


Technical White Paper, by Ian Curington 6 December 2001**AVS/EXPRESS** **AVS Advanced Visual Systems**

AVS/EXPRESS PRODUCTS EXAMPLES COMPANY CONTACT SITE MAP HOME

HIGH PERFORMANCE ANALYSISsearch *with the world's leading data visualization toolkit*

The following white paper explains the fundamental concepts behind the Multipipe Edition of AVS/Express.

AVS/Express Multipipe Edition**ABSTRACT**

As the design and implementation of visualization applications is now widespread in industry, researchers and engineers seek better ways to represent and interact with large models. Software layers to control and use multi-pipe rendering systems, coupled to multi-channel immersive displays is often out of reach for most application developers. In this paper we describe a visualization software framework that embeds support for multi-pipe rendering systems, enabling existing or new visualization applications to use immersive virtual reality environments. Users can develop applications at the visualization level, without needing to understand or interact with graphics libraries. With an implementation for multi-pipe rendering inside the framework, a single application can be moved between very different immersive environments without having to recompile. In this context this paper discusses the implementation of the AVS/Express Multi-Pipe Edition (XP/MPU).

Keywords: visualization, immersive environments, high performance graphics, multi-pipe rendering, application development framework, virtual reality.

Introduction

The combination of complex multi-pipe rendering systems and multi-channel immersive display environments necessarily leads to complex application software models. An application development framework with abstracted support for multi-pipe and multi-channel rendering configurations has been developed, along with graphics data structures and software architectural design features. The system allows rapid application design, which is portable between immersive environments, providing a broad-based and reliable development platform. This framework is applied in the development of immersive visualization environments (IVEs).

Visualization software development is a complex subject, requiring technical data models to be processed and presented through computer graphics subsystems. Application development frameworks that already include visualization toolkits dramatically reduce the complexity of this task. Yet most frameworks and toolkits are designed for single-seat graphics presentations. The continued growth of very large data models means that researchers and engineers seek better ways view and interact with the data to gain insight and understanding. Immersive environments can enable this process, giving users a more accurate insight of complex structures.

Rather than build specific applications that use immersive environments, this project chose to extend the core visualization framework for multi-pipe and multi-channel support. In this way any visualization application development effort can take advantage of immersive display technology.

Multi-Pipe Rendering Problem Domain

The software environment for the multi-pipe rendering framework project is primarily high-performance computing systems, such as the Onyx-2, with high performance graphics (Infinite Reality) pipes. The display channels are projected onto a number of large surfaces, creating an immersive visual effect for the viewer. The immersive display surfaces can be arranged in a wide variety of configurations, such as one large curved screen covered with multiple projectors (Visionarium, RealityCenter, PowerWall), distinct wall, floor and ceiling projectors creating a room space (CAVE), or large desk/table systems with one or more adjoining display surfaces (HoloBench, ImmersaDesk).

A number of large, commercial visualization applications have been developed using a visualization toolkit, such as AVS/Express, for single-channel interactive use. With much investment in application architecture, behavior, specialized visualization methods and data interfaces, a need was recognized to develop a framework that would enable these applications to use immersive environments without substantial re-writes. These commercial visualization applications are designed for very specific areas, such as Oil Reservoir Engineering, Computational Fluid Dynamics (CFD), Structural Analysis (FEA), Computational Electromagnetic Modeling (CEM), and 3D Medical Feature Reconstruction [9].

As these applications are currently derived from the AVS/Express visualization development system, which itself uses OpenGL [8], a solution was designed to extend the OpenGL graphics rendering subsystem to use multi-channel immersive display environments using the SGI Multi-Pipe Utility (MPU) system [1]. By placing the multi-pipe and multi-channel support inside the virtual renderer layer of AVS/Express, all applications, including customized or third party code, could easily be accommodated without change.

Related Systems for Multi-Channel Immersive Display

Most immersive display systems have evolved from University research projects in Virtual Reality. Only recently have systems become available in the commercial market. As a result, there is very little commercial software available to support these environments. Most applications are monolithic applications, designed for a specific set of users. The AVS/Express Multi-Pipe Edition (XP/MPU) described in this paper is designed for a more flexible approach to configuration management and multi-pipe display utilization than other existing systems. In addition, as a visualization framework, is applicable to a very wide class of visualization problems, data structures, and visualization techniques, without requiring detailed knowledge of a graphics API.

In addition to MPU, another generalized support software layer is CAVElib [3]. This library grew out of developments of the CAVE VR system. The CAVElib software provides an API for management of scene content, update, and interactive devices. The physical configuration detail, such as wall projection angles and channel assignment, is managed through an external configuration file. The CAVElib is now a commercially supported product through VRCO. Due to the origins of the software, these two configurations have strongly influenced the software design of CAVElib. The CAVElib system is oriented to device control, and exposes parallel thread execution models to the application programmer, significantly more complex than the framework approach taken in this paper.

Some general purpose visualization systems developed in research laboratories also make use of immersive VR environments for the display and interaction of data. One such system is COVISE, from the University of Stuttgart [6]. The COVISE system is aimed at 3D fluid dynamics and mechanical structure visualization, computer supported cooperative working (CSCW), with display on CAVE or multiple projector walls. The renderer subsystem in COVISE uses the SGI Performer system for display, control and configuration in the VR system, thus is not as modular and portable as the multi-pipe configuration control in the AVS/Express MPU renderer or the CAVElib systems.

Other projects such as VIVRE, had integrated visualization systems with VR technology using existing mechanisms, without adding any additional controls or flexibility in VR system management [2]. By simply coupling existing commercial visualization systems with an existing commercial VR system, visualization data can be displayed quickly, with minimum development effort. However, the control of the system is made more complex for a user, with two different interface systems. Direct interaction with visualization methods is also more difficult, since graphics context information the VR system has to be translated back to the visualization system for action. These issues are avoided if the VR system is coupled directly inside the visualization system, as in this paper.

For the needs of a specific sector of the geologic structure modeling field, the GOCAD software system has recently been modified to use the MPU system [7]. This gives similar benefits as above for a specific class of problems, such as the immersive display of wells, seismic data, and geologic layers.

The system described in this paper has been preceded by a development of the Naval Research Labs, where a general C++ library vrlib was developed to support a wide range of VR systems, using the SGI performer graphics library. This library was coupled into AVS 5.0 visualization system to create a VR enabled scientific visualization system called the AVS VR Viewer [5]. It has been successfully applied in projects covering solar physics, electrodynamics, and computational fluid dynamics.

System Design Goals

The design of the AVS/Express Multi-Pipe Edition [4] is outlined by a wide set of general system requirements, many suggested by potential commercial users of the system:

- Support Parallel Rendering on Onyx-2 Architecture
- Support all display configurations (CAVE, HoloBench, PowerWall, etc.)
- Any application domain (CFD, Telecoms, Oil & Gas, Medical, etc.)
- Leverage existing OpenGL viewer system infrastructure
- Use a Component library for deployment, with full configuration control
- Have a full visualization environment structure, not a low level graphics library

System Architecture

SGI (Silicon Graphics Inc.) manufacture a high-end range of graphics systems called Onyx-2, with reality engines. These can render complex scenes in parallel, and provide multi-channel output. The Onyx-2 systems are used to drive displays such as RealityCenters, Visionariums, CAVEs, ImmersaDesks, HoloBenches, and several other large screen or immersive graphics environments. The software to drive these systems and individual configuration issues has presented a high barrier to software developers. The SGI Multi-Pipe Utility (MPU) system significantly reduces the software complexity to support these devices, and allows AVS to produce one new renderer, creating applications with one SGI executable that can run on all of the above configurations with only an ASCII configuration file change.

The SGI MPU is part of the EMEA Developer Program, designed as a new problem-solving API. The design is to allow OpenGL applications to easily migrate to multi-pipe computing and graphics environments. The MPU system supports full portability, so the same application can run on a single display low-end desktop system, through to an Onyx2 with multiple Rendering Engines. Unlike Inventor or Performer, MPU does not impose a scene-graph structure, so dynamic update of scene contents, as often happens in visualization application, is handled in an efficient way. The MPU system provides a transparent, call-back driven programming interface, and takes care of inter-process communication, parallel execution, and display configuration issues.

The AVS/Express system is a full-featured visualization system and application development environment. In addition to over 800 visualization operators and interfaces, it has a generalized viewer and interaction

system for the display of visualization data. The viewer is split into device (graphics API) independent and dependent layers. The mapping between these layers uses a virtual renderer dynamic binding structure. In this way, multiple renderers can be supported in the same runtime environment. The AVS/Express Multi-Pipe Edition extends the virtual renderer base class, and adds callbacks to the MPU subsystem.

The following sections detail the design and implementation issues of the various components that comprise the AVS/Express Multi-Pipe Edition. A brief introduction to those components is presented here.

1. Shared Memory Store

All scene graph information as generated by the main AVS/Express application is copied into shared memory, so that the spawned MPU processes which handle the multiple pipes, can access this information and render appropriately. A number of C++ classes are defined for storing various aspects of the scene graph. Existing code interprets the renderer independent scene graph, and calls the required OpenGL library calls; as this is no longer required at the application level, these GL are replaced, and reproduced within the MPU processes. The shared memory store essentially stockpiles a representation of these GL calls, and higher-level versions as necessary for efficiency.

2. Frame Store

The iframe decomposition mode in version 2.0 of the MPU library, requires a mechanism is necessary to store a full frame of information. The shared memory store is abstracted to store a number of frames as needed by the current MPU rendering mode. Each recorded frame stores the representation of GL calls as would have been made by the original OpenGL renderer, for a single frame.

3. Scene Store

The design is taken further with the introduction of the scene store, which is used for handling multiple frames, as and when needed by MPU. The scene store will either create one frame (two frames for double-buffering) or N+1 frames, where N is the number of frames used for decomposition. Also at this level is the concept of the 'build' frame and the 'ready' frame; the latest frame that is currently being constructed, and the latest complete frame ready for rendering. Finally this layer handles caching of higher-level objects to take advantage of static objects from frame to frame, by storing only a single instance of the GL representation for multiple frames.

4. Chunk Arrays

Low level OpenGL calls are stored as a sequence of tokens representing the original call, along with application geometric data as necessary. These tokens and data primitives are stored in chunk arrays (another C++ class) which will increase in size, when the array has become full. In this way the scene storage is well managed, and does not result in a large memory overhead.

Performance & Operational Modes

The functional model for operation of the AVS/Express MPU renderer system uses a frame manager process (FMP) to decouple the visualization system from the parallel graphics system. In this way the visualization system can continue to compute and update the visualization objects such as isosurface or streamlines through vector field data, while the graphics system remains busy updating the display. This architecture also allows higher frame rate updates for motion tracking or camera position changes. As the camera update is likely to be at a faster frame rate than the visualization scene content update, the multiple-frame buffer system of the FMP allows these to operate in parallel.

The FMP also allows control in iframe decomposition, or imonster mode. In this mode rendering pipelines are kept busy continuously, and overlapped in time. With N display pipes, the frame rate to the display screen can be N time a single pipe frame rate.

Conclusion

The AVS/Express Multi-Pipe Edition, based on the SGI MPU system, is designed to operate efficiently on a wide range of VR system configurations. As an application framework, it is unique from many VR applications that do not offer full development frameworks. Any application designed using this framework can deploy solutions on both single-screen workstations or full multi-channel immersive display systems. A wide range of application types, data structures, and visualization techniques are available in the framework, and are independent of the graphics environment. The framework is designed for rapid deployment of research prototypes or full commercial project development at a higher level than graphics programming libraries. The architecture takes full advantage of parallel rendering through multi-pipe display systems, and uses multi-cpu systems to control both visualization and graphics operations. The system has already been tested in a variety of configurations. The very simple ASCII configuration file allows applications to be portable across any hardware environment. The user simply defines the number of screens, their geometric projections, and the underlying system will take care of parallel rendering and display. The framework architecture allows wide deployment of visual applications, including immersive display and interaction.

Acknowledgements

This work was supported by Advanced Visual Systems Inc., Kubota Graphics Technology Ltd., SGI Inc., and the University of Manchester (UK). We wish to thank Patrick Bouchaud of SGI Switzerland, and Yoshihito Kikkawa of KGT Ltd. for extensive support and guidance during the multi-pipe rendering development. Special thanks goes to Paul Lever, for leading the design and implementation effort, and to George Leaver and James Perrin on the development team at the University of Manchester.

References

1. Bouchaud, P. "Writing Multipipe Applications with the MPU", SGI EMEA Developer Program document, 1998. <http://www-devprg.sgi.de/devtools/tools/MPU/index.html>
2. Boyd, Gallop, Palmen, Platon, Seelig "IVRE: User-Centred Visualization" EU ESPRIT HPCN Project Report, 1999. http://www.tessella.co.uk/projects/vivre/hpcn99paper_revised.htm
3. Pape, D. "The CAVE Virtual Reality System" Description of CAVElib programming environment, 1998, <http://www.vrco.com/CAVE/>
4. Curington, Lever "AVS/Express Multi-Pipe Rendering Project" International AVS Centre web site (IAC) 1999, <http://www.iavsc.org/general/news/mpu.html>
5. Kuo, Lanzagorta, Rosenberg, Summers, Scannell "The AVS Virtual Reality Viewer", IEEE Computer Society Technical Committee on Computer Graphics, October 1999.
6. Rantau D., Frank K., Lang U., Rainer D., W-sner U., "COVISE in the CUBE: An Environment for Analyzing Large and Complex Simulation Data", 2nd Workshop on Immersive Projection Technology (IPT '98), Ames, Iowa 11.-12. May 1998
7. Winkler, Bosquet, Cavin, Paul "Design and Implementation of an Immersive Geoscience Toolkit" Proceedings of Vis99, IEEE Computer Society Technical Committee on Computer Graphics, October 1999
8. "OpenGL High Performance 2D & 3D Graphics" resource site, 1999 <http://www.opengl.org/>
9. "Gaining Insight Through Data Visualization" AVS Technology White Paper, Advanced Visual Systems Inc. 1998.