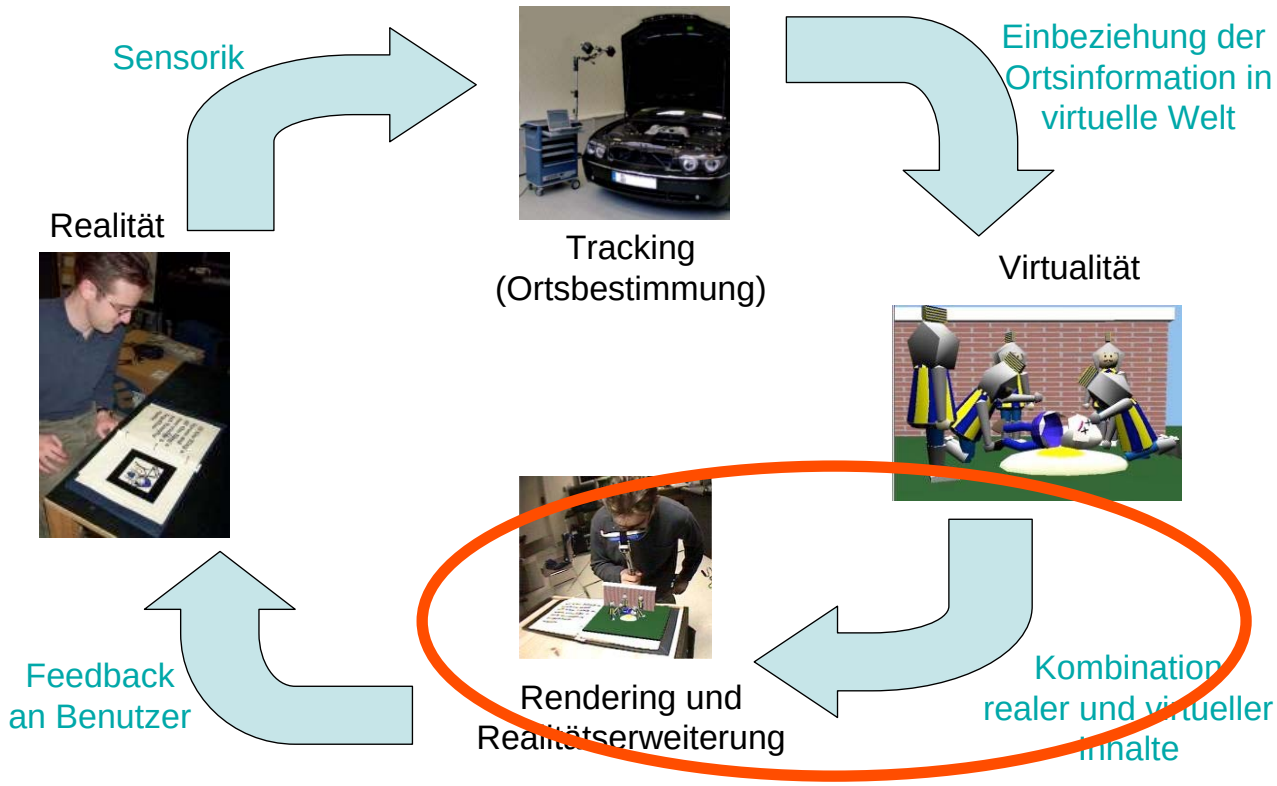
Rendering für Augmented Reality

Vorlesung „Augmented Reality”
Prof. Dr. Andreas Butz
WS 2006/07
Folien heute von Dr. Martin Wagner

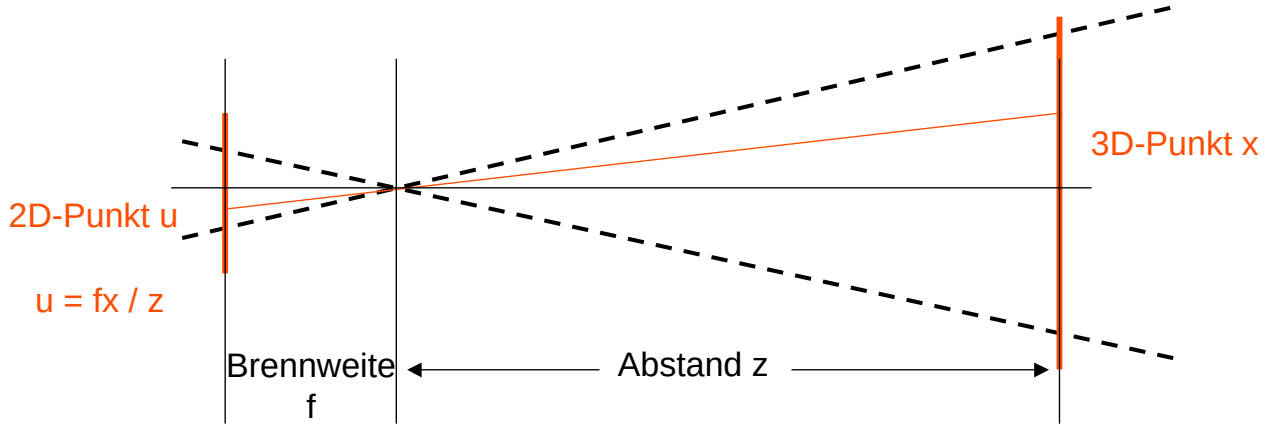
Ein Generisches AR-System



Überblick

- Grundlage: Perspektivisches Zeichnen
- 3D Rendering Pipeline
- Verdeckungen von realen und virtuellen Objekten
- Stereo Rendering
- Rendering für Projektoren
- Softwarehilfen: Szenengraphen

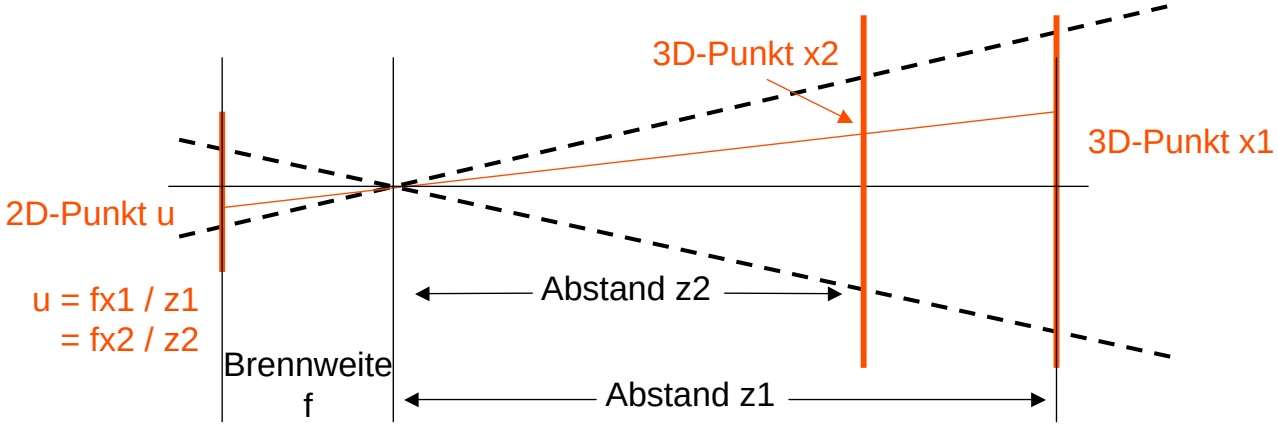
Grundlage: Perspektivische Projektion



Grundidee:

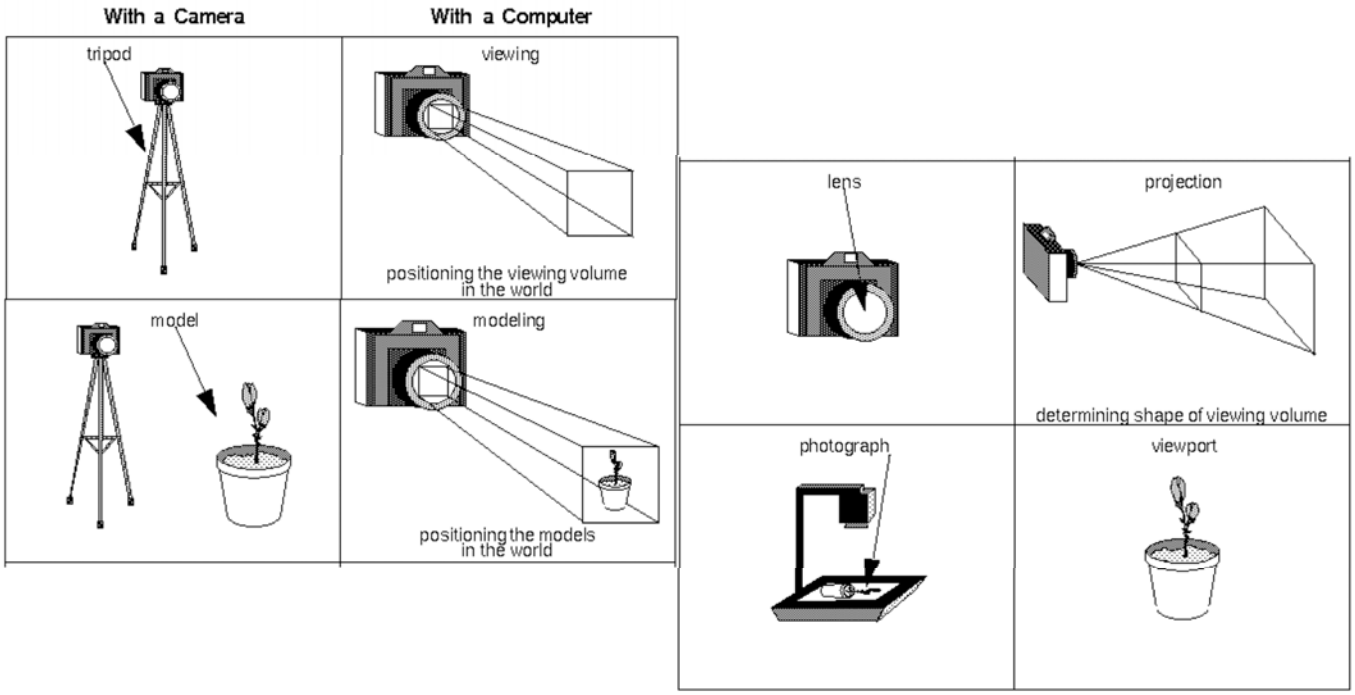
- Lage aller Objekte in 3D bestimmen und zeichnen
- Objekte auf 2D-Bildebene projizieren
- Passenden Bildausschnitt wählen und anzeigen

Objektverdeckung



- Punkt x2 verdeckt x1
- Idee: z-Buffering
  - Zeichne Punkte nur, falls nicht an der selben Stelle der Bildebene schon etwas ist
  - Wird in Grafikhardware höchst effizient implementiert

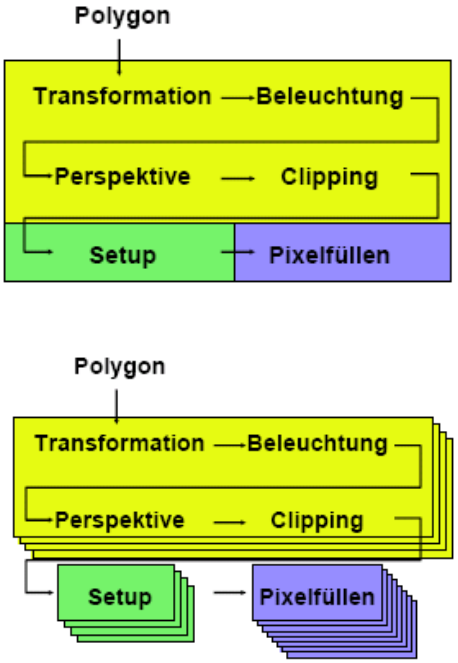
Kameraanalogie



[OpenGL Programming Guide (Red Book), Chapter 3]

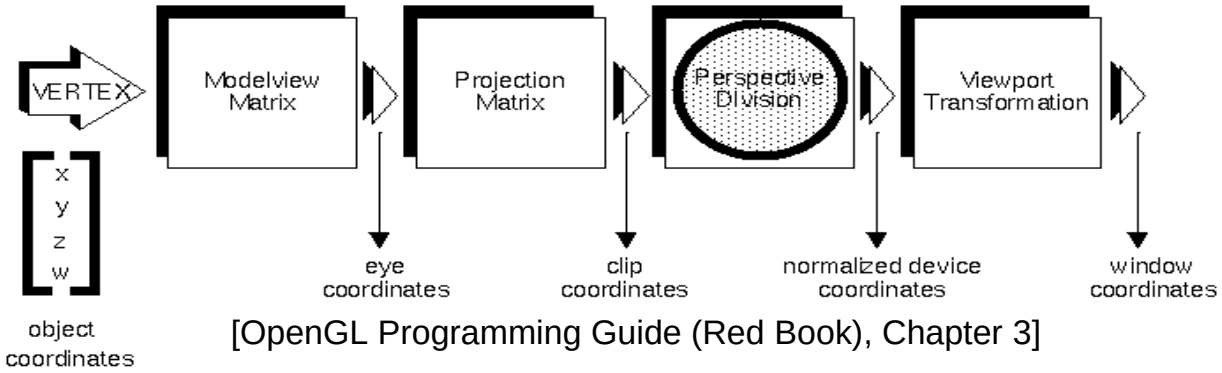
3D Rendering Pipeline

- Ziel: effiziente Berechnung aller Transformationen
- Vor allem lineare Berechnungen können hocheffizient in 3D-Hardware gegossen werden
- Pipeline-Konzept: Sequentielle, teils parallelisierte Verarbeitung



[Müller, Vorlesung VR/AR, 2003]

OpenGL-Pipeline



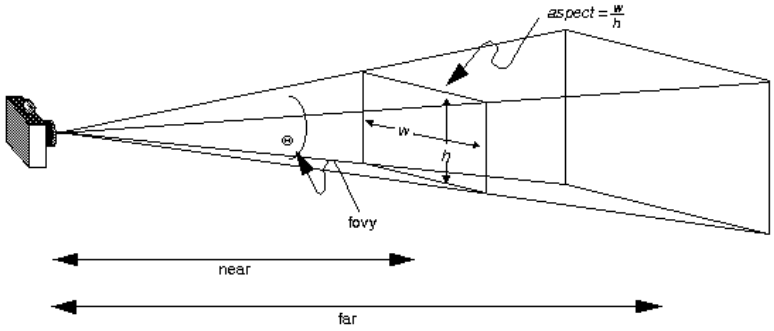
[OpenGL Programming Guide (Red Book), Chapter 3]

- Nur lineare Operationen bis zu clip coordinates
- „Perspective Division“ teilt durch w-Wert homogener Koordinaten
  - Normalized Device Coordinates sind in Würfel mit Seitenlänge 2: [-1; +1]

$$\begin{bmatrix} clip_x/clip_w \\ clip_y/clip_w \\ clip_z/clip_w \\ clip_w/clip_w \end{bmatrix} = \begin{bmatrix} NDC_x \\ NDC_y \\ NDC_z \\ 1 \end{bmatrix}$$

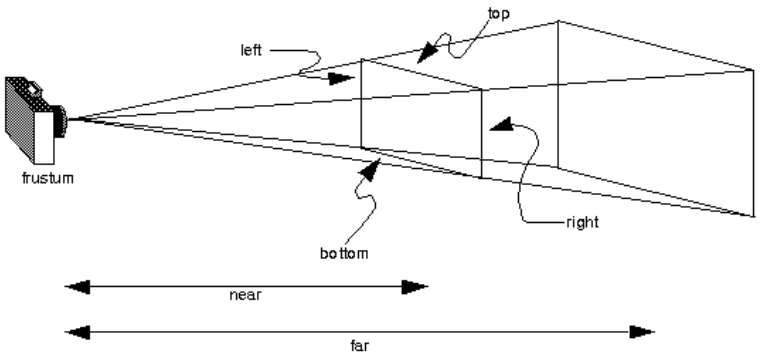
Perspektivische Projektion in OpenGL (1)

- Einfache Methode: ***gluPerspective***
- Angabe von:
  - Öffnungswinkel Kamera in y-Richtung
  - Aspect ratio w/h
  - Near/far clipping plane



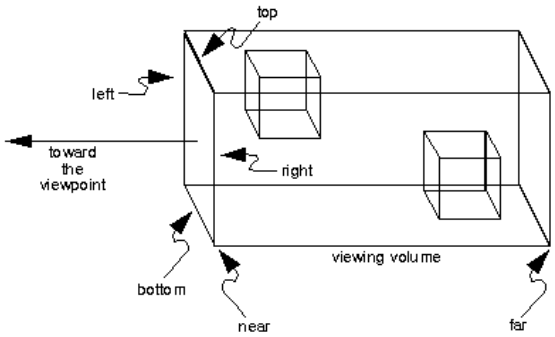
Perspektivische Projektion in OpenGL (2)

- Mächtige Methode: ***glFrustum***
- Angabe von:
  - Near/far clipping plane
  - Allen Rändern der near clipping plane
- Hierdurch auch off-axis Projektion möglich



Parallele Projektion in OpenGL (3)

- Orthographische Projektion: ***glOrtho***
- Angabe der Ränder des Viewing Volume Quaders
- Ideal zum Zeichnen von 2D-Elementen (Video Background, HUD)

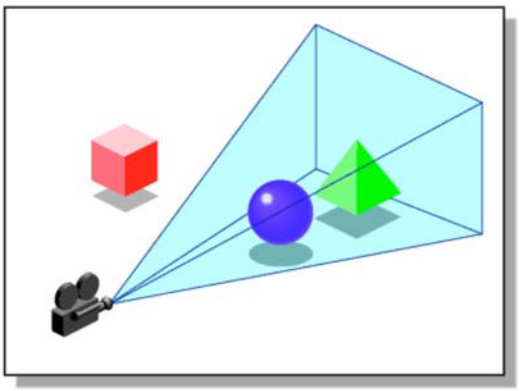


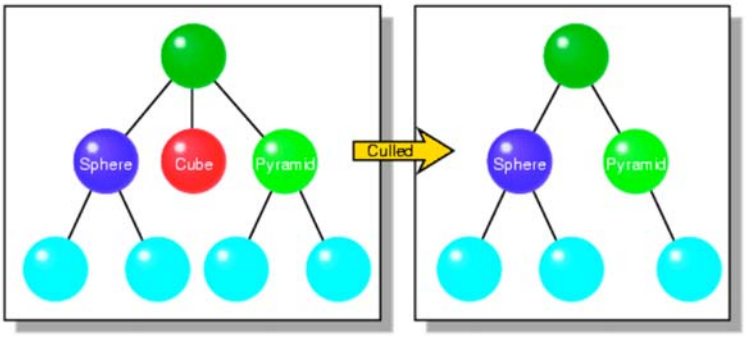
Echtzeit-Anforderungen

- Grundproblem: beschränkte Wiederholrate aller Anzeigegeräte
  - Projektor mit 60 Hz führt ***immer*** zu 16 2/3 ms Latenz
  - Sinnvoll: Synchronisation des Rendering mit Bildwiederholrate (Genlock, Framelock)
- Immersionsgefühl erst ab 10Hz
- Flüssiges Arbeiten (insbesondere mit HMDs) erst ab 60Hz

Effizienzsteigerung: View Culling

- Idee:
  Nur sichtbare Objekte zeichnen
- Hohe Performanz durch konservative Entscheidungen





[SGI, OpenGL Performer Getting Started Guide, chapter 15]

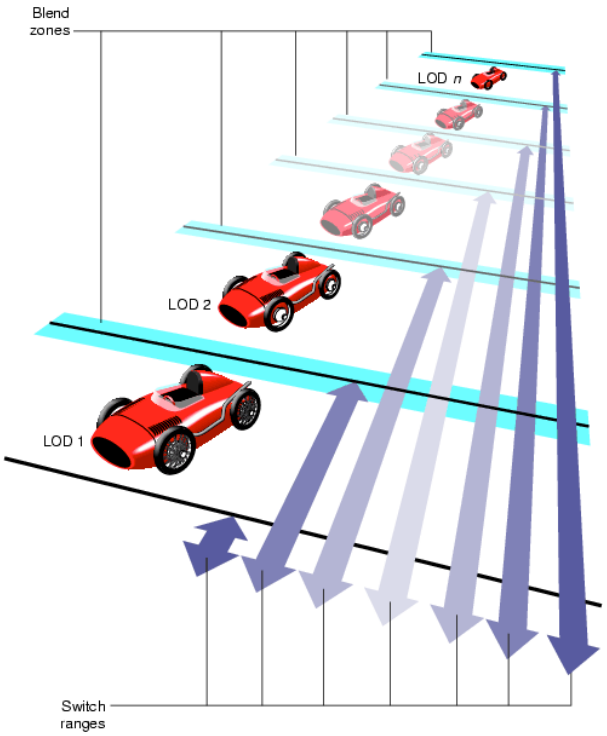
Effizienzsteigerung: Hierarchisches View Culling

- Beobachtung:
  Objekte sind oft hierarchisch organisiert
- Idee: durch Test eines Vaterobjekts können viele Kinder wegoptimiert werden

[Müller, Vorlesung VR/AR, 2003]

Effizienzsteigerung: Level of Detail

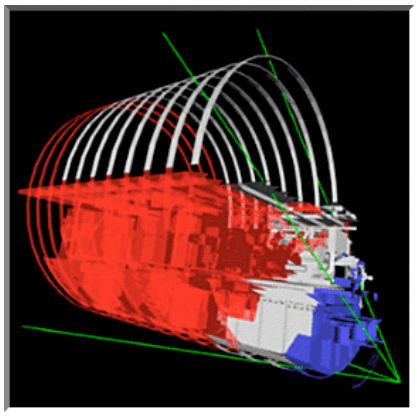
- Beobachtung: entfernte Objekte sieht man ungenauer
- Idee: zeichne Objekte nur dann detailliert, wenn sie nah am Betrachter sind



[SGI, OpenGL Performer Getting Started Guide, chapter 3]

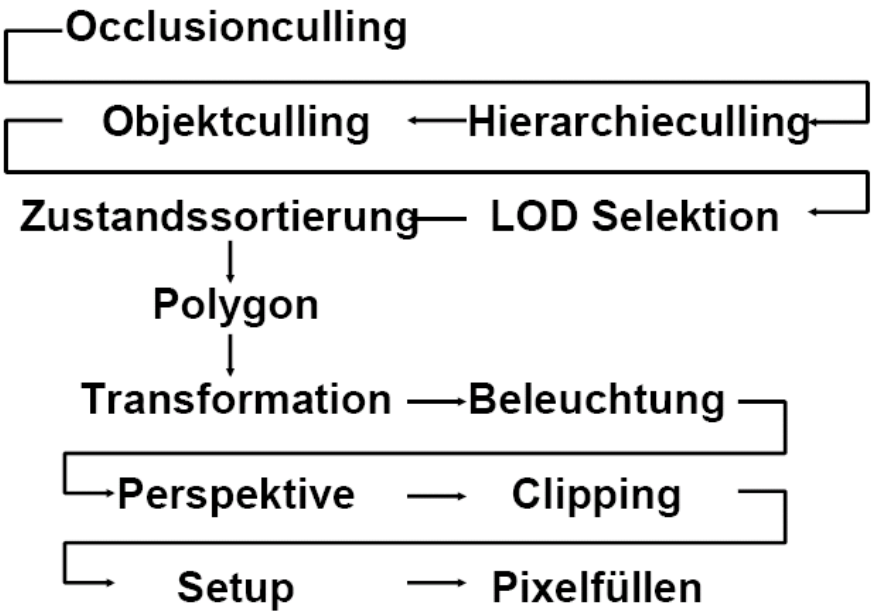
Effizienzsteigerung: Occlusion Culling

- Idee: Zeichne keine Objekte, die ohnehin verdeckt werden
- Allgemein schwieriges Problem
  - Konservative Lösung genügt
  - Möglicher Ansatz: hierarchische Tests



[Zhang, Effective Occlusion Culling for the Interactive Display of Arbitrary Models, Ph.D. Thesis, UNC 1998]

3D Rendering: erweiterte Pipeline



Verdeckungen

- Grundproblem:
  Wie schafft man korrekte Verdeckungen von realen und virtuellen Objekten?
- (Recht) einfacher Fall:
  virtuelle Objektverdeckung
  - Occlusion Culling
  - Z-Buffering

  → Kann auf Grafikkarte schnell realisiert werden

Verdeckungen: Transparente Objekte (1)

- Z-Buffer:
  zeichne Objekt nur, wenn nicht an der selben (x,y) Stelle schon ein niedrigerer z-Wert steht
- Problem:
  Was ist, wenn Objekte durchscheinen?

[Quake 2, ID soft]

Verdeckungen: Transparente Objekte (2)

- Alpha-Wert: gibt Transparenz eines Objekts an (0 ist transparent, 1 ist opak)
- Sortierung von Objekten:
  - Zuerst alle opaken Objekte (korrektes z-Buffering)
  - Dann alle transparenten Objekte mit eingeschaltetem Blending
    Obacht: müssen von hinten nach vorne sortiert sein.
- Und was ist mit realen Objekten?

Verdeckung virt. durch reale Objekte

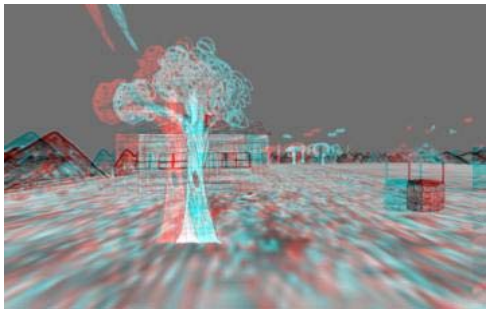
- Idee: Zeichne volltransparentes Objekt
  - Z-Wert wird korrekt gesetzt
  - Hintergrund (Videobild bei Video-See-Through, schwarz bei Optical-See-Through) bleibt unverändert
  - Muss als erstes passieren
- Also neue Reihenfolge:
  - Modelle realer Objekte volltransparent
  - Opake virtuelle Objekte
  - Transparente virtuelle Objekte, *von hinten nach vorne*

Stereo Rendering

- Bisher: Monoanzeige
  - Meist einfach & gut
  - Ideal für Anzeige auf 2D-Bildschirmen
- Aber: AR ist per Definition dreidimensional
  - Verbesserter Tiefeneindruck
  - Verbesserte Interaktion mit virtuellen Objekten
- Nur wie auf einem Rechner mit einer Grafikkarte zwei Bilder für zwei Augen erzeugen?

Stereo Rendering: Techniken (1)

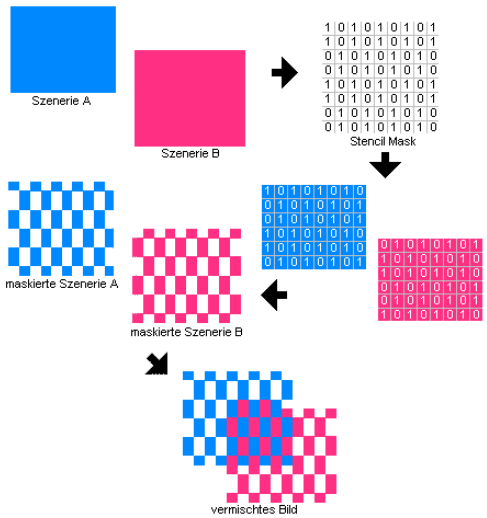
- Anaglyphic Stereo
  - Zwei Farben für links/rechts und Brille mit Filtern
  - Günstig, aber Farbinfo geht verloren
- Quad Buffering
  - Abwechselnd ein Bild links/rechts ausgeben
  - Stereo-HMD sortiert Bilder passend
  - Nur in High-End Grafikkarten (ATI FireGL, NVidia Quadro)



[Pape, Anstey, SIGGRAPH 2004]

Stereo Rendering: Techniken (2)

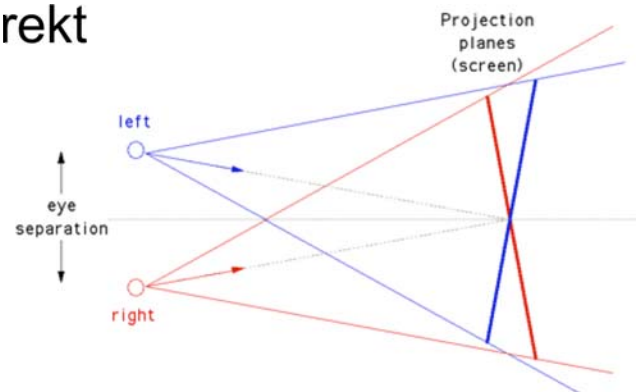
- Stencil Buffering
  - Alle ungeraden Zeilen geben linkes Bild
  - Alle geraden Zeilen geben rechtes Bild
  - Problem: Halbierung der Auflösung
- Zwei Grafikkarten
  - Bei HMD einfach
  - Komplex bei Projektoren (→ Polarisierungsfilter)



[3DChip - Das Grafikkarten Onlinemagazin, 2005]

Stereo Rendering: Projektion (1)

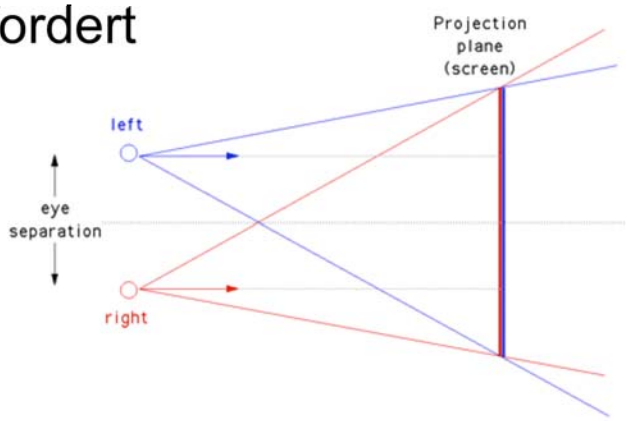
- Toe-in Methode:
  - Kamera für linkes und rechtes Auge zeigen auf gemeinsamen Blickpunkt
  - Einfache Implementierung mit ***gluPerspective***
  - Geometrisch nicht korrekt
  - Vertikale Parallaxe



[http://astronomy.swin.edu.au/~pbourke/opengl/stereogl/]

Stereo Rendering: Projektion (2)

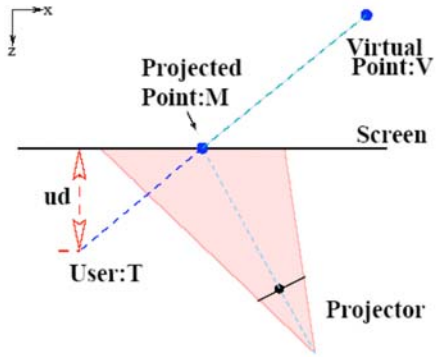
- Parallel axis asymmetric frustum perspective projection
  - Identische Projektionsebene für beide Augen
  - Geometrisch korrekt
  - Implementierung erfordert ***glFrustum***



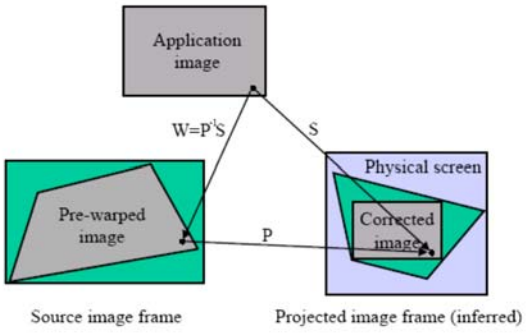
[http://astronomy.swin.edu.au/~pbourke/opengl/stereogl/]

Rendering für Projektoren (1)

- Projektion auf planare Oberflächen
  - Homographie gibt lineare Abbildung einer 3D-Ebene auf andere 3D-Ebene
  - Hier: Abbildung Projektionsebene (Wand) auf near clipping plane (Projektor)
  - Homographie kann in Projektionsmatrix integriert werden
    → Echtzeit kein Problem
  - Achtung auf Depth-Buffer Werte



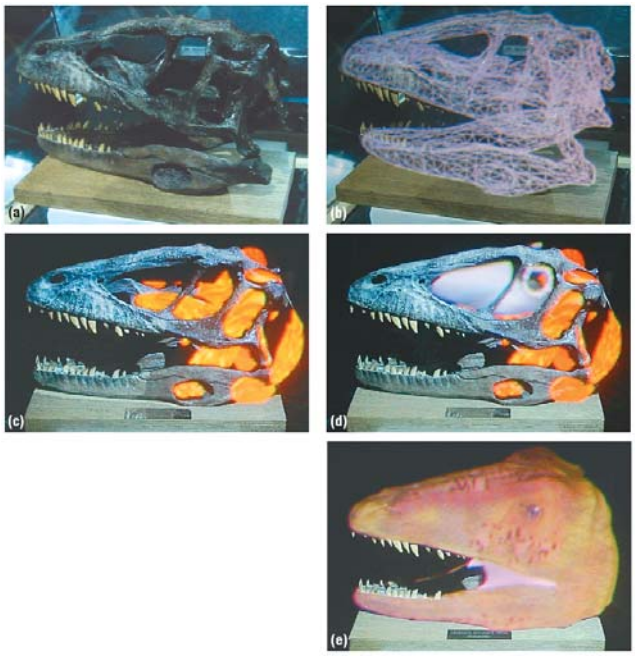
[Raskar 1999]



[Sukthankar 2001]

Rendering für Projektoren (2)

- Multi-Pass Rendering für nichtplanare Oberflächen
  - Erster Durchlauf: zeichne Bild aus Sicht des Benutzers (in Framebuffer)
  - Zweiter Durchlauf: nutze Framebuffer als Textur, die auf Modell der Realität gelegt wird, zeichne Bild aus Sicht des Projektors

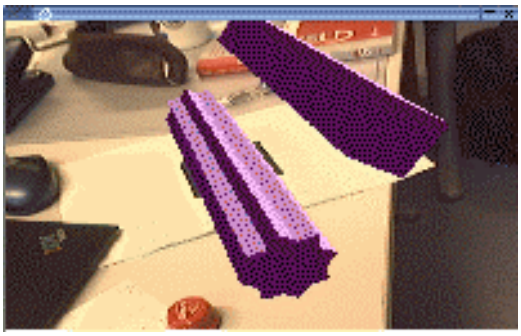


[Bimber et al. 2002]

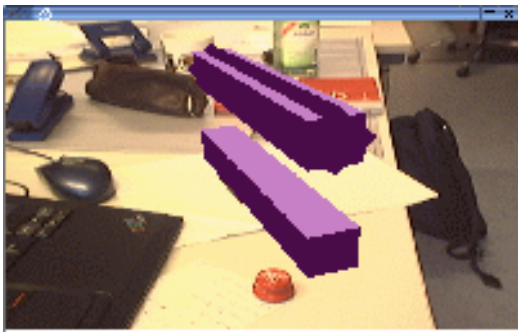
Szenengraphen

- Wiederholung:
  Hierarchische Transformationen

Zuerst Rotation, dann Translation:



Zuerst Translation, dann Rotation:



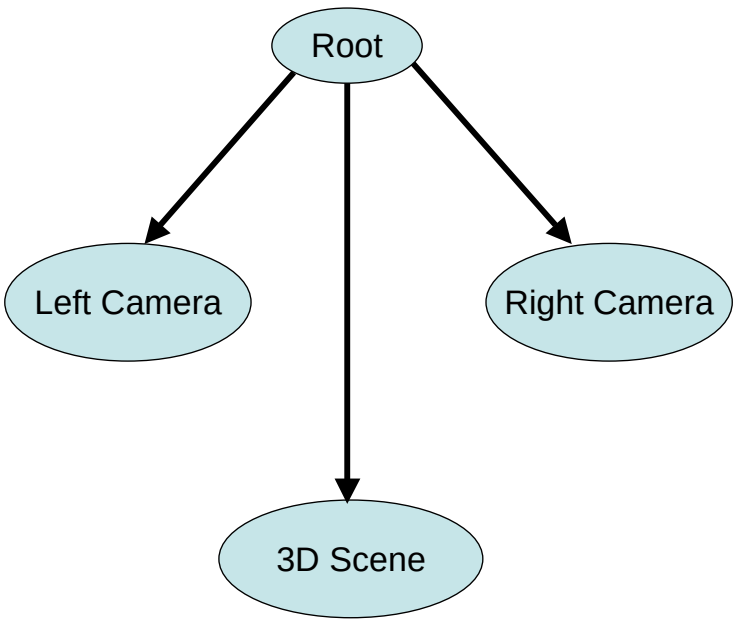
Szenengraphen: Zweck (1)

- Angenehme Programmierung hierarchischer Szenen
- Vorbearbeitung:
  - Occlusion Culling
  - View Culling
  - Level of Detail
- Sortierung von Primitiven
  - Effizienzsteigerung
  - Korrektes Rendern transparenter Objekte

Szenengraphen: Zweck (2)

- Wiederverwendung von Szenen
  - Szenengraph ist ein DAG
- Angenehmes Multi-Pass Rendering
- Import/Export von 3D-Szenen, Formate
  - 3DStudio Max
  - VRML
  - Open Inventor

Szenengraphen: Stereo Rendering

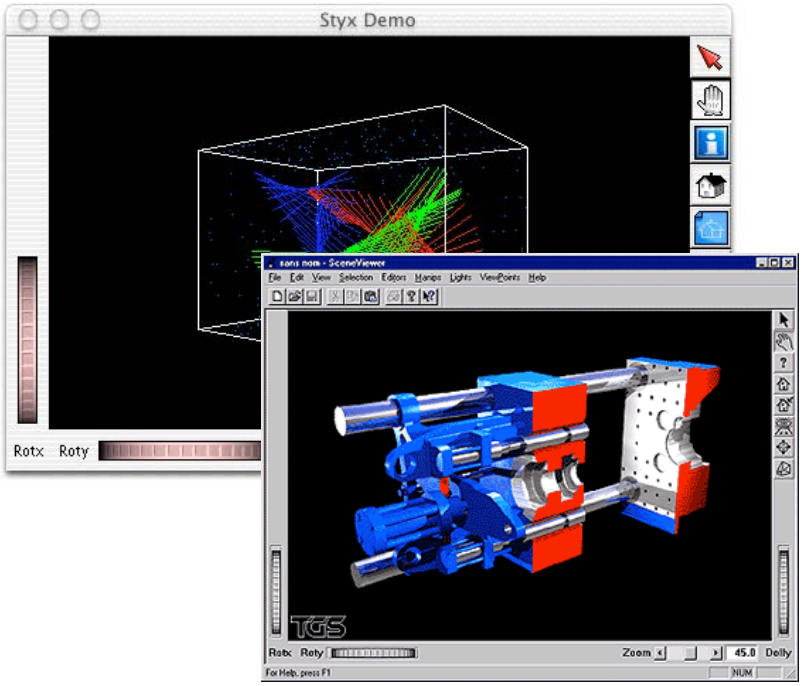


Szenengraphen: Open Inventor

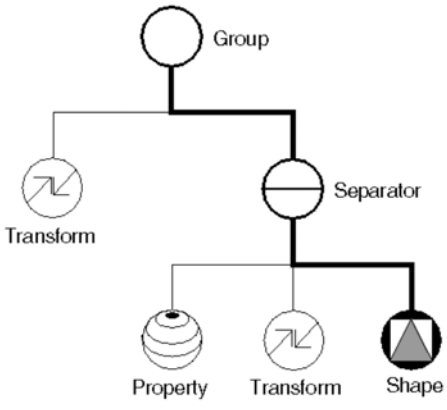
- C++ Bibliothek von SGI
- Varianten:
  - SGI Inventor: Original, mittlerweile Open Source
  - TGS Inventor: kommerzielle Weiterentwicklung
  - Coin3D: GPL-Variante aus Norwegen
- Haupteinsatz für interaktive Darstellung virtueller Welten
- Erweiterungen durch Ableiten gegebener Klassen
- Bsp.: Studierstube (TU Graz)

Szenengraphen: Open Inventor (2)

[http://www.coin3d.org]



[http://www.tgs.com]

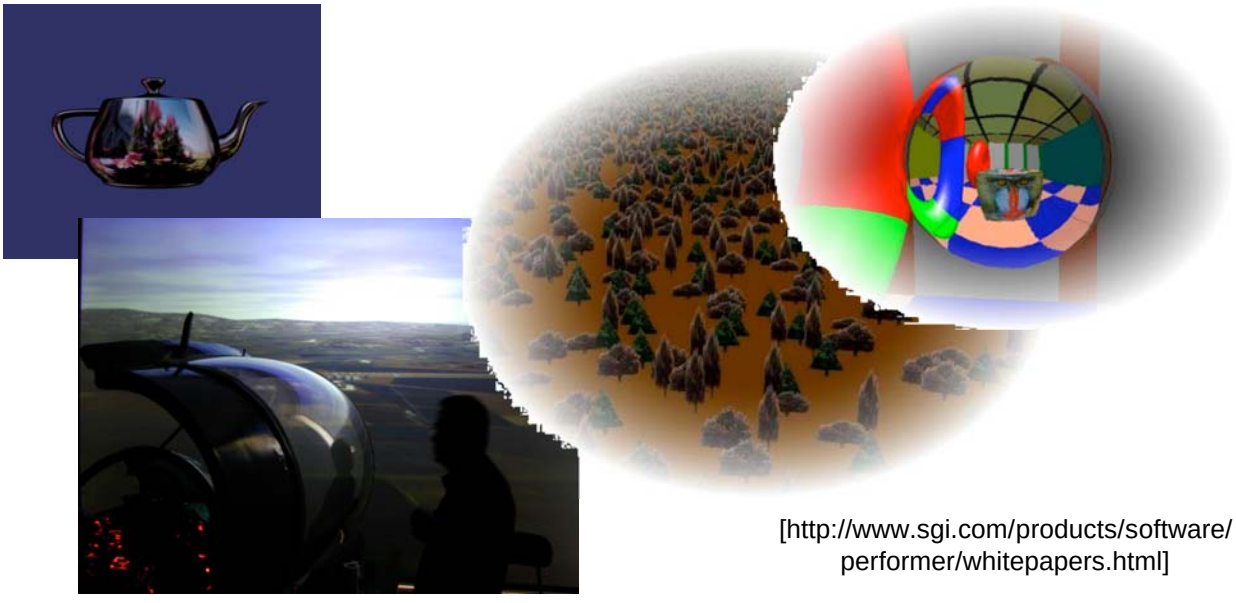


[The Inventor Mentor]

Szenengraphen: OpenGL Performer

- C++ Bibliothek von SGI
- Auf höchste Performance optimiert
- Unterstützt Multiprocessing (mehrere GPUs mit gleicher Szene)
- Freie Alternative: openscenegraph.org
  - Wird auch in Fluidum verwendet
  - Import/Export zahlreicher 3D-Formate

Szenengraphen: OpenGL Performer (2)



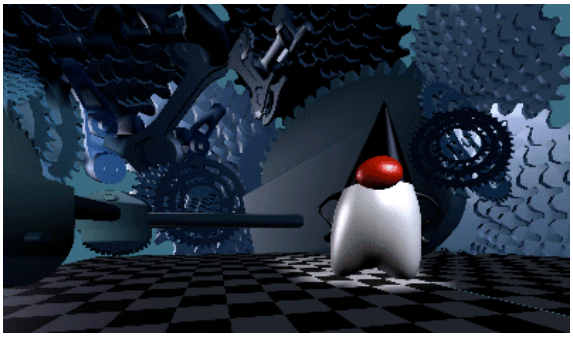
[http://www.sgi.com/products/software/performer/whitepapers.html]

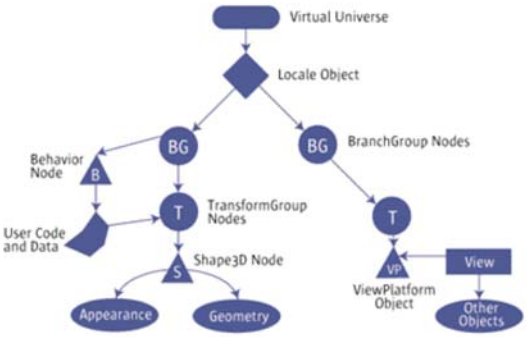
[http://www.openscenegraph.org/]

Szenengraphen: Java 3D

- Szenengraphbibliothek für Java, von Sun
- Setzt auf OpenGL oder DirectX auf, Ziel: plattformunabhängige 3D-Programme
- Schwerpunkte:
  - Flexible virtuelle Kameras (HMD, Stereo etc.)
  - Integration von 3D-Sound

Szenengraphen: Java 3D (2)





[http://java.sun.com/products/java-media/3D/collateral/datasheet/java3d-page2.html]