Raytracing und Szenengraphen

Abschlussvortrag zur Diplomarbeit von

Björn Schmidt

17. November 2006

J. W. Goethe-Universität Frankfurt am Main

Inhalt

- Ziel der Arbeit
- Grundlagen
- Anforderungsanalyse
- Eigenes Konzept und Implementierung
- Evaluation
- Demo
- Ausblick

Ziel der Arbeit

- Integration eines Echtzeit-Raytracers in eine Open Source Szenengraph-API
- Raytracing statischer und dynamischer Szenen
- Erreichen interaktiver Bildraten

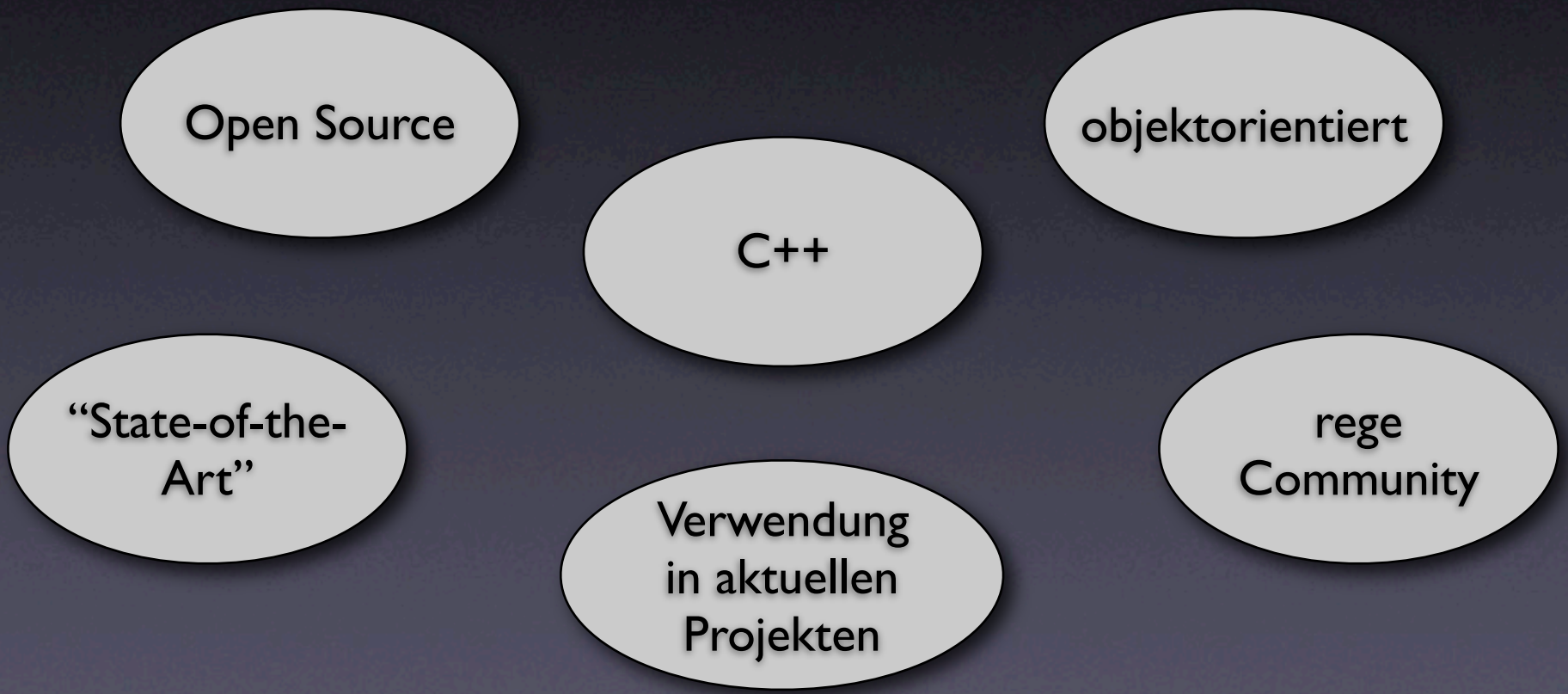
Grundlagen

Szenengraphen

- Beginn der Echtzeit-Computergrafik: Modellierung von Szenen mit Hilfe von Low-Level-Befehlen
- Szenengraphen erlauben einfaches Modellieren und Animieren von komplexen Szenen

Kurzevaluierung existierender Szenengraphen

Vorauswahl



Kurzevaluierung existierender Szenengraphen

OpenSceneGraph

OpenSG

Irrlicht

OGRE 3D

Rasterisierung

- Derzeit dominierendes Verfahren in der Echtzeit-Computergrafik
- Auf Hardwareebene verfügbar (GPU)
- Liefert gute Ergebnisse bei sehr hohen Bildraten
- Aber: Keine physikalisch korrekte Darstellung globaler Beleuchtungseffekte

Raytracing

- Physikalisch korrektes globales Beleuchtungsverfahren
- Darstellung von Spiegelungen und Brechungen in einer 3D-Szene
- Basiert auf der Ausbreitung von Lichtstrahlen

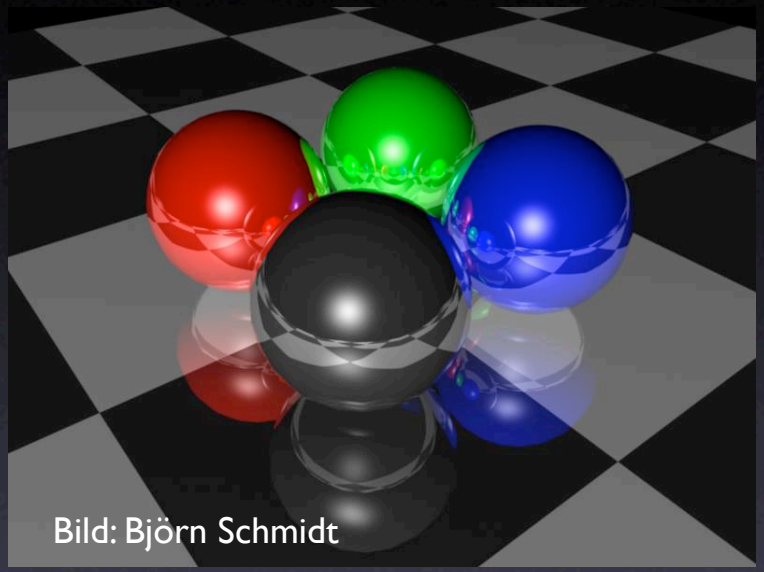


Bild: Björn Schmidt

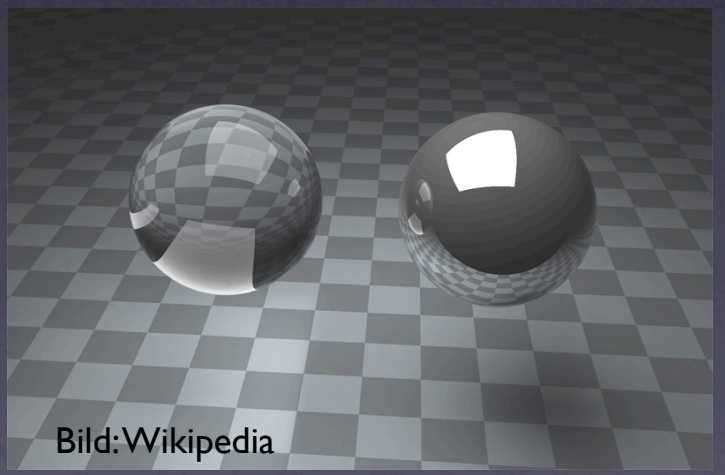


Bild: Wikipedia

Raytracing

- Entsendet durch jeden Pixel der Bildebene einen Strahl in die Szene (Primärstrahlen)
- Berechnet den Schnittpunkt des Strahls mit dem entsprechenden Primitiv der Szene
- Erzeugt Sekundärstrahlen und verfolgt diese rekursiv
- Strahlen der Pixel werden unabhängig voneinander verfolgt: Verfahren ist somit parallelisierbar

Beschleunigungsdatenstrukturen

- Erster Ansatz: Schnitttest mit jedem Primitiv der Szene
- Hüllkörperhierarchie (BVH)
- Reguläres Gitter
- kd-Baum
- Octree

GPU-basiertes Raytracing

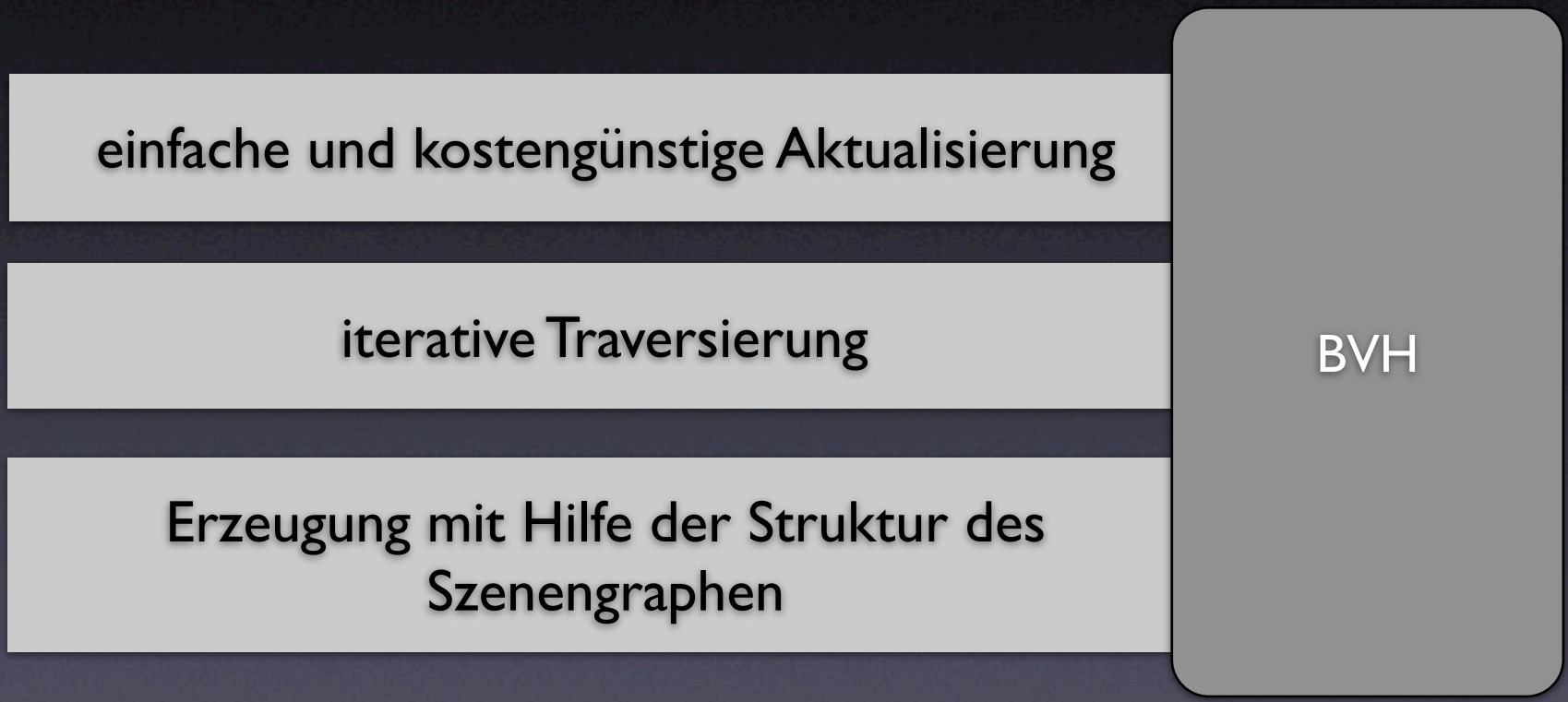
- GPU kann für allgemeine, parallele Berechnungen verwendet werden
- Raytracing kann auf der GPU implementiert werden

Anforderungsanalyse

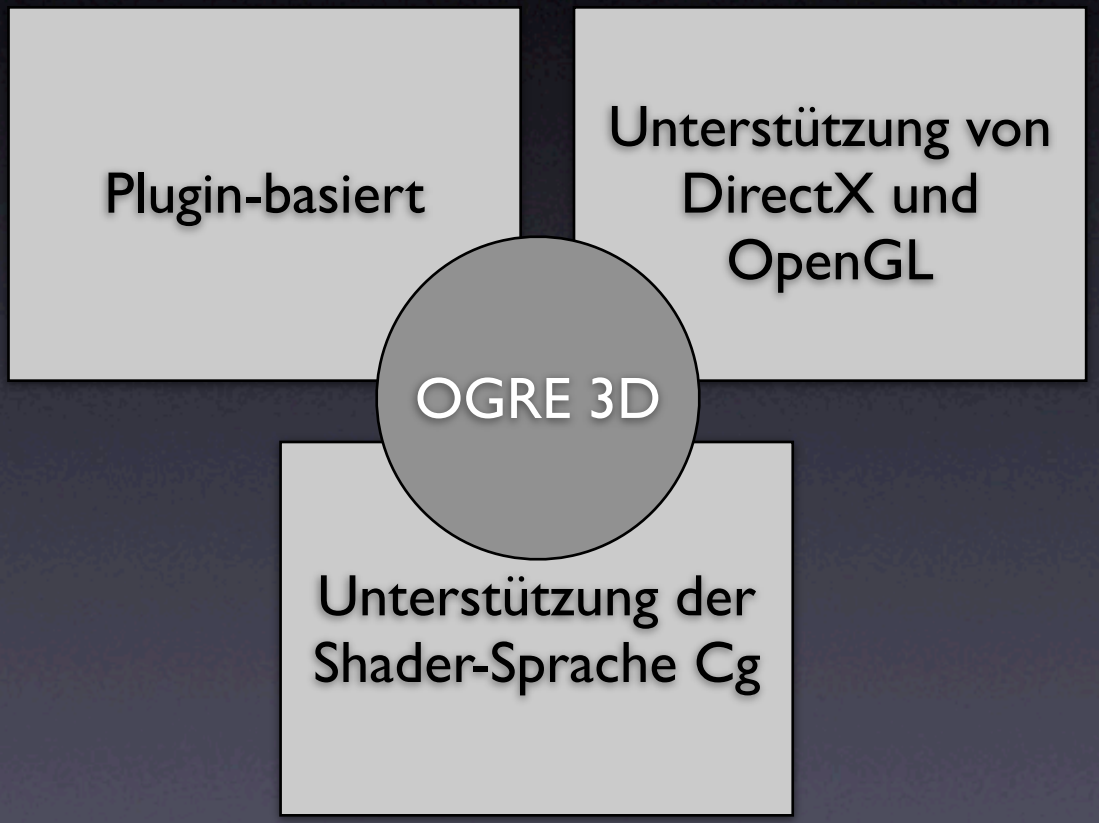
Anforderungen an den Raytracing-Kern

Ausführung auf der GPU	Erreichen hoher Bildraten	Verarbeitung dynamischer Szenen
Darstellung von Reflexionen und Refraktionen	Einsatz auf einem Consumer-PC	Erweiterbarkeit

Anforderungen an die Beschleunigungsdatenstruktur

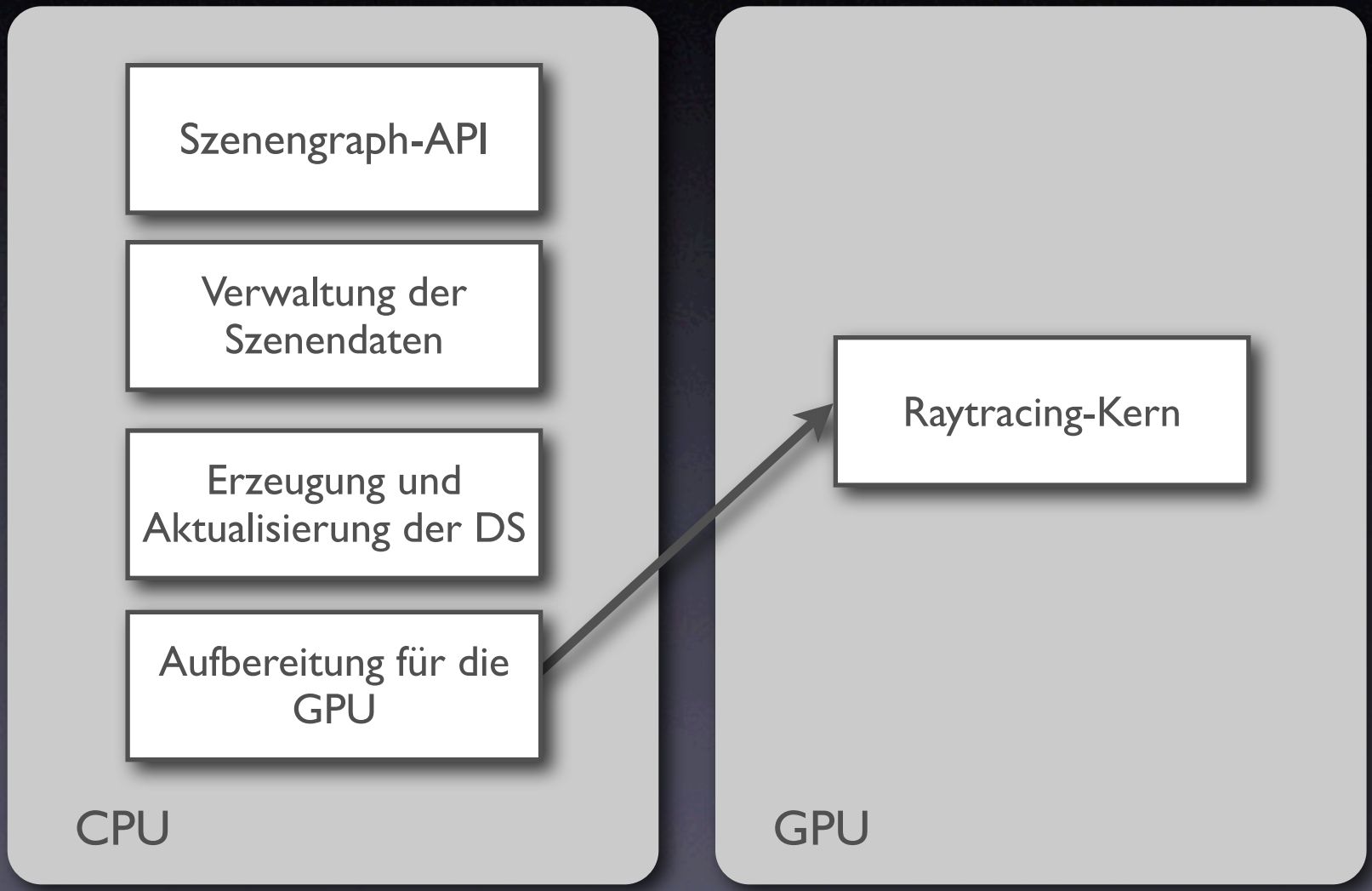


Anforderungen an die Szenengraph-API



Eigenes Konzept und Implementierung

Grundkonzept



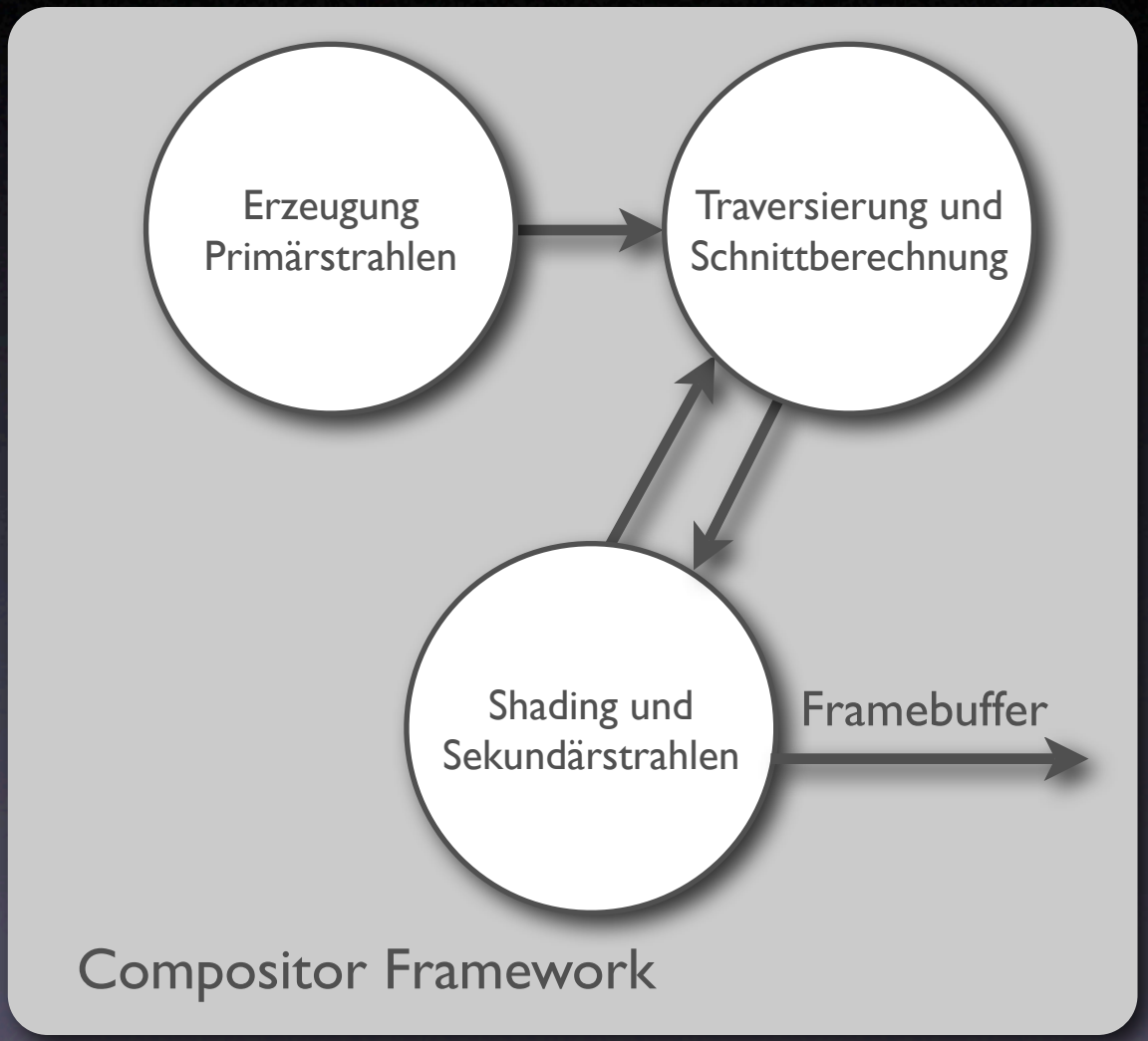
Erzeugen der Beschleunigungsdatenstruktur

- Für jedes animierbare Objekt der Szene wird eine eigene BVH erzeugt
- Diese BVHs werden in einem Baum angeordnet
- Ändert sich die Transformation eines Objekts, so muss nur der entsprechende Zweig des Baums aktualisiert werden

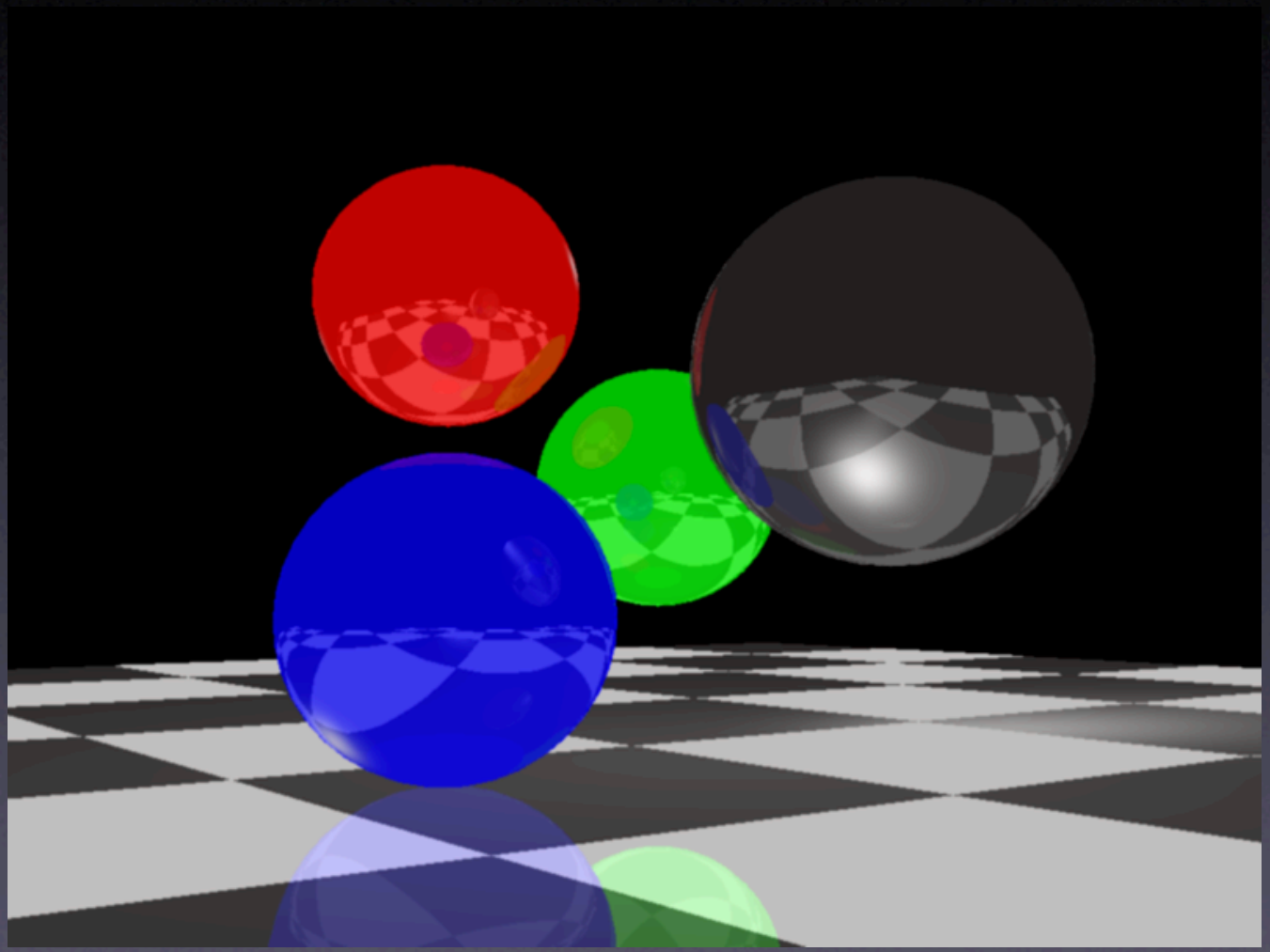
Aktualisierung der Beschleunigungsdatenstruktur

- Translation und Skalierung:
  Ändern die Ausrichtung der AABBs nicht, die BVH kann mit Hilfe der neuen Transformationsmatrix aktualisiert werden
- Rotation:
  Zerstört die Ausrichtung der AABBs, die BVH dieses Zweigs muss neu erstellt werden

Der Raytracing-Kern



Evaluation



Evaluation - Messergebnisse

Testszene	Typ	Polygone	fps
1	statisch	6.540	5,7
2	statisch	33.066	2,8
3	dynamisch	2224	2,6
4	dynamisch	1008	10,2
5	interaktiv	6.540	4,7

Demo

Ausblick

- Grundstein für ein Open Source High-Quality-Realtime-Rendering-System
- Denkbare Erweiterungen:
  - Verwendung von Texturen
  - Material-Framework
  - Anti-Aliasing
  - Erweiterungen des klassischen Raytracing-Verfahrens

Fragen?

weitere Informationen:

http://www.iz-media.de/diplom_bjoern

bs@iz-media.de

Vielen Dank für Ihre Aufmerksamkeit!