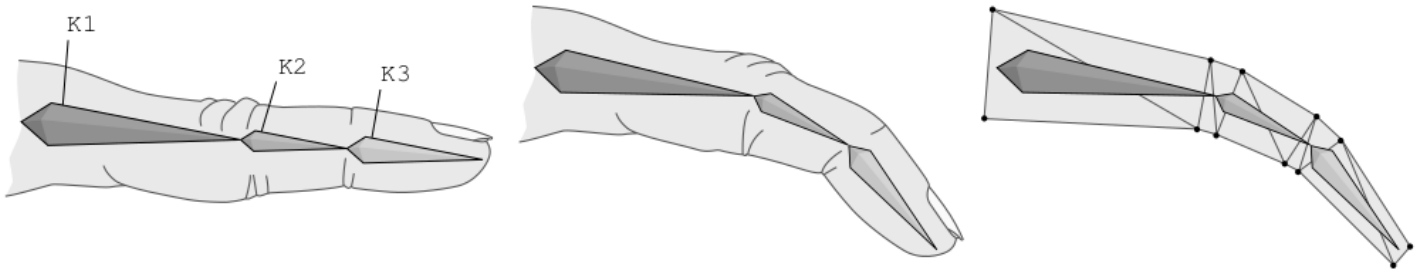
Computergrafik

Übung: Distributed Raytracing

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



Der Knochen `K1` ist die Wurzel der Skeletthierarchie. Sein Kindknochen ist `K2`. `K3` ist das Kind von `K2`.

Es sind drei Transformationsmatrizen gegeben. `M1` beschreibt die Lage des gesamten Fingers in Weltkoordinaten. `M2` beschreibt die Lage von `K2` *relativ zu* `K1`. `M3` beschreibt die Lage von `K3` *relativ zu* `K2`.

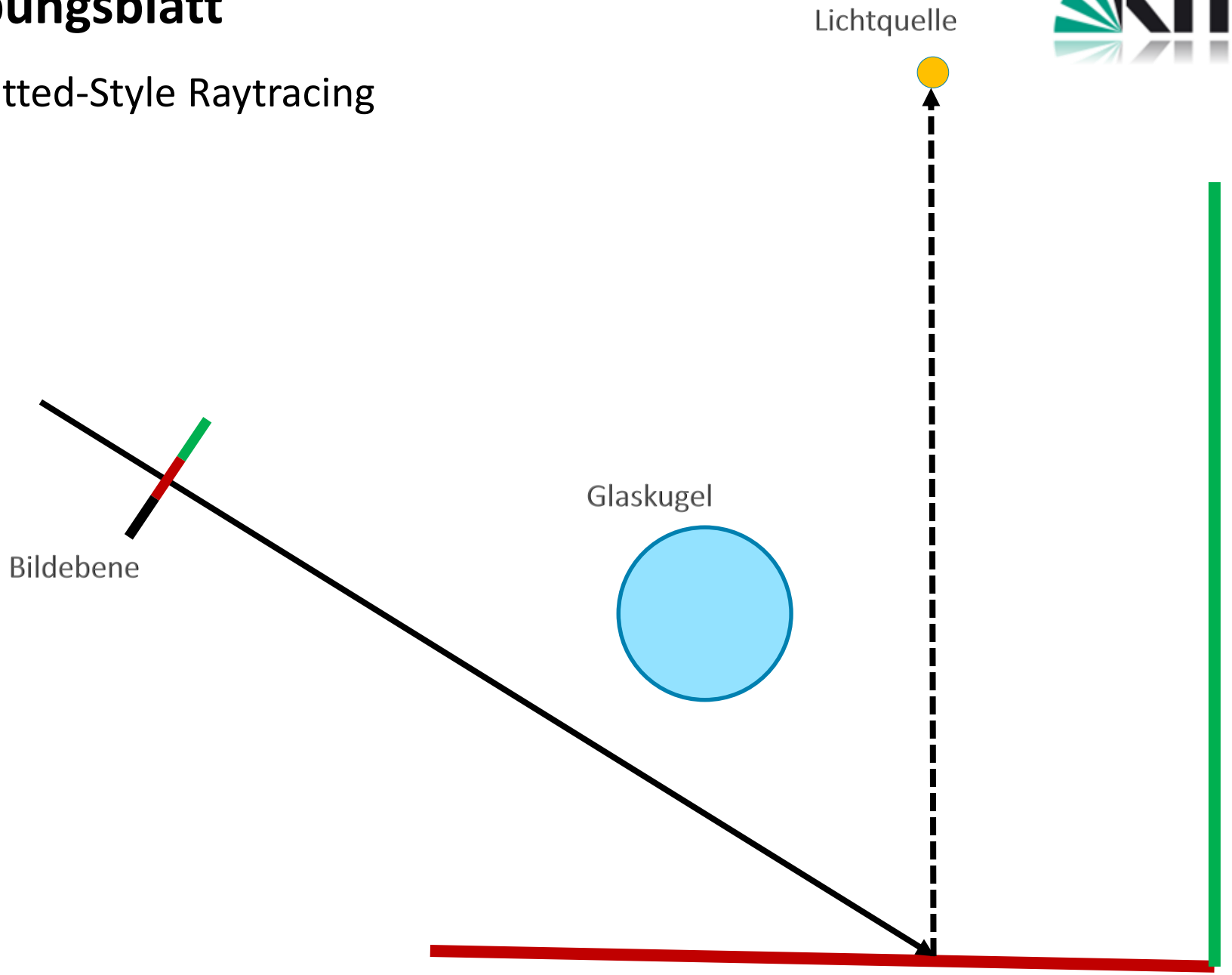
Für jeden Vertex sind Gewichte `w[0]`, `w[1]` und `w[2]` gegeben. Sie beschreiben, wie viel Einfluss die Transformation des entsprechende Knochens auf den Vertex hat.

Vervollständigen Sie den Vertex-Shader, sodass die Position des Vertex `P` wie angegeben gewichtet transformiert wird! In `gl_Position` muss dann letztendlich die Position in Clip-Koordinaten geschrieben werden.

```
in vec3 P;            // Position des Vertex in Objektkoordinaten.
in float w[3];        // Einfluss der Knochen.

uniform mat4 matVP;   // View-Projection-Matrix.
uniform mat4 M1;      // Transformation des Fingers (Objekt -> Welt).
uniform mat4 M2;      // Transformation von K2 relativ zu K1.
uniform mat4 M3;      // Transformation von K3 relativ zu K2.

void main()
{

    ...

    gl_Position = ...
}
```

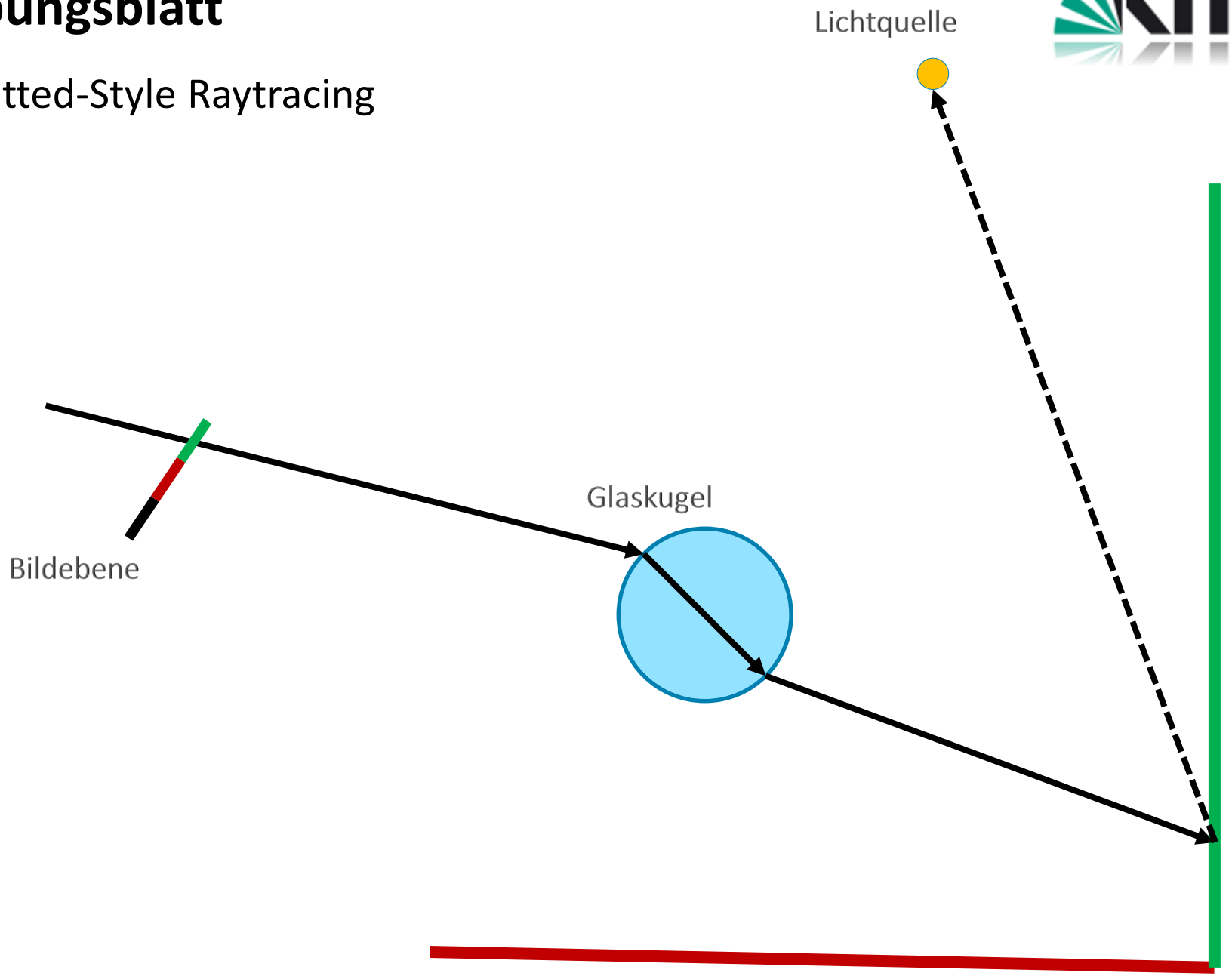
2. Übungsblatt

- Whitted-Style Raytracing



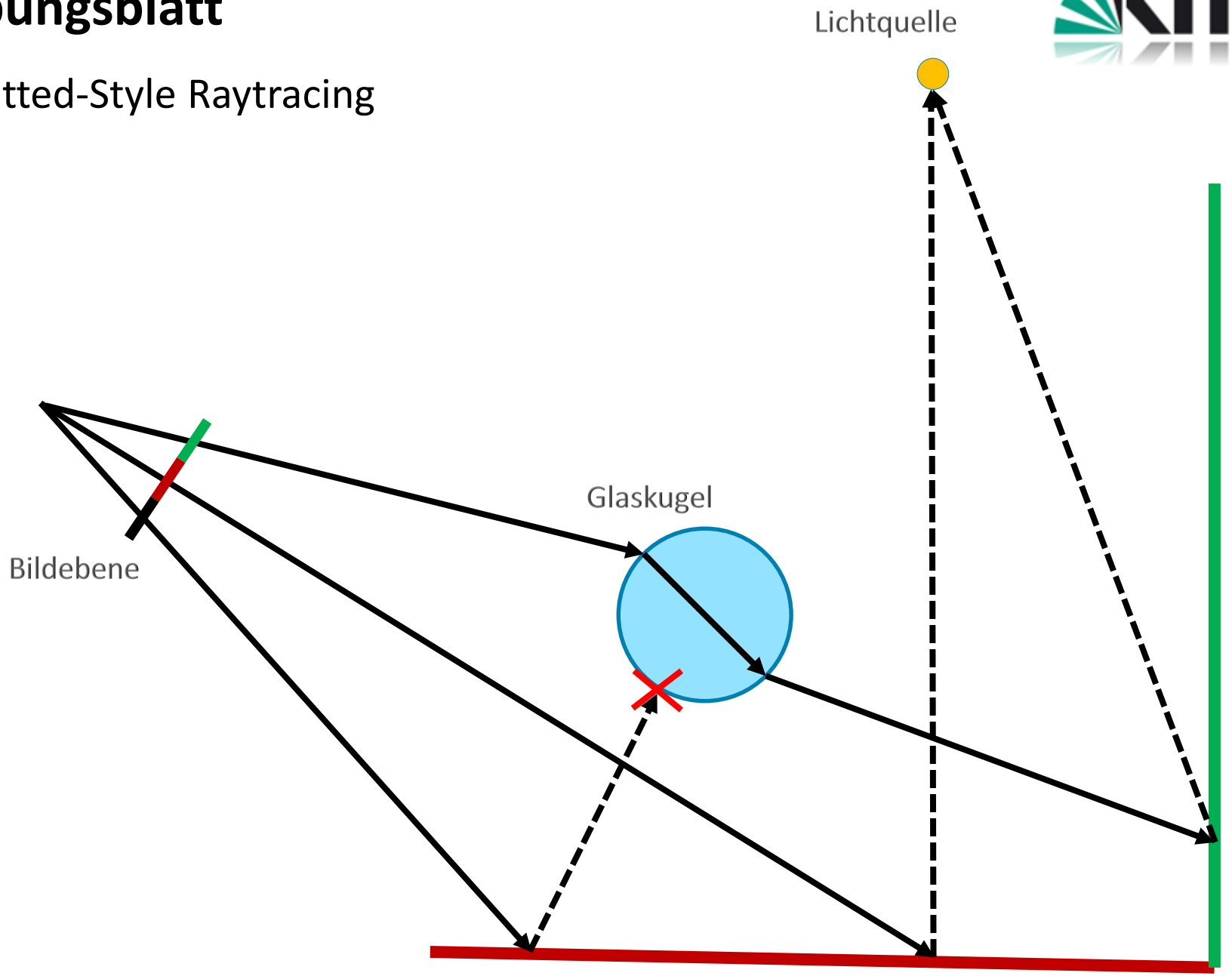
2. Übungsblatt

- Whitted-Style Raytracing



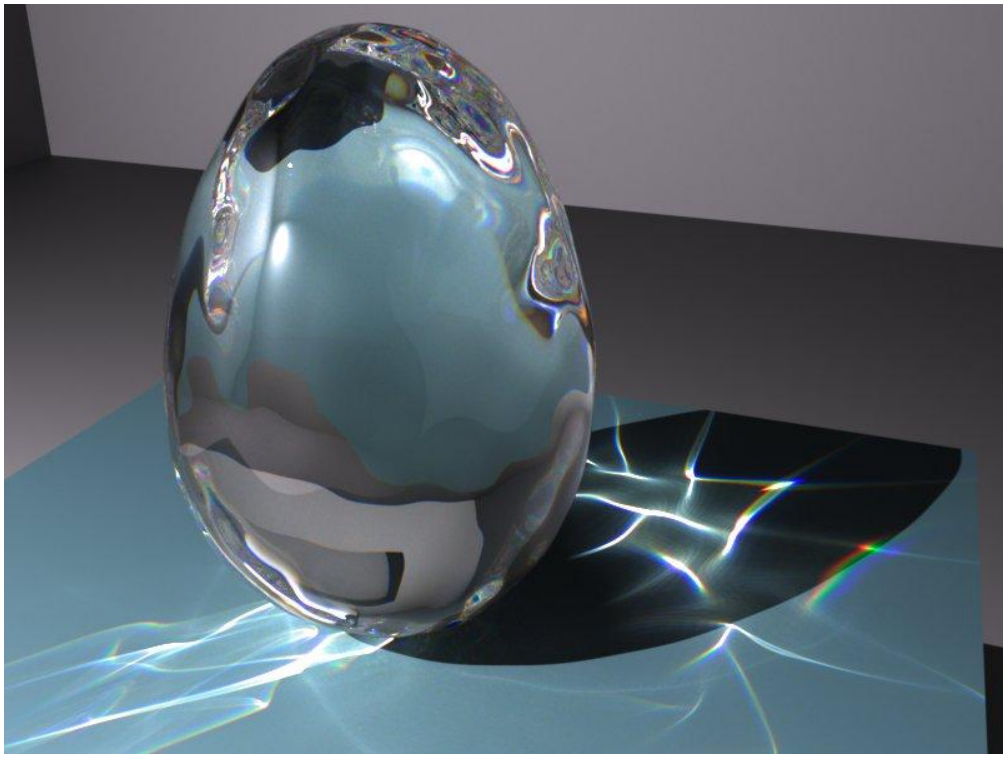
2. Übungsblatt

- Whitted-Style Raytracing



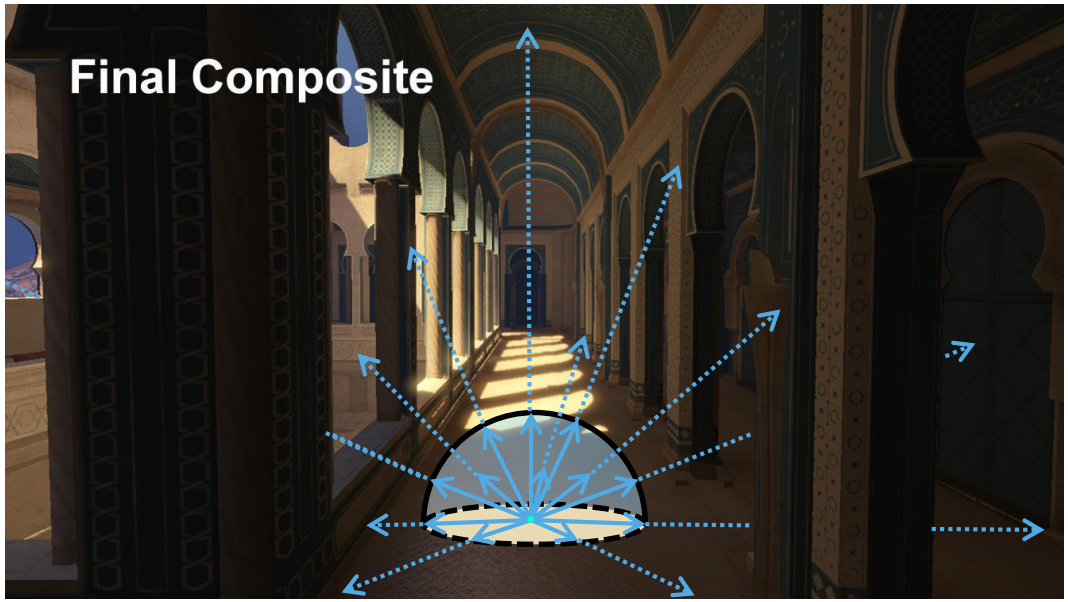
Limitationen von Whitted-Style Raytracing

- Keine Kaustiken oder korrekte Dispersion
- Keine indirekte Beleuchtung
- Keine Flächenlichtquellen
- Kein Motion-Blur oder Tiefenunschärfe
- Mögliche Lösung => Distributed Raytracing

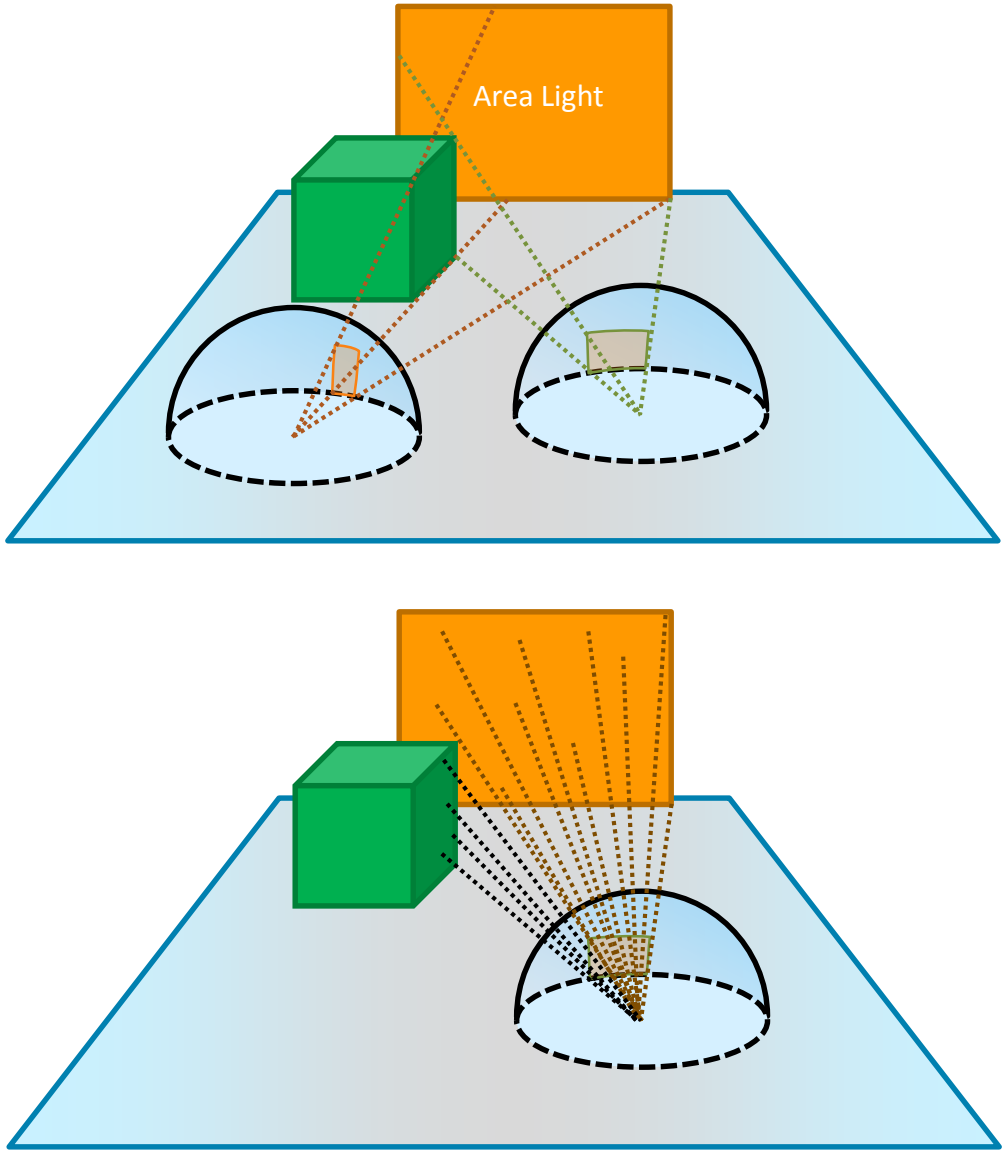


Distributed Raytracing

- Erlaubt es (fast) alles zu rendern
  - Tiefenunschärfe
  - Weiche Schatten
  - Indirekte Beleuchting
  - …



Distributed Raytracing - Flächenlichtquellen



Distributed Raytracing - Flächenlichtquellen

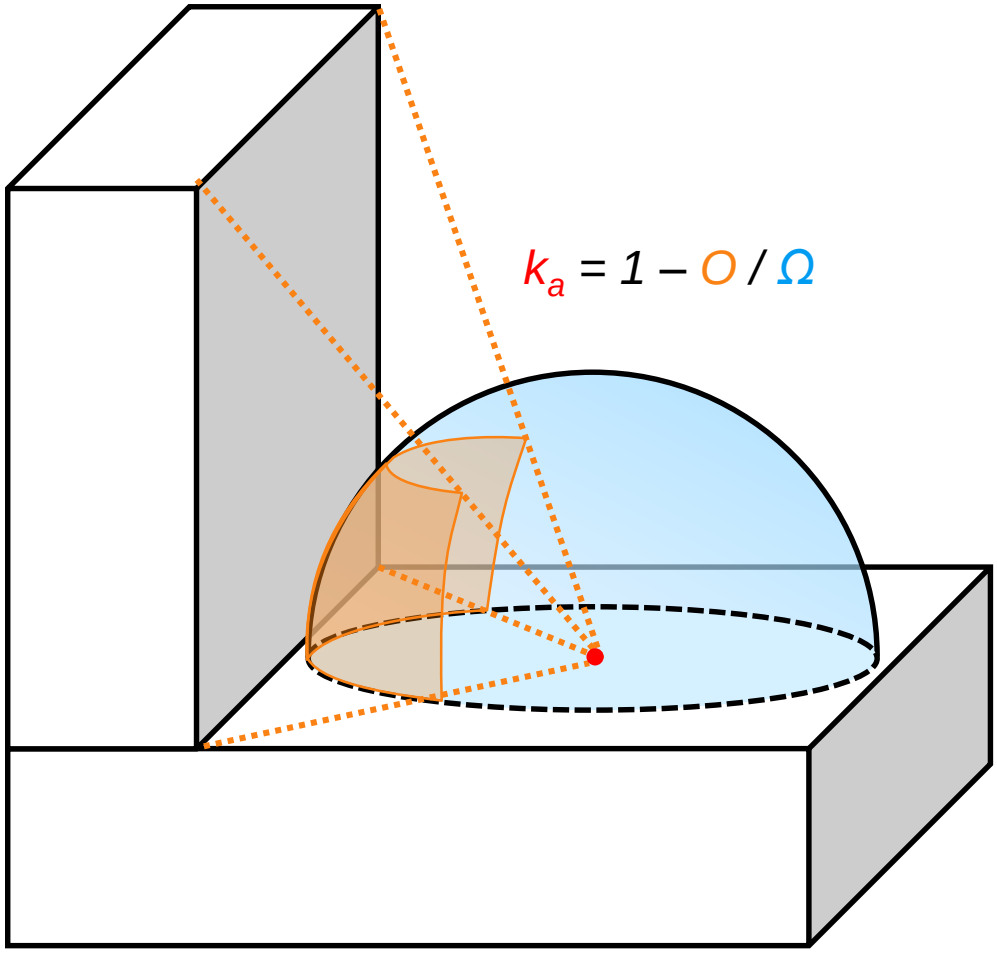
$$L(x,\omega) = \int_{\Omega^+} f(\omega_i, x, \omega) L_i(x, \omega_i) cos\theta_i d\omega_i$$

$$= \sum_{k=1}^{K} \int_{A_k} f(-\omega_o, x, \omega) L_e(x_o, \omega_o) V(x, x_o) \frac{cos\theta_i cos\theta_o}{\|x - x_o\|^2} dx_o$$

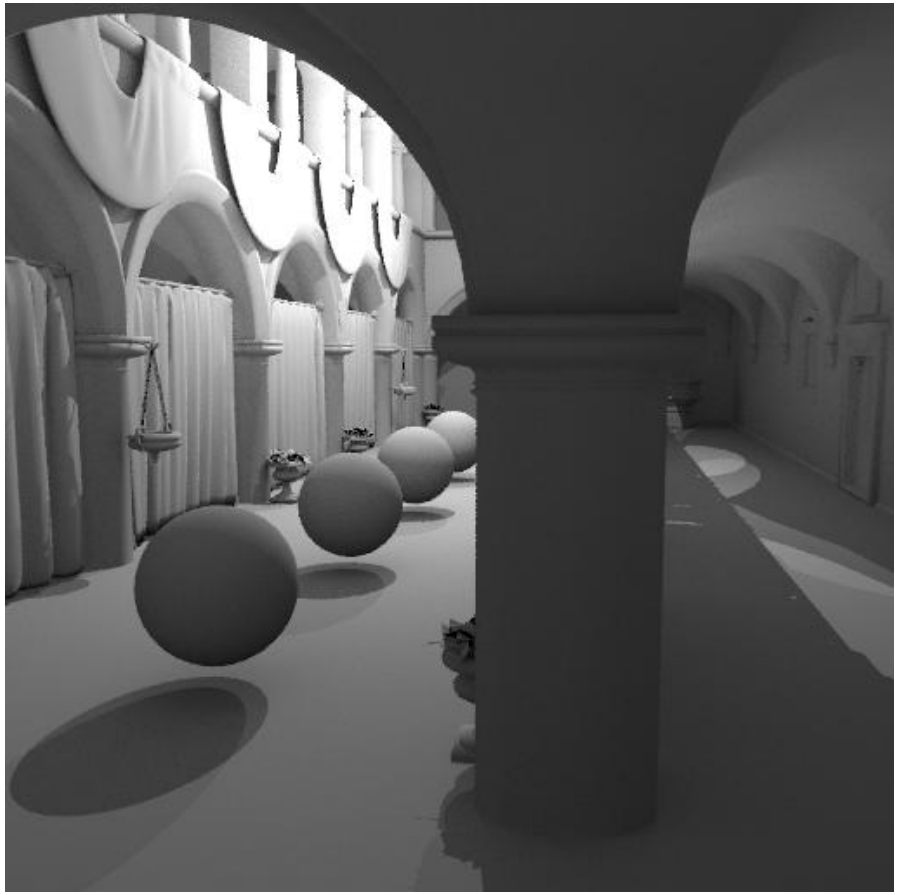
$$= \sum_{k=1}^{K} \frac{\|A_k\|}{N} \sum_{l=1}^{N} f(-\omega_o) L_e\left(x_o^{(l)}, \omega_o\right) V(x, x_o^{(l)}) \frac{cos\theta_i cos\theta_o}{\left\|x - x_o^{(l)}\right\|^2}$$

Distributed Raytracing - Flächenlichtquellen

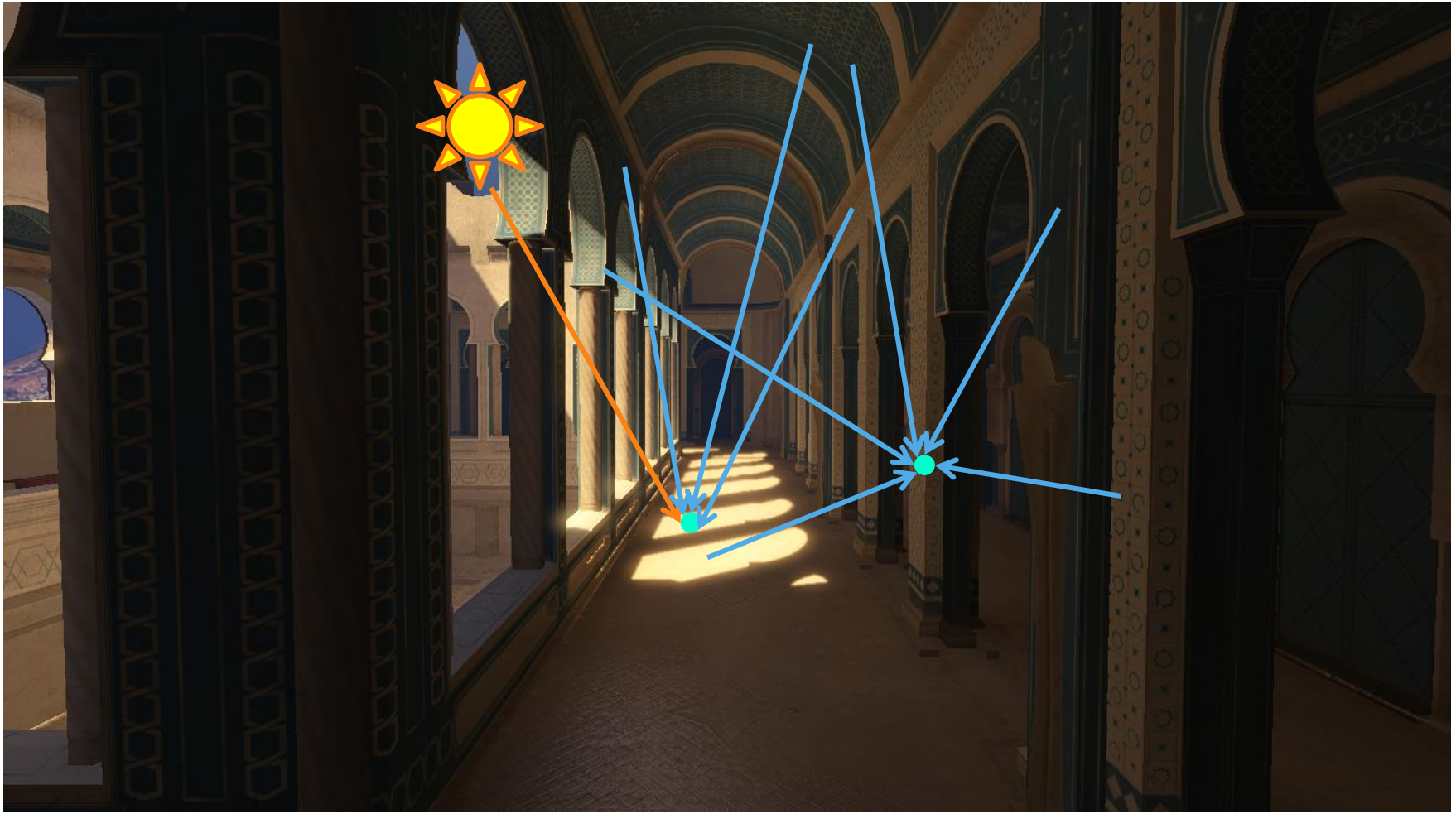
Distributed Raytracing – Ambient Occlusion



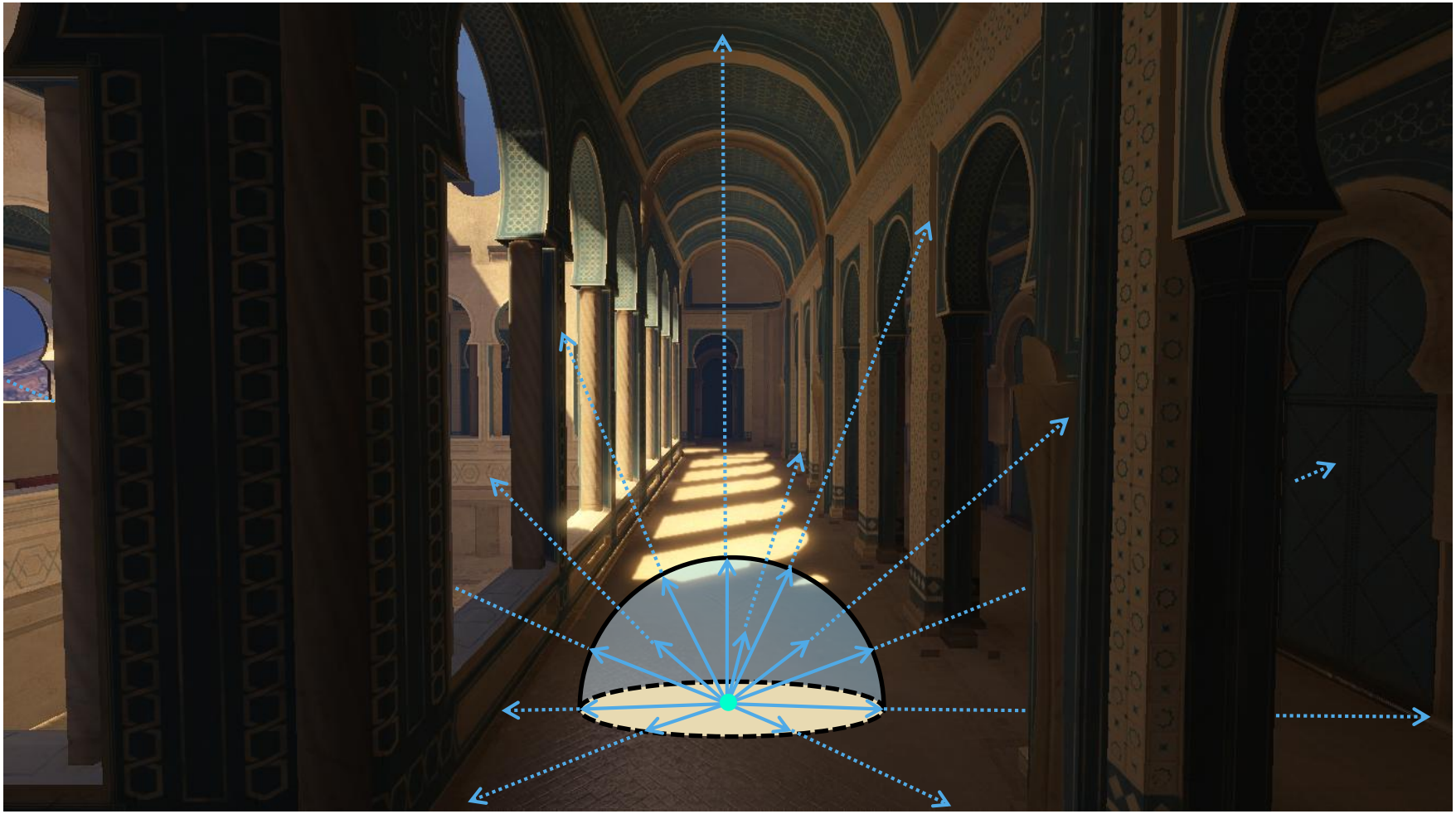
Distributed Raytracing – Ambient Occlusion



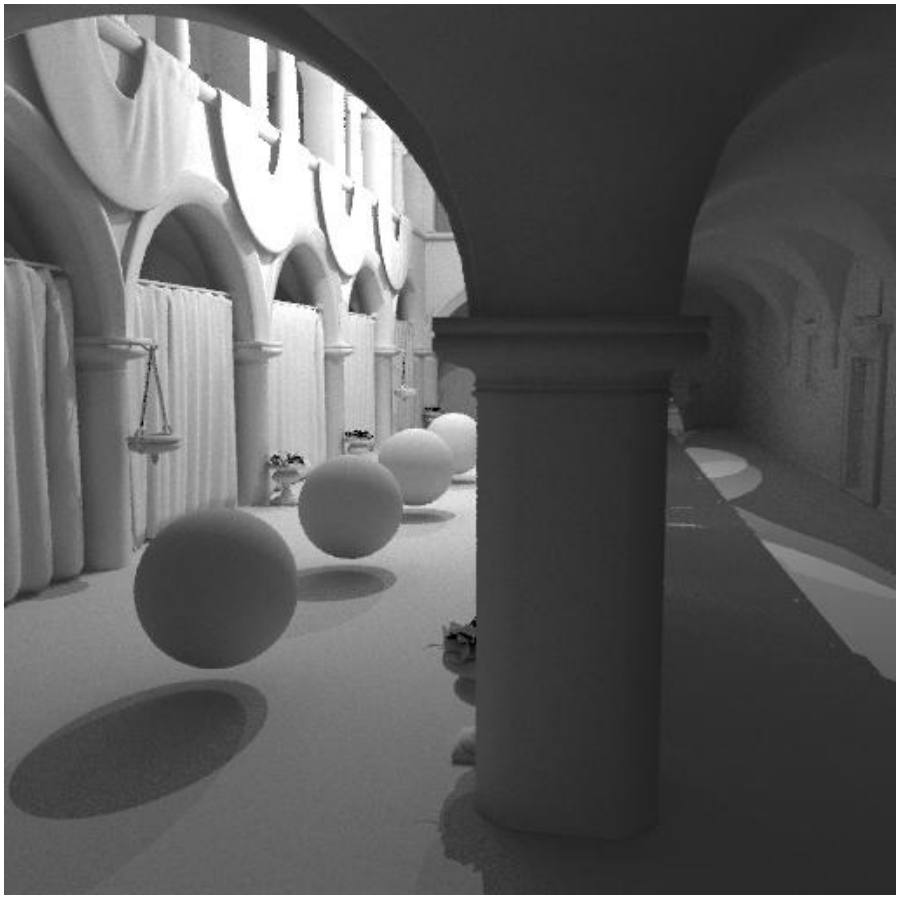
Distributed Raytracing – Indirekte Beleuchtung



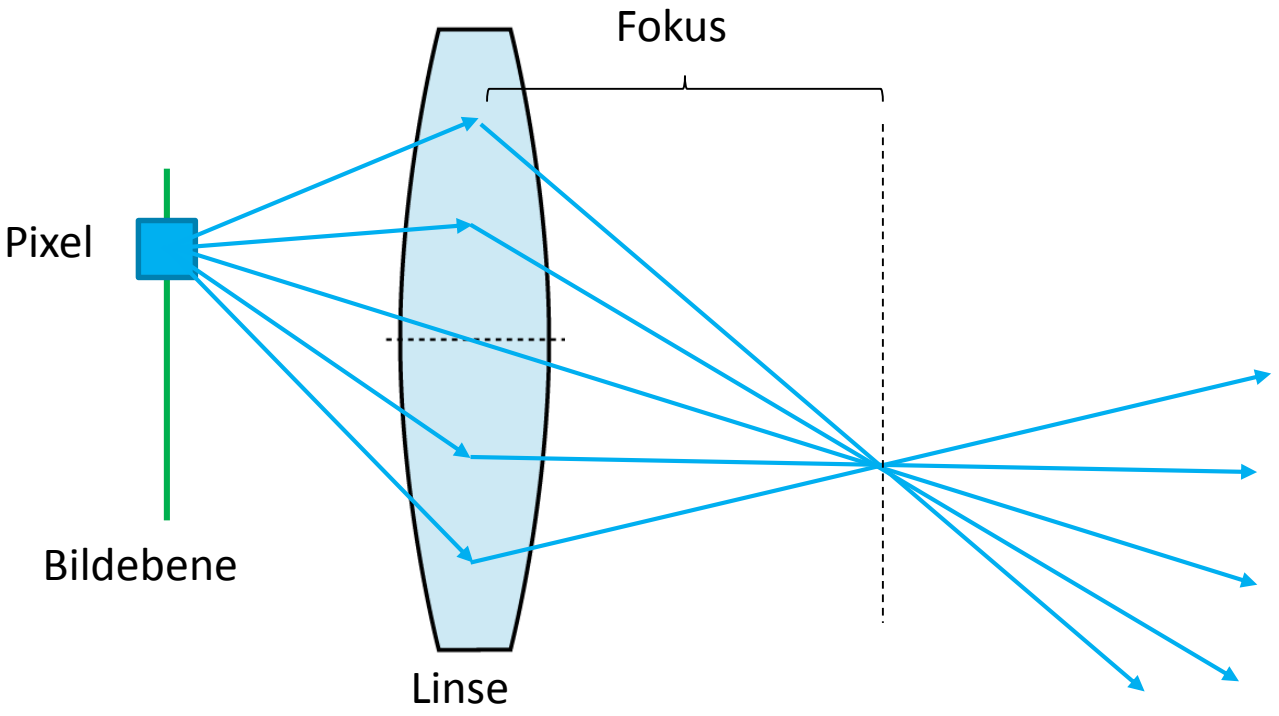
Distributed Raytracing – Indirekte Beleuchtung



Distributed Raytracing – Indirekte Beleuchtung

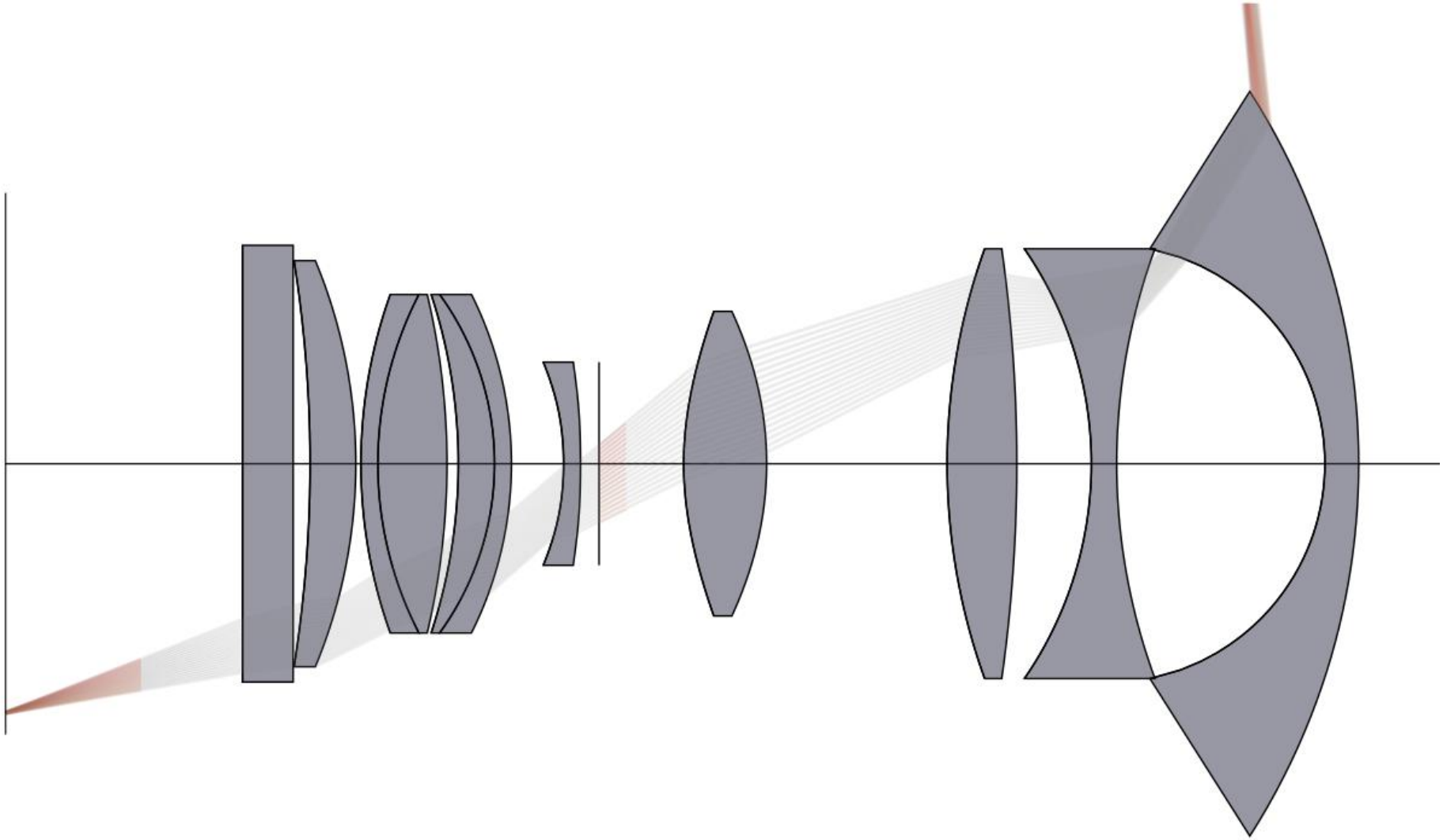


Distributed Raytracing - Tiefenunschärfe



Distributed Raytracing - Tiefenunschärfe

Realistische Linsen

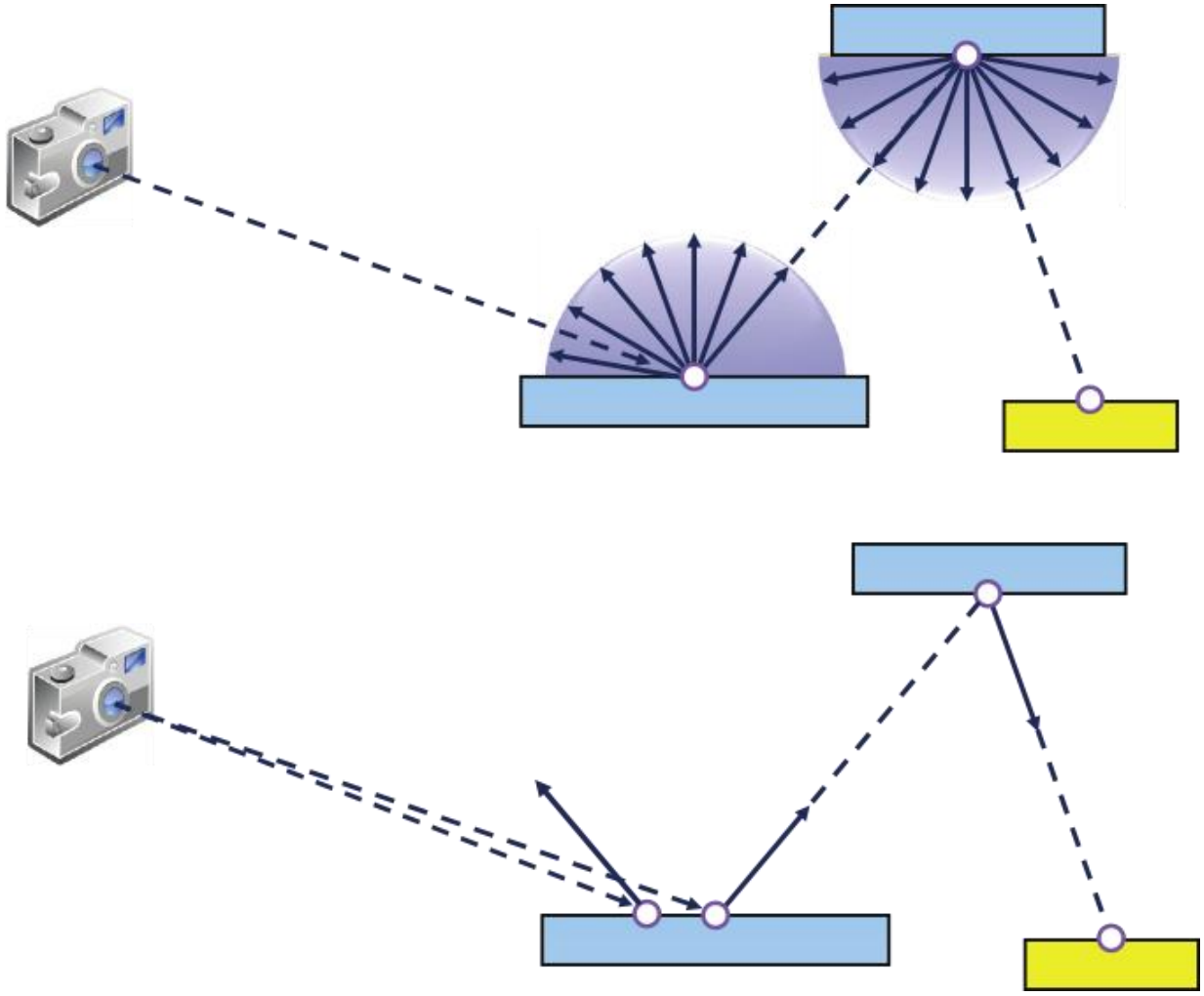


Realistische Linsen

Realistische Linsen

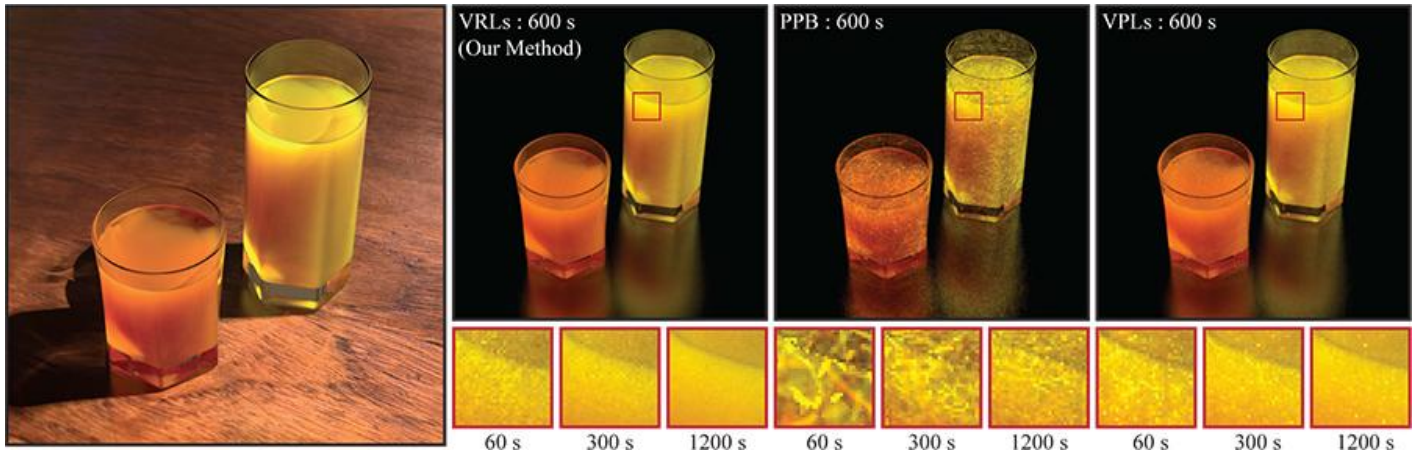
Distributed Raytracing → Pathtracing

- Strahlenbaum von DT “explodiert”
- Lösung Pathtracing

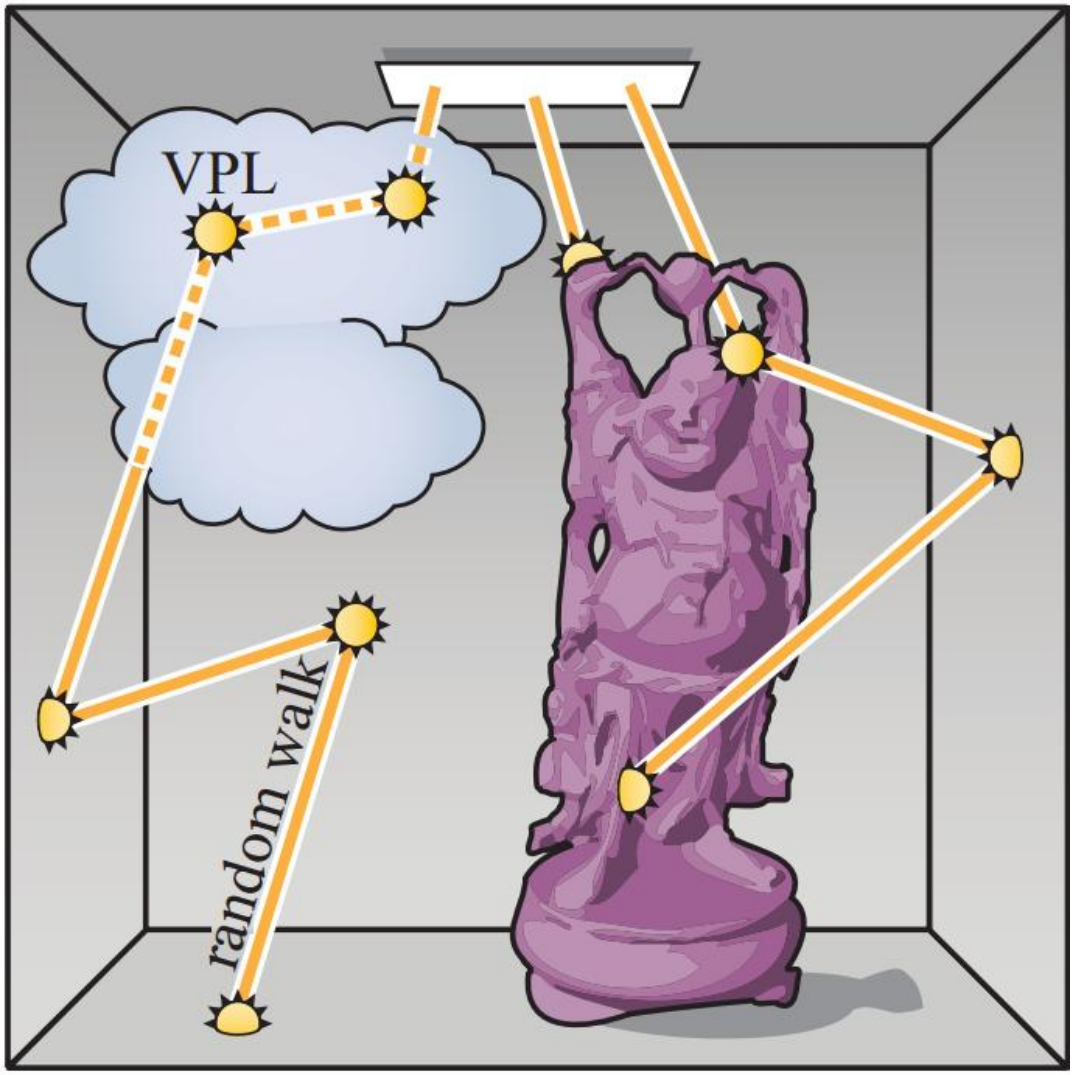


Partizipierende Medien

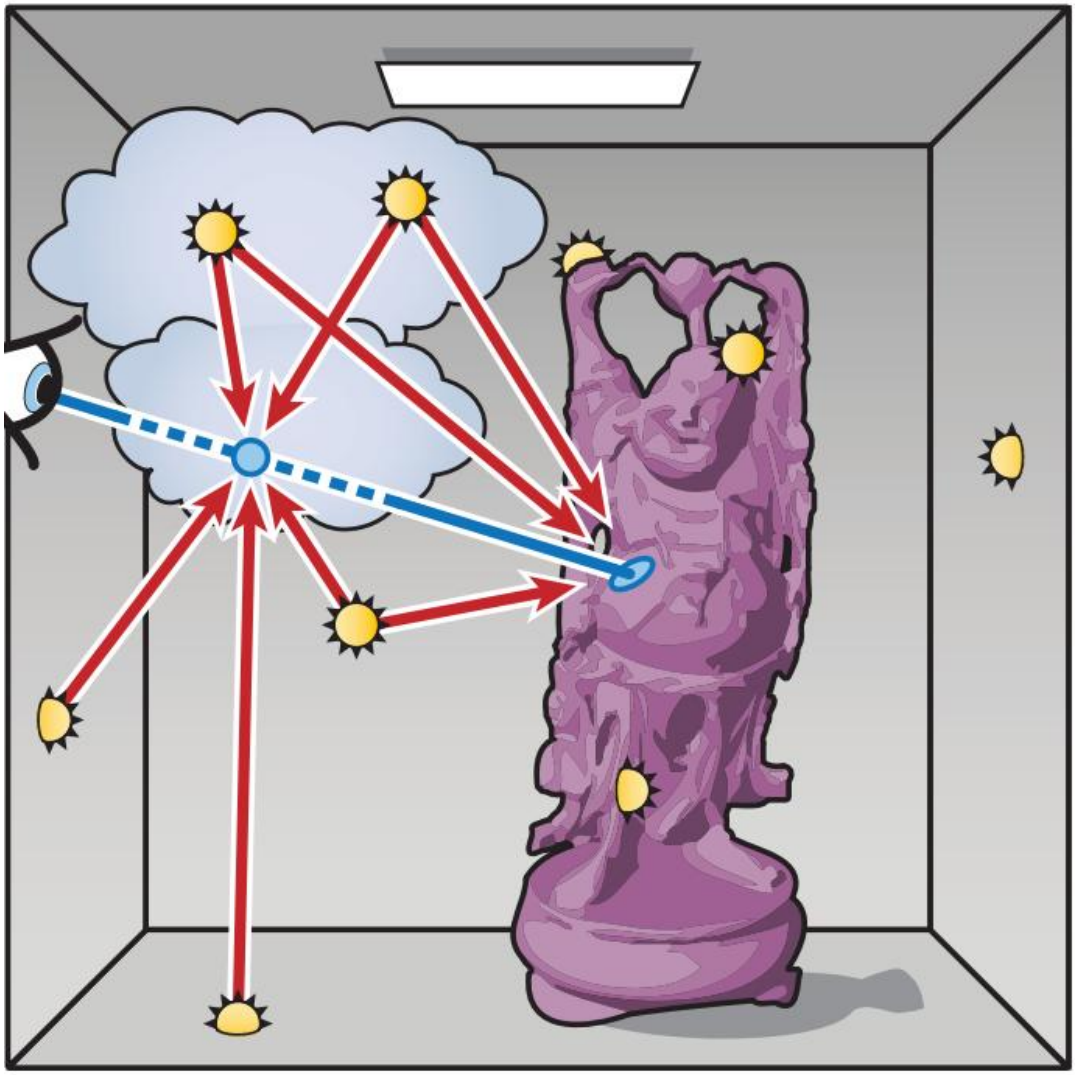
- Aufwendige Berechnung



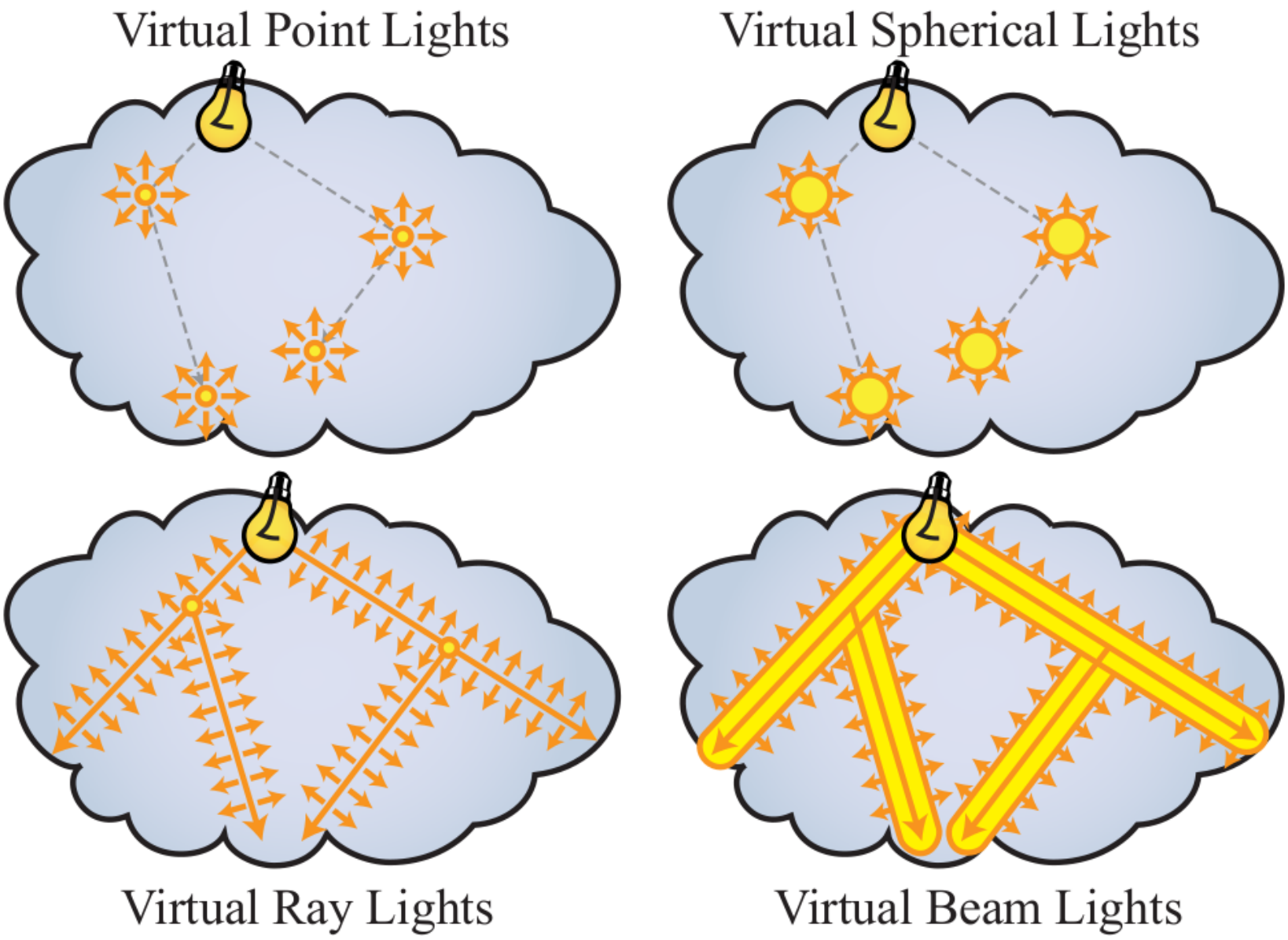
Many-Light Methods



Many-Light Methods



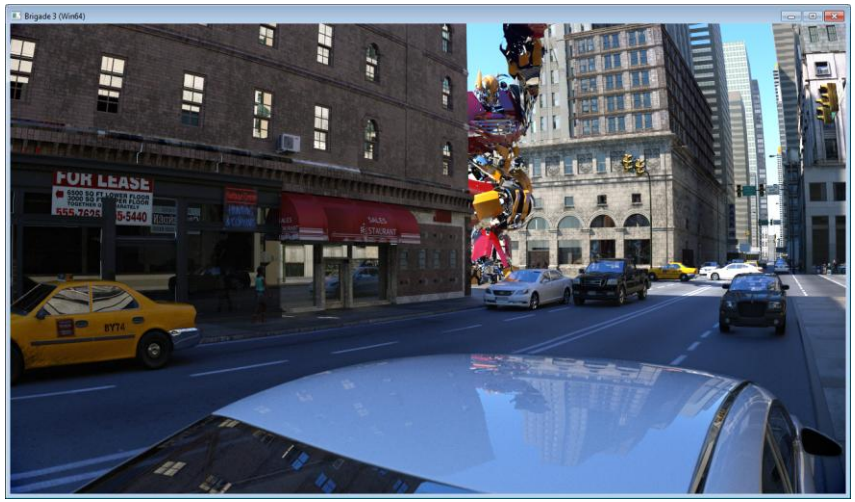
Many-Light Methods

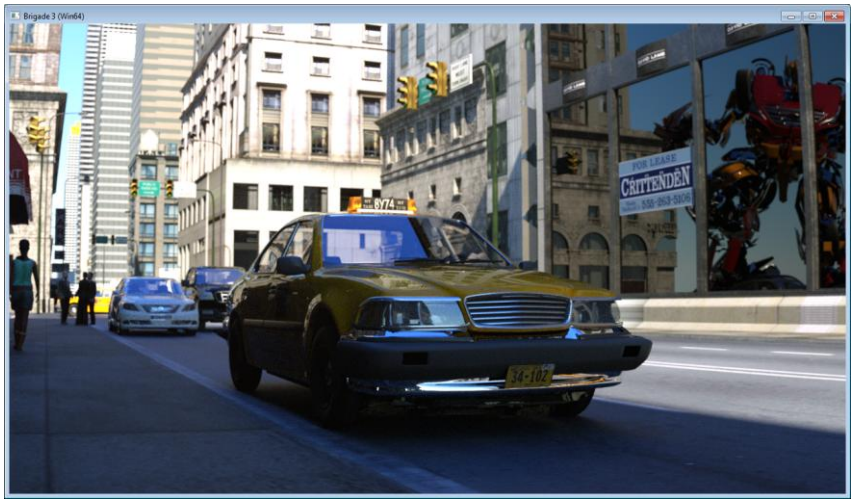


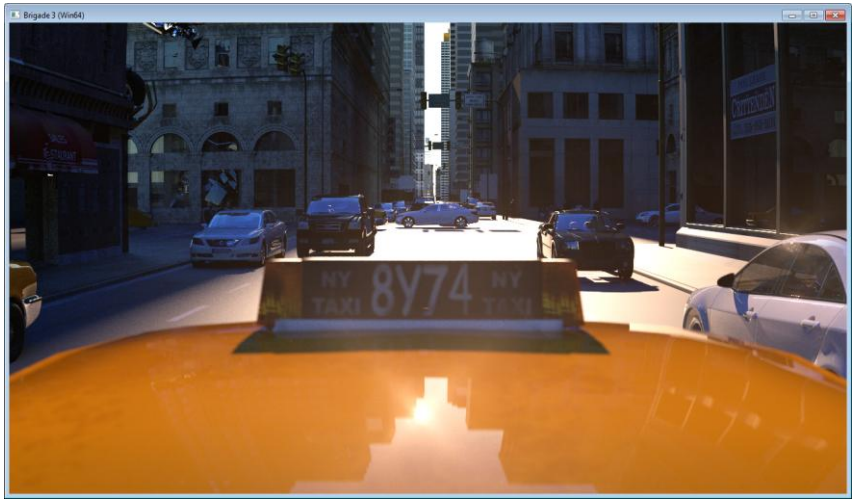
Demo Real-time Ray Tracer

- https://www.shadertoy.com/view/4sfGDB

Interactive Path Tracing







2x Geforce Titan , 0.5 s – 1 s
360 spp (12 spp or 11 Mio paths @ 30 fps)

Fixed-function Hardware Today?

- Fixed-function Operations
  - Ray / AABB
  - Ray / Triangle
- 44x weniger Fläche
  - Energie …
  - Pipelining …

www.gdcvault.com/play/1020741/New-Techniques-Made-Possible-by

HW-accelerated Ray Tracing

- PowerVR mobile GPUs 2014
  - 300 Mio rays / second (@O(1) Watt)
- x5 für gleiche Performance
- x150 für gleiche Qualität in Real Time

HW-accelerated Ray Tracing

- PowerVR mobile GPUs 2014
  - 300 Mio rays / second (@O(1) Watt)

- x5 für gleiche Performance
  - 5 Jahre? (GFLOPS)

- x150 für gleiche Qualität in Real Time
  - 15 Jahre?! (GFLOPS)

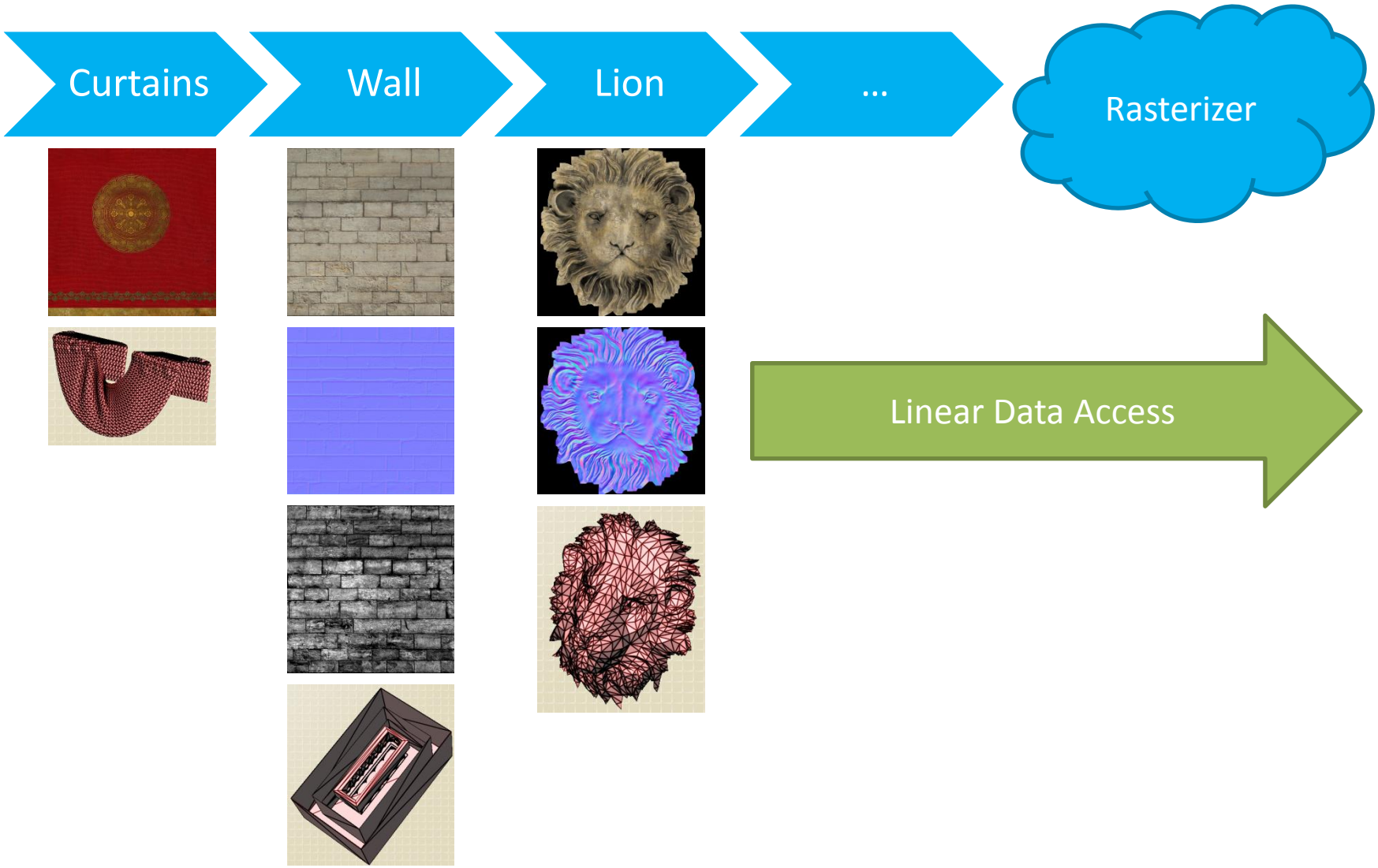
Reine & naive Spekulation!

en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units

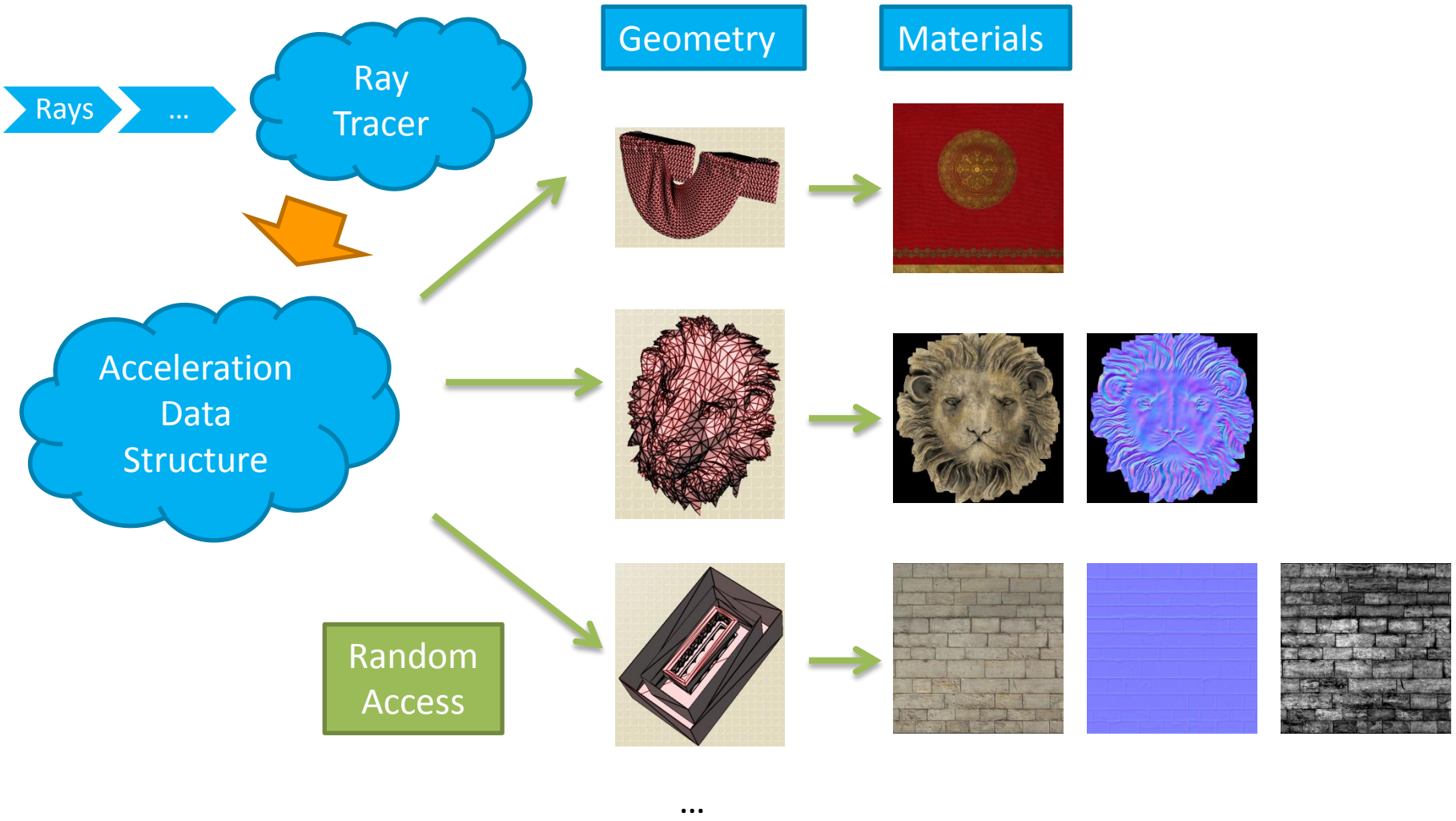
Ray Tracing Bottlenecks

- GFLOPS
  - Traversal
  - Intersection Testing
  - Sampling
  - BRDF
- **Bandwidth**
  - Traversal
  - **Material Data (Textures!)**

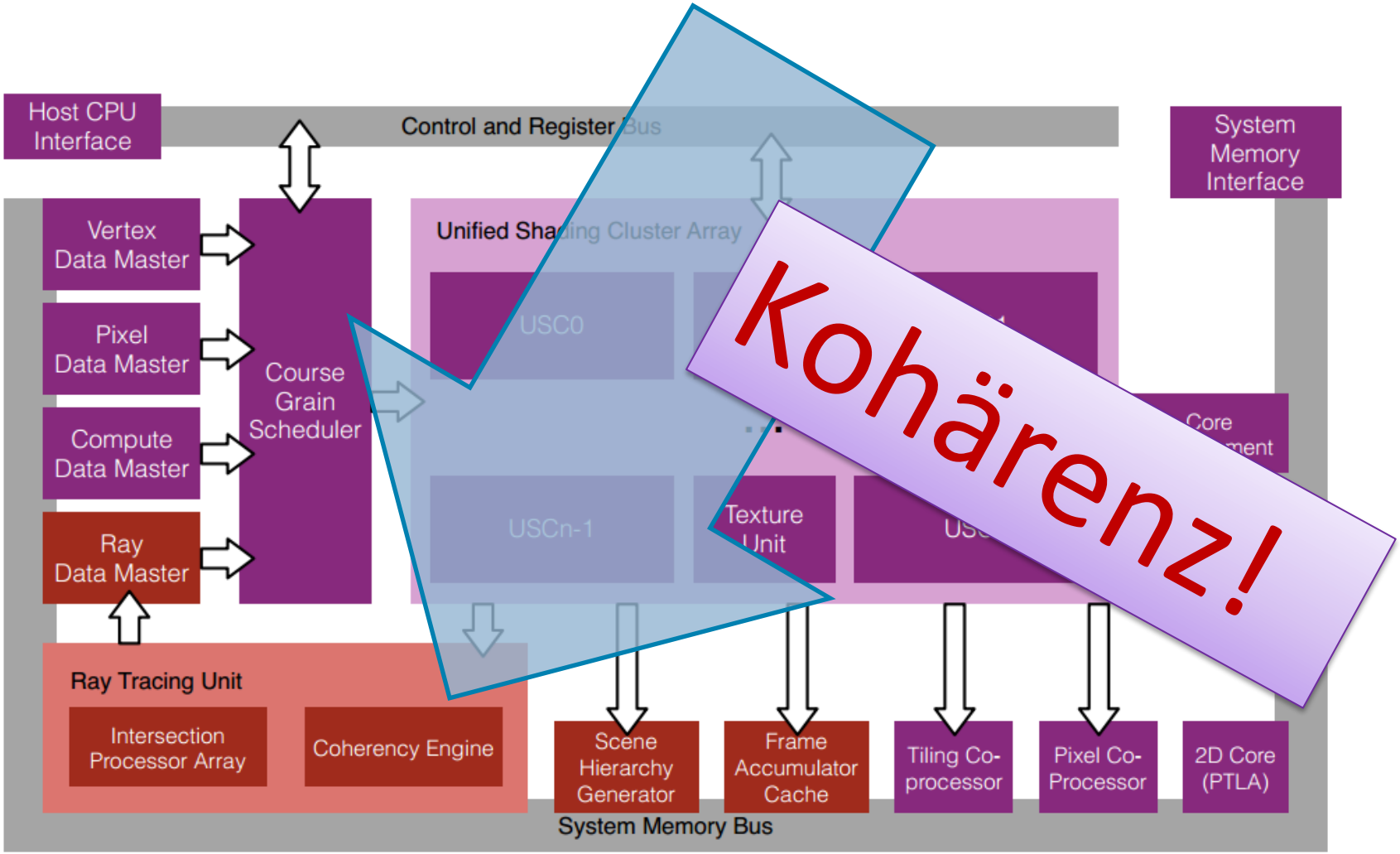
Rasterization Data Access



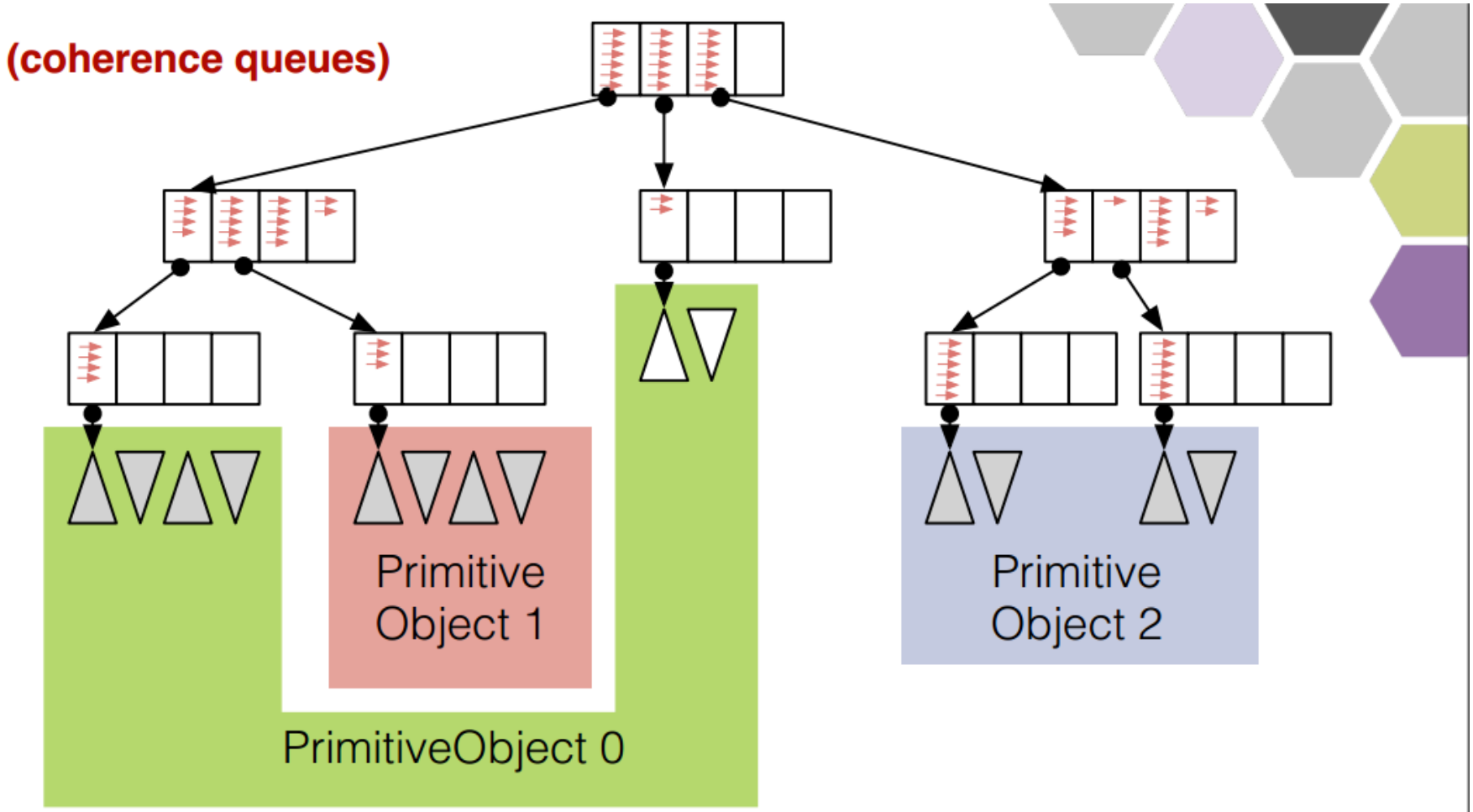
Ray Tracing Data Access



PowerVR mobile GPU 2014



Online Ray Reordering: Traversal



HW-accelerated Ray Tracing

- PowerVR mobile GPUs 2014
  - 300 Mio rays / second (@O(1) Watt)

- x5 für gleiche Performance
  - 5 Jahre? (GFLOPS)

- x150 für gleiche Qualität in Real Time
  - 15 Jahre?! (GFLOPS)

Reine & naive Spekulation!

en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units

HW-accelerated Ray Tracing

- PowerVR mobile GPUs 2014
  - 300 Mio rays / second (@O(1) Watt)

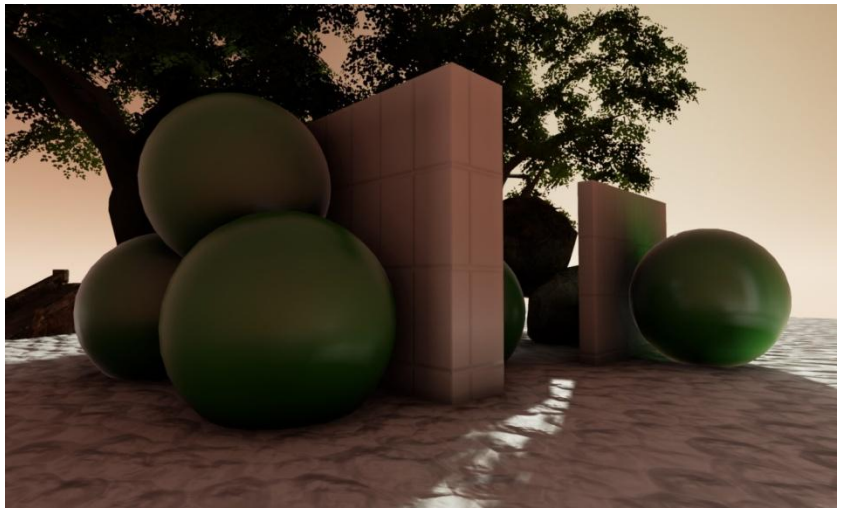
- x5 für gleiche Performance
  - 9 Jahre? (Bandwidth)

- x150 für gleiche Qualität in Real Time
  - 27 Jahre?! (Bandwidth)

Reine & naive Spekulation!

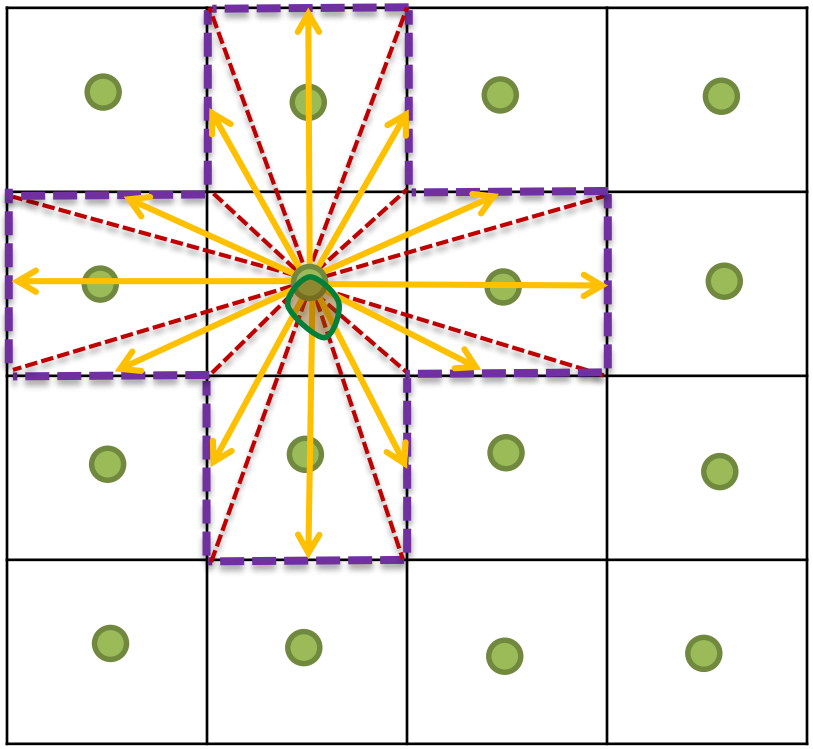
en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units

Light Propagation Volumes

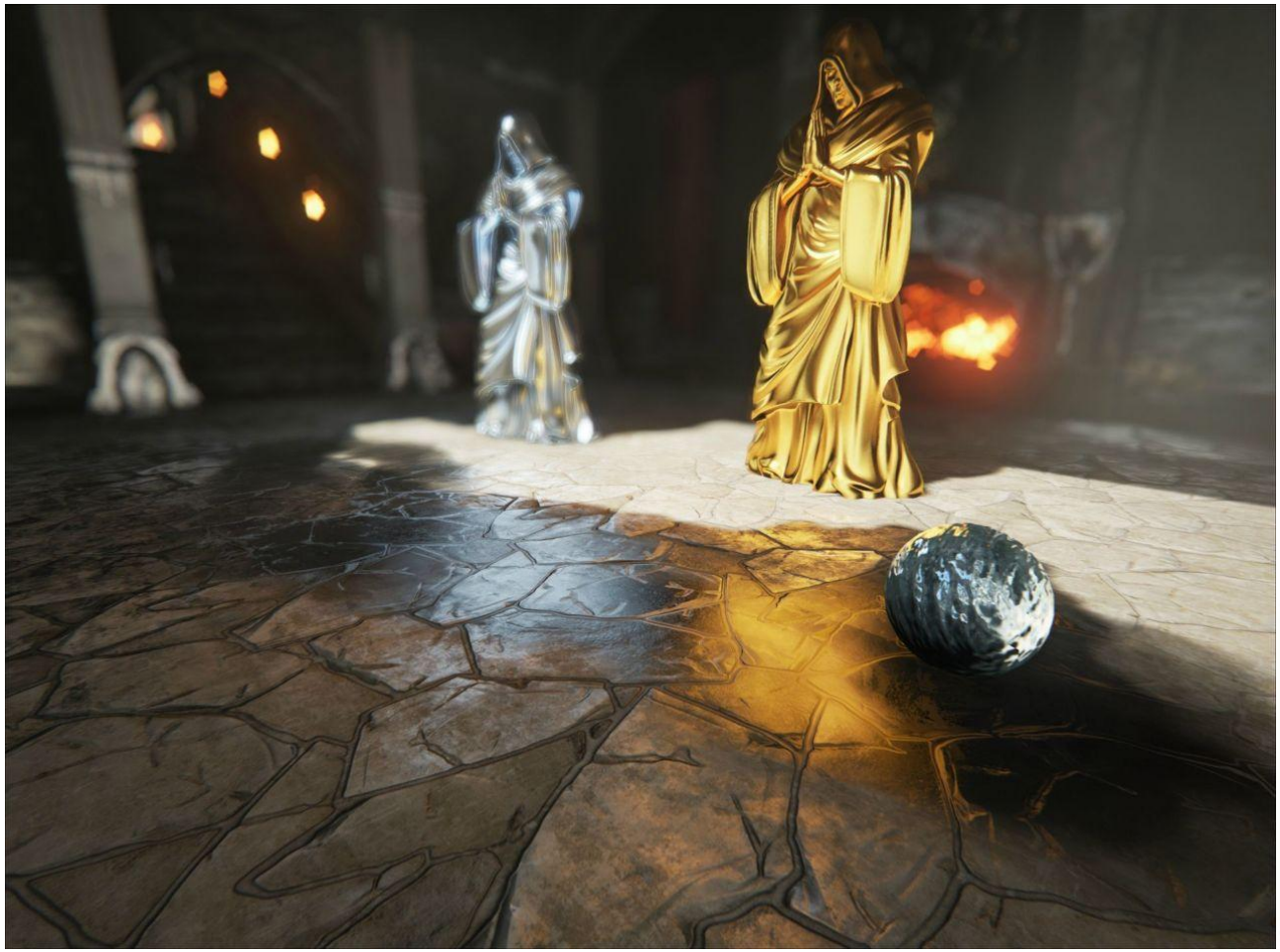


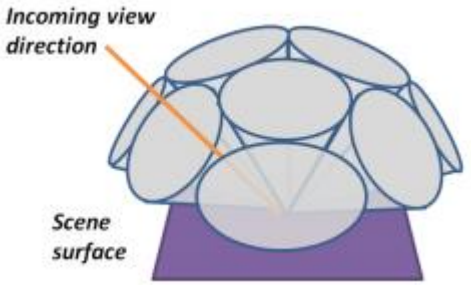
https://www.youtube.com/watch?v=0OPOyiRapsQ

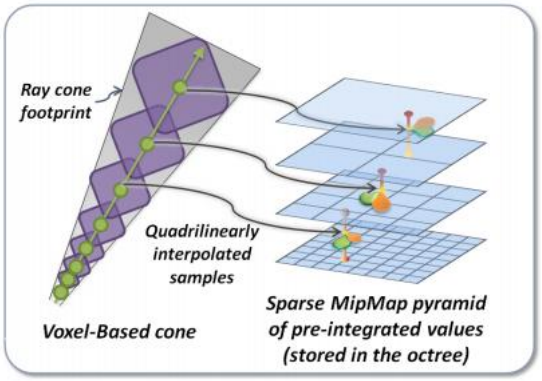
Light Propagation



Voxel Cone Tracing





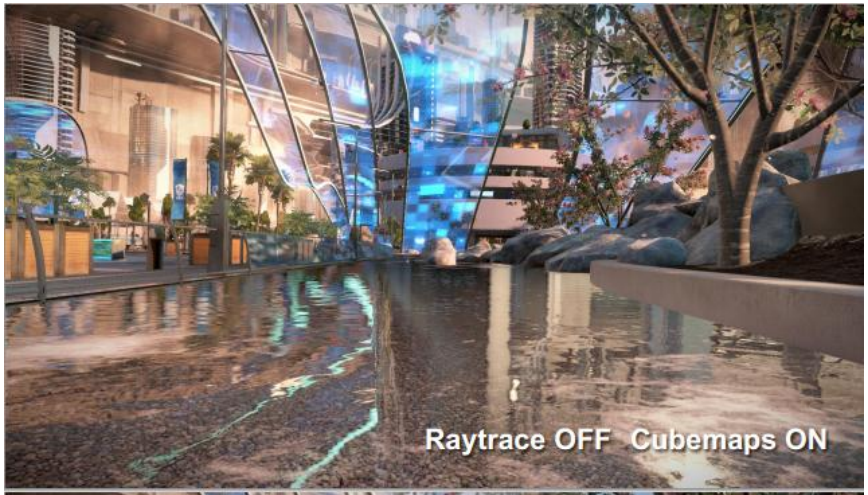


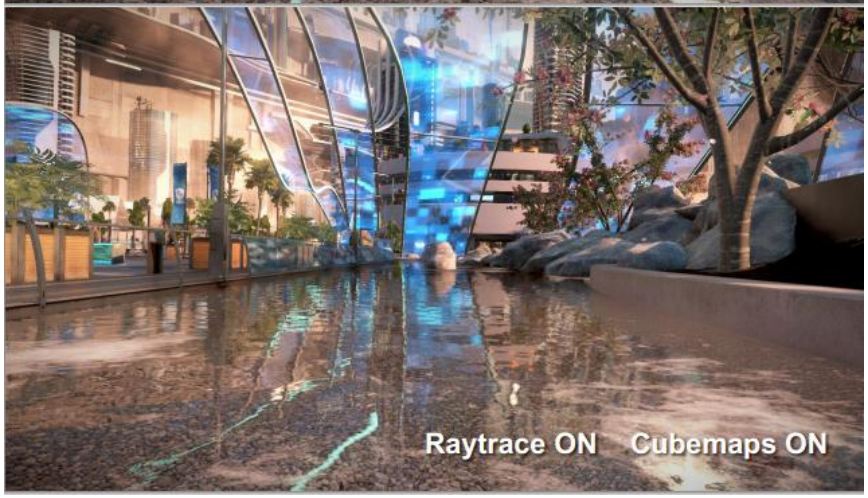
Komplexität Rasterisierung?

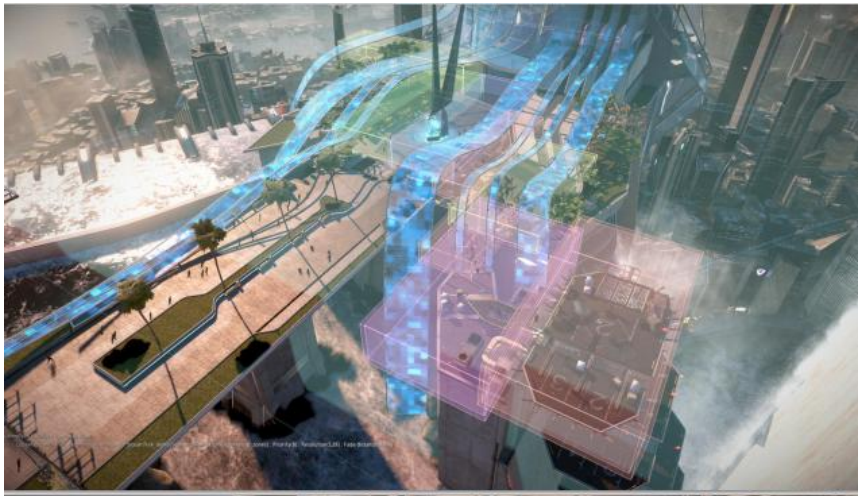
„ Modern rasterization pipelines are already disgustingly complicated... Now we are excited to rough voxelize our scene and add another shading stage? Wake me up in 5 years, i dont want to work with this nonsense“

- Das Internet

Screen-Space Methoden







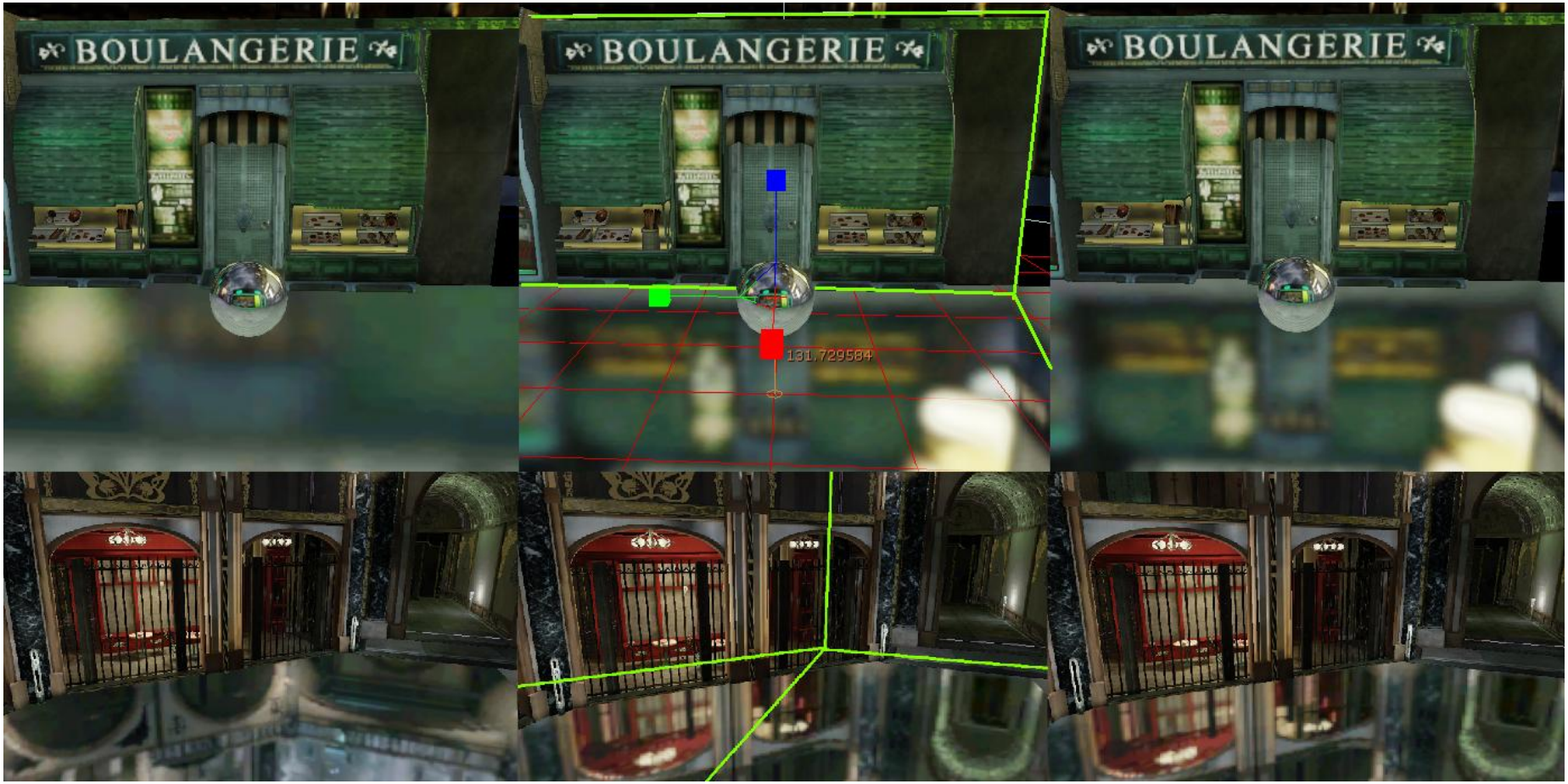
Screen-Space Methoden

- Lokale Lösung für globales Problem?!

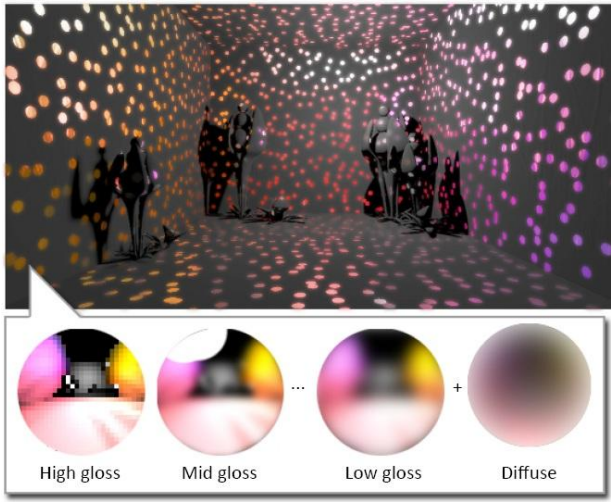
- „Nur“ adaptive Detailverbesserung
  - Mehr Detail für unmittelbar Sichtbares (lokal)
  - Rest muss anders approximiert werden (global)

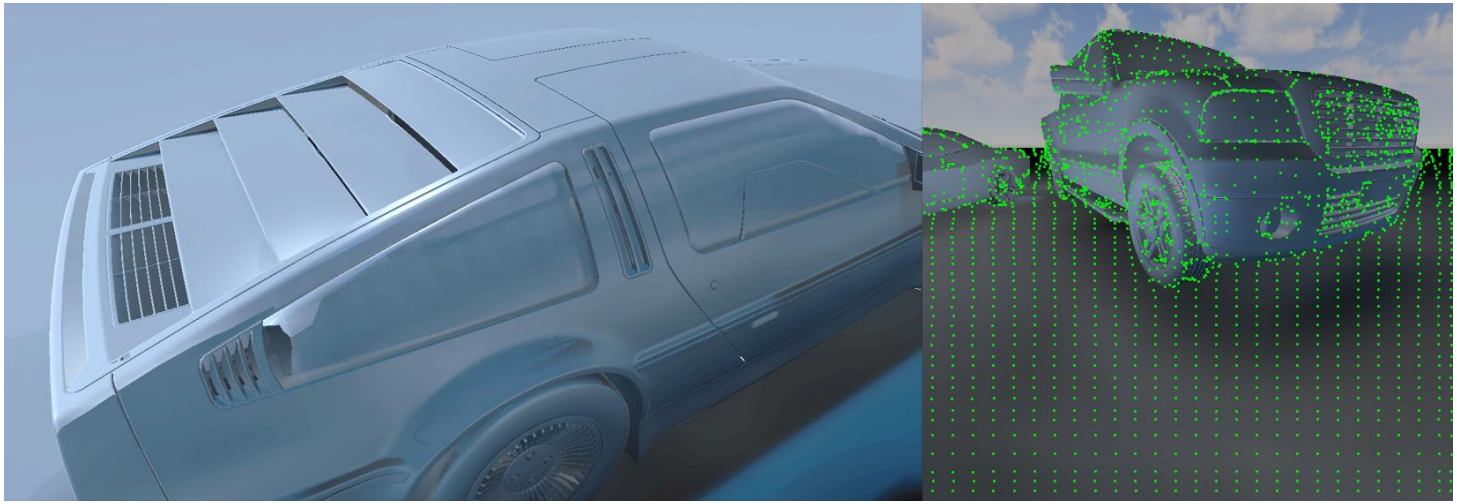
- Ohne globalen Teil unvollständig & instabil
- Globale Approximation -> andere Methoden

Image Proxies



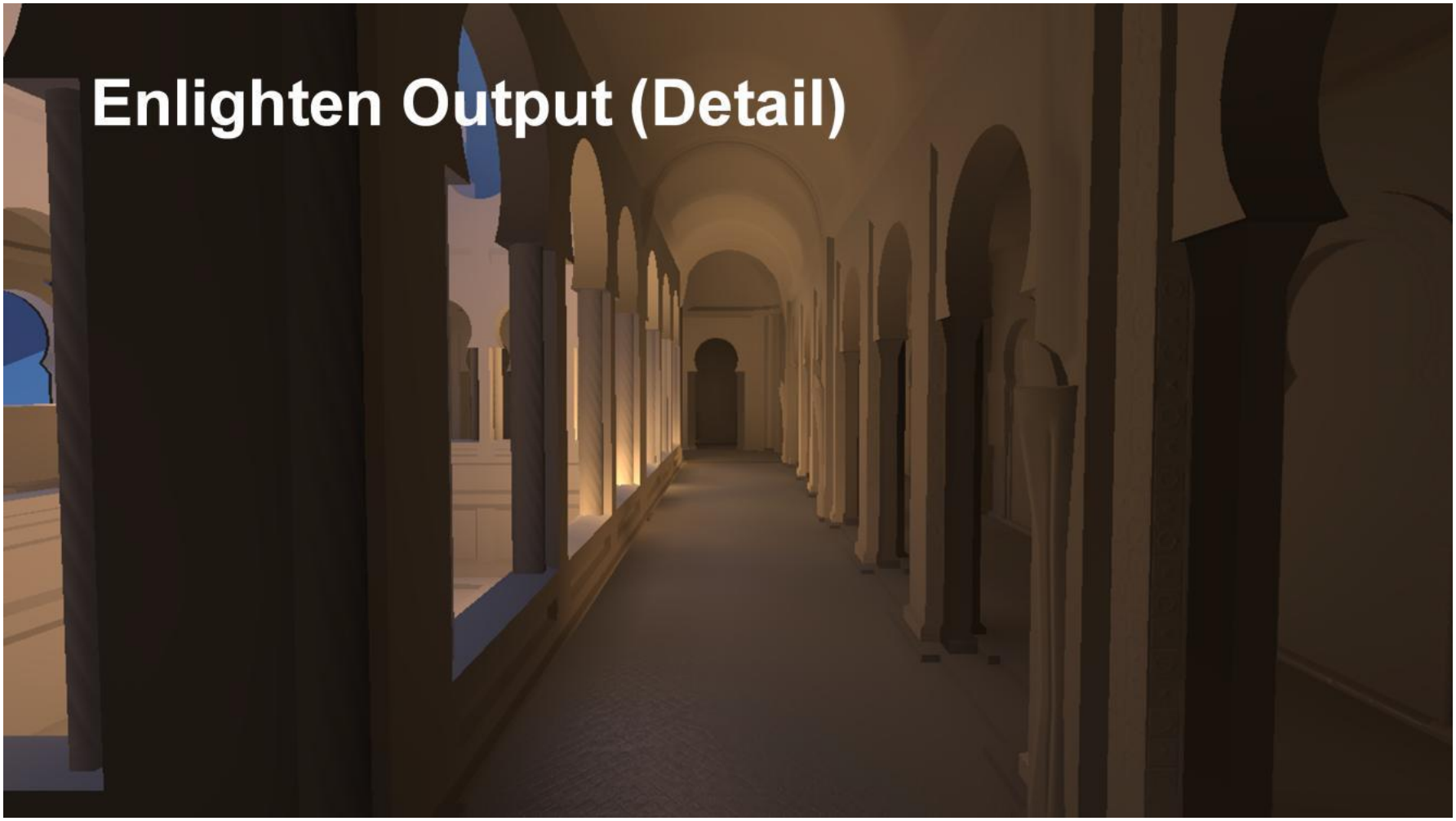
Radiance Caches



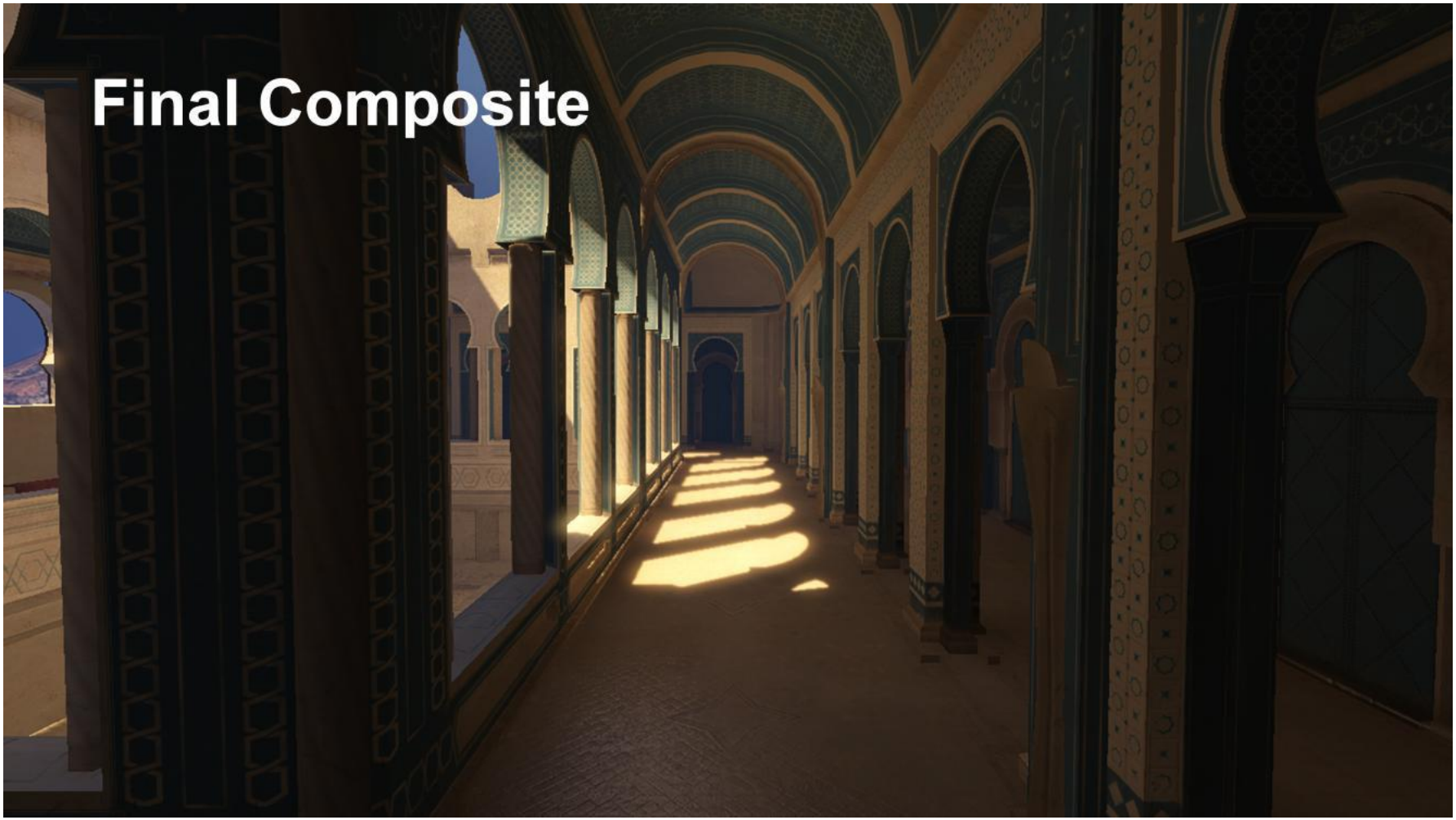


Finite Elemente / Radiosity

Finite Elemente / Radiosity



Finite Elemente / Radiosity



Komplexität Ray Tracing

- Hierarchien und Algorithmen
  - Geometrie, Objekte, Lichter
- Out-of-order Execution und Scheduling
  - Hit Test & Shading Reordering für maximale Kohärenz
- Speicher und Kompression
  - Hierarchien
  - Coherency Queues
  - Aktive Strahlen/Pfade mit Rekursionsstapel

Komplexität Ray Tracing

- Hierarchien und Algorith
  - Strahlen implizit geordnet in Bildraum-Hierarchie
    → Dreiecke durchwandern „Strahlenbaum“
- Out-of-order Executio
  - Object-order erzwingt i.d.R. Minimum an Kohärenz
    Shading Reordering f
    Tatsächlich: Einordnen von Dreiecken in Bildschirm-Tiles
    → „Coherency Queues“ im „Strahlenbaum“!
- Sp Kompression
  - Hierarchien
  - Coherency Queue
  - Weniger Daten, kürzere Aufbewahrungsdauer
    → Einmal rasterisiert, dann nicht länger benötigt
    sionsstapel