

# 1. Introduction

Welcome to OpenGL Volumizer

OpenGL Volumizer is a volume rendering API which provides application writers with a simple interface to graphics features, such as 3D texture mapping and hardware-supported transfer functions, supported on OpenGL-based systems, along with other components essential in designing volume visualization applications. The 2.9 release provides Scalability enhancements for Silicon Graphics Prism platforms and adds support for OpenGL Shading Language. See below for a complete list of new features.

These are the release notes for the product *OpenGL Volumizer* release 2.9. This document contains the following chapters:

- [Introduction](#)
  - [Features](#)
  - [Online Release Notes](#)
  - [Platforms Supported](#)
  - [Documentation](#)
  - [Previous releases](#)
- [Installation Notes](#)
  - [Installation notes for Linux](#)
  - [Installation notes for Windows](#)
  - [Compiling sample applications and libraries](#)
  - [Sample data sets](#)
- [Known Problems and Workarounds](#)
  - [Issues with specific features](#)
  - [Platform-specific issues](#)

## 1.1 Features

Volumizer is a high-level, descriptive API for volume rendering. The 2.9 release provides support for following new features:

- Scalability enhancements for Silicon Graphics Prism platforms
- Support for OpenGL Shading Language(GLSL)
- Improved build environment for sample code on Windows platforms.
- Integration of TFEEditor with the XMLViewer MFC application on windows.
- Bug fixes and performance enhancements

Note: OpenGL Volumizer 2.9 requires a run-time license to run on all supported platforms. A product evaluation license can be downloaded from the OpenGL Volumizer product website at <http://www.sgi.com/software/volumizer>.

## 1.2 Online Release Notes

## 1.3 Platforms Supported

**32-bit Linux systems:** OpenGL Volumizer 2.9 can be used on 32-bit Linux systems running -

- RedHat 8.0, RedHat 9.0 and RedHat Enterprise Linux Workstation and Advanced Server editions.
- SUSE LINUX 9.1 and above

The Volumizer libraries are compiled using GNU compilers (GCC) version 3.2 which are not binary compatible with GCC versions 2.x and below.

**64-bit Linux systems:** OpenGL Volumizer 2.9 can be used on Silicon Graphics Prism systems. The system should be running SGI Advanced Server Environment 3 (or later) with SGI ProPack 3.2 (or later). The Volumizer libraries and sample utilities are compiled using GNU compilers (GCC) version 3.2. Volumizer is also known to run on other 64-bit Linux systems with the appropriate graphics subsystems.

**Microsoft Windows systems:** OpenGL Volumizer 2.9 can be used on Windows NT-based operating systems including Windows 2000, XP and NT. The Volumizer libraries and executables are compiled using Microsoft Visual Studio version 6.0.

The Texture Mapping Render Action requires support for 3D texture mapping. On Linux systems, you can verify this by running the following command on your system to check for the required extension -

```
% glxinfo | grep EXT_texture3D
```

On Microsoft Windows, you can use wglinfo utility to verify 3D texture mapping support.

The following is the list of supported graphics cards which provide the above functionality:

- ATI Radeon 9000 series and above
- ATI Fire GL series

- NVidia GeForce FX and above

Other machine-dependent restrictions are :

- The ARB\_fragment\_program OpenGL extension is required for the following functionality to work on the system:
  - TMFragmentProgram shader class
  - Specular shading in TMGradientShader
  - GPU-computed gradients in TMGradientShader
  - Extremely large lookup tables in TMLUTShader
- The ARB\_fragment\_shader OpenGL extensions are required for the following functionality to work on the system:
  - vzTMFragmentShader class interface for OpenGL shading Language
- Floating internal textures require the presence of the OpenGL ATI\_texture\_float extension on the system.
- On Octane systems with High Impact and Maximum Impact graphics, using a vzParameterLookupTable with external format of VZ\_RGBA does not work correctly. The alpha component of the table does not have any effect.
- On Indigo2 systems with High Impact and Maximum Impact graphics, using a vzParameterVolumeTexture with the internal texture format of VZ\_INTENSITY does not work.

Following is the complete list of systems on which the software has been tested extensively -

- Silicon Grapics Prism System for Linux with Fire GL X1(R300)/X2(R350)/X3(R420) graphics chipset.
- RedHat Enterprise Linux WS 3.0 with GeForce FX 5600 graphics
- RedHat Enterprise Linux WS 3.0 with ATI Radeon 9500 graphics
- SUSE Linux 9.1 and 9.2 with GeForce FX 5600, Radeon 9700 graphics
- Fedora core3(runtime environment only) with GeForce FX 6600.
- RedHat 8.0 with Radeon 9700, GeForce FX Go5600 graphics
- Windows XP with GeForce FX 5600, Radeon 9800 XT, Radeon 9700 graphics
- Windows 2000 with GeForce FX 5200 graphics

For a detailed list of known platform-specific issues, see [section 3.2](#).

## 1.4 Documentation

The Volumizer Programming guide and Reference manuals can be found from the product web page at <http://www.sgi.com/software/volumizer/> or at SGI technical publications web site at <http://docs.sgi.com>.

### OpenGL Volumizer Programming Guide

**Linux systems:**

A PDF version of the programming guide can be found under </usr/share/Volumizer2/doc/developer/>

**Windows systems:**

A PDF version of the programming guide can be found under <C:\Program Files\Silicon Graphics\OpenGL Volumizer\doc\developer> (Default installation path )

**OpenGL Volumizer User and Reference Manuals****Linux systems:**

User manual information is included in man (1) pages only.

Man pages - /usr/man/cat1

Reference manual information is included in two forms: HTML pages and online man pages.

HTML pages - </usr/doc/Volumizer2-2.9/>

Man pages - /usr/man/cat3

**Windows systems:**

User manual information is included in HTML pages only.

HTML pages - <C:\Program Files\Silicon Graphics\OpenGL Volumizer\doc\user\manindex.html>

Reference manual information is included in HTML pages only.

HTML pages - <C:\Program Files\Silicon Graphics\OpenGL Volumizer\doc\developer\Volumizer2Index.html>

The documentation for the sample distribution can be found in </usr/share/Volumizer2/index.html> on Linux and in <C:\Program Files\Silicon Graphics\OpenGL Volumizer\index.html> on Windows.

To find out more about OpenGL Volumizer 2.9 visit our web page at

<http://www.sgi.com/software/volumizer/>

## 1.5 Previous Releases

In addition to the features listed above, the following list provides the features included in the previous releases of OpenGL Volumizer.

### OpenGL Volumizer 2.8

- Support for 32-bit SUSE LINUX systems.
- Extended support for Silicon Graphics Visualization System for Linux:
  - Support for GCC 3.x binaries

- Sources for Multipipe SDK sample applications
  - Sample Multipipe SDK configuration files for multi-pipe rendering
- High quality shading algorithms
  - Support for specular shading in vzTMGradientShader
  - Support for GPU-computed gradients in vzTMGradientShader
  - Support for very large lookup tables in vzTMLUTShader
- Synthetic data generation tool, volgen.
- Sample code enhancements:
  - New vzmpk library
    - Automatic MPK config generation
    - MPK callback setup
  - Enhancements to volview application
    - New user interaction modes
    - Correct rendering of polygonal geometry with DB-decomposition
    - Support for DB decomposition in volviewXML module
    - Enhanced time-varying volume rendering module volviewTV
- Bug fixes and performance enhancements

#### OpenGL Volumizer 2.7

- Support for Microsoft Windows-based systems (Windows XP/2000/NT)
- Extended support for Silicon Graphics Visualization System for Linux
- Dynamic 2D load-balancing in volview
- DB-decomposition as a separate plug-in module in volview
- Textures with data format of GL\_RGB
- Textures with floating point internal data types

#### OpenGL Volumizer 2.6

- Support for Linux-based systems (see section 1.3 for details)
- Extended support for Silicon Graphics UltimateVision Onyx4 platforms (see section 1.3 for details)
- Sample code cleanup and documentation

#### OpenGL Volumizer 2.5

- Support for SGI Onyx4 systems
- Direct rendering of Irregular Grids
- Programmable shading interface using OpenGL Shader's ISL
- Enhanced DICOM support
- volview enhancements / feature additions
- Support for multiple lookup tables in the same appearance
- Support for multiple overlapping clip-textures in the same appearance

#### OpenGL Volumizer 2.4

- New XML-based Volumizer file interface (libvzxml)
- An API to represent a collection of volumetric shapes (vzShapeSet)
- Higher level interface for rendering shape sets (vzClipRenderAction)
- New multi-threading scheme for optimal disk paging of clip-textures
- ClipGen3d Optimizations - Multi-threading and Efficient file formats
- Sample code for rendering and manipulating shape sets (XMLViewer)
- Use of plug-in modules with volview using a command line interface.
- Bug fixes

### OpenGL Volumizer 2.3

- Custom multi-pass volumetric shading interface (vzTMShader).
- Support for management of multiple clip-textures using the same clip-render action.
- Optimizations for ClipGen3d and support for new filtering/data processing modes.
- Sample code for integration with Open Inventor (SoVolumeShape)
- Sample code for rendering multiple clip-textures (ClipTex3d)
- Support for OpenGL Multipipe SDK 2.0 in volview
- New packaging for shipped libraries and sample code.
- Bug fixes

### OpenGL Volumizer 2.2

- New Large Data API for 3D Clip Textures. This includes a new parameter class, vzParameterClipTexture and a new render action, vzClipRenderAction.
- New geometry class, vzPolyGeometry, derived from vzGeometry, to render arbitrary polygonal geometry.
- Utility application, dicomToIFL, to convert DICOM files to IFL format.
- Utility application, ClipGen3d, to generate 3D clip textures.
- Sample code for integration with the Visualization Toolkit (VTK).
- Sample code for using 3D clip textures in src/apps/clipTex3d and src/apps/clipRenderer/
- Sample code for generating volume data files using volume writers.
- New sample volume data loaders for loading raw binary and bricked volume data files.
- Sample volume viewer application, volviewMR, installed in /usr/sbin/ to do multiresolution volume rendering.

### OpenGL Volumizer 2.1

- Optimized paging techniques for texture memory oversubscription.
- Support for multiple overlapping volume textures.
- New multi-volume shaders namely, vzTMGradientShader and vzTMTAGShader.
- Support for two and four way texture interleaving on InfiniteReality systems.

- Sample code for DB and DPLEX decomposition using OpenGL Multipipe SDK 1.2.
- Sample code for time varying volume rendering with asynchronous disk paging.
- Sample code for integration with OpenGL Performer.

### OpenGL Volumizer 2.0

- High level, descriptive, C++ API
- Works well with other scene graph APIs
- Thread-safe
- Integrated volume lighting
- Immediate-mode rendering without transient geometry overhead
- Built-in support for rendering slice planes
- Optimized for InfiniteReality systems
- Volume application-based on Multipipe SDK
- Transfer function editor
- Volume data loaders

## 2. Installation Notes

### 2.1 Installation Notes for Linux

The information listed here is specific to RedHat Linux and Suse Linux.

#### 2.1.1 Subsystems

The following table lists the subsystems installed with the OpenGL Volumizer 2.9 package and the sizes of the rpms in KBs.

Note: The listed subsystem sizes are approximate.  
OpenGL Volumizer 2.9 includes these subsystems:

Subsystem Name	Description	Size in KB - 32-bit Linux	Size in KB - 64-bit Linux
sgi-volumizer	2.9 Execution Environment	25677	35511
sgi-volumizer-devel	2.9 Development Environment	94056	62988
sgi-volumizer-data	2.9 Sample data sets	74184	74196
Sgi-volumizer-license	2.9 Run-time license utilities	13710	N/A

## 2.1.2 Installation Method

All of the subsystems for OpenGL Volumizer can be installed under Red Hat Linux or Suse Linux. You need to be logged in as root to execute the following commands.

While not necessary, it may be useful to remove old versions of the product:

```
cd /usr/share/Volumizer2/src
make clobber
rpm -e Volumizer2_dev Volumizer2_eoe Volumizer2_data sgi-volumizer sgi-
volumizer-devel sgi-volumizer-data
```

Use the following commands to install OpenGL Volumizer 2.9 on 32-bit Linux.

```
rpm -Uvh sgi-volumizer-2.9.i386.rpm sgi-volumizer-devel-2.9.i386.rpm
sgi-volumizer-data-2.9.i386.rpm sgi-volumizer-license-2.9.i386.rpm
```

Use the following commands to install OpenGL Volumizer 2.9 on 64-bit Linux.

```
rpm -Uvh sgi-volumizer-2.9.i386.rpm sgi-volumizer-devel-2.9.i386.rpm
sgi-volumizer-data-2.9.i386.rpm
```

## 2.1.3 Installation Locations

OpenGL Volumizer is installed into the following directories:

/usr/lib/libvz*.so	DSOs
/usr/demos/Volumizer2/	Sample demos, DSOs and Sample Multipipe SDK configuration files
/usr/share/Volumizer2/src	Example source code
/usr/share/Volumizer2/bin	Conversion Utilities
/usr/share/Volumizer2/data	Sample data sets
/usr/include/Volumizer2/	Header files
/usr/man/cat1/	User Man pages
/usr/man/cat3/	Developer Man pages
/usr/doc/Volumizer2-2.9/	Developer reference pages

## 2.1.4 Prerequisites

Your system must be running a compatible version of RedHat Linux or Suse Linux in order to use this version of OpenGL Volumizer.

To run all of the demos, the following images must be installed. Note that all the libraries need to be compiled using GCC 3.x compiles, on both 32-bit and 64-bit Linux systems.

- openmotif 2.1 or later (Motif runtime software)
- libtiff 3.0 or greater (TIFF - Tagged Image File Format library)
- performer\_eoe 2.5 or greater (OpenGL Performer Library Executable Software).
- inventor\_eoe (Open Inventor Library Executable Software)

To compile all of the demos, you will also need:

- gcc, gcc-c++ and libgcc 2.3 or later (GNU C++ compilers, headers and libraries)
- openmotif-devel 2.2 or later (Motif development software)
- libtiff 3.5 or later (TIFF - Tagged Image File Format Development Environment)
- glut-devel 3.7 or above (OpenGL Utility Toolkit)
- performer\_dev 2.5 or greater (Performer development environment)
- inventor\_dev (Open Inventor Library Executable Software)
- VTK version 4.x or greater to compile the VTK sample code

**Note:** The build system for the sample code uses gmake which needs to be installed on the system as well.

### 2.1.5 Get Started

To check successful installation of Volumizer and get started with Volumizer, kindly go through the *Get Started* section in manual page of Volumizer. You can see Volumizer man page simply by typing 'man Volumizer'.

## 2.2 [Installation notes for Windows](#)

### 2.2 Installation Notes for Microsoft Windows

Volumizer 2.9 is supported on Windows NT based systems including Windows 2000, XP and NT.

#### 2.2.1 Subsystems

The following table lists the subsystems installed with the OpenGL Volumizer 2.9 package and the sizes of the installed components in KBs.

Note: The listed subsystem sizes are approximate. OpenGL Volumizer 2.9 includes these subsystems:

<b>Subsystem Name</b>	<b>Description</b>	<b>Size in KB</b>
Volumizer2_dev	2.9 Development Environment	<b>15306</b>
Volumizer2_eoe	2.9 Execution Environment	<b>25188</b>
Volumizer2_data	2.9 Sample data sets	<b>57042</b>

## 2.2.2 Installation Method

All of the subsystems for OpenGL Volumizer can be installed under Microsoft Windows XP/2000/NT.

While not necessary, it may be useful to remove old versions of the product by going to 'Start Menu -> All Programs -> SGI OpenGL Volumizer' and selecting the 'Uninstall OpenGL Volumizer' item.

To install OpenGL Volumizer 2.9 on Windows, simply launch the installer by double clicking on the icon (or launching the command from the command line). Follow the instructions to complete the installation.

## 2.2.3 Installation Locations

By default, OpenGL Volumizer is installed under C:\Program Files\Silicon Graphics\OpenGL Volumizer. This path can be changed during the installation process if you need to do so for any reason. Assuming that the software is installed under VZROOT, the location of the various components is given by the following table:

VZROOT\bin\ vz*.dll	DLLs
VZROOT\lib\ vz*.lib	Import Libraries
VZROOT\bin\demos	Demo Applications
VZROOT\bin\volgen	Volume Generation Utility
VZROOT\bin\dicom	DICOM Conversion Tool
VZROOT\src	Example source code
VZROOT\data	Sample data sets
VZROOT\include	Header files
VZROOT\doc\developer	Developer reference pages
VZROOT\doc\user	User Manual pages
VZROOT\release_notes	Release Notes

## 2.2.4 Prerequisites

Your system must be running a compatible version of Microsoft Windows in order to use this version of OpenGL Volumizer.

To run all of the demos, the following software must be installed. Note that all the installed Volumizer libraries have been compiled using Microsoft Visual Studio 6.0:

- libtiff 3.0 or greater (TIFF - Tagged Image File Format library)
- GLUT 3.0 or later (OpenGL Utility Toolkit runtime software)
- OpenGL Performer development environment 3.1 or later

To compile all of the demos, you will also need:

- Microsoft Visual Studio 6.0 or later
- GLUT 3.0 or later (OpenGL Utility Toolkit development software)
- libtiff 3.0 or later (TIFF - Tagged Image File Format Development Environment)
- OpenGL Performer development environment 3.1 or later
- VTK version 4.2 or later to compile the VTK sample code

## 2.2.5 Get Started

To build all the Volumizer sample code on windows, Volumizer by default uses the msdev Build Environment provided with this release. Run the BUILD.bat file in VZROOT/src directory or click *Program Files->OpenGL Volumizer->Build Sample Code*, which will ask you to set some paths and then launch msdev. All the sample code is arranged in corresponding project files(.dsp) under Volumizer.dsw(workspace). Set BUILDALL project as Active project and build it.

**Note:** An alternate method to build all the sample code is to run BUILD\_NMAKE.bat file in VZROOT/src directory which uses nmake to build all the sample code. For using nmake it has to be installed on the system as well. The Microsoft development environment has to be initialized by running the VCVARS32.BAT batch file provided by Microsoft Visual Studio before compiling the sample code.

The simplest way to check successful installation of Volumizer on your Windows system is to run Volumizer's already installed demo applications.

The Volumizer-based sample application **simple** is installed in VZROOT\bin\demos directory. Either run it directly from the start menu by clicking *Program Files->OpenGL Volumizer->Simple Test Application*, or alternatively run the batch file VZROOT\bin\demos\RUN\_simple. If everything is fine then you should see a Head dataset rendered.

Another MFC based application **XMLViewer** is also installed in VZROOT\bin\demos directory. As above you can either run it directly through start menu by clicking *Program*

*Files->OpenGL Volumizer->XML Viewer Application*, or alternatively run the batch file VZROOT\bin\demos\RUN\_xml. By default no data is loaded; To render the data simply open simple.vz file through 'File' menu. If everything is fine then you should again see a head data set rendered. With this MFC application, you can also use **TFEDITOR** to change the interpolation methods and format and use other built-in lookup tables.

### **2.2.5.1 Issues when running installed demo applications**

Go through the following checklist in case you are having problems running the installed demo applications mentioned above :

- Check if you have valid license file license.dat in VZROOT directory.
- Check if you have the right environment variables set to allow Volumizer to find the license file. You can try setting the environment variable VZROOT to your Volumizer installation directory. Default installation directory is *C:\Program Files\Silicon Graphics\OpenGL Volumizer*. (On XP, use *Control Panel -> Performance and Maintenance -> System -> Advanced -> Environment Variables* to add this value)
- Check if you have 3D texture mapping support with your graphics hardware. Also make sure that you have installed appropriate drivers for your graphics hardware. See section [Section 1.3](#) for supported graphics cards and machine dependent restrictions.
- Volumizer's runtime DLLs are installed in the directory VZROOT\bin, make sure that your PATH environment variable includes this directory. In case the PATH is not set properly the demo application will throw an error that it could not find vz.dll and other Volumizer DLLs. (On XP, use *Control Panel -> Performance and Maintenance -> System -> Advanced -> Environment Variables* to add this directory to your PATH variable.)
- Sometimes we have also seen some licensing issues when multiple users are logged on to the Windows system. If there are multiple users on your Windows machine, make sure you are the only one currently logged on.

If the problem still persists, kindly send an email to [info-volumizer@els.sgi.com](mailto:info-volumizer@els.sgi.com)

### **2.2.5.2 Issues when compiling installed sample code**

Go through the following checklist in case you are having problems when compiling the installed sample code in VZROOT\src (or your own code):

- Check if you have Microsoft development environment set with VCVARS32.BAT and you are able to run nmake command.
- Volumizer assumes preprocessor macro `_WINDOWS` to be defined on Windows within its source files. If it is missing from the compile flags, add it with `/D "_WINDOWS"`

## 2.3 Compiling sample applications and libraries

### 2.3.1 Compiling sample sources

To build the example code on Linux, go to `/usr/share/Volumizer2/src` and type "make" within the directory. On Windows, you can use the short-cut menu item under "OpenGL Volumizer" and select the "Build Sample Code" option to launch the build. Alternatively go to `VZROOT\src` and type `nmake` manually. This action will recursively build all of the example libraries (under `src/lib`) and applications under (`src/apps`) shipped with the product. Another way to build the sample code on windows is to run the `BUILD.bat` file in `VZROOT\src` directory. This batch file will ask you to set the environment variables and then launch `msdev` to build all the sample code. Please note the following if using your own build environment to compile the sample sources. Note:

- On Windows, you will need to make sure that you use the `/D_WINDOWS` flag on the Visual C++ compile line.

### 2.3.2 Location of compiled libraries

On Linux, the compiled libraries will be placed within the `/usr/share/Volumizer2/lib/` directory under a sub-directory corresponding to the appropriate ABI. For eg. 32-bit Linux versions of Linux libraries will be copied under `linuxia32` while 64-bit Linux versions of Linux libraries will be copied under `linuxia64`.

On Windows, the compiled import libraries will be placed under the `VZROOT\lib` directory and the corresponding DLLs will be placed under the `VZROOT\bin` directory.

### 2.3.3 Environment variables

On Linux, you can edit the `vzcommondefs` file directly or set the equivalent environment variable to customize the build.

Note the following about the current release:

- To compile using MesaGLw library instead of the default GLw library, set `MESA_GLW` to 1 (Linux only)

### 2.3.4 Sample code documentation

Please read the respective READMEs in each of the directories for a description of usage and implementational details can be found in the respective `index.html` file.

## 2.4 Sample data sets

### 2.4.1 RAW Binary data sets

This release of Volumizer comes with the bonsai data set which is in raw binary format. The sample code distribution also provides loaders (src/lib/loaders/BinLoader.\*) for loading such data sets. Additionally, the libvzxml loader has a built-in module (name "bin") for loading such data sets. More sample data sets can be downloaded from the following website(s):

- [www.volvis.org](http://www.volvis.org)

### 2.4.2 TIF data sets

This release of Volumizer comes with the head data set which is in TIF(Tagged Image File) format.

### 2.4.3 DICOM format volume data

This release of Volumizer does not come with any sample data sets in the DICOM format. Freely available DICOM data sets can be downloaded from any of the following websites:

- Courtesy of Sébastien Barré: [Medical Samples](http://www.barre.nom.fr/medical/samples/) (http://www.barre.nom.fr/medical/samples/)
- Courtesy of Washington University School of Medicine: [RSNA96 Dicom Image Repository](ftp://wuerlim.wustl.edu/pub/dicom/images/version3/RSNA96/) (ftp://wuerlim.wustl.edu/pub/dicom/images/version3/RSNA96/)

## 3. Known Problems and Workarounds

### 3.1 Issues with specific features

#### Texture Mapping Render Action

- When using the vzTMShader class for rendering multiple overlapping volumes, both the volumes should have the same data ROI and geometry ROI. Different ROIs would work only if the shape is not bricked internally.
- A shape's appearance should have atleast one parameter of the name "volume" and type vzParameterVolumeTexture.
- Multi-threaded mode for vzTMRenderAction is disabled. Hence, the "maxThreads" parameter to the vzTMRenderAction constructor is ignored right now.

- The vzTMRenderAction will crash when extremely high sampling rates are set (on the order of 50x50x50!).
- Texture interpolation is always set to GL\_LINEAR. Other filtering modes are not supported.
- The slicing direction for vzTMRenderAction is always viewport aligned; there is no support for axis-aligned slicing.

### **Clip-Texture Parameter**

- Multiple overlapping clip-textures should have the same data and brick dimensions.
- Physical memory size and brick dimensions used for vzParameterClipTexture can be modified only before the clip-texture shape has been managed/drawn for the first time.
- The setRegionDirty method is not implemented. It is a place holder for future implementations.
- The dimensions of the physical memory window cannot be controlled. They are computed automatically using the size of the window allowed by the application (using setPhysicalMemorySize()).
- Multi-threaded access to a clip-level in the heirarchy is disabled by default using appropriate locking mechanisms. This can reduce the disk paging performance but is critical for non-thread safe loaders (see man vzParameterClipTexture for details. Concurrent access to a clip-level can be enabled by setting the environment variable VOLUMIZER\_THREAD\_SAFE\_LOADER. This should be done only if the data loaders for the clip-levels are thread-safe. The variable is set by default by ClipLoader class (src/lib/loaders/ClipLoader.\*) if it detects the use of thread-safe loaders (src/lib/loaders/RoamLoader.\*, file version 3.0).

### **Clip-Texture Render Action**

- The vzClipRenderAction uses an approximate sorting algorithm to sort multiple shapes in a shape set when using the new draw(vzShapeSet \*) interface. In order to use a more accurate sorter, use the manage/unmanage/draw shape interface.
- All clip-texture shapes need to have a lookup table hence the vzTMSimpleShader will not work correctly with clip-textures. When the clip render action is used to render shapes with volume textures, the behaviour is the same as that of the TMRenderAction.
- The vzClipRenderAction is based on the TMRenderAction, so it also has the same restrictions.

## 3.2 Platform-specific issues

### 64-bit Linux systems

- There are certain incompatibilities with the GLw libraries shipped on various 64-bit Linux systems. Volumizer sample code allows choosing between MesaGLw and GLw libraries by setting/unsetting the environment variable MESA\_GLW to 1. Note that the volview executable is currently linked with the GLw library by default which is available on SGI's Visualization System for Linux. The application might not work on systems which do not have the appropriate runtime libraries.

### 32-bit Linux systems

- Volview application is unstable with multi-threaded configuration files. To get multi-window configurations to run, simply make the configurations single threaded by inserting the following lines for each window -

```
attributes
{
    hints
    {
        thread n
    }
}
```

Or add MPK\_WATTR\_HINTS\_THREAD 0 under the global attribute section of the config file.

### Microsoft Windows

- Polygonal data loader, sgiobj, is broken. While using the loader to load the knot.sgo file, the loader gives an error and returns NULL.
- Environment variable VZROOT is not set by the build process. The variable can be set manually by using the 'Control Panel->System->Advanced->Environment Variables' menu on the system.
- If using Visual C++ .NET compilers, version 7.0 or above, programs including TMSheader.h header file (or other files which include this file) will not compile. This is because these compilers do not allow function pointers with default parameters as in the vzTMBindParameterCallback callback function.
- 3D Clip-textures are not supported
- ClipGen3D application is not shipped
- TFEEditor MFC application crashes sometimes while loading or saving a lut.
- TFEEditor crashes if opened without loading a volume in XMLViewer MFC application.