Workshop on Parallel Visualization and Graphics

Realtime Realism: Interactive Ray-Tracing and Lighting Simulation

Philipp Slusallek
Computer Graphics Lab
Saarland University
inTrace GmbH

Saarland University

- Location:
  - Heart of Europe, right on the border to France
  - Close to Frankfurt, Paris, Luxemburg, Brussels, …

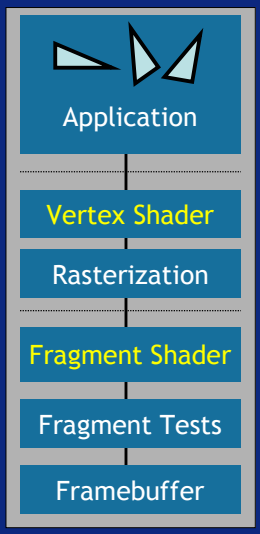
Saarland University

Overview: Realtime Ray Tracing

- What is Wrong With Current 3D Graphics?
- Why Realtime Ray Tracing?
- What is Possible Today?
- Where Are We Going?

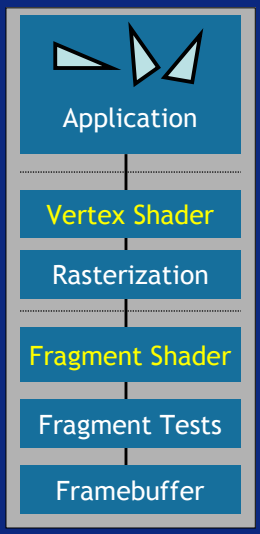
What is Wrong?

- Today: Rasterization
  - Highly successful VLSI technology
  - From graphics supercomputers to an add-on in a PC chip-set
- Advantages
  - Simple and proven algorithm
  - Can generate very nice images
  - Getting faster quickly
  - Trend towards full programmability



What is Wrong?

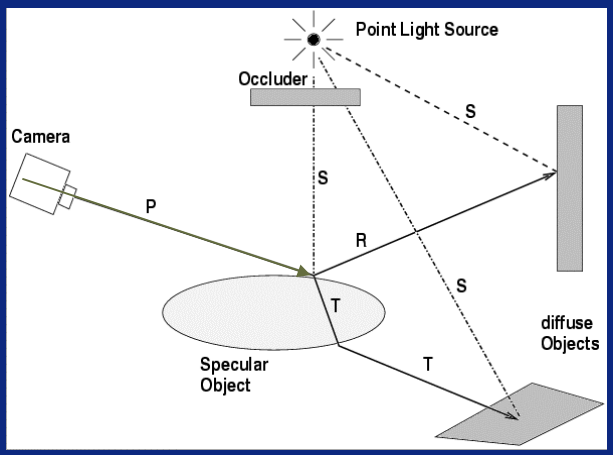
- Technological Limitations
  - Single triangle at a time
  - Fast but fixed, regular sampling
  - Many approximations
- Technological Monopoly
  - All of todays interactive 3D graphics is based on the same algorithm

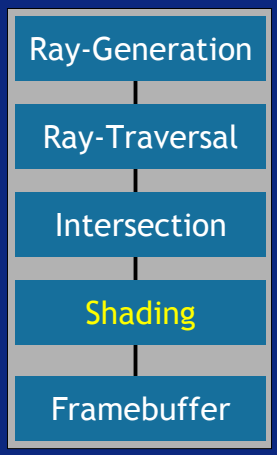


What is Wrong?

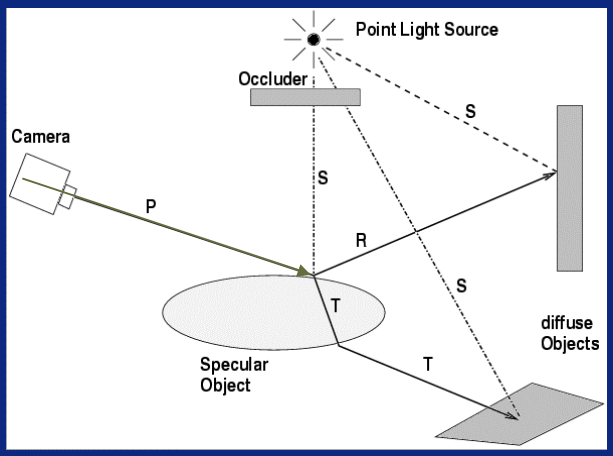
- We Have Come To Accept Its Inherent Problems
  - Image Quality
  - Interactivity
  - Scalability
  - Content Creation
  - Programmability
  - Reliability
  - …

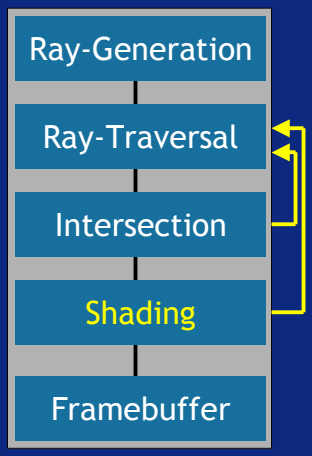
What is Realtime Ray Tracing?





What is Realtime Ray Tracing?





Why Ray Tracing?

- Models Physics of Global Light Transport
  - Right framework for advanced rendering effects
- Simple and Fully Automatic Algorithm
  - Plug&Play of geometry and shaders
- More Accurate
  - No approximations or multi-pass rendering
- More Efficient
  - Output sensitivity with build-in occlusion culling
- More Scalable
  - Logarithmic (!!) in scene complexity
  - Linear in number of pixels, rays, and processors

Why Ray Tracing?

- Models Physics of Global Light Transport
  - Right framework for advanced rendering effects
- Simple and Fully Automatic Algorithm
  - Plug&Play of geometry and shaders
- More Accurate
  - No approximations or multi-pass rendering
- More Efficient
  - Output sensitivity with build-in occlusion culling
- More Scalable
  - Logarithmic (!!) in scene complexity
  - Linear in number of pixels, rays, and processors

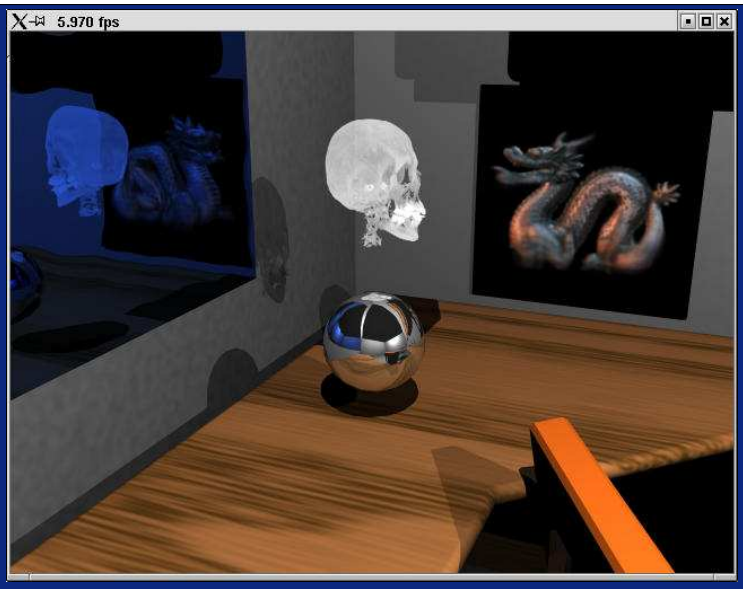
Accurate Global Light Transport

Accurate Global Light Transport

Why Ray Tracing?

- Models Physics of Global Light Transport
  - Right framework for advanced rendering effects
- Simple and Fully Automatic Algorithm
  - Plug&Play of geometry and shaders
- More Accurate
  - No approximations or multi-pass rendering
- More Efficient
  - Output sensitivity with build-in occlusion culling
- More Scalable
  - Logarithmic (!!) in scene complexity
  - Linear in number of pixels, rays, and processors

Plug&Play Shading



Why Ray Tracing?

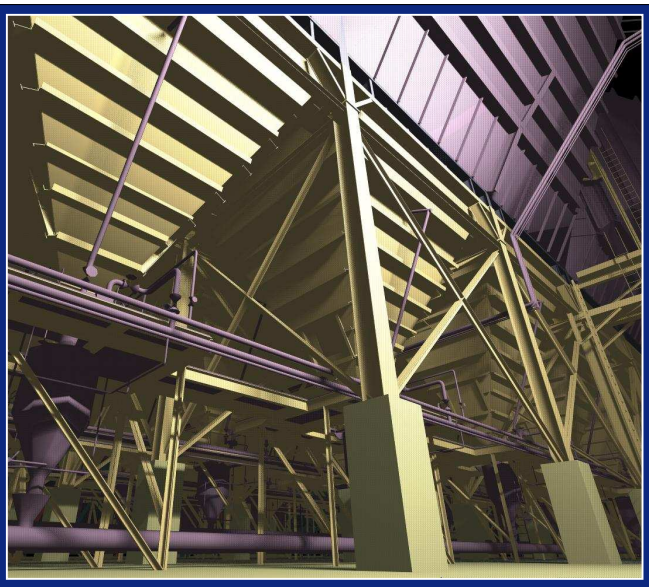
- Models Physics of Global Light Transport
  - Right framework for advanced rendering effects
- Simple and Fully Automatic Algorithm
  - Plug&Play of geometry and shaders
- More Accurate
  - No approximations or multi-pass rendering
- More Efficient
  - Output sensitivity with build-in occlusion culling
- More Scalable
  - Logarithmic (!!) in scene complexity
  - Linear in number of pixels, rays, and processors

More Accurate

Why Ray Tracing?

- Models Physics of Global Light Transport
  - Right framework for advanced rendering effects
- Simple and Fully Automatic Algorithm
  - Plug&Play of geometry and shaders
- More Accurate
  - No approximations or multi-pass rendering
- More Efficient
  - Output sensitivity with build-in occlusion culling
- More Scalable
  - Logarithmic (!!) in scene complexity
  - Linear in number of pixels, rays, and processors

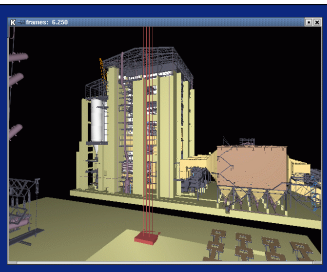
More Efficient

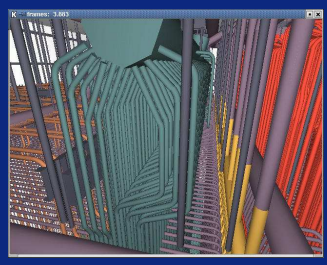


Why Ray Tracing?

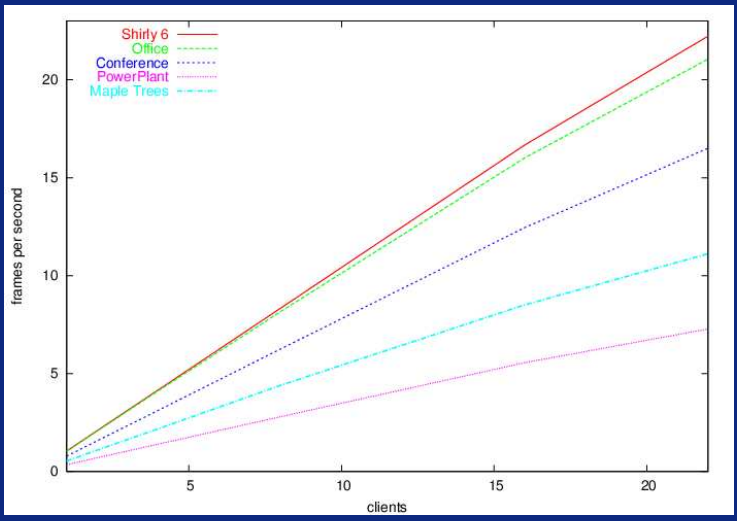
- Models Physics of Global Light Transport
  - Right framework for advanced rendering effects
- Simple and Fully Automatic Algorithm
  - Plug&Play of geometry and shaders
- More Accurate
  - Does not rely on approximations and multipass
- More Efficient
  - Output sensitivity with build-in occlusion culling
- More Scalable
  - Logarithmic (!!) in scene complexity
  - Linear in number of pixels, rays, and processors

More Scalable





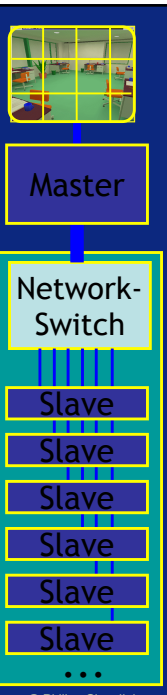
More Scalable



Scalability graph on dual processor AMD 1800+ clients (up to 48 CPUs)

What is Possible?

- Hardware Configuration
  - Setup with commodity HW
    - Dual Athlon/Pentium-4 PCs
    - Fast- and Gigabit Ethernet
  - Master:
    - Application and OpenRT library
    - Job distribution and load-balancing
  - Slaves:
    - Ray tracing computation only



What is Possible?

VIS 2003

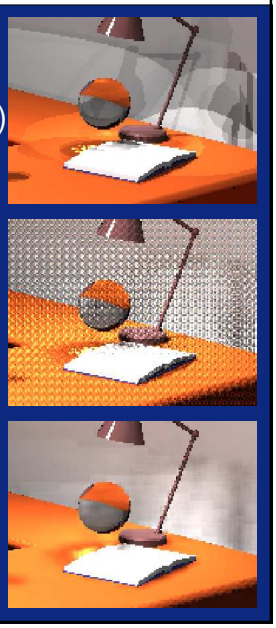
- Distributed Ray Tracing on Commodity PCs
  - Our mantra: Efficiently expose and use coherence
    - Demand-driven & output-sensitive computation
  - Highly optimized algorithms and implementation
    - Partial breadth first processing (packet tracing)
    - Cache coherence & data layout
    - SIMD processing using SSE
    - Up to 7 Mrays per second and CPU
  - Efficient parallelization and distribution
    - Multi-threading per node
    - Images-based dynamic load balancing
    - Asynchronous & pipelined communication
    - Linear scalability for most workloads

What is Possible?

- Hardware Support for Ray Tracing
  - High-performance CPUs today
    - Future: many FPUs in multi-core designs
  - Ray tracing on GPUs
    - Completely on GPU [Purcell'02]
    - Scalable ray tracing on GPUs
      - Fully pipelined processing on CPU & GPU
  - SaarCOR hardware architecture
    - Low external memory bandwidth
    - High HW utilization
      - Same performance at half the FP resources
    - Coherence allows efficient use of virtual memory
    - Scalable HW design

What is Possible?

- Realtime Lighting Simulation
  - Generation of Virtual Point Lights (VPLs) through particle tracing
  - Efficient & accurate sampling of many VPLs per pixel
  - Separate VPLs sets for all pixels in 3x3 image tiling
  - Illumination filtering with 3x3 pixel footprint
  - Good results with low sampling rate
  - All computations performed on clients
  - Highly scalable solution



Where Are We Going?

- Full System Approach
  - Novel applications [STAR'02, STAR'03]
  - OpenRT ray tracing API [OpenSG'03]
  - Programmable Shading [OpenSG'03]
  - Realtime lighting simulation [RW'02, EG'03]
  - Handling highly complex lighting situations [RW'03]
  - Support for dynamic scene changes [PVG'03]
  - Scalable distributed processing [RW'01, EUROPAR'03]
  - Highly optimized software [EG'01, STAR'02, STAR'03]
  - Scalable ray tracing on CPUs & GPUs [NvidiaU'03]
  - SaarCOR Hardware Architecture [GH'02, C&G'03]

Video

Videos

Where Are We Going?

- Significant Interest from Industry
  - Volkswagen, Audi, DaimlerChrysler, Evobus, …
  - EADS/Airbus, Boeing, …
  - Computer games, entertainment, …
  - Architecture, urban & landscape planning, GIS, …
- Main Topics
  - Large models
  - Better image quality
  - Less preprocessing
  - Fast & reliable results for early decision making
  - Unified visualization platform across applications

Where Are We Going?

- Future Work
  - Better support for dynamic scene changes
  - Scenes of unlimited size
  - Scalable caustics and glossy effects
  - Separate sampling of geometry and shading
  - Support for accurately measured materials & lights
  - Support for other primitives (e.g. splines, points, volumes)
  - Hardware support for ray tracing (GPU, FPGA, ASIC, …)
  - Completely new research and applications, e.g.
    - Simulation-based games, sampled geometry proc., …
  - Practice and experience in a ray tracing context

Interested?

- Contact Points
  - Technology — http://www.OpenRT.de
  - inTrace GmbH — http://www.inTrace.com
Email: info@inTrace.com
  - CG at Saarland U. — http://graphics.cs.uni-sb.de
  - Direct Email — slusallek@cs.uni-sb.de

Acknowledgements

- Contributors
  - Ingo Wald, Carsten Benthin, Jörg Schmittler, Andreas Dietrich
  - Many students
- Collaborators
  - Intel, Stanford U., U. Utah, U. Bonn, U. Tübingen
- Funding
  - Intel, AMD, DFG, Motorola, Saarland

Questions?

Any Questions?