

Mai 2011

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.	<input type="checkbox"/>	<input type="checkbox"/>
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	<input type="checkbox"/>	<input type="checkbox"/>

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

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
2	Systemarchitektur	2
2.0.1	Begrifflichkeiten	2
2.1	Paradigmen	2
2.2	KlassenDiagramm	2
2.3	Dependencies	2
2.3.1	OpenGL3/4	2
2.3.2	OpenCL 1.0	2
2.3.3	GLFW	2
2.3.4	Grandlee	2
2.3.5	ogl math	2
3	Simulation	3
3.1	Die visuelle Domäne	3
3.2	Motivation	3
3.3	Die mechanische Domäne	4
3.4	Motivation	4
4	Ergebnisse	5
4.1	Motivation	5
5	Ausblick	6
5.1	Motivation	6
6	Fazit	7

1 Einleitung

Im Rahmen dieser Bachelorarbeit wurde der Frage nachgegangen, inwiefern eine sogenannte Unified Rendering-Engine, welche verschiedene Simulationsdomänen vereint, einen Mehrwert darstellen kann gegen $\sim\frac{1}{4}$ ber 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 Unified Rendering Engine erwächst 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-Internas und Engine-Design, frei strampeln von sehr veralteten Technologien etc pp..

2 Systemarchitektur

Bla bla

2.0.1 Begrifflichkeiten

erklären, was ich unter Unified Rendering verstehe, was Rendering dadurch fuer eine generalistische Bedeutung bekommt;

2.1 Paradigmen

moeglichst symmetrischer Ansatz zwischen den Simulationsdomaenen, zentralistische 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 body-Simulation)

2.2 KlassenDiagramm

2.3 Dependencies

2.3.1 OpenGL3/4

2.3.2 OpenCL 1.0

2.3.3 GLFW

expliziter GL3 core profile creation, einfaches fullscreen, multisampling, mousgrab, alles viel besser als GLUT :)

2.3.4 Grantlee

die string template engine die CL und GL code erzeugt

2.3.5 ogl math

leichte, aber doch recht maechtige mathe-bibliothek

2.4 Die Buffer-Abstraktion

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

2.5 Das WorkObject

basiklasse fuer alles was unified simuliert wird: pure visuelle objekt, uniform grid, fluid, rigid body etc.. erwaechnung des SubObjects;

2.6 Materials

was stellt welches material in welcher Domain dar?

2.7 Geometry

Abstract, Buffer based, Vertex based etc.. ein paar konzepte (implementier/genutzt mit VertexBased)

3 Simulation

3.1 Die visuelle Domäne

Bla bla

3.2 Motivation

blubb

3.3 Die mechanische Domäne

Bla bla

3.4 Motivation

blubb

4 Ergebnisse

Bla bla

4.1 Motivation

blubb

5 Ausblick

Bla bla

5.1 Motivation

blubb

6 Fazit