



Grundlagen der Computergraphik (Graphik 1)

Holger Theisel

Daniel Stelter

Otto-von-Guericke-Universität
Magdeburg, SS 2024



Allgemein:

Prof. Holger Theisel
theisel@ovgu.de,
Büro: G29-217 , Tel: +49-391-6758773



Daniel Stelter
daniel.stelter@ovgu.de





Homepage:

http://vc.cs.ovgu.de/index.php?article_id=792&clang=1

<https://elearning.ovgu.de/course/view.php?id=16588>

Registrierung im e-learning Portal

- Übungen: Registerung e-learning Portal:
- Einschreibeschlüssel : raytracer

Im e-learning Portal:

- Aufzeichnung der Vorlesungen (vom SS 2021)
- Folien der Vorlesungen
- Übungsaufgaben, Programmieraufgaben (Abgabe der Übungsblätter erfolgt nur elektronisch über das Elearning)



Einordnung:

Studiengänge

Studiengang	Abschluss	Empfohlenes Semester	Kategorie	ECTS	Modul
<u>Computervisualistik (82152)</u>	Bachelor	2 - 3	PF		
<u>Digital Engineering (86158)</u>	Master	1 - 3	WPF		
<u>Informatik (82150)</u>	Bachelor	4 - 6	WPF		
<u>Ingenieurinformatik (82157)</u>	Bachelor	4 - 6	WPF		
<u>Medienbildung AVKK (82735)</u>	Bachelor	4 - 5	WPF	14	
<u>Sport+Technik (86720)</u>	Master	2 - 2	WPF		
<u>Wirtschaftsinformatik (82159)</u>	Bachelor	4 - 6	WPF		



Einordnung:

Pflicht Informatik Profilrichtung Computergames

Computer Games

Du willst mehr als nur spielen? Lernen, was sich im Inneren von Computerspielen verbirgt? Du willst wissen, welche Algorithmen hinter den tollen Graphiken stecken? Du willst selbst eigene Spiele entwickeln oder später in der Computerspieleindustrie arbeiten?

Mit der Profillinie „Computer Games“ innerhalb des Bachelorstudiengangs Informatik lernst Du, wie Spiele entwickelt werden. Zusätzlich kannst Du Dich beim an der Uni tätigen Verein „Acagamics e.V.“ mit Gleichgesinnten austauschen und mehr über Industrie und Forschung im Bereich Computer-Spiele erfahren.

Ergänzend zu den Pflichtveranstaltungen im Informatikstudium belegst Du folgende vier Lehrveranstaltungen: Introduction to Computer Games, Introduction to Computer Graphics, Introduction to Simulation, Algorithmische Geometrie. Dazu wählst Du weitere vier Vorlesungen aus einem größeren Pool von passenden Veranstaltungen. Zum Studium gehörende Seminare, Projekte und Praktika kannst Du ebenfalls zum Thema Computerspiele auswählen.

Profilmodule

Die Übersicht über die angebotenen Profilmodule kann hier abgerufen werden:

- › [Modulübersicht Profil "ComputerGames"](#).
- › Zur besseren Übersicht gibt es auch einen entsprechenden [Regelstudienplan](#).

Verantwortliche

Für das Studienprofil ist verantwortlich:

- › [Prof. Dr. Holger Theisel](#)



Allgemein:

Vorlesung Mo, 15:00 - 17:00 Uhr

Beginn 08.04.2024



Allgemein:

Übungen:

Gruppe 2 (Stelter) Mi, 15:00 -17:00 Uhr, G29-336

Gruppe 3 (Stelteri) Mi, 13:00 -15:00 Uhr, G29-K053

Beginn: 10.04.2024

- Besprechen der Übungsaufgaben



Abschluss:

- (1) 2/3 der möglichen Punkte für Übungsaufgaben
- (2) 2/3 der möglichen Punkte für Programmieraufgaben

- (3) schriftliche Prüfung (Klausur) am Semesterende

- (1) und (2) sind Voraussetzungen zur Zulassung für (3)

Keine Anwesenheitspflicht in Vorlesung oder Übung!
Schein: (3) mit mind. Note 4

Nicht-FIN Studiengänge: nur (1) und (3)



Übungsaufgaben:

- **Ausgabe:** jede Woche Mo ab 18 Uhr im eLearning
- **Abgabe der Lösungen:**
 - bis Mo 15 Uhr (vor der Vorlesung) im E-learning system

Besprechung: nächste Übung nach der Abgabe

- **Ausgabe der ersten Serie:** 08.04.2024



Programmieraufgaben:

Raytracer in C++,

Ausgabe: alle 2 -3 Wochen mit Übungsaufgaben

Abgabe der Lösungen:

- Mo 15 Uhr (vor der Vorlesung) im E-learning system

Besprechung: nächste Übung nach der Abgabe

- Ausgabe der ersten Serie: 22.04.2024.



Plan der nächsten Wochen:

- Mo, 08.04.: 1. Vorlesung
- Mo, 08.04. Ausgabe Übungsblatt 1
- Mi, 10.04.: 1. Übung: Einführung C++

- Mo 15.04.: 2. Vorlesung, davor Abgabe Übungsblatt 1
- Mo 15.04.: Ausgabe Übungsblatt 2
- Mi 17.04.: 2. Übung: Besprechen Übungsblatt 1

- Mo, 22.04.: 3. Vorlesung, davor Abgabe Übungsblatt 2
- Mo, 22.04.: Ausgabe Übungsblatt 3 mit Programmierblatt 1
- Mi 24.04. 3. Übung: Besprechen Übungsblatt 2

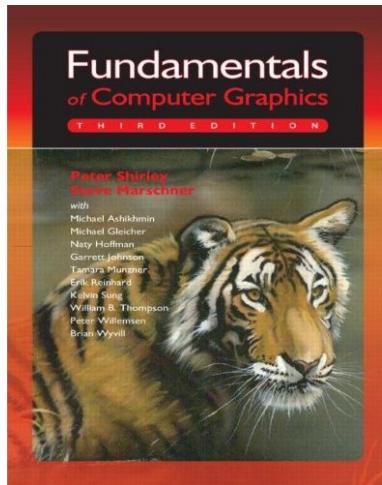
- Mo, 29.04.: 4. Vorlesung, davor Abgabe Übungsblatt 3
- Mo, 29.04.: Ausgabe Übungsblatt 4
- Mi 01.05.: keine Übung (1.Mai)



Für Übungsaufgaben und Programmieraufgaben

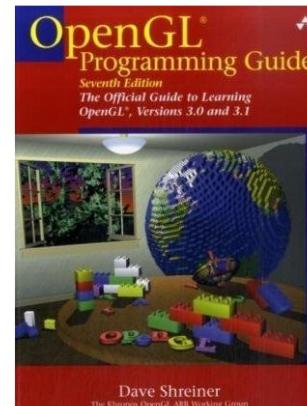
- Gruppen 1-3 Personen möglich
- nach erster gemeinsamer Einreichung können keine neuen Mitglieder in eine Gruppe kommen
- erlaubt: Gruppenverkleinerungen und Gruppensplits
- bitte immer alle Gruppenmitglieder und Matrikelnummer auf der Abgabe vermerken!

Literatur



Peter Shirley
Fundamentals of Computer Graphics
AK Peters, 3. Auflage

Dave Shreiner
OpenGL Programming Guide
Morgan Kaufmann, 7. Auflage



Beide Bücher sind mehrfach in der
Bibliothek als Präsenz – und
Ausleihexemplar vorhanden

Literatur



■ Bücher

- J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes: Computer Graphics - Principles and Practice (second Edition). Addison-Wesley Publishing Company, Inc., 1996
- J. Encarnacao, W. Straßer, R. Klein: Gerätetechnik, Programmierung und Anwendung graphischer Systeme Teil I und II. Oldenbourg, München, Wien, 1996, 1997
- D. Salomon: Computer Graphics Geometric Modeling, Springer, 1999
- A. Watt: 3D Computer Graphics. Addison-Wesley Publishing Company, Inc., 2000

■ Zeitschriften

- Computer Graphics Forum
- IEEE CG & Applications
- ACM Transactions on Graphics
- ...

Acknowledgements



Diese Vorlesungsskripte basieren teilweise auf Skripten von

- Prof. Heidrun Schumann (Universität Rostock)
- Prof. Marcus Magnor (Universität Braunschweig)
- Prof. Thorsten Grosch (Universität Clausthal-Zellerfeld)

und weiteren KollegInnen.

Dankeschön!



1. Gliederung

- 1. Einführung
- 2. Modellierung und Akquisition graphischer Daten
- 3. Rendering
- 4. Raytracing
- 5. Aktuelle Themen der Computergraphik im Überblick



1. Einführung



1. Einführung

1.1 Einordnung und Begriffsbestimmung

1. Einführung

1.1 Einordnung und Begriffsbestimmung



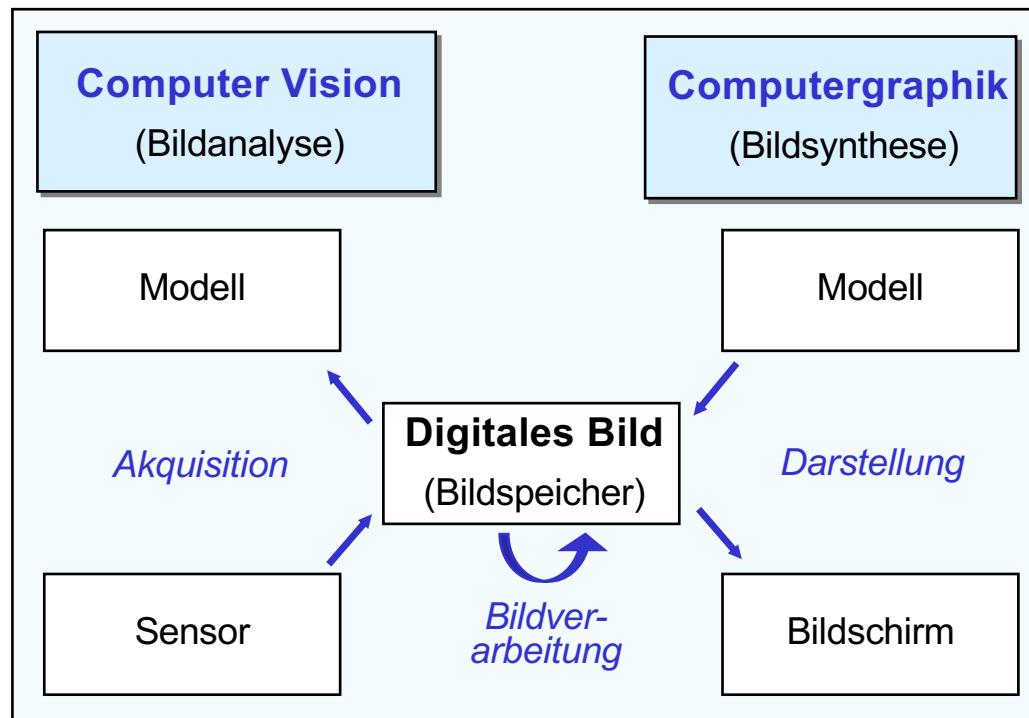
Die graphische Datenverarbeitung ist in der ISO-Norm wie folgt definiert (ISO DIS 2382/13, 1982):

*“Methods and techniques for converting data
to and from
graphics displays via computer”*

Mit “to” und “from” sind zwei Wege aufgezeichnet, einmal von der Beschreibung (Modell) zum Bild und einmal vom Bild zur Beschreibung.

1. Einführung

1.1 Einordnung und Begriffsbestimmung



Dualität von Computergraphik und Computervision

1. Einführung

1.1 Einordnung und Begriffsbestimmung



- Die Computergraphik behandelt die Erzeugung und Manipulation künstlicher Bilder mit dem Computer, die in Form von Bildbeschreibungen (Modell) vorliegen.
- Die klassischen Hauptrichtungen der Computergraphik sind:
 - **Modellierung/Akquisition**
Generierung der Bildbeschreibung in maschinell-interpretierbarer Form als Voraussetzung der Computergraphik
 - **Bearbeitung (Processing)**
Hierarchien, Segmentierungen, Komprimierungen, Umwandlungen
 - **Rendering (Darstellung)**
Erzeugung des Bildes aus der Bildbeschreibung
 - **Interaktion**
Manipulation des Bildes bzw. der Bildbeschreibung



1. Einführung

1.2 Geschichte der Computergraphik

1. Einführung

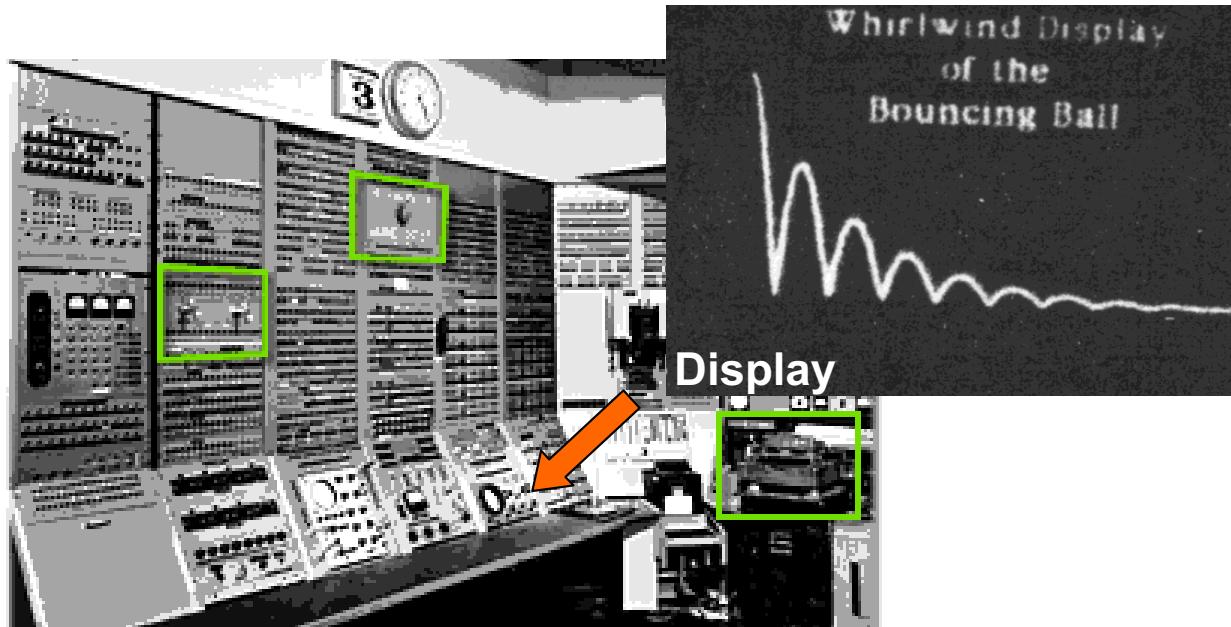
1.2 Geschichte der Computergraphik



Geschichte der Computergraphik

1949 Erste Computergraphik auf dem Whirlwind

Computer des MIT (**bouncing Ball Program** von C. Adams);



1. Einführung

1.2 Geschichte der Computergraphik



1952 Einsatz der Computergraphik zur Kennzeichnung von Flugobjekten auf Radarbildschirmen
(SAGE Computer mit 82 Graphikkonsolen zur Luftüberwachung, erster Einsatz eines Lichtgriffels);



Sage Consolen

1. Einführung

1.2 Geschichte der Computergraphik



- **1962** Erste **3D Computergraphiken** von L.G. Roberts auf dem TX2 des MIT;



TX2 Computer

1. Einführung

1.2 Geschichte der Computergraphik



- **1963** **Sketchpad** - erstes interaktives Computergraphiksystem von Sutherland mit
 - Bildkomposition aus graphischen Standardelementen,
 - Interaktion mit Tastatur und Lichtgriffel zur Arbeit mit Menüs,
 - entsprechenden Datenstrukturen zur Verwaltung graphischer Daten;



Ivan Sutherland an der Konsole des TX-2 Computers

1. Einführung

1.2 Geschichte der Computergraphik

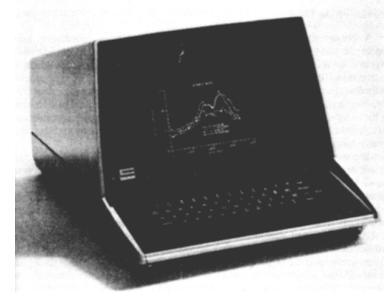


Mitte der 60er-Jahre: Beginn einer Vielzahl von CG-Forschungsprojekten (am MIT, General Motors, Bell Telephone Lab., Lockheed Aircraft usw.)

1965: Erstes kommerzielles Vektor-Display von IBM

(Preis ca. 100.000 US\$);

1967: Erster Bildspeicher-Display
von Tektronix (bietet einem
breiten Interessentenkreis Zugang
zur Computergraphik, Preis ~10.000 USD);



1. Einführung

1.2 Geschichte der Computergraphik



Anfang der 70er Jahre: erste kommerzielle CAD/CAM-Systeme kommen auf den Markt.

1971: **Raster-Scan-Prinzip** von M. Noll (Bell Lab.) vorgeschlagen;

1972: Erster Flugsimulator (General Electronics);

Erster **Heimcomputer** als Bausatz unter dem Namen „Altair 8800“ auf dem Markt

1. Einführung

1.2 Geschichte der Computergraphik



Anfang der 70er Jahre: erste kommerzielle CAD/CAM-Systeme kommen auf den Markt.

- **1973:** Erste Konferenz der **SIGGRAPH** (Special Interest Group on Computer **Graphics**) der ACM (Association of Computing Machinery), die sich ausschließlich mit Computer Graphik beschäftigt (ca. 1200 Teilnehmer);

1. Einführung

1.2 Geschichte der Computergraphik



- Ab Mitte der 70er Jahre:
 - Graphische Unterprogrammpakete (PLOT10, CAL-Comp),
 - Graphische Programmiersprachen (DIGRA 73),
 - Erste **kommerzielle Raster-Displays** (max. Auflösung 512 x 512 Pixel, 8 Bit pro Pixel, Preis ~100.000DM);
 - Erste Verfahren zur schattierten Objektdarstellung:
 - Beleuchtungsverfahren (Phong 1975, Blinn 1977),
 - Schattierungsverfahren (Gouraud 1971, Phong 1975),
 - Texturierung (Catmull 1974),
 - Schattenwurf (Crow 1977);



*M. Newell (Univ. of Utah) modelliert
1975 den **Utah tea pot** –
eine Ikone der Computer Graphik.*

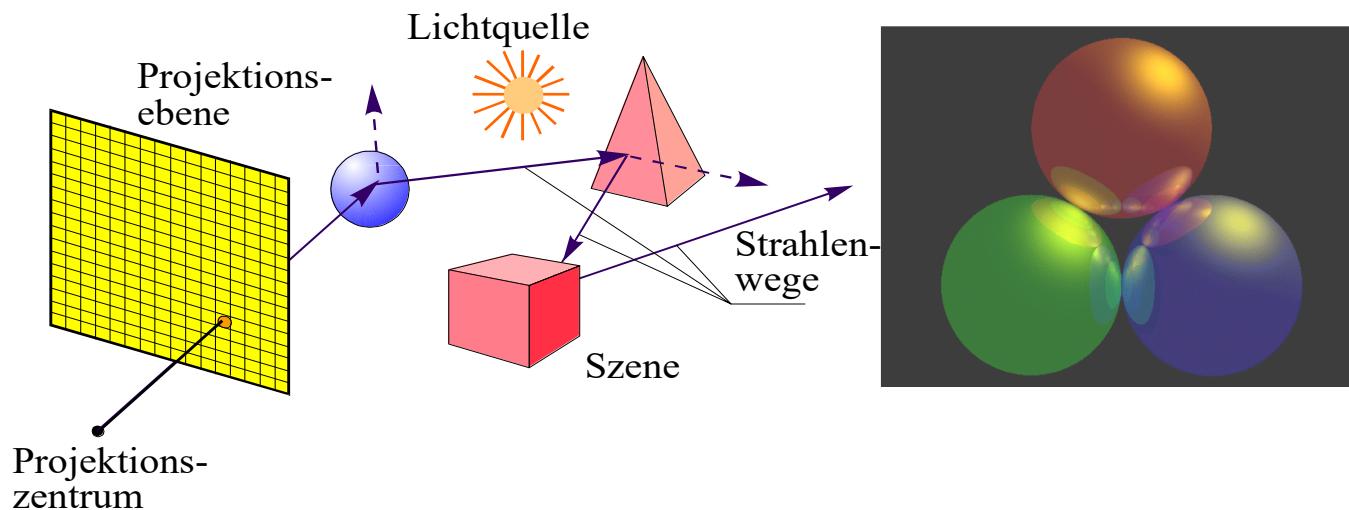


1. Einführung

1.2 Geschichte der Computergraphik



- **1977** Erste Vorschläge zur Standardisierung von Graphiksoftware - **CORE**;
- **1979** Erstmalige Darstellung von spiegelnder Reflexion und Transparenz mit Hilfe des **Raytracing** (Kay);

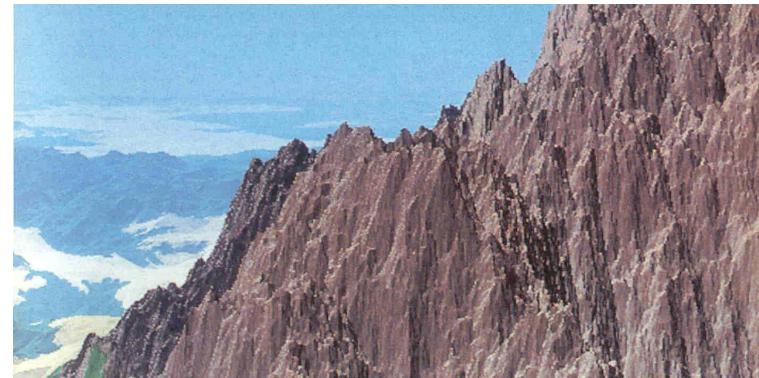


1. Einführung

1.2 Geschichte der Computergraphik



- **1980:** Vorführung des Films *Vol Libre* (von L. Carpenter, Boeing) auf der SIGGRAPH'80 (in dem Film wird der Flug durch eine **fraktale Landschaft** gezeigt);



Carpenter's Kunst-Gebirge wurde bei der Prämierung von der SIGGRAPH-Jury mit der Begründung ausgeschlossen, weil

„... es nicht wie eine Computer Graphik aussieht !“

1. Einführung

1.2 Geschichte der Computergraphik



1980: ~30 Min. Computeranimationen im Film **Tron**, (Film floppt, große Hollywood Filmstudios reagieren mit Zurückhaltung gegenüber Computer Graphik);

1981: Erstes „Rendering“-System **REYES** („Rendering everything you ever saw“, von L. Carpenter für LucasFilm – wird später zu **Renderman** weiterentwickelt);

Beginn der Entwicklung des **Volume-Rendering**;

1. Einführung

1.2 Geschichte der Computergraphik



1982: Erste Filmsequenz, in der sich Frau in Luchs verwandelt (T.Brigham, SIGGRAPH'82);

Diese Technik wird später
Morphing genannt -

(wurde bis 1987 nicht weiter
beachtet, bis LucasFilm sie in
dem Film „Willow“ einsetzt);



1. Einführung

1.2 Geschichte der Computergraphik



- **1982:** Gründung **Silicon Graphics Inc. (SGI)**, J. Clark, (Entwicklung von Hochleistungsrechnern für graphische Anwendungen);
- **1983:** J. Lanier (Atari Research Center) entwickelt **Datenhandschuh**;



1. Einführung

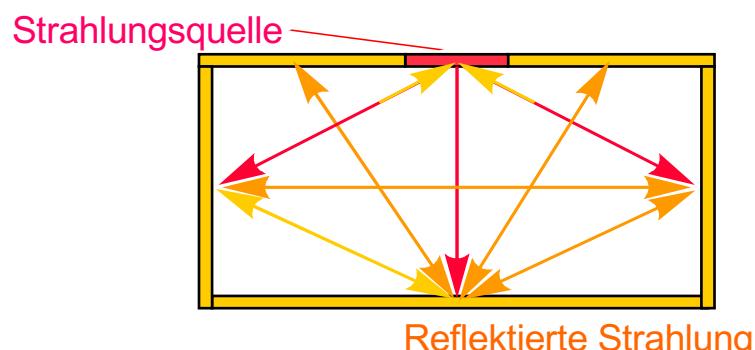
1.2 Geschichte der Computergraphik



■ **1984:** Globale Beleuchtungs-Simulation

mit **Radiosity** (Goral u.a., Nishita);

Gründung Wavefront Technologies
für **Animations-Software**;



1. Einführung

1.2 Geschichte der Computergraphik



- **1985:** Gründung der Fa. VLP durch J. Lanier, um die ersten kommerziellen **Virtual Reality** Produkte zu entwickeln;
GKS - erster graphischer ISO-Standard
(1988 GKS-3D);

1. Einführung

1.2 Geschichte der Computergraphik



1986: Gründung der Fa. Pixar durch Ed Catmull und A.R. Smith nach Abspaltung von Lucas Film;

- Pixars **RenderMan** wird Industrie-Standard;
- **1988:** Film **The Abyss**, James Cameron

(ILM stellt dabei die Szene mit der Wasser-Kreatur her, die die Gesichter der Mannschaft imitiert);

*aus J.D. Foley, et al
Computer Graphics – Principles
and Practice*



1. Einführung

1.2 Geschichte der Computergraphik



1989: Einführung von **Motion Capture**, Nutzung mechanischer Eingabegeräte für Computeranimation;

1992: Neue Maßstäbe bei computer-generierten Spezialeffekten, (Animationen des „T1000“-Roboters in J. Cameron's Film **Terminator 2**);



1. Einführung

1.2 Geschichte der Computergraphik



1993: Steven Spielberg's Film

Jurassic Park

(anstelle der ursprünglich geplanten
Puppenanimationen werden Computer-
animationen für die Dinosaurierszenen
eingesetzt);



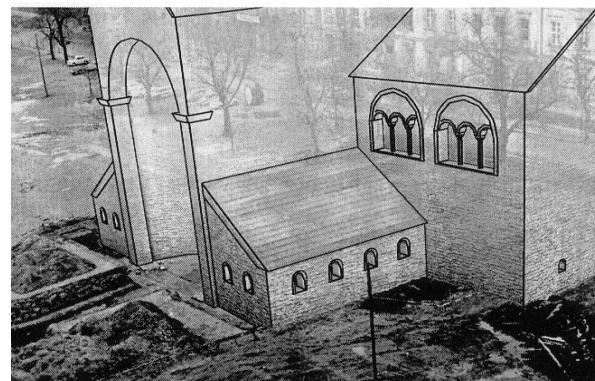
1. Einführung

1.2 Geschichte der Computergraphik



→ Anfang- Mitte der 90er Jahre:

- Verbindung moderner Kommunikationstechnologien mit Graphik:
 - ⇒ Multimedia in verteilten Umgebungen,
 - ⇒ CSCW Computer supported cooperative work,
 - ⇒ Graphik im Internet,
 - ⇒ Standards zur Bild- und Bewegtbildübertragung...,
- Daten- und Informations-Visualisierung,
- Methoden des Non-Photorealistic Rendering;



1. Einführung

1.2 Geschichte der Computergraphik



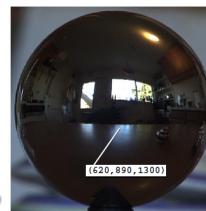
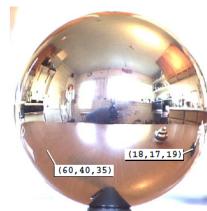
- 1995: **Toy Story** kommt in die Kinos; erster vollständig computeranimierte Film von Pixar;
 - Rendering: 800 000 Std. Berechnungszeit für 70 min. Film auf 177 Sun Sparc 20
 - Der Film ist ein großer Kassenerfolg, wird für drei Oskars nominiert, erhält jedoch keinen davon ...





Geschichte der Computergraphik

1998: High Dynamic Range (**HDR**), Paul Debevec
8 Bit → 32 Bit float pro Pixel
Digitale Kamera nimmt echte Helligkeiten auf



HDR Light Probe



Fiat Lux 1999



HDR



LDR

Image-based Lighting

Geschichte der Computergraphik



Grafikkarten

1999 - heute: Graphics Processing Unit (GPU)

SGI Grafik-Workstation wird immer mehr von PC mit Grafikkarte abgelöst

Starke Weiterentwicklung durch
Spieleindustrie



GPU wird zum schnellen Coprozessor, bisherige Offline-Techniken werden Echtzeitfähig

1. Einführung

1.2 Geschichte der Computergraphik



2001: sehr gute Gesichtsanimationen

von virtuellen Charakteren im

Film **Shrek** ;



Simulation von Haaren: **Monster AG**

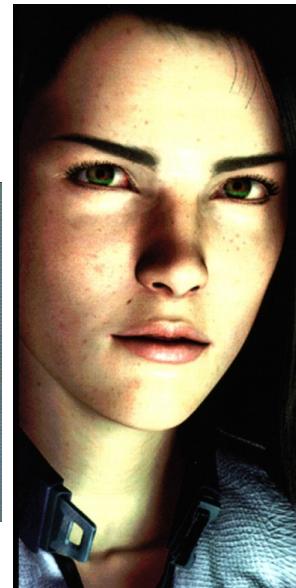
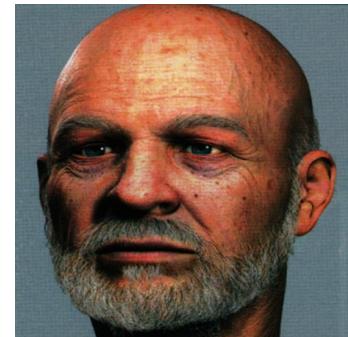


1. Einführung

1.2 Geschichte der Computergraphik



2001: Final Fantasy, erster
komplett computer-
animierter Spielfilm mit
realistischen virtuellen
Charakteren;
die Produktion des Films
dauerte ~4 Jahre, wobei etwa
170 Computeranimatoren
tätig waren



*aus: digital production,
Heft 3/01*



Geschichte der Computergraphik

Sehr große, virtuelle Welten...

2003: Herr der Ringe



2009: Avatar



2019... ?



“Point-based Global Illumination”

Pixar, Industrial Light & Magic



Point-Based Approximate Color Bleeding

Per H. Christensen
Pixar Technical Memo #08-01
Pixar Animation Studios



Figure 1: (a) Point-based ambient occlusion test from "Surf's Up" © Sony Pictures Imageworks. (b) Point-based color bleeding in a final frame from "Pirates of the Caribbean: Dead Man's Chest" © Disney Enterprises, Inc. and Jerry Bruckheimer, Inc., image courtesy of Industrial Light & Magic.

Abstract

This technical memo describes a fast point-based method for computing diffuse global illumination (color bleeding). The computation is 4–10 times faster than ray tracing, uses less memory, no noise, and its run-time does not increase due to displaced mapped surfaces, complex shaders, or many complex light sources. These properties make the method suitable for movie rendering.

The input to the method is a point cloud (visual representation) of the directly illuminated geometry in the scene. The surfaces in the point cloud are clustered together in an octree, and the power for each cluster is approximated using spherical harmonics. To do this, the indirect illumination at a spherical position, ω , is computed by summing all surface weights at three degrees of freedom, ω , from each cluster. This is done by ray casting the octree and summing the surface weights along the ray, by ray casting the octree nodes and summing the surface weights along them. Variations of the method efficiently compute area light illumination and soft shadows, fast gathering for photon mapping, HDR environment map illumination, multiple diffuse reflections, horizons, ambient occlusion, and glossy reflection.

The method has been used in production of more than a dozen feature films, for example for rendering Davy Jones and his crew.

Keywords: Global illumination, color bleeding, radiosity, lights, ambient occlusion, point clouds, point-based rendering.

feels, complex scenes, movie production.

1 Introduction

Standard methods for global illumination (such as radiosity [Goral et al. 1984], distribution ray tracing [Ward et al. 1988], and pre-

geometry — the light sources, surface and displacement shaders are also very complex and take a long time to evaluate.

In this technical memo we describe a two-pass point-based approach that is much faster and uses less memory than the standard methods. First a useful representation of the directly illuminated geometry is created in a precomputation phase and stored in a point cloud file. (A *surfel* is a disk-shaped surface element, i.e. a point with associated normal, radius, and other data such as color. In our case, each surfel represents colored reflections from a small part of a surface.) Then the surfels in the point cloud are organized into an octree hierarchy, and the illumination from the surfels in each octree node is approximated using spherical harmonics.

To compute the global illumination at a surface point, we add the illumination from all surfaces using three degrees of accuracy: overlapping surfaces are traced, other nearby surfaces are approximated as disks, and distant surfaces are approximated by evaluating a spherical harmonic representation of clusters. The clusters are assigned according to distance and rasterized onto a cube map of "pixels". Finally, the indirect illumination (color shading) is computed by multiplying the pixel colors with the BRDF for the chosen reflection model (for example cosine weights for diffuse reflection) and the solid angle of the raster pixels.

Huge point sets are handled efficiently by reading the octree nodes and warfs on demand and caching them.

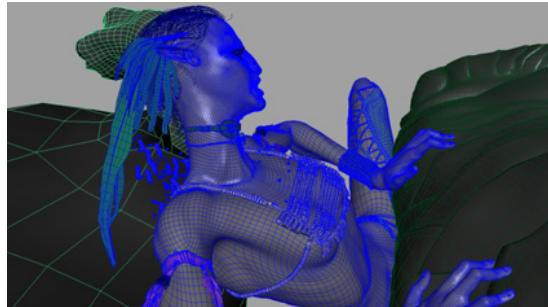
The advantages of our point-based global illumination method are faster computation time, the geometric primitives do not have to be kept in memory, no ray-tracing acceleration data structure is needed, only a small fraction of the octrees and sunels need to be in memory at any given time, no noise, displacement mapping and complex light source and surface shaders does not slow it down, environment illumination does not take any additional time. The disadvantages are mainly that it is a two-pass approach, and that the results are not guaranteed to be as precise as ray tracing. However, our method is not surprised by convexity, it handles concave geometry correctly.

[Fluch der Karibik 2, 2006]

“PantaRay – Visibility Precomputing”

Nvidia, Weta Digital

[Avatar, 2009]



PantaRay: Fast Ray-traced Occlusion Caching of Massive Scenes

Jacopo Pantakos^{*} NVIDIA Research Luca Falcione[†] Weta Digital Martin Hill[‡] Weta Digital Timo Aila^{*} NVIDIA Research

Abstract

We describe the architecture of a novel system for precomputing scene directional occlusion caches. These caches are used for accelerating a fast cinematic lighting pipeline that works in the spherical harmonic domain. Our system is designed to handle the complex lighting technology in the movie Avatar, and is able to efficiently handle massive scenes of unprecedented complexity through the use of a novel cache organization and a novel out-of-core ray tracing algorithm. We also introduce a novel out-of-core GPU ray tracing algorithm for the computation of directional occlusion and spherical integrals at arbitrary points.

CRC Categories: I.3.3 [Graphical Systems]: C.2.1, C.2.4, C.3.1
General audience; Application Domains: Computer Graphics and Visualization; Applications: Realistic Colorizing, Animation, and Interactive Applications

Keywords: global illumination, precomputed radiance transfer, caching, out-of-core

^{*}E-mail: {jpantakos,timo.aila}@nvidia.com
[†]E-mail: luca.falcione@weta.co.nz
[‡]E-mail: martin.hill@weta.co.nz

ACM Reference Format:
Pantakos, J., Falcione, L., Hill, M., and Aila, T. 2010. PantaRay: Fast Ray-traced Occlusion Caching of Massive Scenes. In Proceedings of the ACM SIGGRAPH 2010 Paper Competition, 1–10. New York, NY, USA, July 25–29, 2010. © 2010 ACM, Inc. 978-1-4503-0095-6/10/07 \$15.00.
Figure 1: The geometric complexity of scenes rendered in the movie Avatar often exceeds a billion polygons and spans widely distant rocks and vegetation are tessellated to a level of meters and centimeters, while the faces of even distant characters are modeled to cover distances from forehead to chin. The spatial resolution of occlusion caches precomputed by our system also spans several orders of magnitude.

1 Introduction

The movie Avatar features unprecedented geometric complexity (Figure 1), with production shots containing anywhere from six million to over one billion polygons.

To handle such a large number of polygons makes scene management while satisfying the need to provide fast lighting iterations for lighting artists and the director, modern rendering methods based on spherical harmonics and precomputed radiance transfer (PRT) [Kajiya 1984] and ray-based lighting [Debevec 1998] were used. These methods can speed up the lighting iterations significantly, but unfortunately require the computation of directional occlusion and spherical integrals at arbitrary points.

We describe PantaRay¹, a system designed to make this precomputation practical by leveraging the development of modern multi-GPU ray tracing [Haines et al. 2009], novel cache organization [Aila et al. 2009] and combining them with new out-of-core [Aila et al. 2009] and indirect lighting information for billions of pixels with highly complex scenes.

The PantaRay engine is an out-of-core, massively parallel ray tracer designed to handle scenes that are roughly an order of magnitude larger than available system memory, and that can handle baking and real-time rendering of distributed environments (3D cities) and indirect lighting information for billions of pixels with highly complex scenes.

Our key contribution is the introduction of a flexible, scene-based geometry processing architecture, a novel out-of-core algorithm for the computation of directional occlusion and spherical integrals, and a novel out-of-core GPU ray tracing algorithm for the computation of directional occlusion and spherical integrals. These are

¹A twist on the Greek verb *anerkenen*, i.e. *acknowledging*.

“Volumetric Lighting”

Disney Research

[Rapunzel, 2010]



A Programmable System for Artistic Volumetric Lighting

Derek Nowrouzezeh¹ Jared Johnson² Andrew Selle² Dylan Laine² Michael Karchuk² Wojciech Jarosz¹

¹Disney Research Zurich ²Walt Disney Animation Studios



Figure 1: Our system was used to author artistic volumetric effects for the movie *Tangled*. Our technique’s ability to produce curling light beams is used to match the organic artistic style of the film.

Abstract

We present a method for generating art-directed volumetric effects, ranging from physically-accurate to highly stylized. Our system allows the expressive artist direct control over volumetric effects by using an intuitive lighting primitive, and decoupling the manipulation of shading from the lighting. To enable this, we generalize the rendering of volumetric effects to be orthogonal to artistically programmable simulation and shading phases. This provides an artist with a wide range of tools for manipulating volumetric effects at physically-based as well as plausible, but exaggerated, volumetric levels. We integrate our approach into a real-world production pipeline, and couple our volumetric effects to camera tracking.

CC Category: I.3.3 [Computer Graphics]: Three-dimensional Graphics and realism—Color, shading, shadowing, and texture.

Keywords: Lighting design, artist control, participating media

Link: [DOI](#) [PDF](#)

1 Introduction

Light scattering in participating media is responsible for many natural phenomena. Simulating the evolution of volumetric media over time, as well as the complex interactions between different media components (e.g., [Huang et al. 2001; Janusz et al. 2003]) and rendering [Jones and Christensen 1998; Janusz et al. 2003] have been an active area of research for a wide range of applications in feature animation production; however, most of this work has focused on automating computation and rendering of volumetric effects, leaving the artist to manipulate physical parameters to attain a target look in a challenging

(or even, art-) style (the author’s version of the work). It is printed here by permission of the author, and is not to be reproduced without the express written permission of the author. This version was published in *ACM SIGGRAPH Asia 2011*, which provides the primary source of information about the content of the paper.

© 2011, art-. While the author’s version of the work is available online, the definitive version of the work is the version published in *ACM SIGGRAPH Asia 2011*.

● We observe that manipulating physical parameters of participating media results in undesirable changes to the final image. I address this by defining a physically-based scattering properties interface that allows the artist to directly manipulate space for appearance modeling of participating media.

● To allow for non-physical effects, we generalize both the physics and interface, and introduce new types of the photo media models. We replace each stage with a component that is programmable. While each component could implement the physically-based approach, this provides enough po-

ing process. Previous work addresses this problem by investigating the manipulation of physically-based scattering media [Kerr et al. 2010]. However, artistic authoring and manipulation of volumetric lighting is much more complex.

Gaining accurate fluid animation using arbitrary source terms while maintaining a physically-based framework is previously unexplored [McNamee et al. 2004; McNamee et al. 2005]. Our framework allows for programmable and art-directed injection of source terms into physically-based volumetric light transport.

While physically-based and art-directed rendering share some high-level goals, most efforts in implementing physically-based rendering into production have shown great potential [Taubin et al. 2004; Karpov et al. 2009]. Such techniques are often very involved and difficult to learn. In contrast, our volumetric lighting which would otherwise take extensive manual configuration and tuning, can be easily controlled by artists. Physically accurate rendering is often not sufficiently expressive for the caricatured nature of motion picture film, though physically-based rendering is a great starting point. This is why we focus on the extremes introducing controls necessary to obtain a desired aesthetic vision. We can also combine the physically-based framework to general art-directed rendering by applying approaches for non-physical scattering lighting to art-directed shading and simulation (Section 4).

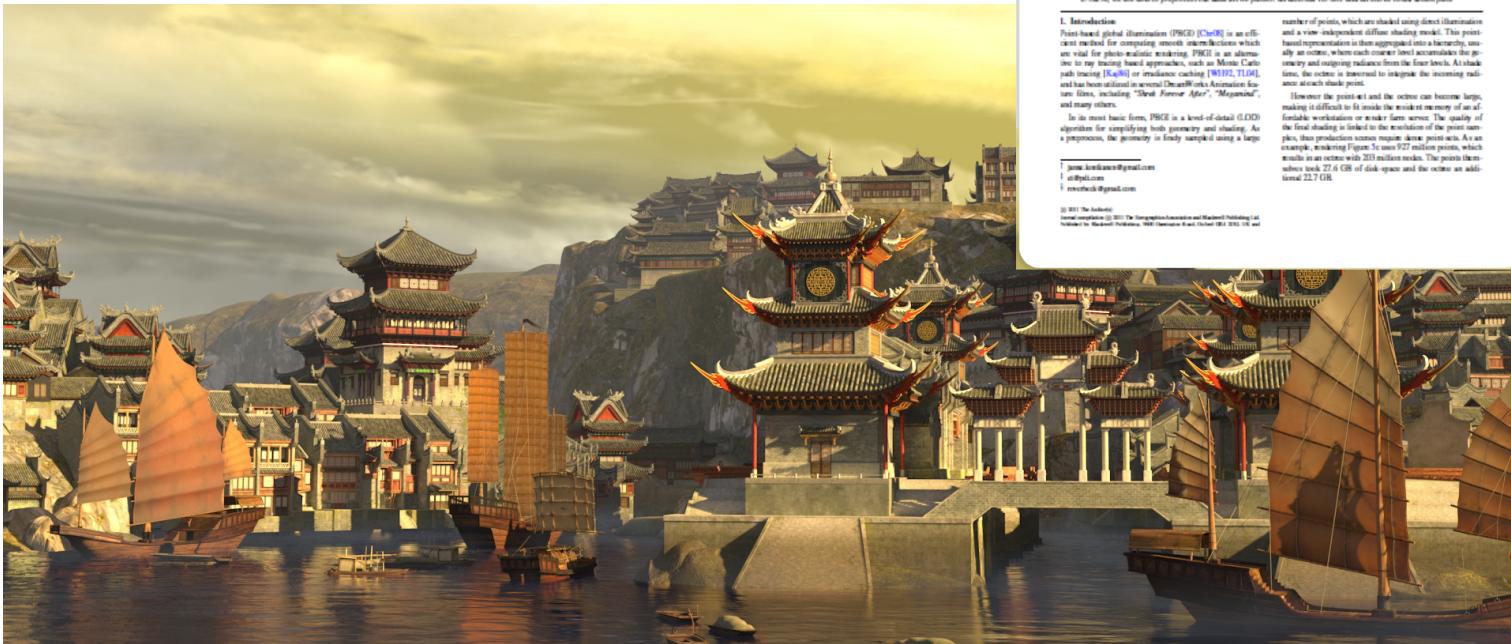
We present a system for generating large populations of volumetric effects, and tracking the way performance artist hand draw them on set. We have developed a physically-based rendering framework which provides physically-based rendering of participating media. We provide a physically-based scattering properties interface to allow for artistic control of volumetric effects.

● We observe that manipulating physical parameters of participating media results in undesirable changes to the final image. I address this by defining a physically-based scattering properties interface that allows the artist to directly manipulate space for appearance modeling of participating media.

● To allow for non-physical effects, we generalize both the physics and interface, and introduce new types of the photo media models. We replace each stage with a component that is programmable. While each component could implement the physically-based approach, this provides enough po-

“Out-of-Core Global Illumination”

DreamWorks Animation
[Kung Fu Panda 2, 2011]



Coherency in representations for rendering scenes
Eric Keenleyside and Eric Overbeck
(Siggraph 2011)

Coherent Out-of-Core Point-Based Global Illumination
Jason Krikun¹ and Eric Tabellion² and Ryan S. Overbeck³
DreamWorks Animation

Figure 1: A screenshot from the DreamWorks Animation movie “Kung Fu Panda 2”. The out-of-core method described in our paper shaded the global illumination for the whole frame in 6 minutes 23 seconds. It used 170 million points, and it took an additional 4 minutes 18 seconds to build the out-of-core octree, resulting in 27 million octree nodes. The total amount of out-of-core data was 27 GB while the in-core point and octree cache, memory, did not exceed 1 GB. The cache hit rate was 99%. In Section 7, we demonstrate storage of up to 27 billion points (800 GB) of disk-space within a 1 GB memory cap.

Abstract
We describe a new technique for coherent out-of-core point-based global illumination and ambient occlusion.
Point-based global illumination (PBGI) is used in production to render tremendously complex scenes, so its core storage of point and octree data structures quickly becomes a problem. However, a simple out-of-core extension of a classic PBGI algorithm can be used to efficiently manage this data without losing quality or performance.
Our method extends previous PBGI algorithms with an out-of-core technique that uses minimal I/O and stores data on disk compactly and in coherent chunks for later access during shading. Using properties of a space-filling Z-curve, we are able to preprocess the data in two passes: an external ID sort and an octree construction pass.

1. Introduction
Point-based global illumination (PBGI) [Car08] is an efficient method for computing global illumination effects for real-time rendering systems. PBGI is similar to ray-tracing based approaches, such as Monte Carlo path tracing [Kas08] or irradiance caching [Wit02, Tl04], and has been utilized in several DreamWorks Animation feature films, including “Shrek Forever After”, “Megamind”, and many others.

In its most basic form, PBGI is a level-of-detail (LOD) algorithm for simplifying both geometry and shading. As a prepass, the geometry is finely sampled using a large number of points, which are shaded using direct illumination and a view-independent diffuse shading model. This point-based representation is then aggregated into a hierarchy, usually a Z-curve [Car08], which allows for efficient access to geometry and rendering radiance from the four levels. At shade time, the octree is traversed to integrate the incoming radiance at each shade point.

However, the geometry and the octree can become increasingly difficult to fit inside the memory of many of affordable workstations or a single farm server. The quality of the final shading is linked to the resolution of the point samples, and the number of points per node grows exponentially. For example, rendering Figure 3c may 927 million points, which results in an octree with 203 million nodes. The points themselves take 27.6 GB of disk-space and the octree an additional 22.7 GB.

jason.krikun@dreamworks.com
eric.tabellion@dreamworks.com
ryan.s.overbeck@dreamworks.com

© 2011 The Author(s). Published by ACM in 2011. “The Morgan Kaufmann and Morgan Publishing LLC.”
Published by Morgan Kaufmann, 849 Howard Street, San Francisco, CA 94103-1434, and
London, England.



"Artistic Simulation of Curly Hair"

Disney, Pixar



Figure 1: Example of stylized curly hair simulated with our method. © Disney/Pixar

ABSTRACT

Artistic simulation of hair presents many challenges - ranging from separating artistic control of styling with extreme motions of racism, to maintaining performance requirements. In this paper, we present a method for simulating curly hair that needs to be fast and results need to be usable "out of the box".

(without extensive parameter modifications) in order to provide artists with tools. These challenges are only increased when simulating curly, oily hair.

We present a method for stably simulating stylized curly hair that meets these artistic needs and performance demands. To simulate the hair in a production environment, we propose a physics-based simulation that handles extreme motion. We introduce a method for stably computing a frame rate independent solution for the motion of the hair.

Our hair model uses a spring for controlling the bending of the curl radius. For maintaining the helical shape during simulation, it adds a helical constraint to the hair model. To handle the handling hair contact interactions by efficiently parallelizing the simulation, we propose a technique for partitioning both hair-hair and hair-particle interactions.

The method has been used on two full-length feature films and has even to be robust and stable over a wide range of animated motions. On a variety of hair styles, from straight to wavy to curly, it provides a stable and predictable simulation that allows the artists to use the simulation allowing our artists to achieve their desired performance when facing strict scheduling demands.

CATEGORIES: I.3.7 [Computer Graphics]: Three-Dimensional graphics and Realism—Animation

KEYWORDS: Hair simulation, Mass-spring models

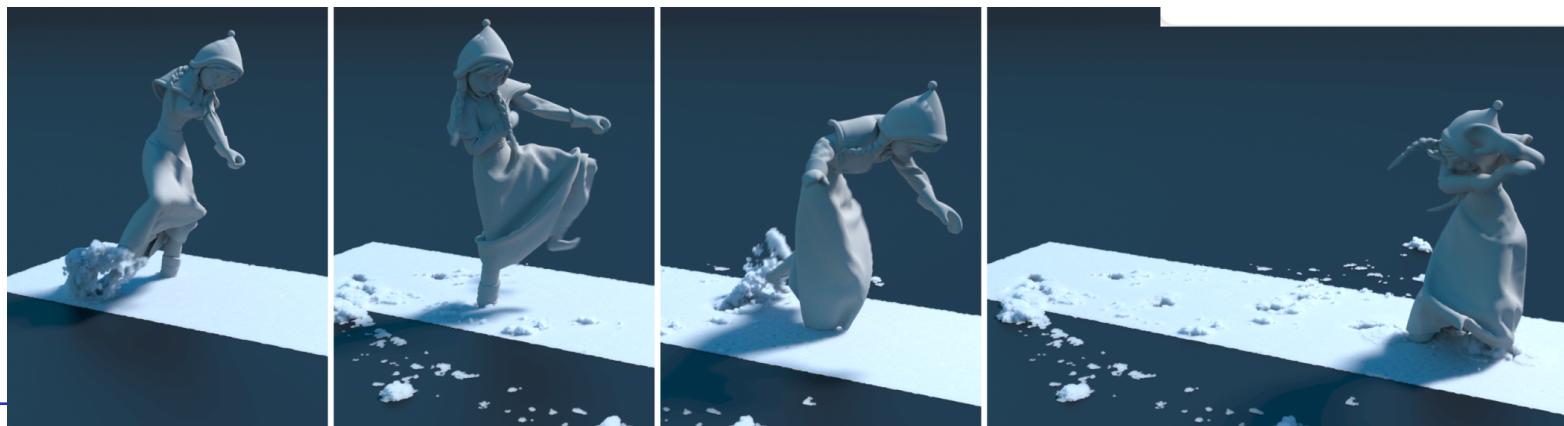
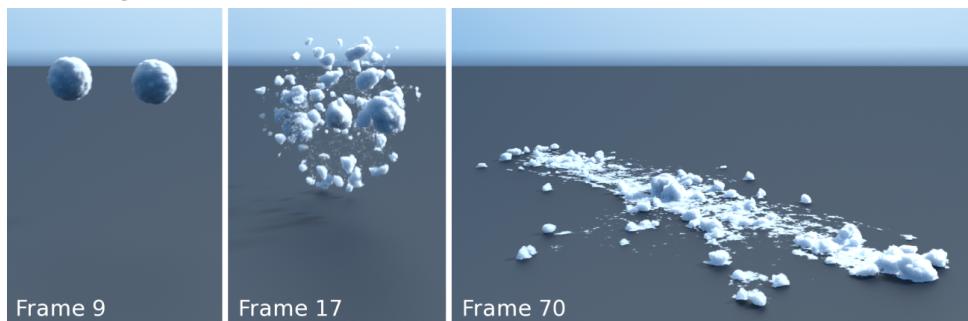


[Merida, 2012]



“Simulation of Snow”

Disney, Pixar [Eiskönigin, 2013]



A MATERIAL POINT METHOD FOR SNOW SIMULATION
Alexey Stomakhin[†] Craig Schroeder[‡] Lawrence Chair[‡] Joseph Tribar[†] Andrew Selle[†]
[†]University of California Los Angeles *Walt Disney Animation Studios

Abstract

Snow is a challenging natural phenomenon to visually simulate. While the graphics community has previously simulated accumulation and melting of snow, animation of snow dynamics has not been fully addressed. Additionally, existing techniques for solid and fluid simulation often do not handle snow well. Specifically, soft or dense snow that has both solid and fluid properties is difficult to handle. Currently, the paper presents a new material point method called MPMS. The paper proposes a semi-implicit material point method allowing a semi-controllable elasto-plastic constitutive model integrated with a hybrid Eulerian-Lagrangian scheme. The Eulerian part is continuous based and its hybrid nature allows us to use a regular Cartesian grid to simulate treatment of self-collision and fracture. It also allows us to use a semi-implicit Eulerian scheme for integration where that has conditioning independent of the number of Lagrangian particles. We demonstrate the power of our method with a variety of snow phenomena including complex character interactions.

CC Category: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.3.5 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: material point, snow simulation, physically-based modeling

Link: [DOI](#) [PDF](#) [WEB](#)

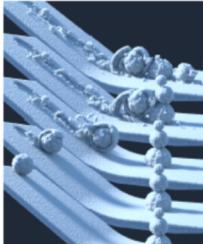


Figure 1: Rolling snowball. As the snowball moves down the slope, it picks up snow sticks, demonstrating that we can handle complex packing structures.

©Oliver

1 Introduction

Snow dynamics are amazingly beautiful yet a challenge. Whether it is possible to make a snowman or a child's winter coat disappear, creating an snow crest or even packing snow rolled into balls to make a snowman, it is snow's rich properties that makes it simultaneously compelling and difficult to simulate. While there are many ways to simulate snow, artists typically use simpler techniques combined in various ways to achieve some effect. For this reason, we believe that we must simultaneously handle a continuum of material properties of snow, from the most brittle, through the elastic, through a soft, and finally to the most viscous. This suggests the need for a specialized solver that handles difficult snow in a single shot.

Simulating snow for specific physics requires a complex and graphics-oriented computational code because achieving maximum resolution (and thus visual quality) requires efficient physics. While a fluid simulator can produce solid-like static effects (and vice versa), it is not the most optimal strategy. When solids and fluids are treated simultaneously, researchers have developed two-way coupling systems that are computationally heavy and perhaps not the best phenomena. Unfortunately, snow is a highly varying phase effects, sometimes behaving as a rigid-deforming solid and sometimes as a fluid. In addition, snow is a granular material, so we must simultaneously handle a continuum of material properties of snow, from the most brittle, through the elastic, through a soft, and finally to the most viscous. This suggests the need for a specialized solver that handles difficult snow in a single shot.

We present two main contributions that achieve these aims. First, we develop a semi-implicit Material Point Method (MPMs) [Selle et al. 1995] specifically designed to efficiently treat the wide range of snow behaviors. Second, we propose a semi-implicit Eulerian scheme for complex snow sources. To the knowledge, this is the first time MPM has been used to propagate snow from a source. Notably, material particles interact with Eulerian Continuum grids. Notably, there is no inherent need for Lagrangian mesh connectivity. Many numerical methods for snow simulation use Eulerian finite difference methods. For example, [Zhu and Yang 2012] simulate sand as a fluid using a PCTELP incompressible fluid solver. In fact, MPMs are also Eulerian finite difference methods, but for complex solids. As with PCTELP, MPMs implicitly handle self-collision and fracture with the use of the background Eulerian grid. Our Eulerian scheme is able to capture the behavior exhibited by practical snow dynamics. Our second contribution is a new snow constitutive model designed for intuitive user control of realistic snow behavior. This is also illustrated to achieve our goal of



Computergraphik in Magdeburg

AG Visual Computing:

Prof. Holger Theisel

AG Visualisierung:

Prof. Bernhard Preim

AG Computer-Assisted Surgery:

Jun.-Prof. Christian Hansen



Computergraphik in Magdeburg

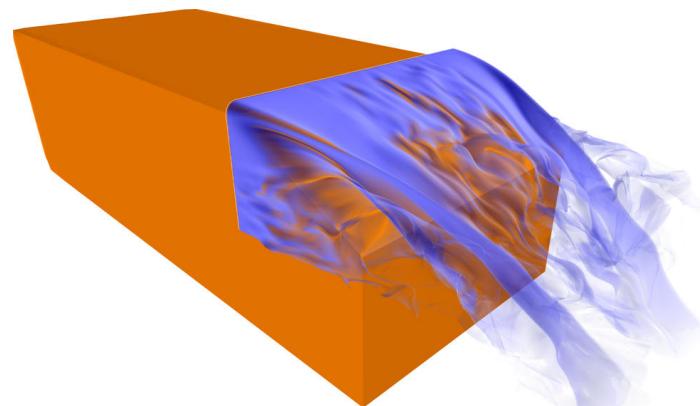
AG Visual Computing
Prof. Holger Theisel

- Visualisierung
 - speziell Strömungsdaten
- Information Visualization
- Visual Analytics
- Geometric Modelling, geometric Design

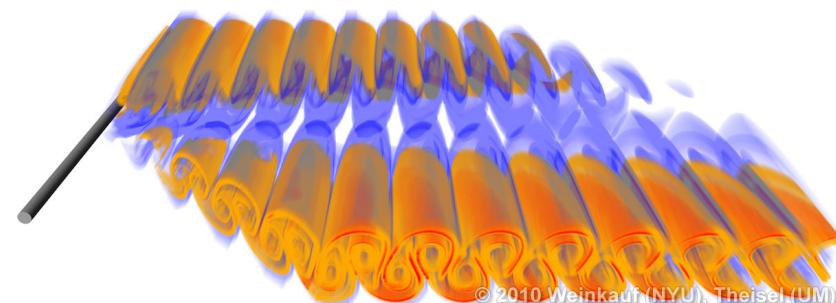




AG Visual Computing: Flow Visualization



© 2008 Weinkauf (ZIB), von Funck (MPII), Theisel (UM), Wassen (TUB)



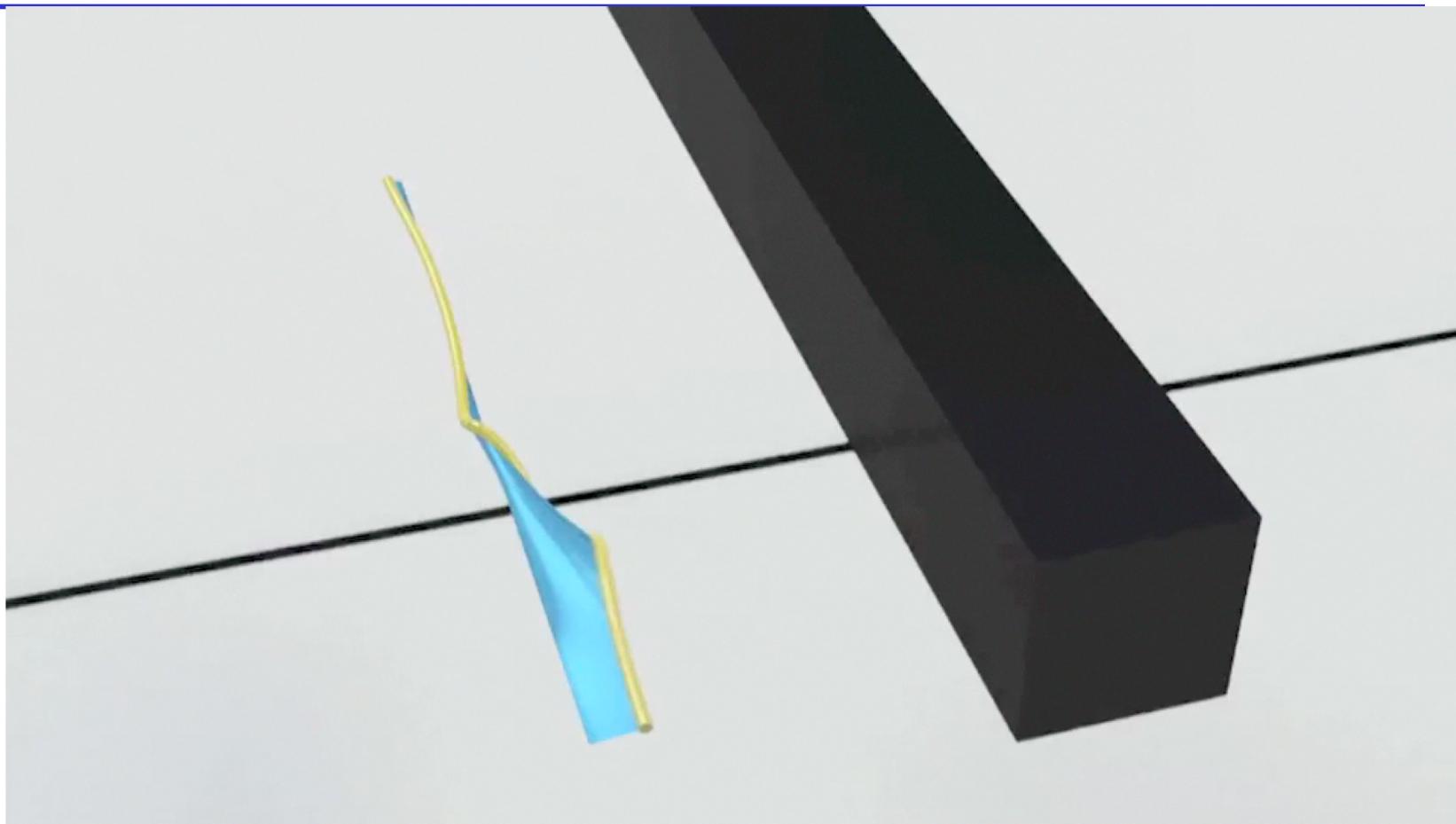
© 2010 Weinkauf (NYU), Theisel (UM)



© 2010 Weinkauf (NYU), Theisel (UM)

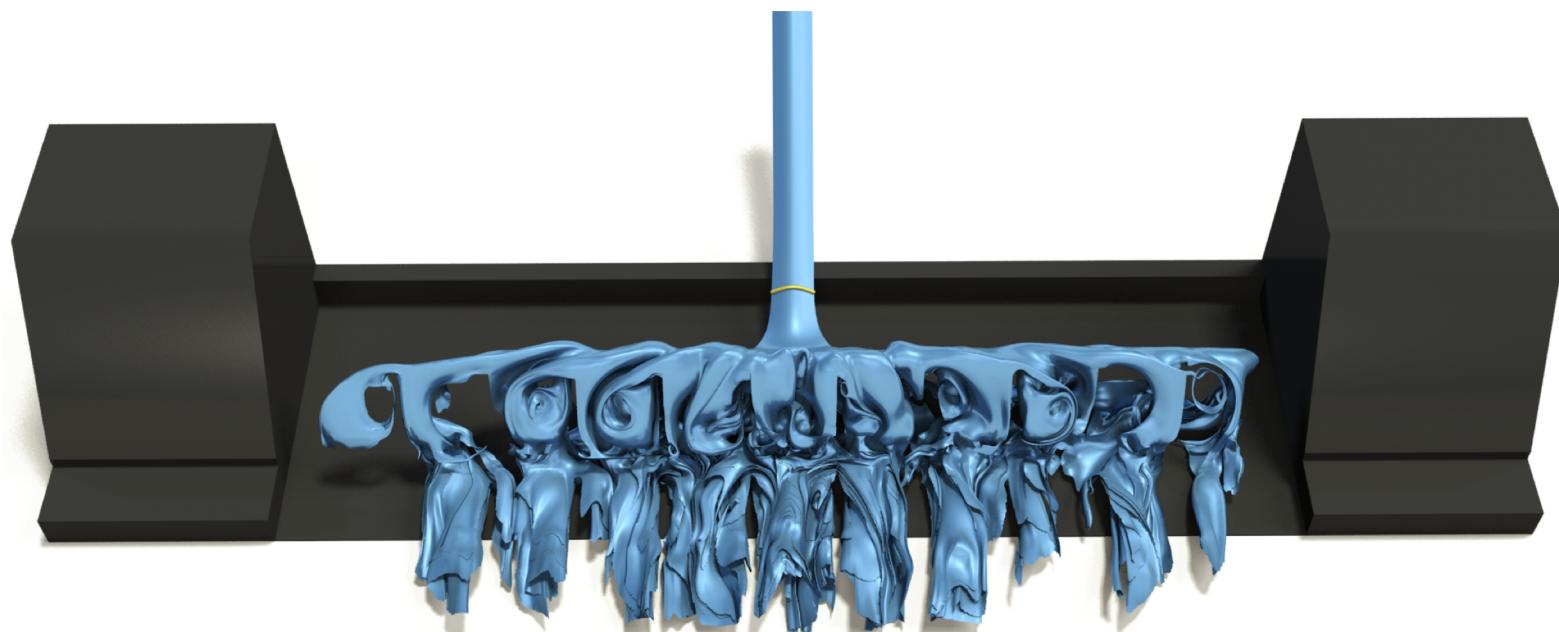


Real Data – squared cylinder



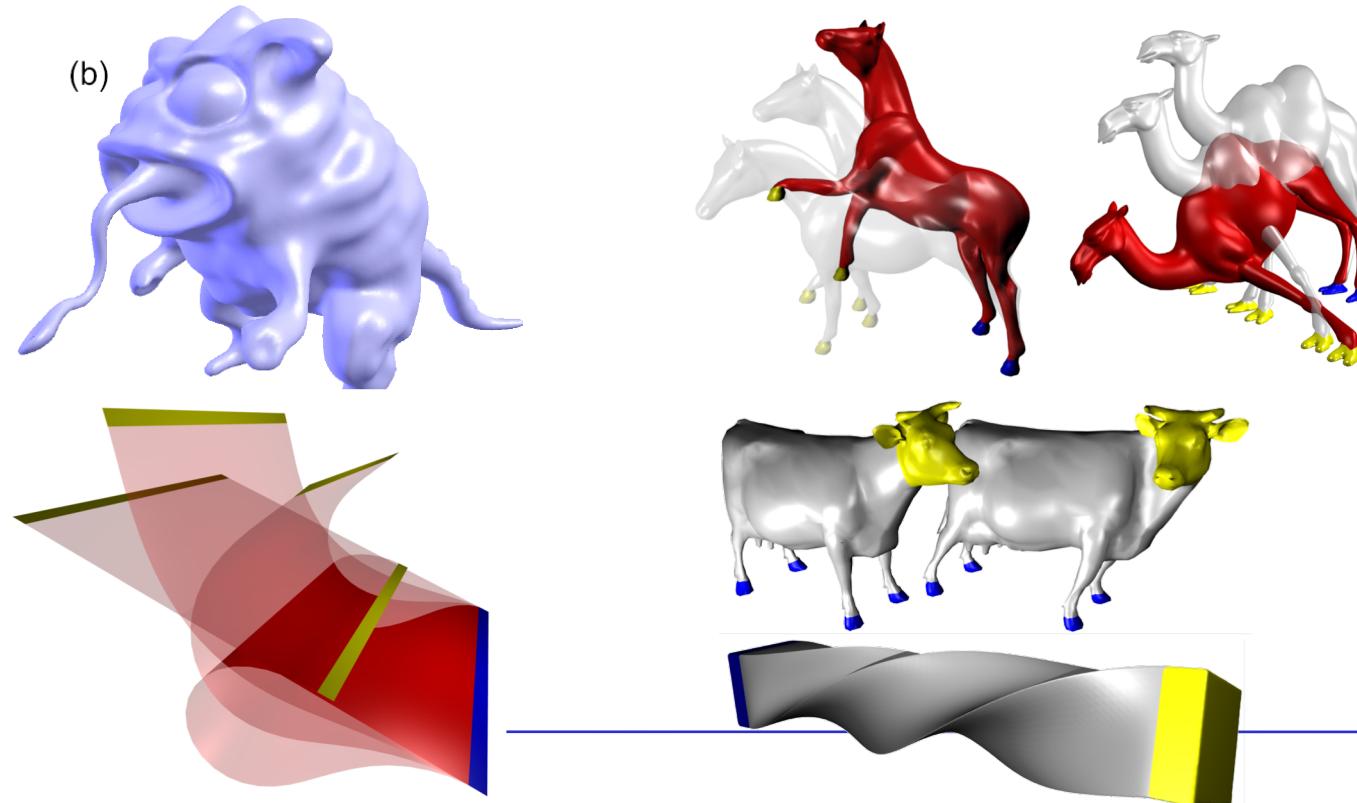


Results: Air Conditioner Outlet Box

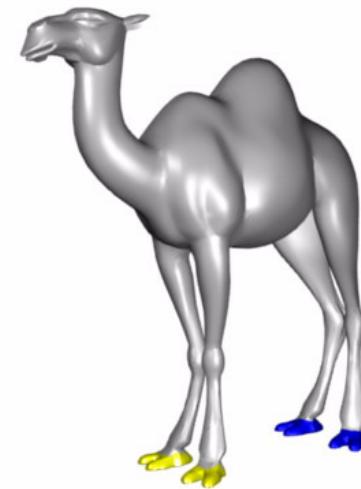
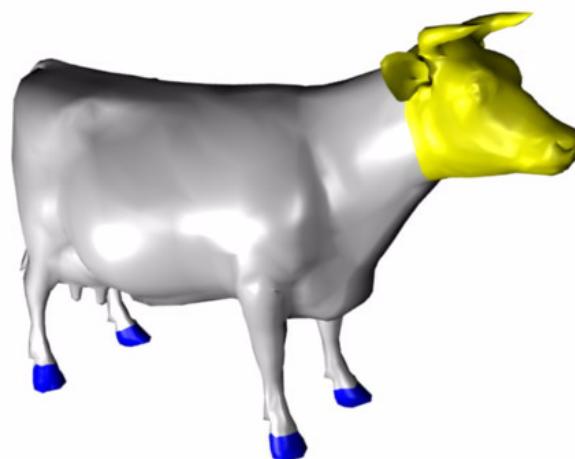




AG Visual Computing: Geometric Modelling, Deformations

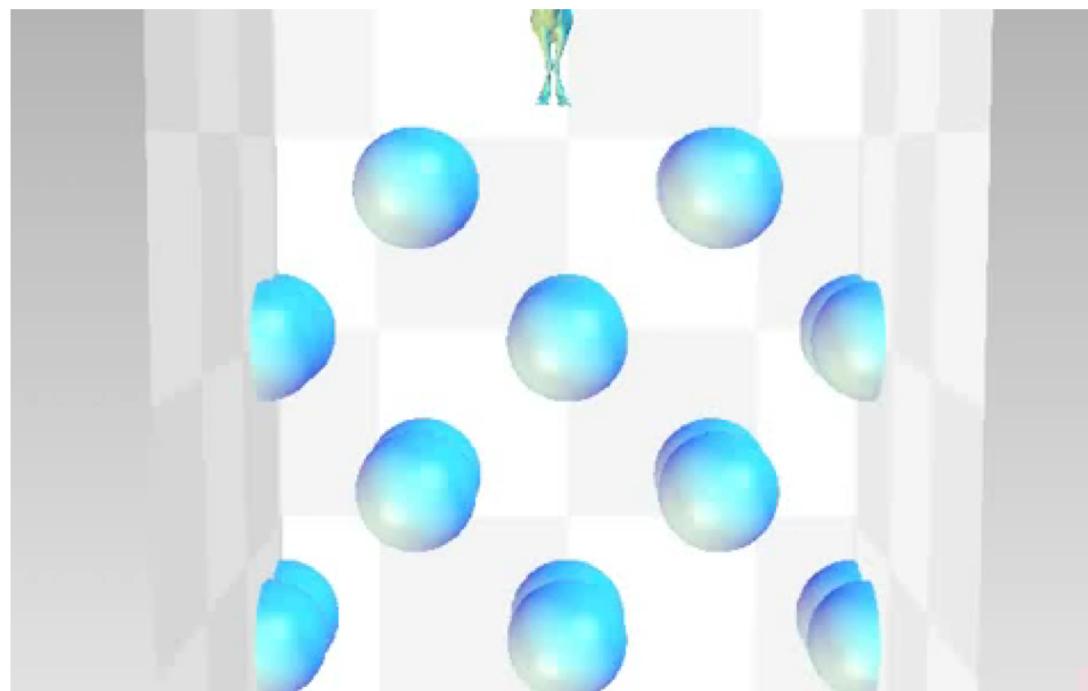


Introduction to Computer Graphics



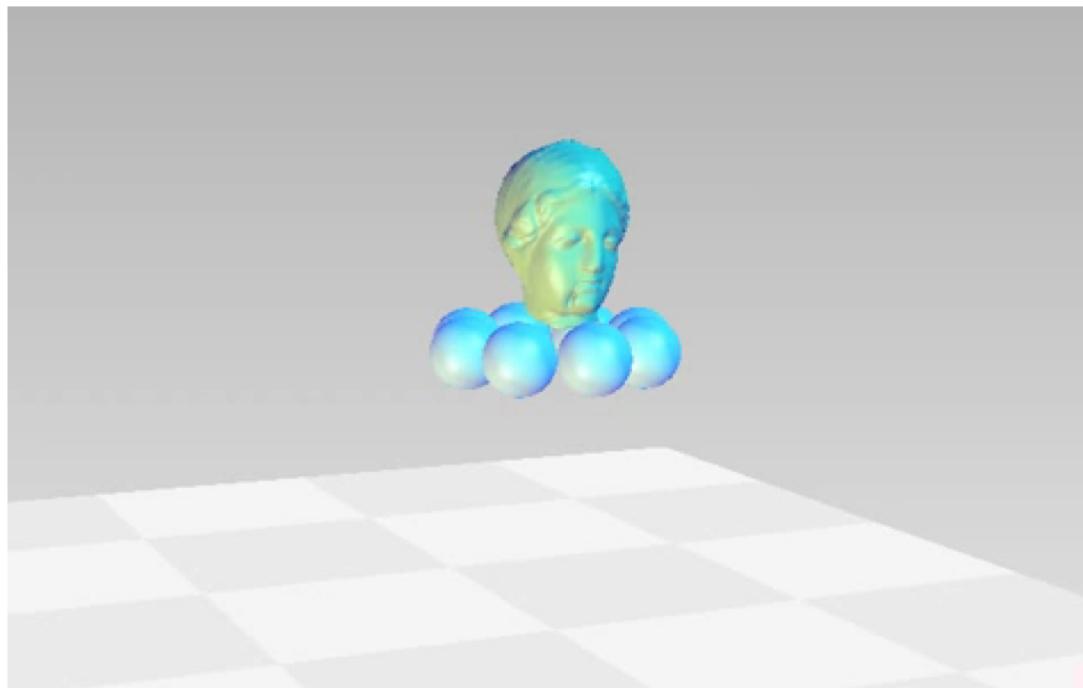


AG Visual Computing: Geometric Modelling, Deformations



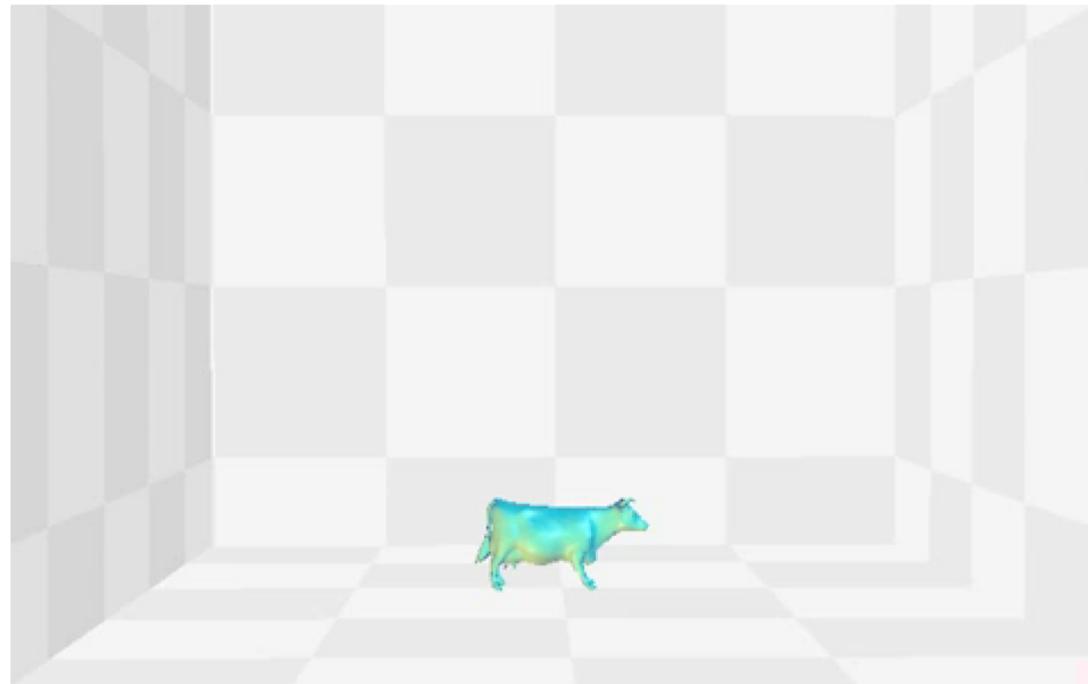


AG Visual Computing: Geometric Modelling, Deformations





AG Visual Computing: Geometric Modelling, Deformations

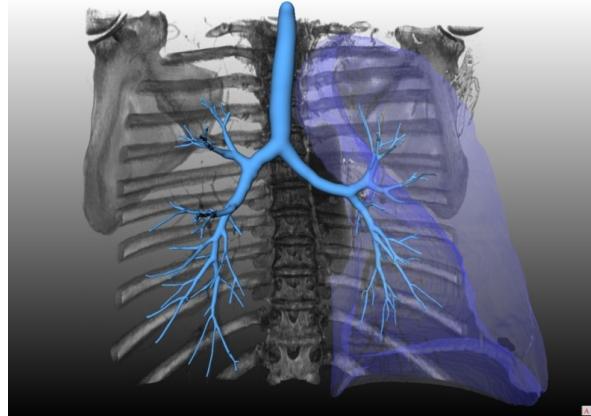


Computergraphik in Magdeburg:

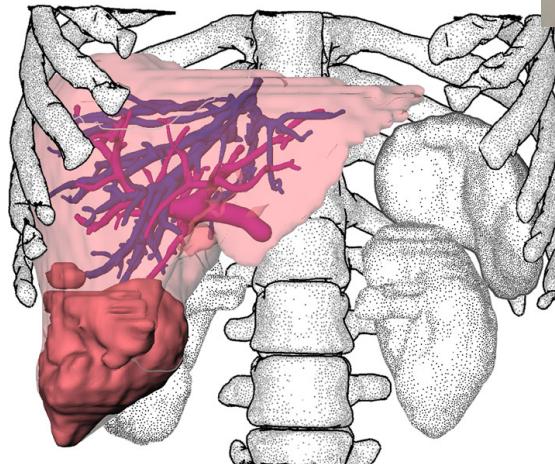


AG Visualisierung (Prof. Preim)

Gefäßvisualisierung und illustrative Visualisierung



Ausgusspräparat der Bronchialgefäße einer menschlichen Lunge. Der linke Lungenlappen ist transparent dargestellt zusammen mit einem Volume Rendering weiterer Gefäße und umgebender Knochen.



Fokus- und Kontextdarstellung durch farblich hervorgehobene Fokusobjekte und Repräsentation von Kontextstrukturen mittels Stippling und Silhouetten.

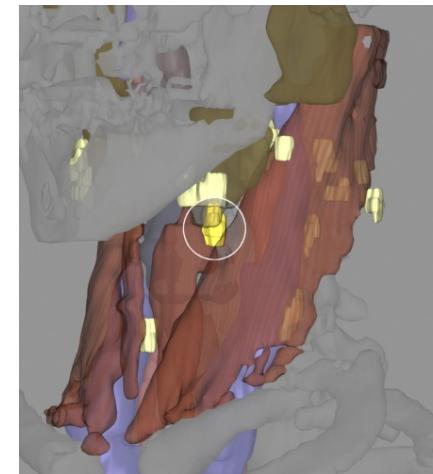
Computergraphik in Magdeburg:



AG Visualisierung (Prof. Preim)
Planung und Training von chirurgischen Eingriffen



Der Nutzer kann in einer 3D-Darstellung eine virtuelle Resektion vornehmen.



Kombination von Farbe, Transparenz-Mapping und lokaler Transparenz um einen einzelnen Hals-Lymphknoten hervorzuheben.

Arbeitsgruppe Computerassistierte Chirurgie



- Augmented Reality zur Unterstützung von Operationen
 - Visualisierungstechniken zur besseren Navigation chirurgischer Instrumente
- Viele Analogien zur Navigation in 3D Computerspielen



Navigierte Leberoperation, Asklepios Klinik Hamburg



Jun. Prof. Dr. Christian Hansen

Computergraphik in Magdeburg:



Fraunhofer-Institut für Fabrikbetrieb
und -automatisierung





1. Einführung

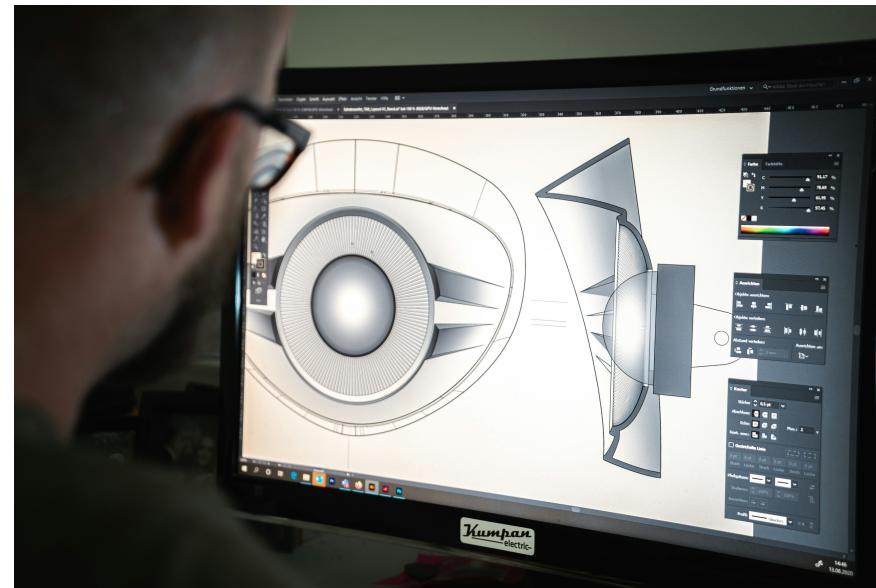
1.3 Anwendungen der Computergraphik

1. Einführung

1.3 Anwendungen der Computergraphik



- Rechnergestützter Entwurf, Konstruktion und Manipulation (CAE)
 - Entwurf und Konstruktion von Produkten - Flugzeuge, Automobile, Häuser, Produktions- / Energieanlagen, etc. (CAD),



1. Einführung

1.3 Anwendungen der Computergraphik



■ Graphik in Steuerungs-, Überwachungs- u. Informations-systemen

- Leitstände, Prozeßüberwachung und - darstellung,
- Meßwertaufnahme u. –aufbereitung,



■ Ausbildung und Training

- Flug- und Fahrsimulatoren,
- militärische Anwendungen;

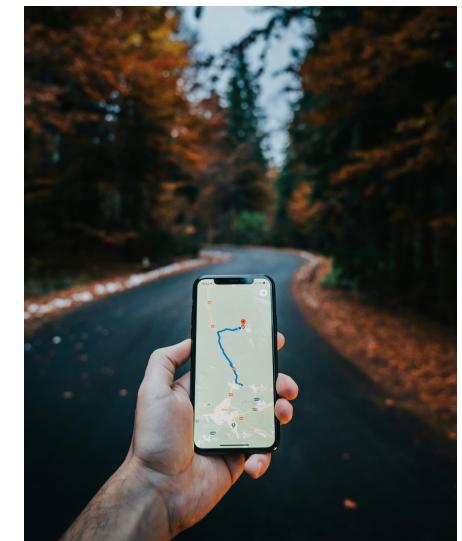


1. Einführung

1.3 Anwendungen der Computergraphik



- Plotsysteme, Zeichnungserstellung, Kartographie
 - Geschäfts-, Technik- und Ingenieurgraphiken,
 - Darstellung geologischer und geographischer Daten, z.B. Landkarten Liegenschaftskarten, Reliefkarten, Wetterkarten, etc.
- Visualisierung von Daten und Informationen
 - Physikalische / chemische Prozesse, Strömungsmechanik,
 - Wetter- / Klimadaten,
 - Medizinische Daten;



1. Einführung

1.3 Anwendungen der Computergraphik



- Kunst, Werbung, Design, Unterrichtsfilme;
Unterhaltungsfilme
Graphik für Multimedia und Internet-Anwendungen,
- Virtual und Augmented Reality,
- Computerspiele !





Raytracing

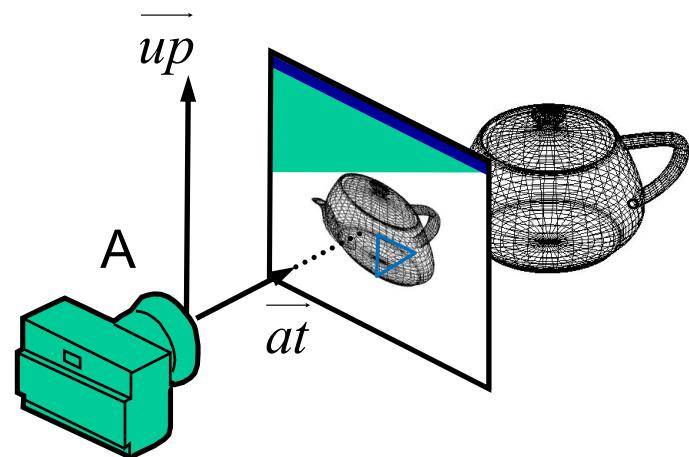
Eine ganz kurze Einführung



-
- In der Computergraphik gibt es 2 grosse Ansätze zur Bilderzeugung (rendering)
 - Projektion und Rasterisierung
 - Raytracing

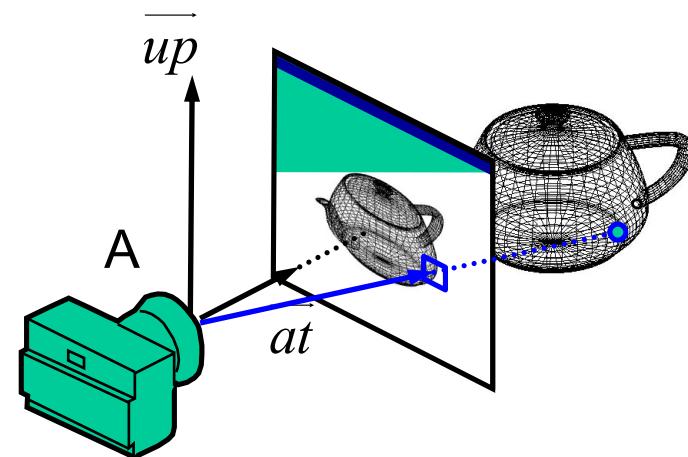


Projektion/Rasterisierung



Die Polygone werden der Reihe nach rasterisiert

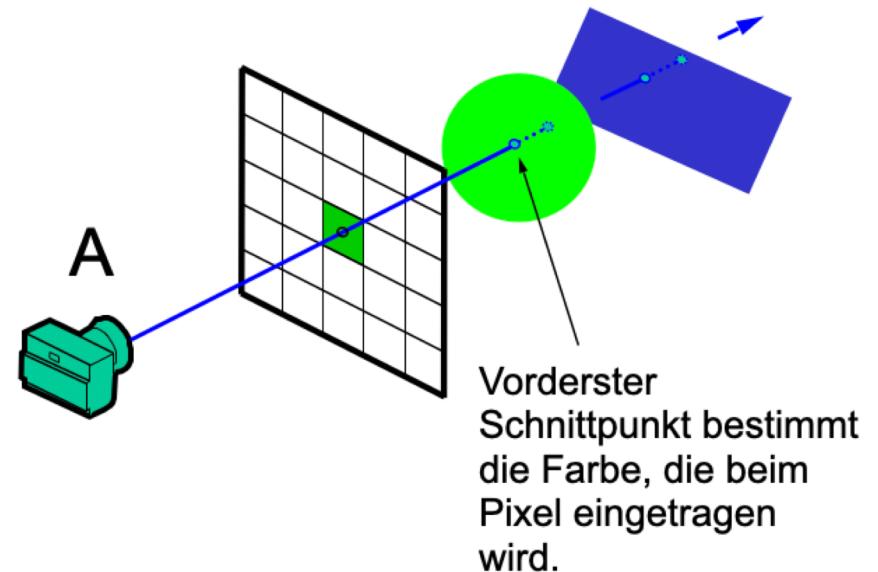
Ray Tracing



Das Bild wird durch Sehstrahlen abgetastet



- ... steht für Strahlverfolgung.
- Man geht in einer Schleife über alle Pixel und berechnet einen Strahl vom Augpunkt über das jeweilige Pixel hinweg.
- Dieser Strahl wird mit der Geometrie der Szene geschnitten und der vorderste Schnittpunkt (das sichtbare Objekt) ermittelt
- Um sicherzustellen, dass es der vorderste Schnittpunkt ist, müssen im Prinzip alle Objekte der Szene getestet werden



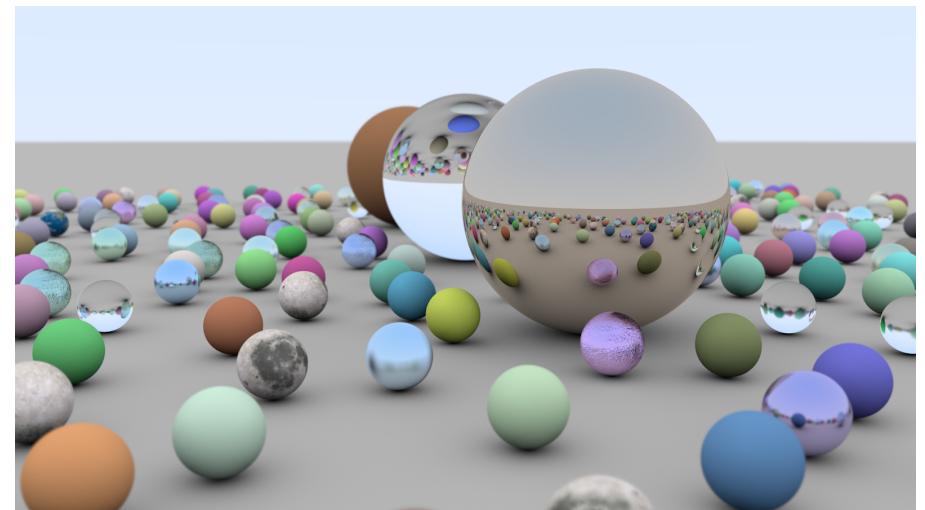
Vorderster
Schnittpunkt bestimmt
die Farbe, die beim
Pixel eingetragen
wird.



Schnitttest Strahl -Kugel

- Kugel in impliziter Beschreibung: $(\mathbf{x}-\mathbf{c})^2 - r^2 = 0$
 - \mathbf{c} : Kugelmittelpunkt
 - r : radius
- Strahl in Punkt-Richtungsgleichung: $\underline{\mathbf{r}}(t) = \mathbf{o} + t \mathbf{d}$
 - $\mathbf{o} = (o_x, o_y, o_z)$
 - $\mathbf{d} = (d_x, d_y, d_z)$
- zu lösen: $(\mathbf{o} + t \mathbf{d} - \mathbf{c})^2 - r^2 = 0$
-> quadratische Gleichung in t

Leicht lösbar mit Standard-Techniken





Schnitttest Strahl Dreieck

- Strahl: $\underline{r}(t) = \mathbf{o} + t \mathbf{d}$
- Dreieck: 3 punkte $\mathbf{a}, \mathbf{b}, \mathbf{c}$
 - \mathbf{x} ist innerhalb des Dreiecks wenn es drei Zahlen u, v, w gibt mit $\mathbf{x} = u \mathbf{a} + v \mathbf{b} + w \mathbf{c}$ und $u+v+w = 1$ und $0 \leq u, v, w \leq 1$
 - u, v, w : baryzentrische Koordinaten of \mathbf{x}
- -> löse $[\mathbf{o} + t \mathbf{d} = u \mathbf{a} + v \mathbf{b} + w \mathbf{c}, u+v+w = 1]$
- -> lineares Gleichungssystem mit den 4 Unbekannten u, v, w, t
- -> i.a. eindeutige Lösung
- -> Strahl trifft Dreieck falls $0 \leq u, v, w \leq 1$ und $0 < t$.



■ Baryzentrische Koordinaten

- gegeben: 3 Punkte $\mathbf{a}, \mathbf{b}, \mathbf{c}$ (nicht-kollinear). Jeder weitere Punkt \mathbf{p} in der Ebene $\mathbf{a}, \mathbf{b}, \mathbf{c}$ kann als Linearkombination von \mathbf{a}, \mathbf{b} and \mathbf{c} beschrieben werden:

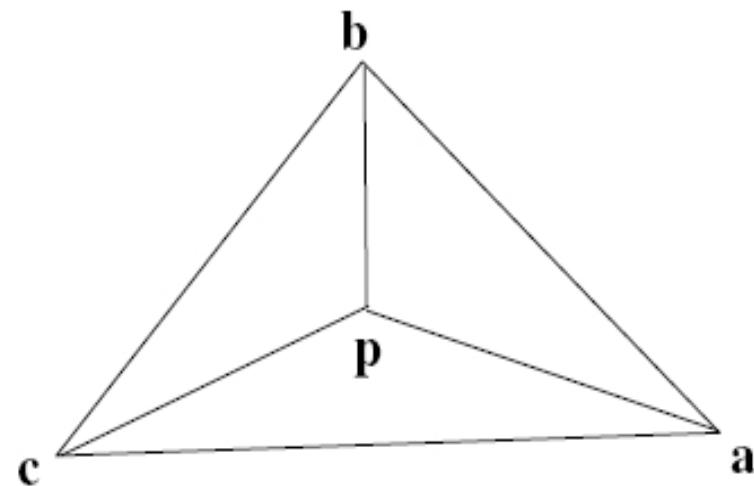
$$\mathbf{p} = u \cdot \mathbf{a} + v \cdot \mathbf{b} + w \cdot \mathbf{c}$$

mit $u + v + w = 1$.

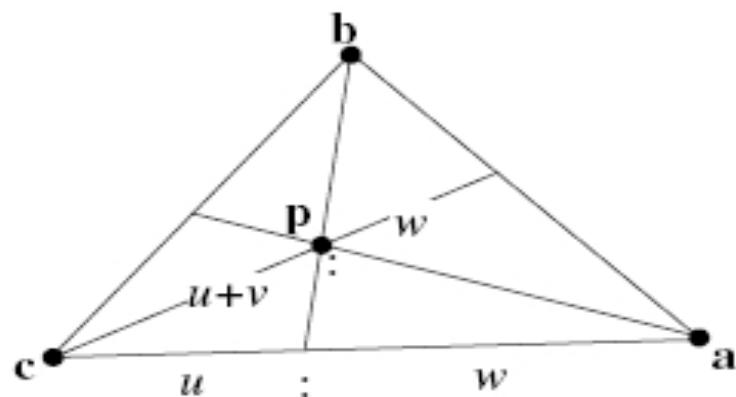
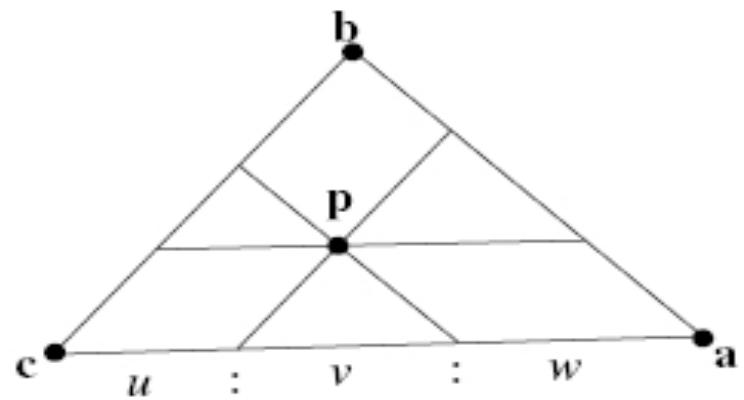
Dann sind, (u, v, w) die baryzentrischen Koordinaten \mathbf{p} relativ zu $\mathbf{a}, \mathbf{b}, \mathbf{c}$.



- $0 \leq u, v, w \leq 1 \Leftrightarrow \mathbf{p}$ ist innerhalb des Dreiecks $\mathbf{a}, \mathbf{b}, \mathbf{c}$

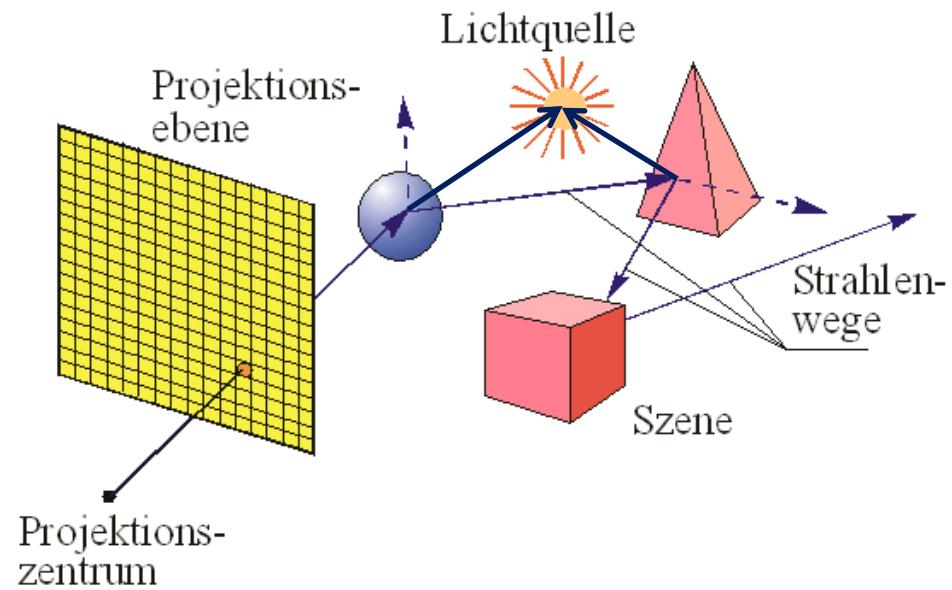


$$u = \frac{\text{area}(\mathbf{p}, \mathbf{b}, \mathbf{c})}{\text{area}(\mathbf{a}, \mathbf{b}, \mathbf{c})} ; \quad v = \frac{\text{area}(\mathbf{p}, \mathbf{a}, \mathbf{c})}{\text{area}(\mathbf{a}, \mathbf{b}, \mathbf{c})} ; \quad w = \frac{\text{area}(\mathbf{p}, \mathbf{a}, \mathbf{b})}{\text{area}(\mathbf{a}, \mathbf{b}, \mathbf{c})}$$



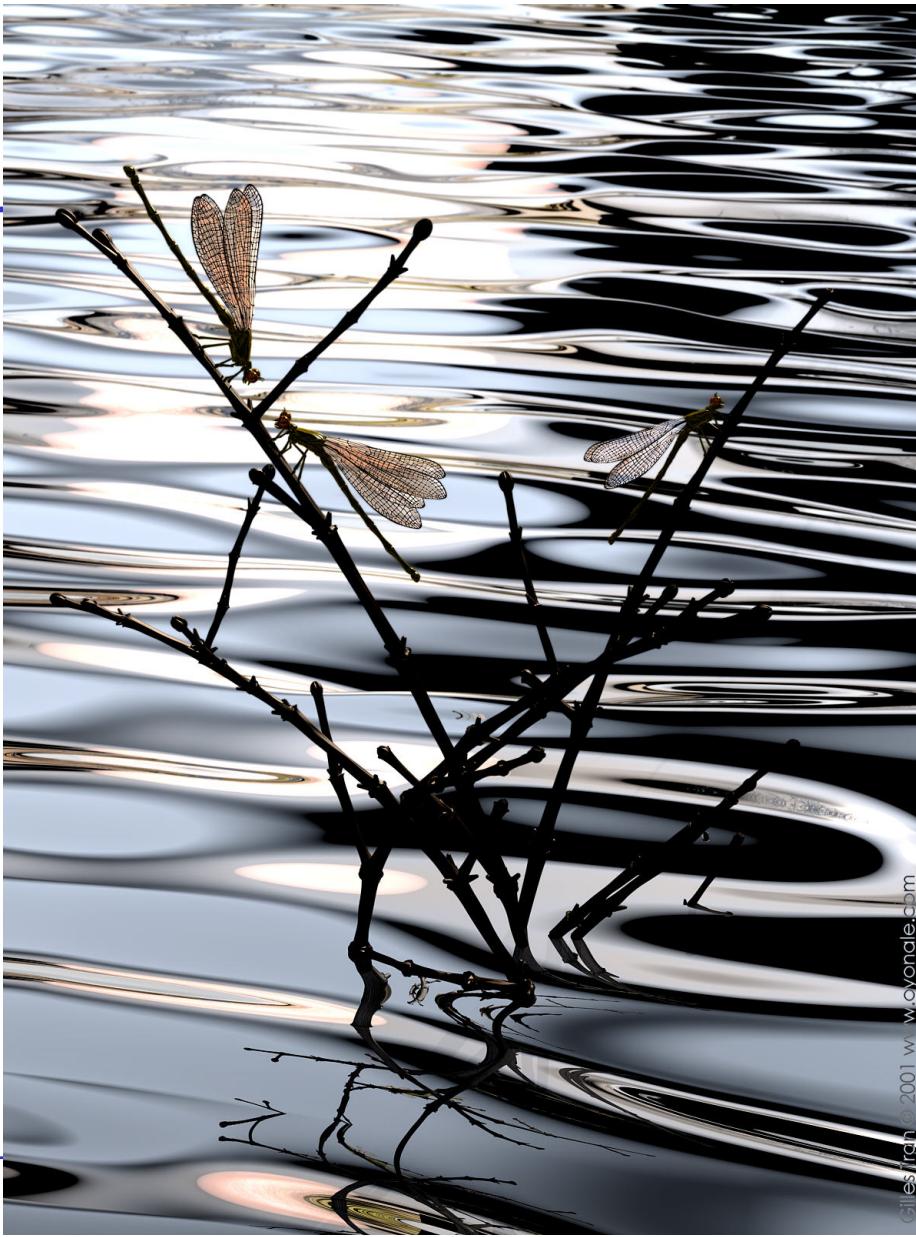


Schatten
Reflexionen
Brechungen
Sichtbarkeit
Schattierung

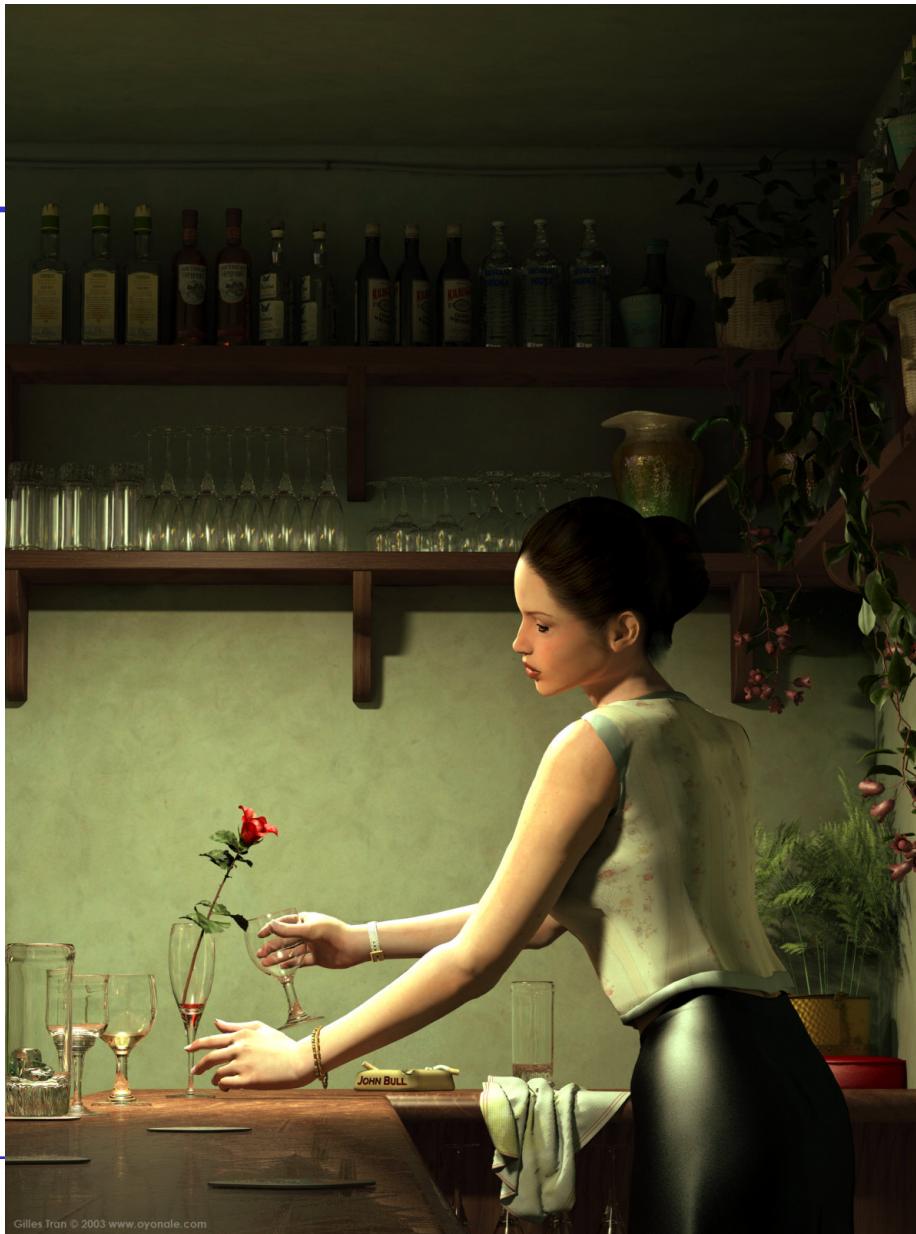




Raytracing-Bild,
Rendered mit PovRay



Gilles irah © 2001 www.oyonale.com





© 2003 Hildur Kolbrún



Gilles Tran © 2000 www.yonale.com

