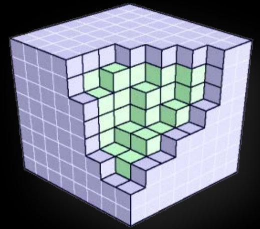
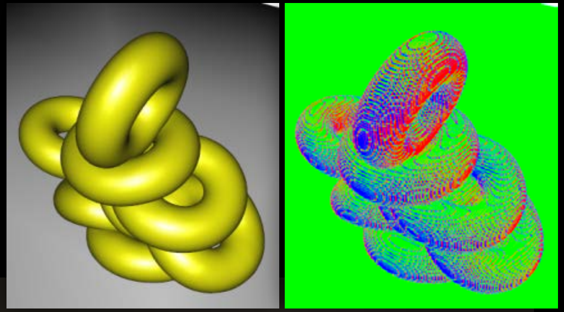
GPU TECHNOLOGY CONFERENCE

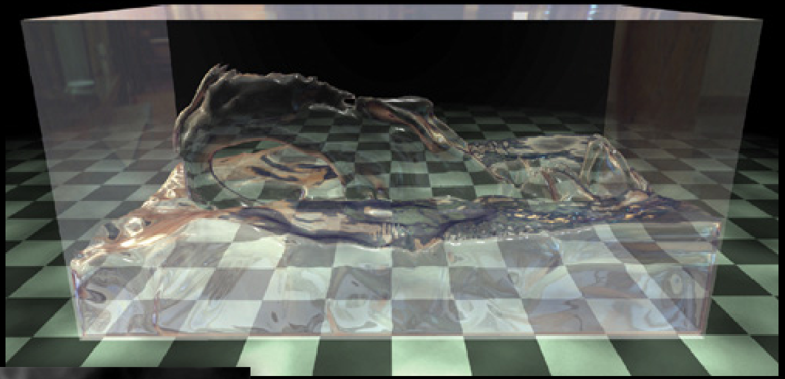
Octree-Based Sparse Voxelization for Real-Time Global Illumination

Cyril Crassin
NVIDIA Research

Voxel representations



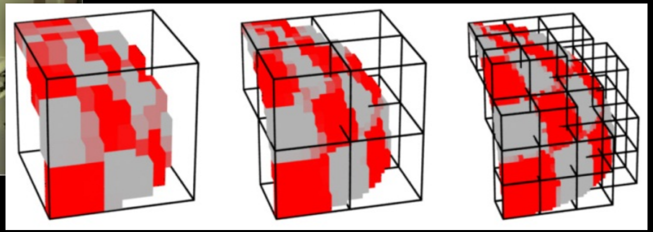




Crane et al. (NVIDIA) 2007

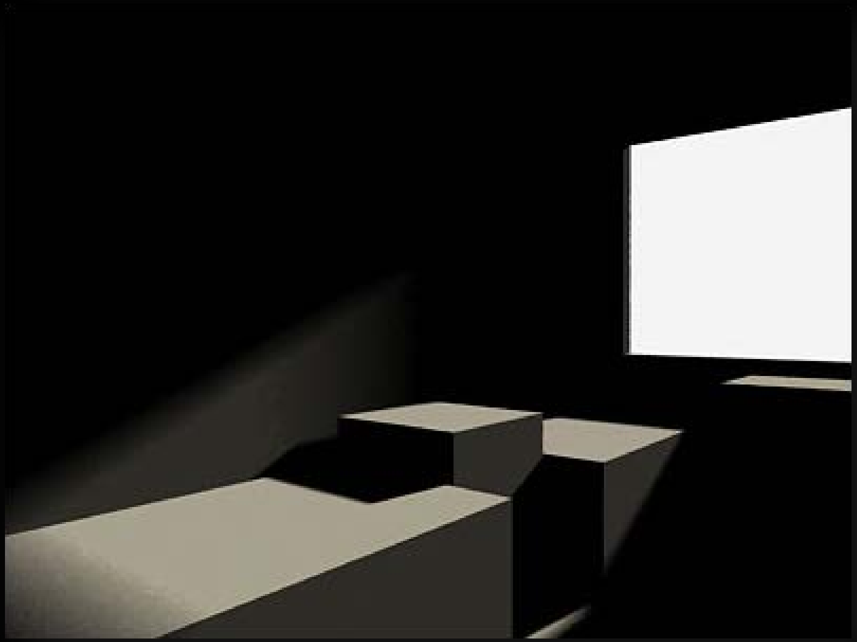
Allard et al. 2010

Christensen and Batali (Pixar) 2004

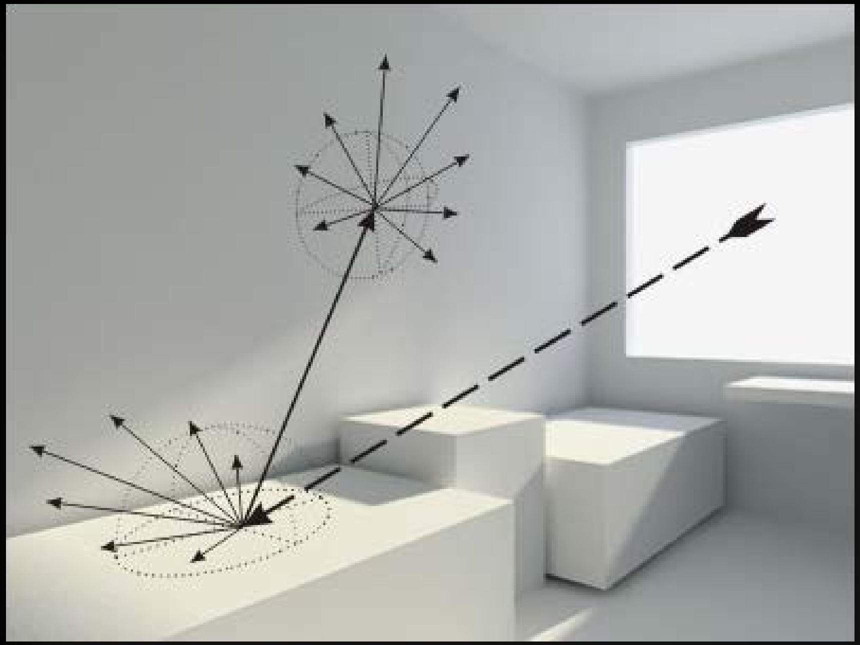


Global Illumination

- Indirect effects
- Important for realistic image synthesis



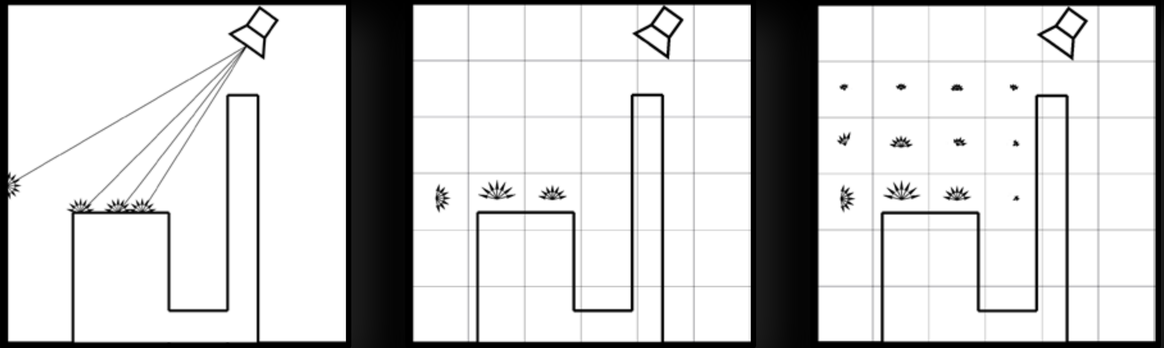
Direct lighting

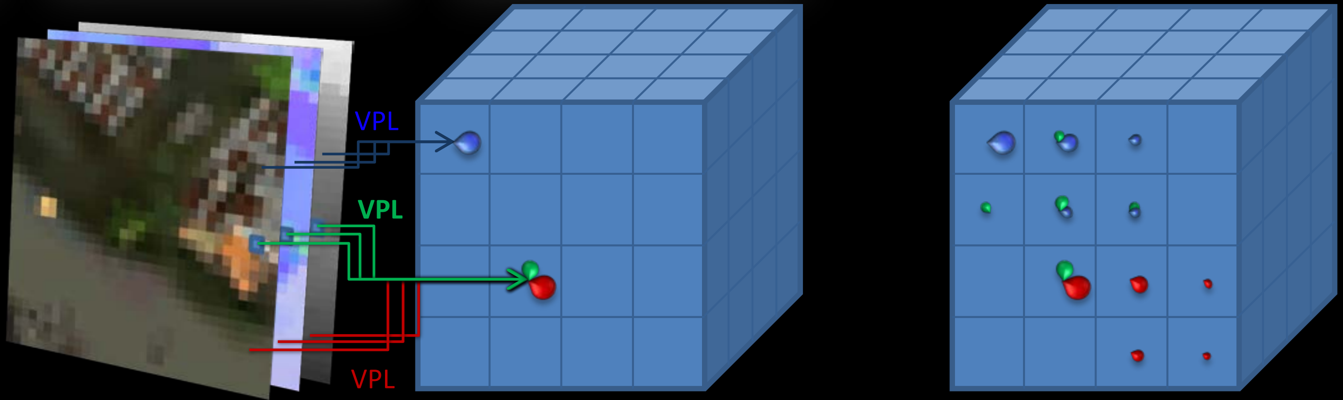


Direct+Indirect lighting

Light Propagation Volumes

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



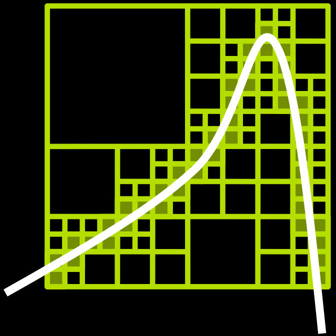


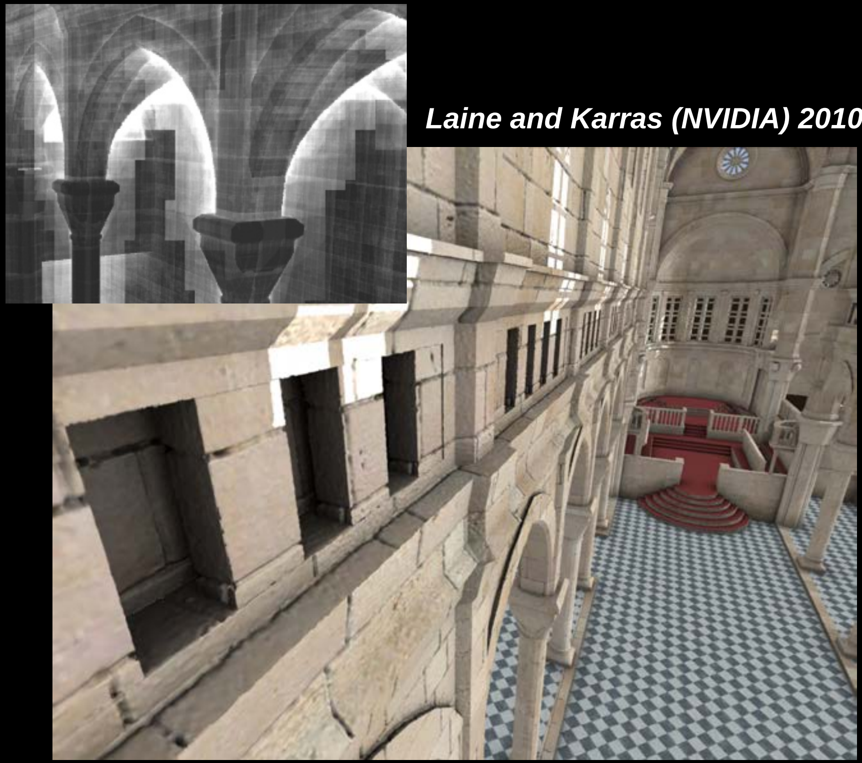
Reflective shadow maps Radiance volume gathering Iterative propagation

Sparse Voxel Octree

- Detailed geometry rendering
  - Structured LODs

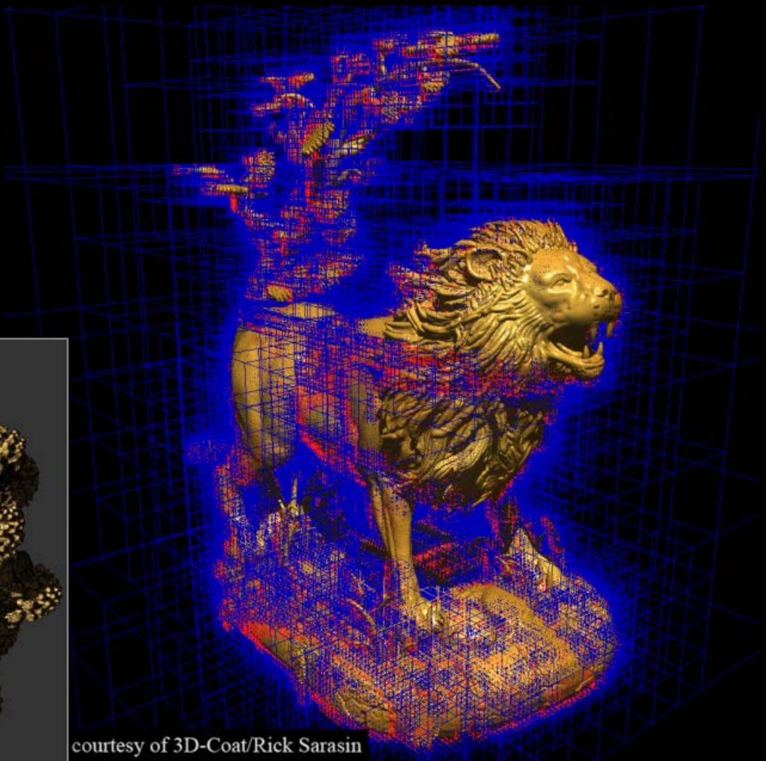
Olick. 2008





Laine and Karras (NVIDIA) 2010

Crassin et al. 2009 (GigaVoxels)



Interactive indirect illumination using voxel cone tracing

120 FPS @ 512x512 -- 16 FPS @ FullHD

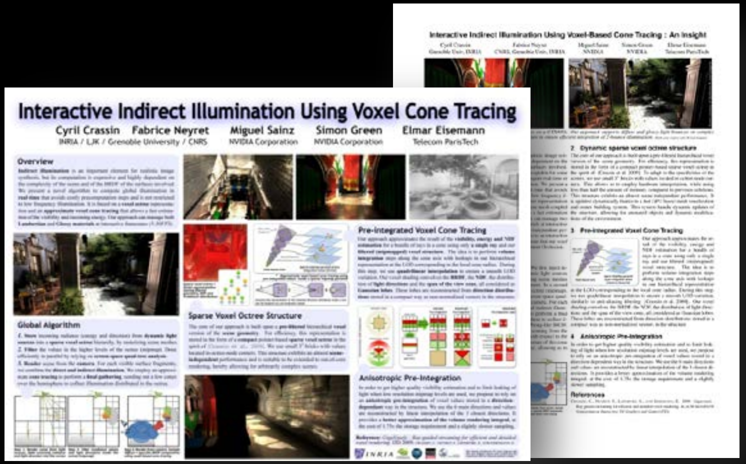
Publications

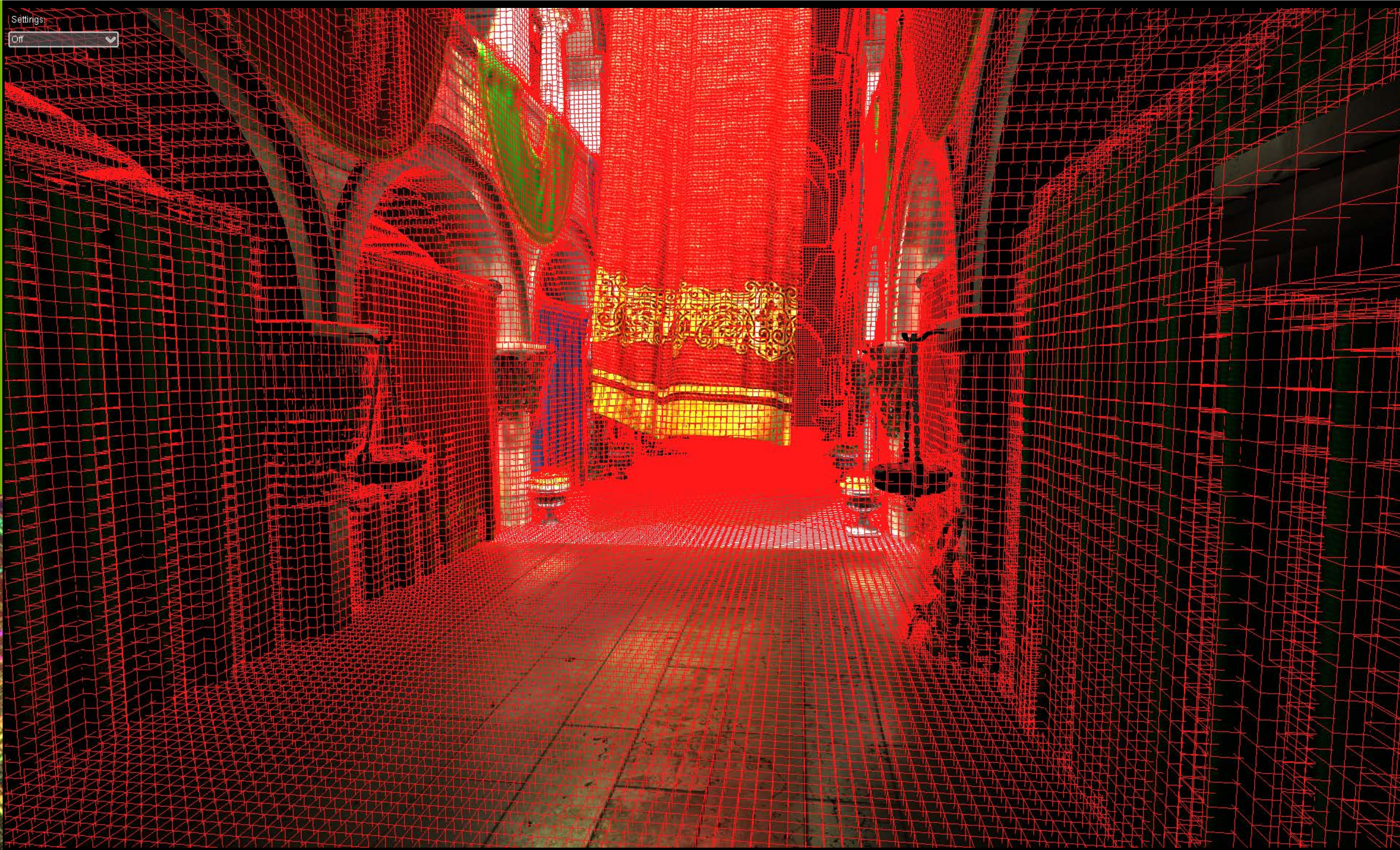
Interactive indirect illumination using voxel cone tracing

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/*




Settings
Off

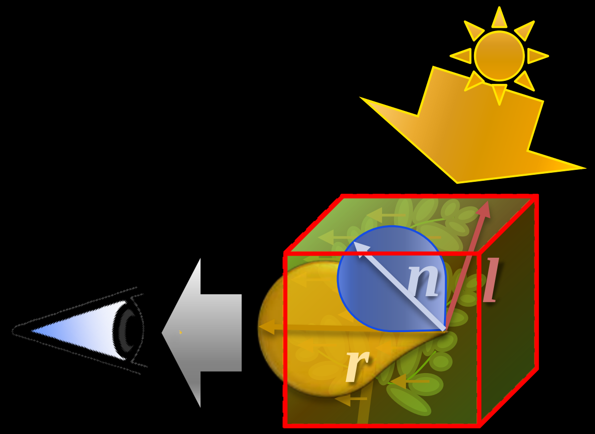

Settings:
Off

Voxel cone tracing

- Geometry pre-filtering

  Traced like a <u>participating media</u>

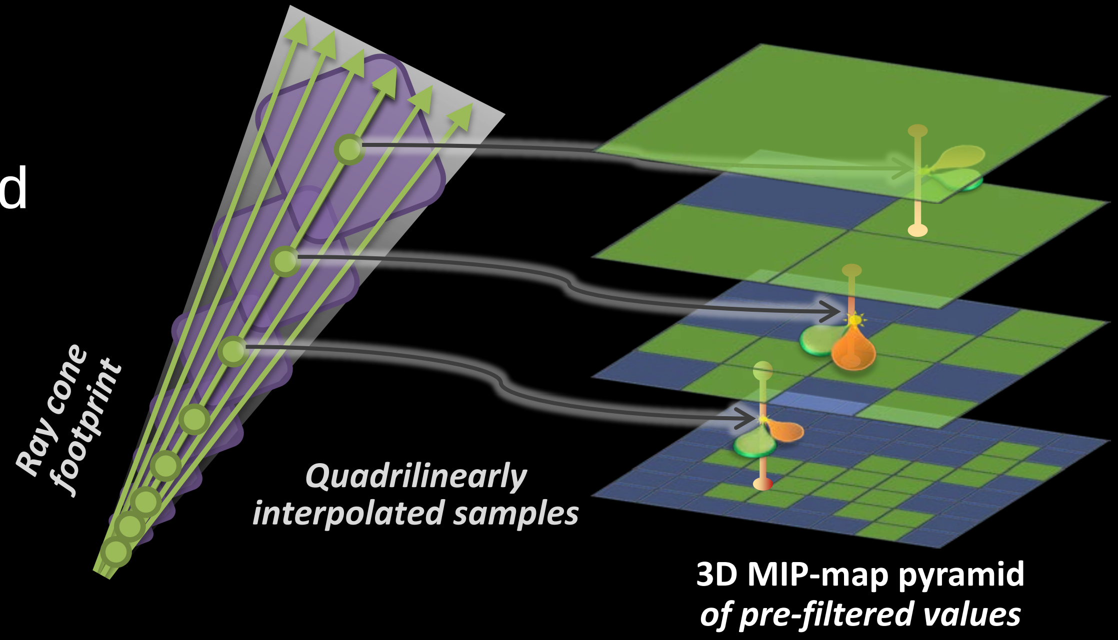
  – Volume ray-casting



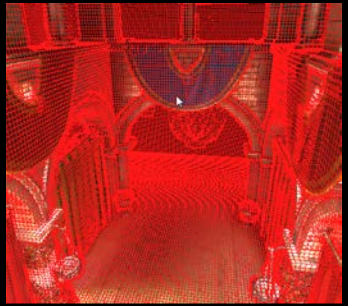
- Voxel representation

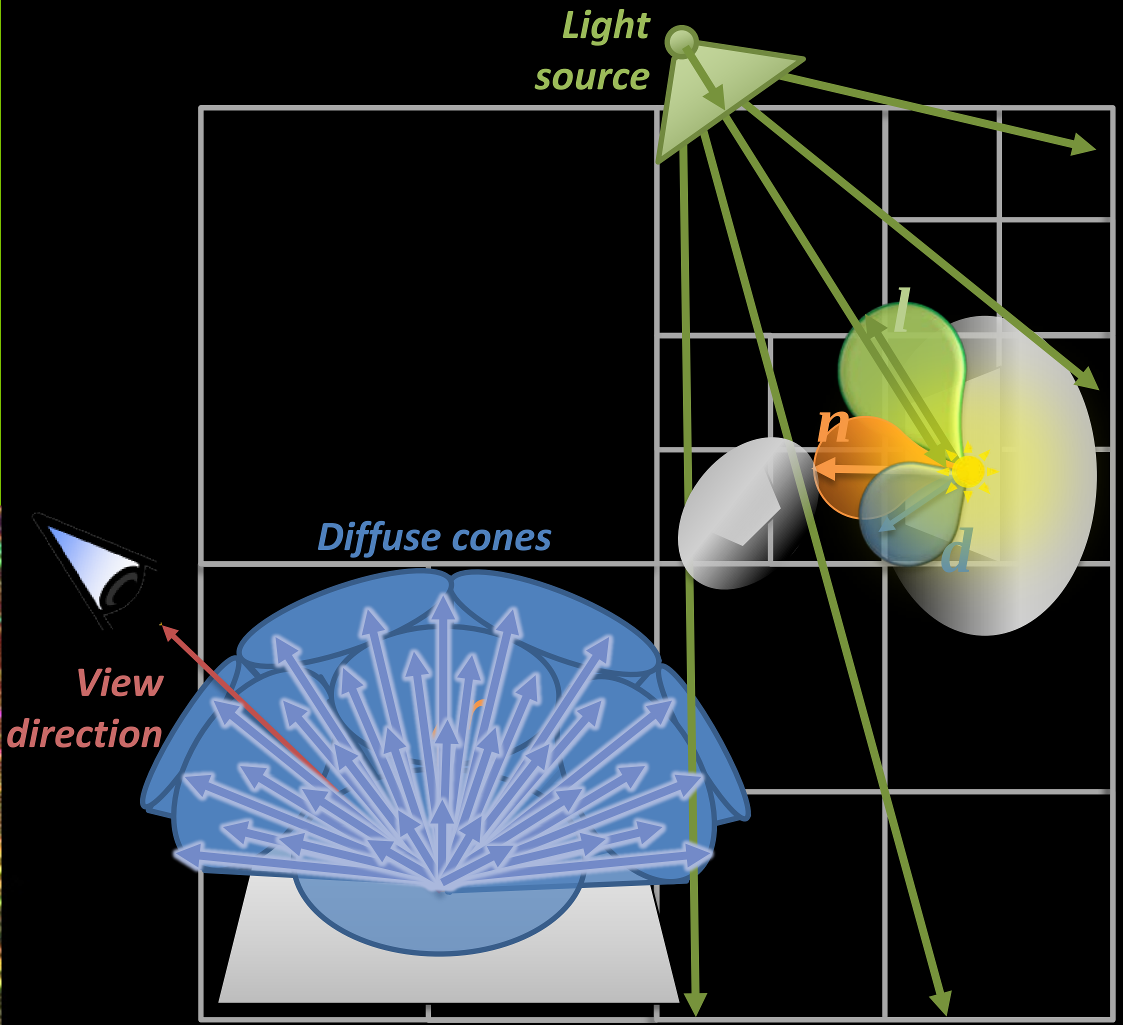
  Scene geometry : Opacity field

  + 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:*

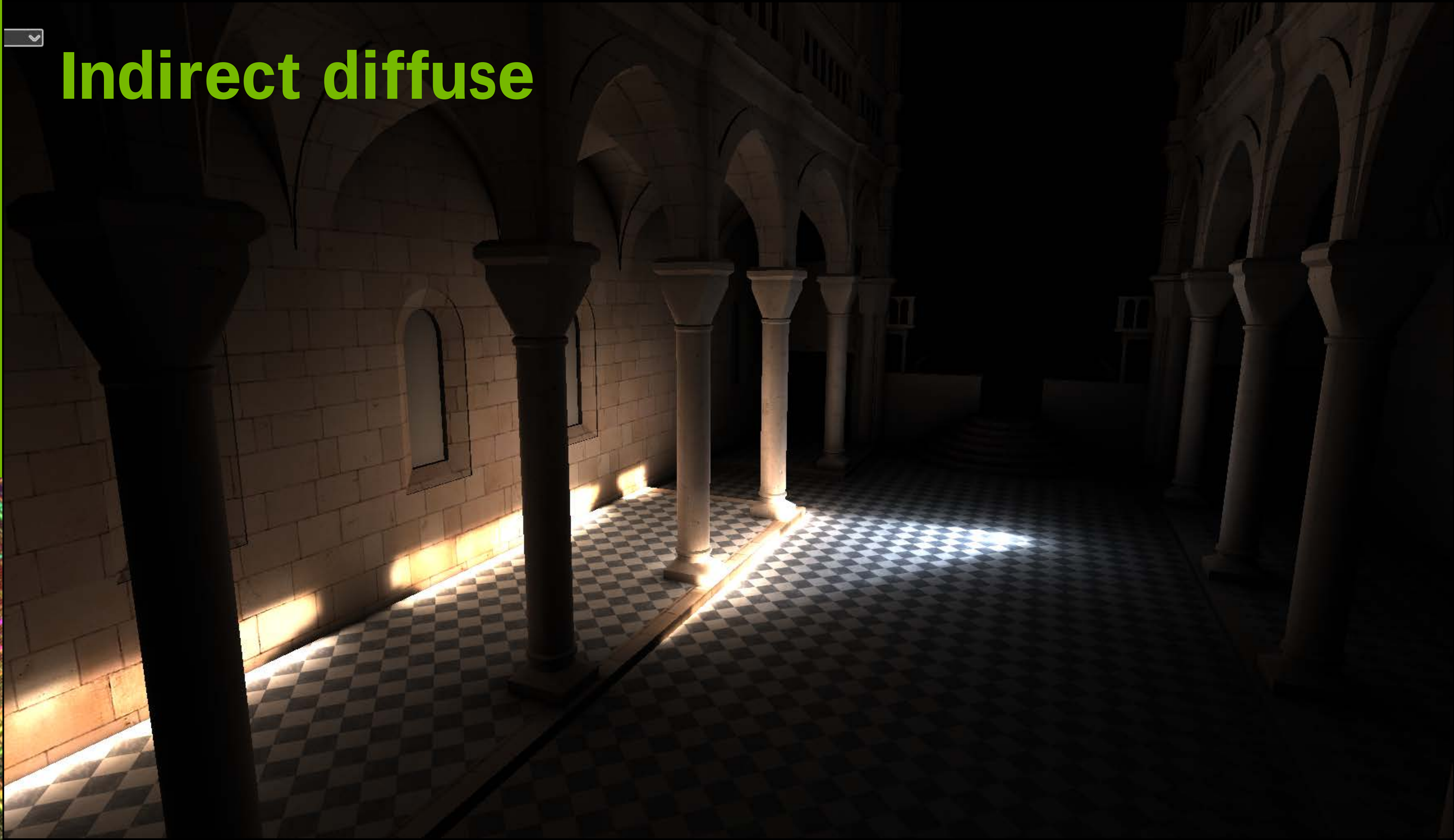
   Gather indirect radiance

Ambient Occlusion

Scene model courtesy of Guillermo M. Leal Llaguno

Indirect diffuse

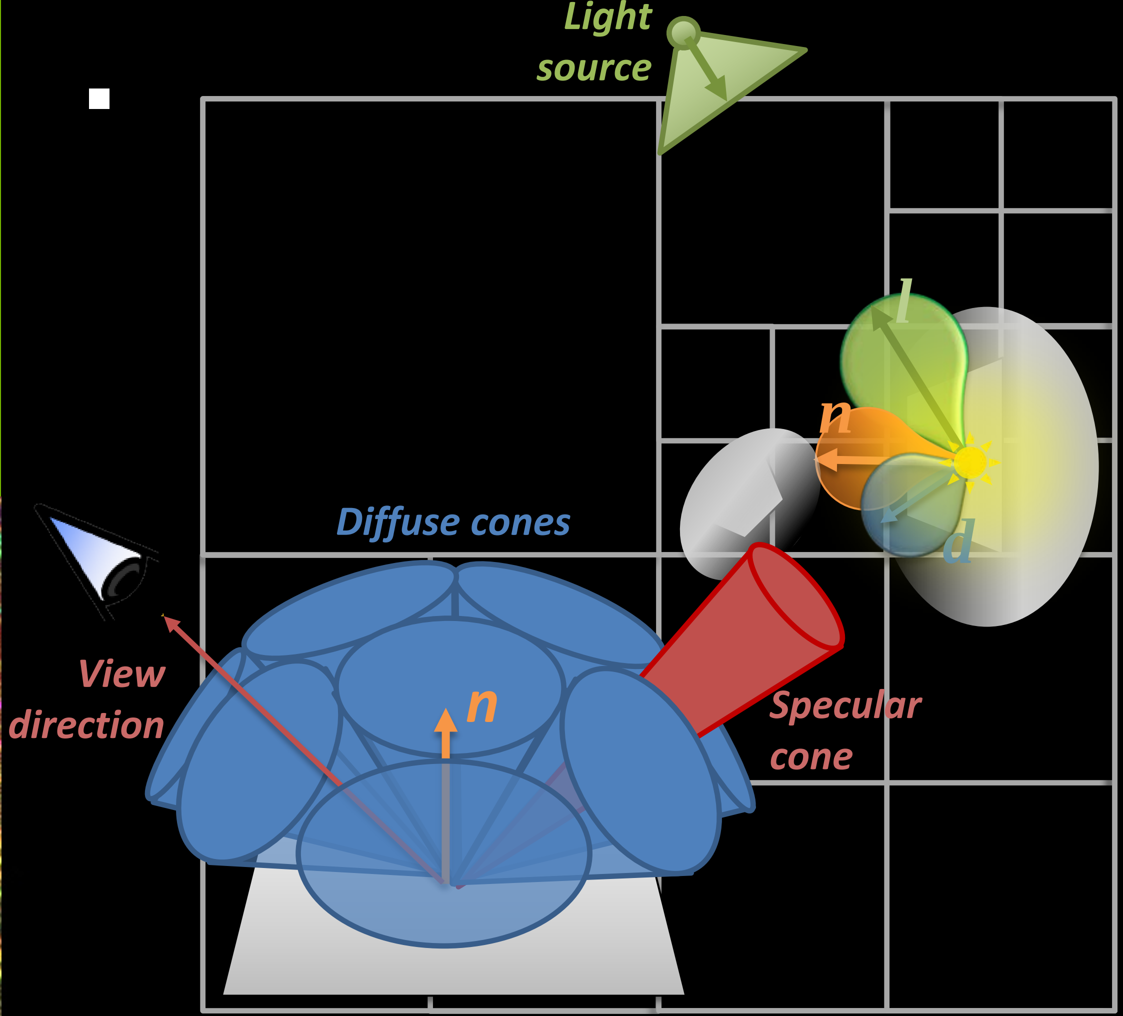
Indirect diffuse

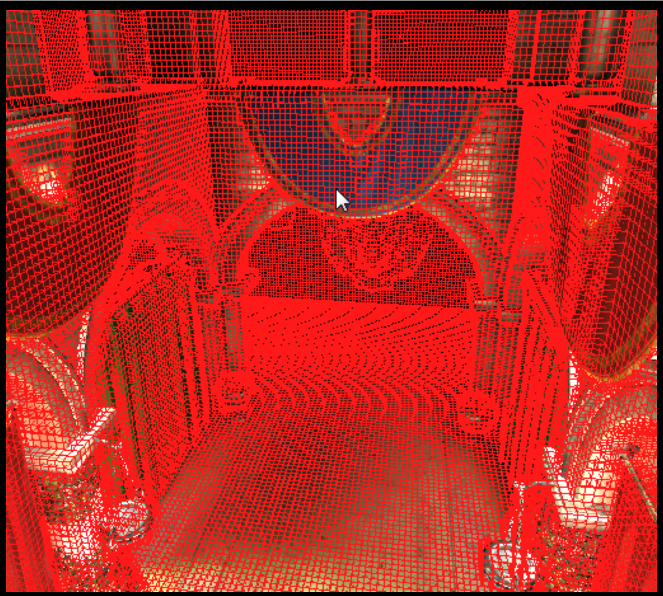


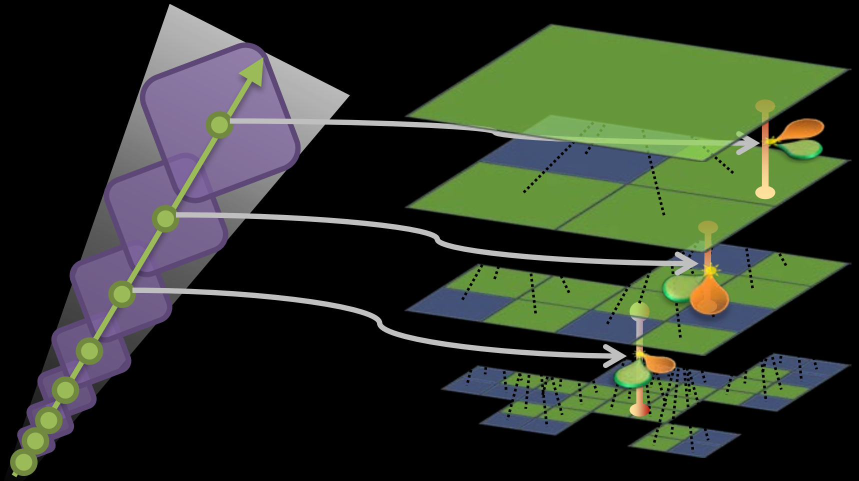

Settings:
Off

Specular tracing

▪



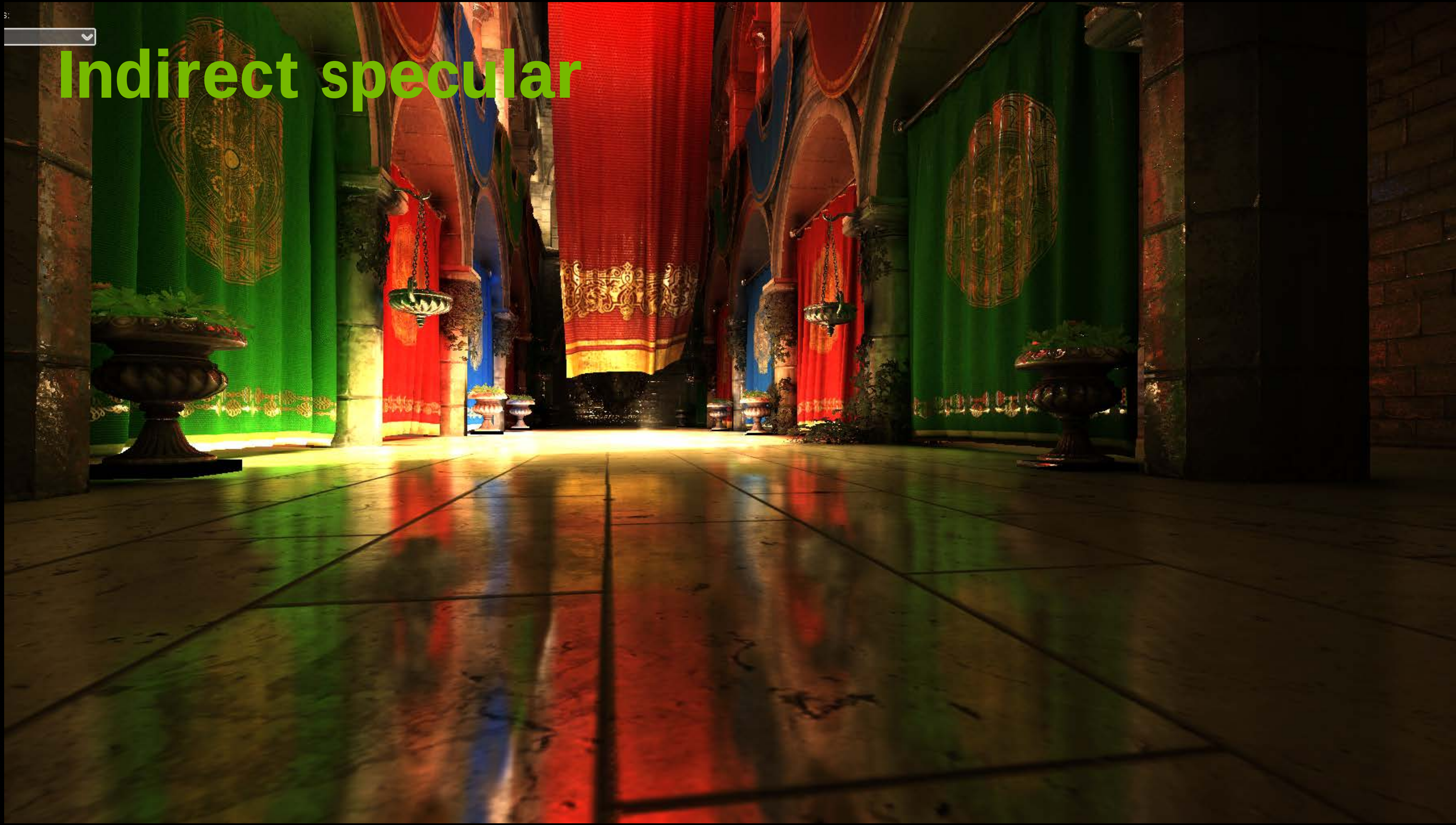




Voxel-based cone

Indirect specular

Indirect specular

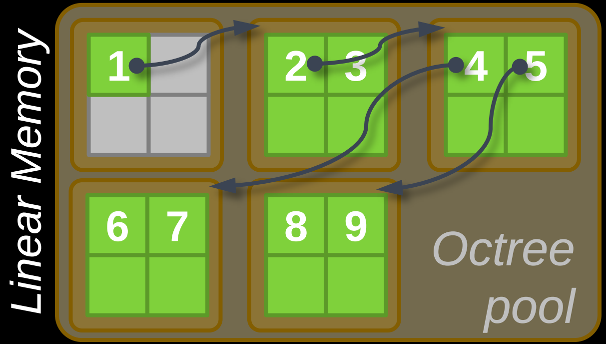


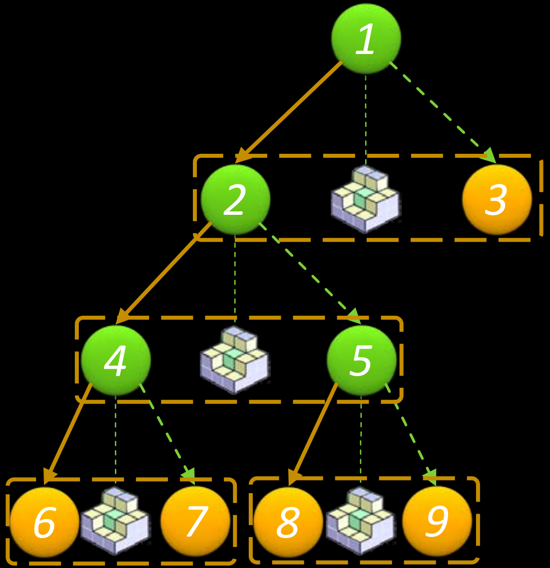

Settings:
Indirect specular config
Cone half-angle: 0.000
Radiance multiplier: 1.136

GPU TECHNOLOGY CONFERENCE

GPU Voxel Octree

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

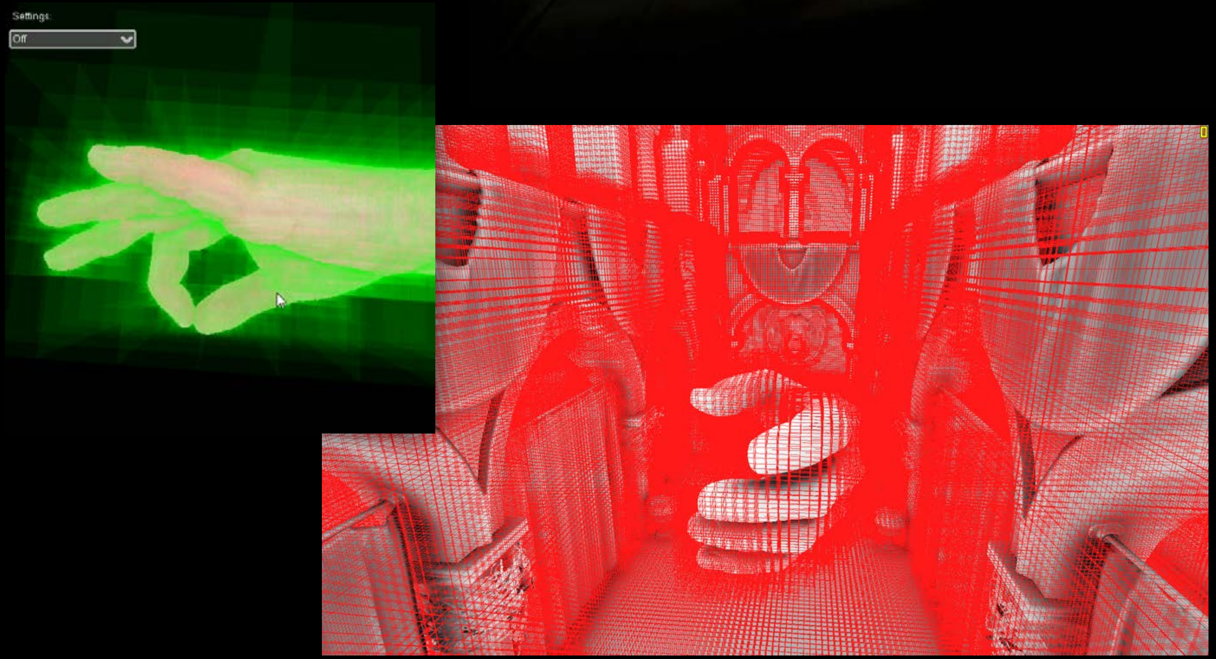




Dynamic Voxelization

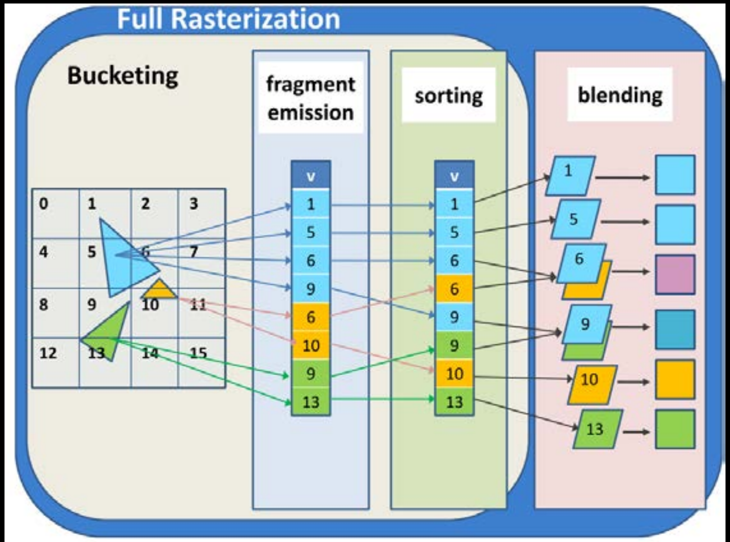
- Entirely done using the GPU **graphics pipeline**
  - Sparse (No plain grid allocation)

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



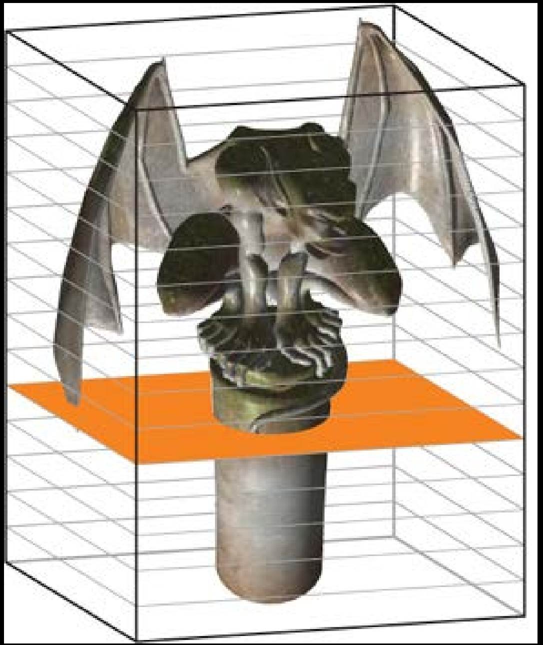
Previous GPU approaches

- *Compute*-based
  [Schwarz and Seidel 10, Pantaleoni 11]
  - Not using hw rasterizer



VoxelPipe [Pantaleoni 11]

- 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]



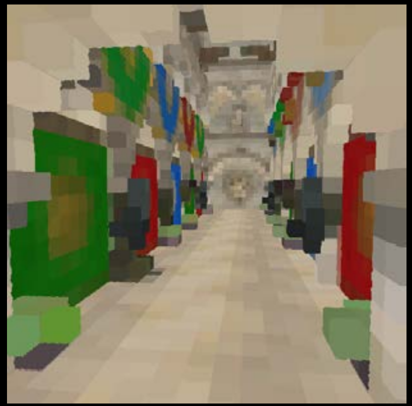
Crane et al. 2007

GPU TECHNOLOGY CONFERENCE

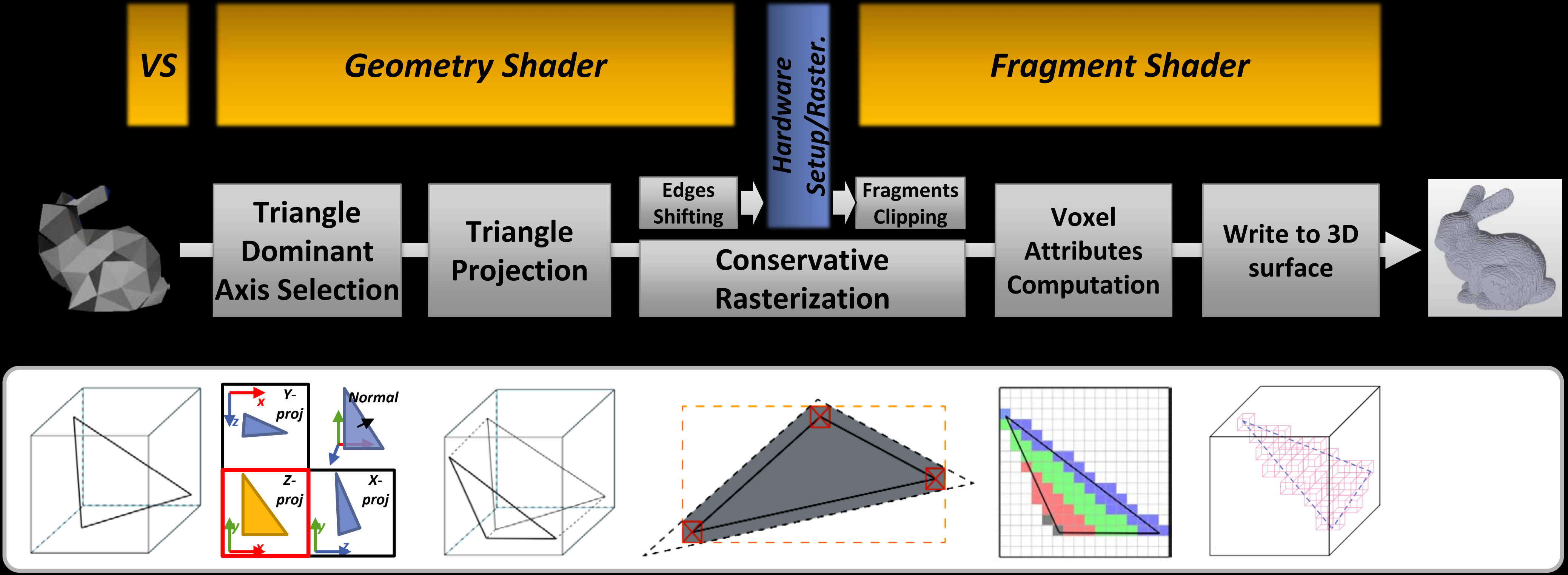
OpenGL 4.2 Image Load/Store

- 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;`

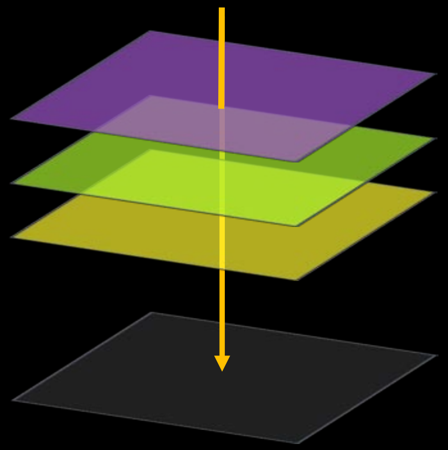
One pass voxelization pipeline



- *Thin surface voxelization*



Compositing voxel fragments

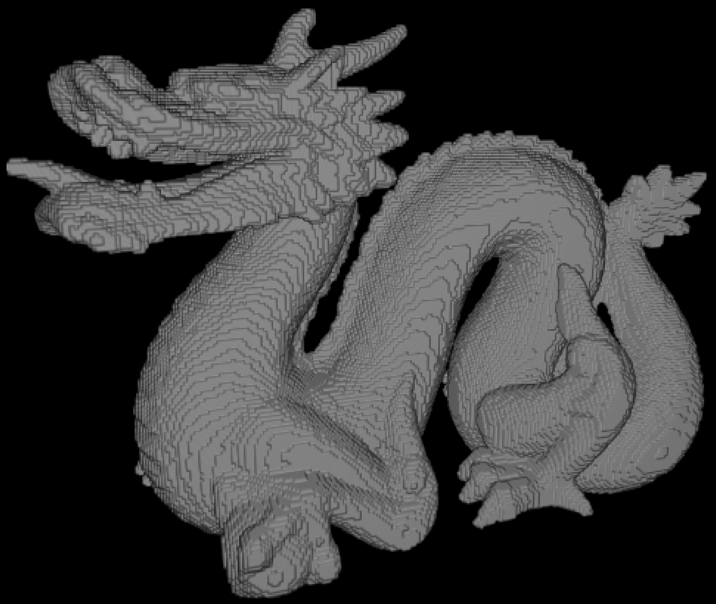


- To texture or linear buffer (global memory)
- <u>Native</u> **`AtomicAdd`**
  - — INT32
  - — INT64 *(NVIDIA Only, global memory)*
  - — FP32 *(NVIDIA Only, `NV_shader_atomic_float` extension)*
- <u>Emulation</u> for any format *(RGBA8, RGBA16F...)*
  - — **`AtomicCompSwap / AtomicCompSwap64`**
    - *(2x-3x speed penalty)*
  - — Moving average *(RGBA8)* + *Voxel Anti-Aliasing (coverage mask)*

Results

- Stanford Dragon
  - GTX 480 (GF100)

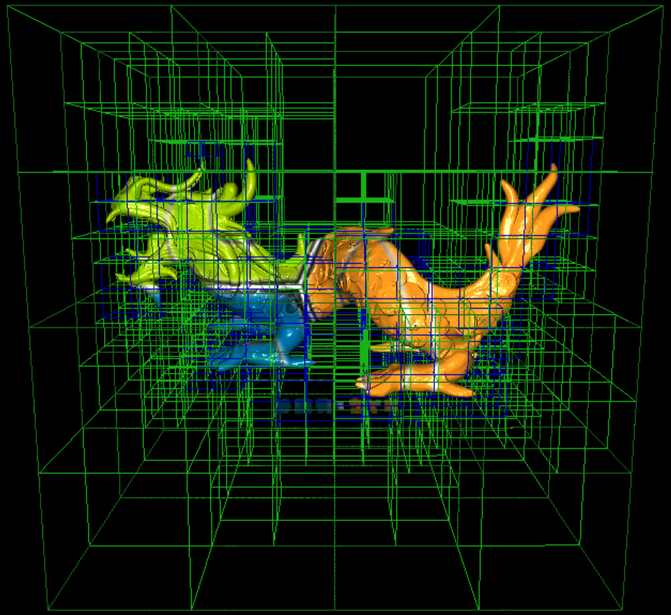
- Usually as good as, or even faster than Voxel Pipe *[Pantaleoni 11]*



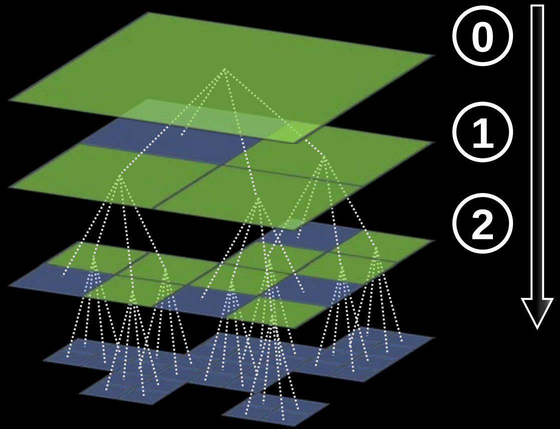
Times in ms		Std. raster.		Cons. raster.	
Format	Res	Write	Merge	Write	Merge
R32F	128	1.19	1.24	1.63	2.41
	512	1.38	2.73	1.99	5.30

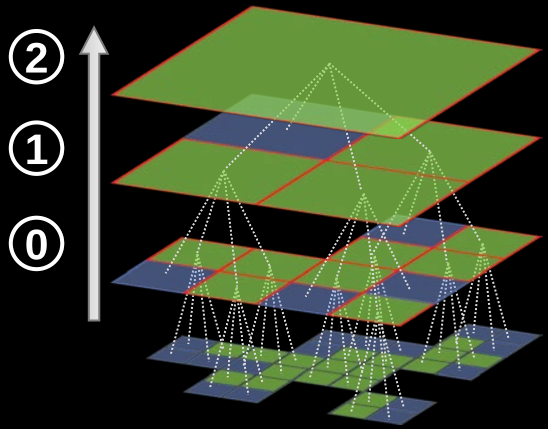
Sparse Octree construction

- **Sparse voxelization**
  - No plain grid allocation

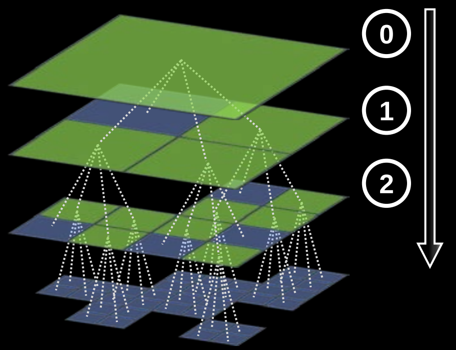


- Two steps:
  1. Octree subdivision
  2. Values MIP-mapping

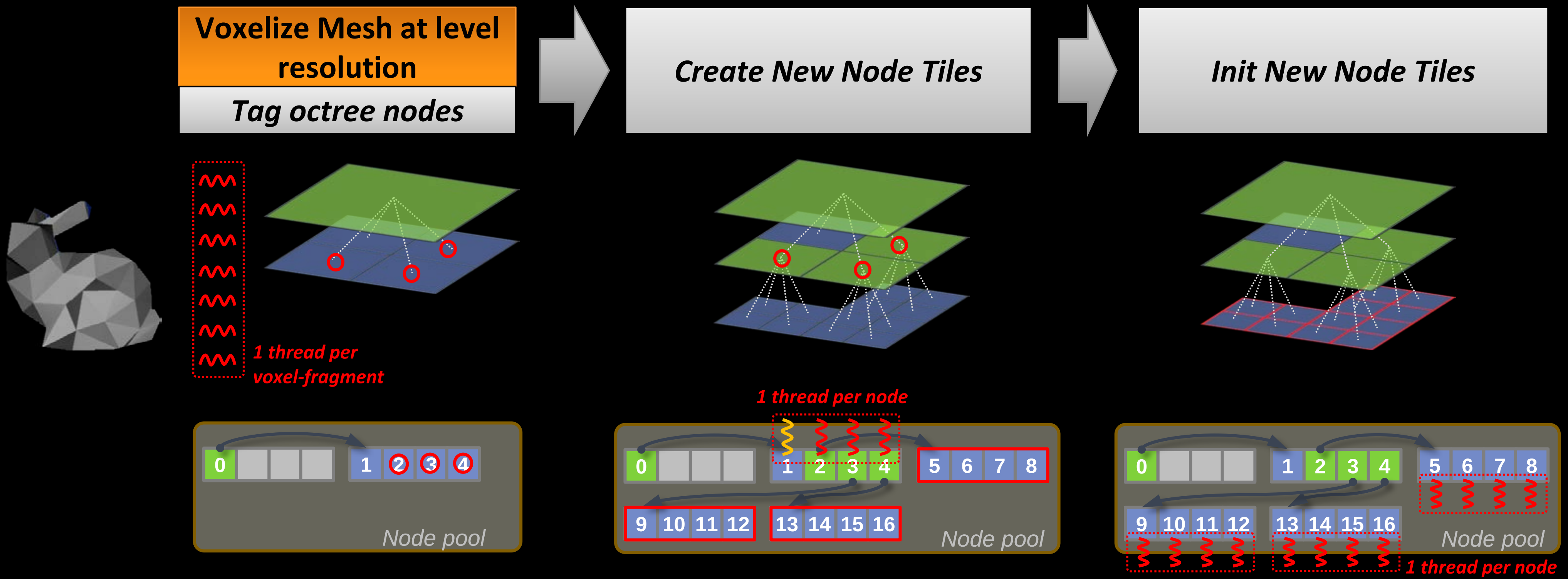




Octree construction (1/2)



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



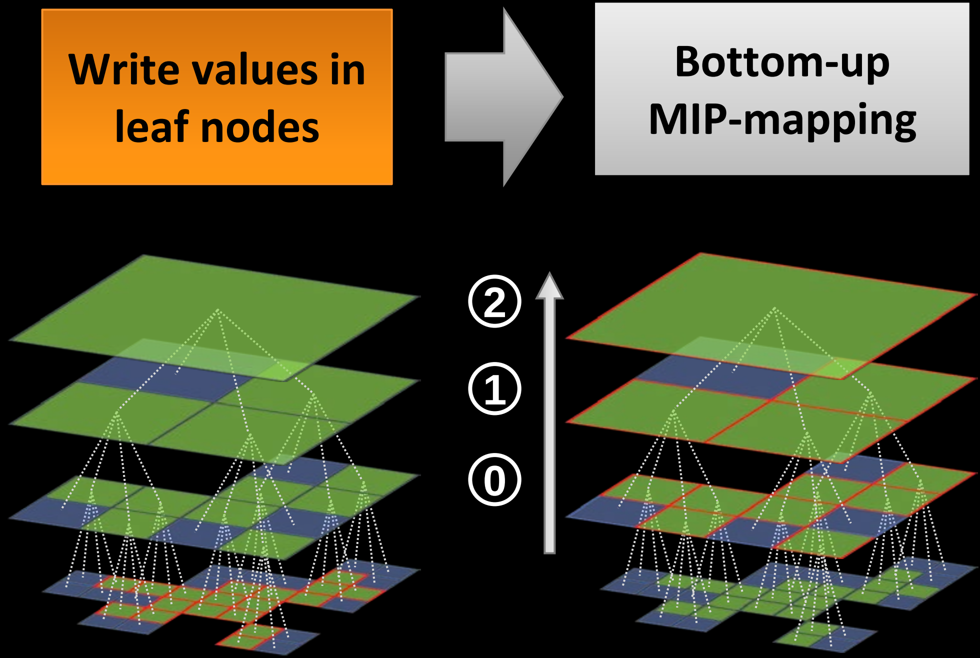
OpenGL compute kernel emulation

- 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)

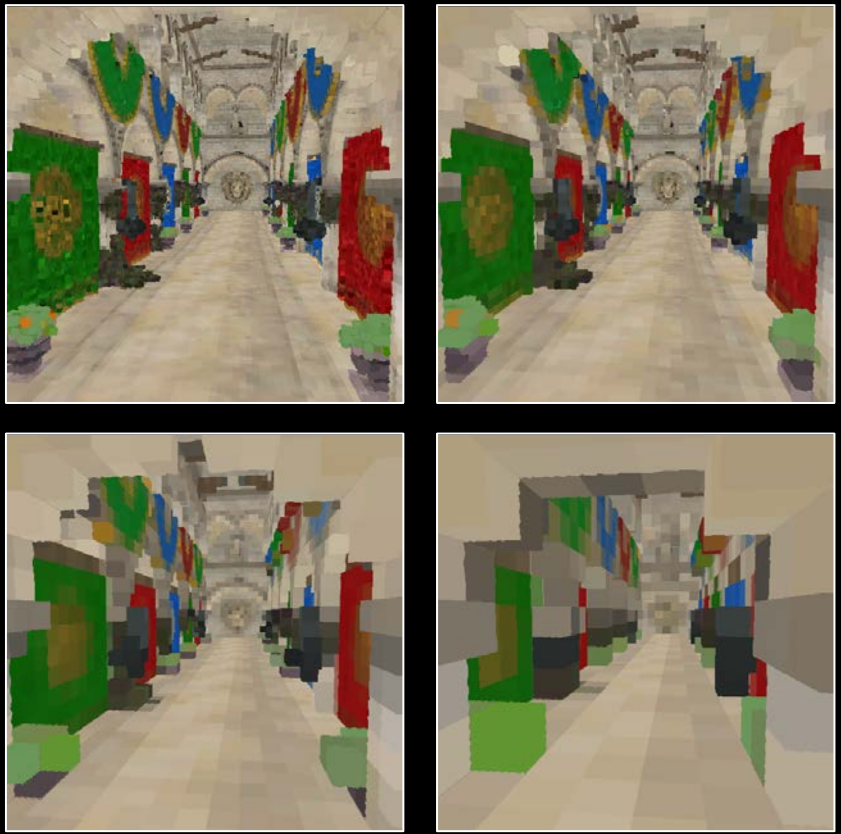
- Step 2: Populating octree with values

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 Scene	Frag list	Octree construction: Flag	Octree construction: Create	Octree construction: Init	Octree construction: Total	Write	MIP map	Total
Hand	0.17	0.89	0.18	0.35	1.42	0.9	0.55	3.04
Sponza	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