioned above. We are currently in a phase of development where we are running from a soon-to-be-released OS rather than a not-ready-for-prime-time-soon-to-be-stable-alpha. At such a time it is more appropriate to point ROOT and TOOLROOT to /. The intent is to always use the most current libraries, header files and tools that are available.

Q76: What directories should I automount on my system and how do I do it?
(Author is dana)

Exported directories on other people's systems that you occasionally want to use are good candidates for automounting. Automounting radically reduces the overhead of using many NFS file systems, it keeps filesystems you are not using from cluttering up your system, and when you access an automounted directory that is not available, automount gives up. (If the same directory were NFS-mounted, your command might hang.) The disadvantage of automounting is that there will be some delay in establishing the mount if it is not already in place.

There are a variety of ways to configure automount. Simply activating automount via chkconfig, as shown below, will allow you to access any filesystem exported by <hostname> via the path /hosts/<hostname>. Three files are needed to set up the more elaborate example shown here. A more elaborate example is provided by the three files given below. In this example, the SCR and alphamore bug tracking tools from the machine rains are automounted, and the -hosts option is used. The -hosts option enables you to reach almost any machine's exported filesystems by mentioning a pathname to that machine. The -hosts option provides a very general mechanism for easily accessing files that have been exported by others. It will mount all exported file systems from the hosts you are accessing which can cause significant delay. See Q77 for information on

faq.sgi

exporting file systems (accessed by the -hosts option). If there are files you regularly access, it may make sense to establish your own automount map which automounts within just the file system or use an NFS mount in /etc/fstab.

In the example, names of files are surrounded by **** and their contents are after their names. Angle brackets, <>, indicate portions of pathnames that can be customized for your machine. The names inside the brackets are good choices. Don't include any blank lines in the files you create.

```
**** /etc/config/automount.options ****

vi /etc/auto.master

**** /etc/auto.master ****

/<net>                    /etc/auto.indirect            -ro, hard
/<hosts>                    hosts

**** /etc/auto.indirect ****

# bug tracking tools and bug database
<bugtools>                rains.wpd:/dl/bin
<bugs>                    rains.wpd:/dl/Bugs
```

Give these commands to start automount (note that you will be rebooting your system):

```
% su
# chkconfig automount on
# mkdir /<net>
# mkdir /<hosts>
# mkdir /usr/lib/faces
# reboot

% su
# ln -s /<net>/<bugtools>/* /usr/local/bin/
# ln -s /<net>/<faces>/* /usr/lib/faces
```

Note that automount won't mount directories until you access them. So, if you give the command 'ls /<net>', you won't see any files or directories listed, other than those previously mounted. If you give the command 'ls /<net>/<bugs>', you'll get ls output for rains:/dl/Bugs. Similarly 'ls /<hosts>' will show just previously mounted filesystems, but 'ls /<hosts>/bacall' will show you the exported directories in bacall:/.

If you get a message about stale NFS handles for an automounted directory then you can either wait till automount naturally disposes of the directory or you can type:

```
% su
# umount -h <hostname>
# killall -HUP automount
```

If you ever need to restart automount (which should never really come up) you can type:

```
% su
# killall TERM automount
# /usr/etc/automount `cat /etc/config/automount.options`
```

94/05/03
11:18:58

faq.sgi

Q77: How do I let others automount (or mount) my files?
(Author is dana)

The `exportfs` and `exports` man pages outline the mechanism to allow access from your file systems to others via NFS. The file `/etc/exports` contains the file systems to export and their export characteristics. Typically, we have two file systems `/` and `/usr`. You may have an additional drive that is another file system. For instance, `/etc/exports` might contain the following to give read only access to file system `/`, `/usr` and `/d2`:

```
/      -ro
/usr   -ro,nohide
/d2    -ro,nohide
```

You can use the command `"exportfs"` with no parameters to see your current exports (if any). `Nohide` is an SGI extension which makes the contents of this filesystem available without explicitly mounting it as long as its parent is mounted.

Q78: How do I set up a software development environment on my workstation?

There are four steps to setting up your workstation for software development. You should complete each step before proceeding to the next.

- Step 1. Your workstation must be connected to the network.
- Step 2. You must be able to send and receive mail on your workstation.
- Step 3. Certain programs must be installed and certain directories should be NFS mounted on your workstation.
- Step 4. You must create a software development "workarea" and build environment.

See your system administrator (and possibly others such as your manager or your administrative assistant) for help with Step 1. See Q28 for instructions on accomplishing Step 2.

Some of the software that must be installed on your machine (Step 3) is available from the distribution directories listed in Q11. The subsystems that you need in addition to the basic software on the system are:

<code>dev.sw.cc</code>	C compiler
<code>noship.misc.gthead</code>	Internal Development Header Structure
<code>dwb.sw.dwb</code>	<code>troff</code> & fonts
<code>emacs.sw.editors</code>	<code>emacs</code>
<code>noship.misc.swtools</code>	some local software development tools
<code>noship.misc.kernels</code>	master.d file for better net performance
<code>trans.sw.trans</code>	<code>psroff</code>
<code>dps_eoe.sw.dps</code>	<code>xpsview</code>

If you need to package software for `inst` then you will need:

<code>noship.misc.distgen</code>	Software Distribution Generation Tools
<code>noship.misc.idbtools</code>	Software Distribution Generation Tools

In addition, you need a news reader (see Q12) and `ptools` (see Q79). The rest of the programs you need are in the NFS directories you should mount on your system. Q75 lists the directories that you should NFS mount and tells you how to do it.

To create a software development "workarea" and set up your build environment (Step 4), you'll need to set some environment variables and run `p_setup`. See the `ptools` man page for how to do this.

You will also need to have an account on the source machine. To get an account on `bonnie` (the source machine for trees such as `jake` and `cypress`) you need to enlist the help of someone who already has an account. You will need to have a valid YP account (See your division system administrator to set that up). You `compatriot` with the account will need to type:

```
% rlogin bonnie
bonnie % nacct <your_user_name>
bonnie % exit
```

Q79: How do I get a copy of `ptools`, and how do I find out about new releases?
(Author is mini)

`Ptools` is a collection of source code management tools. They are the tools that everyone uses to check out source code from the SGI source trees and check it back in. To use `ptools` on your system, the recommended method to get access to them is to `inst` them (see Q11). To get `ptools` just type:

```
% su
# inst -f dist.wpd:/sgi/ptools/new
Inst> install ptools
Inst> go
Inst> exit
```

To get just the manual pages for `ptools` just type:

```
% su
# inst -f dist.wpd:/sgi/ptools/new
Inst> install ptools.man.ptools
Inst> go
Inst> exit
```

When a new set of tools is released, messages are sent to `sgi.bugs.ptools` and `sgi.engr.all`. If you have the tools NFS mounted you don't need to do anything, except possibly `inst` new man pages. New users should do `'man ptools'` to get started on using them and run the command `p.classnotes` which provides additional information about `ptools`. `ptools` require that RCS be installed on both the source machine and your machine. Use the command `'handbook ptools'` to get information on installing RCS. See Q55 to get the handbook command.

Q80: How do I get the document on SGI Makefile conventions?
(Author is murphy)

You can copy it to your system to print out or view (with `xpsview`):

```
% rcp dist.wpd:/sgi/doc/swdev/makeconv.ps /tmp
```

It is a Postscript document.

94/05/03
11:18:58

faq.sgi

Q81: What bug tracking system does SGI use?
(Authors are digit, murphy, and fota)

SGI uses two complimentary bug tracking systems that can be treated as a single bug tracking system. The oldest one is electronic mailer and newreader based. Bugs are submitted and modified via a mailer and viewed and modified via a newsreader. The advantage of this technique is that it is convenient to scan all bugs via the newreader. The downside is that bug submittal does not automatically gather much pertinent information. In particular, there is no inefficient way to assign a bug to the appropriate engineer. See the man page for pvintro for guidelines on using a mailer to submit bugs (remember this is not the recommended way to submit bugs.)

The newer system is based on Sybase. There is a GUI, pv, and a command line oriented report generator, pvquery. The advantage of this technique is that all sgi bug newsgroups are in the one database and easily searched. It is easy to submit bugs with much data already collected for you on your behalf. There is even a keyword analysis of your bug possible to aid you in assigning the bug to an appropriate group.

Lengths are gone to keep the two views of the bugs system consistent. You can count on this. Questions on the bug system will be gladly fielded by Joseph Fota.

Q82: What are the newsgroups to which I should post bugs?
(Author is olson)

If you send mail to bugs@mars.esd you will be mailed a current (as of last week) list of bug newsgroups.

The news groups and their purposes are:

News Group	Purpose
sgi.bugs.aspen	bug reports and changes for aspen
sgi.bugs.aspen.3p	bug reports and changes for 3rd party aspen products
sgi.bugs.aspen.gfx	graphics & UI bug reports and changes for aspen
sgi.bugs.aspen.irix	UNIX bug reports and changes for aspen
sgi.bugs.bug_tracking	the bug tracking system
sgi.bugs.calypso	bug reports for Secure Unix
sgi.bugs.coltrane	bug reports about XTP
sgi.bugs.cypress	bug reports and changes for cypress (4.0 based releases)
sgi.bugs.cypress.3p	bug reports and changes for 3rd party cypress products
sgi.bugs.cypress.gfx	graphics & UI bug reports and changes for cypress
sgi.bugs.cypress.irix	UNIX bug reports and changes for cypress
sgi.bugs.dogwood	the next release after Cypress
sgi.bugs.dso	Dynamic Shared Object ELF bugs
sgi.bugs.explorer	bug reports about the son of conman
sgi.bugs.express	new ESD graphics
sgi.bugs.frame	bug reports for Framemaker
sgi.bugs.gl5	GL5.0 bug reports
sgi.bugs.gl5.tests	GL5.0 bug reports (test cases?)
sgi.bugs.gldebug	GL Debugger
sgi.bugs.imagevision	bug reports about Imagevision
sgi.bugs.iv	bugs in IRISVision products
sgi.bugs.lonestar	bugs in the IP17 release

sgi.bugs.netls	bug reports about the Network License server
sgi.bugs.netvis	NetVisualizer discussions and bug reports
sgi.bugs.networker	bug reports about the Legato backup system
sgi.bugs.pca	bug reports about Power C
sgi.bugs.perf	compiler related performance issues
sgi.bugs.pfa	bug reports about Power Fortran
sgi.bugs.ptools	P.Tools
sgi.bugs.scenario	Scenario 1.0 bugs, suggestions, and requests
sgi.bugs.showcase	bugs related to Showcase!
sgi.bugs.swexpress	bug reports about "swexpress"
sgi.bugs.video	ASD video problems
sgi.bugs.volmgr	Iris Volume Manager
sgi.engr.case.bugs	VIEW project
sgi.src5000	jake only changes

Q83: How do I install bug tracking software on my workstation?
(Authors are digit, murphy, and fota)

Well, you need only type:

```
% su
# inst /usr/dist/wpd:/sgi/pv/new
Inst> install *
Inst> go
Inst> exit
# exit
% echo "make sure /usr/local/bin is in your search path."
```

Q84: How do I get color schemes to work with pv?
(Authors are ib, and mini)

You must let the schemes directory know about pv by typing:

```
% su
# /usr/lib/EV/X11/Schemes/achScheme
# exit
```

Make sure your .Xdefaults file does not have any lines of the form:

```
*scheme: foobar
```

You can comment them out (prefix with a !) or remove them.

Q85: Where can I find documentation on the SGI Bug system?
(Authors are digit, murphy, and fota)

There is a man page for pv and pvquery which outline these tools. pv_resources includes documentation of the large number of resources available to customize pv and pvquery. The man page for pvintro gives an overview and the subject line keywords that are possible if interacting with the bug system via a mailer. (See Q83 for info on how to obtain the man pages (and the software))

You can also read the Bug System class notes found in the Engineer's Handbook. Type 'handbook bugsys' to view them. See Q55 to obtain the handbook command.

94/05/03
11:18:58

faq.sgi

Q86: How do I find out about known bugs?
(Authors are digit, murphy, and fota)

There are three techniques for viewing the text of reported bugs. Some of these techniques can be used to view summaries of bugs as well.

- A) You can look at the bugs and other messages submitted to a bug group, `sgi.bugs.*` (or `sgi.engr.case`). (See Q82 for a list).
- B) You can use the command, `'pvquery'`, to do simple viewing and printing of bugs. See Q83 to learn how to install it. (See Q85 for information on bug system manual pages.)
- C) You can use the GUI, `pv`, to query for bugs and to view them. (same installation and documentation source as above.)

Q87: How do I remove a source file from the source tree?

There is currently no `ptools` command to do this (one will shortly be added though. Note that `p_purge` removes a file ONLY from your local workarea). If you need to obsolete a source file you need to login to the source machine(s) where the source tree lives and use good old `'rm'` to blow it and its corresponding RCS file away. For example:

```
% rlogin jake (Note: You don't need to login as anyone other than
yourself if you're in nuucp or engr (alternate names
for group 10). If you're not in group 10, you
shouldn't be doing this!)
jake% cd /jake/att/usr/src/cmd/cruft
jake% /bin/rm -f garbage.c
```

You have now removed the "working file" `garbage.c`, but `./RCS/garbage.c,v` is still around and that's really the file that `ptools` cares about. At this point, you must do one of two things; either remove the RCS file or "archive" it.

To remove the RCS file, simply use `/bin/rm`:

```
% cd RCS
% rm -f garbage.c,v
```

To "archive" the RCS file, `cd` to the RCS directory, create a directory called "old" and move the RCS file into it.

```
% cd RCS
% mkdir old
% mv garbage.c,v old/garbage.c,v
```

By moving the RCS file into the "RCS/old" subdirectory, you are hiding it from `ptools`; `p.update` and `p.link` won't see it so it looks like it's been removed from the source tree.

Whether you remove the RCS file or "archive" it is entirely up to you and is usually determined by your confidence that no one will ever need this file again or if they do, that you can find the correct backup tape. Under most circumstances, files are not archived and are simply removed from the source tree altogether.

Q88: How do I manage all the environment variables I have to set?

(Author is murphy)

If you only have one workarea, then you might want to set them in your `.cshrc` (or its moral equivalent for your shell of preference.) Another alternative is to create shell aliases to set them for you. Yet another possibility is to use the `newdirorg` command. (I favor the third, probably because I wrote it to suit myself.)

`newdirorg` helps enable the setting of the environment variables as a side effect of `cd'ing` around your system (or `pushd'ing` or `pop'ing`, if you are so inclined). It takes just a bit of extra work to set this up (but then you are set for life (as far as ease of environment variable setting goes)). Namely, You need to replace the `cd` (and `pushd` and `popd`) with a shell alias/function that invokes a specially named script. The `newdirorg` man page guides you in performing this task. `newdirorg` does the part of generating the shell scripts for `.csh`, `.ksh` and `.tsh`.

Q89: What is an ISM?

(Author is murphy)

ISM stands for Independent Software Module. It is a work in progress at SGI. Perhaps we should call it `SM`, it'd just be a lot harder to pronounce. The intent of ISMs is to move to uncouple our products from each other so that development, build and test can happen more smoothly.

"What does being an ISM mean to you?" might be a better question. It is really pretty simple. You will need to specify what files you produce (and some characteristics of these files.) You will need to specify the product structure you want (as seen by a customer trying to inst your stuff). You need some things from other developers (other ISMs) in order to do your build, e.g. most likely you will need a compiler, various Unix commands and some common SGI include files. The things you need from other ISMs are broken into two groups. You will get header files and libraries from a `ROOT` directory and build time tools from a `TOOLROOT` directory. Both of these directories are quite relative to a specific release (and sometimes even quite relative to a particular hunk of iron, I mean new hardware platform.) Your ISM may produce some of these components for other ISMs. There is a mechanism to help you provide these things. Note: these header files and libraries of `ROOT` document the interface between ISMs (which helps justify the `I` in the `ISM.I`).

Q90: If I got an ISM, how do I produce it?

(Author is murphy)

Lets assume you have the environment variables `ROOT` and `TOOLROOT` set (Q93 for more info) and `WORKAREA` set (Q79 for more info). All you have to do is run the command "makeism". If all goes well, you will have a set of installable inst images in the directory, `images`. (See Q98 for testing suggestions.)

If all does not go well, you might want to look at Q95.

By default, `makeism` will execute these steps:

```
setup:
    currently a no op
update:
    - Remove obsolete files from the workarea (i.e. ones no longer
      in the source tree) via a "p_check | p_purge y 1"
    - Get latest version of readonly files via a "p.update -v"
```

94/05/03
11:18:58

faq.sgi

headers:

- Generate header files and libraries to \$ROOT via a "make headers exports"

rawidb:

- Set the RAWIDB envariable to build/IDB and execute install target via a "make rawidb"

ism:

- Generate the files that may be needed by the build group via a "make ism"

images:

- Generate inst'able images via a "make images"

version:

- Document info that will help recreate the images via a "make version"

Other states are:

purge:

- Start from scratch via a 'make clobber'
- List candidates for the LINT target (i.e. extra non source files that remain in the workarea) via a "p.check w"

exports:

- alias for headers, i.e always produces both targets

addrawidb:

- RAWIDB=\$IDBFILE make install

By default, it will create a log directory named \$WORKAREA/logs.<today's_date>@<current_time>.<day_of_the_week>. In it you will find log files for each phase of the generation of your ISM. Most some of the phases have three versions, a full version of the log, a strained version to extract warnings (*.warnings) and the tersest version strained for just errors (*.strain).

You can tailor makeism to match the characteristics of your ISM. For instance, suppose you are creating a new ISM that has not yet been added to the source tree. Thus you do not yet want the ptools commands to be run (steps setup and update). (You do still need to define the WORKAREA environment variable.) Suppose also that you produce header files for other ISMs but you use them from files internal to your ISM rather than from ROOT (which you NFS mount as a read only public root.) Thus you do not want the headers step taken. Suppose none of your ISMs subsystems are mangled into a part of eoe1 or eoe2 (which means you don't need *ism phase). Your execution of makeism becomes:

```
makeism *rawidb *ism *images
```

See the man page for makeism for more information.

Q91: What do I need to get started as an ISM?

(Author is murphy)

You will need to either install or mount a current ROOT and TOOLROOT. (See Q93 to see how to do this.)

You will also need to honor some targets in your main Makefile and the Makefile in the build directory just off your main directory. (See Q92 for a discussion of them.) Luckily for you, it turns out there are a buncha common Makefile include files in heavy use at SGI that help out a lot. These combined with an ISM directory structure template that is available to you means that all you have to do is a bit of customizing.

Run the command 'handbook makeconv' for a document on SGI Makefile conventions. See Q55 to get the handbook command.

Run the command 'handbook commoninclude' to see the notes for the common SGI Makefile include files class.

See Q94 for information on getting the ISM directory template and doing the customization.

You will need to maintain your idb and spec files. These define the content and structure of your product. (See Q94 for a discussion of the idb and spec files. You can also run the command 'handbook sw4inst' for a detailed discussion of idb, spec and other issues concerning the generation of instable images.)

It is best to do a nightly build, i.e. generate instable images, to ensure that all aspects of your ism remain current, i.e your ism is compilable and has a correct product structure. Some groups even stitch in regression testing to ensure the product continues to function the same. (See Q95 for some of the issues concerning the ongoing care and feeding of your ISM.)

Q92: What are the ISM targets I need to support?

(Author is murphy)

The notes to the Common SGI Makefile Include File Class cover these targets in more detail. Even more detail will be found in the files themselves (see the directory \$(ROOT)/usr/include/make). (Run the command 'handbook commoninclude' to obtain the class notes. Run the command 'handbook makeconv' for more details on SGI Makefile conventions in general. See Q55 to obtain the handbook command.)

I will briefly cover the targets needed in the main ISM Makefile, the targets needed in the build directory Makefile and the targets expected in a source directory Makefile. In general, a target will operate on the files in the directory in which it is invoked, if any, and all pass the target to a reinvoication of make in some or all of its subdirectories.

Please remember that the only parts you really need to specify is how to make a particular product and the contents of the install target (a block of INSTALL command lines.) (Running 'handbook commoninclude' define the targets you get directly from a common include file. See Q94 for the targets you can get by using a standard ism template directory.) The targets shown below will be tagged in parenthesis with where you would look to get the definition of the template. These targets should, in general, not depend on each other. For instance, a headers should not first do a clean or a clobber. It should just install the header files. An exception is clobber which depends on clean.

The targets you need in the main Makefile are:

clobber (commonrules) completely clear the directory (and subdirectories) of everything but source

clean (commonrules) remove all the byproducts of making your product. The .o files are a canonical example of such byproducts.

startversion (ism:commonrules) write as 10 digit version number to \$(WORKAREA)/.version_number. (The 5 digit alpha number is also available in \$(WORKAREA)/.alpha_number.) See the man page for the command mkversionnum which does the actual calculation.

94/05/03
11:18:58

faq.sgi

You will want to run startversion any time you judge you need a new version number.

headers (template) - this target installs to ROOT the header files that are needed by at least one ISM other than your own in order for it to build. It is primarily used by the build group to build to bootstrap a ROOT to use for the remainder of the build. If you produce no headers for other ISMs, you provide a do nothing target. You typically pass the headers target to the source directories containing the header files and then use the INSTALL command to perform the installation. (make sure the RAWIDB envvar is not set.) You do this setting the macro HEADERS_SUBDIRS to the list of subdirectories to visit (and you also uncomment the command found with the headers target.)

exports (template) - this target installs to ROOT the libraries that are needed by at least one ISM other than your own in order for it to build. It is primarily used by the build group to build to bootstrap a ROOT to use for the remainder of the build. If you produce no libraries for other ISMs, you provide a do nothing target. You typically pass the exports target to the source directories containing the header files and then use the INSTALL command to perform the installation. (make sure the RAWIDB envvar is not set.) You do this setting the macro EXPORTS_SUBDIRS to the list of subdirectories to visit (and you also uncomment the command found with the headers target.)

The headers target for all ISMs is run first, followed by the exports targets for all ISMs. A most current of ROOTS has thus been generated. Note that the ROOT thus produced should correspond exactly to the root product defined in the spec file and produced by the images target.

rawidb (template) - this generates your product, typically by setting the RAWIDB environment variable to build/IDB and passing the install target to your source directories.

images (template) - this target causes instable images of your ism to be generated in the images directory. This target should be run on a nightly basis to ensure that your ISM remains buildable.

ism (template) - this target causes an external idb and spec file to be generated in the idbs directory.

The targets you need in the build directory Makefile are:

clobber (commonrules) - completely clear the directory (and subdirectories) of everything but source. In the case of the build directory, this means the Makefile, the idb file and the spec file.

clean (commonrules) - remove all the byproducts of making your product. The .o files are a canonical example of such byproducts. In the case of the build directory, the clean target does the same thing as the clobber target.

finalidb (ismcommonrules) - this target assumes that a rawidb has been generated in the build directory (IDB). This target merges the rawidb file with the checked in idb file (idb) to form the finalidb file (finalidb). The rawidb file contains the information provided by the INSTALL lines in the source directory Makefiles. The checked in idb file provides the idb tag information (corresponding to that used in the spec file).

buildimages (ismcommonrules) - this is the target called by the images target of the main Makefile.

The targets you need in the main Makefile are:

clobber (commonrules) - completely clear the directory (and subdirectories) of everything but source

clean (commonrules) - remove all the byproducts of making your product. The .o files are a canonical example of such byproducts.

default (developer) - Produce the product. For instance, if you used this in the source directory for the command is. It would compile and link the command is. This should also be the default target when you invoke make with no specified targets.

install (developer) - this depends on the default target (to guarantee that the product is made.) It then executes the install command to install the product in the desired place on the workstation. The SGI install command has an interesting and useful side effect. When you set the RAWIDB environment variable to point to a file, then no files are actually installed and instead characteristics of each file is written to the file pointed to by the RAWIDB environment variable, one line per file in a quite ascii manner. This file is one of the files used to generate instable images. (first is the SGI software installation command.)

rmtargets (commonrules) - remove just the product. This is typically done when it is needed to relink your product without having to regenerate all the intermediate files, such as .o's.

Q93: How do I obtain the latest ROOT and TOOLROOTS?
(Author is murphy)

First you need to decide if you want to obtain a writable ROOT/TOOLROOT that you inst or a read-only ROOT/TOOLROOT that you mount.

To install the latest TOOLROOT:

```
% mkdir -p home/toolroot
% export TOOLROOT=/home/toolroot
% cd $TOOLROOT
```

Now use step 1a or 1b (but not both). They are basically equivalent. However 1a puts a file TOOLROOT_HISTORY in \$TOOLROOT to tell you what you have installed.

1a)

```
% rcp guest@babylon:/usr/dist/sherwood/bin/mktoolroot .
% ./mktoolroot /hosts/babylon/usr/dist/sherwood/LATEST
```

1b)

```
% rcp guest@babylon:/usr/dist/sherwood/bin/inst .
% set src=/hosts/babylon/usr/dist/sherwood/LATEST/toolroot
% ./inst -x -f $src -r $TOOLROOT
```

To install the latest ROOT:

94/05/03
11:18:58

faq.sgi

```
% mkdir -home/root
% export ROOT=~home/root
% cd $ROOT
% set src=/hosts/babylon/usr/dist/sherwood/LATEST/root
% $TOOLROOT/inst -x -f $src -r $ROOT
```

To mount the latest TOOLROOT:

```
% mkdir -home/toolroot
% export TOOLROOT=~home/toolroot
% set src=/hosts/babylon/usr/dist/sherwood/LATEST/ptoolroot
% mount $src -home/toolroot
```

To mount the latest ROOT:

```
% mkdir -home/root
% export ROOT=~home/root
% set src=/hosts/babylon/usr/dist/sherwood/LATEST/proot
% mount $src -home/root
```

Q94: How do I use the ISM template directory?
(Author is murphy)

Install the ismtools

```
% su
# inst -f dist.wpd:/sgi/ismtools
Inst> install *
Inst> go
Inst> exit
# exit
```

Create the template structure. Decide on a short name for your ism. For instance, if you were working with the Diagnostics ISM, you might choose diag as your short name. Also decide where you want the template structure to be. If you are going to modify the files to become your ism then you probably want it in your ptools workarea, since you will be copying it into the source tree. If you will be copying files from the template into the workarea then decide on that place.

```
% ism_setup -w <location_for_template> -i <short_ism_name>
```

In the main Makefile you will want to change the SUBDIRS macro to reflect the source directories that should have make run in them with the same target being passed to it. You will want to have the HEADERS_SUBDIRS list the subdirectories to produce header files needed at build time by other ISMs. Each will need to accept the headers target and either pass it to its subdirectory and/or install the appropriate header files to ROOT. Similarly, if your ISM produces libraries for use by other ISMs you will need to set the EXPORTS_SUBDIRS macro with the list of contributing subroutines and each will need to accept the exports target. Remember to uncomment the command you see associated with the definition of the targets headers and exports so that each appropriate subdirectory will be cd'd to and have make run in it. There are no other changes you should need to make in this Makefile.

In the build directory, you will need to have an idb file enumerating, one per line, each file that you produce. An easy way to do this is to set up your source directories, with the install targets as described in the Makefile conventions document. (Run the command 'handbook makeconv' to view it online. See Q55 to get the handbook command.) Now run the rawidb target. This will produce the file IDB in your build directory.

Remove all but the first six blank delimited fields. You will need to add idb tags at the end of each line. These are just short upper case names that the spec file is going to refer to in order to identify which files of the idb go where in the inst image that will be created. For instance, some very common tags are MAN to refer to man pages, EOE to refer to files needed by end users, DEV to refer to files needed by developers, TOOLS to refer to files this ISM is contributing to the TOOLROOT for the use of other ISMs. You may have more than one idb tag on a line.

The file spec.proto gives you a very basic spec file that should be of use. Note the block structuring of the product, image, and subsystem sections. The product names <ism_name> root, <ism_name> toolroot and <ism_name> noship have predefined meaning. The <ism_name> root product should correspond exactly to the files produced by the headers and exports targets. The <ism_name> toolroot product contains the contributions from this ISM to the TOOLROOT. The <ism_name> noship product contains those files that are for internal use at SGI only and must never be shipped to a customer. Run the command 'handbook sw4inst' for more detail on the options for the idb and spec files.

Q95: What things do I need to do to maintain my ISM (nightly builds etc)?
(Author is murphy)

Actually the paradigm of the nightly build is the right way to look at it. If you can build a clean inst'able image each night. Then it is likely that the build group snaps of your ISM will go smoothly. Of course, it is nice if all the latest revision (top of trunk) of all the files of your ISM play well together when you use the ISM. The nightly build only helps with this if you've woven some automatic testing mechanism into it.

Here are the typical issues of a nightly build. Lets assume you will generate a writable ROOT and TOOLROOT on your build machine.

Run makeism (see Q90 for more info). Errors could happen at various phases. If in the exports+headers file, then you had trouble building libraries. This obviously stops the build till things compile and install well. If errors during the rawidb phase then you have troubles building your executables or the Makefile \$INSTALL lines.

If during the ism or images phase, then you have trouble in your product structure (as defined by your idb or spec file) or in your product structure (as defined by which files you choose to make part of your ism). You will have a message that you have missing files when a file is mentioned in your checked in idb file (build/idb), but does not appear in your rawidb file (build/IDB, generated from your Makefile install lines). This could mean you really did eliminate a file or it could mean the build broke and you stopped adding entries to the rawidb file. You could also have an extra file appear. This reflects new files that appear in the rawidb file but not in the checked in idb. One way of remedying the situation is to adjust the checked in idb file. Delete a line for the missing case. Add a new line for the extra case (make sure you also add the appropriate idb tags to the end of the added line). Another possibility is to sidestep the issue by saying that the checked in idb file is correct i.e. ignore the missing and extra files. You can do this by defining the appropriate macro in your localdefs file, namely:

```
LIDEJOINFILAGE c
```

Note the use of the localdefs file. This is one of the few valid uses of it. Namely, to do something at your build time that does not happen at

94/05/03
11:18:58

the build group build time. You can do the equivalent thing on a one shot basis via the commands:

```
% cd build
% make VIDRJOINFLAGS=-c finalidb
```

Please note that this sidestepping should only be a temporary measure prior to your checking in your idb file. The build group does not use these options and you need to make sure an error free finalidb can be produced.

See the man page for the idbjoin command for info (this is the workhorse performing the merge of the rawidb and the checked in idb files to produce the finalidb file.)

Q96: How do I generate version numbers for my ISM?
(Author is murphy)

Just run the command "make startversion".

Oh, you want more details? Run the target whenever you want to begin preparing a new instable image that you want to distinguish from the last one you shipped internally or externally. It is good to get into the habit of generating it before you compile anything since it is now possible to embed the version number within an executable (extractable via the ident command.)

How is this version number remembered? Well, it is remembered in the file \$WORKAREA/.version_number. The format of the version number is defined in the man page for the mkversionnum command. The six digit alpha number portion of the version number is kept in \$WORKAREA/.alpha_number. By the way, \$WORKAREA was used rather than \$ROOT since \$WORKAREA is always writable, which cannot be said for \$ROOT.

You can use the command showversionnum to decode the version number into a human readable form. It also works on more ancient forms of the version number as well. You can generate a particular alpha number, corresponding to a particular date by using the mkversionhours command.

There are a number of macros you can set in your makefile that will affect the generation of the version number. Typically, the default action is exactly what you want. Typically, you will at most want to set BUILDER. The rest are really just for the use of the build group.

BUILDER - reflects who built the image (0 or unset if by engineer, 1 if by local build group, 2 if by the build group)

TREE_ID - reflects the OS of which the file is a part (0 if the main source tree, 1-9 reflects a specialized OS usually for a particular hardware platform). It defaults to the value defined in \$(ROOT)/usr/include/make/releasedefs.

PATCH_RELEASE - defines the first 8 digits of a patch release version number. Otherwise the first eight digits of the current version will be used.

PATCH_NUM - defines a specific patch number. (the last two digits of the patch number are 30 + PATCH_NUM)

VNUM_OVERRIDE - defines all ten digits to use for the version number. Some call this the Arsenal. Be careful, it's not time

faq.sgi

Note that you can accomplish much the same thing by just rewriting the files \$WORKAREA/.version_number and \$WORKAREA/.alpha_number. Do this for a one shot change that you really don't want to externalize via a Makefile change. Note the average mortal (developer or buildmeister) should never need to use this macro.

Q97: How do I construct an exitop for inst?
(Author is murphy)

Very sparingly. If you can accomplish the same or similar thing another way, then please do so for exitops (and preops and postops) do things unknown to inst and thus inst cannot undo them when the subsystem needs to be removed. End of commercial.

Here is an example of an exitop that sends mail when the target is installed. Note the layerings of escapes to deal with the globbing that happens at Makefile time and at gendist time. You also want to make sure that you expand envariables at Makefile time because it will probably not be around at inst time on the installation machine. For instance in this case, I want to record the version number that is found in the file \$WORKAREA/.version_number on the build machine. (See Q96 for more on version numbers.) The first line shown is a Makefile macro defining the exitop. The next two lines are an \$(INSTALL) line found in the install target.

```
EXITOP="exitop(A"echo XXX"XXX"[Mail -s XXX"VERSION:$(VERSION)XXX"  
murphy\]")"
```

```
VERSION:=cat $WORKAREA/.version_number; A  
$(INSTALL) -F $(DEST) $(EXITOP) $(TARGETS)
```

Read the chapter of the Engineer's Handbook obtained by running the command 'handbook sw4inst' for all the gory details. (See Q55 to get the handbook command).

Q98: How do I test an inst image that I've generated?
(Author is murphy)

First off, make a cross check that all files that appear in your checked in idb file get picked up by your spec file, even if only in a no-ship subsystem. In other words, make sure that at least one idb tag on a checked in idb line appears in an exp expression for some subsystem.

When you run "make images" make sure that no product, image or subsystem show up empty, i.e. add something to them or get rid of them.

Do an installation. Note that you do not need to install this to / on your workstation. You can install it someplace out of the way just for testing. For instance:

```
% cd images
% inst -x r foo f .
```

This will install your inst images rooted at directory foo as yourself (the -x means you do not need to be superuser). When you are done testing, just 'rm -rf foo'. You can run 'versions-inv r foo' to see that the right version numbers were picked up.

94/05/03
11:18:58

faq.sgi

Q99: How do I select which products to build?
(Author is murphy)

Most people live with the default behavior. This is to build all products defined in the spec file. Some people override this by setting the ALLIMAGES macro in the Makefile of the build directory. Note this can result in confusing behavior when someone adds a new product to the spec file which doesn't appear in the generated image (since it was not also added to the ALLIMAGES macro). If you want to build a subset of images on a one shot basis then you can generate images via a line something like:

```
make ALLIMAGES=<product_list> images
```

Q100: Who needs to know about my product release?
(Author is jlw)

Many groups need to know about any upcoming software release:

- Release Engineering
- Tech Pubs
- Documentation Services
- Manufacturing
- CSD
- Finance

Before your product can be released, each of these group must sign an Engineering Change Order (ECO) which signifies completion of their responsibilities.

Q101: Do I need to talk to to each group individually?
(Author is jlw)

An easy way to ensure they know of an upcoming release is by posting information about your product on the S/W Project Status Report. This report is sent to a mail alias which includes managers and some directors of the various departments who need to know about your upcoming release. The report is also posted to sgi.engr.all. To ensure your products prompt placement on the S/W Project Status Report contact jlw@wpd (x1828) or beth@wpd (x3732) with the following information:

- Product Name
- Projected MR Date
- OS Compatibility
- Business Team Mtg
- Project Manager
- Product Manager
- Technical Contact

It is also important to hold regular Business Team Meetings. By making sure that a rep from each group mentioned above is present, any specific questions they may have can be answered at the meeting. This will help your MR process run more smoothly.

Q102: When do I need to tell them this info?
(Author is jlw)

You need to begin communicating early in you product development cycle.

This way, your product will be tracked on the S/W Project Status Report and updated as changes are made. Thus, changes in schedule only need to be communicated to a few people and will be relayed via the report.

Q103: How do I get the Release Engineering group to build my product?
(Author is jlw)

Contact beth@wpd (x3732) to get a build engineer assigned to build your product. Providing the following information to the build engineer is also helpful:

- The name of the ism does your product belongs to.
- The size of your sources.
- The OS which your product will be based on.
- Any exports/hdrs.
- When / how often the builds should occur.
- Mail aliases/engineering contact for build problems.

Q104: When do I need to start an ECO?
(Author is jlw)

Contact Documentation Services immediately to begin structuring your ECO. They will need to know such information as:

- Reason for product MR
- Product structuring
- Marketing codes required
- Media label information
 - list prices
- Licensing information
- Product dependencies
- All hardware platforms supported
- Whether there are any superseded products
- Material disposition of last revision.

Q105: How can I run my own background program with IndigoMagic (perhaps a nightsky with shooting stars)?
(Author is joel)

In release 5.3, the desktop can coexist with arbitrary backgrounds, such as the twilight program. It does this through the use of the shaped window extension. This may cause a slight performance penalty. (Note that the penalty is not incurred if the simple tiled backgrounds are used). To eliminate the performance penalty, you might wish to disable icons on the background.

In 5.2 and earlier, any background not set with the background control panel is ignored if there are icons on the background. So what is a workstation configurer to do?

To disable icons on the background create the file
~/desktop-<yourHostname>/nodesktop. (in 5.1 it was just .desktop/nodesktop). This will disable the icons on the background, and allow you to run your own background.

In addition, for support of desks, the desktop will attempt to set a different background for each desk you have (including the initial one when you log in). How to disable that depends on what release you are

94/05/03
11:18:58

running. In 5.2 and later you can disable it by setting the resource 4Dwm*SG_useBackgrounds:FALSE in your .Xdefaults or Xresources file. In 5.1 you could issue the command killall bgdaemon in your .sgisession.

Note that in 5.3, some special backgrounds that use executable programs, such as twilight, are found in the background control panel. Setting such a background from the control panel will cause that background to be invoked every time you visit that desk. So it is possible to run a different background program in each desk. (There is no support yet for setting such backgrounds yourself, but we hope to get it there.)

Q106: How do I bypass the password prompt when switching to root?

Of course we are speaking of switching to root on your own workstation. Gaining root access for someone else's workstation is the subject for a question in a FAQ of a more dubious color.

If your user-id is 'ab' and you enter the command 'su' to switch to 'root,' you will typically get a prompt for the 'root' password. This can be very annoying on a private system on which user 'ab' often needs to switch to a superuser account. You will get the password prompt even when the passwords for 'ab' and 'root' are the same.

To eliminate this password prompt, enter the commands:

```
% su
# vi /rhosts
```

Enter the line:

```
localhost ab
```

in the file /rhosts.

Q107: How do I arrange for file name completion from the c shell?

Enter the line:

```
set filec
```

in the file .cshrc in your HOME directory.

Now when you want to enter a command that requires an entry from the current directory, such as a file name, enter the name of that command, followed by the first few characters of the file name you want to enter. Then press the Esc key. If there is no entry in the current directory that starts with the specified characters or if there is more than one entry that starts with those characters, you will hear a beep. The text you entered will either not be changed, or it will be partially completed. If there is only one entry in the current directory that starts with the specified characters, the entry name will be completed automatically.

For example, enter:

```
cd /etc
cd t
```

and press Esc. You should hear a beep, because there is more than one

faq.sgi

entry in the directory /etc that starts with t. Add the character 'r,' and press Esc:

```
cd tr
```

Since there is only one entry in the directory /etc that starts with 'tr,' the name 'transferDevice' will be automatically completed. You should see:

```
cd transferDevice
```

You can also press -Control D- to see all possible matches. File name completion works with both absolute and relative paths.

If you use file completion a lot, or are frustrated by the simplicity of csh file completion, give tcsh a try (see Q59). Its main claims to fame are interactive line editing and "kitchen sink" file completion.

Q108: How can I let someone be root on my system, without giving out my password?

You must first determine the full path of their machine. If you know the individual's email name you can find out the path to their machine (at least to the machine on which they receive mail) by typing:

```
% ypmatch <email name> aliases
```

or you can just ask them. Just add a line to the file /rhosts containing this path plus the word root.

For instance to give Vic Mitnick root permission to your machine, you could run the command:

```
ypmatch vic aliases to discover that Vic's machine is
wookie.wpd.sgi.com. You would then add the line:
wookie.wpd.sgi.com root to /rhosts on your machine for as long as
you wanted Vic to be able to logon to your machine as root.
```

Q109: When should I turn ypserver on (using the chkconfig command)? (Author is ian)

Only if your machine really is a ypserver. If it is not and you turn it on then Bad Things will happen. Machines which bind to yours will be unable to resolve hostnames, passwords or aliases. This makes people unhappy.

Q110: How do I set up a dial-back connection from work to home?

People who log in from home typically use a dial-back procedure for security reasons (and SGI picks up the bill for the call): they dial in from home, give an ID number and password, hang up, and then get a call from SGI which enables them to connect to their system at work.

To get your dial-back ID and password you have only to send a bit of email to dialback@sgi.com. Specifically, the email should include:

```
Person's name and extension
Modem brand and model
Modem speed to be used
```

94/05/03
11:18:58

Manager's or supervisor's name
Modem phone number
Voice phone number (if separate voice line at location of the modem)

To create a dialback account, the modem type and speed is needed to plan for facilities to cover people's changing needs. The manager or supervisor's name is needed to confirm that the request is coming from a valid person. And of course the home modem number is needed to program into the system for it to place the outgoing call. A voice number (if they are separate) is also needed in case they need to get in touch with the person by voice.

Questions about dial-back can be sent to the mail alias dialback@sgi.com or you can call the help desk at X4357.

Q111: How do I set use the dial back connection from an SGI machine?
(Author is murphy)

And you might think this would be easy!

This is what worked for me (with help from Jeff Z and Ian C and a fair bit of trial and error.) I am more than open to a more knowledgeable person to take over this question (and correct/stabilize my instructions).

- 0.1) Obtain a blessed modem. Blessed modems are those with a fix script in /etc/uucp. The current ones listed are dsi, hayes, intel, telebit, and zyxel.
- 0.2) Obtain a correct SGI-to-modem cable. For an Indigo, this is a X5-25to8 (Rev B) part.
- 0.3) Attach modem to SGI machine using cable. This example will assume a connection to port 1 of an Indigo.
- 0.4) Inst uucp software (eoe2.sw.uucp) from your favorite source of OS.
- 0.5) Unfortunately, kermit does not seem to appear in latest 5.X releases. This is the software I have been using so far, but I inst'd it from a 4.0.5 release. (You can always ftp it from Columbia.)

1) Modify /etc/inittab to define the tty to your system. In my case

```
t1:23:respawn:/sbin/getty ttyd1 co_9600 # alt console
```

with

```
t1:23:respawn:/usr/lib/uucp/uugetty -Nt60 -izyin,com ttyd1 dx 19200
```

The personal System Amin guide calls for a 'telinit q' at this point, but I was always conservative and just rebooted.

2) Define the modem to your System. To the file /etc/uucp/devices I added (after the line '# --Standard modem line')

```
ACU ttyd1 null 19200 212 x zyl496
```

3) Fix the modem. The first lines of the fix scripts contain comments that will help you run them. In my case I ran

faq.sgi

```
/etc/uucp/fix-zyxel -io -s 19200 1
```

4) I tried using cu to establish the connection, but it kept failing for me. (I hadn't yet tried using ttyd1 instead of ttyf1 instead of ttyml). I have successfully used kermit. My \$HOME/.kermit file contains the lines

```
set line /dev/ttyd1  
set speed 19200
```

5) Make the connection. I dial 415 960 1881. I respond with my 7 digit dialback code once the connection is made. I type the current password, 'playhard'. The connection breaks, the phone rings, the modem answers and I log in as normal.

Now that wasn't too hard? HA!

Q112: How do I get blank 1/4" tapes?

To get blank 1/4" cartridge tapes for backups or whatever, first fill out a Materials Transfer Request (MTR) form. Only a few lines need to be filled out (note that MTR forms are revised frequently so the form you found might look slightly different):

- * In the FROM - LOCATION block, on the STORES line, write 'MCS9'.
- * In the TO - INVENTORY EXPENSE OR FIXED ASSET block, fill out the EXPENSE / COST line with your department number (including company code) and account number '69501'. Get your manager's signature on the OTHER APPROVAL line.
- * In the middle section of the form, the PART NUMBER is '9486002', the DESCRIPTION is '1/4" cartridge tape', the U/M is 'EA' and you fill in the number of tapes you want under QTY REQ.
- * In the block at the lower left corner of the form fill in your division, your name, the date, your department and your extension.

Mail your form through interoffice mail to Jose Sanchez, M/S 4L 160. Your tapes should arrive via inter-office mail in a week or so (longer if it's the end of the quarter).

Q113: How do I get a blank Exabyte tape?

Two ways to get blank Exabyte tapes are to buy them at Fry's and to order them from Exabyte with a Purchase Requisition. The information you'll need for a Purchase Requisition is:

Exabyte Corporation
1685 38th St.
Boulder, CO 80301

8mm data cartridge 1/N 180181
\$22.75

You'll have to get someone to advise you on whether to submit your Purchase Requisition to your divisional purchasing people or to Corporate Purchasing.

94/05/03
11:18:58

faq.sgi

Q114: Can I electronically extract the serial number of my machine? On many systems you can type `/etc/nvram eaddr` and you will see something like

```
08:00:69:07:24:81
```

which will give you the serial number (| tr -s ':').

Another possibility is to type `/usr/etc/netstat -ia` and you will see something like

```
Name Mtu Network Address Ipkts ... other stuff
ec0 1500 b9U-ng wookie 35690193 ...
224.8.9.11
allhosts-mcast
08:00:69:07:24:81
lo0 8304 loopback localhost 266843 ...
allhosts-mcast
```

The last line of the address for the name 'ec0' is the serial number, sans colons.

This may not always work. I am still using what once was a serialless lab machine so `/usr/etc/netstat -ia` returns a value that was not used as my serial number. Also `eaddr` is not a non volatile ram variable on this system so `/etc/nvram eaddr` returns nothing. BTW, the value returned by these commands is the internal ethernet address of the machine.

Q115: What is the difference among the r4 processors and what does it mean to me?
(Author is wje)

R4000 R4400 R4600

Internal MHZ	100	150	100, 133, 150
(later)			
External MHZ	50	50	50, 44, 150
Primary I-Cache	8 KB	16 KB	16 KB
Primary D-Cache	8 KB	16 KB	16 KB
Primary organization	direct map	direct map	two set
Secondary cache	0 or 1 MB	0 or 1 MB	0 or 256 KB
(later)			
SPECint92	58	~ 88	61, 73
SPECfp92	60	~ 92	49, 58
Pipe stages	8	8	5

What does it mean to you? To answer this you must first know the sound of one chip propogating.

Q116: How can I learn more about X and Motif?
(Authors are karlton, and mars)

See the X related Frequently Asked Questions referenced in Q2.

See the X bibliography in `fudge.asd:karlton/archive/x.bib`.

The following sources are to be updated to compile under 5.X. In the meantime, they are available as reference. The sources can be found in

`directory clubted.asd:/usr/graphics/classes/motif/`

`xcode/xref` - source code found in O'Reilly Vol. 1 (The Xlib Programming Manual)

`xcode/xtrief` - source code found in O'Reilly Vol. 4 (The X Toolkit Intrinsic Programming Manual)

`xcode/xmotif` - source code found in O'Reilly Vol. 4/Motif edition (The X Toolkit Intrinsic Programming Manual/Motif edition)

`lib` - Mason Woo's Motif class libraries `src` - Mason Woo's Motif class source code

`src/mixed` - examples of merging X and GL

Q117: Is it possible to scan a picture into an SGI image file?

Yes.

There are scanners in various departments around campus that you can learn of via word of mouth. There is a pseudo public scanner being tested by Roger Chickering, the scanner driver guru, which you can use. It is a Ricoh FS2 attached to machine pod located outside of Ian Clements office in building 9 upper. For the techies out there (aren't we all), it does a blazing 600dpi 24 bit color scan. Use the command `scan` on pod to do the scanning. Please report any problems via email to `sgi.bugs.impressario`.

Q118: What do I do if I suspect I have Carpal Tunnel Syndrome?
(Author is douglas)

HR has been drafting a policy and process on how to respond to this medical problem. Something more formal will be announced. Here is what is known:

1. If you are experiencing pain in your wrists, STOP WORKING, apply ice, and keep your wrists as calm and straight as possible. ASAP visit the Peninsula Industrial Clinic (visit HR area for a MAP). WHY?, because IF you do get Carpal Tunnel Syndrome and you may need physical therapy and miss some work, you need to CERTIFY when the injury started and Peninsula Industrial Clinic does that for you no charge.
2. Call Sue Wathrich x3168 for more info about modifying your workspace, workman's comp benefits, etc.
3. Larry Rogers from Hartford insurance is another resource for workplace design and ergonomics specialist.
4. Modify your own workspace ASAP before pain. Get a wrist pad for the keyboard and the mouse pad from Frys. Position your body so your are looking straight ahead at your monitor, legs should be at 90 degree angle, and forearms parallel to the floor. DO NOT BEND YOUR WRISTS. If you use your mouse, there are considerations you need to know.

Q119: How can I link up with a carpool or some other commute alternative?

94/05/03
11:18:58

faq.sgi

SGI maintains a list of people who are interested in carpooling. Call the Commute Hotline 3-3748 where you will speak to Julie Krieger, our commute alternatives coordinator. To add yourself to the database and get a list of likely carpool partners (i.e. those coming from roughly the same area and working roughly the same hours.)

Another alternative is to use the RIDES program, which will attempt to link you with people likely from outside of SGI (since it is a state program). Call 800-755-POOL to add yourself to their database and get a matchlist.

Q120: What does the name IRIS mean?

IRIS stands for Integrated Raster Imaging System.

Q121: Who are SGI's Stock Brokers?

(Author is kubey)

Alex Brown

Polite, but many employees have had bad service (missed messages, failing to return phone calls, clerical mistakes, etc.)

Cowen & Co 415 676-2880

No major complaints

Morgan Stanley 415 576-2000

Not Polite, but no major complaints

Charles Schwab 408 730-0111

Prefers that you open a separate account for SGI ESPP shares
Has Telebroker service (buy, sell, get quotes using a touchtone phone)

1-800-4-1-PRICE

They advertise a flat \$30 fee for upto 5000 shares with no limit on the price.

Q122: Are there any notary publics at SGI?

(Author is murphy)

Yup. There are at least two that you can contact. Make sure you call first to arrange a time convenient to both of you. You can contact Margie Wandel or Devon Johns.

Engineer's Handbook General Stuff

May 5, 1994

Ptools Class notes

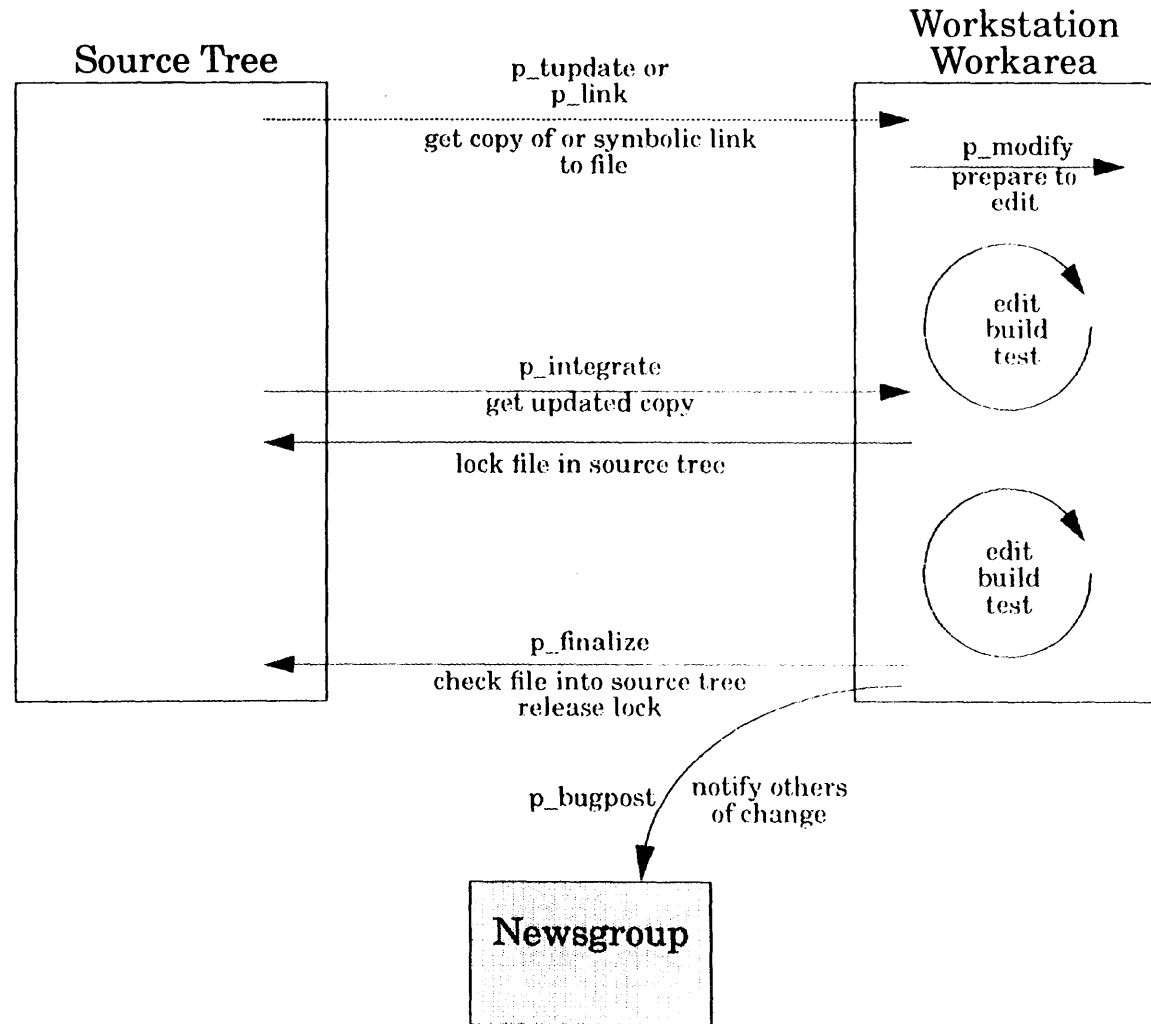
probable data location: `dist.wpd:/sgi/doc/swdev/ptools.ps`
access via: handbook ptools

ptools

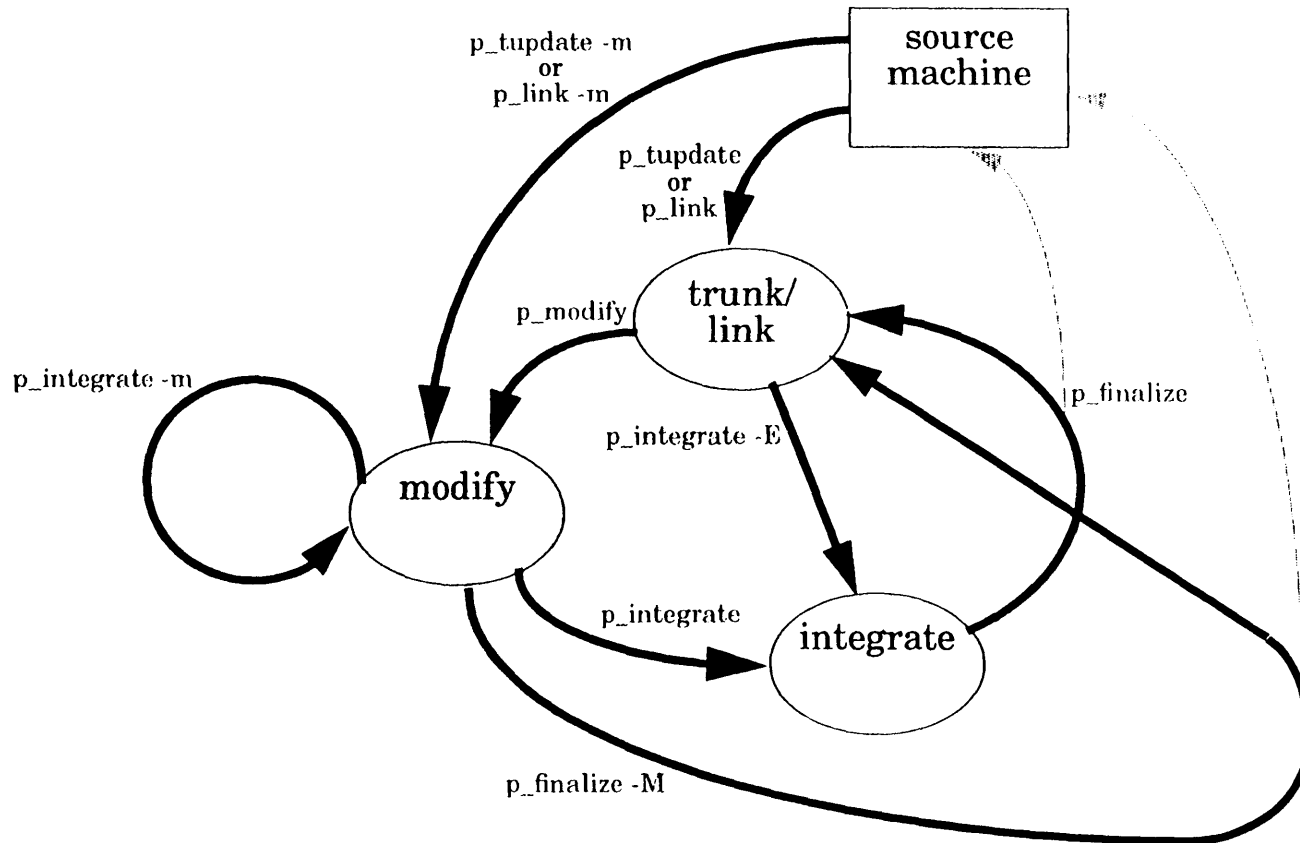
SGI Software Development Tools and Conventions

- Introduction
- Basic Development Model (page 2)
 - company-wide source tree
 - local workstation development
 - deferred locking
 - project based partially linked trees
- What is ptools?
 - based on ctools using RCS underpinning
 - state diagrams (page 3)
 - administrative commands (page 7)
 - overview of command options (page 8)
- Using ptools on your workstation
 - workarea structure (page 12)
 - census file structure (page 13)
 - How to set up a ptools workarea from scratch (page 14)
 - standard ptools actions (page 20)
 - configuration options (page 21)
- How to set up a source tree (page 27)

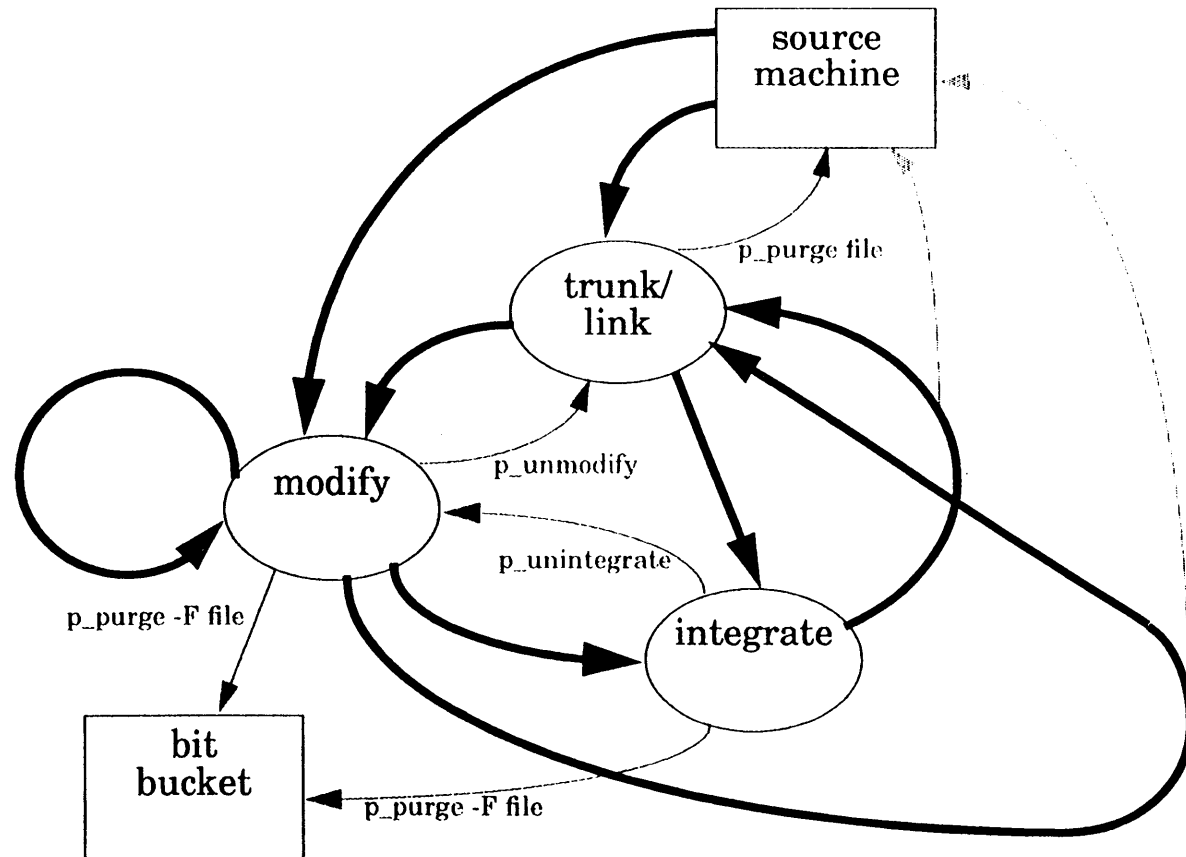
Basic Development Model



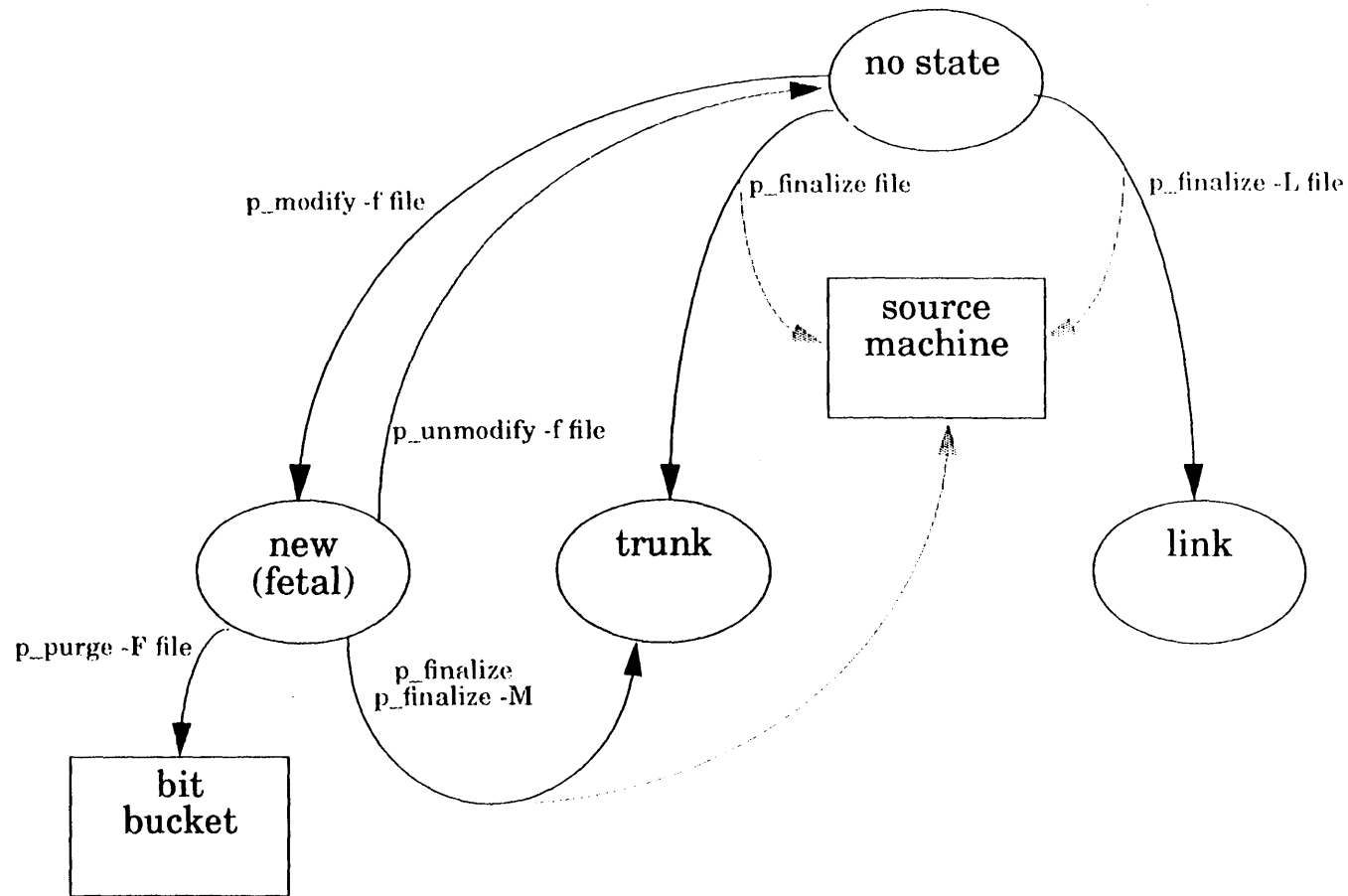
Basic ptools State Diagram



ptools State Diagram (undo operations)



ptools State Diagram (adding new files)



Basic ptools commands

p_tupdate	populate/update a workarea with copies of source tree files
p_link	populate a workarea with symbolic links to source tree files
p_modify	change workarea files so they can be edited
p_integrate	prepare modified file for check-in by merging with and locking the file in the source tree
p_finalize	check new, fetal, or integrated files into the source tree
p_purge	remove files from a workarea
p_unintegrate	change the state of a file from integrated to modified
p_unmodify	put workarea files back into trunk or link state

Administrative ptools commands

<code>p_setup</code>	Obtain ptools and establish ptools work environment
<code>p_state</code>	Report revision number and state of a file in workarea
<code>p_list</code>	List all files in a particular ptool state
<code>p_check</code>	Check that a workarea has valid files
<code>p_test</code>	Test validity of the workarea (similar to <code>p_check</code>)
<code>p_rdiff</code>	Compare differences between source file revisions
<code>p_rlog</code>	Display RCS log for a source file
<code>p_error</code>	Provide detailed ptools error and recovery information
<code>p_bugpost</code>	Post message to SGI bug tracking system

Basic ptools Command Options (1 of 2)

	i	l	n	q	v	x	R	V	dirs	files	empty	c	k	m	r	D	
	use stdin	list files	dry run	quiet	verbose	exclude files & dirs	turn off rfind	list config opts	do dir & all subdirs	do files	use cwd	repair work- area	RCS keyword expansion	move to modify	RCS rev #	rev date of source	Exceptions
p_tupdate	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	d - RCS difference threshold e - create no new files f - do rev history a la rlog g - use supplied census file o - fetch revision as non-ptools file E - create no new directories
p_link	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓			
p_modify	✓	✓	✓	✓	✓		✓	✓	✓	✓							f - put new file into fetal state
p_integrate	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	↳				m - merge file only, no integrate B - do not backup files E - create lock for trunk/link state file O - lock file, no merge, update census file
p_finalize	✓	✓	✓	✓	✓		✓	✓	✓	✓		✓	↳	✓	✓		b - bug # for email message m - checkin message for all files t - tree from which to tagalong u - refresh to trunk state after finalize y - no questions on tagalong, assume yes B - do not post bugs F - force tagalog L - make files become link state M - finalize from modify state O - disregard any RCS merge overlap S - suppress all questions T - finalize tagalongs only Y - suppress tagalong tree check

Basic ptools Command Options (2 of 2)

	i	l	n	q	v	x	R	V	dirs	files	empty	c	k	m	r	D	Exceptions
	use stdin	list files	dry run	quiet	verbose	exclude files & dirs	turn off rfind	list config opts	do dir & all subdirs	do files	use cwd	repair work- area	RCS keyword expansion	move to modify	RCS rev #	rev date of source	
p_purge	✓	✓	✓	✓	✓			✓	✓	✓							f - force purge on all files listed u - suppress usage message y - suppress confirmation message F - force purge on all census files listed
p_unintegrate	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓						
p_unmodify	✓	✓	✓	✓	✓		✓	✓	✓		✓						f - make fetal files regular files u - highest rev in returning to trunk state B - do not backup files

Administrative ptools Command Options (1 of 2)

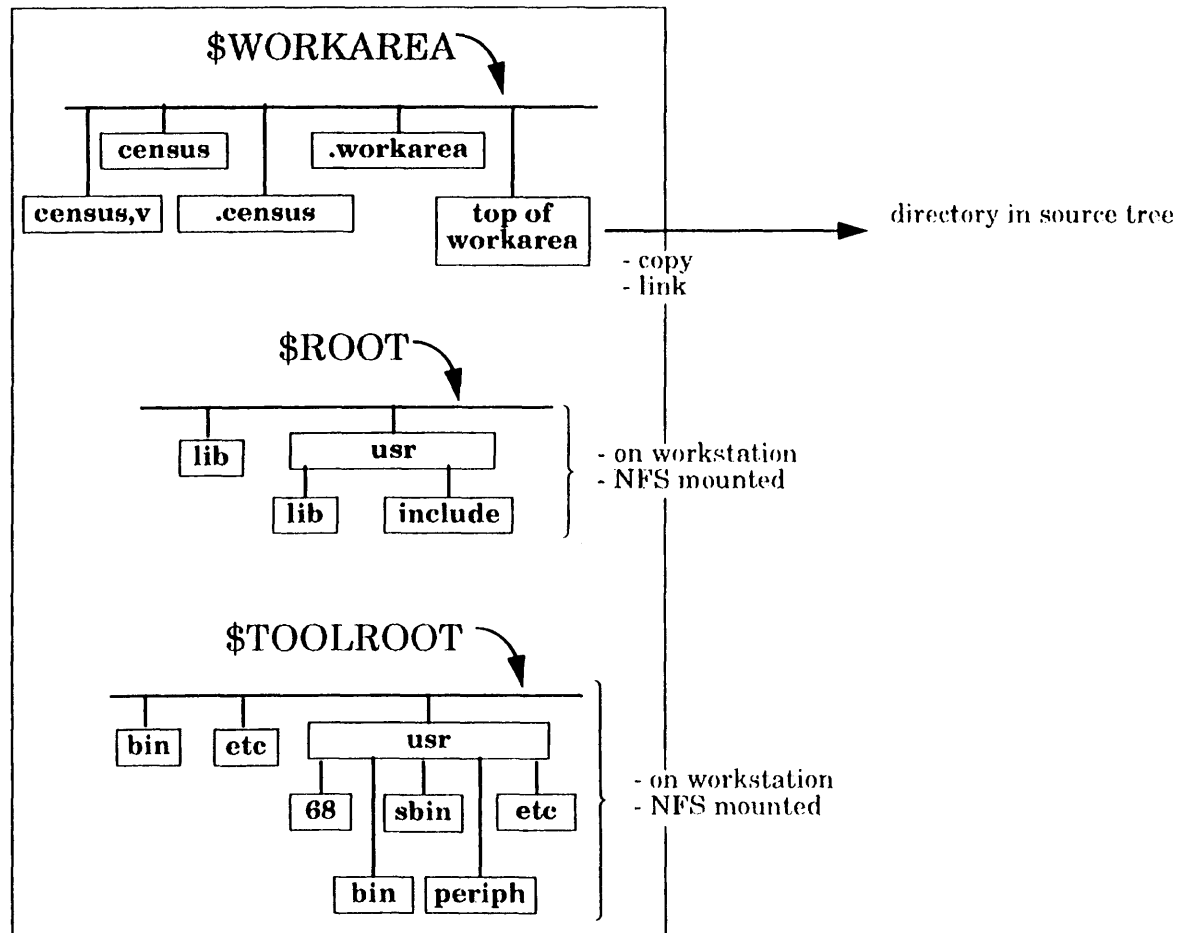
	i	l	n	q	v	x	R	V	dirs	files	empty	Exceptions
	use stdin	list files	dry run	quiet	verbose	exclude files & dirs	turn off rfind	list config opts	do dir & all subdirs	do files	use cwd	
p_setup			✓	✓	✓			✓				c - ask before modifying configuration files
p_state	✓				✓			✓	✓	✓	✓	
p_list	:>	:>						✓			✓	a - list all files f - list fetal state files i - list integrate state files l - list link state files m - list modify state files (default) t - list trunk state files A - operate on directory \$(WORKAREA)
p_check					✓		✓	✓	✓		✓	s - report files in census, but not in source tree w - report files in workarea, but not in census
p_test	✓	:>			✓		✓	✓	✓	✓	✓	c - query only census not source tree f - check census for corruption, interactively fix l - list files locked in source tree s - query only specified state u - report files in census, but not in workarea

Administrative ptools Command Options (2 of 2)

	i	l	n	q	v	x	R	V	dirs	files	empty	Exceptions
	use stdin	list files	dry run	quiet	verbose	exclude files & dirs	turn off rfind	list config opts	do dir & all subdirs	do files	use cwd	
p_rdiff	r>		r>		✓		✓	✓	✓	✓	✓	g - use gdiff rather than diff b, c, e, f, h, i, n, t, w - passed to diff k - RCS keyword expansion r - use file of rev # rather than top of trunk r (second)- diff files of two rev #s C - like c except with an argument
p_rlog					✓		✓	✓	✓	✓	✓	d, h, l, r, s, t, w, L, R - passed to rlog
p_error												<i>error_number</i>
p_bugpost												b - bug number c - people to receive carbon copies f - text file to use as the mail message g - people to receive the message m - subject line of the message t - message type S - suppress questions from being asked

Workarea Structure

Workstation



Census File Structure

- Format 1

```
FORMAT1SORTED720404094
```

```
/jake/buildtools/ismttools/Makefile%murphy%1.9|92.10.14.16.00.29(t)
```

```
/jake/buildtools/ismttools/idbdelete.sh%murphy%1.3|92.03.17.12.42.54(t)
```

- Format 2 (same file with census_db.output_format set to 2)

```
FORMAT2SORTED720482541
```

```
buildtools/ismttools/Makefile%murphy%1.9|92.10.14.16.00.29(t)
```

```
buildtools/ismttools/idbdelete.sh%murphy%1.3|92.03.17.12.42.54(t)
```

- First line defines census format (and date/time of last modification)

- Each subsequent line represents a file (with fields % separated)

- file name (absolute with format 1, relative with format 2)
- owner of file
- revision level of file in source tree
- revision date
- ptools state of file

How to set up a ptools workarea from scratch

1. Turn on automounter (page 15)
2. Install ptools (page 16)
3. Determine source tree machine and establish access to it (page 17)
4. Run p_setup (page 18)
5. Set appropriate environment variables (page 19)

How to set up a ptools workarea from scratch

Turn on automounter

- You need to do this only once per system
- Run the command **chkconfig**. It will report if the automounter is on.

```
% chkconfig
```

- Run these commands to turn on the automounter. See the man page for **automount** or the SGI Frequently Asked Questions for more information about **automount** issues.

```
% su  
# chkconfig automount on  
# mkdir /hosts  
# reboot
```

How to set up a ptools workarea from scratch

Install ptools

- You need to do this every time a new versions of ptools comes out
- Make sure `/usr/local/bin/ptools` is in your path
- Type these commands to install ptools.

```
% su
# inst -a -f dist.wpd:/sgi/ptools/new
# exit
% rehash
```

- Make sure you have RCS installed. The command “`versions eoe2.sw.rcs`” will tell you if you have RCS installed or not.

If you do not have it installed, consider installing from

`dist.wpd:/released/<release_num>/all/eoe2`,
where `<release_num>` is something like “4.0.5”. You can determine this value from the release number that appears at the end of the first line of the output from the command “`versions eoe2`”.

- If you don't have RCS installed and you execute a ptools command you may get the cryptic error message `Error # 7 EXECV_FAILED`.

How to set up a ptools workarea from scratch

Determine source tree machine & establish access to it

- You need to do this for each source tree you wish to access
- Run the command:

```
% /usr/bsd/rsh <source_machine> date
```
- This will verify your access to the source machine.
 - If just one line prints and it is the date then all is OK.
 - If you get “**Login incorrect**”, then you need to run the command:

```
/usr/bsd/rsh guest@<source_machine> nacct <login_name>
```
 - If you get “**Permission denied**”, then you need to adjust `<source_machine:$HOME/.rhosts` so a password is not required (see next bullet). Make sure group and other have no more than read permission. This was done for you, if you used the `nacct` command.
 - If you see more than one line of output, then you need to adjust `<source_machine:$HOME/.cshrc` so it does not generate any output. See the man page for `rsh` for more information. The default one created by the `nacct` command does not generate any output

How to set up a ptools workarea from scratch

Run p_setup

- You need to do this for each workarea you wish to create
- Run the command **p_setup**, (do this as yourself, not as root), and follow the steps below:
 - 1) Create a new ptools workarea "1"
 - 2) Modify source tree selection "2"
 - 3) Create a Sherwood 5.1 workarea "10"
(i.e. bonnie.wpd:/proj/irix5.1/isms)
(This step changes if you need to access a different tree.)
 - 4) Modify workarea selection "3"
 - 5) Name your workarea "~ /proj5.1root"
(see convention below)
 - 6) Create the workarea now "1"
using settings shown above
 - 7) quit p_setup "Q"
- A convention for naming your workarea is to take the leaf source tree directory name and append the word root to it.
- Make the NFS mountpoint use automount
modify the `workarea.sm_nfs_mount_point` in `$WORKAREA/.workarea`
to be of the form:
`workarea.sm_nfs_mount_point : /hosts/bonnie.wpd/proj`

How to set up a ptools workarea from scratch

Set appropriate environment variables

- You only once for each workarea you create. (if you use the `newdirorg` command, otherwise you need to do this every time you use ptools or use an SGI style makefile.)
- See Workarea Structure on page 12, for more information on the environment variables `WORKAREA`, `ROOT` and `TOOLROOT`
- `newdirorg` plus an alias for `cd`, `pushd` and `popd` will automatically set the environment variables you choose when you enter the workarea or one of its subdirectories.
- If you do not have the `newdirorg` command you can get it by typing:

```
% su
# inst -a -f dist.wpd:/sgi/infotools
# exit
% rehash
```

- See the man page for the `newdirorg` command for information on how to setup `newdirorg` for use with the kind of shell you use.
- Add these lines to your `$HOME/.dirorg` file to add a new workarea.

```
~/<path_to_workarea>
setenv WORKAREA
setenv ROOT </, other local, or mounted directory>
setenv TOOLROOT </, other local, or mounted directory>
```

Standard ptools Actions

- Populate your workarea with existing source files
 1. Suppose your workarea is rooted at `jake:/jake` and you want to work with the directory `jake:/jake/irix/cmd/yacc`
 2. Make sure your `WORKAREA` envariable is set
 3. `p_tupdate $WORKAREA/irix/cmd/yacc`
- See who has lock on your file
 1. `p_rlog file_name` (look at name associated with highest revision)
- See differences between your changes and latest source tree version
 1. Remain in modify state (i.e. rcs merge has not happened yet)
 2. `p_rdiff -g filename`
- Add an additional buggroup
 1. execute 'p_bugpost' to see number of current bug groups defined
 2. ensure 'p_bugpost.bug_groups : *next_number*' is in `~/.pdefaults`
 3. add 'p_bugpost.bug_groups.*next_number*: *name description*'
- For more info, see `ptools(1)` and `p_resources(1)`

ptools Configuration Files

- Workarea Configuration Hierarchy
 - Each level overrides the preceding level
 1. defaults for all workareas on your machine
 - /usr/local/bin/ptools/app_defaults
 2. defaults for all workareas in your account
 - ~/.pdefaults
 3. defaults local to each workarea
 - \$WORKAREA/workarea

ptools Configuration Options

- Sample configuration lines (to tagalong to jake automatically)

```
p_finalize.option.t.: on
p_finalize.option.t.value : hostname:/dir
```

- General command resources

Defaults

```
p_tool.option.option_letter : { on
                             off }                                off
```

```
p_tool.option.option_letter.value : value                          NONE
```

```
p_tool.message_handler. { output
                          error } . { tool_name_is_displayed
                                       message_type_is_displayed } : { on
                                                                                   off }
```

```
p_tool.message_handler. { output
                          error } .destination : { stdout
                                                    stderr
                                                    file
                                                    logfile
                                                    none }          { stdout
                                                                                   stderr
                                                                                   stderr }
```

Note: next line is needed if `file` or `logfile` is used in prior resource

```
p_tool.message_handler. { output
                          error } .file_name : /filename
                          warning }
```


ptools Configuration Options (continued)

● Tool specific resources

```

p_finalize.edits_first : { true }           off
                        { false }
p_finalize.disable_message_formatting: { true }       false
                                        { false }
p_bugpost.bug_groups : 1 .. 100           NONE
p_bugpost.bug_groups.number : emailname description  NONE
p_bugpost.authorcopy: /filename          $AUTHORCOPY

```

● Access to source machine (these are the resources written by p_setup)

```

workarea.sm_machine : machine           NONE
workarea.sm_location : /dir            NONE

workarea.sm_transport_method : { local
                                { nfs_rw }
                                { nfs_ro }
                                network }       nfs_ro

workarea.sm_nfs_mount_point : /dir          workarea.sm_location

```

OR

```

workarea.sm_tree_alias : machine:/path    NONE

```

ptools Configuration Options (continued)

● Access to tagalong trees

```
p_finalize.option.t : on
p_finalize.option.t.value : { machine:/dir,
                             alias
```

OR

```
workarea.secondary_trees : { machine:/dir } list NONE
                             alias
```

```
workarea.secondary_trees.tree_machine.alias : machine NONE
```

```
workarea.secondary_trees.tree_location.alias : /dir NONE
```

```
workarea.secondary_trees.tree_transport_method.alias : { local
                                                         nfs_rw } nfs_ro
                                                         nfs_ro
                                                         network
```

```
workarea.secondary_trees.tree_nfs_mount_point.alias: /dir
```

```
workarea.secondary_trees.tree_location.alias
```

ptools Configuration Options (continued)

- Location of workarea (using \$WORKAREA is preferred)

```
workarea.location : /dir                                $WORKAREA
```

- Workarea characteristics

```
workarea.editor : /dir                                $VISUAL else $EDITOR
workarea.backup_directory : /dir                    $PBACKD else /usr/tmp/pbackup
workarea.temp_directory : /dir                      $TMPDIR else /usr/tmp
```

- Source machine characteristics

```
source_machine.time_out_period : seconds (0 to ~68 years)    60
source_machine.ignore_rfind_changes : { true                false
                                         false }
```

- Arcane source machine characteristics

```
source_machine.agent_directory : /dir                  /usr/local/bin/ptools
source_machine.remote_rcs_directory : /dir             /usr/sbin
source_machine.local_rcs_directory : /dir             /usr/sbin
source_machine.quiet_option : { on                    off
                                off }
```

ptools Configuration Options (continued)

● Triggering census db backup

```

census_db.backup_required : { true }           true
                           { false }
census_db.rcs_backup_required : { true }       true
                               { false }
census_db.time_between_backups : seconds (0 to ~68 years)    14400 (4 hours)
census_db.time_between_rcs_backups : seconds (0 to ~68 years) 172800 (2 days)
census_db.percentage_increase_before_backup : 1 .. 100      33
  
```

● Census database characteristics

```

census_db.census_filename: /filename           workarea.location/census
census_db.backup_filename: /filename           workarea.location/.census
census_db.wa_location: /dir                    workarea.location
census_db.sm_location: /dir                    workarea.sm_location
census_db.locking_timeout: seconds (0 to ~68 years)          5
census_db.verbose_option: { true }             false
                           { false }
census_db.locking_method: { exclusive }        exclusive
                           { shared }
census_db.preallocation_limit: bytes (1 to ~2 gigabytes)    max(1 Meg, 2*census)
census_db.output_format: { 1 }                 1
                           { 2 }
  
```

How to set up a ptools source tree

- This info will be of use to just a handful of people, chances are most people just need to set up a ptools workarea.
- Install the necessary software
 - inst ptools.sw.ptools_sm from dist.wpd:/sgi/ptools/new
 - inst rfind.sw.server from dist.wpd:/sgi/rfind
 - inst eoe2.sw.rcs
 - rcp dist.wpd:/usr/local/bin/nacct /usr/local/bin
this will allow anyone with NIS account to get an account
- Rfind daemon
 - will be started as part of inst's exitops
 - type 'ps -ef | grep rfindd' to verify it is running
- Rfind accuracy
 - More frequent fsdumps mean more accurate ptools file info
 - More frequent fsdumps mean degraded system performance
 - You get to determine balance appropriate for your source tree
 - see fsdump man page for info on changing fsdump frequency
- Externalize multiple filesystems as one correctly
 - use relative symlinks
 - export filesystems with nohide (including /)

Engineer's Handbook General Stuff

May 5, 1994

SGI Bug System Class Notes

probable data location: `dist.wpd:/sgi/doc/swdev/bugsys.ps`
access via: `handbook bugsys`



Silicon Graphics Inc SGL Bug System

NOTES

Publication Date: 2/12/93

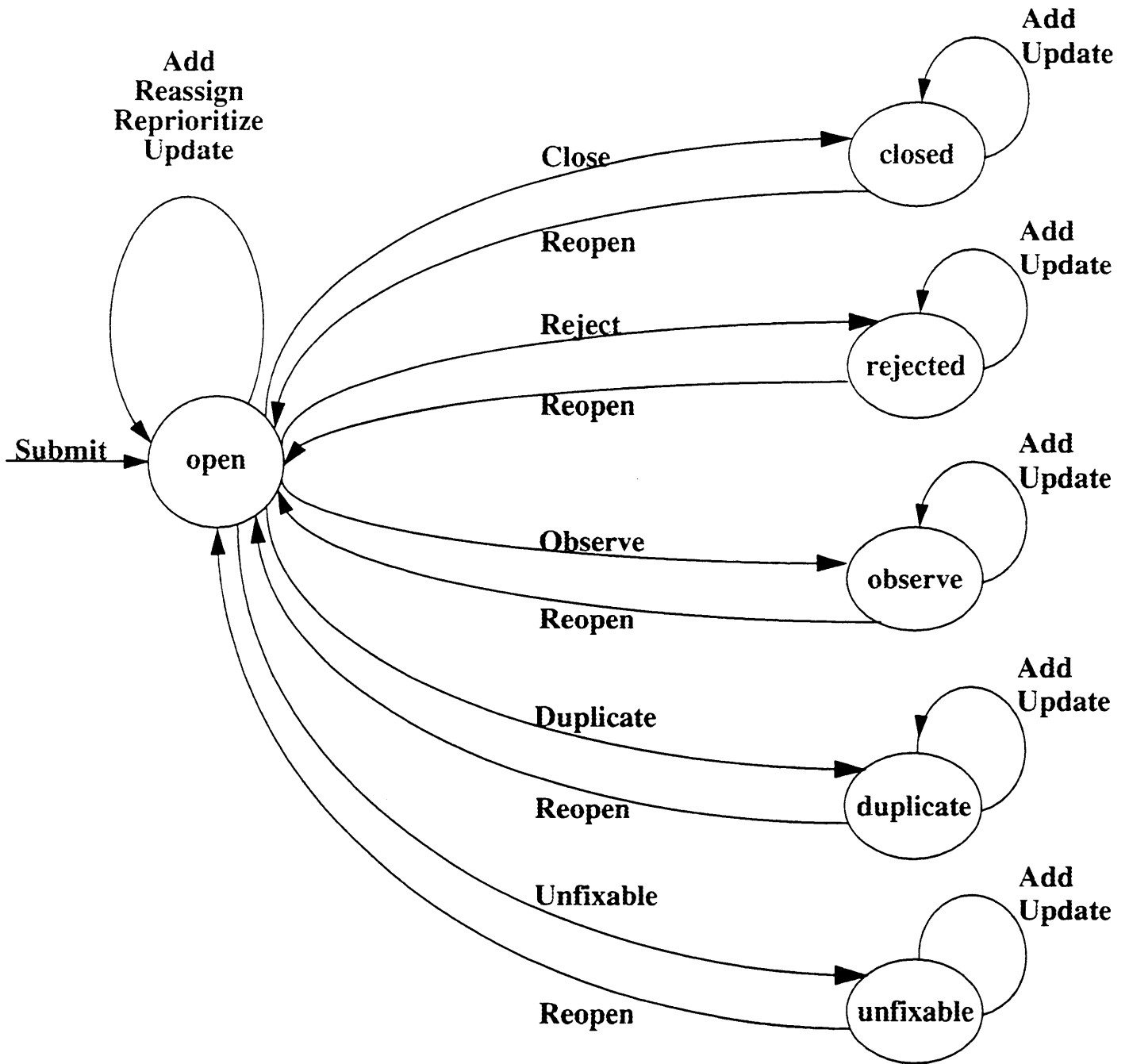
Agenda

- Background (page 2)
- pv State Diagram (page 3)
- Walk through pv windows (page 4)
- Data Flow (page 5)
- Sorting and Searching (page 7)
- Installation of software (page 8)
- pvquery (page 9)
- Configuration files (page 10)
 - ~/.pvdefaults (page 11)
 - ~/.Xdefaults (page 19)

Background

- Prior system
 - newsgroup based
 - Engineer assigned by manual process
- New system
 - Sybase based C++ with gui and report generator
 - added search capabilities
 - assist in assigning to engineer
 - assist in providing info in mail messages
 - immediate database update

pv State Diagram

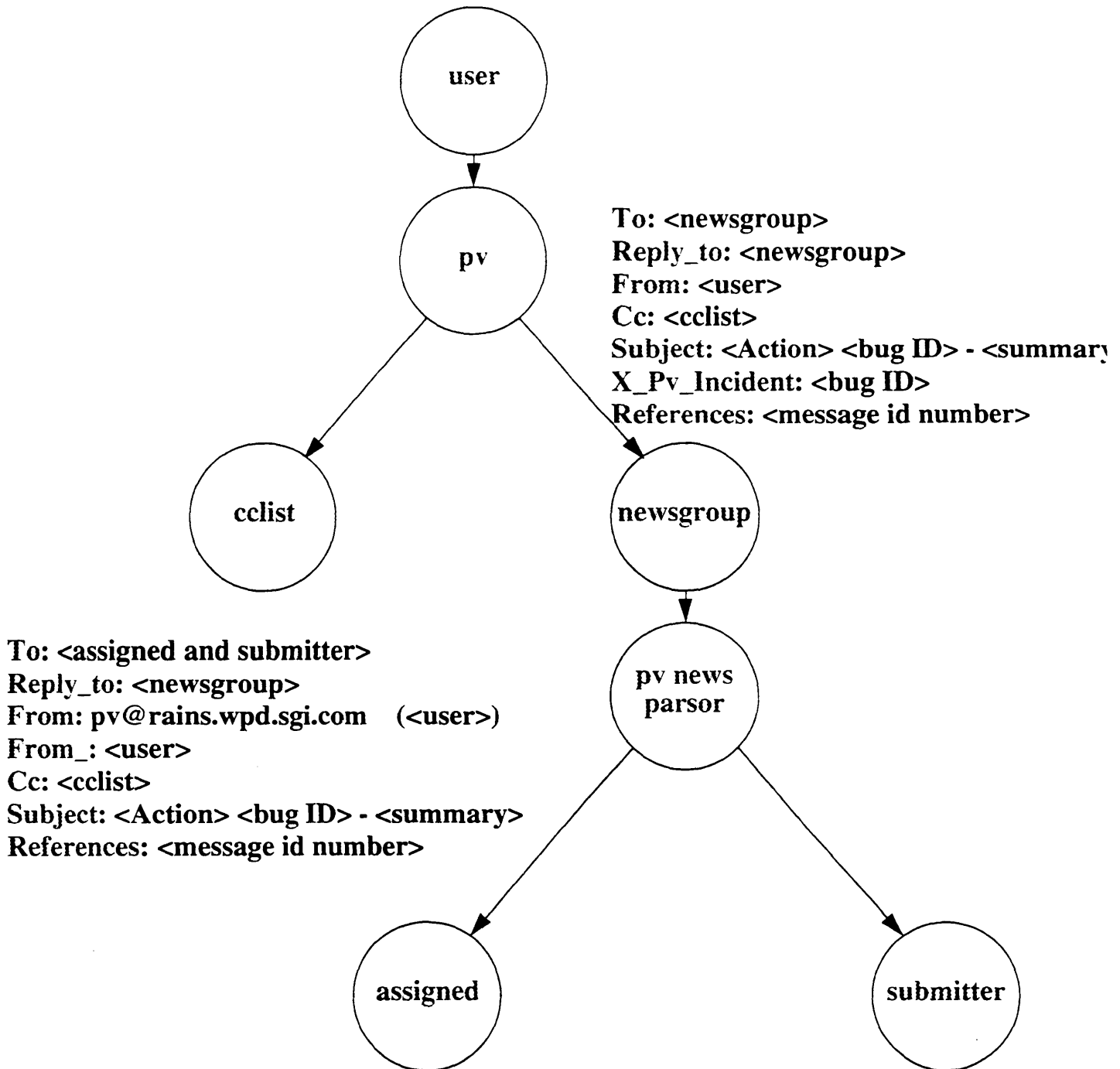


Note: Actions that end in the **open** state are logged to the **description** field. Actions that end in a **non-open** state are logged to the **fix_description** field.

Walk through pv windows

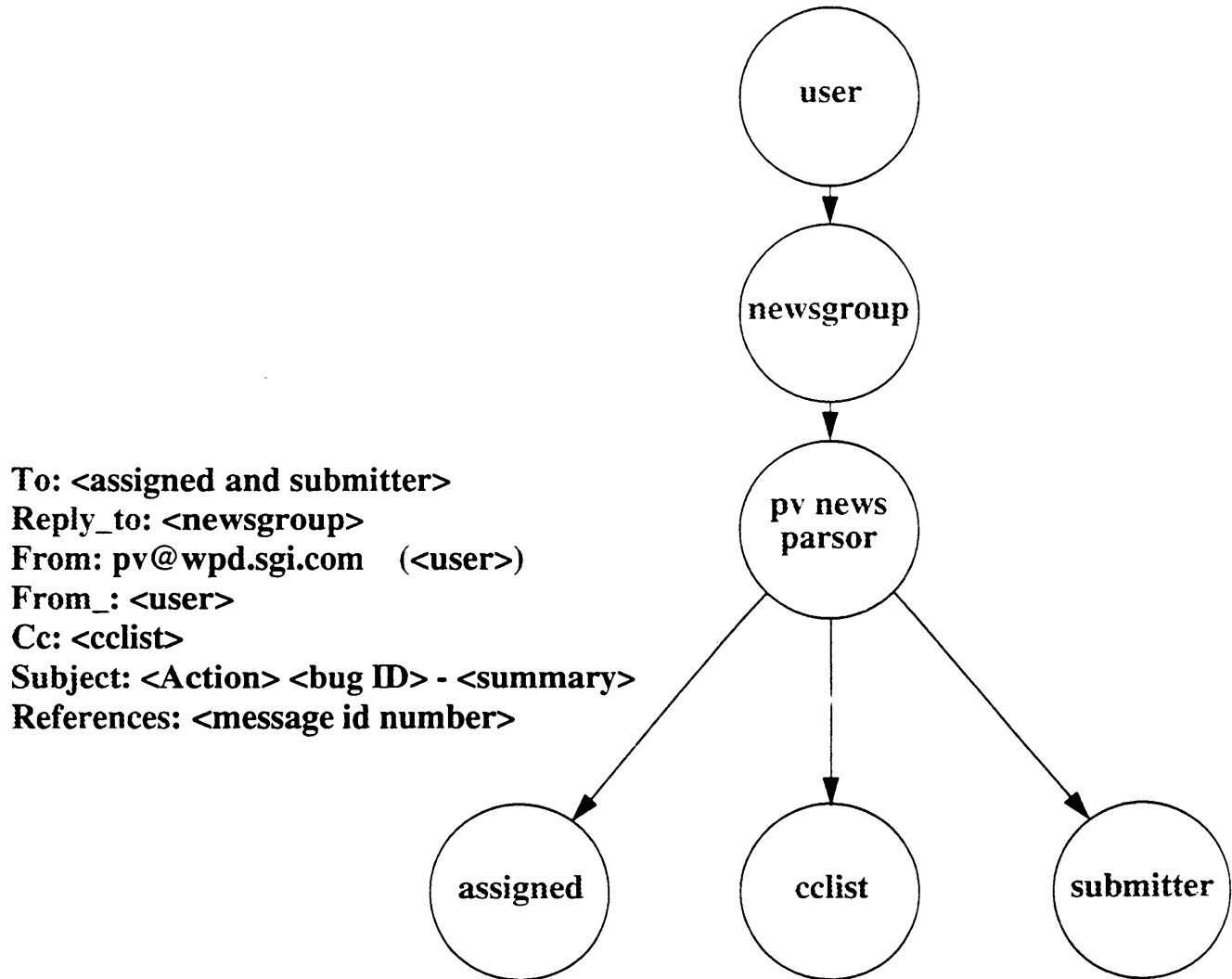
- Main window
 - scrolled list
 - Query parameters
 - default query
 - custom queries
 - buttons
 - Menu pull downs
 - Expanded query sub-window
- Read window
 - View/update fields
 - Action pull down (actions in state diagram)
 - other buttons
 - Menu pull downs
 - expanded form of read window
 - frozen windows
- Submit window
 - must set fields (highlighted)
 - preset fields by pv
 - buttons
 - default submit
 - custom submit

Mail Flow through pv



Note: The **References** field is not used with the initial **bug** or **rfe**, since there is no other message to reference.

Mail Flow through newsgroup



Note: The pv news parser is an automated process that (when X-Pv-incident header is not present)

- 1) Assigns/identifies bug IDs
- 2) derives assigned engineer/group for the pv database
- 3) **References** field does not appear if bug or rfe has just been generated..

Sorting and Searching

- Sort Index
 - available from options pull-down menu
 - see `'pvquery -w'` to see available field names
 - preface with a `'!` to reverse sort the field
- Searching
 - see `pvquery(1)` for additional information
 - Predefined values
 - `null` - match if entry is null
 - `today [- <number>]` - alias date relative to today's date
 - range operator
 - `<value> : <value>`
 - list operator
 - `<value>, <value>`
 - relational operators (entry is implied first operand)
 - `= <value>`
 - `> <value>`
 - `< <value>`
 - `>= <value>`
 - `<=<value>`
 - `!= <value>`
 - `!> <value>` (equivalent to `<=`)
 - `!< <value>` (equivalent to `>=`)
 - implied "and" between fields
 - wildcards (like file name completion)
 - `?` - matches one character
 - `*` - matches zero or more characters
 - `[...]` - matches any one of enclosed characters
 - `[^...]` - matches any one character but those enclosed
 - (**Note:** `[a-zA-Z]` specifies a range of characters)

Installation of Software

% inst -f dist.wpd:/sgi/pv/new

- yields
 - pv
 - pvquery
 - man pages (including pvintro(1))

% inst -f dist.wpd:/released/4.0.5/products/WORKSHOP/CaseVision

Inst> install CaseVision.sw.help

- yields
 - CaseVision online help
 - CaseVision schemes (useful to replace default pv scheme)

pvquery

- line oriented
- accepts superset of alphamore parameters
- some report generation facilities

Configuration files

- ~/.pdefaults
 - Main Window (page 11)
 - Choose fields in scrolled list
 - Choose fields in query window
 - Read Window (page 12)
 - Choose fields displayed
 - Choose actions available from **Action** button
 - Choose fields that can be updated
 - Choose fields to include in mail messages (by action)
 - Submit window (page 15)
 - Choose default view
 - Choose fields to include in mail messages (by view)
 - Miscellaneous options (page 16)
 - pv.editor_command
 - must open own window - i.e. **xwsh -e vi**
 - else uses **\$WINEDITOR** - forks own window
 - avoid the endless loop
 - else uses **\$EDITOR** - doesn't fork own window
 - pvquery options (page 17)
- ~/.Xdefaults
 - focus policy (page 19)
 - explicit (tabs are active)
 - pointer (don't need to click first)
 - reset button labels (page 20)
 - reset labels and mnemonics in pull downs (page 22)
 - key translations (page 25)
 - miscellaneous (page 26)

Sample ~/.pvdefaults file (main window)

```
# Application based defaults
# - /usr/local/bin/pv-app-defaults/Pv
# User application based defaults - ~/.pvdefaults

# Note: you can obtain field names by typing 'pvquery -w'

#####
# Scrolled list portion of main window
#####

# fields to use
bug.pvprojectincident.Main.List.attributes : incident_id,\
      priority, fix_policy, assigned_engineer, age,\
      classification, summary

#####
# Query portion of the main window
#####

# fields to use
bug.pvprojectincident.Main.Query.attributes : incident_id,\
      assigned_engineer, priority, status, opened_date,\
      submitter, assigned_group, project, closed_date, summary

# number of columns
bug.main.columns      : 2
```

Sample ~/.pvdefaults file (read window)

```
#####  
# Read window  
#####  
  
# fields to use  
bug.pvprojectincident.Read.attributes : incident_id,\  
    submitter, priority, status, opened_date,\  
    assigned_engineer, assigned_group, project, summary,\  
    description, fix_description  
  
# number of columns  
bug.Read.small_format.columns      : 2  
  
# List of actions for button in read menus  
bug.pvprojectincident.actions : Close, Reject, Add, Append,\  
    Duplicate, Reassign, Reopen, Reprioritize, Observe,\  
    Update  
  
# List of fields that can be updated via update button  
bug.pvprojectincident.update_fields : command,\  
    classification, summary, project, fix_policy, model_gfx,\  
    reproducible, dev_priority, fixed_by, irix_release,\  
    product_version, machine, peripheral, SGI_only,\  
    category, alpha, model_cpu, doc_affected, CSD_priority  
  
# number of columns in expanded read window  
bug.Read.full_format.columns      : 3
```

Sample ~/.pvdefaults file (read window mail options)

```
#####  
# Read window Mail options  
#####  
  
# fields to include in mail generated by the Close action  
bug.pvprojectincident.mail_fields.close :\  
    assigned_engineer, submitter, status, priority,\  
    fix_description, description  
  
# fields to include in mail generated by the Reject action  
bug.pvprojectincident.mail_fields.reject :\  
    assigned_engineer, submitter, status, priority,\  
    fix_description, description  
  
# fields to include in mail generated by the Add action  
# (only one using default)  
bug.pvprojectincident.mail_fields.default :\  
    assigned_engineer, submitter, status, priority,\  
    description  
  
# fields to include in mail generated by the Append action  
bug.pvprojectincident.mail_fields.append :\  
    assigned_engineer, submitter, status, priority,\  
    fix_description, description  
  
# fields to include in mail generated by the Duplicate action  
bug.pvprojectincident.mail_fields.duplicate :\  
    assigned_engineer, submitter, status, priority,\  
    fix_description, description  
  
# fields to include in mail generated by the Reassign action  
bug.pvprojectincident.mail_fields.reassign :\  
    assigned_engineer, submitter, status, priority,\  
    description
```

Sample ~/.pvdefaults file (read window mail options - continued)

```
# fields to include in mail generated by the Reopen action
bug.pvprojectincident.mail_fields.reopen :\
    assigned_engineer, submitter, status, priority,\
    fix_description, description

# fields to include in mail done by the Reprioritize action
bug.pvprojectincident.mail_fields.reprioritize :\
    assigned_engineer, submitter, status, priority,\
    description

# fields to include in mail generated by the Update action
bug.pvprojectincident.mail_fields.update :\
    assigned_engineer, submitter, status, priority,\
    description

# fields to include in mail generated by the Observe action
bug.pvprojectincident.mail_fields.observe :\
    assigned_engineer, submitter, status, priority,\
    fix_description, description

# fields to include in mail done by a Submit (from engr view)
bug.pvengrincident.mail_fields.default : assigned_engineer,\
    submitter, status, priority, description

# fields to include in mail done by a Submit (from csd view)
bug.pvcsdincident.mail_fields.default : assigned_engineer,\
    submitter, status, priority, description
```

Sample ~/.pvdefaults file (submit window)

```
#####  
# Submit Window  
#####  
  
# default view when doing a submit  
# Choices are: PvEngrIncident, PvCsdIncident, PvCustomer  
bug.primary_submit_view: PvEngrIncident  
  
# fields to include in mail generated by the Engr view  
bug.pvengrincident.mail_fields.default : assigned_engineer,\  
    submitter, status, priority, description  
  
# fields to include in mail generated by the CSD view  
bug.pvcsdincident.mail_fields.default : assigned_engineer,\  
    submitter, status, priority, description  
  
# fields to include in mail generated by the Customer view  
# gotcha --- customers don't generate a submit,  
# so no mail fields.
```

Sample ~/.pvdefaults file (miscellaneous)

```
#####
# Editor options (must open a new window)
#####

pv.editor_command: xwsh -e vi
# Note: If pv.editor_command is not set then the envariable
# WINEDITOR will be used, which must open a new window (via
# xwsh if necessary).
# If WINEDITOR is not set then the envariable EDITOR will be
# used which must not open a new windoe via xwsh.
# Wrong setting can result in an infinite loop.

#####
# Immediate custom query
#####

# If the resource is false (default) then selecting a custom
# query will fill fields in both the main and expanded
# windows. The expanded window fields will only be used for
# searches made from the expanded window.
#
# If the resource is true then an immediate query will be made
# from both the main and expanded window fields. Subsequent
# searches will only uses the expanded window fields if the
# search is made from the expanded window

pv.search_on_query_retrieval: F

#####
# Immediate search after sort order change
#####

pv.search_on_sort_change: T
```

Sample ~/.pvdefaults file (pvquery options)

```
#####  
# Set summary fields (corresponds to -B and default options)  
#####  
  
pvquery.summary_fields: assigned_engineer, summary  
pvquery.summary_format: %-7.7s %s\n  
  
#####  
# Set long fields (corresponds to -L option)  
#####  
  
pvquery.long_fields: incident_id, assigned_engineer  
  
#####  
# Set report fields (corresponds to -F option)  
#####  
  
pvquery.report_fields: incident_id, summary  
pvquery.report_format: %-6.6s %s\n
```

Sample pv entries for ~/.Xdefaults file (focus policy)

```
! Application based defaults
!   /usr/lib/X11/app-defaults/Pv
! User X defaults (for all X programs) - ~/.Xdefaults

! Note: when specifying names, a parenthetical entry will be
! used to show the actual default value when it differs from
! the name.
! For instance, in the main window, the button named "new"
! has a label of "Submit", so it will appear as "new(Submit)".

!!!!!!!!!!!!!!
! focus policy
!!!!!!!!!!!!!!

! two possible values:
!   explicit (can use tabs to move between fields)
!   pointer (no need to click in field, just move pointer)

pv*keyboardFocusPolicy:   explicit

! Note: X bug may require that resource `*keyboardFocusPolicy`
! be set instead
```

Sample pv entries for ~/.Xdefaults file (button labels)

```
!!!!!!!!!!!!!!!!!!!!
! button labels
!!!!!!!!!!!!!!!!!!!!

!
! Button label names are specified
! (local to a particular window) via the line:
!
!pv*<window_name>*<button_name>.labelString: <string>

! for instance:
pv*main*quit.labelString: QuitPv

!
! You can also set all occurrences of a <button_name> via:
!

!pv*<button_name>.labelString: <string>

! for instance:
pv*close.labelString: CloseWin
```

Sample pv entries for ~/.Xdefaults file (names of button label by window)

```
! Names of button label by window

! <window_name> = main
!   read          new(Submit)      reports
!   quit

! <window_name> = read
!   apply         undo             expand
!   shrink       next            previous
!   quit(Close)

! <window_name> = submit
!   apply(Submit) expand          shrink
!   new          clear          close

! <window_name> = query
!   count       clear          search
!   close      expand

! <window_name> = text
!   apply(Done) cancel
```

Sample pv entries for ~/.Xdefaults file (menu bar)

```
!!!!!!!  
Menu bar  
!!!!!!!  
  
!  
! Menu bar label names and a hot key (mnemonic) letter  
! in the name are defined as:  
!  
!pv*<menu_bar_name>.labelString: <string>  
!pv*<menu_bar_name>.mnemonic: <letter>  
  
! for instance:  
pv*file.labelString: File  
pv*file.mnemonic: F  
  
!  
! Within a menu bar  
!  
  
! Button label names and a hot key (mnemonic) letter  
! in a button name are defined as:  
  
!pv*<menu_bar_name>*<button_name>.labelString: <string>  
!pv*<menu_bar_name>.<button_name>.mnemonic: <letter>  
  
! for instance:  
pv*file.quit.labelString: Quit  
pv*file.quit.mnemonic: Q
```

Sample pv entries for ~/.Xdefaults file (menu bar and names of its buttons)

```

! Optionally, accelerator keystrokes
! and the text labeling them are defined as:

!pv*<menu_bar_name>.<button_name>.accelerator: <string>
!pv*<menu_bar_name>.<button_name>.acceleratorText: <string>

! for instance:
pv*file.quit.accelerator: Ctrl<Key>C
pv*file.quit.acceleratorText: Ctrl+C

! Accelerator keystrokes define a character sequence that is
! globally available to perform the button's action

! Names of button label by menu bar name

! <menu_bar_name> = file
!   save          saveas          print
!   iconify       uniconify(Raise) freeze
!   close         quit           mail

! <menu_bar_name> = options
!   ordering(Sort Order ...)query(Custom Querying ...)
!   tracing

! <menu_bar_name> = views
!   read          submit

! <menu_bar_name> = related
!   none

```


Sample pv entries for ~/.Xdefaults file (menu label in View menu bar)

```

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Read Menu labels in View menu bar
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! labels are defined as:
!
!pv*readpv<item_name>.labelString: <string>

!   projectincident      customer
!   people                group(Groups)
!   projectnewsgroup(Project -> Newsgroups)
!   projectmap(Project -> Subproject)
!   groupepeople(People -> Groups)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! submit Menu labels in View menu bar
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! labels are defined as:
!
!pv*submitpv<item_name>.labelString: <string>

!   engrincident          csdincident
!   customer              people
!   group(Groups)
!   projectnewsgroup(Project -> Newsgroups)
!   projectmap(Project -> Subproject)
!   groupepeople(People -> Groups)

```

Sample pv entries for ~/.Xdefaults file (Key translations for text editing)

```

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Key translations for text editing
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Pv*XmText.translations: #augment\n\
    !Ctrl<Key>p:          previous-line()\n\
    !Ctrl<Key>n:          next-line()\n\
    !Ctrl<Key>b:          backward-character()\n\
    !Ctrl<Key>f:          forward-character()\n\
    !Ctrl<Key>a:          beginning-of-line()\n\
    !Ctrl<Key>e:          end-of-line()\n\
    !Meta<Key>b:          backward-word()\n\
    !Meta<Key>f:          forward-word()\n\
    !Meta<Key>[:          backward-paragraph()\n\
    !Meta<Key>]:          forward-paragraph()\n\
    !Ctrl<Key>v:          next-page()\n\
    !Meta<Key>v:          previous-page()\n\
    !Meta<Key><:          beginning-of-file()\n\
    !Meta<Key>>:          end-of-file()\n\
    !Meta Shift<Key>comma: beginning-of-file()\n\
    !Meta Shift<Key>.:    end-of-file()\n\
    !Ctrl<Key>d:          delete-next-character()\n\
    !Ctrl<Key>h:          delete-previous-character()\n\
    !Meta<Key>d:          kill-next-word()\n\
    !Meta<Key>BackSpace: kill-previous-word()\n\
    !Ctrl<Key>k:          kill-to-end-of-line()\n\
    !Ctrl<Key>w:          kill-selection()\n\
    !Ctrl<Key>y:          unkill()\n\
    !Ctrl<Key>l:          redraw-display()\n\

```

Sample pv entries for ~/.Xdefaults file (miscellaneous labels)

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Help menu bar labels and mnemonics
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
pv*helpMenu.labelString:           Help
pv*helpMenu.mnemonic:              H
pv*helpMenu.aboutApplication*labelString:  On Version...
pv*helpMenu.aboutApplication*mnemonic:    V
pv*helpMenu.helpOnContextMenuItem.labelString:  On Context
pv*helpMenu.helpOnContextMenuItem.mnemonic:    C
pv*helpMenu.helpOnWindowMenuItem.labelString:  On Window...
pv*helpMenu.helpOnWindowMenuItem.mnemonic:    W
pv*helpMenu.helpIndexMenuItem.labelString:    Index...
pv*helpMenu.helpIndexMenuItem.mnemonic:      I
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Query labels in Options menu bar
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
pv*query.create.labelString:       Create...
pv*query.create.mnemonic:          C
pv*query.remove.labelString:       Remove...
pv*query.remove.mnemonic:          R
```


Engineer's Handbook General Stuff

May 5, 1994

Paul Haeberli's Big Book of Names

access via: handbook bookofnames
probable data location: macke.asd:/usr/people/paul/BBON

4DDN	DECnet support for IRIS-4D
4DLT	LAT support for IRIS-4D
4Dwm	The IRIS extended motif window manager
6U	width of 6 unit wide VME bus board
9U	width of 9 unit wide VME bus board
A-law	CCITT G.711 64 kb/sec compression
AAU	Alpha Acid Units; oz. hops * alpha acid rating
ABI	Application Binary Interface
ACE	Advanced Computing Environment
ACM	Association for Computing Machinery
ADPCM	Adaptive Differential Pulse Code Modulation
AGD	Advanced Graphics Division
AL	Audio Library
API	Application Programmatic Interface
ASAP	As Soon As Possible
ASD	Advanced Systems Division (deceased)
ATM	Asynchronous Transfer Mode; part of B-ISDN
ATT	American Telephone and Telegraph, Inc.
AUD2	Audio i/o chip for Indigo
AXM	Ada with X and Motif Bindings
Arques	MIPS internal name for R2010
Aspen	Internal name for IRIX 3.3; Same as Birch?
B-ISDN	Broad-band ISDN; CCITT standards for high-speed networking
BB	Basic block, also Black Bird, development name for TFP
BBON	Big Book Of Names; a product of KHI
BCC	Blind Carbon Copy
BE	Big Endian
BGO	Big Glimpse of the Obvious
BOM	Bill of Materials
BSD	Berkeley Software Distribution
BSD	Business Support Division in VSG
BTW	By The Way
BVD	source tree for 4.0.5[D-G]; (Blackjack-Venice-Digital media)
BVO	Broadcast Video Option
Betacam	Sony-developed YUV recording medium
Birch	Internal name for IRIX 3.3; same as Aspen??
Blackjack	Internal name for IP20; Indigo R4000
BlueLight	PI with a 15" 1024x768 display
CASE	Computer-Aided Software Engineering
CASEvision	SGI's CASE product
CC	Carbon Copy
CCA	Customer call Administrator; Obsoleted by CSR
CCIR	Consultative Committee on International Radio; an ITU committee
CCIR-601	Component digital video format; aka D1, aka 4:2:2
CCITT	Consultative Committee on International Telephony and Telegraphy; an
CD	Compact Disk
CD-ROM	Compact Disk - Read Only Memory
CDDI	FDDI using unshielded twisted pair wiring for lower cost
CG2	Genlock and Composite Video Option (9U) ASD machines only
CG3	Genlock and Composite Video Option (6U) ASC and ESD machines
CID	Clip ID, provides hardware clipping for windows
CIF	Common Intermediate Format; 352 by 288 pels.
CMYK	Cyan, Magenta, Yellow and Key (Black)
COFF	Common Object File Format; pre-Sherwood object file format
CPI	Continuing Process Improvement
CRC	Camera Ready Copy, documentation turned over to production (before MF
CRC	Cyclic Redundancy Checksum
CSD	Customer Support Division
CSE	Customer Services Engineering. This is customer phone support
CSR	Customer Support Representative
Charleston	MIPS internal name for R2000
Clover1	IRIS 4D/G; pre GT graphics
Clover2	IRIS 4D/GT 4D/GTX graphics
Crimson	SGI's first R4000 system from ASD
Cypress	Internal name for IRIX 4.0 project
D1	Component digital video format

D2	Composite digital video format
DAT	Digital Audio Tape
DBMS	Data Base Management System
DCT	Discrete Cosine Transform
DDA	Digital Differential Analyzer
DEC	Digital Equipment Corporation
DECO	Desktop Conferencing
DFT	Discrete Fourier Transform
DID	Display ID, determines pixels display mode
DMA	Direct Memory Access
DME	Diastatic Malt Extract
DME	Distributed Management Environment; another OSF middleware product
DME	Dry Malt Extract
DMS	Document Management System, where docs are checked in and out.
DPE	Documentation Prose Engineer; see "Slacker"
DPS	Display PostScript
DSM	Distributed Shared Memory
DSO	Dynamic Shared Object; SVR4's shared library
DSP	Digital Signal Processing
DSSD	Digital Sight and Sound Division
DTD	Document Type Definition; an SGML term defining the structure of a doc
DTD	Document Technical Description; an SGML ism
DTS	Dead Toolkit Society
DVI	Digital Video Interactive; Intel system for Multimedia on CDROM
DWB	Documenter's WorkBench
Da Vinci	PI 24-bitplane graphics without Z-buffer ???
ECO	Engineering Change Order; used to tell manufacturing to make a change
ECOFF	Extended COFF; same as XCOFF
EDI	Electronic Data Interchange; electronic exchange of business data
EEPROM	Electrically Erasable PROM
EFIB	Entry systems Field Information Bulletin
EFS	Extent File System; the native IRIX file system
EISA	Extended Industry Standard Architecture
ELF	Extended Linking Format; object file format for SVR4 and Sherwood
ELF	Executable and Linking Format: object file format for SVR4 and Sherwo
ELF	Extremely Low Frequency; Used by the Navy for submarine communication
EMI	Electro-Magnetic Interference
EMR	Electro-Magnetic Radiation
EMT	Emergency Medical Technician
EOE	Execution Only Environment
EPROM	Erasable PROM
EPSF	Encapsulated PostScript
ER	IRIS InSight Error Reporting tools
ESD	Electro-Static Discharge
ESD	Entry Systems Division (deceased)
EUC	End User Computing
Eclipse	internal name for the PI
Elan	4 GE version of the Express graphics system
EntryGraphics	Internal name for Starter Graphics
Everest	MP R4000/R4400 system
Express	high-performance graphics system for Indigo and 4D/35
Extreme	8 GE version of Express graphics system (?)
FAQ	Frequently Asked Question; or summary of answers to FAQ's
FCS	First Customer Ship
FDDI	Fiber Distributed Data Interconnect
FDR	Full Disclosure Reports; submitted to all the members of the TPC
FE	Field Engineer
FFT	Fast Fourier Transform
FIB	Field Information Bulletin
FIFO	First In, First Out
FIR	Finite Impulse Response Filter
FSF	Free Software Foundation
FTN	Fortran
FYFYF	F*ck you Frank you F*ck; a palindrome, for which the Frank conference
FYI	For Your Information
FlexKit	Old, defunct user interface toolkit

Fullhouse	Indigo 2; SGI machine with EISA bus.
G.723	CCITT mumble foo.24 kb/s ADPCM
GDI	Windows NT Graphics Device Interface
GE	Geometry Engine
GIO	??? - a bus
GL	Graphics Library
GLX	OpenGL X extension
GNU	"GNU's Not Unix"; a Unix clone under development by the FSF
Grinch	Development name for GIO to MP bus adapter
Guinness	Former internal name for Indy
H.261	CCITT Recommendation H.261; aka p*64. ISDN compression standard
HAL	Windows NT Hardware Abstraction Layer
HAL2	Audio i/o chip for Guinness and Full House
HDTV	High Definition Television; available as a video format on VGX and RE
HSI	Hue, Saturation, Intensity; a color system
HSV	Hue Saturation and Value color model
HSg	Hue, Saturation and Value; a color system
HZ	Hertz; cycles per second
Hi-8	Highband 8mm; A consumer/industrial quality video tape format
Hollywood	Internal name for Indigo system
Hotline	Customer support hotline. Obsoleted by TAC
HyTime	Hypermedia/Time-based Structuring Language - ISO/IEC 10744
ICE	low-cost, low-power version of the R4000 being codeveloped with NEC
IDB	Image Database; as in inst idb tags
IDO	IRIX Development Option
IEC	The International Electrotechnical Commission (Swiss Name)
IIR	Infinite Impulse Response Filter
IL	ImageVision Library
IM	IRIS InterfaceMaker; Name for SGI's Motif port
IMHO	In My Humble Opinion
IMPC	Impressario, Client
IMPD	Impressario, Developers
IMPS	Impressario, Server
INITTAB	Initialize Network interfaces, ttys, and bring-up.
INRI	Jesus of Nazareth, King of the Jews
IO1	???
IO2	???
IO3	???
IO4	???
IP	Internet Protocol
IP	Iris Processor
IP	Instruction Processor
IP10	20MHz PI processor (4D/25)
IP12	The processor board for the Magnum 4D30 (30MHz) and 4D35 (~37MHz) and
IP13	2x33MHz R3000A multiprocessor for 4D/300 series
IP15	2x40MHz R3000A multiprocessor for 4D/400 series
IP17	50MHz R4000 processor board for Crimson (4D/510)
IP19	Everest multiprocessor board
IP20	50MHz R4000 processor board for Indigo, aka Blackjack
IP22	Guinness processor board
IP4	12MHz R2000A processor board for 4D/70 (Clover)
IP4.5	16.7MHz R2000A processor board for 4D/80
IP5	2x16.7MHz R3000[A] multiprocessor board (4D/120)
IP6	12.5MHz R3000 PI processor (4D/20)
IP7	2x25MHz R3000A multiprocessor for 4D/200 series
IP9	25MHz R3000A processor for 4D/210
IRE	Institute of Radio Engineers
IRIS	Integrated Raster Imaging System
IRIS-IM	IRIS InterfaceMaker; Name for SGI's Motif port
IRIX	IRIS Unix; SGI's version of ATT System V Unix
ISAC	Information Services Assistance Center
ISD	Interactive Systems Division
ISDN	Integrated Services Digital Network; It Still Does Nothing
ISM	Inst Selectable Module; any installable software package
ISO	The International Organization for Standardization (Swiss Name)
ISV	Independent Software Vendor

ITU	International Telecommunications Union; an agency of the UN
Impressario	PostScript printing tools for the IRIS
InSight	IRIS online documentation viewer
Indigo	IRIS Indigo workstation (also, that family of workstations)
Indy	SGI's latest low-end Indigo-family machine
Inst	Our IRIX software installation program
Intrepid	old name for Neon.
JPEG	An image compression standard; Joint Photographic Experts Group
Jasper	???
Juniper	3000 Product Internal Code Name
KAI	Kuck & Associates - contractors for PFA, PCA
KHI	Karsh-Haeberli Industries; the producers of the BBON
LAT	Local Area Transport; DEC protocol for terminal servers
LE	Little Endian
LG1	bluelight
LPC	Linear-predictive coding
LPF	League for Programming Freedom
LPS	Laser Printer Support
LSMFT	Lucky Strike Means Fine Tobacco
LVD	Live Video Digitizer
Lego	system administration tools for Guinness.
Lonestar	Internal name for IP17 (Crimson) project
M2	Panasonic version of Betacam; aka MII
MBX	X MultiBuffering extension
MCO	Multi-Channel Option, the VS2
MG1	board name for Grinch
MIDI	Musical Instrument Digital Interface
MII	Panasonic version of Betacam; aka M2
MIPS	Million Instructions Per Second
MM	Media Mosaic
MOD	Manager On Duty. For angry customers
MP	Multi-processor system
MPEG	An movie compression standard; Motion Picture Experts Group
MR	Manufacturing Release
MTI	MIPS Technologies Inc.
MTR	Material Transfer Request
MTS	Member of Technical Staff
Magnum	System that used an IP12 processor in a modified PI chassis
Mirage	Venice graphics on Everest
Motif	a standard toolkit for use under X
NB	Nota Bene; literally "Note Well"
NEC	Nippon Electric Company
NFS	Network file system
NIS	Network Information Service; Same as YP
NQS	Network Queueing System
NT	Windows New Technology; Microsoft's new operating system
NTSC	National Television Standards Committee
NTSC	Never Twice the Same Color
NTSC	TV broadcast standard for North America, and much of South America ar
NURB	Non-Uniform Rational B-spline
NURBS	Non-Uniform Rational B-Spline
Neon	3D multimedia generalized environment whatever that is
Newport	graphics hardware for Guinness
Newpress	Indigo with 1 Newport head and 1 Express head
OEM	Original Equipment Manufacturer
OSF	Open Software Foundation
OSI	Open Systems Interconnect ???
OSI	Open Systems Interconnection networking model produced by ISO
Oasis	CSE's central info repository. bugs, newsgroups, calls, are archived
OpenGL	the next generation graphics library
OpenLook	Sun's X user interface look and feel
Oz	new desktop software (update of Workspace???)
PAL	Phase Alternating Line aka Please All Lobbies
PAL	TV broadcast standard for Western Europe (except France), Australia,
PCA	Power C Analyzer
PDA	Processor Data Area

PDC	Pilot Document Controller ???
PDQ	Pretty Darn Quick
PEX	PHIGS-styled X extension for 3D, competes with OpenGL
PFA	Power Fortran Analyzer
PFTN	Power Fortran
PIO	Physical (raw) Input and/or Output
PIO	Pilot Induced Oscillation
PO	Purchase Order
POCN	Purchase Order Completion Notice
POD	Potential Organ Donor
PROM	Programmable Read-Only Memory
PS	PostScript page description language from Adobe
PSE	Product Support Engineer. The people who call the customers back and
PV	ProjectVision
PWRC	Power C
ProjectVision	One of SGI's new bug tracking systems;
QA	Quality Assurance
QCIF	Quarter-size CIF. 176 by 144 pels
R4K	MIPS R4000 processor
RCS	Revision Control System, software revision software used by SGI
RDBMS	Relational Data Base Management System
RE	RealityEngine
RE1	Raster Engine 1; graphics chip in GR1.1 PI
RE2	Raster Engine 2; graphics chip in GR1.2 PI
RFC	Request for Comment; mostly for network standards
RFE	Request For Enhancement
RGB	Red Green and Blue color model
RMS	Record Management System; a DEC file system
RMS	Richard Stallman; founder of the FSF and LPF
RMS	Root mean squared; the square root of the mean squared value
RN	Rendering Node, virtual gfx pipe supplied by RRM
ROOT	environment variable for base directory for libraries, headers, etc.
ROXY	project to develop integrated user interface development tools
RPC	Remote Procedure Call; used by NFS
RRM	Rendering Resource Manager, kernel graphics support
RTC	Release To Control; Manufacturing owns product after this
RTFM	Read the F*cking Manual; Read the Fine Manual
RealityEngine	Venice graphics hardware
Repository	former name for DMS
Roxy	Indigo Magic Developer Kit
Ruby	Internal name for tech pubs project to redesign our printed and on-li
S-VHS	Super VHS; A consumer/industrial quality video tape format
S-Video	a 2-component video format; used by many Hi-8 and S-VHS decks
S4D	Part of Marketing code name for server OS
SBWC	Stink Bush Works Company (will be famous one day)
SCCS	Source Code Control System, software revision software by AT&T
SE	Systems Engineer
SECAM	Sequential Couleur Avec Memoire (Sequential Color With Memory) French
SECAM	TV broadcast standard used in France, the Middle East, parts of Africa
SECAM	Something Essentially Contrary to the American Method
SEM	IRIS InSight System Event Monitor
SGI	Silicon Graphics, Inc.; the mug and t-shirt company
SGI	Soka Gokai International
SGI	Sporting Goods, Inc.
SGML	Standard Generalized Markup Language; a document interchange format
SIGGRAPH	ACM Special Interest Group for Computer Graphics
SKU	Stock Keeping Unit
SMD	Storage Module Device
SMOP	Small Matter Of Programming
SMPTE	Society of Motion Picture and Television Engineers time code for video
SNA	System Network Architecture
SO	Significant Other
SONET	Synchronous Optical Network; Bellcore's optical transmission interface
SSE	Systems Support Engineer; higher grade than FE
STREAMS	AT&T modular device driver interface
SV1	Internal designation for IndigoVideo product

SVID	System V Interface Description
SVR4	System 5 Unix Release 4 from USL
SW	Software
Shamrock	Clover1
Sherwood	IRIX 5.0 based on ATT SVR4 from USL
Shockwave	presentation/seminar on SGI's software
SkyWriter	Dual-headed 10-span VGXT or RealityEngine
Slacker	technical publications writer
Span	Vertical columns of pixels i.e., "10-span VGX"
Stapuft	Development name for VGX graphics
Starter	Lowest cost 1024x768 graphics system for Indigo, aka Entry Graphics
T5	Next gen processor project, as far as we know;
TAC	Technical Assistance Center. Obsoleted by CSE
TBD	ThunderBunny Deskside (obsolete)
TBD	To Be Determined
TBD	To Be Done
TCP	Transmission Control Protocol
TFP	Twin peaks floating point; Twin F*ckin' Peaks
TIA	Thanks In Advance
TIDO	Trusted IRIS Development Option
TIRIX	Trusted IRIX; version of IRIX under Government evaluation at B1
TLA	Three Letter Acronym
TLB	Translation Lookaside Buffer
TLI	Transport Layer Interface (STREAMS network programming interface)
TOOLROOT	environment variable for base directroy for compilers, buid tools, et
TPC	Transaction processing Performance Council; database vendors who defi
TRIX	Trusted IRIX
ThunderBunny	Jasper
TinyUI	New, possibly defunct user interface toolkit
TrIRIX	Trusted IRIX; same as TIRIX
Twin Peaks	High-performance floating point version of R4000
UABBI	Uncle Art's Big Book of IRIX; pronounced "wah bee"
UDP	User Datagram Protocol
UDS	Unix Domain Sockets
UIMX	User Interface Manager for X; Motif interface builder
UN	United Nations
USL	Unix System Laboratories, Inc., owned by Novell
UTS	Unix Time-sharing System
Ultra	8 GE version of the Express graphics system
V-LAN	Video-LAN; a product of Videomedia for controlling video device
VAD	Value Added Dealer
VAR	Value Added Reseller
VC	VideoCreator
VCR	Video Cassette Recorder
VFR	VideoFramer
VFS	Virtual File System; SUN's file system switch architecture, used by S
VGX	1990 highest end graphics machine
VHS	Vertical Helical Scan (not sure about 'v')
VIGRA	Video, Graphics and Audio; a company
VINO	Video in, no out; video chip for Guinness
VL	Video library (under development)
VL	VideoLab
VLI	Another name for VideoLab
VMD	Visual Magic Division
VME	VersaModule Eurocard; 32-bit bus SGI machines use
VO1	VideoLab
VO2	Next generation VidieoLab
VPDA	Virtual Processor Data Area
VS1	VideoSplitter
VS2	VideoSplitter2
VTR	Video Tape Recorder
Venice	1992 highest end graphics machine
W4D	Part of Marketing code name for workstation OS
WCDFT	????
WCID	Worldwide Customer Information Database
WRT	With Regard To, With Respect To

X	The X window system
XCOFF	Extended COFF; same as ECOFF
XDMCP	X Display Manager Control Protocol, used for X terminals
XDR	eXternal Data Representation library; used by NFS & Sun RPC
XS	1 GE Express with 8-bit graphics
XS24	1 GE Express with 24-bit graphics
XS24Z	1 GE Express with 24-bit graphics and hardware z-buffer
XXX	Indicates alcoholic beverages on bottles or in cartoons
XXX	as a C code comment, indicates that the code has room for improvement
XZ	2 GE Express (?)
Xlib	The X client-side protocol library for C
Xsgi	The Silicon Graphics X Server
YAA	Yet Another Acronym
YACC	Yet Another Compiler Compiler
YC	Luminance-Chrominance; a color system
YIQ	A color system used for NTSC encoding
YP	Yellow Pages; A distributed name service now called NIS
YUV	A color encoding scheme Y is perceived brightness, U and V are color
dyDDX	/usr/lib/X11/dyDDX, dir for dynamically linked X server subsystems
il8n	internationalization (i, 18 letters, n)
jwag	Chris Wagner
kb	Kilo byte
l10n	localization (l, 10 letters, n)
lg1	Starter graphics hardware name, appears in code
libui	Old, defunct user interface toolkit
m-law	Mu-law encoding. CCITT G.711. 64 kb/s
ng1	Newport graphics hardware name, appears in code
ninst	new version of inst
p*64	see CCITT H.261 recommendation. p times 64 kilobits/sec
pandora	SGI's graphical login program, ie. chkconfig visuallogin on
pel	See Pixel
pixel	picture element
ptools	software engineering tools for source tree management
pv	projectvision, SGI bug tracking software
ragnorok	2nd generation compilers especially for TFP (bad name, I know)
sec	Seconds
shmiq	SHared Memory Input Queue, input interface for X server
spaceball	3D input device by Spaceball Technologies
svideo	IDB designation for IndigoVideo software
tk	Old, defunct user interface toolkit
toto	mail alias for Guinness user software
ucode	Intermediate code used by MIPS compilers
ViewKit	C++ Application Framework for building Motif Apps
vjs	Vernon Schryver; guru
xdm	X Display Manager, manages X login sessions
Orlando	Time-Warner Full Service Network project
Mardigras	High performance gfx for Indigo II

Engineer's Handbook General Stuff

May 5, 1994

Manuals Section of Price Book

access via: handbook manuals
probable data location: dist.wpd:/sgi/doc/swdev/manuals.ps

TABLE OF CONTENTS

Description	Page
• TPC Benchmark Full Disclosure Reports _____	MAN-1
• STANDARD SYSTEM _____	MAN-1
• DEVELOPMENT OPTION _____	MAN-1
• INDIVIDUAL MANUALS FROM STANDARD AND DEV SYSTEMS _____	MAN-2
• DIGITAL MEDIA DEVELOPMENT LIBRARIES _____	MAN-2
• IMAGEVISION LIBRARY _____	MAN-2
• IRIS EXPLORER _____	MAN-2
• IRIS INVENTOR _____	MAN-3
• IRIS PERFORMER _____	MAN-3
• WINDOW SYSTEMS _____	MAN-3
• CASEVISION a la carte _____	MAN-3
• COMPILERS _____	MAN-3
• NETWORKING _____	MAN-4
• VIDEO OPTIONS _____	MAN-5
• APPLICATIONS/DOCUMENTATION TOOLS _____	MAN-5
• SOFTWARE EXPRESS MANUALS _____	MAN-5



Manuals

CONFIDENTIAL

Marketing Code	Description	List Price	Disc Sch	Full Extended Warranty (Annual)	Basic Extended Warranty (Annual)	Parts Care Year 1 (Annual)	On-Site Parts Care Year 1 (Annual)	IRIXCare sm (Monthly)	Full Support (Monthly)	Basic Support (Monthly)	Parts Care Year 2+ (Monthly)	On-Site Parts Care Year 2+ (Monthly)
TPC Benchmark Full Disclosure Reports												
M4-B001-1.0	TPC Benchmark B Full Disclosure Report for Silicon Graphics CHALLENGE XL Server and ORACLE7	25	0									
STANDARD SYSTEM												
M4-W4D-5.1.1	IRIS Workstation Owner's Manuals (for IRIX 5.1.1, Non-Indlgo Magic Users)	175	0									
M4-W4D-4.0.5	IRIS Workstation Owner's Manuals (for IRIX 4.0.5)	175	0									
M4-W4D-4.0.1	IRIS Workstation Owner's Manuals (for IRIX 4.0.1)	175	0									
M4-S4D-5.1.1	Server Owner's Manuals (for IRIX 5.1.1)	550	0									
M4-S4D-4.0.5	Server Owner's Manuals (for IRIX 4.0.5H)	450	0									
M4-S4D-4.0.1	Server Owner's Manuals (for IRIX 4.0.x)	450	0									
M4-TRIX-4.0.5IPR	Trusted IRIX/B 4.0.5IPR Manuals	250	0									
M4-TRIX-4.0.4	Trusted IRIX Manuals (for IRIX 4.0.4)	250	0									
M4-TRIX-4.0.1	Trusted IRIX Manuals (for IRIX 4.0.1)	250	0									
DEVELOPMENT OPTION												
M4-ADMN-5.1.1	IRIX Admin, IRIX Admin Man Pages, NetLS Admin, NFS Admin, NIS Admin, and Diskless Admin Guides	500	0									
M4-ADMN-4.0.1	IRIX Advanced Site and Server Administration Guide and IRIX System Administration Man Pages	300	0	180	180	N/A	N/A	240	20	20	N/A	N/A
M4-IDO-5.1	IRIX Development Manuals-IRIX 5.1.1 (contains hard copy of the online manuals shipped standard with IDO)	950	0									
M4-IDO-4.1.1	IRIX Development Kit (for IRIX 4.0.x, 3.10.1 Compiler)	800	0									
M4-UDS1-5.1	IRIX Programming Manuals (for IRIX 5.1.1)	125	0									
M4-UDS1-4.0.1	IRIS-4D Series Operating System Manuals	125	0									
M4-ANSIC-5.1	ANSI C Programming Manuals (for IRIX 5.1.1)	100	0									
M4-ANSIC-2.0.1	ANSI C Programming Manuals (for IRIX 4.0.x)	100	0	63	63	N/A	N/A	84	7	7	N/A	N/A
M4-OGLPG-1.0	OpenGL Programming Guide	49	0									
M4-OGLPORT-5.1	OpenGL Porting Guide	50	0									
M4-OGLMAN-1.0	OpenGL Reference Manual	46	0									
M4-XDEV-3.1	X11 Development Manuals (for IRIX 5.1.1)	185	0									
M4-X113D-2.0	X11 Development Manual (for IRIX 4.0.x)	185	0									
M4-MOTIF-1.2.2	Motif Developers Manuals (for IRIX 5.1.1)	300	0									
M4-MOTIF-2.0.1	Motif Developers Manuals (for IRIX 4.0.x)	300	0									

US Pricing

MAN-1

February 1, 1994



Manuals

CONFIDENTIAL

Marketing Code	Description	List Price	Disc Sch	Full Extended Warranty (Annual)	Basic Extended Warranty (Annual)	Parts Care Year 1 (Annual)	On-Site Parts Care Year 1 (Annual)	IRIXCare SM (Monthly)	Full Support (Monthly)	Basic Support (Monthly)	Parts Care Year 2+ (Monthly)	On-Site Parts Care Year 2+ (Monthly)
DEVELOPMENT OPTION												
M4-DESK-5.1	Desktop Integration Manuals (for IRIX 5.1.1)	100	0									
M4-GLGT-5.1	Graphics Library Manuals (for IRIX 5.1.1)	125	0									
M4-GLGT-4.1	Graphics Library Manuals (for IRIX 4.0.x)	125	0									
M4-GLGT-4.0.1	Graphics Library Manuals (for IRIX 4.0.x, 2.20 Compilers)	125	0									
M4-WSINTEG-1.0	IRIS WorkSpace Integration Guide (for IRIX 4.0.x)	50	0	27	27	N/A	N/A	36	3	3	N/A	N/A
M4-DPSDEV-2.0.1	PostScript Manuals (for IRIX 5.1.1)	185	0									
M4-DPS-1.0.1	PostScript Manual (for IRIX 4.0.x)	185	0									
INDIVIDUAL MANUALS FROM STANDARD AND DEV SYSTEMS												
M4-DVDR-3.0	IRIX Device Driver Programming Guide and IRIX Device Driver Reference Pages (for IRIX 5.1.1)	200	0									
M4-DVDR-2.0	IRIX Device Driver Programming Guide (for IRIX 4.0.x)	100	0									
M4-IRXMP-4.0	IRIX Man Pages (for IRIX 4.0.x)	350	0									
M4-GLC-2.0	Graphics Library C Man Pages (for IRIX 4.0.x)	100	0									
M4-GLFTN-3.4	Graphics Library Fortran Man Pages (for IRIX 4.0.x)	100	0									
M4-GLPAS-1.2	Graphics Library Pascal Man Pages (for IRIX 4.0.x)	100	0									
M4-GLADA-4.1	Graphics Library Ada Man Pages (for IRIX 4.0.x)	100	0									
DIGITAL MEDIA DEVELOPMENT LIBRARIES												
M4-DMDEV-1.2	Digital Media Programming Guide	250	0									
M4-DMMAN-1.1	Digital Media Development Reference Pages (for SC4-DMDEV-1.1 only)	125	0									
IMAGEVISION LIBRARY												
M4-IL-2.2	ImageVision Library Programming Guide (for IRIX 5.1)	195	0									
M4-IL/C++-2.2	ImageVision Library C++ Reference Manual	70	0									
M4-IL/C-2.2	ImageVision Library C Reference Manual	70	0									
M4-IL/FORTRAN-2.2	ImageVision Library Fortran Reference Manual	70	0									
M4-IL-2.0	ImageVision Library Programming Guide (for IRIX 4.0.X)	155	0									
M4-IL/C++-2.0	ImageVision Library C++ Reference Manual	70	0									
M4-IL/C-2.0	ImageVision Library C Reference Manual	70	0									
M4-IL/FORTRAN-2.0	ImageVision Library Fortran Reference Manual	70	0									
IRIS EXPLORER												
M4-EXPLR-2.0	IRIS Explorer Manual Set (for IRIX 4.0.x and 5.1.1)	195	0									

Manuals

CONFIDENTIAL

Marketing Code	Description	List Price	Disc Sch	Full Extended Warranty (Annual)	Basic Extended Warranty (Annual)	Parts Care Year 1 (Annual)	On-Site Parts Care Year 1 (Annual)	IRIXCare sm (Monthly)	Full Support (Monthly)	Basic Support (Monthly)	Parts Care Year 2+ (Monthly)	On-Site Parts Care Year 2+ (Monthly)
IRIS INVENTOR												
M4-INVENTOR-1.1.2	IRIS Inventor Programming Guide (for IRIX 5.0.1 and 5.1.1)	195	0									
M4-INVENTOR-1.0.1	IRIS Inventor Programming Guide (for IRIX 4.0.x)	195	0									
M4-INVC++-1.0	IRIS Inventor C++ Reference Manual	100	0									
M4-INVC-1.0	IRIS Inventor C Reference Manual	180	0									
IRIS PERFORMER												
M4-PERF-1.1	IRIS Performer Manuals (for IRIX 5.X)	195	0									
M4-PERF-1.0	IRIS Performer Manuals (for IRIX 4.0.x)	195	0									
WINDOW SYSTEMS												
M4-MSG-1.0	OSF/Motif Style Guide (for IRIX 4.0.x)	40	0									
M4-MPGR-1.0	OSF/Motif Programmer's Reference Manual (for IRIX 4.0.x)	95	0									
M4-MPGD-1.0	OSF/Motif Programmer's Guide (for IRIX 4.0.x)	75	0									
CASEVISION a la carte												
M4-CM-1.1.4	CASEVision/ClearCase 1.1.4 Manuals (for IRIX 4.0.x)	150	0									
M4-BT-2.0	CASEVision/Tracker Design Guide	85	0									
M4-BT-1.0.1	CASEVision/Tracker Manuals	125	0									
M4-CMBT-1.1	CASEVision/ClearCase and Run Time CASEVision/Tracker Manuals	200	0									
M4-TOOLTALK-1.2	ToolTalk Administration Guide (for IRIX 5.1.1)	240	0									
M4-TOOLTALK-1.0	ToolTalk Administration Guide (for IRIX 4.0.x)	240	0									
M4-TTDEV-1.2	ToolTalk Programming Guide (for IRIX 5.1)	240	0									
M4-TTDEV-1.0	ToolTalk Programming Guide (for IRIX 4.0.x)	240	0									
M4-WS-2.1	CASEVision/WorkShop Manuals	250	0									
M4-WS-1.1.1	CASEVision/WorkShop Manuals	250	0									
M4-PROMPF-1.1	CASEVision/WorkShop Pro MPF Manuals	175	0									
COMPILERS												
M4-FTN-4.0+	Fortran 77 Manuals (for IRIX 5.1.1)	250	0									
M4-FTN-3.5.1	Fortran 77 Manuals (for IRIX 4.0.x, 3.10.1 Compiler)	250	0									
M4-FTN-3.4.1	Fortran 77 Manuals (for IRIX 4.0.x, 2.20 Compiler)	250	0									
M4-PFTN-4.0	Power Fortran Manual (for IRIX 5.1.1)	225	0									

US Pricing

MAN-3

February 1, 1994



Manuals

CONFIDENTIAL

Marketing Code	Description	List Price	Disc Sch	Full Extended Warranty (Annual)	Basic Extended Warranty (Annual)	Parts Care Year 1 (Annual)	On-Site Parts Care Year 1 (Annual)	IRIXCare sm (Monthly)	Full Support (Monthly)	Basic Support (Monthly)	Parts Care Year 2+ (Monthly)	On-Site Parts Care Year 2+ (Monthly)
COMPILERS												
M4-PFTN-3.1.1	Power Fortran Manuals (for IRIX 4.0.x, 3.10.1 Compiler)	225	0									
M4-PFTN-3.0	Power Fortran Manual (for IRIX 4.0.x, 2.20 Compiler)	225	0									
M4-C++-3.2	C++ 3.2 Translator Manuals (for IRIX 5.1)	250 ²⁰⁸	0									
M4-C++-3.0.1	C++ Translator Manuals (for IRIX 4.0.x, 3.10.1 Compiler)	250	0									
M4-C++-2.1.1	C++ Translator Manuals (for IRIX 4.0.x, 2.20 Compiler)	250	0									
M4-PWRC-2.4	IRIS Power C Manual (for IRIX 5.1)	225	0									
M4-PWRC-2.1.1	IRIS Power C Manual (for IRIX 4.0.x, 3.10.1 Compiler)	225	0									
M4-PWRC-2.0	IRIS Power C Manual (for IRIX 4.0.x, 2.20 Compiler)	225	0									
M4-PAS-1.4.2	Pascal Manual (for IRIX 5.1.1)	250	0									
M4-PAS-1.3.1	Pascal Manual (for IRIX 4.0.x, 3.10.1 Compiler)	250	0									
M4-PAS-1.2.1	Pascal Manual (for IRIX 4.0.x, 2.20 Compiler)	450	0									
M4-AXM-6.2.1	Verdix AXM 6.2.1 Manuals (for IRIX 5.0.1)	450	0									
M4-MPXM-6.2.1	Verdix MPXM 6.2.1 Manuals (for IRIX 5.0.1)	450	0									
NETWORKING												
M4-NFS-4.0.1	NFS Manuals (for IRIX 4.0.x). For NFS 5.1 manuals, see M4-ADMN-5.1.1.	90	0									
M4-4DDN-3.2	4DDN Manuals (for IRIX 5.1.1)	250	0									
M4-4DDN-3.0	4DDN Manuals (for IRIX 4.0.x)	250	0									
M4-4DLT-1.1	4DLT Manuals	150	0									
M4-FDDIXP-3.0.4	FDDIXPress Manuals	400	0									
M4-FDDIXP-3.0.1	FDDIXPress Manuals	400	0									
M4-4DTR-2.0	IRIS Token Ring Manual	150	0									
M4-NDK/SGI-1.2	NetLS Development Optlon (for IRIX 5.1.1)	100	0									
M4-NDK/SGI-1.0	NetLS Development Optlon (for IRIX 4.0.x)	100	0									
M4-NETV-2.0	NetVisualyzer Manuals	100	0									
M4-NETWKR-1.2	IRIS NetWorker Manuals	90	0									
M4-DVM-2.0	IRIS Volume Manager Manuals (for IRIX 5.X)	90	0									
M4-DVM-1.0	IRIS Volume Manager Manuals (for IRIX 4.0.X)	90	0									
M4-SNA-3.1	SGI SNA Server Manuals	150	0									
M4-HC327-2.1	High Performance 3270 Host File Transfer Software Manual	100	0									
M4-I3270-5.0.1	IRIS SNA 3270 Manuals (for IRIX 4.0.5x)	150	0									

Manuals

CONFIDENTIAL

Marketing Code	Description	List Price	Disc Sch	Full Extended Warranty (Annual)	Basic Extended Warranty (Annual)	Parts Care Year 1 (Annual)	On-Site Parts Care Year 1 (Annual)	IRIXCare SM (Monthly)	Full Support (Monthly)	Basic Support (Monthly)	Parts Care Year 2+ (Monthly)	On-Site Parts Care Year 2+ (Monthly)
NETWORKING												
M4-I3270-5.0	IRIS SNA 3270 Manuals - 4.0.5x (for 3270+5080 product)	150	0									
M4-WS62-2.0	IRIS SNA LU 6.2 Manual	150	0									
M4-WS377-1.2.1	IRIS SNA 3770 Manual	75	0									
M4-ITR-2.0	IRIS Token Ring for IRIS Indigo Manuals	150	0									
M4-508S-3.0	IRIS 5080 Terminal Emulator Manuals (for IRIX 4.0.5x)	250	0									
M4-508G-3.0	IRIS 5080 Gateway Manual (for IRIX 4.0.5x)	200	0									
M4-ICA-1.0	IRIS Channel Adapter Administration Guide	100	0									
M4-HFTS-VM-1.0	FastFile: Host 5080/3270 High-Speed File Transfer for IBM VM OS Manuals	250	0									
M4-HFTS-MVS-1.1	FastFile: Host 5080/3270 High-Speed File Transfer for IBM MVS OS Manuals	250	0									
VIDEO OPTIONS												
M4-VL-1.2	VideoLab Manuals	100	0									
M4-BV-1.2	Broadcast Video Manuals	55	0									
M4-LVD-1.2	Live Video Digitizer Manuals	100	0									
M4-VC-1.2	VideoCreator Manuals	55	0									
M4-VF-2.0.1	VideoFramer Manuals	65	0									
M4-VS-1.2	VideoSplitter Manuals	280	0									
APPLICATIONS/DOCUMENTATION TOOLS												
M4-DWB-4.1.1	Documenter's Workbench Manuals (for IRIX 5.1.1)	295	0									
M4-DWB-4.0.1	Documenter's Workbench Manuals (for IRIX 4.0.x)	295	0									
M4-EMCS-3.4.1	EMACS Manuals (for IRIX 4.0.x)	225	0									
M4-IMP-1.1	Impressario Manuals (for IRIX 4.0.5x and later)	150	0									
M4-IMPD-1.1	Impressario Developers Manuals (for IRIX 4.0.5x and later)	250	0									
M4-LPS-4.0.1	Laser Printer Support Manuals	50	0									
SOFTWARE EXPRESS MANUALS												
EXPRESS	Refer to Software Express Section	N/A	0									

US Pricing

MAN-5

February 1, 1994



Manuals

CONFIDENTIAL

Marketing Code	Description	List Price	Disc Sch	Full Extended Warranty (Annual)	Basic Extended Warranty (Annual)	Parts Care Year 1 (Annual)	On-Site Parts Care Year 1 (Annual)	IRIXCare SM (Monthly)	Full Support (Monthly)	Basic Support (Monthly)	Parts Care Year 2+ (Monthly)	On-Site Parts Care Year 2+ (Monthly)
----------------	-------------	------------	----------	---------------------------------	----------------------------------	----------------------------	------------------------------------	----------------------------------	------------------------	-------------------------	------------------------------	--------------------------------------

[208] This product will operate only in IRIX 5.x Operating Systems. It will not operate on IRIX 4.0.x systems.



Engineer's Handbook General Stuff

May 5, 1994

Using X

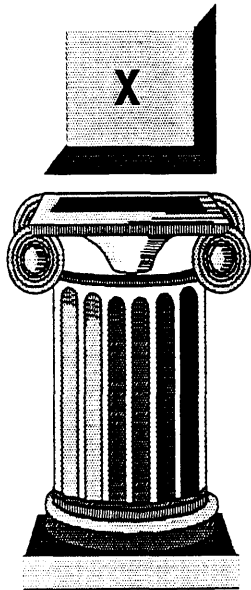
probable data location: `clubted.csd:/usr/graphics/develop/motif/book/x1.ps`
access via: `handbook using_x`



SiliconGraphics
Computer Systems

X Window System Using X

Student Workbook



Publication Number: T41W0 IN 001
September 3, 1991

RESTRICTION ON USE

This document is protected by copyright and contains information proprietary to Silicon Graphics, Inc. Any copying, adaptation, distribution, public performance, or public display of this document without the express written consent of Silicon Graphics, Inc., is strictly prohibited. The receipt or possession of this document does not convey the rights to reproduce or distribute its contents, or to manufacture, use, or sell anything that it may describe, in whole or in part, without the specific written consent of Silicon Graphics, Inc.

Copyright © 1991 Silicon Graphics, Inc. All rights reserved.

U.S. GOVERNMENT RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure of the data and information contained in this document by the Government is subject to restrictions as set forth in FAR 52.227-19(c)(2) or subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and/or in similar or successor clauses in the FAR, or the DOD or NASA FAR Supplement. Unpublished rights reserved under the Copyright Laws of the United States. Contractor/manufacturer is Silicon Graphics, Inc., 2011 N. Shoreline Blvd., P.O. Box 7311, Mountain View, CA 94039-7311.

The contents of this publication are subject to change without notice.

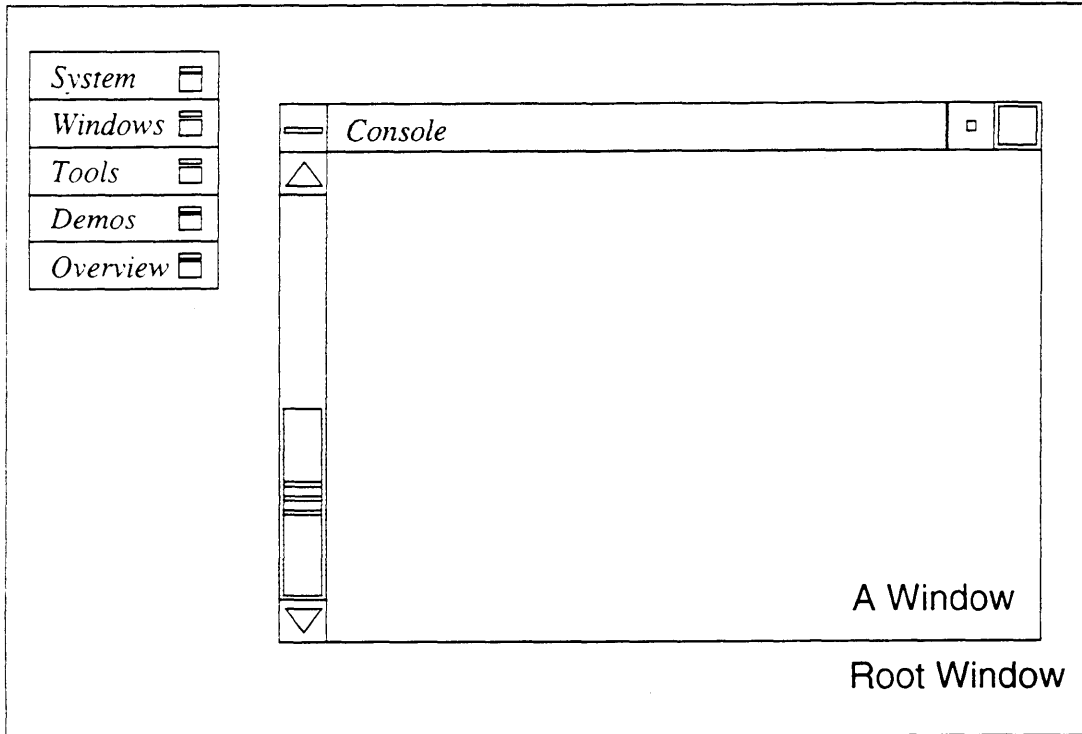
Objectives

- Use the mouse and keyboard to control the `dwm` window manager to open, position, resize, iconify, and close windows.
- Identify and run several standard X clients, such as `xclock`, `xcalendar`, and `xcalc`, to do ordinary activities.
- Define common X Window System terminology, such as "server", "client", and "display."

Characteristics of X

- Available from MIT
- On many different computing platforms
- Device-independent layer
- Supports multiple displays and screens
- Supports multiple input devices
- Look and feel completely reconfigurable
- Applications can be run remotely

4Dwm Window Manager



<i>4D Window Manager</i>
<i>Log out</i>

Root Window Menu

<i>Restore</i>	<i>ALT+F5</i>
<i>Move</i>	<i>ALT+F7</i>
<i>Size</i>	<i>ALT+F8</i>
<i>Minimize</i>	<i>ALT+F9</i>
<i>Maximize</i>	<i>ALT+F10</i>
<i>Raise</i>	<i>ALT+F1</i>
<i>Lower</i>	<i>ALT+F3</i>
<i>Close</i>	
<i>Quit</i>	

Window Menu

4Dwm Window Manager

- Pointer button actions
 - The mouse is the pointer device

Table 4: 4Dwm Mouse Actions

<i>button</i>	<i>context</i>	<i>action</i>
LEFT	root window (background)	root window menu
LEFT	frame (border)	pop (raise) window and move or resize
LEFT	icon	open icon
MIDDLE	frame or icon	move window or icon
RIGHT	root window (background)	root window menu
RIGHT	frame or icon	window menu

- Using mnemonics
 - Alt/Space Bar key pair invokes pull-down window menu
 - also Shift/ESC key pair similarly invokes window menu
 - press mnemonic letter, while menu appears

Log Out

- also to move borderless windows (e.g., toolchest)

Pointer Vocabulary

Pointing

- sliding the mouse to position the pointer on an object on the screen

Pressing

- holding down a mouse button
- buttons are numbered 1-3, from left to right

Clicking

- pressing and releasing a mouse button without moving the pointer

Double-clicking

- clicking a mouse button twice quickly
- the maximum time (in milliseconds) between clicks can be set by the user

Dragging

- pressing and holding down a mouse button while moving the pointer

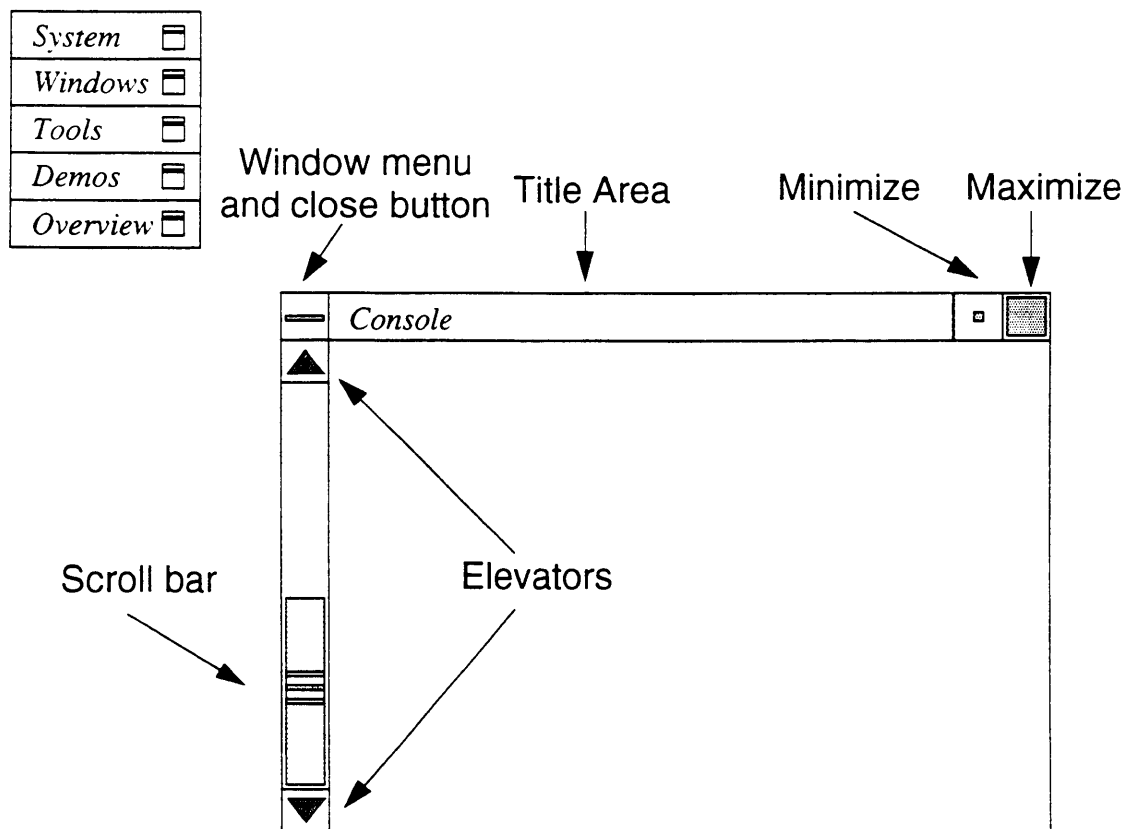
Using 4Dwm Exercise

Purpose:

This exercise will give you practice in using the 4Dwm window manager.

Instructions:

1. Log into the system as the user *guest*. The 4Dwm window manager is invoked. The toolchest menu appears in the upper left. A terminal emulator also appears. If the terminal initially appears as an icon, use the LEFT mouse button to open the window.



2. Move the cursor in and out of the terminal emulator window. Note when the cursor is within the window, the window receives input focus. When input focus is received, you can type into the window and use the mouse in the program. This is shown by the window frame being highlighted.
3. Try using the mouse buttons.

To use the scroll bar and elevator buttons down the left side of the terminal emulator window, use the LEFT mouse button. To see the pull-down menu (the Window menu), press the LEFT mouse button on the button at the far left of the title bar. Double-clicking (pressing twice quickly) on this button will close the window.

To minimize (typically, iconify) and maximize (usually, to the full screen) a window, press the LEFT mouse button over the two buttons to the far right of the title bar. To return from a full-sized window back to regular size, press the LEFT mouse button over the Maximize icon an additional time.

Clicking the LEFT mouse button over an icon will open the icon.

Over the root (main background) window, the LEFT mouse controls a pop-up menu which is used to quit the window manager. This menu is the Root Window menu. The RIGHT mouse button similarly controls the Root Window menu.

To move an entire window or icon, move the cursor over the border or title bar of a window, press the MIDDLE mouse button and drag the mouse. Another way to move a window is, while the cursor is over the frame (border) of a window, to press the LEFT mouse button and drag the mouse. To resize a window, press the LEFT mouse near any of the "handles" on the frame of the window, and drag the mouse to the new location.

The RIGHT mouse button controls the pop-up menus. If the cursor is outside all the windows, the root (background) pop-up menu is displayed. If the cursor is over the title bar of a window, the Window menu is displayed (the same menu, as if the LEFT mouse button is pressed on the leftmost part of the title bar). If the cursor is over the body of a window, a client-specific pop-up menu may appear.

You may also use the menus to close, raise, lower, resize and move windows.

4. 4Dwm adheres to the **ICCCM** (Inter-Client Communication Conventions Manual). That means that different programs (clients) work with each other and can communicate. The standard is that data can be **selected** (with the LEFT mouse button) and stored in a buffer. This buffer can be **pasted** (with the MIDDLE mouse button) into another program.

For example, within the terminal emulator window, to select and highlight text, press the LEFT mouse button and drag the mouse over a region of text. The highlighted text is automatically copied into a text buffer. Using the MIDDLE mouse button, the highlighted text can be pasted to many other windows.

5. The toolchest in the upper left corner of the window is activated by pressing the

LEFT or RIGHT mouse button. Go through the pull-down and rollover menus to start some clients (applications). For example with the Tools menu, start the xcalc (calculator) and create another terminal emulator shell window. Press the LEFT or RIGHT mouse button over the "Overview" toolchest entry and choose entries from the pull-right menu. These Showcase slides can guide you through an introductory tour.

Note there are keyboard shortcuts for all the window menu actions. The shortcuts are activated by either pressing the Alt/Space Bar keys or the Shift/Escape keys. Press either key pair to cause the menu to appear. Now, while the menu appears, if you press the underlined letter (the **mnemonic**), that menu entry will be chosen.

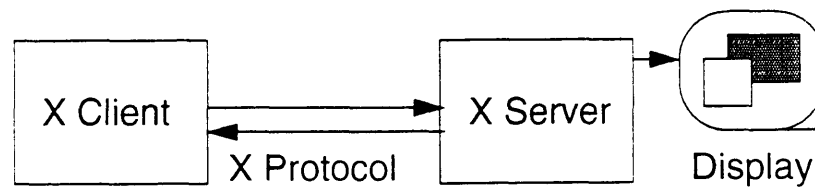
Also, several keys have ALT+Fkey **accelerators**. These shortcuts do not require first pressing the Alt/Space or Shift/Escape key pair to view the menu. They allow you to execute some menu entries without using the mouse.

6. Enter a UNIX shell window and run these X clients. Note that X clients, unlike IRIS Graphics Library programs, run in the foreground. Also note that the LEFT mouse button is used to position the windows. (You can use the man pages in O'Reilly, Volume 3, as a reference for what each of these clients do.) You can remove this list for later reference.

```
% xterm &--terminal emulator
% xclock &
% xclock -digital &
% xbiff &--mail notifier
% xload &--system usage monitor
% xman &--manual page browser
% xedit filename &--text editor
% xcalendar &--datebook
% xcalc &--calculator
% xfd -fn 9x15 &--font displayer
% xfontsel &--interactive font displayer
% bitmap filename &--bitmap editor
% xdpyinfo--list display and window information
```

```
% xlsclients--list active client information
% xlswins--list window information
% xlsfonts--list available fonts
% xwd -out file--store window image in a dump file (not the SGI .rgb format)
% xwd -in file--display window image from xwd
% xpr file -device ps -output outfile--convert xwd file to PostScript outfile
```

Server/Client



X Window System

- the entire X architecture
- not "kernel-based," part of the operating system
- client-server model
- client and server are separate processes

Server

- controls hardware display and input devices
- receives and responds to (multiple) client messages
- runs asynchronously with respect to clients

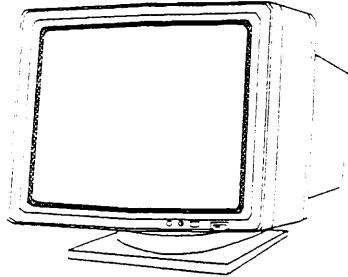
Client

- an application which uses services provided by X server
- can run distributed across a network
- can connect to one or more servers

Protocol

- the (collective) messages between servers and clients

Display/Screen



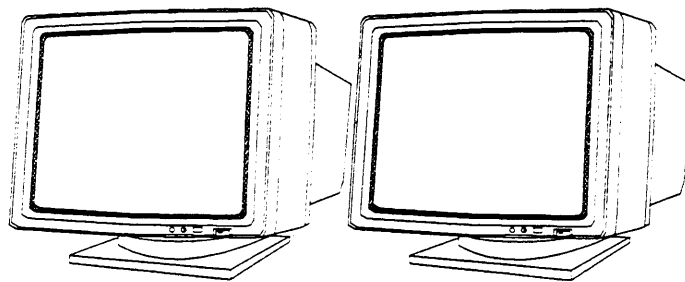
Typical case:
Display is entire screen

Display

- controlled by display server
- one set of input devices (e.g., mouse & keyboard) for each display
- typically, the display server controls one screen, BUT a display server CAN control multiple screens

Screen

- a physical monitor



Also possible:
Display is several screens

Display/Screen

Display

- first display is display 0
- additional displays available (such as X-terminals)

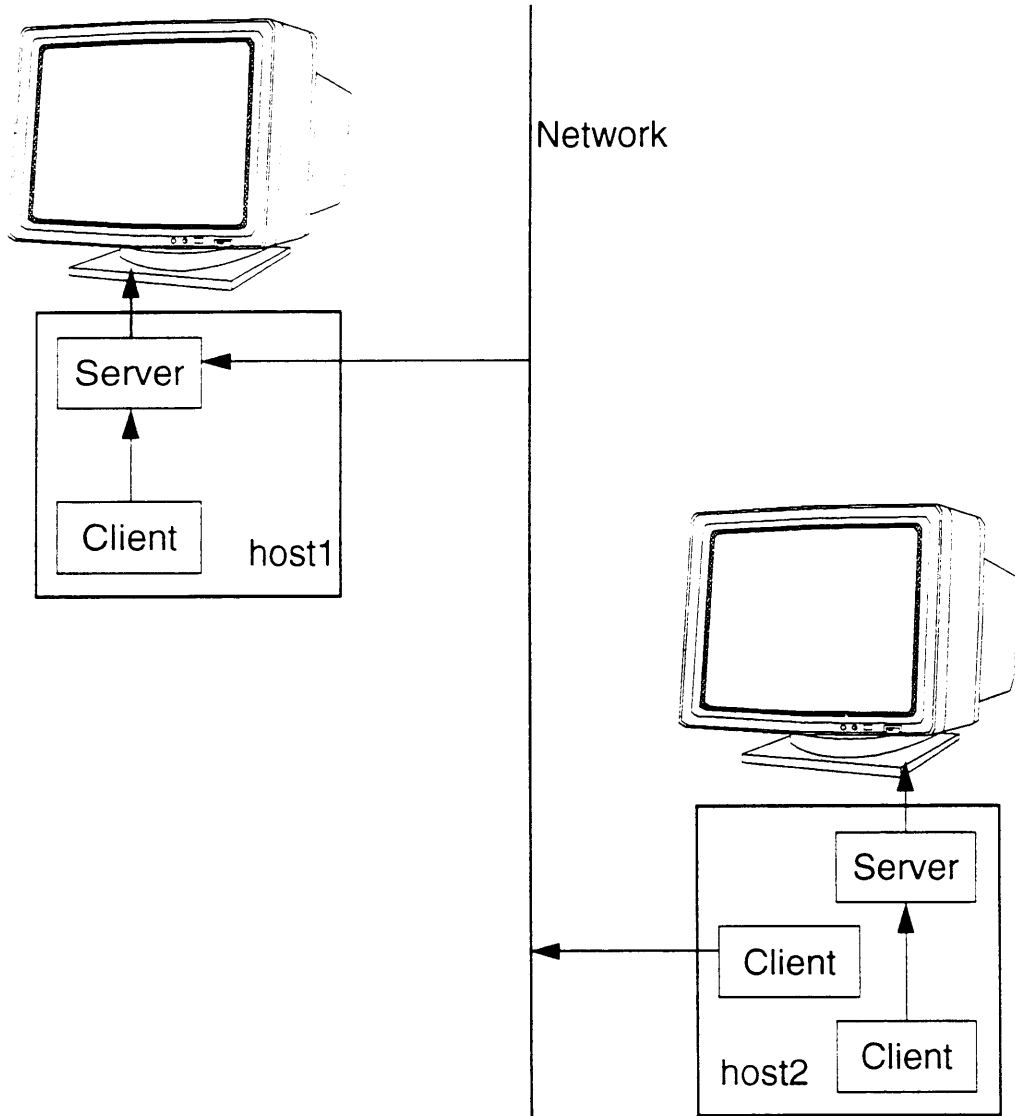
Screen

- first screen is screen 0
- additional screens available (multi-head option)

Display/Screen syntax

- `[host]:display.screen`
- `local:0.0` is typical
 - first screen on first display on local host

Distributed System



Distributed System

- `xhost`
- maintain (add & delete) list of machines which can connect to the X server
- command line options

```
xhost
```

query--what hosts can access this machine

```
xhost +hostname or xhost hostname
```

hostname is added to list allowed to connect with X server

```
xhost -hostname
```

hostname is removed from list of machines

```
xhost +
```

access granted to everyone

```
xhost -
```

access restricted to current list of machines

- run an Xclient on one server to a remote display

```
xcalc -display host1:0.0 &
```

```
xload -display host1:0.0 &
```

```
or setenv DISPLAY host1:0.0
```

Window Management

- **Window manager**
 - an ordinary X Client
 - supervises all windows
 - several different window managers
 - 4Dwm--SGL window manager
 - mwm--OSF/Motif window manager
 - twm--Tab (formerly Tom's) window manager
 - found in `/usr/bin/X11`
- **Window manager adds "decorations"**
 - such as title bar, menus, scroll bars, window buttons, etc.
- **Window manager traps window requests**
 - primarily open, resize, move, and close requests
 - window manager may override or alter request
- **"Mechanism, Not Policy"**
 - look and feel may be reconfigured

Client Execution Exercise

Purpose:

This lab will give you practice in using X applications across the network. Also, it will introduce you to different window managers.

Instructions:

1. Log into the system as the user *guest*. The 4Dwm window manager is invoked.
2. Run a client across the network to another display.

First, be friendly and use `xhost` to ensure other machines to access your machine.

Then use the `-display` option with an X client. For example, type:

```
xcalc -display host1:0.0 &
```

3. (This exercise may not be available, especially if following these notes as a self-paced student. Skip to #4, if no such directory exists.)

`cd` to the `final` directory.

Run the program `finalsurvey`

This is a sample Motif program. All X Toolkit programs (and that includes all Motif programs) support the `-display` option. Use the `-display` option to execute the program on a server across the network.

4. * Quit your current window manager. You can do this from the Toolchest menu.

Now run a different window manager, such as `twm` or `mwm`. If you cannot get out of the window manager, ask the instructor for assistance.

Glossary

client	an X application program which makes requests (for information, processes) to the server
display	one or more monitors which share a common keyboard and pointer (mouse, tablet, etc.) driven by a single X server
icon	a symbol which represents a window in an inactive state
input focus	the window to which keyboard and mouse input are directed
"mechanism, not policy"	the Window System protocol (mechanism) is standard on all X implementations, but the higher-level operations (such as Look & Feel) are not specified and imposed.
pointer	a symbol which tracks device (mouse) movement.
protocol	set of messages communicated between clients and servers
screen	a physical monitor
server	the machine (CPU) which distributes user input to and accepts output requests from various client programs. It is an intermediary between client programs and local display hardware.
terminal emulator	window which functions as a standard terminal
window	a rectangular region on the screen controlled as a single object
window manager	A client program which controls "look and feel" of the X Window System. It allows the user to change size and position of windows on the display.

Bibliography

For End-users

Mansfield, Niall, *The X Window System: A User's Guide*, Addison-Wesley, Amsterdam, 1990. ISBN 0-201-51341-2. A tutorial on X for end-users, not programmers. The first edition of this book, based on X11R3, is available only in Europe. The second edition, based on X11R4, is available in both Europe and the U.S.

O'Reilly and Associates, *The X Window System in a Nutshell*, O'Reilly and Associates, 1990. ISBN 0-937175-24-2. A quick reference guide to Xlib and Xt and some associated configuration files.

O'Reilly and Associates, *The X Window System Series (Volume 3)*, O'Reilly and Associates, 1988, 1989, 1990. This is a 7 (and growing) volume set of books. Volume 3 is the end-users reference manual. It is an edited version of the MIT manuals (popular client manual pages), including additional tutorial and introductory chapters.

Open Software Foundation, *OSF/Motif Series (5 volumes)*, Prentice Hall, 1990. ISBN 0-13-640491-X, 0-13-640525-8, 0-13-640517-7, 0-13-640509-6, 0-13-640483-9. The volumes include *Motif Style Guide*, *Programmer's Guide*, *Programmer's Reference*, *User's Guide*, and *Application Environment Specification (AES) User Environment Volume*. For many end-users, the *User's Guide* is enough. For sophisticated end-users, the *Programmer's Guide* is also necessary.

For Hackers Only

Scheifler, Robert and Jim Gettys, "The X Window System," *ACM Transactions on Graphics*, vol. 5, no. 2, pp. 79-109, April, 1986. The first published description of X. Although it discusses X10, it is still one of the most comprehensive overviews of X. This paper is being updated to X11 by Jim Gettys, Phil Karlton, and Scott McGregor.

Summary

This chapter summary is excerpted from the manual page for X(1).

This material is Copyright 1984, 1985, 1986, 1987, 1988, and 1989, by the Massachusetts Institute of Technology.

DESCRIPTION

X Window System servers run on computers with bitmap displays. The server distributes user input to and accepts output requests from various client programs through a variety of different interprocess communication channels. Although the most common case is for the client programs to be running on the same machine as the server, clients can be run transparently from other machines (including machines with different architectures and operating systems) as well. X supports overlapping hierarchical subwindows and text and graphics operations, on both monochrome and color displays.

The number of programs that use X is growing rapidly. Of particular interest are: a terminal emulator (xterm), a window manager (4Dwm), a display manager (xdm), mail managing utilities (xmh and xbiff), a manual page browser (xman), a bitmap editor (bitmap), access control programs (xauth and xhost), user preference setting programs (xrdb, xset, xsetroot, and xmodmap), a load monitor (xload), clocks (xclock and o'clock), a font displayer (xfont), utilities for listing information about fonts, windows, and displays (xlsfonts, xfontsel, xlswins, xwininfo, xlsclients, xdpwininfo, and xprop), a diagnostic for seeing what events are generated and when (xev), screen image manipulation utilities (xwd, xwud, xpr, and xmag), and various demos (xeyes, ico, muncher, puzzle, xgc, etc.).

DISPLAY NAMES

From the user's prospective, every X server has a display name of the form:

hostname:displaynumber.screennumber

This information is used by the application to determine how it should connect to the server and which screen it should use by default (on displays with multiple monitors):

hostname

The hostname specifies the name of the machine to which the display is physically connected. If the hostname is not given, the most efficient way of communicating to a server on the same machine will be used.

displaynumber

The phrase "display" is usually used to refer to collection of monitors that share a common keyboard and pointer (mouse, tablet, etc.). Most workstations tend to only have one keyboard, and therefore, only one display. Larger, multi-user systems, however, will frequently have several displays so that more than one person can be doing graphics work at once. To avoid confusion, each display on a machine is assigned a display

number (beginning at 0) when the X server for that display is started. The display number must always be given in a display name.

screennumber

Some displays share a single keyboard and pointer among two or more monitors. Since each monitor has its own set of windows, each screen is assigned a screen number (beginning at 0) when the X server for that display is started. If the screen number is not given, then screen 0 will be used.

Since there can be more than one way of contacting a given server, The hostname part of the display name is used to determine the type of channel (also called a transport layer) to be used. The sample servers from MIT support the following types of connections:

local

The hostname part of the display name should be the empty string. For example: :0, :1, and :0.1. The most efficient local transport will be chosen.

TCP/IP

The hostname part of the display name should be the server machine's IP address name. Full Internet names, abbreviated names, and IP addresses are all allowed. For example: expo.lcs.mit.edu:0, expo:0, 18.30.0.212:0, bigmachine:1, and hydra:0.1.

WINDOW MANAGERS

The layout of windows on the screen is controlled by special programs called window managers. Although many window managers will honor geometry specifications as given, others may choose to ignore them (requiring the user to explicitly draw the window's region on the screen with the pointer, for example).

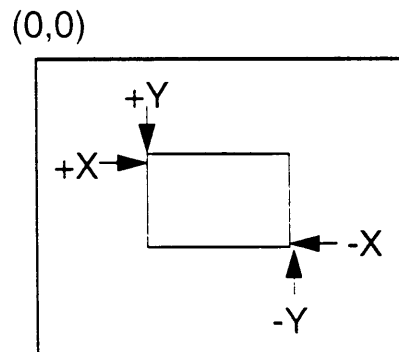
Since window managers are regular (albeit complex) client programs, a variety of different user interfaces can be built. The SGI distribution comes with a window manager named 4Dwm which supports overlapping windows, popup menus, title bars, and nice icons.

Resource Management

- **XAPPLRESDIR** or **XUSERFILESEARCHPATH** environment variable sets the directory (or directory path) to find customized, application resource files
- **XENVIRONMENT** sets a specific resource file
- **RESOURCE_MANAGER** property database
 - determines default values for resources
 - `/usr/bin/X11/xrdb` command line options
 - query
 - merge file (add file resource values to the list)
 - load file (remove old resources; establish new values)
 - remove
 - in the `Xsession` file, the `$HOME/.Xresources` file is loaded by `xrdb`, directly into the server
- `$HOME/.Xdefaults` file
 - if there is no **RESOURCE_MANAGER** property
- Most X clients have default resource definition files in the `/usr/lib/X11/app-defaults` directory
 - For example:
 - `4Dwm uses /usr/lib/X11/app-defaults/4Dwm`
 - `xterm uses /usr/lib/X11/app-defaults/XTerm`

Positioning Windows

- format for `-geometry` option
- `widthxheight+-xoff+-yoff`
 - to many clients, `width`, `height`, `xoff`, `yoff` are in pixels
 - for `xterm` and `xwsh`, `width` *and* `height` values are columns/rows of text
- `widthxheight` is size of the client
- `+xoff+yoff` is distance from the left and top screen edges
- `-xoff-yoff` is distance from the right and bottom edges



Command Line Specification

- Overrides all other resource specifications
- Standard options (recognized by all X Toolkit applications)

`-background color OR -bg color`

`-foreground color OR -fg color`

`-bordercolor color OR -bd color`

`-font name OR -fn name (fontList resource is often more useful)`

`-name string`

`-title string`

`-geometry geometryspec`

use quoted string as a resource specification

`-xrm "resource:value"`

start application as an icon

`-iconic`

run program on specified display

`-display [host]:display.screen`

reverse foreground and background (not use for color IRIS)

`-rv/-reverse/+rv`

`-borderwidth number OR -bw number`

Resource Customization Exercise

1. Application Customization

Modify a `$HOME/.Xdefaults` file to define the location, font, and color of several `xclients` (such as `xbiff` and `xcalc`).

* If a `RESOURCE_MANAGER` property exists, the `$HOME/.Xdefaults` file will be ignored. How do you include the resources from the `$HOME/.Xdefaults` file into the `RESOURCE_MANAGER` property?

2. The `finalsurvey` program is a Motif application, so it supports all X Toolkit command line options. If this file is not available on your system, you may try another application which supports X Toolkit options, such as `xcalc` or `xedit`.

All Motif applications accept all the command line options (and standard resources) of X Toolkit applications. Try modifying the `Finalsurvey` resource specification file. (which is in a directory named `Resource`) For example, change the label names (and thus the interests of the survey to help people make new friends).

Also try command line options, such as:

```
finalsurvey -bg brown -iconic &
```

```
finalsurvey -geometry +100-100 &
```

* How could these command line options be placed into a resource description file?

Terminal Emulators-xterm

- DEC vt102 or Tektronix 4014 compatibility

Note: the Tektronix emulator is supposed to do that!

- `xterm` command line options
 - help (print message describing options)
 - bg color (background color--default: white)
 - fg color (text color--default: black)
 - ms color (pointer color--default: foreground color)
 - bd color (border color--default: black)
 - cr color (text cursor color--default: same as text color)
 - fn fontname (default: 'fixed')
 - geometry geometryspecification (location and size)
 - title titlestring
 - iconic (start terminal emulator as an icon)
 - +t/-t (start in vt102 mode or Tektronix 4014 mode)
 - C (window receives console output)
 - tm string (specify terminal setting)
 - tn terminaltype (set `$TERM` environmental variable)

Terminal Emulators-xterm

- `xterm` command line options

`-bw number` (width of window border in pixels--default: 1)

`+sb/-sb` (set scroll bar--default: on)

`+j/-j` (set jump scrolling--default: on)

`-sl number` (set number of scrolled lines retained)

`+rw/-rw` (set reverse wrap around--default: off)

`+mb/-mb` (set margin bell--default: off)

`+ls/-ls` (set terminal as login shell/read user's `.login` or `.profile`--default: starts as subshell)

`-display host:display.screen`

(run on specified display)

Note: The values you see on the screen may not seem to match the ones listed as defaults. In which files might the values be set?

Terminal Emulator-xwsh

- `xwsh` command line options
 - help (print message describing options)
 - bg color (background color)
 - fg color (text color)
 - cursorfg color (text cursor foreground color)
 - cursorbg color (text cursor background color)
 - selfg color (selection foreground color)
 - selbg color (selection background color)
 - fn fontname (primary rendition font)
 - geometry geometryspecification (location and size)
 - min COLSxLINES
 - max COLSxLINES
 - title titlestring
 - iconic
 - icontitle titlestring
 - console (window receives console output)

Terminal Emulator-xwsh

- `xwsh` command line options
 - name WM_CLASS (changes instance name)
 - vb (visual bell)
 - ibm (IBM RT keyboard)
 - vt100 (vt100 mode)
 - display host:display.screen

User Preference Utilities

- `xset`
utility to control keyboard and mouse preferences
- command line options
 - not all options fully supported

`q`

query

`[-]b [volume pitch duration] OR [on/off]`

bell

`[-]c [on/off] OR [volume]`

key click

`[-]led [on/off] [number]`

LED's

`m [acceleration threshold] OR [default]`

mouse

`[-]r [on/off]`

autorepeat

`s [time cycle] [blank/noblank] [expose/noexpose]
[on/off] [default]`

screen saver

User Preference Utilities

- `xsetroot`
utility to control root window (background) appearance
- **command line options**
 - `-help`
 - `-def`
set default pattern, root weave
 - `-cursor cursorbitmapfile maskbitmapfile`
default cursor is X pointer
 - `-bitmap bitmapfile`
look in `/usr/include/X11/bitmaps`
 - `-mod x y`
makes plaid-like grid pattern
x, y are integers from 1 to 16
 - `-fg color` (only useful for `-cursor`, `-bitmap`, `-mod`)
 - `-bg color` (only useful for `-cursor`, `-bitmap`, `-mod`)
 - `-gray` or `-grey`
 - `-solid color`

.sgisession file

- A simple shell script
- Put in standard executable UNIX shell commands

For example:

```
xclock -fg white -bg black -geometry 164x164+1099+133 &  
xbiff -fg green -bg gray20 -geometry 88x48+1184+31 &  
xcalc -fg NavyBlue -bg PeachPuff -geometry 188x231+1073+345 &
```

Terminal Emulator and Session Customization Exercise

Purpose:

This exercise will give you practice in customizing the login sequence and make one's window environment tailored and unique.

Instructions:

1. Using command line options, create a terminal emulation window:
 - with a specified title string
 - with a different font
 - with a specified location and size
 - as an icon, initially
2. Using `xset` from the command line, query the current preferences. Then turn on and off keyboard LED's, bell, and/or key click.
3. Use `xsetroot` to change the background. Try several bitmaps. (You will find several bitmaps in `/usr/include/X11/bitmaps`. The instructor may also load additional bitmaps in files and subdirectories under the `bitmaps` directory. There are several bitmaps with the `.xbm` suffix.) For an extra challenge, create your own bitmap, using the utility, `bitmap`, and use it for a background.
4. What happens if you log in and you have neither an `.sgisession` file, nor an `.xsession` file. Try it.
5. Create a simple `.sgisession` file which customizes your session at login. For example:
 - open additional terminal emulators, with different characteristics (title, font, location, etc.)
 - establish keyboard and mouse characteristics (such as key click, bell, etc.)
6. * If you create an `.xsession` file, what effect does your `.sgisession` file have? Try to create one.
 - Warning: Do not simply copy `/usr/lib/X11/xdm/Xsession` into your home directory. If you copy it into your home directory, edit the file, so it does

not look for an `.xsession` file in your home directory, thus potentially causing a loop.

7. * Modify a resource file, such as `$HOME/.Xdefaults`. Add the line:

```
fred*Background: Aquamarine
```

Then use `xrdb` to merge or load the property. Then run `wsh -n fred`. What happened?

8. * How do you make changes to a resource for a specific client which are:
- system-wide?
 - for only this user, everytime he or she logs in?
 - for only this user, until he or she logs out?

Toolchest

- `/usr/bin/X11/toolchest--executable`

command line options

`[-icon] [-horizontal] [-geometry] [-decal]`

- **resource specification file**

`/usr/lib/X11/app-defaults/Toolchest`

```
Toolchest*fontList: -adobe-helvetica-bold-o-normal--14-*
Toolchest*geometry: +10+10
Toolchest*background: sgiGray76
Toolchest*horizontal: False
Toolchest*showDecal: True
```

- **resource description file**

bindings for menus

First looks for `$HOME/.chestr`

If none, then `/usr/lib/X11/system.chestr`






- **description file functions**

`f.menu`, `f.title`, `f.exec`, `f.checkexec`, `f.separator`,
and `f.label`

- **description file format**

label function argument[s]

Toolchest

<i>System</i>	
<i>Windows</i>	
<i>Tools</i>	
<i>Demos</i>	
<i>Overview</i>	

- a portion of `/usr/lib/X11/system.chestrc`

```
# Top Level Menu Description
Menu ToolChest
{
    "System"          f.menu System
    no-label          f.separator
    "Windows"         f.menu Windows
    no-label          f.separator
    "Tools"           f.menu Tools
    no-label          f.separator
    "Demos"           f.menu Demos
    no-label          f.separator
    "Overview"        f.menu Overview
}
Menu System
{
    "WorkSpace"       f.checkexec "/usr/sbin/workspace"
    "Transfer Manager" f.checkexec "/usr/sbin/transfermanager"
    "Print Manager"   f.checkexec "/usr/lib/vadmin/printers"
    no-label          f.separator
    "System Manager"  f.checkexec "/usr/sbin/vadmin -x75.0 -
y640.0"
    "Legal Notice"    f.checkexec "/usr/bin/X11/xconfirm -geometry
550x225 -z 'Legal Notice' -b Dismiss -o /usr/etc/legal_notice >/
dev/null"
    "System Shutdown" f.checkexec "/usr/sbin/systemdown"
    no-label          f.separator
    "Log Out"         f.checkexec "/usr/bin/X11/endsession"
}
}
```

Window Manager

- several window managers to choose from

all in `/usr/bin/X11`

`4Dwm`, `mwm`, `twm`

- to choose a window manager other than `4Dwm`, create `$HOME/.xsession`
- resource definition file (for `4Dwm`)

`/usr/lib/X11/app-defaults/4Dwm`

better to make changes in `$HOME/.Xresources` OR

`$HOME/.Xdefaults`

- common bindings

menus and buttons

```
4Dwm*buttonBindings:      4DwmButtonBindings
4Dwm*keyBindings:         4DwmKeyBindings
4Dwm>windowMenu:          4DwmDefaultWindowMenu
```

fonts

```
4Dwm*menu*fontList:       -adobe-helvetica-bold-o-normal--14-*
4Dwm*icon*fontList:       -adobe-helvetica-medium-r-normal--14-*
```

toolchest binding

```
4Dwm*Toolchest*clientDecoration:  none
```

4Dwm Resources

- more common bindings
- active window color

```
4Dwm*activeBackground:  sgibrightgray
```

- icons

arrange icons on particular area of the screen

```
4Dwm*iconAutoPlace:  True
```

describe scheme for icon placement (row, column, etc.)

```
4Dwm*iconPlacement:  left top tight
```

```
4Dwm*SG_iconPlacementTopMargin:  100
```

```
4Dwm*lowerOnIconify:  False
```

icons are placed in a scrollable icon box

```
4Dwm*useIconBox:  False
```

icon appearance

```
4Dwm*iconDecoration:  label image
```

```
4Dwm*iconImageMaximum:  85x67
```

icon files

SGL format

filename: `instance.icon`, `class.icon` or `default.icon`

directory search path: `4Dwm*bitmapDirectory`,

`$HOME/.icons`, `/usr/lib/images`

More 4Dwm Resources

- **input focus**

When does window receive keyboard input?

when pointer is over the window OR

when the window is explicitly chosen (click on title bar)

```
4Dwm*keyboardFocusPolicy: pointer
4Dwm*keyboardFocusPolicy: explicit
```

If focus policy is explicit, make window raised when active

```
4Dwm*raiseKeyFocus: False
```

- **mouse clicking**

maximum time between button presses for double-click (in millisecs)

```
4Dwm*doubleClickTime: 600
```

- **feedback while sizing windows**

```
4Dwm*showFeedback: behavior restart quit kill
4Dwm*feedback*fontList: -adobe-helvetica-bold-r-normal--14-*
```

.4Dwmrc

- customizing the resource description file (`$HOME/.4Dwmrc`)
- resource description files (for 4Dwm)
 - bindings for menus, buttons, keys
 - looks first for `$HOME/.4Dwmrc`
 - then in `/usr/lib/X11/system.4Dwmrc`
 - make minor or wholesale changes
- formats
 - button/key context function [argument] (for buttons and keys)
 - label mnemonic accelerator function [argument] (for menus)
- functions include:
 - `f.raise`, `f.lower`, `f.resize`, `f.move`, `f.circle_up`,
`f.circle_down`, `f.kill`, `f.refresh`, `f.restart`,
`f.quit_mwm` to manipulate windows
 - `f.menu`, `f.post_wmenu` to control menus
 - `f.title`, `f.separator` to create menu labels
 - `f.exec`, `f.checkexec` to spawn other system calls

4Dwm Button Bindings

- sample button bindings

Note: On the IRIS, Meta is Alt

```
#
# SGI 4Dwm Button Binding Descriptions
#
Buttons 4DwmButtonBindings
{
    <Btn1Down>          root          f.menu 4DwmRootMenu
!    <Btn1Down>          window        f.raise
    <Btn1Click>         frame         f.raise
!    Click, so we don't disable the Btn1Down ability to move the
icon.
!    Btn1Down doing an f.move is in the OSF manuals.
    <Btn1Click>         icon          f.normalize
    Ctrl<Btn1Down>      frame|icon    f.lower
    Shift<Btn1Down>     frame|icon    f.raise
    Shift Ctrl<Btn1Down> window        f.raise_resize

    <Btn2Down>          frame|icon    f.move
    Ctrl<Btn2Down>      frame|icon    f.lower
    Shift<Btn2Down>     frame|icon    f.raise
    Shift Ctrl<Btn2Down> window        f.raise_move

    <Btn3Down>          root          f.menu 4DwmRootMenu
    <Btn3Down>          frame|icon    f.post_wmenu
    Ctrl<Btn3Down>      frame|icon    f.post_wmenu
    Shift<Btn3Down>     frame|icon    f.post_wmenu
    Meta<Btn3Down>      window|icon   f.post_wmenu
    Shift Ctrl<Btn3Down> window        f.post_wmenu
}
```

4Dwm Key Bindings

- sample key bindings

```
#
# SGI 4Dwm key binding descriptions
#
Keys 4DwmKeyBindings
{
# Same as the standard OSF Key Bindings
  Shift<Key>Escape      window|icon          f.post_wmenu
  Meta<Key>space        window|icon          f.post_wmenu
  Meta<Key>Tab          root|icon|window     f.next_key
  Meta Shift<Key>Tab    root|icon|window     f.prev_key
  Meta<Key>Escape       root|icon|window     f.next_key
  Meta Shift<Key>Escape root|icon|window     f.prev_key
  Meta<Key>F6           window               f.next_key transient
  Meta Shift<Key>F6     window               f.prev_key transient
  <Key>F4              icon                  f.post_wmenu

# SGI added key bindings
  Shift<Key>Escape      root                  f.menu 4DwmRootMenu
  Meta<Key>space        root                  f.menu 4DwmRootMenu
}
```

4Dwm Menu Bindings

- sample menu bindings

Note: Note keyboard Accelerators and Mnemonics for menu selections

```
#
# SGI 4Dwm Root Menu Descriptions.
#
Menu 4DwmRootMenu
{
    "4D Window Manager"          f.title
    "Log Out"                    _L    f.exec "/usr/bin/X11/endsession"
}

#
# SGI 4Dwm Window Menu Description.
#

Menu 4DwmWindowMenu
{
    Restore          _R    Alt<Key>F5    f.normalize
    Move             _M    Alt<Key>F7    f.move
    Size             _S    Alt<Key>F8    f.resize
    Minimize         _n    Alt<Key>F9    f.minimize
    Maximize         _x    Alt<Key>F10   f.maximize
    Raise            _a    Alt<key>F1    f.raise
    Lower            _L    Alt<Key>F3    f.lower
    no-label
    Close            _C
    Quit             _Q    f.quit_app
}
```


Key Mapping

- **keycode**--value that the physical key generates
 - cannot be mapped to other keys
- **keysym**--symbolic name that represents the label on the key
 - corresponds to a function
 - keysym (function) can be mapped onto another keycode
- **modifier**--special key which, when active, changes codes returned by keyboard
 - Examples: Caps_Lock, Control, Shift, Meta
- **xmodmap**
 - modify input device key and button assignments (mapping)
 - command line options

Table 5: xmodmap command line options

no options	display modifier key assignments
-pk	display keysym/keycode assignments
-pp	display pointer button assignments

- **xev**
 - display events which occur in this window

Key Mapping

- `xmodmap -`
 - start mode for remapping
 - read from standard input until End-of-File (control-D)
- **Commands to change modifier keys**

```
% xmodmap -  
clear MODIFIER  
add MODIFIER = KEYSYM  
remove MODIFIER = KEYSYM  
^d
```

- For example, switching `Caps_Lock` and `Control_L`

```
% xmodmap -  
clear lock  
remove control = Control_L  
add control = Caps_Lock  
add lock = Control_L  
^d
```

- **Change keysym assignment**
 - For example, make the Delete key act like the BackSpace

```
% xmodmap -e 'keysym Delete = BackSpace'
```

- **Totally screw up your mouse buttons (pointer)**

```
% xmodmap -e 'pointer = 1 2 3'  
% xmodmap -e 'pointer = 2 3 1'  
% xmodmap -e 'pointer = default'
```

Total Customization Exercise

Purpose:

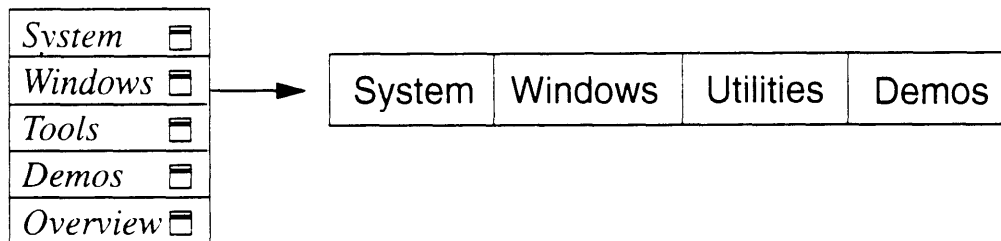
This exercise will give you practice in customizing the toolchest, window manager, and input devices.

Instructions:

1. Toolchest Customization

Change the menus on the toolchest. For example, change the label "Tools" to "Utilities." Remove the label "Overview."

* Upon login, make the toolchest come up horizontally. Change the default color, font, and position (geometry) of the toolchest.



2. Window Manager Customization

Change to a different window manager. Upon login, have the system start the `twm` or `mwm` window managers. Identify where the resource definition files for these window managers are.

Go back to `4Dwm`. Create a local resource description file for your window manager. (You can copy one from the `/usr/lib/X11` directory.) Modify the SGI window pop-up menu so that it says "Push" instead of "Lower." Change or remove the mnemonic.

Change the fonts used by the window manager. Change the `4Dwm*menu*fontList:` resource.

Modify a key binding. For example, make one of the function keys bring up a simple client, such as the flight simulator, window shell, or X calculator. Or make the PrintScreen key print a screen dump (see `xdpr` command). If no printer is available, make the PrintScreen key save the screen into a file (see `xwd`

and `xpr` commands).

3. Keyboard and mouse customization

Make the Caps Lock button a control key, and make the right control key the Caps Lock button. Rotate the operation of the mouse buttons. Restore all operations of the keyboard and mouse back to defaults.

Final Challenge

Can you do everything learned in this chapter?

Purpose:

This exercise will give you review several customization topics from this chapter.

Instructions:

1. Edit an `.sgisession` file so that `xbiff` (the mail notifier), `xclock`, and `xcalc` start up in a given position with specified colors. How would you make the calculator start up in an iconified form? Create a file which specifies reasonable defaults (colors, positions, fonts, and other resources) for these and other clients.
2. Set your window manager to put your icons along the top of your screen in the upper right. Then create an icon box there. Restart the window manager for your changes to take effect.
3. Remap keys so that `Caps_Lock` and `Control` exchange functions.
4. Modify your toolchest. Change the color, font, and location of the toolchest. Alter items in the toolchest menus so that your favorite applications are available. Make the toolchest horizontal or vertical.
5. Alter items in the window manager menus. Change the "Minimize" label to "Iconify"
6. At login, create more than one terminal emulator. Specify different positions, colors, and fonts, for those terminal emulators. Have one of the terminal emulators startup as an icon.
7. * At login, have a root window (background) come up with a repeating bitmap pattern. Start a window manager other than `4Dwm`. Make sure you load the `RESOURCE_MANAGER` property of the server with preferred resource bindings.
8. * Change the mouse warp and key warp factors. Turn on and off the keyclick or bell. Remap the mouse buttons completely (for left-handers). Remap window manager buttons to different functions.

Glossary

accelerator	a keyboard key (or keys) used to cause an action, in place of another input device. For example, accelerators can choose a menu entry without using the mouse.
binding	association of a resource and a value
class	general group of widgets which share common data and methods
context	the location of the keyboard input focus: window, icon or root (background) window
instance	individual copy of a data structure containing the current state of the widget
keycode	value that the physical key generates. The keycode cannot be mapped to other keys
keysym	symbolic name that represents the label on the key and function of the key
modifier	special key (such as Control, Caps Lock, Shift, Alt) which, when active, changes codes returned by keyboard
mnemonic	a single character of a menu selection which can be used to accelerate menu selection
property	information widely available to other programs
resource	a program parameter that controls appearance or behavior or a client
widget	a user interface component consisting of an X window and some procedures which operate and communicate with the window.

Bibliography

This chapter was heavily based on materials in these books. Liberal appropriation of material took place.

O'Reilly and Associates, *The X Window System Series (Volume 3)*, O'Reilly and Associates, 1988, 1989, 1990. Volume 3, the end-users reference manual, was a major source of information.

Open Software Foundation, *OSF/Motif Series (5 volumes)*, Prentice Hall, 1990. ISBN 0-13-640525-8. The Programmer's Guide was used most significantly in this chapter.

Young, Douglas A., *The X Window System: Programming and Applications with Xt*, Prentice Hall, 1990. ISBN 0-13-497074-8. This book is definitely the recommended first purchase for anyone thinking of programming with higher-level toolkits in X. This book is available in different flavors: Xt and Motif. A C++ version is being written in 1991. Doug's work was an inspiration for this course and his personal assistance has been greatly appreciated.

Summary

This chapter summary is excerpted from the manual page for X(1).

This material is Copyright 1984, 1985, 1986, 1987, 1988, and 1989, by the Massachusetts Institute of Technology.

xdm (the X Display Manager)

This program is started by the system at boot time and takes care of keeping the server running. You will see a window on the screen welcoming you to the system and asking for your username and password. After you have successfully logged in, xdm will start up your X environment. By default, if you have an executable file named `.xsession` in your home directory, xdm will treat it as a program (or shell script) to run to start up your initial clients (such as terminal emulators, clocks, a window manager, user settings for things like the background, the speed of the pointer, etc.).

COLOR NAMES

Most applications provide ways of tailoring (usually through resources or command line arguments) the colors of various elements in the text and graphics they display.

Colors are usually specified by their commonly-used names (for example, red, white, or medium slate blue). The server translates these names into appropriate screen colors using a color database that can usually be found in `/usr/lib/X11/rgb.txt`. Color names are case-insensitive, meaning that red, Red, and RED all refer to the same color.

Many applications also accept color specifications of the following form:

```
#rgb
#rrggb
#rrrggbbb
#rrrrgggbbbb
```

where `r`, `g`, and `b` are hexadecimal numbers indicating how much red, green, and blue should be displayed (zero being none and `ffff` being on full). Each field in the specification must have the same number of digits (e.g., `#rrgb` or `#gbb` are not allowed). Fields that have fewer than four digits (e.g. `#rgb`) are padded out with zero's following each digit (e.g. `#r000g000b000`).

FONT NAMES

Collections of characters for displaying text and symbols in X are known as fonts. A font typically contains images that share a common appearance and look nice together (for example, a single size, boldness, slant, and character set). Similarly, collections of fonts that are based on a common type face (the variations are usually called roman, bold, italic,

bold italic, oblique, and bold oblique) are called families. Sets of font families of the same resolution (usually measured in dots per inch) are further grouped into directories (so named because they were initially stored in file system directories). Each directory contains a database which lists the name of the font and information on how to find the font. The server uses these databases to translate font names (which have nothing to do with file names) into font data.

The list of font directories in which the server looks when trying to find a font is controlled by the font path. Although most installations will choose to have the server start up with all of the commonly used font directories, the font path can be changed at any time with the `xset` program. However, it is important to remember that the directory names are on the server's machine, not on the application's. The default font path for the sample server contains four directories:

```
/usr/lib/X11/fonts/misc  
/usr/lib/X11/fonts/75dpi  
/usr/lib/X11/fonts/100dpi  
/usr/lib/fmfonts
```

The `xlsfonts` program can be used to list all of the fonts that are found in font databases in the current font path. Font names tend to be fairly long as they contain all of the information needed to uniquely identify individual fonts. However, the sample server supports wildcarding of font names, so the full specification

```
-adobe-courier-medium-r-normal--10-100-75-75-m-60-iso8859-1
```

could be abbreviated as:

```
-*-courier-medium-r-normal--*-100-*-*-*-*
```

or, more tersely (but less accurately):

```
*-courier-medium-r-normal--*-100*
```

Because the shell also has special meanings for `*` and `?`, wildcarded font names should be quoted:

```
% xlsfonts -fn '*-courier-medium-r-normal--*-100*'
```

If more than one font in a given directory in the font path matches a wildcarded font name, the choice of which particular font to return is left to the server. However, if fonts from more than one directory match a name, the returned font will always be from the first such directory in the font path. The example given above will match fonts in both the 75dpi and 100dpi directories; if the 75dpi directory is ahead of the 100dpi directory in the font path, the smaller version of the font will be used.

GEOMETRY SPECIFICATIONS

One of the advantages of using window systems instead of hardwired terminals is that

applications don't have to be restricted to a particular size or location on the screen. Although the layout of windows on a display is controlled by the window manager that the user is running (described below), most X programs accept a command line argument of the form `-geometry WIDTHxHEIGHT+XOFF+YOFF` (where `WIDTH`, `HEIGHT`, `XOFF`, and `YOFF` are numbers) for specifying a preferred size and location for this application's main window.

The `WIDTH` and `HEIGHT` parts of the geometry specification are usually measured in either pixels or characters, depending on the application. The `XOFF` and `YOFF` parts are measured in pixels and are used to specify the distance of the window from the left or right and top and bottom edges of the screen, respectively. Both types of offsets are measured from the indicated edge of the screen to the corresponding edge of the window. The X offset may be specified in the following ways:

`+XOFF`

The left edge of the window is to be placed `XOFF` pixels in from the left edge of the screen (i.e. the X coordinate of the window's origin will be `XOFF`). `XOFF` may be negative, in which case the window's left edge will be off the screen.

`-XOFF`

The right edge of the window is to be placed `XOFF` pixels in from the right edge of the screen. `XOFF` may be negative, in which case the window's right edge will be off the screen.

The Y offset has similar meanings:

`+YOFF`

The top edge of the window is to be `YOFF` pixels below the top edge of the screen (i.e. the Y coordinate of the window's origin will be `YOFF`). `YOFF` may be negative, in which case the window's top edge will be off the screen.

`-YOFF`

The bottom edge of the window is to be `YOFF` pixels above the bottom edge of the screen. `YOFF` may be negative, in which case the window's bottom edge will be off the screen.

Offsets must be given as pairs; in other words, in order to specify either `XOFF` or `YOFF` both must be present. Windows can be placed in the four corners of the screen using the following specifications:

`+0+0` upper left hand corner.

`-0+0` upper right hand corner.

`-0-0` lower right hand corner.

`+0-0` lower left hand corner.

In the following examples, a terminal emulator will be placed in roughly the center of the

screen and a load average monitor, mailbox, and clock will be placed in the upper right hand corner:

```
xterm -fn 6x10 -geometry 80x24+30+200 &
```

```
xclock -geometry 48x48-0+0 &
```

```
xload -geometry 48x48-96+0 &
```

```
xbiff -geometry 48x48-48+0 &
```

OPTIONS

Most X programs attempt to use the same names for command line options and arguments. All applications written with the X Toolkit Intrinsics automatically accept the following options:

-display display

-geometry geometry

-bg color, -background color

-bd color, -bordercolor color

-bw number, -borderwidth number

-fg color, -foreground color

-fn font, -font font

-iconic

-name

This option specifies the name under which resources for the application should be found. This option is useful in shell aliases to distinguish between invocations of an application, without resorting to creating links to alter the executable file name.

-rv, -reverse, +rv

-selectionTimeout

This option specifies the timeout in milliseconds within which two communicating applications must respond to one another for a selection request.

-synchronous

This option indicates that requests to the X server should be sent synchronously, instead of asynchronously. Since Xlib normally buffers requests to the server, errors do not necessarily get reported immediately after they occur. This option turns off the buffering so that the application can be debugged. It should never be used with a working

program.

-title string

-xnllanguage language[_territory][.codeset]

This option specifies the language, territory, and codeset for use in resolving resource and other filenames.

-xrm resourcestring

This option specifies a resource name and value to override any defaults. It is also very useful for setting resources that don't have explicit command line arguments.

RESOURCES

To make the tailoring of applications to personal preferences easier, X supports several mechanisms for storing default values for program resources (e.g. background color, window title, etc.) Resources are specified as strings of the form

appname*subname*subsubname...: value

that are read in from various places when an application is run. By convention, the application name is the same as the program name, but with the first letter capitalized (e.g. Bitmap or Emacs) although some programs that begin with the letter "x" also capitalize the second letter for historical reasons.

RESOURCE_MANAGER root window property

Any global resources that should be available to clients on all machines should be stored in the RESOURCE_MANAGER property on the root window using the xrdp program. This is frequently taken care of when the user starts up X through the display manager or xinit.

application-specific files

Programs that use the X Toolkit Intrinsics will also look in the directories named by the environment variable XUSERFILESEARCHPATH or the environment variable XAPPLRESDIR, plus directories in a standard place (usually under /usr/lib/X11/, but this can be overridden with the XFILESEARCHPATH environment variable) for application-specific resources. See the X Toolkit Intrinsics - C Language Interface manual for details.

XENVIRONMENT

Any user- and machine-specific resources may be specified by setting the XENVIRONMENT environment variable to the name of a resource file to be loaded by all applications. If this variable is not defined, a file named \$HOME/.Xdefaults-hostname is looked for instead, where hostname is the name of the host where the application is executing.

-xrm resourcestring

Applications that use the X Toolkit Intrinsics can have resources specified from the command line. The resource string is a single resource name and value as shown above. Note that if the string contains characters interpreted by the shell (e.g., asterisk), they must be quoted. Any number of `-xrm` arguments may be given on the command line.

Program resources are organized into groups called classes, so that collections of individual resources (each of which are called instances) can be set all at once. By convention, the instance name of a resource begins with a lowercase letter and class name with an upper case letter. Multiple word resources are concatenated with the first letter of the succeeding words capitalized. For example, here are some resources:

`background` (class `Background`)

This resource specifies the color to use for the window background.

`borderWidth` (class `BorderWidth`)

This resource specifies the width in pixels of the window border.

`borderColor` (class `BorderColor`)

This resource specifies the color to use for the window border.

Most applications using the X Toolkit Intrinsics also have the resource `foreground` (class `Foreground`), specifying the color to use for text and graphics within the window.

By combining class and instance specifications, application preferences can be set quickly and easily. Users of color displays will frequently want to set `Background` and `Foreground` classes to particular defaults. Specific color instances such as text cursors can then be overridden without having to define all of the related resources. For example,

```
XTerm*multiScroll: on
```

```
XTerm*Font: 6x10
```

```
XTerm*scrollbar*thickness: 5
```

```
XTerm*multiClickTime: 500
```

```
XTerm*ttyModes: intr ^c erase ^? kill ^u
```

```
XLoad*Background: gold
```

If these resources are stored in a file called `.Xresources` in your home directory, the session startup will send your requests to the server with the command, `xrdb`.

KEYS

The X keyboard model is broken into two layers: server-specific codes (called keycodes) which represent the physical keys, and server-independent symbols (called keysyms) which represent the letters or words that appear on the keys. Two tables are kept in the server for converting keycodes to keysyms:

modifier list

Some keys (such as Shift, Control, and Caps Lock) are known as modifier and are used to select different symbols that are attached to a single key (such as Shift-a generates a capital A, and Control-l generates a formfeed character ^L). The server keeps a list of keycodes corresponding to the various modifier keys. Whenever a key is pressed or released, the server generates an event that contains the keycode of the indicated key as well as a mask that specifies which of the modifier keys are currently pressed. Most servers set up this list to initially contain the various shift, control, and shift lock keys on the keyboard.

keymap table

Applications translate event keycodes and modifier masks into keysyms using a keysym table which contains one row for each keycode and one column for various modifier states. This table is initialized by the server to correspond to normal typewriter conventions, but is only used by client programs.

Engineer's Handbook General Stuff

May 5, 1994

Customizing X

access via: handbook config_x
probable data location: clubted.csd:/usr/graphics/develop/motif/book/x2.ps

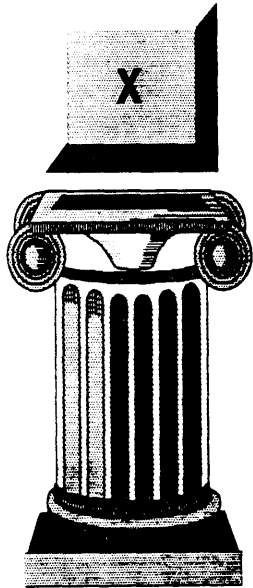




SiliconGraphics
Computer Systems

X Window System Customizing X

Student Workbook



Publication Number: T41W0 IN 001
September 3, 1991

RESTRICTION ON USE

This document is protected by copyright and contains information proprietary to Silicon Graphics, Inc. Any copying, adaptation, distribution, public performance, or public display of this document without the express written consent of Silicon Graphics, Inc., is strictly prohibited. The receipt or possession of this document does not convey the rights to reproduce or distribute its contents, or to manufacture, use, or sell anything that it may describe, in whole or in part, without the specific written consent of Silicon Graphics, Inc.

Copyright © 1991 Silicon Graphics, Inc. All rights reserved.

U.S. GOVERNMENT RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure of the data and information contained in this document by the Government is subject to restrictions as set forth in FAR 52.227-19(c)(2) or subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and/or in similar or successor clauses in the FAR, or the DOD or NASA FAR Supplement. Unpublished rights reserved under the Copyright Laws of the United States. Contractor/manufacturer is Silicon Graphics, Inc., 2011 N. Shoreline Blvd., P.O. Box 7311, Mountain View, CA 94039-7311.

The contents of this publication are subject to change without notice.

Module Objectives

- Identify the sequence of window system events during boot-up and log in.
- Specify colors, fonts, and other resources.
- Identify and modify the appropriate files to customize (text, color, location, fonts, etc.) the login, icons, toolchests, window manager, etc.
- Specify command line options for commonly used X clients, such as `xterm` and `xwsh`.
- Customize mapping of keyboard keys and mouse buttons.

Sequence of Events

- System boot-up

`/usr/bin/X11/xdm`--the X display manager

- Login prompt--Authentication Widgets

SGI visual login--`/usr/sbin/pandora`

standard X

`/usr/lib/X11/xdm/Xresources`

`/usr/lib/X11/xdm/Xlogin`

- Logging in (session start)

`/usr/lib/X11/xdm/Xsession`

Sequence of Events cont'd

- `/usr/bin/X11/4Dwm` window manager
resource definition file
`/usr/lib/X11/app-defaults/4Dwm`
window manager preferences
`/usr/lib/X11/system.4Dwmrc`
`$HOME/.4Dwmrc`
- `$HOME/.sgisession` startup script
- `/usr/bin/X11/toolchest`
resource definition file
`/usr/lib/X11/app-defaults/Toolchest`
toolchest preferences
`/usr/lib/X11/system.chestrc`
`$HOME/.chestrc`
- `/bin/wsh`
resource definition file
`/usr/lib/X11/app-defaults/XWsh`
- `/usr/bin/X11/reaper`
manages and closes a login session

Customizing Your Desk

- specifying colors
- specifying fonts
- `xterm`, `xwsh`--terminal emulators
 - command line options or default files
 - specifying placement, font, color
- `xsetroot`, `bitmap`--making root (background) windows
- `xset`--display and keyboard user preferences

xdm-X Display Manager

- **executable**-- /usr/bin/X11/xdm
 - provides services similar to `init`, `getty`, and `login`
 - launched from `/etc/rc2.d` when starting multi-user mode

```
# Start X Display Manager
#
# $Revision: 1.2 $
#

IS_ON=/etc/chkconfig
XSGI=/usr/bin/X11/Xsgi
XDM=/usr/bin/X11/xdm

case "$1" in
  'start')
    if test -x $XDM; then
      if $IS_ON windowsystem && test -x $XSGI ||
        $IS_ON xdm; then
        exec $XDM
      fi
    fi
    ;;

  'stop')
    /etc/killall -TERM xdm
    ;;

  *)
    echo "usage: /etc/init.d/xdm {start|stop}"
    ;;
esac
```

xdm-X Display Manager

- related files--/usr/lib/X11/xdm

xdm-config--where are the files

```
DisplayManager.servers:           /usr/lib/X11/xdm/Xservers
DisplayManager.errorLogFile:      /usr/lib/X11/xdm/xdm-errors
DisplayManager.pidFile:           /usr/lib/X11/xdm/xdm-pid
DisplayManager*resources:        /usr/lib/X11/xdm/Xresources
DisplayManager*reset:             /usr/lib/X11/xdm/Xreset
DisplayManager*authorize:         off
DisplayManager*startup:           /usr/lib/X11/xdm/Xstartup-remote
DisplayManager._0.startup:        /usr/lib/X11/xdm/Xstartup
DisplayManager*session:           /usr/lib/X11/xdm/Xsession-remote
DisplayManager._0.session:        /usr/lib/X11/xdm/Xsession
DisplayManager._0.openTimeout:    90
DisplayManager._0.startAttempts:  1
DisplayManager._0.authFile:       /usr/lib/X11/xdm/xdm-auth
DisplayManager._0.loginProgram:   /usr/lib/X11/xdm/Xlogin
DisplayManager._0.terminateServer: False
```

Xservers

list of servers local to this host

Xlogin, Xresources

control appearance of login screen

Xstartup

run after user/password pair is authenticated (login)

Xsession

script run as user's session

Xreset--run when session is terminated

Login Authentication

- choice of 2 authentication screens

pandora--SGI proprietary visual login OR standard X authentication

- `/usr/lib/X11/xdm/Xlogin`
 1. gamma correction
 2. screen background and cursors
 3. pandora or standard X authentication chosen

```
glGammaFile=/etc/config/system.glGammaVal
glGammaDefault="1.7"

if [ -r $glGammaFile ]; then
    glGammaVal=`cat $glGammaFile`
else
    glGammaVal=$glGammaDefault
fi

screens=`/usr/bin/X11/xlistscrns`
for screen in $screens
do
    HOME=/ DISPLAY=$screen /usr/sbin/gamma $glGammaVal
    /usr/bin/X11/xsetroot -solid sgitlightblue -display $screen
    /usr/bin/X11/xsetroot -cursor_name X_cursor \
        -fg red -bg white -display $screen
done

/usr/bin/X11/xlistscrns -i

if /etc/chkconfig visuallogin ; then
    if /etc/chkconfig noiconlogin ; then
        /usr/sbin/pandora -s
    else
        /usr/sbin/pandora
    fi
fi
```

Login Authentication (cont'd)

- /usr/lib/X11/xdm/Xresources--standard X

```
#ifdef COLOR
xlogin*Background:      grey70
xlogin*Foreground:      white
xlogin*greetColor:      #0000b6
xlogin*failColor:       #aa0000
#else
xlogin*Background:      white
xlogin*Foreground:      black
xlogin*greetColor:      black
xlogin*failColor:       black
#endif /* COLOR */
xlogin*greeting:        Welcome to IRIX 4.0.1
xlogin*promptColor:     black
xlogin*promptFont:      -*-charter-medium-i-normal-**-240-100-
100-**-***-
xlogin.Login.font:      -*-helvetica-medium-r-normal-**-240-100-
100-**-***-

xlogin*login.translations: #override\
    <Key>F1: set-session-argument(failsafe) finish-field()\n\
    <Key>Return: set-session-argument() finish-field()

xlogin*borderWidth:     1
```

Resources

- What's a resource?
 - almost every feature of a program controllable by a variable
- Resources can be specified on the command line or in a file
- Examples of resources
 - background (color)
 - foreground (color of text or bitmapped graphics)
 - font
 - geometry (size & placement of windows)
- Resource variable types
 - numeric
 - `xlogin*borderWidth: 1`
 - Boolean
 - `scrollBar: true`
 - string value
 - `xlogin*promptColor: black`
 - event translation
 - `xlogin*login.translations: <Key>Return: \
set-session-argument() finish-field()`

Resources

- **Specifying resources**
 - Resource definition file (can be in many places!)
 - Command line options
- **Syntax of Resource Definition File**
 - Components of a binding

```
object.subobj[.subobj...].attribute: value
```

`object` is the client program or specific instance of the program

`subobj` is widget hierarchy level (major structures, such as windows, menus, scrollbars, etc.)

`attribute` is the feature of the last subobject, such as the label appearing on it

`value` is the setting of the resource

Resources

- **Loose Binding**

represented by an asterisk *

a wildcard which substitutes for any number of levels of subobjects

- **Tight Binding**

represented by a period .

components adjacent to period must match hierarchy exactly

- **Examples**

`xcalendar*daynumbers*Background: Orange`

`xcalendar*daynumbers.21*Background: Yellow`

- changes color behind days of the month to orange, except for the third sunday of the month (21st rectangle), which is yellow

Color Specification

- 2 methods of color specification
- hexadecimal specification

each field (r, g, b) must have same number of digits

#rgb

#rrggbb

#rrrrggbbbb

#rrrrggggbbbb

- color database names

case insensitive

names are in `/usr/lib/X11/rgb.txt`

127	255	212	aquamarine
102	205	170	medium aquamarine
0	0	0	black
0	0	255	blue
95	158	160	cadet blue
100	149	237	cornflower blue
72	61	139	dark slate blue
173	216	230	light blue
176	196	222	light steel blue

Font Specification

- list of font names in `fonts.dir` and `fonts.alias` files in these directories:

```
/usr/lib/fmfonts
```

```
/usr/lib/X11/fonts/100dpi
```

```
/usr/lib/X11/fonts/75dpi
```

```
/usr/lib/X11/fonts/misc
```

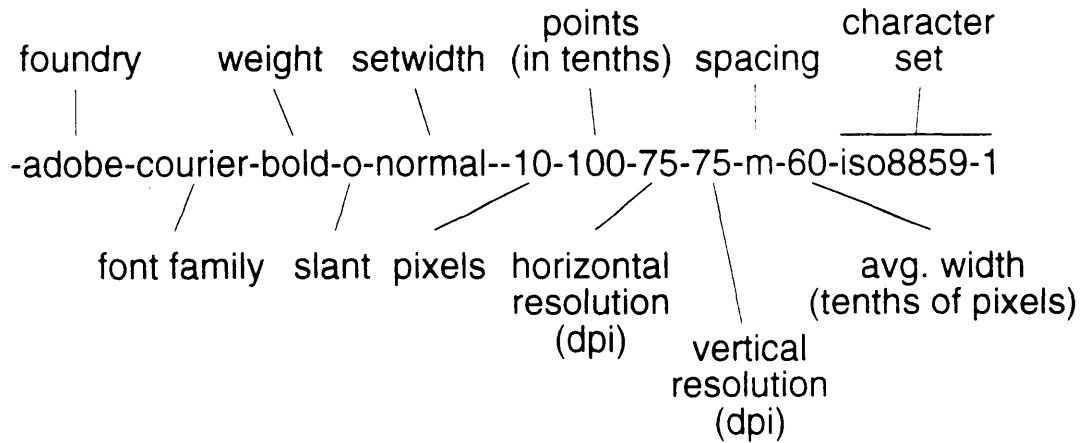
- to list all font names, use `xlsfonts`
- to examine a font, use:

```
xfontsel -print &
```

```
xfd -fn fontname &
```

Font Specification

- naming convention



- foundry: font supplier
 - font family: Courier, Helvetica, Times, Symbol, etc.
 - weight: medium or bold
 - slant: roman, oblique, or italics
 - spacing: monospace (fixed width) or proportional (variable)
 - character set: iso--International Standards Organization
- alias names are shorter

Resource Classes and Instances

- Classes and Instances
- Each component of a resource specification belongs to a *class*
- For example, `xclock`
- the color for the foreground, hands, and highlight are *instances* of the *class* `Foreground`

- Therefore:

```
xclock*foreground: HotPink
```

```
xclock*hands: HotPink
```

```
xclock*highlight: HotPink
```

is the same as:

```
xclock*Foreground: HotPink
```

- Class names always begin with a capital letter
- Instance names always begin with a lowercase letter

Resource Classes and Instances

- Precedence Rules--"The more specific, the better"
 - Instance names take precedence over class names
 - Tight bindings take precedence over loose bindings

- Therefore:

```
xclock*Foreground: HotPink
```

```
xclock*highlight: Green
```

- results in pink tick marks and hands and green highlights around the hands

Login Authentication Customization Exercise

Purpose:

This exercise will give you practice in customizing color and font resources.

Instructions:

Note: You must be the super-user to have access to the files to do this lab assignment.

1. Make a copy of the `/usr/lib/X11/xdm/Xresources` file. Edit the file. Change the color of the text or background. Use either a hexadecimal specification or a mnemonic name. Log out to see your changes.

Please note that this file is a **system-wide** resource. When done with the exercise, restore the original file.

2. * Edit the `Xresources` file again. Change the font of the text. Use `xlsfonts` to determine which fonts are available. Use `xfontsel` or `xfd` to preview chosen fonts.

Xstartup

- startup file invoked at login
- on IRIS, does not do much

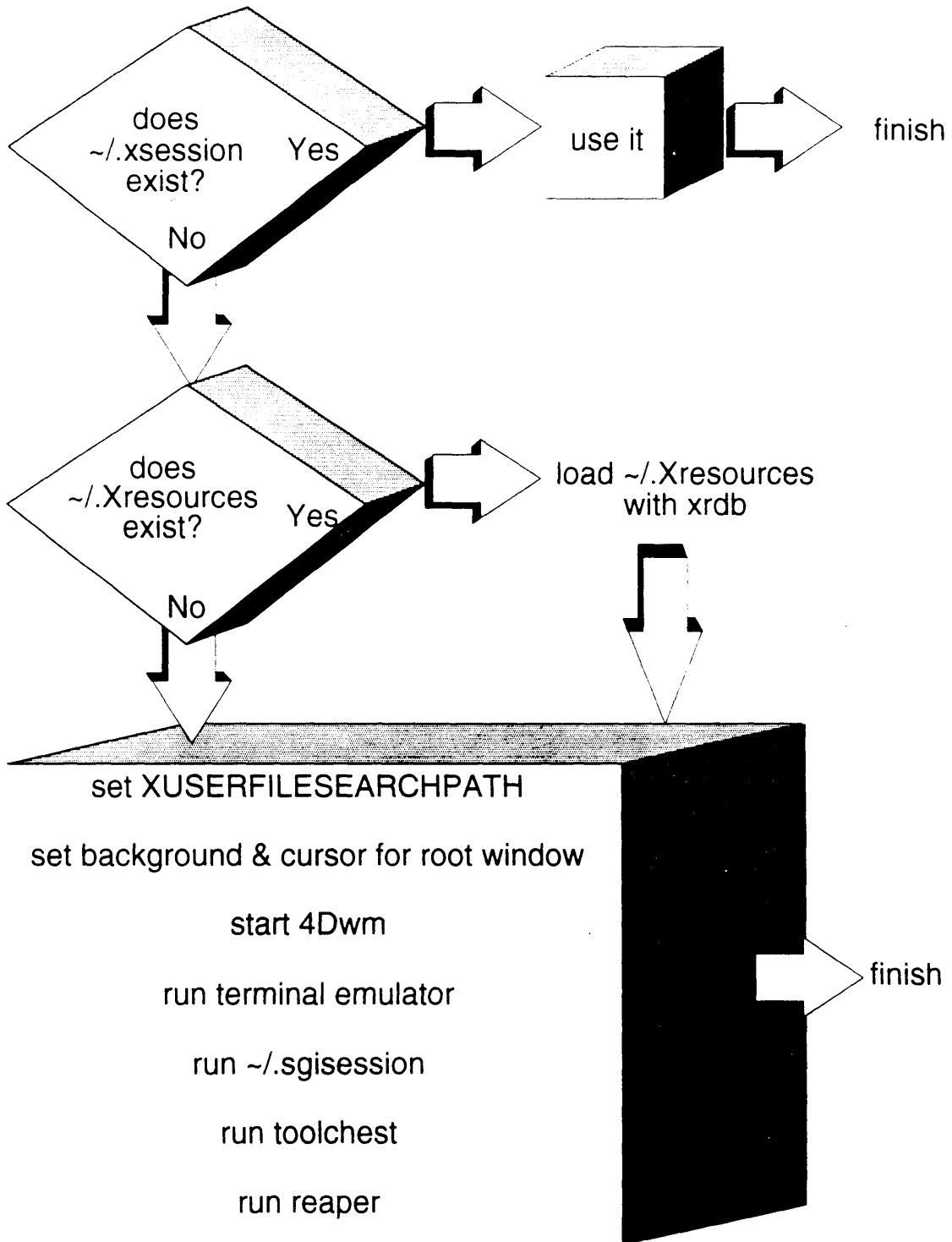
gamma correction

```
glGammaFile=/etc/config/system.glGammaVal
glGammaDefault="1.7"

if [ -x /usr/sbin/gamma ]; then
    if [ -r $glGammaFile ]; then
        glGammaVal=`cat $glGammaFile`
    else
        glGammaVal=$glGammaDefault
    fi

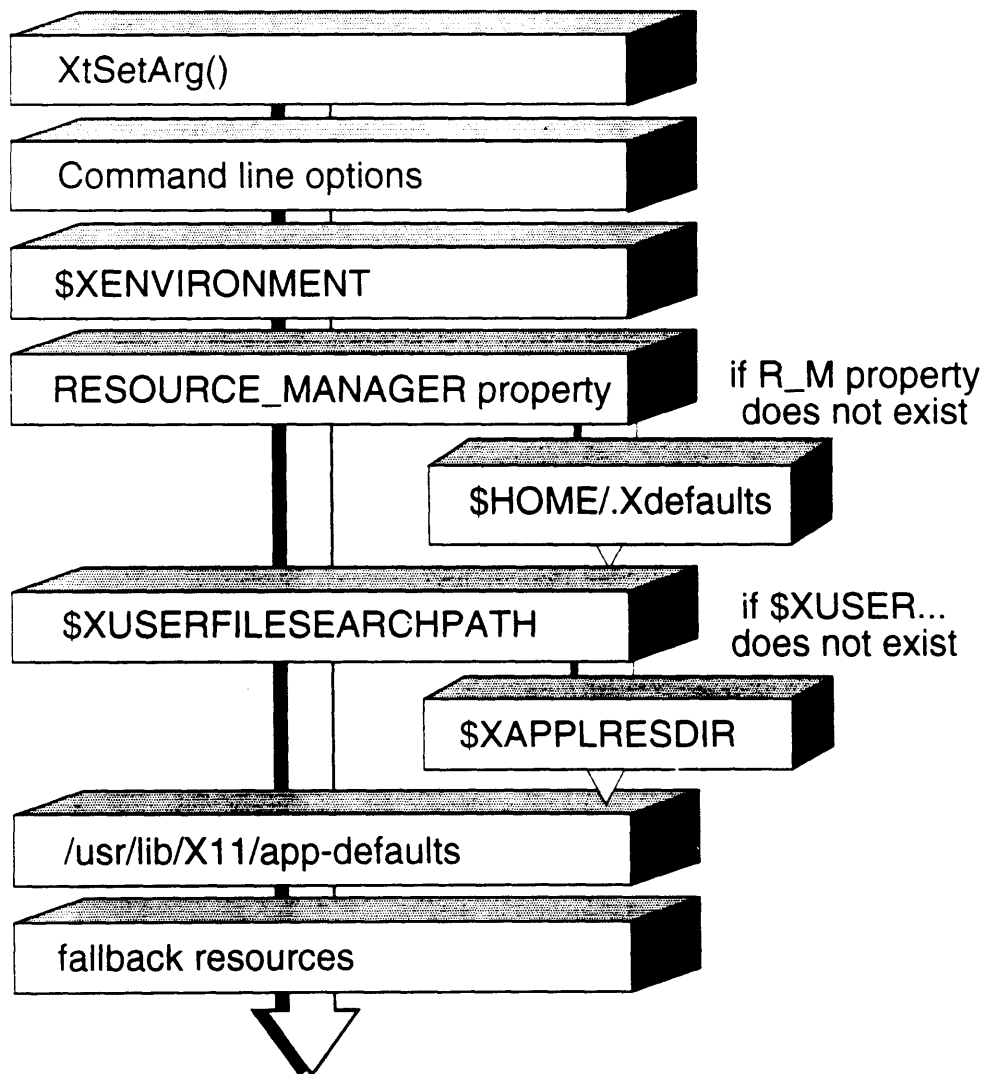
    screens=`/usr/bin/X11/xlistscrns`
    for screen in $screens
    do
        HOME=/ DISPLAY=$screen /usr/sbin/gamma $glGammaVal
    done
fi
```

Xsession Script



Resource Management

- Application resources can be altered without recompilation
- `4Dwm` and `xwsh` are typical X clients
- A typical X client would look for resource definitions in this sequence (the first change sets the resource):



Engineer's Handbook General Stuff

May 5, 1994

Open GL & X

access via: handbook opengl_and_x_1

access via: handbook opengl_and_x_2

access via: handbook opengl_and_x_3

probable data location: dist.wpd:/sgi/doc/swdev/opengl_and_x_[123]

OpenGL™ and X, Part 1: An Introduction

Mark J. Kilgard *
Silicon Graphics Inc.
Revision : 1.16

October 4, 1993

Abstract

The OpenGL™ graphics system is a high-performance, window system independent 2D and 3D graphics interface. The technology was developed by Silicon Graphics and is now controlled by the OpenGL Architecture Review Board. OpenGL's GLX extension integrates OpenGL with the X Window System. This article describes OpenGL's functionality and how it is used with X. A simple OpenGL program using Xlib is presented. OpenGL is compared and contrasted with PEX, a 3D graphics interface designed specifically for X. The two subsequent articles in this series describe how to integrate OpenGL with Xlib and Motif programs.

1 Introduction

The OpenGL™ graphics system is a powerful software interface for graphics hardware that allows graphics programmers to produce high-quality color images of 2D and 3D objects. The technology was developed by Silicon Graphics Inc. (SGI) and is the result of ten years of experience designing production software interfaces for a full spectrum of graphics hardware.

OpenGL is now controlled by an industry consortium known as the OpenGL Architectural Review Board (ARB) currently composed of Digital Equipment, IBM, Intel, Microsoft, and SGI. The interface is licensed to a large number of computer software and hardware vendors and OpenGL implementations are now appearing on the market.

This article is the first of a series of three articles explaining OpenGL to the users of the X Window System. This article introduces the reader to OpenGL's features.

*Mark graduated with B.A. in Computer Science from Rice University and is a Member of the Technical Staff at Silicon Graphics. He can be reached by electronic mail addressed to mjk@sgi.com

particularly how they apply to X. This section will introduce the reader to OpenGL's philosophy and history. Section 2 will explore OpenGL's rich feature set. Section 3 discusses OpenGL's integration with the X Window System via the GLX extension. Section 4 presents a simple OpenGL program for X. Section 5 compares and contrasts OpenGL to PEX, a 3D graphics interface designed specifically for X. Section 6 tells where to find more information about OpenGL.

The second article in the series will explain in more detail how to use OpenGL in conjunction with Xlib. The third article will describe how to use OpenGL with Motif.

1.1 Design Philosophy

To appreciate OpenGL it is useful to understand its design philosophy. OpenGL provides a layer of abstraction between graphics hardware and an application program. It is visible to the programmer as a set of routines consisting of about 120 distinct commands. Together these routines make up the OpenGL application programming interface (API). The routines allow graphics primitives (points, lines, polygons, bitmaps, and images) to be rendered to a frame buffer. Using the available primitives and the operations that control their rendering, high-quality color graphics images of 3D objects can be rendered.

The designers of OpenGL present the graphics system as a state machine [7] that controls a well-defined set of drawing operations. The routines that OpenGL supplies provide a means for the programmer to manipulate OpenGL's state machine to generate the desired graphics output. Figure 1 shows a simplified view of OpenGL's abstract state machine. Specifying OpenGL as a state machine allows consistent, precise specification and eliminates ambiguity about what a given operation does and does not do.

The model used for interpretation of OpenGL com-

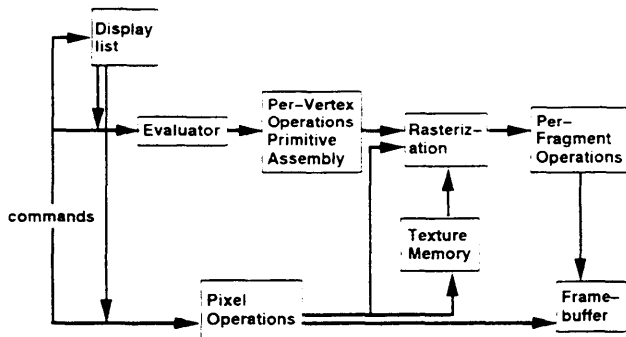


Figure 1: High-level, abstract OpenGL machine.

mands is *client-server*. This is an abstract model and does not demand OpenGL be implemented as distinct client and server processes. A client-server approach means the boundary between a program and the OpenGL implementation is well-defined to clearly specify how data is passed between the program and OpenGL. This allows OpenGL to operate over a *wire protocol* much as the X protocol operates but does not mandate such an approach.

The OpenGL specification is *window system independent* meaning it provides rendering functionality but does not specify how to manipulate windows or receive events from the window system. This allows the OpenGL interface to be implemented for distinct window systems. For example, OpenGL has been implemented for both the X Window System and Windows NT.

The specification which describes how OpenGL integrates with the X Window System is known as GLX. It is an extension to the core X protocol for communicating OpenGL commands to the X server. It also supports window system specific operations such as creating rendering contexts, binding those contexts to windows, and other window system specific operations.

GLX does not demand OpenGL commands be executed by the X server. The GLX specification explicitly allows OpenGL to render directly to the hardware if supported by the implementation. This is possible when the program is running on the same machine as the graphics hardware. This potentially allows extremely high performance rendering because OpenGL commands do not need to be sent through the X server to get to the graphics hardware.

Graphics systems are often classified as one of two types: procedural or descriptive. Procedural means the programmer is determining what to draw by issuing a specific sequence of commands. Descriptive means the programmer sets up a model of the scene to be rendered and leaves how to draw the scene up to the graphics system. OpenGL is procedural. In a descriptive system, the programmer gives up control of exactly how the scene is to be rendered. Being procedural allows the programmer a high degree of control to achieve the best performance. It is expected that descriptive graphics systems will be implemented us-

ing OpenGL as a low level interface. SGI's Inventor toolkit [8] is one example of such a descriptive graphics system.

An overriding goal of OpenGL is to allow the construction of portable and interoperable 3D graphics programs. For this reason, OpenGL's rendering functionality must be implemented in its entirety. This means all the complex 3D rendering functionality described later in the article can be used with any OpenGL implementation. Previous graphics standards often allowed subsetting; too often the result was programs that could not be expected to work on distinct implementations.

1.2 History of OpenGL

A brief history of OpenGL explains how OpenGL came to be and what inspired its development. OpenGL is the successor to a graphics library known as IRIS GL (GL stands for graphics library) developed by SGI as a hardware independent graphics interface for use across a full line of graphics workstations. IRIS GL [4] is used by more than 1,500 3D graphics applications. IRIS GL was developed over the last decade and has been implemented on numerous graphics devices of varying sophistication.

OpenGL is not backward-compatible with IRIS GL. OpenGL has removed dated IRIS GL functionality or replaced it with more general functionality. The routines and symbols comprising the OpenGL API have been named to avoid name space conflicts (all names start with either `gl` or `GL`). The window system dependent portions of IRIS GL are not part of OpenGL. What has been preserved is the spirit of the API. OpenGL retains IRIS GL's ability to render 3D objects quickly and efficiently.

OpenGL has been proposed as a graphics standard to bring 3D graphics programming into the mainstream of applications programming. For this reason, the OpenGL ARB was formed. The ARB licenses OpenGL and directs further development. Currently, over 20 companies have licensed OpenGL and intend to release or have already released commercial implementations. Numerous universities have also licensed OpenGL.

2 OpenGL's Functionality

OpenGL is not a high-level 3D graphics interface. When you build a graphics program using OpenGL, you start with a few simple primitives. The sophistication comes from combining the primitives and using them in various modes. Figure 2 shows the available geometric primitives. Notice the ordering of the vertices, in particular for primitives such as the `GL_TRIANGLE_STRIP` and the `GL_TRIANGLE_FAN`.

To begin a primitive, the `glBegin` routine passes in the primitive type as an argument. Then a list of vertex coordinates are given. OpenGL has a family of routines to specify vertex coordinates. All the routines begin with

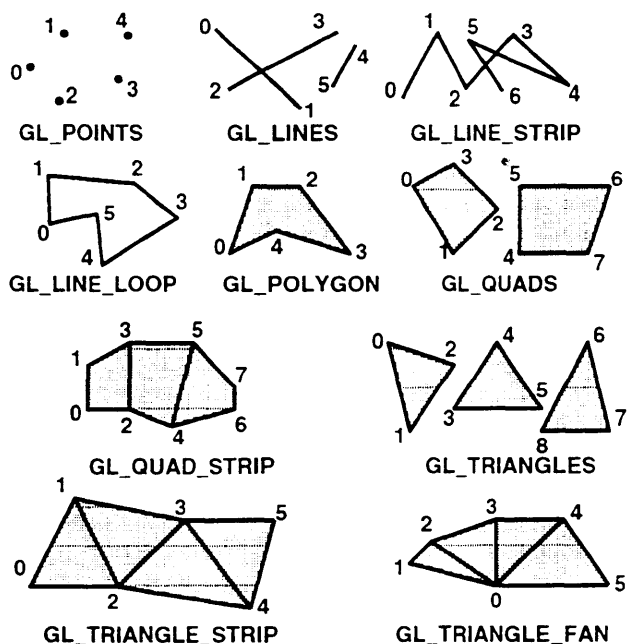


Figure 2: OpenGL Geometric Primitives.

the name `glVertex`. The suffix to a specific `glVertex` routine tells the type and number of coordinates for the vertex. For example, `glVertex3f` indicates a three coordinate vertex consisting of floating point values is to be generated.

An OpenGL primitive is completed by calling `glEnd`. Along with the coordinates of each vertex, per-vertex information about color, material, normals, edge drawing, and texturing can be specified between a `glBegin` and `glEnd`. Figure 3 shows an example of how a polygon might be generated. Notice how `glColor3f` is used to change the current color. Each vertex is drawn according to the current color.

```

glShadeModel(GL_SMOOTH);
glBegin(GL_POLYGON); /* pentagon */
glColor3f(0.0, 1.0, 0.0); /* green */
glVertex3f(0.0, 1.0, 0.0);
glVertex3f(0.7, 1.0, 0.0);
glColor3f(0.0, 0.0, 1.0); /* blue */
glVertex3f(1.4, 0.6, 0.0);
glVertex3f(1.4, 0.4, 0.0);
glVertex3f(0.0, 0.0, 0.0);
glEnd();

```

Figure 3: Example of generating a 3D polygon with smooth shading between vertices.

2.1 Observations About Primitives

OpenGL tends to be function call intensive. There is not a complex `RenderPolygonWithGratuitousArguments` command. Instead primitives are constructed by calling multiple OpenGL routines. Calling multiple routines gives the program flexibility and control over the primitives generated.

OpenGL is flexible about what format information is passed to it. For example, the `glVertex3i` accepts integers while `glVertex3f` and `glVertex3d` take single and double precision floating point respectively. It is very advantageous for OpenGL to have several basically identical routines which accept different data types. It allows the programmer the flexibility to decide how to store the data. A programmer whose data is in integer format does not want to convert it to floating point to pass it to the graphics system. And another programmer does not want to convert floating point data into integers. Conversions between data types can be expensive. High performance graphics hardware can be designed to accept multiple data formats and totally off load the task of format conversion from the host processor.

You can start to see why it makes sense to consider OpenGL as a state machine. Commands such as `glColor3f` change the state of the current color. Subsequent vertices use the current color. `glBegin` puts OpenGL into a state to start drawing the specified primitive. The multiple `glVertex` routines load up one at a time the vertices for a given primitive. Nearly all of OpenGL's state that can be set by the programmer can also be queried by the programmer. The `glGetFloatv(GL_CURRENT_COLOR, &float_array)` call, for example, will retrieve the setting of the current color.

2.2 Two Color Models

OpenGL has two different color modes: *RGBA* and *color index*. The `glColor3f` call has already been demonstrated but not explained. This call assumes OpenGL's *RGBA* color mode. The routine takes three floating point parameters between 0.0 and 1.0 which specify the degree of red, green, and blue for the current color. For X users, *RGBA* roughly corresponds to the *TrueColor* visual type while *color index* corresponds to *PseudoColor*. The color mode is fixed for a given window the same way X windows are created with a single, fixed visual.

You should be able to guess that the RGB in *RGBA* stands for red, green, and blue. The A may be unfamiliar. It stands for *alpha*. The alpha value is used when two colors are to be averaged together for blending operations. Alpha represents the opacity of the color. 1.0 is totally opaque while 0.0 is totally transparent. For example, one could use alpha to render a scene with green glass. The frame buffer can support an alpha component which al-

lows alpha values to be stored. Each pixel in the frame buffer would have an associated alpha value. The alpha value is not visible on the display. It is just used to determine how a pixel to be drawn is blended with the current pixel value in the frame buffer. The `glAlphaFunc` and `glBlendFunc` routines control precisely how alpha buffering operates. The `glColor4f` command is a variation on `glColor3f` which takes a fourth parameter specifying alpha (`glColor3f` implicitly sets alpha to 1.0).

RGBA supports a tri-linear palette for the full range of colors, making it very useful for rendering realistic scenes. OpenGL supports lighting, fog, and smooth shading most effectively in RGBA mode. Since a lot of hardware has limited color resolution, an application can request OpenGL use dithering for better color resolution (at the expense of spatial resolution).

Many modes in OpenGL, such as dithering, are enabled and disabled using the `glEnable` and `glDisable` commands. For example, dithering is enabled by calling `glEnable(GLDITHER)`. Then drawing would be done with dithering enabled. You can think of `glEnable` and `glDisable` as ways to affect the operation of the OpenGL state machine.

The color index model assumes a readable and writable linear colormap. Usually window systems specify how colors are allocated and arranged so OpenGL does not have any specific routines to allocate colors. For example in X, an Xlib color allocation routine such as `XAllocColor` would be used. The `glIndex` family of routines is used to set the current color index. The advantage of color index is that the color of a given pixel value can be changed. There is a level of indirection between the pixel values in the frame buffer and the colors on the screen.

2.3 Ancillary Buffers

The drawing surface for OpenGL is generically referred to as the *frame buffer*. In actuality, the frame buffer might be a window created by your computer's window system or an in-memory data structure (like an X pixmap). OpenGL's frame buffer can logically be considered a set of buffers. A buffer is logically just a two-dimensional array of values. The most important buffer is the image buffer which contains the actual color information and possibly the alpha component but there are also other types of buffers. A window system might support multiple frame buffer configurations, each supporting different types of buffers. Multiple windows of different configurations can be displayed at one time though a single window has a fixed frame buffer configuration. In X, visuals are overloaded to also describe supported OpenGL frame buffer configurations.

The non-image buffers are often referred to as *ancillary* or helper buffers. While they do not contain the image itself, they can be essential in properly generating the im-

age.

2.3.1 The Depth Buffer

For 3D graphics, the *depth buffer* (also commonly referred to as a Z buffer) is nearly essential. While the screen only has two dimensions, 3D graphics seeks to simulate a third. When 3D primitives are rendered, they are rasterized into a collection of *fragments*. Each fragment corresponds to a single pixel and includes color, depth, and sometimes texture-coordinate values. The X and Y values for a fragment determine where on the screen the pixel should appear. A fragment's Z value or depth is used to determine how "near" the fragment is. When the depth buffer is enabled, the fragment is drawn only if its Z value is "nearer" than the current Z value for the corresponding pixel in the depth buffer. When the fragment is drawn into the frame buffer, its Z value replaces the previous value in the depth buffer. Normally, when the scene starts to be rendered, the entire depth buffer is cleared to the "farthest" value. As a 3D scene is rendered, the depth buffer automatically sorts the fragments being drawn so only the nearest fragment at each pixel location gets drawn. Things logically behind other things are automatically eliminated from the scene. This is the normal use for a depth buffer, although other uses are possible.

2.3.2 The Stencil Buffer

Another buffer supported by OpenGL is the stencil buffer. Like the depth buffer, the stencil buffer can be used to eliminate certain pixels from being drawn. The stencil buffer acts in much the same way as a cardboard stencil used with a can of spray paint. You can "draw" values into the stencil buffer using the normal OpenGL rendering primitives. Then a stencil test can be defined and stenciling enabled.

One possible use of the stencil buffer is in a flight simulator. Imagine that the view outside the plane is to fit into an irregularly shaped windshield. The rendering of the view outside the plane should not interfere with the rendering of the instruments "inside" the cockpit. If the windshield area is drawn in the stencil buffer, then a stencil test can be set up to make sure the windshield view is only drawn where the windshield stencil has been drawn. There are many other uses for stencil buffers.

2.3.3 The Accumulation Buffer

Yet another buffer supported by OpenGL is the accumulation buffer [2] which can be used for antialiasing, motion blur, simulating photographic depth of field, and rendering soft shadows from multiple light sources. You do not render directly into the accumulation buffer. Instead, you render a series of images, accumulating each into the accumulation buffer, combining the images. Then the accu-

mulated image can be dumped back into the image buffer for display. The effect is much the same as the one a photographer gets from multiply exposing a piece of film.

Motion blur is one use. Imagine drawing a scene several times with each frame corresponding to a slightly different point in time. By accumulating the frames (with decayed intensity for earlier frames), you can achieve an effect similar to motion blur since still objects are sharp but moving objects are blurred by their accumulation in slightly differing locations.

2.3.4 Double Buffering

Double buffering means having two sets of image buffers, one *front* visible buffer and another *back* non-visible buffer. Unlike simple 2D, 3D images may take substantially more time to generate. And depth, alpha, and accumulation buffers all mean that the image being drawn at any moment might be quite different from the final image. It would be quite distracting for the viewer to see each scene while it was "under construction" and would destroy the illusion of a smoothly animated scene. Double buffering allows for one image to be rendered while another is being displayed.

OpenGL supports this notion. The `glDrawBuffer` routine can be used to determine to what buffer primitives should be drawn. A window system specific routine is available to make the back buffer visible.

Double buffering is often achieved by rendering the non-visible image buffer into memory and then quickly copying the buffers contents to screen memory. A better alternative is to build hardware that actually supports two sets of image buffers. Then the cost of a buffer swap can be extremely low since no data has to be copied. Instead, the video controller can just change to scanning image pixels out of the other buffer.

2.3.5 Stereo

Stereo is similar to double buffering in that more than one image buffer is supported. Instead of front and back, left and right are provided (though generally stereo and double buffering are combined, requiring four image buffers). Special stereo video hardware alternates between scanning out the left and right buffers every screen refresh. Goggles synchronized with the vertical refresh of the screen alternately open and close LCD shutters so the left eye sees the left frame and the right eye sees the right frame. By carefully drawing the scene twice with slightly different perspective into the left and right buffers, the viewer experiences an optical illusion of 3D.

While double buffering is common on graphics workstations, stereo requires special hardware and tends to be rather expensive so many OpenGL implementations may not support stereo.

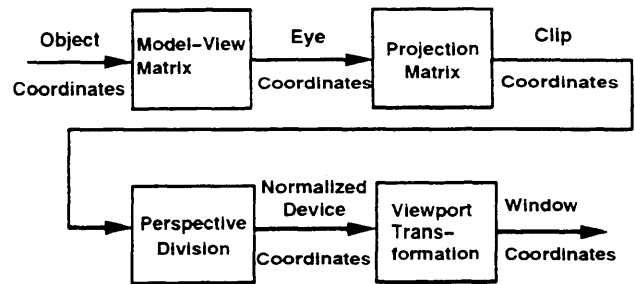


Figure 4: Stages of vertex transformation.

2.4 Viewing

One of the most difficult initial hurdles in learning 3D graphics programming is how to properly set up a view. It is very easy to get a black screen because the viewing for the scene is not properly initialized.

3D computer graphics uses matrix transformations to properly orient, view, clip, and map the model to the screen. OpenGL's various stages in mapping vertices in object coordinates into pixels in window coordinates are pictured in Figure 4.

An OpenGL programmer is responsible for loading the *modelview* and *projection* matrices. The *modelview* matrix determines how the vertices of OpenGL primitives are transformed to eye coordinates. The *projection* matrix transforms vertices in eye coordinates to clip coordinates.

A number of OpenGL routines deal with manipulating these matrices. The `glMatrixMode` routine is called with an argument of `GL_MODELVIEW` or `GL_PROJECTION` to determine what is the current modifiable matrix. Then `glLoadIdentity` may be called to set the currently modifiable matrix to the identity matrix. Then routines such as `glRotatef`, `glTranslatef`, and `glScalef` may be called to manipulate the currently modifiable matrix. `glLoadMatrixf` loads a specific matrix and `glMultMatrixf` multiplies the current matrix by some specified matrix and store the result as the current matrix. Understanding exactly how these different commands should be properly used is beyond the scope of this article.

The final step in establishing a view of your model is the *viewport* transformation. It determines how the scene gets mapped onto the computer screen. The `glViewport` routine specifies the rectangle in the window of into which the final image is to be mapped. By default, the entire window is used. `glViewport` is commonly invoked when an OpenGL window is resized.

2.5 Other Features

There are a large number of OpenGL features worth mentioning but their full introduction is beyond the scope of this article.

Just specifying 3D primitives and determining how to

map them to the screen is not enough to achieve realistic images. OpenGL also supports a number of lighting models that simulate the effects of lighting on primitives. Light sources can be defined and material properties can be specified to achieve realistic lighting effects.

So far polygons have been described as basically shaded or flat surfaces. But OpenGL allows polygons to be rendered which have a 1D or 2D texture mapped onto the polygon. For example, the surface of a desk could be textured with a wood grain image for greater realism. Texture mapping can greatly enhance the visual impact of a scene without increasing the geometric complexity.

Polygons are the basic primitive for much 3D rendering but OpenGL also supports bitmaps and images. And OpenGL provides evaluator commands for the efficient rendering of curves and surfaces.

Because 3D rendering eventually appears on a screen with limited resolution, OpenGL provides various techniques to eliminate “jaggies” resulting from aliasing problems. OpenGL provides antialiasing support for points, lines, and polygons. Techniques using the alpha, stencil, or accumulation buffers can also be used to minimize aliasing problems.

Computer images often appear unrealistically sharp and well-defined. OpenGL supports “fog” to provide an effect that simulates atmospheric effects. Haze, mist, smoke, and pollution can all be simulated. When fog is enabled, objects farther away begin to fade into the specified fog color.

Users of 3D want to do more than just see 3D images; they want to interact with them. OpenGL supports a *selection* mechanism that allows the user to pick an object or objects drawn to a certain region of the screen. And *feedback* can be used to obtain the results of rendering calculations.

Often a sequence of OpenGL commands are rendered repeatedly. OpenGL supports display lists which allow commands to be compiled for later execution. Display lists can even call other display lists allowing hierarchies of display lists. For networked 3D applications, display lists can greatly minimize the protocol bandwidth needed and increase performance. The `glNewList` and `glEndList` are used to create a display list. A created display list can be executed using the `glCallList` routine.

One thing to keep in mind about OpenGL is that the features described above are not isolated functionality. Each feature can be combined with others for advanced effects. For example, lighting, fog, display lists, texture mapping, and double buffering can all be used simultaneously.

2.6 The GLU Library

The core OpenGL API focuses on rendering functionality but there are a number of tasks common to many

3D programs that are not strictly related to rendering. For this reason, the OpenGL standard also provides the OpenGL Utility Library (GLU). The GLU routines (all prefixed with `glu`) fall into one of the following areas:

- Manipulating images for use in texturing.
- Transforming Coordinates.
- Polygon tessellation.
- Rendering spheres, cylinders, and disks.
- Non-Uniform Rational B-Spline (NURBS) curves and surfaces.
- Describing errors.

The GLU is a separate but standard library that any OpenGL application can use.

3 OpenGL’s X Support

GLX is an official part of the OpenGL standard for supporting the X Window System. It provides additional routines (prefixed by `glX`) for interfacing OpenGL with X. It also defines a wire protocol for supporting OpenGL as an X server extension. The GLX wire protocol allows workstations from different vendors to interoperate using 3D graphics the same way the X protocol provides 2D graphics interoperability. Some of the issues about integrating X and OpenGL are discussed by Karlton [3].

GLX allows rendering into X windows and pixmaps. An X server can support different visuals to describe the different types of windows supported by the server. For the core X protocol, a visual specifies one (or more) depths for the frame buffer and how pixel values are mapped to colors on the screen. X treats a drawable as basically a 2D array of pixels, but OpenGL has a much more sophisticated view of a drawable’s frame buffer capabilities. GLX overloads the core X notion of a visual by associating additional information about OpenGL’s frame buffer capabilities. In addition to an image buffer, OpenGL supports various types of ancillary buffers. For example, a window might also have a stencil buffer and a depth buffer. Modes such as stereo and double buffering are also supported. Multiple different frame buffer configurations can be supported by a single X server by exporting multiple visuals.

All OpenGL implementations for the X Window System must support at least one RGBA visual and at least one color index visual. Both visuals must support a stencil buffer of at least 1 bit and a depth buffer of at least 12 bits. The required RGBA visual must have an accumulation buffer. The alpha component of the image buffer is not required for the RGBA visual (but input alpha is still used in all rendering calculations). Many implementations will support many more than two visuals.

The GLX API supplies two routines, `glXGetConfig` and `glXChooseVisual`, to help programmers select an appropriate visual. Once the appropriate visual is selected, call `XCreateWindow` with the selected visual to create the window.

GLX supports off-screen rendering to pixmaps. First create a standard X pixmap of the desired depth using `XCreatePixmap`. Then call `glXCreateGLXPixmap` with the desired OpenGL visual. A new drawable of type `GLXPixmap` is returned which can be used for drawing OpenGL into the pixmap.

To render using OpenGL, an OpenGL rendering context must be created. The `glXCreateContext` routine creates such a context. An option to `glXCreateContext` allows the programmer to specify that direct rendering to the hardware should be done if supported by the implementation.

Before rendering, a rendering context must be bound to the desired drawable using `glXMakeCurrent`. OpenGL rendering commands implicitly use the current bound rendering context and one drawable. Just as a program can create multiple windows, a program can create multiple OpenGL rendering contexts. But a thread can only be bound to one rendering context and drawable at a time. Once bound, OpenGL rendering can begin. `glXMakeCurrent` can be called again to bind to a different window and/or rendering context.

The GLX stream of commands is considered distinct from the stream of X requests. Sometimes you may want to mix OpenGL and X rendering into the same window. If so synchronization can be achieved using the `glXWaitGL` and `glXWaitX` routines.

To swap the buffers of a double buffered window, `glXSwapBuffers` can be called. X fonts can be converted into per-glyph OpenGL display lists using the `glXUseXFont` routine.

4 A Simple Example Using X

Appendix A contains the C source code for a simple OpenGL program. This example demonstrates what is involved when programming OpenGL with Xlib. The program creates a window and draws a 3D cube (missing two faces) and allows the user to rotate the cube around the X, Y, and Z axes using the mouse buttons.

Besides demonstrating how to properly establish an X window for OpenGL rendering, the example demonstrates the use of double buffering, display lists, and establishing the proper viewing parameters.

4.1 Initialization

The following describes the steps involved in setting up a window to render OpenGL into it. The numbers listed

correspond to numbers in the comments of the OpenGL program in Appendix A.

1. As in all X programs, `XOpenDisplay` should be called to open a connection to the X server.
2. Make sure the OpenGL GLX extension is supported by the X server.
3. Before creating the window, the program needs to select an appropriate visual. The GLX routine `glXChooseVisual` makes it easy to find the right visual. In the example, an RGBA (and TrueColor) visual with a depth buffer is desired and if possible, it should support double buffering.
4. Create an OpenGL rendering context by calling `glXCreateContext`.
5. Create a window with the selected visual. Most X programs always use the default visual but OpenGL programmers will need to be comfortable with using visuals other than the default. `XCreateWindow` is called.
6. Bind the rendering context to the window using `glXMakeCurrent`. Subsequent OpenGL rendering commands will use the current window and rendering context.
7. To display the window, `XMapWindow` should be called.
8. Set the desired OpenGL state. In this example, depth buffering is enabled, the clear color is set to black, and the 3D viewing volume is specified.
9. Begin dispatching X events.

Button presses change the angle of rotation for the object to be viewed and cause a redraw. Expose events also cause a redraw (without changing the rotation). Window resizes call `glViewport` to ensure the OpenGL viewport corresponds to the maximum dimensions of the window.

4.2 Scene Update

The `redraw` routine does all the OpenGL rendering. The code is slightly complicated by constructing a display list to draw the cube. The first time `redraw` is called, `glNewList` and `glEndList` are used to construct a display list for the object to be rendered. Subsequent redraws call the display list instead of rendering the object each time.

Creating a display list potentially allows improved performance since the commands can be compiled for faster execution. In the case of OpenGL across a network, display lists save having to send all the commands to render the scene whenever the window is redrawn.

The commands to render the object consist of four 3D rectangles of different colors. Notice the rectangles

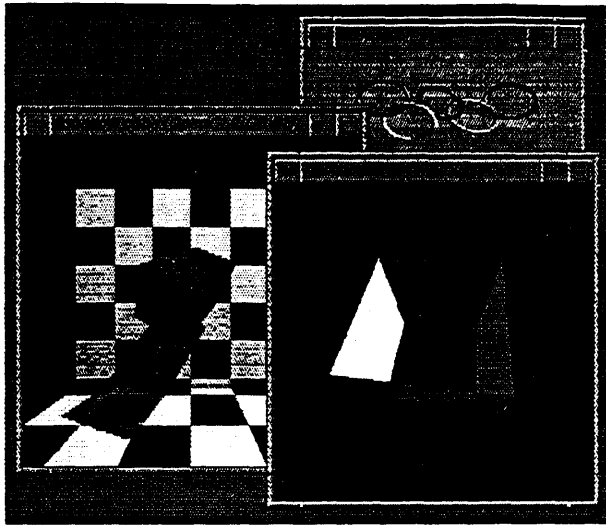


Figure 5: Screen snapshot of `glxsimple` with two other simple 3D OpenGL programs.

are generated by first calling `glBegin(GL_QUADS)` and ended with `glEnd`. Each rectangle is specified by four `glVertex3f` calls that specify the four vertexes of each rectangle. The `glColor3f` invocations tell what color each rectangle should be rendered. Figure 5 shows how the program looks.

If the window is double buffered, `glXSwapBuffers` is called on the window. By default rendering to double buffered windows takes place in the non-visible back buffer. Swapping buffers will quickly swap the front and back buffers avoiding any visual artifacts (the contents of the back buffer should be considered undefined after a swap). In effect, the rendering of each frame can be done “behind the scenes.”

`glFlush` is called to ensure that the OpenGL rendering commands are actually sent to the graphics hardware. A flush is implicitly done by `glXSwapBuffers` so the `glFlush` is only needed explicitly in the single buffered case.

5 Comparing OpenGL to PEX

OpenGL is not the only means for extending the X Window System to support 3D. PEX [9] is an extension developed by the X Consortium to add 3D capabilities to X. The currently available version is 5.1. A future release known as PEX 6.0 is intended to address many of PEX 5.1’s problems but its specification is not yet finalized. An in-depth analysis of PEX 5.1 and OpenGL 1.0 is presented by Akin [1]. Here we discuss some of the most prominent distinctions.

5.1 Subsets and Baselines

One thing that makes PEX difficult to compare to OpenGL is that PEX allows much of its functionality to be optionally implemented. PEX classifies its functionality into one or more of three subsets: the immediate mode subset, the structure subset, or the PHIGS workstation subset. (PHIGS is a 3D graphics standard and stands for Programmer’s Hierarchical Interactive Graphics System.) The PEX specification *explicitly* allows implementations to support one, two, or all three subsets. The result is that an application cannot depend on any given PEX server to supply the subset functionality the application might depend on. This problem is commonly referred to as “sub-setting.”

OpenGL mandates that that all its rendering functionality be supported. Even advanced features such as depth buffering, fog, lighting, anti-aliasing, and texturing must be supported in *all* implementations.

But still all OpenGL implementations are not totally identical. Rendering functionality is not a complete picture of OpenGL’s capability. Rendering performance will depend on the implementation. And frame buffer capabilities will vary between implementations. Different depths of ancillary buffers will be supported: stereo and double buffering hardware may or may not actually be present: a frame buffer may or may not support the alpha component. But despite the possibility for variation, OpenGL for the X Window System does *mandate* that two visuals (one RGBA, the other color index) will be present with frame buffer capabilities sufficient for most common 3D applications. Stencil and depth buffers must be supported for the two required visuals. And an accumulation buffer must be supported for the RGBA visual. These required visuals guarantee all OpenGL implementations have a standard baseline of both rendering and frame buffer functionality which applications can rely on being present.

5.2 Programming Interfaces

There is an essential difference between PEX and OpenGL in how the two graphics systems are specified. OpenGL is fundamentally specified as an application programming interface. Like the X Window System, the fundamental specification for PEX is a wire protocol.

In PEX the choice of programming interface is left to the programmer. In X11R5 a PHIGS style API was supplied but this API for PEX has not gained much acceptance. Currently the PEX community is standardizing the PEXlib API which more readily exposes the wire protocol. But PEX implementation dependencies are also exposed, leaving the programmer to work around functionality missing due to subsetting in PEX implementations.

With OpenGL there is a single API which promises to be standard even across differing window systems (such as X and NT) and the full functionality of the API is available in

all OpenGL implementations. The GLX specification does provide a wire protocol for network-transparent operation but the wire protocol is not the fundamental specification of OpenGL.

5.3 Rendering Functionality

PEX and OpenGL both support basic 3D rendering functionality. Both allow 3D and 2D lines and polygons to be rendered using standard modeling and viewing methods. PEX (depending on the implementation) and OpenGL also support picking, lighting, depth cueing, and hidden line and surface removal.

There are a number of sophisticated rendering features supported by OpenGL that PEX completely lacks. Alpha blending, texture and environment mapping, antialiasing (though some PEX implementations supply it as a non-standard extension), accumulation buffer methods, and stencil buffering are all missing from PEX.

PEX does support features not available in OpenGL. PEX has extensive text support for stroke fonts which are fully transformable in 3D. B-Spline surfaces and curves are supported directly by PEX while OpenGL supports NURBS functionality via routines which are part of the GLU library. PEX can support cell arrays but the functionality is seldom implemented. Markers and quadrilateral meshes are supported by PEX as a rendering primitive; neither are supported as primitives by OpenGL. PEX supports self-intersecting contours and polygon lists with shared geometry, while OpenGL does not.

Double buffering and stereo support are built into OpenGL (though not all implementations will support double buffered or stereo visuals) while PEX relies on proprietary support or not yet nonstandardized X extensions for double buffering and stereo.

5.4 Display Lists

PEX and OpenGL both provide a means to store commands for later execution. In PEX (for implementations that support the structure or PHIGS workstation subsets), editable *structures* can be created and edited. A structure contains graphics primitives such as a polygon. Structures may also contain calls to execute other structures allowing them to be arranged in a hierarchical fashion. PHIGS supports structures so PEX does so too. Entire 3D models can be constructed out of a hierarchy of structures so that a redraw requires only retraversing the structure hierarchy.

OpenGL does not support structures in the same way PEX does. Instead *display lists* can be constructed which contain sequences of OpenGL commands. Like structures, a display list can contain a command to execute another display list, effectively allowing display lists to be combined into arbitrary networks. Unlike structures, OpenGL display lists are *not* editable. Once one is created, it

is sealed and cannot be changed (except by destroy and recreating it). This write-only nature allows optimizations to be performed on display lists unavailable to structures. The commands in the display list can be optimized for faster execution.

Even though display lists cannot be edited, this should not be considered a disadvantage. The same effect as editing can be achieved by rewriting display lists called by other display lists.

Display lists and structures both minimize the amount of transfer overhead when running PEX or OpenGL over a network since the commands in a structure or display list can be executed repeatedly by only calling the display list by name. The commands themselves need to be transferred across the wire only once.

5.5 Portability

While PEX was designed to be vendor-independent and portable, the subsetting allowed by the PEX standard allows implementations of greatly varying functionality to claim to be "standard" PEX implementations. The fact that PEX explicitly allows multiple subsets perhaps indicates the PEX standard may be too large to implement fully and completely in a timely fashion. Anyone who has been disappointed by the functionality of the X11R5 sample implementation understands the problem.

OpenGL does not allow any subsetting of rendering functionality and therefore can expect much greater application portability. The need for interoperability testing for OpenGL is greatly reduced because OpenGL demands more consistent implementations.

Neither OpenGL nor PEX is *pixel exact*. This means neither specification is completely explicit about what pixels must be modified by each rendering operation (the core X protocol is largely pixel exact). Pixel exactness is not a totally desirable feature for 3D since much 3D graphics is done with floating point where numerical errors make exactness nearly impossible. But the OpenGL specification is much more rigorous than PEX about what is considered conformant behavior. Not only does this make conformance test design easier, but OpenGL programmers can have high confidence their scene will be rendered accurately on all compliant OpenGL implementations.

The OpenGL release kit includes a suite of conformance tests to verify rendering accuracy. No comprehensive test suites are yet available to validate PEX implementations.

5.6 Window System Dependency

PEX is very tightly coupled to the X Window System. Not only was it designed in the context of X but its semantics depend on X notions of drawables, events, and execution requirements.

But X is not the only significant window system on the market. For this reason, OpenGL was designed to be window system independent. This means its API can also be used with Windows NT and future window systems. Application developers wishing to develop 3D applications for both X and Windows machines will appreciate having a consistent model for 3D across the two window systems.

6 Finding Out More

The best place to find more information about graphics programming using OpenGL is the OpenGL Technical Library published by Addison-Wesley. Currently available is the OpenGL Reference Manual [6] and the OpenGL Programming Guide [5]. The first volume contains complete descriptions of all the OpenGL routines including the GLU and GLX routines. The second volume is an excellent introduction to OpenGL including all its advanced rendering features.

Those with Internet access can obtain OpenGL documentation and sample program source code by using anonymous `ftp` to `sgi.com`. PostScript documentation for all the routines that are part of the OpenGL, GLU, and GLX APIs may be obtained. Example code from the OpenGL Programming Guide (including the aux library) is also available.

Of course the best way to learn OpenGL is to program with it. Systems supporting OpenGL are currently shipping from a number of workstation hardware and software vendors. Check with your vendor for availability.

A glxsimple.c

```
1 /* compile: cc -o glxsimple glxsimple.c -lGL -lX11 */
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <GL/glx.h>          /* this includes the necessary X headers */
5 #include <GL/gl.h>

6 static int snglBuf[] = {GLX_RGBA, GLX_DEPTH_SIZE, 16, None};
7 static int dblBuf[] = {GLX_RGBA, GLX_DEPTH_SIZE, 16, GLX_DOUBLEBUFFER, None};

8 Display      *dpy;
9 Window       win;
10 GLfloat      xAngle = 42.0, yAngle = 82.0, zAngle = 112.0;
11 GLboolean    doubleBuffer = GL_TRUE;

12 void
13 fatalError(char *message)
14 {
15     fprintf(stderr, "glxsimple: %s\n", message);
16     exit(1);
17 }

18 void
19 redraw(void)
20 {
21     static GLboolean  displayListInited = GL_FALSE;

22     if (displayListInited) {
23         /* if display list already exists, just execute it */
24         glCallList(1);
25     } else {
26         /* otherwise compile and execute to create the display list */
27         glNewList(1, GL_COMPILE_AND_EXECUTE);
28         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
29         /* front face */
30         glBegin(GL_QUADS);
31         glColor3f(0.0, 0.7, 0.1);      /* green */
32         glVertex3f(-1.0, 1.0, 1.0);
33         glVertex3f(1.0, 1.0, 1.0);
34         glVertex3f(1.0, -1.0, 1.0);
35         glVertex3f(-1.0, -1.0, 1.0);
36         /* back face */
37         glColor3f(0.9, 1.0, 0.0);     /* yellow */
38         glVertex3f(-1.0, 1.0, -1.0);
39         glVertex3f(1.0, 1.0, -1.0);
40         glVertex3f(1.0, -1.0, -1.0);
41         glVertex3f(-1.0, -1.0, -1.0);
42         /* top side face */
43         glColor3f(0.2, 0.2, 1.0);     /* blue */
44         glVertex3f(-1.0, 1.0, 1.0);
45         glVertex3f(1.0, 1.0, 1.0);
46         glVertex3f(1.0, 1.0, -1.0);
47         glVertex3f(-1.0, 1.0, -1.0);
48         /* bottom side face */
49         glColor3f(0.7, 0.0, 0.1);     /* red */
50         glVertex3f(-1.0, -1.0, 1.0);
51         glVertex3f(1.0, -1.0, 1.0);
52         glVertex3f(1.0, -1.0, -1.0);
53         glVertex3f(-1.0, -1.0, -1.0);
```

```

54     glEnd();
55     glEndList();
56     displayListInited = GL_TRUE;
57 }
58 if(doubleBuffer) glXSwapBuffers(dpy, win); /* buffer swap does implicit glFlush */
59     else glFlush(); /* explicit flush for single buffered case */
60 }

61 void
62 main(int argc, char **argv)
63 {
64     XVisualInfo     *vi;
65     Colormap        cmap;
66     XSetWindowAttributes swa;
67     GLXContext      cx;
68     XEvent          event;
69     GLboolean       needRedraw = GL_FALSE, recalcModelView = GL_TRUE;
70     int             dummy;

71     /* (1) open a connection to the X server */

72     dpy = XOpenDisplay(NULL);
73     if (dpy == NULL) fatalError("could not open display");

74     /* (2) make sure OpenGL's GLX extension supported */

75     if(!glXQueryExtension(dpy, &dummy, &dummy)) fatalError("X server has no OpenGL GLX extension");

76     /* (3) find an appropriate visual */

77     /* find an OpenGL-capable RGB visual with depth buffer */
78     vi = glXChooseVisual(dpy, DefaultScreen(dpy), dblBuf);
79     if (vi == NULL) {
80         vi = glXChooseVisual(dpy, DefaultScreen(dpy), snglBuf);
81         if (vi == NULL) fatalError("no RGB visual with depth buffer");
82         doubleBuffer = GL_FALSE;
83     }
84     if(vi->class != TrueColor) fatalError("TrueColor visual required for this program");

85     /* (4) create an OpenGL rendering context */

86     /* create an OpenGL rendering context */
87     cx = glXCreateContext(dpy, vi, /* no sharing of display lists */ None,
88                         /* direct rendering if possible */ GL_TRUE);
89     if (cx == NULL) fatalError("could not create rendering context");

90     /* (5) create an X window with the selected visual */

91     /* create an X colormap since probably not using default visual */
92     cmap = XCreateColormap(dpy, RootWindow(dpy, vi->screen), vi->visual, AllocNone);
93     swa.colormap = cmap;
94     swa.border_pixel = 0;
95     swa.event_mask = ExposureMask | ButtonPressMask | StructureNotifyMask;
96     win = XCreateWindow(dpy, RootWindow(dpy, vi->screen), 0, 0, 300, 300, 0, vi->depth,
97                       InputOutput, vi->visual, CWBorderPixel | CWColormap | CWEventMask, &swa);
98     XSetStandardProperties(dpy, win, "glxsimple", "glxsimple", None, argv, argc, NULL);

99     /* (6) bind the rendering context to the window */

100    glXMakeCurrent(dpy, win, cx);

```

```

101  /*** (7) request the X window to be displayed on the screen ***/
102  XMapWindow(dpy, win);

103  /*** (8) configure the OpenGL context for rendering ***/

104  glEnable(GL_DEPTH_TEST); /* enable depth buffering */
105  glDepthFunc(GL_LESS); /* pedantic, GL_LESS is the default */
106  glClearDepth(1.0); /* pedantic, 1.0 is the default */
107  /* frame buffer clears should be to black */
108  glClearColor(0.0, 0.0, 0.0, 0.0);
109  /* set up projection transform */
110  glMatrixMode(GL_PROJECTION);
111  glLoadIdentity();
112  glFrustum(-1.0, 1.0, -1.0, 1.0, 1.0, 10.0);
113  /* establish initial viewport */
114  glViewport(0, 0, 300, 300); /* pedantic, full window size is default viewport */

115  /*** (9) dispatch X events ***/

116  while (1) {
117      do {
118          XNextEvent(dpy, &event);
119          switch (event.type) {
120              case ButtonPress:
121                  recalcModelView = GL_TRUE;
122                  switch (event.xbutton.button) {
123                      case 1: xAngle += 10.0; break;
124                      case 2: yAngle += 10.0; break;
125                      case 3: zAngle += 10.0; break;
126                  }
127                  break;
128              case ConfigureNotify:
129                  glViewport(0, 0, event.xconfigure.width, event.xconfigure.height);
130                  /* fall through... */
131              case Expose:
132                  needRedraw = GL_TRUE;
133                  break;
134          }
135      } while(XPending(dpy)); /* loop to compress events */
136      if (recalcModelView) {
137          glMatrixMode(GL_MODELVIEW);
138          /* reset modelview matrix to the identity matrix */
139          glLoadIdentity();
140          /* move the camera back three units */
141          glTranslatef(0.0, 0.0, -3.0);
142          /* rotate by X, Y, and Z angles */
143          glRotatef(xAngle, 0.1, 0.0, 0.0);
144          glRotatef(yAngle, 0.0, 0.1, 0.0);
145          glRotatef(zAngle, 0.0, 0.0, 1.0);
146          recalcModelView = GL_FALSE;
147          needRedraw = GL_TRUE;
148      }
149      if (needRedraw) {
150          redraw();
151          needRedraw = GL_FALSE;
152      }
153  }
154 }

```

References

- [1] Allen Akin. "Analysis of PEX 5.1 and OpenGL 1.0." Silicon Graphics. August 3, 1992.
- [2] Paul Haeberli, Kurt Akeley. "The Accumulation Buffer: Hardware Support for High-Quality Rendering." *Proceedings of SIGGRAPH '90*. August 1990. pp. 309-318.
- [3] Phil Karlton. "Integrating the GL into the X Environment: A High Performance Rendering Extension Working with and Not Against X." *The X Resource: Proceeding of the 6th Annual X Technical Conference*. O'Reilly & Associates. Issue 1. Winter 1992.
- [4] Patricia McLendon. *Graphics Library Programming Guide*. Silicon Graphics. 1991.
- [5] Jackie Neider, Tom Davis, Mason Woo. *OpenGL Programming Guide: The official guide to learning OpenGL, Release 1*. Addison Wesley. 1993.
- [6] OpenGL Architecture Review Board. *OpenGL Reference Manual: The official reference document for OpenGL, Release 1*. Addison Wesley. 1992.
- [7] Mark Segal, Kurt Akeley. *The OpenGL™ Graphics System: A Specification, Version 1.0*. Silicon Graphics. June 30, 1992.
- [8] Paul Strauss, Rikk Carey. "An Object-Oriented 3D Graphics Toolkit." *Proceedings of SIGGRAPH '92*. July 1992. pp. 341-347.
- [9] Paula Womack, et.al., "PEX Protocol Specification, Version 5.1." The X Consortium. August 31, 1992.

OpenGL™ and X, Part 2: Using OpenGL with Xlib

Mark J. Kilgard *
Silicon Graphics Inc.
Revision : 1.21

January 18, 1994

Abstract

This is the second article in a three-part series about using the OpenGL™ graphics system and the X Window System. A moderately complex OpenGL program for X is presented. Depth buffering, back-face culling, lighting, display list modeling, polygon tessellation, double buffering, and shading are all demonstrated. The program adheres to proper X conventions for colormap sharing, window manager communication, command line argument processing, and event processing. After the example, advanced X and OpenGL issues are discussed including minimizing colormap flashing, handling overlays, using fonts, and performing animation. The last article in this series discusses integrating OpenGL with the Motif toolkit.

1 Introduction

In the first article in this series, the OpenGL™ graphics system was introduced. Along with an explanation of the system's functionality, a simple OpenGL X program was presented and OpenGL was compared to the X Consortium's PEX extension. In this article, a more involved example of programming OpenGL with X is presented. The example is intended to demonstrate both sophisticated OpenGL functionality and proper integration of OpenGL with the X Window System.

This article is intended to answer questions from two classes of programmers: first, the X programmer wanting to see OpenGL used in a program of substance; second, the OpenGL or IRIS GL programmer likely to be unfamiliar with the more mundane window system setup necessary when using the X Window System at the Xlib layer.

*Mark graduated with B.A. in Computer Science from Rice University and is a Member of the Technical Staff at Silicon Graphics. He can be reached by electronic mail addressed to mjk@sgi.com

The example program called `glxdino` renders a 3D dinosaur model using OpenGL. Hidden surfaces are removed using depth buffering. Back-face culling improves rendering performance by not rendering back-facing polygons. Hierarchical modeling is used to construct the dinosaur and render it via OpenGL display lists. The OpenGL Utility Library (GLU) polygon tessellation routines divide complex polygons into simpler polygons renderable by OpenGL. Sophisticated lighting lends realism to the dinosaur. If available, double buffering smoothes animation.

The program integrates well with the X Window System. The program accepts some of the standard X command line options: `-display`, `-geometry`, and `-iconic`. The user can rotate the model using mouse motion. Top-level window properties specified by the Inter-Client Communication Convention Manual (ICCCM) are properly set up to communicate with the window manager. Colormap sharing is done via ICCCM conventions. And the proper way of communicating to the window manager a desire for a constant aspect ratio is demonstrated.

A walk through of the `glxdino` source code is presented in Section 2. While `glxdino` tries to demonstrate a good number of OpenGL features and many of the issues concerning how X and OpenGL integrate, it is only an example. Section 3 explores more of the issues encountered when writing an advanced OpenGL program using Xlib. The third and last article in this series discusses how to integrate OpenGL with the Motif toolkit.

2 Example Walk Through

The source code for `glxdino` can be found in Appendix A. I will refer to the code repeatedly throughout this section. Figure 1 shows a screen snapshot of `glxdino`.

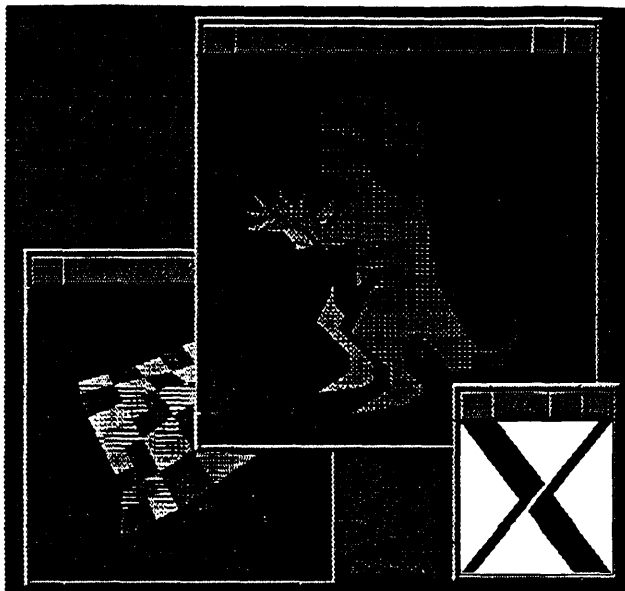


Figure 1: Screen snapshot of `glxdino`.

2.1 Initialization

The program's initialization proceeds through the following steps:

1. Process the standard X command line options.
2. Open the connection to the X server.
3. Determine if OpenGL's GLX extension is supported.
4. Find the appropriate X visual and colormap.
5. Create an OpenGL rendering context.
6. Create an X window with the selected visual and properly specify the right ICCCM properties for the window manager to use.
7. Bind the rendering context to the window.
8. Make the display list hierarchy for the dinosaur model.
9. Configure OpenGL rendering state.
10. Map the window.
11. Begin dispatching X events.

Comments in the code correspond to these enumerated steps.

In the program's `main` routine, the first task is to process the supported command line arguments. Users of the X Window System should be familiar with `-display` which specifies the X server to use, `-geometry` which specifies the initial size and location of the program's main window, and `-iconic` which requests the window be initially

iconified. Programmers used to the IRIS GL (the predecessor to OpenGL) may not be familiar with these options. While nothing requires an X program to accept standard X options, most do as a matter of consistency and convenience. Most X toolkits automatically understand the standard set of X options

The `-keepaspect` option is not a standard X command line option. When specified, it requests that the window manager ensure that the ratio between the initial width and height of the window be maintained. Often for 3D programs, the programmer would like a constant aspect ratio for their rendering window. In IRIS GL, a call named `keepaspect` is available. Maintaining the aspect ratio of a window is something for the window system to do so there is no call analogous to IRIS GL's `keepaspect` in OpenGL. Remember that the core OpenGL Application Programmer Interface (API) attempts to be window system independent. IRIS GL programmers used to the IRIS GL interface will need to become aware of X functionality to do things that used to be done with IRIS GL calls.

Normally `glxdino` tries to use a double buffered window but will use a single buffered window if a double buffered visual is not available. When the `-single` option is present, the program will look only for a single buffered visual. On many machines with hardware double buffering support, color resolution can be traded for double buffering to achieve smooth animation. For example, a machine with 24 bits of color resolution could support 12 bits of color resolution for double buffered mode. Half the image bit-planes would be for the front buffer and half for the back buffer.

Next, a connection to the X server is established using `XOpenDisplay`. Since `glxdino` requires OpenGL's GLX extension, the program checks that the extension exists using `glXQueryExtension`. The routine indicates if the GLX extension is supported or not. As is convention for X routines that query extensions, the routine can also return the *base error code* and *base event code* for the GLX extension. The current version of GLX supports no extension events (but does define eight protocol errors). Most OpenGL programs will need neither of these numbers. You can pass in `NULL` as `glxdino` does to indicate you do not need the event or error base.

OpenGL is designed for future extensibility. The `glXQueryVersion` routine returns the major and minor version of the OpenGL implementation. Currently, the major version is 1 and the minor version is 0. `glxdino` does not use `glXQueryVersion` but it may be useful for programs in the future.

2.1.1 Choosing a Visual and Colormap

The GLX extension overloads X visuals to denote supported frame buffer configurations. Before you create an OpenGL window, you should select a visual which sup-

ports the frame buffer features you intend to use. GLX guarantees at least two visual will be supported. An RGBA mode visual with a depth buffer, stencil buffer, and accumulation buffer must be supported. Second, a color index mode visual with a depth buffer and stencil buffer must be available. More and less capable visuals are likely to also be supported depending on the implementation.

To make it easy to select a visual, `glXChooseVisual` takes a list of the capabilities you are requesting and returns an `XVisualInfo*` for a visual meeting your requirements. `NULL` is returned if a visual meeting your needs is not available. To ensure your application will run with any OpenGL GLX server, your program should be written to support the base line required GLX visuals. Also you should only ask for the minimum set of frame buffer capabilities you require. For example, if your program never uses a stencil buffer, you will possibly waste resources if you request one anyway.

Since `glxdino` rotates the dinosaur in response to user input, the program will run better if double buffering is available. Double buffering allows a scene to be rendered out of view and then displayed nearly instantly to eliminate the visual artifacts associated with watching a 3D scene render. Double buffering helps create the illusion of smooth animation. Since double buffering support is not required for OpenGL implementations, `glxdino` resorts to single buffering if no double buffer visuals are available. The program's `configuration` integer array tells what capabilities `glXChooseVisual` should look for. Notice how if a double buffer visual is not found, another attempt is made which does not request double buffering by starting after the `GLX_DOUBLBUFFER` token. And when the `-single` option is specified, the code only looks for a singled buffered visual.

`glxdino` does require a depth buffer (of at least 16 bits of accuracy) and uses the RGBA color model. The RGBA base line visual must support at least a 16 bit depth buffer so `glxdino` should always find a usable visual.

You should not assume the visual you need is the default visual. Using a non-default visual means windows created using the visual will require a colormap matching the visual. Since the window we are interested in uses OpenGL's RGBA color model, we want a colormap configured for using RGB. The ICCCM establishes a means for sharing RGB colormaps between clients. `XmuLookupStandardColormap` is used to set up a colormap for the specified visual. The routine reads the ICCCM `RGB_DEFAULT_MAP` property on the X server's root window. If the property does not exist or does not have an entry for the specified visual, a new RGB colormap is created for the visual and the property is updated (creating it if necessary). Once the colormap has been created, `XGetRGBColormaps` finds the newly created colormap. The work for finding a colormap is done by the `getColormap` routine.

If a standard colormap cannot be allocated, `glxdino` will create an unshared colormap. For some servers, it is possible (though unlikely) a `DirectColor` visual might be returned (though the GLX specification requires a `TrueColor` visual be returned in precedence to a `DirectColor` visual if possible). To shorten the example code by only handling the most likely case, the code bails if a `DirectColor` visual is encountered. A more portable (and longer) program would be capable of initializing an RGB `DirectColor` colormap.

2.1.2 Creating a Rendering Context

Once a suitable visual and colormap are found, the program can create an OpenGL rendering context using `glXCreateContext`. (The same context can be used for different windows with the same visual.)

The last parameter allows the program to request a direct rendering context if the program is connected to a local X server. An OpenGL implementation is not required to support direct rendering, but if it does, faster rendering is possible since OpenGL will render directly to the graphics hardware. Direct rendered OpenGL requests do not have to be sent to the X server. Even when on the local machine, you may not want direct rendering in some cases. For example, if you want to render to X pixmaps, you must render through the X server.

GLX rendering contexts support sharing of display lists among one another. To this end, the third parameter to `glXCreateContext` is another already created GLX rendering context. `NULL` can be specified to create an initial rendering context. If an already existent rendering context is specified, the display list indexes and definitions are shared by the two rendering contexts. The sharing is transitive so a share group can be formed between a whole set of rendering contexts.

To share, all the rendering contexts must exist in the *same* address space. This means direct renderers cannot share display lists with renderers rendering through the X server. Likewise direct renderers in separate programs cannot share display lists. Sharing display lists between renderers can help to minimize the memory requirements of applications that need the same display lists.

2.1.3 Setting Up a Window

Because OpenGL uses visuals to distinguish various frame buffer capabilities, programmers using OpenGL need to be aware of the required steps to create a window with a non-default visual. As mentioned earlier a colormap created for the visual is necessary. But the most irksome thing to remember about creating a window with a non-default visual is that the border pixel value *must* be specified if the window's visual is not the same as its parent's visual. Otherwise a `BadMatch` is generated.

Before actually creating the window, the argument to the `-geometry` option should be parsed using `XParseGeometry` to obtain the user's requested size and location. The size will be needed when we create the window. Both the size and location are needed to set up the ICCCM size hints for the window manager. A fixed aspect ratio is also requested by setting up the right size hints if the `-keepaspect` option is specified.

Once the window is created, `XSetStandardProperties` sets up the various standard ICCCM properties including size hints, icon name, and window name. Then the ICCCM window manager hints are set up to indicate the window's initial state. The `-iconic` option sets the window manager hints to indicate the window should be initially iconified. `XAllocWMHints` allocates a hints structure. Once filled in, `XSetWMHints` sets up the hint property for the window.

The final addition to the window is the `WM_PROTOCOLS` property which indicates window manager protocols the client understands. The most commonly used protocol defined by ICCCM is `WM_DELETE_WINDOW`. If this atom is listed in the `WM_PROTOCOLS` property of a top-level window, then when the user selects the program be quit from the window manager, the window manager will politely send a `WM_DELETE_WINDOW` message to the client instructing the client to delete the window. If the window is the application's main window, the client is expected to terminate. If this property is not set, the window manager will simply ask the X server to terminate the client's connection without notice to the client. By default, this results in Xlib printing an ugly message like:

```
X connection to :0.0 broken
(explicit kill or server shutdown).
```

Asking to participate in the `WM_DELETE_WINDOW` protocol allows the client to safely handle requests to quit from the window manager.

The property has another advantage for OpenGL programs. Many OpenGL programs doing animation will use `XPending` to check for pending X events and otherwise draw their animation. But if all a client's animation is direct OpenGL rendering and the client does not otherwise do any X requests, the client never sends requests to the X server. Due to a problem in `XPending`'s implementation on many Unix operating systems,¹ such an OpenGL program might not notice its X connection was terminated for sometime. Using the `WM_DELETE_WINDOW` protocol eliminates this problem because the window manager notifies

¹Operating systems using `FIONREAD` ioctl calls on file descriptors using Berkeley non-blocking I/O cannot differentiate no data to read from a broken connection; both conditions cause the `FIONREAD` ioctl to return zero. MIT's standard implementation of `XPending` uses Berkeley non-blocking I/O and `FIONREAD` ioctls. Eventually, Xlib will do an explicit check on the socket to see if it closes but only after a couple hundred calls to `XPending`.

the client via a message (tripping `XPending`) and the client is expected to drop the connection.

Using the `WM_DELETE_WINDOW` protocol is good practice even if you do not use `XPending` and the Xlib message does not bother you.

All these steps (besides creating a window with a non-default visual) are standard for creating a top-level X window. A top-level window is a window created as a child of the root window (the window manager may choose to reparent the window when it is mapped to add a border). Note that the properties discussed are placed on the *top-level* window, not necessarily the same window that OpenGL renders into. While `glxdino` creates a single window, a more complicated program might nest windows used for OpenGL rendering inside the top-level window. The ICCCM window manager properties belong on top-level windows only.

An IRIS GL programmer not familiar with X will probably find these details cumbersome. Most of the work will be done for you if you use a toolkit layered on top of Xlib.

Now a window and an OpenGL rendering context exist. In OpenGL (unlike Xlib), you do not pass the rendering destination into every rendering call. Instead a given OpenGL rendering context is bound to a window using `glXMakeCurrent`. Once bound, all OpenGL rendering calls operate using the current OpenGL rendering context and the current bound window. A thread can only be bound to one window and one rendering context at a time. A context can only be bound to a single thread at a time. If you call `glXMakeCurrent` again, it unbinds from the old context and window and then binds to the newly specified context and window. You can unbind a thread from a window and a context by passing `NULL` for the context and `None` for the drawable.

2.2 The Dinosaur Model

The task of figuring out how to describe the 3D object you wish to render is called *modeling*. Much as a plastic airplane model is constructed out of little pieces, a computer generated 3D scene must also be built out of little pieces. In the case of 3D rendering, the pieces are generally polygons.

The dinosaur model to be displayed is constructed out of a hierarchy of display lists. Rendering the dinosaur is accomplished by executing a single display list.

The strategy for modeling the dinosaur is to construct solid pieces for the body, arms, legs, and eyes. Figure 2 shows the 2D sides of the solids to construct the dinosaur. Making these pieces solid is done by *extruding* the sides (meaning stretching the 2D sides into a third dimension). By correctly situating the solid pieces relative to each other, they form the complete dinosaur.

The work to build the dinosaur model is done by the routine named `makeDinosaur`. A helper routine

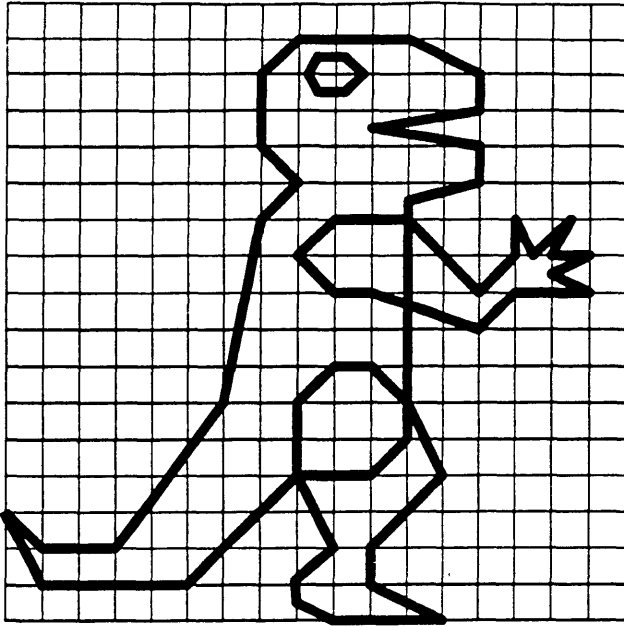


Figure 2: 2D complex polygons used to model the dinosaur's arm, leg, eye, and body sides.

`extrudeSolidFromPolygon` is used to construct each solid extruded object.

2.2.1 The GLU Tessellator

The polygons in Figure 2 are irregular and complex. For performance reasons, OpenGL directly supports drawing only convex polygons. The complex polygons that make up the sides of the dinosaur need to be built from smaller convex polygons.

Since rendering complex polygons is a common need, OpenGL supplies a set of utility routines in the OpenGL GLU library which make it easy to *tessellate* complex polygons. In computer graphics, tessellation is the process of breaking a complex geometric surface into simple convex polygons.

The GLU library routines for tessellation are:

`gluNewTess` - create a new tessellation object.

`gluTessCallback` - define a callback for a tessellation object.

`gluBeginPolygon` - begin a polygon description to tessellate.

`gluTessVertex` - specify a vertex for the polygon to tessellate.

`gluNextContour` - mark the beginning of another contour for the polygon to tessellate.

`gluEndPolygon` - finish a polygon being tessellated.

`gluDeleteTess` - destroy a tessellation object.

These routines are used in the example code to tessellate the sides of the dinosaur. Notice at the beginning of the program static arrays of 2D vertices are specified for the dinosaur's body, arm, leg, and eye polygons.

To use the tessellation package, you first create a tessellation object with `gluNewTess`. An object of type `GLUTriangulatorObj*` is returned which is passed into the other polygon tessellation routines. You do not need a tessellation object for every polygon you tessellate. You might need more than one tessellation object if you were trying to tessellate more than one polygon at a time. In the sample program, a single tessellation object is used for all the polygons needing tessellation.

Once you have a tessellation object, you should set up callback routines using `gluTessCallback`. The way that the GLU tessellation package works is that you feed in vertices. Then the tessellation is performed and your registered callbacks are called to indicate the beginning, end, and all the vertices for the convex polygons which correctly tessellate the points you feed to the tessellator.

Look at the `extrudeSolidFromPolygon` routine which uses the GLU tessellation routines. To understand exactly why the callbacks are specified as they are, consult the OpenGL Reference Manual [4]. The point to notice is how a single tessellation object is set up once and callbacks are registered for it. Then `gluBeginPolygon` is used to start tessellating a new complex polygon. The vertices of the polygon are specified using `gluTessVertex`. The polygon is finished by calling `gluEndPolygon`.

Notice the code for tessellating the polygon lies between a `glNewList` and `glEndList`; these routines begin and end the creation of a display list. The callbacks will generate `glVertex2fv` calls specifying the vertices of convex polygons needed to represent the complex polygon being tessellated. Once completed, a display list is available that can render the desired complex polygon.

Consider the performance benefits of OpenGL's polygon tessellator compared with a graphics system that supplies a polygon primitive that supports non-convex polygons. A primitive which supported complex polygons would likely need to tessellate each complex polygon on the fly. Calculating a tessellation is not without cost. If you were drawing the same complex polygon more than once, it is better to do the tessellation only once. This is exactly what is achieved by creating a display list for the tessellated polygon. But if you are rendering continuously changing complex polygons, the GLU tessellator is fast enough for generating vertices on the fly for immediate-mode rendering.

Having a tessellation object not directly tied to rendering is also more flexible. Your program might need to tessellate a polygon but not actually render it. The GLU's system of callbacks just generate vertices. You can call OpenGL `glVertex` calls to render the vertices or supply

your own special callbacks to save the vertices for your own purposes. The tessellation algorithm is accessible for your own use.

The GLU tessellator also supports multiple contours allowing disjoint polygons or polygons with holes to be tessellated. The `gluNextContour` routine begins a new contour.

The tessellation object is just one example of functionality in OpenGL's GLU library which supports 3D rendering without complicating the basic rendering routines in the core OpenGL API. Other GLU routines support rendering of curves and surfaces using Non-Uniform Rational B-Splines (NURBS) and tessellating boundaries of solids such as cylinders, cones, and spheres. All the GLU routines are a standard part of OpenGL.

2.2.2 Hierarchical Display Lists

After generating the complex polygon display list for the sides of a solid object, the `extrudeSolidFromPolygon` routine creates another display list for the "edge" of the extruded solid. The edge is generated using a `QUAD_STRIP` primitive. Along with the vertices, normals are calculated for each quad along the edge. Later these normals will be used for lighting the dinosaur. The normals are computed to be unit vectors. Having normals specified as unit vectors is important for correct lighting. An alternative would be to use `glEnable(GL_NORMALIZE)` which ensures all normals are properly normalized before use in lighting calculations. Specifying unit vectors to begin with and not using `glEnable(GL_NORMALIZE)` saves time during rendering. Be careful when using scaling transformations (often set up using `glScale`) since scaling transformations will scale normals too. If you are using scaling transformations, `glEnable(GL_NORMALIZE)` is almost always required for correct lighting.

Once the edge and side display lists are created, the solid is formed by calling the edge display list, then filling in the solid by calling the side display list twice (once translated over by the width of the edge). The `makeDinosaur` routine will use `extrudeSolidFromPolygon` to create solids for each body part needed by the dinosaur.

Then `makeDinosaur` combines these display lists into a single display list for the entire dinosaur. Translations are used to properly position the display lists to form the complete dinosaur. The body display list is called; then arms and legs for the right side are added; then arms and legs for the left side are added; then the eye is added (it is one solid which pokes out either side of the dinosaur's head a little bit on each side).

2.2.3 Back-face Culling

A common optimization in 3D graphics is a technique known as *back-face culling*. The idea is to treat polygons as essentially one-sided entities. A front facing polygon

needs to be rendered but a back-facing polygon can be eliminated.

Consider the dinosaur model. When the model is rendered, the back side of the dinosaur will not be visible. If the direction each polygon "faced" was known, OpenGL could simply eliminate approximately half of the polygons (the back-facing ones) without ever rendering them.

Notice the calls to `glFrontFace` when each solid display list is created in `extrudeSolidFromPolygon`. The argument to the call is either `GL_CW` or `GL_CCW` meaning clock-wise and counter-clockwise. If the vertices for a polygon are listed in counter-clockwise order and `glFrontFace` is set to `GL_CCW`, then the generated polygon is considered front facing. The static data specifying the vertices of the complex polygons is listed in counter-clockwise order. To make the quads in the quad strip face outwards, `glFrontFace(GL_CW)` is specified. The same mode ensures the far side faces outward. But `glFrontFace(GL_CCW)` is needed to make sure the front of the other side faces outward (logically it needs to be reversed from the opposite side since the vertices were laid out counter-clockwise for both sides since they are from the same display list).

When the static OpenGL state is set up, `glEnable(GL_CULL_FACE)` is used to enable back-face culling. As with all modes enabled and disabled using `glEnable` and `glDisable`, it is disabled by default. Actually OpenGL is not limited to back-face culling. The `glCullFace` routine can be used to specify either the back or the front should be culled when face culling is enabled.

When you are developing your 3D program, it is often helpful to disable back-face culling. That way both sides of every polygon will be rendered. Then once you have your scene correctly rendering, you can go back and optimize your program to properly use back-face culling.

Do not be left with the misconception that enabling or disabling back-face culling (or any other OpenGL feature) must be done for the duration of the scene or program. You can enable and disable back-face culling at will. It is possible to draw part of your scene with back-face culling enabled, and then disable it, only to later re-enable culling but this time for front faces.

2.3 Lighting

The realism of a computer generated 3D scene is greatly enhanced by adding lighting. In the first article's sample program, `glColor3f` was used to add color to the faces of the 3D cube. This adds color to rendered objects but does not use lighting. In the example, the cube moves but the colors do not vary the way a real cube might as it is affected by real world lighting. In this article's example, lighting will be used to add an extra degree of realism to the scene.

OpenGL supports a sophisticated 3D lighting model to achieve higher realism. When you look at a real object,

its color is affected by lights, the material properties of the object, and the angle at which the light shines on the object. OpenGL's lighting model approximates the real world.

Complicated effects such as the reflection of light and shadows are not supported by OpenGL's lighting model though techniques and algorithms are available to simulate such effects. Environment mapping to simulate reflection is possible using OpenGL's texturing capability. OpenGL's stencil buffers and blending support can be used to create shadows, but an explanation of these techniques is beyond the scope of this article. (See the topics in the final chapter of the *OpenGL Programming Guide*).

2.3.1 Types of Lighting

The effects of light are complex. In OpenGL, lighting is divided into four different components: emitted, ambient, diffuse, and specular. All four components can be computed independently and then added together.

Emitted light is the simplest. It is light that originates from an object and is unaffected by any light sources. Self-luminous objects can be modeled using emitted light.

Ambient light is light from some source that has been scattered so much by the environment that its direction is impossible to determine. Even a directed light such as a flashlight may have some ambient light associated with it.

Diffuse light comes from some direction. The brightness of the light bouncing off an object depends on the light's angle of incidence with the surface it is striking. Once it hits a surface, the light is scattered equally in all directions so it appears equally bright independent of where the eye is located.

Specular light comes from some direction and tends to bounce off the surface in a certain direction. Shiny metal or plastic objects have a high specular component. Chalk or carpet have almost none. Specularity corresponds to the everyday notion of how shiny an object is.

A single OpenGL light source has a single color and some combination of ambient, diffuse, and specular components. OpenGL supports multiple lights simultaneously. The programmer can control the makeup of a light as well as its position, direction, and attenuation. Attenuation refers to how a light's intensity decreases as distance from the light increases.

2.3.2 Lighting in the Example

The example uses two lights. Both use only the diffuse component. A bright, slightly green-tinted *positional* light is to the right, front of the dinosaur. A dim, red-tinted *directional* light is coming from the left, front of the dinosaur. Figure 3 shows how the dinosaur, the lights, and the eye-point are arranged. A positional light is located at some finite position in modeling space. A directional light

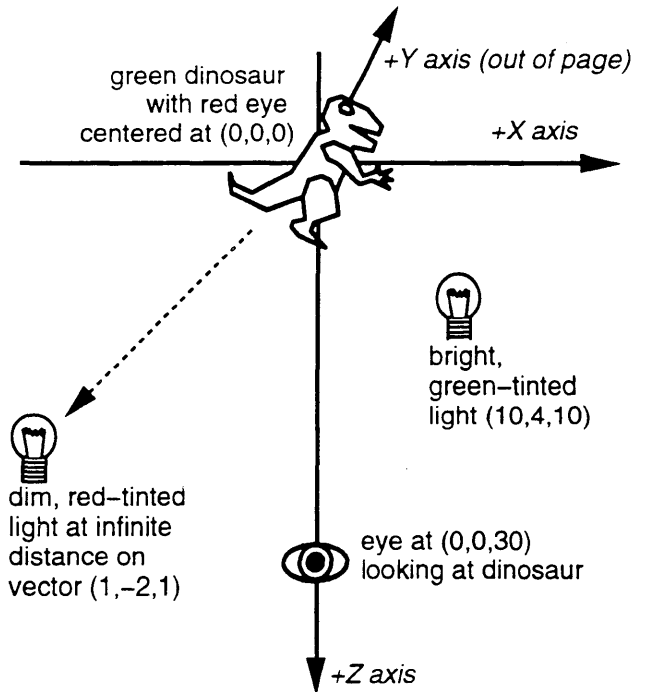


Figure 3: Arrangement of lights, eye, and dinosaur in modeling space.

is considered to be located infinitely far away. Using a directional light allows the OpenGL to consider the emitted light rays to be parallel by the time the light reaches the object. This simplifies the lighting calculations needed to be done by OpenGL.

The `lightZeroPosition` and `lightOnePosition` static variables indicate the position of the two lights. You will notice each has not three but four coordinates. This is because the light location is specified in *homogeneous* coordinates. The fourth value divides the X, Y, and Z coordinates to obtain the true coordinate. Notice how `lightOnePosition` (the infinite light) has the fourth value set to zero. This is how an infinite light is specified.²

The dinosaur can rotate around the Y axis based on the user's mouse input. The idea behind the example's lighting arrangement is when the dinosaur is oriented so its side faces to the right, it should appear green due to the bright light. When its side faces leftward, the dinosaur should appear poorly lit but the red infinite light should catch the dinosaur's red eye.

Section 9 of the program initialization shows how lighting is initialized. The `glEnable(GL_LIGHTING)` turns on lighting support. The lights' positions and diffuse com-

²Actually all coordinates are logically manipulated by OpenGL as three-dimensional homogeneous coordinates. The *OpenGL Programming Guide's* Appendix G [3] briefly explains homogeneous coordinates. A more involved discussion of homogeneous coordinates and why they are useful for 3D computer graphics can be found in Foley and van Dam [1].

ponents are set using via calls to `glLightfv` using the `GL_POSITION` and `GL_DIFFUSE` parameters. The lights are each enabled using `glEnable`.

The attenuation of the green light is adjusted. This determines how the light intensity fades with distance and demonstrates how individual lighting parameters can be set. It would not make sense to adjust the attenuation of the red light since it is an infinite light which shines with uniform intensity.

Neither ambient nor specular lighting are demonstrated in this example so that the effect of the diffuse lighting would be clear. Specular lighting might have been used to give the dinosaur's eye a glint.

Recall when the edge of each solid was generated, normals were calculated for each vertex along the quad strip. And a single normal was given for each complex polygon side of the solid. These normals are used in the diffuse lighting calculations to determine how much light should be reflected. If you rotate the dinosaur, you will notice the color intensity changes as the angle incidence for the light varies.

Also notice the calls to `glShadeModel`. OpenGL's shade model determines whether flat or smooth shading should be used on polygons. The dinosaur model uses different shading depending on whether a side or edge is being rendered. There is a good reason for this. The `GL_SMOOTH` mode is used on the sides. If flat shading were used instead of smooth, each convex polygon composing the tessellated complex polygon side would be a single color. The viewer could notice exactly how the sides has been tessellated. Smooth shading prevents this since the colors are interpolated across each polygon.

But for the edge of each solid, `GL_FLAT` is used. Because the edge is generated as a quad strip, quads along the strip share vertices. If we used a smooth shading model, each edge between two quads would have a single normal. Some of the edges are very sharp (like the claws in the hand and the tip of the tail). Interpolating across such varying normals would lead to an undesirable visual effect. The fingers would appear rounded if looked at straight on. Instead, with flat shading, each quad gets its own normal and there is no interpolation so the sharp angles are clearly visible.

2.4 View Selection

In 3D graphics, *viewing* is the process of establishing the perspective and orientation with which the scene should be rendered. Like a photographer properly setting up his camera, an OpenGL programmer should establish a view. Figure 4 shows how the view is set up for the example program.

In OpenGL, establishing a view means loading the projection and model-view matrices with the right contents. To modify the projection matrix, call

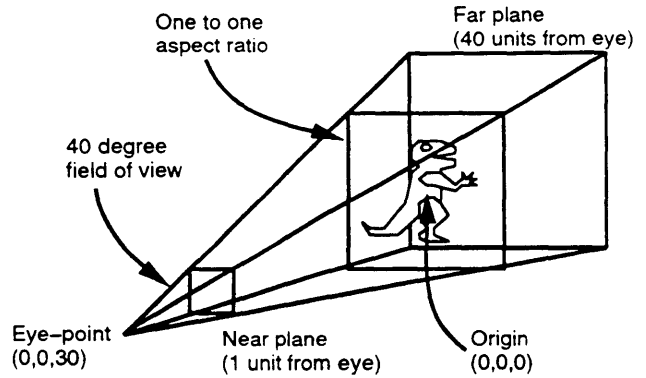


Figure 4: Static view for `glxdino`.

`glMatrixMode(GL_PROJECTION)`. Calculating the right matrix by hand can be tricky. The GLU library has two useful routines that make the process easy.

GLU's `gluPerspective` routine allows you to specify a field of view angle, an aspect ratio, and near and far clipping planes. It multiplies the current projection matrix with one created according to the routine's parameters. Since initially the projection matrix is an identity matrix, `glxdino`'s `gluPerspective` call effectively loads the projection matrix.

Another GLU routine, `gluLookAt`, can be used to orient the eye-point for the model-view matrix. Notice how `glMatrixMode(GL_MODELVIEW)` is used to switch to the model-view matrix. Using `gluLookAt` requires you to specify the eye-point's location, a location to look at, and a normal to determine which way is up. Like `gluPerspective`, `gluLookAt` multiplies the matrix it constructs from its parameters with the current matrix. The initial model-view matrix is the identity matrix so `glxdino`'s call to `gluLookAt` effectively loads the model-view matrix.

After the `gluLookAt` call, `glPushMatrix` is called. Both the model-view and projection matrices exist on stacks that can be pushed and popped. Calling `glPushMatrix` pushes a copy of the current matrix onto the stack. When a rotation happens, this matrix is popped off and another `glPushMatrix` is done. This newly pushed matrix is composed with a rotation matrix to reflect the current absolute orientation. Every rotation pops off the top matrix and replaces it with a newly rotated matrix.

Notice that the light positions are not set until after the model-view matrix has been properly initialized.

Because the location of the viewpoint affects the calculations for lighting, separate the projection transformation in the projection matrix and the modeling and viewing transformations in the model-view matrix.

2.5 Event Dispatching

Now the window has been created, the OpenGL renderer has been bound to it, the display lists have been constructed, and OpenGL's state has been configured. All that remains is to request the window be mapped using `XMapWindow` and begin handling any X events sent to the program.

When the window was created, four types of window events were requested to be sent to our application: `Expose` events reporting regions of the window to be drawn, `ButtonPress` events indicating mouse button status, `KeyPress` events indicating a keyboard key has been pressed, `MotionNotify` events indicating mouse movement, and `ConfigureNotify` events indicating the window's size or position has changed.

X event dispatching is usually done in an infinite loop. Most X programs do not stop dispatching events until the program terminates. `XNextEvent` can be used to block waiting for an X event. When an event arrives, its type is examined to tell what event has been received.

2.5.1 Expose Handling

For an `Expose` event, the example program just sets a flag indicating the window needs to be redrawn. The reason is that `Expose` events indicate a single sub-rectangle in the window that must be redrawn. The X server will send a number of `Expose` events if a complex region of the window has been exposed.

For a normal X program using 2D rendering, you might be able to minimize the amount needed to redraw the window by carefully examining the rectangles for each `Expose` event. For 3D programs, this is usually too difficult to be worthwhile since it is hard to determine what would need to be done to redraw some sub-region of the window. In practice the window is usually redrawn in its entirety. For the dinosaur example, redrawing involves calling the dinosaur display list with the right view. It is not helpful to know only a sub-region of the window actually needs to be redrawn. For this reason, an OpenGL program should not begin redrawing until it has received all the `Expose` events most recently sent to the window. This practice is known as *expose compression* and helps to avoid redrawing more than you should.

Notice that all that is done to immediately handle an `Expose` is to set the `needRedraw` flag. Then `XPending` is used to determine if there are more events pending. Not until the stream of events pauses is the `redraw` routine really called (and the `needRedraw` flag reset).

The `redraw` routine does three things: it clears the image and depth buffers, executes the dinosaur display list, and either calls `glXSwapBuffers` on the window if double buffered or calls `glFlush`. The current model-view matrix determines in what orientation the dinosaur is drawn.

2.5.2 Window Resizing

The X server sends a `ConfigureNotify` event to indicate a window resize. Handling the event generally requires changing the viewport of OpenGL windows. The sample program calls `glViewport` specifying the window's new width and height. A resize also necessitates a screen redraw so the code "falls through" to the `Expose` code which sets the `needRedraw` flag.

When you resize the window, the aspect ratio of the window may change (unless you have negotiated a fixed aspect ratio with the window manager as the `-keepaspect` option does). If you want the aspect ratio of your final image to remain constant, you might need to respecify the projection matrix with an aspect ratio to compensate for the window's changed aspect ratio. The example does not do this.

2.5.3 Handling Input

The example program allows the user to rotate the dinosaur while moving the mouse by holding down the first mouse button. We record the current angle of rotation whenever a mouse button state changes. As the mouse moves while the first mouse button is held down, the angle is recalculated. A `recalcModelView` flag is set indicating the scene should be redrawn with the new angle.

When there is a lull in events, the model-view matrix is recalculated and then the `needRedraw` flag is set, forcing a redraw. The `recalcModelView` flag is cleared. As discussed earlier, recalculating the model-view is done by popping off the current top matrix using `glPopMatrix` and pushing on a new matrix. This new matrix is composed with a rotation matrix using `glRotatef` to reflect the new absolute angle of rotation. An alternative approach would be to multiply the current matrix by a rotation matrix reflecting the change in angle of rotation. But such a relative approach to rotation can lead to inaccurate rotations due to accumulated floating point round-off errors.

2.5.4 Quitting

Because the `WM_DELETE_WINDOW` atom was specified on the top-level window's list of window manager protocols, the event loop should also be ready to handle an event sent by the window manager asking the program to quit. If `glxdino` receives a `ClientMessage` event with the first data item being the `WM_DELETE_WINDOW` atom, the program calls `exit`.

In many IRIS GL demonstration programs, the Escape key is used by convention to quit the program. So `glxdino` shows a simple means to quit in response to an Escape key press.

3 Advanced Xlib and OpenGL

The `gldino` example demonstrates a good deal of OpenGL's functionality and how to integrate OpenGL with X but there are a number of issues that programmers wanting to write advanced OpenGL programs for X should be aware of.

3.1 Colormaps

Already a method has been presented for sharing colormaps using the ICCCM conventions. Most OpenGL programs do not use the default visual and therefore cannot use the default colormap. Sharing colormaps is therefore important for OpenGL programs to minimize the amount of colormaps X servers will need to create.

Often OpenGL programs require more than one colormap. A typical OpenGL program may do OpenGL rendering in a subwindow but most of the program's user interface is implemented using normal X 2D rendering. If the OpenGL window is 24 bits deep, it would be expensive to require all the user interface windows also to be 24 bits deep. Among other things, pixmaps for the user interface windows would need to be 32 bits per pixel instead of the typical 8 bits per pixel. So the program may use the server's (probably default) 8 bit `PseudoColor` visual for its user interface but use a 24 bit `TrueColor` visual for its OpenGL subwindow. Multiple visuals demand multiple colormaps. Many other situations may arise when an OpenGL program needs multiple colormaps within a single top-level window hierarchy.

Normally window managers assume the colormap that a top-level window and all its subwindows need is the colormap used by the top-level window. A window manager automatically notices the colormap of the top-level window and tries to ensure that that colormap is installed when the window is being interacted with.

With multiple colormaps used inside a single top-level window, the window manager needs to be informed of the other colormaps being used. The Xlib routine `XSetWMColormapWindows` can be used to place a standard property on your top-level window to indicate all the colormaps used by the top-level window and its descendants.

Be careful about using multiple colormaps. It is possible a server will not have enough colormap resources to support the set of visuals and their associated colormaps that you desire. Unfortunately, there is no standard way to determine what sets of visuals and colormaps can be simultaneously installed when multiple visuals are supported. Xlib provides two calls, `XMaxCmapsOfScreen` and `XMinCmapsOfScreen`, but these do not express hardware conflicts between visuals.

Here are some guidelines:

- If `XMaxCmapsOfScreen` returns one, you are guaranteed a single hardware colormap. Colormap flashing

is quite likely. You should write your entire application to use a single colormap at a time.

- If an 8 bit `PseudoColor` visual and a 24 bit `TrueColor` visual are supported on a single screen, it is extremely likely a different colormap for each of the two visuals can be installed simultaneously.
- If `XMaxCmapsOfScreen` returns a number higher than one, it is possible that the hardware supports multiple colormaps for the same visual. A rule of thumb is the higher the number, the more likely. If the number is higher than the total number of visuals on the screen, it must be true for at least one visual (but you cannot know which one).

Hopefully multiple hardware colormaps will become more prevalent and perhaps a standard mechanism to detect colormap and visual conflicts will become available.

3.2 Double Buffering

If you are writing an animated 3D program, you will probably want double buffering. It is not always available for OpenGL. You have two choices: run in single-buffered mode or render to a pixmap and copy each new frame to the window using `XCopyArea`.

Note that when you use `glXChooseVisual`, booleans are matched exactly (integers if specified are considered minimums). This means if you want to support double buffering but be able to fall back to single buffering, two calls will be needed to `glXChooseVisual`. If an OpenGL application has sophisticated needs for selecting visuals, `glXGetConfig` can be called on each visual to determine the OpenGL attributes of each visual.

3.3 Overlays

X has a convention for supporting overlay window via special visuals [2]. OpenGL can support rendering into overlay visuals. Even if an X server supports overlay visuals, you will need to make sure those visuals are OpenGL capable. The `glXChooseVisual` routine does allow you to specify the frame buffer layer for the visual you are interested in with the `GLX_LEVEL` attribute. This makes it easier to find OpenGL capable overlay visuals.

IRIS GL programmers are used to assuming the transparent pixel in an overlay visual is always zero. For X and OpenGL, this assumption is no longer valid. You should query the transparent mode and pixel specified by the `SERVER_DVERLAY_VISUALS` property to ensure portability.

IRIS GL programmers are also used to considering overlay planes as being "built-in" to IRIS GL windows. The X model for overlay planes considers an overlay window to be a separate window with its own window ID. To use overlays as one does in IRIS GL, you need to create a

normal plane window, then create a child window in the overlay planes with the child's origin located at the origin of the parent. The child should be maintained to have the same size as the parent. Clear the overlay window to the transparent pixel value to see through to the parent normal plane window. Switching between the overlay and normal planes windows requires a `glXMakeCurrent` call.

It is likely that the overlay visuals will not support the same frame buffer capabilities as the normal plane visuals. You should avoid assuming overlay windows will have frame buffer capabilities such as depth buffers, stencil buffers, or accumulation buffers.

3.4 Mixing Xlib and OpenGL Rendering

In IRIS GL, rendering into an X window using core X rendering after IRIS GL was bound to the window is undefined. This precluded mixing core X rendering with GL rendering in the same window. OpenGL allows its rendering to be mixed with core X rendering into the same window. You should be careful doing so since X and OpenGL rendering requests are logically issued in two distinct streams. If you want to ensure proper rendering, you *must* synchronize the streams. Calling `glXWaitGL` will make sure all OpenGL rendering has finished before subsequent X rendering takes place. Calling `glXWaitX` will make sure all core X rendering has finished before subsequent OpenGL rendering takes place. These requests do not require a protocol round trip to the X server.

The core OpenGL API also includes `glFinish` and `glFlush` commands useful for rendering synchronization. `glFinish` ensures all rendering has appeared on the screen when the routine returns (similar to `XSync`). `glFlush` only ensures the queued commands will eventually be executed (similar to `XFlush`).

Realize that mixing OpenGL and X is not normally necessary. Many OpenGL programs will use a toolkit like Motif for their 2D user interface component and use a distinct X window for OpenGL rendering. This requires no synchronization since OpenGL and core X rendering go to distinct X windows. Only when OpenGL and core X rendering are directed at the same window is synchronization of rendering necessary.

Also OpenGL can be used for extremely fast 2D as well as 3D. When you feel a need to mix core X and OpenGL rendering into the same window, consider rendering what you would do in core X using OpenGL. Not only do you avoid the synchronization overhead, but you can potentially achieve faster 2D using direct rendered OpenGL compared to core X rendering.

3.5 Fonts

Graphics programs often need to display text. You can use X font rendering routines or you can use the GLX

`glXUseXFont` routine to create display lists out of X fonts.

Neither of these methods of font rendering may be flexible enough for a program desiring stroke or scalable fonts or having sophisticated font needs. In the future, an OpenGL font manager will be available to meet these needs. In the meantime, you can use `glXUseXFont` or X font rendering or roll your own font support. An easy way to do this is to convert each glyph of your font into a display list. Rendering text in the font becomes a matter of executing the display list corresponding to each glyph in the string to display.

3.6 Display Lists

OpenGL supports immediate mode rendering where commands can be generated on the fly and sent directly to the screen. Programmers should be aware that their OpenGL programs might be run indirectly. In this case, immediate mode rendering could require a great deal of overhead for transport to the X server and possibly across a network.

For this reason, OpenGL programmers should try to use display lists when possible to batch rendering commands. Since the display lists are stored in the server, executing a display list has minimal overhead compared to executing the same commands in the display list immediately.

Display lists are likely to have other advantages since OpenGL implementations are allowed to compile them for maximum performance. Be aware you can mix display lists and immediate mode rendering to achieve the best mix of performance and rendering flexibility.

4 Conclusion

The `glxdino` example demonstrates the basic tasks that must be done to use OpenGL with X. The program demonstrates sophisticated OpenGL features such as double buffering, lighting, shading, back-face culling, display list modeling, and polygon tessellation. And the proper X conventions are followed to ensure `glxdino` works well with other X programs.

The `glxdino` example program and the hints for advanced OpenGL programming should provide a good foundation for understanding and programming OpenGL with Xlib. The next article will explain how to integrate OpenGL with the Motif toolkit.

A glxdino.c

```
1 /* compile: cc -o glxdino glxdino.c -lGLU -lGL -lXmu -lX11 */
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5 #include <math.h>           /* for cos(), sin(), and sqrt() */
6 #include <GL/glx.h>        /* this includes X and gl.h headers */
7 #include <GL/glu.h>        /* gluPerspective(), gluLookAt(), GLU polygon
8                             * tesselator */
9 #include <X11/Xatom.h>      /* for XA_RGB_DEFAULT_MAP atom */
10 #include <X11/Xmu/StdCmap.h> /* for XmuLookupStandardColormap() */
11 #include <X11/keysym.h>     /* for XK_Escape keysym */

12 typedef enum {
13     RESERVED, BODY_SIDE, BODY_EDGE, BODY_WHOLE, ARM_SIDE, ARM_EDGE, ARM_WHOLE,
14     LEG_SIDE, LEG_EDGE, LEG_WHOLE, EYE_SIDE, EYE_EDGE, EYE_WHOLE, DINOSAUR
15 } displayLists;

16 Display *dpy;
17 Window win;
18 GLfloat angle = -150; /* in degrees */
19 GLboolean doubleBuffer = GL_TRUE, iconic = GL_FALSE, keepAspect = GL_FALSE;
20 int W = 300, H = 300;
21 XSizeHints sizeHints = {0};
22 GLdouble bodyWidth = 2.0;
23 int configuration[] = {GLX_DOUBLEBUFFER, GLX_RGBA, GLX_DEPTH_SIZE, 16, None};
24 GLfloat body[][2] = { {0, 3}, {1, 1}, {5, 1}, {8, 4}, {10, 4}, {11, 5},
25     {11, 11.5}, {13, 12}, {13, 13}, {10, 13.5}, {13, 14}, {13, 15}, {11, 16},
26     {8, 16}, {7, 15}, {7, 13}, {8, 12}, {7, 11}, {6, 6}, {4, 3}, {3, 2},
27     {1, 2} };
28 GLfloat arm[][2] = { {8, 10}, {9, 9}, {10, 9}, {13, 8}, {14, 9}, {16, 9},
29     {15, 9.5}, {16, 10}, {15, 10}, {15.5, 11}, {14.5, 10}, {14, 11}, {14, 10},
30     {13, 9}, {11, 11}, {9, 11} };
31 GLfloat leg[][2] = { {8, 6}, {8, 4}, {9, 3}, {9, 2}, {8, 1}, {8, 0.5}, {9, 0},
32     {12, 0}, {10, 1}, {10, 2}, {12, 4}, {11, 6}, {10, 7}, {9, 7} };
33 GLfloat eye[][2] = { {8.75, 15}, {9, 14.7}, {9.6, 14.7}, {10.1, 15},
34     {9.6, 15.25}, {9, 15.25} };
35 GLfloat lightZeroPosition[] = {10.0, 4.0, 10.0, 1.0};
36 GLfloat lightZeroColor[] = {0.8, 1.0, 0.8, 1.0}; /* green-tinted */
37 GLfloat lightOnePosition[] = {-1.0, -2.0, 1.0, 0.0};
38 GLfloat lightOneColor[] = {0.6, 0.3, 0.2, 1.0}; /* red-tinted */
39 GLfloat skinColor[] = {0.1, 1.0, 0.1, 1.0}, eyeColor[] = {1.0, 0.2, 0.2, 1.0};

40 void
41 fatalError(char *message)
42 {
43     fprintf(stderr, "glxdino: %s\n", message);
44     exit(1);
45 }

46 Colormap
47 getColormap(XVisualInfo *vi)
48 {
49     Status          status;
50     XStandardColormap *standardCmaps;
51     Colormap        cmap;
52     int              i, numCmaps;

53     /* be lazy; using DirectColor too involved for this example */
```

```

54     if (vi->class != TrueColor)
55         fatalError("no support for non-TrueColor visual");
56     /* if no standard colormap but TrueColor, just make an unshared one */
57     status = XmuLookupStandardColormap(dpy, vi->screen, vi->visualid,
58         vi->depth, XA_RGB_DEFAULT_MAP, /* replace */ False, /* retain */ True);
59     if (status == 1) {
60         status = XGetRGBColormaps(dpy, RootWindow(dpy, vi->screen),
61             &standardCmaps, &numCmaps, XA_RGB_DEFAULT_MAP);
62         if (status == 1)
63             for (i = 0; i < numCmaps; i++)
64                 if (standardCmaps[i].visualid == vi->visualid) {
65                     cmap = standardCmaps[i].colormap;
66                     XFree(standardCmaps);
67                     return cmap;
68                 }
69     }
70     cmap = XCreateColormap(dpy, RootWindow(dpy, vi->screen),
71         vi->visual, AllocNone);
72     return cmap;
73 }

74 void
75 extrudeSolidFromPolygon(GLfloat data[][2], unsigned int dataSize,
76     GLdouble thickness, GLuint side, GLuint edge, GLuint whole)
77 {
78     static GLUttriangulatorObj *tobj = NULL;
79     GLdouble     vertex[3], dx, dy, len;
80     int          i;
81     int          count = dataSize / (2 * sizeof(GLfloat));

82     if (tobj == NULL) {
83         tobj = gluNewTess();    /* create and initialize a GLU polygon
84                                 * tessellation object */
85         gluTessCallback(tobj, GLU_BEGIN, glBegin);
86         gluTessCallback(tobj, GLU_VERTEX, glVertex2fv); /* semi-tricky */
87         gluTessCallback(tobj, GLU_END, glEnd);
88     }
89     glNewList(side, GL_COMPILE);
90     glShadeModel(GL_SMOOTH); /* smooth minimizes seeing tessellation */
91     gluBeginPolygon(tobj);
92     for (i = 0; i < count; i++) {
93         vertex[0] = data[i][0];
94         vertex[1] = data[i][1];
95         vertex[2] = 0;
96         gluTessVertex(tobj, vertex, &data[i]);
97     }
98     gluEndPolygon(tobj);
99     glEndList();
100    glNewList(edge, GL_COMPILE);
101    glShadeModel(GL_FLAT); /* flat shade keeps angular hands from being
102                            * "smoothed" */
103    glBegin(GL_QUAD_STRIP);
104    for (i = 0; i <= count; i++) {
105        /* mod function handles closing the edge */
106        glVertex3f(data[i % count][0], data[i % count][1], 0.0);
107        glVertex3f(data[i % count][0], data[i % count][1], thickness);
108        /* Calculate a unit normal by dividing by Euclidean distance. We
109         * could be lazy and use glEnable(GL_NORMALIZE) so we could pass in
110         * arbitrary normals for a very slight performance hit. */
111        dx = data[(i + 1) % count][1] - data[i % count][1];

```

```

112         dy = data[i % count][0] - data[(i + 1) % count][0];
113         len = sqrt(dx * dx + dy * dy);
114         glNormal3f(dx / len, dy / len, 0.0);
115     }
116     glEnd();
117 glEndList();
118 glNewList(whole, GL_COMPILE);
119     glFrontFace(GL_CW);
120     glCallList(edge);
121     glNormal3f(0.0, 0.0, -1.0); /* constant normal for side */
122     glCallList(side);
123     glPushMatrix();
124         glTranslatef(0.0, 0.0, thickness);
125         glFrontFace(GL_CCW);
126         glNormal3f(0.0, 0.0, 1.0); /* opposite normal for other side */
127         glCallList(side);
128     glPopMatrix();
129 glEndList();
130 }

131 void
132 makeDinosaur(void)
133 {
134     GLfloat        bodyWidth = 3.0;

135     extrudeSolidFromPolygon(body, sizeof(body), bodyWidth,
136         BODY_SIDE, BODY_EDGE, BODY_WHOLE);
137     extrudeSolidFromPolygon(arm, sizeof(arm), bodyWidth / 4,
138         ARM_SIDE, ARM_EDGE, ARM_WHOLE);
139     extrudeSolidFromPolygon(leg, sizeof(leg), bodyWidth / 2,
140         LEG_SIDE, LEG_EDGE, LEG_WHOLE);
141     extrudeSolidFromPolygon(eye, sizeof(eye), bodyWidth + 0.2,
142         EYE_SIDE, EYE_EDGE, EYE_WHOLE);
143     glNewList(DINOSAUR, GL_COMPILE);
144         glMaterialfv(GL_FRONT, GL_DIFFUSE, skinColor);
145         glCallList(BODY_WHOLE);
146         glPushMatrix();
147             glTranslatef(0.0, 0.0, bodyWidth);
148             glCallList(ARM_WHOLE);
149             glCallList(LEG_WHOLE);
150             glTranslatef(0.0, 0.0, -bodyWidth - bodyWidth / 4);
151             glCallList(ARM_WHOLE);
152             glTranslatef(0.0, 0.0, -bodyWidth / 4);
153             glCallList(LEG_WHOLE);
154             glTranslatef(0.0, 0.0, bodyWidth / 2 - 0.1);
155             glMaterialfv(GL_FRONT, GL_DIFFUSE, eyeColor);
156             glCallList(EYE_WHOLE);
157         glPopMatrix();
158     glEndList();
159 }

160 void
161 redraw(void)
162 {
163     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
164     glCallList(DINOSAUR);
165     if (doubleBuffer)
166         glXSwapBuffers(dpy, win); /* buffer swap does implicit glFlush */
167     else glFlush(); /* explicit flush for single buffered case */
168 }

```

```

169 void
170 main(int argc, char **argv)
171 {
172     XVisualInfo    *vi;
173     Colormap      cmap;
174     XSetWindowAttributes swa;
175     XWMHints      *wmHints;
176     Atom          wmDeleteWindow;
177     GLXContext    cx;
178     XEvent        event;
179     KeySym        ks;
180     GLboolean     needRedraw = GL_FALSE, recalcModelView = GL_TRUE;
181     char          *display = NULL, *geometry = NULL;
182     int           flags, x, y, width, height, lastX, i;

183     /** (1) process normal X command line arguments ***/
184     for (i = 1; i < argc; i++) {
185         if (!strcmp(argv[i], "-geometry")) {
186             if (++i >= argc)
187                 fatalError("follow -geometry option with geometry parameter");
188             geometry = argv[i];
189         } else if (!strcmp(argv[i], "-display")) {
190             if (++i >= argc)
191                 fatalError("follow -display option with display parameter");
192             display = argv[i];
193         } else if (!strcmp(argv[i], "-iconic")) iconic = GL_TRUE;
194         else if (!strcmp(argv[i], "-keepaspect")) keepAspect = GL_TRUE;
195         else if (!strcmp(argv[i], "-single")) doubleBuffer = GL_FALSE;
196         else fatalError("bad option");
197     }

198     /** (2) open a connection to the X server ***/
199     dpy = XOpenDisplay(display);
200     if (dpy == NULL) fatalError("could not open display");

201     /** (3) make sure OpenGL's GLX extension supported ***/
202     if (!glXQueryExtension(dpy, NULL, NULL))
203         fatalError("X server has no OpenGL GLX extension");

204     /** (4) find an appropriate visual and a colormap for it ***/
205     /* find an OpenGL-capable RGB visual with depth buffer */
206     if (!doubleBuffer) goto SingleBufferOverride;
207     vi = glXChooseVisual(dpy, DefaultScreen(dpy), configuration);
208     if (vi == NULL) {
209         SingleBufferOverride:
210         vi = glXChooseVisual(dpy, DefaultScreen(dpy), &configuration[1]);
211         if (vi == NULL)
212             fatalError("no appropriate RGB visual with depth buffer");
213         doubleBuffer = GL_FALSE;
214     }
215     cmap = getColormap(vi);

216     /** (5) create an OpenGL rendering context ***/
217     /* create an OpenGL rendering context */
218     cx = glXCreateContext(dpy, vi, /* no sharing of display lists */ NULL,
219                          /* direct rendering if possible */ GL_TRUE);
220     if (cx == NULL) fatalError("could not create rendering context");

221     /** (6) create an X window with selected visual and right properties ***/

```

```

222 flags = XParseGeometry(geometry, &x, &y,
223     (unsigned int *) &width, (unsigned int *) &height);
224 if (WidthValue & flags) {
225     sizeHints.flags |= USSize;
226     sizeHints.width = width;
227     W = width;
228 }
229 if (HeightValue & flags) {
230     sizeHints.flags |= USSize;
231     sizeHints.height = height;
232     H = height;
233 }
234 if (XValue & flags) {
235     if (XNegative & flags)
236         x = DisplayWidth(dpy, DefaultScreen(dpy)) + x - sizeHints.width;
237     sizeHints.flags |= USPosition;
238     sizeHints.x = x;
239 }
240 if (YValue & flags) {
241     if (YNegative & flags)
242         y = DisplayHeight(dpy, DefaultScreen(dpy)) + y - sizeHints.height;
243     sizeHints.flags |= USPosition;
244     sizeHints.y = y;
245 }
246 if (keepAspect) {
247     sizeHints.flags |= PAspect;
248     sizeHints.min_aspect.x = sizeHints.max_aspect.x = W;
249     sizeHints.min_aspect.y = sizeHints.max_aspect.y = H;
250 }
251 swa.colormap = cmap;
252 swa.border_pixel = 0;
253 swa.event_mask = ExposureMask | StructureNotifyMask |
254     ButtonPressMask | Button1MotionMask | KeyPressMask;
255 win = XCreateWindow(dpy, RootWindow(dpy, vi->screen),
256     sizeHints.x, sizeHints.y, W, H,
257     0, vi->depth, InputOutput, vi->visual,
258     CWBorderPixel | CWColormap | CWEventMask, &swa);
259 XSetStandardProperties(dpy, win, "OpenGLosaurus", "glxdino",
260     None, argv, argc, &sizeHints);
261 wmHints = XAllocWMHints();
262 wmHints->initial_state = iconic ? IconicState : NormalState;
263 wmHints->flags = StateHint;
264 XSetWMHints(dpy, win, wmHints);
265 wmDeleteWindow = XInternAtom(dpy, "WM_DELETE_WINDOW", False);
266 XSetWMPprotocols(dpy, win, &wmDeleteWindow, 1);

267 /* (7) bind the rendering context to the window */
268 glXMakeCurrent(dpy, win, cx);

269 /* (8) make the desired display lists */
270 makeDinosaur();

271 /* (9) configure the OpenGL context for rendering */
272 glEnable(GL_CULL_FACE); /* ~50% better performance than no back-face
273     * culling on Entry Indigo */
274 glEnable(GL_DEPTH_TEST); /* enable depth buffering */
275 glEnable(GL_LIGHTING); /* enable lighting */
276 glMatrixMode(GL_PROJECTION); /* set up projection transform */
277 gluPerspective( /* field of view in degree */ 40.0, /* aspect ratio */ 1.0,
278     /* Z near */ 1.0, /* Z far */ 40.0);

```

```

279     glMatrixMode(GL_MODELVIEW); /* now change to modelview */
280     gluLookAt(0.0, 0.0, 30.0, /* eye is at (0,0,30) */
281              0.0, 0.0, 0.0, /* center is at (0,0,0) */
282              0.0, 1.0, 0.); /* up is in postivie Y direction */
283     glPushMatrix(); /* dummy push so we can pop on model recalc */
284     glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, 1);
285     glLightfv(GL_LIGHT0, GL_POSITION, lightZeroPosition);
286     glLightfv(GL_LIGHT0, GL_DIFFUSE, lightZeroColor);
287     glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 0.1);
288     glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 0.05);
289     glLightfv(GL_LIGHT1, GL_POSITION, lightOnePosition);
290     glLightfv(GL_LIGHT1, GL_DIFFUSE, lightOneColor);
291     glEnable(GL_LIGHT0);
292     glEnable(GL_LIGHT1); /* enable both lights */

293     /*** (10) request the X window to be displayed on the screen ***/
294     XMapWindow(dpy, win);

295     /*** (11) dispatch X events ***/
296     while (1) {
297         do {
298             XNextEvent(dpy, &event);
299             switch (event.type) {
300                 case ConfigureNotify:
301                     glViewport(0, 0,
302                                event.xconfigure.width, event.xconfigure.height);
303                     /* fall through... */
304                 case Expose:
305                     needRedraw = GL_TRUE;
306                     break;
307                 case MotionNotify:
308                     recalcModelView = GL_TRUE;
309                     angle -= (lastX - event.xmotion.x);
310                 case ButtonPress:
311                     lastX = event.xbutton.x;
312                     break;
313                 case KeyPress:
314                     ks = XLookupKeysym((XKeyEvent *) & event, 0);
315                     if (ks == XK_Escape) exit(0);
316                     break;
317                 case ClientMessage:
318                     if (event.xclient.data.l[0] == wmDeleteWindow) exit(0);
319                     break;
320             }
321         } while (XPending(dpy)); /* loop to compress events */
322         if (recalcModelView) {
323             glPopMatrix(); /* pop old rotated matrix (or dummy matrix if
324                            * first time) */
325             glPushMatrix();
326             glRotatef(angle, 0.0, 1.0, 0.0);
327             glTranslatef(-8, -8, -bodyWidth / 2);
328             recalcModelView = GL_FALSE;
329             needRedraw = GL_TRUE;
330         }
331         if (needRedraw) {
332             redraw();
333             needRedraw = GL_FALSE;
334         }
335     }
336 }

```

References

- [1] James Foley, Andries van Dam, Steven Feiner, and John Hughes, *Computer Graphics: Principles and Practice*, 2nd edition, Addison-Wesley Publishing, 1990.
- [2] Mark Kilgard, "Programming X Overlay Windows," *The X Journal*, SIGS Publications, July 1993.
- [3] Jackie Neider, Tom Davis, Mason Woo, *OpenGL Programming Guide: The official guide to learning OpenGL, Release 1*. Addison Wesley, 1993.
- [4] OpenGL Architecture Review Board, *OpenGL Reference Manual: The official reference document for OpenGL, Release 1*. Addison Wesley, 1992.

OpenGL™ and X, Part 3: Integrating OpenGL with Motif

Mark J. Kilgard *
Silicon Graphics Inc.
Revision : 1.18

March 31, 1994

Abstract

The OpenGL™ graphics system can be integrated with the industry-standard OSF/Motif user interface. This article discusses how to use OpenGL within a Motif application program. There are two approaches to using OpenGL with Motif. One is to render into a standard Motif drawing area widget, but this requires each application window to use a single visual for its window hierarchy. A better approach is to use the special OpenGL drawing area widget allowing windows used for OpenGL rendering to pick freely an appropriate visual without affecting the visual choice for other widgets. An example program demonstrates both approaches. The X Toolkit's work procedure mechanism animates the example's 3D paper airplanes. Handling OpenGL errors is also explained.

1 Introduction

OSF/Motif is the X Window System's industry-standard programming interface for user interface construction. Motif programmers writing 3D applications will want to understand how to integrate Motif with the OpenGL™ graphics system. This article, the last in a three-part series about OpenGL, describes how to write an OpenGL program within the user interface framework provided by Motif and the X Toolkit.

Most 3D applications end up using 3D graphics primarily in one or more "viewing" windows. For the most part, the graphical user interface aspects of such programs use standard 2D user interface objects like pulldown menus, sliders, and dialog boxes. Creating and managing such common user interface objects is what Motif does well. The "viewing" windows used for 3D are where OpenGL

rendering happens. These windows for OpenGL rendering can be constructed with standard Motif drawing area widgets or OpenGL-specific drawing area widgets. Bind an OpenGL rendering context to the window of a drawing area widget and you are ready for 3D rendering.

Programming OpenGL with Motif has numerous advantages over using "Xlib only" as described in the first two articles in this series [2, 3]. First and most important, Motif provides a well-documented, standard widget set that gives your application a consistent look and feel. Second, Motif and the X Toolkit take care of routine but complicated issues such as *cut and paste* and window manager conventions. Third, the X Toolkit's work procedure and timeout mechanisms make it easy to animate a 3D window without blocking out user interaction with your application.

This article assumes you have some experience programming with Motif and you have a basic understanding of how OpenGL integrates with X.

Section 2 describes how to use OpenGL rendering with either a standard Motif drawing area widget or an OpenGL-specific drawing area widget. Section 3 discusses using X Toolkit mechanisms for seamless animation. Section 4 provides advice on how to debug OpenGL programs by catching OpenGL errors. Throughout the discussion, a Motif-based OpenGL program named *paperplane* is used as an example. The complete source code for *paperplane* is found in the appendix. The program animates the 3D flight paths of virtual paper airplanes. The user can interact with the program via Motif controls. The program can be compiled to use either a standard Motif drawing area widget or an OpenGL-specific drawing area widget. Figure 1 shows *paperplane* running.

*Mark graduated with B.A. in Computer Science from Rice University and is a Member of the Technical Staff at Silicon Graphics. He can be reached by electronic mail addressed to mjk@sgi.com

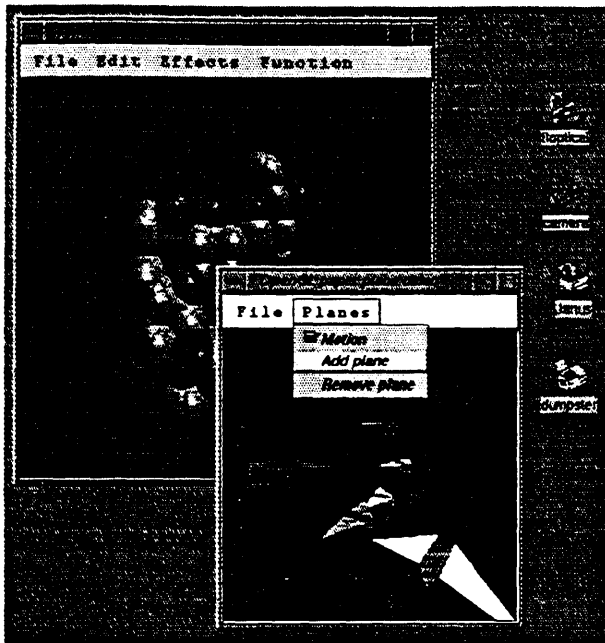


Figure 1: Screen snapshot of `paperplane` with another OpenGL Motif program for molecular modeling.

2 OpenGL with Widgets

Your application's 3D viewing area can be encapsulated by an X Toolkit widget. There are two approaches to rendering OpenGL into a widget. You can render OpenGL into a standard Motif drawing area widget, or you can use a special OpenGL drawing area widget.

The Motif drawing area widget would seem a natural widget for OpenGL rendering. Unfortunately, the X Toolkit's design (upon which Motif relies) allows programmers to specify a widget's visual only if its class is derived from the shell widget class. Shell widgets are often called "top level" widgets because they are designed to communicate with the window manager and act as containers for other widgets. Non-shell widgets inherit the depth and visual of their parent widget. The Motif drawing area widget class (like most widget classes) is not derived from the shell widget class. It is impossible (without resorting to programming widget internals) to set the visual of a standard non-shell Motif widget differently than its ancestor shell widget.

But in OpenGL, the X notion of a visual has expanded importance for determining the OpenGL frame buffer capabilities of an X window. In many cases, an application's 3D viewing area is likely to demand a deeper, more capable visual than the default visual which Motif normally uses.

There are two options:

1. Use the standard Motif drawing area widget for your OpenGL rendering area and make sure that the top

level shell widget is created with the desired visual for OpenGL's use.

2. Use an OpenGL drawing area widget that is specially programmed to overcome the limitation on setting the visual and depth of a non-shell widget.

Either approach works.

The `paperplane` example in the appendix is written to support either scheme depending on how the code is compiled. By default, the code compiles to use the OpenGL-specific widget. If the `noGLwidget` C preprocessor symbol is defined, the standard Motif drawing area widget will be used, forcing the use of a single visual throughout the example's widget hierarchy. The code differences between the two schemes in the `paperplane` example constitute seven changed lines of code.

The preferable approach is to use the OpenGL-specific widget, since you can run most of the application's user interface in the default visual and use a deeper, more capable visual only for 3D viewing areas. Limiting the use of deeper visuals can save memory and increase rendering speed for the user interface windows. If you use a 24-bit visual for your 3D viewing area and use the same visual for your entire application, that means that the image memory for pixmap used by non-OpenGL windows is four times what it would be for an 8-bit visual.¹ Some X rendering operations might also be slower for 24-bit windows compared with 8-bit windows.

There can be advantages to running your entire application in a single visual. Some workstations with limited colormap resources might not be capable of showing multiple visuals without colormap flashing. Such machines which support OpenGL should be rare. Even if running in a single visual is appropriate, nothing precludes doing so using an OpenGL-specific widget.

2.1 The OpenGL-specific Widget

There are two OpenGL-specific drawing area widget classes. One is derived from the Motif primitive widget class (*not* the Motif drawing area widget class). The other is derived from the X Toolkit core widget class. Both have the same basic functionality; the main difference is that the Motif-based widget class gains capabilities of the Motif primitive widget class. If you use Motif, you should use the Motif OpenGL widget. If you use a non-Motif widget set, you can use the second widget for identical functionality.

The Motif OpenGL widget class is named `glwMDrawingAreaWidgetClass`; the non-Motif OpenGL widget class is named `glwDrawingAreaWidgetClass` (the difference is the lack of an `M` in the non-Motif case). Since

¹Even though a 24-bit pixel requires only three bytes of storage, efficient manipulation of the pixels demands each pixel is stored in an even 4 bytes.

the Motif OpenGL widget is subclassed from the Motif primitive widget class. the Motif OpenGL widget inherits the capabilities of the primitive class like a help callback and keyboard traversal support (keyboard traversal is disabled by default for the Motif OpenGL widget). The `paperplane` example uses the Motif widget by default but the non-Motif widget can be used by defining the `noMotifGLWidget` C preprocessor symbol when compiling `paperplane.c`. The difference is two changed lines of code with no functional difference in the program.

When you create either type of widget, you need to specify the visual to use by supplying the widget's `GLWVisualInfo` resource. The attribute is of type `XVisualInfo*` making it easy to find an appropriate visual using `glXChooseVisual` which returns a `XVisualInfo*` for a visual with the capabilities you request.

Although this practice is not recommended, the widgets also allow you to specify the OpenGL capabilities you desire for the widget directly using widget resources. Because the X Toolkit widget creation process is not expected to fail, there is no way for a widget creation routine to indicate failure. If a visual that matches the desired OpenGL capabilities cannot be found, the widget code prints an error and exits without giving the program a chance to handle the failure. If you request a specific `XVisualInfo*` that has already been determined to be acceptable using `glXChooseVisual` or calls to `glXGetConfig`, you will not have this problem. As a rule, always specify the visual using the `GLWVisualInfo` resource.

The OpenGL widgets also do extra work that might go unnoticed. Because the OpenGL widget uses a different visual, the widget's creation code creates a colormap matching the visual. It also posts an `ICCCM WM_COLORMAP_WINDOWS` top level window property to let the window manager know that the program uses multiple colormaps.

More information about the OpenGL widgets can be found in the Silicon Graphics *OpenGL Porting Guide* [4] and the widgets' man pages.

2.2 The Motif Drawing Area Widget

Using the standard Motif drawing area widget with OpenGL has some extra caveats. The main caveat is that you must create the top level widget with the correct visual for the program's OpenGL rendering.

When you start a widget program, there is generally a call to `XtAppInitialize` to establish the connection to the X server and create the top level widget. Both steps are done in the same routine. So how can we call `glXChooseVisual` to know what visual the top level widget should use until we have established a connection to the X server?

It would appear that it is impossible to create the top level widget with an appropriate visual for OpenGL.

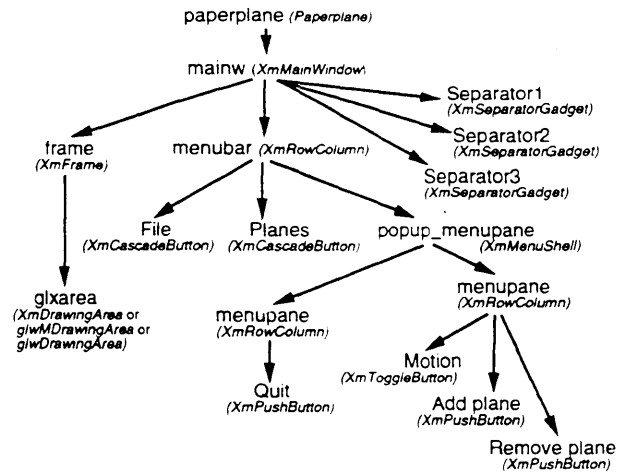


Figure 2: Diagram of the widget hierarchy for `paperplane`. The `glxarea XmDrawingArea` widget is the only widget rendered using OpenGL.

`XtAppInitialize` connects to the X server and creates the top level widget, but it does not *realize* the top level widget. The X window for the top level widget is not created until `XtRealizeWidget` is called. This allows `XtSetValues` to be used after the top level widget's creation (and before its realization) to specify the widget's visual. The `paperplane` sample code in the non-OpenGL widget case demonstrates this.

A second caveat is due to the X Toolkit's inconsistent inheritance of the visual, depth, and colormap widget resources. The default visual of a widget's window is copied from its *parent window's* visual. But the default colormap and depth of a widget are copied from the widget's *parent widget*.²

This means that if you create a widget derived from the shell widget and the widget's parent uses a non-default depth or colormap for a non-default visual, you will need to specify the same visual as the new widget's parent widget. If you do not, a `BadMatch` X protocol error will result. For this reason the `paperplane` example's `XmCreatePulldownMenu` calls specify the visual of the created widget's parent widget in the Motif drawing area version of `paperplane`.

Realize that it is not possible to bind an OpenGL rendering context to a widget's window until the widget has been realized. Until the widget is realized, the widget's window does not yet exist. Notice `paperplane` does not call `glXMakeCurrent` until after `XtRealizeWidget` has been called.

To see how the 3D viewing area widget fits into the `paperplane` widget hierarchy example, Figure 2 shows the complete hierarchy including widget class names.

²If the widget has no parent, the depth and colormap are determined by the default depth and colormap of the screen.

These caveats are not unique to OpenGL. The problem comes from using non-default visuals with the X Toolkit. PEXlib 5.1 programs have a similar need for non-default visuals and require the same jumping through hoops[1]. Fortunately, if you use the OpenGL drawing area widgets, you can avoid the caveats of using the standard Motif drawing area.

2.3 Drawing Area Callbacks

Applications using the Motif drawing area widget or the OpenGL drawing area widgets for their 3D rendering will want to register routines to handle expose, resize, and input callbacks using `XtAddCallback`. In `paperplane.c`, the `draw`, `resize`, and `input` routines handle these callbacks.

`paperplane`'s drawing area adjusts OpenGL's viewport by calling `glViewport`. Note how the `made_current` variable is used to protect against calling `glViewport` before we have done the `glXMakeCurrent` to bind to the drawing area window. In the X Toolkit, the `resize` callback can be called before the `XtRealizeWidget` routine returns. Since the program does not call `glXMakeCurrent` until after the program returns from `XtRealizeWidget`, the OpenGL rendering context would not be bound. Calling an OpenGL routine before a context is bound has no effect but generates an ugly warning message.³ An example of when the `resize` callback can be called before `XtRealizeWidget` returns is when a `-geometry` command line option is specified.

Note that `glXMakeCurrent` is defined to set a context's viewport to the size of the first window it is bound to. (This happens only on the context's first bind.) This is why `paperplane.c` makes no initial call to `glViewport`; `glXMakeCurrent` sets the viewport implicitly.

The `paperplane` example uses a single window for OpenGL rendering. For this reason, `glXMakeCurrent` is called only once to bind the OpenGL context to the window. In a program with multiple OpenGL windows, each `expose` and `resize` callback should make sure that `glXMakeCurrent` is called so that OpenGL rendering goes to the correct window.

The `draw` callback routine issues the OpenGL commands to draw the scene. If the window is double buffered, `glXSwapBuffers` swaps the window's buffers. If the context is not direct, `glFinish` is called to avoid the latency from queuing more than one frame at a time; interactivity would suffer if we allowed more than one frame to be queued. Direct rendering involves direct manipulation of the hardware so it generally has less latency than a potentially networked indirect OpenGL context.

Note that you can render OpenGL into *any* widget (as long as it is created with an OpenGL capable visual).

³The exact behavior is undefined by the OpenGL specification.

There is nothing special about the Motif or OpenGL-specific drawing area widgets, though drawing area widgets tend to be the most appropriate widget type for a 3D viewing area.

2.4 Handling Input

The `input` routine handles X events for the drawing area. Input events require no special handling for OpenGL. But remember that the coordinate systems for X and OpenGL are distinct, so pointer locations need to be mapped into OpenGL's coordinate space. OpenGL generally assumes that the origin is in the lower left-hand corner, while X always assumes an origin at the upper left-hand corner.

3 Animation Via Work Procedures

The X Toolkit's work procedure facility makes it easy to integrate continuous OpenGL animation with Motif user interface operation. Work procedures are application supplied routines that execute while the application is idle waiting for events. Work procedures should be used to do small amounts of work; if too much time is spent in a work procedure, X events will not be processed and program interactivity will suffer.

Rendering a single frame of OpenGL animation is a good use for work procedures. `XtAppAddWorkProc` and `XtRemoveWorkProc` are used to add and remove work procedures. `XtAppAddWorkProc` is passed a function pointer for the routine to be called as a work procedure. The function to be called returns a `Boolean`. If the function returns `True`, the work procedure should be removed automatically; returning `False` indicates the work procedure should remain active. `XtAppAddWorkProc` returns an ID of type `WorkProcId` which can later be given to `XtRemoveWorkProc` to remove the work procedure.

The `paperplane` example uses a work procedure to manage the update of its 3D scene. In response to changing the state of the "Motion" toggle button on the "Planes" pulldown menu, the `toggle` callback routine will add and remove the `animate` work procedure.

The `animate` routine calls `tick` which advances the position of each active plane; `animate` then calls `draw` to redraw the scene with the new plane locations. Finally, `animate` returns `False` to leave the work procedure installed so that the animation will continue.

Because `paperplane` uses a work procedure, animation of the scene does not interfere with window resizing and user input. The X Toolkit manages both the animation and events from the X server.

3.1 Handling Iconification

When the `paperplane` window is open, we want the `animate` work procedure to update the 3D scene continuously. If the user iconifies the window, it would be wasteful to continue animating a no longer visible scene. To avoid wasting resources rendering to an unmapped window, `paperplane` installs an event handler called `map_state_changed` for the top-level widget to notice `UnmapNotify` and `MapNotify` events. The handler makes sure the work procedure is removed or added to reflect the map state of the window.

3.2 Timeouts

X Toolkit timeouts are similar to work procedures, but instead of being activated whenever event dispatching is idle, they are called when a given period of time has expired. The `XtAppAddTimeout` and `XtRemoveTimeOut` routines can be used to add and remove X Toolkit timeouts.

OpenGL programmers may find timeouts useful to maintain animation at rates slower than "as fast as OpenGL will render." Timeouts can be used to give animation a sustained frame rate. Timeouts can also be used to redraw a scene with higher detail when the user has stopped interacting with the program. For example, a 3D modeling program might redraw its model with lighting enabled and finer tessellation after the program has been idle for two seconds. Timeouts can also be used to trigger simple real-time state changes useful for visual simulation.

4 Debugging Tips

As well as demonstrating the use of widgets with OpenGL, `paperplane` also demonstrates detection of OpenGL errors for debugging purposes. Some debugging code has been added to the bottom of `paperplane`'s `draw` function to test for any OpenGL errors. A correct OpenGL program should not generate any OpenGL errors, but while debugging it is helpful to check explicitly for errors. A good time to check for errors is at the end of each frame. Errors in OpenGL are not reported unless you explicitly check for them, unlike X protocol errors which are always reported to the client.

OpenGL errors are recorded by setting "sticky" flags. Once an error flag is set, it will not be cleared until `glGetError` is used to query the error. An OpenGL implementation may have several error flags internally that can be set (since OpenGL errors might occur in different stages of the OpenGL rendering pipeline). When you look for errors, you should call `glGetError` repeatedly until it returns `GL_NO_ERROR` indicating that all of the error flags have been cleared.

The OpenGL error model is suited for high performance rendering, since error reporting does not slow down the

error-free case. Because OpenGL errors should not be generated by bug-free code, you probably want to remove error querying from your final program since querying errors will slow down your rendering speed.

When an OpenGL error is generated, the command which generated the error is not recorded, so you may need to add more error queries into your code to isolate the source of the error.

The `gluErrorString` routine in the OpenGL Utility library (GLU) converts an OpenGL error number into a human readable string and helps you output a reasonable error message.

5 Conclusion

OpenGL and Motif are a powerful combination. Using both APIs allow X applications programmers to get the most out of both Motif and OpenGL.

Still another way to integrate OpenGL rendering with widgets is the Open Inventor object-oriented 3D graphics toolkit which renders using OpenGL and integrates with X Toolkit widgets. Open Inventor allows you to specify 3D scenes in an object-oriented fashion instead of low-level OpenGL rendering primitives. If you are interested in object-oriented 3D, check out the recently published *Inventor Mentor* [5].

The source code presented in this series is available by anonymous ftp to `sgigate.sgi.com` in the `pub/opengl/xjournal` directory.

Acknowledgments

Writing these three articles on OpenGL required the assistance from numerous engineers and managers at Silicon Graphics. In particular I would like to thank Kurt Akeley, David Blythe, Simon Hui, Phil Karlton, Mark Segal, Kevin Smith, Joel Tesler, Tom Weinstein, Mason Woo, and David Yu.

A paperplane.c

```
1 /*
2  * paperplane can be compiled to use a "single visual" for the entire window
3  * hierarchy and render OpenGL into a standard Motif drawing area widget:
4  *
5  * cc -o sv_paperplane paperplane.c -DnoGLwidget -lGL -lXm -lXt -lX11 -lm
6  *
7  * Or paperplane can be compiled to use the default visual for most of
8  * the window hierarchy but render OpenGL into a special "OpenGL widget":
9  *
10 * cc -o glw_paperplane paperplane.c -lGLw -lGL -lXm -lXt -lX11 -lm
11 */
12 #include <stdlib.h>
13 #include <stdio.h>
14 #include <unistd.h>
15 #include <math.h>
16 #include <Xm/MainW.h>
17 #include <Xm/RowColumn.h>
18 #include <Xm/PushB.h>
19 #include <Xm/ToggleB.h>
20 #include <Xm/CascadeB.h>
21 #include <Xm/Frame.h>
22 #ifdef noGLwidget
23 #include <Xm/DrawingA.h>      /* Motif drawing area widget */
24 #else
25 #ifdef noMotifGLwidget
26 #include <GL/GLwDrawA.h>     /* pure Xt OpenGL drawing area widget */
27 #else
28 #include <GL/GLwMDrawA.h>    /* Motif OpenGL drawing area widget */
29 #endif
30 #endif
31 #include <X11/keysym.h>
32 #include <GL/gl.h>
33 #include <GL/glu.h>
34 #include <GL/glx.h>
35 static int dblBuf[] = {
36     GLX_DOUBLEBUFFER, GLX_RGBA, GLX_DEPTH_SIZE, 16,
37     GLX_RED_SIZE, 1, GLX_GREEN_SIZE, 1, GLX_BLUE_SIZE, 1,
38     None
39 };
40 static int *snglBuf = &dblBuf[1];
41 static String fallbackResources[] = {
42 #ifdef IRIX_5_2_or_higher
43     "*sgiMode: true",          /* try to enable IRIX 5.2+ look & feel */
44     "*useSchemes: all",       /* and SGI schemes */
45 #endif
46     "**title: OpenGL paper plane demo",
47     "**glxarea*width: 300", "**glxarea*height: 300", NULL
48 };
49 Display      *dpy;
50 GLboolean    doubleBuffer = GL_TRUE, moving = GL_FALSE, made_current = GL_FALSE;
51 XtAppContext app;
52 XtWorkProcId workId = 0;
53 Widget       toplevel, mainw, menubar, menupane, btn, cascade, frame, glxarea;
54 GLXContext   cx;
55 XVisualInfo  *vi;
56 #ifdef noGLwidget
57 Colormap     cmap;
```

```

58 #endif
59 Arg      menuPaneArgs[1], args[1];

60 #define MAX_PLANES 15

61 struct {
62     float      speed;      /* zero speed means not flying */
63     GLfloat    red, green, blue;
64     float      theta;
65     float      x, y, z, angle;
66 } planes[MAX_PLANES];

67 #define v3f glVertex3f /* v3f was the short IRIS GL name for glVertex3f */

68 void draw(Widget w)
69 {
70     GLfloat    red, green, blue;
71     int        i;

72     glClear(GL_DEPTH_BUFFER_BIT);
73     /* paint black to blue smooth shaded polygon for background */
74     glDisable(GL_DEPTH_TEST);
75     glShadeModel(GL_SMOOTH);
76     glBegin(GL_POLYGON);
77         glColor3f(0.0, 0.0, 0.0);
78         v3f(-20, 20, -19); v3f(20, 20, -19);
79         glColor3f(0.0, 0.0, 1.0);
80         v3f(20, -20, -19); v3f(-20, -20, -19);
81     glEnd();
82     /* paint planes */
83     glEnable(GL_DEPTH_TEST);
84     glShadeModel(GL_FLAT);
85     for (i = 0; i < MAX_PLANES; i++)
86         if (planes[i].speed != 0.0) {
87             glPushMatrix();
88             glTranslatef(planes[i].x, planes[i].y, planes[i].z);
89             glRotatef(290.0, 1.0, 0.0, 0.0);
90             glRotatef(planes[i].angle, 0.0, 0.0, 1.0);
91             glScalef(1.0 / 3.0, 1.0 / 4.0, 1.0 / 4.0);
92             glTranslatef(0.0, -4.0, -1.5);
93             glBegin(GL_TRIANGLE_STRIP);
94                 /* left wing */
95                 v3f(-7.0, 0.0, 2.0); v3f(-1.0, 0.0, 3.0);
96                 glColor3f(red = planes[i].red, green = planes[i].green,
97                     blue = planes[i].blue);
98                 v3f(-1.0, 7.0, 3.0);
99                 /* left side */
100                glColor3f(0.6 * red, 0.6 * green, 0.6 * blue);
101                v3f(0.0, 0.0, 0.0); v3f(0.0, 8.0, 0.0);
102                /* right side */
103                v3f(1.0, 0.0, 3.0); v3f(1.0, 7.0, 3.0);
104                /* final tip of right wing */
105                glColor3f(red, green, blue);
106                v3f(7.0, 0.0, 2.0);
107            glEnd();
108            glPopMatrix();
109        }
110     if (doubleBuffer) glXSwapBuffers(dpy, XtWindow(w));
111     if (!glXIsDirect(dpy, cx))
112         glFinish(); /* avoid indirect rendering latency from queuing */

```

```

113 #ifdef DEBUG
114     { /* for help debugging, report any OpenGL errors that occur per frame */
115         GLenum error;
116         while((error = glGetError()) != GL_NO_ERROR)
117             fprintf(stderr, "GL error: %s\n", gluErrorString(error));
118     }
119 #endif
120 }

121 void tick_per_plane(int i)
122 {
123     float theta = planes[i].theta += planes[i].speed;
124     planes[i].z = -9 + 4 * cos(theta);
125     planes[i].x = 4 * sin(2 * theta);
126     planes[i].y = sin(theta / 3.4) * 3;
127     planes[i].angle = ((atan(2.0) + M_PI_2) * sin(theta) - M_PI_2) * 180 / M_PI;
128     if (planes[i].speed < 0.0) planes[i].angle += 180;
129 }

130 void add_plane(void)
131 {
132     int i;

133     for (i = 0; i < MAX_PLANES; i++)
134         if (planes[i].speed == 0) {

135 #define SET_COLOR(r,g,b) \
136     planes[i].red=r; planes[i].green=g; planes[i].blue=b; break;

137         switch (random() % 6) {
138             case 0: SET_COLOR(1.0, 0.0, 0.0); /* red */
139             case 1: SET_COLOR(1.0, 1.0, 1.0); /* white */
140             case 2: SET_COLOR(0.0, 1.0, 0.0); /* green */
141             case 3: SET_COLOR(1.0, 0.0, 1.0); /* magenta */
142             case 4: SET_COLOR(1.0, 1.0, 0.0); /* yellow */
143             case 5: SET_COLOR(0.0, 1.0, 1.0); /* cyan */
144         }
145         planes[i].speed = (random() % 20) * 0.001 + 0.02;
146         if (random() & 0x1) planes[i].speed *= -1;
147         planes[i].theta = ((float) (random() % 257)) * 0.1111;
148         tick_per_plane(i);
149         if (!moving) draw(glxarea);
150         return;
151     }
152     XBell(dpy, 100); /* can't add any more planes */
153 }

154 void remove_plane(void)
155 {
156     int i;

157     for (i = MAX_PLANES - 1; i >= 0; i--)
158         if (planes[i].speed != 0) {
159             planes[i].speed = 0;
160             if (!moving) draw(glxarea);
161             return;
162         }
163     XBell(dpy, 100); /* no more planes to remove */
164 }

```



```

165 void resize(Widget w, XtPointer data, XtPointer callData)
166 {
167     Dimension      width, height;

168     if(made_current) {
169         XtVaGetValues(w, XmNwidth, &width, XmNheight, &height, NULL);
170         glViewport(0, 0, (GLint) width, (GLint) height);
171     }
172 }

173 void tick(void)
174 {
175     int i;

176     for (i = 0; i < MAX_PLANES; i++)
177         if (planes[i].speed != 0.0) tick_per_plane(i);
178 }

179 Boolean animate(XtPointer data)
180 {
181     tick();
182     draw(glxarea);
183     return False;          /* leave work proc active */
184 }

185 void toggle(void)
186 {
187     moving = !moving; /* toggle */
188     if (moving)
189         workId = XtAppAddWorkProc(app, animate, NULL);
190     else
191         XtRemoveWorkProc(workId);
192 }

193 void quit(Widget w, XtPointer data, XtPointer callData)
194 {
195     exit(0);
196 }

197 void input(Widget w, XtPointer data, XtPointer callData)
198 {
199     XmDrawingAreaCallbackStruct *cd = (XmDrawingAreaCallbackStruct *) callData;
200     char      buf[1];
201     KeySym    keysym;
202     int       rc;

203     if(cd->event->type == KeyPress)
204         if(XLookupString((XKeyEvent *) cd->event, buf, 1, &keysym, NULL) == 1)
205             switch (keysym) {
206                 case XK_space:
207                     if (!moving) { /* advance one frame if not in motion */
208                         tick();
209                         draw(w);
210                     }
211                     break;
212                 case XK_Escape:
213                     exit(0);
214             }
215 }

```

```

216 void map_state_changed(Widget w, XtPointer data, XEvent * event, Boolean * cont)
217 {
218     switch (event->type) {
219     case MapNotify:
220         if (moving && workId != 0) workId = XtAppAddWorkProc(app, animate, NULL);
221         break;
222     case UnmapNotify:
223         if (moving) XtRemoveWorkProc(workId);
224         break;
225     }
226 }

227 main(int argc, char *argv[])
228 {
229     toplevel = XtAppInitialize(&app, "Paperplane", NULL, 0, &argc, argv,
230                               fallbackResources, NULL, 0);
231     dpy = XtDisplay(toplevel);
232     /* find an OpenGL-capable RGB visual with depth buffer */
233     vi = glXChooseVisual(dpy, DefaultScreen(dpy), dblBuf);
234     if (vi == NULL) {
235         vi = glXChooseVisual(dpy, DefaultScreen(dpy), snglBuf);
236         if (vi == NULL)
237             XtAppError(app, "no RGB visual with depth buffer");
238         doubleBuffer = GL_FALSE;
239     }
240     /* create an OpenGL rendering context */
241     cx = glXCreateContext(dpy, vi, /* no display list sharing */ None,
242                          /* favor direct */ GL_TRUE);
243     if (cx == NULL)
244         XtAppError(app, "could not create rendering context");
245     /* create an X colormap since probably not using default visual */
246 #ifdef noGLwidget
247     cmap = XCreateColormap(dpy, RootWindow(dpy, vi->screen),
248                           vi->visual, AllocNone);
249     /*
250      * Establish the visual, depth, and colormap of the toplevel
251      * widget _before_ the widget is realized.
252      */
253     XtVaSetValues(toplevel, XtNvisual, vi->visual, XtNdepth, vi->depth,
254                  XtNcolormap, cmap, NULL);
255 #endif
256     XtAddEventHandler(toplevel, StructureNotifyMask, False,
257                      map_state_changed, NULL);
258     mainw = XmCreateMainWindow(toplevel, "mainw", NULL, 0);
259     XtManageChild(mainw);
260     /* create menu bar */
261     menubar = XmCreateMenuBar(mainw, "menubar", NULL, 0);
262     XtManageChild(menubar);
263 #ifdef noGLwidget
264     /* Hack around Xt's unfortunate default visual inheritance. */
265     XtSetArg(menuPaneArgs[0], XmNvisual, vi->visual);
266     menupane = XmCreatePulldownMenu(menubar, "menupane", menuPaneArgs, 1);
267 #else
268     menupane = XmCreatePulldownMenu(menubar, "menupane", NULL, 0);
269 #endif
270     btn = XmCreatePushButton(menupane, "Quit", NULL, 0);
271     XtAddCallback(btn, XmNactivateCallback, quit, NULL);
272     XtManageChild(btn);
273     XtSetArg(args[0], XmNsubMenuId, menupane);
274     cascade = XmCreateCascadeButton(menubar, "File", args, 1);

```

```

275     XtManageChild(cascade);
276 #ifndef noGLWidget
277     menupane = XmCreatePulldownMenu(menuubar, "menupane", menuPaneArgs, 1);
278 #else
279     menupane = XmCreatePulldownMenu(menuubar, "menupane", NULL, 0);
280 #endif
281     btn = XmCreateToggleButton(menupane, "Motion", NULL, 0);
282     XtAddCallback(btn, XmNvalueChangedCallback, (XtCallbackProc)toggle, NULL);
283     XtManageChild(btn);
284     btn = XmCreatePushButton(menupane, "Add plane", NULL, 0);
285     XtAddCallback(btn, XmNactivateCallback, (XtCallbackProc)add_plane, NULL);
286     XtManageChild(btn);
287     btn = XmCreatePushButton(menupane, "Remove plane", NULL, 0);
288     XtAddCallback(btn, XmNactivateCallback, (XtCallbackProc)remove_plane, NULL);
289     XtManageChild(btn);
290     XtSetArg(args[0], XmNsubMenuId, menupane);
291     cascade = XmCreateCascadeButton(menuubar, "Planes", args, 1);
292     XtManageChild(cascade);
293     /* create framed drawing area for OpenGL rendering */
294     frame = XmCreateFrame(mainw, "frame", NULL, 0);
295     XtManageChild(frame);
296 #ifndef noGLWidget
297     glxarea = XtVaCreateManagedWidget("glxarea", xmDrawingAreaWidgetClass,
298                                     frame, NULL);
299 #else
300 #ifdef noMotifGLWidget
301     /* notice glwDrawingAreaWidgetClass lacks an 'M' */
302     glxarea = XtVaCreateManagedWidget("glxarea", glwDrawingAreaWidgetClass,
303                                     frame, NULL);
304 #else
305     glxarea = XtVaCreateManagedWidget("glxarea", glwMDrawingAreaWidgetClass,
306                                     frame, GLwNvisualInfo, vi, NULL);
307 #endif
308     XtAddCallback(glxarea, XmNexposeCallback, (XtCallbackProc)draw, NULL);
309     XtAddCallback(glxarea, XmNresizeCallback, resize, NULL);
310     XtAddCallback(glxarea, XmNinputCallback, input, NULL);
311     /* set up application's window layout */
312     XmMainWindowSetAreas(mainw, menuubar, NULL, NULL, NULL, frame);
313     XtRealizeWidget(toplevel);
314     /*
315      * Once widget is realized (ie, associated with a created X window), we
316      * can bind the OpenGL rendering context to the window.
317      */
318     glXMakeCurrent(dpy, XtWindow(glxarea), cx);
319     made_current = GL_TRUE;
320     /* setup OpenGL state */
321     glDepthFunc(GL_LEQUAL);
322     glClearDepth(1.0);
323     glClearColor(0.0, 0.0, 0.0, 0.0);
324     glMatrixMode(GL_PROJECTION);
325     glFrustum(-1.0, 1.0, -1.0, 1.0, 1.0, 20);
326     glMatrixMode(GL_MODELVIEW);
327     /* add three initial random planes */
328     srandom(getpid());
329     add_plane(); add_plane(); add_plane();
330     /* start event processing */
331     XtAppMainLoop(app);
332 }

```

References

- [1] Tom Gaskins, "Using PEXlib with X Toolkits." *PEXlib Programming Manual*, O'Reilly & Associates, Inc., 1992.
- [2] Mark Kilgard, "OpenGL and X, Part 1: An Introduction," *The X Journal*, SIGS Publications, Nov/Dec 1993.
- [3] Mark Kilgard, "OpenGL and X. Part 2: Using OpenGL with Xlib," *The X Journal*, SIGS Publications, Jan/Feb 1994.
- [4] Silicon Graphics, *The OpenGL Porting Guide*, supplied with the IRIX 5.2 development option, 1994.
- [5] Josie Wernecke, *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor*, Addison-Wesley, 1994.

Engineer's Handbook General Stuff

May 5, 1994

Graphics Library Quick Reference Guide

probable data location: `dist.wpd:/sgi/doc/gl/glrefcard.ps`
access via: `handbook glrefcard`



Graphics Library™

Quick Reference Guide

© 1991 Silicon Graphics
Version 3.3.2

Please send any bugs, comments or suggestions to:
Silicon Graphics, Inc.
Technical Education, M/S 12-170
2011 North Shoreline Blvd.
Mountain View, CA 94039

Accumulation Buffer

```
void acbuf( long operation, float value );
    operation: AC_CLEAR, AC_ACCUMULATE,
    AC_CLEAR_ACCUMULATE, AC_RETURN, AC_MULT, AC_ADD
void acsize( long planes );
    planes: 0, 16
```

Alpha Blending

see also Color

```
void afunction( long ref, long function );
    ref: 0 - 255
    function: AF_NOTEQUAL, AF_ALWAYS
void blendfunction( long source_factor, long
    dest_factor );
    source_factor: BF_ZERO, BF_ONE, BF_DC, BF_MDC,
    BF_SA, BF_MSA, BF_DA, BF_MDA, BF_MIN_SA_MDA
    dest_factor: BF_ZERO, BF_ONE, BF_SC, BF_MSC, BF_SA,
    BF_MSA, BF_DA, BF_MDA
```

Animation

```
void backbuffer( long b );
    b: TRUE, FALSE
void doublebuffer( void );
long buffer getbuffer( void );
void mswapbuffers( long framebuffers );
    framebuffers: NORMALDRAW | OVERDRAW | UNDERDRAW
void swapbuffers( void );
void swapinterval( short interval );
```

Anti-Aliasing

see also Accumulation Buffer

```
void linesmooth( unsigned long mode );
    mode: SML_OFF, SML_ON, SML_SMOOTHER, SML_ENDCORRECT
void pntsmooth( unsigned long mode );
    mode: SMP_OFF, SMP_ON, SMP_SMOOTHER
void polysmooth( long mode );
    mode: PYSM_OFF, PYSM_ON, PYSM_SHRINK
```

Arcs

```
void arc( float xc, float yc, float rad, short start,
    short end );
```

```
void arcf( float xc, float yc, float rad, short start,
    short end );
void arcfi( long xc, long yc, long rad, short start,
    short end );
void arcfs( short xc, short yc, short rad, short start,
    short end );
void arci( long xc, long yc, long rad, short start,
    short end );
void arcs( short xc, short yc, short rad, short start,
    short end );
    start, end: tenths of degrees
```

Backface Removal

see also Z-Buffering

```
void backface( long b );
    b: TRUE, FALSE
void frontface( long b );
    b: TRUE, FALSE
long b getbackface( void );
```

Backgrounds

```
void imakebackground( void );
```

Bounding Boxes

```
void bbox2( short xmin, short ymin, float x1, float y1,
    float x2, float y2 );
void bbox2i( short xmin, short ymin, long x1, long y1,
    long x2, long y2 );
void bbox2s( short xmin, short ymin, short x1, short y1,
    short x2, short y2 );
void getscrbox( long *left, long *right, long *bottom,
    long *top );
```

Circles

```
void circ( float xc, float yc, float rad );
void circf( float xc, float yc, float rad );
void circfi( long xc, long yc, long rad );
void circfs( short xc, short yc, short rad );
void circi( long xc, long yc, long rad );
void circs( short xc, short yc, short rad );
```

Clearing

```
void clear( void );
void czclear( unsigned long color, long zval );
void sclear( void );
void zclear( void ); Clipping Planes
void clipplane( long index, long mode, float params[] );
    mode: CP_DEFINE, CP_ON, CP_OFF
```

Color

```
void c3f( float rgb[3] );
void c3i( long rgb[3] );
void c3s( short rgb[3] );
void c4f( float rgba[4] );
void c4i( long rgba[4] );
void c4s( short rgba[4] );
```

```
void color( Colorindex );
void colorf( float colorindex );
void cpack( unsigned long 0xAABBGGRR );
void gRGBcolor( short *red, short *green, short *blue );
void RGBcolor( short red, short green, short blue );
void RGBmode( void );
```

Color Map

```
void blink( short rate, Colorindex, short red, short
    green, short blue );
void cmode( void );
void color( unsigned short colorindex );
void colorf( float colorindex );
void cyclemap( short duration, short map, short nextmap );
long getcmode( void );
long colorindex getcolor( void );
long mapnum getmap( void );
void getmcolor( Colorindex, short *red, short *green,
    short *blue );
void mapcolor( Colorindex, short red, short green, short
    blue );
void multimap( void );
void onemap( void );
void setmap( short mapnum );
```

Cursors

See also Drawing Modes

```
void attachcursor( unsigned short xdevice, unsigned
    short ydevice );
void curorigin( short cursorindex, short xorigin, short
    yorigin );
void cursoff( void );
void curson( void );
void curstype( long type );
void defcursor( short cursorindex, unsigned short
    glyph[128] );
void getcursor( short *cursorindex, Colorindex
    *obsolete, Colorindex *obsolete, long *visible );
void setcursor( short cursorindex, Colorindex obsolete,
    Colorindex obsolete );
```

Curves and Surface

```
void bgncurve( void );
void endcurve( void );
void bgnsurface( void );
void endsurface( void );
void bgntrim( void );
void endtrim( void );
void nurbscurve( long knot_count, double knot_list[],
    long offset, double control_points[], long order,
    long type );
    type: N_V3D, N_V3D, N_ST, N_STW
void nurbsurface( long s_knot_count, double s_knot[],
    long t_knot_count, double t_knot[], long s_offset,
    long t_offset, double control_points[], long S-
    order, long t_order, long type );
    type: N_V3D, N_V3D, N_C4D, N_C4DR, N_T2D, N_T2DR
```

```

void pwlcurve( long points, double points[], long
  offset, long type );
  type: N_ST
void getnurbsproperty( long property, float *value );
void setnurbsproperty( long property, float value );
  property: N_ERRORCHECKING, N_PIXEL_TOLERANCE

```

Debugging

```
void foreground( void );
```

Depth Cueing

```

void depthcue( long b );
  b: TRUE, FALSE
long b getdcm( void );
void lRGBRange( short rmin, short gmin, short bmin,
  short rmax, short gmax, short bmax, long znear, long
  zfar );
void lsetdepth( long znear, long zfar );
void lshaderange( Colorindex low_intensity, Colorindex
  high_intensity, long znear, long zfar );

```

Devices

see also Dial and Button Box, Event Queueing

```

#include <gl/device.h>
long getbutton( unsigned short button );
void getdev( long num_devices, unsigned short devs[],
  short vals[] );
long getvaluator( unsigned short valuator );
void noise( unsigned short valuator, short delta );
void setvaluator( unsigned short valuator, short init,
  short minval, short maxval );
void tie( unsigned short button, unsigned short
  valuator1, unsigned short valuator2 );

```

Dial and Button Box

```

void dbtext( String text );
void getdev( long num_devices, unsigned short devs[],
  short vals[] );
void setdblights( unsigned long mask );

```

Display Lists

```

void bbox2( short xmin, short ymin, float x1, float y1,
  float x2, float y2 );
void bbox2i( short xmin, short ymin, long x1, long y1,
  long x2, long y2 );
void bbox2s( short xmin, short ymin, short x1, short y1,
  short x2, short y2 );
void getscrbox( long *left, long *right, long *bottom,
  long *top );
void callobj( Object );
void chunksize( long minsize );
void closeobj( void );
void compactify( Object );
void delobj( Object );
void deltag( Tag );
void editobj( Object );
Object genobj( void );

```

```

Tag gentag( void );
Object getopenobj( void );
long isobj( Object );
long istag( Tag );
void makeobj( Object );
void maketag( Tag );
void mapw( Object, short sx, short sy, float *wx1, float
  *wy1, float *wz1, float *wx2, float *wy2, float *wz2
  );
void mapw2( Object, short sx, short sy, float *wx, float
  *wy );
void newtag( Tag newtag, Tag oldtag, Offset );
void objdelete( Tag from_tag, Tag to_tag );
void objinsert( Tag );
void objreplace( Tag );

```

Distributed Graphics Library

```

void dglclose( long serverid );
long dglopen( String servername, long type );

```

Drawing Modes

```

void drawmode( long mode );
long mode getdrawmode( void );
  mode: NORMALDRAW, OVERDRAW, UNDERDRAW, PUPDRAW,
  CURSORDRAW
long enabled getbuffer( void );
  enabled: BCKBUFFER | FRNTBUFFER | DRAWZBUFFER
long mode getdisplaymode( void );
  mode: DMSINGLE, DMDOUBLE, DMRGB, DMRGBDOUBLE
void polymode( long mode );
void popattributes( void );
void pushattributes( void );
void shademodel( long model );
void subpixel( long b );
  b : TRUE, FALSE

```

Event Queueing

```

long numwords blkqread( short data[], short
  num_elements );
long b isqueued( unsigned short device );
void qdevice( unsigned short device );
void qenter( unsigned short device, short val );
long qqetfd( void );
long device qread( short *val );
void qreset( void );
long device qtest( void );
void unqdevice( unsigned short device );

```

Feedback

```

long elements endfeedback( short/float buffer[] );
void feedback( short/float buffer[], long size );
void passthrough( short token );

```

Fog

```
void fogvertex( long mode, float params[] );
```

```

  mode: FG_DEFINE, FG_ON, FG_OFF
void scrsubdivide( long mode, float params[] );
  mode: SS_OFF, SS_DEPTH

```

Gamma Correction

```
void gammaramp( short red[256], short green[256],
  short blue[256] );
```

Geometry Pipeline

see also Hardware Configuration

```

void finish( void );
void gexit( void );
void gflush( void );

```

Hardware Configuration

```

void cmode( void );
void doublebuffer( void );
void gconfig( void );
long value getgdesc( long inquiry );
  inquiry: GD_XMMAX, GD_YMMAX, GD_XPMAX, GD_YPMAX,
  GD_ZMAX, GD_ZMIN, GD_BITS_ACBUF, GD_BITS_ACBUF_HW,
  GD_BITS_CURSOR, GD_BITS_NORM_DBL_ALPHA,
  GD_BITS_NORM_DBL_CMODE, GD_BITS_NORM_DBL_MMAP,
  GD_BITS_NORM_DBL_RED, GD_BITS_NORM_DBL_GREEN,
  GD_BITS_NORM_DBL_BLUE, GD_BITS_NORM_SNG_ALPHA,
  GD_BITS_NORM_SNG_CMODE, GD_BITS_NORM_SNG_MMAP,
  GD_BITS_NORM_SNG_RED, GD_BITS_NORM_SNG_GREEN,
  GD_BITS_NORM_SNG_BLUE, GD_BITS_NORM_ZBUFFER,
  GD_BITS_OVER_SNG_CMODE, GD_BITS_UNDR_SNG_CMODE,
  GD_BITS_PUP_SNG_CMODE, GD_BITS_STENCIL,
  GD_AFUNCTION, GD_ALPHA_OVERUNDER, GD_BLEND,
  GD_CIFRACT, GD_CLIPPLANES, GD_CROSSHAIR_CINDEX,
  GD_DDBOX, GD_DITHER, GD_FOGVERTEX,
  GD_FRAMEGRABBER, GD_LIGHTING_TWOSIDE,
  GD_LINESMOOTH_CMODE, GD_LINESMOOTH_RGB,
  GD_LOGICOP, GD_NBLINKS, GD_NMMAPS, GD_NSCRNS,
  GD_NURBS_ORDER, GD_NVERTEX_POLY,
  GD_OVERUNDER_SHARED, GD_PATSIZE_64,
  GD_PNTSMOOTH_CMODE, GD_PNTSMOOTH_RGB, GD_POLYMODE,
  GD_PUP_TO_OVERUNDER, GD_READSOURCE,
  GD_READSOURCE_ZBUFFER, GD_SCRBOX, GD_SCRNTYPE,
  GD_STEREO, GD_SUBPIXEL_LINE, GD_SUBPIXEL_PNT,
  GD_SUBPIXEL_POLY, GD_TEXTPORT, GD_TEXTURE,
  GD_TIMERHZ, GD_TRIMCURVE_ORDER, GD_WSYS,
  GD_ZDRAW_GEOM, GD_ZDRAW_PIXELS
long bitplanes getplanes( void );
long enabled getzbuffer( void );
long gversion( char v[12] );
  v: GL4D-m.n, GL4DGT-m.n, GL4DGTX-m.n, GL4DPI2-m.n,
  GL4DPIT-m.n, GL4DPI-m.n
void glcompat( long mode, long value );
  mode: GLC_OLDPOLYGON, GLC_ZRANGEMAP
void lsetdepth( long near, long far );
void overlay( long planes );
  planes: 0, 2, 4, 8
void RGBmode( void );
void setmonitor( short type );
long type getmonitor( void );
  type: HZ30, HZ30_SG, HZ60, NTSC, PAL, STR_RECT
void singlebuffer( void );

```



```
void stensize( long planes );
    planes: 0 - 8
void underlay( long planes );
void zbuffer( long ); Icons
void iconsize( long x, long y );
void icontitle( String );
```

Keyboard

```
void clkoff( void );
void clkon( void );
void lampoff( unsigned char lamps );
void lampon( unsigned char lamps );
void ringbell( void );
void setbell( unsigned char duration );
```

Lighting

see also Normals

```
    mode: SS_OFF, SS_DEPTH
void lmbind( short target, short index );
    target: MATERIAL, BACKMATERIAL, LIGHT0 - LIGHT7,
    LMODEL
void lmcOLOR( long mode );
    mode: LMC_COLOR, LMC_EMISSION, LMC_AMBIENT,
    LMC_DIFFUSE, LMC_SPECULAR, LMC_AD, LMC_NULL
void lndef( short deftype, short index, short numprops,
    float props[] );
    deftype: DEFMATERIAL
    properties: ALPHA, AMBIENT, COLORINDEXES, DIFFUSE,
    EMISSION, SHININESS, SPECULAR, LMNULL
    deftype: DEFLIGHT
    properties: AMBIENT, LCOLOR, POSITION,
    SPOTDIRECTION, SPOTLIGHT, LMNULL
    deftype: DEFMODEL
    properties: AMBIENT, ATTENUATION, ATTENUATION2,
    LOCALVIEWER, TWOSIDE, LMNULL
void mmode( short mode );
    mode: MSINGLE, MVIEWING, MPROJECTION, MTEXTURE
```

Lines

see also Linestyle, Vertices

```
void bgnclosedline( void );
void bgnline( void );
void endclosedline( void );
void endline( void ); Linestyles
void deflinestyle( short index, unsigned short pattern
    );
    long index getlstyle( void );
void linesmooth( unsigned long mode );
    mode: SML_OFF, SML_ON, SML_SMOOTHER, SML_ENDCORRECT
void linewidth( short pixels );
    long width getlwidth( void );
void lsrepeat( long factor );
    long factor getlsrepeat( void );
void setlinestyle( short index );
```

Matrix Operations

See also Modeling, Viewing and Projection Transformations

```
void getmatrix( float[4][4] );
void loadmatrix( float[4][4] );
void mmode( short mode );
    long mode getmmode( void );
    mode: MSINGLE, MVIEWING, MPROJECTION, MTEXTURE
void multmatrix( float[4][4] );
void popmatrix( void );
void pushmatrix( void );
```

MOUSE

see DEVICES Modeling Transformations

```
void rot( float angle, char axis);
    angle: degrees
    axis: 'x','X','y','Y','z','Z'
void rotate( short angle, char axis);
    angle: tenths of degrees
    axis: 'x','X','y','Y','z','Z'
void scale( float sx, float sy, float sz );
void translate( float tx, float ty, float tz );
```

Normals

```
void n3f( float normal[3] );
void nmode( long mode );
    mode: NAUTO, NNORMALIZE
```

Nurbs

see Curves and Surface

Patterns

```
void defpattern( short index, short size, unsigned short
    pattern[] );
    long index getpattern( void );
void setpattern( short index );
```

Picking

```
long hits endpick( short buffer[] );
long hits endselect( short buffer[] );
void gselect( short buffer[], long size );
void initnames( void );
void loadname( short name );
void pick( short buffer[], long size );
void picksize( short xpixels, short ypixels);
void popname( void );
void pushname( short name );
```

Pixel Operations

see Raster Operations

Points

see also Vertices

```
void bgnpoint( void );
void endpoint( void );
```

Polygons

see also Vertices

```
void bgnpolygon( void );
void endpolygon( void );
void bgnqstrip( void );
void endqstrip( void );
void bgntmesh( void );
void endtmesh( void );
void concave( long b );
    b: TRUE, FALSE
```

Pop Up Menus

```
void addtopup( long pup, char *menustring, ... );
long pup defpup( char *menustring, long args, ... );
long val dopup( long pup );
void freepup( long pup );
long pup newpup( void );
void setup( long pup, long entry, unsigned long mode );
    mode: PUP_NONE, PUP_GREY
```

Projection Transformations

```
void fullscrn( void );
void endfullscrn( void );
void ortho( float left, float right, float bottom, float
    top, float near, float far );
void ortho2( float left, float right, float bottom,
    float top );
void perspective( short fovy, float aspect, float near,
    float far );
void screenspace( void );
void window( float left, float right, float bottom,
    float top, float near, float far );
```

Quad Strips

see Polygons

Stencil Planes

```
void sclear( unsigned long clearval);
void stencil( long enable, unsigned long ref, long
    fuction, unsigned long mask, long fail, long pass,
    long zpass );
    enable: TRUE, FALSE
    function: SF_NEVER, SF_LESS, SF_EQUAL, SF_LEQUAL,
    SF_GREATER, SF_NOTEQUAL, SF_GEQUAL, SF_ALWAYS
    fail, pass, zpass: ST_KEEP, ST_ZERO, ST_REPLACE,
    ST_INCR, ST_DECR, ST_INVERT
void stensize( long planes );
    planes: 0 - 8
void writemask( unsigned long mask );
```

Triangle Meshes

see Polygons

Raster Operations

See also Drawing Modes

```
void logicop( long opcode );
    opcode: LO_ZERO, LO_AND, LO_ANDR, LO_SRC, LO_ANDI,
    LO_DST, LO_XOR, LO_OR, LO_NOR, LO_XNOR, LO_NDST,
    LO_ORR, LO_NSRC, LO_ORI, LO_NAND, LO_ONE
```

```

long numpixels lrectread( short x1, short y1, short x2,
    short y2, unsigned long pixels[] );
long lrectwrite( short x1, short y1, short x2, short y2,
    unsigned long pixels[] );
void pixmode( long mode, long value );
    mode: PM_SHIFT, PM_EXPAND, PM_C0, PM_C1, PM_ADD24,
    PM_TTOB, PM_RTOL, PM_SIZE, PM_OFFSET, PM_STRIDE,
    PM_ZDATA
void readsourc( long buffer );
    buffer: SRC_AUTO, SRC_FRONT, SRC_BACK, SRC_ZBUFFER,
    SRC_FRAMEGRABBER, SRC_OVER, SRC_UNDER, SRC_PUP
void rectcopy( short x1, short y1, short x2, short y2,
    short newx, short newy );
long numpixels lrectread( short x1, short y1, short x2,
    short y2, unsigned short index[] );
long lrectwrite( short x1, short y1, short x2, short y2,
    unsigned short index[] );
void rectzoom( float xfactor, float yfactor );

```

Rectangles

```

void rect( float x1, float y1, float x2, float y2 );
void rectf( float x1, float y1, float x2, float y2 );
void rectfi( long x1, long y1, long x2, long y2 );
void rectfs( short x1, short y1, short x2, short y2 );
void recti( long x1, long y1, long x2, long y2 );
void rects( short x1, short y1, short x2, short y2 );

```

Screen Masks

see also Writemasks

```

void getscrmask( short *left, short *right, short
    *bottom, short *top );
void scrbox( long mode );
    mode: SB_RESET, SB_TRACK, SB_HOLD
void scrmask( short left, short right, short bottom,
    short top );

```

Text

```

void charstr( char *text );
void cmov( float x, float y, float z );
void cmovi( long x, long y, long z );
void cmovs( short x, short y, short z );
void cmov2( float x, float y );
void cmov2i( long x, long y );
void cmov2s( short x, short y );
void defrafterfont( short index, short maxheight, short
    numchars, Fontchar chars[], short numrastersints,
    unsigned short raster[] );
void font( short index );
void getcpos( short *x, short *y );
long pixels getdescender( void );
long index getfont( void );
long pixels getheight( void );
long pixels strwidth( String );

```

Texture Mapping

```

void scrsubdivide( long mode, float params[] );
    mode: SS_OFF, SS_DEPTH

```

```

void t2d( double t[2] );
void t2f( float t[2] );
void t2i( long t[2] );
void t2s( short t[2] );
void tevbind( long target, long index );
    target: TV_ENV0
void tevdef( long index, long numprops, float props[] );
void texbind( long target, long index );
void texdef2d( long index, long numcomponents, long
    width, long height, long image[], long numprops,
    float props[] );
    props: TX_MINFILTER, TX_MAGFILTER, TX_POINT,
    TX_BILINEAR, TX_MIPMAP_POINT, TX_MIPMAP_LINEAR,
    TX_MIPMAP_BILINEAR, TX_WRAP, TX_WRAP_S, TX_WRAP_T,
    TX_REPEAT, TX_CLAMP, TX_TILE, TX_NULL
void texgen( long coord, long mode, float params[] );
    coord: TX_S, TX_T
    mode: TG_CONTOUR, TG_LINEAR, TG_ON, TG_OFF

```

Vertices

```

void v2d( double v[2] );
void v2f( float v[2] );
void v2i( long v[2] );
void v2s( short v[2] );
void v3d( double v[3] );
void v3f( float v[3] );
void v3i( long v[3] );
void v3s( short v[3] );
void v4d( double v[4] );
void v4f( float v[4] );
void v4i( long v[4] );
void v4s( short v[4] );

```

Video

see also Hardware Configuration

```

void blankscreen( long b );
    b: TRUE, FALSE
void blanktime( long numframes );
void gsync( void );
void setvideo( long register, long value );
long value getvideo( long register );
void videocmd( long command );
    command: VP_INITNTSC_COMP, VP_INITNTSC_RGB,
    VP_INITPAL_COMP, VP_INITPAL_RGB

```

Viewing Transformations

```

void lookat( float vx, float vy, float vz, float px,
    float py, float pz, short twist );
void polarview( float dist, short azim, short inc, short
    twist );

```

Viewports

```

void getviewport( short *left, short *right, short
    *bottom, short *top );
void popviewport( void );
void pushviewport( void );

```

```

void viewport( short left, short right, short bottom,
    short top );

```

Window Management

```

void fudge( long xfudge, long yfudge );
void getorigin( long *orgx, long *orgy );
void getsize( long *sizeX, long *sizeY );
long screennum getwscrn( void );
void keepaspect( long x, long y );
void maxsize( long sizeX, long sizeY );
void minsize( long sizeX, long sizeY );
void noborder( void );
void noport( void );
void preposition( long x1, long x2, long y1, long y2 );
void prefsize( long sizeX, long sizeY );
void reshapeviewport( void );
long prevscreennum scrnattach( long screennum );
long prevscreennum scrnselect( long screennum );
void stepunit( long stepx, long stepy );
long gid swinopen( long parentgid );
void winclose( long gid );
void winconstraints( void );
long depth windepth( long gid );
long gid winget( void );
void winmove( long orgx, long orgy );
long gid winopen( String title );
void winpop( void );
void winposition( long x1, long x2, long y1, long y2 );
void winpush( void );
void winset( long gid );
void wintitle( String title );

```

Writemasks

```

void gRGBmask( short *redmask, short *greenmask, short
    *bluemask );
void RGBwritemask( short red, short green, short blue );
long mask getwritemask( void );
void wmpack( unsigned long 0xAABBGGRR );
void writemask( unsigned short indexmask );

```

Z-Buffering

see also Backface Removal

```

void zclear( unsigned long color, long zval );
long getzbuffer( void );
void lsetdepth( long zmin, long zmax );
void zbuffer( long b );
    b: TRUE, FALSE
void zclear( void );
void zdraw( long );
void zfunction( long function );
void zsource( long source );
void zwritemask( unsigned long mask );

```



