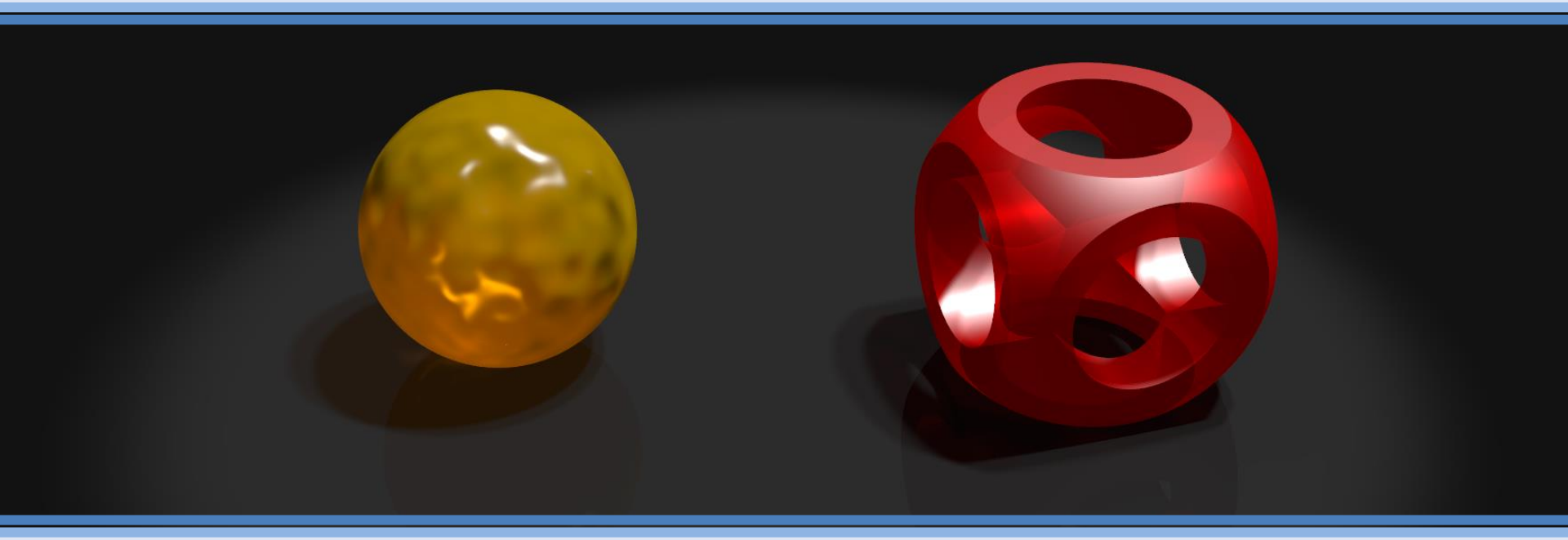
Computergrafik

T. Hopp



Organisatorisches

- 33 Vorlesungsstunden
- 9 Vorlesungstermine: jeweils Mittwochnachmittags
  - Ab 10.10.2018, jeweils 12:30 – 15:45 Uhr

Organisatorisches

- Kontakt: torsten.hopp@kit.edu

- Vorlesungsunterlagen:
  - Folien werden auf der eLearning-Plattform (https://moodle.dhbw.de/course/view.php?id=2000) jeweils nach der Vorlesung bereitgestellt.

  Raum: „**TINF16 Grafik WiSe**“

  - Selbsteinschreibung mit Einschreibeschlüssel: **blinnPhong**

- Wie kann ich Sie kontaktieren?

Organisatorisches

- Organisation der Vorlesung
  - Vorlesung zu ca. 9-10 Themenbereichen
  - Themen allgemein in der Computergrafik angesiedelt, Beispielanwendungen teilweise etwas medizinlastig
  - Begleitende Übungen
    - Beispielaufgaben (z.B. Algorithmen, Berechnungen)
    - Begleitend zur Vorlesung schauen wir uns Programmierübungen an.
      - Wir nutzen OpenGL
      - Grundkenntnisse in C-Programmierung?
      - Wie sieht es mit der Computerinfrastruktur aus?
      - Bitte richten Sie sich eine Programmierumgebung ein um die Übungen nachvollziehen zu können.
  - Klausur: 60 min. Vorläufiger Termin in der Klausurwoche
    - Wissensfragen zur Vorlesung (siehe z.B. Übungsfragen am Ende des Kapitels)
    - Aufgaben wie in den Übungen

Literaturhinweise

- A. Nischwitz, M. Fischer, P. Haberäcker, G. Socher: „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik, Vieweg und Teubner Verlag, 3. Auflage 2011
- J.F. Hughes, A. van Dam, M. McGuire, D.F. Sklar, J.D. Foley, S.K. Feiner, K. Akeley: „Computer Graphics. Principles and Practice“, 3rd edition, Addison-Wesley, 2014
- J. Vince: „Mathematics for Computer Graphics“, 4th edition, Springer, 2014
- [K. Zeppenfeld: „Lehrbuch der Grafikprogrammierung“, 1. Auflage, Spektrum Akademischer Verlag, 2004]
- [F. Klawonn: „Grundkurs Computergrafik mit Java“, 3. Auflage, Vieweg und Teubner, 2010]

1.1. EINLEITUNG

Definition – worum geht es?

- Wikipedia: „Die Computergrafik ist ein Teilgebiet der Informatik, das sich mit der computergestützten Erzeugung, im weiten Sinne auch mit der Bearbeitung, von Bildern befasst.“
- Foley: „Computer graphics is the science and art of communicating visually via a computer‘s display and its interaction devices“.

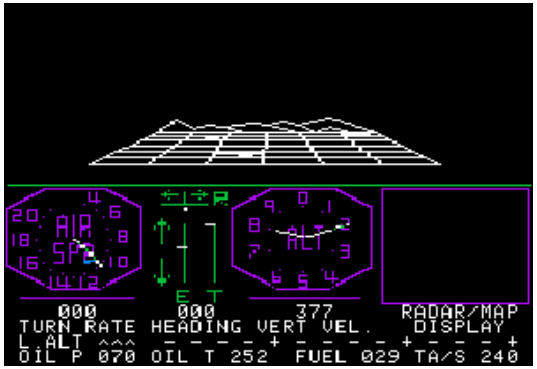
- Grafische Datenverarbeitung
  - **Generative Computergrafik:** Visualisierung komplexer Zusammenhänge, Animationen, Virtual Reality etc.
  - **Bildverarbeitung:** Bildfilterung, Segmentierung, Bildrestauration etc.
  - **Bildanalyse:** Automatische Extraktion von Informationen aus Bildern, z.B. Mustererkennung

Anwendungsbeispiele

- Generative Computergrafik: breites Anwendungsspektrum in zahlreichen Disziplinen
  - Simulation
  - Computer Aided Design
  - Visualisierung
  - Unterhaltung
  - …
- Nachfolgend eine Auswahl typischer Anwendungsgebiete und -beispiele

Ausbildungssimulation

- Triebfeder der modernen Computergrafik, z.B. Flugsimulation



Flight Simulator 1 für Apple II (1979)

Flugtraining-Simulator, Rockwell Collins

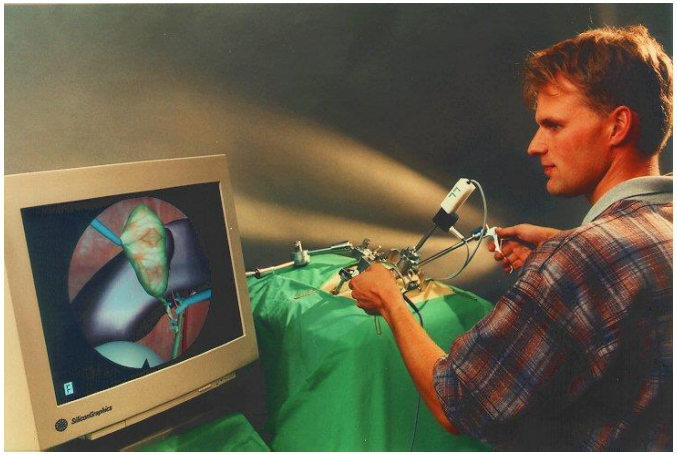
Full Motion Flight Simulator, Lufthansa Flight Training

http://en.wikipedia.org/wiki/History_of_Microsoft_Flight_Simulator
https://www.rockwellcollins.com/Data/Products/Simulation/Operator_Training_Systems/Full_Flight_Simulator.aspx
http://de.wikipedia.org/wiki/Flugsimulation

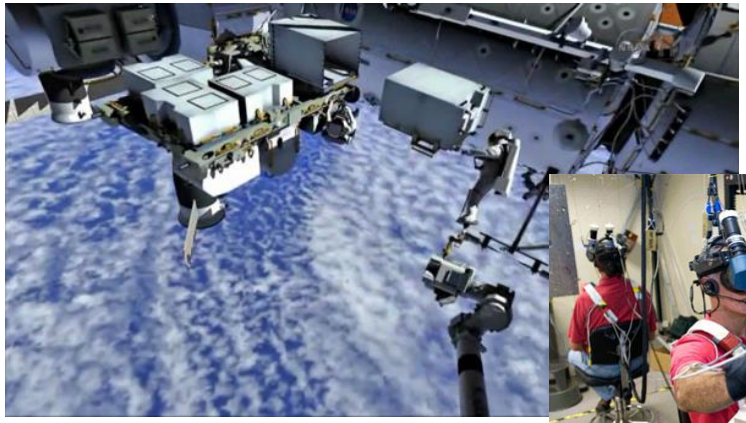
Ausbildungssimulation

Straßenbahnsimulator, Berliner Verkehrsbetriebe

Fluglotsenausbildung, Wien



Karlsruhe Endoskopie Trainingssystem, KIT (1997)

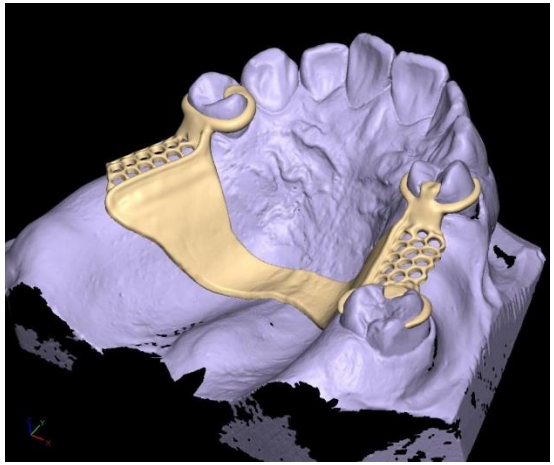


Space Walk Simulator, NASA

https://www.iai.kit.edu/www-extern/index.php?id=2033
http://www.rheinmetall-defence.com/de/rheinmetall_defence/public_relations/news/archive_2013/aktuellesdetailansicht_2753.php
http://www.wien.gv.at/verkehr-stadtentwicklung/fluglotse.html
http://www.nasa.gov/centers/johnson/engineering/robotics_simulation/virtual_reality/index.html
http://www.cbsnews.com/network/news/space/home/spacenews/files/e93930fcf48b6cbdf795f3f81efae7b7-177.html

CAD/CAM

- CAD/CAM = Computer Aided Design/Manufacturing
- Rapid bzw. Virtual Prototyping
- Z.B. Design von Autos, virtuelle Crashtests, Cockpit-Design, …
- Z.B. Patientenspezifisches Design von Prothesen etc.



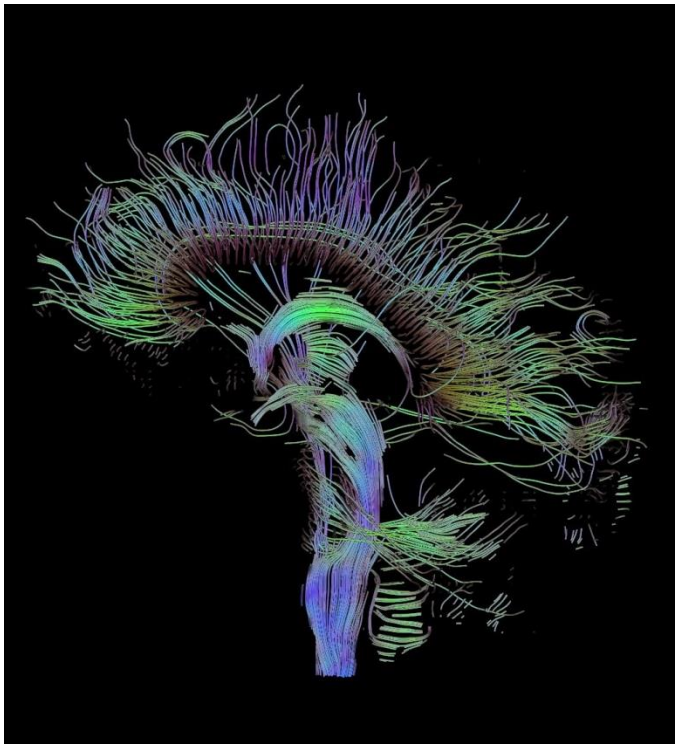
http://www.directindustry.com/prod/esi-group/crash-test-simulation-software-8972-131175.html
http://www.dentsable.com/industries-medical-dental.htm

CAD/CAM

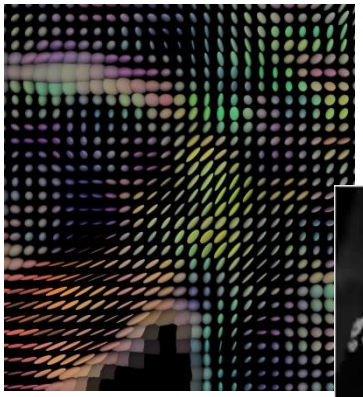
- Z.B. beim Entwurf in der Architektur/Stadtplanung

http://www.pro-eleven.de/

Visualisierung in der Medizin



Visualisierung von Nervenbahnen im Gehirn mit Diffusion-Tensor-MRT

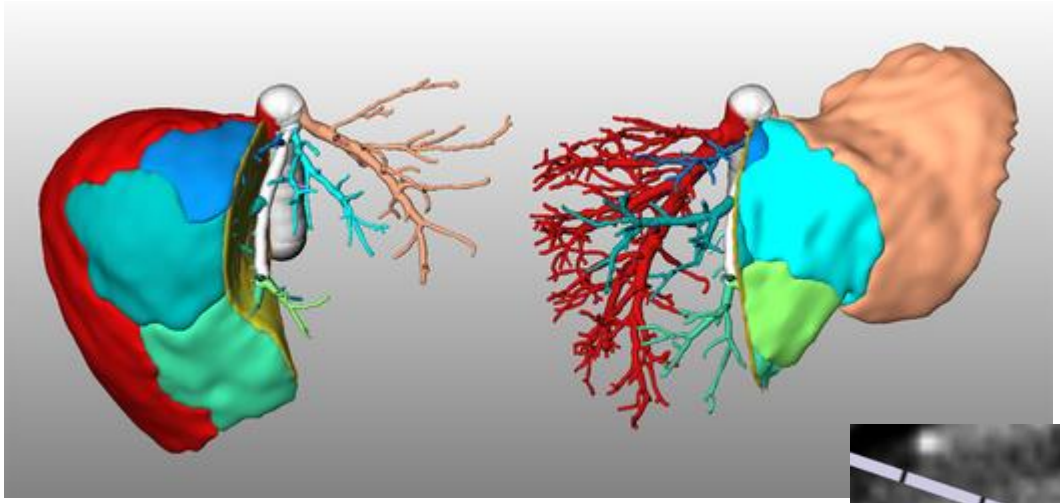


Abstrahierung mit Diffusions-Ellipsen

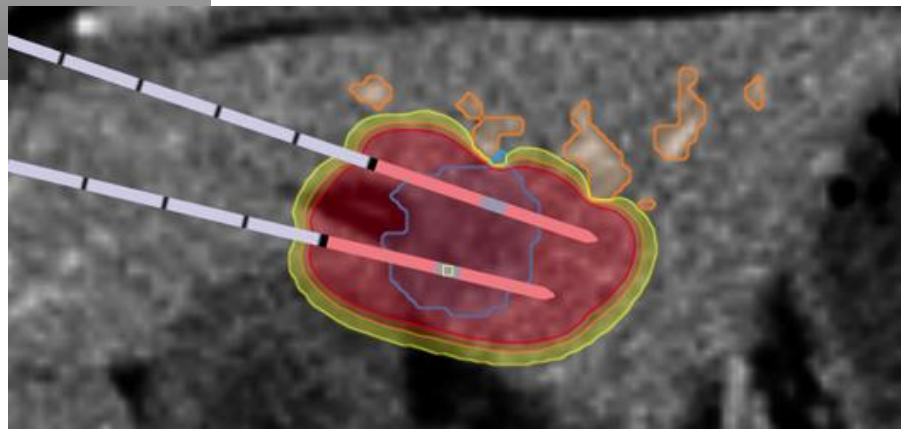
Blutfluss in der Aorta

http://de.wikipedia.org/wiki/Diffusions-Tensor-Bildgebung
http://www.uniklinik-freiburg.de/neurologie/forschung/neurologische-arbeitsgruppen/neurovaskulaere-bildgebung.html

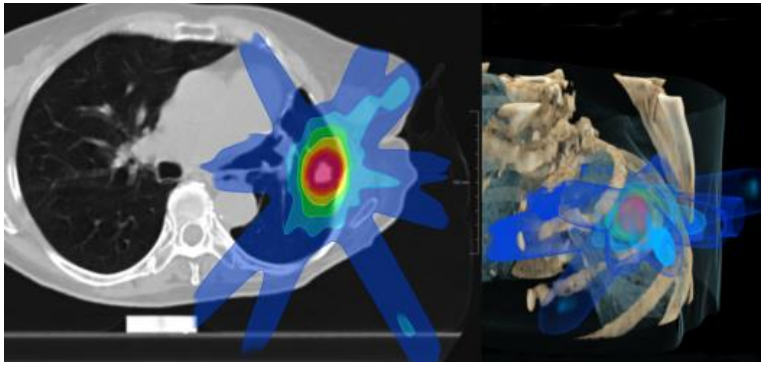
Visualisierung in der Medizin



Optimale Leberpartitionen bei Leberlebendspende, Fraunhofer MEVIS



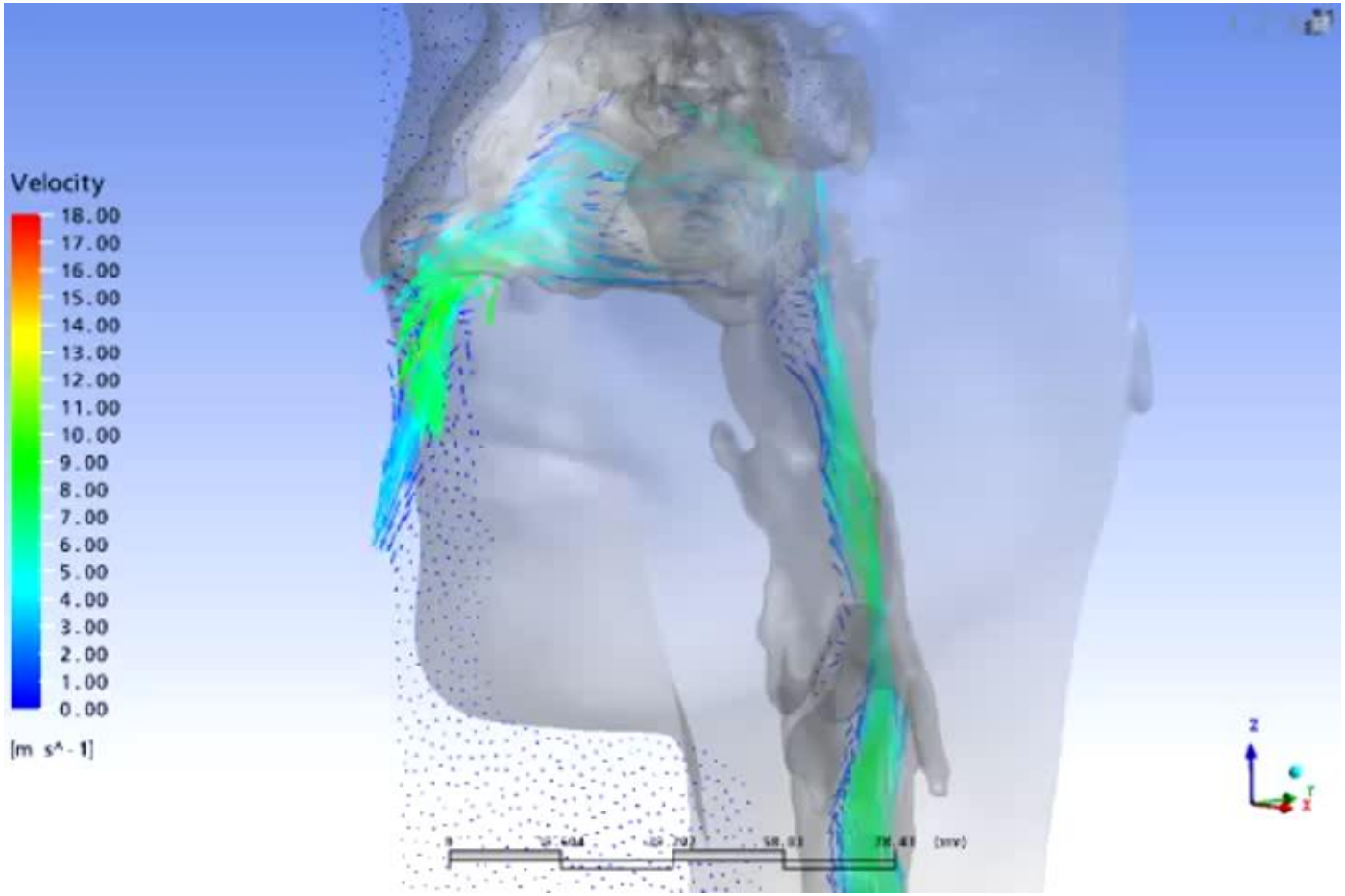
Tumorablations-Planung- und Visualisierung, Fraunhofer MEVIS



Planung von Tumorbestrahlung, Fraunhofer MEVIS

http://www.mevis.fraunhofer.de/klinische-kompetenzen/leber.htm

Visualisierung in der Medizin

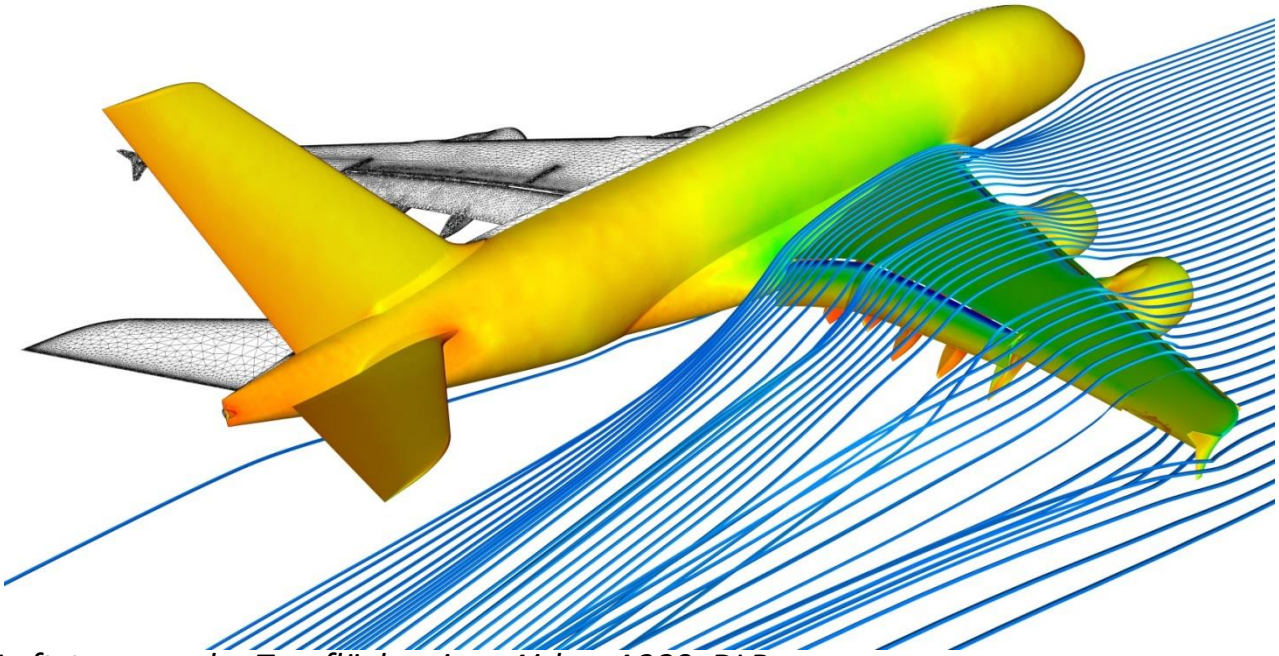


Strömungssimulation der Atmung

https://www.youtube.com/watch?v=1wHTm9JrkFI

Visualisierung

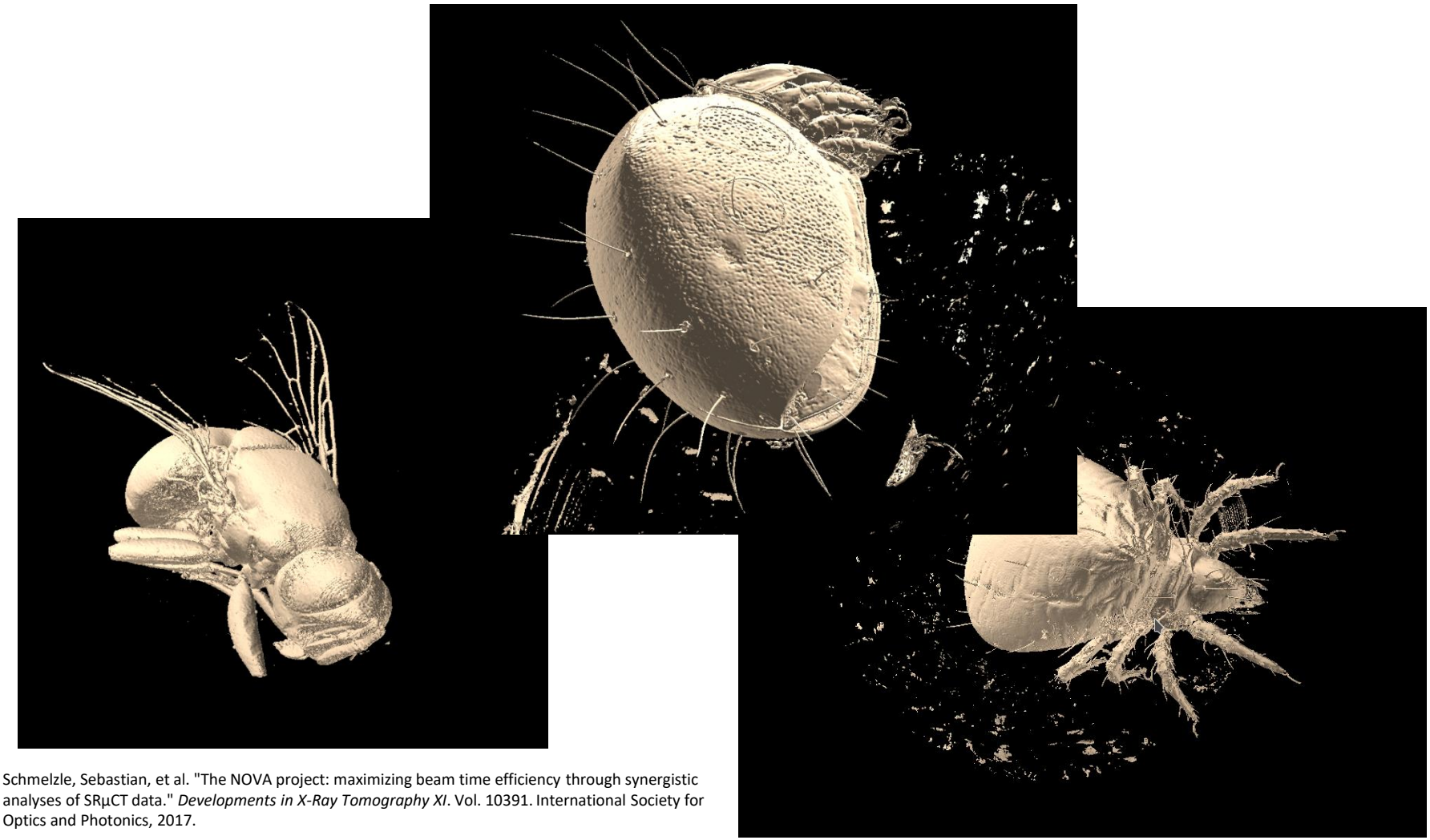
- Darstellung von gemessenen (räumlich bzw. zeitlich aufgelösten) Daten oder Simulationsergebnissen
- Z.B. Strömungssimulation



Luftstrom an der Tragfläche eines Airbus A380, DLR

http://www.dlr.de/media/desktopdefault.aspx/tabid-4985/8422_page-6//8422_read-13200

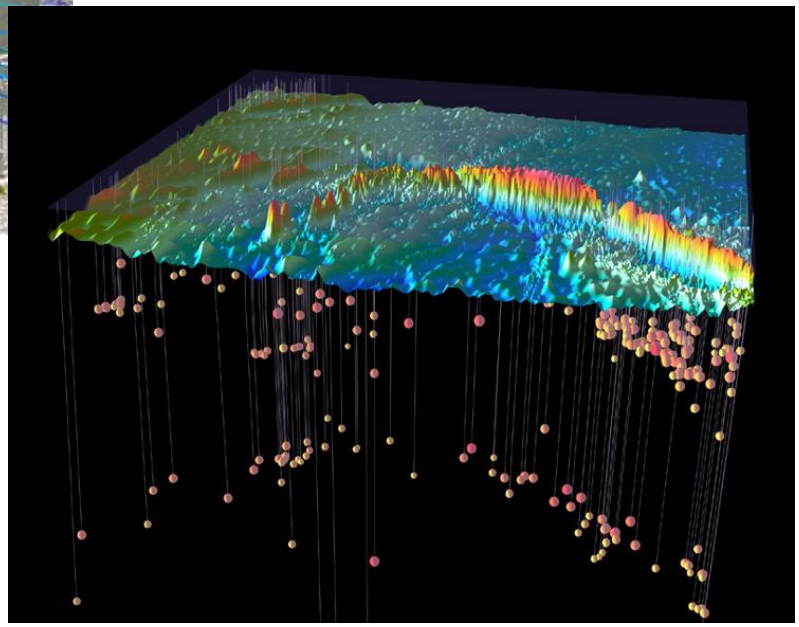
Visualisierung



- Schmelzle, Sebastian, et al. "The NOVA project: maximizing beam time efficiency through synergistic analyses of SRµCT data." *Developments in X-Ray Tomography XI*. Vol. 10391. International Society for Optics and Photonics, 2017.

Visualisierung

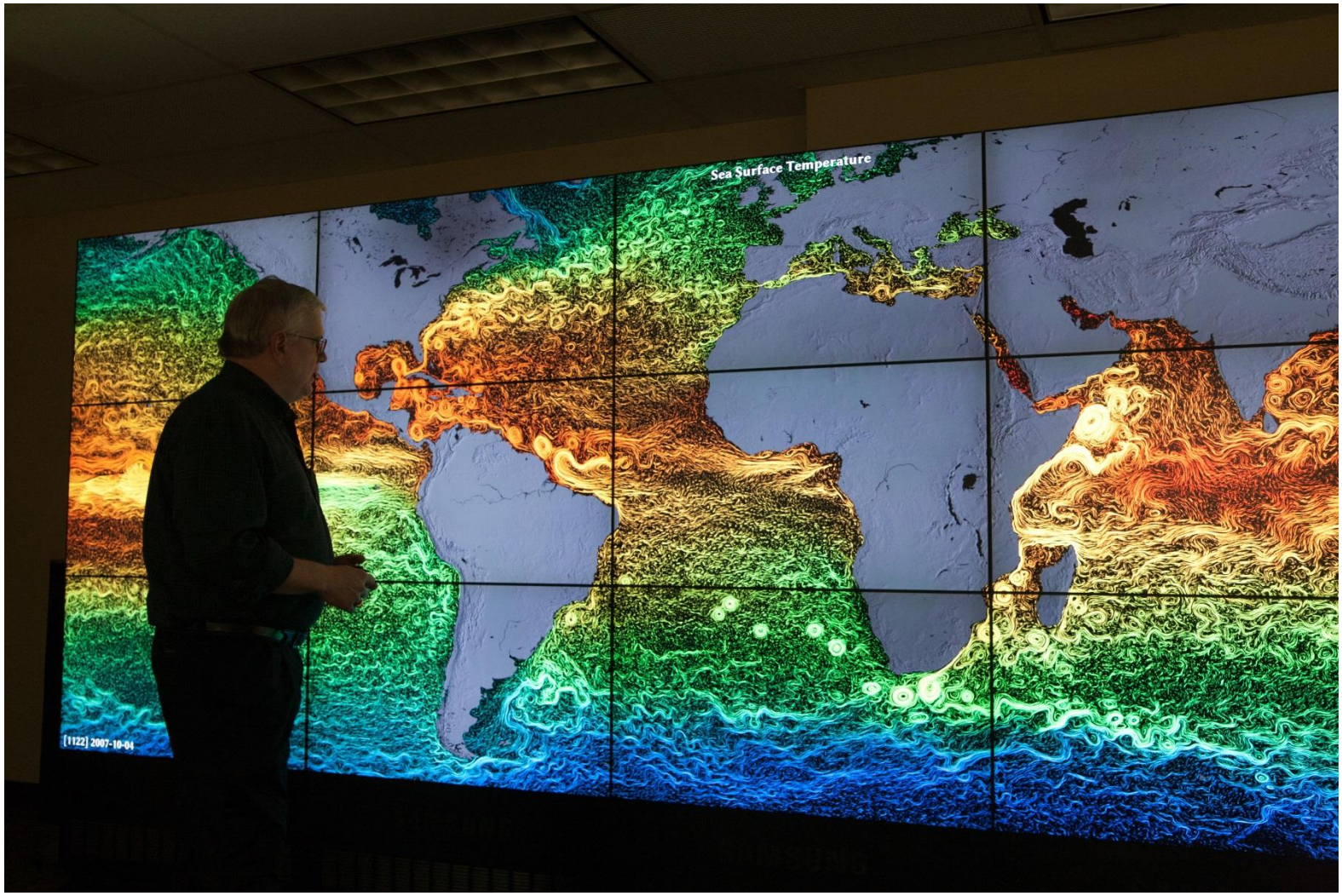
Visualisierung einer Wind-Simulation im Karlsruher 3D Stadtmodell, KIT



Macquarie Fault Zone, Meeresboden-Höhenprofil mit Epizentren, ACES Visualiztion Laboratory

http://www.hitechmex.org/Viz.html
http://www.math.kit.edu/ianm4/seite/visualization/de

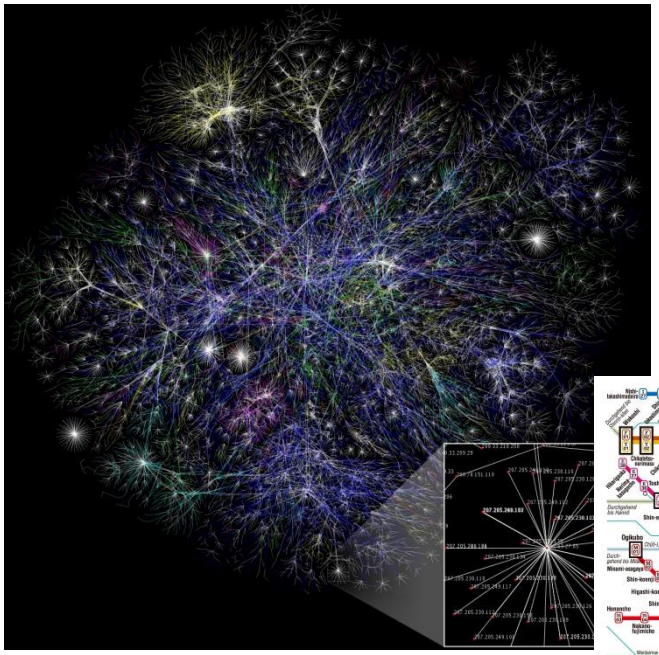
Visualisierung



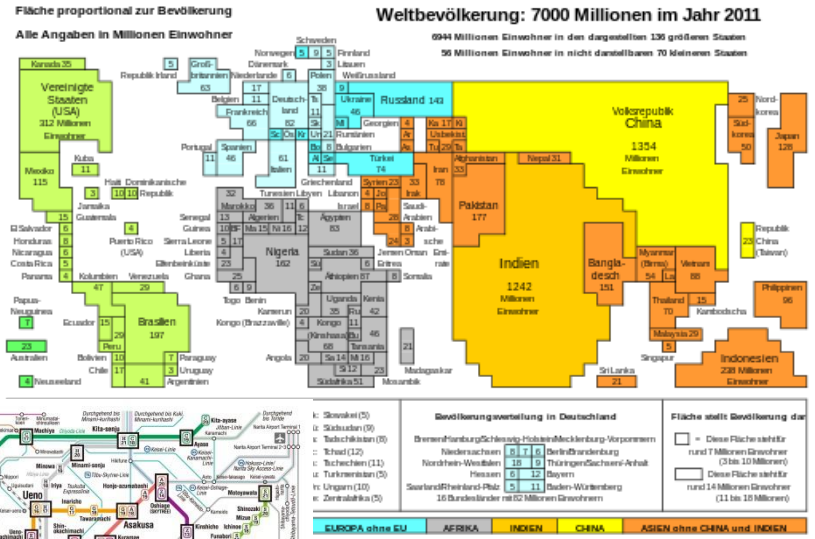
http://blogs.agu.org/wildwildscience/2013/03/18/nasas-science-visualization-wall-cool-is-an-understatement/

Informationsvisualisierung

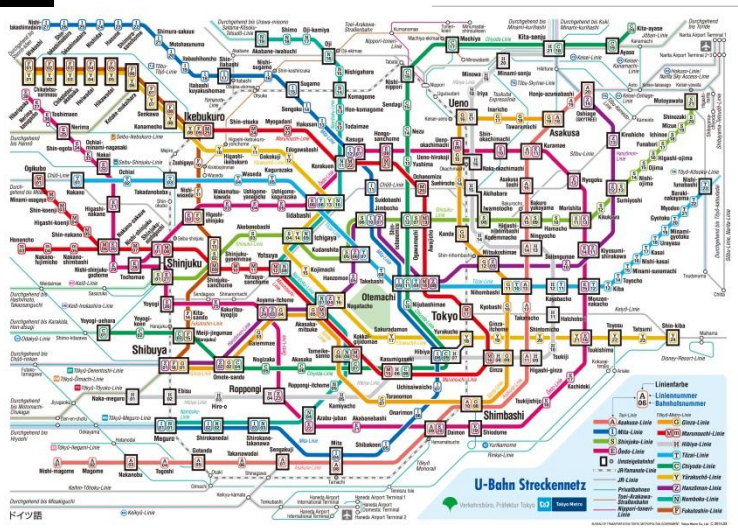
- Intuitive Darstellung von Informationen



Visualisierung des Internets



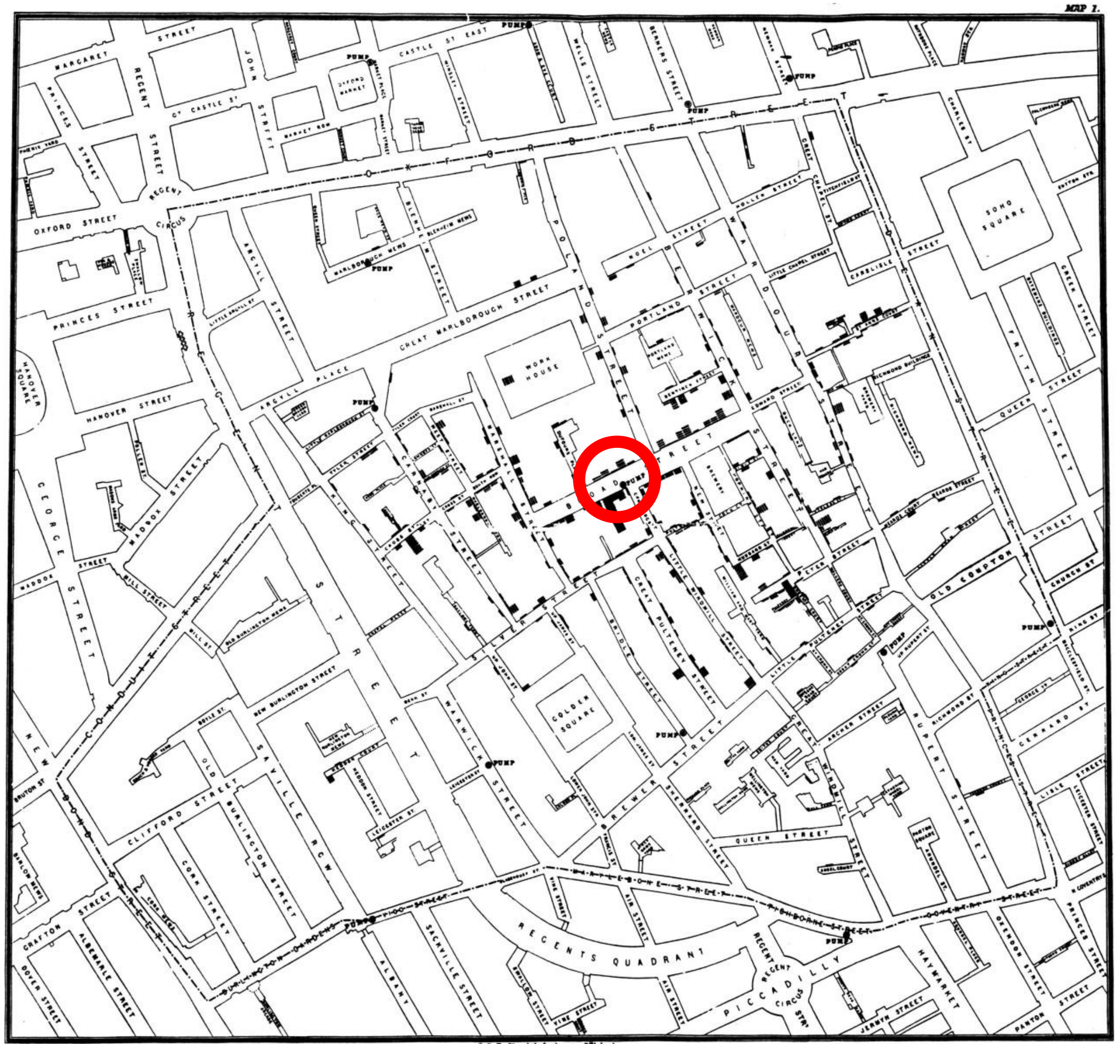
Darstellung der Weltbevölkerung 2011



U-Bahn Plan Tokio

http://www.tokyometro.jp/en/subwaymap/pdf/routemap_de.pdf
http://de.wikipedia.org/wiki/Weltbevölkerung
http://de.wikipedia.org/wiki/Internet

Informationsvisualisierung in der Medizin

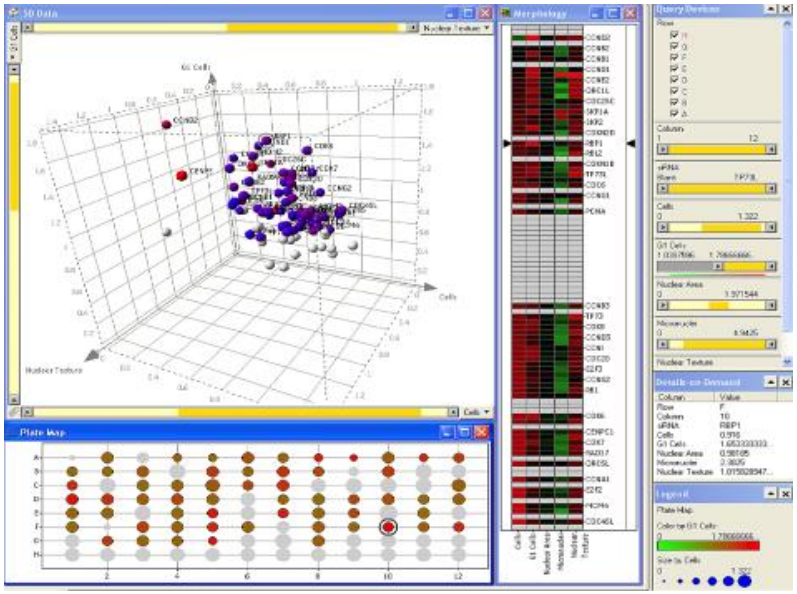


https://en.wikipedia.org/wiki/1854_Broad_Street_cholera_outbreak

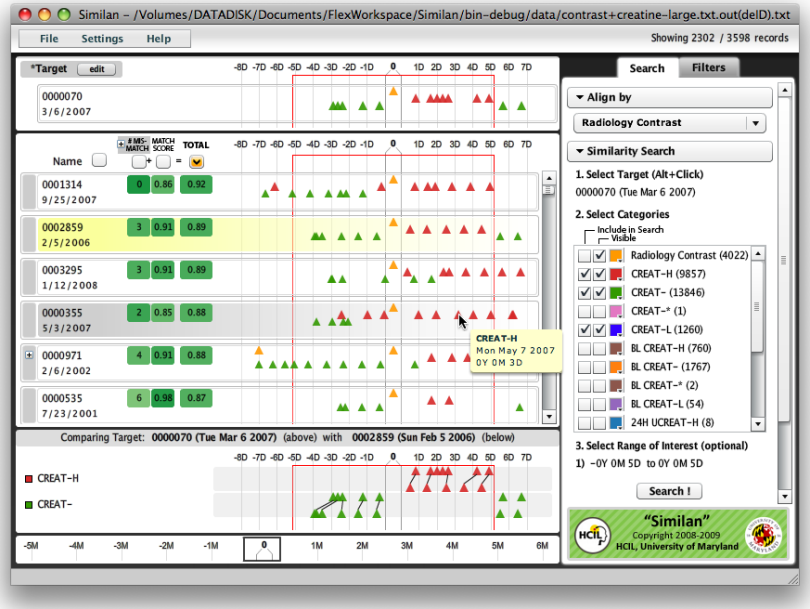
1854: Cholera map (John Snow)

Informationsvisualisierung in der Medizin

- Z.B. Visualisierung von Zeitreihen, Visualisierung in der Epidemologie



Visualisierung mehrdimensionaler Daten

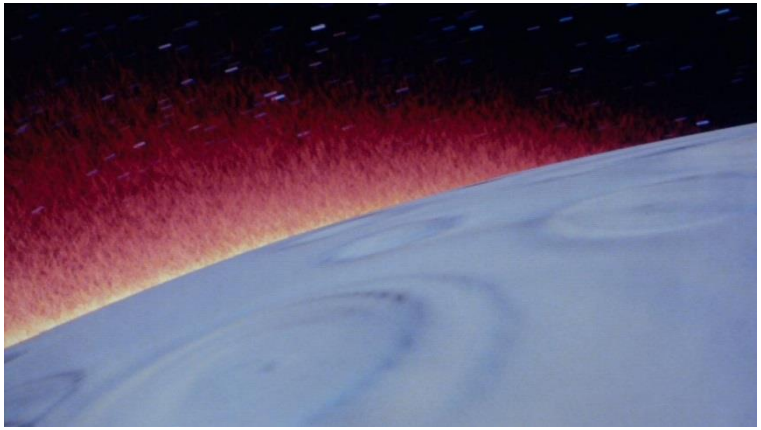


Visualisierung von Zeitreihendaten

http://uxmag.com/articles/beyond-the-medical-chart
http://www.cs.umd.edu/hcil/temporalviz/

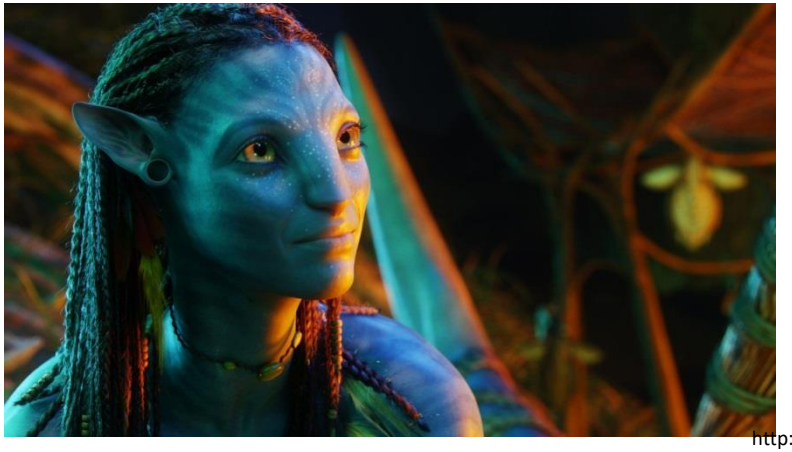
Unterhaltung: Filme

- Computer Generated Imagery (CGI) in Filmen



Star Trek II, erste CGI-Szene in einem Spielfilm, 1982

Planet der Affen: Revolution, 2014



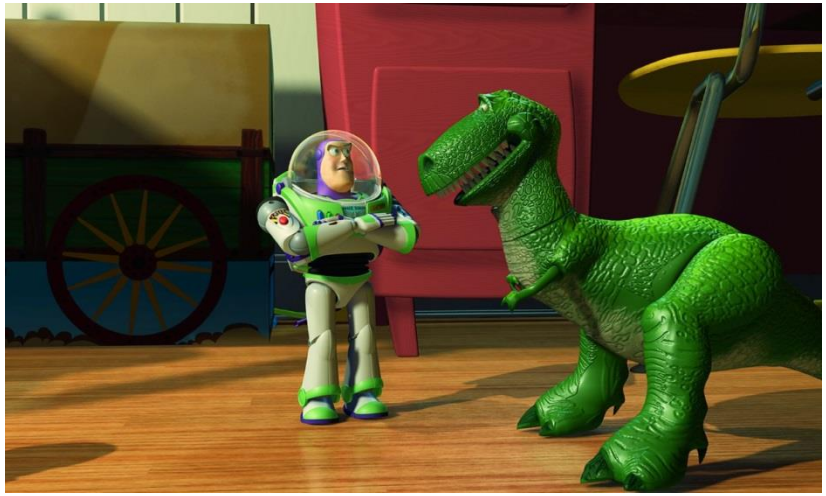
Avatar, 2009

Jungle Book, 2016

http://de.memory-alpha.org/wiki/CGI
http://www.imdb.com/title/tt0499549/
http://www.welt.de/wirtschaft/webwelt/article133262129/Geschichte-der-Hollywood-Bilder-aus-dem-Computer.html
https://www.inverse.com/article/14351-how-the-jungle-book-made-its-animals-look-so-real-with-groundbreaking-vfx

Unterhaltung: Filme

- Animationsfilme



Toy Story 1, Pixar, 1995. Erster vollständig am Computer animierter Spielfilm

Ice Age, 2002

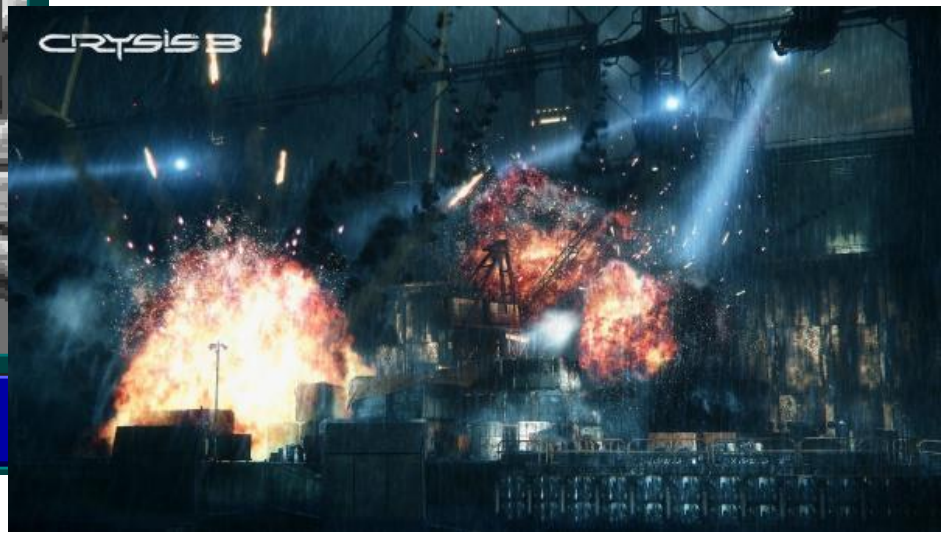
- Vorreiter Pixar unter Steve Jobbs als Geschäftsführer
  - Pixars RenderMan Industriestandard für Rendern von CGI/3D-Animation

- http://www.imdb.com/title/tt0114709/
- http://www.imdb.com/title/tt0268380/

Unterhaltung: Computerspiele

- Heute meist basierend auf einer 3D-Grafik-Engine zum Rendern der Szenen (z.B. CryEngine, Frostbite)
  - geometrische Objektbeschreibung, Oberflächentexturen, Licht und Schatten (Shading), Transparenz, Spiegelungen, physikalische Effekte usw.

Wolfenstein 3D, 1992



Crysis 3, 2013

- http://www.mobygames.com/game/wolfenstein-3d/screenshots
- http://www.gamestar.de/spiele/crysis-3/bilder/screenshots/crysis_3,46254,94722.html

Unterhaltung: Computerspiele

- Simulationsspiele sind professionellen Simulatoren in der 3D-Grafik mittlerweile fast gleichwertig

X-Plane 10

Gran Turismo 6

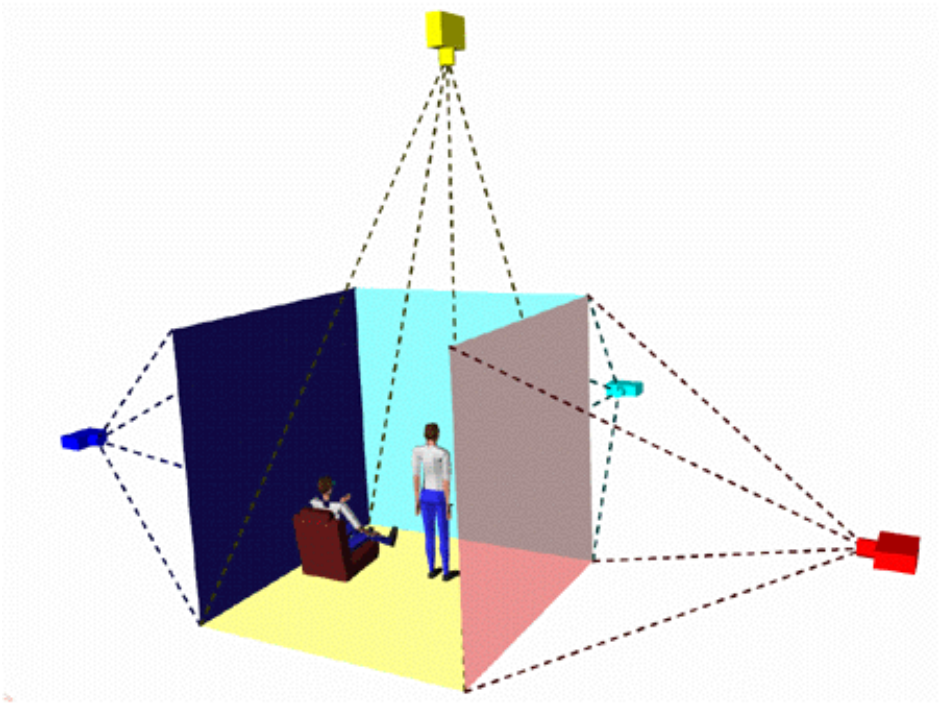
- http://www.x-plane.com/desktop/multimedia/
- http://www.computerbild.de/fotos/Gran-Turismo-6-Das-wuenscht-sich-die-Redaktion-8293248.html

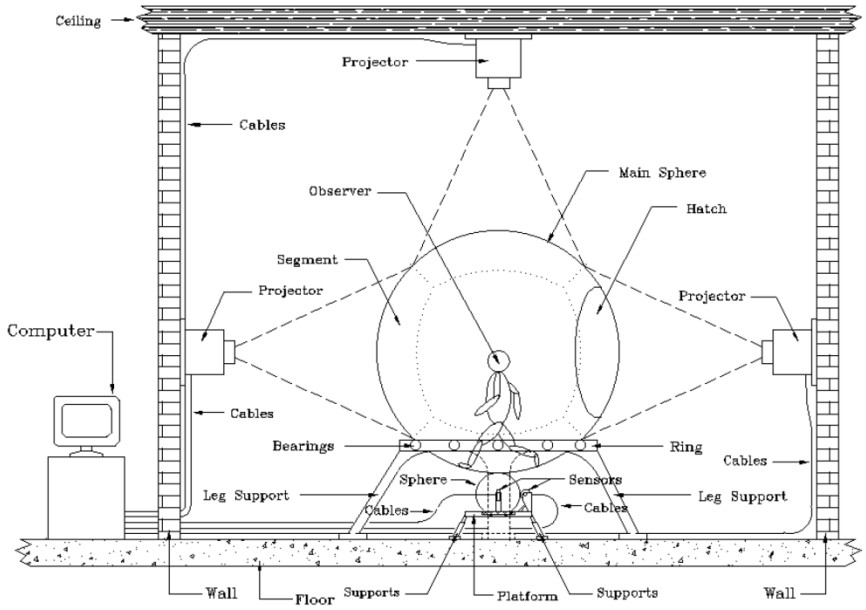
Virtual Reality

- **Darstellung** + Wahrnehmung vollständig computergenerierter virtueller Umgebungen
- **Interaktion** über Eingabegeräte
  - Z.B. Datenhandschuh/Controller, 3D Maus, optisches Tracking
- **Rückkopplung** durch Force Feedback
- Herausforderungen für ein „realistisches Erlebnis“:
  - Darstellung eines großen Sichtfeldes
  - Hohe Bildauflösung
  - Schnelle Reaktionszeit des Displays < 3ms
  - Hohe Bildwiederholfrequenz
  - Hardware zur Darstellung, bei Head Mounted Display z.B. optische Herausforderungen
  - Genaues Tracking der Bewegungen der Person in der Szene
  - Geringe Latenzzeit zwischen Eingabe und Darstellung des Bildes (<25 ms)

CAVE

- = Cave Automatic Virtual Environment
- Stereoprojektion auf umgebende Wände eines Raumes

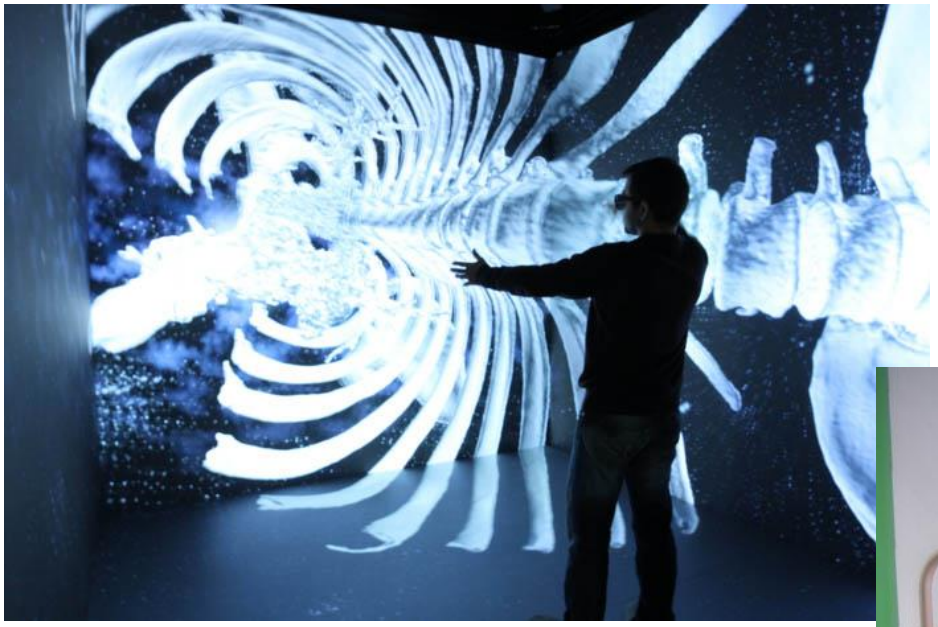




Cybersphere

- http://escience.anu.edu.au/lecture/cg/Display/cave.en.html
- K.J. Fernandes, V. Raja, J. Eyre: „Cybersphere: The Fully Immersive Spherical Projection System“, Communication of the ACM 46, 2003, 141-147

CAVE: Beispiele



Wanderung durch den Körper, Steuerung per Gesten

Planung des Einbaus von Kabinenelementen in ein Flugzeug

- http://phys.org/news/2013-04-high-performance-cave-automatic-virtual-environment.html
- http://www.lufthansa-technik.com/de/virtual-cabin-visit

Head mounted display

- Darstellung der Szene…
  - … auf augennahem Bildschirm
  - … oder auf die Netzhaut projiziert
- Enthält Sensoren zur Bewegungserfassung
- Heute gängige Spezifikation
  - Sichtfeld: ~90°, teilweise bis 180°
  - Auflösung: 1280x1024 und höher
  - Gewicht: <1kg

Virtuelles Fallschirmspringen als Training

Anwendung in der Chirurgie

http://de.wikipedia.org/wiki/Head-Mounted_Display
http://www.innovationtoronto.com/2013/08/sonys-head-mounted-3d-video-display-gives-surgeons-an-inside-view/

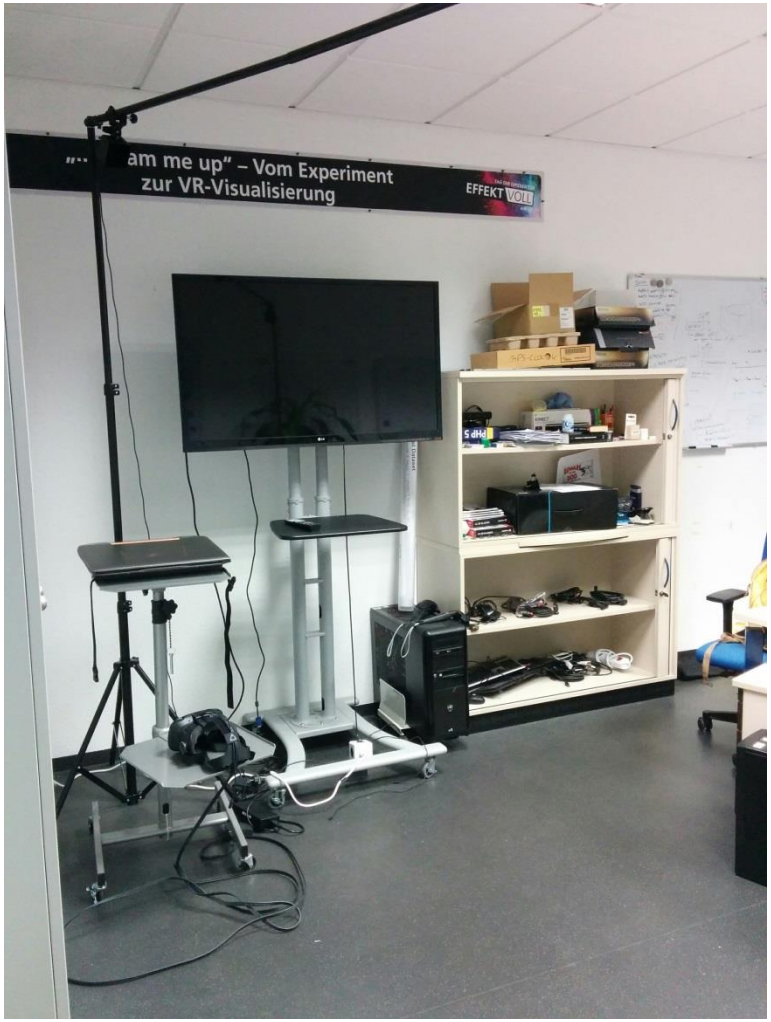
Head mounted display

- „Revival“ durch Kombination mit Smartphones

https://www.youtube.com/watch?v=-4vhtpa8JRA

VR Lab am KIT/IPE

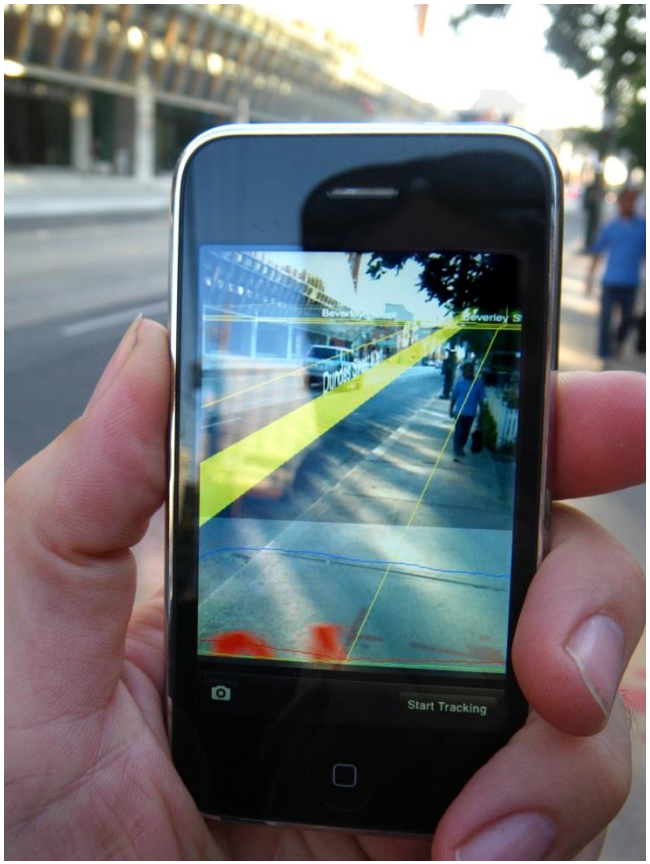
- HTC Vive
- ROG Laptop (GTX 1070)



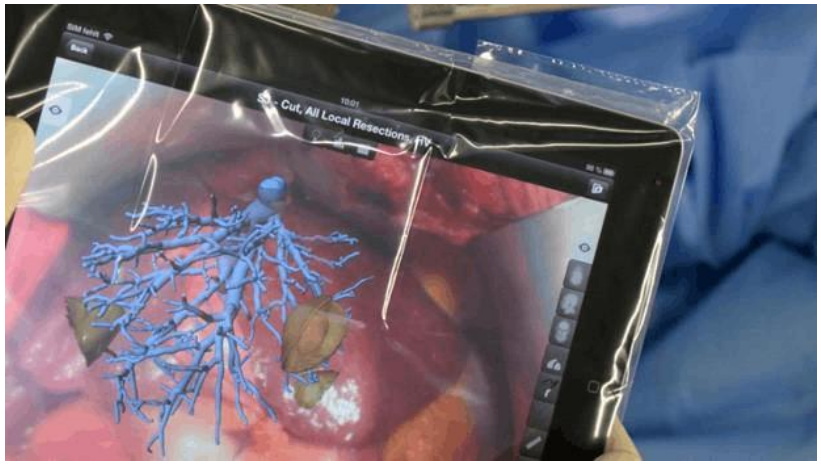
Augmented Reality

- Computergestützte **Erweiterung** der Realitätswahrnehmung
  - ≠ Virtual Reality!
- Kombination von realen Bildern und Computergrafik
  - Oft in Echtzeit
- Herausforderungen:
  - Echtzeitanforderung
  - Nachführung bei Bewegung
  - Geometrische Information über reale Szene notwendig

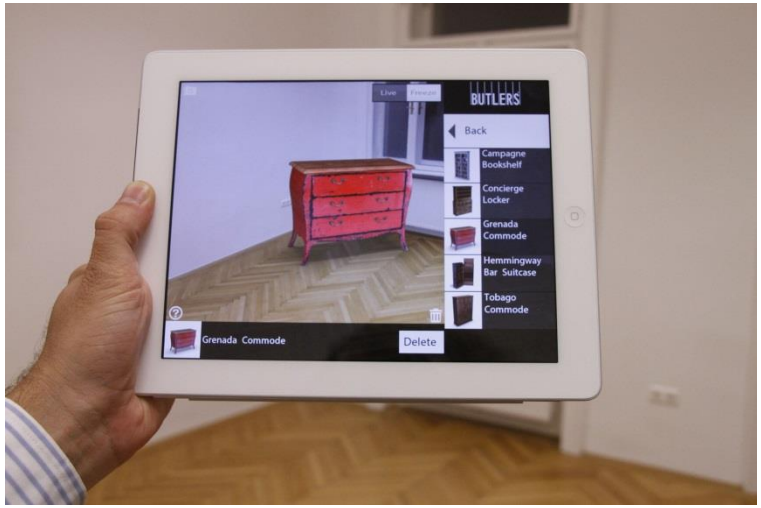
Augmented Reality: Beispiele



Navigation am Smartphone



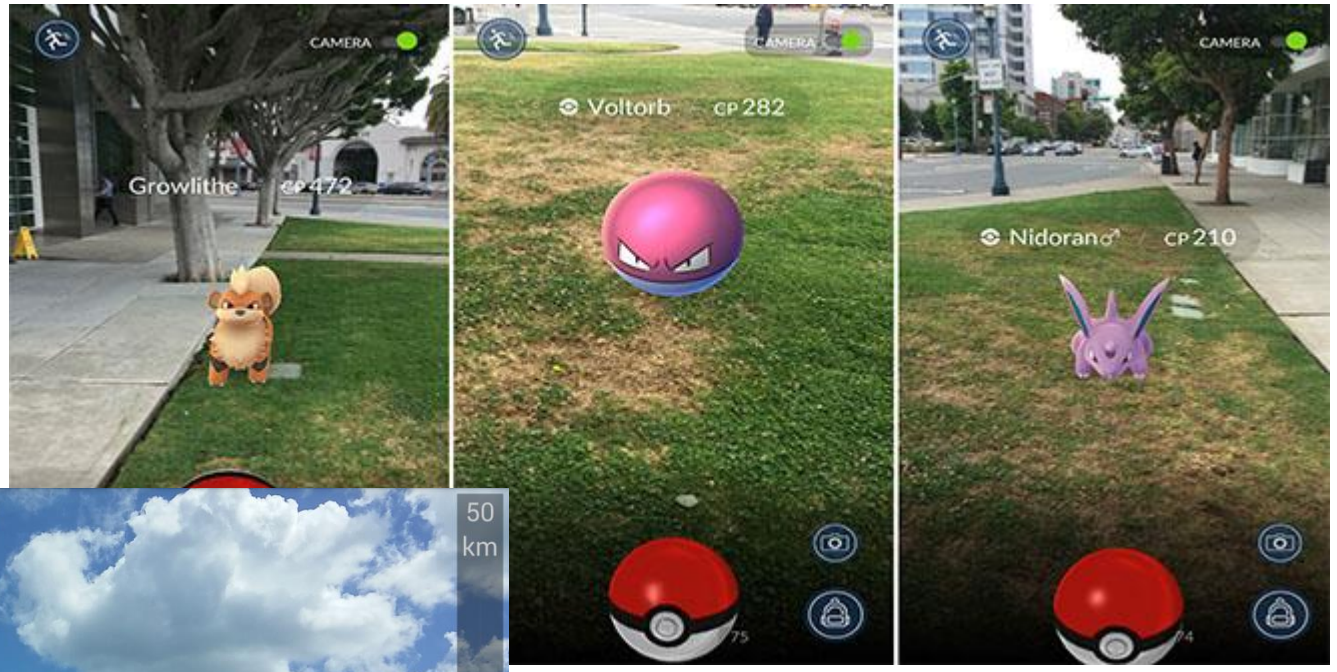
Überlagertes Lebervenen-Modell bei OP



Kamerabild und virtuelles Möbelstück

- http://de.wikipedia.org/wiki/Erweiterte_Realität
- http://www.computerwoche.de/a/augmented-reality-app-hilft-leberchirurgen-bei-der-op,2544964

Augmented Reality: Beispiele



Pokémon Go

Flightradar 24

http://www.vrguru.com/2016/07/07/pokemon-go-now-available-android-ios/
http://www.flightradar.org/flightradar-24-pro-for-iphone-and-ipad-devices-ipad-devices/

Augmented Reality: Beispiele

https://www.youtube.com/watch?v=29OSzQbxpcQ

Augmented Reality: Beispiele

https://www.youtube.com/watch?v=vDNzTasuYEw

Echtzeit?

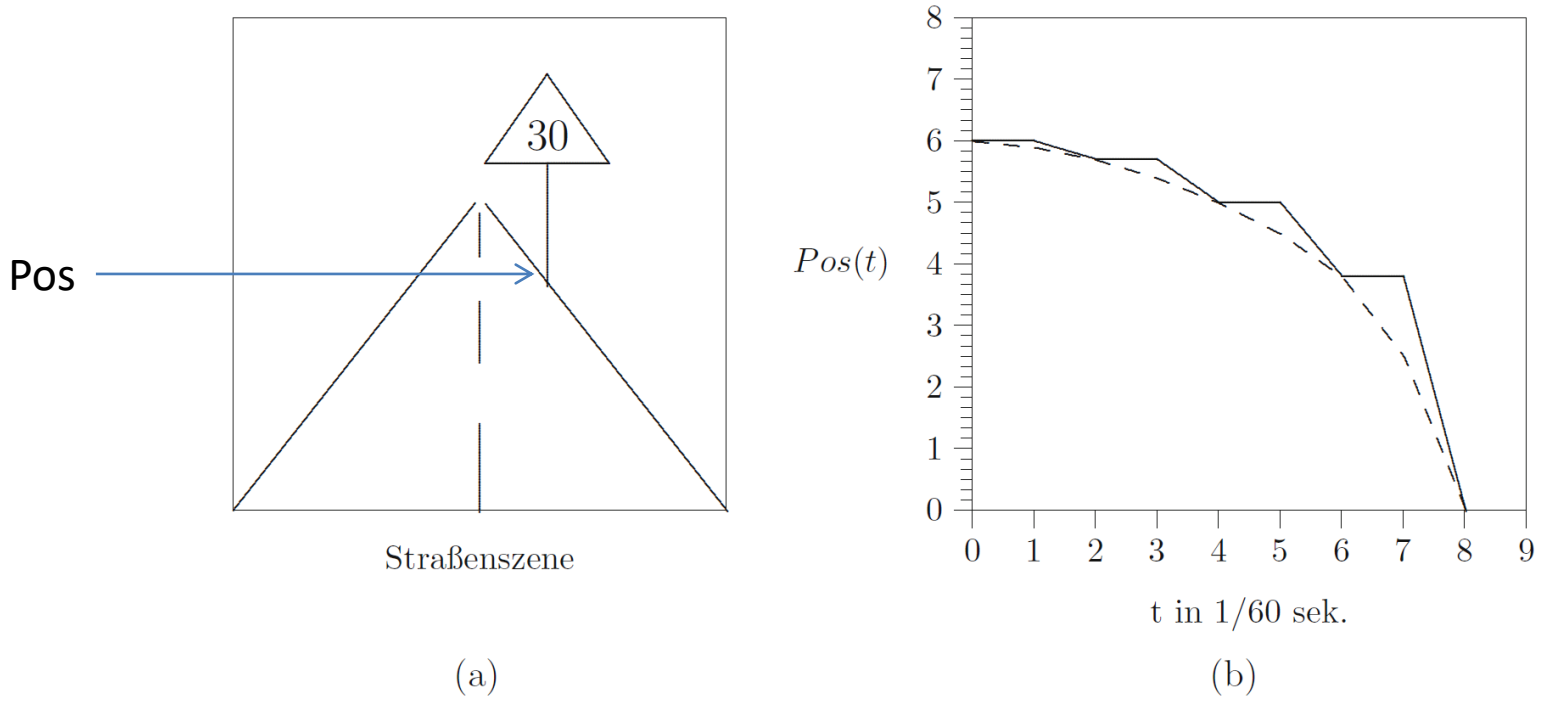
- Interaktive/Echtzeit-CG ⇔ Nicht-Echtzeit-CG
  - Anwendungsabhängig – z.B. Interaktion mit Benutzer?
  - Anforderung an Reaktionszeit bestimmt u.U. Komplexität eines Modells

  - Grenzen verwischen durch gesteigerte Rechenleistung zunehmend

- Bildwiederholfrequenz
  - Anzahl der Bilder pro Sekunde in ***Hz*** oder ***fps***
  - Flüssiger Bildeindruck ab ~30Hz

Bildgenerierrate, Bildanzeigerate

- Oft feste Bildanzeigerate (z.B. Monitor 60Hz)
- Zwischenspeicher im Anzeigegerät
- Doppelbilder wenn Bildgenerierrate < Bildanzeigerate



Nischwitz et al.: Computergrafik und Bildverarbeitung Band 1, Computergrafik, 3. Auflage, Vieweg+Teubner

Zeilensprungverfahren (Interlace)

- Erhöhung der Bildwiederholfrequenz: zeilenweiser Bildaufbau

Halbbild A Halbbild B

- Beispiel: analoges Fernsehen (PAL Fernsehnorm)
  - Übertragung von 50 Halbbildern/Sekunde → 25 Vollbilder/Sekunde
- Vorteile:
  - Erhöhung der zeitlichen Auflösung bei gleicher Datenmenge
  - Reduktion des subjektiven Flimmereindrucks/Doppelbildern
- Nachteile:
  - Artefakte (z.B. an horizontalen Kanten)

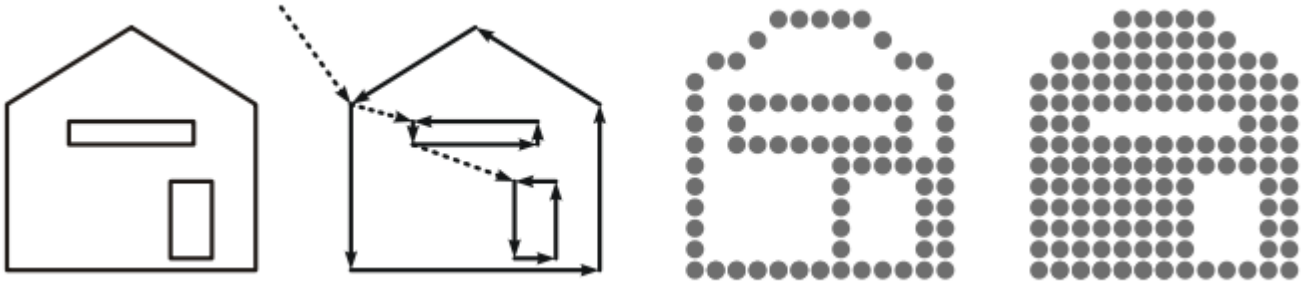
Vollbildverfahren (Progressive Scan)

- Übertragung von vollständigen Bildern
  - Bsp: zeilenweiser Bildaufbau bei Röhrenmonitoren

- Vorteile:
  - Schärferes, ruhigeres Bild, reduzierte Artefakte
- Nachteile:
  - Mehr Daten oder geringere Bildwiederholfrequenz

Dimensionalität, Raster- / Vektorgrafik

- Dreidimensionale Grafik
  - Beschreibung einer 3D-Szene
  - Rendern eines 2D Bildes zur Anzeige auf dem Bildschirm (Bildsynthese)
- Zweidimensionale Grafik
  - Raster- ⇔ Vektorgrafik

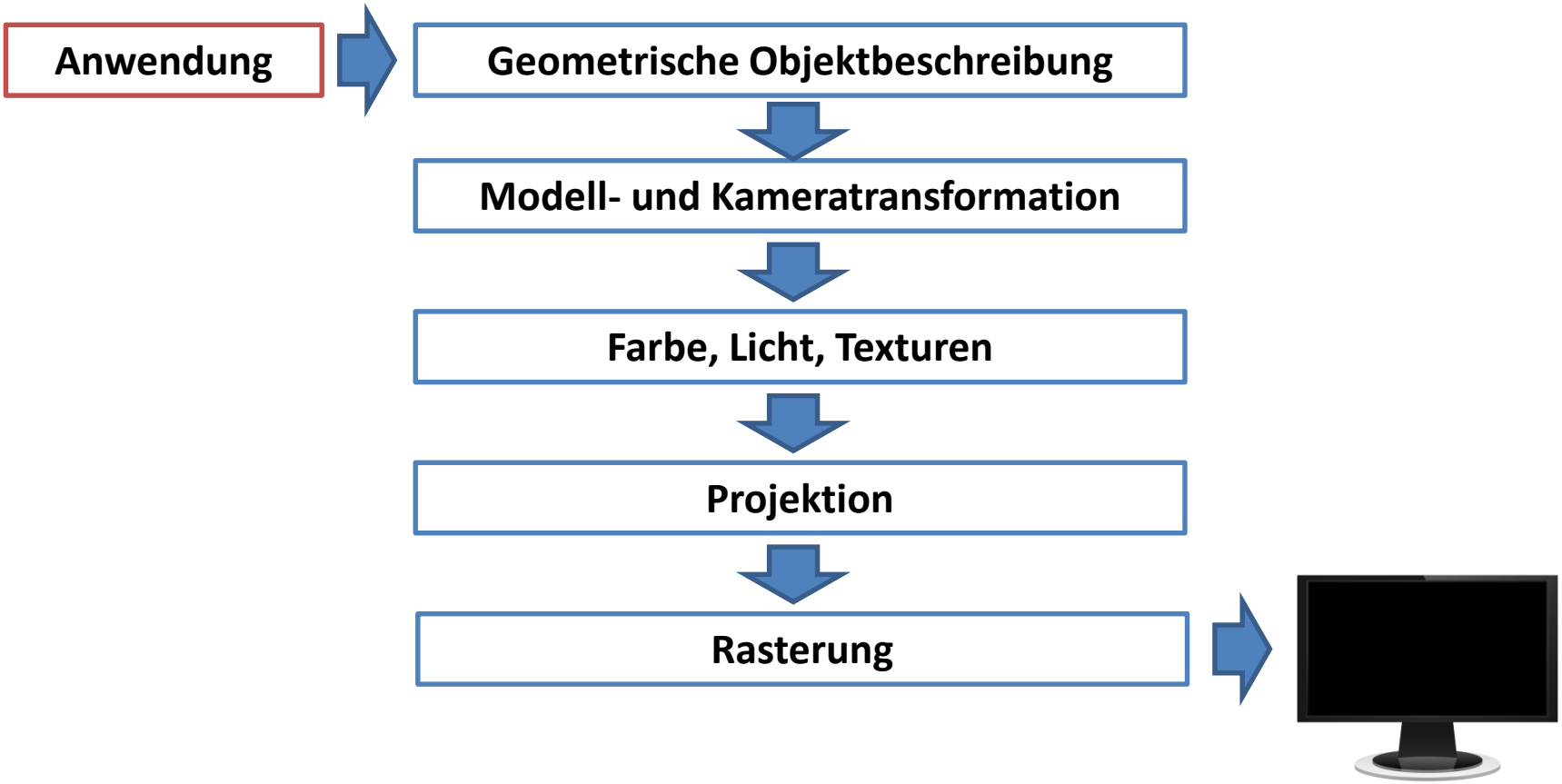


  - Heute nur noch Rasterbildschirme: Rastern von Szenen

    → Rasteralgorithmen

http://de.wikipedia.org/wiki/Computergrafik

Vereinfachte Grafikpipeline



Themenübersicht

1. **Einführung**
2. Programmierbibliotheken / OpenGL
3. Geometrische Repräsentation von Objekten
4. Koordinatensysteme und Transformationen
5. Zeichenalgorithmen
6. Buffer-Konzepte
7. Farbe, Beleuchtung und Schattierung
8. Texturen
9. Animationen
10. Raytracing
11. Volumenvisualisierung

1.2 EXKURS: KURZER GESCHICHTLICHER ABRISS

Geschichtlicher Abriss

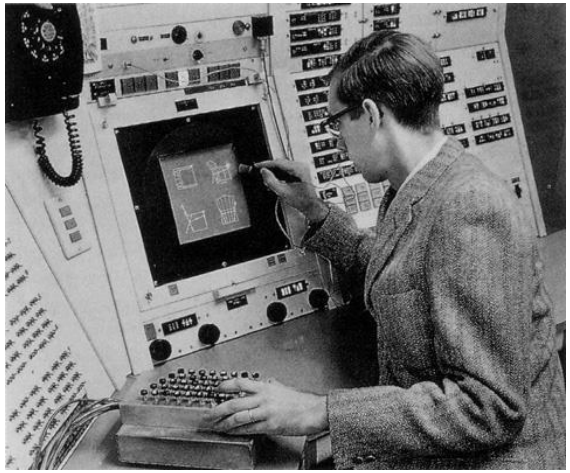
- Bis 1950: Grafikausgabe per Plotterzeichnung
- 1950: Whirlwind MIT. Erster Einsatz einer Kathodenstrahlröhre als Bildschirm
- 1952: Erstes Videospiel „Tennis for two“

- 1959: Erstes CAD-System (IBM DAC-1)

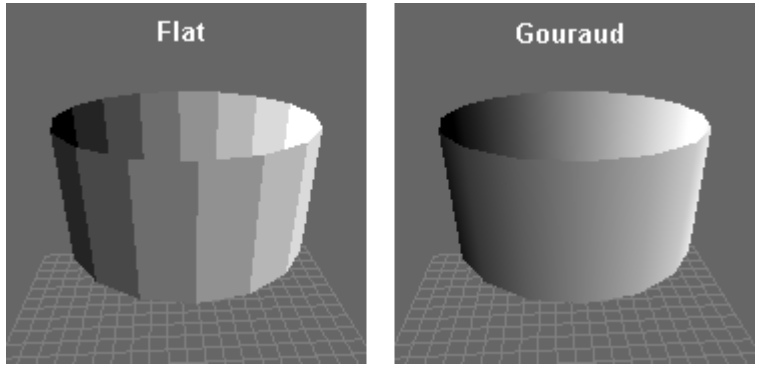
http://design.osu.edu/carlson/history/tree/images/dac.JPG
http://de.wikipedia.org/wiki/Tennis_for_Two

Geschichtlicher Abriss

- 1963: Beginn der modernen Computergrafik: Sketchpad
  Ivan Sutherland „Sketches and Systems“, MIT.
- 1962: Bresenham-Algorithmus zum Rastern von Linien
- 1965: CAD in der Flugzeugindustrie (Lockheed)
- 1968: Erste speicherröhren-basierte grafische Computerterminals (Computer Displays, Tektronix)
- 1968: Gründung von Evans & Sutherland



- 1971: Gouraud Shading (Henri Gouraud)



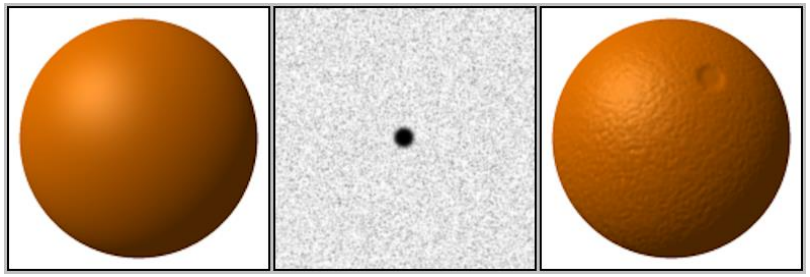
- 1974: Depth-Buffer (Edwin Catmull)

http://www.mprove.de/diplom/text/3.1.2_sketchpad.html
http://en.wikipedia.org/wiki/Gouraud_shading

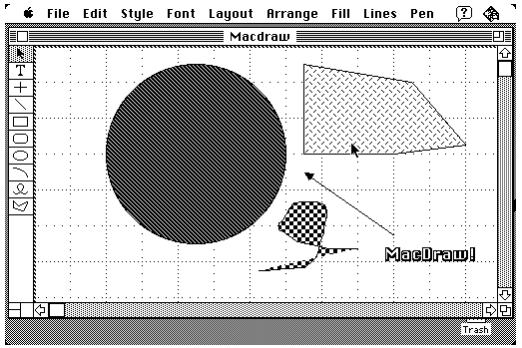
Geschichtlicher Abriss

- 1974: Anti-Aliasing (Herbert Freeman)
- 1975: Phong-Shading, Phong-Beleuchtungsmodell (Bui-Toung Phong)
- 1976: Reflection Mapping (Jim Blinn)

- 1978: Bump Mapping (Jim Blinn)

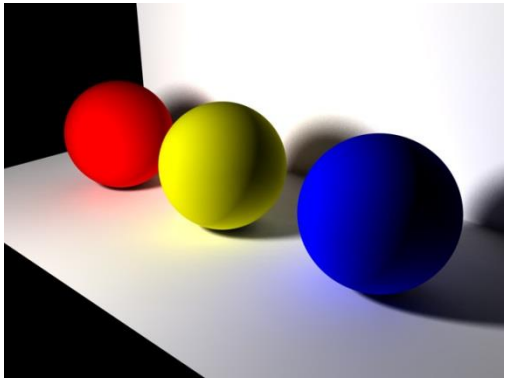


- Ca. 1979/1980: Raytracing für Reflexionsberechnungen
- Ende 70er/Anfang 80er: erste Spielfilme mit CGI-Anteil
- 1981: Gründung von Silicon Graphics
- 1980er: Verbreitung von grafikfähigen PCs
- 1984: MacDraw, MacPaint: Consumer-Markt!



http://de.wikipedia.org/wiki/Computergrafik
http://de.wikipedia.org/wiki/MacDraw

Geschichtlicher Abriss



- 1984: Radiosity (Cornell University)
- 1980er: erste Standards für 2D und 3D Computergrafik
  - PHIGS: Programmer's Hierarchical Interactive Graphics System, API für 3D Szene
- 1986: Gründung von Pixar, 1989 erste Version von RenderMan
- 1986: Veröffentlichung der Rendergleichung und von Path Tracing
  - Mathematisches Fundament für globale Beleuchtung

- 1992: OpenGL 1.0 Standard wird von Silicon Graphics veröffentlicht
- 1995: Erster vollständig als Computeranimation erstellter Kinofilm *Toy Story*

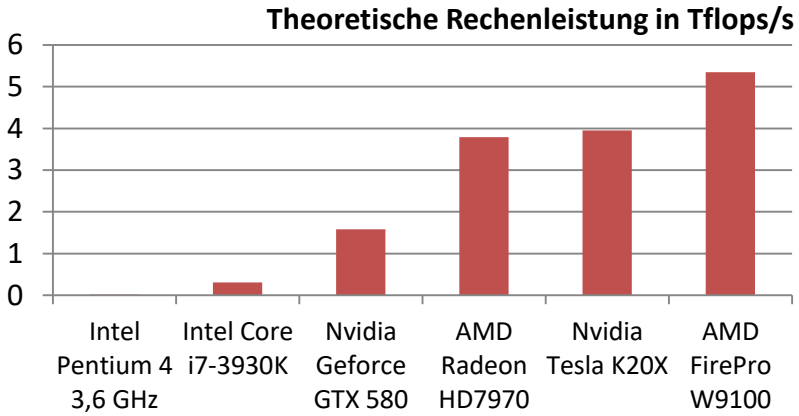
http://de.wikipedia.org/wiki/OpenGL
http://de.wikipedia.org/wiki/Radiosity_%28Computergrafik%29

Geschichtlicher Abriss

- Mitte der 90er Jahre
  - 3D Game Engines, z.B. Unreal Engine
  - Computergrafik im Internet
- 1996: 3dfx Vodoo Graphics. Erster 3D-Grafikchip im nicht-professionellen Bereich
  - 3D Grafik erobert den Consumer-Markt

- 2001: erster komplett computeranimierter Spielfilm mit realistischen Charakteren *Final Fantasy.*
- Ab Mitte der 2000er Jahre: extreme Leistungssteigerung der Grafikkarten
  - Shader-Programmierung
  - General Purpose Computation

- 2004: OpenGL Shading Language



http://de.wikipedia.org/wiki/Non-photorealistic_Rendering

ZUSAMMENFASSUNG

Zusammenfassung

- Generative Computergrafik: Erstellung künstlicher Bildwelten
- Typische Anwendungsgebiete:
  - Ausbildungssimulation
  - CAD/CAM
  - Visualisierung
  - Informationsvisualisierung
  - Unterhaltung
- Virtual Reality ⇔ Augmented Reality
- Echtzeitanforderungen ⇔ Realitätstreue
- Grafikpipeline: typische Verarbeitungsschritte von der Anwendung/Modell zur Darstellung auf dem Bildschirm

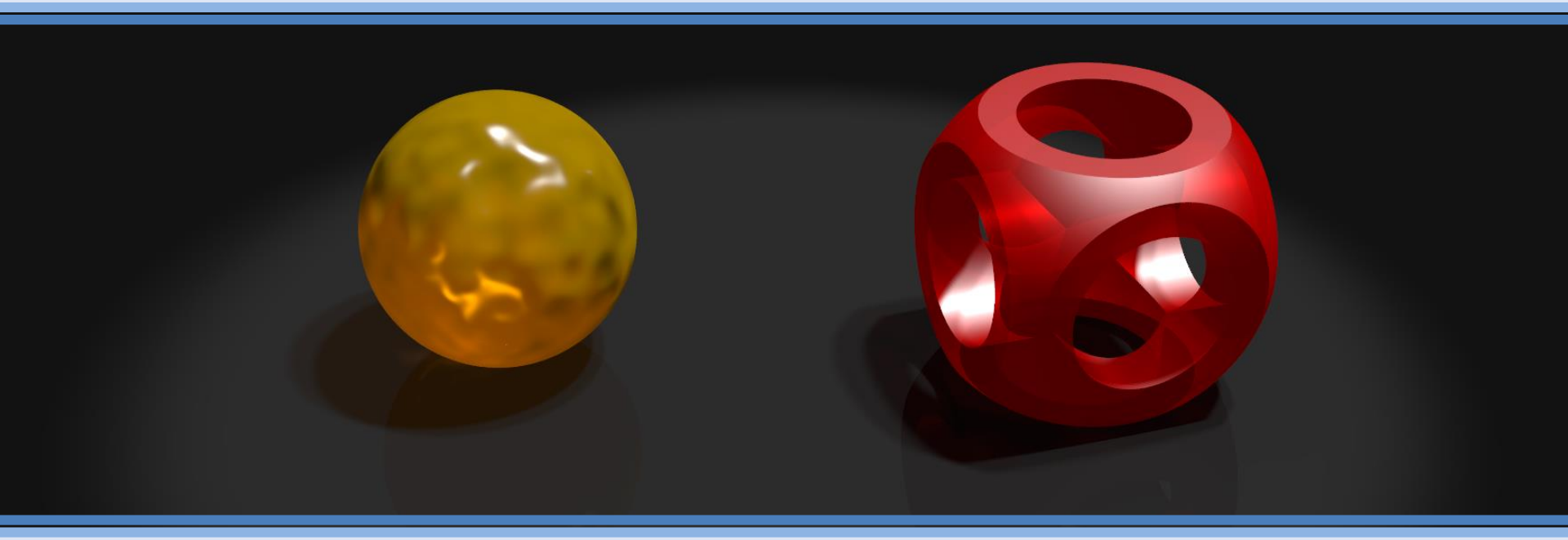
ÜBUNGSAUFGABEN

Übungsfragen Kapitel 1

- Beschreiben Sie die Begriffe „Virtual Reality“ und „Augmented Reality“ – was ist der Unterschied?
- Was ist der Unterschied zwischen Bildgenerierrate und Bildanzeigerate?

Computergrafik

T. Hopp



Themenübersicht

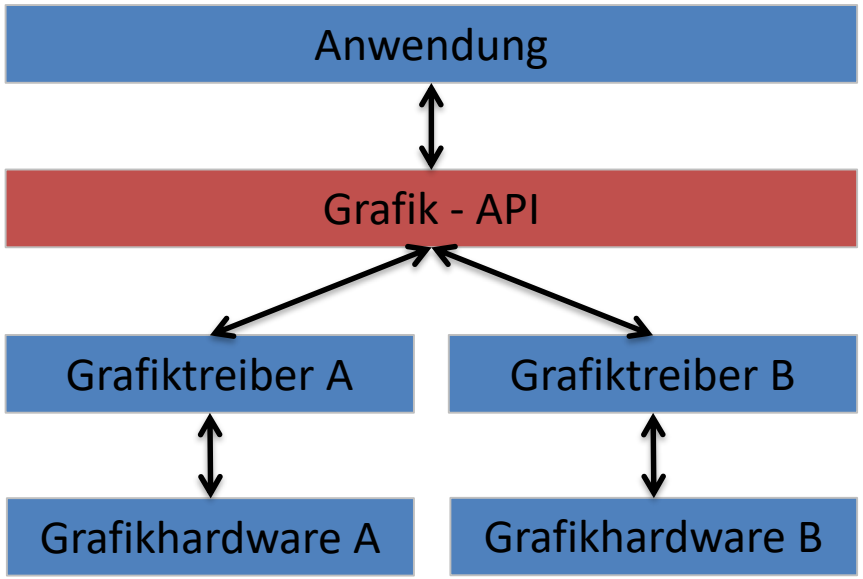
1. Einführung
2. **Programmierbibliotheken / OpenGL**
3. Geometrische Repräsentation von Objekten
4. Koordinatensysteme und Transformationen
5. Zeichenalgorithmen
6. Buffer-Konzepte
7. Farbe, Beleuchtung und Schattierung
8. Texturen
9. Animationen
10. Raytracing
11. Volumenvisualisierung

2.1 PROGRAMMIERBIBLIOTHEKEN

Mittel zur Abstraktion

- Entwicklung von GPUs zur Beschleunigung
- Spezifische Instruktionen, Schnittstellen für verschiedene Hersteller, Generationen etc.

  ➔ API zur **Abstraktion des Hardwarezugriffs**



Grafik-API Standards

- Beispiele für international normierte Standards (ISO):
  - GKS / GKS-3D: Graphical Kernel System
  - PHIGS / PHIGS+: Programmer‘s Hierarchical Interactive Graphics System

- Sehr allgemein und schwerfällig
- Nutzen Möglichkeiten der Beschleunigung durch Grafikkarten nicht aus
- Haben sich in der Praxis nicht durchgesetzt

De-facto-Standards der Industrie:

- **Direct3D** (Microsoft, ***proprietär***)
- **OpenGL** (ursprünglich SGI, ***open source, plattformunabhägig***)
  - Heute: OpenGL Architecture Review Board Working Group

Evolution der Programmierbibliotheken

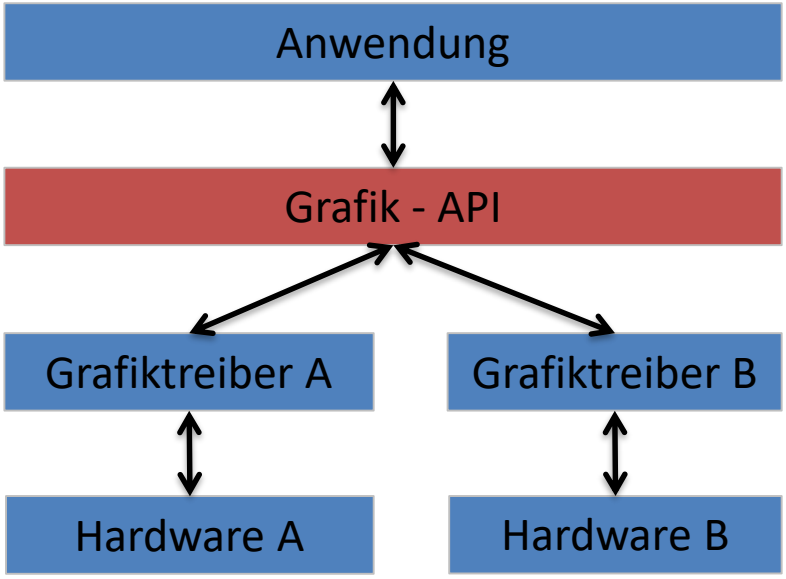
- Fixed Function
  - Konfigurierbare Module für z.B. Beleuchtung, Puffer etc.
  - Zusammensetzbar zur Grafik-Pipeline
  - „Klassisches“ OpenGL, Direct3D
- Shader
  - Teile der Grafik-Pipeline können frei definiert werden
  - Z.B. Implementierung eines eigenen Beleuchtungsmodells
  - OpenGL Shading Language (GLSL), C for graphics (Cg), High Level Shading Language (HLSL)
- Programmierbare Pipeline
  - Volle Freiheit in der Programmierung von GPUs mit Nutzung der vollen Beschleunigungsmöglichkeiten
  - OpenCL, CUDA

Historische Entwicklung

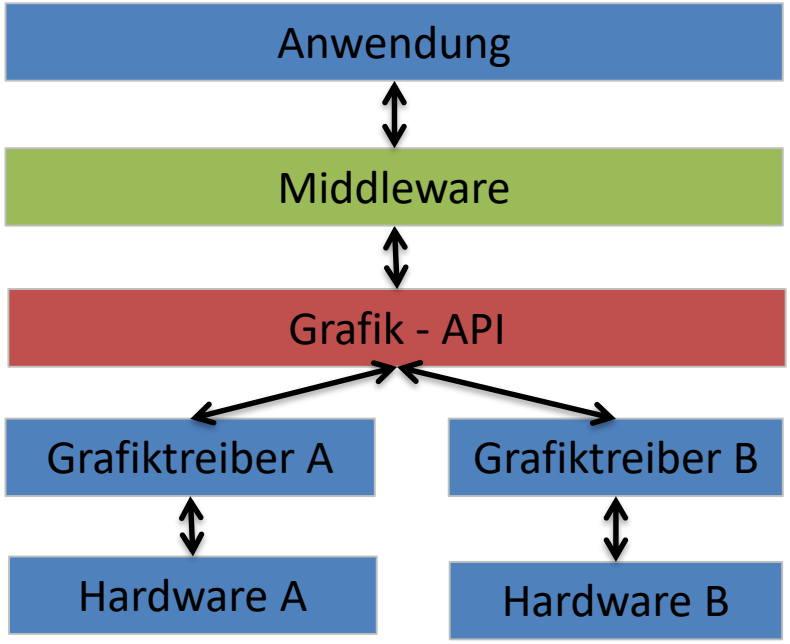
Jahr	Normen	Industrie-Standards		Shading-Sprachen		GPGPU	
1982		IrisGL (SGI)					
1985	GKS						
1988	GKS-3D						
1989	PHIGS						
1992		OpenGL 1.0					
1995			Direct3D 1.0				
1997		OpenGL 1.1	Direct3D 5.0				
1999		OpenGL 1.2	Direct3D 7.0				
2001		OpenGL 1.3	Direct3D 8.0				
2002		OpenGL 1.4	Direct3D 9.0		Cg 1.0		
2003		OpenGL 1.5	Direct3D 9.0a	GLSL 1.0	Cg 1.1		
2004		OpenGL 2.0	Direct3D 9.0b	GLSL 1.1	Cg 1.3		
2006		OpenGL 2.1	Direct3D 9.0c	GLSL 1.2	Cg 1.5		
2007			Direct3D 10.0		Cg 2.0		CUDA 1.0
2008		OpenGL 3.0	Direct3D 10.1	GLSL 1.3	Cg 2.1		CUDA 2.0
2009		OpenGL 3.2	Direct3D 11.0	GLSL 1.5	Cg 2.2	OpenCL 1.0	CUDA 3.0
2010		OpenGL 4.1		GLSL 4.1	Cg 3.0	OpenCL 1.1	CUDA 3.2
2012		OpenGL 4.3		GLSL 4.3	Cg 3.1		CUDA 5.0
2013			Direct3D 11.2	GLSL 4.4		OpenCL 2.0	CUDA 6.0
2014		OpenGL 4.5		GLSL 4.5			
2015			Direct3D 12.0			OpenCL 2.1	CUDA 7.0
2016							CUDA 8.0
2017		OpenGL 4.6		GLSL 4.6		OpenCL 2.2	CUDA 9.1

Architekturmodelle

Immediate-mode (IM)

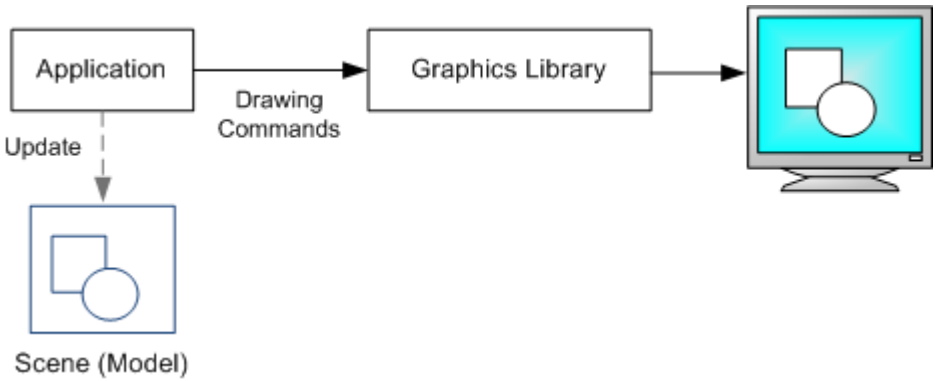


Retained-mode (RM)



Architekturmodelle

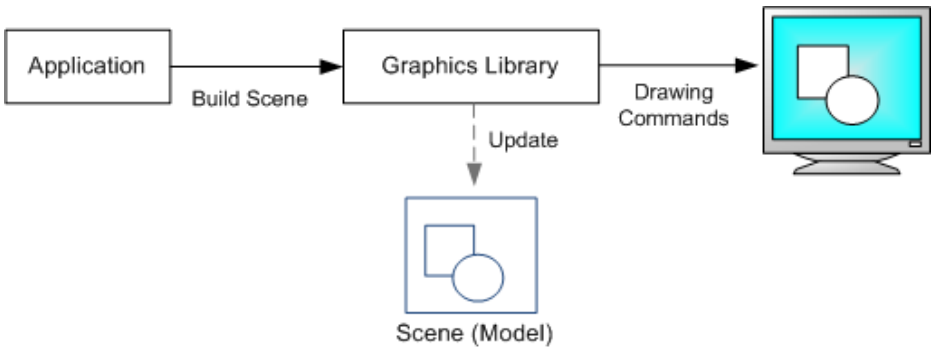
- Immediate Mode



- Minimaler Overhead
- Anwendung verantwortlich für Szene

z.B. OpenGL, Direct3D

- Retained Mode



- Vereinfachtes Szenen-Management in der Anwendung
- Geringere Performanz

z.B. Java3D (als Middleware zwischen Applikation und OpenGL / Direct3D)

https://msdn.microsoft.com/en-us/library/windows/desktop/ff684178%28v=vs.85%29.aspx

2.2 OPEN GL

Programmierinterface OpenGL

- Definition Architecture Review Board:

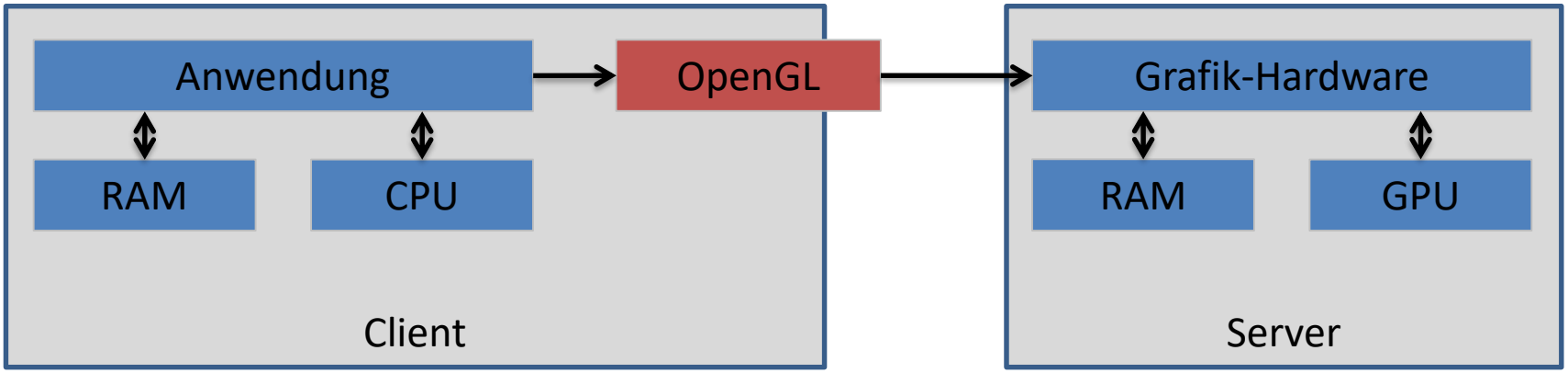
  „ein Software-Interface zur Hardware. Zweck ist die Darstellung von zwei- und dreidimensionalen Objekten mittels eines Bildspeichers.“

- Sprachunabhängig: C/C++, FORTRAN, …
- Betriebssystem- und Windowsystem-unabhängig
- Weite Unterstützung durch Hardwarehersteller

- In Anwendung verantwortlich für das Rendern einer Szene
  - Implementierung des OpenGL-API erfolgt in der Regel durch Systembibliotheken oder Grafikkarten-Treiber
- Immediate Mode: Anwendung definiert und verwaltet Szene

Programmiermodell

- Client-Server-Modell



- Serverseite: Treiber für Hardware setzt OpenGL in Bilder um
- Emulation auch in Software möglich
- „Extensions“: Hardware-spezifische Erweiterung möglich

Versionen

- OpenGL 1.x: Fixed Function Pipeline
- OpenGL 2.x: Shader
  - OpenGL Shading Language (GLSL): Vertex + Fragment Shader
- OpenGL 3.x: weitere Shader
  - OpenGL Shading Language (GLSL): Geometry Shader + Deprecation
  - Ab 3.2: „Compatibility Profile“ und „Core Profile“
- OpenGL 4.x: Programmierbare Rendering Pipeline
  - OpenCL
  - Shader: Tesselation

- Varianten:
  - „OpenGL ES“ (Embedded Systems): z.B. Smartphones
  - „WebGL“: 3D-Inhalte in Browsern

Bibliotheken

- **GL**: Low-Level-API (Basisfunktionalität)
  - Plattformunabhängig: keine Darstellung von Fenstern, kein Behandeln von Benutzereingaben

```
#include <GL/gl.h>
```

- **GLU**: OpenGL Utility Library
  - Erweiterung der Low-Level-API um
    - Vereinfachte geometrische Funktionen
    - High-Level-Primitive (z.B. Kugel, Zylinder, Torus)
    - Non-uniform rational B-Splines (NURBS)
    - Polygon Tesselation
    - Skalieren von Bildern
    - Generieren von Texturen aus Bildern
    - Fehlermeldungen

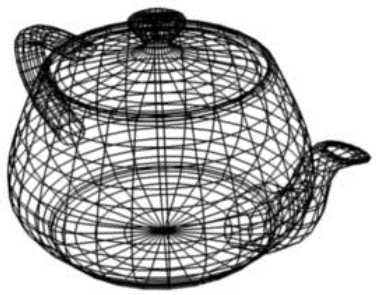
```
#include <GL/glu.h>
```
→ beinhaltet bereits `gl.h`!

Bibliotheken

- **GLUT**: OpenGL Utility Toolkit
  - Erweiterung der Grafik-API von GL und GLU um
    - Fensterverwaltung
    - Callback-Mechanismen (Keyboard, Mouse)
    - Timer-Prozeduren
    - Einfache Menüsteuerungen
    - Highest-Level Primitive (z.B. Teapot)

```
#include <GL/glut.h>
```

→ beinhaltet bereits `glu.h`!



http://battis.net/staging/tag/opengl/

Namenskonventionen

- Bibliotheksaufrufe beginnen mit folgendem Präfix:

`gl`	Alle OpenGL Funktionen, CamelCase-Schreibweise
`glu`	Alle Aufrufe der Utility Bibliothek
`glut`	Alle Aufrufe des Utility Toolkits
`GL_`	Vordefinierte Konstanten der Headerdateien Flags zum Steuern der Modi

Datentypen

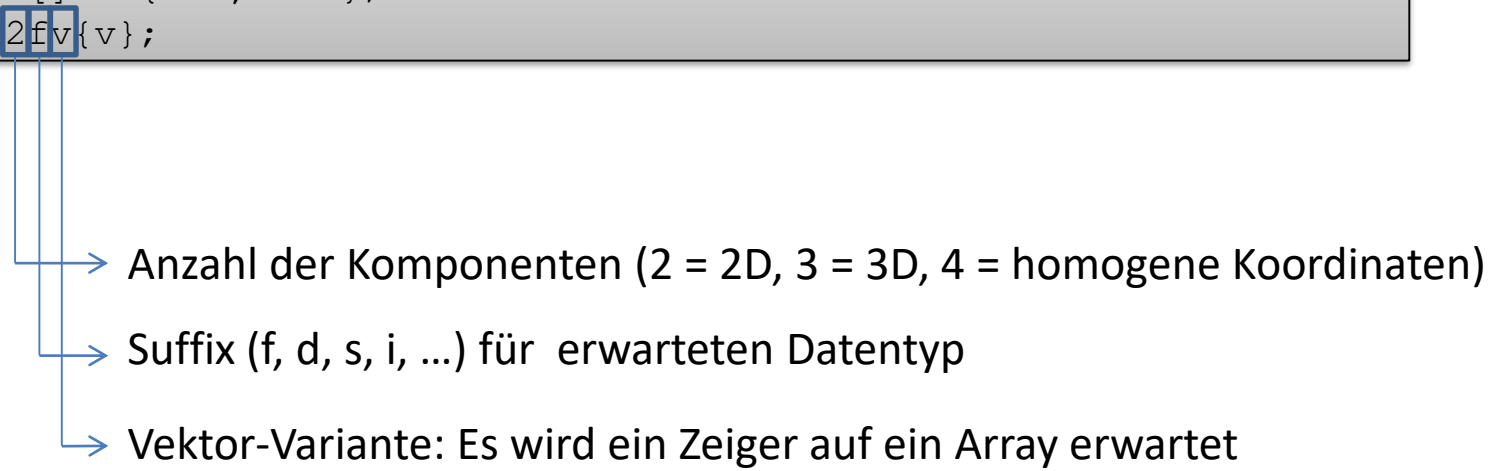
- OpenGL definiert eigene Datentypen → plattformunabhängig!
- Bibliotheksaufrufe enthalten Suffixe die auf erwarteten Datentyp hinweisen

Datentyp	Suffix	OpenGL-Typ	Bits
signed char	b	GLbyte	8
short	s	GLshort	16
int	i	GLint	32
float	f	GLfloat	32
double	d	GLdouble	64
unsigned char	ub	GLubyte	8
unsigned short	us	GLushort	16
unsigned int	ui	Gluint, GLenum	32
signed int 64	i64	GLint64	64
unsigned int 64	ui64	Gluint 64, GLbitfield	64

Beispiel: Datentypen und Namenskonventionen

- OpenGL-Aufrufe zur Definition eines Punktes mit Koordinate $x = 1, y = 2$

```
glVertex2i(1, 2);
glVertex2f(1.0, 2.0);

GLfloat v[] = {1.0, 2.0};
glVertex2fv(v);
```



OpenGL ist eine State Machine

- Periodisches Neuzeichnen der Szene
- API-Zugriff setzt eine Zustandsvariable, z.B. aktuelle Farbe
- Zustand bleibt so lange erhalten bis er explizit geändert wird
- Default-Werte für alle Variablen, z.B.
  - Ansicht- und Projektionstransformationen
  - Polygon Drawing Modes
  - Positionen und Eigenschaften des Lichtes, Materialeigenschaften
- Zustandsänderungen und -abfrage

```
glColor3f(1.0,0.0,0.0);             // current color set to red

glEnable(GL_DEPTH_TEST);            // Enable depth buffer testing
glDisable(GL_DEPTH_TEST);           // Disable depth buffer testing

GLboolean state;
state = glIsEnabled(GL_DEPTH_TEST);         // false
state = glIsDisabled(GL_DEPTH_TEST);        // true
glGetBooleanv(GL_DEPTH_TEST, &state);       // false
```

Setup

- Voraussetzungen:
  - Compiler + Entwicklungsumgebung für C/C++
  - Treiber vom Grafikkartenhersteller / Alternative: Software Version
    - Siehe https://www.opengl.org/wiki/Getting_Started#Downloading_OpenGL
    - Enthält i.d.R. GLUT Umgebung
    - Z.B. freeglut: http://www.transmissionzero.co.uk/software/freeglut-devel/
- Einrichtung:
  - Linken der OpenGL / GLU / GLUT Bibliothek
    - „OpenGL32.lib“ / „freeglut.lib“ (Windows)
    - „libGL“ (Linux): -lGL
  - Einbinden der Include-Verzeichnisse für Headerdateien

Hello World Beispiel

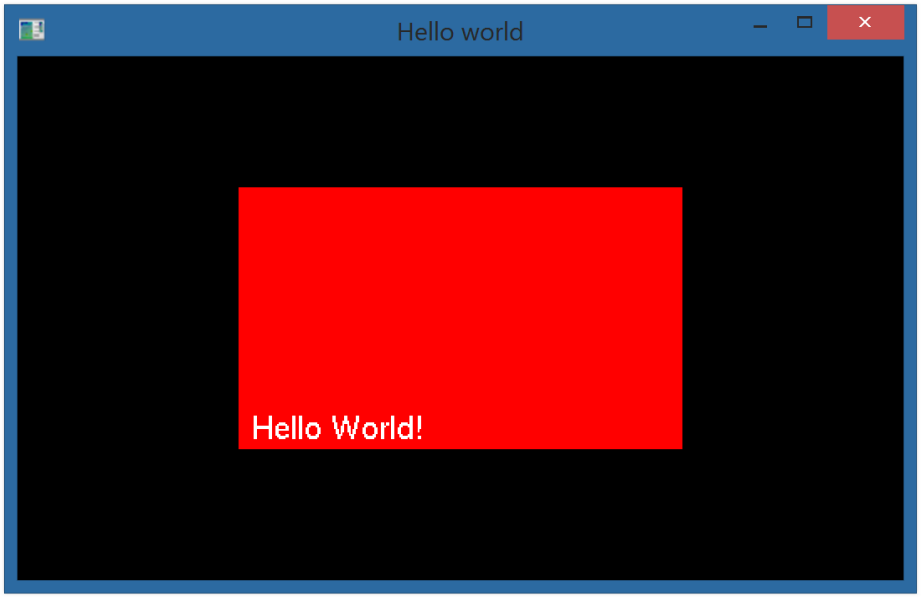
```
#include <glut.h>

int width = 640;
int height = 480;

void init(int argc, char** argv) {
   glutInit(&argc, argv);                    // init GLUT library
   glutInitDisplayMode(GLUT_SINGLE);         // Single buffered
   glutInitWindowSize(width, height);        // Window Size + Pos.
   glutInitWindowPosition(100, 100);
   glViewport(0,0,width,height);             // Viewport Size + Pos.
   glutCreateWindow("Hello world");
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(0, width, 0, height);          // orthographic projection
   glMatrixMode(GL_MODELVIEW);
}
```

Hello World Beispiel

```
void display(void) {

   char *myText = "Hello World!";
   int j;

   glColor3f(1.0,0.0,0.0);

   glBegin(GL_POLYGON);
      glVertex3f((width/2)-(width/4), (height/2)-(height/4), 0.0);
      glVertex3f((width/2)+(width/4), (height/2)-(height/4), 0.0);
      glVertex3f((width/2)+(width/4), (height/2)+(height/4), 0.0);
      glVertex3f((width/2)-(width/4), (height/2)+(height/4), 0.0);
   glEnd();

   glColor3f(1.0,1.0,1.0);
   glRasterPos2i((width/2)-(width/4)+10,(height/2)-(height/4)+10);
   for (j=0; j<strlen(myText); j++) {
      glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,myText[j]);
   }

   glFlush();
}
```

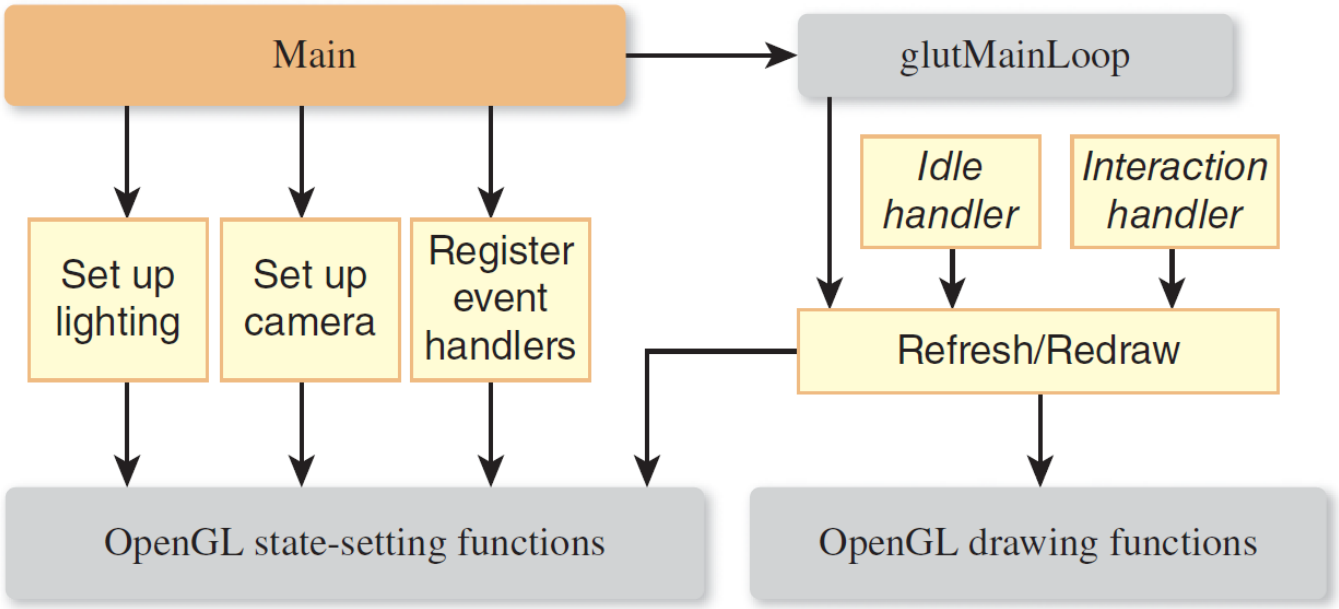
Hello World Beispiel

```
int main(int argc, char** argv) {
   init(argc, argv);
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
}
```



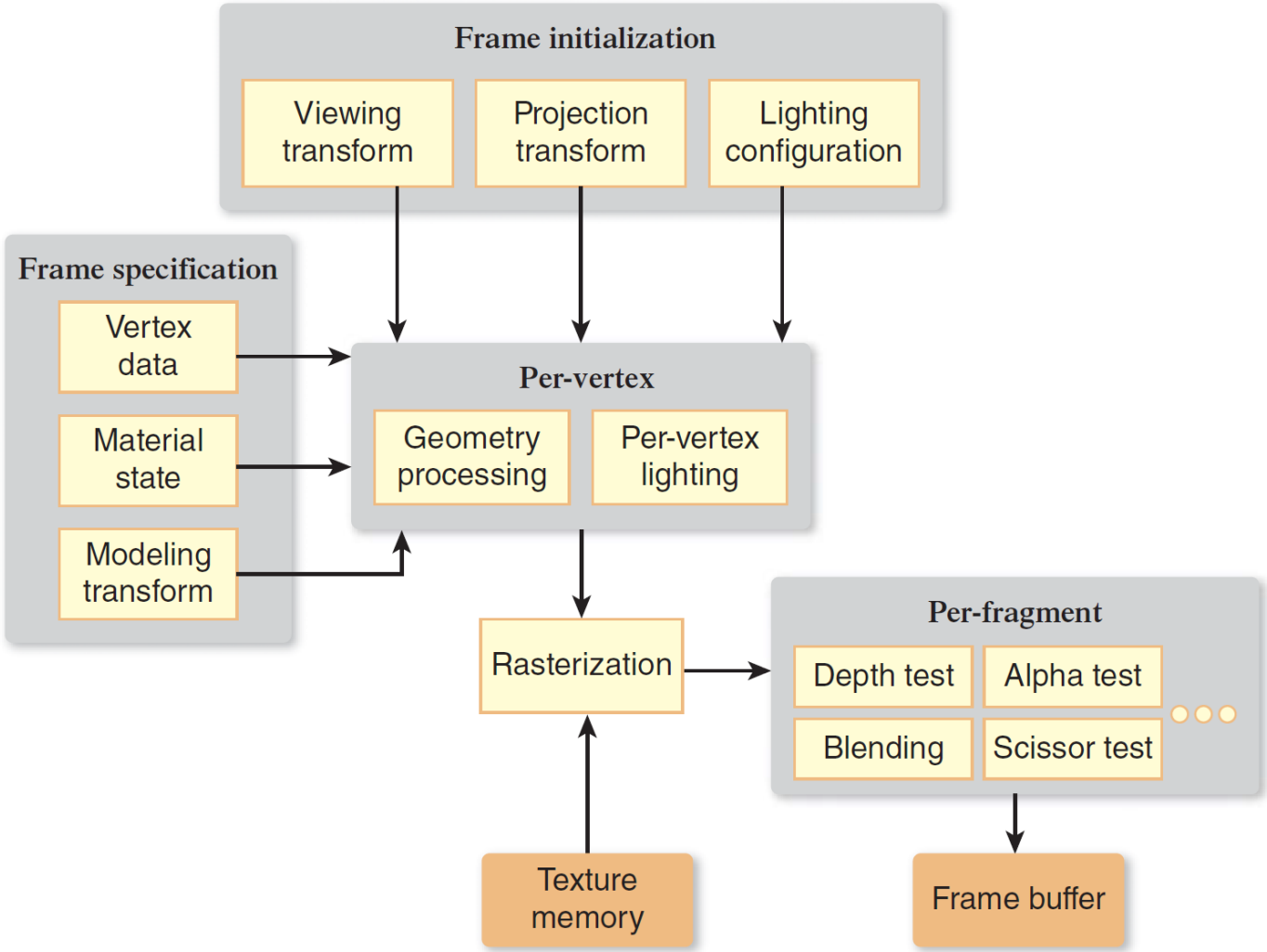
Typischer Programmablauf mit GLUT

1) Initialisierung der Grafikpipeline (glutInit)
2) Definition von Event Handlern
3) Spezifikation von Fenster, Bildausschnitt, Kamera, Licht, Laden von Daten (Meshes, Texturen)
4) Abgabe der Kontrolle an das Event-Pooling (glutMainLoop)



J.F. Hughes, A. Van Dam, M. McGuire, D. F. Sklar, J.D. Foley, S. K. Feiner, K. Akeley: „Computer Graphics Principles and Practice“ Third Edition, Addison-Wesley

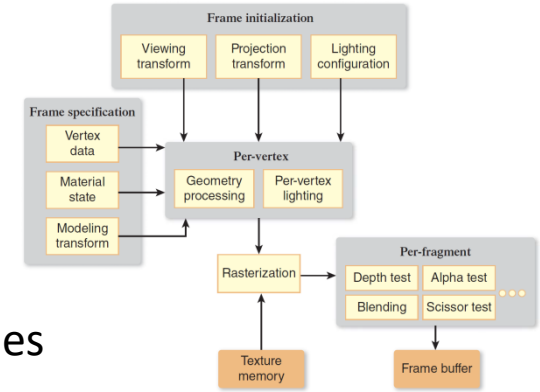
OpenGL Rendering Pipeline



J.F. Hughes, A. Van Dam, M. McGuire, D. F. Sklar, J.D. Foley, S. K. Feiner, K. Akeley: „Computer Graphics Principles and Practice“ Third Edition, Addison-Wesley

OpenGL Rendering Pipeline

1) Frame Initialization
   - Z.B. Definition der Kameraposition, Blickrichtung, Projektionsart
   - Z.B: Setzen von Statusvariablen für Licht
2) Frame Specification: Definition der Szene
   - Geometrische Daten, z.B. geometrische Körper, Meshes definiert durch Vertices (Punktkoordinaten)
   - Festlegung von Materialeigenschaften (Farben, Verhalten bei Licht)
   - Transformationen des Modells (Verschiebung, Rotation, …)
3) Per-Vertex-Operationen
   - Konvertierung der geometrischen Daten aus Modellkoordinaten zu Fensterkoordinaten auf dem Bildschirm
   - Berechnung der beleuchtungs- und materialabhängigen Farbe eines geometrischen Objektes



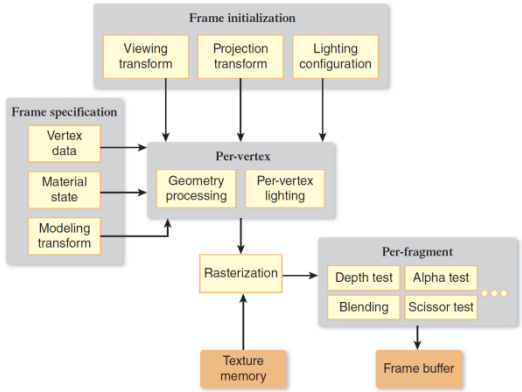
J.F. Hughes, A. Van Dam, M. McGuire, D. F. Sklar, J.D. Foley, S. K. Feiner, K. Akeley: „Computer Graphics Principles and Practice“ Third Edition, Addison-Wesley

OpenGL Rendering Pipeline

4) Rasterisierung
   - Umwandlung der „kontinuierlichen“ Geometrie in diskrete Pixel
   - Festlegung von Pixel-Farben basierend auf Licht, Texturen, etc.

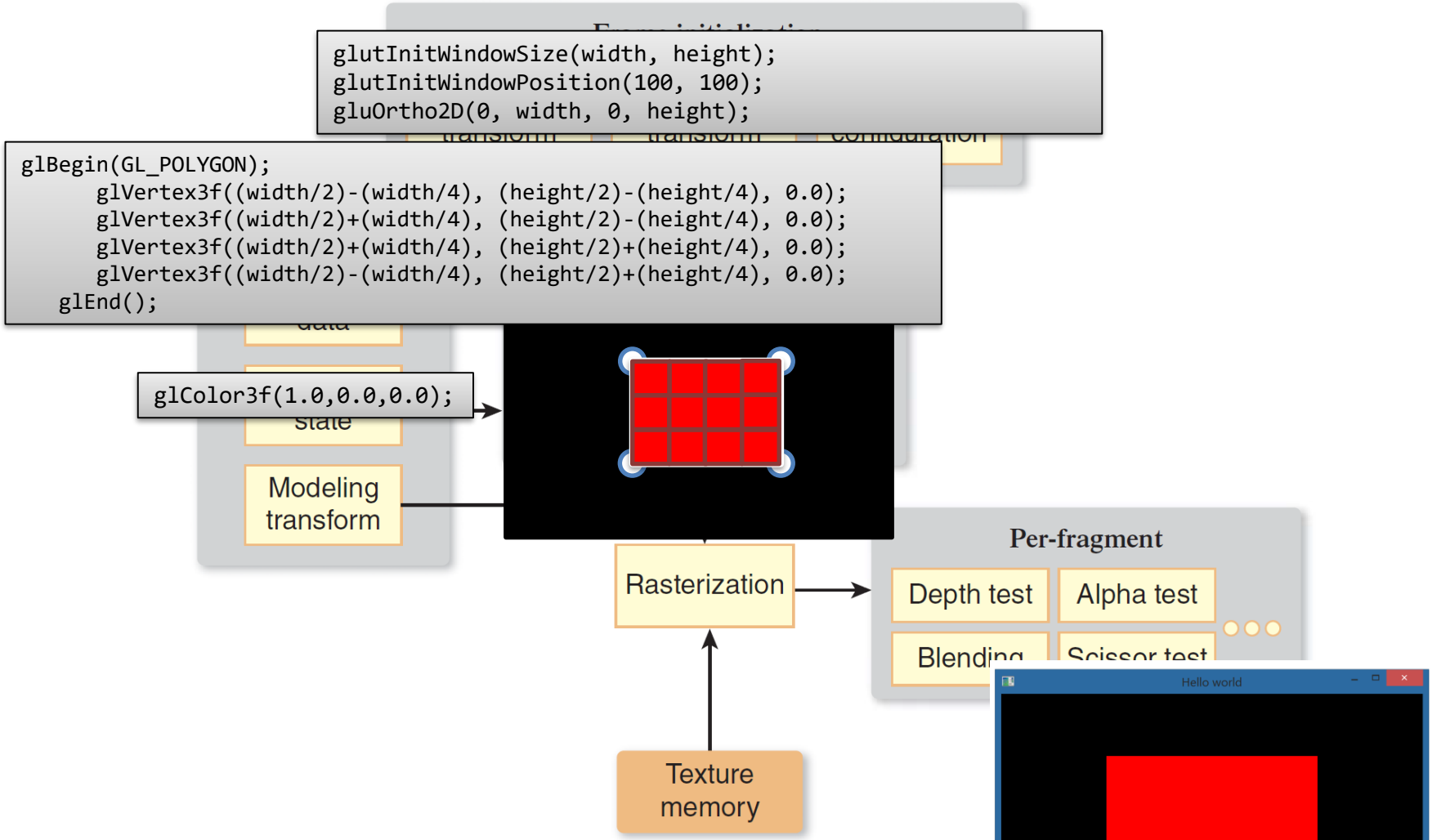
5) Per-Fragment Operationen
   - Festlegung welche/wie die Pixel auf dem Bildschirm dargestellt werden, z.B. Verdeckung von Objekten, Transparenz

6) Frame-Buffer
   - Zwischenpuffer für die Anzeige der gerenderten Szene



J.F. Hughes, A. Van Dam, M. McGuire, D. F. Sklar, J.D. Foley, S. K. Feiner, K. Akeley: „Computer Graphics Principles and Practice“ Third Edition, Addison-Wesley

OpenGL Rendering Pipeline für unser Beispiel



J.F. Hughes, A. Van Dam, M. McGuire, D. F. Sklar, J.D. Foley, S. K. Feiner, K. Akeley ... Idison-Wesley

Für Interessierte…

- http://glslsandbox.com/ - Beispiele für Implementierungen mit der OpenGL Shading Language, interaktiv veränderbar!
- Google: „openGL examples“
- http://wiki.delphigl.com/index.php/Hauptseite - deutsches Wiki zu OpenGL-Themen mit Funktionsübersicht und Erläuterungen
- D. Shreiner, G. Sellers, J. Kessenich, B. Licea-Kane: „OpenGL Programming Guide“, Eighth Edition, The official guide to learning OpenGL, Version 4.3: http://www.ics.uci.edu/~gopi/CS211B/opengl_programming_guide_8th_edition.pdf

ZUSAMMENFASSUNG

Zusammenfassung

- Programmierbibliotheken abstrahieren Zugriff auf Hardware
- Immediate mode ⇔ retained mode
- Fixed functions → Shaders → Programmierbare Pipeline

- OpenGL
  - Client-Server-Modell
  - Wichtige Bibliotheken: GL, GLU, GLUT mit entsprechenden Namenskonventionen für die Funktionsaufrufe
  - OpenGL definiert eigene Datentypen
  - OpenGL ist eine State Machine
  - OpenGL Rendering Pipeline: notwendige Verarbeitungsschritte bis man eine Szene auf dem Bildschirm dargestellt bekommt.

ÜBUNGSAUFGABEN

Programmierübung

1. Richten Sie sich eine Entwicklungsumgebung für die OpenGL Programmierung ein
2. Implementieren sie das Hello-World-Beispiel aus den Folien.
3. Ändern Sie die Farbe des Fensterhintergrundes. Verwenden Sie hierzu den Befehl **`glClearColor`** und **`glClear(GL_COLOR_BUFFER_BIT)`**
4. Ändern Sie Größe und Position des Fensters auf dem Bildschirm.
5. Ändern Sie die Farbe und Position des Rechtecks. Verwenden Sie bei der Farbeinstellung die Vektor-Variante
6. Zeichnen Sie statt des Rechtecks ein Dreieck in der Mitte des Fensters.
7. Fügen Sie der Szene ein weiteres Polygon in einer anderen Farbe hinzu.

Lösung

3. Änderung der Farbe des Fenster-Hintergrundes: Einfügen zu Beginn der Display-Funktion

```
glClearColor(0.0,0.0,1.0,1.0);
glClear(GL_COLOR_BUFFER_BIT);
```

4. Ändern der Größe und Position des Fensters: Ändern der Einträge in der init-Funktion, z.B.

```
glutInitWindowSize(width*2, height);
glutInitWindowPosition(200, 100);
```

5. Ändern von Farbe und Position des Rechtecks: Ändern in der Display Funktion: z.B. gelbes Rechteck nach rechts oben um je 50 Pixel verschoben

```
GLfloat myColor[3] = {1.0,1.0,0.0};
glColor3fv(myColor);

glBegin(GL_POLYGON);
glVertex3f((width/2)-(width/4)+50, (height/2)-(height/4)+50, 0.0);
glVertex3f((width/2)+(width/4)+50, (height/2)-(height/4)+50, 0.0);
glVertex3f((width/2)+(width/4)+50, (height/2)+(height/4)+50, 0.0);
glVertex3f((width/2)-(width/4)+50, (height/2)+(height/4)+50, 0.0);
glEnd();
```

Lösung (2)

6. Dreieck statt Rechteck zeichnen. Änderung in der Display-Funktion, z.B.

```
glBegin(GL_POLYGON);
glVertex3f((width/2)-(width/4), (height/2)-(height/4), 0.0);
glVertex3f((width/2)+(width/4), (height/2)-(height/4), 0.0);
glVertex3f((width/2), (height/2)+(height/4), 0.0);
glEnd();
```

7. Hinzufügen eines weiteren Polygons in einer anderen Farbe: Änderung in der Display-Funktion, z.B.

```
GLfloat myColor[3] = {1.0,1.0,0.0};
GLfloat myColor2[3] = {1.0,0.0,0.0};

glColor3fv(myColor);

glBegin(GL_POLYGON); // Rechteck
glVertex3f((width/2)-(width/4)-50, (height/2)-(height/4)-50, 0.0);
glVertex3f((width/2)+(width/4)-50, (height/2)-(height/4)-50, 0.0);
glVertex3f((width/2)+(width/4)-50, (height/2)+(height/4)-50, 0.0);
glVertex3f((width/2)-(width/4)-50, (height/2)+(height/4)-50, 0.0);
glEnd();

glColor3fv(myColor2);

glBegin(GL_POLYGON); // Dreieck
glVertex3f((width/2)-(width/4)+100, (height/2)-(height/4)+100, 0.0);
glVertex3f((width/2)+(width/4)+100, (height/2)-(height/4)+100, 0.0);
glVertex3f((width/2)+100, (height/2)+(height/4)+100, 0.0);
glEnd();
```

Übungsfragen Kapitel 2

- Welche Aufgabe übernehmen Programmierbibliotheken wie OpenGL bei der Grafikprogrammierung?
- Worin liegen die Unterschiede zwischen Immediate Mode und Retained Mode Programmierbibliotheken?
- Was sind Shader und welchen Vorteil bringen sie gegenüber Fixed Functon Bibliotheken?