Probelektion zum Thema

Shadow Rendering

Shadow Maps

Shadow Filtering

Einleitung


Renderman, 2006


Real reconstruction in Windisch, 2013


CityEngine 2011 Viewport

Einleitung

- Schatten bringen viel Realismus in eine Szene
- Schatten sind visuelle Hilfen für die Raumwahrnehmung
- Schatten für Effekte wie z.B. Spots

Ziele

- Wichtige Techniken kennenlernen die Schatten in Echtzeitanwendungen erlauben
- Grundprinzip und Funktionsweise von Shadow Maps verstehen
- Einfluss von Filtern verstehen

"Wo Licht ist, gibt es auch Schatten"

Fixed Pipeline:

```
glEnable ( GL_LIGHTING )
```

→ Leider keine Schatten

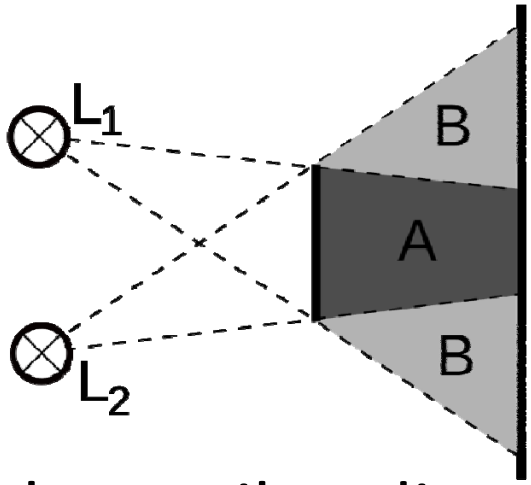
```
glEnable ( GL_SHADOWING )
```

→ Gibt's (leider) nicht

- OpenGL (wie auch DirectX) sind auf ein lokales Beleuchtungsmodell ausgerichtet
- Schatten sind ein *globales Beleuchtungsproblem* (welche Teile eines Objekt ist im Schatten von anderen Objekten?)

Einleitung

- A - Kernschatten (Umbra)
- B - Halbschatten (Penumbra)



- Es gibt viele weitere Effekte von Lichtanteilen die reflektiert, refraktiert oder absorbiert werden
  - → Global Illumination
  - → Ambient Occlusion, Photonmapping, Raytracing, Radiosity

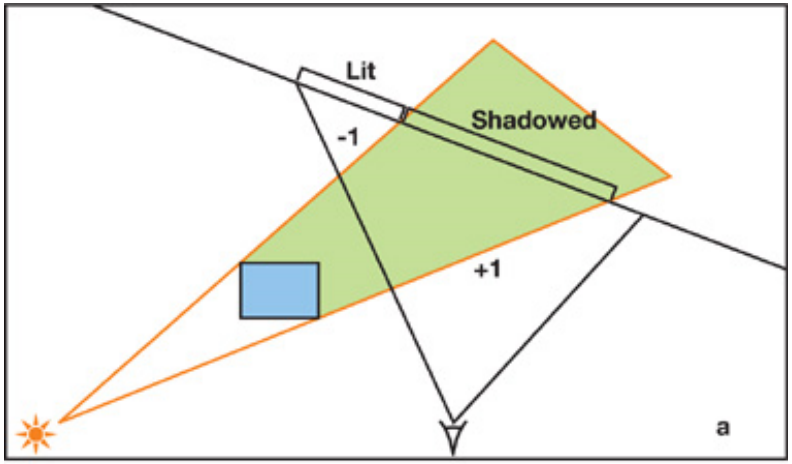
Einleitung

- Schatten für Echtzeitanwendungen werden normalerweise mit Deferred Rendering implementiert (siehe vorherige Vorlesung)

- D.h. in einem ersten Schritt werden Informationen gesammelt die dann für die Beleuchtungsberechnung in einem zweiten Schritt verwendet werden

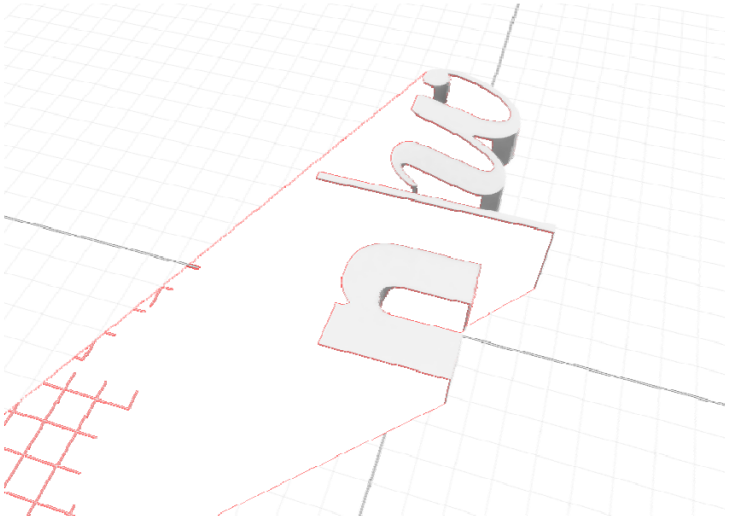
Shadow Volumes

- Jedes Polygon erzeugt ein Schattenvolumen (Pyramidenstumpf)
- Für jeden “Sichtstrahl” wird ein Zähler inkrementiert beim Eintreten in das Schattenvolumen und dekrementiert beim Austreten.
- Zonen mit einem Zähler == 0 sind beleuchtet, != 0 sind im Schatten
- Wie kann man das erreichen ohne Ray-Tracing?

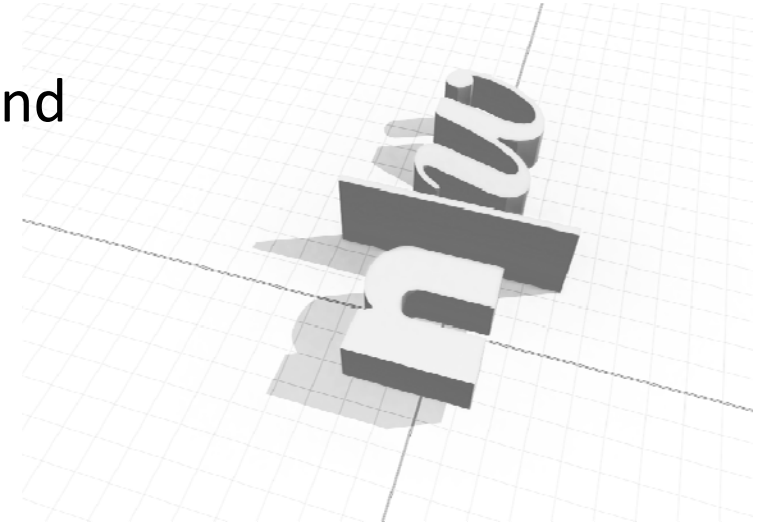


Shadow Volumes

- Lösung:
  - Stencil Buffer verwenden
  - Extrudierte Sahdow Volumes zeichnen mit
    - Front-Facing polygons +1
    - Back-Facing polygons -1



- Verfahren
  1. Rendern der Szene um Farb- und Tiefenwerte zu erhalten
  2. Rendern der Shadow Volumes
  3. Kombinieren der Resultate



Shadow Volumes
Vor- und Nachteile

- Einfaches Verfahren
- Alles aus der Kameraperspektive gezeichnet
- Präzise, geometrische Schatten

- Braucht Szenen Geometrie
- Fillrate (Zeichnen der Schattenvolumen)

Shadow Maps
Gruppenarbeit (3 min)

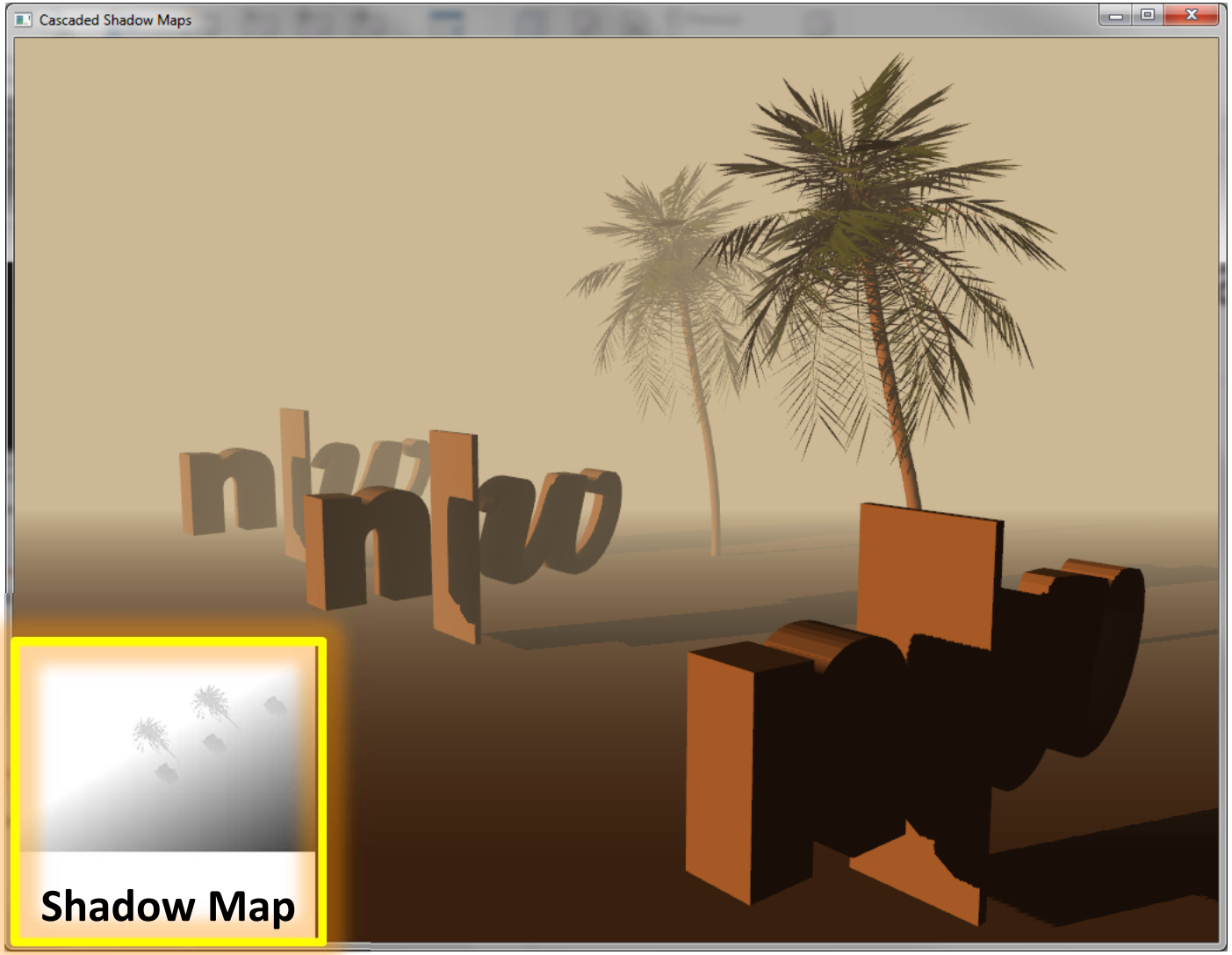
- Szene wird von der Kamera aus betrachtet
  →Blickrichtung

- Schatten hat etwas mit der Position der Lichtquelle zu tun
  →Lichtrichtung

- Schatten hat etwas mit Objektverdeckung zu tun
  →Depth-Buffer

- **Wie könnte man daraus Schatten «bauen» ?**

Shadow Maps



Shadow Maps

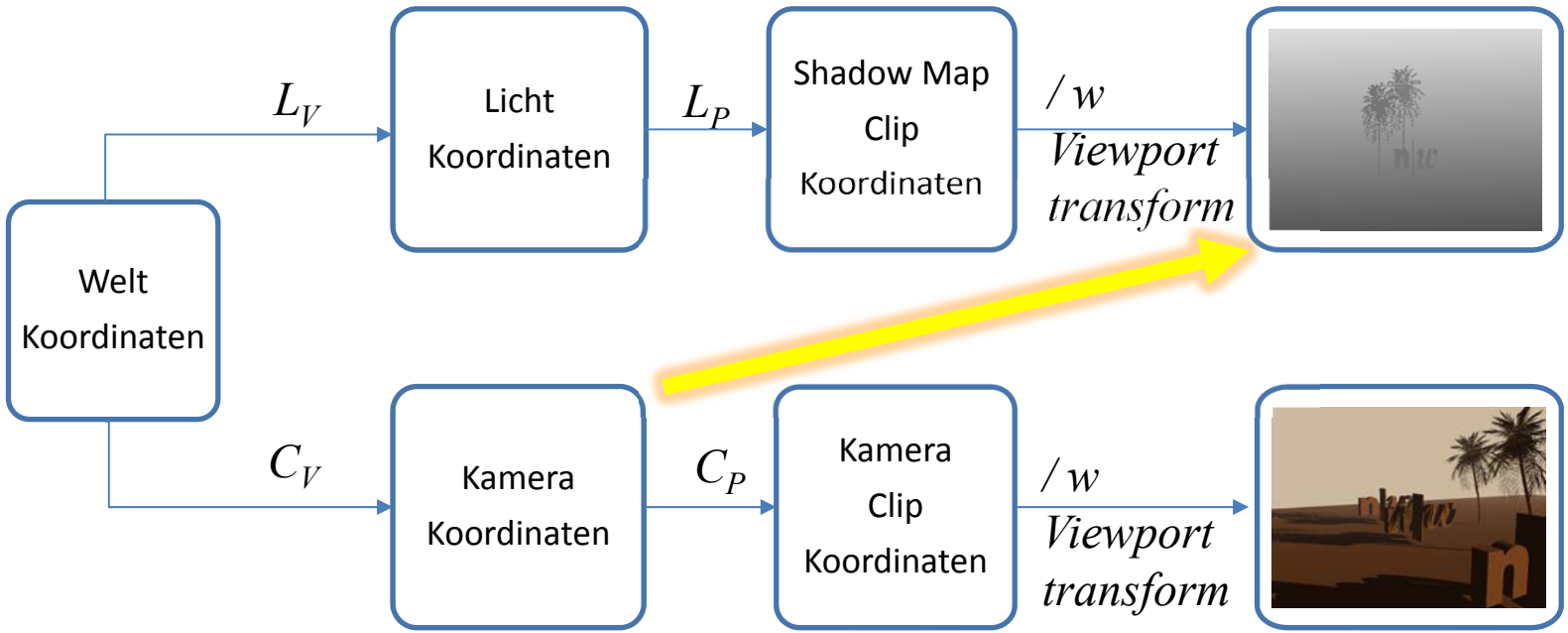
- Zwei-Schritt-Verfahren

1. Erzeugen der Shadow Map
2. Verwenden der Shadow Map in der Szene um Fragmente im Schatten abzudunkeln

Erzeugen der Shadow Map

- Rendern der Szene aus der Lichtrichtung
- Frustum muss Kamerafrustum vollständig beinhalten
- Nur Depth-Buffer von Interesse – nur Geometrie ohne Materialen und Beleuchtung etc. zeichnen
- Front-Face Culling und / oder Polygon Offset verwenden um Artefakte auf den Beleuchteten Teilen zu vermeiden

Anwenden der Shadow Map auf die Szene - Koordinatensysteme



L_V Licht View Matrix

L_P Licht Projektions Matrix

C_V Kamera View Matrix

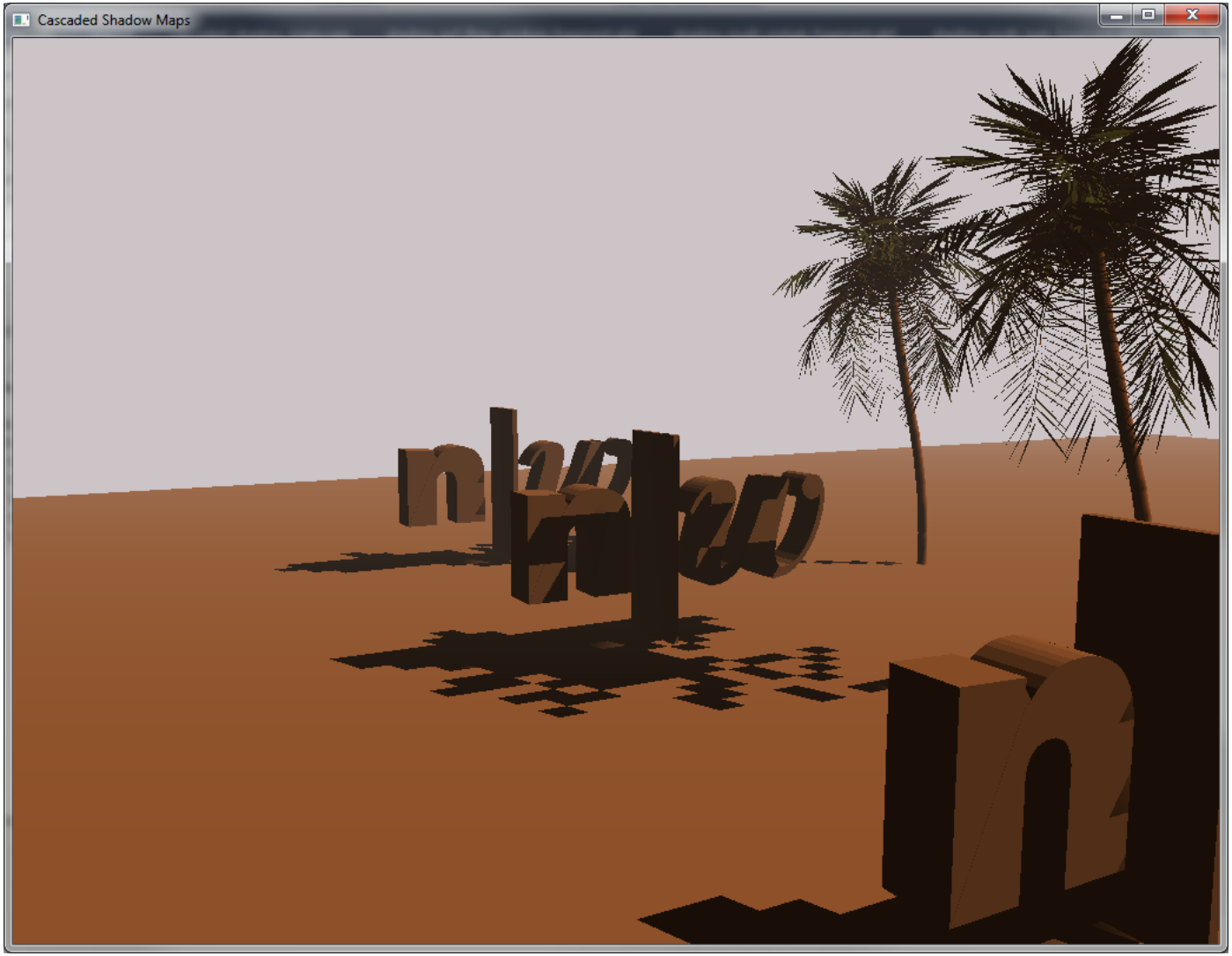
C_P Kamera Projektions Matrix

Anwenden der Shadow Map auf die Szene - Shadow Map auslesen

- Im Fragment Shader müssen wir für Kamera Koordinaten die Textur-Koordinate in der Shadow Map finden

1. Von Kamera- zu Welt-Koordinaten
   → $\boldsymbol{w = C_V^{-1} * c}$
2. Von Welt-Koordinaten zu projizierten Licht Koordinaten
   → $\boldsymbol{l = L_P * L_V * w}$
3. Da $\boldsymbol{l}$ zwischen [-1..1] liegt müssen wir sie noch auf [0..1] skalieren und verschieben um Texturkoordinaten zu bekommen

Anwenden der Shadow Map auf die Szene - Resultat



Shadow Map
Vor- und Nachteile

- Wie können wir das Verfahren verbessern?

- «Viel hilft viel»

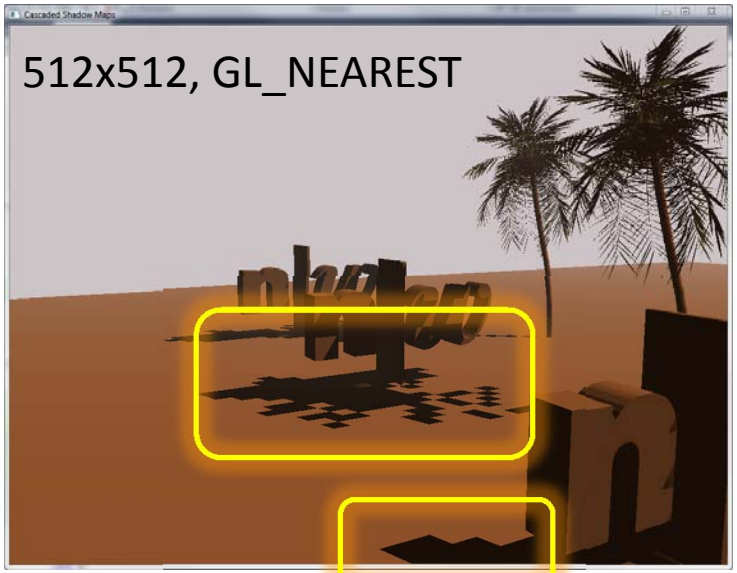
  → Shadow Map Auflösung vergrössern

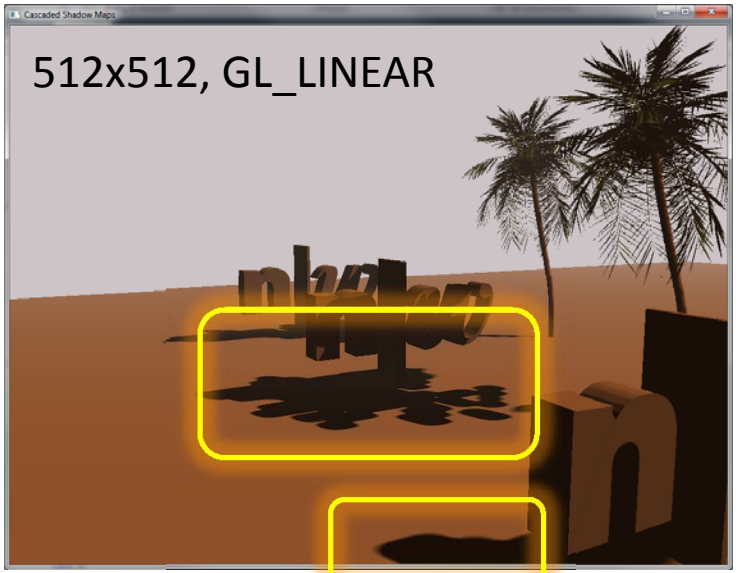
- Unschöne Ränder:

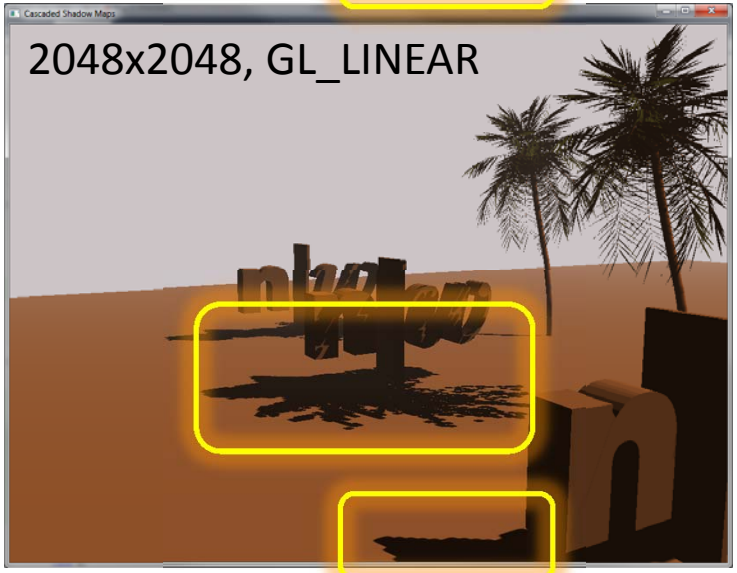
  →Weiche Schatten «Penumbra»

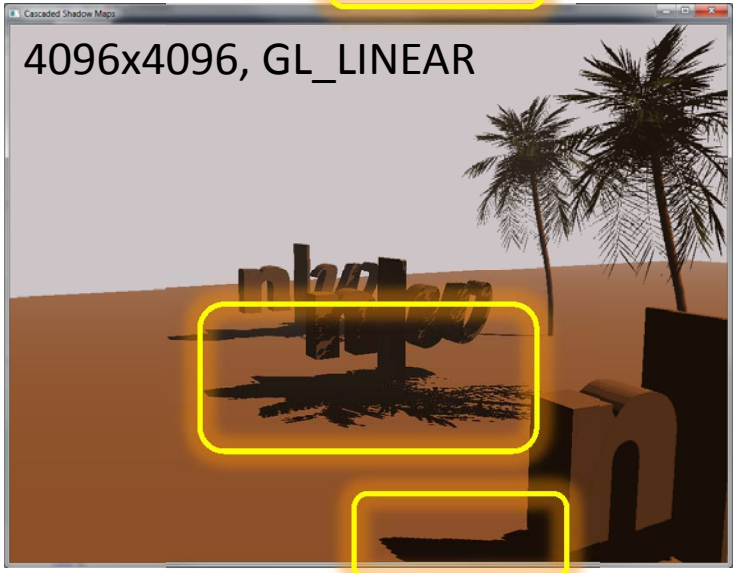
  →Filtern

Shadow Map
Vor- und Nachteile









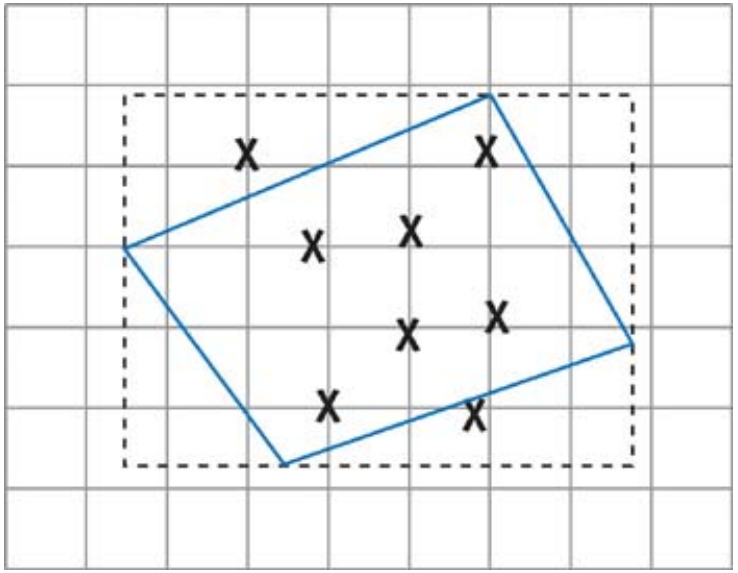
Shadow Map
Vor- und Nachteile

- Einfaches Prinzip
- Keine Geometrie Daten nötig (funktioniert z.B. auch mit Alphamasken)
- Schnell (Szene muss nur pro Lichtquelle zusätzlich einmal gezeichnet werden)

- Ungeeignet für grosse Szenen (immense Shadow Map für gute Auflösung)
- Unschöne Ränder

Percentage Closer Filtering

- Transformieren einer Region im Viewport in die Shadow Map
- Zufälliges Auslesen (Sampling) der Shadow Map
- Verschattung = Anteil der Samples die in der Region liegen
- Aufwand steigt mit Anzahl der Samples



Percentage Closer Filtering

1x1 Region

15x15 Region

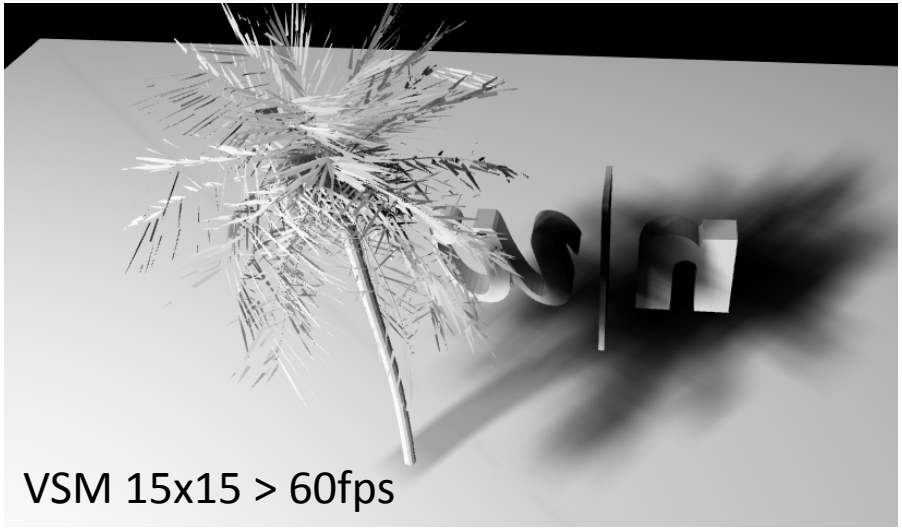
Variance Shadow Maps

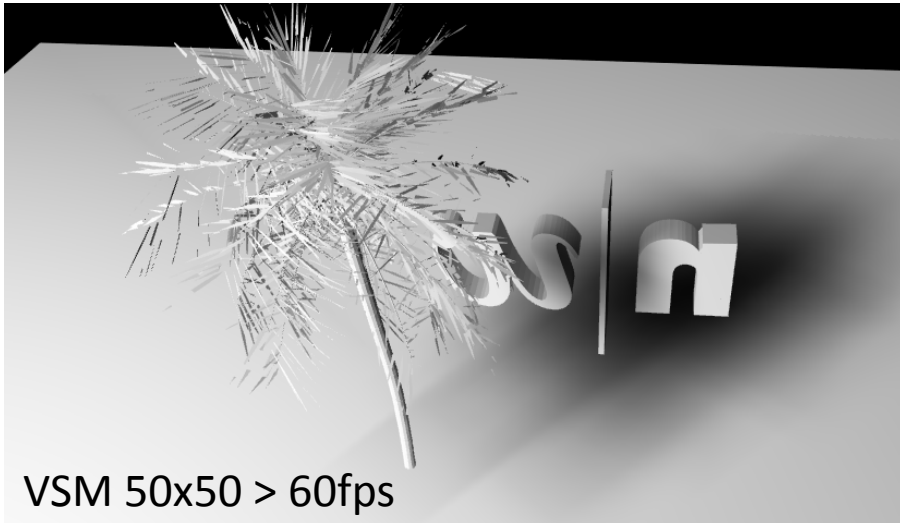
- Idee: Multisampling vermeiden, Filter der GPU verwenden
- Wie bei PCF interessiert uns der Zusammenhang der Shadow Map Werte einer Region und des Tiefenwerte des Fragments das wir zeichnen (t)
- Statt Shadow Map Anteile (PCF) betrachtet man die Verteilung der Tiefenwerte
- Die Tschebyscheff-Ungleichung liefert uns die Wahrscheinlichkeit bei bekanntem Mittelwert und Varianz ob ein Wert (t) auftritt oder nicht

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

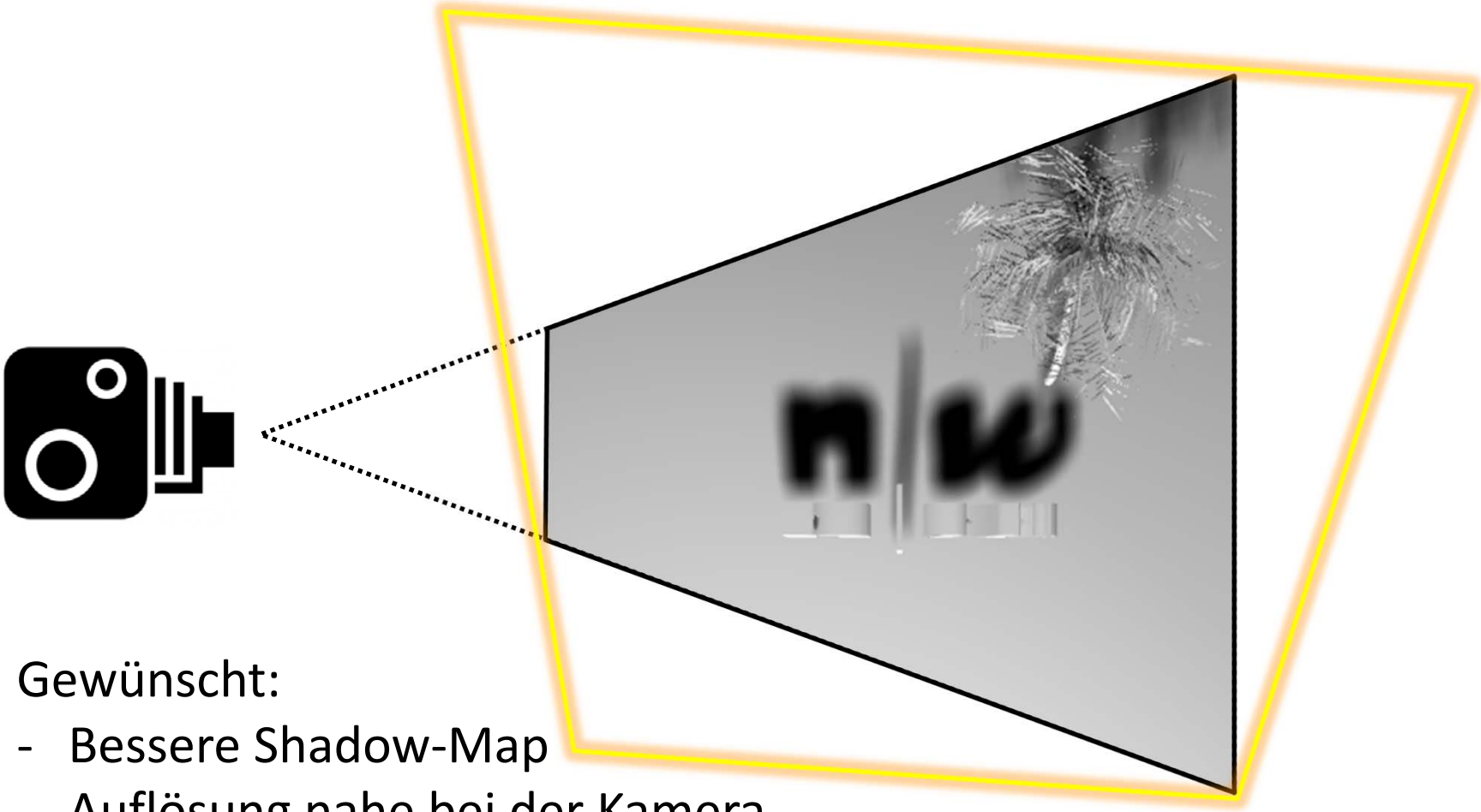
- Mittelwert: $\mu = E(x)$ - Berechnung mit GPU: Mip-Mapping, Blur, GLSL Shader, …
- Varianz: $\sigma^2 = E(x^2) - E(x)^2$
- Dazu speichern wir (x, x^2) z.B. in 2 Farbkanälen einer Float Textur

Variance Shadow Maps





Cascade Shadow Maps



Gewünscht:

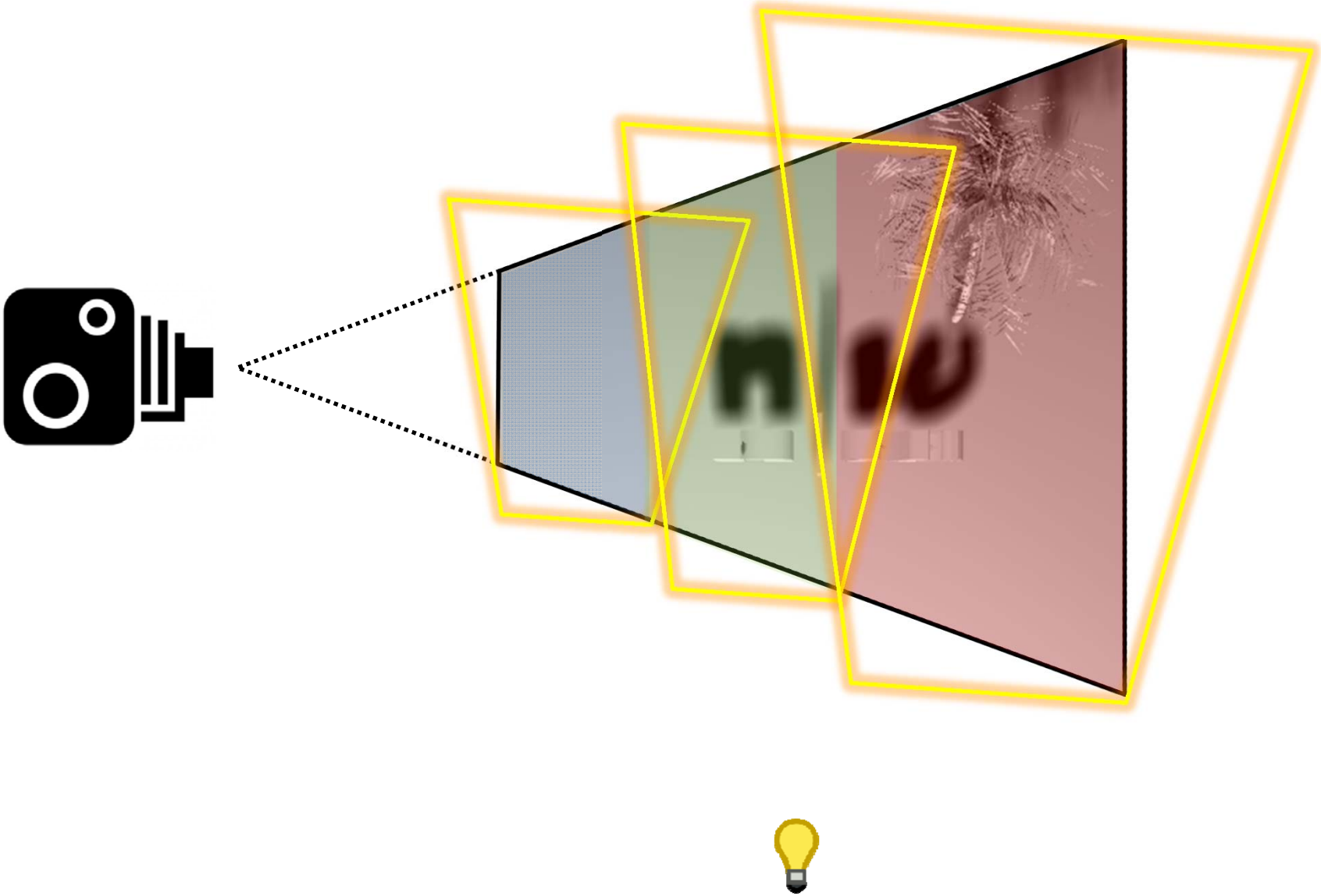
- Bessere Shadow-Map Auflösung nahe bei der Kamera
- Bessere Ausnutzung des Shadow Map Frustum

Cascade Shadow Maps

- Idee: Mehrere Licht-Frusta optimieren für aktuelle Kameraposition

- Ähnliche Verfahren: Perspective Shadow Maps, Light-Space Perspective Shadow Maps

Cascade Shadow Maps



Cascade Shadow Maps

Die Techniken im Vergleich

	Vorteile	**Nachteile**
Shadow Volumes	• Präzise, geometrische Schatten • Unabhängig von der Abmessung der Szene	• Braucht Szenen Geometrie (funktioniert nicht Alphamasken) • Nur begrenzt brauchbar für Echtzeitanwendungen
Shadow Maps	• Einfaches Prinzip • Keine Geometrie Daten nötig (funktioniert z.B. auch mit Alphamasken) • Schnell (Szene muss nur einmal zusätzlich pro Lichtquelle gezeichnet werden)	• Ungeeignet für grosse Szenen (Auflösung der Shadow Map) • Unschöne (scharfe) Ränder
Variance Shadow Maps	• Ähnlich gute Filterung wie PCF • GPU verwenden für Statistik • Effizient	• Mehrkanal Float32 Texturen (wegen $Depth^2$) → sind z.B. Extensions in OpenGL ES
Cascade Shadow Maps	• Kamera-abhängige Auflösung • Sehr gute Resultate auch bei grossen Szenen • Kann einfach mit Filterung kombiniert werden	• Idealerweise implementiert mit 3D Texturven • Szene muss pro Shadow-Map neu gezeichnet werden
Photonmapping, Raytracing, Radiosity, …	• Beste bekannte Verfahren für Global Illumination • Viele Lichteffekte (nicht nur Schatten) werden miteinbezogen	• Sehr Rechenaufwendig • Noch nicht brauchbar für Echtzeitanwendungen

Referenzen

- Wikipedia
- GPU Gems (“Part II: Lighting and Shadows”)
  - http://http.developer.nvidia.com/GPUGems/gpugems_part02.html
- Shadow Volumes:
  - https://developer.nvidia.com/content/gpu-gems-3-chapter-11-efficient-and-robust-shadow-volumes-using-hierarchical-occlusion
- Shadow Mapping:
  - Lance Williams “Casting Curved Shadows on Curved on Curved Surfaces” (http://design.osu.edu/carlson/history/PDFs/shadowmaps.pdf)
- Variance Shadow Maps:
  - http://www.punkuser.net/vsm/vsm_paper.pdf
- Cascade Shadow Maps + Filtering:
  - http://developer.download.nvidia.com/SDK/10/opengl/src/cascaded_shadow_maps/doc/cascaded_shadow_maps.pdf