

**SIXTH FRAMEWORK PROGRAMME  
PRIORITY IST-2002-2.3.1.8  
Networked Audiovisual Systems**

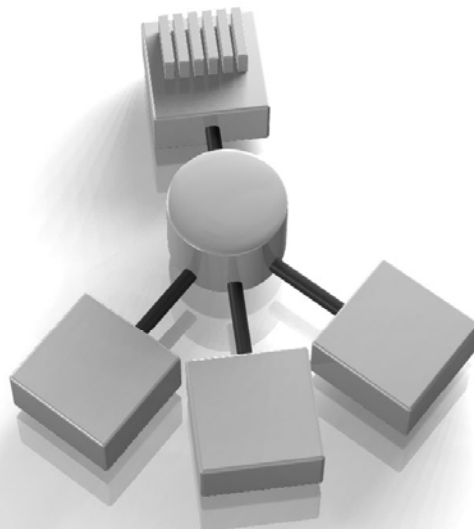


**Uni-verse project**

**Deliverable D 4.2**

**High Performance Rendering Client**

**June 1, 2006**



STREP project

Project acronym: Uni-Verse

Project full title: A Distributed Interactive Audio-Visual Virtual Reality System

Proposal/Contract no.: 002228

Distribution: Public

|   |          |
|---|----------|
| <b>WP 4.1 RENDERING CLIENTS – HIGH PERFORMANCE RENDERING CLIENT</b> | <b>3</b> |
| <b>1 Introduction</b>   | <b>3</b> |
| <b>2 Description</b>  | <b>3</b> |
| <b>3 Subdivision Surfaces</b>                                       | <b>4</b> |
| <b>4 Integration of the Radiosity Module (D5.5)</b>                 | <b>4</b> |
| <b>5 Application Areas</b>  | <b>5</b> |
| <b>6 Conclusions</b>  | <b>5</b> |
| <b>7 Availability</b>   | <b>5</b> |
| <b>8 References</b>   | <b>6</b> |

# WP 4.1 Rendering clients – High Performance Rendering Client

## 1 Introduction

In the context of this project the term “rendering client” denotes an application that connects to a Verse host with the main purpose of visualizing its data content. This is opposite to e.g. modeling tools, which aim to give users the ability to create new content. While such tools will need to render Verse graphics, they have different goals. In the early start of the project it was decided to develop two rendering platforms for the Uni-Verse System (see D2.3/2.4). The High Performance Rendering Client is one of these two Rendering Engines developed within the project.

## 2 Description

The High Performance Rendering Client developed at Fraunhofer IGD uses version 1.6 of the open source scene graph OpenSG [1] as rendering base. The rendering client runs multithreaded and allows through the connection to OpenSG the rendering of Verse data to large and immersive devices using a cluster. The client was developed using C++, Visual Studio .NET 2003, QT [4]. The user interface and framework of the application is realized using QT functionality for widgets and threads. The initial development of the renderer started with the implementation of release 4 of Verse and was updated to release 5 during the life time of the project.

The application consists of two threads, one responsible for the network Verse communication and the other one for rendering and managing the user interface. The Verse commands are transformed and mapped into appropriate OpenSG commands as far as the design of OpenSG allows these mappings. Figure 1 shows the communication structure of the rendering client.

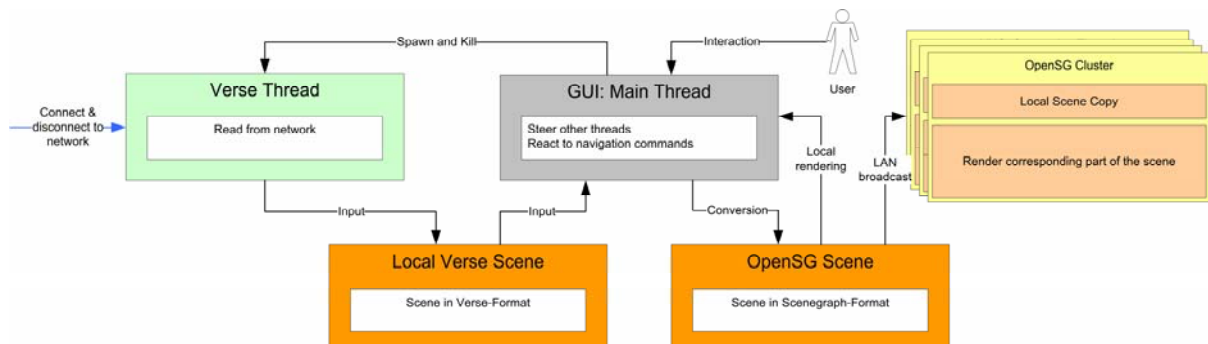


Figure 1: Communication structure of the High Performance Rendering Client.

The OpenSG engine allows the user to distribute the rendered scene to multiple output screens to allow the rendering of the scene to composed screen walls or multiple monitors. The user interface contains a structured list of all Verse nodes downloaded and rendered from the server. This list can be used to select nodes or objects and center the view around them.

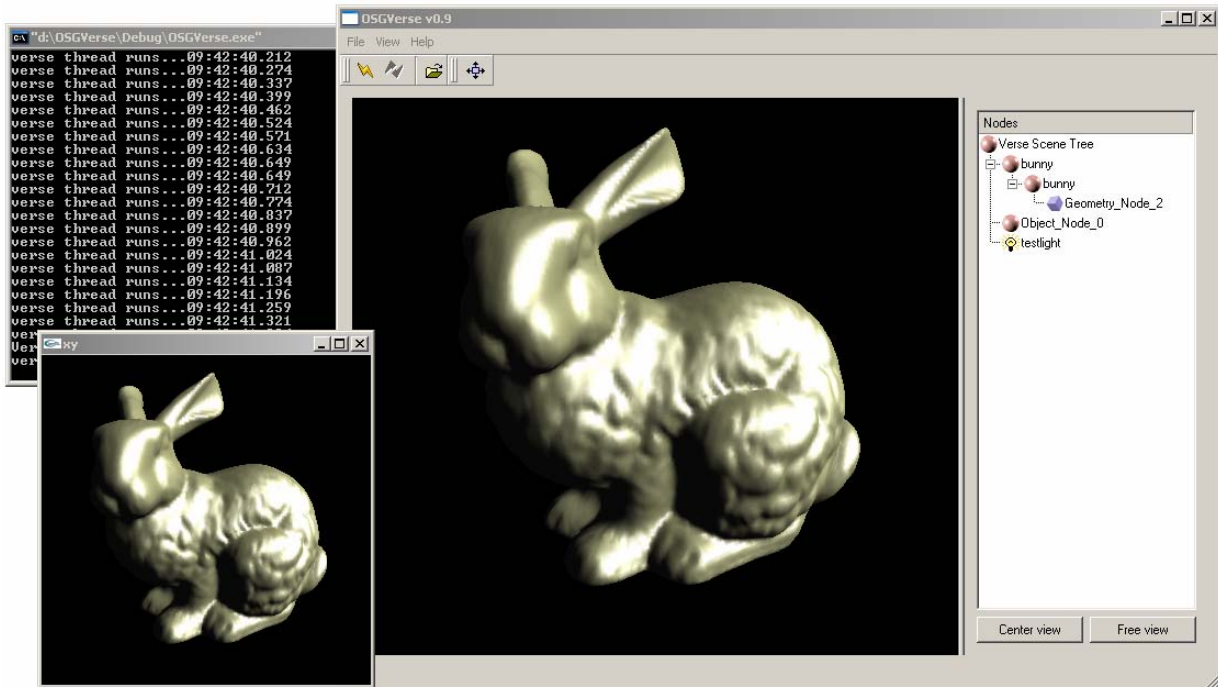


Figure 2: User interface and multiple output windows showing the Stanford Bunny downloaded from the Verse server

The selection of two navigation modes is integrated into the user interface, one for a centered view and one for a free view. The centered view mode allows the user to select a center point and turn the camera around, e.g. A Verse object. The free view mode is a "fly around" mode. It allows a game-like navigation of the camera through the scene. The avatar name of the renderer in the Verse scene can be renamed and the position and orientation of the avatar is uploaded back to Verse every time the user changes the camera. Verse defines its polygon data in a clockwise order. Because many other applications and data formats define the polygons in counter clockwise order a command was integrated into the view menu to switch between these modes for a maximum flexibility using the renderer for different data sources.

### 3 Subdivision Surfaces

Because Verse stores all data as subdivision surface representation, a Catmull Clark subdivision surface algorithm [3] was implemented and integrated into the rendering pipeline. The subdivision algorithm evaluates the Verse data and calculates a subdivision representation before it is pipelined into the scene graph engine. The subdivision algorithm can be toggled on or off in the user interface.

### 4 Integration of the Radiosity Module (D5.5)

The Radiosity Module developed in work package 5 was integrated into the High Performance Rendering Client. The module is available as an additional rendering option within the clients user interface. The radiosity module enables the user to view the data

from the Verse server with a realistic real-time light simulation. For more detailed information about the Radiosity Module take a look at the deliverable report D5.5.

## **5 Application Areas**

OpenSG itself allows the rendering of a 3D scene on a cluster. A very impressive application example of this technology is the HEyeWall [3] at Fraunhofer IGD. It is a very high resolution presentation wall consisting (in this case at IGD) of 6144 x 3072 pixels at a presentation area of about 6mx4m. The HEyeWall is driven by 48 beamers. The OpenSG scene graph enables the possibility to render a 3D scene into a cluster with 48 computers that each drives one of the beamers. The scene graph splits the scene into tiles and distributes them to the cluster nodes. Each node renders a tile and uses the beamers to display them. More Information about this technology can be found at <http://www.heyewall.de>. Because the High Performance Renderer makes use of OpenSG it is possible to present Verse data on such an immersive device.

## **6 Conclusions**

In work package 4.1 a rendering system was developed that includes Verse access via the Verse server, a subdivision algorithm, a real-time radiosity module implementation, the OpenSG scene graph technology and with that, the possibility to use immersive devices like the HEyeWall as presentation area for the rendered 3D scenes. The client was implemented and tested under Windows32 but no windows specific commands (like MFC) are used during the implementation of the renderer. The QT interface and thread implementation is platform independent as well as the used STL library. Therefore the client can be compiled on other systems as well.

## **7 Availability**

The client is available as source code and binaries at the Uni-Verse website.

## **8 References**

- [1] Open SG scene graph, <http://www.opensg.org> (last visited June 2006)
- [2] HEyeWall, <http://www.heyewall.de>
- [3] CATMULL, E., AND CLARK, J. Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes. *Computer Aided Design* 10, 6 (1978), 350–355.
- [4] QT from Trolltech, <http://www.trolltech.com>