SGI® OpenGL Vizserver™ Administrator's Guide

Version 3.4

007-4481-007

CONTRIBUTORS
Written by Jenn McGee and Ken Jones
Illustrated by Chrystie Danzer
Production by Karen Jacobson
Engineering contributions by Younghee Lee and Yochai Shefi-Simchon

New Features in This Guide

In addition to miscellaneous corrections, this revision of the guide contains the following changes:

- Changes to the following sections to support the ability to select the active screens, screens OpenGL Vizserver will serve to the clients:
  - — "Allocating Graphics Pipes for OpenGL Vizserver" in Chapter 2
  - — "Graphics Pipe Allocation Guidelines" in Chapter 2
  - — "Hardware Readback" in Chapter 2
- A new section "Optimizing for High-Latency Networks" in Chapter 3

Record of Revision

Version	Description
001	September 2002 Original publication; supports OpenGL Vizserver 3.0.
002	November 2002 Supports OpenGL Vizserver 3.0.1.
003	March 2003 Supports OpenGL Vizserver 3.1.
004	August 2003 Supports OpenGL Vizserver 3.2.
005	October 2003 Supports OpenGL Vizserver 3.2.1.
006	January 2004 Supports OpenGL Vizserver 3.3.
007	July 2004 Supports OpenGL Vizserver 3.4.

Contents

Figures

Tables

About This Guide

This document is intended for system administrators and gives information about installing, configuring, tuning, and troubleshooting OpenGL Vizserver.

System Requirements

OpenGL Vizserver consists of client and server modules.

The OpenGL Vizserver server module requires one of the following system types:

- SGI Onyx 3000 series with InfiniteReality3 or InfiniteReality4 graphics
- SGI Onyx 3000 series with InfinitePerformance graphics
- SGI Onyx 300 systems with InfinitePerformance graphics
- SGI Onyx 300 systems with InfiniteReality3 or InfiniteReality4 graphics
- SGI Onyx 350 systems with InfinitePerformance graphics
- SGI Onyx 350 systems with InfiniteReality3 or InfiniteReality4 graphics
- Silicon Graphics Onyx2 systems with InfiniteReality2, InfiniteReality3, or InfiniteReality4 graphics
- Silicon Graphics Octane or Octane2 systems
- Silicon Graphics Onyx4 UltimateVision systems
- Silicon Graphics Fuel systems
- Silicon Graphics Tezro systems

A server module must have the following software installed:

- IRIX 6.5.11 or later

OpenGL Vizserver supports clients running the following software platforms:

- IRIX 6.5.11 or later
- Solaris 2.5.1 or later
- Red Hat Linux 6.2 or later with XFree86 v4
- Windows NT 4.0 with service pack 6a or later
- Windows 2000 with service pack 2 or later
- Windows XP

Related Publications

The following documents contain additional information that may be helpful:

- *SGI OpenGL Vizserver User's Guide*
- *Performance Co-Pilot User's and Administrator's Guide*
- *IRIX Admin: Networking and Mail*
- *IRIX Admin: Software Iinstallation and Licensing*

Obtaining Publications

You can obtain SGI documentation in the following ways:

- See the SGI Technical Publications Library at http://docs.sgi.com. Various formats are available. This library contains the most recent and most comprehensive set of online books, release notes, man pages, and other information.
- If it is installed on your SGI system, you can use InfoSearch, an online tool that provides a more limited set of online books, release notes, and man pages. With an IRIX system, select **Help** from the Toolchest, and then select **InfoSearch**. Or you can type `infosearch` on a command line.
- You can also view release notes by typing either `grelnotes` or `relnotes` on a command line.
- You can also view man pages by typing `man` <*title*> on a command line.

Conventions

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
function	This bold font indicates a function or method name. Parentheses are also appended to the name.
variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.
[]	Brackets enclose optional portions of a command or directive line.
...	Ellipses indicate that a preceding element can be repeated.
`manpage`(*x*)	Man page section identifiers appear in parentheses after man page names.
GUI element	This bold font denotes the names of graphical user interface (GUI) elements, such as windows, screens, dialog boxes, menus, toolbars, icons, buttons, boxes, and fields.

Reader Comments

If you have comments about the technical accuracy, content, or organization of this document, contact SGI. Be sure to include the title and document number of the manual with your comments. (Online, the document number is located in the front matter of the manual. In printed manuals, the document number is located at the bottom of each page.)

You can contact SGI in any of the following ways:

- Send e-mail to the following address:

  techpubs@sgi.com

- Use the Feedback option on the Technical Publications Library webpage:

  `http://docs.sgi.com`

- Contact your customer service representative and ask that an incident be filed in the SGI incident tracking system.
- Send mail to the following address:

  Technical Publications
  SGI
  1500 Crittenden Lane, M/S 535
  Mountain View, CA 94043-1351

SGI values your comments and will respond to them promptly.

Installation

This chapter explains how to install the OpenGL Vizserver server module and client modules. The following topics are covered:

It is assumed that your operating system (OS) is already installed and configured before installing OpenGL Vizserver. For OS installation and configuration information, refer to your system installation and administration guide.

Once your operating system is properly configured, use the following sections for installing OpenGL Vizserver over a network or from a CD.

You can also go to http://www.sgi.com/software/vizserver and click the **Download** link.

Installing the Server

The OpenGL Vizserver 3.*x* server module consists of the file subsystems shown in Table 1-1.

Table 1-1 Server File Subsystems

Subsystem	Description
`vizserver_server.sw.vizserver`	The OpenGL Vizserver server's main software. It contains session manager and server manager executables, as well as libraries and scripts needed for server systems.
`vizserver_server.modules.auth`	Authentication module for the server.

Table 1-1 Server File Subsystems **(continued)**

Subsystem	Description
`vizserver_server.modules.pam_auth`	Pluggable Authentication Modules (PAM) for the server. It requires the installation of IRIX 6.5.22 or later, including `eoe.sw.pam`.
`vizserver_server.modules.comp`	Compression module for the server.
`vizserver_server.modules.perf`	Performance Co-Pilot module for the server. It requires the installation of `pcp_eoe.sw.eoe` and `pcp_eoe.sw.monitor`.
`vizserver_server.modules.readback`	Hardware readback module for the server. It requires the installation of `ml_sgc.sw.eoe` and can be installed only on Onyx4 systems.
`vizserver_server.modules.resclient`	Reservation client module for the server.
`vizserver_server.collab.client`	Collaborative session support. It requires the installation of `vizserver_client.sw.client`.
`vizserver_server.books.Vizserver_AG`	OpenGL Vizserver Administrator's Guide.
`vizserver_server.man.relnotes`	Release notes.
`vizserver_server.man.vizserver`	Man pages.
`vizserver_server.web.res`	Web module for OpenGL Vizserver pipe reservation. It requires either Netscape or SGI Apache web server to operate.
`vizserver_server.web.doc`	OpenGL Vizserver online documentation. It requires either Netscape or SGI Apache web server to operate.

Note: Please note that the OpenGL Vizserver 3.*x* server software is not compatible with a client running OpenGL Vizserver 2.0.1 or earlier and vice versa.

The following steps describe how you install OpenGL Vizserver.

1. Before installing the OpenGL Vizserver server, check if you have any previous versions of it in your system by entering one of the following commands:

   # **versions vizserver***

   OR

   # **showprods vizserver***

   This will show all of the OpenGL Vizserver server software as well as the OpenGL Vizserver clients, if any, on your system.

2. Log in as `root` and use `inst` or `swmgr` to install the OpenGL Vizserver server software. For more details on installation software on IRIX, see the *IRIX Admin: Software Installation and Licensing* manual.

3. If you are installing from a CD, enter the following command:

   # **inst -f /CDROM/dist**

   OR

   If you are installing over the network, enter a command similar to the following:

   # **inst -f** *machine*:*distribution_directory*

4. Resolve conflicts, if any, and continue the installation process. When done, verify your installation by entering the following command:

   # **versions vizserver***

While installing the OpenGL Vizserver server software, you may also want to install an OpenGL Vizserver IRIX client because the installation process is the same and the installable images are packaged in the same directory in the OpenGL Vizserver CD. Also, if you want to run a collaborative session, you need to install `vizserver_server.collab.client`, which requires the installation of the `vizserver_client.sw.client` module.

Installing the Client

OpenGL Vizserver supports clients running IRIX, Solaris, Linux, and Windows operating systems.

Note: Please note that the OpenGL Vizserver 3.*x* clients are not compatible with a server running OpenGL Vizserver 2.0.1 or earlier.

IRIX

The OpenGL Vizserver IRIX client module consists of the file subsystems shown in Table 1-2.

Table 1-2 IRIX Client File Subsystems

Subsystem	Description
`vizserver_client.sw.client`	The OpenGL Vizserver client GUI program. It also contains authentication and compression libraries.
`vizserver_client.books.Vizserver_UG`	The OpenGL Vizserver User's Guide.
`vizserver_client.man.relnotes`	Release notes.
`vizserver_client.man.vizserver`	Man pages.
`vizserver_dev.sw.base`	The OpenGL Vizserver compression, authentication, and reservation APIs.
`vizserver_dev.sw.examples`	Code examples of compression, authentication, and reservation APIs. Located in the `/usr/share/vizserver/src` directory.
`vizserver_dev.man.vizserver`	Man pages of APIs. HTML pages are located in the `/usr/share/vizserver/doc/developer` directory.

Note: If you installed the OpenGL Vizserver IRIX client while installing OpenGL Vizserver server software at the same time, skip the rest of this section.

1. Before installing the OpenGL Vizserver IRIX client, check if you have any previous versions of it in your system by entering one of the following commands:

   ```
   # versions vizserver_client
   ```

   OR

   ```
   # showprods vizserver_client
   ```

2. Log in as `root` and use `inst` or `swmgr` to install the OpenGL Vizserver client software. For more details on installation software on IRIX, see the *IRIX Admin: Software Installation and Licensing* manual.

3. If you are installing from a CD, enter the following command:

   ```
   # inst -f /CDROM/dist
   ```

   OR

   If you are installing over the network, enter a command similar to the following:

   # **inst -f** *machine*:*distribution_directory*

4. Resolve conflicts, if any, and continue the installation process. Verify your installation by entering the following command:

   ```
   # versions vizserver\*
   ```

Solaris

The OpenGL Vizserver Solaris client module consists of the subsystems shown in Table 1-3.

Table 1-3 Solaris Client Modules

Subsystem	Description
`SGIvizsvr-solaris`	The OpenGL Vizserver client GUI program. It also contains authentication and compression libraries, man pages, release notes, a user's guide, etc.
`SGIvizdev-solaris`	The OpenGL Vizserver compression and authentication development toolkit. (Optional)

1. To check if there is a previous version of the OpenGL Vizserver client on your system, enter the following command:

   ```
   # pkginfo | grep SGIviz
   ```

2. For a clean installation, you may want to remove any previous installation. Enter the following command:

   ```
   # pkgrm SGIvizsvr
   ```

3. Log in as `root` and install the client modules using the following commands. The product is installed in the `/opt/SGIvizsvr` directory by default.

   ```
   # pkgadd -d /CDROM/solaris/SGIvizsvr-solaris
   ```

4. To verify the installation, enter the following command:

   ```
   # pkginfo -i SGIvizsvr
   ```

For the installation of the optional subsystem `SGIvizdev-solaris` follow similar procedures as those described above. For more details about installation and removal of software in a Solaris system, see the `pkginfo`(1), `pkgrm`(1m), and `pkgadd`(1m) man pages.

Linux

The OpenGL Vizserver Linux client module consists of the subsystems shown in Table 1-4.

Table 1-4 Linux Client Modules

Subsystem	Description
`SGIvizsvr-linux.i386.rpm`	The OpenGL Vizserver client GUI program. It also contains authentication and compression libraries, man pages, release notes, a user's guide, etc.
`SGIvizdev-linux.i386.rpm`	The OpenGL Vizserver compression and authentication development toolkit. (Optional)

The following steps describe how you install OpenGL Vizserver.

1. Check if there is a previous installation of the OpenGL Vizserver client software in your system by entering the following command:

   ```
   # rpm -qa | grep SGIviz
   ```

2. Log in as `root` and enter the following command:

   ```
   # rpm -Uvh /CDROM/linux/SGIvizsvr-linux.i386.rpm
   ```

3. To verify the installation, enter the following command:

   ```
   # rpm -qi SGIvizsvr
   ```

For the installation of the optional subsystem `SGIvizdev-linux.i386.rpm` follow similar procedures to those described above. For more details about installation and removal of software in a Linux system, see the `rpm`(8) man page.

Windows

The OpenGL Vizserver Windows client module is packaged in a self-extracting executable, `SGIvizsvr-win32.exe`. It contains the following optional components:

- Documentation: *SGI OpenGL Vizserver User's Guide*
- SDK: The OpenGL Vizserver compression and authentication development options
- SDK Documentation: The OpenGL Vizserver compression and authentication modules man pages
- SDK Example Code: The OpenGL Vizserver compression and authentication modules example codes

Double clicking on the `SGIvizsvr-win32.exe` file leads you to the installation procedure. Just follow the instructions on your screen.

Configuration

This chapter explains how to configure OpenGL Vizserver. The following topics are covered:

Configuring the Server

In most cases, no additional configuration needs to be done to the default configuration of the OpenGL Vizserver server (`vsserver`(1m)) before using it. However, in some cases, you might want to configure the server for your environment's specific needs. This section describes how to configure the server using `vsconfig`(1m).

Setting up the network between the OpenGL Vizserver client and server will not be discussed here. For that information, see the *IRIX Admin: Networking and Mail* manual.

Configuring the OpenGL Vizserver server can be done manually or by using the graphical configuration tool called `vsconfig`. In the following sample, we will use `vsconfig`. The `vsconfig` tool provides an easy-to-use GUI for OpenGL Vizserver server configuration.

To launch the `vsconfig` tool, enter the following command as a `root` user:

```
# vsconfig
```

The `vsconfig` command shows in the first page the current status of the `vsserver` in your system. If there is no `vsserver` running on your system, this configuration GUI will appear as shown in Figure 2-1.

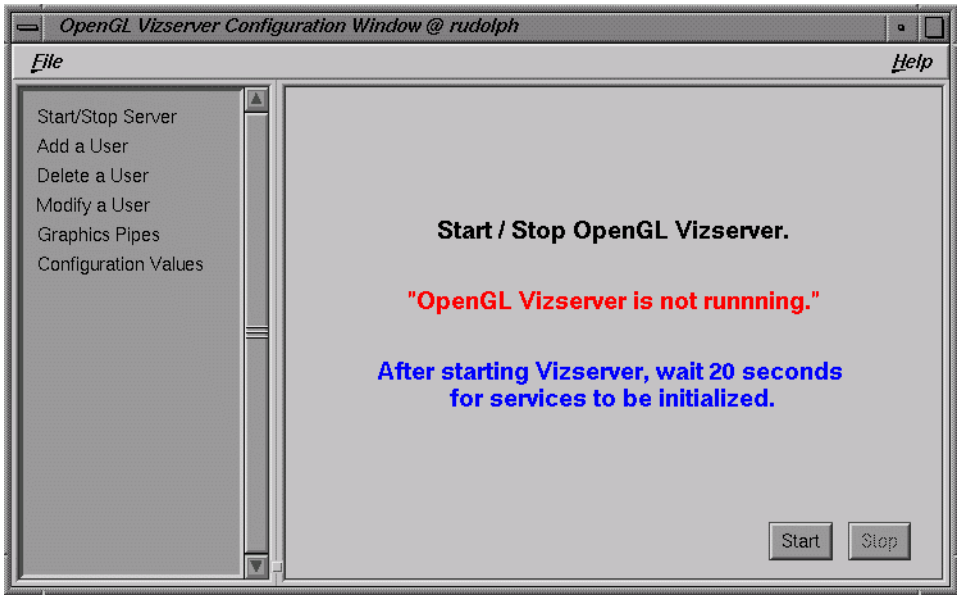

Figure 2-1 Configuration GUI

If you already started `vsserver` and have it running on the system, the result would be as shown in Figure 2-2 on page 11.

Starting and Stopping the Server Using the GUI

You can start and stop the OpenGL Vizserver server by pressing the **Start** or **Stop** button on the **Start / Stop Server** panel.

Pressing the **Start** button from the status shown in Figure 2-1 will give you a window as shown in Figure 2-2.

Once the OpenGL Vizserver server manager is running, you can start to use OpenGL Vizserver right away. For the instructions about how to use OpenGL Vizserver, see the *OpenGL Vizserver User's Guide*.

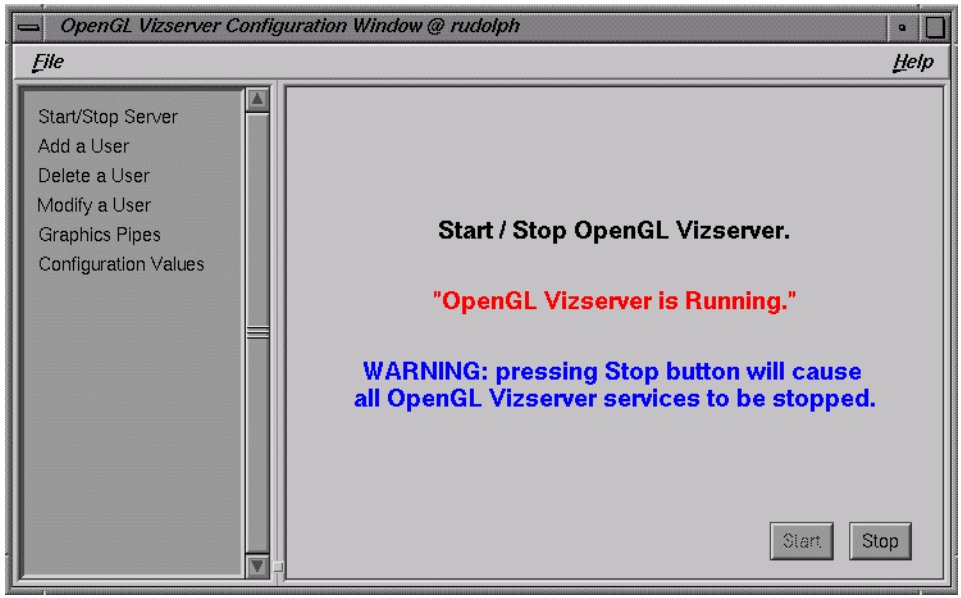

Figure 2-2 Starting the OpenGL Vizserver Server Manager

Starting and Stopping the Server Using the Command Line Interface

You can also start and stop the OpenGL Vizserver server using the command line interface.

To check whether the OpenGL Vizserver server manager is running, enter the following command:

```
$ ps -ef | grep vsserver
```

To stop the OpenGL Vizserver server manager (`vsserver`), enter the following command:

```
# /etc/init.d/vizserver stop
```

To start `vsserver`, enter the following commands:

```
# chkconfig vizserver on
# /etc/init.d/vizserver start
```

Allocating Graphics Pipes for OpenGL Vizserver

When a user starts an OpenGL Vizserver session, one or more graphics pipes need to be allocated for the session by the OpenGL Vizserver server manager. All graphics pipes allocated for a session are used by the session's X server, although it might be the case that only a subset of these pipes are actually being served by OpenGL Vizserver. The status of the **Active Screens** check boxes in the **Session Start** window of the client determines which pipes/screens are served. See "Graphics Pipe Allocation Guidelines" on page 30 for more details.

One case in which a graphics pipe does not need to be allocated for a session is when a user with a local X server on the server machine starts a collaborative session. In this case, OpenGL Vizserver uses the graphics pipes that are managed by a local X server.

At startup, `vsconfig` extracts the number of graphics pipes resident in the system, the graphics type, the X server name, and the display size associated with each pipe. This information is shown in the top half of the **Graphics Pipes** panel, as shown in Figure 2-3.

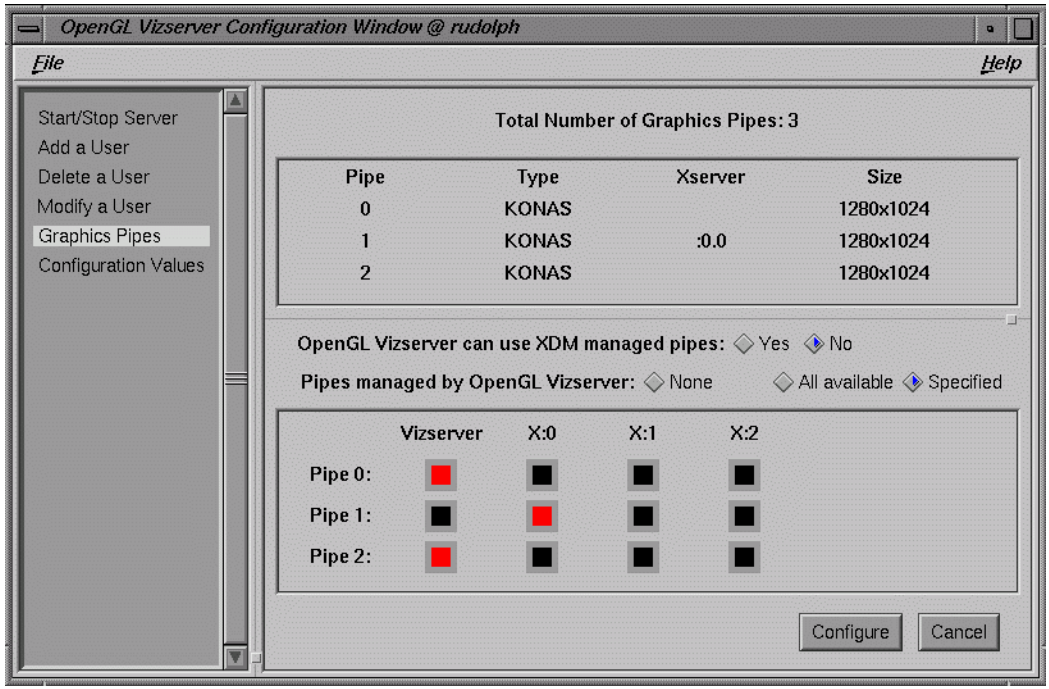

Figure 2-3 Graphics Pipes Panel

A graphics pipe in a system can be in one of the following three states:

- Managed by X display manager (XDM). This means that an X server defined in the `DisplayManager.servers` file of XDM's configuration file (usually `/var/X11/xdm/xdm-config`) is using the graphics pipe.
- Managed by OpenGL Vizserver. This means that the graphics pipe is not managed by XDM and the value of the **Pipes managed by OpenGL Vizserver** radio buttons is either **All available** or **Specified**.
- Managed by nobody. This means that neither XDM nor OpenGL Vizserver manages the graphics pipe.

The system in Figure 2-3 has three graphics pipes. Pipe 1 is used by X server `(:0)`, that is, managed by XDM. And pipes 0 and 2 are not managed by `XDM`. So, OpenGL Vizserver can use pipe 0 and pipe 2 for itself.

The bottom half of the panel shows the current configuration of XDM and OpenGL Vizserver in the two-dimensional array format. Each row represents a graphics pipe and each column represents OpenGL Vizserver or one of the X servers. If there are *k* graphics pipes in your system, the system can have at most *k* X servers, assigning one pipe to one X server. Of course, you can put all the pipes in one X server. X server numbers usually start from 0.

The **OpenGL Vizserver can use XDM managed pipes** radio buttons are related to dynamic pipe allocation. See "Graphics Pipe Allocation Guidelines" on page 30 for more details.

The **Pipes managed by OpenGL Vizserver** radio buttons specify how the OpenGL Vizserver managed pipes are determined. **None** means that no pipes are managed by OpenGL Vizserver; **All available** means that every pipe that is not managed by XDM is managed by OpenGL Vizserver; and **Specified** means that pipes that are specifically selected are managed by OpenGL Vizserver. In the last case, the **Vizserver** column in the bottom table is enabled for pipe selection.

To allocate or deallocate a graphics pipe, click the square corresponding to the graphics pipe and the server that you want to set up. The allocated one becomes a red-colored square. Press the **Configure** button. This will overwrite your current `/var/X11/xdm/Xservers` file.

For details about the X server, see the `xdm`(1) and `Xserver`(1) man pages.

Configuration Parameters

The values in the **Configuration Values** panel, as shown in Figure 2-4 on page 15, are from entries of the `/var/vizserver/config` file. If this file does not exist, `vsconfig` will use its internal default values to set the fields in the panel. Usually you do not need to change these values, but there are some entries that you might want to modify depending on your system.

The **Session Kill Notify** edit box is enabled only if the **Reservation System** radio boxes are in **Active** state.

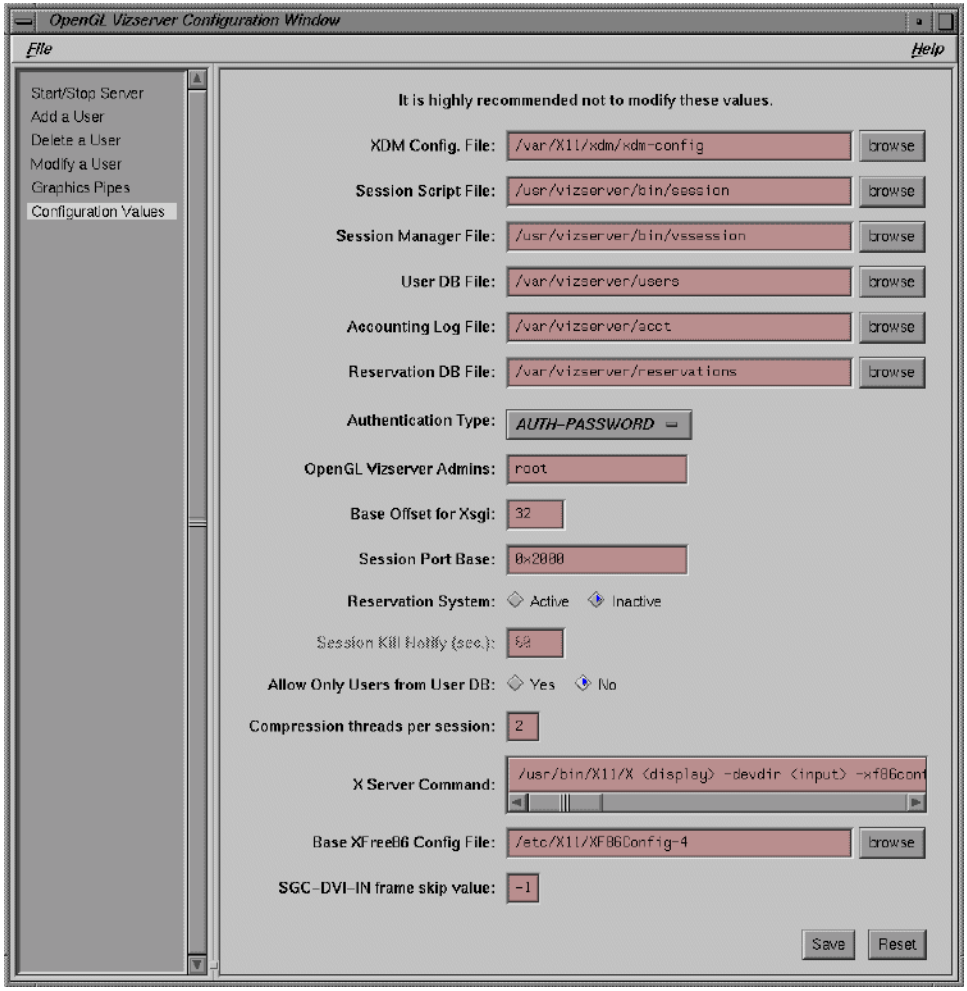

Figure 2-4 Configuration Values Panel

For a more detailed description of each entry, see “The /var/vizserver/config File” on page 22.

Managing Users

Users on the machine where the OpenGL Vizserver server is running can use OpenGL Vizserver by default, but if the entry `Vizserver*UserDBStrictInterp` in `/var/vizserver/config` is set to `True`, they should be listed in the user database file. See "The /var/vizserver/config File" on page 22 for more details.

This section describes how to add, delete, or modify a user in the user database file by using `vsconfig`.

All the operations on Adding/Modifying/Deleting a user will not update the `/var/vizserver/users` file until you save them by selecting **File** > **Users** > **Save users**. They are updated on the internal database of `vsconfig`.

For more details about the `/var/vizserver/users` file, see "The /var/vizserver/users File" on page 21.

Adding a User

Select **Add a User** to view that panel, as shown in Figure 2-5.

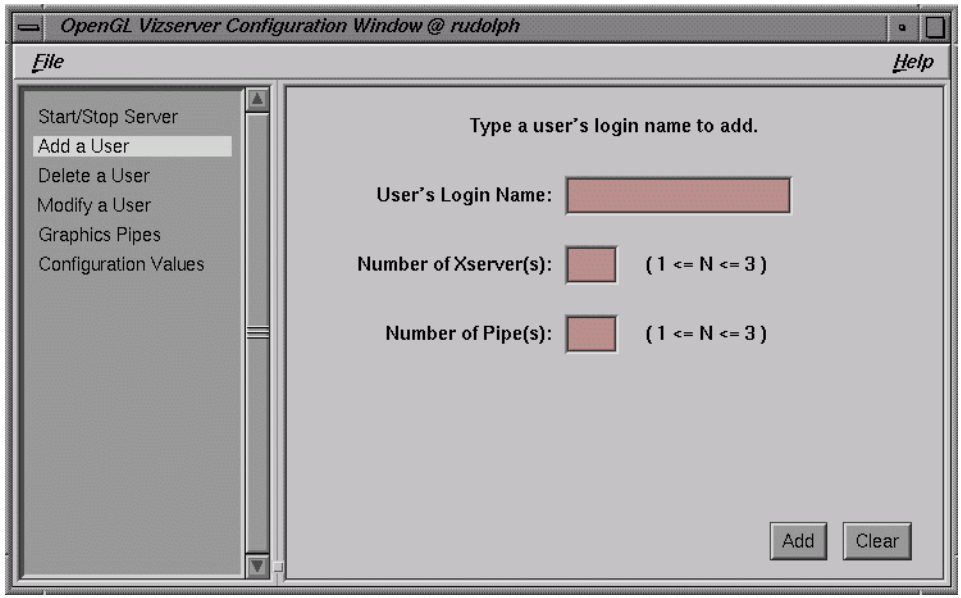

Figure 2-5 Add a User Panel

To add a new user for OpenGL Vizserver, fill in the blanks and press the **Add** button.

The user's login name should be the same as given in the `/etc/passwd` or NIS password database if enabled.

The number of X servers is the number of active X servers allowed to this user. Since each X server requires at least one graphics pipe, you cannot have more X servers than the number of grahics pipes in your system.

The number of pipes are the maximum number of pipes assigned to this user. Again, the user cannot have more pipes than the number of graphics pipes in the system. So a brief line next to these fields (`1 <= N <=` *k*), where *k* is the number of graphics pipes in your system and is automatically configured depending on the system) is helpful to help decide which number to put into these fields.

To save your work, select **File** > **Users** > **Save users**.

Modifying a User

Select **Modify a User** to view that panel, as shown in Figure 2-6.

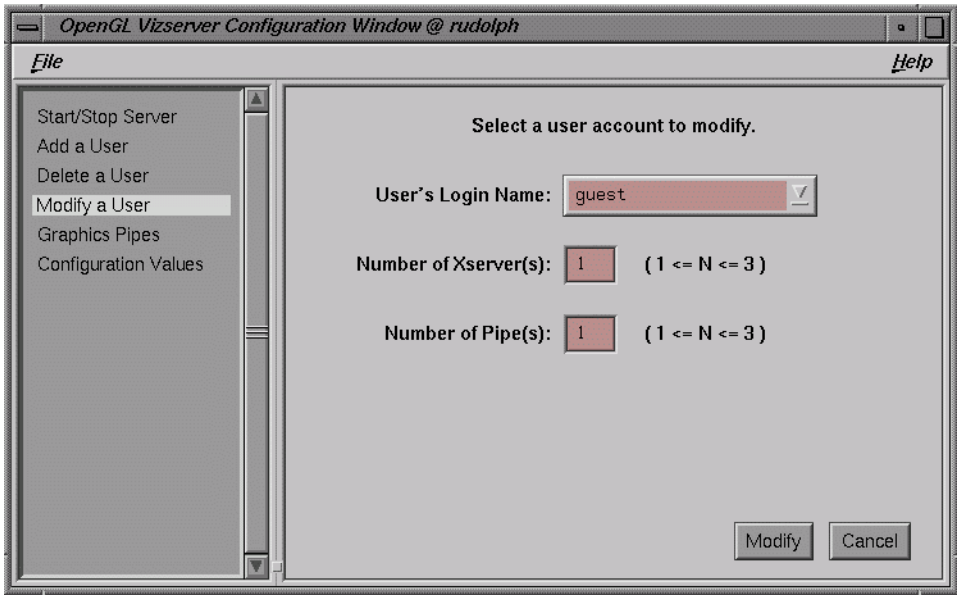

Figure 2-6 Modify a User Panel

To modify the number of X servers or pipes assigned to a user, choose the user's login name from the pulldown list by clicking a downward-pointing arrow. Change the values in these fields as needed. Press the **Modify** button to save your work.

Deleting a User

Select **Delete a User** to view that panel, as shown in Figure 2-7.

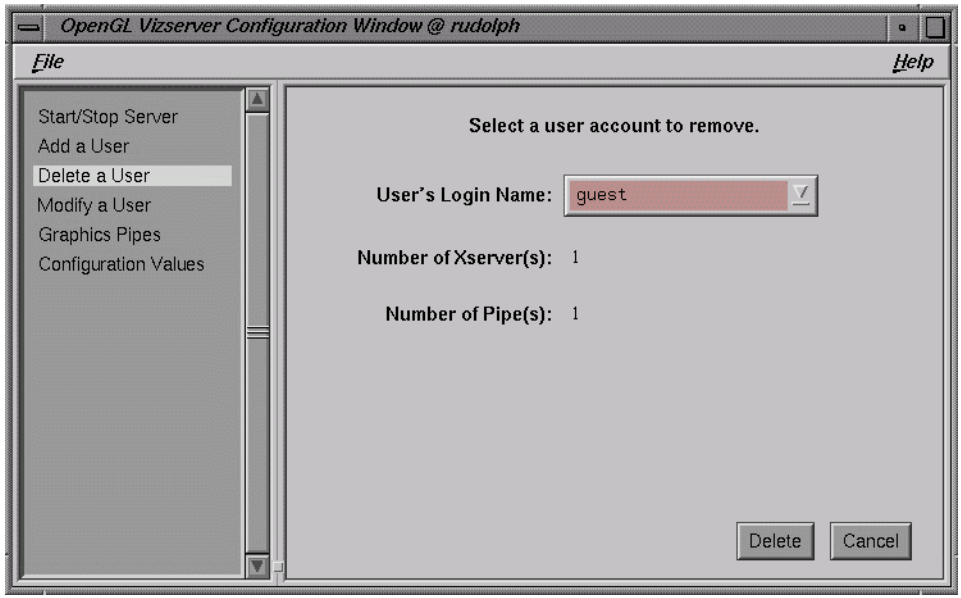

Figure 2-7 Delete a User Panel

To delete a user from user database, select a user from the pulldown list by clicking a downward-pointing arrow. Press the **Delete** button.

Configuring the Reservation Web Interface

OpenGL Vizserver provides a reservation mechanism that allows a user of the OpenGL Vizserver system to reserve a pipe at a specific time slot. To use this mechanism, you must have the `vizserver_server.web` module and either Netscape Fasttrack or SGI Apache web server installed.

In the **Configuration Values** panel of `vsconfig`, select the **Active** radio button of the **Reservation System** field. Then the **Session Kill Notify (sec.)** field becomes visible, as shown in Figure 2-8.

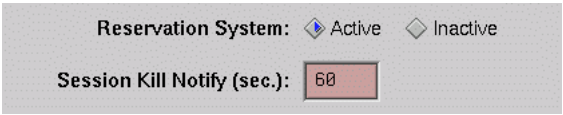

Figure 2-8 Setting Reservation System Parameters

Making the **Reservation System** active means that a reservation is required for a user to start a session. **Session Kill Notify** indicates how many seconds before the end of a session a warning message is sent to the user.

The installed web interface is configured to work under the Netscape Fasttrack or SGI Apache web servers. A sample reservation web interface can be started from the following URL:

http://*remote-host*/vizserver/reservation

For *remote-host*, specify the OpenGL Vizserver server's hostname. For more details about how to use the reservation web interface, see the *OpenGL Vizserver User's Guide*.

You can configure the web interface to work with other web servers. To do so, you must ensure the following:

- All the CGI files must be in the same directory.
- The `index.html` file must redirect the web browser to `welcome.cgi`. If `welcome.cgi` is moved, `index.html` should be edited accordingly.
- The `Reservation*ImagePath` entry in `/var/vizserver/reservation_client.conf` must point to the directory that holds the images.

For more details, see "The /var/vizserver/reservation_client.conf File" on page 27.

Configuration Files

There are three important files related to the OpenGL Vizserver server configuration, as described in the following sections:

- "The /var/vizserver/users File" on page 21
- "The /var/vizserver/config File" on page 22
- "The /var/vizserver/reservation_client.conf File" on page 27

The `vsadmin` command can be used to load the server's configuration files dynamically after changing the values in them.

The `/var/vizserver/users` File

The `/var/vizserver/users` file contains a list of user login names that are allowed to connect to the OpenGL Vizserver server. Each entry is listed in a separate line and has three fields separated by colons, as follows:

name`:`*servers*`:`*pipes*

The default configuration of the OpenGL Vizserver server is to allow users who can log in to the machine to use OpenGL Vizserver without any limitation on the number of pipes. If you want to change this access scheme, set the value of `Vizserver*UserDBStrictInterp` in the `/var/vizserver/config` file to `True` and add users in this file. Then only the users listed in this file can use OpenGL Vizserver with the number of pipes assigned to them.

name`:`

- This entry specifies the user's login name as given in `/etc/passwd` or NIS password database if enabled.
- User must have the access to use the system first.

servers`:`

- This entry specifies the number of active X servers allowed to the user. In practice, a user cannot have more than one X server.
- Cannot be empty.

- Since an X server needs at least one graphics pipe, the number cannot be greater than the number of graphics pipes that the server system has.

pipes:

- This entry specifies the maximum number of pipes assigned to the user.
- Cannot be empty.
- The number cannot be greater than the number of the graphics pipes on the server system.

When you install a new version of the OpenGL Vizserver server module, the old `/var/vizserver/users` file is kept, unchanged, and a new one is installed as `/var/vizserver/users.N`.

The `/var/vizserver/config` File

The `/var/vizserver/config` file contains entries in the following format:

`Vizserver*`*entry*`:` *value*

Generally default values in this file are enough to start the OpenGL Vizserver server manager and you do not need to modify entries in this file. But if you want to customize your system for your specific needs, you can change them by using `vsconfig` or by modifying this file manually. It is recommended that you use `vsconfig`.

`Vizserver*UserDBPath:`

- This entry specifies the location of the user database file.
- The default value is `/var/vizserver/users`.

`Vizserver*UserDBStrictInterp`:

- This entry specifies whether only users from the user database can use OpenGL Vizserver.
- If `True`, a user should be listed in the user database file to use OpenGL Vizserver. For information on how to add a user in the database file, see "Allocating Graphics Pipes for OpenGL Vizserver" on page 12.

- Setting this to `False` means that if a user does not exist in the user database file, that user is still allowed to use OpenGL Vizserver without any limit on the number of servers and pipes.
- The default value is `False`.

`Vizserver*AuthType:`

- This is the user authentication mechanism to be used by the server.
- The authentication mechanisms provided with the server are `AUTH-PASSWORD` and `AUTH-PAM`. See section "User Authentication" on page 29 for more details about these two mechanisms.
- Depending on your specific needs, other authentication modules can be developed using the OpenGL Vizserver authentication API.
- The default value is `AUTH-PASSWORD`.

`Vizserver*AcctPath:`

- This entry specifies an accounting log file in which `vsserver` writes when a session starts and stops.
- This data can be dumped using `vsacct`(1m).
- The default value is `/var/vizserver/acct`.

`Vizserver*SessionPath:`

- This entry specifies the session startup shell script file.
- This is passed to the OpenGL Vizserver session manager.
- The default value is `/usr/vizserver/bin/session`.

`Vizserver*SessionMgrPath`:

- This entry specifies the OpenGL Vizserver session manager path.
- This is run by the OpenGL Vizserver server manager when a session is started.
- The default value is `/usr/vizserver/bin/vssession`.

`Vizserver*XDMConfigPath:`

- This entry specifies the XDM configuration file, which specifies resources to control the behavior of XDM.

- OpenGL Vizserver gets a value from the `DisplayManager*servers` field in this file because it is a system-wide default X server file, usually `/var/X11/xdm/Xservers`, used by XDM.
- OpenGL Vizserver reads `/var/X11/xdm/Xservers` to determine which pipes are managed and ready for use by OpenGL Vizserver.
- The default value is `/var/X11/xdm/xdm-config`.

`Vizserver*UseXDMPipes`:

- This entry specifies whether OpenGL Vizserver can use XDM-managed graphics pipes for its sessions.
- If `False`, this activates static pipe allocation mode. See "Graphics Pipe Allocation Guidelines" on page 30 for more details.
- This corresponds to the **OpenGL Vizserver can use XDM managed pipes** radio buttons in the **Graphics Pipes** panel of `vsconfig`.
- The default value is `True`.

`Vizserver*Boards:`

- This entry defines the set of graphics pipes available to the OpenGL Vizserver server manager.
- There are three possible scenarios:
  - `all` indicates that any pipes not currently managed by XDM should be managed by `vsserver`.
  - `none` indicates that no pipe is managed by `vsserver`.
  - *b#, ..., b#* indicates that the specified pipes not managed by XDM should be managed by `vsserver`.
- The graphics board numbers can be determined by examining the report from `gfxinfo`(1G).
- The default value is `all`.

`Vizserver*DisplayOffset:`

- This entry specifies the base display offset number for the X server started by an OpenGL Vizserver session.
- X server issued by XDM starts its number from 0.

- The maximum number of graphics pipes in a system is 32. So the biggest number for an XDM X server would be 31.
- The default value is `32`.

`Vizserver*ServerCommand`:

- This entry specifies the command-line entry for the X server started by an OpenGL Vizserver session.
- The default value differs between Onyx4 servers and other SGI servers.

  Onyx4 servers use the XFree86 command line:

  `/usr/bin/X11/x <`*display*`> -devdir <`*input*`> -xf86config <`*config*`> -layout <`*layout*`>`

  Other SGI servers use the `Xsgi` command line:

  ```
  /usr/bin/X11/X <display> -boards <boards> -devdir <input> -bs
  -nobitscale -depth 8 -class PseudoColor -c -solidroot sgiblue
  -cursorFG red -cursorBG white
  ```

- The <*display*>, <*boards*>, <*config*>, <*input*>, and <*layout*> parameters are replaced by OpenGL Vizserver in run time. Modify only the parts of the command line that does not involve these parameters (from `-bs` onwards).

`Vizserver*Admins:`

- This entry specifies a comma-separated list of user login names that have administrative access to the OpenGL Vizserver server manager.
- The user listed in this field can kill sessions on the server as well as update server configuration changes in the `/var/vizserver/config` file or XDM configuration.
- The default value is `root`.
- See the `vsadmin`(1) man page.

`Vizserver*SessionPortBase:`

- This entry specifies the base port that OpenGL Vizserver should use when starting a session.
- Each session uses three ports (*base* + 3 x *pipe*, *base* + 3 x *pipe* + 1, and *base* + 3 x *pipe* + 2), where *base* is the value of this entry and *pipe* is the graphics pipe number that the session is running on. For example, if the `SessionPortBase` is 0x2000 and a session is running on pipe 0, the session uses ports 0x2000, 0x2001, and 0x2002.

- Total 3 x *npipes* number of ports should be opened in the server's firewall to enable access through firewalls, where *npipes* is the number of graphics pipes in the server.
- OpenGL Vizserver listens on port 7051 (0x1b8b) for initial connections.
- The default value is `0x2000`.

`Vizserver*ReservationPath:`

- This entry specifies a `mdbm` database file path for reservation data.
- The default value is `/var/vizserver/reservations`.

`Vizserver*ReservationActive:`

- If `True`, a reservation is required for a user to start a session.
- If `False`, a user can start a session using any graphics pipes that are managed by OpenGL Vizserver.
- The default value is `False`.

`Vizserver*ReservationEndSessionNotify:`

- This entry specifies how many seconds before the end of a session a warning is sent to the user.
- If `Vizserver*ReservationActive` is set to `False`, this value is of no use.
- The default value is `60`.

`Vizserver*CompressionThreads:`

- This entry specifies how many compression threads are available for compressors that support multithreading.
- Currently, all compressors included in OpenGL Vizserver support multiple threads. This will allow for performance scalability on machines with a large number of CPUs.
- The default value for this setting is `2`.

`Vizserver*BaseXF86Config:`

- This entry specifies which file should be used by OpenGL Vizserver as the basis for generating the temporary XFree86 configuration files needed for starting X servers on Onyx4 systems.

- This entry is applicable (and will be displayed in `vsconfig`) only on Onyx4 systems.
- Change this value if you want OpenGL Vizserver to use an XFree86 configuration file that is customized to your needs.
- Default value is `/etc/X11/XF86Config`.

`Vizserver*SGCFrameSkip:`

- This entry specifies the frame-skip value when using an SGC-DVI-IN hardware readback device.
- This entry is applicable (and will be displayed in `vsconfig`) only on Onyx4 systems with SGC-DVI-IN devices and when the subsystem `vizserver_server.modules.readback` is installed.
- If set to `–1`, the OpenGL Vizserver session manager will determine the optimal frame-skip value automatically.
- Change this parameter to get better performance or lower system resources usage. This change will override the value automatically set by the OpenGL Vizserver session manager.
- Default value is `–1`.

The `/var/vizserver/reservation_client.conf` File

The `/var/vizserver/reservation_client.conf` file contains the necessary information to configure the reservation web interface shipped with OpenGL Vizserver. This file contains entries in the following format:

`Reservation*`*`entry`*`:` *value*

The default values in this file are set for the reservation web interface shipped in the OpenGL Vizserver reservation module. If you develop your own reservation web interface using the OpenGL Vizserver reservation API, you may need to modify these values.

Note: The `reservation_client.conf` file is used by the reservation web interface that is shipped with OpenGL Vizserver. It is not used by the OpenGL Vizserver server manager or by any other reservation program developed using the reservation API.

`Reservation*Servers:`

- This entry specifies the name of the OpenGL Vizserver server machine where the reservation is made. Only one host is supported at this time.
- The host specified in this entry must have a running OpenGL Vizserver server manager for the reservation interface to work.
- The default value is the local host.

`Reservation*ImagePath:`

- This entry specifies the directory in which the images used by the web interface are kept.
- This path is relative to the web server's HTML directory.
- The default value is `/vizserver/images`.

`Reservation*ConnectionTimeout:`

- This entry specifies the timeout (in minutes) for disconnecting a non-active user from the reservation web interface.
- After this amount of time, a reservation session is closed, and the user should log in again. Each operation of the user on the web interface resets the timer.
- The default value is `5`.

`Reservation*MinimalTimeslot:`

- This entry specifies the minimal length (in minutes) of a reserved time slot.
- It must be at least 1 minute.
- The default value is `30`.

`Reservation*MaximalTimeslot:`

- This entry specifies the maximum length (in minutes) of a reserved time slot.
- A value of `0` indicates that there is no maximum to the reservation length.
- The default value is `240`.

See the `vsreservation`(1m) man pages for more details.

User Authentication

Two authentication mechanisms are currently provided with OpenGL Vizserver:

AUTH-PASSWORD

AUTH-PAM

AUTH-PASSWORD

The defualt authentication scheme used by the server is AUTH-PASSWORD, an unencrypted user/password mechanism based on the system's `passwd` database.

AUTH-PAM

This mechanism uses the Pluggable Authentication Modules (PAM) mechanism provided with IRIX 6.5.22 or later. When using the AUTH-PAM module, the OpenGL Vizserver PAM configuration file (`/etc/pam.d/vizserver`) determines how authentication is performed by the server.

Notes:

- In order to use the AUTH-PAM mechanism, you need to install the `vizserver_server.modules.pam_auth` subsystem.
- When using the AUTH-PAM module in the server, no authentication module is needed in the client.

For more information about PAM usage and configuration, see the following documents:

- *Linux-PAM System Administrators' Guide* (`/usr/share/doc/pam/html/pam.html`)
- PAM(8) man page

Graphics Pipe Allocation Guidelines

As mentioned in “Allocating Graphics Pipes for OpenGL Vizserver” on page 12, the managed graphics pipes in the OpenGL Vizserver system are managed by either XDM or OpenGL Vizserver.

When graphics pipes are allocated to the OpenGL Vizserver sessions by the OpenGL Vizserver server manager, there are two types of allocation methods used: static pipe allocation and dynamic pipe allocation. The terms *static* and *dynamic* refer to the mobility of graphics pipes between XDM and OpenGL Vizserver.

As noted earlier, all graphics pipes allocated for a session are used by the session’s X server, although it might be the case that only a subset of these pipes are actually being served by OpenGL Vizserver. The status of the **Active Screens** check boxes in the **Session Start** window of the client determines which pipes/screens are served. For example, a user might request to start a four-pipe session, of which only screens 0 and 2 are specified to be served. In this case, the following will happen:

- Four pipes will be allocated to the session by the server manager.
- The X server used by this session will include four screens: 0, 1, 2 and 3.
- Only screens 0 and 2 will be actually served by OpenGL Vizserver to the client.

This section describes the policy for allocating graphics pipes by the server manager and the process for configuring with this policy in the following subsections:

Relevant Configuration File Parameters

The three parameters in the server's configuration that affect graphics pipe allocation are the following:

- `Vizserver*Boards`: Specifies which graphics pipes are managed by OpenGL Vizserver.
- `Vizserver*UseXDMPipes`: Specifies whether OpenGL Vizserver can use XDM-managed graphic pipes for its sessions.
- `Vizserver*ReservationActive`: Specifies whether a graphics pipe reservation by the user is required in order to use the OpenGL Vizserver managed pipes.

For more details about these parameters, see "The /var/vizserver/config File" on page 22.

Static Pipe Allocation

When the `Vizserver*UseXDMPipes` parameter's value is `False`, the server operates in a *static pipe allocation* mode. In this mode, OpenGL Vizserver can allocate only the graphics pipes that it manages.

If the `Vizserver*ReservationActive` parameter's value is `False`, a user can open a session using any graphics pipes that are managed by OpenGL Vizserver (subject to availability).

If the `Vizserver*ReservationActive` parameter's value is `True`, a user cannot have a session using more than the maximum number of graphics pipes reserved. If no reservation was made by a user, the user cannot open a session at all.

Dynamic Pipe Allocation

When the `Vizserver*UseXDMPipes` parameter's value is `True`, the server operates in a *dynamic pipe allocation mode*. In this mode, OpenGL Vizserver can allocate the graphics pipes that it manages, as well as the graphics pipes managed by XDM.

OpenGL Vizserver allocates XDM-managed pipes for a session's use only if the X server that currently uses the graphics pipes is not logged in. In other words, the X server is in the *login* stage and the login screen is displayed.

In order for the server to know which X servers are logged in and which are not, three scripts used by XDM need to be changed to record the X server's state in the system's `utmpx` database. This change is made automatically when installing the server, by installing the new scripts on the system (in `/var/X11/xdm`), and modifying the XDM configuration file (`/var/X11/xdm/xdm-config`).

The following are the pertinent scripts:

`Xlogin`	This script starts the login process of the X server. Upon installation of OpenGL Vizserver, this script is replaced by `Xlogin.vizserver`.
`Xstartup`	This script is run after a user has logged into the X server. Upon installation of OpenGL Vizserver, this script is replaced by `Xstartup.vizserver`.
`Xreset`	This script is run after a user has logged out of the X server. Upon installation of OpenGL Vizserver, this script is replaced by `Xreset.vizserver`.

Note: The new `*.vizserver` scripts are based on the default scripts installed by IRIX. If these scripts on the installed machine were changed, you will need to copy the relevant lines from the `*.vizserver` scripts into your local scripts manually.

The server reads an XDM X server file (usually `/var/X11/xdm/Xservers`) to understand the current state of the system graphics pipes. It also changes the file every time XDM-managed graphics pipes are allocated dynamically or returned to XDM. Therefore, it is strongly recommended not to modify the contents of the file externally while X servers are dynamically allocated.

Note: Changing graphics pipes allocation configuration might cause active OpenGL Vizserver sessions to terminate.

Dynamic Pipe Allocation Policy

When the server is in *dynamic pipe allocation* mode and the `Vizserver*ReservationActive` parameter's value is `False`, the graphic pipe allocation policy is as follows:

1. The server allocates as many of the OpenGL Vizserver managed graphics pipes as it can.
2. If that is not enough to fullfill the session's needs, the server tries to allocate available graphic pipes that belong to X servers from which graphic pipes were already allocated.
3. If that is not enough to fullfill the session's needs, the server tries to allocate XDM-managed pipes from X servers that are not logged in.

If the `Vizserver*ReservationActive` parameter's value is `True`, the policy is similar, with one difference: the server will not allocate graphics pipes from its own managed pipes more than the number of pipes reserved by the user (that is, if no reservation was done, only XDM-managed graphics pipes will be allocated).

When the server allocates an XDM-managed graphics pipe, the XDM X server using the pipe is killed and the session's own X server can use this pipe.

As mentioned previously, more than one session might use graphics pipes from the same XDM X server. When all the graphics pipes used by the sessions are freed, after sessions end, the XDM X server is restarted and returned to a login state.

Hardware Readback

OpenGL Vizserver supports the use of hardware readback using an SGC-DVI-IN device in conjunction with the digital output from an Onyx4 graphics pipe. An SGC-DVI-IN device is a digital video interactive (DVI) capture device. It connects to the server's PCI-X bus and is used to drastically improve the readback rate of OpenGL Vizserver from the server-side X server by connecting the pipe's DVI output to the SGC-DVI-IN device input using a DVI cable.

OpenGL Vizserver will use SGC-DVI-IN devices to perform the readback needed for its operation if the following conditions are true:

- The OpenGL Vizserver server host is an Onyx4 system.
- The OpenGL Vizserver server host has SGC-DVI-IN devices installed.
- All pipes belonging to the subset of "active screens"—that is, screens user-selected for OpenGL Vizserver to serve to clients—are connected to SGC-DVI-IN devices.

Note that if any one of the session's allocated pipes belonging to an active screen is not connected to an SGC-DVI-IN device, readback from all pipes will revert to software.

Upon session startup, OpenGL Vizserver will automatically detect the connections among the session's allocated pipes and the SGC-DVI-IN devices and use the required devices appropriately.

Pipe allocation by the OpenGL Vizserver server manager is independent of the SGC-DVI-IN devices. That is, if some of the pipes on your system are connected to an SGC-DVI-IN device and some are not, the only way to force the allocation of pipes connected to SGC-DVI-IN devices is by changing the `Vizserver*Boards` value in the server's configuration file `/var/vizserver/config` to indicate which pipes are connected to SGC-DVI-IN devices.

To provide a uniform user experience where all sessions run fast regardless of which graphics pipes they are allocated, ensure that all graphics pipes used by OpenGL Vizserver are connected to an SGC-DVI-IN device.

Tuning

This chapter explains the overall architecture of OpenGL Vizserver 3.*x*, how to measure and monitor OpenGL Vizserver performance, and how to estimate the network bandwidth required by OpenGL Vizserver. The following topics are covered:

Tuning generally implies matching the system capacity and your workload to get a better performance from your system. You can change the system hardware or software to match the workload or you can reduce the workload to match the system.

So why do we need tuning? Many answers are possible, but they can be summarized as follows:

- Isolate and understand performance behavior
- Use resources more efficiently
- Understand performance bottlenecks
- Get a better performance

There are many components that you can tune, depending on your needs on the system. This chapter discusses the tuning issues on OpenGL Vizserver only.

To extract top performance from a system, it is important to understand the architecture of the system. The next section describes the architecture of the OpenGL Vizserver system.

Understanding the OpenGL Vizserver Pipeline

The overall diagram of OpenGL Vizserver is shown in Figure 3-1.

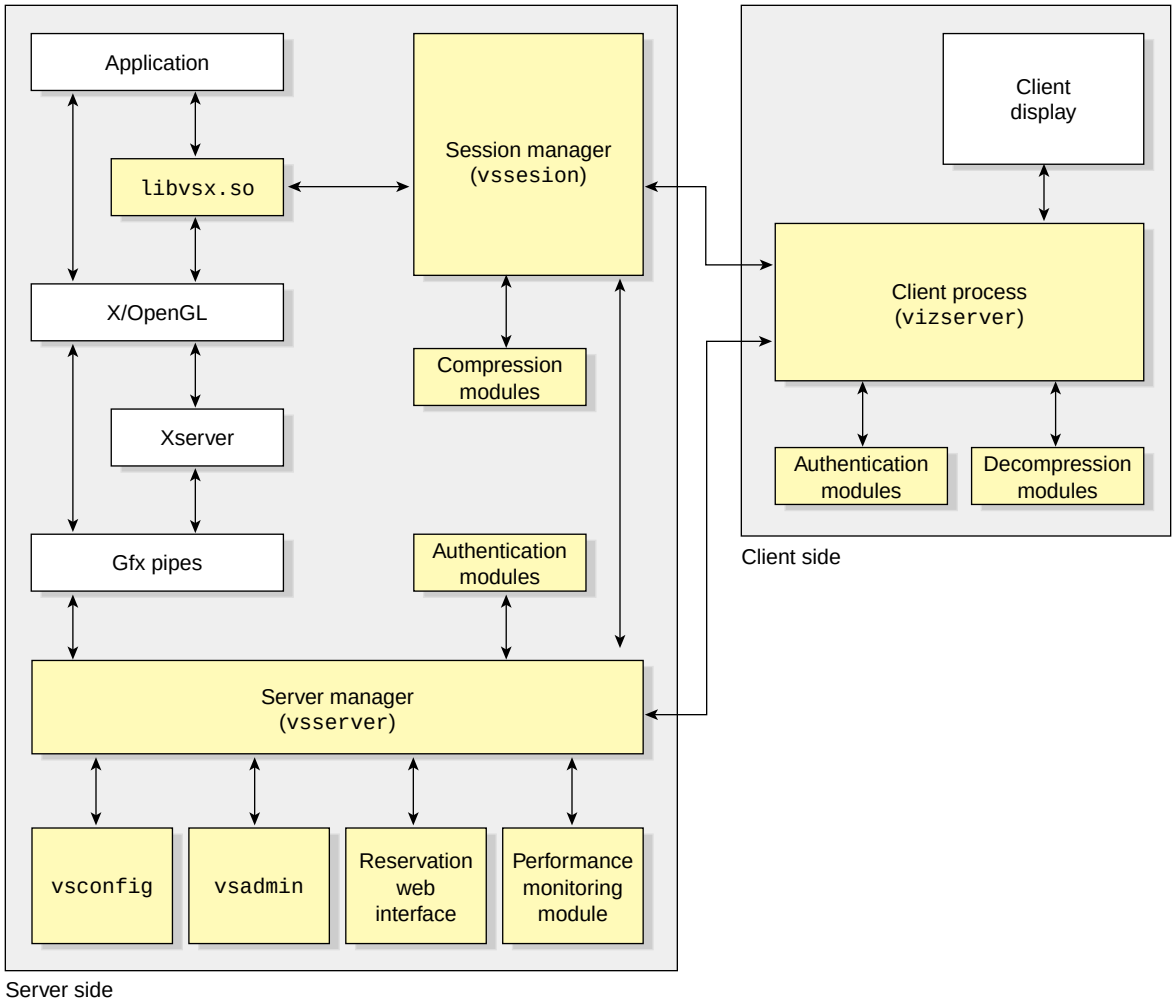

Figure 3-1 Overall Diagram of OpenGL Vizserver

The following steps describe how the OpenGL Vizserver components work from the point that a user connects to OpenGL Vizserver to the point that the user logs out of the OpenGL Vizserver server after running the application under OpenGL Vizserver.

For detailed instructions on how to use the OpenGL Vizserver client GUI, see the *OpenGL Vizserver User's Guide*.

How It Operates

OpenGL Vizserver provides two types of sessions: single-user sessions and collaborative sessions. Single-user sessions involve one client user using graphics pipes on the OpenGL Vizserver server. Collaborative sessions allow multiple distant users, one of them being a master and the rest of them being nonmasters, to display and interact with the same application in real time.

Single-User Session

The following steps describe running a single session:

1. The server manager (`vsserver`) is started as a daemon and listens for client's incoming connections.
2. The OpenGL Vizserver client program (`vizserver`) calls the server with a hostname.
3. `vsserver` replies.
4. `vizserver` sends the authentication data (usually user's login name and password).
5. `vssserver` dynamically loads an authentication module and verifies the user.
6. If the user is authenticated to `vsserver`, the **Start Session**, **Join Session**, and **Log out** buttons in the client GUI (`vizserver`) become active.
7. The user presses the **Start Session** button and chooses a **Single-User** session type in a **Session Start** window. The user chooses other initial configuration options if necessary.
8. `vsserver` allocates the requested number of graphics pipes and launches a session process.
9. `vssession` initializes the allocated graphics pipes and calls a session script.

10. `vizserver` presents an **OpenGL Vizserver Session Control** window and an **OpenGL Vizserver Console** window to the user.
11. The user launches applications on the **OpenGL Vizserver Console** window displayed on the client. The component `libvsx.so` is loaded by the applications and handles some X11, GL, and GLX calls by overriding them.
12. `vssession` captures the images rendered by the applications, compresses the images, and transmits them to the remote client.
13. The user presses the **Stop Session** button to end the session after running applications.
14. The user presses the **Log out** button to log out from the OpenGL Vizserver server.

Collaborative Session

There are two types of collaborative sessions: local and remote. A local collaborative session is a session that involves a local client on the server running an X server and multiple remote clients to work together on the same application. A remote collaborative session is a session that involves a remote client starting a session and multiple remote clients working together on the same application. Whoever starts a session becomes the master of the session. In local collaborative sessions, only the local client can start a session.

The following steps describe running a collaborative session. Steps 1 to 6 are the same as running a single-user session:

1. The server manager (`vsserver`) is started as a daemon and listens for client's incoming connections.
2. The OpenGL Vizserver client program (`vizserver`) calls the server with a hostname.
3. `vsserver` replies.
4. `vizserver` sends the authentication data (usually user's login name and password).
5. `vssserver` dynamically loads an authentication module and verifies the user.
6. If the user is authenticated to `vsserver`, the **Start Session**, **Join Session**, and **Log out** buttons in the client GUI (`vizserver`) become active.

7. The user presses the **Start Session** button and chooses a **Collaboration** session type and types the session's name in a **Session Start** window. The user who starts the session becomes the master of the session.
8. `vsserver` distinguishes whether the session is local or remote and allocates the requested number of graphics pipes for a remote collaborative session and launches a session process. For a local collaborative session, `vsserver` just uses the number of graphics pipes that are already allocated to a currently running X server.
9. `vssession` initializes the allocated graphics pipes and calls a session script. For local collaborative sessions, `vssession` does not initialize the graphics pipe.
10. `vizserver` presents an **OpenGL Vizserver Session Control** window and an **OpenGL Vizserver Console** window to the master.
11. The master launches applications on the **OpenGL Vizserver Console** window or waits for the other users to join the session. When the applications are launched, `libvsx.so` is loaded and handles some X11, GL, and GLX calls by overriding them.
12. `vssession` captures the images rendered by the applications, compresses the images, and transmits them to all the clients in the session. For local collaborative sessions, `vssession` does not compress the images and sends them to the local client, that is, the master.
13. To join the session, a user completes steps 1 through 6 and then presses the **Join Session** button and fills in the **Session Name** field in a **Session Join** window.
14. `vsserver` sends the message to the master of the requested session to ask whether or not the master accepts the request.
15. If the request is approved by the master, `vsserver` sends the message to `vssession` to accept the user (nonmaster) joining and also notifies all the clients that participated in the session that a new user has joined.
16. The nonmaster presses the **Leave Session** button to leave the session and the session continues. If the master leaves the session, the session is stopped.
17. Each user presses the **Log out** button to log out from the OpenGL Vizserver server.

Main Components

OpenGL Vizserver consists of client side components and server side components.

Client Process (`vizserver`):

- Basic GUI for the user.
- Initiates a connection to a server.
- Reads the compressed image from the server and decompresses it using a decompression module.
- Displays the decompressed images.

Server manager (`vsserver`):

- A daemon process running on the server.
- Keeps configuration parameters and provides them to other components in the system.
- Waits for an initial connection from clients.
- Responsible for launching `vssession` after user's authentication.
- Allocates the number of graphics pipes the user requests.
- Responsible for handling different types of sessions and joining in collaborative sessions.

Session manager (`vssession`):

- Captures the images rendered by the application.
- Compresses the images using a compression module.
- Transmits them to remote clients.
- Receives keyboard or mouse events from clients and transfers them to the application.

`libvsx.so`:

- Transparent interface library.
- Loaded by the application and overrides some X11, GL, and GLX calls.

- Keeps track of the application windows' creation and destroys and notifies `vsssession`, which keeps track of the application window.
- Catches **glFlush()**, **glFinish()**, and **glXSwapBuffers()** to make `vssession` grab the frame buffer contents.

`vsconfig`:

- A GUI for the OpenGL Vizserver server's configuration.
- Starts and stops the `vsserver` process.

`vsadmin`:

- A simple command-line administration tool for checking and managing active client connections.

Reservation web interface:

- A set of Common Gateway Interface (CGI) programs, which provides a web-based interface to the reservation system.

Performance monitoring module:

- Performance Co-Pilot (PCP) OpenGL Vizserver Performance Metrics Domain Agent (PMDA).
- Provides an interface to PCP monitoring tools.

Tuning Objectives

There are different tuning objectives, depending on your situation and various tuning options.

Since OpenGL Vizserver reads the frame buffer images and sends them to the client, there are several important factors that affect the OpenGL Vizserver performance.

- Capturing frame buffer image
- Compressing the image
- Transmitting the image to the client
- Decompressing the compressed image at the client side

So OpenGL Vizserver tuning objectives are the following:

- Maximum network bandwidth
- Maximum frame readback rate
- Minimum frame drop rate
- Minimum network latency

These objectives allow the user in a remote client to feel that the application is running locally on a huge, powerful graphics machine.

Understanding the Environment

It is important to understand how your system is configured and what the system capacity is, such as number of CPUs, memory size, number of graphics pipes, and so on, when you measure the application or system performance.

Not all applications require the same amount of system resources. So determine the application that you use most and how many applications will be used at the same time. Also determine the acceptable response time for interactive users.

Measuring the Application Performance Locally

If an application itself, not running on OpenGL Vizserver, already oversaturates most of the system resources and shows a poor performance, there would be no performance improvement on measuring the application performance with OpenGL Vizserver. Measure the application performance locally and try to get a better performance from the application itself first, before running it with OpenGL Vizserver.

SGI provides a collection of monitoring tools that can be used with applications: `top`, `sar`, `osview`, `gr_osview`, `timex`, Performance Co-Pilot (PCP), and so on. Each monitoring tool provides different performance metrics and features. So it is also important to choose the right tools for monitoring your application.

When measuring the application performance, you can launch the application and the monitoring tools together or use `cron` to get the performance data over a period of time.

The `timex` utility is good at determining the source of the problem. It reports how a particular application is using its CPU processing time. The following will show real, user, and system time spent executing your application:

`timex` *your_application*

When used with the `-s` option, `timex` reports total system activity that occurred during the execution interval of your application.

The `osview` and `gr_osview` utilities dynamically display various parts of the operating system's activity data. If you have a graphics workstation, you can use `gr_osview`. You can configure `gr_osview` to display several different types of information about your system's current status.

The `sar` utility reports the system's activity by category and essentially the same information as `osview`, but it also represents a snapshot of the system status. This utility is useful for monitoring system usage over a period of time to determine bottlenecks and system resource limitations.

`sar [`*options*`] [`*interval*`] [`*samples*`]`

It has options that allow sampling of a different category, such as cpu utilization (`-u` option) or graphics activity (`-g` option). Each option displays the data differently.

The command in the following example prints information about graphics activity 10 times at 5 second intervals.

```
$ sar -g 5 10
16:24:50 gcxsw/s ginpt/s gintr/s fintr/s swpbf/s
16:24:55       0       2      72       5       0
16:25:00       0       0      72       2       0
16:25:05       0       0      72       2       0
16:25:10       0       0      72       2       0
16:25:15       0       1      72       3       0
16:25:20       3      37      72      39       0
16:25:25       0      40      72      13       0
16:25:30       0      36      72       8       0
16:25:35       1      13      72      35       0
16:25:40       1      51      72      45       0
Average        1      18      72      15       0
```

It is also useful to take a snapshot of your system activity before and after an application, as shown in the following example:

```
/usr/lib/sa/sadc 1 1 report_file
run your_application
/usr/lib/sa/sadc 1 1 report_file
sar -A -f report_file
```

For more information about these monitering tools, see their respective man pages.

Measuring Network Bandwidth and Latency

Network bandwidth between two systems can be measured easily by using `ttcp`. The `ttcp` tool can be used to time the transmission and reception of data between two systems using the TCP or UDP protocols.

For testing, the receiver should be started first, with `-s` and `-r` options, and the transmitter later, with `-t` and `-r` options. The `-t` option means to start in transmit mode and the `-r` option means to start in receive mode.

To test TCP, use the following commands:

- On the receiving host: `/usr/etc/ttcp -r -s`
- On the transmitting host: `/usr/etc/ttcp -t -s` *receiving host*

To test UDP, use the following commands:

- On the receiving host: `/usr/etc/ttcp -r -s -u`
- On the transmitting host: `/usr/etc/ttcp -t -s -u` *receiving host*

Example 3-1 shows the testing of TCP performance from a server (`rampage`) to a client (`o2-alto`).

Example 3-1 Testing TCP Performance between Two Systems by Using `ttcp`

1. Enter the command from a receiving host (`o2-alto`).

```
o2-alto:~> /usr/etc/ttcp -r -s
ttcp-r: buflen=8192, nbuf=2048, align=16384/0, port=5001 tcp
ttcp-r: socket
```

2. Enter the command from a transmitting host (`rampage`).

```
rampage:~> /usr/etc/ttcp -t -s o2-alto
ttcp-t: buflen=8192, nbuf=2048, align=16384/0, port=5001  tcp  ->
o2-alto
ttcp-t: socket
```

3. After pausing, the results, similar to the following, are displayed on each host:

   In `o2-alto`:

```
ttcp-r: accept from 130.62.46.200
ttcp-r: 16777216 bytes in 1.58 real seconds = 10340.35 KB/sec +++
ttcp-r: 3775 I/O calls, msec/call = 0.43, calls/sec = 2382.50
ttcp-r: 0.0user 0.3sys 0:01real 23% 20maxrss 0+0pf 3583+326csw
```

   In `rampage`:

```
ttcp-t: connect
ttcp-t: 16777216 bytes in 1.58 real seconds = 10373.98 KB/sec +++
ttcp-t: 2048 I/O calls, msec/call = 0.79, calls/sec = 1296.75
ttcp-t: 0.0user 0.2sys 0:01real 15% 936maxrss 0+0pf 1369+493csw
```

Here we can see that network bandwidth from `rampage` to `o2-alto` is 10,340 KB/s.

There are other tools to use for measuring the network traffic.

- `ping` : To test the network access layer
- `netstat -s` : To view the configuration
- `ifconfig -a` : To see the status for all interfaces on the machine
- `traceroute` : To test the Internet layer

Monitoring OpenGL Vizserver Performance

OpenGL Vizserver total performance is affected by many factors, including the graphics frame buffer image readback rates, CPU speeds on both OpenGL Vizserver server and client for compression/decompression and network bandwidth.

The main purpose of monitoring OpenGL Vizserver performance is to find performance bottlenecks and ensure that an application running under OpenGL Vizserver gives the same performance as the application running locally on a huge graphics machine without using OpenGL Vizserver.

It is assumed that your application is already optimized and runs reasonably well in the current system configuration. To monitor OpenGL Vizserver performance, you can use various system tools, mentioned in “Measuring the Application Performance Locally” on page 42. However, with the values from these tools, it is difficult to understand how OpenGL Vizserver performs.

Starting with the OpenGL Vizserver 3.0 release, a Performance Co-Pilot (PCP) OpenGL Vizserver Performance Metric Domain Agent (PMDA) module and a text-based tool, `vsmonitor`(1m), are available. They are easy to use and useful to monitor the performance of each stage in the OpenGL Vizserver pipeline.

Performance Co-Pilot

Performance Co-Pilot (PCP) is an SGI product designed for monitoring and managing system-level performance. It provides a system-level suite of tools that cooperate to deliver distributed and integrated performance management services.

To use PCP OpenGL Vizserver PMDA to monitor OpenGL Vizserver performance, you need to install at least `pcp_eoe.sw.eoe` and `pcp_eoe.sw.monitor` in your system. The base `pcp_eoe` product is included in your IRIX 6.5 CD set and can be run without licenses. If you want more fully covered PCP services, you will need to install `pcp`, which requires PCP licenses.

To learn more about Performance Co-Pilot, see the following URL:

http://www.sgi.com/software/co-pilot

PCP OpenGL Vizserver PMDA

PCP OpenGL Vizserver PMDA acts as a gateway between a collection of performance data from the OpenGL Vizserver server and the Performance Metrics Collection Daemon (PMCD). PMCD acts as a mediator between PCP monitoring tools and PCP OpenGL Vizserver PMDA.

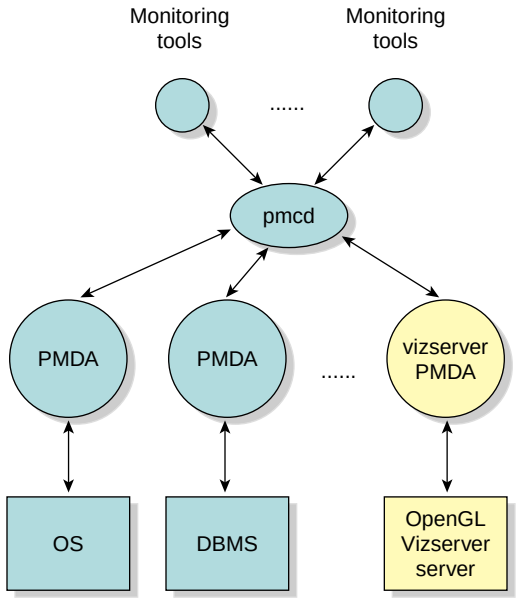

Figure 3-2 OpenGL Vizserver PMDA

Once OpenGL Vizserver PMDA is installed, the performance data from the PMDA becomes available immediately to all monitoring tools that connect to the PMDA process (`pmcd`). The PMDA can also be added or removed while `pmcd` continues operation.

To use OpenGL Vizserver PMDA, `vizserver_server.modules.perf` should be installed on your system. OpenGL Vizserver PMDA files are located in the `/var/pcp/pmdas/vizserver` directory and the executable image for OpenGL Vizserver PMDA is `/var/pcp/pmdas/vizserver/pmdavizserver`, using domain number `222`.

Starting a `pmcd` process automates the start of OpenGL Vizserver PMDA. Start the `pmcd` process by entering the following commands:

```
# chkconfig pmcd on
# /etc/init.d/pcp start
```

If `pmcd` is already running on your system, the above command will stop and restart the `pmcd` process.

If you have problems running `pmcd`, see the *Performance Co-Pilot User's and Administrator's Guide*.

You can also start OpenGL Vizserver PMDA without restarting the `pmcd` process. Go to the `/var/vizserver/pmdas/vizserver` directory and run the `Install` script. Choose both the `collector` and the `monitor` installation configuration options. Everything else is automated. If you still enounter problems, see the `README` file in the directory.

```
# cd /var/pcp/pmdas/vizserver
# ./Install

You will need to choose an appropriate configuration for installation
of the "vizserver" Performance Metrics Domain Agent (PMDA).

   collector     collect performance statistics on this system
   monitor       allow this system to monitor local and/or remote
   systems
   both          collector and monitor configuration for this system

Please enter c(ollector) or m(onitor) or b(oth) [b] both

Updating the Performance Metrics Name Space (PMNS) ...
Compiled PMNS contains
       357 hash table entries
       1533 leaf nodes
       239 non-leaf nodes
       15974 bytes of symbol table
Installing pmchart view(s) ...
Terminate PMDA if already installed ...
Installing files ...
Updating the PMCD control file, and notifying PMCD ...
Check vizserver metrics have appeared ... 23 metrics and 47 values
```

After the successful installation, you can see that the PMCD configuration file (`pmcd.conf`) has the OpenGL Vizserver PMDA as an entry.

```
$ cat /etc/pmcd.conf

# Name  Id      IPC     IPC Params      File/Cmd
irix    1       dso     irix_init       libirixpmda.so
pmcd    2       dso     pmcd_init       pmda_pmcd.so
proc    3       dso     proc_init       pmda_proc.so
vizserver       222     pipe    binary
/var/pcp/pmdas/vizserver/pmdavizserver -d 222
```

If `pcp.sw.base` is installed on your system, you can also use `pcp` to view the summary of PCP installation.

```
$ pcp

Performance Co-Pilot configuration on alto.engr.sgi.com:
platform: IRIX64 alto 6.5 10100655 IP30 64
hardware: 1 R10000 cpu, 3 disks, 1 xbow, 256MB RAM
timezone: PST8PDT
licenses: Collector Monitor
   pmcd: Version 2.2, 4 agents
   pmda: irix pmcd proc vizserver
```

Once the OpenGL Vizserver PMDA has been successfully installed, you can monitor it using any PCP monitoring tools, such as `pminfo`, `pmval`, `pmchart`, and so on.

The command `pminfo` displays various types of information about performance metrics. With the `-t` option, it lists all of the exported metrics and one-line help messages. The `-T` option shows more verbose help messages. With the `-f` option, it fetches and prints the values for all instances. See the `pminfo`(1) man page for more information.

```
$ pminfo -t vizserver
vizserver.nsession [number of ongoing sessions]
vizserver.npipe [number of pipes allocated to OpenGL Vizserver]
vizserver.period [sampling duration(sec)]
vizserver.all.readback.rate [average rate(KB/s) of readback]
vizserver.all.readback.time [average time(ms) spent on readback at each
frame]
vizserver.all.compress.rate [average rate(KB/s) on compression for all
sessions]
vizserver.all.compress.time [average time(ms) spent for compressing a
frame]
vizserver.all.network.rate [average rate(KB/s) on network transfer]
vizserver.all.network.time [average time(ms) spent on writing a frame
to network]
vizserver.all.frames.total [total number of frames per second]
vizserver.all.frames.spoiled [number of spoiled frames per second]
vizserver.session.readback.rate [data rate(KB/s) of readback per
session]
vizserver.session.readback.time [time(ms) spent on readback at each
frame per session]
vizserver.session.compress.rate [data rate(KB/s) on compression per
session]
vizserver.session.compress.time [time(ms) spent on compressing a frame
per session]
vizserver.session.network.rate [data rate(KB/s) on network transfer per
session]
vizserver.session.network.time [time(ms) spent on writing a frame to
network per session]
vizserver.session.frames.total [total number of frames per session]
vizserver.session.frames.spoiled [number of spoiled frames per session]
vizserver.pipe.readback.rate [data rate(KB/s) of readback per pipe]
vizserver.pipe.readback.time [time(ms) spent on readback at each frame
per pipe]
vizserver.pipe.frames.total [total number of frames per pipe]
vizserver.pipe.frames.spoiled [number of spoiled frames per pipe]
```

OpenGL Vizserver PMDA has 23 metrics as shown in the above example. They specify the characteristics of each stage of the OpenGL Vizserver pipeline. The `vizserver.*.time` metrics represent the average time spent processing a frame in each stage and the `vizserver.*.rate` metrics represent the average number of kilobytes per second that get into each stage of the pipeline.

The `vizserver.all.*` metrics represent the overall data, the `vizserver.session.*` metrics represent the data per each session, and the `vizserver.pipe.*` metrics represent the data per each graphics pipe.

To view a detailed description for each of the performance metrics, use the `pminfo -T` *metric* command.

The following examples show the number of active sessions and the average data rate and time of the image readback stage in the pipeline per session.

```
$ pminfo -ft vizserver.nsession
vizserver.nsession [number of ongoing sessions]
      value 3
$ pminfo -ft vizserver.session.readback
vizserver.session.readback.rate [data rate(KB/s) of readback per
session]
      inst [0 or "yolee:"] value 12836.378
      inst [1 or "guest:"] value 11454.321
      inst [2 or "joch:"] value 12372.332
vizserver.session.readback.time [time(ms) spent on readback at each
frame per session]
      inst [0 or "yolee:"] value 12.499605
      inst [1 or "guest:"] value 13.661285
      inst [2 or "joch:"] value 14.156073
```

The `pmchart` tool shows the performance metrics against time. It displays the selected metrics in a chart. You can select **metrics** from the **New Plot** option from the **File** menu or **predefined view** from the **Open View** option from the **File** menu. A predefined view for OpenGL Vizserver is the `/var/pcp/config/pmchart/Vizserver` file.

For more information about how to use `pmchart`, see the *Performance Co-Pilot User's and Administrator's Guide*. Figure 3-3 shows an example of using `pmchart` with OpenGL Vizserver PMDA.

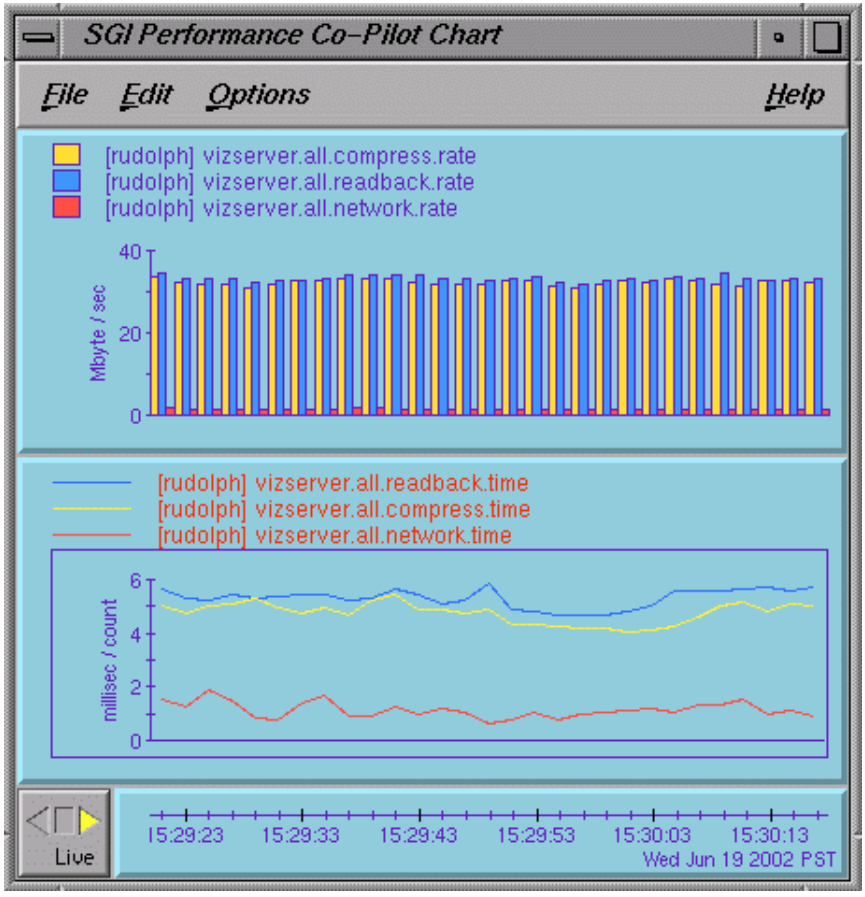

Figure 3-3 `pmchart` Using OpenGL Vizserver PMDA

Note: The `pmchart` tool is in the `pcp.sw.monitor` package subsystem. This tool is not available if you have only `pcp_eoe.sw` installed on your system.

The `pmval` command provides a text-based display of the values for one or more instances of a selected performance metric. For example, the following command shows the values of the performance metric `vizserver.session.readback.rate` at a 1 second interval.

```
$ pmval vizserver.session.readback.rate
metric:    vizserver.session.readback.rate
host:      localhost
semantics: instantaneous value
units:     Kbyte / sec
samples:   all
interval:  1.00 sec

       guest:        yolee:         joch:
       1.181E+04     1.597E+04     1.138E+04
       1.137E+04     1.588E+04     1.128E+04
       1.236E+04     1.625E+04     1.182E+04
       1.059E+04     1.523E+04     1.315E+04
       1.400E+04     1.449E+04     1.119E+04
       1.095E+04     1.301E+04     1.208E+04
       1.412E+04     1.542E+04     1.161E+04
       1.148E+04     1.593E+04     1.128E+04
```

vsmonitor

The `vsmonitor` tool is text-based and is used to display various performance metrics all together in a current shell window. It reports the current values of all the metrics from the OpenGL Vizserver server every 5 seconds (by default). The period can be changed by using the `-f` option.

The `vsmonitor` tool categorizes the performance metrics into three sections: server, sessions, and pipes. A server section displays the average, or sum, of the performance data of currently running sessions. A sessions section shows activity per session. And a pipes section shows activity per pipe.

The `read/conv/comp/output` times in a server section are averaged values of the currently running sessions and `read/conv/comp/output` rate and `total/spoil` frame are the sum of the sessions data.

In Figure 3-4, it shows that the OpenGL Vizserver server rudolph has three sessions running and each session uses one pipe. Session `yolee` shows the compression rate as 0 bytes and the number of spoiled frames per second at 17.0. This implies the session is currently running with no compression and spoiling turned on and a lot of frames are spoiled. Session `joch` shows the number of spoiled frame as 0, which means the spoiling is turned off.

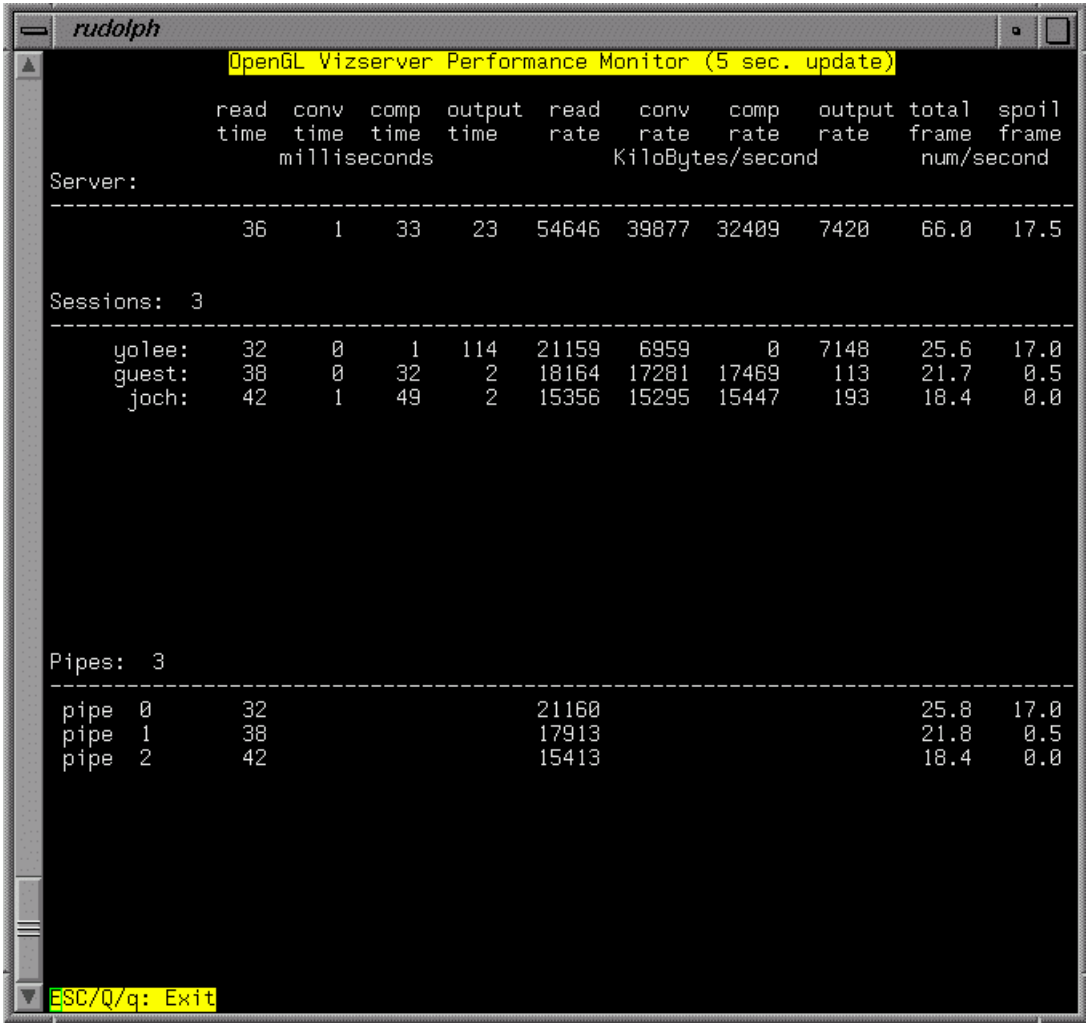

Figure 3-4 `vsmonitor`

See the `vsmonitor`(1m) man page for more detail description about each performance metric.

Estimating the Network Bandwidth Required by OpenGL Vizserver

In most cases, the network bandwidth is a major factor of bottlenecks in the OpenGL Vizserver pipeline. To alleviate this problem, OpenGL Vizserver provides several compression types. However, there is a minimum network bandwidth size needed to use OpenGL Vizserver with reasonably good performance. The application window size and compression rate are also key factors to define the data size in a network transfer.

This section discusses the formula for figuring out how many frames per second are on a given network bandwidth and the formula for estimating how large of a network bandwidth is necessary to get a certain number of frames per second. It is assumed that an entire image is changed in each frame. This assumption is true in a worst case scenario: since the compression modules shipped with OpenGL Vizserver are based on frame-differences, the bandwidth requirement is expected to be lower in practice.

Calculating Frames Per Second on a Given Network Bandwidth

A frame is the period of time that it takes to update the display with the new image. For example, a frame rate of 60 Hz means that the display is updated 60 times per second.

In OpenGL Vizserver, a frame size is represented as the size of an application's window, which is handled as an image. An image size is represented as the combination of width, height, and the depth of pixels, as follows:

*frame_size = width * height * depth_of_pixel*

If you set the variables of the frame size, width, height, and depth of pixel as follows:

- *frame_size = f*
- *width = w*
- *height = h*
- *depth_of_pixel = d*

You get the following formula, because *d* is usually 24 bits (3 bytes):

f = `d * w * h = 3 * w * h`

Since the network bandwidth between two systems is usually given in units of MB or KB, suppose that the network bandwidth is *n* KB.

n `KB = 1024 *` *n* `(KB = 1024)`

Then, the number of frames on a given network bandwidth is the result of network bandwidth divided by a frame size. In other words,

`(1024 *` *n*`) /` *f* `= (1024 *` *n*`) / (3 *` *w* `*` *h*`)`

For c:1 compression, you can get *c* times more compression than no compression. The number of frames per second on a given network bandwith with c:1 compression is:

`(1024 *` *n* `*` *c*`) / (3 *` *w* `*` *h*`)`

For no compression, *c* is 1.

For example, assume that the network bandwidth, image size, and compression ratio are as follows:

network bandwidth: 10,340 KB/sec
image size: 1280 x 1024 pixels
4:1 CCC compression

The number of frames per second on a given network bandwidth that OpenGL Vizserver can get is calculated as follows:

```
(1024 * 10340 * 4) / (3 * 1280 * 1024) ~= 10.77
```

In this example, you can get roughly 10 frames per second.

Calculating Network Bandwidth Necessary for *k* Frames Per Second

To estimate the network bandwidth (*n*KB) required by OpenGL Vizserver when you want to get *k* number of frames per second, use the formula from the previous section as follows:

k `= (1024 *` *n* `*` *c*`) / (3 *` *w* `*` *h*`)`

`1024 *` *n* `= (3 *` *w* `*` *h* `*` *k*`) /` *c*

That is,

n `KB = (3 *` *w* `*` *h* `*` *k*`) /` *c*

For example, if the image size and number of frames per second are given as below, calculate how large a network bandwidth is required.

image size : 512 x 512 pixels
no compression
at least 10 frames per second

The amount of network bandwidth necessary for *k* frames per second is calculated as follows:

```
1024 * n = (3 * 512 * 512 * 10) / 1
n = 7680.
```

This shows that you need to have at least 7,680 KB/s of network bandwidth.

Optimizing for High-Latency Networks

In order to keep the round-trip latency viewed by the client (for example, the time between moving the mouse and seeing the result on the client side), OpenGL Vizserver uses a frame acknowledgment mechanism at the application level. A client-side environment variable, `VS_LATENCY`, determines how many frames are allowed to be "in transit" from the session to the client over the network at the same time. This means that if the session has sent `VS_LATENCY` frames to the client but has not received an acknowledgment from the client that the first frame was received, the session does not send the next frame until the session gets this first acknowledgment from the client.

By default, `VS_LATENCY` is set to `2`. This value is optimal for very low-latency networks, such as an Ethernet-based local area network (LAN). If the network's latency is high, as in a wide area network (WAN), this default value will cause OpenGL Vizserver to utilize the network's bandwidth poorly and to achieve a sub-optimal frame rate.

For example, consider the following case:

Network bandwidth	5 Mbyte/sec
Network round-trip latency	0.3 sec
Typical OpenGL Vizserver frame size	0.5 Mbyte

In this case, the best frame rate possible is 10 frames/sec:

```
5/0.5 = 10 frames/sec
```

However, in reality, the default VS_LATENCY value of 2 will reduce the frame rate to approximately 6 frames/sec:

```
2/0.3 ~= 6 frames/sec
```

Setting VS_LATENCY to 3 will increase the frame rate to 10 frames/sec, thus, better utilizing the network:

```
3/0.3 = 10 frames/sec
```

In general, the optimal VS_LATENCY is one that does the following:

- Fully utilizes the network bandwidth and the client's capabilities (the least of the two).
- Makes sure it does not send frames too fast for the client to process, thus creating high latencies.

A little algebra yields the following formula for calculating the optimal VS_LATENCY value:

VS_LATENCY = min{ ceil[(*L*+*S*/*B*)***C*], ceil[1+*L***B*/*S*] }

The variable items in the formula are defined as follows:

L	Round-trip latency ("ping time" in seconds)
S	Frame size (bytes)
B	Network bandwidth (bytes/seconds)
C	Client processing speed (frames/second)

The following are some typical network environments:

- A 100-Mbit LAN

  There is practically no latency (*L* = 0), and the optimal VS_LATENCY value is 1. The default value of 2 might add one frame of latency in this case, but the default value provides more robustness with respect to network bandwidth and latency fluctuations.

- A WAN with medium latency, high network bandwidth, and deep compression

  In this case, *S*/*B* is very small, and the optimal VS_LATENCY value is **ceil(***L***C***)**. With 300 ms latency, and a client capable of drawing 15 frames/sec, the optimal VS_LATENCY value is 5.

- A WAN with very high latency (*L* = 1000 ms), high network bandwidth, deep compression

  The default value of `2` will get you 2 frames/sec, regardless of your bandwidth. If you have high bandwidth (45 Mbit/sec) and a fast client capable of 30 frames/sec, setting `VS_LATENCY` to `31` will achieve the full 30 frames/sec. Note that there is an inherent 30 frames of latency in such a network, which might make it very hard to use for OpenGL Vizserver sessions.

To summarize, the default value of `VS_LATENCY` is optimized for LANs. If you run a session over a high-latency network, adjusting the `VS_LATENCY` value (using the formula just described) might provide considerably higher frame rates.

Troubleshooting and Known Problems

In most cases, when there are problems, error messages are shown in the console window, application windows, or log files.

This chapter describes how to look at log files to track down the errors and what the known problems are and how to resolve them. The following topics are covered:

Looking at Log Files

OpenGL Vizserver uses the following log files:

If you have problems running OpenGL Vizserver, first look at the server and session log files and then the system log file. If by looking at the server log file you find out that the X server failed to start on your Onyx4 system, looking at its XFree86 log file can give you a hint as to why the X server failed.

The accounting files can also be useful in establishing a chronology of user events. The following subsections describe each type of log file.

Server Log File

The OpenGL Vizserver server manager writes its status messages in its server log file:

```
/var/vizserver/logs/vsserver.log
```

The server log contains the following entries:

- Server mananger start/stop notice
- Licenses that were found
- Licenses that checked out/in
- Clients logged in/out
- Number of pipes found
- Number of pipes used by XDM
- Number of pipes marked for OpenGL Vizserver use
- X server start notification

- Session start success/failure
- Error message from the server

Note: After each run, the server log is overwritten.

Session Log File

The OpenGL Vizserver session manager writes its status messages in its session log file:

`/var/vizserver/logs/vssession.<`*username*`>.log`

The <*username*> variable specifies the master user's login name for the session.

The session log contains the following entries:

- Session manager start/stop notice
- Session name, server display, and master username
- Client join/leave
- Compressor change
- Spoiling change
- Control passing change
- Error messages from the session

Note: After each run, the session log is overwritten.

System Log File

The OpenGL Vizserver server processes also write critical error messages in the system log file:

`/var/adm/SYSLOG`

The processes use the following tags for the system log entries:

`vsserver` Entry posted by the OpenGL Vizserver **server** manager

`vssession` Entry posted by the OpenGL Vizserver **session** manager

If you are having problems while using the PAM authentication module with OpenGL Vizserver, the `/var/adm/SYSLOG` file might include relevant information about it. You should look for entries having *module_name*[*pid*] in them—for example, `pam_rhosts_auth[2112]:`, where the module used is `pam_rhosts_auth.so` and the PID of `vsserver` is `2112`.

Since there are so many other processes on your system that write their messages to the `/var/adm/SYSLOG` file, look carefully at lines between user connects to the OpenGL Vizserver server and user disconnects from the OpenGL Vizserver server. Check that the session was started and exited normally and look at the messsages related to graphics processes between the user connect and user disconnect messages.

XFree86 Log File

When the OpenGL Vizserver server host is an Onyx4 system, the X server that is used on the system is XFree86. When XFree86 is run, it generates a very detailed log file, `/var/log/XFree86.`*x*`.log`, where *x* is the X server's number (for example, if the X server is `:32`, the log file is `/var/log/XFree86.32.log`).

Based on the file set in the server configuration parameter `Vizserver*BaseXF86Config` (see "The /var/vizserver/config File" on page 22), OpenGL Vizserver generates temporary XFree86 configuration files to be used by the X servers it starts. In the XFree86 log file, you can also look for the line starting with the following string:

```
Using config file:
```

This line will identify which XFree 86 configuration file was used when starting this X server.

Something might be incorrect or inadequate for OpenGL Vizserver's needs in this file especially if you have customized this file. If you have customized the file, you might resolve the issue by creating a new default XFree86 configuration file using `/etc/X11/gen-XF86Config` and using it as the base XFree86 configuration file for OpenGL Vizserver. Do not forget to back up your customized XFree86 configuration file.

Accounting Log Files

Looking at accounting log files can be useful to determine actual time, user, and session type when an error happened.

OpenGL Vizserver records a client login and logout and a session start and stop into an accounting log file (usually `/var/vizserver/acct`). This file can be viewed using `vsacct`. See the `vsacct`(1m) man page for more details.

```
$ vsacct /var/vizserver/acct
```

Each OpenGL Vizserver session is also logged to the wtmpx database of the system (typically `/var/adm/wtmpx`), for use with `utmpx` based utilities, such as `last`.

OpenGL Vizserver sessions appear in the file as the device `vss`*pipe#*, where *pipe#* is the graphics pipe number used by a session. If the session uses more than one graphics pipe, a line per each graphics pipe is used.

Since the `last` command also shows other records in `/var/adm/wtmpx`, use it with the `grep` command to extract the information related only to OpenGL Vizserver sessions. (Actual results on your system will be different.)

```
$ last | grep vss
yolee    vss1    130.62.55.27      Tue Aug 13 16:39 - 16:49  (00:09)
guest    vss0    130.62.53.103     Tue Aug 13 16:33 - 16:43  (00:10)
guest    vss0    130.62.53.103     Tue Aug 13 15:27 - 15:56  (00:28)
root     vss1    130.62.55.66      Mon Aug 12 12:46 - 13:07  (00:20)
yolee    vss0    130.62.55.27      Mon Aug 12 13:42 - 13:53  (00:10)
yolee    vss2    130.62.55.27      Mon Aug 12 13:42 - 13:53  (00:10)
root     vss0    130.62.52.83      Mon Aug 12 12:46 - 13:07  (00:20)
```

See the `last`(1) and `utmpx`(4) man pages for more details.

Shared Memory Input Queue (shmiq) Problem

If OpenGL Vizserver cannot use all of the available graphics pipes in your system and your system's `SYSLOG` shows something similar to the following, it is a *shmiq* problem.

```
Dec  4 15:46:36 5B:ontario vizserver: Failed to open shmiq control
device.: No such file or directory
Dec  4 15:46:36 3D:ontario Xsgi35[17597]:
Dec  4 15:46:36 5B:ontario vizserver: Xsgi35[17597]:
```

```
Dec  4 15:46:36 2D:ontario Xsgi35[17597]: Fatal server error:
Dec  4 15:46:36 5B:ontario vizserver: Xsgi35[17597]: Fatal server
error:
Dec  4 15:46:36 2D:ontario Xsgi35[17597]: Error Starting SHMIQ I/O!
Dec  4 15:46:36 5B:ontario vizserver: Xsgi35[17597]: Error Starting
SHMIQ I/O!
Dec  4 15:46:36 2D:ontario Xsgi35[17597]:
Dec  4 15:46:36 5B:ontario vizserver: Xsgi35[17597]:
```

What is shmiq?

A shmiq (pronounced *shmick*) is a fast way of receiving input device events by eliminating the operating system overhead to receive data from input devices. Instead of reading the input devices through UNIX file descriptors, the kernel deposits input events directly into a region of the X server's address space, organized as a ring buffer.

Why Does This Cause a Problem?

Associated with the shmiq driver, a character device called qcntl is needed for the X server (`Xsgi`). The qcntl device allows `Xsgi` to process character input from the shmiq driver. To use multiple X servers in a system, you need at least the same number of `/dev/qcntl` nodes as that of `Xsgi` to be used. For example, if your system has only qcntl0 and qcntl1 nodes, you can have at most two `Xsgi` servers running on your system.

As of IRIX 6.5, the systems with graphics capabilities are preconfigured with 9 shmiq drivers, 2 input directories (`/dev/input0`, `/dev/input1`), and 8 qcntl nodes (`/dev/qcntl0`, `/dev/qcntl1`, `...`, `/dev/qcntl7`): therefore, usually you do not have to worry about these values. However, the preconfigured values are sometimes wiped out when the system is rebooted.

How To Resolve It

Check how many qcntl nodes are in the `/dev` directory and create an additional number of qcntl character devices by using `mknod` as follows. Create one qcntl node for each pipe in your configuration.

```
# mknod qcntl2 c 55 2
# mknod qcntl3 c 55 3
...
# mknod qcntl7 c 55 7
```

The default `/var/sysgen/master.d/shmiq` file defines `NSHMIQS` as 9, so you can have a maximum of 8 qcntl nodes.

Note: If your system has 16 pipes, you can change `NSHMIQS` to 17 and make 16 qcntl nodes. In that case, you need to create a new kernel (`autoconfig -fv`) because you modified the `/var/sysgen/master.d/shmiq` file.

No Appearance of OpenGL Vizserver Console Window in Windows 2000

As a normal procedure, after starting a session, the **OpenGL Vizserver Session Control** window is shown first and then the **OpenGL Vizserver Console** window later. However, if the O**penGL Vizserver Console** window is not shown after the session control window has appeared, make sure that your Windows 2000 system has service pack 2 or later installed.

Cleaning Up Shared Memory

After running an OpenGL Vizserver sessions many times continuously, if the performance of OpenGL Vizserver shows the slowdown considerably, check the shared memory in an OpenGL Vizserver server system. This can be done by using an `ipcs` command.

Usually, an OpenGL Vizserver session removes the shared memory for its use when the session is exited. However, there might be cases where the shared memory is not deleted when the session or applications do not exit normally. Then it is stacked up and occupied as a long list of active shared memory. This might cause the problem in OpenGL Vizserver performance. The shared memory can be removed by using an `ipcrm` command. See the `ipcs`(1) and `ipcrm`(1) man pages for more details.

An OpenGL Vizserver session uses a message queue and a shared memory segment with keys 0x12340000 to 0x1234FFFF. These resources are necessary for the session and applications to communicate. Ensure that they are not deleted while the session is running.

Using Window Managers Other Than 4Dwm

In an OpenGL Vizserver client using a window manager other than `4Dwm`, the application windows running under the OpenGL Vizserver session accept user inputs, such as key press/release and mouse button press/release, which are only conformant to `4Dwm`.

Application Not Updated

Sometimes an application window is updated on an expose event. When spoiling is on, some of the updates are missed. So it appears as if the window on the client side never got updated. Try to turn spoiling off and you will see the updates. The updated rates are also dependent on your system configuration and network bandwidth.

Applications Masked as a Cross-Hatch Pattern Image

When an application is started by a user who does not have the privilege to run and the user did not start the OpenGL Vizserver session, the application is masked as a cross-hatch pattern image.

For example, an OpenGL Vizserver session is started by a guest user and the user is switched to `root` later in the OpenGL Vizserver console window. Then if `vsconfig` is issued, the application is masked as a cross-hatch pattern image in the console window.

In addition, in a local collaborative session, when the master's other application windows overlap the application windows running from OpenGL Vizserver, nonmaster clients in the session see the cross-hatch pattern in their OpenGL Vizserver application windows.

For example, if the OpenGL Vizserver console window or application windows are covered by other windows on the local master's monitor, the OpenGL Vizserver application window of nonmaster clients displays the overlapped region as a cross-hatch pattern image. This is an expected behavior and it was implemented as a security feature to prevent the local master's private contents from being transmitted to the other clients.

Back-to-Front Rendering

OpenGL is not inheritently frame-based. Therefore, OpenGL Vizserver uses **glFlush()**, **glFinish()**, or **glXSwapBuffers()** calls to trigger a framebuffer readback.

Applications that do the back-to-front rendering and do not make these calls often might get less than optimal frame updates.

Using Customized XDM in Dynamic Pipe Allocation

As mentioned in the section "Dynamic Pipe Allocation" on page 31, OpenGL Vizserver uses `*.vizserver` scripts and the `xdm-config` configuration file in the `/var/X11/xdm` directory by default.

When using a customized XDM with a dynamic pipe allocation scheme, you must inform OpenGL Vizserver server of its location and the configuration files related to the XDM. This involves modifying some of the scripts and the XDM configuration path (XDM Config File) in `vsconfig`. Otherwise, OpenGL Vizserver might have problems allocating pipes even though there are available pipes on the server.