

# Design Issues in the AVS/Express Multi-Pipe Edition

Paul G. Lever, George W. Leaver, Ian Curington\*, James S. Perrin, Andrew W. Dodd, Nigel W. John, W.T. Hewitt  
International AVS Centre/Manchester Visualization Centre, \*Advanced Visual Systems, Inc.  
paul.lever@man.ac.uk

## Abstract

*The initial design and implementation issues for the development of the AVS/Express Multi-Pipe Edition (MPE) are described. MPE integrates the widely used commercial product AVS/Express with support for multi-pipe architectures including Virtual Environments (VE) using the SGI Multi-Pipe Utility library (MPU). Implementation is at the AVS/Express source code level, rather than the user level through the standard module interface, giving better performance and alleviating the problems of extending a visualization toolkit which has not been designed for multi-pipe architectures.*

## Introduction

The goal of any visualization tool is to provide insight into data and this may be enhanced by placing the investigators in a large-screen and/or virtual environment. AVS/Express already provides an extensive set of visualization tools but currently lacks any standard VE support. Existing solutions have extended the functionality via the usual method of adding user modules and with projects such as VIVRE have shown this to be a valid approach. However such solutions have the drawback of requiring changes to be made to the application, adding modules to an existing application network. This impacts on application portability, a strength of AVS/Express with its multiple platform support. Having the advantage of access to the AVS/Express source code afforded us the opportunity to integrate support more tightly, to minimize the impact on the interface, retain portability across application platforms, introduce portability between VE platforms and make support as transparent as possible.

For VE support a number of solutions are available but the MPU library from SGI provides a wide-range of output configurations including support for desktop systems, Immersadesks, CAVES and RealityCenters. MPU not only makes it possible to use large projection output but also allows multiple rendering pipes to be combined for any display thereby improving rendering performance.

## Approach

Support for MPU rendering has been added directly as a new renderer layer inside AVS/Express. Existing support for a number of platforms means that AVS/Express is already well structured. Figure 1 shows how the application and Object Manager works through an Independent Renderer Layer which transparently passes instructions to the selected renderer.

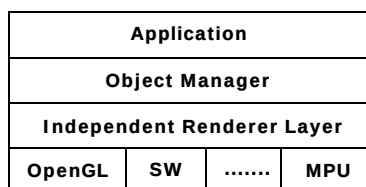


Figure 1. AVS/Express layers, showing independence from new MPU renderer.

Using MPU means that a number of renderer processes are started which can take advantage of both multi-processor and multi-pipe systems. Each MPU process now provides distributed OpenGL rendering capabilities. To maintain consistency with other renderers, the existing OpenGL renderer layer was extended to take advantage of MPU. As the independent layer traverses the scene graph to render a frame, calls are made to the MPU layer. These

calls are tokenized and stored in shared memory (which is used to pass data to MPU). The tokens are later processed by the MPU renderers and the original OpenGL calls recreated.

AVS/Express already includes a cache system and this functionality was included in the MPU layer to reduce the amount of data transfer to shared memory and improve performance. A two level cache system was employed: firstly keeping a copy of an object in shared memory as more than one MPU renderer process may require access to it at different times. Secondly, each renderer process makes a GL List that may be repeatedly called in preference to traversing the same OpenGL tokens.

The MPU library blocks while a frame is being rendered which stops the application from performing visualization tasks. This problem is further exacerbated when considering realtime updates in an immersive environment, for which AVS has not been designed. An intermediate Frame Management Process (FMP) was introduced to alleviate this problem by separating the handling of the rendering from the application process. When a frame is ready (fully tokenized) the application notifies the FMP which in turn notifies the MPU processes and is, itself, blocked. The application continues with its tasks and produces the next frame. For head-tracking support this means that the camera can now be decoupled from the application, the FMP piloting the centre of the view with camera offsets without interruption of the application. If a new frame is ready then the FMP will immediately use it, otherwise it will re-render the previous frame according to the new camera position.

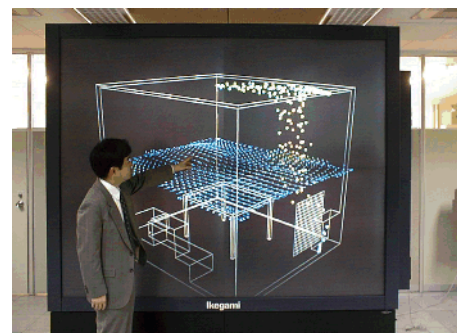


Figure 2. Multi-pipe Powerwall display using MPE.

## Conclusions

The AVS/Express Multi-Pipe Edition is now available as a product and has been tested extensively at a number of key sites. Currently support is included for rendering the majority of geometric and volumetric primitives that AVS/Express provides, within most configurations supported by MPU. The next phase of work includes the addition of further support for head-tracking, wand interaction and VE user interface tools.

Problems have arisen during the implementation, in particular picking of objects has proved difficult to solve, as the existing picking mechanism in AVS/Express is tightly coupled with the standard renderers and assumes a single window is in use rather than multiple windows/channels as found in some MPU configurations.

In most applications the MPU renderer layer remains transparent and very little effort has been needed to get existing applications working with MPE. The future addition of interactors suitable for use in a VE will require changes at the application level but such interactors will still work when using the standard renderers and, as such, the transparency of the MPE will be retained.