QDE – A system for composing real time computer graphics

Degree programme: Master of Science in Engineering | Specialisation: Informations- und Kommunikationstechnologien
Thesis advisor: Claude Fuhrer
Expert: Eric Dubuis (COMET Group)

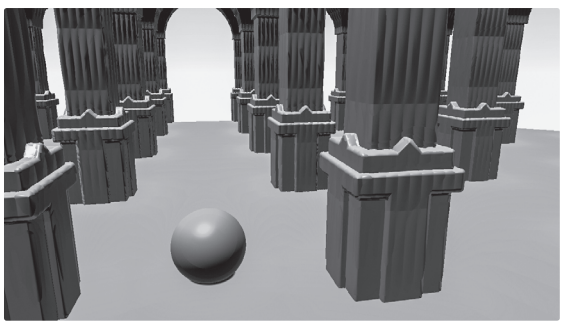
The design and development of a program for modeling, composing and rendering computer graphics in real-time is presented. Modeling and composing are facilitated by a graphical user interface providing toolbox elements. For rendering, a highly optimized algorithm based on ray tracing is used: sphere tracing. For the development and documentation of the software, a method called literate programming is used.

Introduction

Computer science has always strived to create representations of scenes and models as near to human reality as possible. One such representation is ray tracing, based on the physics of light as well as the properties of surface materials. In contrast to ray tracing, sphere tracing allows the rendering of ray traced images in real-time. However, the de facto way of representing objects using triangle based meshes cannot be used directly with this method. Instead, an alternative basis uses distance fields defined by implicit functions. 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 program which provides both a modular, node based approach for modeling and animating objects using implicit functions, and allows rendering in real-time using sphere tracing.

Approach

To reach the intended goal, the approach was to develop a software architecture and to use literate programming and the agile methodology of extreme programming for development. Literate programming was introduced in 1984 by Donald Knuth and considers programs as works of literature. A literate program includes documentation which explains in human terms what the computer must do. To overcome one of the main challenges when developing the software – continual change – an adapted version of extreme programming was used. This methodology was selected because of experiences with projects involving continual adaption in which exact planning, analysis and design (as traditional methodologies require) would not have been very practical.



A scene modelled using signed distance functions and rendered with sphere tracing.

Implementation

The results of this thesis are an architecture for a software program, and the program itself, written using the literate programming paradigm. Three aspects define the software architecture:

1) the model-view-view model software design pattern using controllers in addition,
2) the layered 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 are represented by a tree view. Each scene can contain one or more objects, defined by external files and represented as nodes in a node graph structure. The parameters of objects are used for interconnections between nodes. For rendering, the sphere tracing algorithm established in the preceding project work is used.

Conclusion and outlook

Literate programming takes some accustomization as it requires a different way of thinking to that of conventional software development. Despite this, the basic goals could be reached. Sphere tracing is a very interesting and promising approach to rendering which is coming into use in the industry, for example for calculating ambient occlusion or soft shadows. Time will tell if the method will establish itself further and become practicable for rendering conventional meshes.

Sven Osterwalder