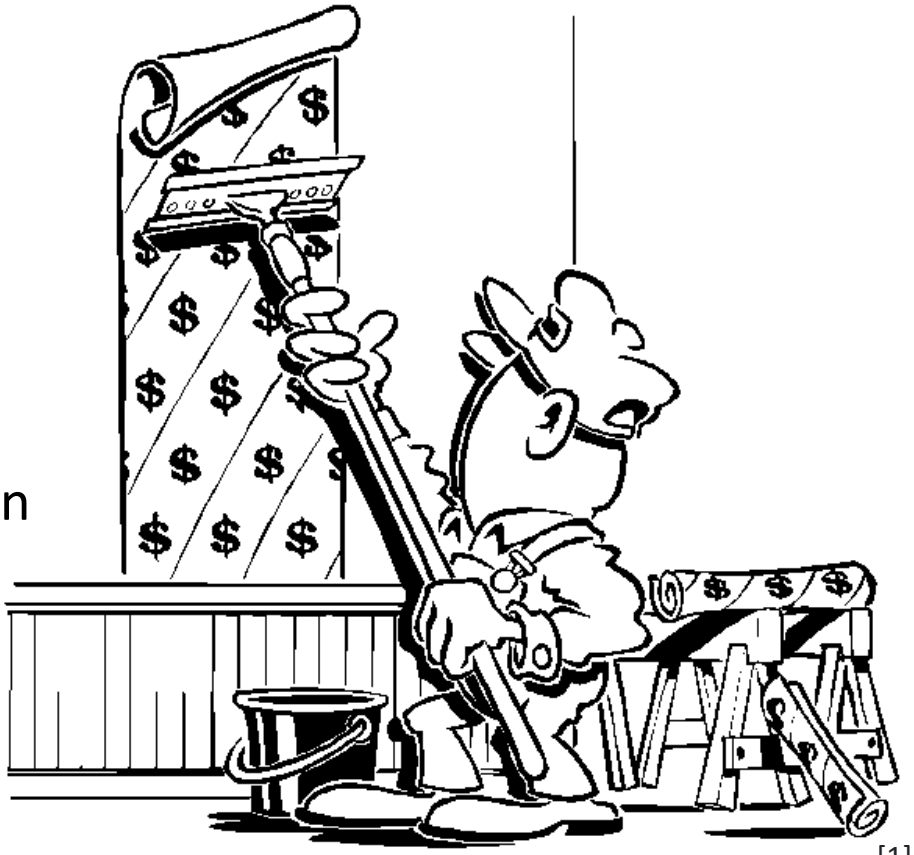
Computergrafik

Vorlesung im Wintersemester 2024/25
Kapitel 4: Texture Mapping

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

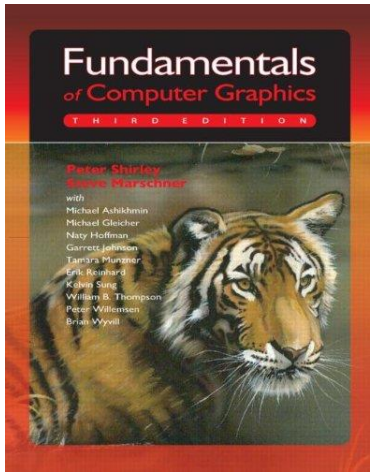
Inhalt

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



[1]

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

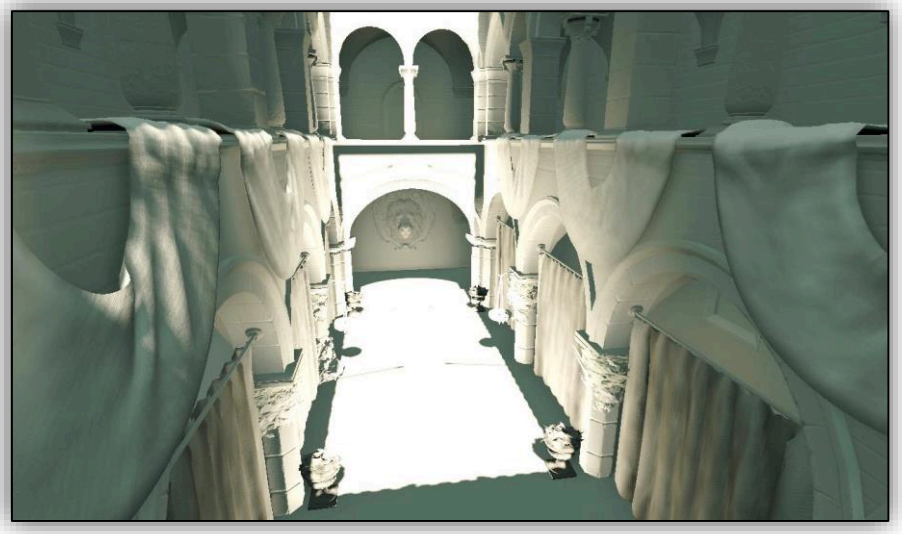


[2]

Texturen und Oberflächendetail

Grundidee

- realistischeres Aussehen einer Oberfläche kann man durch „Feinstrukturierung pro Pixel“ erreichen
- kombiniere Geometrie mit Bildern
- dadurch erreicht man einfach hohe Detailgrade, auch mit gröberen Netzen

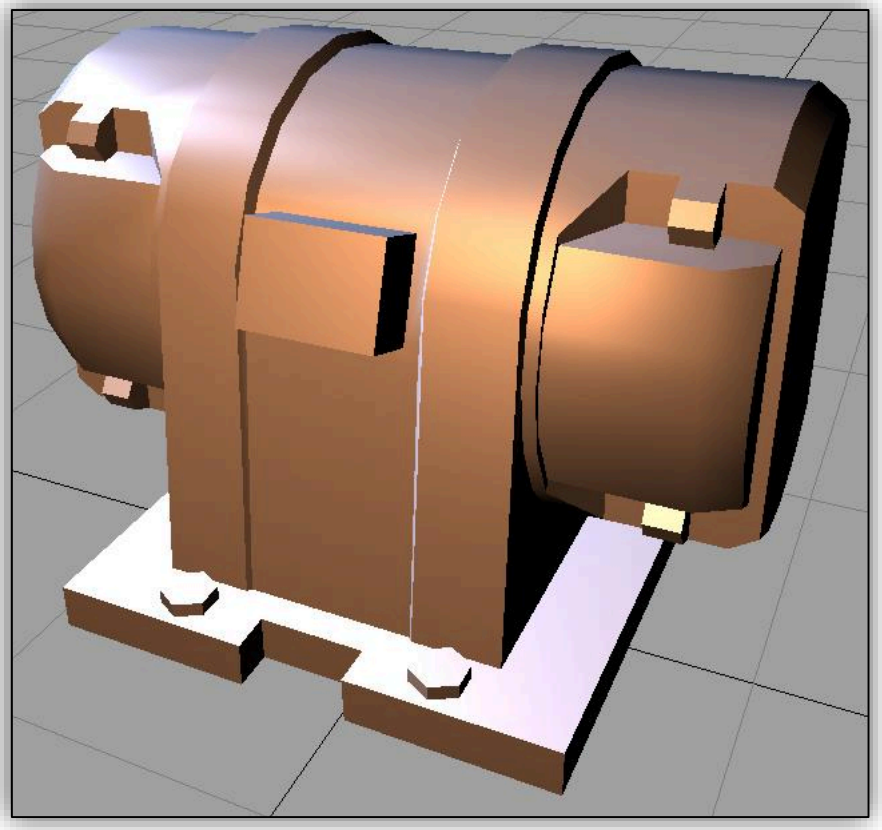


Texturierungstechniken

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

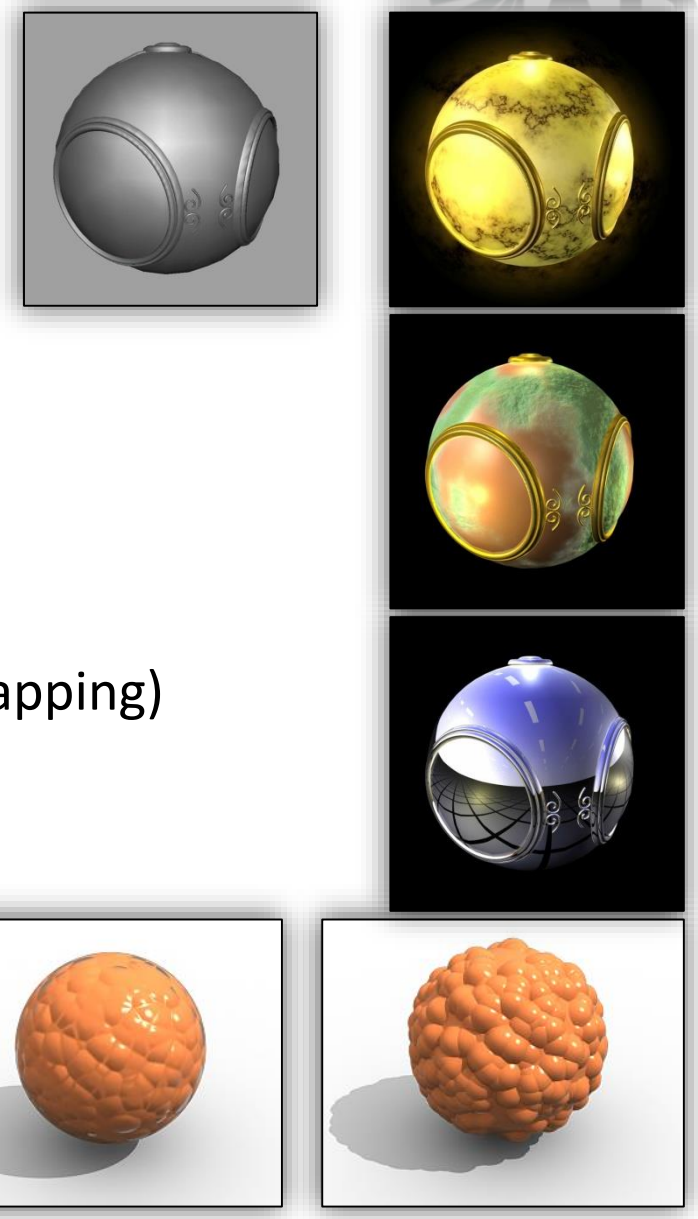
- Bsp. Kontrolle der Parameter des Phong-Beleuchtungsmodells (blau eingefärbt):

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



Texturierungstechniken

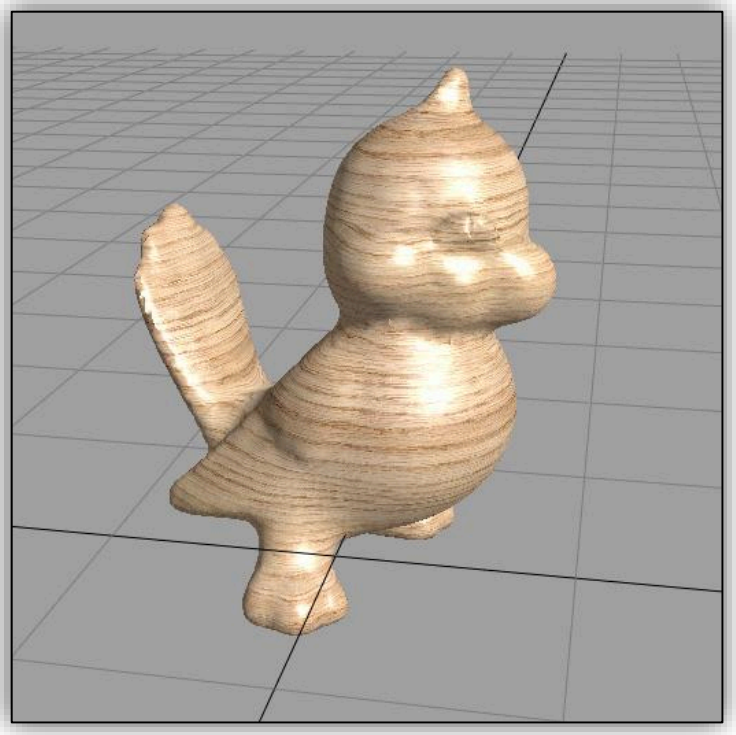
- klassisches Texture Mapping
  - Reflexionseigenschaften: Farbe, div. Koeffizienten, Transparenz
- Bump Mapping oder Normal Mapping
  - Veränderung der Normalen für die Beleuchtungsberechnung
- Beleuchtung
  - Environment Mapping (auch Reflection Mapping)
  - Shadow Mapping, Light Maps
- Displacement Mapping
  - tatsächliche Veränderung der Geometrie/von Flächen durch Verschiebung



Angel, Shreiner, Interactive Computer Graphics: A Top-Down Approach With Shader-Based OpenGL [3]

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)

Mapping: Beispiele

- hat man eine Abbildung von Oberfläche zu Textur, dann kann man diese Parametrisierung oft noch anderweitig nutzen!

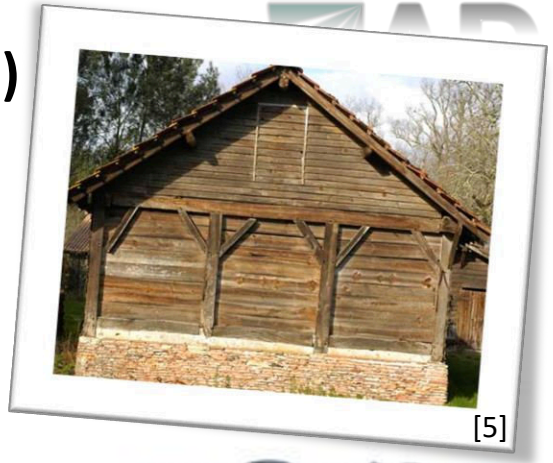
- Bsp. Beleuchtung und Streuung des Lichts unter der Haut im 2D-Texturraum approximieren

Matrix: Reloaded, Sketch SIGGRAPH 2003[4]

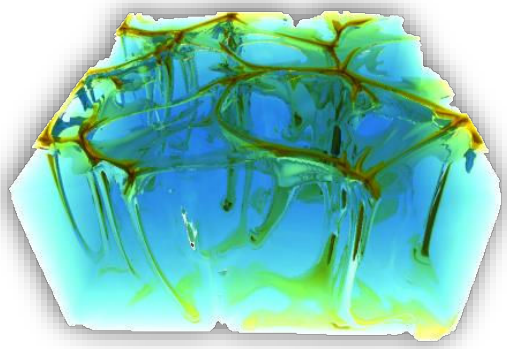
Textur-Quellen

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

- Texturen sind (meist) Rasterbilder bestehend aus **Texels** (Texture Elements, vgl. Pixel)
- oft einfache Erstellung oder Akquisition
  - 1D-Texturen (z.B. Farbtabellen)
  - 2D-Texturen (Fotografie, berechnet, aus Bildbearbeitungsprogramm)
- hoher Speicherbedarf bei vielen, großen Texturen
  - Texturkompression: blockweise, typ. 4×4 Texel-Blöcke (bei 2D), 4:1 oder 8:1 Kompression
- 3D-Textur aus prozeduraler Auswertung und aus einer Simulation (rechts Mitte/unten)


[5]


[9]



Textur-Quellen – Ausblick auf Kapitel 8

Prozedurale Texturen durch Algorithmen (in sog. Shader)

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

Textur-Quellen – Ausblick auf Forschungsarbeiten

Vektorgrafik/Diffusion für Texturen (wenig praxisrelevant in Bildsynthese)

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

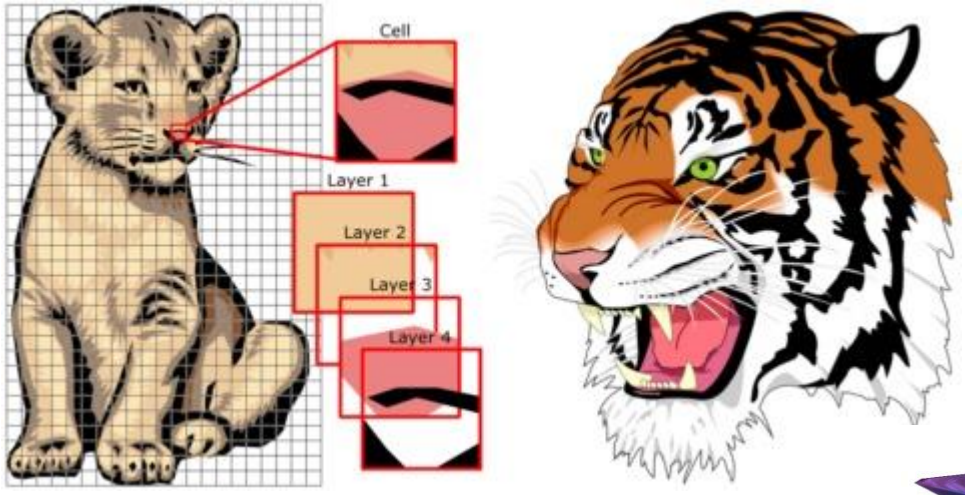


Bild: Random-access rendering of general vector graphics,
D. Nehab und H. Hoppe 08 [6]

Rendering Surface Details with
Diffusion Curves, Jeschke et al. 09 [7]

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

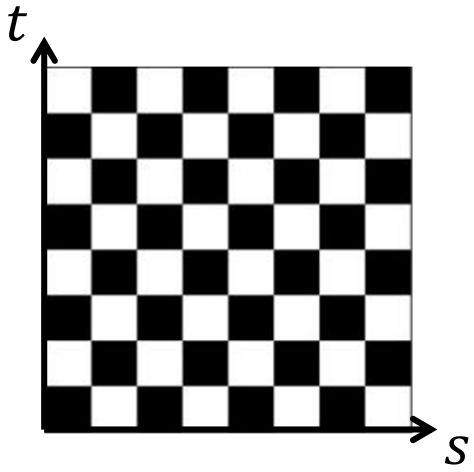
Mapping von 2D-Texturen

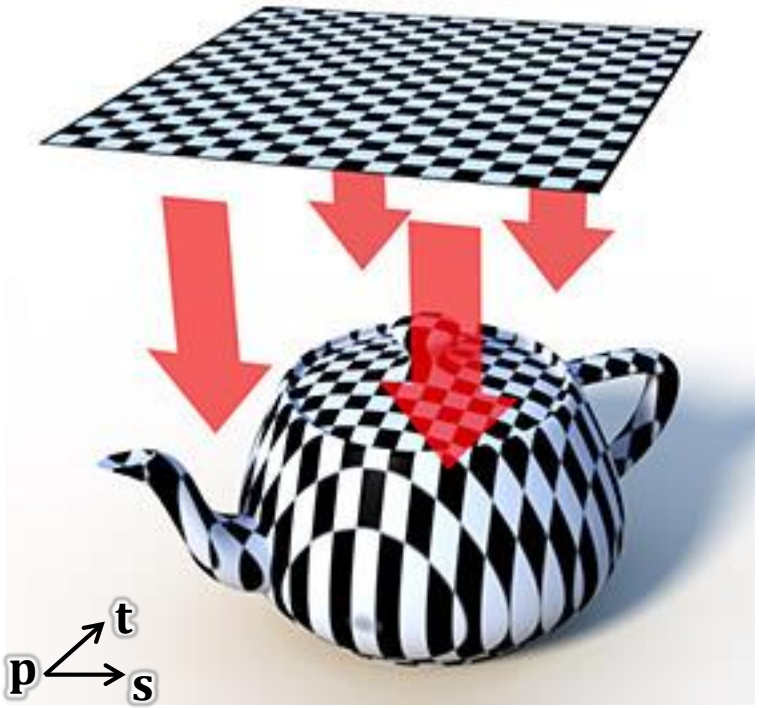
Planare Projektion

- definiere Ebene, z.B. durch einen Punkt $\mathbf{p}$ und zwei aufspannende Vektoren $\mathbf{s}, \mathbf{t}$
- Texturkoordinate eines Oberflächenpunkts $\mathbf{x}$

  $s = (\mathbf{x} - \mathbf{p}) \cdot \mathbf{s}$

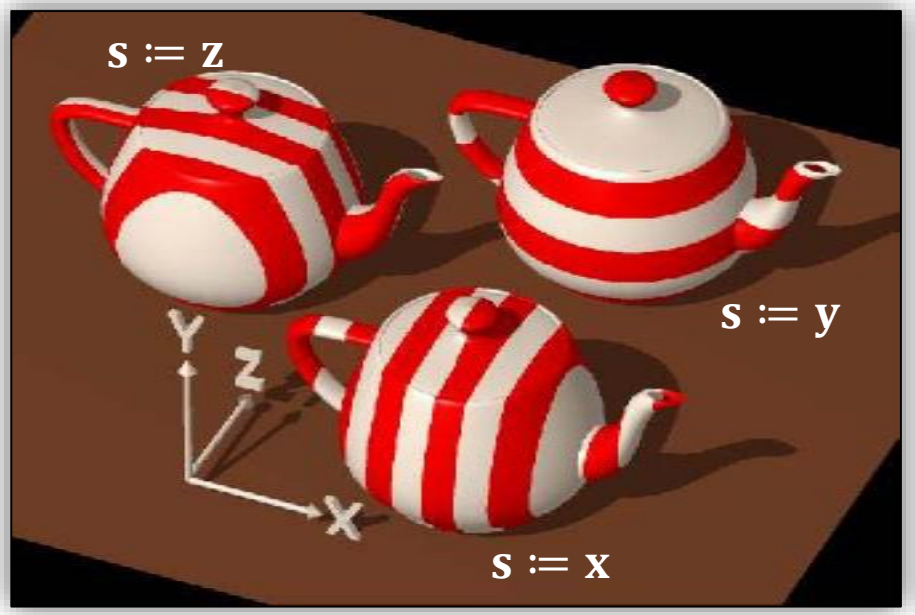
  $t = (\mathbf{x} - \mathbf{p}) \cdot \mathbf{t}$
- der Bereich $[0,1]^2$ repräsentiert die ganze Textur → Unabhängigkeit von der tatsächlichen Auflösung



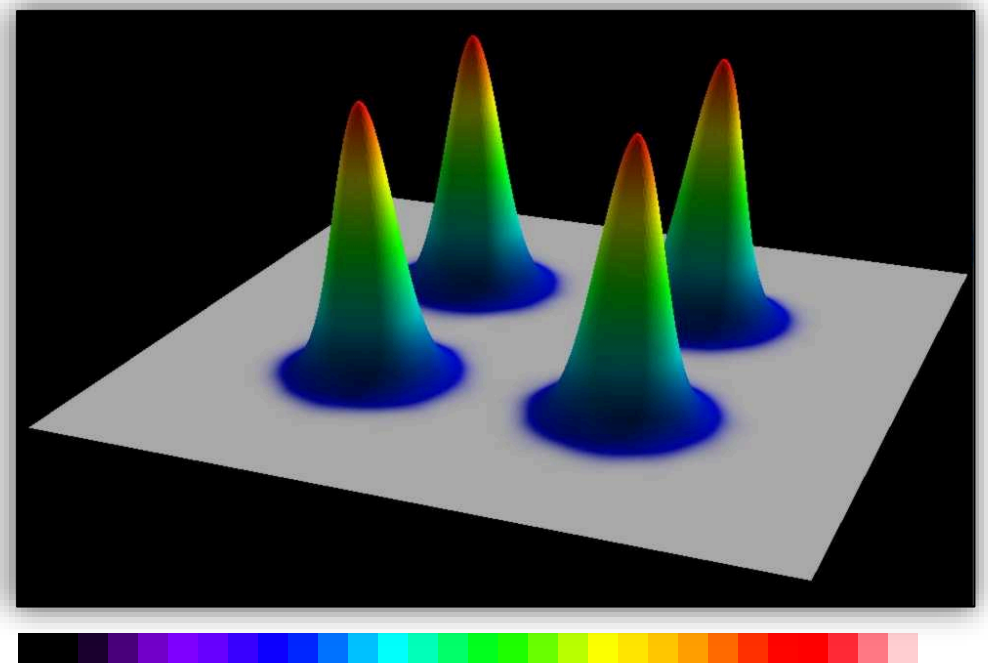


Mapping von 1D-Texturen

- Parameter einer Linie, Höhe, Temperatur, …
  - z.B. Texturkoordinate $s = (\mathbf{x} - \mathbf{p}) \cdot \mathbf{s}$
  - links: Parameter entlang einer der drei Achsen
  - rechts: 1D-Textur entlang der vertikalen Achse



[40]

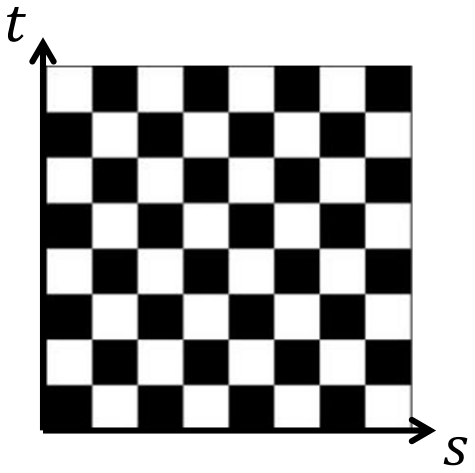


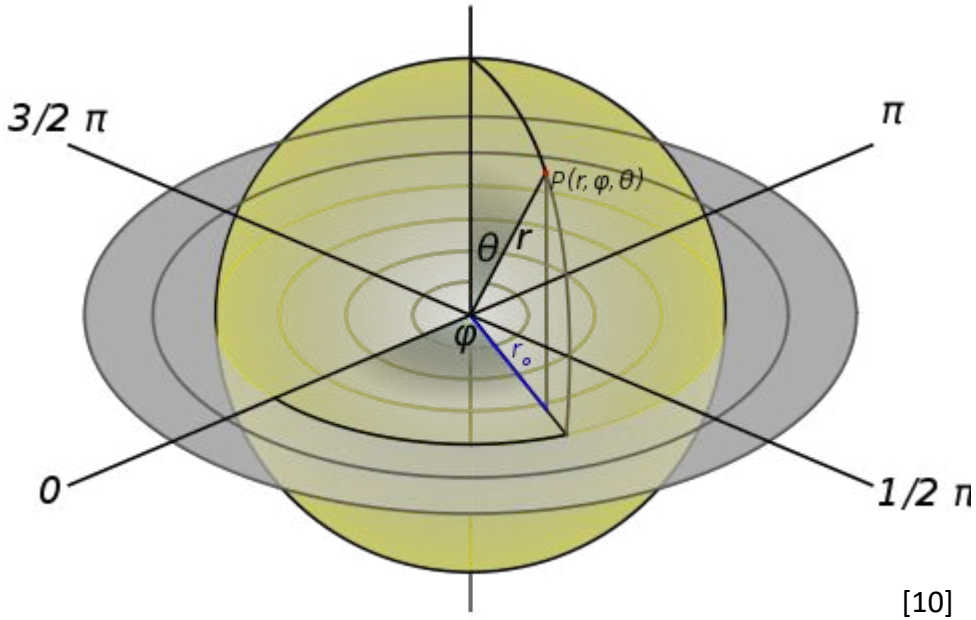
Standardkörper mit natürlicher Parametrisierung

Kugel-Parametrisierung

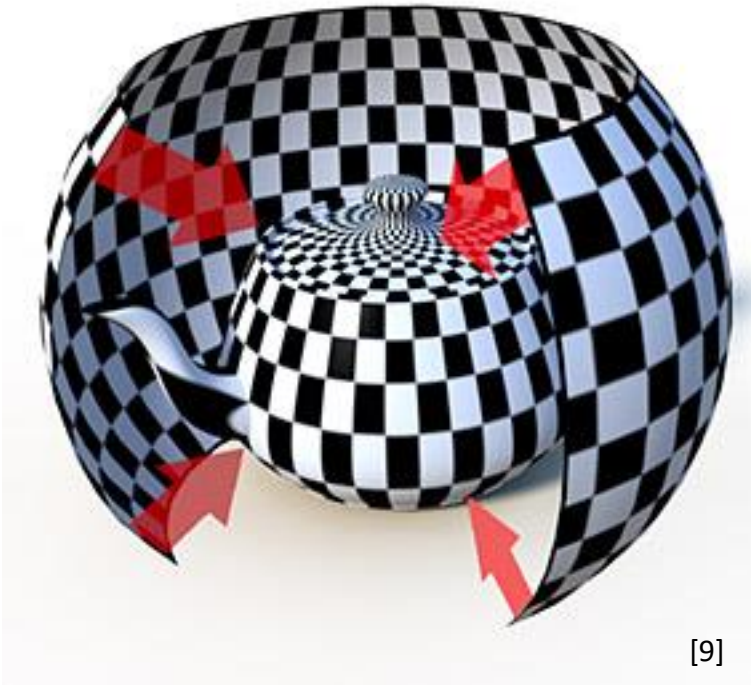
- Punkte auf der Oberfläche einer Kugel im Ursprung lassen sich mit Polarkoordinaten (r, ϕ, θ) ausdrücken
- die zwei Winkel werden für den Zugriff auf eine 2D-Textur verwendet:

  Texturkoordinate $\begin{pmatrix} s \\ t \end{pmatrix} \coloneqq \begin{pmatrix} \phi/2\pi \\ \theta/\pi \end{pmatrix}$





[10]

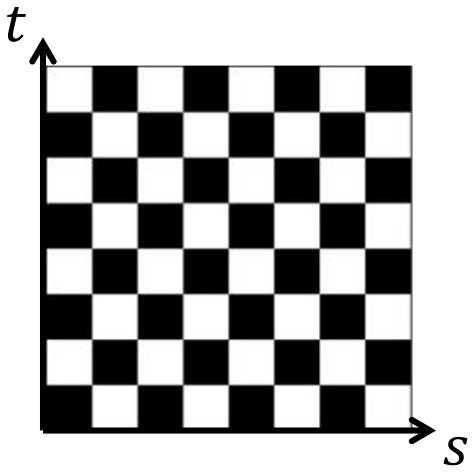


[9]

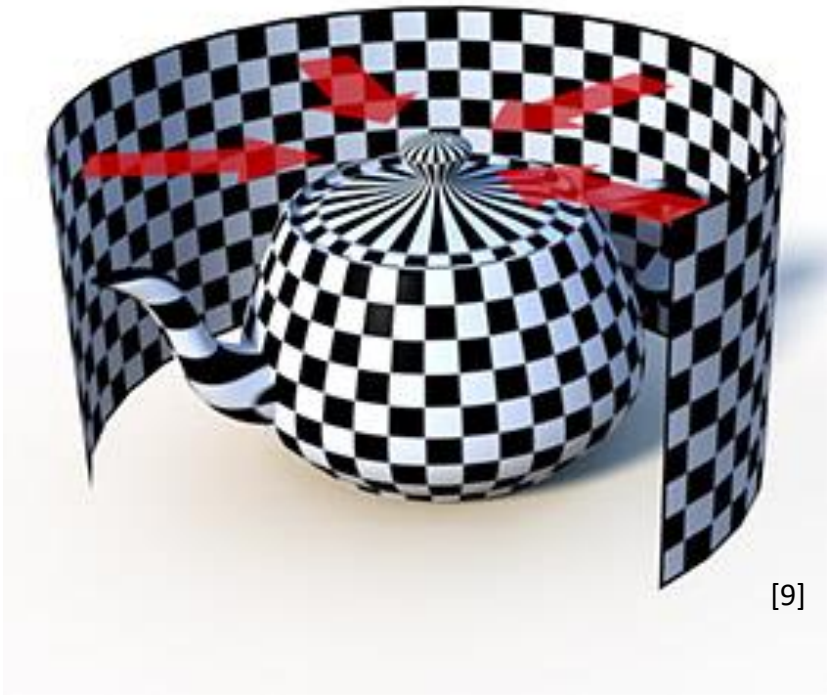
Standardkörper mit natürlicher Parametrisierung

Zylindrische Parametrisierung

- Punkte auf der Oberfläche in Zylinderkoordinaten (r, ϕ, y)
- Texturkoordinaten $\begin{pmatrix} s \\ t \end{pmatrix} \coloneqq \begin{pmatrix} \phi/2\pi \\ y/h \end{pmatrix}$



- beachte: wie schon bei der Kugelparametrisierung starke Verzerrungen der Textur



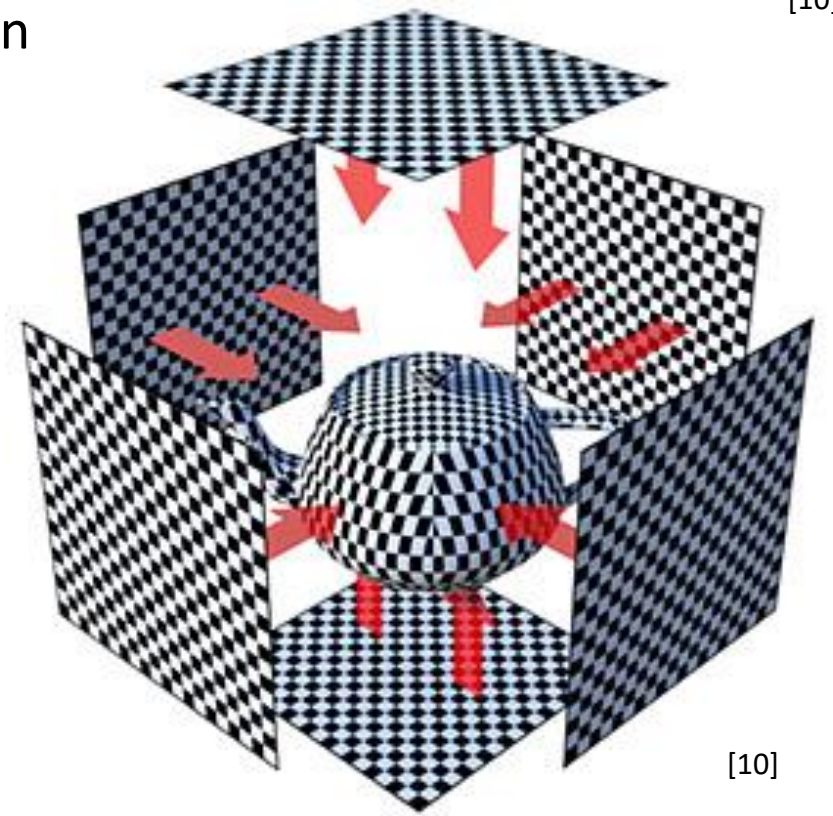
[9]

Standardkörper mit natürlicher Parametrisierung

Würfel-Parametrisierung

- mehrere mögliche Abbildungen:
  - Projektion von 6 Ebenen je nach Oberflächennormale (Bild), oder
  - lese die Würfeltextur dort aus, wo ein Strahl vom Objektmittelpunkt durch einen Oberflächenpunkt einen umgebenden Würfel schneidet
  - oder …

[10]

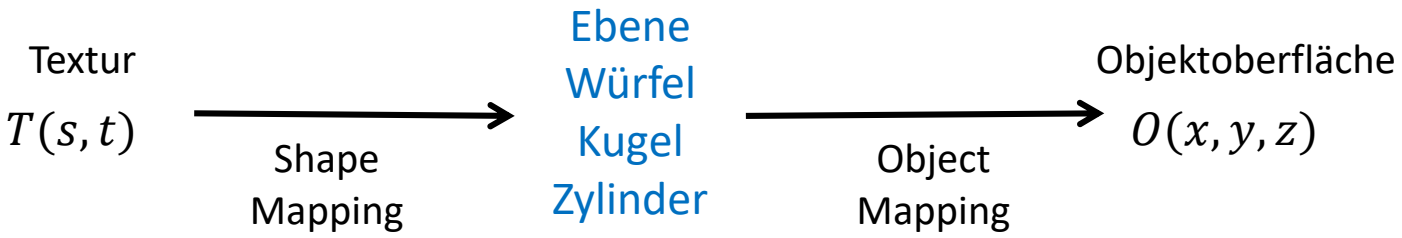


[10]

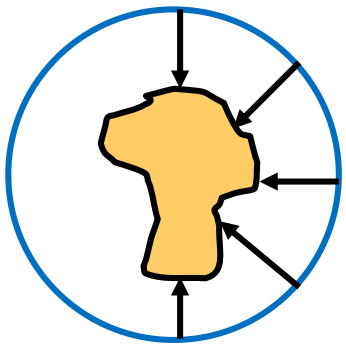
Standardkörper mit natürlicher Parametrisierung

Texturierung beliebiger Objekte durch einfache Parametrisierungen

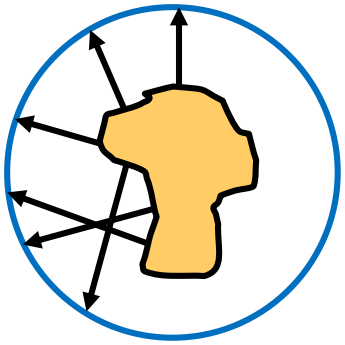
- allg.: verwende Standardkörper als Zwischenschritt bzw. Hilfsfläche



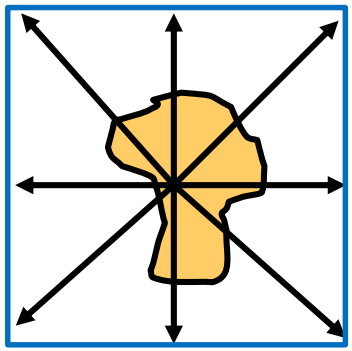
- mögliche Abbildungen zw. Objektpunkt und Standardkörper:



Normale der Hilfsfläche
→ Oberfläche



Normale der Oberfläche
→ Hilfsfläche

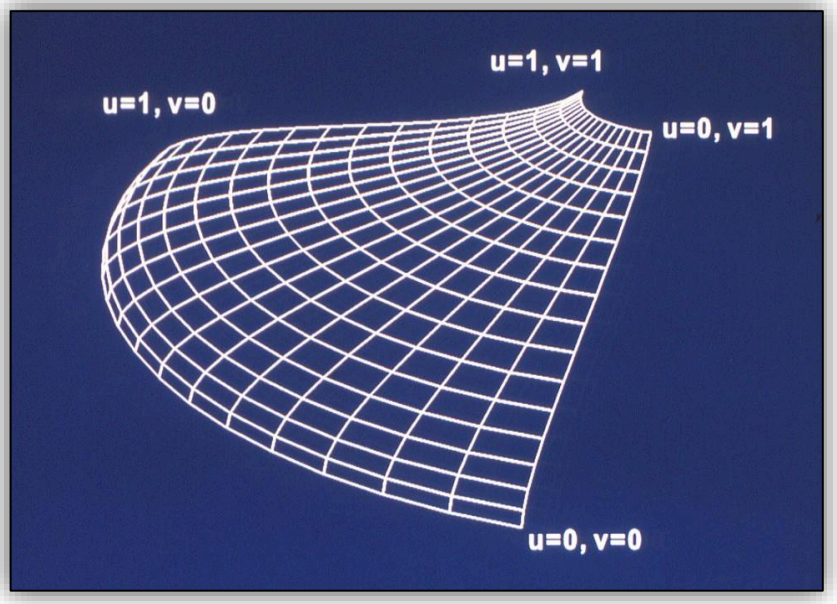


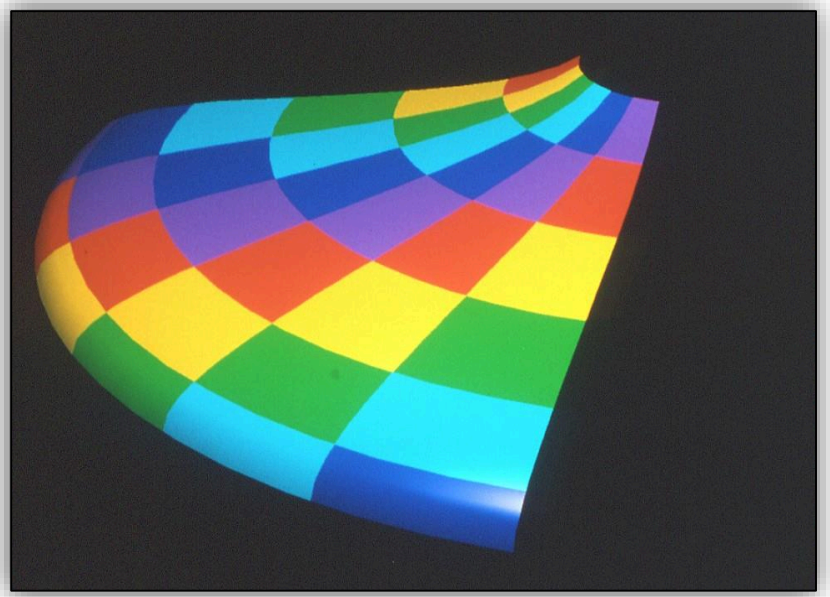
Linie durch Mittelpunkt

- i.A. Teile der Textur mehrfach auf der Oberfläche abgebildet!

Flächenparameter als Texturkoordinate

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





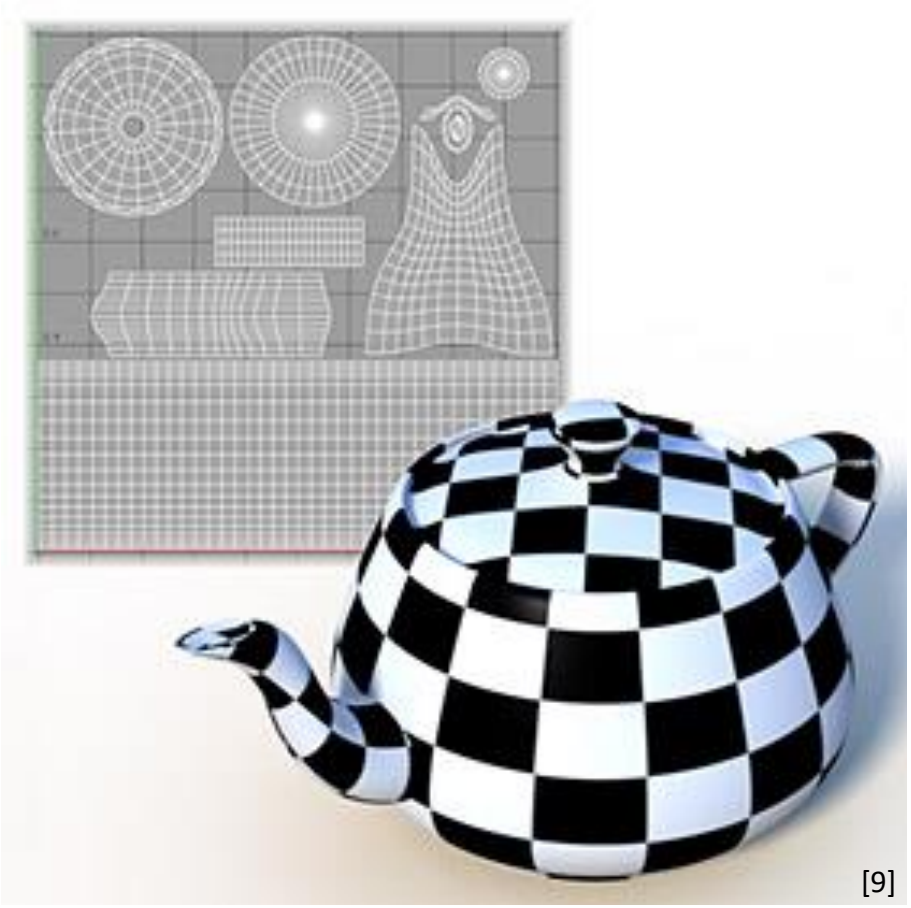
[42]

Flächenparameter als Texturkoordinate

- der „Utah Teapot“ ist aus parametrischen Flächen zusammengesetzt
  - glutSolidTeapot(), 3DSMax Standardkörper (neben Quader, Ebene, …)

2D- und 3D-Texturierung

Ziel: Minimieren von Verzerrungen, bijektive Abbildungen


[9]

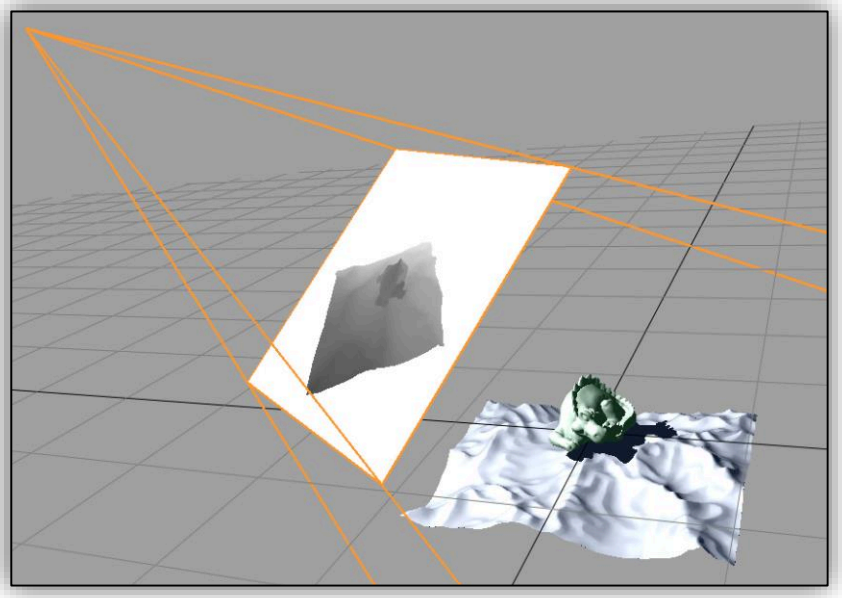
2D: Texture Mapping
(hier mit einem sog. Texturatlas, später mehr hierzu)


[9]

3D: Solid Texturing
$(u, v, w) = \mathbf{x}$

Texturabbildungen: Überblick

- Projektion auf andere Flächen (Kugel, Zylinder, Box, Ebene, ...)
- Flächenparameter (u, v) (z.B. Bézier-Patch)
- explizites Festlegen von sog. Texturkoordinaten (s, t, r) (auch $(u, v, \dots)$)
- Objekt- (x_o, y_o, z_o) oder Weltkoordinaten (x_w, y_w, z_w)
- Bildschirm- bzw. Kamerakoordinaten (x_s, y_s) (Shadow Maps)
- Richtungsvektoren $(\mathbf{R}, \mathbf{N}, \mathbf{H}, \dots)$ (Environment Mapping)
- Funktion der obigen Parameter (z.B. in „Shadern“=pro Pixel Berechnung)

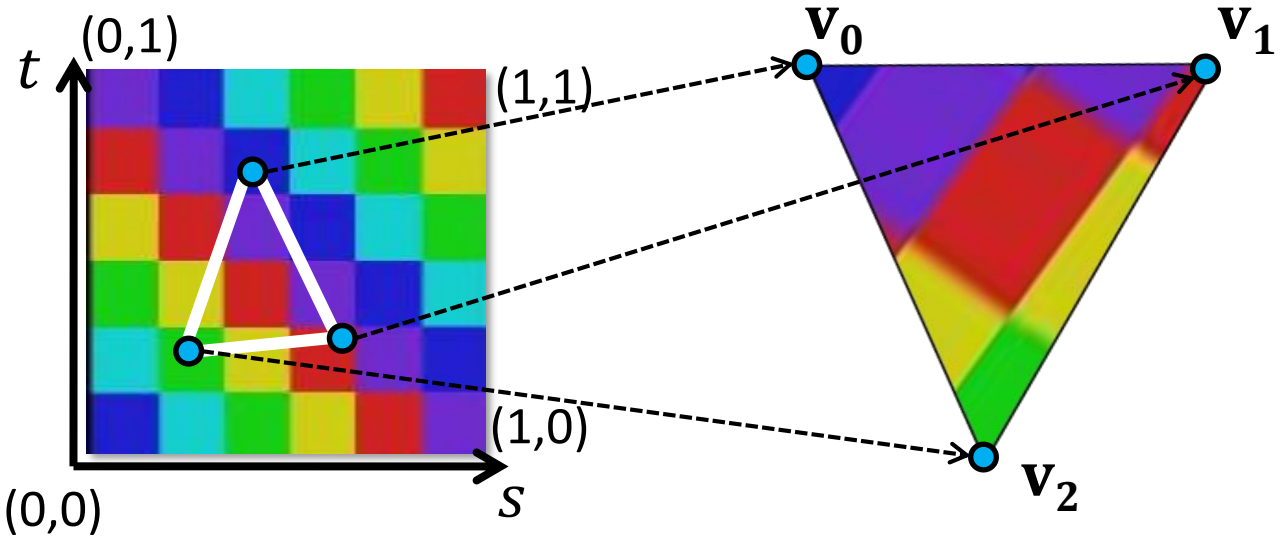


Texturkoordinaten für Dreiecksnetze

Explizites Speichern der Parametrisierung

- bisher: Vorgabe der Berechnungsvorschrift, die die Texturkoordinate für einen Punkt auf der Oberfläche bestimmt

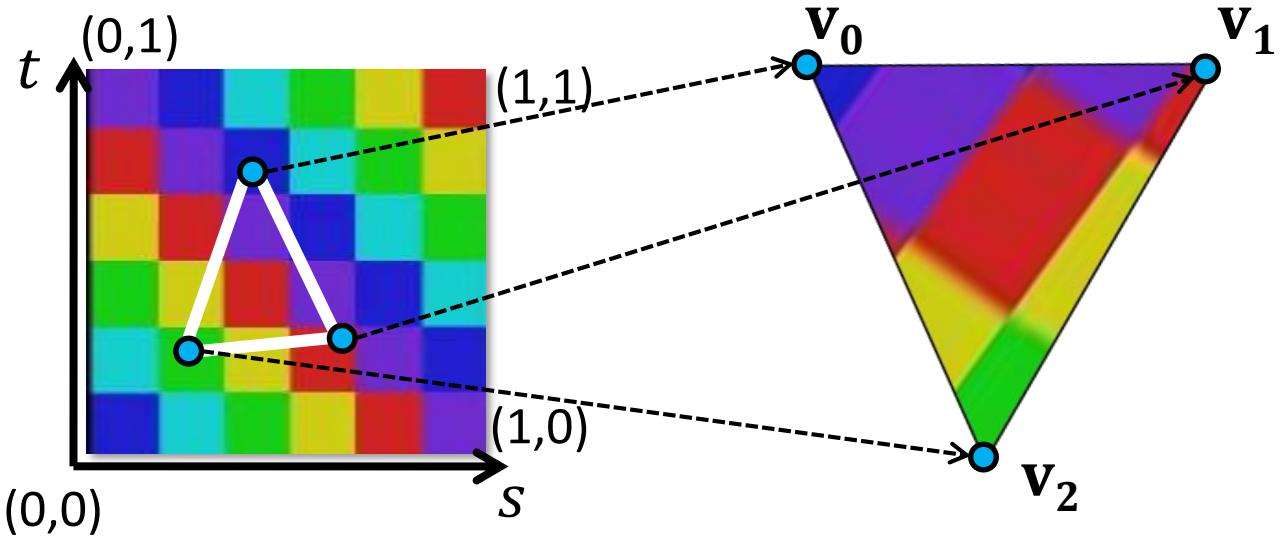
- bei Dreiecksnetzen geht man üblicherweise anders vor:
  - Parametrisierung wird in **Texturkoordinaten** gespeichert
  - jedem Eckpunkt/Vertex $\mathbf{v_i} = (x_i, y_i, z_i)$ eines Dreiecks wird eine Texturkoordinate (s_i, t_i) zugewiesen



Texturkoordinaten für Dreiecksnetze

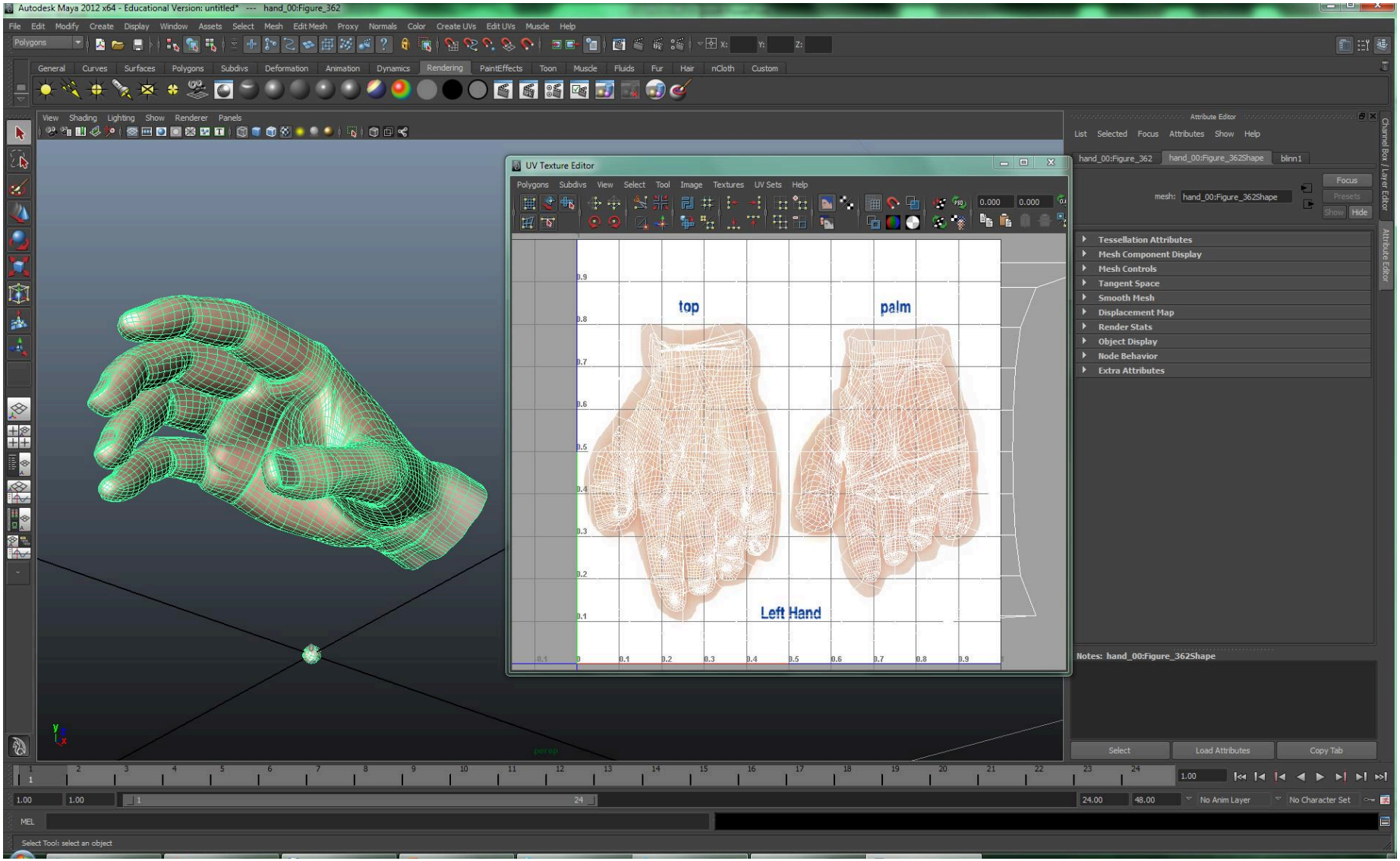
Explizites Speichern der Parametrisierung

- Parametrisierung in Texturkoordinaten gespeichert, indem jedem Eckpunkt $\mathbf{v}_i = (x_i, y_i, z_i)$ eine Texturkoordinate (s_i, t_i) zugewiesen wird
- Texturkoordinaten können
  - …für jeden Vertex wie bisher berechnet werden
  - …bei Triangulierung parametrischer Flächen erzeugt werden
  - …für Texturatlanten bestimmt werden
  - werden oft auch (semi-)manuell festgelegt



Texturkoordinaten für Dreiecksnetze

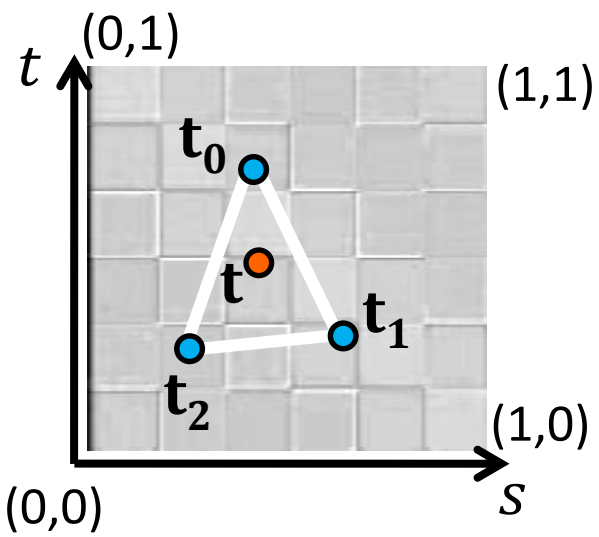
Texturkoordinaten

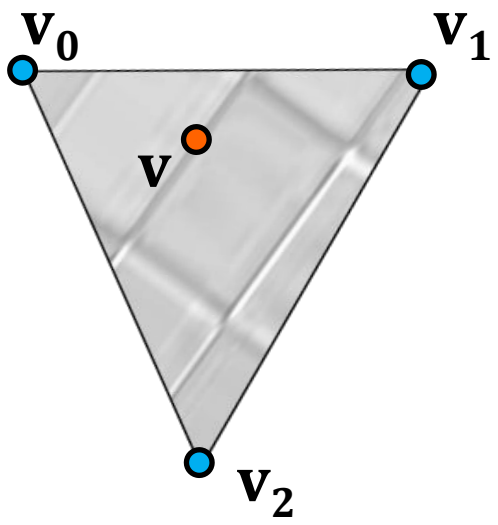


Texturkoordinaten für Dreiecksnetze

Texturkoordinaten

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



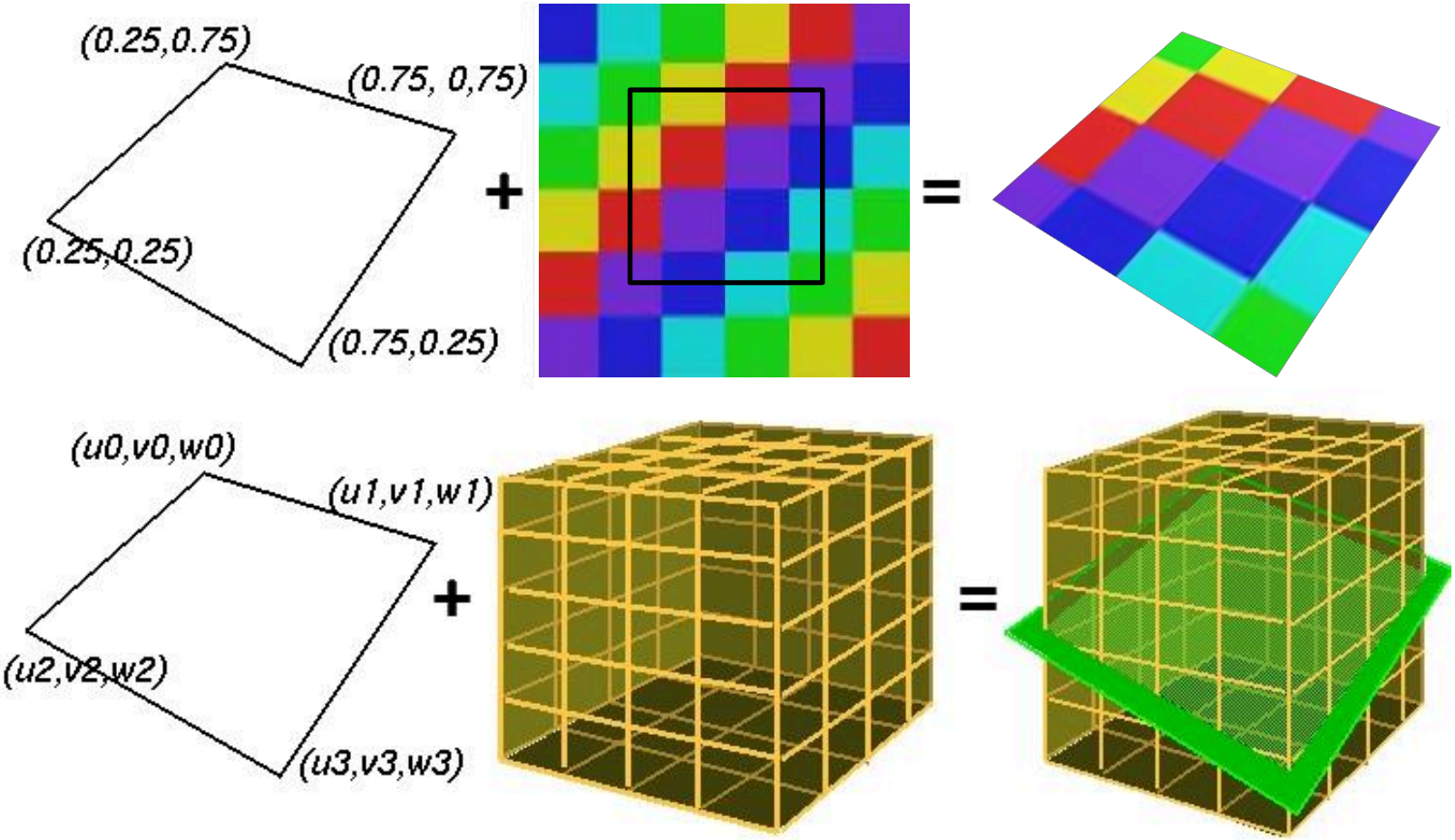


- **affine Abbildung** zwischen 2D Texturraum und 3D Objektraum:
  - Interpolation mit baryzentrischen Koord. (analog zu Farbe/Normale)

$$\mathbf{v} = \mathbf{v_0} + \lambda_2 \cdot (\mathbf{v_1} - \mathbf{v_0}) + \lambda_3 \cdot (\mathbf{v_2} - \mathbf{v_0})$$
$$\mathbf{t} = \mathbf{t_0} + \lambda_2 \cdot (\mathbf{t_1} - \mathbf{t_0}) + \lambda_3 \cdot (\mathbf{t_2} - \mathbf{t_0})$$

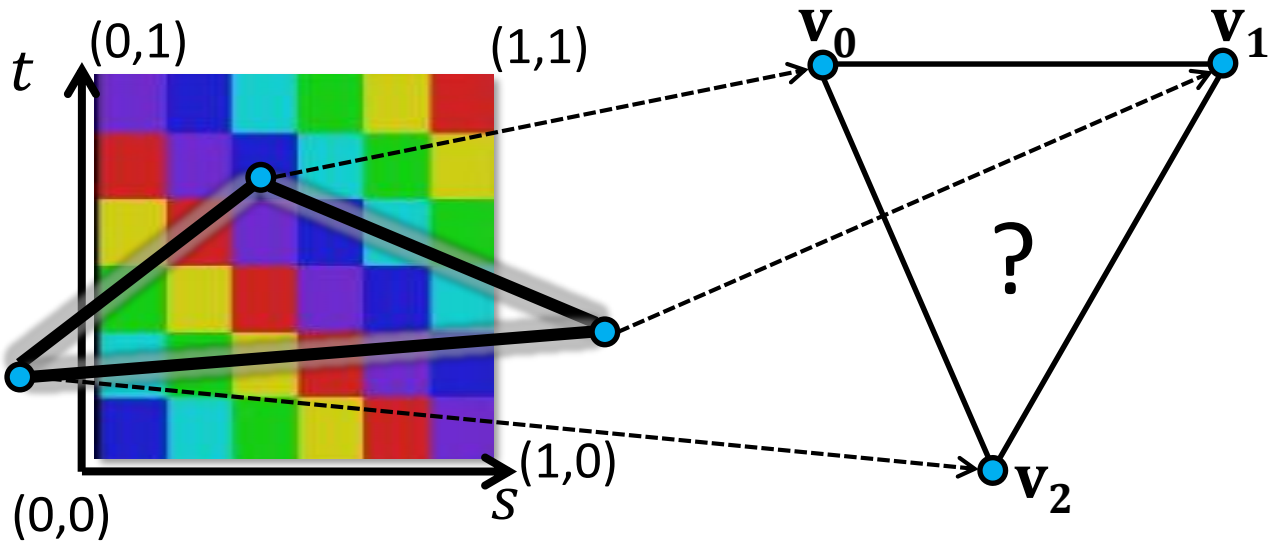
- → stückweise lineare Approximation einer Parametrisierung!
  (es sei denn die Parametrisierung ist selbst schon linear)

Texturkoordinaten in 2D und 3D



Texturkoordinaten

- was passiert bei Texturkoordinaten > 1.0 oder < 0.0?

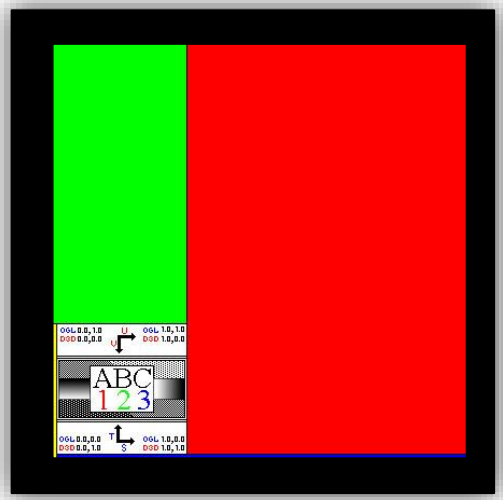


- es gibt eine Reihe verschiedener „Adressierungsmodi“, die in der CG verwendet werden (und von Grafik-Hardware unterstützt werden)

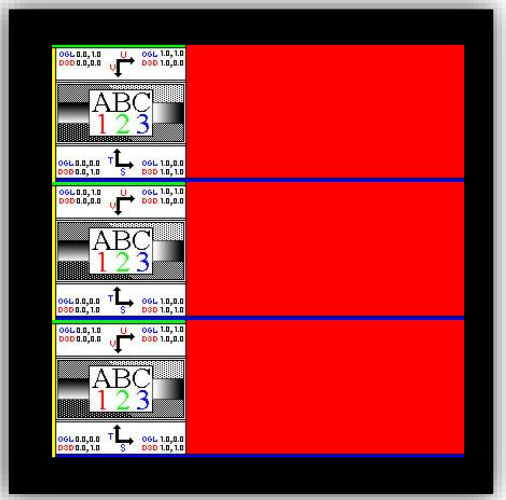
Texture Wrapping

- Repeat/Wrapping: Fortsetzen/Kacheln einer Textur über $[0,1]$ hinaus
- Adressierung wird für jede Dimension separat gewählt
- Texturkoordinaten des Quadrats hier $[0,3]^2$

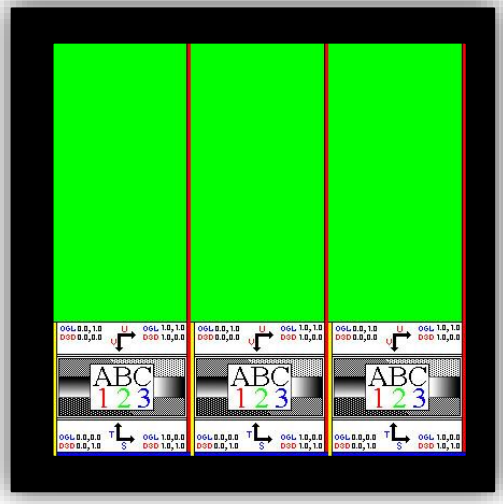
clamp/
clamp



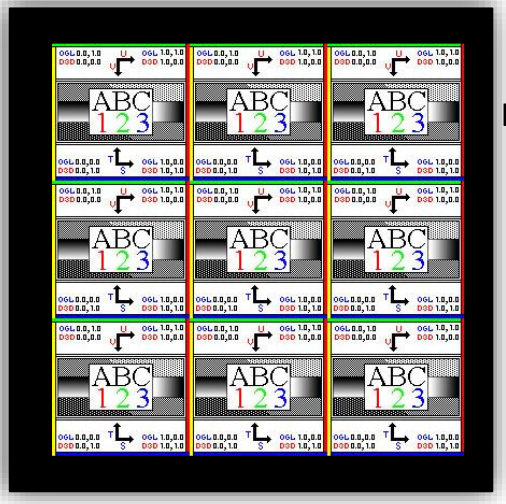
clamp/
repeat



repeat/
clamp



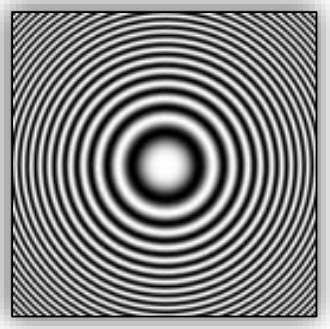
repeat/
repeat

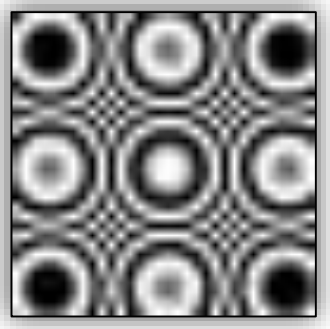


weitere:
clamp to border,
mirror, …

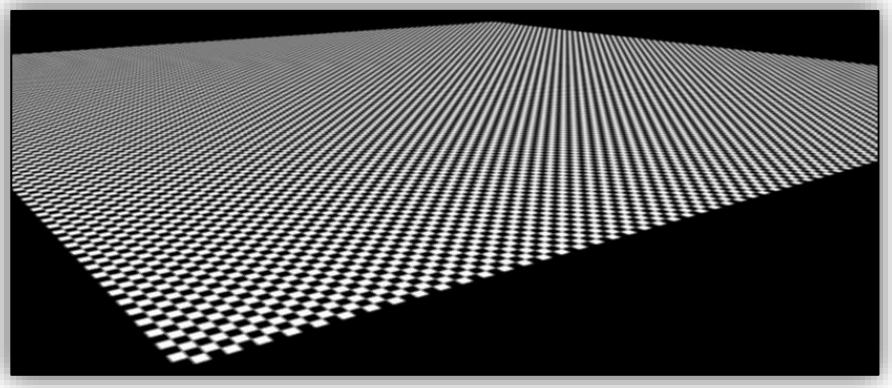
Texture Mapping und Aliasing

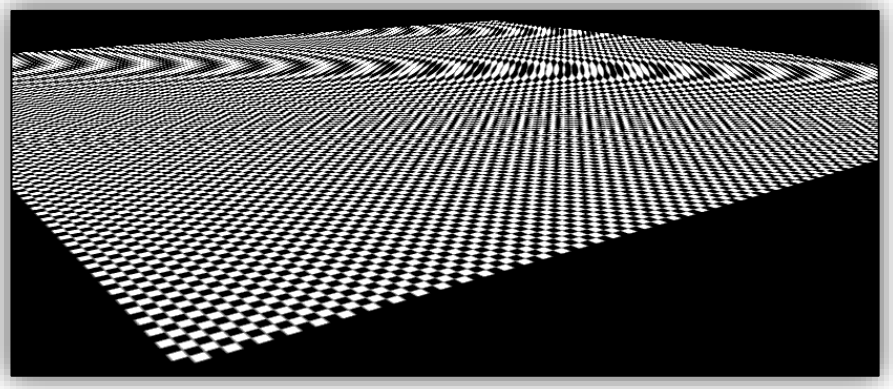
- wir kennen Aliasing schon von der Abtastung von Signalen
  - hier (Unter-)Abtastung einer kontinuierlichen Funktion:





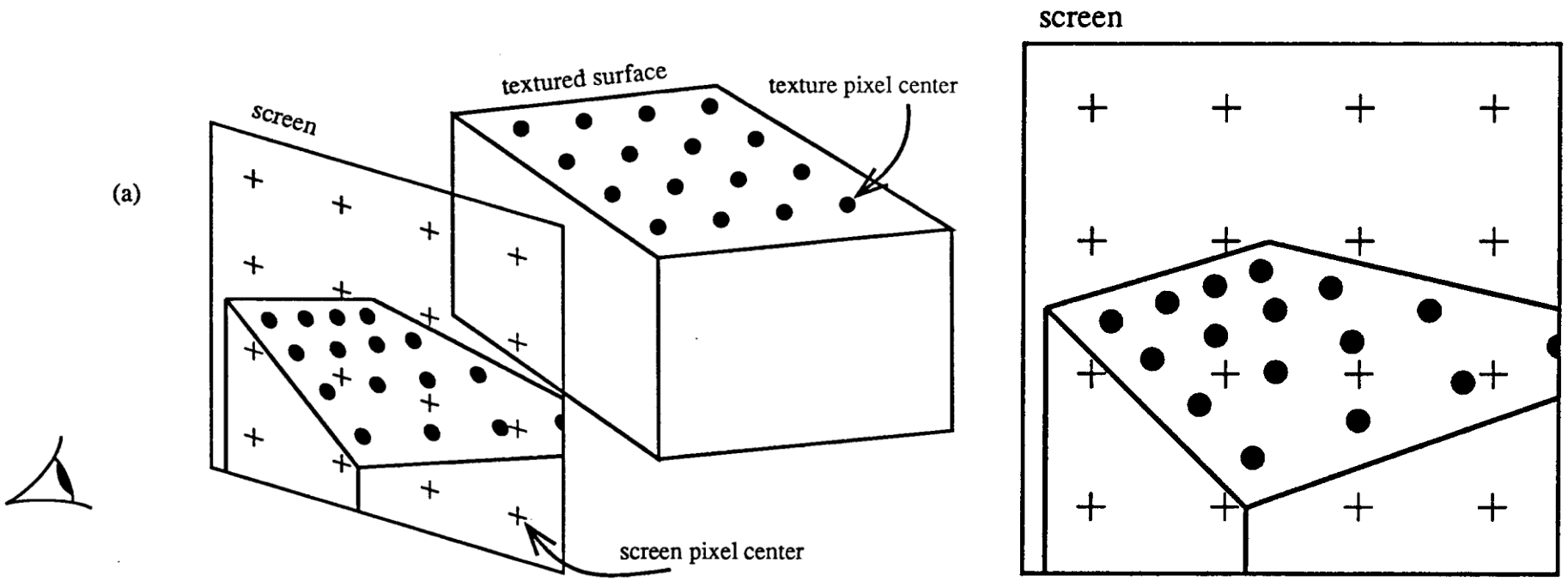
- das gleiche Problem tritt beim Texture Mapping auf
  - Textur = (meist diskrete) Funktion, abgetastet bei der Bilderzeugung





Texture Mapping und Aliasing

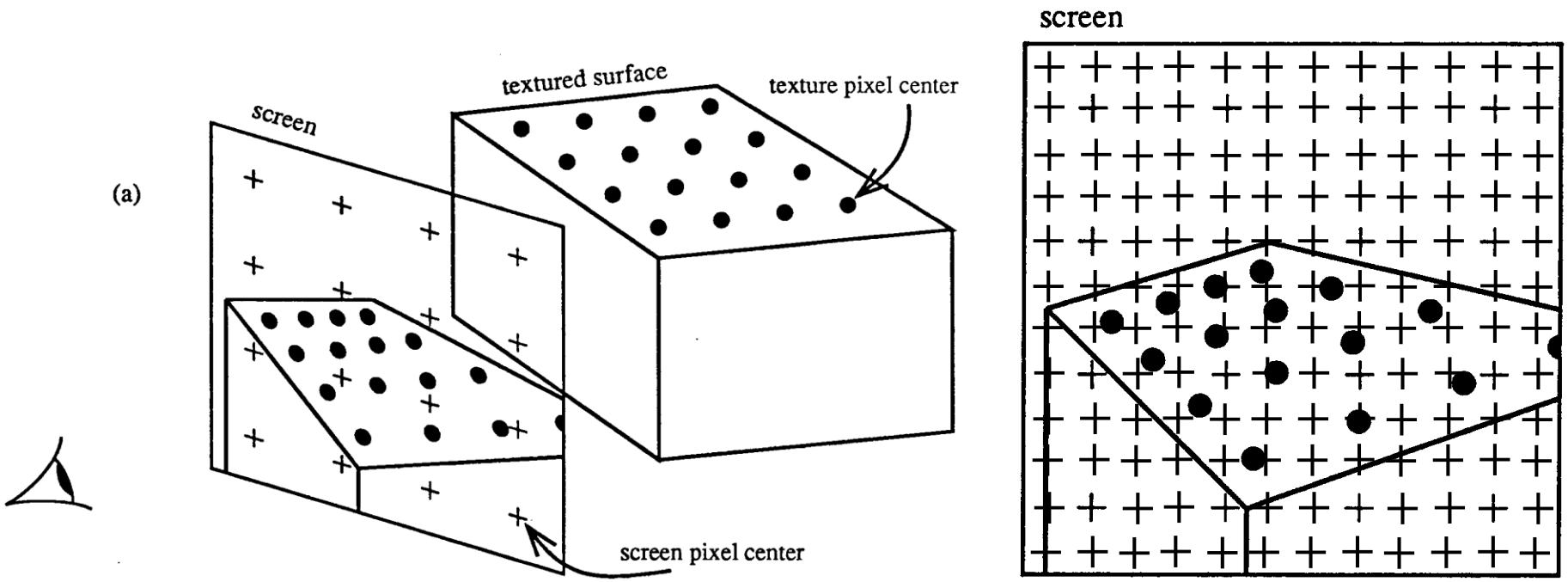
- Abbildungen des Textursignals
  - 2D Texturraum ↔ 3D Objektraum: Parametrisierung
  - 3D Objektraum ↔ 2D Bildraum: Projektion/Perspektive
  - hier gezeigt für 2D-Texturen, das Abtastproblem tritt analog für 1D- und 3D-Texturen auf



Fundamentals of Texture Mapping and Image Warping,
Paul Heckbert, Master's thesis, UCB/CSD 89/516,
CS Division, U.C. Berkeley, June 1989 [12]

Texture Mapping und Aliasing

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

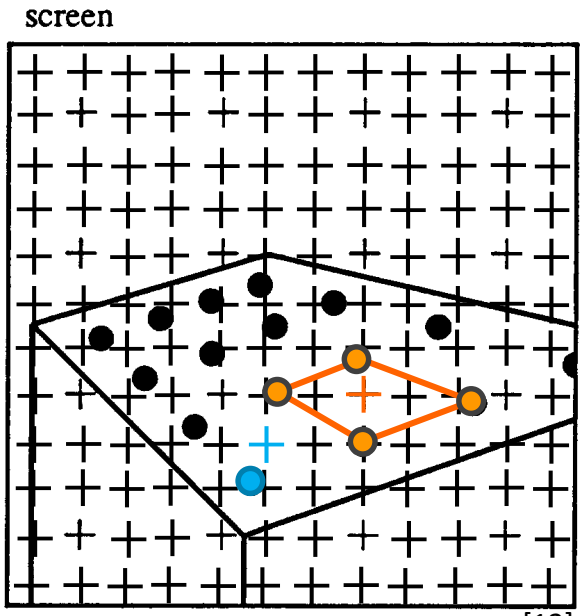


Fundamentals of Texture Mapping and Image Warping,
Paul Heckbert, Master's thesis, UCB/CSD 89/516,
CS Division, U.C. Berkeley, June 1989 [12]

Textur-Filterung

Vergrößerung (Magnification)

- Abbildung weniger Texel auf viele Pixel
- **Nearest Neighbor** (links)
  - verwende Farbe des nächstliegenden Texels
- **Bilineare Interpolation** (rechts)
  - Interpolation der 4 nächsten Texel, gemäß der relativen Abstände
  - dadurch wird das Textursignal geglättet



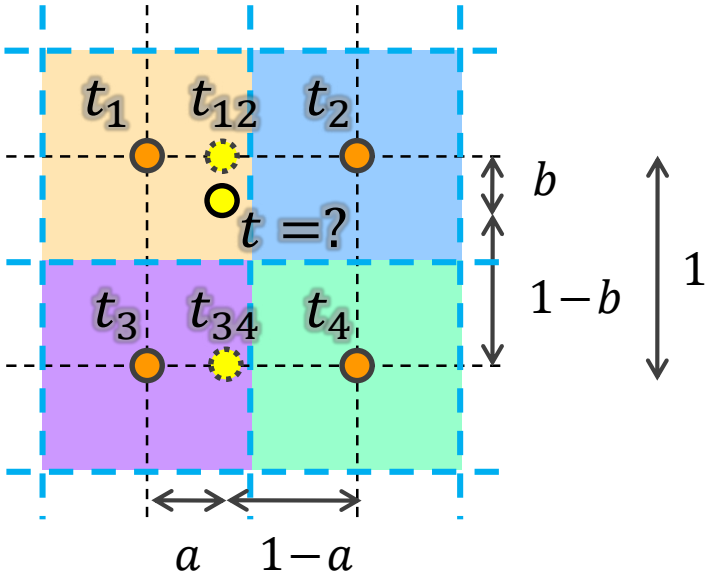
[12]

[13]

Vergrößerung/Magnification

Bilineare Interpolation (im Texturraum)

- 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

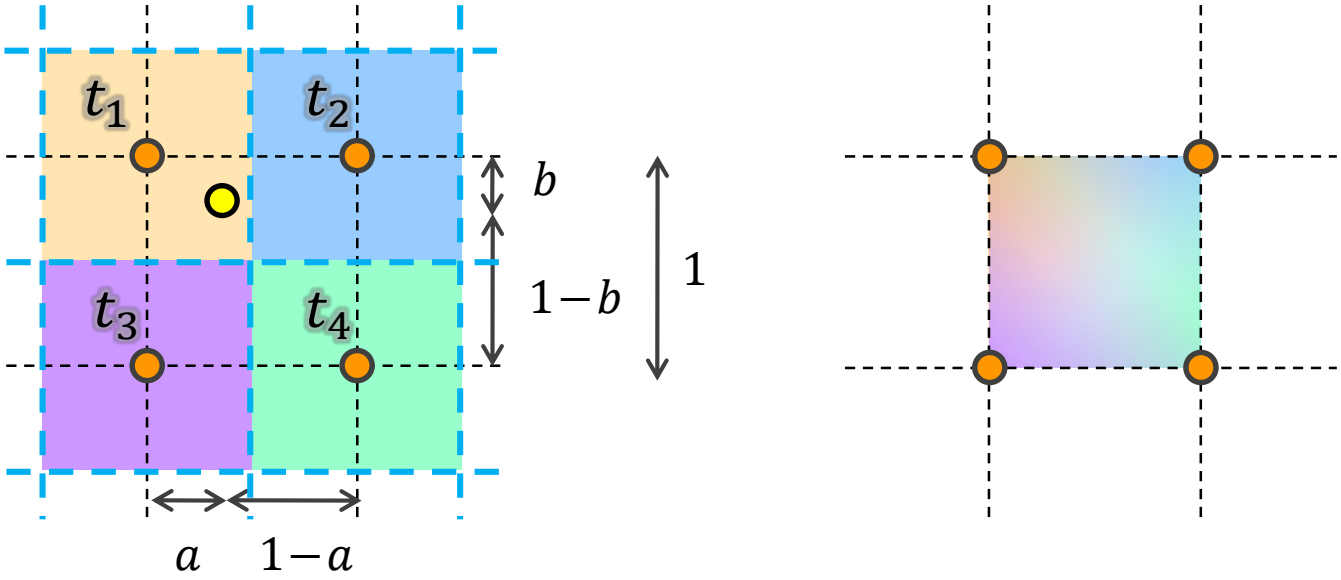


- zweimal lineare Interpolation: horizontal, dann vertikal (oder umgekehrt)
  - $t_{12} = t_1 + a(t_2 - t_1) = (1-a)t_1 + at_2$
  - $t_{34} = (1-a)t_3 + at_4$
  - $t = (1-b)t_{12} + bt_{34}$

Vergrößerung/Magnification

Bilineare Interpolation (im Texturraum)

- 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

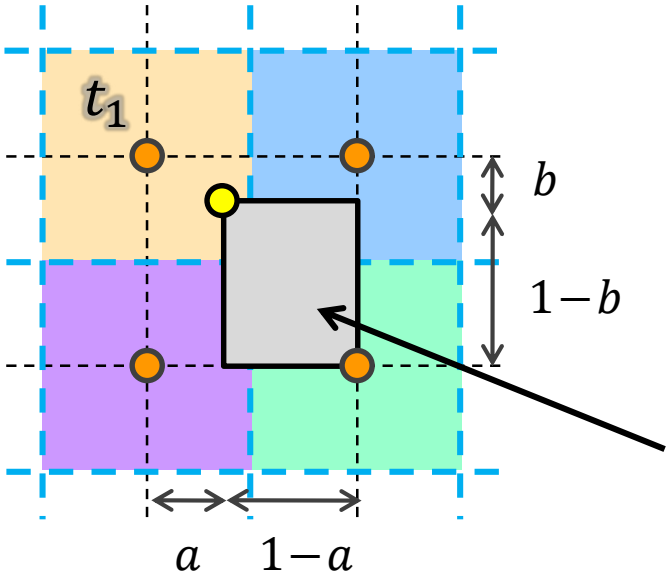


- Farbe = $(1-a)(1-b) +$ $a(1-b) +$ $(1-a)b +$ ab

Vergrößerung/Magnification

Bilineare Interpolation (im Texturraum)

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

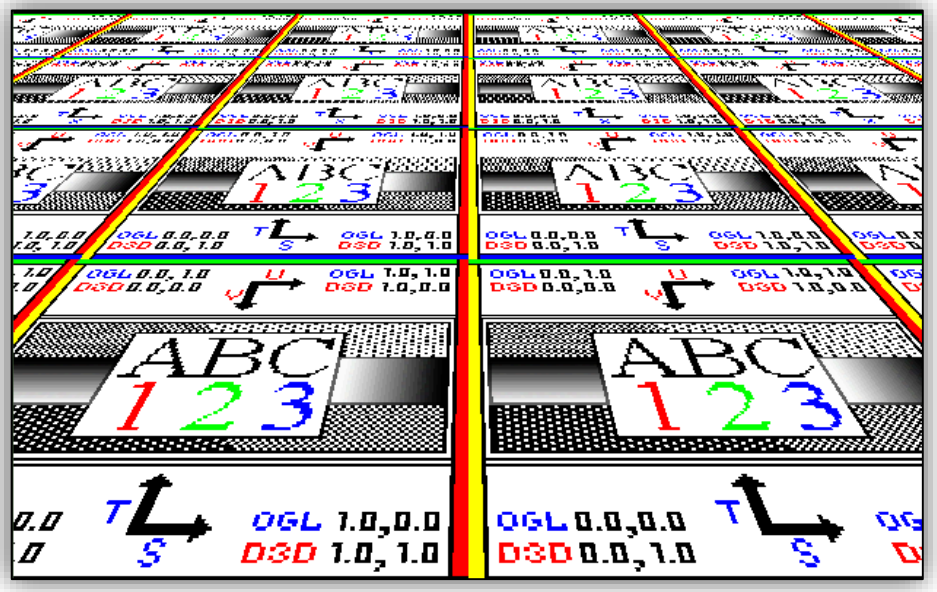


Flächeninhalt ist relatives Gewicht des gegenüberliegenden Texels t_1

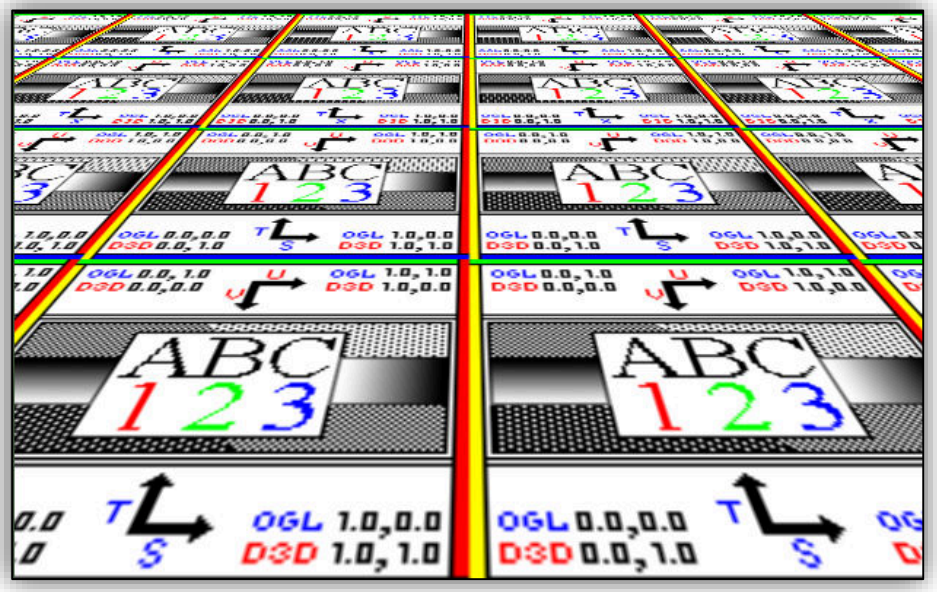
- Farbe = $(1-a)(1-b) + a(1-b) + (1-a)b + ab$

Vergrößerung/Magnification

- bilineare Interpolation ist nicht linear, sondern quadratisch (abgesehen von Interpolation in horizontaler und vertikaler Richtung)



Nearest Neighbor

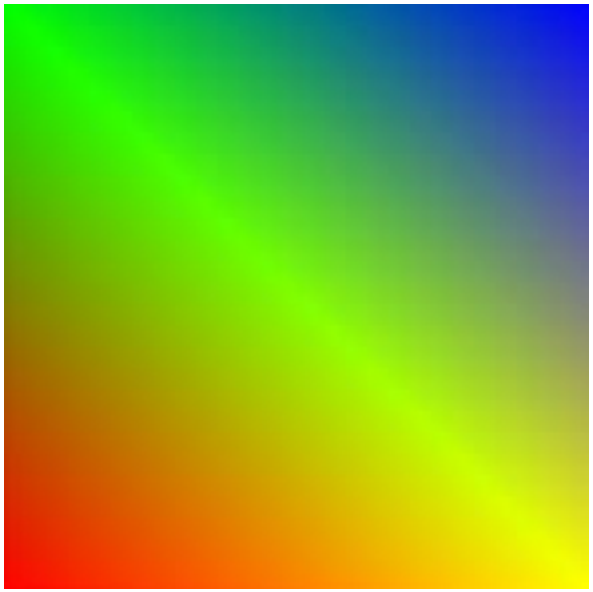
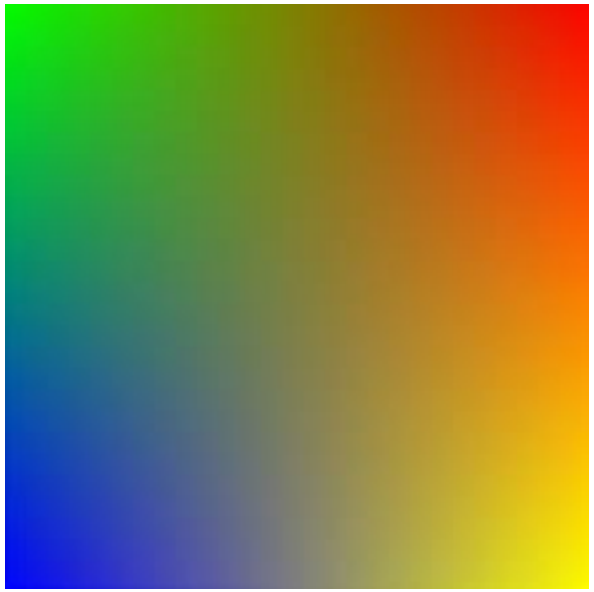


bilineare Interpolation

- welche Interpolation in einer 2D-Textur wäre denn dann linear?

Vergrößerung/Magnification

- welche Interpolation in einer 2D-Textur wäre denn dann linear?
  - … zwei lineare Interpolationen in Teildreiecken:
    es kommt auf die Wahl der Diagonale an (links, mitte)
  - diese Beispiel ist nicht praxis-relevant für Texturinterpolation – das Problem der Wahl der Diagonale in anderem Kontext schon
  - zum Vergleich: bilinear (rechts)

Vergrößerung/Magnification

- welche Interpolation in einer 2D-Textur wäre denn dann linear?
  - … zwei lineare Interpolationen in Teildreiecken:
    es kommt auf die Wahl der Diagonale an (links, mitte)
  - diese Beispiel ist nicht praxis-relevant für Texturinterpolation – das Problem der Wahl der Diagonale in anderem Kontext schon:
    z.B. bei der Triangulierung von Höhenfeldern (Terrain Visualisierung)

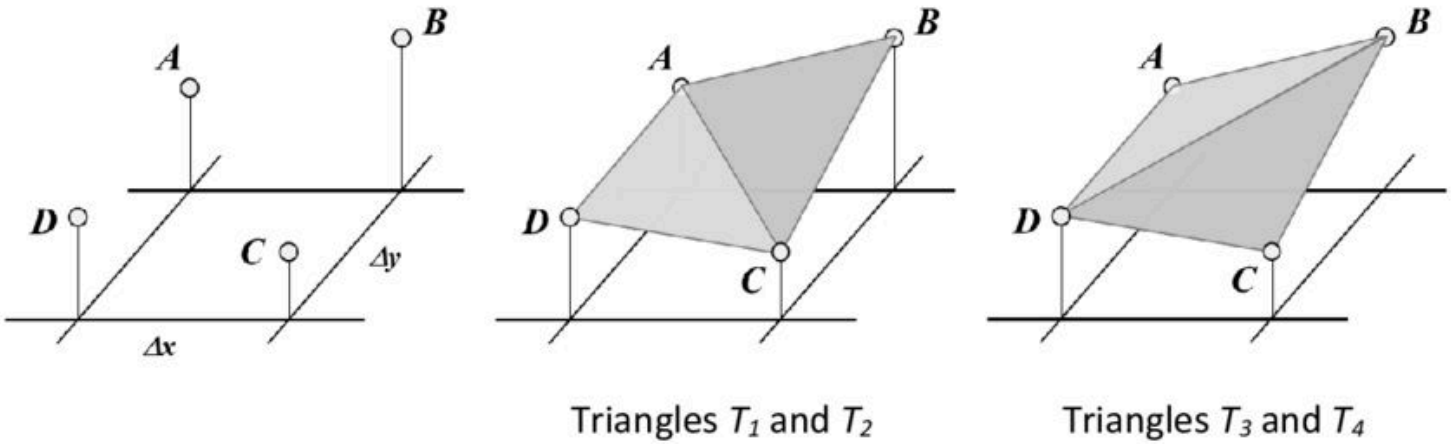


Bild: https://www.researchgate.net/figure/calculating-the-area-by-triangulation_fig6_279185907

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)

Differenz ×4

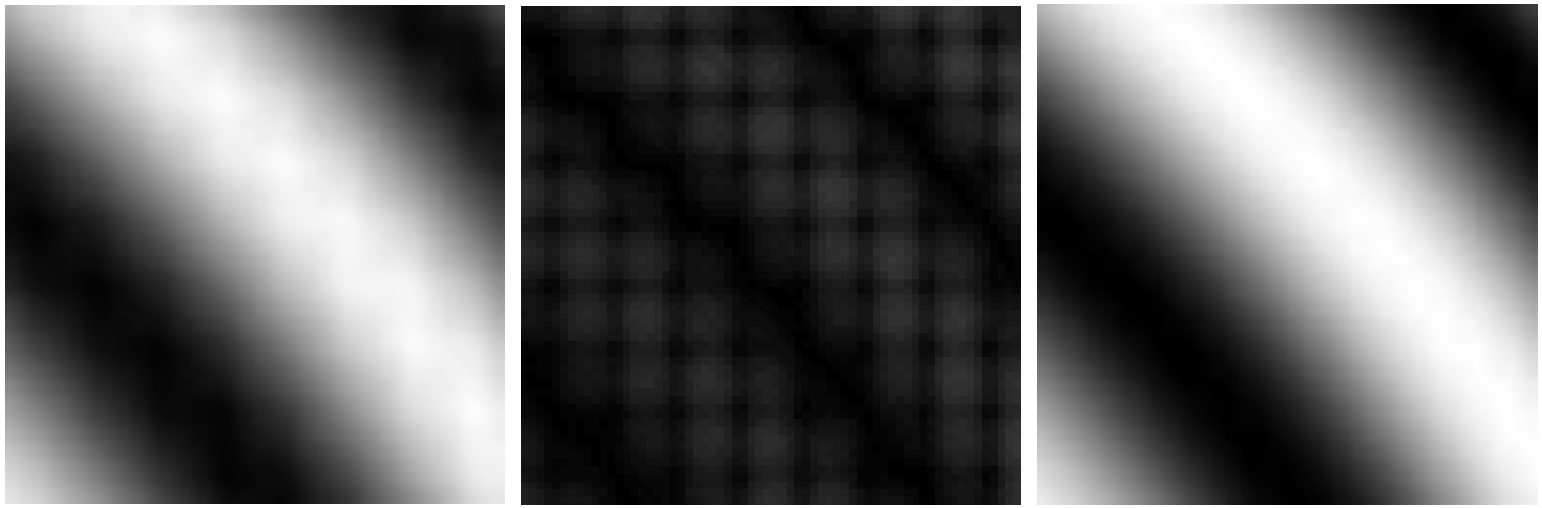


Bild aus: Hardware Adaptive High-Order Interpolation for Real-Time Graphics, Lin et al., High Performance Graphics 2021

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)

Differenz ×4

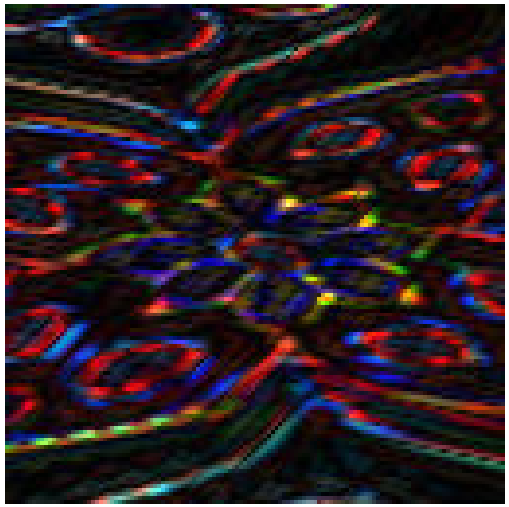
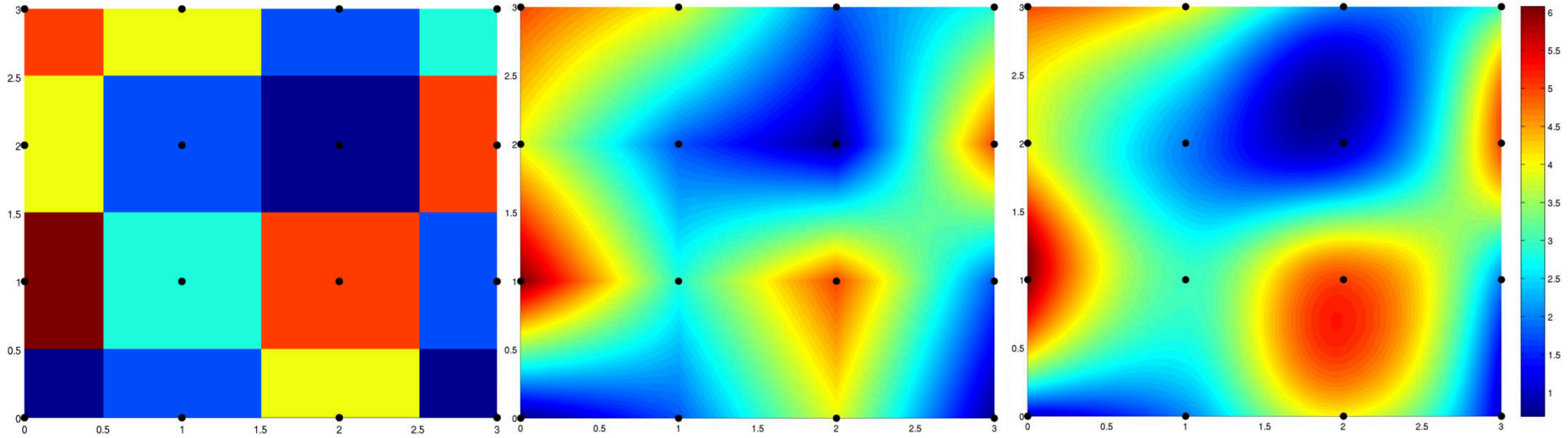


Bild aus: Hardware Adaptive High-Order Interpolation for Real-Time Graphics, Lin et al., High Performance Graphics 2021

Textur-Filterung

Beispiel: Nearest Neighbor, bilineare und bikubische Interpolation

- dieses Beispiel zeigt keine Filterung von Texel-Farben, sondern die Interpolation eines Skalarwerts, der anschließend mit der Farbskala rechts abgebildet wird

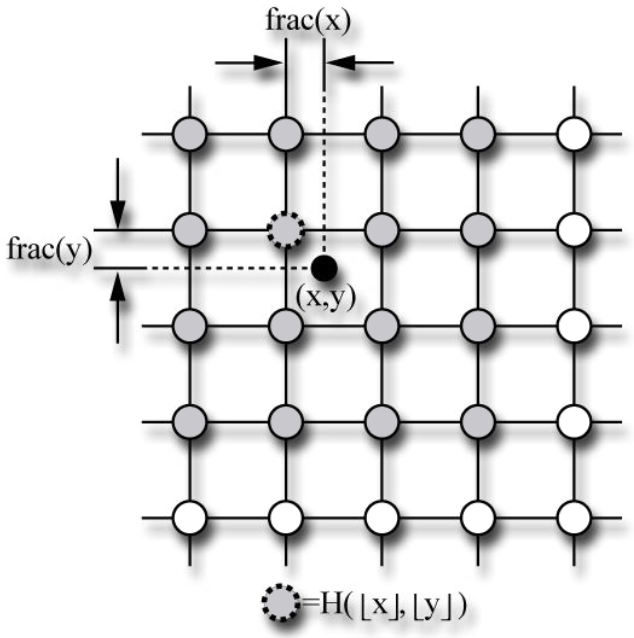


[15]

Textur-Filterung

Texturfilter höherer Ordnung

- Beispiel: bikubische Interpolation (nur damit Sie es mal gesehen haben)
  - benötigt Zugriff auf 4×4 Texel bei einer 2D-Textur
  - benötigt Zugriff auf $4^3 = 64$ Texel bei einer 3D-Textur
  - teuer und nicht nativ von Grafik-Hardware unterstützt (Approximation siehe Lin et al.)



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

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

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

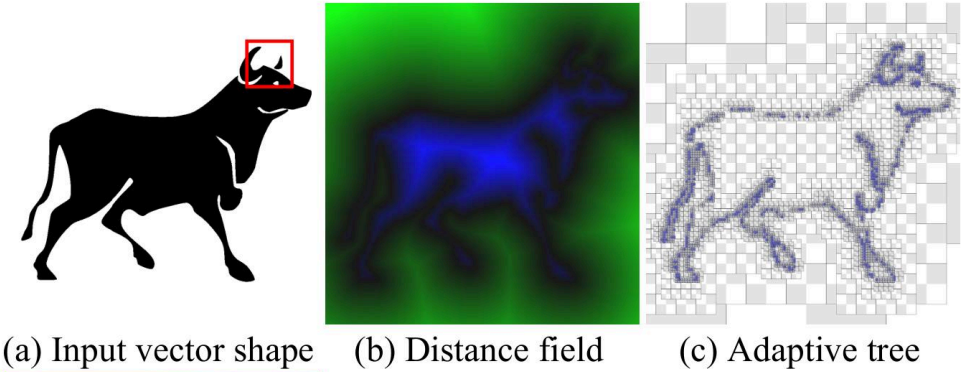
Textur-Filterung

Scharfe Kanten mit Texturen?

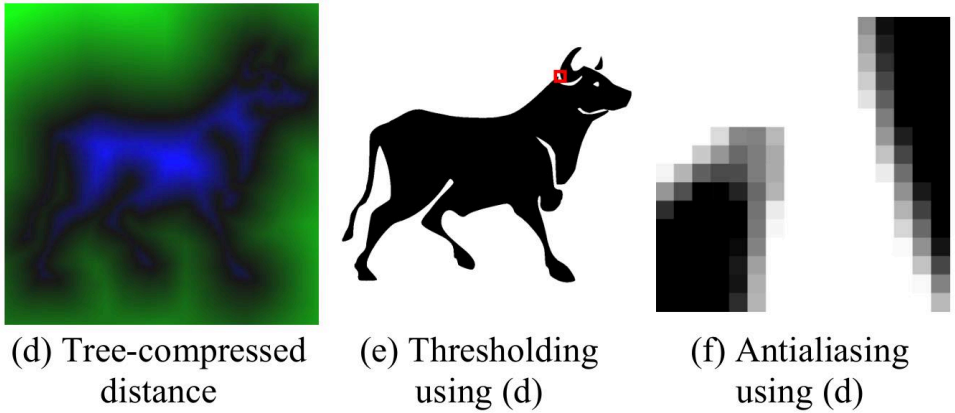
- bei Farbtexturen nur mit Zusatzaufwand
- bei Binärbildern: Umweg über sogenannte Distanzfelder

[14]

Compressed Random-Access Trees for Spatially Coherent Data, Lefebvre, Hoppe, 2007 (links) [16]
Improved Alpha-Tested Magnification for Vector Textures and Special Effects, Greene, 2007 (rechts)



(a) Input vector shape (b) Distance field (c) Adaptive tree



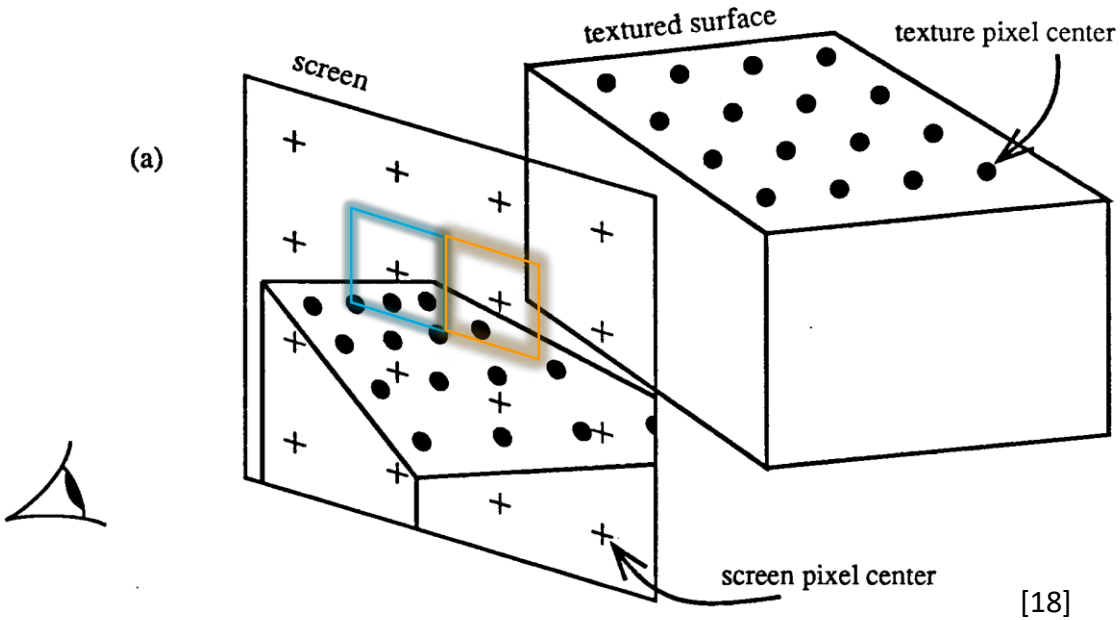
(d) Tree-compressed distance (e) Thresholding using (d) (f) Antialiasing using (d)

[17]

Textur-Filterung

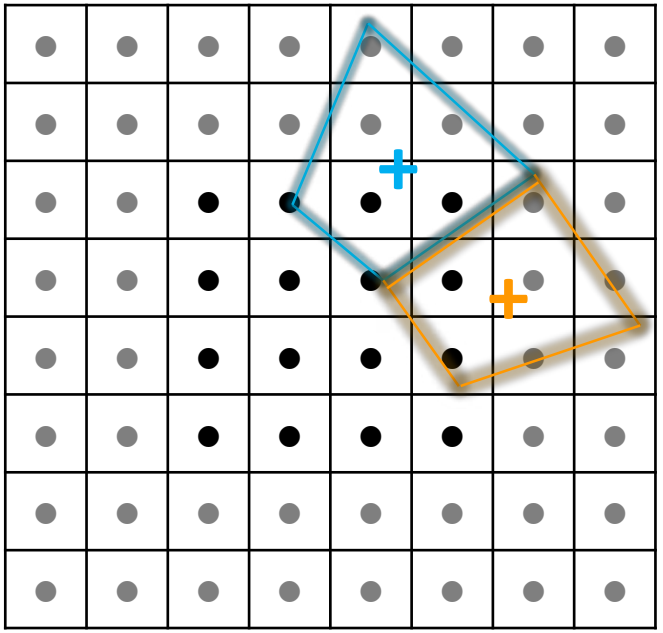
Verkleinerung (Minification)

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



[18]

Abdruck („Footprint“) eines Pixels im Texturraum

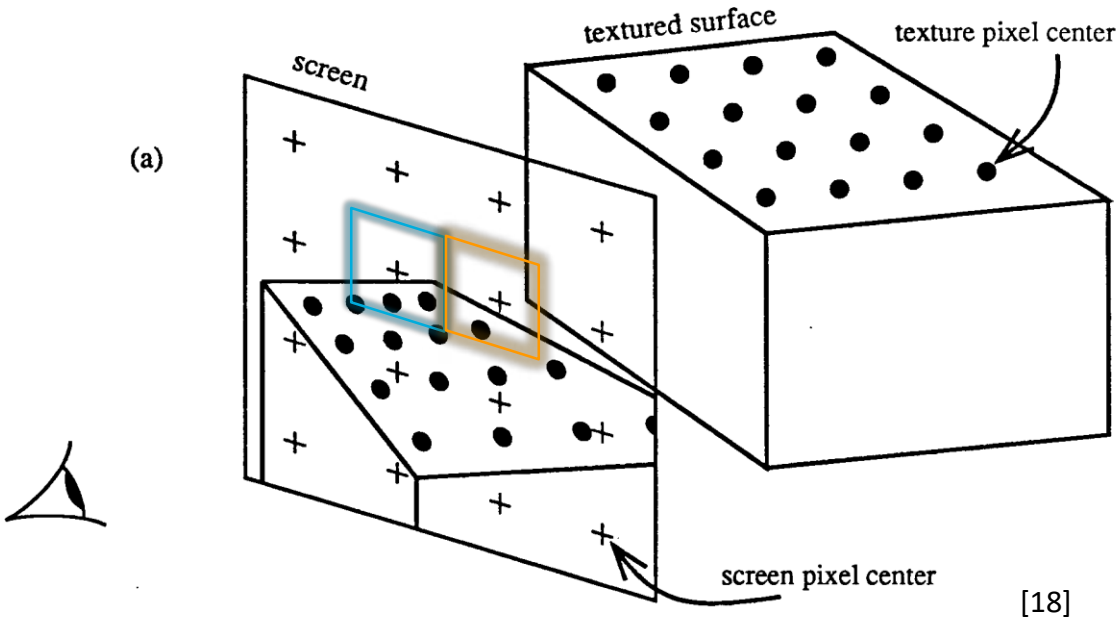


Textur

Textur-Filterung

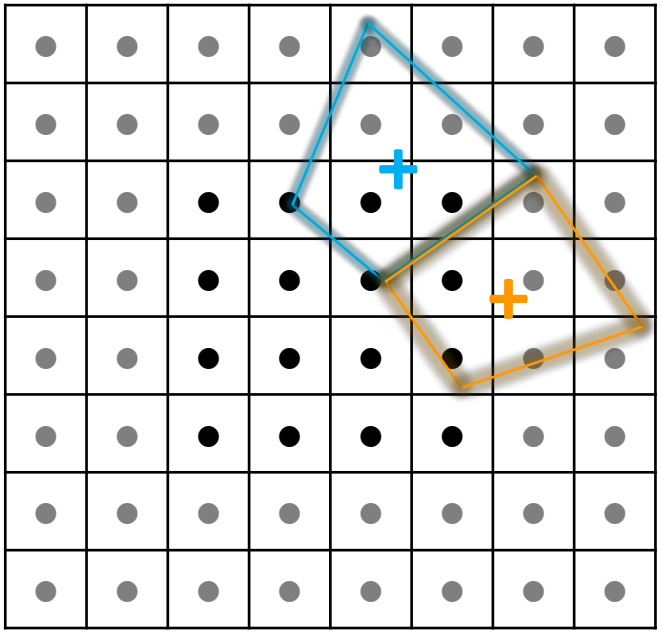
Verkleinerung (Minification)

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



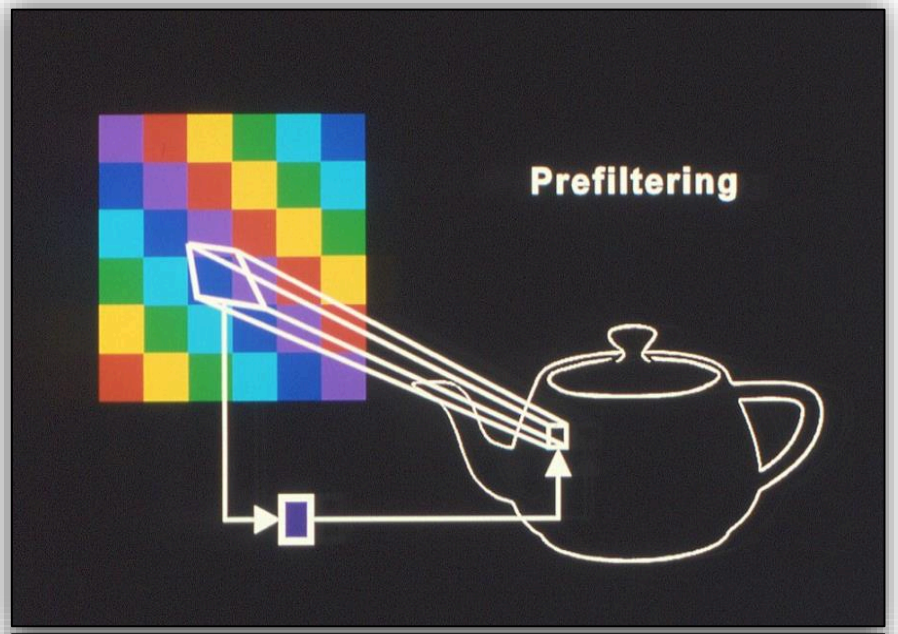
[18]

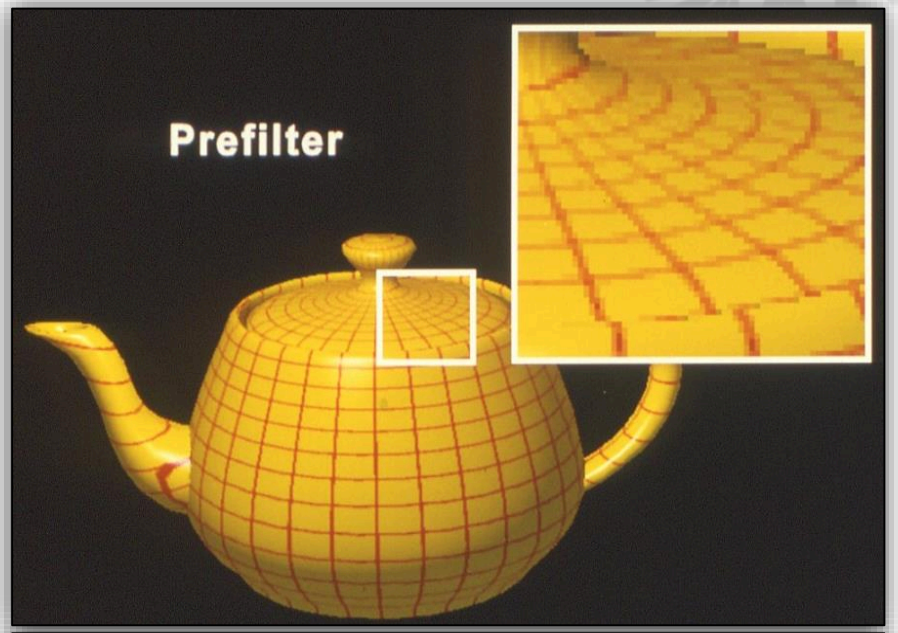
Abdruck („Footprint“) eines Pixels im Texturraum

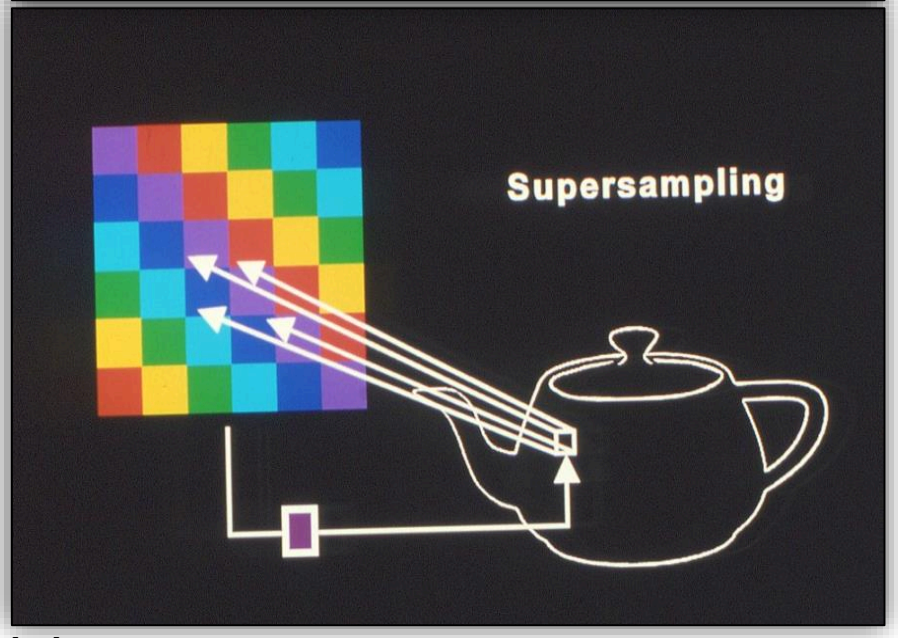


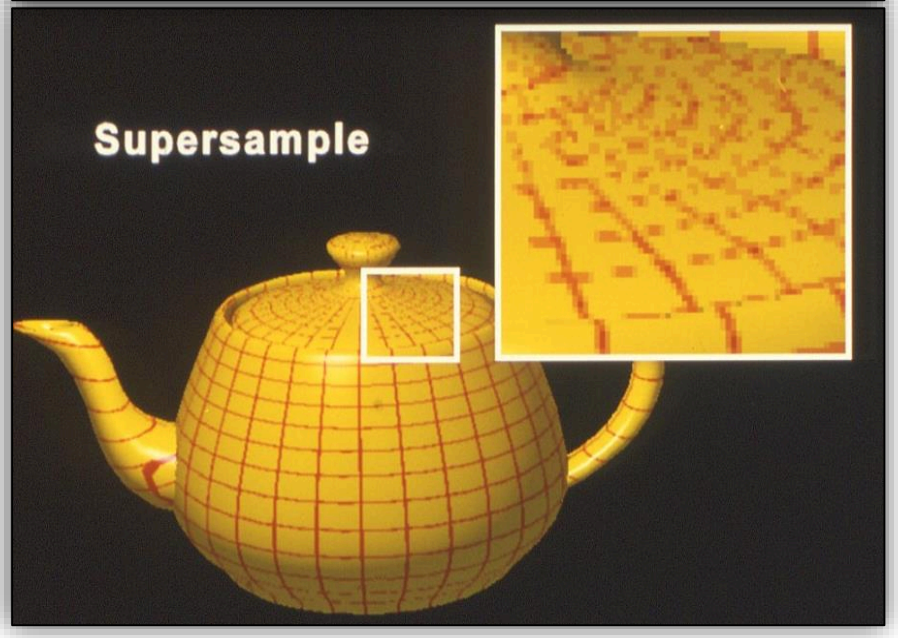
Textur

Aliasing: Lösungsstrategien?









[19]

ausreichende Abtastfrequenz

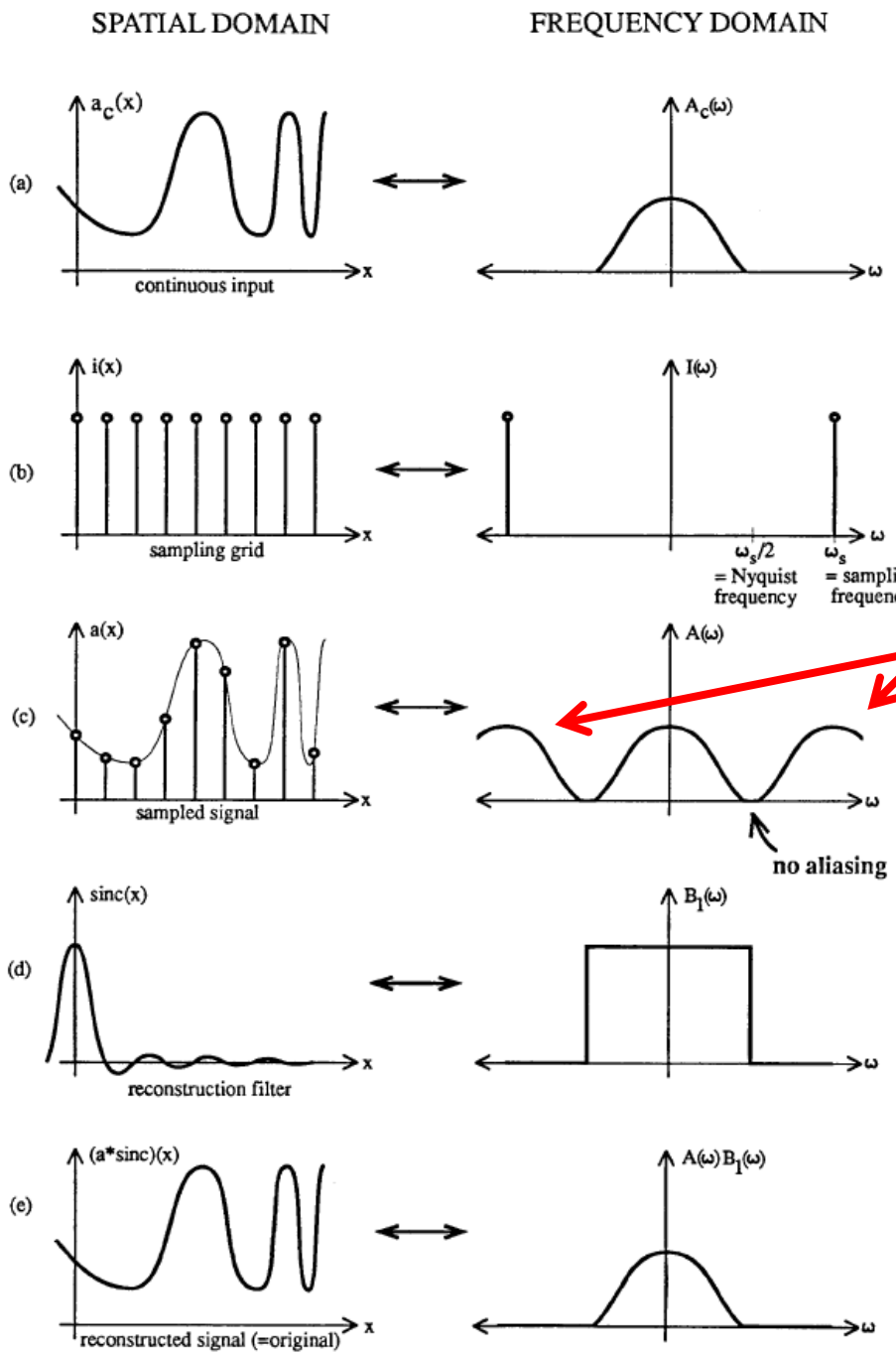


Figure 3.4: Sampling above the Nyquist rate (no aliasing). [20]

Woher kommen die Kopien?
period. Dirac im Ortsbereich→ Dirac im Frequenzbereich
Multiplikation im Ortsraum= Faltung um Frequenzraum

durch das Abtasten entstehen Kopien des Frequenzspektrums

hier kein Problem: sie überlappen sich nicht!

Faltung mit dem Rekonstruktionsfilter, idealer Tiefpaß-filter: Sinc

zu niedrige Abtastfrequenz

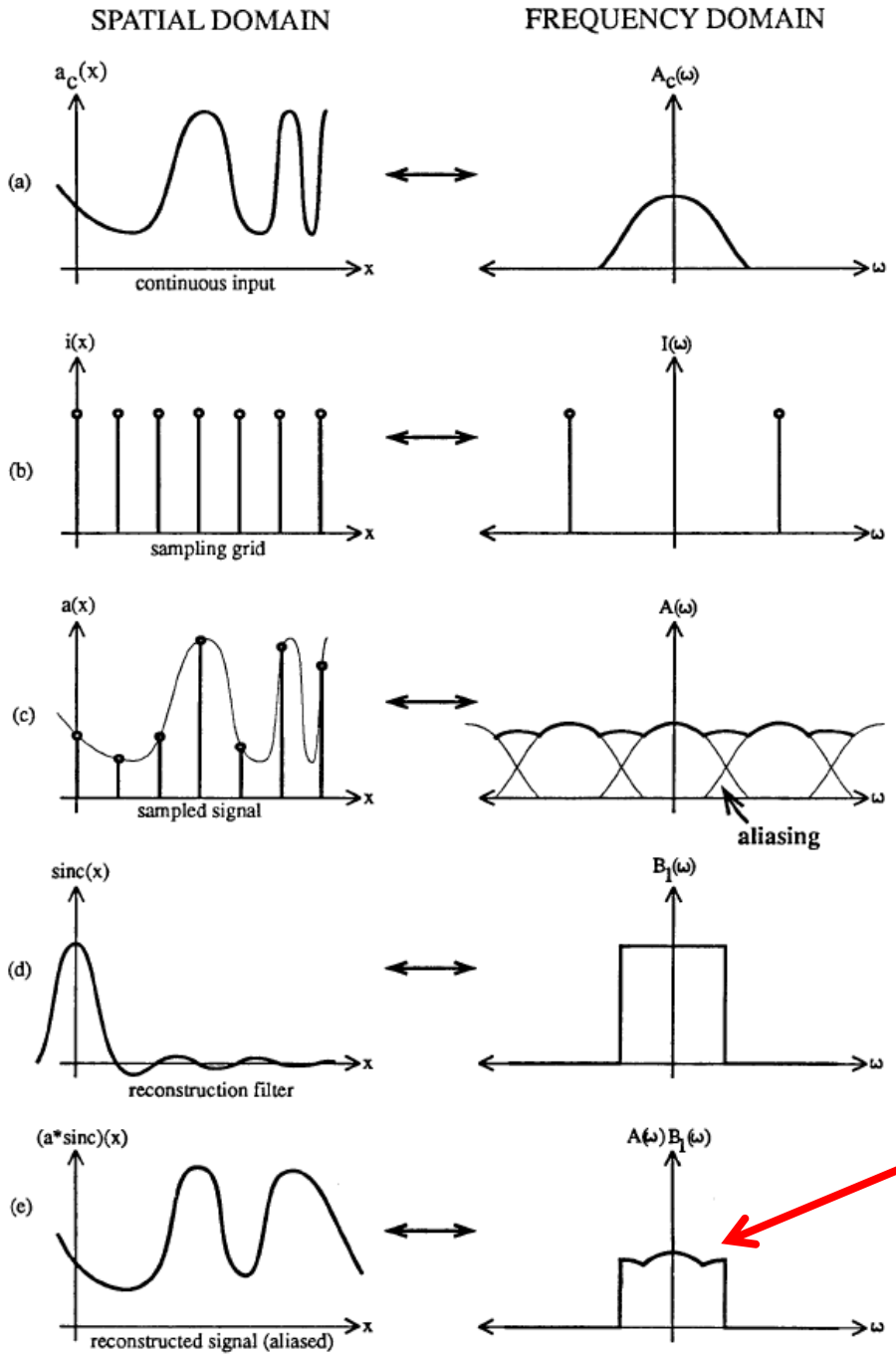
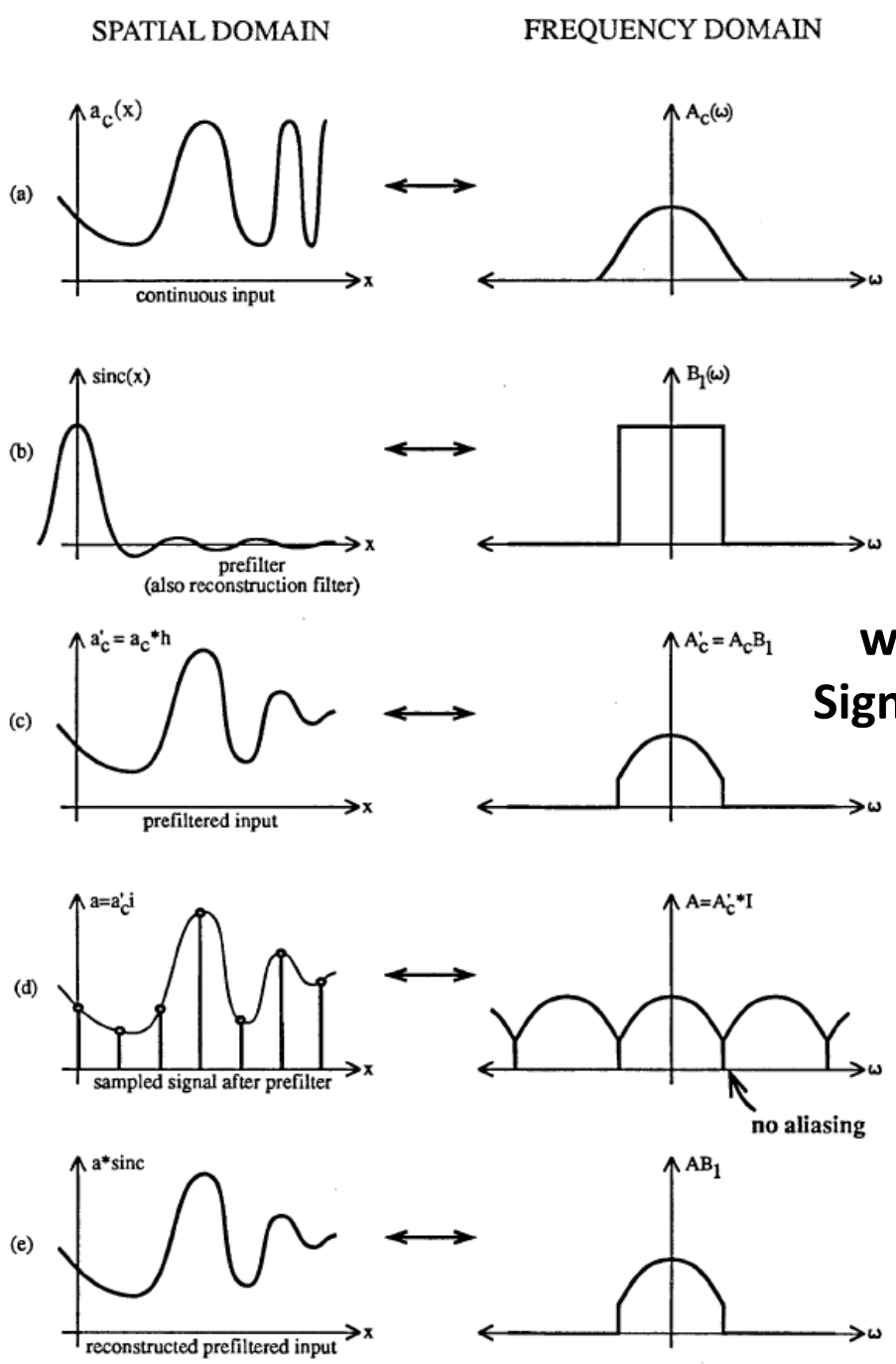


Figure 3.5: Sampling below the Nyquist rate (aliasing).

Überlappung!

Aliasing: Frequenzanteile, die hier nicht sein sollten!

Vorfilterung:
hier wird zuerst
das Signal gefiltert



wir erhalten ein neues Signal mit einem kleineren Frequenzspektrum

dann abtasten:
die Spektren über-
lappen nicht mehr!

[20]

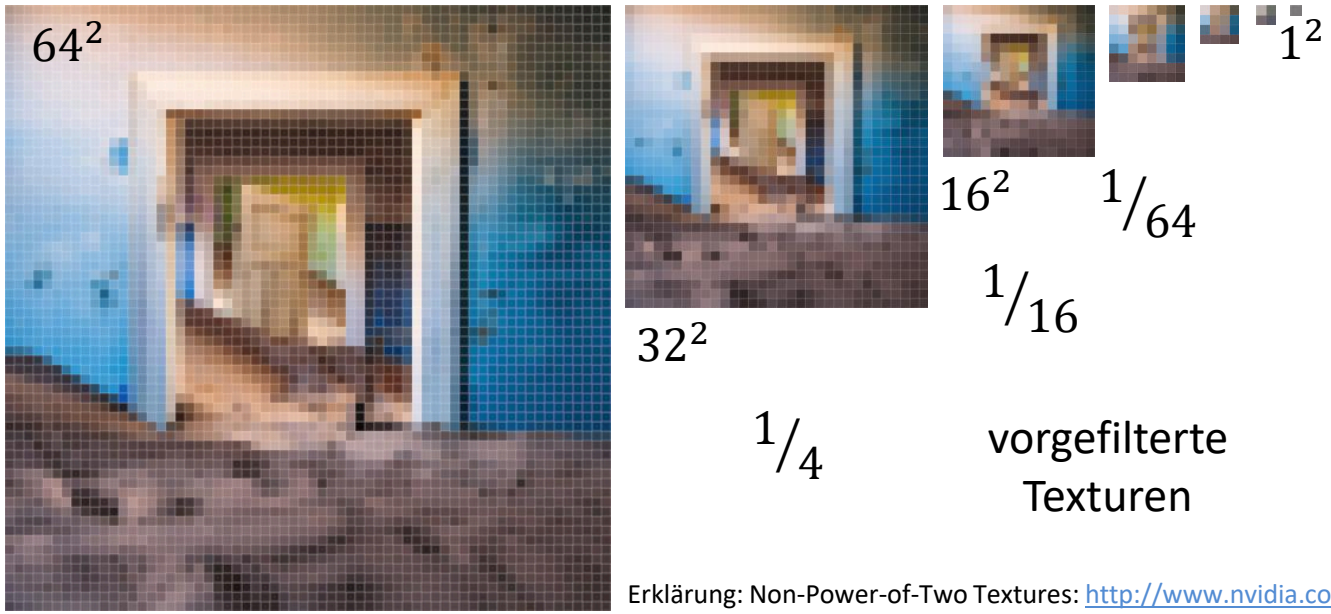
Figure 3.7: Ideal prefiltering eliminates aliasing.

Textur-Filterung

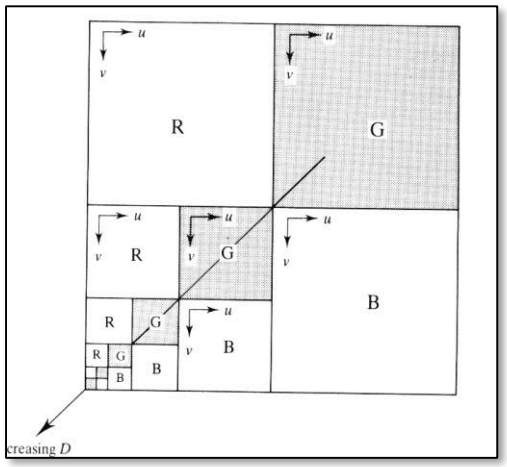
Mip-Mapping (lat. multum in parvo: viel in wenig)

- einfache Vorfilterung von Texturen
- speichere rekursiv Texturen mit $^1/_4$ Größe (Halbierung entlang jeder Achse) → „Auflösungspyramide“ mit nur 33% mehr Speicherbedarf: $1 + {}^1/_4 + {}^1/_{16} + {}^1/_{64} + \cdots = {}^4/_3$
- meist Mittelung über je 2×2 Texel (**kein** optimaler Tiefpass!)
- beachte: gleichzeitige Filterung und Auflösungsreduktion

Originaltextur

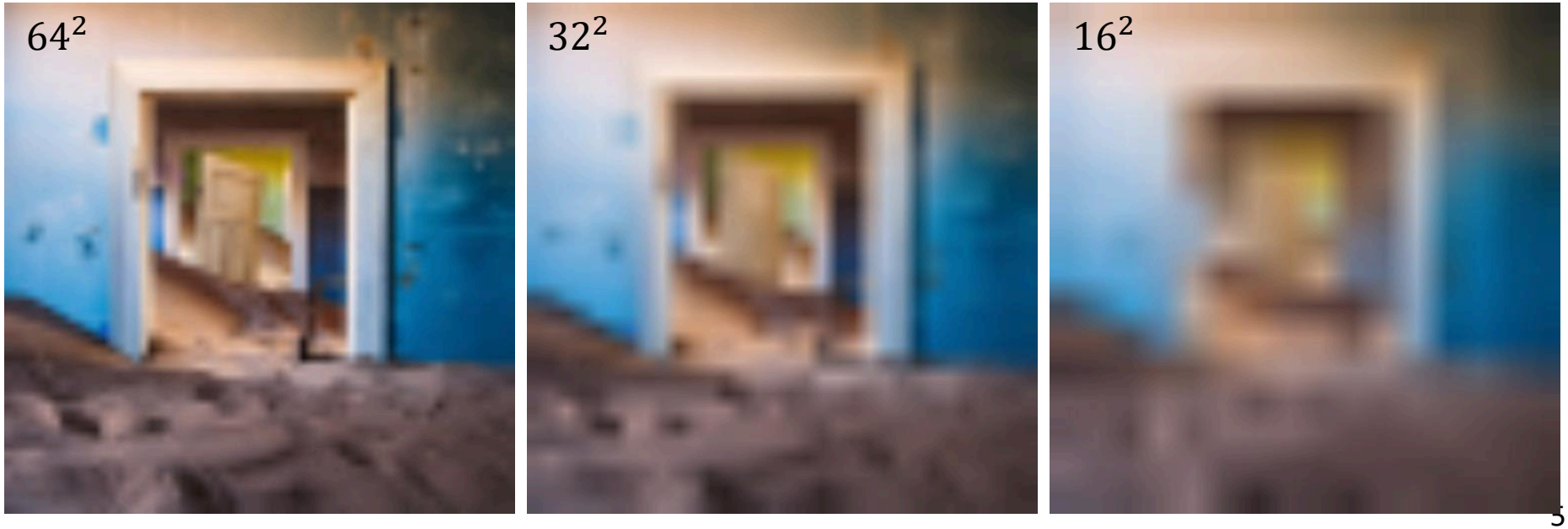


$^1/_3$ mehr Speicherbedarf



Erklärung: Non-Power-of-Two Textures: http://www.nvidia.com/object/np2_mipmapping.html
Bild: https://www.gettyimages.de/detail/foto/kolmanskop-ghost-town-lizenzfreies-bild/985903030

Mip-Mapping – Downsampling und Upsampling



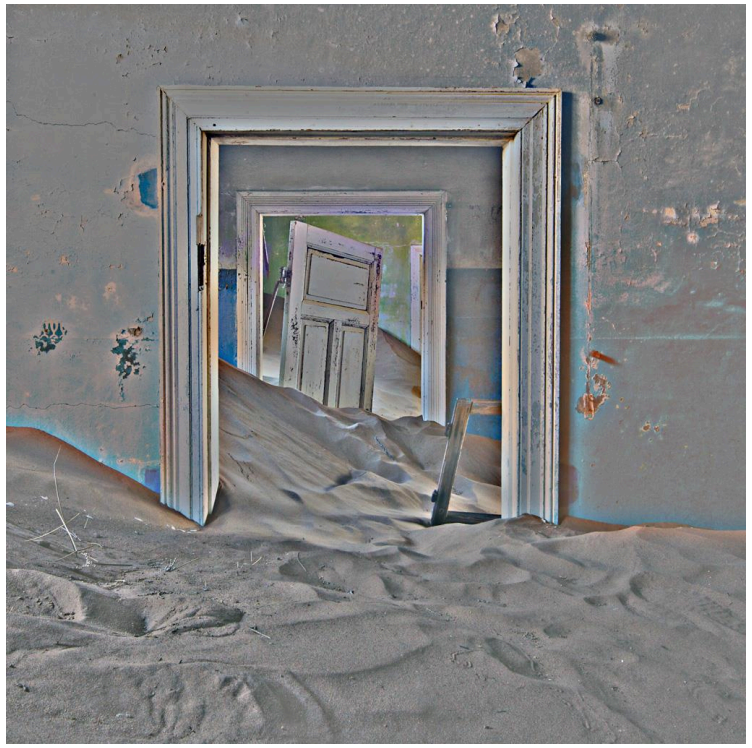
Mip-Mapping – Bilineares Upsampling

Tiefpass – Hochpass

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

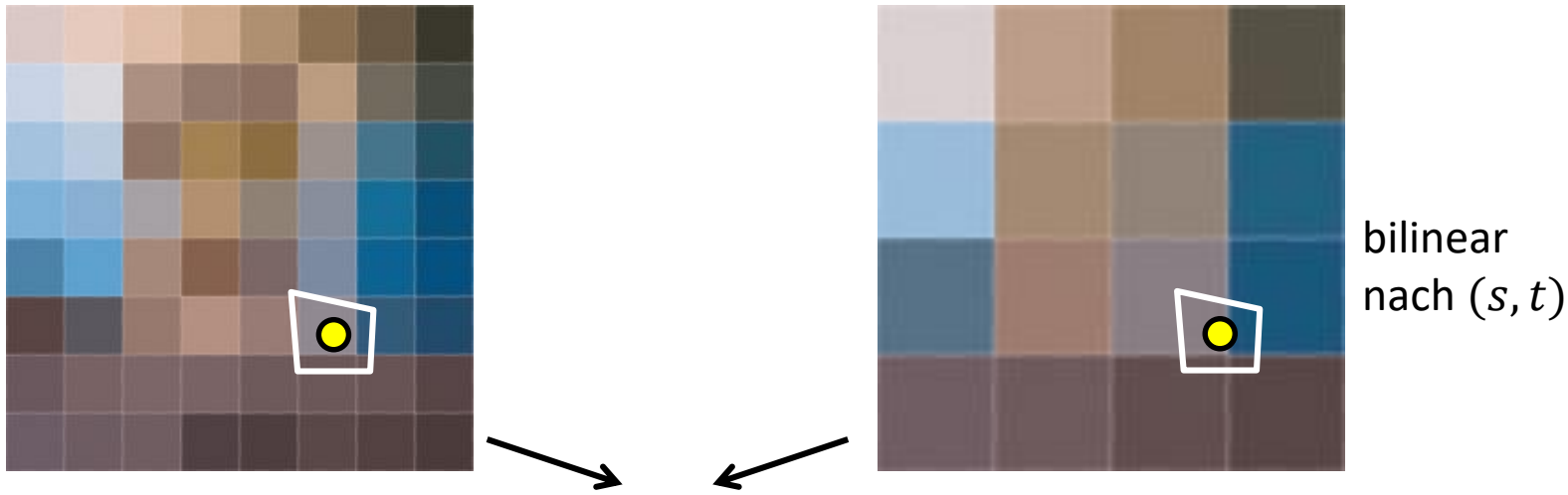
Tiefpass – Hochpass

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



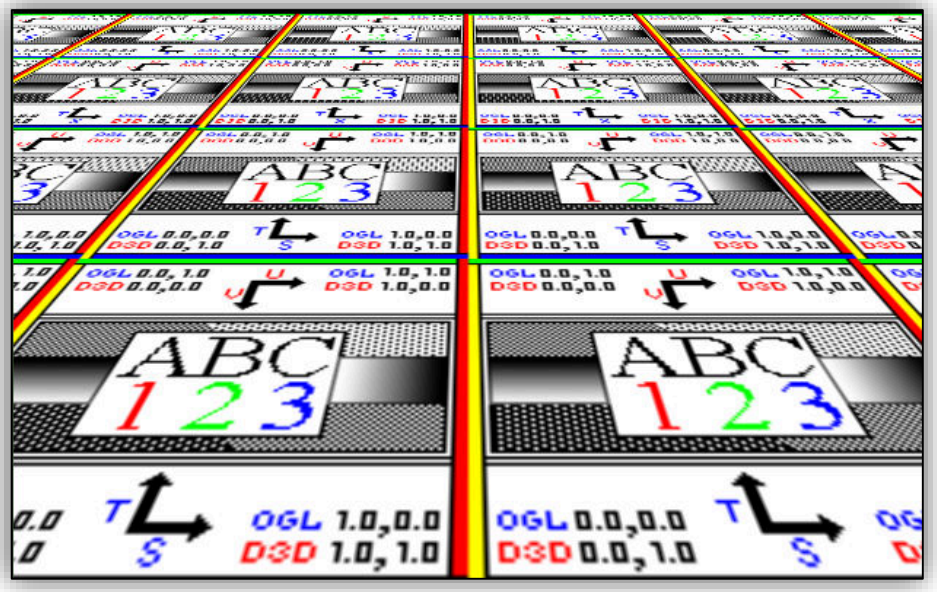
Mip-Mapping

- wähle Texturauflösung (Mip-Map Stufe n) so, dass
  - Texelgröße(n) $\leq$ Größe Pixelfootprint auf Textur < Texelgröße($n+1$) (gebräuchliche Regel für die Wahl der Mip-Map Stufe)
  - $n = \min(n_{max} - 1, \log_2(a \cdot \text{"Footprint}-\text{Größe"}))$, $n = 0$ entspricht höchster Auflösungsstufe
  - der endgültige Farbwert wird durch **trilineare Interpolation** zwischen zwei Mip-Map-Stufen (also 8 Texel, je 4 auf einer Stufe) bestimmt
    - bilinear auf Stufe n, bilinear auf Stufe $n+1$
    - anschließend linear zwischen diesen Farben mit Gewicht $n - \lfloor n \rfloor$

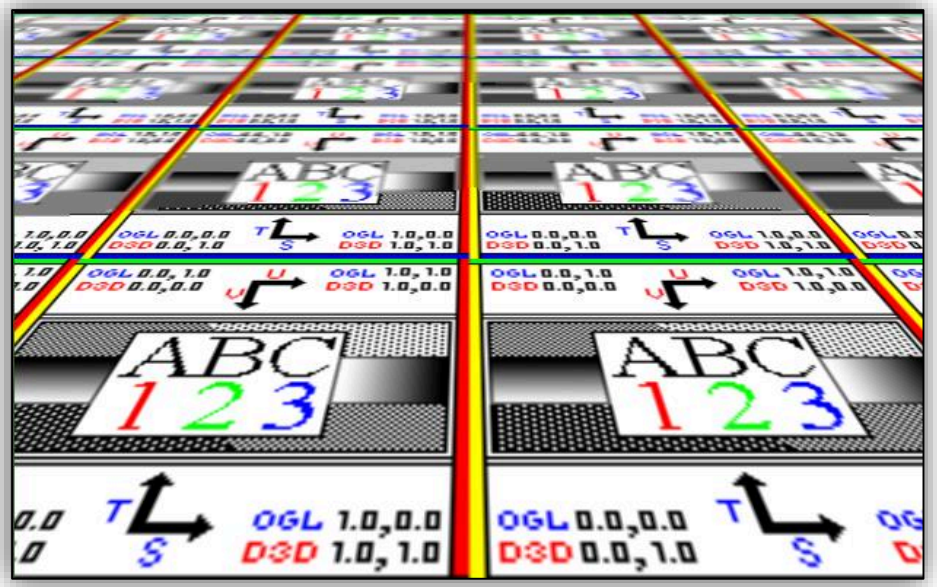


Mip-Mapping *cont.*

ohne Mip-Mapping

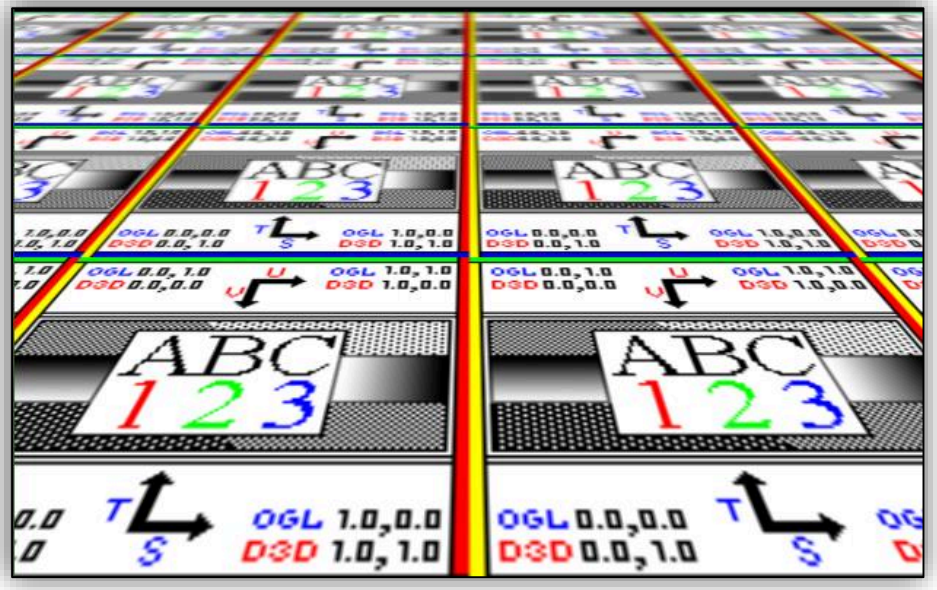


1) Mip-Mapping, nur Mip-Level n



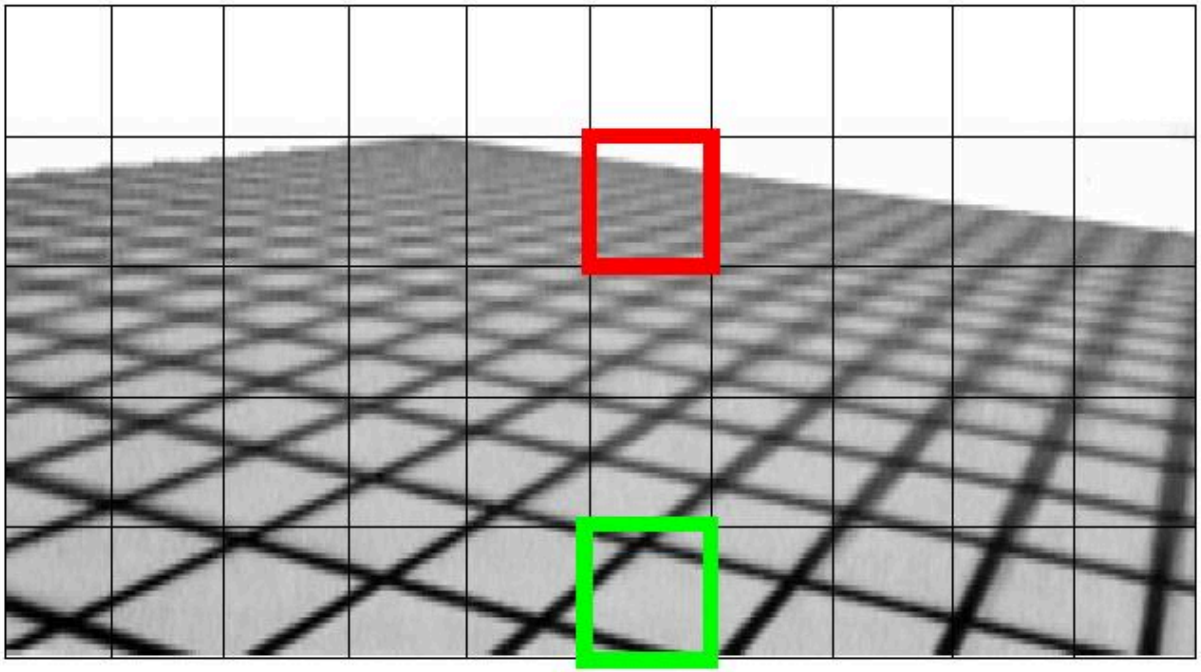
2) Mip-Mapping mit Mip-Level n und $n + 1$

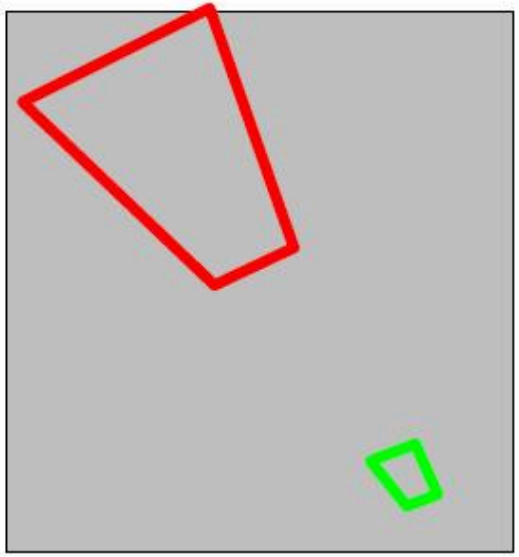
→ trilineare Interpolation



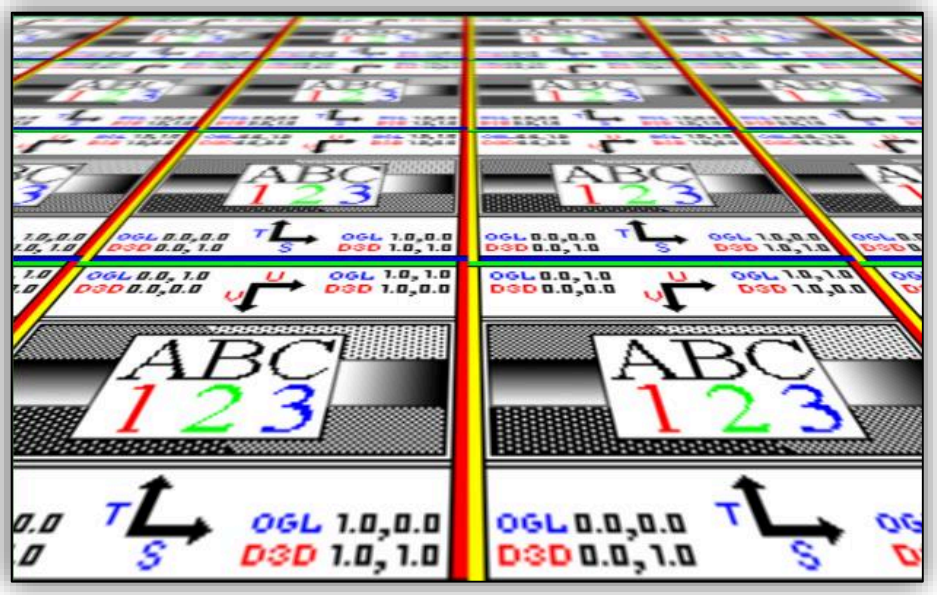
Anisotrope Texturfilterung

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

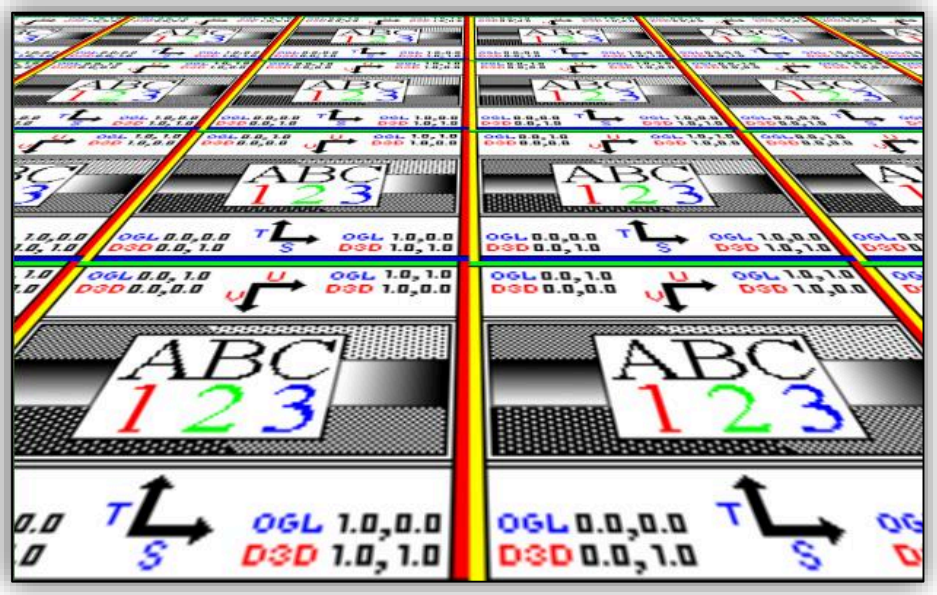




Anisotrope Texturfilterung

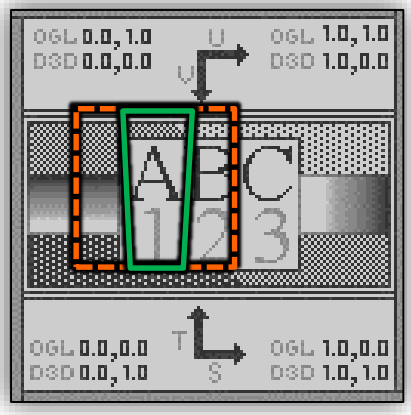


Mip-Mapping



Anisotrope Texturfilterung

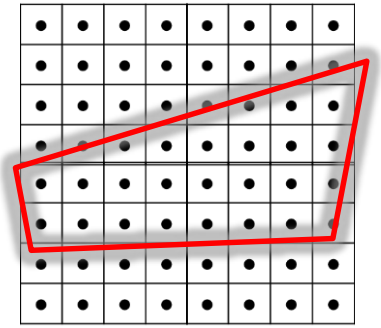
trilineare Interpolation



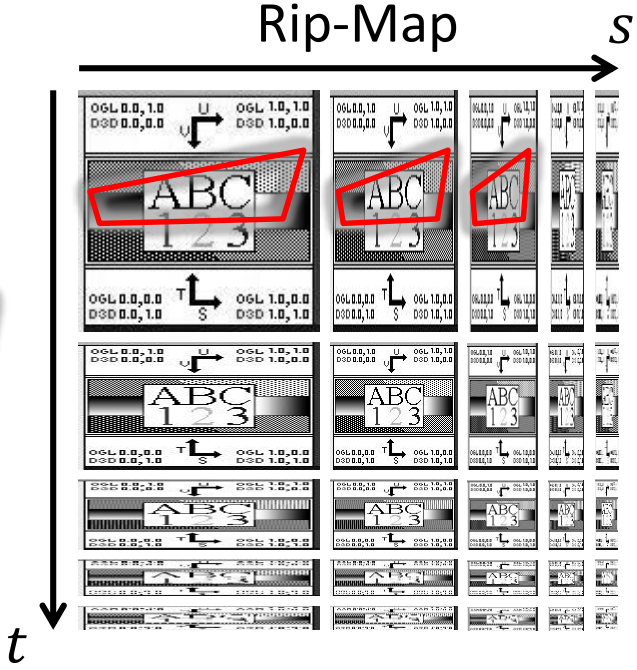
Anisotrope Texturfilterung

RIP-Maps (Rectangular Mip-Maps, kaum in der Praxis)

- enthält Vorfilterungen unabhängig in jeder Achse
- länglicher Abdruck  im Texturraum
  → wähle entsprechend vorgefilterte Textur
- 4-facher Speicherbedarf
- löst das Problem nur teilweise und behandelt nur Anisotropie entlang der Achsen

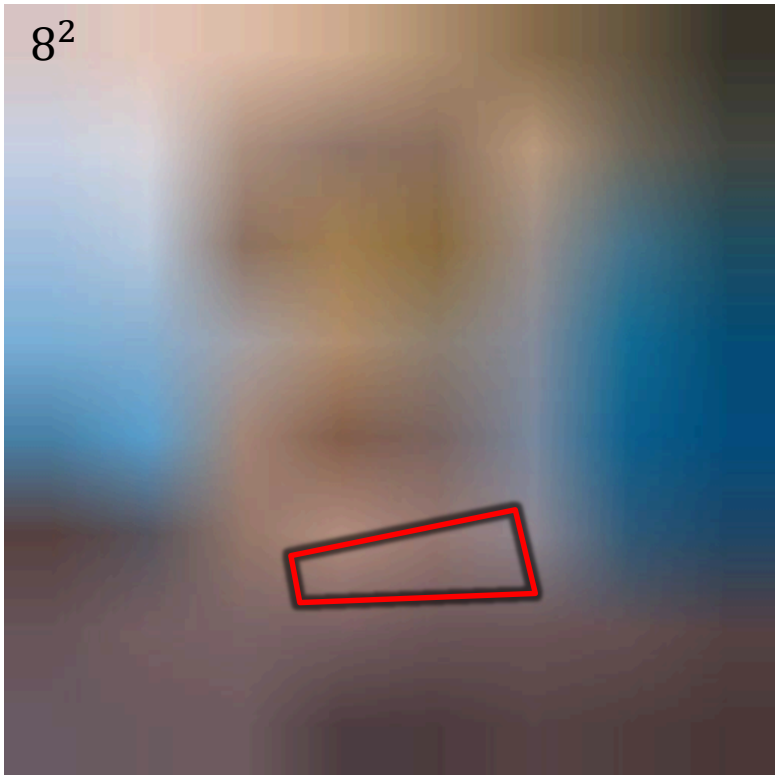


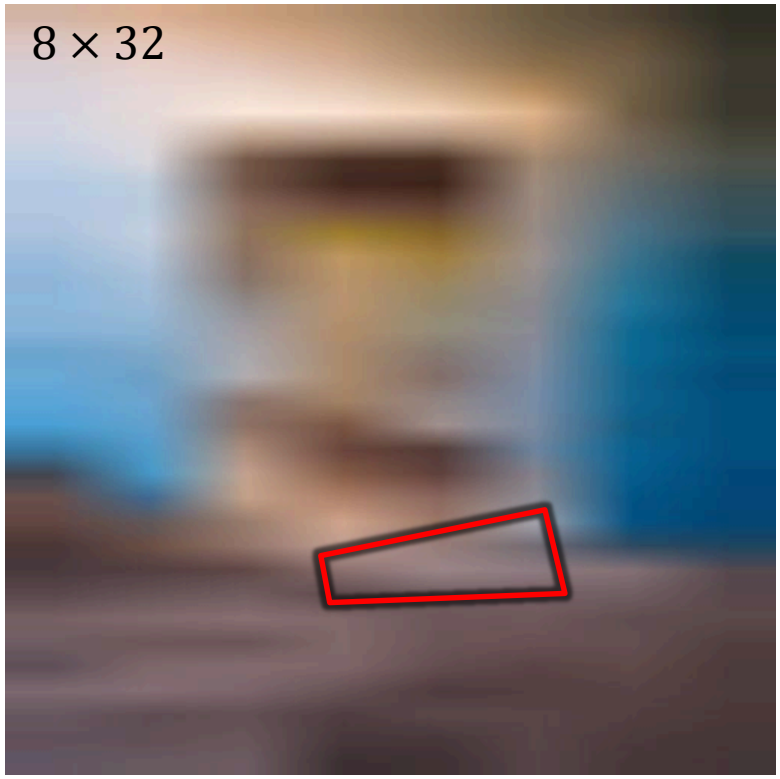
Footprint



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)

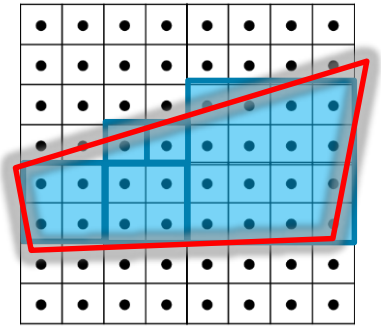




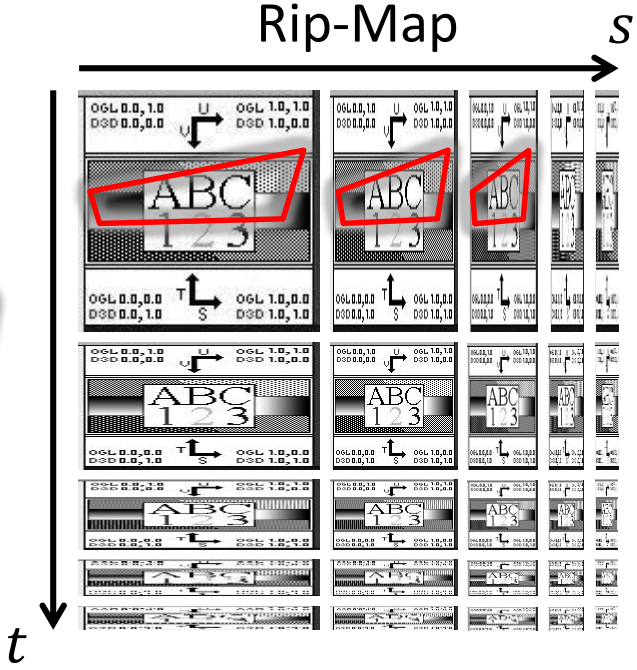
Anisotrope Texturfilterung

RIP-Maps (Rectangular Mip-Maps, kaum in der Praxis)

- enthält Vorfilterungen unabhängig in jeder Achse
- länglicher Abdruck  im Texturraum
  → wähle entsprechend vorgefilterte Textur
- 4-facher Speicherbedarf
- löst das Problem nur teilweise und behandelt nur Anisotropie entlang der Achsen



Footprint



Anisotrope Filterung in der Praxis (und in Grafik-Hardware)

- Abtasten des Bereichs durch eine geschickte Kombination von Abtastungen verschiedener Mip-Map-Stufen, typischerweise 8-16 Abtaststellen
- noch besser: Approximation und Gewichtung des Footprints mit elliptischem Gauß-Filter (sog. EWA-Filter)

Summed Area Tables (SAT) [Crow84]

- interessantes Konzept für Summation über größere Bereiche
- jedes Element s_{mn} einer SAT speichert die Summe aller Elemente darüber und links aus der Eingabetextur/-tabelle
- benötigt daher höhere Genauigkeit als z.B. 8-Bit pro (Farb-)Wert

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

Eingabe t_{ij}

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

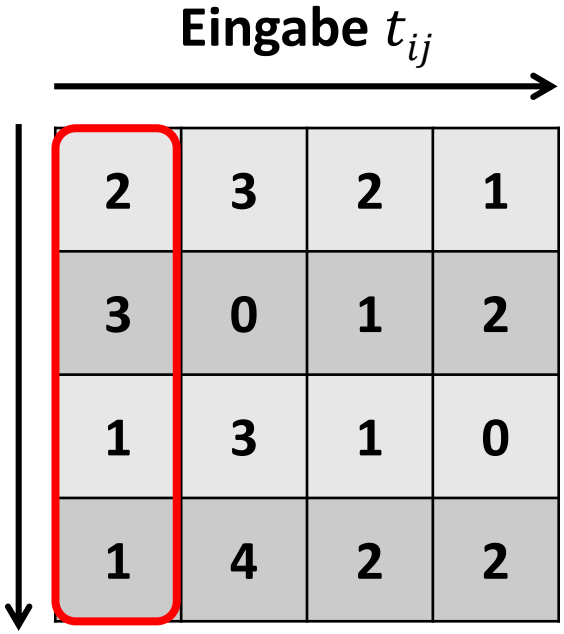
SAT s_{mn}

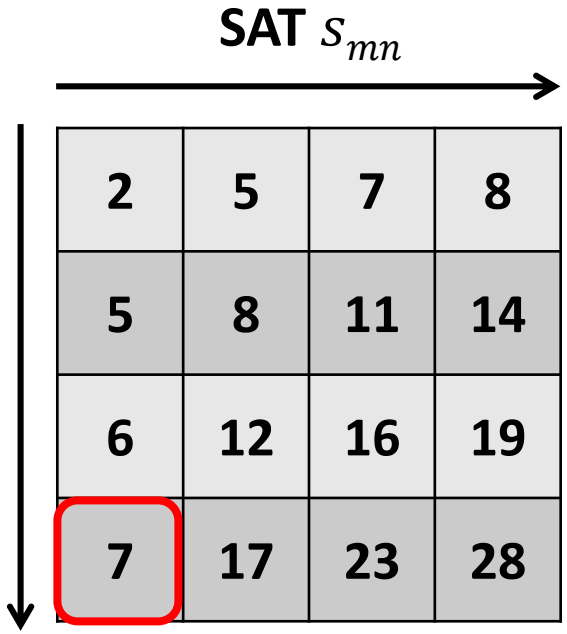
2	5	7	8
5	8	11	14
6	12	16	19
7	17	23	28

Summed Area Tables (SAT) [Crow84]

- interessantes Konzept für Summation über größere Bereiche
- jedes Element s_{mn} einer SAT speichert die Summe aller Elemente darüber und links aus der Eingabetextur/-tabelle
- benötigt daher höhere Genauigkeit als z.B. 8-Bit pro (Farb-)Wert

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

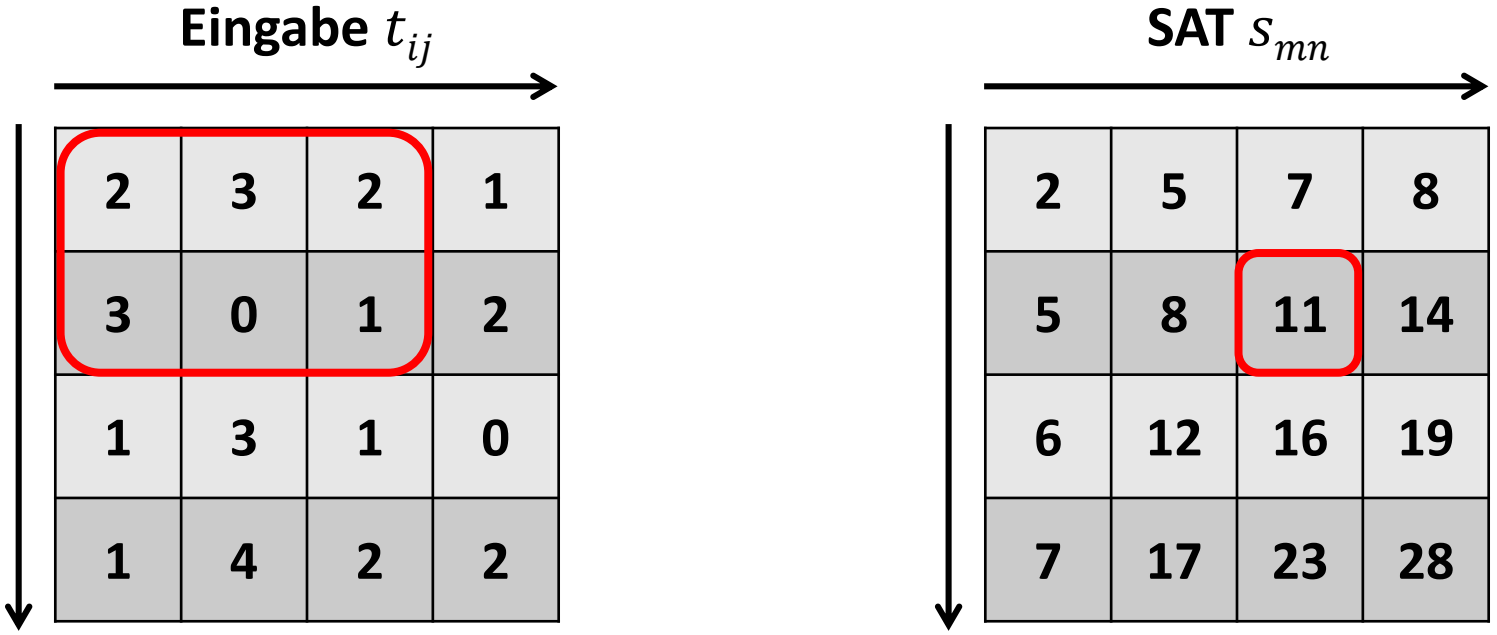




Summed Area Tables (SAT) [Crow84]

- interessantes Konzept für Summation über größere Bereiche
- jedes Element s_{mn} einer SAT speichert die Summe aller Elemente darüber und links aus der Eingabetextur/-tabelle
- benötigt daher höhere Genauigkeit als z.B. 8-Bit pro (Farb-)Wert

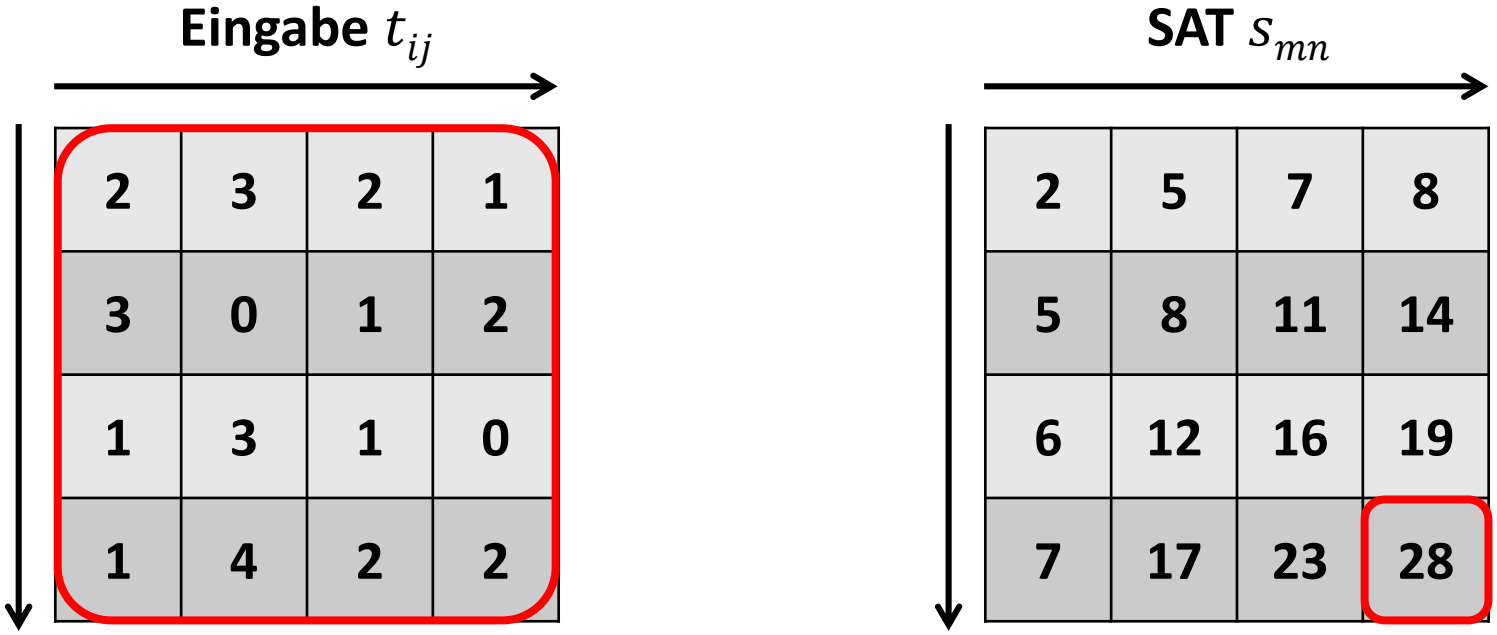
$$S_{mn} = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij}$$



Summed Area Tables (SAT) [Crow84]

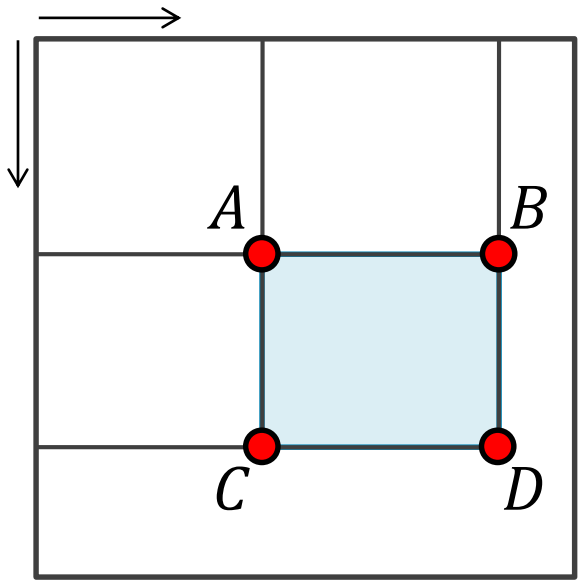
- interessantes Konzept für Summation über größere Bereiche
- jedes Element s_{mn} einer SAT speichert die Summe aller Elemente darüber und links aus der Eingabetextur/-tabelle
- benötigt daher höhere Genauigkeit als z.B. 8-Bit pro (Farb-)Wert

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



Summed Area Tables (SAT) [Crow84]

- Anwendung: Filterung der Eingabetextur mit einem beliebigen rechteckigen, nach den Achsen ausgerichteten Box-Filter **in konstanter Zeit** (Berechnung der SAT in logarithmischer Zeit möglich)



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

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

- Approximation eines Pixel-Footprints

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

2+3+3+5+1+6=20

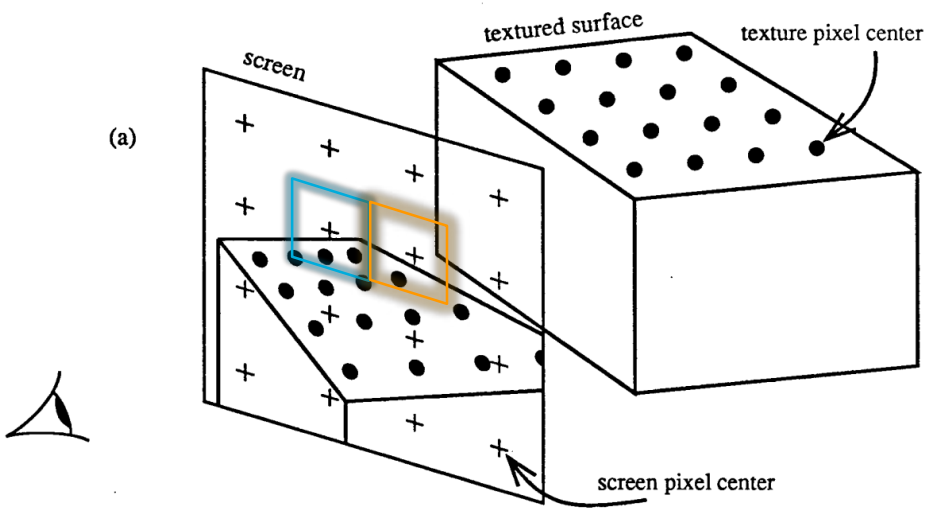
1	2	4	6
2	5	10	16
4	10	20	32
6	13	29	49

29-6-4+1=20

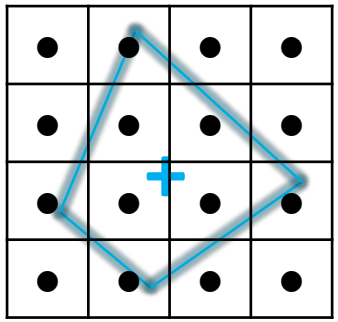
Bestimmung des Footprints

Einfachste Möglichkeit den Footprint zu bestimmen

- schicke einen Primärstrahl durch die Ecken eines Pixels oder betrachte einen 2×2 Pixelblock und bestimme die 4 Texturkoordinaten (was tun an einer Objektkante? gleich!)
- Differenz der Texturkoordinaten liefert Größe/Form des Footprints
  - Mip-Mapping: bestimme $\max(\text{Breite}, \text{Höhe})$ und daraus n
  - Rip-Mapping, SAT: betrachte Breite und Höhe separat
- Textur-Lookup an berechneter Stelle mit entsprechender Filterung



Abdruck („Footprint“) eines Pixels im Texturraum

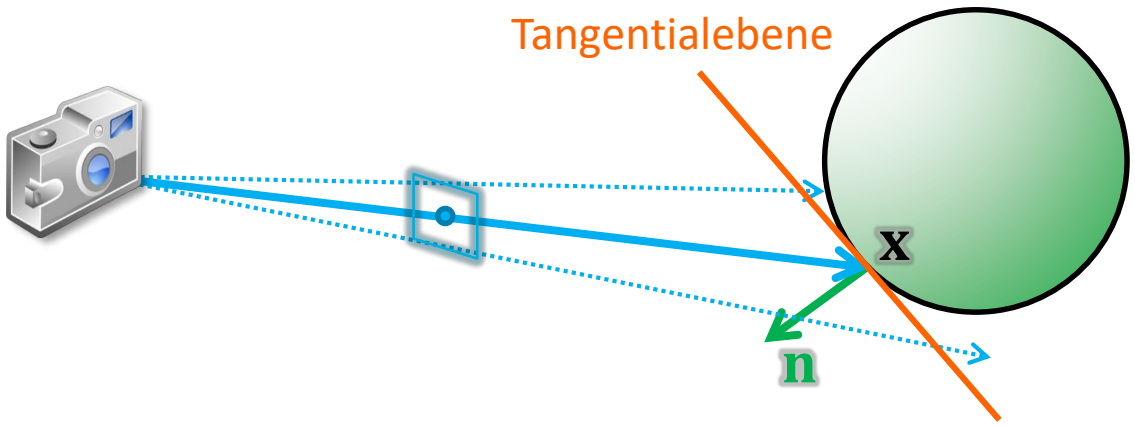


Textur

Bestimmung des Footprints

Grundidee der sogenannten *Ray Differentials*

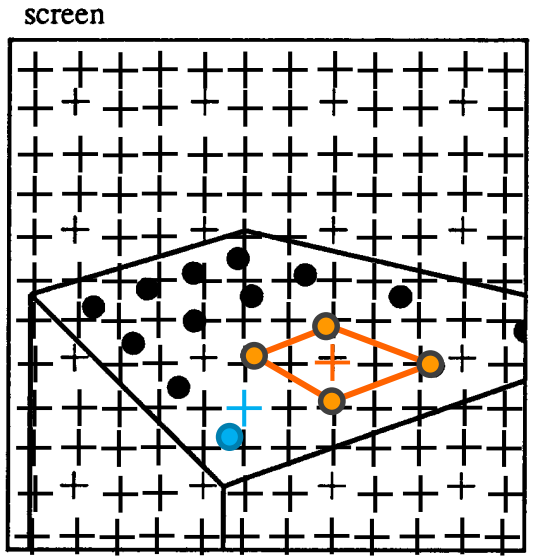
- Schnitt eines Primärstrahls mit einem Objekt liefert
  - Position $\mathbf{x}$ und Normale $\mathbf{n}$ → definieren Tangentialebene
  - i.d.R. affine Abbildung zwischen Texturkoordinate und Position
  - betrachte die Veränderung der Texturkoordinaten (Ableitung nach den Bildschirmachsen, z.B. wieder über Differenzen)
  - bestimme daraus Größe/Form des Footprints
  - lässt sich verallgemeinern und fortsetzen für Spiegelungen, Transmission etc.



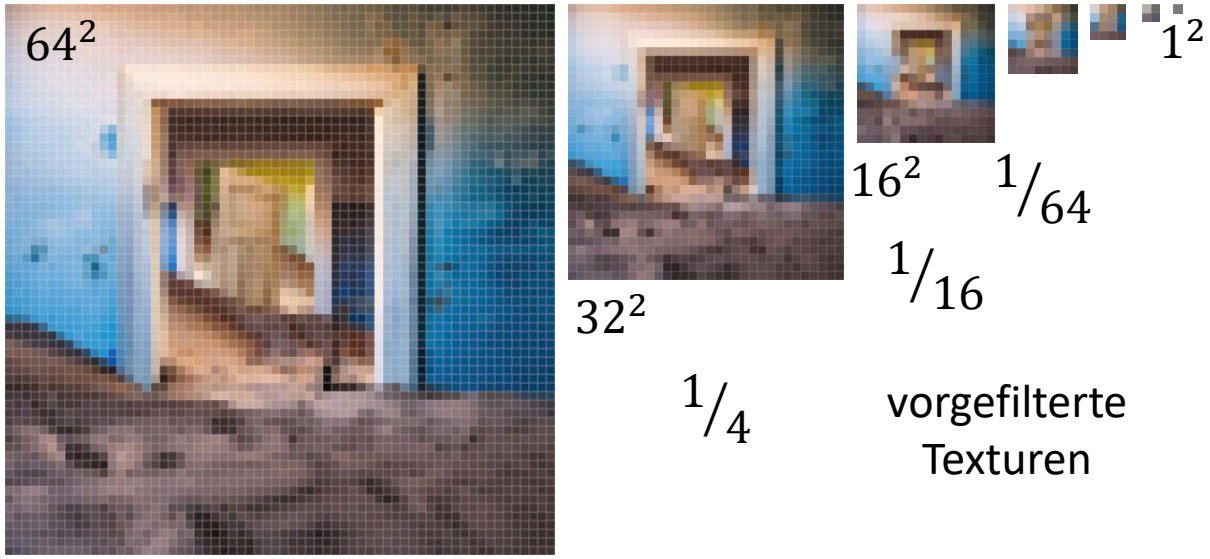
Zusammenfassung Textur-Filterung

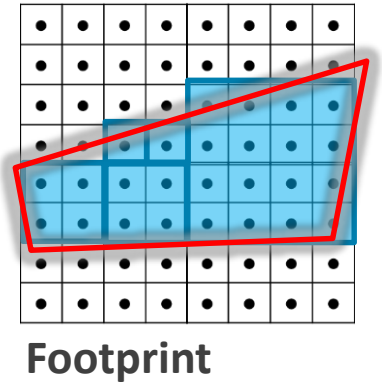
Vergrößerung (Magnification)

- bilineare Interpolation:
  Interpolation der 4 nächsten Texel



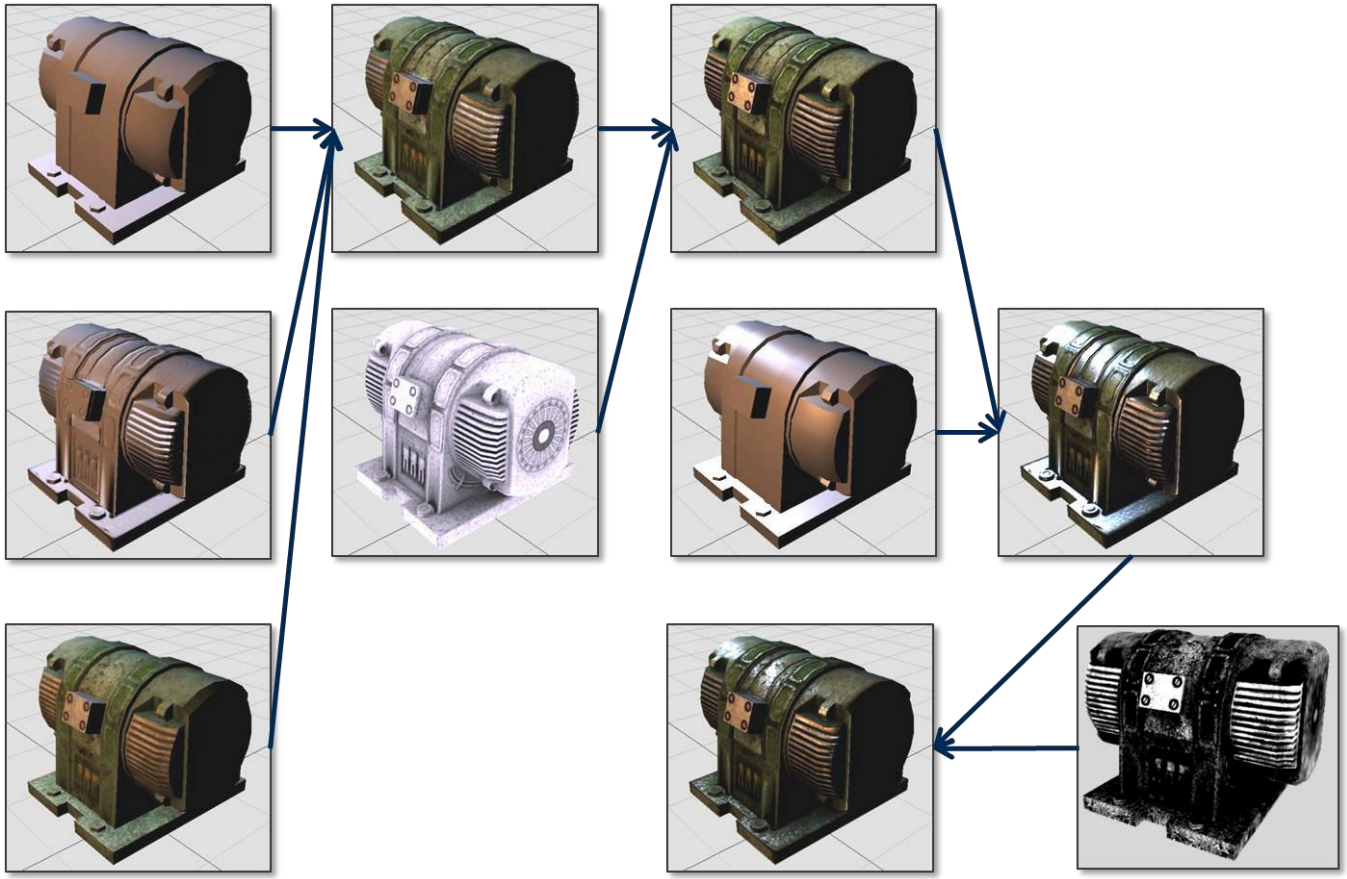
- **Verkleinerung (Minification)**
  - Vorfilterung statt Überabtasten
  - Mip-Mapping + trilineare Interpolation
  - anisotrope Texturfilterung





Multitexturing Beispiel

- „komplexe“ optische Effekte durch Kombinationen mehrerer Texturen
- Texturen enthalten auch andere Informationen, nicht nur Farbe
- mehrere Texturen → ggf. mehrere Texturkoordinaten pro Vertex

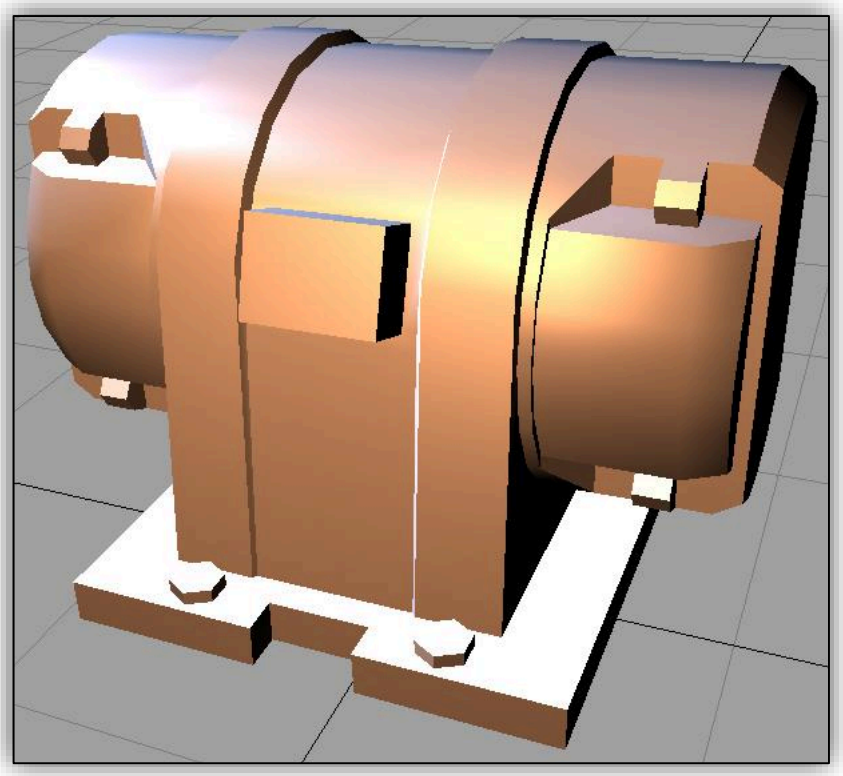


Texturierungstechniken

Diffuse Textur

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

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

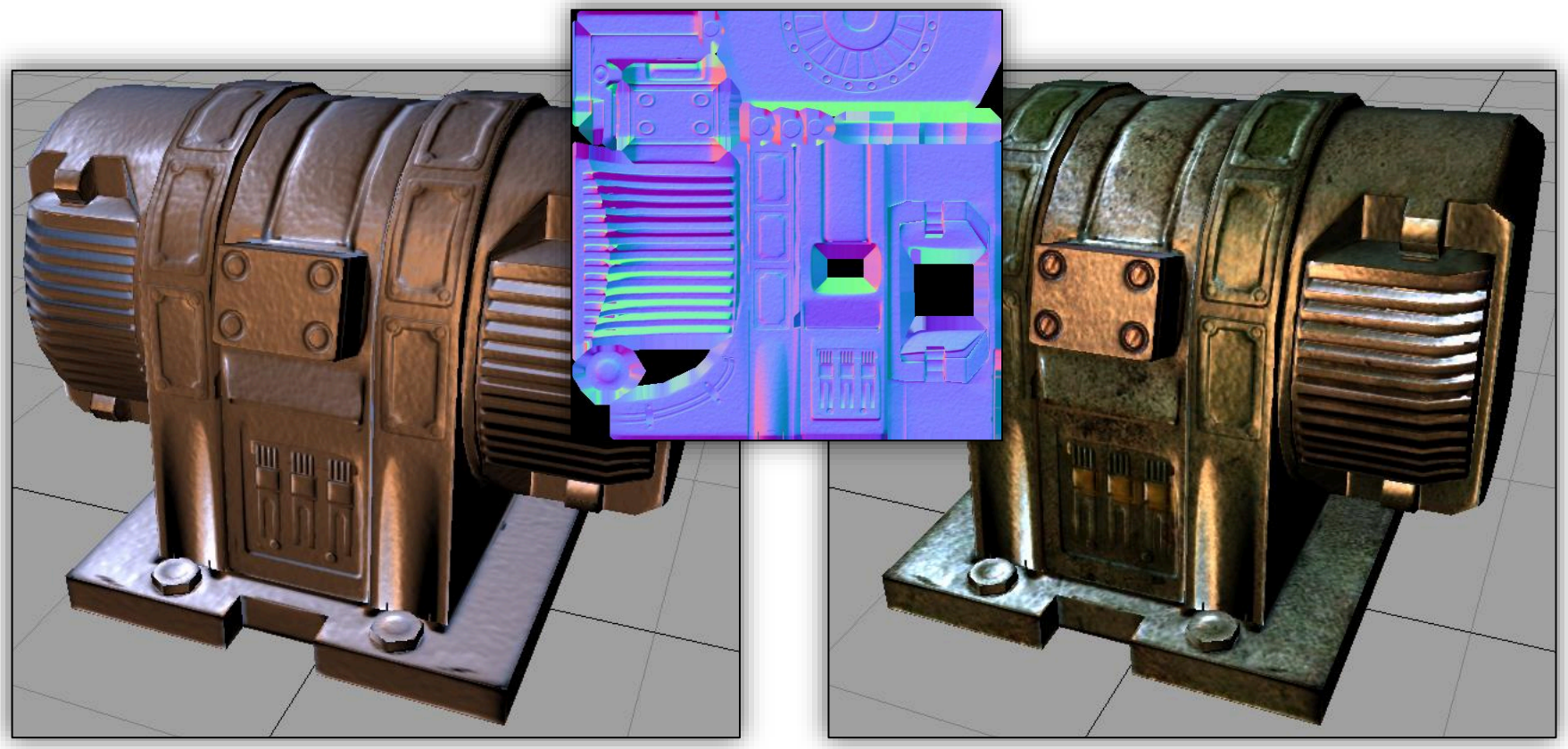


Texturierungstechniken

Bump oder Normal Mapping

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

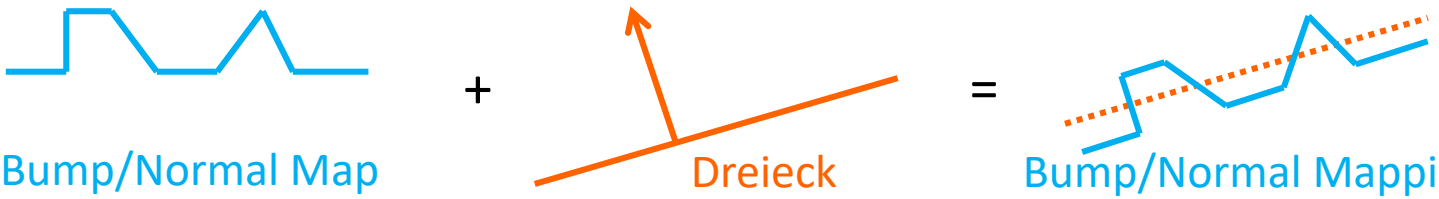
$$I = k_a \cdot I_{\mathrm{L}} + k_d \cdot I_{\mathrm{L}} \cdot (\mathbf{N} \cdot \mathbf{L}) + k_s \cdot I_{\mathrm{L}} \cdot (\mathbf{R} \cdot \mathbf{V})^n$$



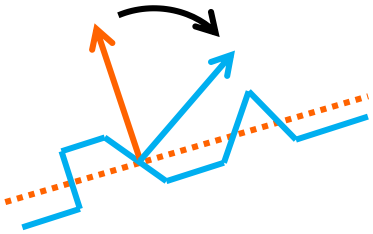
Texturierungstechniken

Bump oder Normal Mapping

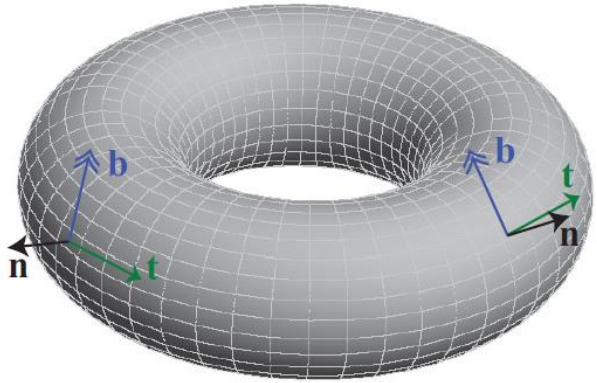
- Variation der Normale einer Oberfläche berechnet aus der Veränderung einer Basisfläche durch eine Bump Map oder Normal Map
- Fläche bleibt aber geometrisch flach und nur die Normalen variieren
- Berechnung meist im sog. Tangentenraum (siehe KoSys auf Torus)



Bump/Normal Map | Dreieck | Bump/Normal Mapping



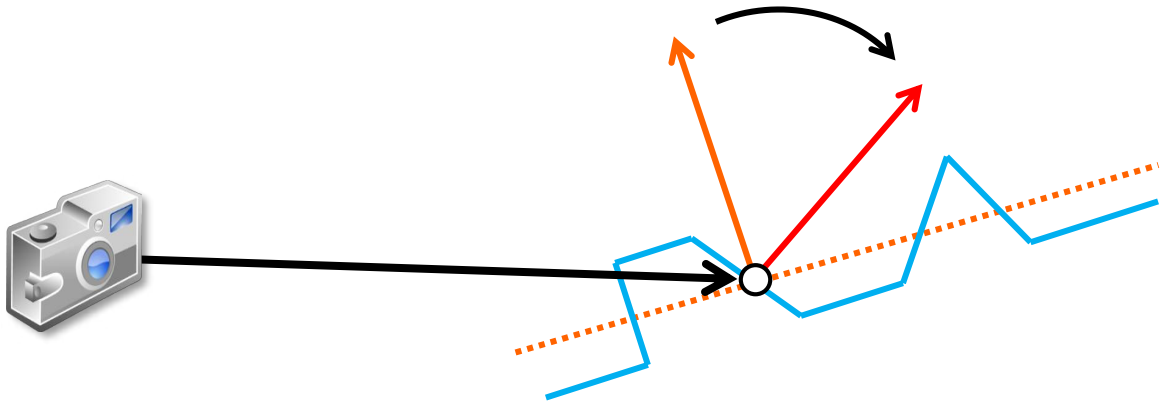
veränderte Normale
für die Beleuchtungsberechnung



Texturierungstechniken

Bump oder Normal Mapping

- … erzeugen eine Inkonsistenz bei der Schattierungsberechnung, die bei physikalisch-basierter Bildsynthese zu Problemen führt: den Schnittpunkt mit einer Fläche mit dieser Normale hätte es nie geben dürfen!
  - daher unterscheidet man im Rendering auch Shading-Normalen und geometrische Normalen



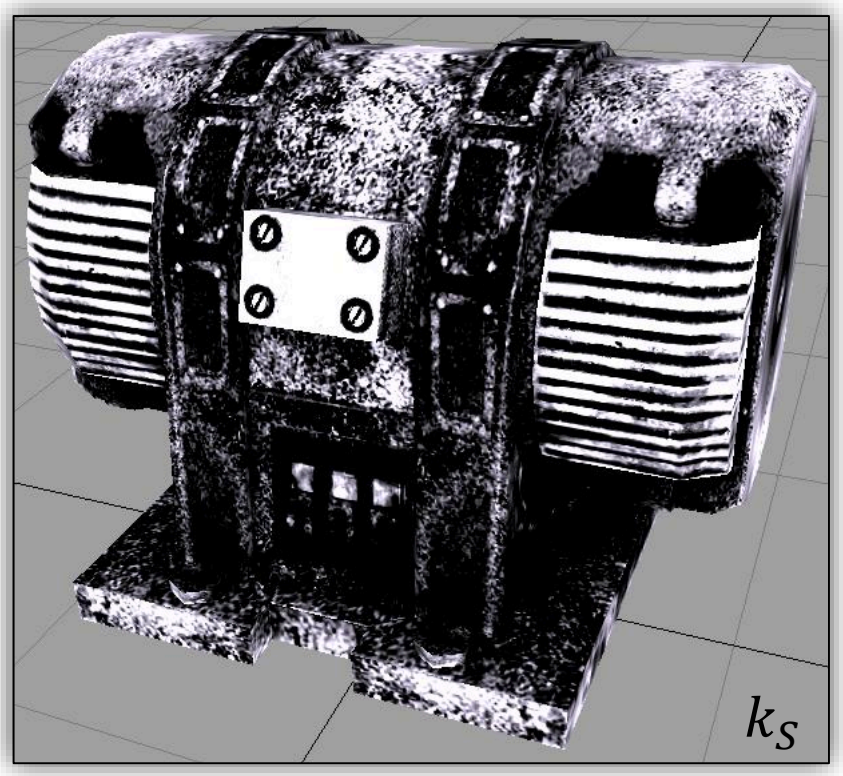
veränderte, eigentlich unzulässige Normale
für die Beleuchtungsberechnung

Texturierungstechniken

Gloss-Map / Gloss-Textur

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

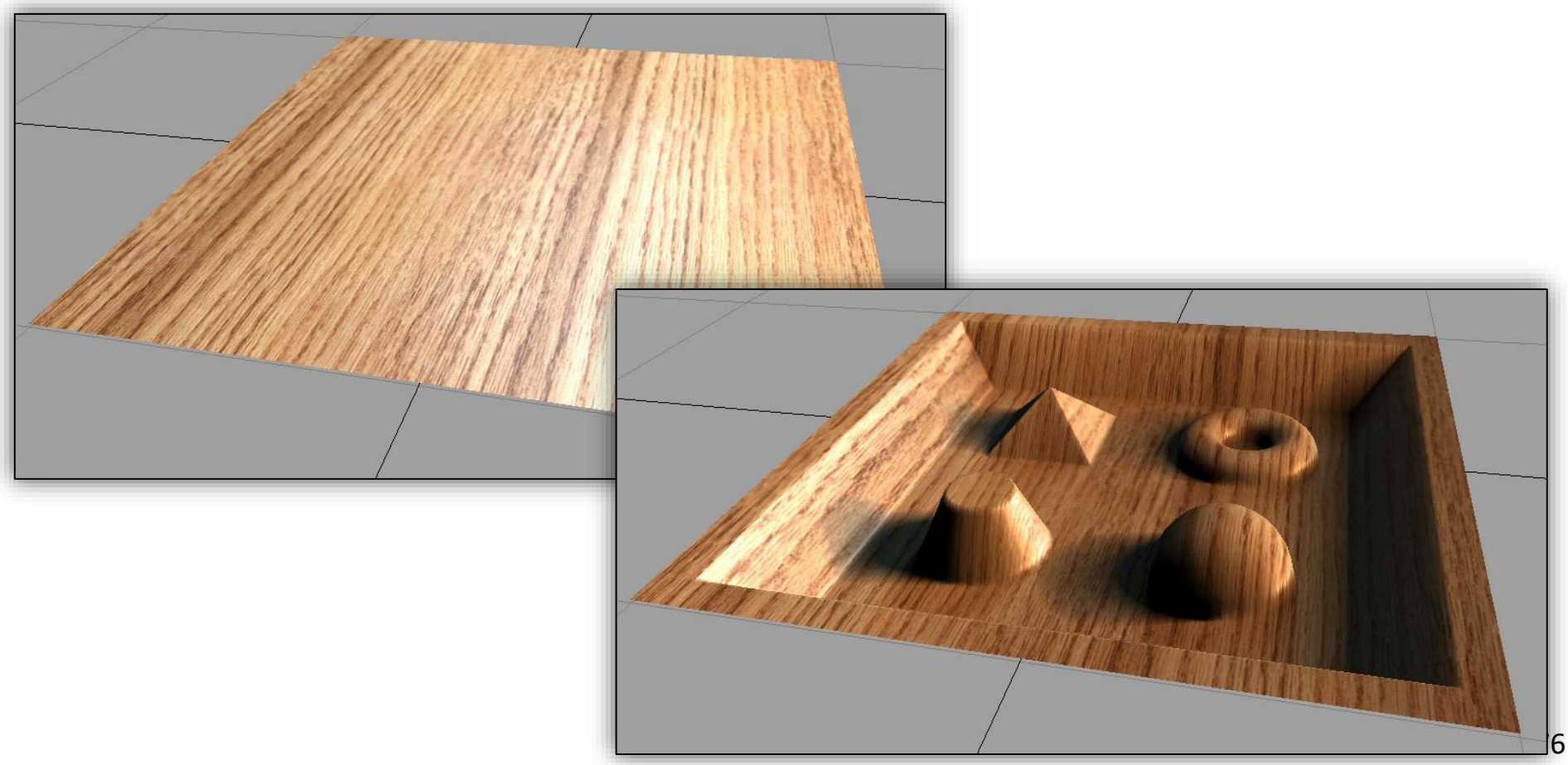
$$I = k_a \cdot I_{\mathrm{L}} + k_d \cdot I_{\mathrm{L}} \cdot (\mathbf{N} \cdot \mathbf{L}) + k_s \cdot I_{\mathrm{L}} \cdot (\mathbf{R} \cdot \mathbf{V})^n$$



Texturierungstechniken

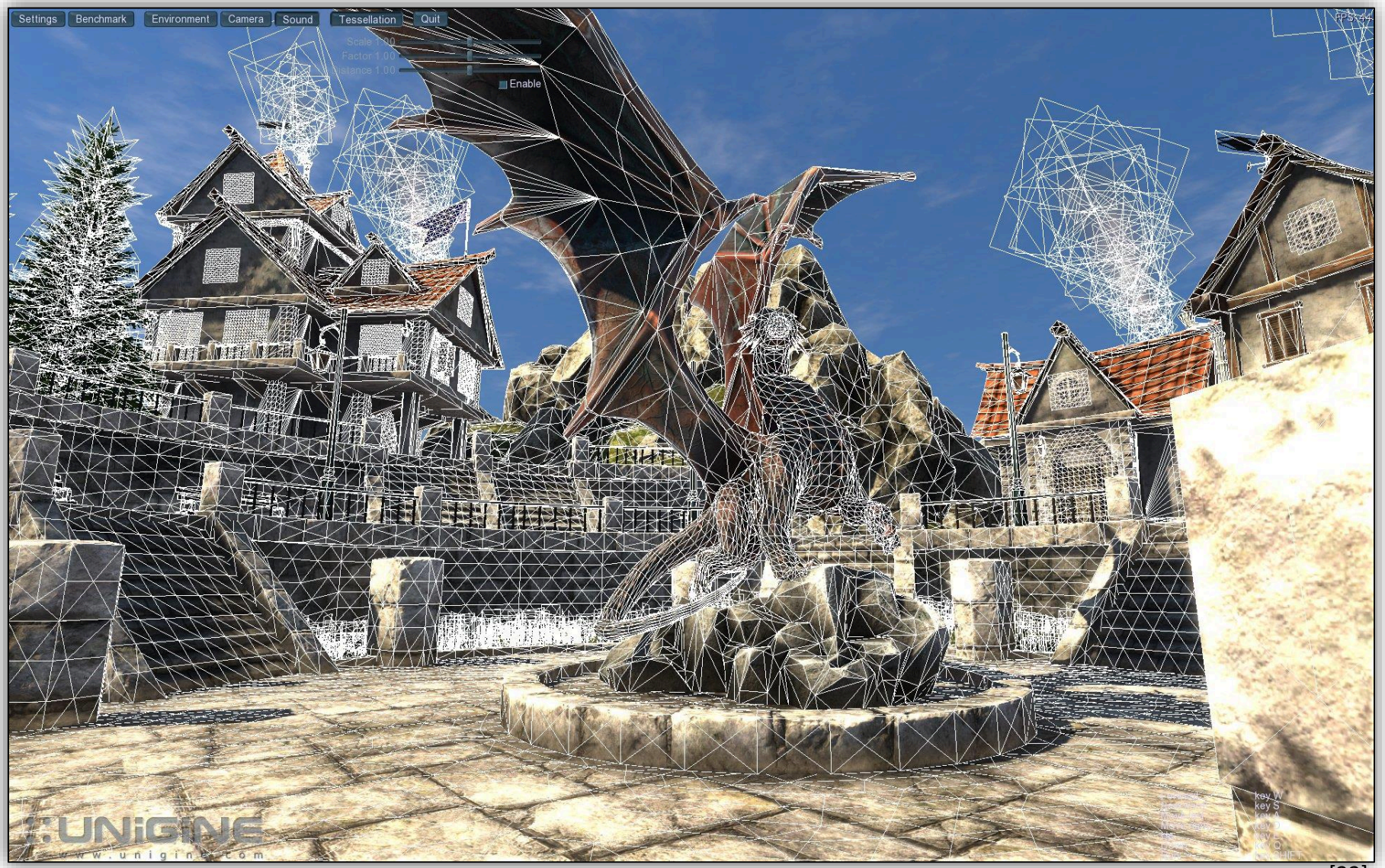
Displacement Mapping

- Verschiebung der Oberfläche und Änderung der Normale
- mehr als nur Änderung der Beleuchtungsberechnung
- z.B. durch Geometrie-Tessellierung und Verschiebung (GPU-unterstützt)



Beispiel Tessellation – Unigine Benchmark

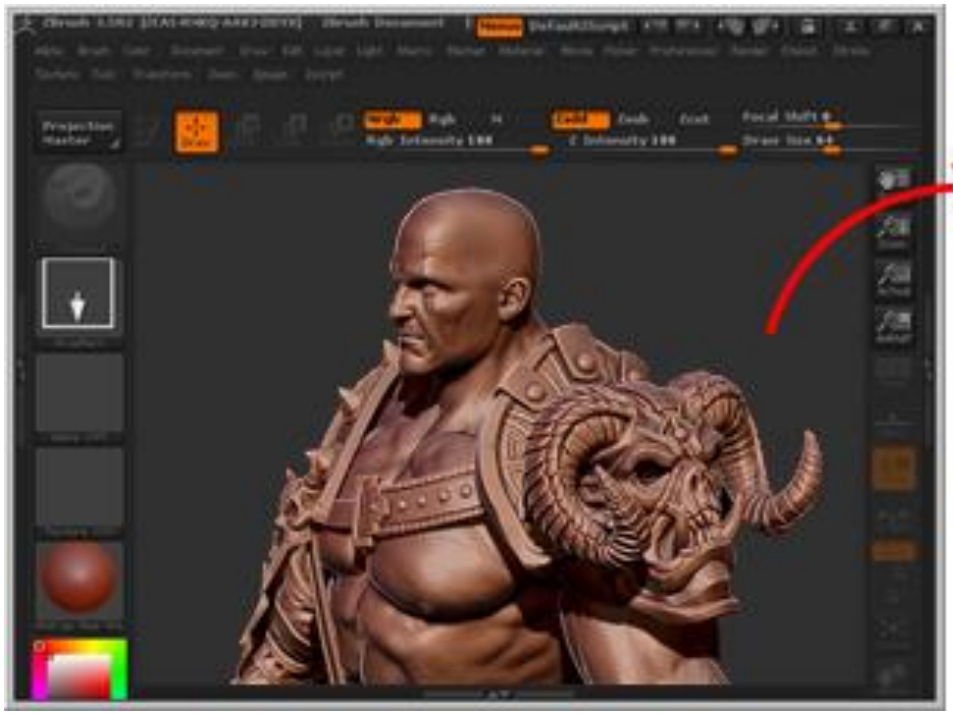
Eingabegeometrie vor der Unterteilung durch die Tessellation-Einheit



Beispiel Tessellation – Unigine Benchmark

Displacement Mapping: Unterteilung und Verschiebung der neuen Vertizes

Vector Displacement Maps (Sculpting in Zbrush)

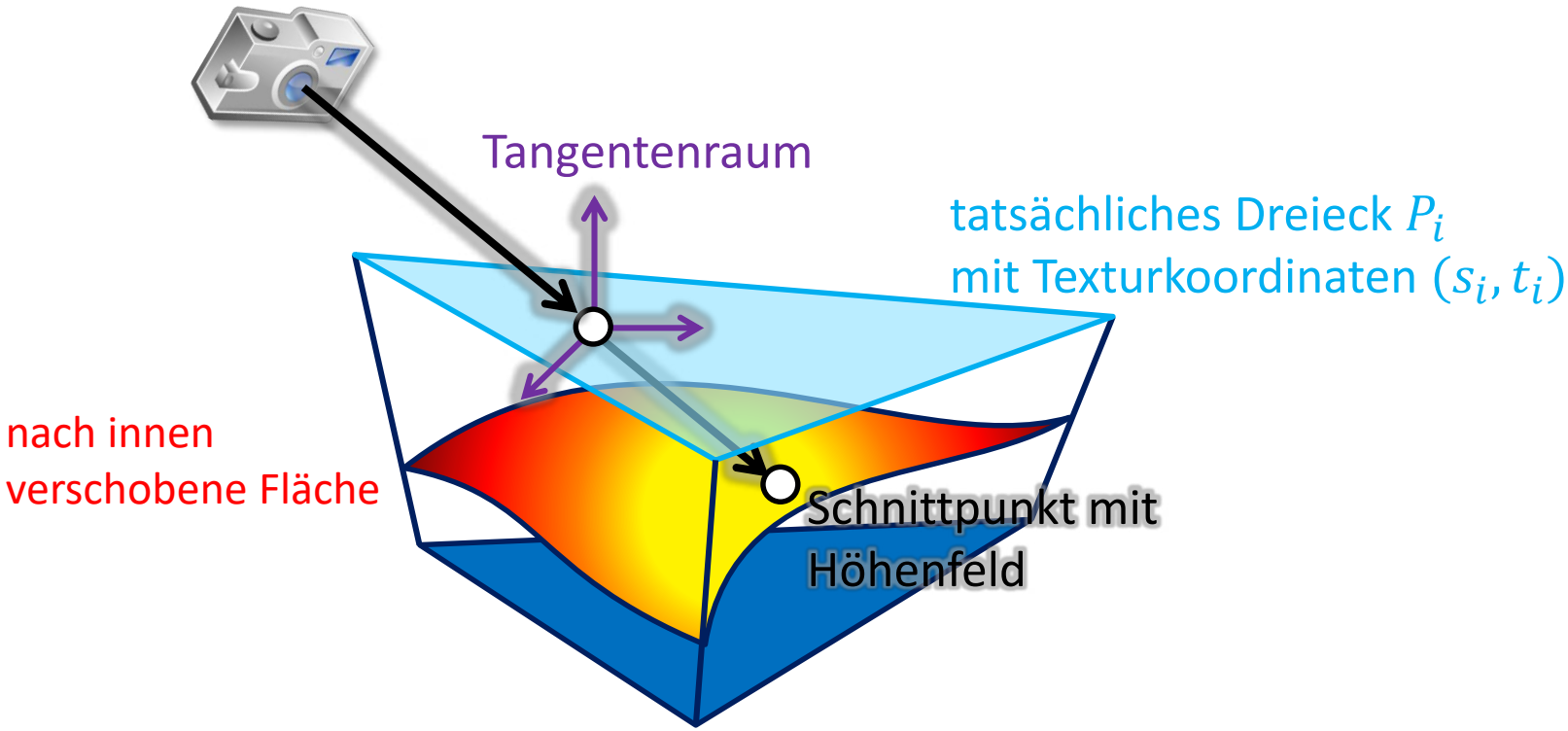


Bilder: http://manual.reallusion.com/iclone_6/enu/pro_6.0/15_Tessellation_Displacement/Using_Vector_Displacement_Maps.htm [23]

Texturierungstechniken

Inverse Displacement Mapping

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

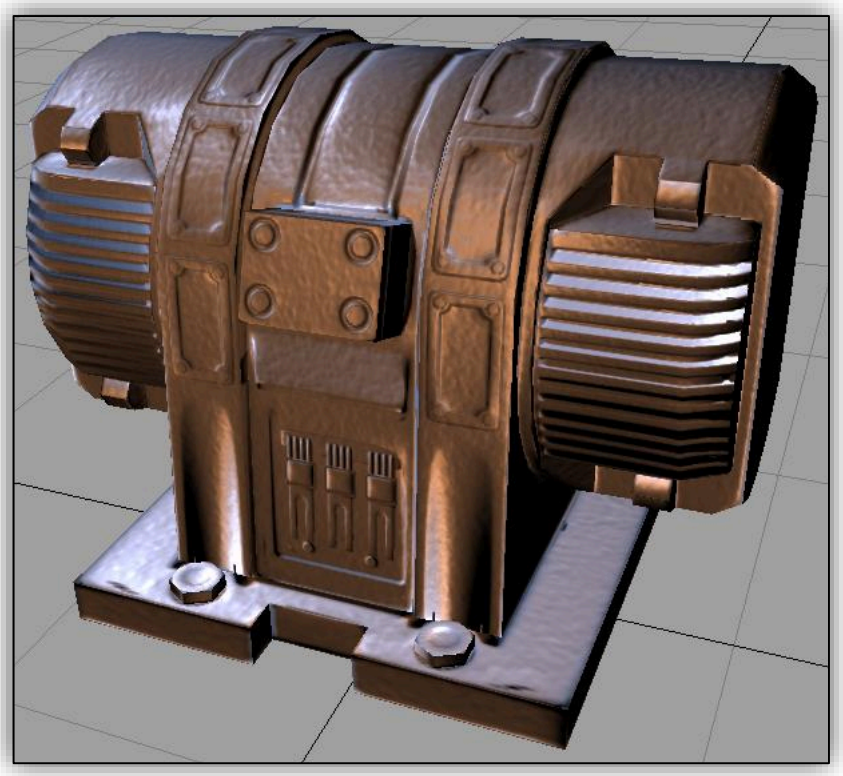


Texturierungstechniken

Ambient Occlusion

- Kontrolle der ambienten Beleuchtung (Umgebungslicht)
- k_a aus Textur, meistens wird auch der diffuse Term modifiziert

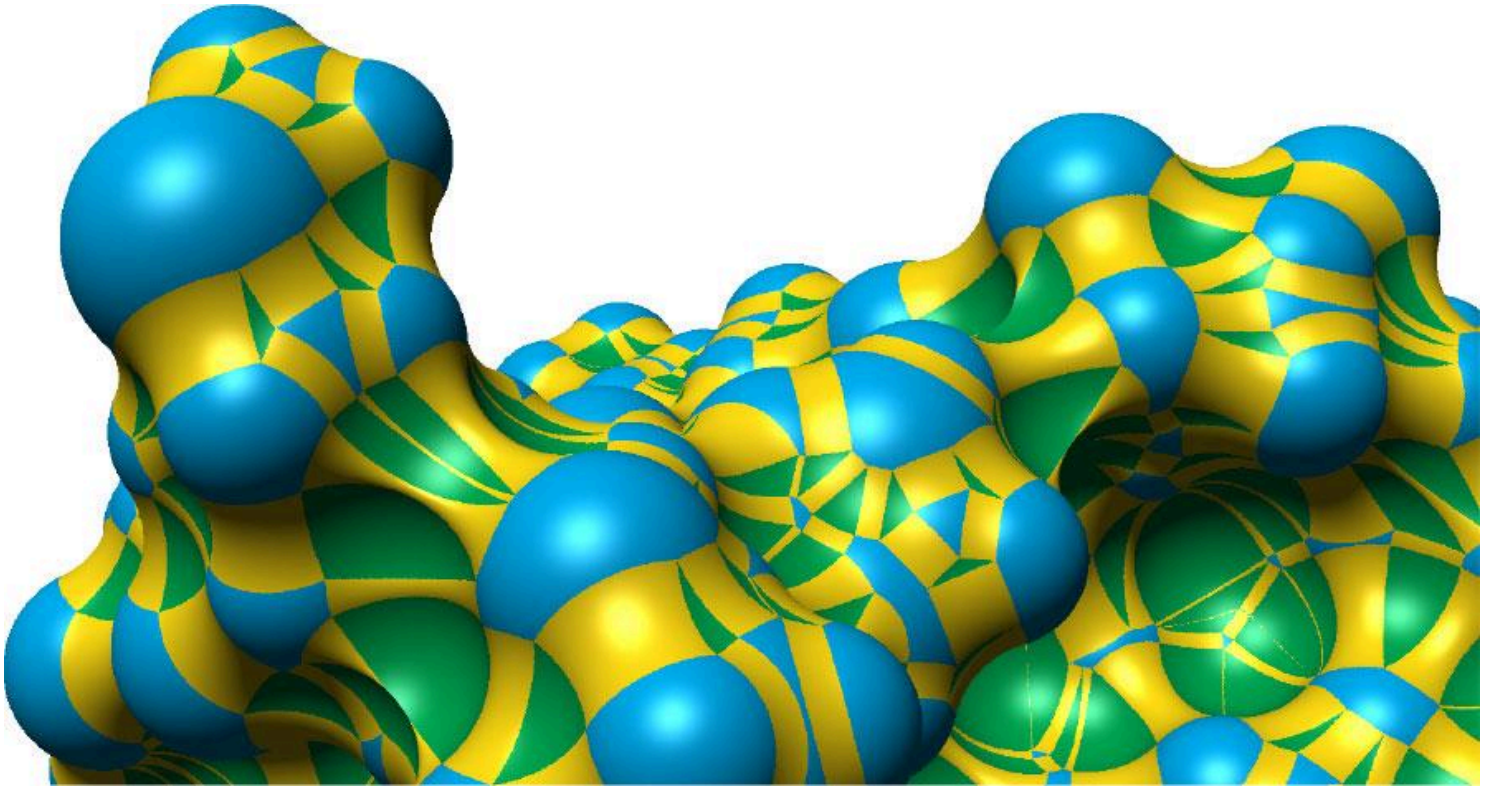
$$I = k_a \cdot I_{\mathrm{L}} + k_d \cdot I_{\mathrm{L}} \cdot (\mathbf{N} \cdot \mathbf{L}) + k_s \cdot I_{\mathrm{L}} \cdot (\mathbf{R} \cdot \mathbf{V})^n$$



Texturierungstechniken

Ambient Occlusion

- Beispiel aus der Visualisierung: Solvent Excluded Surface
  Bild: Interactive GPU-based generation of solvent-excluded surfaces, Hermosilla et al., The Visual Computer, 2017
- Anmerkung: Ambient Occlusion muss nicht in einer Textur gespeichert werden, häufig auch zur Laufzeit berechnet/approximiert



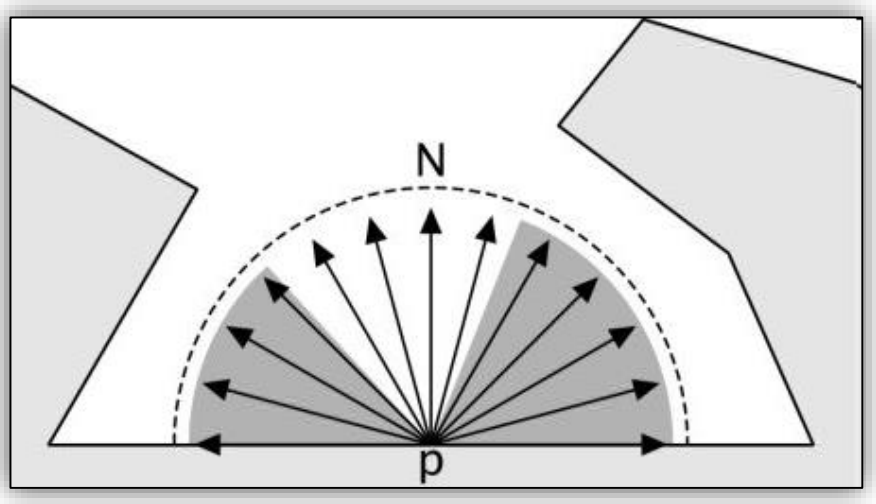
Texturierungstechniken

Ambient Occlusion

- Kontrolle der ambienten Beleuchtung (Umgebungslicht)
- k_a aus Textur, meistens wird auch der diffuse Term modifiziert

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

- Vorberechnung: wie viel Licht aus der Umgebung trifft auf einen Oberflächenpunkt?
- … mittels Raycasting: welcher Teil der Strahlen trifft keine Geometrie in der Nähe
- bei Vorberechnung von AO: gespeichert pro Vertex (wenn ausreichend fein tesselliert) oder …



[24]

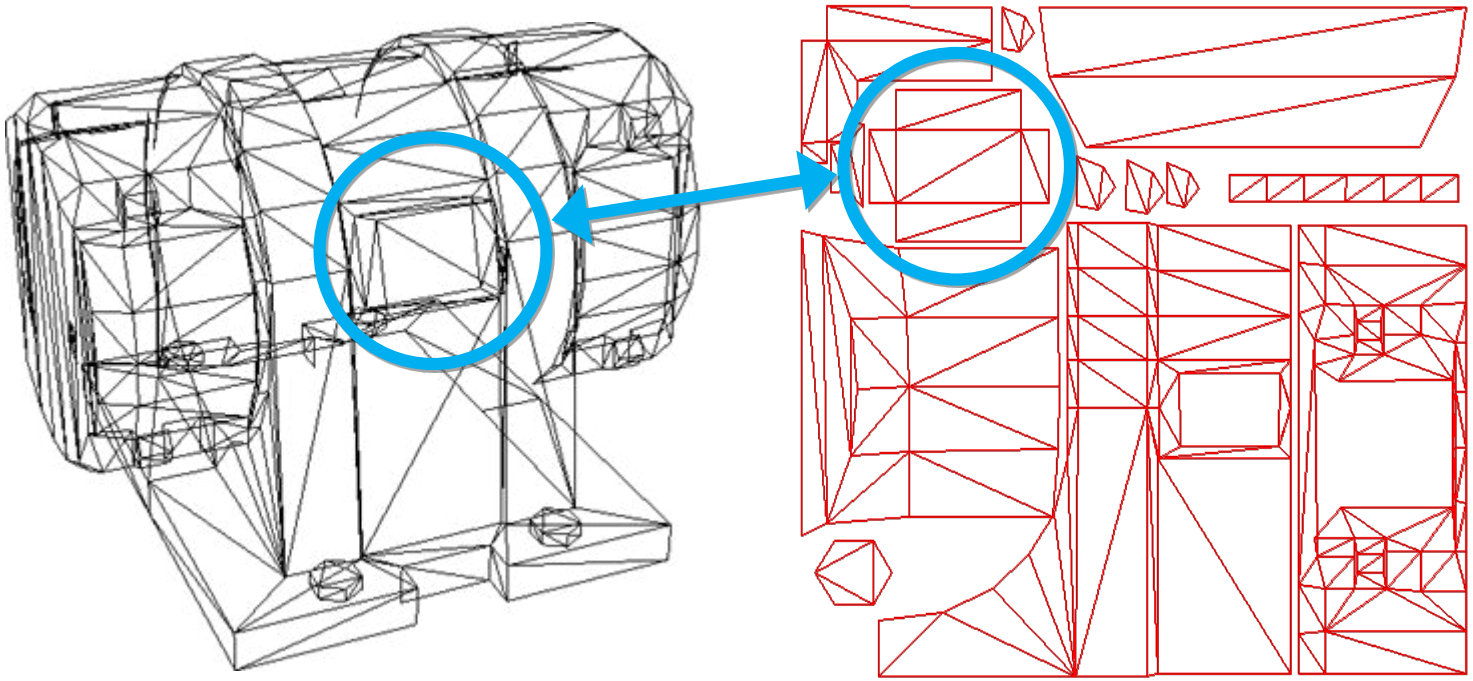
Textur-Atlas

- 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)

  z.B. https://github.com/microsoft/UVAtlas

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, z.B. der Generator unten)
- 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, z.B. der Generator unten)
- Erstellung aufwändig per Hand oder automatisch (Zerschneiden und Ausrollen des Netzes, Verzerrung vs. Anzahl der Schnitte)

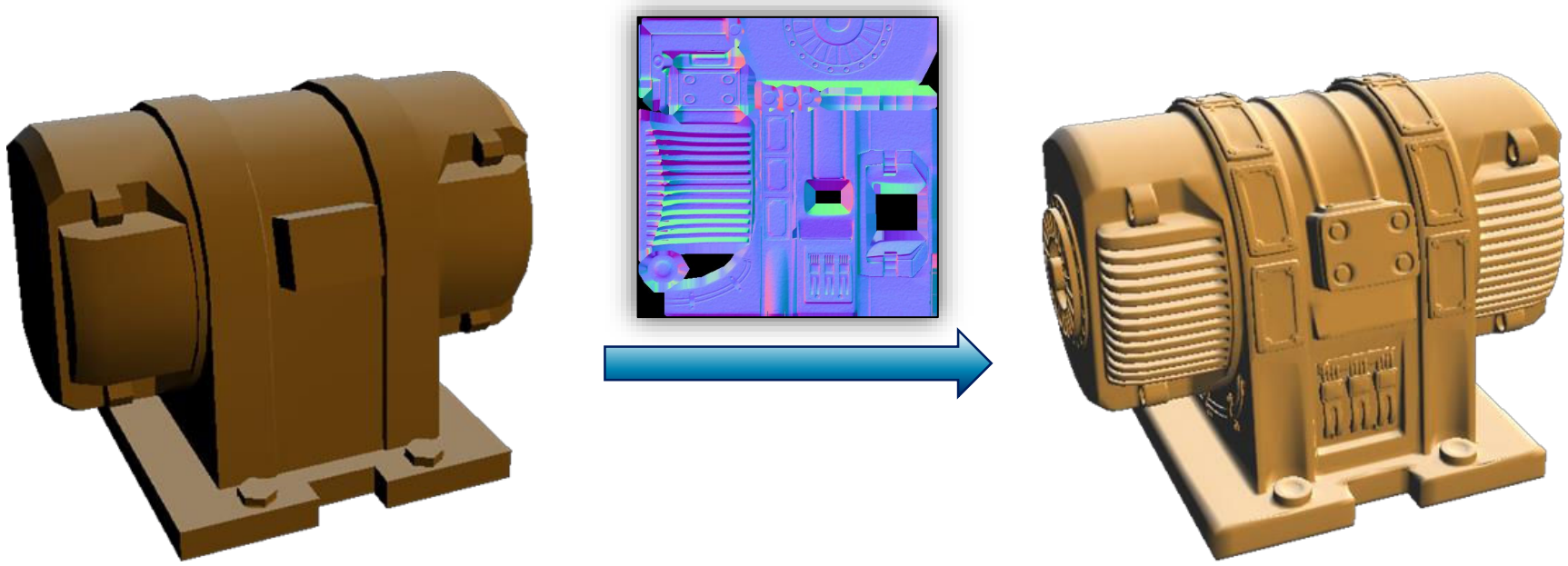
erkennen Sie ein Problem
mit Textur-Atlanten,
das hier „gelöst“ wurde?

Textur-Atlas

Anwendungen

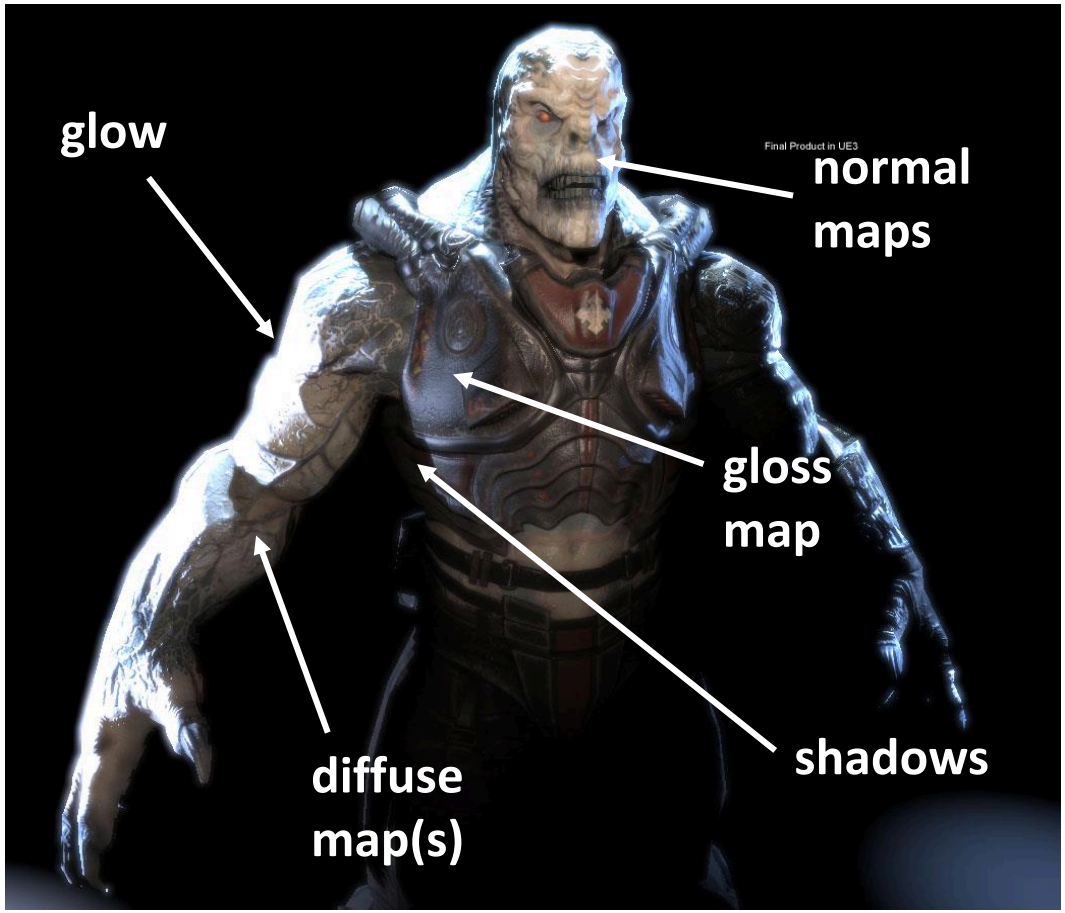
- Speichern einer Funktion/Daten auf der Oberfläche: Ambient Occlusion
- „Painting“: Erstellen der Textur direkt auf dem Objekt
- Berechnung von Normal-Maps (für Bump/Normal-Mapping) aus fein aufgelösten Dreiecksnetzen
- Vorsicht bei Textur-Filterung und Mip-Mapping!

Normal-Map
(Normalen im Tangentenraum)



Texturen und Beleuchtungstechniken

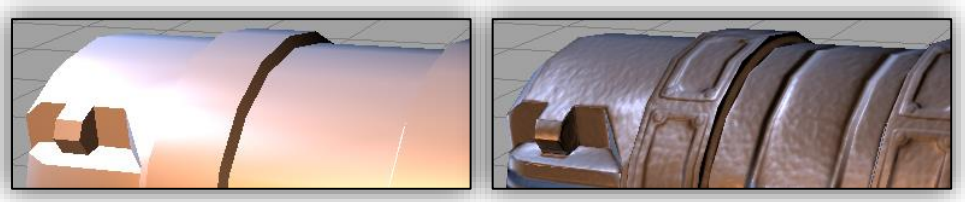
- Bild: Unreal Engine



Zusammenfassung: Texturierungstechniken

Bump/Normal Mapping

- Veränderung bzw. Ersetzen der Normale für die Beleuchtungsberechnung

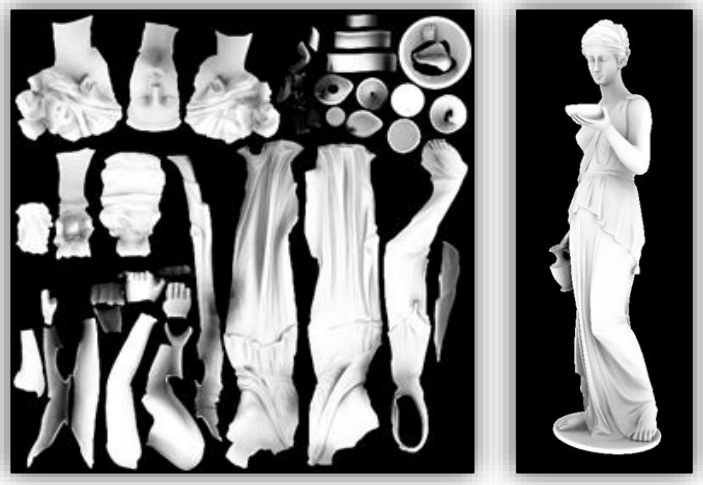


Displacement Mapping

- (Unterteilung und) Veränderung der Geometrie mit Texturinformation

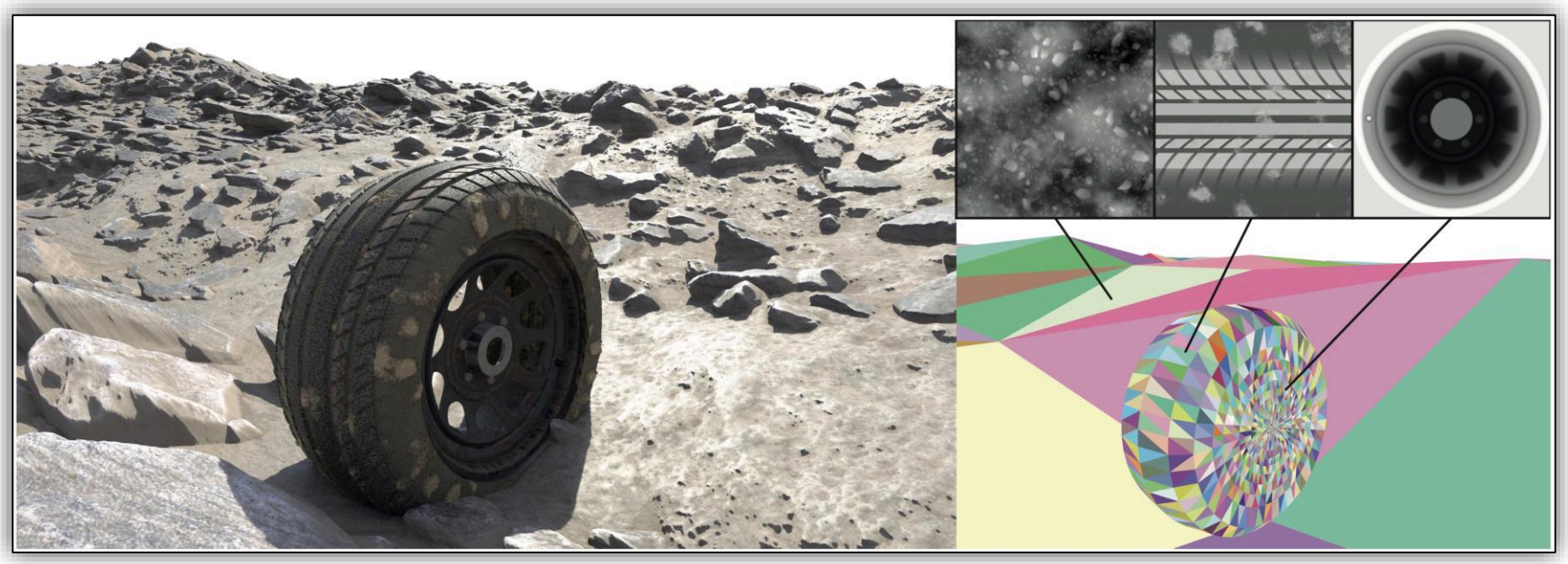
Textur-Atlanten

- Ambient Occlusion, Farbe, Normal Maps, …
- Achtung bei (Vor-)Filterung



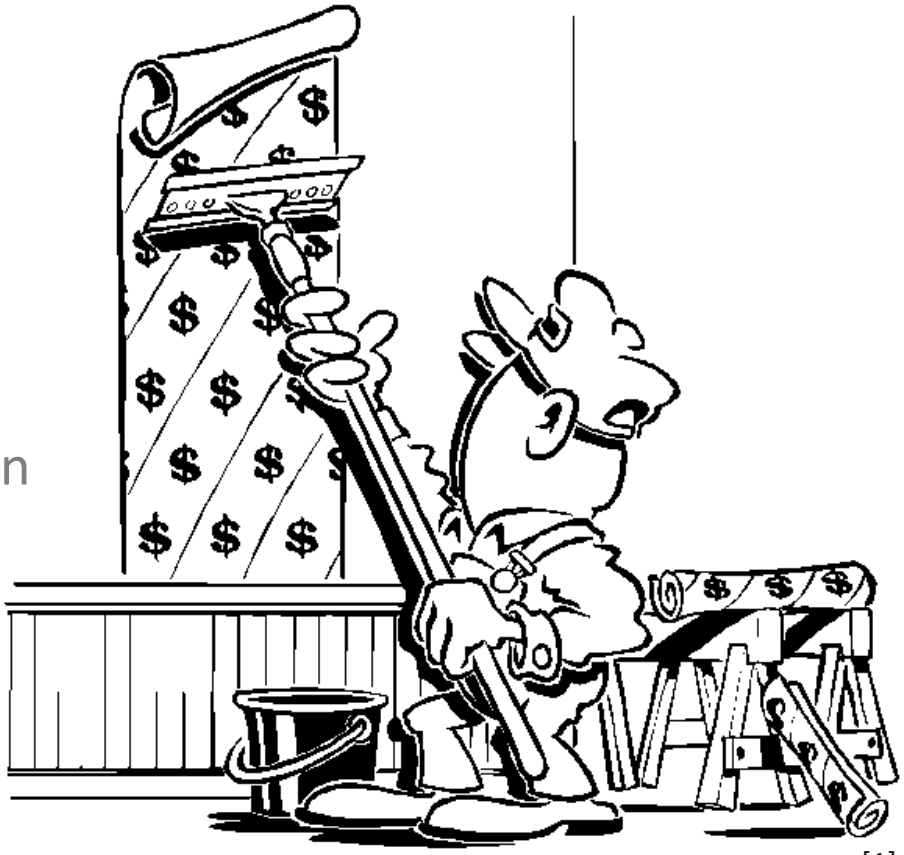
Beispiel: Aktuelle Forschung

- **RMIP: Displacement ray tracing via inversion and oblong bounding**
  Théo Thonat, Iliyan Georgiev, François Beaune, Tamy Boubekeur
  Adobe Research, SIGGRAPH Asia 2023
- Datenstruktur: rectangular minmax image pyramid (RMIP), um Minimum und Maximum in einem rechteckigen Bereich der Displacement Map zu bestimmen



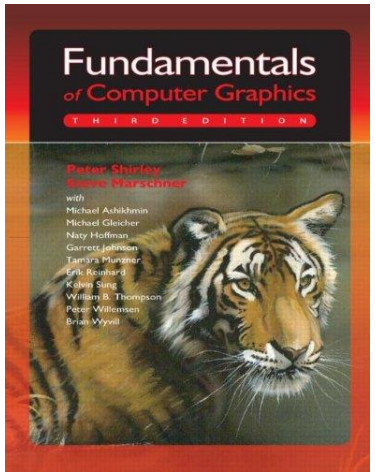
Inhalt

- Einführung, Texturquellen
- Abbildung, Parametrisierung, Texturkoordinaten
- Filterung von Texturen
- Texturen und Beleuchtungstechniken
- „Sonstiges“
- Environment-Mapping und bildbasierte Beleuchtung
- Shadow Mapping (eigentlich später)



[1]

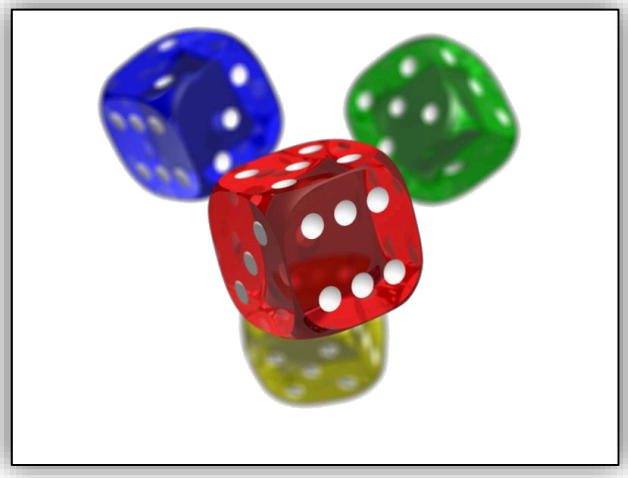
- **Fundamentals of Computer Graphics**, P. Shirley, S. Marschner, 3rd Edition, AK Peters
  → Kapitel 9 (Signal Processing)
  → Kapitel 11 (Texture Mapping)

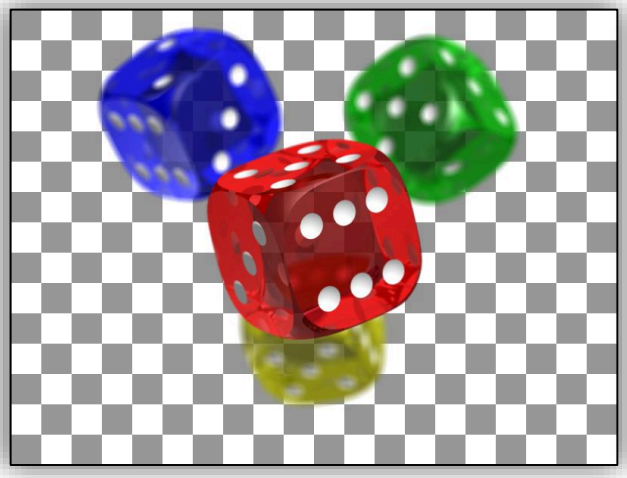


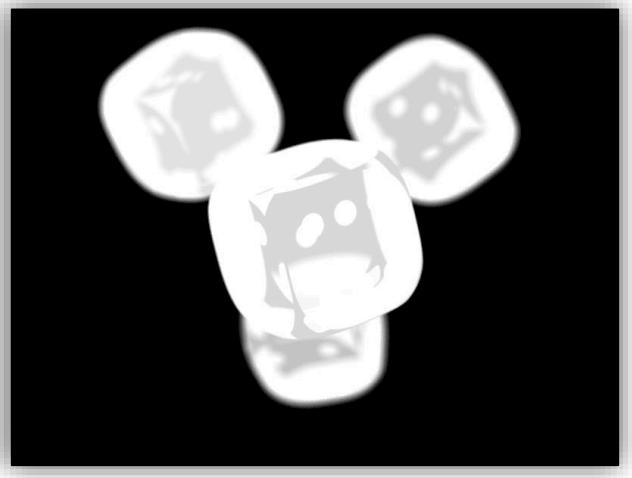
[2]

Transparenz und Alpha-Test

- Rasterbilder (und v.a. Texturen) oft mit 4-Kanälen gespeichert, z.B.
  - 24 Bit Farbinformation und
  - 8 Bit Alpha-Kanal (α = Opazität, Gegenteil von Transparenz)
  - RGBA-Format häufig bei Texturen

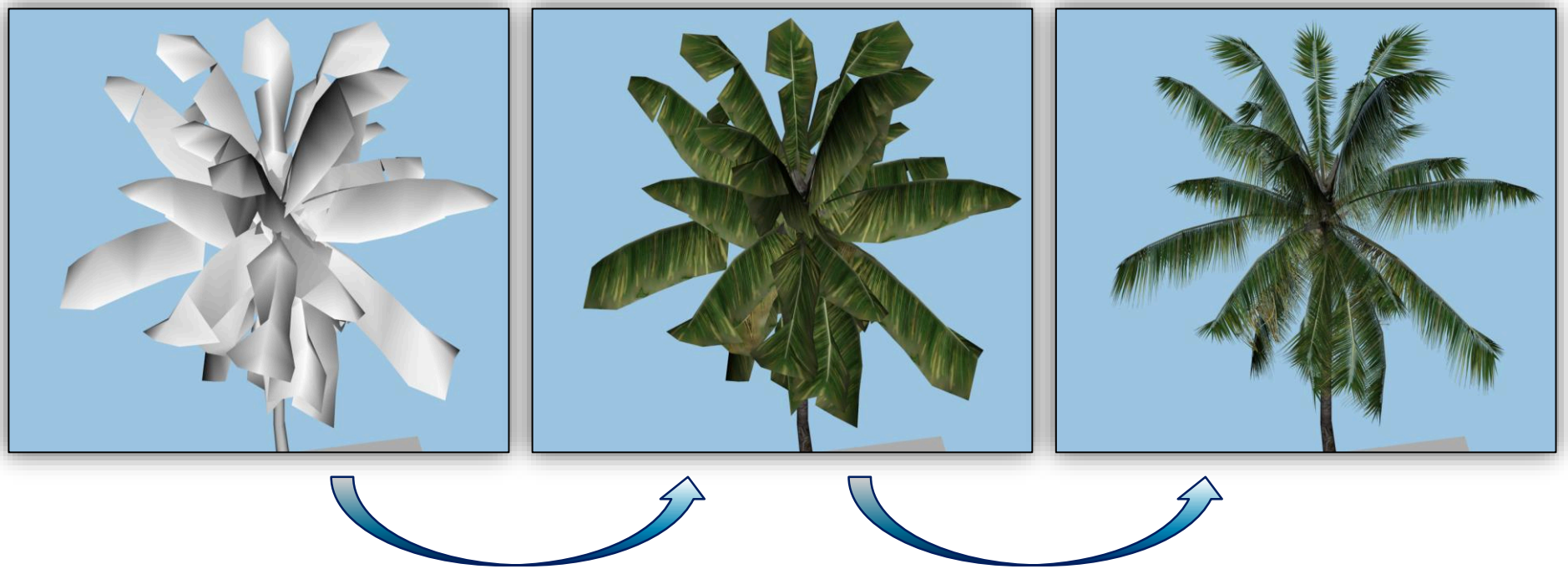


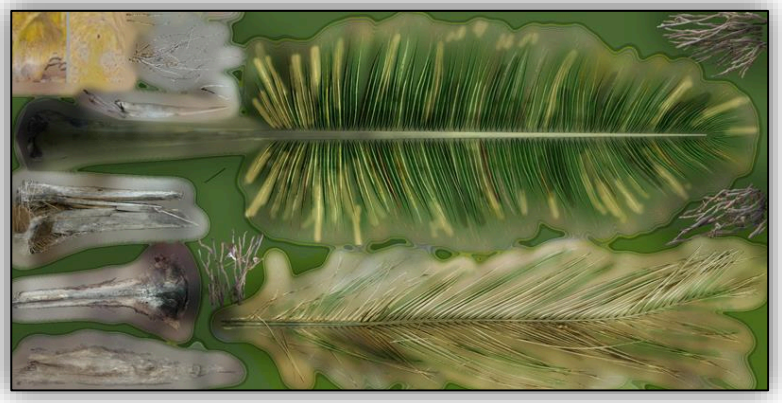




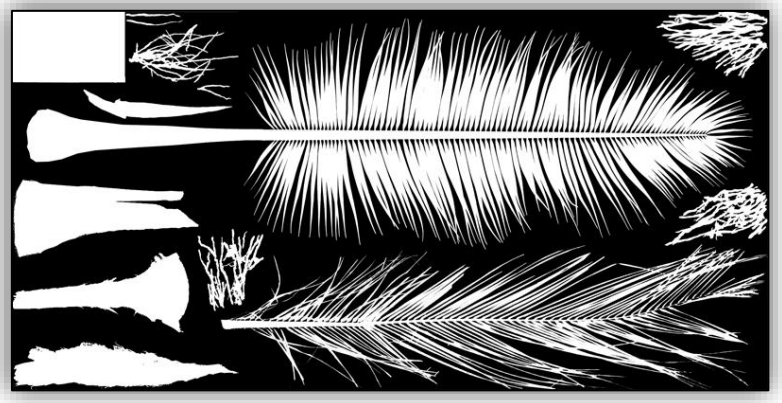
[25] http://de.wikipedia.org/wiki/Alphakanal

Transparenz und Alpha-Test





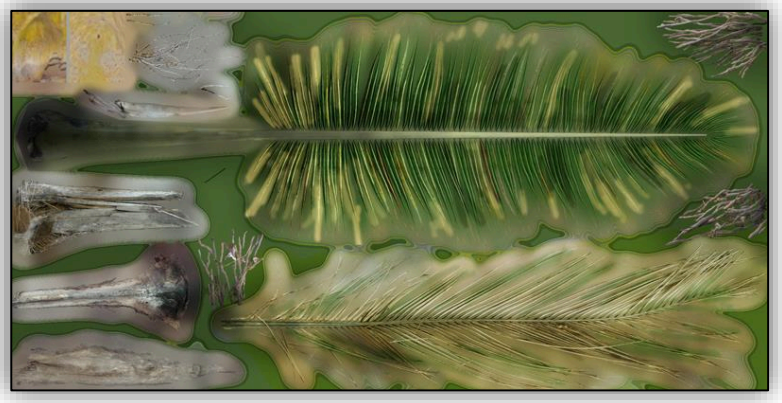
24 Bit Farbinformation



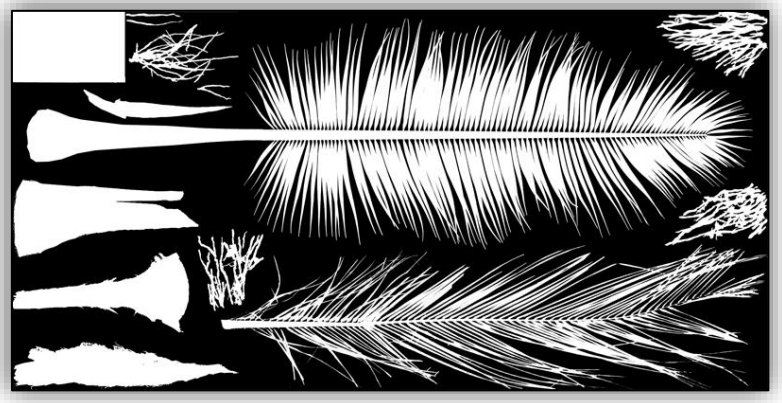
8 Bit Alpha-Kanal
schwarz = transparent

Transparenz und Alpha-Test

- **(Semi-)Transparenz**: verwende Alpha-Kanal, um die Transparenz (also z.B. k_t im Raytracing) eines Oberflächenpunkts zu bestimmen (Alpha/Opacity **Map**)
- **Alpha-Test**: verwerfe Objekt-/Schnittpunkt, wenn $\alpha < threshold$ (Alpha/Opacity **Mask**)
- (Vorgriff: beim Raytracing bereiten transparente Texel in Alpha Maps mit $k_t = 1$ keine Probleme, beim Rasterisieren schon!)



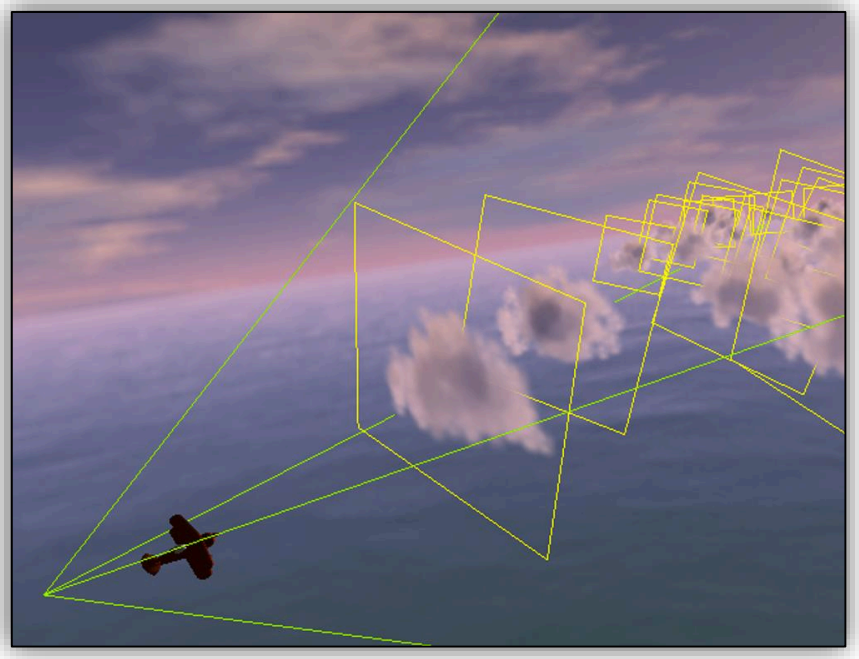
24 Bit Farbinformation



8 Bit Alpha-Kanal
schwarz = transparent

Impostors

- Impostor: texturiertes i.d.R. senkrecht zur Kamera ausgerichtetes Polygon
- geeignet insb. für weit entfernte „Objekte“, um die sich der Betrachter nicht (schnell) herumbewegt oder Partikelsysteme (Rauch, Funken, …)

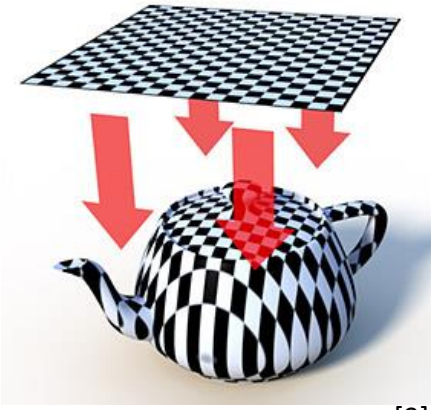


Mark Harris,
http://www.markmark.net/clouds/

Niniane Wang, http://niniane.org/clouds/

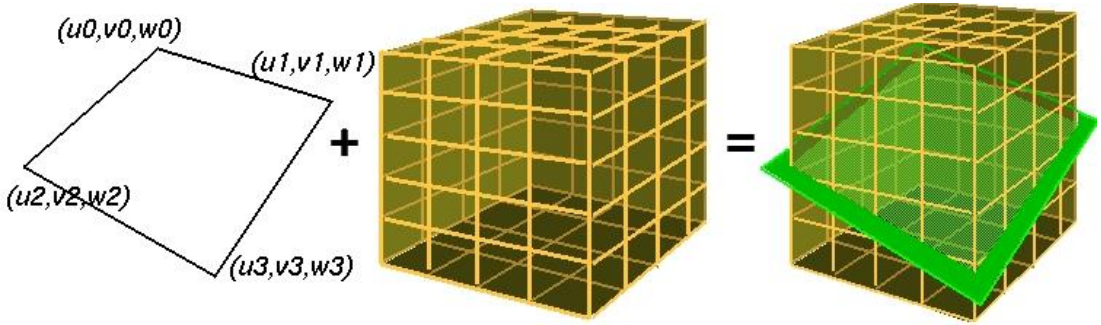
3D-Texturen für Oberflächen

- Probleme von 2D-Texturen
  - manchmal erkennbarer Tapeten-/Furnier-Effekt
  - Bestimmung der Texturkoord. für komplexe Objekte, Verzerrung, Filterung bei Textur-Atlanten…



[9]

- Solid-Textures (3D-Texturen)
  - „Herausschneiden einer Skulptur“ durch Zuweisung von 3D-Texturkoordinaten an Vertices
  - Vorteil: Parametrisierungsproblem fällt praktisch weg
  - Nachteil: großer Speicherplatzbedarf der Textur (RGB $512^3 \rightarrow 384$MB)
  - woher bekommt man eine 3D-Textur? Aufnahme einzelner Schichtbilder?

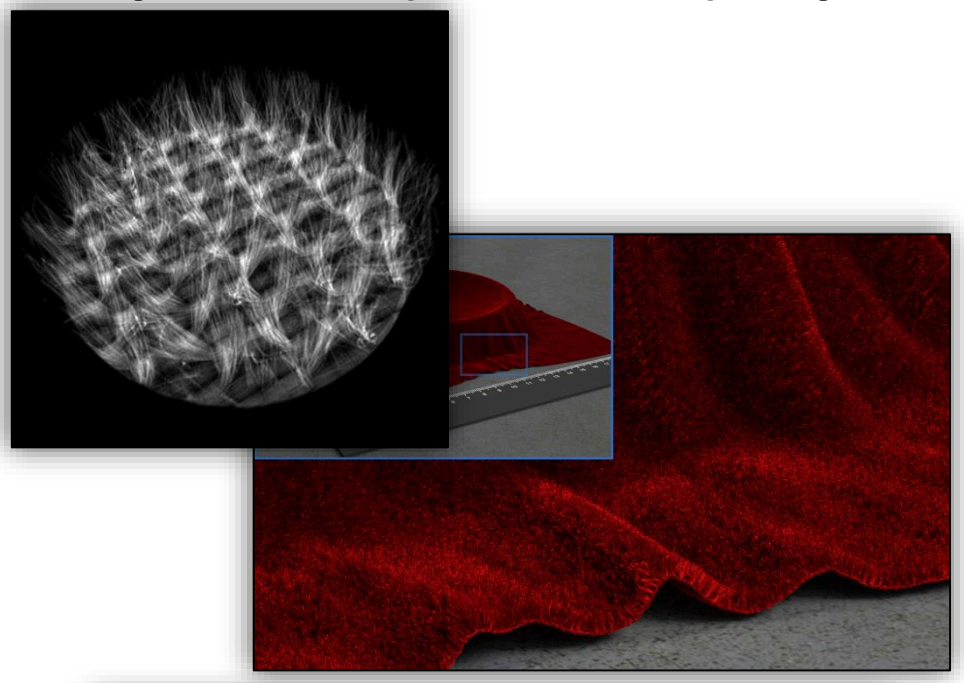


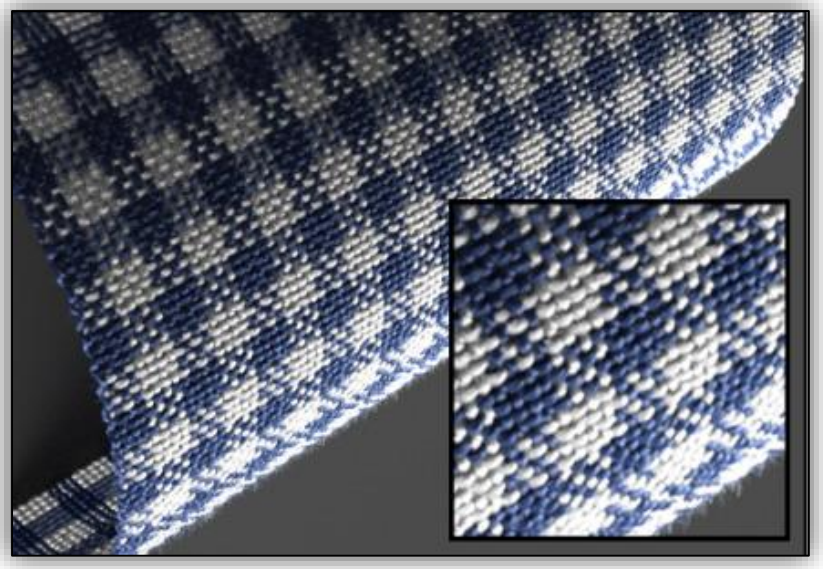
[9]

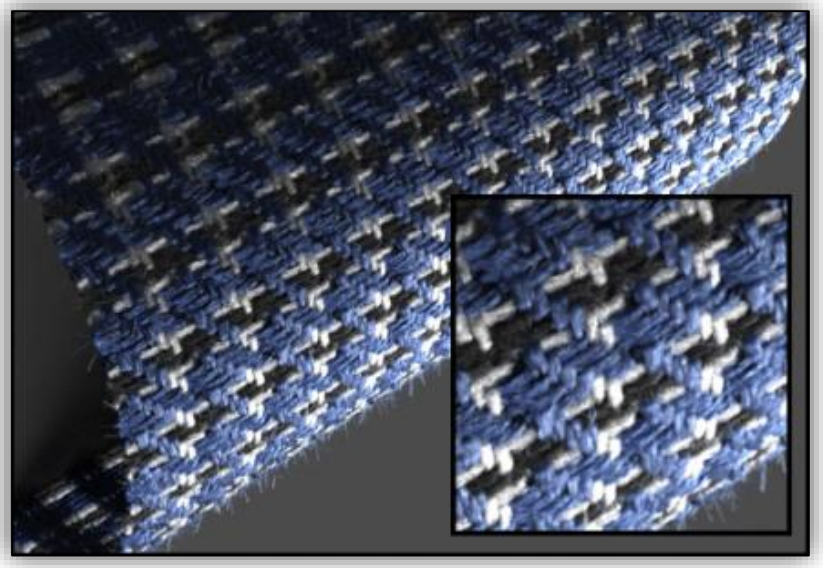
3D-Texturen

- Simulation einer Explosion

Akquirierte (Mikro-CT) & prozedurale 3D-Texturen





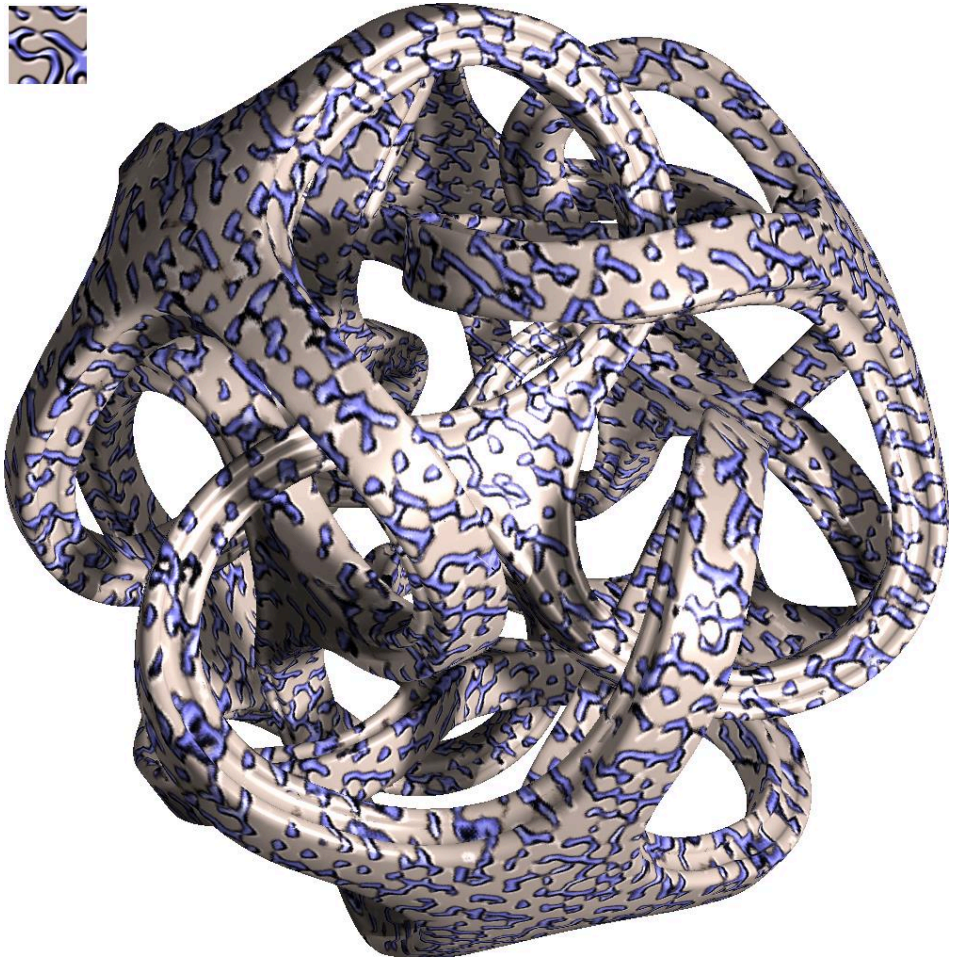


3D-Texturen

- Prozedurale Wolken aus SIGGRAPH-Kurs „Advances in Real-Time Rendering 2017“

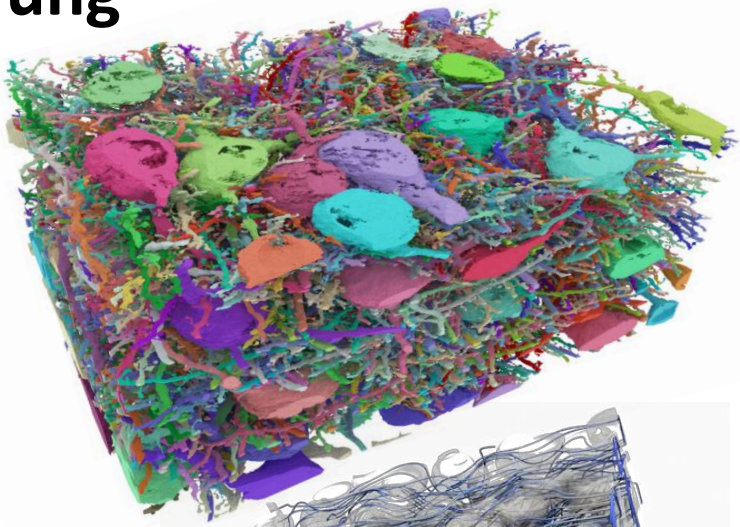
Lazy Solid Textures

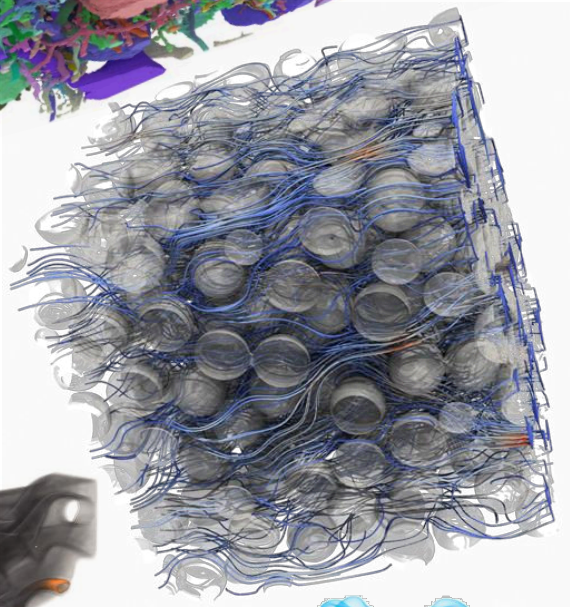
- Verfahren zur Berechnung von 3D Texturen aus 2D Texturen (http://www-sop.inria.fr/reves/Basilic/2008/DLTD08/ [27])

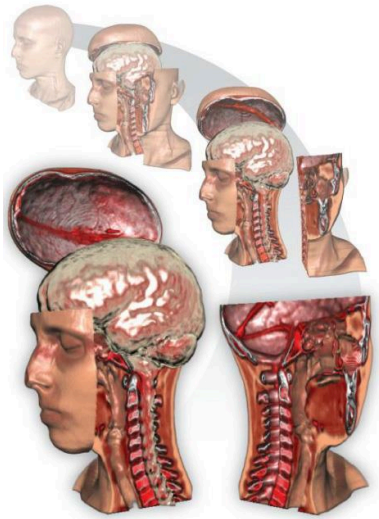


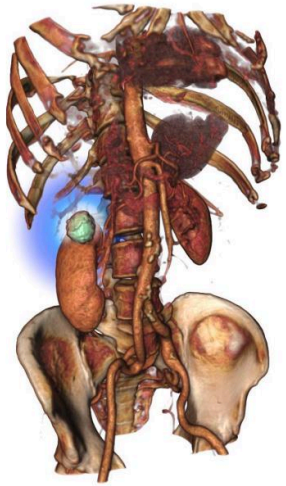
3D-Texturen und Volumenvisualisierung

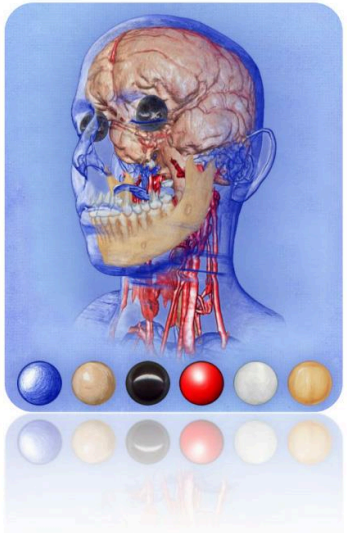
- … von Simulations- oder Messdaten
  - Röntgenabsorption im Körper/Material (Computertomographie), MRTs
  - Feuchtigkeit in Atmosphäre
  - Dichteverteilung im Erdinneren
- Daten oft auf uniformen 3D-Gittern
  - speichern/verwenden als 3D-Textur
  - oft zeitabhängig (3D+t)
  - Millionen bis Milliarden von Zellen (**Voxel**)





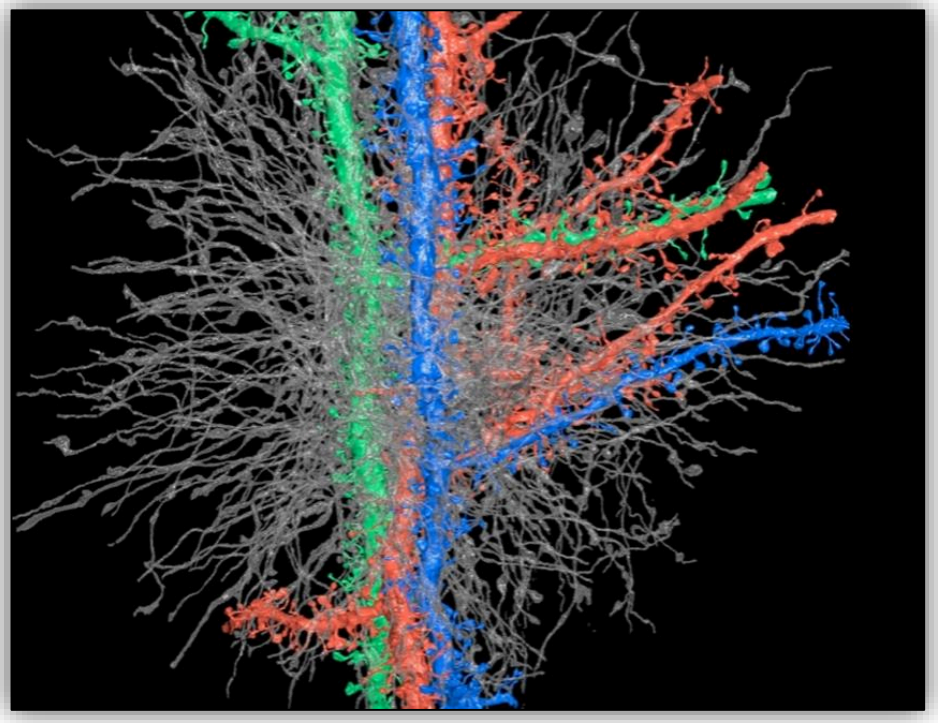






Beispiel: Ultra-Thin Section Electron Microscopy

- Visualisierung neuronaler Verbindungen
  (links 3 Dendriten/Zellfortsätze und Axone, rechts: Axone eingefärbt)
- 1mm^3 Gehirngewebe entspricht 20000 Slices à 40 Gigapixel, **800 TB**
- Schichtweise Aufnahme (aufwändige Vorbereitung der Probe, gefolgt von Registrierung/Stitching, Auflösung 5nm Pixel, 50nm pro Schicht)



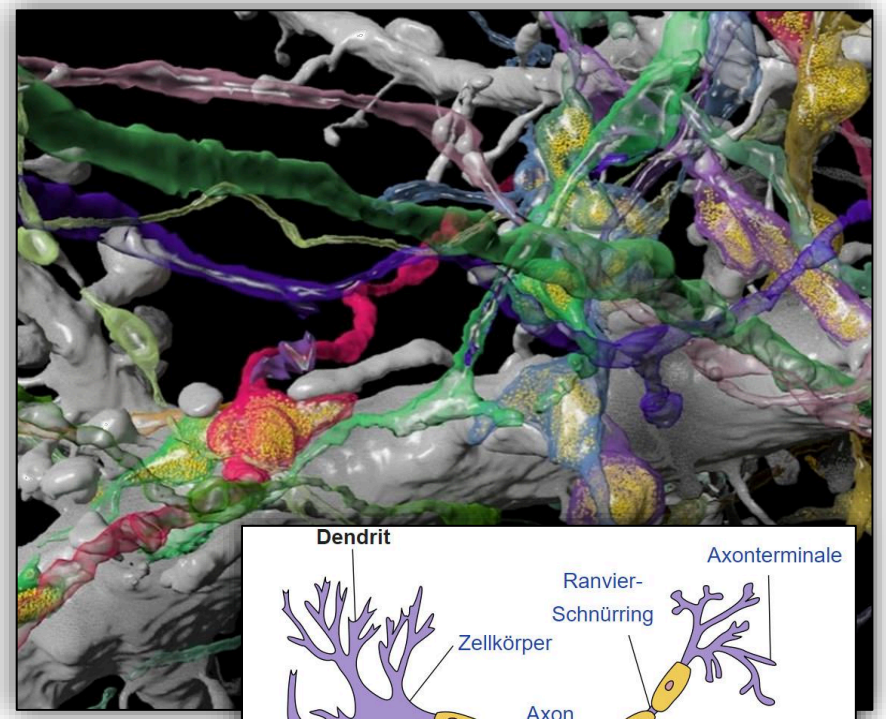
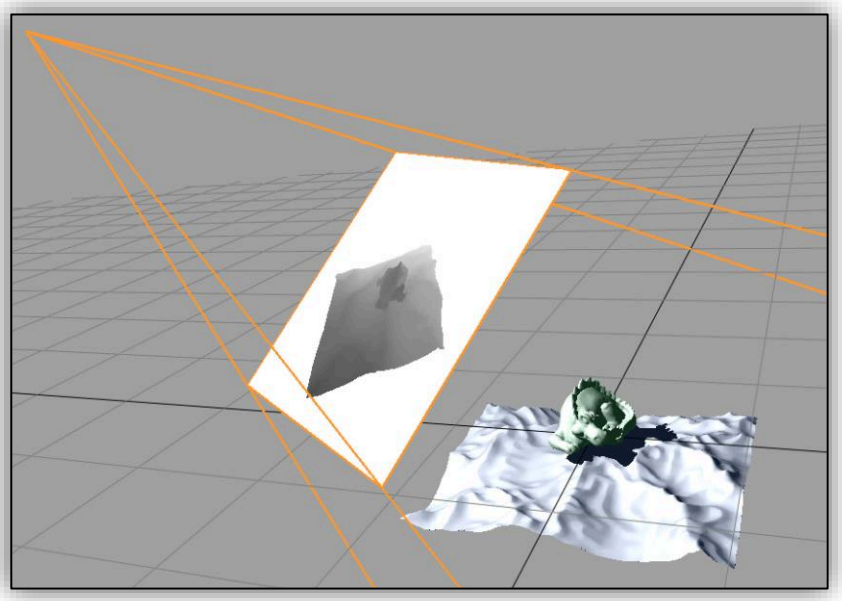


Bild oben: Pfister et al., Visualization in Connectomics, arXiv 2012 [29]
Bild rechts: Wikipedia [30]

Texturabbildungen: Überblick

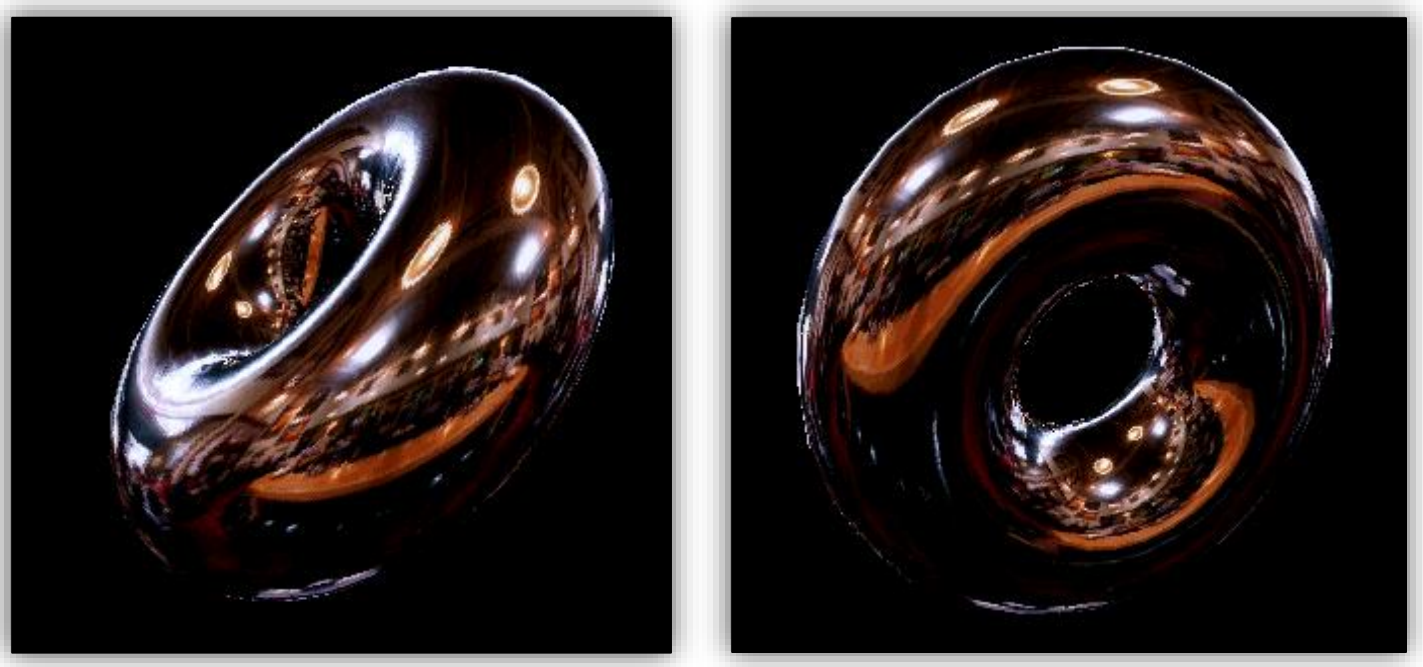
- Projektion auf andere Flächen (Kugel, Zylinder, Box, Ebene, ...)
- Flächenparameter (u, v) (z.B. Bézier-Patch)
- explizites Festlegen von sog. Texturkoordinaten (s, t, r) (auch $(u, v, \dots)$)
- Objekt- (x_o, y_o, z_o) oder Weltkoordinaten (x_w, y_w, z_w)
- Bildschirm- bzw. Kamerakoordinaten (x_s, y_s) (Shadow Maps)
- Richtungsvektoren $(\mathbf{R}, \mathbf{N}, \dots)$ (Environment Map, Image-based Lighting)
- Funktion der obigen Parameter (z.B. in „Shadern“=pro Pixel Berechnung)



Environment Mapping[1] (selten auch Reflection Mapping)

Motivation

- Darstellung reflektierender Objekte mit Spiegelung (auch realer aufgenommener) Umgebungen *ohne geometrische Repräsentation*
  → (auch geeignet zur) Approximation der Reflexion ohne Raytracing, d.h. ohne einen Reflexionsstrahl wirklich zu verfolgen
- Grundidee: speichere Bild der Umgebung in einer Textur



[1] Blinn, J. F. and Newell, M. E. Texture and reflection in computer generated images Communications of the ACM Vol. 19, No. 10 (October 1976), 542-547 [31]

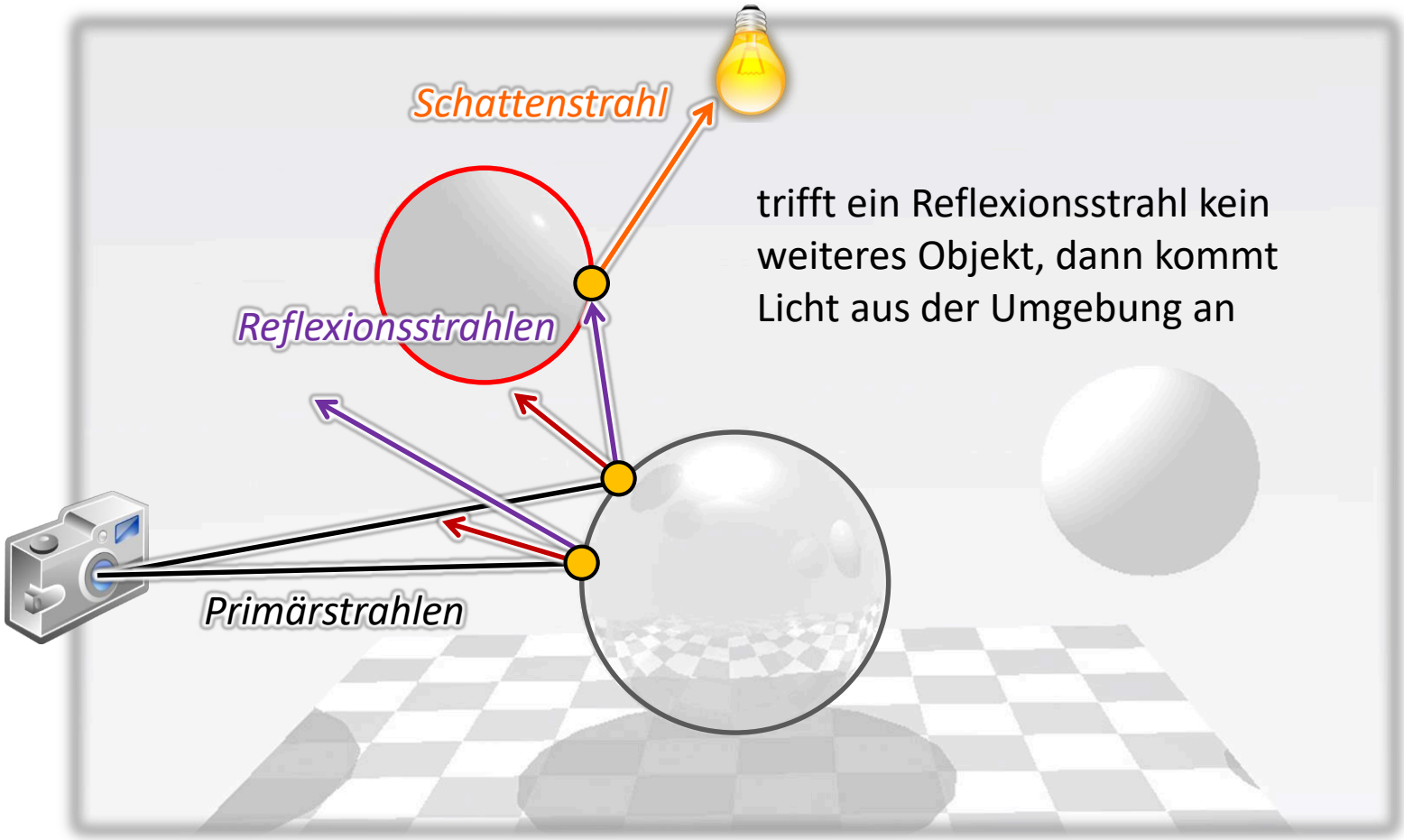
Environment Mapping

Beispiel

- … eine der ersten (kommerziellen) Anwendungen: Terminator 2 (1991)

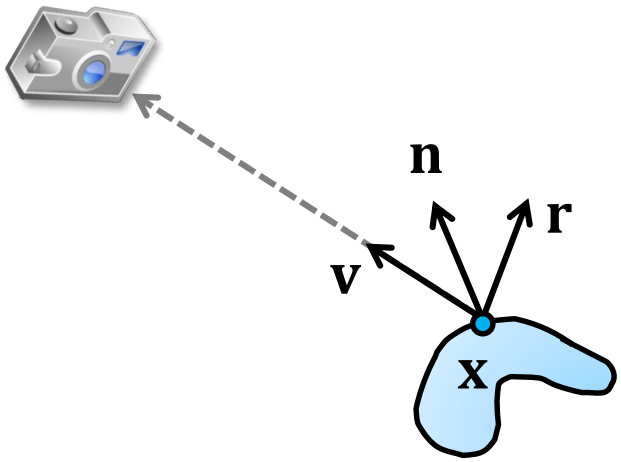
Environment Mapping

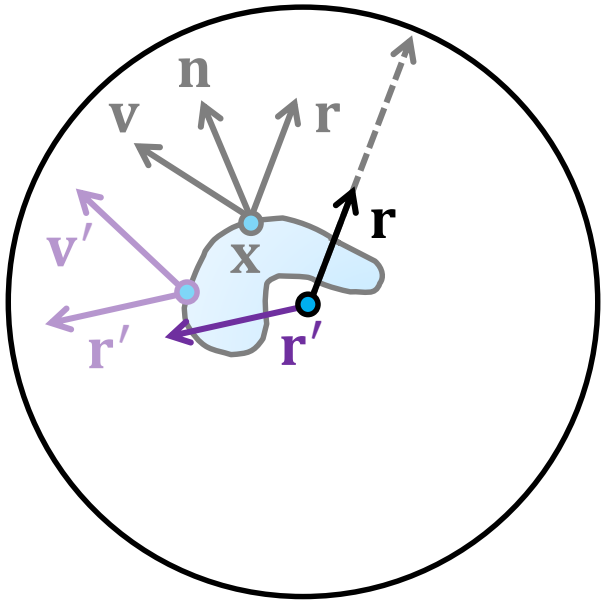
- (Environment Mapping wird auch oft mit Grafik-Hardware eingesetzt)
- beim Raytracing: trifft ein Strahl kein Objekt, dann wird das ankommende Licht aus der Environment Map ausgelesen



Environment Mapping

- eine Environment Map können wir uns zunächst als Textur auf einer virtuellen Kugel um ein Objekt/die Szene vorstellen
- **grundlegende vereinfachende Annahme**:
  nur die Richtung **r** des reflektierten Strahls wird beim Environment Mapping verwendet, der Ausgangspunkt **x** des Strahls wird ignoriert!

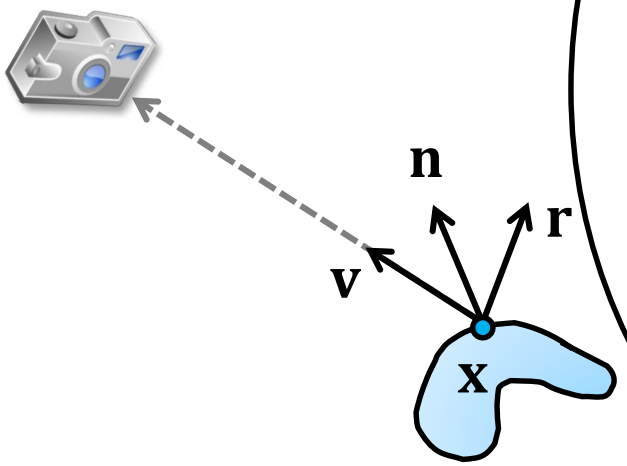


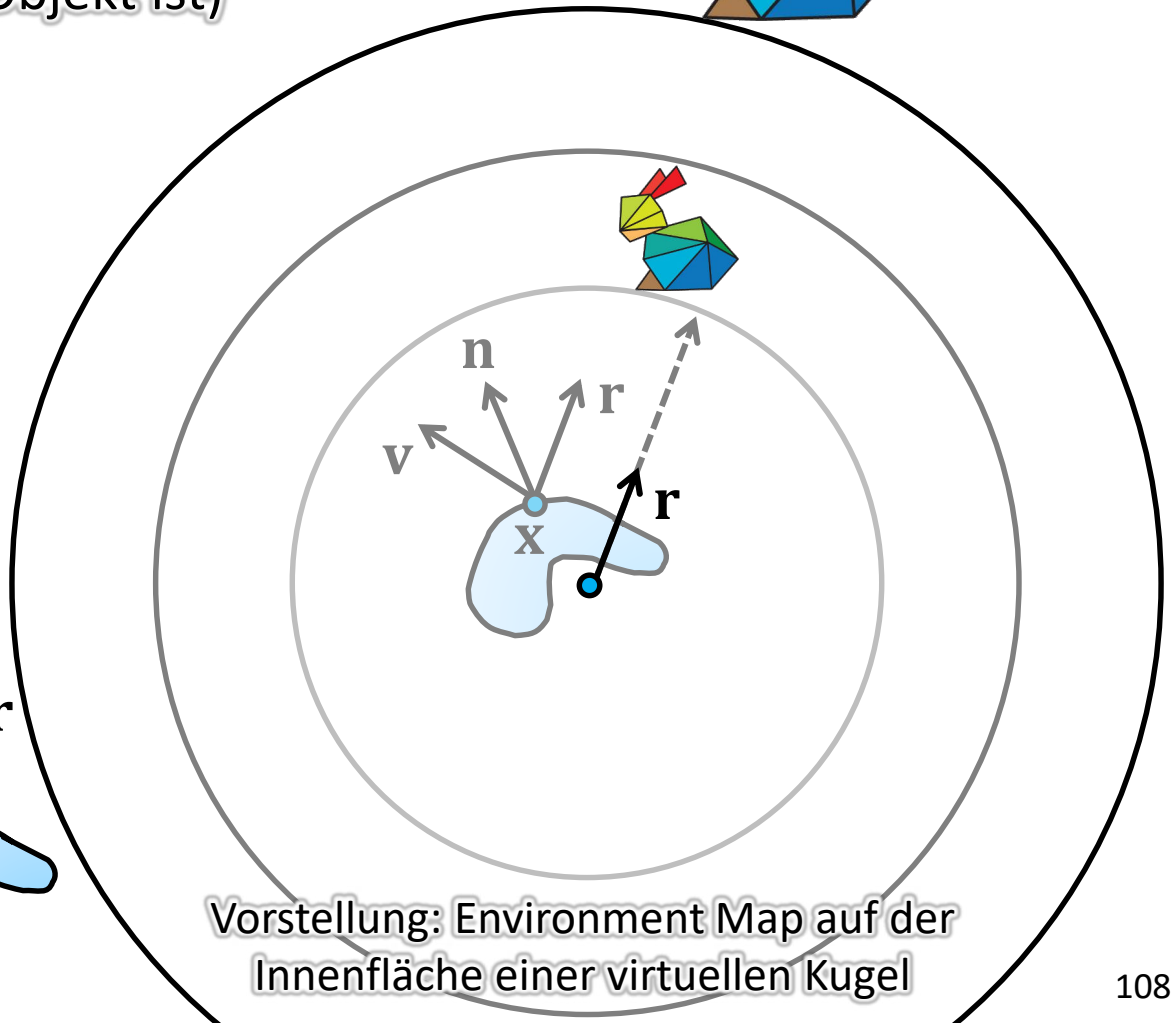


Vorstellung: Environment Map auf der Innenfläche einer virtuellen Kugel

Environment Mapping

- **Annahme:** nur die Richtung **r** wird verwendet
  - akzeptabel, wenn die Kugel sehr groß ist (gleichbedeutend mit: Umgebung weit weg vom spiegelnden Objekt ist)

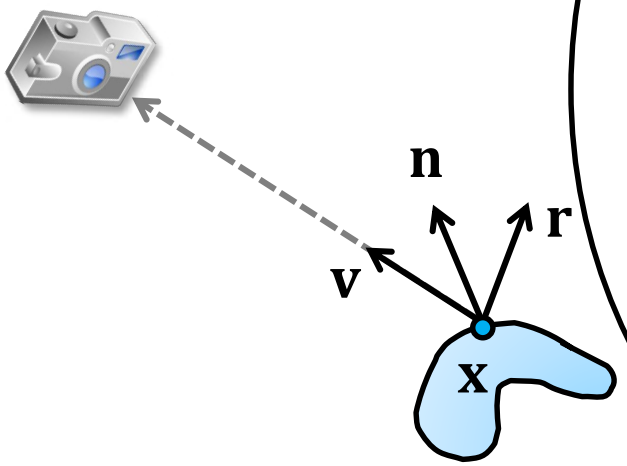


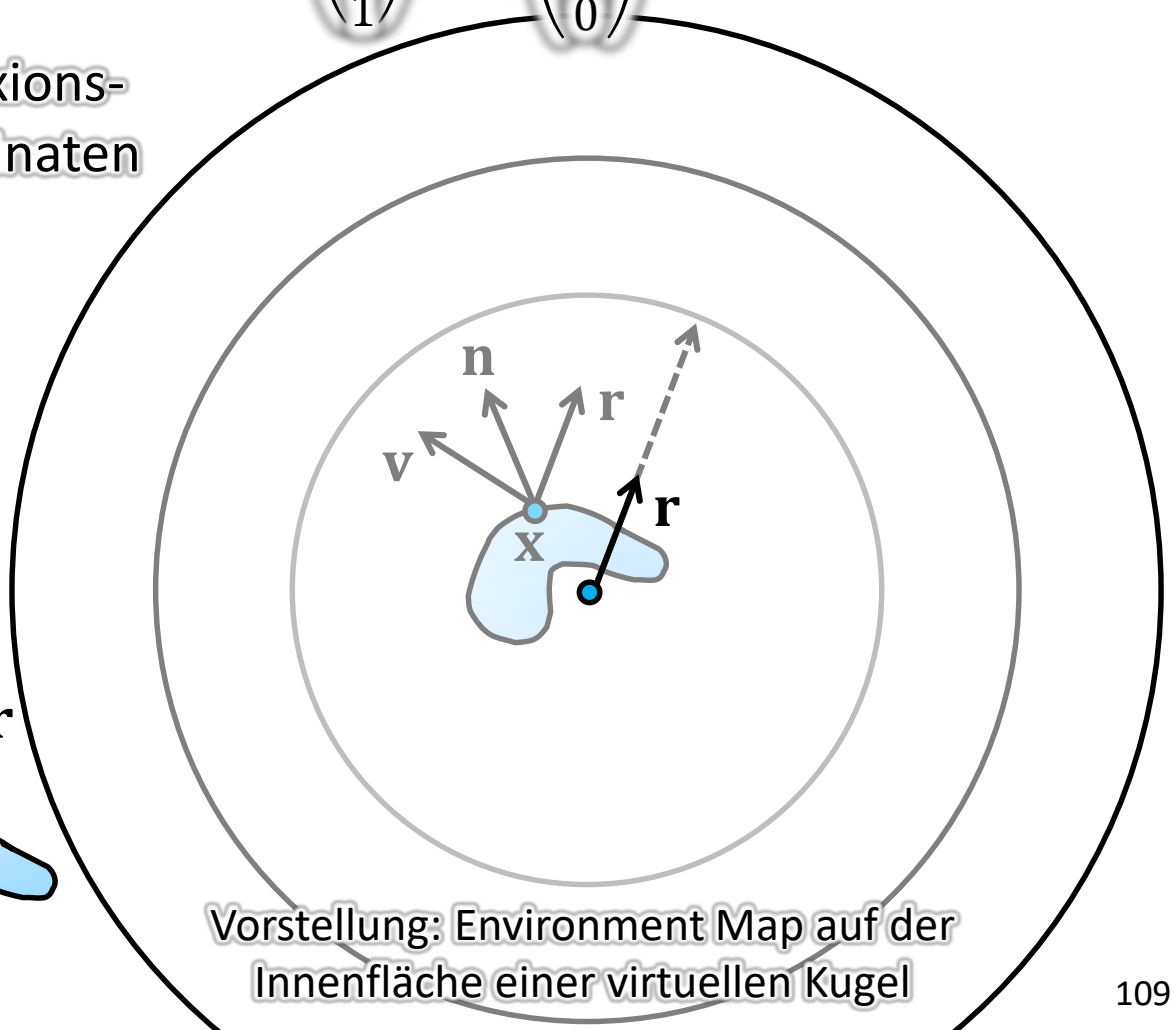


Vorstellung: Environment Map auf der Innenfläche einer virtuellen Kugel

Environment Mapping

- **Annahme:** nur die Richtung $\mathbf{r}$ wird verwendet
- Umgebung unendlich weit entfernt (homogene Koordinaten) mit $d \to \infty$: $\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} + d \cdot \begin{pmatrix} r_x \\ r_y \\ r_z \\ 0 \end{pmatrix}$
- Zugriff: konvertiere Reflexionsrichtung $\mathbf{r}$ in Texturkoordinaten





Vorstellung: Environment Map auf der Innenfläche einer virtuellen Kugel

Environment Mapping Parametrisierung

Latitude/Longitude-Maps (Kugel-Parametrisierung)

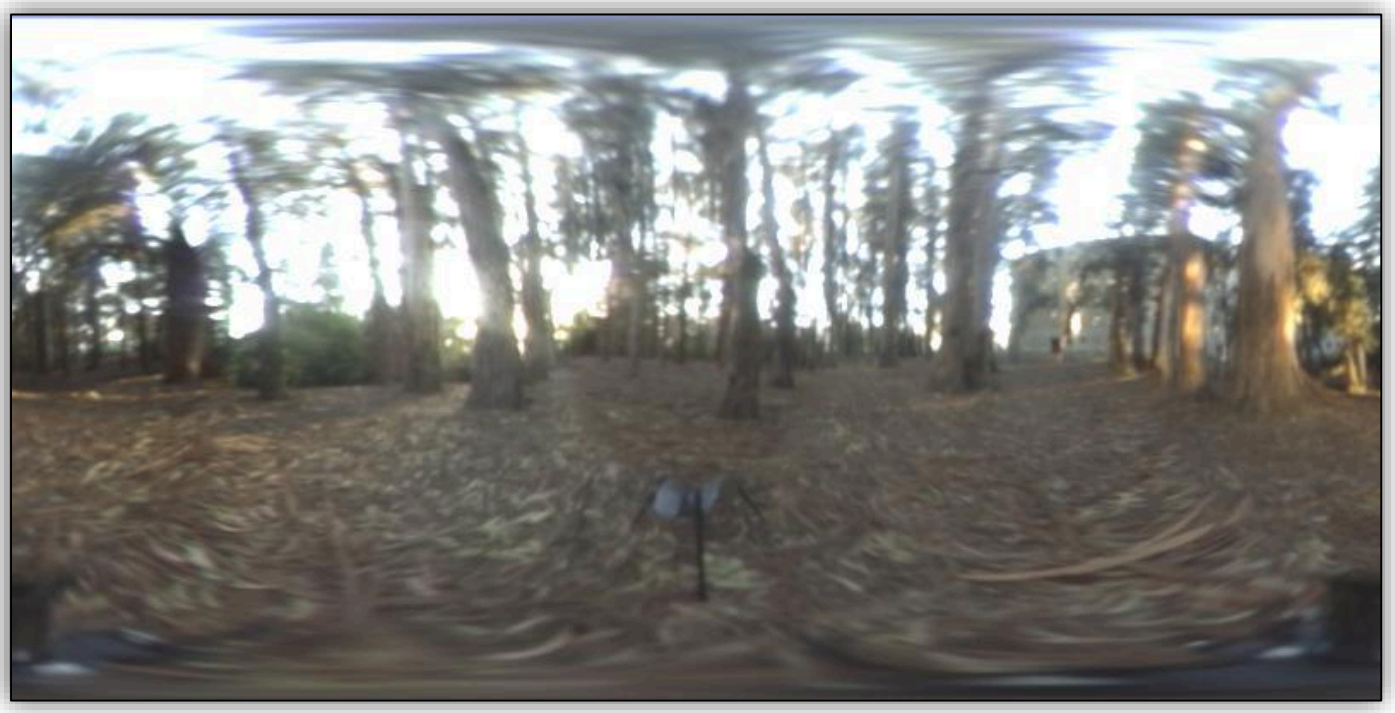
- Parametrisierung über Polarwinkel $\theta \in [0..\pi]$ zur z-Achse und Azimuthalwinkel $\phi \in [0..2\pi]$
- Berechnung der Winkel aus Richtungsvektor $\mathbf{r} = (r_x, r_y, r_z)$ mit $|\mathbf{r}| = 1$
  - $\theta = \text{acos}(r_z)$ bzw. $\phi = \text{atan2}(r_y, r_x)$, $t = \theta/\pi$ und $s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

Quelle ursprüngliche Environment Map: www.debevec.org [32]

Environment Mapping Parametrisierung

Latitude/Longitude-Maps

- vergleichsweise „teure“ Berechnung der Texturkoordinaten aus $\mathbf{r}$
- sehr ungleichmäßige Abtastung an den Polen: viele Texel für wenige Richtungen (vgl. oberer/untere Rand mit Äquator)



[32]

- gibt es bessere Parametrisierungen? Welche kann man direkt aufnehmen?

Environment Mapping Parametrisierung

- Experimente mit Garten-Dekoration…
  (Hintergrund und Bilder: https://www.pauldebevec.com/ReflectionMapping/)

- … etwas kalibrierter …

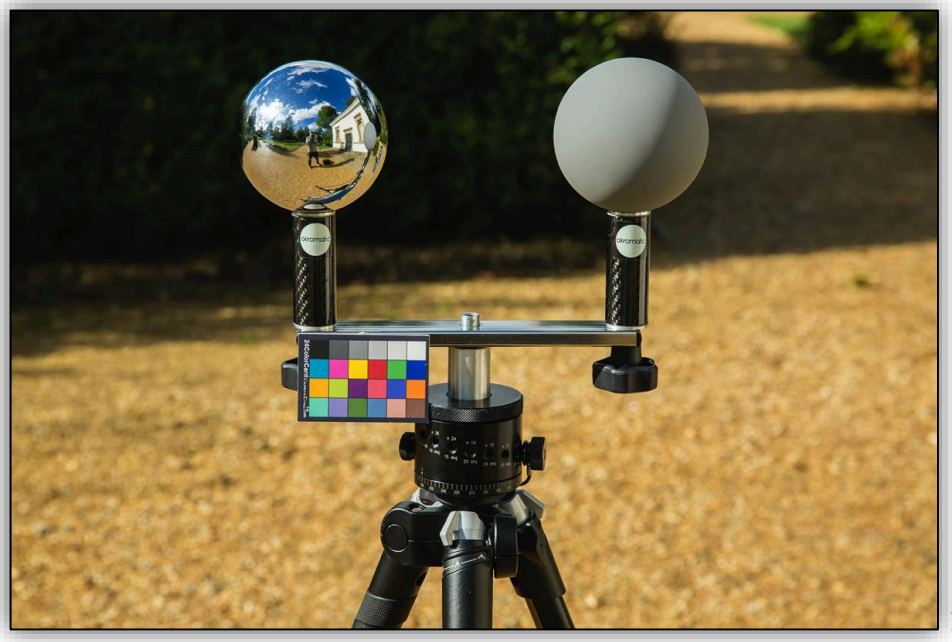
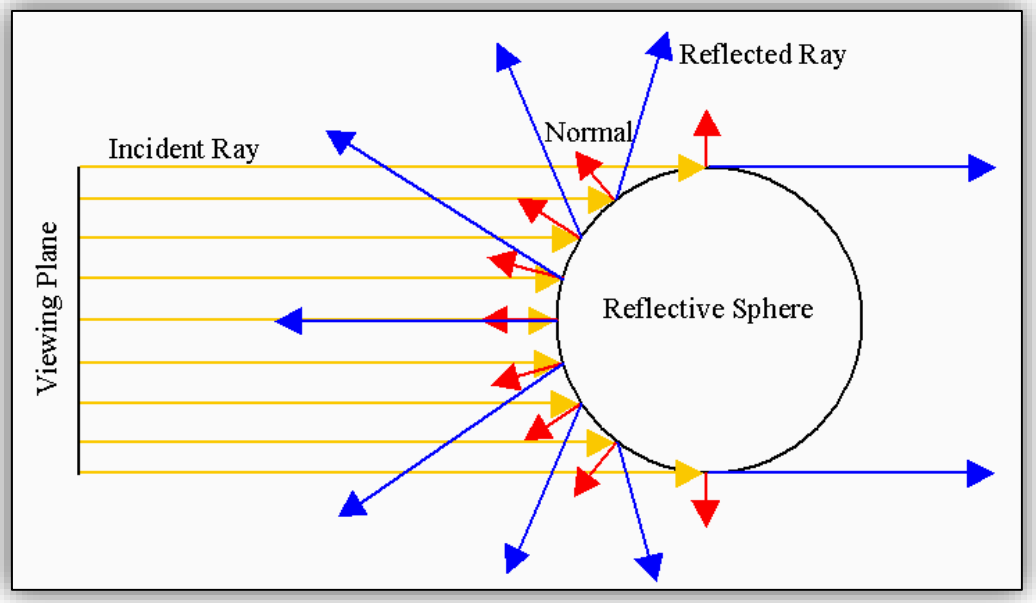


Bild:
https://beforesandafters.com/2021/04/27/~vfx-firsts-what-was-the-first-film-to-use-a-hdri-chrome-ball/

Environment Mapping Parametrisierung

Grundidee Sphere Mapping

- fotografiere kleine Spiegelkugel (zentriert im Bild) mit einem Teleobjektiv: „Kamerastrahlen“ sind beinahe parallel
- Bild auf der Spiegelkugel wird zur sog. *Sphere Map*
- blickpunktabhängig: um die Textur für eine Richtung $\mathbf{r}$ auszulesen, muss man die Aufnahmerichtung $\mathbf{v_0}$ kennen
  (d.h. ein $\mathbf{v_0}$ ist in Weltkoord. festgelegt)

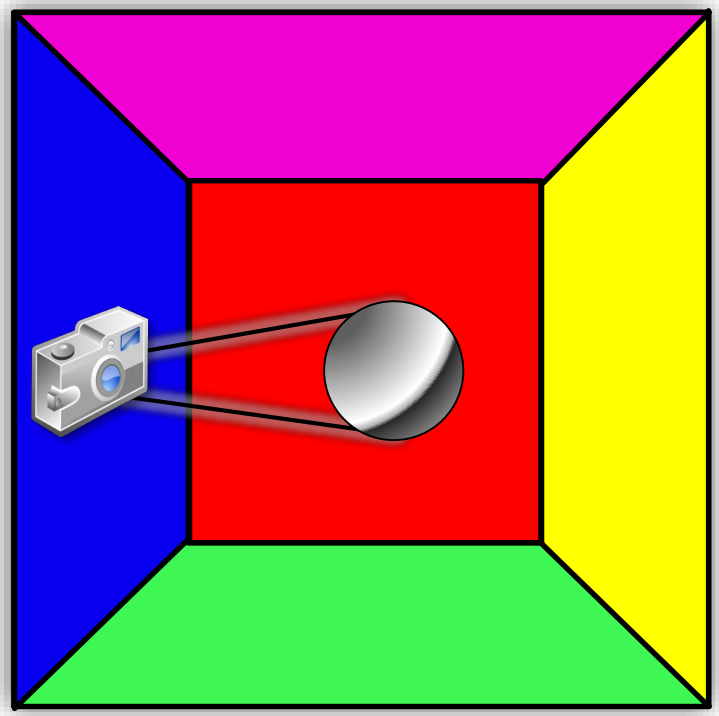


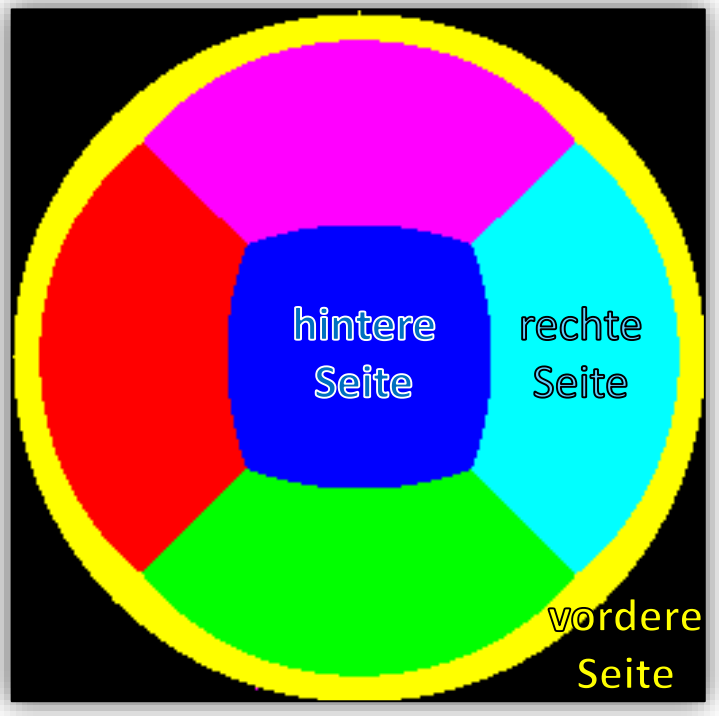
[34]

Environment Mapping Parametrisierung

Eigenschaften Sphere Mapping

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

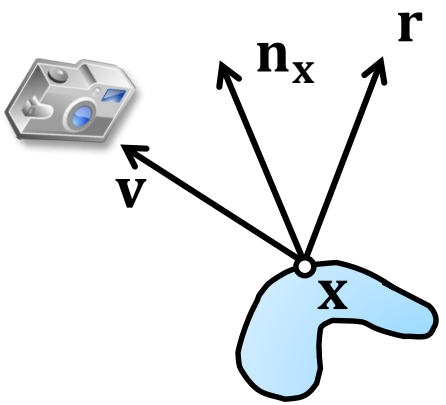




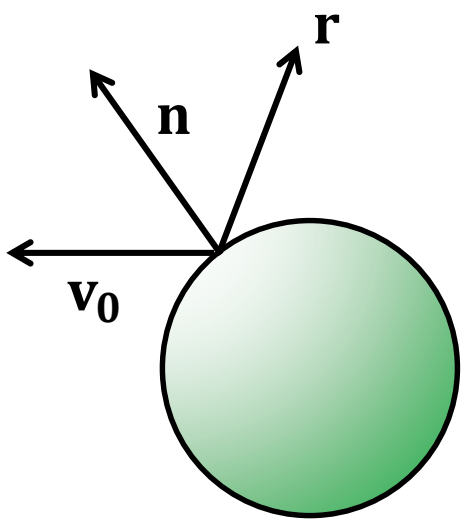
Sphere Mapping

Berechnung der Texturkoordinaten

- gegeben (Annahme: alle Vektoren normalisiert)
  - aktuelle Betrachterrichtung $\mathbf{v}$
    (bzw. Richtung wohin das Licht reflektiert wird)
  - Oberflächennormale $\mathbf{n_x}$
  - Reflexionsrichtung $\mathbf{r}$
  - Richtung $\mathbf{v_0}$ aus der die Sphere Map aufgenommen wurde



- gesucht: an welchem Punkt auf der Spiegelkugel konnten wir die Reflexionsrichtung $\mathbf{r}$ aufnehmen?



- Beobachtung:
  gesucht ist der Punkt, dessen Normale $\mathbf{n}$ mittig zwischen $\mathbf{r}$ und $\mathbf{v_0}$ liegt
  - wir haben aus Richtung $\mathbf{v_0}$ aufgenommen und bei perfekter Spiegelung war die Spiegelungsrichtung $\mathbf{r}$ genau dort zu finden

Sphere Mapping

Berechnung der Texturkoordinaten

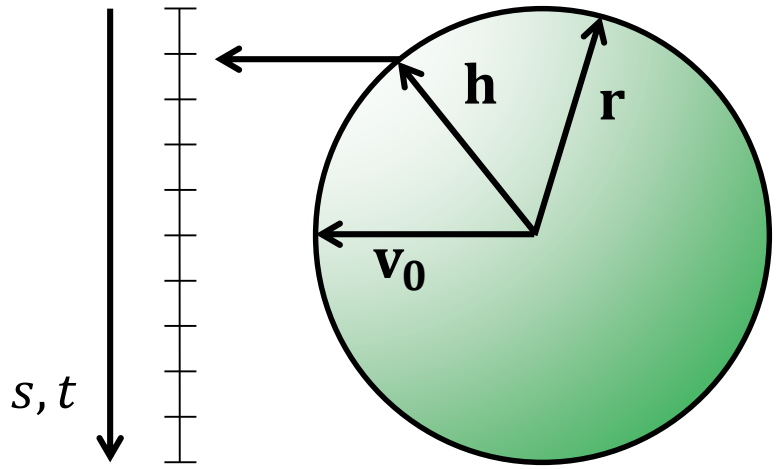
- wir suchen also den Punkt auf der Kugeloberfläche, dessen Normale $\mathbf{n}$ mittig zwischen $\mathbf{r}$ und $\mathbf{v_0}$ liegt, weil $\mathbf{r} = 2(\mathbf{v} \cdot \mathbf{n})\mathbf{n} - \mathbf{v}$

- auf der (Einheits-)Kugel gilt: ein Punkt $\mathbf{p} = (x, y, z)$ hat Normale (x, y, z)
- der sog. Half-Way Vector $\mathbf{h} = (\mathbf{r} + \mathbf{v_0})/|\mathbf{r} + \mathbf{v_0}|$ liegt genau zw. $\mathbf{r}$ und $\mathbf{v_0}$
  - … ist also die gesuchte Normale $\mathbf{n}$ und gleichzeitig der gesuchte Punkt

- wir erhalten die Texturkoordinaten durch orthogonale Projektion (hier: 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 $\mathbf{h}$

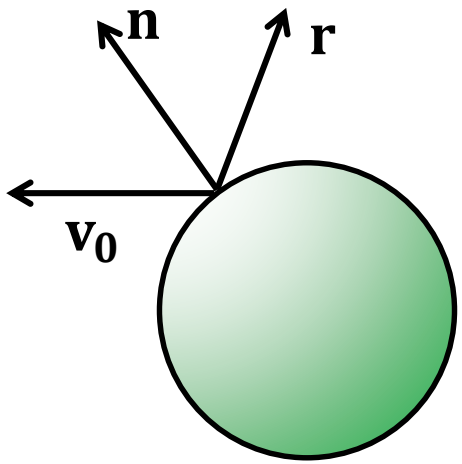


Sphere Mapping

Probleme und Nachteile

- wieder ungleichmäßige Abtastung, hier an den Rändern
  - Abtastrate maximal für Richtungen entgegen der Aufnahmerichtung
  - Singularität in Aufnahmerichtung (am Rand)
  - ungleiche Abtastung:
    deshalb eigentlich nur für Blickrichtungen ähnlich $\mathbf{v_0}$ akzeptabel/gut

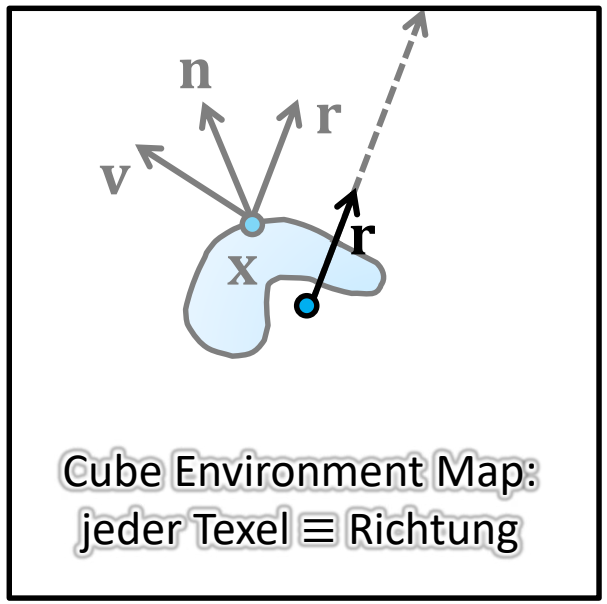
[32]



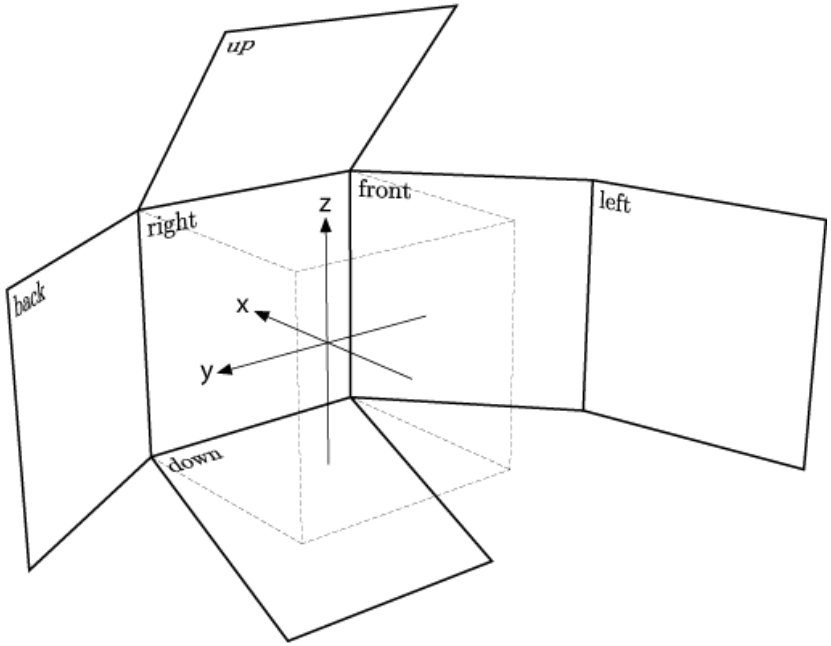
Environment Mapping Parametrisierung

(Sehr gebräuchliche) Parametrisierung: Cube (Environment) Maps

- jeder Punkt auf einer Würfeloberfläche entspricht einer Richtung
  - Cube Map besteht aus 6 × 2D-Texturen
  - Abbildung/Aufnahme einer Würfelseite: perspektivische Kamera mit FOV 90°
  - Texturfilterung wie bei Mip-Mapping möglich (aber Ränder beachten)



Cube Environment Map:
jeder Texel ≡ Richtung

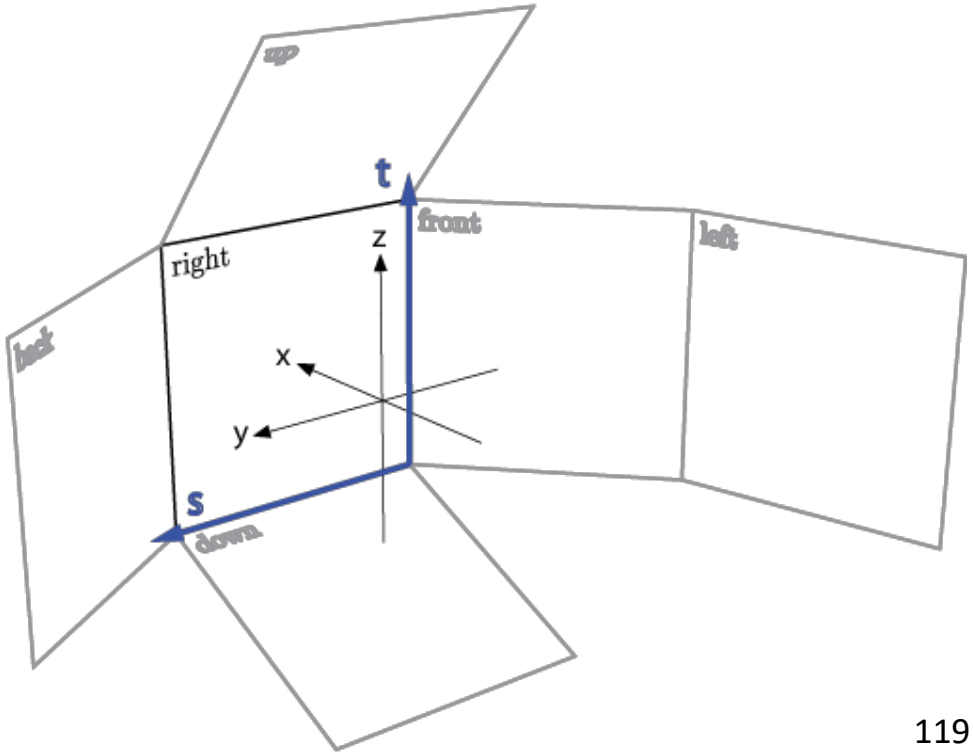


[32]

Cube Environment Maps

Berechnung der Texturkoordinaten

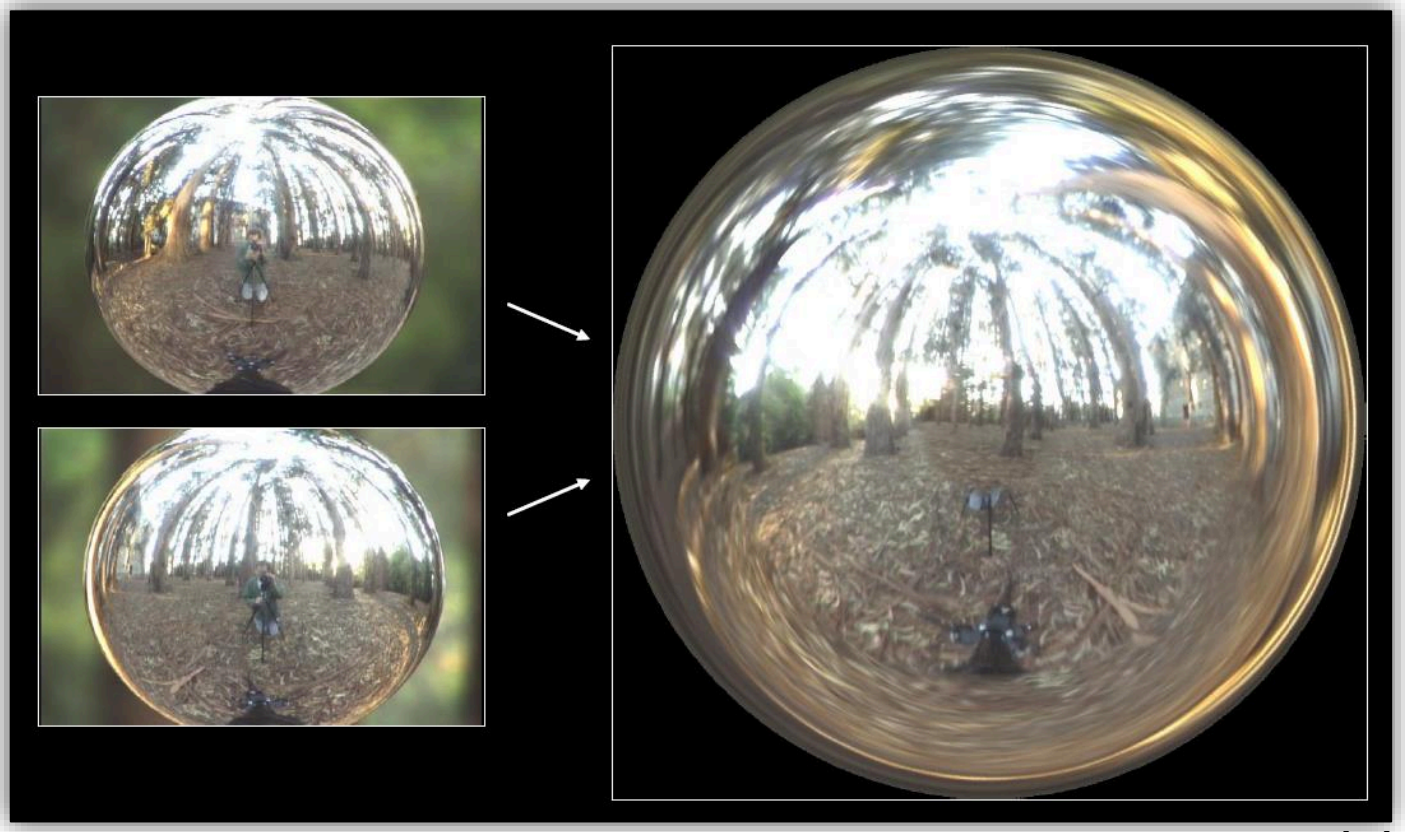
- gegeben Reflexionsrichtung $\mathbf{r} = (r_x, r_y, r_z)$ ($|\mathbf{r}| \neq 1$ erlaubt)
- betragsmäßig größte Komponente von $\mathbf{r}$ wählt Würfelfläche
  - $|r_x| > |r_y| \wedge |r_x| > |r_z|$ ⇒ „right“, wenn $r_x > 0$, „left“ sonst
  - $|r_y| > |r_x| \wedge |r_y| > |r_z|$ ⇒ „back“, wenn $r_y > 0$, „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** → löst das Problem der ungleichmäßigen Abtastung, ermöglicht nachträgliches Entfernen des Fotoapparats
  - Umrechnung in andere Parametrisierungen möglich

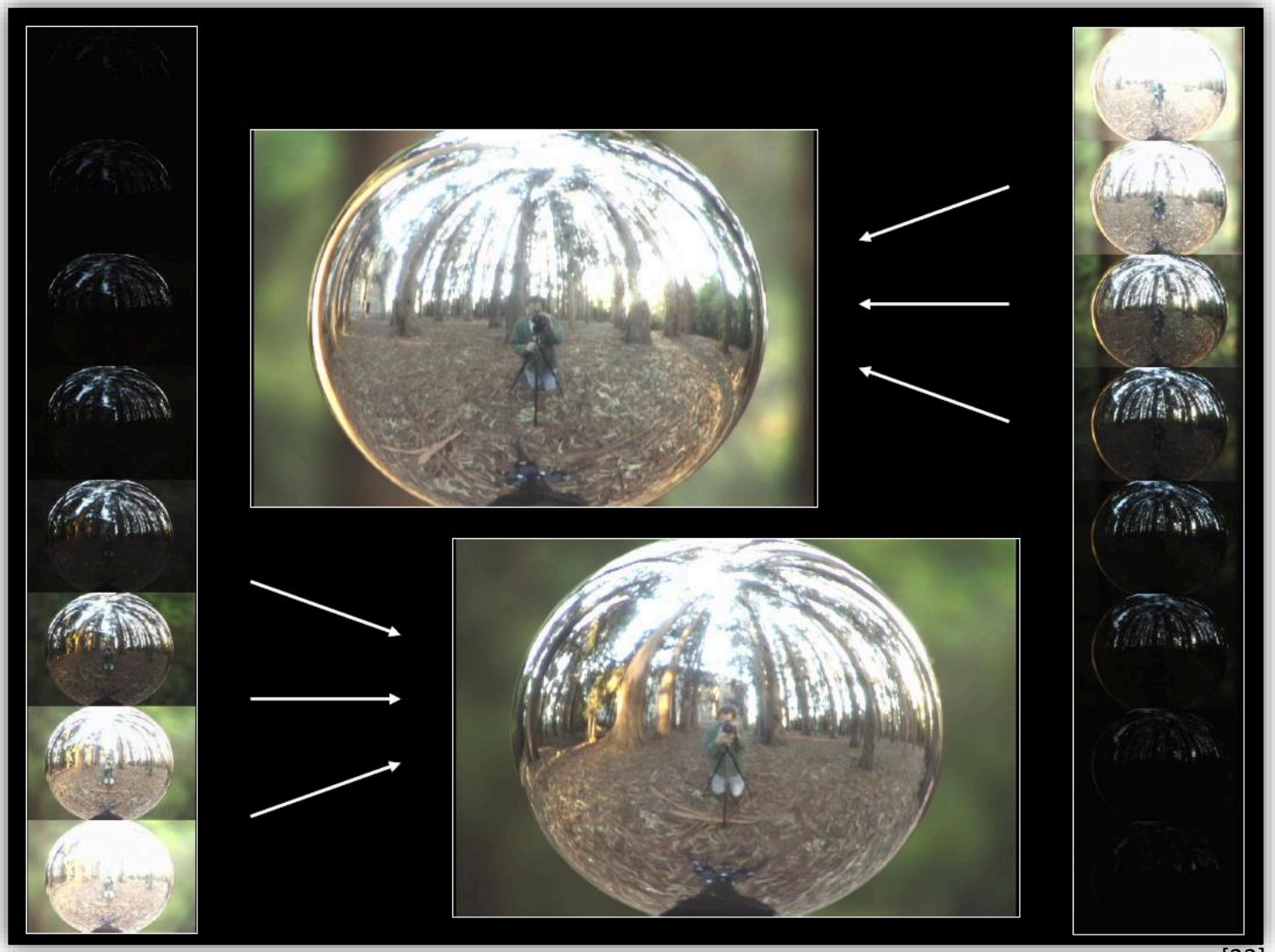


[32]

Aufnahme von Environment Maps

Aufnahme mit Belichtungsserien (HDR-Fotografie)

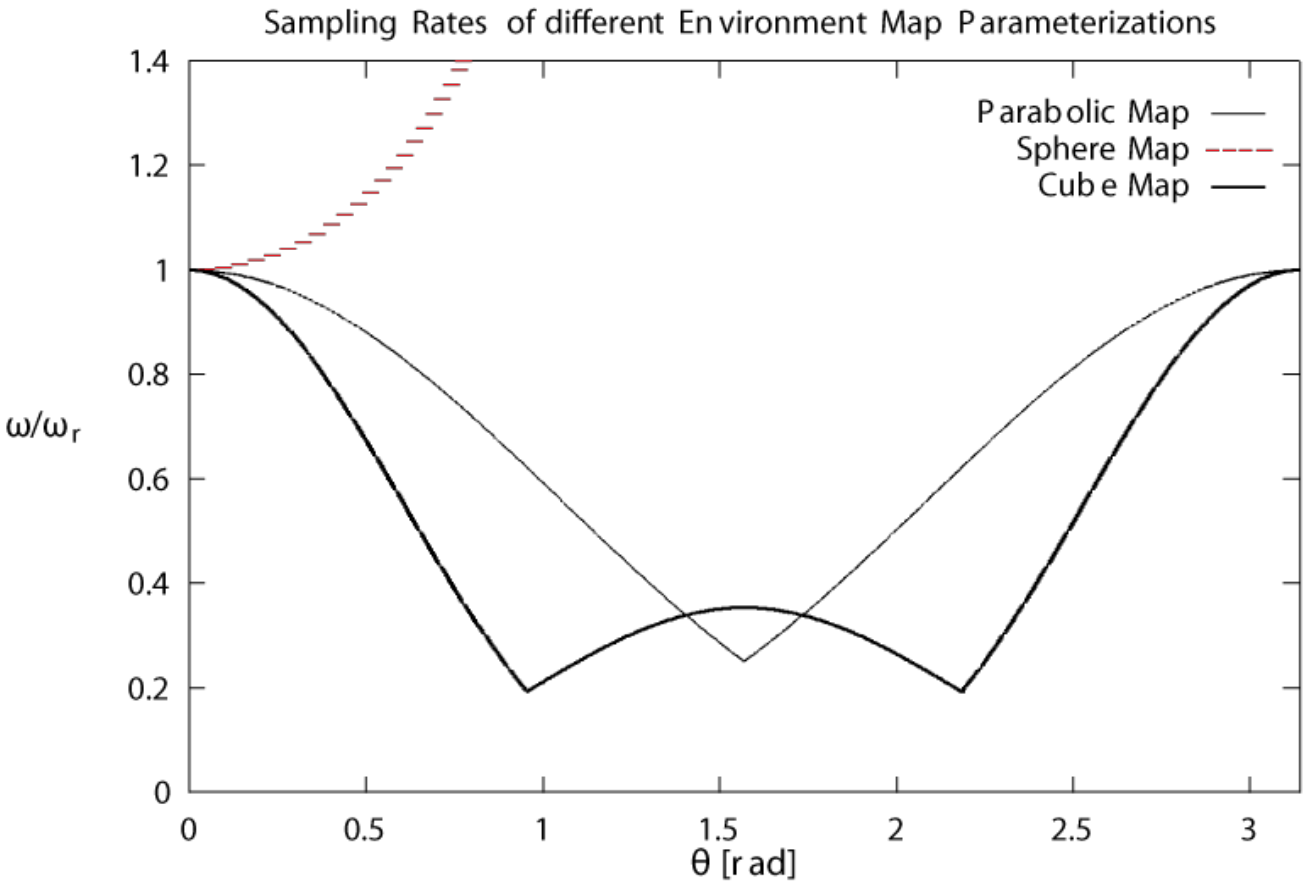
- … zur Rekonstruktion eines größeren Dynamikumfangs



[32]

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 $\mathbf{v_0}$



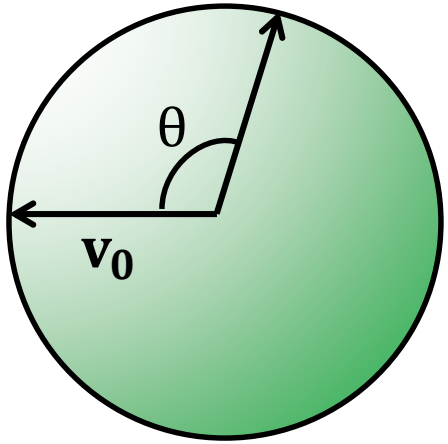
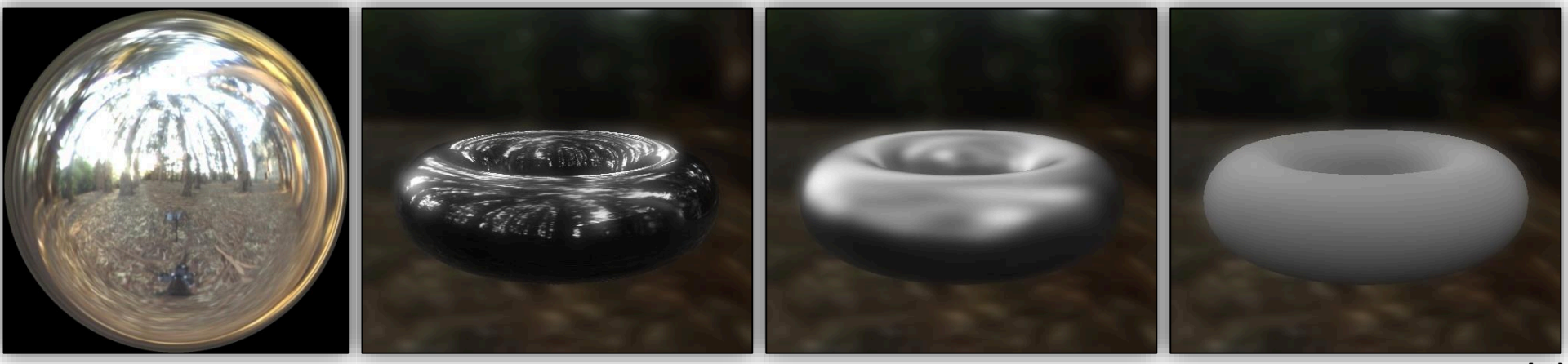


Image-based Lighting: Environment Maps…

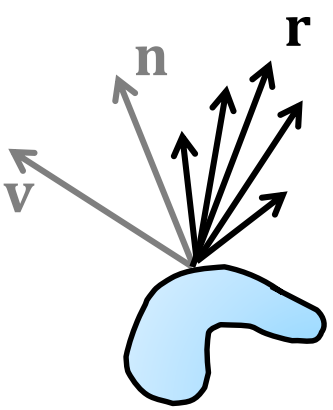
…für imperfekte Spiegelungen und diffuse Beleuchtung

- allg. verwenden von Beleuchtungsinformation in Texturen/EnvMaps
- Anwendung in der Vorlesung: Darstellung mit verschiedenen Materialien unter Umgebungsbeleuchtung ohne hohen Aufwand



EnvMap von [32]

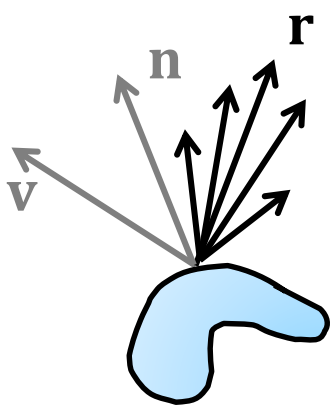
- Bsp. imperfekte Spiegelung:
  - i.W. trägt Licht um $\mathbf{r}$ zur Reflexion zum Betrachter bei (Phong-Modell: $I_{\mathrm{L}} \cdot (\mathbf{r} \cdot \mathbf{l})^n$)
  - mehrfaches Abtasten der Textur (analog zu Distributed Raytracing) ist teuer



Vorfilterung von Environment Maps

Grundidee: Vorfilterung (Prefiltered EnvMaps)

- Beobachtung: bei imperfekter Spiegelung trägt v.a. Licht aus einem Bereich um $\mathbf{r}$ zum reflektierten Licht bei (z.B. „Kosinuskeule“ $(\mathbf{r} \cdot \mathbf{l})^n$)
- statt mehrfachem Abtasten wird die Textur **vorgefiltert**
- einmaliges Auslesen liefert (gewichtete) Summe über einen Winkelbereich
- beachte:
  Größe und Form des „richtigen“ Winkelbereichs (und auch die Gewichtung der Texel) hängt vom Material (BRDF) und der Parametrisierung ab

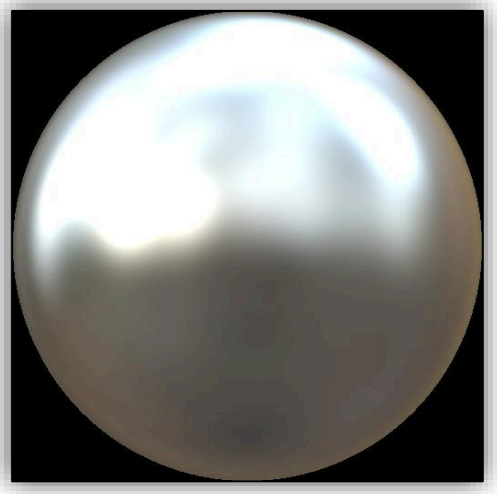


[32]

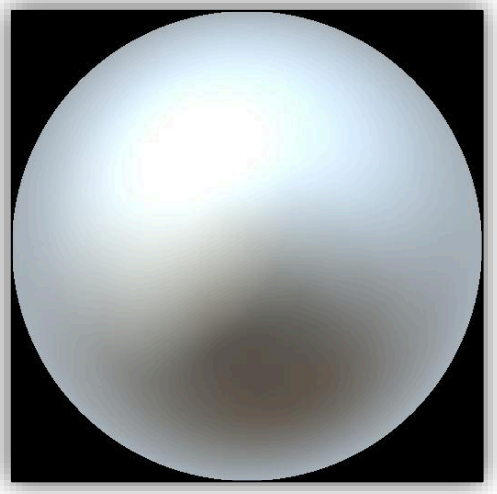
Vorfilterung von Environment Maps

- eine 2D-Textur kann nur ein 2D-Signal (z.B. eine Richtung) repräsentieren
  - Reflexionsrichtung: imperfekte Reflexion, einfallendes Licht aus Richtungen $\mathbf{d}$ gewichtet analog zu spekularem Phong-Term $(\mathbf{r} \cdot \mathbf{d})^n$
  - 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 $\mathbf{r}$)



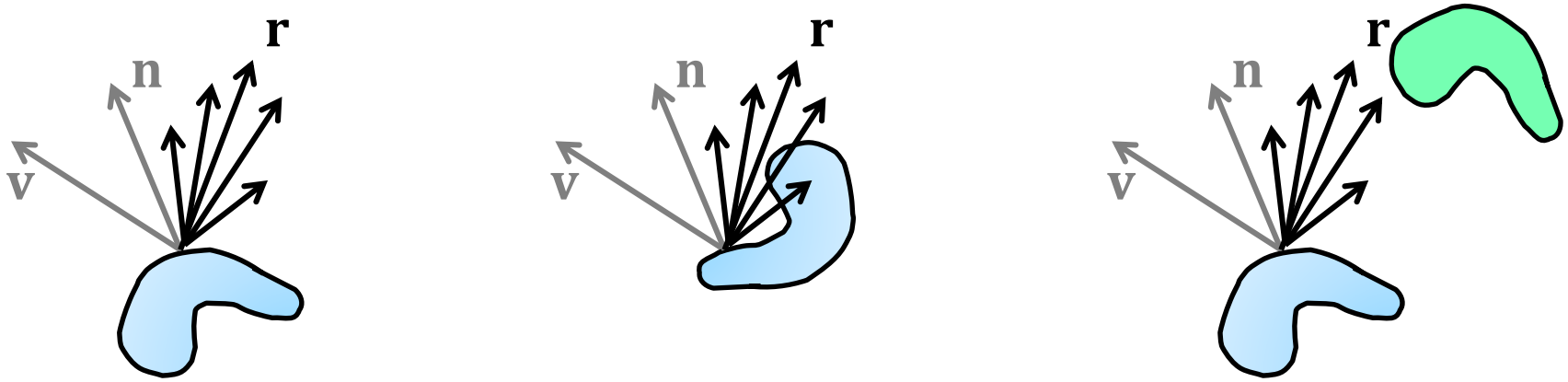
vorgefiltert für imperfekte Spiegelung (Zugriff mit $\mathbf{r}$, für einen bestimmten Phong-Exp.)



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

(Selbst-)Verschattung

- korrekte Verschattung ist nicht vereinbar mit vorgefilterten EnvMaps
- wenn Effizienz wichtig ist, dann wird Verschattung durch das Objekt selbst (oder durch andere Objekte) – wenn überhaupt – nur anhand von einem od. wenigen Schattenstrahlen, z.B. um Richtung $\mathbf{r}$, entschieden
- als Notlösung wird oft (vorberechnetes) Ambient Occlusion verwendet



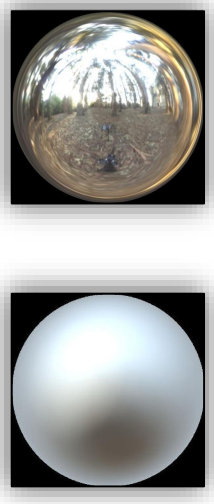
- mit vereinfachender Annahme in der Form:

$$\frac{2\pi}{N}\sum\left(L_i(\boldsymbol{\omega}_i)V_{\mathbf{x}}(\boldsymbol{\omega}_i)\right) \approx \left(\frac{2\pi}{N}\sum L_i(\boldsymbol{\omega}_i)\right)\cdot\left(\frac{2\pi}{N}\sum V_{\mathbf{x}}(\boldsymbol{\omega}_i)\right)$$

- Sichtbarkeitsfkt. $V_{\mathbf{x}}(\boldsymbol{\omega}) = \begin{cases} 0 & \text{Strahl } \mathbf{x} + t\boldsymbol{\omega} \text{ trifft Geometrie} \\ 1 & \text{sonst} \end{cases}$

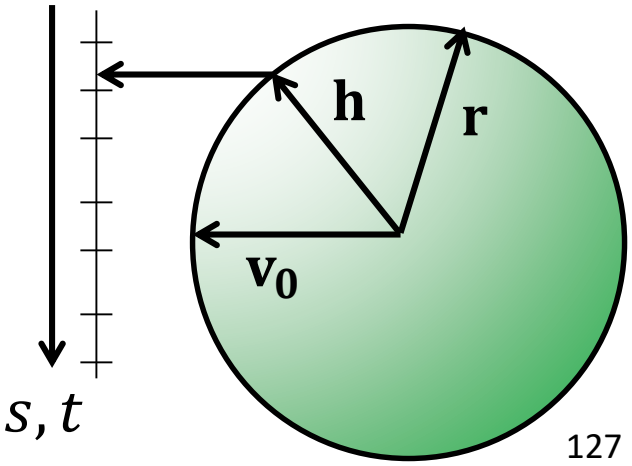
Vorfilterung von Environment Maps

Bsp. Vorfilterung einer Sphere Map für diffuse Beleuchtung



- verwende die (im Prinzip beliebige EnvMap-)Parametrisierung um Texel ↔ Richtungen zuzuordnen
  - Eingabe: Funktion $L(\mathbf{r})$ (einfallendes Licht) in einer Textur
  - Ausgabe: in der Beleuchtungstextur wollen wir für Flächen mit Normalen $\mathbf{n}$ das diffus reflektierte Licht speichern

- Hilfsfunktion: Abbildung von (s,t) auf eine Richtung $\mathbf{r}$
  - auf der folgenden Folie: `vec3 getDirection( float s, float t )`
  - Abb. $\mathbf{r}$ auf Texturkoord.: $s = (h_x + 1)/2$ bzw. $t = (h_y + 1)/2$
  - $\Rightarrow h_x = 2s - 1,\ h_y = 2t - 1$ und $h_z^2 = 1 - h_x^2 - h_y^2$ (mit $\mathbf{v_0} = (0,0,-1)^T$)
  - (s,t) entspricht dann $\mathbf{v_0}$ gespiegelt an $\mathbf{n} = \mathbf{h}$, also $\mathbf{r} = 2(\mathbf{v_0} \cdot \mathbf{n})\mathbf{n} - \mathbf{v_0}$ mit $\mathbf{n} = \mathbf{h} = (2s - 1, 2t - 1, \sqrt{1 - \cdots})$



Vorfilterung von Environment Maps

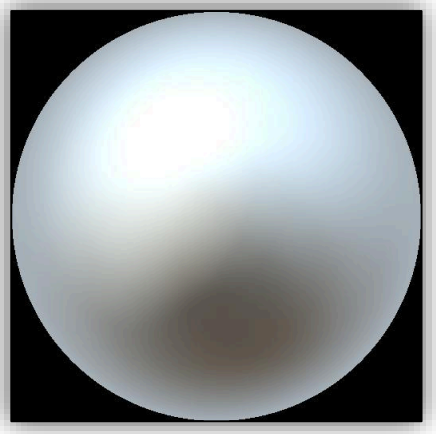
Prinzip Vorfilterung einer Sphere-Map für diffuse Beleuchtung

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

// für alle Orientierungen
// nicht gezeigt: ignorieren ungenutzter Texturbereiche
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

[32]

prefiltered

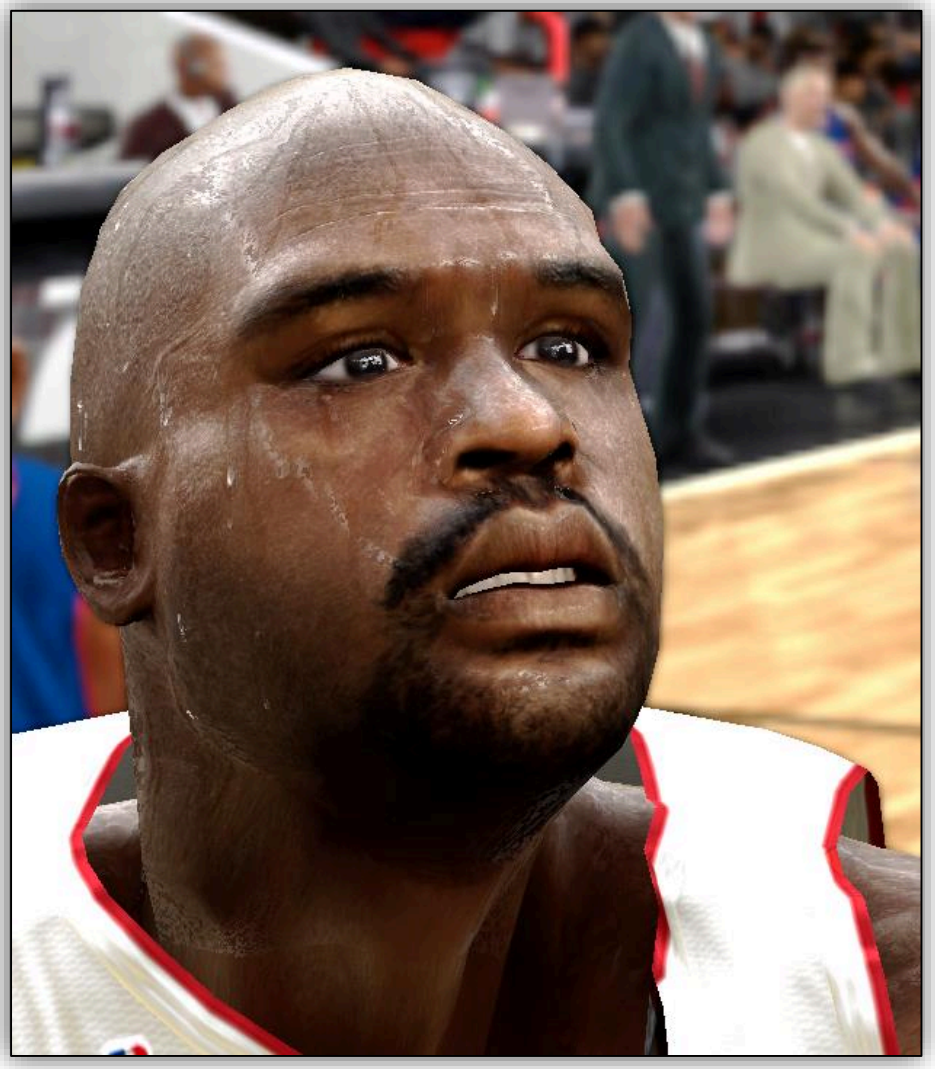


Das Ergebnis ist so nicht ganz korrekt:
die Abtastung der Lichtrichtungen ist nicht uniform
und dies müsste kompensiert werden (Raumwinkel)

Environment Mapping in Spielen

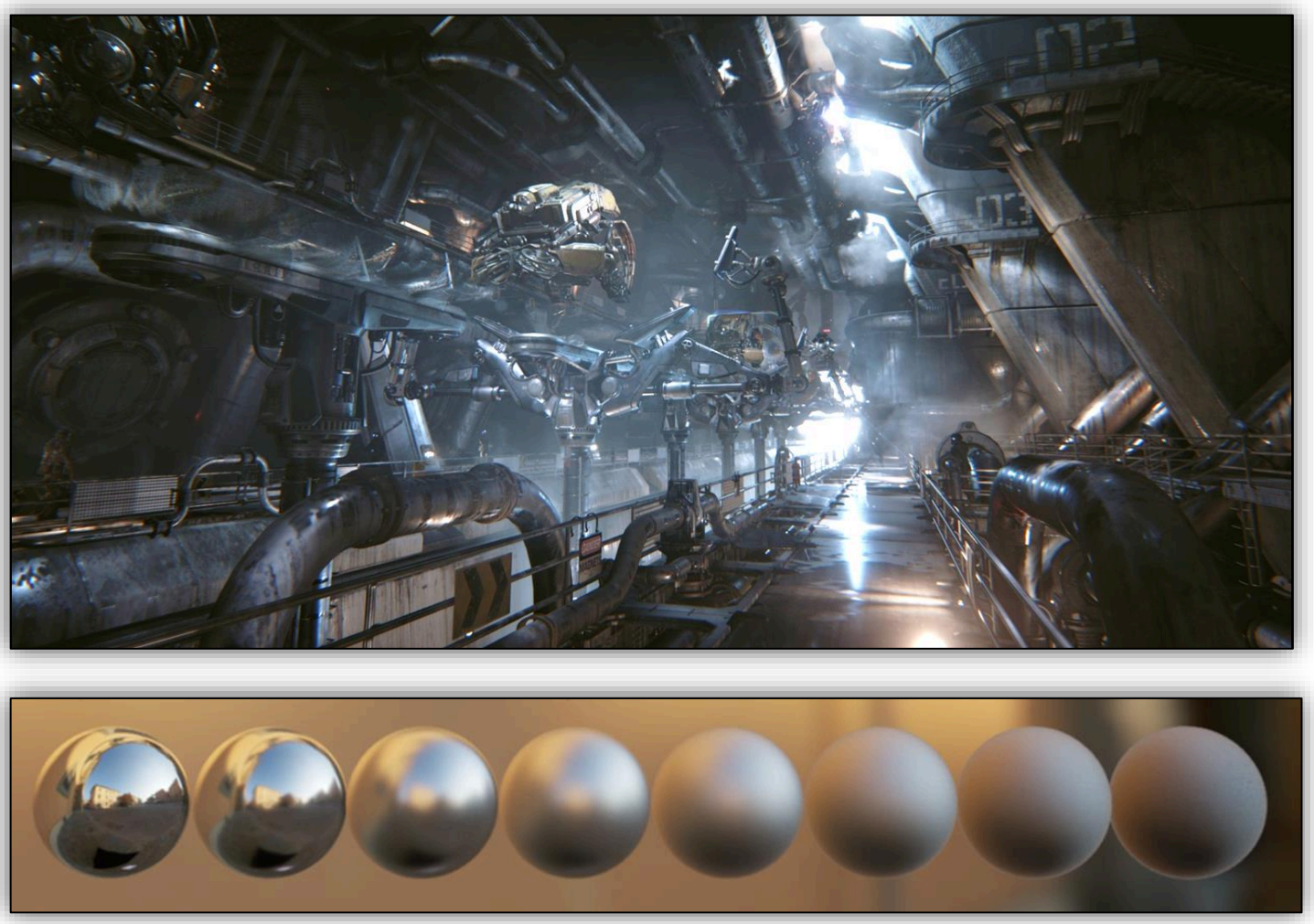
Beispiel aus der Echtzeitgrafik: NBA2k7 – Visual Concepts

- grobe Approximation: Vorfilterung durch Mip-Mapping statt Faltung



Environment Mapping in Spielen

Beispiel aus Unreal Engine 4 Infiltrator Demo



Details: http://blog.selfshadow.com/publications/s2013-shading-course/ [35]

Environment Mapping in Spielen

Beispiel aus Unreal Engine 4 Infiltrator Demo

Details: http://blog.selfshadow.com/publications/s2013-shading-course/ [36]

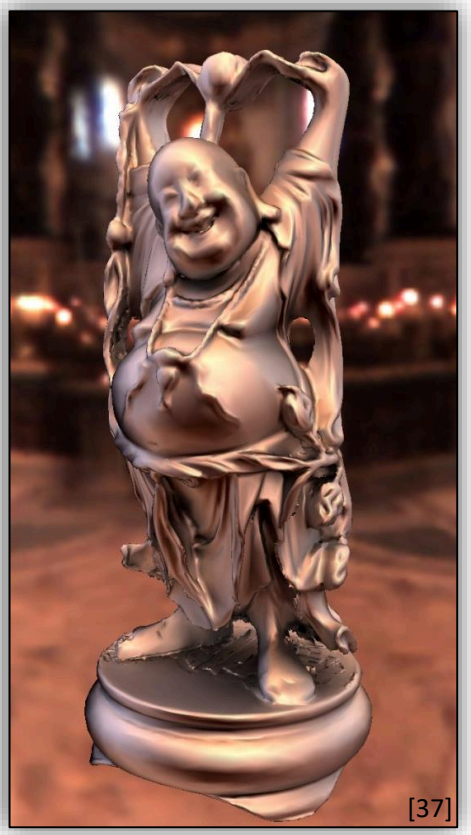
Precomputed Radiance Transfer

- Vorberechnung des Lichttransports auf den Oberflächen:
  „wie setzt sich das reflektierte Licht aus dem einkommenden zusammen“
- → Vorlesung Interaktive Computergrafik im Sommersemester

diffus

diffus, verschattet
mit Interreflexion

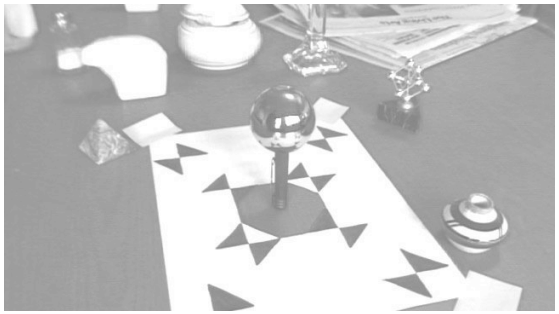
glossy



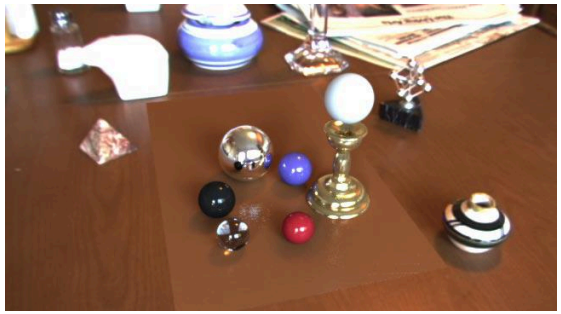
glossy, verschattet
mit Interreflexion

Anwendungen Image-based Lighting

Fotografie

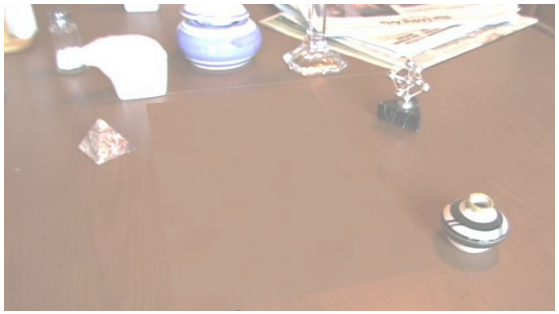


Light Probe und Kalibrierungsmuster



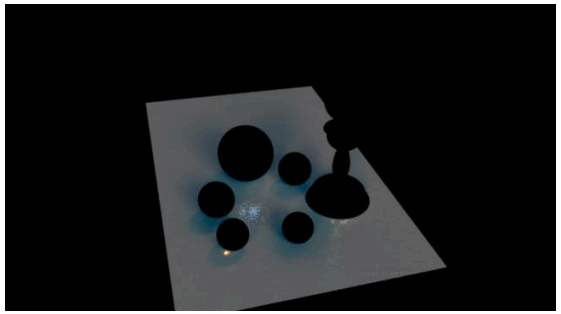
synthetische Objekte + Ebene

–

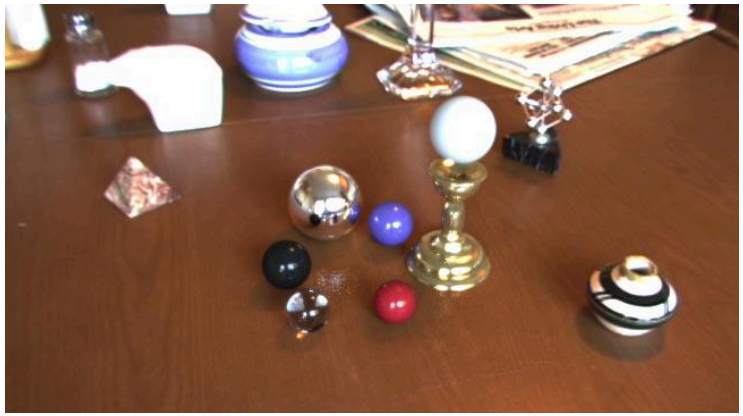


nur Ebene

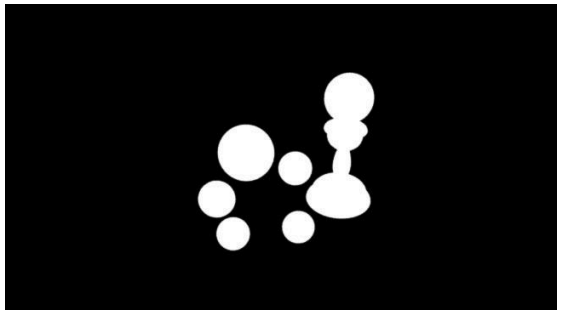
=



Differenz (nur Ebene)



fertiges Bild



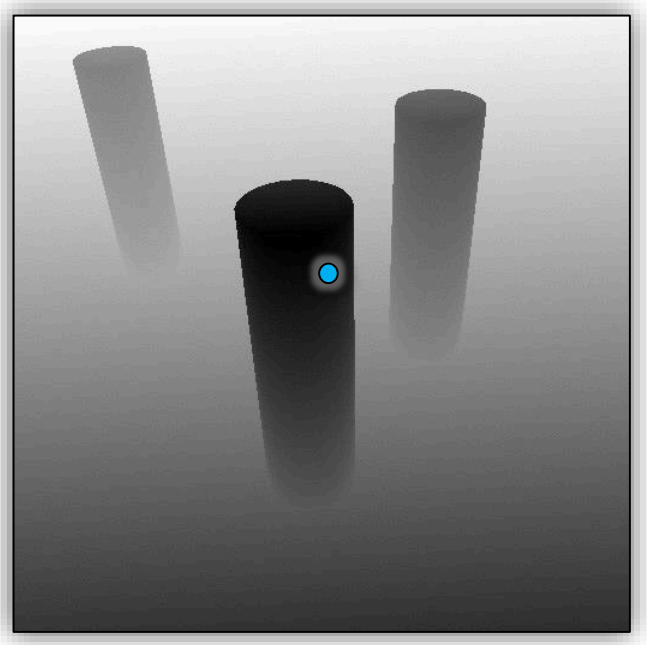
Objektmaske

[39]

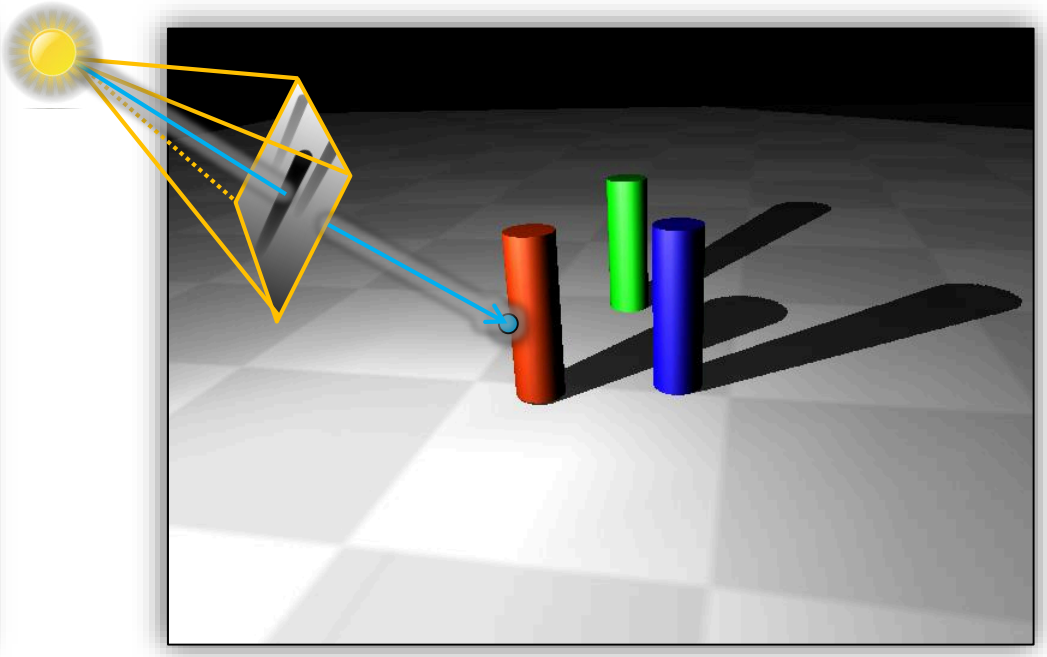
Shadow Mapping (Williams 1978)

Shadow Mapping Grundidee

- eine Textur speichert – für viele Richtungen – wie weit die Oberflächen von der Lichtquelle entfernt sind
- → im OpenGL Kapitel…



Shadow Map



Texture Mapping

Fazit

- Texture Mapping ist eine fundamentale Technik in der Computergrafik
- erlaubt Detailreichtum, der mit reiner Geometrie nicht möglich wäre
- vielseitiger Einsatz
  - Farb-, Gloss-, Ambient Occlusion-Texturen und Atlanten
  - Normal und Displacement Mapping
  - Environment Mapping und Image-Based Lighting
  - Shadow Mapping
  - Visualisierung
  - Lookup-Tabellen für aufwändige Berechnungen (z.B. auch bei vorgefilterten Environment Maps mit aufwändigeren BRDFs)
  - …

Bildreferenzen

- [1] Texture Mapping & Other Fun Stuff MIT EECS 6. 837, Durand Cutler
- [2] P. Shirley, S. Marschner, 3rd Edition, AK Peters
- [3] Edward Angel, Dave Shreiner. 2011. Interactive Computer Graphics: A Top-Down Approach With Shader-Based Opengl. 7th Edition. Pearson
- [4] Realistic Human Face Rendering for "The Matrix Reloaded“, George Borshukov, SIGGRAPH 2003
- [5] https://www.textures.com/download/BuildingsBarn0003/12172
- [6] Random-access rendering of general vector graphics, D. Nehab und H. Hoppe, SIGGRAPH Asia, 27(5), 2008
- [7] Rendering Surface Details with Diffusion Curves, Stefan Jeschke, David Cline, Peter Wonka, 2009
- [8] https://stackoverflow.com/questions/23583853/lighting-vbo-using-gllightfv-and-glmaterialfv
- [9] https://learn.foundry.com/modo/14.2/content/help/pages/shading_lighting/shader_items/projection_type_samples.html
- [10] https://de.wikipedia.org/wiki/Datei:Sphericalcoordinates.svg
- [11] https://slidetodoc.com/visual-appearance-shading-texture-mapping-course-note-credit/
- [12] Fundamentals of Texture Mapping and Image Warping, Paul Heckbert, Master's thesis, UCB/CSD 89/516, CS Division, U.C. Berkeley, June 1989
- [13] https://www.heise.de/newsticker/meldung/20-Jahre-Unreal-Tournament-Als-Shooter-Meilenstein-geboren-von-Fortnite-gekillt-4594107.html
- [14] http://http.developer.nvidia.com/GPUGems2/

Bildreferenzen

- [15] https://www.wikiwand.com/es/Interpolaci%C3%B3n_multivariable
- [16] Compressed Random-Access Trees for Spatially Coherent Data, Lefebvre, Hoppe, 2007
- [17] Improved Alpha-Tested Magnification for Vector Textures and Special Effects, Greene, 2007
- [18] Texturing & Procedural Methods, Hendrik Lensch, Computer Graphics WS07/08
- [19] https://education.siggraph.org/static/HyperGraph/mapping/r_wolfe/r_wolfe_mapping_9.htm
- [20] http://theempireoflights.blogspot.com/2012/11/
- [21] https://www.gettyimages.de/detail/foto/kolmanskop-ghost-town-lizenzfreies-bild/985903030
- [22] https://unigine.com/
- [23] http://manual.reallusion.com/iclone_6/enu/pro_6.0/15_Tessellation_Displacement/Using_Vector_Displacement_Maps.htm
- [24] https://www.gamedeveloper.com/programming/hardware-accelerating-art-production
- [25] http://de.wikipedia.org/wiki/Alphakanal
- [26] http://www.klaasnienhuis.nl/2011/09/3d-wood-texture-by-jerry-ylilammi-berconmaps/
- [27] http://www-sop.inria.fr/reves/Basilic/2008/DLTD08/
- [28] Advances in Real-time Rendering 2015, HORIZON: ZERO DAWN, SIGGRAPH
- [29] Visualization in Connectomics Hanspeter Pfister, Verena Kaynig, Charl P. Botha, Stefan Bruckner, Vincent J. Dercksen, Hans-Christian Hege, and Jos B.T.M. Roerdink, arXiv 2012
- [30] https://de.wikipedia.org/wiki/Nervenzelle
- [31] Blinn, J. F. and Newell, M. E. Texture and reflection in computer generated images Communications of the ACM Vol. 19, No. 10 (October 1976), 542-547

Bildreferenzen

- [32] www.debevec.org
- [33] https://de.m.wikipedia.org/wiki/Datei:Eglise_icone.svg
- [34] https://www.cg.tuwien.ac.at/courses/Realtime/slides/11Shading_2016.pdf
- [35] http://blog.selfshadow.com/publications/s2013-shading-course/
- [36] http://blog.selfshadow.com/publications/s2013-shading-course/
- [37] https://zhuanlan.zhihu.com/p/92893911
- [38] https://360rumors.com/insta360-pro-2-review/
- [39] https://www.uni-koblenz.de/~cg/veranst/ss03/cgseminar_vortraege/EinfuehrungAugment.pdf
- [40] https://slideplayer.com/slide/13992546/
- [41] http://romain.vergne.free.fr/teaching/IS/SI05-textures.html
- [42] http://romain.vergne.free.fr/teaching/IS/SI05-textures.html