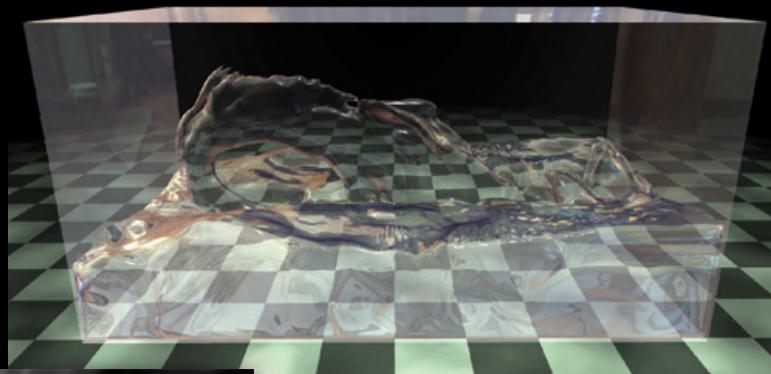


# Octree-Based Sparse Voxelization for Real-Time Global Illumination

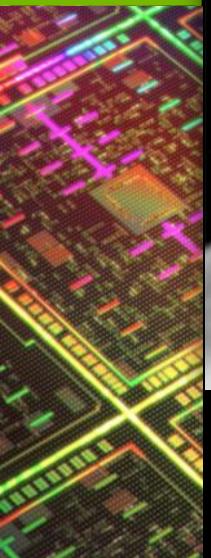
Cyril Crassin  
*NVIDIA Research*

# Voxel representations

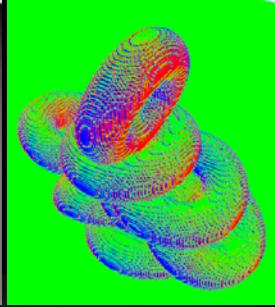
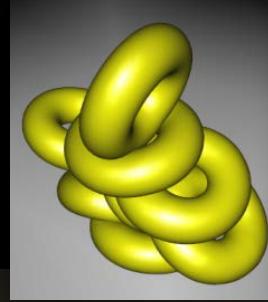
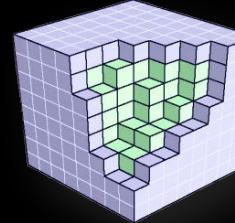
- 



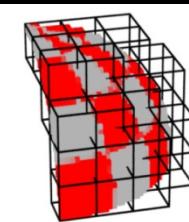
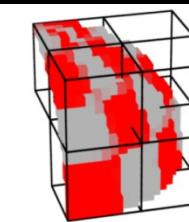
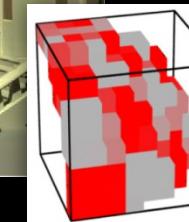
Crane et al. (NVIDIA) 2007



Christensen and Batali (Pixar) 2004

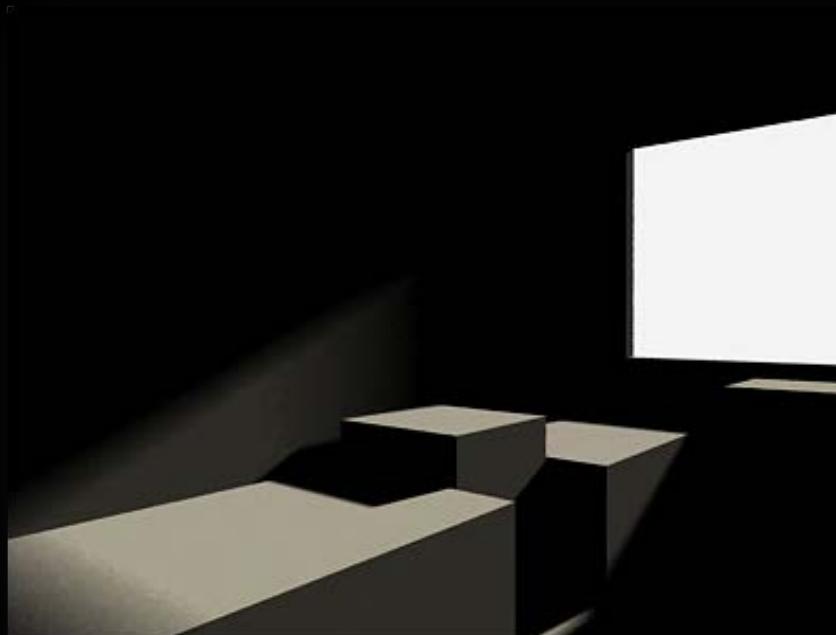


Allard et al. 2010

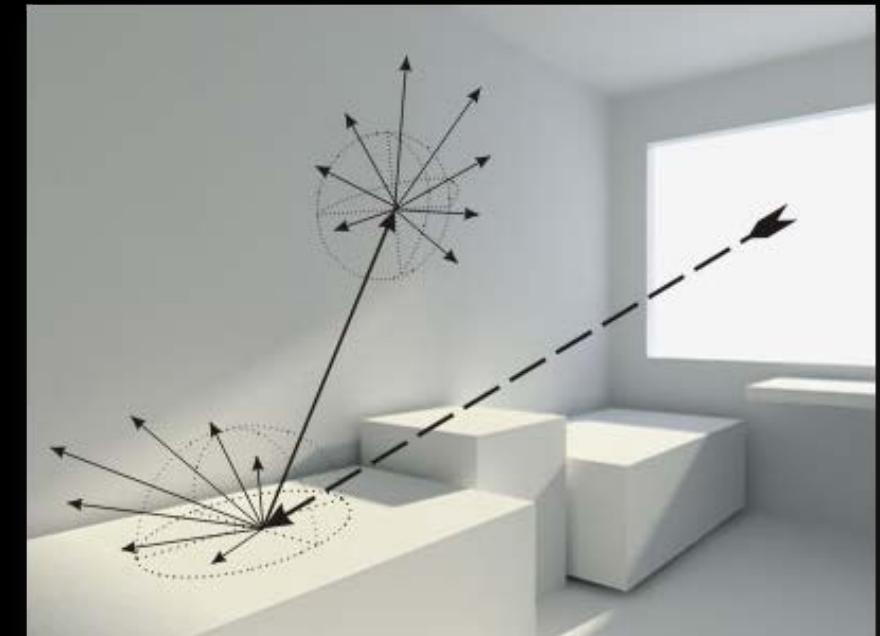


# Global Illumination

- Indirect effects
- Important for realistic image synthesis



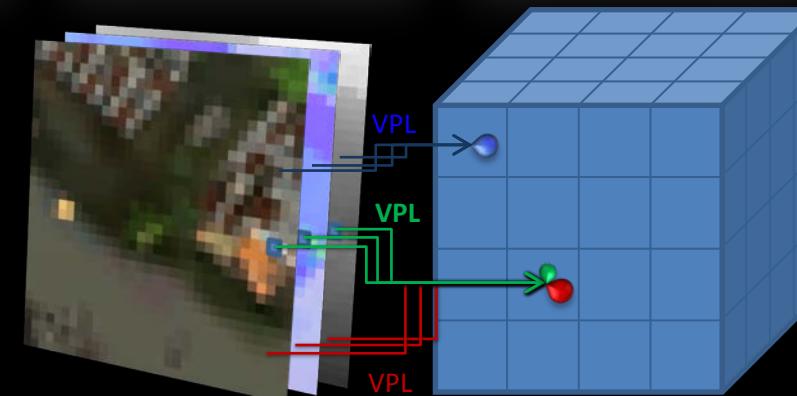
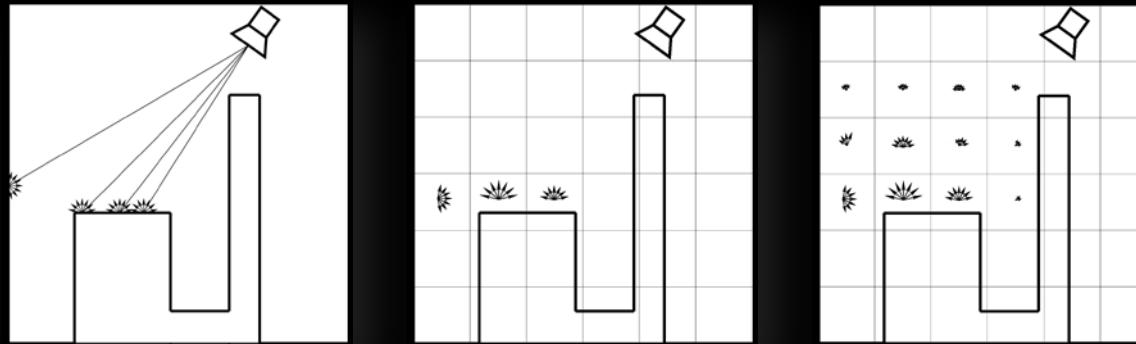
Direct lighting



Direct+Indirect lighting

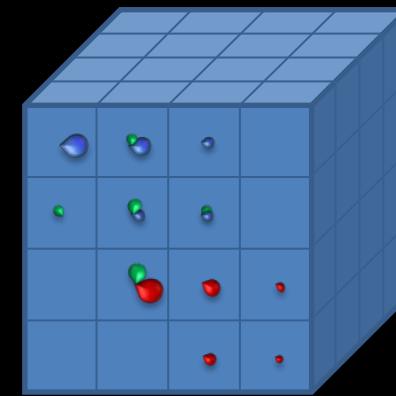
# Light Propagation Volumes

- [Kaplanyan & Dachsbacher 2010]
  - Limited resolution + Mostly diffuse



Reflective shadow maps

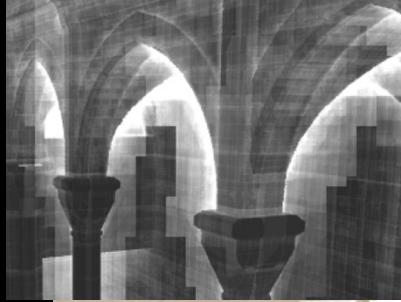
Radiance volume gathering



Iterative propagation

# Sparse Voxel Octree

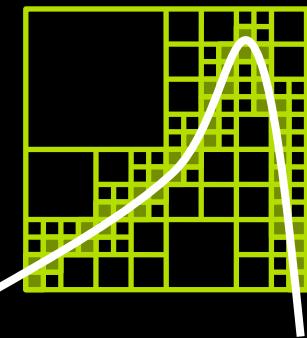
- Detailed geometry rendering
  - Structured LODs



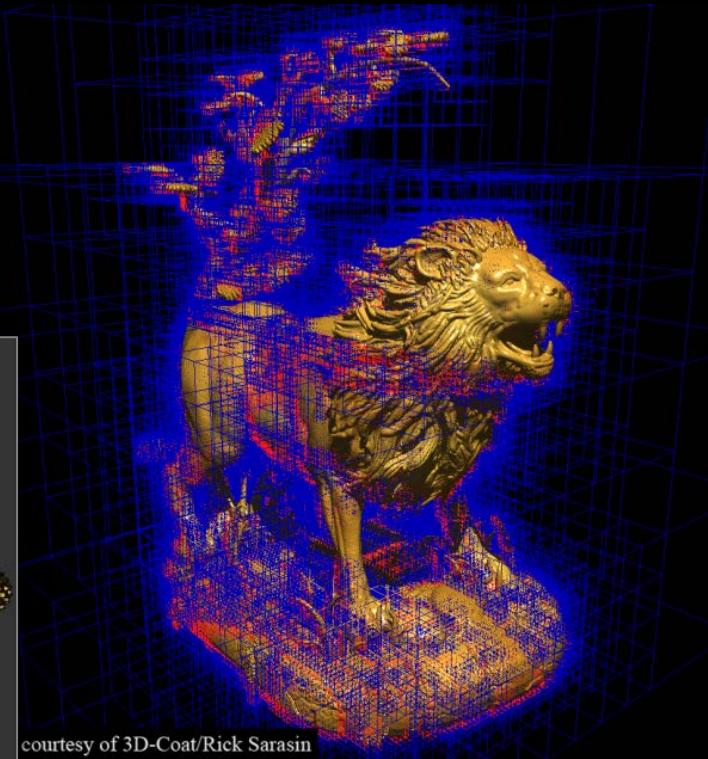
*Laine and Karras (NVIDIA) 2010*



*Olick. 2008*



*Crassin et al. 2009  
(GigaVoxels)*



courtesy of 3D-Coat/Rick Sarasin

# *Interactive indirect illumination using voxel cone tracing*

120 FPS @ 512x512 -- 16 FPS @ FullHD



# Publications

## *Interactive indirect illumination using voxel cone tracing*

C. Crassin, F. Neyret, M. Sainz, S. Green, E. Eisemann

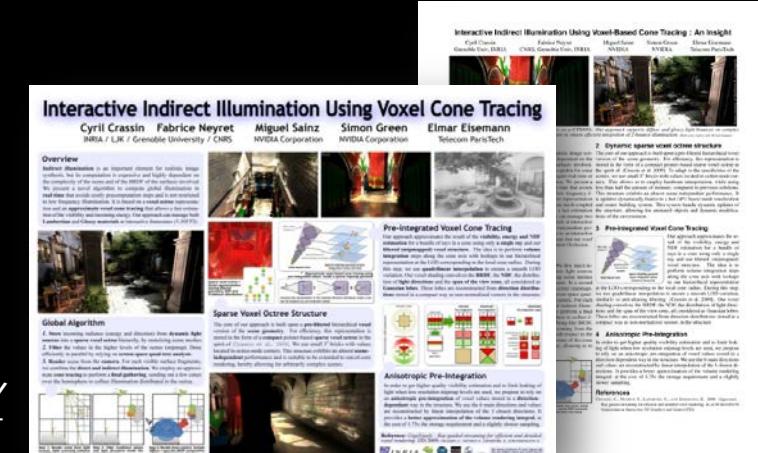
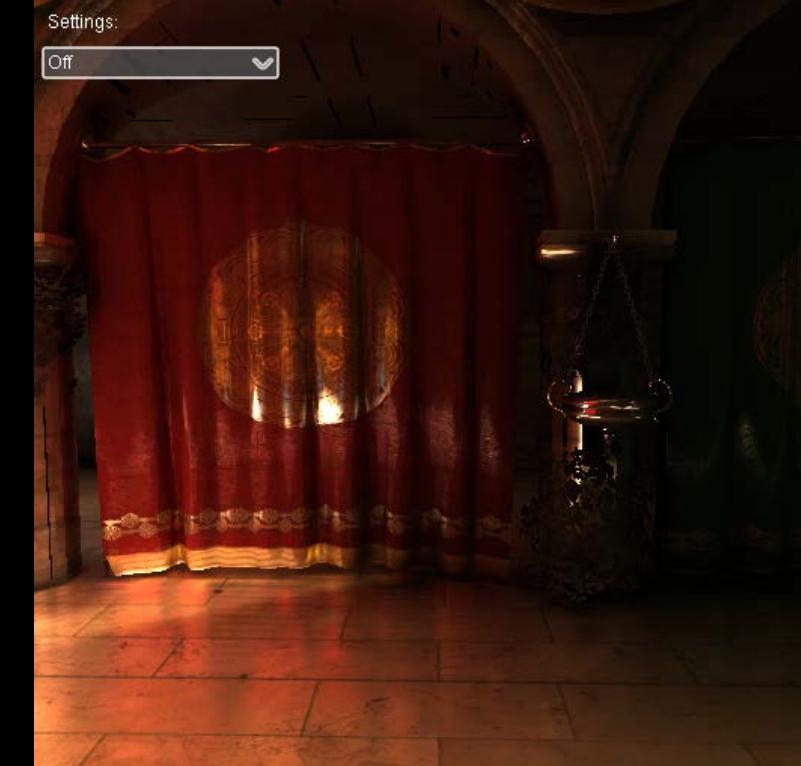
- Computer Graphics Forum  
(*Proc. of Pacific Graphics 2011*)
- <http://research.nvidia.com/publication/interactive-indirect-illumination-using-voxel-cone-tracing>

## ■ I3D 2011 Poster

- <http://maverick.inria.fr/Publications/2011/CNSGE11/>

## ■ Siggraph 2011 Talk

- <http://maverick.inria.fr/Publications/2011/CNSGE11a/>



**Interactive Indirect Illumination Using Voxel Cone Tracing : An Insight**  
Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, Elmar Eisemann  
INRIA / LJK / Grenoble University / CNRS      NVIDIA Corporation      NVIDIA Corporation  
Telecom ParisTech

**Overview**  
Indirect illumination is an important element for realistic image synthesis. In this paper we propose a novel approach to render indirect illumination in real-time using voxel-based cone tracing. This idea is to perform volumetric integration of the radiance field using a cone of finite radius centered at each pixel. The cone is subdivided into smaller cones that are used to evaluate the radiance field at the center of each subcone. This allows us to provide credit pre-computation maps and to limit memory access to the radiance field. We also propose a novel way to handle the light directions in the space of the view rays. All considered in this paper are based on the same principle: the cone is subdivided into smaller ones to compute ray or voxel-based cone tracing.

**Pre-Integrated Voxel Cone Tracing**  
The approach approximates the result of the radiance query and VBR using a pre-integrated voxel cone tracing. This method is based on the idea of using a cone of finite radius to evaluate the radiance field. The cone is subdivided into smaller cones that are used to evaluate the radiance field at the center of each subcone. This allows us to provide credit pre-computation maps and to limit memory access to the radiance field. We also propose a novel way to handle the light directions in the space of the view rays. All considered in this paper are based on the same principle: the cone is subdivided into smaller ones to compute ray or voxel-based cone tracing.

**Sparse Voxel Octree Structure**  
The core of our approach is to build a pre-integrated voxel octree structure that stores the radiance field. The structure is built at the time of a complex pre-trace step where the radiance field is subdivided into smaller volumes. These volumes are then subdivided into smaller ones until they reach a certain size. This allows us to provide credit pre-computation maps and to limit memory access to the radiance field. We also propose a novel way to handle the light directions in the space of the view rays. All considered in this paper are based on the same principle: the cone is subdivided into smaller ones to compute ray or voxel-based cone tracing.

**Anisotropic Pre-Integration**  
This approach approximates the result of the radiance query and VBR using a pre-integrated voxel cone tracing. This method is based on the idea of using a cone of finite radius to evaluate the radiance field. The cone is subdivided into smaller cones that are used to evaluate the radiance field at the center of each subcone. This allows us to provide credit pre-computation maps and to limit memory access to the radiance field. We also propose a novel way to handle the light directions in the space of the view rays. All considered in this paper are based on the same principle: the cone is subdivided into smaller ones to compute ray or voxel-based cone tracing.

**References**

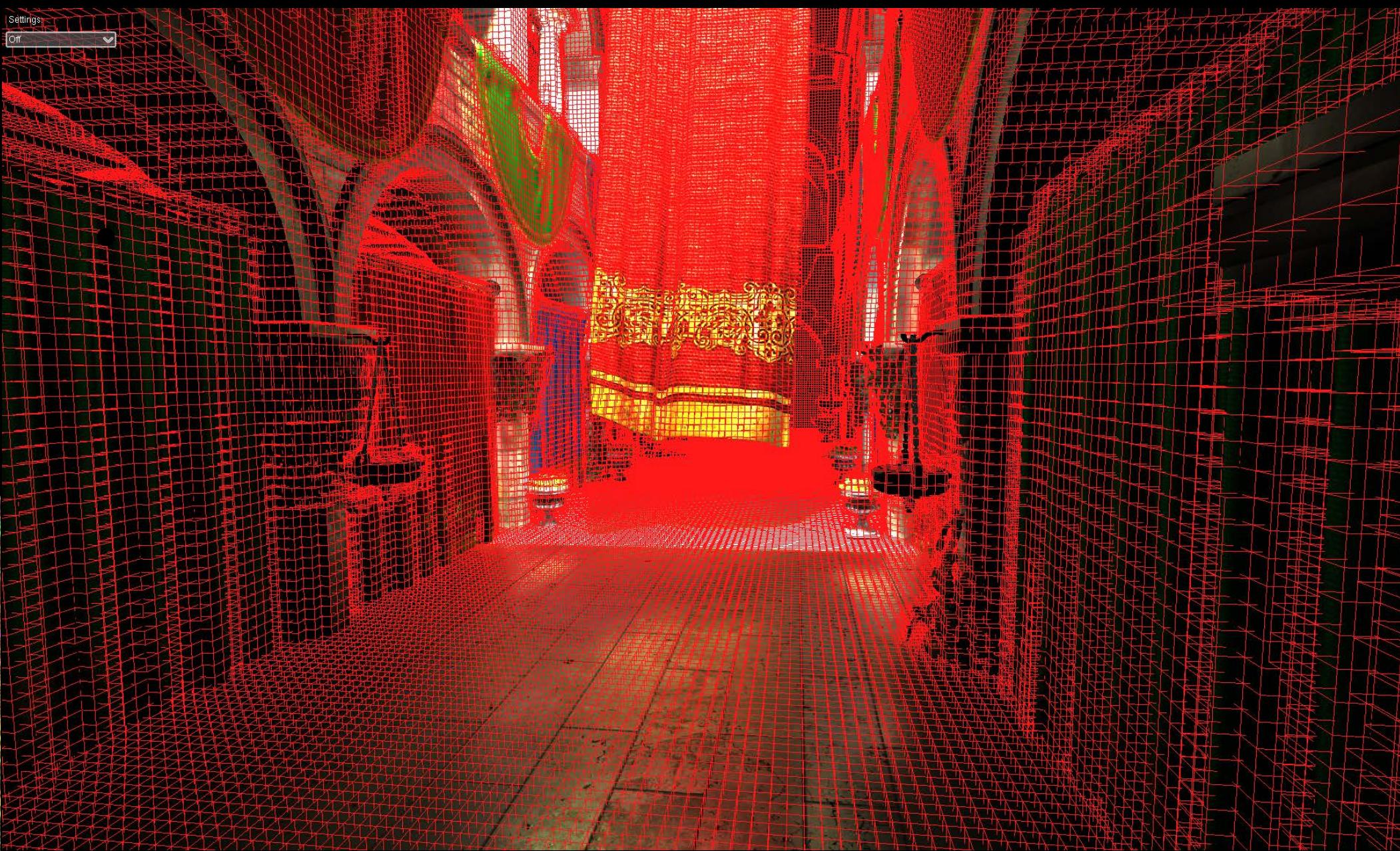
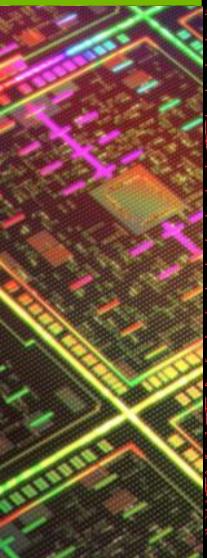
1. C. Crassin, F. Neyret, and M. Sainz. Interactive Indirect Illumination Using Voxel-Based Cone Tracing. In Proc. of Pacific Graphics 2011, pages 1–10, 2011. <http://research.nvidia.com/publication/interactive-indirect-illumination-using-voxel-cone-tracing>

2. Dynamic sparse voxel octree structures. <http://www.cs.cmu.edu/~davide/pubs/octree.pdf>

3. Interactive Indirect Illumination Using Voxel Cone Tracing : An Insight. <http://maverick.inria.fr/Publications/2011/CNSGE11/>

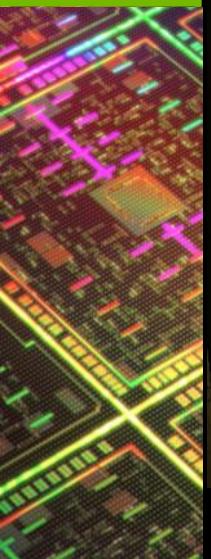
4. Anisotropic Pre-Integration. <http://maverick.inria.fr/Publications/2011/CNSGE11a/>

# GPU TECHNOLOGY CONFERENCE



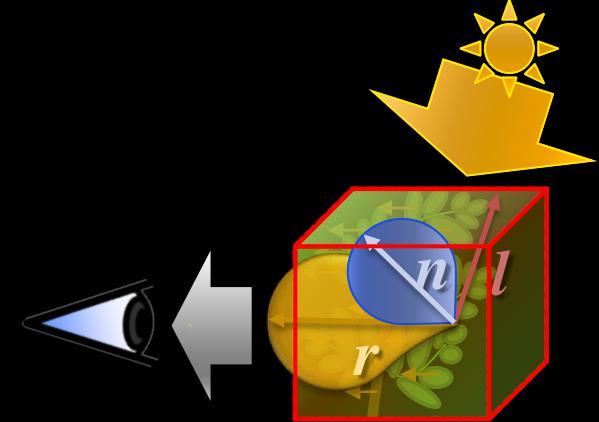
Settings  
 Off

# GPU TECHNOLOGY CONFERENCE

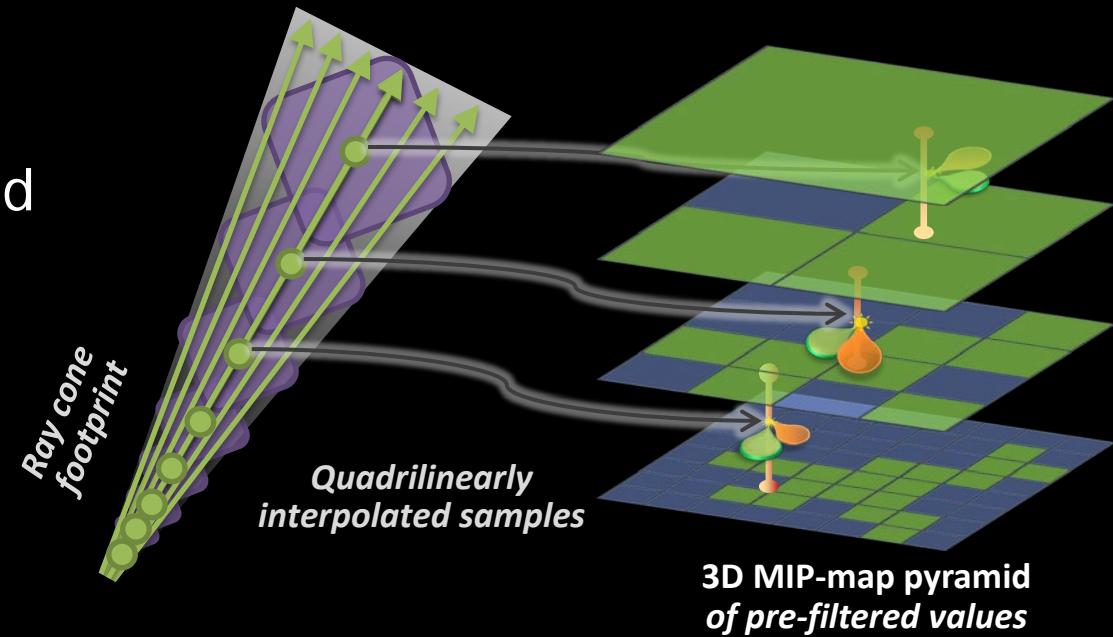


# Voxel cone tracing

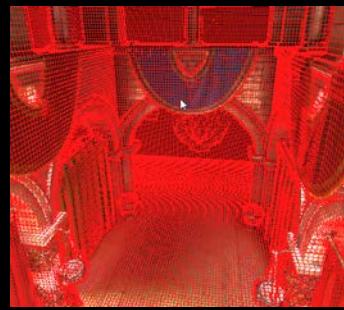
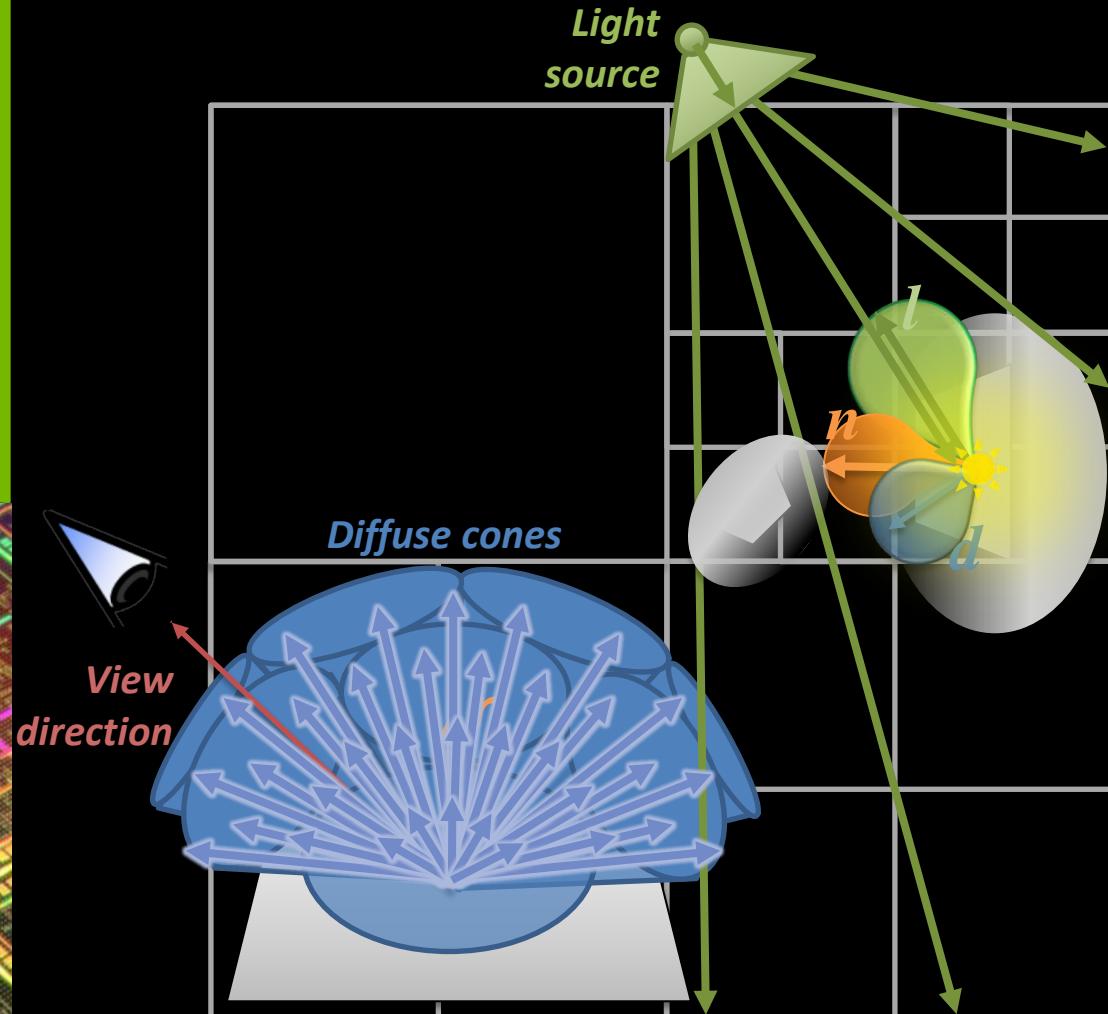
- Geometry pre-filtering
  - Traced like a participating media
  - Volume ray-casting



- Voxel representation
  - Scene geometry : Opacity field
  - + Incoming radiance



# Rendering algorithm



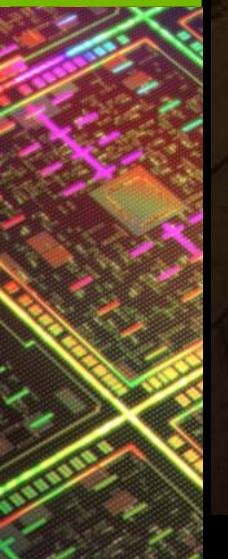
1. Light pass (es)
  - *Bake irradiance (RSM)*
2. Filtering pass
  - *Down-sample radiance in the octree*
3. Camera pass
  - *For each visible fragment:*  
*Gather indirect radiance*



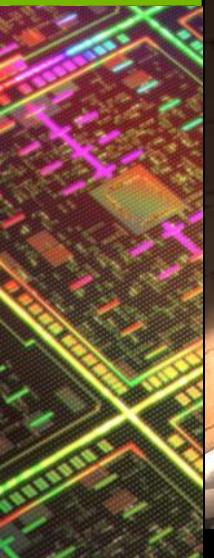
# Ambient Occlusion

Scene model courtesy of Guillermo M. Leal Llaguno

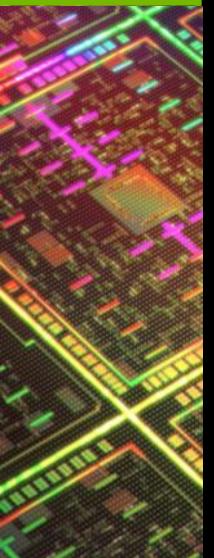
# Indirect diffuse



# Indirect diffuse



# GPU TECHNOLOGY CONFERENCE

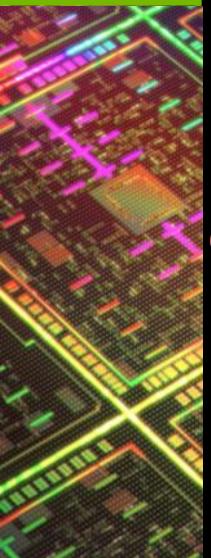
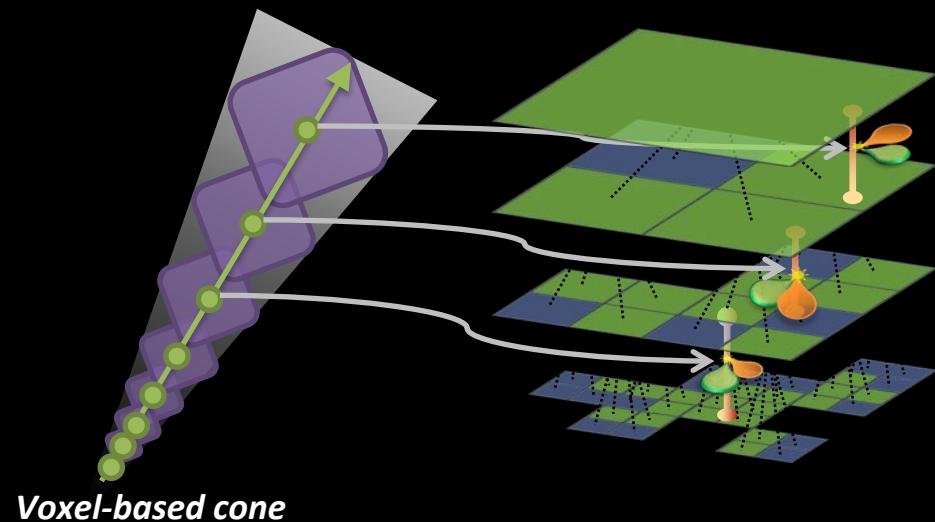
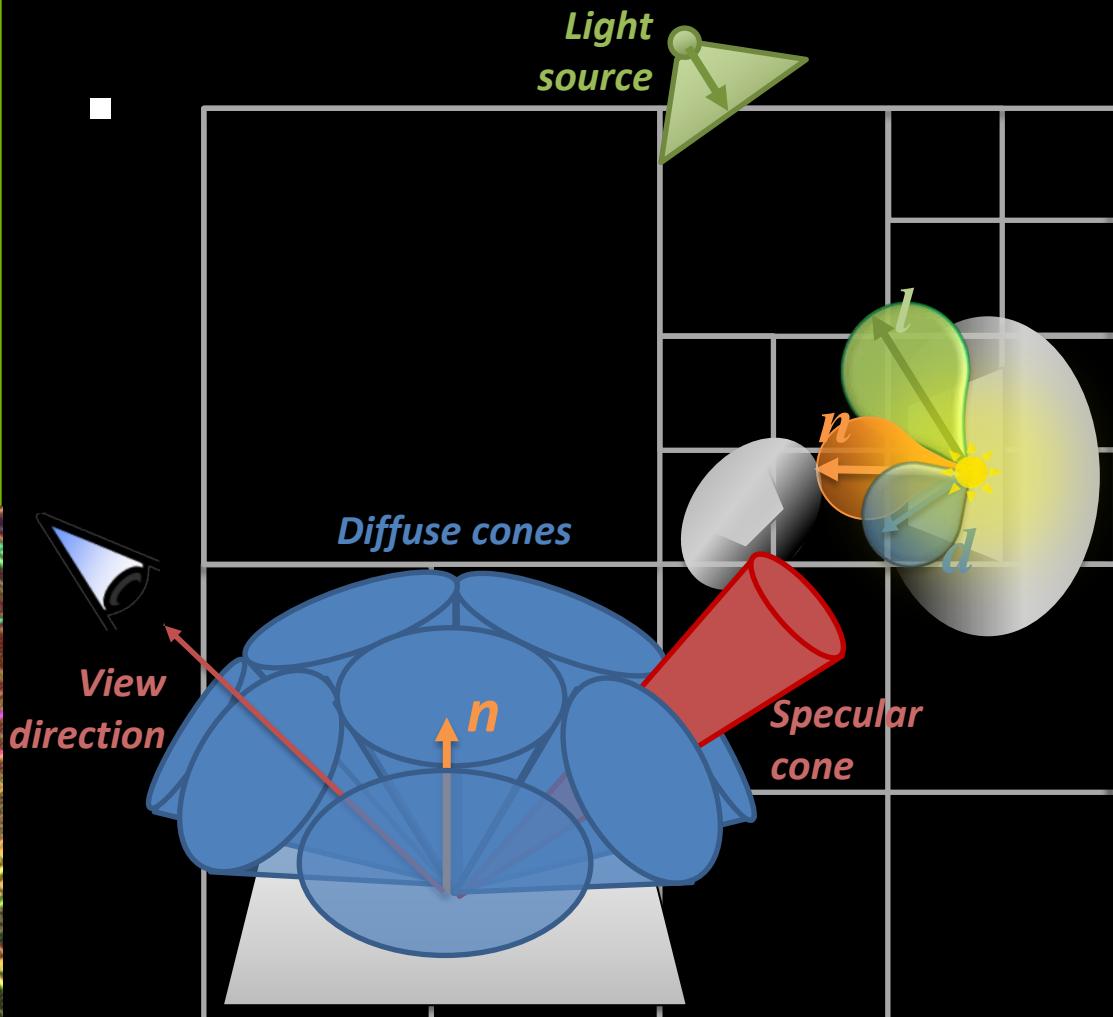


Settings:

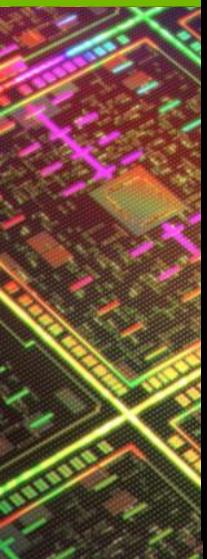
Off ▾



# Specular tracing

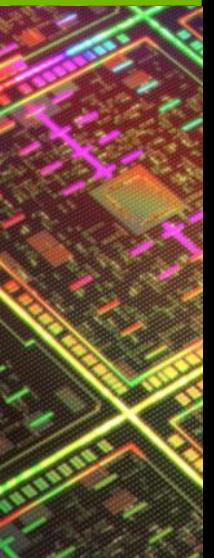


# Indirect specular

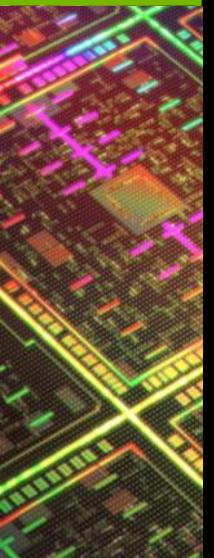


Indirect specular



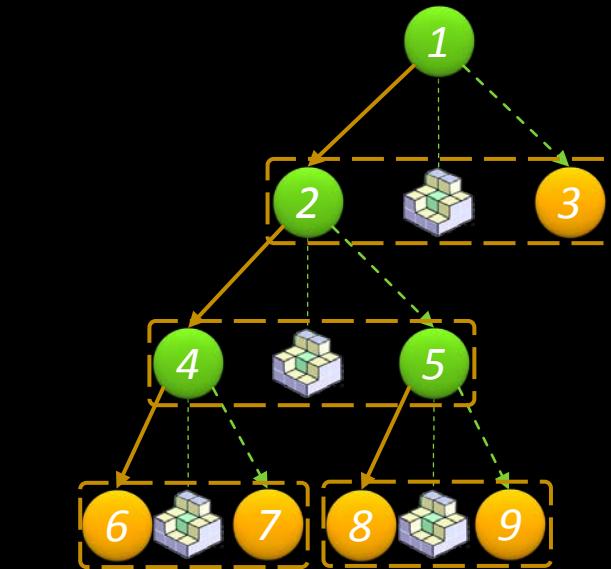
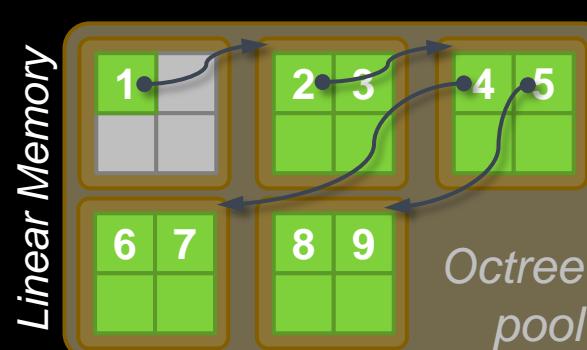


# GPU TECHNOLOGY CONFERENCE



# GPU Voxel Octree

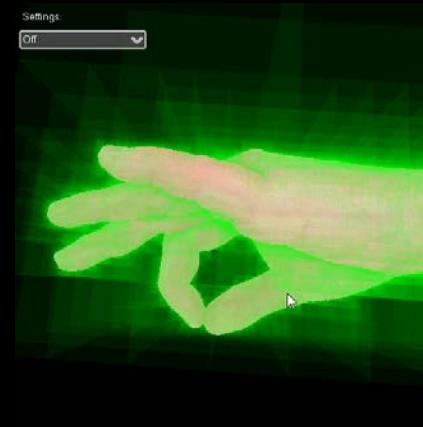
- Linked nodes in linear video memory (*Octree Pool*)
  - 2x2x2 nodes tiles
  - 1 pointer per node to a node-tile
- Voxels stored into a 3D texture (*Brick Pool*)
  - Allows hardware tri-linear interpolation

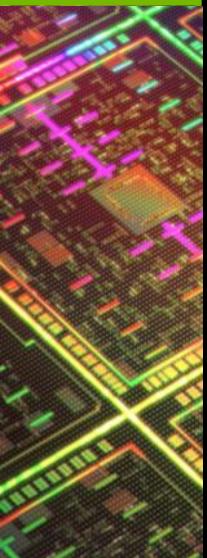


# Dynamic Voxelization

- Entirely done using the GPU graphics pipeline
  - Sparse (No plain grid allocation)

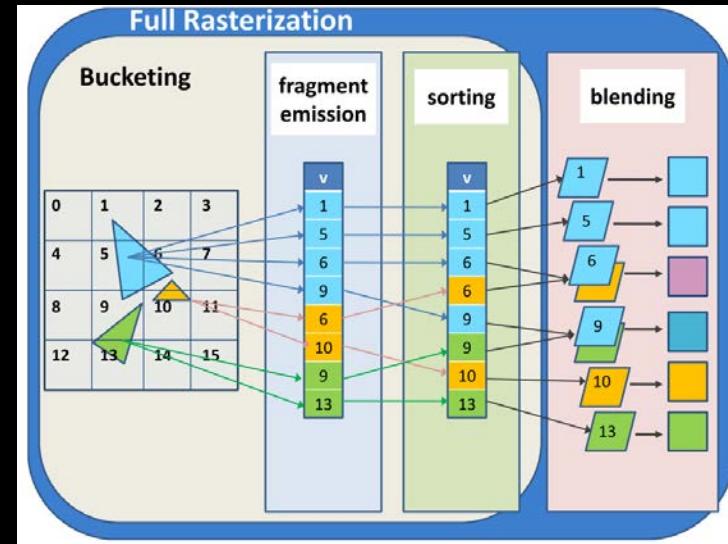
- Two modes :
  - Static environment
    - Pre-voxelized (~20ms)
  - Dynamic objects
    - Added to the structure at runtime (~4-5ms)



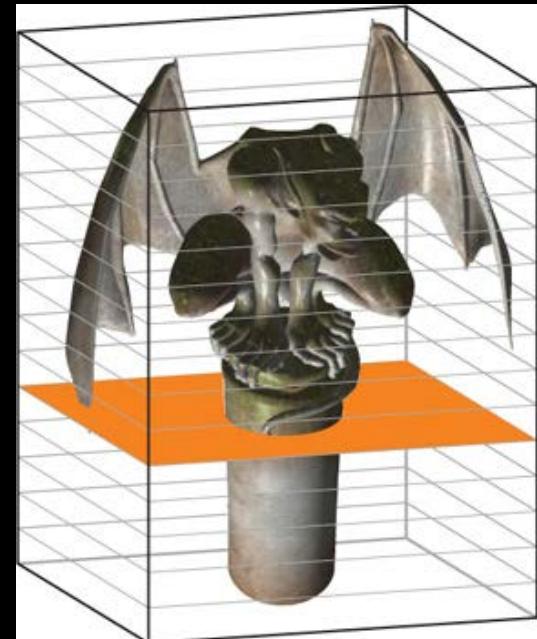


# Previous GPU approaches

- Compute-based  
[Schwarz and Seidel 10, Pantaleoni 11]
  - Not using hw rasterizer
- Multi-pass graphics-based
  - Slice-by-slice  
[Fang et al. 00, Crane et al. 07, Li et al. 05]
  - Multiple-slices through MRT  
[Dong et al. 04, Zhang et al. 07, Eisemann and Decoret 08]



VoxelPipe [Pantaleoni 11]



Crane et al. 2007

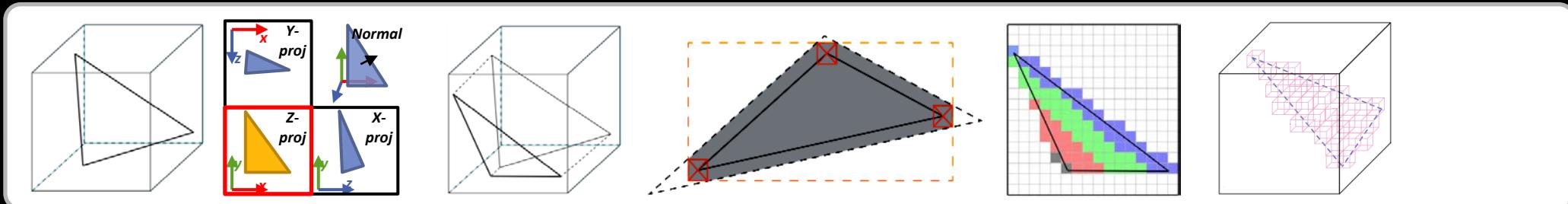
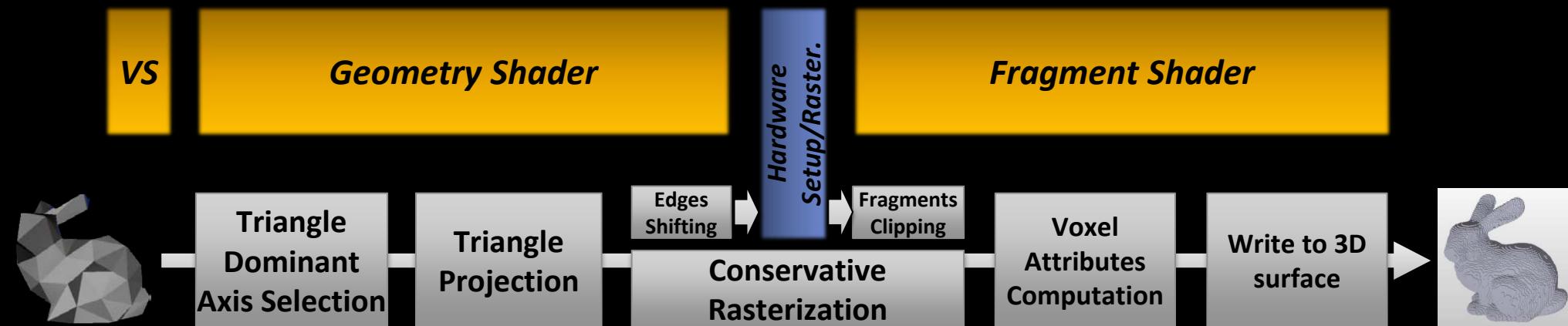
# OpenGL 4.2 Image Load/Store

- Random read/write access to textures
  - Shaders with side effects !
    - Shader Model 5 hw (*NVIDIA Fermi / Kepler*)
    - Similar to DX11 UAV
  - Uniform layout(`rgba32f`) **image3D** `voxData`;
    - **imageStore**(`voxData`, `ivec3(coords)`, `val`);
- NVIDIA *Bindless Graphics*
  - *Pointers to global memory* : `NV_shader_buffer_load/store`
  - Uniform `vec4 *voxData`;



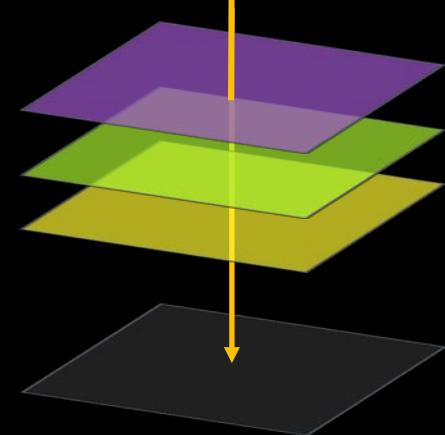
# One pass voxelization pipeline

- *Thin surface voxelization*



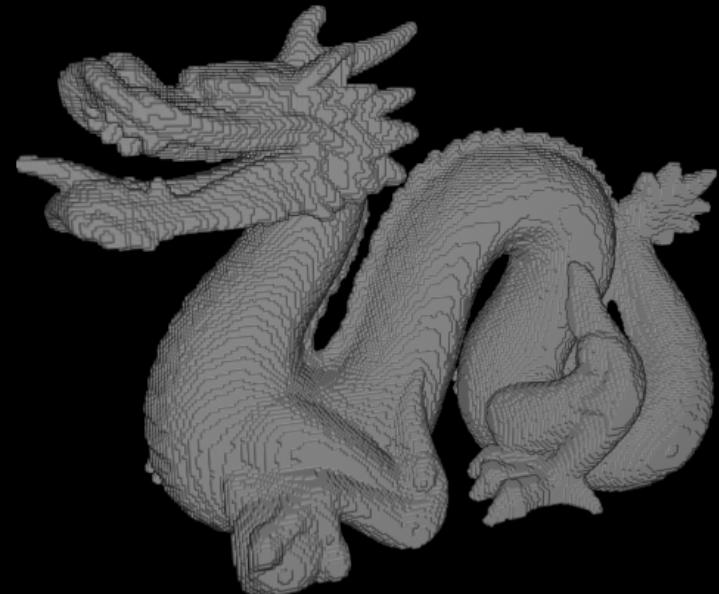
# Compositing voxel fragments

- To texture or linear buffer (global memory)
- Native AtomicAdd
  - INT32
  - INT64 (*NVIDIA Only, global memory*)
  - FP32 (*NVIDIA Only, NV\_shader\_atomic\_float extension*)
- Emulation for any format (*RGBA8, RGBA16F...*)
  - **AtomicCompSwap / AtomicCompSwap64**
    - (*2x-3x speed penalty*)
  - Moving average (*RGBA8*) + *Voxel Anti-Aliasing (coverage mask)*



# Results

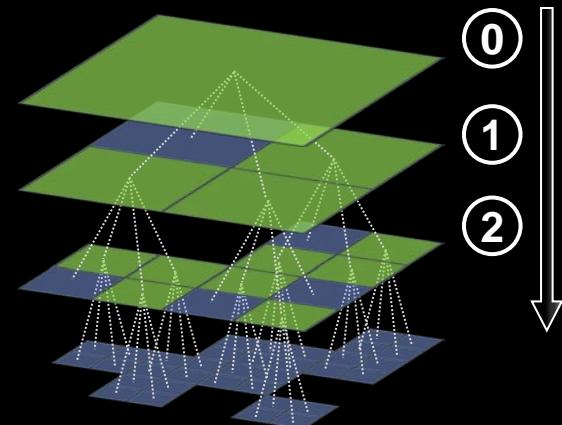
- Stanford Dragon
  - GTX 480 (GF100)
- Usually as good as, or even faster than Voxel Pipe [*Pantaleoni 11*]



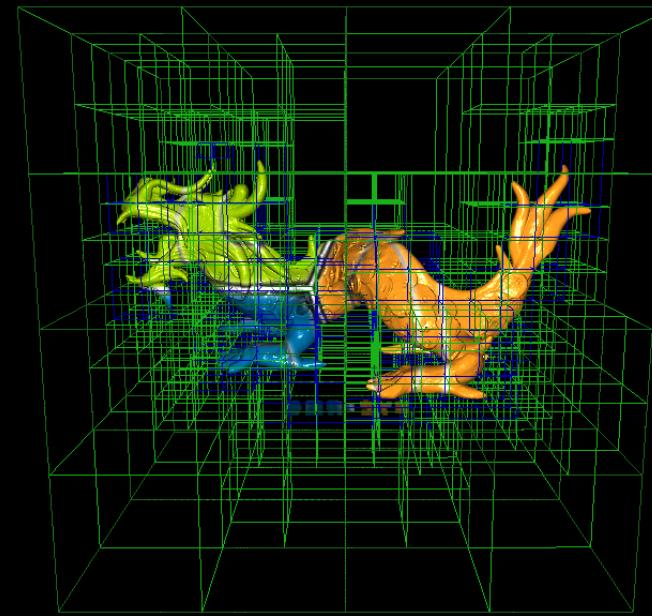
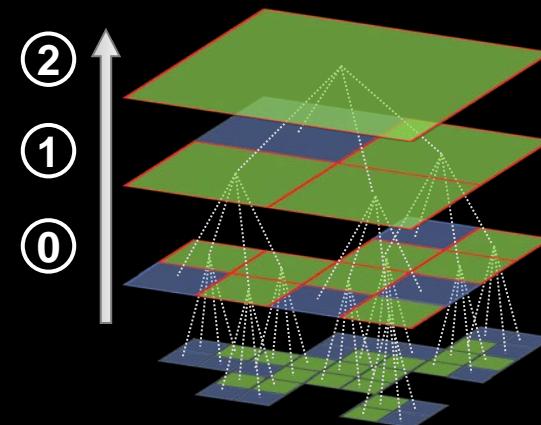
Times in ms		Std. raster.		Cons. raster.	
Format	Res	Write	Merge	Write	Merge
R32F	128	1.19	1.24	1.63	2.41
	512	1.38	2.73	1.99	5.30

# Sparse Octree construction

- Sparse voxelization
  - No plain grid allocation
- Two steps:
  1. Octree subdivision

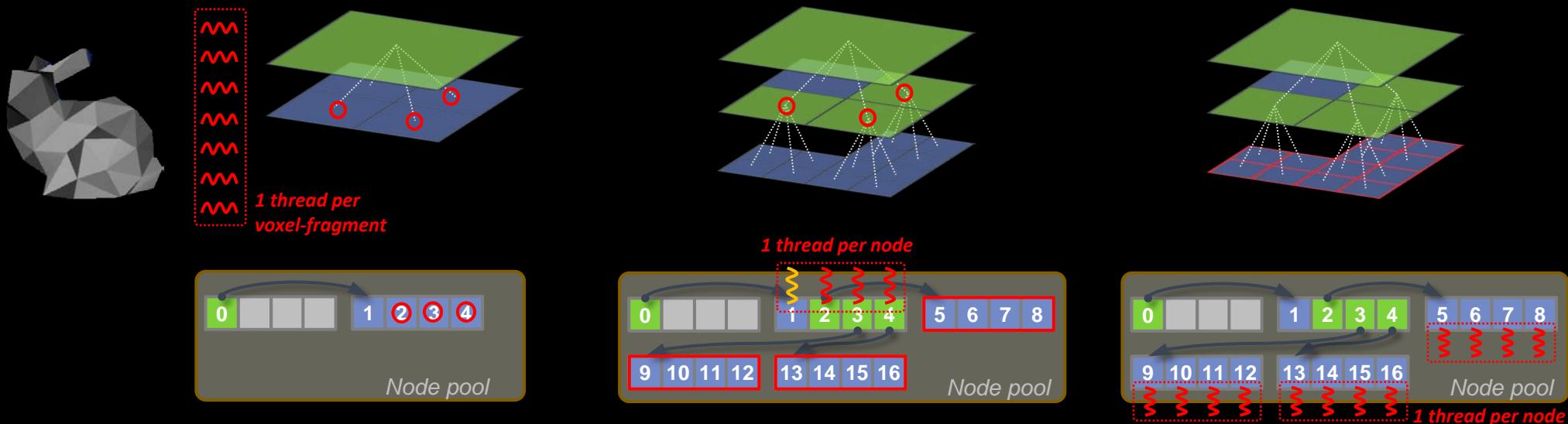
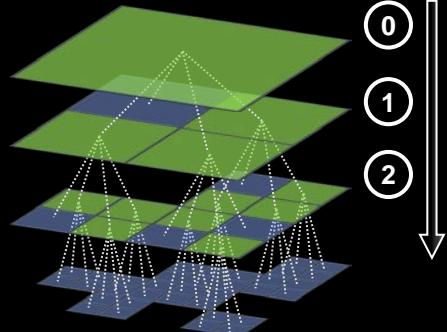


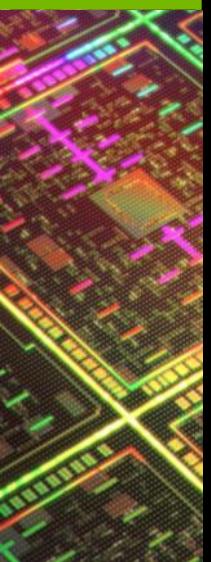
2. Values MIP-mapping



# Octree construction (1/2)

- Step 1 : Top-down construction
  - For each level from the root:





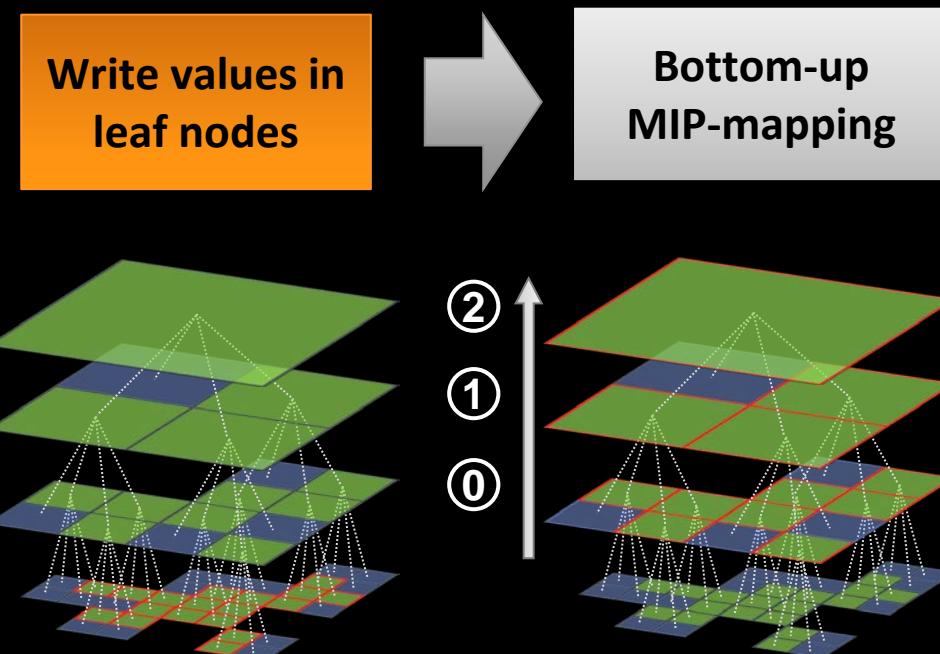
# OpenGL compute kernel emulation

- Emulated using a vertex shader
  - `gl_VertexID` == ThreadID
  - No input attribute
- Synchronization-free: Indirect draw calls
  - `glDrawArraysIndirect()`
  - Parameters read in video memory
    - No CPU read-back
    - Memory barriers: `glMemoryBarrier()`
  - Batching all construction steps

# Octree construction (2/2)

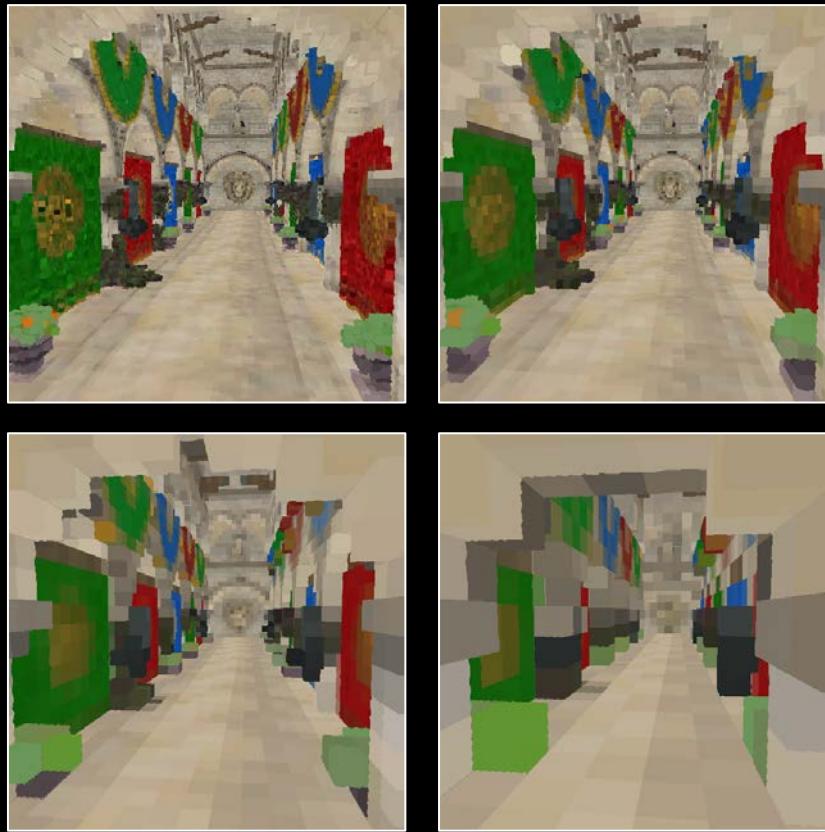
- Step 2: Populating octree with values

1. Voxelize mesh into leaf nodes
  - Average all incoming values per voxel
2. MIP-map values into interior nodes



# Results

- 9 levels octree ( $512^3$ )
  - RGBA32F
- Kepler GK104 performance
  - 30% - 58% faster than Fermi GF100
  - Atomic merging up to 80% faster.



Times in ms	Frag list	Octree construction				Write	MIP map	Total
		Flag	Create	Init	Total			
<b>Hand</b>	0.17	0.89	0.18	0.35	1.42	0.9	0.55	3.04
<b>Sponza</b>	2.07	5.65	0.37	1.32	7.34	3.94	2.09	15.44

# OpenGL Insights

- *Octree-Based Sparse Voxelization Using The GPU Hardware Rasterizer*

Cyril Crassin and Simon Green

- To be released for Siggraph 2012

Patrick Cozzi & Christophe Riccio



# GPU TECHNOLOGY CONFERENCE

Thank you !

# Talk

- S0610 - Octree-Based Sparse Voxelization for Real-Time Global Illumination

Cyril Crassin (NVIDIA)

Discrete voxel representations are generating growing interest in a wide range of applications in computational sciences and particularly in computer graphics. A new real-time usage of dynamic voxelization inside a sparse voxel octree is to compute voxel-based global illumination. When used in real-time contexts, it becomes critical to achieve fast 3D scan conversion (also called voxelization) of traditional triangle-based surface representations. This talk describes an new surface voxelization algorithm that produces a sparse voxel representation of a triangle mesh scene in the form of an octree structure using the GPU hardware rasterizer. In order to scale to very large scenes, our approach avoids relying on an intermediate full regular grid to build the structure and constructs the octree directly.

Topic Areas: Computer Graphics

Level: Intermediate

Day: Tuesday, 05/15

Time: 2:30 pm - 2:55 pm

Location: Room B