

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

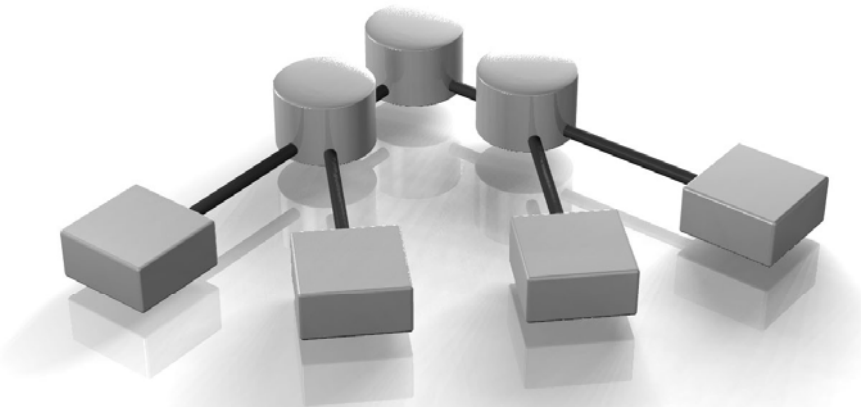


Uni-Verse Project

D4.2.2 High-Quality Rendering Client

October 1, 2006

Distribution: Public



STREP project

Project acronym: Uni-Verse

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

Proposal/Contract no.: 002228

Table of Contents

Table of Contents	2
Introduction	3
Implementation.....	3
Conclusions	4
Availability.....	4

Introduction

This report describes the high-quality rendering client “Quel Solaar” developed by Eskil Steenberg, KTH. This rendering client is based on the rendering library "Persuade" making much of the code shareable with future projects.

Quel Solaar is fully dynamic and requires no pre-processing of data. It has been designed to be used for any application demanding high quality real time visuals, such as games, simulation, visualization, education and social activities. The main difference from other engines is that Quel Solaar is designed to be fully networked. This means that all data visualized in the engine comes from a Verse server and that any part of the scene can be changed at any time

Quel Solaar and Persuade are released as open source with the GNU license.

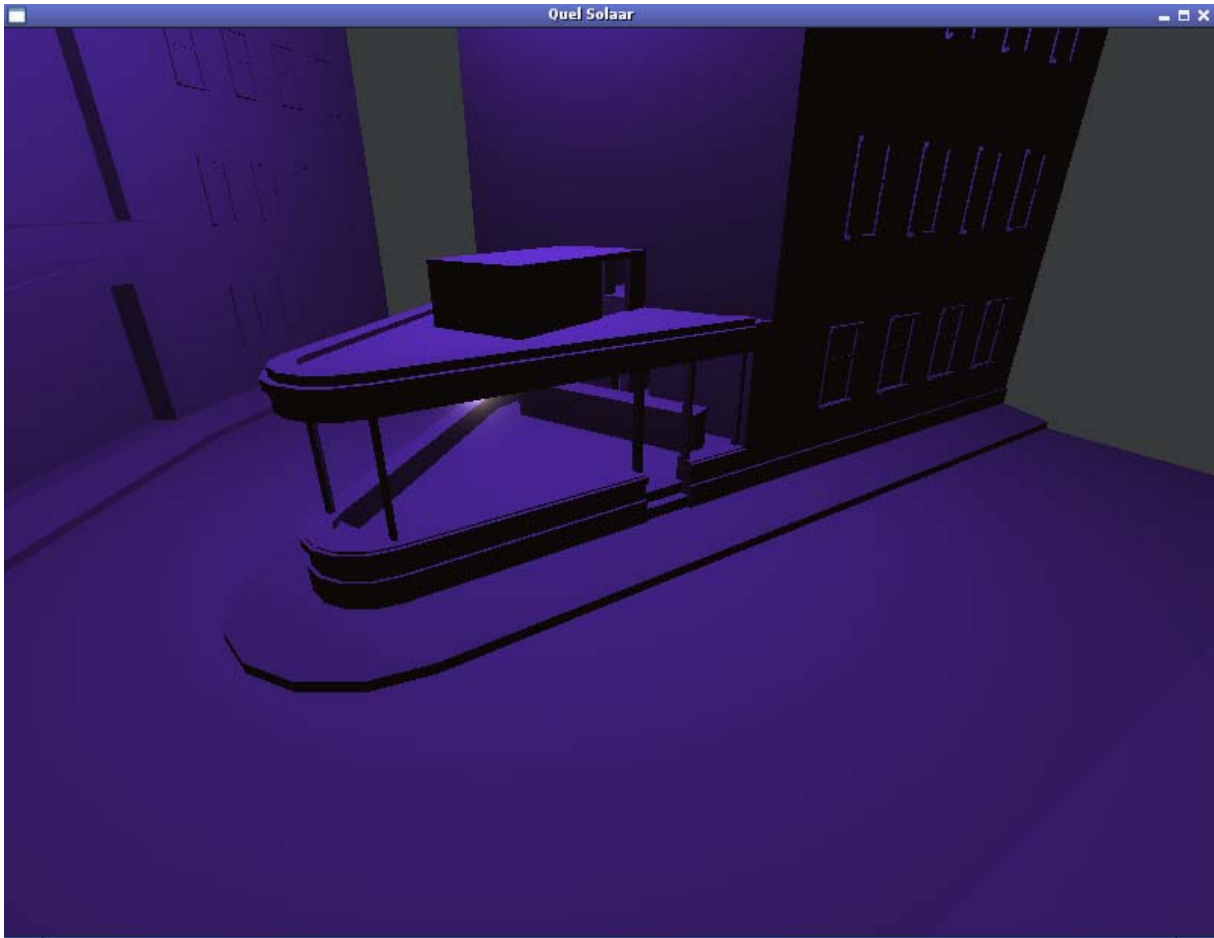
Implementation

Quel Solaar is an application written in C using the SDL (<http://www.libsdl.org/>) or Glut (<http://www.opengl.org/resources/libraries/glut/>) libraries for user interaction and the "Persuade" rendering library, the latter developed by Eskil Steenberg, for the rendering of Verse data. Persuade uses the Enough library and OpenGL. Any application with a Open GL context and Enough is able to use the Persuade library, and it is also utilized in Connector and Loq Airou. It is based on OpenGL 2.0 but also has a 1.x fallback, however this fallback is fairly trivial.

The Persuade library features many advanced graphical concepts such as global illumination, displacement mapping and programmable shaders The Persuade API enables the user to draw entire scenes or individual objects. Persuade also includes code for displaying bitmap nodes as Open GL textures.

This rendering engine is intended to be very advanced. The heart of the engine consists of an operating system like task manager that divides up all computation to ensure a smooth frame rate even if large sets of data change. The data coming from verse is processed in real-time and go through many stages of optimization in order to be displayed. The geometry pipeline goes through a series of stages, and depending on how the geometry is manipulated it only recomputes the smallest possible number of stages in order to display the changes. The geometry pipeline first subdivides the geometry using Catmull/Clark subdivision surfaces scheme also taking creases into account. Then the engine creates Level Of Details (LODs) using a reduction algorithm that removes unwanted geometry. The sub division surface (SDS) engine does not actually compute the new shape, but rather how the control mesh relates to the surface. This means that when the control mesh is modified or animated, one can quickly recompute the shape.

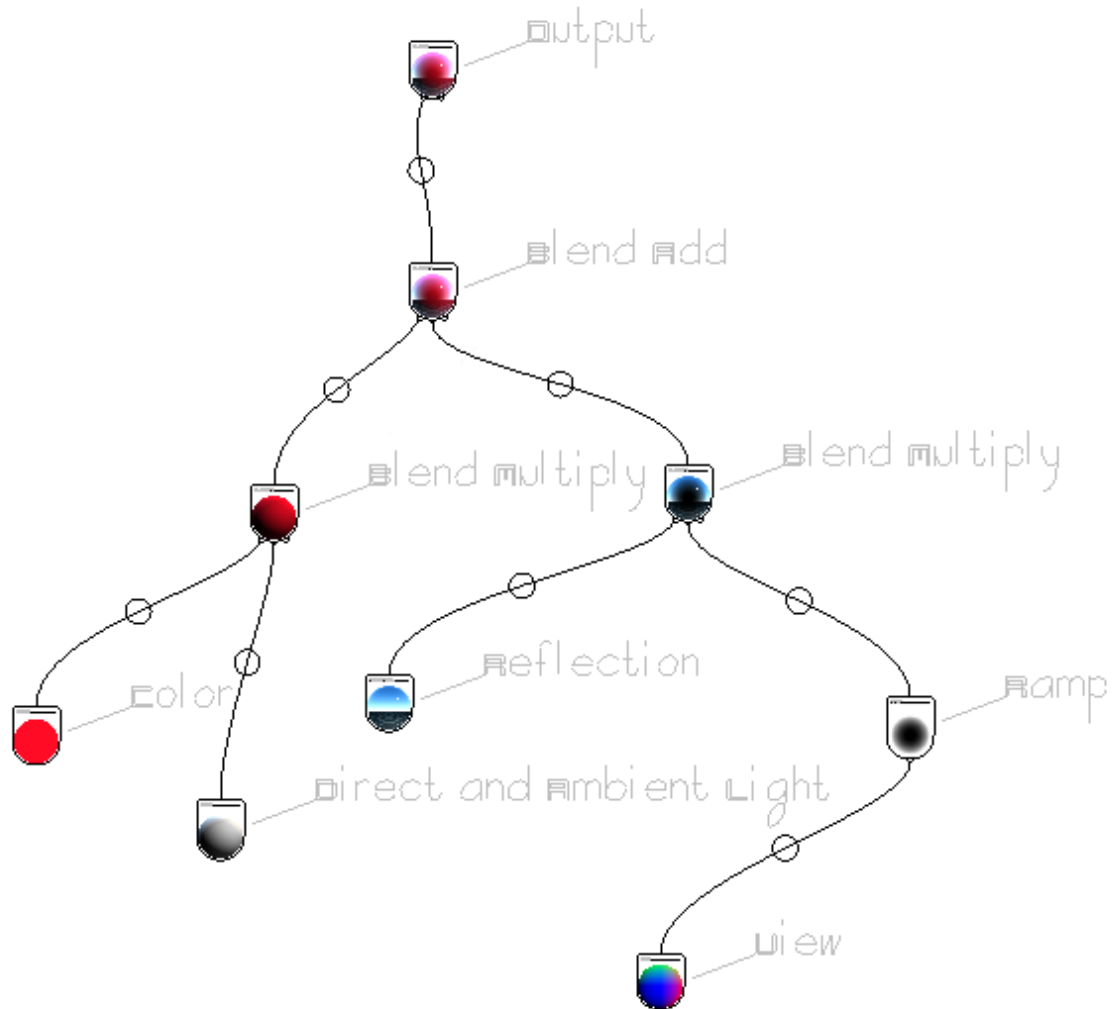
D4.2 High-performance rendering client



Figur 1

The geometry engine also supports displacement mapping, as well as geometry computed for use as stencil shadow geometry. The surfaces are rendered using OpenGL 2.0s shading language and compute the surface shading per pixel. The engine takes the verse shading tree and converts it into GL's shading language, compile, link and run it in real time.

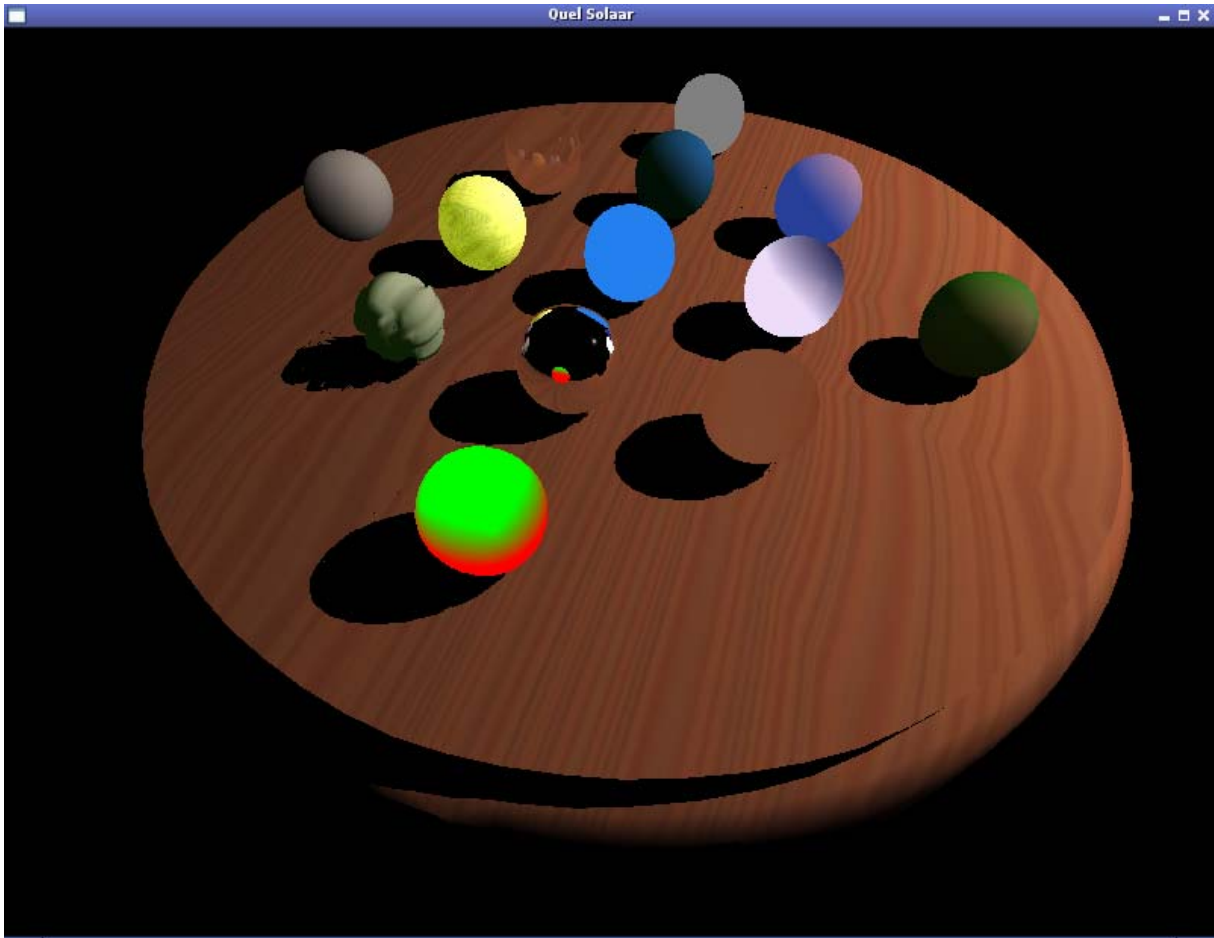
D4.2 High-performance rendering client



```
vec3 v, light = vec3(0, 0, 0), ambient;
float f, dist;
v = gl_LightSource[0].position.xyz - pixel_pos.xyz;
dist = length(v);
v = normalize(v);
f = max(0.0, dot(normal, v)) / dist;
light += gl_LightSource[0].diffuse.rgb * vec3(f, f, f);
ambient = textureCube(diffuse_environment, normal).xyz;
gl_FragColor = vec4(vec3((((color_1 * vec4(light + ambient, 0)) +
(vec4(textureCube(environment,
reflect(pixel_pos.xyz, normal.xyz) *
gl_NormalMatrix).xyz, 1) * (texture1D(ramp_7,
((vec4(normal.xyz, 1)).b - -0.017347) /
0.995731)))))).rgb), 1.0);
```

Text 1: Sample generated shading code

This allows users to see in real-time the changes they make in material editors. The shading language will allow us to replicate the verse shading tree very closely in real-time. Reflections, Refractions and Transparency can be simulated using the Verse material system.



Figur 2 Material test model

The engine features a simple but effective global illumination algorithm. Each object maintains a list of surrounding objects, and light and draw them into environment cube maps. The shading engine will then use these cube maps to get the ambient term on the lighting calculations.

Conclusions

A rendering client with high-quality implementing all aspects of the Verse protocol has been implemented. We believe that this engine will be rated unique because it does all computation on the fly, whereas most engines require off-line pre computation of all data. Even if there is no pre computation of data we believe that the feature set of this engine will rival the most advanced engines on the market.

Availability

The client is available as source code and binaries for Linux, Windows and Mac OS X at the Uni-Verse website.