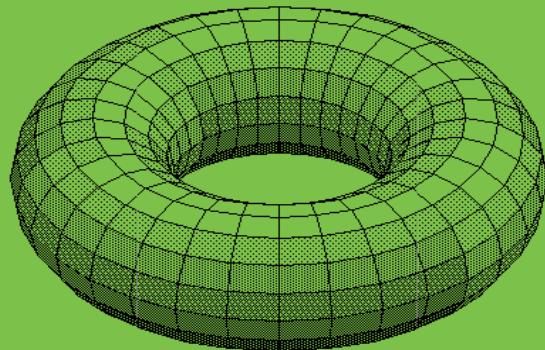


Visual Computing

– Einführung



Yvonne Jung

Organisatorisches

- Vorlesung mit begleitenden Übungen im Rechnerraum
 - Pro Praktikumsgruppe fünf Termine (je 14-tägig)
 - Jeder Termin – eine Abgabe (5 Stück, in Zweiergruppen)
 - Bei Nichterscheinen ist Krankmeldung nötig
 - Lösung muss trotzdem nachgezeigt werden
 - Nur bei erfolgreicher Testierung Klausurteilnahme möglich
 - Jede(r) muss alle Lösungen erklären können
 - Bewertung: erfolgreich / nicht erfolgreich
- Prüfung: Klausur
 - Hilfsmittel: ein zweiseitig selbst handschriftlich beschriebenes DIN-A4-Blatt



Literaturvorschläge

- Alfred Nischwitz, Max Fischer, Peter Haberäcker, Gudrun Socher: *Computergrafik und Bildverarbeitung. Band 1: Computergrafik. Band 2: Bildverarbeitung.* Springer, Wiesbaden 2019
- Thomas Möller, Eric Haines, Naty Hoffman: *Real-Time Rendering.* 4th Edition. Taylor & Francis Ltd., 2018
- Eric Lengyel: *Mathematics for 3D Game Programming and Computer Graphics.* 3rd Ed., Cengage Learning, Boston, MA, 2012
- Dave Shreiner, Graham Sellers, John Kessenich: *OpenGL Programming Guide: The Official Guide to Learning OpenGL.* 9th Ed., Addison Wesley, 2016
- Matt Pharr, Wenzel Jakob, Greg Humphreys: *Physically Based Rendering: From Theory To Implementation.* 4th Ed. The MIT Press, 2023
- J. Hughes, A. van Dam, M. McGuire, D. Sklar, J. Foley, S. Feiner, K. Akeley: *Computer Graphics: Principles and Practice,* 3rd Ed. Addison-Wesley, 2013
- Richard Szeliski: *Computer Vision: Algorithms and Applications.* 2nd Ed. Springer, 2022

Um was geht es hier eigentlich?

...meistens um Pixel

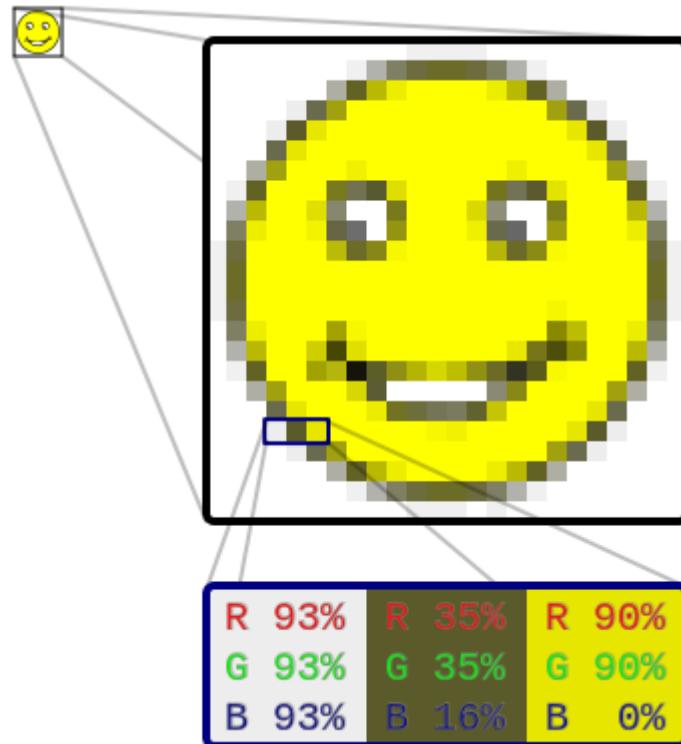
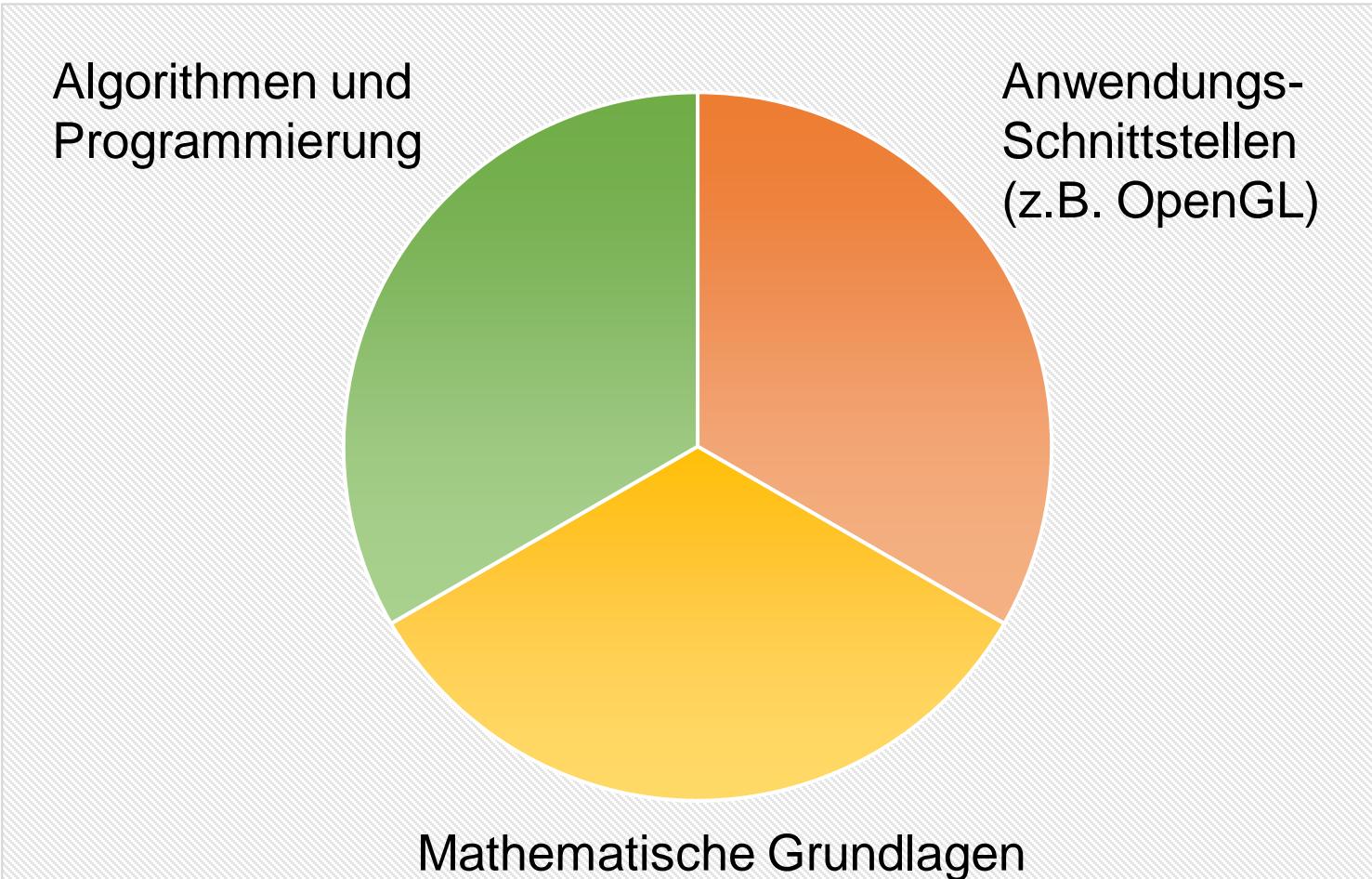


Bild : <https://developer.tizen.org/forums/native-application-development/what-rasterisation-opengl-graphic-pipeline>

- Computergrafik
 - Von der Idee einer virtuellen Szene zu Pixeln
- Bildverarbeitung
 - Pixel auf Informationen hin analysieren u.ä.

Computergraphik (CG)



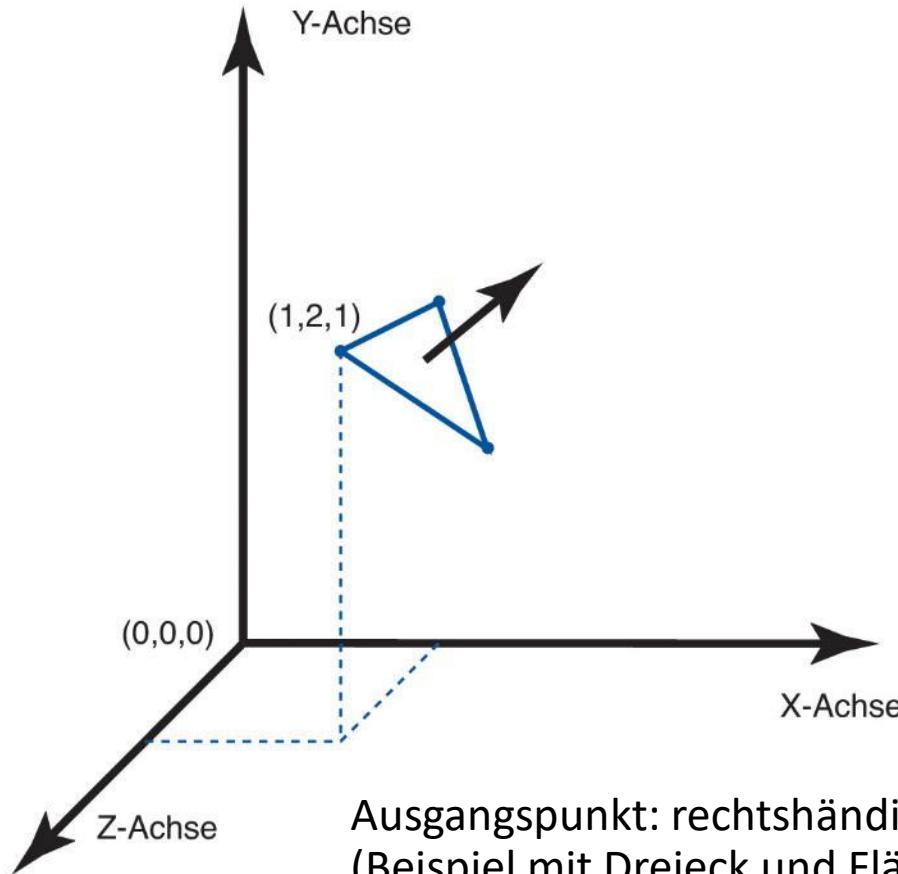
Computergraphik

- Image generation + manipulation with computers
 - Main areas are modeling, animation, rendering
 - ...about digital models for three dimensional geometric objects and images. Research goals are the generation of plausible, informative images, and computation with reasonable resources. This combines knowledge from different areas of mathematics and computer science.
- ...the visual manifestation of mathematics.

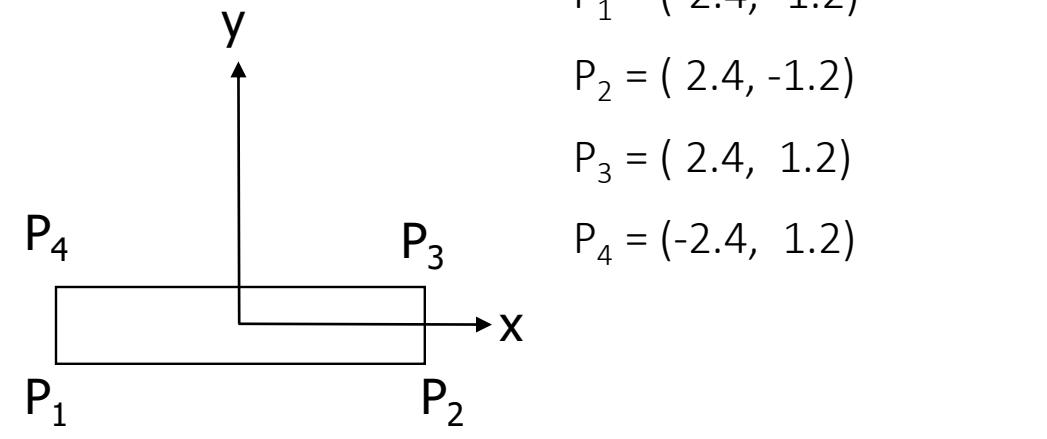
[Slightly shortened and taken from <http://www.cg.tu-berlin.de/>]

[Dave Shreiner, SIGGRAPH 2014 Conference Chair and author of several OpenGL books]

Einschub: Koordinatensysteme



Ausgangspunkt: rechtshändiges dreidimensionales kartesisches Koordinatensystem
(Beispiel mit Dreieck und Flächennormale)



Definition eines Rechtecks in kartesischen 2D-Koordinaten

$$P_1 = (-2.4, -1.2)$$

$$P_2 = (2.4, -1.2)$$

$$P_3 = (2.4, 1.2)$$

$$P_4 = (-2.4, 1.2)$$

Erzeugung graphischer Objekte

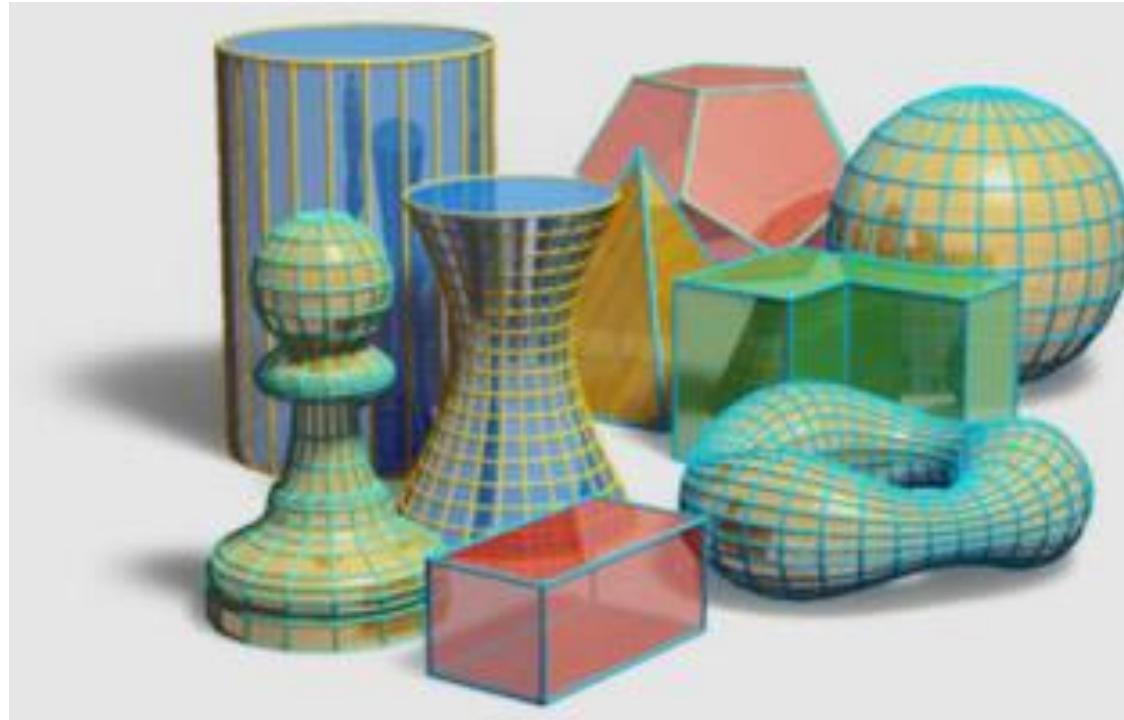
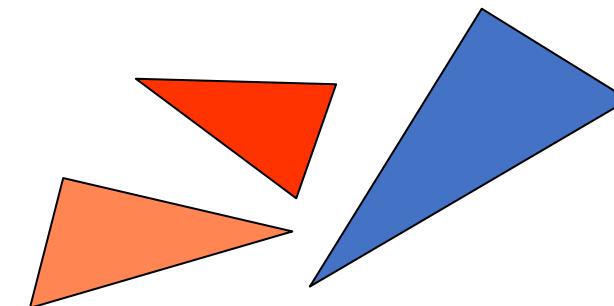
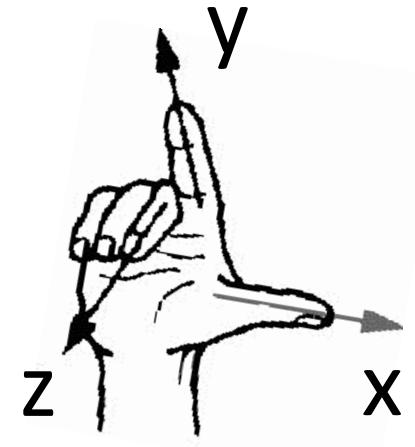
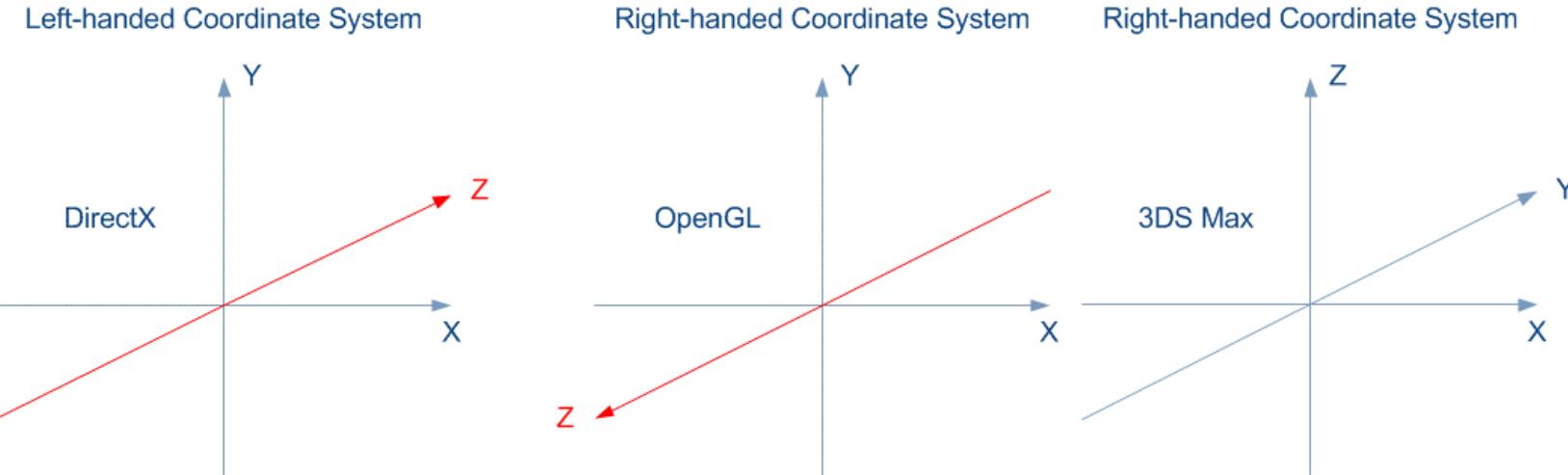


Bild entnommen aus: http://www.pytha.de/produkt/modeler_1.de.php

- Alle grafischen Objekte bestehen i.d.R. aus Punkten, die mit Kanten zu Dreiecken verbunden werden
- Jede Geometrie wird durch Punkte, Kanten und Flächen in einem Koordinatensystem definiert



Händigkeit von Koordinatensystemen



- Moderne Grafik-APIs unterstützen prinzipiell jedes System
 - Aber Vorsicht mit verwendeten Bibliotheken bzw. 3D-Daten!
- In Spielen wird z.B. gerne 3DS Max Koordinatensystem verwendet

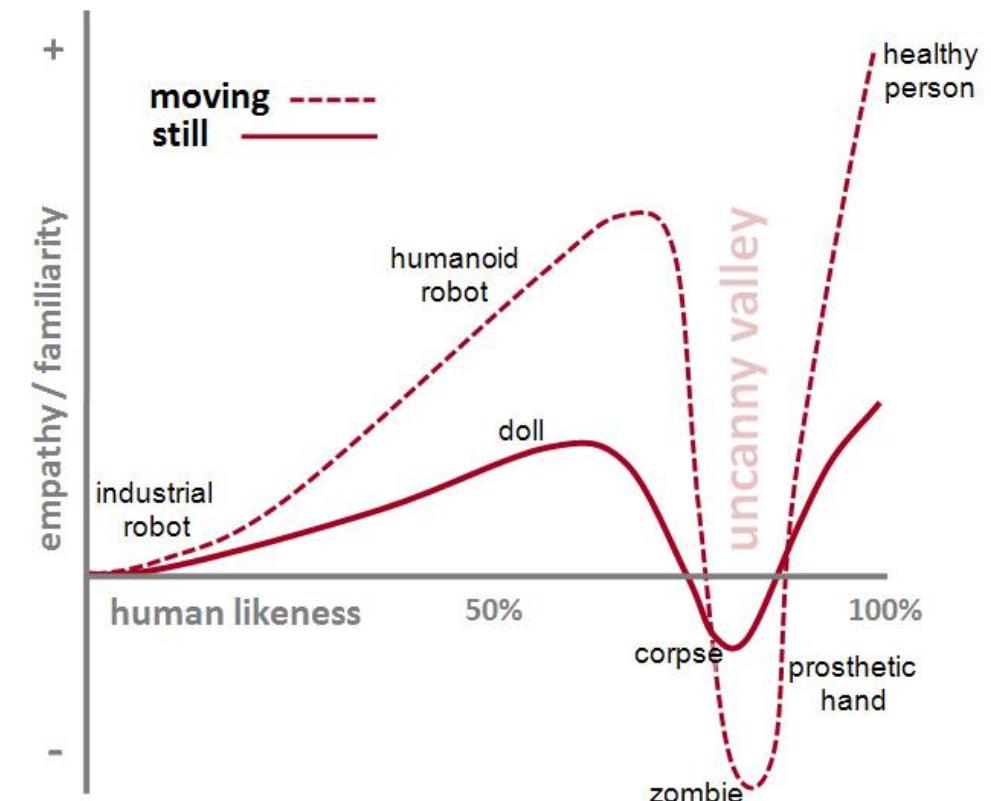
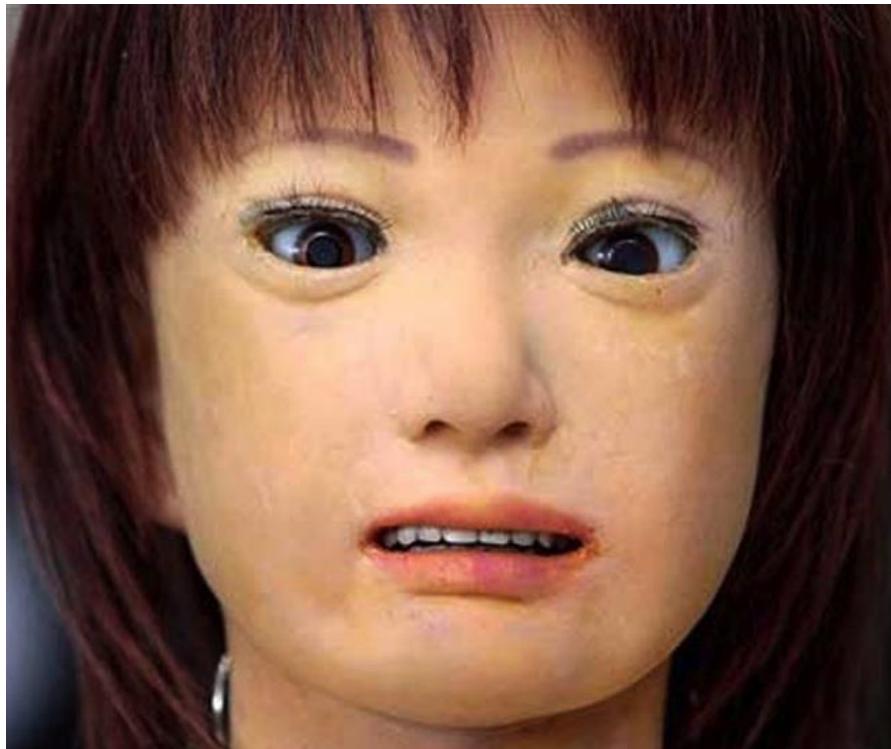
Bahnbrechendes Rendering im Film



- Durch moderne Grafikkarten und GPU-optimierte Verfahren ist ähnliche Qualität z.T. bereits in Computerspielen möglich
- Einige Engines:
 - CryEngine
 - Unreal Engine
 - Unity
 - Id Tech

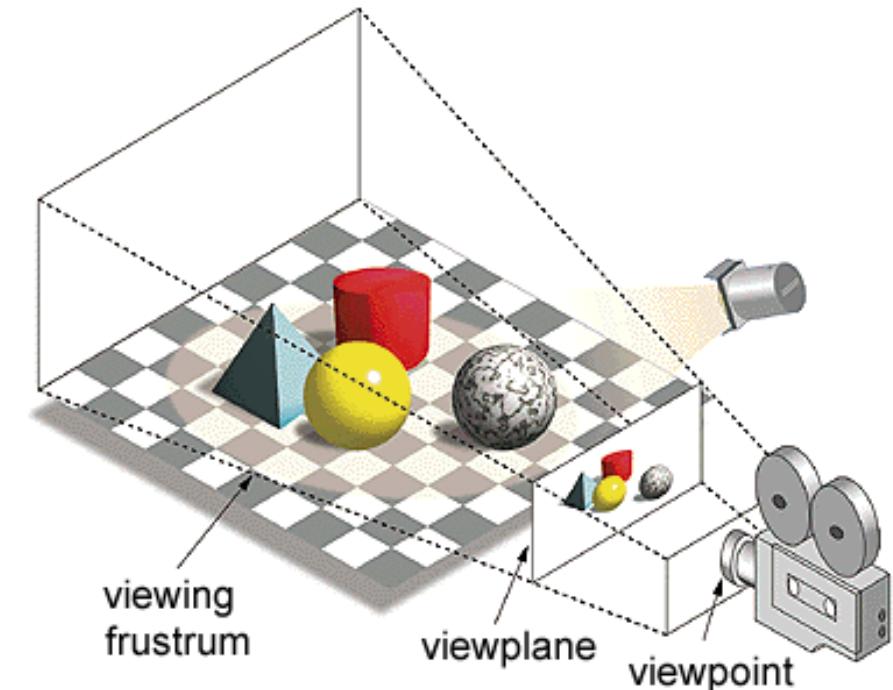
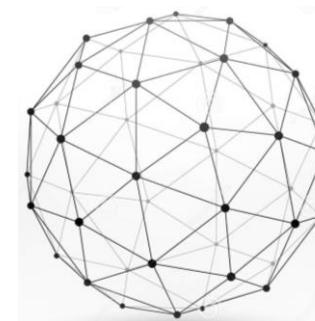
Vorsicht bei virtuellen Charakteren

- „Uncanny Valley“ (Masahiro Mori, 1970)
 - Wirken bei "Fast-Realismus" gruselig



3D-Computergraphik

- Modellierung des Bildinhaltes
 - Dreidimensionale Geometrien
 - Formen der 3D-Objekte in Szene werden durch dreidimensionale Dreiecksgitter angenähert
 - Lichtquellen
 - Virtuelle Kamera (Viewpoint)
 - Sichtbarer Bereich (Viewing Frustum)
 - Position der Bildebene (Viewplane)
- Ziel: Pixel auf Bildebene anhand mit virt. Kamera „aufgenommener“ 3D-Szene korrekt einzufärben



Einschub: Rastergraphik

Das **Generieren einer Rastergrafik** ist in der 3D-Echtzeitgraphik die gebräuchlichste Möglichkeit, um vom Dreiecksnetz zur gerenderten Darstellung zu kommen (ist ein sog. „lokales“ Verfahren)

1. Eckpunkte des Dreiecks (Vertices) auf Pixel der Bildebene (Viewplane) abbilden

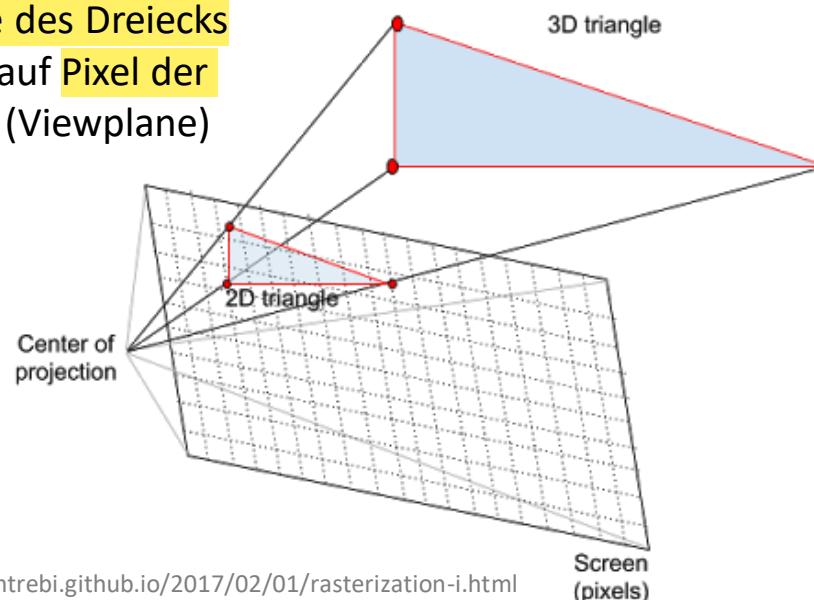


Bild aus: <https://mtrebi.github.io/2017/02/01/rasterization-i.html>

2. Anhand der Pixel der Eckpunkte des Dreiecks die restlichen Dreieckspixel in der Bildebene bestimmen und einfärben

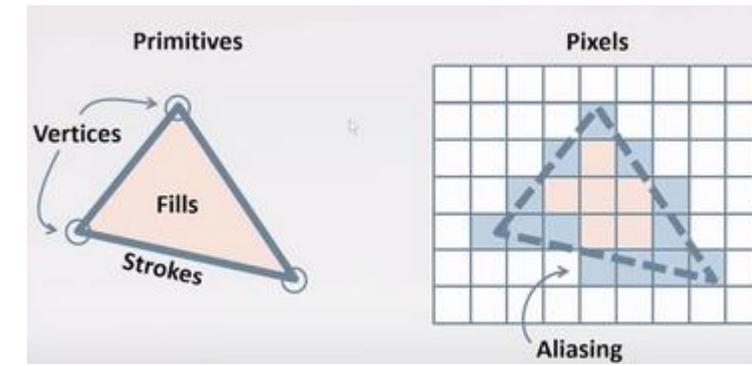


Bild aus: <https://developer.tizen.org/forums/native-application-development/what-rasterisation-opengl-graphic-pipeline>

Exkurs: Global Illumination

Das Generieren eines Bildes mit **Raytracing** oder **Pathtracing** ist eine andere Möglichkeit, um vom Dreiecksnetz zur gerenderten Darstellung zu kommen (eher im Offline-Rendering üblich)

Statt von den Eckpunkten aus die Darstellung der Geometrien zu erzeugen, versucht man die Szene mit „Sehstrahlen“, die von der virtuellen Kamera ausgehen und reflektieren, darzustellen

Sog. „globales“ Verfahren, da je gesamte Szene betrachtet wird

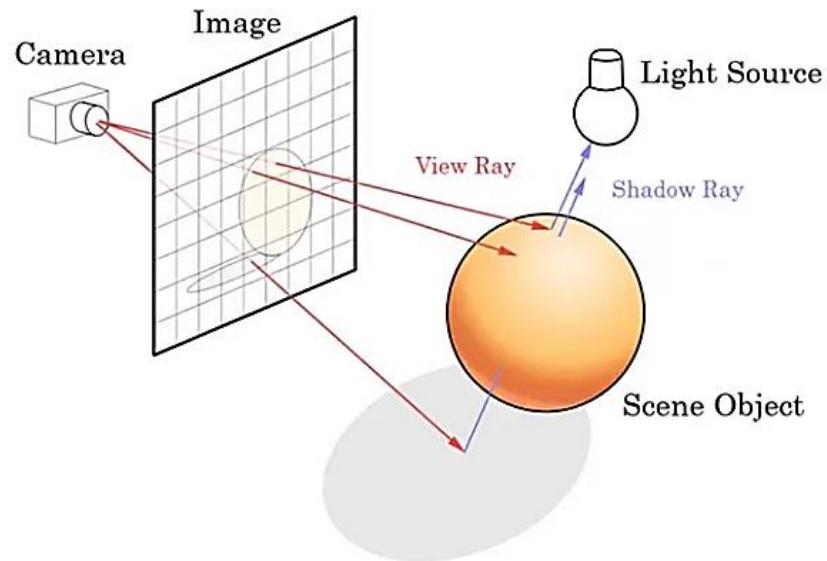
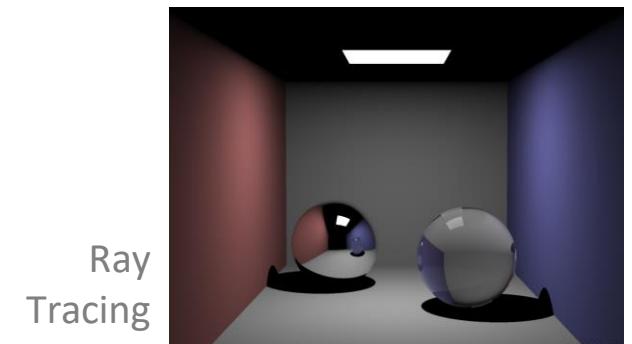
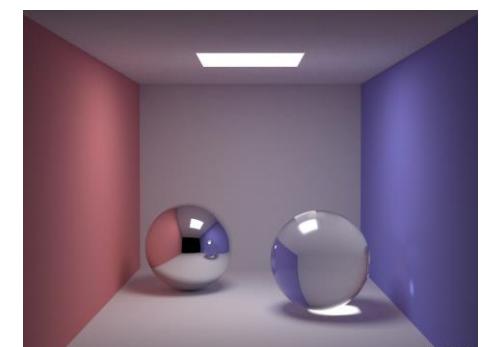


Bild aus: <https://medium.com/@junyingw/future-of-gaming-rasterization-vs-ray-tracing-vs-path-tracing-32b334510f1f>



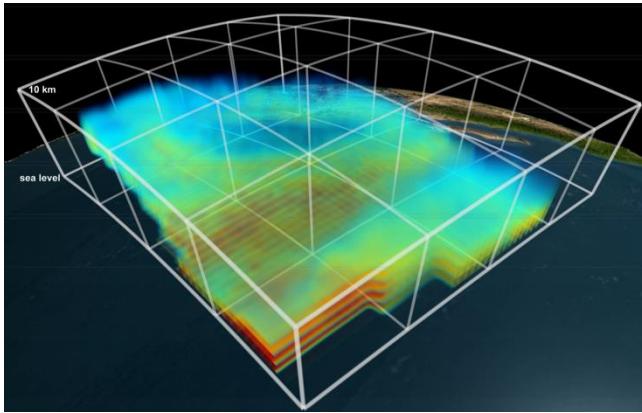
Ray
Tracing



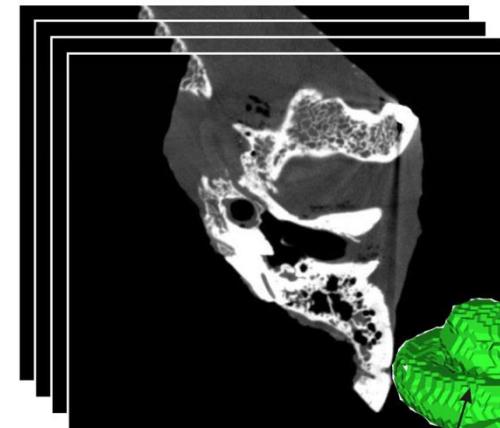
Path
Tracing

Erzeugung von 3D-Modellen

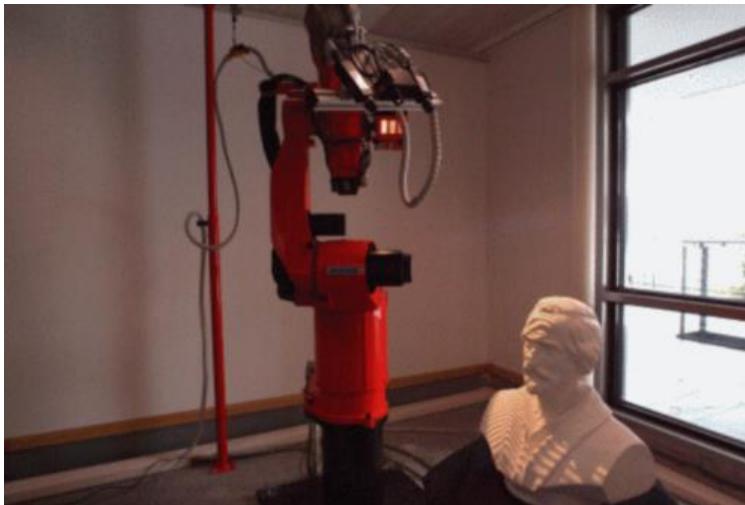
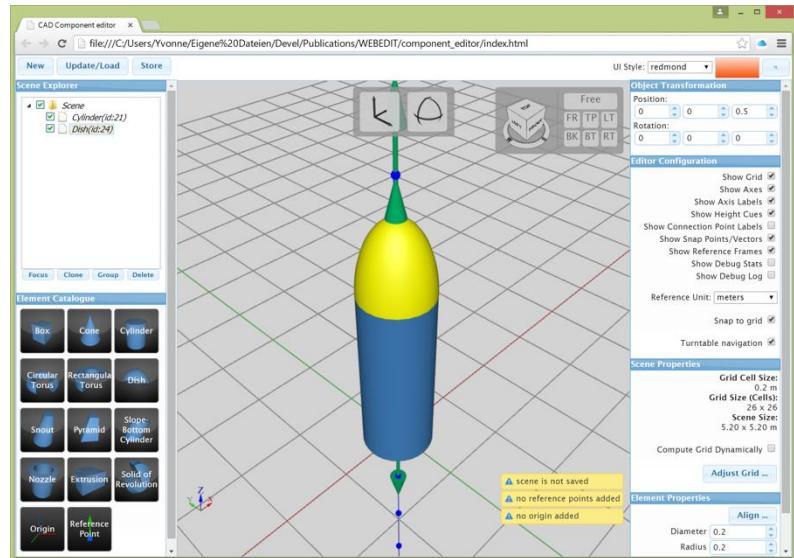
Messung von (Wetter-)Daten
Bildgebende Verfahren (CT u.ä.)
Computer Aided Design (CAD)
3D-Rekonstruktion/-Scanner



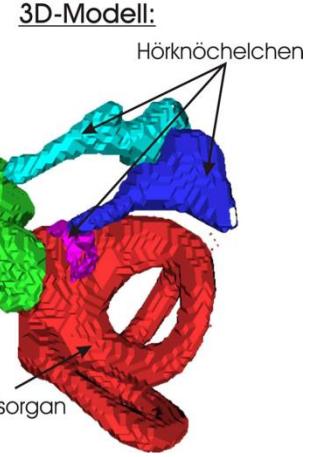
CT-Schichtbilder:



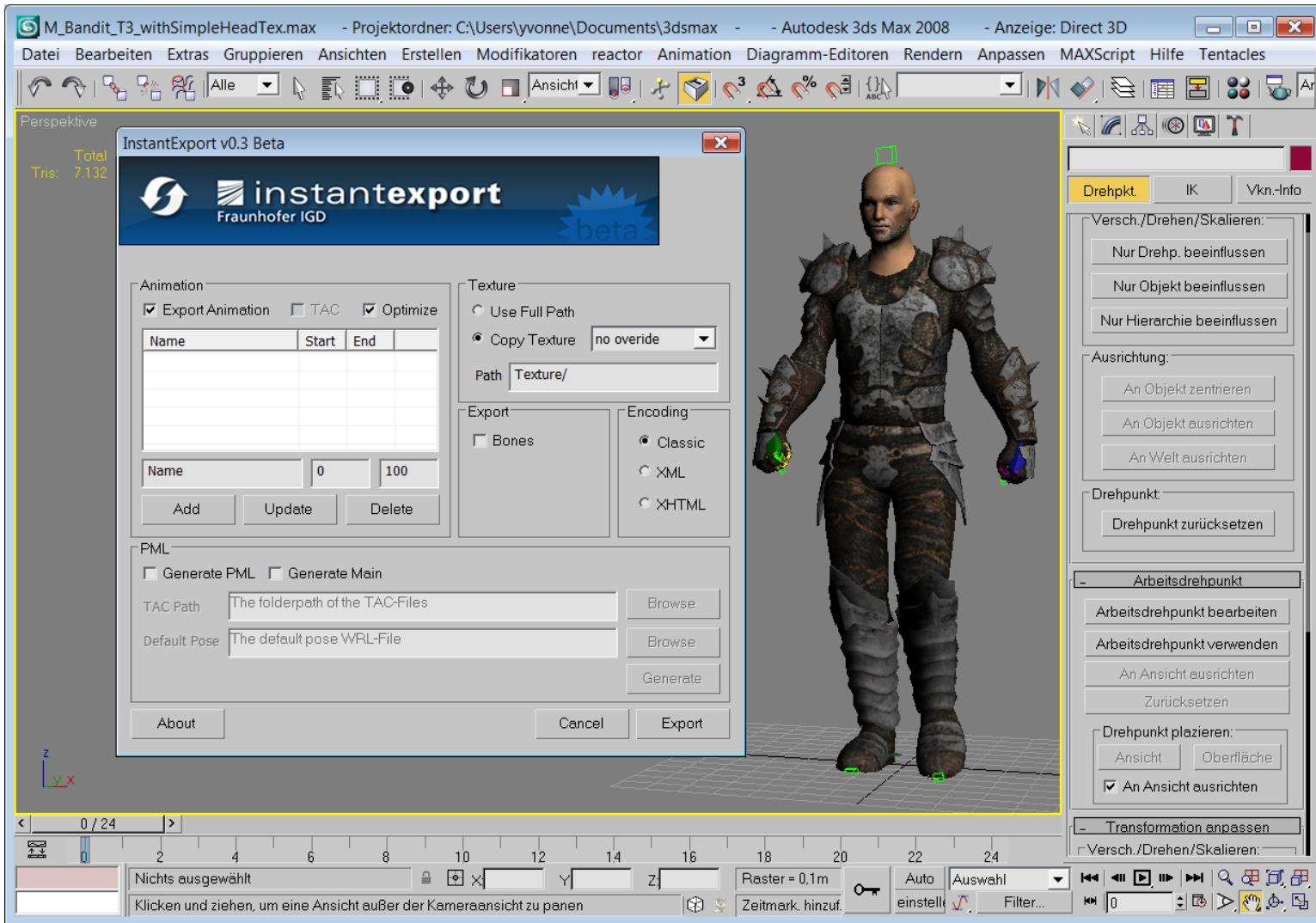
3D-Modell:
Hörknöchelchen



Hörschnecke
(Cochlear)
Gleichgewichtsorgan



3D-Modellierung (→ Artists)



... vs. Toolprogrammierung!

The screenshot shows the Microsoft Visual Studio interface with the following details:

- Title Bar:** 3dsMax_InstantExport - Microsoft Visual Studio
- Menu Bar:** DATEI, BEARbeiten, ANSICHT, QTs, PROJEKT, ERSTELLEN, DEBUGGEN, TEAM, EXTRAS, TEST, ARCHITEKTUR, ANALYSIEREN, FENSTER, HILFE
- Toolbars:** Standard, Debugging, Task List, Solution Explorer, Properties, Task List, Status Bar.
- Code Editor:** Displays the `Node_Geometry.cpp` file. The code implements a `parse` method for a `IGameNode` object, handling geometry and normal data.
- Project Explorer:** Shows the project structure for `3dsMax_InstantExport`, including source files like `DlgOptions.cpp`, `DLLEntry.cpp`, and `Node_Geometry.cpp`.
- Properties Explorer:** Shows properties for the `Node_Geometry.cpp` file, including Name: `Node_Geometry.cpp`, Dateityp: `C/C++-Code`, Inhalt: `False`, Relativer Pfad: `..\unicode\Node_Geometry`, Vollständiger Pfad: `C:\Users\Yvonne\Documents\3dsMax_InstantExport\..\unicode\Node_Geometry\Node_Geometry.cpp`, and Zu Projekt hinzufügen: `True`.
- Ausgabe (Output):** Shows build logs for cleaning and building the project.
- Status Bar:** Bereit, Z 52, S 1, Zei 1, EINFG.

Beispiel-Problem: Ozean Rendern

Math and physics

Find theoretical model that is practical

Based on fluid dynamics? Maybe simple fake works, too?

Review optical properties of water, etc.

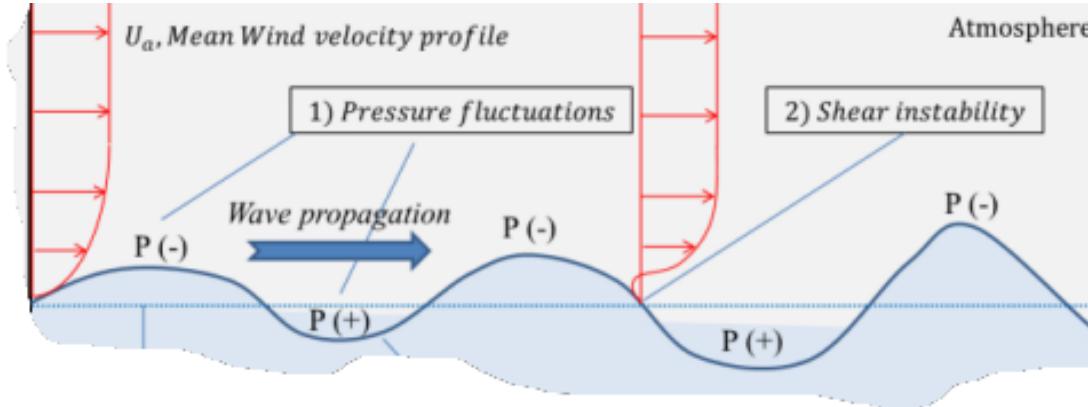
$$f(x) = a_0 + \text{smoothstep}(x) \cdot (a_1 - a_0) \text{ for } 0 \leq x \leq 1$$

$$c = \sqrt{\frac{g\lambda}{2\pi} \tanh\left(\frac{2\pi d}{\lambda}\right)}$$

$$E = \frac{1}{8} \rho g H^2 = \frac{1}{2} \rho g a^2.$$



Beispiel-Problem: Ozean Rendern

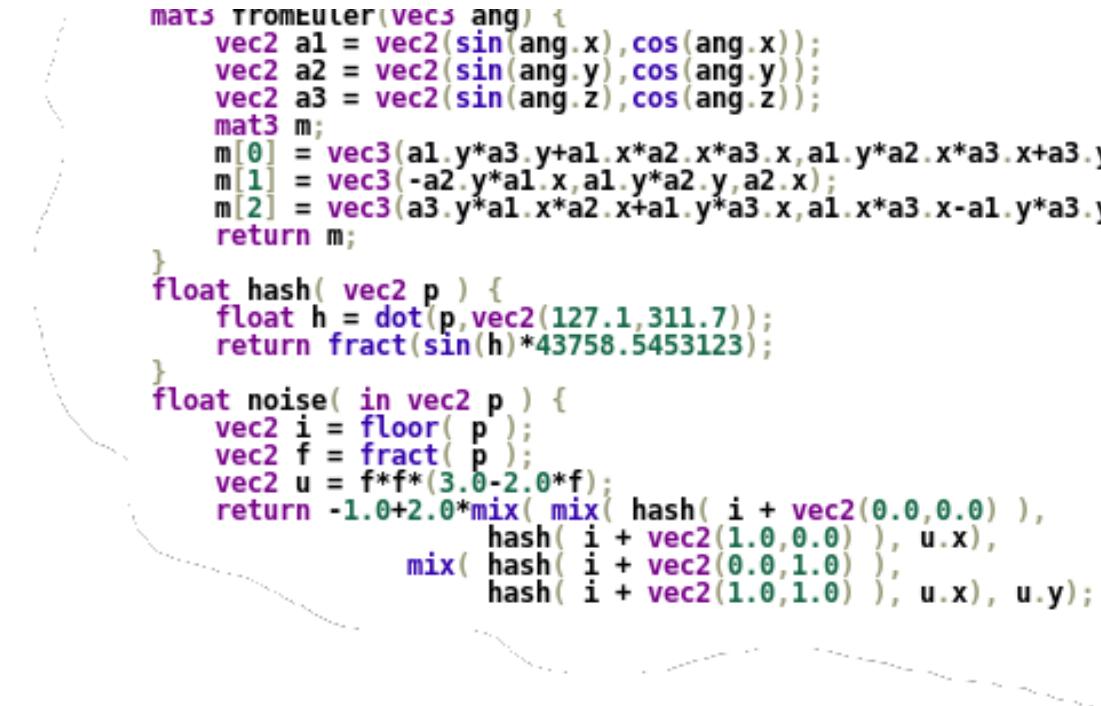


Modeling, Animation, Rendering

Find computer graphics model that is practical

What geometric representation? How to animate? How to shade?

Beispiel-Problem: Ozean Rendern



```
mat3 rotiereLer(vec3 ang) {
    vec2 a1 = vec2(sin(ang.x),cos(ang.x));
    vec2 a2 = vec2(sin(ang.y),cos(ang.y));
    vec2 a3 = vec2(sin(ang.z),cos(ang.z));
    mat3 m;
    m[0] = vec3(a1.y*a3.y+a1.x*a2.x*a3.x,a1.y*a2.x*a3.x+a3.y*a1.x, a1.y*a2.y*a3.y);
    m[1] = vec3(-a2.y*a1.x,a1.y*a2.y,a2.x);
    m[2] = vec3(a3.y*a1.x*a2.x+a1.y*a3.x,a1.x*a3.x-a1.y*a3.y*a2.x, a1.y*a3.y*a2.x);
    return m;
}
float hash( vec2 p ) {
    float h = dot(p,vec2(127.1,311.7));
    return fract(sin(h)*43758.5453123);
}
float noise( in vec2 p ) {
    vec2 i = floor( p );
    vec2 f = fract( p );
    vec2 u = f*f*(3.0-2.0*f);
    return -1.0+2.0*mix( mix( hash( i + vec2(0.0,0.0) ),
        hash( i + vec2(1.0,0.0) ), u.x ),
        mix( hash( i + vec2(0.0,1.0) ),
        hash( i + vec2(1.0,1.0) ), u.x ), u.y );
}
```

Software engineering

Find implementation that is practical

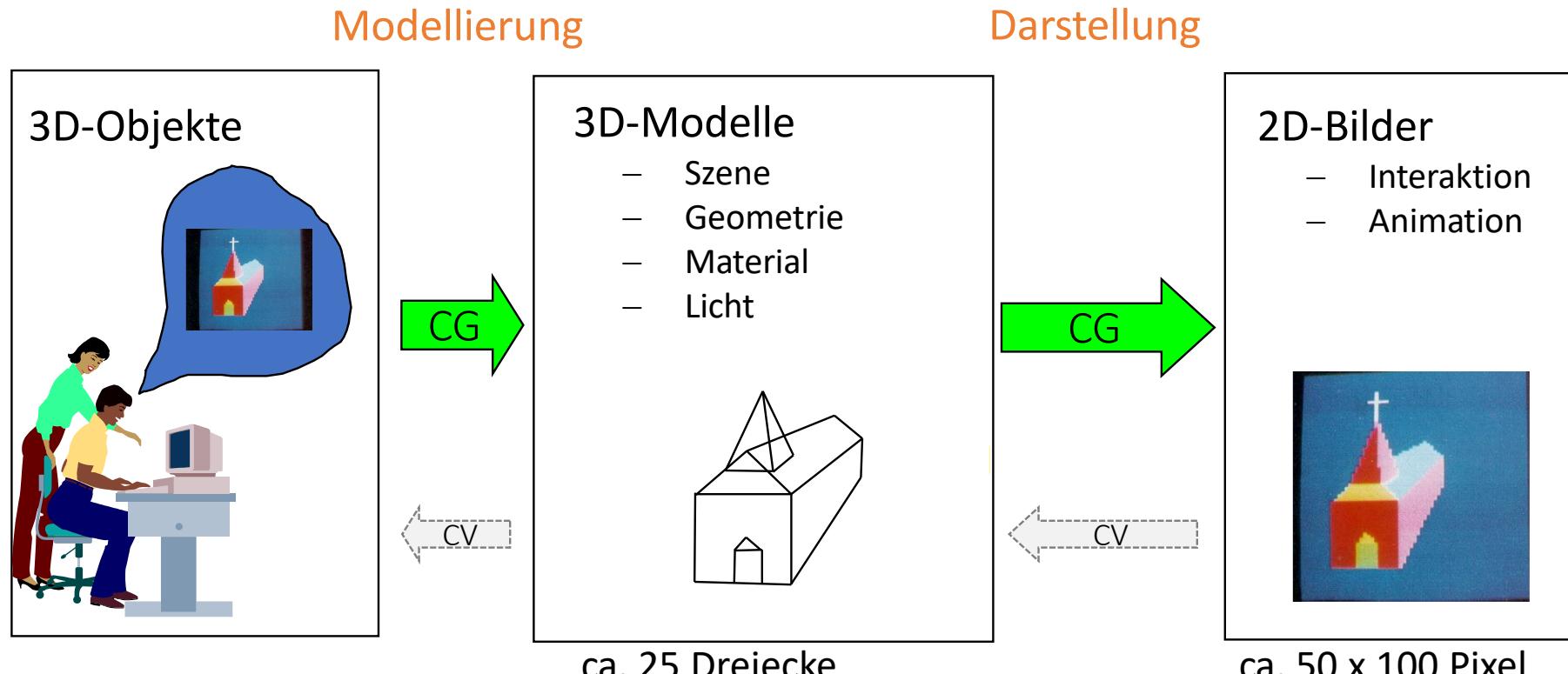
Review SW environment, develop clean and maintainable design

Take hardware into account, optimize, if necessary

Beispiel-Problem: Ozean Rendern

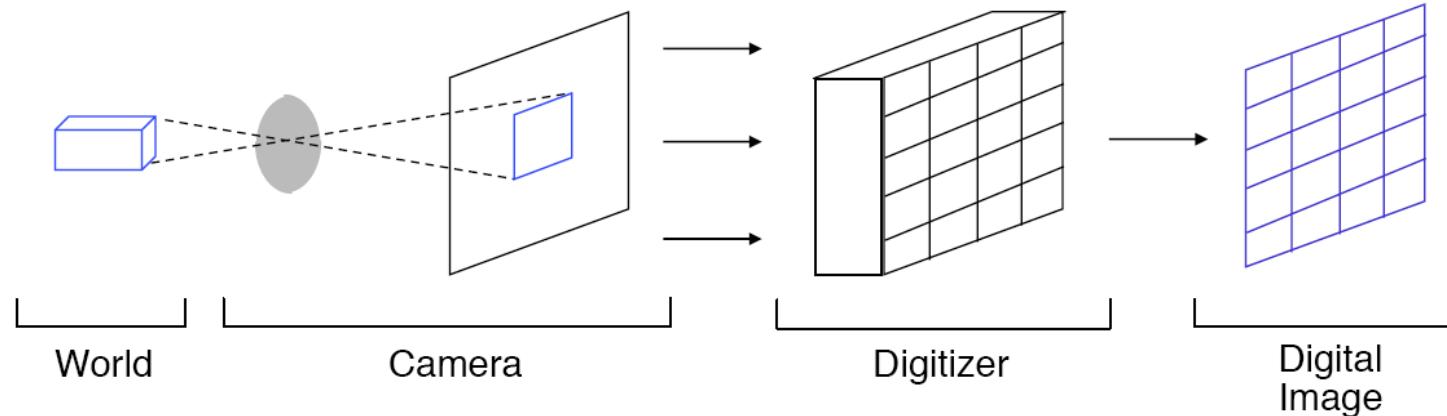


Einschub: 3D-Graphik-Pipeline



Schematische Darstellung

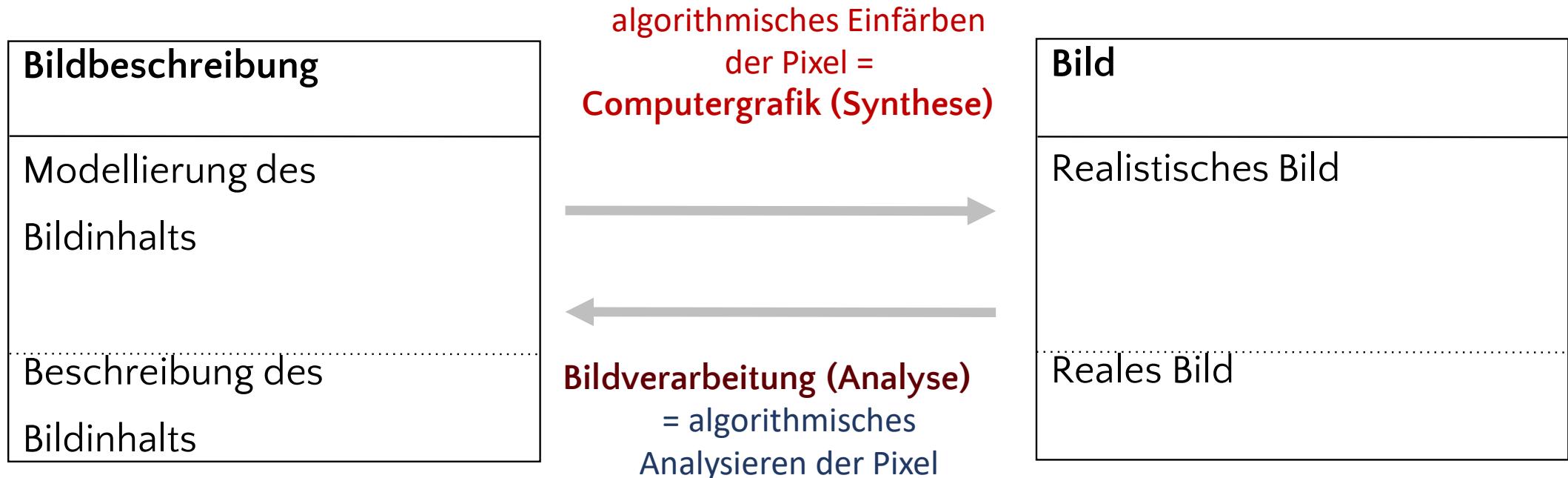
Visual Computing



- Graphische Datenverarbeitung (Computer Graphics)
 - Prozess von links nach rechts (Bildsynthese)
- Bildverarbeitung (Computer Vision)
 - Prozess von rechts nach links (Bildanalyse)
- Visual Computing
 - Fusion von CG und CV



Bildverarbeitung vs. Computergrafik



- Achtung: Bildverarbeitung betrachtet Bildinhalte (z.B. für Objekterkennung und -verfolgung)
 - Bildbearbeitung umfasst aus der Photographie übernommene Verfahren (z.B. Sepia-Tonung u.ä.)

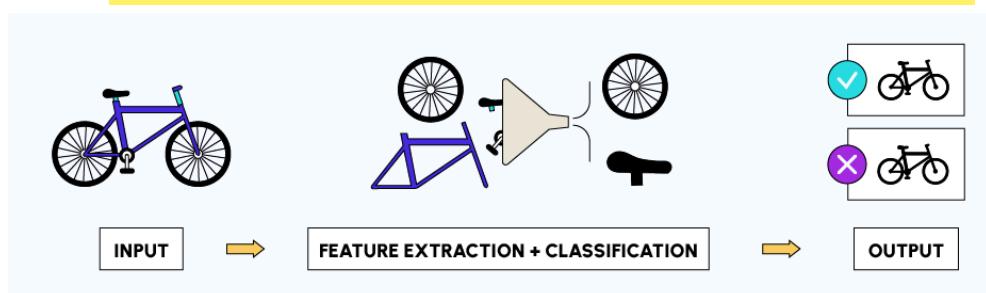
Bildverarbeitung vs. Computergrafik

- Computer Vision

- Wie kann man Früchte aus einem Array mit Graustufenwerten erkennen?
- Wie kann man Tiefe aus einem Array mit Graustufenwerten erkennen?

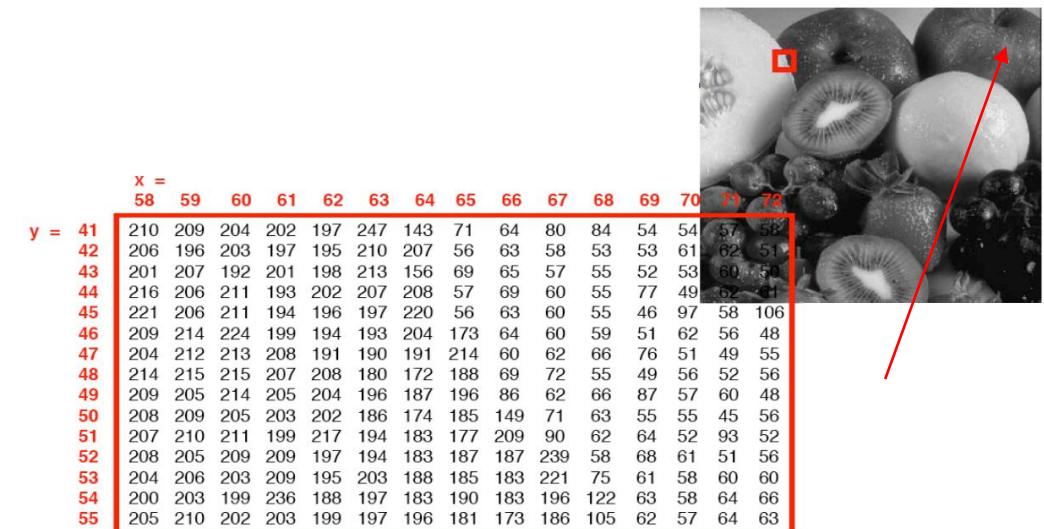
- Problem inverser Graphik?

- ...oder ist es mehr?
- Wie erkennt man z.B. einen Apfel?
- CV nutzt Machine Learning Algorithmen



- Computergraphik

- Wie kann man ein Array mit Werten erzeugen, das aussieht wie Früchte?
- Wie kann man ein Array mit Werten erzeugen, so dass ein Betrachter darin Tiefe wahrnimmt?



Bildverarbeitung

Bilder lassen sich durch „Faltungsmatrizen“ (engl. Convolutions) analysieren (z.B. zur Kantenerkennung):

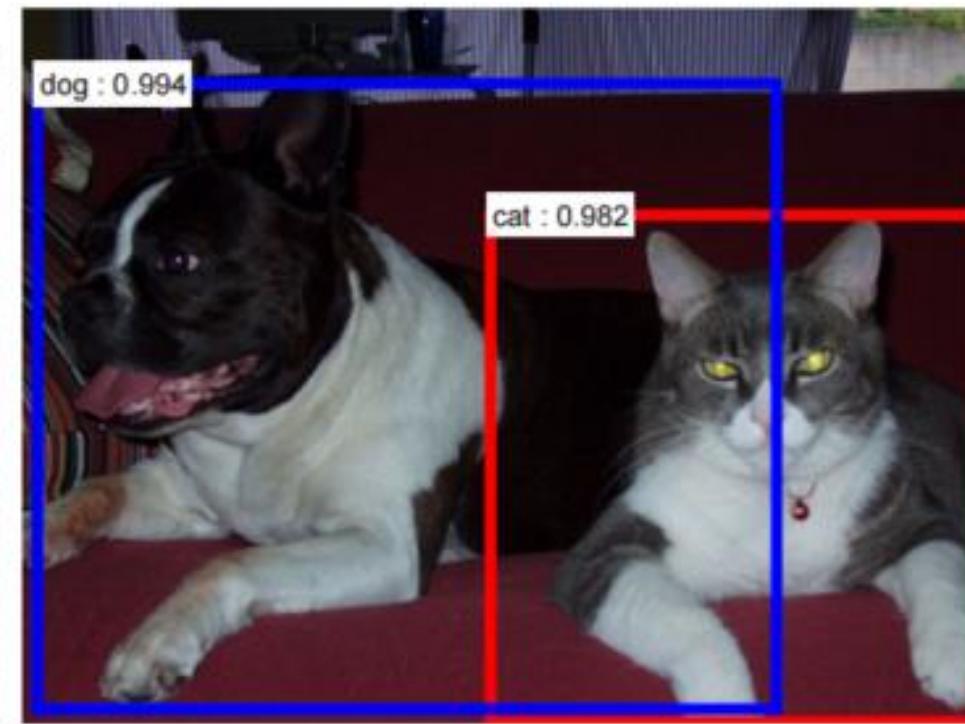
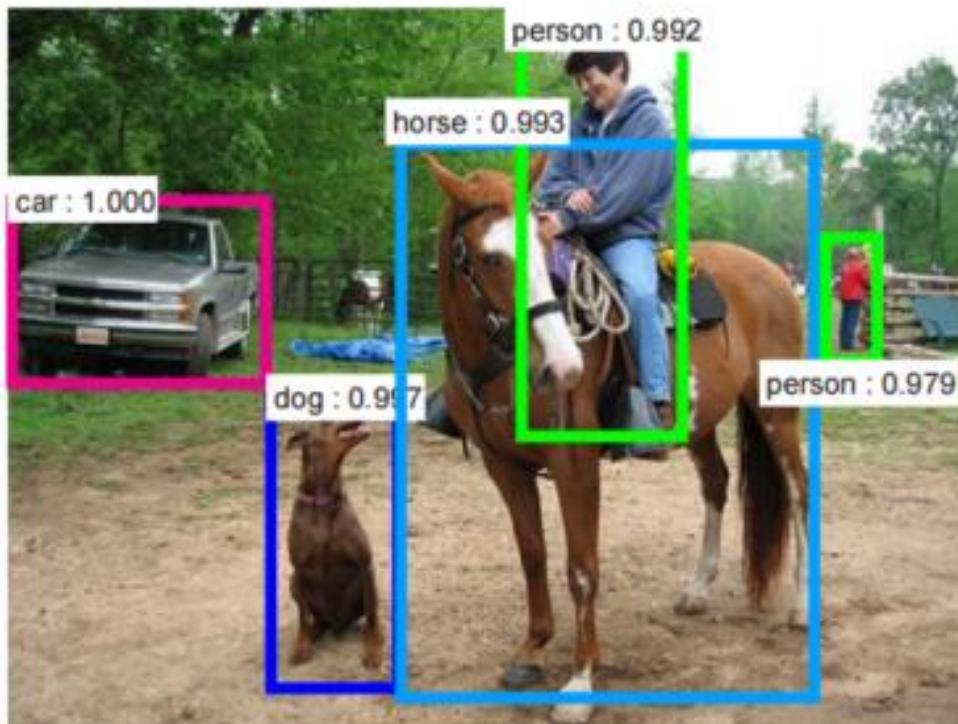


	Operation	Filter	Convolved Image
Identity		$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection		$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
Sharpen		$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
Box blur (normalized)		$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)		$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Bilder aus: <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

Bildverarbeitung

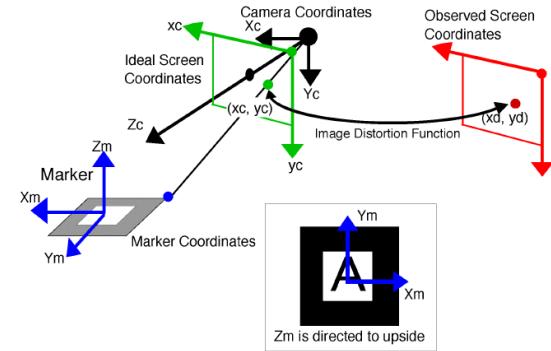
Mit einem **neuronalen Netz aus Faltungsmatrizen** (Deep Learning), den **Convolutional Neural Networks** (CNN), kann man **komplexe Objekte im Bild oder Video erkennen** (funktioniert z.T. auch für 3D-Objekte)



Bilder aus: <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>

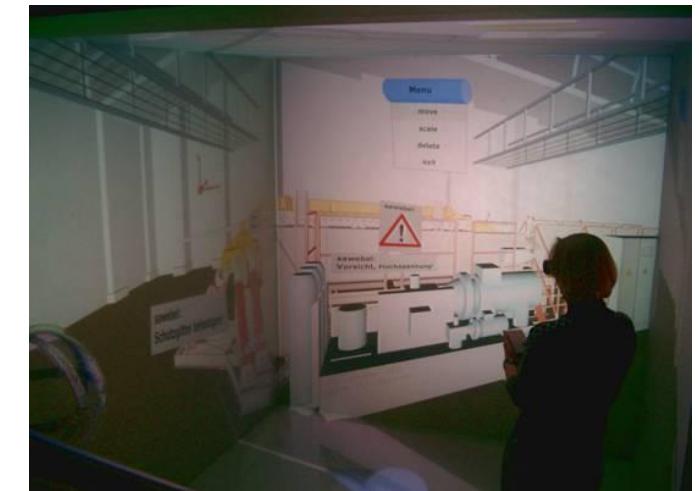
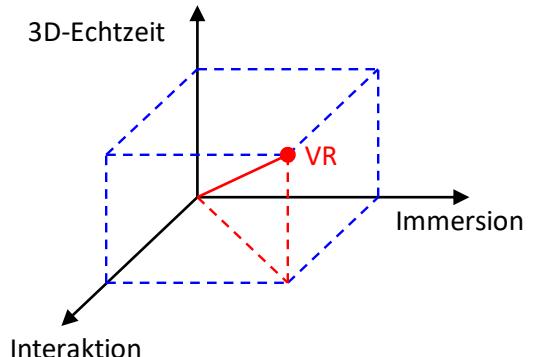
Augmented Reality (AR)

- Mobile Geräte
 - Leistungsfähige CPU / GPU
 - GPS u. andere Sensoren, Kamera
- Georeferenzierte Informationen
 - Karten, 360° Bilder usw.
- Tracking
 - Benötigt Position u. Orientierung des Geräts
 - Über Sensoren wie GPS und Beschleunigungssensor
 - Über Kamera (Computer-Vision-basiert)
 - Erweitern der realen Welt mit virtuellen Informationen (2D/3D)
 - → „Outernet“ & „Internet der Dinge“



Virtual Reality (VR)

- Technologie zur Schaffung virtueller Welten
 - Echtzeit-Interaktion mit 3D-Szene
 - Echtzeit-Simulation u. -Animation
 - Echtzeit-Darstellung (mind. 24 fps)
 - Multimodal: visuell, auditiv, haptisch
- Typische Ausgabegeräte
 - Monitor
 - Aktualisierungsrate begrenzt Rendergeschwindigkeit: 62.5 Hz \sim 16 ms
 - Latenz auch durch Sensoren bzw. andere Gerätedaten ($\rightarrow < 100$ ms !)
 - Head Mounted Display (HMD)
 - Stereoskopische Darstellung auf 2 getrennten Bildschirmen (Klassiker)
 - Immersive Projektionssysteme
 - Mehrere Benutzer, Stereobrillen, nur ein Benutzer getrackt, z.B. CAVE



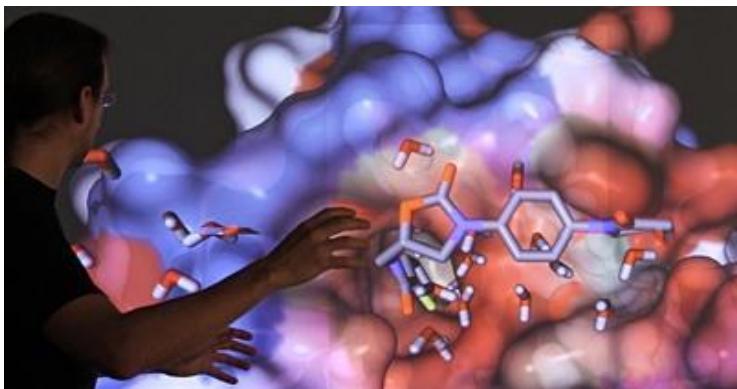
Virtual Reality (Beispiele)



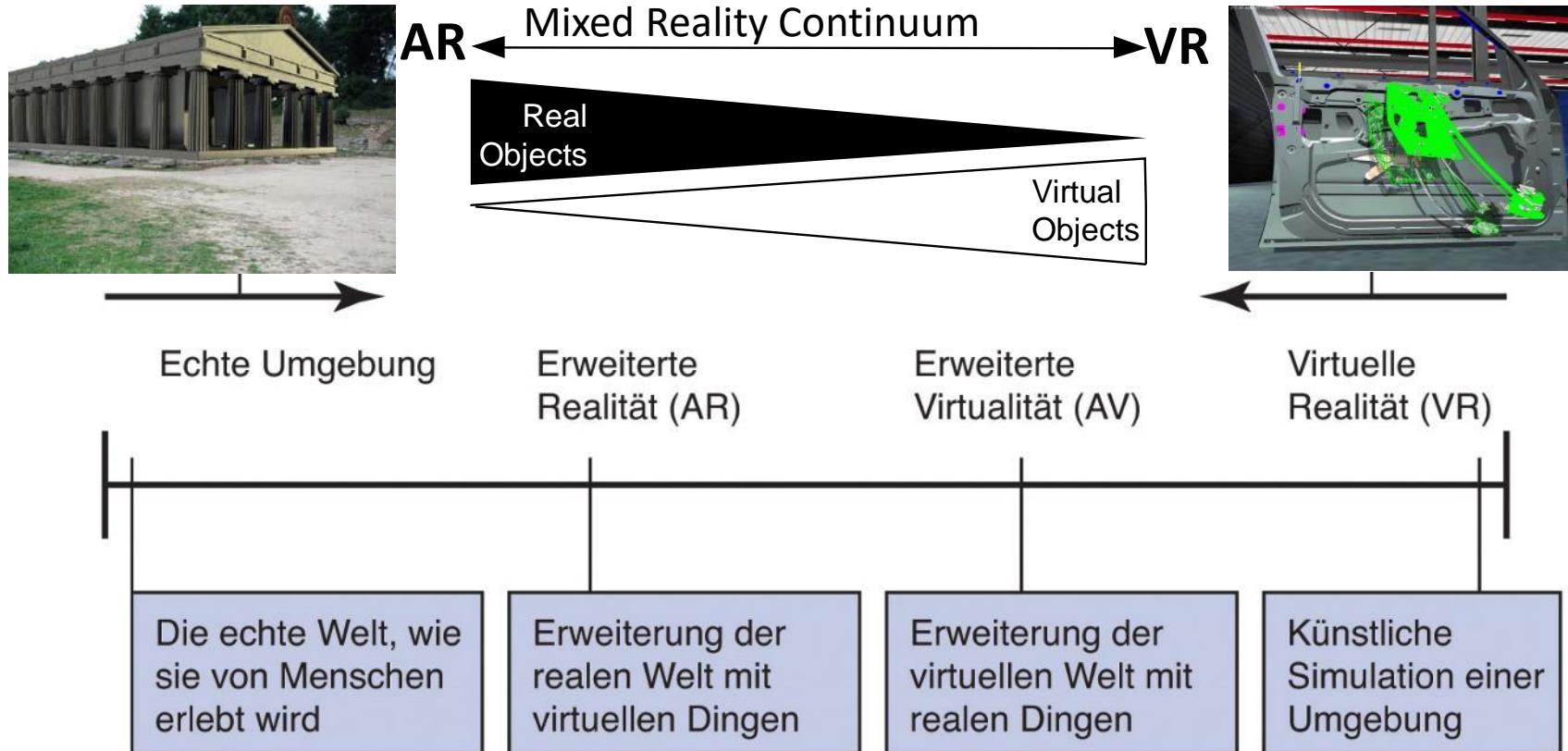
Ein- und Ausgabegeräte



h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES



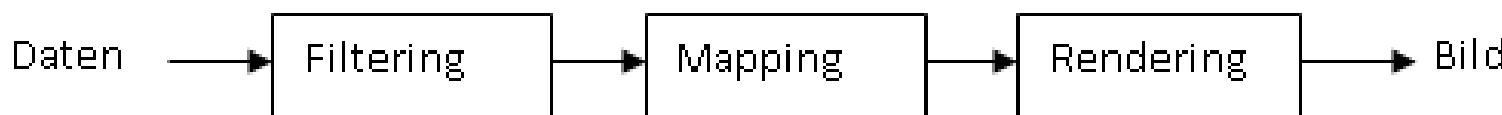
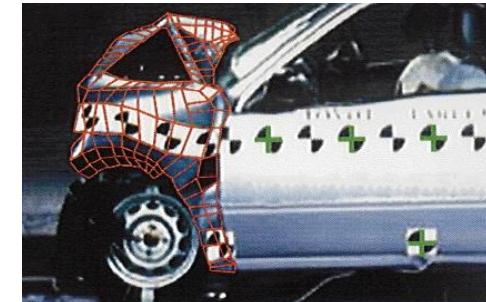
Extended Reality (XR)



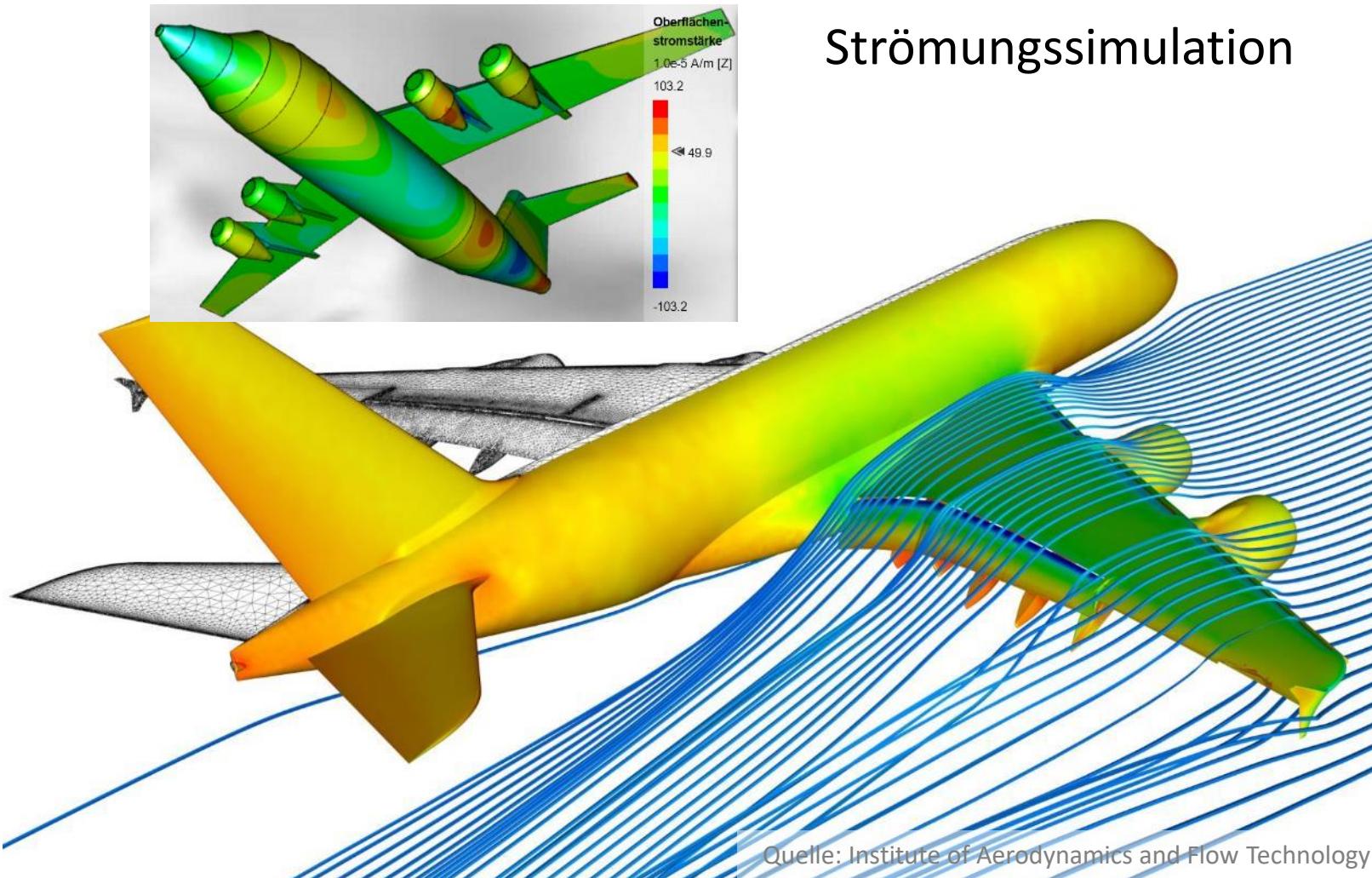
Darstellung nach Milgram und Kishino

Wissenschaftliche Visualisierung

- Ziel: Erzeugung visueller Repräsentationen von abstrakten Datenmengen zur Erleichterung des Verständnisses und Analyse der Daten
- Beispiele
 - Darstellung medizinischer Bilddaten (MRT, CT, PET)
 - Skalar-/vektorwertige Simulationsdaten (FEM, CFD)
- Visualisierungs-Pipeline:

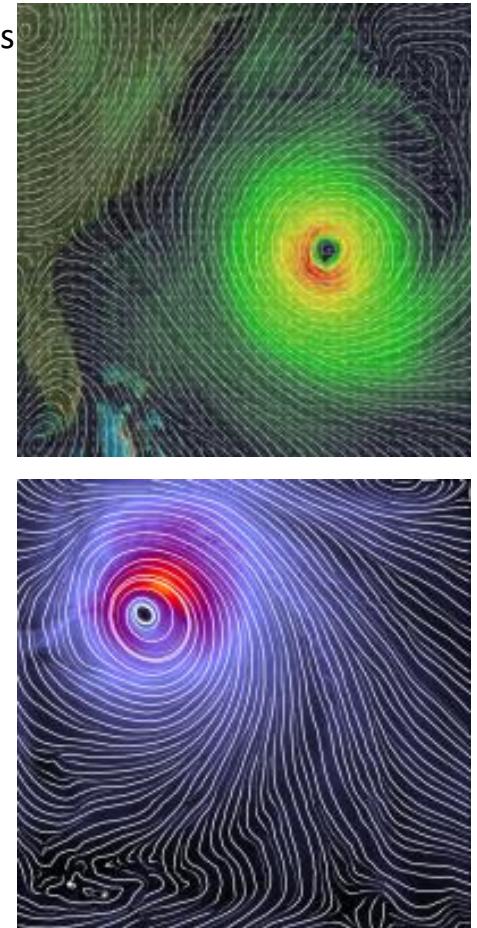


Wissenschaftliche Visualisierung



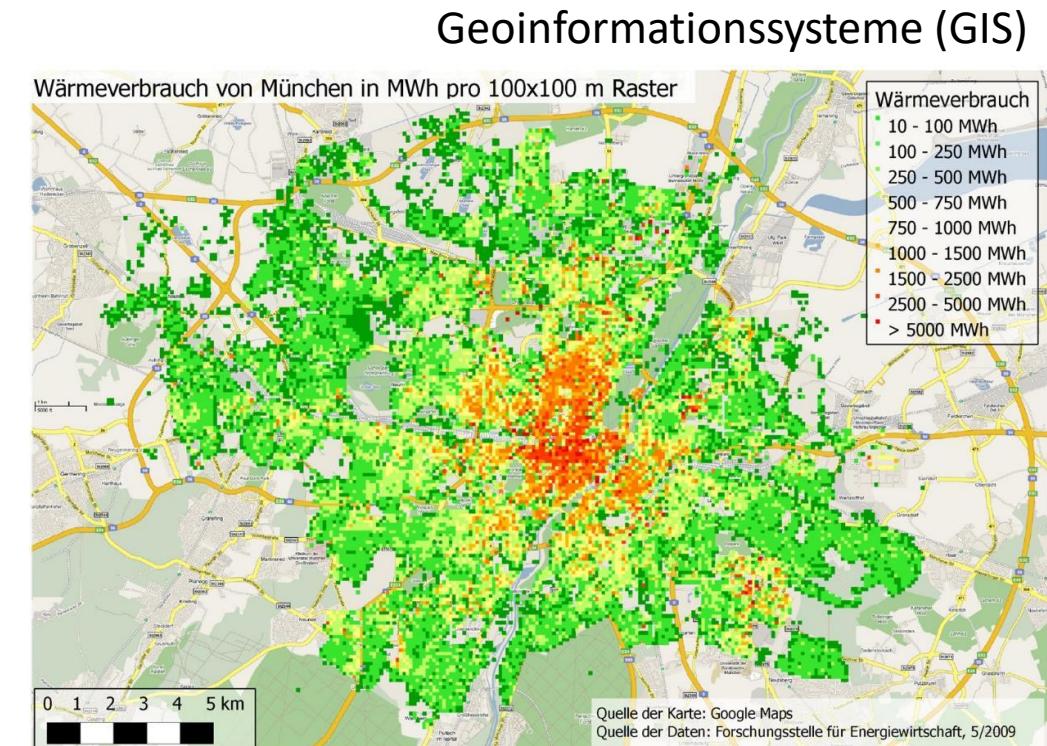
Simulation eines Hurrikans

Quelle: TU Wien,
Institut für CG u.
Algorithmen



Datenvisualisierung

- Visualisierung bedeutet sichtbar machen, darstellen
- Computergrafik beschäftigt sich nicht nur mit Abbildung realer Objekte, sondern auch mit Darstellung von Informationen und wissenschaftlich-technischen Daten

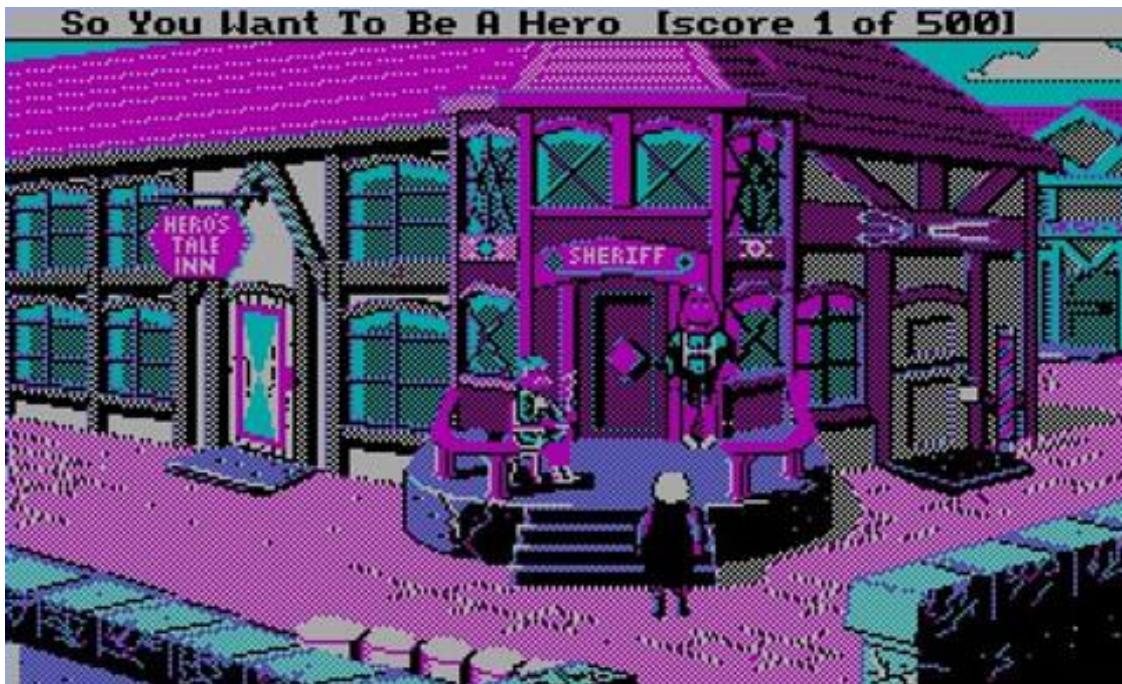




Entwicklung Graphikausgabe

Color Graphics Adapter (CGA, 1981)

- 320x200 Pixel bei 4 Farben
- 640x200 Pixel bei 2 Farben

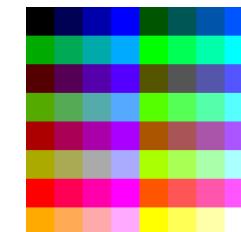
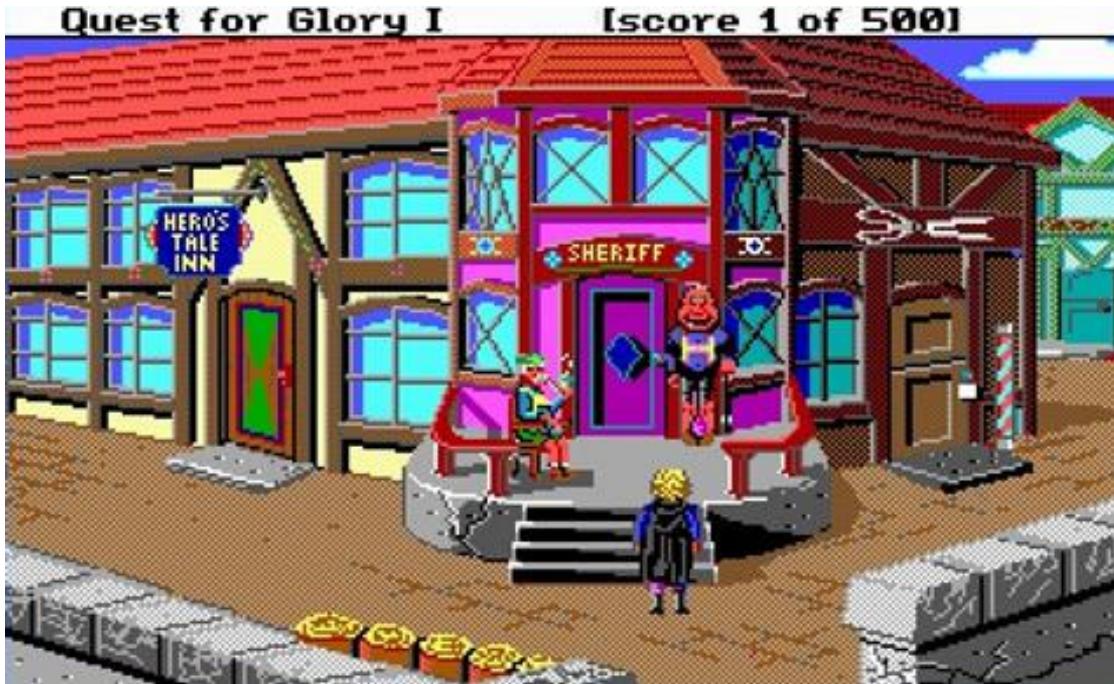


Quelle: Quest for Glory I, Sierra On-line

CGA-Palette 1	
default	5 — Magenta
3 — cyan	7 — weiß (hellgrau)

Enhanced Graphics Adapter (EGA, 1984)

- 640x350 Pixel bei 16 Farben
- Auswahl aus Palette mit 64 Farben



Quelle: Quest for Glory I, Sierra On-line

Video Graphics Array (VGA, 1990)

- 320×200 px bei 256 Farben aus Palette mit 2^{18} Farben
- 640×480 px bei 16 Farben



Quelle: Quest for Glory I, Sierra On-line

Erste Standards

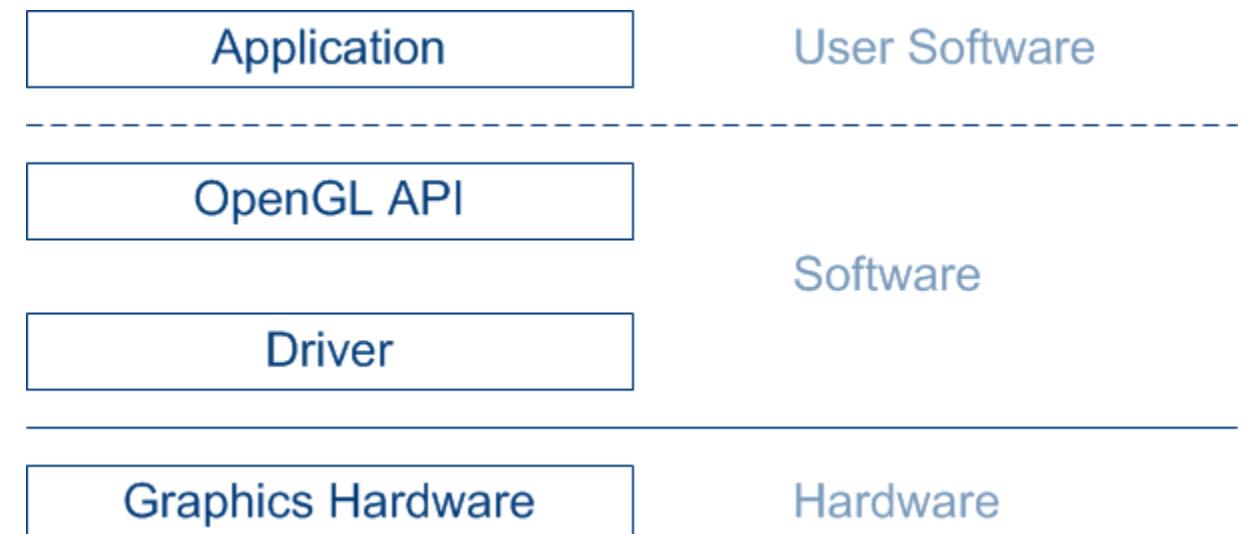
- Video Electronics Standards Association (VESA)
- VESA BIOS Extension (VBE); erster Standard



Quelle: Siedler 1, BlueByte

OpenGL Standard

- Open Graphics Library (OpenGL), Silicon Graphics, 1992
 - OpenGL ist ein Renderingsystem und spezifiziert eine low-level API für 3D-Graphik
 - Befehle werden von Grafikkarte ausgeführt und sind im Grafikkartentreiber implementiert
- Verfügbarkeit auf allen wichtigen Plattformen
 - Windows, Unix/Linux, MacOS, Mobile
 - Basis für Forschungs-/Industrieprojekte
- Einfache Schichtenarchitektur
 - Ansteuerung der Hardware über:
OpenGL, Vulkan, Metal, Direct3D
 - Hardware besteht u.a. aus Grafikkarte
(beinhaltet Speicher mit Bilddaten) und
Monitor (liest Bildspeicher aus)

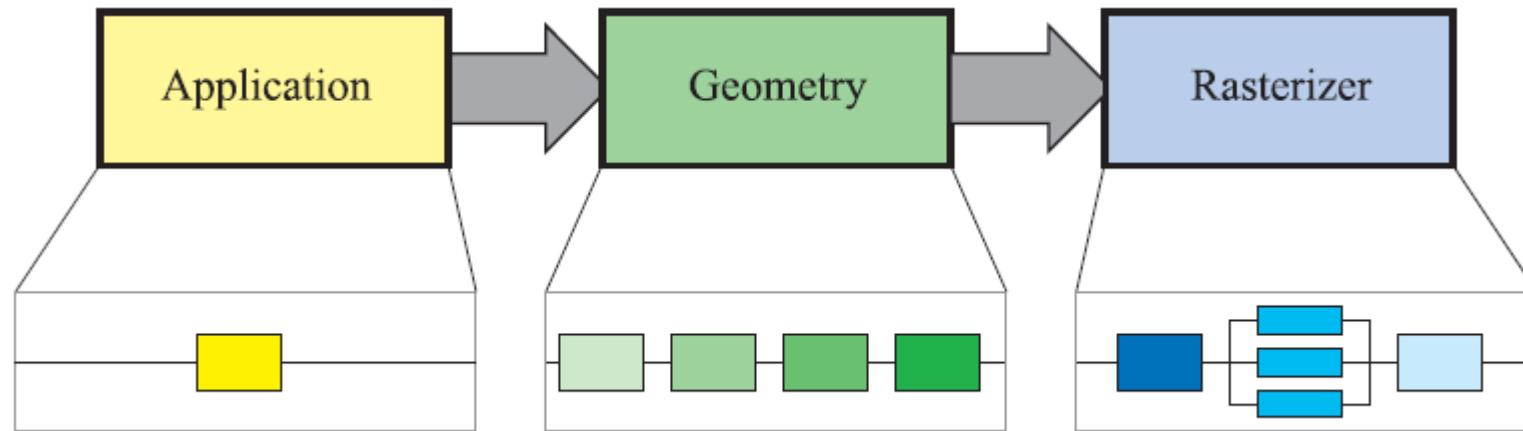


WebGL: OpenGL & GLSL im Web



- JavaScript-Binding für OpenGL ES 2.0 (bzw. 3.0 bei WebGL 2.0) im Webbrowser
 - Alle Browser (auf Desktop und mobilen Systemen)
 - Wie OpenGL (und Vulkan) verwaltet von Khronos Group
- Rein GLSL-Shader basiert
 - D.h. keine sog. Fixed-Function Pipeline mehr
 - Keine Variablen aus GL-State, kein Matrix-Stack, keine Mathe-Lib... (man muss alles selbst programmieren)
- HTML5 `<canvas>` Element bietet neben 2D auch 3D-Rendering-Context
 - API-Aufrufe über GL-Objekt, z.B.:
`gl = canvas.getContext('webgl');`
`gl.viewport(0, 0, 400, 300);`



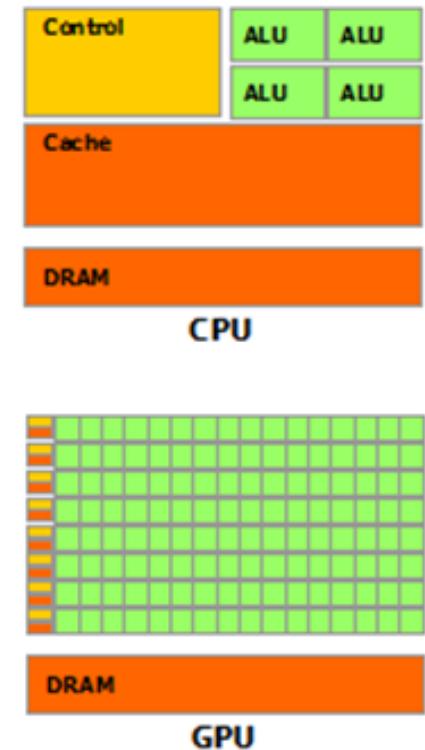


Hardwarebeschleunigung (Arbeitsweise einer GPU)



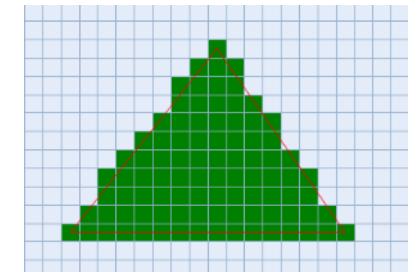
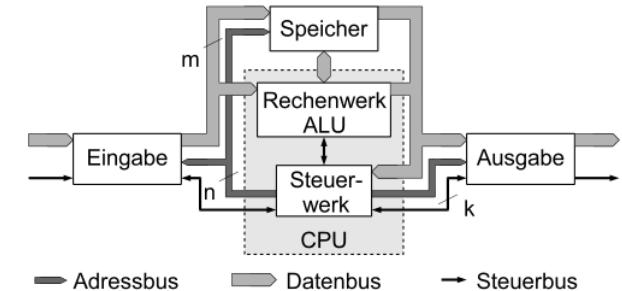
GPU (Graphics Processing Unit)

- Erste Graphikprozessoren mit 3D-Beschleunigung seit Mitte 90er
- Spezialisiert auf Graphikberechnungen
 - Entlastet universell ausgelegte CPU
 - Höhere Integrationsdichte und Parallelisierung
 - Selbständige Abarbeitung dreiecksbasierter Algorithmen
 - Transform & Lighting, Texture Mapping, Z-Buffer etc.
 - Zunächst nur sog. Fixed-Function-Pipeline, später Shader
 - Programmierung über APIs wie DirectX und OpenGL
 - Implementierung Teil des Grafiktreibers
 - OpenGL-API ist State Machine, Fktn. setzen Render States
- Streaming-Multiprozessoren
 - Threadblöcke (erlaubt sog. GPGPU-Anwendungen)
 - Hunderte von ALUs mit Ztausenden von Threads



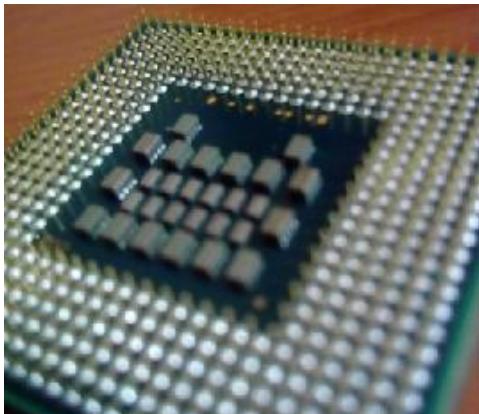
CPU vs. GPU

- CPU
 - Sequentieller Ablauf weniger komplexer Kontrollflüsse
 - Von-Neumann-Rechner (Single Instruction, Single Data)
 - Viele verschiedene Instruktionen
 - Bedingte Verzweigungen der Kontrollflüsse
- GPU
 - Massiv parallele Abarbeitung der gleichen Abläufe
 - I.d.R. müssen für Vielzahl von Pixeln die gleichen, einfachen Instruktionen ausgeführt werden
 - SIMD (Single Instruction, Multiple Data)
 - Wenige arithmetische Instruktionen
 - Geringe Anzahl von Verzweigungen



Zusammenspiel CPU & GPU

- OpenGL bietet Funktionen, die sowohl CPU als auch GPU ansprechen
 - Ziel: Upload einer großen Menge von Daten auf die GPU zur Beschleunigung



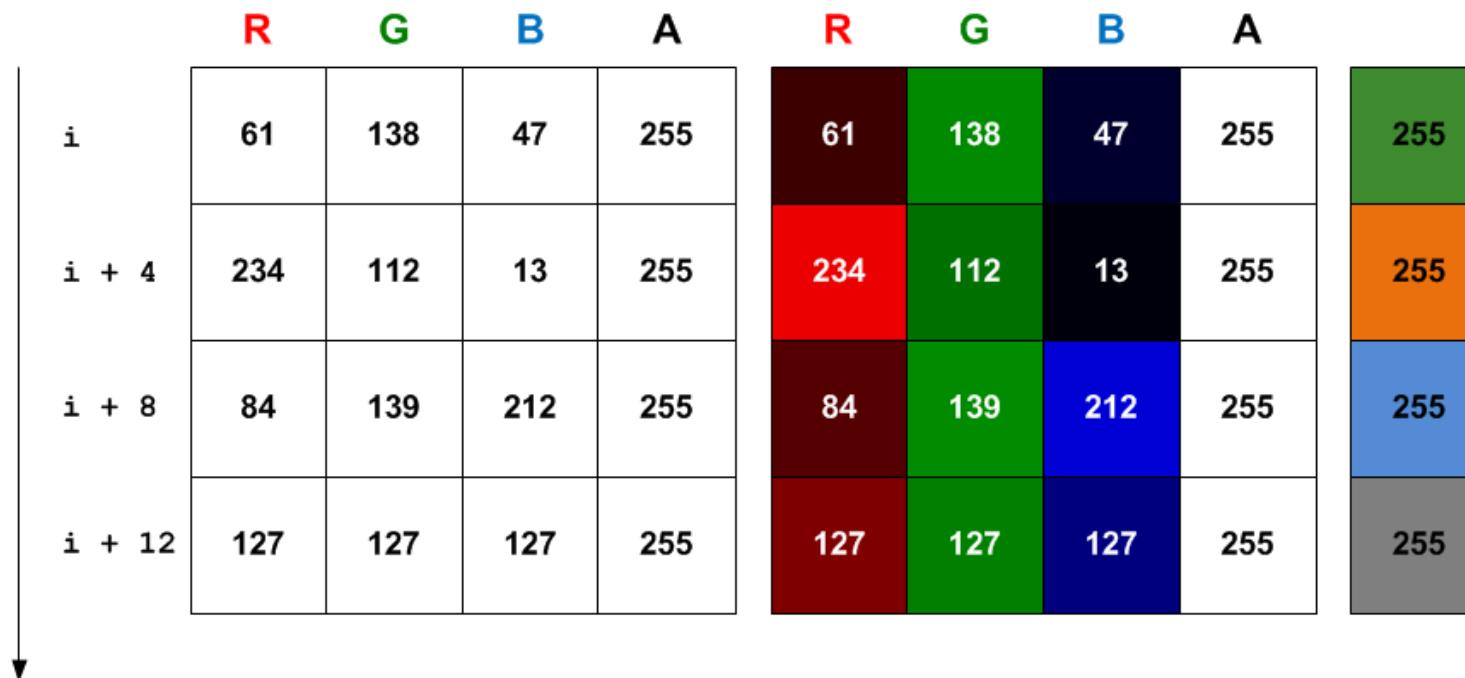
Upload von Daten
auf die GPU



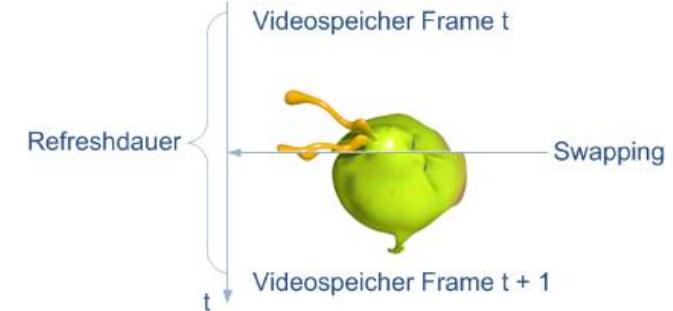
- Datenerstellung/-manipulation
- Programmlogik
- Datenverarbeitung
- Bilderzeugung

Videospeicher

- Verwaltung des Bildspeichers geschieht über Pixel
 - Bei LCD-Monitor mit Refresh Rate von 62.5 Hz wird Videospeicher alle 16 ms neu ausgelesen
- Jeder Pixel belegt auf heutiger Hardware 4 Bytes (RGBA-Kanäle)
 - Für Rot, Grün, Blau, Alpha (Transparenz) je 8 Bit



Double Buffering



- Grafikprogrammierung
 - Daten in Videospeicher schreiben (Darstellung eines Einzelbildes heißt Frame)
 - Gefahr des sichtbaren Bildaufbaus
- Lösung
 - Mind. zwei Videospeicher: Double Buffering
 - Sichtbarer Videospeicher wird gelesen, unsichtbarer Videospeicher wird geschrieben
 - Nach Frame werden Speicherbereiche vertauscht
 - Swapping (erfolgt durch Grafikkarte); nicht synchronisiert mit Refreshdauer des Monitors
 - Framerate der GPU (d.h. der Grafikkarte) meist davon verschieden
 - Synchronisation zwischen GPU (Graphics Processing Unit) und Monitor über sog. Vsync (verhindert Aktualisierung, während Bild aufgebaut wird)

Vielen Dank!

Noch Fragen?

