QDE.

A SYSTEM FOR COMPOSING REAL TIME COMPUTER GRAPHICS.

MTE7103 — MASTER THESIS

Major:	Computer science
Author:	Sven Osterwalder[1]
Advisor:	Prof. Claude Fuhrer[2]
Expert:	Dr. Eric Dubuis[3]
Date:	2017-05-24 13:56:31
Version:	ae34fc5

Berne University of Applied Sciences

[1] sven.osterwalder@students.bfh.ch
[2] claude.fuhrer@bfh.ch
[3] eric.dubuis@comet.ch

Revision History

Revision	Date	Author(s)	Description
89c7544	2017-02-28 17:09:43	SO	Set up initial project structure, provide first content
10192d8	2017-03-02 22:37:17	SO	Set up project schedule
54f4b23	2017-03-05 22:53:15	SO	Set up project structure, implement main entry point and main window
ffda0e5	2017-03-15 10:58:51	SO	Scene graph, logging, adapt project schedule
34b09b7	2017-03-24 17:08:22	SO	Update meeting minutes, thoughts about node implementation
06fe268	2017-04-03 23:00:35	SO	Add requirements, node graph implemenetation
d264175	2017-04-10 16:07:01	SO	Conversion from Org-Mode to Nuweb, revise editor implementation
f8b524b	2017-04-30 23:42:57	SO	Approach node graph and nodes
2f46832	2017-05-04 21:13:41	SO	Impel node definitions further
ae34fc5	2017-05-24 13:56:31	SO	Change class to tufte-book, title page, introduction, further implementation

Abstract

Provide correct abstract.

A highly optimized rendering algorithm based on ray tracing is presented. It outperforms the classical ray tracing methods and allows the rendering of ray traced scenes in real-time on the GPU. The classical approach for modelling scenes using triangulated meshes is replaced by mathematical descriptions based on signed distance functions. The effectiveness of the algorithm is demonstrated using a prototype application which renders a simple scene in real-time.

Contents

List of Figures

List of Tables

Introduction

The subject of computer graphics exists since the beginning of modern computing. Ever since the subject of computer graphics has strived to create realistic depictions of the observable reality. Over time various approaches for creating artificial images (the so called rendering) evolved. One of those approaches is ray tracing. It was introduced in 1968 by Appel in the work "Some Techniques for Shading Machine Renderings of Solids" [1]. In 1980 it was improved by Whitted in his work "An Improved Illumination Model for Shaded Display" [2].

Ray tracing captivates through simplicity while providing a very high image quality including perfect refractions and reflections. For a long time although, the approach was not performant enough to deliver images in real time. Real time means being able to render at least 25 rendered images (frames) within a second. Otherwise, due to the human anatomy, the output is perceived as either still images or as a too slow animation.

Sphere tracing is a ray tracing approach introduced in 1994 by Hart in his work "Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces" [3]. This approach is faster than the classical ray tracing approaches in finding intersections between rays and objects. The speed up is achieved by using signed distance functions for modeling the objects to be rendered and by expanding volumes for finding intersections.

Graphics processing units (GPUs) have evolved over time and have gotten more powerful in processing power. Since around 2009 GPUs are able to produce real time computer graphics using sphere tracing. While allowing ray tracing in real time on modern GPUs, sphere tracing has also 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.

Purpose and situation

Motivation

To this point in time there are no solutions (at least none are known to the author), that provide a convenient way for modeling, animating and rendering objects and scenes using signed distance functions for modeling and sphere tracing for rendering. Most of the solutions using sphere tracing implement it by having one or multiple big fragment shaders containing everything from modeling to lighting. Other solutions provide node based approaches, but they allow either no sphere tracing at all, meaning they use rasterization, or they provide nodes containing (fragment-) shader code, which leads again to a single big fragment shader.

This thesis aims at designing and developing a software which provides both: a node based approach for modeling and animating objects using signed distance functions as well as allowing the composition of scenes while rendering objects, or scenes respectively, in real time on the GPU using sphere tracing.

Objectives and limitations

The objective of this thesis is the design and development of a software for *modeling*, *composing* and *rendering* real time computer graphics through a graphical user interface.

Modeling is done by composing single nodes to objects using a node based graph structure.

Compositing includes two aspects: the composition of objects into scenes and the composition of an animation which is defined by multiple scenes which follow a chronological order. The first aspect is realized by a scene graph structure, which contains at least a root scene. Each scene may contain nodes. The second aspect is realized by a time line, which allows a chronological organization of scenes.

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. Contingent upon the used rendering algorithm all models are modeled using implicit surfaces. In addition mesh-based models and corresponding rendering algorithms may be implemented.

Required objectives are the following:

- Development of an editor for creating and editing real time rendered scenes, containing the following features.

  - A scene graph, allowing management (creation and deletion) of scenes. The scene graph has at least a root scene.
  - A node-based graph structure, allowing the composition of scenes using nodes and connections between the nodes.
  - Nodes for the node-based graph structure.
    * Simple objects defined by signed distance functions: Cube and sphere
    * Simple operations: Merge/Union, Intersection, Difference
    * Transformations: Rotate, Translate and Scale
    * Camera
    * Renderer (ray traced rendering using sphere tracing)
    * Lights

OPTIONAL OBJECTIVES are the following:

- Additional features for the editor, as follows.
  - A sequencer, allowing a time-based scheduling of defined scenes.
  - Additional nodes, such as operations (e.g. replication of objects) or post-processing effects (glow/glare, color grading and so on).
- Development of a standalone player application. The player allows the playback of animations (time-based, compounded scenes in sequential order) created with the editor.

Related works

PRELIMINARY to this thesis two project works were done: "Volume ray casting — basics & principles" [4], which describes the basics and principles of sphere tracing, a special form of ray tracing, and "QDE — a visual animation system, architecture" [5], which established the ideas and notions of an editor and a player component as well as the basis for a possible software architecture for these components. The latter project work is presented in detail in the chapter about the procedure, the former project work is presented in the chapter about the implementation.

Document structure

This document is divided into N chapters, the first being this introduction. The second chapter on *administrative aspects* shows the planning of the project, including the involved persons, deliverables and the phases and milestones.

The administrative aspects are followed by a chapter on the *procedure*. The purpose of that chapter is to show the procedure concerning the execution of this thesis. It introduces a concept called

literate programming, which builds the foundation for this thesis. Furthermore it establishes a framework for the actual implementation, which is heavily based on the previous project work, "QDE — a visual animation system, architecture" [5] and also includes standards and principles.

The following chapter on the *implementation* shows how the implementation of the editor and the player component as well as how the rendering is done using a special form of ray tracing as described in "Volume ray casting — basics & principles" [4]. As the editor component defines the whole data structure it builds the basis of the thesis and can be seen as main part of the thesis. The player component re-uses concepts established within the editor.

Given that literate programming is very complete and elaborated, as components being developed using this procedure are completely derived from the documentation, the actual implementation is found in the appendix as otherwise this thesis would be simply too extensive.

The last chapter is *discussion and conclusion* and discusses the procedure as well as the implementation. Some further work on the editor and the player components is proposed as well.

After the regular content follows the *appendix*, containing the requirements for building the before mentioned components, the actual source code in form of literal programming as well as test cases for the components.

Administrative aspects

Some administrative aspects of this thesis are covered, they are although not required for understanding of the result.

The whole documentation uses the male form, whereby both genera are meant equally.

Involved persons

Role	**Name**	**Task**
Author	Sven Osterwalder[1]	Author of the thesis.
Advisor	Prof. Claude Fuhrer[2]	Supervises the student doing the thesis.
Expert	Dr. Eric Dubuis[3]	Provides expertise concerning the thesis's subject, monitors and grades the thesis.

Table 2: List of the involved persons.

[1] sven.osterwaldertudents.bfh.ch
[2] claude.fuhrer@bfh.ch
[3] eric.dubuis@comet.ch

Deliverables

Deliverable	**Description**
Report	The report contains the theoretical and technical details for implementing a system for composing real time computer graphics.
Implementation	The implementation of a system for composing real time computer graphics, which was developped during this thesis.

Table 3: List of deliverables.

Organization of work

Meetings

Various meetings with the supervising professor, Prof. Claude Fuhrer, and the expert, Dr. Eric Dubuis, helped reaching the defined goals and preventing erroneous directions of the thesis. The supervisor supported the author of this thesis by providing suggestions throughout the held meetings. The minutes of the meetings may be found under meeting minutes.
Add correct reference

Phases of the project and milestones

Phase	**Week / 2017**
Start of the project	8
Definition of objectives and limitation	8-9
Documentation and development	8-30
Corrections	30-31
Preparation of the thesis' defense	31-32

Table 4: Phases of the project.

Milestone	**End of week / 2017**
Project structure is set up	8
Mandatory project goals are reached	30
Hand-in of the thesis	31
Defense of the thesis	32

Table 5: Milestones of the project.

Schedule

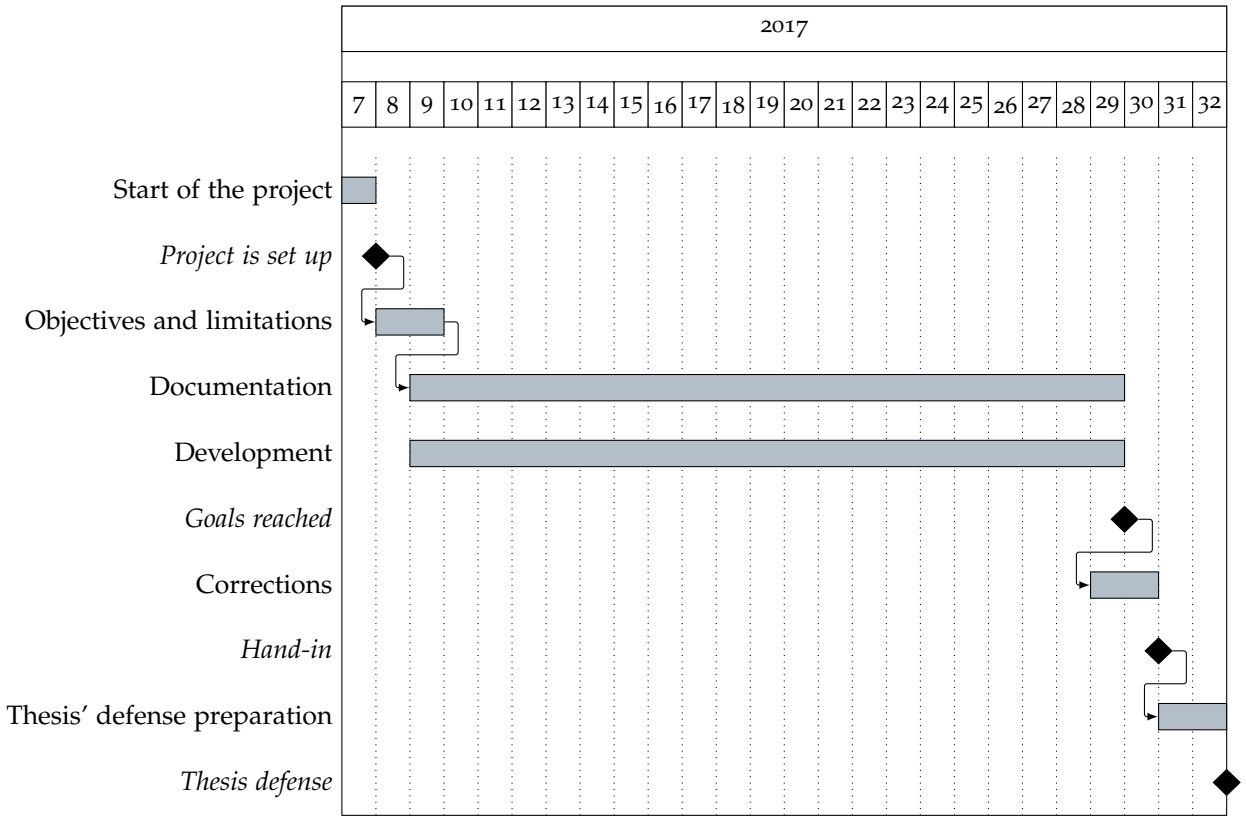

Figure 1: Schedule of the project. The subtitle displays calendar weeks.

Fundamentals

Write chapter.

Software architecture

Write chapter.

Rendering

Write chapter.

Methodologies

Write chapter.

Results

Write chapter.

Discussion and conclusion

Write chapter.

fix appendix

Bibliography

[1] A. Appel, "Some Techniques for Shading Machine Renderings of Solids", in *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, ser. AFIPS '68 (Spring), New York, NY, USA: ACM, 1968, pp. 37–45. DOI: 10.1145/1468075.1468082.

[2] T. Whitted, "An Improved Illumination Model for Shaded Display", *Commun. acm*, vol. 23, no. 6, pp. 343–349, Jun. 1980, ISSN: 0001-0782. DOI: 10.1145/358876.358882.

[3] J. C. Hart, "Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces", *The visual computer*, vol. 12, pp. 527–545, 1994.

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

[5] ——, *QDE - a visual animation system. software-architektur.* Bern University of Applied Sciences, Aug. 5, 2016.

Fix glossaries

Print index