AVS on UNIX WORKSTATIONS INSTALLATION / RELEASE NOTES

Release 5.5 Final (50.86 / 50.88)
November, 1999

Advanced Visual Systems Inc.

Part Number: 330-0120-02 Rev L

NOTICE

Advanced Visual Systems Inc.
300 Fifth Ave.
Waltham, MA 02451

Printed in U.S.A.

CONTENTS

3 AVS 5.5 on UNIX Platforms

4 AVS 5.5 for Compaq Tru64 UNIX

AVS 5.5 RELEASE NOTES

CHAPTER ONE

Release Highlights

This document describes Advanced Visual Systems Inc.'s Application Visualization System (version 5.5) as it runs on UNIX workstations.

AVS5.5 NOTES: As a convenience to the reader, changes made for AVS 5.5 are summarized at the beginning of each chapter in "AVS 5.5 NOTES" blocks.

This release contains a number of operating system updates, platform specific improvements (OpenGL for HP, AVSGraph for Linux, OpenGL stereo extensions and ports to HP, Sun, Alpha), new features and documentation (including an online AVS5 doc set) and bug fixes. The AVS Animator is now unlicensed and included in the standard AVS 5 package; its source code is provided under *$AVS_PATH/examples/animator* **and described further under Chapter 3, "AVS 5.5 on UNIX Platforms."**

This release provides the following enhancements:

Operating System Updates

New native support for IBM AIX 4.3, Compaq Tru64 UNIX 4.0D (formerly Digital Unix), SGI IRIX 6.5 (N32/Mips3) and N64/Mips4), Solaris 2.7 (Solaris 7) as well as continued support for HPUX 10.20. These changes are described in the appropriate chapters below.

AVS 5.5 is also now available for RedHat Linux 6.0 for the PC as a separately purchased product. Please contact your local sales representative if you are interested in this platform.

With AVS 5.5, support is being discontinued for SGI IRIX5.x/O32. For users who require this platform or other older operating systems, please contact Customer Support (support@avs.com) or your sales office to acquire copies of AVS 5.4 or older releases.

OpenGL Graphics Library Support

Support for OpenGL is now provided on all platforms including the HP-UX platform. OpenGL provides improved remote graphics support and takes advantage of newer graphics hardware adapters on a number of systems; it provides a standard cross platform graphics layer for shared enhancements like the expanded stereo support in this release. On several platforms, a pre-OpenGL version of AVS 5 is still provided to aid users in migrating to this new standard and in providing support for older graphics adapters. See Chapter 3, *AVS 5.5 on UNIX platforms* for a general introduction and each platform specific chapter for more information.

Foundation Improvements

A number of product improvements are in this release. These include stereo enhancements and ports, several new example modules, a new Geometry Camera module, and new documentation of existing features. See Chapter 3 for cross platform changes and the individual platform chapters for platform specific issues.

Documentation

Several AVS 5 documentation improvements have been made with this release.

The AVS 5 documentation set is now available online in Adobe PDF files providing the core set of books in a viewable, searchable, and printable format. The set can be installed (**AVS5_DOC** product archive) or read from the CD (*/cdrom/avs5_doc* on most platforms). For more information, see Chapter 2, *AVS 5.5 Installation* for installation options and Chapter 3, *AVS 5.5 on UNIX Platforms* for information on obtaining an Adobe PDF reader for UNIX or Windows platforms.

A new chapter has been added to this release note document that provides an overall summary of how to approach debugging modules in AVS 5.5 and prior releases. This chapter draws together information and techniques from throughout the AVS 5 product and is targeted towards both new and experienced module writers.

Licensing

The AVS Animator and its supporting modules no longer require a separate license but are included in the standard AVS package; all users may now freely use the AVS Animator. The BTF renderer also no longer requires a license.

There have been no changes in basic licensing support - Globetrotter's FlexLM 5.12 is still the version used with AVS 5.5 and existing AVS5.x licenses will continue to work with AVS5.5.

NOTE: Existing AVS 5.5 users do **not** require a new license file to

use AVS 5.5, nor will they need to make any changes to their "license daemon" programs. Users of AVS 5.3 or earlier releases **will** need to reinstall their "license daemon" programs to use FlexLM 5.12. For more information, see the Chapter 2, *AVS 5.5 Installation*.

Organization

These *Installation and Release Notes* contain the following chapters. A printable PostScript version of this document is available in the *relnotes* subdirectory of AVS's home directory; it is now available in PDF as well.

Chapter 2: AVS 5.5 Installation
Describes how to install AVS on your system, and how to upgrade an existing license. For first-time installers, there is an overview of the licensing mechanism and a description of how to obtain a new license file with the correct values for your platform.

Chapter 3: AVS 5.5 on UNIX Platforms
This introductory chapter provides information common or important to all platforms, including how AVS works under X windows and common programming considerations. This chapter also includes a list of general product improvements for the AVS 5.5 release. All users should review this chapter in addition to the specific chapter for their platform.

Each subsequent chapter describes platform specific information for a particular platform including system prerequisites, release notes, and recent improvements and known limitations. You should review the appropriate chapter for your platform before installing on a new machine.

Chapter 4: AVS 5.5 for Compaq Tru64 UNIX (Alpha)

Chapter 5: AVS 5.5 for Hewlett Packard

Chapter 6: AVS 5.5 for IBM

Chapter 7: AVS 5.5 for Silicon Graphics

Chapter 8: AVS 5.5 for Sun Solaris

Chapter 9: Extended Features
Describes a number of extensions made to AVS 5 in previous releases, including the Cool CD, AVS/Voxel, Japanese Help and additional Demonstration networks and scripts.

Chapter 10: Administering Licenses
Describes how to configure, install, and manage the licensing

mechanism. You should read this chapter if you encounter problems installing licensing or wish to customize licensing at your site.

Chapter 11: Debugging in AVS 5 (New in AVS 5.5)
Provides a summary of suggestions and tips on developing and debugging AVS modules; it draws together a number of techniques and features that are available in different areas of the product. Intended for both new and experienced module writers.

AVS 5.5 Media Kit

The AVS 5.5 media kit provided to new users contains two CD's with accompanying manuals.

- The AVS 5.5 CD is provided along with the consolidated release notes covering all current UNIX platforms. Existing AVS 5 users are provided with this media kit as an upgrade from AVS 5.4. Installation of software on this CD is covered in Chapter 2, *Installation*.

- The Cool CD provides a wealth of third party software from the IAC and other sources, including the UCD Builder product and manual. For AVS 5.4 and beyond, the Cool CD has not updated; for the latest public domain modules and information, users are invited to visit the International AVS Center web site at the University of Manchester at *http://www.iavsc.org*. For more information see Chapter 9, *Extended Features.*

License Requirements

AVS is a licensed software product of Advanced Visual Systems Inc. You obtain the license from Advanced Visual Systems Customer Support or your local distributor, not from your platform vendor. AVS will not run until you have obtained a license.

The *AVS 5.5 Installation* chapter contains a section called "Step 6: Install License." This section explains how to obtain the necessary license file that contains the licensing codes that are correct for your installation. It describes both the "new AVS installation" and the "upgrade an existing AVS installation" cases. Read this section *before* you contact Customer Support or your local distributor. It explains what site and machine-dependent information you must supply to us before we will be able to generate your license code. **Note: AVS 5.5 does not require a new license for existing AVS 5.4 users nor does it require that license daemon's be updated.**

The *Administering Licenses* chapter describes how to install the license(s) and customize the licensing mechanism.

Implementation Overview

This product is a full implementation of Advanced Visual Systems Inc.'s Application Visualization System (Release 5.5) on various UNIX workstations.

Features introduced in recent releases are described in the *Extended Features* chapter in these *Installation and Release Notes.*

AVS is fully described in the following manuals that accompany this product in hardcopy and in the PDF archives on the CD:

- *AVS Technical Overview*
- *AVS User's Guide*
- *AVS Tutorial Guide*
- *AVS Module Reference Manual*
- *AVS Developer's Guide*
- *AVS Application's Guide*
- *AVS 5 Update*
- *AVS/Graph User's Guide*

The AVS Animator, now included in the standard AVS 5.5 package without licensing, is documented in its own manual, *Animating AVS Data Visualizations.* Source code is now provided under *$AVS_PATH/examples/animator*; for more information see Chapter 3, "AVS 5.5 on UNIX Platforms."

The *UCD Builder User's Guide* describes the UCD Builder, an unsupported product delivered on the Cool CD. This product has *not* been updated for the AVS5.5 release. For more information see Chapter 9, *Extended Features.*

The Molecule Data Type, the *libchem* library that manipulates it, and the collection of chemistry and example modules that show how to program with *libchem* are documented in the *Chemistry Developer's Guide* manual. This manual is not included in hardcopy with the AVS release but is available in the online PDF archive. It can be requested separately from support@avs.com and will be provided *free of charge.*

Note: A few books were not available in PDF format at release time (*AVSGraph User's Guide, Technical Overview,* and *UCD Builder's Guide*); contact AVS Customer Support for possible future availability of these book files from our web site.

AVS 5.5 INSTALLATION

CHAPTER TWO

Introduction

AVS 5.5 is supplied on a multi-product compact disk (CD-ROM) providing AVS for different hardware platforms along with optional products such as AVS Demos and AVS5 Doc.

AVS5.5 NOTE: The AVS 5 Documentation package is new with AVS 5.5 and can be installed from the AVS5_DOC package or used directly on the CD under the *avs5_doc* directory (see Chapter 3 for more information). No other significant changes in installation have been made from the AVS 5.4 release. If you are already familiar with this process, proceed with mounting the CD, running *install.avs*, and selecting the platform(s) you wish to install. You should not need to change your existing AVS 5.x license or the license daemons installed with AVS 5.4. A new "Quick Start" installation summary has also been added.

Quick Start Instructions

Because it provides so much detail, the installation chapter for AVS 5.5 can seem more intimidating than it needs to be. This is a simplified process that should work for most customers:

- System Prerequisites: AVS 5.5 requires 60-90 Megabytes disk space, 16-32 Megabytes memory, X windows, and OpenGL, which is usually installed on recent workstations. If AVS doesn't initially run in your configuration, read the appropriate platform chapter for more detailed information.

- Installation: Mount the CD according to the booklet instructions (many systems automount on insertion) and run */cdrom/install.avs* (the CD folder name and filename case varies by platform). Indicate which platform you have and the installation directory you want to use (the default is */usr/avs*). Set the **AVS_PATH** environment variable to the installation directory you chose ("setenv AVS_PATH <dir>").

- Licensing:
  - Evaluation customers: demo_license makes licensing easy. Just run *$AVS_PATH/license/demo_license*, give it the password you were given by your sales representative and make a temporary *license.dat* file; type "setenv LM_LICENSE_FILE ‘pwd‘/license.dat" to tell AVS where to find the license file and that’s it.
  - Existing users: No need to change existing licenses for AVS 5.5.
  - New users: If you have not licensed AVS 5 before, you will need to fill out and submit a license request form. This can be done online at *http://help.avs.com/licensing/request.asp*.
  - All users: Use *$AVS_PATH/license/licdiag* to help you get your hostid and change or install licenses. If licensing becomes a major obstacle, contact AVS Customer Support for help and potential alternatives.
- Running: Startup *$AVS_PATH/bin/avs*, select "AVS Applications" and "AVS Demo" then select which demos you would like to run. Running the demos will provide a good test of the installation and also help to familiarize you with AVS 5 modules and features.

If you have problems after installation, read the appropriate platform chapter for your system to check the prerequisites in more detail. If necessary you can easily uninstall AVS 5 ("rm -rf $AVS_PATH"); uninstalling will remove any other data files or changes you have made in the AVS 5 installation area.

Installation Steps

This is the full installation story with all the details.

NOTE: There are two distinct SGI products, both of which are based on IRIX 6.5: "*SGN32*" is based on a 32-bit architecture using the N32 ABI and compiled for Mips3 architectures; and "*SGN64*" is based on a N64 ABI (64-bit) architecture and compiled for Mips4 architectures. For installation purposes, there is no significant difference between these two versions of the SGI OS except where noted; use the *SGI* instructions for both.

There are six basic steps to an AVS installation:

Step 1

Decide where you want to install AVS.

Step 2
Mount the CD-ROM media.

Step 3
Install AVS using the *install.avs* script. This reads the remainder of AVS off the CD-ROM into the specified directory. Optionally install links from *usr* and install the man pages.

Step 4
Install or update the AVS license. This is a multi-step procedure. AVS will not execute without a license (AVS 5.4 users do not require a new license for AVS 5.5 nor do they need to update license daemon's).

Step 5 (Optional)
Run the AVS Demos to confirm installation

Step 6
Inform your users how to set the variables necessary to find AVS when they try to start it.

Remote Installations

The installation script assumes that you are logged in on the system that is connected to CD-ROM drive. There is no provision for "remote" data reads across the network. If the system with the CD-ROM drive can't access the file space where you want to install AVS, then do the following:

- Mount the CD-ROM where you can read it (see Step 2 for more information) and list out the CD contents using *ls* to see what you need.

- Copy the installation script (*INSTALL.AVS*) and the compressed tar archives you need (*<platform>.Z, AVS_DEMO.Z, etc*) to a hard disk area the target machine ("target") can access. For example,

```
rcp /cdrom/hp1020.z target:/tmp
rcp /cdrom/install.avs target:/tmp
```

- Log onto the target machine and run *INSTALL.AVS* from the temporary space you copied it to. You will need to tell it which directory the archives are in, but after that it will work as if it is coming off the CD.

Note: You can also copy installed AVS directories to other systems but beware that there is a circular link, $AVS_PATH/include/avs that must first be removed to avoid an infinite loop. Remove this link before copying and then restore it after installation as follows:

```
cd $AVS_PATH/include
ln -s ../include avs
```

Step 1: Decide Where to Install AVS

AVS can be installed anywhere in a file system hierarchy.

AVS requires different amounts of disk space to install on different platforms, ranging from 50 to 100 megabytes The exact size required will be reported as you do the installation. You can check the amount of free disk space on your system by using the *df*, *df -k*, or *bdf* command (system dependent).

The file system must be mounted and available to each system that will execute AVS. Each system must have a useable file pathname to the AVS directory.

A description of what a user must do to find AVS at startup is described below in "Step 6: Tell Users How to Find AVS", and also in the "$Path—Installing and Finding AVS Anywhere in a Directory Tree" section of the *AVS 5 Update* manual.

By default, AVS 5.5 expects to be found in */usr/avs*, but it can be installed anywhere in a file system hierarchy. If you wish to create special links to place AVS in */usr/avs* or to remove existing links, see "Making and Unmaking Links" in Step 3 below.

You may keep multiple versions of AVS 5 on the same system as long as they are in different directories, either using links or different AVS paths.

Step 2: Mount the CD

The next step is to mount the CD-ROM and access the *install.avs* script. Follow these steps:

1. *login* or *rlogin* to the system that is connected to the CD-ROM drive. On some systems, you will need to be "root" user to mount the CD initially.

2. Insert the CD into the CD-ROM drive.

   If there is already another CD in the drive, you should first follow the unmounting instructions below ("Step 3: Install AVS", last instruction) to properly unmount it.

3. Create the CD mount directory. The default mount directory on most systems is */cdrom*. (You can use a different directory, if necessary.) If */cdrom* does not already exist, create it (as root):

```
mkdir /cdrom
```

Note: Some systems (notably SunOS 5.5 and SGI platforms) will "automount" a CD once it is inserted in the drive, using a standard mount directory. See the table below for more information.

4. As root, *mount* the CD onto the */cdrom* directory. All *mount* commands take a CD-ROM device name, the mount directory (*/cdrom* or your name) and a set of options that vary from platform to platform. Consult your system administrator if you do not know the CD-ROM device name for your system.

Table 2-1. CD Mount Commands on Different Platforms

Platform	**Device / Mount Command**
Compaq Tru64	CD-ROM drive's SCSI ID example: */dev/rz4c* for SCSI ID 4 # mount -r -t cdfs /dev/rz4c /cdrom
HP 9000/7*xx*	Query with SAM utility. CD-ROM drive's SCSI ID example: */dev/dsk/c1t2d0* for SCSI ID 2 or use SAM on some systems, device may be */dev/cdrom* # mount -r -F cdfs */dev/dsk/c1t2d0* /cdrom
IBM RS/6000	Query with: /etc/lsdev -C -c cdrom -H example: cd0 Available 00-08-00-10 CD-ROM Drive means */dev/cd0* # mount -r -v cdrfs */dev/cd0* /cdrom
SGI	See note below. CD-ROM drive's SCSI ID example: */dev/scsi/sc0d4l0* for SCSI ID 4 Note: "*l*" is a lowercase L, not "one" # mount -r -t iso9660 /dev/scsi/sc0d4l0 /cdrom
Sun SunOS 5.x	See note below. CD-ROM drive's SCSI ID example: */dev/dsk/c0t6d0s0* for SCSI ID 0 # mount -r -f hsfs */dev/dsk/c0t6d0s0* /cdrom

SGI Note: IRIX systems usually come with a CD-ROM device already "mounted" at */CDROM*. Any CD inserted into this drive is automatically available. If this is how your system is configured, you do not need to issue a *mount* command.

SunOS 5.x Note: In the default configuration of a SunOS 5.x system, the *vold* system daemon automatically mounts the CD as */cdrom/<cd-name>* where *<cd-name>* is the CD title, such as "avs". You need to specify this full directory (i.e. */cdrom/avs*) when you run the

INSTALL.AVS script and tell it where the CD is mounted. If the *vold* daemon is not running, mount the CD as indicated in the table above.

General Note: HP 9000/7xx and Compaq Tru64 UNIX platforms may not recognize the ISO 9660 format used in the AVS5 CD; the ISO drivers are an optional CD format selected during operating system installation. See your network administrator if you have difficulties.

5. At this point, the *install.avs* script should exist in the */cdrom* directory (in some cases you may see it as *INSTALL.AVS*. Verify with the *ls -l /cdrom* command as appropriate.

System administrators can examine the *install.avs* script before using it to verify that it is acceptable to their system's administrative restrictions.

Once the CD is *mount*ed, you do not need to be root. (However, you will again need to be root to *umount* the CD at the end of the installation.) Systems which automount the CD do not require that you be root during mounting or dismounting.

Step 3: Install AVS

The installation script uses the *tar* command to read AVS off of the CD. In almost all cases, *tar* will keep the AVS file protections as they are on the media. The script will catch and warn you when this may not be true.

During installation *tar* will assign the files the user and group ownership of the user executing the installation script. Be conscious of this and execute the installation script as the entity—either root or otherwise—suitable for the pattern of use at your installation.

1. Run the *install.avs* installation script. Depending on your system, this may appear as all uppercase or all lowercase. Type the case that you see.

```
     /cdrom/INSTALL.AVS

or   /cdrom/avs/install.avs   (SunOS 5 with vold running)

or   /CDROM/install.avs       (SGI)
```

2. If your default file protection *umask* will alter the file protections on the media (for example, **HP** platforms), then you will see this message:

```
Installation on this platform is affected by the 'umask' value
which will change the file protections of the installed files
possibly limiting public and group access.

If you want the original file protections to be setup
you should quit now and change the 'umask' setting as follows:

    umask 0
    install.avs
    umask <original umask value>

Press return to continue or type quit:
```

You should do as the message says: *quit* out of the installation script, reset the *umask* as shown, rerun the *install.avs* script, then restore the original *umask.*

3. **Product Selection:**

   You will see a table of possible product choices.

```
This CD-ROM provides the following AVS products:

Name          Platform    Product         Version       Size in Megabytes
AVS_JHELP     all         AVS_JHELP       5.00 (1.06)   3  (3072 Kbytes)
DAO           DAO         AVS             5.5 (50.88)   71  (88064 Kbytes)
HP1020        hp1020      AVS             5.5 (50.86)   70  (59392 Kbytes)
SGN32         sgN32       AVS             5.5 (50.86)   87  (70656 Kbytes)
SGN64         sgN64       AVS             5.5 (50.88)   90  (74752 Kbytes)
S7            SunOS5      AVS             5.5 (50.86)   76  (61440 Kbytes)
IBM           ibm         AVS             5.5 (50.86)   81  (65536 Kbytes)
DEMOS         all         AVS_DEMOS       5.5 (1.4)     27  (25600 Kbytes)
AVS5_DOC      all         AVS5_DOC        5.5 (1.0)     40  (40000 Kbytes)

Please enter the Name of the product to install [default HP1020] or type quit:
```

   Enter the name of the product you want to install, for example:

```
IBM
```

   The script confirms the selection with a message like the following:

```
Selecting IBM
```

4. The script asks to confirm the CD-ROM mount directory:

```
The default CD-ROM directory is /cdrom
If this is OK, hit return, otherwise enter the appropriate directory:
```

Note that the default is */cdrom*. This is different from the */CDROM* that you need on most SGI systems, and the */cdrom/avs* that you need on SunOS 5.x systems running the *vold* daemon.

Either press return or enter a different CD-ROM directory.

5. **Select Installation Directory**

   The script prompts for an installation directory. The default ("[/u2]:" for the example) is the current directory. **Note:** the script creates the *avs* or *relnotes* directory automatically. Pressing return at this point will create */u2/avs*.

   ```
   Enter directory in which to install AVS
   [/u2]:
   ```

   Since you've already changed to the directory in which you want to install AVS, press return.

6. If this directory contained an existing copy of a directory called *avs*, you will see the following:

   ```
   There is an existing directory called /u2/avs.
   Move it now or it will be removed.
   [Hit return to continue]:
   ```

   If you do not want the existing copy deleted, then use another window to move it or type Control-C to terminate the installation.

7. Next you will see:

   ```
   mkdir /u2/avs; cd /u2/avs
   ```

   followed by a message like the following. The exact format and size estimate varies from platform to platform.

```
Installing AVS in:  /u2/avs

AVS requires approximately:  60 Megabytes (61440 Kbytes) of data
df -k  yields:
File System            kbytes    used   avail capacity  Mounted on
/dev/sd0a                7735    5052    1909    73%    /
/dev/sd0g              151399  126178   10081    93%    /usr
/dev/sd1c              299621  185600   84058    69%    /u2
/dev/sd3c              189534  126055   44525    74%    /home2
```

```
Ensure that /u2 has enough space then hit return or type quit:
```

Note that the actual Kbyte size may vary from what is listed here.

8. Hit return and the installation will begin.

Note: At the end of the installation the script tells you that you may optionally run "install.avs links to setup links from /usr". Read the discussion under "Making and Unmaking Links" below before deciding to do this. The links from */usr* are no longer required.

```
Reading AVS from media
Reading IBM archive
tar -xvof /dev/rmt/0m
x .version, 33 bytes, 1 blocks
x ./bin/avs_dbx, 2708 bytes, 6 blocks
x ./bin/avs, 2589380 bytes, 5058 blocks
  .
  .
x ./license/error_hints, 4852 bytes, 10 blocks
122978 blocks
Read from media successfully
Installation of AVS is complete.
You may optionally run "./install.avs links" to setup links from /usr.
```

Note that the actual block count may vary from what is listed here.

Again, read the discussion under "Making and Unmaking Links" before deciding to make links from */usr*. These links are no longer necessary.

9. **Unmounting the CD**

If you mounted the CD as root, you will need to unmount as root also. This is usually done with the *umount* command with either the directory name or device name as argument:

```
# umount /cdrom
```

SGI Note: On SGI workstations, you should type the following instead. It both "unmounts" and ejects the CD. You do not have to be root to type this command.

```
# /usr/sbin/eject
```

SunOS 5 Note: On SunOS 5 workstations running the *vold*

utility, you should type the following instead. It both "unmounts" and ejects the CD. You do not have to be root to type this command.

```
% /usr/5bin/eject
```

Install man page

The *avs* man page is supplied in two forms:

- an *avs.txt* pre-formatted ASCII text file. This file can be copied to one of the pre-formatted */usr/man/cat* man page directories and renamed appropriately (for example: */usr/man/cat1/avs.1*).

- an *avs.6 nroff/troff* source file. This file can be copied to one of the unformatted */usr/man/man* man page directories and renamed appropriately (for example: */usr/man/man1/avs.1*).

Both man page files are located in the installation directory *<installdir>/avs/runtime/help/modules*. Copy the file, in whichever form is suitable for your system, to its appropriate *man* page directory. See your platform's documentation for more information on how man pages are handled on your system.

You may need to manually update the *whatis* database with the one-line description at the top of the man page to make it findable with the *man -k* command. You may need to do this on both a server, and any standalone workstations.

Making and Unmaking /usr Links (Optional)

Links from */usr* to the AVS home directory are optional. Instead, most AVS users specify AVS's location with either the **AVS_PATH** environment variable, **Path** .avsrc file option, or **-path** command line option (see *Step 6* below for more details.)

Some users may wish to continue to use links if they are already using them or to simplify access by local users. The *install.avs* script can make ("links") and unmake ("rm_links") the AVS links which include */usr/avs*, */usr/bin/avs_dbx*, */usr/bin/avs*, and */usr/include/avs*.

Type *install.avs usage* for more information. Note that you will probably need to be "root" user to make or break links in the */usr* directory.

```
Usage:  install.avs [<Flag>]

The install.avs script is used to install AVS and other software products from
CD-ROM media. With no arguments, it will present the available
products and assist in installing them into a specified target area.
```

```
In addition, AVS may optionally install or remove soft links used to help
locate key AVS components from /usr directories (links or rm_links).

Flags:
   links                | Create all standard avs links in /usr
   rm_links             | Remove all standard avs links in /usr
   usage                | Print out this usage message
```

One disadvantage of using */usr/avs* links is that it complicates uninstalling AVS. If there are no extraneous links, then you can uninstall AVS just by removing its main directory:

```
rm -rf $AVS_PATH
```

Step 4: Install license

Your license installation procedures depend on whether you are installing licensing for the first time, upgrading an existing license, or using a short-term demonstration license. This section gives specific instructions on what you must do in each of these cases. This section also includes a brief overview of the Flexible License Manager (FLEXlm) licensing system that is used to administer licensing for AVS. More detailed information about FLEXlm can be found in the *Administering Licenses* chapter. **Note: Existing AVS 5.4 users do not need a new AVS 5.5 license nor do they need to reinstall and restart the license daemons.** If licensing becomes a serious obstacle in your access to AVS 5, please contact AVS Customer Support for potential alternatives.

AVS 5.5 Note: SGI Users: In IRIX 6.5, SGI has begun using FlexLM to license its compilers, storing the license file in */var/flexlm*. If you use the LM_LICENSE_FILE environment variable to set the license file name, you may end up ignoring this SGI license and the compilers will not be licensed. See "Known Problems" in the SGI chapter for more information on this problem and how to work around it.

If you are installing a new license:

If you are a new AVS user installing a license for the first time, you need to install the FLEXlm licensing mechanism from scratch. To do so, you must do the following:

1. Read "Before you begin" below to determine whether your site has any unusual licensing requirements.

2. Find out what kind of AVS license you have purchased (a node-locked or floating license). Installation for all of these licenses are identical in most cases, but you need to know what you have

purchased in case of differences. If you want to know more about the different kinds of licenses, see Chapter 10, *Administering Licenses*, "How licensing works."

3. Select the host machine in your network that is to be the licensing server. FLEXlm works on the client/server model so you must decide where the licensing server processes will run. See "Selecting your host server" below.

4. Make arrangements with Advanced Visual Systems Inc. to obtain a *license.dat* file. See "Obtaining a license" below.

5. Install the *license.dat* file on your system. Follow the instructions in "Installing a new *license.dat* file" below.

6. Run the *licdiag* utility to verify that licensing is installed properly. See "Running licdiag to verify your licensing installation" below.

7. Install and start the FLEXlm-based licensing software on your system. See "Installing the FLEXlm software daemons" below.

If you are upgrading an existing license:

If you are upgrading to AVS 5.5 from AVS 5.4, you already have the FLEXlm licensing mechanism installed on your system and will **NOT** need to update it. You will also **NOT** need a new license for AVS 5.5 if you already have a valid AVS 5.0 or higher license.

If you need to install an AVS 5.5 license, you must:

1. Read "Before you begin" to determine whether your site has any unusual licensing requirements.

2. Make arrangements with Advanced Visual Systems Inc. to obtain a new *license.dat* file only if you need to transfer an existing license. See "Obtaining a license" below for instructions on how to get this information.

3. Replace or modify your existing *license.dat* file with the information in the new *license.dat* file. Follow the instructions in "Upgrading an existing *license.dat* file" below.

4. Run the *licdiag* utility to verify that licensing is installed properly and to make sure you are using the correct version of FLEXlm. You must be using FLEXlm 5.12 or higher. See "Running *licdiag* to verify your licensing installation" below.

5. If necessary, install and start the FLEXlm-based licensing software on your system. See "Installing the FLEXlm software daemons."

If you are using a demonstration license:

If you are installing AVS 5.5 for only a short time - if, for example, you are evaluating AVS on your system - you only need to install a short-term demonstration license. Follow the instructions in "Installing a demonstration license" below.

Before you begin

In most cases, installing and using AVS licenses is straightforward and transparent to the end-user of AVS. There are, however, certain situations that require additional attention - either when you install your licenses or as part of on-going license administration. You should be aware of the following special cases before you install your licenses.

- FLEXlm is used by other software vendors on your system

The FLEXlm licensing system used to administer AVS licenses may already be in use on your system to administer licenses for other products. On Sun networks, for example, use of the SunSoft compiler(s) is controlled by FLEXlm. If other software on your system is already being administered by FLEXlm, see *Administering Licenses*, "Multiple vendors using FLEXlm" for information on how to set up your licensing.

- You want to customize how licenses are allocated on your system

If you are responsible for license administration for your system and you would like to customize the way licenses get allocated, you must be familiar with how licensing works; specifically, you need to understand how the *license.dat* file is used and customized. See all of the overview information in *Administering Licenses*, especially the discussion in "*license.dat* control options."

How licensing works

This section gives a very brief sketch of how the FLEXlm licensing system works. If you want a more detailed description of FLEXlm, see *Administering Licenses.*"

AVS and its associated applications are licensed products of Advanced Visual Systems Inc. Access to AVS is controlled by the Flexible License Manager (FLEXlm) network-wide floating licensing system provided by Globetrotter Software. The FLEXlm system, which is supplied with the AVS product, consists of licensing software, several utility programs that support the licensing mechanism, and associated documentation.

IMPORTANT NOTE: *licdiag* will help you examine and administer your licensing setup. This utility, found in *$AVS_PATH/license*, is designed to help make license installation and administration easier. *licdiag* can be used for most licensing procedures; it is described in detail in *Administering Licenses*, "licdiag".

The license.dat file

The FLEXlm licensing mechanism uses the information in a file named *license.dat* to administer licenses. The *license.dat* file resides in either the */usr/local/flexlm/licenses* directory (its default location) or at the location defined by the **LM_LICENSE_FILE** environment variable. The contents of the file varies somewhat, depending on the kind of license you have purchased; here is a sample of what a *license.dat* file might look like:

```
SERVER menelaus 67069f38 1700
DAEMON avs_lmd /usr/avs/lic/avs_lmd
FEATURE AVS avs_lmd 5.000 1-jan-00 4 FB4FB43AFD04E643099C "" 67069f38
```

The contents of this file determine the assigning of licenses; therefore, it is this file that you install or modify when you add a new license or upgrade an existing one. When you obtain a license from Advanced Visual Systems, you will receive a *license.dat* file that you must then install on your system. **Note:** The "1-jan-00" date indicates "permanent", not "2000".

If you want to know more about the *license.dat* file and how it works, see *Licensing Administration*, "How licensing works."

The licensing daemons

FLEXlm works on the client/server model. The FLEXlm license manager daemon, named *lmgrd*, is the server process, which runs continuously on a host server machine somewhere in your network. The AVS process is the client. When you start AVS, a license request is passed to the *lmgrd* process which, along with another daemon process - the *avs_lmd* vendor daemon - handles and administers the request.

Thus, in order to have licensing work correctly at your installation, you need to have:

- a valid *license.dat* file,
- the *lmgrd* license daemon (FLEXlm version 5.12 or higher) running on the host server, and
- a copy of the *avs_lmd* vendor daemon running on the host server.

Note: Short-term demonstration licenses do not use the licensing daemons. See "Installing a demonstration license" below.

If you want to know more about the licensing daemons and how they work, see *Licensing Administration*, "How licensing works."

Selecting your host server

You must decide which machine in your network will be the host server on which the licensing daemons, *lmgrd* and *avs_lmd* will run; in order to obtain your *license.dat* file, you must provide Advanced Visual Systems with the machine id of this machine. When selecting the host server, some points you should consider are:

- If you have purchased a node-locked license (where AVS executes on a single host workstation), will that same workstation be used as the license server, or will you be using a different machine as the license server? If you are using different machines, you will have to provide Advanced Visual Systems with the machine ids for both machines.

To find out more about node-locked licenses, see *Licensing Administration*, "Using node-locked versus floating licenses."

- Is FLEXlm already being used on your network and, if so, do you plan to use the same licensing daemon and license.dat file for AVS? If so, you must provide Advanced Visual Systems with the same machine id being used in the existing *license.dat* file.

To find out more about multiple-vendor use of FLEXlm, see *License Administration*, "Multiple vendors using FLEXlm."

Once you decide which machine in your network will act as the licensing server, you must obtain a valid *license.dat* file from Advanced Visual Systems or your local distributor. See the next section, "Obtaining a license," for instructions.

Obtaining a license

Whether you are installing a new license or upgrading an existing license, you must obtain the appropriate licensing information from Advanced Visual Systems or your local distributor. This is the information that goes into the *license.dat* file that FLEXlm uses to control licenses. Follow these procedures to obtain the new or upgrade license information.

1. Determine your AVS order number. The order number (format 00xxxx) is written on the envelope inside of the AVS shipment box. A copy of the number may also be on the packing slip

outside the box. Write this order number down. Without it, Advanced Visual Systems cannot quickly verify your purchase.

2. Find the machine id of the host system that you have selected as your licensing server; this is the machine where the *lmgrd* license manager daemon and the *avs_lmd* vendor daemon will be running. If you have purchased a node-locked license, you will also need to get the machine id for the node machine if it is different from the host server.

3. The simplest way to find your machine ids depends on whether you are upgrading an existing license or obtaining a new license.

   - If you are upgrading an existing license:

     The easiest way to re-identify the machine id that is required by your licensing configuration is to locate your existing *license.dat* file and use the machine id that appears on the **SERVER** line of the file. The *license.dat* file is either in the */usr/local/flexlm/licenses* directory or at the location defined by the **LM_LICENSE_FILE** environment variable. In the following example of a *license.dat* file, the machine id is shown in **boldface** type:

     ```
     SERVER sherman machineid 1700
     DAEMON avs_lmd /usr/avs/license/avs_lmd
     FEATURE AVS avs_lmd 5.000 1-jan-00 4 FB9FE40AF9AA12EEC80A ""
     ```

     If you have purchased a node-locked license, the machine id of the node where AVS runs appears at the end of the **FEATURE** line. If this is different from the server machine, you must provide this machine id to Advanced Visual Systems as well:

```
FEATURE AVS avs_lmd 5.000 1-jan-00 4 FB9FE40AF9AA12EEC80A "" machineid
```

     SGI platform note: SGI users cannot use the hexadecimal machine id as it appears in the *license.dat* file. Instead, use the decimal representation produced by the */etc/sysinfo -s command.*

   - If you are installing a new license:

Use option #9 of the *licdiag* utility to obtain the machine id on any UNIX platform. (See *Administring Licenses*, "licdiag", for more information about *licdiag*.)

If the *licdiag* utility is not available, enter the appropriate command - using *rlogin* or *rsh* - on the system(s) you are trying to identify. The machine id is usually not the hostid. Rather, it is a unique number,

akin to a serial number. Case ("E" or "e") is not significant. Different platforms have different commands for finding the machine id, as listed in the following table.

Table 2-2. Getting Machine ID on Different Platforms

System type	**Command to obtain machine id**	**Sample result**
HP 9000/7xx	$AVS_PATH/license/lmhostid	77DBB9B3
IBM RS/6000	/bin/uname -m	000018221800
SGI	/etc/sysinfo -s (See SGI platform note below.)	1662007164
SunOS 5.x	/usr/ucb/hostid or /usr/5bin/hostid	230001cd
Compaq Tru64	$AVS_PATH/license/lmhostid	0000f82536ce

SGI platform note: On SGI workstations the command shown above produces a decimal representation of the machine id. This is the number that you should report to Advanced Visual Systems when discussing licensing issues. However, the *license.dat* file requires the hexadecimal representation of the machine id. The license codes that you receive from Advanced Visual Systems are in the correct hexadecimal representation.

4. File a request for a license with Advanced Visual Systems Customer Support. You must provide your AVS order number, your machine ids, and information about your site. The easiest way to do this is using the interactive license request form on our web site *http://www.avs.com*. Look under "Services" and "Licensing" for further information. This web site also provides additional information on common problems users have with licensing and other helpful topics.

   If you are outside the United States or Canada, or if you do not have web access, fill out the "Request for License" form that is inside of the envelope that came with the AVS shipping box. FAX the completed form to the location appropriate for you:

   In the US and Canada, contact Advanced Visual Systems Customer Support:

   ```
   FAX:   781-890-6903
   e-mail:   support@avs.com
   Phone:   1-800-4AVS-001 (800-428-7001)
   ```

   - Outside of the US and Canada, contact the local sales office through which you purchased the product.

Once you have relayed this information, Advanced Visual Systems will send you your new *license.dat* file either by FAX or by e-mail. If you are installing a license for the first time, see the following section, "Installing a new license.dat file." If you are upgrading an existing license, turn to "Upgrading an existing license.dat file."

Installing a new license.dat file

If you are installing an AVS license for the first time, you must obtain a *license.dat* file from Advanced Visual Systems. If you have not yet obtained this file, follow the instructions in "Obtaining a license" below to get this information from Advanced Visual Systems.

Once you have obtained your *license.dat* file:

1. Make a backup copy of the original file that you received from Advanced Visual Systems for safekeeping. It is extremely important to keep a copy of the original file for trouble-shooting purposes.

2. Edit the non-backup copy and replace the "unknown" string on the **SERVER** line with the hostname of the server system.

3. On the **DAEMON** line, modify the pathname to the *avs_lmd* binary file if it is incorrect, or if you have moved *avs_lmd* to another location. By default, *avs_lmd* is installed in *$AVS_PATH/license.*

4. Install the file either in */usr/local/flexlm/licenses/license.dat* (its default location) or at another location if you prefer. If you install it in a non-default location, you must set the **LM_LICENSE_FILE** environment variable to point to the new location.

Be extremely careful when editing and copying the *license.dat* file. Let long lines wrap; do not insert carriage returns; and do not modify the file in any way unless specifically instructed to do so.

Once you have installed the *license.dat* file, use the *licdiag* utility to confirm that the file is correct and complete so that licensing will work properly. Turn to "Running licdiag to verify your licensing installation" below.

Upgrading an existing license.dat file

If you are upgrading to AVS 5.5 from a previous release you do **NOT** have to obtain a new *license.dat* file to replace your existing *license.dat* file unless you require a transfer. If necessary, follow the instructions in "Obtaining a license" to get a new license file from Advanced Visual Systems.

Once you have obtained the new *license.dat* file:

1. Locate your existing *license.dat* file. This file is either in the */usr/local/flexlm/licenses* directory (its default location) or at the location defined by the **LM_LICENSE_FILE** environment variable.

2. Make backup copies of both your existing *license.dat* file and the new *license.dat* file for safekeeping. It is extremely important to keep these copies for trouble-shooting purposes.

3. Either replace the old *license.dat* file with the new file, or edit the old file and add the new **FEATURE** and **DAEMON** lines to it, replacing the old **DAEMON** and **FEATURE** lines for the earlier Advanced Visual Systems product(s). The old **FEATURE** line(s) will start with the following:

   ```
   FEATURE AVS . . .
   ```

   It is extremely important that you enter the line exactly as sent. If you received the new **FEATURE** line through e-mail, cut and paste the information into *license.dat*. Let long lines wrap; do not insert carriage returns.

Once you have updated the *license.dat* file, use the *licdiag* utility to confirm that the file is correct and complete so that licensing will work properly. Turn to "Running licdiag to verify your licensing installation."

Installing a demonstration license

If you are installing AVS 5.5 for only a short time - if, for example, you are evaluating AVS on your system - follow the instructions in this section to install a short-term demonstration license.

1. Obtain the daily demo license password from Advanced Visual Systems. In all cases, contact the local Advanced Visual Systems sales office that is working with you.

2. From a UNIX shell, run the program:

   ```
   $AVS_PATH/license/demo_license
   ```

   where *$AVS_PATH* is the name of the directory in which you installed AVS 5.5. The *demo_license* program creates a temporary *license.dat* file in the current directory.

3. When prompted, enter the daily demo license password you obtained from Advanced Visual Systems.

4. Enter the number of days for which the license will be valid; the maximum number shown is the default. You may enter a smaller, but not a larger number. The program then exits, printing an explanation of what to do with the temporary license.dat file that has been created in the current directory.

5. Follow the instructions provided by the *demo_license* program that will make the temporary *license.dat* file available to the licensing system. The easiest approach is to set the **LM_LICENSE_FILE** variable as follows:

   ```
   setenv LM_LICENSE_FILE ‘pwd‘/license.dat
   ```

   Note: If you are using the FLEXlm licensing mechanism for other products in your environment, **LM_LICENSE_FILE** may already be set for this purpose. If that is the case, append the contents of the temporary *license.dat* file to the *license.dat* file currently in use on your system. Always make a backup copy of the file before you edit it.

Alternatively, if FLEXlm is not currently being used on your system, you can use *mkdir* to create the default */usr/local/flexlm/licenses* directory and copy your new *license.dat* file into that directory. This is the default location for the *license.dat* file, so you do not need to set **LM_LICENSE_FILE** in this case.

Short-term demonstration licenses do not use the licensing daemons so you are ready to run the AVS product once you have completed these installation procedures.

Running licdiag to verify your licensing installation

The *licdiag* licensing diagnostic utility is designed to help you examine and administer your licensing setup. Use this utility, which is installed in *$AVS_PATH/license*, make sure that your *license.dat* file is complete and correct, and that you are using the appropriate version of the FLEXlm-based licensing software. For more information about *Administering Licenses*, "licdiag."

When you invoke *licdiag*, the utility prints out information about your licensing installation. Check this information to make sure that it is complete and accurate. In particular, make sure that machine you have selected as the licensing server is correctly identified. If the server is incorrect or absent, contact Advanced Visual Systems. Although it is unlikely, you may need to obtain a new *license.dat* file.

Once you have confirmed that your installation is set up properly, you must install and/or start the licensing software. See the following

section, "Installing the FLEXlm software daemons," for instructions on how to do so.

Installing the FLEXlm software daemons

There are two software daemon processes that are used to administer licensing for AVS: the FLEXlm *lmgrd* license manager daemon and the *avs_lmd* vendor daemon. You must install these software daemons on your system if:

- you are installing an AVS license for the first time, or
- FLEXlm is already in use in your network (either for a previous Advanced Visual Systems product or another vendor's product) but is not FLEXlm version 5.12 or later.

In particular, if you are already running AVS 5.4, you are most likely using FLEXlm version 5.12 already. Earlier versions of FLEXlm do not work with AVS 5.4 and above; you must install and start FLEXlm version 5.12 instead, which will also work for all earlier versions of AVS5.x.

If another vendor is using FLEXlm to administer licensing, see *Administering Licenses*, "Multiple vendors using FLEXlm" for information on how to set up your licensing.

You can use option #4 of the *licdiag* utility to find out what version of FLEXlm is currently running. See *License Administration*, "licdiag" for more information.

Installing the daemons

When you install AVS, the *lmgrd* and *avs_lmd* binary files are put into the $AVS_PATH/licensed directory.

You can either leave these binaries in their default directory or, if you want, move them to another directory on your host server system. Note that if you move *avs_lmd* to another directory, you will need to edit the **DAEMON** line in your *license.dat* file to reflect its new location (see step 3 in "Installing a new *license.dat*' below.

Starting the daemons

You only need to arrange to start the *lmgrd* license manager daemon; *lmgrd* starts the *avs_lmd* vendor daemon (and all other vendor daemons, if there are any) when it initializes. The basic command to start the lmgrd license daemon is:

```
lmgrd
```

Option #1 of the *licdiag* utility provides an easy way to start or restart the *lmgrd* license daemon. See *License Administration*, "licdiag", more information about licdiag.

The easiest way to start the *lmgrd* license daemon for ongoing use is to put the command into your server system's boot file so that *lmgrd* will be automatically started when the system boots. The following command will start *lmgrd* running in background, and direct all transaction messages to the output file *license.logfile:*

```
lmgrd > license.logfile &
```

See *Administering Licenses*, "Starting the lmgrd license daemon", for more information about the *lmgrd* command line. For a complete description of *lmgrd* command line options, see the *lmgrd* manpage.

Step 5: Test Install

Once AVS has been installed and licensed, you can verify its operation by running some of the demos:

First, just try to execute AVS:

```
$AVS_PATH/bin/avs
```

A message such as "Checkout failed. No such feature exists" indicates a licensing problem (See "Step 4: Licensing" above).

Select "AVS Applications" then "AVS Demo" to access the demo menus. Each pull down menu presents functional areas or sets of demos. For starters, select "General AVS" then "Geometry Viewer". Under the "Objects" menu, try "Transforming Objects" to see basic Geometry Viewer operations. Explore the other areas as your interests dictate. When done, use the "Return" selection on the "Control" menu to return to the top level, then use "Exit" on the "Control" menu to leave the demos. You can abort or pause any script using the script brower window at the bottom.

You will also find a list of demo scripts under the Network Editor. Select the "Help" button on the main Network Editor window, then select "Help Demos" from that panel. This will provide a list of module specific demos useful in learning how the modules work.

Step 6: Tell Users How to Find AVS

This section summarizes what users must do to find AVS. The story is re-described in more detail in the "$Path—Installing and Finding AVS Anywhere in a Directory Tree" section of the *AVS 5 Update* manual.

In order to find AVS, you and all of your users will have to define the AVS **Path** variable. In addition, if the FLEXlm *license.dat* file is located anywhere except in the file */usr/local/flexlm/licenses/license.dat*, users will need to define the **LM_LICENSE_FILE** environment variable.

Briefly, assuming AVS was installed in */u2/avs*, both you and your users would:

1. Add */u2/avs/bin* to your directory file search path.

2. Tell AVS where to find itself by one of three methods, in this order of precedence:

   - Start AVS with the **-path** option:

     ```
     avs -path /u2/avs
     ```

   - or, define **Path** in your personal *.avsrc* file:

     ```
     Path      /u2/avs
     ```

   - or, set the **AVS_PATH** environment variable in one of the user's startup files, usually either *.login* or *.cshrc* (*csh*), or *.profile* (*sh/ksh*).

     ```
     csh:     setenv AVS_PATH /u2/avs

     sh/ksh:  AVS_PATH=/u2/avs
              export AVS_PATH
     ```

     It is convenient to define **AVS_PATH** even if the other mechanisms are used so that one can find AVS files by referring to $AVS_PATH/*filespec*.

   - If none of these is explicitly defined, **Path** will default to */usr/avs*.

3. If the *license.dat* file is anywhere except in the default */usr/local/flexlm/licenses/license.dat* (as viewed from each user's workstation file system hierarchy), then the user must explicitly tell AVS where to find the critical *license.dat* file. To do this, each user must define the **LM_LICENSE_FILE** environment variable in

one of their personal system startup files (usually *.login*, *.cshrc*, or *.profile*).

Assuming the *license.dat* file is being kept in */u2/avs/license/license.dat*:

```
csh:     setenv LM_LICENSE_FILE /u2/avs/license/license.dat

sh/ksh:  AVS_PATH=/u2/avs/license/license.dat
         export AVS_PATH
```

More information on AVS licensing from a user's perspective is found in the online files *<install-dir>/avs/license/user.ps* (PostScript) and *<install-dir>/avs/license/usermessages* (ASCII).

Distribution Contents

All files for this distribution are kept in the directory in which you installed AVS.

The following is a detailed listing of the contents of this release:

avs/LUI
: Contains LUI-specific runtime files, the pixmaps for the icons used in AVS.

avs/applications
: Contains the command files for the canned applications.

avs/avs_library
: Contains the supported module executables. Also contains the Animator executable and the animation module executables.

avs/bin
: Contains most of the geometry conversion executables that were created in the *filter* directory and the *avs* and *avs_dbx* programs.

avs/chem_lib
: Contains the Chemistry module executables.

avs/data
: Contains sample datasets for use with AVS.

avs/demo
: Contains a number of demonstration CLI scripts that allow you to more easily explore AVS functionality. You can access the scripts through the Help Demos button on the Help Browser.

avs/demosuite
Contains all the CLI scripts and help text associated with the AVS Demo Suite. You can access the scripts through the AVS Demo button under AVS Applications.

avs/examples
Contains source files for example AVS modules. It also contains a *Makefile* that you should use as a template for your own module make files.

avs/filter
Contains the source and some example data for building geometry tools and for template/example geometry converters.

avs/include
Contains the header files necessary for using the AVS libraries.

avs/lib
Contains the library files necessary for using the AVS libraries.

avs/license
Contains binaries for the license daemons and administrative commands, a sample *license.dat*, and licensing software man pages.

avs/math
Contains information and files to enable a direct interface between AVS and Mathematica™. (See the README file in that directory for more information.)

avs/networks
Contains links to the viewer subdirectories of *avs/applications*, which can be used to access sample networks that make up the canned viewer applications.

avs/relnotes
Contains printable PostScript versions of these *Installation and Release Notes*. The printer must support the Palatino font.

avs/runtime
Contains AVS-specific files that must exist in order to run *bin/avs*, including the online *help* files.

avs/test
Contains an automated test procedure for verifying AVS operation.

avs/unsupp_mods
Contains the unsupported module executables.

Source Code

Source code provided with this release consists of templates and examples that are useful for creating new geometry converters (in <*installdir*>*/avs/filters*) and AVS modules (in <*installdir*>*/avs/examples*). The AVS/Animator module source code is now under the *examples* directory as well. For more information see Chapter 3, "AVS 5.5 on UNIX Platforms".

Source code for AVS supported modules may be requested through AVS Customer Support. Please see Chapter 11, "Debugging in AVS5.5" for more information.

AVS 5.5 ON UNIX PLATFORMS

CHAPTER THREE

Introduction

This chapter discusses information common to all platforms - new features and documentation; implementation information regarding the X Window environment and OpenGL graphics libraries; programming tips and topics; and defects fixed for this release.

AVS 5.5 NOTE: Several new features have been added (Geometry Camera module, new examples, Event masking) and the Animator source code is included; OpenGL stereo support has been broadened to include higher end SGI workstations, Compaq Tru64 Alpha, Solaris, and HP Workstations (IBM and Linux have not been tested); the AVS 5 documentation set is now online on the CD in PDF format; new documentation has been added on a number of topics; and a significant number of bugs have been fixed affecting all platforms. See the appropriate sections below.

NOTE: This document only covers platforms and operating systems that are currently supported for AVS 5.5. If you require versions of AVS 5 to run on older operating systems, please contact AVS Customer Support or your local distributor for appropriate media and release notes.

Following this chapter are platform specific chapters which document the differences, special considerations and known problems unique to that platform. Each chapter provides information on the following:

- hardware prerequisites (graphics hardware, memory, disk space and swap space)
- software prerequisites (operating system, graphics software, and other topics)
- general usage information (how AVS subsystems work)
- programming considerations for C, C++, and FORTRAN module writing.

New Features in AVS 5.5

AVS 5.5 includes the following new features:

- Geometry Camera module provides a better secondary camera to support the Geometry Viewer module
- Event Masking provides a means of masking out undesirable mouse or function key input to the Geometry Viewer
- Additional example modules provide a sample FORTRAN coroutine, a new test field data generator, and a GEOM data output/viewer module

Geometry Camera

A new builtin "Geometry Camera" module has been added to provide improved support for secondary cameras associated with a Geometry Viewer module. This wraps a camera window within a user interface like the Geometry Viewer module's main camera window, but does not provide image output and other Geometry Viewer features.

The Geometry Viewer module produces a main camera ("view") with the following characteristics:

- a title bar editable by the Layout Editor and recorded in networks
- an upper-left-dimple activated menu including zoom, unzoom, etc.
- resistance to being closed using ordinary window manager commands
- position and layout recorded in networks and scripts

In the Geometry Viewer (as opposed to the module with the same name), adding a new camera (Camera/New Camera) produces a camera without these characteristics.

The new Geometry Camera module takes as input a specific Geometry Viewer module output ("Trigger" - colored pink and now visible) to associate itself with that module and adds a new camera to this main view with all the characteristics noted above. A Geometry Camera can be detached from one viewer and reattached to another, restored from a network file properly, edited by the Layout Editor, and destroyed without adverse effects.

Demonstration:

- Start the Network Editor, then on "AVS Network Editor" panel, hit Help, then hit Demos. Close Help window.
- Select "..(demo)" at the top of the menu to go up one directory then go down into the "geom_viewer" directory.
- Run geom_camera script to create multiple Geometry Viewers with geom_camera modules attached to them.

Function Key, Arrow Key, Mouse Event Masking

A new feature has been added to "mask out" events coming from specific function keys, arrow keys or mouse events. This permits customers to prevent their end users from inadvertently performing various Geometry Viewer transformations linked to these keys. The developer encodes the key values in a "mask" in which each bit represents a function key, arrow key or mouse click. The combined mask is then set in the *.avsrc* file.

Function Key mask (KeyMask):

- Bits, 1-12: Function Key 1-12

Arrow Key mask (ArrowMask):

- Bits 1-4: Left / Right / Up / Down
- Bits 5-8: Shift Left / Right / Up / Down

Mouse key values: (MouseMask)

- Bits 1,2,3: Left / Middle / Right
- Bits 4,5,6: Control - Left / Middle / Right
- Bits 7,8,9: Shift - Left / Middle / Right

Sample *.avsrc* file: (Values may be in hex, decimal, octal, etc)

```
KeyMask         0x3   (Bits 1, 2 - masks out Function Key 1, 2)
ArrowMask       0x1   (Bits 1 - masks out Arrow key 1 (left)
DebugMask       0x1   (Bits 1 - prints out debug messages)
MouseMask       0x48  (Bits 4, 7 - mask out Control-Left and Shift-Left)
```

Demonstration:

- Create a *.avsrc* file in your home directory or the current directory with the sample shown above. See the *AVS User's Guide*, page 5-12 "Transforming Objects" which details what mouse key strokes should do.

- The DebugMask will show debug messages like "Mask Values:" and "Masking out xxx" as appropriate events occur and are masked out.

New Example modules

Several new example modules have been added in AVS 5.5.

- Corout f.f - a second FORTRAN coroutine example showing how basic data types are passed in and out

- Test field.c - a more advanced field data pattern generator for testing

- Write geom.c - a diagnostic output module for the GEOM data type

Corout f.f

This module is a more complete example of a FORTRAN coroutine that shows all basic data types being passed in as inputs and parameters and then copied out as outputs.

A test script is provided to show all inputs and outputs in use - *$AVS_PATH/demo/examples/corout_f.scr*

Test field

This module has been used extensively in internal testing at AVS and is being provided to help users produce a wider range of diagnostic test field data for testing their own modules or providing simple test cases back to AVS Customer Support to demonstrate problems they are having with supported modules. It also provides a more extensive example of a module that is actively managing and reconfiguring its user interface using CLI commands and the AVScommand function. **NOTE:** Test field's name is close to that of two earlier and simpler test field examples, used to demonstrate C and FORTRAN field creation.

Documentation is provided as an online module man page under *$AVS_PATH/runtime/help/modules/test_fld.txt* which should appear when you select module documentation for the module.

A test script is provided to show several different sample outputs - *$AVS_PATH/demo/examples/test_fld.scr.*

Write Geom

This module provides a variant on the "print field" diagnostic module that provides both binary and text output from GEOM data. It is useful to help diagnose GEOM data issues by displaying the data in a more readable form; often issues relate to extreme data values and invalid values such as "NaNs" (not-a-number). It also provides a simple example of how modules such as "print field" are able to display text files in a text window as part of an AVS 5 network.

Write Geom writes GEOM format data to a binary output file then uses an external filter program (*$AVS_PATH/bin/geom_to_text*) to convert the binary file to a text file. It then uses an AVS text browser widget to view the text file within the AVS user interface. The binary file is readable by "read geom"; the text file can be read using any text editor.

The binary file name must be set before any output is generated. Once this is set, the binary file is written out and the text file automatically produced and read into the browser. A very large GEOM file might generate enough data to cause the module to crash when read in. Unsetting the text file name in the module will prevent it being generated or read in; you can use *geom_to_text* externally as needed.

A test script is provided to show this module in use - *$AVS_PATH/demo/examples/write_geom.scr*.

AVS/Animator source code

The AVS/Animator module source code is now provided with the example modules, both to provide a more extensive example module and to offer users the opportunity to extend the Animator to provide alternate output formats not currently supported. For more information in general, see the associated end user documentation in the *Animating AVS Data Visualizations* manual included in the main doc set and in the online PDF files provided with AVS 5.5.

The source code is located under *$AVS_PATH/examples/animator* and consists of a number of subdirectories containing the Animator module, supporting libraries, and the associated modules that make up the Animator package. It consists of the following subdirectories:

- *anim_lib:* Animation operations needed by the Animator module.
- *asf:* Applications Support Functions library which is responsible for the parsing of the menu file and the generation of the user interface menus.
- *avs_library:* Empty directory that the modules will be installed into

- *bfa:* The Animator module source code, starting with *bfa_main.c* which defines the Animator user interface and module compute operations.

- *lib:* Empty directory that *anim_lib* and *asf* will copy libraries to; the Makefile will add links to standard AVS libraries here also.

- *modules/animator:* Source code for supporting modules such as **output ImageNode, output VideoCreator, prepare video, read frame seq** and **write frame seq.**

In order to make the animator from source code, do the following:

```
cd $AVS_PATH/examples/animator
make
```

This will create a few links from the animator source to the standard *$AVS_PATH* installation areas, then visit the subdirectories to make the supporting libraries and modules.

Documentation updates

AVS 5 Documentation set in PDF format

The AVS 5 documentation is now available online in Adobe PDF (Portable Document Format) files providing the core set of books in a viewable, searchable, and printable format. The set can be installed from the **AVS5_DOC** product archive or read directly from the CD (*/cdrom/avs5_doc* on most platforms). Advanced Visual Systems grants users permission to print and make copies of part or all of the documentation set for internal use at their site. **Note:** A few books were not available in PDF format at release time (*AVSGraph User's Guide, Technical Overview*, and *UCD Builder's Guide*); contact AVS Customer Support for possible future availability of these book files from our web site.

Installation

The AVS 5 Documentation package can be installed onto your hard disk if you use *install.avs* and select the **AVS5_DOC** package to install in *$AVS_PATH/avs5_doc* or a directory of your choice. The same files are available for direct access on the CD in the *avs5_doc* directory at the top level; the full pathname may be */cdrom/avs5_doc* or */CDROM/AVS5_DOC* depending on your platform's CD conventions.

You will need a copy of the Adobe Acrobat Reader in order to read or print the PDF files. If you don't have a copy you can download the software at no cost from the Adobe web site at

http://www.adobe.com/prodindex/acrobat/readstep.html. Most UNIX platforms are supported as well as Windows 95 and NT.

Getting Started

Once you have obtained an Acrobat reader and have installed the package or mounted the CD for direct access, open up the introductory file, *$AVS_PATH/avs5_doc/intro.pdf*, which provides a general overview of the organization and evolution of the AVS 5 documentation set, information on where to get started, and direct links into the individual manual PDF files stored under the *books* subdirectory. Live links are highlighted with a red border and will take you directly to the appropriate section or manual file as appropriate; the Table of Contents of each manual also contains live links into the chapters and subsections of each manual for easier access.

If you should have any troubles using the external links in the *intro.pdf* file, you can directly access the individual manuals in the *books* subdirectory. **Note:** If you are printing out sections of the PDF files, you must set the Postscript level to "Level 1" in the print dialog; selecting "Level 2" will cause problems with some of the images.

New Chapter: Debugging in AVS 5

A new chapter has been added to this release note document that provides an overall summary of how to approach debugging modules in AVS 5.5. This chapter draws together information and techniques from throughout the AVS 5 product and is targeted towards both new and experienced module writers.

AVSfield_points_array_size

The AVSfield_points_array_size routine was not previously documented but has been available since AVS 5.0.

C:
AVSfield_points_array_size (*uniform, ndim, nspace, dimensions* **)**
int *uniform, ndim, nspace, *dimensions;*

FORTRAN:
AVSFIELD_POINTS_ARRAY_SIZE (*UNIFORM, NDIM, NSPACE, DIMENSIONS* **)**
INTEGER *UNIFORM, NDIM, NSPACE*
INTEGER *DIMENSIONS(ndim)*

This routine calculates the number of items in a field points array depending on the type of field it is.

```
Inputs: uniform         - One of UNIFORM, RECTILINEAR, IRRREGULAR,
                        or UNIFORM_DONTCARE.
        ndim            - Number of dimensions in computational space.
        nspace          - Number of dimensions in model space.
```

```
            dimensions      - Array of ndim values.

        Returns: Number of floating point values required for the points array.
```

AVSoutput_string

The AVSoutput_string function is a utility to help FORTRAN modules allocate strings for use as function outputs. It has existed since at least AVS 5.3 but has not been formally documented.

FORTRAN:
INTEGER AVSOUTPUT_STRING (*OUTPTR, STRING* **)**
INTEGER *OUTPTR*
CHARACTER*(*) *STRING*

This routine converts a FORTRAN string value to a newly allocated copy that can be sent to output. It will automatically allocate/reallocate storage space as needed.

```
Inputs: outptr - pointer to string copy passed into the compute function
                 to hold the output string.
        string - FORTRAN string buffer to be copied
Returns: 0 if failed, 1 for success
```

AVSpath variable

AVS 5 modules can find out the path to the home AVS 5 directory by using the AVSpath variable. This is now included in the *$AVS_PATH/include/avs.h* header file but in releases prior to AVS 5.5 needs to be directly declared as shown in this example from the AVS-Graph module which is using AVSpath while looking for the *$AVS_PATH/runtime/AVSGraph directory:*

```
extern char *AVSpath;

if (AVSpath != NULL)
    putenv(strcat(strcat(tmpstr,AVSpath),"/runtime/AVSGraph"));
else
    putenv("UNIDIR=/usr/avs/runtime/AVSGraph");
```

Choice value output example

Some customers have had trouble in successfully sending a string value from one module to act as a choice value input or parameter value in another module downstream. The following is an example of of a generic choice output module description and compute function. The resulting string will be typed as a choice and passed to any module needing a choice parameter downstream. In this case it is up to the user to know valid choices that will be acceptable downstream.

```
MODchoice()
{
   int choice_compute(), param;

   AVSset_module_name("choice", MODULE_DATA);
   AVSset_module_flags( COOPERATIVE | REENTRANT );
   AVScreate_output_port("choice_value", "choice");
   param = AVSadd_parameter("choice_string", "string", 0, 0, "");

   AVSset_compute_proc(choice_compute);
}

static choice_compute(outvalue1,invalue)
char  *invalue, **outvalue1;
{
   if (!invalue)
     return(1);

   *outvalue1 = strdup(invalue);
   return(1);
}
```

Read Field module accepts short data

The current Read Field documentation does not mention it, but the module does accept short data and has done so since AVS 5.0 on all platforms.

Geometry Viewer object parameter

The man page does not provide information on this parameter. It is used to see what the current object is (used in the Data Viewer module for example). You can NOT change the current object by setting this, as it is for informational use only.

Unexposed command line options

A number of existing AVS command line options have previously not been exposed or documented but were used for internal testing purposes. Because some of these may be useful to customers they have been documented in the usage command line option and are described below:

- -mod_host <host>: default host to run modules on. This will result in nearly all supported modules being run as remote modules on the given machine. The machine must be present in the *.hosts* file.

- -mod_time: print out time spent in each module executed. This option can be useful for analysing where execution time is being

spent.

- -no_display: run without visible windows. AVS always requires access to an open X display in order to allocate resources for the Geometry and Image Viewers. This option will prevent virtually all windows from being mapped to the display, running invisibly without disturbing existing users of the workstation.

- -swrender: Select software renderer as default instead of the hardware renderer. The hardware renderer may still be selected, unlike *-nohw*.

- -test_path: Specify testing directory, exposed to test networks and scripts as the CLI variable $TestPath; used as a pathname reference during internal testing.

- -vistype: (New in AVS 5.5) Specify the visual type in the same way that the "VisualType" avsrc option works. Recognizes the first letter of the visual types for convenience and ignores the rest of the token. Recognized values include D(irectColor), P(seudoColor), T(rueColor), and V(isualID) <vid>. For example, the following would set avs to use VisualID 0x31 (use *xdpyinfo* to see a list of your available VisualIDs).

  ```
  avs -vistype v 0x31
  ```

New Arbitrary Slice parameter Slice Size

The arbitrary slice module selects a plane size based on the XY extent of the object being viewed. Depending on the object, this plane may not be large enough as it is rotated around within the overall dimensions of the object - for example, an oblong brick object with the XY cutting across the small dimension of the brick would use a small square slice plane that won't extend across the length of the brick.

This has been fixed by adding a new choice parameter to the module called "Slice Size" with the following choices:

- XY slice: use the XY slice as the plane size

- YZ slice: use the YZ slice as the plane size

- ZX slice: use the ZX slice as the plane size

- Diagonal slice: use the diagonal of the extents (across opposite corners) to determine the size of the plane.

AVS runs as an X Window System client. It uses X protocol requests (via Xlib) to the X server to produce its user interface. The interface includes:

- the main AVS menu
- the Image, Geometry, and Graph Viewer control panels
- the Network Editor
- all control widgets such as file browsers, pop-up menus, dialog, message, and help panels
- the Image viewer's viewport windows, and the **display image** module's output window
- the Graph Viewer's plot windows

The images that appear in Geometry Viewer scenes windows can be produced by two different rendering mechanisms:

- A **software renderer** that implements a set of graphics primitives (polygons, lighting model, perspective, shading, color, etc.) in software, rendering the resulting image into an X Window System image. The image is transferred to the X server using the *xlib* XPutImage call.
- A **hardware renderer** that uses the hardware graphics library to create renderings that are displayed on the screen using the platforms's native hardware graphics subsystem. Depending on the platform this library will be GL, OpenGL, XGL, or PHIGS.

You can switch between hardware and software renderers while AVS is running using the rendering switches provided on the Geometry Viewer **Cameras** submenu. You can control which renderer will be the default active renderer when AVS first starts using the **-renderer** command line option or **Renderer** keyword in your personal *.avsrc* file. When multiple hardware renderers are available, you need to start different *avs* kernels to access them (*avs, avs.gl, avs.phigs*, etc.).

Z Buffer Required for Hardware Rendering: On IBM and SGI platforms, the hardware renderer requires the presence of a Z-buffer in order to function. A Z-buffer should be present on all SGI platforms

except the Personal IRIS, where it is an option. On the IRIS Indigo, the hardware Z-buffer and GL graphics functions are emulated in software. On IBM or SGI systems without a Z-buffer, you must use the software renderer.

Both hardware and software renderers will function on 8-plane pseudocolor or 24-plane true color frame buffers. However, you will need to establish the correct X color visual, since the default X visual used by the X server when it starts is not always the most powerful that the system is capable of supporting. This is discussed in the "Starting AVS" section of the appropriate platform specific chapter.

All screen output that AVS performs on the display via X Window System calls is bounded by the capabilities of the X server implementation on the workstation.

All rendering AVS does via hardware graphics library calls is bounded by the capabilities of the particular library and its underlying hardware graphics adapter.

The software renderer implements its own 3D graphics model, and is bounded only by what graphics primitives it does or does not implement in software. However, the final color rendition on the screen is constrained by the number of color planes that the X server creating the window can support: 8 bits or 24 bits.

See the table and notes in the "Geometry Viewer" section in each platform specific chapter for further details on rendering capabilities.

Graphics Libraries: OpenGL

AVS 5.5 expands the use of OpenGL as a common graphics interface for 3D hardware rendering, providing OpenGL as the default renderer for all supported platforms, now including HP-UX 10.20.

OpenGL provides a more standard cross platform graphics layer for AVS 5.5 and future releases. It enhances remote display graphics support and centralizes AVS 5 cross platform graphics support, permitting better use of shared extensions in new releases.

Common features

A number of features are common to OpenGL renderers regardless of platform. On UNIX platforms (X Windows), GLX is the OpenGL extension. It embodies a programming interface and a protocol for client-server communication. GLX provides a mechanism for OpenGL to allocate and control display resources on the X server. It binds an OpenGL rendering context to a specific X visual that is selected from the list of visuals supported by the X server.

There are certain restrictions on the visuals that can be used with OpenGL. Only displays that use an RGB pixel format with a Z-buffer support the OpenGL renderer. To double buffer the view window under GLX, OpenGL must be supported in a double-buffered TrueColor or DirectColor X visual with an associated Z-buffer. The GLX specification requires that at least one such visual is supported.

Several UNIX tools enumerate the available X visuals and their support for OpenGL via the GLX extension. A standard X Window program, called *xdpyinfo*, lists X server parameters and all suported visuals with their X Window characteristics. A GLX inquiry utility, *xglinfo*, is usually supplied with OpenGL examples. *xglinfo* lists general GLX parameters as well as each X visual and its support for OpenGL. OpenGL installations can repeat X visuals, which differ only in the configuration of additional (non-X) buffers used by OpenGL (for example, alpha, depth, stencil). RGB visual formats can be 8-bit (3-3-2), 12 bit (4-4-4), or 24-bit (8-8-8). The OpenGL renderer does not use blending modes that require the framebuffer to store alpha values. **Note:** If you do not have *xglinfo* available on your SGI system, you can download it from *ftp://sgigate.sgi.com/pub/opengl/contrib/xglinfo.tar.Z*.

OpenGL Renderer Information

When AVS instances a view, a set of inquiries is performed when the OpenGL renderer is first instanced. The values returned from the inquiries contain information about the platform that is used as the OpenGL server (perhaps remotely), configuration of the OpenGL implementation, supported extensions, and characteristics of the view window. This information is used to configure the behavior of the renderer, and it is written to the command terminal if the environment variable AVS_OGL_INFO is set before running AVS.

```
setenv AVS_OGL_INFO 1
```

The following information is displayed when AVS_OGL_INFO is set:

- Confirmation that GLX is running on the server and the GLX version number.
- The OpenGL version number, vendor name, hardware name, and a list of supported extensions.
- OpenGL parameters are displayed. These are ARGB color buffer depths, Z-buffer depth, maximum number of lights, and maximum 2D texture image dimensions.
- The type of connection to the X server (Direct or Indirect), X visual id, and X visual class (TrueColor or DirectColor). The connection tells whether OpenGL and GLX have direct access to the

display when client and server are on the same machine. If the type is Indirect, the client and server are remote or are they are on the same server but display requests pass through the X server. In general, direct connections are faster than indirect connections when the display is local.

Performance

OpenGL performance will vary widely depending on the level of support for the graphics acceleration hardware your system is using. Most newer graphics adapters will be well supported by OpenGL; older adapters will usually be supported but may run slower than in previous AVS 5 releases. Certain vendor specific graphics features may not be as well supported in OpenGL as in prior graphics libraries.

For this reason, AVS 5.5 provides alternate AVS executables on different platforms to continue support for the prior hardware rendering interfaces: XGL on Solaris, GL on IBM and PHIGS on HP-UX 10.20. These executables are provided in the *$AVS_PATH/bin* directory as *avs.xgl*, *avs.gl*, or *avs.phigs* respectively. It is hoped that for most users, the default OpenGL renderer will be most appropriate. When that is not the case, the user may wish to make the alternate renderer the default executable as follows:

```
cd $AVS_PATH/bin
mv avs avs.ogl
ln -s avs.gl avs     /* Or "avs.xgl" or "avs.phigs" as appropriate */
```

The renderer is always listed at the end of the version line to help minimize confusion for the user and AVS Customer Support when the need arises. For example

```
avs -version
```

might show "AVS version: 5.5 (50.85 SunOS5 ogl)".

Optimizing Performance

When using AVS's software renderer option on lower-end platforms, you will probably want to switch on **Bounding Box** on the Geometry Viewer's control panel to reduce the amount of rendering required when you transform objects.

If you will be rendering polygonal spheres such as those produced by the **bubbleviz/scatter dots** and **particle advector** modules, or molecular ball and stick models, you should use the **Subdivision** slider at the bottom of the Geometry Viewer's **Object** menu to reduce the number of polygons used to approximate a sphere. The 1 value represented by having the slider all the way at the left will render a sphere as an 8-

sided diamond. You can always increase the value after you have arranged the scene for high-quality output.

Texture Mapping Limitation

Texture map size is limited by the local implementation of OpenGL and varies across vendors and platforms, ranging from a limit of 64 by 64 pixels to 4096 by 4096 pixels or more. OpenGL also requires that texture maps be scaled to powers of two. When you provide a texture map that exceeds your local systems limit or is not a power of two, you will see warning messages (OGL_load_texture errors). If this causes problems, select the "Filter Texture" button, then reload the texture; this will scale your texture to fit the hardware limitations and smooth out the result.

OpenGL Errors

The following explains some of the more common errors you may encounter when running OpenGL.

Initialization Errors (GLX).

- *no GLX extension on this X server* OR *cannot get version of GLX:* The GLX extension that supports OpenGL is not available or was not configured in the X server when it started. On some platforms, you must start the X server with command line options to get OpenGL and double buffering. See the system prerequisites chapter for your platform.

- *cannot find OpenGL visual:* GLX and OpenGL are supported, but do not provide a satisfactory visual for use by AVS. The specification of GLX ensures that at least one single buffered visual is available. Try to re-instance the OpenGL renderer for a view in single-buffered mode.

- *glXCreateContext failed* OR *cannot create colormap:* GLX and OpenGL are available, and a satisfactory visual was found on the server. However, other resources were not found and the server cannot create an OpenGL renderer window or the associated X colormap.

General Errors.

- *no support for 3D texture:* The application tried to display an object with 3D texture on a system that does not support the OpenGL 3D texture extension. Note that 3D textures cannot be displayed from a remote client unless the client also supports the

relevant extensions. If you do not have one of these high-end machines, use the Software Renderer to display 3D texture.

- *cannot create texture display list:* OR *texture dimensions (mxn) exceeds system limit (N)* The graphics display supports texture, but ran out of resources allocating texture memory from the host's main memory or from the graphics adapter. Reduce the size of the texture using filter modules (for example, downsize), or allocate more swap space, main memory, or hardware texture memory.

- *OpenGL error X in gluBuild2DMipmaps (glTexImage3DExt, glTexImage2D, gluScaleImage):* A standard OpenGL error occurred in one of these functions. The OpenGL error message should follow this error. Possibly, texture memory overflowed, in which case, reduce the size of the texture using filter modules (for example, downsize) or allocate more swap space, main memory, or hardware texture memory.

X Server Color

On 24-plane true color systems, each pixel can have one of 256 red x 256 green x 256 blue (16,777,216) color values. There are 8 bits to represent red tones, 8 bits for green tones, and 8 bits for blue tones. The red, green, and blue tones combine to create the actual pixel color.

On 8-plane pseudo color systems, each pixel can have one of 216 color values. There are 6 red tones, 6 green tones, and 6 blue tones. To display a true color image on an 8-plane pseudo color device, AVS takes the original red value for each pixel and finds the closest numeric value from among the 6 reds available. It does the same for green and blue.

AVS then takes the pixmap that is made up of these three best-matches and applies a dithering algorithm to the pixmap. Dithering uses the fact that the human eye will interpolate between dots of color, creating the impression of a color value between two actual color values. The dithering process corrects for information lost in the 256-to-6 reduction by comparing how far off each final pixel value was from the original value against a dithering matrix. Some pixel values have their red/green/blue values adjusted up or down to create a closer approximation to the original true color image. This might sound very limited, but the end result is surprisingly satisfactory.

The Image Viewer and **display image** module have an option that turns off dithering on 8-plane systems when no change to the original image appearance is desired.

Note: The IBM 24-bit High Performance Adapter only supports the 8-plane pseudo color X visual under some OS levels. See the IBM chapter below for more information.

Warning: reducing color usage

On some 8-plane systems, the applications already running on the workstation may have already allocated so many cells in the 256 entry colormap that there are not enough for AVS to use 6 tones for red, green, and blue. In this case, you will see a message like the following when AVS starts up:

```
Warning: reducing color usage:  R-5, G=5, B=5, Grey=17
```

indicating that AVS is reducing its colormap cell usage.

If you get this message on a truecolor system, it means you have the wrong default visual. See the "Visual Type" section later in this chapter.

Dark Display: Gamma Correction

On some workstations, the AVS interface (all non-graphics windows) may appear too dark. You can lighten the interface using the **-gamma** command line option, or the **Gamma** *.avsrc* startup file option. For example, in your personal *.avsrc* file:

```
Gamma 1.7
```

You will have to experiment to find a satisfactory value. Values between 1.7 and 2.2 are good starting points for experimentation. Higher real values produce a lighter display.

How to Check the Default Visual

You can check to see what the default visual is. Type the */usr/bin/X11/xdpyinfo* command and look for the "default screen number" line for your screen:

```
name of display:    :0.0
version number:    11.0
vendor string:    Silicon Graphics
     .
     .
     .
default screen number:    0        <--- the relevant line
number of screens:    1
```

Next, find the description information of the default screen and look for the "default visual id" line.

```
screen #0:
  dimensions:    1280x1024 pixels (340x270 millimeters)
  resolution:    96x96 dots per inch
  depths (6):    1, 2, 4, 8, 12, 24
  root window id:    0x2d
  depth of root window:    8 planes
  number of colormaps:    minimum 1, maximum 8
  default colormap:    0x2b
  default number of colormap cells:    256
     .
     .
     .
  number of visuals:    8
  default visual id:  0x20        <---- the relevant line
```

Now, look down the visual definitions until you find the hexadecimal visual id code that matches. This is the default visual that the X server and AVS will use.

```
  visual:
    visual id:    0x23
    class:    PseudoColor
    depth:    2 planes
    size of colormap:    4 entries
    red, green, blue masks:    0x0, 0x0, 0x0
    significant bits in color specification:    8 bits
     .
     .
     .
  visual:
    visual id:    0x20          <---- the relevant line
    class:    PseudoColor
    depth:    8 planes
    size of colormap:    256 entries
    red, green, blue masks:    0x0, 0x0, 0x0
    significant bits in color specification:    8 bits
     .
     .
     .
```

This is an 8-plane pseudo color visual, as specified by the "class" and "depth" lines.

You need to tell AVS to use a different visual. Continue to look down the list of visual definitions until you find one with a "class" of true color and a "depth" of 24 planes. Note its "visual id", in this case 0x29.

```
   .
   .
   .
visual:
  visual id:    0x29
  class:    TrueColor             <---- A 24-plane true color visual
```

```
depth:    24 planes
size of colormap:    256 entries
red, green, blue masks:    0xff, 0xff00, 0xff0000
significant bits in color specification:    8 bits
  .
  .
  .
```

You will see 12-plane true color visuals, and various others such as StaticColor. Do not use these visuals.

Changing the Default Visual for AVS

To start AVS using the correct visual, make the following addition to your personal *.avsrc* file:

```
VisualType  VisualID n
```

Replace *n* with the hexadecimal visual id of the correct true color visual.

```
VisualType  VisualID 0x29
```

With AVS 5.5, there is also a new command line option, *-vistype*, which provides the same functionality at runtime. See the section above on "Unexposed command line options".

Window Manager

AVS works with any of the standard X Window System window managers, including *mwm*, the Motif window manager; *dxwm*, the DECwindows window manager; *4Dwm*, the SGI IRIX window manager; HP VUE; and the Common Desktop Environment (CDE) window manager. No user modifications are necessary in order to use AVS with these window managers.

However, there are some modifications that you may want to make as a matter of personal preference. The following examples are more oriented towards Motif, but similar changes can be made to the other window managers.

Click to Type

By default Motif is a "click to type" window manager. It is not enough in all cases to simply move the mouse cursor into an input window such as an AVS file browser typein panel or the Transformation Options panel and begin to type. You may need to click on the left mouse button first to make the AVS window receive events.

To disable "click to type" in Motif, add or modify the following line to your personal *.Xdefaults* file to match what is shown here:

```
Mwm*keyboardFocusPolicy:   pointer
```

(On some systems, the "k" in keyboard may need to be an uppercase "K".) You may need to make additional changes to ensure that AVS gets all button events. Consult your Motif documentation.

In the HP VUE window manager, to disable "click to type" behavior, follow this selection sequence:

```
Style Manager
  Window
     FocusFollowsMouse
         OK
```

For CDE window managers, the sequence is this:

```
Style Manager
  Window
     Click in Window To Make Active
         OK
```

Intercepting Mouse Events

In a manner conforming to the interaction style of the older window managers under which AVS was originally developed, AVS makes heavy use of all mouse buttons. Under Motif, the window manager may try to intercept mouse button events, particularly the left mouse button, for its own purposes such as making a window the current window and automatically raising it. If you have any lines like the following in your *.mwmrc* file:

```
<Btn1Down>        frame|icon|window       f.raise
```

You may want to change them to:

```
<Btn1Down>        frame|icon              f.raise
```

to ensure that AVS gets the mouse button events that occur within its windows.

Close Function

Pop-ups menus include a **Close** function. However, selecting this **Close** function is actually a command to kill the window, not to just iconify (minimize) or unmap its window from the screen. Using the

Close function on many AVS windows will either not work, or will put AVS into an unuseable state. You should use the various **Close** buttons on AVS's windows instead.

Window Decoration

If you do not want all AVS windows to be surrounded with the Motif window border controls, put this line in your *.Xdefaults* file:

```
Mwm*avs*clientDecoration:none
```

Layout Editor

The Layout Editor relies on some knowledge of which window manager you are using to reliably determine window positions for some operations. If you have a problem in which windows move unexpectedly when selected in the Layout Editor, use the *window_mgr* CLI command to inform AVS which window manager you are using. For more information see Chapter 5 in the *AVS Developer's Guide.*

X Terminal and Remote Display Support

In most cases, you can run AVS as a remote X client on one UNIX workstation from another workstation with a color monitor and an X server that supports at least an 8-bit PseudoColor visual. You can also run AVS on a UNIX workstation from a color X Terminal. The X Terminal's X server must also support at least an 8-bit PseudoColor visual.

avs -nohw Required

Except as noted below for specific platforms, you must start AVS using the *-nohw* command line option or the **NoHW 1** keyword in your personal *.avsrc* file. This will force the use of AVS's software renderer. Without this option, AVS will attempt to initialize the hardware renderer when you enter the Geometry Viewer the first time and will fail.

The software renderer will perform 3D rendering into an X image and send the image to the X server for display. Color rendition will be up to the capabilities of the local X server: pseudo color on X servers with a PseudoColor visual, and true color on X server that support a TrueColor visual. See the "AVS on Color X Servers" appendix in the *AVS User's Guide* for more information.

Remote hardware rendering

In some cases when you are rendering from one hardware platform to display on another using the same hardware graphics library you may not need to rely on the software renderer.

- OpenGL: The OpenGL library is capable of running as a distributed graphics subsystem, even across different platforms, as long as the remote system is configured appropriately, providing GLX support in the X server.

- HP: When you are running between HP PHIGS workstations, you do not need to specify *-nohw*. The HP VMX (Virtual Memory X) facility will emulate remote hardware rendering transparently.

- SGI: The SGI OpenGL library is capable of running as a distributed graphics subsystem. This means that, in the special case where you are running AVS on an SGI workstation from *another SGI workstation*, you will still be able to use the hardware renderer and take advantage of its superior rendering speed. AVS on the remote system will send OpenGL library instructions to the local graphics subsystem to be executed. No special steps are needed to make this happen.

- IBM: If you get the following error message when you start AVS:

```
gl: gversion:  1345-072  The requested X Windows Server extension is not available
```

  then you need to set the AVS_GRAPHICS environment variable to **xterm**.

  csh users would set this environment variable as follows:

  ```
  setenv AVS_GRAPHICS xterm
  ```

  ksh or *sh* users would set this environment variable as follows:

  ```
  AVS_GRAPHICS=xterm
  export AVS_GRAPHICS
  ```

- Compaq Tru64 UNIX: The **-nohw** command line option or the **NoHW 1** *.avsrc* keyword should not be necessary as AVS automatically detects the graphics adapter type and initializes the appropriate renderer(s).

Stereo Support

Support for stereo viewing is now provided for SGI (N32 and N64), Sun SunOS5 (Solaris) (OpenGL and XGL), Compaq Tru64 Alpha, and HP Workstations (Visualize-FX4 and -FX6 graphics adapters only). In this release, stereo support on SGI will support either "full screen" or "stereo-in-a-window" depending on the capabilities of your platform. Stereo on Solaris, Compaq Tru64 Alpha, and the HP workstations also supports the "stereo-in-a-window" approach. Note: IBM and Linux have not been tested but may work.

Stereo is supported using CrystalEyes stereo goggles from Stereo-Graphics. One source for these is Qualix Direct, found at *http://www.qualixdirect.com*. You must consult the vendor's documentation for information on how to connect this device to your workstation.

Stereo Display

Stereo display is controlled from the Geometry Viewer's **Cameras** submenu.

To turn on stereo:

- Toggle the **Stereo** button on the **Cameras** submenu. You may have to scroll to make this button visible.
- Type **Ctrl-left mouse button**.

The entire screen is taken over by the stereo view of the object when using "full screen" stereo; "quad-buffered" or "stereo-in-a-window" only affects the Geometry Viewer windows. You will not see stereo unless you have the CrystalEyes stereo goggles.

To exit stereo mode, type **Ctrl-left mouse button** again or use the Camera/Stereo toggle button. **Note:** You should exit stereo mode before leaving AVS or it may leave stereo enabled. If you do leave stereo on by accident, restart AVS and toggle stereo on/off to leave stereo mode; or use the "off" stereo command as shown in the stereo documentation below to exit stereo manually.

Enabling Stereo

Each platform has a different way of enabling and disabling stereo mode. You should see the appropriate platform chapter for more information.

Most platforms enable stereo in the X server configuration. For the SGI, you can see the current settings for on and off commands using

two debug environment flags:

```
setenv AVS_STEREO_CMDS_ARE 1
setenv AVS_OGL_INFO 1
```

In order to override these values, there are two environment variables to set the "on" and "off" commands:

- **AVS_STEREO_ON_CMD** specifies the command that enables stereo mode
- **AVS_STEREO_OFF_CMD** specifies the command that disables stereo mode

Because AVS does NOT "remember" your previous mode before it entered stereo, you may want to at least set the AVS_STEREO_OFF_CMD variable.

Stereo Adjustment Parameters

In some cases, you may wish to make minor adjustments in the stereo control parameters controlling apparent eye separation between the two viewpoints or the apparent distance between the eyes and the focal point. These are controlled differently depending on which platform you are using.

On platforms supporting OpenGL stereo, the stereo parameters are accessible through a GEOM CLI command, *geom_set_stereo_params*.

```
geom_set_stereo_params  Sets the stereo viewing parameters
     Usage: geom_set_stereo_params -eye <V> -dist <V> -near <V> -far <V>
```

This command ultimately controls the values being used in setting up the stereo viewing matrix.

- eye: This is the "eye offset" controlling the apparent eye separation from the mid plane (nose). Used along with a fixed constant in setting the left and right vertical clipping planes. Initial value can also be set using the environment variable called AVS_STEREO_EYE_OFFSET; set by default to 0.025.
- dist: Distance of the eye from the center of the space being viewed. Initial value can be set using the environment variable AVS_STEREO_EYE_DIST; set by default to 2.0.
- near/far: Specify distances to the near and far depth clipping planes; both distances must be positive. Initial value can be set using the environment variables AVS_STEREO_NEAR and AVS_STEREO_FAR; set by default to 1.0 and 6.0 respectively.

The source code for the internal function that uses these values shows exactly how they are used to make a *glFrustum* call in OpenGL to create the perspective viewing matrix.

```
void
stereoFrustum(GLfloat near, GLfloat far,
              GLfloat eyeDist, GLfloat eyeOffset)
{
    GLfloat eyeShift = (eyeDist - near) * (eyeOffset / eyeDist);
    GLfloat limit = 0.425; /* from Express routine to try to match stereo */

    glFrustum(eyeShift - limit, eyeShift + limit, -limit, limit, near, far);
    glTranslatef(-eyeShift, 0.0, 0.0);
}
```

With no parameters ("geom_set_stereo_params"), the current values will be displayed. Use of either or both parameters will modify the current values. Further information on CLI commands can be found in the *Command Line Interpreter* chapter of the *AVS Developer's Guide.*

Japanese, Greek, and Cyrillic Labels in Geometry Viewer

A font type called **Kanji** has been defined for labels in the Geometry Viewer. Kanji labels are supported in the software renderer using X window fonts. The X font pattern for Kanji is:

```
-jis-fixed-medium-r-normal-*
```

Use the **xlsfonts** program to discover if your X server has any matching fonts installed. Typically, you will find one or more point sizes for the JIS X 0208 1983 character set. Though named "Kanji," this character set can also represent all of the following fonts:

- Roman, Greek, and Cyrillic alphabets
- Hiragana and Katakana (Japanese syllabaries)
- JIS Level 1 and Level 2 Kanji (Chinese characters used in Japanese)

At present, there is no way to interactively type in or edit Kanji test in the **Labels** typein. However, Kanji strings will be displayed correctly in the **Labels** typein if the label is selected.

The font options **Bold** and **Italics** will not have any effect for these Kanji fonts.

Label strings can be specified in JIS, Shift-JIS, or EUC encodings. These are extended two-byte character codes.

The labels are typeset horizontally reading from left to right.

You input the text by coding the test into a user module with the GEOMadd_label function.

Here is a sample code fragment using an EUC coding for the three kanji characters. Each character is assigned two octal codes.

```
char kanji_string[] = { "306374313334270354" };
int font_number, label_flags;
GEOMobj *kanji_obj;

font_number = GEOMget_font_number( "Kanji", 0, 0 );
label_flags = GEOMcreate_label_flags( font_number, remaining params );
kanji_obj = GEOMcreate_label( GEOM_NULL, label_flags);
GEOMadd_label( kanji_obj, kanji_string, remaining params, -1 );
```

Consult your platform's release notes chapter to see whether Kanji labels are supported in your hardware renderer.

Programming Considerations

There are some issues that are important for developers to know for all platforms.

Use the Example Makefile as Template

You should use the makefile in *<installdir>/avs/examples/Makefile* as the template for your own FORTRAN and C module makefiles. This makefile in turn includes the file *<installdir>/avs/include/Makeinclude* that contains additional macro definitions appropriate to the platform you are using.

GEOMint_color

The GEOM library treats **integer** and **integer color** primitive data in a similar way, so these types should be the same length (32-bits). A new type has been defined for integer colors in the GEOM library which users should use for portability in both 32 bit and 64 bit platforms. For the C binding, **GEOMint_color** is defined in *geom.h* to be **unsigned int** on 64-bit platforms, such as Compaq Tru64 UNIX and SGI N64 and **unsigned long** on 32-bit plaforms, such as AIX and Solaris. FORTRAN should use INTEGER*4 in these cases. **Note:** Only programmers writing modules that use the *libgeom.a* library should be affected by this change; new compiler warnings may arise if function prototypes are used (C++ or Ansi-C compilers).

The programming interface to the following functions has been modified to accommodate this new type:

- GEOMcreate_mesh_with_data
- GEOMcreate_polyh_with_data
- GEOMcreate_sphere
- GEOMset_color
- GEOMset_pickable
- add_to_vlist_from_int
- create_vlist_from_int

C++ Support

Users writing C++ modules should use a special version of the subroutine module library called *$AVS_PATH/lib/libflow_C.a*. This library uses a *main()* function that has been compiled under C++, thereby including C++ initialization calls that were not invoked using *libflow_c.a*. Subroutine modules making use of C++ stream I/O functions must use this library to work properly, while other C++ modules may not require it.

The C++ examples in *$AVS_PATH/examples* should build directly in most environments. The *Makefile* and *Makeinclude* files were changed to use the most common names and paths for the C++ compilers and libraries. Read the Makefile for more specific information and examples.

FORTRAN Modules on 64-bit systems

On 32-bit platforms, integer (int) and pointer (int*, void*, etc) types are the same size and have historically been used as if they were the same. On most 64-bit platforms (Compaq Tru64 UNIX and SGI N64) pointers are 8 bytes and integers are 4 bytes; the two data types can NOT be used interchangeably without causing problems (bizaare values, 0's, etc.)

Portability issues and tools

Many AVS FORTRAN functions pass pointer values which have usually been declared as FORTRAN INTEGER variables on 32 bit platforms; FORTRAN does not have a standard "pointer" data type. On the Compaq Tru64 UNIX and SGI N64, these pointers MUST be declared as INTEGER*8 (8 byte) variables to avoid the module crashing with bad data. This primarily affects compute function arguments (input and output data pointers for complex data types like fields) and array offset values used in some data access functions such as

AVSfield_data_offset.

There is no standard data type to handle pointers on both architectures so truly portable FORTRAN code is not possible. One solution is to use C-like macros for the "pointer" data type and a macro preprocessor but this is not commonly used in FORTRAN and is not universally available on all platforms.

Instead, AVS 5 relies upon a set of "tags" to mark pointer variable declarations and a utility program to convert source code from 32-bit to 64-bit compliant. This allows portability to be managed but does not rely upon a compile-time macro preprocessor.

AVS 5.5 provides the source for the tools that have been used internally for converting files. These utility shell scripts are in the *$AVS_PATH/examples/f77_tools* directory in the Compaq Tru64 UNIX and SGI N64 platforms. They consist of a makefile, a generic converter program (*f77_converter.c*) and a shell script to apply the tool to a file tree (*convert_f77*). For historical reasons, the "tag" value used in AVS is "ALPHA_OSF" but other values can be used in the shell script. **NOTE:** These tools only provide one possible way you may address the portability problem - you are not required to use them, particularly if you do not plan on porting your code.

As an example, let's look at the sample FORTRAN file *CHEMelestf.f* in */usr/avs/examples/chemistry*.

You'll notice that all integer variables that are really pointers in disguise are declared as INTEGER*8.

```
C %IFDEF ALPHA_OSF
      integer*8 mol_input,ep_field
C %ELSE
C     integer mol_input,ep_field
C %ENDIF ALPHA_OSF
```

AVSfield function argument changes

IMPORTANT NOTE: Because of problems encountered with 64-bit pointers under SGI N64, several functions needed to be modified for 64 bit systems. In particular functions which pass "array offset" values had to be modified to use INTEGER*8 arguments to be able to hold large enough offset values. The following changes were made and affect the Compaq Tru64 UNIX and SGI N64 platforms ONLY.

```
AVSfield_data_offset( field, offset, basevec)
AVSfield_points_offset( field, offset, basevec)
AVSload_byte(base, offset)
AVSstore_byte(base, offset, value)
AVSload_short(base, offset)
AVSstore_short(base, offset, value)

   offset is INTEGER under 32-bit platforms, INTEGER*8 under 64-bit platforms
```

Compaq Tru64 UNIX data offsets

Due to the way the FORTRAN compiler generates code for subscripting, the trick with **AVSfield_data_offset** and **AVSfield_points_offset** doesn't work. The compiler does support %VAL() for function arguments, so the more straightforward **AVSfield_data_ptr** and **AVSfield_points_ptr** can be used.

```
C %IFDEF ALPHA_OSF
      call load_pnts(%val(AVSfield_points_ptr(ep_field)),
     1                            res,extents,xinc,yinc,zinc)
C %ELSE
C     iresult=AVSfield_points_offset(ep_field, coords, ocoords)
C     call load_pnts(coords(ocoords+1),res,extents,xinc,yinc,zinc)
C %ENDIF ALPHA_OSF
```

Fixed in AVS 5.5

The following bugs have been fixed in the AVS 5.5 release. Those affecting multiple platforms are covered here. Additional information may be found in the platform specific chapters.

User Interface Issues

7389/7517: AVS Message popup window should be a regular window

Problem Description:

The popup message window used to support the AVSmessage, AVSwarning, AVSerror, and AVSfatal information calls did not have window decorations or other conventional window manager access, preventing it from being iconified, moved, resized, or reordered amidst other windows. It has been changed to work like all the other standard windows.

7260: Typein widgets don't handle strings over 64 characters well

Problem Description:

Typein widgets will handle value strings of arbitrary length but do not scroll or resize dynamically with longer values.

Workaround: The typein widget will generally size itself to at least contain the initial value string being used. Setting the default to a long string initially will stretch it out. Using the width property is another way to control the size to make it large enough for the desired value.

17567: Demos: AVS5 Features/AVSGraph menu empty

Problem Description:

This submenu was still referencing an discontinued version of AVSGraph from an earlier release. It has been updated to reference the current AVSGraph module's sample scripts.

AVS Kernel, LibFlow or CLI issues

7824: image_list_image_names command returns error incorrectly

Problem Description:

The *image_list_image_names* CLI command added in AVS 5.4 always returned an error code at the end incorrectly. This has been fixed.

17213: Demo script information window clipping comments

Problem Description:

The demo viewer script controller window and the network editor script controller window were clipping long comments from view. This has been fixed.

17441: Demos leave some viewer windows behind after exit

Problem Description:

Many of the demos create geometry viewers and their associated windows while they run, and have only cleaned up these windows after a full run is completed and the clean up commands are run at the end of the script. This would tend to leave around unused windows after incomplete demos. This has been fixed so that starting a new script or leaving the AVS Demos application will cause a general cleanup of left over geometry and image viewer windows.

7728: AVSdata_alloc allocates fewer bytes than needed

Problem Description:

In some cases, the AVSdata_alloc routine allocated fewer bytes than were needed for user data structures, depending on alignment issues on different platforms. The allocation size has been padded to cover these differences.

8064: Some AVS modules ignore the AVS -path option

Problem Description:

AVS 5 uses the AVSpath value to determine its home location and this can be set in various ways - command line option, environment variable, etc. While the kernel itself worked with the AVSpath value and communicated it to the external module processes, many of the modules paid no attention to it, so they relied on finding AVS in the default */usr/avs* location.

This has been fixed by modifying all AVS 5 supported modules to use the value of the AVSpath variable when they need to know where AVS 5 is. Modules affected include the Geometry Viewer, the Data Viewer, vbuffer, and AVSGraph among others.

7322: UCD_TRIANGLE definition missing from FORTRAN include files

Problem Description:

The UCD_TRIANGLE definition was missing because of a bug in the FORTRAN interface program, *f77_binding*. This has been fixed and the definition is included in the *avs.inc* files for AVS 5.5.

17656: geom_save_postscript documentation missing argument

Problem Description:

The **geom_save_poscript** CLI command does not mention how or where it wants to get a filename argument, leaving the user to guess. It expects the final argument "filename" to come after the preceding -option phrases. The builtin help has been updated to reflect this.

Geometry Viewer issues

16962: Stereo picking inaccurate

Problem Description:

Picking in stereo was problematic because it was using the non-stereo transformation matrix to determine what was being picked. This has been improved.

16967: Stereo view different from original non-stereo view

Problem Description:

When switching to stereo, a new transformation matrix must be used for the new view. The matrix used substantially changed the appearance of the scene. This has been fixed by adjustments in the stereo transformation matrix that much more closely match the original non-stereo view.

Licensing Issues

7793/7812: Developer's AVS: AVS-Runtime encryption problems

Problem Description:

AVS 5.4 and earlier releases used a different encryption code for runtime licensing than what was used in the standard licenses. With the upgrade to FlexLM 4.x in the AVS 5.3 upgrade, support for this second encryption code was lost causing problems with existing licenses in the field and temporary difficulty making working licenses. This has been fixed for AVS 5.5 and offers the Developer customer two options:

1. AVS 5.4 and earlier: AVS-Runtime features use the alternate encryption code and must be run using the old FlexLM (2.4c) AVS lmgrd daemons provided with AVS5.02. AVS5.3 and AVS5.4 use these codes but their lmgrd daemons won't recognize them.

2. AVS 5.5 and AVS 5.4 patch: AVS-Runtimes are generated using AVS 5.5 or AVS 5.4 with a patch available from AVS Customer Support. They use the same encryption codes as standard AVS, and must run using FlexLM 4.x or higher daemons. AVS5.5 and beyond will make this the default code. Existing AVS 5.4 or earlier runtime licenses should be reissued by Customer Support.

If you are affected by this issue, please contact AVS Customer Support (support@avs.com) for more information or new licenses.

11139: Demo_license Y2K problem

Problem Description:

The demo_license program used for creating temporary licenses for users evaluating AVS 5.4 was using 2-digit years. The resulting licenses still appeared to work, but the program has been upgraded to handle 4-digit years. This issue did not affect the permanent licenses given to registered users.

17568: Animator and BTF Renderer need to be unlicensed

Problem Description:

Both the Animator module and its supporting modules, and the BTF renderer modules have been unlicensed and are now available to all users.

8529: Module Generator puts 'struct' in user data declarations

Problem Description:

The Module Generator erroneously was adding the "struct" keyword to user data declarations, causing compiler errors and requiring manual editing. This has been fixed.

7822: Arbitrary slice plane may use wrong field extents

Problem Description:

The arbitrary slice module selects a plane size based on the XY extent of the object being viewed. Depending on the object, this plane may not be large enough as it is rotated around within the overall dimensions of the object - for example, an oblong brick object with the XY cutting across the small dimension of the brick would use a small square slice plane that won't extend across the length of the brick.

This has been fixed by adding a new choice parameter to the module called "Slice Size" with the following choices:

- XY slice: use the XY slice as the plane size
- YZ slice: use the YZ slice as the plane size
- ZX slice: use the ZX slice as the plane size
- Diagonal slice: use the diagonal of the extents (across opposite corners) to determine the size of the plane.

3428: Set View module doesn't work properly on SGI platforms

Problem Description:

The set view module was producing a bad transformation matrix on the SGI N64 platform causing the geometry viewer to show nothing in some views. This has been fixed.

14838: read_ucd does not recognize tab delimited files

Problem Description:

The read_ucd module did not recognize the use of tabs as delimiters in data files. It has been fixed to recognize these as it does other white space characters such as space and end-of-line.

7426: Module Generator produces incorrect FORTRAN code for UCD calls

Problem Description:

The Module Generator was producing references to "UCDstructure_x" calls based on the C naming convention rather than "UCDstruct_x" as used by the FORTRAN API. It was also using the function call convention ("call UCDstructure_free") rather than the subroutine calling convention (result = UCDstructure_free()"). Both of these problems have been fixed.

7738/7825: AVSGraph needs AVS_PATH to be set

Problem Description:

The AVSGraph module needs to find some underlying Toolmaster definition files in the AVS home directory. When AVS was not installed in */usr/avs* it would fail to run at all. This has been fixed so that AVSGraph checks the definition of the AVSpath variable to find the home directory.

7745: AVSGraph does not remove temp file

Problem Description:

The AVSGraph module was always writing debugging information to a log file in the */tmp* directory that was named AVSGraph.log<processid> which it left around when it exited. AVSGraph has been modified to NOT write out any file unless the user indicates that the log file is desired by defining the AVS_AGX_DEBUG environment variable to any value.

7756: AVSGraph writes to unnecessary temp file

Problem Description:

The AVSGraph module was always writing debugging information to a log in the /tmp directory named AVSGraph.log. When multiple users tried to use AVSGraph, the file was written out owned by the first user, then subsequent users would fail because they couldn't gain write access to the same file. The file naming convention has been changed to AVSGraph.log<processid> to always create unique names to avoid this problem.

16167: UCD_Legend typein parameter override

Problem Description:

Difficulty was reported using typein parameters for 'value', 'lo value' or 'hi value'. Whenever a parameter or the radio dials change, the module compute routine ensures that the 'radio dials' are displayed correctly and are in sync with the parameter widget dials and legend scale.

The problem occured when the parameters were given a typein value instead of using the dials. The value was scaled to the radio dial, which was then scaled to the data, which then over rode the

typed in value, preventing the user from controlling the precision as desired.

This has been fixed by preventing this circular update when the user provides typed in data.

16417: Minmax module crash

Problem Description:

The minmax module was crashing with repeated use due to memory not being reallocated correctly. This has been fixed.

17422: Color legend: Text disappears when increasing thickness

Problem Description:

When modifying the thickness parameter, the spacing of the text below or to the side of the color legend bar was adjusted proportionately, often sending the text off the screen for a thick bar. The spacing calculation has been changed to not reflect the thickness but to make some adjustment for the font height instead.

13262: geom_set_camera_name has no effect on window decoration

Problem Description:

The **geom_set_camera_name** CLI command can be used to set the name of geometry viewer cameras, which are names like "Camera 1" by default. However these did not have any visible impact on the user interface and were of limited value.

AVS 5.5 has been modified to display the Camera name as part of the standard window decoration. Changes made using this command are immediately reflected in the window title and icon names, making it easier to identify cameras. Also see the new "Geometry Camera" module documented under "New Features" that provides a camera "wrapper" similar to the main view window associated with the "Geometry Viewer" module, providing Layout Editor capability, pull down menu, close-window resistence, etc.

Known Problems

8195: Inconsistent ambient light coeeficient

Problem Description:

The default coefficient for ambient light (hardware mode) on GL platforms is 0.2, and the default object coefficient (software mode) for ambient light is 0.3; this results in brighter objects when switching from hardware renderering to software rendering.

8549: Blank lines in CLI scripts cause problems

Problem Description:

Empty lines in CLI scripts may cause problems during script playback. Under some circumstances the CLI may interpret the previous line alright but then get stuck on the blank line. AVS is still interactive and the user can abort or interrupt the script. Remove empty lines or use the "#" leader to indicate the line is a comment line.

14421: Shared memory segment left after AVS exits

Problem Description:

AVS allocates an initial shared memory segment to act as an index of all the other shared memory segments used by the kernel and module for passing Field and UCD data around. On exit, this segment may be left in use as various modules are in the process of exiting and cleaning up the data memory segments. Sometimes this results in this index memory segment being left around after all other processes have completed, something you may see when using the *ipcs -m* command. It will be reused by the next AVS session by the same user so it is not a cumulative problem. If desired you can manually remove this segment using the *ipcrm* command.

17399: AVS Demos - Network Editor 'exit' button may cause crash

Problem Description:

When running the AVS Demos (main menu, Applications, Demos), the user can exit using the pull down menu Control/Return-Exit button or using the "exit" button that may appear on the left panel with module user interfaces. The second approach does not exit as cleanly as the first and in some animation demos, may cause AVS to crash. Please use the pull down menu approach instead.

AVS 5.5 FOR COMPAQ TRU64 UNIX ALPHA

CHAPTER FOUR

Introduction

This chapter describes AVS 5.5 as it runs on Compaq Tru64 UNIX Alpha AXP workstations (note Compaq Tru64 UNIX was previously called Digital UNIX or OSF/1 and is equivalent). It covers hardware and software prerequisites needed to run AVS; differences and special considerations for running AVS on this platform; and known problems.

AVS5.5 NOTE: The supported operating system has been updated from Compaq Tru64 UNIX 4.0B in AVS 5.4 to 4.0D; the supported version of OpenGL has also been updated from 4.4 to 4.7. Stereo support has also been added for this platform, providing quad-buffered or "stereo in a window" support.

NOTE: Additional information on using AVS under X windows, general programming considerations, and bugs fixed in AVS 5.5 can be found in the *AVS 5.5 on UNIX Platforms* chapter above.

Hardware Prerequisites

Workstation Models

This release AVS runs on Compaq 3000 Model 400 AXP, Compaq 3000 Model 500 AXP, and Compaq 4000 AXP computers using Compaq Tru64 UNIX 4.0D.

Graphics Hardware

This release of AVS uses two mechanisms to produce its screen renderings of Geometry Viewer objects and scenes:

- a **software renderer** that implements a set of graphics primitives (polygons, lighting model, perspective, shading, color, etc.) in software, rendering the resulting image into an X Window

System image.

The **software renderer** will work on any supported workstation model with an 8- or 24-plane color frame buffer, the XPutImage *xlib* call, and a local X server that supports them.

- a Compaq Tru64 UNIX OpenGL **hardware renderer** that uses the Compaq Open3D graphics library and will take advantage of any Compaq Corporation graphics hardware using shared libraries.

  The OpenGL **hardware renderer** will work with any graphics adapter supported by Compaq Open3D, particularly the ZLX and Powerstorm graphics adapters. For more information, see the section on Compaq Tru64 UNIX OpenGL support below.

In most cases, AVS automatically detects which graphics subsystem is present and initializes the correct renderer(s).

See the subsequent section on "Software Prerequisites" for the system software needed to be able to use the hardware rendering capability.

Memory

This version of AVS requires a minimum of 32 megabytes of physical memory to run properly. More real memory will improve performance. In addition to real memory requirements, there is a minimum system swap space size requirement that is described below.

Total System Memory

To see how much memory is on your system, use the *uerf -R | more* command as root. This lists the system boot log. Look for lines like the following:

```
physical memory = 126.00 megabytes.
available memory = 106.23 megabytes.
```

The first figure is the total system memory, the second is the amount remaining after resident pages of the operating system have been loaded at boot time.

Memory Per Process

Compaq Tru64 UNIX systems can restrict the amount of memory available to any given process on a per-shell basis. There is both a "hard" system limit, and a "soft" system limit.

To see the "soft" limit, enter this *csh* command:

```
limit
```

This produces an output like the following:

```
cputime       unlimited
filesize      unlimited
datasize      131072 kbytes   <-- memory limit
stacksize     2048 kbytes
coredumpsize  unlimited
memoryuse     unlimited
descriptors   4096
```

The same command, entered as root, shows the system hard limits.

Generally, the system default is ample. If it is not, you can increase the limit for the duration of the shell by typing, for example:

```
limit datasize 500000
```

which increases the limit for that shell to 500000 kbytes. To increase the system's hard limit, use the *doconfig* command described in the Compaq publications *Guide to Installing Compaq Tru64 UNIX* and *Guide to System Administration*.

Disk Space

It requires approximately 71 megabytes of disk storage to install AVS from the release CD. More disk storage may be required for swap space (see below).

Swap Space

Recommended Swap Space Size

Many of the algorithms used in AVS modules are memory intensive. Scientific datasets, especially image and volume data, can grow quite rapidly. For example, while a single 64x64x64 element scalar byte field may take up 256 KB of memory, a 256x256x256 element scalar byte field takes up 16 MB, and a 512x512x512 element field takes up 128 MB.

It is recommended that a swap space of at least 128 MB be allocated on systems on which AVS is installed.

Determining Swap Space Size

You can determine the current memory utilization of your Compaq Tru64 UNIX system with the */usr/sbin/swapon -s* and *ps aux* commands.

Executing the following command will show you the the total amount of swap space available, as well as the amount of swap space which is reserved and free:

```
/usr/sbin/swapon -s
```

Executing the following command will give you information about the swap space (VSZ) and current swapped-in memory size (RSS) assigned to each process:

```
ps aux
```

Pre-allocated vs Use-as-Needed Swap Space

Compaq Tru64 UNIX systems can use one of two swap space allocation schemes. In "pre-allocated," the system must have available sufficient swap space to hold all processes running in memory at once. In "use as needed," swap space can be "overcommitted." This means that you can add the amount of memory to the amount of swap space to find the total space available to the process.

The default is pre-allocated. You can change this to use-as-needed. However, as noted below, an OS with use-as-needed swap space turned on may kill a process that requests more swap space than is available. This may be AVS, or it might be a critical system process whose demise could hang the system.

To switch from pre-allocated to use-as-needed, as root simply move the *swapdefaults* file aside and reboot the system:

```
mv /sbin/swapdefault  /sbin/swapdefault.old
sync
sync
reboot
```

Recognizing When You Have Run Out of Swap Space

When you run out of main memory or swap space, the symptoms can be varied.

- If AVS was making a memory allocation request when the shortage occurred, then you may receive an error message box from AVS to this effect:

```
"Failure allocating memory:
See Installation/Release Notes to increase swap space
and/or shared memory segment size"
```

or possibly a message in the terminal emulator window from which AVS was started like:

```
name@system [8] swap space below 10 percent free
Unable to obtain requested swap space
geom_file.c line 528, 5760 bytes
GEOM fatal error:  malloc fails allocating vert list
```

- If some other part of the system (such as the X server) was making the memory allocation request when the shortage occurred, then you will usually receive an error message such as "Not enough memory," or "Running out of swap space" in the terminal emulator window from which AVS was started.

When use-as-needed is enabled, in some cases, no error message will appear. Instead, the X server becomes extremely slow. Indeed, the OS may kill the process causing the swap. If this is a critical system process, such as the X server, the system may hang.

For help on configuring your system's swap device, refer to the *swapon* man page and the Compaq Tru64 UNIX *Advanced Installation Guide* and *Guide to Disk Maintenance.*

Software Prerequisites

Operating System: Compaq Tru64 UNIX

AVS 5.5 will run on version 4.0D of the Compaq Tru64 UNIX operating system; it will likely not run on earlier versions.

The *uname -r* command will show the current revision that is running on your workstation.

Graphics Software

OpenGL

Software Requirements:. If you will be doing hardware rendering, then you must have installed:

- Compaq Open3D version 4.7 (or later) for Compaq Tru64 UNIX on the system where AVS will *execute*. OpenGL is part of the Compaq Open3D package. AVS 5.5 was built and tested using

version 4.7 and may not run reliably under earlier versions. To be able to use OpenGL, you must obtain and register an LMF PAK for Compaq Open3D.

- An OpenGL capable X server on the system where AVS will *display*.

If the system upon which AVS will *display* is not OpenGL capable, then AVS will only initialize its software renderer.

If the system upon which AVS will *display* is OpenGL capable, then AVS will try to initialize its OpenGL renderer. If it cannot find Compaq Open3D and no LMF PAK is registered, then AVS will fail. In this case, you must explicitly start AVS with the "no hardware rendering" option (*avs -nohw*) so that AVS will only initialize its software renderer. (See the "Starting AVS" chapter of the *AVS User's Guide* for more information.)

Finding Out if Software is Installed and Licensed. To find out whether Compaq Open3D is installed on your system, you can type:

```
setld -i | grep Open3D
```

and look for the appropriate OpenGL entries.

To find out whether Open3D software is licensed, as root type:

```
lmf list all
```

Again, look for the appropriate entries.

Performance. AVS 5.5 relies on shared libraries to access OpenGL so in most cases it will take advantage of the local graphics accelerator hardware. In some cases, OpenGL may not fully support the graphics hardware you are using and will not perform substantially better than the software renderer. Review the Compaq Open3D Release Notes to see which hardware is supported and what restrictions may apply to your system. In most installations, the release notes can be found in */usr/lib/OPEN3D*. The release notes can also be found online at *http://www.digital.com/info*. This web site can also be used to download the Open3D software.

Compaq OpenGL 4.7 issues. These are several issues reported in the Compaq Open3D 4.7 release notes that may affect AVS 5.5 users.

Front vs Back Orientation of Polygon Faces:

Problem Description:

According to the Compaq Open3D Version 4.7 Release notes, there are cases where Open3D will miscompute whether a

polygon is front facing or back facing. According to section 2.5.10 of the release notes, the workaround is to edit the /usr/lib/X11/Xserver.O3D.conf file and insert the following line:

```
DoFrontBackTestTheOpenGLWay  True
```

RGB Format:

Problem Description:

On all PowerStorm 4D40T, 4D50T, 4D60T options running under UNIX, there is also an 8-plane TrueColor visual *not* supported by OpenGL. This visual is in "BGR" format where the least-significant 2 bits are the blue component, the next 3 bits are the green component, and the most significant 3 bits are the red component.

Kanji Support. If you wish to use Kanji labels in the Compaq OpenGL hardware renderer you will need these system font files from the Compaq OpenGL/Japanese software subset:

```
font_jisx0201.fnt
font_jisx0208.fnt
```

They should be installed in directory:

```
/usr/lib/X11/strokefonts/pex
```

If you try to render Kanji strings without either font file, you will see raw 16-bit character codes rendered in a standard 8-bit font. If you only have one of the files installed correctly, you may get the following X error:

```
X Error of failed request:
  PEXFontError, specified font ID invalid
  Major opcode of failed request:  139 (X3D-PEX)
  Minor opcode of failed request:  88 (PEX_OpenFont)
```

Stereo

Compaq Tru64 UNIX Alpha now supports "stereo-in-a-window" stereo viewing. For general information on stereo support in AVS 5.5, see the "AVS 5.5 on UNIX Platforms" chapter. The following platform specific stereo information is from the OpenGL release notes:

Supported Hardware

This stereo implementation requires a PowerStorm 4D40T, 4D50T, or 4D60T graphics adapter and compatible MultiSync monitor, such as Digital's VRC15, VRC17, or VRC21 series monitors. StereoGraphics and NuVision stereo hardware are both supported by this implementation.

Note that the stereo connector on the back of the PowerStorm 4DT cards is different from the stereo connector on the back of other Digital graphics adapters. Contact the manufacturer of your stereo hardware to obtain the proper cables to connect your stereo hardware to your PowerStorm 4DxxT card. The cable should match the mini-DIN connector on the back of the PowerStorm card.

Supported Screen Resolutions and Refresh Rates

In order to support normal aspect ratio pixels in stereo mode, the frame buffer is divided into two full height left and right buffers. Because of the increased memory requirements associated with maintaining separate left and right buffers, the maximum screen resolution for each graphics adapter will be slightly smaller than in non-stereo mode.

Maximum Stereo Resolutions

PowerStorm Model	Planes Per Pixel	Max Resolution
4D60T	102	1280x1024
	128	1152x900
4D50T/4D40T	102	800x600
	128	800x600

To select a particular screen resolution and vertical refresh rate, use the '- screen' and '-vsync' command line options in the X server configuration file. To select an additional vertical refresh rate to be used when the screen is in stereo mode, use the '-I -e3stereo <rate>' command line arguments in the Xserver configuration file. Replace <rate> by one of the vertical refresh rates in the table below. Note that the maximum refresh rates indicated may not be supported by all hardware.

If you omit the '-I -e3stereo <rate>' arguments in the Xserver configuration file, the server will continue to use the same refresh rate that it was using when the screen was in monoscopic mode. This will be the value that was specified with the '-vsync' parameter, or the default refresh rate for this resolution.

The first time that the screen is switched into stereo mode, the Xserver will print out the refresh rate that is being used for stereo. If no '-

vsync' or '- e3stereo' arguments were given, the Xserver will note that stereo mode is using the default refresh rate. This information can be found in the Xserver error log (*/var/dt/Xerrors* or */var/X11/xdm/xdm-errors*).

See your system administrator or refer to your documentation and release notes for more information on configuring the Xserver via the configuration file.

Shared Memory Usage

We recommend that your UNIX kernel be configured for at least 64 shared memory segments per process and that the maximum size of each segment to be 32MB. This can be done by increasing the shmseg system parameter to be at least 64, and the shmmax system parameter to be at least 33554432 (32MB). These parameters should be changed in your system configuration file. After the changes have been make to the configuration file, the UNIX kernel will need to be rebuilt, moved into the system area, and the machine will need to be rebooted.

For help on reconfiguring the UNIX kernel, refer to your UNIX *Guide to System Administration* manual or your *Guide to Installing Compaq Tru64 UNIX*.

Using AVS on Compaq Tru64 UNIX

The following sections describe in detail the differences between using AVS on a Compaq Tru64 UNIX Alpha AXP workstation and AVS as it is described in the standard documentation.

Peripherals

Dial Box

The Dial Box I/O device mentioned in the *AVS User's Guide* is not supported in this release of this product.

Spaceball

The Spaceball device is supported.

Colormap Sharing

Colormap sharing can be a problem if you are using a pseudo color visual type and have hardware rendering enabled. The current AVS system can only run if it can get 135 color table entries from the default X window colormap. If you are running other applications that make heavy use of the color map, this may prevent AVS from

being able to start up. In particular, on 8 bit systems, you will be unable to run multiple AVS sessions simultaneously using the same display.

Renderers

See the "Graphics Software" section under "Software Prerequisites" section above for information on what software must be present to run different renderers.

Image Viewer

All functions and behavior in the Image Viewer are as described in the AVS *User's Guide.* The Image Viewer will use as many bits of color as are available on the graphics adapter.

If you are using Compaq ZLX hardware, consult the "Graphics Software" section above for important information about the Image Viewer.

Network Editor

The Network Editor functions as described in the AVS *User's Guide* and *Module Reference* manual, with these few exceptions:

Modules

The following modules documented in the *AVS Module Reference* manual do not appear in the AVS release for Compaq Tru64 UNIX because they require rendering features that are not supported in this release:

alpha blend (requires hardware-assisted alpha transparency)
transform pixmap (requires 2D texture mapping)

Remote Module Execution

This release of AVS 5.5 on Compaq Tru64 UNIX does not currently support DECnet remote module execution. Conventional remote module execution using "rsh" as described in the AVS *User's Guide* is supported however.

Image Output

While using a hardware renderer, if you are using the **image to postscript** module to get output from the **geometry viewer** module, or any of the Animation Application output modules such as **write frame seq**, all parts of the Geometry Viewer scene window must be visible. No parts of the display window can be off the screen or obscured by

other windows. If they are, any module downstream of **geometry viewer** will not receive the off-screen or obscured portion of the pix-map.

Graph Viewer

There are no differences between AVS Graph Viewer behavior on Compaq Tru64 UNIX and the Graph Viewer descriptions in the AVS documentation.

Geometry Viewer

The Geometry Viewer (or the **geometry viewer** module) will be using either the software renderer or one of the two hardware renderers to produce the contents of its scene windows.

Rendering Features

The following table lists the rendering features described in the "Geometry Viewer" chapter of the *AVS User's Guide*, including new AVS features, down the left column, and the AVS software renderer, and the Compaq OpenGL hardware renderer across the top. The table intersections show which features are present on each platform, and draw your attention to more detailed explanations of behavior later in this section.

Table 4-1. Geometry Viewer Behavior

Rendering Feature	Software Renderer	Hardware Renderer (OpenGL)
Arbitrary Clip Planes	yes, 8 planes	no (note 1)
Geometric		
Volume Rendering	yes (note 2)	no (note 2)
Vertex Transparency	no (note 3)	no (note 3)
Vertex Colors	yes (note 4)	yes
Edit Property		
RGB/HSV Colors	yes (note 4)	yes
Ambient	yes	yes
Diffusion	yes	yes
Specular Highlights	yes	yes
Gloss	yes	yes
Transparency	yes (note 5)	yes
Metallic	yes	yes
Edit Texture	yes	no
2D Texture Mapping	yes	yes (note 8)
3D Texture Mapping	yes	no
Filtered Textures	no	yes
Tile Textures	no	no
Alpha Textures	yes	no
Rendering Options		
Points	yes	yes
Lines	yes	yes
Smooth Lines	no	no
No Lighting	yes	yes
Outline No Lighting	yes	yes
Flat Shading	yes	yes
Outline Flat	yes	yes
Gouraud Shading	yes	yes
Outline Gouraud	yes	yes
Phong Shading	no	no
Backface Properties		
Cull front	yes	no
Cull back	yes	yes
Flip normals	yes	no
Lights		
Number of Sources	8	8
Ambient	yes	yes
Directional	yes	yes
Bi-Directional	yes	yes, up to 3
Point	yes	yes
Spot	no	yes
Light Colors	yes	yes

Table 4-2. Geometry Viewer Behavior Across Platforms (continued)

Rendering Feature	Software Renderer	Hardware Renderer (OpenGL)
Cameras		
Depth Cue	yes	yes
Global Anti-Alias	no	no
Accelerate	no	no
Perspective	yes	yes
Axes for Scene	yes	yes
Front/Back Clipping	yes	yes
Sorted Transparency	no	no
Polygonal Spheres	yes (note 6)	no
Double Buffer	no	yes (note 7)
Stereo	no	yes
Labels		
Drop Shadow	no	no
Title	yes	yes
Stroke	no	yes
Kanji	yes (note 8)	no

Notes

1. Arbitrary clipping planes are used by the **clip geom** module.

2. "Volume Rendering" is here defined in a very narrow sense: support of a specific option to the **GEOMedit_texture_options** call that is used by the **volume render** module. The fact that the hardware renderer does not support this form of volume rendering only means that it cannot use one of the features of the **volume render** module; it does not mean that the hardware renderer cannot render volumes of data using other rendering techniques.

3. Vertex transparency is used by the **colorize geom** module.

4. When using the software renderer, the number of colors that will appear on the screen (216 pseudocolor, 16,777,216 true color, or some number in between) is dependent upon the X server visual support present on the *display* hardware. Internally, 24-bit true color is always used and will be output on the **geometry viewer** module's image output port.

5. The software renderer supports single-pass transparency. Single-pass transparency correctly renders a transparent surface that is over an opaque surface. (Multi-pass transparency is required to correctly render multiple overlapping transparent surfaces.)

6. A feature in the software renderer is the ability to render spheres directly instead of subdividing them into polygons. This saves a tremendous amount of memory when rendering spheres. The algorithm's execution speed is bound by the size of the spheres it has to render. This feature may be disabled by toggling the **Polygonal Spheres** button under the Cameras menu. Spheres will now be rendered by subdividing them into polygons. Double Buffering is supported on both types of hardware renderer, and is on by default.

7. Label color may be affected by the use of 2D texture mapping.

Programming Considerations

AVS, its modules and component libraries were compiled using the Compaq C compiler (5.6-071) and the standard Unix C libraries located in */lib/libc*. Its FORTRAN components were compiled using the Compaq FORTRAN 77 compiler (5.1).

Compiling and Linking Modules

Note: Please see the information on "GEOMint_color" in Chapter 3, *AVS5.5 on UNIX Platforms* for an issue that will affect modules using the GEOM library on 64 bit platforms.

Use the Example Makefile as Template

You should use the makefile in *<installdir>/avs/examples/Makefile* as the template for your own FORTRAN and C module makefiles. This makefile in turn includes the file *<installdir>/avs/include/Makeinclude* that contains additional macro definitions appropriate to the Compaq Tru64 UNIX platform.

FORTRAN Modules

On 32-bit platforms, pointer and integer data types are the same size and have historically been used as if they were the same. On most 64-bit platforms (Compaq Tru64 UNIX and SGI N64 (*SGN64*)) pointers are 8 bytes and integers are 4 bytes; the two data types can NOT be used interchangeably. See the section on "FORTRAN Modules on 64-bit systems" in the *UNIX Platforms* chapter for critical information on this topic.

AVS on Compaq Tru64 UNIX: Known Problems

This is the list of known problems for AVS 5.5 on the Compaq Tru64 UNIX platforms.

Problem: Label color affected by 2D texture mapping

Problem Description:

Use of 2D Texture mapping in a view may affect the color of labels in the same view.

X10051: va_args does not work with doubles

Problem Description:

Compaq Tru64 UNIX va_args doesn't work with doubles. AVS uses va_args for **AVSmessage** functions, so modules using **AVSerror** to print out doubles will have problems on this platform.

X10277: FORTRAN field modules may dump core

Problem Description:

When you use a module written in FORTRAN that uses **AVSFIELD_POINTS_OFFSET**, if AVS is using shared memory for field data transfer, then the module will core dump. Shared memory transfer of data for fields is the default.

Workaround 1: Run AVS with -noshm.

Workaround 2: Use %VAL as an alternate technique for writing FORTRAN modules. For an example of how to rewrite your FORTRAN modules to use this technique, look at the example module in *$AVS_PATH/examples/test_fld2_f.f*. On Compaq Tru64 UNIX systems, this example will be compiled so that the **use_offset** parameter switches the code between the %VAL and the **AVSFIELD_POINTS_OFFSET** examples. When you are using shared memory transfer, when you toggle the **use_offsets** button on the module control panel, the module will dump core.

X10301: Module Generator FORTRAN should use integer*8

Problem Description:

The Module Generator fails to provide special %IFDEF statements for pointer parameters like the example FORTRAN modules do or to directly use INTEGER*8. Example:

```
      integer function colorizer2(in_field, cmap, out_field)
C %IFDEF ALPHA_OSF
      integer*8 in_field, cmap, out_field
C %ELSE
```

```
C      integer in_field, cmap, out_field
C %ENDIF ALPHA_OSF
```

X10766: Certain float fields can crash AVS

Problem Description:

A floating point exception can be generated by several standard field modules in response to extreme data values which should be still representable by the machine's floating point specification.

AVS on Compaq Tru64 UNIX: Fixed Problems

No platform specific problems were fixed for AVS 5.5 on the Compaq Tru64 UNIX platforms.

AVS 5.5 FOR HEWLETT PACKARD

CHAPTER FIVE

Introduction

This chapter describes AVS 5.5 as it runs on Hewlett Packard HP9000/7xx workstations using HP-UX 10.20 (HP1020). It covers hardware and software prerequisites needed to run AVS; differences and special considerations for running AVS on this platform; and known problems.

AVS5.5 NOTE: HP-UX 10.20 now supports OpenGL (1.1) as the default renderer, an addition since AVS 5.4; read this chapter for more information on this new renderer and the features it supports, including stereo. PHIGS is still supported in a secondary *avs.phigs* **kernel, which was compiled against a updated version of PHIGS (3.40). The default setting for** *avs* **is to enable the hardware renderer; previous releases came up with** *-nohw* **set by default. HP-UX 10.10 is no longer supported; if you require this platform, please request AVS 5.4 from support@avs.com or your local sales office.**

NOTE: Additional information on using AVS under X windows, general programming considerations, and bugs fixed in AVS 5.5 can be found in the *AVS 5.5 on UNIX Platforms* chapter above.

Hardware Prerequisites

Workstation Models

AVS will execute on all of the workstations in the HP 9000 7*xx* series.

Graphics Hardware

This release of AVS uses two mechanisms to produce its screen renderings of Geometry Viewer objects and scenes: a **hardware renderer** that uses the native graphics hardware and an interface graphics library (either OpenGL (default) or PHIGS); and a **software renderer** that implements a set of graphics primitives (polygons, lighting model,

perspective, shading, color, etc.) in software, rendering the resulting image into an X Window System image.

The default *avs* executable supports the OpenGL hardware renderer. A second *avs.phigs* executable supports PHIGS.

The **software renderer** will work on any supported HP workstation model with an 8- or 24-plane color frame buffer, the XPutImage *xlib* call, and a local X server that supports them. This includes the PVRX graphics adapter.

The **hardware renderer** will work with the CRX, CRX24, CRX24Z, CRX48Z, HCRX-8, HCRX-8Z, HCRX-24, HCRX-24Z, Visualize-8, Visualize-24, Visualize-48, TVRX and Visualize-FX{2,4,6} graphics adapters as well as possibly other built-in graphics adapters that come on HP platforms. See the subsequent section on "Software Prerequisites" for the system software needed to be able to use the hardware rendering capability.

To interactively determine which graphics adapter your HP workstation has, first determine which device file is in use by your primary X display as follows:

```
grep /dev/ /usr/lib/X11/X0screens | grep -v #
```

then type the following command using the device file that is returned, e.g. /dev/crt:

```
graphinfo /dev/crt
```

or

```
graphinfo /dev/crt_hyper
```

The following is a partial listing of sample output from the *graphinfo* command. In this example, the graphics adapter is a VISUALIZE-FX4.

```
HEWLETT PACKARD WORKSTATION GRAPHICS CONFIGURATION

PRODUCT INFORMATION

    graphics product:          HP VISUALIZE-FX4 (Rev. A)
    driver name:               hpvisx
    archive library:           /opt/graphics/common/lib/libddvisx.a
    shared library:            /opt/graphics/common/lib/libddvisx.sl
                               /opt/graphics/OpenGL/lib/libddvisxgl.sl
    device pathname:           /dev/crt
    spu description:           9000/782

CONFIGURATION INFORMATION
```

```
image planes:              48
overlay planes:            8
resolution:                1280 X 1024
color or grayscale:        color
Powershade installed:      yes
PHIGS supported:           yes
hardware accelerator:      yes
geometry accelerator:      yes
image accelerator:         no
texture accelerator:       yes
hardware zbuffer:          yes
software zbuffer:          no
video out:                 yes
```

Memory

AVS requires that the HP 9000/7*xx* be configured with a minimum of 16 megabytes of main memory. More real memory, such as 32 or 64 megabytes, will improve performance.

To interactively determine the amount of memory that your HP workstation has, log in as *root* and type the command:

```
/etc/dmesg
```

This command generates a listing of various system messages produced during system startup. Look for a message similar to the one below in the command output. This message gives the amount of memory in your system in bytes.

```
Memory Information:
    Physical:  98304 KBytes ...
```

In addition to real memory requirements, there is a minimum system swap space size requirement that is described below.

Disk Space

It requires approximately 70 megabytes of disk storage to install either version of AVS 5.5 from the CD. More disk storage may be required for swap space (see below).

Swap Space

Recommended Swap Space Size

We recommend that swap space be a minimum of twice the amount of main memory or 64 megabytes, whichever is greater. Very large datasets and complex networks may require more than the minimum recommended swap space.

Determining How Much Swap Space You Have

To determine the current amount of swap space available on your system, log in as *root* and type the following command:

```
/etc/swapinfo -mt
```

This produces a listing like the following:

```
             Mb      Mb      Mb   PCT  START/      Mb
TYPE      AVAIL    USED    FREE  USED   LIMIT RESERVE  PRI  NAME
dev         512       3     509    1%       0       -    1  /dev/vg00/lvol2
reserve       -      39     -39
memory      188     130      58   69%
total       700     172     528   25%       -       0    -
```

The last line lists the totals across all swap partitions. This system has over 700 meg of swap space.

Increasing System Swap Space

If the amount of swap space currently available on your system is inadequate, follow the following procedure to augment the existing swap space with file system swap space.

1. Log in as *root* and invoke the System Administration Manager by typing:

   ```
   sam
   ```

2. Select the entry labeled *Disk and File Systems* -> from the main menu. Then select the *Swap* -> entry. A panel appears listing the mount directories and their "Total Kbytes."

```
Device File/                Total      Current     Enable  Enable
Mount Directory     Type    Kbytes     Priority    Now     on Boot

/dev/dsk/c201d5so   dev      71766     0           y       y
/dev/dsk/c201d6so   dev     101855     0           y       n
```

3. Select *Add File System Swap* under the *Actions* menu. A new panel appears listing the "Unused Kbytes" and the "Total Kbytes," as well as the mount of point for the disks.

4. Edit the *Maximum Swap* typein to increase the amount of file system swap in your system.

5. Press the *OK* button to execute your request.

Software Prerequisites

HPUX

AVS 5.5 will run on the HP-UX 10.20 operating system; it will not run on earlier versions. This platform appears as *HP1020* in the installation menu.

The *uname -r* command will show the current revision of HP-UX that is running on your HP workstation.

OpenGL

The default kernel, *avs*, requires that the OpenGL libraries be installed on the target machine. AVS was built using OpenGL 1.1. You can determine which version of OpenGL you have by typing the following as root user:

```
/usr/sbin/swlist | grep OpenGL

B6196AA_APZ B.10.20.01 HP-UX 700 OpenGL 3D Graphics API Run Time Environment
PHSS_15838  B.10.00.00.AA OpenGL 1.1 Revision 1.07 Developers patch
PHSS_15839  B.10.00.00.AA OpenGL 1.1 Revision 1.07 Runtime patch
```

End users of AVS require the Run Time Environment to use the OpenGL kernel.

Older systems may not be OpenGL compatible or perform as well with OpenGL as they do with the PHIGS interface. If you don't have OpenGL installed check with Hewlett Packard to determine what is the best interface for your configuration. If you have neither PHIGS nor OpenGL installed, use the *avs.phigs* kernel and the dummy libraries noted below with "-nohw".

PHIGS and PowerShade

Hardware Rendering: PHIGS and PowerShade Required

The *avs.phigs* renderer supports hardware rendering on the graphics adapters mentioned previously. In order to access this funtionality, AVS requires that version 3.40 or later for HP-UX 10.20. PowerShade must also be installed on the workstation.

You can determine which version of PHIGS you have by typing the following as root user:

```
/usr/sbin/swlist | grep PHIGS
```

These products are not included with base HP workstations and may need to be ordered separately. However, all of the "Z" adapters (e.g., CRX24Z, CRX48Z, HCRX-8Z, etc) have PowerShade built in. To verify that PowerShade is available on your system, select "General" from the "Toolboxes" pop-up menu on the HP VUE panel, and then select "System Info".

If the PHIGS library is not present and hardware rendering is enabled in *avs.phigs*, the following message will appear when AVS is started:

```
/lib/dld.sl: Can't open shared library: /opt/graphics/phigs/lib/libphigs.sl
/lib/dld.sl: No such file or directory
Terminated
```

Software Rendering: Dummy PHIGS Library Required

If you do not have the PHIGS runtime library and PowerShade, AVS will still work with the software renderer since it does not make any PHIGS library calls.

However, prior to running AVS using the software renderer, you need to supply a dummy library that will masquerade as the PHIGS library to satisfy the dynamic loader at AVS startup. We supply such a dummy library in *<install-dir>/lib/libphigs.sl*.

Login as root and copy:

```
mkdir -p /opt/graphics/phigs/lib
cp  $AVS_PATH/lib/libphigs.sl  /opt/graphics/phigs/lib/libphigs.1
ln -s /opt/graphics/phigs/lib/libphigs.1 /opt/graphics/phigs/lib/libphigs.sl
```

Then, make *libphigs.1* world readable and executable.

In this case, you should start AVS using the *-nohw* command line option or include the **NoHW** keyword in your personal *.avsrc* file. If you don't do this, you may see a cryptic message about not being able to find the "popph" function.

Error Message: Invalid version for shared library

The revision date of the */opt/graphics/phigs/lib/libphigs.sl* library must be greater than or equal to the date of the PHIGS library AVS Inc. used to link the AVS application. If you somehow run afoul of this limitation, this is the error message you will see:

```
/lib/dld.sl:  Invalid version for shared library:  /usr/lib/libphigs.sl
/lib/dld.sl:  Exec format error
IOT trap
```

and AVS will exit.

Using AVS on an HP 9000/7*xx* Workstation

The following sections describe in detail the differences between using AVS on an HP workstation and AVS as it is described in the standard documentation.

Peripherals

Dial Box

The Dial Box I/O device mentioned in the *AVS User's Guide* is not supported in this release of this product.

Spaceball

The Spaceball device is supported.

Colormap Sharing

Colormap sharing can be a problem if you are using a pseudo color visual type and have hardware rendering enabled. This affects the CRX graphics adapter and the built-in graphics adapters.

There are several steps that need to be taken in order to have AVS, the HP X server and OpenGL or PHIGS share the same colormap.

For Both Hardware and Software Rendering

By default, HP VUE will allocate about forty colors from the default colormap that AVS will try to share. If a message similar to the following appears when AVS starts up, it means that AVS was forced to use less colors than is optimal:

```
Warning: reducing color usage: R=5,G=5,B=5,Grey=17
```

In order to reduce VUE's color usage, so AVS can acquire all the colors that it would like, follow this selection sequence:

```
Style Manager
  Color
     HPVue Color Use
        Low Color
           OK
```

Then, logout and login again.

Next, you must also use the Edit Resources action to add the following line to your resource database:

```
*shadowPixmaps    True
```

To pick Edit Resources, follow this selection sequence:

```
Toolboxes
  General
    System_Admin
      Edit Resources
```

In order to have these resources take effect properly, all clients (e.g. *hpterms*) must be killed and you must logout and log back in.

Hardware Rendering Only

There is an environment variable that needs to be set to tell applications to share the existing colormap, if possible, instead of defining their own. The following line should be inserted in the X configuration file */usr/lib/X11/vue/Vuelogin/Xconfig*.

```
Vuelogin*environment:    SB_X_SHARED_CMAP=true
```

After doing this, the X server needs to be restarted so the variable will take effect. The easiest way to restart the X server is to select *Restart Server* from the *Options* menu on the logon screen. See the chapter "The HP A1659A (CRX) and HP A1924A (GRX) Devices" in the *HP PHIGS Workstation Characteristics and Implementation* manual for an in depth discussion of this environment variable.

All Rendering

Finally, AVS needs to have its internal gamma correction disabled so it can share the colormap with X and OpenGL or PHIGS. This can be done by using the *-gamma* command line option or the **Gamma** keyword in your personal *.avsrc*. In either case, the variable should be set to *1.0*.

A side of effect of disabling AVS's gamma correction is that the user interface in AVS becomes a little dim. To correct this, start AVS with the *-class X* command line option or edit the *avs.Xdefaults* file as described in the "AVS on Color X Servers" appendix of the *AVS User's Guide*.

Enabling Double Buffering on Builtin Graphics Boards

If you have an HP system with a builtin graphics board (e.g., 705, 710, 715, 725, etc.) and you want to use hardware rendering, you will want to enable double buffering. This will give you access to full 8/8 double buffering using virtual memory.

Set the following environment variable to enable this feature:

```
setenv HP_VM_DOUBLE_BUFFER  TRUE
```

There are two tradeoffs that should be considered when enabling this feature: speed and memory usage. Virtual memory rendering uses pure software algorithms. As a result, rendering can be significantly slower. Virtual memory rendering also consumes a part of memory that would otherwise be available to the application. As a result, you should carefully consider whether it might just be more effective to use AVS's software renderer.

Starting AVS

The only issues here that differ from the account in the AVS *User's Guide* and the *avs* man page are the default selection of the software renderer and selecting the correct X server VisualType for your graphics adapter.

This involves making changes to your personal AVS startup file (*.avsrc*) in your HOME directory. If you don't already have a *.avsrc* file, copy the sample file from *<installdir>/avs/runtime/avsrc* to your home directory.

Renderers

The default kernel *avs* requires the OpenGL runtime libraries to be present to run. The *avs.phigs* renderer requires that PHIGS and PowerShade be present on the workstation for hardware rendering to work. These packages are not part of the base HP configuration and must be purchased from HP separately; AVS enables hardware rendering by default so if you do not have PHIGS available you need to use the dummy library and set NoHW on as follows. This is done with the following line in the AVS startup file that is located in *<installdir>/avs/runtime/avsrc*.

```
NoHW    1
```

If you have both PHIGS and PowerShade on your workstation and would like to use hardware rendering, they should work by default. **Note:** AVS 5.4 used NoHW as the default; AVS 5.5 enables hardware rendering.

Visual Type

No matter what X server visuals are available, AVS will always use the default visual. The default visual on all the supported HP graphics adapters is pseudo color. However, if you have one of the 24-bit graphics adapters (CRX24, CRX24Z, CRX48Z, TVRX, HCRX-24, HCRX-24Z) you will want to use either a 24-bit direct color visual or true color visual.

To see which 24-bit visuals exist for your display, type the command:

```
/usr/contrib/bin/X11/xdpyinfo
```

Note: this standard X utility may not be available on your HP.

To have AVS use a 24-bit direct color visual or true color visual that these graphics adapters support, add this line in your personal *.avsrc* file or use the new *-vistype* command line option documented above in the "Unexposed command line options" section.

```
VisualType      DirectColor  (or TrueColor)
```

If you do not, AVS will use a visual type that is only 8-bits when the graphics adapter is able to support a 24-bit visual.

Image Viewer

All functions and behavior in the Image Viewer are as described in the AVS *User's Guide.* The Image Viewer will use as many bits of color as are available on the graphics adapter.

Network Editor

The Network Editor functions as described in the AVS *User's Guide* and *Module Reference* manual, with these few exceptions:

Modules

The following modules documented in the *AVS Module Reference* manual do not appear in the AVS release for HP workstation because they require rendering features that are not supported in this release:

alpha blend (requires hardware-assisted alpha transparency)
transform pixmap (requires 2D texture mapping)

Remote Module Execution

The command to establish contact with a remote host to execute remote modules is different on HP workstations. Instead of using *rsh* as on most other systems, use *remsh* in the *hosts* file.

Image Output

While using the hardware renderer, if you are using the **image to postscript** module to get output from the **geometry viewer** module, or any of the Animation Application output modules such as **write frame seq**, all parts of the Geometry Viewer scene window must be visible. No parts of the display window can be off the screen or obscured by other windows. If they are, any module downstream of **geometry viewer** will not receive the off-screen or obscured portion of the pixmap.

Graph Viewer

Other than the fact that image data loaded as a background to a plot on 8-bit graphics adapters will be dithered to pseudocolor rather than appear in true color, there are no differences between AVS Graph Viewer behavior on HP workstations and the Graph Viewer descriptions in the AVS documentation.

Geometry Viewer

The Geometry Viewer (or the **geometry viewer** module) will be using either the software renderer or the hardware renderer to produce the contents of its scene windows.

Stereo Support

The default OpenGL *avs* kernel now includes basic support for stereo viewing using quad-buffered "stereo-in-a-window". The stereo viewing device supported is the CrystalEyes stereo goggles from StereoGraphics. See the "AVS 5.5 on UNIX Platforms" chapter for more general information.

HP stereo is only supported on the Visualize-FX4 and Visualize-FX6 graphics adapters. Stereo needs to be enabled as the root user using either the *SAM* System Administrator Utility or *setmon* or by making changes in the Boot ROM. More information on enabling stereo and selecting the right hardware and cables can be found in the following whitepaper on the HP web site:

http://www.hp.com/visualize/support/library/stereographics.pdf.

In order to enable stereo you need to run the *setmon* utility as follows:

```
/opt/graphics/common/bin/setmon
```

Select one of the "Stereo in a Window" monitor entries, usually 4 or 5 depending on your system. The utility will switch over to the new entry for 15 seconds to see if it is working and if you don't confirm it, it will revert to the original setting. It will automatically reset the X server so existing applications will be closed.

Once selected, you can enable stereo using the Geom Viewer / Camera / Stereo button or Control-Mouse-Left in the view window. If stereo is not enabled, enabled stereo will cause a shift in the view but will not cause any problems.

Rendering Features

The following table lists the rendering features described in the "Geometry Viewer" chapter of the *AVS User's Guide*, including new AVS features, down the left column, and the AVS software renderer and the AVS hardware renderer's PHIGS and OpenGL implementations across the top. The table intersections show which features are present on each platform, and draw your attention to more detailed explanations of behavior later in this section.

Table 5-1. Geometry Viewer Behavior

Rendering Feature	Software Renderer	PHIGS	OpenGL
Arbitrary Clip Planes	yes, 8 planes	no (note 1)	no (note 1)
Geometric			
Volume Rendering	yes (note 2)	no (note 2)	no (note 2)
Vertex Transparency	no (note 3)	no (note 3)	yes (note 3)
Vertex Colors	yes (note 4)	yes	yes
Edit Property			
RGB/HSV Colors	yes (note 4)	yes	yes
Ambient	yes	yes	yes
Diffusion	yes	yes	yes
Specular Highlights	yes	yes	yes
Gloss	yes	yes	yes
Transparency	yes (note 5)	yes (note 6)	yes (note 6)
Metallic	yes	yes	yes
Edit Texture	yes	no	yes
2D Texture Mapping	yes	no	yes
3D Texture Mapping	yes	no	yes
Filtered Textures	no	no	yes
Tile Textures	no	no	no
Alpha Textures	yes	no	no
Rendering Options			
Points	yes	yes	yes
Lines	yes	yes	yes
Smooth Lines	no	no	no
No Lighting	yes	yes	yes
Flat Shading	yes	yes	yes
Outline Gouraud	yes	yes (note 10)	yes
Gouraud Shading	yes	yes	yes
Phong Shading	no	no	no
Backface Properties			
Cull front	no	yes	no
Cull back	yes	yes	yes
Flip normals	no	no	no
Lights			
Number of Sources	16	16	8
Ambient	yes	yes	yes
Directional	yes	yes	yes
Bi-Directional	yes	yes	yes
Point	no	yes	yes
Spot	no	yes	no
Light Colors	yes	yes	yes

Table 5-2. Geometry Viewer Behavior Across Platforms (continued)

Rendering Feature	Software Renderer	PHIGS	OpenGL
Cameras			
Depth Cue	yes	yes	yes
Global Anti-Alias	no	yes (note 7)	yes
Accelerate	no	no	no
Perspective	yes	yes	yes
Axes for Scene	yes	yes	yes
Front/Back Clipping	yes	yes	yes
Sorted Transparency	no	no	no
Polygonal Spheres	yes (note 8)	no (note 9)	no
Double Buffer	no	no	yes
Stereo	no	no	yes (note 11)
Labels			
Drop Shadow	no	no	yes
Title	yes	yes	yes
Stroke	no	yes	yes
Kanji	yes	no	no

Notes

1. Arbitrary clipping planes are used by the **clip geom** module.

2. "Volume Rendering" is here defined in a very narrow sense: support of a specific option to the **GEOMedit_texture_options** call that is used by the **volume render** module. The fact that the hardware renderer does not support this form of volume rendering only means that it cannot use one of the features of the **volume render** module; it does not mean that the hardware renderer cannot render volumes of data using other rendering techniques.

3. Vertex transparency is used by the **colorize geom** module.

4. When using the software renderer, the number of colors that will appear on the screen (216 pseudocolor, 16,777,216 true color, or some number in between) is dependent upon the X server visual support present on the *display* hardware. Internally, 24-bit true color is always used and will be output on the **geometry viewer** module's image output port.

5. The software renderer supports single-pass transparency. Single-pass transparency correctly renders a transparent surface that is over an opaque surface. (Multi-pass transparency is required to correctly render multiple overlapping transparent surfaces.)

6. Screen door transparency is supported through PHIGS when hardware rendering is enabled in the *avs.phigs* renderer.

7. Global Anti-Aliasing of lines is supported in PHIGS. This is almost the same as the **Smooth Lines** rendering option, but it is on a per view basis instead of a per object basis.

8. A feature in the software renderer is the ability to render spheres directly instead of subdividing them into polygons. This saves a tremendous amount of memory when rendering spheres. The algorithm's execution speed is bound by the size of the spheres it has to render. This feature may be disabled by toggling the **Polygonal Spheres** button under the Cameras menu. Spheres will now be rendered by subdividing them into polygons.

9. Spheres are subdivided into polygons when using the PHIGS based hardware renderer.

10. Turn on **Front/Back Clipping** to remove rendering artifacts.

11. Stereo support requires Visualize-FX4 or Visualize-FX6

Programming Considerations

AVS, its modules, and component libraries were compiled using the HP-UX C compiler (10.32.05) and the standard Unix C libraries located in */lib/libc*. Its FORTRAN components were compiled using the HP-UX FORTRAN 77 compiler (10.20.09)

Note: Please see the information on "GEOMint_color" in Chapter 3, *AVS5.5 on UNIX Platforms* for an issue that will affect modules using the GEOM library on 64 bit platforms.

Module Generator

The AVS Module Generator automatically invokes an *xterm* window when you press **Compile** to compile a module. However, the */usr/bin/X11* directory that contains the *xterm* command is not in the default path given to new users on HP-UX. To use the Module Generator, you should add */usr/bin/X11* to your default path.

csh users would add a line like the following to one of their startup files, usually *.cshrc* or *.login*:

```
set path=($path /usr/bin/X11)
```

sh or *ksh* users would add a line like the following to one of their startup files, usually *.profile*:

```
PATH=$PATH:/usr/bin/X11
```

Compiling and Linking Modules

Use the Example Makefile as Template

You should use the makefile in *<installdir>/avs/examples/Makefile* as the template for your own FORTRAN and C module makefiles. This makefile in turn includes the file *<installdir>/avs/include/Makeinclude* that contains additional macro definitions appropriate to the HP platform. These include references to the libraries to be found in */opt/graphics* (see definitions for *PHIGSINC* and *PHIGSLIB* in Makeinclude).

Fortran

The way in which the offset returned by the field accessor routines, **AVSptr_offset** and **AVSfield_points_offset**, is used can cause segmentation faults. This is due to the fact that on the HP PA-RISC architecture, the address space is actually divided into four segments. Depending on the instruction being executed, a test may be done to see if a segment is being crossed. This can happen in AVS because the Fortran accessor routines were written in order to be portable to different platforms. See bug **X7787** in the section "AVS on HP Workstations: Known Problems" for a description and a workaround.

All of the Fortran code in AVS was compiled using the *+ppu* switch. This causes an underscore to be postpended to all definitions and references to externally visible symbols. This may be an issue if you are trying to use an existing library of Fortran routines without recompiling.

AVS on HP Workstations: Known Problems

This is the list of known problems for AVS 5.5 on the HP 9000/7*xx* platforms.

X7787: Failure using field accessor routines in Fortran

Problem Description:

The example *threshold_f.f* fails when trying to use the offset returned by **AVSptr_offset**. The same behavior is exhibited by the routine **AVSfield_points_offset**.

Workaround: Refer to the documentation in the *AVS Developer's Guide* on **AVSptr_offset.** Of the three ways illustrated to use the offset returned by the accessor routine, only the second fails. As a workaround, call a routine passing it the address returned by the offset routine. Once inside this routine, the data can be accessed

correctly.

X9736: Transparency not working on CRX24Z, CRX48Z, and TVRX

Problem Description:

avs.phigs supports all the HP platforms using PHIGS. While this provides a consistent interface across the graphics adapters, it does not allow access to some of the functionality on all graphics adapters. Alpha transparency is an example of this on the CRX48Z, CRX24Z and TVRX.

The PHIGS API only allows AVS to access screen door transparency. As a result, depending on the transparency level specified, a more or less apparent "screen door" pattern appears on the geometry.

X10207: HP CRX48Z objects get dimmer as become smaller

Problem Description:

The CRX48Z and CRX24Z sometimes exhibit a problem where lighting seems to get turned off as the object is made smaller. Lighting comes back on as the object is made larger again. A problem report has been filed with HP but no fix for the problem is available at this time.

X10640: Hardware rendering dependent on shared PHIGS library

Problem Description:

The PHIGS kernel *avs.phigs* depends on finding a compatible copy of the shared PHIGS library (*libphigs.sl*) in */usr/lib* even when the hardware renderer is not being used. If AVS does not find any shared PHIGS library it will print this error message and exit:

```
/lib/dld.sl: Can't open shared library: /usr/lib/libphigs.sl
/lib/dld.sl: No such file or directory
IOT trap
```

Workaround: This is covered in the "System Prerequisites" section above.

Problem: AVS/Graph statically linked

Problem Description:

The HP-UX 10.20 platform uses Toolmaster 7.1 for AVS/Graph support. During development of AVS 5.5, it was necessary to statically link in *libhpgfx.a* because of linker problems. Let support know if you have any trouble on different configurations for any reason (support@avs.com).

17547: HP: AVSGraph module may crash on exit

Problem Description:

In some HP-UX 10.20 configurations the AVSGraph module crashes when exiting, leaving a core file and listing a traceback; on other machines, it exits cleanly. This is still under investigation. It does not impact the utility of the module; repeated crashes will overwrite the same core file.

AVS 5.5 FOR IBM RS/6000

CHAPTER SIX

Introduction

This chapter describes AVS 5.5 as it runs on IBM RS/6000 workstations running AIX 4.3. It covers hardware and software prerequisites needed to run AVS; differences and special considerations for running AVS on this platform; and known problems.

AVS 5.5 NOTE: The supported operating system has been updated from AIX 4.2 to AIX 4.3.1 and corresponding changes have been made in the compilers and OpenGL and GL libraries used. Support for the dialsbox peripheral has been dropped due to IBM discontinuing the libraries we were using. No other significant platform specific changes have been made. AIX 4.2 is no longer supported under AVS 5.5 but is available as part of AVS 5.4. If you require this platform please contact support@avs.com or your local sales office.

NOTE: Additional information on using AVS under X windows, general programming considerations, and bugs fixed in AVS 5.5 can be found in the *AVS 5.5 on UNIX Platforms* chapter above.

Hardware Prerequisites

Workstation Models

AVS will execute on any of the RS/6000 workstations: 2*xx*, 3*xx*, 5*xx*, or 7*xx*-series models.

Graphics Hardware

The RS/6000 workstation should be equipped with one (or more) of the following color graphics adapter boards:

- POWER Gt4 and POWER Gt4x 3D Graphics Adapters. Both adapters have Z-buffers. Both adapters can be either 8- or 24-bit.

- POWER Gt4e Graphics Adapter. This is a special 8-bit version of the Gt4.
- POWER Gt1 or POWER Gt3 2D 8-bit Graphics Adapters.
- Gxt800 or Gxt1000 graphics adapters. It must have a Z-buffer.
- 8-bit Color Graphics Display Adapter.
- GTO 3D 24-bit Graphics Adapter with Z-buffer. (This same graphics adapter is called the Supergraphics Subsystem when purchased with the IBM RISC System/6000 Model 730.) This graphics adapter is referred to as the **GTO Graphics Adapter** in the remainder of this document.
- High Performance 3D 24-bit Color Graphics Processor with Z-buffer option. This graphics adapter is referred to as the **High Performance Adapter** in the remainder of this document.

The AVS 3-dimensional graphics rendering features available to you will vary depending upon which graphics adapter your RS/6000 workstation has installed. These variations are discussed in detail in the "Rendering Features" section below.

The High Performance Adapter also comes in an 8-bit color version, and in a version that is not equipped with a Z-buffer. (A Z-buffer is a hardware device that correctly removes hidden surfaces from objects in 3-dimensional scenes. The Color Graphics Display Adapter supports only a 2-dimensional graphics model, and thus has no Z-buffer option; while the GTO Graphics Adapter always has a Z-buffer.)

The GTO Graphics Adapter also comes in an 8-bit color version.

AVS can also run on these more limited adapters using AVS's **software renderer** facility, although with less 3D rendering performance. See the "Rendering Features" section below.

How to Find Out Which Adapter You Have

With one exception, it is possible to interactively determine which graphics board your RS/6000 has. As root, type the command:

```
lsdev -C -c adapter
```

This will produce a listing of various board options, including the graphics adapters.

The lines:

```
gda0      Available 00-02    Color Graphics Display Adapter
```

means that you have the low-end 8-bit 2D Color Graphics Display Adapter.

```
wga0      Available 00-0J    Power Gt1x Graphics Adapter
```

means that you have the Gt1 2D 8-bit graphics adapter.

```
ppr0      Available 00-03    POWER Gt3 Graphics Adapter
```

means that you have the Gt3 2D 8-bit graphics adapter.

```
hiprfmge0 Available 00-07    High Performance 3D Color Graphics Processor
hiprfmrv0 Defined   00-06    Color Graphics Video Card
hiprfmev0 Defined   00-06-01 24-bit Color Graphics frame buffer card
hiprfmzb0 Defined   00-06-02 24-bit Z-buffer option card
```

means that you have the low end ("Sabine") High Performance 3D 24-bit Color Graphics Processor with Z-buffer option. If the Color Graphics frame buffer card says 8-bit instead of 24-bit, or if the final 24-bit Z-buffer option card line is missing, then you must use the AVS software renderer option described in the "Rendering Features" section below.

```
hispd3d0  Available 00-07  High Speed 3D Graphics Accelerator
```

means that you have the high-end GTO Graphics Adapter. However, no message is produced that reveals whether you have an 8-bit or 24-bit GTO Graphics Adapter.

```
ppr0      Available 00-02    POWER Gt4 Midrange Graphics Adapter
pgr0      Defined   00-03    POWER Gt4 Midrange 8-bit Frame Buffer
pop0      Defined   00-04    POWER Gt 4 Midrange 24-bit Frame Buffer option
```

means that you have the POWER Gt4. If *just* pgr0, the 8-bit frame buffer, is listed then you have an 8-bit adapter. If both frame buffers are listed, then you have a 24-bit adapter.

```
ppc0      Defined   00-02-01 POWER Gt4x Midrange Graphics Adapter Turbo option
```

This additional message means that you have the POWER Gt4x graphics accelerator.

```
ppr1      Available 00-01    POWER Gt4e Graphics Adapter
```

means you have the special 8-bit version of the Gt4.

```
rby0      Available 00-03    7250 POWER GXT1000 Graphics Accelerator Model Z
```

means that you have the Gxt1000.

Machines can be equipped with more than one graphics adapter.

You might not see any of these messages. If you have an IBM RS/6000 workstation that supports color, then you can probably still run AVS with limited color performance and software graphics rendering.

Memory

AVS 5.5 requires that the RS/6000 workstation be configured with a minimum of 16 megabytes of main memory. More real memory, such as 32 or 64 megabytes, will improve performance.

You can tell how much memory your RS/6000 has with the following command:

```
lsattr -l sys0 -E
```

The "realmem" line in the output describes the amount of usable physical memory in Kbytes.

Swap Space

Recommended Swap Space Size

The AIX installation instructions usually recommend configuring an RS/6000 with a swap (paging) space size at least equal to the size of real memory on the root volume group, not to exceed 20% of the disk space available on that group. Other volumes also have default swap space values supplied. If you have large datasets and use many modules with many connections between them, or you are creating flipbook animations with many frames, this may not be large enough to run AVS.

We recommend at least 64 megabytes of swap space, or twice the amount of real memory, whichever is larger. With large scientific datasets and networks with many modules and connections, AVS may require more swap space. It is not unusual to need more than 100 megabytes of swap space.

Determining How Much Swap Space You Have

To see how much swap space you have, use the *smit* utility. The sequence of menu selections to follow is:

Physical and Logical Storage
Paging Space
List All Paging Spaces

This will show the amount of swap space defined on each physical volume and volume group, along with its percentage utilization.

Running Out of Swap Space: Process nnn killed: no swap space

AIX monitors swap space utilization. When it sees that it is running out, it finds the process using the most swap space pages and sends it a warning message. If the swap space situation does not improve, it then kills the process using the most swap space. This should be accompanied by an error message to the effect "Process *nnn* killed: no swap space". These error messages have numbers ranging from 917-031 through 917-064.

How to Increase System Swap Space

If AIX begins sending you swap space warning messages, or you find AVS terminating suddenly and your swap space allocations are marginal, you should begin to increase the available swap space in 20 megabyte increments using the *smit* utility and the instructions contained in the IBM manual *Getting Started: Managing RISC System/6000*.

Shared Memory Limitation: shmat: too many open files

The AIX Operating System restricts each process to 10 attached shared memory segments.

AVS modules use shared memory segments to store AVS field and UCD data. If a network contains more than 10 modules in the same process, then you may run into this limitation. As a rule of thumb, you can count connections between field/field, field/ucd, and ucd/ucd ports to see if you are approaching this limit. (Most mapper modules output geometries, which are not kept in shared memory segments and their output connections don't have to be counted.)

The symptom that this has occurred is an error message "shmat: too many open files" followed by an informative message. In the case of field data, AVS says "can't put field in shared memory. Using malloc and sockets as workaround." With UCD data, AVS says "UCD: shared memory allocation failed."

One of two things can happen at this point. If the downstream modules are in the same process as the module whose shared memory call has exceeded the limit, there is no performance loss. The modules

will pass data by passing pointers. If, however, the downstream modules are in a different process, then data will be passed by copying it to the AVS kernel, which then sends a copy of it to the downstream module. This latter approach will use more swap space and be somewhat slower to execute.

It is also possible that the process in which the modules are running (typically "mongo" or "ucd_multm" or "sv_multm") may die when the shared memory limit is reached. In this case, you will get an AVS fatal error message box.

To avoid either error, you can regroup the modules into different processes using the **Group** field in the AVS Module Editor. Then, write out the network and read it back in. AVS will create a separate process for groups that have different names. In this way, you can reduce the number of modules in a process until the shared memory limit is no longer reached. See the "Advanced Network Editor" chapter of the *AVS User's Guide* for more details on module regrouping.

There is another workaround that may help in some instances if the attached shared memory segment limit is being exceeded by non-builtin AVS modules. You can invoke AVS with the **-separate** command line option. This will force each module to run as a separate process. However, AVS will use more memory and need to swap more frequently.

Software Prerequisites

AIX

AIX 4.3

AVS 5.5 for the IBM RS/6000 was built and tested under release 4.3.1 of the IBM AIX Operating System. It is not binary compatible with AIX 4.2 or earlier versions of the operating system.

To see what base version of AIX you are running, you can issue the following command:

```
% /usr/bin/oslevel
```

If you have 4.3 or above you should see output like this

```
4.3.1.0
```

In all cases, make sure that you have the most recent (even post 4.3.1.0) drivers for your graphics boards. If you have any rendering problems, check with the IBM Defect Support Line for the most recent fixes.

Graphics Library: OpenGL

AVS 5.5 provides a default AVS executable called *avs* that uses OpenGL for hardware rendering. It provides accelerated graphics for some graphics adapters more than others. Older frame buffers are supported by OpenGL but performance may be slower than hardware rendering using the GL graphics library. AVS 5.5 provides a separate executable, *avs.gl*, discussed in the next section to deliver GL hardware rendering.

You can always determine which version of AVS 5.5 you are running using the "-version" command line option. The graphics library is listed at the end of the version string.

The OpenGL runtime environment is required for both hardware and software rendering. It was a licensed product available from IBM for AIX 4.1 and AIX 4.2; with the AIX 4.3 operating system release it is is now bundled with the operating system. If you do not already have OpenGL installed on your system, you will need to contact IBM to obtain the necessary runtime libraries.

To see if the OpenGL runtime environment is present, use the *lslpp* utility. Look for lines similar to the following:

```
OpenGL.OpenGL_X.rte.base   4.3.1.0    C    OpenGL Base Runtime Environment
OpenGL.OpenGL_X.rte.soft   4.3.1.0    C    OpenGL Soft Runtime Environment
```

Graphics Library: GL

AVS 5.5 provides an optional AVS executable called *avs.gl* that requires the presence of version 4.3 of the IBM AIX GL graphics library in order to initialize. If you normally use the GL executable, you may want to make it your default version of AVS as follows:

```
cd $AVS_PATH/bin
mv avs avs.ogl
ln -s avs.gl avs
```

You can always determine which version of AVS you are running using the "-version" command line option. The graphics library is listed at the end of the version string.

Hardware Rendering: could not load program avs

On the High Performance Adapter with Z-buffer and 24-bits, and on the GTO, Gt4, Gt4x, Gxt800, and Gxt1000 graphics adapters, AVS supports hardware based rendering. In order to access this functionality, AVS requires that the GL runtime library be installed on the workstation if you are using the *avs.gl* renderer.

Beginning with AIX 3.2, the GL graphics library was no longer bundled with AIXwindows. Now it is part of the separately-sold IBM X11_3D Program Product.

If you have not purchased the GL library and try to run AVS with hardware rendering enabled, you will get the following error message:

```
Could not load program avs
Could not load library libgl.a[shr.o]
Error was: No such file or directory
```

Software Rendering: Dummy GL Library Required

The GL kernel, *avs.gl*, will still run with the software renderer since it does not make any GL library calls. However, you will need to make a link to a dummy GL library that AVS has supplied to resolve the undefined symbols.

```
ln -s <install_dir>/avs/lib/libgl_stubs.a /lib/libgl.a
```

where <*install_dir*> is the directory where AVS is installed.

Using AVS on the IBM RS/6000

The following sections describe in detail the differences between using AVS on the IBM RS/6000 workstation and AVS as it is described in the standard documentation.

General

Color Interference

The Display Postscript copyright window produced by the

```
/usr/lpp/DPS/bin/copyright &
```

line in the default */usr/lpp/X11/defaults/xinitrc* executed when the X server is started can interfere with AVS's interface color scheme as defined in the <*installdir*>*/avs/runtime/avs.Xdefaults* file on some platforms. The symptom that this is happening is that the buttons on the

main AVS menu will appear as a series of blue-gray to gray stripes rather than as a series of dark gray to light gray stripes.

If you have these lines in your own *.xinitrc* file, you should comment them out as shown:

```
#if [ -z "$XSTATION" ]
#then
#  /usr/lpp/DPS/bin/copyright &
#fi
```

As you create and move AVS windows on the screen, you will often briefly see randomly-colorful rectangles appear before AVS draws its output into them. This appears to be a GL artifact and does not affect performance.

Dial Box

The Dial Box I/O device mentioned in the *AVS User's Guide* is no longer supported. The supporting libraries used in the AVS implementation of the device drivers are no longer included in AIX 4.3.

Spaceball

The Spaceball device is supported.

Window Manager

AVS works with any of the standard X Window System window managers, including *mwm*, AIX's implementation of the Open Software Foundation's Motif window manager. No user modifications are necessary in order to use AVS with Motif. However there are some modifications that you might want to make as a matter of personal preference. See the "Window Manager" section in Chapter 3, *AVS 5.5 on UNIX Workstations* for more information on modifying how AVS behaves under Motif.

Starting AVS

This section lists special cases in which the IBM version of AVS 5.5 differs from the description given in the "Starting AVS" chapter of the *AVS User's Guide.*

GL Gamma Correction

"GL Gamma correction" is a process by which a color output display is brightened or darkened in software or hardware to compensate for the display characteristics of different color monitors. The same output can appear darker or lighter on different monitors depending upon their individual gamma factors.

In the absence of any information, AVS will use its own gamma correction factor to brighten all of its display windows *except* the GL output window produced by the Geometry Viewer, and the GL output window produced by the **display 24bit image** module. This may leave the GL window slightly dark.

You can override the AVS gamma correction and establish a gamma correction factor for the GL window on the High Performance Adapter using the AVS_GL_GAMMA environment variable. A *csh* user would set this variable as follows:

`setenv AVS_GL_GAMMA` *correction-factor*

either interactively, or in one of their startup files.

A *ksh* or *sh* user would set it by saying:

```
AVS_GL_GAMMA=correction-factor
export AVS_GL_GAMMA
```

either interactively, or in one of their startup files.

Experiment with gamma factor values to find a satisfactory value. A factor of 2.2 works well on the IBM 6091 19" monitor.

AVS_GL_GAMMA has no effect on the GTO Graphics adapter, the Color Graphics Display Adapter, the Gt1, or Gt3 adapters.

Unrecognized Graphics Adapters and AVS_GRAPHICS Environment Variable

There are only two cases where you may need to explicitly tell AVS which adapter it should use:

- If you have a graphics adapter that AVS does not know about. The graphics adapters that AVS supports are listed in the "Graphics Hardware" section above.

  You may have a newer adapter not on this list. In this case, you can try to make AVS work with the unsupported adapter by setting AVS_GRAPHICS equal to the "closest" supported adapter.

The choices are listed below.

- If you receive an error message when you try to run AVS on an RS/6000, while displaying on another system or X terminal. See the "X Terminal Support" section earlier in this chapter for a more complete explanation.

To explicitly select a graphics adapter, you specify the **AVS_GRAPHICS** environment variable appropriate for your system. This will control which adapter is used to render native GL calls.

A *csh* user would set this environment variable as follows:

```
setenv AVS_GRAPHICS value
```

either interactively or in one of their startup files.

A *ksh* or *sh* user would set this environment variable as follows:

```
AVS_GRAPHICS=value
export AVS_GRAPHICS
```

either interactively or in one of their startup files.

The *value*s possible are:

8bitcolor

This category includes all 2D unsupported adapters that are similar to:

The 8-bit Color Graphics Adapter
A High Performance Adapter with 8-bits and/or no Z-buffer
An 8-bit GTO Graphics Adapter
The Gt1 or Gt3 Graphics Adapters

This will cause AVS to use its software renderer to perform 3D rendering.

highperf

Use this option if you have an unsupported adapter that is similar to the High Performance Adapter with 24-bit color and a Z-buffer. This causes AVS to use native GL calls to perform 3D rendering. Some aspects of the Geometry Viewer will also be tailored to GL as it functions with the High Performance Adapter hardware.

gto

Use this option if you have an unsupported adapter that is similar to the GTO Graphics Adapter with 24-bit color. This causes AVS to use native GL calls to perform 3D rendering. Some aspects of the Geometry Viewer will also be tailored to GL as it functions with these graphics adapters.

gt4

Use this option if you have an unsupported adpater that is similar to the Gt4, Gt4e, Gt4x 8- or 24-bit, Gxt800 or Gxt1000 adapters. This causes AVS to use native GL calls to perform 3D rendering. Some aspects of the Geometry Viewer will also be tailored to GL as it functions with these graphics adapters.

Network Name Servers and Slow Startup Performance

If you seem to be experiencing slow performance, especially at AVS startup, you may be having a problem with the name server configuration on your system. This typically happens when you move your system off of a network and your machine expects to connect to a name server which no longer exists. You may also just have a slow name server connection.

To determine if this is the problem, use *smit* to stop using name servers (see IBM system documentation). Another way to disable name server use is to temporarily remove the */etc/resolv.conf* file. If performance improves, you will need to consult your Network or System Administrator to help you resolve the problem permanently.

Image Viewer

All functions and behavior in the Image Viewer are as described in the AVS *User's Guide*, with these exceptions.

Color

The main difference between the Image Viewer running on a true color system versus running on a pseudo color X server is the appearance of the images.

On 24 plane true color systems, each pixel can have one of 256 red x 256 green x 256 blue (16,777,216) color values. There are 8 bits to represent red tones, 8 bits for green tones, and 8 bits for blue tones. The red, green, and blue tones combine to create the actual pixel color. The sample image files in *<installdir>/avs/data/image* are all true color images.

On 8 plane pseudo color systems, in AVS each pixel can have one of 216 color values. There are 6 red tones, 6 green tones, and 6 blue tones. (Note that there are 6 tones, not 6 bits to represent tones.) To display a true color image on an 8 plane pseudo color device, AVS takes the original red value for each pixel and finds the closest numeric value from among the 6 reds available. It does the same for green and blue.

AVS then takes the pixmap that is made up of these three best-matches and applies a dithering algorithm to the pixmap. Dithering uses the fact that the human eye will interpolate between dots of color, creating the impression of a color value between two actual color values. The dithering process corrects for information lost in the 256-to-6 reduction by comparing how far off each final pixel value was from the original value against a dithering matrix. Some pixel values have their red/green/blue values adjusted up or down to create a closer approximation to the original true color image. This might sound very limited, but the end result is surprisingly satisfactory.

On pseudo color systems, both the **image viewer** and **display image** modules have additional controls that let you select the dithering algorithm applied. Also, to make it possible to see 24-bit true color images on AVS on an RS/6000 equipped with 24 planes and Z-buffer, we have added an additional module, **display 24bit image**, that uses the GL library to display images on the screen rather than the X server. The GL library does use all 24 planes available on the High Performance 24-bit Adapter.

Network Editor

The Network Editor functions as described in the AVS *User's Guide* and *Module Reference* manual, with these few exceptions.

Modules

The following modules documented in the *AVS Module Reference* manual do not appear in AVS for the RS/6000 because they require rendering features that are not supported in this release:

alpha blend (requires hardware-assisted alpha transparency)
transform pixmap (requires 2D texture mapping)

There is an additional module, **display 24bit image**, in the "Data Output" column of the supported module library. This module must be used in order to see true color renditions of AVS 5.x image files and images produced by AVS networks on the screen if you have a 24-bit High Performance Adapter.

Using the Geometry Viewer's Image Output Port

If you are using the **image to postscript** module to get output from the **geometry viewer** module while using the hardware renderer, be sure that no parts of the display window are off the screen or obscured by other windows. No output will be generated for the off-screen or obscured portion of the pixmap.

The software renderer will generate a full 24-plane image on its image output port, even on pseudo color devices, including obscured and offscreen portions of the image.

Graph Viewer

Other than the fact that image data loaded as a background to a plot will be dithered to 8-bit pseudo color rather than appear in true color, there are no differences between AVS Graph Viewer behavior on RS/6000's and the Graph Viewer descriptions in the AVS documentation.

Geometry Viewer

The Geometry Viewer (or **geometry viewer** module) will be using one of three "platforms" to render into its output windows:

- The software renderer on:
  - the 8-bit Color Graphics Display Adapter
  - either the High Performance Adapter, or the GTO Graphics Adapter if they are missing either the 24-bit color option, or the hardware Z-buffer option
  - the Gt1 or Gt3 Graphics Adapters
  - any remote X Terminal
- The OpenGL library with the High Performance 24-bit Adapter with Z-buffer, or the Gt4, Gt4e, or Gt4x Graphics Adapters
- The OpenGL library with the GTO 24-bit Graphics Adapter with Z-buffer

Note: You may also decide to use an alternate AVS executable *avs.gl* that provides GL library support.

Color

The 24-bit High Performance Adapter, and the 24-bit GTO, Gxt800 or Gxt1000 Graphics Adapter platforms differ in one major feature:

- The High Performance Adapter uses two 12-bit color buffers in double buffer mode, and a 24-bit color buffer in single buffer mode. (You switch between double and single buffer mode using the **Double Buffer** switch on the **Cameras** submenu.)
- The GTO, Gxt800 and Gxt1000 Graphics Adapters use two 24-bit color buffers in double buffer mode, and a single 24-

bit color buffer in single buffer mode.

"Double buffer" means that OpenGL renders its output image into an off-screen pixmap, then transfers the image to the screen, switching back and forth between the two buffers. "Single buffer" means that OpenGL renders its output image directly to the screen. If single buffer mode is set, you can watch the object being drawn; if double buffer mode is set, the completed picture simply appears in the window.

Thus, output images created by the High Performance Adapter in double buffer mode are 12-bit true color images. There are four bits used for each component (red, green, and blue), giving sixteen possible values for each component. This produces 16 red x 16 green x 16 blue possible colors, or 4096 total possible colors. The hardware performs the dithering from 24 to 12 bits. To see a 24-bit true color rendition of a scene, turn off **Double Buffer** mode under the Cameras submenu.

Output images produced by the GTO, Gxt800 or Gxt1000 Graphics Adapters are full 24-bit true color in both single and double buffer modes.

Rendering Features

The following table lists the rendering features described in the "Geometry Viewer" chapter of the *AVS User's Guide* down the left column; and the AVS software renderer; the AVS GL implementation on the High Performance Adapter and the Gt4 and Gt4x adapters (differences with the GTO Graphics adapter are shown in []'s); and the AVS OpenGL renderer across the top. The table intersections show which features are present on each platform, and draw your attention to more detailed explanations of behavior later in this section.

Table 6-1. Geometry Viewer Behavior Across Platforms

Rendering Feature	**Color Graphics Adapter Gt1, Gt3,** or **Software Renderer**	**GL Gt4, Gt4x|e Gxt800 or Gxt1000 [GTO]**	**OpenGL**
Arbitrary Clip Planes	yes, 8 planes	no (note 1)	no (note 1)
Geometric			
Volume Rendering	yes (note 2)	no (note 2)	no (note 2)
Vertex Transparency	no (note 3)	no[yes] (note 3)	yes (note 3)
Vertex Colors	yes (note 4)	yes	yes
Edit Property			
RGB/HSV Colors	yes, (note 4)	yes	yes
Ambient	yes	yes (note 5)	yes (note 5)
Diffuse Lighting	yes	yes (note 5)	yes (note 5)
Specular Highlights	yes	yes	yes
Gloss	yes	yes	yes
Transparency	yes (note 6)	no [GTO, Gxt1000 yes] (note 12)	yes (note 12)
Metallic	yes	yes	yes
Edit Texture	yes	no	yes
2D Texture Mapping	yes	no	yes
3D Texture Mapping	yes	no	no
Filtered Textures	no	no	yes
Alpha Textures	yes	no	no
Rendering Options			
Points	yes	yes	yes
Lines	yes	yes	yes
Smooth Lines	no	no	no
No Lighting	yes	yes	yes
Flat Shading	yes	yes	yes
Outline Gouraud	yes	yes	yes
Gouraud Shading	yes	yes	yes
Phong Shading	no	no	no
Backface Properties			
Cull front	no	no	no
Cull back	yes	yes	yes
Flip normals	no	no	no
Lights			
Number of Sources	16/8 (note 7)	8	8
Ambient	yes	yes	yes
Directional	yes	yes	yes
Bi-Directional	yes	yes	yes
Point	no	yes	yes
Spot	no	no	no
Light Colors	yes	yes	yes

Table 6-2. Geometry Viewer Behavior Across Platforms (continued)

Rendering Feature	**Color Graphics Adapter Gt1, Gt3,** or **Software Renderer**	**GL Gt4, Gt4x|e Gxt800 or Gxt1000 [GTO]**	**OpenGL**
Cameras			
Depth Cue	yes	yes (note 8)	yes (note 8)
Perspective	yes	yes	yes
Accelerate	no	no	no
Axes for Scene	yes	yes	yes
Front/Back Clipping	yes	yes	yes
Polygonal Spheres	yes (note 10)	no	no (note 11)
Double Buffer	no	yes (see "Color" section above)	yes
Stereo	no	no	no
Labels	yes	yes (note 9)	yes (note 9)
Drop Shadow	no	yes	yes
Stroke	yes	yes	yes
Kanji	yes	no	no

Notes

1. Arbitrary clipping planes are used by the **clip geom** module.

2. "Volume Rendering" is here defined in a very narrow sense: support of a specific option to the **GEOMedit_texture_options** call that is used by the **volume render** module. The fact that the hardware renderer does not support this form of volume rendering only means that it cannot use one of the features of the **volume render** module; it does not mean that the hardware renderer cannot render volumes of data using other rendering techniques.

3. Vertex transparency is used by the **colorize geom** module.

4. When using the software renderer, the number of colors that will appear on the screen (216 pseudo color, 16,777,216 true color, or some number in between) is dependent upon the X server visual support present on the *display* hardware. Internally, 24-bit true color is always used and will be output on the **geometry viewer** module's image output port.

5. The **Ambient** and **Diffuse** sliders have no effect on objects with vertex colors.

6. The software renderer supports single-pass transparency. Single-pass transparency correctly renders a transparent surface that is over an opaque surface. (Multi-pass transparency is required to correctly render multiple overlapping transparent surfaces.)

7. When, at AVS startup, you initialize the software renderer only (*-nohw* or **NoHW**), then there will be 16 light sources in the software renderer. If, by default, both hardware and software renderers are initialized, then the minimum common number of light sources will be used. Thus, both renderers will have eight light sources.

8. On all adapters, depth cueing (**Depth Cue** button under the Cameras menu) only works on shaded objects if lighting is turned off (**No Lighting** button under the Objects menu) *and* if an object is not vertex-colored. This limitation occurs on a per-object basis; that is, if you have a scene with multiple objects and one of them is vertex-colored, only that object is not depth-cued. This is a GL limitation. Depth cueing works fine in **Lines** or **Points** representations.

9. On both the High Performance Adapter and the GTO Graphics Adapter, text whose origin (lower left corner) is outside the left border of the display window is not rendered on the screen.

10. A feature in the software renderer is the ability to render spheres directly instead of subdividing them into polygons. This saves a tremendous amount of memory when rendering spheres. The algorithm's execution speed is bound by the size of the spheres it has to render. This feature may be disabled by toggling the **Polygonal Spheres** button under the Cameras menu. Spheres will then be rendered by subdividing them into polygons.

11. Spheres are subdivided into polygons when using the GL based hardware renderers.

12. On adapters that support hardware transparency, the transparency of the first vertex of a triangle strip is applied to the whole strip. Thus, this is not true vertex transparency.

Programming Considerations

There are very few platform-specific issues to take into account when developing AVS modules on the RS/6000.

Compiling and Linking Modules

AVS 5.5 was compiled using version 3.6.4.0 of the AIX C Compiler and 3.1.4 of the Fortran compiler.

The C and C++ compilers have moved to a new home location, */usr/ibmcxx/bin*, that may cause problems when compiling modules. If you have problems, check your PATH environment variable (printenv PATH) and add the */usr/ibmcxx/bin* directory to your path if necessary.

You should use *<installdir>/avs/examples/Makefile* as the template for your own FORTRAN and C module make files. This make file in turn includes the file *<installdir>/avs/include/Makeinclude* that contains additional macro definitions appropriate to the IBM platform.

Note: Please see the information on "GEOMint_color" in Chapter 3, *AVS 5.5 on UNIX Platforms* for an issue that will affect modules using the GEOM library on 64 bit platforms.

Compiling with C

When compiling modules written in C, you must use the *cc* compiler, not the *xlc* or *c89* compilers. These are actually all the same compiler, but *xlc* and *c89* restrict code to ANSI-C compliance. AVS requires the somewhat greater leniency afforded by *cc*. *cc* is the compiler called in the sample Makefile in *<installdir>/avs/examples*.

Using FORTRAN in AVS

Avoiding Duplicate Symbol Names

The use of FORTRAN in AVS relies on the ability of FORTRAN to call C functions and C to call FORTRAN functions. This is true both for FORTRAN modules, which are using the AVS libraries which are written in C, or for C user modules in which some FORTRAN support code is being included.

On most hardware platforms, there is some compiler convention to differentiate the names of functions and subroutines in one language from those in the other to avoid ambiguity (C and FORTRAN use different conventions for passing arguments, handling strings, etc.). In many cases an underbar suffix is added or the name is modified to all upper case.

On the RS/6000 the default convention is that there is no such distinction; however, the underbar suffix can be added *as an option*. AVS relies on using this option by requiring that all FORTRAN modules use this compiler flag and that all FORTRAN-C interlanguage functions generated by *f77_binding* (an AVS provided utility program) also are built using this suffix. This can be easily done by following the conventions in the *<installdir>/avs/examples/Makefile*.

This Makefile defines two symbols (F77FLAGS=$(AFFLAGS) and FFLAGS=$(F77FLAGS)) that are further expanded in the *<installdir>/avs/include/Makeinclude* file to include two options to the *xlf* FORTRAN compiler. One of these options (-qextname) prevents the *f77_binding* utility from producing duplicate symbol names to corresponding C functions. The second (-qcharlength=1024) ensures that *xlf* will allow symbol names long enough for AVS's requirements.

Using this naming convention may cause problems linking with existing FORTRAN libraries that the user does not wish to recompile using this compiler option; the problem will be evident if there are complaints about unresolved symbols that have the underbar suffixes. The developer can work around these problems by using an option on the linker (*ld*) to modify a limited number of symbols to use underbars or not. For example:

```
ld -r -brename:myfunction, myfuntion_ myfile.o -o mynewfile.o
```

will ask the linker to modify an existing object file (*myfile.o*) and change a specific symbol (myfunction) to have an underbar (myfunction_) and produce a new version of the object file (*mynewfile.o*). The -r option tells the linker that it is producing a partial link, i.e. it is not doing the final link and worrying about unresolved symbols. Multiple -brename option clauses may be given.

If you are having this kind of problem, modify your make file to post-process your object files to adjust the mismatched symbols appropriately. If there are many library calls, you may want to separate out the bulk of your compute code into a separate source file that you compile without the -qextname flag and then use the linker operation to fix a smaller number of entry points between the AVS module and this separate source file.

Linking C and FORTRAN

When using *cc* to link a combination of C and FORTRAN object files, always add the -lxlf flag at the end of the compile command. This will allow the linker to find the symbols that the FORTRAN object files are looking for in the standard FORTRAN library (*libxlf.a*).

No FORTRAN byte Data Type

The IBM RS/6000 FORTRAN compiler does not support a **byte** data type. The *<installdir>/avs/examples/colorizer_f.f* example shows a FORTRAN program that manipulates an AVS field containing byte data by using an **integer** type instead, and calling special accessor functions.

AVS on IBM RS/6000 Workstations: Known Problems

This is the list of known problems for AVS 5.5 on the IBM platform.

X6610: Shared Memory Segments

Problem Description:

A current limitation on all IBM RS/6000 platforms concerns the number of attached shared memory segments. In the current implementation of AVS, a maximum of 10 shared memory segments can be attached to a process at any one time. Users will need to be aware that large networks with more than 10 different AVS fields or UCD structures will exceed this limit.

Workaround: AVS implements its own workaround to this limitation. If the downstream modules are in the same process, then they pass data by passing pointers. If the downstream modules are in a different process, then AVS falls back on using non-shared memory and UNIX sockets to communicate the data between modules and the AVS kernel. AVS does not fail, but this latter approach does consume more swap space and is somewhat slower since the input data must be copied between modules.

X8601: text (fonts) in Graph Viewer garbled on ibm

Problem Description:

Some fonts in the **Graph Viewer** produce clipped characters. It looks like the left half of the character is ok, but the right is clipped away.

X10736: Color legend has a white color bar

Problem Description:

GTO (IBM) only: The **color legend** module's color bar is full white only.

X10793: AVS port.h _ABS macro conflicts with IBM C++

Problem Description:

The AVS include file *port.h* contains a macro called _ABS which conflicts with */usr/lpp/xlC/math.h*, an include file that is automatically found by the IBM C++ compiler.

AVS on IBM RS/6000 Workstations: Known Problems (continued)

Problem: *avs.gl* problems with suspect fonts

Problem Description:

During internal testing, the GL based renderer was found to crash under some conditions when rendering labels in the Geometry Viewer. In other cases, labels would be invisible until the window was rescaled to force a different scaling factor. These problems were traced to a number of fonts that reported zero-size for some scaling factors and are most likely testing artifacts. Please contact support@avs.com if you encounter problems of this sort.

AVS on IBM RS/6000 Workstations: Fixed Problems

This is the list of fixed problems for AVS 5.5 on the IBM RS/6000 platforms.

10986: AVS 5.4 doesn't run on AIX 4.3

Problem Description:

AIX 4.3 has dropped support for the XListInputDevices routine used by the dialsbox driver code in AVS 5.4 causing AVS to not run. The dialsbox driver has been dropped from AVS 5.5 to fix this problem.

17555: cc compiler path changed

Problem Description:

The default path for the IBM cc compiler has changed from */usr/bin* to */usr/ibmcxx/bin*. This may affect some scripts or the Module Generator that default that have troubles finding "cc". The end user needs to modify their PATH environment variable to include */usr/ibmcxx/bin* in their path.

AVS 5.5 FOR SILICON GRAPHICS

CHAPTER SEVEN

Introduction

This chapter describes AVS 5.5 as it runs on Silicon Graphics Incorporated workstations. There are two different versions of AVS 5.5 for SGI: one for 32-bit architectures (N32) and one for 64-bit architectures (N64); both versions are addressed in this chapter. It covers hardware and software prerequisites needed to run AVS; differences and special considerations for running AVS on this platform; and known problems.

AVS5.5 NOTE: The supported operating system has been updated to IRIX 6.5 for both 32 bit and 64 bit architectures. The O32 platform has been dropped and the N32/Mips3 has been added. The N64/Mips4 platform has been updated from IRIX 6.2 to IRIX 6.5. Stereo support has been reworked to provide quad-buffered (stereo-in-a-window) support in addition to full screen stereo, borrowing from the AVS/Express 5.0 implementation. Several SGI specific bugs have been addressed. No other significant platform specific changes have been made.

NOTE: Additional information on using AVS under X windows, general programming considerations, and bugs fixed in AVS 5.5 can be found in the *AVS 5.5 on UNIX Platforms* chapter above.

Hardware Prerequisites

Workstation Models

AVS will execute on any workstation in the Silicon Graphics 4D product line, including the IRIS Indigo, Crimson, Challenge, Onyx, Octane and O2. There are two versions of AVS 5.5 for Silicon Graphics (SGI) platforms:

SGN32 supports 32-bit architectures:
This version executes on any SGI workstation or server using the 32-bit processors running IRIX 6.5 or above, and may be used on 64-bit

processors if necessary. The executables use the "n32" (new 32 bit) binary format and can be run on "n32" or "n64" platforms; the code was compiled for Mips3 compatible architectures. Products that use this architecture include Indy, Indigo, Crimson, Challenge, Onyx, and O2, running a version of IRIX 6.5. OpenGL support only is provided; there is no longer any GL renderer due to SGI no longer supporting this interface in IRIX 6.5.

SGN64 requires 64-bit architectures:
This version executes on any SGI workstation or server using a 64-bit processor. The executables use the "n64" (new 64 bit) binary format and can only run on 64 bit platforms; the code was compiled for Mips4 compatible architectures. Products that use this architecture include POWER Indigo 2, POWER Challenge, and Octane, running a version of the IRIX 6.5 operating system or later.

Graphics Hardware

This release of AVS uses two mechanisms to produce its screen renderings of Geometry Viewer objects and scenes: a **hardware renderer** that uses the native graphics hardware and standard graphics library; and a **software renderer** that implements a set of graphics primitives (polygons, lighting model, perspective, shading, color, etc.) in software, rendering the resulting image into an X Window System image.

NOTE: Both SGN32 and SGN64 now use OpenGL for hardware rendering.

The **software renderer** will work on any SGI workstation with an 8- or 24-plane color frame buffer, the XPutImage *xlib* call, and a local X server that supports them.

In principle, the **hardware renderer** should work on any SGI workstation equipped with an 8- or 24-plane display, and a graphics subsystem—either hardware, or software-emulated (as on the Indigo)—that supports the standard graphics library. However, in addition, the graphics subsystem *must* support a Z-buffer. If your system does not support a Z-buffer, then you will need to use the software renderer exclusively.

See the "Geometry Viewer" section below for specific information on which rendering features are supported by the software and hardware renderers.

Memory Requirements

AVS requires that the system upon which it will execute be configured with a minimum of 16 megabytes of main memory. More real memory, such as 32 or 64 megabytes, is strongly recommended.

Performance will be improved if there is more real memory. In addition to real memory requirements, there is a minimum system swap space requirement (see below).

Disk Space

AVS 5.5 requires approximately 87 megabytes (SGN32 32-bit) or 90 megabytes (SGN64 64-bit) of disk storage to install AVS from the CD. More disk storage may be required for swap space (see below). The *df -k* command reports available disk space.

Determining Your Configuration

To interactively determine your SGI workstation's hardware configuration, type the */bin/hinv* command. This produces a listing like the following:

```
1 30 MHZ IP12 Processor
FPU: MIPS R2010A/R3010 VLSI Floating Point Chip Revision: 4.0
CPU: MIPS R2000A/R3000 Processor Chip Revision: 3.0
On-board serial ports: 4
Data cache size: 64 Kbytes
Instruction cache size: 64 Kbytes
Main memory size: 32 Mbytes
Integral Ethernet: ec0, version 0
Tape drive: unit 2 on SCSI controller 0: QIC 150
Disk drive: unit 1 on SCSI controller 0
Integral SCSI controller 0: Version WD33C93A
Graphics board: GR1.2 Bit-plane, Z-buffer, Turbo options installed
```

The "Main memory size" line reports the memory size. The "Graphics board" line reports the hardware graphics present.

For more information on graphics hardware options, such as the number of texture memory modules installed, use the */usr/gfx/gfxinfo* command. This produces a listing like this:

```
Graphics board 0 is "IMPACT" graphics.
   Managed ("0:0") 1280x1024
   Product ID 0x0, 1 GE, 1 RE, 1 TRAM
   MGRAS revision 3, RA revision 5
   HQ rev A, GE11 rev B, RE4 rev A, PP1 rev A,
   VC3 rev A, CMAP rev D, MC rev C
   unknown, assuming 19" monitor (id 0x9)
```

Software Prerequisites

IRIX Operating System

There are two different versions of AVS for SGI.

These products appear as different options during installation on multi-product media:

SGN32
AVS for IRIX release 6.5 or later. Requires that 6.5 or later of the IRIX operating system and its accompanying libraries and compilers be installed on the system upon which AVS will execute. This version was compiled on an R10000 with the Mips3 flag.

SGN64
AVS for IRIX release 6.5 or later. Requires that 6.5 or later of the IRIX operating system and its accompanying libraries and compilers be installed on the system upon which AVS will execute. This version was compiled on the R10000 using the Mips4 flag and will not run on R3000 or R4000 machines.

The differences between the SGN32 and the SGN64 AVS versions are:

- The SGN32 libraries are the "N32" Application Binary Interface (ABI), whereas the SGN64 libraries are in the "n64" ABI. A older format, the "o32" ABI, is no longer supported.

The */bin/uname -a* command will list which version of IRIX is running on the system.

Swap Space

Recommended Swap Space Size

We recommend that your system be configured with a minimum swap space size that is twice the real memory size or 64 megabytes, whichever is *greater*. Very large datasets flowing through networks with many modules and connections may require even larger swap spaces.

Determining How Much Swap Space You Have

Most SGI workstations come with the default system swap space size set to about 50 megabytes. To see how much swap space is available to your system, type the */etc/swap -l* command as follows:

```
/etc/swap -l
path               dev  swaplo blocks   free
/dev/dsk/dks0d1s1 22,33      0  81728  62000
```

The "blocks" column is the total swap space reserved on the system in 512-byte blocks, both in the system swap space partition and any additional swap space that might have been defined for the system on other partitions; "free" is the amount currently available for allocation and will vary with the load on the system.

Running out of Swap Space: Message, Abort, or Hang

Depending upon the circumstances, when AVS runs out of swap space it either produces a message box, aborts, or hangs. In many cases, AVS itself receives the error during a *malloc* call. In this instance, either individual modules or the entire AVS system may abort with the following error message:

```
Failure allocating memory:
See Installation/Release Notes to increase swap space and/or
shared memory segment size
```

This message may be followed by many additional "tcp read" error messages as each module expires. Scroll to the top of the error messages to see this first message that indicates the actual cause of the failure.

In some cases, other parts of the system will generate the error and you will not see the above message. If you suspect that swap space may be the culprit, you can restart AVS, repeat the sequence of steps that led to the abort or hang, while monitoring AVS's consumption of swap space by entering successive */etc/swap -l* commands.

How to Increase System Swap Space

See the Silicon Graphics *IRIX System Administrator's Guide* for information on how to increase the system swap space size.

Using AVS on an SGI Workstation

The following sections describe in detail the differences between using AVS on an SGI workstation and AVS as it is described in the standard documentation.

General

Dial Box

The Dial Box I/O device mentioned in the *AVS User's Guide* is not supported in this release of this product.

Spaceball

The Spaceball device is supported. See bug report **X8581** in the "Known Problems" section below for information on how to correctly activate the Spaceball.

Window Manager

AVS works with any of the standard X Window System window managers, including *4Dwm*, SGI's implementation of a Motif-style window manager. No user modifications are necessary in order to use AVS with *4Dwm*. However, *4Dwm*'s interaction norms do differ somewhat from the earlier X window managers under which AVS was originally developed. See the "Window Manager" section in Chapter 3, *AVS 5.5 on UNIX Platforms.*

Starting AVS

There are four differences to starting AVS on an SGI workstation from the account in the AVS *User's Guide* and the *avs* man page. These are

- changing your display size if you have a small monitor.
- setting the correct X server VisualType for your graphics adapter (if necessary).
- setting a gamma correction factor (if necessary).
- setting a value for the **AVS_XOR_VALUE** environment variable to make certain display lines, such as that used for the Bounding Box, visible. This is only necessary on some graphics adapters, e.g., the Crimson with Elan graphics.

Reconfiguring the first three items listed involves making changes to your personal AVS startup file (*.avsrc*) in your HOME directory. If you don't already have a *.avsrc* file, copy the sample file from *<installdir>/avs/runtime/avsrc* to your home directory and edit it to include the necessary lines. **AVS_XOR_VALUE** is an environment variable that must be set through your system's shell.

Display Size

AVS normally expects to execute on 1280x1024 resolution displays. If the display is smaller or larger than this, AVS will try to automatically rescale its interface to fit on the screen. If AVS's automatic efforts are not satisfactory (for example, the bottoms of menus are being clipped by the window manager or the Network Editor panel is overlapping other windows), you can explicitly set a virtual screen size and a position for the Network Editor window by adding lines such as the following to your personal *.avsrc* file or using the *-size* command line option:

```
NetworkWindow       850x850+250+20
ScreenSize          1080x864
```

Visual Type: Seeing Pseudocolor on a Truecolor Adapter

This issue affects some 24-plane true color graphics systems.

IRIX often starts the X Window System server using a pseudo color X visual (as reported by the *xdpyinfo* command) as its default visual, even though the display hardware is capable of supporting 24-plane true color. By default, AVS always uses this default X visual when it starts.

On 8-plane pseudo color systems, this is fine. AVS will use the correct pseudo color visual for the 8-bit platform.

However, on 24-plane graphics systems, AVS will also come up using the 8-bit default pseudo color visual, unnecessarily limiting the color resolution.

Permanently Changing the Default Visual Systemwide. You can permanently change the X Window System's default visual.

1. As root, edit the file */usr/lib/X11/xdm/Xservers*.

2. Change the line:

   ```
   :0 secure /usr/bin/X11/X -bs -c -pseudomap 4sight
   ```

   To read:

   ```
   :0 secure /usr/bin/X11/X -class TrueColor -depth 24 -bs -c -pseudomap 4sight
   ```

3. Kill and restart the X server.

SGI Crimson, Invisible Bounding Boxes, and AVS_XOR_VALUE Environment Variable

On a few SGI platforms, such as the SGI Crimson with Elan graphics, the "xor'd" pixel value used to draw lines such as the Geometry Viewer's Bounding Box are invisible.

To make these lines visible, you must set the AVS_XOR_VALUE environment variable in one of your startup files (usually *.login* or *.csh* for *csh*, and *.profile* for *sh* or *ksh*). Set the environment variable as follows:

```
csh:          setenv AVS_XOR_VALUE 0xnnnnnn

sh or ksh:    AVS_XOR_VALUE=0xnnnnnn; export AVS_XOR_VALUE
```

0x*nnnnnn* is a pixel value. The first two characters are a zero (not an "oh") and the lowercase character "x". This is a 24-bit number, with 1 byte for red, 1 byte for green, and 1 byte for blue. The *nnnnnn* is thus two hexadecimal characters each (range 0 through f) for red, green, and blue. On truecolor systems, all three bytes are used to determine the line color. On pseudocolor systems, only the low order byte is significant. It is used as an index into the hardware colormap.

You will need to experiment (e.g., 0x8a8a8a, 0xa2a2a2, etc.) to determine a good combination for your platform. On truecolor systems, equal values for red, green, and blue produce a whitish line. (0xaaaaaa is the default AVS value that produces the nearly invisible lines.) On pseudocolor systems, since the lower order byte is a colormap index, it is hard to predict the result of any given number.

Image Viewer

All functions and behavior in the Image Viewer are as described in the AVS *User's Guide*. Color rendition will be up to the capabilities of the X server.

On pseudocolor systems, the Image Viewer's **Images** submenu will have a choice of dithering options.

Network Editor

The Network Editor functions as described in the AVS *User's Guide* and *Module Reference* manual, with these few exceptions.

Modules

The following modules documented in the *AVS Module Reference* manual do not appear in the AVS release for SGI workstations.

alpha blend
transform pixmap

Image Output

While using the hardware renderer, if you are using the **image to postscript** module to get output from the **geometry viewer** module, or any of the Animation Application output modules such as **write frame seq**, all parts of the Geometry Viewer scene window must be visible. No parts of the display window can be off the screen or obscured by other windows. If they are, any module downstream of **geometry viewer** will not receive the off-screen or obscured portion of the pixmap.

Graph Viewer

There are no significant differences between AVS Graph Viewer behavior on SGI workstations and the Graph Viewer descriptions in the AVS documentation. On pseudo color systems image data loaded as a background to a plot will be dithered to 8-bit pseudo color rather than appear in true color.

Geometry Viewer

The Geometry Viewer (or the **geometry viewer** module) will be using either the software renderer or the hardware renderer to produce the contents of its scene windows.

Color

The OpenGL graphics subsystem that produces the contents of Geometry Viewer scene windows under the hardware renderer uses its own visuals (as distinct from X Window System visuals). On some systems and visual types, OpenGL may split the number of planes in two in order to support "double buffering". For example on 24-plane systems, it may use 12-planes for each buffer, providing 4096 colors instead of the 16 million supported by the full 24 planes.

"Double buffering" just means that OpenGL renders its output image into an off-screen pixmap, then transfers the image to the screen, switching back and forth between the two buffers. "Single buffering" means that OpenGL renders its output image directly to the screen. If single buffer mode is set, you can watch the object being drawn; if double buffer mode is set, the completed picture simply appears in the

window.

To see a full 24-plane true color rendering, you might have to change to single buffer mode by switching off the **Double Buffer** button on the **Cameras** submenu. There is, at present, no AVS startup option that will automatically switch off double buffering.

Stereo Support

Both versions of AVS 5.5 include basic support for stereo viewing using OpenGL. The stereo viewing device supported is the CrystalEyes stereo goggles from StereoGraphics. See the "AVS 5.5 on UNIX Platforms" chapter for more general information.

Stereo is enabled using either the *setmon* command or the *ircombine* command depending on the type of machine (higher end graphics adapters like the Infinite Reality cards usually use *ircombine* though they may also still support *setmon*).

If */usr/gfx/ircombine* exists, it is used automatically with the following settings to enable and disable stereo:

```
on  - /usr/gfx/ircombine -destination active -channel 0 format=1024x768_96s.vfo
off - /usr/gfx/ircombine -destination active -channel 0 format=1280x1024_72.vfo
```

Otherwise if */usr/gfx/setmon* exists, it is used with the following settings:

For Reality Engine:

```
on - /usr/gfx/setmon -n 1025x768_96s
off - /usr/gfx/setmon -n 1280x1024_60
```

For VGX and VTX:

```
on - /usr/gfx/setmon -n STR_BOT
off - /usr/gfx/setmon -n 60HZ
```

All others:

```
on - /usr/gfx/setmon -n STR_BOT
off - /usr/gfx/setmon -n 72HZ
```

You can see the current setting using two debug environment flags:

```
setenv AVS_STEREO_CMDS_ARE 1
setenv AVS_OGL_INFO 1
```

In order to override these values, there are two environment variables to set the "on" and "off" commands:

- **AVS_STEREO_ON_CMD** specifies the command that enables stereo mode
- **AVS_STEREO_OFF_CMD** specifies the command that disables stereo mode

Because AVS does NOT "remember" your previous mode before it entered stereo, you may want to at least set the "off" variable, AVS_STEREO_OFF_CMD.

To exit stereo mode, type **Ctrl-left mouse button** again or use the Camera/Stereo toggle button. **Note:** You should exit stereo mode before leaving AVS or it may leave stereo enabled. If you do leave stereo on by accident, restart AVS and toggle stereo on/off to leave stereo mode; or use the "off" stereo command shown above to exit stereo manually.

For information on known problems wtih SGI stereo, see the Known Problems section at the end of this chapter.

Rendering Features

The following table lists the rendering features described in the "Geometry Viewer" chapter of the *AVS User's Guide*, including new AVS 5 features, down the left column, and the AVS software renderer and the AVS hardware renderer's OpenGL implementation across the top. The table intersections show which features are present on each platform, and draw your attention to more detailed explanations of behavior later in this section.

Table 7-1. Geometry Viewer Behavior Across Platforms

Rendering Feature	Software Renderer	Hardware (OpenGL)
Arbitrary Clip Planes	yes, 8 planes	no (note 1)
Geometric		
Volume Rendering	yes (note 2)	yes (note 2)
Vertex Transparency	no (note 3)	yes (note 3)
Vertex Colors	yes (note 4)	yes
Edit Property		
RGB/HSV Colors	yes (note 4)	yes
Ambient	yes	yes (note 5)
Diffuse Lighting	yes	yes (note 5)
Specular Highlights	yes	yes
Gloss	yes	yes
Transparency	yes (note 6)	yes (note 7)
Metallic	yes	yes
Edit Texture	yes	yes (note 7)
2D Texture Mapping	yes	yes
3D Texture Mapping	yes	yes (note 2)
Filtered Textures	no	yes
Alpha Textures	yes	no
Rendering Options		
Points	yes	yes
Lines	yes	yes
Smooth Lines	no	no
No Lighting	yes	yes
Flat Shading	yes	yes
Gouraud Shading	yes	yes
Outline Gouraud	yes	yes
Phong Shading	no	no
Backface Properties		
Cull front	no	no
Cull back	yes	yes
Flip normals	no	no
Lights		
Number of Sources	16/8 (note 8)	8
Ambient	yes	yes
Directional	yes	yes
Bi-Directional	yes	yes
Point	no	yes
Spot	no	no
Light Colors	yes	yes

Table 7-2. Geometry Viewer Behavior Across Platforms (continued)

Rendering Feature	Software Renderer	Hardware (OpenGL)
Cameras		
Depth Cue	yes	yes
Perspective	yes	yes
Accelerate	no	no
Front/Back Clipping	yes	yes
Axes for Scene	yes	yes
Double Buffer	no	yes
Sorted Transparency	no	no
Shadows	no	no
Polygonal Spheres	yes	no
Stereo	no	yes
Head Tracking	no	no
Labels		
Drop Shadow	no	yes
Stroke Text	no	no
Kanji	yes	no

Notes

1. Arbitrary clipping planes are used by the **clip geom** module.

2. Currently, only the very high-end platforms (Reality Engine and Onyx) support this feature.

3. Vertex transparency is used by the **colorize geom** module.

4. When using the software renderer, the number of colors that will appear on the screen (216 pseudo color, 16,777,216 true color, or some number in between) is dependent upon the X server visual support present on the *display* hardware. Internally, 24-bit true color is always used and will be output on the **geometry viewer** module's image output port.

5. The **Ambient** and **Diffuse** sliders have no effect on objects with vertex colors.

6. The software renderer supports single-pass transparency. Single-pass transparency correctly renders a transparent surface that is over an opaque surface. Multi-pass sorted transparency is required to correctly render multiple overlapping transparent surfaces.

7. Transparency and texture mapping are supported on those SGI systems where these capabilities are available. On systems that do

not have support for them, enabling the feature has no effect.

8. When, at AVS startup, you initialize the software renderer only (*-nohw* or **NoHW**), then there will be 16 light sources in the software renderer. If, by default, both hardware and software renderers are initialized, then the minimum common number of light sources will be used. Thus, both renderers will have eight light sources.

9. A feature in the software renderer is the ability to render spheres directly instead of subdividing them into polygons. This saves a tremendous amount of memory when rendering spheres. The algorithm's execution speed is bound by the size of the spheres it has to render. This feature may be disabled by toggling the **Polygonal Spheres** button under the Cameras menu. Spheres will now be rendered by subdividing them into polygons.

Programming Considerations

There are very few platform-specific issues to take into account when developing AVS modules on SGI workstations.

Compiling and Linking Modules

AVS 5.5 was compiled using the 7.2 versions of the FORTRAN and C compilers and their associated libraries that accompany the IRIX 6.5 system.

Note: Please see the information on "GEOMint_color" in Chapter 3, *AVS5.5 on UNIX Platforms* for an issue that will affect modules using the GEOM library on 64 bit platforms.

Use the Example Makefile as Template

You should use *<installdir>/avs/examples/Makefile* as the template for your own FORTRAN and C module make files. This make file in turn includes the file *<installdir>/avs/include/Makeinclude* that contains additional macro definitions appropriate to the SGI platform, its compilers, compiler options, include files, and libraries.

Compiling with C

As an examination of the *Makeinclude* file would show, when compiling modules written in C you must use the *cc* compiler with the **-cckr** option. This option uses a C language standard as defined in the Kernighan and Ritchie C book, rather than ANSI C. AVS requires the somewhat greater leniency afforded by **-cckr**.

FORTRAN Modules

On 32-bit platforms, pointer and integer data types are the same size and have historically been used as if they were the same. On most 64-bit platforms (Compaq Tru64 UNIX and SGI IRIX 6.5 64-bit (*SGN64*)) pointers are 8 bytes and integers are 4 bytes; the two data types can NOT be used interchangeably. See the section on "FORTRAN Modules on 64-bit systems" in the *UNIX Platforms* chapter for critical information on this topic.

AVS on SGI Workstations: Known Problems

This section lists the known problems with this release of AVS 5.5 that are unique to SGI platforms.

X8328: OpenGL windows in Geometry Viewer cannot follow AVS visual

Problem Description:

The OpenGL **Geometry Viewer** scene windows on the SGI are governed by the GLX interaction and use their own visuals. They cannot follow the LUI_Visual representation of the X server visual that AVS uses.

These problems only occur with the pixmap data type.

If you construct a network that attempts to post-process the pixmap output of the **render geometry** module (e.g., **render geometry**--> **pixmap to image**-->**display image**), the following anomalies occur:

- Your machine is true color, but you have not set the AVS visual to true color, and thus end up using a pseudo color visual. The **pixmap to image** module will produce a solid black image.
- You are in double buffer mode with true color. Only every other frame will actually contain an image. The others will be black.

Workaround: Use the **geometry viewer** module and its image output port for scene content postprocessing, not the unsupported **render geometry** module.

X8371: X terminal support on SGI version isn't automatic

Problem Description:

This happens if the user is running remotely and the local machine is not a distributed OpenGL machine. AVS attempts to start up OpenGL and crashes.

Workaround: Use the **-nohw** option when running remotely in this situation.

X8581: Spaceball on SGI requires special serial port treatment

Problem Description:

Using the spaceball with AVS on the SGI requires that the port the spaceball is connected to be "turned off" in the SGI System Manager utility and then made read/write accessible.

There is an option for serial ports in the SGI OS called "Space-Ball", but setting this option seems to interfere with the spaceball interface. Also, the port (usually */dev/ttyd1*) is only writable by default and is reset to this when altered with the System Manager.

Workaround: Turn off serial port in software (using the System Manager): *chmod a+r /dev/ttyd1* (whatever is appropriate).

X10694: Texture mapping has strange limitations

Problem Description:

The SGI Reality Engine can do 3D texture mapping, but the dimensions of the map must all be powers of 2, and the size must not exceed a certain amount (depending on how much Reality Engine memory you have - typically 128x128x64). Unpredictable results or crashes may result if these are not observed.

X10785: Dual headed operation with different visuals may not work

Problem Description:

If AVS is run on one host and tries to display its **Geometry Viewer** window on a display with a different default visual, problems can occur finding enough colors.

X10301: Module Generator FORTRAN should use integer*8

Problem Description:

The Module Generator fails to provide special %IFDEF statements for pointer parameters like the example FORTRAN modules do or to directly use INTEGER*8. Example:

```
      integer function colorizer2(in_field, cmap, out_field)
C %IFDEF ALPHA_OSF
      integer*8 in_field, cmap, out_field
C %ELSE
C      integer in_field, cmap, out_field
C %ENDIF ALPHA_OSF
```

16241: Stereo Graphics displays incorrectly on Octane and Onyx

Problem Description:

AVS stereo mode may not work on some SGI Octane and Onyx platforms. Problems have been reported in which the system tries to go into stereo mode but the display isn't right, showing top/bottom split images, etc.

We were not able to resolve this by release time but did find information suggesting that SGI patches may be available. It has been reported that patches 1822, 1945, and 1953 break quadbuffer stereo on the MXI graphics cards; this is supposedly fixed with patch 2289. It has also been reported that patch 2290 breaks full screen stereo on the Octane; patch 2448 supposedly fixes this problem. For further information check with SGI Customer Support or visit *http://reality.sgi.com/christig/stereo.html*.

Problem: Screen Flicker from Reset Hertz Rate

Problem Description:

On newer SGI platforms, when you exit stereo mode the monitor is automatically reset to 60HZ. This results in screen flicker.

To reset to 72HZ, type:

```
/usr/gfx/setmon  72HZ
```

The man page for *setmon* describes other ways in which you can control the monitor.

Problem: Label color affected by 2D texture mapping

Problem Description:

Use of 2D Texture mapping in a view may affect the color of labels in the same view.

Problem: XDR Reading Short data type

Problem Description:

Some problems have been reported reading the "short" data type in the XDR format.

18326: IRIX 6.5 FlexLM conflict with SGI compiler licenses

Problem Description:

SGI started using FlexLM to license its compilers in IRIX 6.5. It stores the licenses in the */var/flexlm* directory by default. The */var/flexlm/licensefile.db* file usually contains a list of the files to check for licenses and this usually includes the default path AVS normally uses, */usr/local/flexlm/licenses/license.dat*.

This will normally all work without problems but if the end user redefines the LM_LICENSE_FILE environment variable to identify a demo license or other local license, the SGI C compilers won't be able to find their licenses and will fail.

Workaround:

- don't use LM_LICENSE_FILE if possible
- combine the /var/flexlm license into the new license file you are working with
- set LM_LICENSE_FILE to a longer list of multiple files to check including the /var/flexlm license that has the compiler licenses.

AVS on SGI Workstations: Fixed Problems

This is the list of fixed problems for AVS 5.5 on the SGI platforms.

7823: Coroutine integer output does not work on SG3 (N64)

Problem Description:

The animated integer module was not working on the N64 platform in AVS 5.4 - modules downstream only received "0" because the wrong half of the 64 bit integer was being communicated. This has been fixed.

13157: Memory leak in SGI texture mapping

Problem Description:

Using texture mapping in the Geometry Viewer on the SGI platforms would leak when the textures were changed as the low level copy of the texture map was not being freed. This has been fixed by reenabling shared OpenGL contexts within the renderer, made possible by improvements in the IRIX 6.5 release.

17860: SGI stereo initialization problems on Infinite Reality

Problem Description:

Problems were reported initializing stereo on the IR graphics adapters; attempts to enter stereo mode, left all-red view windows and prevented exiting stereo using the control-mouse-left binding. A temporary workaround of using the Geometry Viewer/Camera/Stereo button three times (on/off/on) seemed to initialize stereo correctly. This has been fixed.

AVS 5.5 FOR SUN SUNOS5 (SOLARIS)

CHAPTER EIGHT

Introduction

This chapter describes AVS 5.5 on Sun Microsystem's SPARC-series workstations running SunOS 5.7 (also called Solaris 2.7 or Solaris 7). It covers hardware and software prerequisites needed to run AVS; differences and special considerations for running AVS on this platform; and known problems.

AVS 5.5 NOTE: The supported operating system has been updated from SunOS 5.5 to SunOS 5.7 (also called Solaris 7) and related changes have been made in the versions of the supporting graphics libraries. OpenGL stereo and the Elite graphics adapter are now supported.

NOTE: Additional information on using AVS under X windows, general programming considerations, and bugs fixed in AVS 5.5 can be found in the *AVS 5.5 on UNIX Platforms* chapter above.

Hardware Prerequisites

Configurations

Sun workstations usually exist in a network environment that can be configured in a variety of ways.

- A system can be *diskless*, meaning that it executes programs locally, but gets all of its software and swap space from a server machine.
- A system can be *dataless* where it executes code, swaps locally, and has its own local kernel, but gets its */usr* partition from a server.
- A system can be *standalone* where it executes and swaps locally, and has its own copy of the kernel and the */usr* software.

However, a remote server may be providing bulk storage, such as the space required to store AVS.

- A system can be truly standalone, where it performs all functions and has all storage local.

- A system can be a *server* that may not execute AVS at all, but provides AVS to some combination of the above systems. These systems may have different, heterogeneous kernel architectures.

Some aspects of AVS installation, licensing, and use require that you be aware of the network configuration under which you are trying to use AVS. These "Installation Notes" make frequent use of phrases such as "on which AVS will execute," "on which AVS swaps," and "on which AVS is stored." Note these phrases when they occur as they are critical to correctly configuring your system to run AVS.

SPARC Models

AVS will execute on any Sun SPARC-series workstation supported by SunOS 5.7 and later. At present, this means machines with *sun4c*, *sun4m*, *sun4e* and *sun4u* kernel architectures. Typing the */usr/bin/showrev* command will produce a listing that describes the kernel and application architecture of your hardware such as the following:

```
Hostname: seymour
Hostid: 807853f9
Release: 5.7
Kernel architecture: sun4u
Application architecture: sparc
Hardware provider: Sun_Microsystems
Domain: avs
Kernel version: SunOS 5.7 Generic October 1998
```

Graphics Hardware

Tested Platforms

This release of AVS uses two mechanisms to produce its screen renderings of Geometry Viewer objects and scenes: a **software renderer** that implements a set of graphics primitives (polygons, lighting model, perspective, shading, color, etc.) in software, rendering the resulting image into an X Window System image; and a **hardware renderer** that uses the native graphics hardware and the OpenGL graphics library. An additional AVS executable, *avs.xgl* is also included to provide continued XGL graphics library support. **Note:** OpenGL only provides acceleration for the Creator3D and later framebuffers; use *avs.xgl* for older framebuffers for improved performance.

The **software renderer** will work on any supported SPARCstation model with an 8- or 24-plane color frame buffer, the XPutImage *xlib* call, and a local X server that supports them.

You can use the *xdpyinfo* command to list the X server support for your display.

The **hardware renderer** will work on the 24-plane GS, GT, ZX, TZX, SX, Creator3D and Evans and Sutherland Freedom graphics adapters. Note that from SunOS 5.5 onwards the GS and GT device drivers are no longer supported by Sun Microsystems.

See the "Geometry Viewer" section below for specific information on which rendering features are supported by the software and hardware renderers.

This release was developed and tested on Sun SPARCstations equipped with the following color graphics adapter boards:

- GX graphics adapter (8-bit color frame buffer)
- S24 graphics adapter (24-bit color frame buffer)
- TZX graphics adapter (24-bit color frame buffer)
- SX graphics adapter (8-bit PseudoColor and 24-bit TrueColor)
- Creator3D graphics adapter (24-bit color frame buffer)
- Elite graphics adapter

Determining Your Graphics Adapter

To interactively determine which graphics board your SPARCstation has, type the command:

```
/usr/sbin/prtconf -F
```

This prints the device pathname of the console framebuffer. The following are examples (this will depend on the system type and configuration):

```
/sbus@1,f8000000/cgtwelve@1,0        a GS (a.k.a a cgtwelve)
/obio/cgfourteen@1,0                 an SX (a.k.a. a cgfourteen)
/iommu@f,e0000000/sbus@f,e0001000/SUNW,leo@3,0
                                     a ZX (a.k.a. a "LEO")
/SUNW,ffb@1e,0                       a Creator3D (a.k.a. the "FFB")
```

Graphics adapters are described on various *man(4S)* device driver man pages:

```
man cgsix          the GX adapter
man cgtwelve       the GS adapter
man gt             the GT adapter
man leo            the ZX adapter
man cgfourteen     the SX adapter
man tcx            the S24 adapter
man ffb            the Creator3D adapter
```

The absence of any local color graphics display hardware means that, at best, you will only be able to run AVS on your SPARCstation as a remote X client from another color workstation or a color "X terminal".

In all cases, you should read the Configuration Guide that comes with your Sun platform to see if special setup procedures are required.

SX Memory and sxconfig

Note that in order to run the SX system using the TrueColor visual (and have AVS do accelerated hardware rendering on this adapter), the system command */usr/kvm/sxconfig* may have to be run to configure sufficient memory for the SX video system. Please type *man sxconfig* to find out how to determine your system configuration and how to reserve memory for the video system. Note that the machine will have to be rebooted using the *-r* option after running *sxconfig* for the changes to take effect.

Creator3D and ffbconfig

AVS 5 requires that the FFB frame buffer be configured to use gamma-corrected visuals, and that the gamma-corrected visuals are first in the list of visuals supported by the frame buffer. You can use the frame buffer configuration command */usr/sbin/ffbconfig -propt* to confirm that the device is correctly configured.

This command prints the current configuration. For information on what the current options mean, use the *man ffbconfig* command to display the *ffbconfig* man page.

The recommended configuration command is:

```
ffbconfig -res 1280 -deflinear true -linearorder first
```

This configuration sets a 1280x1024x76 video mode and, more importantly, it sets gamma-corrected visuals and makes them first in the X11 visuals list. You will have to be root to reconfigure the device; you should do this only when the window system is not running. You can

use the *-propt* option at any time.

If the device is not correctly configured, the graphics (geometry, images, and so forth) rendering will be too dark; that is, not gamma-corrected.

If you are running the device in stereo mode, you will want to set a different -res option, but with the same -deflinear and -linearorder options. Again, see the output of *man ffbconfig*.

Dialbox and bdconfig

If you wish to use a dialbox with AVS (see the User's Guide for usage) you will need to configure the port into which the device is physically connected. The command to use is */usr/sbin/bdconfig*: type the command *man bdconfig* for details. To enable the device in AVS you may use the *-dials* command line option to the avs command or add a *DialDevice* line to your .avsrc file.

Memory

AVS requires that the SPARCstation upon which AVS will execute be configured with a minimum of 16 megabytes of main memory. More real memory, such as 24 or 32 megabytes, is strongly recommended. Performance will be improved if there is more real memory. In addition to real memory requirements, there are minimum system swap space and shared memory segment size requirements. These are described below.

To see how much memory your system has, enter:

```
/usr/sbin/prtconf
```

"Memory size" is printed within the first few lines of output.

Disk Space

It requires approximately 76 megabytes of disk storage to install AVS from the release media. More disk storage may be required for swap space (see below).

Software Prerequisites

Some of the procedures described in this section require superuser (root) privileges to perform. You may require the assistance of a System or Network Administrator to carry out these procedures.

SunOS 5.7 (Solaris 7)

AVS 5.5 for Sun SPARCstations requires release 5.7 or later of the SunOS operating system (Solaris 7).

A different version of AVS 5 is required for SunOS 5.5 and 5.6 systems. You will need to use AVS 5.4 or contact support@avs.com or your local sales office for further information.

The */usr/bin/showrev* will show you which version of the operating system is running.

Graphics Libraries: OpenGL

AVS 5.5 provides a default AVS executable called *avs* that uses OpenGL for hardware rendering. It provides accelerated graphics support on Creator3D and more recent framebuffers. Older frame buffers are supported by OpenGL but performance will be slower than hardware rendering using the XGL graphics library. AVS 5.5 provides a separate executable, *avs.xgl*, discussed in the next section to deliver XGL hardware rendering.

You can always determine which version of AVS 5.5 you are running using the "-version" command line option. The graphics library is listed at the end of the version string.

The OpenGL runtime environment is required for both hardware and software rendering. If you do not already have OpenGL installed on your system, you will need to download and install the runtime environment from Sun Microsystems. The web site address is

```
http://www.sun.com/solaris/opengl/download
```

The installation will require that you stop and restart the window system in order to load the required X GLX extension support OpenGL.

To see if the XGL runtime environment is present, use the */usr/bin/pkginfo* utility. Look for the following lines:

```
application SUNWglrt       Solaris OpenGL Runtime Libraries
application SUNWglrtu      Solaris OpenGL Platform Specific Runtime Libraries
```

You can also check to see if the library */usr/openwin/lib/libGL.so.1* exists. The AVS executable *avs* expects to find the OpenGL runtime library in this location.

If it is not, you will need to set the LD_LIBRARY_PATH environment variable to include its non-standard directory.

You can use the *ldd* command to determine which shared object libraries AVS requires, whether they can be found, and the library locations for those libraries found.

Graphics Libraries: XGL

AVS 5.5 provides an optional AVS executable called *avs.xgl* that requires the presence of version 3.x of the Sun XGL graphics library in order to initialize. If you normally use the XGL executable, you may want to make it your default version of AVS as follows:

```
cd $AVS_PATH/bin
mv avs avs.ogl
ln -s avs.xgl avs
```

You can always determine which version of AVS you are running using the "-version" command line option. The graphics library is listed at the end of the version string.

The XGL runtime environment is required for both hardware *and* software rendering when using the *avs.xgl* secondary kernel. XGL is shipped bundled on the main SunOS CD, and should be installed automatically as part of the main OS installation.

To see if the XGL runtime environment is present, use the */usr/bin/pkginfo* utility. Look for the following lines:

```
application SUNWxgldg      XGL Generic Loadable Libraries
application SUNWxgler      XGL English Localization
application SUNWxglft      XGL Stroke Fonts
application SUNWxglrt      XGL Runtime Environment
```

You can also check to see if the library */opt/SUNWits/Graphics-sw/xgl-3.0/lib/libxgl.so.3* exists. The AVS executable *avs.xgl* expects to find the XGL runtime library in this location.

If it is not, you will need to set the LD_LIBRARY_PATH environment variable to include its non-standard directory.

You can use the *ldd* command to determine which shared object libraries AVS requires, whether they can be found, and the library locations for those libraries found.

Window System

Sun OpenWindows 3.x

AVS runs as an X Window System client. AVS 5 was implemented and tested under Sun OpenWindow and the Common Desktop Environment (CDE).

Other X Window Systems

AVS may also be able to execute under the MIT X Window System Version 11 Release 6 Sun server (color) distributed on the MIT X Window System tape. However, Advanced Visual Systems has not tested AVS under this configuration.

The server must have been built to be aware of the particular color graphics adapter in your SPARCstation (cgsix0, cgtwelve, gt0, etc.) If it was not, you will not be able to start the X server.

Openwin Libraries: can't find libXext.so Error Messages

AVS expects to find the X11 libraries in their default installed location (*/usr/openwin/lib*).

If they are not, then you will need to set the LD_LIBRARY_PATH environment variable to include their non-standard path in order for AVS to run.

You can use the *ldd* command to determine which shared object libraries AVS requires, whether they can be found, and the library locations of libraries found.

Swap Space: Increasing Limits

Recommended Swap Space Size

Most SPARCstations come with the default system swap space size set to about twice the amount of real memory.

We recommend increasing the swap space available to your system to at least 64 megabytes, or twice the amount of available memory, whichever is *larger*. Very large datasets flowing through networks with many modules and connections may require even larger swap spaces. Some modules—such as the **dot surface** module that renders data values as spheres, and the **compute gradient** module that produces 12 bytes for every input value— require very large output areas.

People with very large real memory sizes and smaller datasets may require less than 64 megabytes.

Determining How Much Swap Space You Have

To see how much swap space is available to your system, type the following command:

```
/usr/sbin/swap -s
total: 9088k allocated + 1500k reserved = 10588k used, 32724k available
```

Add the last two figures together to find the total swap space available on the system, both in the system swap space partition and any additional swap space that might have been defined for the system on other partitions. In this case, the system has approximately 43 megabytes of swap space (10588k + 32724k).

Running Out of Swap Space: Abort or Hang

When AVS runs out of swap space, it aborts or hangs. In many cases, AVS itself receives the error during a *malloc* call. In this instance, either individual modules or the entire AVS system will abort with the following error message:

```
Failure allocating memory:
See Installation/Release Notes to increase swap space and/or
shared memory segment size
```

This message may be followed by many additional "tcp read" error messages as each module expires. Scroll to the top of the error messages to see this first message that indicates the actual cause of the failure.

In some cases, other parts of the system will generate the error and you will not see the above message.

Monitoring Swap Space and Shared Memory Segment Use

If you suspect that swap space may be the reason AVS is dying, you can restart AVS, repeat the sequence of steps that led to the abort or hang, while monitoring AVS's consumption of swap space by entering successive */usr/sbin/swap -s* commands. The *ipcs -a* command, if entered as root, will also show you the amount of shared memory in use (SEGSZ column).

How to Increase System Swap Space

To increase the system swap space size, see the Sun document *SunOS 5.x Routine System Administration Guide* in the chapter titled "Configuring Additional Swap Space." This document is available

online through the Sun Answerbook facility. Also, typing *man swap* and reading the *-a* option gives useful information.

The procedure involves optionally using the */usr/sbin/mkfile* command to create a swap space file, then using the */usr/sbin/swap -a* command to add the swap space. To make the increase in swap space permanent rather than just for the current session, you must also edit the */etc/vfstab* file. Again, see the "Configuring Additional Swap Space" chapter in the *Routine System Administration Guide* and the *swap* man page for more information.

If you are supporting diskless workstations on a central server, you may need to increase the swap space available to each client workstation that will run AVS.

Shared Memory Segment Size

AVS makes heavy use of shared memory to improve performance. Shared memory reduces the amount of memory AVS needs, and makes AVS run faster because less data must be copied between modules.

Shared Memory Problems: shmget failed Error Messages

If shared memory is not enabled on your system, or if the amount of shared memory AVS requests is larger than your system limit, or if the number of shared memory segments requested is larger than your system limit, you will see a message like the following:

```
shmget failed: invalid argument
```

AVS then falls back on making individual copies of the datasets in each module and the kernel instead of using shared memory, consuming substantially more real memory, swap space, and time copying the data between modules and the AVS kernel.

Finding Out If Shared Memory Is Enabled and Increasing Your System Limits

As SunOS 5.7 comes "out of the box," there are several shared memory parameters that may not be configured correctly:

- Shared memory may be altogether disabled (this is often the default).
- Any one of these limits may be too small:

shmmax

This is the maximum shared memory segment size (in bytes) that a process can acquire in a single request. The documentation states that the default is 131072 bytes. This is far too small for AVS.

For example, a 512x512 image requires 1 megabyte to store. A 40x32x32 5-vector single precision floating point curvilinear dataset with accompanying points information requires around 1.3 megabytes. Many datasets are much larger. AVS modules usually request at least as much memory as the size of the input dataset, and often considerably more. **compute gradient**, for example, needs 12 output bytes for every input data value. This will result in shared memory failing for large datasets.

shmseg

This is the maximum number of shared memory segments that a single process can have. The documentation states that the default is 6. This is, again, far too small for AVS.

To see if shared memory is turned on, type this command:

```
ipcs -a
```

If you see a message, "Shared memory facility not in system...", then shared memory is either not turned on or has not been used by the system. You turn it on by establishing the limits in */etc/system* described in the next paragraph.

If the **shmseg** is less than about 30, and the **shmmax** size is less than at least 10 megabytes, you should raise these limits. People with very large, complex networks may need a **shmseg** limit greater than 30. People with larger datasets, and who use modules with large output sizes will need a larger **shmmax** size. (There is no apparent disadvantage to setting limits extremely high.)

To enable shared memory and/or modify its parameters, edit the */etc/system* file as root:

- Add lines like the following (or edit the existing lines to have these values):

```
set shmsys:shminfo_shmmax=65536000
set shmsys:shminfo_shmmin=1
set shmsys:shminfo_shmmni=100
set shmsys:shminfo_shmseg=32
```

- Reboot the system to have the changes take effect.

Using AVS on Sun SPARCstations

The following sections describe in detail the differences between using AVS on a SPARC workstation and AVS as it is described in the standard documentation.

Peripherals

Dial Box

The Dial Box I/O device is supported in SunOS 5.7.

Spaceball

The Spaceball I/O device mentioned in the *AVS User's Guide* is supported in this release of the product.

Stereo Support

Both versions of AVS 5.5 (OpenGL and XGL) include basic support for stereo viewing using CrystalEyes stereo goggles from StereoGraphics. See the "AVS 5.5 on UNIX Platforms" chapter for more general information.

Stereo needs to be enabled before the X server is started up; if stereo is not enabled you would need to do the following:

- Exit from the X server using the CDE "Exit" button and return to the main login prompt.
- Under the Options menu, select "Command Line Login"; hit Enter or Return to get a new login prompt and login as root.
- Type "ffbconfig -res stereo try" to attempt to switch into stereo. This will switch into a possibly lower resolution mode that will support stereo. The text on the screen may be scrambled but this doesn't seem to be a problem once the X server restarts. If you get messages about this mode not being supported for your monitor, etc. then please consult your system documentation for further information.
- Exit and hit the "Enter" button to restart the X server and log in again.
- If the screen is a smaller resolution, type "avs -size 900x700" to shrink AVS to better fit the new screen size. OpenGL stereo should now work.

- To return to the original resolution, repeat the steps above but type the following instead:

  ```
  ffbconfig -res 1280 try
  ```

Window Manager

AVS works with any of the standard X Window System window managers, including the Common Desktop Environment and Motif. No user modifications are necessary in order to use AVS with these window managers but there are some modifications that you might want to make as a matter of personal preference. See the "Window Manager" section in Chapter 3, *AVS 5.5 on UNIX Workstations* for more information on modifying how AVS behaves under Motif and OPEN LOOK.

Starting AVS

The issues here that differ from the account in the AVS *User's Guide* and the *avs* man page are: scaling the AVS interface to fit your display resolution, and setting the correct X server VisualType for your graphics adapter.

All involve making changes to your personal AVS startup file (*.avsrc*) in your HOME directory. If you don't already have a *.avsrc* file, copy the sample file from <*install_dir*>*/avs/runtime/avsrc* to your home directory and edit it to include the necessary lines. <*install_dir*> is replaced with the directory in which AVS is installed.

Display Size: GX, GS, SX, S24

This affects GX, GS, SX, and S24 graphics adapters.

AVS normally expects to run on 1280 x 1024 resolution monitors, and sizes its interface accordingly. The display usually supplied with GX and GS graphics adapters is 1152 x 900. Add the following lines to your personal *.avsrc* file or use the *-size* command line option:

```
NetworkWindow          850x850+250+20
ScreenSize             1080x864
```

to size AVS reasonably for your monitor. ZX, Creator3D, Evans and Sutherland Freedom, GT and GXplus adapters have 1280 x 1024 monitors.

The AVS "-size" command line option provides the same functionality that the ScreenSize startup option does and can be used to experiment with the best screen size.

Visual Type: Pseudocolor on Truecolor Adapters

This affects GS, GT, ZX, SX, Creator3D and Freedom graphics adapters. It may also affect other 24-plane graphics adapters that you may have, such as the S24.

No matter which graphics adapter is present in your SPARCstation, the X server on SPARCstations always has a PseudoColor visual (as reported by the *xdpyinfo* command) as its default visual. By default, AVS always uses the default X visual when it starts.

On GX graphics adapters, this is fine. AVS will use the correct PseudoColor visual for this 8-bit platform.

However, on the graphics adapters mentioned above, AVS will also come up using the 8-bit default PseudoColor visual. To force AVS to use the 24-bit TrueColor visual that these adapters are capable of supporting, add this line to your *.avsrc* file:

```
VisualType      TrueColor
```

You may also want to use the new *-vistype* command line option documented in the "Unexposed command line options" section of the UNIX chapter. For information on checking and changing your default visual, see Chapter 3, *AVS5.5 on UNIX Platforms*.

Changing the OpenWindows Default Visual Permanently

You can also force OpenWindows to use a particular X Window System default visual when it initializes.

In your sequence of startup files, find the line that starts OpenWindows. For a "vanilla" user that starts OpenWindows when they login, this is usually the line "*/usr/openwin/bin/openwin*" in the user's *.login* file.

Change this line to read:

```
/usr/openwin/bin/openwin -dev /dev/graphicsadapter defdepth 24
```

where *graphicsadapter* is the id of the graphics adapter in your workstation, as described in the "Graphics Hardware" section above. This practice should not, however, be necessary and is not recommended.

Loader Messages at Startup

AVS 5 was dynamically linked against certain system libraries. These libraries must be present for AVS to start, if they are not you will see a message of the form

```
ld.so.1:  avs:  can't find libGL.so.1
```

or if you are using *avs.xgl*, a message like this:

```
ld.so.1:  avs:  can't find libxgl.so.3
```

See the "Graphics Libraries" sections above for information on how to proceed.

Image Viewer

All functions and behavior in the Image Viewer are as described in the AVS *User's Guide*. The Image Viewer will use 8-bits of color on GX graphics adapters, and 24-bits on graphics adapters using the TrueColor visual.

On pseudocolor systems, the Image Viewer's **Images** submenu will include a choice of dithering options.

Network Editor

The Network Editor functions as described in the AVS *User's Guide* and *Module Reference* manual, with these few exceptions.

Modules

The following modules documented in the *AVS Module Reference* manual do not appear in the AVS release for Sun SPARCstations because they require rendering features that are not supported in this release:

alpha blend (requires hardware-assisted alpha transparency)
transform pixmap (requires 2D texture mapping)

Graph Viewer

Other than the fact that image data loaded as a background to a plot on a GX graphics adapter will be dithered to 8-bit pseudocolor rather than appear in true color, there are no differences between AVS Graph Viewer behavior on SPARCstations and the Graph Viewer descriptions in the AVS documentation.

Geometry Viewer

The Geometry Viewer (or the **geometry viewer** module) will be using either the software renderer or the hardware renderer to produce the contents of its scene windows.

OpenGL differences

The default AVS executable *avs* provides hardware rendering using OpenGL does not support several Geometry Viewer controls that were custom implemented for XGL renderer features. The following controls will not appear in the OpenGL executable:

- Normal Mapping
- VLTM
- Adaptive Subdivision
- Nsegs Subdivision
- Combined Subdivision
- Global Anti-Aliasing

Color and Single/Double Buffering

On the 24-plane GS system, XGL uses a 12-plane true color visual. Thus, when you are in the default double buffer mode, you are seeing only 12 planes of color (4096 possible colors), rather than the full 24 planes (16,777,216 possible colors). To see a full 24-plane true color rendering, you must change to single buffer mode by switching off the **Double Buffer** button on the **Cameras** submenu.

There is, at present, no AVS startup option that will automatically switch off double buffering.

"Double buffer" means that XGL renders its output image into invisible planes, then makes those planes visible, switching back and forth between two sets of planes. "Single buffer" means that XGL renders its output image directly to visible planes. If single buffer mode is set, you can watch the object being drawn; if double buffer mode is set, the completed picture simply appears in the window.

OpenGL does not provide support for the GS and GT graphics adapters. See *http://www.tgs.com* for alternate OpenGL drivers for boards not supported by Sun.

Rendering Features

The following table lists the rendering features described in the "Geometry Viewer" chapter of the *AVS User's Guide*, including AVS 5 features down the left column, and the AVS software renderer and the AVS hardware renderer's XGL implementation across the top. The table intersections show which features are present on each platform, and draw your attention to more detailed explanations of behavior later in this section.

Table 8-1. Geometry Viewer Behavior Across Platforms

Rendering Feature	Software Renderer	Hardware (XGL)	Hardware (OpenGL)
Arbitrary Clip Planes	yes, 8 planes (note 1)	yes	no
Geometric			
Volume Rendering	yes (note 2)	no	no
Vertex Transparency	no (note 3)	no	no
Vertex Colors	yes (note 4)	yes	yes
Edit Property			
RGB/HSV Colors	yes (note 4)	yes	yes
Ambient	yes	yes	yes
Diffuse Lighting	yes	yes	yes
Specular Highlights	yes	yes	yes
Gloss	yes	yes	yes
Transparency	yes (note 5)	yes/no (note 7)	no
Metallic	yes	yes	yes
Edit Texture	yes	yes	yes
2D Texture Mapping	yes	yes (note 8)	yes
3D Texture Mapping	yes	no	no
Filtered Textures	no	yes	yes
Tile Textures	no	yes	no
Alpha Textures	yes	yes	yes
Rendering Options			
Points	yes	yes	yes
Lines	yes	yes	yes
Smooth Lines	no	yes/no (note 9)	no
No Lighting	yes	yes	yes
Flat Shading	yes	yes	yes
Gouraud Shading	yes	yes	yes
Outline Gouraud	yes	yes	yes
Phong Shading	no	no	no
Backface Properties			
Cull front	no	yes	yes
Cull back	yes	yes	yes
Flip normals	no	no	no
Lights			
Number of Sources	16	8/16 (note 11)	8
Ambient	yes	yes	yes
Directional	yes	yes	yes
Bi-Directional	yes	yes	yes
Point	no	yes	yes
Spot	no	yes	no
Light Colors	yes	yes	yes

Table 8-2. Geometry Viewer Behavior Across Platforms (continued)

Rendering Feature	Software Renderer	Hardware (XGL)	Hardware (OpenGL)
Cameras			
Depth Cue	yes	yes	yes
Global Anti-Alias	no	yes / no (note 13)	no
Accelerate	no	no	no
Perspective	yes	yes	yes
Axes for Scene	yes	yes	yes
Front / Back Clipping	yes	yes	yes
Sorted Transparency	no	no	no
Polygonal Spheres	yes (note 6)	no (note 10)	no
Double Buffer	no	yes / no (note 12)	yes
Stereo	no	yes (note 13)	yes (note 13)
Labels			
Drop Shadow	no	no	no
Title	yes	yes	yes
Stroke	no	no	no
Kanji	yes	no	no

Notes

1. Arbitrary clipping planes are used by the **clip geom** module.

2. "Volume Rendering" is here defined in a very narrow sense: support of a specific option to the **GEOMedit_texture_options** call that is used by the **volume render** module.

3. Vertex transparency is used by the **colorize geom** module.

4. When using the software renderer, the number of colors that will appear on the screen (216 pseudocolor, 16,777,216 true color, or some number in between) is dependent upon the X server visual support present on the *display* hardware. Internally, 24-bit true color is always used and will be output on the **geometry viewer** module's image output port.

5. The software renderer supports two-pass transparency. Two-pass transparency correctly renders a transparent surface that is over an opaque surface. (Multi-pass transparency is required to correctly render multiple overlapping transparent surfaces.)

6. The software renderer is able to render spheres directly instead of subdividing them into polygons. This saves a tremendous amount of memory when rendering spheres. The algorithm's

execution speed is bound by the size of the spheres it has to render. This feature may be disabled by toggling the **Polygonal Spheres** button under the Cameras menu. Spheres will now be rendered by subdividing them into polygons.

7. The XGL hardware renderer supports two-pass transparency on the ZX, SX and GT adapters, but not on the GS. The SX does this in software so you can expect some performance degradation on that system when there are transparent objects in the view.

8. XGL currently does all texture mapping in software and there is a considerable performance degradation when using this feature, particularly with large (greater than 256x256) texture images. There are also issues regarding texturing quality on non-planar surfaces. Sun Microsystems is addressing these issues. Please call AVS Customer Support for the latest information if this is of concern to you.

9. The XGL hardware renderer supports antialiased lines on the ZX, SX and GT adapters, but not on the GS.

10. Spheres are subdivided into polygons when using the XGL based hardware renderer.

11. The GS adapter supports 8 light sources, all others support 16.

12. The GS and S24 adapters do 12 bit dithered double buffering. Toggling the **Double Buffer** button *off* will render geometry in 24 bits non-dithered. The double buffer button will be absent on all other adapters since they render at full 24-bit resolution when double buffered.

13. Supported on the (T)ZX and Creator3D adapters. You will need to configure the frame buffer device for stereo viewing, see the *leoconfig* and *ffbconfig* commands, respectively.

Programming Considerations

AVS, it's modules, and component libraries were compiled using the ANSI C SPARCompiler *cc* version SC4.2. Executables were linked using libraries located in */opt/SUNWspro/lib*. FORTRAN components were compiled using the FORTRAN *f77* SPARCompiler version SC4.2. Executables were **dynamically** linked against system libraries.

Compiling and Linking Modules

You should use the make file in *<install_dir>/avs/examples/Makefile* as the template for your own FORTRAN and C module make files. (*<install_dir>* is the directory in which AVS is installed. This make file in turn includes the file *<install_dir>/avs/include/Makeinclude* that contains additional macro definitions appropriate to the Sun Solaris platform.

Static Versus Dynamic Linking

AVS 5 was linked dynamically against system libraries. The STATICFLAG symbol in *<install_dir>/avs/include/Makeinclude* is set to NULL, making dynamic linking the default when you compile and link modules.

The module user may see error messages stating that the system cannot locate certain libraries. In this case the LD_LIBRARY_PATH environment variable can be modified to include the path of the versions of the missing shared object libraries.

AVS on SunOS 5.7: Known Problems

This section lists the known problems with this release of AVS 5.5 that are unique to SunOS 5.7 on SPARCstation platforms.

X8689: SUN ZX, GT: Layout Editor widget outline invisible

Problem Description:

When moving a widget from the main panel onto a page, the widget outline is invisible when it is displayed in windows of depth 24 (which includes all AVS windows). This is a problem when using the TrueColor visual for AVS.

Workaround: A partial workaround has been implemented within AVS. When using the TrueColor visual (on the ZX and GT only) widget outlines are drawn using the parent window visual instead of the 8 bit deep RootWindow. This means the outline will be visible only while in its parent window. So to place a widget within a page, for example, move the widget to the page and release the mouse button so that the widget is reparented to the page. Now move the widget again and its outline will be visible while in the page.

X10615: SX: AVS logo is random when using TrueColor Visual

Problem Description:

The AVS logo is usually not drawn correctly on the SX when

using the TrueColor visual.

Problem: Label color affected by Perspective and Depth Cueing

Problem Description:

Use of Perspective and Depth Cueing in a view may affect the color of labels in the same view.

AVS on SunOS 5.x: Fixed Problems

This is the list of fixed problems for AVS 5.5 on the SunOS 5.7 on SPARCstation platforms.

17582: Elite 3D graphics adapter support

Problem Description:

Support has been added for the Elite board. Customers may require the following patches to work properly: 105363-06, 105361-04 and for OpenGL support, 106022-02.

EXTENDED FEATURES

CHAPTER NINE

Overview

AVS 5.5 is primarily a platform maintenance release; its main purpose is to update AVS to run on major new operating system releases and provide critical bug fixes. For an overview of what is specifically in the AVS 5.5 release, please see the "Release Highlights" in Chapter 1.

AVS5.5 NOTE: The AVS_DEMOS have been revived and retested on all platforms and a number of bugs have been fixed. The BTF renderer (AVS/Voxel) has been unlicensed to help simplify AVS 5 licensing. AVS/Graph has been updated to use Toolmaster 7.1 for its underlying graphics support - most platforms previously used Toolmaster 6.5A.

This chapter also describes the Cool CD and UCD Builder materials provided with AVS 5.3 and other features added in earlier releases of AVS 5, such as AVS/Graph, AVS/Voxel, Japanese Online Help, and the AVS Demos. Fixes for known problems are discussed in the introductory UNIX chapter and the chapters for the specific platforms.

Cool CD and UCD Builder

The Cool CD is an extensive collection of public domain and 3rd party modules and data sets, conference proceedings, and other valuable material to expand your use of AVS 5 products.

Much of the Cool CD comes from the International AVS Centre (IAC) through the contribution of users like yourselves who have developed AVS 5 modules and data sets and offered them to the AVS user community. The IAC, now based at the University of Manchester in England, makes these modules available in source form through its World Wide Web site. The Cool CD delivers them to you for convenience but make sure to visit the new IAC Web site for ongoing contributions and updates at *http://www.iavsc.org*.

Additional 3rd party modules are also on the Cool CD, including modules that help you -

- Read and write VRML (Virtual Reality Markup Language), an emerging 3D modeling language for the World Wide Web
- Read input from FEA and CFD applications to create UCD data structures (UCD Builder)
- Read I-DEAS FEM data (FEMbridge) or write data to a V-LAN based video recorder (VIDEOkit), available for evaluation or purchase through KGT, Inc.

Additional information on other third party AVS5 and AVS/Express modules and applications available through other companies is also included.

The Cool CD also included a snapshot of the corporate website for Advanced Visual Systems; this has become substantially outdated since the CD was last updated. Please visit our online website at at *http://www.avs.com*.

The Cool CD is specially formatted to allow you to browse the contents with your favorite Web browser without having to install anything. Just mount, browse, and choose what you want to use; in some cases, the software can be run off the CD with no installation.

Support

Because the Cool CD contents are primarily from third parties and are not part of the AVS 5 product line, AVS Customer Support will not support them or answer questions about them. The contents of the Cool CD are provided "AS IS", with no warranty or support of any kind. We are making them available to you as a service to make it easier for you to explore and experiment with the wide world of public domain AVS 5 modules available.

If you do have problems, you should contact the appropriate third party for help or suggestions. For modules from the International AVS Centre, start by contacting the IAC directly or the original module contributor if they have provided contact information.

For most of the third Party modules (FEMbridge, VIDEOkit, VRML reader and writer), follow instructions on the Cool CD to contact KGT, Inc. for further information.

The UCD Builder product was developed by Scientific Visualization Associates Inc. Working with Advanced Visual Systems, the company is making its product available to you in binary form for free as a contribution to the AVS 5 community. This is an unsupported product and Advanced Visual Systems assumes no responsibility for it; if you do have problems or questions about it, see the related Web page for instructions.

AVS 5.5 NOTE: The UCD Builder has **NOT** been updated since AVS 5.3; however, modules should be upward compatible with the newer OS versions supported by AVS 5.4 and AVS 5.5.

AVS/Graph

AVS 5.5 NOTE: The AVS/Graph module has been updated to the latest version of Toolmaster 7.1.

AVS/Graph is a graphing tool which allows you to plot AVS field data in a variety of ways (curve, scatter, bar, area, polar, pie, images), control the appearance of the graph (titles, axes, logarithmic scales, legends, tick marks) and generate hardcopy output in a variety of formats (PostScript EPS, PostScript Color EPS, CGM Binary, CGM Clear Text, etc.) AVS/Graph consists of the Data Output module named **AVS/Graph** with its own extensive user interface, a set of demonstration scripts, and a set of example networks.

The **AVS/Graph** module is a part of the Supported AVS Module Library. The module is a synchronous coroutine module (i.e., it runs as an independent process that is triggered when one of its inputs changes).

AVS/Graph is implemented using the Advanced Visual Systems product Toolmaster-agX. The source to the module is included with the release. If you also purchase Toolmaster-agX, you can use the module source as a guide to extending AVS/Graph functionality to perform additional tasks such as contouring, data interpolations and smoothing, 3D graphs, and note and arrow annotations.

For complete AVS/Graph documentation, see the *AVS/Graph User's Guide* included in the standard doc set.

Japanese Online Help

This release contains a complete set of AVS 5 online help in Japanese. All of the material in *runtime/help* has been translated, including the module man pages and the help files for the various subsystems. In addition, users can create their own Japanese language online help files to support their applications.

The k14 and a14 X fonts must be present on your system.

To install Japanese Online help:

1. When the *install.avs* script presents its list of installable products, select AVS_JHELP. This one help directory supports all platforms and takes approximately 3 megabytes of disk space.

The installation creates an *<install-dir>/avs_jhelp* directory. This directory can be anyway in the file system hierarchy, both within the *avs* installation directory or outside it.

2. With the AVS_PATH environment variable set to point to the AVS directory, execute the script *<install-dir>/avs_jhelp/test/SETUP*. If unset, AVS_PATH defaults to */usr/avs*.

This script adds the correct fonts to the *runtime/avs.Xdefaults* file and sets the AVS_HELP_PATH environment variable to point to the Japanese help files.

3. You can check that the installation went correctly by executing the <install-dir>/avs_jhelp/test/RUNME script.

For information on creating your own Japanese online help files, see:

- *test/kanji.eucf*: Man page that describes how to make a module with a Japanese help file. (This file is in Japanese).
- *test/kanji.scr*: AVS script for making a module with a Japanese help file.
- *test/kanjiall.txt*: The table of EUC codes.

AVS/Voxel

AVS/Voxel is a Back-to-Front (BTF) direct volume renderer consisting of a set of modules provided in the Unsupported library for customers to use and experiment with. **NOTE:** The BTF renderer is not provided on Compaq Tru64 UNIX.

The BTF renderer renders volumes one slice at a time, beginning from the back of the volume, and composites each new slice as it renders the volume. It is based on a high performance algorithm that achieves its speed by exploiting memory coherence and by optimizing cases involving transparency. In addition, BTF has support for a binary occupancy volume that increases rendering performance for cases that involve clipping, region growing, and segmentation. This volume renderer is extremely well suited for data sets where much of the data in the volume will be transparent (e.g. seismic interpretation and medical imaging).

The BTF renderer is a set of modules that are currently placed in the Unsupported library. It consists of the following modules:

btf shade
: Performs back-to-front (BTF) volume rendering. This module takes a volume, which can be visualized as a block of cubic "voxels" (volume elements), and generates a 2D image using a back-to-front

(btf) direct volume rendering technique.

btf anim
: Performs back-to-front (BTF) volume rendering like **btf shade** except that it allows the user to select a region of the volume and show intermediate rendering results, layer by layer to more clearly see the internal structure. Provides parameters to select starting and ending layers, number of steps between views, and run control.

btf bitvol
: Creates a bit volume for use with **btf volume** renderer. This module generates a bit volume from a 3D scalar byte or 16-bit integer field using the opacity channel from a colormap. The rendering speed of the **btf shade** and **btf anim** modules can be increased substantially by using the **btf bitvol** module. Performance can be increased by a factor of 2-4 depending on the occupancy level of the volume, i.e. the number of non-zero voxels in the volume.

norm8 encode
: Computes gradient vectors for 3D data sets and encodes them into bytes. The **norm8 encode** module computes the gradient vector at each point in a 3D field of data. The gradient vector can be used as a "pseudo surface normal" at each point. This vector is then encoded into a byte and packed into a 16-bit integer value along with the original input value.

norm8 table
: Generates a lookup table for an encoded gradient. The **norm8 table** module accepts an optional transformation matrix as input. It builds a 256-entry intensity lookup table using the ambient and diffuse coefficients and the light source direction for all possible values of the encoded normal information (gradient vector). This table is then used during the shading process by the **btf shade** and **btf anim** modules to interpret the gradient information that was encoded into a byte by the **norm8 encode** module.

Using AVS/Voxel

On platforms that support AVS/Voxel, the BTF modules are located in the Unsupported library. There are several demonstration scripts that can be found in the *$AVS_PATH/demo/man_scripts* directory, all named with titles beginning with "BTF volume rendering". These are found by selecting the **Help** button and then selecting **Help Demos** to bring up the Script Controller which presents a list of demonstration scripts.

The BTF modules no longer require any additional licensing from Advanced Visual Systems. You may see a "AVS/Voxel" entry in your AVS5 license but this is no longer needed for AVS 5.5. You may wish

to retain this feature line for older versions of AVS 5 still in use at your site.

Documentation

The man pages for the BTF modules are available online in the *$AVS_PATH/runtime/help/modules* directory along with the other AVS 5 modules. Press the module's dimple to get the Module Editor panel and then press "Show Documentation".

Demos

This release provides an optional package of AVS Demos that can be loaded from the CD after AVS has been installed. These demos can be used to see additional ways that AVS can be used and to obtain new demonstration datasets to augment those found in *$AVS_PATH/data.* Most of these datasets and demonstrations have been contributed by AVS users for the benefit of the AVS user community. While our users have permission to use them for demonstration purposes, they are not necessarily public domain and should not be used in products outside of AVS without express written permission from Advanced Visual Systems Inc.

AVS5.5 NOTE: These demos have been rebuilt and retested under all AVS 5.5 platforms and problems have been addressed.

Installing the Demos

You will require approximately 25 Megabytes to unload the Demos from the CD and another 5 to 10 Megabytes to build them for a total of 50 Megabytes. The demos are NOT precompiled but must be built in order to work.

1. Install AVS if you haven't already done so. The Demos will expect to reference a standard AVS product in $AVS_PATH (*/usr/avs* by default).

2. Install the **DEMOS** product from the AVS CD using the *install.avs* script used for installing the standard product. They will be installed into a directory called *avs_demos.*

3. If you are installing the demos from a remote login window you must make sure that your DISPLAY environment variable is set to an open display, e.g. "setenv DISPLAY unix:0.0". The installation procedure will attempt to run *avs* to run some data conversion scripts and this will fail if there is no DISPLAY set. **NOTE:** If the remote system is an SGI and the system you are on is not, you need to set the **__SGI_NO_REMOTE_GL** environment variable to avoid problems running *avs* remotely.

4. Change directory (*cd*) to the *avs_demos* directory and run the *install_demo* script found there to install a reference (link) from the *$AVS_PATH/demosuite* area to the place where the Demos are located. The *install_demo* script will do the following:

    - Move aside the existing *$AVS_PATH/demosuite/demo_menu* file to *demo_menu.orig* and install a new version of the *demo_menu* file with the Demos scripts appended as an "Extended" menu.

    - Create a link from *$AVS_PATH/demosuite/DEMOS* to the *avs_demos* directory. This link is required for making the Demos files since some of them expect to find *DEMOS* in *$AVS_PATH/demosuite*.

    - Offer to "make" the Demos. You should reply "yes" to immediately make the modules and local data files that make up the Demos.

Once the Demos have been "made", the *install_demo* script can be used to have multiple systems point to the same demo area. Just re-run *install_demo* and answer "no" when asked if you want to "make" the Demos again.

NOTE: Many of the demos need to be "made" differently for different platforms. For example, an SGI and a SUN can *not* share the same area. The file *DEMOS/PLATFORM* describes what platform the Demos are currently built for.

Uninstalling the Demos

Change directory (*cd*) to the *avs_demos* directory and run the script *uninstall_demo* which will do the following:

- Remove the Demos version of *$AVS_PATH/demosuite/demo_menu* and move the original *demo_menu.orig* back into place.

- Remove the link from *$AVS_PATH/demosuite/DEMOS* to the *avs_demos* directory.

- Offer to remove the entire current directory (assumed to be the *avs_demos* directory).

Running the Demos

Once installed, the Demos are accessible as a top level menu in the standard AVS demosuite. To run them, run *avs*, select **AVS Applications** and then select **AVS Demo**. The Demos should appear as a new

menu called **Extended** and are organized by different market segments. Each demo consists of one or more demonstration scripts.

The following demos are part of the AVS Demos package:

Medical Imaging

Helen's Neck

MRI (Magnetic Resonance Imaging) study of Helen's neck. The data set consists of eleven slices of 256x256 images taken sagittally. On slice 5, you can clearly see her brain, spinal column, and the vertebrae surrounding the spinal cord. This is a real life example based on a study performed on a friend of Advanced Visual Systems Inc.

Brain

Stereotactic Radiosurgery: This demonstration shows the planned treatment for a real patient. The patient was a 2 year old girl with a brain tumor located dangerously close to her brain stem and inoperable by conventional surgical methods. Provided by Dr. Hanne Kooy at the Dana Farber Cancer Institute.

Geographic Information Systems (GIS)

World

This demo shows outlines for the following global features: continental outlines; islands; lakes; rivers; country borders; United States borders.

FlightPath

The **flight path** module takes a scatter field position list and animates the camera path along the trajectory. The modules is derived from *$AVS_PATH/examples/camera.c*. The scatter path field is in the form of "field 1D 3-space irregular float" where the coordinate information in the field specifies the path. The data values are ignored. It moves %top, not the camera, so multiple camera views can be set up to watch the path. The **file_descriptor** or **read_field** modules can be used to get the path from an external file. The **animated integer** module can be used to control time, for a moving sequence.

Terrain

Terrain Reconstruction from Digital Elevation data. This demo shows an elevation map and a landsat image (full color) of Orange County, with a perspective view. The image is rendered with a synthetic light source direction. The texture processing is performed by the AVS software renderer, or hardware adapters that support this operation.

Seal Tracks

Southern Elephant Seal movement and dive data in the South Atlantic. Provided by the SEA Mammal Research Unit, British Antartic Survey, NERC, Cambridge, UK. Ref: Dr. Ollie Cox and team.

Meteorology

Weather

Weather Satellite Images: If you have ever watched TV weather forecasts, you probably have seen the short "movies" of the clouds moving over the earth, especially when there is a hurricane off the coast. A new satellite photo is available every hour via the Internet. We have put together a series of 12 such photos from daytime on July 21, 1992. The images register infrared light and therefore indicate temperature. These images were taken by the GOES-7 satellite and retrieved from a machine at University of Illinois at Urbana-Champaign, Illinois.

Oil and Gas

Velocity 3D

This is a demo of a synthetic 3D velocity profile field, with a "plumb line" interactive fence diagram picking module.

Intera

This is a series of representative models of oil and gas exploration projects produced by InteraView, a product of Intera Information Technologies Limited, Petroleum Production Division that uses AVS to deliver end user solutions.

Computational Fluid Dynamics (CFD)

Phoenics

PHOENICS fluid flow simulation, courtesy of CHAM (UK). The PHOENICS CFD code is interfaced to AVS for both pre-processing and post-processing. Includes Hot Electric Box, Smoke Spread in House (two versions), Cooling Tower and Turbine Blade.

Vortex

CFD RIBBONS - generate ribbon representation for streamlines. The **ribbons** module generates a set of geometric ribbons by taking the polyline output of the **streamlines** module and replacing them with finite width colored and textured polytriangle ribbons. The data set shows a 3D vortex field, simulated by researchers at NCSA.

Mechanical

Automobile

Max vonMises Stress on a Mercedes: In this demonstration, NAS-TRAN was used to compute the Max vonMises stresses for a Mercedes-Benz automobile. This finite element dataset contains 15,843 cells. Provided by Keith Redner, Scientific Visualization Associates.

Crankshaft

Displacement on a Crankshaft: In these demonstrations, stress is applied to a crankshaft. This finite element dataset contains 609 cells. Provided courtesy of Pafec, Ltd.

Chemistry

Fast Animate

Water over Clay: Dr. Keith Refson, University of Oxford, UK uses a molecular dynamics program to simulate the interaction of water and cat ions with layers of clay. The simulation requires significant processing per time step, so it is run offline with the molecules' positions at each time step saved into files. This demonstration shows Dr. Refson's AVS animation module.

BGF Ribbon

The code in this directory is for the **PROTEIN** module. This module reads a Biograf file and creates various displays of the molecule.

Statistics

City Scape

City Scape Data Analysis Display: Network Line Statistics are displayed using a 3D Bar-Chart technique, called "City Scape". The height and color of each cell shows two variables, while row and column averages are computed dynamically and shown as projected side panels. Provided by Advanced Visual Systems, Inc.

Jigsaw

The **jigsaw** module is a tool to show irregular boundary regions, defined in 2D, like map boundaries, and apply data to loft them into 3D with data defined height, and to apply color to indicate a second variable. Provided by Advanced Visual Systems, Inc.

Magnetics

Vector Fields

Vector Fields Magnetic Flux Modeling: This structure represents the result of an electro-magnetic flux analysis using "Tosca" from Vector Fields Ltd. The magnet core is shown as a ring structure, with air cells surrounding, with interpretive vector display modules. The model is axial-symmetric, so a geometry duplication method is used to construct the complete model for viewing. The Vector Fields interface, this network, and the visualization project is by Janet Haswell, Rutherford Appleton Labs, Oxfordshire, UK.

Texture

Texture Sampler

Sample texture image files: A set of image files containing textures such as rock, sand, and clouds. Contributed by Evans and Sutherland.

ADMINISTERING LICENSES

CHAPTER TEN

Introduction

This chapter provides instructions for administering AVS 5.5 licenses with the Flexible License Manager (FLEXlm).

AVS 5.5 NOTE: No significant licensing changes have been made for AVS 5.5 except that the AVS Animator and BTF renderer modules no longer require a special license. Existing AVS 5.4 licenses will continue to work without new licenses or changes in the license daemons used to support licensing.

This chapter discusses:

- About licensing
- How licensing works
- Installation issues
- Licensing configurations
- License administration tools
- What can go wrong

About licensing

You must have a license from Advanced Visual Systems in order to run AVS 5.5 which is a licensed product of Advanced Visual Systems Inc. Access to AVS is controlled by the Flexible License Manager (FLEXlm) network-wide floating licensing system provided by Globetrotter Software. The FLEXlm system, which is supplied with the AVS 5.5 product, consists of licensing software, several utility programs that support the licensing mechanism, and associated documentation.

When correctly installed and functioning, the licensing mechanism is largely transparent to the AVS end user. However, licensing does

require some installation procedures (as described in Chapter 2, *Installation*, "Step 6: Install the AVS license"), and may require some additional installation and administrative tasks, depending upon the particulars of your installation.

How licensing requirements vary

The installation and administration of licensing varies from site to site. These differences depend primarily on two issues:

- whether other software products on your network are also using the FLEXlm licensing system, and
- the kind of license you are using: node-locked, floating, or some combination.

Multiple FLEXlm software vendors

A relatively complex licensing scenario arises when more than one software product is using the FLEXlm licensing system to administer licenses in a networked environment. On Sun networks, for example, use of the SunSoft compiler(s) is controlled by FLEXlm.

If you are using AVS 5.5 on a system where other products are using FLEXlm, you need to install and maintain AVS licensing carefully so that it won't conflict with other FLEXlm users. More information on this situation can be found below in *Multiple vendors using FLEXlm*.

Using node-locked versus floating licenses

There are two kinds of licenses available for Advanced Visual Systems products: node-locked (or fixed) licenses and floating licenses.

A node-locked license is tied to a specific host system. Up to *n* simultaneous users can run the AVS product, where *n* is the number of licenses purchased. The product can execute only on a single host workstation. This kind of license is appropriate either to a timesharing model (one powerful central system with many X terminals connected to it), or to a single independent user with his or her own workstation.

In contrast, a floating license allows any workstation in a network to execute the product, up to *n* simultaneous users, where *n* is the number of licenses purchased. Floating licenses are appropriate on a network with many users, each with his or her own workstation, each of whom wants to run AVS locally. Rather than buying a license for each workstation, a floating license provides a "pool" of licenses that users share.

For the most part, license installation and administration is identical for both types of licenses. Both require that a specific system (or specific group of systems) be established as the license server, because licensing is administered in a client/server model; that is, a license *daemon* running on the server system handles all license requests from other systems in the network. The license daemon must run continuously on the server system, and workstations that are running AVS must be able to communicate with the server.

The license.dat files for node-locked and floating licenses are only slightly different. In a node-locked license.dat file, the machine id of the node machine appears at the end of the **FEATURE** line. In most cases, the node and license server will be the same machine.

License administration options and tools

The FLEXlm licensing system provides a great deal of flexibility in regulating access to AVS. The license administrator on your system can use a variety of tools and procedures to customize your licensing setup. Specifically, you can:

- customize the license.dat file to control access to and use of AVS (see "license.dat control options" below)
- monitor and administer licensing use and behavior by using the various license administration tools provided by Advanced Visual Systems and FLEXlm (see "License administration tools" below).

One particularly useful tool that AVS provides is a licensing diagnostic utility called *licdiag*. This utility, which is installed in the *$AVS_PATH/bin* directory, is a shell script that examines your license file and licensing configuration, troubleshoots licensing problems, helps you to start and stop license daemons, and makes various recommendations depending on your platform. Many of the procedures described here and in *Installation*, "Step 6: Install the AVS license", can be taken care of more easily by using the *licdiag* utility. For a complete description of *licdiag*, see below.

Documentation for FLEXlm

Manpages, message files, and other documentation for the FLEXlm licensing system is installed in the *$AVS_PATH/license* directory when you install AVS 5.5.

How licensing works

The FLEXlm licensing mechanism works on the client/server model. The server process is the FLEXlm license manager daemon, named *lmgrd*; the AVS kernel process is the client. A file named *license.dat*

contains the information that the *lmgrd* daemon uses to determine whether you are a licensed user of AVS and, so, have permission to start up the AVS kernel process.

When you first type the command "avs" to start the AVS kernel you start a client process named avs. This process looks for the *license.dat* file. When it finds this file, the avs process contacts the license manager daemon, *lmgrd*, which is continuously running on a server system somewhere in your network. The *lmgrd* daemon then references the *license.dat* file to locate and contact the *avs_lmd* vendor daemon. *lmgrd* passes the license request to *avs_lmd*, the process which actually dispenses AVS licenses.

The *avs_lmd* process determines if a license is available. If so, then *avs_lmd* notifies *lmgrd*, which then returns permission to execute, and the avs process starts up. If no licenses are available or if, for any reason, this licensing sequence fails (if, for example, the *lmgrd* license daemon cannot be found), avs will not start up.

When you exit from avs - whether normally or abnormally with Ctrl-C - the license is "checked back in", thereby available for use by another avs process.

The license.dat file

The administration of AVS licenses depends on the contents of your *license.dat* file. All of the processes - *avs*, *lmgrd*, and avs_lmd - *read the information in license.dat* to locate and verify all steps in the licensing procedure. The information contained in *license.dat* is slightly different for each kind of license, but its use as the central "database" for assigning licenses is the same.

You obtain your license.dat file from Advanced Visual Systems when you purchase a license for AVS 5.5.

The lmgrd license daemon

In the client/server model used to administer AVS licenses, the *lmgrd* license daemon is the master server process that handles all licensing requests. One *lmgrd* process runs continuously on a host server system in your network; all licensing requests (that is, all attempts to start an avs process) are passed to the *lmgrd* license daemon.

In a case where multiple software vendors on your network are using FLEXlm for license administration, *lmgrd* handles all license requests (not just those from avs). See "Multiple vendors using FLEXlm" for details on how to handle this case.

The executable file for the *lmgrd* license daemon is installed during the AVS 5.5 installation process. By default, it is installed in the *$AVS_PATH/license* directory. It is your responsibility to arrange for *lmgrd* start-up on the system you have chosen as your server system.

The avs_lmd vendor daemon

The *avs_lmd* vendor daemon is the process that actually dispenses AVS licenses. The *lmgrd* license daemon serves as a "clearing house" for all licensing requests; it, in turn, passes those requests along to the *avs_lmd* daemon. The *lmgrd* daemon starts up the *avs_lmd* vendor daemon (and all other vendor daemons) when it initializes.

The executable file for the *avs_lmd* vendor daemon is installed during the AVS 5.5 installation process. By default, it is installed in the *$AVS_PATH/license*.

The licensing sequence

The following represents a typical series of license transactions for establishing and dispensing an AVS license.

1. A user in the network types the avs command to start AVS.

2. The *avs* process looks for the *license.dat* file. It locates the file either in its default location, */usr/local/flexlm/licenses/license.dat*, or at the location defined by the **LM_LICENSE_FILE** environment variable, if that variable has been set.

3. In the *license.dat* file, the *avs* process finds the information necessary to locate the *lmgrd* license daemon within your network. This is the information that appears on the **SERVER** line in the *license.dat* file. avs passes the request for a license to the *lmgrd* daemon.

4. When the *lmgrd* daemon receives the license request, it reads the *license.dat* file to locate the *avs_lmd* vendor daemon. *lmgrd* uses the information that appears on the **DAEMON** line in the *license.dat* file to locate the *avs_lmd* vendor daemon, which is the process that keeps track of AVS licenses.

5. The *avs_lmd* daemon determines whether a license can be granted. *avs_lmd* keeps a tally of licenses in use compared to the total number of available licenses. If a license is available, *avs_lmd* returns authorization back to the *lmgrd* daemon which then gives authorization to the original requestor.

6. With the license granted, the avs process starts up the AVS product.

7. Throughout the running AVS session, avs makes periodic checks to make sure the *lmgrd* daemon is still running. If it loses contact with *lmgrd* (if, for example, the host server has crashed) a warning is issued and, after 15 minutes of no *lmgrd* response, the avs process terminates.

8. When the avs process exits, either normally or abnormally through Ctrl-C or a crash, the license is checked back in to the *lmgrd* and *avs_lmd* daemons so that it is available to the next AVS user.

How the license.dat file is used

All of the information necessary to administer AVS licenses is contained in the *license.dat* file that you obtain from Advanced Visual Systems when you purchase an AVS license. By default, the *license.dat* file is expected to be located in */usr/local/flexlm/licenses/license.dat*. See "If you move files" if you want to move or rename this file.

An example license.dat file

A license.dat file has the following format:

```
SERVER <hostname> <host machineid> <TCP socket>
DAEMON avs_lmd <pathname>/avs_lmd [options file]
FEATURE <FEATURE> <vendor daemon> <vers> <date> <#>
         <code> <"vendor_string"> [node machineid]
```

There may be many more lines in this file if FLEXlm is being used to administer other software products, or if you have obtained several different licenses for different Advanced Visual Systems products or versions.

SERVER: This line contains the information that the avs process uses to locate and contact the *lmgrd* license manager daemon that is running in your network. It defines which network host runs the *lmgrd* license daemon, both by name and by unique machine id. The information on this line includes:

- <hostname> is the name of the system where the *lmgrd* license daemon is running. This is usually just the local hostname (e.g., ccn), but if your network name service is not correctly configured you may need to specify the full Internet hostname (ccn.uc.edu).

- <host machineid> is the machine id of the system where the *lmgrd* license daemon is running. You will have supplied this machine id to Advanced Visual Systems when you purchased your licenses.

- <TCP socket> is the TCP/IP socket number that the avs process uses to connect to the license server. This number may be absent if the socket is already defined in the /etc/services file.

Note: If you are using redundant licensing daemons, there will be multiple **SERVER** lines in the *license.dat* file. See "Creating license daemon redundancy".

DAEMON: This line contains the information that the *lmgrd* license daemon reads in order to locate the *avs_lmd* vendor daemon.

- *avs_lmd* is the name of the Advanced Visual Systems vendor daemon.

- <*pathname*>*/avs_lmd* is the fully-qualified pathname to the *avs_lmd* executable file. By default, the *avs_lmd* executable is installed in the *$AVS_PATH/license* directory. You will need to edit the pathname on the **DAEMON** line if it is incorrect, or if you have moved the *avs_lmd* executable to another location.

- [options file] is the fully-qualified pathname to an options file - if you are using one - which contains FLEXlm local control options that fine-tune licensing for your installation. See "license.dat control options" for a description of these files.

FEATURE: These lines contain the information that the avs process and the *avs_lmd* vendor daemon use to administer licenses. The information on **FEATURE** lines define specific aspects of the licensed feature that are subject to license control; this information is used internally by both the avs process and the *avs_lmd* vendor daemon. Much of the information on this line is encrypted; do not attempt to modify this line or licensing will not work properly.

The information on a **FEATURE** line includes:

- <FEATURE> is the name of the feature being licensed. The feature name for AVS 5.5 is AVS.

- <vendor daemon> is the name of the vendor daemon associated with the licensed feature; all Advanced Visual Systems products use the *avs_lmd* vendor daemon.

- <vers> is the version number of the licensed feature. The version number for this release is 5.000 which permits you to execute AVS 5.5 and earlier AVS 5 releases.

- <date> is the expiration date of the license; 1-jan-00 means there is no expiration date.

- <#> is the number of licenses that have been purchased for the feature.

- <code> is an alphanumeric string containing encrypted licensing information.

- <"vendor_string"> is not currently used for AVS 5 products.

Note: You can find out the licensed components as specified in the vendor string by using the *licdiag* utility. See "licdiag" section below.

- [node machineid] appears only for node-locked licenses and is the machine id of the system where the feature executes. If the host server and execution node are the same (as is usually the case), this is the same machine id that appears on the **SERVER** line.

There is one **FEATURE** line for each licensed product that you have purchased. When you upgrade an AVS 5.5 license, you should remove the **FEATURE** lines associated with earlier releases.

Changing the license.dat file

The *lmgrd* license and *avs_lmd* vendor daemons read the *license.dat* file once, when they initialize. If modifications are made to the file on the fly, then you must either restart the *lmgrd*license daemon, or force it to re-read the file, either with the *lmreread* command or option #5 of the *licdiag* utility. For information about both of these utilities, see "License administration tools."

license.dat control options

The FLEXlm licensing system allows you to customize local use of licenses, arranging for things like reserved licenses and software timeouts. To make these customizations, you create an options file, then add the fully-qualified pathname of the options file to the **DAEMON** line in your license.dat file. The *avs_lmd* vendor daemon, which is responsible for dispensing AVS licenses, uses the information in your options file to determine whether a license should be granted to the requestor. If you create an options file named */usr/local/local.options*, you would modify the *license.dat* **DAEMON** line like this:

```
DAEMON avs_lmd /usr/avs/license/avs_lmd /usr/local/local.options
```

The daemon options file is a text file containing a series of keyword-based instructions. This file has the following format (where keywords are in uppercase and command options are indicated here with the use

of <> angle brackets):

```
RESERVE <number> <feature> {USER | HOST | DISPLAY | GROUP} <name>
INCLUDE <feature> {USER | HOST | DISPLAY | GROUP} <name>
EXCLUDE <feature> {USER | HOST | DISPLAY | GROUP} <name>
GROUP   <name> <list_of_users>
TIMEOUT <feature> <timeout_in_seconds>
NOLOG   {IN | OUT | DENIED | QUEUED}
REPORTLOG <file>
```

You can include comment lines in this file by putting the sharp character (#) as the first character in the line. The following keywords are supported:

```
RESERVE <number> <feature> <OPTION> <name>
```

Ensures that AVS or one of its licensed features is always available to the specified user, host computer system, specific displays, or group of users. This keyword requires a number (count of licenses to be reserved), the name of the feature being reserved (for example, AVS), the option to which this reservation applies (USER, HOST, DISPLAY, or GROUP), and the name associated with that option (username, hostname, display name, or group name). For example,

```
RESERVE         1       AVS             USER    pat
```

reserves one license for the AVS feature for the user named "pat".

```
GROUP <name> <list_of_users>
```

Specifies a group of users for use in other commands in the options file. For example,

```
GROUP           all_of_us               tom dick harry
INCLUDE <feature> <OPTION> <name>
```

Specifies a list of users, hosts, displays, or groups of users who are allowed access to AVS or one of its licensed features. This keyword requires a feature name, the option to which inclusion applies (user, host, display, or group of users), and the name associated with that option (username, hostname, display name, or group name). For example,

```
INCLUDE         AVS/Animator            GROUP   all_of_us
```

specifies that all users in the group named "all_of_us" can use the AVS/Animator feature.

```
EXCLUDE <feature> <OPTION> <name>
```

Specifies a list of users, hosts, displays, or groups of users who are prohibited access to AVS or one of its licensed features. This keyword requires a feature name, the option to which exclusion applies (user, host, display, or group of users), and the name associated with that option (username, hostname, display name, or group name). For example,

```
EXCLUDE          AVS              USER     unabomber
```

specifies that the user named "unabomber" is prohibited from using the AVS feature.

```
TIMEOUT <feature> <timeout_in_seconds>
```

Allows idle licenses to be returned, after a specified time, to the free pool, for use by another user. This keyword requires the feature name and a timeout interval, in seconds. For example,

```
TIMEOUT          AVS              600
```

```
REPORTLOG <filename>
```

Specifies that an *avs_lmd* logfile be written that is suitable for use by the Enhanced User Tools report writer. If you prepend the filename with the + sign, the file is opened for append (rather than overwritten).

```
NOLOG <OPTION>
```

Causes messages of the specified type to be filtered out of the *avs_lmd* daemon's log output. Message types that can be filtered out are IN, OUT, DENIED, and QUEUED messages.

Installation issues

There are a number of different situations that may affect how you install and administer FLEXlm licensing on your system. This section describes these issues.

Where files get installed

With the exception of the *license.dat* file, the various components of the FLEXlm licensing system are installed when you install AVS 5.5. All of these files are located in $AVS_PATH/license.

The default location for the *license.dat* file is the */usr/local/flexlm/licenses* directory of the licensing server.

If you move files

You can move most of the licensing components to other directories on your system, but if you do so you may have to make the following adjustments:

- If you move or rename the *license.dat* file, you must set the **LM_LICENSE_FILE** environment variable to point to it; for example,

  ```
  setenv LM_LICENSE_FILE /u2/avs/license/license.dat
  ```

  or:

  ```
  setenv LM_LICENSE_FILE /usr/local/flexlm/licenses/avslic.dat
  ```

  Be sure to notify everyone who will be using AVS to set this variable in their environment.

- If you move *avs_lmd*, you must modify its pathname on the **DAEMON** line in the *license.dat* file to reflect its new location.

- If you move the *lmgrd* binary to run on a system other than the host system (as specified on the **SERVER** line in the license.dat file), you must obtain a new *license.dat* file from Advanced Visual Systems. The machine id for the host server is encrypted in the *license.dat* file and cannot be modified.

Starting the lmgrd license daemon

Since the *lmgrd* license daemon needs to be running continually on the host server system, the easiest way to arrange this is to add the *lmgrd* start-up command line to your system's boot file. The simplest command to start the *lmgrd* license daemon is:

```
lmgrd &
```

You may need or want to use the fully-qualified pathname to the *lmgrd* binary file; for example:

```
/u2/avs/license/lmgrd &
```

Option #1 of the *licdiag* utility can also be used to start or restart the *lmgrd* daemon. See the "licdiag" section below, for information about this utility.

The *lmgrd* command line has a number of options available (see the *lmgrd* manpage for a complete list). The following example illustrates those options which Advanced Visual Systems recommends:

```
lmgrd -c <license_file_path> -t <connect-timeout> > license.log &
```

where:

`-c <license_file_path>`

The fully-qualified name of the license file. By default, *lmgrd* looks for a file named *license.dat* either in the */usr/local/flexlm/licenses* directory or in the directory specified by the **LM_LICENSE_FILE** variable, if it is set. If you have renamed or moved the file, you can use the -c option to specify its new name or location. The -c option supersedes the location specified by **LM_LICENSE_FILE**.

`-t <connect_timeout>`

The timeout, in seconds, for "connect" calls to other daemons if you are operating redundant license daemons on multiple servers (see below, "Creating license daemon redundancy").

`license.log`

Since the *lmgrd* license daemon generates transaction messages, you will probably want to redirect the output away from stdout and into a log file. Redirecting to */dev/null* effectively eliminates the log file. A list of log messages and their meanings is available in the *$AVS_PATH/license/logmessages* file.

Adding lmgrd start-up to your boot file

You can add your *lmgrd* start-up command to your host system boot file so that the license daemon will automatically be started when the system is booted. The following lists and notes the boot files for the platforms supported by AVS. Option #7 of the *licdiag* utility provides help on starting the *lmgrd* daemon at boot time. See the section on "licdiag".

Compaq Tru64 UNIX: */sbin/init.d* startup file

HP 9000/7xx: */etc/rc* startup file. (This applies to HP-UX 9.05 only; no comparable option is currently available for HP-UX 10.10.)

IBM RS/6000: */etc/rc* startup file

SGI: The */etc/rc2.d* directory contains links to specific feature startup scripts. The scripts themselves can be anywhere, including the */etc* directory.

SunOS 5.x: */etc/init.d* startup file

Note: If you are concerned with security, you can place an "su username <" at the beginning of your *lmgrd* start-up line. Without this restriction, any ordinary user can perform the *lmgrd* start-up manually. It is usually desirable to leave this option open so that users can restart the license daemon without administrator intervention.

Creating license daemon redundancy

If the *lmgrd* license daemon is not accessible to the AVS process, the program cannot run. By running redundant license daemons on multiple host servers, you can create a situation in which work can continue if one of the *lmgrd* daemons fails. You must specifically arrange for the use of redundant license daemons when you arrange to obtain your *license.dat* file from Advanced Visual Systems. If you wish to add redundancy at a later time, you will have to contact Advanced Visual Systems to obtain a new *license.dat* file.

You can create redundancy by running an odd number of *lmgrd* daemons on different hosts. As long as a majority of the daemons are active, then if AVS can contact two of the three (or three of the five), AVS will be satisfied that it has reestablished contact with a suitable license daemon and will continue to execute. See "Redundant license servers" for a description and diagram of this kind of licensing configuration.

Multiple vendors using FLEXlm

In the case where multiple software vendors install FLEXlm-based products at a single end-user site, the potential arises for license file location conflicts. If you are running AVS on a system or network where other products are using FLEXlm, there are several ways to handle it. You can use an existing lmgrd license daemon to control AVS licenses. Or, you can make the AVS licensing mechanism independent by running a separate license daemon on a different host server.

Using the correct FLEXlm version

You must be using FLEXlm version 5.12 or later in order to use AVS 5.4 and above. If a vendor is already using FLEXlm on your system and it is an earlier version, install version 5.12 (which is provided with AVS 5.4 and 5.5) and restart the licensing daemons.

This is also true if you are already using FLEXlm to license earlier products or versions from Advanced Visual Systems. These earlier releases use earlier versions of FLEXlm that are incompatible with AVS 5.4 and 5.5. They will continue to work with FLEXlm 5.12.

You can use option #4 of the *licdiag* utility to find out what version of FLEXlm is currently running.

Using a single licensing file

You can combine the *license.dat* files of all software products using FLEXlm and have one *lmgrd* license daemon administer all licenses for all products. To do this, you must be sure to specify the machine id of the licensing server that is already in use when you obtain your license from Advanced Visual Systems.

Append the **DAEMON** and **FEATURE** lines of your AVS license.dat file to the existing *license.dat* file on your system. The combined license file can be located in the default location *(/usr/local/flexlm/licenses/license.dat)*, or you can put it in another location and set the **LM_LICENSE_FILE** variable to point to it.

Using one lmgrd daemon and multiple license files

If you want to use a single license daemon but keep the license files of the various products separate, you can use the **LM_LICENSE_FILE** variable to point to multiple license files. Since you are using a single license daemon, be sure to specify the machine id of the licensing server that is already in use when you obtain your license from Advanced Visual Systems.

For example, you may already have a product that is using a *license.dat* file that is in the default */usr/local/flexlm/licenses* directory. You might rename your AVS license.dat file to *avslic.dat* and copy it to the *$XP_PATH/license* directory. By then setting the **LM_LICENSE_FILE** variable to point to both files, the *lmgrd* daemon will be able to find the individual licensing information for each product. When setting **LM_LICENSE_FILE** to include multiple filenames, use a colon to separate the pathnames (must be all one line):

```
setenv LM_LICENSE_FILE /usr/local/flexlm/licenses/license.dat:
       /u2/avs/license/avslic.dat
```

Using multiple license daemons

You can create a separate *license.dat* file and run a different *lmgrd* license daemon to handle AVS licensing independently of other products being licensed by the FLEXlm licensing system. Each *different lmgrd* daemon must run on a separate host server. You can then either set **LM_LICENSE_FILE** to point to all of the possible license files, or start each *lmgrd* daemon using the -c option specifying the location of

the license file to be read.

Socket numbers in a multi-vendor environment

In your *license.dat* file, the TCP socket number that appears on the **SERVER** line is usually set to 1700. However, there may be other programs on your network that already use number 1700, in which case you will receive "socket already in use" errors. If that is the case, change the TCP number on the **SERVER** line to an unused TCP socket number. (You can use option #1 of *licdiag* to check the availability of the socket.)

Licensing configurations

The two different kinds of licenses that are available for AVS 5.5 - node-locked licenses and floating licenses - reflect the two basic licensing configurations. Other configurations, such as combining node-locked and floating licenses or running redundant licensing servers, are also possible.

Basic node-locked license configuration

Node-locked, or fixed, licenses reflect a configuration in which AVS is installed and executes on a single, designated host system; AVS users access the host from the main console or via rlogin from other workstations and run AVS on the host machine. In most node-locked configurations, the execution node and the licensing server is the same machine, although this is not always the case.

In the following figure, AVS is installed and executes on the machine, hostA. hostA also serves as the host server for the licensing daemons, *lmgrd* and *avs_lmd*. Up to *n* users can log in to hostA and run AVS there (*n* reflects the number of licenses purchased). *The license.dat* file resides on hostA.

The *license.dat* file for such a configuration would look something like this:

```
SERVER hostA <machineidA> 1700
DAEMON avs_lmd /usr/avs/license/avs_lmd
FEATURE AVS avs_lmd 5.000 1-jan-00 <n> FB4FB43AFD04E643099C "" <machineidA>
```

Basic floating license configuration

Floating licenses use one central host server to administer licenses for users who are running AVS locally on their own, networked workstations.

In the following figure, hostA is the host license server where the *lmgrd* license daemon runs continually. Requests for licenses come in to the *lmgrd* daemon from client systems; when a license is granted AVS runs locally on the client workstations (client host 1, client host 2, and so forth). The *license.dat* file resides on hostA. (Note that any host in the configuration could actually be used as the license server.)

For this floating license configuration, the *license.dat* file might look something like this:

```
SERVER hostA <machineidA> 1700
DAEMON avs_lmd /usr/avs/license/avs_lmd
FEATURE AVS avs_lmd 5.000 1-jan-00 <n> FB4FB43AFD04E643099C ""
```

Redundant license servers

There may be situations when you want to set up redundant licensing so that if one *lmgrd* license server fails, licensing will continue to be administered by an alternate server. In that case, you can create redundancy by running an odd number of *lmgrd* daemons on different license servers. This would be reflected by multiple **SERVER** lines in your *license.dat* file, an identical copy of which resides on each server. An *lmgrd* license daemon would run on each machine.

As long as a majority of the daemons are active, then if AVS can contact two of the three (or three of the five), AVS will be satisfied that it has reestablished contact with a suitable license daemon and will continue to execute.

You must specifically arrange for the use of redundant license daemons when you arrange to obtain your *license.dat* file from Advanced Visual Systems. If you wish to add redundancy at a later time, you will have to contact Advanced Visual Systems to obtain a new *license.dat* file.

In the following figure, three license servers are being used. The license daemons communicate and a majority constitutes a quorum that will permit AVS access.

The *license.dat* file for this configuration might look something like:

```
SERVER hostA <machineidA> 1700
SERVER hostB <machineidB> 1700
SERVER hostC <machineidC> 1700
DAEMON avs_lmd /usr/avs/license/avs_lmd
FEATURE AVS avs_lmd 5.000 1-jan-00 <n> FB4FB43AFD04E643099C ""
```

Mixed floating and node-locked licenses

It is perfectly possible to have both node-locked and floating licenses within one configuration. As long as the node-locked **FEATURE** line precedes the floating **FEATURE** line in the *license.dat* file, when the AVS process starts on a node-locked workstation, it will try to use the node-locked license first (and run AVS locally). Otherwise, it will issue a request to the *lmgrd* license daemon for a floating license.

License administration tools

Advanced Visual Systems provides a number of licensing utilities to facilitate the installation and administration of licensing. This section describes these utilities, which include:

- *licdiag* - a shell script that helps with various licensing administration and diagnostic tasks.
- *lmstat* - to monitor the status of network licensing activities
- *lmdown* - to gracefully shut down the licensing and vendor daemons
- *lmlist* - to list current users of licensed features
- *lmremove* - to remove a single user's license of a specified feature
- *lmreread* - to have the *lmgrd* license daemon re-read the license file
- *lmhostid* - to report the machine id of the system

licdiag

The *licdiag* utility is installed in the *$AVS_PATH/license* directory. It is a shell script which provides a convenient way to perform many license administration tasks and to diagnose problems. *licdiag* examines your license file and license setup, troubleshoots the most common licensing problems, helps you to start and stop the daemons, and makes various recommendations depending on the platform.

When you start *licdiag*, the script first determines your host's type. It then checks the validity of your AVS path and looks for a valid license

file, analyzing it for syntax errors and consistency. It tells you if the license file exists, if it is readable, and whether it has any valid **FEATURE** AVS lines; checks for **SERVER** lines for limited floating licenses; and identifies evaluation licenses.

licdiag then prompts you with an eleven-item menu that provide the following options:

1. Start or restart the daemons.

   - The license file must contain exactly one **DAEMON** *avs_lmd* line.
   - The license file must contain at least one **SERVER** line.
   - The machine's id must be listed in one of the **SERVER** lines.
   - No other *lmgrd* or *avs_lmd* processes should be running; they are killed before proceeding.
   - All lock files are removed.
   - Ensure that the TCP/IP port number specified in the **SERVER** line is not being used by another application.
   - After starting the daemons, scan the log file for error messages.

2. Stop the daemons.

   - Run *lmdown* to gracefully shut down *lmgrd* and *avs_lmd.*
   - Check the process id of *lmgrd* and *avs_lmd* to ensure they are shut down.

3. Get status.

   - Execute *lmstat* to determine whether any daemons are running.
   - Verify the ability to write and lock files in */tmp.*

4. Show *lmgrd* version.

   - Run *lmgrd* -version to determine which version of *lmgrd* is running.

5. Reread license file.

   - Run *lmreread.*

6. Check networking.

   - Makes recommendations on network troubleshooting.

7. Help on starting at bootup.

   - Makes recommendations depending on the platform and operating system revision level.

8. Show license file.

   - Displays the contents of the *license.dat* file being used.

9. Show machine's id.

   - Displays the machine id as it must appear in the *license.dat* file (for node-locked licenses) and the string to be used for obtaining licenses from Advanced Visual Systems Inc.

10. Show the current log file.

    - Displays the contents of the *lmgrd* logfile.

11. Quit

lmstat

The *lmstat* utility is installed in the *$AVS_PATH/license* directory. *lmstat* lets you monitor the status of all FLEXlm network licensing activities instantly, and displays the following:

- which daemons are running
- users of individual features
- users of features served by a specific **DAEMON**

The *lmstat* syntax is as follows:

```
lmstat          [-a] [-S DAEMON] [-f feature_name] [-s server_name]
                [-t value] [-c license_file]
                [-A] [-l regular_expression]

-a                      Display everything
-S DAEMON               List all users of DAEMONs features
-f feature_name         List users of feature_name
-s server_name          Display status of node named server_name
-t value                Set lmstat timeout to value
-c license_file         Use the license file named license_file
-A                      List all active licenses
-l regular_expression   List users of matching license(s)
```

lmdown

The *lmdown* utility is installed in the *$AVS_PATH/license* directory. *lmdown* allows for the graceful shutdown of all license daemons (both *lmgrd* and all vendor daemons) on all nodes. To use *lmdown,* type "lmdown" with the correct license file in either */usr/local/flexlm/licenses/license.dat,* or the license file pathname in the environment variable **LM_LICENSE_FILE**. Alternatively, *lmdown* permits the "-c license_file_path" argument to specify the license file location. The *lmdown* utility can be executed only by the root user. The local administrator should protect the execution of *lmdown,* since shutting down the servers causes the loss of licenses.

lmlist

The *lmlist* utility is installed in the *$AVS_PATH/license* directory. *lmlist* lists the users of any individual feature, in much the same way as "lmstat -f". To use *lmlist,* type "lmlist <feature_name>"; for example,

```
lmlist AVS
```

lists current users of AVS. A list of users of the selected feature is displayed.

lmremove

The *lmremove* utility is installed in the *$AVS_PATH/license* directory. *lmremove* lets you remove a single user's license for a specified feature. For example, this could be required in the case where the licensed user was running the software on a node that subsequently crashed. This situation sometimes causes the license to remain unusable. The *lmremove* utility allows the license to return to the pool of available licenses.

lmremove's syntax is as follows:

```
lmremove [-c file]   feature user host [display]
```

The *lmremove* utility removes all instances of the user on the node host (on display display, if specified) from the use of the feature. If the optional "-c file" is specified, the indicated file is used as the license file. The administrator should protect the execution of *lmremove* since removing a user's license can be disruptive.

lmreread

The *lmreread* utility is installed in the *$AVS_PATH/license* directory. *lmreread* causes the *lmgrd* license daemon to reread the license file and start any new vendor daemons that have been added to the file. In addition, all pre-existing daemons are signaled to reread the license file for changes in feature licensing information. The usage is as follows:

```
lmreread [-c license_file]
```

Note: If the "-c" option is used, the license file specified is read by *lmreread*, NOT by *lmgrd; lmgrd* rereads the file it read originally. Also, you cannot use *lmreread* to change server node names or port numbers. Vendor daemons will reread their license files but will not reread their option files.

lmhostid

The *lmhostid* utility is installed in the *$AVS_PATH/license* directory. *lmhostid* reports what FLEXlm thinks is the machine id of the system on which it is executed.

What can go wrong

The following sections describe some of the more common licensing-related problems that can occur.

Connection to the license daemon has been lost

AVS periodically contacts the *lmgrd* license daemon while it is running. If AVS loses contact with the license daemon - either because the daemon has died or because the system on which the daemon was running has crashed or become inaccessible - a yellow-bordered warning window that displays the following message pops up on your screen:

```
The connection to the license daemon has been lost.
You have 15 minutes to save your work or
restart the license daemon before your session ends.
```

If this occurs, you should save your work and then restart the license daemon.

In the terminal emulator window from which you started AVS , this message appears and periodically repeats:

```
Attempting reconnection to license server for AVS
```

If the license daemon does not restart, the warning message is repeated when 5 minutes remain. At the end of 15 minutes, the AVS session exits. You will need to restart the *lmgrd* license daemon in order to restart AVS (see "Starting the lmgrd license daemon:).

Problems starting AVS

When you start AVS, a number of messages may be written to the terminal emulator window from which you started it. These messages are fairly self-explanatory.

Note: You can use the *licdiag* utility to diagnose a variety of problems with licensing. It may provide additional information beyond what is reported in the error message. For more information, see "licdiag" section above.

cannot connect to license server
AVS cannot make a network connection to the designated license server. Check the **SERVER** line in your license file and make sure that machine can be accessed through the network, at the specified port. It is also possible that that server is no longer running or that network traffic is slow.

cannot connect to a license server
The **DAEMON** name specified in your *license.dat* **FEATURE** line does not match the vendor daemon name.

cannot find license file
The license file cannot be opened. The license mechanism first uses your **LM_LICENSE_FILE** environment variable, if present. Next it looks for */usr/local/flexlm/licenses/license.dat*. If you are using a license server (for floating licenses), you should have a copy of your server's license file on this machine or at least be able to access it.

cannot find SERVER hostname in network database
The hostname specified in the **SERVER** line is not listed in */etc/hosts* or is unknown to this system; check the spelling, and

make sure that hostnames in **SERVER** lines are not longer than 10 characters; if they are, alias them and define the alias in */etc/hosts* or */etc/resolv.conf*, as necessary. For example, hostname *foo.bar.com* should be aliased to *foo*, then *foo* should be defined in */etc/hosts* as follows:

```
IP-address foo foo.bar.com
```

It also possible that the **SERVER** name requires a full DNS name.

cannot read license file
The license file cannot be read; check the path and directory/file permissions; all directories to the license file should be world-readable, as well as the license file itself.

clock difference too large between client and server
The system clock difference between the client (where *avs* is to be used) and the server (where *lmgrd* is running) is too large; maximum allowed time difference between these two machines is 27 minutes.

encryption code in license file is inconsistent
The encryption code for the requested **FEATURE** is inconsistent, i.e. it does not match the rest of the data in the license file, such as the **SERVER** line information, number of licenses, expiration date, etc. Please make sure you are using the license file as given to you by AVS Inc.

feature checkin failure detected at license server
This is somewhat serious and should be brought to the attention of the administrator. When it occurs, it is usually because the host on which the application was running has crashed. A license token may be temporarily "lost" from the pool.

feature has expired
Today's date is after the expiration date in the license file. This may occur for demonstration, evaluation, or beta copies of software.

feature not yet available
The requested **FEATURE** is not enabled yet; the date on the machine is set prior to the feature's start date.

invalid host
The hostid/hostname shown in the **SERVER** line does not match this machine's hostid/hostname; please check the spelling of the hostname, the hostid and make sure that this hostname is listed in */etc/hosts*; also check your domain: the hostname may have to be typed with its full DNS name.

license server does not support this feature

lmgrd does not support the requested **FEATURE;** either the feature has expired, or is no longer supported, or is not supported yet. Make sure that *lmgrd* version 5.12 or higher is currently being used; the *lmgrd* provided with the AVS software is version 5.12.

licensed number of users already reached

There were *n* licenses purchased for this AVS feature, and yours is the *n+1* request. This may also appear as "request for more licenses than this feature supports".

no such feature exists

The requested feature (e.g. AVS, AVS-Animator) was not found in the license file. You are probably using the wrong license file.

user/host not on INCLUDE list for feature

The local administrator has established local access lists for the feature. Contact your administrator.

DEBUGGING IN AVS 5.5

CHAPTER ELEVEN

Introduction

This chapter is a summary of suggestions and tips on developing and debugging AVS modules; it draws together a number of techniques and features that are available in different areas of the product. More detailed documentation is referenced when available elsewhere.

AVS 5.5 NOTE: This entire chapter is new for AVS 5.5. It covers features documented in prior releases as well as new features or documentation in such areas as internal debugging flags, include files, and test data generator modules.

The information is organized into the following sections:

- Coding and Porting - general advice on approaching portability
- Building for Debugging - suggestions on making modules easier to debug
- Examples - borrowing code and techniques from module source code examples
- Command Line Interpreter - setting up reproducible tests and using debug flags
- Debugging Modules - using avs_dbx
- Input and Output Data - using standard test data sets during shakeout
- Kernel Debugging - seeing what is going on in the overall system
- Working with AVS Customer Support - preparing questions, options

Coding and Porting

Module Generator

If you are writing a new module, use the Module Generator module to create and manage the module throughout its life span. Described in Chapter 2 of the Application Guide, the Module Generator will not only create a source code "template" for your new module but it will read your module back in later and allow you to make changes as it evolves. It provides support for the following operations:

- Creating source code - you enter the desired name, properties, inputs, parameters, outputs, etc. and it generates the matching source code for either a subroutine or coroutine, creating the source file with appropriate header file includes, description and compute functions with proper declarations in C or FORTRAN, and initialization calls
- Creating a makefile - a working Makefile using the recommended Makeinclude definitions file is generated, supporting general portability
- Creating man page template - creates a sample man page to fill in
- Compiling the module - compiles the module for debugging or not as desired
- Loading the module - loads the module into the current Network Editor, current module library
- Debugging the module - runs the module in *avs_dbx*, the module debugger
- Reading existing module source - reads the module back in, interpreting the header information (inputs, outputs, parameters) and remembering user code between special markers to include in the new output.

Common issues

There are a few areas that frequently cause problems that you should be aware of.

Porting between 32 and 64 bit platforms

On 32 bit platforms, the integer (int) and pointer (int*, void*, etc) data types are both 32 bits and can be (and often are) treated as the same; poor programming practice has tolerated code in which these types are

cast back and forth to each other, or in which pointer values are accepted and stored as ints, etc.

Unfortunately, the SGI N64 and Dec Alpha platforms are 64 bit platforms that store ints as 32 bits, and pointers as 64 bits. This can result in bizarre values arising inside your 32-bit code when it is run on a 64-bit platform - pointer values can be sliced in half, sometimes resulting in extreme values or 0.

The bottom line is to carefully use ints and pointers correctly and look for problems like this if you are having porting problems.

FORTRAN and 64 bit platforms

When AVS 5 was originally developed, FORTRAN had no pointer variable capabilities. In order to pass pointers to fields and other data structures, FORTRAN functions would receive pointer values as INTEGERs, then pass these into C functions to do operations such as reallocating memory or computing offsets. In order to have FORTRAN code that runs on both 32 and 64 bit platforms, you need to change the declarations for values that receive pointer arguments from INTEGER to INTEGER*8. See "FORTRAN Modules on 64 bit Systems" in Chapter 3, "AVS 5.5 on UNIX Platforms", for more information on how best to do this.

XDR data format

Binary data file formats vary from one platform to the next and can cause serious problems between 32 and 64 bit platforms and also between big- and little-endian systems. Use "XDR" data format whenever possible to minimize these issues; see documentation on the Read Field and Write Field modules for more information.

Incorrect input and output declarations

It is easy to misdeclare the compute function arguments or not handle data allocation correctly, causing unexpected behavior downstream when the data isn't what is expected.

- Use the Module Generator and the "include hints" operation as much as possible; this will help reduce initial coding errors.

- Review the available AVS 5 example modules in *$AVS_PATH/examples* to find an example of data being passed or allocated similarly to what you need. The new *corout_f* FORTRAN coroutine example shows most data types being passed in and copied to outputs. The *widgets* examples are also good

samples of how to pass parameter values.

- FORTRAN string output requires that you allocate a string that will persist after you exit the compute routine, something that is difficult to do in FORTRAN. See the newly documented *AVSoutput_string* function in Chapter 3, "AVS 5.5 on UNIX" under the "New documentation" section for help in doing this.

- Use output diagnostic modules downstream to check that you are getting what you expect (see section below).

- Visit the IAC module repository (*http://www.iavsc.org*) for related module source.

- Request AVS 5 module source code for modules handling similar operations. For more information see "Working with AVS Support" below.

Performance - Reducing data copying and processes

AVS 5 uses a variety of techniques to minimize the number of data copies it needs, including shared memory and direct module communications. These work reasonably well but there are still times where AVS 5 will make extra copies of data, for example to send the data from module A (original copy), through the kernel (2nd copy), to module B downstream (3rd copy). This particularly affects user data types which do not use shared memory.

The best solution, when possible, is to combine modules into a single "mongo" module and clean them up to be "reentrant" and "cooperative" (see next issue about Global Data). When the modules are in the same process, they can pass data just by passing pointers rather than having to serialize the data and transmit it between processes, which takes time and makes extra copies. Cutting down on the number of module processes also saves memory usage and disk space, and reduces the number of processes at run time. See "Multiple Modules in a Single Process", page 4-21 in the *AVS Developer's Guide*, for more information on making modules behave well in shared processes.

You can combine your modules with AVS 5 standard modules as needed as well, because the module binaries are included in a series of libraries. See the *$AVS_PATH/examples/multi_hog.c* for an example of how to do this.

Global Data and AVSstatic

In order to make your module code "cooperative" and "reentrant" (allowing multiple module instances to share the same process), you need to be especially careful about using global variables (variables

declared outside a particular function). These are often used for remembering state information between compute function calls, but when more than one module instance is involved this "state" information may not be saved as originally expected.

The solution is to use the **AVSstatic** variable, documented in the section on Mutiple Modules (page 4-21 *AVS Developer's Guide*), mentioned above. You declare a local "state" data structure that can hold all the "global" state info for your module, make an instance of this data structure and store the pointer to it in AVSstatic. When your module is finished the value is stored away and then reset back when your module runs the next time. The AVSstatic variable is now declared in *$AVS_PATH/include/avs.h*; in releases prior to AVS 5.5 you will need to declare this variable locally as follows:

```
extern char *AVSstatic;
```

Port.h

An undocumented header file, *$AVS_PATH/include/port.h*, contains a number of platform specific compiler flags and declarations that permit AVS supported modules to work across machines. It provides flags to define characteristics such as endian-ness, BSD or not, data sizes of LONGS, etc. If you run into platform specific behavior, you may want to first review this file to see if there are appropriate declarations covering the areas of concern. Few changes have been made in this file over the last few releases.

Building for Debugging

When you write your module, it can be helpful to build in debugging aides that will produce diagnostic output when you want it. There are a few ways to do this.

AVSmessage

The AVSmessage function and its convenience forms (AVSinfo, AVSwarning, AVSerror, and AVSfatal) provide a means of sending multiple levels of information to dialog boxes, stdout, and a log file. Using AVSinfo may allow you to include debug output from your module. For more information on AVSmessage see "Handling Errors in Modules", page 3-13 in the *AVS Developer's Guide.*

The AVSmessage function has four levels of severity - (1) information, (2) warning, (3) error, and (4) fatal - corresponding to the four shortcut function calls. By default there is a "dialog cutoff level" of 2 - levels 2 through 4 will result in a dialog box being displayed with the message,

requiring the user to acknowledge before the module continues. Below this cutoff level (2), lower level messages (in this case just "(1)information") are displayed to standard output (stdout). All messages are recorded to a log file, called */tmp/avs.log_<ProcessId>* for the duration of the session. You can tell AVS to preserve this log file after the session is completed by using the *SaveMessageLog* option in your *.avsrc* file.

This "cutoff" level can be adjusted using the CLI "debug" command on the "AVSmessage_dialog" flag. Start up using "avs -cli" and then type "debug" to see or change the current debug flag settings. The level can be raised or lowered to suppress or enable dialog output above a certain level.

See the example *$AVS_PATH/examples/widgets* for an example of how AVSmessage can be used.

Invisible parameters

You can include extra debug control parameters in your module declaration that are not shown to users by default. Use the *AVSconnect_widget* function to select "none" (or clicking the option in the Module Generator) and no widget will be attached by default. Later when you wish to use this invisible parameter you can either use the CLI "parm_set" command for your module or bring up the Module Editor and from that the Parameter Editor to attach a widget to the parameter, making it accessible. One thing your parameter might control is additional debug output to stdout or AVSinformation().

Text browsers

The **print field** module shows an example of a module that writes text output to a file and then provides a window onto that file using a "text browser" widget. The parameter provides the name of a file that the module then writes to; when the module signals for a widget update, the widget reads in the text file to refresh its output. Though narrow by default, the module can select to make this window wider and parent it to the main shell window if desired.

The end user can do the same thing using the Layout Editor mode in the Network Editor, reparenting the module panel to the shell window, stretching the text browser, and reconfiguring the module panel. For an example of how to use text browser widgets this way see the new example module, *$AVS_PATH/examples/geom_write.c*

Memory leaks

A number of good commercial memory management packages, such as Purify from Rational Software, are available to help you track down memory leaks and mismanagement. AVS 5 also contains a less powerful memory management diagnostic package, invoked by including *$AVS_PATH/include/mem_defs.h* and defining MEM_DEFS_ENABLE during compilation.

This package redefines the front end to many string management and memory management functions and can provide verbose output. These macros insert a layer of error checking routines between the application and the system memory allocation routines. For instance if you call 'malloc', this substitutes a macro for the function call so it calls 'MEMmalloc' instead. It collects statistics, does error checking, and optionally prints out debugging information. These substitutions can cause some strange syntax errors under certain conditions. See "Memory Allocation Debugging", page 4-1 in the *AVS Developer's Guide* for more details.

AVS Examples

When you need module source code examples to see how more complex operations are done there are a number of sources.

- *$AVS_PATH/examples* - the first place to start for examples of C, FORTRAN, and C++ example modules. Several new module examples have been added in AVS 5.5; see Chapter 3, "AVS 5.5 on UNIX Platforms" under the "New Example modules" section.

- *AVS_DEMOS* - the extended demo package includes a number of customer module contributions which are compiled from source when the package is installed. See the chapter on "Extended Features" under the "Installing the Demos" and "Running the Demos" sections for more information.

- International AVS Center: The IAC has a large number of customer and AVS contributed source code modules. Visit their web site at www.iavsc.org. An older snapshot of the IAC web site is available on the "Cool CD" originally shipped with AVS 5.3 and all new shipments.

- AVS supported module source code: You can request the source code for most supported modules from AVS Customer Support after signing a source code agreement. Nearly all non-builtin modules are available. For more information see "Working with AVS Support" below.

Command Line Interpreter

The AVS Command Line Interpreter (CLI) can be useful for testing in the following ways:

- providing access to a number of debug switches for producing more verbose output
- timing how long it takes a particular command to execute
- recording test and demo scripts to test networks or reproduce problems more easily
- playing back test scripts providing either stepping or time delayed execution or early termination.

In order to use the CLI directly, just start "avs -cli" and the "avs>" prompt indicates AVS is listening for CLI commands. For more general information see Chapter 5 in the *AVS Developer's Guide.*

Debug switches

The "debug" command will either list or change a series of debug switches. The values shown are the default settings. A number of these are unsupported and incomplete but work well enough to be very useful.

```
debug  [UNSUP] Set or view internal debug switches
Without arguments will list and describe the known switches.
With just the name of the switch it will show the current value.
All switches take an integer value, with off being 0 in most cases.
     Usage: debug {<switch> <value>}
```

- AVScommand_debug 0 # Echo commands received through AVScommand and their results
- AVScomm_debug 0 # Enable module communications debug output (0 or non-zero)
- AVSfield_debug -1 # Enable field debug output levels -1, 0-4 increasing output
- AVSmessage_dialog 2 # AVSmessage level that presents dialog (0-4 = info-fatal)
- CLIrecord_LUI 0 # Enable recording of LUI button hits during script output (INCOMPLETE AND SUBJECT TO SEVERE LIMITATIONS)

- CLIscript_dialog 1 # AVSmessage (during scripts) present dialog (1)
- CLItime 0 # Display the time each CLI command takes to execute (1)
- EDITORmacro_mode 1 # System mode for implementing macros. 0 means old macros, 1 is new
- FlowConcurrent 0 # Turn on or off concurrent starting of networks with parallel forks
- FlowVerbose 0 # Make internal kernel operations report status information
- MEM_verbose 0 # Print a line for every allocate & free. 1 = within module compute function only; 2 = outside; 3 = Everywhere.
- MEM_check 0 # Fill memory on allocate or free. Also report leaks on second and subsequent module compute function calls.
- MEM_history 0 # If non-zero: Fences and history of allocations. More checking and better messages on free. 2 (bit 1) = check fence posts on every allocate and free. 4 (bit 2) = when exiting, list items allocated and never freed.
- XSynchronize 0 # Control X calls being flushed immediately (1) or not (0)

Some of the more interesting flags are AVScommand_debug, which shows what commands are coming in from modules using AVScommand; CLItime, which activates timing and reporting on individual CLI command executions; and FlowVerbose for gaining insight into what module is currently executing. The "MEM" flags will affect new module instances after they are set if the MEM package is compiled into those modules.

Writing scripts

CLI scripts are helpful for automated testing as well as reproducing specific problems. In order to write a script use the "script" command:

```
% avs -cli
avs> script -open <filename>.scr -echo yes
avs_script> ....  <do what you want>
avs_script> script -close
avs>
```

By saying "echo -yes" you will see the CLI commands being recorded going to *stdout* as well. This helps the user learn basic CLI commands and confirms what is and isn't being recorded. Most interactive operations will be recorded - Network Editor operations and module widget interactions in particular. Some additional operations will be recorded when you enable the debug flag, CLIrecord_LUI, by setting it to 1. Then when you record a script, more button hits and geometry viewer operations will be recorded. **Note:** The CLIrecord_LUI option is not supported as it is incomplete in some areas - direct interaction with the Geometry Viewer views is not recorded for example. However it can sometimes provide the "missing glue" that script writers need to perform a specific operation.

For more information on the CLI, see Chapter 5 of the *AVS Developer's Guide.*

Playing back scripts

Scripts can be played back in several ways - using the "avs -cli" command line option, the Network Editor/Help/Demos script browser, the AVS Demos pull down menus, or the "script -play" command as follows:

```
% avs -cli
avs> script -play <filename>.scr -echo yes -action user -break step
```

This last is the most useful and allows you to break at every command (*step*) or only at special points (*check*), either waiting for a user prompt or pausing for a specified amount of time. You can break at any time by hitting return in the CLI and it will ask if you wish to continue or abort the script.

In combination with other debug switches, you can slow down a prerecorded script and analyse the sequence of events taking place for each operation by using the "script -sleep" option. For more information type "help script" to the CLI to get the online help for this command.

Debugging Modules

The best way to debug modules is using the *avs_dbx* script that effectively wraps around a module and mediates with the debugger of your choice.

This tool is just a shell script that can be printed out or changed if necessary; by default it is set to use the most common debugger on a given platform. It can be run either with command line arguments to select a module or it will interactively query for the minimum information it requires. The steps for running it are as follows:

- Compile your module for debugging using either the Module Generator or by setting the "G" environment variable to the needed local debug flag and calling "make".
- Use the Module Generator "debug" option or invoke *avs_dbx <binary-name>*, selecting a specific module if needed with "-mod". If desired you can select a different debugger using the "-debug" option. Set any break points you need immediately.
- Instance the module in the Network Editor
- Type "run" AFTER the message "<module> instance waiting, fire when ready...".

More information on *avs_dbx* can be found in page 3-20, *AVS Developer's Guide.*

Input and Output Data

During development it can help to use test data patterns in addition to your own data sets. Test patterns can help confirm basic operations are working over a wide range of expected data. They can also help provide sample data sets showing problems with supported modules when working with AVS Customer Support.

Test input

Sample data sets can be either static - as found in *$AVS_PATH/data*, the AVS_DEMOS package, or the IAC - or dynamic, as output by test data generators like the new **test field** module (*$AVS_PATH/examples/test_field.c*) or the older FORTRAN examples, *test_field_f.f* or *test_fld2_f.f*. The new test field module provides a much more extensive set of test data patterns, dimensions, and sizes. It is described in the UNIX chapter and documented in its own man page.

Binary data file formats vary from one platform to the next and can cause serious problems between 32 and 64 bit platforms and also between big- and little-endian systems. Use "XDR" data format whenever possible to minimize these issues; see documentation on the Read Field and Write Field modules for more information.

Diagnostic output modules

Once you have run data through your module without crashing, the question is "how do you know it produced the RIGHT answer ?" There are several ways to see what came out. Note: for more information on these and any other modules, see the online man pages.

The most basic way is to see how the modules downstream respond to the output. If they crash or produce unexpected results, something is likely wrong with the data your module is producing. If you can debug the modules downstream you may see which aspect of your data sets are causing problems.

- **Print Field** - Some supported modules are provided that help you actually look at your data more directly. **Print field** in particular is a very useful tool for examining AVS fields. If attached to a field output, it will display information about the header and some or all of the data in a text browser widget (note: Use the Network Editor/Layout Editor to expand the default user interface layout for print field or use a separate text editor onto the same file name that print field is writing to. Print field allows you to take various text slices through the data by selecting different index ranges.

  What you are usually looking for using Print Field and similar modules is unusual data values, such as "NaN"s (not-a-number values) and extreme values with high exponents that are unexpected. Of course, any serious deviation from your expectations (all zeros, clipped values, etc) can give you a powerful clue to the problem.

- **Print UCD** - a similar module which outputs UCD data instead of fields.

- **Write Geom** - this new example module outputs binary data, calls *geom_to_text* to convert the data, and displays the result in a text window like print field. The source for Write Geom shows how to use the text browser widget to view a file.

- **Geometry Viewer** - The builtin Geometry Viewer module and the Geometry Viewer itself recognize a special environment variable, **AVS_GEOM_VERIFY**, as a request to print a verbose description of what they are seeing as they process a geom data type. This output will display NaN's and other oddities that may give some clue to the problem.

- **AVSGraph** - The AVSGraph module recognizes an environment variable, called **AGX_DEBUG**, to create a log file called */tmp/AVSGraph.log* of internal Toolmaster diagnostic messages. This may help you discern problems with input data or other problems.

- **Compare Field** - The compare field module serves a useful purpose when the data can be directly compared to an expected result or a similar baseline file. For example, in order to certify that similar C and FORTRAN modules produce the same output field, you would use compare field.

Kernel debugging

Debug output

Another source of problems relates more to how the system is transmitting the data or running the module. AVS 5 is capable of producing many different types of diagnostic output, though some of this information may be more useful for AVS Customer Support to help diagnose your problems than for you to make sense out of it without understanding the internal architecture.

The Network Editor is the best way to see the network and which module is currently executing. It provides Verbose mode under the "Module Tools" menu, which will cause diagnostic output about which module is executing and what connections are being made.

A related AVS command line option, "-mod_time", will time how long a given module takes to run when it is being used. Another way to see how long operations are taking is to use the CLI debug flag "CLItimer", which will time individual CLI commands and send the output to *stdout*.

A number of environment variables exist that produce some tracing information to describe communications or module execution. These are set using "setenv" in the C-shell.

- AVS_COMM_DEBUG - provides trace information about communications traffic between the AVS kernel and all other modules, including coroutines. This can highlight data transmission problems. Can also be set using the CLI debug command to set the AVScomm_debug flag.

- AVS_GEOM_VERIFY - mentioned already, as a flag to enable diagnostic output from the Geometry Viewer as it processes field data input.

- AVS_FIELD_DEBUG - provides AVS field data information as fields are allocated and deallocated in local or shared memory. This can provide some idea of the size of fields moving around as they are being created. Can also be set using the CLI debug command to set the AVSfield_debug flag.

- AGX_DEBUG - mentioned already, provides a diagnostic output log from the AVSGraph module.

Runtime conditions

AVS 5 provides a number of means of handling data such as shared memory (shm) and direct module communications (dmc) that expedite communications and reduce data copies, but that may obscure your problem. These communications options can be disabled using the "-noshm" and "-nodmc" options respectively.

The **AVS_OGL_INFO** environment variable is useful for determining information about your graphics adapter and OpenGL implementations. Modules may seem to produce different output when in fact the Geometry Viewer is rendering their output differently on different machines. Set this environment variable to 1 before starting AVS then enter the Geometry Viewer to get diagnostic output about OpenGL initialization.

Another option to consider when isolating problems is disabling modules selectively to pin point which module is the more likely "culprit". Use the Network Editor/Module Tools/Disable Module option or the CLI command, "module -off/-on" to disable individual modules.

Working with AVS support

Finally if you continue to have troubles with your module or application, and are currently on maintenance for AVS 5, you should contact AVS Customer Support for additional help or information.

The biggest problem we often face is being able to reproduce your problem. It helps immensely if you can demonstrate the problem using some of our supported modules and data sets. If not, the ideal is to provide us a copy of your module or application to review the situation you are facing.

Our FTP site (ftp.avs.com) provides the best means of sending large data sets or executables to us for help in investigating your problem and is much more reliable than email for large files. Please contact AVS Customer Support BEFORE you place files on our ftp server to ensure we are expecting your files and can pull them in quickly to attach to your case records. Follow the instructions to place them on the ftp site, in the *incoming* directory.

Less effective but still useful is seeing a Postscript snapshot of your network, obtained using the Network Editor/Network Tools/Print Network option or seeing output views from the system using *xv* or the *Write Image* module.

Requesting Module Source Code

In some situations, you may feel it would be most effective to obtain the source code to some of our supported modules, in order to make minor changes or investigate issues. In most cases it is possible for AVS Support to release the source code to individual AVS5 modules. This will be considered only for customers under a valid support contract.

Before the source can be sent we must receive or have on file a signed copy of our Confidential Disclosure Agreement. In addition a signed Source Code Addendum must be sent for each module or set of modules requested. Both of these documents can be downloaded from our web site at *http://help.avs.com/AVS5/faq/modsrc.asp*. The requests should be faxed to the AVS main office in the U.S. at (781) 890-8287 or through your local distributor or support office.

Once the request has been received, it will be reviewed for approval. Certain modules will not be made available. The following lists some of the modules not available as source code, though there may be more. Most of the following are "builtin" modules that only work as part of the kernel process anyway; these might be made available to Developer AVS customers that require them.

- geometry viewer
- image viewer
- graph viewer
- display image
- isosurface
- colormap manager
- display pixmap
- render geometry
- transform pixmap
- image manager
- render manager
- ucd legend