Welcome to the Seminar Course

Advanced Computer Graphics

For homepage - Google on:
” Seminar course advanced computer graphics”

Requirements

- Should have read
  - TDA361-Computer Graphics or similar
- The beginners course was theoretically intensive
- This follow-up course is to enjoy what you have learned and also learn the advanced techniques

www.cse.chalmers.se/edu/year/2014/course/TDA361
Google YouTube Dictionary.com Eniro Personer Ulf Assarsson's Home Page Översätt Google Maps Insightly - Login badrum Baby Sleep ...mv - YouTube

CHALMERS
Computer Engineering
Computer Science and Engineering – Chalmers University of Technology and Göteborg University

Seminar Course - DAT205/DIT221 Advanced Computer Graphics 2015 lp3+4

Examiner: Ulf Assarsson
uffe@chalmers.se

Home

COURSE TUESDAYS, 15:15 to 17:00, LP 3+4.
EDIT-building room 3364, 3rd floor, (close to Linsen)

NEWS:

Here is a link to Microsoft's free C++ compiler for Windows:

COURSE START:
Tuesday 20/1, 15:15, room 3364, 3rd floor, EDIT-building, Johanneberg.

Schedule:
schedule in TimeEdit
Study period 3, study week 1,2: Tuesdays, 15:15
Study period 3, study week 3: Thursday, 15:15
Study period 3, study week 4,5,6,7: Tuesdays, 15:15
Study period 4: study week 1,2,3: Tuesdays, 15:15
Study period 4: study week 4: Monday, 15:15
Study period 4: study week 5: Possibly cancelled
Study period 4: study week 6: Monday, 15:15
Study period 4: study week 7,8: Tuesdays, 15:15
Project Page

Very important: Frequently do "refresh", to avoid watching a cached page, since this web-page is updated during the course.

COURSE-PM
7,5 Högskolepoäng
Grades: U (failed), 3, 4, 5
Educational Level: Advanced
Institution: 37 - DATA- OCH INFORMATIONSTEKNIK
Teaching language: English

Teacher and Examiner: Ulf Assarsson, intern phone 1775 (031-7721775)
room 4115, floor 4, the corridor along Rännvägen, ED-huset E-mail: see above.
Course assistants: Viktor Kämpe, Erik Sintorn

Course Description
The compulsory introductory course TDA361/DIT220 Computer Graphics was highly theoretically intensive, giving a brief introduction to a vast amount of topics within computer graphics. In this follow-up course, the students are given a chance to dig deeper into a particular subject, in which they perform a project. Compulsory seminars presents more details on a research-level for a selection of topics, e.g. ambient

Tutorial Shadow Al
rographics2010

Schedule

- Tuesdays 15:15 to 17:00
  - Check that this works with participants

- Study period 3, study week 1,2: Tuesdays, 15:15
- Study period 3, study week 3: Thursday, 15:15
- Study period 3, study week 4,5,6,7: Tuesdays, 15:15
- Study period 4: study week 1,2,3: Tuesdays, 15:15
- Study period 4: study week 4: Monday, 15:15
- Study period 4: study week 5: Possibly cancelled
- Study period 4: study week 6: Monday, 15:15
- Study period 4: study week 7,8: Tuesdays, 15:15
- **14** seminars, ~9 occasions with student presentations, ~25 students => 2-3 students presenting per seminar.

- 80% compulsory attendance. I.e., you can max miss 2 seminars.

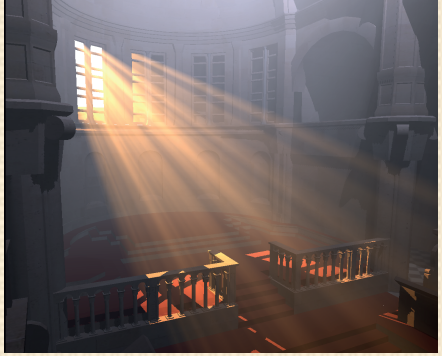
Seminars

- Affects grade (-,0,+)
- Sign up in pairs (you can have individual presentation)
- Choose article of your preferred topic (talk with me). Create 15 minutes presentation
- 1 week before each seminar, email article to group.
- **All students** each week: prepare 2 questions per article before each seminar. Discussions will be based on these.
- 1st presenters, Tues.Feb 26th.

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	

Individual presentations

Seminars

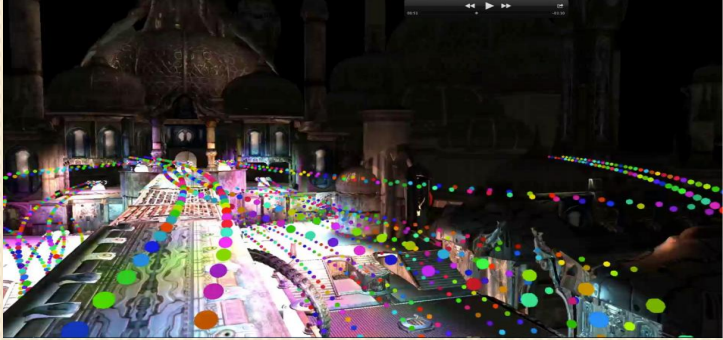


- Seminar 1-4/5
  - Volumetric Shadows
  - Free viewpoint Video, Geometry Compression and GPU Ray tracing of large scenes with shadows, reflections, ao,
  - Shading/Lighting in more detail
  - Advanced Path Tracing
  - Clustered Shading w. 1M lights. Shadows for ~400 lights

Seminars

- Seminar 1-4/5
  - Volumetric Shadows
  - Free viewpoint Video, Geometry Compression and GPU Ray tracing of large scenes with shadows, reflections, ao,
  - Shading/Lighting in more detail
  - Advanced Path Tracing
  - Clustered Shading w. 1M lights. Shadows for ~400 lights

Seminars



- Seminar 1-4/5
  - Volumetric Shadows
  - Free viewpoint Video, Geometry Compression and GPU Ray tracing of large scenes with shadows, reflections, ao,
  - Shading/Lighting in more detail
  - Advanced Path Tracing
  - Clustered Shading w. 1M lights. Shadows for ~400 lights

Project (graded)

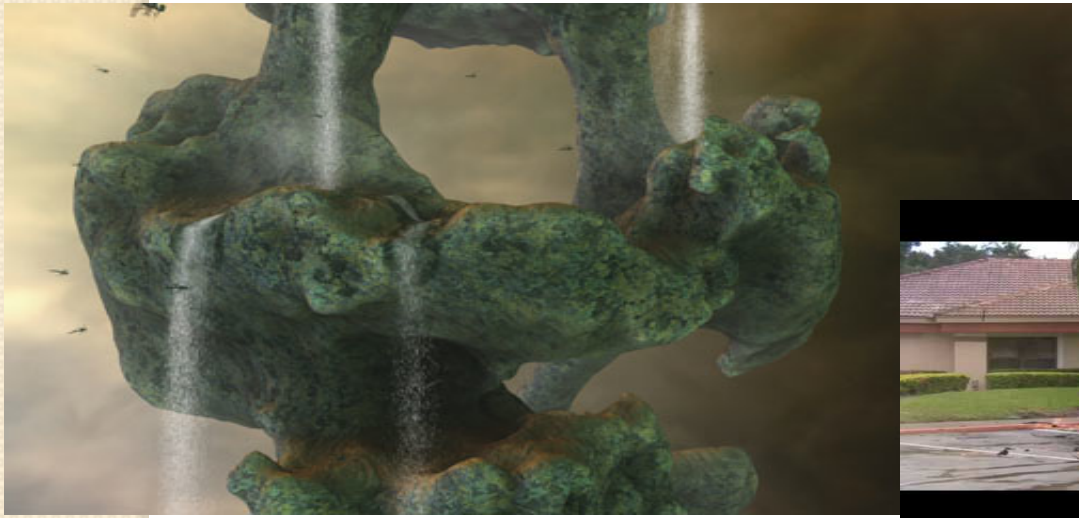
Do a graphics-related project of your choice, e.g.:

- realistic explosions, clouds, fractal mountains (e.g. clip maps/ geomorph/ROAM, ray tracing based a la GPUGems3)
- CUDA program (a general parallel problem)
- Light propagation volumes
- Smaller game
- real-time ray tracer
- ray tracing with photon mapping.
- Ambient occlusion
- Spherical Harmonics
- Collision Detection
- Displacement / parallax mapping / Tesselation shaders

Project (graded)

Do a graphics-related project of your choice, e.g.:

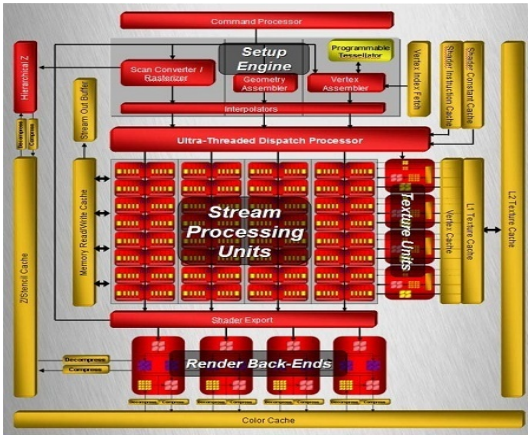
- realistic explosions, clouds, fractal mountains (e.g. clip maps/geomorph/ ROAM, ray tracing based a la GPUGems3)



Project (graded)

Do a graphics-related project of your choice, e.g.:

- realistic explosions, clouds, fractal mountains (e.g. clip maps/geomorph/ ROAM, ray tracing based a la GPUGems3)
- CUDA program (a general parallel problem)

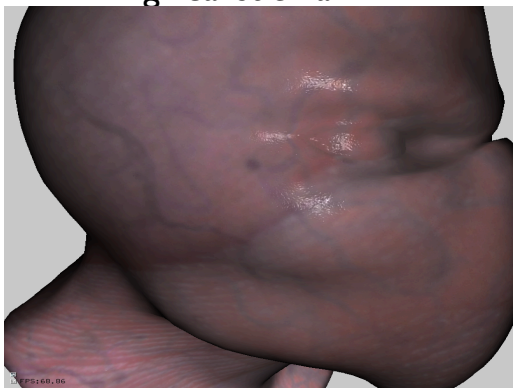


Hair rendering

Project

Do a graphics-related project of your choice, e.g.:

- realistic explosions, clouds, fractal mountains (e.g. clip maps/geomorph/ ROAM, ray tracing based a la GPUGems3)
- CUDA program (a general parallel problem)
- Realistic Skin Rendering
  - Adding realistic hair



Project

Do a graphics-related project of your choice, e.g.:

- real-time ray tracer
- photon mapping.
- Path tracer.

Project

Do a graphics-related project of your choice, e.g.:

- Smaller or larger game, in a group or alone

Project

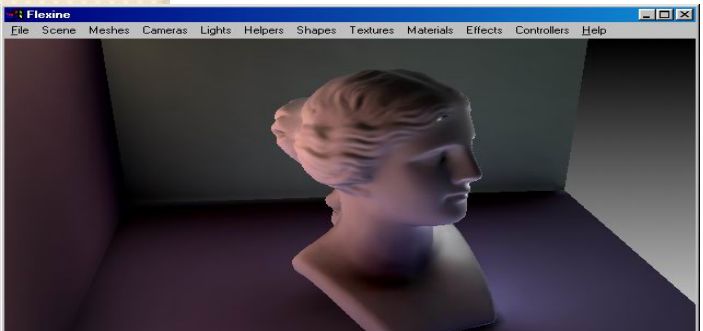
Do a graphics-related project of your choice, e.g.:

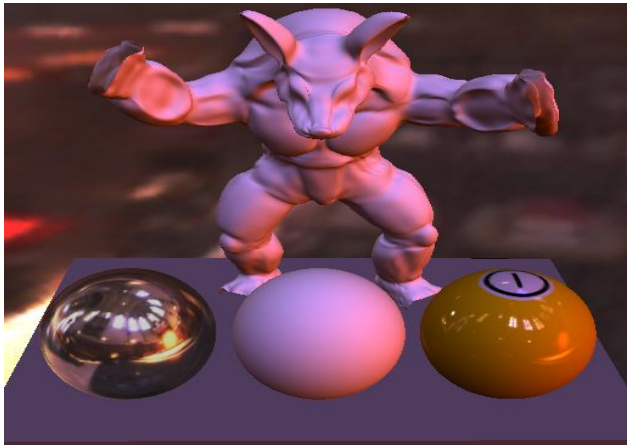
- Smaller or larger game
  - With screen space ambient occlusion (SSAO)

Project

Do a graphics-related project of your choice, e.g.:

- Smaller or larger game
  - With screen space ambient occlusion (SSAO)
  - Spherical Harmonics – for indirect illumination





Project

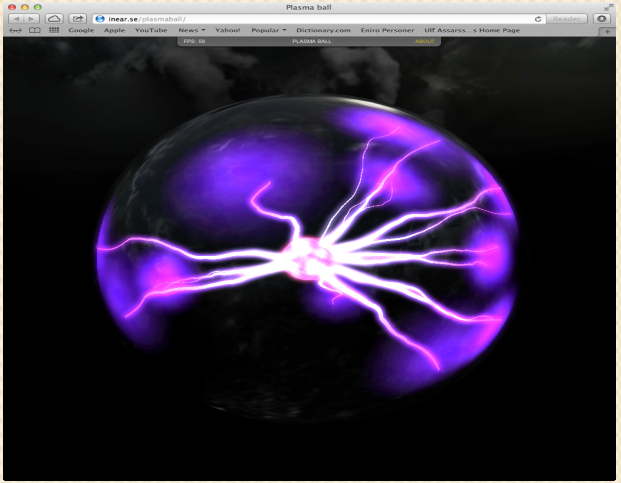

Wireframe
Basic Shading
Split-screen
Tessellation
Daylight:

g.:

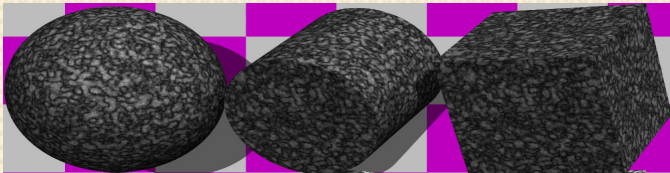
Project

WebGL examples:

- http://inear.se/plasmaball/
- http://www.clicktorelease.com/code/perlin/explosion.html



- E.g., Prodedural texturing



Project

Do a graphics-related project of your choice, e.g.:

- Model and animate using Maya/3DSMax or Blender

Advanced Computer Graphics

- Home page:
- http://www.cse.chalmers.se/edu/course/TDA361/Advanced Computer Graphics/

Volumetric Shadows

Ulf Assarsson

(15 minutes)

God Rays

Outline - Volumetric Shadows

- Real-Time **Single** Scattering in Homogeneous Participating Media
  - Ray-Marching based approaches
  - Shadow-Volume based approaches
- Real-Time **Multiple** Scattering in Homogeneous Participating Media

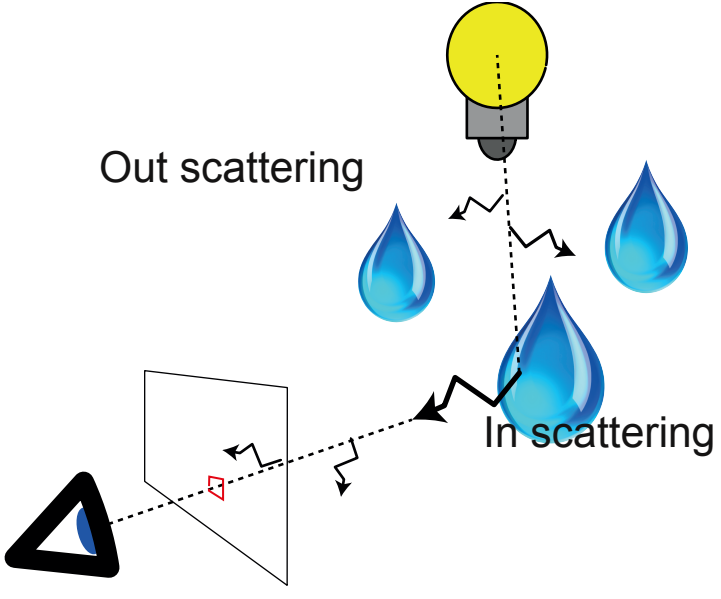
Scattering in Participating Media

- Participating media *scatter* light
  - “Photons bounce off particles in the medium”
- Some scattered light reaches the observer (most doesn’t)
- Thus, we can see the participating medium

Scattering in Participating Media

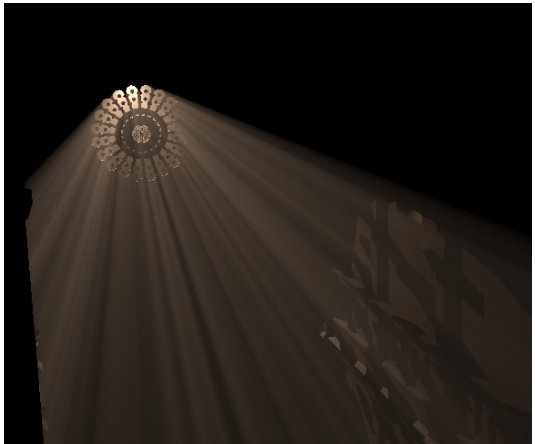
Single Scattering [Blinn 82]:

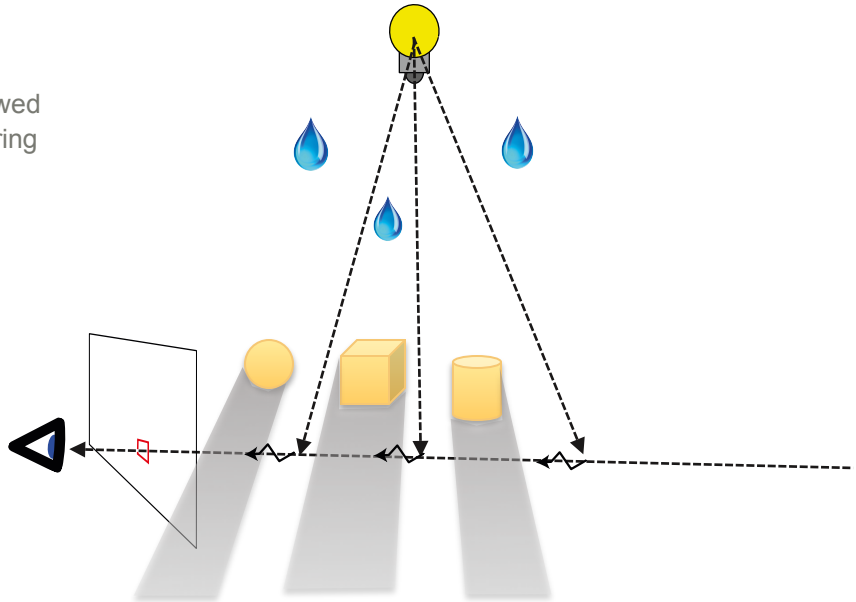
- In scattering
  - Visible image contribution
- Out scattering
  - Attenuation
- Absorption
  - Attenuation



Scattering in Participating Media

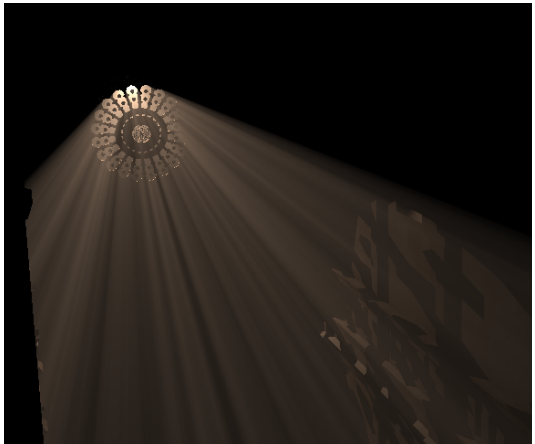
- Blockers cause airlight shadows
  - Only in scattering along non-shadowed regions of eye ray for **single** scattering

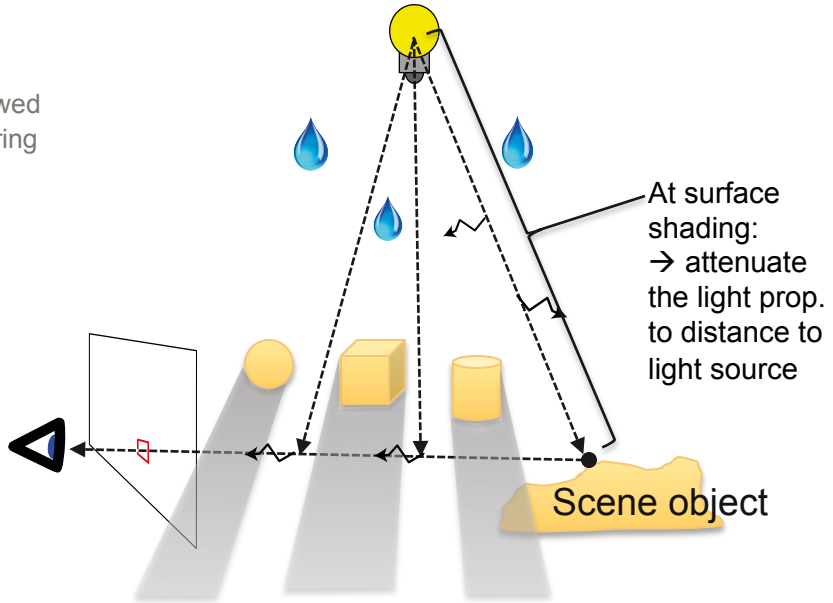




Scattering in Participating Media

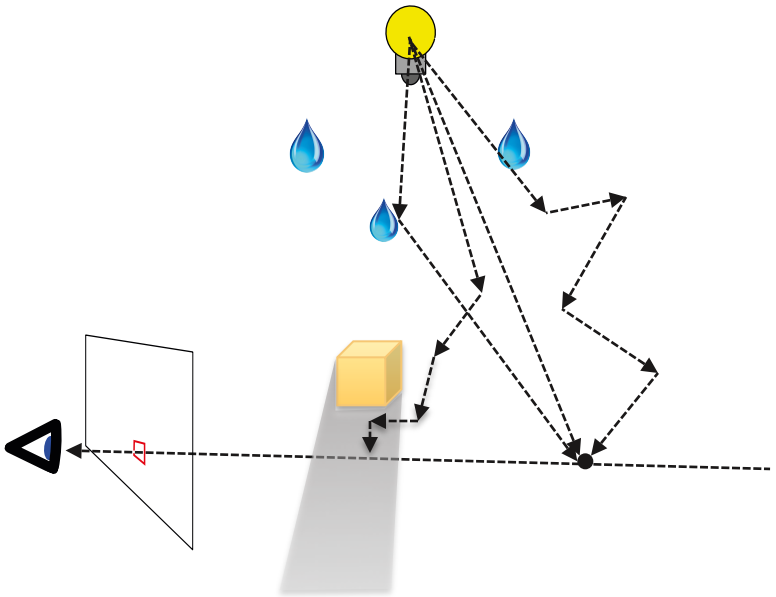
- Blockers cause airlight shadows
  - Only in scattering along non-shadowed regions of eye ray for **single** scattering





Scattering in Participating Media

- Multiple Scattering:
  - In reality, photons can bounce many times before in scattering
  - **In scattering** happens also in shadowed regions.



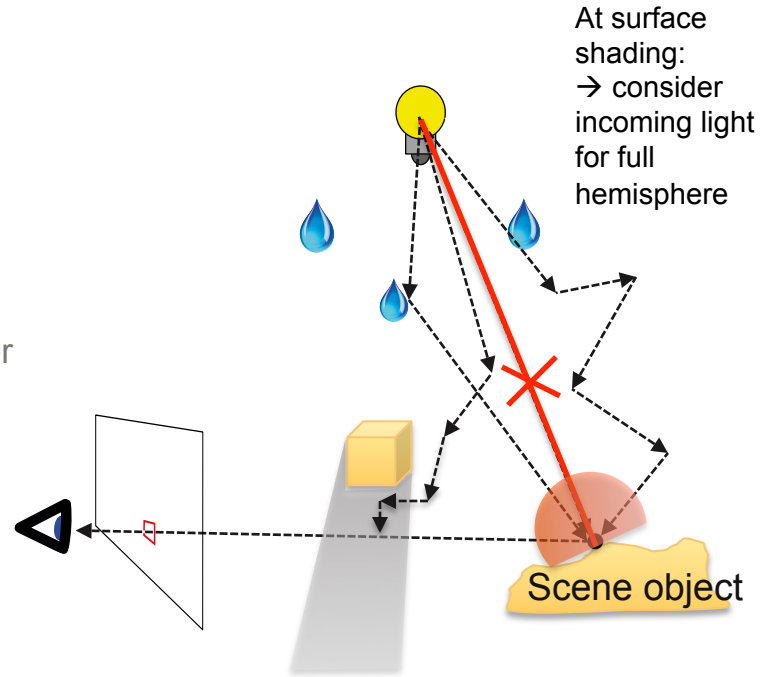
Scattering in Participating Media

Multiple scattering

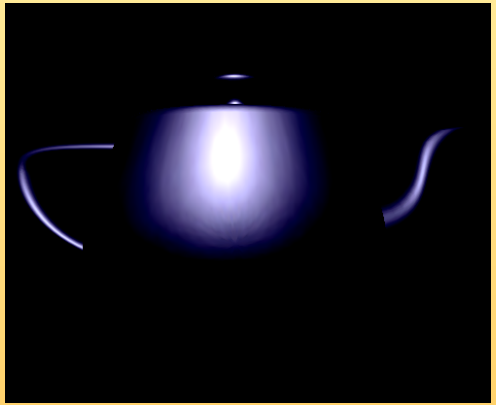
Single scattering

Scattering in Participating Media

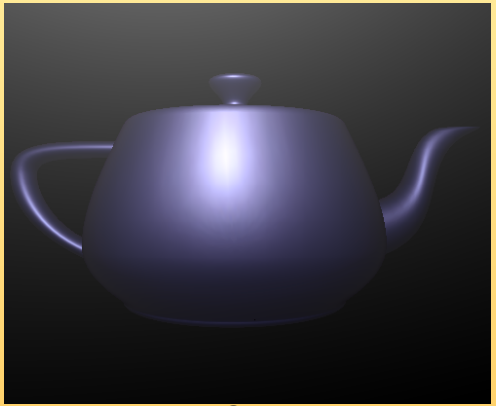
- Multiple Scattering:
  - In reality, photons can bounce many times before in scattering
  - **In scattering** happens also in shadowed regions.
  - **Surface shading** should consider not only one light direction but all incoming non-shadowed directions
- Typically too complex for real-time.



Scattering in Participating Media



One incoming direction

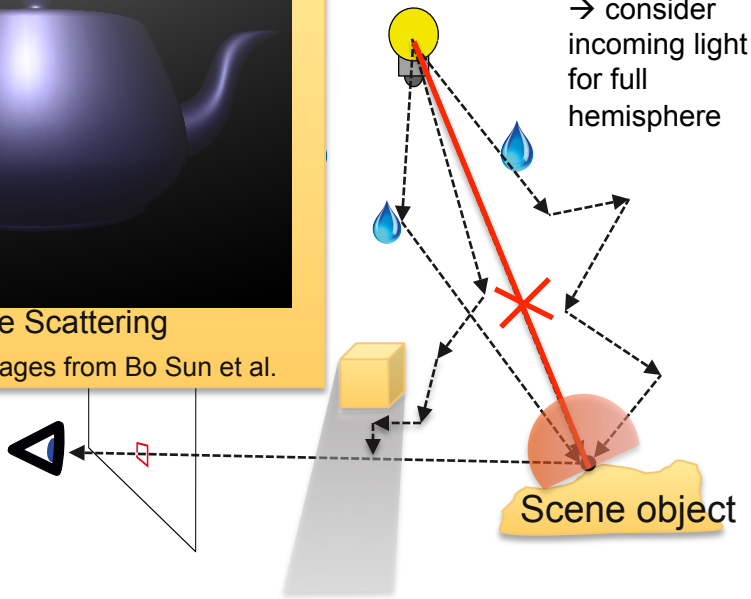


Multiple Scattering

Images from Bo Sun et al.

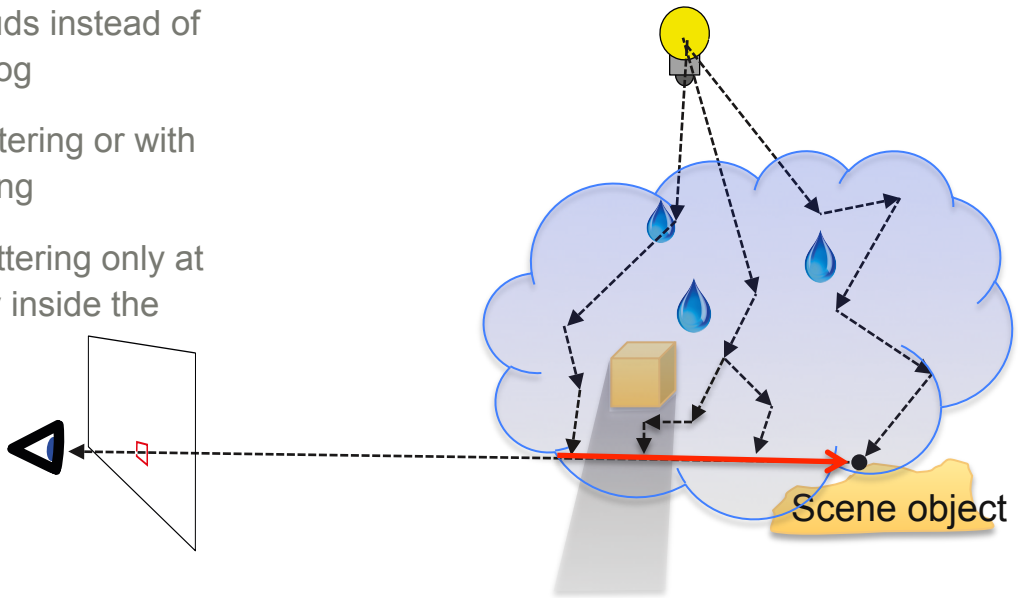
directions

- Typically too complex for real-time.



Scattering in Participating Media

- Non-homogeneous media
  - Smoke and clouds instead of homogeneous fog
  - With single scattering or with multiple scattering
  - Consider in-scattering only at parts of light ray inside the medium

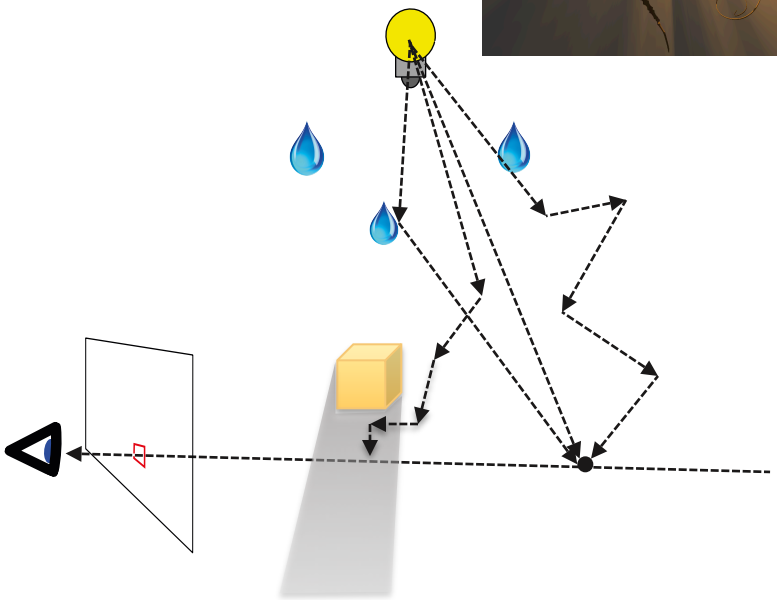


Scattering in Participating Media

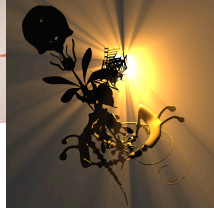
- If medium has low albedo
  - I.e., few small particles such as
    - dust, thin fog
- Then, single scattering is the dominating effect.
  - ➢ Let us first focus on this simplest case for real-time purposes:
    - **single** scattering in **homogeneous** participating media.

Scattering in Participating Media

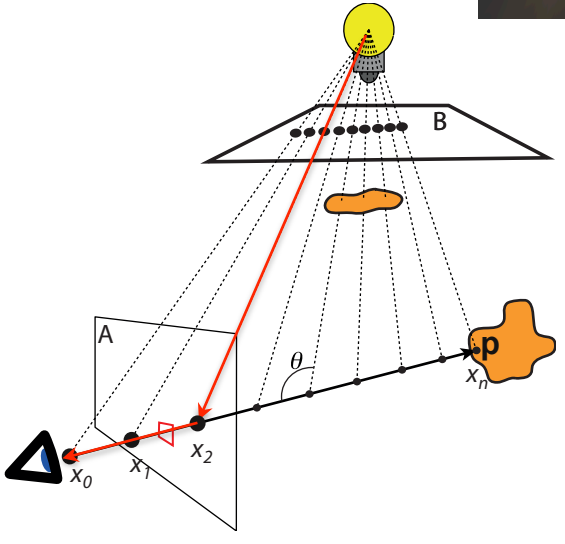
- If medium has low albedo
  - I.e., few small particles such as
    - dust, thin fog
- Then, single scattering is the dominating effect.
  - Let us first focus on this simplest case for real-time purposes:
    - **single** scattering in **homogeneous** participating media.



Ray-Marching Based Single Scattering

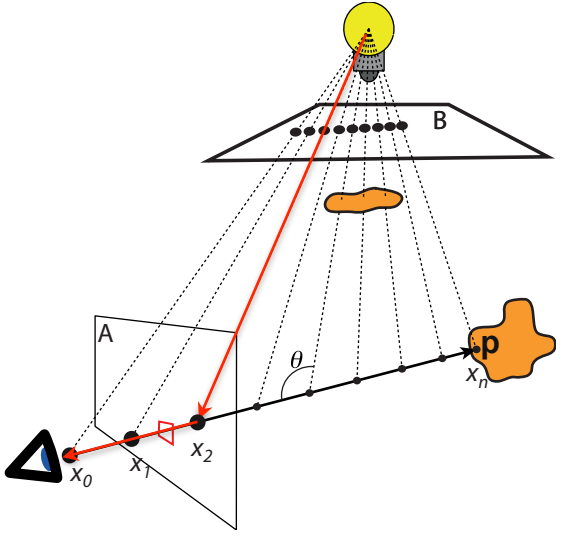


- Pseudo code:
- For every pixel in screen space (A)
  - Step along the eye ray, using ray marching
  - For each sample position x_j
    - Check in shadow map (B) if light source is visible
    - If so, add to pixel's color:
      - in-scattered light contribution with attenuation



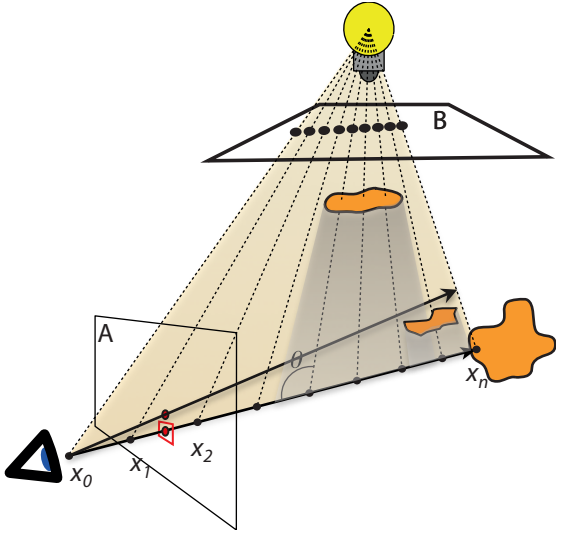
Ray-Marching Based Single Scattering

- Ray-marching: at each point, determine if lit (shadow map)
- Done with:
  - CUDA (Baran et al. 2010)
  - Loop in fragment shader (e.g., Toth and Umenhoffer 2009, Chen et al. 2011)
  - Alpha-blended planes (Dobashi et. al. 2002, Imagire et. al. 2007)
  - CPU (Kajiya and Von Herzen 1984)



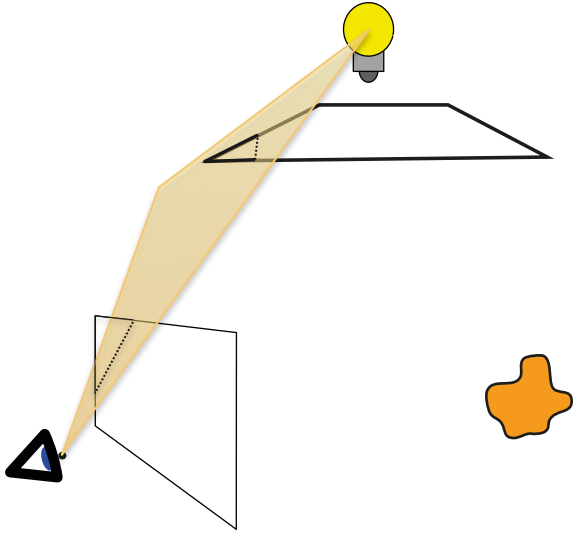
Ray-Marching Based Single Scattering

- Utilize coherency (Baran et al. 2010)
  - Compute ray incrementally from previous ray in same plane as the ray, light and eye
    - Only new shadowed regions can appear (not new lit regions)



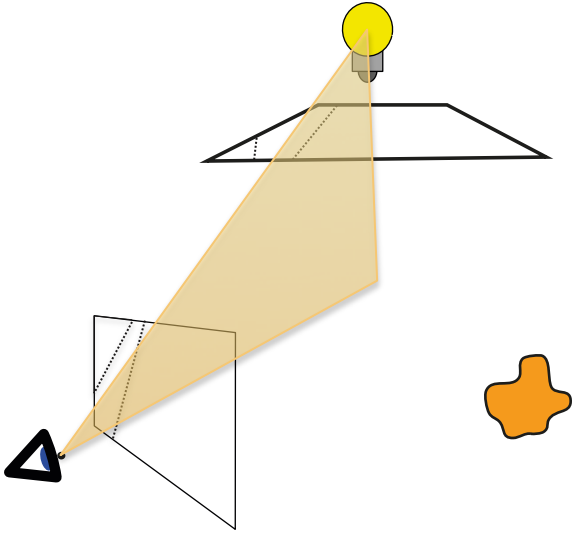
Ray-Marching Based Single Scattering

- On shadow map rendering
  - Skew projection such that these *epipolar* coherent planes become rows of the SM (Chen et al. 2011)



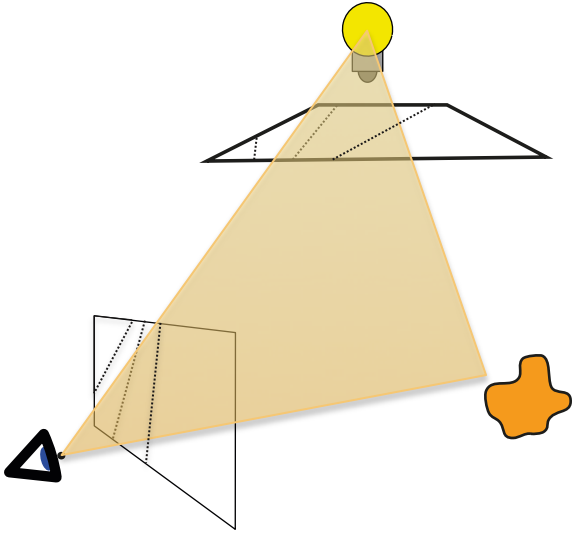
Ray-Marching Based Single Scattering

- On shadow map rendering
  - Skew projection such that these *epipolar* coherent planes become rows of the SM (Chen et al. 2011)



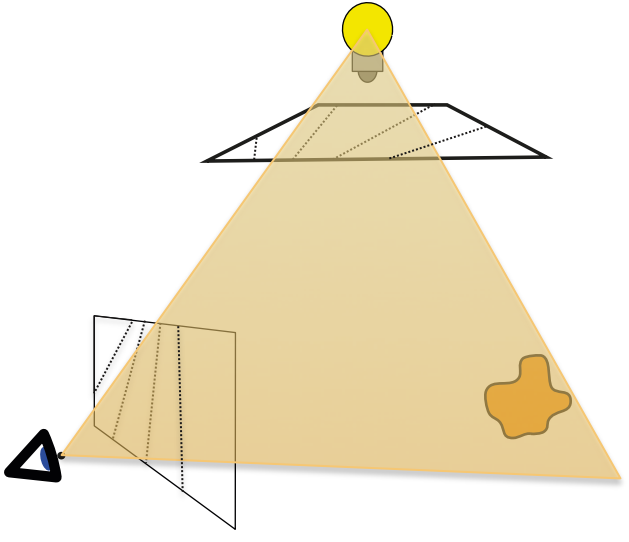
Ray-Marching Based Single Scattering

- On shadow map rendering
  - Skew projection such that these *epipolar* coherent planes become rows of the SM (Chen et al. 2011)



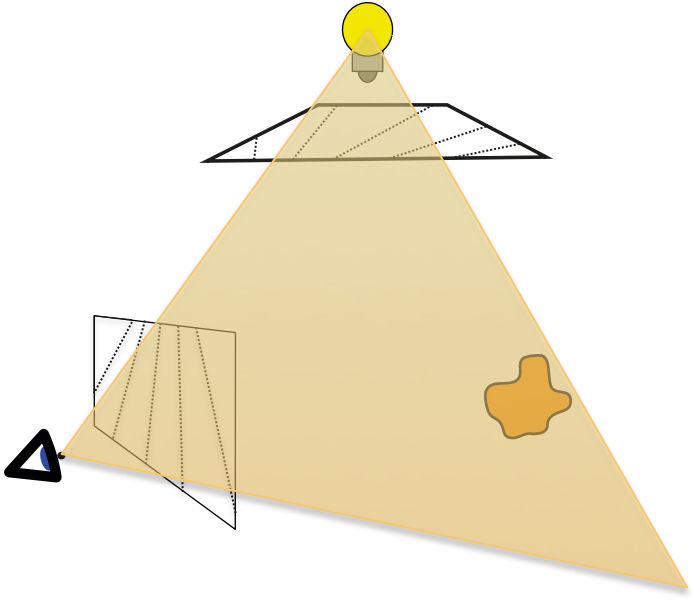
Ray-Marching Based Single Scattering

- On shadow map rendering
  - Skew projection such that these *epipolar* coherent planes become rows of the SM (Chen et al. 2011)



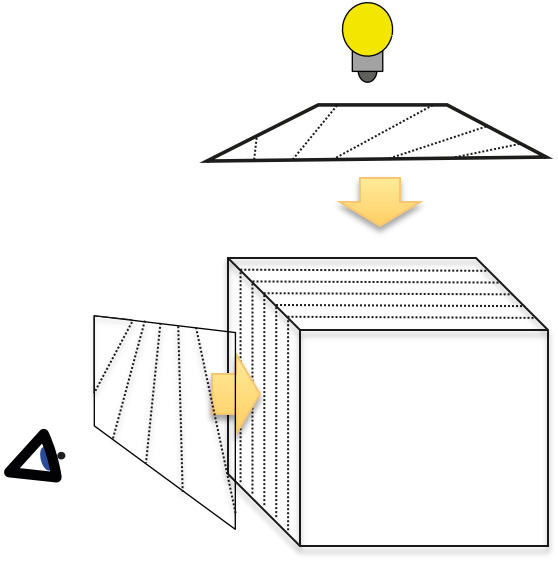
Ray-Marching Based Single Scattering

- On shadow map rendering
  - Skew projection such that these *epipolar* coherent planes become rows of the SM (Chen et al. 2011)



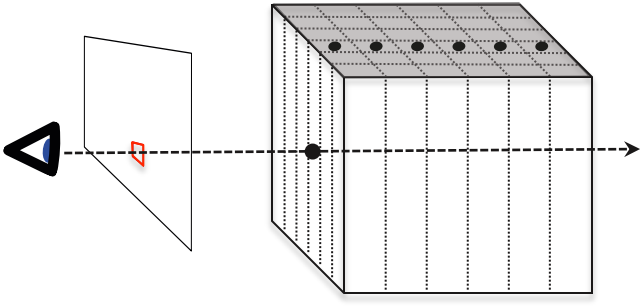
Ray-Marching Based Single Scattering

- On shadow map rendering
  - Skew projection such that these *epipolar* coherent planes become rows of the SM (Chen et al. 2011)



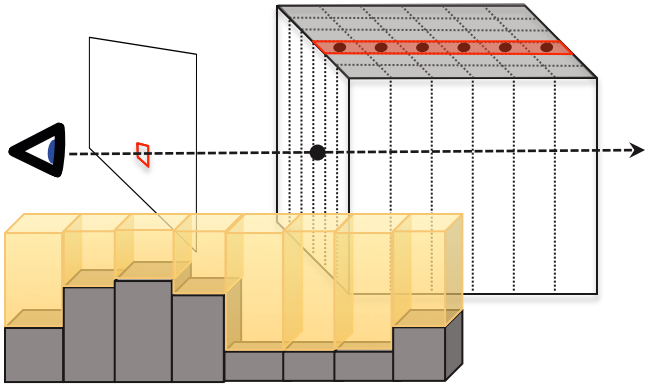
Ray-Marching Based Single Scattering

- On shadow map rendering
  - Skew projection such that these *epipolar* coherent planes become rows of the SM (Chen et al. 2011)
- Raymarching along view ray becomes traversal along SM row



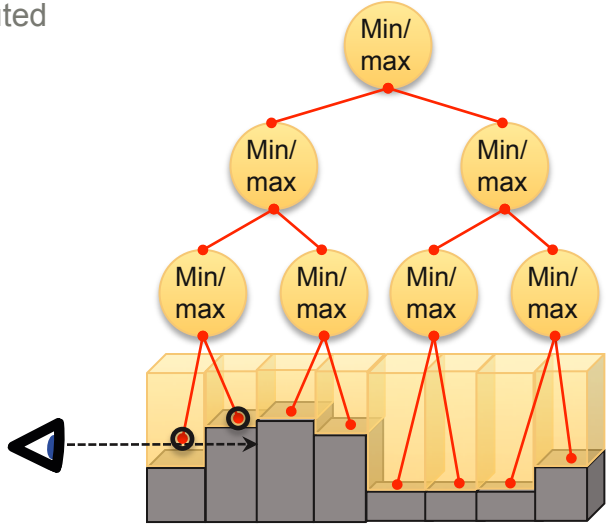
Ray-Marching Based Single Scattering

- Shadow map is a height field
- For each shadow map row (Chen et al.):
  - a 1D min-max mipmap is computed



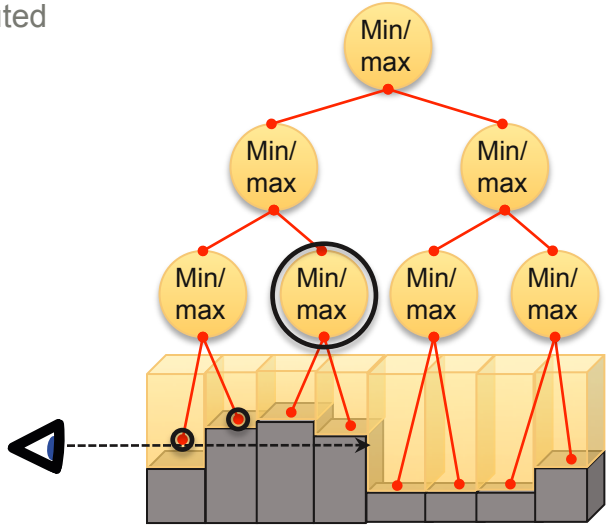
Ray-Marching Based Single Scattering

- Shadow map is a height field
- For each shadow map row (Chen et al.):
  - a 1D min-max mipmap is computed



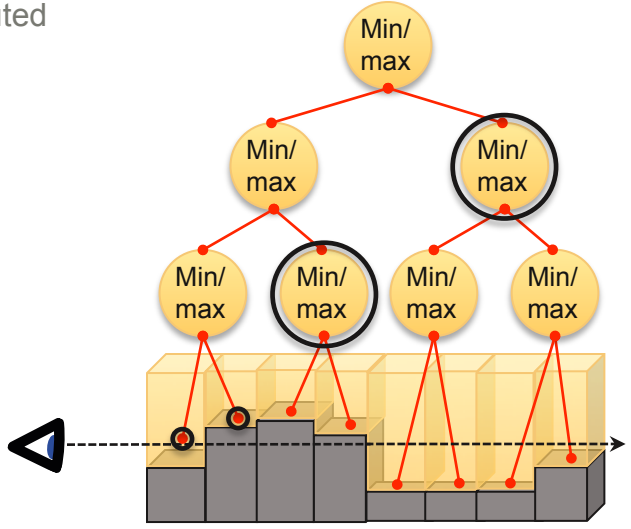
Ray-Marching Based Single Scattering

- Shadow map is a height field
- For each shadow map row (Chen et al.):
  - a 1D min-max mipmap is computed



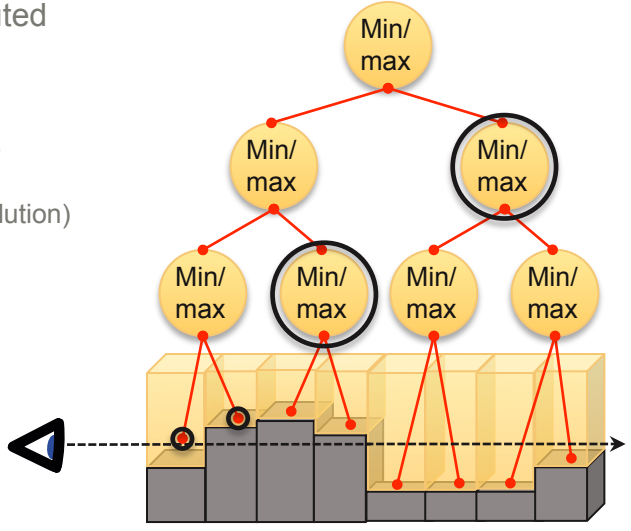
Ray-Marching Based Single Scattering

- Shadow map is a height field
- For each shadow map row (Chen et al.):
  - a 1D min-max mipmap is computed

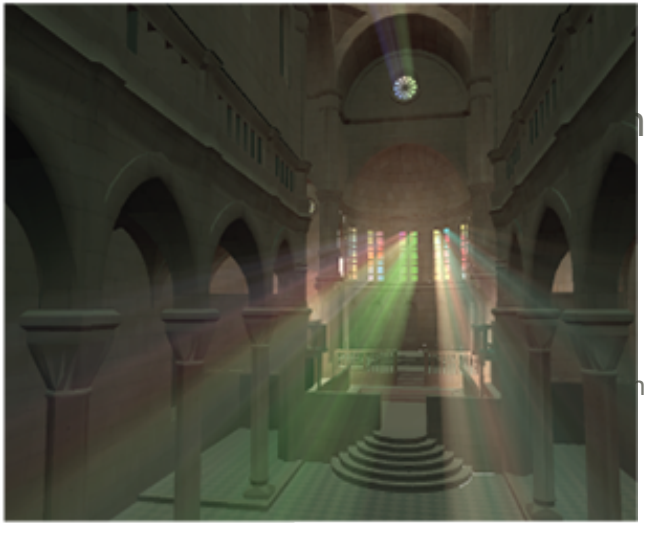


Ray-Marching Based Single Scattering

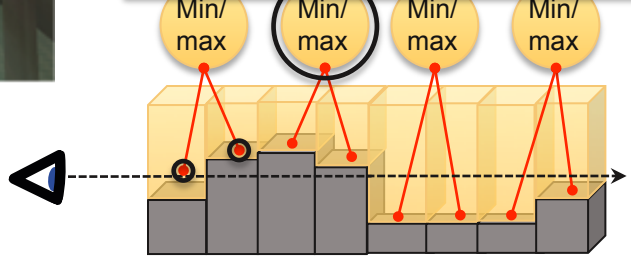
- Shadow map is a height field
- For each shadow map row (Chen et al.):
  - a 1D min-max mipmap is computed
  - Logarithmic search time
    - Algorithm O(#pixels * log d)
      - d=sqrt(shadow_map_resolution)
    - Real-time



Ray-Marching Based Single

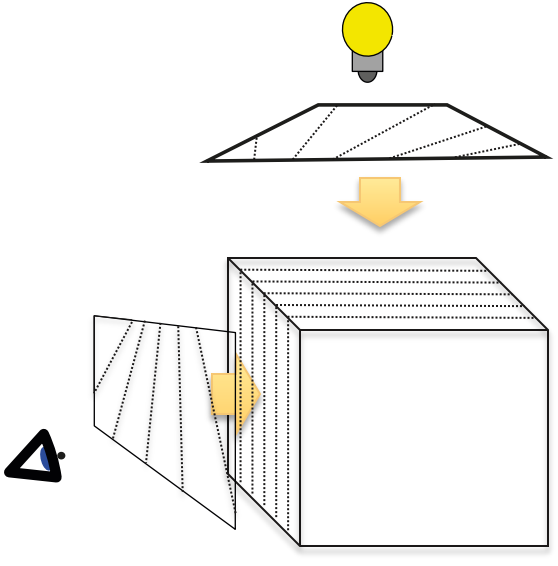


55 fps. Image from Chen et al. 2011



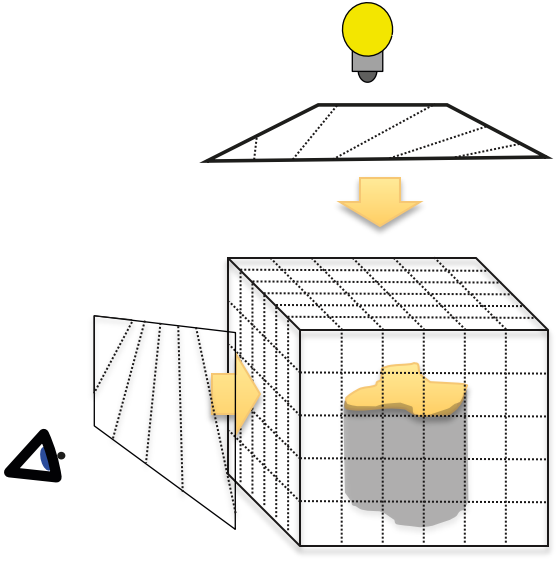
Ray-Marching Based Single Scattering

- Wyman 2011:
  - CUDA
  - Extremely fast
  - Voxelizes shadow casters



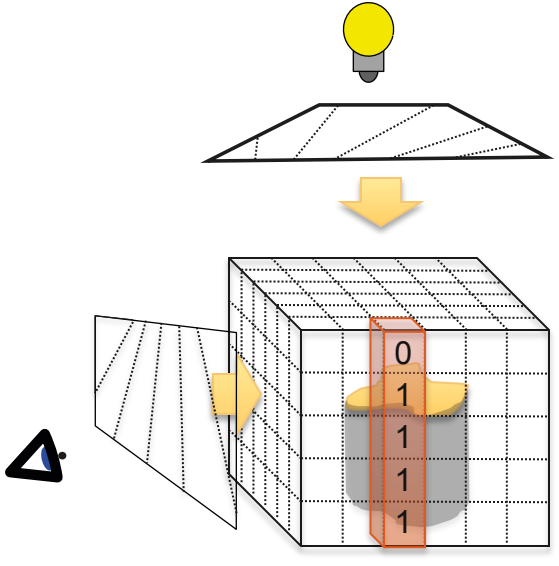
Ray-Marching Based Single Scattering

- Wyman 2011:
  - CUDA
  - Extremely fast
  - Voxelizes shadow casters
    - 3D bit grid



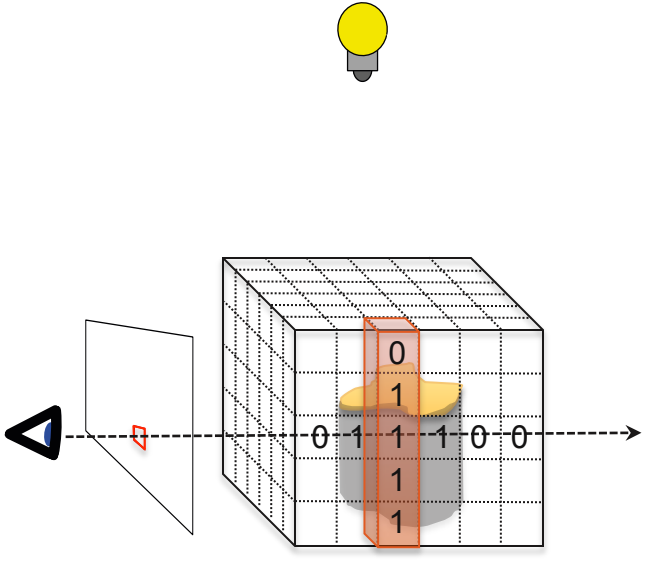
Ray-Marching Based Single Scattering

- Wyman 2011:
  - CUDA
  - Extremely fast
  - Voxelizes shadow casters
    - 3D bit grid
    - For each column
      - Binary prefix sum



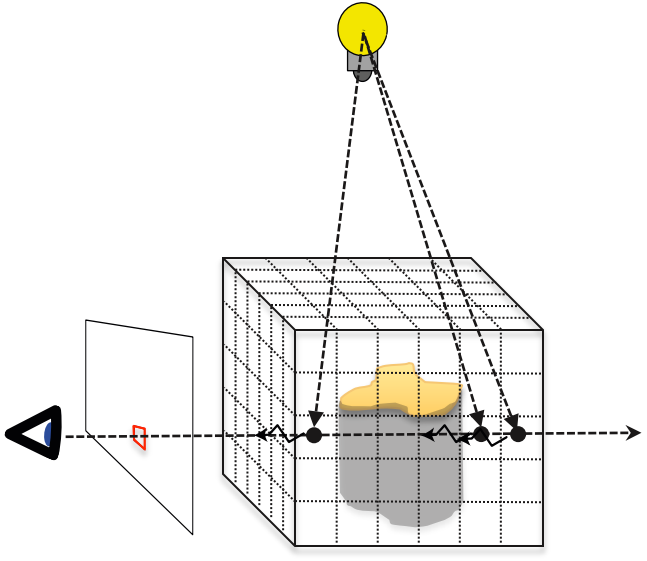
Ray-Marching Based Single Scattering

- Wyman 2011:
  - CUDA
  - Extremely fast
  - Voxelizes shadow casters
    - 3D bit grid
    - For each column
      - Binary prefix sum
    - For each eye ray
      - Compute #zeroes (popc)
        - 128 bits at once

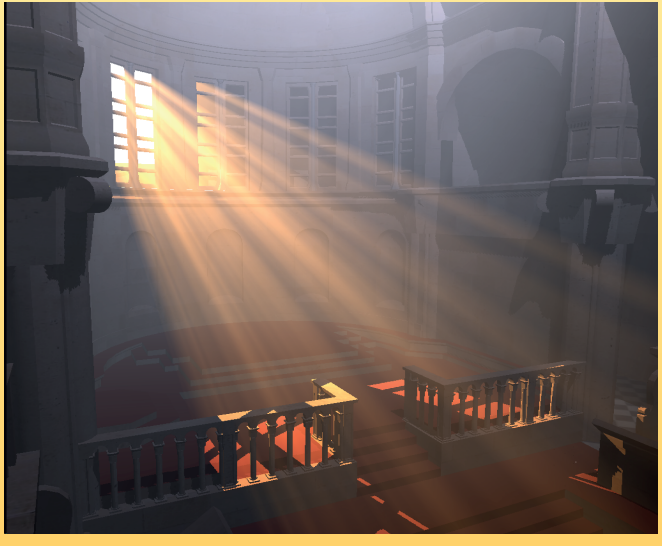


Ray-Marching Based Single Scattering

- Wyman 2011:
  - CUDA
  - Extremely fast
  - Voxelizes shadow casters
  - Less accurate computations
    - In-scattering should actually be different per voxel



Ray-Marching Based Single Sc

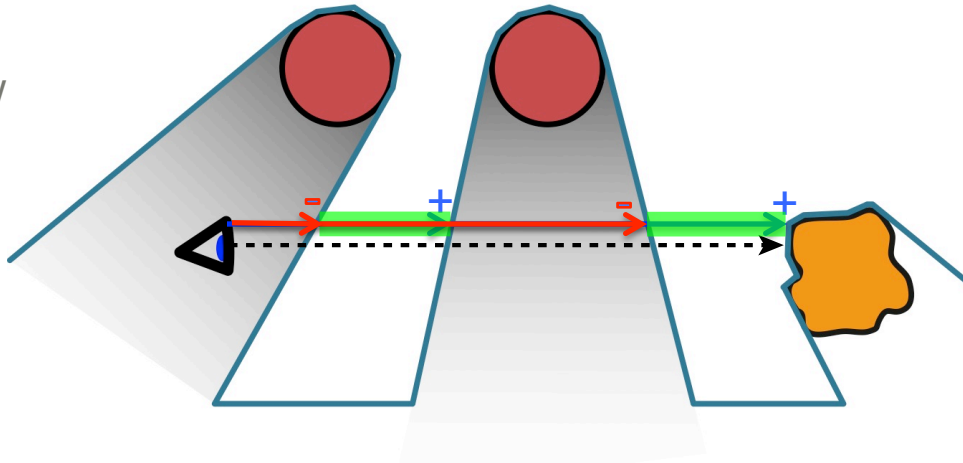


~90 fps. Two lights. Image from Billeter et al. 2010

~118 fps. Image from Wyman 2011

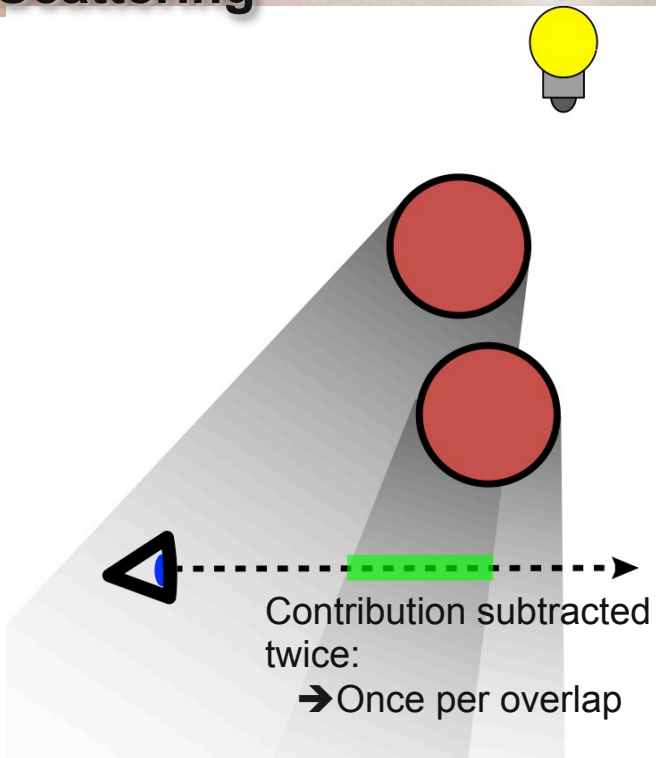
Shadow-Volume Based Single Scattering

- There are of course also Shadow Volume based approaches
  - Billeter et al. 2010, Biri et al. 2006, James 2003, Nishita et al. ’87, Max ’86
  - Add airlight for frontfacing shadow volume polygons
  - Subtract for backfacing ones



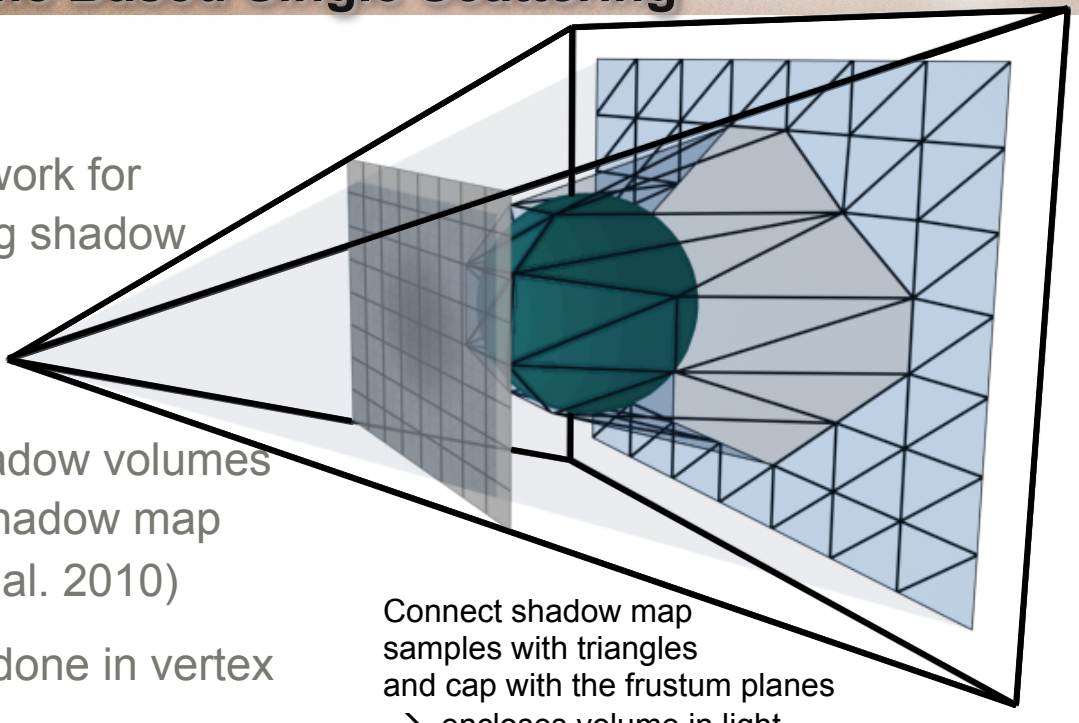
Shadow-Volume Based Single Scattering

- Caveat:
  - Does not work for intersecting shadow volumes
- Solution:
  - Create shadow volumes from the shadow map (Billeter et al. 2010)



Shadow-Volume Based Single Scattering

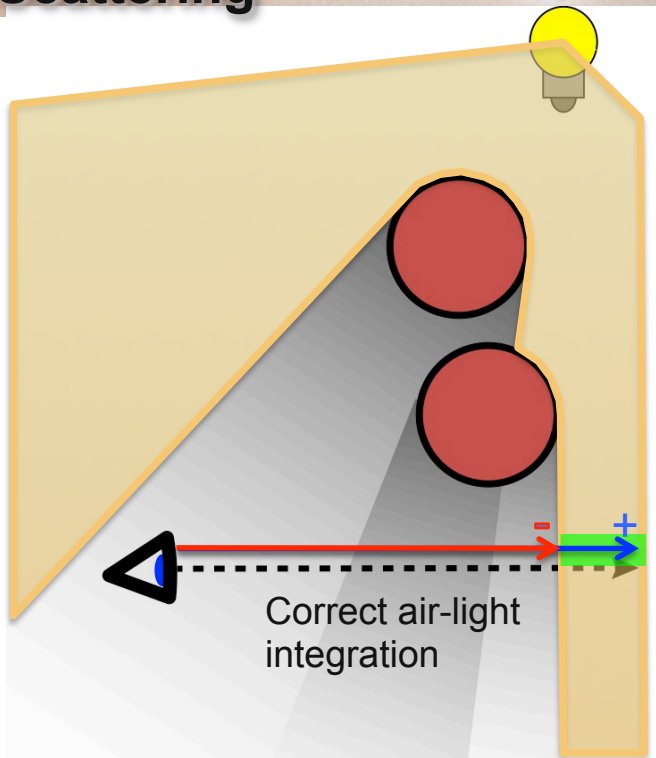
- Caveat:
  - Does not work for intersecting shadow volumes
- Solution:
  - Create shadow volumes from the shadow map (Billeter et al. 2010)
    - Fast if done in vertex shader



Connect shadow map samples with triangles and cap with the frustum planes
→ encloses volume in light

Shadow-Volume Based Single Scattering

- Caveat:
  - Does not work for intersecting shadow volumes
- Solution:
  - Create shadow volumes from the shadow map (Billeter et al. 2010)
    - Fast if done in vertex shader



Shadow-Volume Based Single Scattering

Pseudo code (Billeter et al. 2010):

- Render shadow map
- Render scene from camera
  - Add light attenuation in the shaders

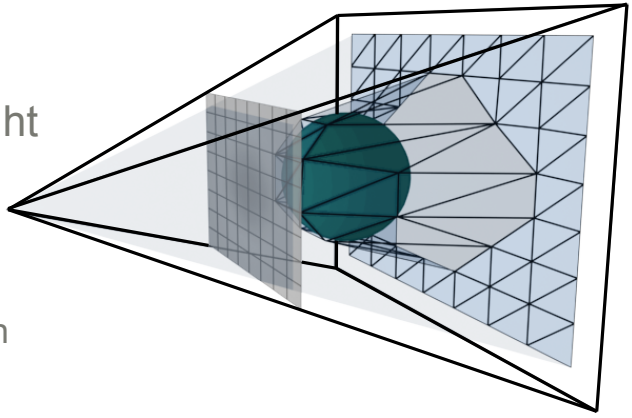
```
vec3f color_from_shading = ...
// Compute light attenuation factor
vec3 lightToObj = lightposition[i] - objPos;
float beta = 0.04; // optical thickness
float attenuation = 0.1 *
     exp(-beta * length(lightToObj))/ length(lightToObj)^2 *
     exp(-beta * length(objPos));
// lightintensity = 1000.0f is a reasonable value
gl_FragData[0].xyz += attenuation * lightintensity[i] *
     lightcolor[i] * color_from_shading;
```

Shadow-Volume Based Single Scattering

Pseudo code (Billeter et al. 2010):

- Render shadow map
- Render scene from camera
  - Add light attenuation in the shaders
- Create the light volume
- Render light-volume mesh to add airlight
  - additive blending
  - Shader computes airlight
  - If polygon backfacing -> negate contribution

```
vec3f color_from_shading = ...
// Compute light attenuation factor
vec3 lightToObj = lightposition[i] - objPos;
float beta = 0.04; // optical thickness
float attenuation = 0.1 *
    exp(-beta * length(lightToObj))/ length(lightToObj)^2 *
    exp(-beta * length(objPos));
// lightintensity = 1000.0f is a reasonable value
gl_FragData[0].xyz += attenuation * lightintensity[i] *
    lightcolor[i] * color_from_shading;
```



Shadow-Volume Based Single Scattering

Pseudo code (Billeter et al. 2010):

- Render shadow map
- Render scene from camera
  - Add light attenuation in the shaders
- Create the shadow volumes
- Render the mesh to add airlight
  - additive blending
  - Shader computes airlight
  - If polygon backfacing -> negate con

```
vec3f color_from_shading = ...
// Compute light attenuation factor
vec3 lightToObj = lightposition[i] - objPos;
float beta = 0.04; // optical thickness
float attenuation = 0.1 *
    exp(-beta * length(lightToObj))/ length(lightToObj)^2 *
    exp(-beta * length(objPos));
// lightintensity = 1000.0f is a reasonable value
gl_FragData[0].xyz += attenuation * lightintensity[i] *
    lightcolor[i] * color_from_shading;
```

```
// Step 4:  Computing airlight with shadows.
// Render the shadow volume mesh with depth testing disabled
// and additive blending enabled.
// Fragment shader:
uniform sampler2D camera_z;
in vec3 objPos, lightPos, viewPos;
void main()
{
    float facing = gl_FrontFacing ?  -1.0 :  1.0;
    vec3 op = objPos; // mesh's fragment position
    vec3 myz; // z-value in depth buffer (scene-fragment)
    myz = texelFetch2D(camera_z, ivec2(gl_FragCoord.xy),0).rgb;
    op = (myz.z >= op.z) ?  myz :  op; // keep z closest to eye
    float ai = facing * airlight(viewPos-lightPos, op-lightPos);
    gl_FragData[0].xyz = ai * light_color;
}
```

ngle Scattering

```
float airlight(vec3 viewPos, vec3 objPos)
{
    vec3 v=-viewPos;
    vec3 d = -viewPos + objPos;

    float dao = dot(v, normalize(d));
    float dbo = length(d) - dao;
    float dlo = sqrt( dot(v,v) - dao*dao );

    float beta = 0.04; // optical thickness
    float tao = beta * dao;
    float tbo = beta * dbo;
    float tlo = beta * dlo;

    vec2 ab = airlight_components( tao, tbo, tlo );

    float ae = 1, be = 1;
    if( dao > 0 && dbo < 0 )
        be = exp( -beta * length(d) );
    else if( dao > 0 && dbo > 0 )
        be = exp( -tao );
    else if( dao < 0 && dbo > 0 )
        ae = be = exp( -tao );
    else
        ae = be = -1000000;

    float abc = sign(tao) * ab.x * ae + sign(tbo) * ab.y * be;
    // lightintensity = 1000.0f is a reasonable value.  The
    // division by tlo is because we premultiply the lookup
    // table by tlo to get a better range of precission.
    float ret = beta*beta*lightIntensity / (4*PI) * abc / tlo;
    return clamp( ret, 0, 1e7 );
}
```

```
uniform sampler2D LUT; // airlight lookup table
float map_x(float t)
{
    return sign(t)*((log(abs(t))+16.1181)/17.9099)/2+0.5;
}
float map_y(float t)
{
    return (log(abs(t))+16.1181)/17.9099;
}
vec2 airlight_components(float tao, float tbo, float tlo)
{
    float at = texture2D(LUT, vec2(map_x(tao), map_y(tlo))).r;
    float bt = texture2D(LUT, vec2(map_x(-tbo),map_y(tlo))).r;
    return vec2(at,bt)
}
```

Demo

Shadow-Volume Based Single Scattering

Videos:

Scene: Sponza2

- 5 light sources
- 512^2 shadow map
- 100 FPS on average

Multiple Scattering

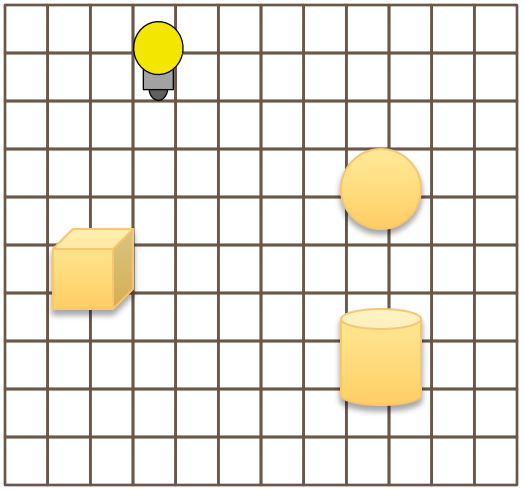
- We've got single-scattering
  - real time
  - produces quite nice results
  - simple algorithms
- What do we have to do to add multiple scattering?

Multiple Scattering - idea

- Simulate light as it passes through the scene
  - “Propagation”
- At every point, compute scattering:
  - extinction (absorption + out-scattering)
  - in-scattering
- We can do this in a 3D grid

Multiple Scattering

- *Real-Time Multiple Scattering with Light Propagation Volumes*, (Billeter et al. 2012)
  - Extends “Light Propagation Volumes”, (Kaplanyan and Dachsbacher, CryEngine3, 2010)
    - Indirect illumination, one or multiple light bounces

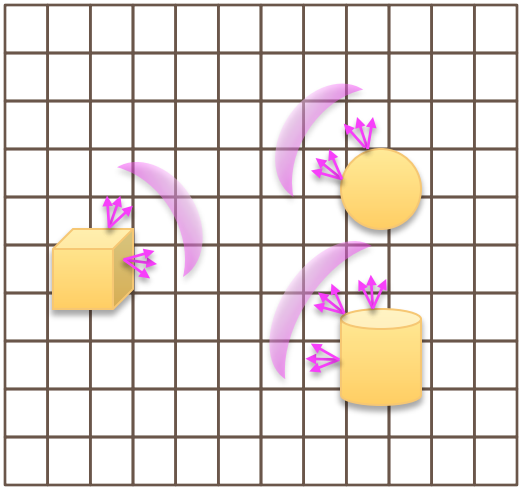


Multiple Scattering

- *Real-Time Multiple Scattering with Light Propagation*

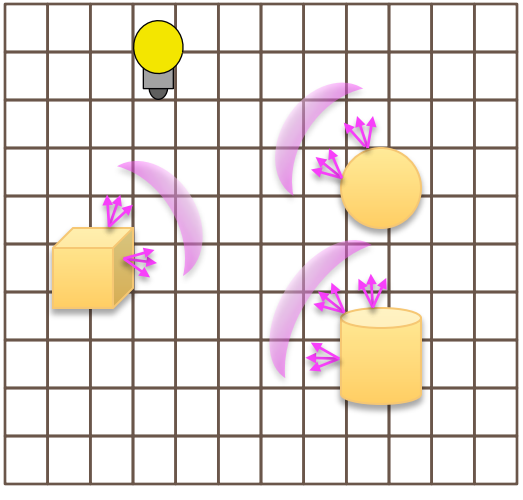
Multiple Scattering with LPVs

- Light Propagation Volumes:
  - Render scene from light
  - Inject 3D grid with radiance at cells containing lit surfaces
    - Each cell stores light as SH
    - Occluders represented by SH
  - Iteratively propagate the light from cell to cell



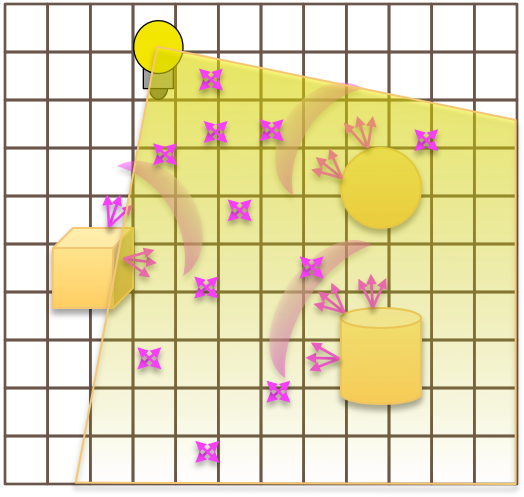
Multiple Scattering with LPVs

- Light Propagation Volumes:
  - Render scene from light
  - Inject 3D grid with radiance at cells containing lit surfaces
    - Each cell stores light as SH
    - Occluders represented by SH
  - Iteratively propagate the light from cell to cell



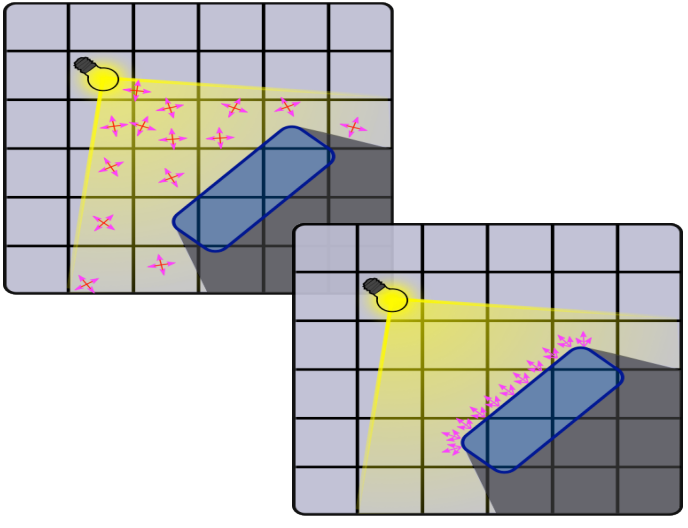
Multiple Scattering with LPVs

- Light Propagation Volumes:
  - Render scene from light
  - Inject 3D grid with radiance at cells containing lit surfaces
    - Each cell stores light as SH
    - Occluders represented by SH
  - Iteratively propagate the light from cell to cell
- Add multiple Scattering:
  - Inject radiance from single scattering and propagate.
    - Identified by shadow map
  - Ray-march grid to visualize multiple scattering
  - (Superimpose SV-based single scattering)

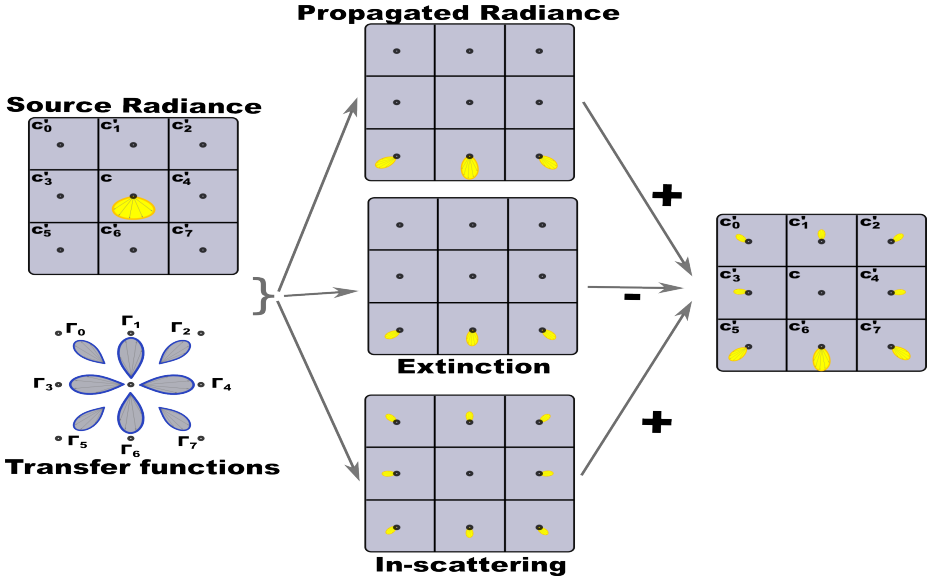


Light Propagation - Injection

- Each frame: start by generating an initial radiance distribution
- Inject
  - radiance from single scattering
    - identified by shadow map
  - radiance from reflective shadow maps
    - as in original LPV method

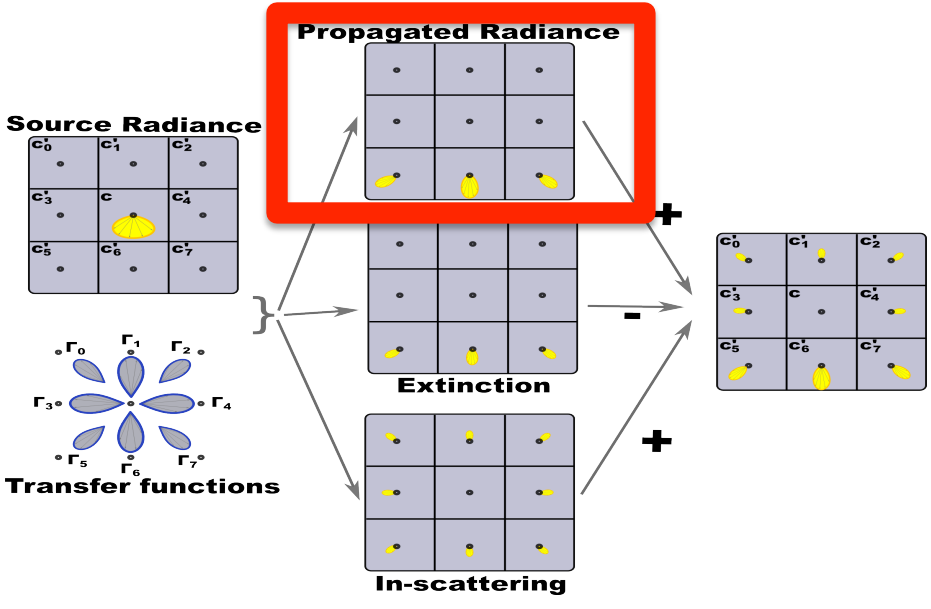


Propagation Scheme

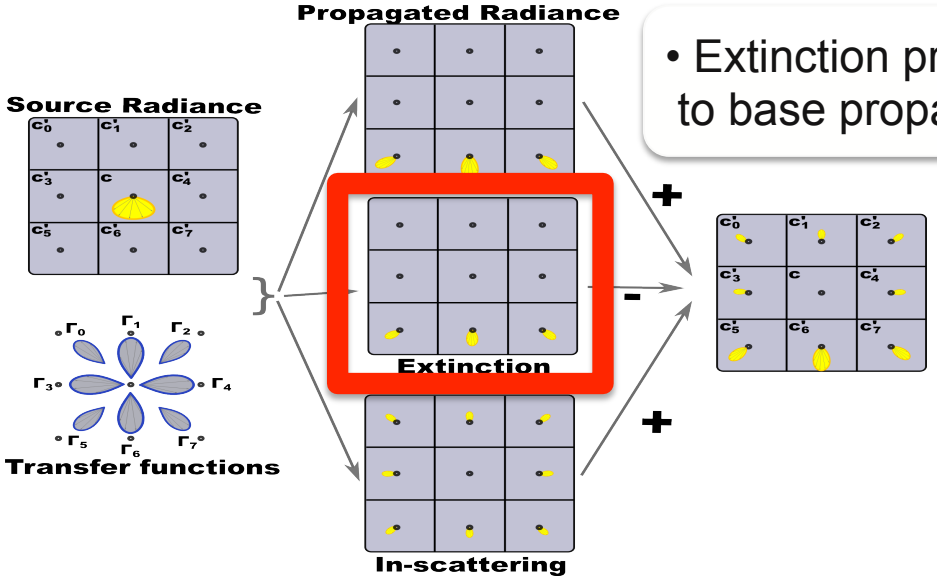


Propagation Scheme

- Base propagation same as LPV

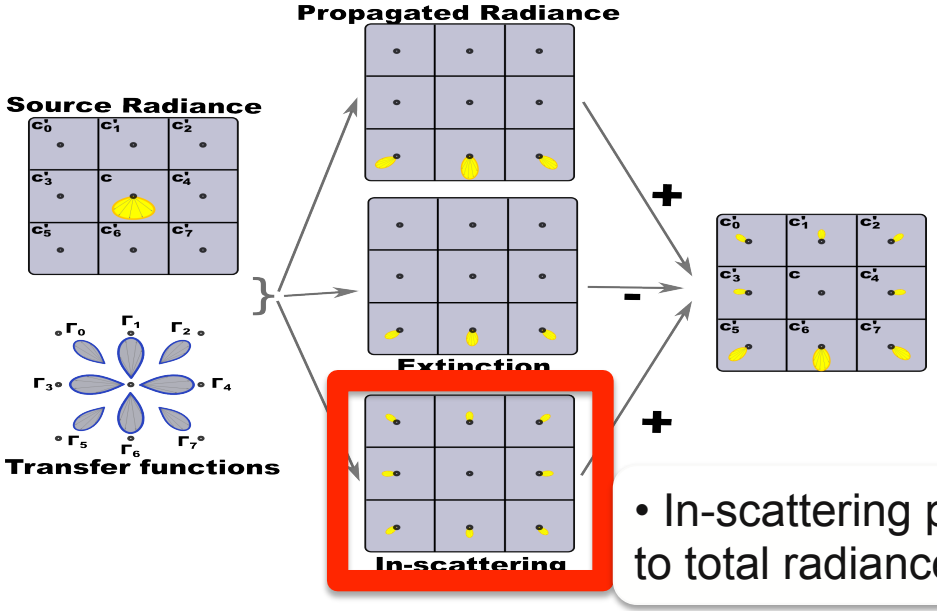


Propagation Scheme



- Extinction proportional to base propagation

Propagation Scheme

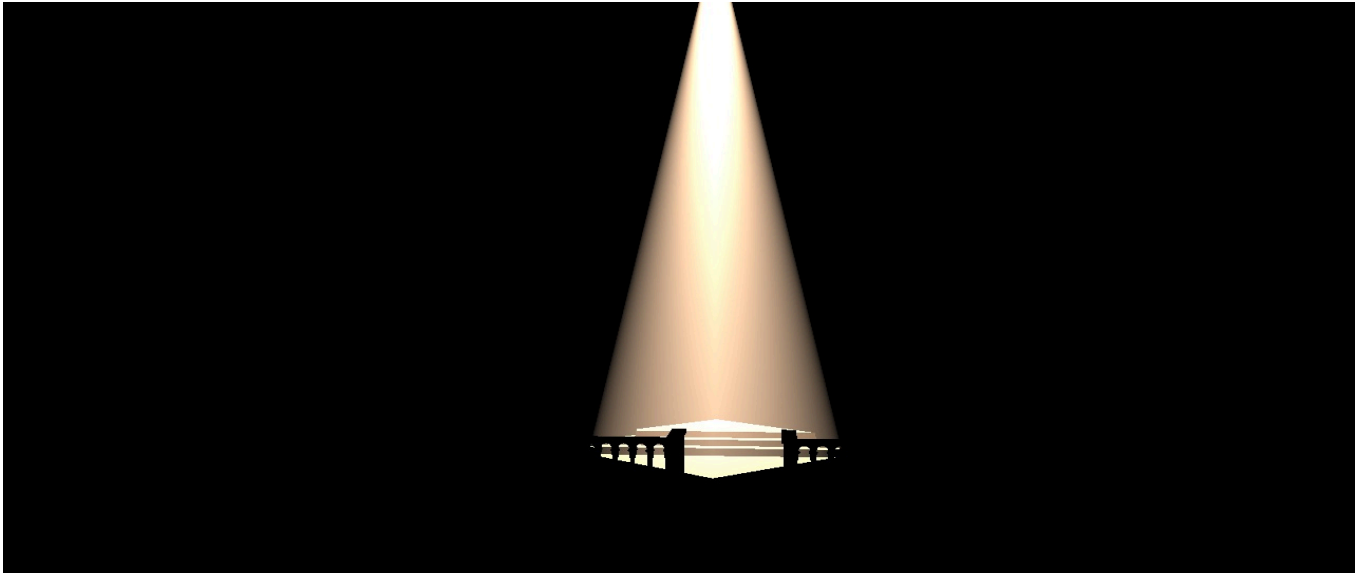


Results

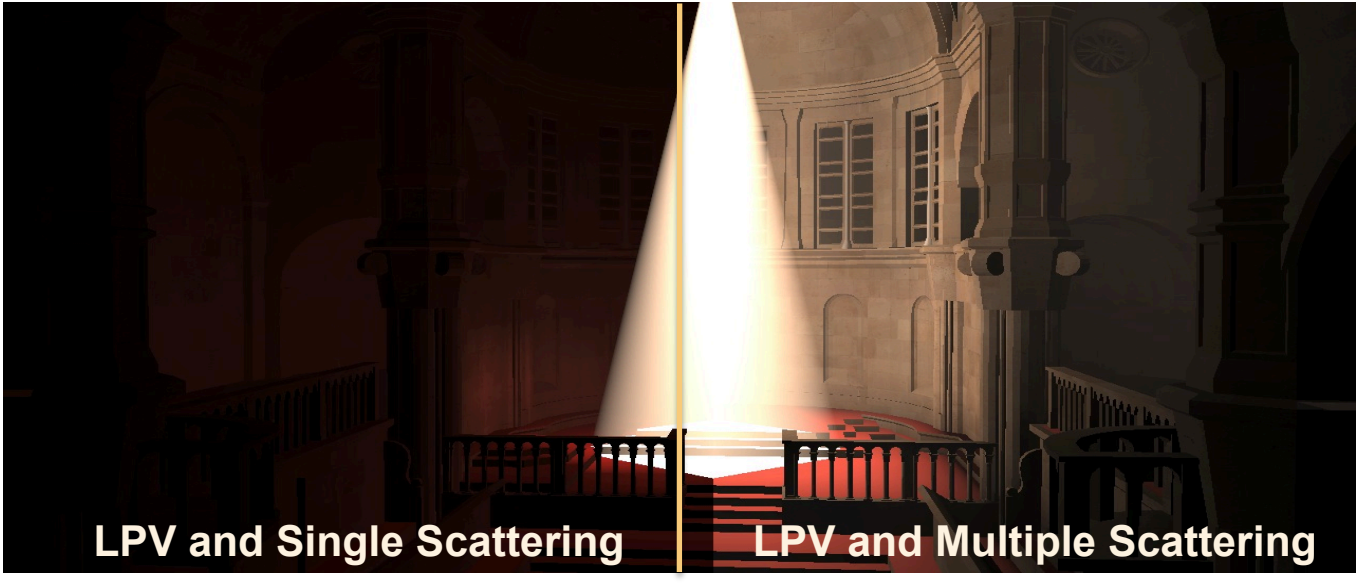
LPV Only

Results

Single Scattering Only



Results



Results

Multiple Scattering

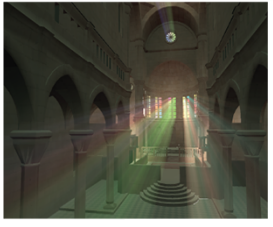
Before and After

Before and After

Results

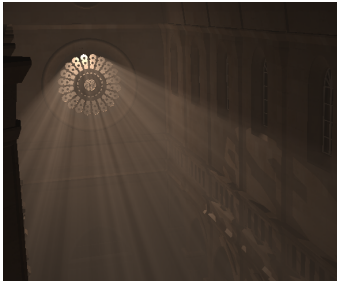
Multiple Scattering

Summary – Homogeneous Participating Media



Single Scattering:

- Chen et al. 2011:
  - Textured Lights
- Wyman 2011:
  - Fastest
  - Only shadows – not unique in-scattering per ray segment
- Billeter et al. 2010
  - No need for ray marching
  - Simplest – OpenGL 2.x



Multiple Scattering:

- Billeter et al. 2012


Volumetric Shadows
THE END