

Computergrafik

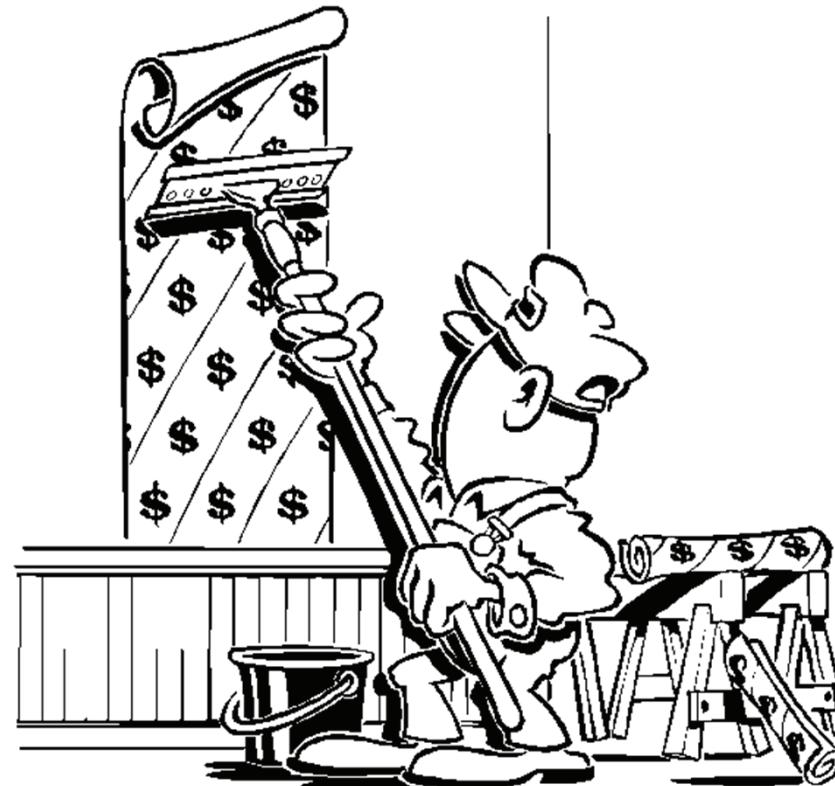
Vorlesung im Wintersemester 2014/15
Kapitel 4: Texture Mapping

Prof. Dr.-Ing. Carsten Dachsbacher
Lehrstuhl für Computergrafik
Karlsruher Institut für Technologie



Inhalt

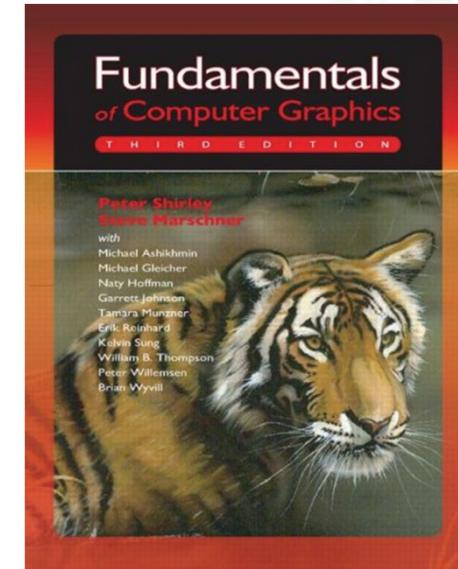
- ▶ Einführung
- ▶ Texturquellen
- ▶ Abbildung, Parametrisierung, Texturkoordinaten
- ▶ Filterung von Texturen
- ▶ Shadow Mapping
- ▶ Environment-Mapping und bildbasierte Beleuchtung



Literatur



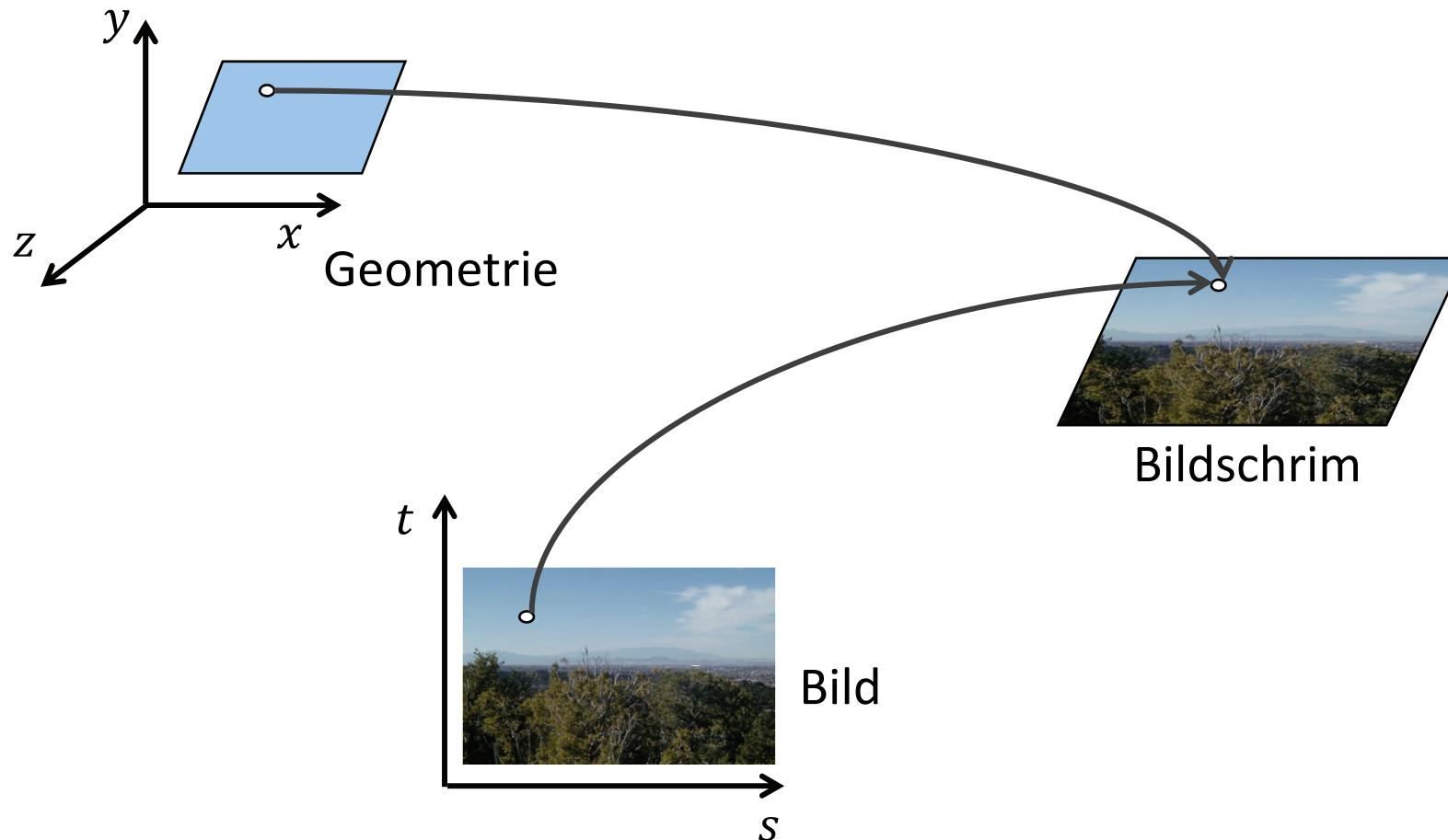
- ▶ **Fundamentals of Computer Graphics,**
P. Shirley, S. Marschner, 3rd Edition, AK Peters
→ Kapitel 9 (Signal Processing)
→ Kapitel 11 (Texture Mapping)



Was ist Texturierung/Texture Mapping?

Grundidee

- realistischeres Aussehen einer Oberfläche kann man durch „Feinstrukturierung“ pro Pixel erreichen
- kombiniere Geometrie mit Bildern



Texturen und Oberflächendetail

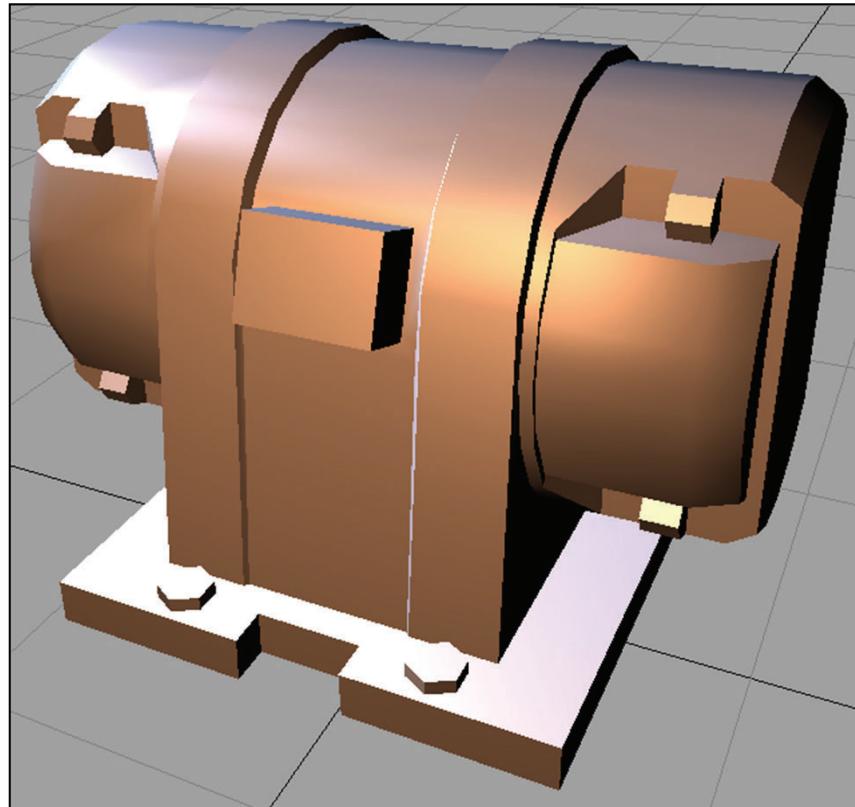


Texturierungstechniken

Was können wir mit den (Farb-)Werten aus Texturen tun?

- Bsp. Kontrolle der Parameter des Phong-Beleuchtungsmodell (blau eingefärbt):

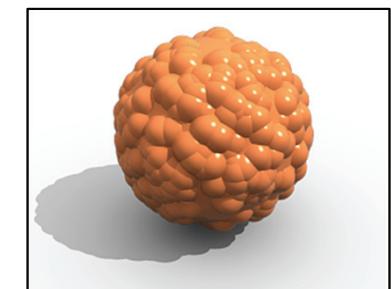
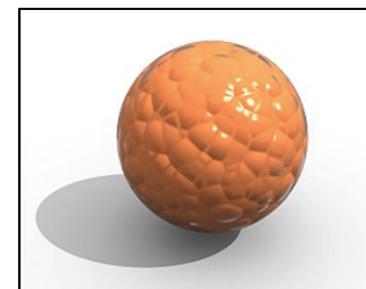
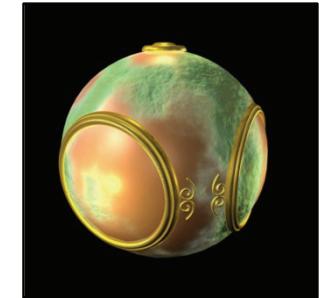
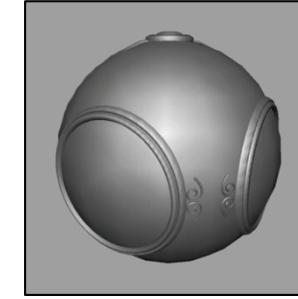
$$I = k_a \cdot I_L + k_d \cdot I_L \cdot (\mathbf{N} \cdot \mathbf{L}) + k_s \cdot I_L \cdot (\mathbf{R} \cdot \mathbf{V})^n$$



Texturen: eine von vielen Mapping-Arten



- ▶ Reflexionseigenschaften: [Demo](#)
 - ▶ Farbe, div. Koeffizienten, Transparenz
 - ▶ klassisches „Texture-Mapping“
- ▶ Normalenvektoren
 - ▶ Veränderung der Normalen für die Beleuchtungsberechnung
 - ▶ „Bump-“ oder „Normal-Mapping“
- ▶ Beleuchtung: [Demo](#)
 - ▶ „Environment-Mapping“, „Reflection-Mapping“
 - ▶ „Shadow-Mapping“, „Light-Mapping“
- ▶ Geometrie: [Demo](#)
 - ▶ verschiebe Flächen
 - ▶ „Displacement-Mapping“



Mapping: Beispiele



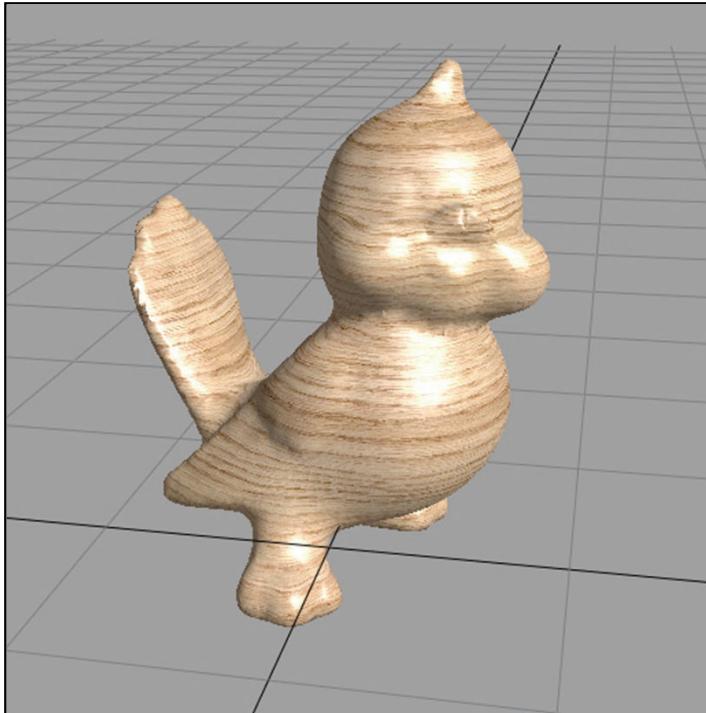
- ▶ Hat man eine Abbildung von Oberfläche zu Textur, dann kann man diese Parametrisierung oft noch anderweitig nutzen!
- ▶ Bsp. Beleuchtung und Streuung des Lichts unter der Haut im 2D-Texturraum approximieren



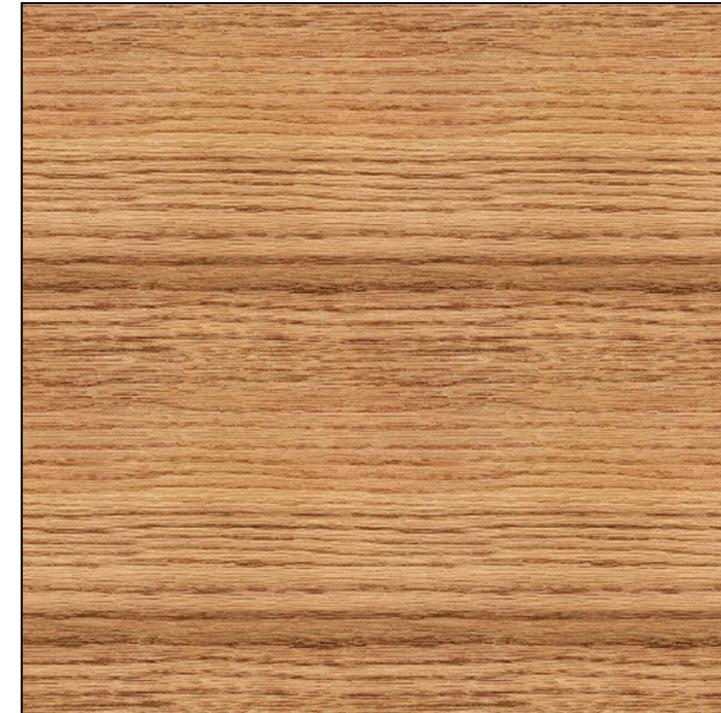
Matrix: Reloaded, Sketch SIGGRAPH 2003

2D Texture Mapping

Wie wird einem Punkt auf der Oberfläche eine Stelle in der Textur zugeordnet?



Position \mathbf{x} , Normale \mathbf{n}_x ,
Position in Kamerakoordinaten \mathbf{x}_{cam} ,
Reflexionsvektor \mathbf{r}_x , ...



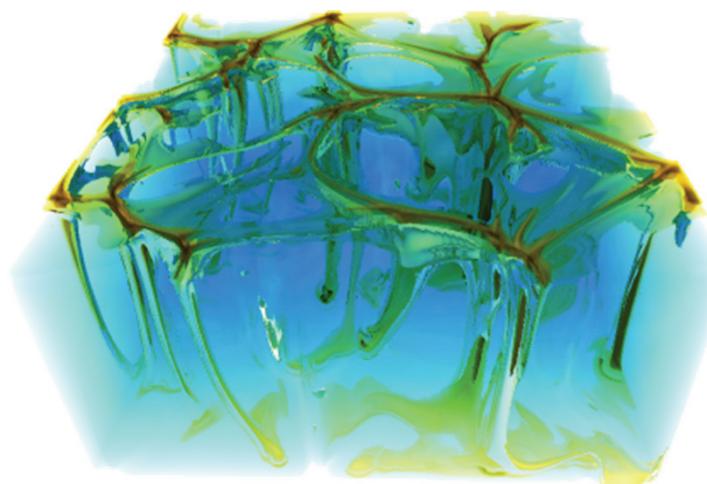
Texturkoordinate (s, t)
(manchmal auch mit (u, v)
bezeichnet)



Texturen aus Bildern (Fotos, Simulationen, Videos, ...)

- ▶ Texturen sind (meist) Rasterbilder bestehend aus „Texels“ (Texture Elements, vgl. Pixel)
- ▶ oft einfache Erstellung (1D Texturen, Farbtabellen) bzw. einfache Akquisition (2D Texturen, „eingefrorener Beleuchtung“)
- ▶ hoher Speicherbedarf bei vielen, großen Texturen (→Texturkompression)

- ▶ 3D Textur aus einer Simulation:

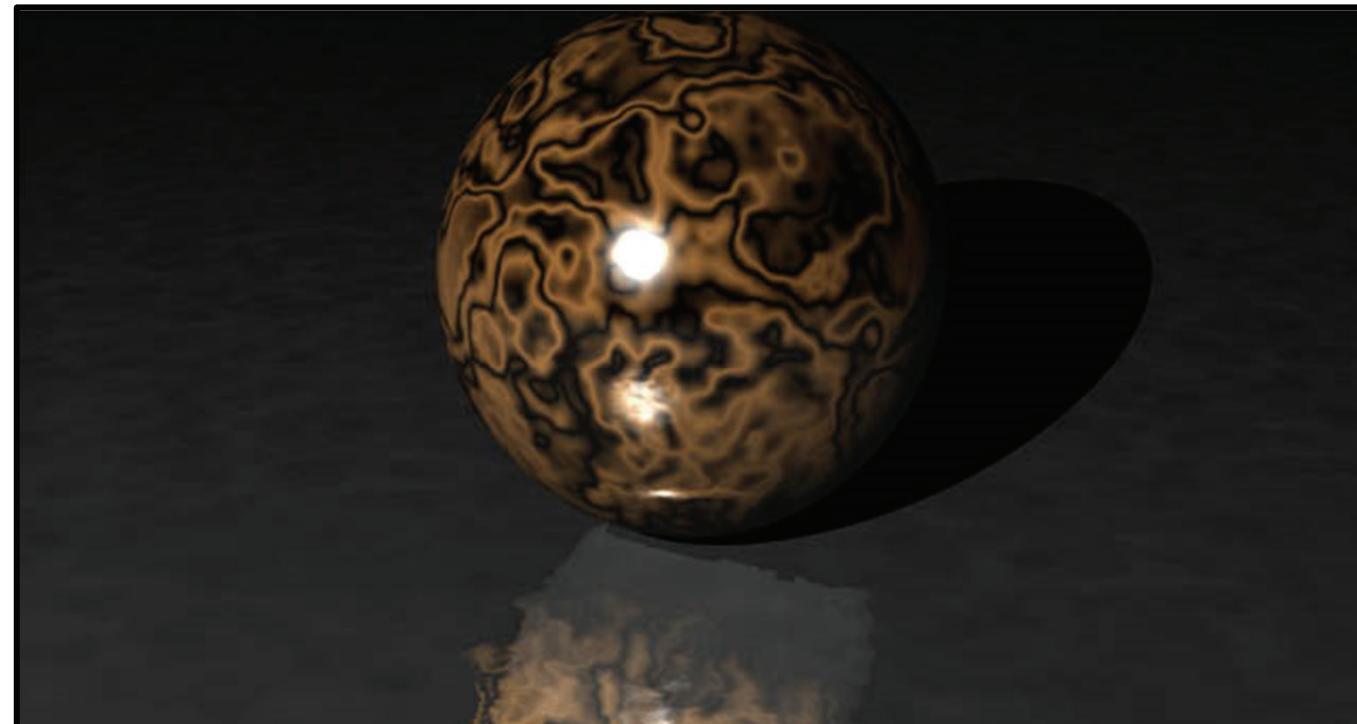


Textur-Quellen – Ausblick auf Kapitel 6



Prozedurale Texturen durch Algorithmen (Shader)

- ▶ Auswertung zur Laufzeit → geringer Speicherbedarf
- ▶ im Prinzip unbeschränkte Auflösung, optimale Genauigkeit
- ▶ (manchmal) nicht-triviale Programmierung
- ▶ große Auswahl an Algorithmen für Holz, Stein/Marmor, Wolken, ...
(i.A. natürliche, „stochastische“ Texturen)



Textur-Quellen – Aktuelle Forschungsthemen



Vektorgrafik und Diffusion für Texturen

- ▶ ebenfalls Auswertung zur Laufzeit → wenig Speicherbedarf
- ▶ im Prinzip unbeschränkte Auflösung
- ▶ nicht-triviale Programmierung und Filterung

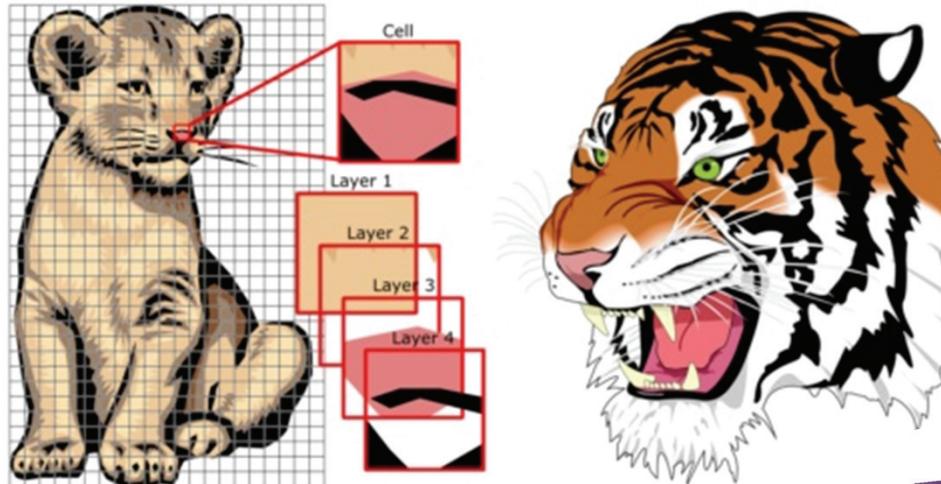


Bild: Random-access rendering of general vector graphics,
D. Nehab und H. Hoppe



Rendering Surface Details with
Diffusion Curves, Jeschke et al.

Mapping von Texturen

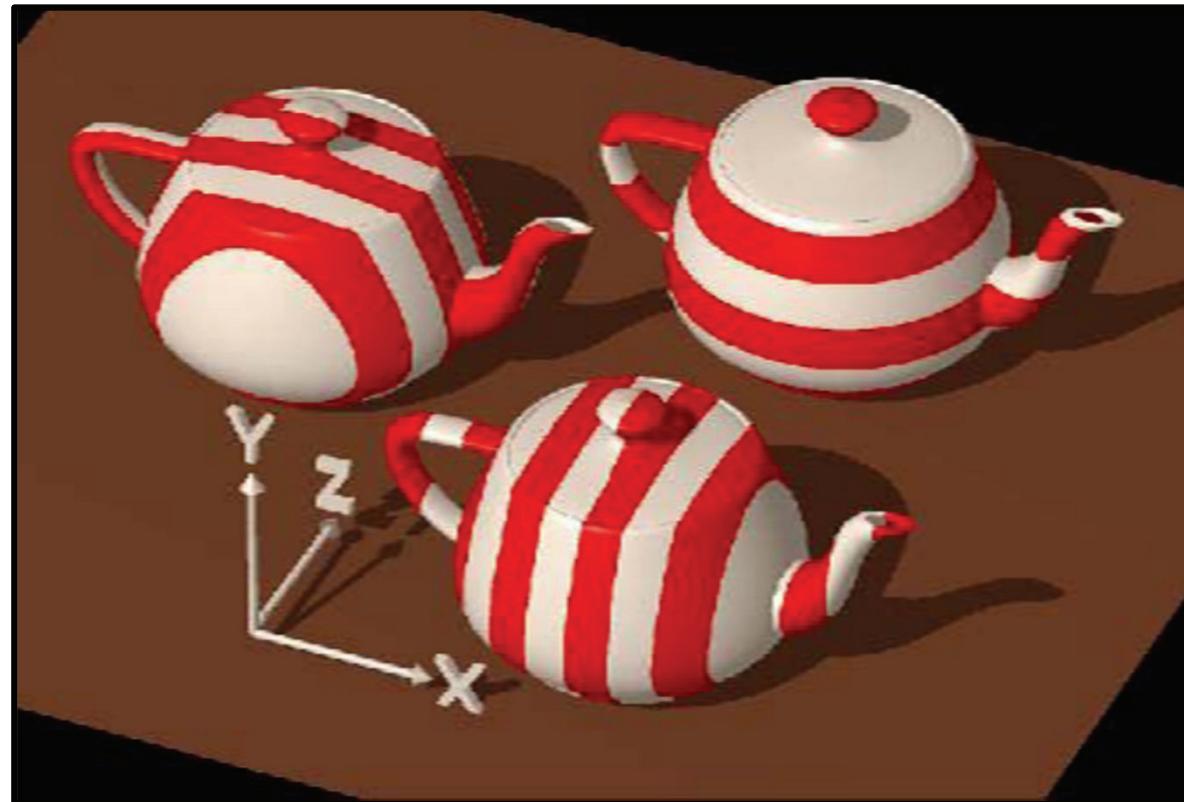


- ▶ wie kommen 1D/2D-Texturen auf Objekte?
 - ▶ oder: welche Möglichkeiten der Parametrisierung gibt es?
 - ▶ Körper mit natürliche Parametrisierung (Ebene, Kugel, Zylinder, ...)
 - ▶ Abbildung über Hilfskörper
 - ▶ viele weitere Parametrisierungen bei speziellen Anwendungen

- ▶ wie speichert man die Parametrisierung?
 - ▶ spezifizieren der Berechnungsvorschrift
 - ▶ explizites Speichern von Texturkoordinaten

1D Textur

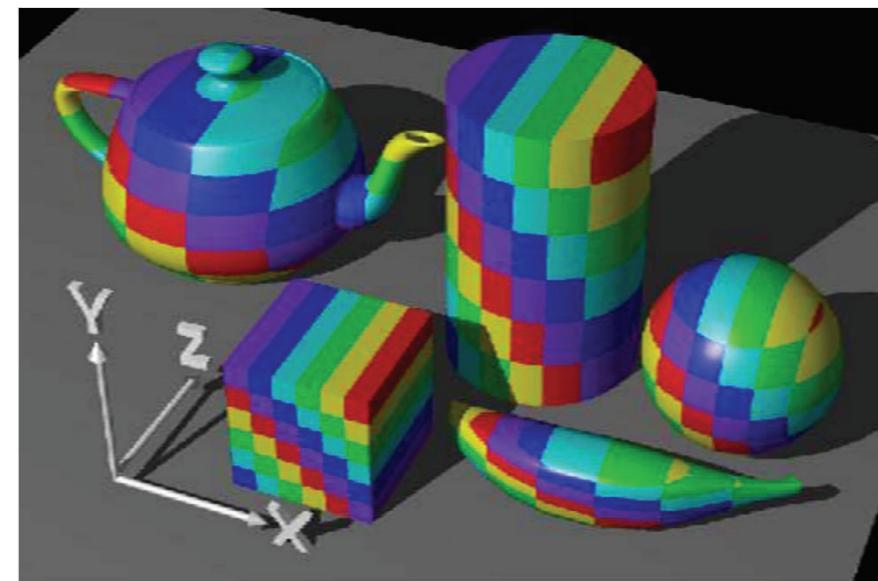
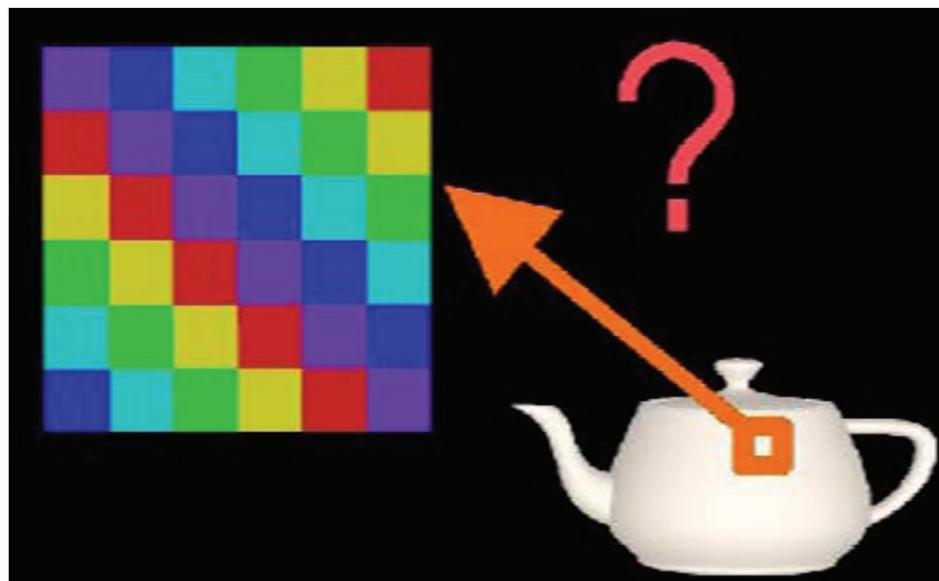
- ▶ Parameter einer Linie, Höhe, Temperatur, ...
 - ▶ hier: Parameter entlang einer der drei Achsen
 - ▶ z.B. Texturkoordinate $s = (\mathbf{x} - \mathbf{p}) \cdot \mathbf{s}$



2D Textur

Planare Projektion

- ▶ Bsp.: Projektion eines Bildes in der X-Y Ebene
- ▶ planare Projektion allgemein:
definiere Ebene, z.B. durch einen Punkt p und
zwei aufspannende Vektoren s, t
- ▶ Texturkoordinate eines Oberflächenpunkts x
 $s = (x - p) \cdot s$
 $t = (x - p) \cdot t$

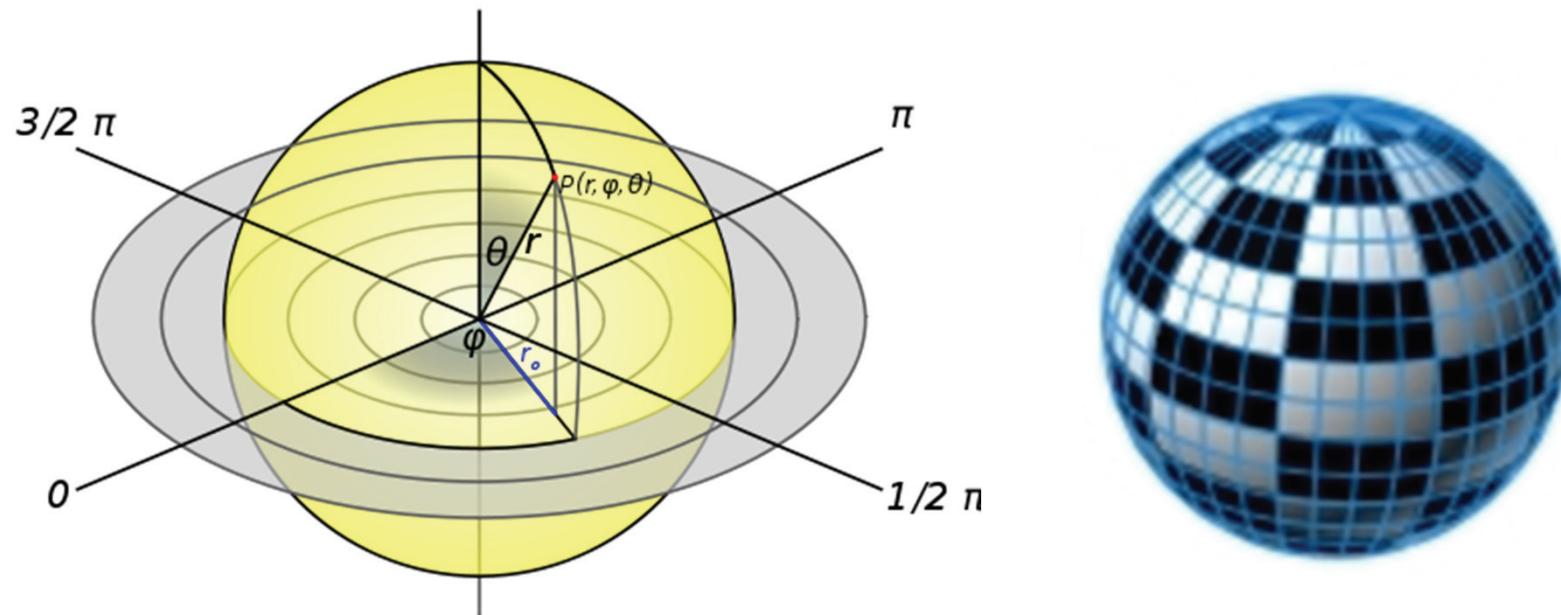


Mapping von 2D Texturen

Standardkörper mit natürlicher Parametrisierung

► Beispiel Kugel:

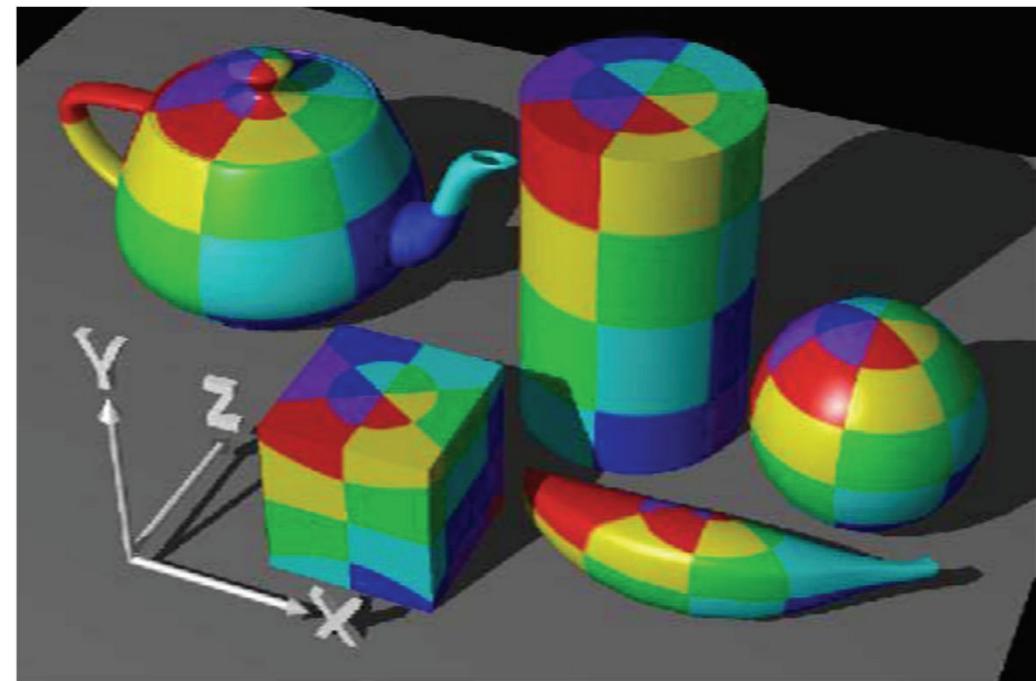
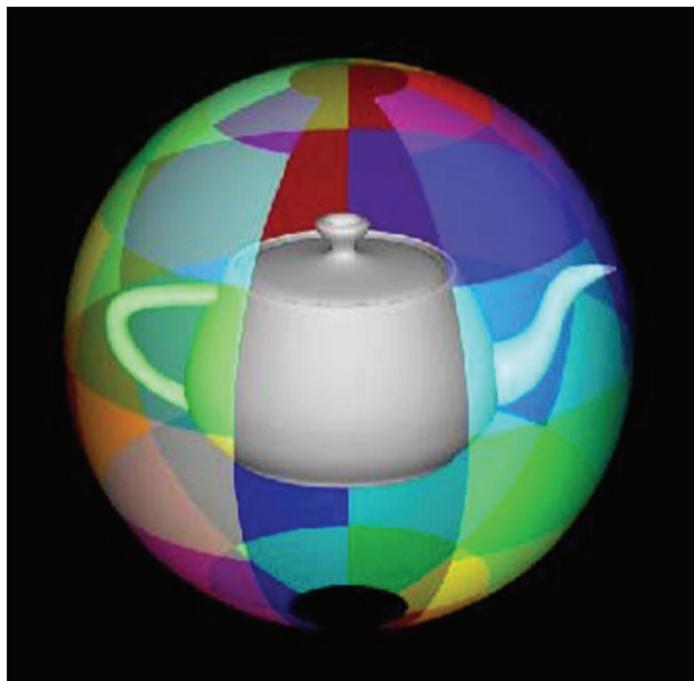
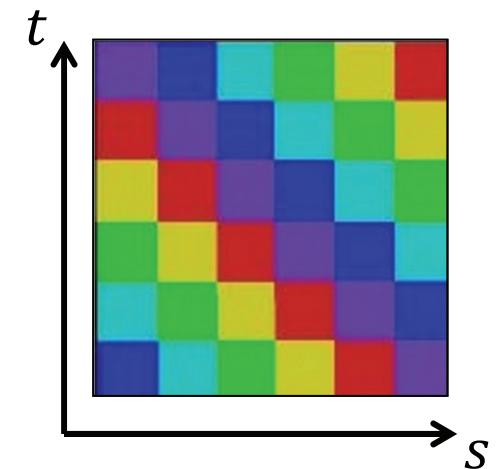
- ▶ Punkte auf der Oberfläche lassen sich mit Polarkoordinaten (r, ϕ, θ) ausdrücken
- ▶ 2 Winkel mit denen auf eine 2D-Textur zugegriffen wird



2D Textur

Sphärische Parametrisierung

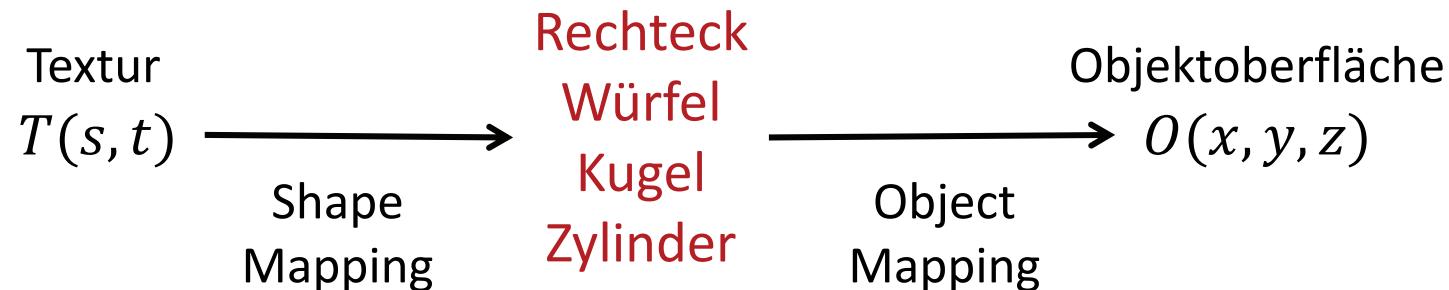
- ▶ Darstellung der Objektkoordinaten in Polarkoordinaten (r, ϕ, θ)
- ▶ Texturkoordinaten $\begin{pmatrix} s \\ t \end{pmatrix} := \begin{pmatrix} \phi / 2\pi \\ \theta / \pi \end{pmatrix}$
- ▶ der Bereich $[0,1]^2$ repräsentiert die ganze Textur
→ Unabhängigkeit von der tatsächlichen Auflösung



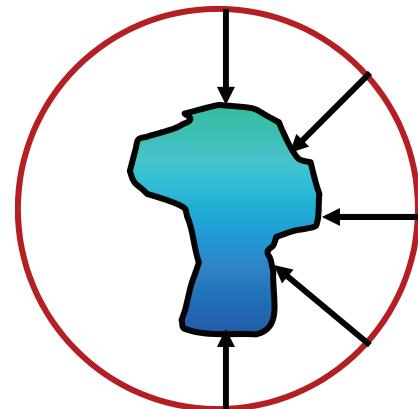
Zweiphasen-Mapping

Texturierung beliebiger Objekte

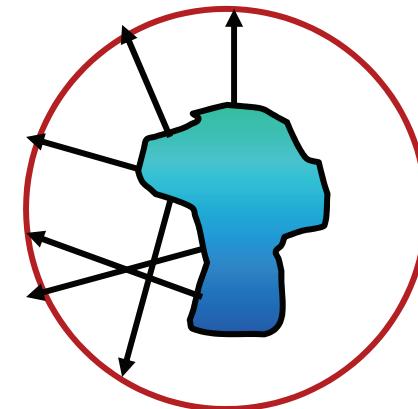
- verwende Standardkörper als Zwischenschritt bzw. Hilfsfläche



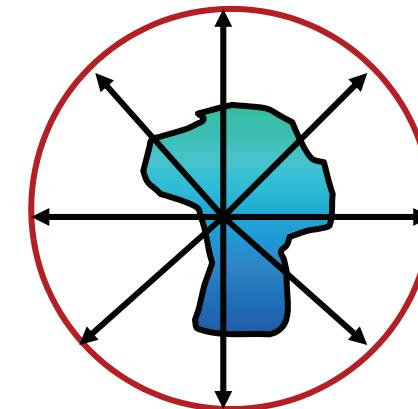
- mögliche „Object Mappings“, also zw. Objektpunkt und Standardkörper:



Normale der Hilfsfläche
→ Oberfläche



Normale der Oberfläche
→ Hilfsfläche

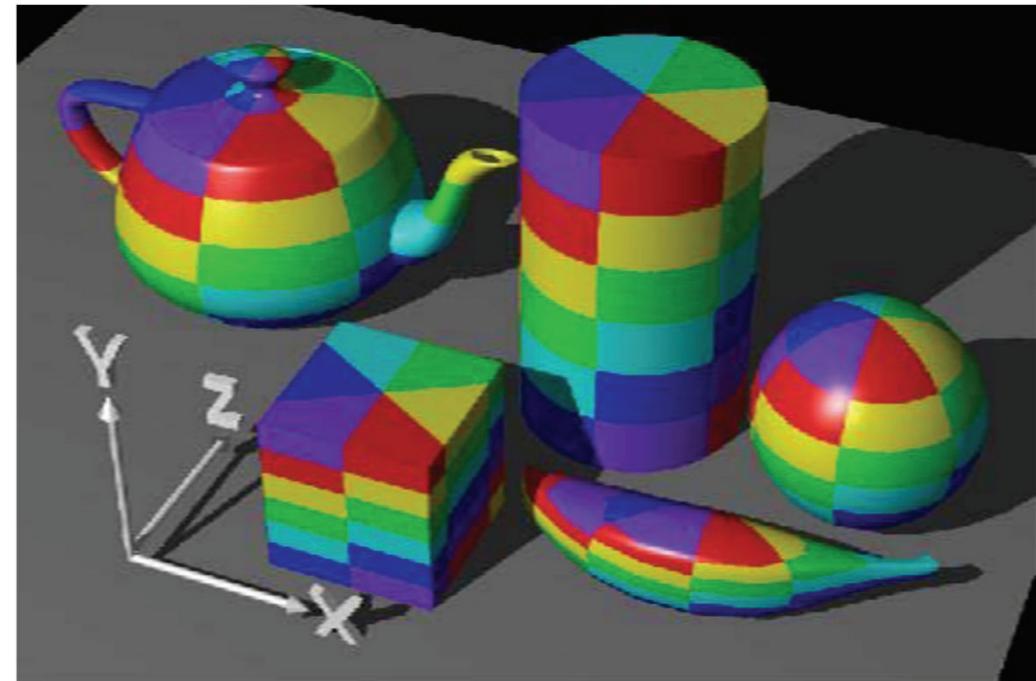
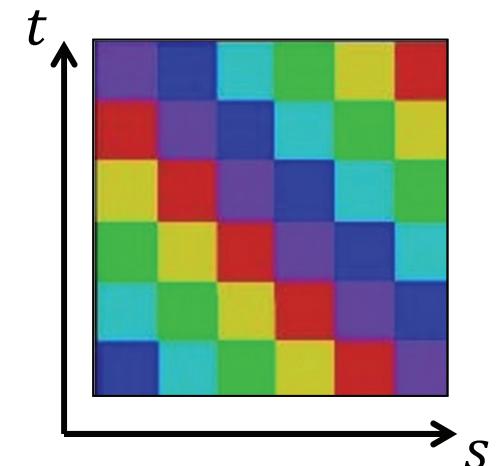


Linie durch Mittelpunkt

2D Textur

Zylindrische Parametrisierung

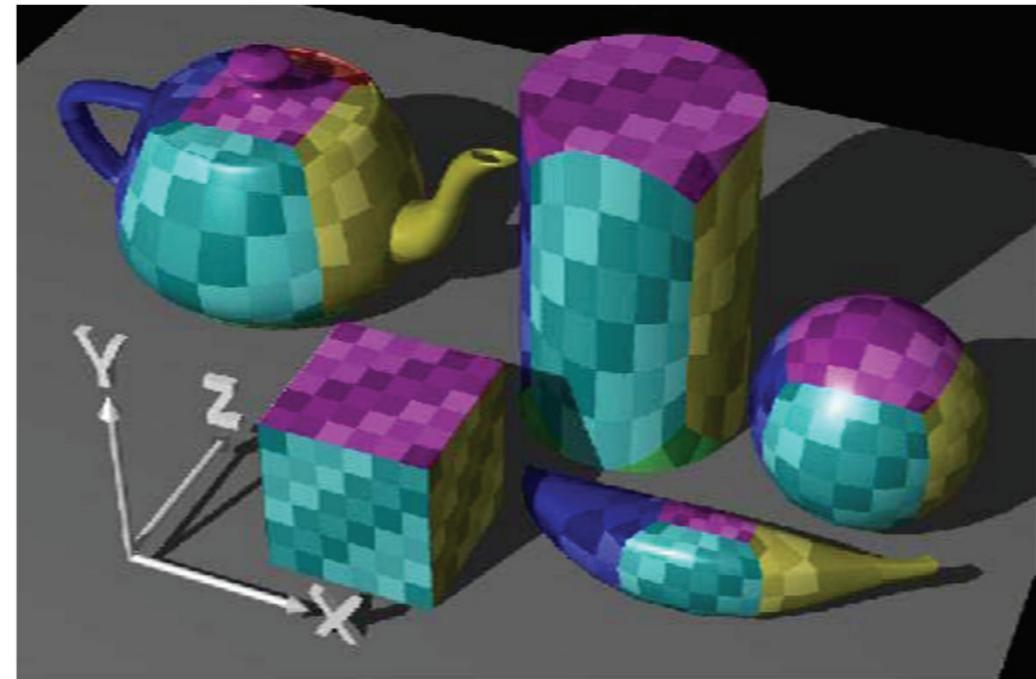
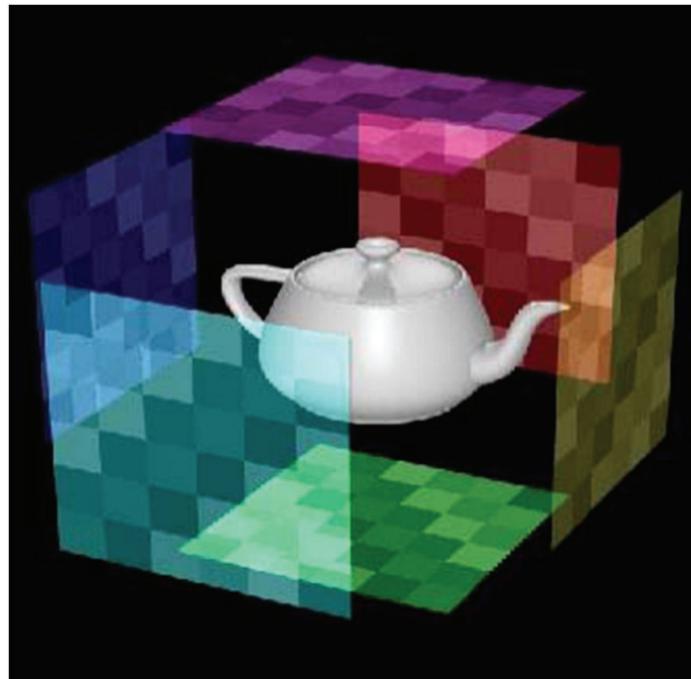
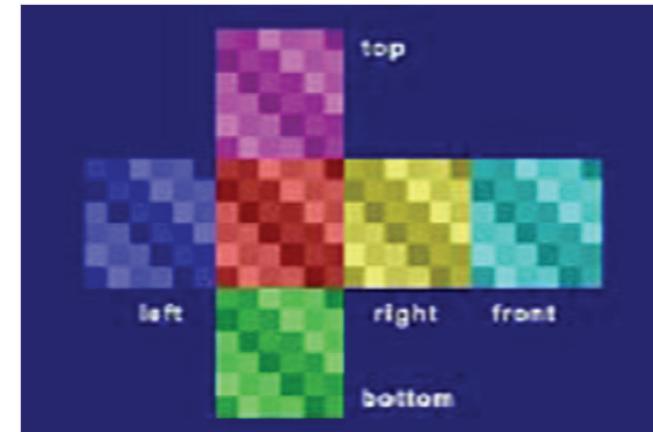
- ▶ Darstellung der Objektkoordinaten in Zylinderkoordinaten (r, ϕ, y)
- ▶ Texturkoordinaten $\begin{pmatrix} s \\ t \end{pmatrix} := \begin{pmatrix} \phi / 2\pi \\ y/h \end{pmatrix}$



2D Textur

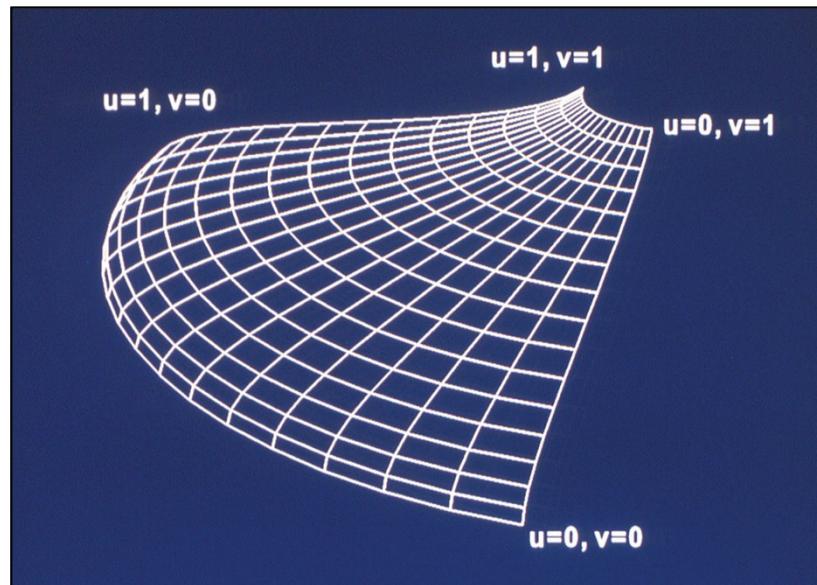
Würfel Parametrisierung

- ▶ Abbildung: lese die Würfeltextur dort aus, wo ein Strahl vom Objektmittelpunkt durch einen Oberflächenpunkt einen umgebenden Würfel schneidet
- ▶ andere Abbildungen Objekt-Hilfskörper natürlich ebenfalls möglich



Flächenparameter als Texturkoordinate

- ▶ Beispiel parametrische Fläche: $\mathbf{x}(u, v)$ (z.B. sog. Bézier-/Spline-Patches)
 - ▶ Zugriff auf Textur mit $(s, t) := (u, v)$



Flächenparameter als Texturkoordinate



- der „Utah Teapot“ ist aus parametrischen Flächen zusammengesetzt



2D und 3D Texturierung

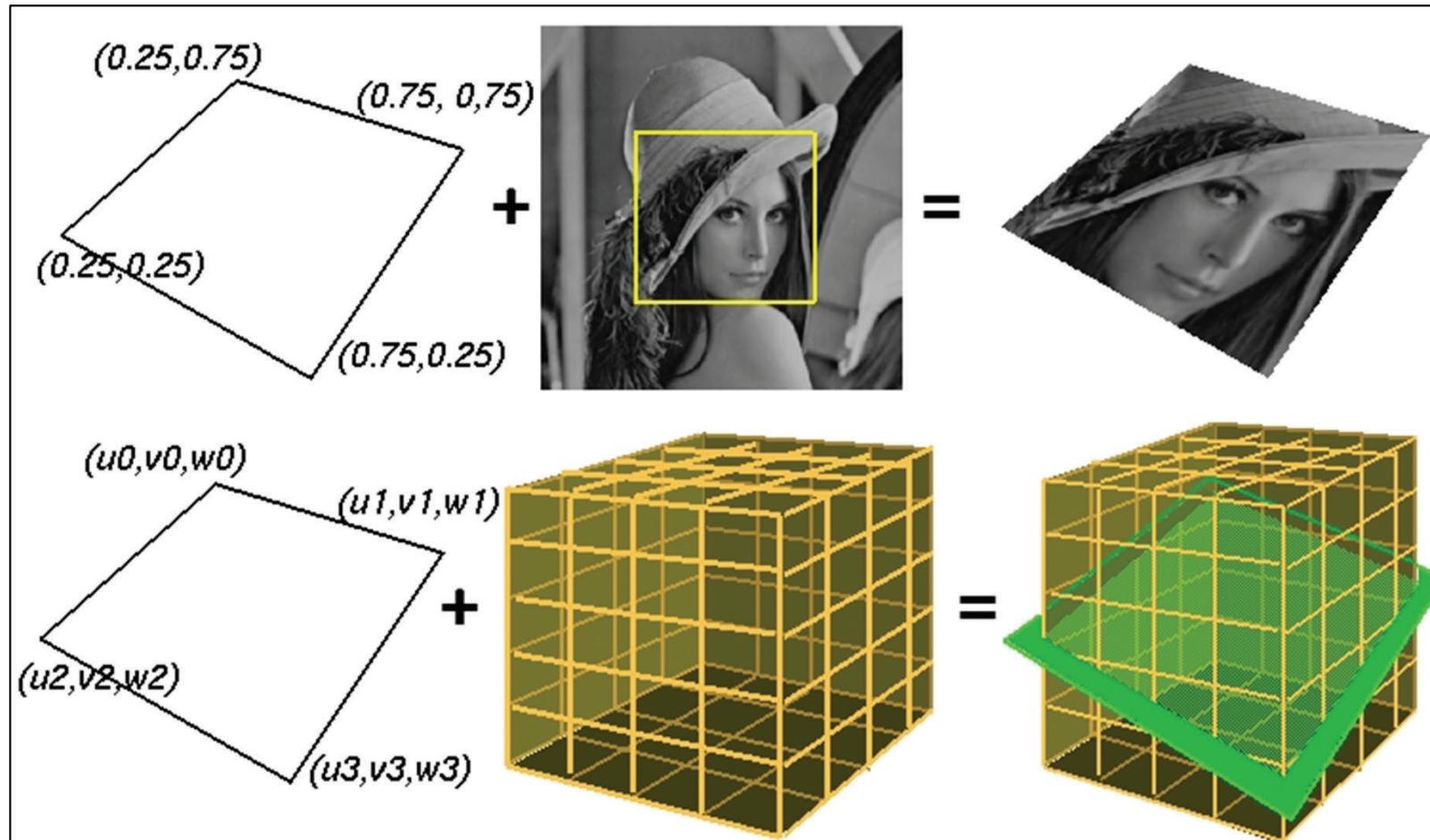


2D: Texture Mapping

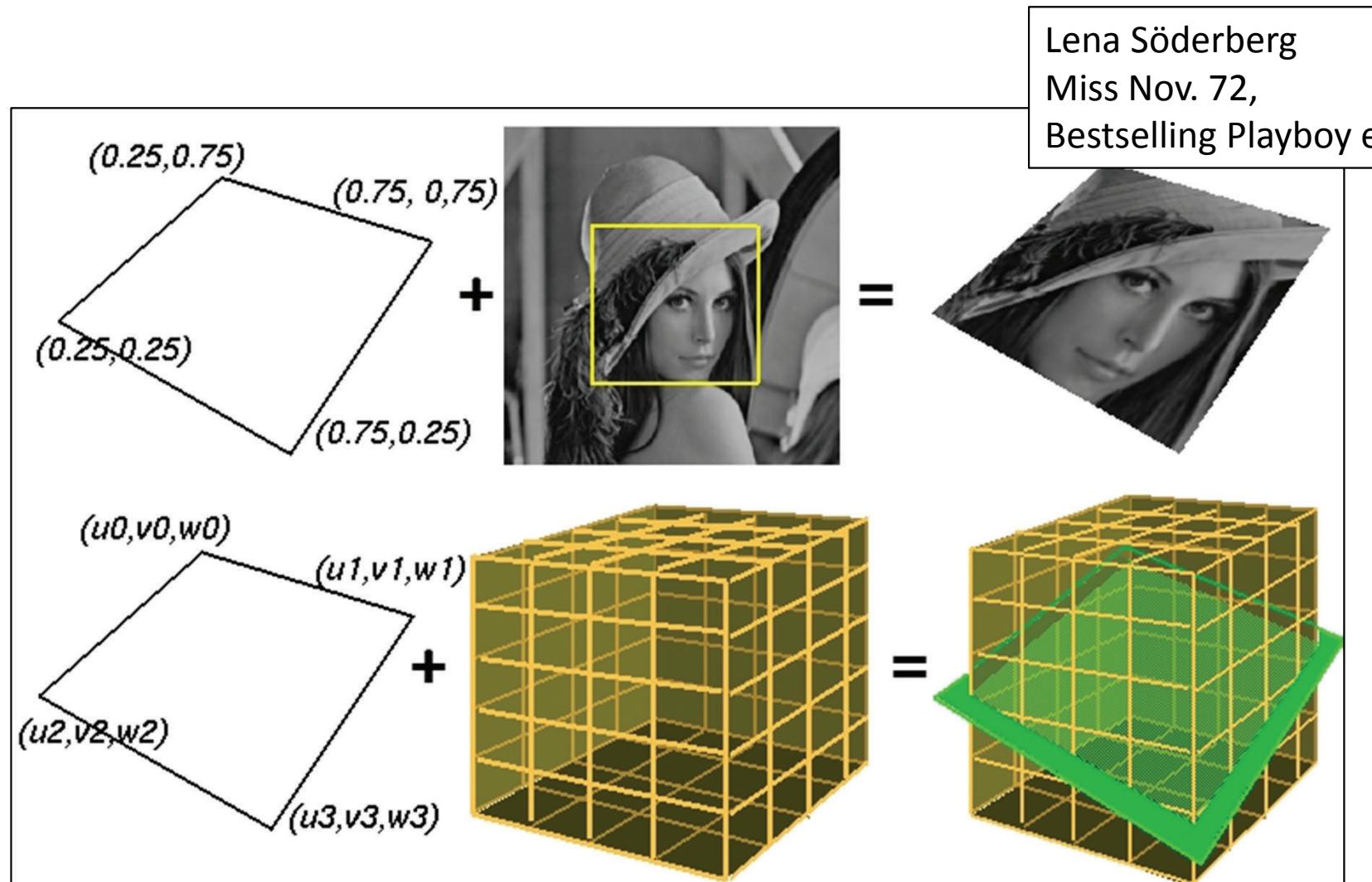


3D: Solid Texturing

Texturkoordinaten in 2D und 3D

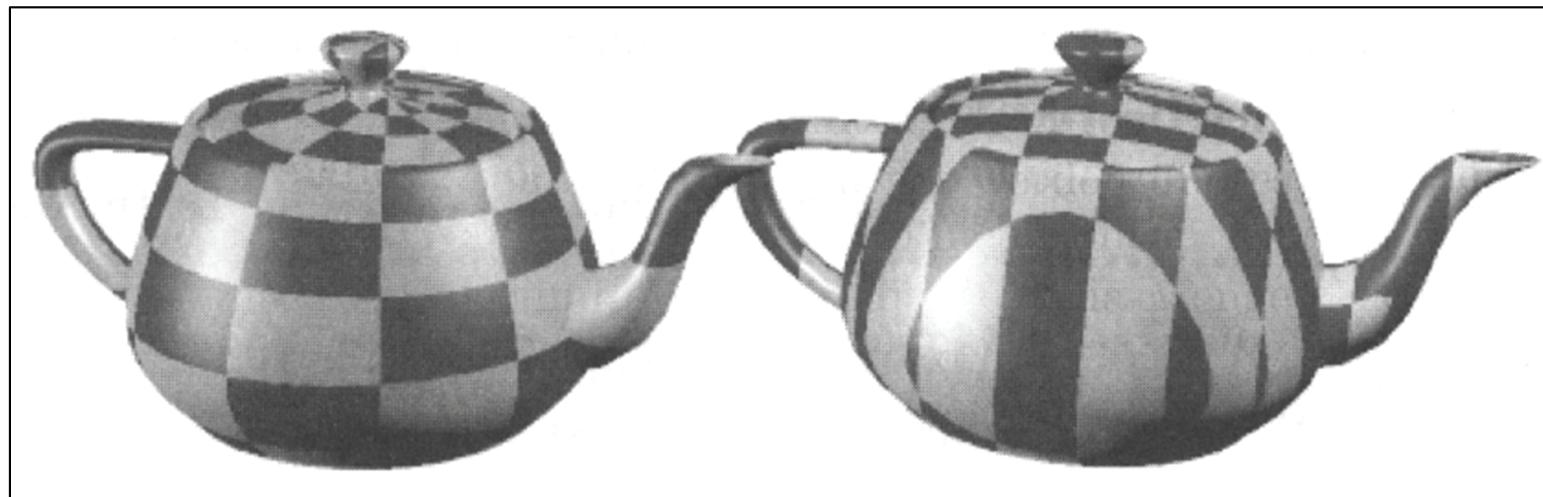


Texturkoordinaten in 2D und 3D



Texturabbildung

- ▶ Projektion auf andere Flächen (Kugel, Zylinder, Box, Ebene, ...)
- ▶ Flächenparameter (u, v) (z.B. Bézier-Patch)
- ▶ Bildschirm- bzw. Kamerakoordinaten (x_s, y_s)
- ▶ Objekt- (x_o, y_o, z_o) oder Weltkoordinaten (x_w, y_w, z_w)
- ▶ Richtungsvektoren (**R, N, H, ...**)
- ▶ Funktion der obigen Parameter (z.B. in „Shadern“=pro Pixel Berechnung)
- ▶ →Texturkoordinaten (s, t, r) (manchmal auch (u, v, \dots))



Mapping von Texturen



- ▶ wie kommen 1D/2D-Texturen auf Objekte?
 - ▶ oder: welche Möglichkeiten der Parametrisierung gibt es?
 - ▶ Körper mit natürliche Parametrisierung (Ebene kennen wir schon)
 - ▶ Abbildung über Hilfskörper
 - ▶ viele weitere Parametrisierungen bei speziellen Anwendungen

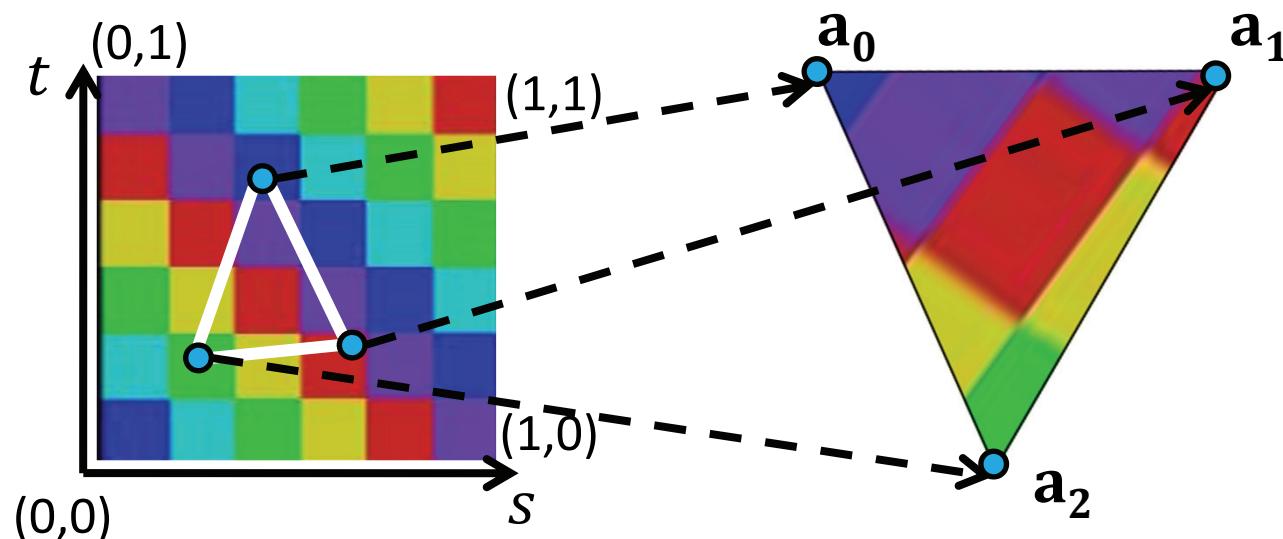
- ▶ wie speichert man die Parametrisierung?
 - ▶ spezifizieren der Berechnungsvorschrift
 - ▶ explizites Speichern von Texturkoordinaten

Texturkoordinaten für Dreiecksnetze



Explizites Speichern der Parametrisierung

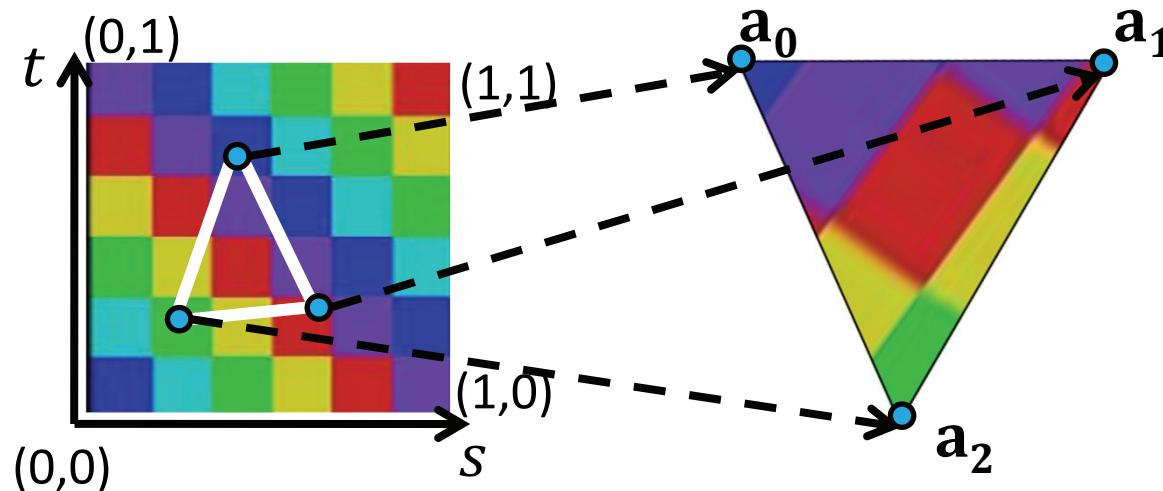
- ▶ bisher: Vorgabe der Berechnungsvorschrift, die die Texturkoordinate für einen Punkt auf der Oberfläche bestimmt
- ▶ bei Dreiecksnetzen geht man üblicherweise anders vor:
 - ▶ Parametrisierung wird in **Texturkoordinaten** gespeichert
 - ▶ jedem Eckpunkt/Vertex $\mathbf{a}_i = (x_i, y_i, z_i)$ eines Dreiecks wird eine Texturkoordinate (s_i, t_i) zugewiesen



Texturkoordinaten für Dreiecksnetze

Texturkoordinaten

- ▶ Parametrisierung in Texturkoordinaten gespeichert, indem jedem Eckpunkt $\mathbf{a}_i = (x_i, y_i, z_i)$ eine Texturkoordinate (s_i, t_i) zugewiesen wird

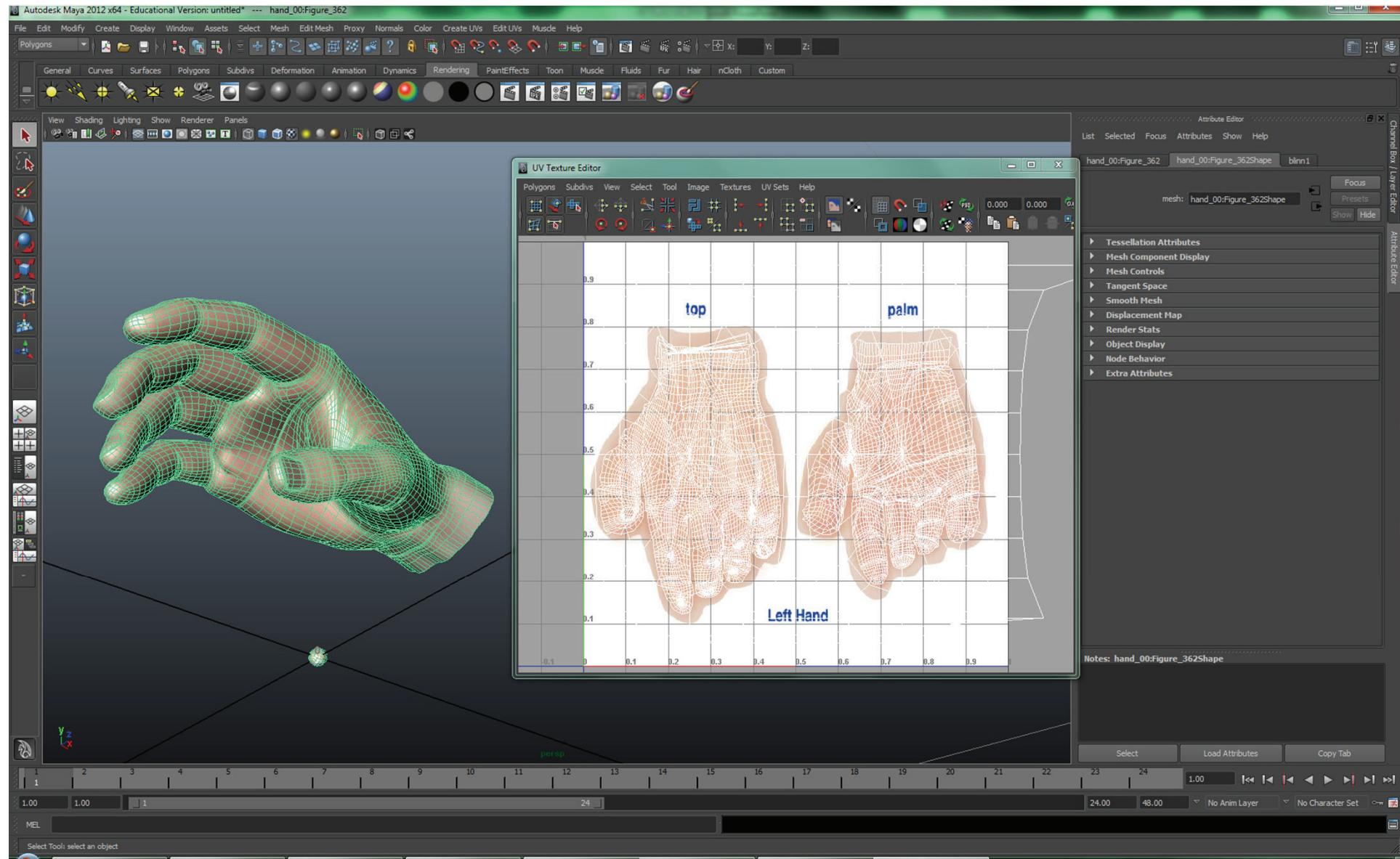


- ▶ Texturkoordinaten können
 - ▶ ...für jeden Vertex wie bisher berechnet werden
 - ▶ ...können bei Triangulierung parametrischer Flächen erzeugt werden
 - ▶ ...werden oft auch manuell festgelegt

Texturkoordinaten für Dreiecksnetze



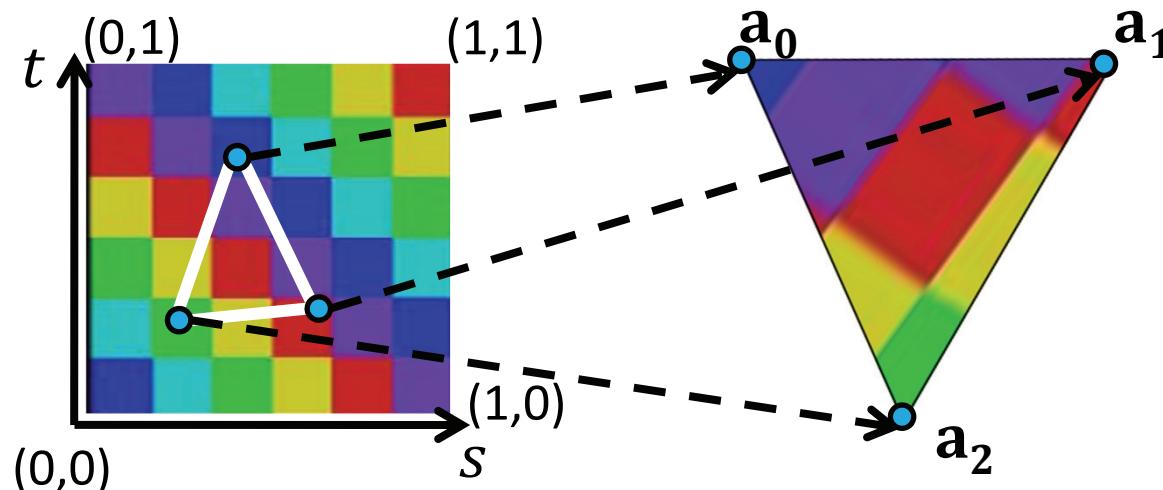
Texturkoordinaten



Texturkoordinaten für Dreiecksnetze

Texturkoordinaten

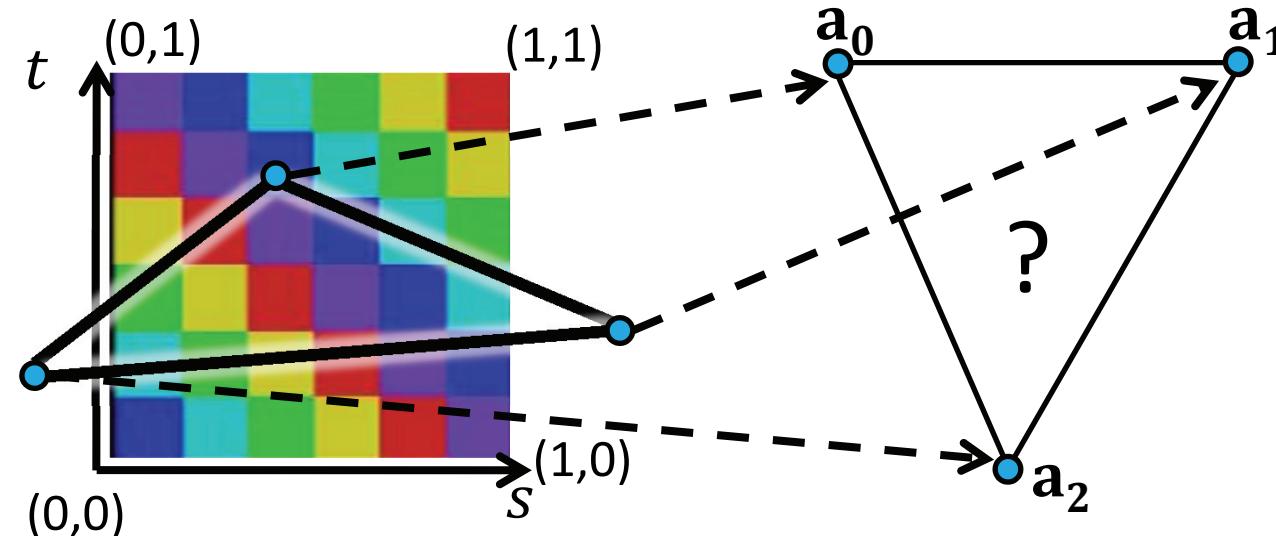
- ▶ Parametrisierung in Texturkoordinaten gespeichert, indem jedem Eckpunkt $\mathbf{a}_i = (x_i, y_i, z_i)$ eine Texturkoordinate (s_i, t_i) zugewiesen wird



- ▶ affine Abbildung zwischen 2D Texturraum und 3D Objektraum
 - ▶ Interpolation mit baryzentrischen Koord. (analog zu Farbe/Normale)
 - ▶ → stückweise lineare Approximation einer Parametrisierung!
(es sei denn die Parametrisierung ist selbst schon linear)

Texturkoordinaten

- ▶ was passiert bei Texturkoordinaten > 1.0 oder < 0.0 ?

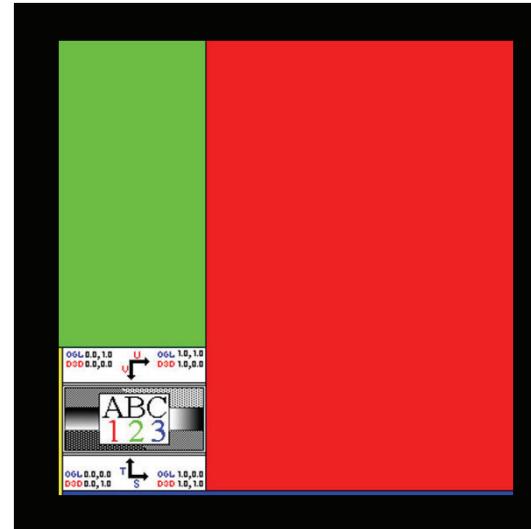


- ▶ es gibt eine Reihe verschiedener „Adressierungsmodi“, die in der CG verwendet werden (und von Grafik-Hardware unterstützt werden)

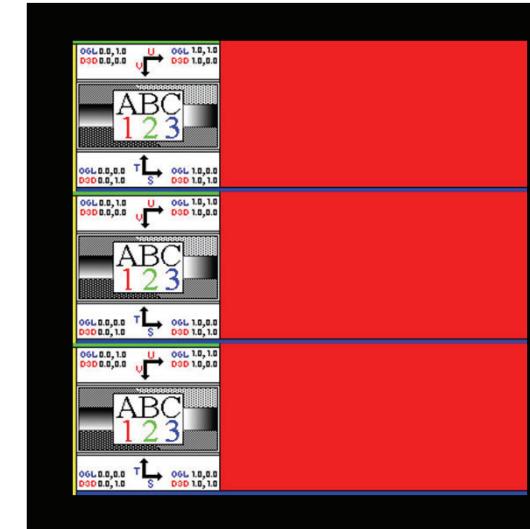
Texture Wrapping

- ▶ Repeat/Wrapping: Fortsetzen einer Textur über $[0,1]^2$ hinaus
- ▶ Adressierung wird für jede Dimension separat gewählt

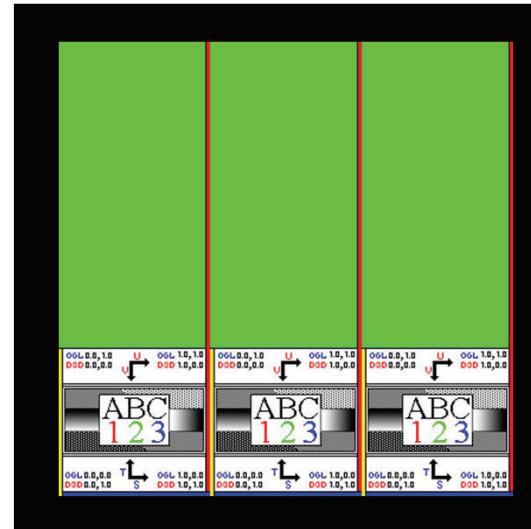
clamp/
clamp



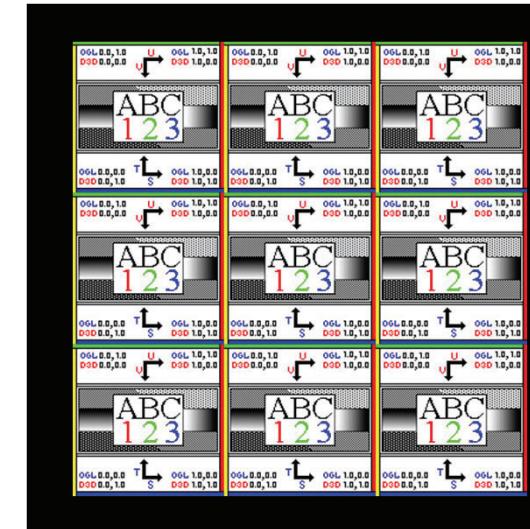
clamp/
repeat



repeat/
clamp

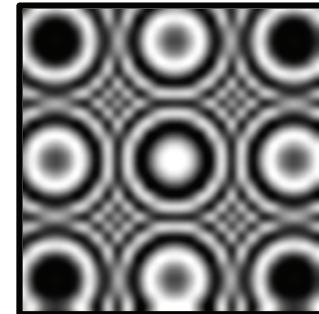
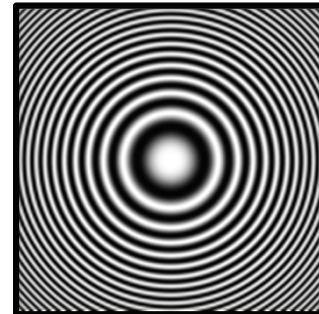


repeat/
repeat

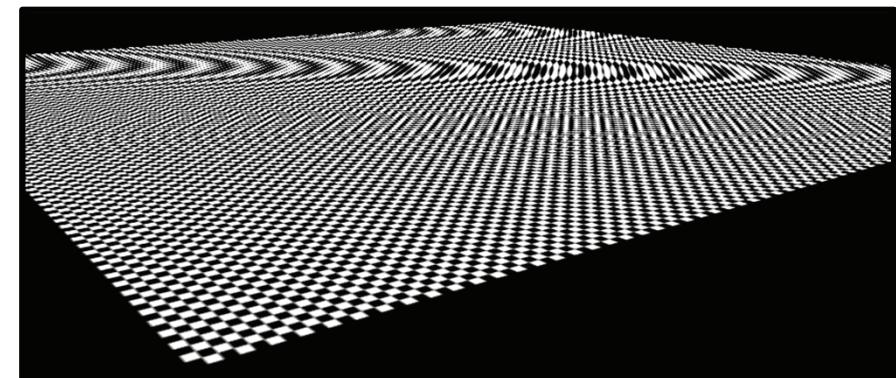
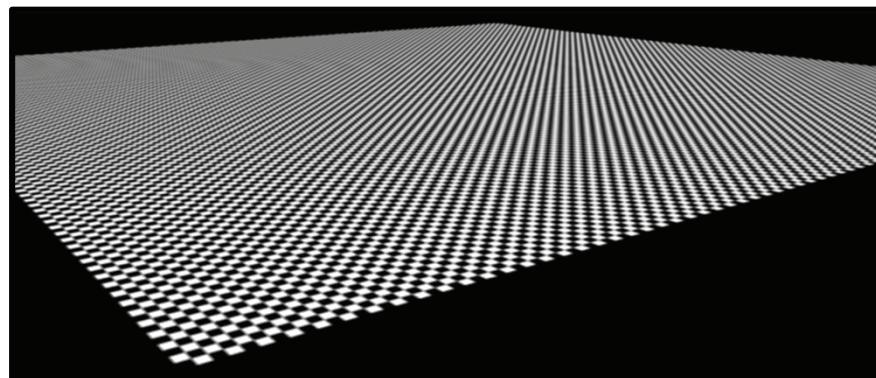


Texture Mapping und Aliasing

- ▶ wir kennen Aliasing schon von der Abtastung von Signalen
 - ▶ hier (Unter-)Abtastung einer kontinuierlichen Funktion:

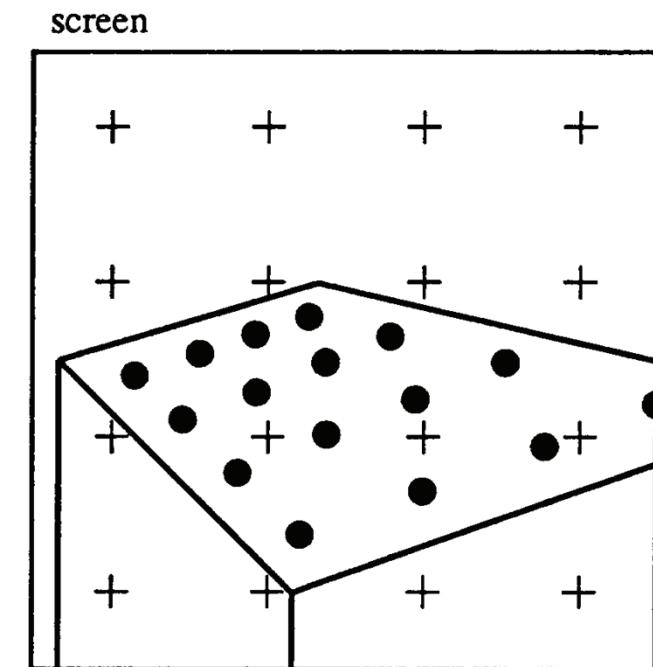
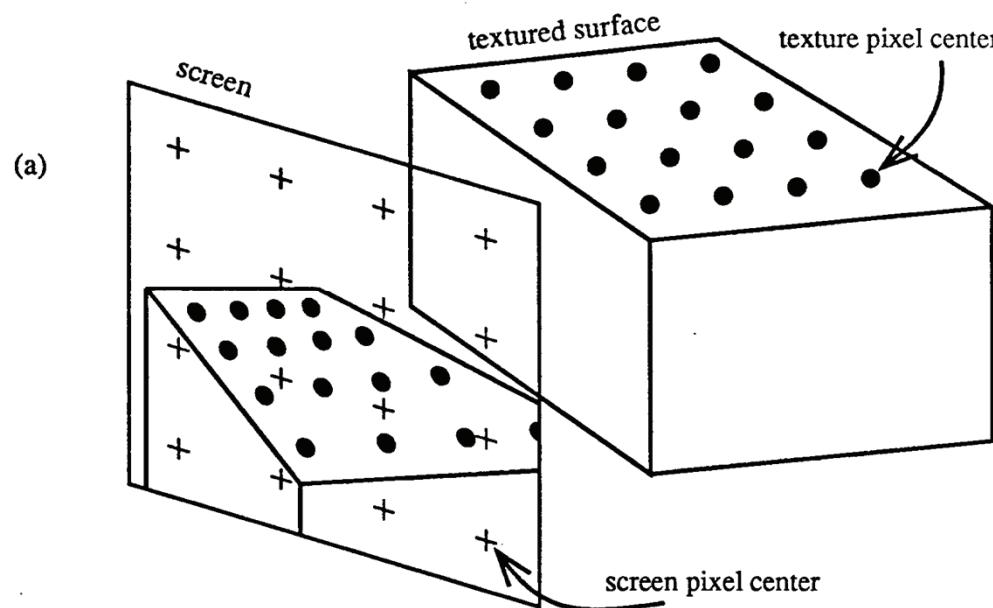


- ▶ das gleiche Problem tritt beim Texture Mapping auf
 - ▶ Textur = (meist diskrete) Funktion, abgetastet bei der Bildzeugung



Texture Mapping und Aliasing

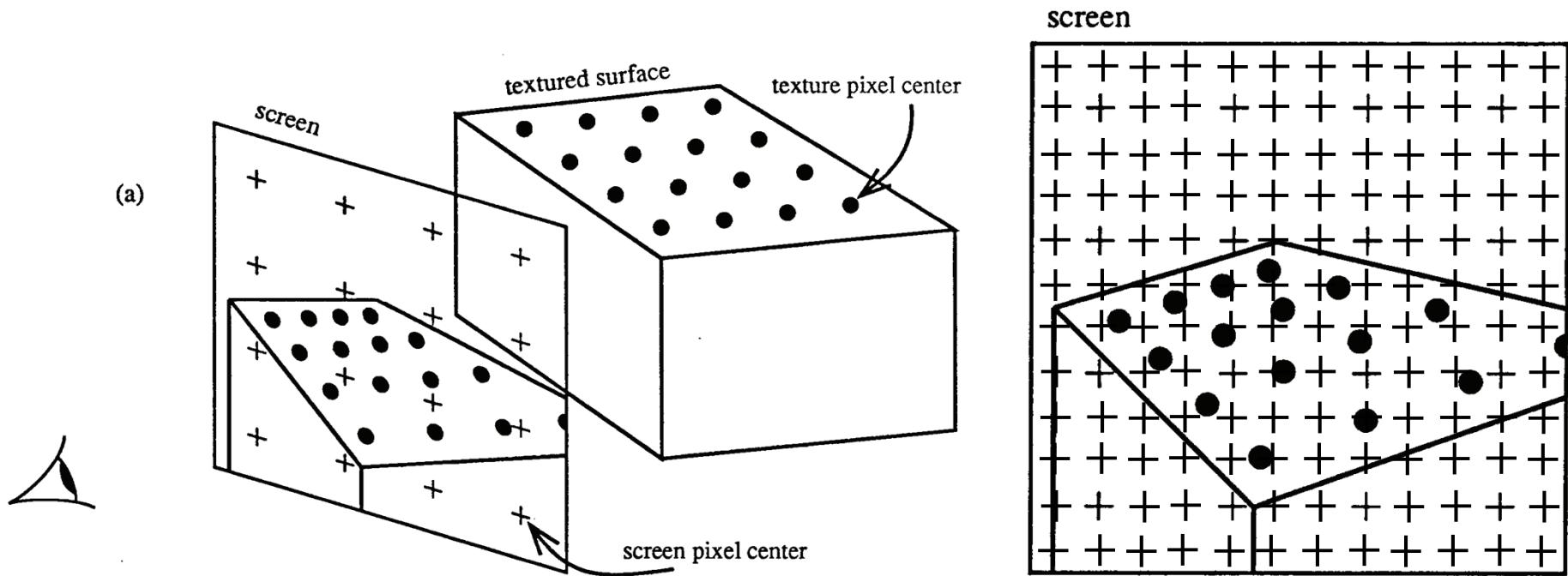
- ▶ Abbildungen des Textursignals
 - ▶ 2D Texturraum → 3D Objektraum: Parametrisierung
 - ▶ 3D Objektraum → 2D Bildraum: Projektion



Fundamentals of Texture Mapping and Image Warping,
Paul Heckbert, Master's thesis, UCB/CSD 89/516,
CS Division, U.C. Berkeley, June 1989

Texture Mapping und Aliasing

- ▶ Abbildungen des Textursignals
 - ▶ 2D Texturraum → 3D Objektraum: Parametrisierung
 - ▶ 3D Objektraum → 2D Bildraum: Projektion
- ▶ entscheidend ist das Verhältnis der Abtastfrequenz am Bildschirm zur Auflösung des projizierten Signals/der projizierten Textur

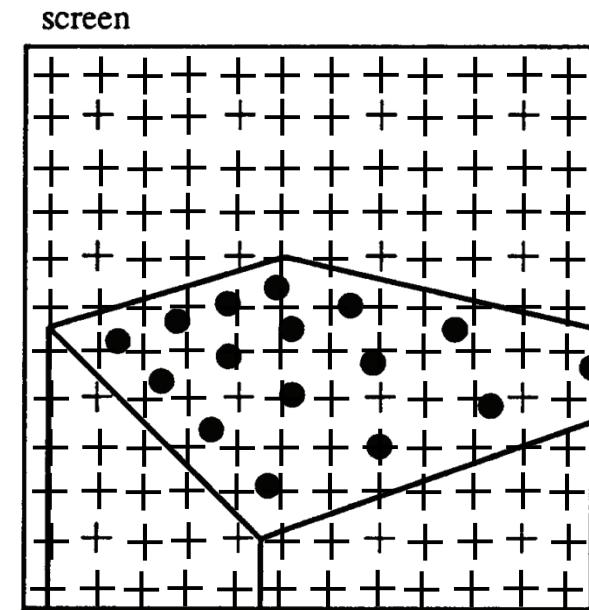


Fundamentals of Texture Mapping and Image Warping,
Paul Heckbert, Master's thesis, UCB/CSD 89/516,
CS Division, U.C. Berkeley, June 1986

Textur-Filterung

Vergrößerung (Magnification)

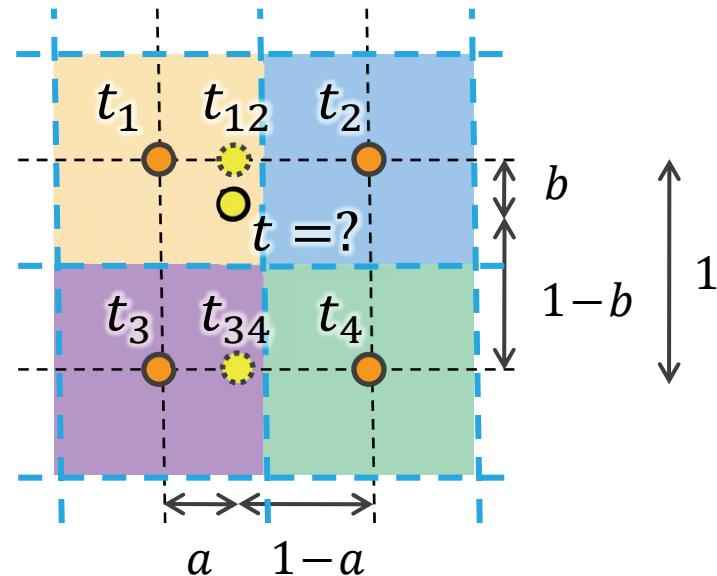
- ▶ Abbildung weniger Texel auf viele Pixel
- ▶ **Nearest Neighbor:**
 - ▶ verwende Farbe des nächstliegenden Texels
- ▶ **Bilineare Interpolation:**
 - ▶ Interpolation der 4 nächsten Texel
 - ▶ Nutzung der „Nachkommastellen“
 - ▶ dadurch wird das Textursignal geglättet:



Vergrößerung/Magnification

Bilineare Interpolation

- ▶ Interpolation für den Punkt ●, aus den Nachbartexeln ●
- ▶ $a, (1 - a), b$ und $(1 - b)$ sind die relativen Abstände zu den Texelmittelpunkten entlang der s - und t -Achsen

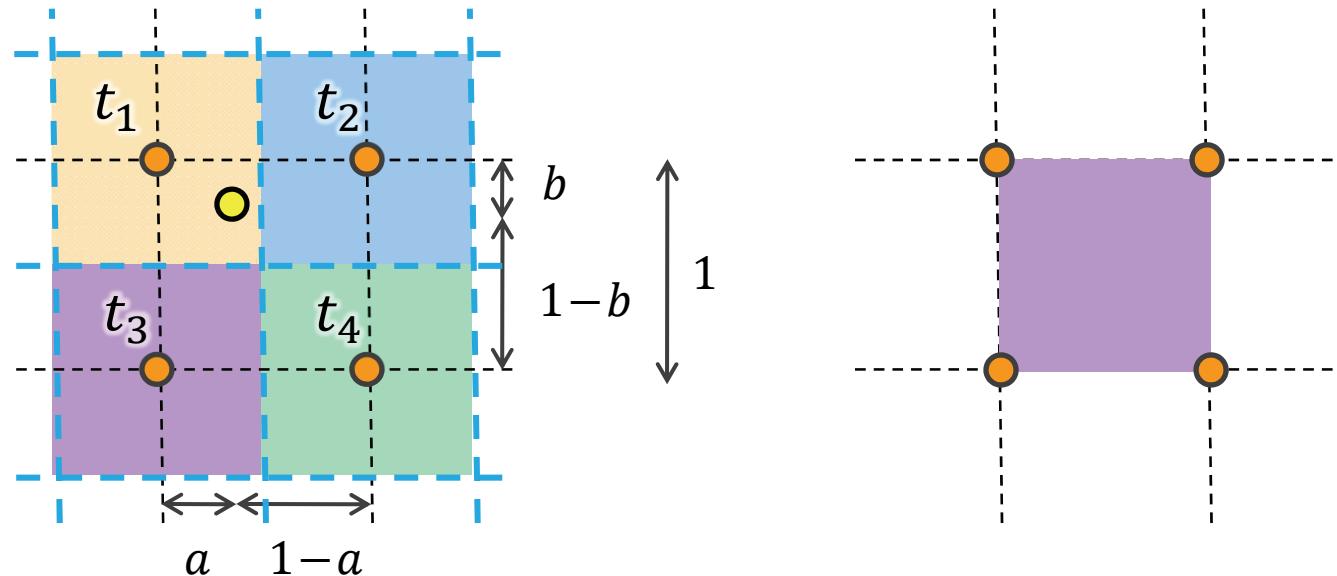


- ▶ zweimal lineare Interpolation: horizontal, dann vertikal (oder umgekehrt)
 - ▶ $t_{12} = t_1 + a(t_2 - t_1) = (1 - a)t_1 + at_2$
 - ▶ $t_{34} = (1 - a)t_3 + at_4$
 - ▶ $t = (1 - b)t_{12} + bt_{34}$

Vergrößerung/Magnification

Bilineare Interpolation

- ▶ Interpolation für den Punkt ●, aus den Nachbartexeln ●
- ▶ $a, (1 - a), b$ und $(1 - b)$ sind die relativen Abstände zu den Texelmittelpunkten entlang der s - und t -Achsen

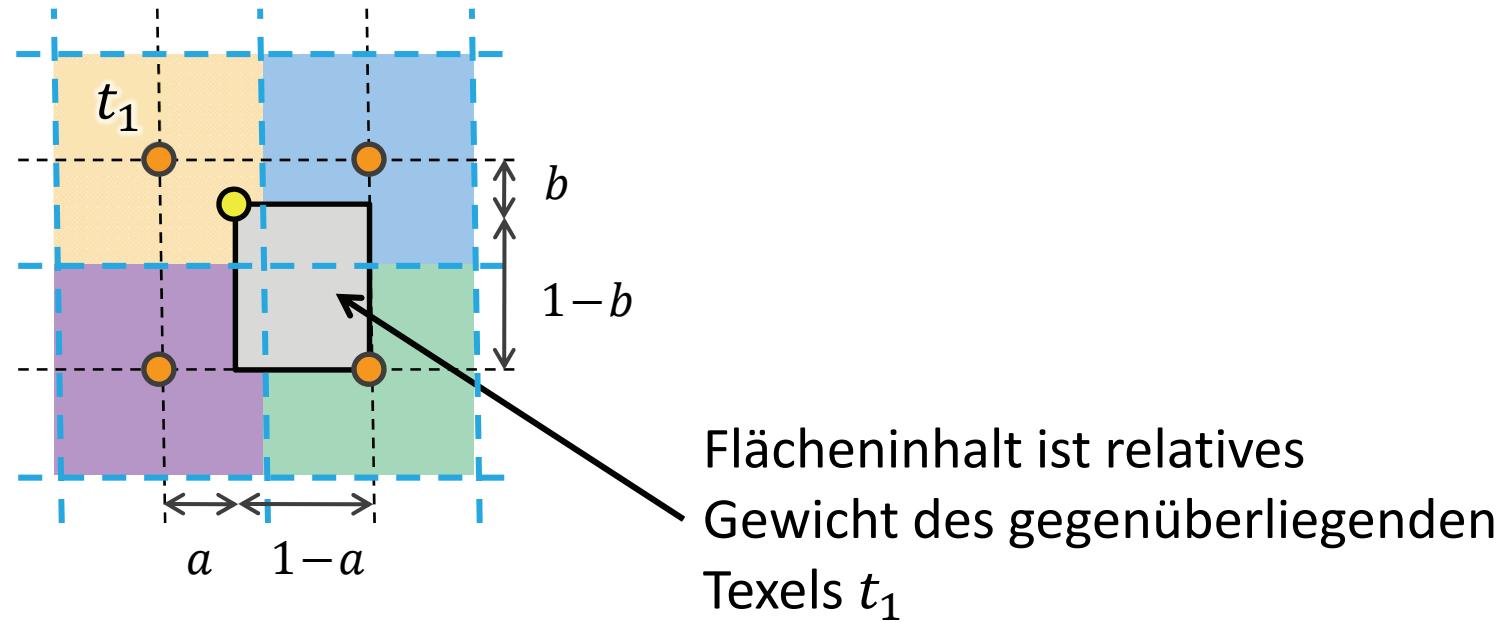


- ▶ Farbe =
$$\begin{matrix} \text{■} & (1-a)(1-b) + \end{matrix} \begin{matrix} \text{■} & a(1-b) + \\ \text{■} & (1-a)b + \end{matrix} \begin{matrix} \text{■} & ab \end{matrix}$$

Vergrößerung/Magnification

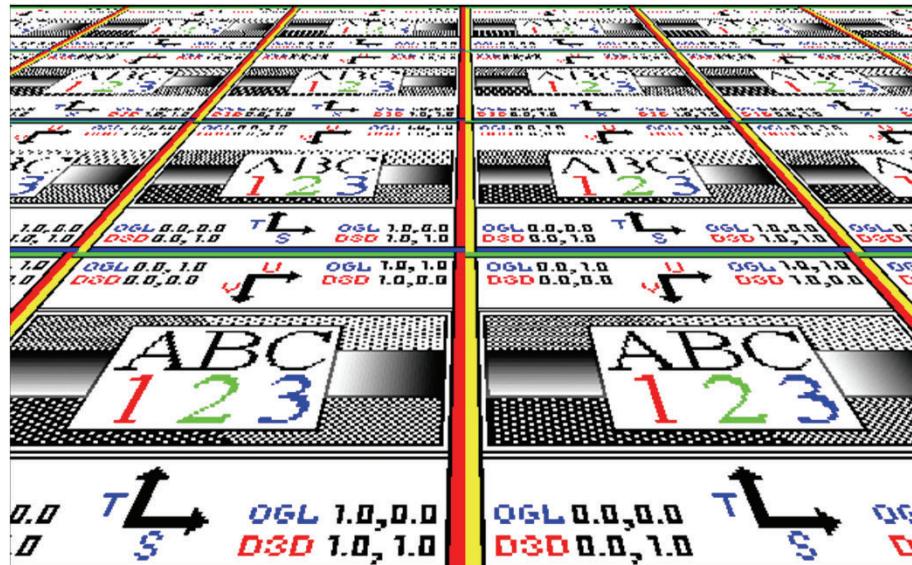
Bilineare Interpolation

- ▶ Interpolation für den Punkt ●, aus den Nachbartexeln ●
- ▶ alternative Interpretation (vgl. baryzentrische Koordinaten im Dreieck)

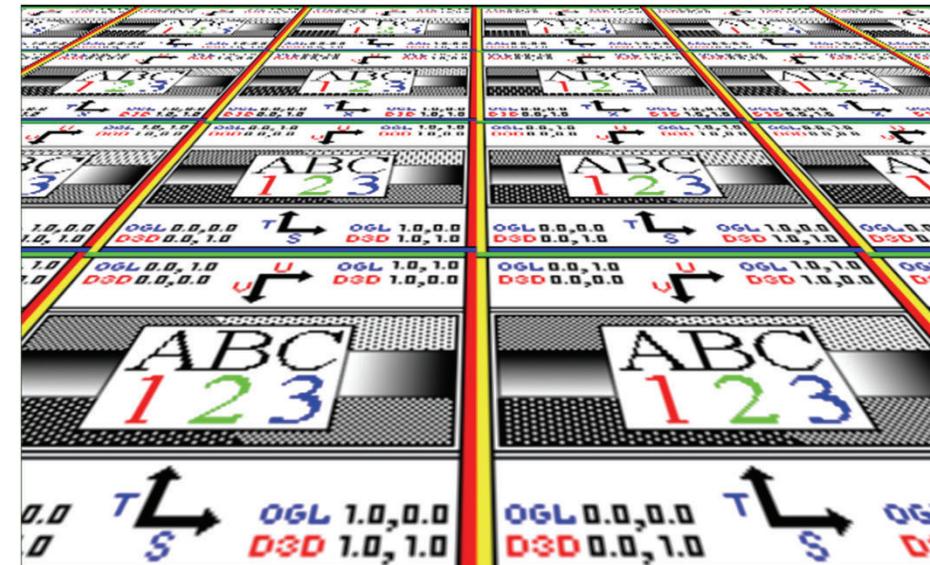


▶ Farbe =  $(1-a)(1-b) + \quad$  $a(1-b) +$
 $(1-a)b + \quad$  ab

Vergrößerung/Magnification



Nearest Neighbor



bilineare Interpolation

Textur-Filterung

Texturfilter höherer Ordnung

- ▶ menschliche Wahrnehmung nimmt Unstetigkeiten in der „Krümmung“ wahr, deshalb verwendet man manchmal Interpolation höherer Ordnung
- ▶ bilineare Interpolation (links), bikubische Interpolation (rechts)

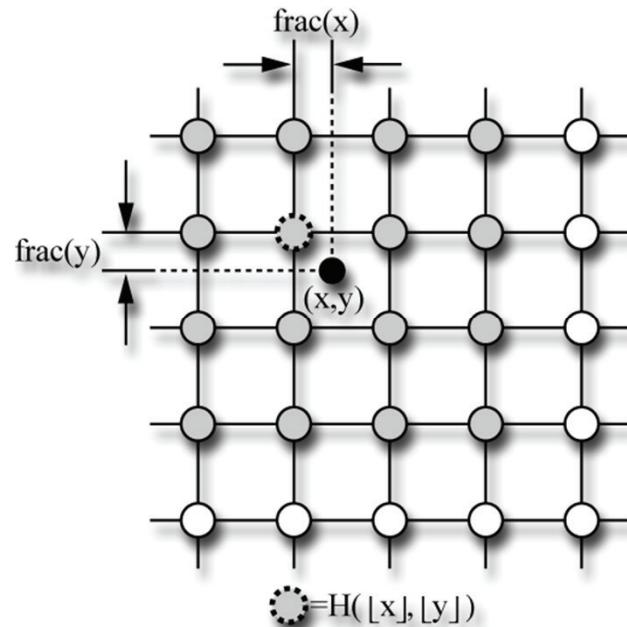


Bild: GPU Gems 2 <http://http.developer.nvidia.com/GPUGems2/>

Textur-Filterung

Texturfilter höherer Ordnung

- ▶ menschliche Wahrnehmung nimmt Unstetigkeiten in der „Krümmung“ wahr, deshalb verwendet man manchmal Interpolation höherer Ordnung
- ▶ Beispiel: bikubische Interpolation (nur damit Sie es mal gesehen haben)
 - ▶ benötigt Zugriff auf 4×4 Texel



$$H(x, y) = \sum_{i=-1}^2 \sum_{j=-1}^2 H(\lfloor x \rfloor + i, \lfloor y \rfloor + j) R(i - \text{frac}(x)) R(j - \text{frac}(y))$$

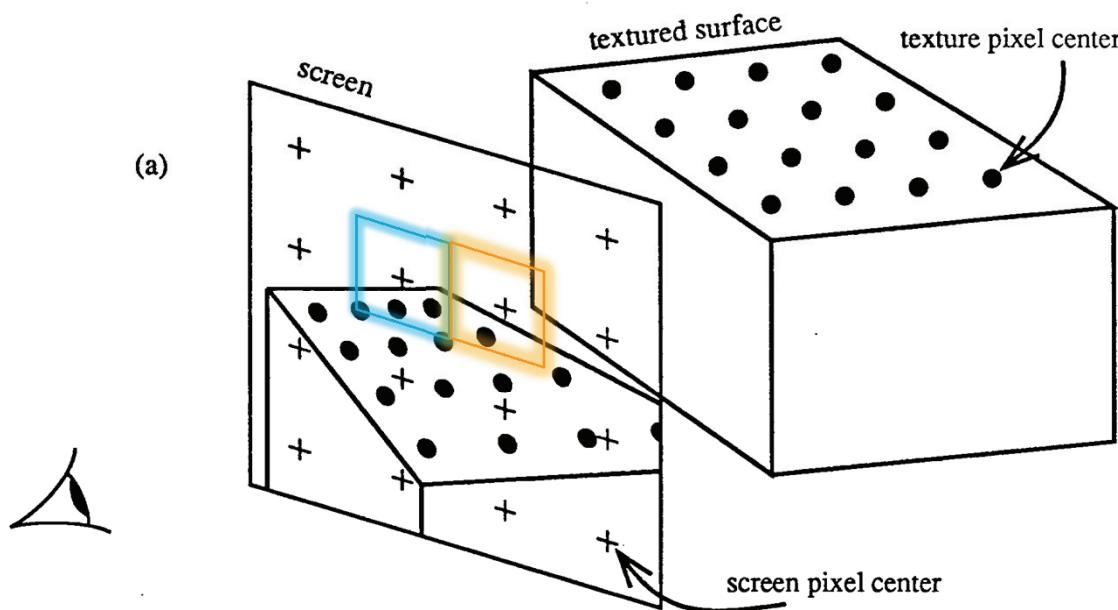
$$R(x) = \frac{1}{6} [P(x+2)^3 - 4P(x+1)^3 + 6P(x)^3 - 4P(x-1)^3], \text{ with}$$

$$P(x) = \max(0, x)$$

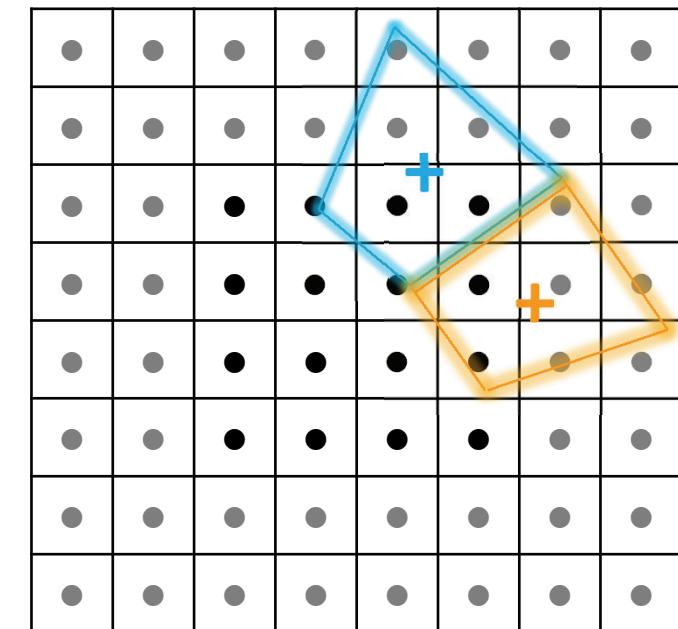
Textur-Filterung

Verkleinerung (Minification)

- ▶ Abbildung mehrerer Texel auf einen Pixel
- ▶ wird nur 1 Texel ausgelesen, obwohl der Pixel im Texturraum mehrere Texel bedeckt entstehen Aliasing Artefakte durch Unterabtastung

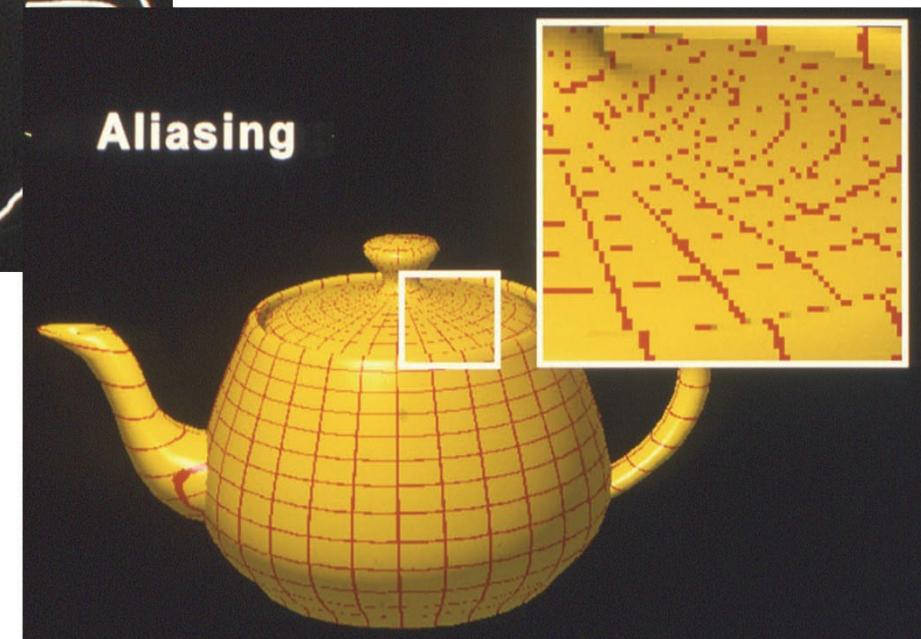
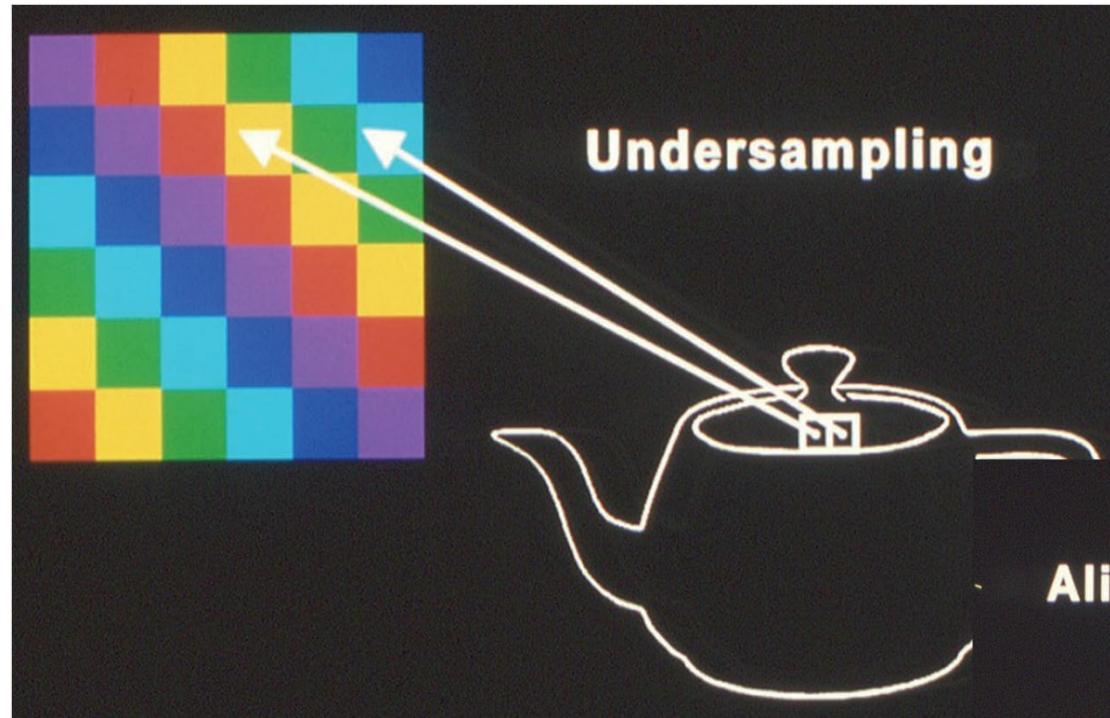


Abdruck („Footprint“) eines Pixels im Texturraum



Verkleinerung/Minification

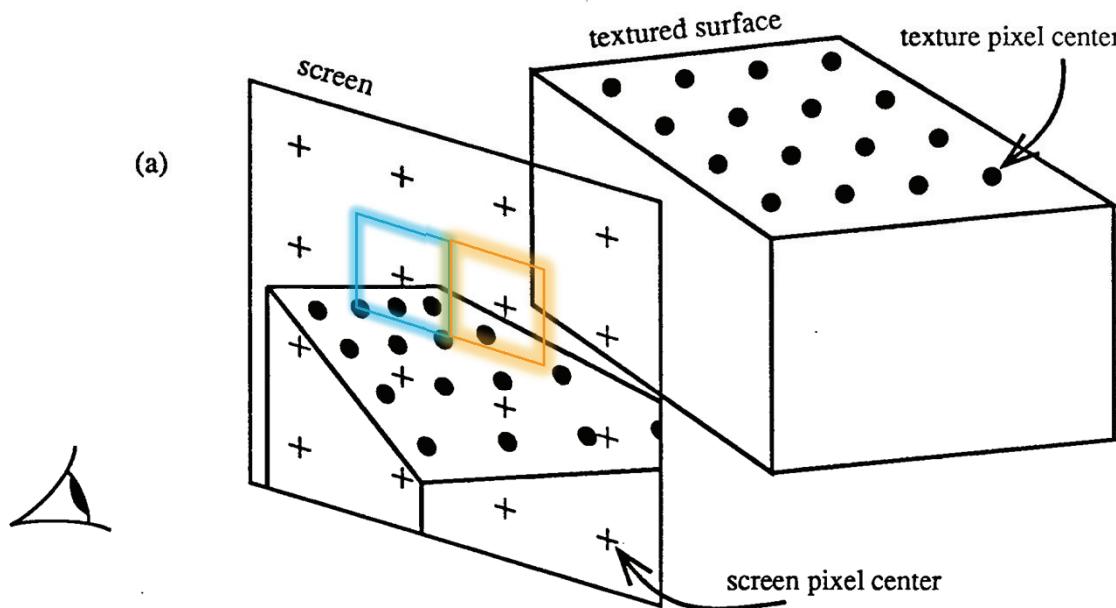
- wird nur 1 Texel ausgelesen, obwohl der Pixel im Texturraum mehrere Texel bedeckt entstehen Aliasing-Artefakte durch Unterabtastung



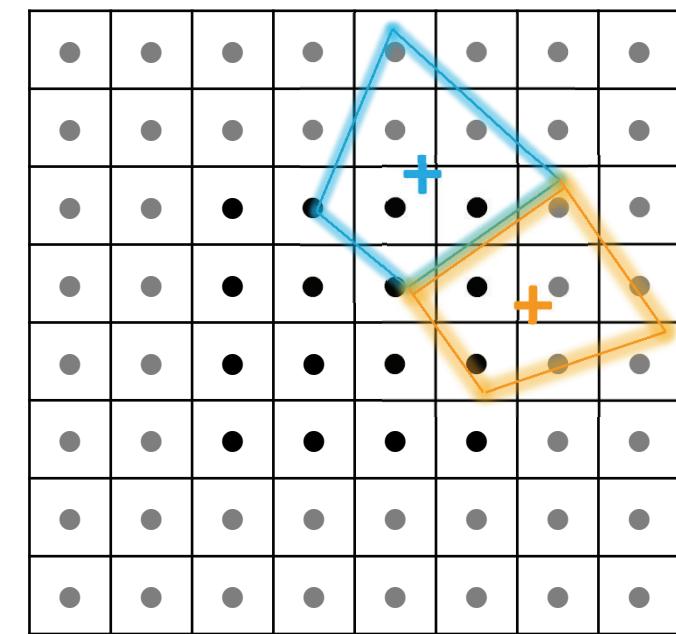
Textur-Filterung

Verkleinerung (Minification)

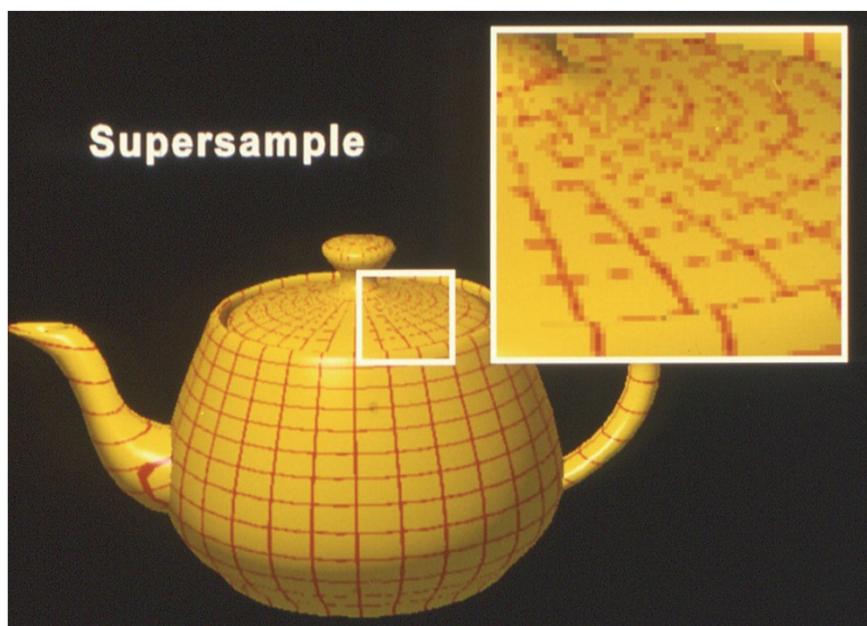
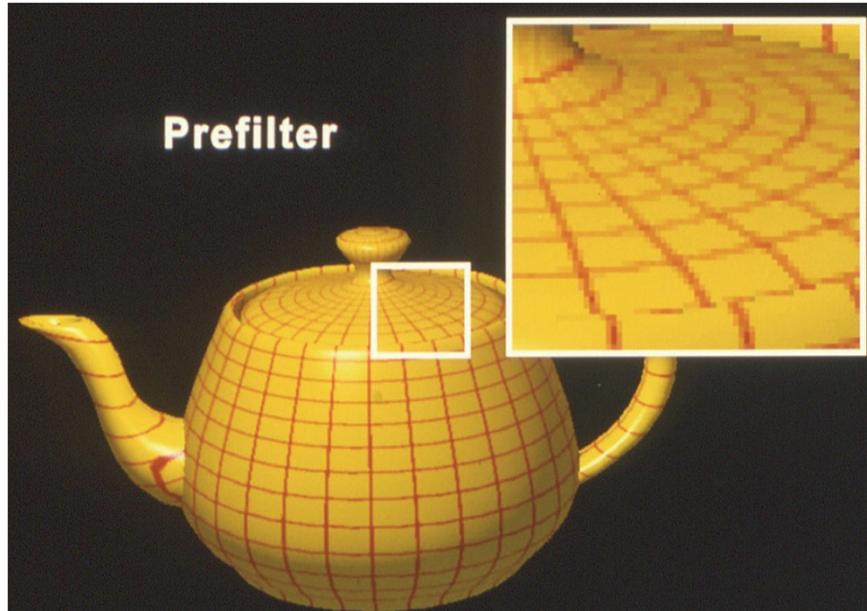
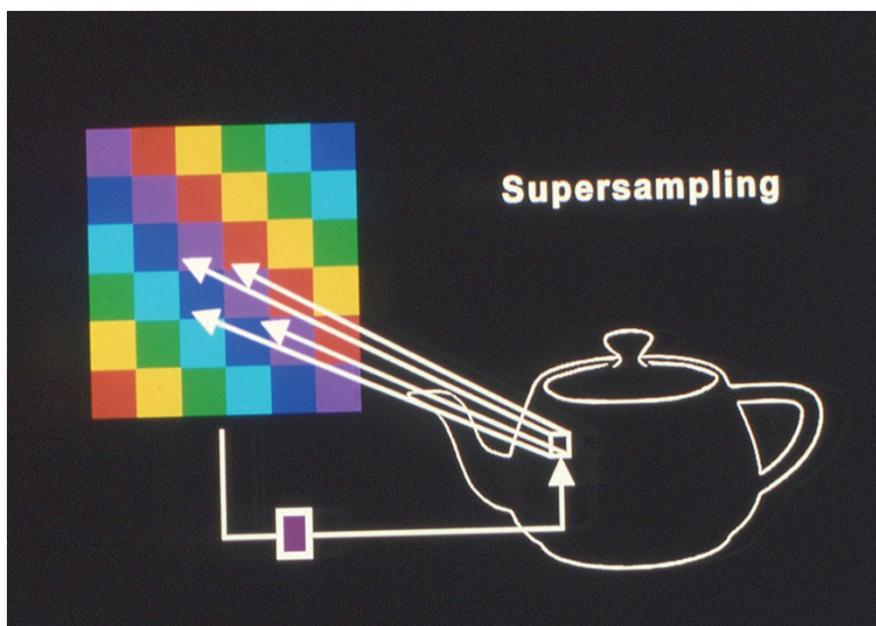
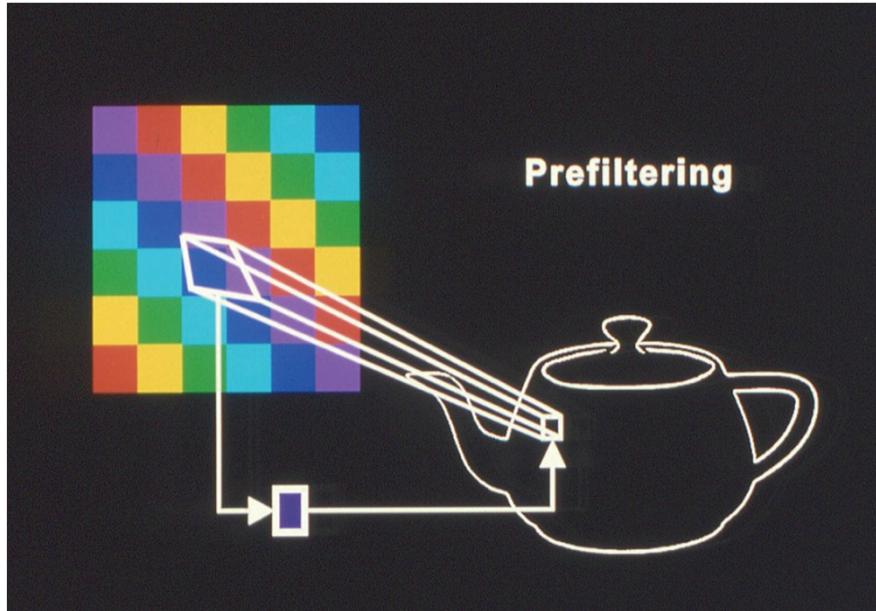
- ▶ zwei Möglichkeiten
 - ▶ Vorfilterung des Signals (hohe Frequenzen vor Abtastung entfernen)
 - ▶ Überabtastung (Supersampling), i.d.R. zu teuer!



Abdruck („Footprint“) eines
Pixels im Texturraum



Aliasing: Lösungsstrategien?



ausreichende Abtastfrequenz

Faltung mit dem Rekonstruktionsfilter, idealer Tiefpaßfilter: Sinc

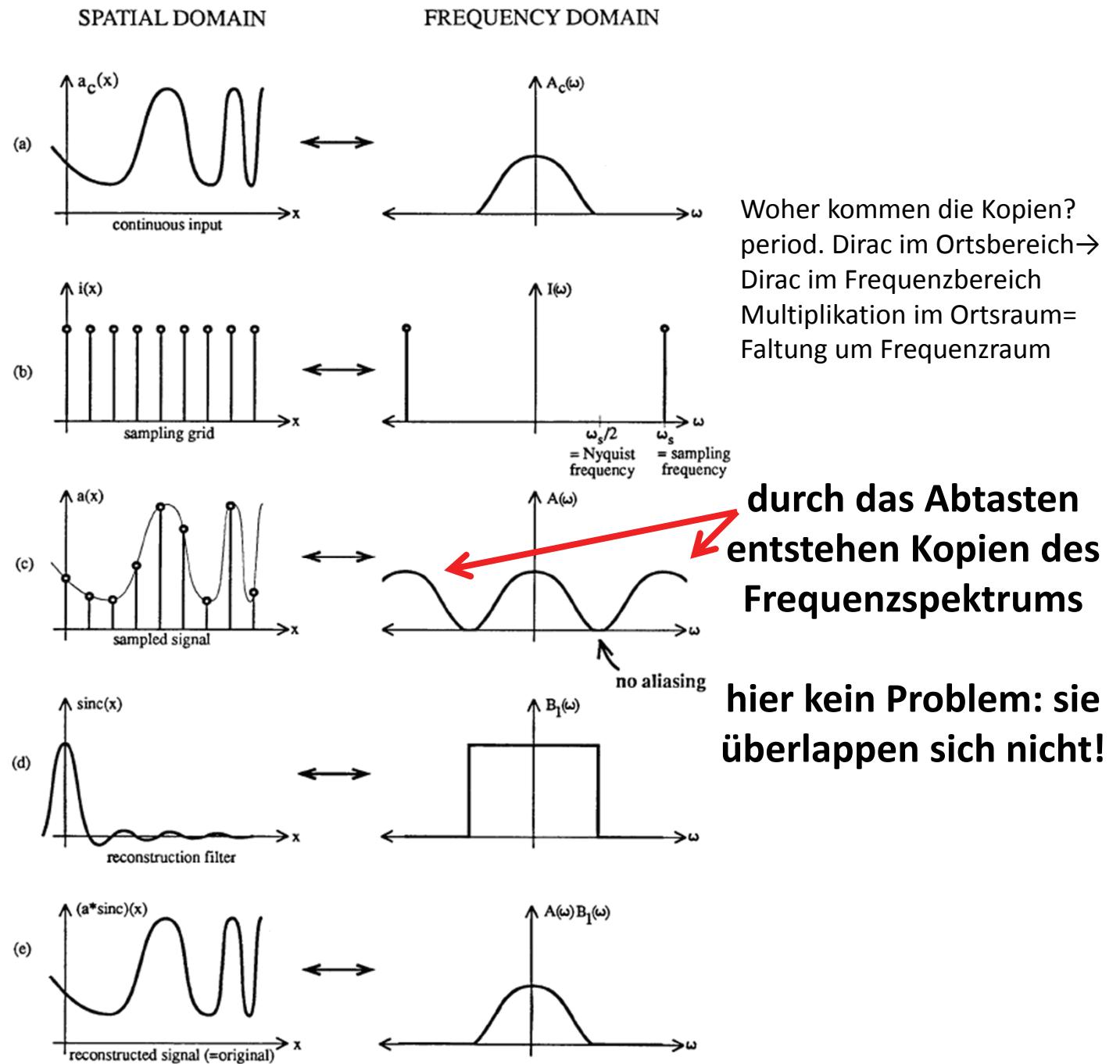
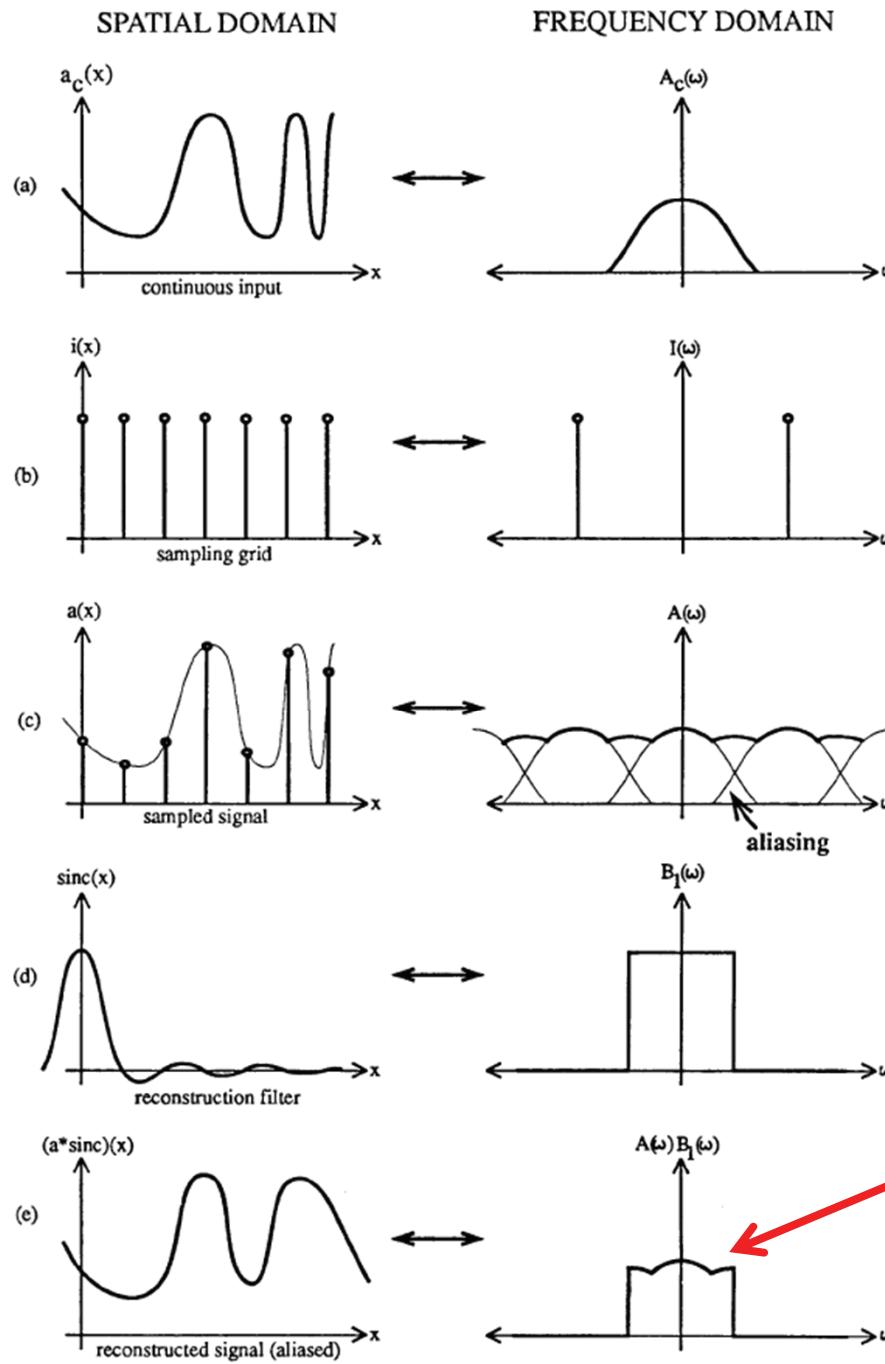


Figure 3.4: Sampling above the Nyquist rate (no aliasing).

**zu niedrige
Abtastfrequenz**

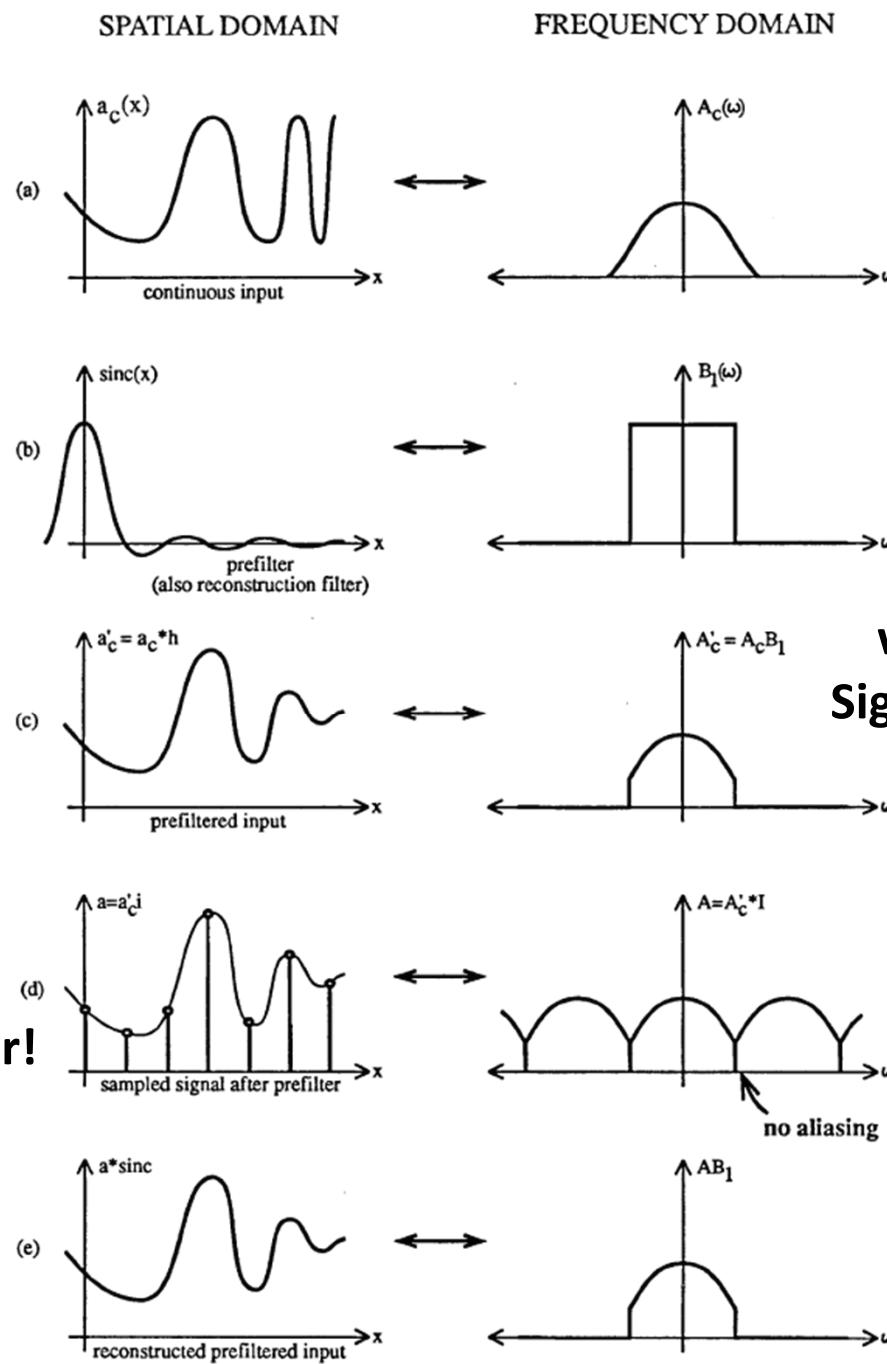


Überlappung!

Aliasing:
Frequenzanteile, die hier nicht sein sollten!

Figure 3.5: Sampling below the Nyquist rate (aliasing).

Vorfilterung:
hier wird zuerst
das Signal gefiltert



wir erhalten ein neues
Signal mit einem kleineren
Frequenzspektrum

**dann abtasten:
die Geisterspektren (d)
überlappen nicht mehr!**

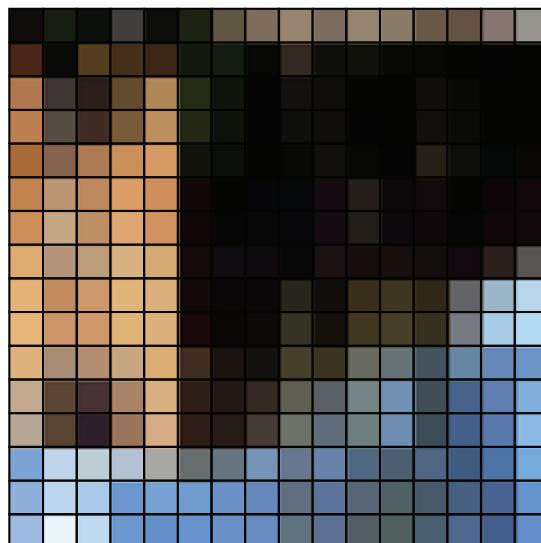
Figure 3.7: Ideal prefiltersing eliminates aliasing

Textur-Filterung

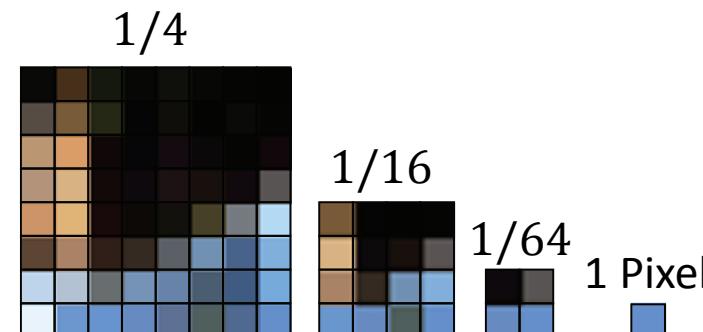
Mip-Mapping (lat. multum in parvo: viel in wenig)

- ▶ einfache Vorfilterung von Texturen
 - ▶ speichere rekursiv Texturen mit $\frac{1}{4}$ Größe
(Halbierung entlang jeder Achse)
 - ▶ meist Mittelung über je 2×2 Texel (**kein** optimaler Tiefpass!)
 - ▶ insgesamt nur 33% mehr Speicherbedarf: $1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \dots = \frac{4}{3}$
 - ▶ man spricht auch von einer „AuflösungsPyramide“

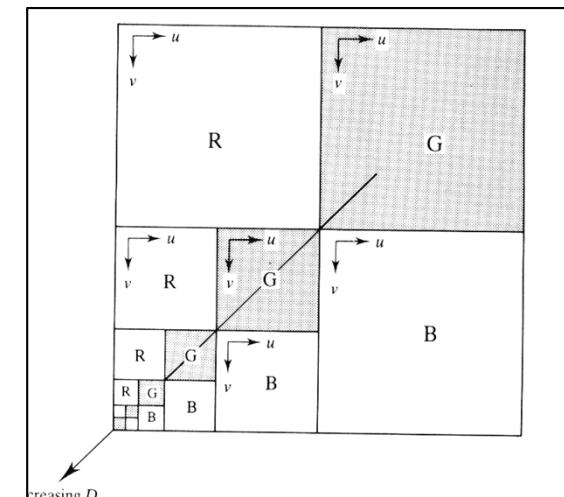
Originaltextur



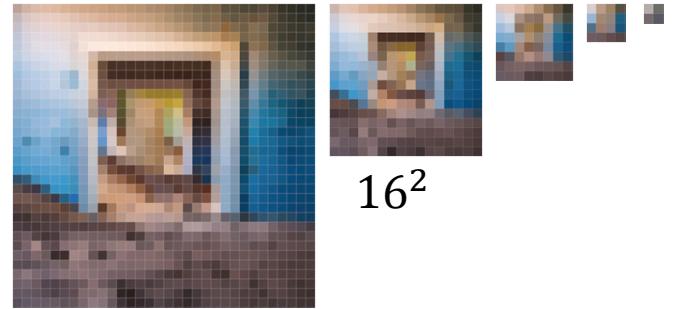
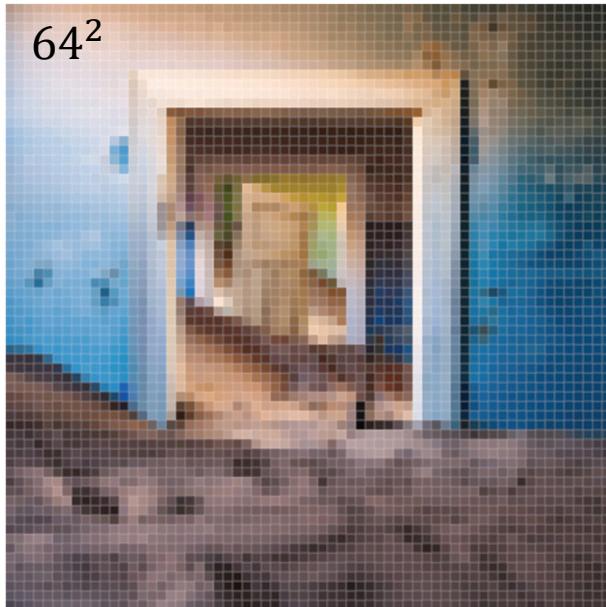
vorgefilterte
Texturen



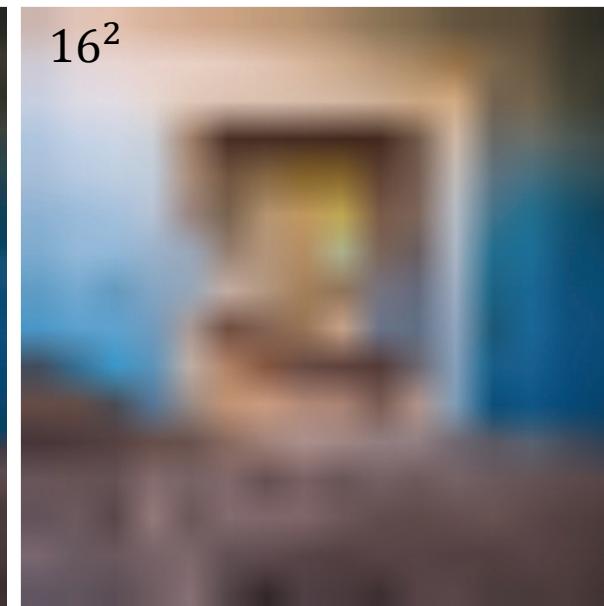
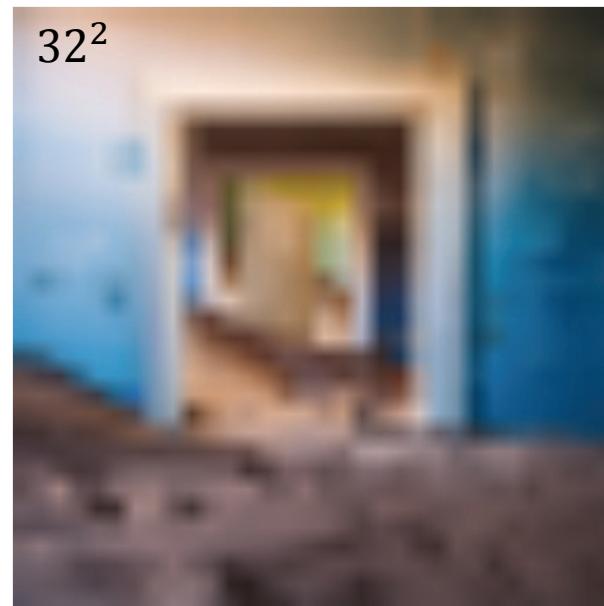
1/3 mehr Speicherbedarf



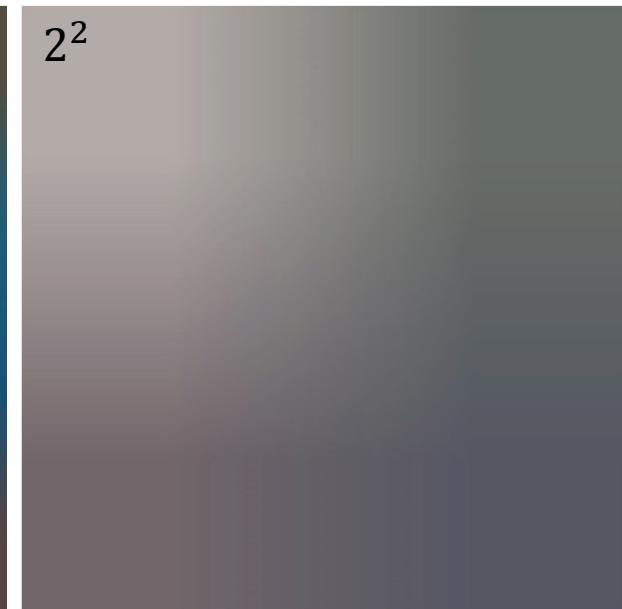
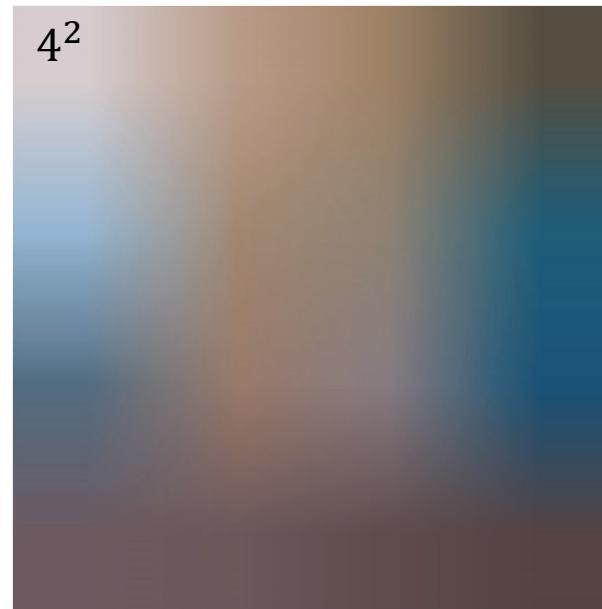
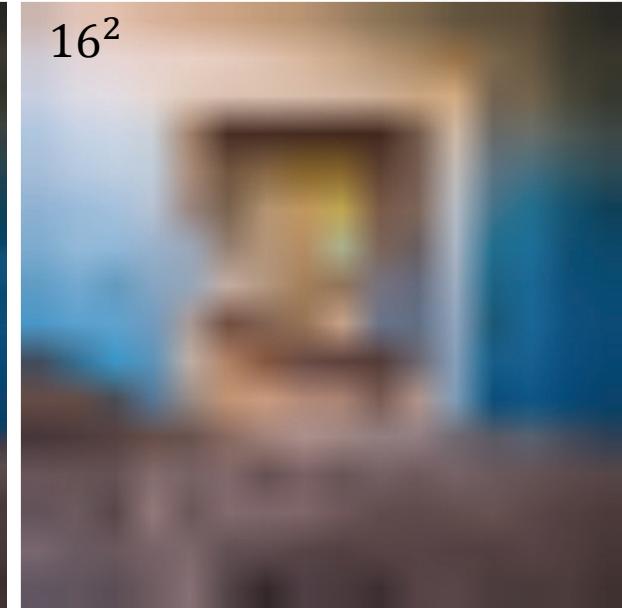
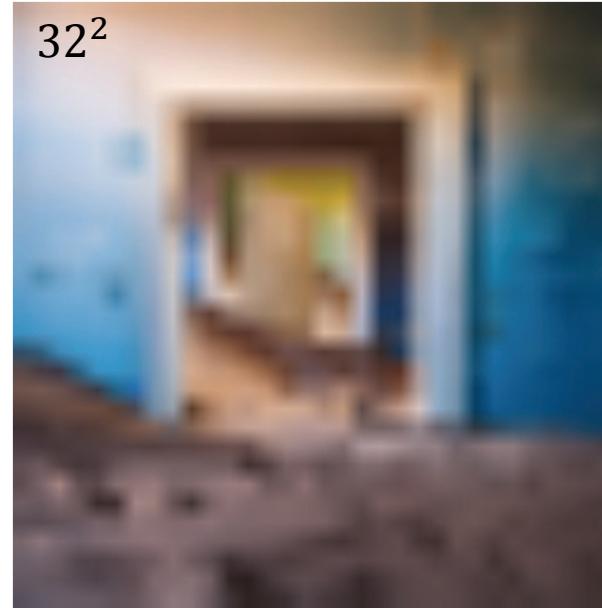
Mip-Mapping – Downsampling und Upsampling



16^2

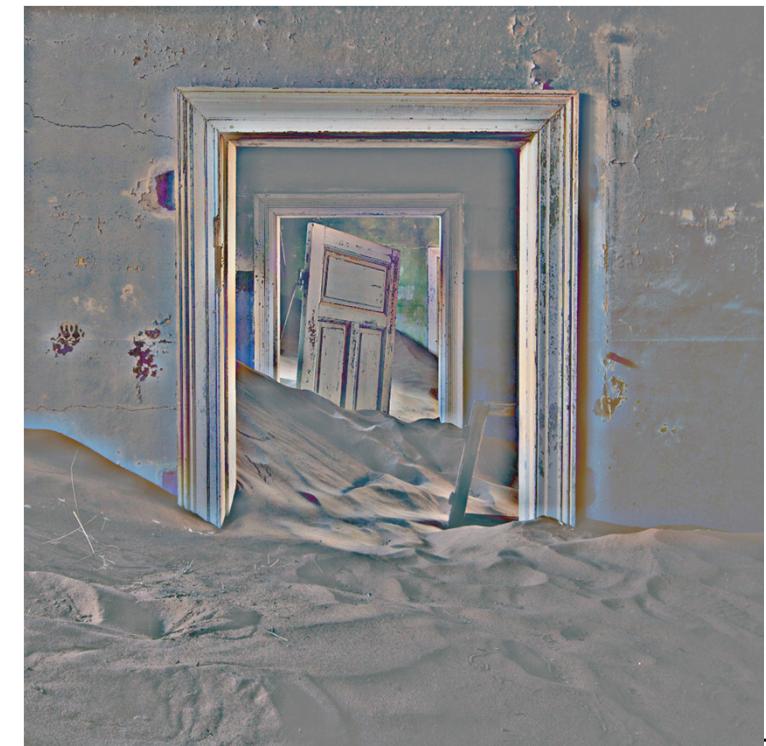


Mip-Mapping – Bilineares Upsampling



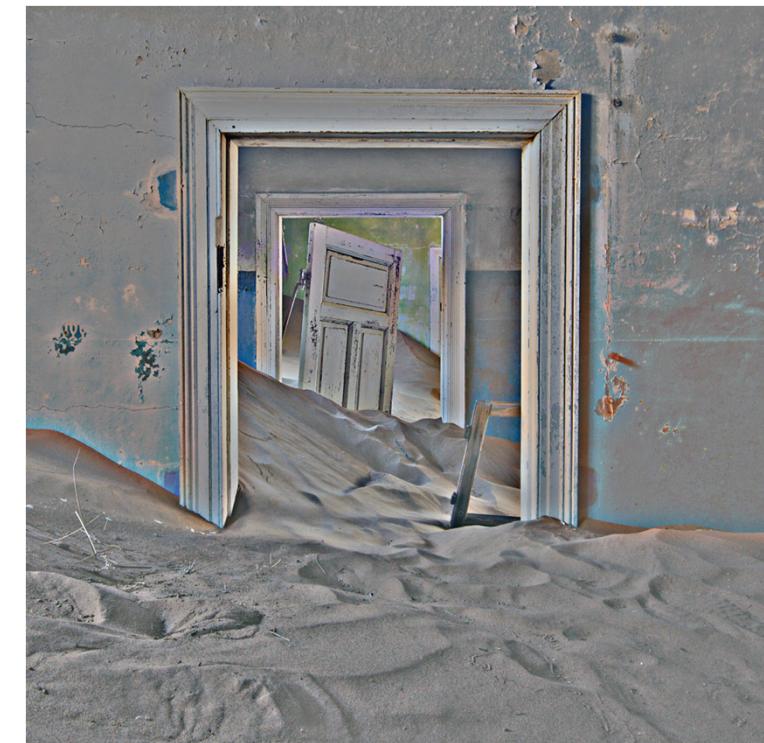
Tiefpass – Hochpass

- ▶ Tiefpass (links): bilineare Interpolation der 32^2 Mip-Map-Stufe
- ▶ Hochpass-Filter durch Differenz (Eingabe-Tiefpass)



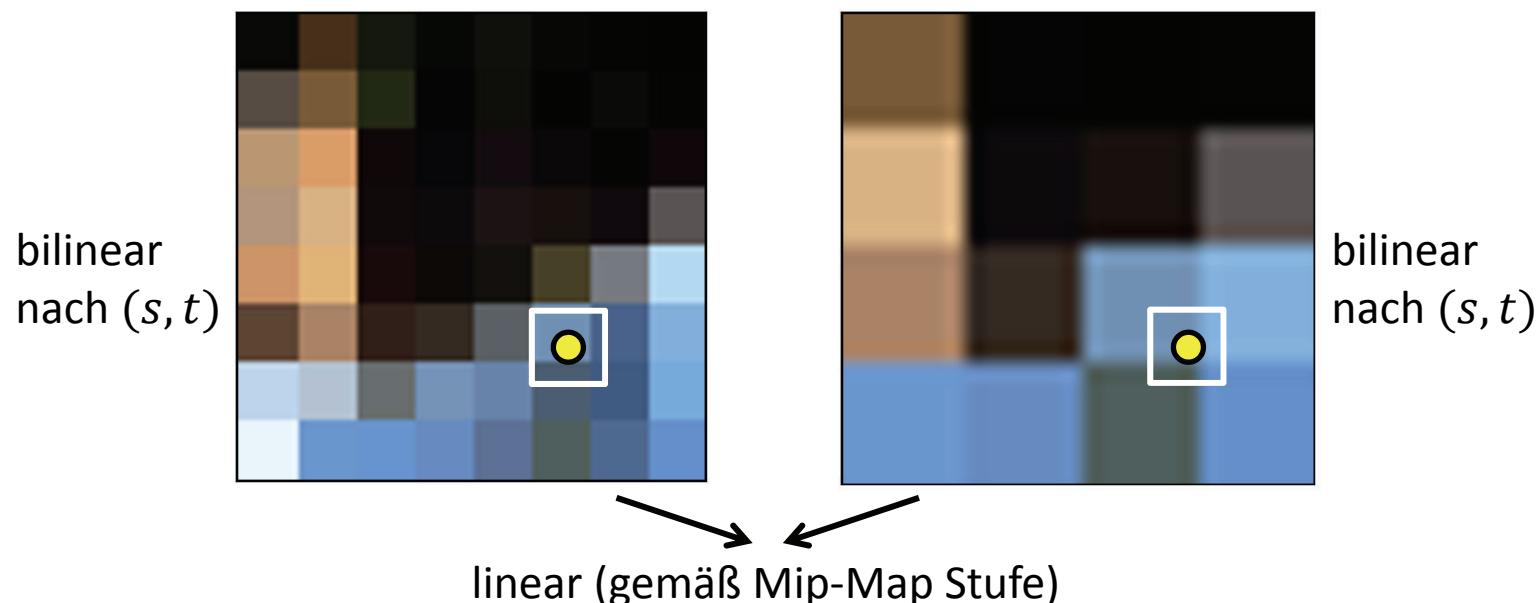
Tiefpass – Hochpass

- ▶ Gauß-Filter als Tiefpass (links)
- ▶ Hochpass-Filter durch Differenz (Eingabe-Tiefpass)



Mip-Mapping

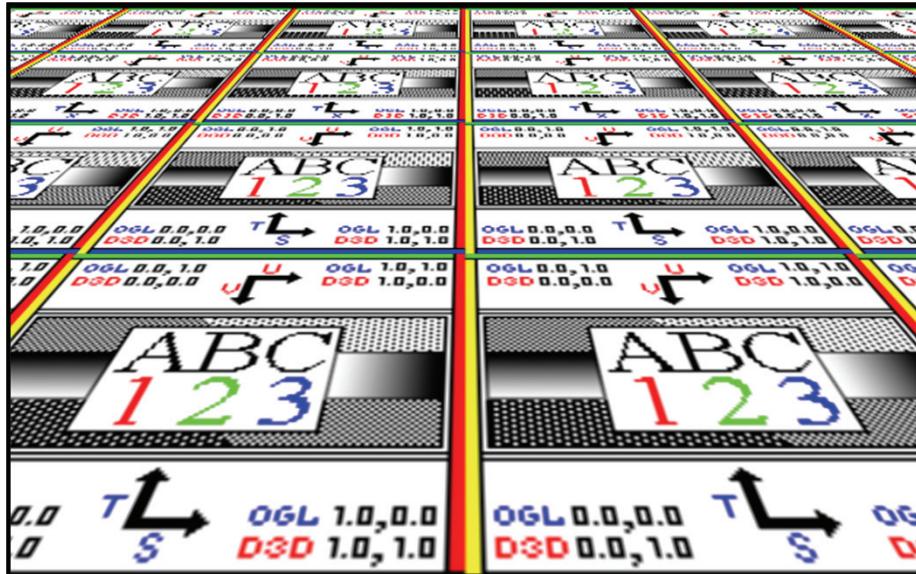
- ▶ wähle Texturauflösung (Mip-Map Stufe n) so, dass
 - ▶ $\text{Texelgröße}(n) \leq \text{Größe Pixelfootprint auf Textur} < \text{Texelgröße}(n + 1)$
(gebräuchliche Daumenregel für die Wahl der Mip-Map Stufe)
 - ▶ $n = 0$ entspricht der höchsten Auflösungsstufe
- ▶ Interpolation der Farbwerte
 - ▶ **trilineare Interpolation** der 8 nächstliegenden Texel
 - ▶ bilinear auf Stufe n , bilinear auf Stufe $n + 1$
 - ▶ anschließend linear zwischen diesen Farben



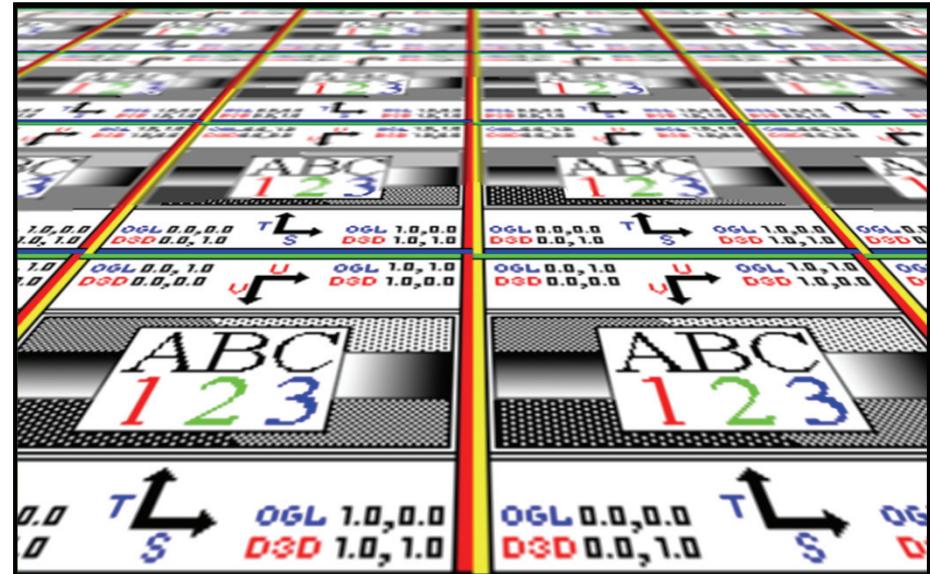
Mip-Mapping cont.



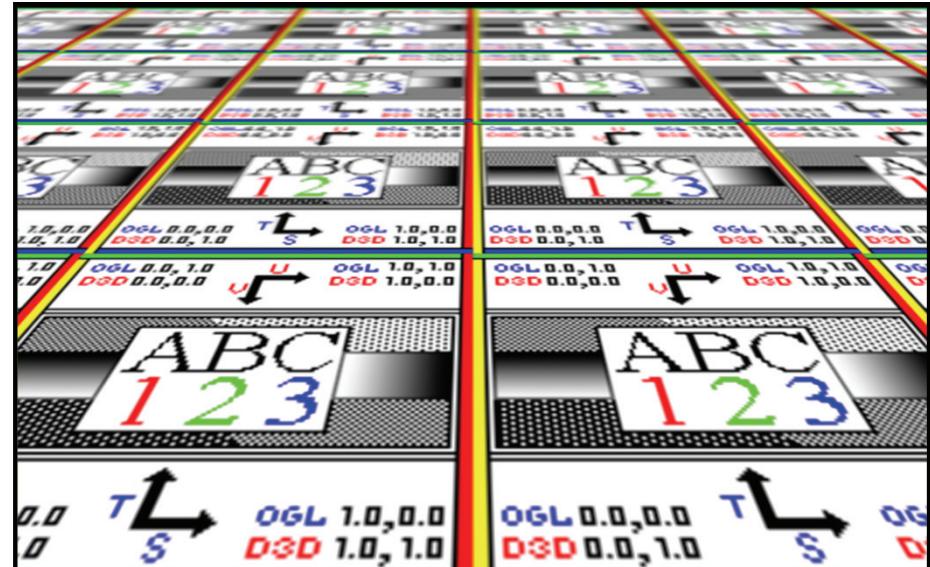
ohne Mip-Mapping



mit Mip-Mapping, nur Mip-Level n

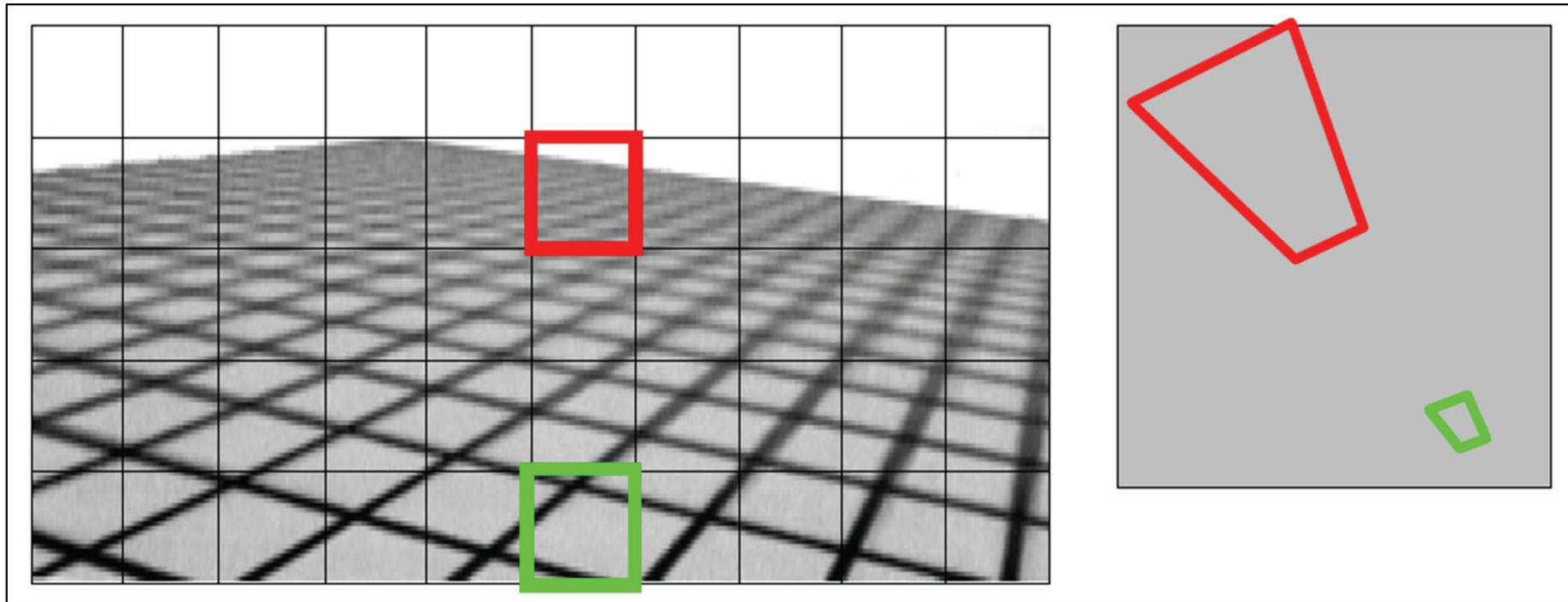


Mip-Mapping mit
Mip-Level n und $n + 1$
→ trilineare Interpolation

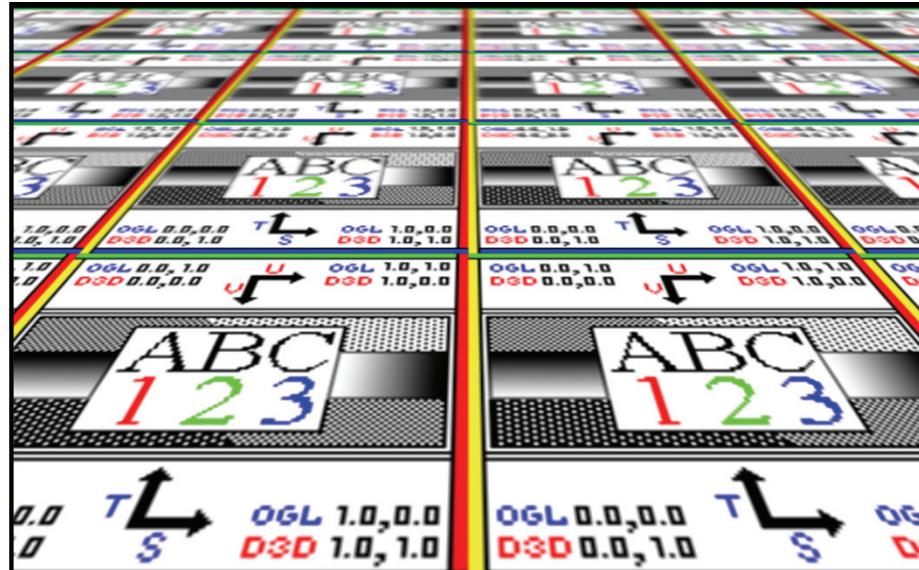


Anisotrope Texturfilterung

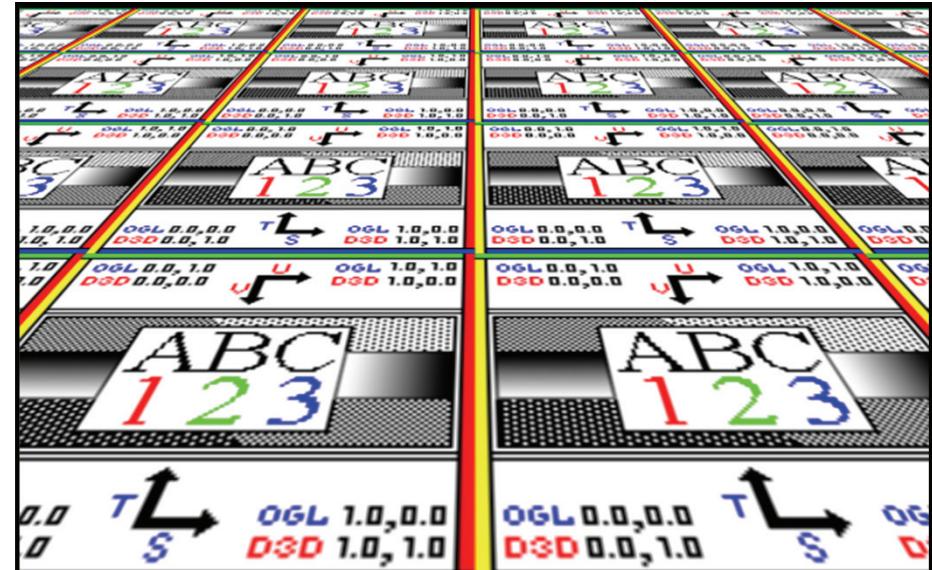
- ▶ Mip-Mapping resultiert oft in sehr verwaschenen Details
- ▶ der Abdruck (Footprint) eines Pixels im Texturraum ist oft eher länglich
 - ▶ die Vorfilterung bei Mip-Mapping ist isotrop (gleichförmig in s und t)
- ▶ wie kann man bei der Vorfilterung darauf Rücksicht nehmen?



Anisotrope Texturfilterung



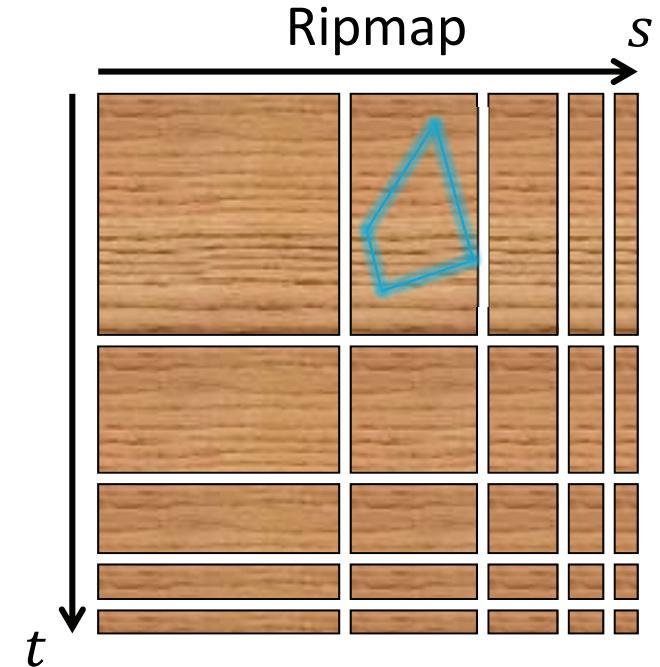
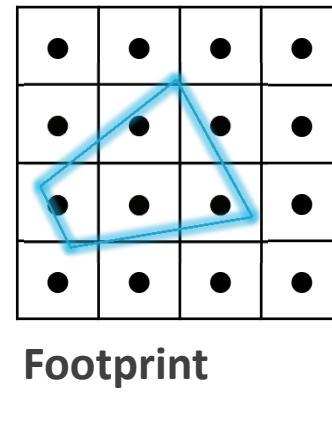
Mip-Mapping
trilineare Interpolation



Anisotrope Texturfilterung

Anisotrope Texturfilterung

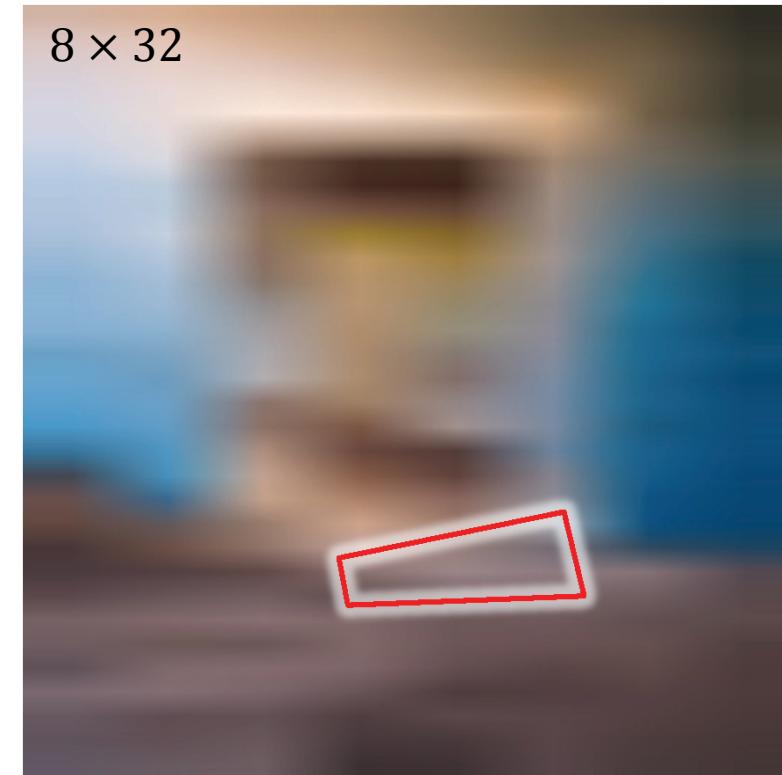
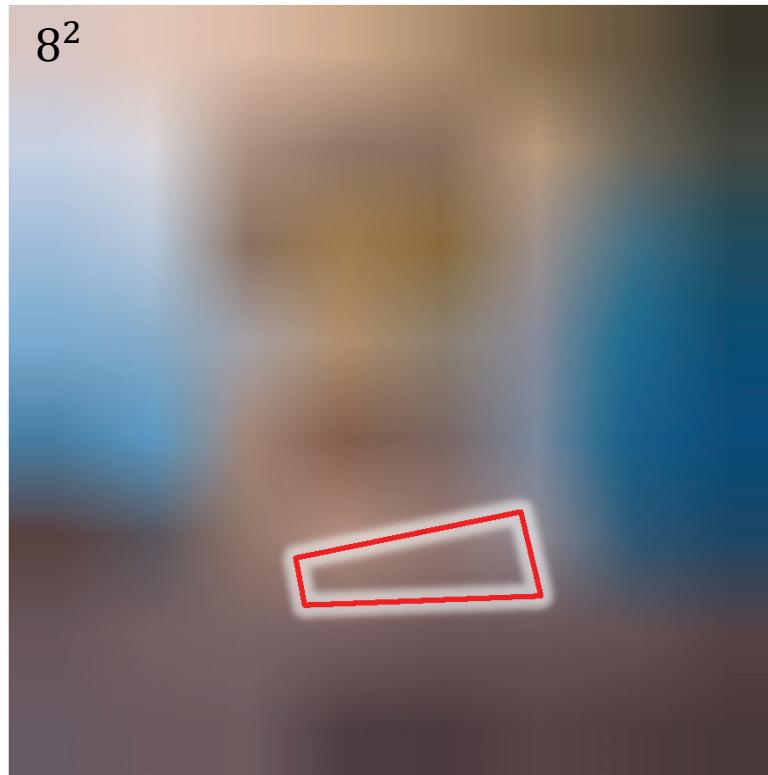
- ▶ RIPmaps (Rectangular MipMaps, selten in der Praxis)
 - ▶ enthält Vorfilterungen unabhängig in jeder Achse
 - ▶ länglicher Abdruck im Texturraum
→ wähle entsprechend vorgefilterte Textur
 - ▶ 4 × Speicherbedarf



- ▶ anisotrope Filterung in der Praxis (und in Grafikhardware)
 - ▶ Abtasten des Bereichs durch eine geschickte Kombination von Abtastungen verschiedener Mip-Map Stufen
 - ▶ typischerweise mehrere Abtaststellen (z.B. 8 oder 16)

Mip-Mapping – RIP-Mapping

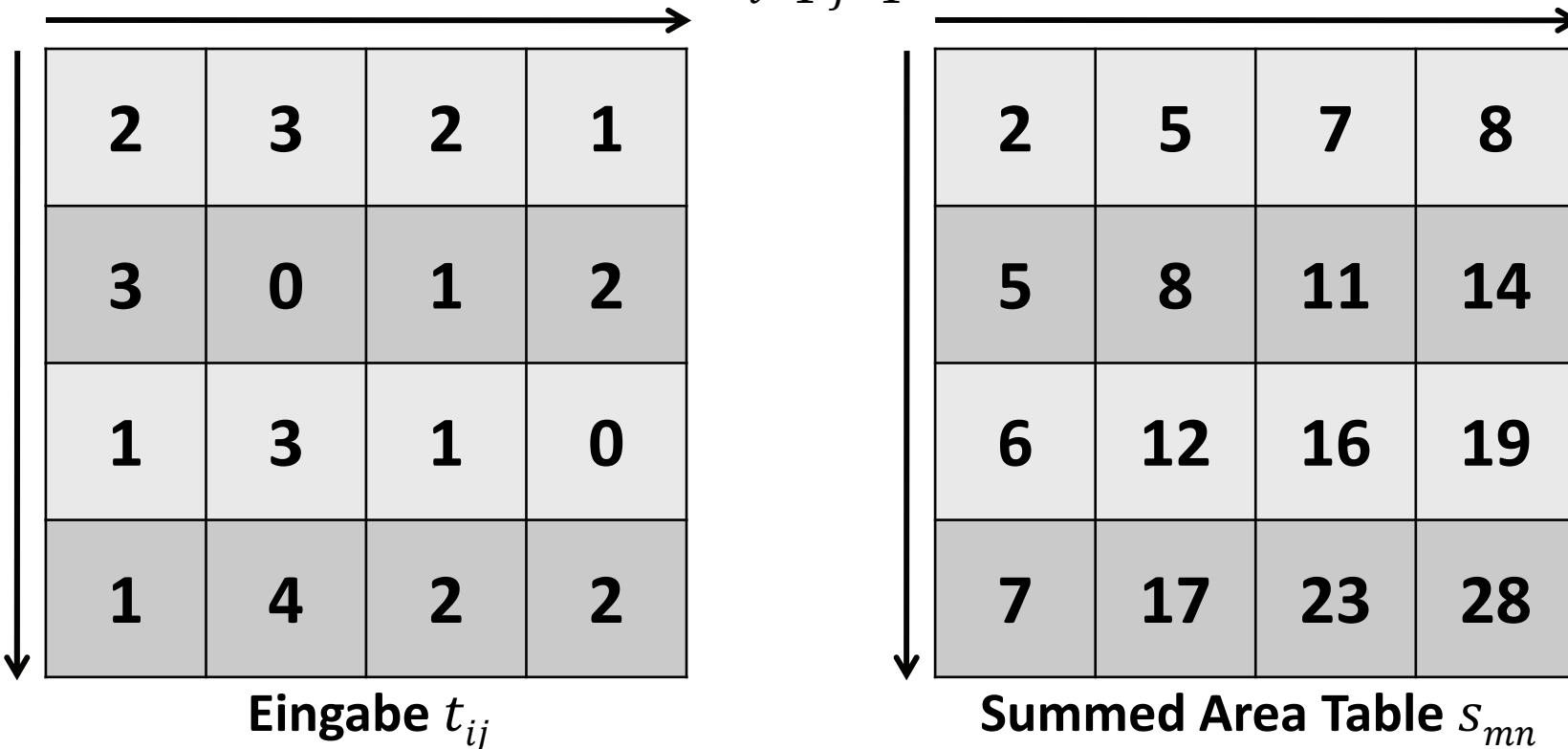
- Mip-Maps stellen eine isotrope Vorfilterung dar, RIP-Maps eine getrennte Vorfilterung nach s bzw. t
- Beispiel: unnötige/zu starke vertikale Vorfilterung bei Mip-Maps (links)



Summed Area Tables [Crow84]

- ▶ Summed Area Table (SAT): interessantes Konzept für viele Berechnungen, benötigt höhere Genauigkeit als 8-Bit pro (Farb-)Wert
- ▶ jedes Element s_{mn} einer SAT enthält die Summe aller Elemente darüber und links aus der Eingabetextur/-tabelle

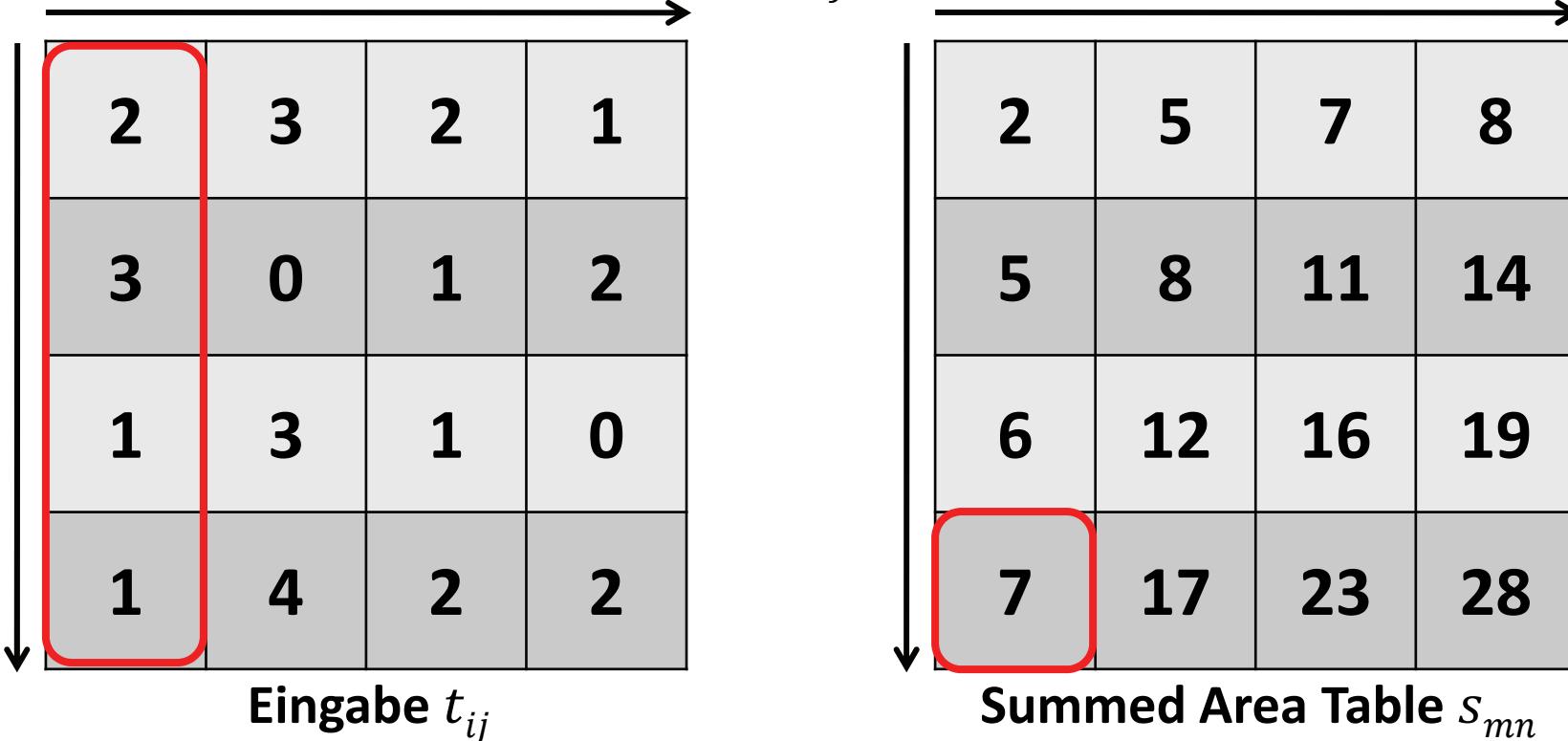
$$S_{mn} = \sum_{i=1}^m \sum_{j=1}^n t_{ij}$$



Summed Area Tables [Crow84]

- ▶ Summed Area Table (SAT): interessantes Konzept für viele Berechnungen, benötigt höhere Genauigkeit als 8-Bit pro (Farb-)Wert
- ▶ jedes Element s_{mn} einer SAT enthält die Summe aller Elemente darüber und links aus der Eingabetextur/-tabelle

$$S_{mn} = \sum_{i=1}^m \sum_{j=1}^n t_{ij}$$

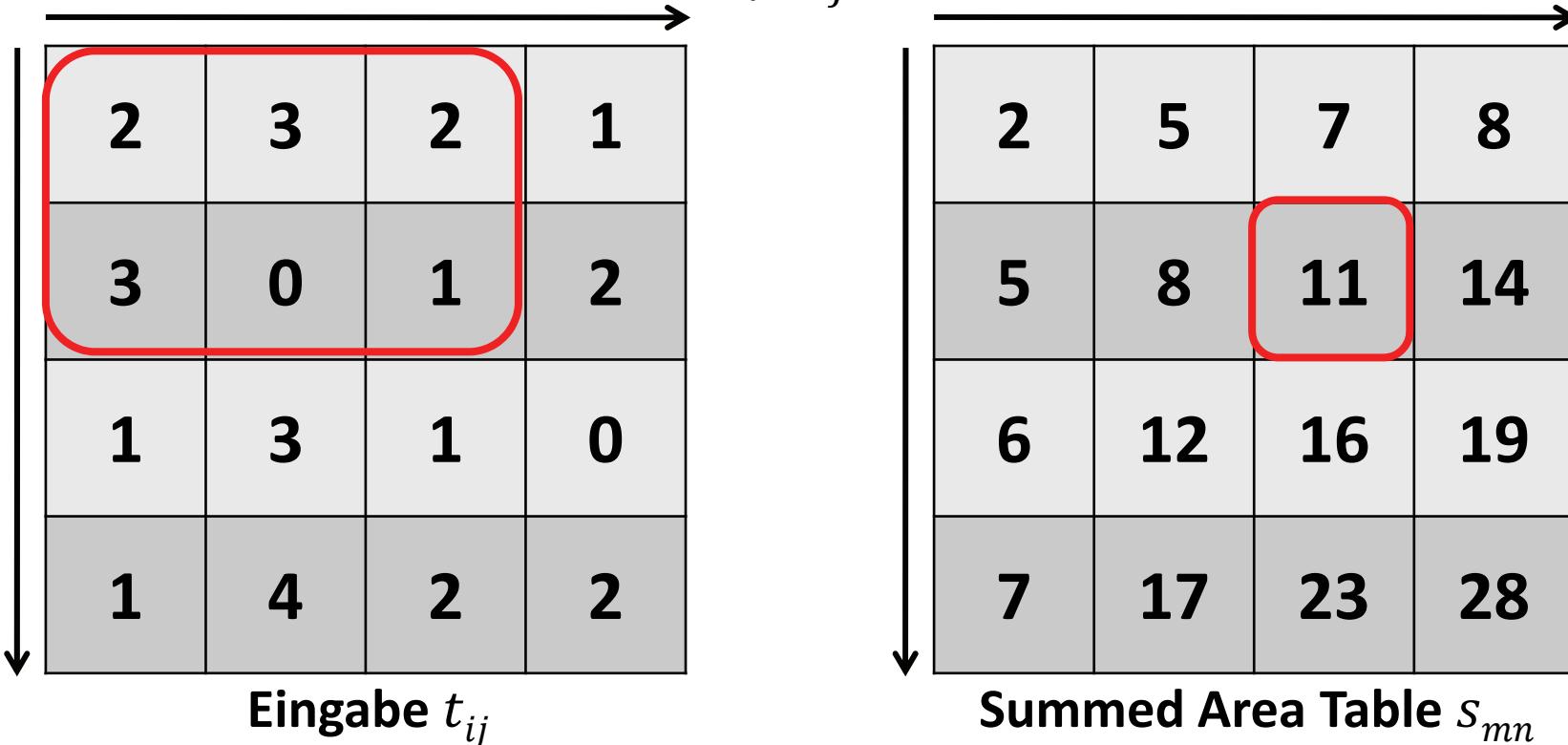


Eingabe t_{ij}				Summed Area Table s_{mn}			
2	3	2	1	2	5	7	8
3	0	1	2	5	8	11	14
1	3	1	0	6	12	16	19
1	4	2	2	7	17	23	28

Summed Area Tables [Crow84]

- ▶ Summed Area Table (SAT): interessantes Konzept für viele Berechnungen, benötigt höhere Genauigkeit als 8-Bit pro (Farb-)Wert
- ▶ jedes Element s_{mn} einer SAT enthält die Summe aller Elemente darüber und links aus der Eingabetextur/-tabelle

$$S_{mn} = \sum_{i=1}^m \sum_{j=1}^n t_{ij}$$



2	3	2	1
3	0	1	2
1	3	1	0
1	4	2	2

Eingabe t_{ij}

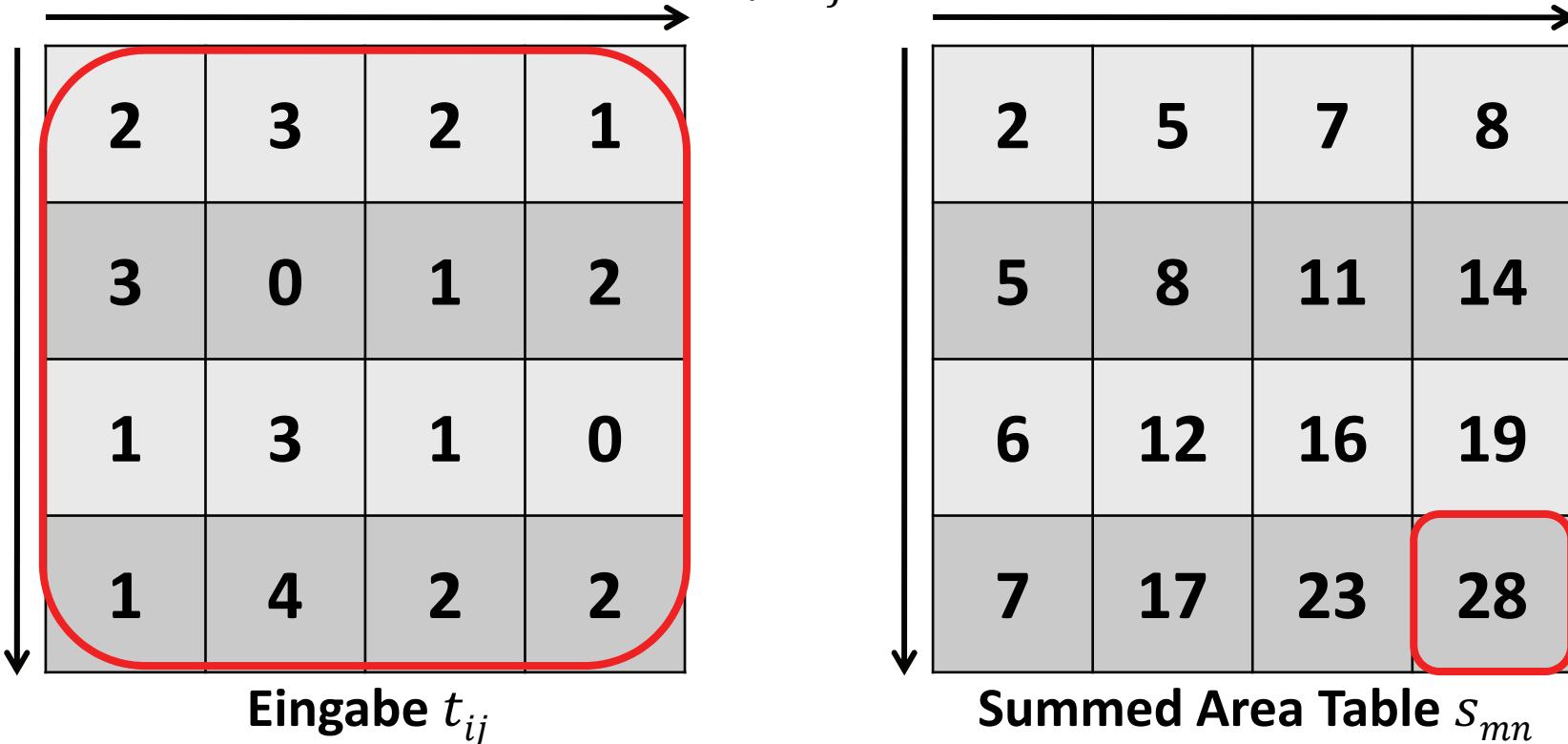
2	5	7	8
5	8	11	14
6	12	16	19
7	17	23	28

Summed Area Table s_{mn}

Summed Area Tables [Crow84]

- ▶ Summed Area Table (SAT): interessantes Konzept für viele Berechnungen, benötigt höhere Genauigkeit als 8-Bit pro (Farb-)Wert
- ▶ jedes Element s_{mn} einer SAT enthält die Summe aller Elemente darüber und links aus der Eingabetextur/-tabelle

$$S_{mn} = \sum_{i=1}^m \sum_{j=1}^n t_{ij}$$

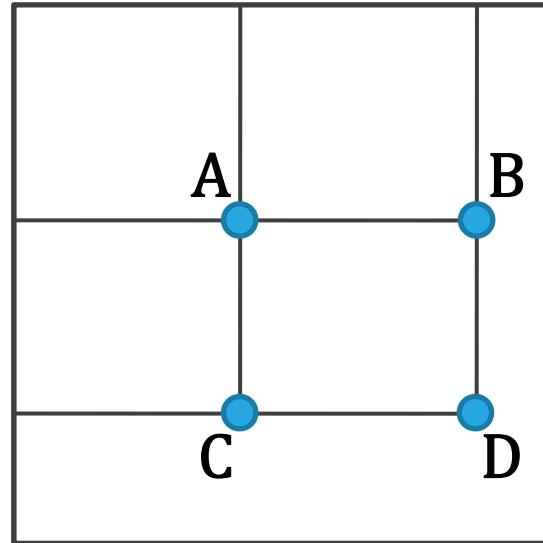


Eingabe t_{ij}				Summed Area Table s_{mn}			
2	3	2	1	2	5	7	8
3	0	1	2	5	8	11	14
1	3	1	0	6	12	16	19
1	4	2	2	7	17	23	28

Summed Area Tables [Crow84]



- ▶ eine Anwendung: Filterung der Eingabetextur mit einem beliebigen rechteckigen, nach den Achsen ausgerichteten Filter in konstanter Zeit



$$A = s_{a_1 a_2} = \sum_{i=1}^{a_1} \sum_{j=1}^{a_2} t_{ij}$$

$$D - B - C + A = \sum_{i=a_1+1}^{d_1} \sum_{j=a_2+1}^{d_2} t_{ij}$$

- ▶ Approximation des Pixel-Footprints

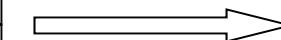
1	0	1	0
0	1	0	1
1	0	1	0
0	1	0	1

Textur

Sum of Area

1	1	2	2
1	2	3	4
2	3	5	6
2	4	6	8

1	1	2	2
1	2	3	4
2	3	5	6
2	1	6	8

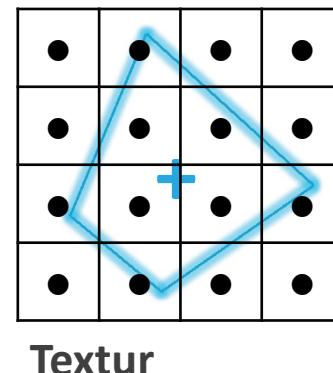


1	1	2	2
1	2	3	4
2	3	5	6
2	4	6	8

Bestimmung des Footprints

- ▶ einfachste Möglichkeit:
 - ▶ schicke einen Primärstrahl durch die Ecken eines Pixels oder betrachte einen 2×2 Pixelblock
 - ▶ bestimme die 4 Texturkoordinaten
 - ▶ Differenz der Texturkoordinaten liefert Größe/Form des Footprints
 - ▶ Mip-Mapping: bestimme $\max(Breite, Höhe)$ und daraus n
 - ▶ Rip-Mapping, SAT: betrachte Breite und Höhe
 - ▶ Textur-Lookup an berechneter Stelle mit entsprechender Filterung

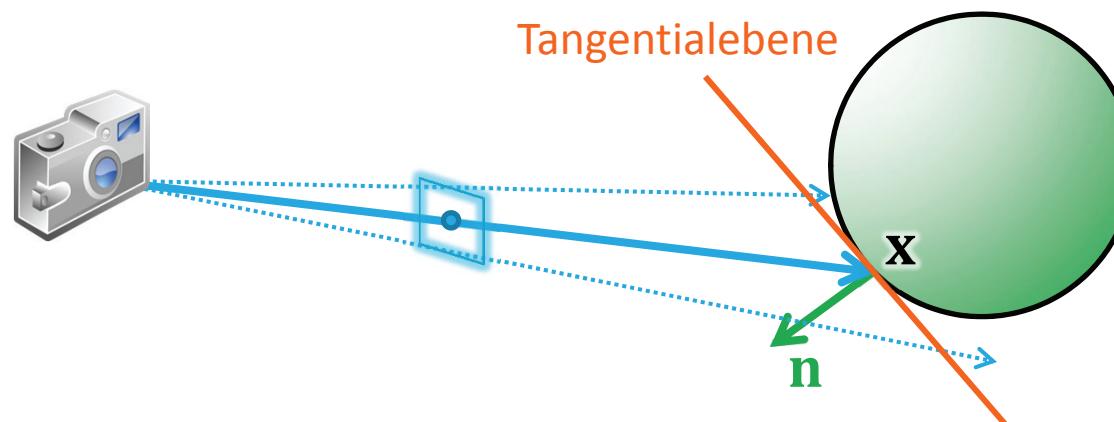
Abdruck („Footprint“) eines
Pixels im Texturraum



Bestimmung des Footprints

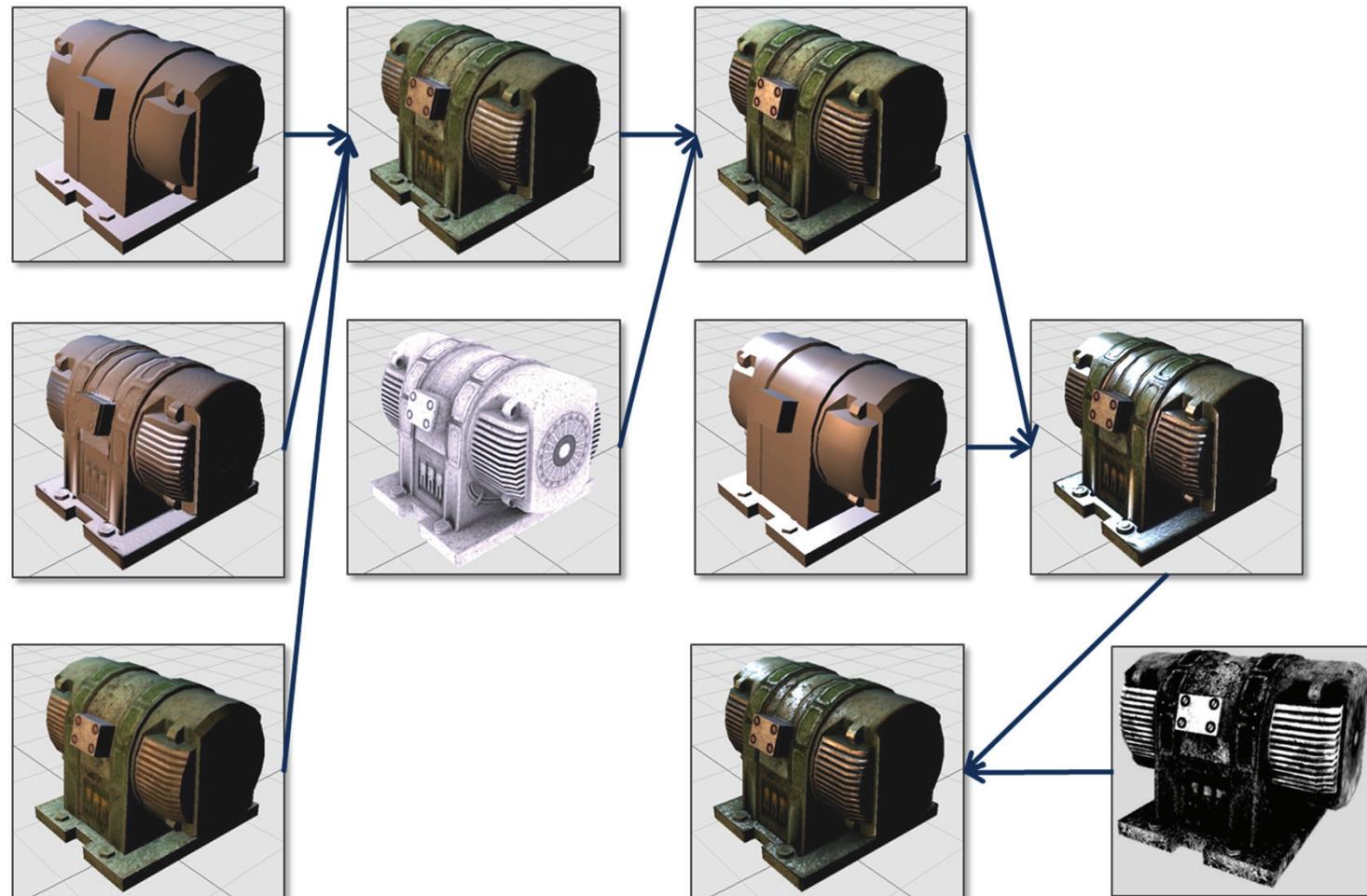
Grundidee der sogenannten *Ray Differentials*

- ▶ Schnitt eines Primärstrahls mit einem Objekt liefert
 - ▶ Position x und Normale n → definieren Tangentialebene
 - ▶ betrachte die Veränderung der Texturkoordinaten (Gradient bzgl. der Bildschirmachsen, z.B. wieder über Differenzen)
 - ▶ bestimme daraus Größe/Form des Footprints
 - ▶ lässt sich verallgemeinern und fortsetzen für Spiegelungen, Transmission etc.



Multitexturing Beispiel

- ▶ „komplexe“ optische Effekte durch Kombinationen mehrerer Texturen
- ▶ Texturen enthalten auch andere Informationen, nicht nur Farbe

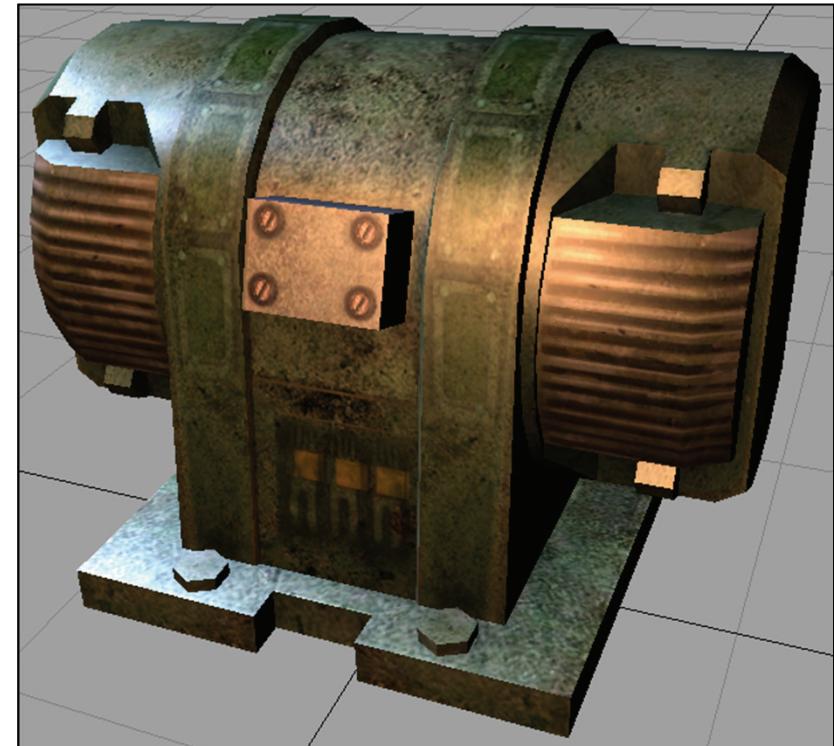
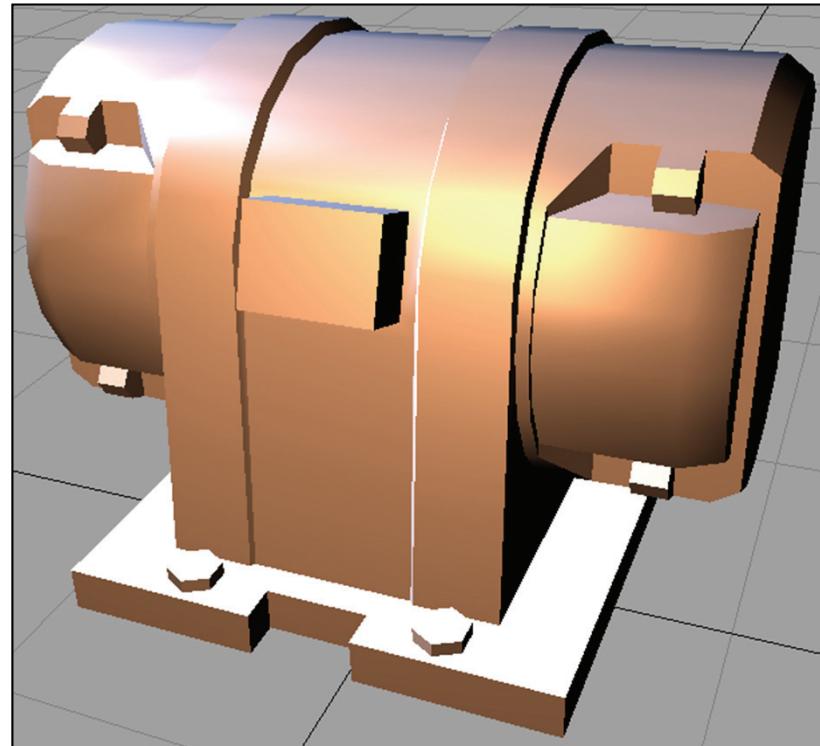


Texturierungstechniken

Diffuse Textur

- ▶ Kontrolle der Eigenfarbe eines Materials
- ▶ Bsp. Phong-Beleuchtungsmodell: k_d aus Textur

$$I = k_a \cdot I_L + k_d \cdot I_L \cdot (\mathbf{N} \cdot \mathbf{L}) + k_s \cdot I_L \cdot (\mathbf{R} \cdot \mathbf{V})^n$$



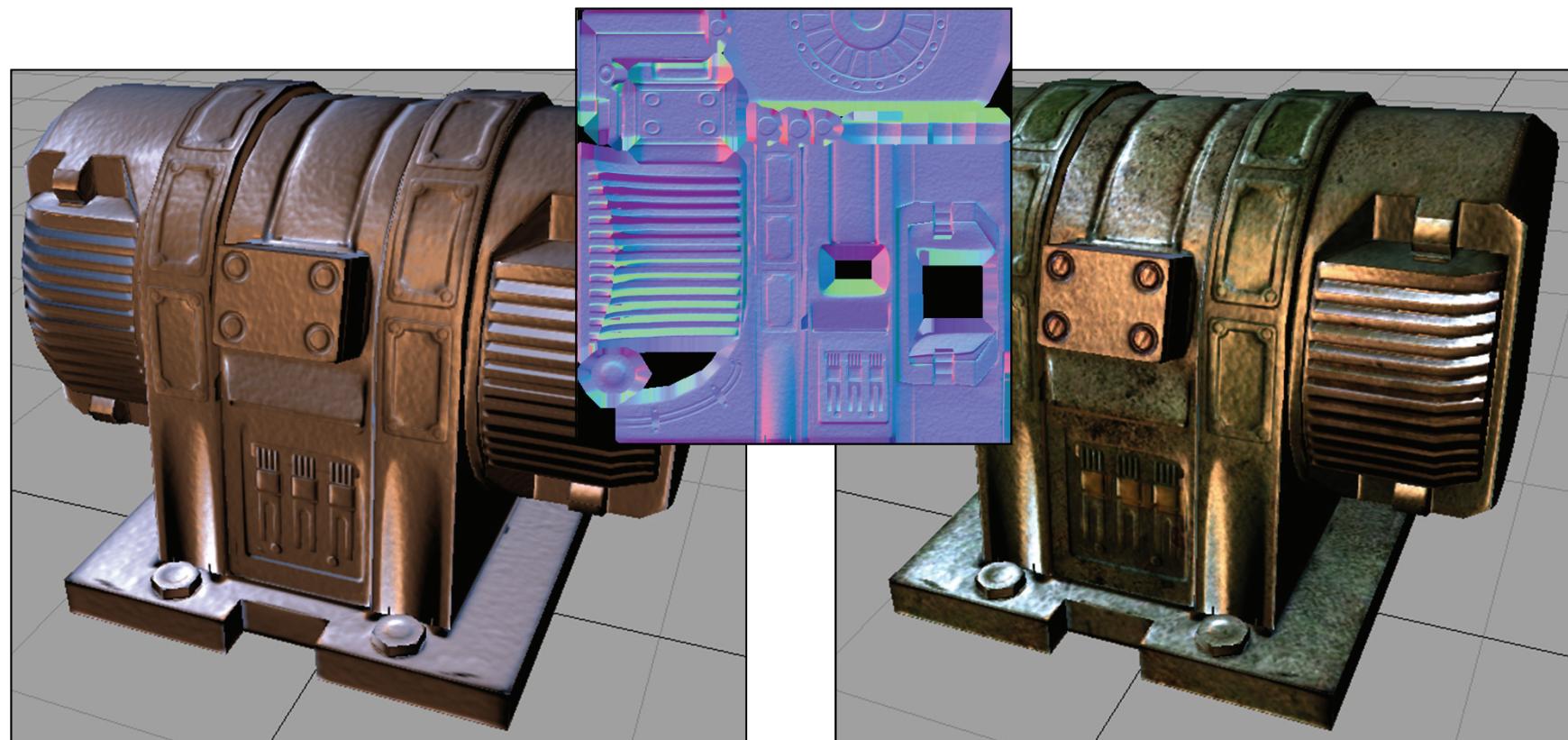
Texturierungstechniken



Bump oder Normal Mapping

- ▶ Variation der Normale einer Oberfläche
- ▶ verändertes **N** aus Textur (und dementsprechend auch anderes **R**)

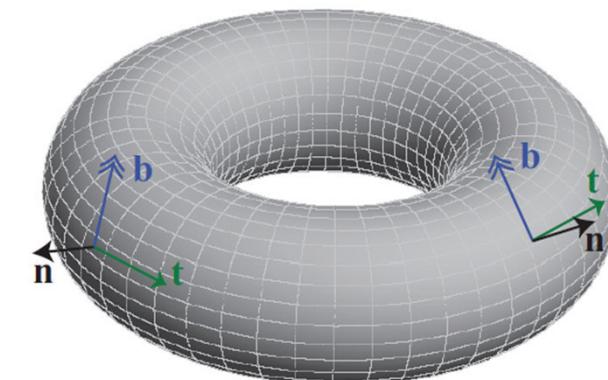
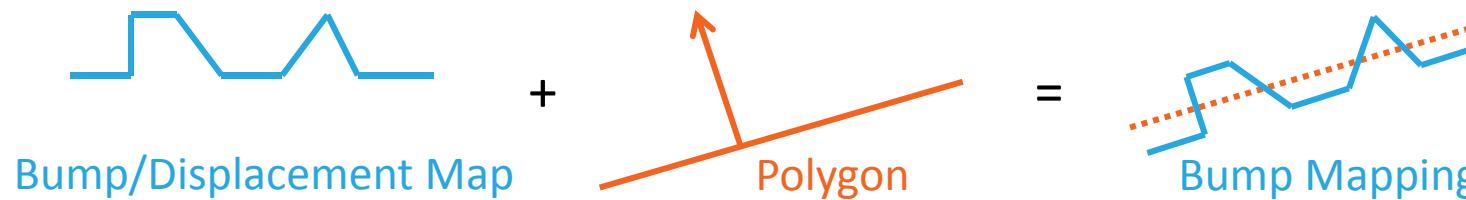
$$I = k_a \cdot I_L + k_d \cdot I_L \cdot (\mathbf{N} \cdot \mathbf{L}) + k_s \cdot I_L \cdot (\mathbf{R} \cdot \mathbf{V})^n$$



Texturierungstechniken

Bump oder Normal Mapping

- ▶ Variation der Normale einer Oberfläche berechnet aus der Veränderung einer Basisfläche durch eine **Bump Map** oder **Displacement Map**
- ▶ Fläche bleibt also geometrisch flach und **nur die Normalen variieren**
- ▶ Berechnung meist im Tangentenraum (siehe KoSys auf Torus)

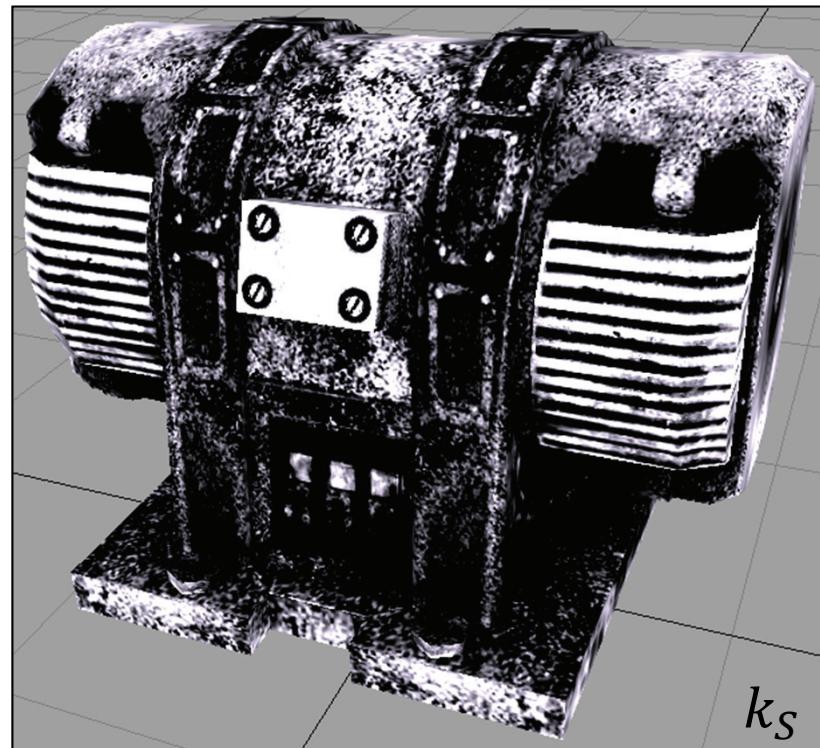


Texturierungstechniken

Gloss-Map / Gloss-Textur

- ▶ Kontrolle der Stärke und Streuung der spekularen Reflexion
- ▶ Bsp. Phong-Beleuchtungsmodell: k_s und n aus Textur

$$I = k_a \cdot I_L + k_d \cdot I_L \cdot (\mathbf{N} \cdot \mathbf{L}) + k_s \cdot I_L \cdot (\mathbf{R} \cdot \mathbf{V})^n$$

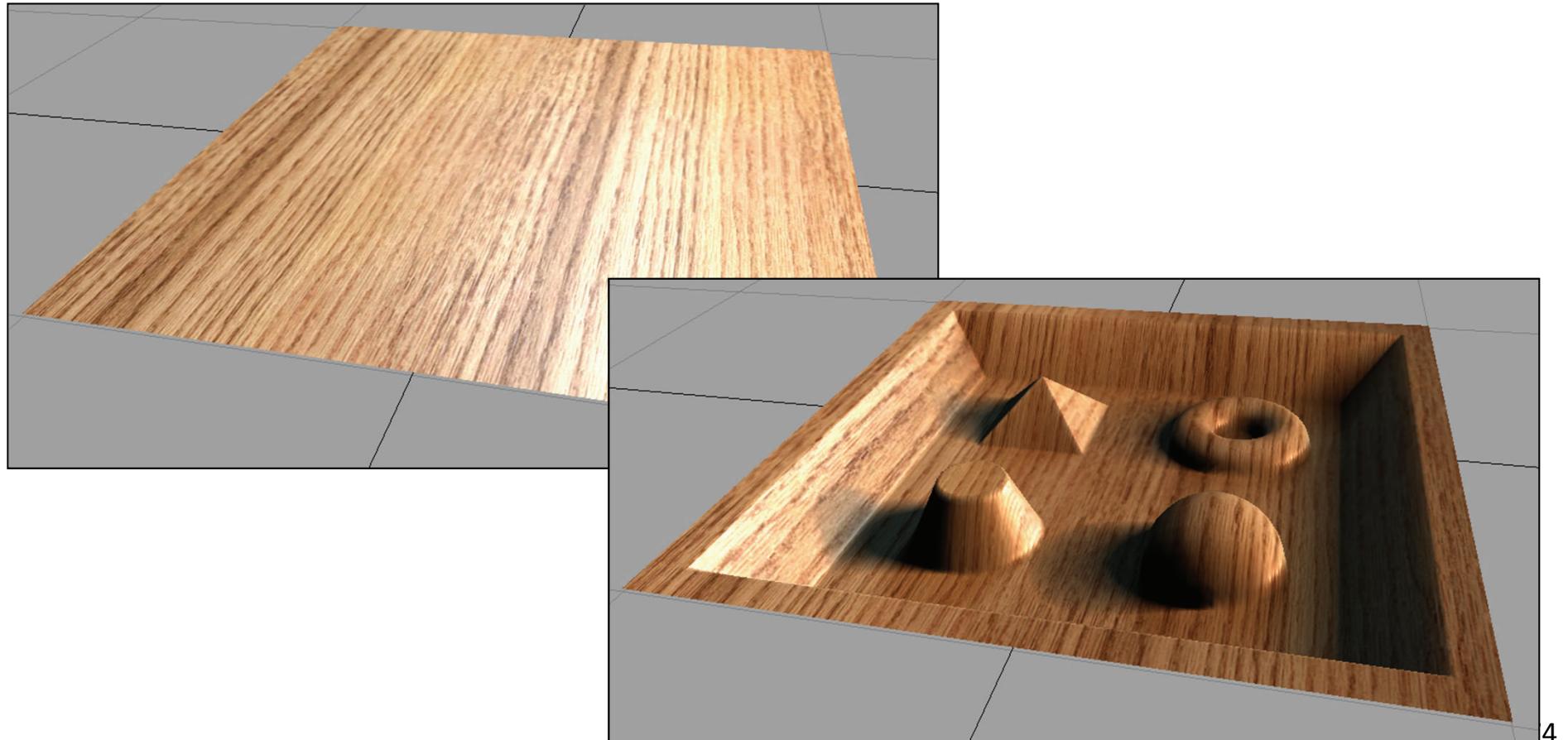


Texturierungstechniken



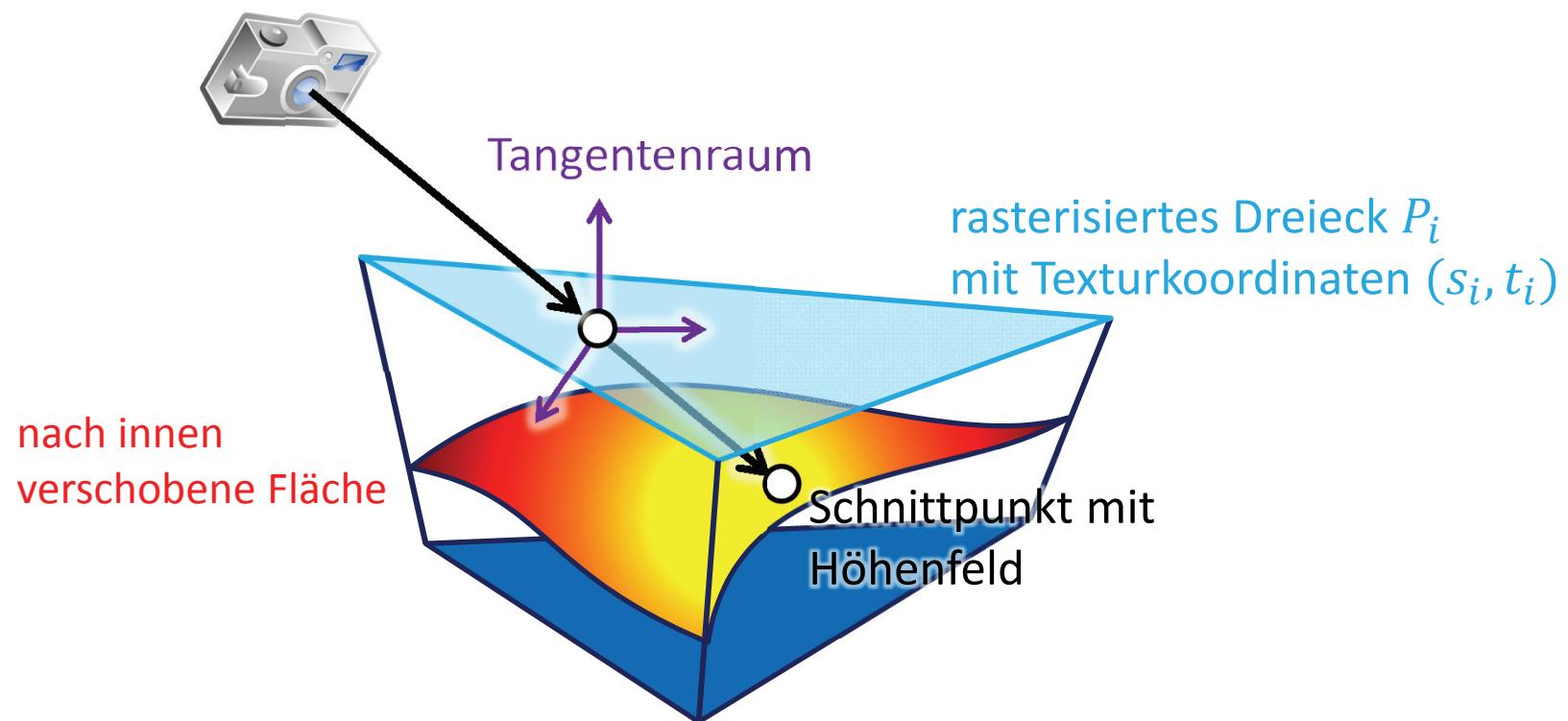
Displacement Mapping

- ▶ Verschiebung der Oberfläche und Änderung der Normale
- ▶ keine reine Änderung der Beleuchtungsberechnung
- ▶ z.B. durch Geometrie-Tesselierung und Verschiebung (GPU-unterstützt)



Inverse Displacement Mapping

- ▶ besonders schnelle Approximation geeignet um „Bodendetail“ darzustellen (Silhouetten sind meist problematisch)
- ▶ Geometrie wird nicht wirklich erzeugt: führe Schnittpunktberechnung stattdessen im Texturraum durch (siehe z.B. Parallax Occlusion Mapping)

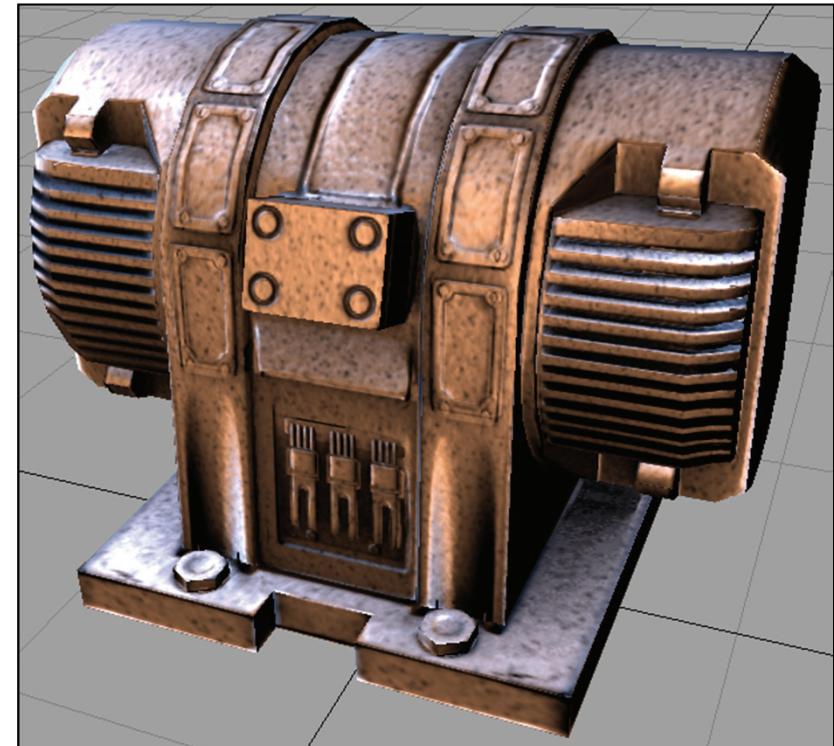
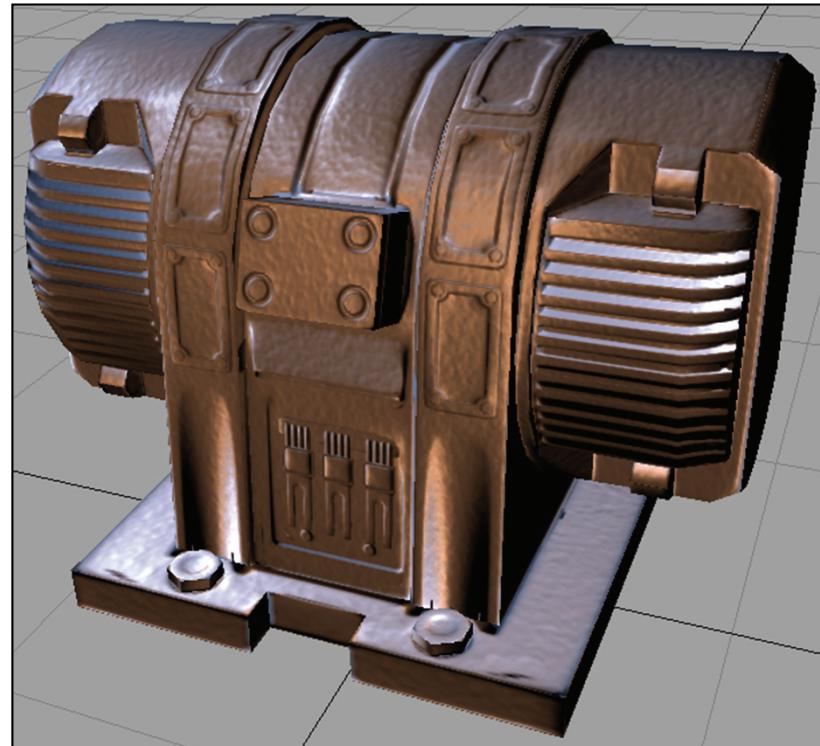


Texturierungstechniken

Ambient Occlusion

- ▶ Kontrolle des ambienten Anteils des Beleuchtungsmodells
- ▶ k_a aus Textur, meistens wird auch der diffuse Term modifiziert

$$I = k_a \cdot I_L + k_d \cdot I_L \cdot (\mathbf{N} \cdot \mathbf{L}) + k_s \cdot I_L \cdot (\mathbf{R} \cdot \mathbf{V})^n$$

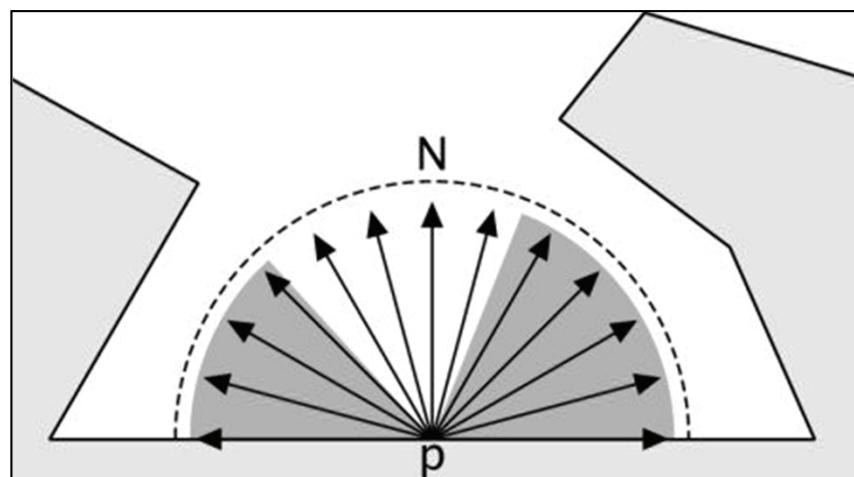


Texturierungstechniken

Ambient Occlusion

- ▶ Kontrolle des ambienten Anteils des Beleuchtungsmodells
- ▶ k_a aus Textur, meistens wird auch der diffuse Term modifiziert

$$I = k_a \cdot I_L + k_d \cdot I_L \cdot (\mathbf{N} \cdot \mathbf{L}) + k_s \cdot I_L \cdot (\mathbf{R} \cdot \mathbf{V})^n$$



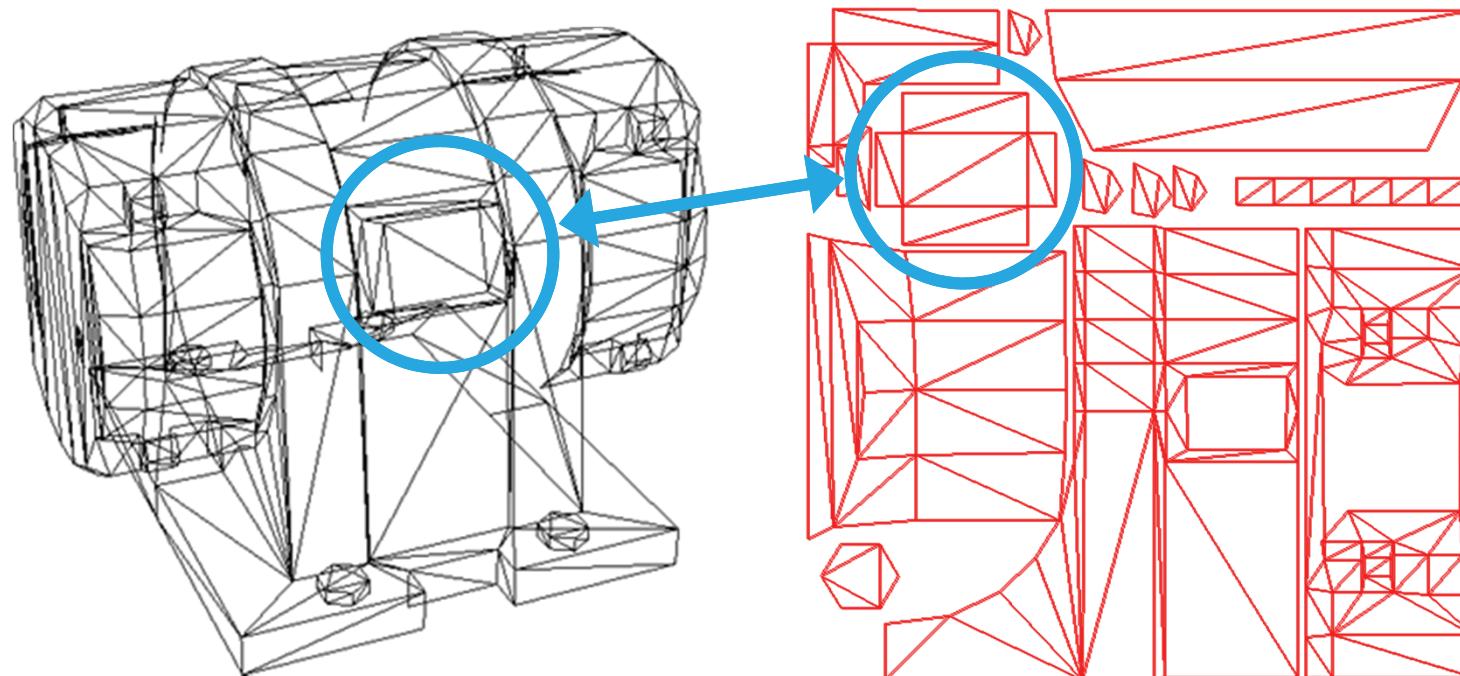
Textur-Atlas

- ▶ Textur-Atlas: spezielle (bijektive) Parametrisierung
 - ▶ jedem Oberflächenpunkt entspricht eine Stelle in der Textur
 - ▶ keine Stelle der Textur taucht an mehr als einem Oberflächenpunkt auf
(Ausnahme bei symmetrischen Objekten)
- ▶ Erstellung aufwändig per Hand oder automatisch („Zerschneiden und Ausrollen“ des Netzes, Verzerrung vs. Anzahl der Schnitte)



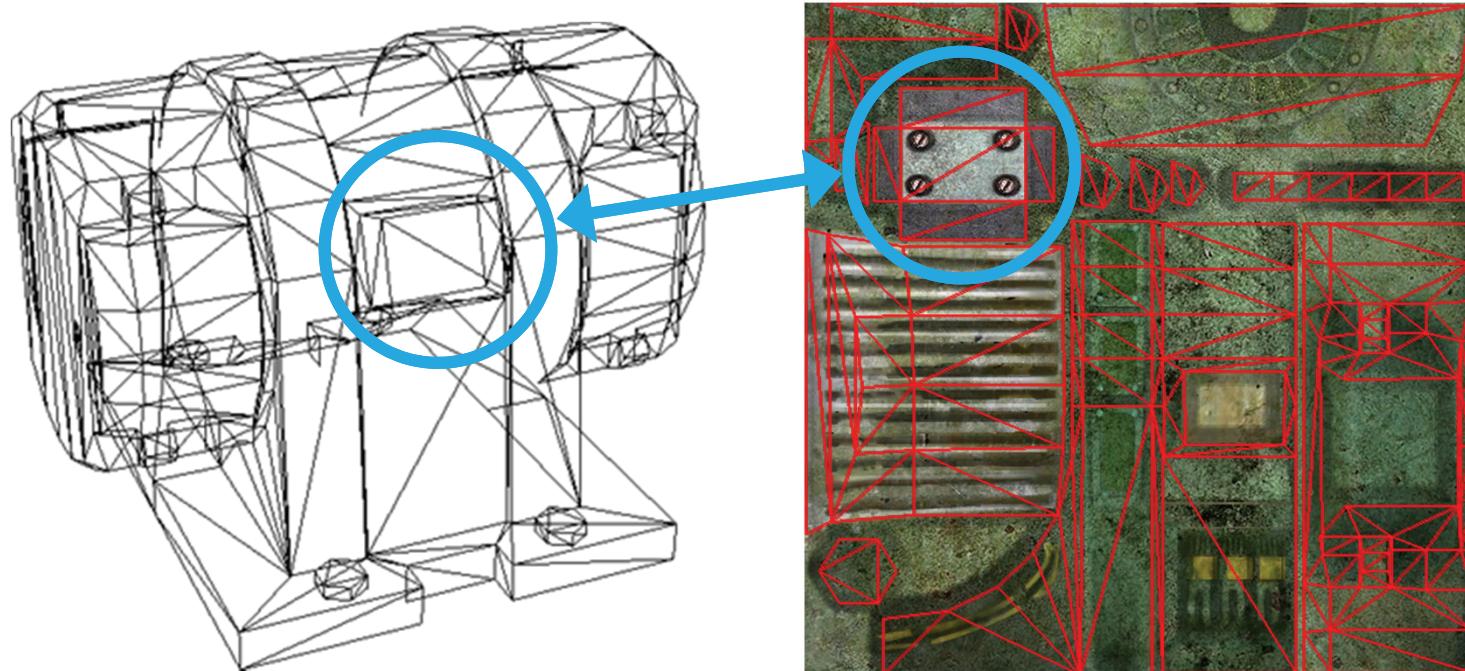
Textur-Atlas

- ▶ Textur-Atlas: spezielle (bijektive) Parametrisierung
 - ▶ jedem Oberflächenpunkt entspricht eine Stelle in der Textur
 - ▶ keine Stelle der Textur taucht an mehr als einem Oberflächenpunkt auf (Ausnahme bei symmetrischen Objekten)
- ▶ Erstellung aufwändig per Hand oder automatisch („Zerschneiden und Ausrollen“ des Netzes, Verzerrung vs. Anzahl der Schnitte)



Textur-Atlas

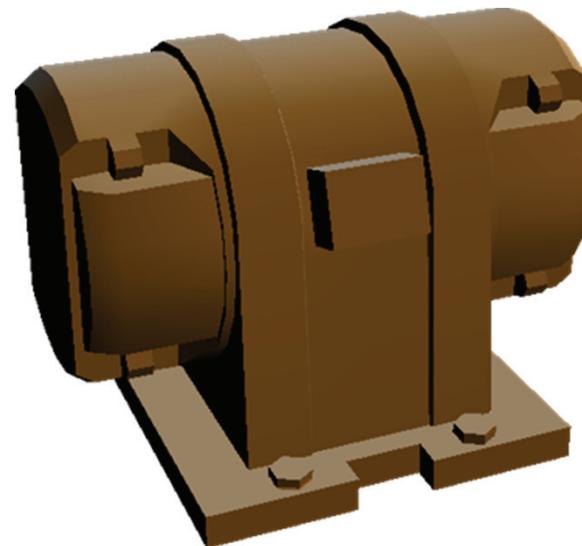
- ▶ Textur-Atlas: spezielle (bijektive) Parametrisierung
 - ▶ jedem Oberflächenpunkt entspricht eine Stelle in der Textur
 - ▶ keine Stelle der Textur taucht an mehr als einem Oberflächenpunkt auf
(Ausnahme bei symmetrischen Objekten)
- ▶ Erstellung aufwändig per Hand oder automatisch („Zerschneiden und Ausrollen“ des Netzes, Verzerrung vs. Anzahl der Schnitte)



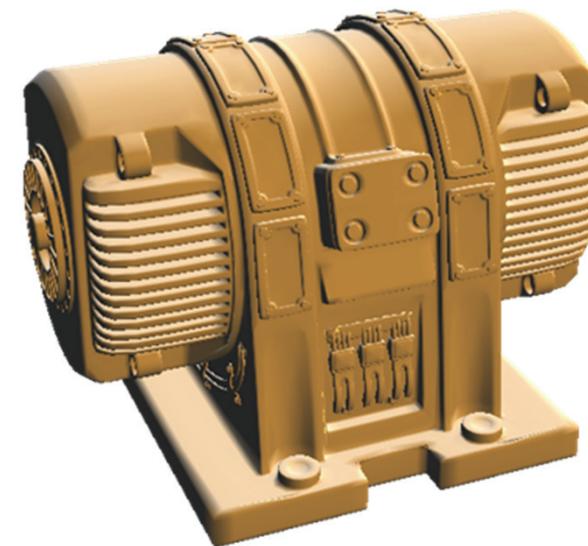
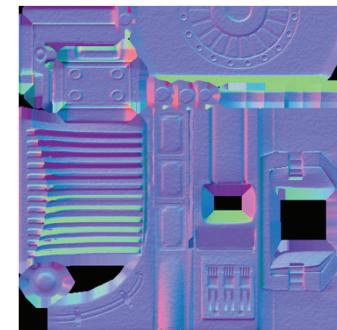
Textur-Atlas

Anwendungen

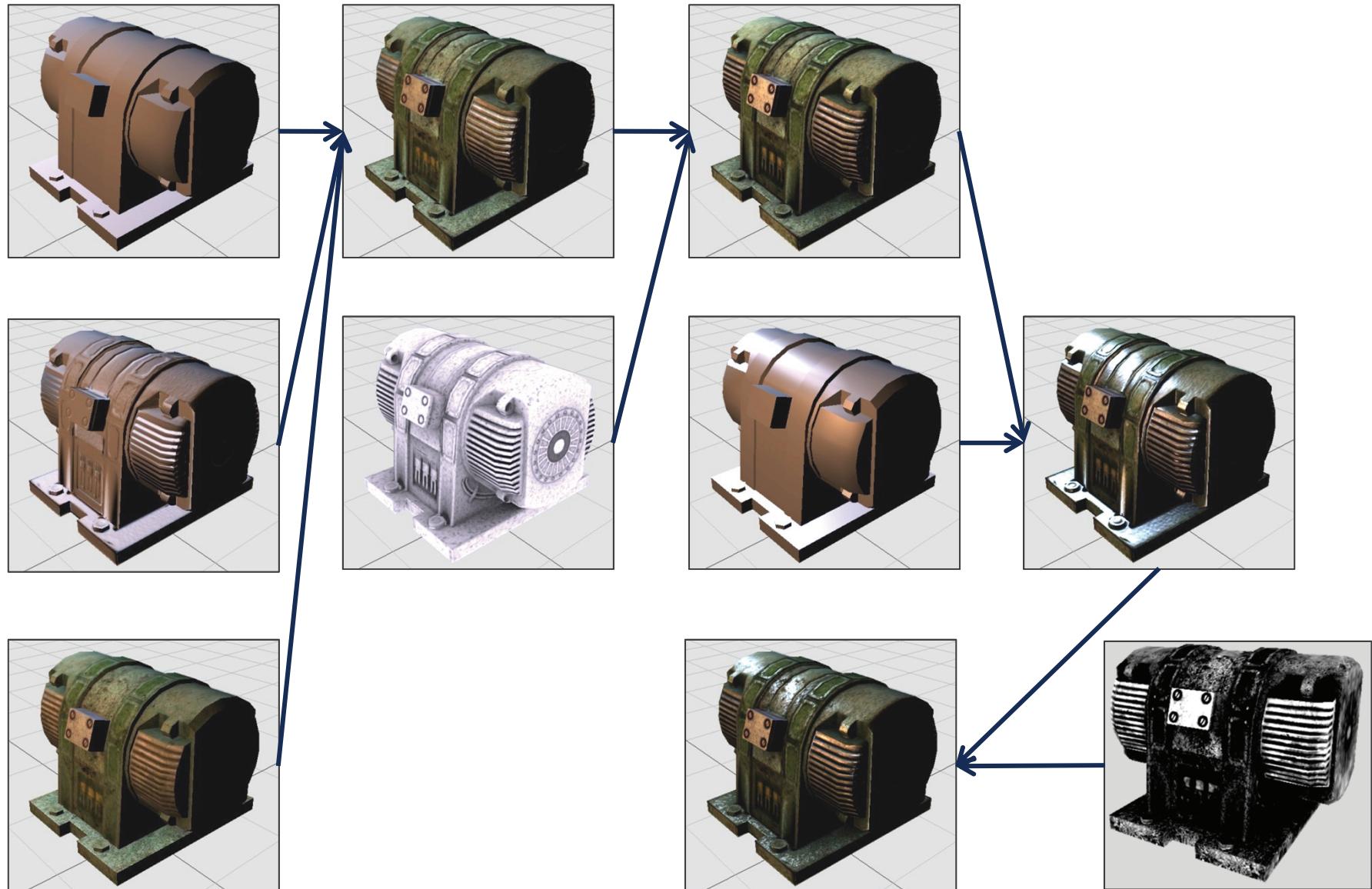
- ▶ Speichern einer Funktion/Daten auf der Oberfläche: Ambient Occlusion
- ▶ „Painting“: Erstellen der Textur direkt auf dem Objekt
- ▶ Berechnung von Normal-Maps (für Bump/Normal-Mapping) aus fein aufgelösten Dreiecksnetzen



Normal-Map



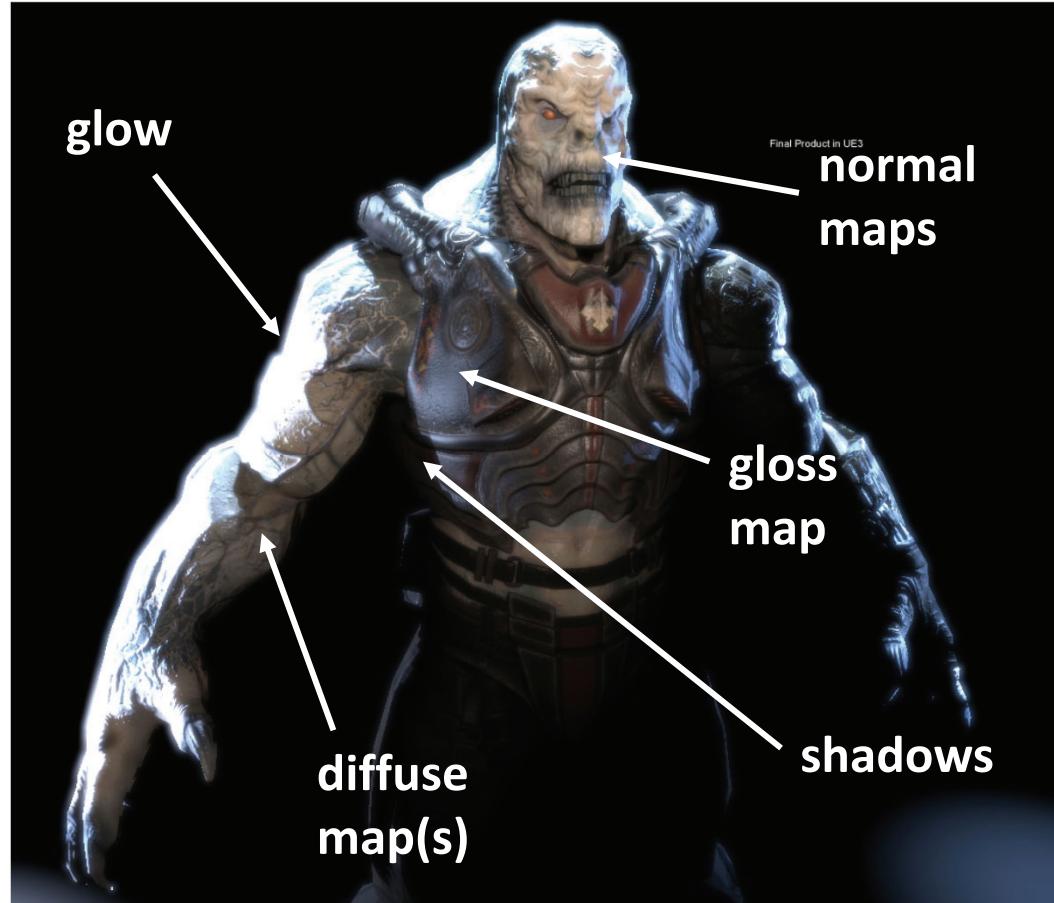
Multitexturing Beispiel



Texturen und Beleuchtungstechniken

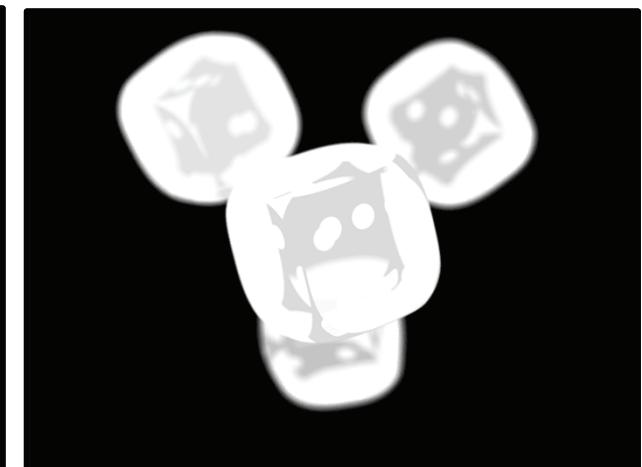
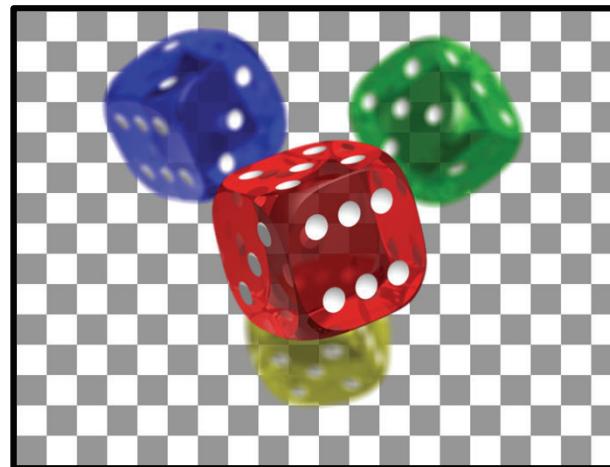


► Bild: Unreal Engine

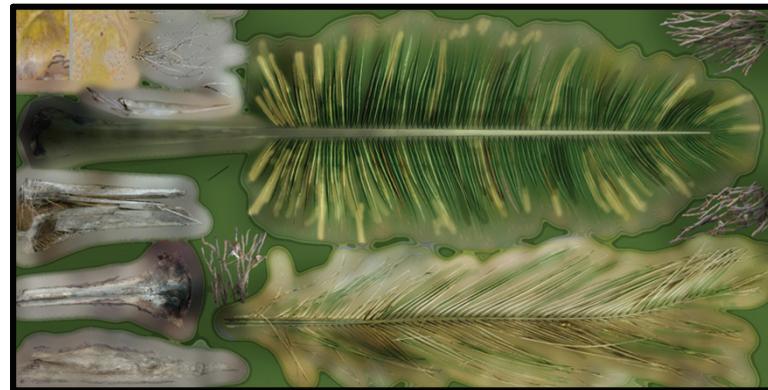
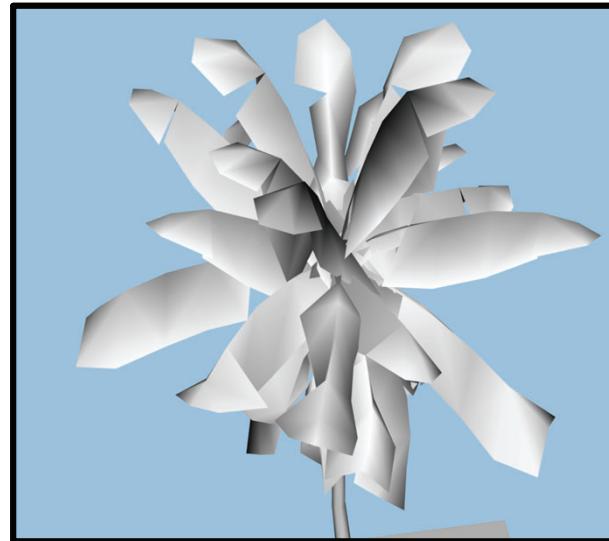


Transparenz und Alpha-Test

- ▶ Rasterbilder (und v. a. Texturen) werden oft mit 32 Bit/Pixel gespeichert
 - ▶ 24 Bit Farbinformation und
 - ▶ 8 Bit Alpha-Kanal (α = Opazität, Gegenteil von Transparenz)
 - ▶ RGBA Format (PNG, TIFF, TGA, ...)



Transparenz und Alpha-Test



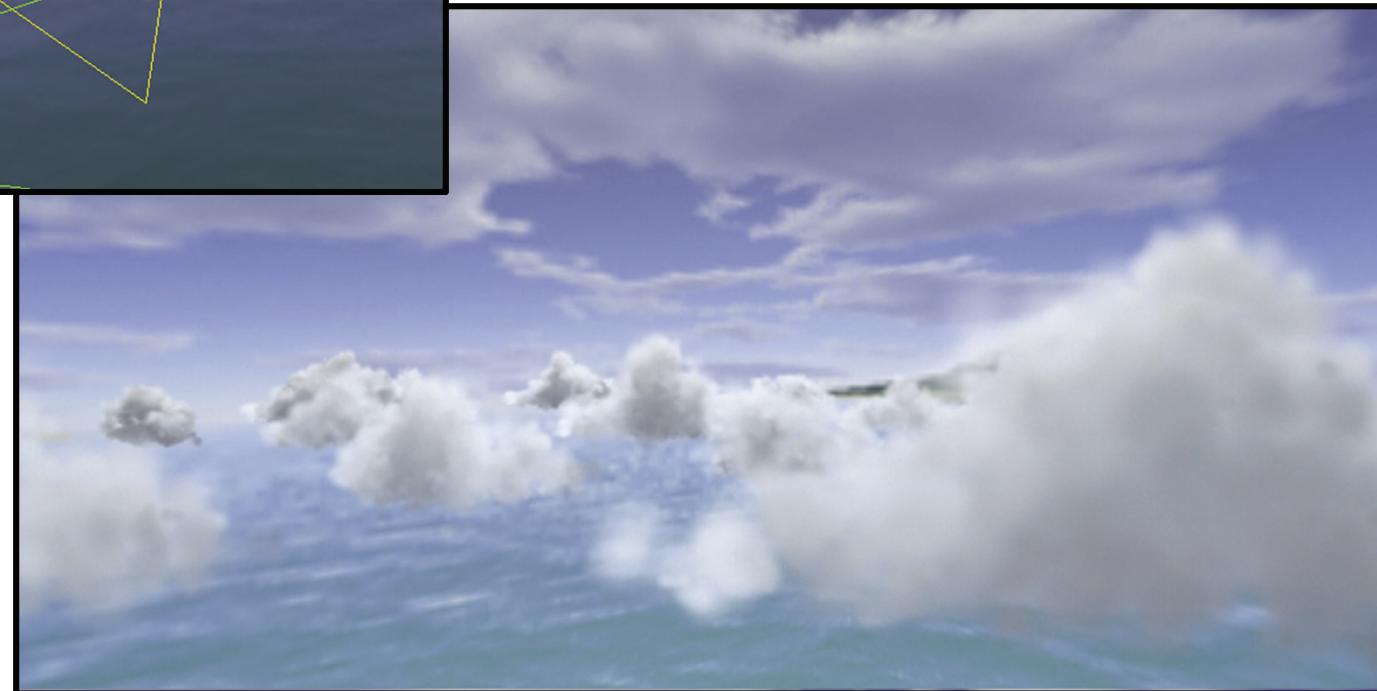
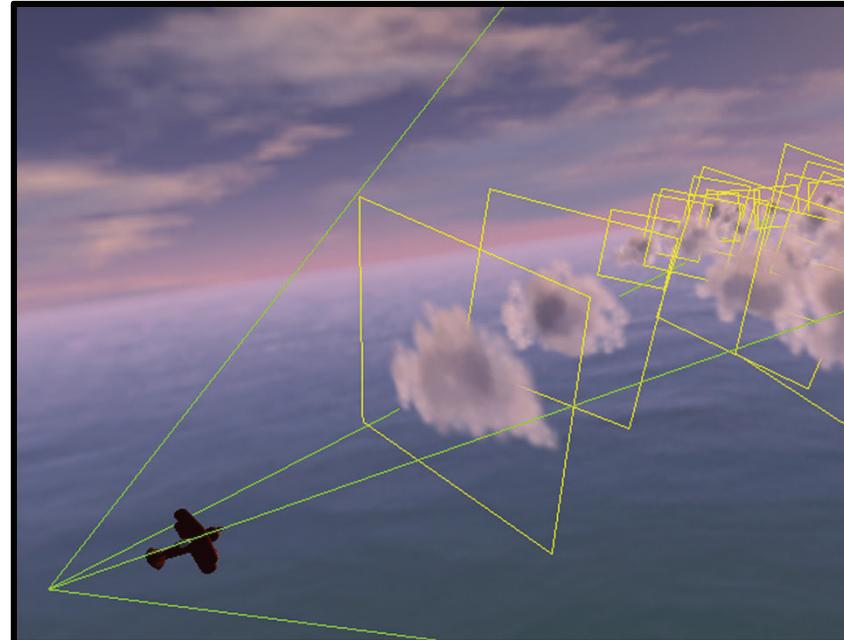
24 Bit Farbinformation



8 Bit Alpha-Kanal
schwarz = transparent

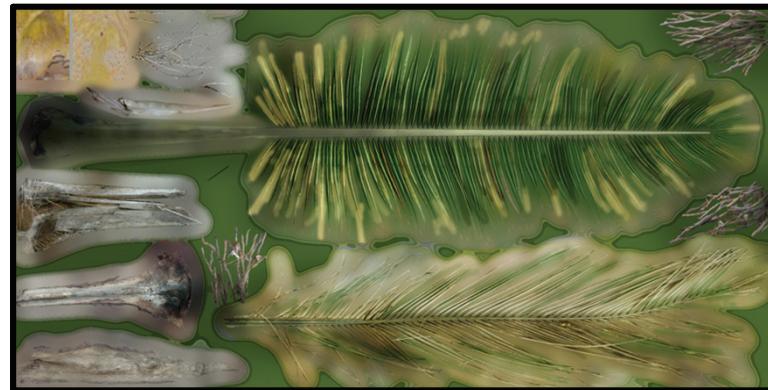
Impostors

- ▶ Impostor: texturiertes, senkrecht zur Kamera stehendes, Polygon



Transparenz und Alpha-Test

- ▶ **(Semi-)Transparenz:** verwende Alpha-Kanal um die Transparenz (also z.B. k_t im Ray Tracing) der Oberfläche an einem Punkt zu bestimmen (**Alpha oder Opacity Map**)
- ▶ **Alpha-Test:** verwerfe Objekt-/Schnittpunkt, wenn $\alpha < threshold$ (**Alpha oder Opacity Mask**)



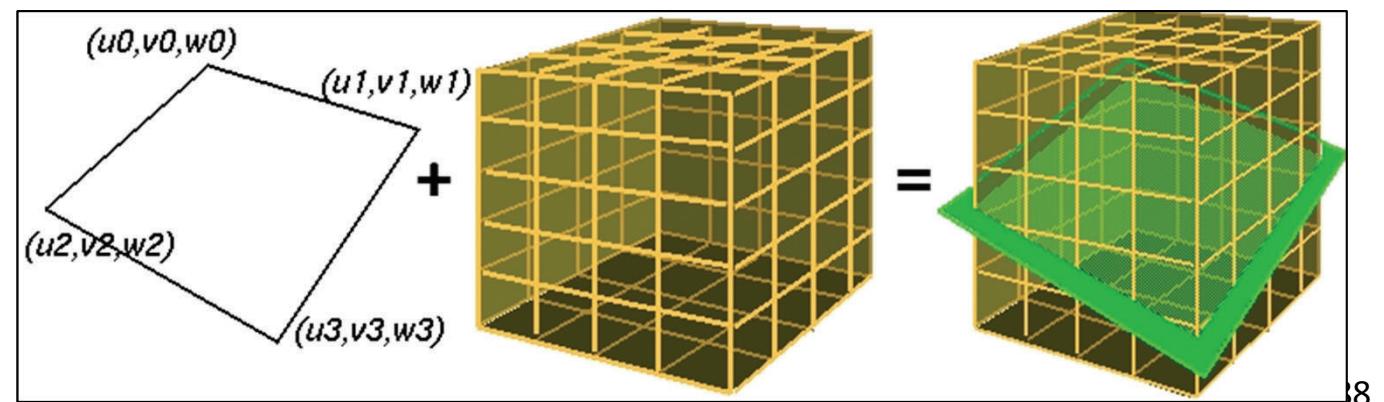
24 Bit Farbinformation



8 Bit Alpha-Kanal
schwarz = transparent

3D-Texturen für Oberflächen

- ▶ Probleme von 2D-Texturen
 - ▶ Tapeten-/Furnier-Effekt („oberflächlich“)
 - ▶ Texturkoord. für komplexe Objekte, Verzerrung bei großer Flächenkrümmung
- ▶ Solid-Textures (3D-Texturen), z.B. Holzblock, Marmorblock
 - ▶ „Herausschneiden einer Skulptur“ durch Zuweisung von 3D Texturkoordinaten an Vertizes
 - ▶ Vorteil: Parametrisierung fällt praktisch weg
 - ▶ Nachteil: großer Speicherplatzbedarf der Textur (RGB $512^3 \rightarrow 384\text{MB}$)
 - ▶ woher bekommt man eine 3D-Textur? Aufnahme einzelner Schichtbilder?

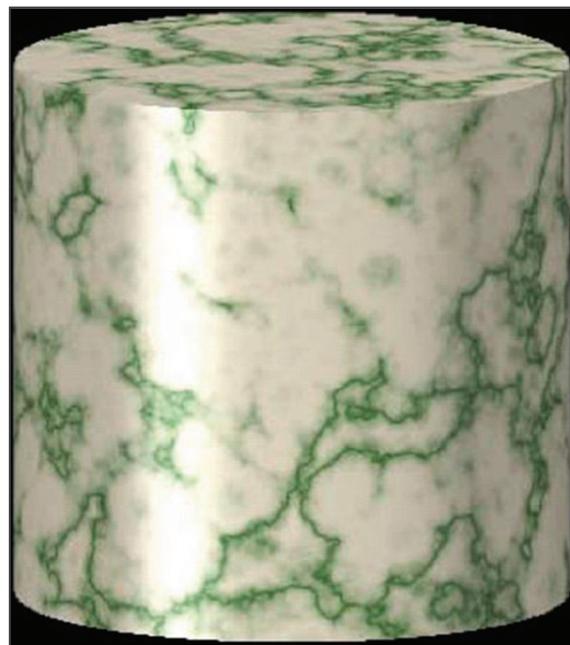
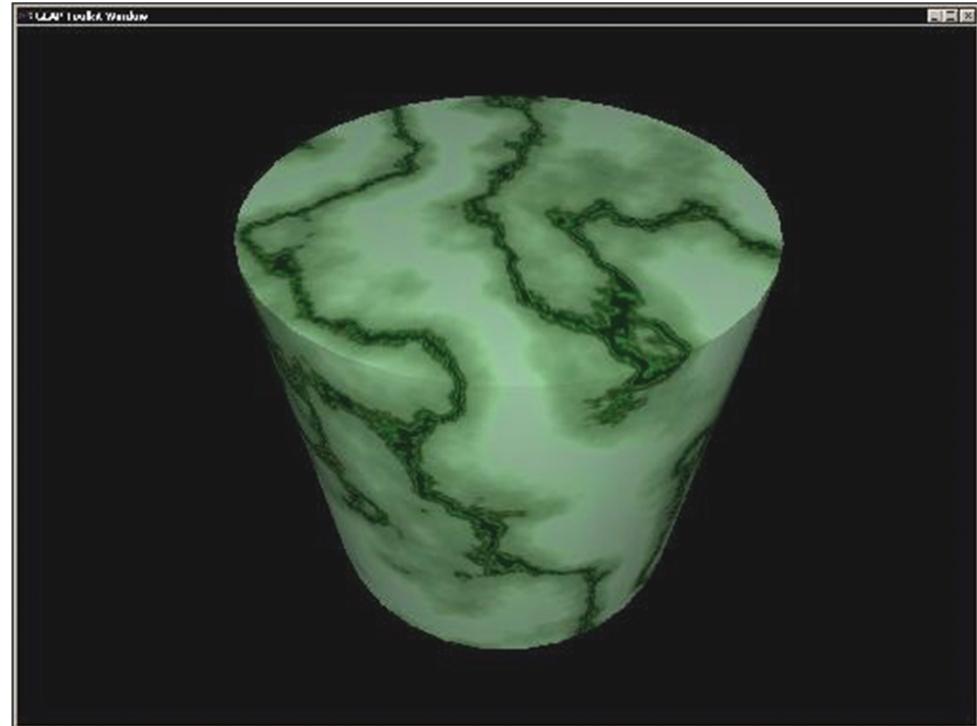
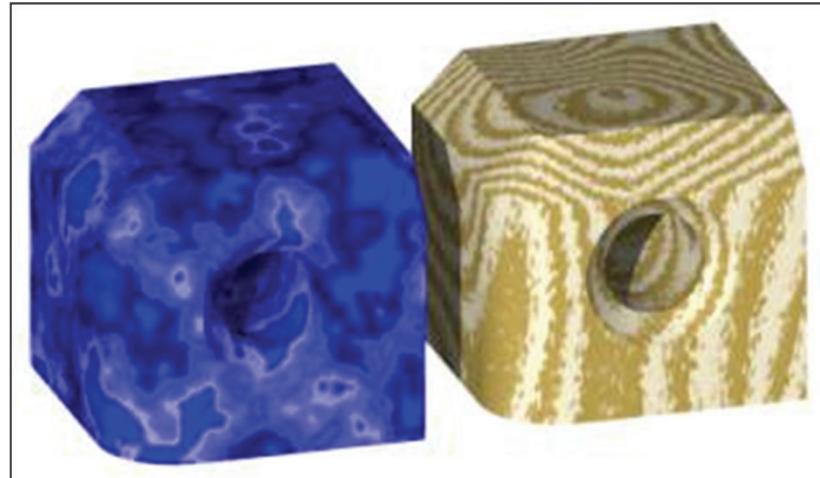


3D-Texturen für Oberflächen

- ▶ Gewinnung von 3D-Texturen ist schwierig
→ werden oft durch prozedurale Beschreibungen generiert
- ▶ $f(\mathbf{x}) = \mathbf{c}$ mit $\mathbf{x} \in \mathbb{R}^3$, \mathbf{c} ist Farbe

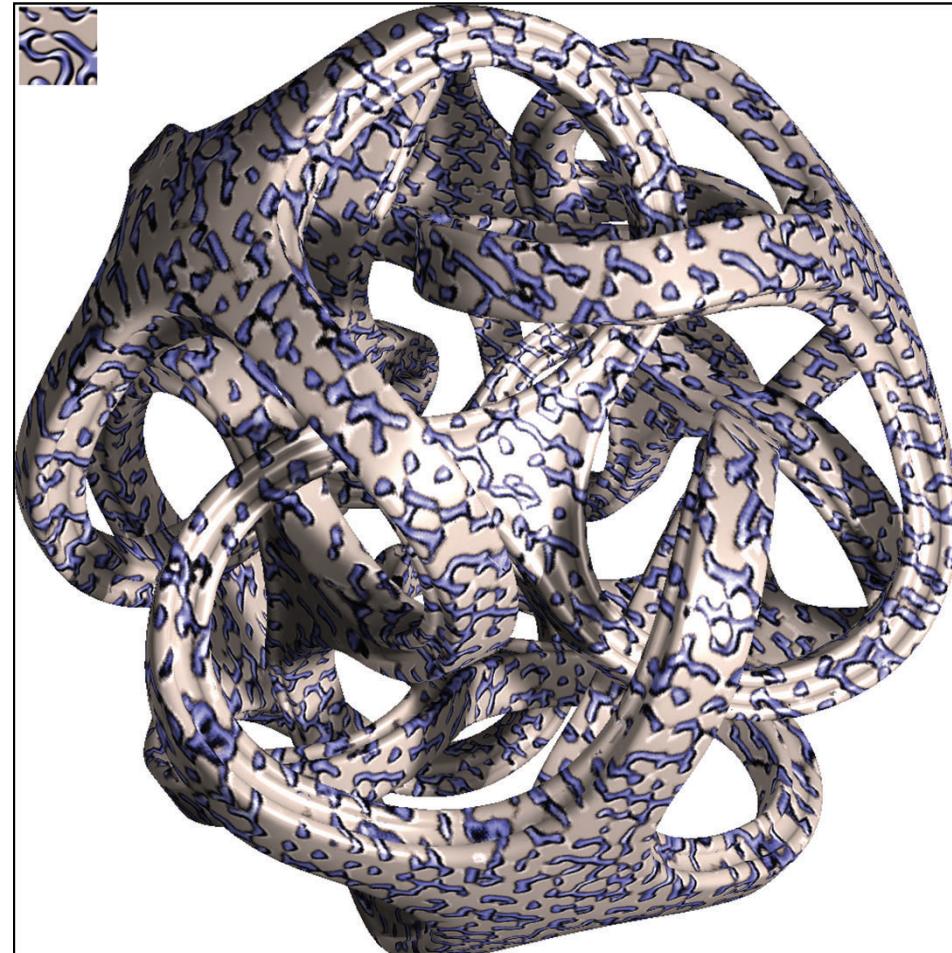


Beispiel: Solid-Textures



Lazy Solid Textures

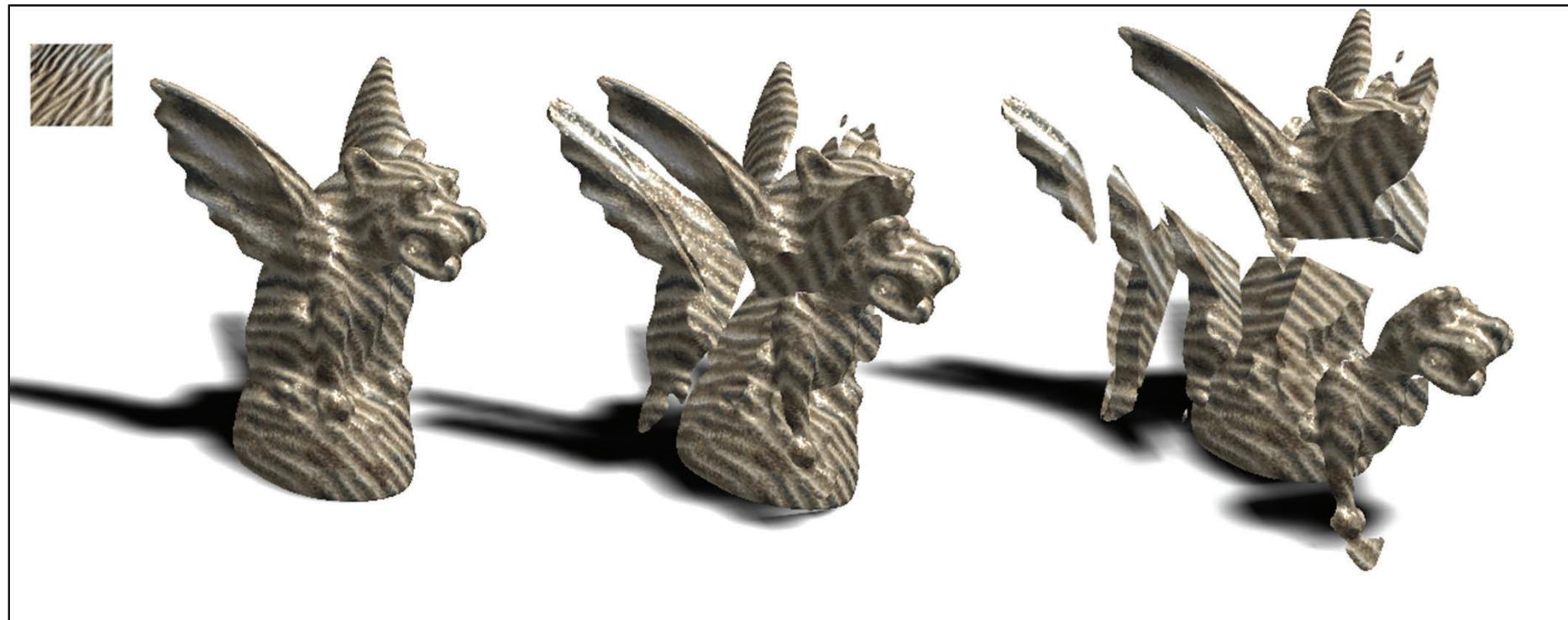
- ▶ Verfahren zur Berechnung von 3D Texturen aus 2D Texturen
(<http://www-sop.inria.fr/reves/Basilic/2008/DLTD08/>)



Lazy Solid Textures



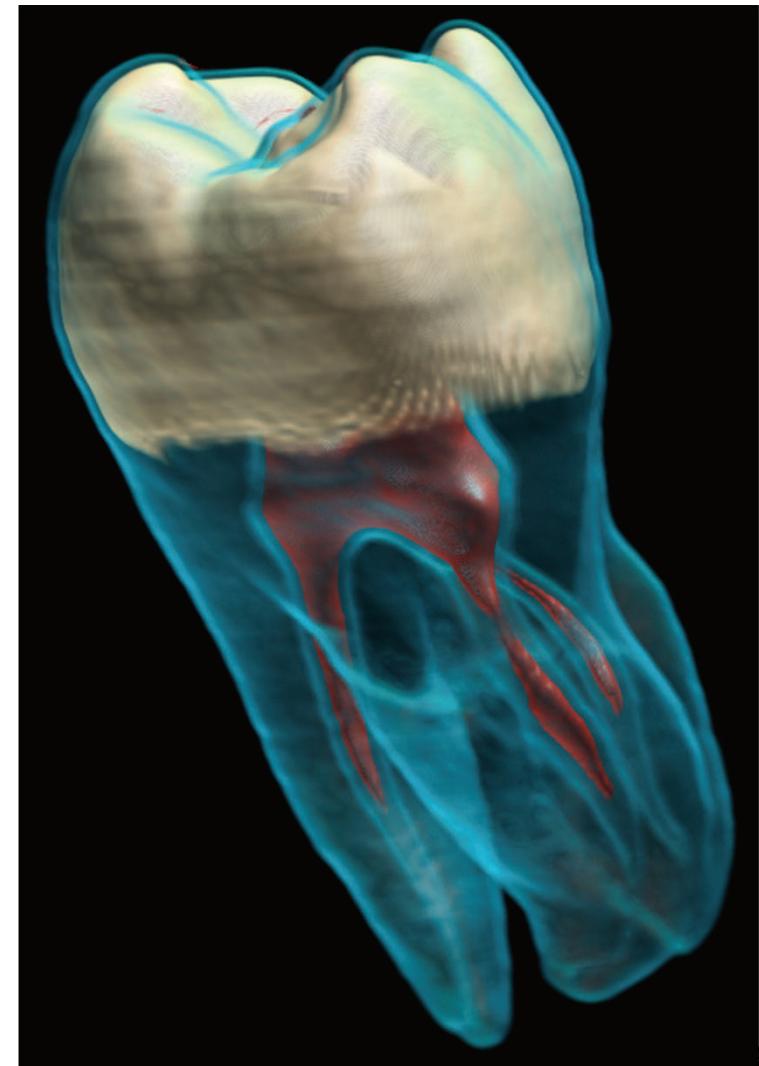
- ▶ Verfahren zur Berechnung von 3D Texturen aus 2D Texturen
(<http://www-sop.inria.fr/reves/Basilic/2008/DLTD08/>)



3D-Texturen und Volumenvisualisierung



- ▶ Ausgangsdaten sind 3D
 - ▶ Röntgenabsorption im Körper/Material (Computertomographie)
 - ▶ Feuchtigkeit in Atmosphäre
 - ▶ Dichteverteilung im Erdinneren
- ▶ Daten oft auf uniformen 3D-Gitter
 - ▶ speichern/verwenden als 3D-Textur
 - ▶ Millionen von Zellen (Voxel)
- ▶ Problem: Verdeckung



Reflection / Environment-Mapping



Motivation

- ▶ Darstellung reflektierender Objekte mit Spiegelung (realer aufgenommener) Umgebungen ohne geometrische Repräsentation
- ▶ Approximation der Reflexion ohne Ray Tracing, d.h. ohne einen Reflexionsstrahl wirklich zu verfolgen
- ▶ Grundidee: speichere Bild der Umgebung in einer Textur



Reflection / Environment-Mapping



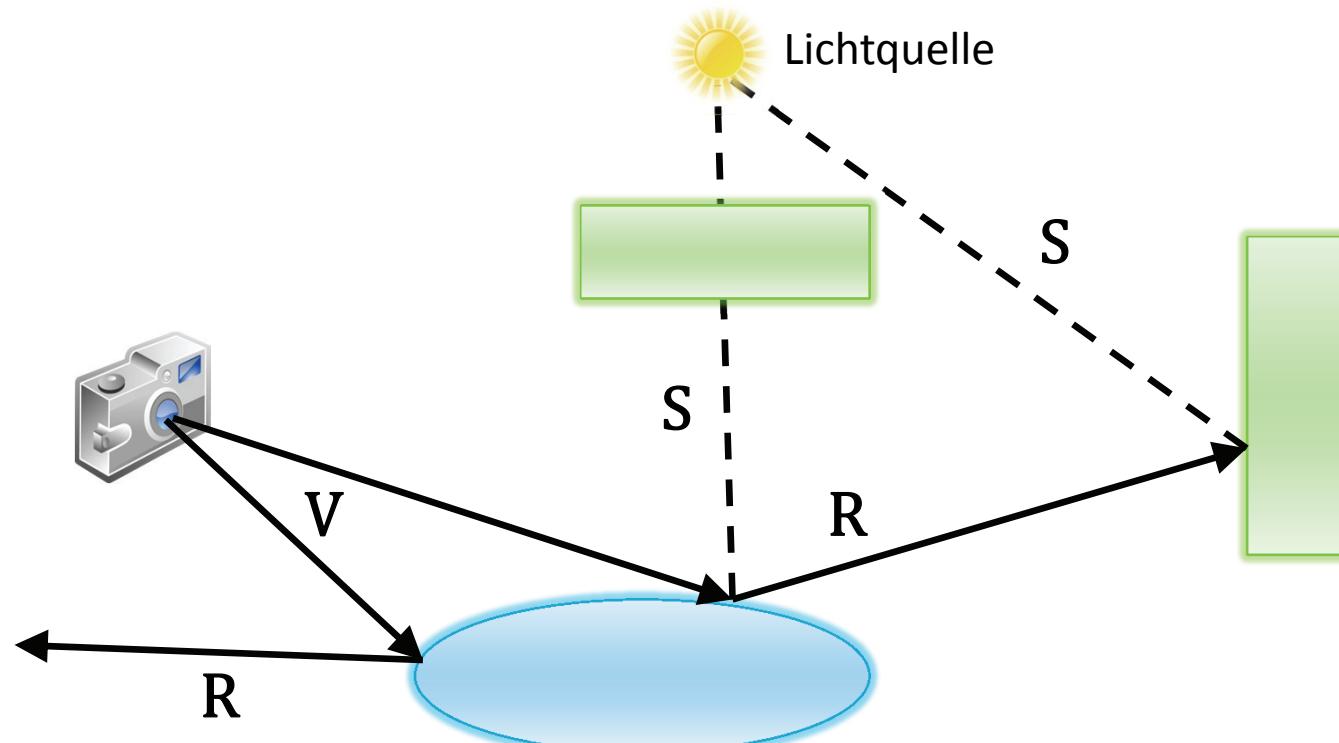
Motivation

- ▶ eine der ersten Anwendungen: Terminator 2



Reflection / Environment-Mapping

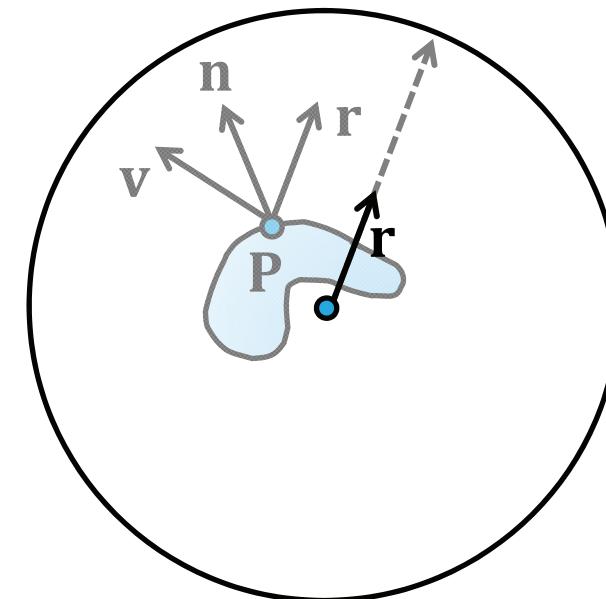
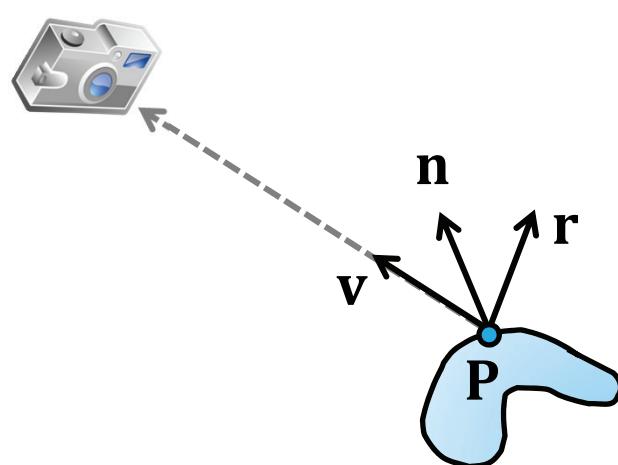
- ▶ Environment Mapping wird verbreitet mit Grafik-Hardware eingesetzt
- ▶ beim Ray Tracing: trifft ein Strahl kein Objekt, dann wird das ankommende Licht aus der Environment Map ausgelesen



Strahl trifft kein weiteres Objekt
→ ankommendes Licht aus der Umgebung

Environment-Mapping

- ▶ Reflection/Environment-Map können wir uns als Textur auf einer virtuelle Kugel um ein Objekt/die Szene vorstellen
- ▶ **grundlegende vereinfachende Annahme:**
nur die Richtung r des reflektierten Strahls wird beim Environment Mapping verwendet, der Ausgangspunkt P des Strahls wird ignoriert!
 - ▶ akzeptabel, wenn die Umgebung weit weg vom spiegelnden Objekt ist
 - ▶ Zugriff: konvertiere Reflexionsrichtung r in Texturkoordinaten

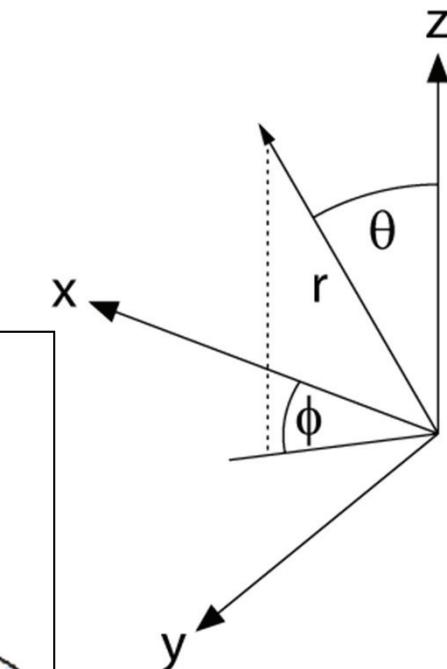


Environment Map auf der
Innenfläche einer virtuellen Kugel

Environment-Mapping

Latitude/Longitude-Maps (eine mögliche Parametrisierung von EnvMaps)

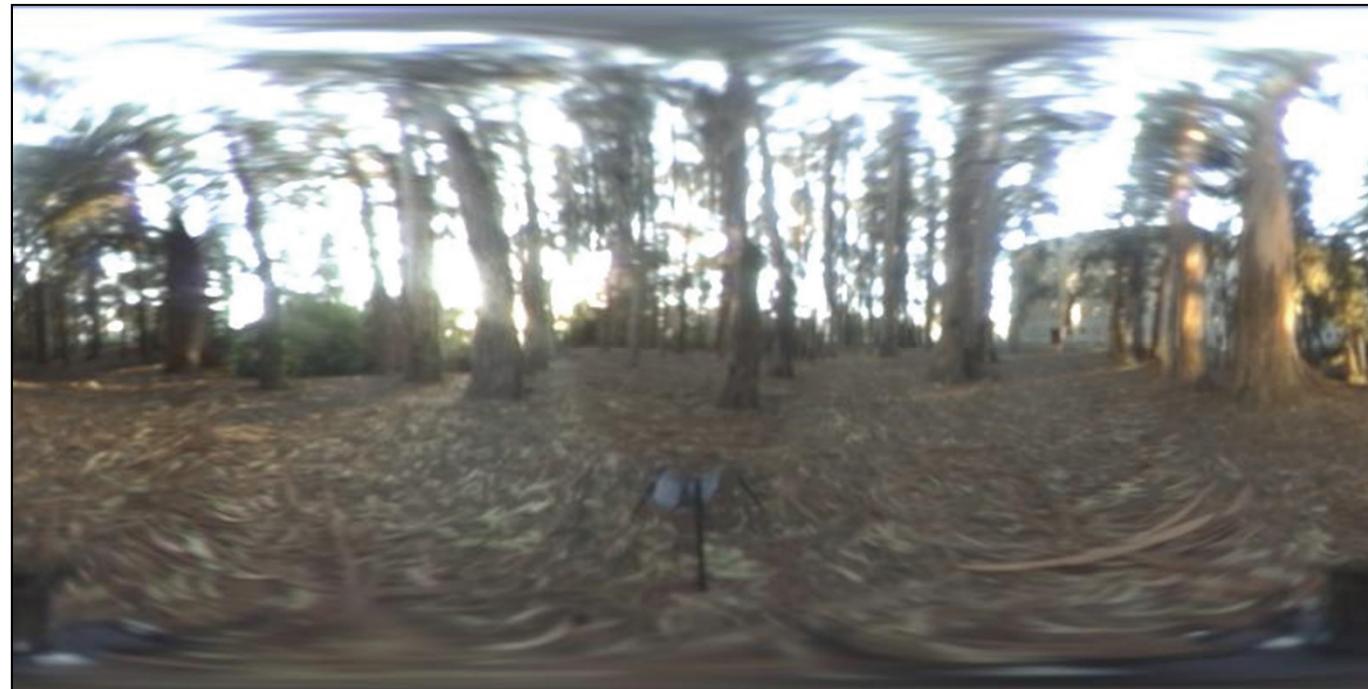
- ▶ Parametrisierung über
Polarwinkel $\theta \in [0.. \pi]$ zur z-Achse, Azimutalwinkel $\phi \in [0.. 2\pi]$
- ▶ Berechnung der Winkel aus Richtungsvektor $\mathbf{r} = (r_x, r_y, r_z)$ mit $|\mathbf{r}| = 1$
 - ▶ $\theta = \arccos(r_z)$ bzw. $\phi = \text{atan2}(r_y, r_x)$, $s = \theta/\pi$ und $t = \phi/2\pi$



Environment-Mapping

Latitude/Longitude-Maps

- ▶ vergleichsweise teure Berechnung der Texturkoordinaten aus r
- ▶ sehr ungleichmäßige Abtastung an den Polen: viele Texel für wenige Richtungen (vgl. oberer/untere Rand mit Äquator)

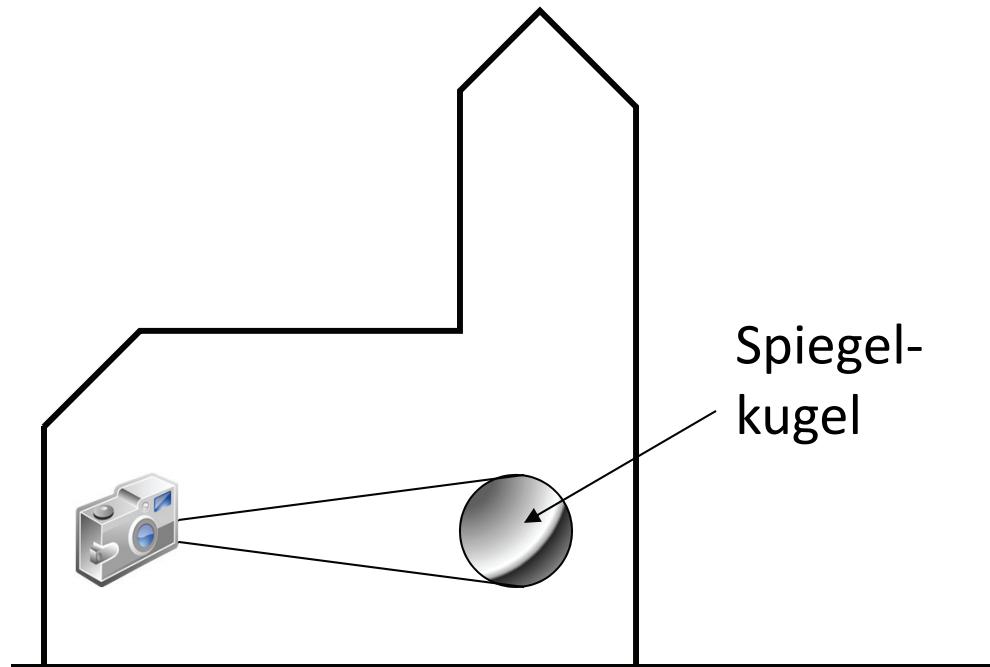


- ▶ gibt es bessere Parametrisierung? Welche kann man direkt aufnehmen?

Sphere-Mapping

Grundidee

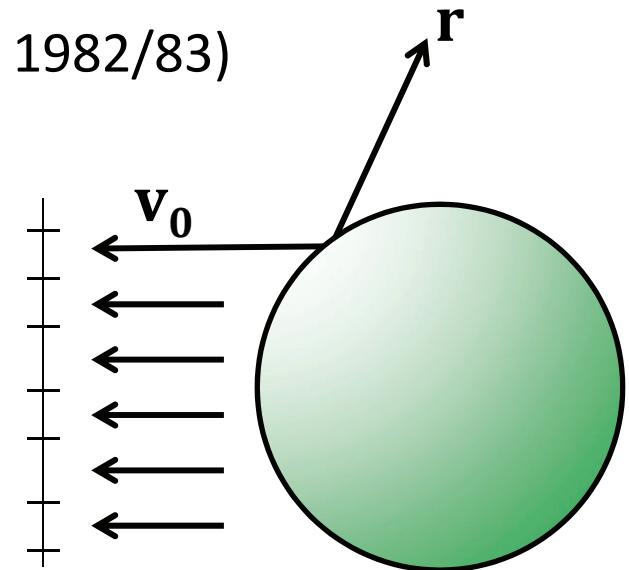
- Bild der Umgebung wird (wie bei Latitude/Longitude-Maps) ebenfalls in einer 2D-Textur gespeichert
- aber: Aufnahme der EnvMap soll mit einem Fotoapparat möglich sein



Sphere-Mapping

Beispiel: Erzeugung

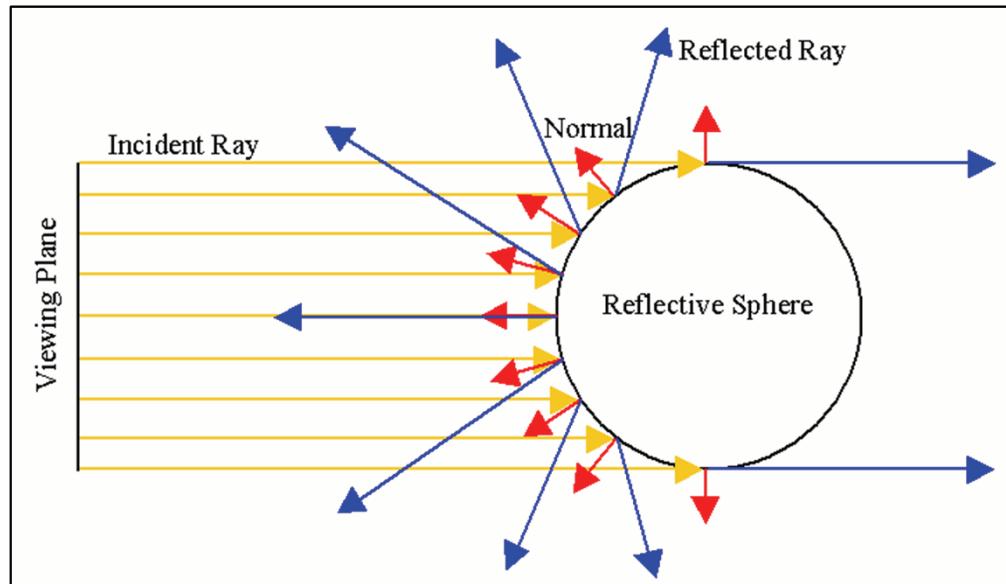
- ▶ Erzeugung der Sphere-Maps (Originalarbeit von 1982/83) durch Aufnahme einer reflektierenden Kugel
- ▶ idealisierte Annahme: alle Primärstrahlen bei der Aufnahme sind parallel



Sphere-Mapping

Eigenschaften

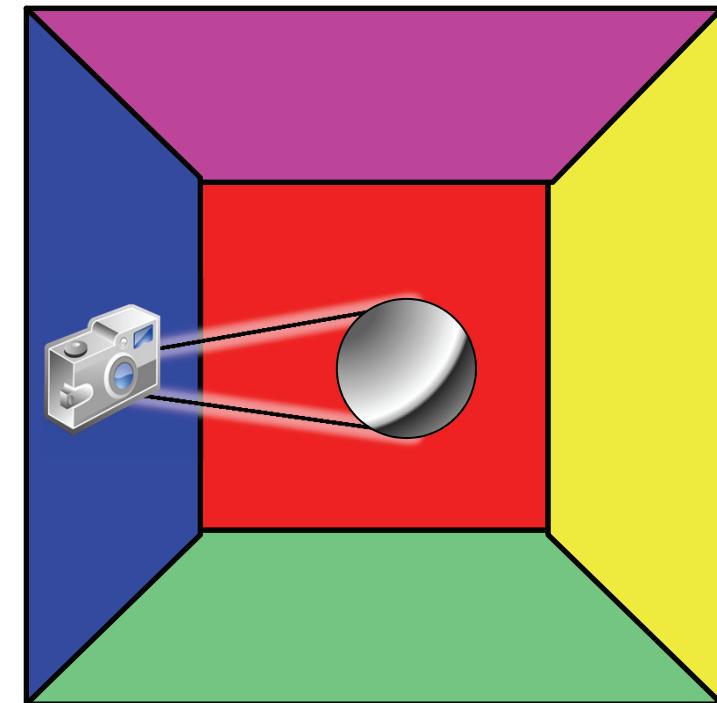
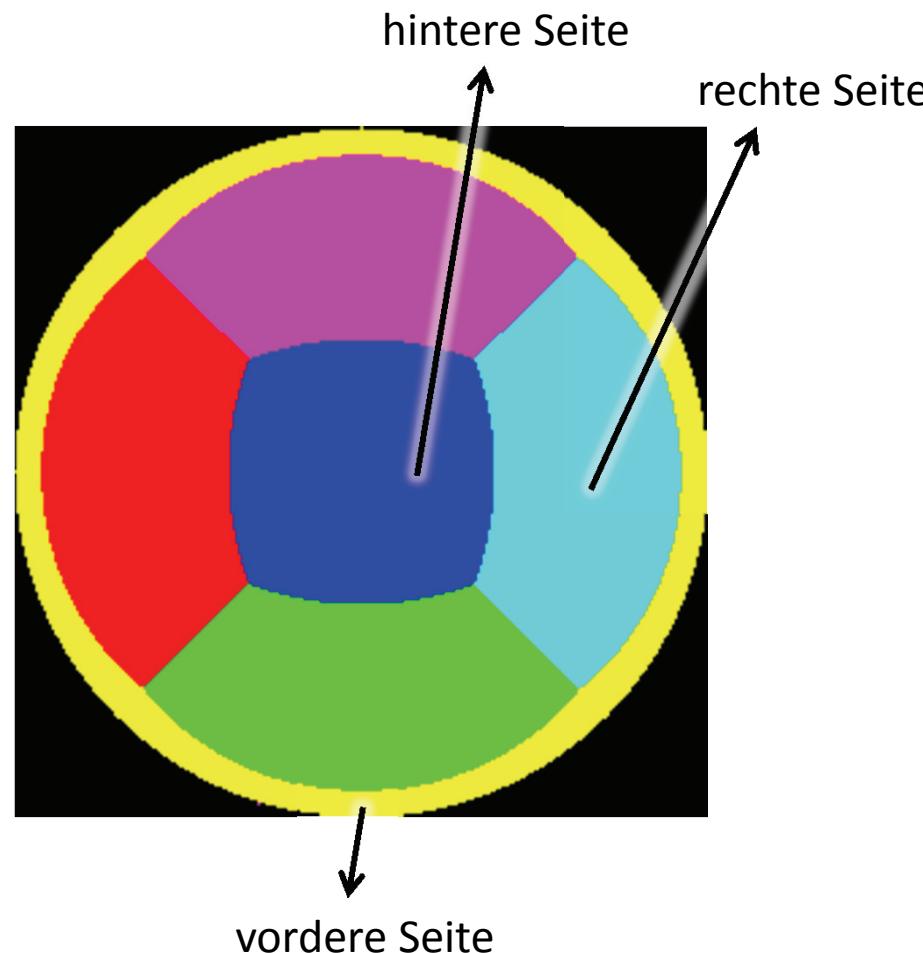
- ▶ fotografiere kleine Spiegelkugel (zentriert im Bild) mit einem Teleobjektiv: „Kamerastrahlen“ sind beinahe parallel
- ▶ Bild auf der Spiegelkugel wird zur Sphere-Map
- ▶ blickpunktabhängig: um die Textur für eine Richtung r auszulesen, muss man die Aufnahmerichtung v_0 kennen



Sphere-Mapping

Eigenschaften

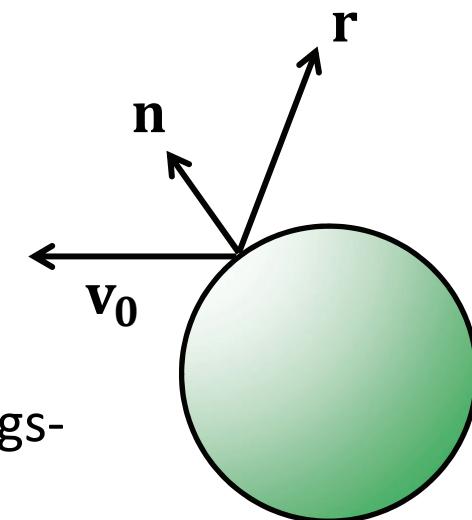
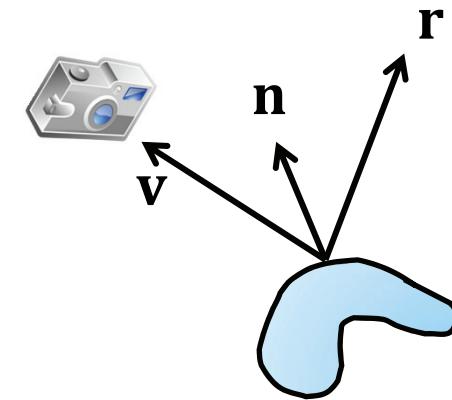
- Bild einer virtuellen Spiegelkugel in einem bunten Würfel: alle 6 Seiten sind in der Spiegelung zu erkennen



Sphere-Mapping

Berechnung der Texturkoordinaten

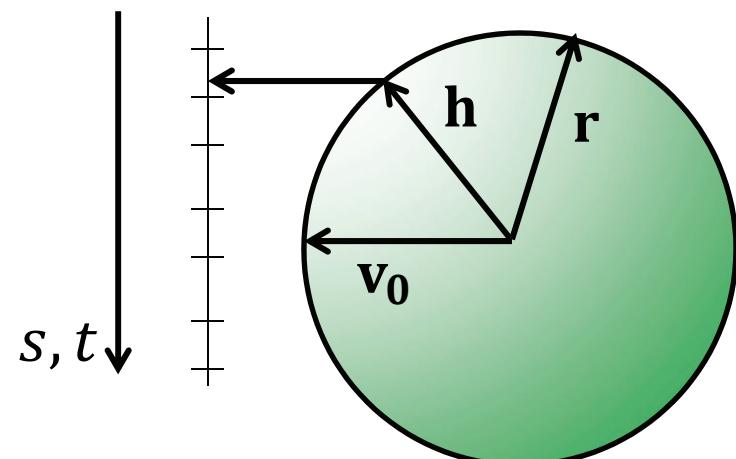
- ▶ gegeben (Annahme: alle Vektoren normalisiert)
 - ▶ aktuelle Betrachterrichtung v
(bzw. Richtung wohin das Licht reflektiert wird)
 - ▶ Oberflächennormale n
 - ▶ Reflexionsrichtung $r = (r_x, r_y, r_z)$
 - ▶ Richtung v_0 aus der die Sphere-Map aufgenommen wurde
- ▶ gesucht: welcher Punkt auf der Spiegelkugel hat die Reflexionsrichtung r ?
- ▶ Lösung:
der Punkt, dessen Normale n
genau zwischen r und v_0 liegt
 - ▶ wir haben aus Richtung v_0 aufgenommen
und bei perfekter Spiegelung war die Spiegelungs-
richtung r genau dort zu sehen



Sphere-Mapping

Berechnung der Texturkoordinaten

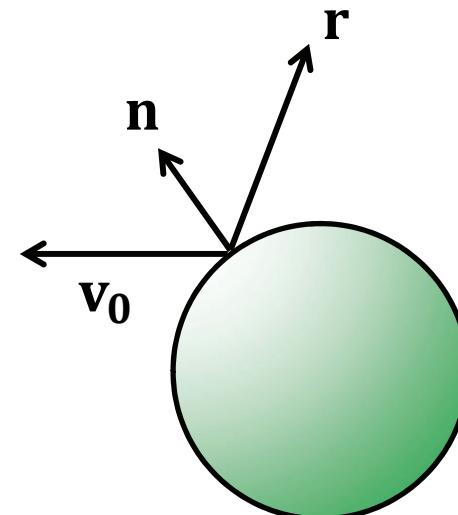
- ▶ wir suchen also den Punkt auf der Kugeloberfläche, dessen Normale \mathbf{n} genau zwischen \mathbf{r} und \mathbf{v}_0 liegt, weil $\mathbf{r} = 2(\mathbf{v} \cdot \mathbf{n})\mathbf{n} - \mathbf{v}$
- ▶ auf der (Einheits-)Kugel gilt: ein Punkt $\mathbf{p} = (x, y, z)$ hat Normale (x, y, z)
- ▶ der Half-Way Vector $\mathbf{h} = (\mathbf{r} + \mathbf{v}_0)/|\mathbf{r} + \mathbf{v}_0|$ liegt genau zw. \mathbf{r} und \mathbf{v}_0
 - ▶ am Punkt \mathbf{h} ist die Normale $\mathbf{n} = \mathbf{h}$, es ist also der gesuchte Punkt
- ▶ wir erhalten die Texturkoordinaten durch orthogonale Projektion (= weglassen der z -Komponente) und Abbildung auf $[0; 1]^2$ mit
 - $s = (h_x + 1)/2$
 - $t = (h_y + 1)/2$
- ▶ beachte die Abhängigkeit von der Aufnahmerichtung \mathbf{v}_0 in der Berechnung von \mathbf{h}



Sphere-Mapping

Probleme bzw. Nachteile

- ▶ wieder ungleichmäßige Abtastung an den Rändern
 - ▶ Abtastrate maximal für Richtungen entgegen der Aufnahmerichtung
 - ▶ Singularität in Aufnahmerichtung (am Rand)
 - ▶ ungleiche Abtastung:
deshalb eigentlich nur für Blickrichtungen ähnlich v_0 geeignet

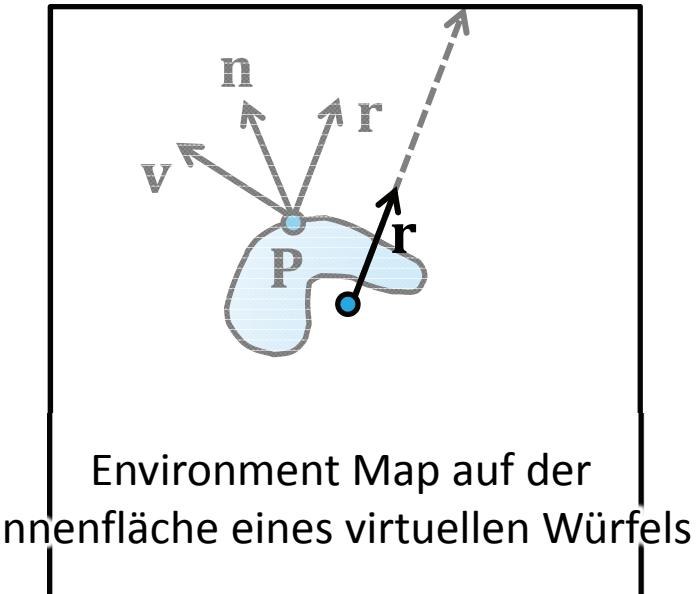
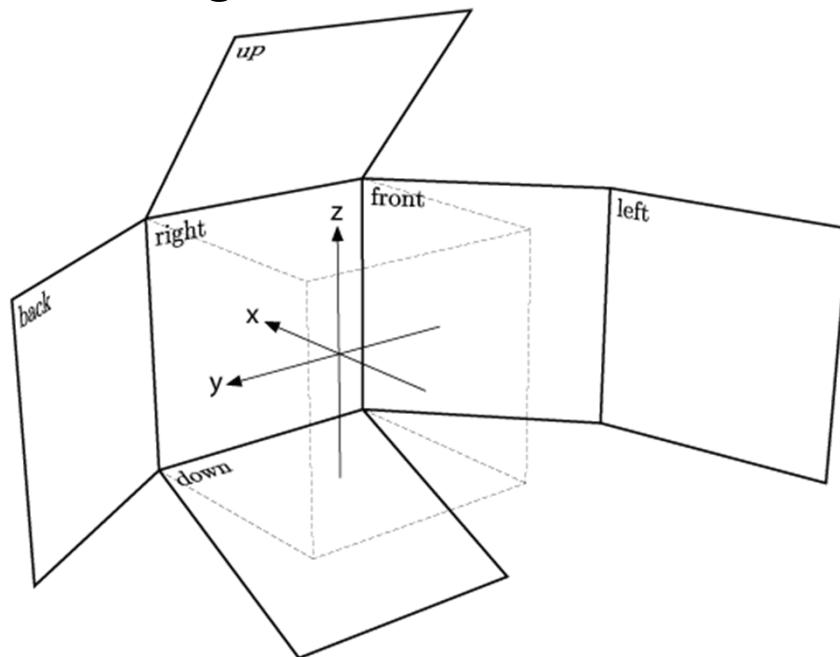


Kubische/Cube Environment Maps

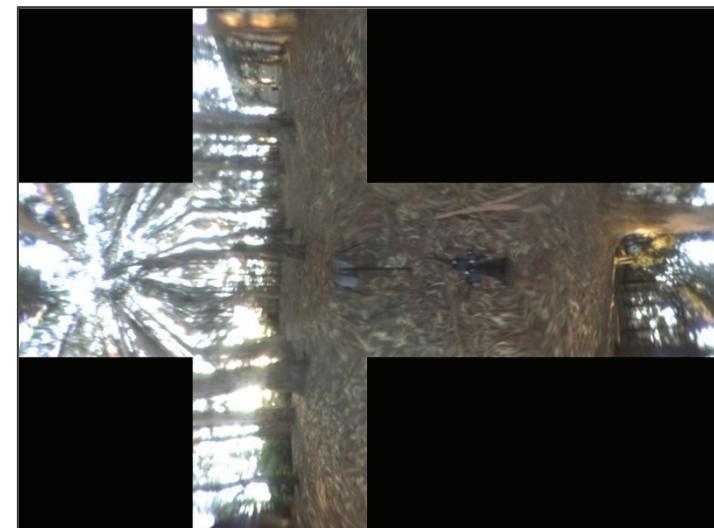


Eine weitere (sehr gebräuchliche) Parametrisierung: CubeMaps

- ▶ Abbilden der Umgebung auf einen Würfel um den Betrachter
 - ▶ benötigt also $6 \times 2D$ -Texturen
 - ▶ Abbildung/Aufnahme einer Würfelseite: perspektivische Kamera mit FOV 90°
 - ▶ Texturfilterung wie bei Mip-Mapping möglich



Environment Map auf der Innenfläche eines virtuellen Würfels



Kubische/Cube Environment Maps



Berechnung der Texturkoordinaten

- ▶ gegeben Reflexionsrichtung $\mathbf{r} = (r_x, r_y, r_z)$ ($|\mathbf{r}| \neq 1$ erlaubt)
- ▶ betragsmäßig größte Komponente von \mathbf{r} wählt Würfelfläche
 - ▶ $|r_x| > |r_y| \wedge |r_x| > |r_z| \Rightarrow$ „right“, wenn $r_x > 0$, „left“ sonst
 - ▶ $|r_y| > |r_x| \wedge |r_y| > |r_z| \Rightarrow$ „back“, wenn $r_y > 0$, „front“ sonst
 - ▶ (das ist eine von verschiedenen möglichen Konventionen)

- ▶ bestimme Schnittpunkt

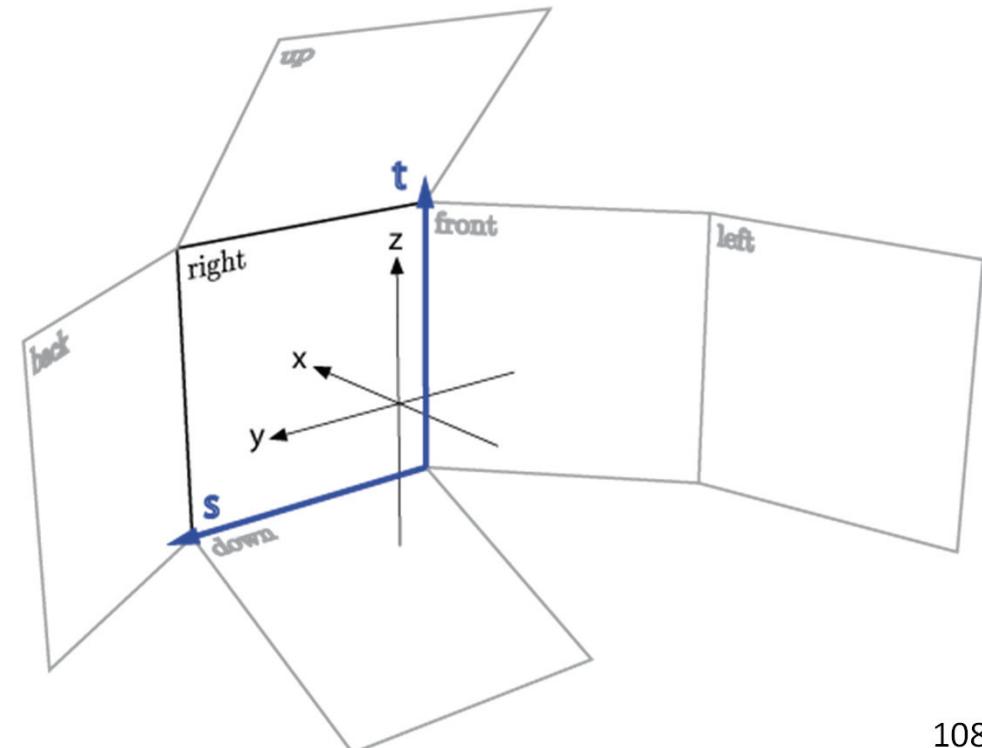
- ▶ Fläche „right“:

- ▶ $s = \frac{r_y}{2r_x} + \frac{1}{2}$ und $t = \frac{r_z}{2r_x} + \frac{1}{2}$

- ▶ Fläche „back“:

- ▶ $s = \frac{-r_x}{2r_y} + \frac{1}{2}$ und $t = \frac{r_z}{2r_y} + \frac{1}{2}$

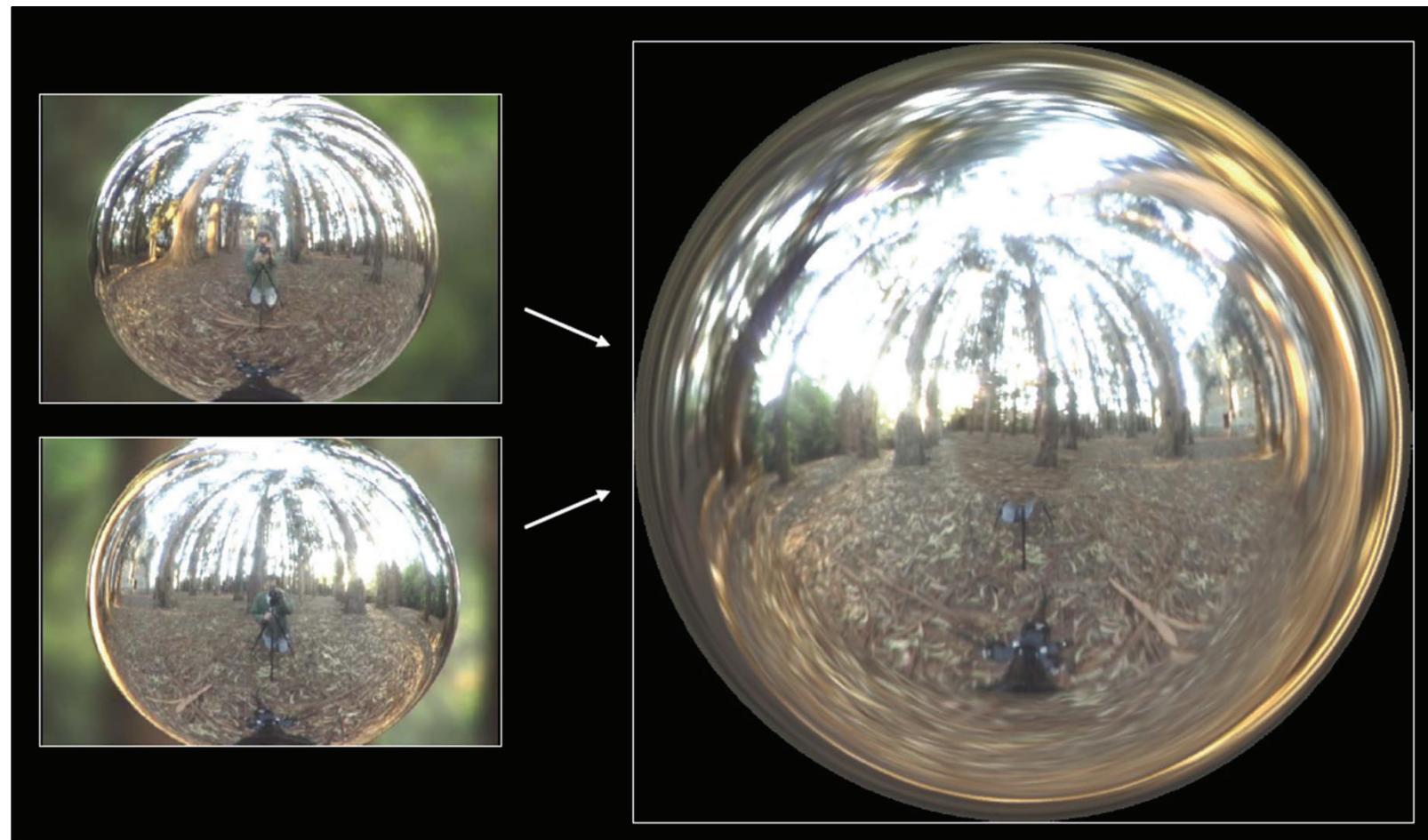
- ▶ ...



Aufnahme von Environment Maps

Aufnahme mit herkömmlichen Kameras

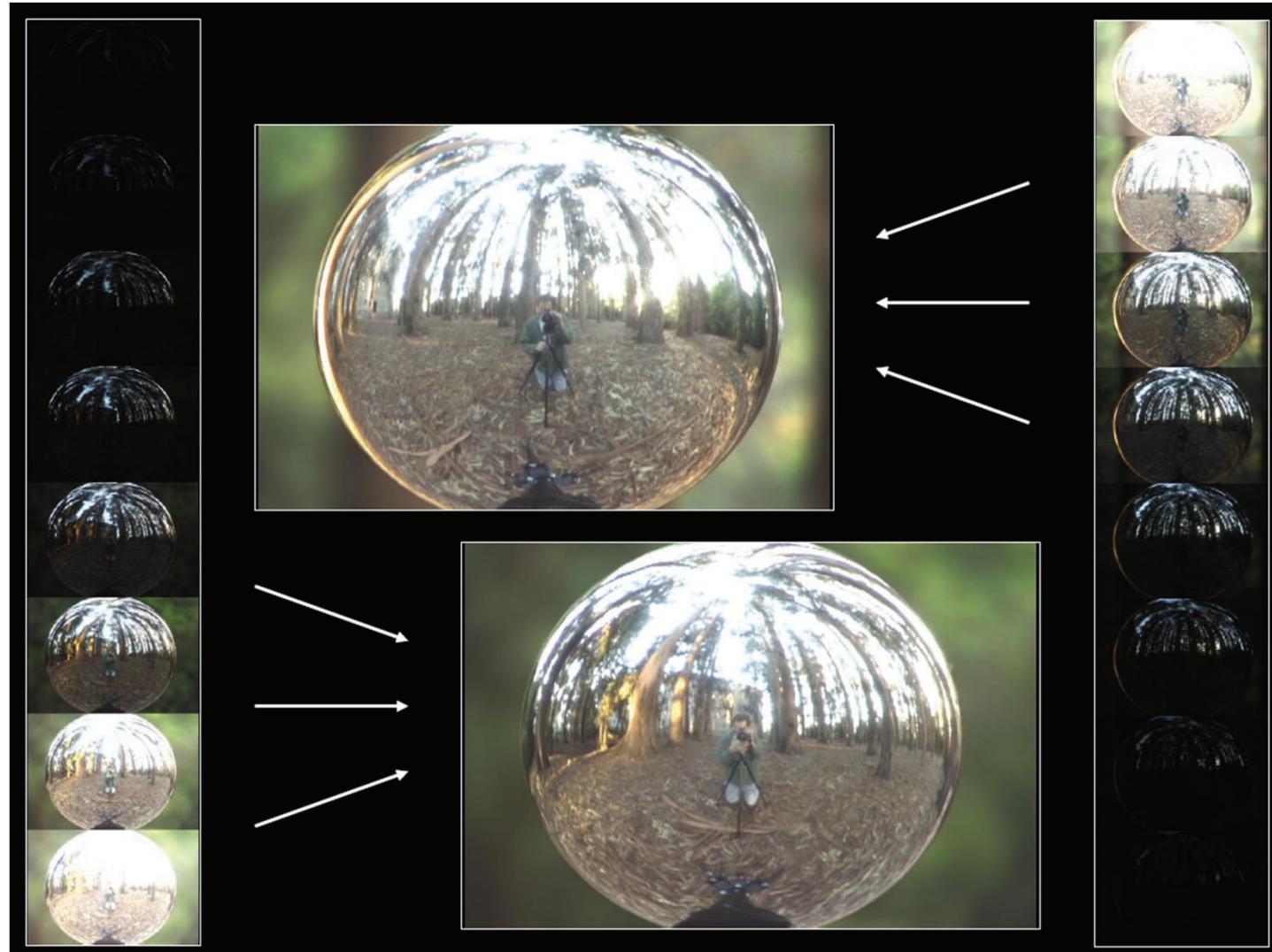
- ▶ in der Praxis: Aufnahme meist mit spiegelnden Chrom-Kugeln
 - ▶ mehrere Aufnahmen → nachträgliches Entfernen des Fotoapparats
 - ▶ Umrechnung in andere Parametrisierungen möglich



Aufnahme von Environment Maps

Aufnahme mit Belichtungsserien (HDR-Fotografie)

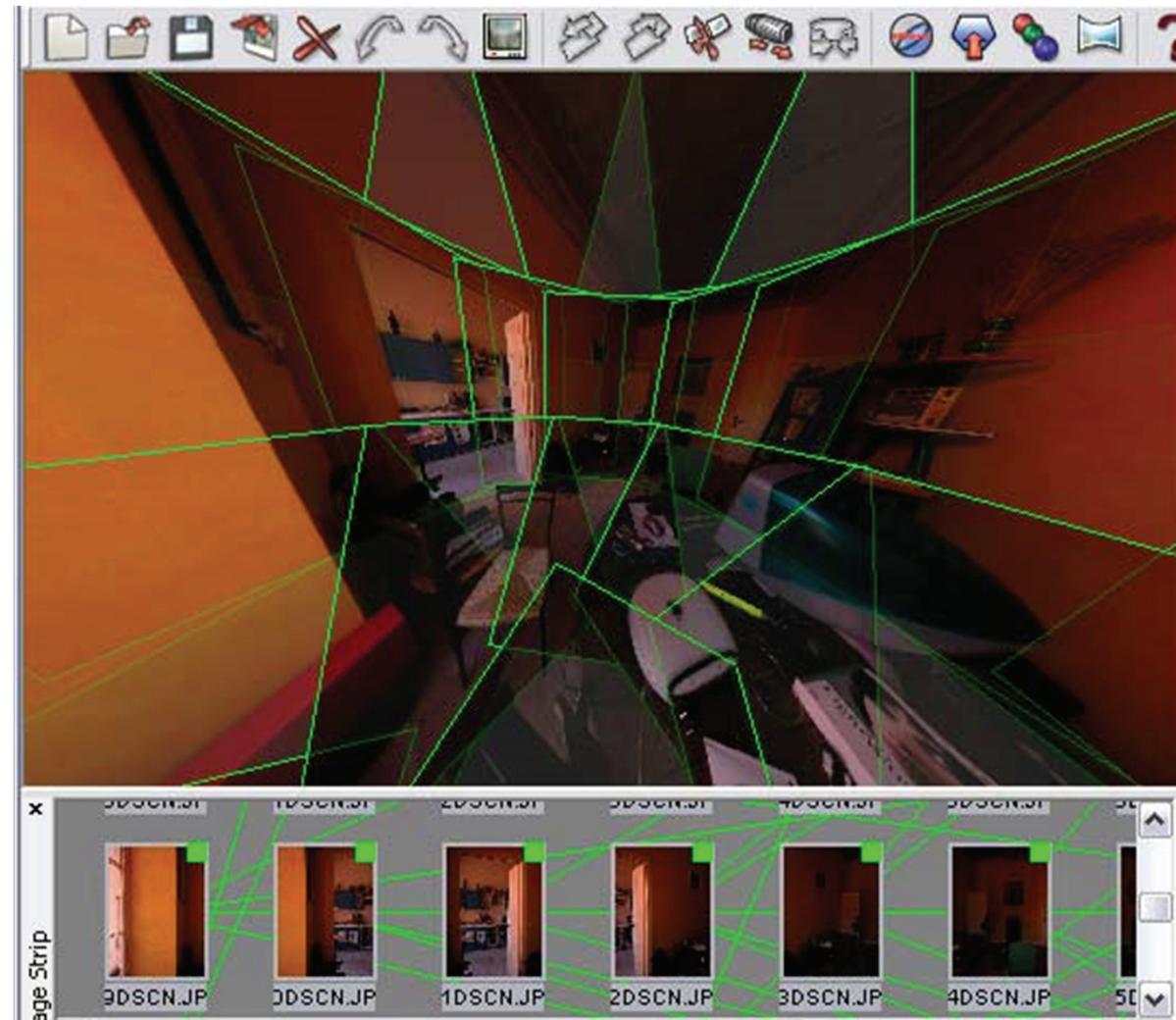
► ... zur Rekonstruktion eines größeren Dynamikumfangs



Aufnahme von Environment Maps

Aufnahme mit „Panorama-Funktion“?

- ... radiometrisch weniger verlässlich



Abtastrate

- ▶ in der Praxis verwendet man Sphere-Maps zur Aufnahme, LatLong zur Darstellung und Cube Maps oder sog. Paraboloid Maps für das Rendering
- ▶ Grund: Winkelauflösung in Abhängigkeit von der Abweichung zu v_0

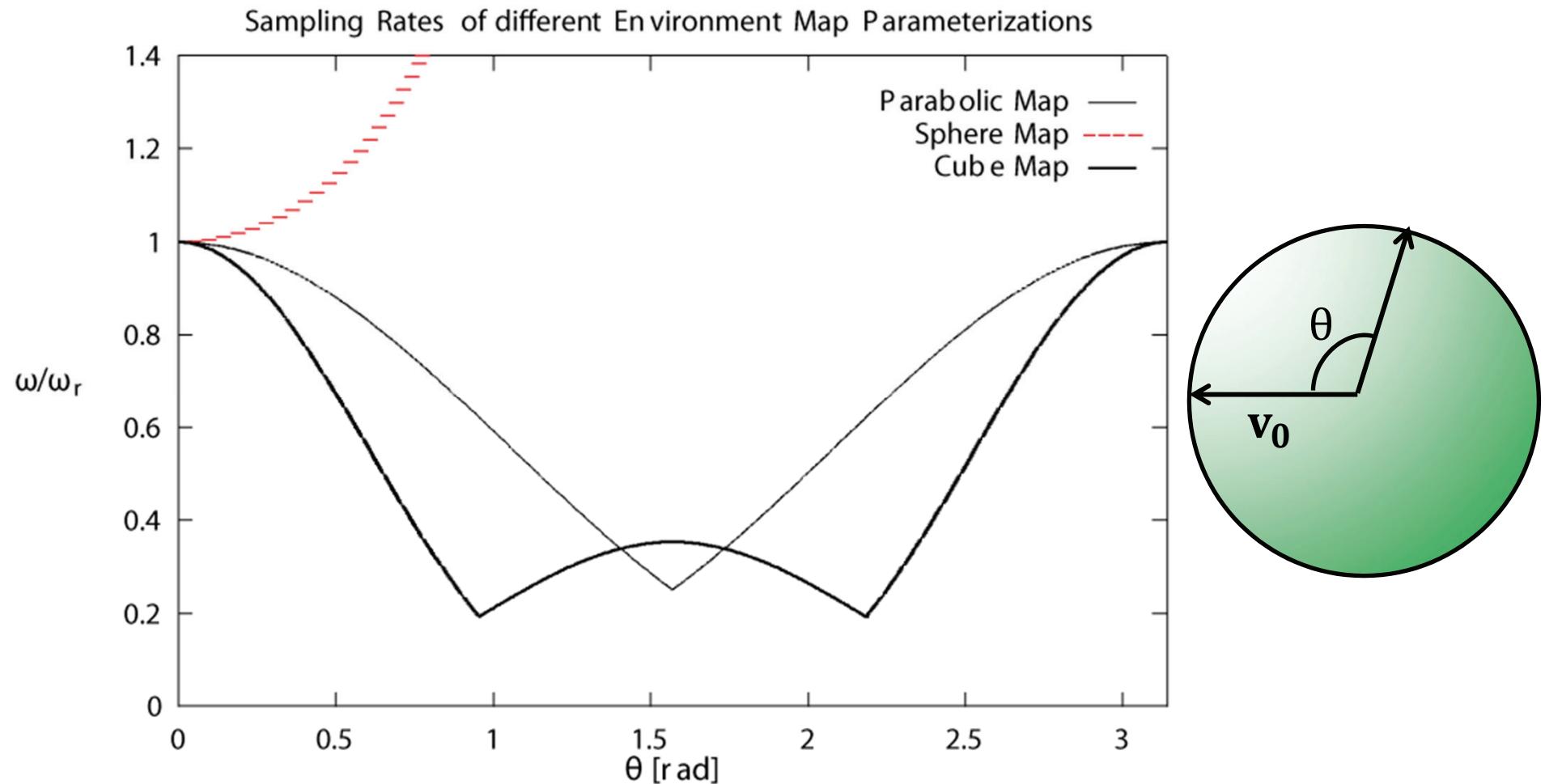
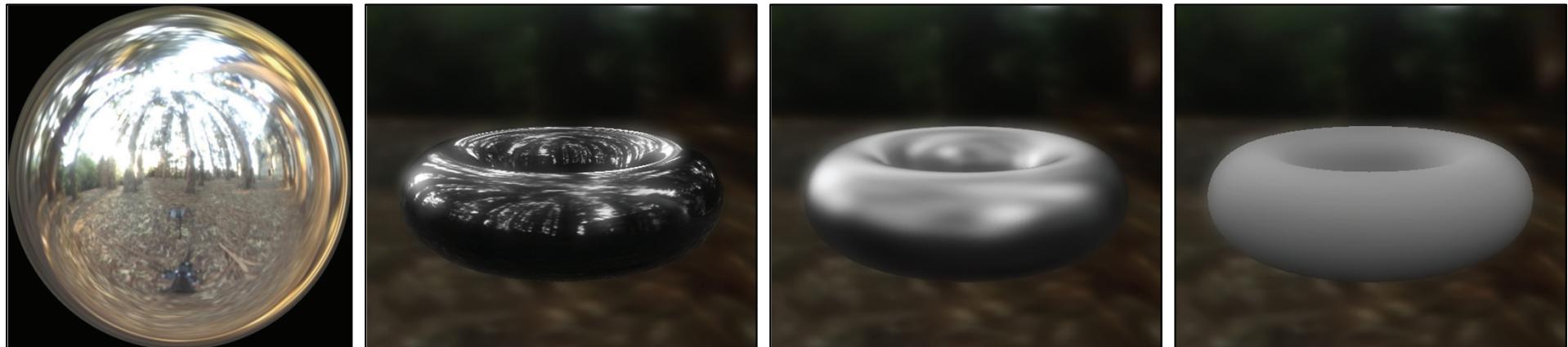


Image-based Lighting: Environment Maps...

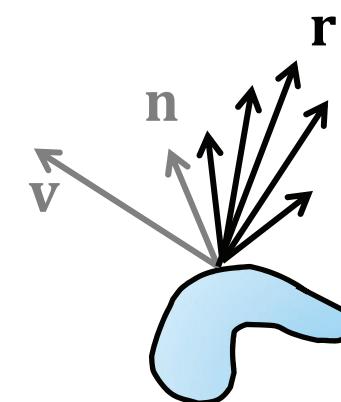


...für imperfekte Spiegelungen und diffuse Beleuchtung

- ▶ Ziel: Darstellung verschiedener Materialien ohne erhöhten Aufwand



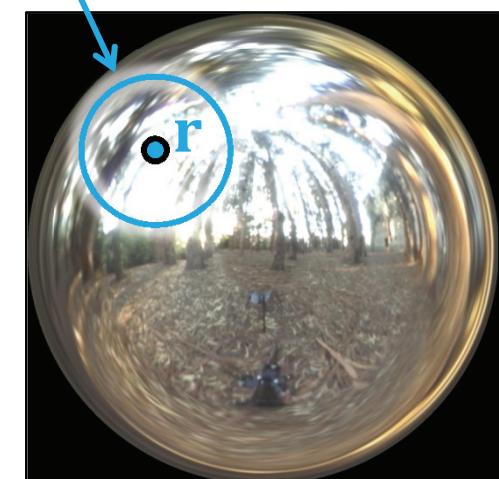
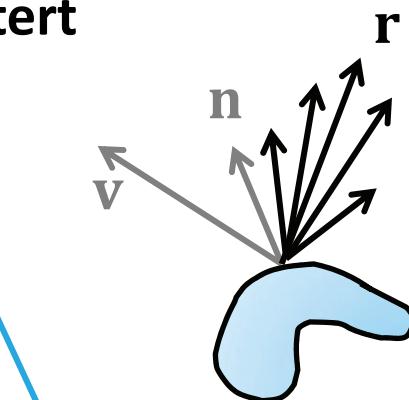
- ▶ imperfekte Spiegelung:
Licht um \mathbf{r} wird zum Betrachter hin reflektiert
- ▶ mehrfaches Abtasten der Textur ist u.U. zu teuer



Vorfilterung von Environment Maps

Grundidee: Vorfilterung (Prefiltered EnvMaps)

- ▶ Beobachtung: bei imperfekter Spiegelung trägt Licht aus einem Bereich um r zum reflektierten Licht bei
- ▶ statt mehrfachem Abtasten wird die Textur **vorgefiltert**
- ▶ einmaliges Auslesen liefert (gewichtete) Summe über einen Winkelbereich
- ▶ beachte:
Größe und Form des „richtigen“ Winkelbereichs hängt ab vom Material und der Parametrisierung



Vorfilterung von Environment Maps



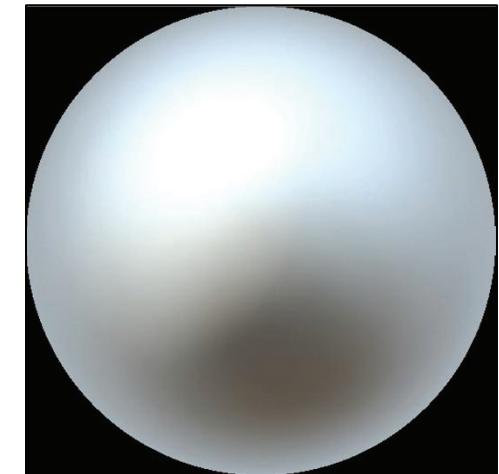
- ▶ eine 2D Textur kann nur ein 2D Signal (z.B. eine Richtung) repräsentieren
 - ▶ **Reflexionsrichtung:** spekulare Reflexion, wie viel Licht fällt aus einem bestimmten Richtungskegel um \mathbf{r} ein
 - ▶ **Normale:** wie viel Licht reflektiert eine diffuse Fläche mit Normale \mathbf{n}
 - ▶ Kombination mehrerer vorgefilterter Texturen möglich/oft verwendet
 - ▶ approx. Vorfilterung mit Summed Area Tables auch on-the-fly möglich



nicht vorgefiltert:
spiegelnde Objekte
(Zugriff mit \mathbf{r})



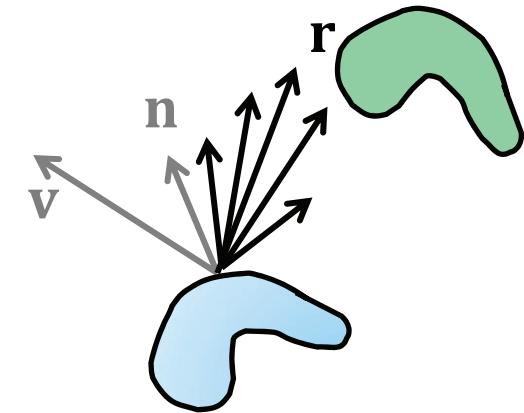
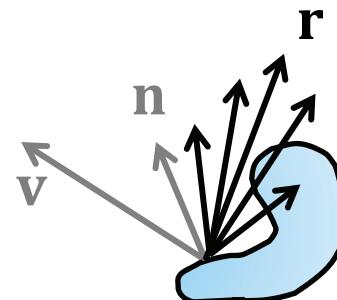
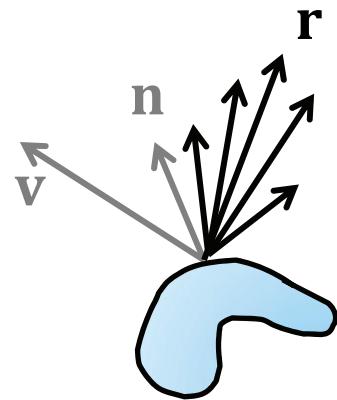
vorgefiltert für
imperfekte Spiegelung
(Zugriff mit \mathbf{r})



vorgefiltert für
diffuse Reflexion
(Zugriff mit \mathbf{n})

(Selbst-)Verschattung

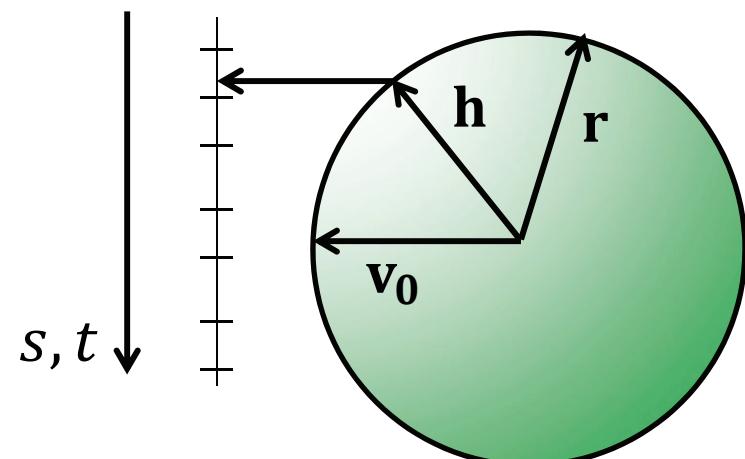
- ▶ wenn Effizienz wichtig ist, dann wird Verschattung durch das Objekt selbst (oder durch ein anderes Objekt) – wenn überhaupt – nur anhand von r , z.B. mit einem Schattenstrahl, entschieden
- ▶ als Notlösung wird oft (vorberechnetes) Ambient Occlusion verwendet



Vorfilterung von Environment Maps

Vorfilterung einer Sphere-Map für diffuse Beleuchtung

- ▶ verwende die Parametrisierung um Texel \leftrightarrow Richtungen zuzuordnen
 - ▶ aus der EnvMap wollen wir Licht aus allen Richtungen \mathbf{r} auslesen
 - ▶ in der Beleuchtungstextur wollen wir für alle Normalen \mathbf{n} diffus reflektiertes Licht speichern
- ▶ Hilfsfunktion: Abbildung von (s, t) auf eine Richtung \mathbf{r}
 - ▶ Texturkoordinaten $s = (h_x + 1)/2$ bzw. $t = (h_y + 1)/2$
 - ▶ $\Rightarrow h_x = 2s - 1$, $h_y = 2t - 1$ und $h_z^2 = 1 - h_x^2 - h_y^2$
 - ▶ Richtung \mathbf{r} ist Spiegelung von \mathbf{v}_0 an $\mathbf{n} = \mathbf{h}$, d.h. $\mathbf{r} = 2(\mathbf{v}_0 \cdot \mathbf{n})\mathbf{n} - \mathbf{v}_0$
 - ▶ auf der folgenden Folie:
`vec3 getDirection(float s, float t)`



Vorfilterung von Environment Maps



Vorfilterung einer Sphere-Map für diffuse Beleuchtung

```
vec3 sphereMap[ W * H ], prefiltered[ W * H ];
vec3 getDirection( float s, float t ) { ... };

// für alle Orientierungen
for ( y = 0; y < H; y++ ) {
    for ( x = 0; x < W; x++ ) {
        float s = x/(float)W, t = y/(float)H;
        vec3 normal = getDirection( s, t );

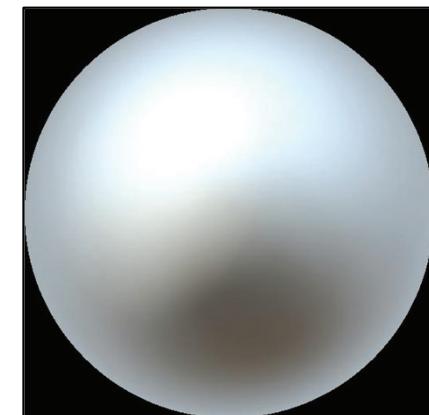
        // für alle Lichtrichtungen in der EnvMap
        vec3 color = 0.0;
        for ( j = 0; j < H; j++ ) {
            for ( i = 0; i < W; i++ ) {
                float u = i/(float)W, v = j/(float)H;
                vec3 r = getDirection( u, v );

                // Berechnung diffuse Reflexion für r und normal
                color += max( 0.0, dot( r, normal ) ) *
                    sphereMap[ i + j * W ];
            }
            prefiltered[ x + y * W ] = color;
        }
    }
}
```

sphereMap



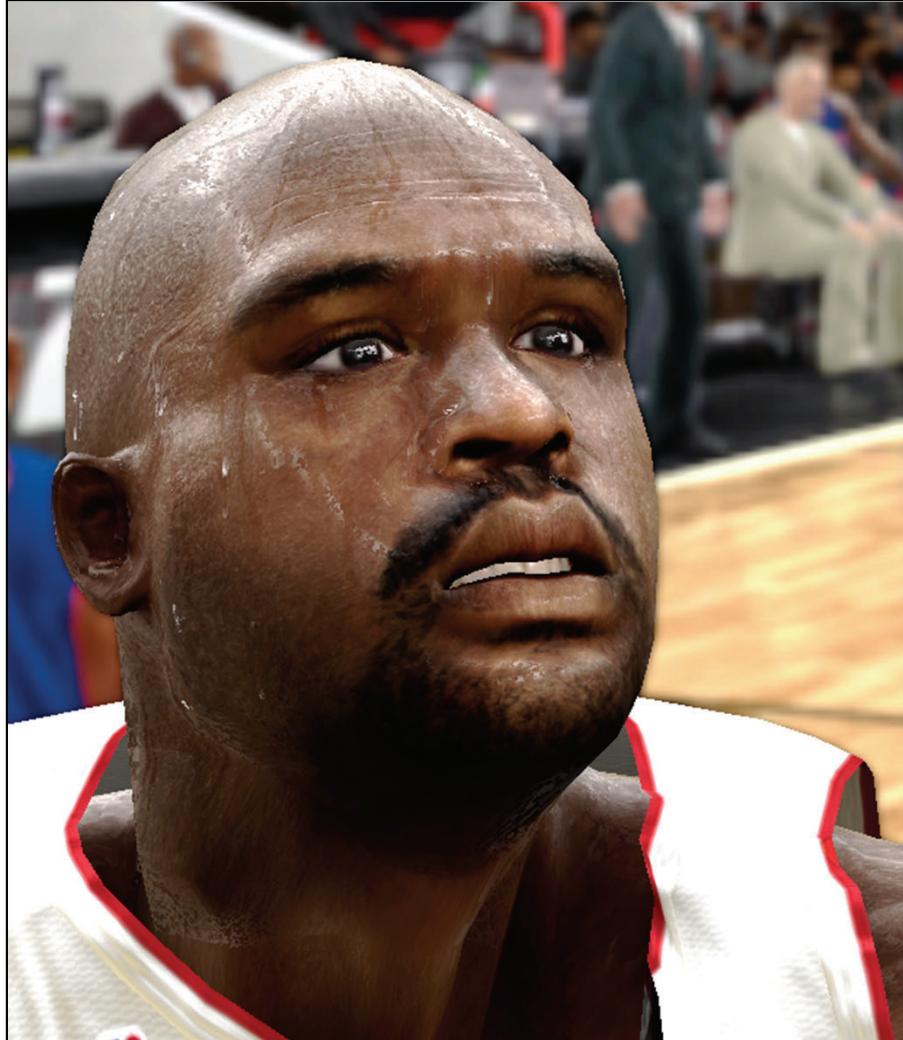
prefiltered



Das Ergebnis ist nicht ganz korrekt:
die Abtastung der Lichtrichtungen ist nicht
gleichmäßig und müsste kompensiert werden

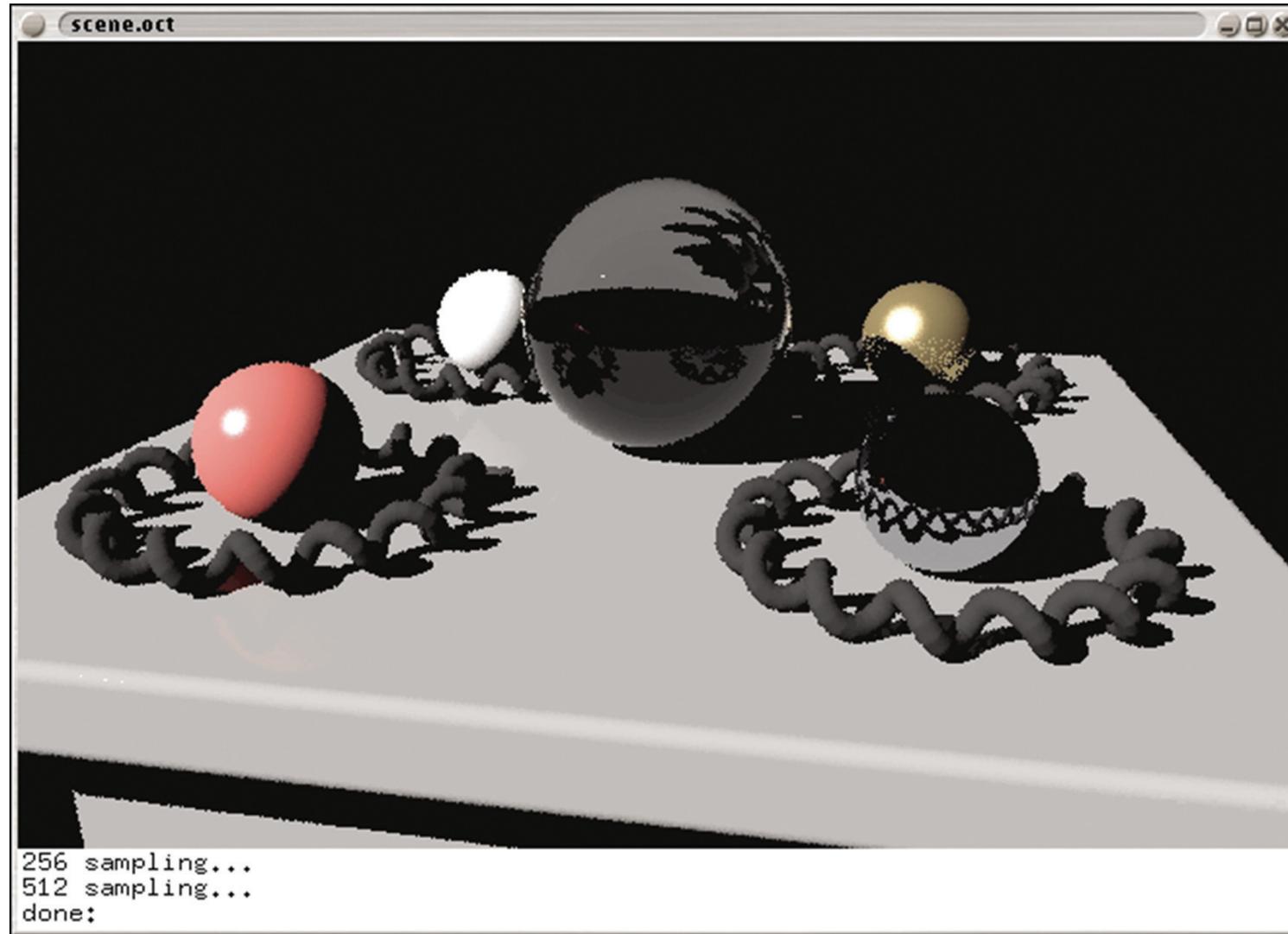
Environment Mapping in Spielen

- ▶ Beispiel aus der Echtzeitgrafik: NBA2k7 – Visual Concepts
 - ▶ grobe Approximation: Vorfilterung durch Mip-Mapping statt Faltung



Anwendungen

- hier: Ray Tracing mit einer direktonalen Lichtquelle



Weitere Anwendungen

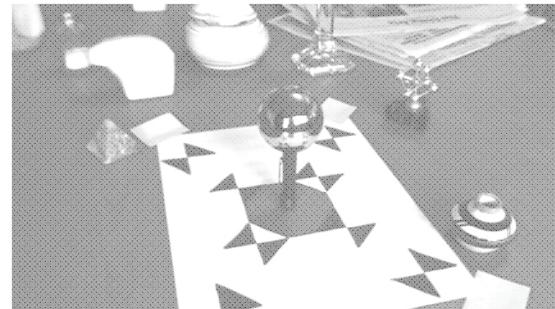
- ▶ Image-Based Lighting: Environment-Map zur Beleuchtung der Szene
- ▶ hier (vereinfacht ausgedrückt): jeder Pixel der Environment-Map wird zu einer direktonalen Lichtquelle



Image-Based Lighting



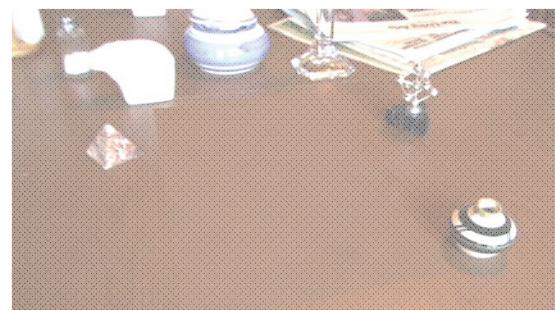
Fotografie



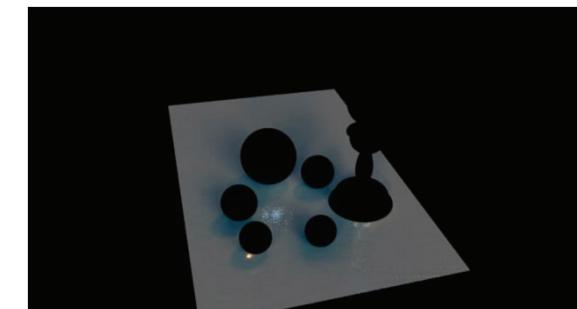
Light Probe und Kalibrierungsgitter



synthetische Objekte + Ebene



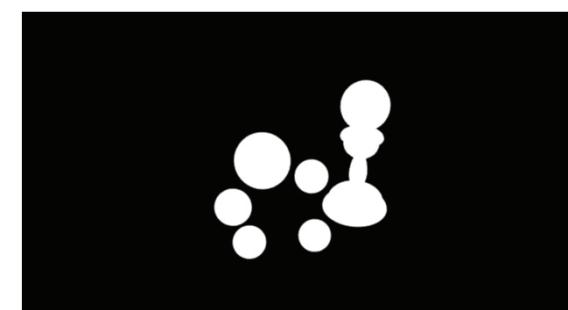
nur Ebene



Differenz



fertiges Bild



Objektmaske

Fazit

- ▶ Texture Mapping ist eine fundamentale Technik in der Computergrafik
- ▶ erlaubt Detailreichtum, der mit reiner Geometrie nicht möglich wäre
- ▶ vielseitiger Einsatz
 - ▶ Farbtexturen
 - ▶ Normalen
 - ▶ Displacement
 - ▶ Environment Mapping, Image-Based Lighting,
 - ▶ Shadow Mapping
 - ▶ Visualisierung
 - ▶ Lookup-Tabellen für aufwändige Berechnungen
 - ▶ ...