QDE. A system for composing real time computer graphics.

Sven Osterwalder
Bern University of Applied Sciences
BFH-TI, CH2502-Biel, Switzerland.
Email: sven.osterwalder@students.bfh.ch

***Abstract*—The design and development of a software for *modeling*, *composing* and *rendering* real time computer graphics is presented. Modeling and composing are facilitated by a graphical user interface providing toolbox like elements. For rendering a highly optimized algorithm based on ray tracing called sphere tracing is used. The algorithm allows rendering ray traced scenes in real time on the GPU. For the development of the software a method called *literate programming* is used.**

I. Introduction

COMPUTER SCIENCE HAS ALWAYS STRIVED to create representations of scenes and models, that are as near to the human reality as possible. One such representation is *ray tracing*, which is based on the physics of light as well as on surface materials. In contrast to ray tracing, *sphere tracing* allows rendering ray traced images in real-time. The de facto way of representing objects however, using triangle based meshes, cannot be used directly with this method. Instead distance fields defined by implicit functions build the basis. At present there are no solutions (known to the author), that provide a convenient way to use implicit functions for modeling and sphere tracing for rendering. This thesis presents the design and development of a software which provides both: a modular, node based approach for modeling and animating objects using implicit functions while rendering in real time on the graphics card using sphere tracing.

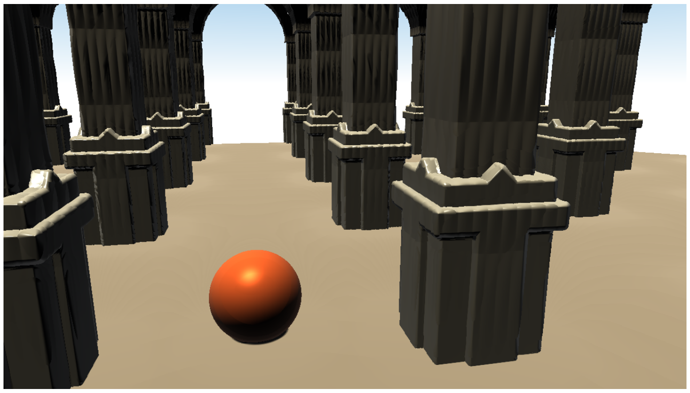

Fig. 1. A scene modeled with signed distance functions and rendered with sphere tracing.

II. Approach

TO REACH THE INTENDED GOAL, the approach was to develop a software architecture, use literate programming and the agile methodology extreme programming for development. *Literate programming* was introduced in 1984 by Knuth and proposes to consider programs to be works of literature and to explain to human beings what the computer shall do to achieve certain goals. To overcome one of the main challenges when developing the software — change — an adapted version of extreme programming was used. This methodology was chosen as after the preceding project work several things were still subject to change and therefore an exact planning, analysis and design, as traditional methodologies require it, would not have been very practical.

III. Implementation

THE RESULTS OF THIS THESIS are an architecture for a software and the software itself, written using the literate programming paradigm. THREE ASPECTS DEFINE THE SOFTWARE ARCHITECTURE: 1) The model-view-view model software design pattern using additionally controllers, 2) the layers software architectural pattern and 3) the observer software design pattern, allowing communication between components of the software. THE SOFTWARE ITSELF is an editor which allows *modeling* objects, *composing* objects to scenes and *rendering* scenes in real-time. Scenes are stored in a scene graph structure and represented by a tree view. Each scene can contain one or more objects, being defined by external files, represented as nodes in a node based graph structure. The parameters of objects are used for interconnection between nodes. For rendering the sphere tracing algorithm was used, which was established in the preceding project work.

IV. Conclusion and Outlook

LITERATE PROGRAMMING TAKES SOME TIME TO GET INTO as it requires another way of thinking as when developing software in conventional ways. Albeit this circumstance, the set of basic goals could be reached. Sphere tracing is a very interesting and promising approach to rendering and it starts to getting used by the industry, for example for calculating ambient occlusion or soft shadows. Time will tell if it will establish itself further and may even be used for rendering conventional meshes.

References

[1] D. E. Knuth, “Literate programming,” *The Computer Journal*, vol. 27, no. 2, pp. 97–111, May 1984, ISSN: 0010-4620. DOI: 10.1093/comjnl/27.2.97.