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.

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.