Volumenvisualisierung: Klassifikation und Shading

Andreas Kolb und Martin Lambers

computergraphik und multimedia systeme
universität siegen

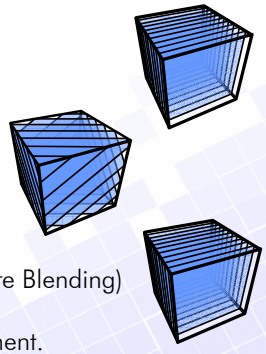
Letzte Stunde

Texturbasierte Volumenvisualisierung
- Ausnutzung der Standard-Grafik-Hardware

2D Texturen:
- bilineare Inteprolation
- achsenparallele Schichten

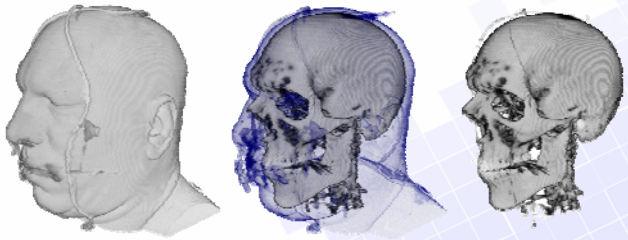
3D Texturen:
- trilineare Interpolation
- Viewport-parallele Schichten
- ein Texturblock, Bricking

2D Multitexturen
- trilineare Interpolation (Texture Blending)
- achsenparallele Schichten
- Effizientes Memory Management.



Klassifikation

- Im Klassifikationsprozess spezifiziert der Benutzer das *„Aussehen“* seiner Daten.
  - Welche Bereiche sollen transparent sein?
  - Welche Bereiche sollen welche Farbe haben?

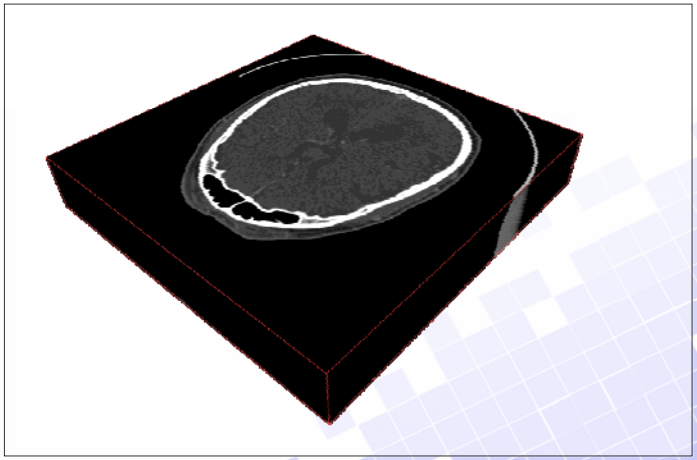


Klassifikation

- Im Klassifikationsprozess spezifiziert der Benutzer das *„Aussehen“* seiner Daten.
  - Welche Bereiche sollen transparent sein?
  - Welche Bereiche sollen welche Farbe haben?
- Der Benutzer definiert eine *Transferfunktion*

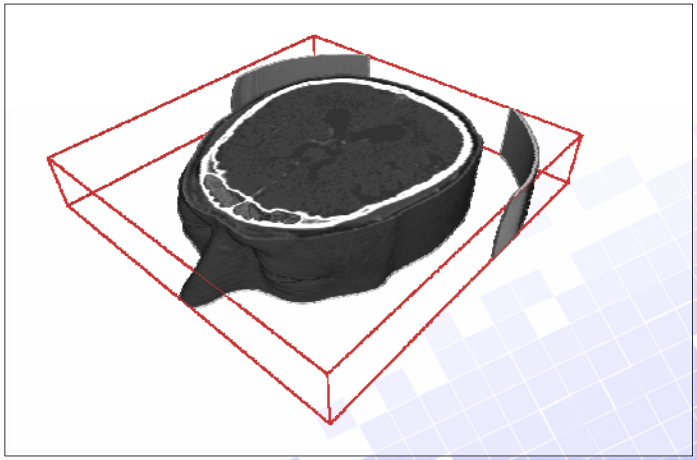
Beispiel: CT Angiographie

5



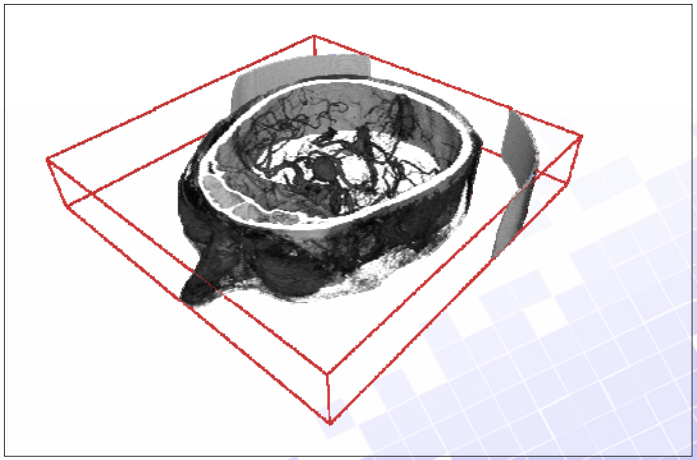
Beispiel: CT Angiographie

6



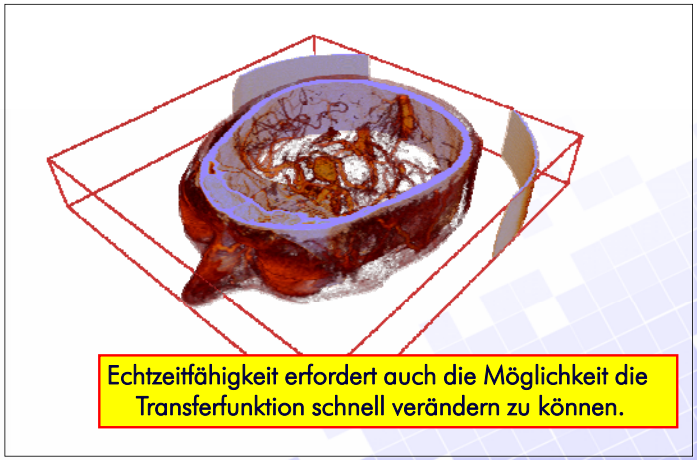
Beispiel: CT Angiographie

7



Beispiel: CT Angiographie

8



Echtzeitfähigkeit erfordert auch die Möglichkeit die Transferfunktion schnell verändern zu können.

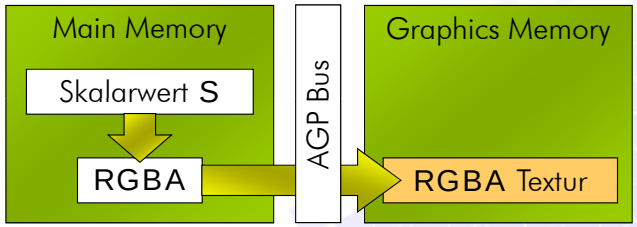
Klassifikation – the easy way

- **Präklassifikation:**
  Farbtabelle wird *vor* der Interpolation angewendet.
  (präinterpolative Transferfunktion)

- Ein Skalarwert wird in den Daten nachgeschaut
  für jeden Voxel
- Ein RGBA Wert wird mittels Farbtabelle bestimmt
  für jeden Voxel

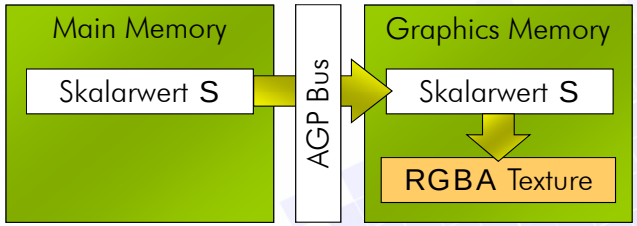
Mögliche Implementierungen

- Der *naive* Ansatz:
  Speichere die Emissions- und Absorptionsterme direkt in der Textur.



Mögliche Implementierungen

- Eine bessere Lösung:
  Anwendung der Farbtabelle während des Textur Transfers vom Main Memory auf die Graphikkarte (standard OpenGL feature)

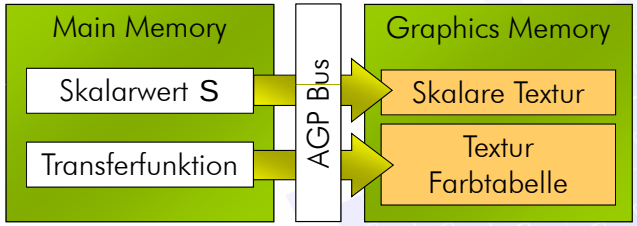


Mögliche Implementierungen

- Eine bessere Lösung:
  Anwendung der Farbtabelle während des Textur Transfers vom Main Memory auf die Graphikkarte (standard OpenGL feature)
- Hoher Speicherbedarf
  - Main Memory (nur Skalarvolumen)
  - Graphics Memory (RGBA Volumen)
- Verringerte Last auf dem Speicherbus
  - Nur das Skalarvolumen wird transferiert.
- Upload notwendig bei Änderung der TF

Mögliche Implementierungen

- Der *beste* Ansatz: Texturpaletten
  Speichere das Skalarvolumen zusammen mit der Farbtabelle direkt im Grafikspeicher.
- Hardwareunterstützung dafür ist notwendig!



Mögliche Implementierungen

- Der *beste* Ansatz: Texturpaletten
  Speichere das Skalarvolumen zusammen mit der Farbtabelle direkt im Grafikspeicher.
- Hardwareunterstützung dafür ist notwendig!
- Geringer Speicherbedarf
  - Main Memory (Skalarvolumen kann gelöscht werden!)
  - Graphics Memory (Skalarvolumen + TF)
- Geringe Last auf dem Speicherbus
  - Skalarvolumen muss **nur einmal** transferiert werden!
- Nur die Farbtabelle muß bei Änderungen der TF hochgeladen werden!

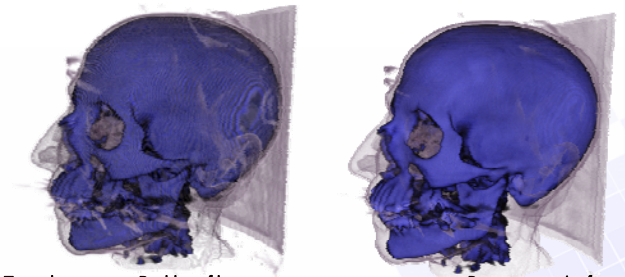
Klassifikation – the easy way

- Zusammenfassung Präklassifikation
  - Anwendung der Transferfunktion vor der Rasterisierung
  - Ein RGBA Lookup für jeden Voxel
  - Unterschiedliche Implementierungen:
    - Texture Transfer
    - Texture Color Tables (paletted textures)
  - Einfach und Effizient

… aber Präklassifikation ist nicht wirklich gut!

Warum Präklassifikation nicht gut ist.

- Vergleiche die Bildqualität mit Raycasting



Texturbasiert mit Präklassifikation — Ray casting (software)

Gleiche TF, gleiche Auflösung, gleiche Abtastrate

Warum ist die Qualität soviel schlechter?

Abtasttheorem

17

Das Abtasttheorem verlangt:

- Das kontinuierliche Skalarfeld ist bandbegrenzt:
  Ein kontinuierliches Signal $f(x)$, dessen Frequenzspektrum $F(t)$ bandbegrenzt ist,
  d.h.

$$F(t) = 0 \quad \text{für} \quad |t| > t_{max}$$

läßt sich ***exakt rekonstruieren*** aus einer diskreten Abtastung mit einer Schrittweite:

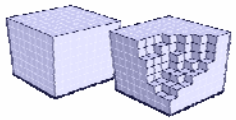
$$\tau > \frac{1}{2t_{max}}$$

Abtasttheorem

18

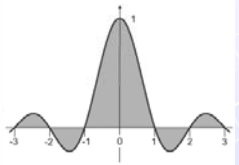
- Kontinuierliches 3D Skalarfeld

Diskrete Voxel Daten → Rekonstruction



Der diskrete Datensatz enthält die gesamte Information des kontinuierlichen Skalarfeldes

ideal: SINC-Filter

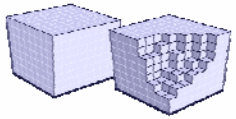


Abtasttheorem

19

- Kontinuierliches 3D Skalarfeld

Diskrete Voxel Daten → Rekonstruction



Der diskrete Datensatz enthält die gesamte Information des kontinuierlichen Skalarfeldes

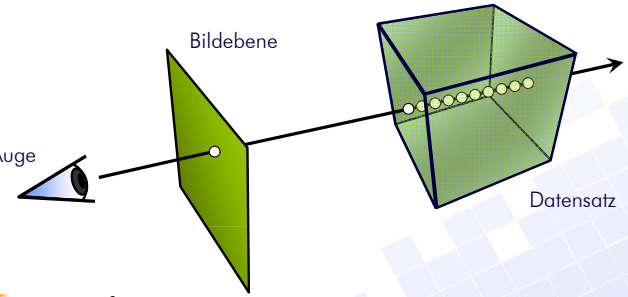
real: Dreiecksfilter

trilineare Interpolation

Rekonstruktion

20

- Wir visualisieren das kontinuierliche Skalarfeld!
  Also wann wird das kontinuierliche Signal rekonstruiert?

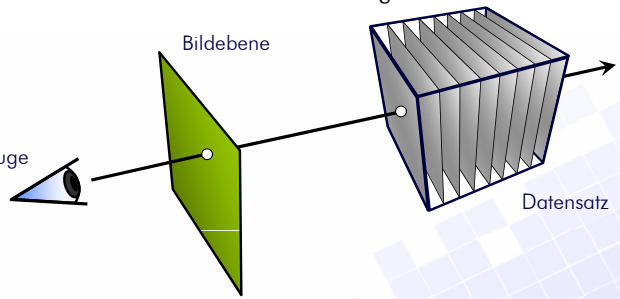


- *Antwort für Raycasting:*
  Rekonstruktion während der Integration (beim Resampling)

Rekonstruktion

21

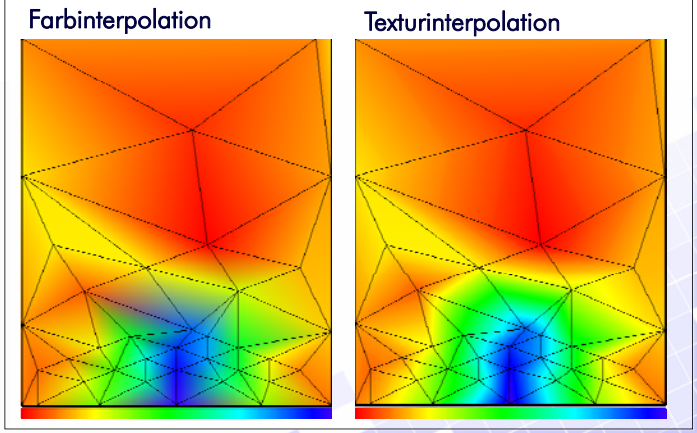
- Wir visualisieren das kontinuierliche Skalarfeld!
  Also wann wird das kontinuierliche Signal rekonstruiert?



- Antwort für texturbasierte Verfahren:
  Rekonstruktion während Texture Mapping/Rasterization

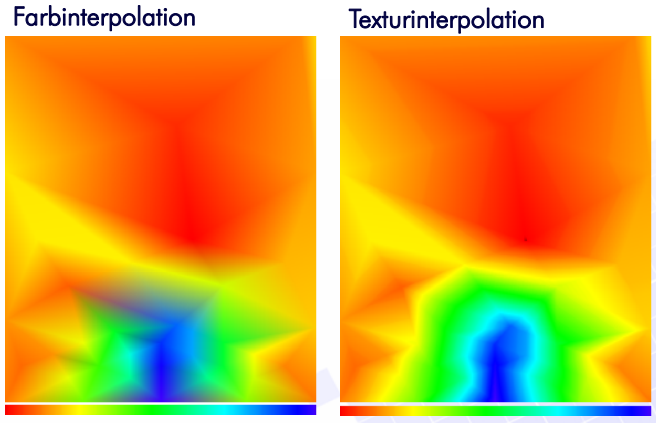
Erinnerung: 2D Skalarfelder

22



Erinnerung: 2D Skalarfelder

23



Warum Präklassifikation nicht gut ist.

24

Abtasttheorem erfüllt für das Skalarfeld S

↓

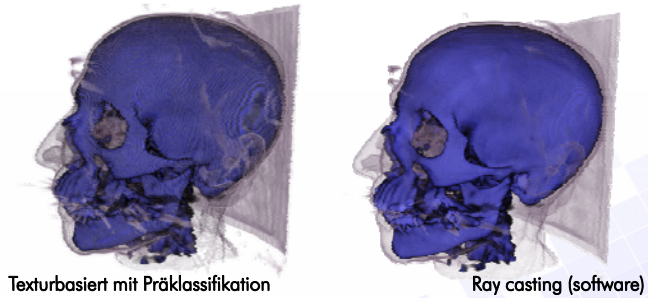
Ersetzte S durch T(S) *für jeden Voxel*

↓

Das Abtasttheorem ist nicht automatisch auch für T(S) erfüllt !!

- Die Transferfunktion T(S) ändert das Frequenzspektrum des Signals. Es werden Frequenzen hinzugefügt, die durch die ursprüngliche Schrittweite nicht erfasst werden.

Warum Präklassifikation nicht gut ist.



Texturbasiert mit Präklassifikation

Ray casting (software)

- Die Rekonstruktion muss das modifizierte Frequenzspektrum berücksichtigen, das durch die Transferfunktion entsteht.

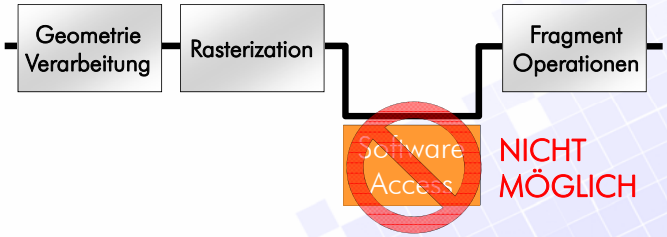
Klassifikation - the hard way

- **Postklassifikation:**
  Die Farbtabelle wird erst nach der Interpolation angewendet (***post-interpolative*** *Transferfunktion*).

- Eine Skalarwert wird interpoliert
  für jedes Fragment (Sample der Integration)!
- Ein RGBA-Wert wird mittels Farbtabelle bestimmt
  für jedes Fragment (Sample der Integration)!

Implementierung

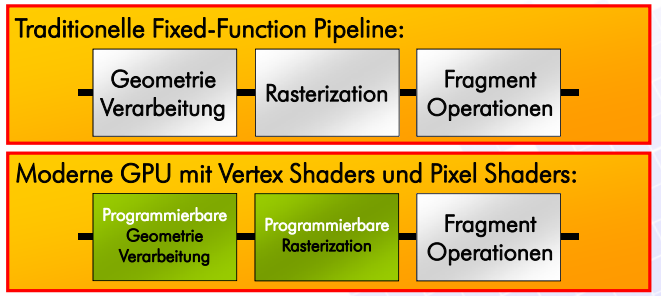
- Die Implementierung ist schwierig:
  Die Hardware erlaubt es nicht Werte aus der Pipeline zu extrahieren, in Software zu modifizieren und sie wieder einzufügen!



Implementations

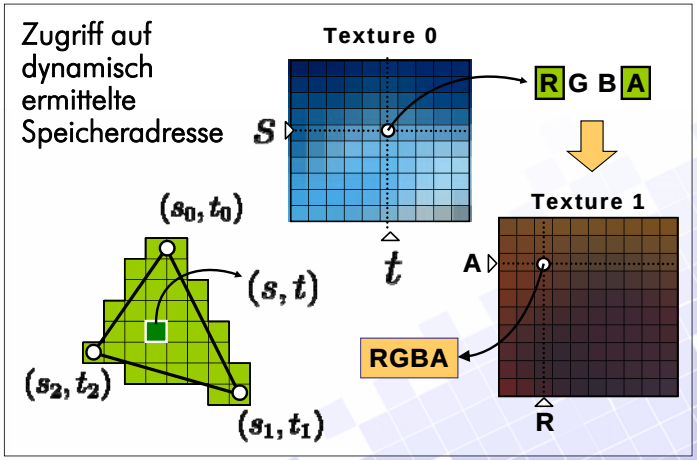
- Wir benötigen einen Mechanismus, der es uns erlaubt eine Farbtabelle anzuwenden, nachdem die Fragmente interpoliert wurden

➡ Solution: Moderne Graphikhardware



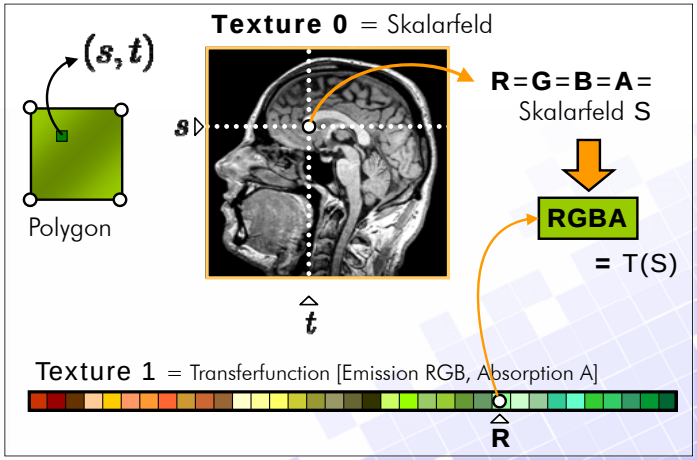
Dependent Texture Lookups

29



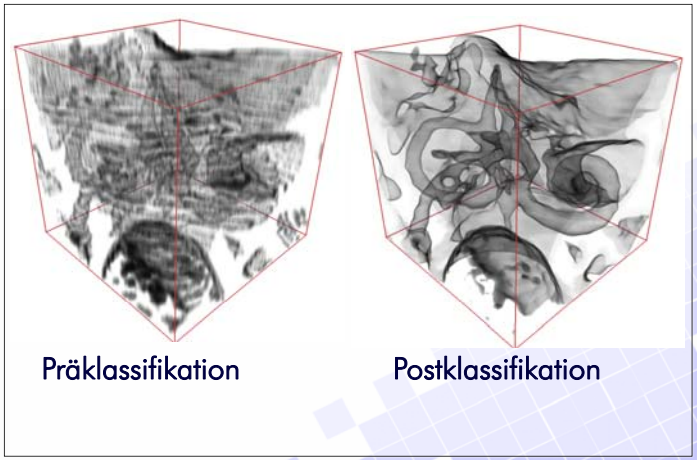
Postklassifikation

30



Vergleich

31



Klassifikation - the hard way

32

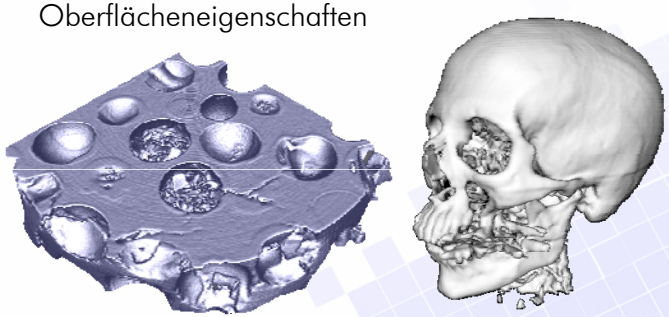
- *Zusammenfassung Postklassifikation*
  - Transferfunktion wird nach der Rasterisierung angewendet
  - Ein RGBA lookup für jedes Fragment (sample)
  - Dependent Textures:
    - Per Fragment Interpolation
    - Per-Fragment Color Lookup
  - Berücksichtigt Transferfunktionen mit hohen Frequenzen
    - einfach durch erhöhen der Abtastrate (Anzahl der Schichten)
  - Effiziente Implementation

Lokale Beleuchtung

33

Wozu brauche ich lokale Beleuchtung?

- Verbessert die Tiefenwahrnehmung
- Verbessert die Wahrnehmung von Oberflächeneigenschaften



Lokale Beleuchtung

34

- *Berücksichtige nur die lokale Form des Objekts*
  - kein Schatten
  - keine „*Self-Shadowing*“
- *Keine globalen Beleuchtungseffekte*
  - Indirekte Beleuchtung
  - Caustics

- *Ansatz: Das (Blinn-)Phong-Beleuchtungsmodell für* ***Oberflächen*** *betrachten auf* ***Volumendaten*** *übertragen.*

Globale Beleuchtung

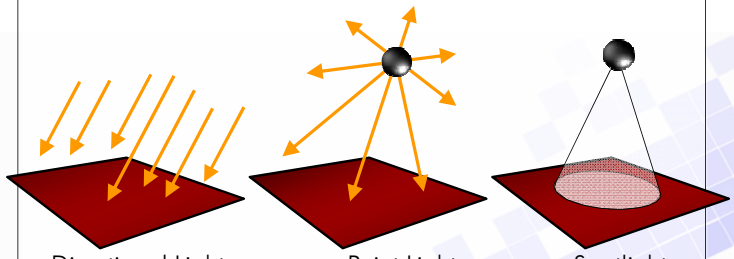
35

Globale Beleuchtungseffekte sollen hier nicht betrachtet werden!

Lichtquellen

36

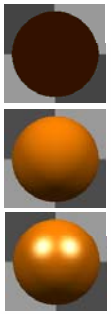
- Unterschiedliche Typen von Lichtquellen



Lokale Beleuchtung

37

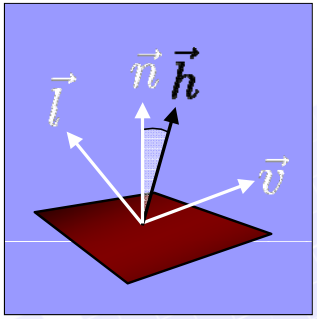
Blinn-Phong Model für Oberflächen



ambient

diffuse

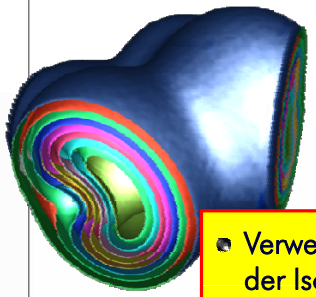
specular



$$I = I_a + I_d(\vec{n} \circ \vec{l}) + I_s(\vec{n} \circ \vec{h})^r$$

Beleuchtung für Voxeldaten

38



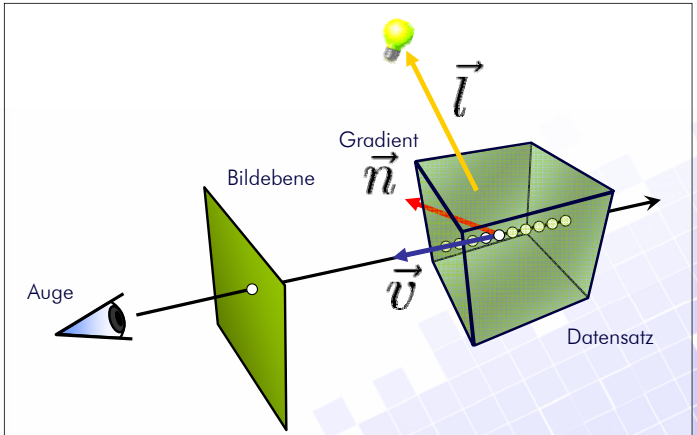
- Das Phong-Model benötigt die Oberflächennormale
- Wie sieht die Oberflächen-normale eines Voxels aus?

- Verwende die Oberflächennormale der Isofläche die den Voxel schneidet.

= *Gradientenvektor*

Raycasting

39



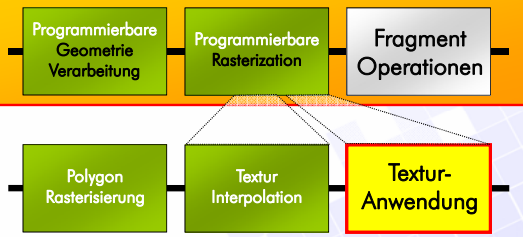
Texturbasierte Ansätze

40

- Wir benötigen einen Mechanismus, der es uns erlaubt Skalarprodukte zu berechnen, nachdem die Fragmente interpoliert wurden.

➡ Solution: Moderne Graphikhardware

Moderne GPU mit Vertex Shaders und Pixel Shaders:



Beleuchtung für Voxeldaten

41

Grundprinzip für Beleuchtung bei Texturbasierten Verfahren:

- Berechne den Gradientenvektor jedes Voxels vor,
- normalisiere ihn und
- speichere ihn in einer (zusätzlichen Multi-) Textur:

$$\vec{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \begin{matrix} \rightarrow \mathbf{R} \\ \rightarrow \mathbf{G} \\ \rightarrow \mathbf{B} \end{matrix}$$

- Bei der *Textur-Interpolation* wird für jedes Fragment ein Normalenvektor interpoliert
- Die Beleuchtungsberechnung (dot products) wird in der *Textur-Anwendung* durchgeführt.

Einfache Implementierung

42

- 1 Directional Light (Lichtrichtung $\vec{l}$ ist konstant)
- Nur diffuse Reflexion (kein spekularer Anteil)

Dies reduziert das Phong-Modell auf

$$I = I_d (\vec{n} \circ \vec{l})$$

Ausserdem setzen wir noch:

$$I_d = 1$$

Vereinfachter Beleuchtungsterm:

$$I = (\vec{n} \circ \vec{l})$$

Implementierungsbeispiel

43

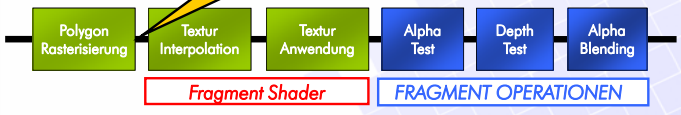
- Darstellung einer beleuchteten Isofläche *ohne vorher die polygonale Oberfläche zu bestimmen*. *(Non-polygonal Isosurfaces)*

$$\vec{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \begin{matrix} \rightarrow \mathbf{R} \\ \rightarrow \mathbf{G} \\ \rightarrow \mathbf{B} \end{matrix}$$

Skalarwert $S \rightarrow \mathbf{A}$

Übergebe Lichtrichtung als uniformen Parameter

$$\vec{l} = \begin{pmatrix} l_x \\ l_y \\ l_z \end{pmatrix}$$



Implementierungsbeispiel

44

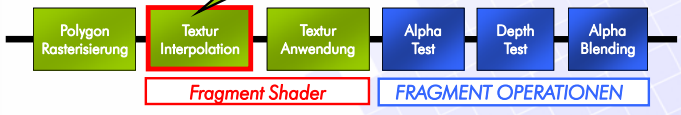
- Darstellung einer beleuchteten Isofläche *ohne vorher die polygonale Oberfläche zu bestimmen*. *(Non-polygonal Isosurfaces)*

$$\vec{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \begin{matrix} \rightarrow \mathbf{R} \\ \rightarrow \mathbf{G} \\ \rightarrow \mathbf{B} \end{matrix}$$

Skalarwert $S \rightarrow \mathbf{A}$

Interpoliere für jedes Fragment eine RGBA Wert aus der Textur:

- Normalenvektor (RGB)
- Skalarwert (A)



Implementierungsbeispiel

45

- Darstellung einer beleuchteten Isofläche *ohne vorher die polygonale Oberfläche zu bestimmen.* *(Non-polygonal Isosurfaces)*

$$\vec{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \\ \text{Skalarwert } S \end{pmatrix} \begin{matrix} \rightarrow \mathbf{R} \\ \rightarrow \mathbf{G} \\ \rightarrow \mathbf{B} \\ \rightarrow \mathbf{A} \end{matrix}$$

Berechne das Skalarprodukt zwischen Normalenvektor (Textur RGB) und Lichtvektor (Primär RGB)

$(\vec{n} \circ \vec{l})$

Implementierungsbeispiel

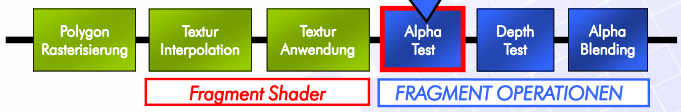
46

- Darstellung einer beleuchteten Isofläche *ohne vorher die polygonale Oberfläche zu bestimmen.* *(Non-polygonal Isosurfaces)*

$$\vec{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \\ \text{Skalarwert } S \end{pmatrix} \begin{matrix} \rightarrow \mathbf{R} \\ \rightarrow \mathbf{G} \\ \rightarrow \mathbf{B} \\ \rightarrow \mathbf{A} \end{matrix}$$

Verwerfe alle Fragmente, die nicht einem bestimmten Referenzwert entsprechen:

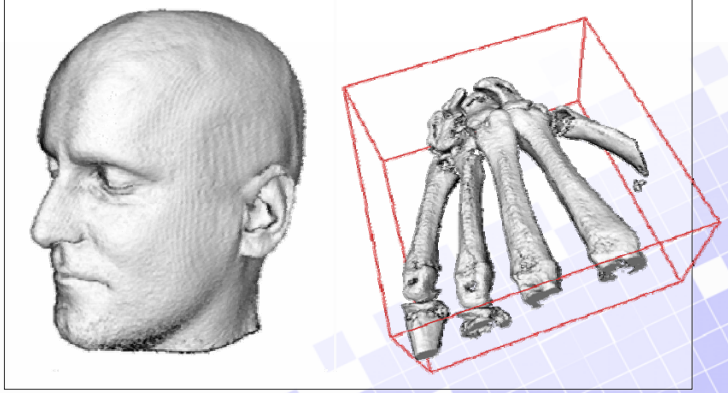
$\text{if } (S \neq S_{iso}) \text{ then discard}$



Beispiele

47

Non-polygonal Isosurface (mit diffuser Beleuchtung)



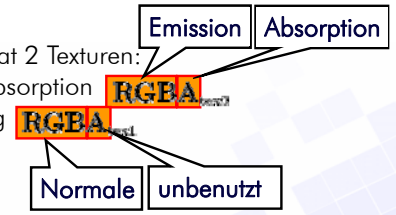
Weitere Möglichkeiten

48

Emission/Absorption mit lokaler Beleuchtung:

Multitexturen

Jedes Schichtpolygon hat 2 Texturen:
1 Textur für Emission/Absorption
1 Textur für Beleuchtung



Weitere Möglichkeiten

Emission/Absorption mit lokaler Beleuchtung:

Multitexturen

Jedes Schichtpolygon hat 2 Texturen:
1 Textur für Emission/Absorption $\mathbf{RGBA}_{tex0}$
1 Textur für Beleuchtung $\mathbf{RGBA}_{tex1}$

Textur-Anwendung:

$$\mathbf{RGB}_{final} = \mathbf{RGB}_{tex0} + \langle \mathbf{RGB}_{prim} \circ \mathbf{RGB}_{tex1} \rangle$$

(Emission) (Lichtrichtung) (Normale)

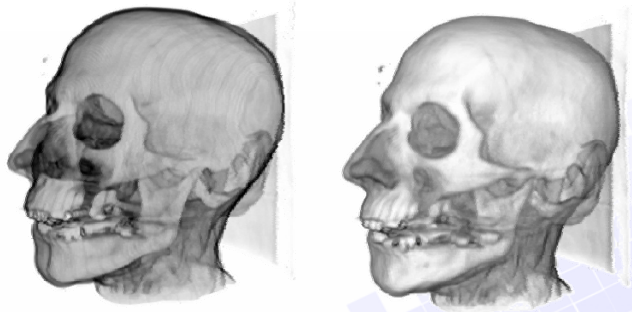
$$\mathbf{A}_{final} = \mathbf{A}_{tex0}$$ (Absorption)

Fragment-Operationen: Alpha-Blending:

$$\mathbf{RGB}_{new} = \mathbf{RGB}_{final} \cdot \mathbf{A}_{final} + \mathbf{RGB}_{old}(1 - \mathbf{A}_{final})$$

Beispiele

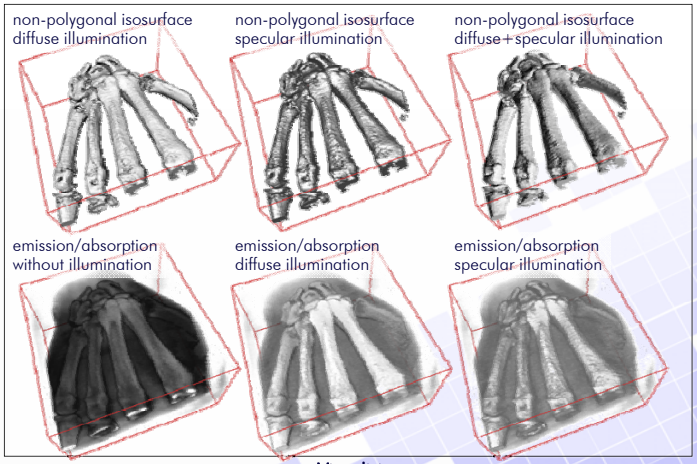
Emission/Absorption Model mit diffuser Beleuchtung



Emission/Absorption Model ohne Beleuchtung

Emission/Absorption Model mit diffuser Beleuchtung

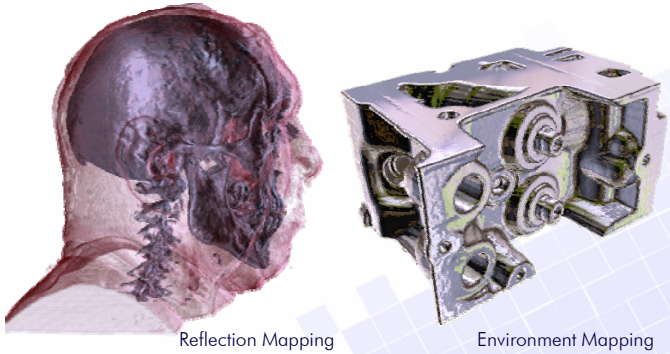
Weitere Möglichkeiten



Weitere Möglichkeiten

Reflection Mapping:

- Komplexe Beleuchtungsumgebungen vorberechnet



Reflection Mapping Environment Mapping

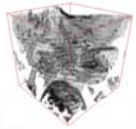
Zusammenfassung

Texturbasierte Volumenvisualisierung

- Klassifikation (Transferfunktionen)

Präklassifikation:

- Farbtabelle vor der Interpolation/Rasterisierung
- Aliasing Artefakte durch hohe Frequenzen in der Transferfunktion.
- Einfache Implementierung



Postklassifikation:

- Farbtabelle nach der Interpolation/Rasterisierung
- Anpassung an hohe Frequenzen in der Transferunktion durch Erhöhung der Schichtenzahl (Überabtastung).
- Aufwändigere Implementierung

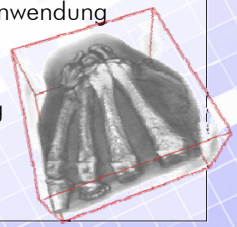
Zusammenfassung

Texturbasierte Volumenvisualisierung

- Lokale Beleuchtung (Shading)

Grundprinzip:

- Vorberechnung des Gradientenvektors als Ersatz für die Oberflächennormale (Blinn-Phong-Modell)
- Speichern in zusätzlicher Textur.
- Beleuchtungsberechnung in Textur-Anwendung

Implementierungsbeispiele:

- Non-polygonal Isosurfaces
- Emission/Absorption mit Beleuchtung