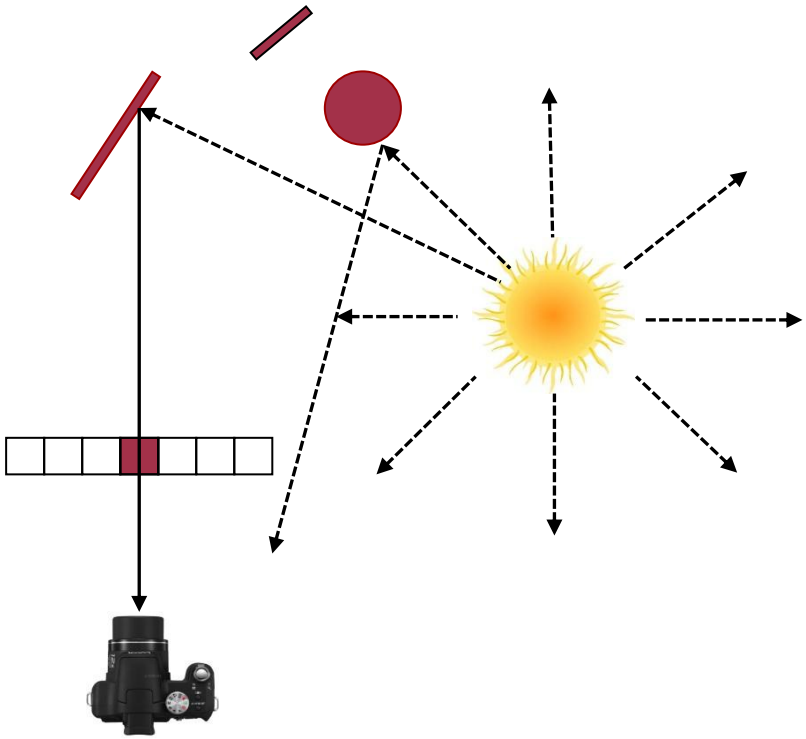
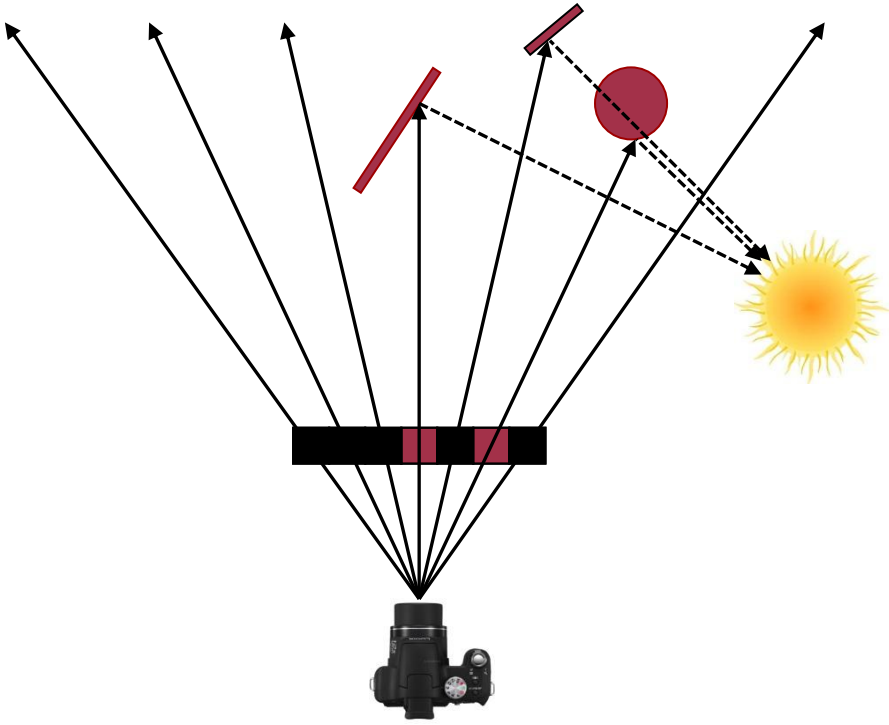
Computergrafik

Universität Osnabrück, Henning Wenke, 2012-07-09

Noch Kapitel XVI





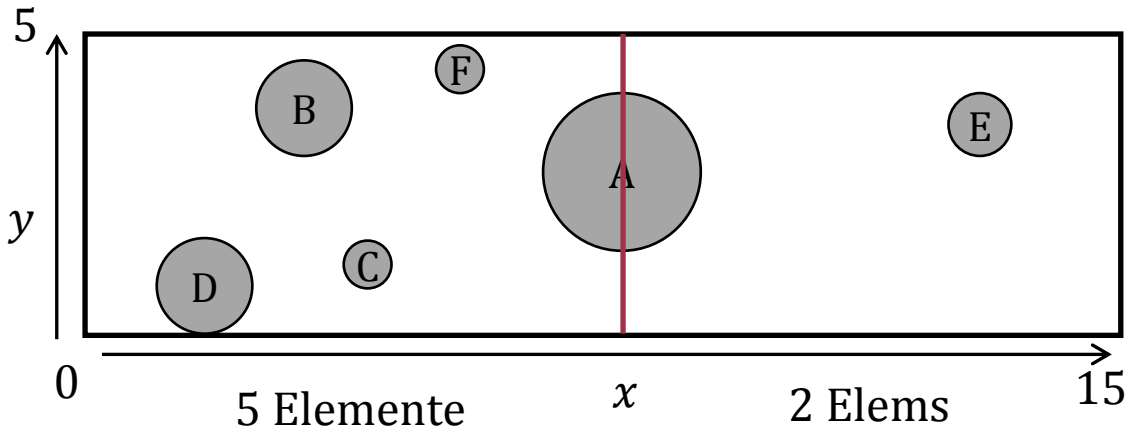
Realtime Ray Tracing

KD-Tree: Surface Area Heuristic

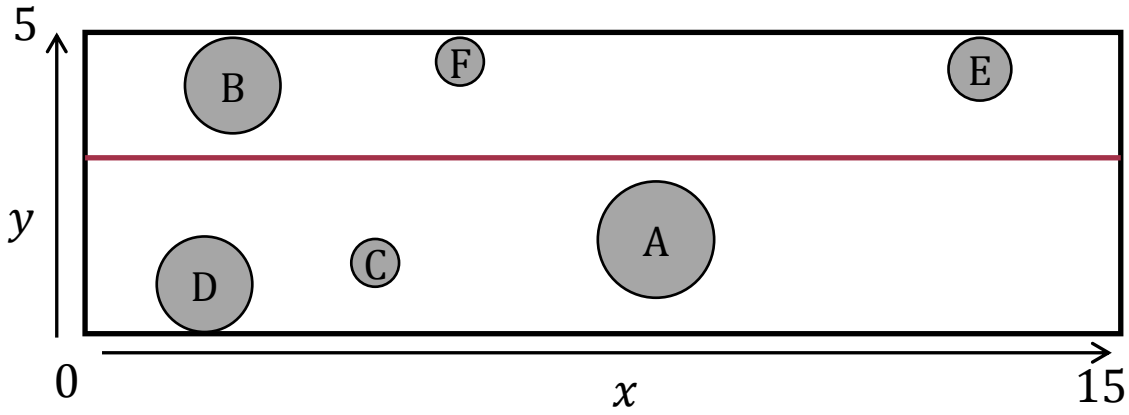
- Ziele der Aufteilung in Child Nodes:
  1. Möglichst gleich viele Objekte in beiden
  2. Möglichst wenig Objekte, die in beide eingetragen werden müssen
  3. Idealerweise wäre es so, dass auch nächste Aufteilung optimal
- Wir nehmen jeweils Achse mit längster Ausdehnung und teilen in der Mitte
  - Grobe Schätzung nur zu Punkt 1
  - Dadurch Reihenfolge der Achsenteilungen fest
- Um beides modellieren zu können, speichern wir trotzdem pro Node:
  - Trennachse (1 Integer: 0:x, 1:y, 2:z)
  - Position der Trennebene (1 Float)
- Left Child: Kleiner als Trennachse
- Right Child: Größer als Trennachse
- Bounding Box Ansatz auch möglich, aber unnötig aufwendig
- Für Gesamtszene aber BB hilfreich

Beispiel

Unser Ansatz:

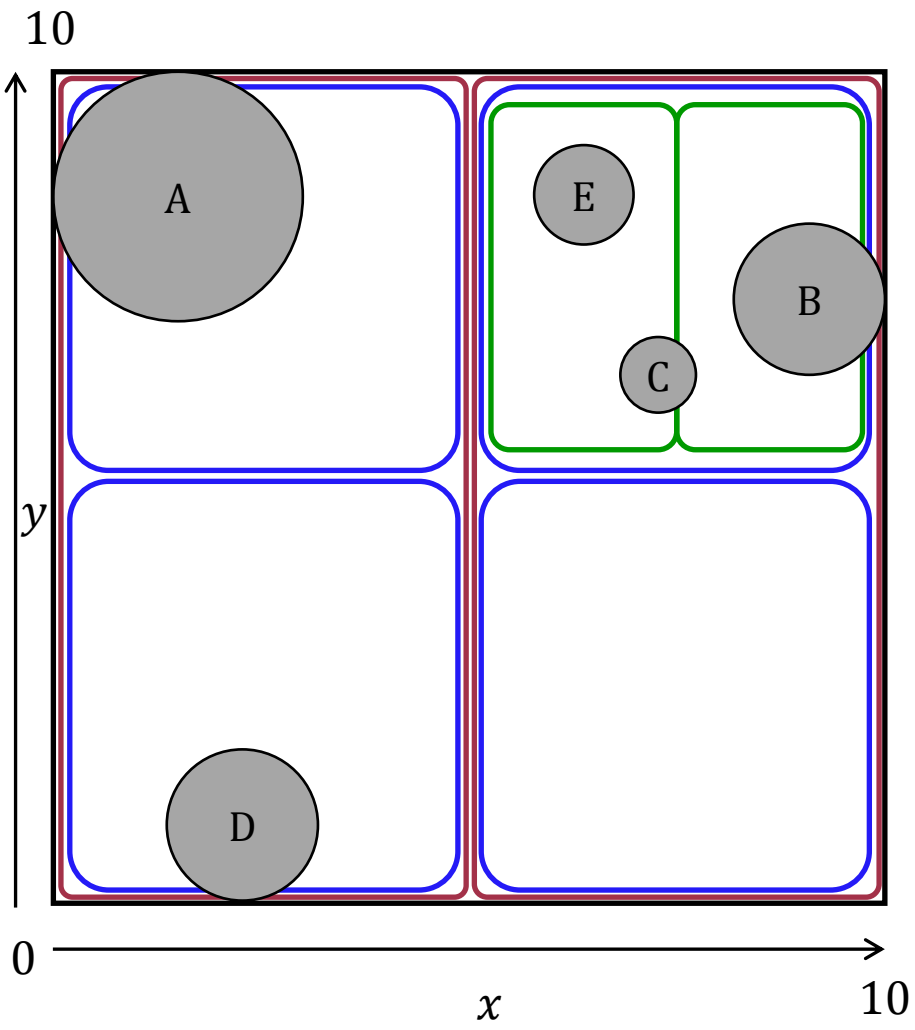


Besser:

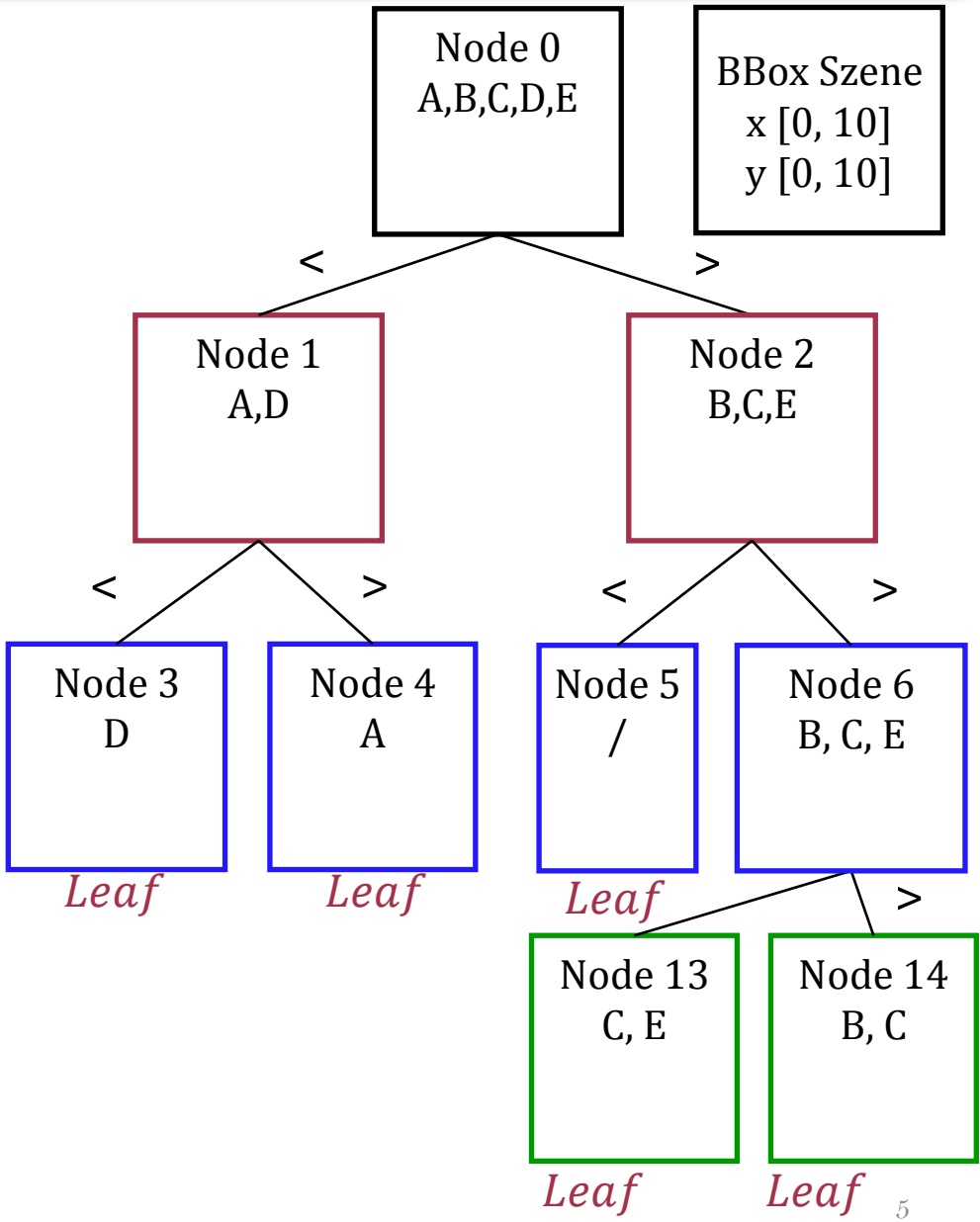


targetElementsPerNode 1, maxDepth 90000

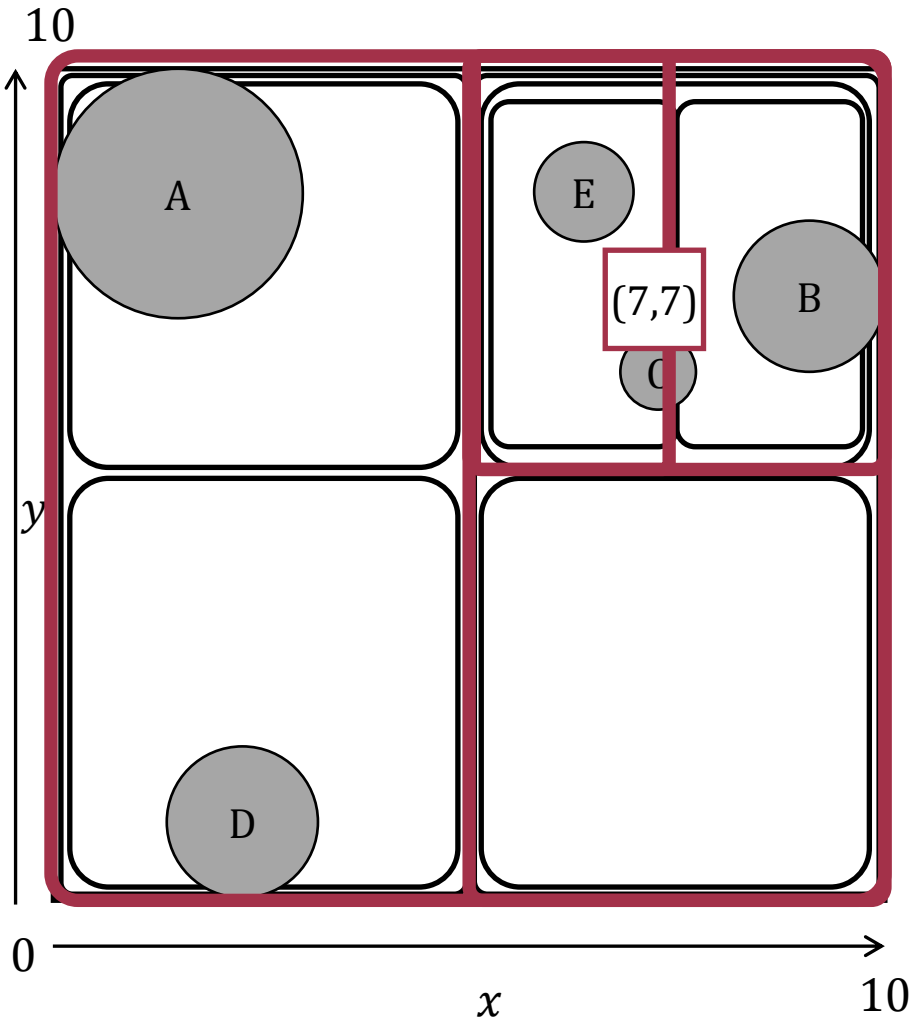
Beispiel: 2D-Tree Aufbau

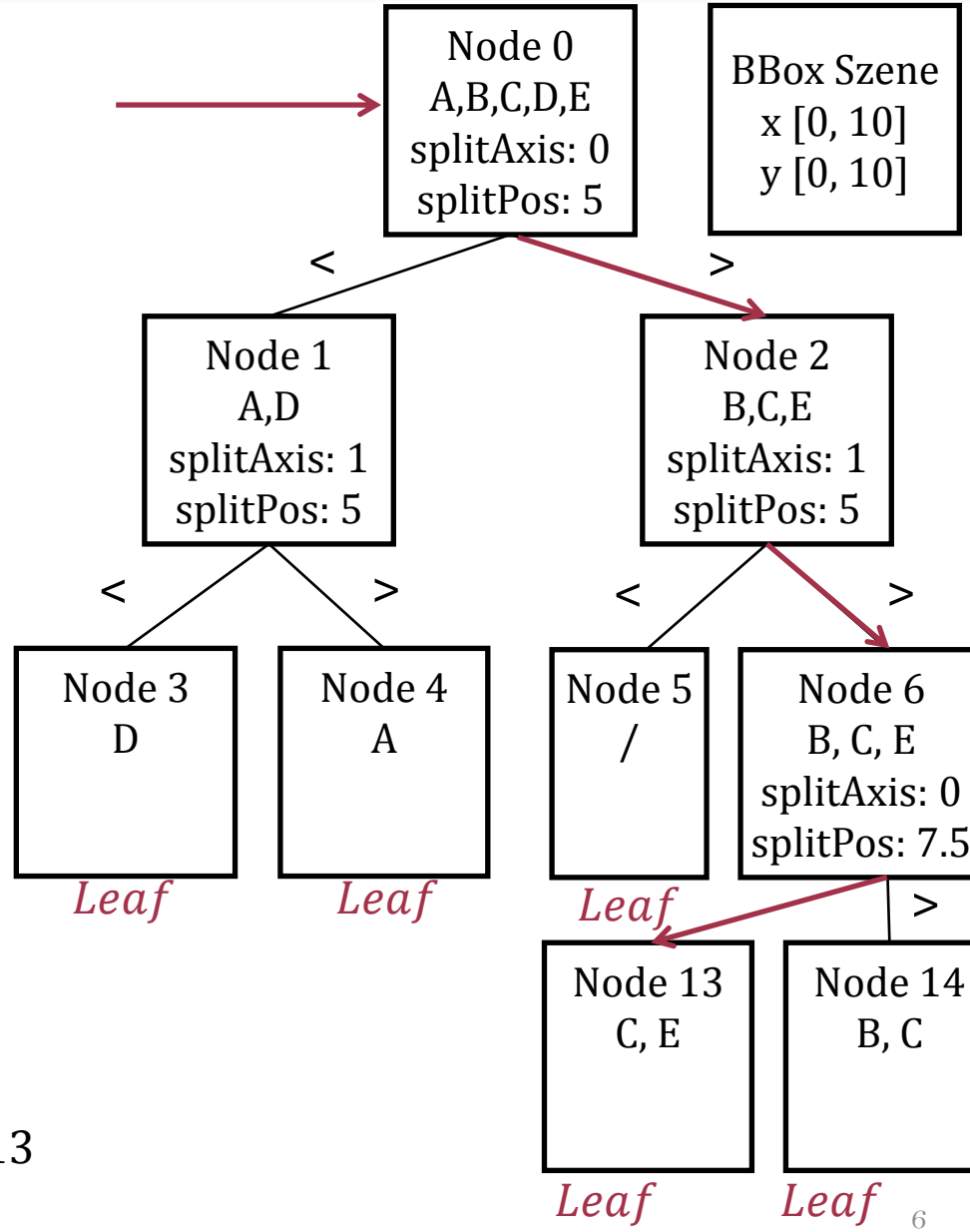


targetElementsPerNode 1
maxDepth 4



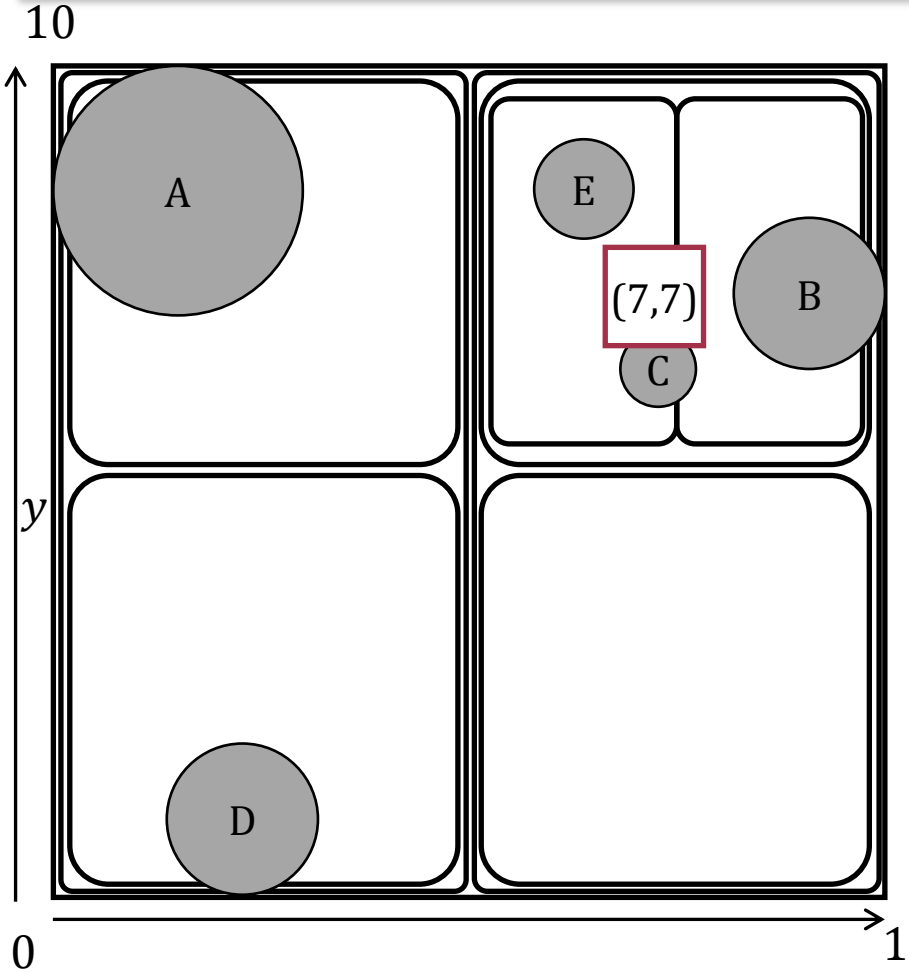
I. Suche Pixel (7,7) enthaltende Leafs

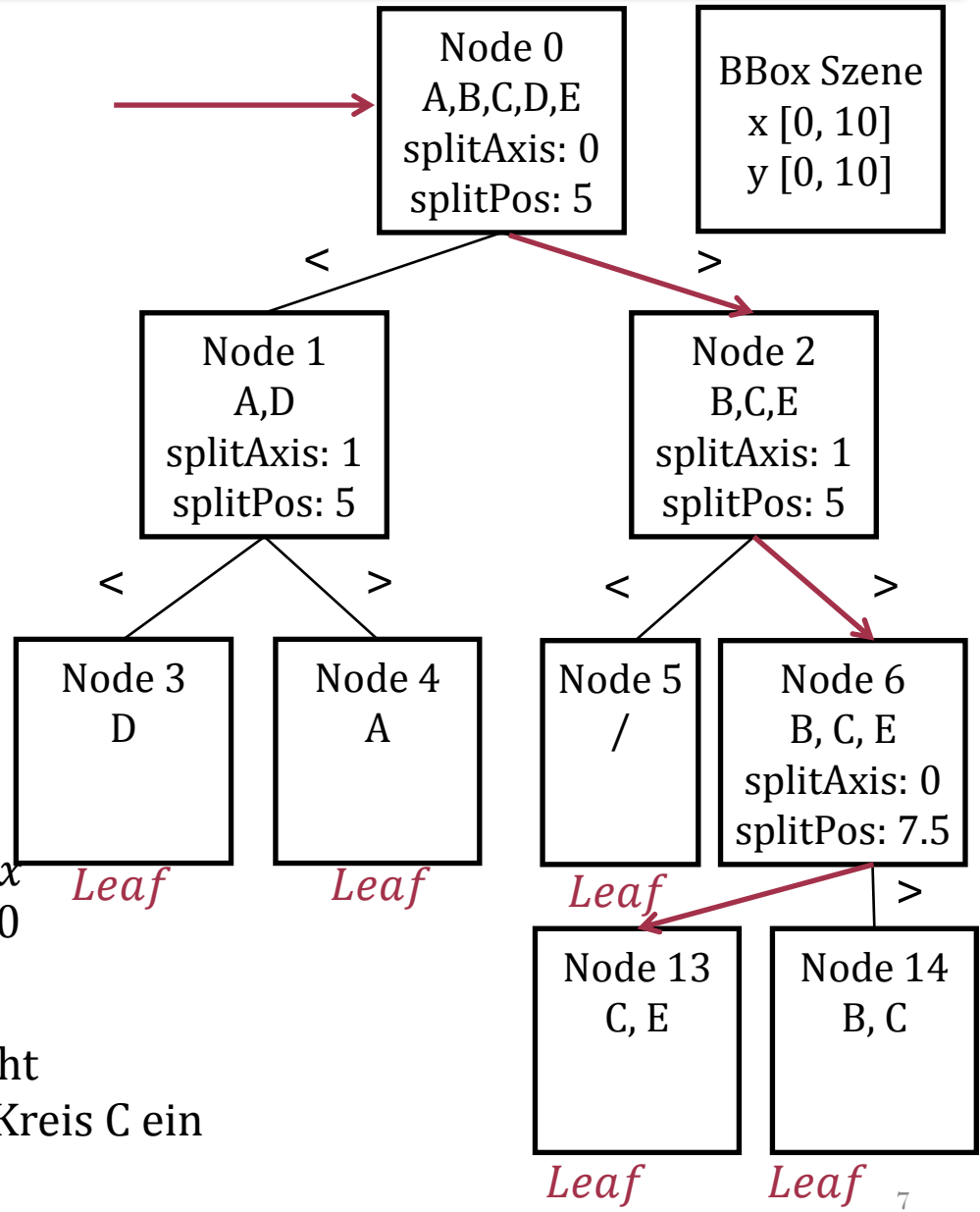




- Pixel (7,7) enthalten in Leaf-Node 13

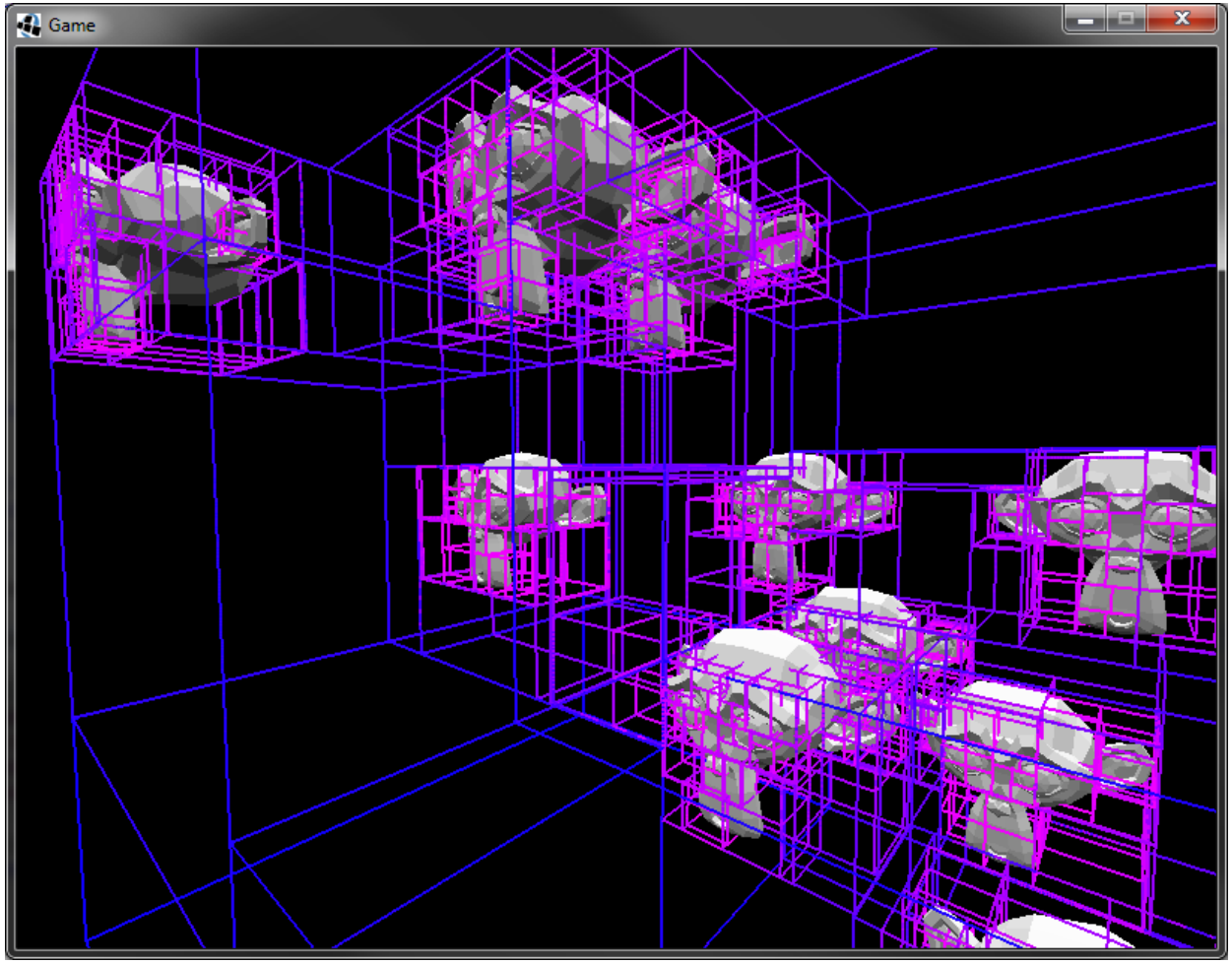
II. Durchsuche Objekte Node 13



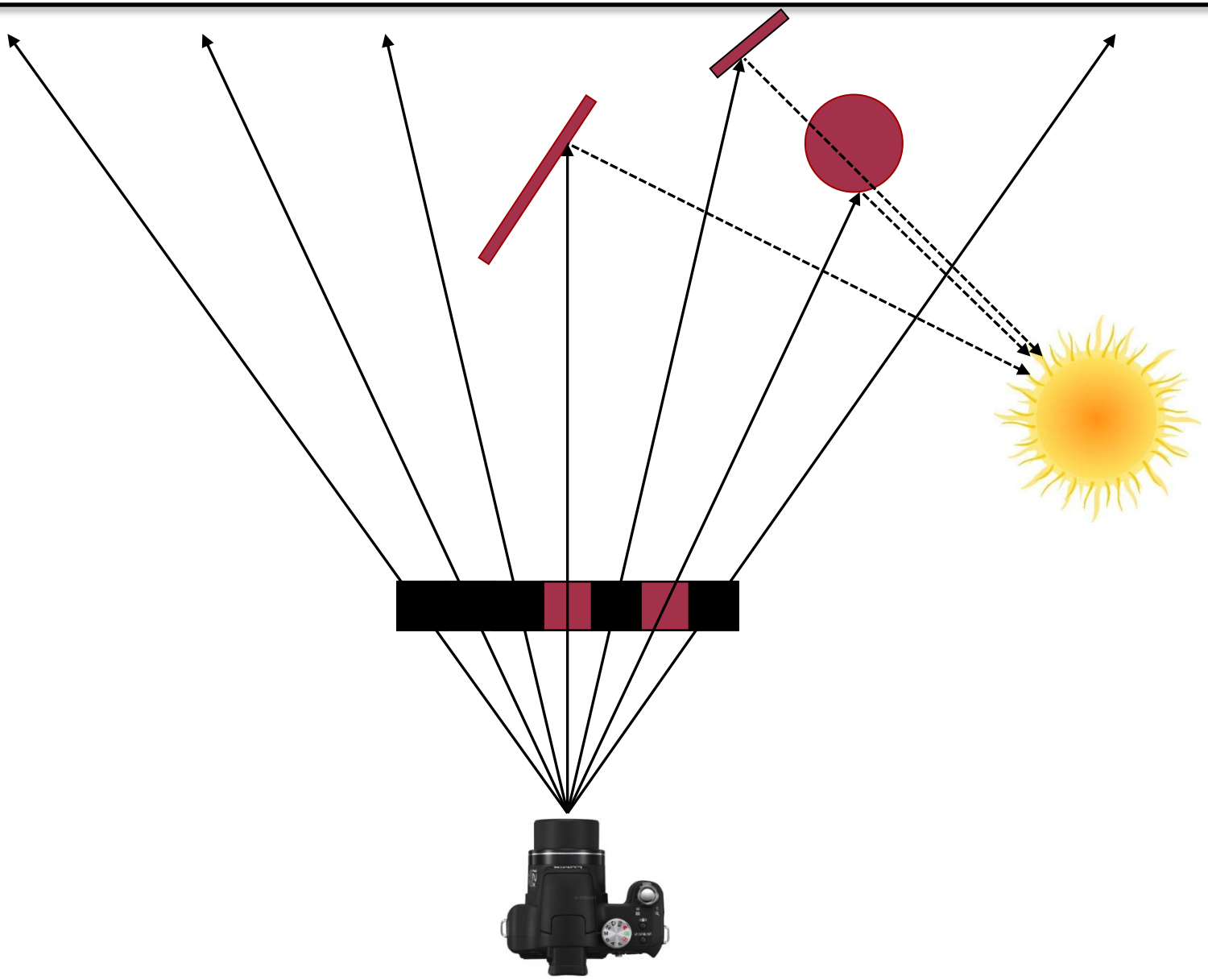


- Node 13: C überlappt Pixel
- Node 13: E überlappt Pixel nicht
- Also: Färbe Pixel (7,7) gemäß Kreis C ein

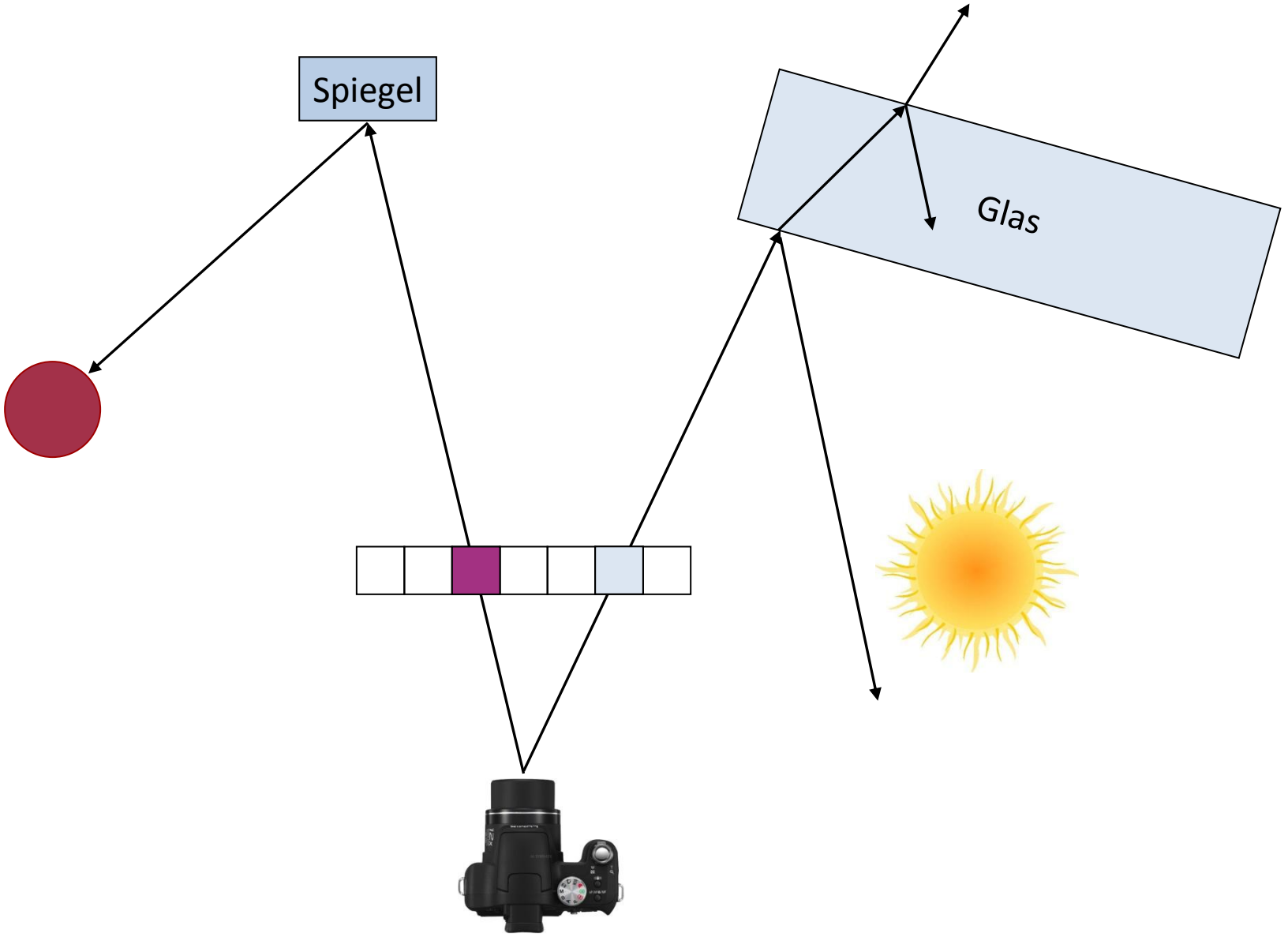
Beispiel



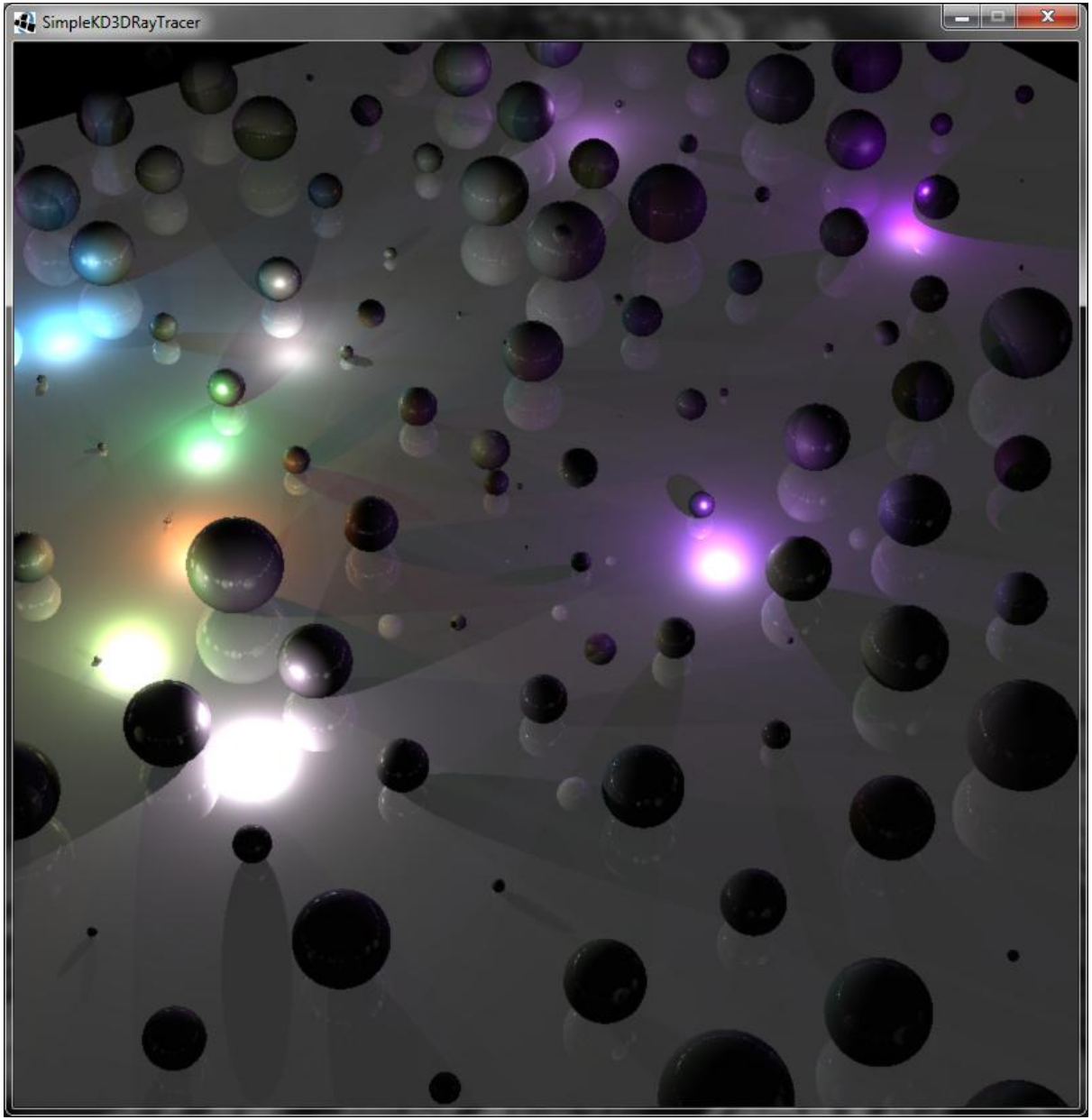
Ray Tracing mit Shadow Rays



Rekursives Ray Tracing



Beispiel



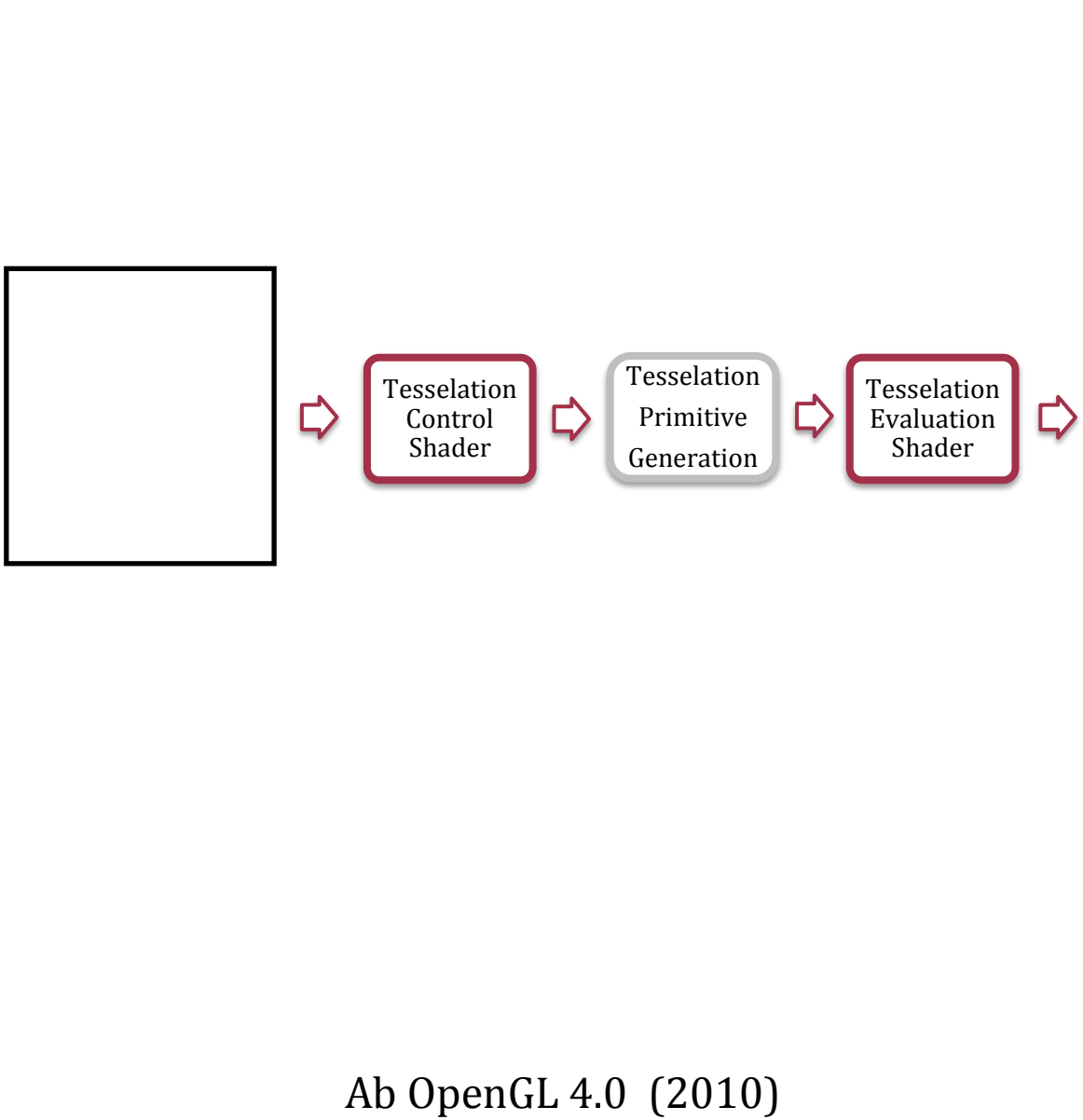
Ende Kapitel XVI

Ab hier nicht mehr Klausurrelevant

I

Moderne OpenGL Pipeline

Tesselation



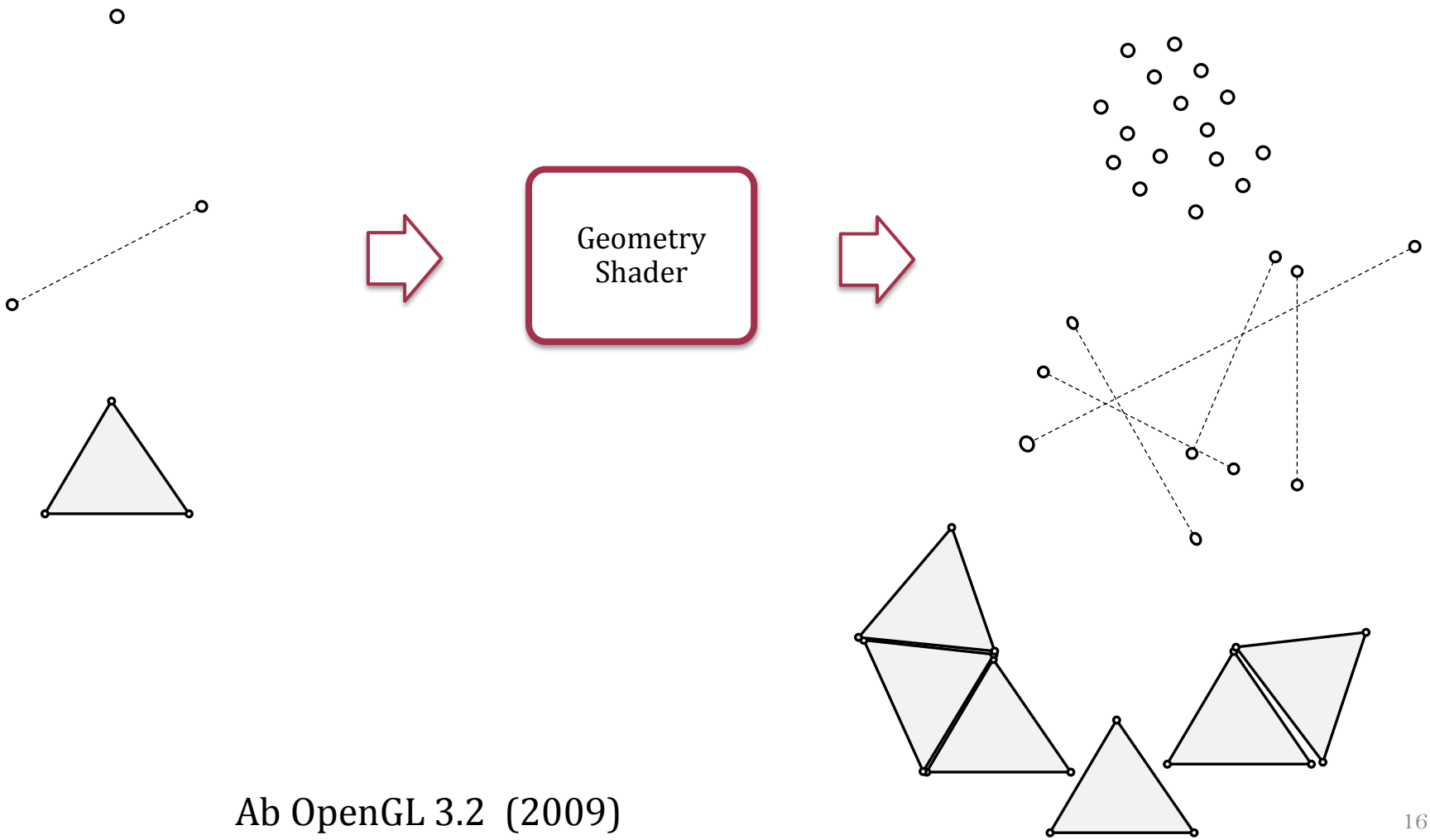
Ab OpenGL 4.0 (2010)

Beispiel


Island11
53.38 FPS, VSync OFF, 1280x720
HARDWARE: NVIDIA GeForce GTX 480
Toggle Full Screen
Toggle REF (F3)
Change Device (F2)
Render Wire(f)rame
Query (P)ipeline Statistics
Dynamic Tessellation LOD: 50
Static Tessellation Factor: 12
Use Dynamic Tessellation (L)OD
Use Frustum Cull in (H)S
Render Refraction (C)austics
Auto Cycle (V)iews (1,2,3,4,5,6)

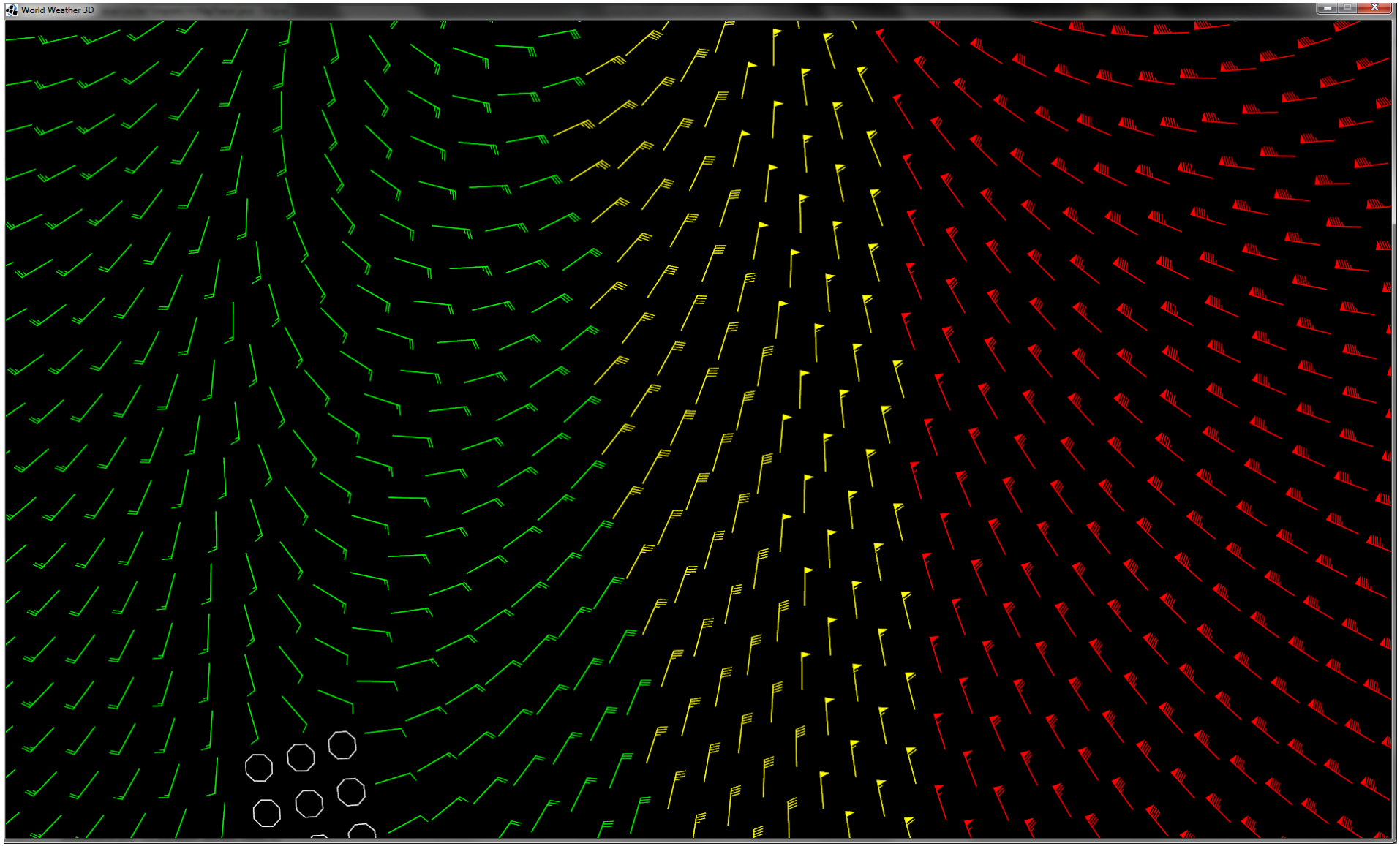
Geometry Shader

%

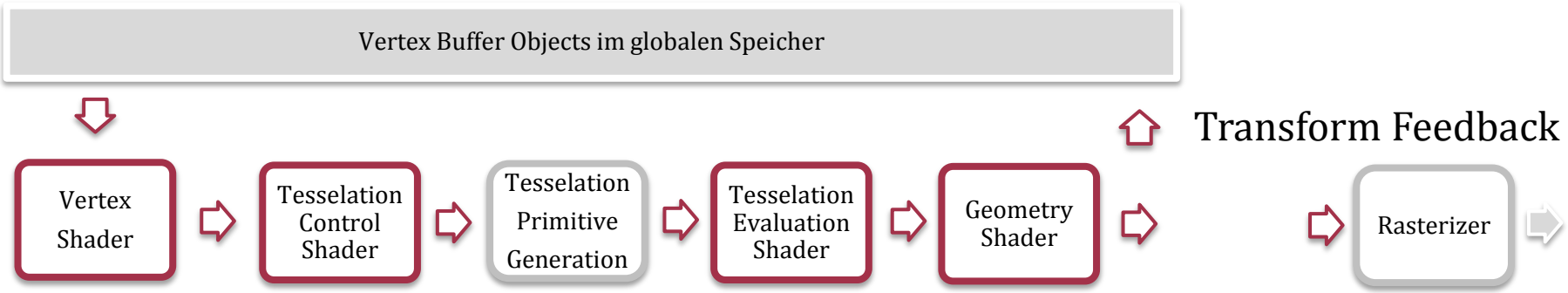


Ab OpenGL 3.2 (2009)

Beispiel



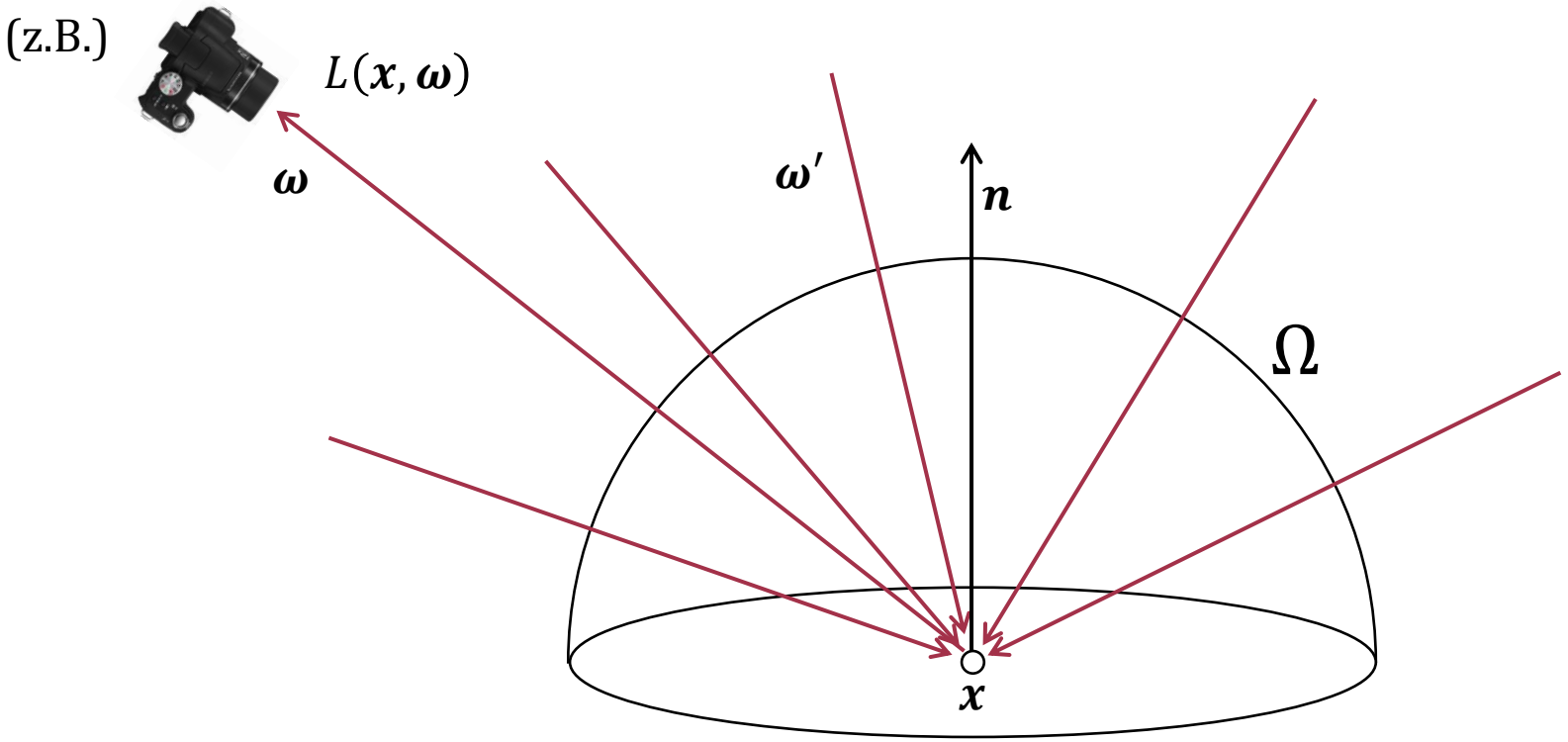
Vereinfachter Überblick



II

Global Illumination

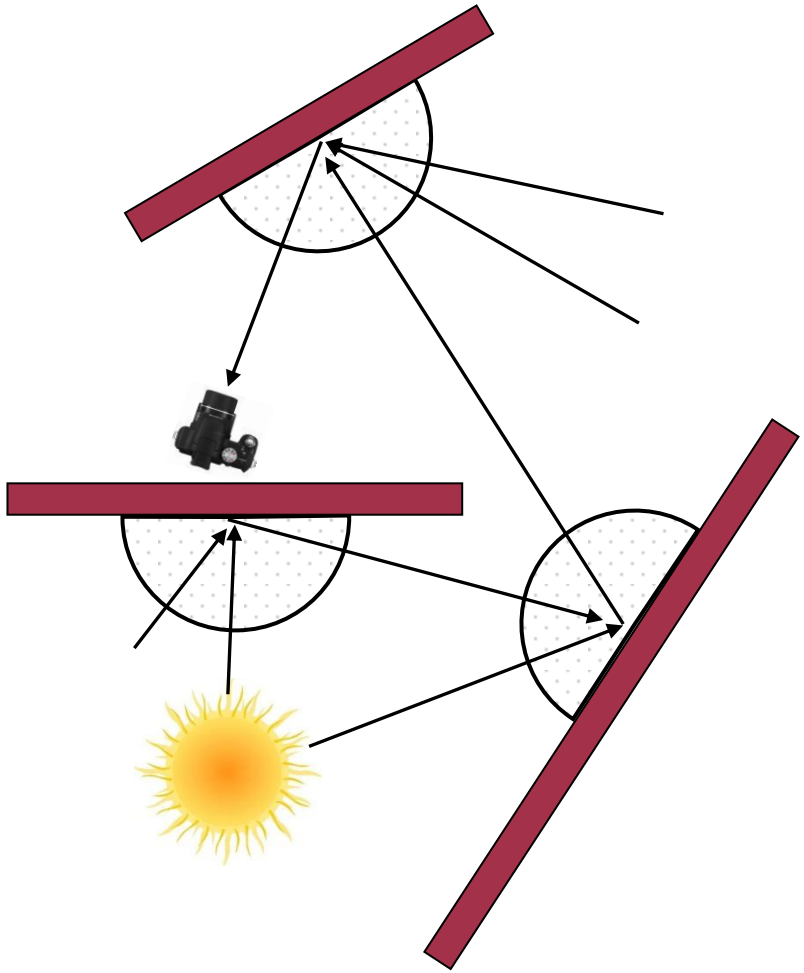
Rendering Equation



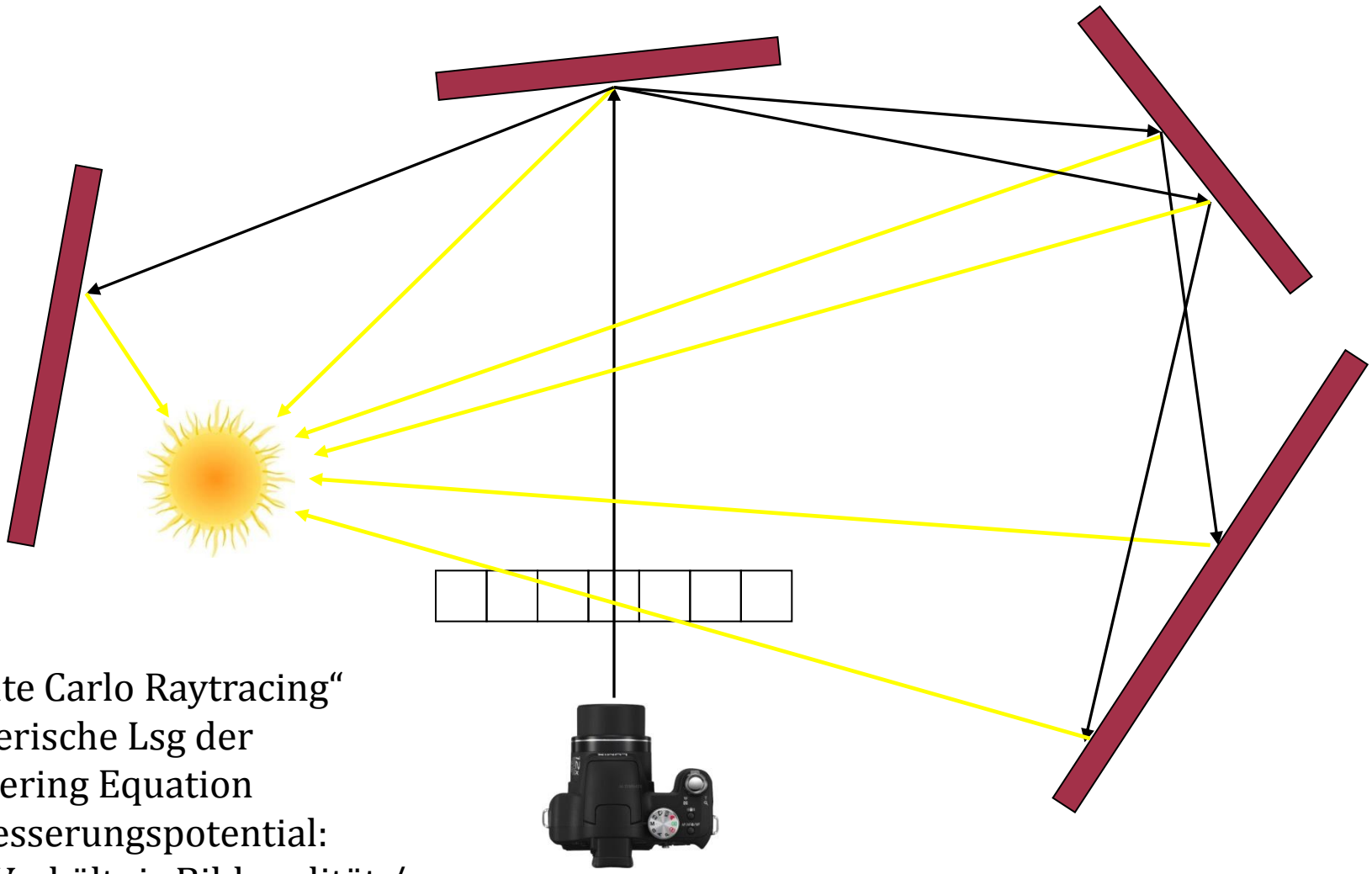
$$L(x, \omega) = L_e(x, \omega) + \int_{\Omega} f_r(x, \omega', \omega) \quad L(x, \omega') \quad (-\omega' n) \quad d\omega'$$

James T. Kajiya: *The rendering equation.* SIGGRAPH 1986

Global Illumination



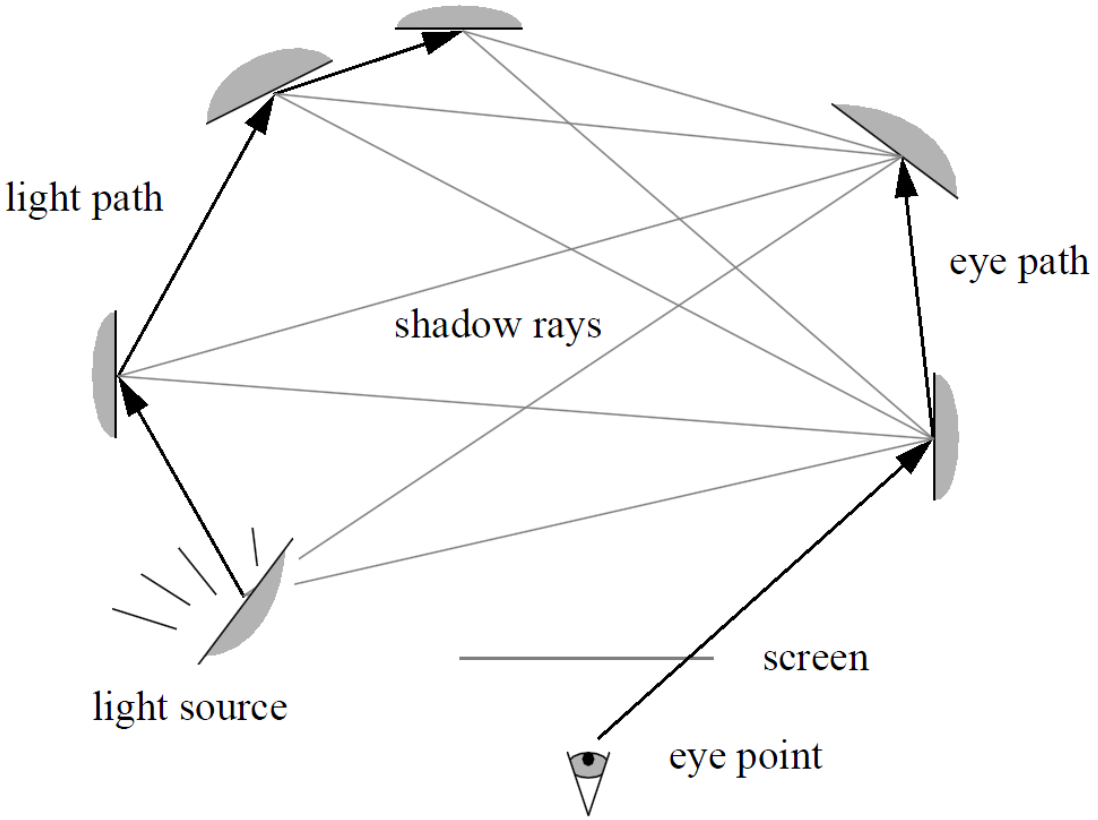
Path Tracing



- “Monte Carlo Raytracing“
- Numerische Lsg der Rendering Equation
- Verbesserungspotential:
  - Verhältnis Bildqualität / Strahlenzahl

James T. Kajiya: *The rendering equation.* SIGGRAPH 1986

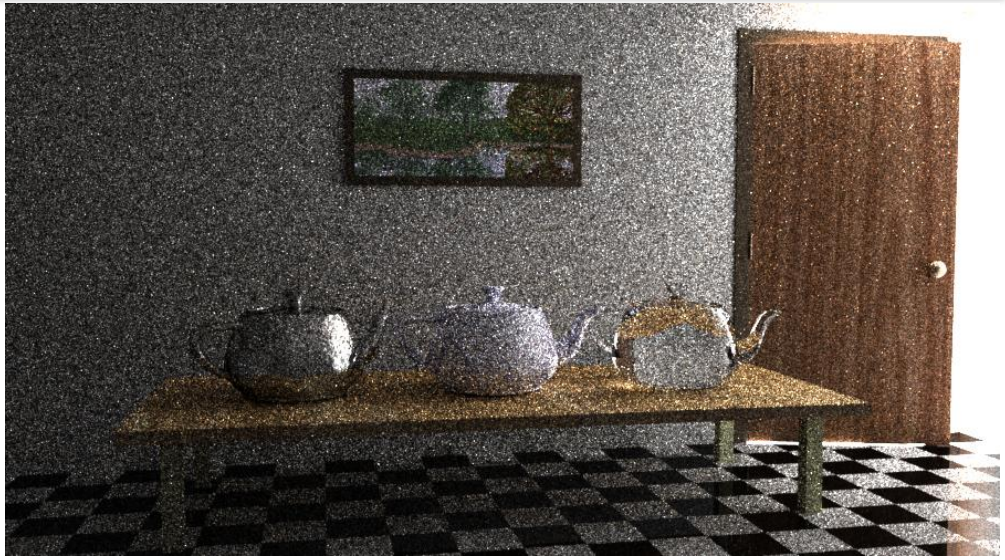
Bi-Directional Path Tracing



- Manche Lichtquelle kaum erreichbar
- Beginne Pfade vom Betrachter (Path Tracing) & von Lichtquellen (Particle Tracing) aus
- Verbinde diese

Lafortune, Willems: *Bi-Directional Path Tracing.* SIGGRAPH 1993

Metropolis Light Transport

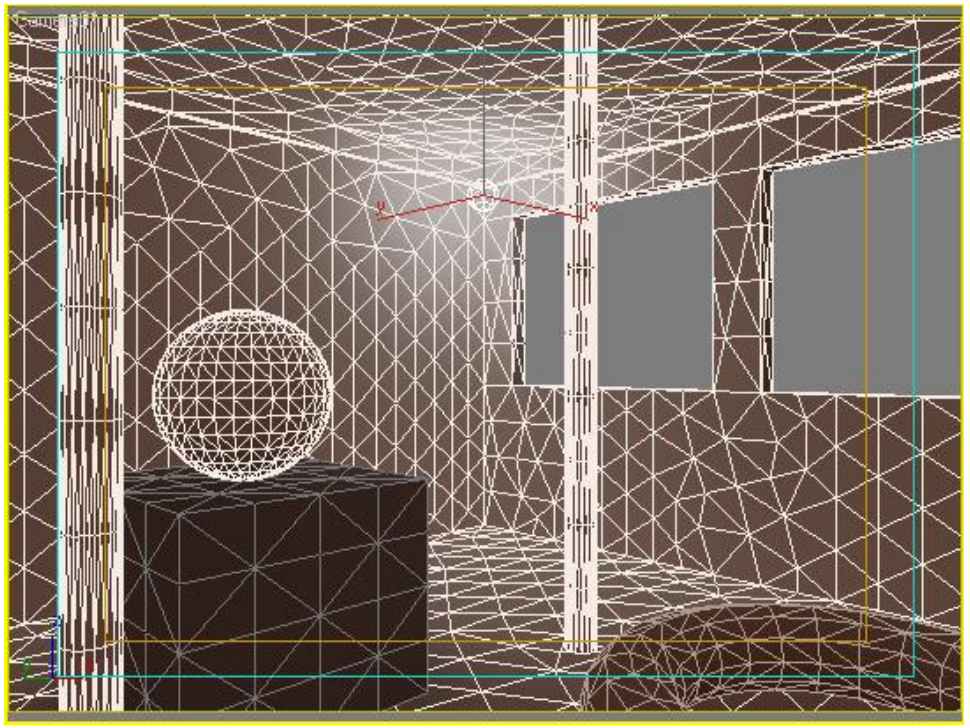


- Speichere Pfade, die viel beitragen
- Bilde Varianten davon

Veach, Guibas: *Metropolis Light Transport.* SIGGRAPH 1997

Radiosity

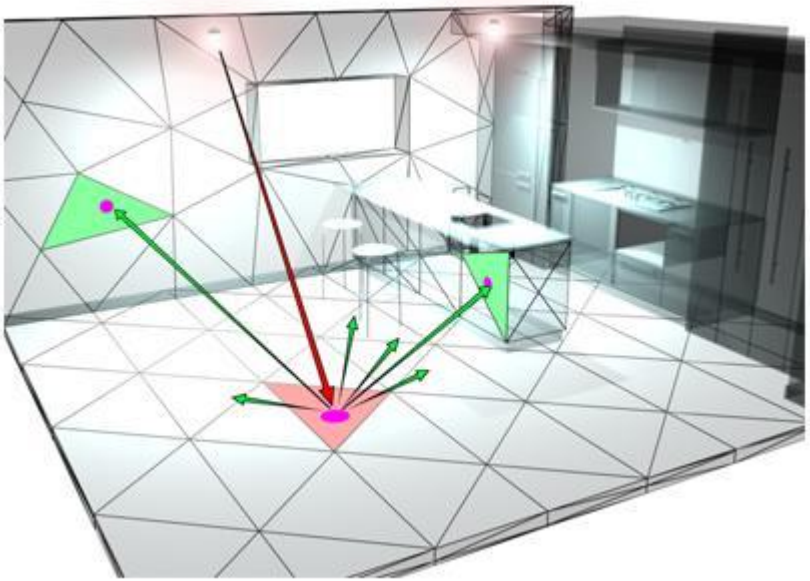
- Zur Ausleuchtung einer Szene unabhängig vom Betrachterstandpunkt
- Getrennt vom Rendering
- Nur Diffuse Komponente der Globalen Beleuchtung
- Berechnet für Patches konstanter Helligkeit
- LGS modelliert Abhängigkeit aller Patches untereinander
  - Analytisch: $O(n^3)$
  - Numerisch: $O(n^2)$



Hanrahan: *A Rapid Hierarchical Radiosity Algorithm*,1993

Radiosity II

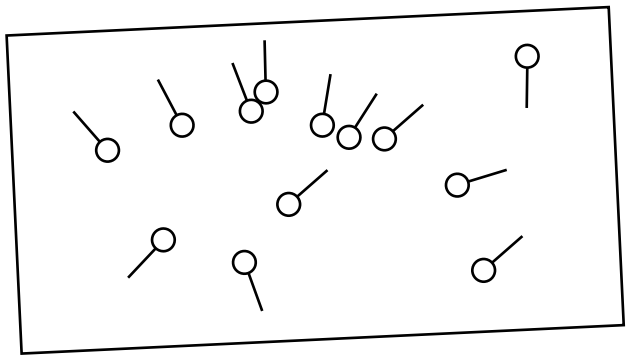
- Progressive Refinement
  - Wähle möglichst helles Senderpatch
  - Versende dessen Radiosity an alle anderen Patches
  - Nimm wieder helles Patch, usw.
- Vorteil:
  - Nur $O(n)$ pro Iteration
  - Kann bei ausreichend guter Näherung abgebrochen werden



Radiosity Anwendung: Lightmaps

Photon Mapping

- Schritt I: Erstellen der Photon Map (statisch)
- Verfolge Photonen ausgehend von Lichtquelle analog zu Path/Particle Tracing durch die Szene
- Speichere Photonen an allen Orten des Auftreffens in KD-Tree
- “Spekulare Photonen“ oft gezielt verschossen & gesondert gespeichert

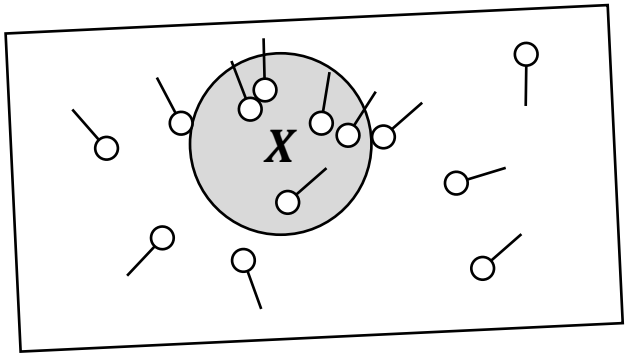


Jensen, ab 1993/1994

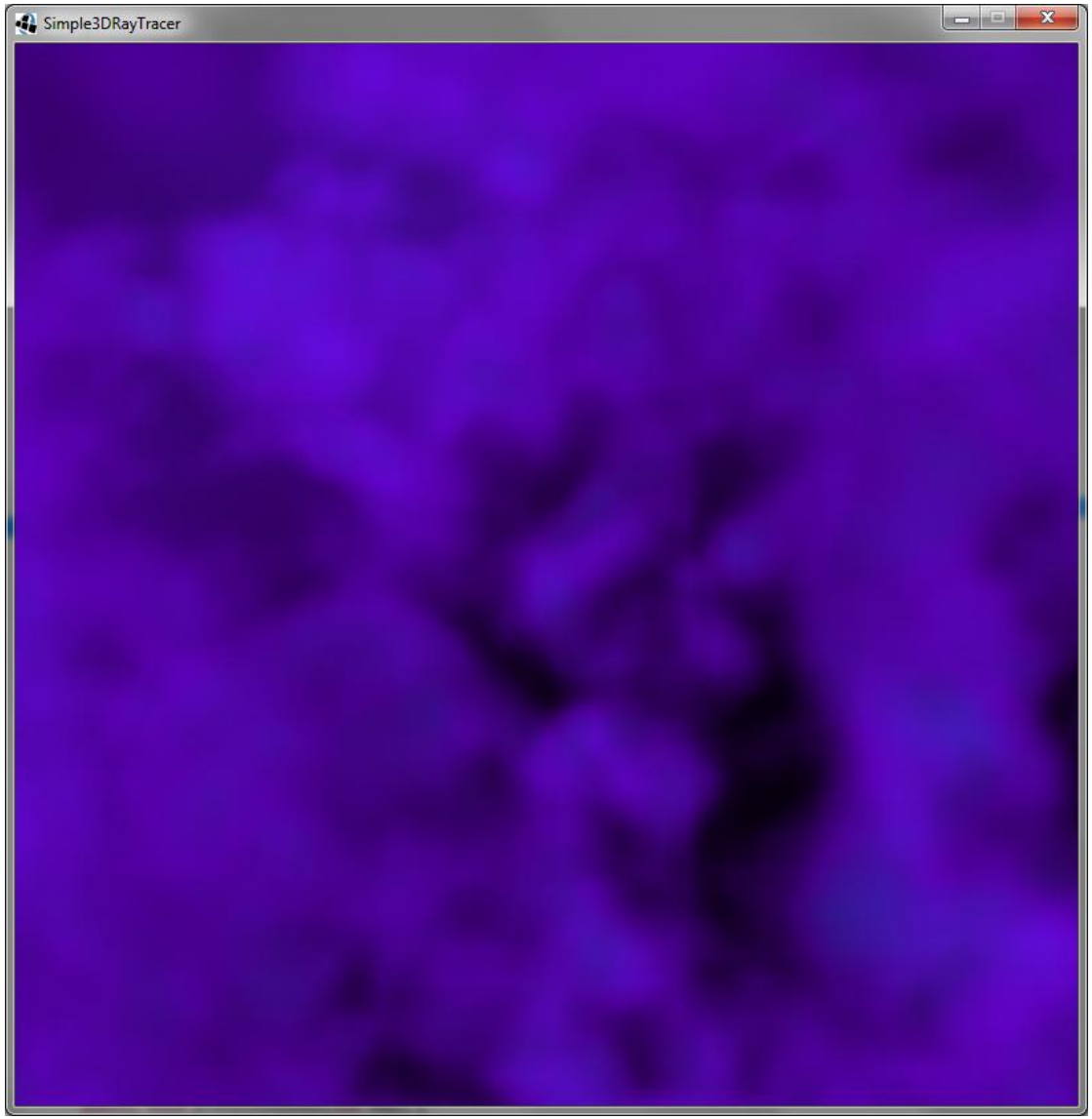
KD-Tree & GPU Implementierungen später

Photon Mapping II

- Schritt II: Rendering
- Für verschiedene Rendertechniken nutzbar
- Beispiel einfaches Raytracing:
  - Berechne Schnittpunkt mit Objekt
  - Suche dann in Photon Map(s) alle Photonen in einer bestimmten Umgebung oder die n Nächsten
  - Berechne damit indirekte Beleuchtung
  - Schneller Ansatz für GI
  - Anders als bei Radiosity auch spekulare Komponente möglich
- Besser: Kombination mit Path Tracing, etc.



Volume Rendering

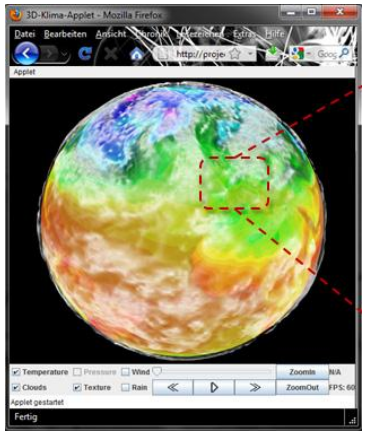


III

Mobile-, Web- & Multiplattform Graphics

Web & Smartphones

Web


3D-Klima-Applet - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
Applet
Temperature
Pressure
Wind
Clouds
Texture
Rain
ZoomIn
ZoomOut
Applet gestartet
Fertig


LIAN LI


TRON


Erinnere mich am 5. Mai an Papas Geburtstag
Hier ist deine Erinnerung für 05.05.2012 um 9 Uhr:
5
Samstag
Mai 2012
Papas Geburtstag
Abbrechen
Bestätigen

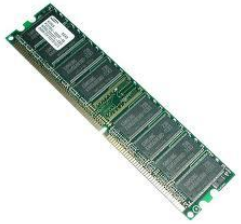
PC

X

X

2.01 (Beta)

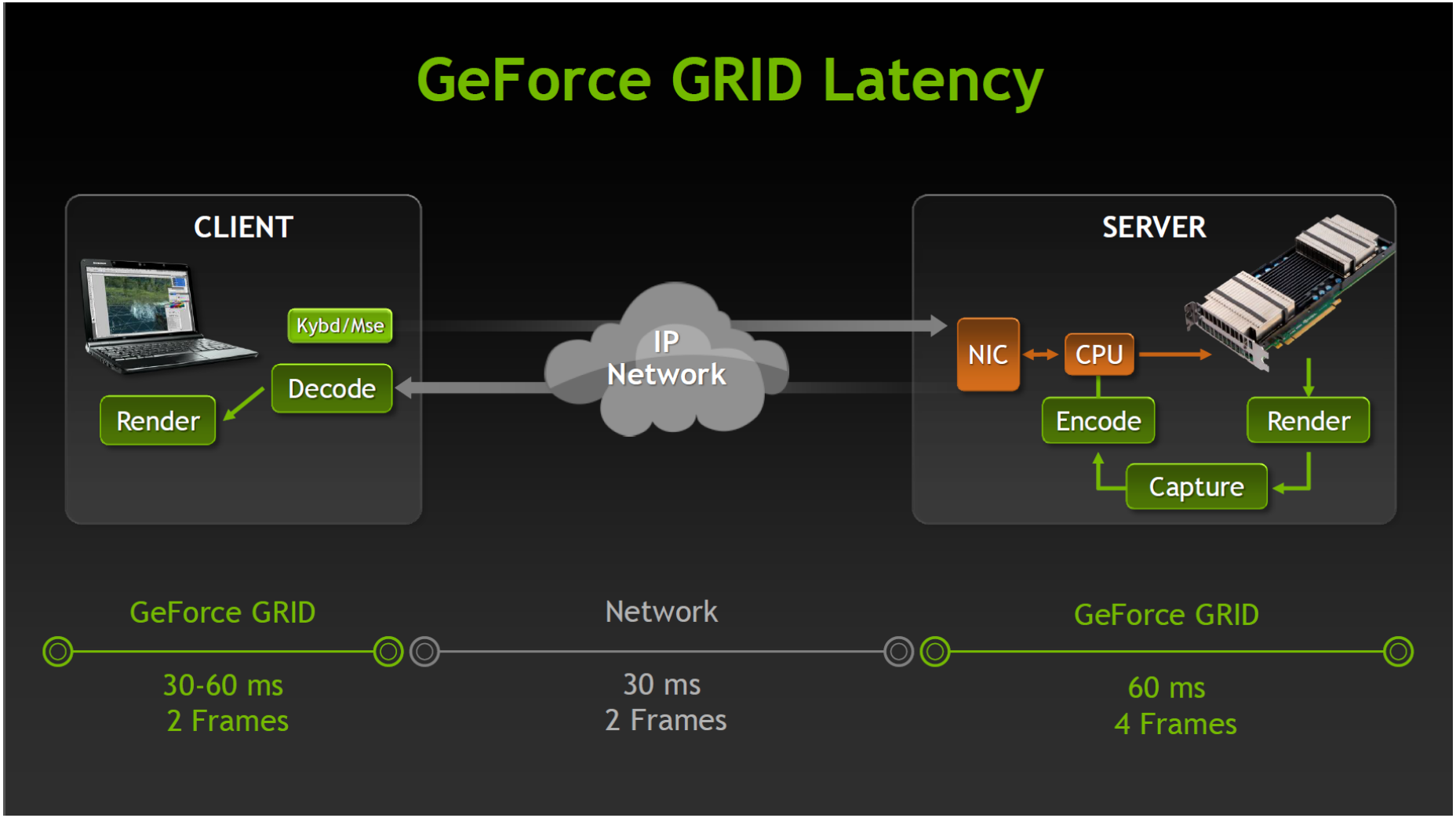
X



Konsolen

Produktzyklus Xbox 360: 10 Jahre

Ausblick: Remote Rendering



Aufbau der Klausur

- Soll Stoff in Breite testen
  - Ca. 17 Aufgaben
  - Math. Grundlagen ca. 20 %
  - Rastergrafik mit OpenGL gut 60 %
  - OpenCL + Partikelsysteme + Raytracing knapp 20 %
- Lösungswege oft recht ähnlich zu bekannten aus Vorlesung, Übung & Probeklausur

Klausurrelevanter Stoff

- Alle Vorlesungs- & Übungsinhalte, **außer**:
- OpenCL /OpenGL Befehle
  - OpenGL Befehle können aber angegeben sein
  - (**Nicht ausgeklammert** sind: GLSL / OpenCL C)
- Baryzentrischen Koordinaten
- Vorlesung 9. 7 .2012 (zweite Hälfte)
- 3D-KD-Tree Raytracing: Nur ganz grob das Prinzip
- Nicos Exkurse in den Übungen
- LWJGL spezifisches (FloatBuffer, …)

Morgen

- Besprechung der Probeklausur
- Evaluation dieser Veranstaltung