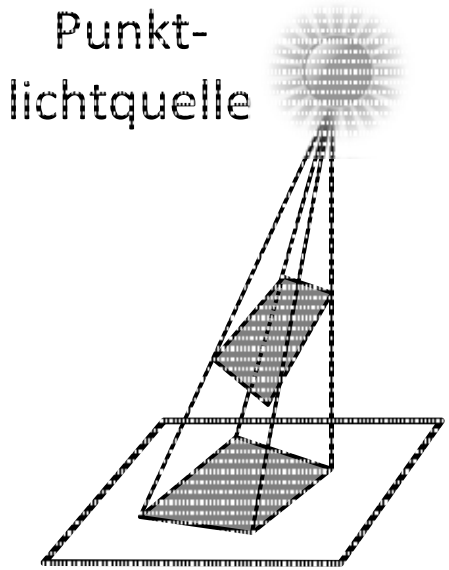
Interaktive Computergrafik

Vorlesung im Sommersemester 2014
Kapitel 2: (Echtzeit-)Schattenverfahren (Teil 2)

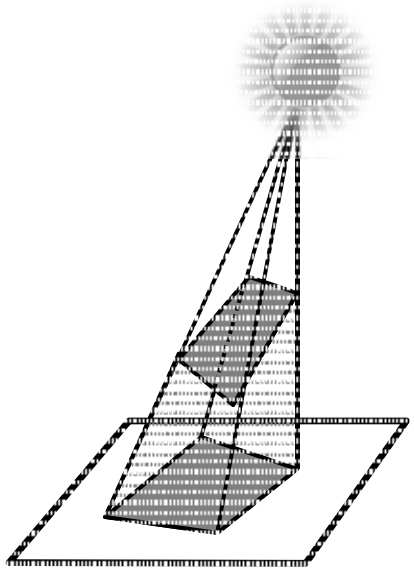
Prof. Dr.-Ing. Carsten Dachsbacher
Lehrstuhl für Computergrafik
Karlsruher Institut für Technologie

Klassifikation Schattenverfahren

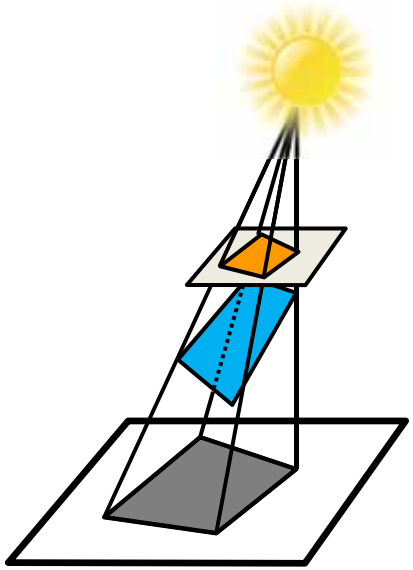
- **vorberechnete Schatten/Beleuchtung** („Light Maps“)
- **projektive Schatten**: nur für Schatten auf Ebenen, keine Selbstverschattung
- **Schattenvolumen**: Objektraum-Verfahren
- **Shadow Maps**: Bildraum-Verfahren



Projektive Schatten



Schattenvolumen

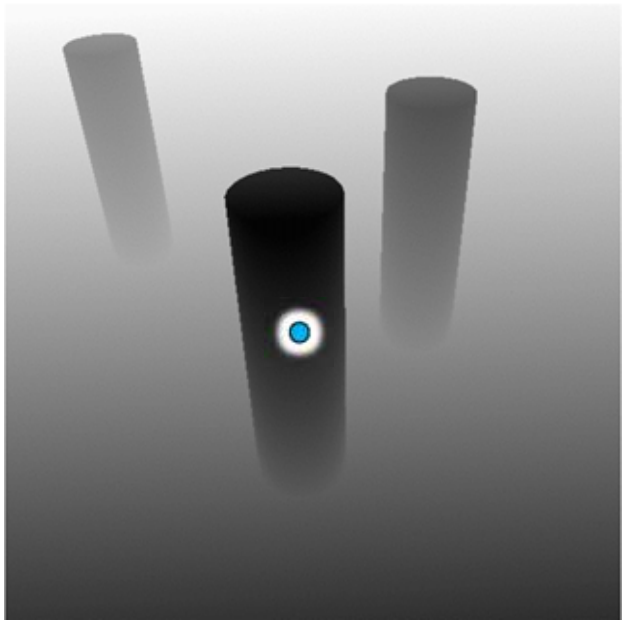


Shadow Maps

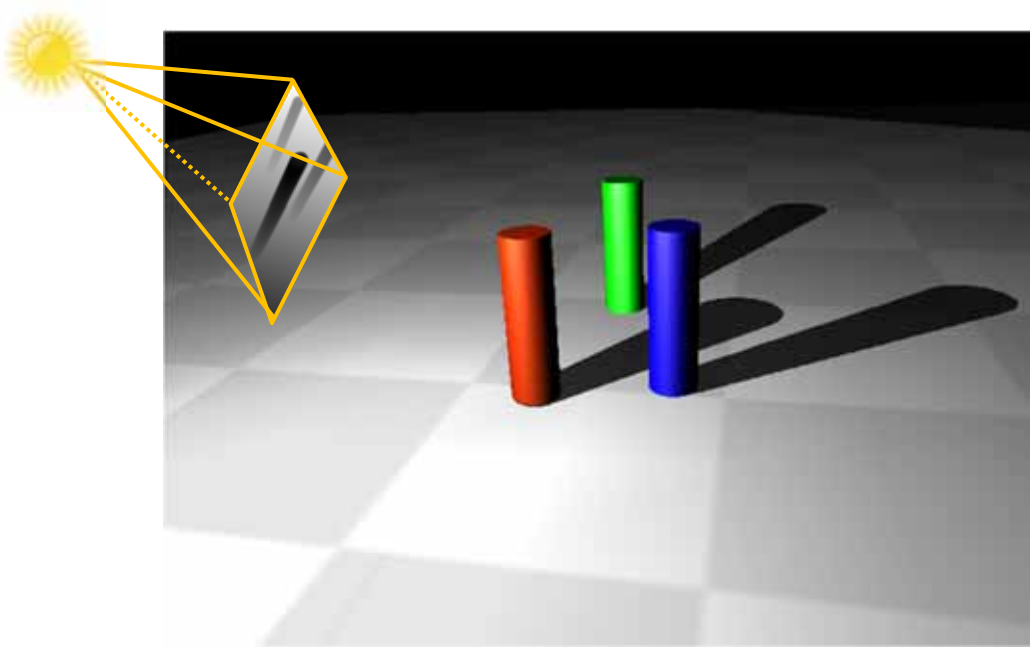
Shadow Mapping (Williams, SIGGRAPH‘78)

Shadow Mapping Grundidee

- eine Textur speichert – für viele Augstrahlen/Richtungen – wie weit die nahsten Oberflächen von der Lichtquelle entfernt sind
- wir betrachten zunächst Lichtquellen, deren Leuchtkegel ein Pyramidenstumpf ist, d.h. wir ersetzen die Lichtquelle gedanklich durch eine perspektivische Kamera



Shadow Map: Szene aus der Sicht der Lichtquelle

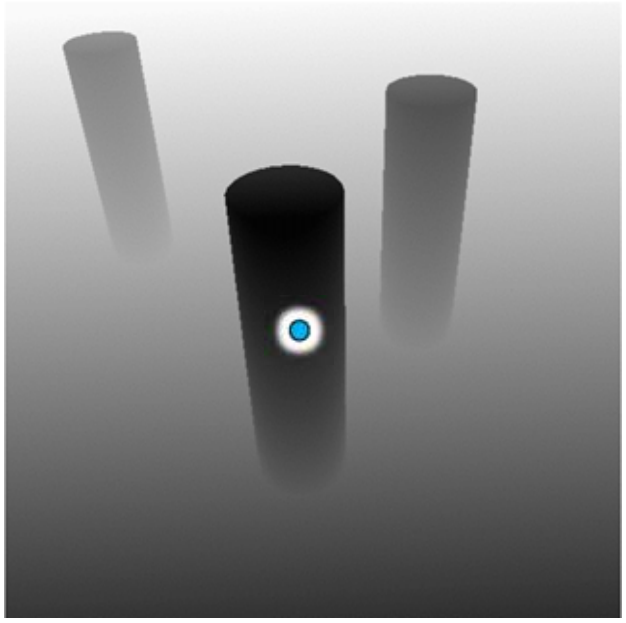


Szene aus der Sicht der Kamera

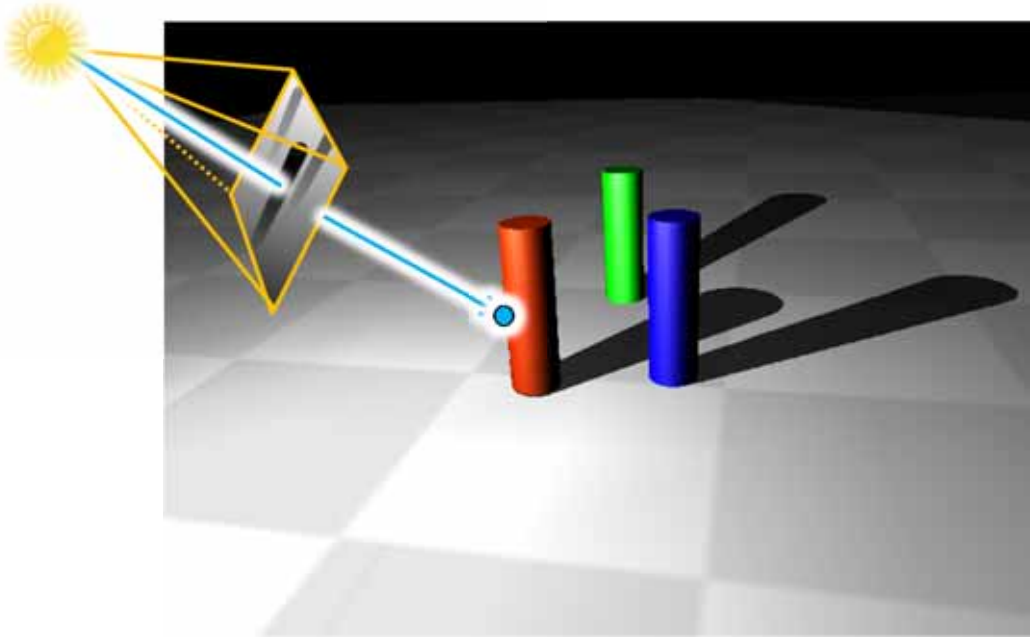
Shadow Mapping (Williams, SIGGRAPH‘78)

Shadow Mapping Grundidee

- eine Textur speichert – für viele Augstrahlen/Richtungen – wie weit die nahsten Oberflächen von der Lichtquelle entfernt sind
- beim Rendering des Kamerabildes:
  - jeder Punkt im Raum entspricht einer Richtung von der LQ
  - für diese Richtung lässt sich die Entfernung zur nahsten Oberfläche nachschlagen



Shadow Map: Szene aus der Sicht der Lichtquelle

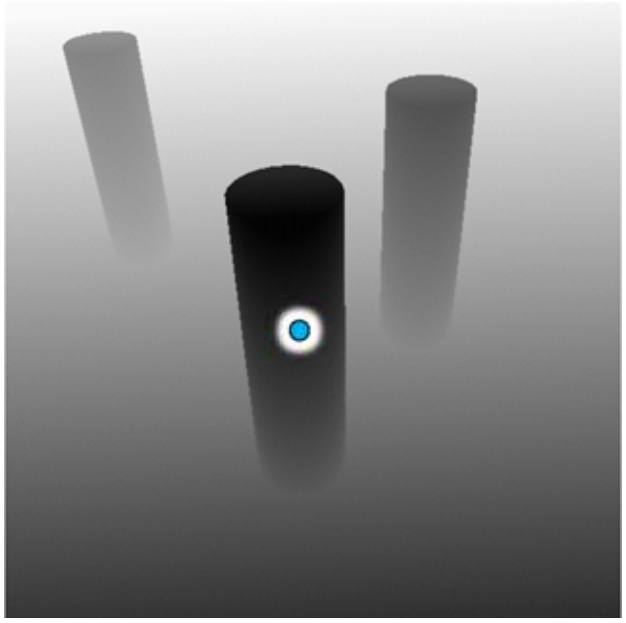


Szene aus der Sicht der Kamera

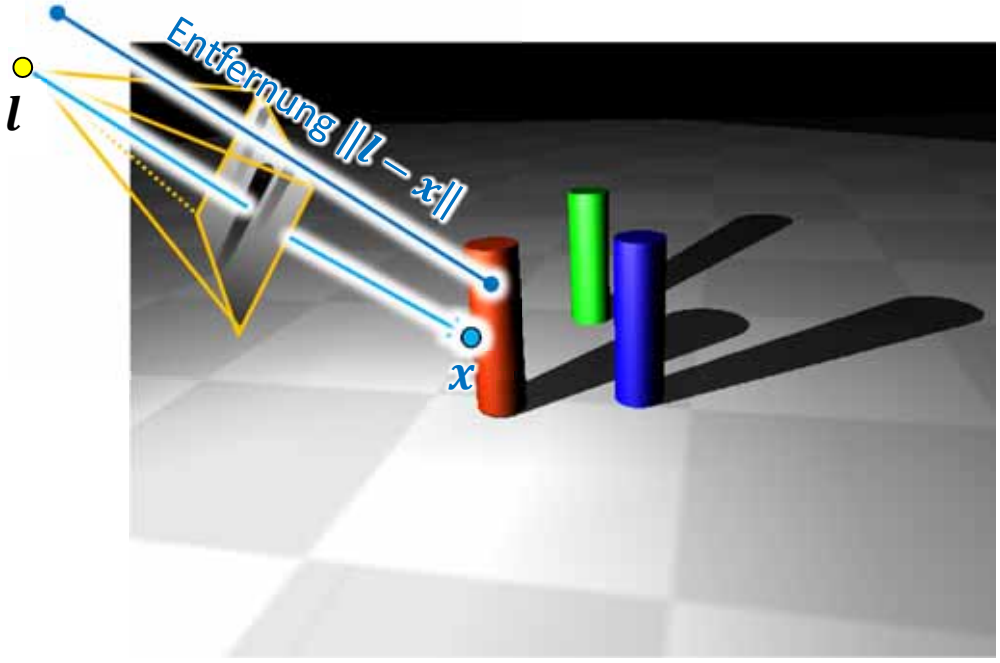
Shadow Mapping (Williams, SIGGRAPH‘78)

Shadow Mapping Grundidee

- der blaue Punkt befindet sich nicht im Schatten
  - die in der Shadow Map gespeicherte Entfernung entspricht der tatsächlichen Entfernung



Shadow Map: Szene aus der Sicht der Lichtquelle

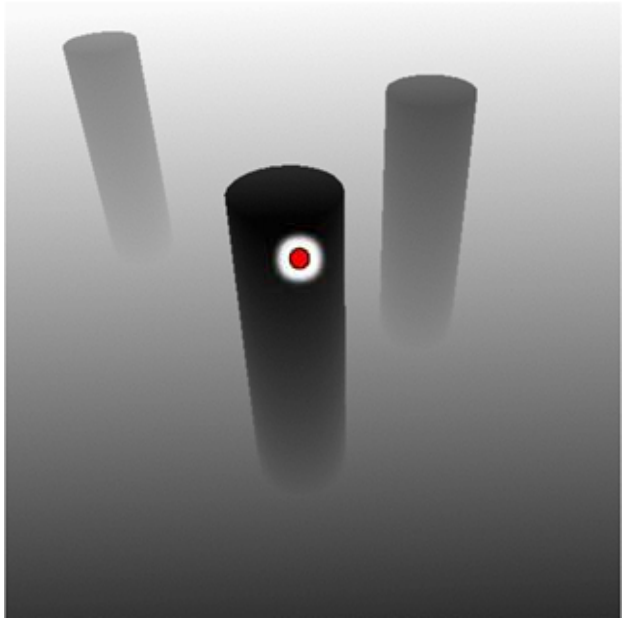


Szene aus der Sicht der Kamera

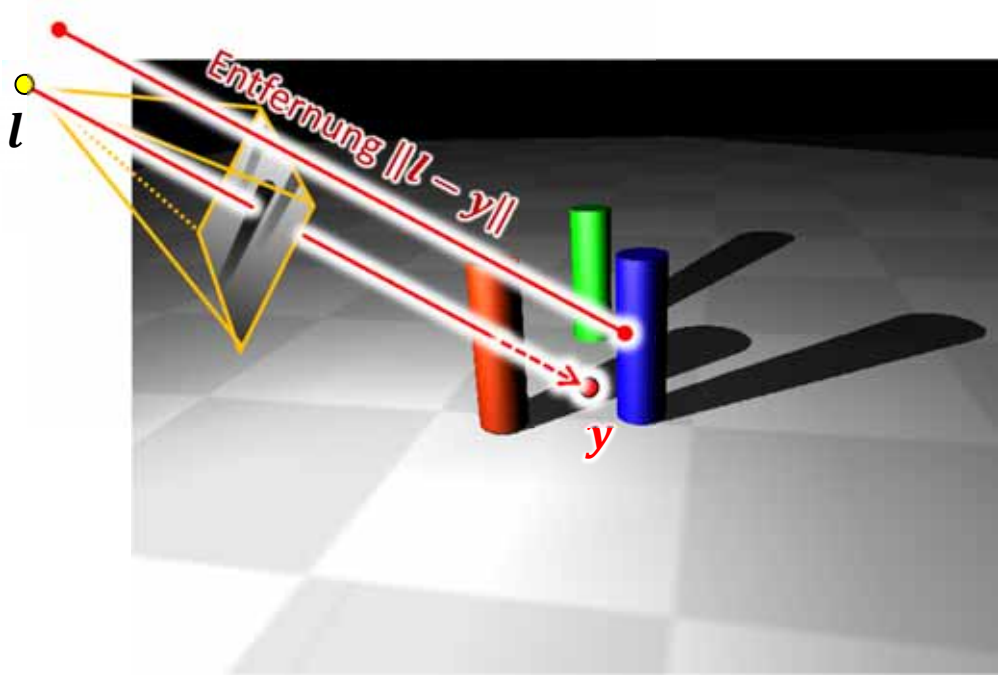
Shadow Mapping (Williams, SIGGRAPH‘78)

Shadow Mapping Grundidee

- der rote Punkt befindet sich im Schatten
  - die in der Shadow Map gespeicherte Entfernung ist kleiner als seine tatsächliche Entfernung zur Lichtquelle



Shadow Map: Szene aus der Sicht der Lichtquelle

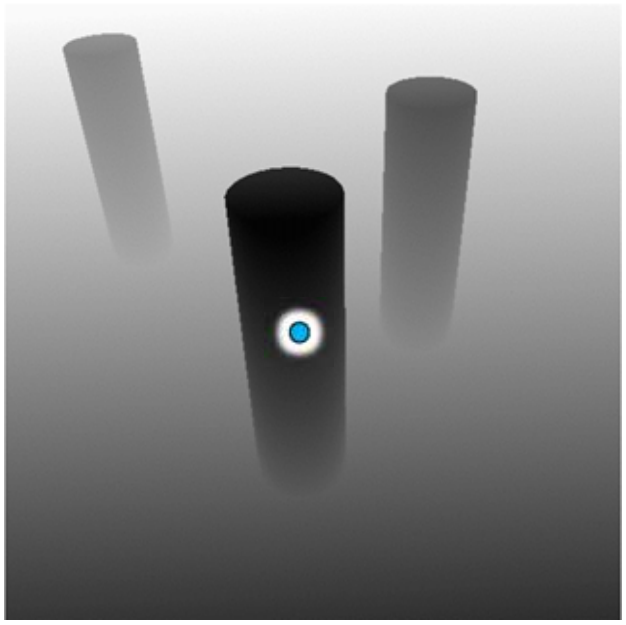


Szene aus der Sicht der Kamera

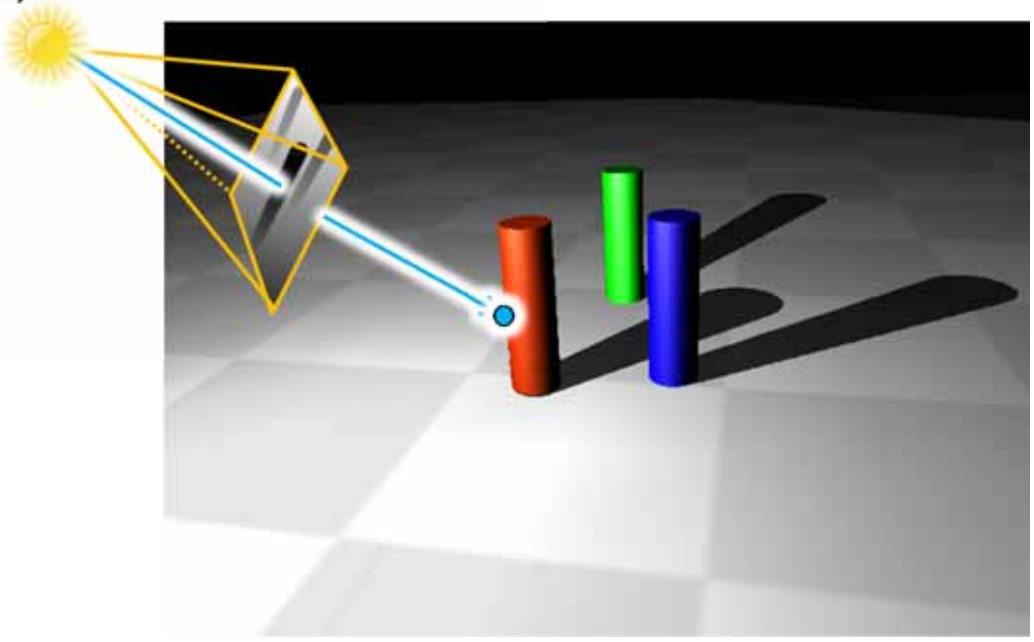
Shadow Mapping (Williams, SIGGRAPH‘78)

Shadow Mapping Grundidee

- erzeuge **pro Lichtquelle eine Shadow Map** → verwende sie während dem Rendering des Kamerabildes für den Schattentest
- es handelt sich um ein **Bildraum-Verfahren**: durch Rendering der Szene aus Sicht der LQ erhalten wir eine diskrete Abtastung der Flächen
- Fragestellungen: omnidirektionale LQ, Artefakte aufgrund der Abtastung, Auflösung der Shadow Map, …



Shadow Map: Szene aus der Sicht der Lichtquelle



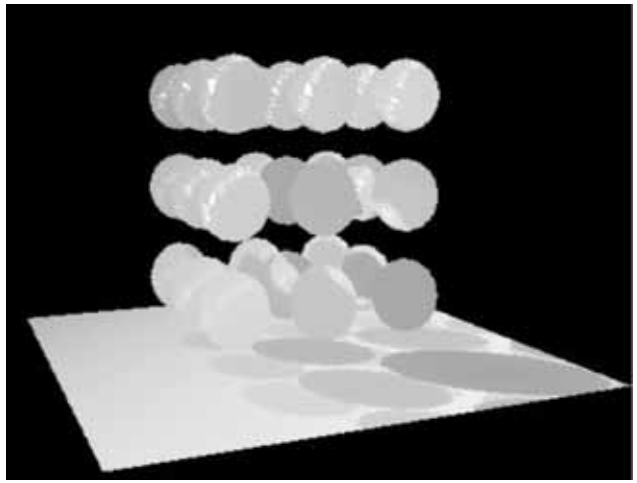
Szene aus der Sicht der Kamera

Shadow Mapping und Koordinatensysteme

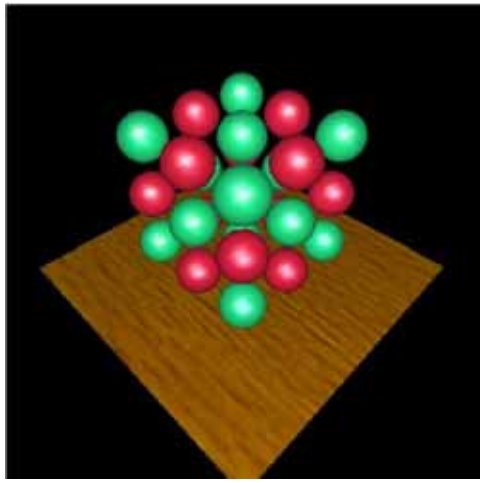
Shadow Map

projektives
Texture Mapping

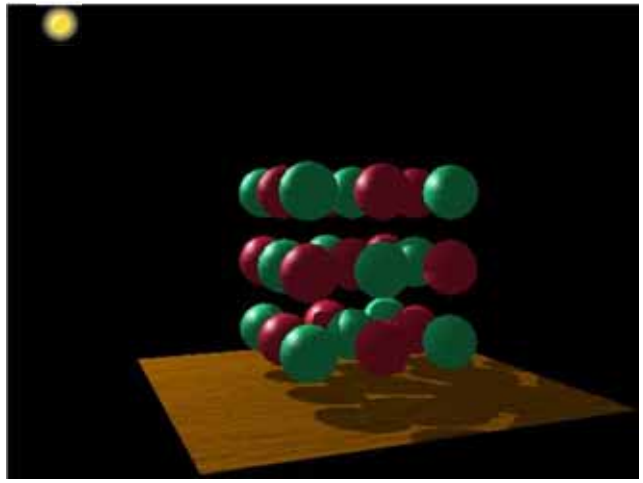
Shadow Map projiziert auf die Szene



Schattentest



Sicht der Lichtquelle
(nur zur Illustration)



Sicht der Kamera

Rendering der Shadow Map

- wie erzeugt man die Shadow Map? wie speichert man die Entfernungen?
- Möglichkeit 1: spezielles Texturformat **GL_DEPTH_COMPONENT**
  - erzeugen einer Tiefen-Textur:

```
GLuint shadowMapTexture;
glGenTextures( 1, &shadowMapTexture );

glBindTexture( GL_TEXTURE_2D, shadowMapTexture );
// OpenGL muss man das Texturformat mitteilen
glTexImage2D( GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT,
              width, height, 0, GL_DEPTH_COMPONENT,
              GL_UNSIGNED_BYTE, NULL );

glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP );
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP );
```

- vollständiges Tutorial:
  http://www.paulsprojects.net/tutorials/smt/smt.html

Rendering der Shadow Map

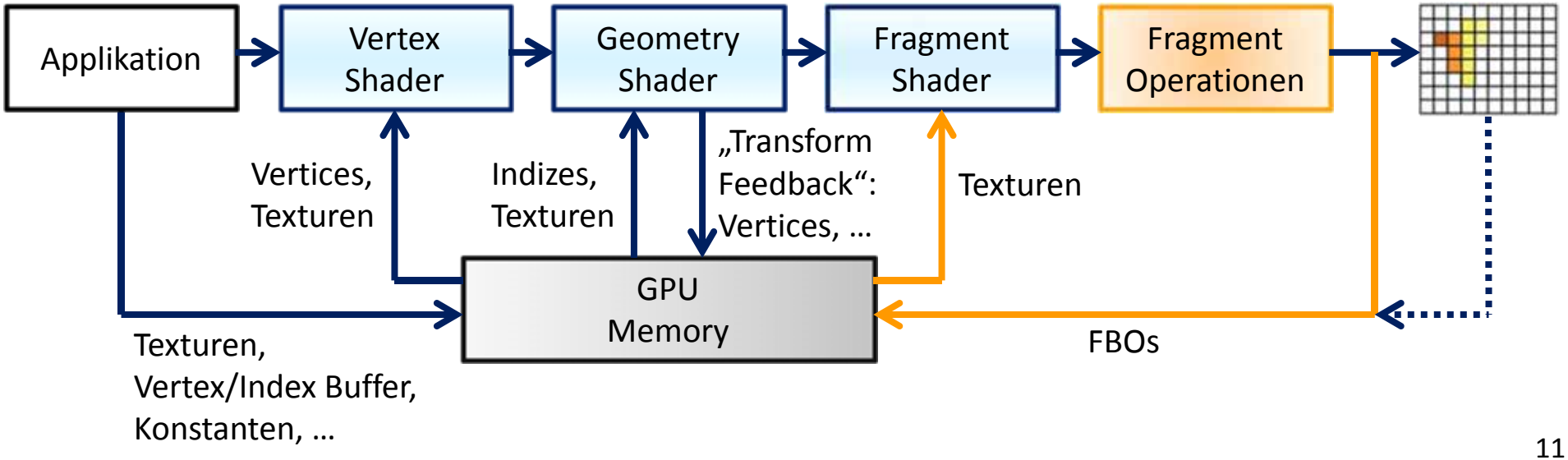
- Rendering der Szene und Kopieren des Tiefenpuffers in die Shadow Map:

```
glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

// Setzen der Matrizen (alternativ Shader Uniforms)
glMatrixMode ( GL_PROJECTION );
glLoadMatrixf( lightProjectionMatrix );
glMatrixMode ( GL_MODELVIEW );
glLoadMatrixf( lightViewMatrix );

// setze den Viewport auf die Größe der Shadow Map
glViewport( 0, 0, width, height );

// Schreiben in den Framebuffer deaktivieren
glColorMask( 0, 0, 0, 0 );

drawScene();

// Zurücklesen des Tiefenpuffers in die Shadow Map
// wird automatisch wg. des Texturformats gewählt
glBindTexture( GL_TEXTURE_2D, shadowMapTexture );
glCopyTexSubImage2D( GL_TEXTURE_2D, 0, 0, 0, 0, 0,
                     shadowMapSize, shadowMapSize );
```

ein sog. Depth-Only Pass, den GPUs besonders schnell durchführen können

Rendering der Shadow Map

- Möglichkeit 2: Frame Buffer Objects (FBOs) (opt. Floating Point Textur)
  - mit FBOs kann man eigene Frame Buffer und Tiefenpuffer erstellen, die anschließend als Texturen verwendet werden können
  - das Rendering erfolgt direkt in die Texturen
  - Vorteile: kein „Zurückkopieren“, beliebige Formate und Größen
    - z.B. ein eigener Frame Buffer mit Floating Point Genauigkeit
    - ohne FBOs kann eine Shadow Map nicht höher aufgelöst sein als der normale Frame Buffer aus dem zurückkopiert wird
  - keine Details an dieser Stelle (aber im nächsten Kapitel)



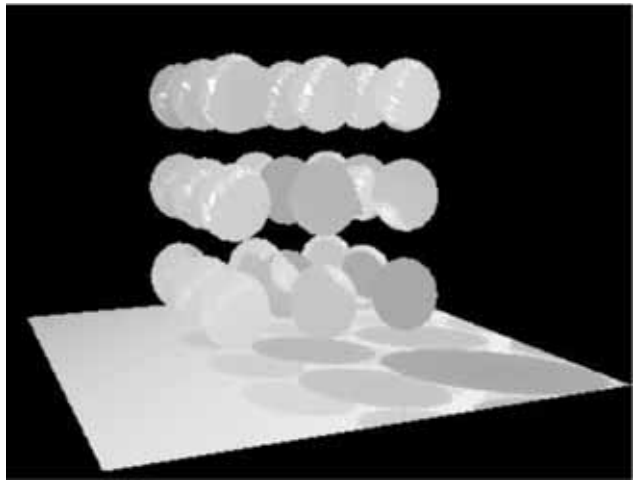
Shadow Mapping und Koordinatensysteme

- wenn wir die Shadow Map verwenden
  - finde für jeden rasterisierten Pixel die entsprechende Texturkoordinate in der Shadow Map heraus
  - oder umgekehrt: projiziere Shadow Map auf die Geometrie der Szene
  - generiere hierzu Texturkoordinaten pro Vertex, die für Fragmente interpoliert werden
- zur Erinnerung: zunächst Rendering der Shadow Map durch eine herkömmliche perspektivische Kamera

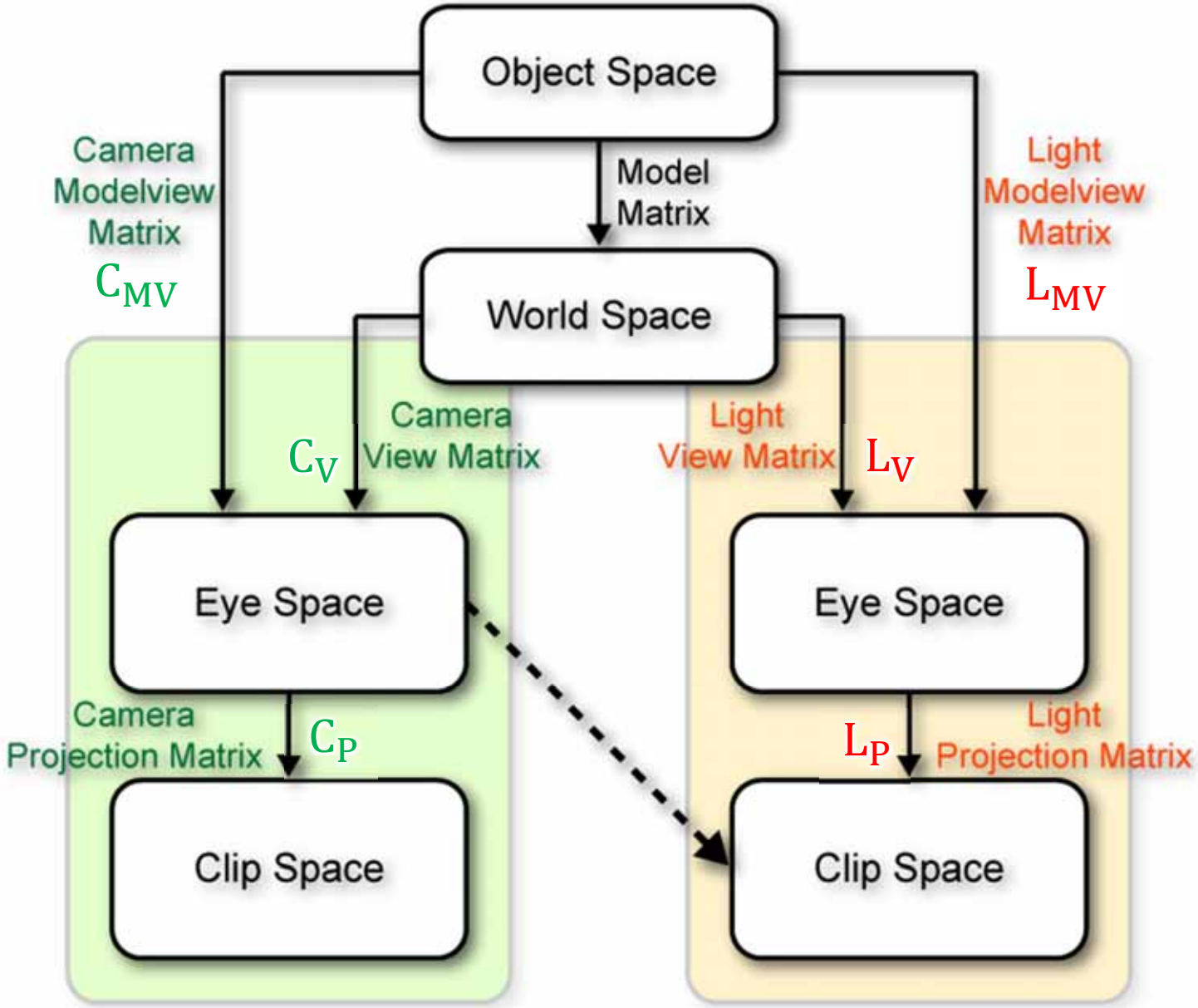
Shadow Map

projektives Texture Mapping

Shadow Map projiziert auf die Szene



Shadow Mapping und Koordinatensysteme



Shadow Mapping und Koordinatensysteme

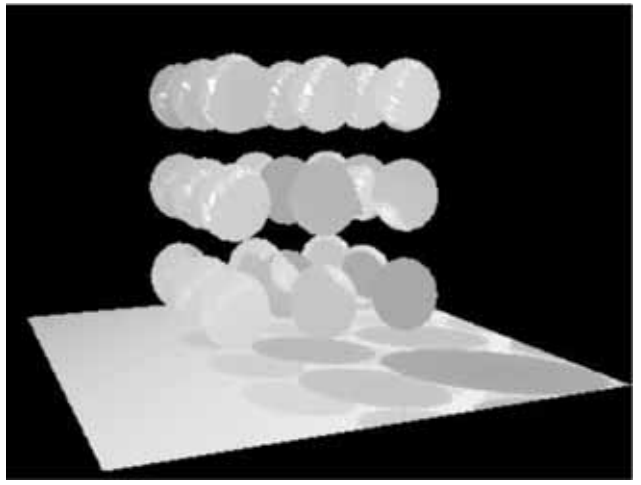
Abbilden von Vertizes in den Raum der Shadow Map

- „klassische Variante“: Texturkoordinaten-Generierung (`glTexGen`) aus Kamera-Koordinaten (in OpenGL sind alle Objekte nach der Modelview-Matrix erstmals in einem gemeinsamen Koordinatensystem)
  - Rendering der Shadow Map mit der Transformation $\mathrm{L_P L_{MV}}$
  - um aus Kamera-Koordinaten in den Clip Space der Shadow Map zu gelangen generieren wir Texturkoordinaten mit der Matrix $\mathrm{W L_P L_V C_V^{-1}}$ (W enthält die Viewport-/Window Transformation $[-1; 1] \rightarrow [0; 1]$)

Shadow Map

projektives Texture Mapping

Shadow Map projiziert auf die Szene



Shadow Mapping und Koordinatensysteme

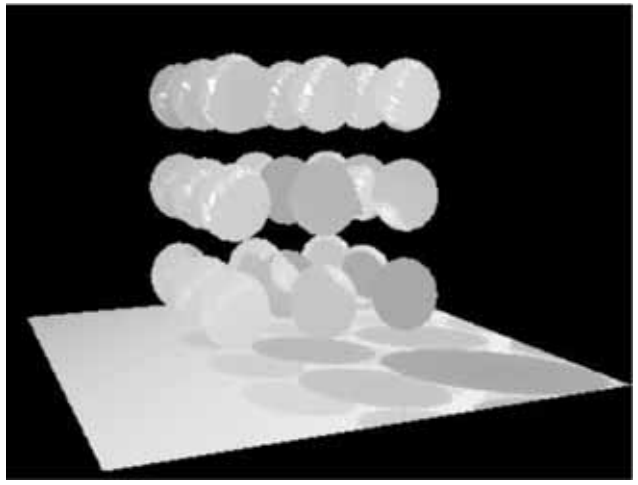
Abbilden von Vertizes in den Raum der Shadow Map

- eine weitere mögliche Variante mit GLSL
  - bei Beleuchtung in Weltkoordinaten, erhält man die Shadow Map Koordinaten nach der Transformation WL_PL_V
- im Prinzip kann man die Texturkoord.-Gen. auch pro Pixel durchführen
  - Berechnung pro Vertex und perspektivische Interpolation der Texturkoordinaten ist aber natürlich weniger aufwändig

Shadow Map

projektives Texture Mapping

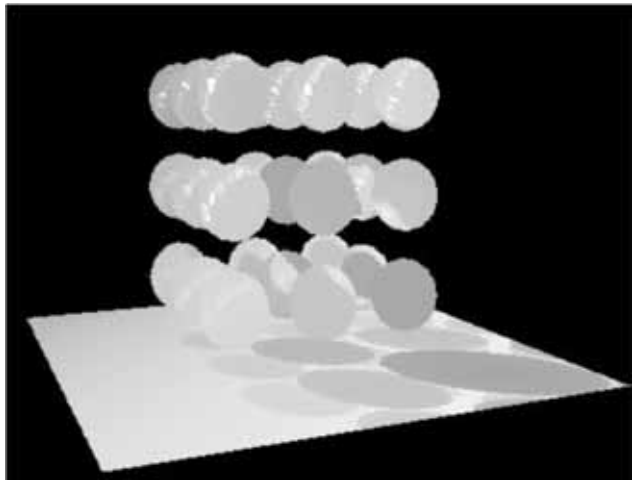
Shadow Map projiziert auf die Szene



Shadow Mapping und der Schattentest

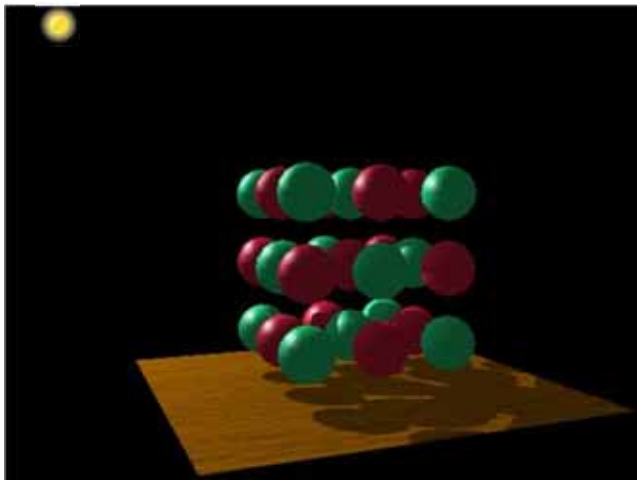
Nachschlagen in der Shadow Map und Schattentest

- der Schattentest ist im Prinzip nichts anderes als ein spezieller Texturierungsmodus von dem wir erhalten
  - 1 für beleuchtete Oberflächen
  - 0 für verschattete Oberflächen
- diese „Binärtextur“ wird verwendet um (diffuse und spekulare) Beleuchtungskomponenten zu modulieren



Shadow Map projiziert auf die Szene

Schattentest



Sicht der Kamera

Schattentest in OpenGL mit Tiefentexturen

- … hierzu muss eine Texturkoordinate (s, t, r) erzeugt werden, mit
  - (s, t) 2D-Koordinate für den Zugriff in der Textur
  - r Tiefenwert des Fragments, für das der Schattentest erfolgt
  - berechnet durch OpenGL Texturkoordinatengenerierung oder Vertex-Shader

- Konfiguration (klassisches OpenGL):

```
// Shadow Map aktivieren
glBindTexture( GL_TEXTURE_2D, shadowMapTexture );
glEnable( GL_TEXTURE_2D );

// Schattentest ist ein spezieller Texturierungsmodus
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_COMPARE_MODE,
                                GL_COMPARE_R_TO_TEXTURE );

// beim Schattentest wird r <= gespeicherte Tiefe überprüft
glTexParameteri( GL_TEXTURE_2D, GL_TEXTURE_COMPARE_FUNC,
                                GL_LEQUAL);
```

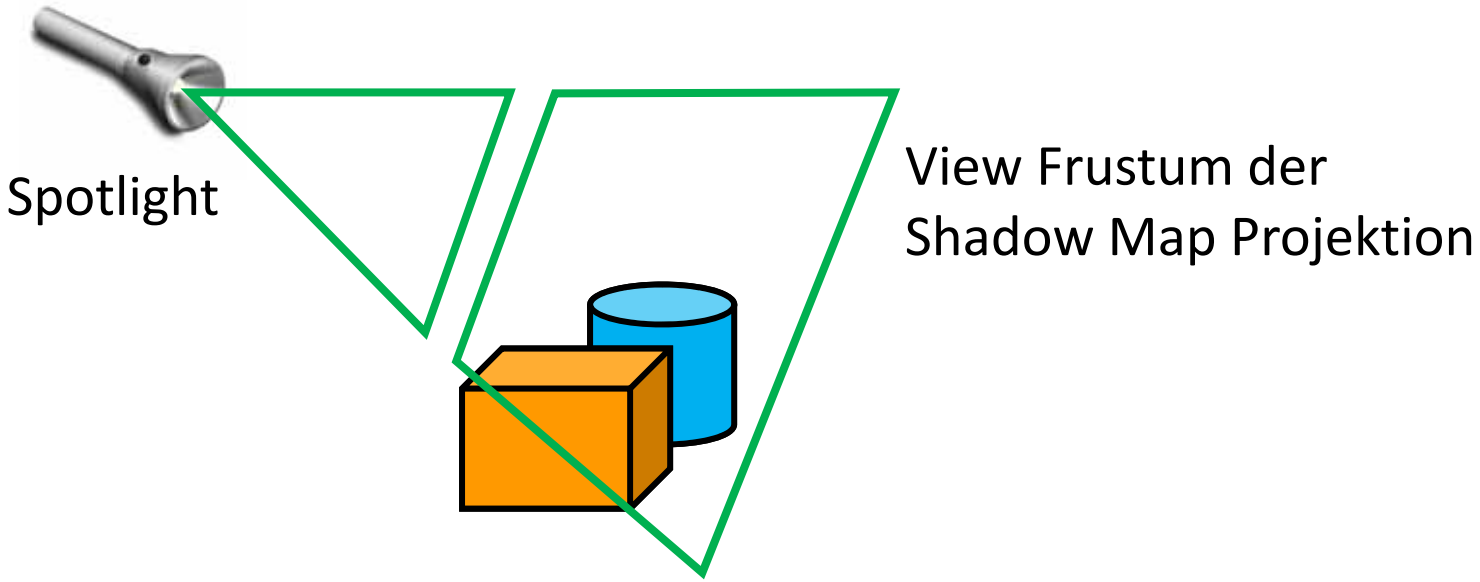
- Verwendung in einem GLSL-Shader

```
uniform sampler2D      tShadowMap;  …
float lit = shadow{1|2}D[Proj]( tShadowMap, smTexCoord );
```

Shadow Map: Lichtquellen

Spotlights/Scheinwerfer

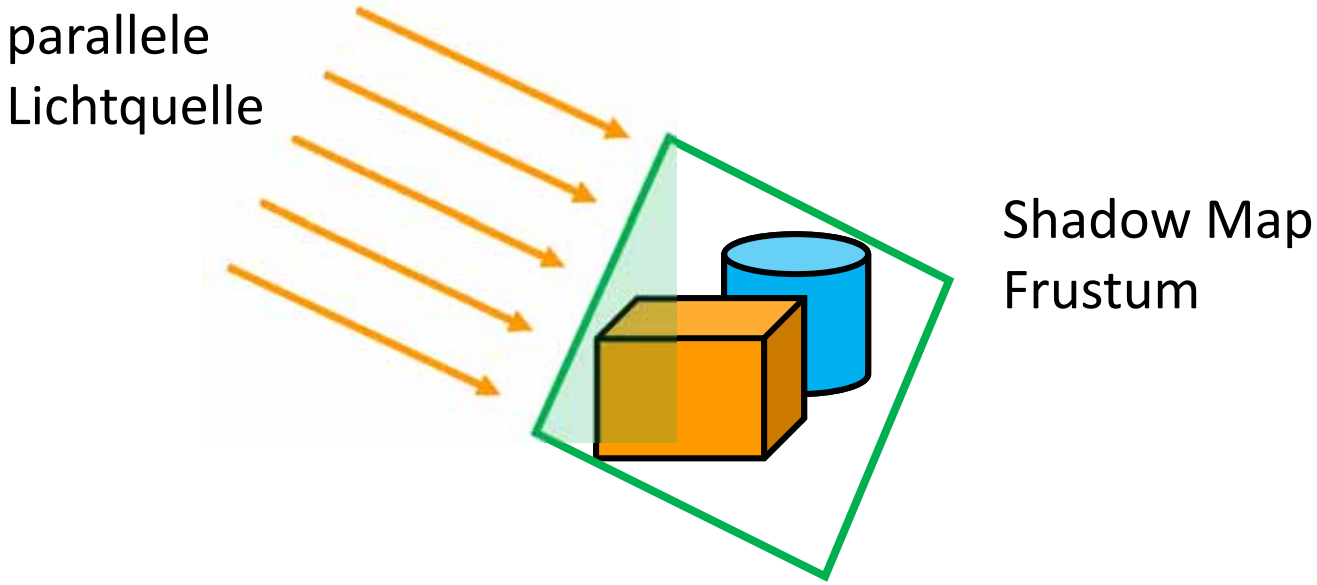
- verwenden eine perspektivische Kamera an der Position der Lichtquelle
- Blickrichtung und Öffnungswinkel gemäß der des Spotlights
- OpenGL Äquivalent: `gluLookAt(...)` und `gluPerspective(...)`
  - mit GLM: `glm::lookAt(...)` und `glm::perspective(...)`



Shadow Map: Lichtquellen

Parallele Lichtquelle

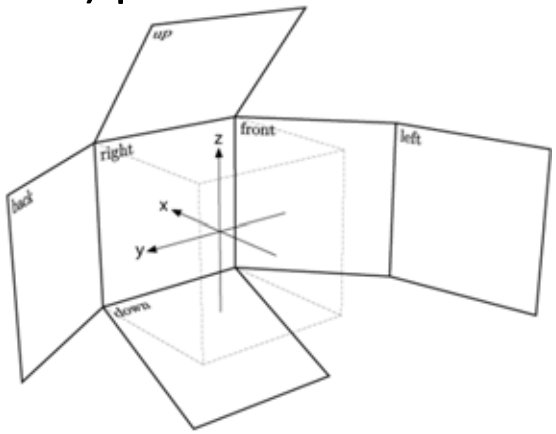
- Projektionsmatrix entspricht einer orthogonalen Projektion
- gesamte Szene muss im Frustum enthalten sein, d.h. **sichtbare und schattenwerfende Objekte**
  - Auflösungsproblem bei räumlich großen Szenen



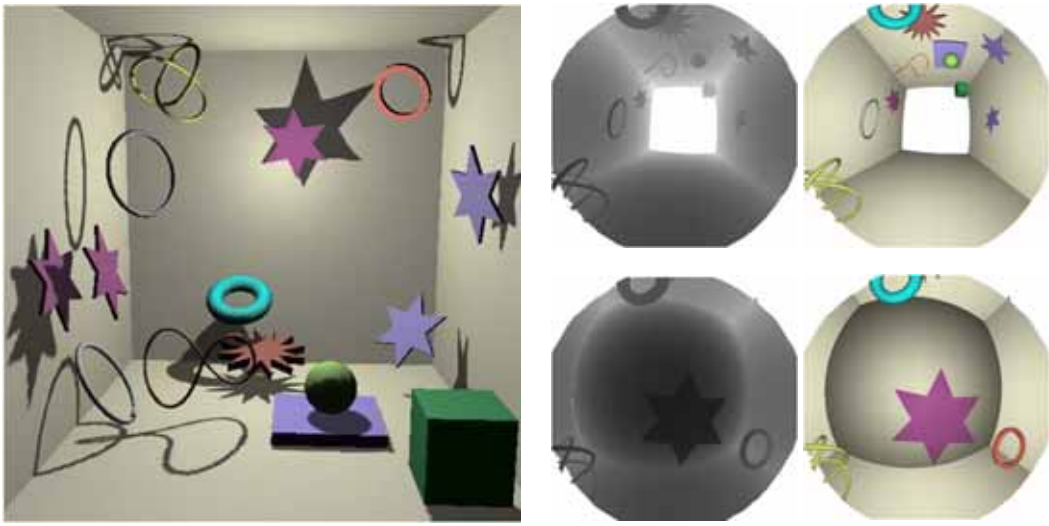
Shadow Map: Lichtquellen

(Omnidirektionale) Punktlichtquellen

- Cube Maps: eine perspektivische Kamera (FOV 90°) pro Würfelseite
  - Anm. mit Geometry Shader ist es möglich in einem Rendering-Durchgang alle 6 Seiten zu zeichnen



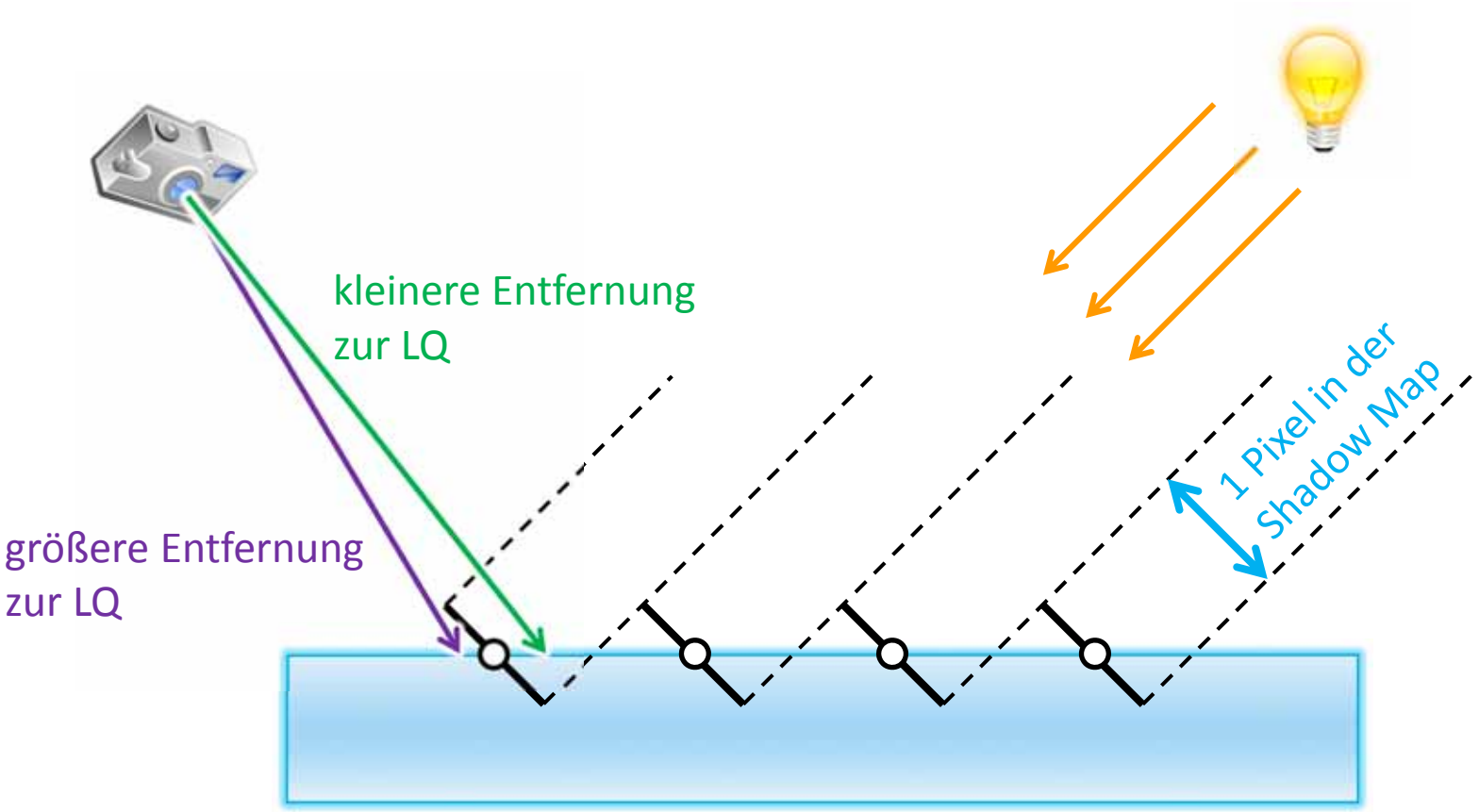
- Dual-Paraboloid Shadow Maps:
  - verwende zwei hemisphärische Projektionen
  - Anm. kann ebenfalls für Environment Mapping verwendet werden
  - Details später!



Shadow Maps: Schwierigkeiten

Shadow Acne / Surface Acne Problem

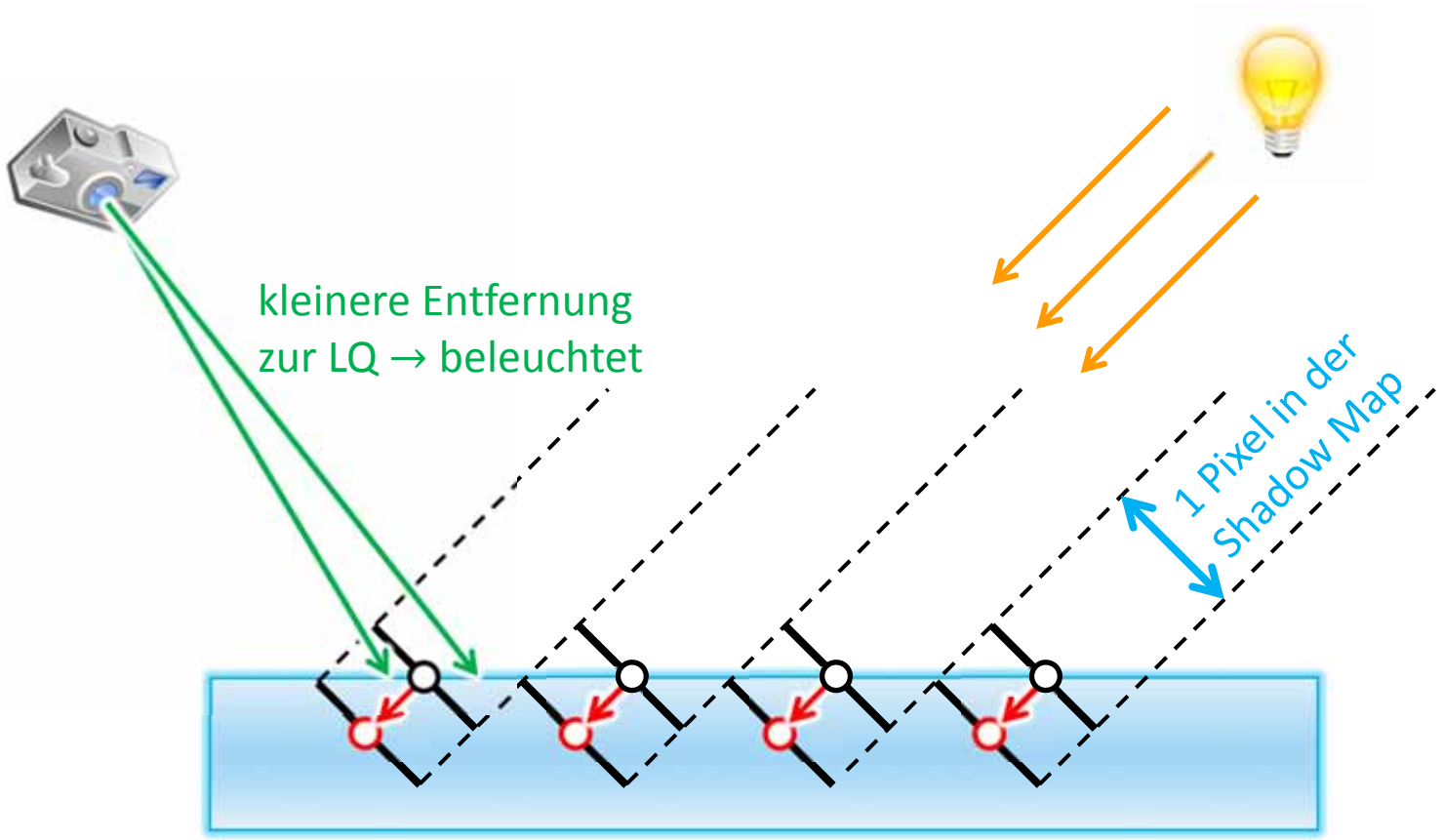
- die Oberfläche wird nur an den Mittelpunkten der Shadow Map Texel abgetastet, die Tiefenwerte stimmen also nicht perfekt überein
  - würde es helfen die Orientierung der Oberflächen mitzuspeichern? nur in so einfachen Fällen (glatte Fläche) wie hier auf der Folie!



Shadow Maps: Schwierigkeiten

Shadow Acne / Surface Acne Problem

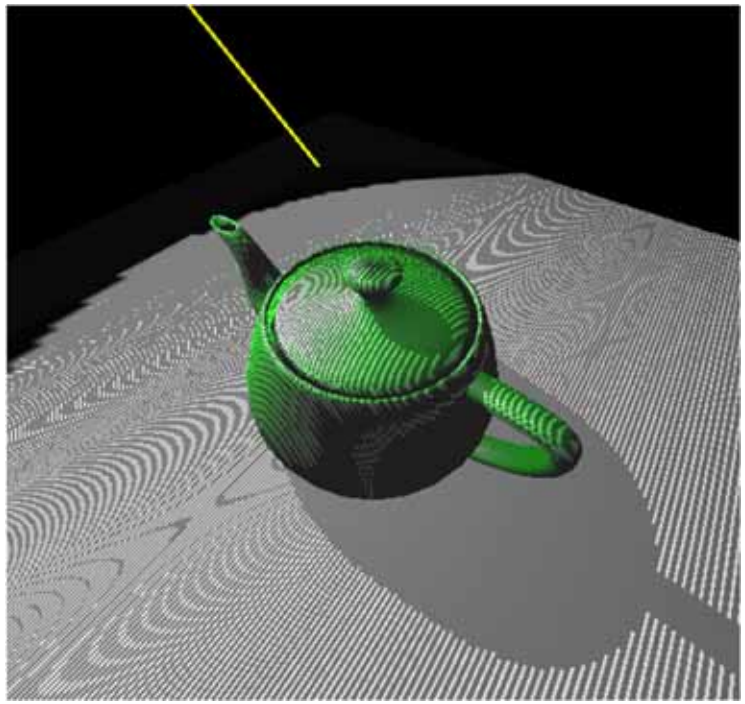
- Lösung: bei der Generierung der Shadow Map werden die Flächen etwas von der LQ weggeschoben (i.d.R. abhängig von der Orientierung)
- OpenGL: `glEnable( GL_POLYGON_OFFSET_FILL )` und `glPolygonOffset( offsetFactor, offsetUnits )`

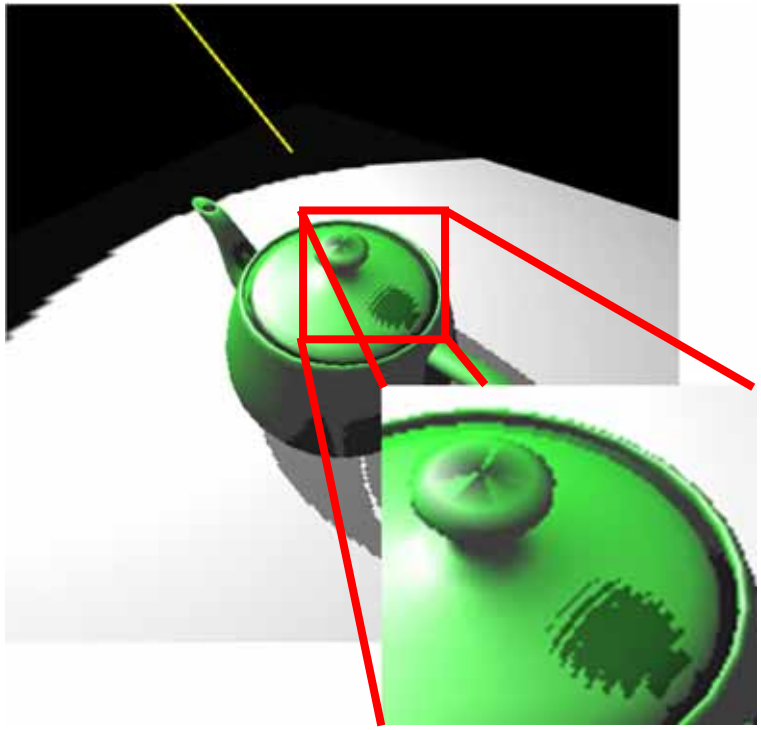


Shadow Maps: Schwierigkeiten

Shadow Acne / Surface Acne Problem

- links: keine Verschiebung
- rechts: durch zu große Verschiebung können Schattenteile verschwinden

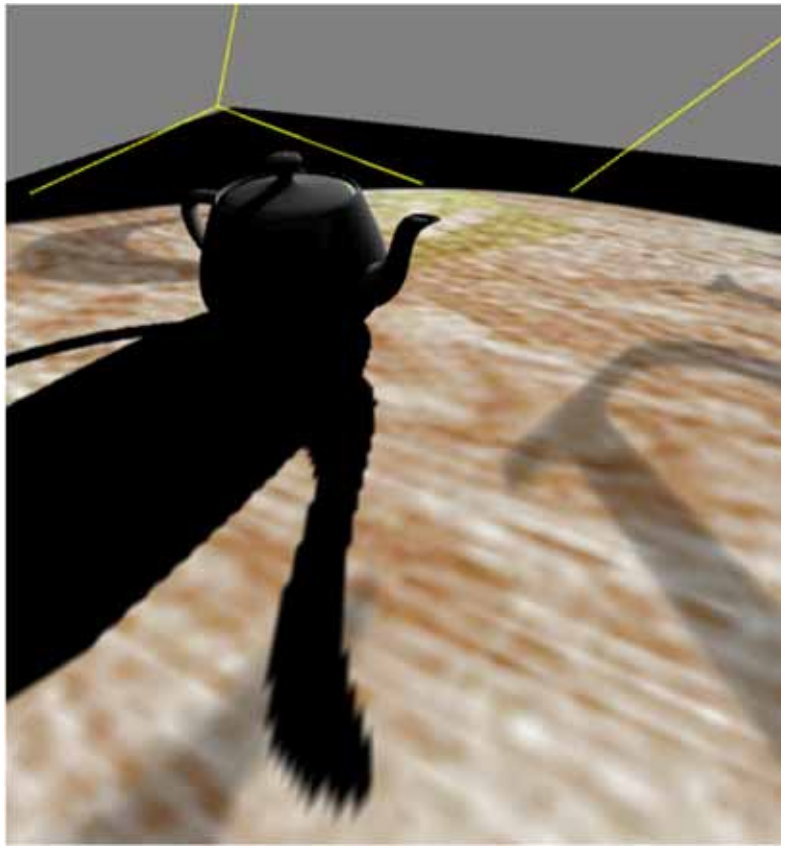




Shadow Maps: Schwierigkeiten

Aliasing Artefakte

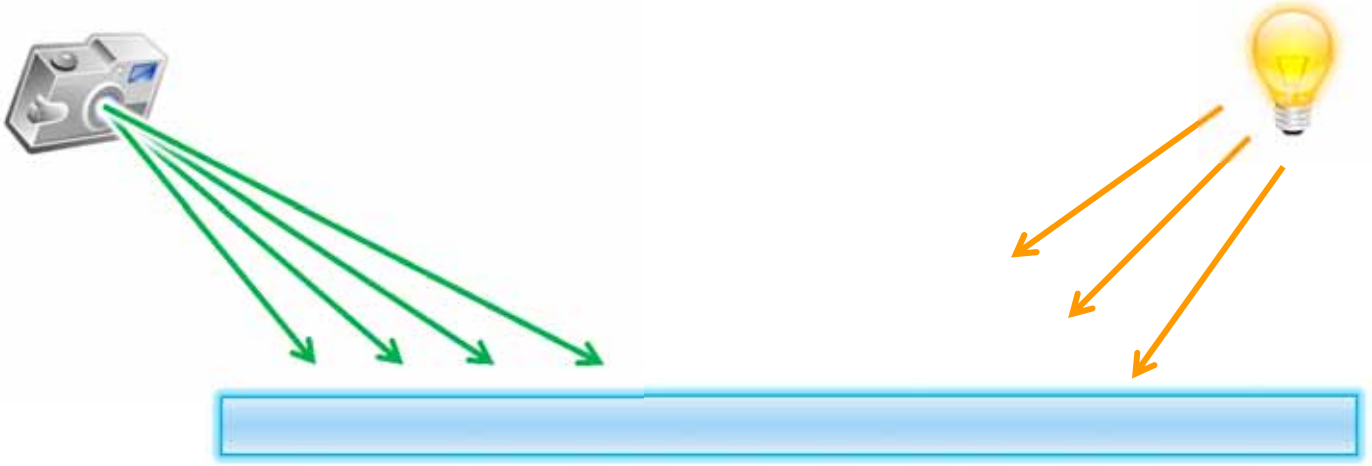
- die Shadow Map tastet die Szenengeometrie mit endlicher Auflösung ab
- je nach relativer Platzierung der Lichtquelle und der Kamera treten dadurch stärkere Artefakte auf und Shadow Map Texel werden sichtbar



Shadow Maps: Schwierigkeiten

Aliasing Artefakte

- der ungünstigster Fall tritt ein, wenn die Blickrichtung der Lichtrichtung entgegengesetzt ist
  - Flächen die fein in der SM abgetastet werden sind klein im Kamerabild
  - Flächen die klein in der SM sind, nehmen viel Platz im Kamerabild ein



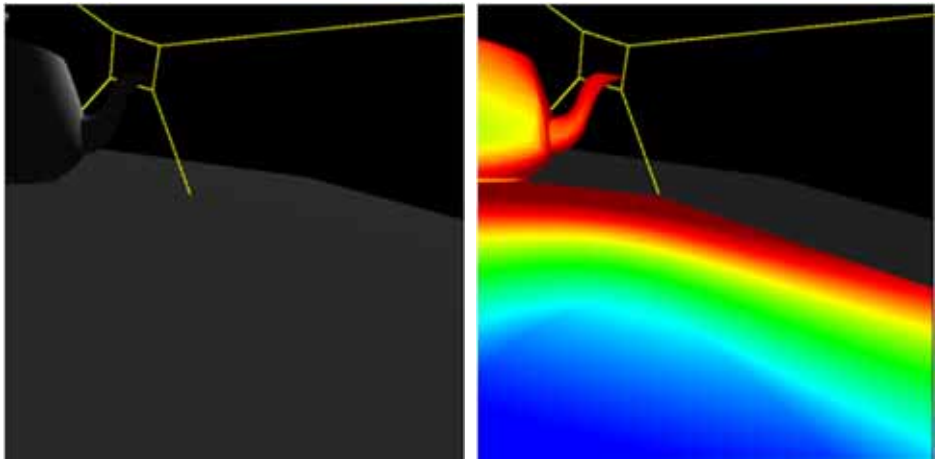
- Abtastung in der SM (blau = hoch/fein, rot = niedrig)

- Abtastung im Kamerabild

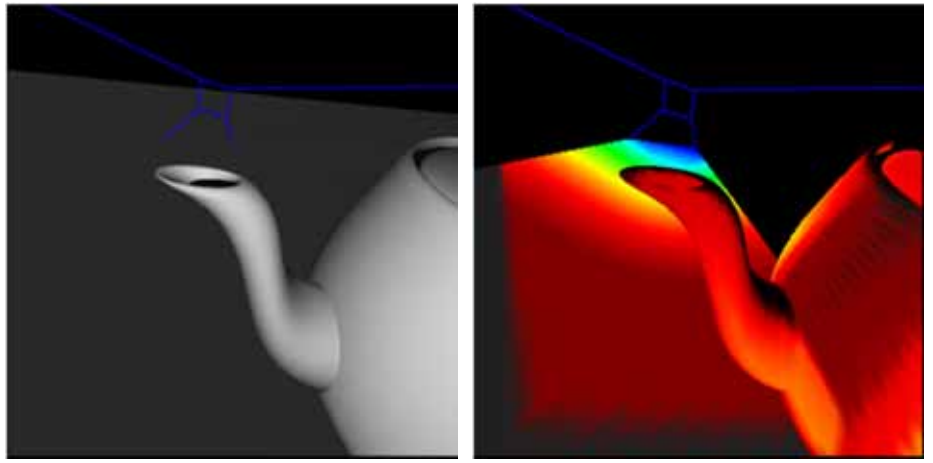
Shadow Maps: Schwierigkeiten

Aliasing Artefakte

- der ungünstigster Fall tritt ein, wenn die Blickrichtung der Lichtrichtung entgegengesetzt ist
  - Flächen die fein in der SM abgetastet werden sind klein im Kamerabild
  - Flächen die klein in der SM sind, nehmen viel Platz im Kamerabild ein



Sicht der Kamera

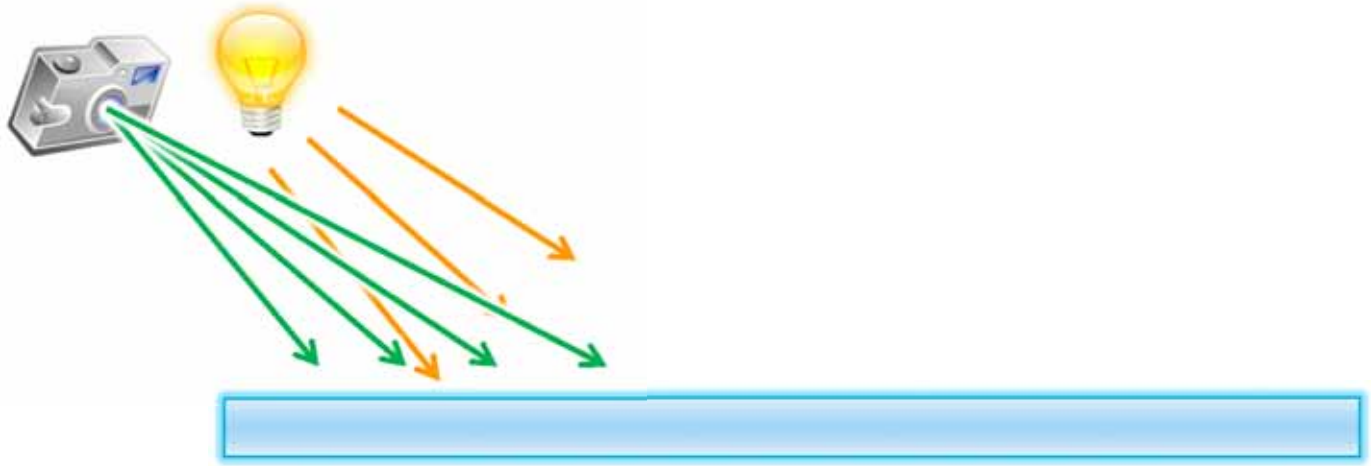


Sicht der Lichtquelle
(gleiche Einfärbung wie links)

Shadow Maps: Schwierigkeiten

Aliasing Artefakte

- der ideale Fall tritt ein, wenn die Position/Richtung der Lichtquelle und die der Kamera in etwa übereinstimmen („Miner‘s Lamp“)
- die Abtastung der Flächen in SM und Kamerabild ist ähnlich verteilt



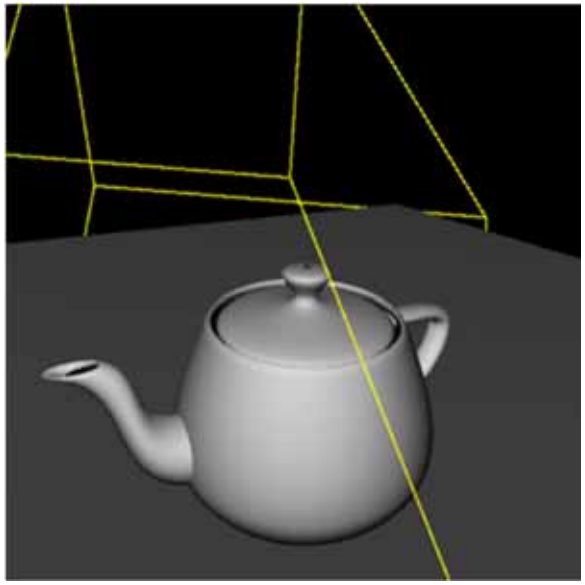
- Abtastung in der SM (rot = fein, blau = grob)

- Abtastung im Kamerabild

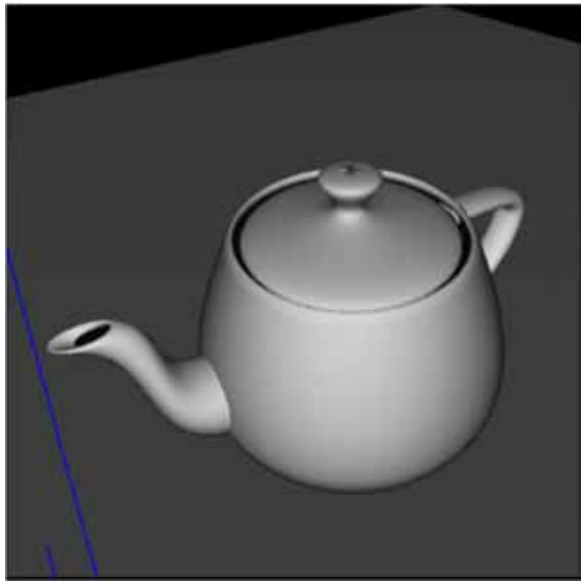
Shadow Maps: Schwierigkeiten

Aliasing Artefakte

- der ideale Fall tritt ein, wenn die Position/Richtung der Lichtquelle und die der Kamera in etwa übereinstimmen („Miner‘s Lamp“)
- die Abtastung der Flächen in SM und Kamerabild ist ähnlich verteilt



Sicht der Kamera

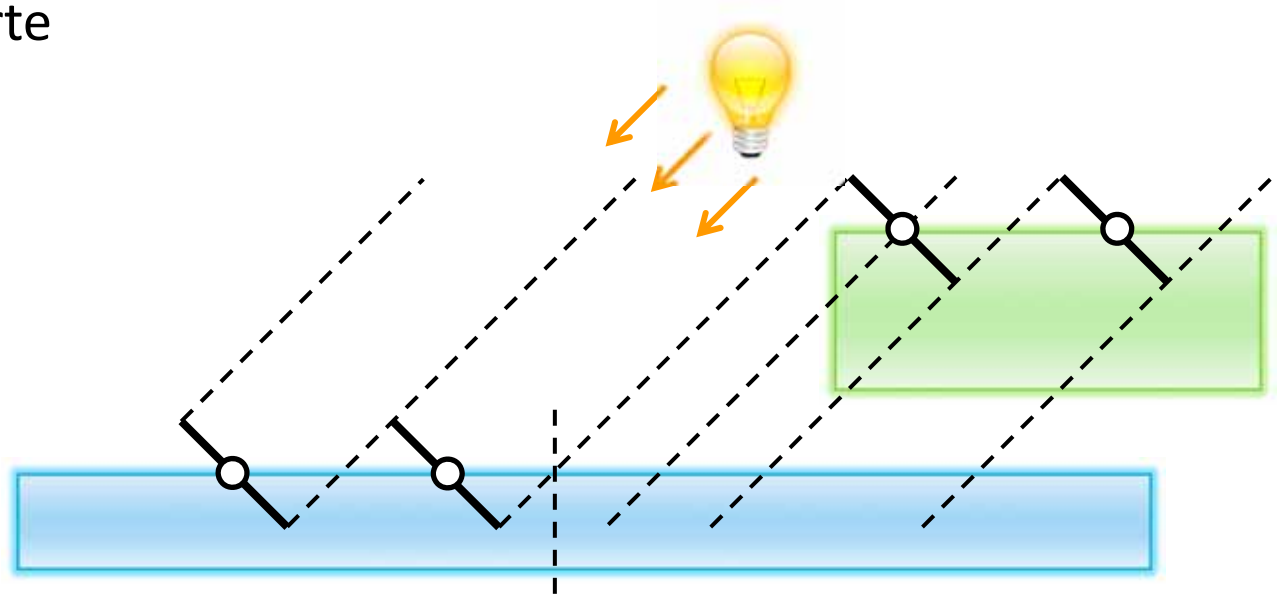


Sicht der Lichtquelle

Filterung für Shadow Maps

Aliasing Artefakte

- Artefakte durch begrenzte Auflösung treten an den Schattenkanten auf
- **Tiefenwerte einer Shadow Map dürfen nicht wie Farben gefiltert werden**, also keine bilineare Interpolation oder Mip-Mapping der Tiefenwerte

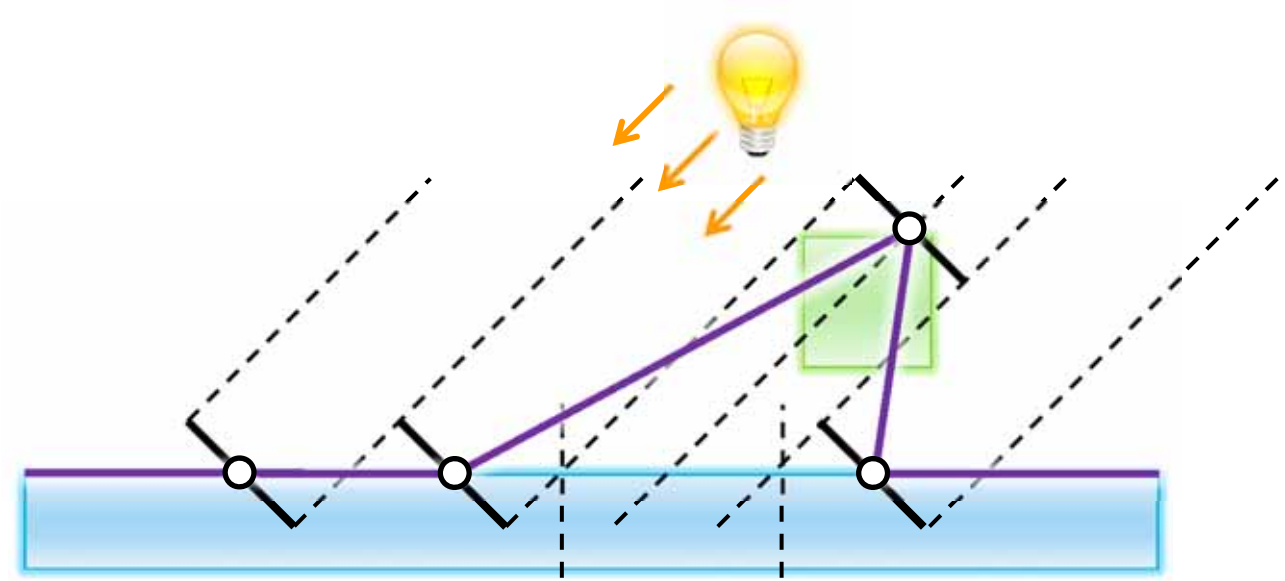


- Helligkeitsverlauf auf der blauen Fläche:

Filterung für Shadow Maps

Aliasing Artefakte

- Tiefenwerte dürfen nicht wie Farben gefiltert werden
- Beispiel: Flächenverlauf durch (bi)lineare Interpolation



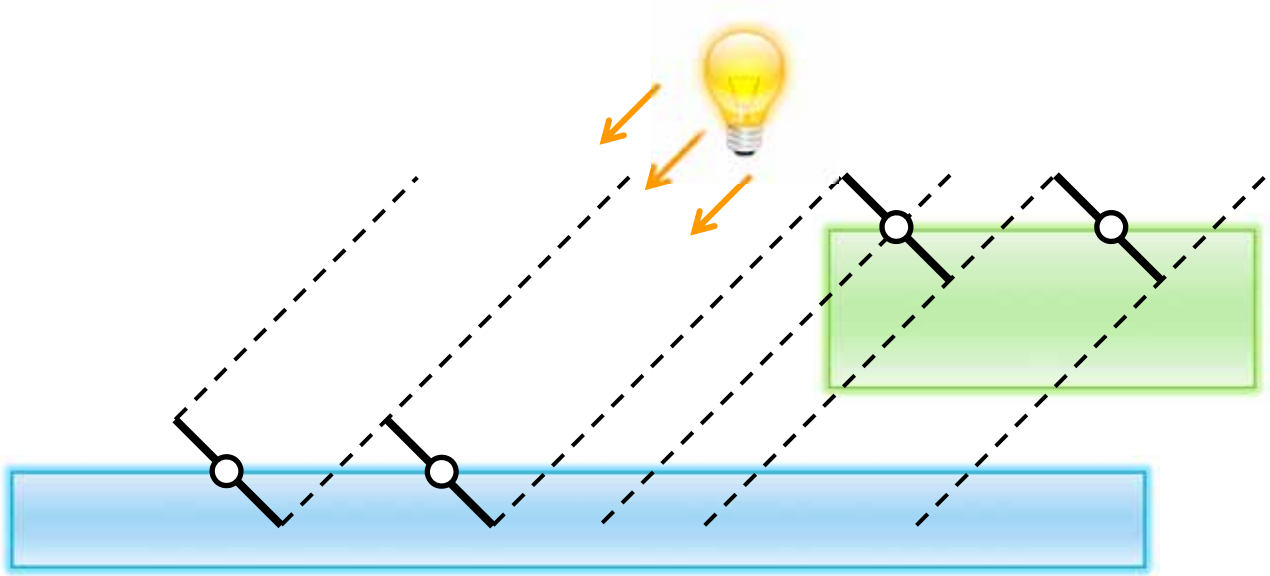
- Helligkeitsverlauf auf der blauen Fläche ohne Interpolation:

- Helligkeitsverlauf auf der blauen Fläche mit Interpolation:

Filterung für Shadow Maps

Aliasing Artefakte

- **Percentage Closer Filtering** [Reeves 1987]: führe den Schattentest mit den 4 **Nachbartexeln** durch und interpoliere das Resultat bilinear



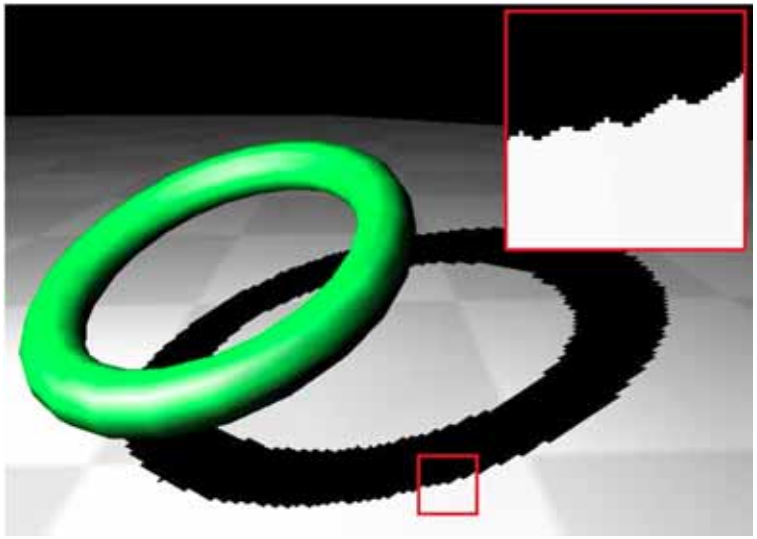
- Helligkeitsverlauf ohne PCF:

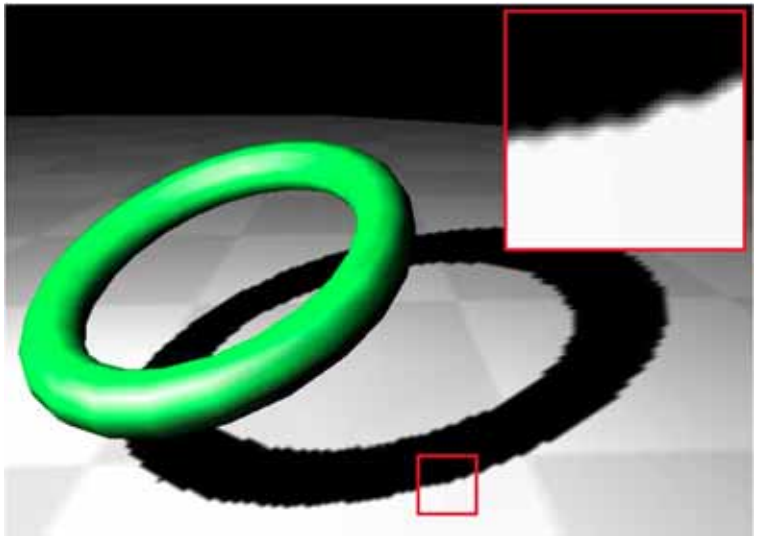
- Helligkeitsverlauf mit PCF:

Filterung für Shadow Maps

Percentage Closer Filtering

- bilineare Interpolation der Resultate des Schattentests
- reduziert die Artefakte etwas, liefert aber keine weichen Schatten

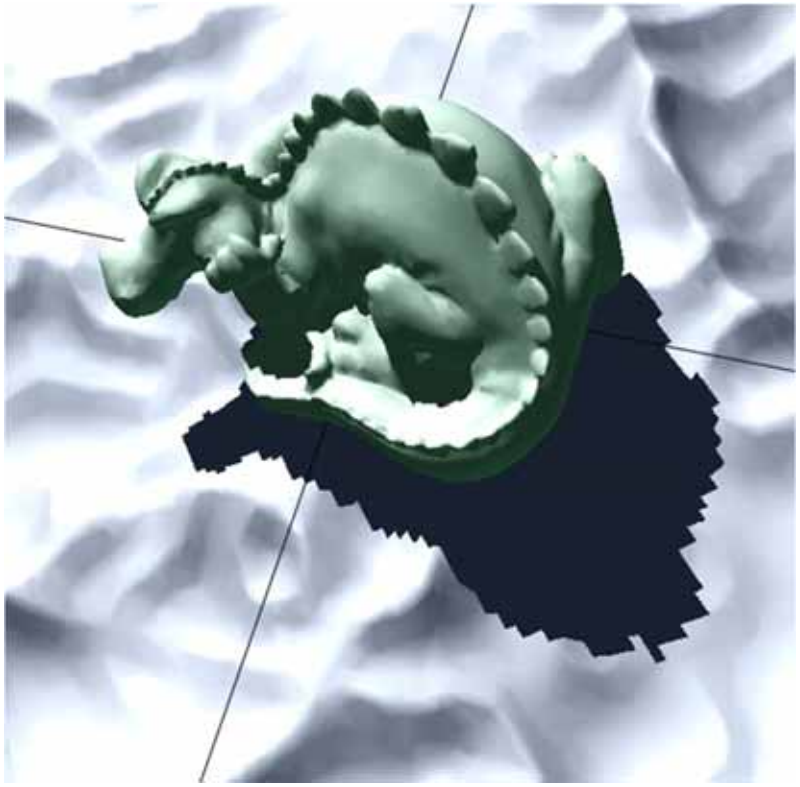


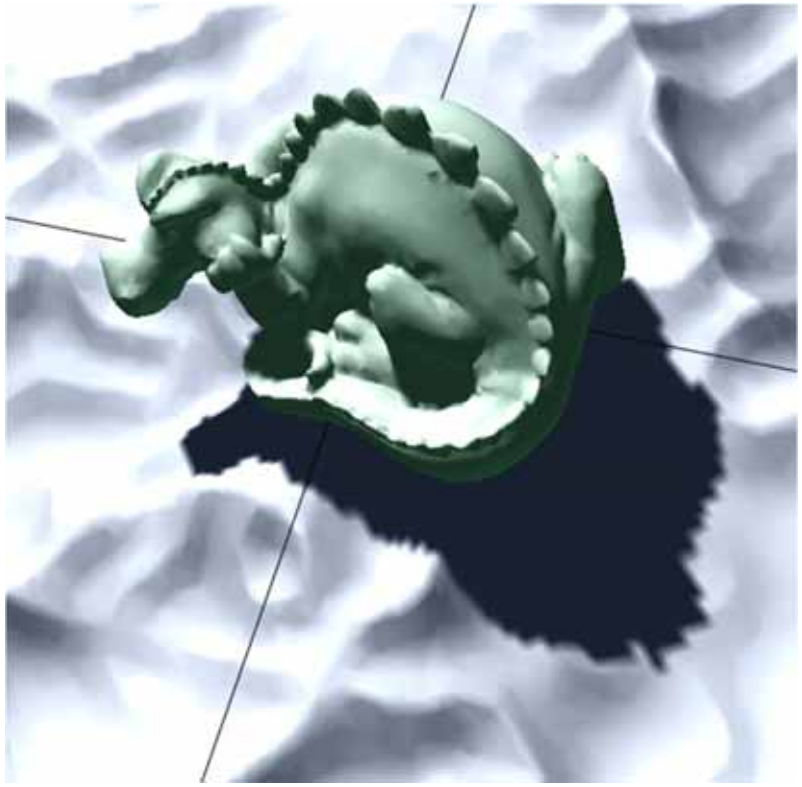


Filterung für Shadow Maps

Percentage Closer Filtering

- bilineare Interpolation der Resultate des Schattentests
- reduziert die Artefakte etwas, liefert aber keine echten weichen Schatten

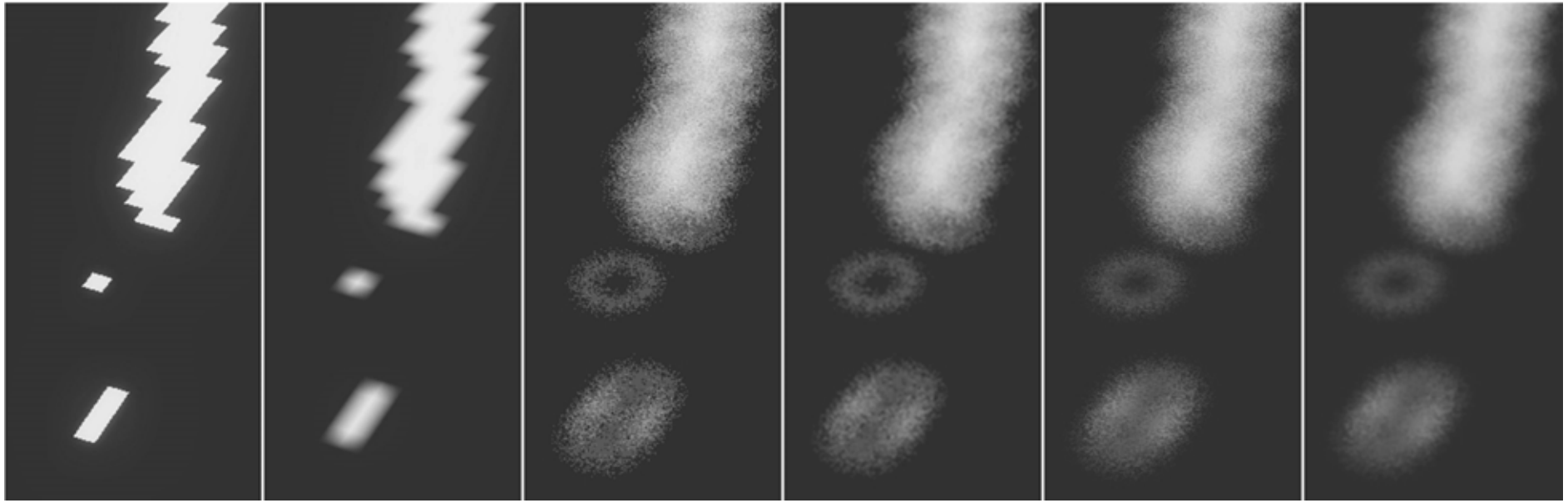




Filterung für Shadow Maps

Stochastische Filterung

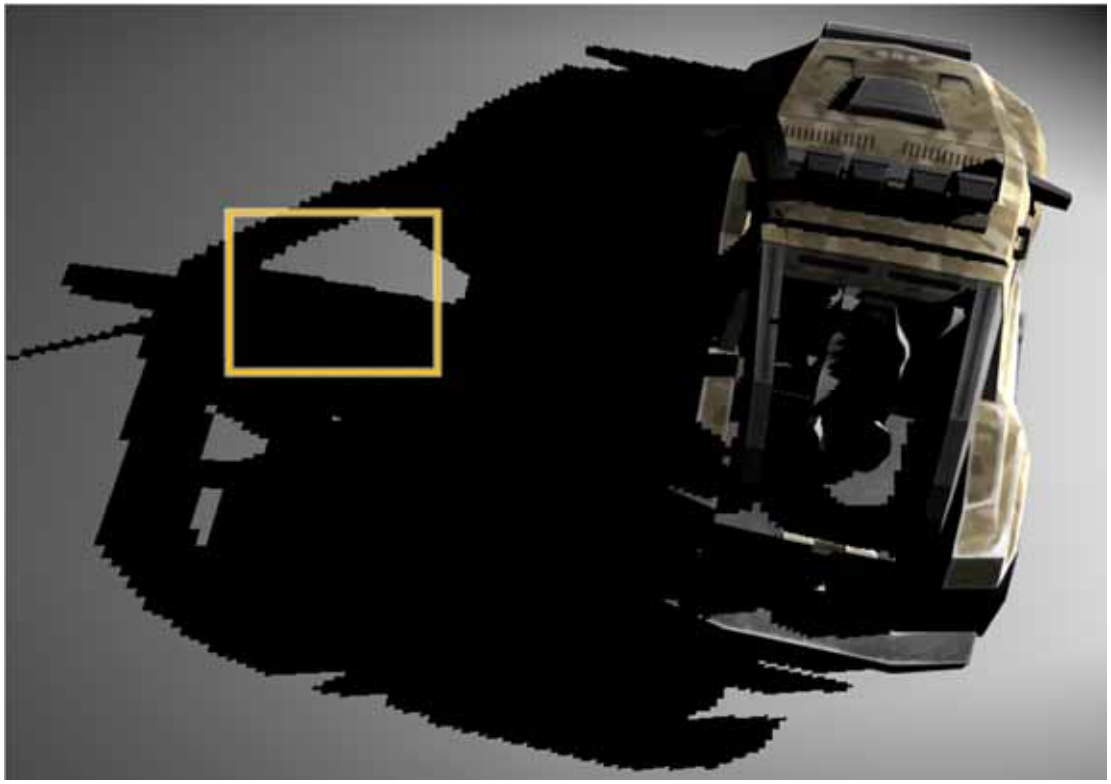
- in der Praxis wird oft mehrmals, an zufällig leicht verschobenen Stellen auf die Shadow Map zugegriffen und die Ergebnisse des Schattentests gemittelt (meist noch gewichtet ähnlich einem Unschärfefilter)



ohne Filter | PCF | 8 Samples | 8 Samp.+Blur | PCF+8 Samples | PCF+8 S.+Blur

Filterung für Shadow Maps

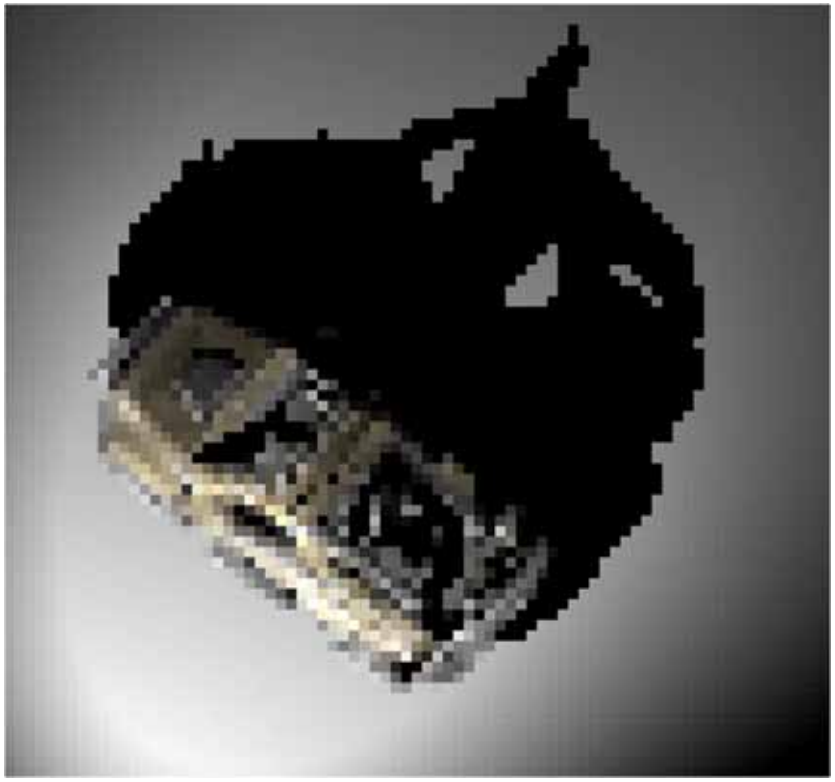
- PCF löst den Fall der Vergrößerung nicht – zumindest nicht wirklich gut
- …und stochastische Filterung ist vergleichsweise teuer…



Bilder: Andrew Lauritzen

Filterung für Shadow Maps

- im Fall der Verkleinerung „schwimmen“ Schattenteile
- Folge der Unterabtastung des Shadow Map bei der Verwendung
  - dieses Problem wird durch stochastische Filterung teilweise reduziert



Bilder: Andrew Lauritzen

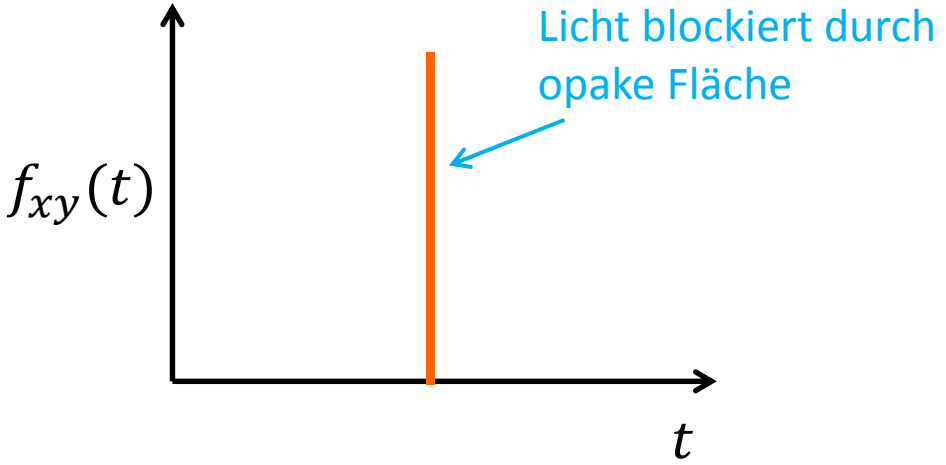
Filterung für Shadow Maps

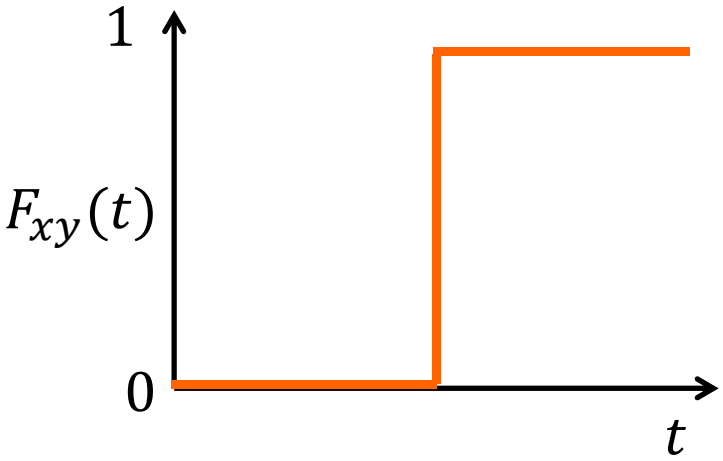
- Probleme treten auch durch Anisotropie auf:
  hohe Abtastrate in eine, niedrige in die andere Richtung
- wir betrachten im Folgenden einen Ansatz der eine Filterung (fast) direkt auf den Tiefenwerten durchführt und das „Minification“-Problem löst
  - **die Verwendung/das Nachschlagen der Shadow Map muss sich dadurch natürlich ändern!**

Bilder: Andrew Lauritzen

Schattentest

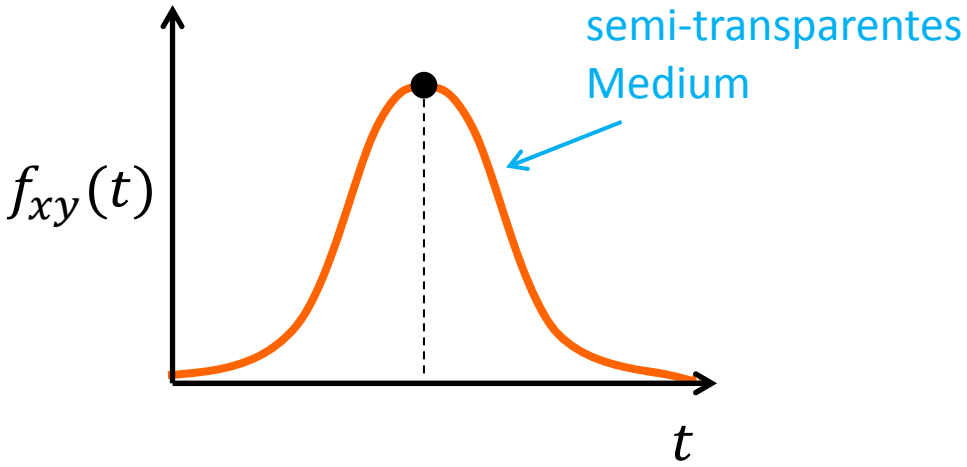
- zuerst noch eine etwas andere Sicht auf den Schattentest
  - sei $f_{xy}(t)$ die „Wahrscheinlichkeit“, dass sich Licht entlang eines Strahls (durch den SM-Pixel (x, y)) im Abstand t blockiert wird
  - dann beschreibt die „kumulative Verteilungsfunktion“ die Verschattung einer Oberfläche im Abstand t
    - $F_{xy}(t) = P(x \leq t)$: Wahrscheinlichkeit, dass ein Fragment mit dem Abstand t (zur Lichtquelle) im Schatten liegt

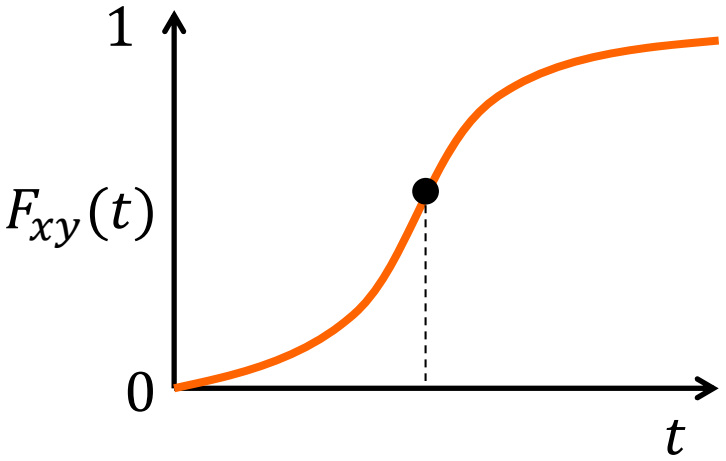




Schattentest

- im Prinzip kann $f_{xy}(t)$ beliebige Werte zwischen 0 und 1 annehmen
  - z.B. bei semi-transparenten Medien oder Flächen
- $F_{xy}(t)$ beschreibt dann die Verschattung (relativ zur unblockierten Beleuchtung) einer Oberfläche mit dem Abstand t zur Lichtquelle





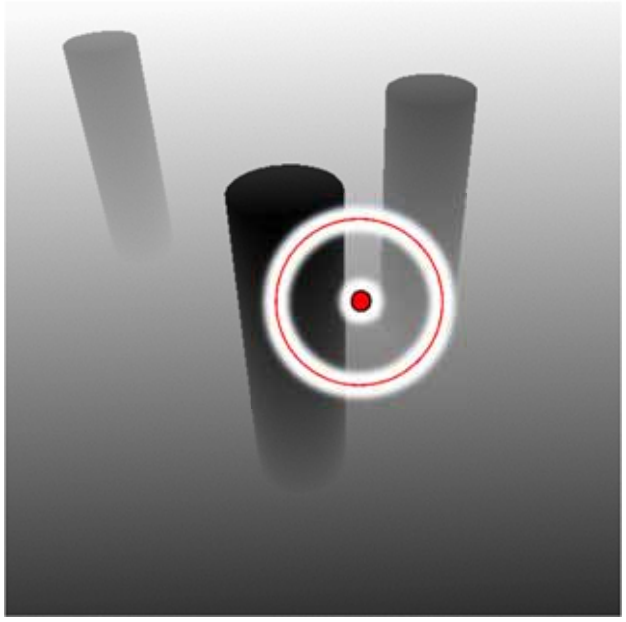
Schattentest

- mit dieser Sichtweise lassen sich zwei Schattenverfahren beschreiben
- **Deep Shadow Maps** (keine Details)
  - für volumetrische Objekte, z.B. Haare, Rauch
  - für jeden Pixel wird $f_{xy}(t)$ abgetastet (z.B. mit Ray Marching) und $F_{xy}(t)$ gespeichert
  - beim Rendering wird $F_{xy}(t)$ nachgeschlagen, ggf. linear interpoliert
  - aufwendig aber gute Resultate
  - Echtzeit-fähige Variante: Adaptive Volumetric Shadow Maps, 2010, http://visual-computing.intel-research.net/art/publications/avsm/

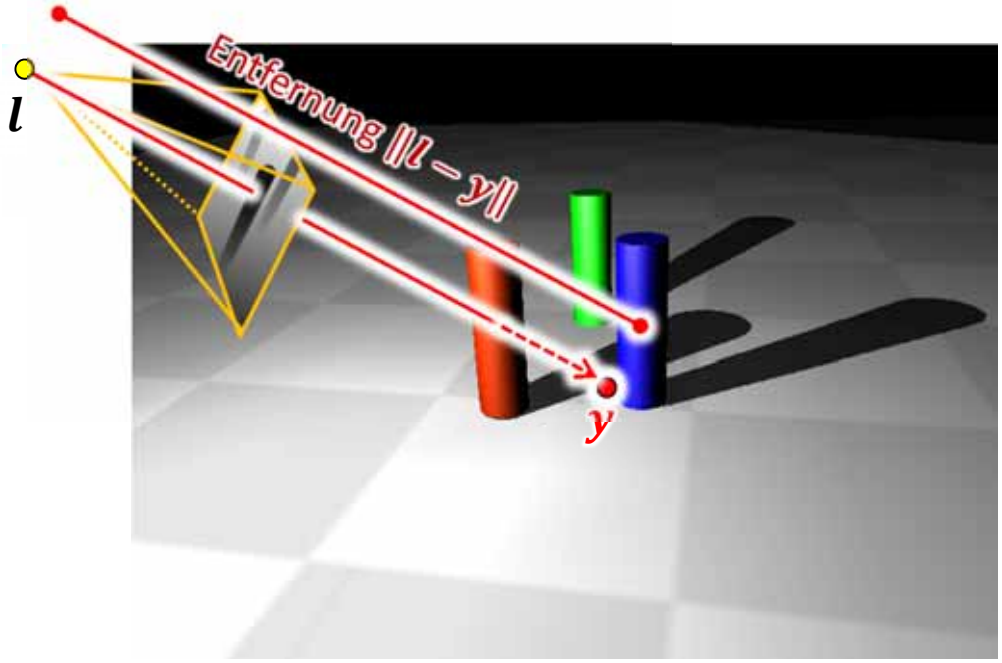
Schattentest

Variance Shadow Maps (VSMs)

- Idee: betrachte die **Verteilung der Tiefenwerte** in einem Bereich der SM als Zufallsvariable
- schätze damit die Wahrscheinlichkeit ab, dass ein Oberflächenpunkt beleuchtet ist



Betrachte die Verteilung der Tiefenwerte in einem Bereich

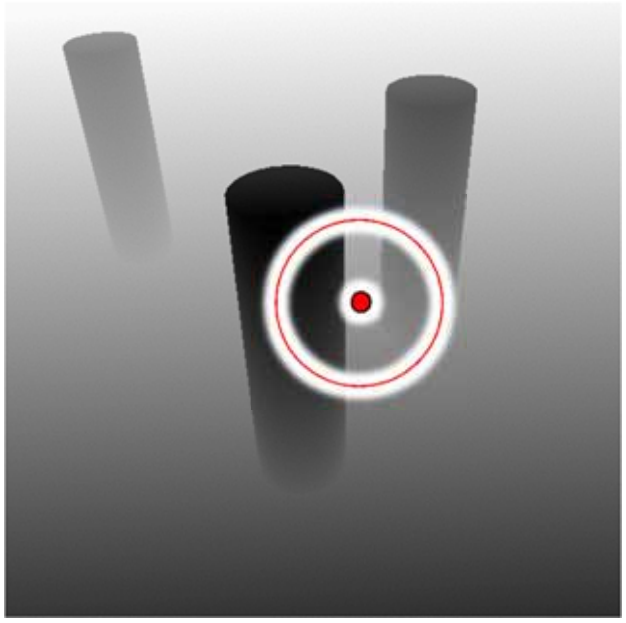


VSM: wie ist Wahrscheinlichkeit, dass ein Punkt mit dem Abstand $\|\boldsymbol{l} - \boldsymbol{y}\|$ im Schatten liegt?

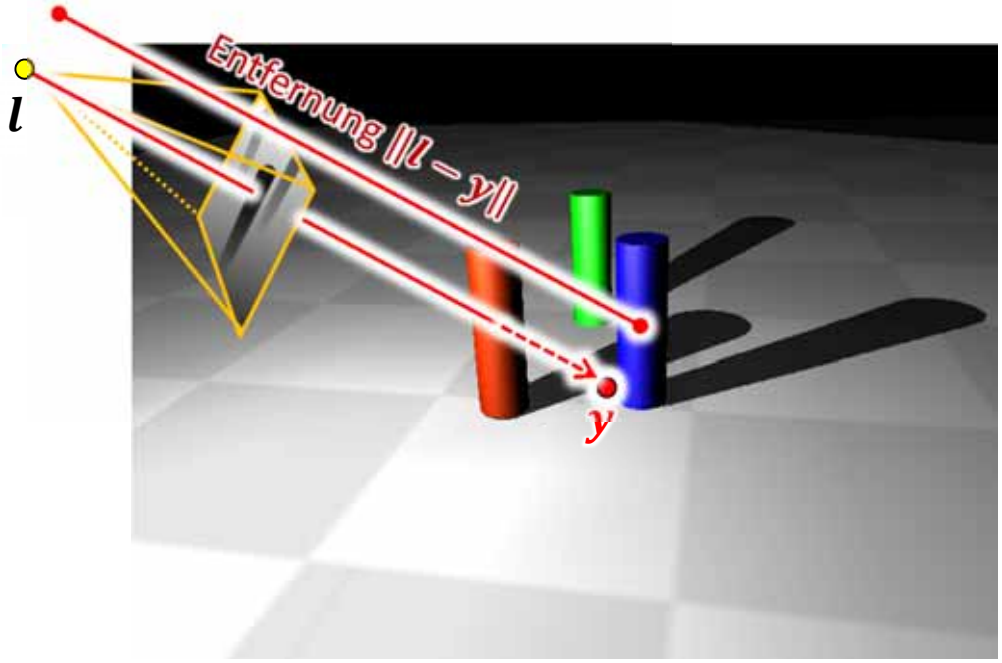
Schattentest

Variance Shadow Maps (VSMs)

- **Verteilung der Tiefenwerte** dargestellt in Form ihrer zentralen Momente: Mittelwert $E(x)$, Varianz $E(x^2) - E(x)^2$, Schiefe, Wölbung, …



Betrachte die Verteilung der Tiefenwerte in einem Bereich



VSM: wie ist Wahrscheinlichkeit, dass ein Punkt mit dem Abstand $\|\boldsymbol{l} - \boldsymbol{y}\|$ im Schatten liegt?

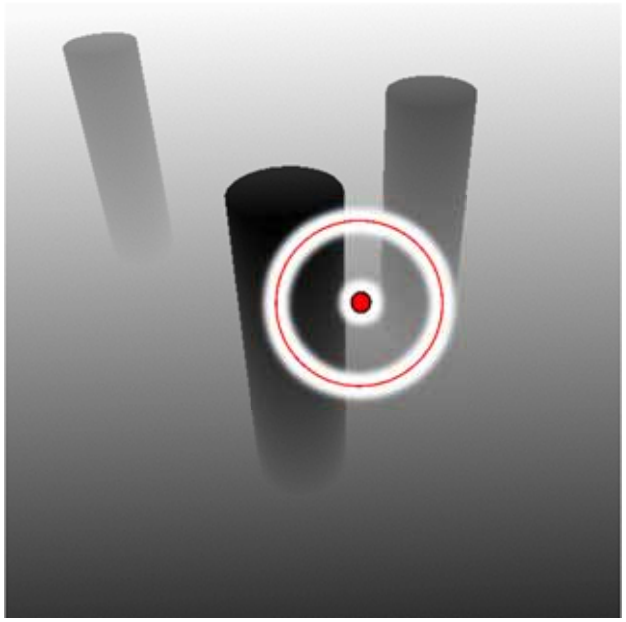
Variance Shadow Maps

Momente einer Verteilung

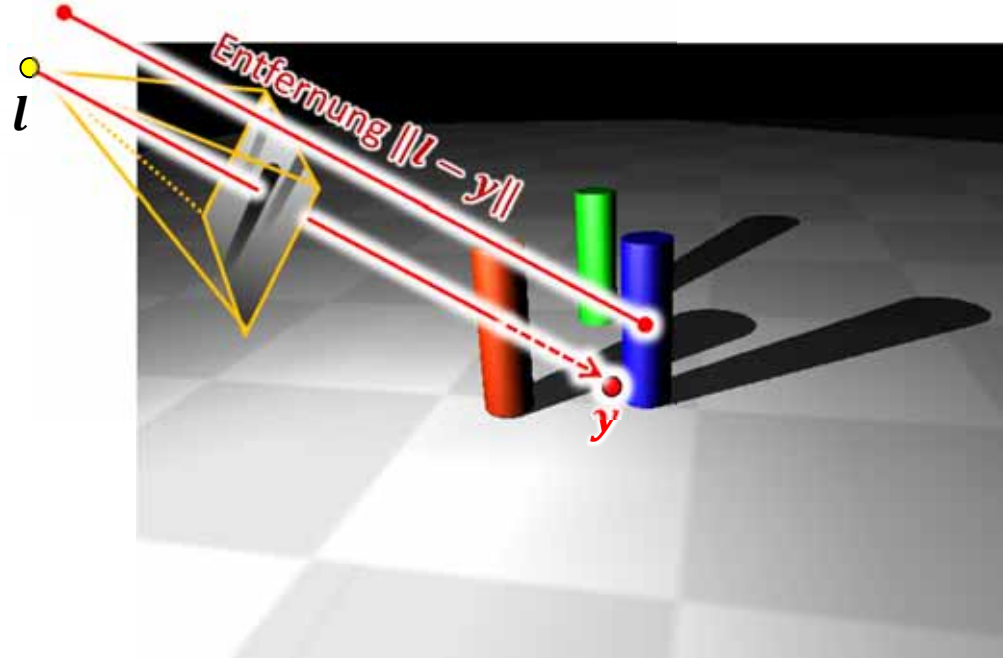
- zur Beschreibung der Eigenschaften von Verteilungen
  - 1. Moment: Mittelwert einer Stichprobe $E(x) \coloneqq \mu = \frac{1}{N}\sum_{i=1}^{N} x_i$
  - r-tes zentrales Moment $\frac{1}{n}\sum_{i=1}^{N}(x_i - \mu)^r$
    - Varianz: $r = 2$ $\sigma^2 = \frac{1}{n}\sum_{i=1}^{N}(x_i - \mu)^2$
    - Schiefe ($r = 3$) und Wölbung ($r = 4$) werden bei den VSMs nicht verwendet (nicht verlässlich bei der Verschattungsabschätzung)
- VSMs speichern pro Pixel die Tiefe x und die quadrierte Tiefe x^2 (in Floating Point Genauigkeit in eigenem Rendertarget mittels FBOs)
  - wichtig ist der Verschiebungssatz: $\sigma^2 = E(x^2) - E(x)^2$
  - nach (linearer) Filterung der VSM (z.B. Mittelung über einen Bereich der VSM) erhalten wir aus den gefilterten (quadrierten) Tiefen
    - $\mu = E(x) = \int x \cdot p(x) dx$ (← „Filter“ $p(x)$)
    - und $\sigma^2 = E(x^2) - \mu^2$ mit $E(x^2) = \int x^2 \cdot p(x) dx$

Variance Shadow Maps

- bestimme $E(x)$ und $E(x^2)$ für einen Bereich der Shadow Map (i.d.R. die Umgebung der Stelle, an der normalerweise getestet würde)
- schätze Anhand Mittelwert und Varianz die Wahrscheinlichkeit ab, dass die Oberfläche dort weiter entfernt ist, als der betrachtete Punkt $\boldsymbol{y}$
- gleichbedeutend mit der Wahrscheinlichkeit, dass eine Oberfläche mit dem Abstand t (zur Lichtquelle) nicht verschattet ist, also $P(x \geq t)$



Betrachte die Verteilung der Tiefenwerte in einem Bereich

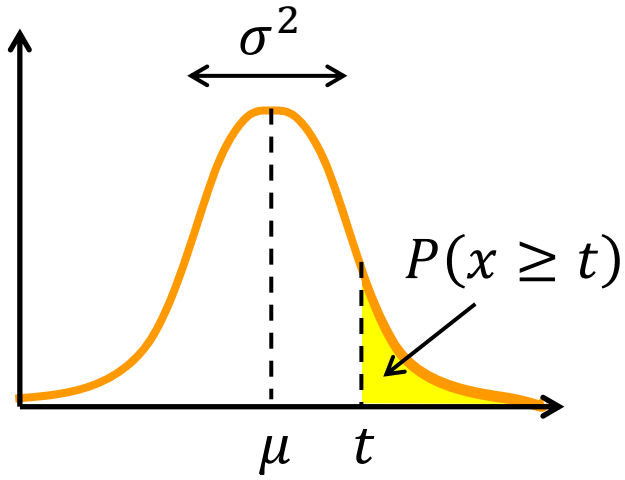


VSM: wie ist Wahrscheinlichkeit, dass ein Punkt mit dem Abstand $\|\boldsymbol{l} - \boldsymbol{y}\|$ im Schatten liegt?

Variance Shadow Maps

- bestimme $E(x)$ und $E(x^2)$ für einen Bereich der Shadow Map (i.d.R. die Umgebung der Stelle, an der normalerweise getestet würde)
- schätze Anhand Mittelwert und Varianz die Wahrscheinlichkeit ab, dass die Oberfläche dort weiter entfernt ist, als der betrachtete Punkt y
- gleichbedeutend mit der Wahrscheinlichkeit, dass eine Oberfläche mit dem Abstand t (zur Lichtquelle) nicht verschattet ist, also $P(x \geq t)$
- wir kennen den Mittelwert μ und die „Streuung“ der Tiefe σ^2
- die Information genügt natürlich nicht, um die Beleuchtung exakt zu bestimmen, aber wir können mit Hilfe der Tschebyschow-Ungleichung (einseitige Variante) eine obere Grenze angeben:

$$P(x \geq t) \leq p_{max}(t) \equiv \frac{\sigma^2}{\sigma^2 + (t-\mu)^2}$$



Variance Shadow Maps

- im VSM-Verfahren nimmt man als Beleuchtungswert L
  - volle Beleuchtung, wenn der „normale“ Schattentest (gegen μ) bestanden wird, ansonsten die einseitige Tschebyschow-Abschätzung

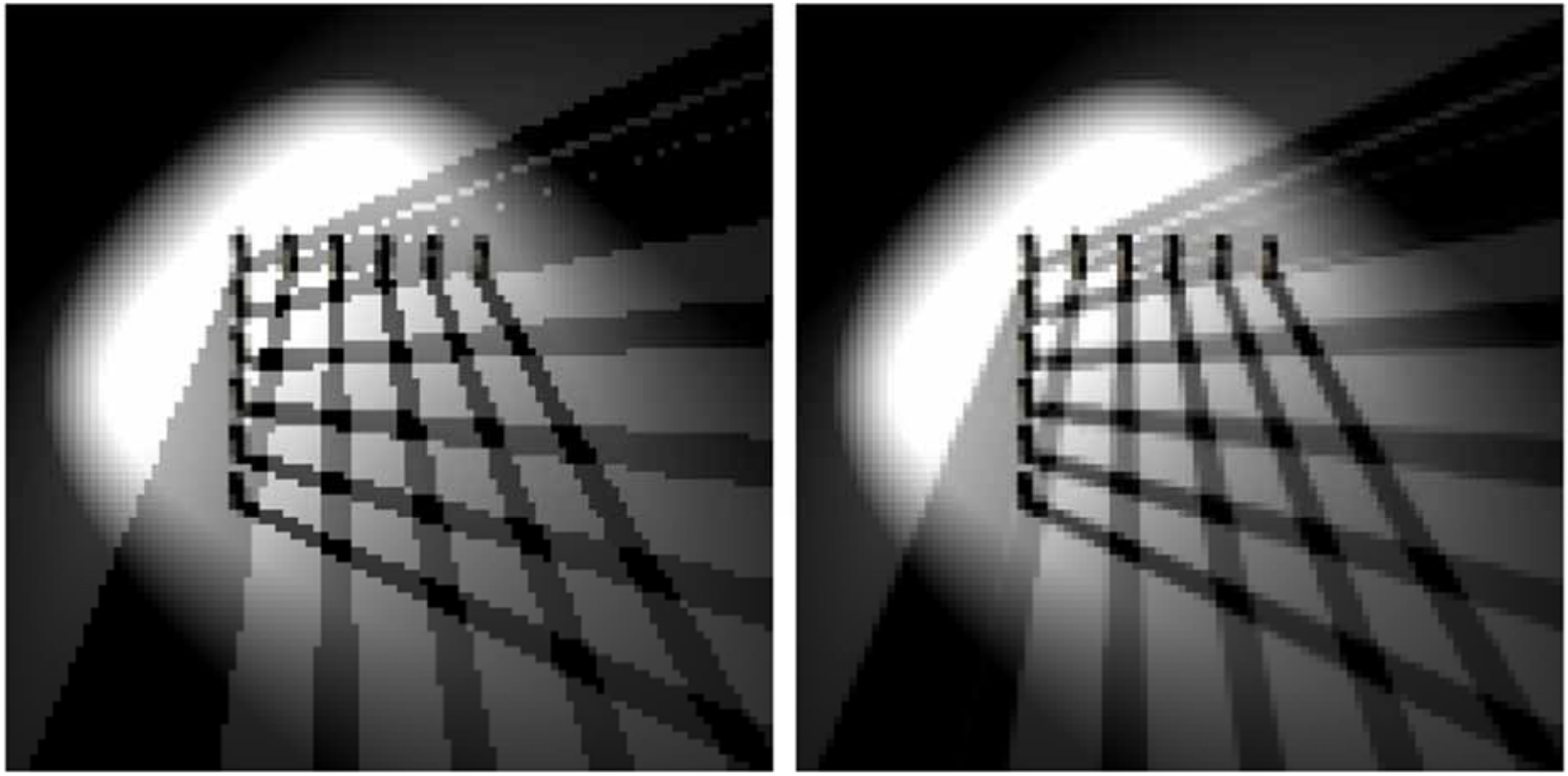
$$L = \max(p_{max}(t), (t < \mu) \,?\, 1 : 0)$$

$$p_{max}(t) = \frac{\sigma^2}{\sigma^2 + (t - \mu)^2}$$

  - im Fall eines planaren Schattenwerfers und eines planaren Schattenempfängers ist das Resultat exakt
  - sinnvolle Approximation für normale Szenen: meist weisen lokale Nachbarschaften (außer an Kanten) bei Verdecker und Empfänger ähnliche Tiefenwerte auf, d.h. kleine Varianz
  - Diskussion wann diese Abschätzung akkurat oder ungenau ist findet sich in der Publikation: http://www.punkuser.net/vsm/

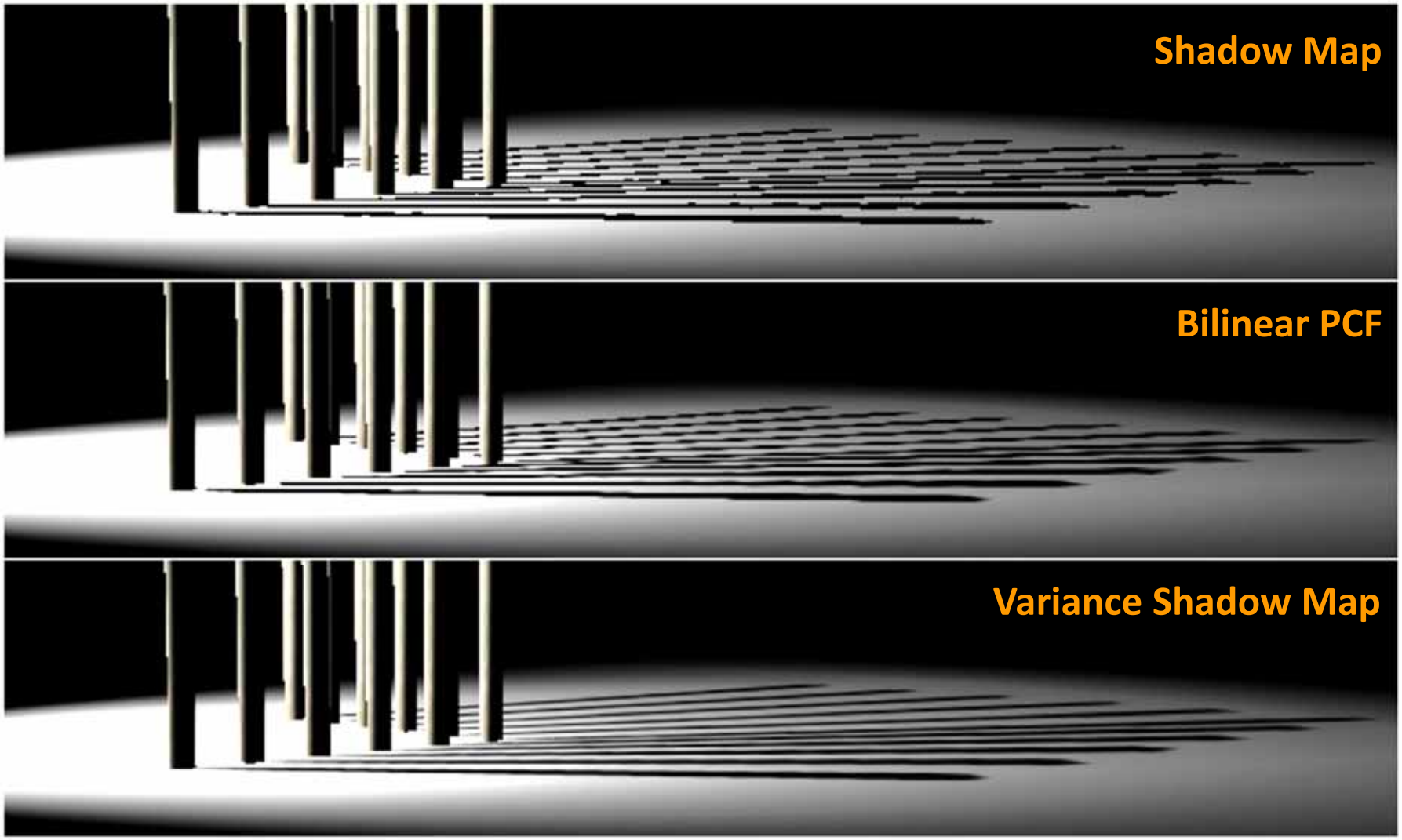
Variance Shadow Maps

- … warum verwendet man VSMs überhaupt?
- ermöglicht die Verwendung von Mip-Mapping für $E(x)$ und $E(x^2)$



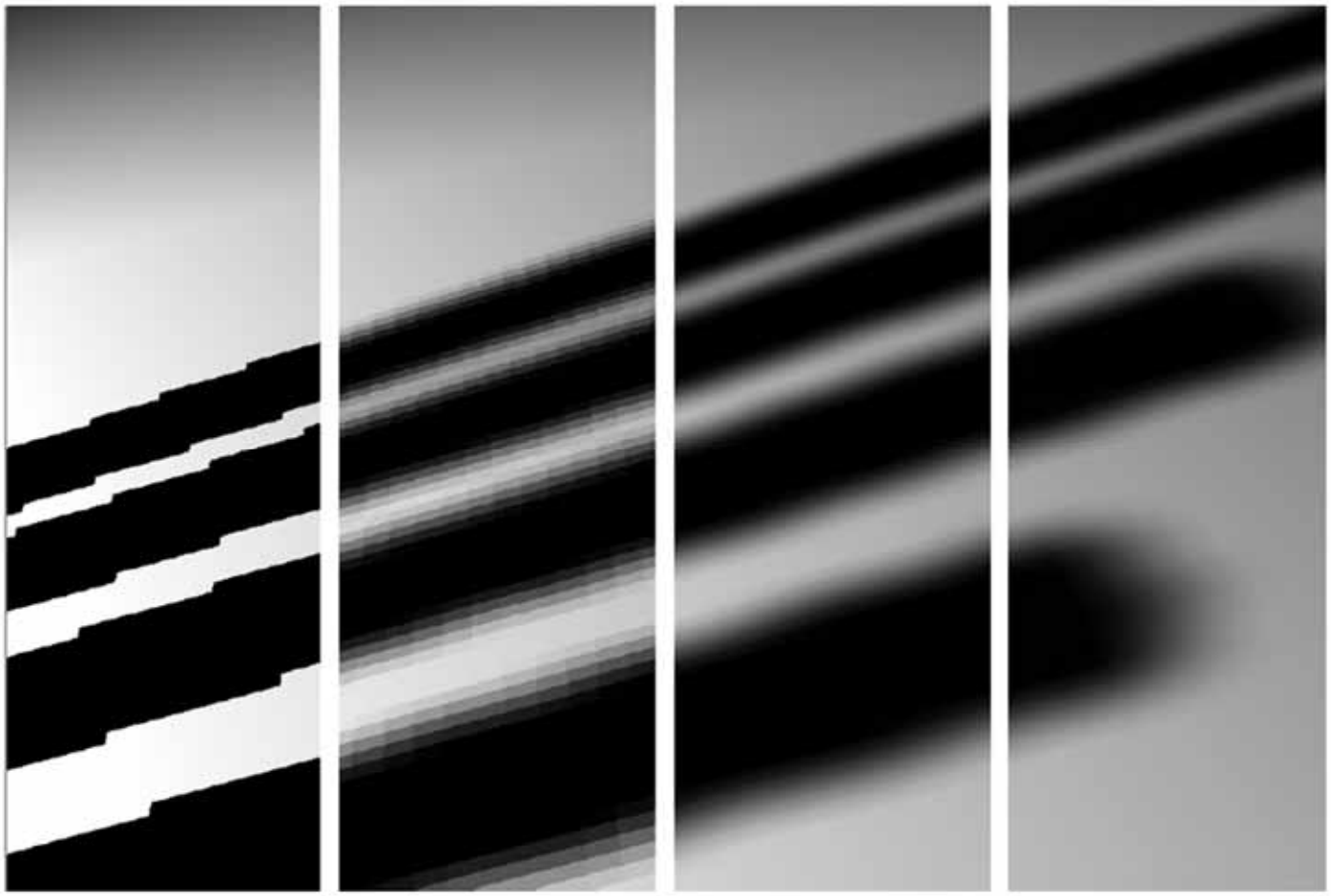
Variance Shadow Maps

- …ermöglicht die Verwendung von anisotropen Texturfiltern…



Variance Shadow Maps

- …und beliebigen linearen Filtern für „weiche“ Schatten
  - z.B. auch den separierbaren Gauß-Filter $O(2n)$



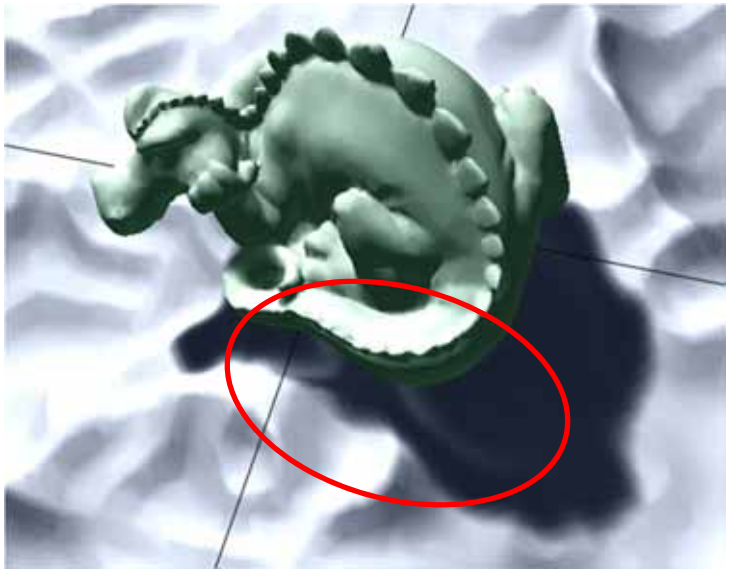
SM | 5x5 Samples | PCF 5x5 Samples | VSM

Variance Shadow Maps

Probleme bei der Abschätzung und Varianzberechnung

- die Berechnung der Varianz, $\sigma^2 = E(x^2) - E(x)^2$, ist numerisch problematisch, wenn $E(x^2) \approx E(x)^2$
- daher benötigt man ein Texturformat mit 2×32 Bit Floating Point (normales Shadow Mapping: 16-24 Bit, wie ein normaler Tiefenpuffer)
- sichtbarstes Artefakt: große Varianz verursacht „light bleeding“
  - Einfluss des Terms $(t - \mu)$ vernachlässigbar, wenn σ^2 zu groß:

$$P(x \geq t) \leq p_{max}(t) \equiv \frac{\sigma^2}{\sigma^2 + (t-\mu)^2}$$



Aliasing und Shadow Mapping

(Zwischen-)Fazit

- jeder Lichtquellentyp erfordert eine bestimmte Projektion/Abbildung
- Shadow Maps haben eine endliche Auflösung
  - Aliasing reduzieren wir mit PCF, stochastischem Filter oder VSM
  - Surface Acne benötigt einen Tiefenoffset (Bias)
  - es gibt günstige (Miner‘s Lamp) und ungünstige Konstellationen
    - kann man Letztere vermeiden? ein Ausblick…