Voxel-Based Global-Illumination

By Thiedemann, Henrich, Grosch, and Müller

Real-Time Near-Field Global Illumination on a Voxel Model

Seyedmorteza Mostajabodaveh (Morteza),
M.Sc. Marc Treib

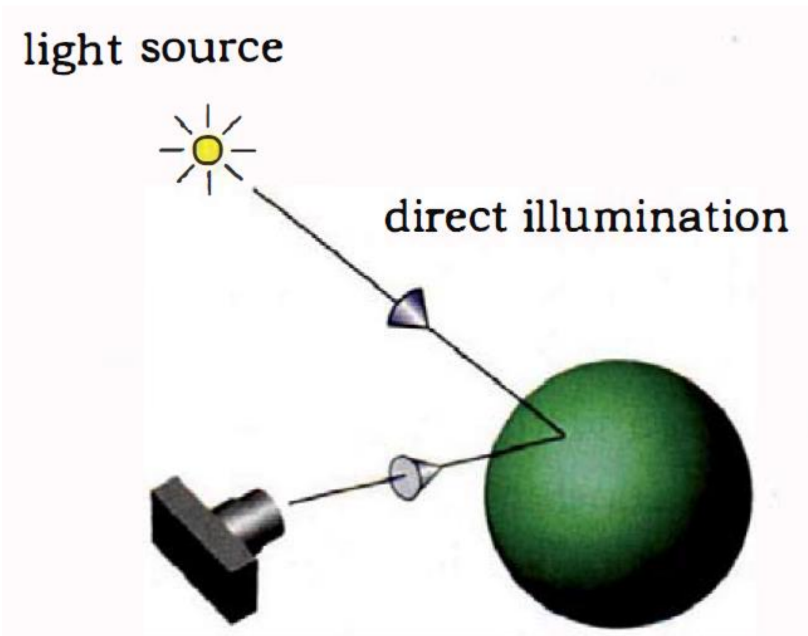
computer graphics & visualization

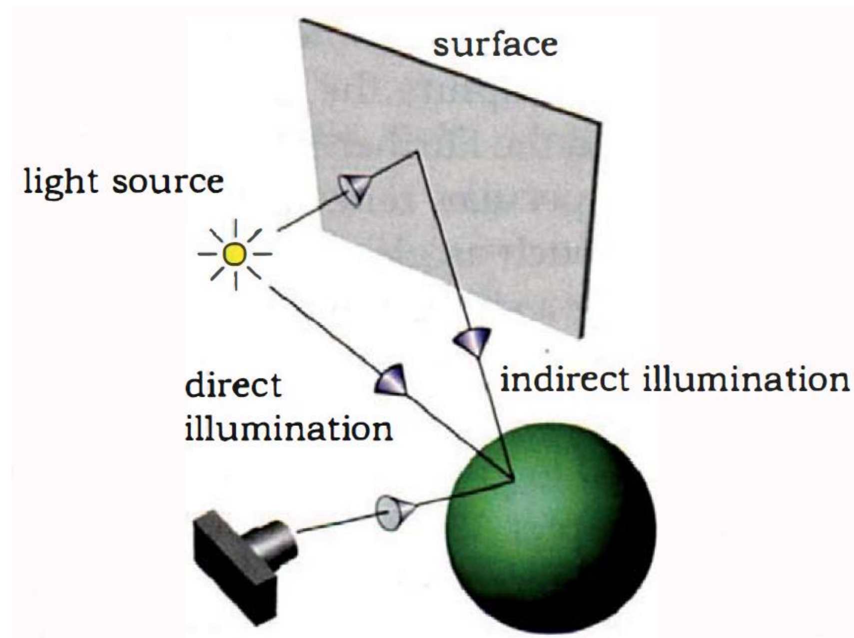
Overview

- Illumination
- What is voxelization?
- Paper’s Voxelization Method
- Paper’s Voxel/Ray Intersection Method
- Near Global-Illumination Based on Voxel-Model
- Results

Illumination

- Direct Illumination
- Indirect Illumination

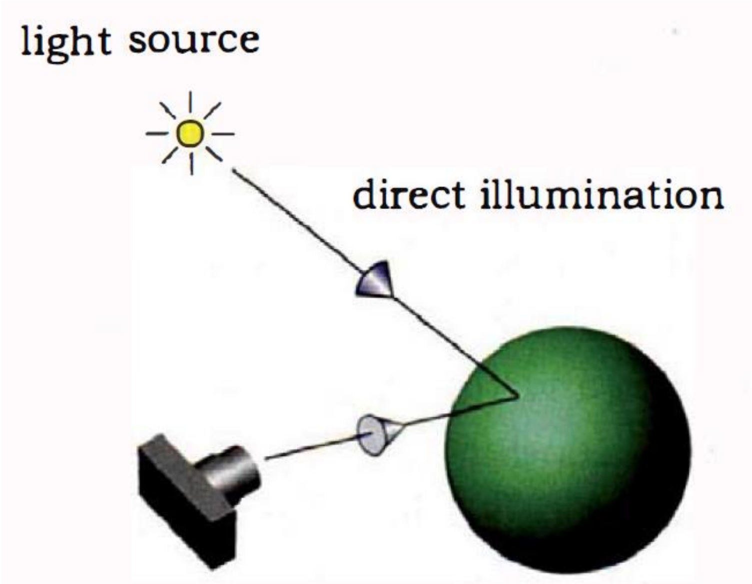




computer graphics & visualization

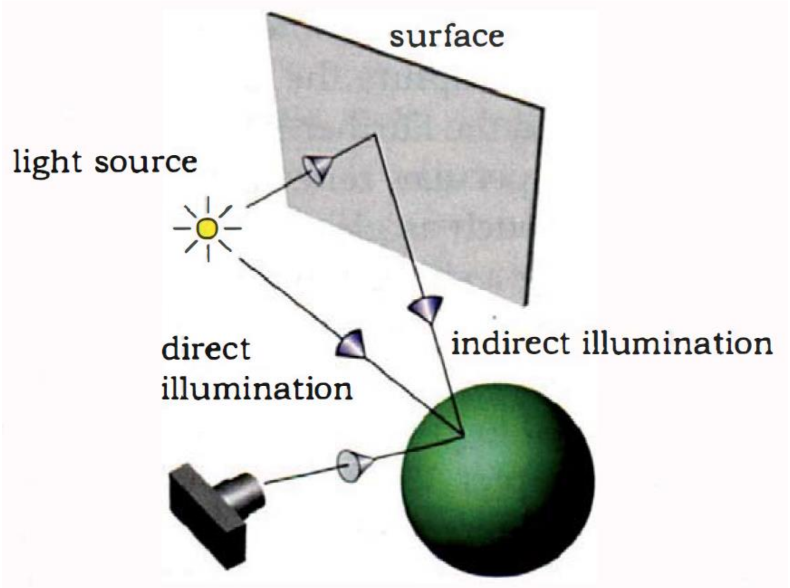
Types of Illumination

- **Direct Illumination**
  - Reflection can be computed locally (using Normal and Material Specification)
  - **Local Illumination**

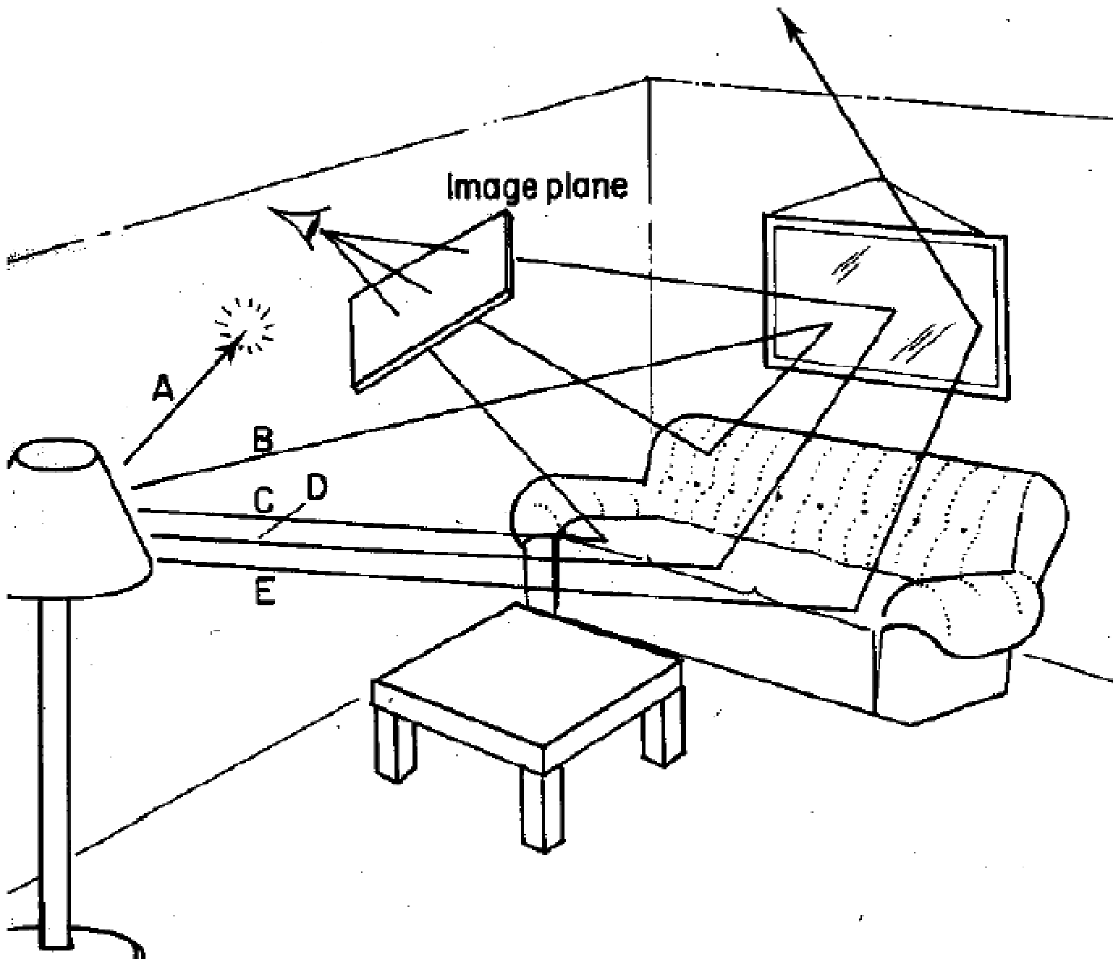


Types of Illumination

- **Indirect Illumination**
- The reflected light cannot be computed locally
  - **Global Illumination**

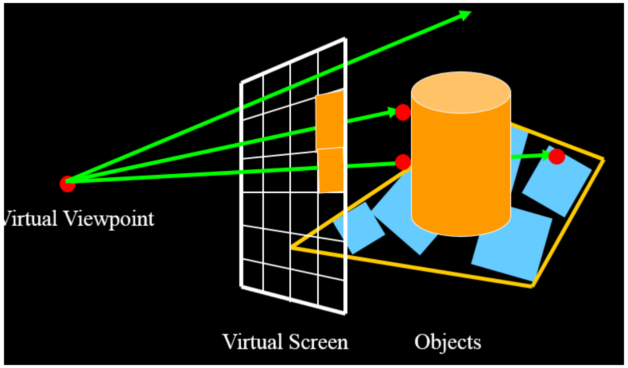


Ray Tracing



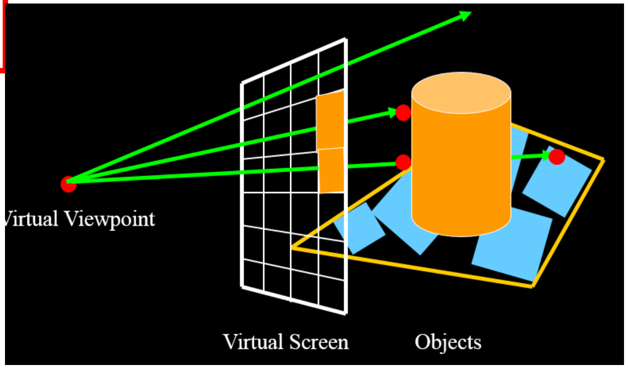
Ray Tracing Algorithm

```
Image Raytrace (Camera cam, Scene scene, int width, int height)
{
    Image image = new Image (width, height) ;
    for (int i = 0 ; i < height ; i++)
        for (int j = 0 ; j < width ; j++) {
            Ray ray = RayThruPixel (cam, i, j) ;
            Intersection hit = Intersect (ray, scene) ;
            image[i][j] = FindColor (hit) ;
            }
    return image ;
}
```



Ray Tracing Algorithm

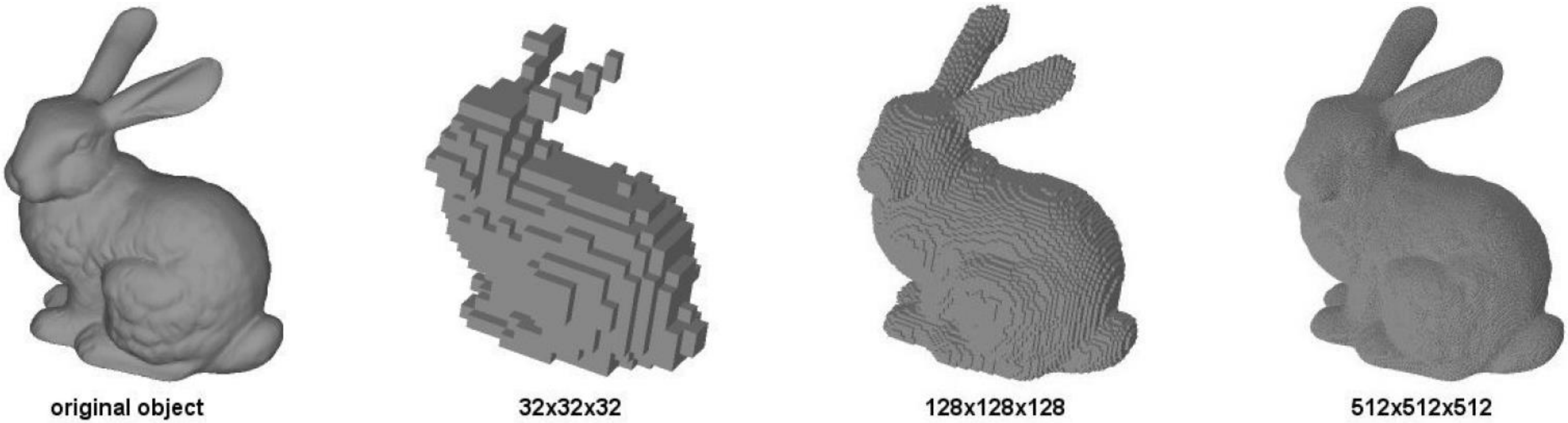
```
Image Raytrace (Camera cam, Scene scene, int width, int height)
{
    Image image = new Image (width, height) ;
    for (int i = 0 ; i < height ; i++)
        for (int j = 0 ; j < width ; j++) {
            Ray ray = RayThruPixel (cam, i, j) ;
            Intersection hit = Intersect (ray, scene) ;
            image[i][j] = FindColor (hit) ;
        }
    return image ;
}
```



Ray Tracing Accelaration

- Classification of Accelaration Techniques for ray tracing
  - Faster ray-object intersections
  - Fewer Ray-Object Intersections
  - Fewer Rays

What is voxelization?



Types of Voxelization

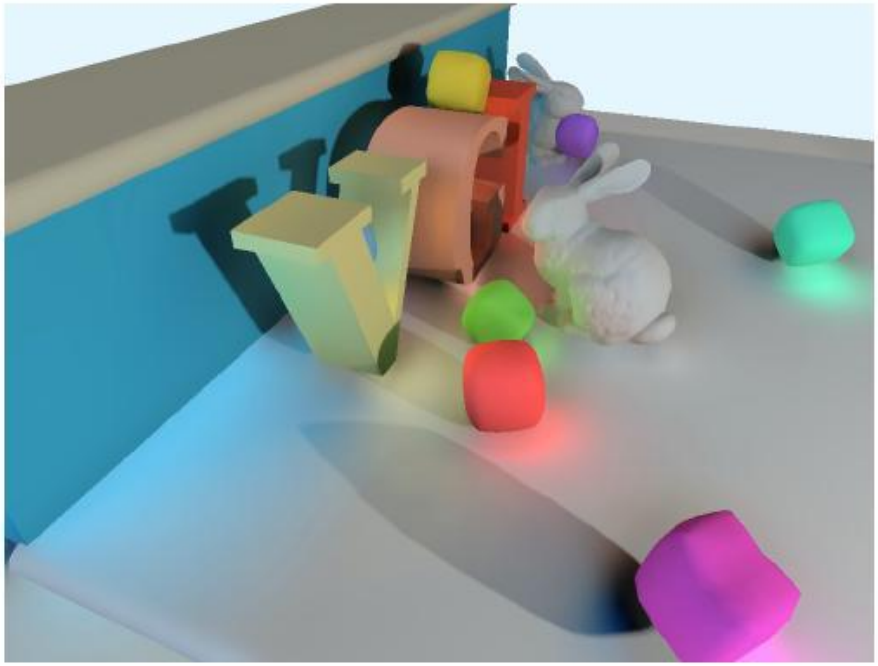
- Binary Voxelization
  - A cell stores whether geometry is present in this cell or not
- Multi-Valued Voxelization
  - A cell can also stores arbitrary other data like BRDF, normal or Radiance
- Boundary Voxelization
  - Encodes the object surfaces only
- Solid Voxelization
  - Captures the interior of a model

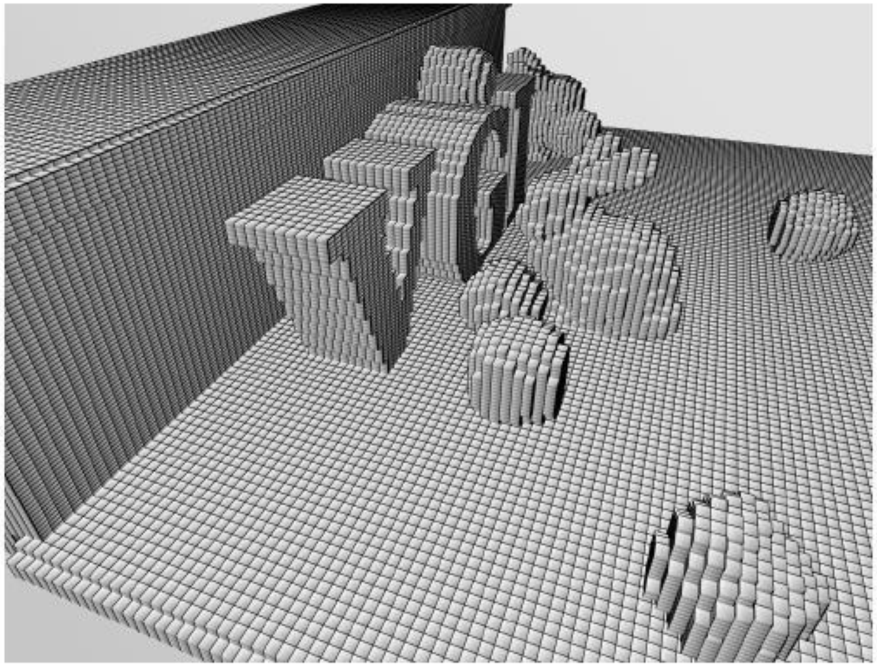
Contributions

- A new atlas-based voxelization method
- An improved ray/voxel intersection test
- Real-time near-field illumination with voxel visibility

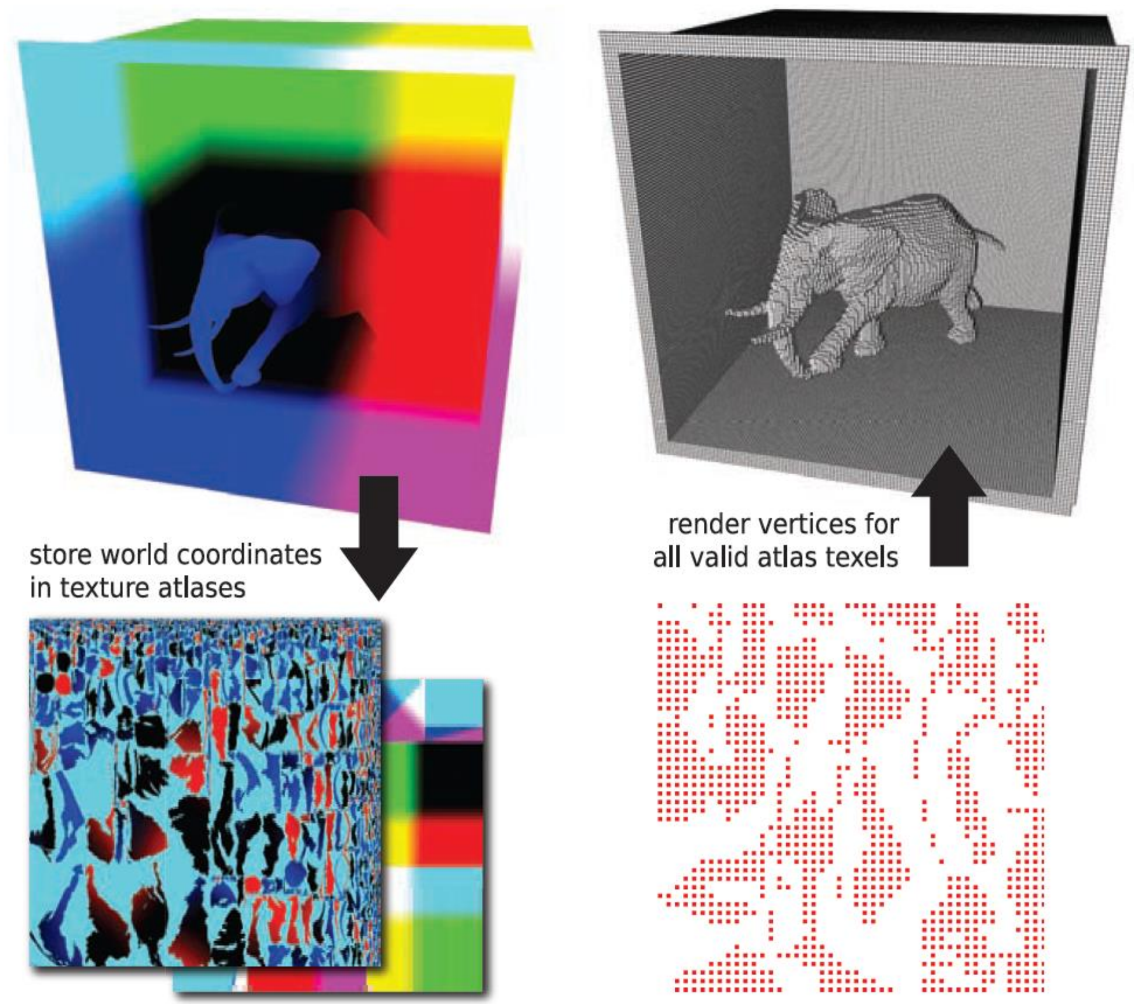
Goal

- Computing global illumination in real time, given a large and dynamic scene.





Atlas-Based Voxelization (1/5)

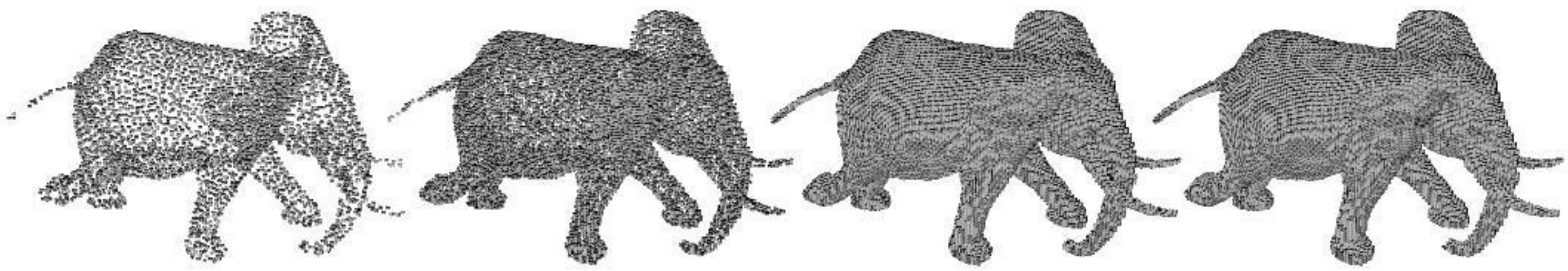


Atlas-Based Voxelization (2/5)

```
uniform sampler2D textureAtlas;
uniform mat4 viewProjMatrixVoxelCam;

varying float mappedZ;

void main ()
{
  // Incoming vertices have positions in the range
  // of [0..atlasWidth-1]x[0,atlasHeight-1].
  // Fetch world space position from atlas
  vec3 pos3D = texelFetch2D(textureAtlas,
                            ivec2(gl_Vertex.xy),0).rgb;

  // Transform into voxel grid coordinates
  gl_Position = viewProjMatrixVoxelCam * vec4(pos3D, 1.0);
}
```

Atlas-Based Voxelization (3/5)

- Performance is directly related to the number of rendered vertices
- Sufficient atlas-texture resolution



Atlas-Based Voxelization (4/5)

- Environment
  - Geforce GTX 295, Intel Core 2 Duo 3.16 Ghz, 4 GB RAM
- Performance

Voxel-grid resolution	Time (ms)	Vertices	Atlas resolution
$64^2 \times 128$	0.52	15k	176×176
$128^2 \times 128$	0.69	65k	368×368
$256^2 \times 128$	1.48	285k	768×768
$512^2 \times 128$	3.37	791k	1280×1280

Atlas-Based Voxelization (5/5)

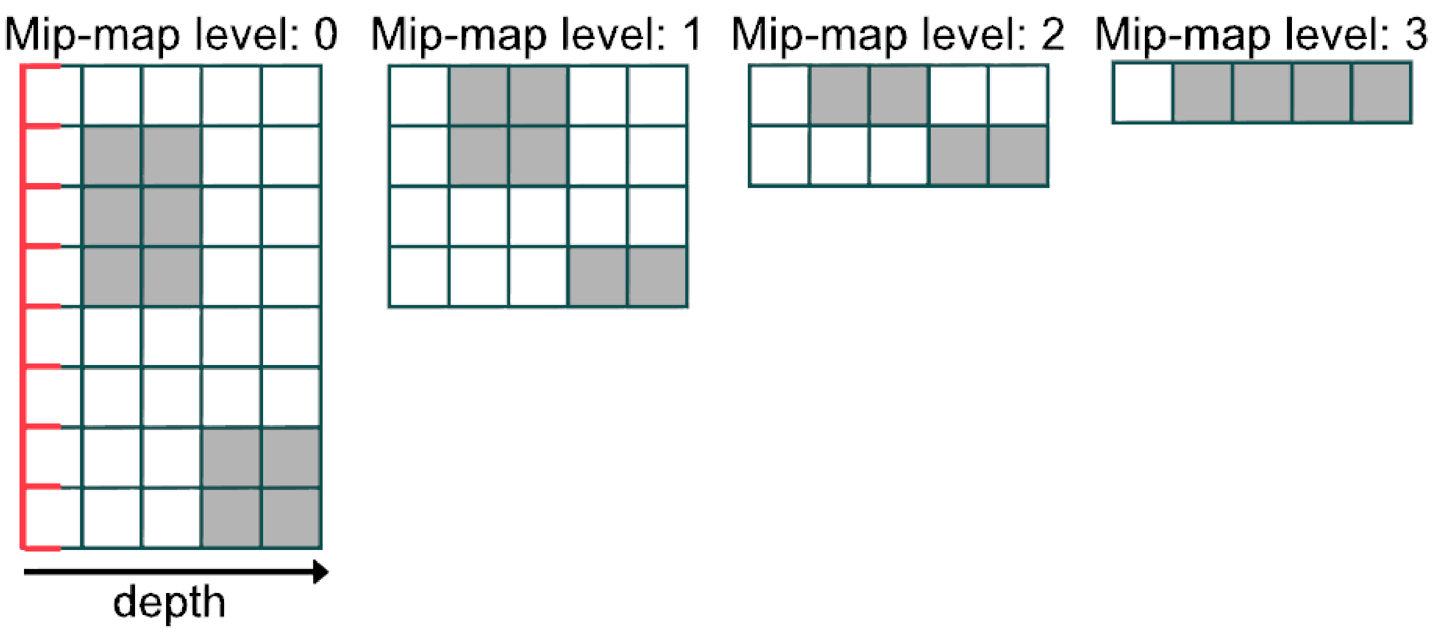
- Pros
  - No restrictions to the objects
  - Applicable for dynamics rigid bodies and moderately deforming models
- Cons
  - Each object has to have a texture atlas
  - Low atlas texture resolution causes holes in voxelization
  - Not allow strong deformation of the objects

Ray Tracing Accelaration

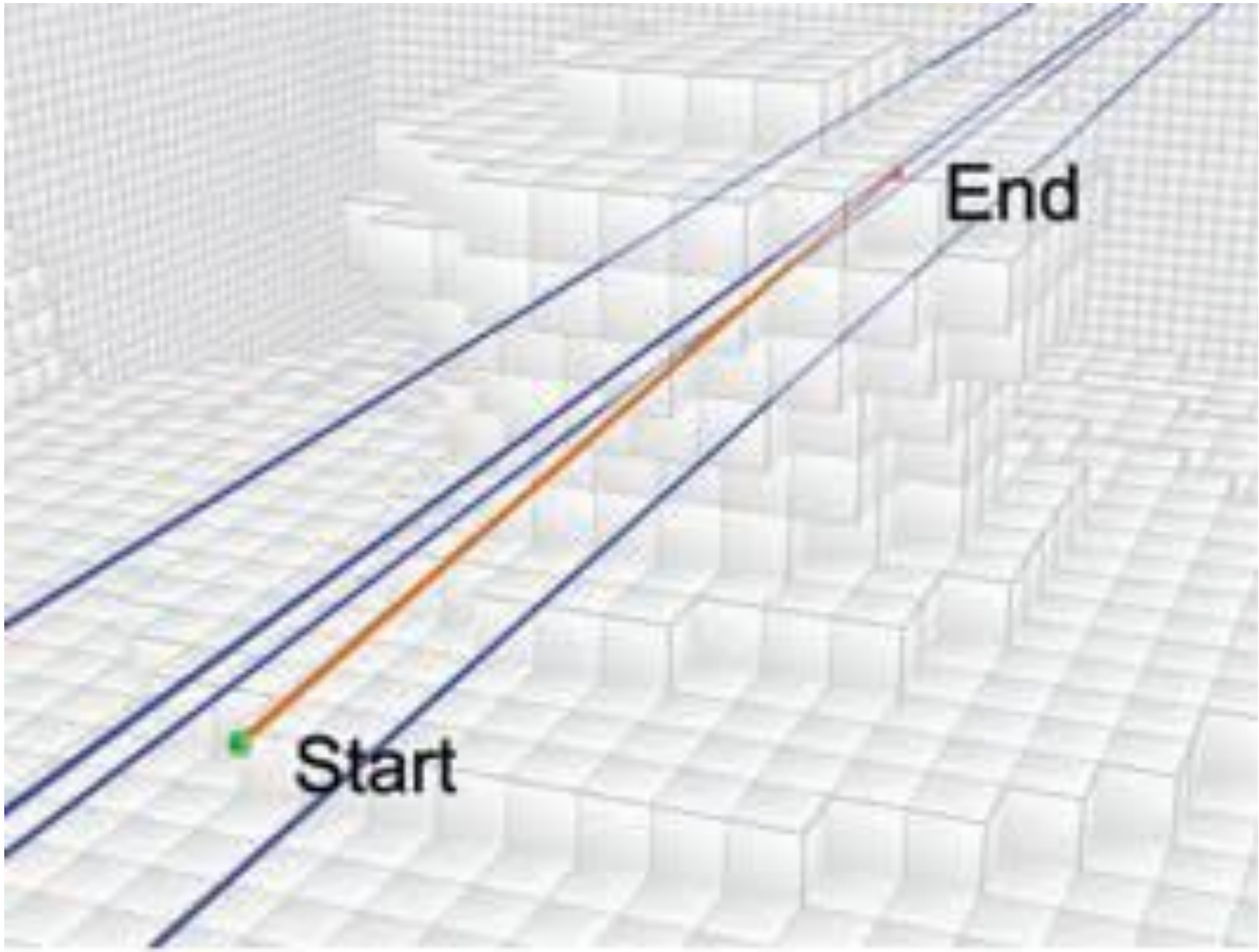
- Classification of Accelaration Techniques for ray tracing
  - **Faster ray-object intersections**
    - Efficient intersectors
  - Fewer Ray-Object Intersections
    - Bounding Volumes (Boxes, Spheres)
    - Space Subdivision
  - Fewer Rays
    - Adaptive Tree-Depth Control
    - Stochastic Sampling

Hierarchical Ray-Voxel Intersection Test (1/3)

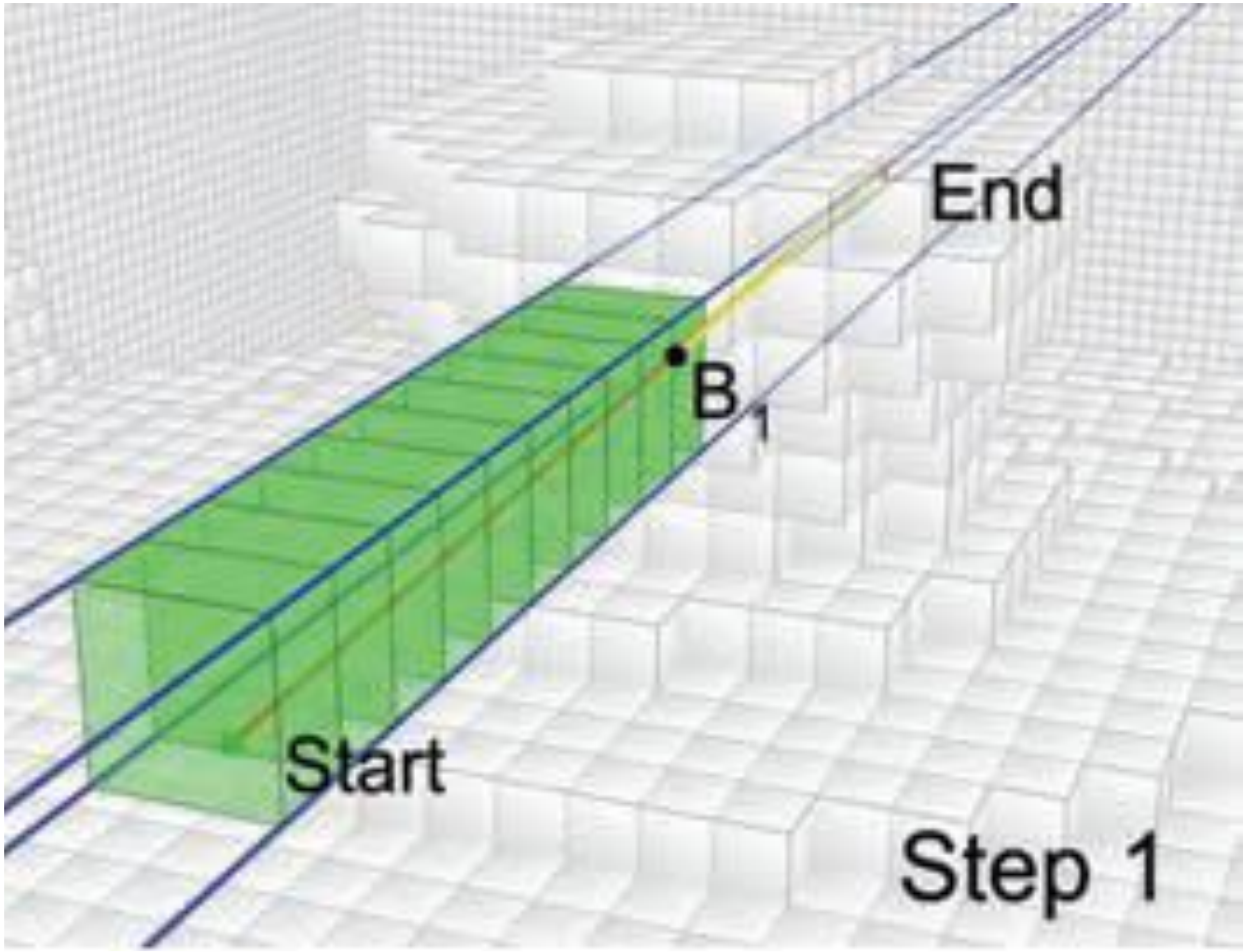
- Use a binary voxelized scene representation(Why only in 2-dimensions)
- Build a Maximum mip-map hierarchy



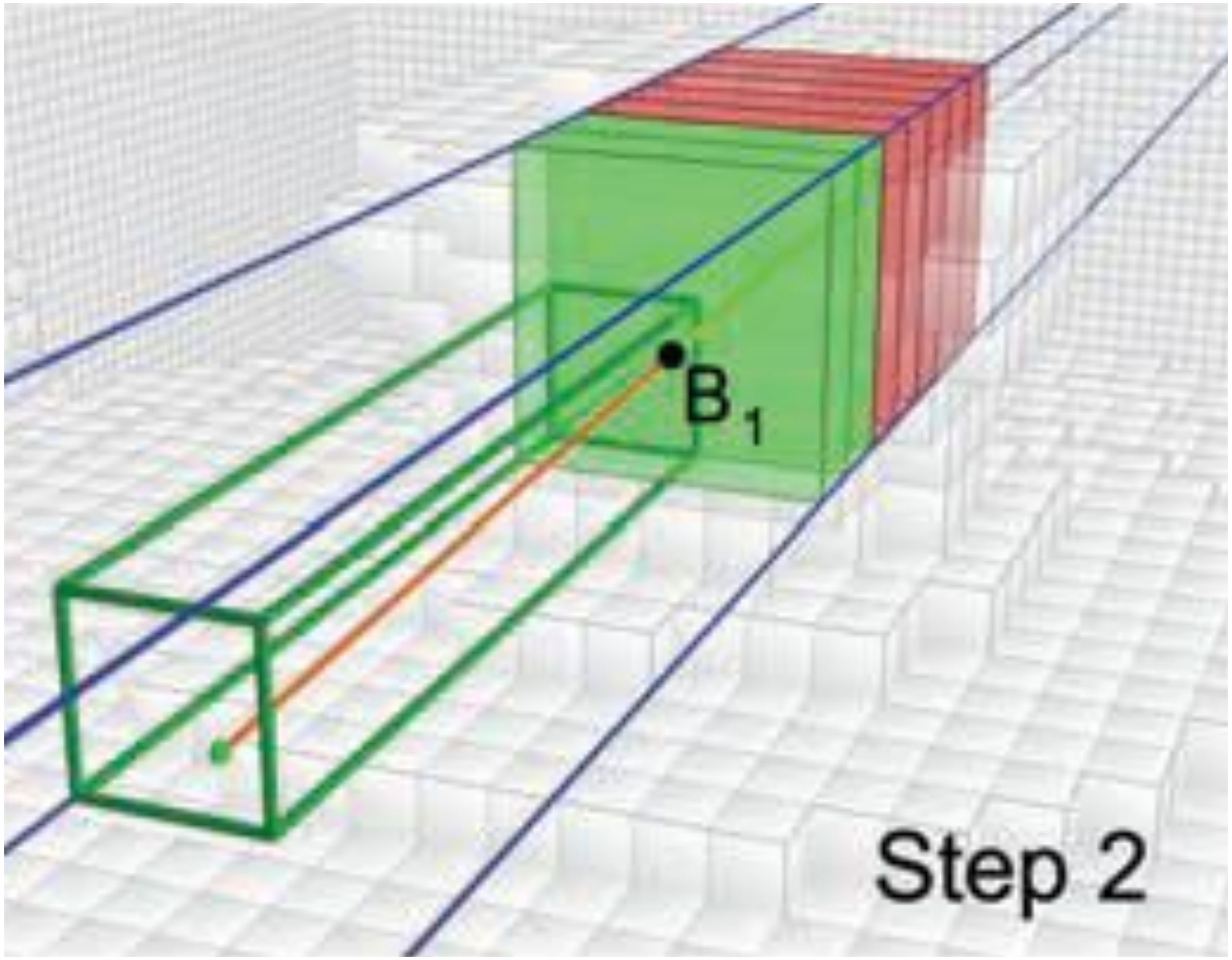
Hierarchical Ray-Voxel Intersection Test (2/3)



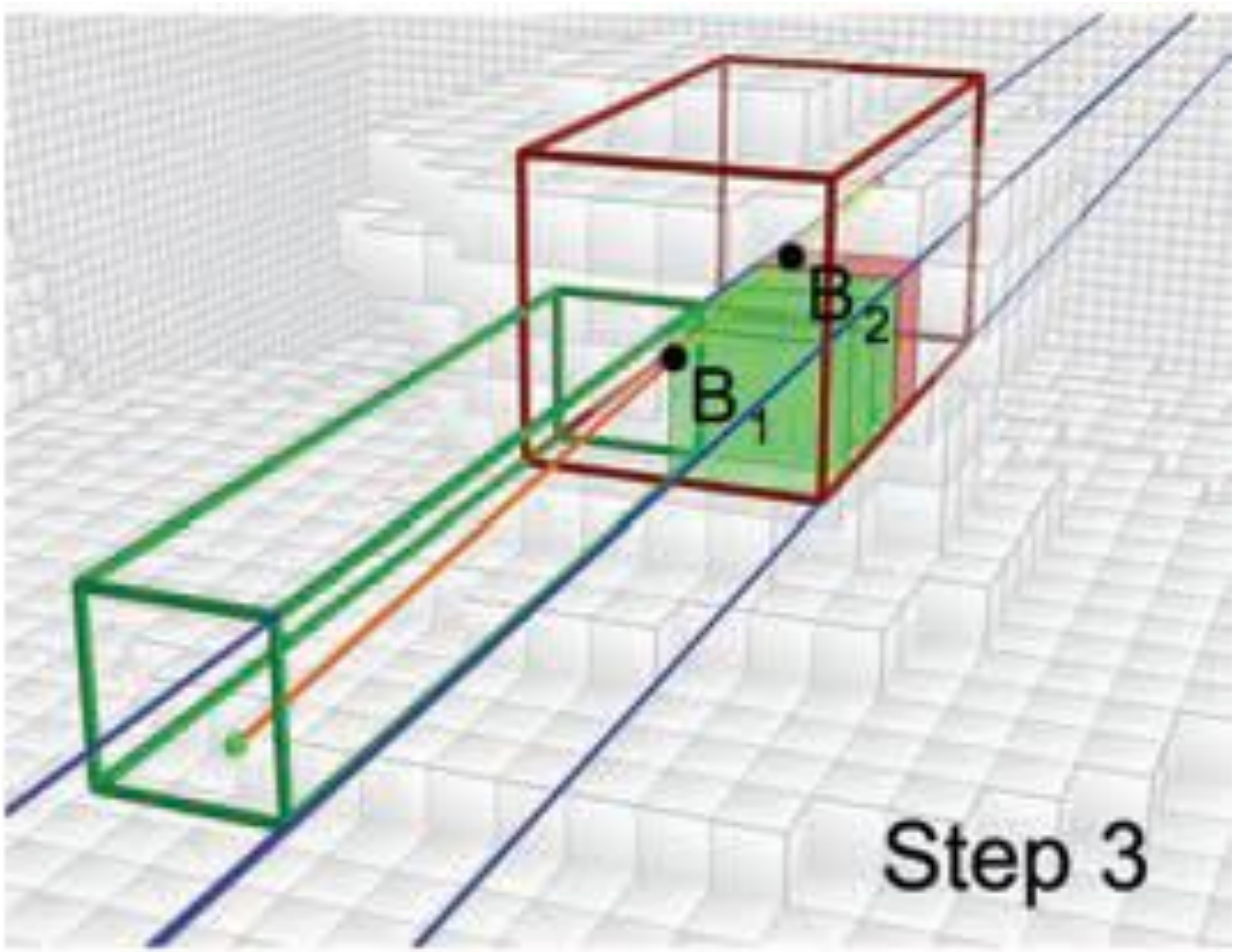
Hierarchical Ray-Voxel Intersection Test (2/3)



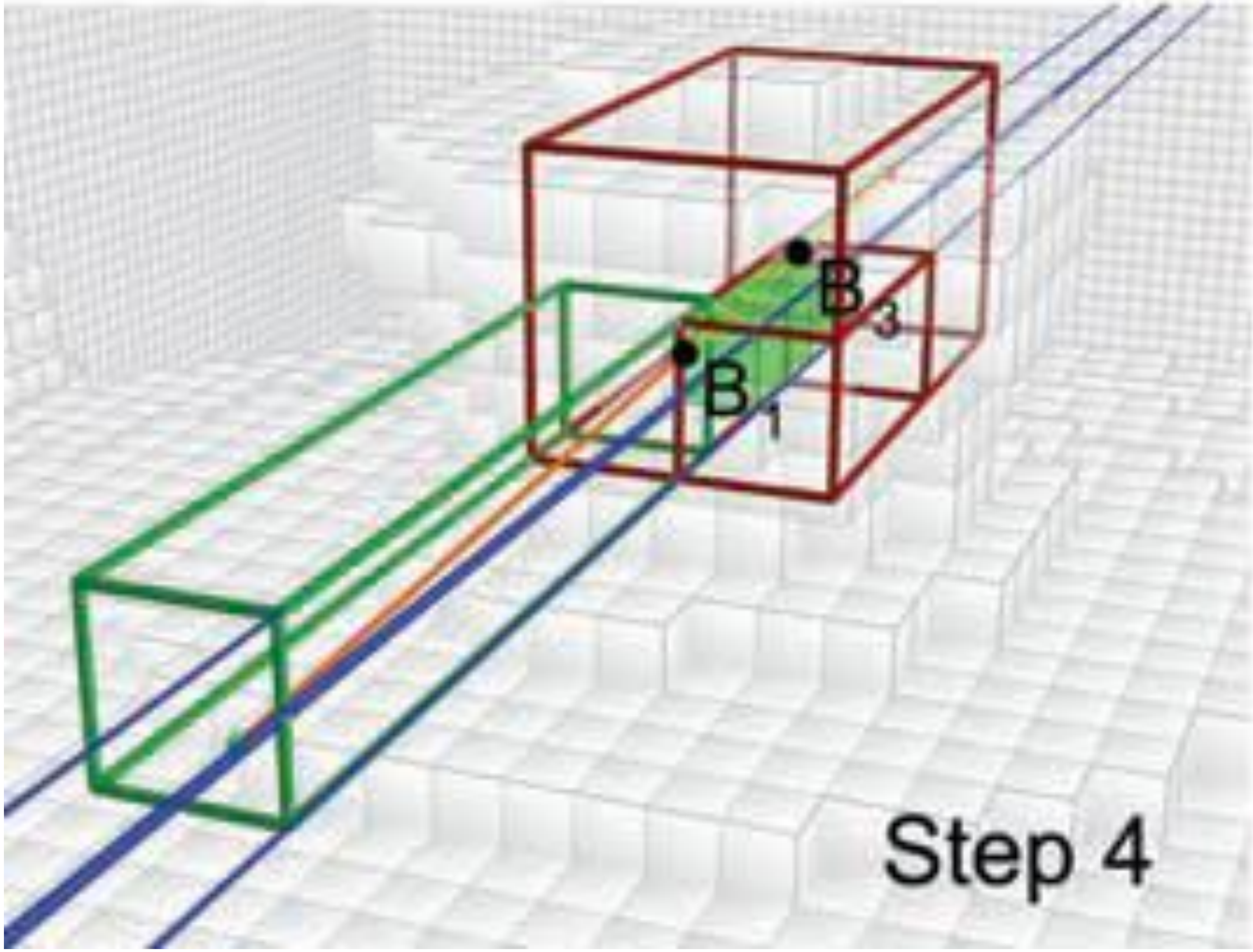
Hierarchical Ray-Voxel Intersection Test (2/3)



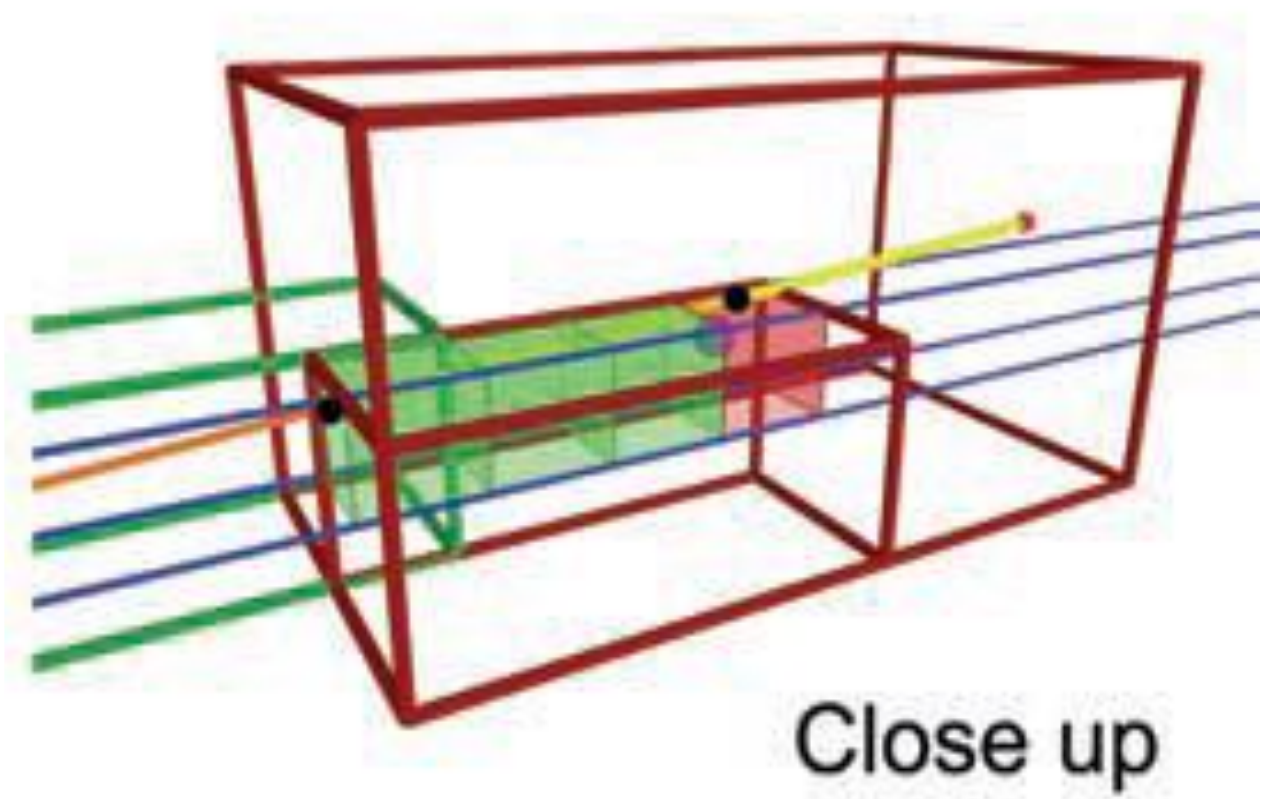
Hierarchical Ray-Voxel Intersection Test (2/3)



Hierarchical Ray-Voxel Intersection Test (2/3)

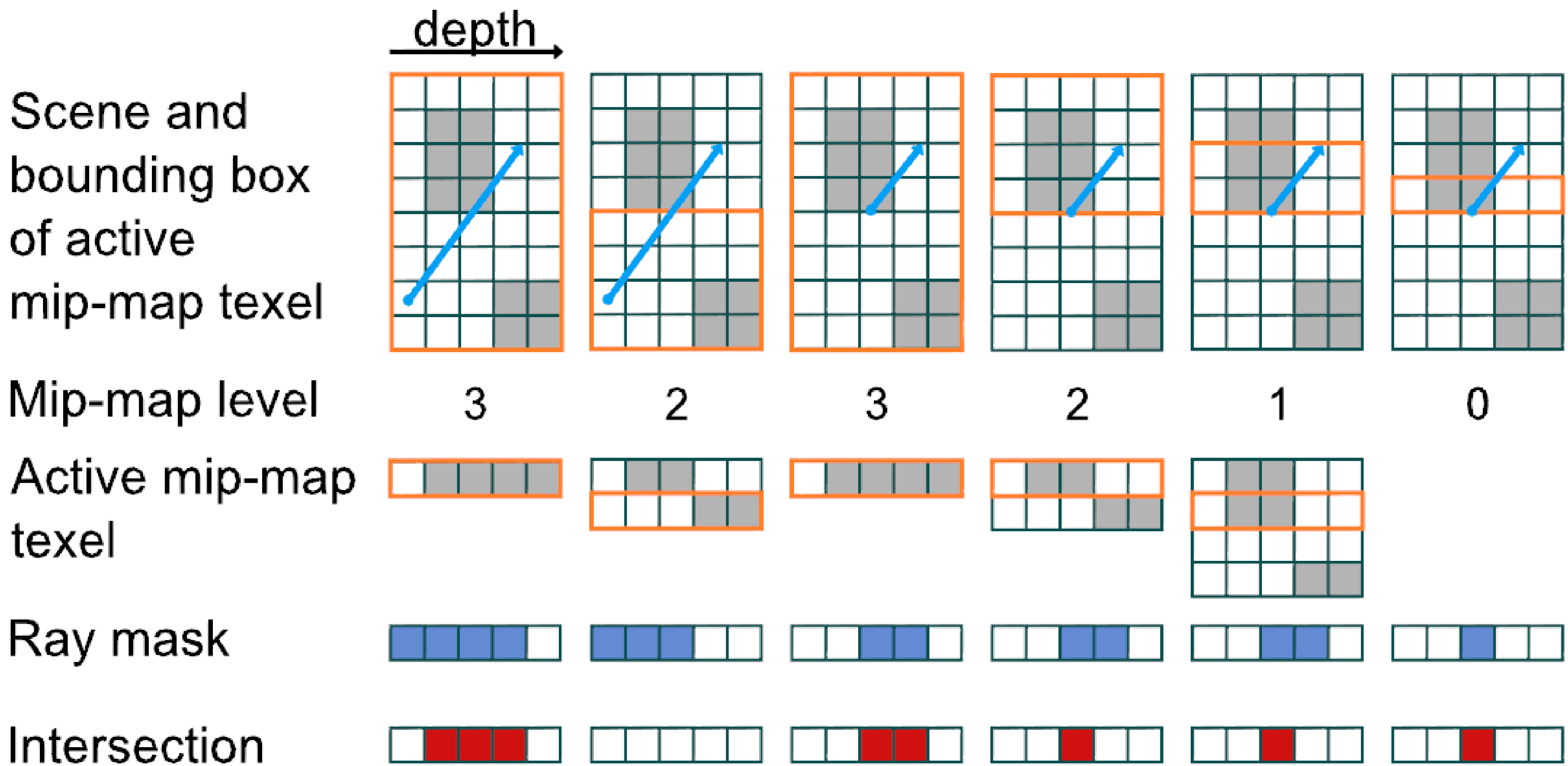


Hierarchical Ray-Voxel Intersection Test (2/3)



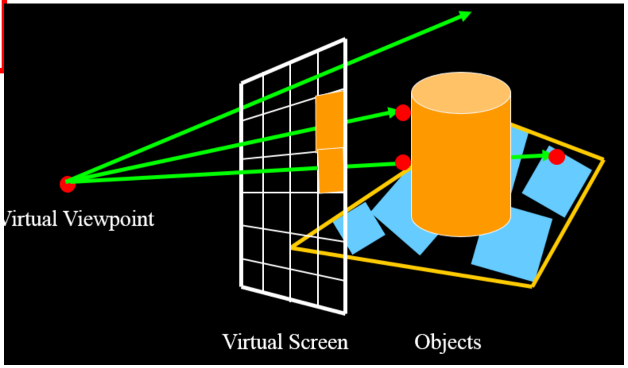
Hierarchical Ray-Voxel Intersection Test (3/3)

- Algorithm



Ray Tracing Algorithm

```
Image Raytrace (Camera cam, Scene scene, int width, int height)
{
    Image image = new Image (width, height) ;
    for (int i = 0 ; i < height ; i++)
        for (int j = 0 ; j < width ; j++) {
            Ray ray = RayThruPixel (cam, i, j) ;
            Intersection hit = Intersect (ray, scene) ;
            image[i][j] = FindColor (hit) ;
            }
    return image ;
}
```



computer graphics & visualization

Outgoing Radiance

$$L_o(p,\omega_o) = \int_{2\pi^+} \boxed{f_r(p,\omega_i,\omega_o)}\, L_i(p,\omega_i) \cos\theta_i \, d_{\omega_i}$$

BRDF

Outgoing Radiance

$$L_o(p, \omega_o) = \int_{2\pi^+} \boxed{f_r(p, \omega_i, \omega_o)} L_i(p, \omega_i) \cos \boxed{\theta_i} \boxed{d_{\omega_i}}$$

BRDF

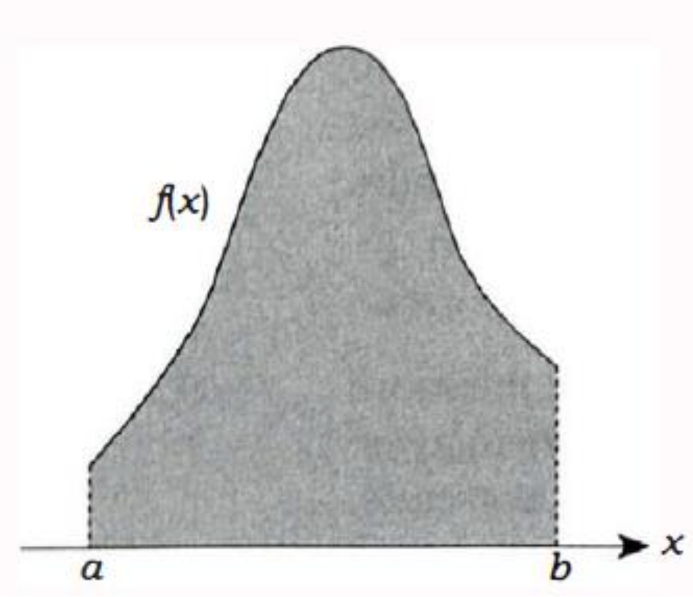
Outgoing Direction

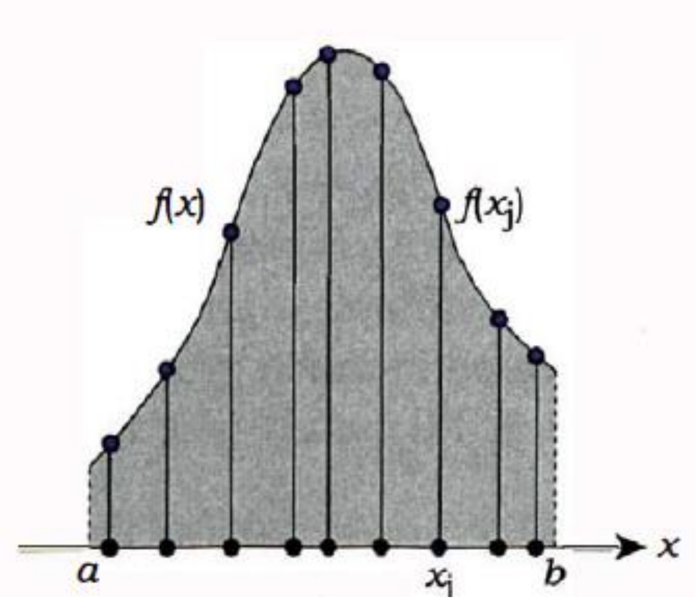
Incoming Direction

Monte Carlo Estimator

$$I = \int_a^b f(x)dx$$

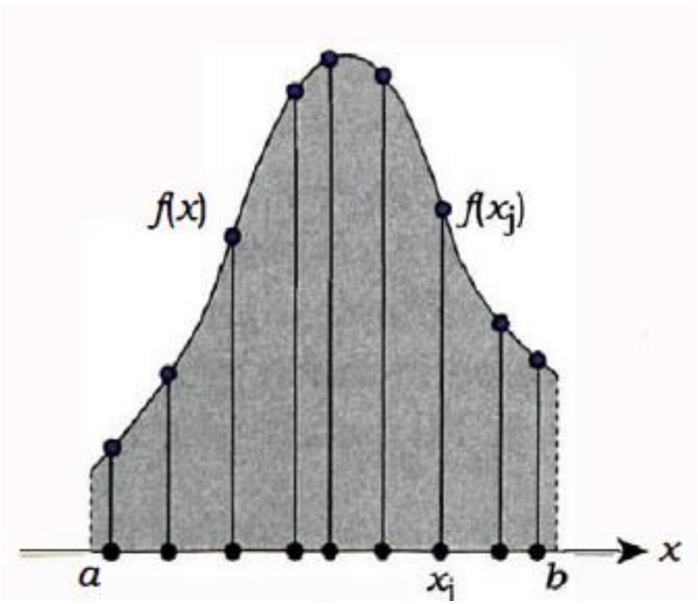
Monte Carlo Estimator: $\langle I \rangle = \frac{b-a}{n}\sum_{j=1}^{n} f(x_j)$

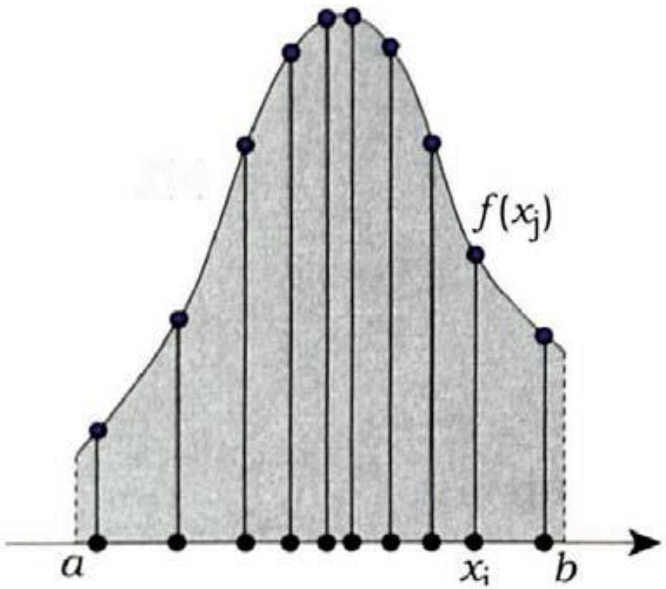




Monte Carlo Importance Sampling

Monte Carlo Estimator: $\langle I \rangle = \frac{b-a}{n} \sum_{j=1}^{n} \frac{f(x_j)}{p(x_j)}$



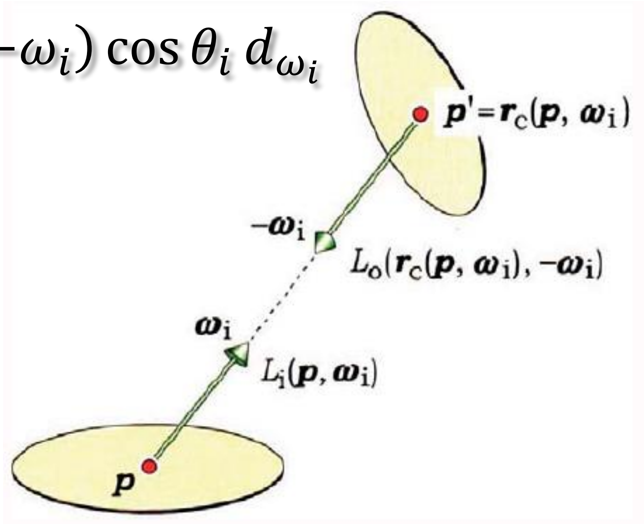


Outgoing Radiance

$$L_o(p,\omega_o) = \int_{2\pi^+} f_r(p,\omega_i,\omega_o)\, L_i(p,\omega_i) \cos\theta_i \, d_{\omega_i}$$

$$L_i(p,\omega_i) = L_o(r(p,\omega_i), -\omega_i)$$

$$L_o(p,\omega_o) = \int_{2\pi^+} f_r(p,\omega_i,\omega_o)\, L_o(r(p,\omega_i), -\omega_i) \cos\theta_i \, d_{\omega_i}$$



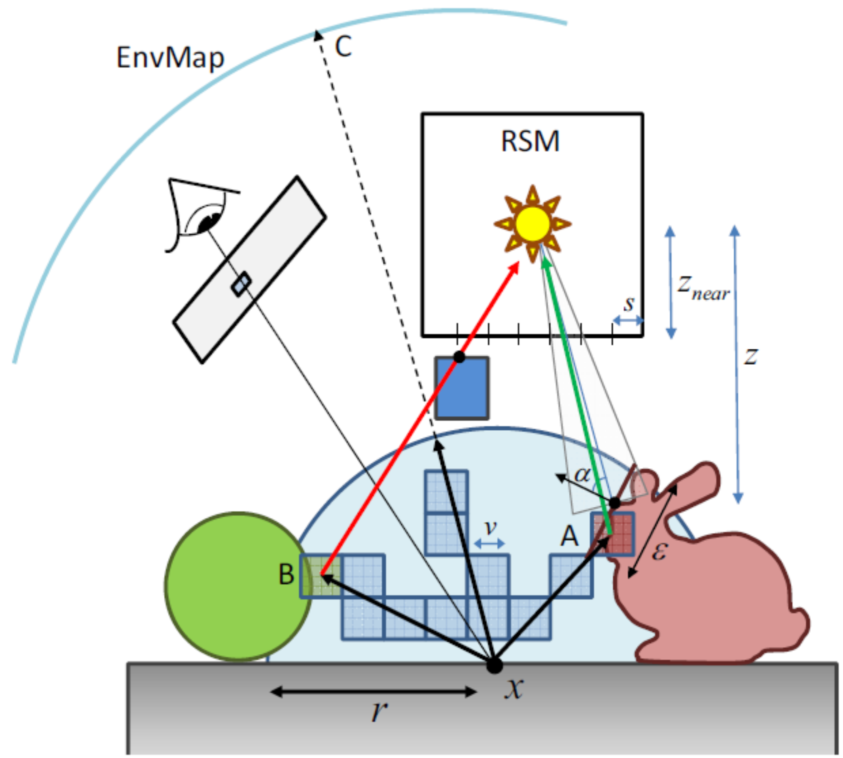
One Bounce is enough!

(a) using a single bounce of indirect light

(b) using multiple bounces of indirect light

Real-Time Near-Field Single Bounce Indirect Light

- Generate RSM (Reflective Shadow Map) for fast near-field illumination
- Shoot N rays from x with maximum distance r
- Find first intersection point using binary voxelization
- Gather direct radiance from RSM

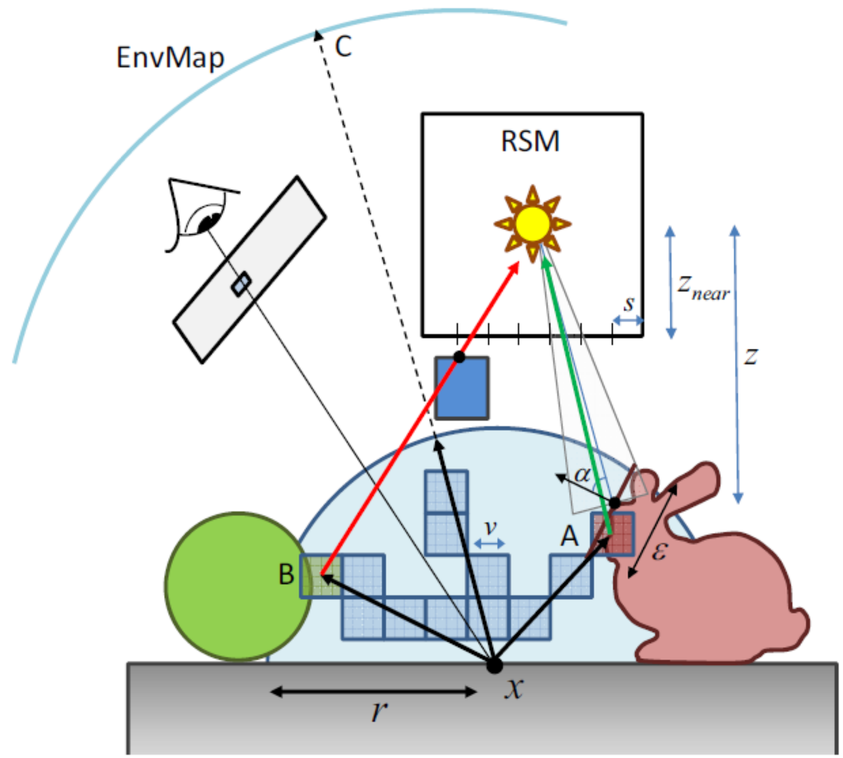


Real-Time Near-Field Single Bounce Indirect Light

$$L_o(\mathbf{x}) \approx \frac{\rho(\mathbf{x})/\pi}{N} \sum_{i=1}^{N} \frac{\tilde{L}_i(\mathbf{x}, \omega_i) \cos\theta}{p(\omega_i)},$$

$$p(\omega_i) = \cos\theta/\pi$$

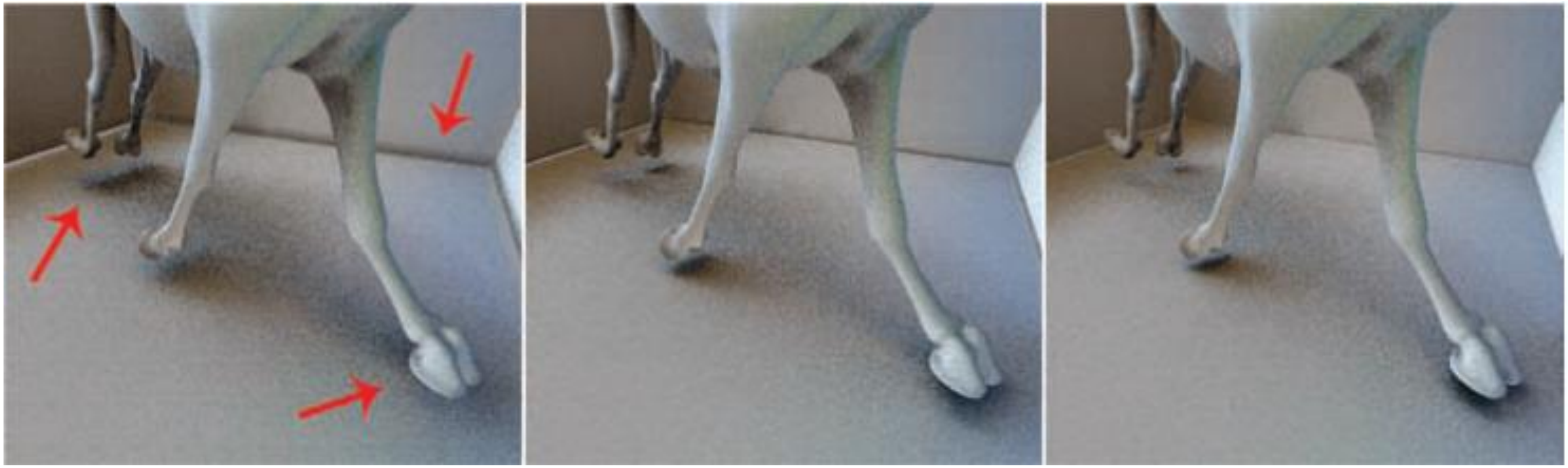
$$L_o(\mathbf{x}) \approx \frac{\rho(\mathbf{x})}{N} \sum_{i=1}^{N} \tilde{L}_i(\mathbf{x}, \omega_i)$$



Results (movie)

Results

- The effect of changing the voxel resolution



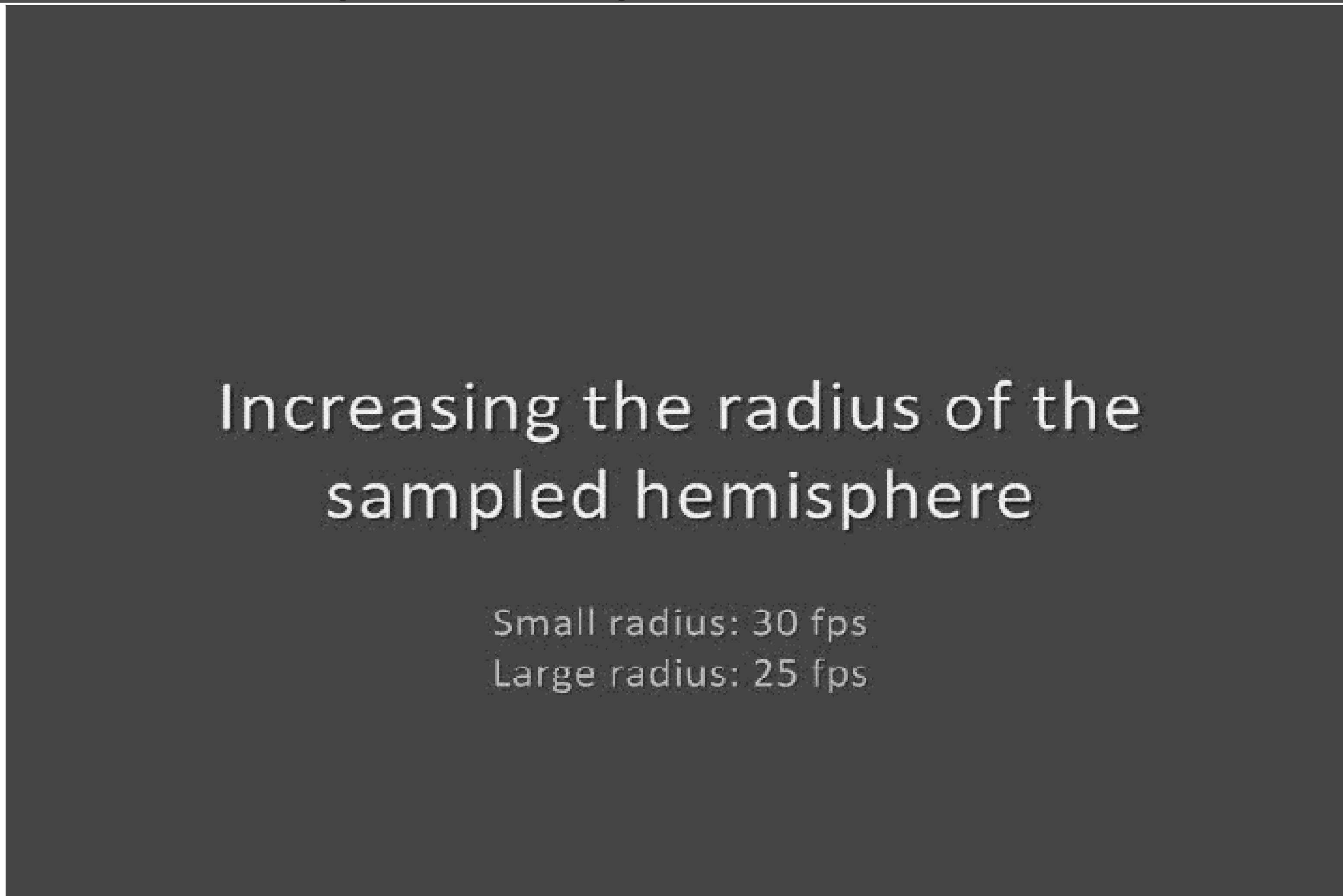
32x32x128 (47fps) **64x64x128 (36fps)** **128x128x128 (25fps)**

Results

- Voxel-Based Single Bounce illumination with different Radiuses R

R = 1.5 (37 fps) **R = 4.0 - center (27 fps)** **R = 4.0 – right (21 fps)**

Results 2 (Movie)



Thank you very much!!! :D

Questions?

References

Reference Name	Author
TUM’s Computer Graphics Course Slides – 2013	Prof. Westermann, TU-Munich
Ray Tracing from Group Up, 2007 by AK Peters Ltd	Kevvin Suffern
GPU Pro 3, 2012 by CRC Press	Wolfgang Engel
An Approximate Global-Illumination System for Computer Generated Films	Tabellion, E. And Lamorlette, A.
Voxel-Based Global Illumination, *Proceedings of ACM Symposium on Interactive 3D Graphics and Games 2011 (I3D'11)*	Thiedemann, Henrich, Grosch, and Müller
Voxel-Based Global Illumination slides at Computer Graphics and Image Processing Lab, SNU http://graphics.snu.ac.kr/class/graphics2011/materials/paper09_voxel_gi.pdf	Jin Hur, Junhyuk Yoon