Grid Visualization Kernel

Überblick

und

Jüngste Entwicklungen

Visualisierung

- Visualisierung als wichtiges Werkzeug
- Ermöglicht Einsicht in
  - Große Datenmengen
  - Komplexe Daten
- Unterstützt
  - Ergebnisinterpretation
  - Steuerung von Simulationen und Experimenten

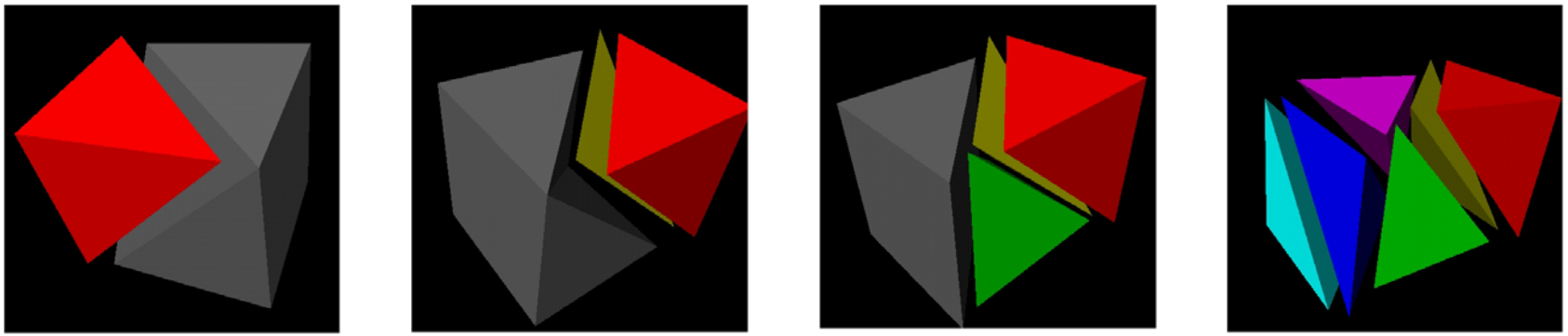
Verteilte Visualisierung

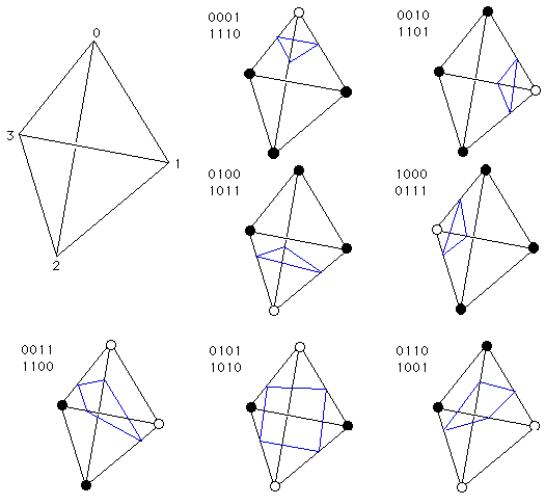
- Visualisierung im HPC Bereich:
  - Verschiedene Visualisierungsverfahren (Volume Rendering, Isoflächen-Extraktion)
  - Große Mengen an Daten
  - Visualisierung selbst wird zur HPC-Applikation
- Lokale Visualisierung Limits:
  - Performance
  - Speicher

High-Performance Computing => High-Performance Rendering

- Ansätze für HPC-basiertes Rendering
  - Parallele Isosurface Extraktion
  - Paralleles Raytracing
  - Data Partitioning & Z-Buffer Merging
- Parallelisierungsansatz:
  - Verteilter Speicher (Grid)
  - Nachrichtenbasierend
  - MPI
- Supercomputer oder MPI über das Grid
- Beispiel:
  Rechner im VR-Lab schneller als Cluster

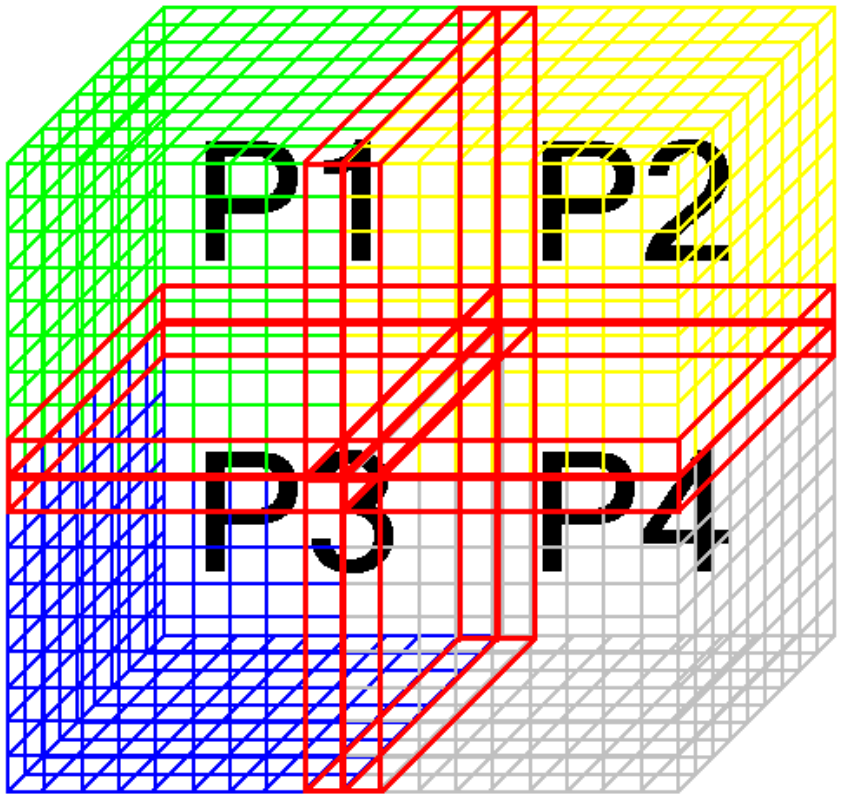
Marching Tetrahedron





Voxelvolumen => Dreiecksnetz

Marching Tetrahedron Parallel



- Aufteilung des Volumens
- Triangulieren der Subvolumina
- Zusammenfügen der Dreiecksnetze

Paralleles Raytracing

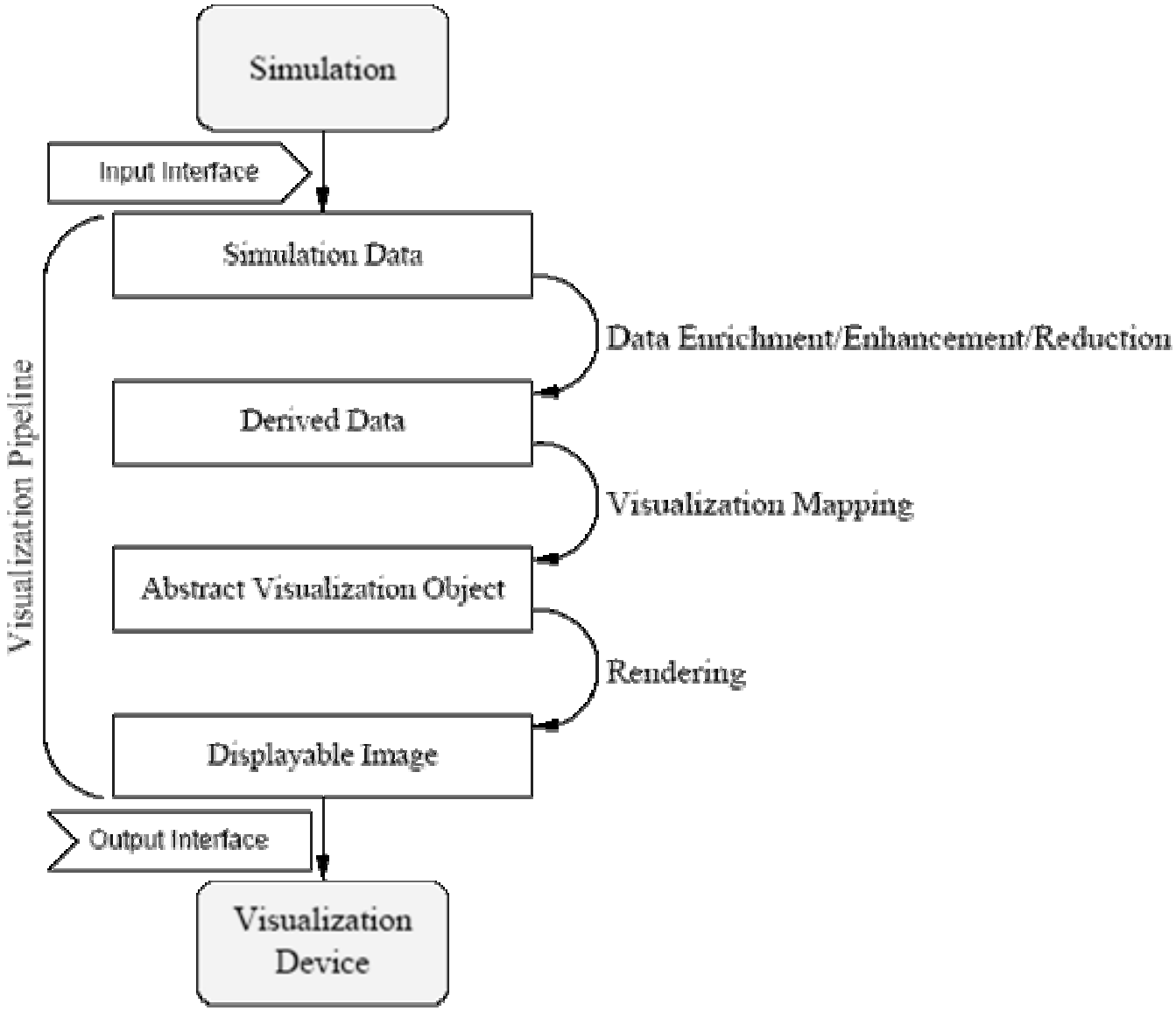
- Parallelisierung auf Pixel-Ebene
- Zuerst Verteilung der Szenendaten
- Rendering von Pixelgruppen pro Prozess
- Zusammenfügen des Bildes

Data Partitioning
Z-Buffer Merging

- Datenaufteilung
- Separates Rendering
- Z-Buffer Merging

  => siehe gestrige Präsentation

Visualisierungs-Pipeline



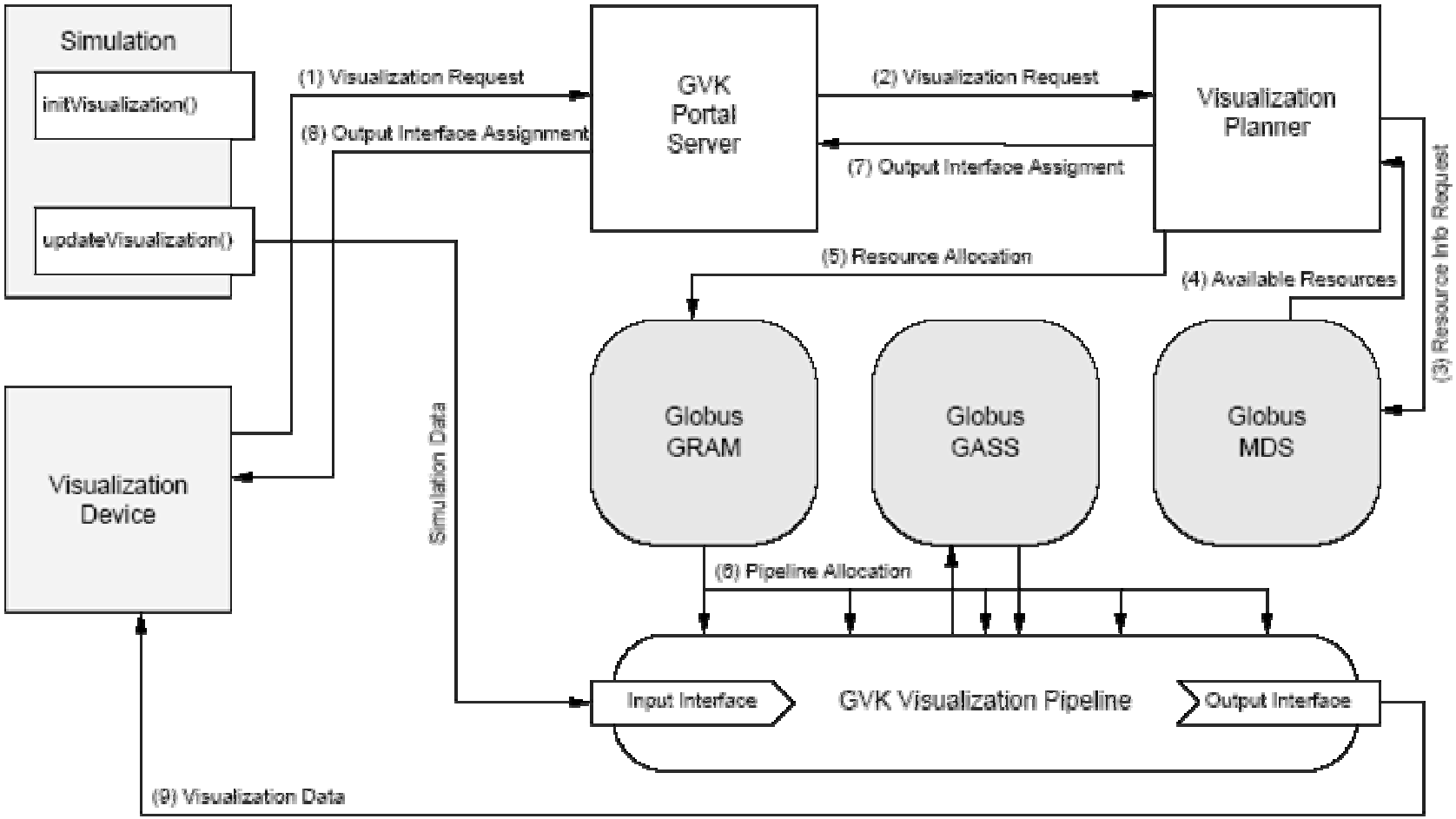
Verteilte Visualisierung

- Pipeline wird über mehrere Rechner verteilt
- Oft langsamer als lokale Visualisierung
- Aber:
  Viele Szenarien sind wegen Datenmenge nicht anders möglich
- Beispiel:
  Rendering Supercomputer => Darstellung Desktop, z.B. Videocodierung

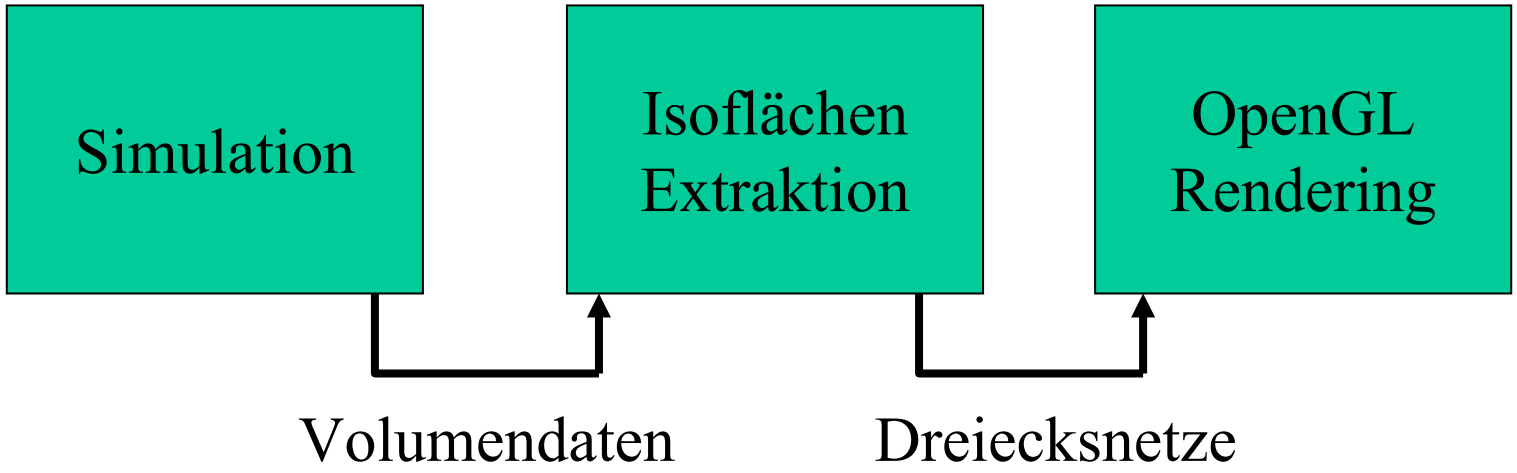
GVK Konzept

- „Gridifikation“ der Visualisierungspipeline
- Daher:
  - GVK ist über das Grid verteilt
  - GVK baut die Visualisierungspipeline auf
  - GVK startet Module die durch Datenströme verbunden werden

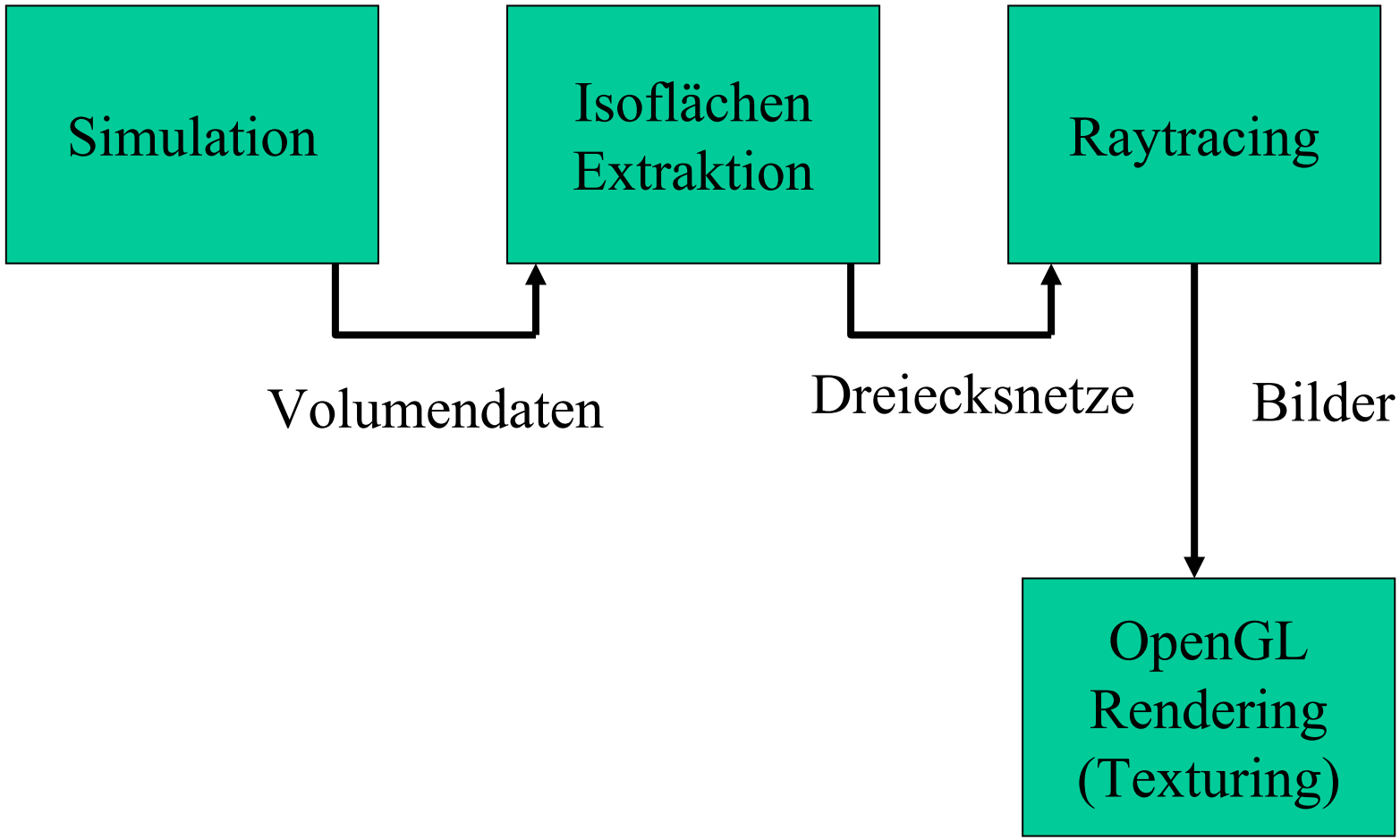
Konzeptionelles Diagramm



Beispiel-Pipeline



Beispiel-Pipeline



Der Visualisierungsprozess

- Der Benutzer will visualisieren
- Er/Sie liefert
  - Zu visualisierende Daten(quelle)
  - Quell-Datenformat
  - Visualisierungs-Zielort
  - Visualisierungstyp
- Visualisierungs-Pipeline Aufbau: Visualisierungsplanung

Die Visualisierungsplanung

- Erforderliche Ausgabe des Planungsprozesses:
  - Pipeline Struktur
    - Benötigte Module
    - Verbindungstypen
  - Visualisierungsalgorithmen
  - Ausführungsorte der Module

  => Statisches Scheduling

Die Visualisierungsplanung

- Schritte im Planungsprozess:
  1. Aufgabenzerlegung
  2. Sammeln von Ressource Informationen
  3. Auswahl des Rendering Algorithmus
  4. Abbildung auf die Ressourcen

Aufgabenzerlegung

- Transformationen im Visualisierungsprozess
- Eingabe:
  Visualisierungsanforderung
- Ausgabe:
  Geordnete Menge an Modulen
  Definiert durch Schnittstellen
- Beschränkungen:
  - Verfügbare Software Module
  - Modulschnittstellen
- Liefert mehrere Zerlegungen, wenn möglich

Sammeln von Ressource Informationen

- Gesammelte Ressource Informationen:
  - Verfügbare Rechner
  - Struktur
  - Prozessoranzahl
  - Speicher
  - Massenspeicher
  - Netzwerk
- Quellen:
  - Globus MDS
  - Messungen (Netzwerk, CPU)

Auswahl des Rendering Algorithmus

- Basierend auf
  - Aufgabenzerlegung
  - Softwaremodule
- Berücksichtigt
  - Verfügbare Hardware
    - Single / Multiprozessor
    - Gemeinsamen / Verteilten Speicher
- Auswahl einer geeigneten Implementierung

Abbildung auf die Ressourcen

- Bildet die Rendering Module auf die Ressourcen ab
- Algorithmusauswahl und Ressourceabbildung sind iterativ
- Leistungsschätzung wird berechnet

Abbildung auf Ressourcen

	Sequentielle Algorithmen	Parallele Alg. Gemeinsamer Speicher	Parallele Alg. Verteilter Speicher
Einzelprozessor System	+ -	-	- -
Multiprozessor Gemeinsamer Speicher	+ -	+ +	+ +
Multiprozessor Verteilter Speicher	+ -	-	+ +

Abbildung auf Resourcen

- Leistungsschätzung / Pipelinestufe

$$PT = \frac{IS}{BW} + \frac{PS}{NP * PP * AS * MF} * \frac{PS}{MS}$$

PT.....Verarbeitungszeit
IS.....Eingabedatengröße
BW...Netzwerkbandbreite
PS.....Problemgröße
NP....Prozessoranzahl
PP.....Prozessorleistung
AS....Algorithmusskalierbarkeit
MF...Abbildungsfaktor
PS.....Problemgröße
MS....Speichergröße

Abbildung auf Resourcen

- Kriterium:
  - Durchsatz der gesamten Pipeline
  - Daher:
    Langsamste Stufe entscheidet

Pipeline Konstruktion

- Module werden von Globus GRAM gestartet
- RSLs (Resource Specification Language) werden vom Visualisierungsplaner generiert
- Startreihenfolge ist wichtig
- Datenaustausch Globus IO

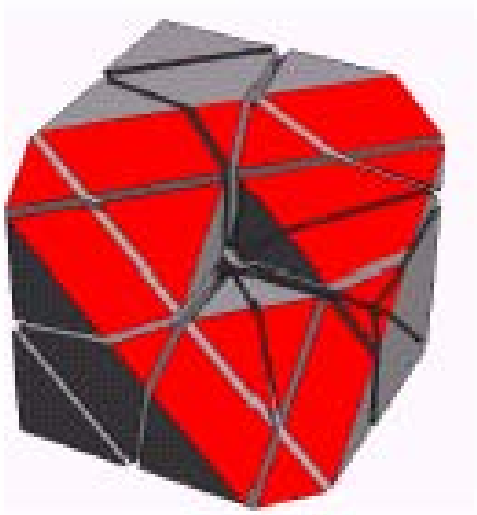
Kommunikation zwischen Modulen

- Gewählter Ansatz:
  Globus IO
- Low-Level Schnittstelle
- Ähnlich BSD Sockets
- Überwachte Verbindungen
- Datentyp-Abhängige Transmitter
  - (JPEG) Bilddaten
  - (Progressive) Dreiecksnetze
  - Volumen

Dynamische Adaptierungen

- Leistungs- und Bandbreitenüberwachung
  - MDS, NWS, spezielle Lösungen, ...
- Netzwerkübertragung
    - Datenkompression
    - Progressive Dreiecksnetze
    - JPEG-Kompression
  - Dreiecksnetzübertragung
    - Occlusion Culling
    - Level of Detail
  - Bildübertragung
    - Image-based Rendering
    - Bildkompression
    - Videokodierung

Dreiecksnetze: Level of Detail

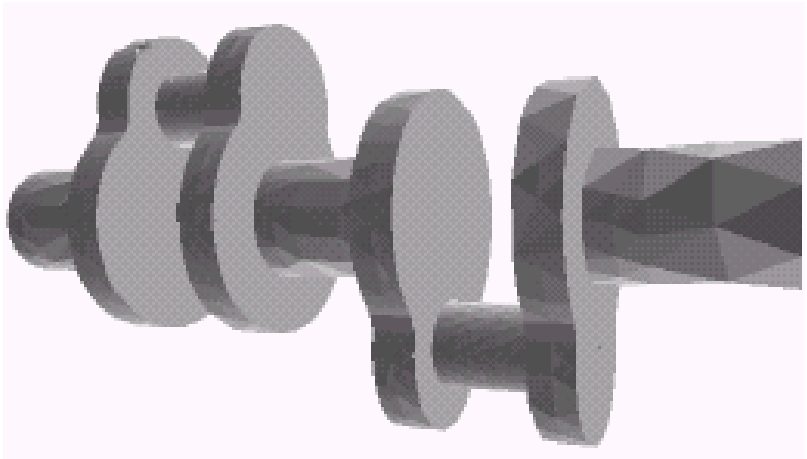


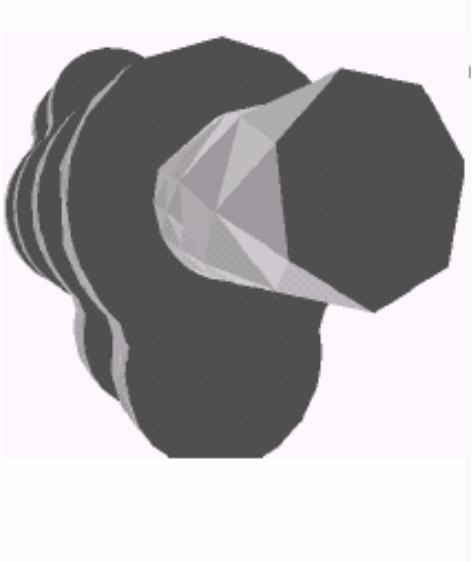
33 tetrahedrons

264 tetrahedrons

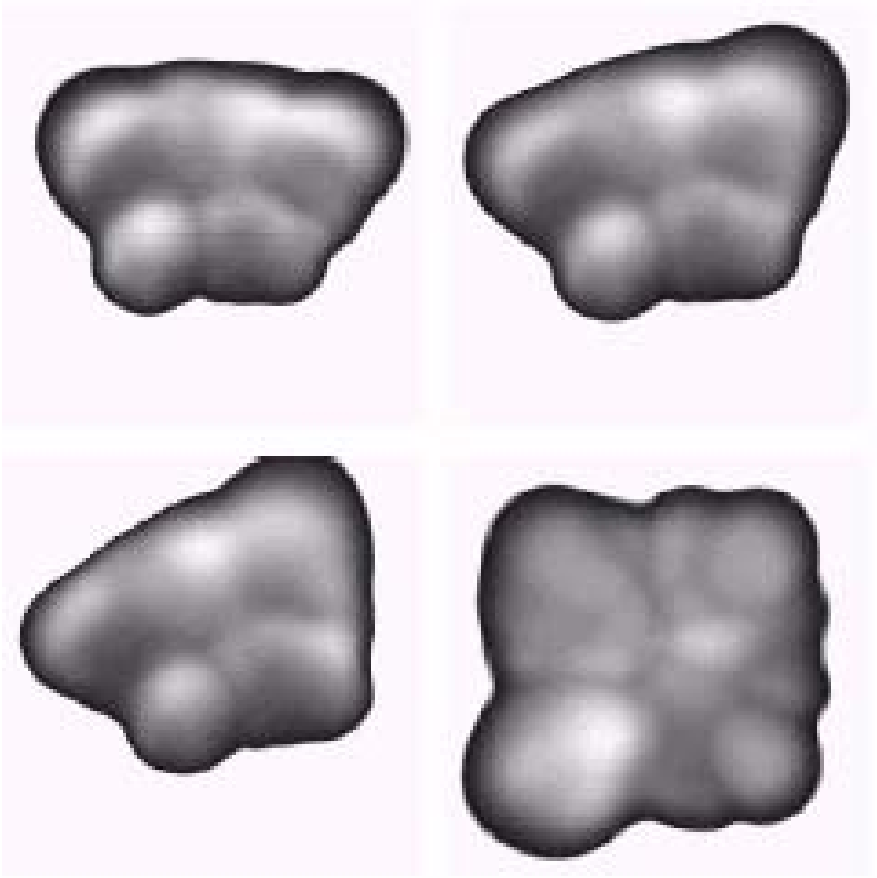
36452 tetrahedrons

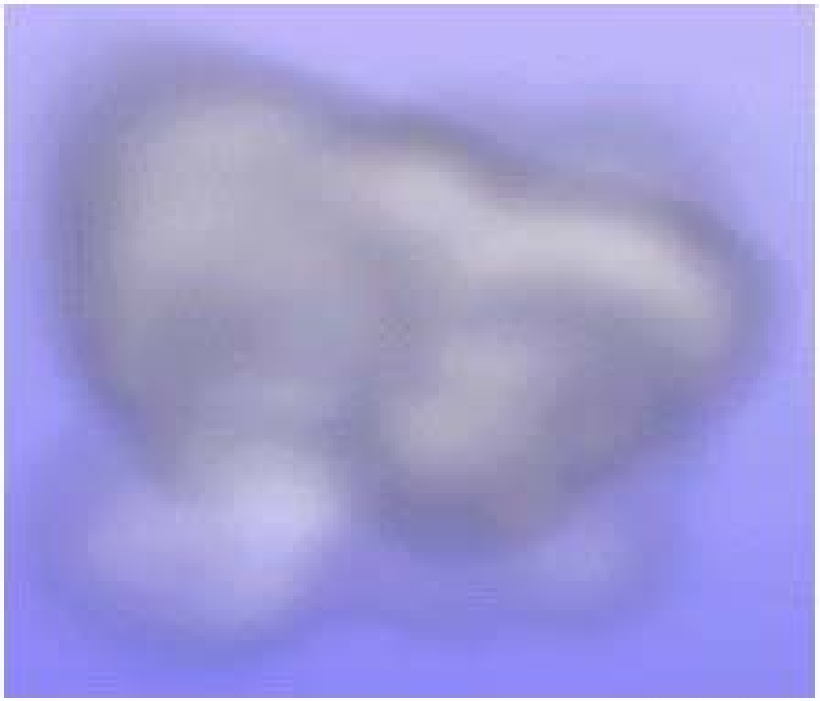
Dreiecksnetze: Occlusion Culling





Bilder:
Image Based Rendering





Dynamische Adaptierungen

- „Migration“ von Modulen
  - Zwischen Simulationsschritten
  - Umkonfiguration der Pipeline
  - Geeignetere Resourcen verfügbar