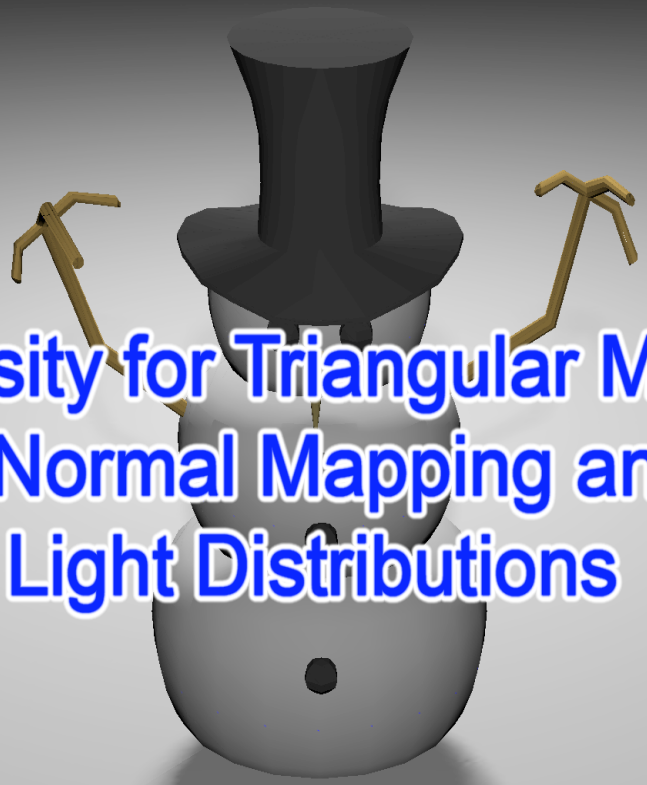


GPU Radiosity for Triangular Meshes with Support of Normal Mapping and Arbitrary Light Distributions

Günter Wallner

University of Applied Arts, Vienna

16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, 2008

di:'ʌngewʌndtə

Introduction

- GOAL: Progressive radiosity on the GPU
- Radiosity is well suited to the SIMD architecture of GPUs [Coombe]
- Textures provide discretization needed for radiosity
- 32bit floating point textures provide necessary precision

[Coombe] G. Coombe et al., Radiosity on graphics hardware, Technical report, Univ. of North Carolina, 2003

Implementation

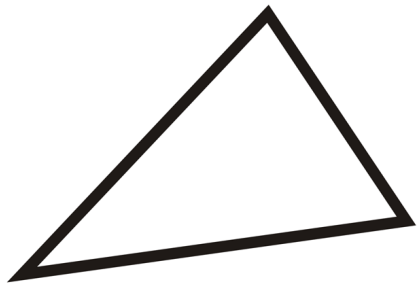
Algorithm Outline

pre-processing

```
while not converged {
        select next shooter
        render visibility texture
        (adaptive subdivision)
        for (every receiver texel) {
                determine visibility
                update texture
        }
}
post-processing
tone-mapping
```

Implementation

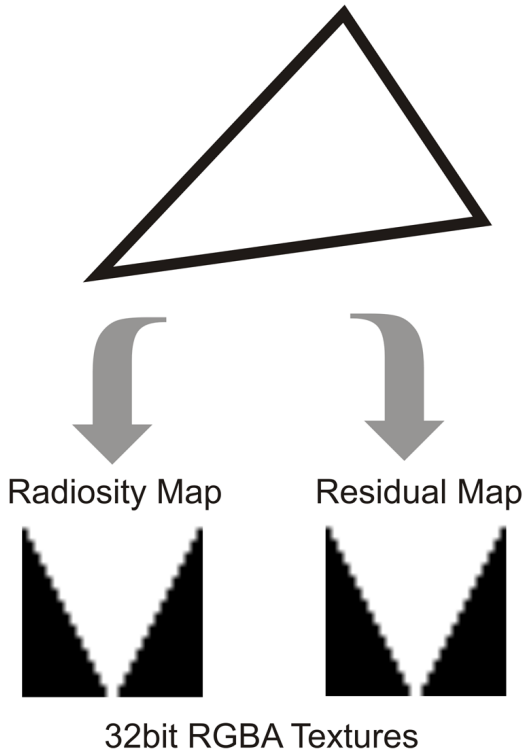
Preprocessing



- render each triangle orthographically into a framebuffer
- issue occlusion query to count occupied texels
- calculate area of element as $\frac{area_{triangle}}{pixels_{occupied}}$

Implementation

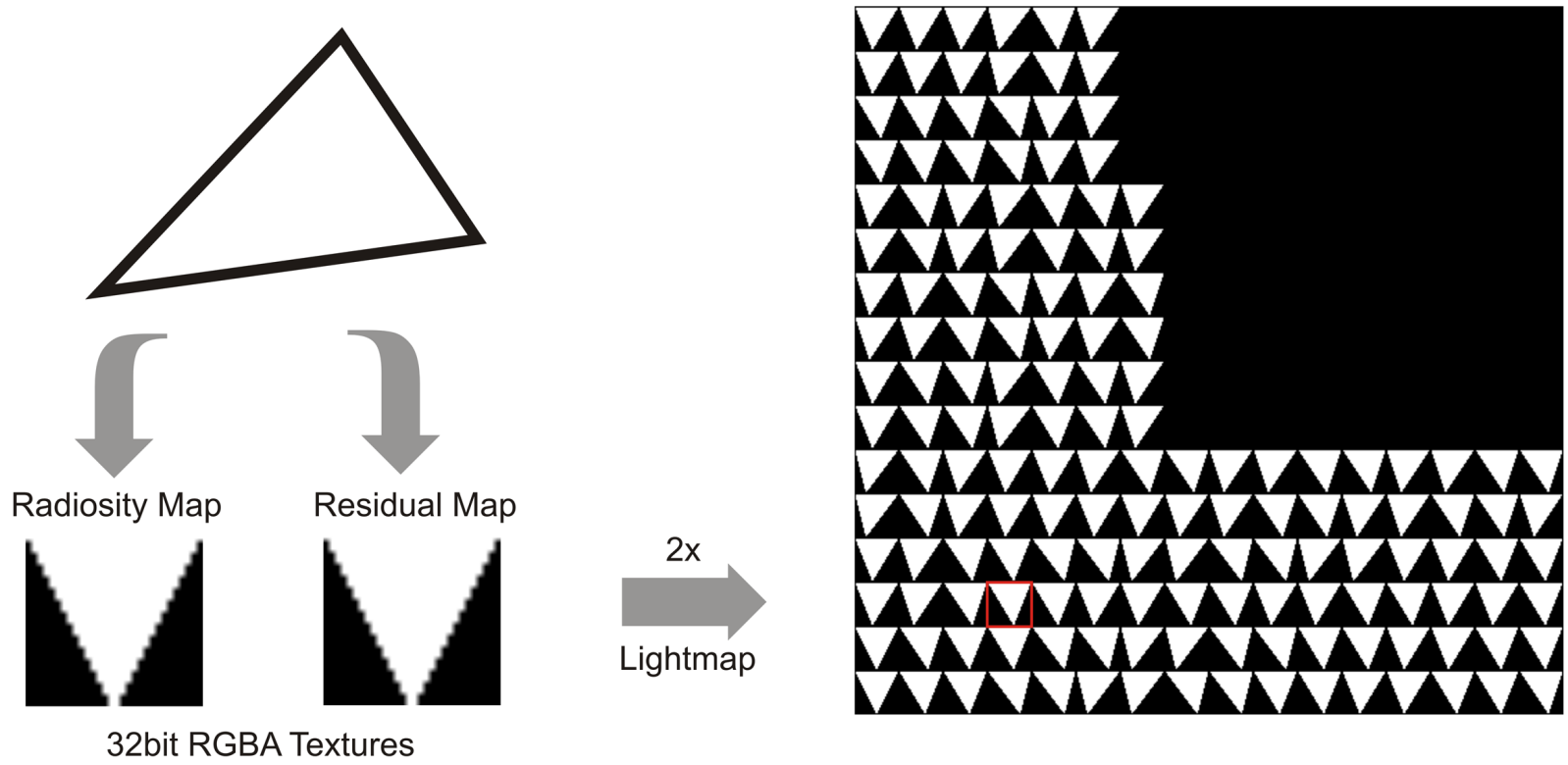
Preprocessing



- render each triangle orthographically into a framebuffer
- issue occlusion query to count occupied texels
- calculate area of element as $\frac{area_{triangle}}{pixels_{occupied}}$

Implementation

Preprocessing



- render each triangle orthographically into a framebuffer
- issue occlusion query to count occupied texels
- calculate area of element as $\frac{area_{triangle}}{pixels_{occupied}}$

Implementation

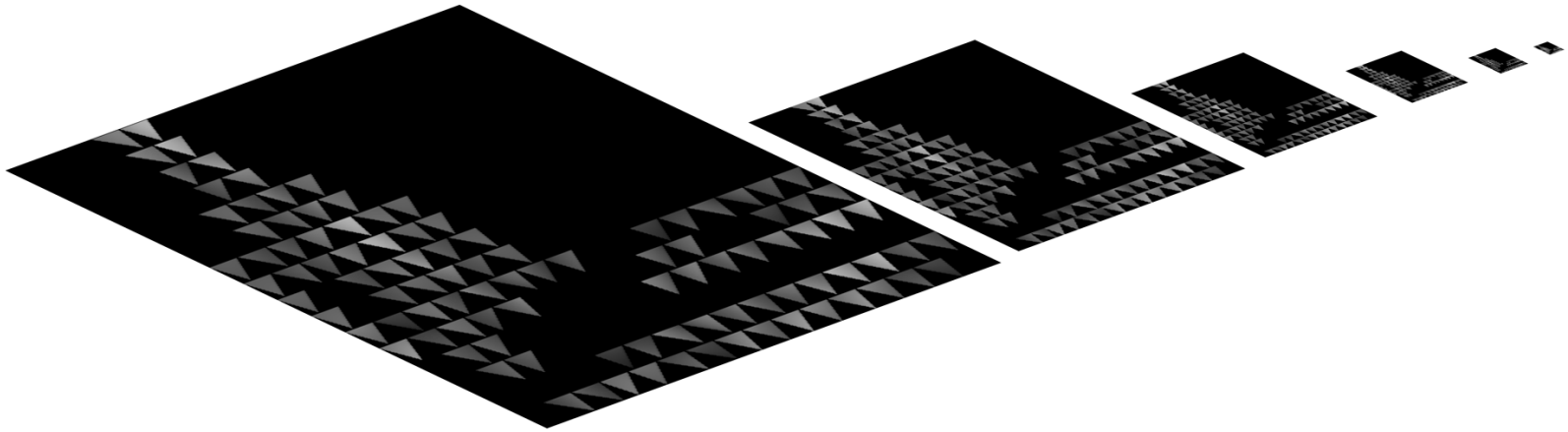
Algorithm Outline

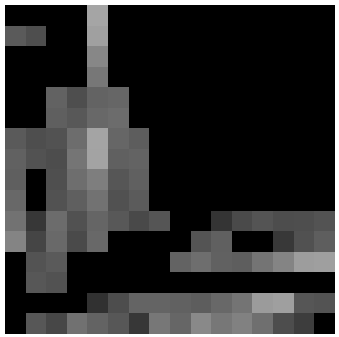
```
pre-processing
while not converged {
        select next shooter
        render visibility texture
        (adaptive subdivision)
        for (every receiver texel) {
                determine visibility
                update texture
        }
}
post-processing
tone-mapping
```

Implementation

Next Shooter Selection

Construct mipmap pyramid from each residual lightmap





read back the values

$power = intensity * area$

Ambient Radiosity Term for free

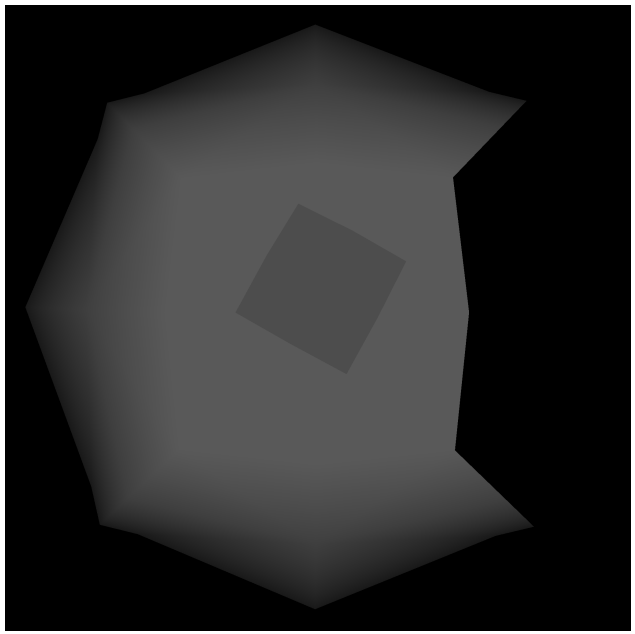
Implementation

Algorithm Outline

```
pre-processing
while not converged {
        select next shooter
        render visibility texture
        (adaptive subdivision)
        for (every receiver texel) {
                determine visibility
                update texture
        }
}
post-processing
tone-mapping
```

Implementation

Visibility Texture



- Stereographic Projection instead of Hemicube
- only vertices are affected
- issue occlusion query for each triangle

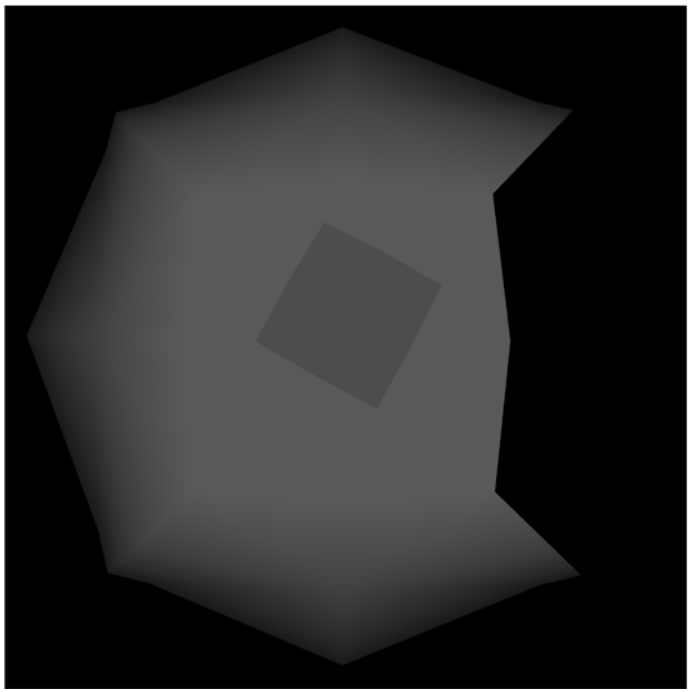
Implementation

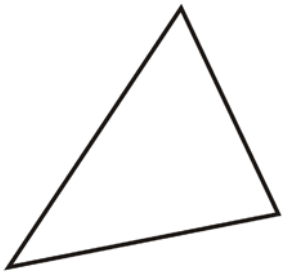
Algorithm Outline

```
pre-processing
while not converged {
        select next shooter
        render visibility texture
        (adaptive subdivision)
        for (every receiver texel) {
                determine visibility
                update texture
        }
}
post-processing
tone-mapping
```

Implementation

Visibility Determination

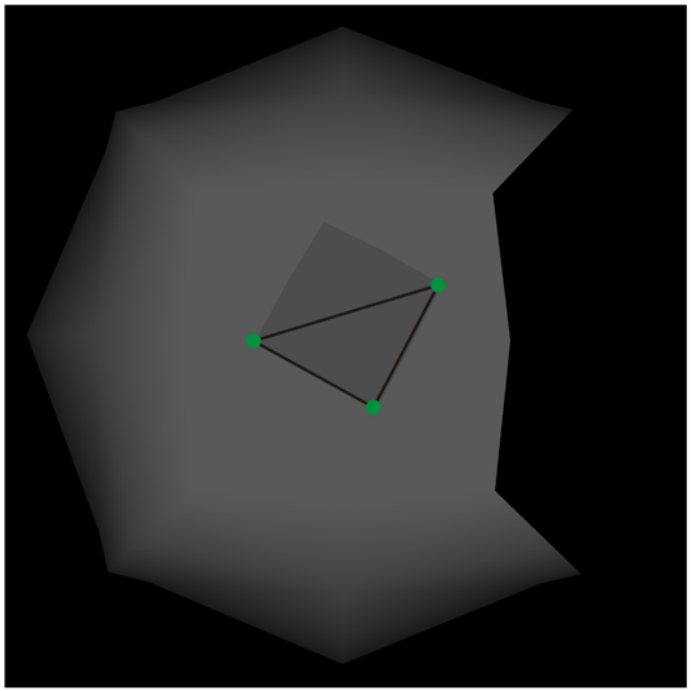


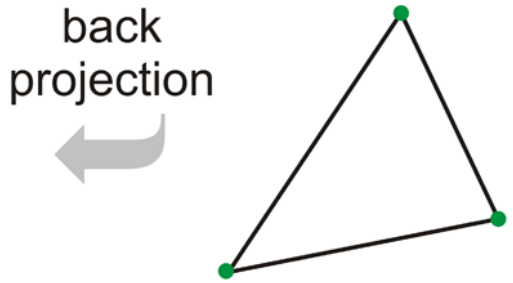


- Perform back-projection in vertex shader
- Two wrongs make a right

Implementation

Visibility Determination

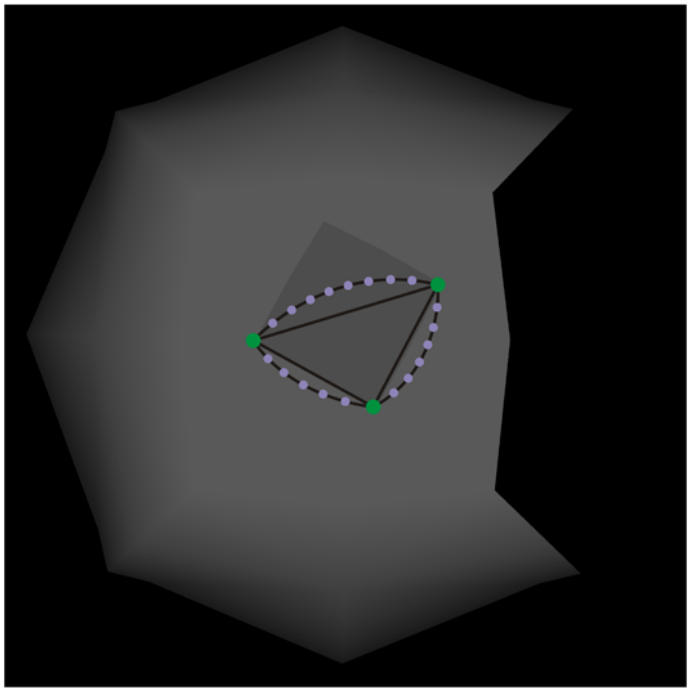


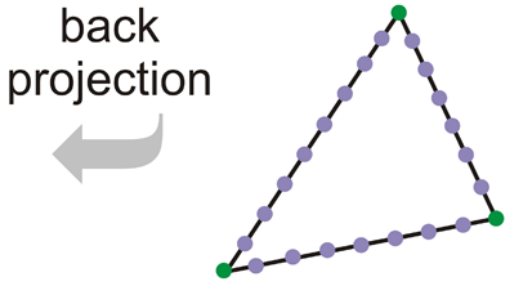


- Perform back-projection in vertex shader
- Two wrongs make a right

Implementation

Visibility Determination





- Perform back-projection in vertex shader
- Two wrongs make a right

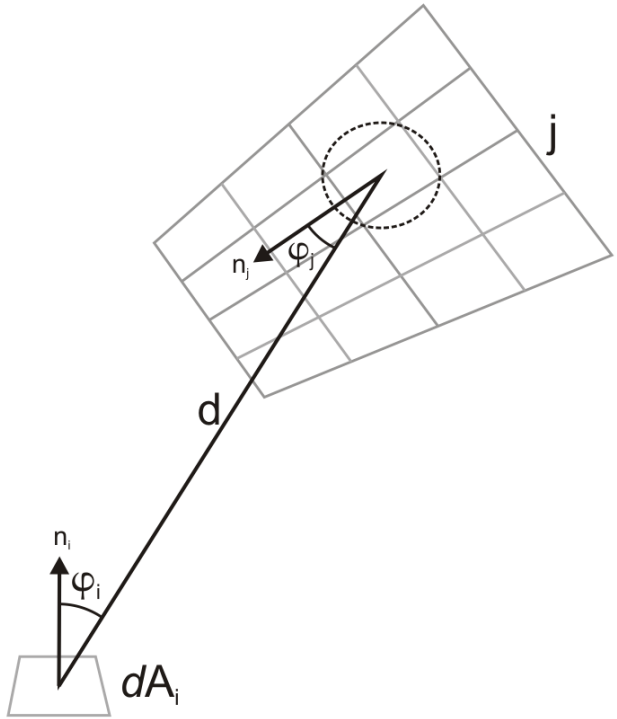
Implementation

Update Textures

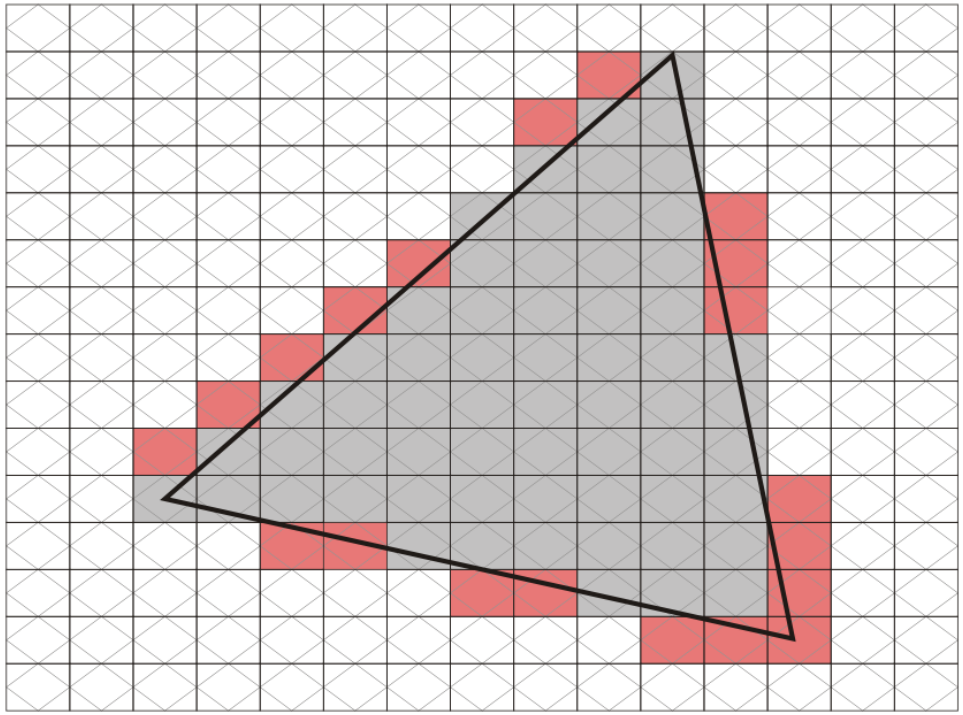
If texel is visible

- compute form factor *ff* between shooter *S* and receiver *R*
- update radiosity and residual texture of *R*
- (2 drawbuffers allow for simultaneous update)

set energy of *S* to zero

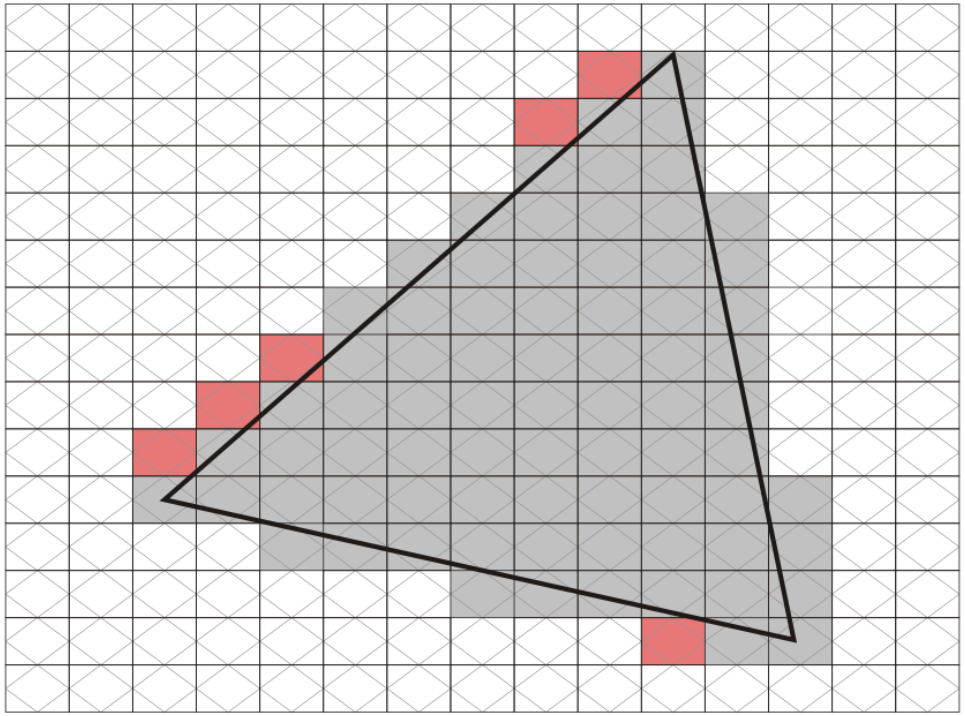


Rasterization of Triangles



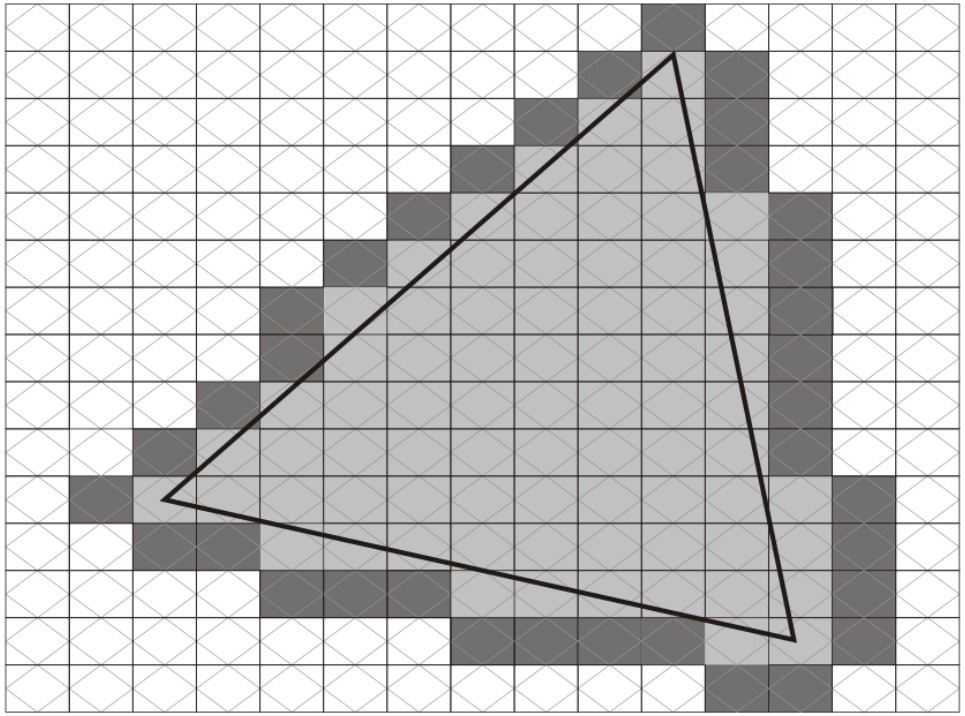
- point-sampling rule
- diamond-exit rule
- interpolation 1st time (post-process)
- interpolation 2nd time (post-process)

Rasterization of Triangles



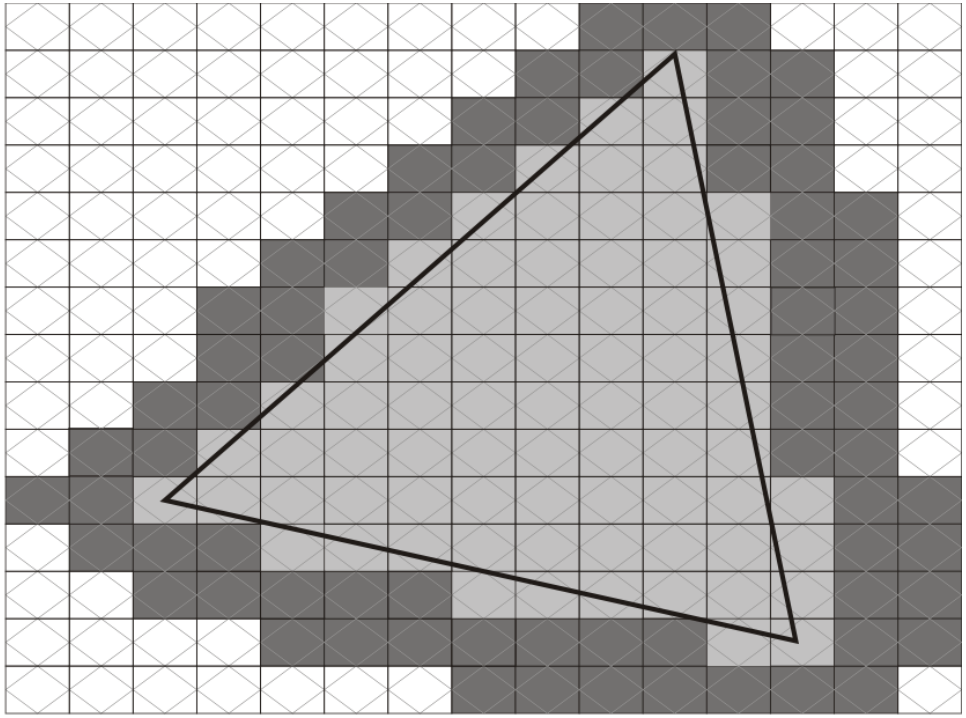
- point-sampling rule
- diamond-exit rule
- interpolation 1st time (post-process)
- interpolation 2nd time (post-process)

Rasterization of Triangles



- point-sampling rule
- diamond-exit rule
- interpolation 1st time (post-process)
- interpolation 2nd time (post-process)

Rasterization of Triangles



- point-sampling rule
- diamond-exit rule
- interpolation 1st time (post-process)
- interpolation 2nd time (post-process)

Implementation

Algorithm Outline

```
pre-processing
while not converged {
        select next shooter
        render visibility texture
        (adaptive subdivision)
        for (every receiver texel) {
                determine visibility
                update texture
        }
}
post-processing
tone-mapping
```

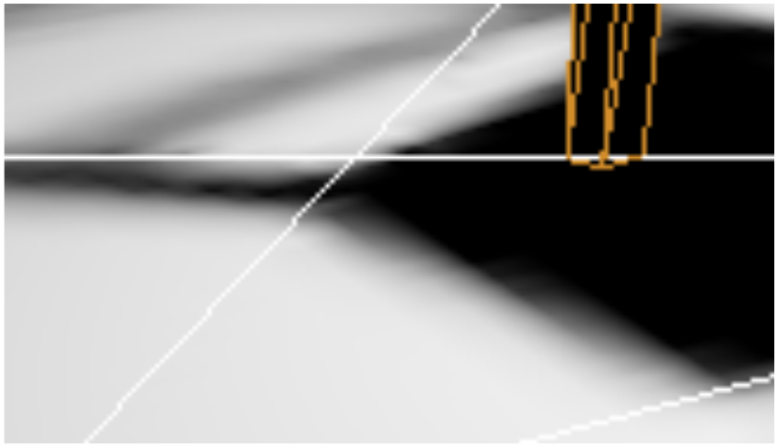
Adaptive Subdivision

Improves Image Quality

Adaptive Subdivision

Improves Image Quality

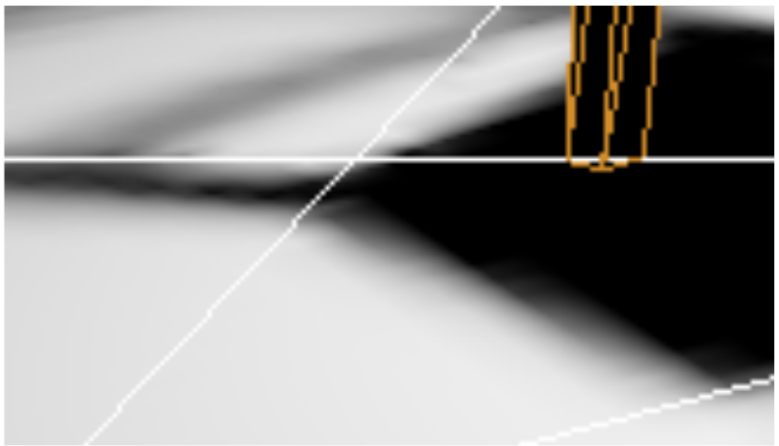
without subdivison



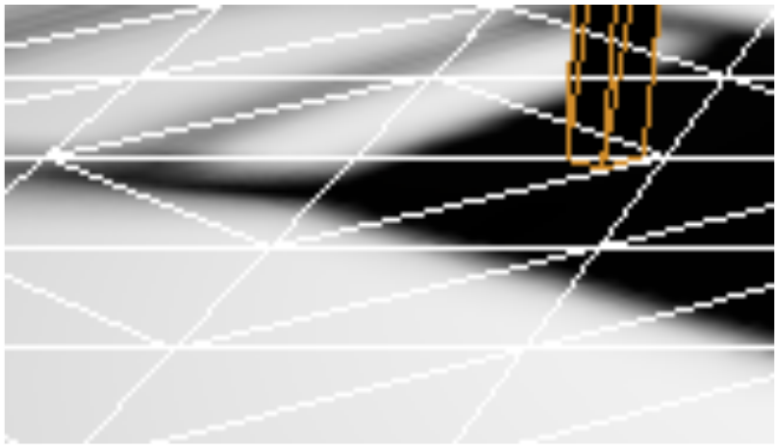
Adaptive Subdivision

Improves Image Quality

without subdivison



with subdivision



Adaptive Subdivision

Subdivision Criteria

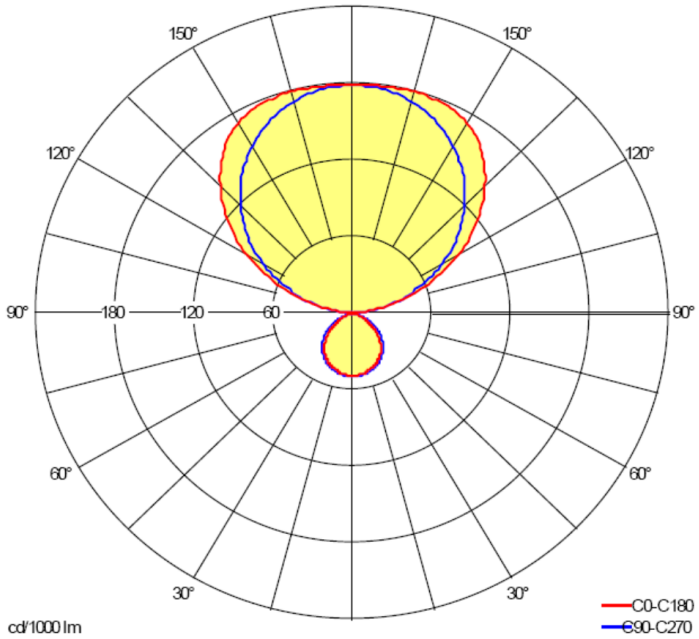
Algorithm: Render the scene *three* times from the point of view of the light source using a hemispherical projection

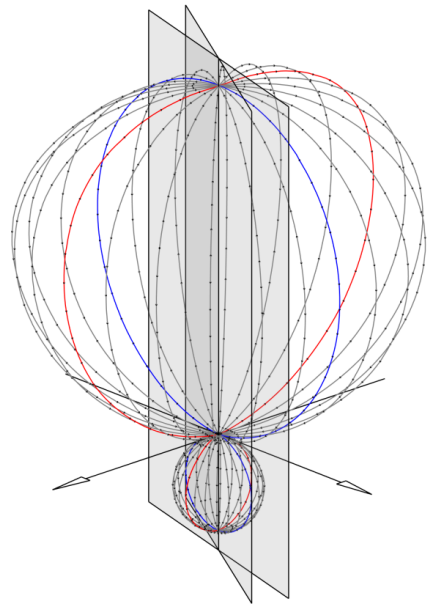
1. Step Render the scene without depth testing and Occlusion Queries enabled (to get complete number of rasterized fragments n_r)
2. Step Render the scene into the depth buffer
3. Step Render the scene with depth testing (GL_LEQUAL) and Occlusion Queries enabled (to get number of visible pixels n_v)

If $n_r \neq n_v \Rightarrow$ there is a shadow boundary on this polygon

Light Distribution Textures

Light Distribution Texture

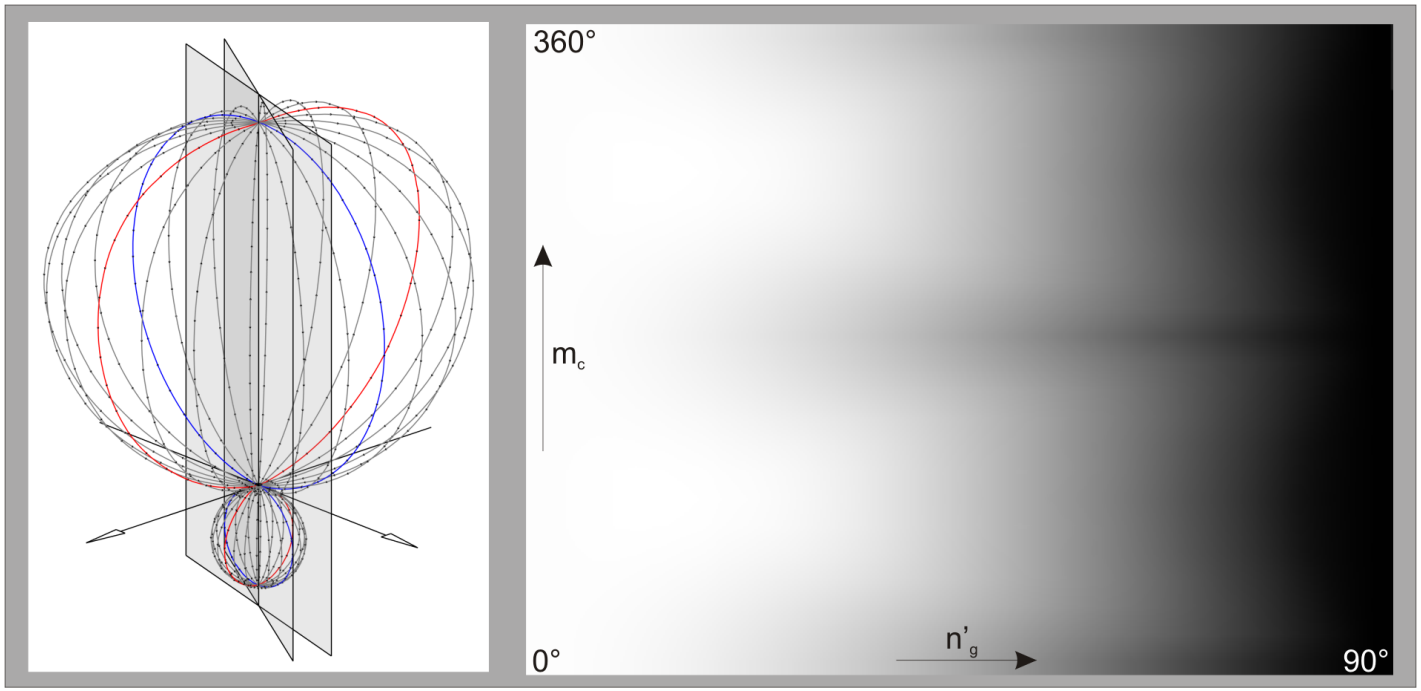




- `GL_LINEAR` to interpolate between discrete measurements
- `GL_REPEAT` in v-direction to ensure continuity
- Access with spherical coordinates

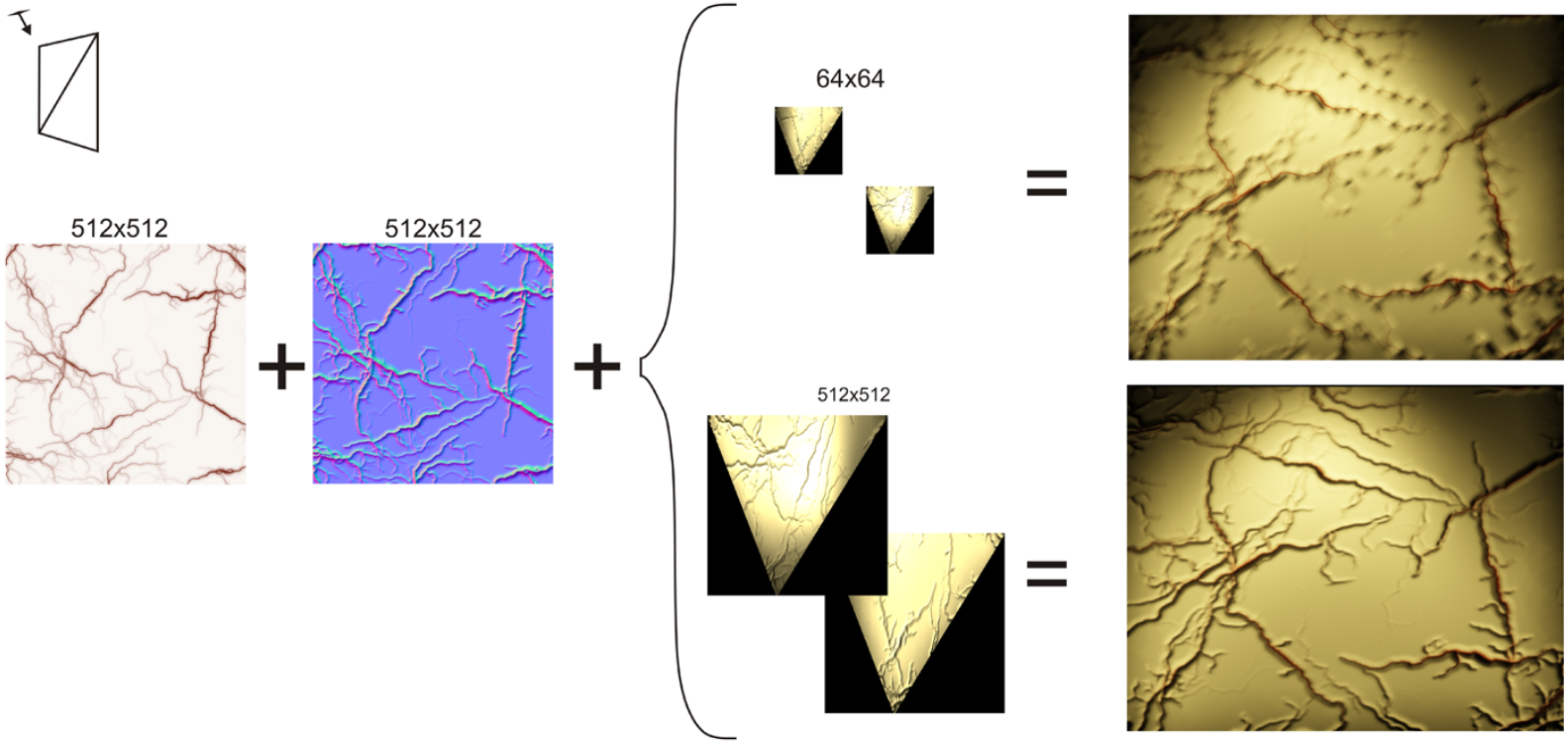
Light Distribution Textures

Light Distribution Texture



- `GL_LINEAR` to interpolate between discrete measurements
- `GL_REPEAT` in v-direction to ensure continuity
- Access with spherical coordinates

Normal Mapping



Radiosity texture size should correspond to normal texture size

Results

Scene 1

- 6.5 million elements, 160 shots
- Heavy use of Occlusion Queries can lead to pipeline stalls

Results

Scene 2

- 4.2 million elements, 256 shots

Thank you for your attention!