Texturbasierte Volumenvisualisierung

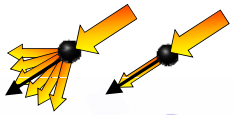
Andreas Kolb und Martin Lambers

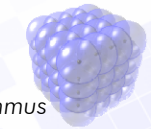
computergraphik und multimedia systeme
universität siegen

Einführung

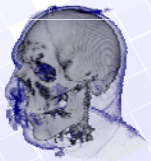
Direkte Volumenvisualisierung

- *Letzte Stunde:* Standardverfahren zur Direkten Volumenvisualisierung
  - Physikalisch basierte Bildsynthese.
  - Bildraum- und Objektraumverfahren
  - *Beispielalgorithmen:*
    Ray-Casting, Splatting, Shear-Warp-Algorithmus





- **Fazit:** Direkte Volumenvisualisierung bedeutet hohen Rechenaufwand (Interpolation und numerische Integration)



Einführung

Direkte Volumenvisualisierung

- *Heute:* Hardwarebeschleunigte Rasterisierungs-verfahren zur Volumenvisualisierung
  - Interaktive bzw. echtzeitfähige Algorithmen
  - Texturbasierte Verfahren
  - Beispielimplementierungen mit 2D Texturen, 3D Texturen und 2D Multitexturen
  - Keine Spezialhardware (mehr) notwendig

Heutzutage kann Raycasting auch auf GPUs unter Einsatz von CUDA echtzeitfähig realisiert werden.

Überblick

Rasterbasiertes Volume Rendering

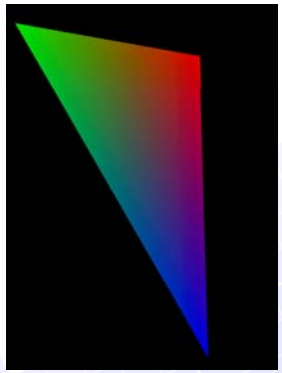
- Erinnerung: Graphik-Hardware
- Texturen: Wie werden Texturen zum Volume Rendering verwendet?
  - 2D Texturbasierte Verfahren
  - 3D Texturbasierte Verfahren
  - 2D Multitexturbasierte Verfahren

Was kann die Hardware?

5

Gegeben:
Eckpunkte der Dreiecke

- *Dreiecke zeichnen*
  - Pixel bestimmen
  - Farbe interpolieren



Was kann die Hardware?

6

Gegeben:
Eckpunkte der Dreiecke

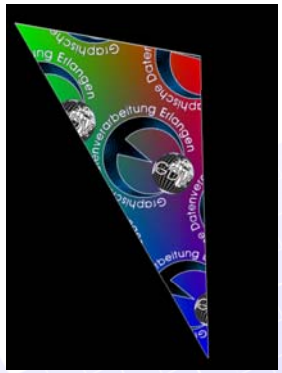
- *Dreiecke zeichnen*
  - Pixel bestimmen
  - Farbe interpolieren
  - Texturen anwenden

Was kann die Hardware?

7

Gegeben:
Eckpunkte der Dreiecke

- *Dreiecke zeichnen*
  - Pixel bestimmen
  - Farbe interpolieren
  - Texturen anwenden
- *Compositing*
  - Primär-Farbe mit Textur

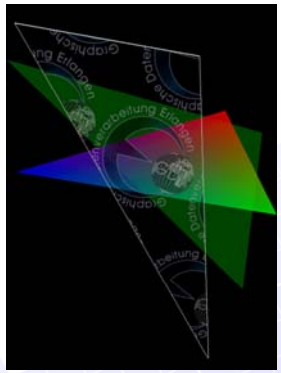


Was kann die Hardware?

8

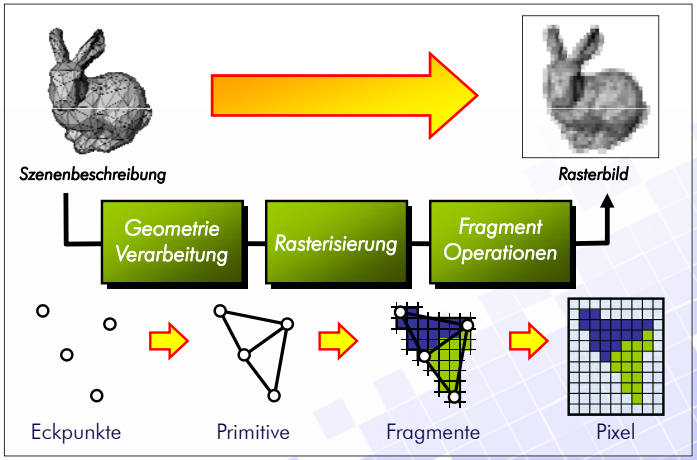
Gegeben:
Eckpunkte der Dreiecke

- *Dreiecke zeichnen*
  - Pixel bestimmen
  - Farbe interpolieren
  - Texturen anwenden
- *Compositing*
  - Primär-Farbe mit Textur
  - Verdeckung
  - Transparenz



Rendering Pipeline

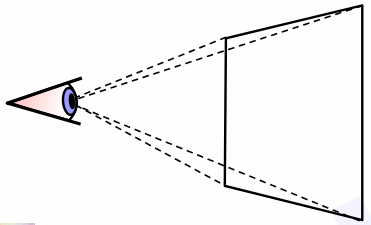
9



Der Frame-Buffer

10

- *Speicherbereich* auf der Graphikkarte, der das darzustellende Bild enthält.

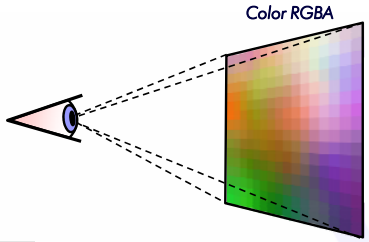


Color Buffer (optional Doublebuffer): Enthält für jeden Pixel den Farbwert *RGB* und die Opazität *A*.

Der Frame-Buffer

11

- *Speicherbereich* auf der Graphikkarte, der das darzustellende Bild enthält.

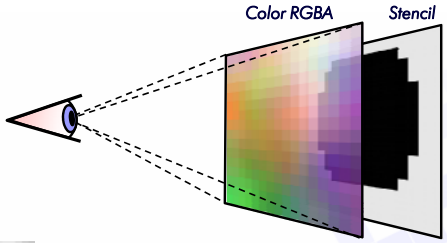


Stencil Buffer (optional): „Schablone“, enthält für jeden Pixel ein Flag, ob der Pixel verändert werden darf oder nicht („Schreibschutz“).

Der Frame-Buffer

12

- *Speicherbereich* auf der Graphikkarte, der das darzustellende Bild enthält.

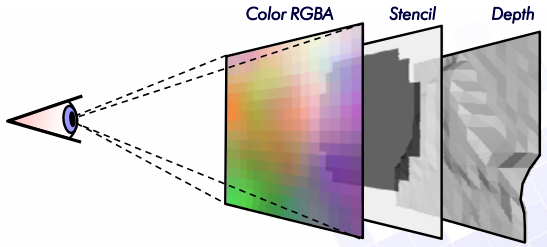


Depth Buffer: (auch z-Buffer), speichert für jeden Pixel den Tiefenwert, der bei der Projektion auf die Ebene verloren geht.

Der Frame-Buffer

13

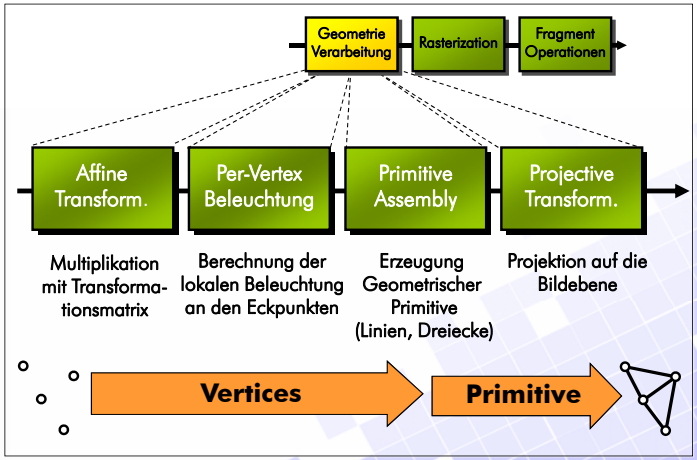
- *Speicherbereich* auf der Graphikkarte, der das darzustellende Bild enthält.



Alpha-, Stencil- und Depth-Buffer werden nur während des Renderings benötigt.
Dargestellt wird natürlich nur der RGB-Color(Front) Buffer.

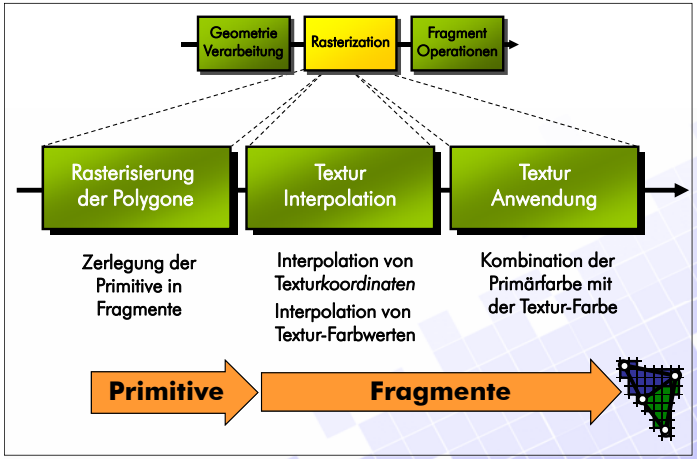
Geometrieverarbeitung

14



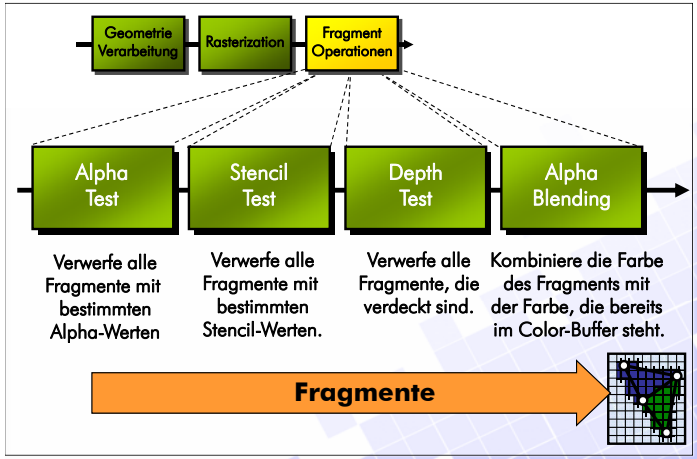
Rasterisierung

15



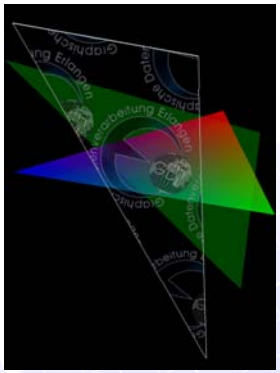
Fragment Operationen

16



Was kann die Hardware?

- Rasterisierung
  - Zerlegung in Fragmente
  - Interpolation der Farbe
  - Texturierung
    - TexCoord-Interpolation (Textur Interpolation)
    - Kombination (Textur-Anwendung)
- Fragment Operationen
  - Tiefentest (Z-Test)
  - Alpha Blending (Compositing)



Literatur Graphik-Hardware

J. Foley, A. van Dam, S. Feiner, and J. Hughes.
Computer Graphics, Principle And Practice.
Addison-Wesley, 1993.

OpenGL ARB.
The OpenGL Programming Guide *(Red Book)*
Addison-Wesley.

Außerdem:
Vorlesung „Computergraphik“

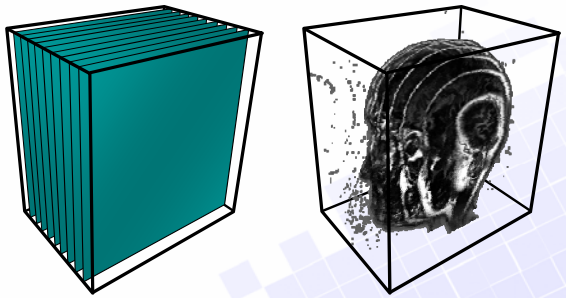
Volume Rendering in Hardware

- **Volume Rendering** (z.B. Ray-Casting) bedeutet eine hohe Anzahl an Interpolationen.
- **Idee:** Nutze die beschleunigte Interpolation der Graphik-Hardware (***Rasterisierung***)
- **Interpolation** findet (u.a.) in der Texture-Unit statt
- **Problem:** Die Rasterhardware unterstützt keine volumetrischen Primitive (nur ebene Polygone bzw. Dreiecke)

Texturbasierte Ansätze

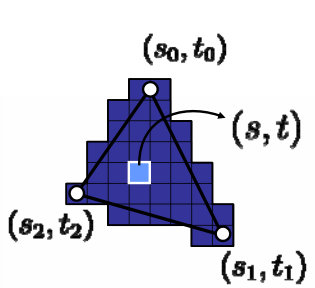
Hardware unterstützt keine volumetrischen Primitive

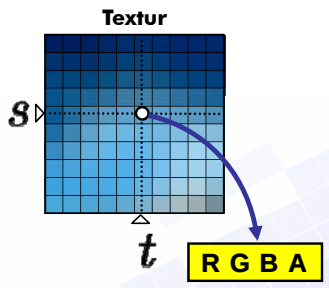
⇨ Hilfsgeometrie ***(Polygonale Schichten)***



Erinnerung Texturen

21





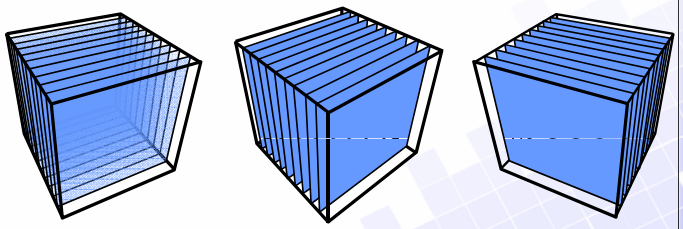
Für jedes Fragment: Interpolation der Texturkoordinaten **(baryzentrisch)**

Texture-Lookup: Interpolation der Texturfarbe **(bilinear)**

2D Texturen

22

- Bilineare Interpolation in Hardware
- ⇨ Zerlegung des Volumens in achsen-parallele Schichten *(analog Shear Warp)*



- 3 Kopien des Datensatzes im Speicher!

2D Texturen

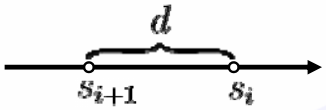
23

- *Was wird in der Textur gespeichert?*

RGB | **A**

Emissionsterm $RGB = \int_{s_i}^{s_i+d} q(s)\,ds$

Absorptionsterm $A = e^{-\tau(s_i, s_i+d)}$

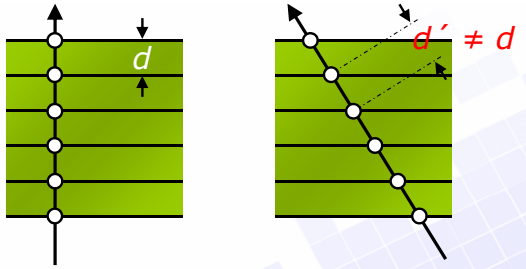


Die RGBA Werte, die in durch, die Transferfunktion spezifiziert und in die Textur geschrieben werden, beziehen sich auf eine konkrete Abtastlänge d

2D Texturen

24

Für die numerische Integration (blend-Operation) muss der Abstand d konstant sein.

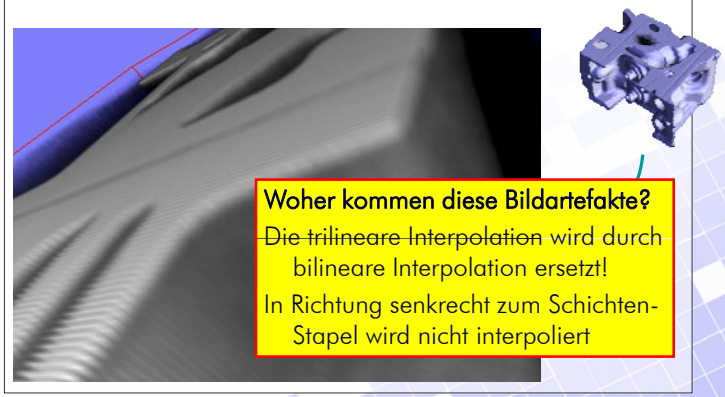


- Ansonsten: Inkorrekte (diskrete) Emission/Absorptionswerte
- Keine Überabtastung (Erhöhung der Schichtenzahl) möglich!

2D Texturen

25

- Nachteil – Visuelle Bildartefakte (bei starkem Zoom)



Woher kommen diese Bildartefakte?

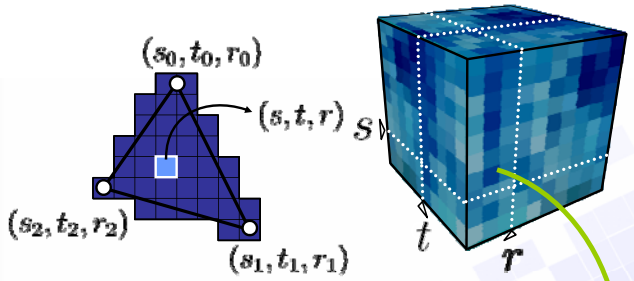
~~Die trilineare Interpolation~~ wird durch bilineare Interpolation ersetzt!

In Richtung senkrecht zum Schichten-Stapel wird nicht interpoliert

Source: C Rezk-Salama, Erlangen

3D Texturen

26



Für jedes Fragment: Interpolation der Texturkoordinaten (baryzentrisch)

Texture-Lookup: Interpolation der Texturfarbe (trilinear)

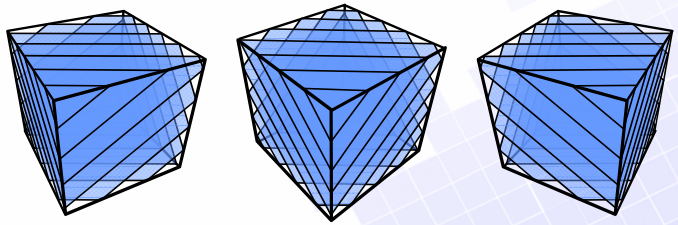
R G B A

3D Texturen

27

- 3D Textur: Volumetrisches Textur-Objekt
- Aber kein volumetrisches Rendering-Primitiv!
- Volumen als großer Texturblock im Speicher

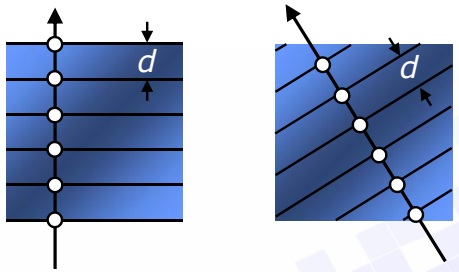
⇨ Rasterisierung durch Schichten/Slices parallel zur Bildebene



Resampling bei 3D Texturen

28

- Abtastrate ist konstant (für Parallelprojektion)

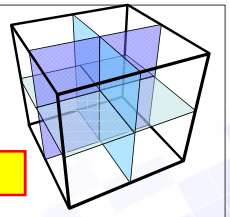


- Überabtastung einfach durch Erhöhung der Schichtenzahl.

Bricking

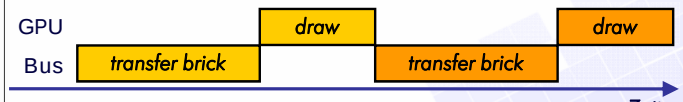
29

- Was wenn Datensatz > GPU-Speicher (aktuelle GPUs können mind. 512^3)
  - ⇨ Teile den Datensatz in kleinere Subvolumina (Bricks)



Problem: Bus-Bandbreite

- *Schlechte Auslastung von GPU und Bus*



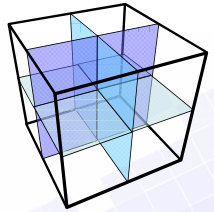
- *GPU und Bus warten aufeinander (idle wait)*

Bricking

30

- Schlechte Auslastung von GPU und Bus

Mögliche Lösungen:



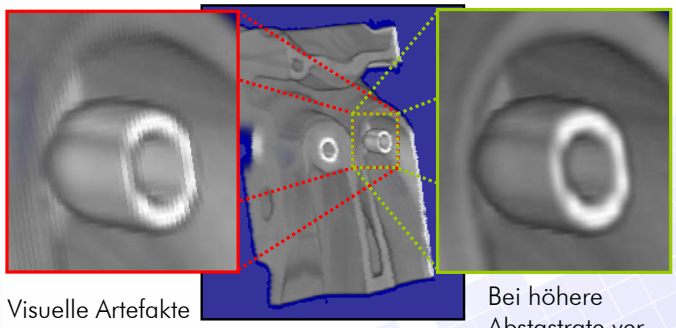
- Mache die Bricks klein genug! Mehr als ein Brick muß in den Graphikspeicher passen!
  - Transfer und Rendering können parallel ausgeführt werden.
  - Erhöhte CPU Last für Schnittberechnung!
  - Effektives Load Balancing ist sehr schwierig!

Verwende 2D Multi-Texturen anstatt 3D Texturen!

Zurück zu 2D Texturen

31

Schneller als 3D Texturen, aber mindere Bildqualität



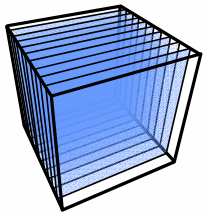
Visuelle Artefakte wegen fester Schichtenzahl

Bei höhere Abstastrate ver-schwinden diese Artefakte

2D Multi-Texturen

32

- Idee: Verwende achsenparallele Schichten
  - ⇨ Zerlegung des Volumens in achsen-parallele Schichten (wie gehabt)



Neu: Erzeuge neue Schichten mit dem nötigen Samplingabstand "zwischen" den Texturen on-the-fly

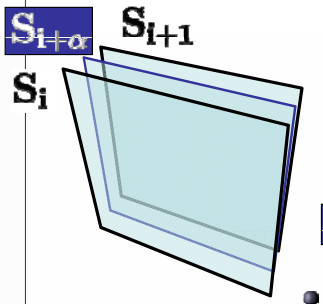
- Interpoliere die neue Texturschicht aus zwei benachbarten Texturen

- 3 Kopien des Datensatzes im Speicher!

2D Multi-Textures

33

Achsenparallele Schichten



- Bilineare Interpolation durch 2D Textur
- Blending zweier Schichten (Texturkombination)

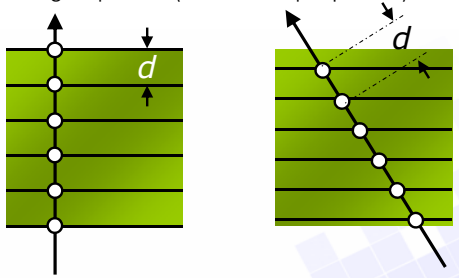
$$S_{i+\alpha} = (1-\alpha)S_i + \alpha \cdot S_{i+1}$$

- Ergebnis:
  Trilineare Interpolation

2D Multi-Textures

34

- Abtastrate ist konstant wenn ich den Schichtabstand an die Blickrichtung anpasse! (für Parallelprojektion)

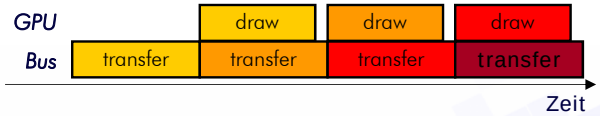


- Überabtastung ist einfach durch Erhöhung der Schichtenzahl. Textur wird on-the-fly interpoliert.

Vorteile

35

- Effizienteres Load-Balancing:



  - Nutze GPU und verfügbare Speicherbandbreite parallel!
  - Transferiere pro Schicht die minimale Information die nötig ist um diese Schicht zu zeichnen (feine Granularität)!
- **Erheblich höhere Performanz**, obwohl 3 Kopien des Datensatzes im Speicher sind!

Literatur

36

B. Cabral, N. Cam, and J. Foran.
Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware.
ACM Symposium on Volume Visualization, 1994.

O. Wilson, A. Van Gelder, and J. Wilhelms.
Direct Volume Rendering via 3D Textures.
Technical Report UCSC-CRL-94-19, Univ. of California, Santa Cruz, 1994.

C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl.
Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization.
Proceedings of the SIGGRAPH/Eurographics Workshop on Graphics Hardware, 2000.

Zusammenfassung

Graphik-Hardware

- Geometrieverarbeitung
- Rasterisierung
- Fragment Operationen

Texturbasierte Verfahren

- Zerlege das Volumen in Schichten (Hilfsgeometrie)
- **2D Texturen** (schnell, aber geringe Qualität)
- **3D Texturen** (schnell, gute Qualität, aber Probleme mit großen Datensätzen!)
- **2D Multi-Texturen** (schnell, gute Qualität)

Nächste Stunde

- Klassifikation und Shading.
  - Präklassifikation und Postklassifikation
  - Lokale Beleuchtung für Volumina