Philipps Universität Marburg

FACHBEREICH MATHEMATIK UND INFORMATIK

Arbeitsgruppe Grafik und Multimedia Programmierung

Entwicklung eines interaktiven Editors für Flugzeugkonfigurationen im Vorentwurf

Abschlussarbeit zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

vorgelegt von

René Frank

Betreuer:	Prof. Dr. Thorsten Thormählen
Betreuer:	Dipl.-Ing. Carsten Liersch
Ausgabedatum:	XX.XX.2014
Abgabedatum:	XX.XX.2015

Philipps-Universität Marburg
Fachbereich Mathematik und Informatik
Hans-Meerwein-Straße
35032 Marburg

Erklärung

Ich erkläre hiermit, dass ich diese Masterarbeit mit dem Titel *Entwicklung eines interaktiven Editors für Flugzeugkonfigurationen im Vorentwurf* selbstständig ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Alle den benutzten Quellen wörtlich oder sinngemäß entnommenen Stellen sind als solche einzeln kenntlich gemacht.

Diese Arbeit ist bislang keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht worden.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Marburg den 11. Oktober 2014

René Frank

Zusammenfassung

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 texttext text (exakte englische "Übersetzung der deutschen Kurzfassung)

Inhaltsverzeichnis

Kapitel 1.
1 Einleitung

Diese Diplomarbeit beschäftigt sich mit den Parallel View-Dependent Compressed Progressive Meshes und deren Umsetzung in die vom Grafikkartenhersteller NVIDIA entwickelte parallele Programmiersprache CUDA. Dazu gehört die Entwicklung einer für die parallele Verarbeitung geeignete effiziente Datenstruktur, sowie eine effiziente Datenverwaltung.

1.1. Motivation

Die aktuelle Entwicklung der Multimediaindustrie versucht zunehmend die Simulation von virtuellen Welten realistisch darzustellen. Die Ansprüche der Anwender werden mit der Zeit immer größer und dementsprechend die generierte virtuelle Realität immer komplexer. So eine Entwicklung ist unweigerlich mit der Steigerung der erforderlichen Rechenleistung verbunden, da die simulierten Objekte aus Millionen von Polygonen bestehen können und in Echtzeit dargestellt werden müssen. Im Laufe der Jahre sind viele verschiedene Verfahren entwickelt worden, die das Ziel hatten, die komplexen Objekte mit einem vertretbaren Qualitätsverlust in Echtzeit darzustellen. Der mit Abstand beste Ansatz, um den Kompromiss zwischen Qualität und Geschwindigkeit zu finden, ist die View-Dependent Progressive Meshes. Einer der Vorteile dieser Herangehensweise ist, dass dieses Verfahren hochgradig parallelisierbar ist, so dass sich mit einer geeigneter Programmiersprache und Hardware eine beachtliche Effizienzsteigerung erzielen lässt.
Die von NVIDIA entwickelte parallele Programmiersprache CUDA setzt auf den aktuellen Trend der GPGPUs und ermöglicht es mit einer kostengünstigen Grafikkarte, die in fast jeden Desktoprechner vorhanden ist, Programme effizient zu parallelisieren. Aus diesem Grund ist CUDA für das Parallelisieren von View-Dependent Progressive Meshing besonders geeignet.

1.2. Ziele

Das Ziel dieser Arbeit ist die Entwicklung einer effizienten parallelen Implementierung von komprimierten View-Dependent Progressive Meshes in CUDA, welche in der Lage ist, Objekte die aus mehreren Millionen von Polygonen bestehen können, in Echtzeit zu verarbeiten.

Echtzeit

Das entwickelte Programm soll selbst sehr große Polygonnetze effizient verarbeiten können. Die Eingaben des Benutzers für die Translation und Rotation des Objekts sollen in Echtzeit umgesetzt werden. Die durchschnittliche Laufzeit des Programms pro Frame soll höchstens drei Mal soviel Zeit wie das Rendering des gegebenen Modells benötigen, um eine Echtzeitdarstellung des Modells zu ermöglichen. Dabei können die Modelle aus mehreren Millionen von Dreiecken bestehen.

Kosten

Das Programm sollte mit der normalen, kostengünstigen Privatanwender-Hardware laufen, sodass für die Ausführung keine Spezialrechner benötigt werden. Die einzige Vorrausetzung an das System ist eine NVIDIA-Grafikkarte die CUDA 1.1 unterstützt. Diese ist aber in den meisten Desktoprechnern vorhanden oder kann kostengünstig nachgerüstet werden.

1.3. Aufbau der Arbeit

Im ersten Abschnitt des Kapitels **??** soll zunächst die Bedeutung der Grafikkarte als Berechnungseinheit verdeutlicht werden. Dann soll im zweiten Abschnitt die Hard- und Softwarearchitektur der Programmiersprache CUDA beschrieben werden, sowie einige Vorschläge zu Codeoptimierung diskutiert, bevor im Kapitel **??** ein Überblick über die wichtigsten Verfahren zur Echtzeitdarstellung komplexer Objekte geben wird. An dieser Stelle werden auch das View-Dependent Progressive Meshing, sowie einige Simplifizierungstechniken genauer erläutern. Kapitel **??** beschäftigt sich mit der Theorie des im Rahmen dieser Diplomarbeit entwickelten Algorithmus. Dabei sollen die Datenstrukturen, die Kompression, sowie die einzelnen Schritte des Algorithmus genauer erläutert werden. Die Implementierung des Algorithmus in CUDA wird im Kapitel 5 besprochen, dabei sollen die benutzten Bibliotheken, sowie die CUDA-spezifische Umsetzung des Programms beschrieben werden. Anschließend werden im Kapitel 6 die durchgeführten Tests und deren Ergebnisse dokumentiert und diskutiert, sowie im Kapitel 7 ein Ausblick auf weiterführende Arbeiten gegeben.

1.4. Verwandte Arbeiten

Im Themenbereich der Progressive Meshes und View-Dependent Progressive Meshes gab es schon am Ende des letzten Jahrzehnts einige Veröffentlichungen [**?**, **?**]. Diese waren zwar eine gute und notwendige Weiterentwicklung vom klassischen LOD-Algorithmus, ermöglichten aber nicht eine effiziente Echtzeitdarstellung von großen Modellen. In [**?**] wurde schließlich ein Versuch unternommen eine effizientere Datenstrucktur zu entwickeln, um den Speicherverbrauch zu optimieren und bessere Geschwindigkeit zu erreichen. Diese effizientere Datenstruktur brachte zwar einige Verbesserungen, ermöglichte aber dennoch keine Echtzeitdarstellung von großen Modellen. Seit dem gab es eine Reihe von Verfahren, die das Ziel hatten eine effiziente Echtzeitdarstellung von großen Modellen zu ermöglichen. Einige von diesen Verfahren nutzten Multi-Triangulationen [**?**], andere Versuchten die View-Dependent Progressive Meshes weiterzuentwickeln [**?**, **?**, **?**]. Doch

keins dieser Verfahren konnte die Anforderungen vollständig erfüllen.
Eine erst kürzlich veröffentlichte Arbeit [?] machte endlich einen Schritt in die richtige Richtung. Die in dieser Arbeit implementierte GPU-Variante von parallelen View-Dependent Progressive Meshes ermöglichte eine akzeptable Echtzeitdarstellung von großen Modellen. Diese braucht durchschnittlich das dreifache der Zeit, die für das Rendering des Modells benötigen wird und lässt somit einen großen Spielraum für die Optimierung offen.

Kapitel 2.
2 Grundlagen

Im Folgenden soll ein Überblick Ã¼ber die in dieser Arbeit verwendeten Technologien gegeben werden. Eine allgemeine Einführung in den Flugzeugentwurfsprozess zeigt anfangs die Einsatzgebiete des SGG-Editors auf. Weiterhin wird auf das zentrale Datenformat CPACS, auf dem der SGG arbeitet, eingegangen und dessen Aufbau erläutert. Im zweiten Teil werden die dargestellten Flugzeugkomponenten und verwendete Generierungsverfahren vorgestellt.

2.1. Flugzeugentwurf

hier steht alles zum Flugzeugentwurfsprozess

2.2. CPACS

Wie schon in Kapitel 2.2 beschrieben ... steht hier alles zu CPACS

2.3. Profile

Die Form des Querschnitts eines Körpers, wird im Folgenden als Profil bezeichnet. In der Aerodynamik ist die Entwicklung und Charakterisierung von Profilen ein wichtiges Teilgebiet. Konstruierte Profile sollen in ihrer Form bestimmten Funktionen genügen wie beispielsweise die Erzeugung eines dynamischem Auftriebs bei geringem StrÃ¶mungswiderstand. In Cpacs wird zwischen Rumpf- und Tragflächenprofilen unterschieden. Beide Profiltypen sind unter dem Konten *profiles* als Listen für x, y und z Koordinaten repräsentiert.

ooo hier steht ein tikz xml editor

2.3.1. Flügelprofile

hier steht allgemeines Zeug zu den Profilen

[1]Quelle: http://www.texample.net/tikz/examples/airfoil-profiles, Zugriff: 27.10.2014

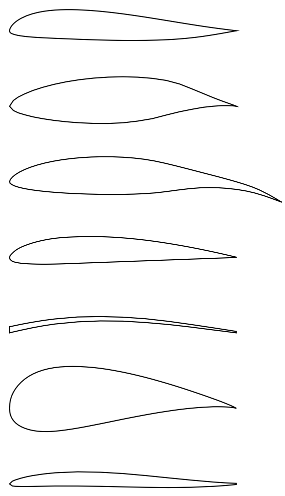


Abbildung 2.1.: Tragflächenprofile[1]

NACA-Serie

Das National Advisory Committee for Aeronautics oder kurz NACA wurde 1915 gegrÃ¼ndet und ist ein direkter Vorgänger der US-BundesbehÃ¶rde fÃ¼r Luft- und Raumfahrt, NASA. Die NACA war eine amerikanische Organisation, die sich mit der Grundlagenforschung in der Luftfahrt beschÃ¤ftigte. Eine bedeutende Entwicklung der NACA-Forschungen, sind optimierte Tragfächenprofile. Durch aerodynamischen Tests im Windkanal wurde bereits früh erkannt, dass die Flügelprofile mit den besten Eigenschaften hinsichtlich Auftriebsbeiwert und Widerstandsbeiwert, viele Gemeinsamkeiten besitzen. NACA-Profile sind somit Variationen eines Ursprungsprofils, die mit Hilfe von analytischen Gleichungen definiert werden. Spezifische Variationen dieses Profils werden durch die Krümmung bzw. Steigung der Skelettlinie sowie die Dicke der Tragfläche oberhalb und unterhalb jener Skelettlinie erzeugt. Im SGG-Editor wurde ein NACA-Generator implementiert, mit dem sich Profile der NACA 4-digit und NACA 5-digit Serie erstellen lassen.

Profile der Vierer-Serie In der vierstelligen NACA-Serie ist ein Profil definiert durch:

1. Ziffer: maximale Profilwölbung
   - angegeben in Prozent, bezogen auf die Länge der Profilsehne
2. Ziffer: WölbungsrÃ¼cklage, Position der maximalen Profilwölbung
   - angegeben in Zehnteln der Länge der Profilsehne

3./4. Ziffer: maximale Profildicke
   - angegeben in Prozent, bezogen auf die Länge der Profilsehne

Ein symmetrisches NACA 4 Profil kann mit Gleichung 2.1 konstruiert werden. Das Profil ist in seiner Form, nur durch die angegebene Profildicke veränderbar, da die

Profilwölbung und somit auch dessen Position die Werte Null haben. Gleichung 2.1 enthält Konstanten, die für eine Profildicke von 20% vorgesehen sind. Um diese Werte an die jeweils angegebene Profildicke anzupassen, wird die eigentliche Berechung mit $\frac{t}{0.2}$ multipliziert. In Gleichung 2.1 werden zusätzlich folgende Parameter verwendet:

c : Länge der Profilsehne

x : Position entlang der Profilsehne auf der Abszissenachse von 0 to c,

y_t : Entfernung der Skelettlinie zur jeweiligen Profilseite an Position x

t : Maximale Dicke des Profils multipliziert mit $\frac{1}{100}$

$$y_t = \frac{t}{0.2}c\left[0.2969\sqrt{\frac{x}{c}} + (-0.1260)\left(\frac{x}{c}\right) + (-0.3516)\left(\frac{x}{c}\right)^2 + 0.2842\left(\frac{x}{c}\right)^3 + (-1.015)\left(\frac{x}{c}\right)^4\right] \quad (2.1)$$

Soll die trailing edge geschlossen sein, also das Profil an dieser Position eine Dicke gleich Null haben, wird als Koeffizient an der letzen Stelle statt -1.015 ein Wert von -0.1036 gewählt. Es ergeben sich folgende Definitionen für Ober- und Unterseite des Profils. Die x-Koordinaten sind für beide Seiten gleich, daher gilt $x_U = x_L = x$. Die y-Koordinaten ebenfalls identisch, nur das diese für die Oberseite positiv: $y_U = +y_t$ und für die Unterseite negativ sind: $y_L = -y_t$.
Die Generierung eines asymmetrischen NACA 4 Profils braucht zusätzlich zu Gleichung 2.1 noch die maximale Profilwölbung und die Wölbungsrücklage, also den Abstand der Profilnase zur maximalen Profilwölbung.

m : Maximale Wölbung multipliziert mit $\frac{1}{100}$

p : Position der maximalen Wölbung multipliziert mit $\frac{1}{10}$

t : Maximale Dicke des Profils multipliziert mit $\frac{1}{100}$

Mit Gleichung 2.5 wird die y-Koordinate der Skelettlinie an einer gegebenen x-Koordinate berechnet.

$$y_c = \begin{cases} m\frac{x}{p^2}\left(2p - \frac{x}{c}\right), & 0 \leq x \leq pc \\ m\frac{c-x}{(1-p)^2}\left(1 + \frac{x}{c} - 2p\right), & pc \leq x \leq c \end{cases} \quad (2.2)$$

Die Dicke des gekrümmten Flügelprofils ist senkrecht zur Skelettlinie festgelegt womit für Ober- und Unterseite des Profils folgendes gilt:

$$x_U = x - y_t \sin\theta \quad , \quad y_U = y_c + y_t \cos\theta \tag{2.3}$$

$$x_L = x + y_t \sin\theta \quad , \quad y_L = y_c + -y_t \cos\theta \tag{2.4}$$

$\theta = \arctan\left(\frac{dy_c}{dx}\right)$

$$\frac{dy_c}{dx} = \begin{cases} \frac{2m}{p^2}\left(p - \frac{x}{c}\right), & 0 \le x \le pc \\ \frac{2m}{(1-p)^2}\left(p - \frac{x}{c}\right), & pc \le x \le c \end{cases} \tag{2.5}$$

text [**?**]

Naca5

Um den maximalen Auftrieb von Tragflächenprofilen zu erhöhen, wurde zusätzlich die 5er Naca Serie entwickelt. Ein NACA 5 Profil hat die Form LPQXX (beispielsweise NACA 23009) und wird wie im Folgenden definiert. Hierbei ist zu beachten, dass die ersten beiden Ziffern zur späteren Berechnung umgrechnet werden.

1. Ziffer: Wert zur Berechnung des optimalen Auftriebskoeffizienten bei optimalem Anstellwinkel
   - $cl = L * 0.15$
2. Ziffer: Position der größten Wölbung entlang der Sehnenlinie, beginnend bei der leading edge
   - $p = P * 5$
3. Ziffer: einfache oder gespiegelte Krümmung
   - 0 oder 1
4. Ziffer und 5. Ziffer: maximale Dicke in % zur Sehnenlänge
   - $t = XX$

Für das obige NACA 23009 würde dies folgendes bedeuten:

Beispiel 2.1 *NACA 23009*

*L = 2 → 2 * 0.15 → Auftriebskoeffizient = 0.3*

*P = 3 → 3 * 5.0 → Position bei = 15%*

Q = 0 → normale Wölbung

XX = 09 → Dicke = 9 %

Die Konstruktion eines NACA 5 Profils sieht zwei Fälle vor. Die ersten beiden Gleichungen beschreiben werden verwendet, wenn das Q gleich 0 ist, also ein Profil mit normaler Wölbung kontruiert werden soll. Die letzten bewirken im Fall, dass Q gleich 1 ist, eine gespiegelte Wölbung.

Wölbung (normal)

$$y_c = \begin{cases} \frac{k_1}{6}(x^3 - 3mx^2 + m^2(3-m)x), & 0 \leq x \leq m \\ \\ \frac{k_1 m^3}{6}(1-x), & m \leq x \leq 1 \end{cases} \tag{2.6}$$

Anstieg (normal)

$$\frac{dy_c}{dx} = \begin{cases} \frac{k_1}{6}(3x^2 - 6mx + m^2(3-m)), & 0 \leq x \leq m \\ \\ -\frac{k_1 m^3}{6}, & m \leq x \leq 1 \end{cases} \tag{2.7}$$

Wölbung (gespiegelt)

$$y_c = \begin{cases} \frac{k_1}{6}\left((x-m)^3 - \frac{k_2}{k_1}(1-m)^3 x - m^3 x + m^3\right), & 0 \leq x \leq m \\ \\ \frac{k_1}{6}\left(\frac{k_2}{k_1}(x-m)^3 - \frac{k_2}{k_1}(1-m)^3 x - m^3 x + m^3\right), & m \leq x \leq 1 \end{cases} \tag{2.8}$$

Anstieg (gespiegelt)

$$\frac{dy_c}{dx} = \begin{cases} \frac{k_1}{6}\left(3(x-m)^2 - \frac{k_2}{k_1}(1-m)^3 - m^3\right), & 0 \leq x \leq m \\ \\ \frac{k_1}{6}\left(3\frac{k_2}{k_1}(x-m)^2 - \frac{k_2}{k_1}(1-m)^3 - m^3\right), & m \leq x \leq 1 \end{cases} \tag{2.9}$$

In Tabelle **??** sind die Konstanten m, k1 und k1/k2 definiert. Diese wurden an der Position der maximale Wölbung bei einem Auftriebsbeiwert von 0.3 bestimmt. Die Ergebnisse für Anstieg und Wölbung können linear bezüglich des gewünschten Auftriebsbeitwertes skaliert werden.

Das Plotten geschieht mit cosinus

$$\frac{x_i}{c} = \frac{1}{2}\left[1 - \cos\left(\frac{i * \pi}{N-1}\right)\right] \tag{2.10}$$

Beschreibung	Position max Wölbung (p)	m	k1	k2/k1
5% normal	0.05	0.0580	361.400	
10% normal	0.10	0.1260	51.640	
15% normal	0.15	0.2025	15.957	
20% normal	0.20	0.2900	6.643	
25% normal	0.25	0.3910	3.230	
10% gespiegelt	0.10	0.1300	51.990	0.000764
15% gespiegelt	0.15	0.2170	15.793	0.00677
20% gespiegelt	0.20	0.3180	6.520	0.0303
25% gespiegelt	0.25	0.4410	3.191	0.1355

Tabelle 2.1.: NACA 5 Konstanten für Auftriebskoeffizient von 0.3

Skelettlinie

```
Data: bottom profile, top profile, trailing edge, leading edge
Result: camber line
chord = line from trailing edge to leading edge;
foreach p in chord do
    perp1 = determine perpendicular of chord through point p;
    foreach p_b in bottom profile do
        perp2 = determine perpendicular of perp1 through point p_b;
        determine intersection point of perp1 and perp2;
        determine distance from intersection point to p_b;
    end
    dist_b = minimum distance from intersection point to p_b;
    foreach p_t in top profile do
        perp2 = determine perpendicular of perp1 through point p_t;
        determine intersection point of perp1 and perp2;
        determine distance from intersection point to p_t;
    end
    dist_t = minimum distance from intersection point to p_t;
    get center of dist_t and dist_b
end
```

Algorithm 1: Berechnung der Skelettlinie

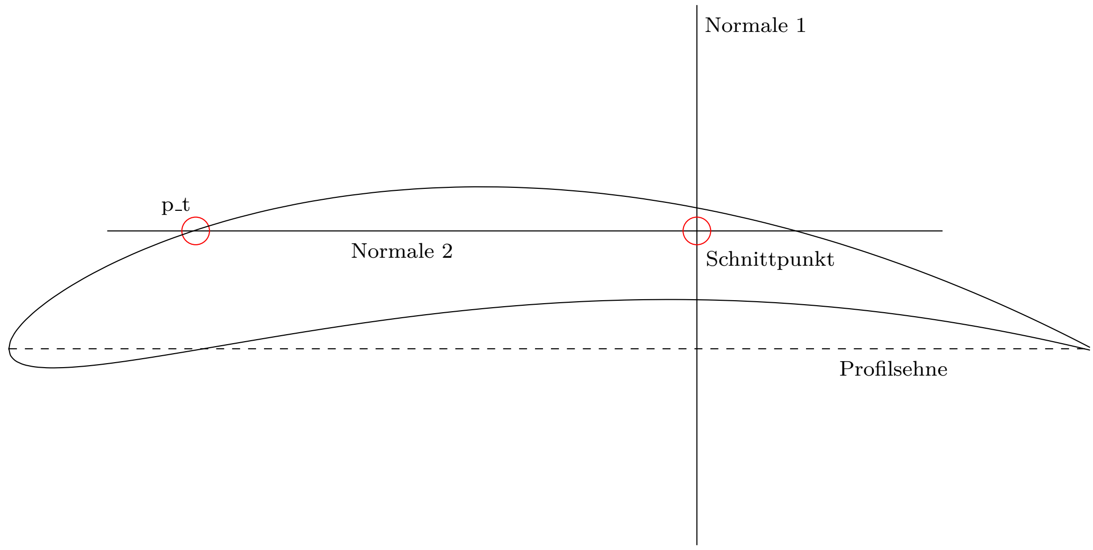


Abbildung 2.2.: Berechnung eines Punktes der Skelettlinie

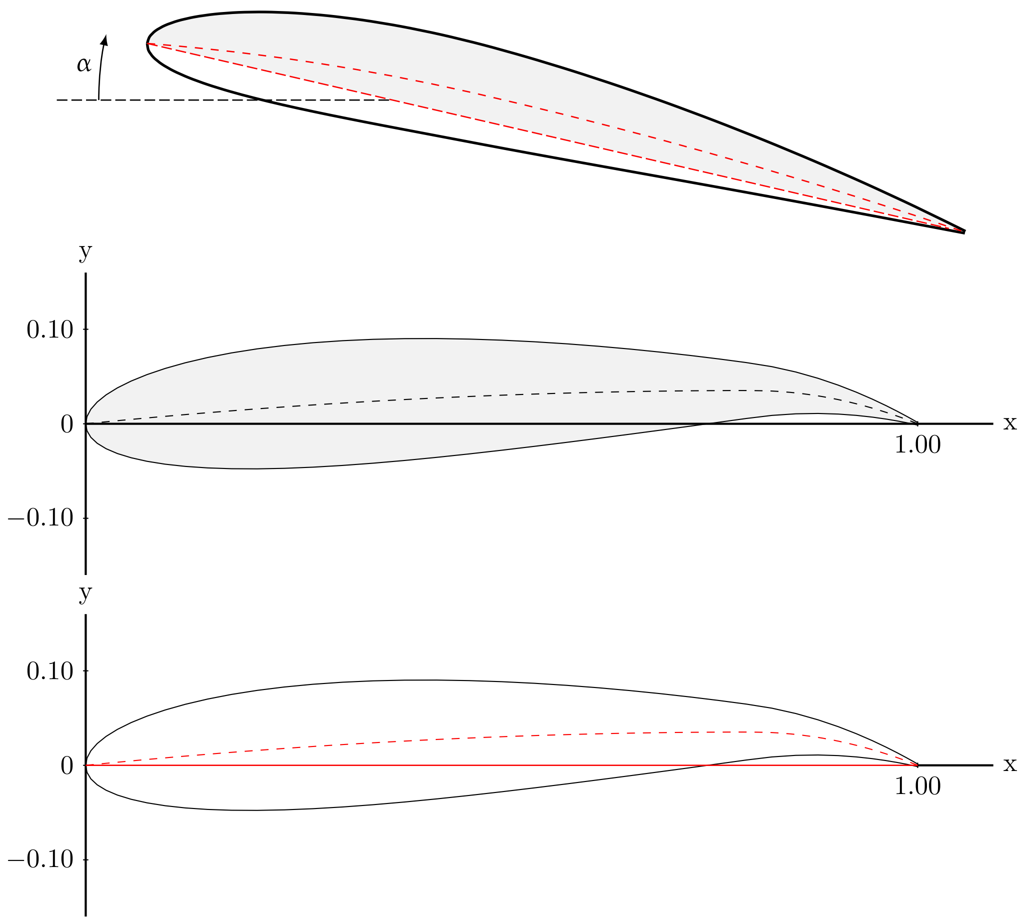


Abbildung 2.3.: Eine Vektorgrafik

Sonstiges

2.3.2. Rumpfprofile

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

3 Kapitel 3. Eigenes Verfahren

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.

3.1. LaTex-Editoren

Ein guter Cross-Plattform (Windows/Linux/Mac) Latex-Editor mit englischer und deutscher Rechtschreibkorrektur ist z.B. TexStudio (http://texstudio.sourceforge.net/). Unter Windows verwende ich diesen Editor gerne zusammen mit dem Sumatra PDF Viewer (http://blog.kowalczyk.info/software/sumatrapdf/free-pdf-reader.html), da dieser ein automatisches Neuladen unterstützt.

3.2. Beispiel für eine Tabelle

In Tabelle 3.1 sind die verwendeten Bibliotheken ausgelistet.

Bibliothek	**Version**
CUDA SDK	2.3
CUDA Toolkit	2.3
OpenGL	3.2
GLUT	3.7
GLEW	1.5.1
CUDPP	1.1

Tabelle 3.1.: Die verwendeten Bibliotheken.

3.3. Formel in Latex und Konventionen zur Verwendung mathematischer Symbole

Mathematische Symbole können in Latex sehr leicht erzeugt werden. Beispiel für Symbole im Text: Gegeben sei ein Skalar $a \in \mathbb{R}$. Tabelle 3.2 liste einige Konventionen zur Verwendung mathematischer Symbole. Abgesetzte Formel lassen sich ebenfalls leicht erzeugen:

$$f(x) = x^2 + 3 \tag{3.1}$$

Außerdem kann leicht auf abgesetzte Formel verwiesen werden (siehe Gleichung 3.1). Für mehrere ausgerichtete Formeln bietet sich die Umgebung `eqnarray` an:

$$f(x) = x^2 + 3 \tag{3.2}$$

$$g(\theta) = \cos(2\theta) = \cos^2\theta - \sin^2\theta \tag{3.3}$$

$$h(x) = \int_0^\infty \mathrm{e}^{-x}\,\mathrm{d}x \tag{3.4}$$

Typ	**Schriftart**	**Beispiele**
Variablen (Skalare)	kursiv	a, b, x, y
Funktionen	aufrecht	$\mathsf{f}, \mathsf{g}(x), \mathsf{max}(x)$
Vektoren	fett, Elemente zeilenweise	$\mathbf{a}, \mathbf{b} = \begin{pmatrix} x \\ y \end{pmatrix} = (x, y)^\top$
Matrizen	Schreibmaschine	$\mathtt{A}, \mathtt{B} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$
Mengen	kalligrafisch	$\mathcal{A}, \{a, b\} \in \mathcal{B}$
Zahlenbereiche	doppelt gestrichen	$\mathbb{N}, \mathbb{Z}, \mathbb{R}^2, \mathbb{R}^3$

Tabelle 3.2.: Konventionen zur Verwendung mathematischer Symbole

3.4. Beispiel für eine Vektorgrafik

Wenn möglich sollten immer Vektorgrafiken verwendet werden. Rastergrafiken sollten nur dann eingesetzt werden, wenn die Original-Quelle ebenfalls eine Rastergrafik ist. Ein Cross-Plattform Editor zur Erstellung von Vektorgrafiken ist z.B. Inkscape: http://inkscape.org/download/. Nach der Erstellung in Inkscape sollte die Grafik zum einen zur späteren Weiterverarbeitung als SVG gespeichert werden. Zum anderen zwecks Import in Latex als PDF. Abbildung 3.1 zeigt ein Beispiel.

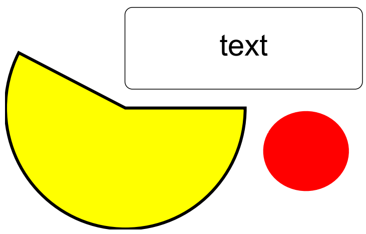


Abbildung 3.1.: Eine Vektorgrafik

3.5. Beispiel für eine Vektorgrafik mit mathematischen Symbolen

Um beliebigen Latex-Code in eine Vektorgrafik einzufügen (z.B. um mathematische Symbole zu setzen) kann in Inkscape beim Speichern der Datei als PDF die Option „Pdf+Latex: Text in PDF weglassen und Latex Datei erstellen“ angewählt werden (siehe Abb. 3.2)

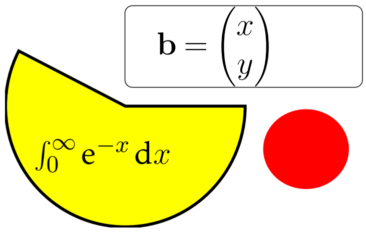


Abbildung 3.2.: Eine Vektorgrafik mit mathematischen Symbolen

3.6. Beispiel für eine Rastergrafik

Abbildung 3.3 zeigt, wie eine Rastergrafik eingebunden werden kann.

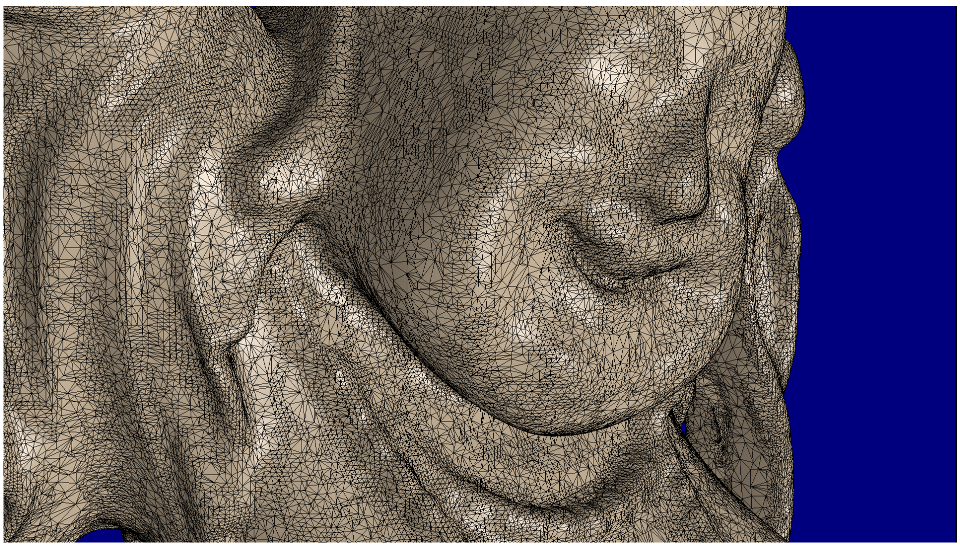


Abbildung 3.3.: Eine Rastergrafik

Bild- bzw. Tabellen-Beschriftungen sollten möglichst informativ sein. Aus der Beschreibung sollte die Bedeutung der Abbildung vollständig hervorgehen, so dass der Haupttext zum Verständnis nicht notwendigerweise gelesen werden muss.

3.7. Beispiel für die Darstellung von Algorithmen

Algorithmus 2 zeigt ...

Phase 1: Reduktion
`for` $d := 0$ `to` $log_2 n - 1$ `in parallel do`
  `for` $k := 0$ `to` $n - 1$ `by` 2^{d+1} `in parallel do`
    $x[k + 2^{d+1} - 1] := x[k + 2^d - 1] + x[k + 2^{d+1} - 1]$
Phase 2: Propagation
`for` $d := log_2 n$ `to` 0 `in parallel do`
  `for` $k := 0$ `to` $n - 1$ `by` 2^{d+1} `in parallel do`
    $t := x[k + 2^d - 1]$
    $x[k + 2^d - 1] := x[k + 2^{d+1} - 1]$
    $x[k + 2^{d+1} - 1] := t + x[k + 2^{d+1} - 1]$

Algorithm 2: Pseudocode der zwei Phasen vom SCAN-Algorithmus [**?**].

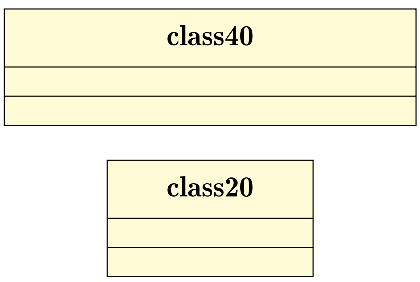
3.8. Beispiel für die Darstellung von Quellcode-Auszügen

Listing 3.1 zeigt ...

```
void Adapt() {
  cudaGLMapBufferObject( verbo);
  cudaGLMapBufferObject( ds , nidVui);
        dd
  cudaGLUnmapBufferObject(vertVbo);
  cudaGLUnmapBufferObject( indVbo);
}
```

Listing 3.1: Pseudocode für die Kontrolle der VBOs.

3.9. Hier ein uml diagramm



Kapitel 4.

4 Ergebnisse und Evaluation

In diesem Kapitel sollen die Ergebnisse dieser Diplomarbeit diskutiert werden.

Kapitel 5.

5 Zusammenfassung und Ausblick

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

Abkürzungsverzeichnis

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

Abbildungsverzeichnis

Tabellenverzeichnis

List of Algorithms

Listings

Anhang A. Anhang

Thema 1

Beispiel für einen Anhang

Thema 2

Danksagung

An erster Stelle möchte ich mich bei Herrn Prof. Dr. B. Freisleben für die Möglichkeit bedan- ken, diese Arbeit in seiner Arbeitsgruppe durchzuführen. Besonders möchte ich mich bei Pablo Graubner für die investierte Zeit und die gute Betreu- ung, sowohl auf menschlicher als auch auf fachlicher Ebene, die er mir zu teil werden ließ herzlich bedanken. Danken möchte ich auch meinen Freunden, die immer an meiner Seite stehen und mir bei der Arbeit mit Verständnis und guten Worten geholfen haben. Der größte Dank gilt allerdings meiner Familie, die mich in meiner Ausbildung sowohl in der Schule, als auch im Studium immer tatkräfig unterstützt hat und auf die ich mich immer verlassen kann. Ohne sie wäre diese Arbeit sicher nicht zustande gekommen.