Are we done with Ray Tracing?

Alexander Keller

Schedule

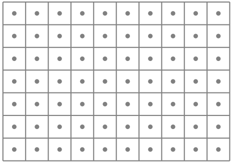
Course web page at https://sites.google.com/view/arewedonewithraytracing

- 9:00 Are we done with Ray Tracing?
  - Alexander Keller, NVIDIA
- 9:40 Acceleration Data Structure Hardware (and Software)
  - Timo Viitanen, NVIDIA
- 10:15 State-of-the-Art and Challenges in Game Ray Tracing
  - Colin Barré-Brisebois, SEED - Electronic Arts
- break
- 11:05 Reconstruction for Real-Time Path Tracing
  - Christoph Schied, Facebook Reality Labs
- 11:40 From Raster to Rays in Games
  - Morgan McGuire, NVIDIA

From Rasterization to Ray Tracing

Principles of Image Synthesis

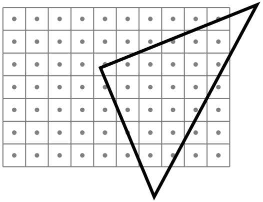
Rasterization



From Rasterization to Ray Tracing

Principles of Image Synthesis

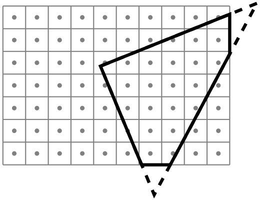
Rasterization



From Rasterization to Ray Tracing

Principles of Image Synthesis

Rasterization

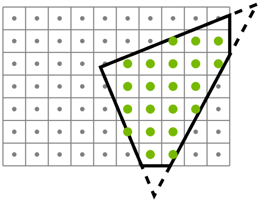


clipping

From Rasterization to Ray Tracing

Principles of Image Synthesis

Rasterization



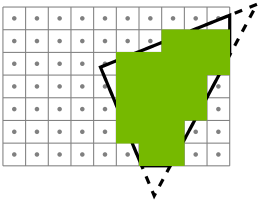
clipping

Z-buffer

From Rasterization to Ray Tracing

Principles of Image Synthesis

Rasterization

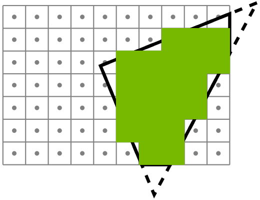
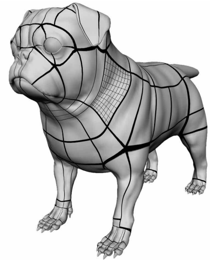


clipping

Z-buffer

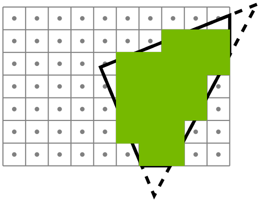
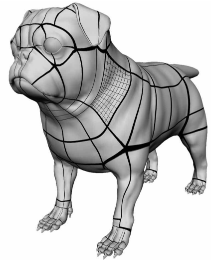
From Rasterization to Ray Tracing

Principles of Image Synthesis

Rasterization	Reyes
	
clipping	dicing
Z-buffer	kind of Z-Buffer

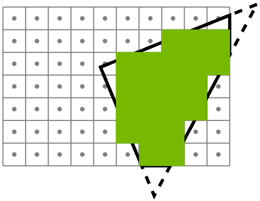
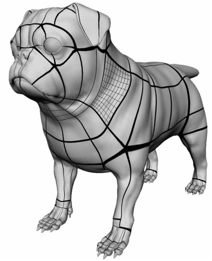
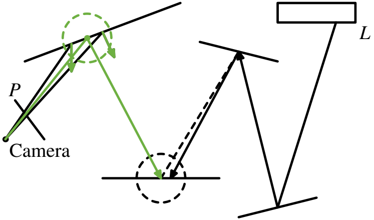
From Rasterization to Ray Tracing

Principles of Image Synthesis

Rasterization	Reyes
	
clipping	dicing
Z-buffer	kind of Z-Buffer
shadow maps	shadow maps

From Rasterization to Ray Tracing

Principles of Image Synthesis

Rasterization	Reyes	Ray Tracing
		
clipping	dicing	acceleration data structure
Z-buffer	kind of Z-Buffer	tracing rays with arbitrary origins
shadow maps	shadow maps	shadow rays

Ray Tracing

How it started

- 1974: "brute-force approach" of rasterization "ridiculously expensive"

  ► A characterization of ten hidden-surface algorithms

- "Ray tracing is the future and ever will be"

  ► SIGGRAPH 2013 Course

Ray Tracing

How it started

- 1974: "brute-force approach" of rasterization "ridiculously expensive"

  ► A characterization of ten hidden-surface algorithms

- "Ray tracing is the future and ever will be"

  ► SIGGRAPH 2013 Course

- The Path Tracing Revolution in Movie Industry

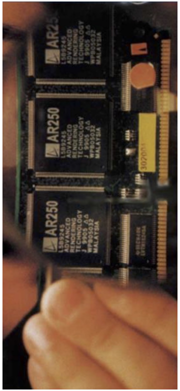
  ► SIGGRAPH 2015 Course
  ► ACM Transactions on Graphics, Volume 37 Issue 3, August 2018
  ► Vectorized Production Path Tracing
  ► The Iray Light Transport Simulation and Rendering System
  ► SIGGRAPH 2018 Course

Ray Tracing Hardware

Ray Tracing Hardware

1995: ART's RenderDrive: Ray Tracing Hardware from before the GPU

- top of bounding volume hierarchy (BVH) cached in each processor
  - geometry streamed against ray buffers
  - processors flag bounding volume intersection to demand broadcast of children
    - processors running idle when rays diverge
    - new rays can be started while tracing old ones



Ray Tracing Hardware

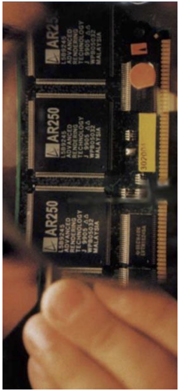
1995: ART's RenderDrive: Ray Tracing Hardware from before the GPU

- top of bounding volume hierarchy (BVH) cached in each processor
  - geometry streamed against ray buffers
  - processors flag bounding volume intersection to demand broadcast of children
    - processors running idle when rays diverge
    - new rays can be started while tracing old ones

- shading processor

- rays sorted by shader in order to increase coherency

- fire-and-forget ray tracing: Results added to pixels

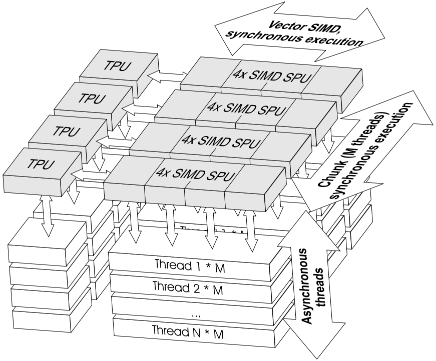


▶ Cold Chips: ART's RenderDrive

Ray Tracing Hardware

2005: Ray Processing Unit

- architecture

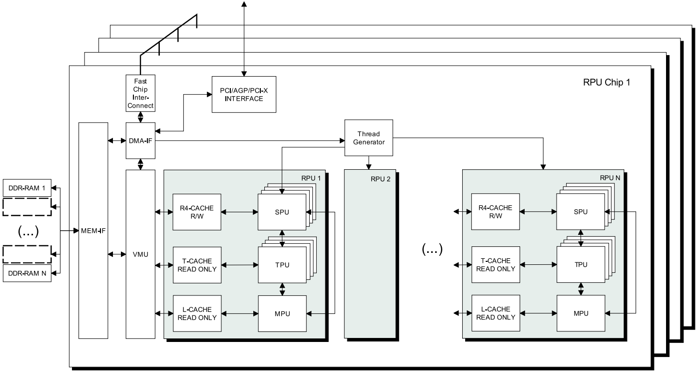


▶ RPU: A Programmable Ray Processing Unit for Realtime Ray Tracing

Ray Tracing Hardware

2005: Ray Processing Unit

- architecture

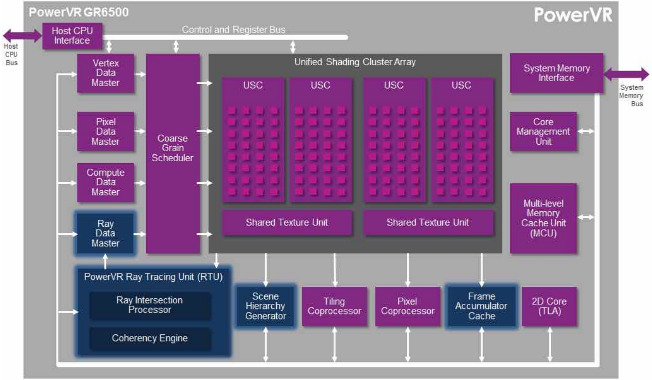


► RPU: A Programmable Ray Processing Unit for Realtime Ray Tracing

Ray Tracing Hardware

2014: Imagination Technologies

- architecture

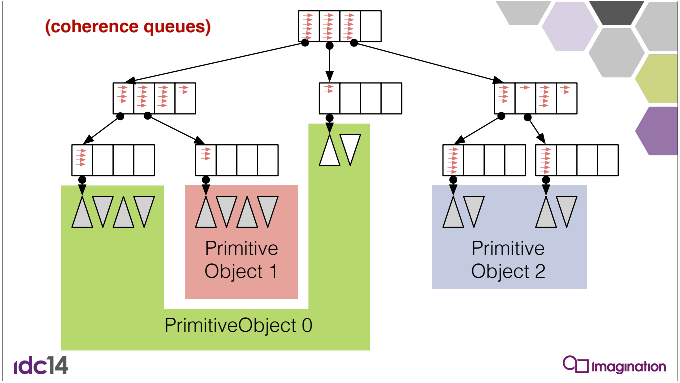


► Introduction to PowerVR Ray Tracing

Ray Tracing Hardware

2014: Imagination Technologies

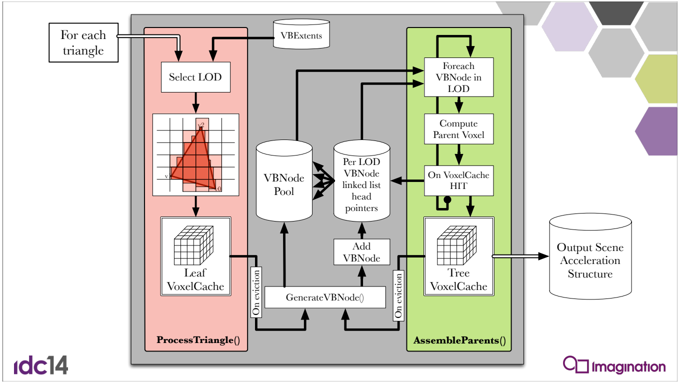
- queuing rays to nodes



Ray Tracing Hardware

2014: Imagination Technologies

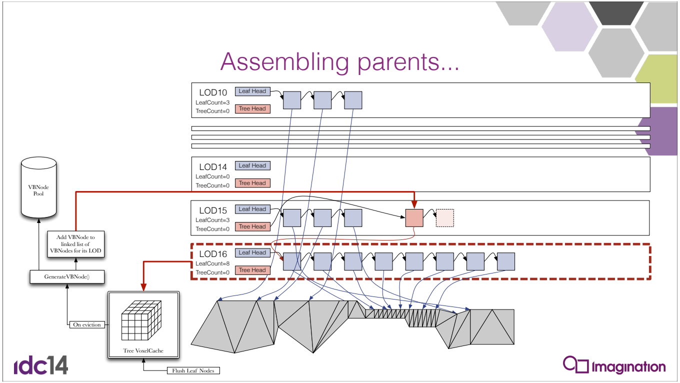
- streaming bounding volume hierarchy construction



Ray Tracing Hardware

2014: Imagination Technologies

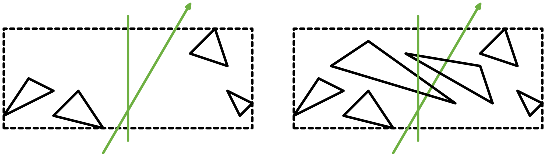
- streaming bounding volume hierarchy construction



Auxiliary acceleration data structure

Efficient culling

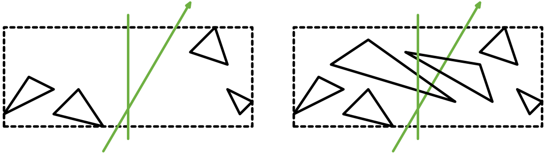
partitioning space: *k*d-tree or (nested) uniform grids



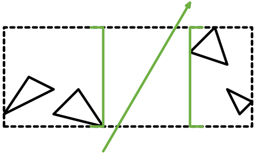
Auxiliary acceleration data structure

Efficient culling

partitioning space: *k*d-tree or (nested) uniform grids



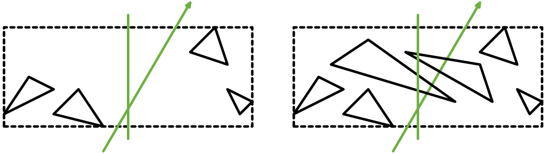
partitioning object lists: convex bounding volumes, e.g. axis aligned boxes, spheres, planes, ...



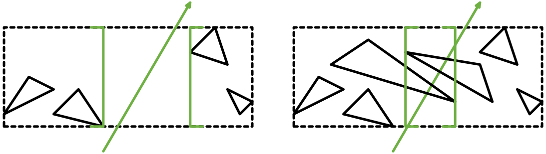
Auxiliary acceleration data structure

Efficient culling

partitioning space: *k*d-tree or (nested) uniform grids



partitioning object lists: convex bounding volumes, e.g. axis aligned boxes, spheres, planes, ...



NVIDIA

Auxiliary acceleration data structure

Parallel bounding volume hierarchy (BVH) construction

- linear bounding volume hierarchy (LBVH)

Auxiliary acceleration data structure

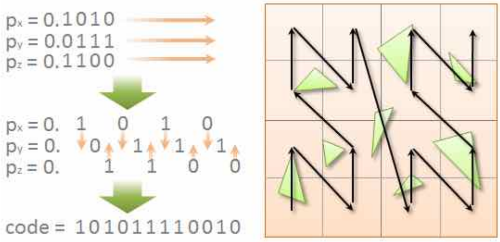
Parallel bounding volume hierarchy (BVH) construction

- linear bounding volume hierarchy (LBVH)
  - map object centroids (p_x, p_y, p_z) onto the unit cube
  - interleaving the bits of the quantized coordinates yields the Morton code of (p_x, p_y, p_z)
  - enumerating the sorted Morton code keys corresponds to a traversal along the Z-curve

Auxiliary acceleration data structure

Parallel bounding volume hierarchy (BVH) construction

- linear bounding volume hierarchy (LBVH)
  - map object centroids (p_x, p_y, p_z) onto the unit cube
  - interleaving the bits of the quantized coordinates yields the Morton code of (p_x, p_y, p_z)
  - enumerating the sorted Morton code keys corresponds to a traversal along the Z-curve

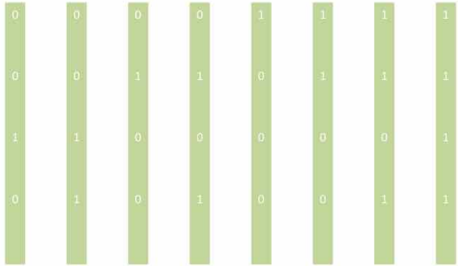


▶ Thinking Parallel, Part III: Tree Construction on the GPU

Auxiliary acceleration data structure

Parallel bounding volume hierarchy (BVH) construction

- *n* sorted Morton code keys

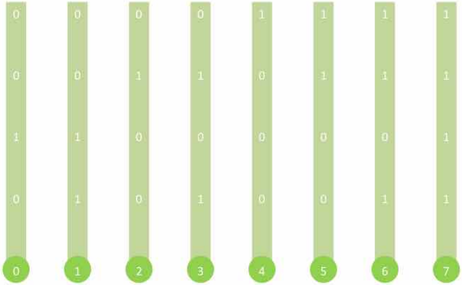


► Fast and simple agglomerative LBVH construction

Auxiliary acceleration data structure

Parallel bounding volume hierarchy (BVH) construction

- *n* sorted Morton code keys stored in an array of leaves

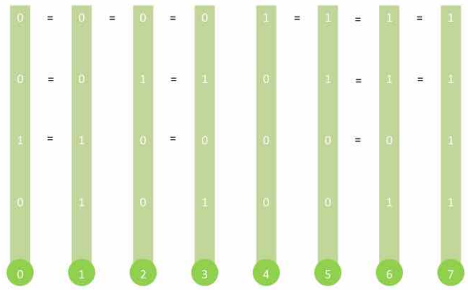


▶ Fast and simple agglomerative LBVH construction

Auxiliary acceleration data structure

Parallel bounding volume hierarchy (BVH) construction

- `xor`-ing neighboring Morton codes yields a metric of how close they are along the Z-curve



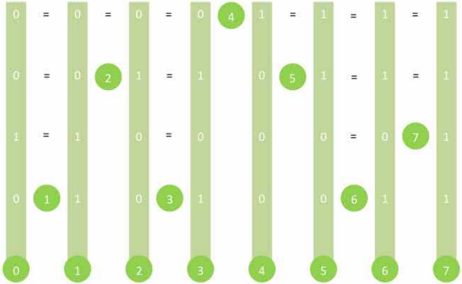
▶ Fast and simple agglomerative LBVH construction

 NVIDIA

Auxiliary acceleration data structure

Parallel bounding volume hierarchy (BVH) construction

- the height of the $n-1$ inner nodes represents this metric, which can be computed on-the-fly



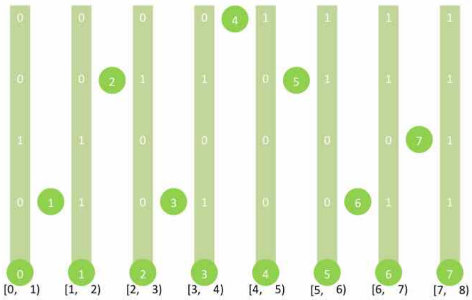
► Fast and simple agglomerative LBVH construction

NVIDIA

Auxiliary acceleration data structure

Parallel bounding volume hierarchy (BVH) construction

- each leaf is assigned an interval, whose boundaries reference its neighboring inner nodes

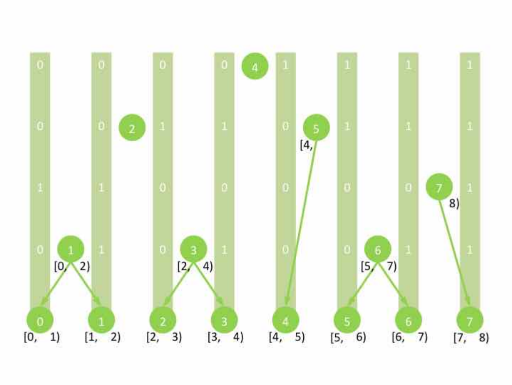


► Fast and simple agglomerative LBVH construction

Auxiliary acceleration data structure

Parallel bounding volume hierarchy (BVH) construction

- parent of a node is the most similar one of the two inner nodes indexed by interval boundaries



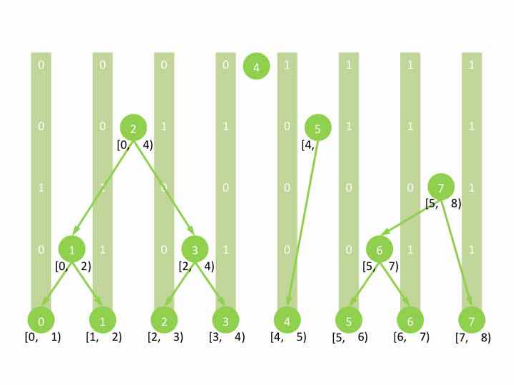
► Fast and simple agglomerative LBVH construction

NVIDIA.

Auxiliary acceleration data structure

Parallel bounding volume hierarchy (BVH) construction

- whenever both children of an inner node are set, its parent can be determined the same way

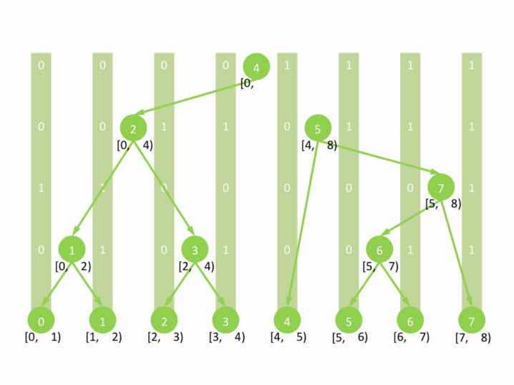


► Fast and simple agglomerative LBVH construction

Auxiliary acceleration data structure

Parallel bounding volume hierarchy (BVH) construction

- left children propagate up their left interval boundary (right children in analogy)

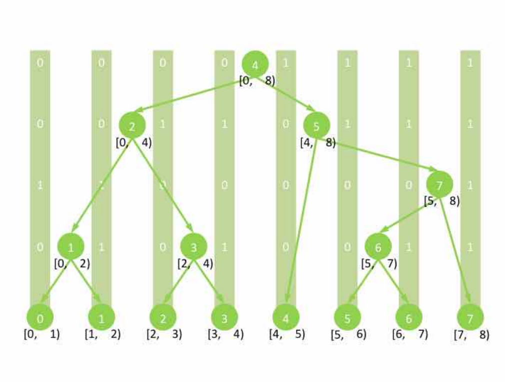


► Fast and simple agglomerative LBVH construction

Auxiliary acceleration data structure

Parallel bounding volume hierarchy (BVH) construction

- the root node interval spans the range of leaves and is copied to the inner node with index 0



► Fast and simple agglomerative LBVH construction

Ray Tracing

Frameworks and application programming interfaces (API)

- Ray Tracing Engine
  - NVIDIA OptiX
- High performance ray tracing kernels
  - Intel Embree
- High-efficiency, high performance heterogeneous Ray Tracing Intersection Library
  - AMD FireRays SDK

Ray Tracing

Frameworks and application programming interfaces (API)

- Ray Tracing Engine
  - NVIDIA OptiX
- High performance ray tracing kernels
  - Intel Embree
- High-efficiency, high performance heterogeneous Ray Tracing Intersection Library
  - AMD FireRays SDK
- The Quest for the Ray Tracing API
  - SIGGRAPH 2016 Course
  - Introduction to NVIDIA RTX and DirectX Ray Tracing
  - Introduction to Real-Time Ray Tracing with Vulkan
  - Metal for Ray Tracing Acceleration

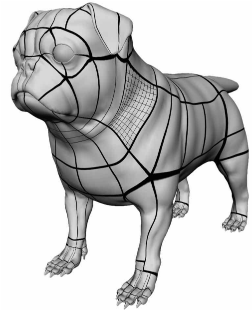
Surface Representation

Surface Representation

Meshes

- patchwork geometry
  - irregular topology at top level
  - regular topology at bottom level

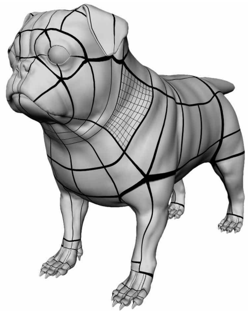
- watertight along shared edges
  - numerical issues
  - compression by quantization

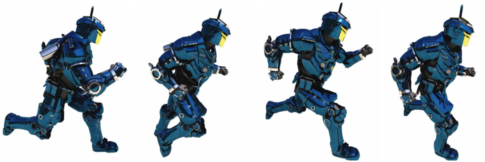


Surface Representation

Meshes

- patchwork geometry
  - irregular topology at top level
  - regular topology at bottom level
- watertight along shared edges
  - numerical issues
  - compression by quantization
- subdivision surfaces
  - adaptive subdivision
  - mathematically meaningful





► Massively Parallel Stackless Ray Tracing of Catmull-Clark Subdivision Surfaces

Surface Representation

Displacement or no displacement

- diffuse texture

► Direct ray tracing of displacement mapped triangles

Surface Representation

Displacement or no displacement

- normal mapping

▶ Direct ray tracing of displacement mapped triangles

Surface Representation

Displacement or no displacement

- parallax occlusion mapping

▶ Direct ray tracing of displacement mapped triangles

Surface Representation

Displacement or no displacement

- displacement mapping

▶ Direct ray tracing of displacement mapped triangles

Surface Representation

Displacement or no displacement

- base mesh: 8,750 vertices and 16,636 vertex indices

▶ Geometry courtesy Henning Sanden

Surface Representation

Displacement or no displacement

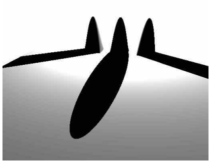
- displaced mesh: 6,327,463 vertices and 12,629,248 vertex indices

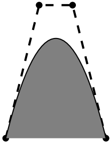
▶ Geometry courtesy Henning Sanden

Surface Representation

Displacement or no displacement

- level of detail may affect visibility

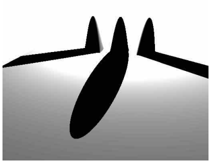


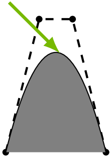


Surface Representation

Displacement or no displacement

- level of detail may affect visibility

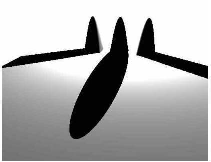


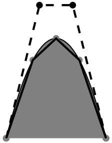


Surface Representation

Displacement or no displacement

- level of detail may affect visibility

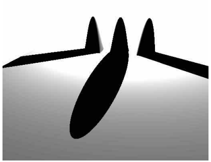


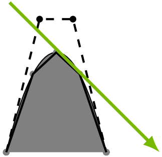


Surface Representation

Displacement or no displacement

- level of detail may affect visibility

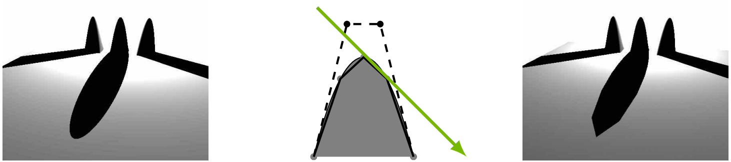




Surface Representation

Displacement or no displacement

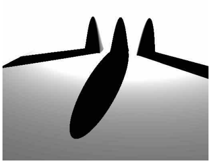
- level of detail may affect visibility

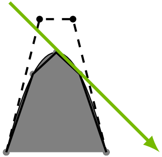


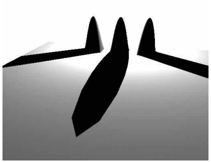
Surface Representation

Displacement or no displacement

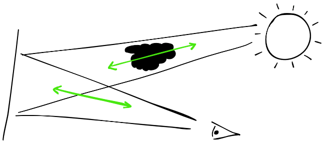
- level of detail may affect visibility







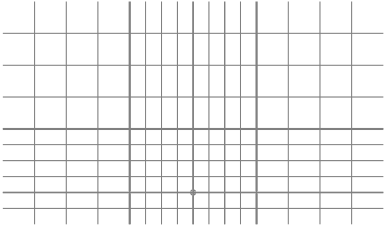
- knowing the level of detail



Surface Representation

Floating point arithmetic

- define 0/0, avoid exceptions, and use consistent numeric formulations
- logarithmic spacing of floating point numbers

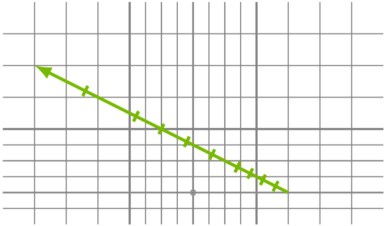


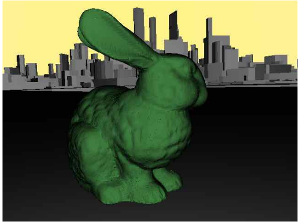
▶ What every computer scientist should know about floating-point arithmetic

Surface Representation

Floating point arithmetic

- define 0/0, avoid exceptions, and use consistent numeric formulations
- logarithmic spacing of floating point numbers
  - ray direction and scale matter



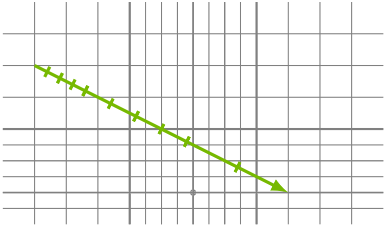


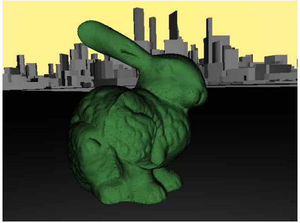
► What every computer scientist should know about floating-point arithmetic

Surface Representation

Floating point arithmetic

- define 0/0, avoid exceptions, and use consistent numeric formulations
- logarithmic spacing of floating point numbers
  - ray direction and scale matter



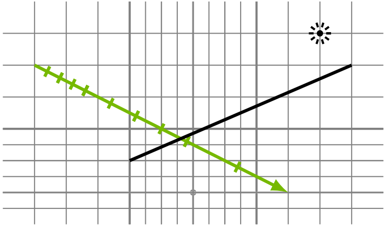


► What every computer scientist should know about floating-point arithmetic

Surface Representation

Floating point arithmetic

- define 0/0, avoid exceptions, and use consistent numeric formulations
- logarithmic spacing of floating point numbers
  - ray direction and scale matter
  - self intersection problem

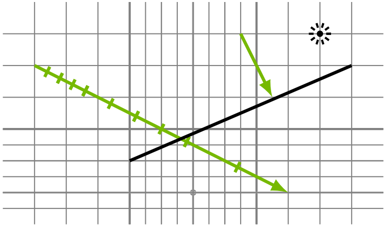


▶ What every computer scientist should know about floating-point arithmetic

Surface Representation

Floating point arithmetic

- define 0/0, avoid exceptions, and use consistent numeric formulations
- logarithmic spacing of floating point numbers
  - ray direction and scale matter
  - self intersection problem

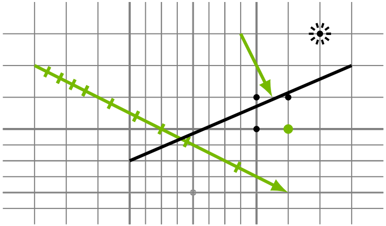


▶ What every computer scientist should know about floating-point arithmetic

Surface Representation

Floating point arithmetic

- define 0/0, avoid exceptions, and use consistent numeric formulations
- logarithmic spacing of floating point numbers
  - ray direction and scale matter
  - self intersection problem

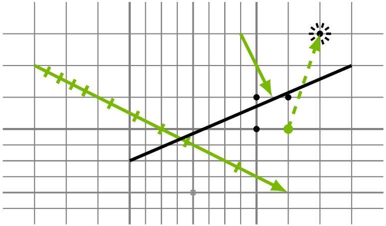


▶ What every computer scientist should know about floating-point arithmetic

Surface Representation

Floating point arithmetic

- define 0/0, avoid exceptions, and use consistent numeric formulations
- logarithmic spacing of floating point numbers
  - ray direction and scale matter
  - self intersection problem

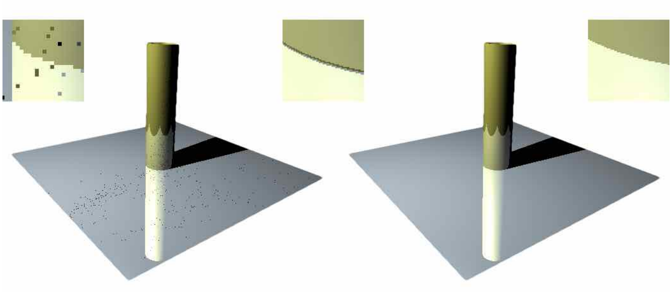


▶ What every computer scientist should know about floating-point arithmetic

Surface Representation

Dealing with the ill-posed self intersection problem

- post iteration to refine hitpoint and guarantee side of plane
- robust epsilon for offsetting



▶ It’s really not a rendering bug, you see...

Ray Tracing Performance

Ray Tracing Performance

Amortizing and balancing the work of a frame

- preprocessing
  - bounding volume hierarchy construction and update on demand

- ray tracing and shading

- postprocessing
  - filtering
  - upscaling

Ray Tracing Performance

Amortizing and balancing the work of a frame

- preprocessing
  - bounding volume hierarchy construction and update on demand

- ray tracing

- shading

- postprocessing
  - filtering
  - upscaling

Ray Tracing Performance

Amortizing and balancing the work of a frame

- preprocessing
  - bounding volume hierarchy construction and update on demand
  - parameter baking on demand
- ray tracing
- shading
- postprocessing
  - filtering
  - upscaling

Ray Tracing Performance

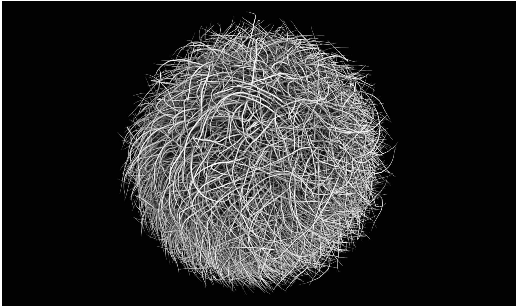
Amortizing and balancing the work of a frame

- preprocessing
  - bounding volume hierarchy construction and update on demand
  - parameter baking on demand
- ray tracing
- shadow rays
- shading
- postprocessing
  - filtering
  - upscaling

Ray Tracing Performance

Geometry

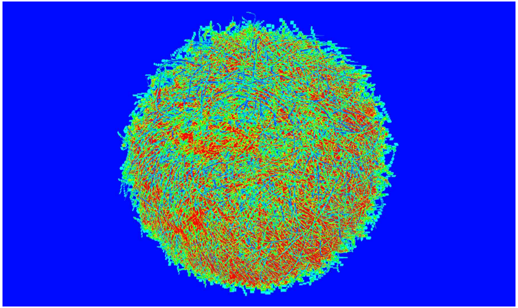
- overlapping bounding volumes
  - partitioning vs. memory footprint
  - e.g. foliage, fur, hair



▸ Geometry courtesy NVIDIA, Giorgio Luciano, and Epic Games

Ray Tracing Performance
Geometry

- overlapping bounding volumes
  - partitioning vs. memory footprint
  - e.g. foliage, fur, hair

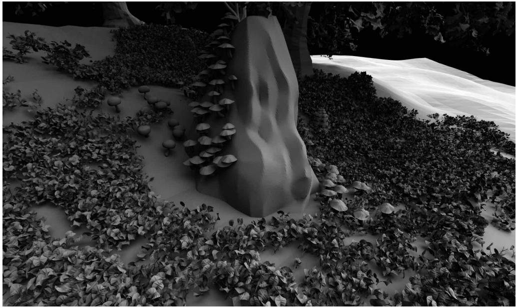


► Geometry courtesy NVIDIA, Giorgio Luciano, and Epic Games

Ray Tracing Performance

Geometry

- overlapping bounding volumes
  - partitioning vs. memory footprint
  - e.g. foliage, fur, hair
- high valence vertices, e.g. triangle fans
- silhouettes, i.e. the cost of missing geometry
  - similar for ray tracing distance fields

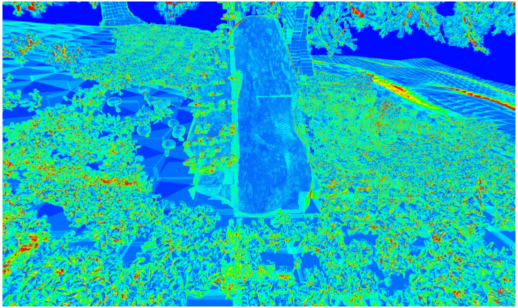


▶ Geometry courtesy NVIDIA, Giorgio Luciano, and Epic Games

Ray Tracing Performance

Geometry

- overlapping bounding volumes
  - partitioning vs. memory footprint
  - e.g. foliage, fur, hair
- high valence vertices, e.g. triangle fans
- silhouettes, i.e. the cost of missing geometry
  - similar for ray tracing distance fields



► Geometry courtesy NVIDIA, Giorgio Luciano, and Epic Games

Ray Tracing Performance

Geometry

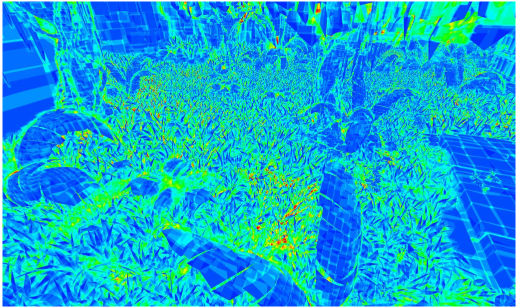
- overlapping bounding volumes
  - partitioning vs. memory footprint
  - e.g. foliage, fur, hair
- high valence vertices, e.g. triangle fans
- silhouettes, i.e. the cost of missing geometry
  - similar for ray tracing distance fields
- parametric, procedural, and on-demand geometry
  - instances
  - skinning

- Geometry courtesy NVIDIA, Giorgio Luciano, and Epic Games

Ray Tracing Performance

Geometry

- overlapping bounding volumes
  - partitioning vs. memory footprint
  - e.g. foliage, fur, hair
- high valence vertices, e.g. triangle fans
- silhouettes, i.e. the cost of missing geometry
  - similar for ray tracing distance fields
- parametric, procedural, and on-demand geometry
  - instances
  - skinning



▶ Geometry courtesy NVIDIA, Giorgio Luciano, and Epic Games

Ray Tracing Performance

Geometry and Shading

- geometry vs. α-maps or even procedural trimming and shading

```
float a = atan(q.x,q.y);
float r = length(q);
float s = 0.5 + 0.5*sin(3.0*a + iGlobalTime);
float g = sin(1.57+3.0*a+iGlobalTime);
float d = 0.15 + 0.3*sqrt(s) + 0.15*g*g;
float h = r/d;
float f = 1.0-smoothstep( 0.95, 1.0, h );
h *= 1.0-0.5*(1.0-h)*smoothstep(0.95+0.05*h,1.0,sin(3.0*a+iGlobalTime));

vec3 bcol = vec3(0.9+0.1*q.y,1.0,0.9-0.1*q.y);
bcol *= 1.0 - 0.5*r;
h = 0.1 + h;
vec3 col = mix( bcol, 1.2*vec3(0.6*h,0.2+0.5*h,0.0), f );
```

► Inigo Quilez

Ray Tracing Performance

Shading

- incoherent texture access
  - stochastic interpolation of bilinear texture interpolation

► Vectorized Production Path Tracing

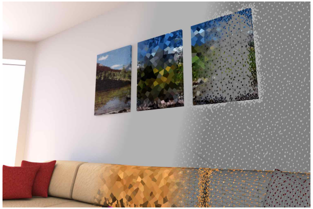
Ray Tracing Performance

Shading

- incoherent texture access
  - stochastic interpolation of bilinear texture interpolation

▶ Vectorized Production Path Tracing

- strict separation of material definition and rendering algorithms
  - generic BSDF models
  - baking BSDF parameters



▶ The Material Definition Language

▶ GI Next: Global Illumination for Production Rendering on GPUs

▶ Manuka: A Batch-Shading Architecture for Spectral Path Tracing in Movie Production

Ray Tracing Performance

What happened since Breakpoint 2005?

- 512 x 384 pixels at 5-10 fps on a Pentium 4M, everything computed live from scratch

► To trace or not to trace - that is the question. State of the art in fast ray tracing (FaRT).

Ray Tracing Performance

Load balancing

- tile-based workload distribution

Ray Tracing Performance

Load balancing

- tile-based workload distribution

Ray Tracing Performance

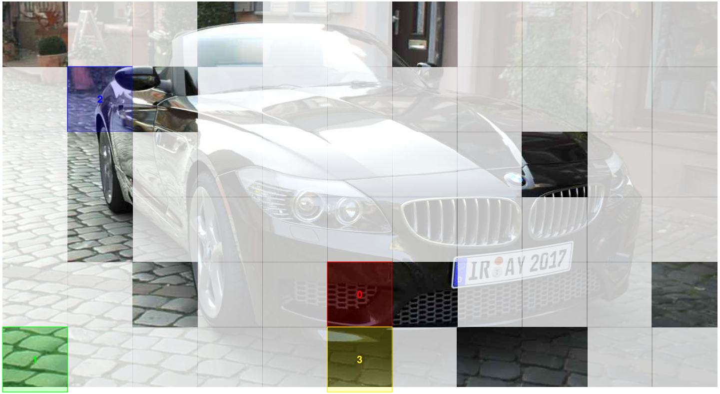
Load balancing

- tile-based workload distribution

Ray Tracing Performance

Load balancing

- tile-based workload distribution



Ray Tracing Performance

Load balancing

- tile-based workload distribution

Ray Tracing Performance

Load balancing

- scanline-based workload distribution

Ray Tracing Performance

Load balancing

- non-uniform cost of pixels/samples
  - not known a priori and thus cannot be taken into account to compute the optimal distribution of work

- non-uniform performance of processing units
  - common in systems that combine GPUs from different generations with a CPU and in network clusters

Ray Tracing Performance

Load balancing

- non-uniform cost of pixels/samples
  - not known a priori and thus cannot be taken into account to compute the optimal distribution of work

$\Rightarrow$ assign a fixed subset of the workload to each processor per frame

- non-uniform performance of processing units
  - common in systems that combine GPUs from different generations with a CPU and in network clusters

Ray Tracing Performance

Load balancing

- non-uniform cost of pixels/samples
  - not known a priori and thus cannot be taken into account to compute the optimal distribution of work

$\Rightarrow$ assign a fixed subset of the workload to each processor per frame

- non-uniform performance of processing units
  - common in systems that combine GPUs from different generations with a CPU and in network clusters

$\Rightarrow$ based on performance measurements, adapt size of the subset between frames

Ray Tracing Performance

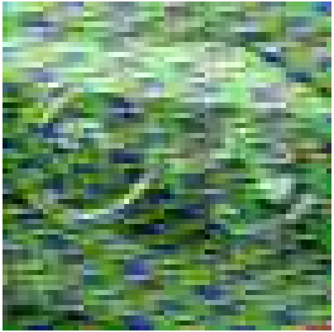
Load balancing

- adaptive striping with relative performance weights w_k of 10%, 50%, 25%, and 15%

Ray Tracing Performance

Load balancing

- adaptive striping with relative performance weights w_k of 10%, 50%, 25%, and 15%



  - note how the headlight pixels are distributed among processing units

▶ Ray Tracing Gems, Ch. 10

The headlight challenge: Sequential singular chains

Are we done with Ray Tracing?

Not quite yet...

- mathematically meaningful level of detail
- separation of shading and geometry
- scalable parallel algorithms for singular chains

Are we done with Ray Tracing?

Not quite yet...

- mathematically meaningful level of detail
- separation of shading and geometry
- scalable parallel algorithms for singular chains
- and of course, ray tracing means sampling

Are we done with Ray Tracing?

Ray tracing is here to stay

- 9:40 Acceleration Data Structure Hardware (and Software)
  - Timo Viitanen, NVIDIA
- 10:15 State-of-the-Art and Challenges in Game Ray Tracing
  - Colin Barré-Brisebois, SEED - Electronic Arts
- break
- 11:05 Reconstruction for Real-Time Path Tracing
  - Christoph Schied, Facebook Reality Labs
- 11:40 From Raster to Rays in Games
  - Morgan McGuire, NVIDIA
- check `https://sites.google.com/view/arewedonewithraytracing`