# Interaktive Computergrafik

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

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

# Schatten

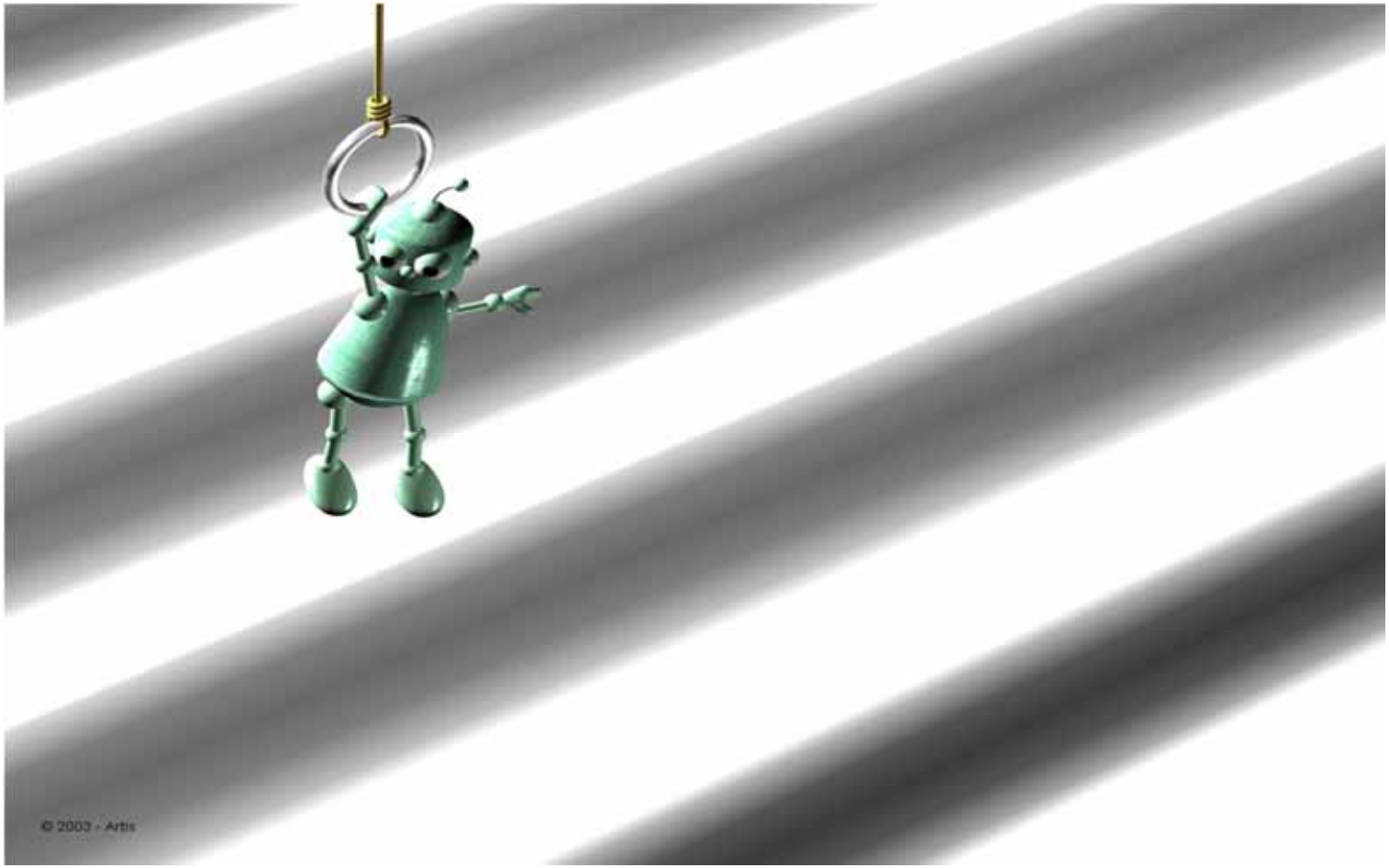
- wichtiger Aspekt für die Wahrnehmung
  - wie ist die räumliche Anordnung von Objekten?
  - woher kommt das Licht?

# Schatten

- vermittelt zusätzliche Informationen über Objekte, die Schatten werfen…

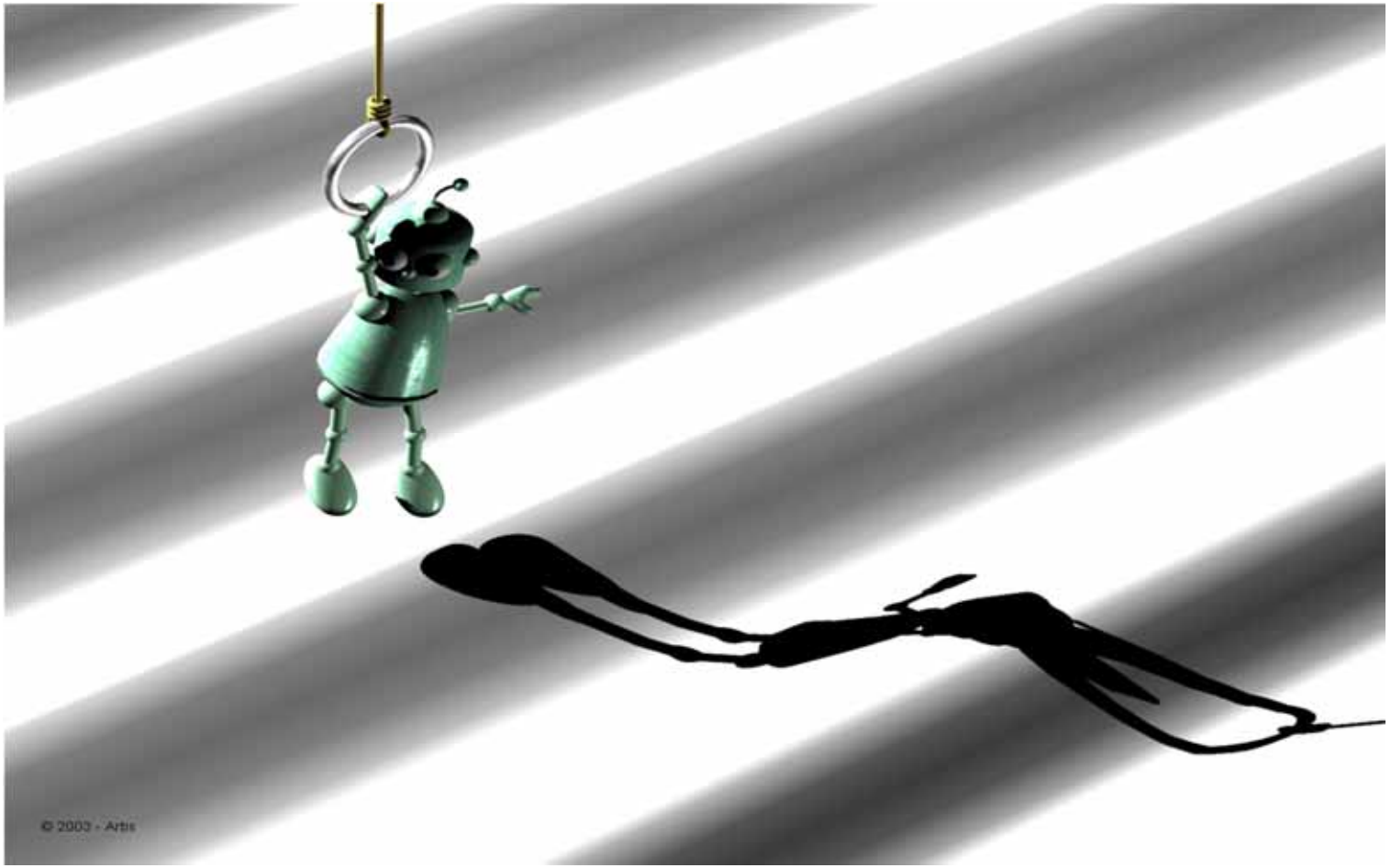
# Schatten

- … und Informationen über Objekte, auf die Schatten geworfen wird…



# Schatten

- … und Informationen über Objekte, auf die Schatten geworfen wird…



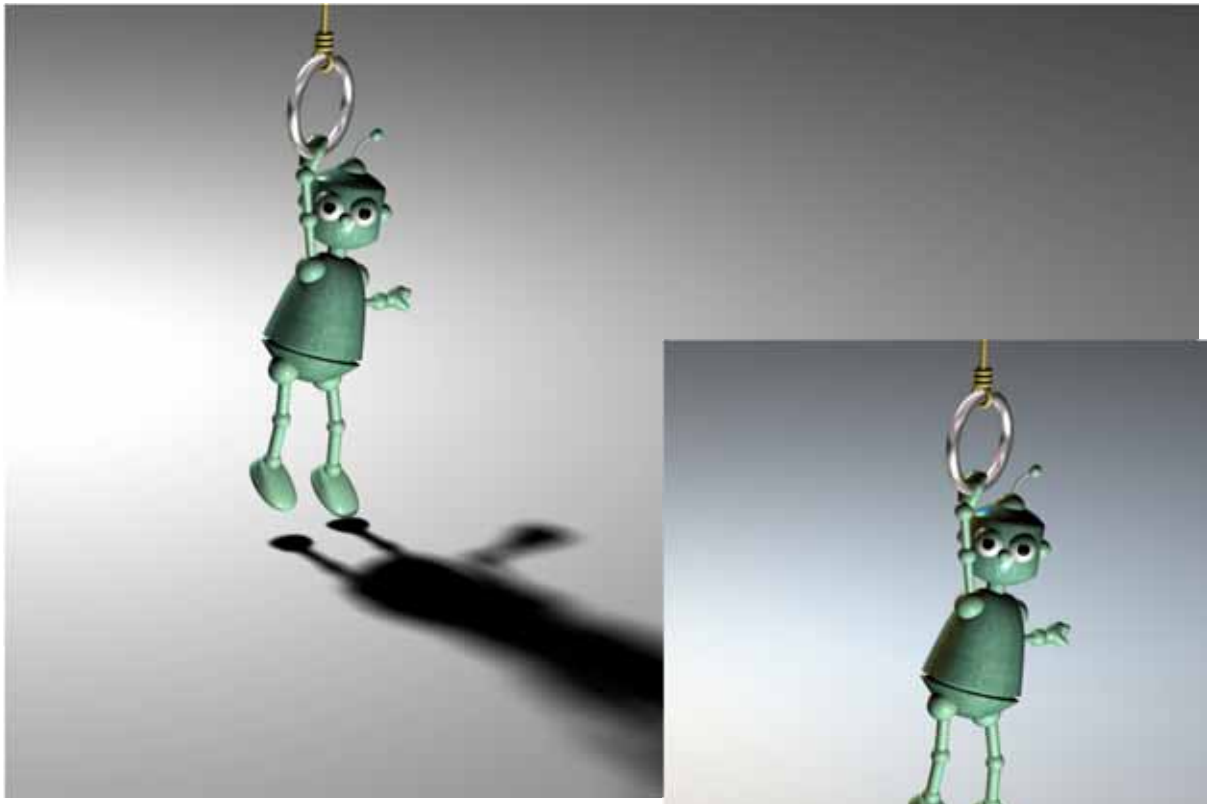
# Schatten

- … und Informationen über die Lichtquellen:
  - Punktlichtquelle: harte Schatten
    - Shadow Volumes [Crow77]
    - Shadow Maps [Williams78]

  - Flächenlichtquelle
    - Ray Casting und Sampling (u.a. mit Voxelisierung)
    - spezielle Echtzeitverfahren

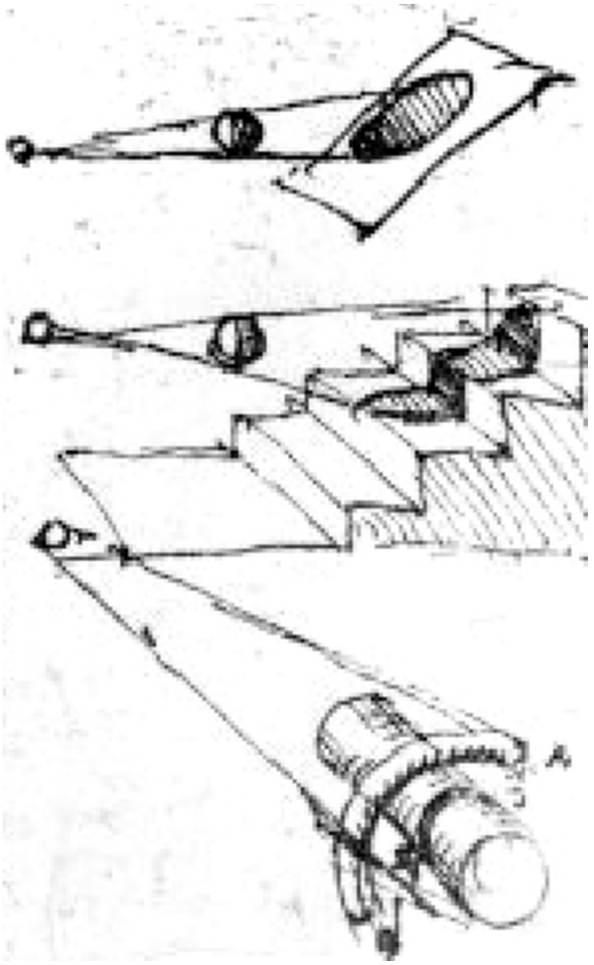
# Schatten

- … und Informationen über die Lichtquellen:

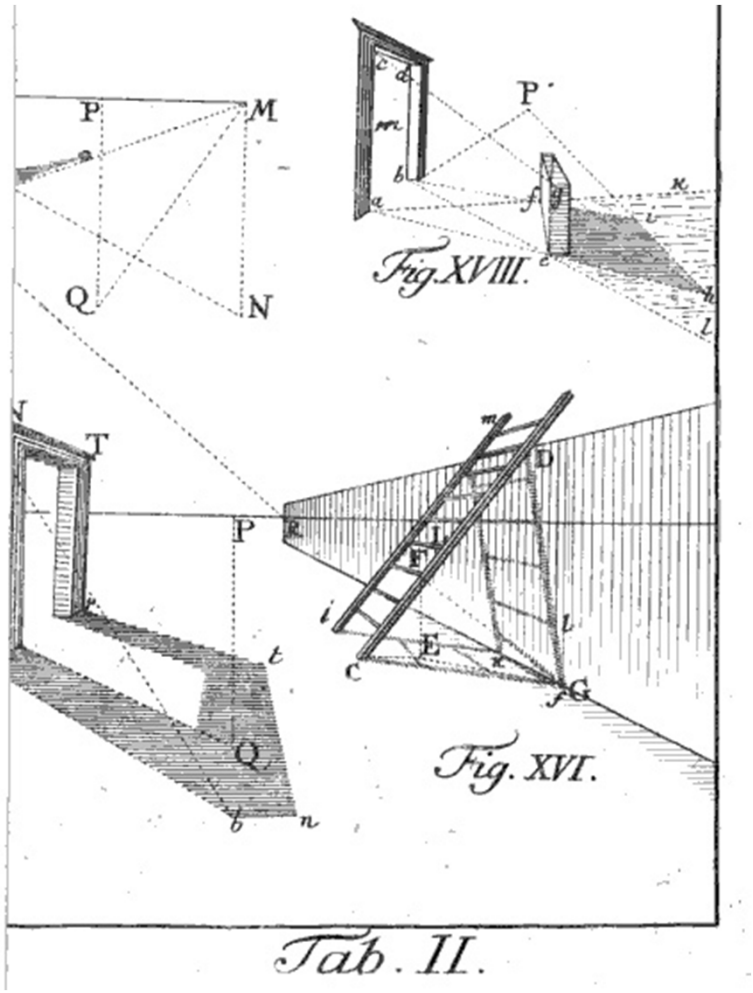


Bilder:
A survey of real-time soft-shadow algorithms, Hasenfratz et al.
http://hal.inria.fr/inria-00441603

# Schatten

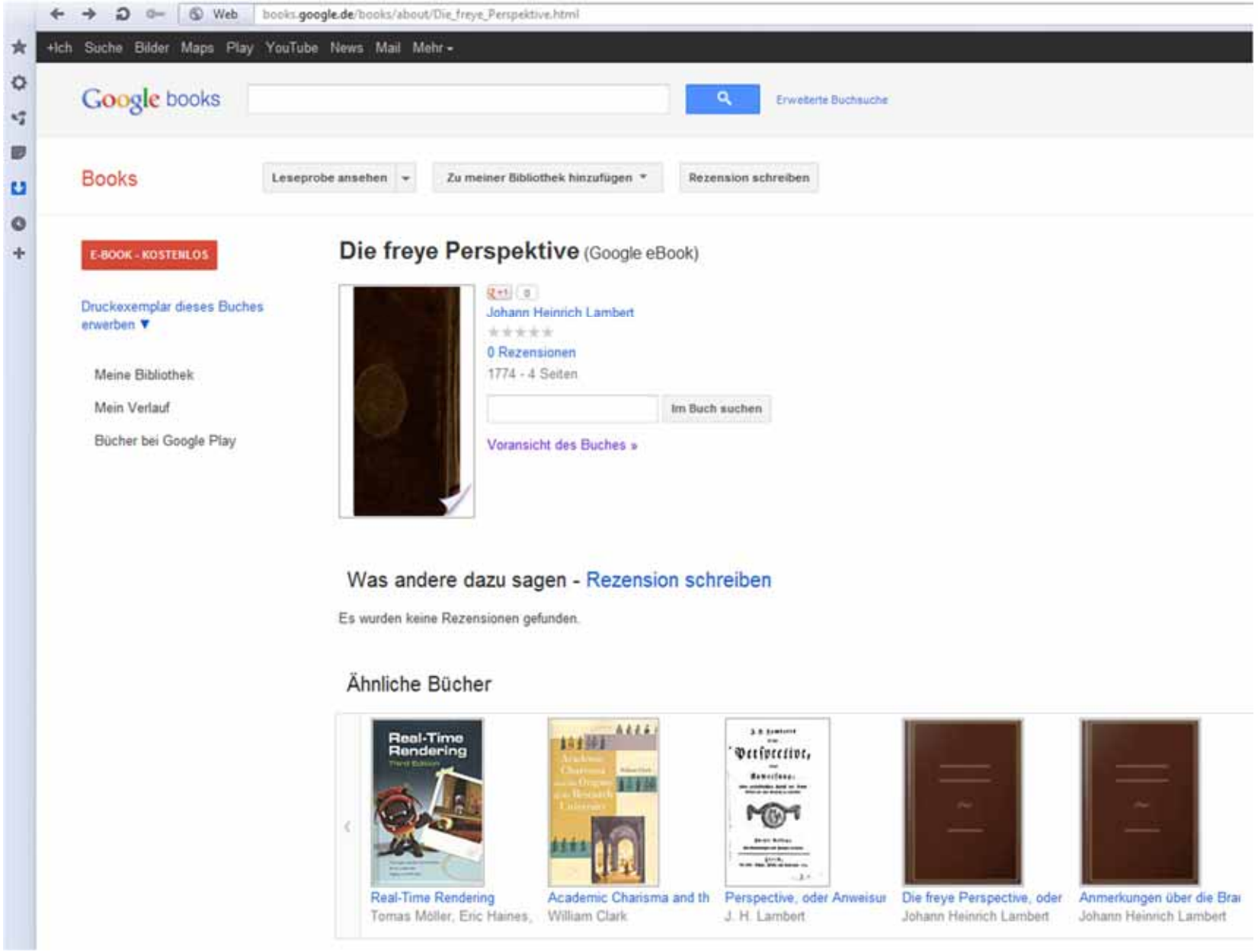


Leonardo Da Vinci, Trattato della pittura
(Codex Urbinas), 1490
Traktat über u.a. technische Probleme der Malerei



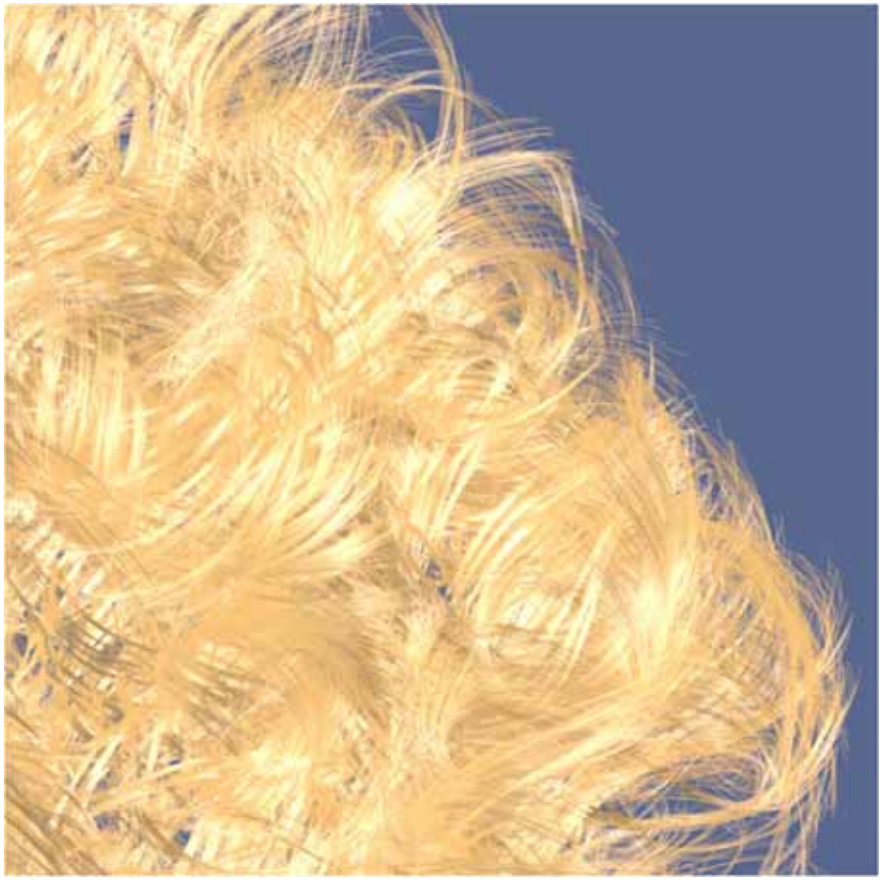

Johann Heinrich Lambert,
Die freye Perspektive, 1759

# Schatten


Web
books.google.de/books/about/Die_freye_Perspektive.html
+Ich Suche Bilder Maps Play YouTube News Mail Mehr
Google books
Erweiterte Buchsuche
Books
Leseprobe ansehen
Zu meiner Bibliothek hinzufügen
Rezension schreiben
E-BOOK - KOSTENLOS
Druckexemplar dieses Buches erwerben
Meine Bibliothek
Mein Verlauf
Bücher bei Google Play
Die freye Perspektive (Google eBook)
Johann Heinrich Lambert
0 Rezensionen
1774 - 4 Seiten
Im Buch suchen
Voransicht des Buches »
Was andere dazu sagen - Rezension schreiben
Es wurden keine Rezensionen gefunden.
Ähnliche Bücher
Real-Time Rendering
Tomas Möller, Eric Haines,
Academic Charisma and th
William Clark
Perspective, oder Anweisur
J. H. Lambert
Die freye Perspective, oder
Johann Heinrich Lambert
Anmerkungen über die Brai
Johann Heinrich Lambert

# Schatten

- es existieren ebenfalls eine Reihe spezieller Verfahren für „volumetrische“ Objekte, wie z.B. Haare, Rauch, …



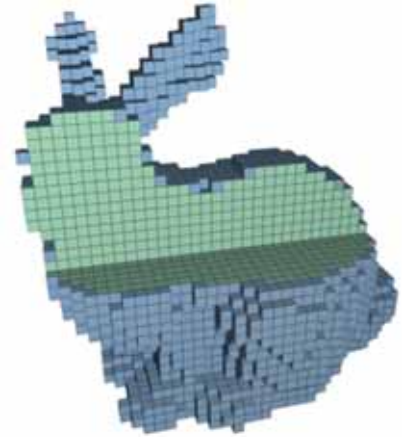
# Schatten

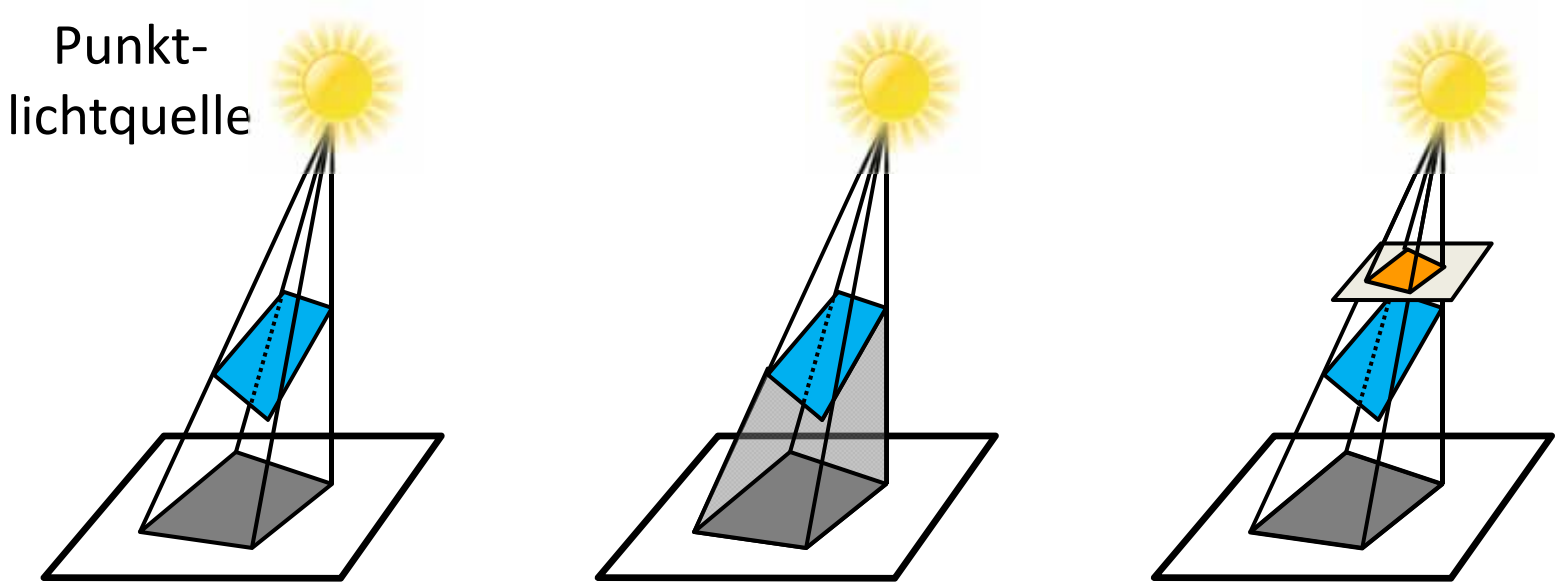
- Schattenberechnung ist ein „globales Problem“:
  - man muss die gesamte Szene kennen um festzustellen, ob ein Punkt im Schatten liegt, also ein „Schattenstrahl“ irgendeine andere Fläche schneidet

- in der Grafik-Pipeline werden alle Primitive unabhängig voneinander (Vorstellung: „nacheinander“) gezeichnet
  - offensichtlich: jedes Primitiv kann Schatten auf jedes vorherige oder spätere Primitiv werfen und umgekehrt
  - diese Information steht während der Verarbeitung eines Primitivs nicht zur Verfügung
  - deswegen kann es kein `glEnable( GL_SHADOWS );` geben

# Klassifikation Schattenverfahren

- **vorberechnete Schatten/Beleuchtung** (sog. Light Maps)
- **projektive Schatten**: nur für Schatten auf Ebenen, keine Selbstverschattung
- **Schattenvolumen**: Objektraum-Verfahren
- **Shadow Maps**: Bildraum-Verfahren
- **Ray Casting / Ray Marching**: Voxelisierung






Projektive Schatten | Schattenvolumen | Shadow Maps

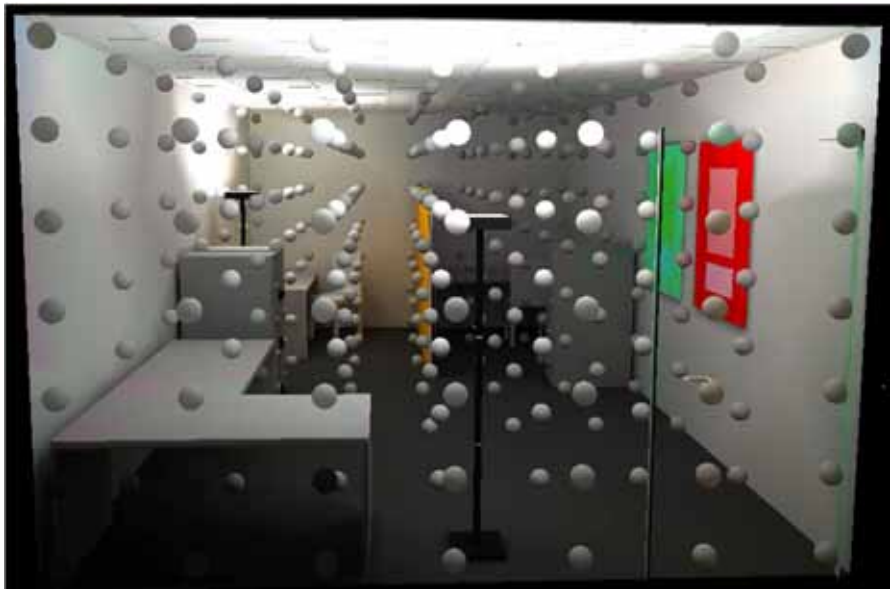
# Vorberechnete Schatten

## Light Maps: Texturen mit vorberechneter Beleuchtung und Schatten

- benötigt eine Parametrisierung (z.B. einen Texturatlas)
- nur für statische Szenen möglich (oder eine begrenzte Variation)
- vorberechnet mit Raytracing, Path Tracing, …
- Light Maps meist niedrig(er) aufgelöst
  - Detail durch andere Texturen (diffuse Farbe, Bump/Gloss Map, …)
- oft wird auch einfallendes Licht richtungsabhängig gespeichert (u.U. sogar in einer 3D Textur → „Irradiance Volume“)


# Irradiance Volumes

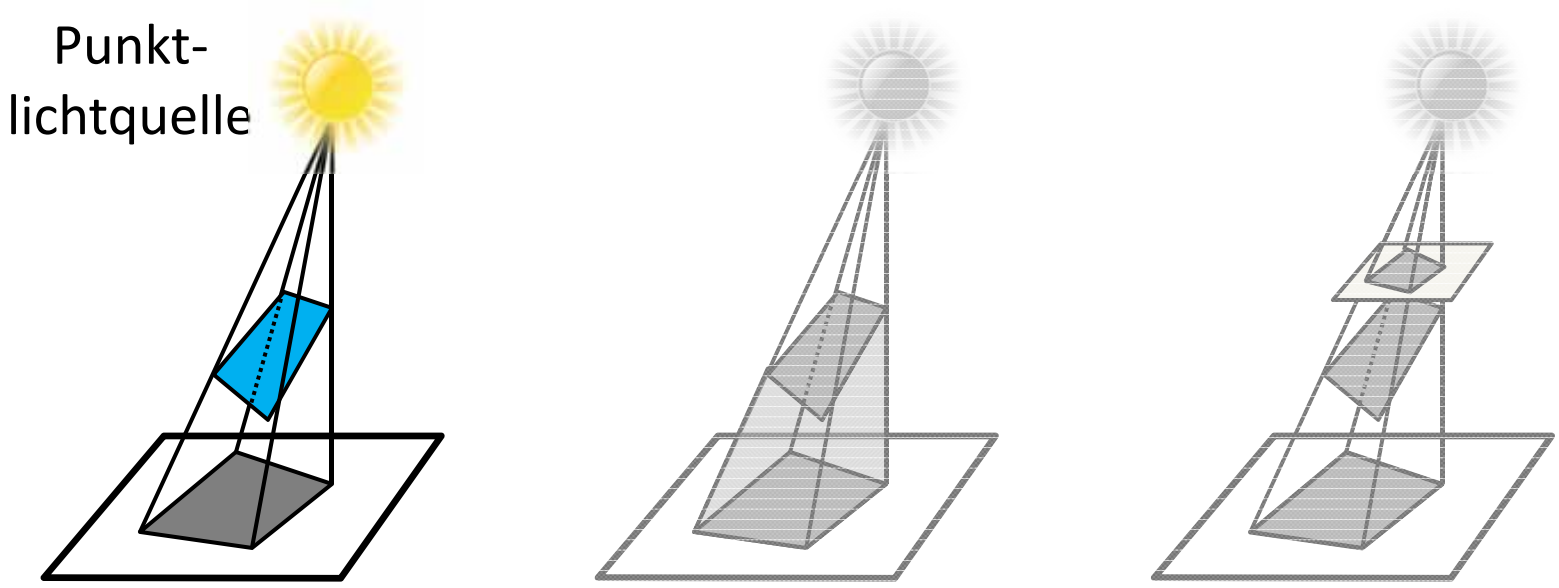
- für viele Punkte im Raum (auf regelmäßigen Gitter oder Octree) wird das einfallende Licht (Irradiance) gespeichert
- Lookup für eine Oberfläche bei $\boldsymbol{x}$ mit Normale $\boldsymbol{n}$
  - lokalisiere die nahsten Gitterpunkte (z.B. für trilineare Interpolation)
  - lese dort Irradiance für Richtung $\boldsymbol{n}$ aus
  - interpoliere Irradiance/Farbe



Bilder: The Irradiance Volume, Gene S. Greger, MSc Thesis

# Klassifikation Schattenverfahren

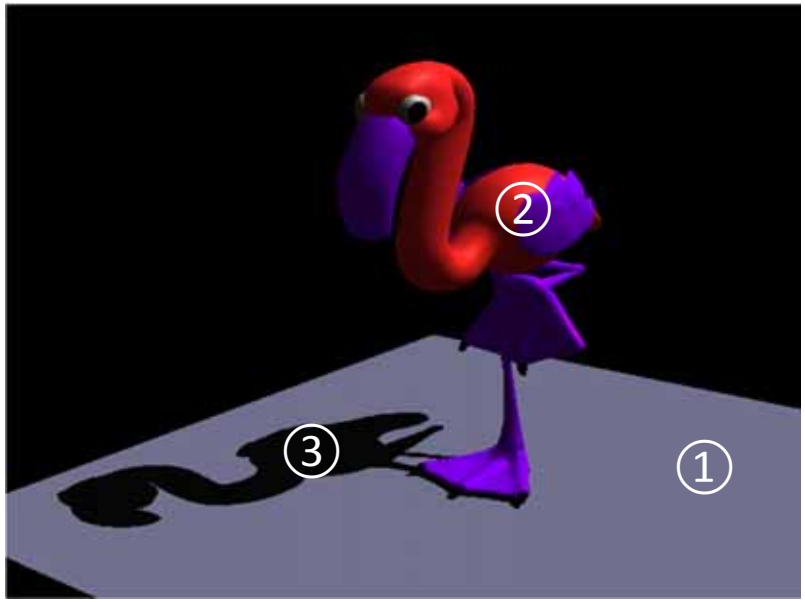
- **vorberechnete Schatten/Beleuchtung** („Light Maps“)
- **projektive Schatten**: nur für Schatten auf Ebenen, keine Selbstverschattung
- **Schattenvolumen**: Objektraum-Verfahren
- **Shadow Maps**: Bildraum-Verfahren




Projektive Schatten | Schattenvolumen | Shadow Maps

# Planare/Projektive Schatten

- Darstellung des Schattens eines Objektes auf einer Ebene
- Idee: Schatten ist eine Projektion des Objekts auf diese Ebene
  → zeichne Objekt mit entsprechender projektiver Abbildung
- Vorgehen
  - zeichne Boden ①
  - zeichne Objekt ②
  - setze Matrizen für die Projektion
  - zeichne das Objekt nochmals in schwarz ③

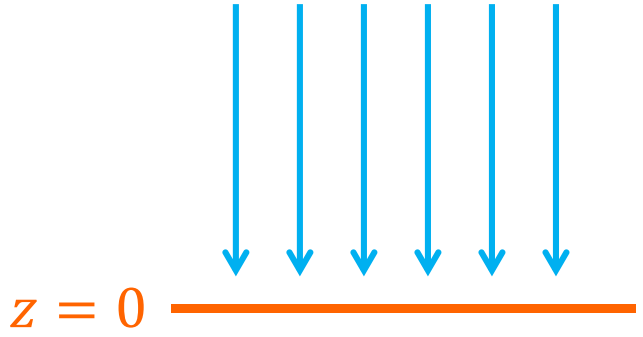


# Projektive Schatten

## Beispiel 1: paralleles Licht aus $z$-Richtung

- der Schatten soll auf der Ebene mit $z = 0$ dargestellt werden
- Projektion eines beliebigen Punktes auf die Ebene: $(x, y, z)^T \rightarrow (x, y, 0)^T$
- die entsprechende Projektionsmatrix für den Schatten ist

$$\begin{pmatrix} x_s \\ y_s \\ z_s \\ w_s \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$




- die Matrix wird nach der Modell- und vor der Kamera-Transformation angewendet

# Projektive Schatten

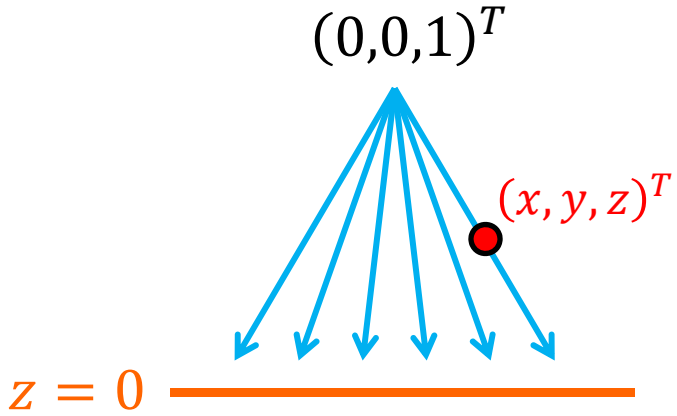
## Beispiel 2: Punktlichtquelle bei $(0,0,1)^T$

- Schatten wieder auf der Ebene E mit $z = 0$
- für die Projektion eines Punktes $(x, y, z)^T$ gilt:

$$(0,0,1)^T + t \cdot [(x, y, z)^T - (0,0,1)^T] \in \mathrm{E}$$
$$\Leftrightarrow 1 + t(z - 1) = 0$$
$$\Leftrightarrow t = -\frac{1}{z-1} = \frac{1}{1-z}$$

- also: $(x, y, z)^T \rightarrow \left(\frac{x}{1-z}, \frac{y}{1-z}, 0\right)^T$
- Abbildung mit folgender Matrix und homogenen Koordinaten:

$$\begin{pmatrix} x_s \\ y_s \\ z_s \\ w_s \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



# Berechnung der Projektionsmatrix

## Punktlichtquelle oBdA im Ursprung

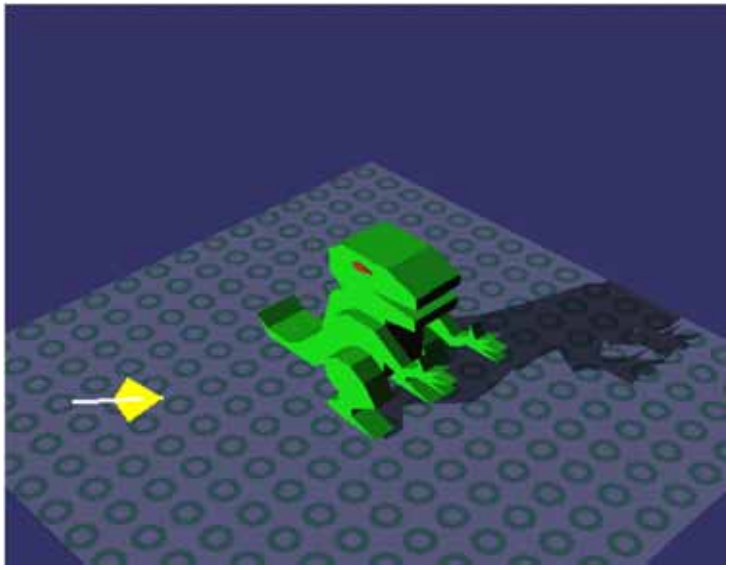
- Strahl durch Punkt $(x, y, z)^T$: $\alpha \cdot (x, y, z)^T$
- Schatten auf einer beliebigen Ebene $Ax + By + Cz + D = 0$
- Schnittpunkt Strahl-Ebene bei $\alpha = -\frac{D}{Ax+Dy+Cz}$
- Punkt in homogenen Koordinaten $(-Dx, -Dy, -Dz, Ax + By + Cz)$
- Transformationsmatrix

$$\begin{pmatrix} x_s \\ y_s \\ z_s \\ w_s \end{pmatrix} = \begin{pmatrix} -D & 0 & 0 & 0 \\ 0 & -D & 0 & 0 \\ 0 & 0 & -D & 0 \\ A & B & C & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

# Projektive Schatten

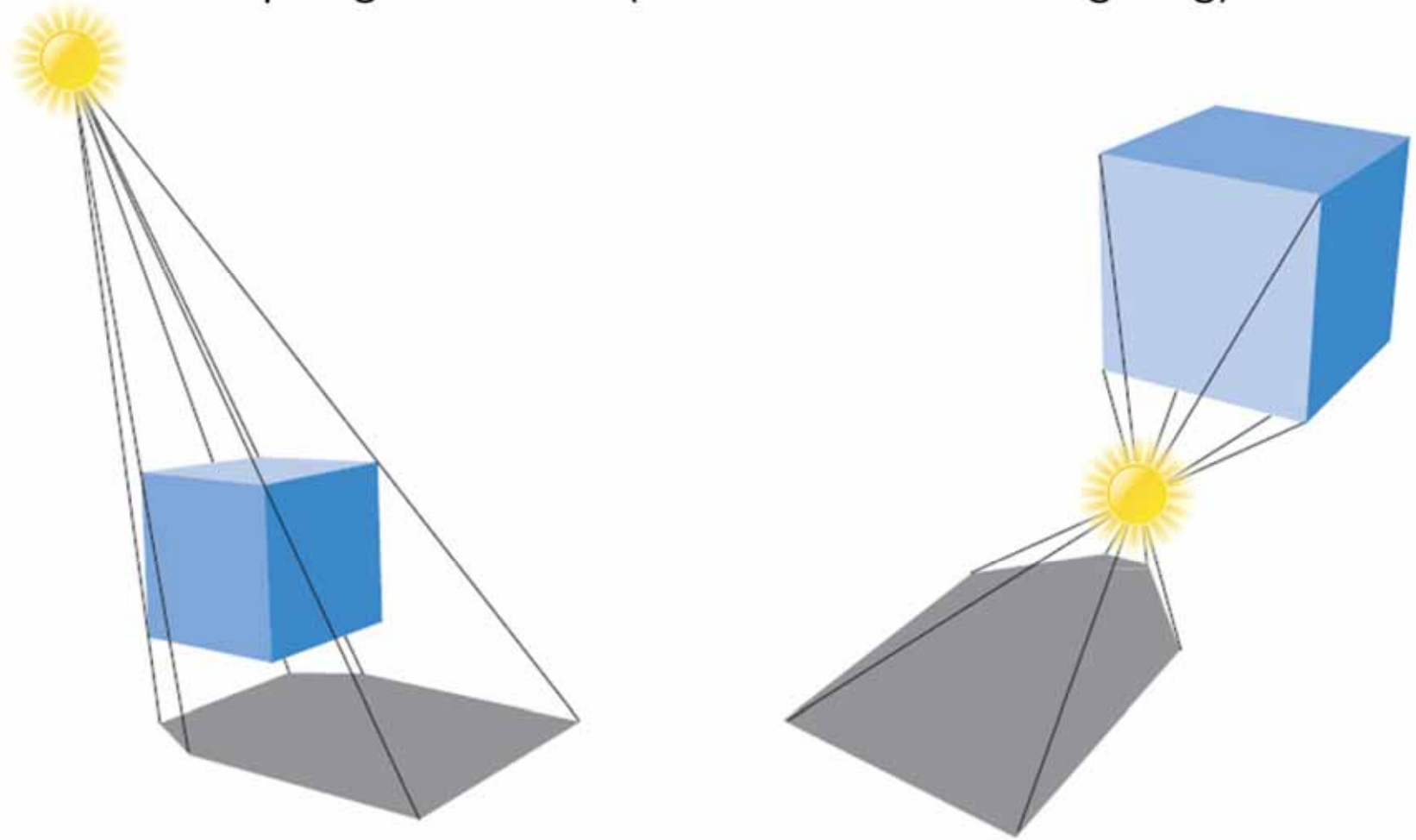
- Probleme:
  - Schatten außerhalb des Boden-Polygons
  - Z-Fighting: der Schatten auf der Ebene wird durch den Tiefentest „zufällig“

- Lösung:
  - zeichne Objekt
  - zeichne Boden, setze Stencil Wert 1 für alle Pixel
  - Tiefentest ausschalten
  - zeichne Schatten mit Stencil Test == 1



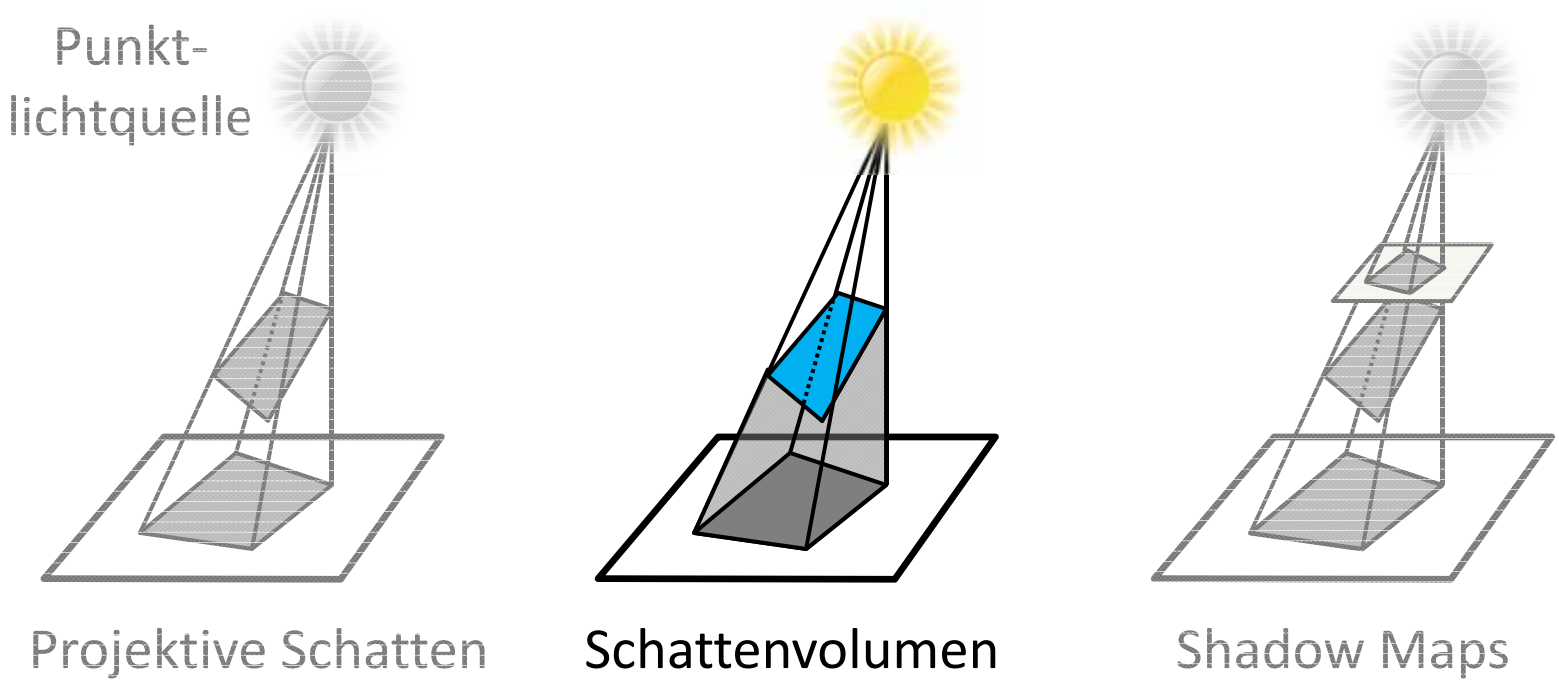
# Projektive Schatten

- Achtung: durch die Projektion kann der Schatten auch „hinter“ der Lichtquelle liegen
- außerdem: projektive Schatten nur bei großen planaren Schattenempfängern sinnvoll (daher: Praxisrelevanz gering)



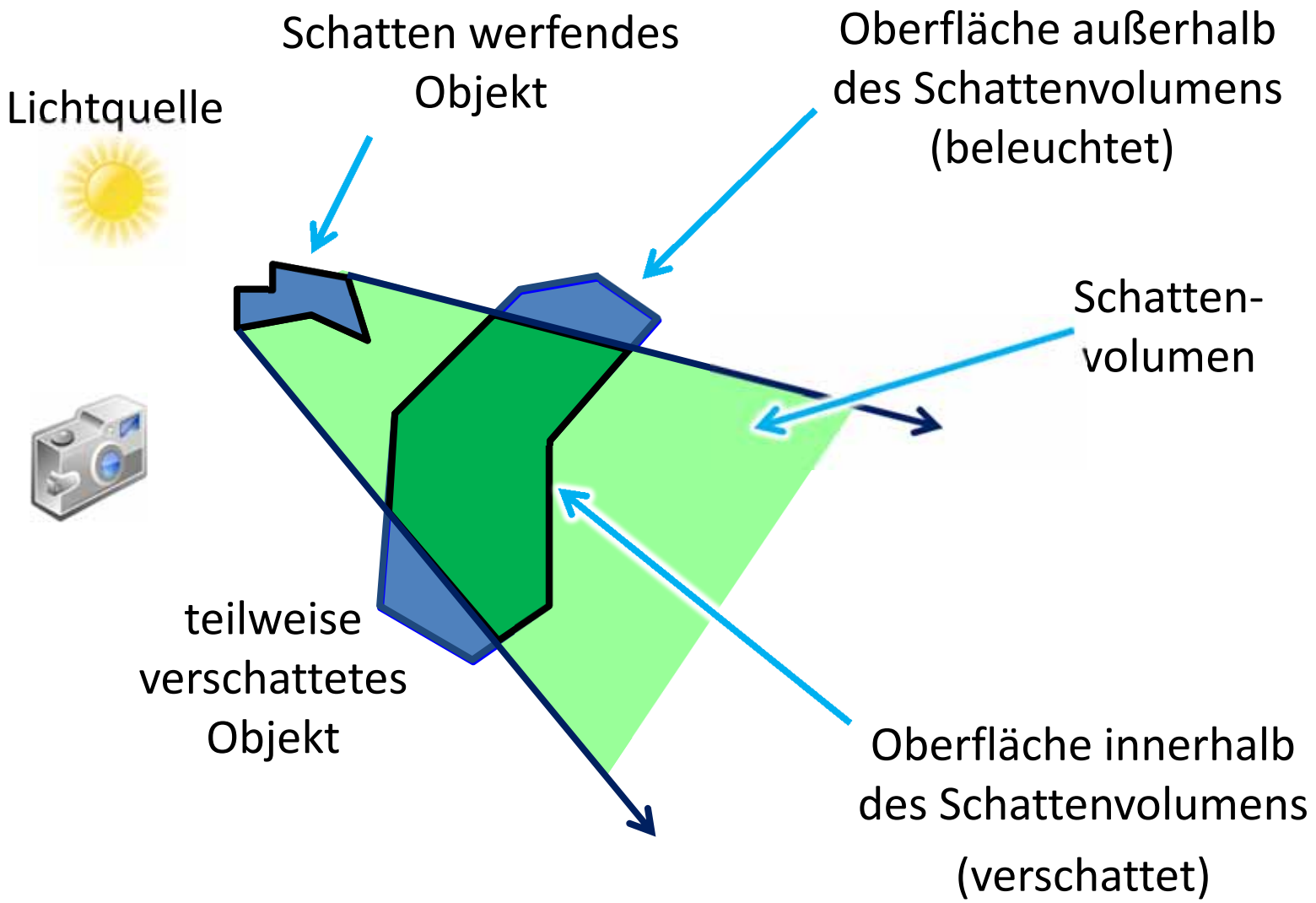
# Klassifikation Schattenverfahren

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



# Schattenvolumen

- ein Schattenvolumen umschließt den durch ein Objekt von einer Punktlichtquelle verschatteten Raum



# Idee der Schattenvolumen

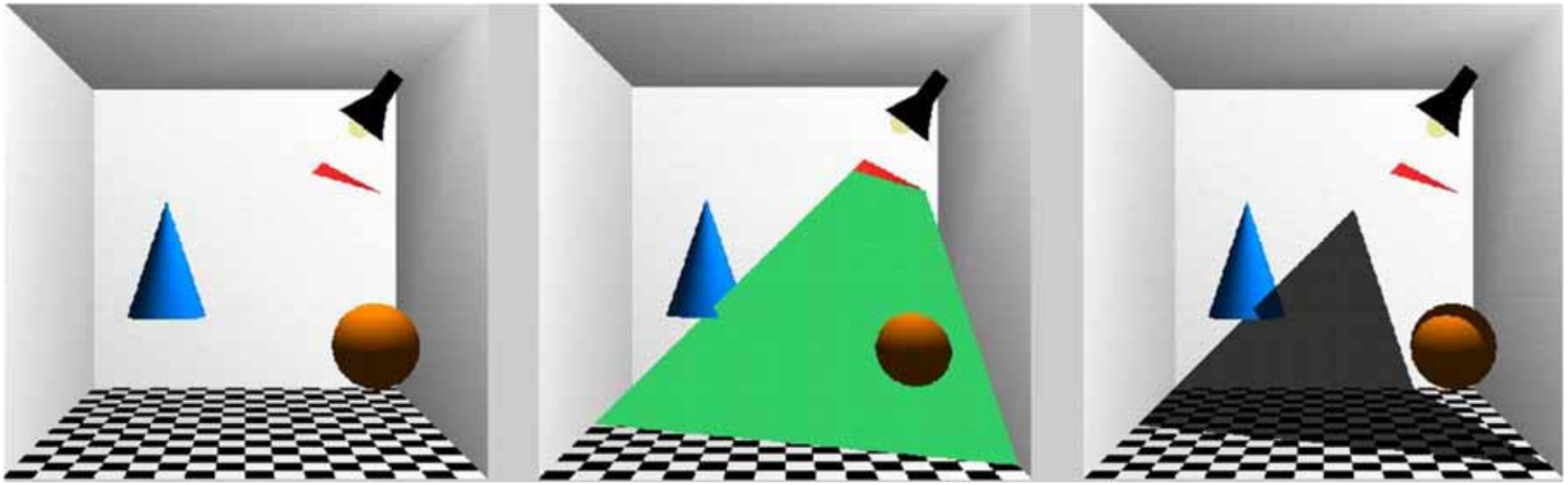
- bestimme zunächst die Schattenvolumen für alle Kombinationen von Objekten und (Punkt-)Lichtquellen
- stelle dann für jeden Pixel fest, ob er von einer LQ beleuchtet wird, oder sich in einem entsprechenden Schattenvolumen befindet
- es handelt sich um ein Objektraumverfahren: wir beschreiben die Oberfläche der Schattenvolumen geometrisch → akkurate Schatten

Schattenvolumen in Doom 3

# Schattenvolumen

## Grundidee

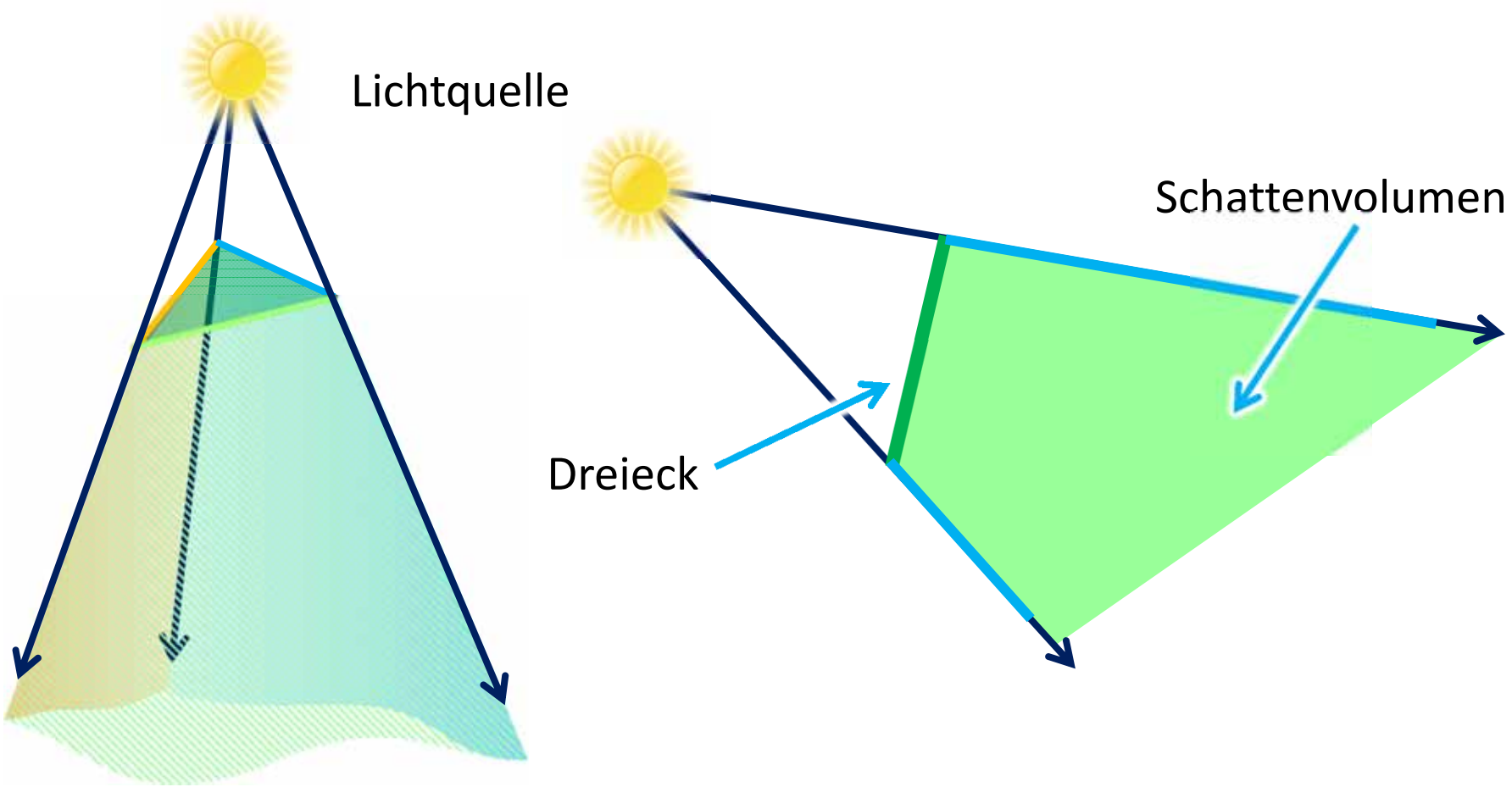
- für jedes Paar aus Lichtquelle und Fläche:
  bestimme das (unendlich große) Schattenvolumen durch Berechnung seiner **Begrenzungsflächen**
- im Prinzip kann man ein SV für jedes einzelne Dreieck erzeugen
- alle Punkte innerhalb des Schattenvolumen sind im Schatten



ohne Schatten | Schattenvolumen | mit Schatten

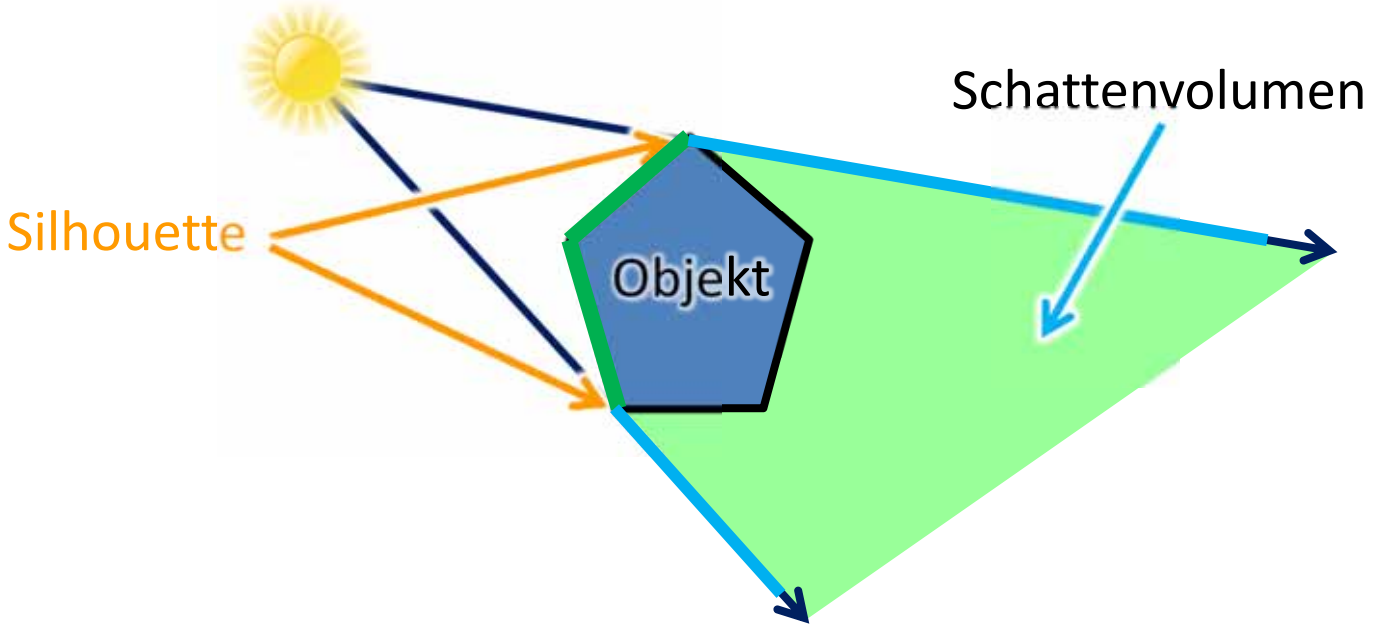
# Schattenvolumen eines Dreiecks

- die Oberfläche des Schattenvolumen eines einzelnen Dreiecks besteht
  - aus dem Dreieck selbst und
  - den Seitenflächen gebildet aus den Dreieckskanten extrudiert von der Lichtquelle weg



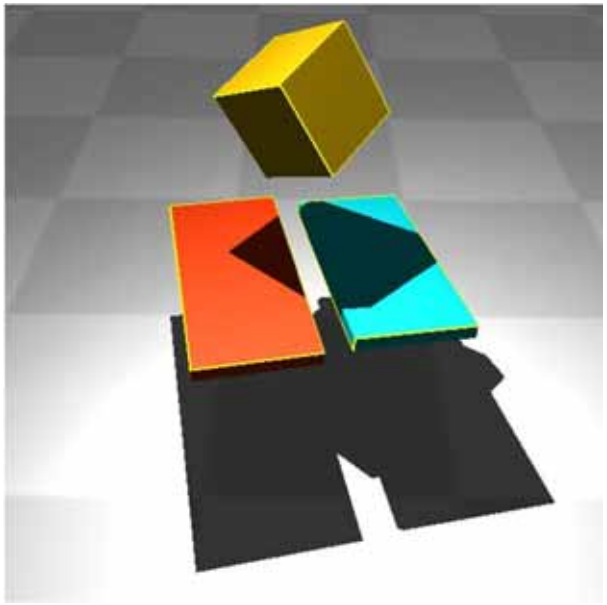
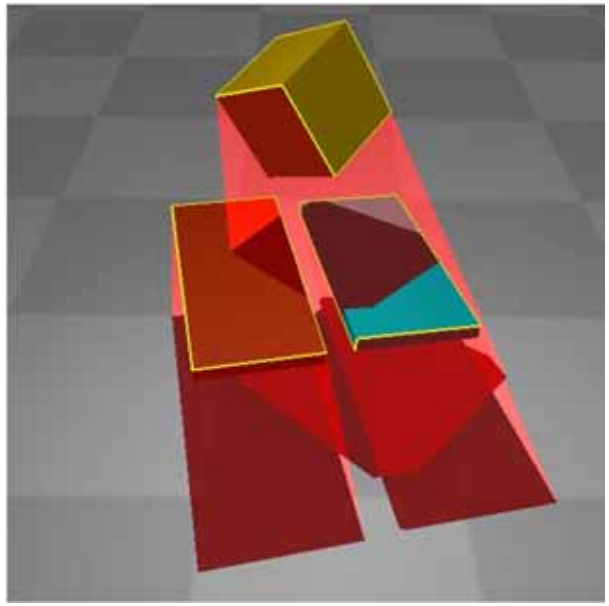
# Schattenvolumen eines Dreiecksnetzes

- die Oberfläche des Schattenvolumens eines **geschlossenen** Dreiecksnetzes besteht aus
  - Dreiecken, die zur Lichtquelle hin orientiert sind und
  - Seitenflächen gebildet aus der Objektsilhouette



# Schattenvolumen eines Dreiecksnetzes

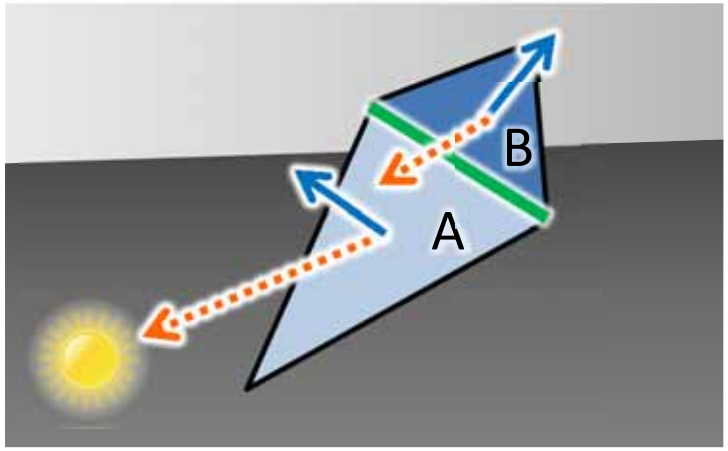
- die Oberfläche des Schattenvolumens eines **geschlossenen** Dreiecksnetzes besteht aus
  - Dreiecken, die zur Lichtquelle hin orientiert sind und
  - Seitenflächen gebildet aus der Objektsilhouette

# Schattenvolumen eines Dreiecksnetzes

## Bestimmung der Objektsilhouette

- Annahme: geschlossene Dreiecksnetze („watertight“, 2-manifold)
- zur Silhouette gehören Kanten von deren angrenzenden Dreiecken eines **zur Lichtquelle zeigt** und eines **von der Lichtquelle abgewandt** ist
- Vorgehen:
  - bestimme für jedes Dreieck, ob es zur LQ oder davon weg zeigt (Skalarprodukt zwischen der Dreiecksnormale und dem Vektor zur LQ)
  - betrachte jede Kante und ihre zwei benachbarten Dreiecke A und B:
    - zeigt A zur LQ und B zeigt weg (oder umgekehrt)
      → Kante ist Teil der Silhouette

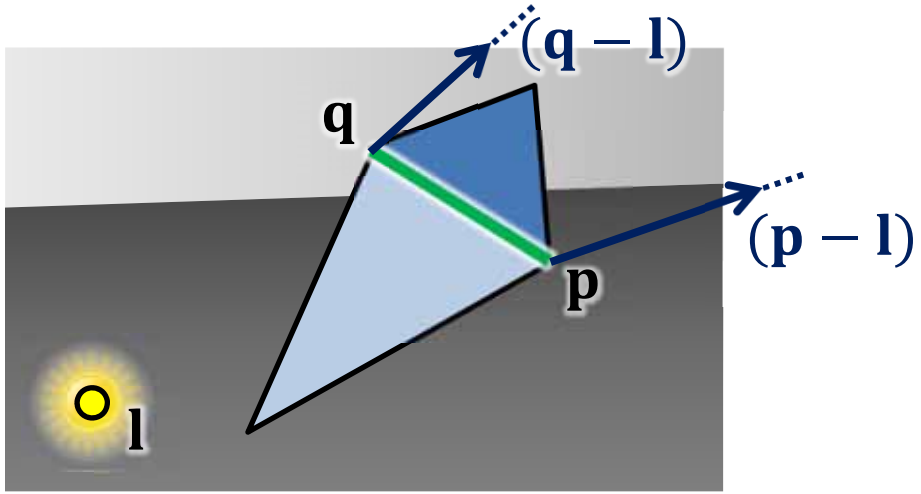


# Schattenvolumen eines Dreiecksnetzes

## Oberfläche des Schattenvolumens

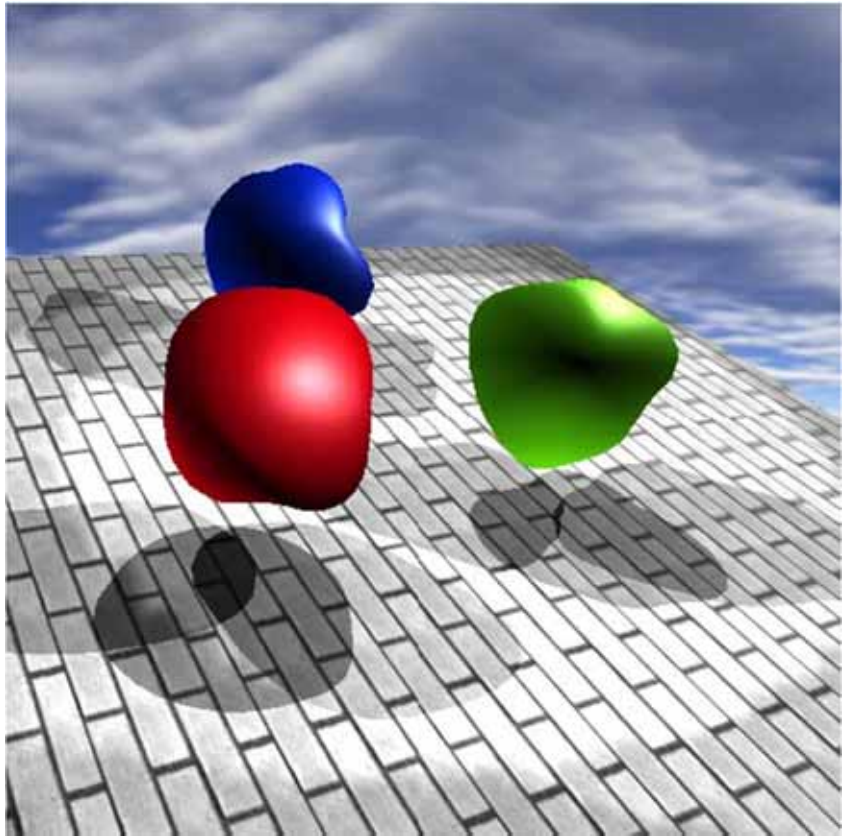
- sie die Kante $\overline{\mathbf{pq}}$ Teil der Silhouette
  - extrudiere $\overline{\mathbf{pq}}$ weg von der Lichtquelle ins Unendliche
  - dadurch entsteht eine **viereckige Seitenfläche des Schattenvolumen** mit den Eckpunkten $\mathbf{p}$, $\mathbf{p} + \infty \cdot (\mathbf{p} - \mathbf{l})$, $\mathbf{q} + \infty \cdot (\mathbf{q} - \mathbf{l})$ und $\mathbf{q}$
  - Punkte im Unendlichen durch homogene Koordinaten dargestellt

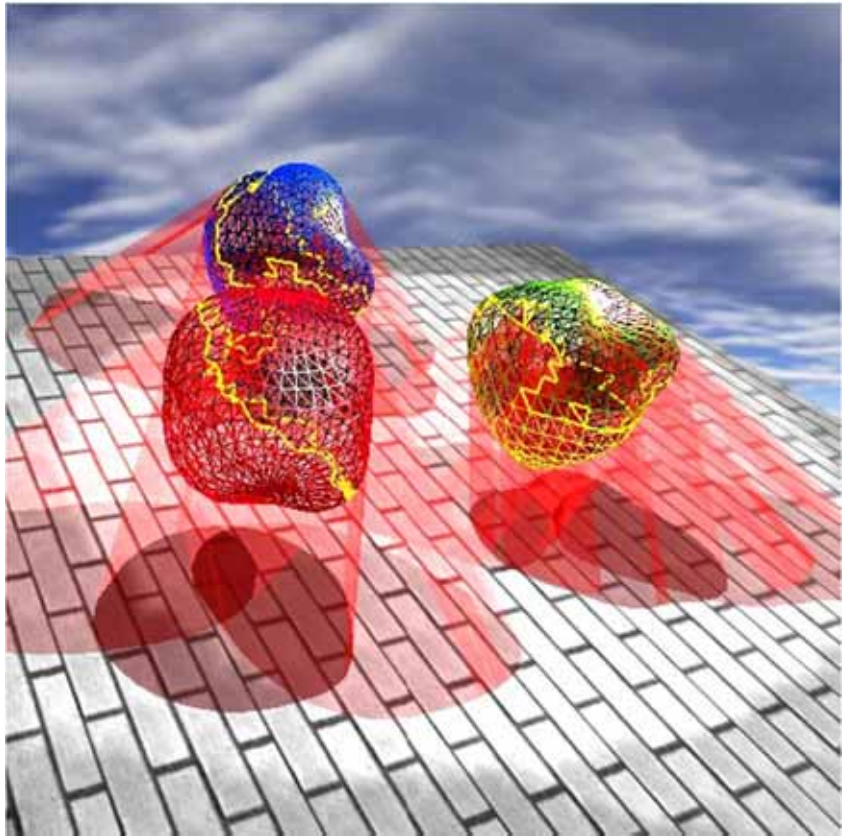
- und: **alle zur LQ gewandten Dreiecke sind ebenfalls Teil des Schattenvolumen**



# Schattenvolumen eines Dreiecksnetzes

- Beispiel: Schattenvolumen für 3 Lichtquellen
  - die gelben Kanten zeigen die Objektsilhouetten bzgl. einer der Lichtquellen

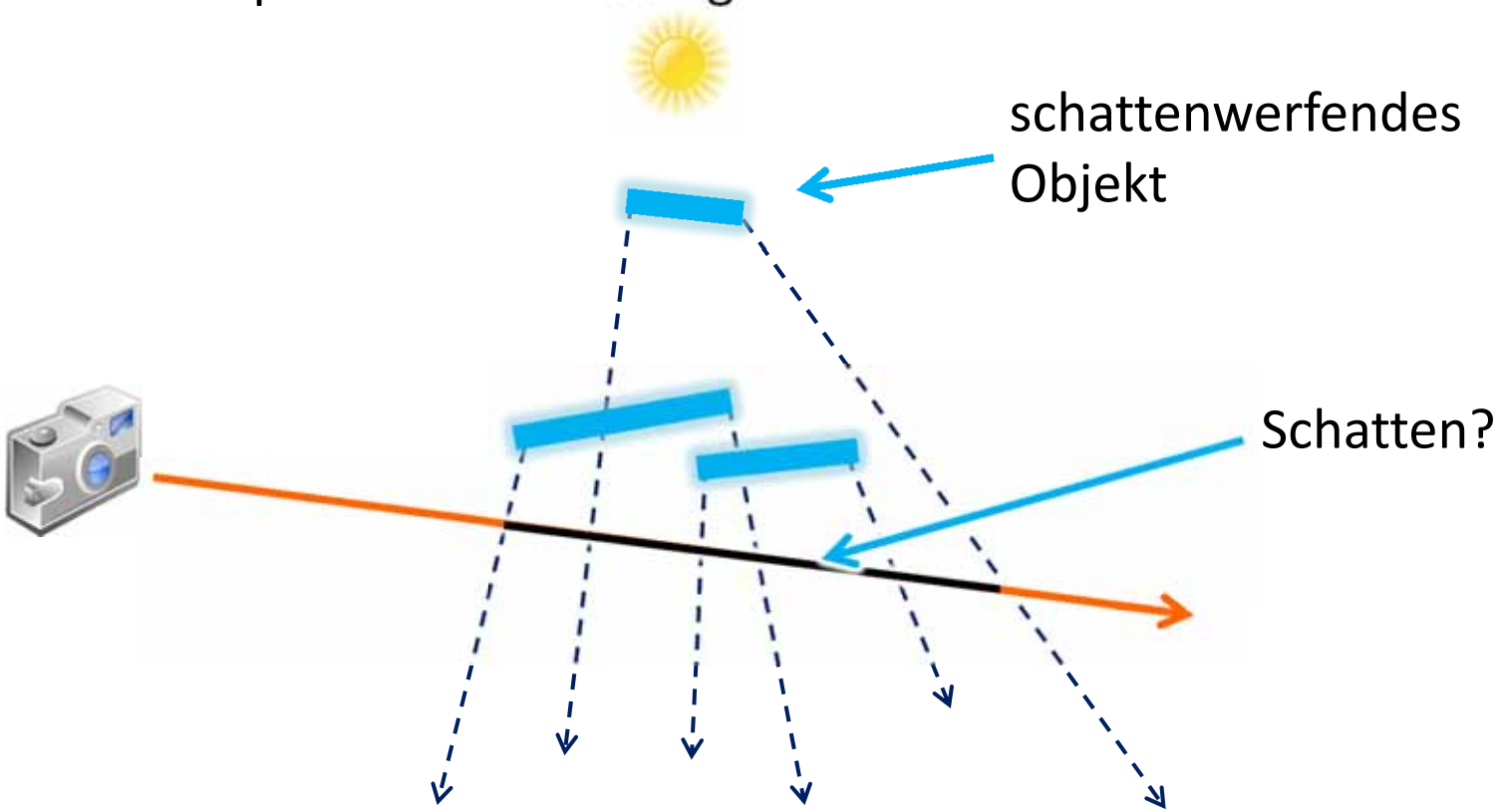




Bilder: Stefan Brabec

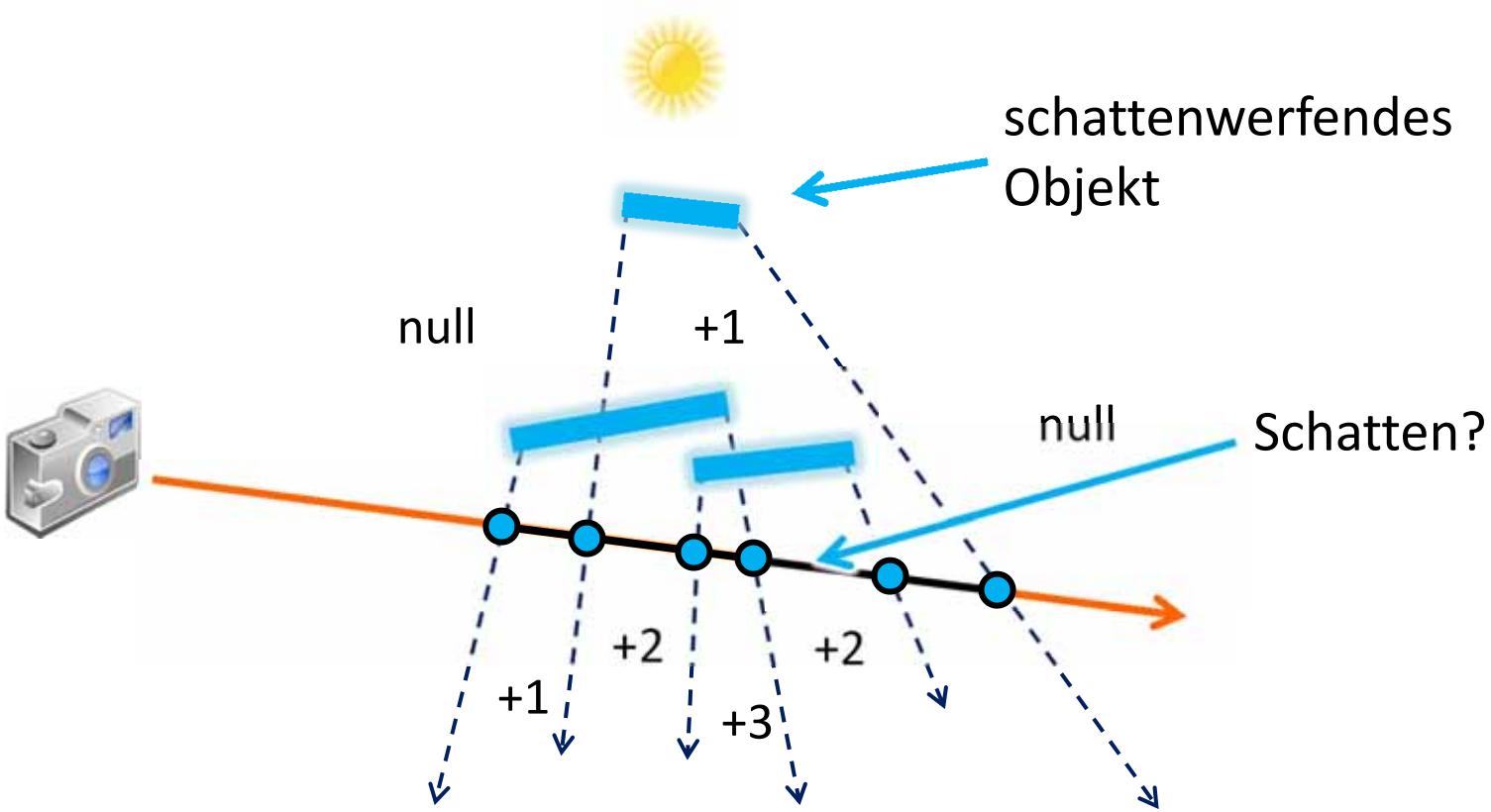
# Schattenvolumen: Schattentest

- mehrere (u.U. überlappende) Schattenvolumen entstehen durch
  - mehrere Objekte und/oder nicht-konvexe Oberflächen
- …aber wie stellt man eigentlich fest, ob sich ein Punkt in einem Schattenvolumen befindet?
  - eine Möglichkeit: Strahlschuss und Odd-Even Test (viel zu teuer!)
  - aber im Prinzip macht man doch genau das…



# Schattenvolumen: Schattentest

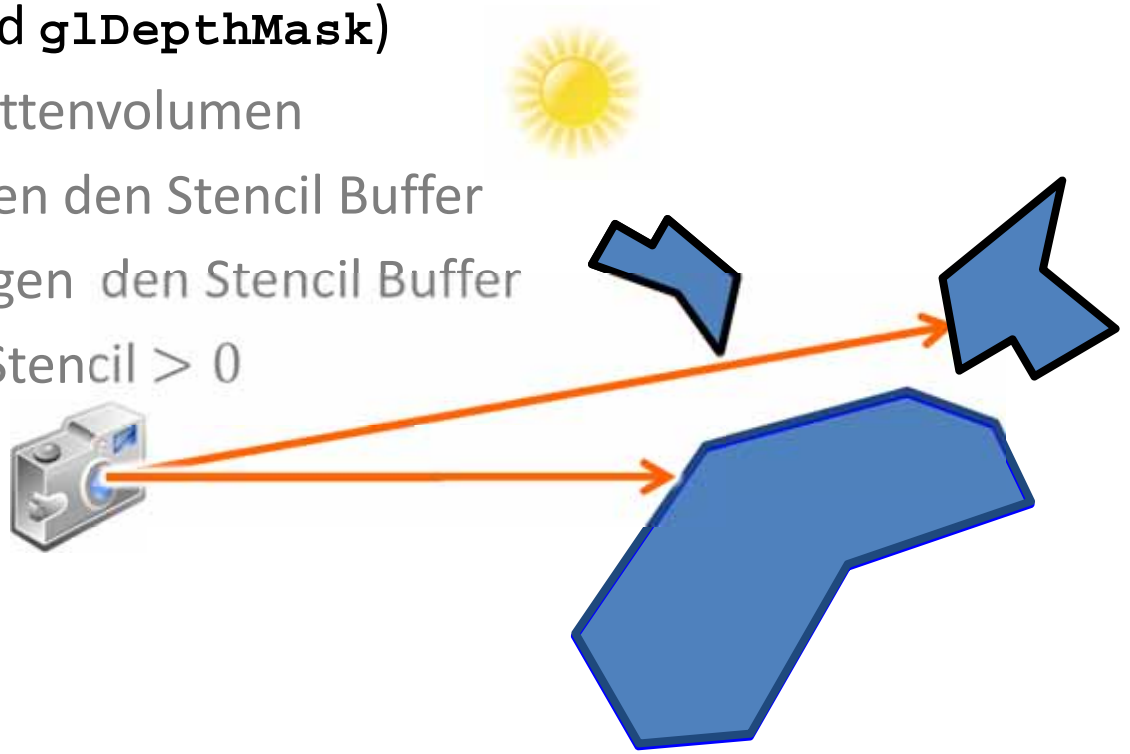
- verfolge einen Strahl vom Betrachter zum Oberflächenpunkt
  - setze einen Zähler zu Beginn auf Null
  - beim Eintritt in ein SV: inkrementiere Zähler
  - beim Verlassen eines SV: dekrementiere Zähler
  - an der Oberfläche: Punkt befindet sich im Schatten, wenn Zähler > 0



# Schattenvolumen: Schattentest

## Stencil Buffer zum Zählen der Schnitte mit Schattenvolumen

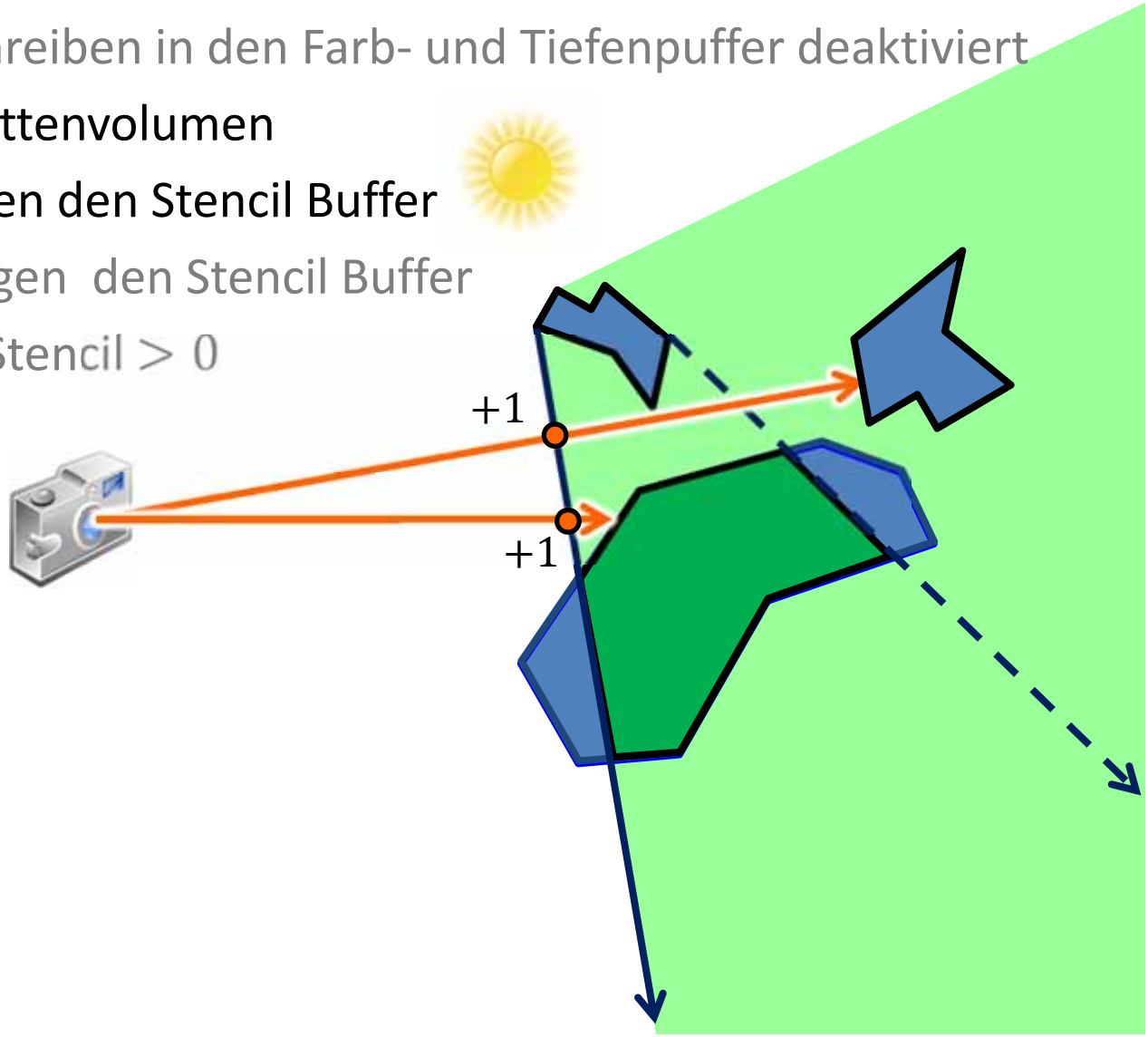
- zuerst werden die Objekte gezeichnet
- anschließend wird Schreiben in den Farb- und Tiefenpuffer deaktiviert (mit `glColorMask` und `glDepthMask`)
- zeichne dann die Schattenvolumen
  - Vorderseiten erhöhen den Stencil Buffer
  - Rückseiten erniedrigen den Stencil Buffer
- Schatten ist dort, wo Stencil $> 0$



# Schattenvolumen: Schattentest

## Stencil Buffer zum Zählen der Schnitte mit Schattenvolumen

- zuerst werden die Objekte gezeichnet
- anschließend wird Schreiben in den Farb- und Tiefenpuffer deaktiviert
- zeichne dann die Schattenvolumen
  - Vorderseiten erhöhen den Stencil Buffer
  - Rückseiten erniedrigen  den Stencil Buffer
- Schatten ist dort, wo Stencil $> 0$



# Schattenvolumen: Schattentest

## Einschub: Zählen mit dem Stencil Buffer (OpenGL)

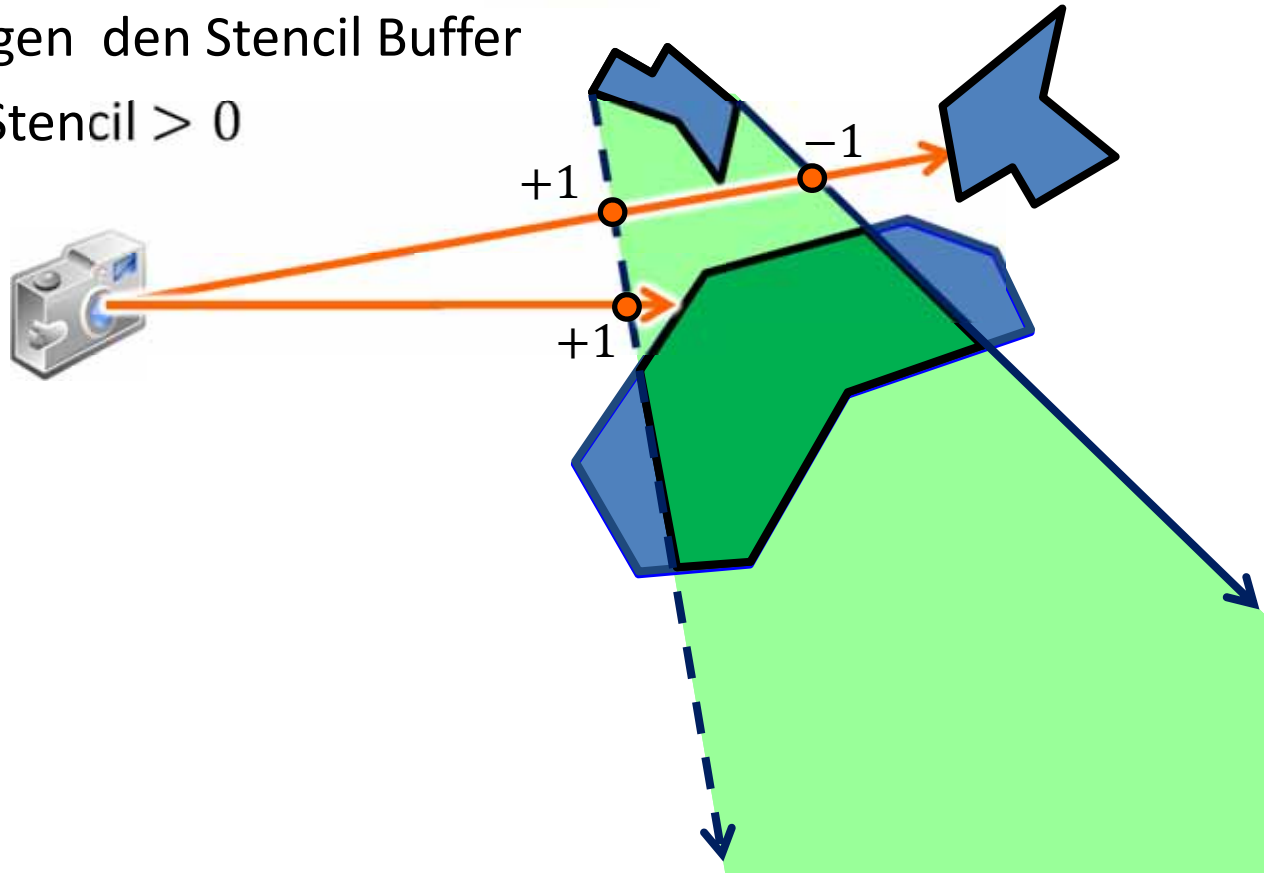
- zeichne Seitenflächen der Schattenvolumen, die zum Betrachter zeigen (= Eintrittspunkte des Strahls in das SV)
  - Auswahl überlässt man OpenGL durch Backface Culling
  - mit `glEnable( GL_CULL_FACE )` und `glFrontFace(...)` wird bestimmt, ob Vorder- oder Rückseiten gezeichnet werden (bei konsistenter Orientierung der Eckpunkte)

- inkrementiere Stencil-Wert, wenn der Tiefentest bestanden wird:
  `glStencilOp( GL_KEEP, GL_KEEP, GL_INCR )`
  - definiert die Operation für: (1) Stencil Test fehlgeschlagen, (2) Tiefentest fehlgeschlagen, (3) Tiefentest bestanden

# Schattenvolumen: Schattentest

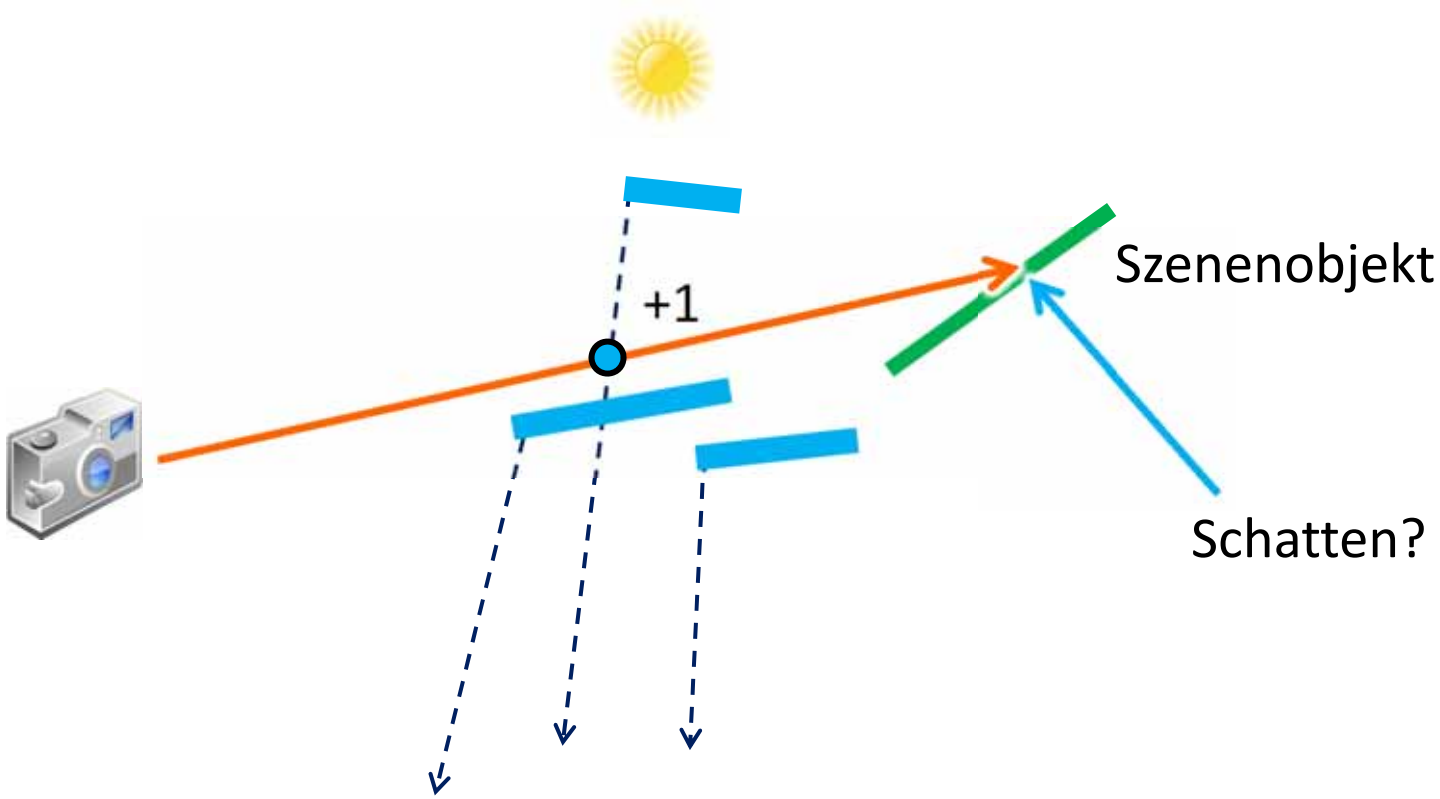
## Stencil Buffer zum Zählen der Schnitte mit Schattenvolumen

- zuerst werden die Objekte gezeichnet
- anschließend wird Schreiben in den Farb- und Tiefenpuffer deaktiviert
- zeichne dann die Schattenvolumen
  - Vorderseiten erhöhen den Stencil Buffer
  - Rückseiten erniedrigen  den Stencil Buffer
- Schatten ist dort, wo Stencil $> 0$



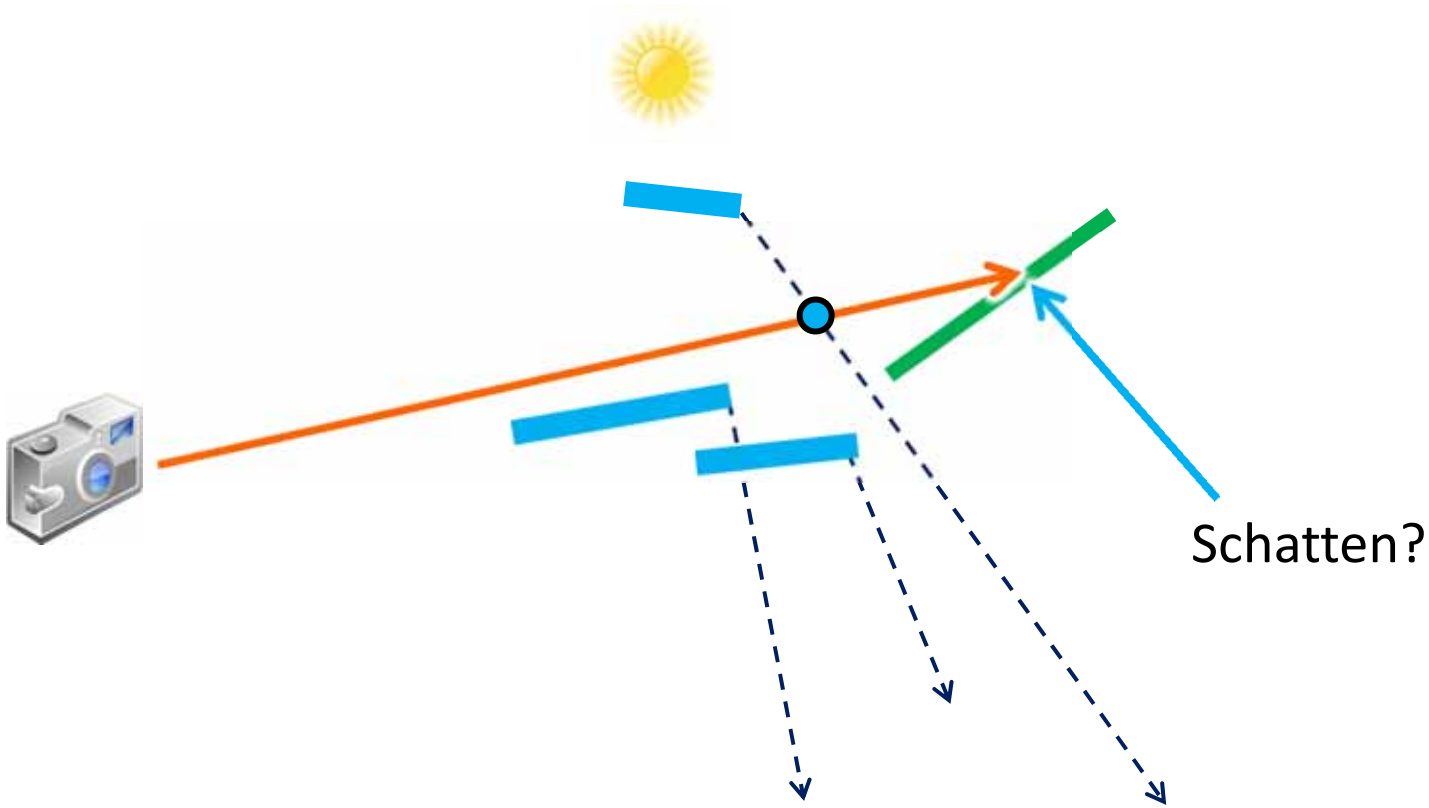
# Beispiel: Zählen mit dem Stencil Buffer

- zeichne Vorderseiten der Schattenvolumen
- inkrementiere Stencil-Wert, wenn der Tiefentest bestanden wird
- am betrachteten Punkt (bzw. für den Augstrahl der dorthin zeigt) ist der zunächst Zähler $= 1$



# Beispiel: Zählen mit dem Stencil Buffer

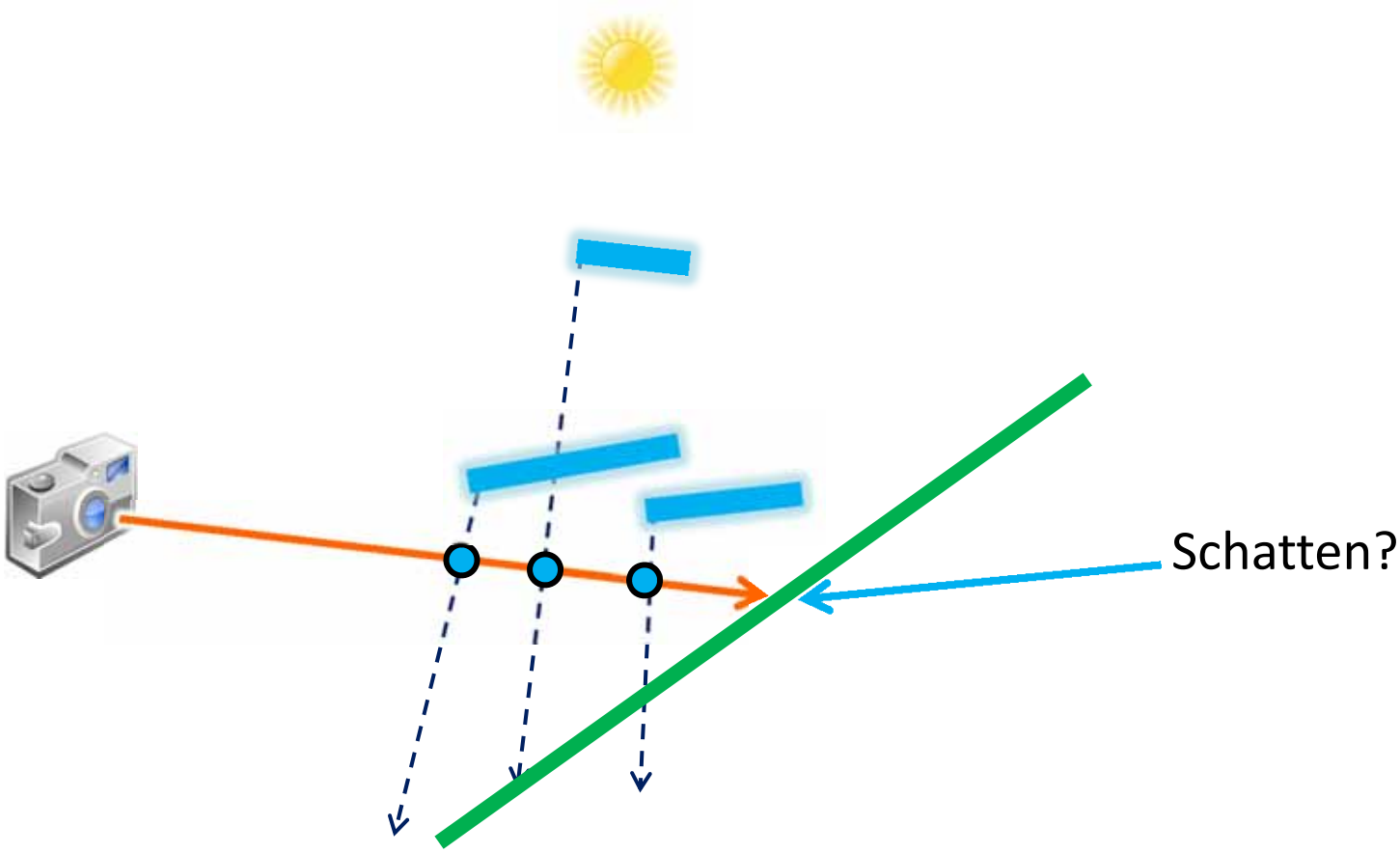
- zeichne Rückseiten der Schattenvolumen
- dekrementiere Stencil , wenn der Tiefentest bestanden wird:
  `glStencilOp( GL_KEEP, GL_KEEP, GL_DECR )`
- am betrachteten Punkt ist der Zähler $= 0$



# Beispiel: Zählen mit dem Stencil Buffer

## Weiteres Beispiel

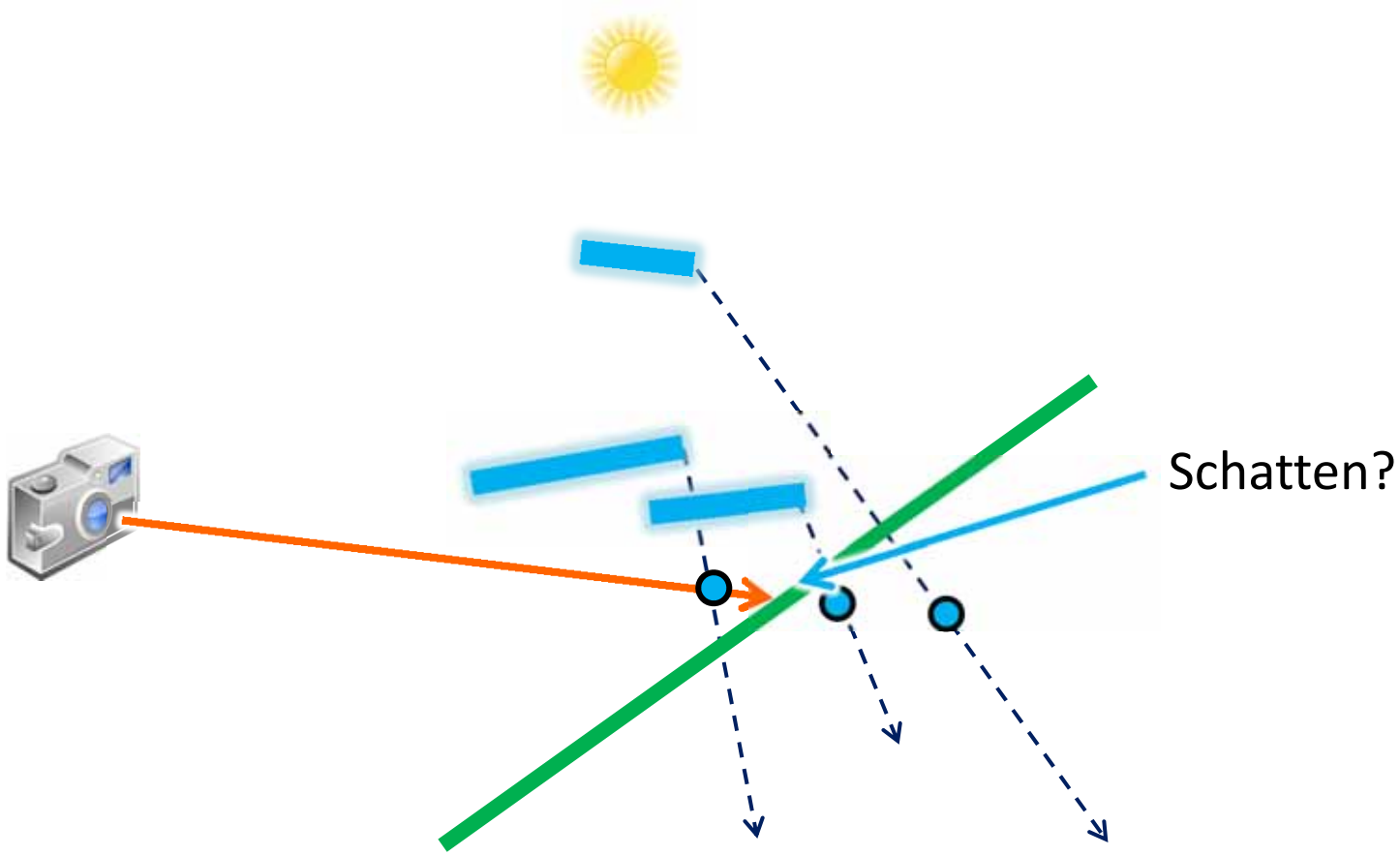
- zeichne Vorderseiten der Schattenvolumen
- Zähler am betrachteten Punkt $= 3$



# Beispiel: Zählen mit dem Stencil Buffer

## Weiteres Beispiel

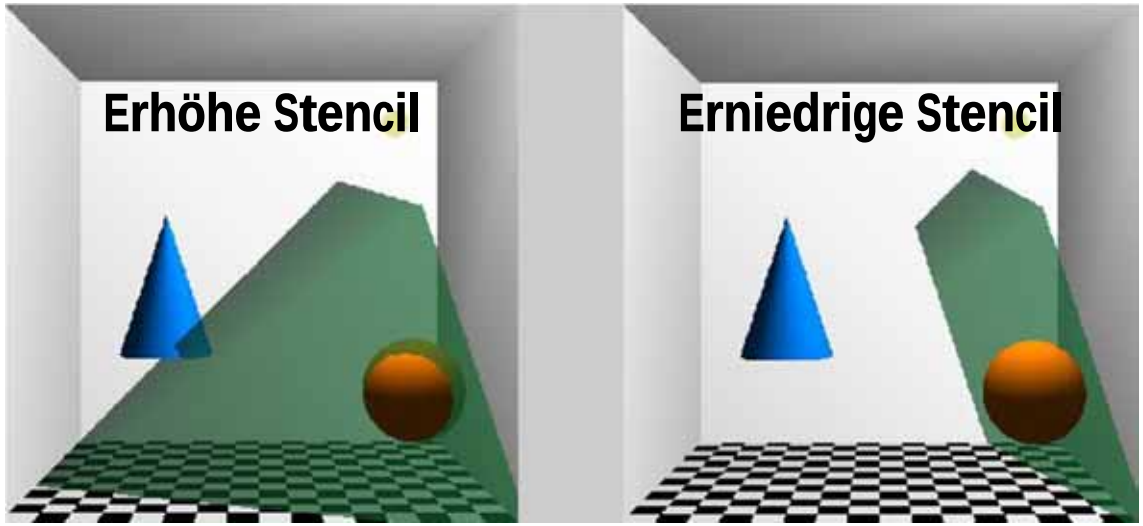
- zeichne Rückseiten der Schattenvolumen
- Zähler am betrachteten Punkt $= 2$
  - die zwei weiteren Schnittpunkte fallen beim Tiefentest durch

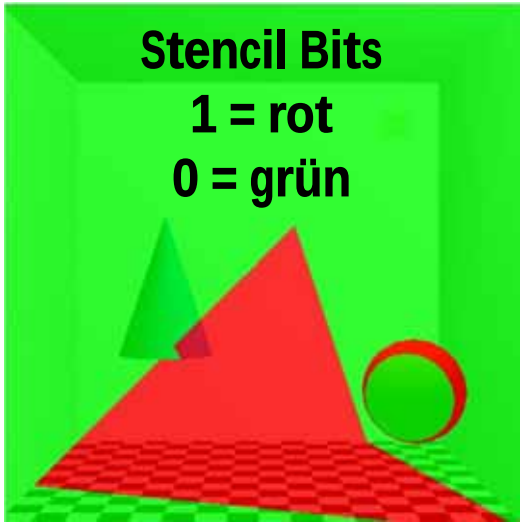


# Schattenvolumen: Zusammenfassung

## Verwendung mit Beleuchtungsberechnung

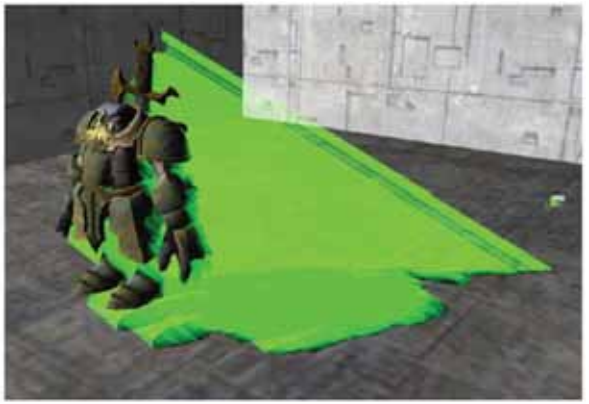
- zeichne die Szene nur mit ambienter Beleuchtung (und Tiefenpuffer)
- deaktiviere Schreibzugriffe auf Tiefen- und Farb-Puffer
- zeichne die Vorder- und Rückseiten des SV mit Stencil-Buffer
- zeichne die Szene mit Beleuchtung dort, wo der Stencilwert $= 0$ ist

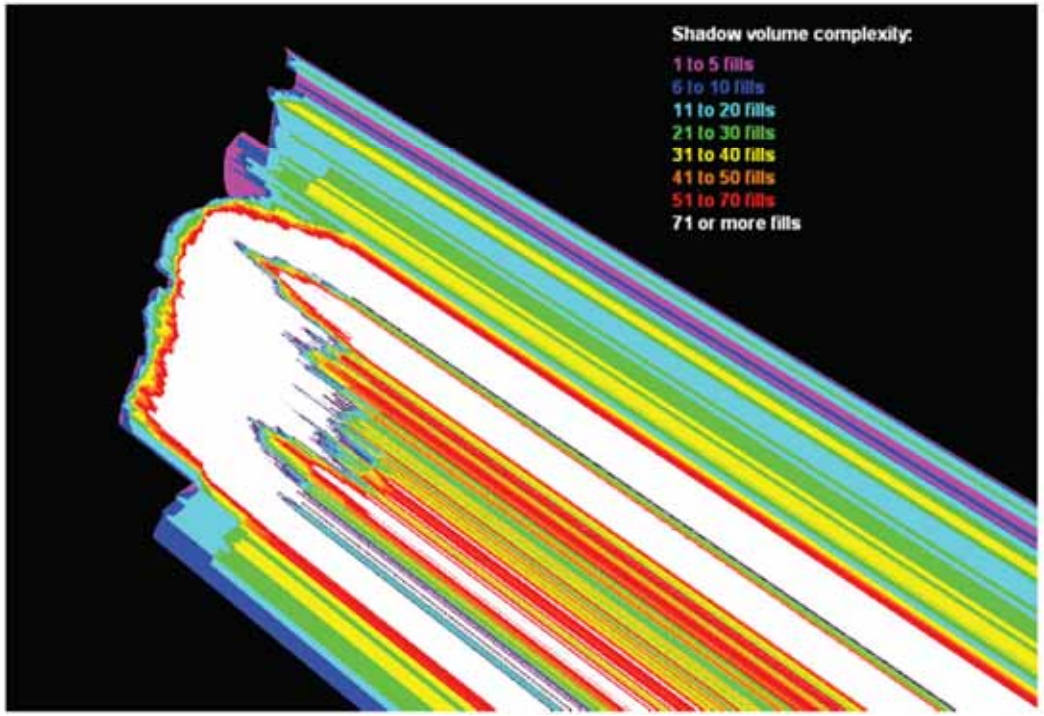


# Schattenvolumen: Problem Tiefenkomplexität

- selbst einfache Objekte erzeugen komplexe Schattenvolumen
- Tiefenkomplexität = Anzahl der Schnitte des Augstrahls mit SV





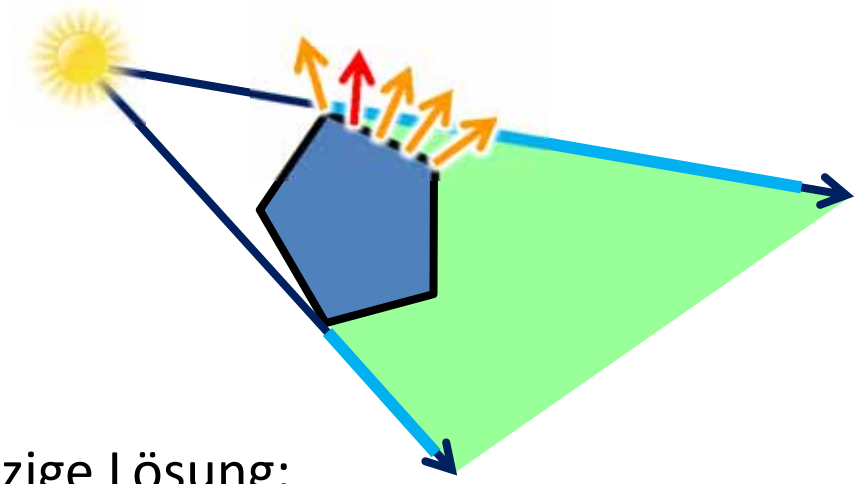
# Schattenvolumen: Technische Details...

- wie erzeugt man unendliche große Schattenvolumen?
  - durch Extrusion eines Vertex $\mathbf{v} = (v_x, v_y, v_z)$ und der Lichtquellenposition $\mathbf{l} = (l_x, l_y, l_z)$ erhalten wir den Punkt im Unendlichen mit:
    $\mathbf{v_h'} = (v_x - l_x, v_y - l_y, v_z - l_z, 0)$
- Unterscheidung der Vorder- und Rückseiten
  - auf einfachsten wie beschrieben durch Backface-Culling
  - OpenGL unterstützt einen "two-sided stencil test"
    - unterschiedliche Stencil-Buffer Operationen können für Vorder- und Rückseiten festgelegt werden:
      `glActiveStencilFace( GL_FRONT ); glStencilOp( ... );`
    - damit genügt ein einmaliges Zeichnen der Schattenvolumen
- Bittiefe des Stencil Buffers ist beschränkt (meist 8 Bit)
  - d.h. der Zähler läuft bei einer Tiefe von 256 über (!)
  - kann zu Problemen bei komplexen Szenen führen
  - deswegen: zeichne abwechselnd Vorder- und Rückseiten, z.B. durch Zeichnen der SV einzelner Objekte und nicht der ganzen Szene oder two-sided Test

# Schattenvolumen: Problem Beleuchtung

## Inkonsistenz zwischen Verschattung und Beleuchtung

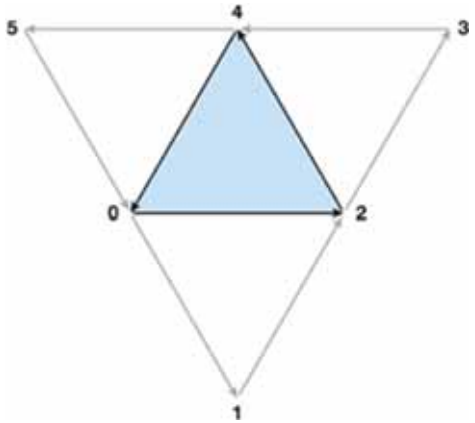
- Entscheidung, ob ein Punkt beleuchtet ist erfolgt durch das SV
  - … berechnet aus der Silhouette, also auf Basis der **Dreiecksnormalen**
- Problem: Beleuchtungsberechnung erfolgt mit interpolierter Normale
- kann zu sichtbaren Artefakten an den Silhouetten führen
- Beispiel: interpolierte Normale
  noch zur Lichtquelle gewandt,
  aber im Schatten



- einzige Lösung:
  feiner tessellierte Netze

# Berechnung der Silhouette auf der GPU

- wie berechnet man die Silhouette **effizient**?
- **Geometry Shader:**
  - im GS ist es möglich auf Nachbardreiecke zuzugreifen, wenn die entsprechenden Indizes der Vertices angegeben werden
  - man zeichnet Primitive vom Typ `GL_TRIANGLES_ADJACENCY_EXT` und übergibt pro Dreieck 6 Vertices statt 3
  - zur Erinnerung: im GS greift man auf die Vertices zu mit `gl_PositionIn[]`
  - Eingabe: Dreieck mit Nachbardreiecken
  - Ausgabe: Seitenflächen des Schattenvolumen und das Dreieck selbst (wenn es zur LQ zeigt)

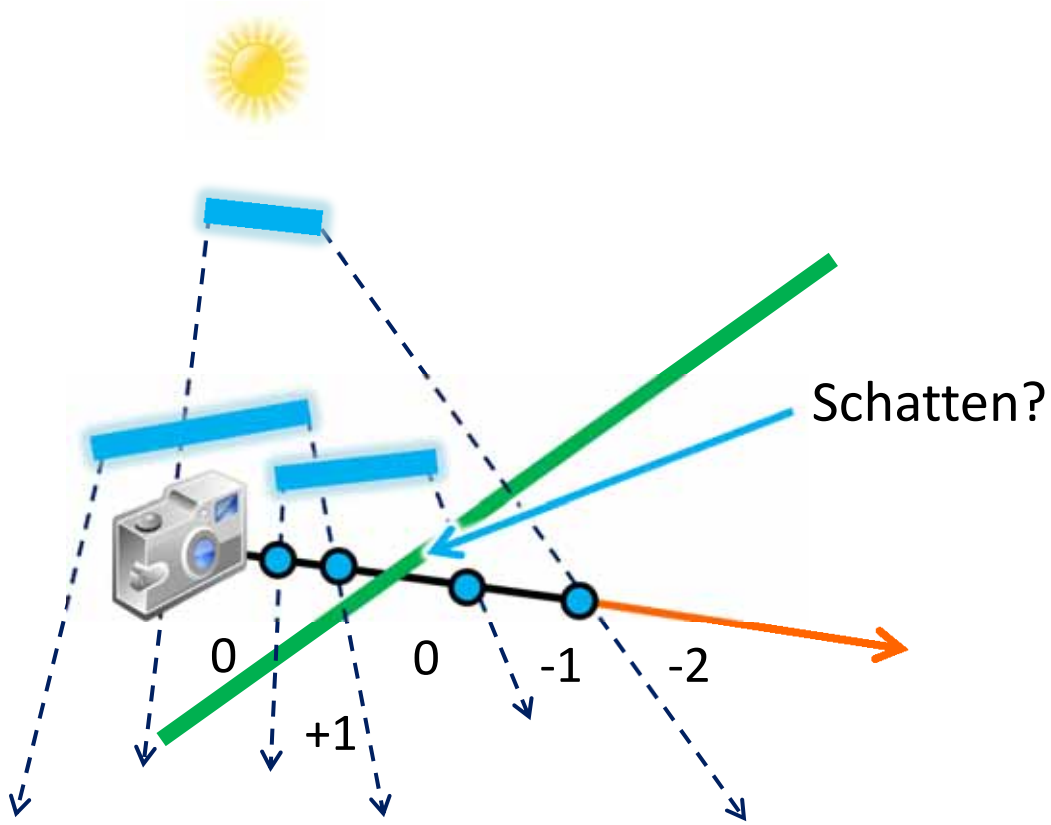



  - detaillierte Beschreibung dieser Technik: http://http.developer.nvidia.com/GPUGems3/gpugems3_ch11.html

# Schattenvolumen: Probleme des Stencil Buffer

## Probleme durch nicht gefundene Ein- und Austrittspunkte

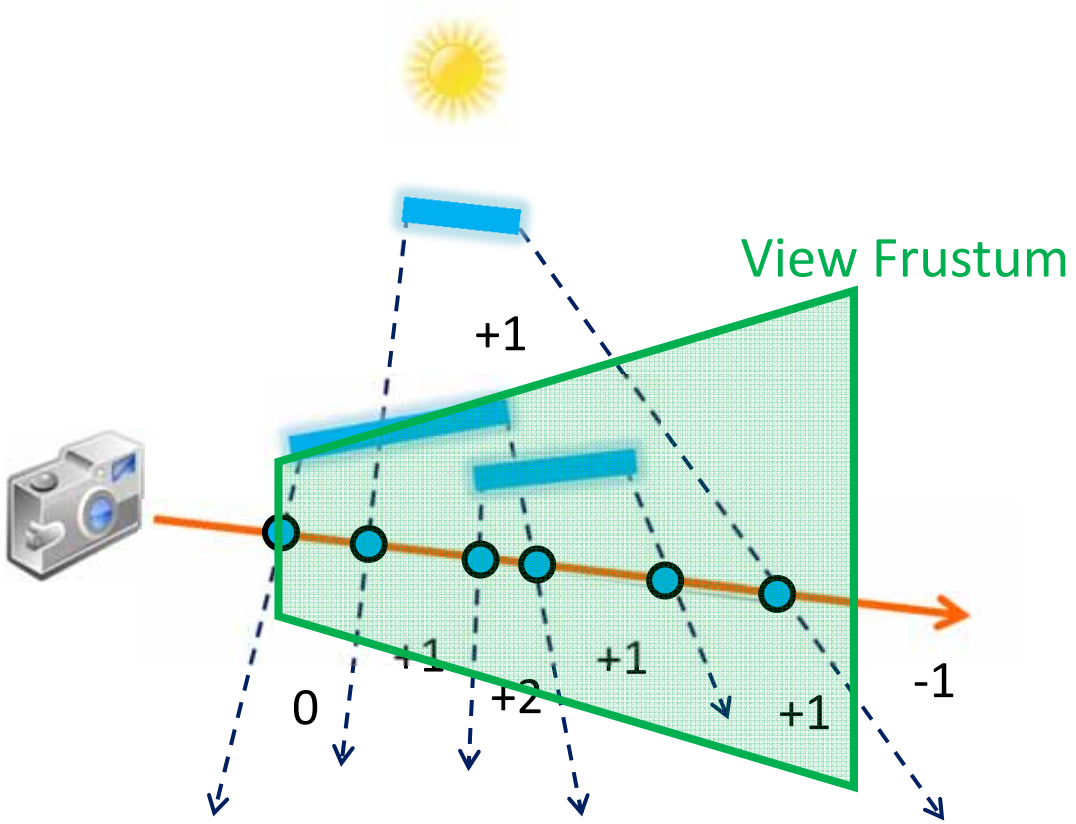
- Ein- und Austrittspunkte werden nur gezählt, wo das SV rasterisiert wird
- was passiert – wie in diesem Beispiel – wenn sich der Betrachter in einem Schattenvolumen befindet?
  - der Startwert des Zählers ist falsch!



# Schattenvolumen: Probleme des Stencil Buffer

## Probleme durch nicht gefundene Ein- und Austrittspunkte

- Ein- und Austrittspunkte werden nur gezählt, wo das SV rasterisiert wird
- die Near Plane kann Eintrittspunkte entfernen und unterschiedliche Pixel müssten eigentlich unterschiedliche Startwerte für den Zähler besitzen
- Lösung: der z-Fail Algorithmus (→ Übung)

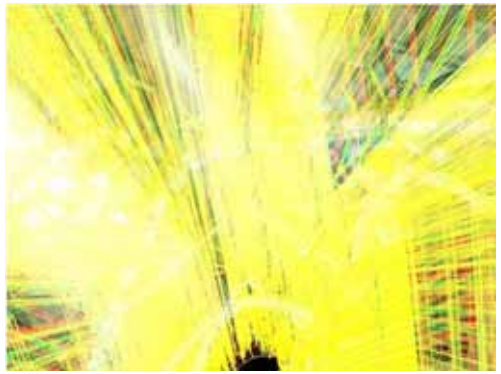


# Schattenvolumen: Zusammenfassung


- wichtige Eigenschaften
  - Objektraum-Verfahren, d.h. Schatten sind „pixelgenau“
  - Schattenvolumen sind unabhängig von Kamera
    (erlaubt fixe Schattenvolumen bei statischen Szenen)
  - Selbstverschattung von Objekten
- Nachteile
  - hoher Rasterisierungsaufwand für das Zeichnen der Schattenvolumen
  - mehrere Rendering-Durchgänge notwendig: Zählen der Ein-/Austritte, Rendering von ambientem Licht und Beleuchtung separat, …
  - mehrere Durchgänge bei mehreren Lichtquellen
- Fazit: werden verwendet, wenn **akkurate harte Schatten** wichtig sind
  - beispielsweise in CAD Anwendungen

# Beispiel

sichtbare Geometrie

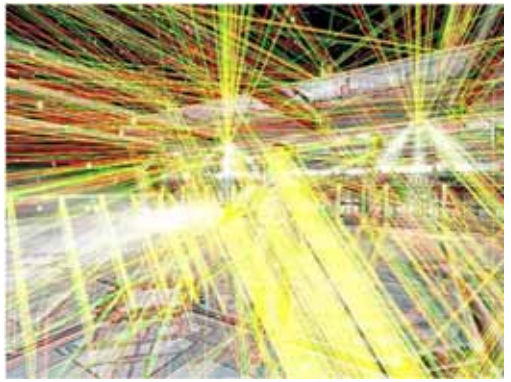


Schattenvolumen

*Abducted g*ame images courtesy
Joe Riedel at Contraband Entertainment

# Beispiel

sichtbare Geometrie

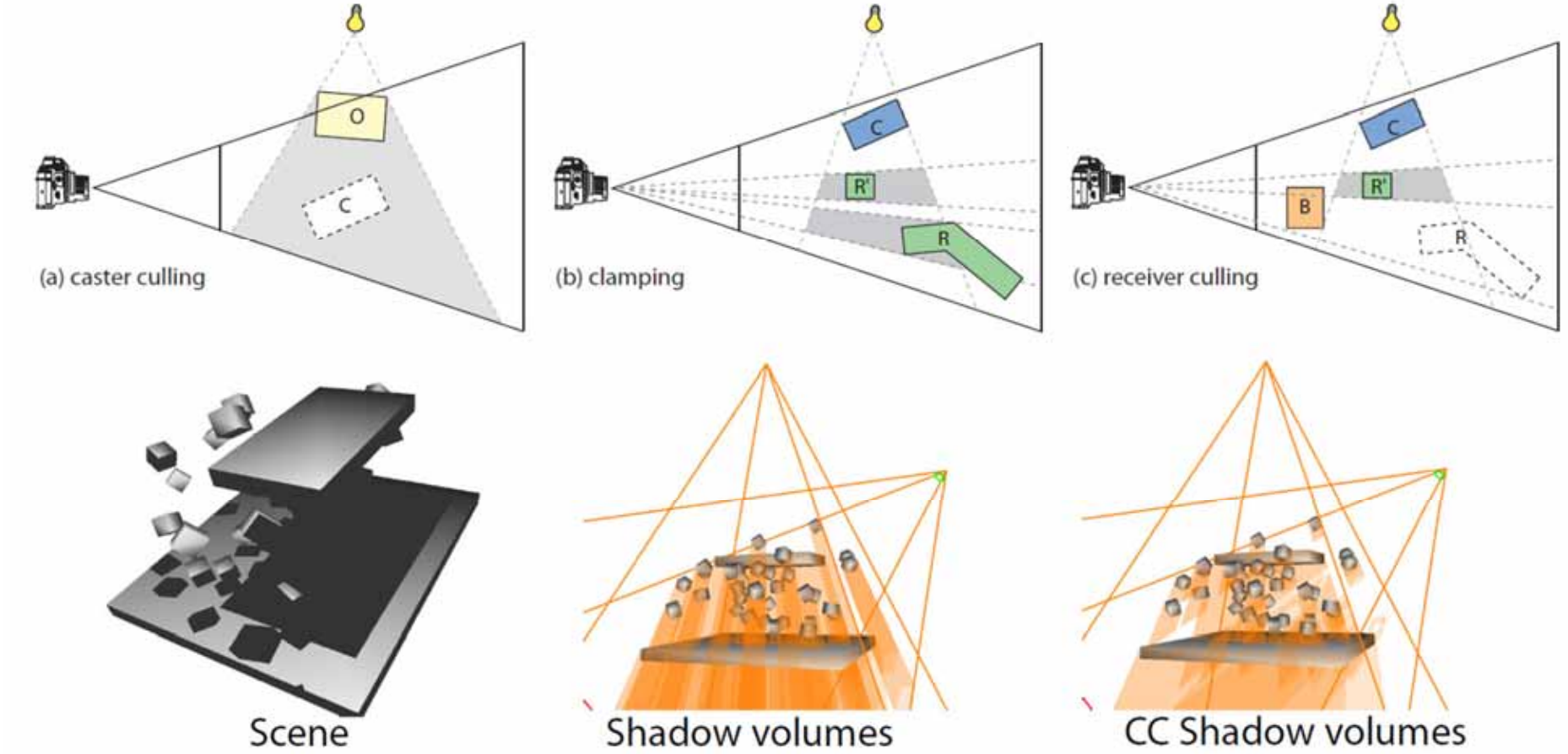


Schattenvolumen

*Abducted g*ame images courtesy
Joe Riedel at Contraband Entertainment

# Optimierung von Schattenvolumen

- Verwerfen von Schattenvolumen für Objekte die selbst vollständig im Schatten liegen (links)
- Begrenzen von Schattenvolumen auf Bereiche in denen Objekte liegen (mitte) und nicht von der Kamera verdeckte Bereiche (rechts)




CC Shadow Volumes, http://gamma.cs.unc.edu/CCSV/