Faculty of Mathematics and Physics
Charles University in Prague
18th April 2016

Graphics for Games

Lab 08 – UE4 – VXGI (real-time GI)

Compiled from / Based on:
http://simonstechblog.blogspot.cz/2013/01/implementing-voxel-cone-tracing.html

Resources

Links

- VXGI – NVidia official site
- VXGI basics (short explanatory video)
- VXGI original paper
- Thorough explanation of the technique (used as basis for this presentation)
  - More links to papers in there
- Another paper on Voxel-Based rendering pipeline

Resources

Links - Examples

- (Semi) Official video UE4 + VXGI

- User video UE4 + VXGI 1
- User video UE4 + VXGI 2

- SVOGI (similar technique) in CryEngine on Kingdom Come: Deliverence

- SVOGI in CryEngine on Miscreated

- Voxel based GI in CryEngine (documentation)

Resources

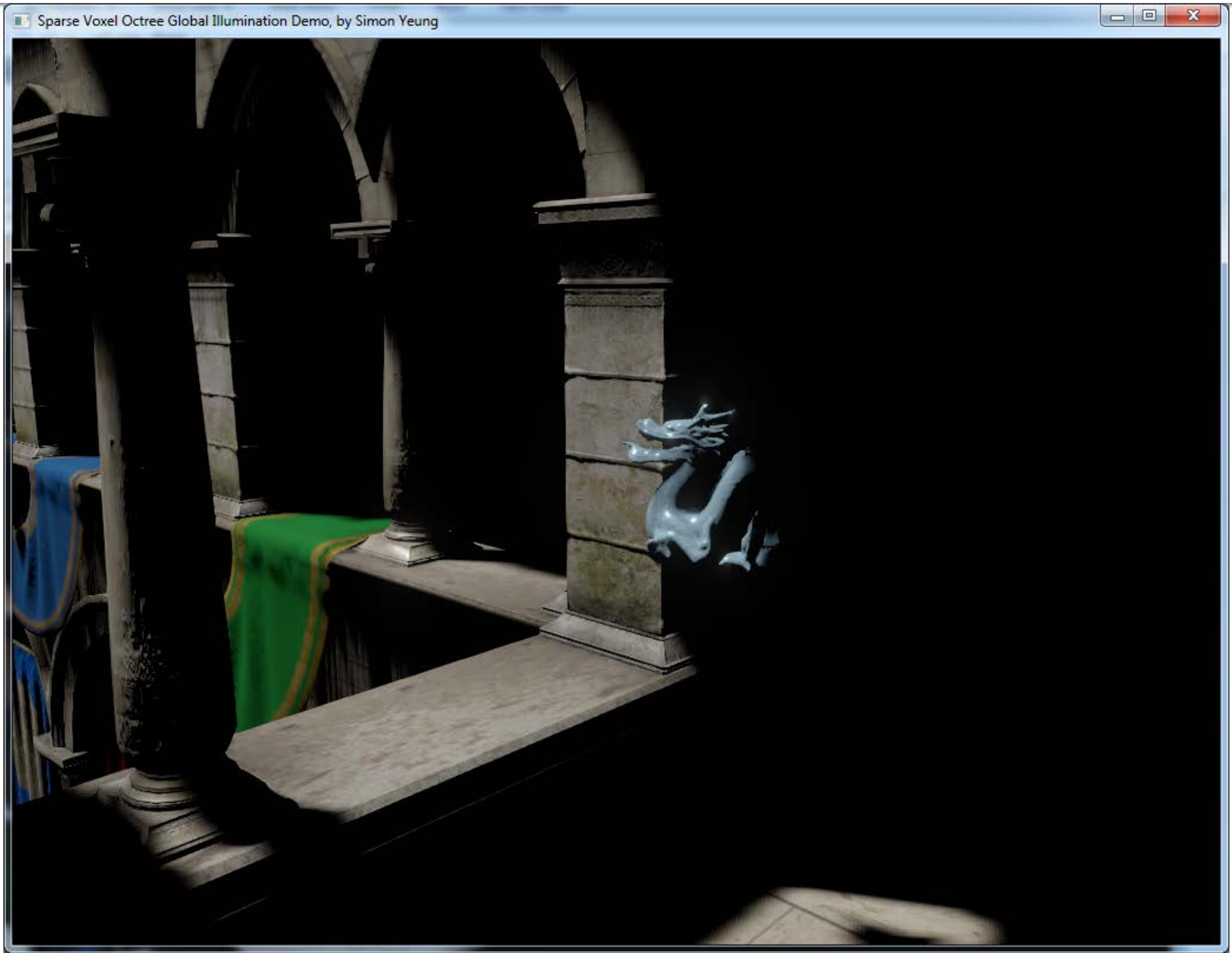
Links - GITHUB

- GITHUB - UE 4.10 + VXGI
  - How to use PDF
- GITHUB - UE 4.9.2 + VXGI + more NVidia tech

Voxel-based Global Ilumination

VXGI

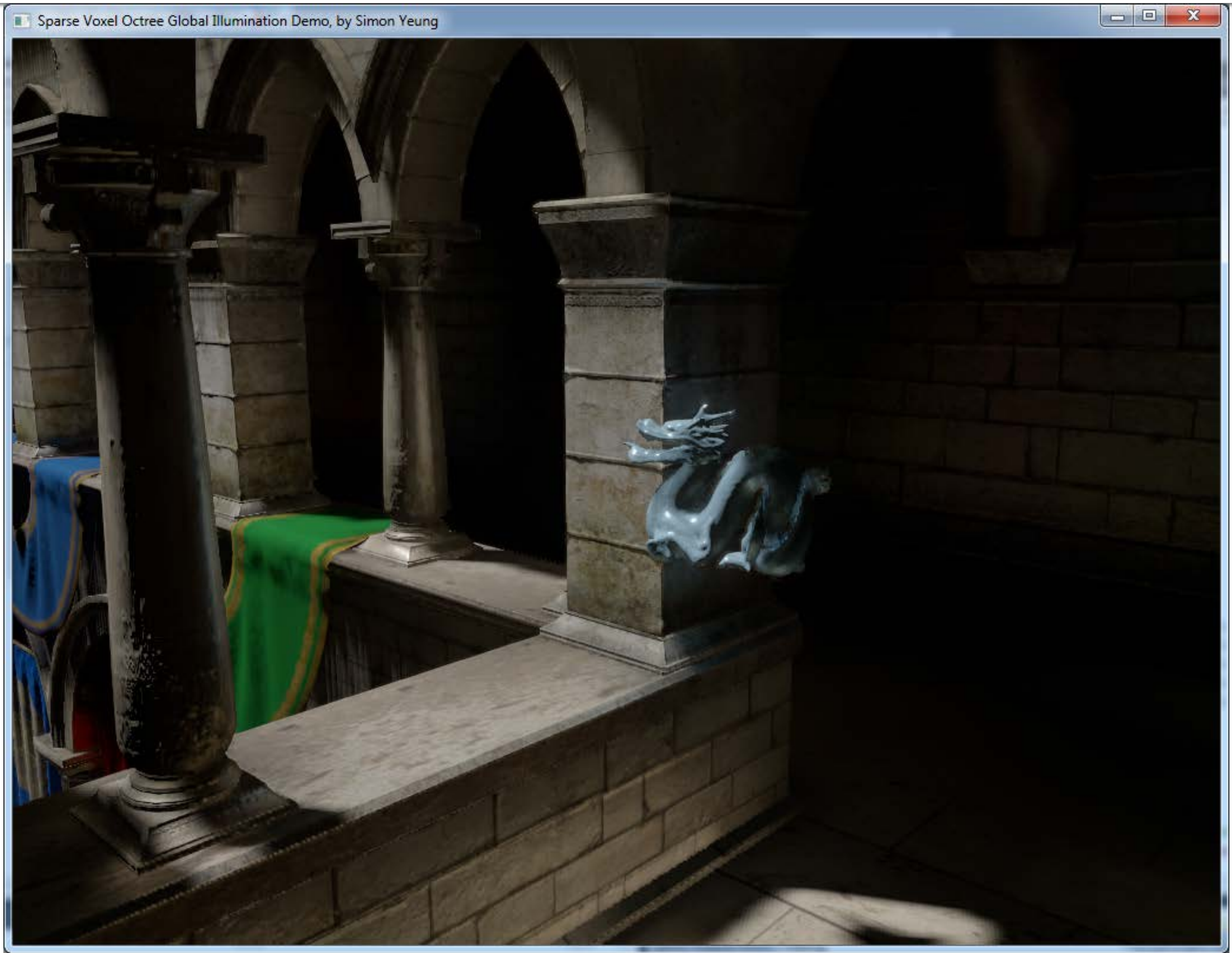
- Without GI (Direct lightning only)


Sparse Voxel Octree Global Illumination Demo, by Simon Yeung

Voxel-based Global Ilumination

VXGI

- With GI (1 bounce)


Sparse Voxel Octree Global Illumination Demo, by Simon Yeung

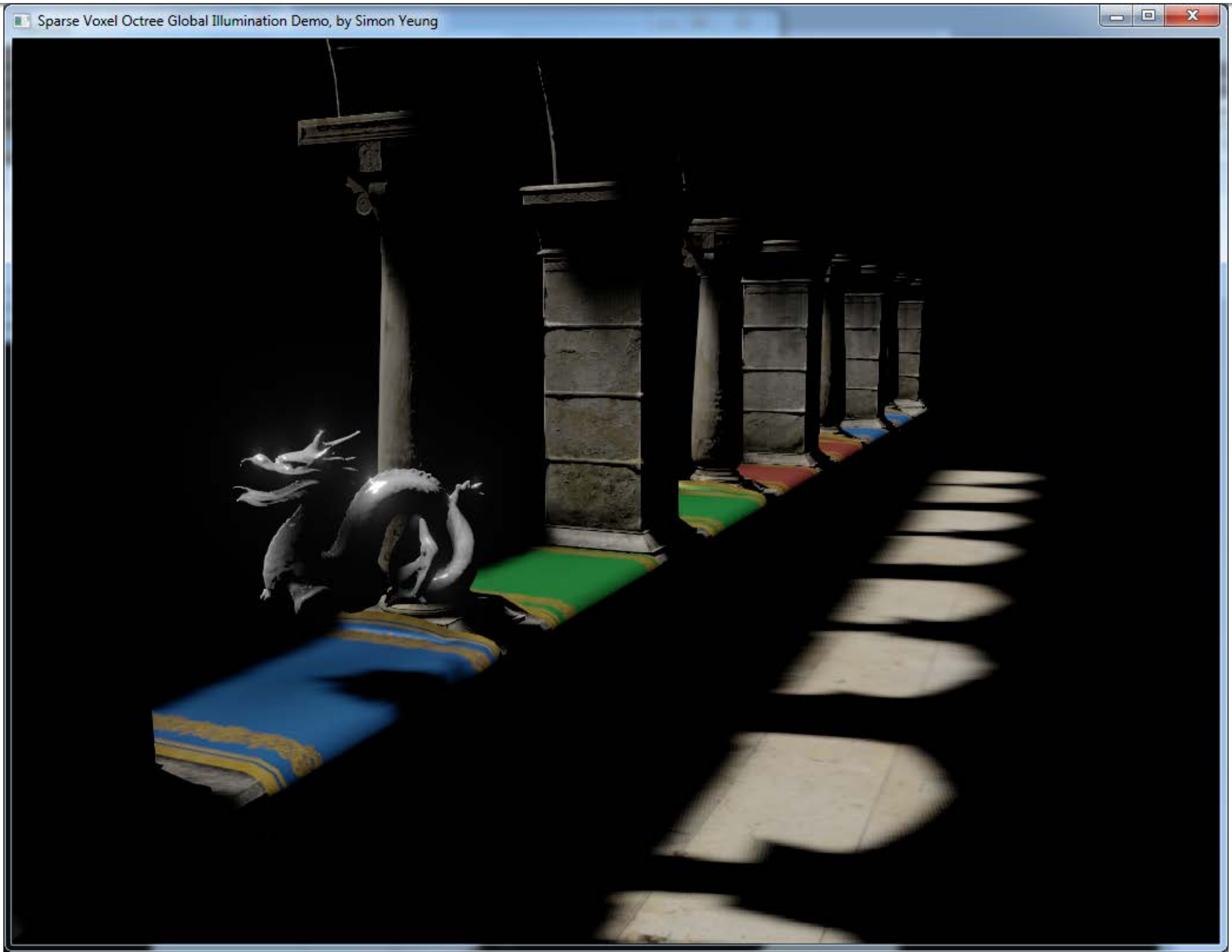
Voxel-based Global Ilumination

VXGI

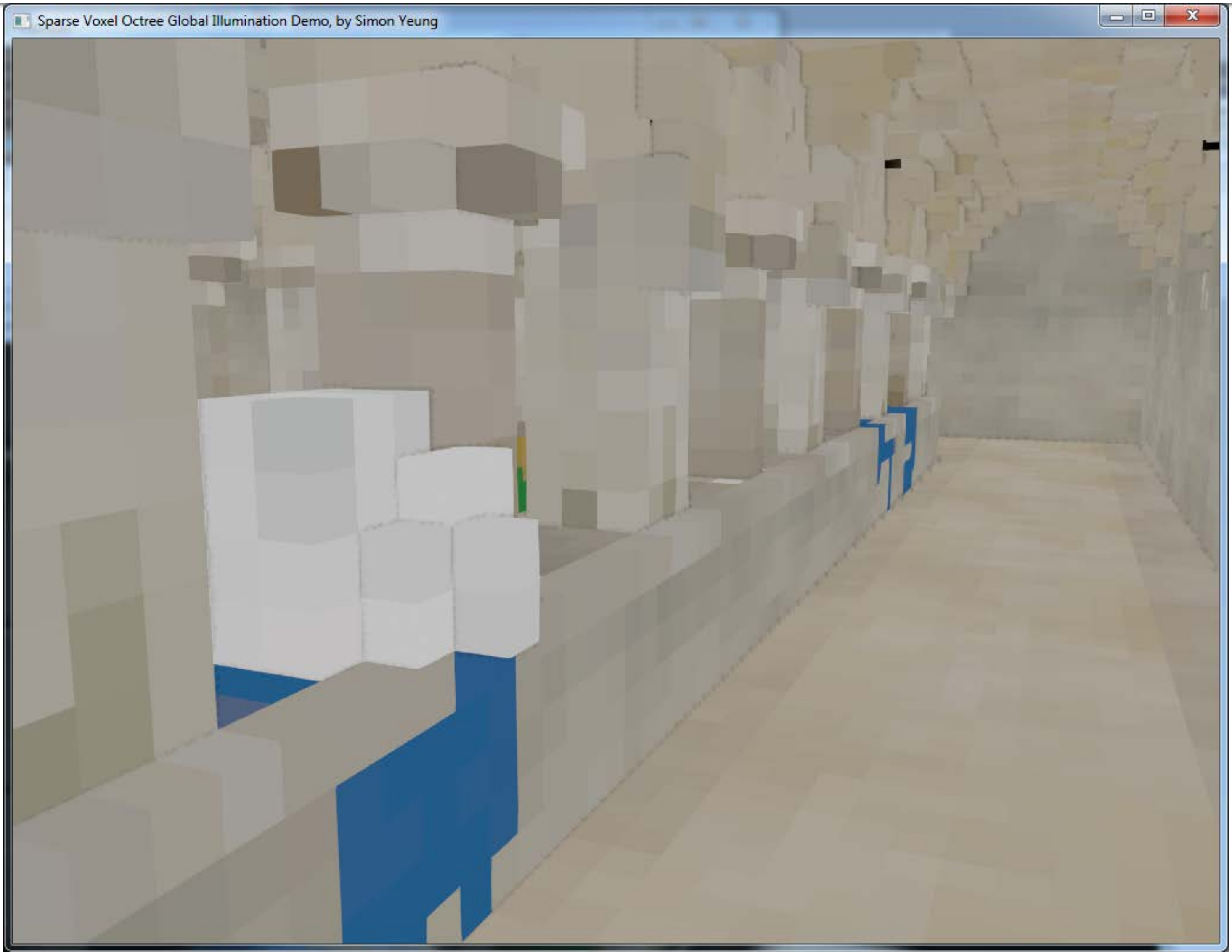
- Given a scene with directly lighting only
- Voxel-based GI involves 5 steps:
  1. Voxelize the triangle meshes
  2. Construct sparse voxel octree
  3. Inject direct lighting into the octree
  4. Filter the direct lighting to generate mip-map
  5. Sample the mip-mapped values by cone tracing

Voxel-based Global Ilumination

Given a scene with directly lighting only

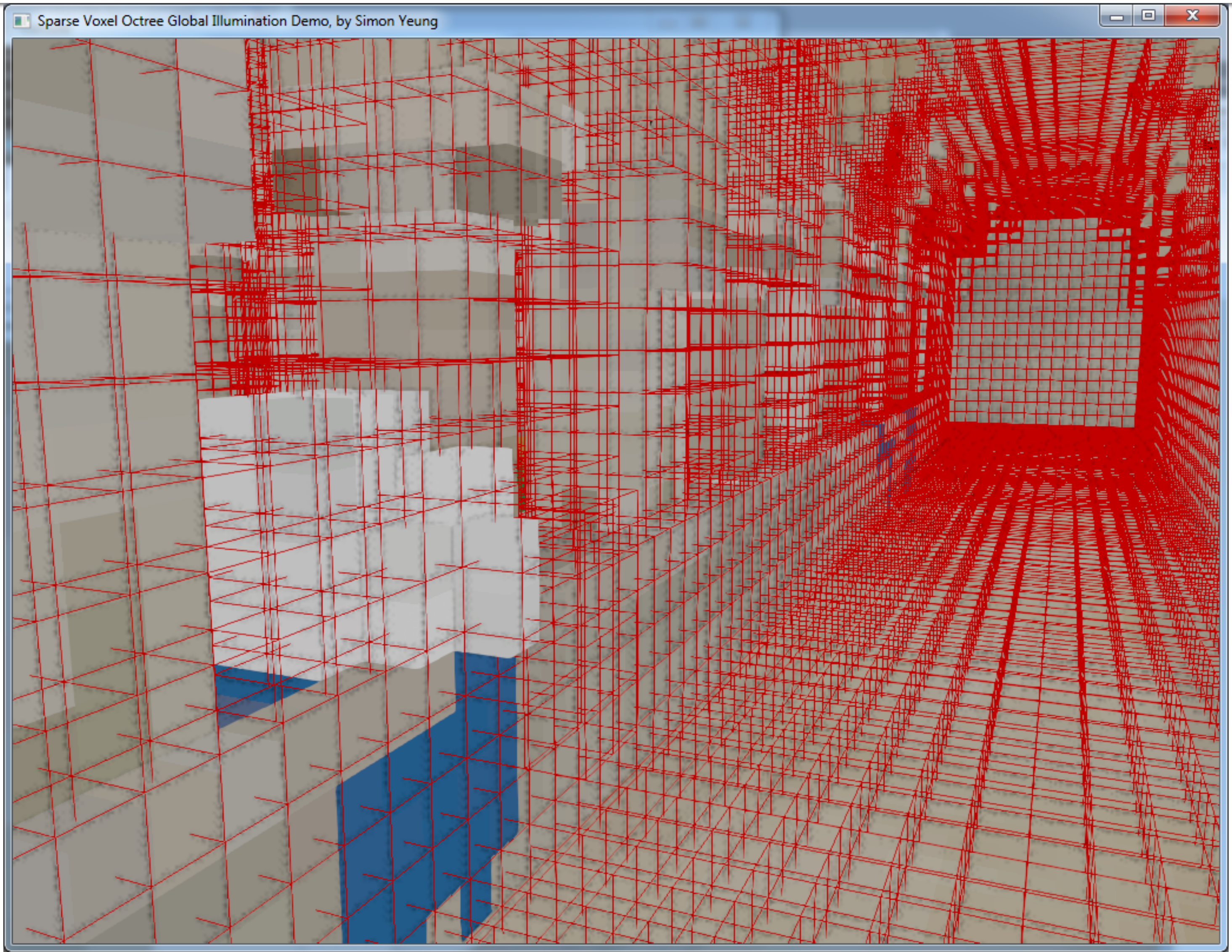

Sparse Voxel Octree Global Illumination Demo, by Simon Yeung

Voxel-based Global Ilumination
1. Voxelize the triangle meshes


Sparse Voxel Octree Global Illumination Demo, by Simon Yeung

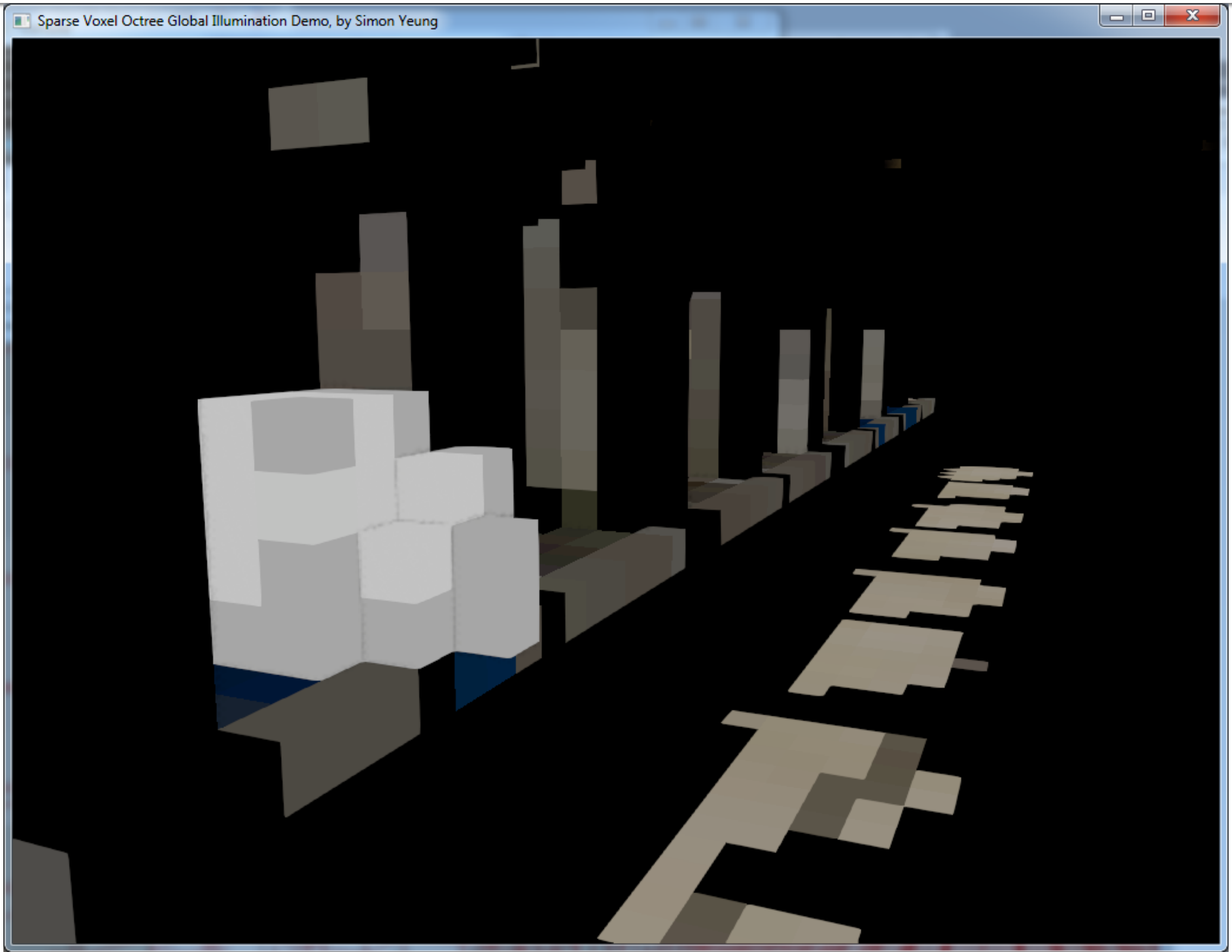
Voxel-based Global Ilumination

2. Construct sparse voxel octree


Sparse Voxel Octree Global Illumination Demo, by Simon Yeung

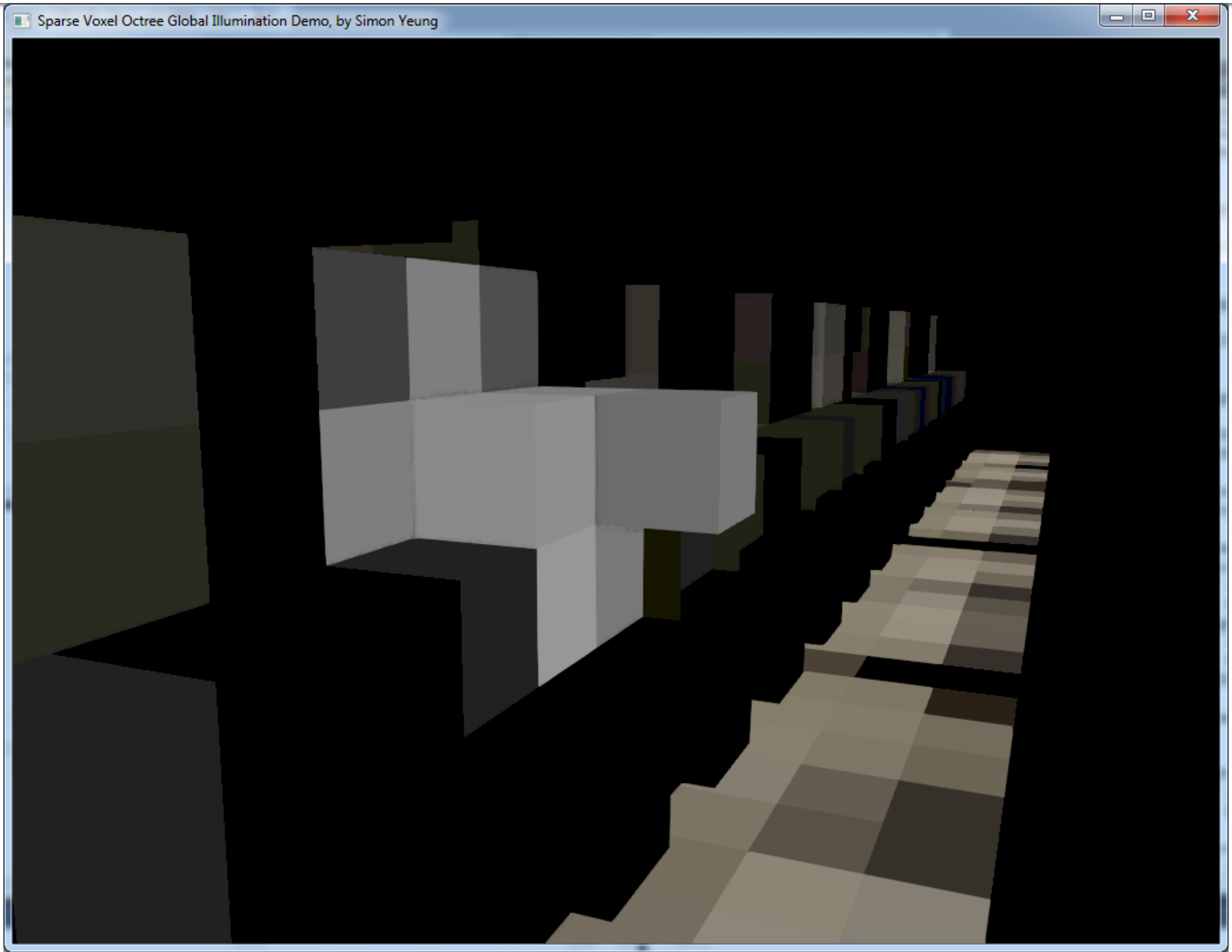
Voxel-based Global Ilumination

3. Inject direct lighting into the octree


Sparse Voxel Octree Global Illumination Demo, by Simon Yeung

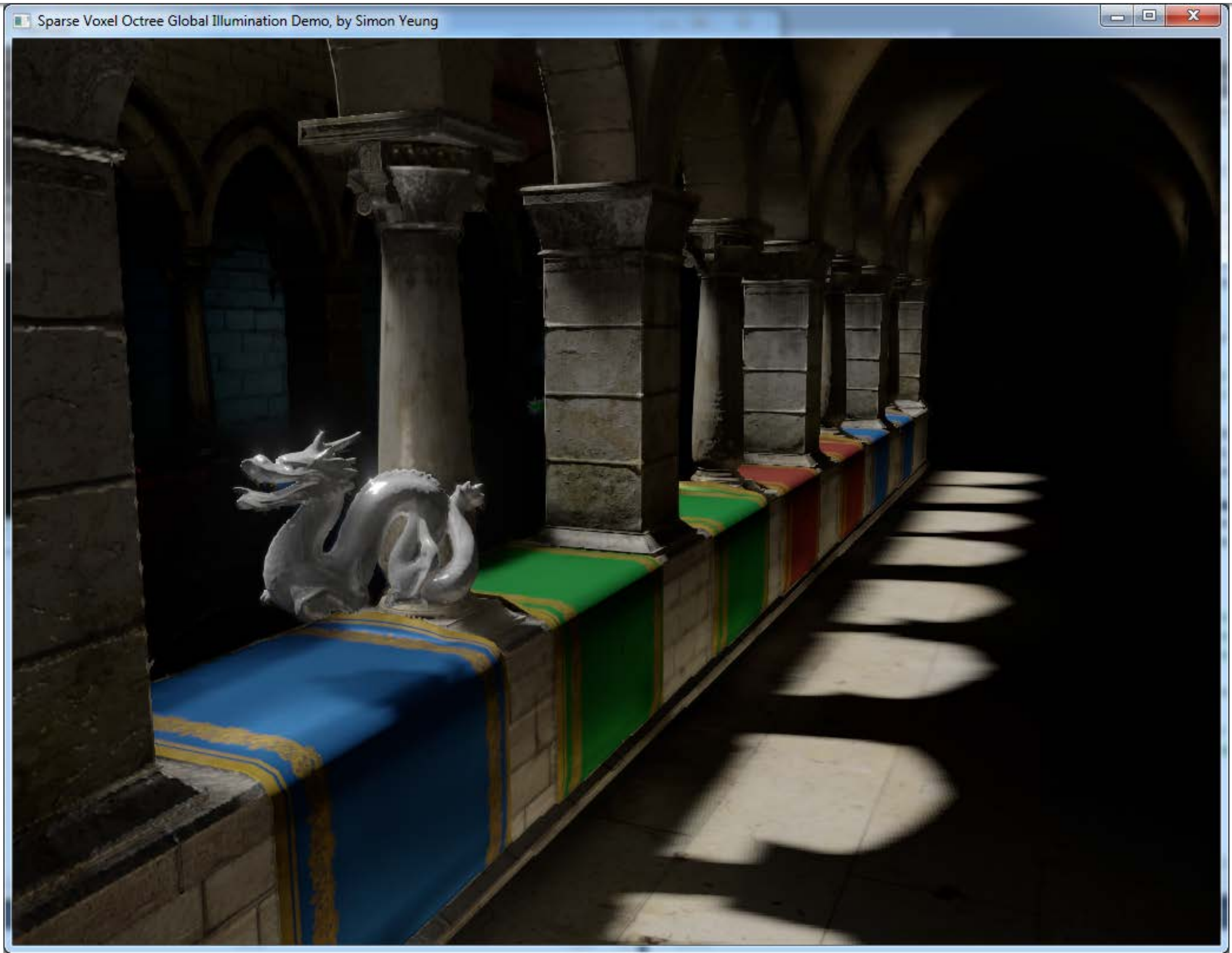
Voxel-based Global Ilumination

4. Filter the direct lighting to generate mip-map


Sparse Voxel Octree Global Illumination Demo, by Simon Yeung

Voxel-based Global Ilumination

5. Sample the mip-mapped values by cone tracing


Sparse Voxel Octree Global Illumination Demo, by Simon Yeung

Voxel-based Global Ilumination
VXGI

LET’S BREAK IT STEP BY STEP

Voxel-based Global Ilumination

VXGI

1. Voxelize the triangle meshes

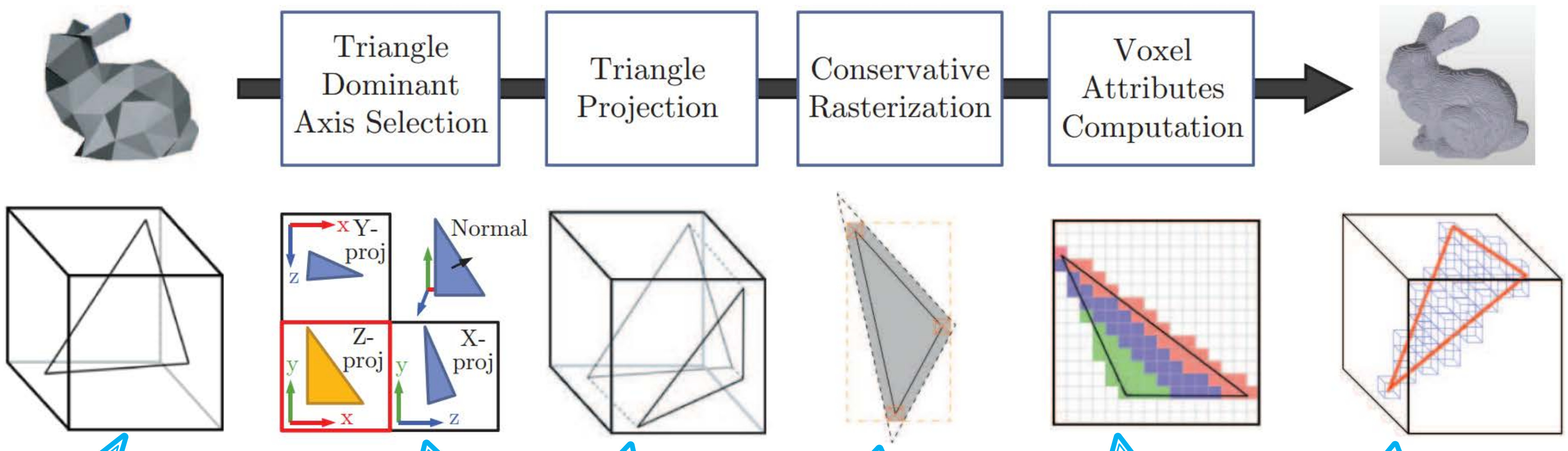
1. Voxelize the triangle meshes

Original scene

Sparse Voxel Octree Global Illumination Demo, by Simon Yeung

1. Voxelize the triangle meshes

Voxelization



Triangle Dominant Axis Selection
Triangle Projection
Conservative Rasterization
Voxel Attributes Computation
x
Y-proj
z
Normal
Z-proj
y
x
X-proj
y
z

[1] A triangle made it into the Geometry Shader (GS)

[2] In GS, select the axis so you maximize the projected area of the triangles

[3] In GS, project the triangle and replace the original one

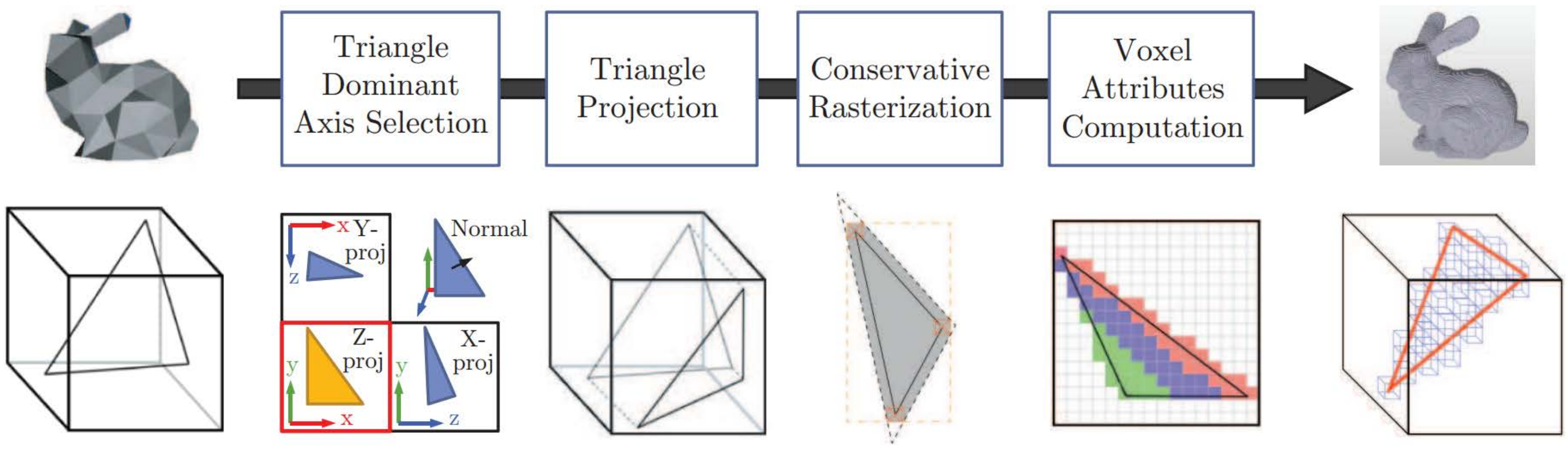
[4] In GS, make the triangle larger, so top fragments are not clipped out.

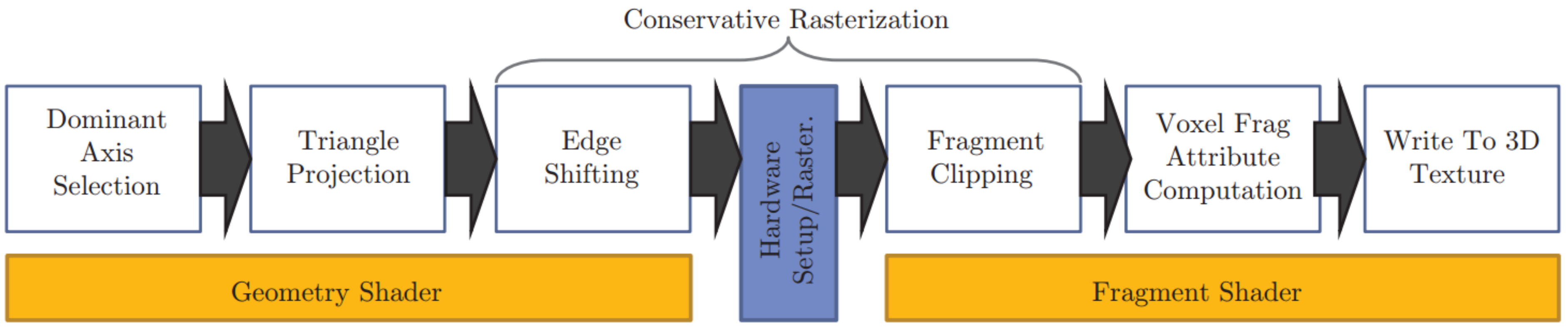
[5] Triangle gets rasterized as usual and we carry on depth, color, …

[6] So we can postprocess 2D image into 3D octree

1. Voxelize the triangle meshes

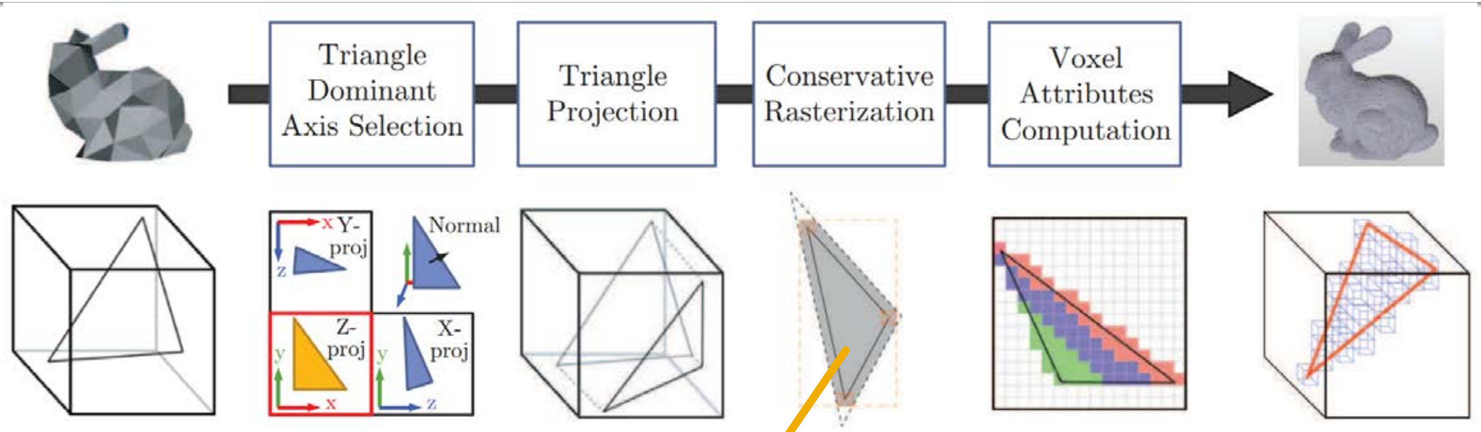
Voxelization

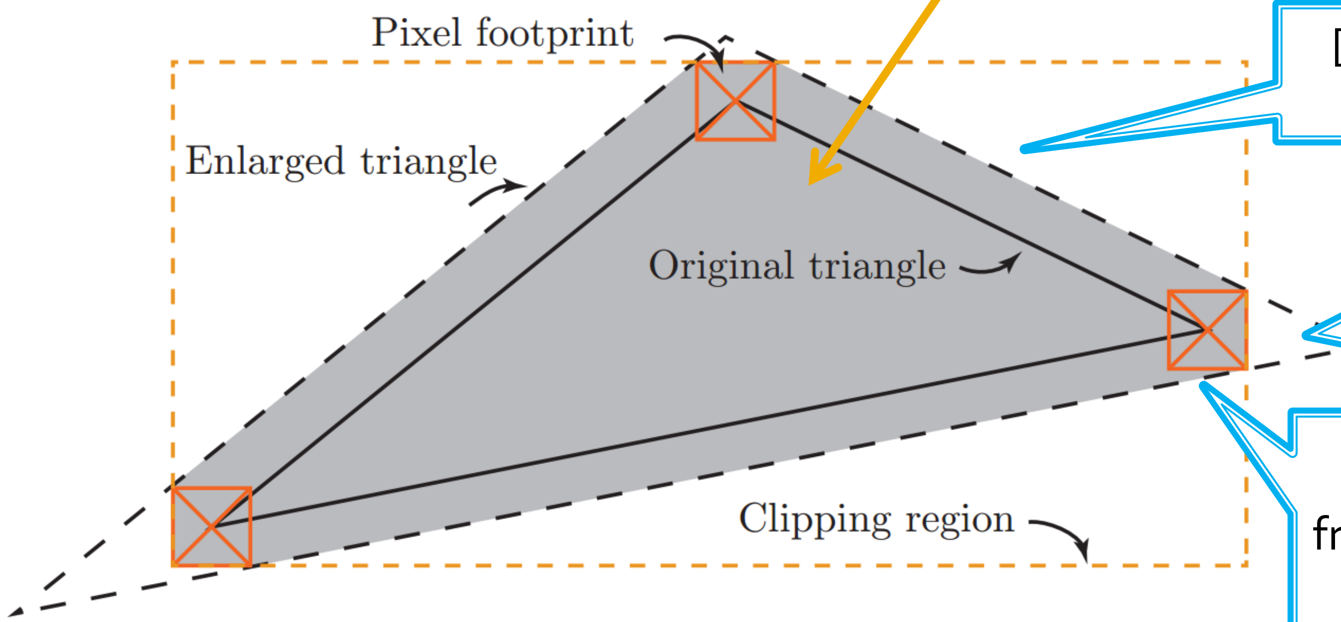

Triangle Dominant Axis Selection
Triangle Projection
Conservative Rasterization
Voxel Attributes Computation
x
Y-proj
z
Normal
Z-proj
y
x
y
X-proj
z



1. Voxelize the triangle meshes

Voxelization


Triangle Dominant Axis Selection
Triangle Projection
Conservative Rasterization
Voxel Attributes Computation
Y-proj
Normal
Z-proj
X-proj
x
y
z



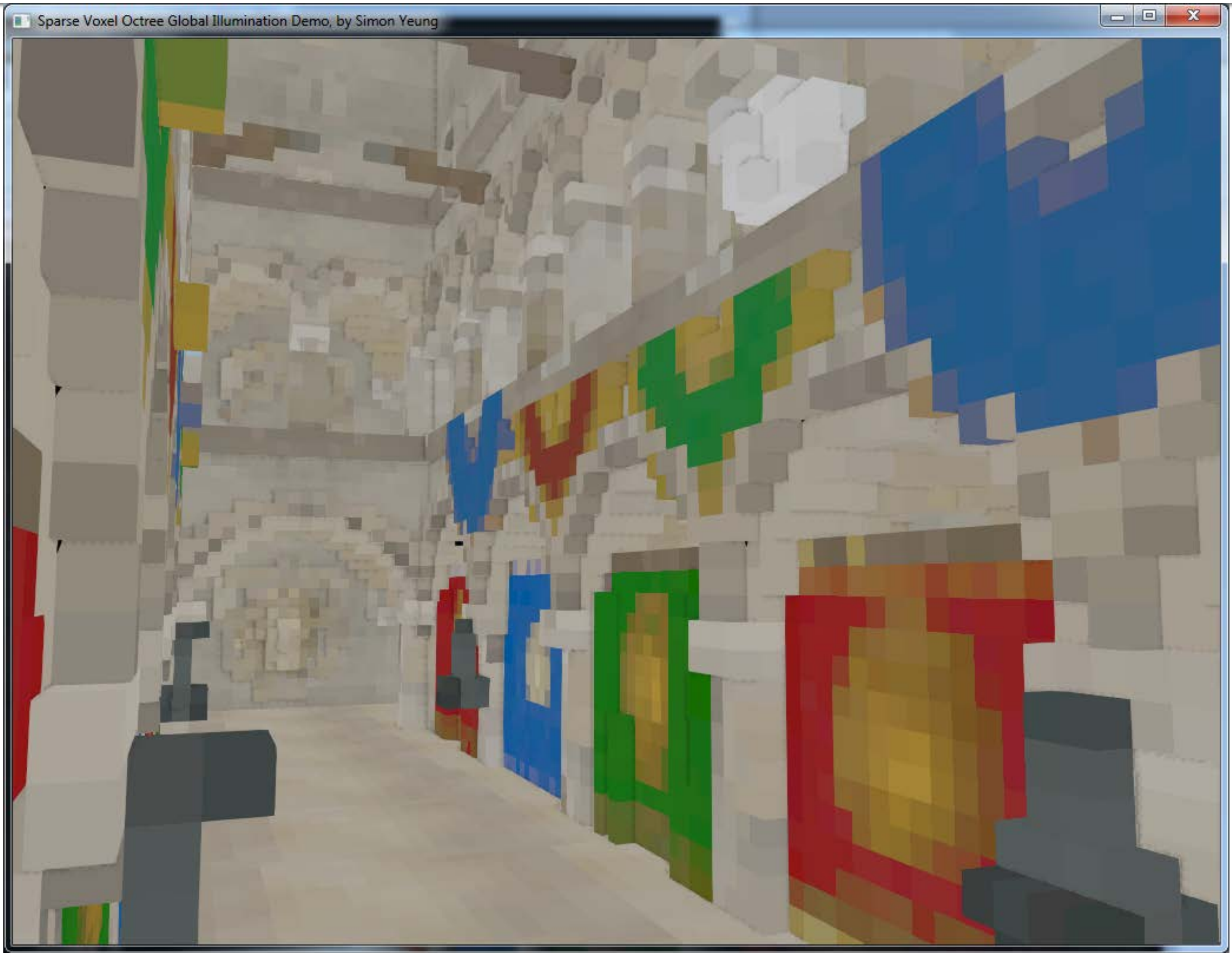
[2] Shift each edge of the triangle

[3] Possible excess pixels must be clipped out within Fragment Shader

[1] Make sure this fragment get dispatched into Fragment Shader

1. Voxelize the triangle meshes

Voxelized scene

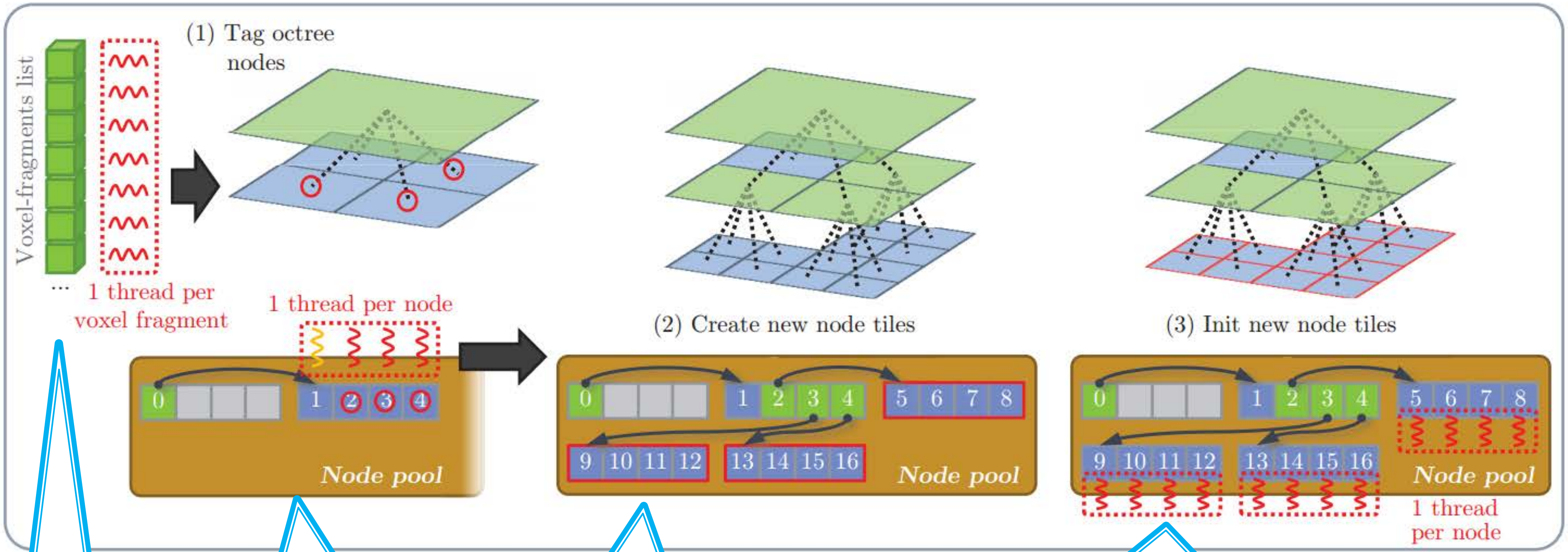

Sparse Voxel Octree Global Illumination Demo, by Simon Yeung

Voxel-based Global Ilumination
VXGI

2. Construct sparse voxel octree

Construct sparse voxel octree

Transforming voxel fragment list



[1] You have your list of voxels processed in parallel

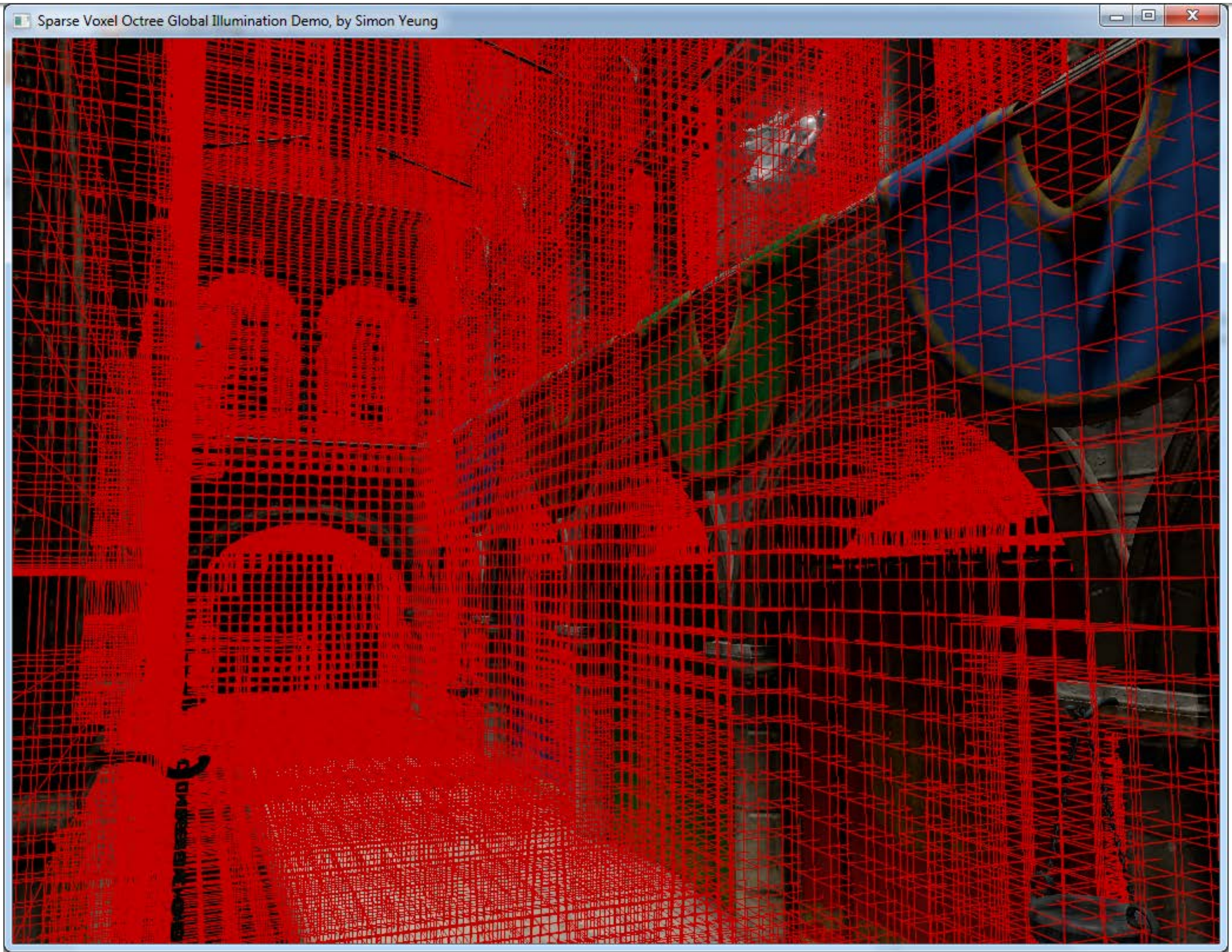
[2] Multiple nodes will be falling into a node that will need to be split

[3] Each "node to be split" can be processed in parallel

[4] Until we reach the atomic size of the node and combine (average) voxel data

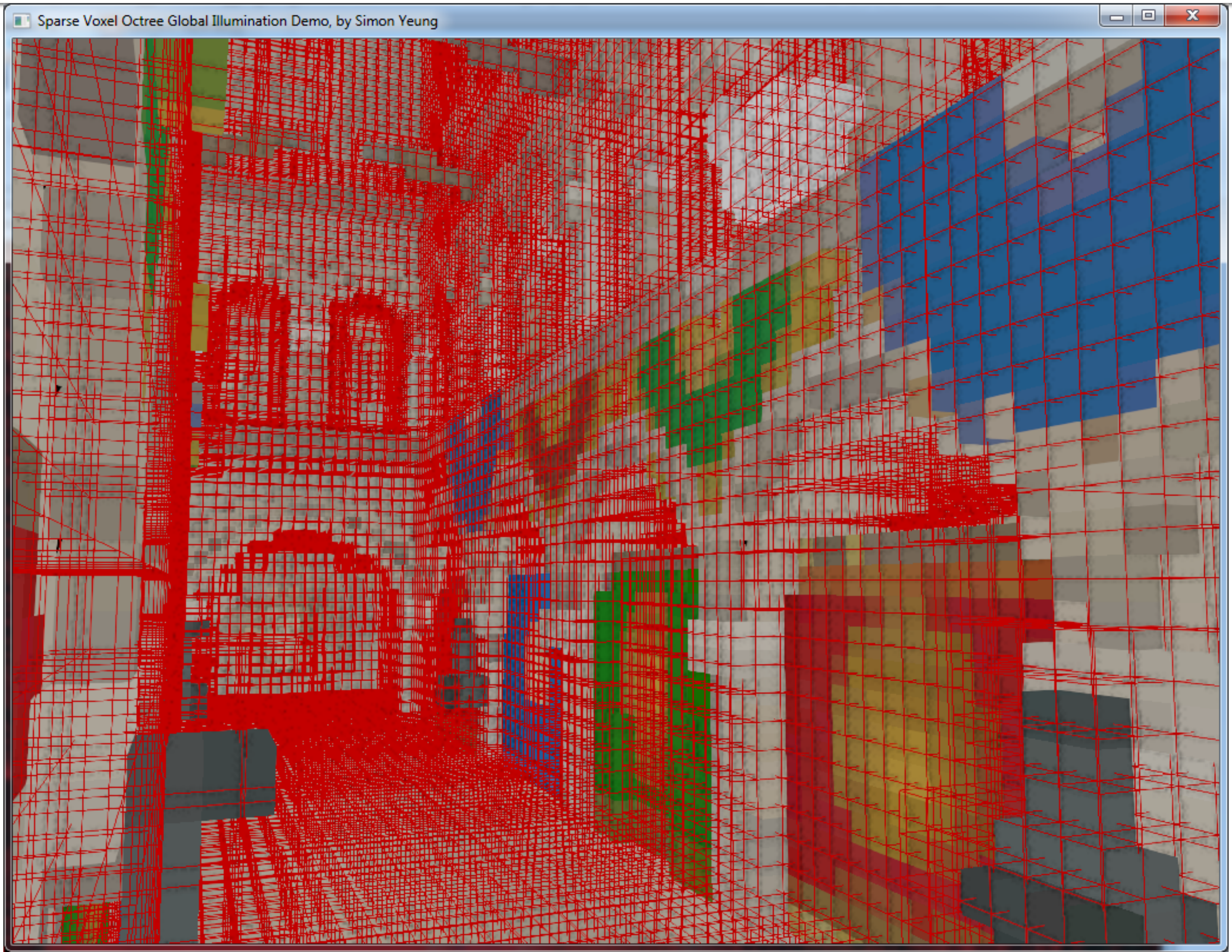
Construct sparse voxel octree

The result – Octree over the scene


Sparse Voxel Octree Global Illumination Demo, by Simon Yeung

Construct sparse voxel octree

The result – Octree over the voxelized scene


Sparse Voxel Octree Global Illumination Demo, by Simon Yeung

Voxel-based Global Ilumination

VXGI

3. Inject direct lighting into the octree

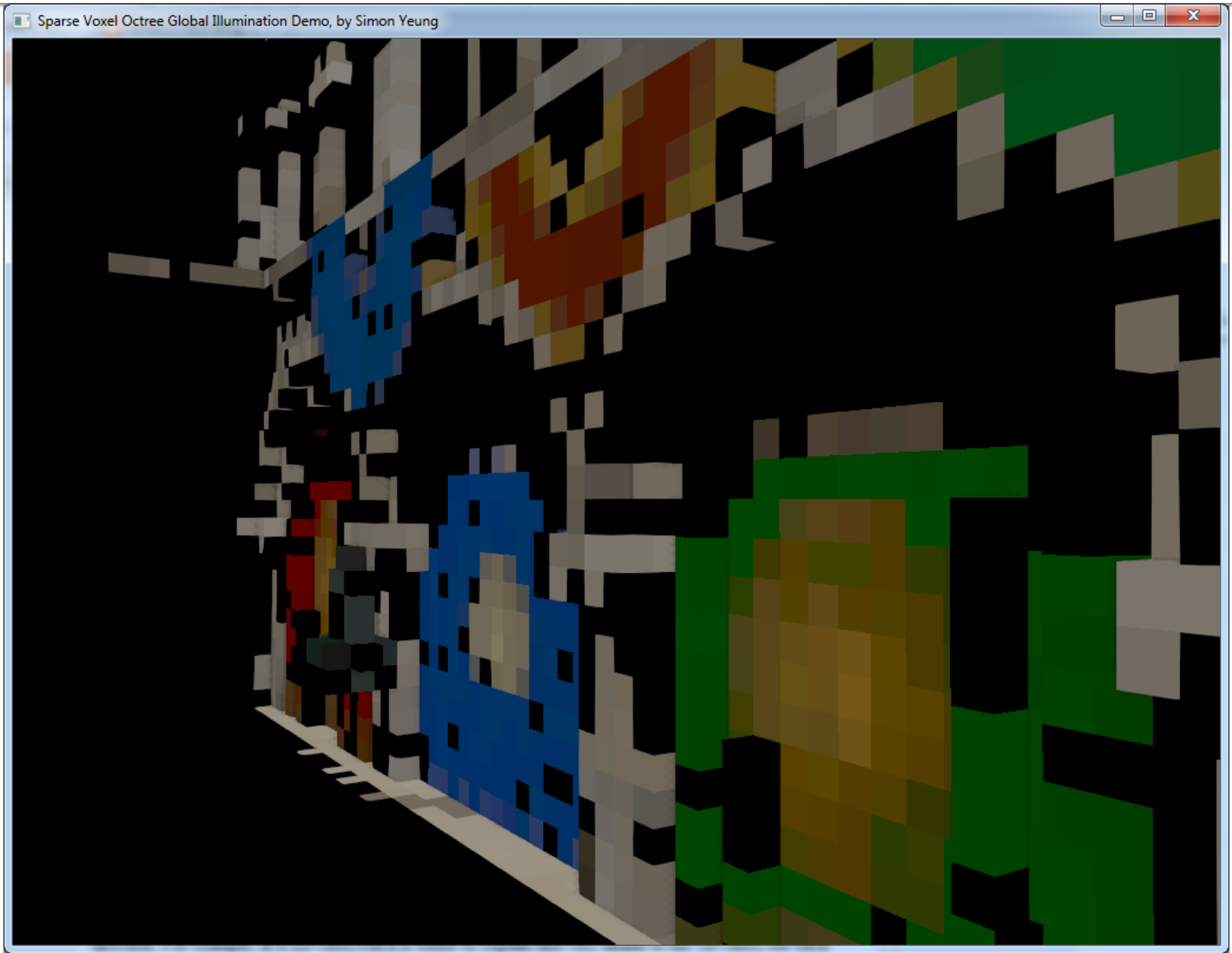
[1] Frankly, render shadow map from the point of view of all lights

[2] Extract world position and traverse octree

[3] Combine reflected radiance (separately for diffuse / specular)

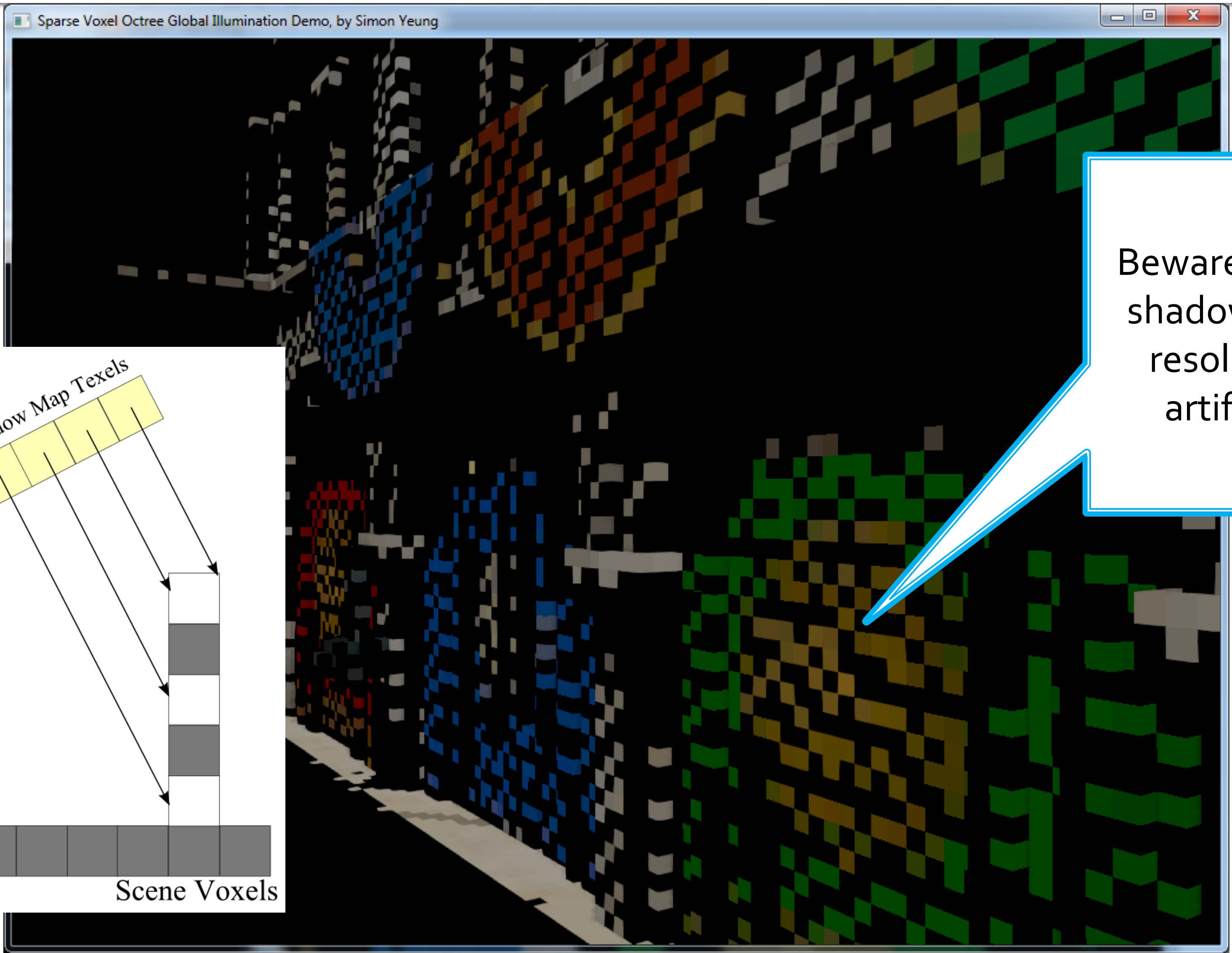
Voxel-based Global Ilumination

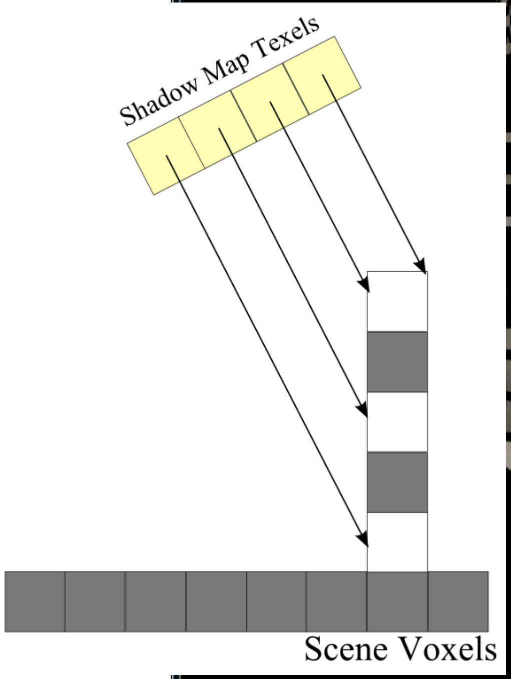
Inject direct lighting into the octree


Sparse Voxel Octree Global Illumination Demo, by Simon Yeung

Voxel-based Global Ilumination

Inject direct lighting into the octree


Sparse Voxel Octree Global Illumination Demo, by Simon Yeung



Beware of the shadow map resolution artifacts

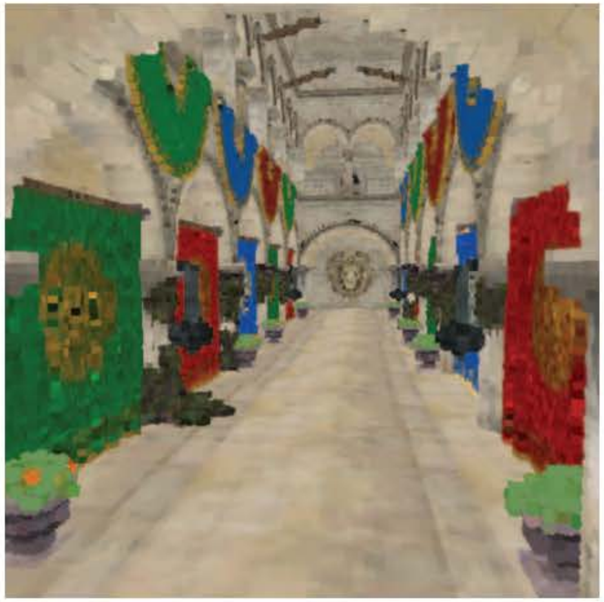
Voxel-based Global Ilumination

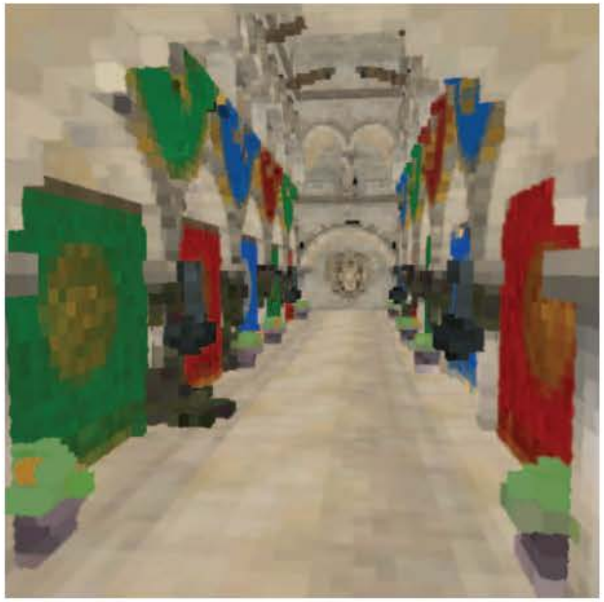
VXGI

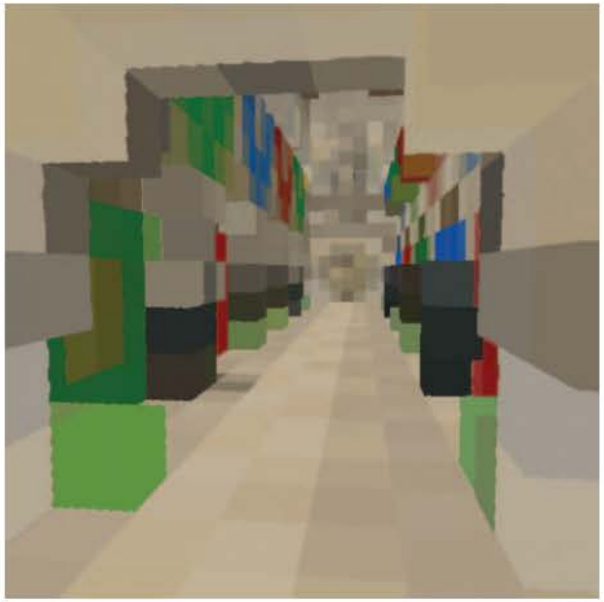
4. Filter the direct lighting to generate mip-map

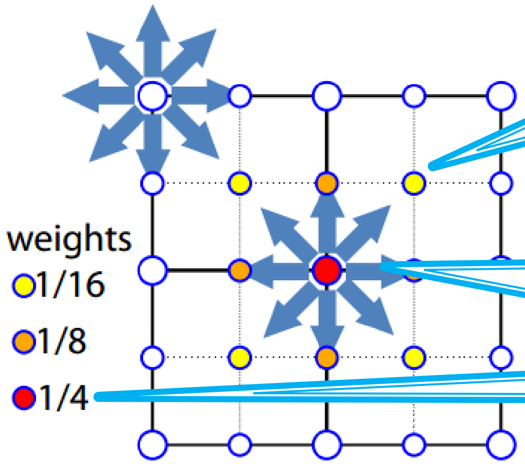
Voxel-based Global Ilumination

Filter the direct lighting to generate mip-map









[1] In VXGI there are vertex-centered voxels

[2] Thus lower-level voxels shares higher-level voxels

[3] So we need to distribute evenly the contribution