Mai 2011

UNIVERSITÄT
KOBLENZ · LANDAU
Fachbereich 4: Informatik

Konzeption und Implementierung eines Unified Rendering Frameworks mit modernen GPU-Computing-APIs

Bachelorarbeit

zur Erlangung des Grades eines Bachelor of Science (B.Sc.)
im Studiengang Computervisualistik

vorgelegt von
Markus Schlüter

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	Dipl.-Inform. Dominik Grüntjens

Koblenz, im Mai 2011

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

. . . . . .
(Ort, Datum) (Unterschrift)

Inhaltsverzeichnis

1 Einleitung

Im Rahmen dieser Bachelorarbeit wurde der Frage nachgegangen, inwiefern eine sogenannte Ünified Rendering-Engine, welche verschiedene Simulationsdomänen vereint, einen Mehrwert darstellen kann gegenÃ¼ber dem klassischen Ansatz, z.B. eine Graphik- und eine Physik-Engine zu verwenden, die zunächst mal keinen Bezug zueinander haben;
Hierbei wurde besonderer Wert auf die Verwendung moderner GPU-Computing-APIs gelegt, namentlich auf OpenGL3/4 und OpenCL.
Ziel war vor allem das Potential und die Erweiterbarkeit des Frameworks, nicht die schnelle Realisierung eines grossen Feature-Sets.

1.1 Motivation

Ursprünglich als Arbeit zur Implementierung von Fluidsimulation geplant, wurde bald ein generalistischer, eher softwaretechnisch orientierter Ansatz verfolgt, der die Fluidsimulation jedoch als Endziel hatte; Der Wunsch nach einer Únified Rendering Engineérwächst aus eigener Erfahrung der Kopplung von Physik- und Graphik-Engines, was einen gewissen Overhead mit sich bringt.... weitere Gründe: Liebe zu High-Performacne-Hardware, Wissensdurst in Bezug auf Rendering, Physik-Simulation, Hardware-Interna und Engine-Design, frei strampelnvon sehr veralteten Technologien etc pp..

2 Systemarchitektur

Bla bla

2.1 Begrifflichkeiten

erklaren, was ich unter Unified Rendering verstehe, was Rendering dadruch fuer eine generalistische Bedeutung bekommt;

2.2 Paradigmen

moeglichst symmetrischer Ansatz zwischen den Simulationsdomaenen, zantralistische Verwaltung von spezifischen Objekten, GPU-Code gernaierung per Template-Engine, moeglichst versuchen, langfristig so viele Aspekte wie moeglich miteinander kombinieren zu koennern (Visualisierungstechniken, verschiedene Physik Partikelsimulation, Rigid bodiy-Simulation)

2.3 KlassenDiagramm

2.4 Dependencies

2.4.1 OpenGL3/4

2.4.2 OpenCL 1.0

2.4.3 GLFW

explizite GL3 core profile creation, einfaches fullscreen, multisampling, mouse grab, alles viel besser als GLUT :)

2.4.4 Grantlee

die string template engine die CL und GL code erzeugt

2.4.5 ogl math

leichte, aber doch recht maechtige mathe-bibliothek

2.5 Die Buffer-Abstraktion

die bombe, die cpu, ogl und ocl vereint, inclusive ping ponging etc.. fundamentale Klassensammlung fuer den Unified-Aspekt

2.6 Das WorldObject

Basis-Klasse fuer alles was unified simuliert wird: pure viuelle objekt, uniform grid, fluid, rigid body etc.. erwaehnung des SubObjects;

2.7 Materials

was stellt welches material in welcher Domain dar?

2.8 Geometry

Abtract, Buffer bused, Vertex based etc.. ein paar konzepte (implementier/genutzt nut VertexBased)

2.9 Massively Parallel Program

Basisklasse von Shader und OpenCL Program

2.9.1 Shader

2.9.2 OpenCLProgram

weitere klassen/konzepte to go...

3 Simulation

3.1 Die visuelle Domäne

Ein paar worte ueber die shading features, wie sie maskiert werden

3.2 Der LightingSimulator

Nochmal drauf hinweisen, dass Rendering etwas generisches in diesem Framework ist, und wir leiber von Lichtsimulation sprechen sollten, auch wenn es monetan nicht photrealistisch ist ;)

3.3 ShaderManager

generiert mit grantlee, assigned an materials und verwaltet Shader , abhaenging von der aktuellen lighting stage, den registierten Materials, der Erzeugten kontext, den vom user aktivierten rendering features etc pp

3.4 Instancing

InstanManager, InstangedGeometry vorstellen, konzept, wie es verwaltet wird, erklaeren

3.5 Uniform Buffers

auch von BufferInterface abstrahiert, vorteile auflisten, aber auch stolperfallen 8alignment etc)

3.6 Tesselation

basics des hardware features erwaehnen fuer den geneigten leser, raptormodell erwaehnen und seinen Aufbereitungsprozess, LOD, displacement mapping erlaeutern

3.7 Die mechanische Domäne

Bla bla

3.8 Motivation

blubb

4 Ergebnisse

Bla bla

4.1 Motivation

blubb

5 Ausblick

Bla bla

5.1 Motivation

blubb

6 Fazit