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 is used. The algorithm is called sphere tracing and allows the rendering of 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. Its usage for real-time rendering, calculating at least 25 images in one second, was due to its computational complexity not feasible until the last few years. An implementation allowing rendering ray traced images in real-time is *sphere tracing*. Albeit this advantage, this method has admittedly a clear disadvantage: the de facto way of representing objects, using triangle based meshes, cannot be used directly. Instead distance fields defined by implicit functions build the basis for sphere tracing. To this point in time there are no, known to the author, solutions that provide a convenient way for modeling, animating and rendering objects and scenes which use implicit functions for modeling and sphere tracing for rendering. This thesis presents the design and development of a software which provides both: a node based approach for modeling and animating objects using implicit functions as well as allowing the composition of scenes while rendering in real time on the graphics card using 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. Often software is not documented properly or the documentation is even neglected as it can be quite costly with seemingly little benefit. But no documentation at all, outdated or irrelevant documentation can lead to unforeseen efforts concerning time and costs. To prevent such unforeseen efforts the developed software and this thesis was written with a paradigm called *literate programming*. The paradigm was introduced in 1984 by Knuth. It 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 represented as nodes in a node based graph structure. Objects are defined by external files and can represent implicit functions, cameras or effects. Every object has one or more parameters (such as size, color and so on) which can be used to connect 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] S. Osterwalder, *QDE - a visual animation system. Software-Architektur.* Bern University of Applied Sciences, Aug. 5, 2016.

[2] ——, *Volume ray casting - basics & principles*. Bern University of Applied Sciences, Feb. 14, 2016.

[3] 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.

[4] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change (2Nd Edition)*. Addison-Wesley Professional, 2004, ISBN: 0321278658.