UNIVERSITÄT BONN

RHEINISCHE
FRIEDRICH-WILHELMS-
UNIVERSITÄT BONN

INSTITUT FÜR
INFORMATIK DER
UNIVERSITÄT BONN

Einführung in die Computergrafik und Visualisierung

Kapitel 13b: Texturierung

Prof. Dr. Matthias Hullin

Institut für Informatik II - Computergraphik
Universität Bonn

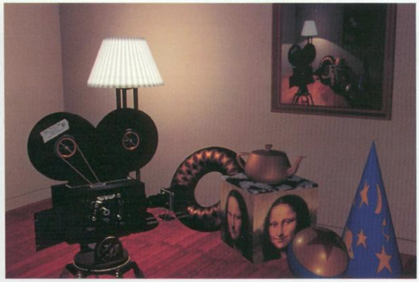
17. Juni 2019

Es gibt ein großes Spektrum geometrischer Formen und physikalischer Materialien, z.B.

- Maserungen und Muster (Holz, Marmorplatten und Tapeten)
- Wolken
- Strukturen unebener Oberflächen (Putzwände, Leder, Orangen und Baumstämme)
- Objekte im Hintergrund (Häuser, Maschinen, Pflanzen und Personen)

Solche Objekte durch Flächen nachzubilden ist in der Regel viel zu aufwendig.

Lösung: Mit Texturen kann man Objekte visuell komplexer gestalten. Man "klebt" über das Objekt eine Tapete bzw. ein Bild. Dies nennt man Texturierung .



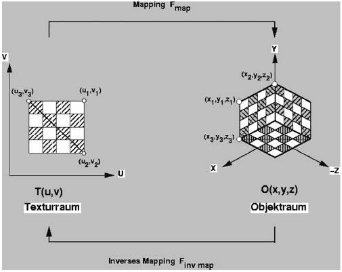
Eine 2D-Textur ist eine Funktion von (u, v) auf die Menge der RGB-Farben. Anschaulich:

$$(r, g, b) = C_{tex}(u, v)$$

Eine Abbildung (engl. Mapping) beschreibt, wie die Textur auf die Fläche aufgebracht wird.

Beim Rendering muss das inverse Mapping-Problem gelöst werden:

$$(u, v) = F_{inv\ map}(x, y, z)$$



Eine 2D-Textur ist eine Funktion von (u, v) auf die Menge der RGB-Farben.
Anschaulich:

- Textur :

$$(r, g, b) = C_{tex}(u, v)$$

- Inverses Mapping:

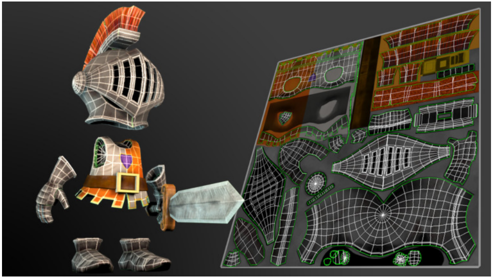
$$(u, v) = F_{inv\ map}(x, y, z)$$

Die Texturierung mit einer 2D-Textur ist die Hintereinanderausführung der Abbildungen:

$$(r, g, b) = C_{tex}(F_{inv\ map}(x, y, z))$$

uv-**Mapping:**

- Auch heterogene Texturen auf komplexen Objekten können so verwaltet werden
- Texturdaten leben in einer großen 2D-Karte (“Texturatlas”)
- Jeder Punkt auf der Objektoberfläche verweist auf eine uv-Koordinate im Texturraum

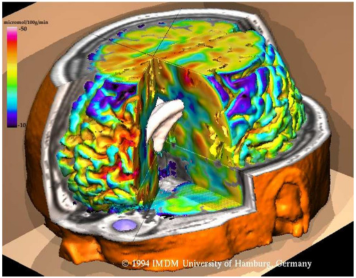


Quelle: https://pluralsight.imgix.net/course-images/3ds-max-uv-mapping-fundamentals-v1.jpg

Eine 3D-Textur ist eine Funktion von (u, v, w) auf die Menge der RGB-Farben. Anschaulich:

$$(r, g, b) = C_{tex}(u, v, w)$$

- 3D-Texturen nennt man auch Festkörpertexturen (z.B. Holz und Marmor)
- Bei inversem Mapping werden (x, y, z) auf (u, v, w) abgebildet. Das Objekt wird quasi aus dem Texturkörper herausgeschnitzt



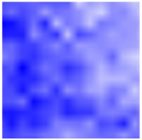
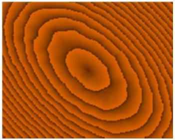
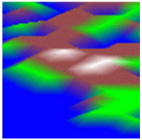
Diskrete Texturen: Eine diskrete N-dimensionale Textur wird als $(N+1)$-dimensionales Zahlenfeld gespeichert.
Eine diskrete 2D-Textur ist somit als 2-dimensionales Array gegeben durch:

$$C[i,j] \quad , \quad 0 \leq i < m, 0 \leq j < n$$

Dabei ist $C[i,j]$ ein RGB-Vektor, auch Texel (Textur Element) genannt.

Prozedurale Texturen: Eine prozedurale Textur ist eine mathematische Funktion und wird beim Lesen erst berechnet:

$$C_{tex}(u,v) := f(u,v) \qquad \text{bzw.} \qquad C_{tex}(u,v,w) := f(u,v,w)$$


Diskrete Texturen:

Vorteile:

- Vorrat an Bildern nahezu unerschöpflich
- Erzeugung einfach (z.B. digitale Photographie)

Nachteile:

- Kontext (Sonnenstand, Schattenwurf, etc.) stimmt meist nicht
- Bilder hoher Auflösung haben großen Speicherbedarf
- Fortsetzung meist sehr kompliziert
- Beim Vergrößern treten Artefakte auf
- Verzerrung beim Mapping auf beliebige Fläche
- Texturwerte an (u, v) müssen aus Texelwerten rekonstruiert werden

Prozedurale Texturen:

Vorteile:

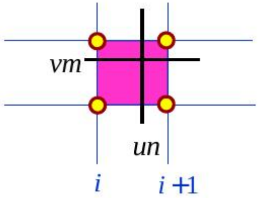
- Speicheraufwand ist minimal (nur Funktion speichern)
- Texturwerte können an jeder Stelle (u, v), bzw. (u, v, w) berechnet werden
- Optimale Genauigkeit (keine Interpolation)
- Texturen sind im gesamten Raum definiert

Nachteile:

- Schwer zu erzeugen (selbst für Experten)
- Mindestens Grundkenntnisse der Fouriersynthese, bzw. fraktaler Geometrie erforderlich
- Komplexere Texturen nahezu unmöglich

1. Nächster Nachbar (nearest neighbour)

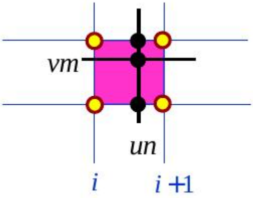
$$C_{tex}(u,v) = C\left[\left\lfloor un + \frac{1}{2}\right\rfloor \bmod n, \left\lfloor vm + \frac{1}{2}\right\rfloor \bmod m\right]$$



n und m beschreiben hierbei die Anzahl der Farbsegmente, zwischen denen interpoliert wird.

2. Bilineare Interpolation

$$\hat{u} = un - \lfloor un \rfloor \qquad \hat{v} = vm - \lfloor vm \rfloor$$
$$C_j(u,v) = \hat{u} \cdot C\left[\lfloor un \rfloor + 1, \lfloor vm \rfloor + j\right] + (1-\hat{u}) \cdot C\left[\lfloor un \rfloor, \lfloor vm \rfloor + j\right]$$
$$C_{tex}(u,v) = \hat{v} \cdot C_1(u,v) + (1-\hat{v}) \cdot C_0(u,v)$$

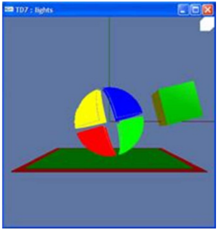


n und m beschreiben hierbei die Indices der Farbsegmente, zwischen denen interpoliert wird.

Wie kann ein Texturwert die Beleuchtungsrechnung beeinflussen?

1. Ersetzen der Objektfarbe (replace)

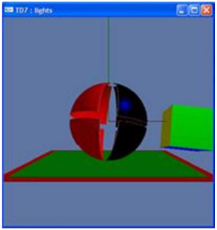
$$L_{out} = C_{tex}(u, v)$$



- Einfachste Art der Texturierung
- Jegliche Beleuchtung wird entfernt

2. A posteriori Skalierung der Farbe (modulate)

$$L_{out} = L_{Phong} \cdot C_{tex}(u, v)$$



- Komponentenweise Skalierung des Farbwertes

3. A priori Skalierung der Materialfarbe

$$r_a = k_a \cdot C_{tex}(u,v) \qquad r_d = k_d \cdot C_{tex}(u,v)$$

- Farbe im wesentlichen durch ambiente (r_a) und diffuse (r_d) Komponenten bestimmt
- Im Unterschied zu 2. bleibt der spekulare Anteil von der Textur unbeeinflusst

4. Modulation der spekularen Reflektion

$$r_s = k_s \cdot C_{tex}(u,v)$$

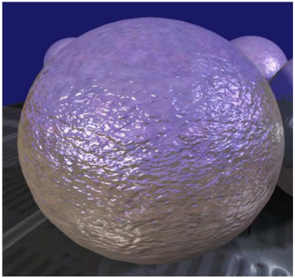
- Analog zu 3. Modulation von r_s
- Erlaubt Gestaltung unregelmäßiger Highlight (z.B. Verschmutzte Flächen)

5. Modulation der Transparenz

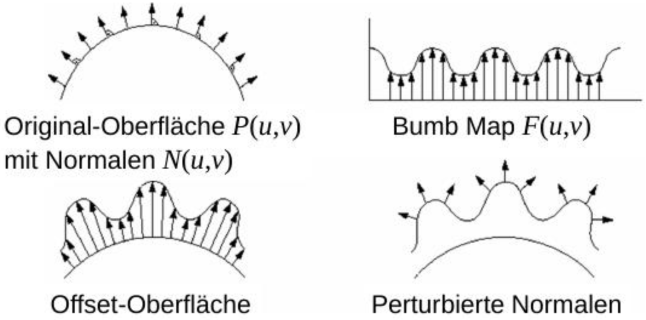
$$\alpha = \alpha_{obj} \cdot \alpha_{tex}(u,v)$$

- Speichern der Durchsichtigkeit in einer Textur
- Pixel mit $\alpha = 0$ sind durchsichtig und Pixel mit $\alpha = 1$ sind undurchsichtig
- Ermöglicht komplexe Formen mit einfacher Geometrie (Billboards)

6. Perturbation der Normale (Bump-Mapping)
    - Speichern von Höhenwerten einer Offsetfläche in einer skalaren Textur
    - Erste Ableitungen in u- und v-Richtung perturbieren die Flächennormale

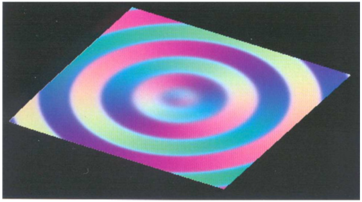


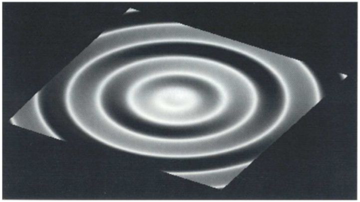
Oberfächennormale wird perturbiert, wie wenn der Punkt versetzt worden wäre:



$$P^{'}(u,v) = P(u,v) + F(u,v) \cdot \frac{N(u,v)}{\|N(u,v)\|_2}$$

Beispiel:





In die Beleuchtungsrechnung geht nicht direkt $P(u,v)$ ein, sondern nur $N(u,v)$.

Idee: Für kleine Unebenheiten reicht die Visualisierung von $P(u,v)$ mit veränderten Normalen $N'(u,v)$. Der Offset der Oberfläche wird also vernachlässigt.

Diese Normalen lassen sich folgendermaßen berechnen:

$$N'(u,v) = \frac{\partial}{\partial u} P'(u,v) \times \frac{\partial}{\partial v} P'(u,v)$$

$$P'(u,v) = P(u,v) + F(u,v) \cdot \frac{N(u,v)}{\|N(u,v)\|_2}$$

Die Richtungsableitungen lassen sich mithilfe von Summen- und Kettenregel berechnen:

$$\frac{\partial}{\partial u} P'(u,v) = \frac{\partial}{\partial u} P(u,v) + \frac{\partial}{\partial u} F(u,v) \cdot \frac{N(u,v)}{\|N(u,v)\|_2} + F(u,v) \cdot \frac{\partial}{\partial u} \frac{N(u,v)}{\|N(u,v)\|_2}$$

$$\frac{\partial}{\partial v} P'(u,v) = \frac{\partial}{\partial v} P(u,v) + \frac{\partial}{\partial v} F(u,v) \cdot \frac{N(u,v)}{\|N(u,v)\|_2} + F(u,v) \cdot \frac{\partial}{\partial v} \frac{N(u,v)}{\|N(u,v)\|_2}$$

Da wir annehmen, dass $F(u,v)$ sehr klein ist (kleine Unebenheiten), kann dieser Term vernachlässigt werden:

$$\frac{\partial}{\partial u} P'(u,v) \approx \frac{\partial}{\partial u} P(u,v) + \frac{\partial}{\partial u} F(u,v) \cdot \frac{N(u,v)}{\|N(u,v)\|_2}$$

$$\frac{\partial}{\partial v} P'(u,v) \approx \frac{\partial}{\partial v} P(u,v) + \frac{\partial}{\partial v} F(u,v) \cdot \frac{N(u,v)}{\|N(u,v)\|_2}$$

Für $N^{'}(u,v)$ folgt dann:

$$\begin{aligned}
N^{'} &= \frac{\partial P'}{\partial u} \times \frac{\partial P'}{\partial v} \\
&\approx \frac{\partial P}{\partial u} \times \frac{\partial P}{\partial v} + \frac{\partial F}{\partial u}\left(\frac{N}{\|N\|_2} \times \frac{\partial P}{\partial v}\right) \\
&+ \frac{\partial F}{\partial v}\left(\frac{\partial P}{\partial u} \times \frac{N}{\|N\|_2}\right) + \frac{\partial F}{\partial u} \cdot \frac{\partial F}{\partial v}\left(\frac{N}{\|N\|_2} \times \frac{N}{\|N\|_2}\right) \\
&= \frac{\partial P}{\partial u} \times \frac{\partial P}{\partial v} + \frac{\partial F}{\partial u}\left(\frac{N}{\|N\|_2} \times \frac{\partial P}{\partial v}\right) + \frac{\partial F}{\partial v}\left(\frac{\partial P}{\partial u} \times \frac{N}{\|N\|_2}\right) \\
&= N + \frac{1}{\|N\|_2}\left(\frac{\partial F}{\partial u}\left(N \times \frac{\partial P}{\partial v}\right) - \frac{\partial F}{\partial v}\left(N \times \frac{\partial P}{\partial u}\right)\right)
\end{aligned}$$

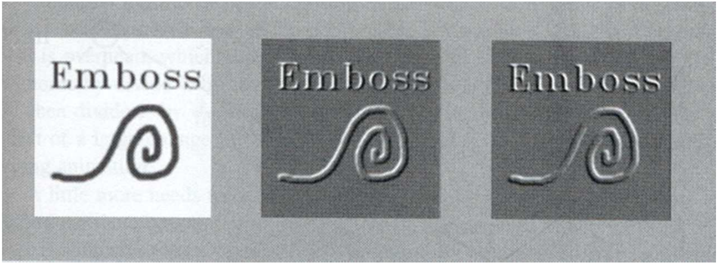
$$N^{'} = N + \frac{1}{\|N\|_2}\left(\frac{\partial F}{\partial u}\left(N \times \frac{\partial P}{\partial v}\right) - \frac{\partial F}{\partial v}\left(N \times \frac{\partial P}{\partial u}\right)\right)$$

Die Ableitungen $\frac{\partial F}{\partial u}$ und $\frac{\partial F}{\partial v}$ können mit finiten Differenzen approximiert werden:

- Vorwärtsdifferenz :
$$f^{'}(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{h}$$
- Rückwärtsdifferenz :
$$f^{'}(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{h}$$
- Zentrale Differenz :
$$f^{'}(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$$

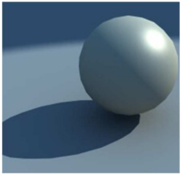
Speichern der Bump-Maps als Höhenfeld:

- Textur als Grauwertbild (mit Malprogramm erstellt)
- Richtungsableitungen (mit finiten Differenzen berechnet) in R/G gespeichert



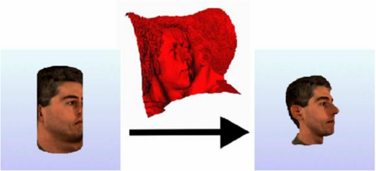
Links: Originalhöhenfeld , Mitte: u-Richtungsableitung , Rechts: v-Richtungsableitung

Beim Displacement Mapping wird die im Bump Mapping vernachlässigte Anpassung der Oberfläche (Offset Oberfläche) mitberücksichtigt:

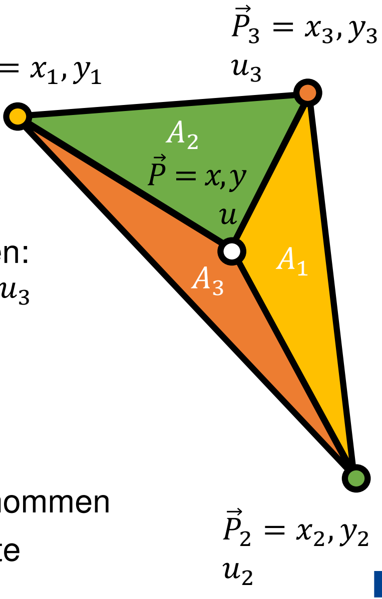


Links: Geometrie , Mitte: Bump Mapping , Rechts: Displacement Mapping

Im Gegensatz zu Bump Mapping sind auch größere Offsets möglich:



Baryzentrische Koordinaten zur Interpolation



- Verwende λ_i um zwischen Vertexattributen zu interpolieren:

$$u = \lambda_1 \cdot u_1 + \lambda_2 \cdot u_2 + \lambda_3 \cdot u_3$$

- Im Raytracer: sehr einfach umzusetzen
- Im Rasterisierer wird dies von spezialisierter Hardware übernommen
- Interpolation von λ_i: siehe letzte Vorlesung

Perspektivisch korrekte Interpolation

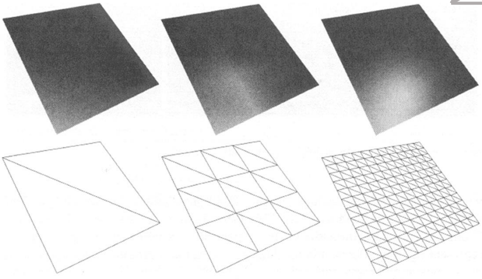
- Was wir gerne hätten:

- Was wir durch Interpolation mit 2D-screenspace-Koordinaten bekommen:

Perspektivische Interpolation

- Die perspektivische Projektion erhält Verhältnisse von Längen und Flächen nicht.
- Der Mittelpunkt einer Fläche liegt nach Projektion nicht mehr in der Mitte

- Einfachste Lösung: Unterteilen



Perspektivisch korrekte Interpolation von Attributen

Gegeben:

- Baryzentrische Koordinaten $\lambda_{1,2,3}$ von Punkt $\vec{P}^s$ (in 2D)
- Vertices in "clip-space" Koordinaten, $\vec{P}^v_{1,2,3} = (x, y, z, w)^v_{1,2,3}$
  [der Einfachheit halber nehmen wir an $w = z$]

Perspektivisch korrekte Tiefeninterpolation durch Interpolation von $\frac{1}{w}$:

$$1/w = \lambda_1 \cdot 1/w_1 + \lambda_2 \cdot 1/w_2 + \lambda_3 \cdot 1/w_3$$

Verwende interpoliertes w, um Attribut u zu interpolieren:

$$u = w \cdot \left(\lambda_1 \cdot u_1/w_1 + \lambda_2 \cdot u_2/w_2 + \lambda_3 \cdot u_3/w_3\right)$$

(vgl. [Lindholm et al. 2008])

Bewertung:

- Setzt voraus, dass für jeden Eckpunkt Texturkoordinaten spezifiziert werden
- Für ein Objekt aus wenigen Dreiecken kann man diese noch von Hand zuweisen
- Ist jedoch zu aufwendig für komplexe Objekte

Zweischrittverfahren (Bier & Sloan 1986):

Grundidee:

1. Umhülle Objekt mit einfacher virtueller Fläche mit kanonischen Texturkoordinaten
2. Übertrage diese Texturkoordinaten auf das umhüllte Objekt

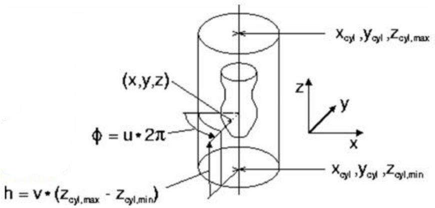
Geeignete umhüllende Flächen sind:

- Zylinder
- Kugel
- Quader

Zylinder:
Ein Zylinder entlang der z-Achse (andere Ausrichtungen durch Rotation) ist gegeben durch:

$$(x_{cyl}, y_{cyl}, z_{cyl\ min}), (x_{cyl}, y_{cyl}, z_{cyl\ max})$$

Die Oberfläche ist parametrisierbar durch den Winkel ϕ und die Höhe h.
Die Projektion eines Punktes (x, y, z) auf den Zylinder ist dann:

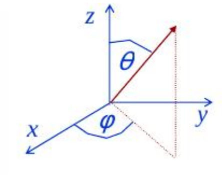


(u, v) erhält man dann durch Normalisierung von (ϕ, h).

Kugel:
Die Oberfläche einer Kugel ist parametrisierbar durch die Winkel θ und ϕ.
Die Projektion eines Punktes (x, y, z) auf die Kugel ist dann:

$$u = \frac{\pi + \arctan(y - y_s, x - x_s)}{2\pi}$$

$$v = \frac{\arctan(\sqrt{(x - x_s)^2 + (y - y_s)^2}, z - z_s)}{\pi}$$



Quader:
Meist wird eine achsenparallele Bounding Box (Quader) des Objektes verwendet. Gilt dann z.B. $(x_{max} - x_{min}) \geq (y_{max} - y_{min}) \geq (z_{max} - z_{min})$, dann ist die Projektion eines Punkt (x, y, z) :

$$u = \frac{x - x_{min}}{x_{max} - x_{min}}$$
$$v = \frac{y - y_{min}}{y_{max} - y_{min}}$$