

Interactive RT on PCs

Philipp Slusallek

SIGGRAPH
2001
EXPLORE INTERACTIVITY
AND DIGITAL IMAGERY

For almost 20 years some researchers have argued that RT will eventually become faster than rasterization

And nothing happened

Answering Some Long-Standing Questions:

- Could RT ever be faster than rasterization?
- Where would the crossover point be?
- What would be a good RT architecture (HW & SW)?
- What algorithms and data structure are required?

SIGGRAPH
2001
EXPLORE INTERACTIVITY
AND DIGITAL IMAGERY

Overview

- **Goals**
- **Why should IRT on PCs be possible today?**
- **Coherent Ray-Tracing**
 - General Optimizations
 - Data Parallel SIMD Computation
 - Coherent Algorithms
- **Comparisons**
 - Software Ray-Tracers & OpenGL Hardware
- **Results**

SIGGRAPH
2001
EXPLORE INTERACTION
AND DIGITAL IMAGERY

Why should IRT be possible at all?

- **PC hardware has changed dramatically**
 - Processors became much faster
 - But increase was gradual
 - Increasing gap between speed of CPU and memory
 - But ray-tracing algorithm did not change
 - SIMD extensions
 - But difficult to take advantage of in ray-tracing
 - Fast networking & network of PCs
 - But good distribution of scene data is hard

SIGGRAPH
2001
EXPLORE INTERACTION
AND DIGITAL IMAGERY

Coherent Ray-Tracing

- **Redesign of the ray-tracing algorithm**
 - Adapt the algorithms to today's hardware
 - Avoiding incoherent algorithms & implementation
 - Exposing hidden coherence
- **Optimizations**
 - General optimization techniques (Factor: > 5)
 - SIMD-extensions (Factor: > 2)
 - Distributed computing (Factor: ~#computers)

SIGGRAPH
2001
EXPLORE INTERACTIVITY
AND DIGITAL IMAGERY

General Optimizations: Simplification

Today's CPUs have very long pipelines

- **Simplify the code** to avoid pipeline stalls
 - Choose simple algorithms
 - E.g. BSP-tree traversal simpler than grids
 - Easier to maintain and optimize (e.g. prefetching)
 - Write tight inner loops
 - E.g. better caching and handling of branches
 - Avoid conditionals in inner loops
 - E.g. only triangles are supported

SIGGRAPH
2001
EXPLORE INTERACTIVITY
AND DIGITAL IMAGERY

General Optimizations: Cache

Main memory is too slow for CPU (1:10)
(bandwidth and latency)

- **Keep relevant data in caches**
 - Design algorithms for cache reuse → coherence
 - Align data to cache lines (32 bytes)
 - Separate data according to usage
 - Store intersection data separate from shading data
 - Prefetch data

SIGGRAPH
2001
EXPLORE INTERACTIVITY
AND DIGITAL IMAGERY

General Optimizations: Mailbox

Writes to mailbox in triangle destroys cache

- **Separate mailbox from triangle data**
 - Build small hash table with (rayId, triangleId)
 - Only few triangles are touched for each ray
 - Hash table can mainly stay in first-level cache
 - Simple hash function is enough
 - Important for large multiprocessor systems

SIGGRAPH
2001
EXPLORE INTERACTIVITY
AND DIGITAL IMAGERY

General Optimizations: Ray Packets

Many rays are very similar

e.g. primary and shadow rays, others too

- **Handle rays together in ray packets**
 - Coherence increases with image resolution
 - Reorder computations to be partly breadth-first
 - Load data once and use it for all rays
 - Reduces memory bandwidth
 - Allows use of SIMD extensions

SIGGRAPH
2001
EXPLORE INTERACTIVITY
AND DIGITAL IMAGERY

SIMD Optimization:

Most CPUs provide SIMD extensions

Intel: SSE (Others: 3D-Now!, AltiVec, ...)

- **Use SIMD: higher speed & lower bandwidth**
 - Up to four parallel floating point operations
 - Fetch data once to reduce bandwidth to cache
 - Overhead due to restricted instruction set
 - Program in assembly language

SIGGRAPH
2001
EXPLORE INTERACTIVITY
AND DIGITAL IMAGERY

Coherent Algorithms: Intersection

- **SIMD best used in data parallel fashion**
 - Little instruction-level parallelism
 - Data parallel: 1 ray \leftrightarrow 4 triangles
 - Hard to always have four triangles ready
 - Data parallel: 4 rays \leftrightarrow 1 triangle
 - Must traverse rays in parallel \rightarrow ray packets
 - Standard intersection code
 - Overhead for terminated rays



Coherent Algorithms: Intersection

- **Performance Results**

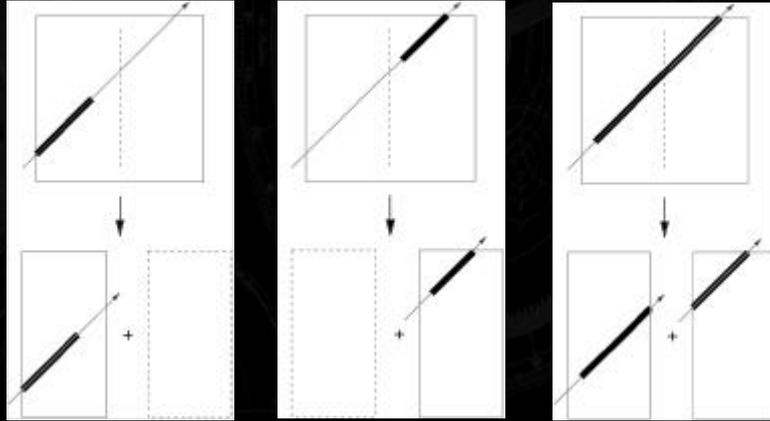
	C	SSE	Speedup
Min. Cycles	78	22	3.5
Max. Cycles	148	40	3.7

- Comparison against already optimized C code
- Amortized cost for SSE code
- 20-36 million intersections/s (P-III, 800 MHz)



Coherent Algorithms: BSP-Traversal

- **Recursive Traversal Algorithm**



Coherent Algorithms: BSP-Traversal

- **SIMD-Traversal**

- Perform intersection and decision in parallel
- Combine decisions flags
 - Make sure order is consistent
 - Same ray origin or sign of direction
- Optimize recursion : Maintain own stack

Coherent Algorithms: BSP-Traversal

- **Overhead of SIMD-Traversal (in %)**

	2x2	4x4	8x8	256 ²	1024 ²
Shirley 6	1.4	4.4	11.8	5.8	1.4
MGF office	2.6	8.2	21.6	10.4	2.6
MGF conf.	3.2	10.6	28.2	12.2	3.2

- Fixed resolution at 1024² (l), fixed 2x2 packet (r)
- Traversal still dominates rendering cost
- Overall speedup factor: 2 to 2.3



Coherent Algorithms: Shading

- **SIMD Phong-Shading**

- Fixed cost per image
- Rearrange data from ray packets
 - Different depth: non-coherent shadow rays
 - Different materials: different shaders
- Algorithm
 - Parallel shadow rays to light sources
 - SIMD shading using shadow flags
- Constant shading & texturing cost (<10%)
- Procedural shading is easy (noise)

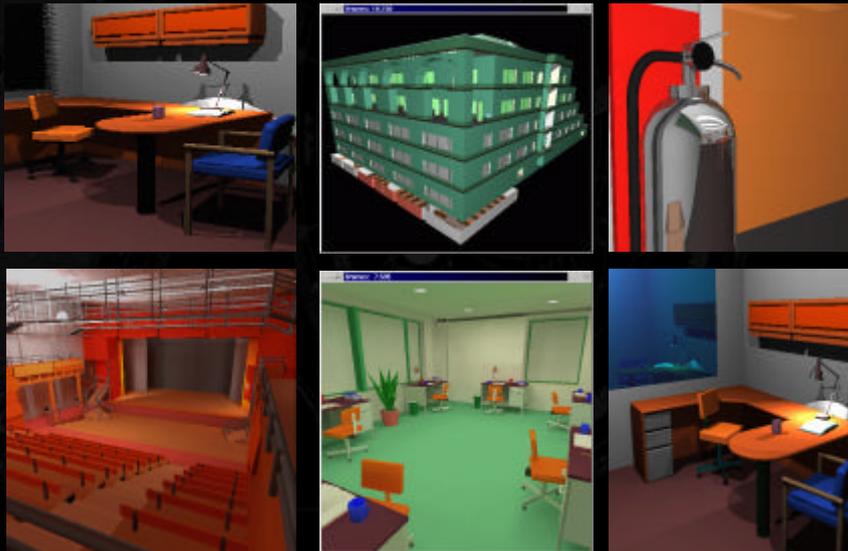


Coherent Ray-Tracing : Summary

- **Speedup**
 - Factor >5: General optimizations
 - Factor >2: SIMD computations
 - Further optimizations
 - Better prefetching, more efficient shading
- **Performance**
 - 200K to 1.5M primary rays/s
 - Almost linear in # of reflection & shadow rays

SIGGRAPH
2001
EXPLORE INTERACTION
AND DIGITAL IMAGERY

Comparison: Test Scenes



Comparison: Software Ray-Tracers

- Time per primary ray (1 CPU, 512², in ms)

	Tris	Rayshade	POV	RTRT	Factor	fps
MGF office	40K	29.0	22.9	2.1	10.9	1.8
MGF conf	256K	36.1	29.6	2.3	12.8	1.6
MGF theater	680K	56.0	57.2	3.6	15.5	1.1
Library	907K	72.1	50.5	3.4	14.8	1.1
Soda, Floor5	2.5M	OOM	OOM	2.9	¥	1.5
Soda Hall	8M	OOM	OOM	4.5	¥	0.8

- Main memory: RTRT 256MB, others up 1GB
- Rayshade: Best grid resolution



Comparison: OpenGL Hardware

- Frame rate with SGI-Performer (512², fps)

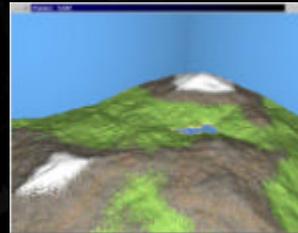
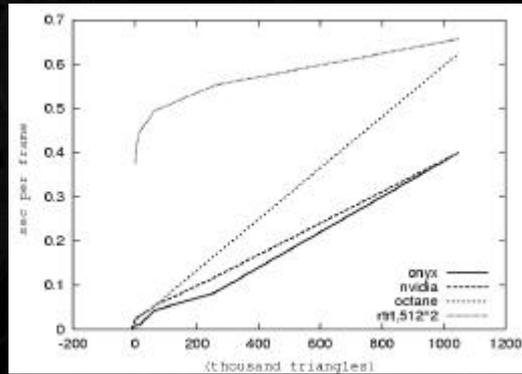
	Tris	Octane	Onyx	PC	RTRT
MGF office	40K	>24	>36	12.7	1.8
MGF conf	256K	>5	>10	5.4	1.6
MGF theater	680K	0.4	6-12	1.5	1.1
Library	907K	1.5	4	1.6	1.1
Soda, Floor5	2.5M	0.5	1.5	0.6	1.5
Soda Hall	8M	OOM	OOM	OOM	0.8

- HW: Octane V8, Onyx3/IR3, Geforce II GTS
- CPUs: Onyx: 8, PC: 2, Other: 1



Comparison: Scaling with Scene Size

- **Render time of subsampled terrain (spf)**



- Typical linear scaling of rasterization HW
- No occlusion to take advantage of

Conclusions

- **Interactive Ray-Tracing is here to stay**
 - Huge speedup to other SW implementation
 - Coherent algorithms, caching, and SIMD
 - Logarithmic complexity in model size
 - SW beats hardware for > 1M triangles (512x512)
 - RT no longer limited by memory bandwidth
- ➔ **Ray-Tracing is faster for large models**
Even if comparing SW to HW

SIGGRAPH
2001
EXPLORE INTERACTIVITY
AND DIGITAL IMAGERY