Philipps-Universität Marburg

Fachbereich 12 - Mathematik und Informatik

Master Thesis

Dynamic Insertion of 3D Objects from CAD Files into Unreal Engine

Matija Miskovic

Septermber 2022

Supervisor:
Prof. Dr. Thorsten Thormählen

Research Group Graphics and Multimedia Programming

Declaration of Originality

I, Matija Miskovic (Computer Science Student at Philipps-University Marburg, Matrikelnummer 3139015), versichere an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Die hier vorliegende Diplomarbeit wurde weder in ihrer jetzigen noch in einer ähnlichen Form einer Prüfungskommission vorgelegt.

Marburg, 17. Dezember 2009

Max Mustermann

Abstract

Viele der in der Computergrafik verwendeten 3D-Modelle werden mit Hilfe der Dreiecksnetze repräsentiert. ... (max. 1 Seite)

Abstract

text text text text text text text text text text text text text text text text text text text text text text (exakte englische Übersetzung der deutschen Kurzfassung)

Table of Contents

1 Introduction

The topic of this thesis the dynamic insertion of 3D objects, defined in CAD Files, into an Unreal Engine program while it is running. Especially important for the project are why this might even a problem and how it can actually be realized. For these purposes an Unreal Engine plug-in was developed, which enables such a feature, and an additional Unreal Engine program which implements the plug-in and demonstrates some ways of interacting with objects generated in such a way.

1.1 Motivation

1.2 Goals

The main goal of this work is to develop an efficient and user-friendly plug-in which will make it possible to load 3D objects from the most common CAD formats during the runtime of an Unreal Engine program. Additionally another software should be developed to use the plug-in and allow simple interactions with these objects in a normal desktop window as well as in a virtual reality environment.

Efficiency

The developed plug-in should be capable of handling large amounts of data seeing as the models which can be found in CAD files can be incredibly large, containing thousands or millions of vertices and polygons. If the plug-in were to effect the runtime performance in a significant way, such as causing stutters or freezing the program all together, it would severely worsen the user experience and invalidate the whole point of the program.

Expandability

The field of computer assisted design is very wide and there are countless programs and formats for all the varying use-cases in which it is being used. That is why creating one solution for all of those is incredibly complicated and way out of the scope and possibilities of this project. Instead it is much better to concentrate on creating a simple to use and understand system which can then be further improved upon and adjusted for the concrete cases of clients or projects.

1.3 Thesis Structure

In Chapter 2 the Unreal Engine will be clarified and explained. Seeing as this is both the tool which is being used for development as well as being the software for which the plug-in is being developed, an understanding of how it works and what its limitations are is needed in order to better grasp the project and what problems might arise. It is a rather expansive tool so not everything will be covered, only the more basic aspects and the concrete parts which play a role for this project. Then, in Chapter 3, the plug-in will be analysed, starting of with how the files are parsed and into what sort of form they are transformed in order to be used. After that comes the actual mesh generation mesh, how it is achieved and where extra attention is required. In Chapter3.3 it will be illustrated in what ways users can interact with the newly created objects, either using mouse and keyboard or a virtual reality headset. In Chapter 4 the developed programs will be presented, evaluated and compared to similar software to see where its strengths and weaknesses are. Lastly in Chapter 5 the reached goals and some possible further projects and improvements will be discussed.

1.4 Related Works

2 Unreal Engine

The Unreal Engine is a 3D graphics video game engine, first created for the first person shooter Unreal in 1998 [**?**].

2.1 Unreal Engine Basics

2.1.1 Actors and Components

2.1.2 Pawns and Player Controllers

text

2.2 C++ and Blueprints

As already described in 2.1 ...

2.3 Unreal Engine Networking

text

3 Dynamic 3D Object Insertion

In diesem Kapitel soll das eigene Verfahren beschrieben werden. Es geht dabei nicht nur darum zu beschreiben was gemacht wurde, sondern ebenfalls darum zu begründen, weshalb bestimmte Design-Entscheidungen getroffen wurden. Das Kapitel sollte nicht „Eigenes Verfahren“ heißen, sondern etwas mit dem Titel der Arbeit zu tun haben.

3.1 File Formats and Parsing

3.1.1 File Sharing

3.2 Runtime Mesh Generation

3.2.1 Runtime Mesh Component

3.2.2 Material Generation

3.3 Object Interaction

3.3.1 Mouse and Keyboard

3.3.2 Virtual Reality

3.4 Plug-in Installation and Usage

4 Results and Evaluation

In diesem Kapitel sollen die Ergebnisse dieser Diplomarbeit diskutiert werden.

4.1 CAD Runtime Loader

4.2 Comparison to related works

4.3 Shortcomings and possible Improvements

5 Conclusion

In diesem Kapitel sollen zunächst die erreichten Ziele diskutiert und abschließend ein Ausblick auf mögliche, weiterführende Arbeiten gegeben werden.

Bibliography

[CEI01] Carl Erikson D. M., III W. V. B.: Hlods for faster display of large static and dynamic environments. *University of North Carolina at Chapel Hill* (2001).

[Cha08] Charpentier F.: *Nvidia Cuda: Das Ende der CPU?* Technical report, Tom's Hardware, Jun 2008. http://www.tomshardware.com/de/CUDA-Nvidia-CPU-GPU,testberichte-240065.html (20.08.2009).

[Cla76] Clark J. H.: Hierarchical geometric models for visible surface algorithms. *Communications of the ACM 19*, 10 (Oct 1976), 547–554. http://design.osu.edu/carlson/history/PDFs/clark-vis-surface.pdf (09.09.2009).

[DFMP98] De Floriani L., Magillo P., Puppo E.: Efficient implementation of multi-triangulations. In *VIS '98: Proceedings of the conference on Visualization '98* (Los Alamitos, CA, USA, 1998), IEEE Computer Society Press, pp. 43–50.

[Eck99] Eckert M.: *Von-Neuman Architektur.* Technical report, TecChannel, sep 1999. http://www.tecchannel.de/server/prozessoren/401364/so_funktioniert_ein_prozessor (19.08.2009).

[Eis06] Eiserle M.: Progressive techniken in der computergrafik. *Universität Rostock* (2006). http://vcg.informatik.uni-rostock.de/assets/publications/theses_sem/SA_Eiserle2006.pdf (15.09.2009).

[ESV99] El-Sana J., Varshney A.: Generalized view-dependent simplification. *Computer Graphics Forum 18*, 3 (1999), 83–94.

[Fos95] Foster I.: *Designing and Building Parallel Programs.* Addison Wesley Pub Co Inc, Reading, MA, USA, 1995.

[Har08] Harris M.: *Parallel Prefix Sum (Scan) with CUDA.* Programming guide, NVIDIA, 2008.

[Hop96] Hoppe H.: Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM, pp. 99–108.

[Hop97] Hoppe H.: View-dependent refinement of progressive meshes. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer*

graphics and interactive techniques (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 189–198.

[Hop98] Hoppe H.: Efficient implementation of progressive meshes. *Computers & Graphics 22*, 1 (1998), 27–36.

[HSH09] Hu L., Sander P. V., Hoppe H.: Parallel view-dependent refinement of progressive meshes. In *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2009), ACM, pp. 169–176.

[JNS08] John Nickölls Ian Buck M. G., Skadron K.: *Scalable Parallel Programming*. Programming guide, UNIVERSITY OF VIRGINIA, 2008.

[Lit08] Littschwager T.: *Neue Grafik-Generation: Alle Details*. Technical report, Chip, Sep 2008. http://www.chip.de/artikel/Neue-Grafik-Generation-Alle-Details_32708718.html (19.08.2009).

[Lue01] Luebke D. P.: A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications 21*, 3 (May/Jun 2001), 24–35. http://www.cs.virginia.edu/~luebke/publications/pdf/cg+a.2001.pdf (11.09.2009).

[MB00] Mrohs Bernd C. R.: Progressive meshes - eine einführung. *test* (Jul 2000). http://www.mrohs.com/publications/Bernd%20Mrohs,%20Christian%20Raeck%20-%20Progressive%20Meshes.pdf(29.07.09).

[MGK03] Michael Guthe P. B., Klein R.: Efficient view-dependent out-of-core visualization. *University of Bonn, Institute of Computer Science II* (2003). http://www.uni-marburg.de/fb12/informatik/homepages/guthe/files/guthe-2003-efficient (12.09.2009).

[Nah02] Nahmias J.-D.: Real-time massive model rendering. *University College London* (Sep 2002). http://www.cs.ucl.ac.uk/research/equator/papers/Documents2002/Jean-Daniel_Nahmias/Massive_Model_Rendering.htm (29.07.09).

[NVI07] NVIDIA: *NVIDIA CUDA Compute Unified Device Architecture*. Programming Guide Version 1.0, NVIDIA Corporation, Santa Clara, CA, USA, 2007.

[NVI08] NVIDIA: *NVIDIA GeForce GTX 295*, 2008. http://www.nvidia.de/object/product_geforce_gtx_295_de.html (02.10.2009).

[NVI09] NVIDIA: *OpenCL Programming Guide for the CUDA Architecture*. Programming Guide Version 2.3, NVIDIA Corporation, Santa Clara, CA, USA, 2009.

[OLG*05] Owens J. D., Luebke D., Govindaraju N., Harris M., Krüger J., Lefohn A. E., Purcell T. J.: A survey of general-purpose computation on graphics hardware. In *Eurographics 2005, State of the Art Reports* (Aug 2005), pp. 21–51.

[Paj01] Pajarola R.: Fastmesh: Efficient view-dependent meshing. *Computer Graphics and Applications, Pacific Conference on 0* (2001), 0022.

[PD04] Pajarola R., DeCoro C.: Efficient implementation of real-time view-dependent multiresolution meshing. *IEEE Transactions on Visualization and Computer Graphics 10*, 3 (2004), 353–368.

[Rißß99] Risska V.: *Test: Intel Core i7 920, 940 und 965 Extreme Edition.* Technical report, Computerbase, Sep 1999. http://www.computerbase.de/artikel/hardware/prozessoren/2008/test_intel_core_i7_920_940_965_extreme_edition/ (20.08.2009).

[Tro01] Troger C.: Levels of detail. *Institute of Computer Graphics and Algorithms Vienna University of Technology* (2001). http://www.cg.tuwien.ac.at/courses/Seminar/SS2001/lod/troger_paper.pdf (09.09.2009).

[WIK] WIKIPEDIA: *Octree.* http://de.wikipedia.org/wiki/Octree (29.07.2009).

List of Abbreviations

ALU Arithmetic Logic Unit
BTF Bidirektionalen Textur Funktion
CPU Central Processing Unit
CU Control Unit
CUDA Compute Unified Device Architecture
FLOPs Floating Point Operations Per Second
FPU Floating Point Unit
GPGPU General Purpose Compution on Graphics Processing Unit
GPU Graphics Processing Unit
HLOD Hierarchische Level of Detail
IFS Indexed-Face-Set
LOD Level of Detail
MIMD Multiple Instruction Multiple Data
OpenCL Open Computing Language
OpenGL Open Graphics Library
PCAM Partitionierung Kommunikation Agglomeration Mapping
PM Progressive Meshes
SFU Spezial Funktion Units
SIMD Single Instruction Multiple Data
SIMT Single Instruction Multiple Threads
SLI Scalable Link Interface
SP Streaming-Prozessoren
SM Streaming-Multiprozessoren
TPC Textur Prozessor Clustern
VBO Vertex Buffer Object

List of Figures

List of Tables

List of Algorithms

Listings