Participating Media

Part II: interactive methods, atmosphere and clouds

Oskar Elek
MFF UK Prague

Computer Graphics
Charles University

Outline

- Motivation
- Introduction
- Properties of participating media
- Rendering equation
- Storage strategies
- Non-interactive rendering strategies
- Part I revision
- Interactive rendering strategies
- Atmospheric rendering
- Cloud rendering
- (References)

Outline

- Motivation
- Introduction
- Properties of participating media
- Rendering equation
- Storage strategies
- Non-interactive rendering strategies
- **Part I revision**
- Interactive rendering strategies
- Atmospheric rendering
- Cloud rendering
- (References)

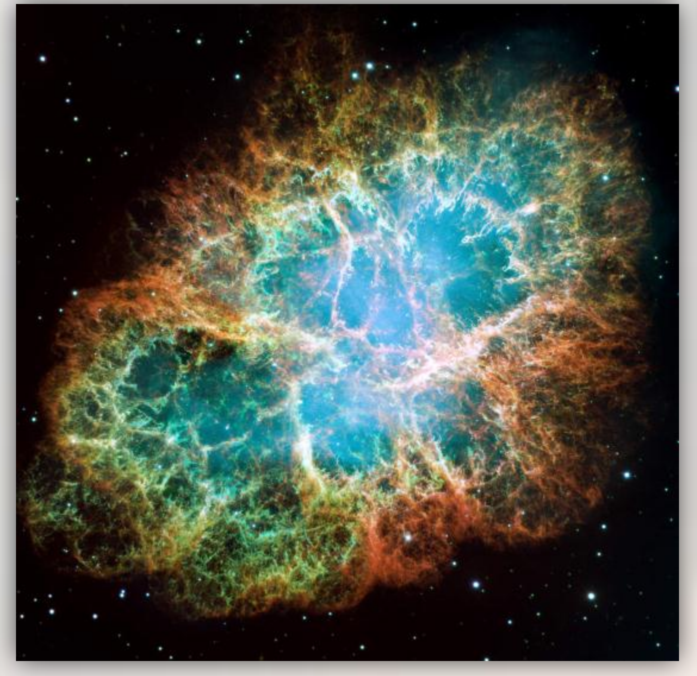
Motivation – fluids

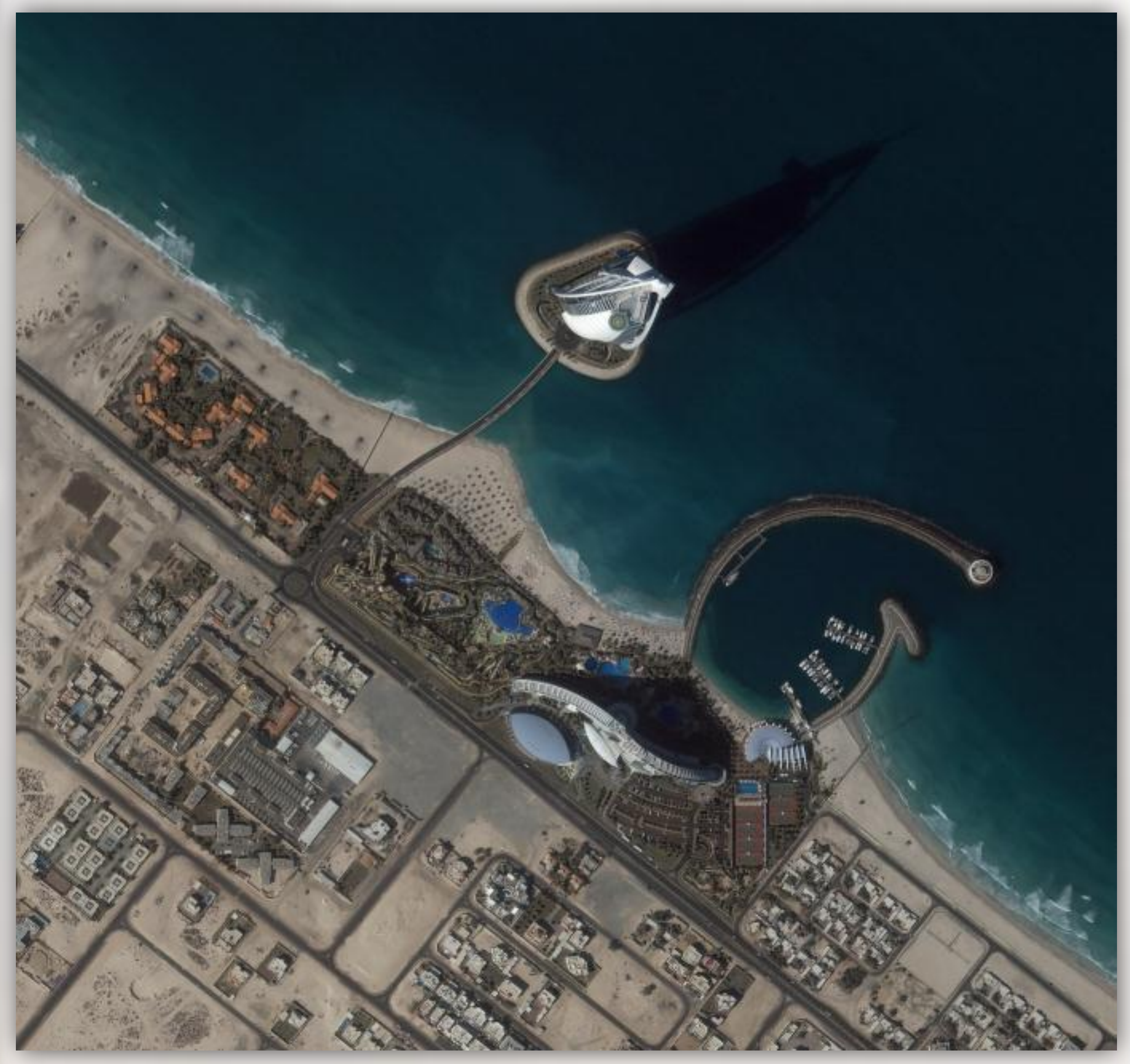
Motivation – solids





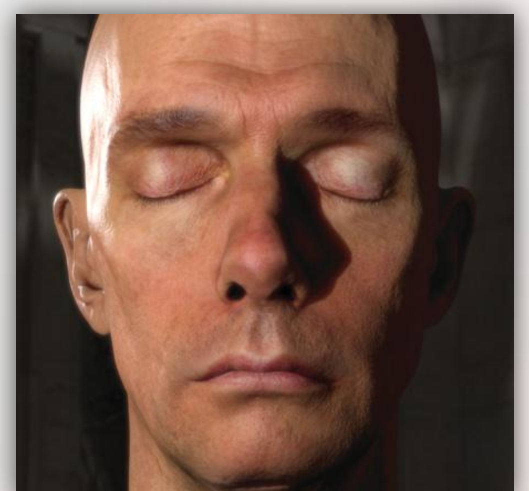

Motivation – beyond rendering





Introduction

- What are participating media (PMa)?
  - General meaning
  - CG connotation
- Why are PMa more challenging than B-rep rendering?
  - At lease 1 DoF more
  - Costly representation
- General scattering vs. sub-surface scattering (BSSRDF)



- 4 basic event types in PMa
  - Single vs. multiple scattering

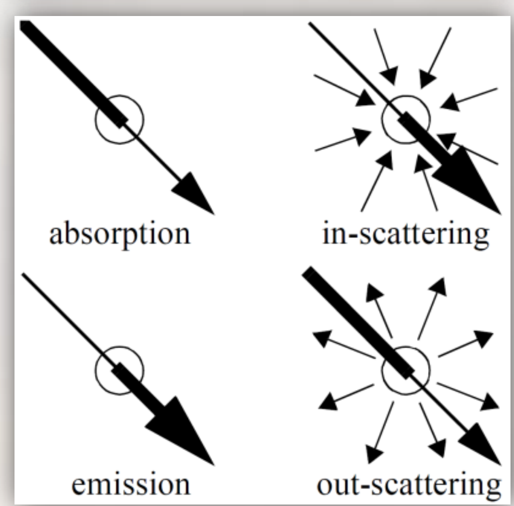
A

E

A
E

S

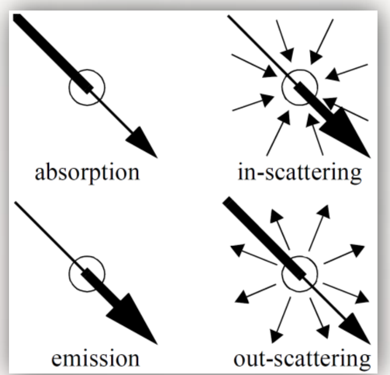
A
E
S



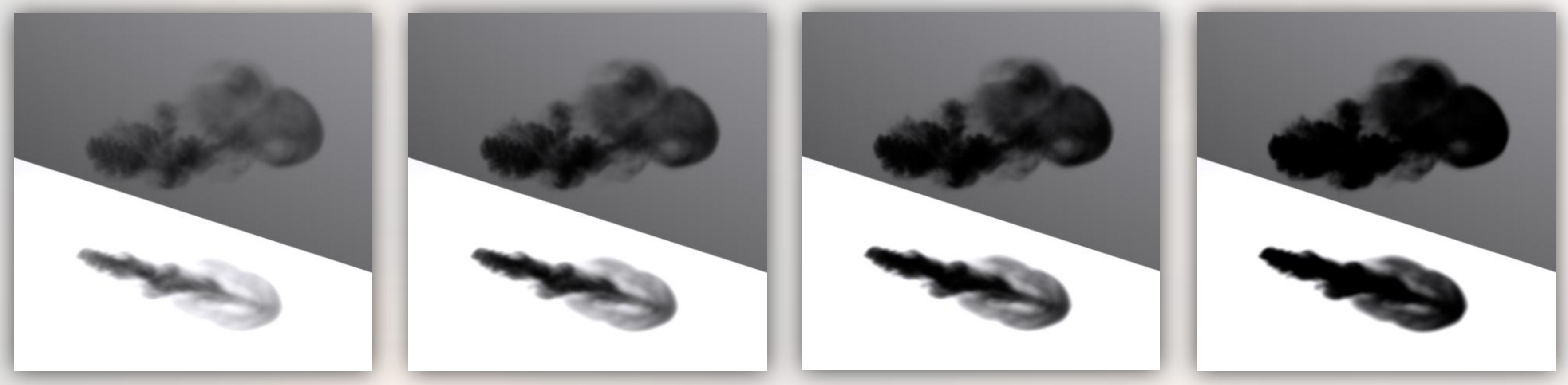
Properties – medium composition

Computer Graphics Charles University

- Main property – medium (particle) density
- Derived characteristics:
  - σ_e – emission coefficient [m^{-1}]
  - σ_a – absorption coefficient [m^{-1}]
  - σ_s – scattering coefficient [m^{-1}]
  - σ_t – extinction coefficient ($\sigma_a + \sigma_s$)
  - $e^{-\sigma}$ dependency ($\sigma = 2 \approx 13.6\%$ transmittance)



σ_a=5 σ_a=10 σ_a=15 σ_a=30

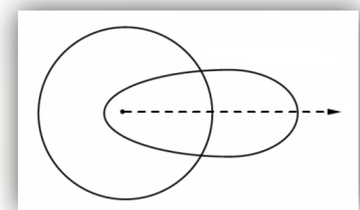


- More particle types → linear combination of coefficients

Properties – scattering directionality

Computer Graphics Charles University

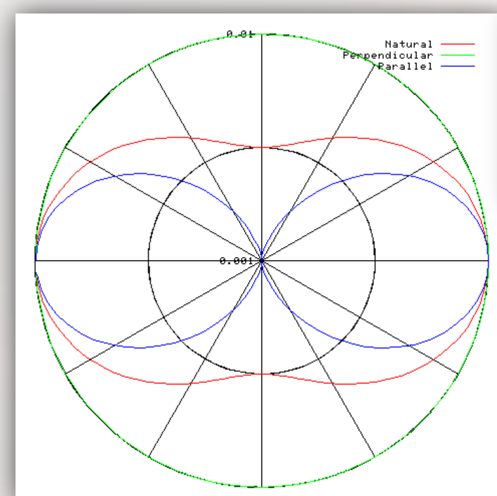
- Phase function
  - Describes directional distribution of scattered light
  - Equivalent of BRDF for surfaces (probability density)
  - Denotes scattering anisotropy (equivalent of diffuse vs. glossy surfaces)

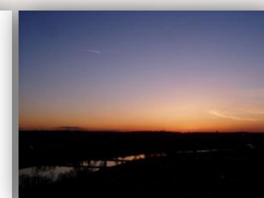


- We recognize Rayleigh and Mie (light) scattering

Uniform:

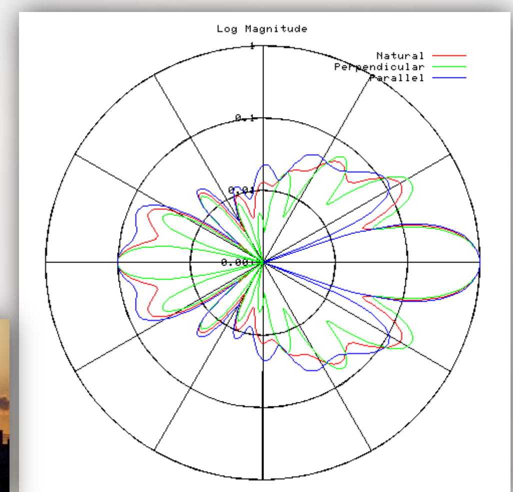
$$p_{\text{uni}}(\theta) = \frac{1}{4\pi}$$





Rayleigh (λ^{-4}-dependent):

$$p_{\text{ray}}(\theta) = \frac{3}{4}(1 + \cos^2(\theta))$$

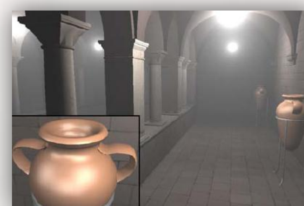


Mie (Henyey-Greenstein approximation):

$$p_{\text{hg}}(\theta, g) = \frac{1}{4\pi} \frac{1-g^2}{(1+g^2-2g\cos(\theta))^{3/2}}$$

Properties – other PMa characteristics

- Albedo – efficiency of a single scattering event
  - Defined as: $100 * \sigma_s / (\sigma_a + \sigma_s)$ [%]
  - Mean number of scattering events depends on it
- Medium homogeneousness



- Medium anisotropy (sundogs, parhelia)

- Shape complexity

- Standard (areal) RE

$$L_o(x,\vec{\omega}) = L_e(x,\vec{\omega}) + \int_{2\pi} f_r(x,\vec{\omega}',\vec{\omega})L_i(x,\vec{\omega}')(-\vec{\omega}'\cdot\vec{n})d\vec{\omega}'$$

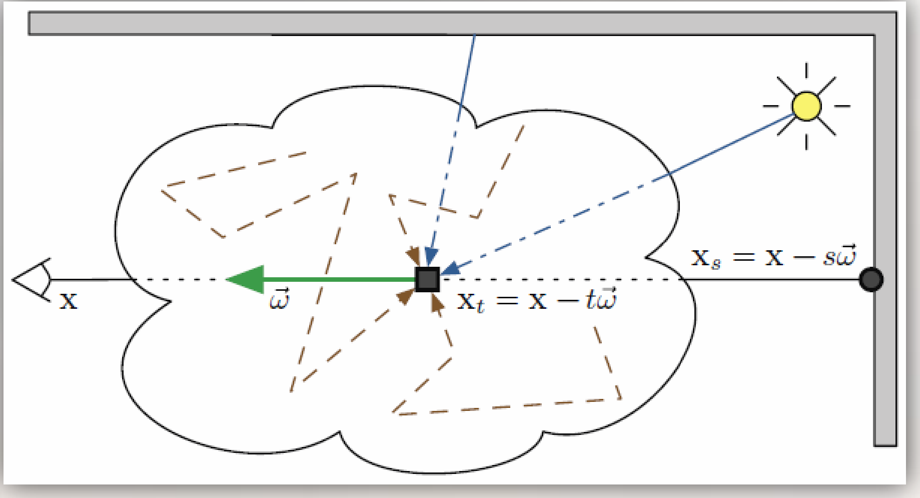
- Volume RE, directional formulation

$$L(x,\vec{\omega}) = \int_0^s T_r(x \leftrightarrow x_t)\sigma_s(x_t)L_i(x_t,\vec{\omega})dt + T_r(x \leftrightarrow x_s)L(x_s,\vec{\omega})$$

$$L_i(x,\vec{\omega}) = \int_{4\pi} p(x,\vec{\omega}',\vec{\omega})L(x,\vec{\omega}')d\vec{\omega}'$$

$$\tau(x \leftrightarrow x') = \int_x^{x'} \sigma_t(u)du$$

$$T_r(x \leftrightarrow x') = e^{-\tau(x\leftrightarrow x')}$$

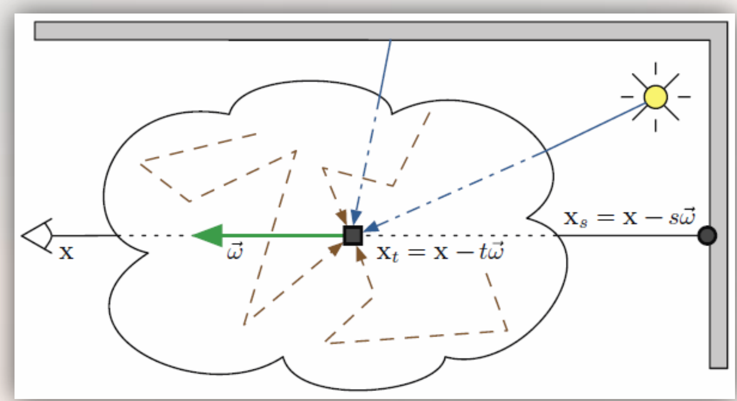


- Standard (areal) RE
- Volume RE, directional formulation

$$L(x,\vec{\omega}) = \int_0^s T_r(x \leftrightarrow x_t)\sigma_s(x_t)L_i(x_t,\vec{\omega})dt + T_r(x \leftrightarrow x_s)L(x_s,\vec{\omega})$$

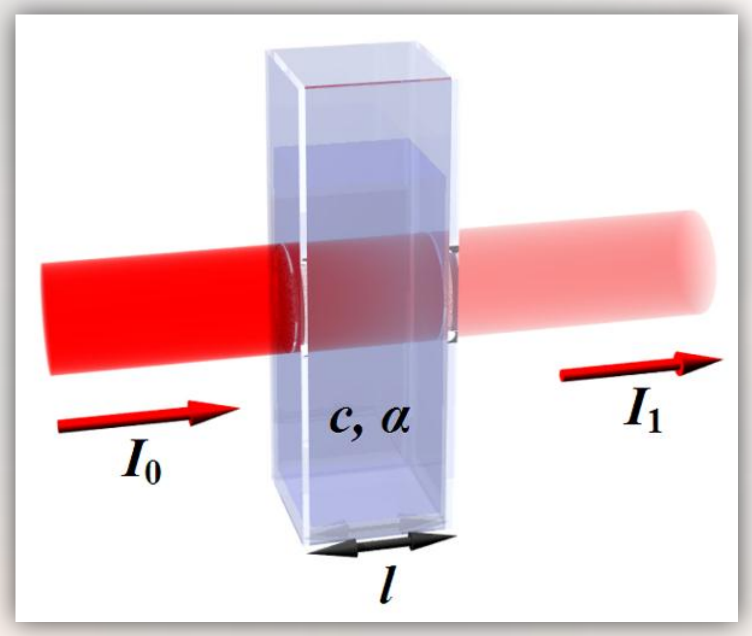
- Volume RE, differential formulation (energy transport equation)

$$\frac{dL(x,\vec{\omega})}{dx} = -\sigma_t L(x,\vec{\omega}) + \sigma_a L_e(x,\vec{\omega}) + \sigma_s \int_{4\pi} L(x,\vec{\omega}')p(x,\vec{\omega}',\vec{\omega})d\vec{\omega}'$$



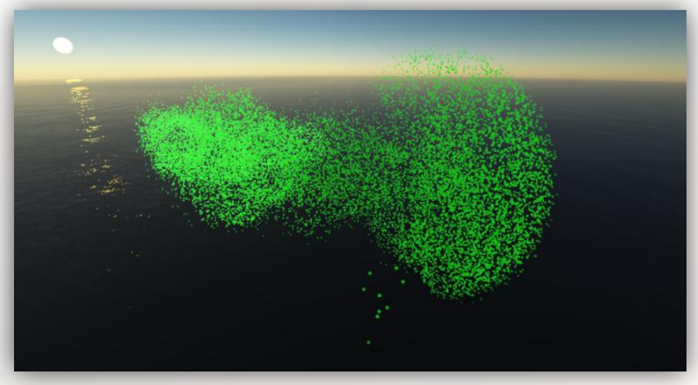
- Defines relation of medium composition to its light attenuating properties

$$T_r = e^{-\sigma_t l} \qquad dI_x = -\sigma_t I_x dx$$

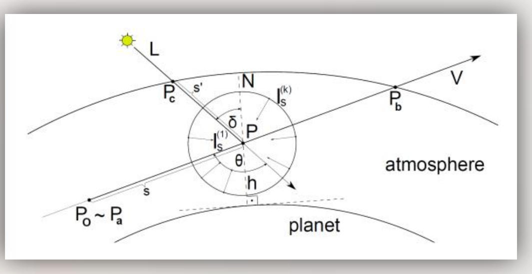


- 3D density grids

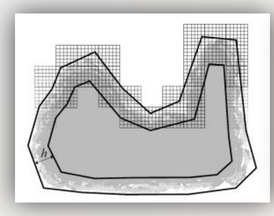

- Point sets

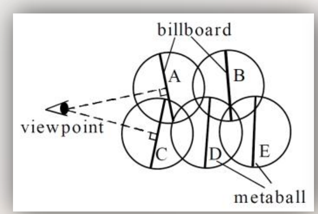


- Analytically defined

- Combined

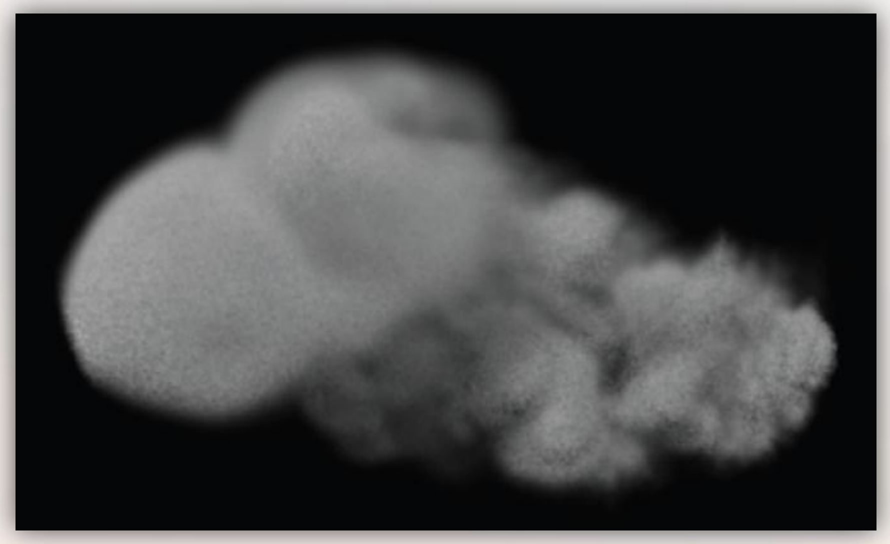


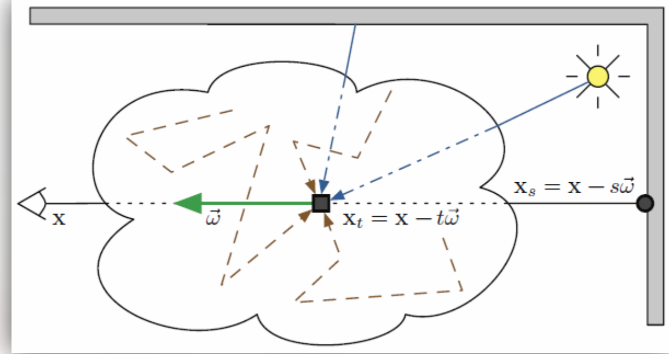


- Similar to areal PT, solves directional VRE by generating random walks in the medium

$$L(x,\vec{\omega}) = \int_0^s T_r(x \leftrightarrow x_t)\sigma_s(x_t)L_i(x_t,\vec{\omega})dt + T_r(x \leftrightarrow x_s)L(x_s,\vec{\omega})$$

- Evaluation
  - Pros: simplicity, not limited to any PMa range, unbiasedness
  - Cons: speed (in certain cases almost pathological), high variance

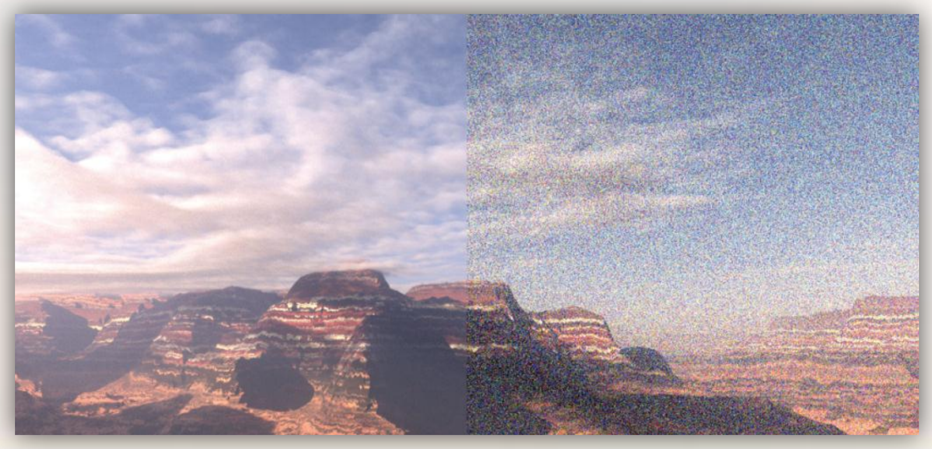




- Choose randomly
- Taking into account extinction

$$\int_0^{d_{next}} \sigma_t(s)\, ds = -\ln(1-\xi)$$

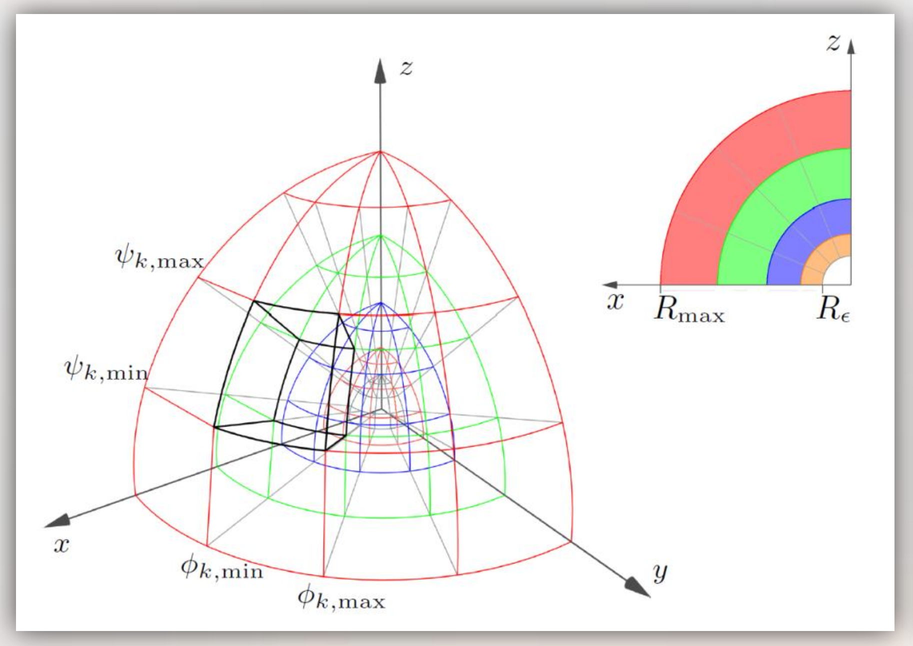
  - Ray marching
  - Woodcock tracking
    - Increment *x* by $-\ln(1-\xi_1)/\sigma_{s_M}$ until $\sigma_s(x)/\sigma_{s_M} < \xi_2$
    - Pros: fast (using adaptive kD-tree scheme), unbiasedness
    - Cons: slightly more complicated

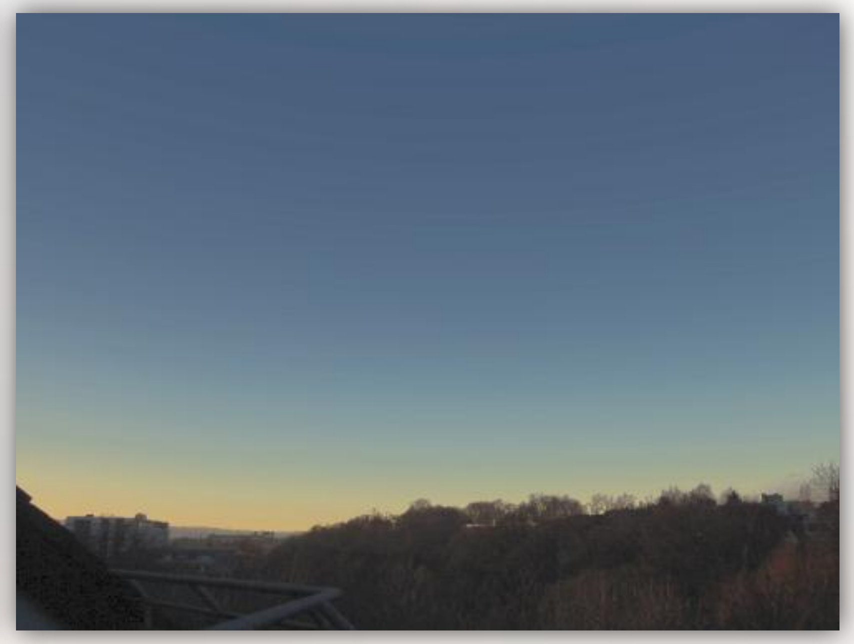


- Similar to areal radiosity, solves energy transport equation

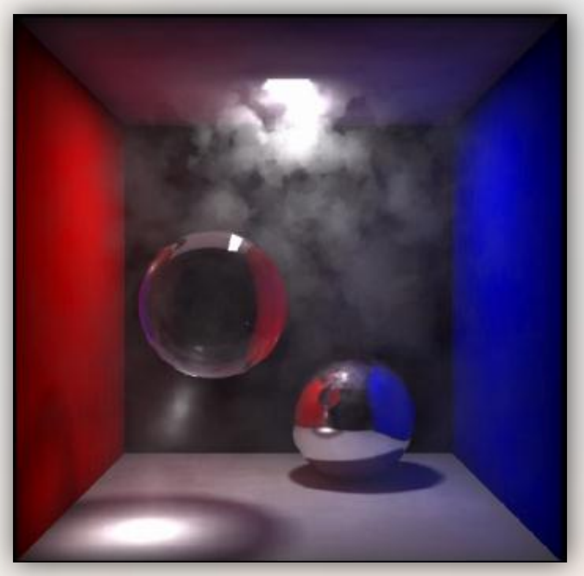
$$\frac{dL(x,\vec{\omega})}{dx} = -\sigma_t L(x,\vec{\omega}) + \sigma_a L_e(x,\vec{\omega}) + \sigma_s \int_{4\pi} L(x,\vec{\omega}')p(x,\vec{\omega}',\vec{\omega})d\vec{\omega}'$$

  - Pros: (theoretically) unbiased, linear scaling with number of scattering orders, computes energy state of the entire scene
  - Cons: rather slow, high storage requirements, problems with inhomogeneous media and additional objects in scene





- Once again, similar to areal PM
  - Generates random walks in the medium, stores photon on each scattering event
  - Gathering more complicated → beam radiance estimate
- Evaluation – widely used
  - Pros: fast, easy extension from B-rep renderers, robust
  - Cons: biasedness, necessary storage of photons



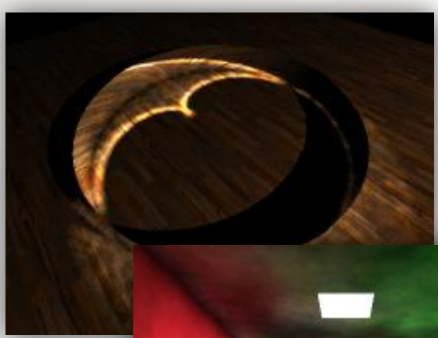
End (part I)

Outline

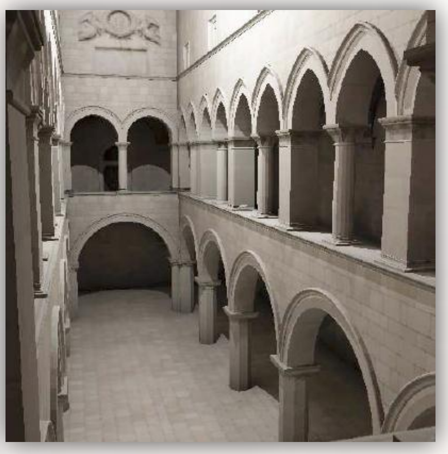
- Motivation
- Introduction
- Properties of participating media
- Rendering equation
- Storage strategies
- Non-interactive rendering strategies
- Part I revision
- **Interactive rendering strategies**
- Atmospheric rendering
- Cloud rendering
- (References)

- Non-interactive vs. interactive methods
  - In surface rendering, these converge
  - Not that much in volume rendering (until recently)

Instant radiosity



Photon mapping on GPU

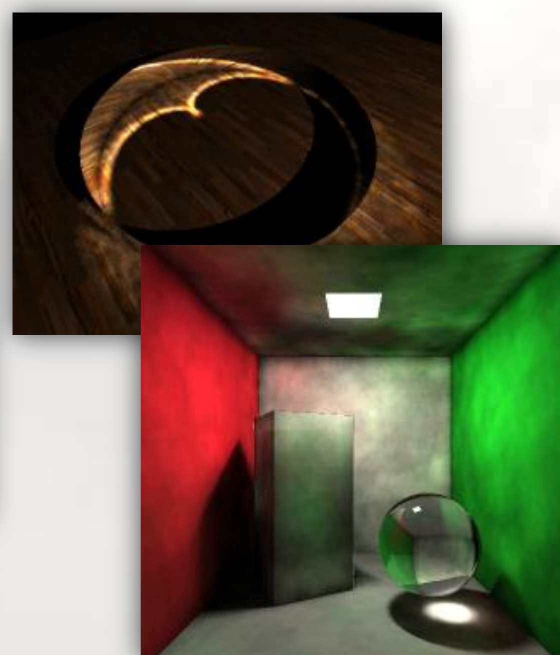


Photon streaming

Interactive rendering strategies

- Non-interactive vs. interactive methods
  - In surface rendering, these converge
  - Not that much in volume rendering (until recently)

Instant radiosity

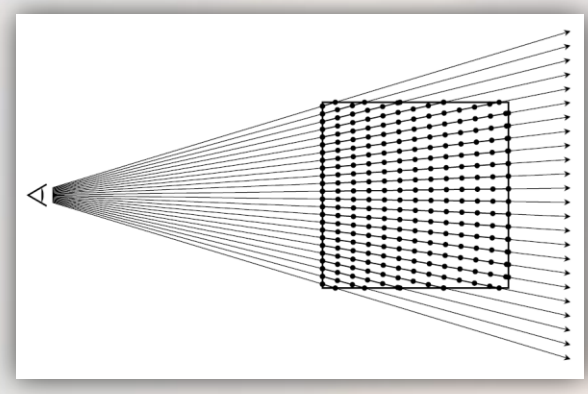


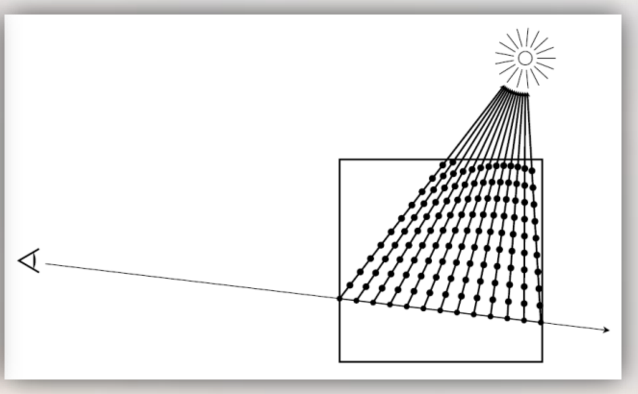
Photon mapping on GPU

Photon streaming

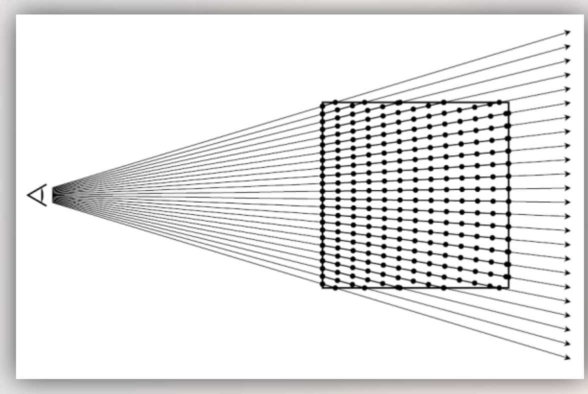
- Interactive methods always rely on simplifying assumptions, limitations, special cases etc.

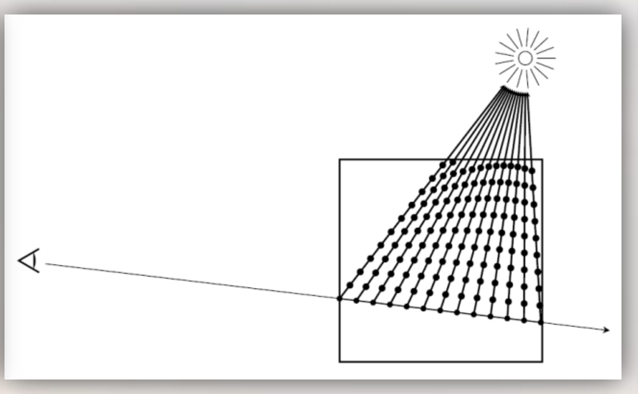
- Backward method



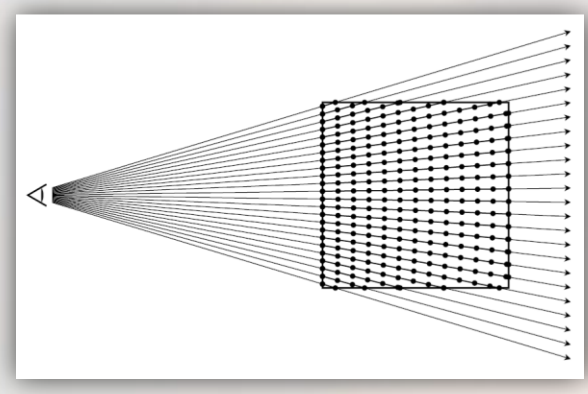


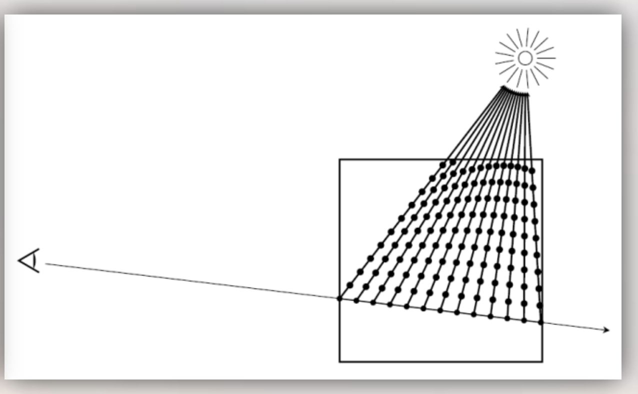
- Backward method
- Practically limited to single-scattering
- Hardly uses anything else than 3D grids



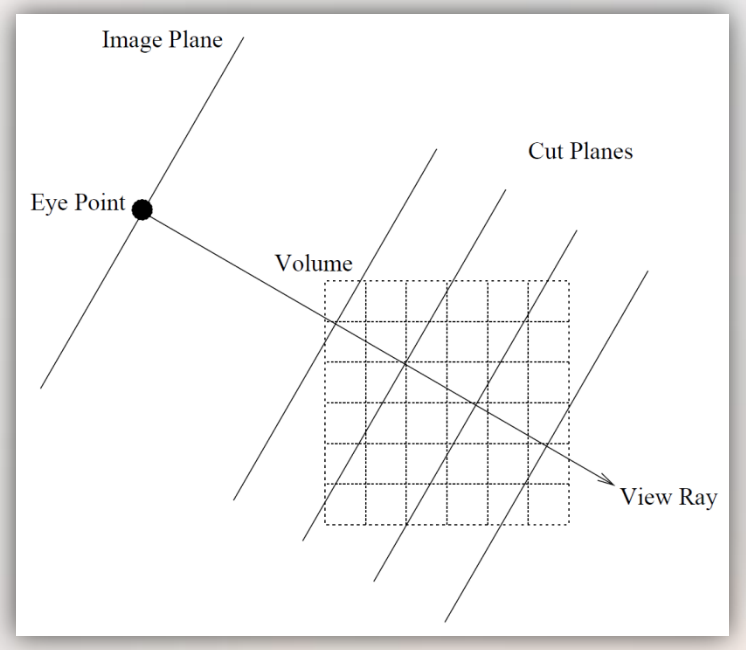


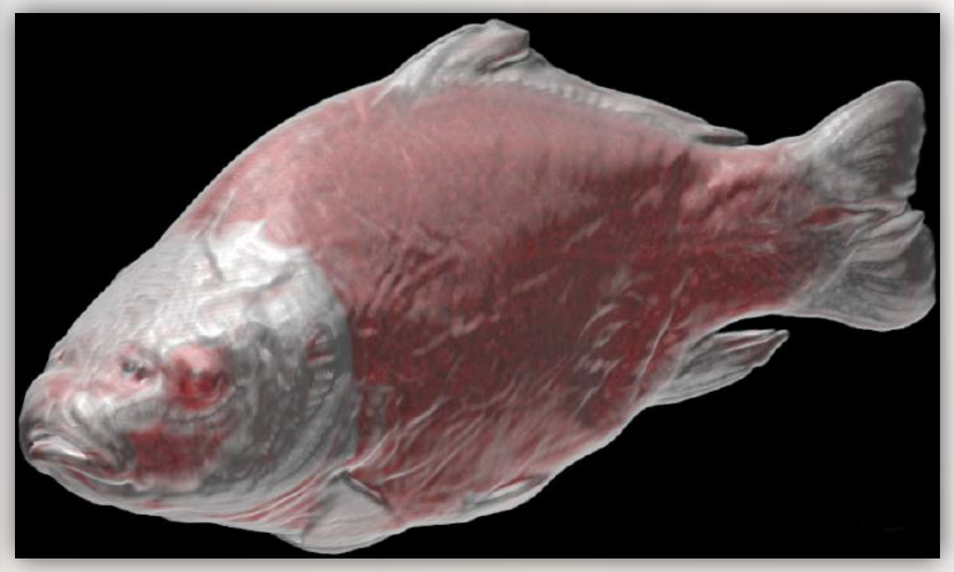
- Backward method
- Practically limited to single-scattering
- Hardly uses anything else than 3D grids
- Evaluation
    - Pros: simplicity
    - Cons: slow (w/o extensive optimizations), prone to aliasing, limited to single-scattering



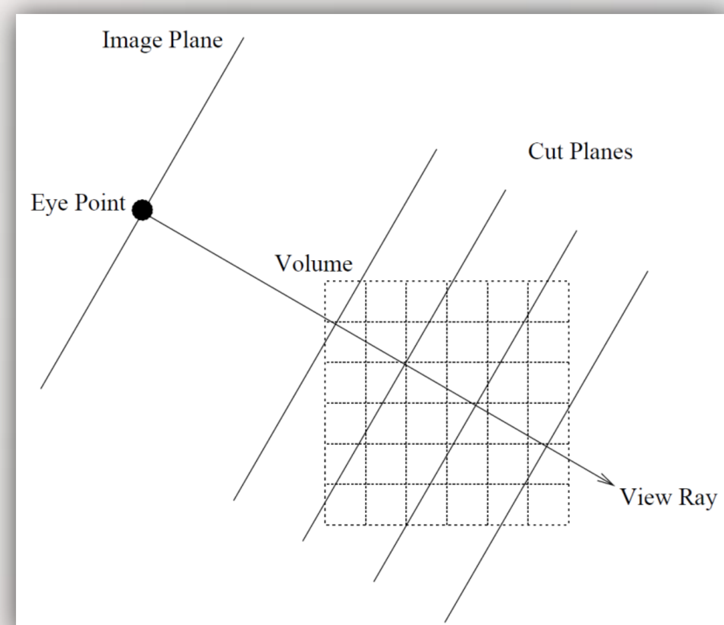


- Forward method

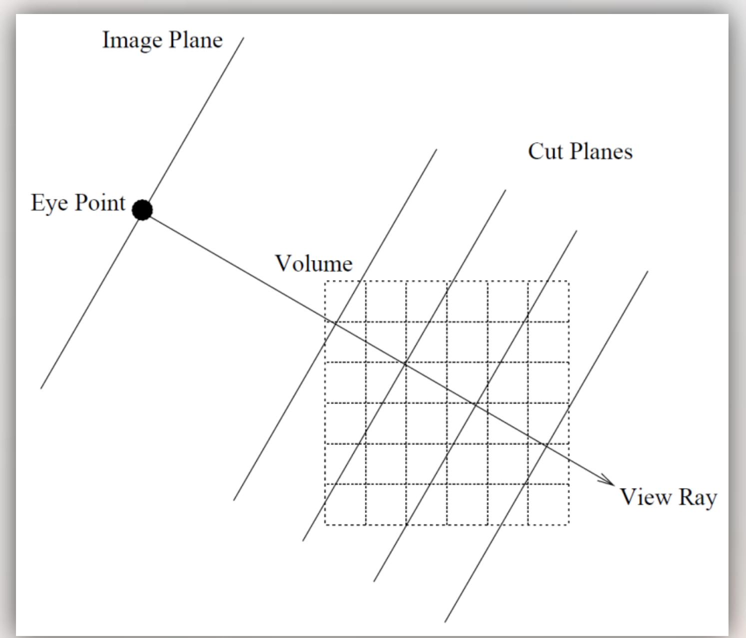


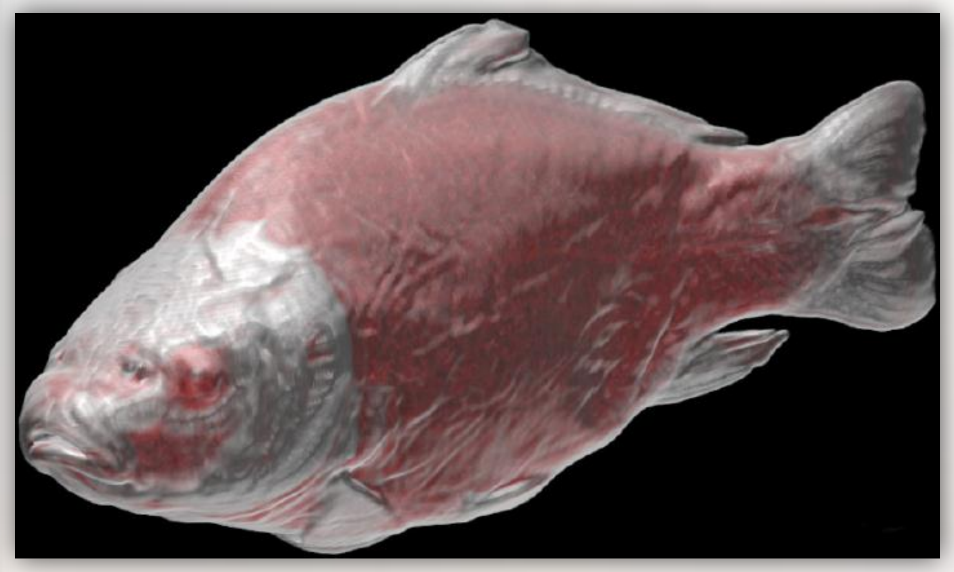


- Forward method
- No intrinsic way to compute light propagation
- Even more tied to 3D grids
- GPU-adaptation of ray-marching

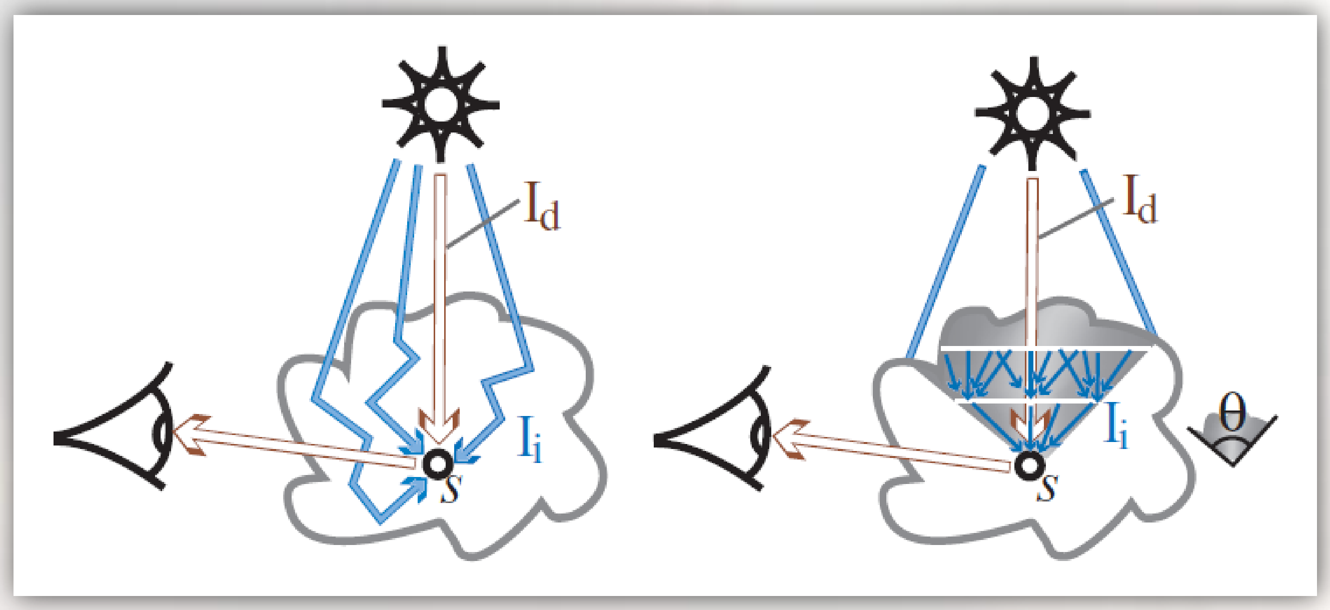


- Forward method
- No intrinsic way to compute light propagation
- Even more tied to 3D grids
- GPU-adaptation of ray-marching
  - Pros: fast, maps well to GPU
  - Cons: no light propagation, prone to aliasing and slicing artefacts

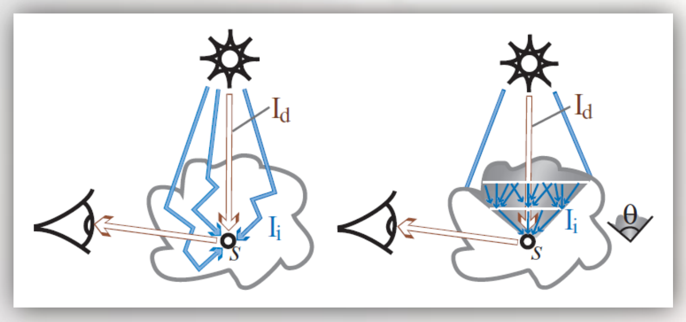




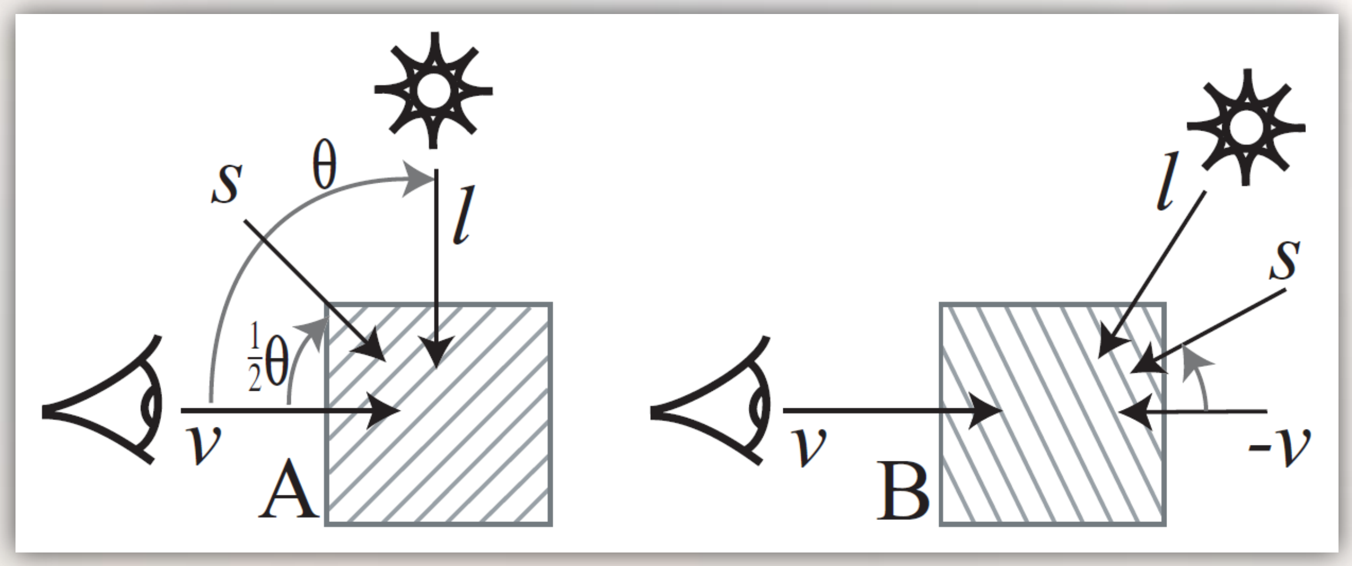
- Extension of slice-based rendering, adds light propagation computation



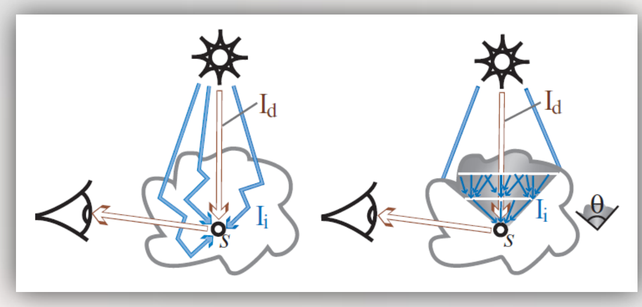
- Extension of slice-based rendering, adds light propagation computation

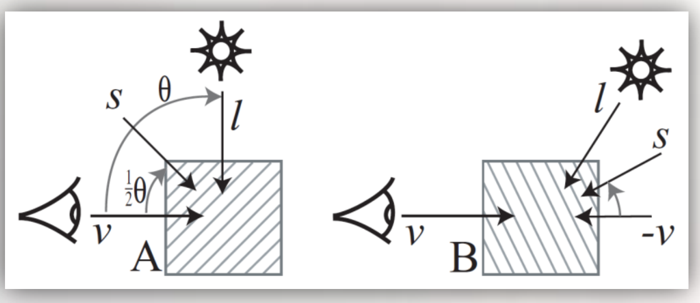


- Slicing in the direction perpendicular to half-vector



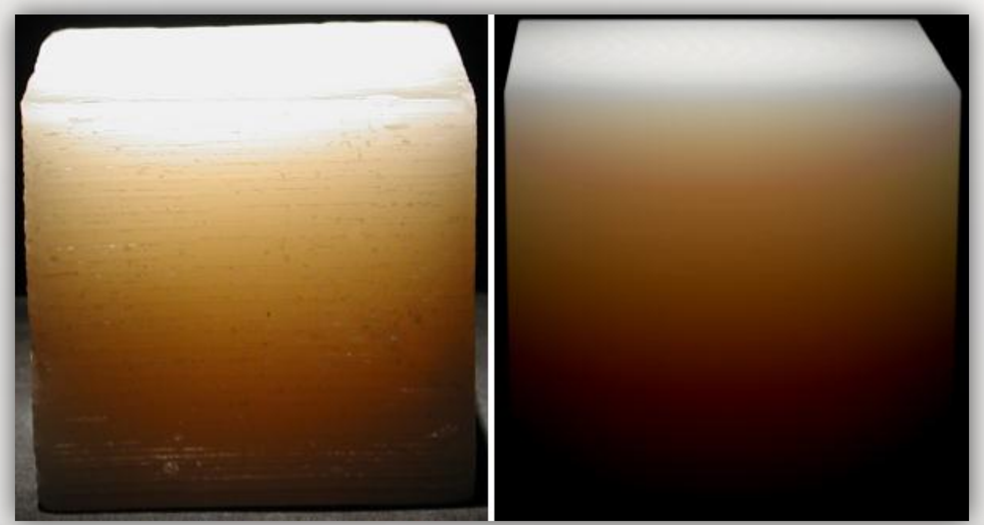
- Extension of slice-based rendering, adds light propagation computation
- Slicing in the direction perpendicular to half-vector

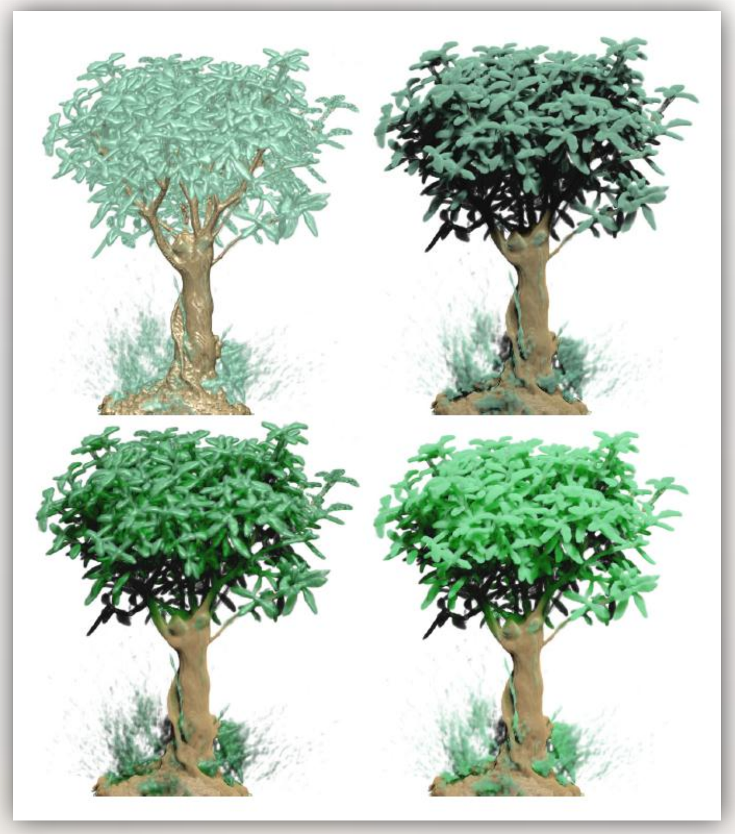




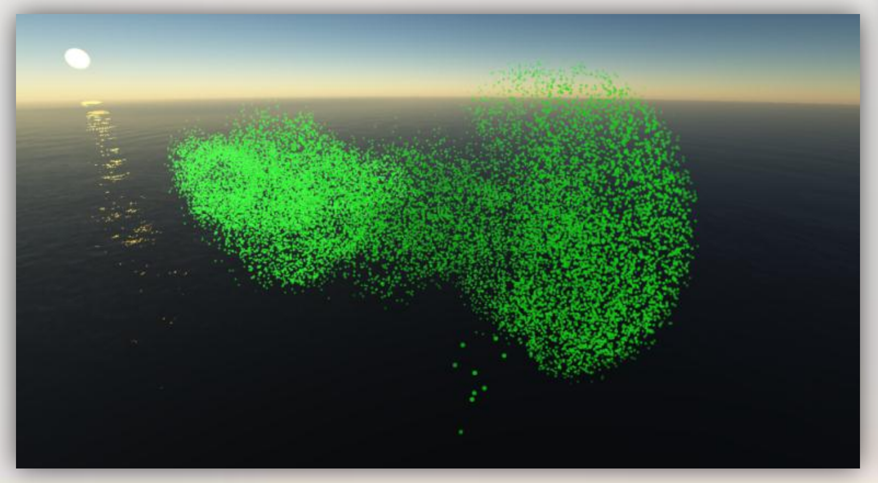
- Evaluation
  - Pros: comparatively fast, adds light propagation scheme
  - Cons: partly empirical

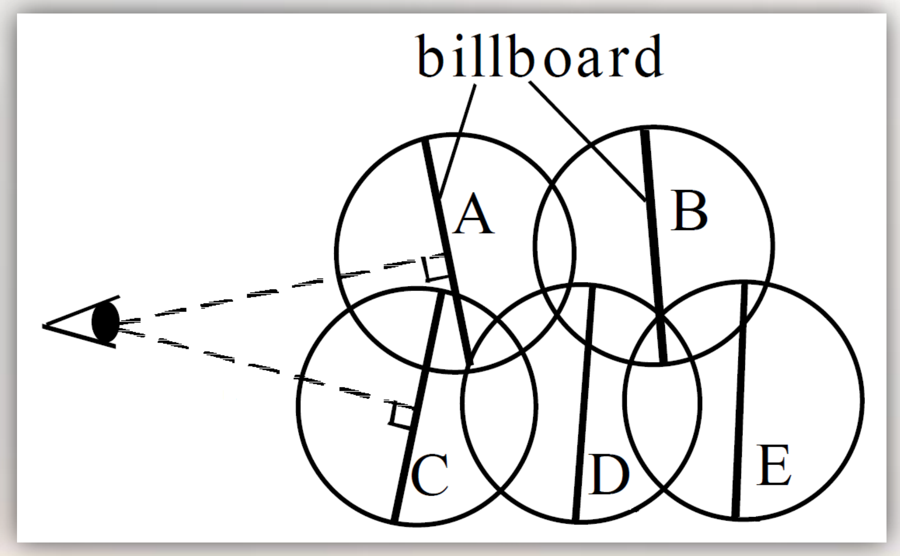
- Extension of slice-based rendering, adds light propagation computation
- Slicing in the direction perpendicular to half-vector



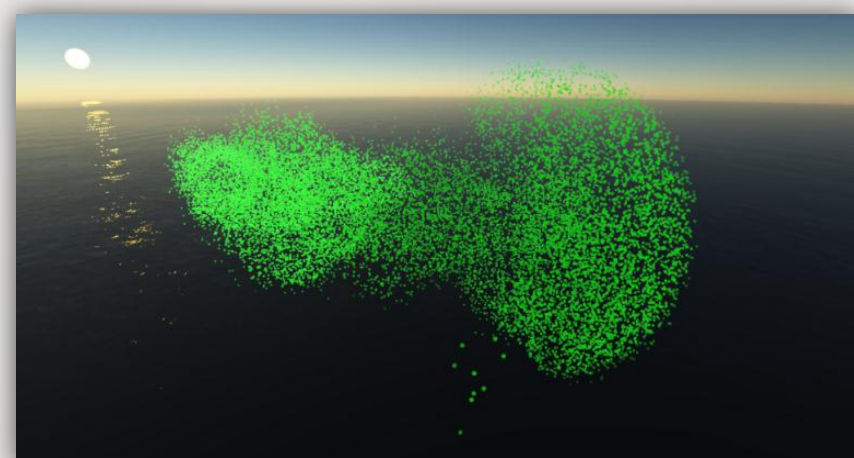


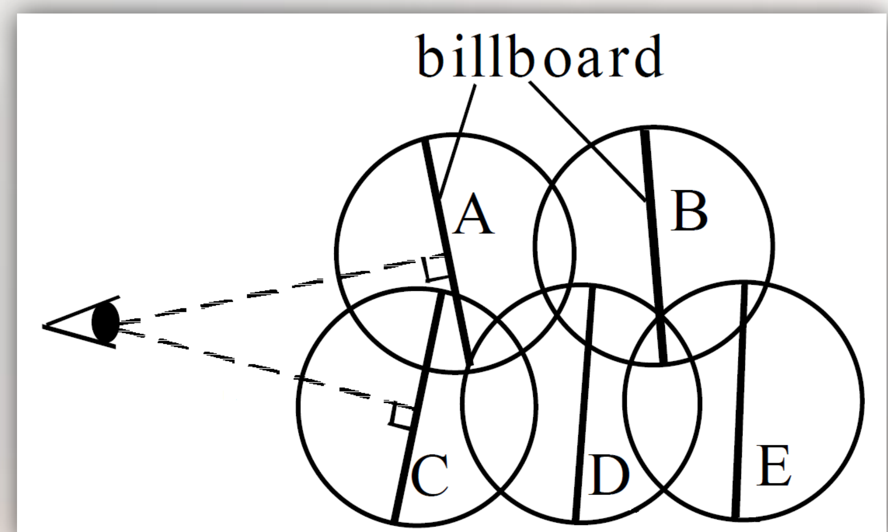
- Forward method, widely used in game engines



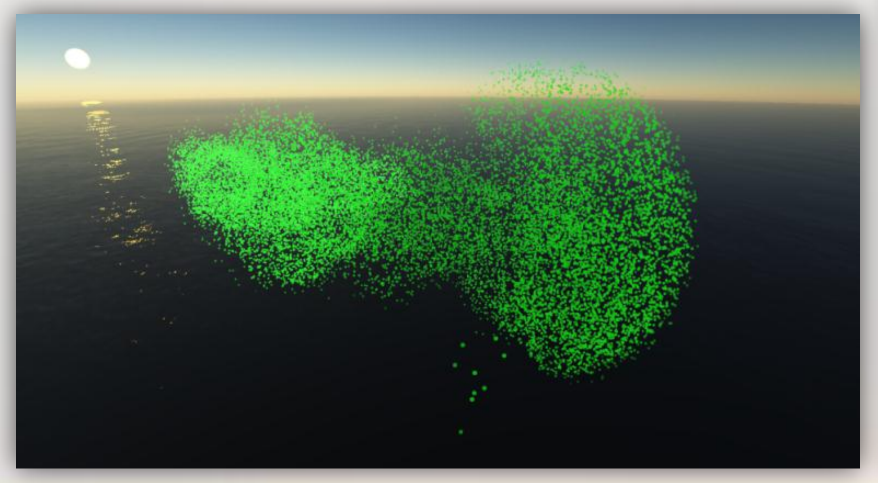


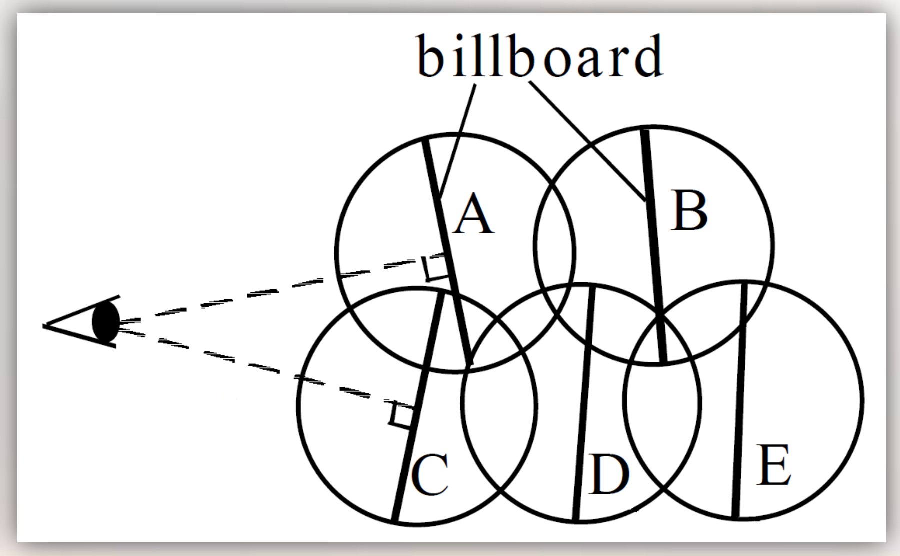
- Forward method, widely used in game engines
- Billboards correspond to units of volume
- Mostly use point/particle-based medium representations





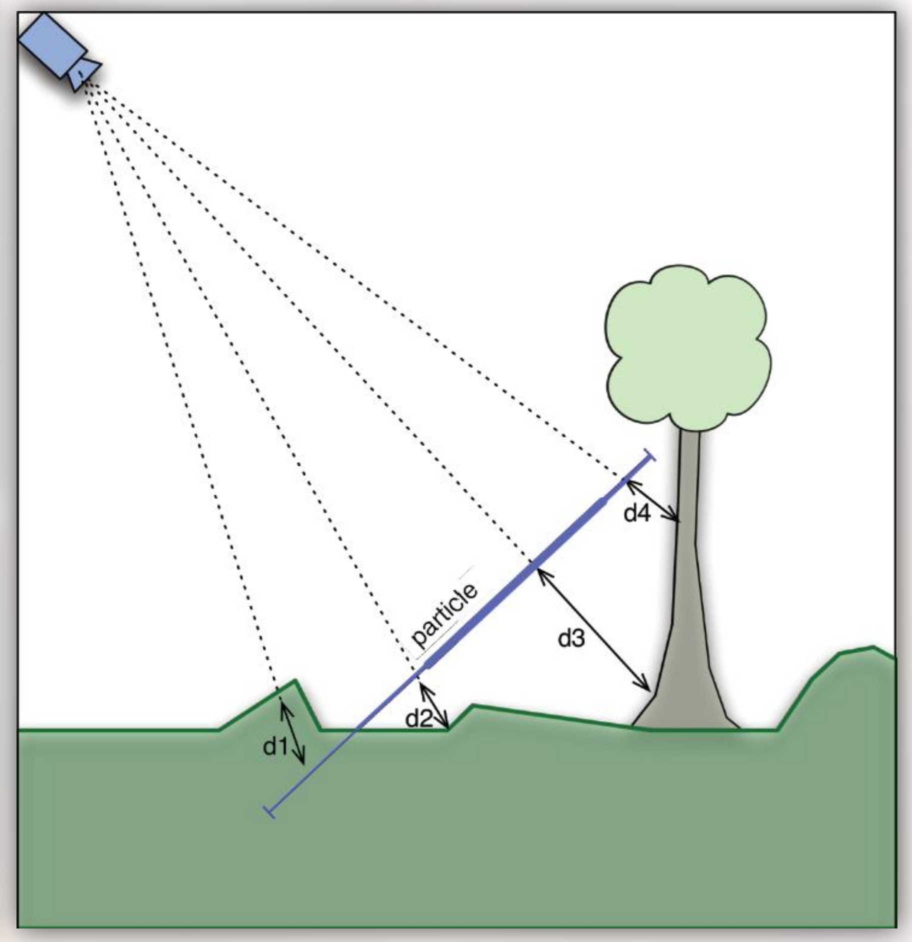
- Forward method, widely used in game engines
- Billboards correspond to units of volume
- Mostly use point/particle-based medium representations
- Evaluation
  - Pros: simple, fast, map well to GPU, easy to animate
  - Cons: low accuracy, again no intrinsic light propagation computation, edging artefacts





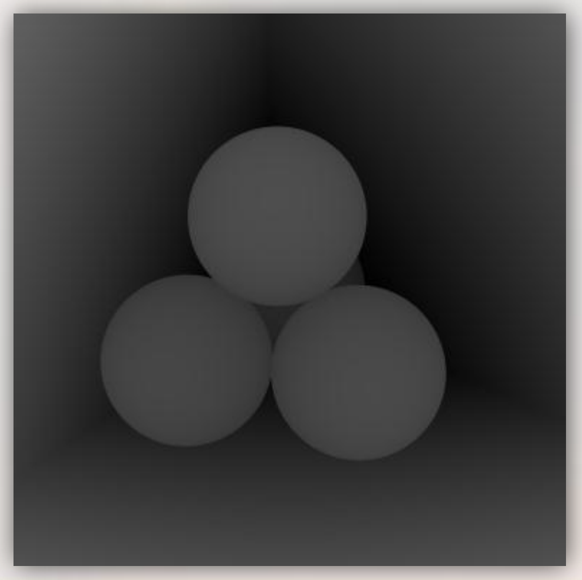
- Extension of billboard-based rendering, tackles the edging artefacts problem

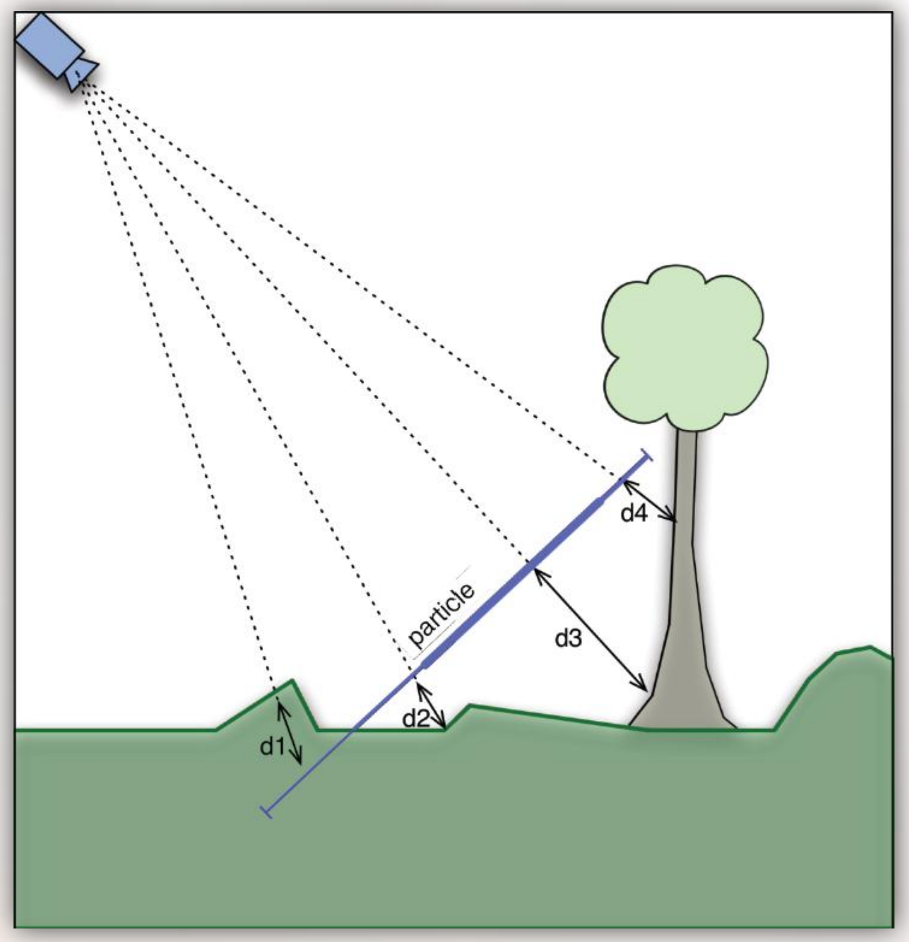
- Extension of billboard-based rendering, tackles the edging artefacts problem



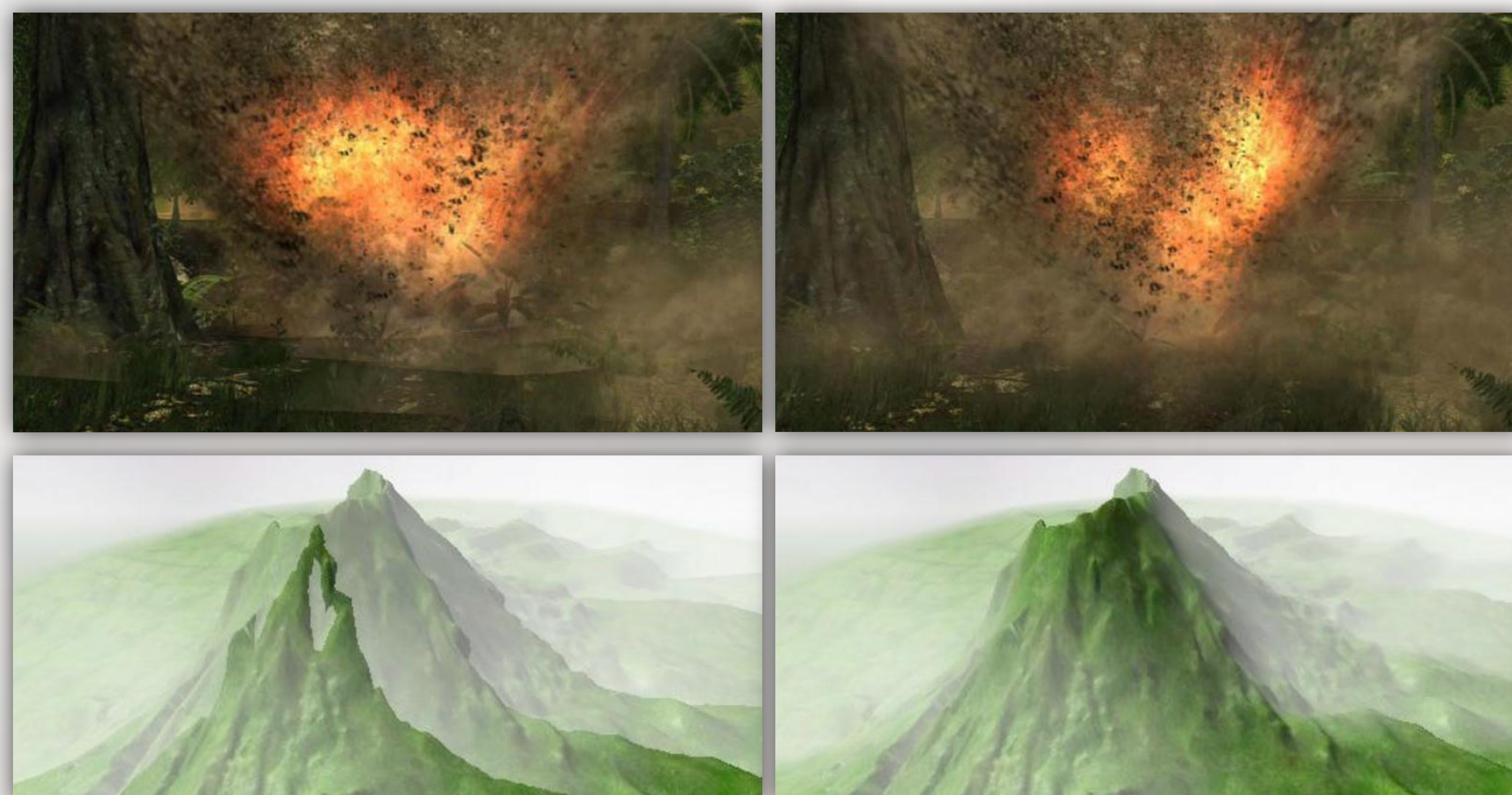
- Extension of billboard-based rendering, tackles the edging artefacts problem
- Solution – modulation of the billboard colour by depth-based factor, e.g.:

$$D = \mathrm{saturate}((Z_{scene} - Z_{particle}) * \mathrm{scale})$$





- Extension of billboard-based rendering, tackles the edging artefacts problem



- Under some specific conditions, scattering might be analytically approximated

- Under some specific conditions, scattering might be analytically approximated
- For instance, let's assume (Sun et al.):
  - Homogeneous medium, spanning the entire visible scene
  - Only single scattering
  - Isotropic point light sources

- Under some specific conditions, scattering might be analytically approximated
- For instance, let's assume (Sun et al.):
  - Homogeneous medium, spanning the entire visible scene
  - Only single scattering
  - Isotropic point light sources

$$L_a = \frac{\beta I_0}{4\pi} \int_0^{D_{vp}} \frac{e^{-\beta\sqrt{D_{sv}^2 + x^2 - 2xD_{sv}\cos\gamma}}}{D_{sv}^2 + x^2 - 2xD_{sv}\cos\gamma} \cdot e^{-\beta x}\, dx$$

——>substitute $T_* = \beta D_*$ and $t = \beta x$

$$= \frac{\beta^2 I_0}{4\pi} \int_0^{T_{vp}} \frac{e^{-\sqrt{T_{sv}^2 + t^2 - 2tT_{sv}\cos\gamma}}}{T_{sv}^2 + t^2 - 2tT_{sv}\cos\gamma} \cdot e^{-t}\, dt$$

——>substitute $z = t - T_{sv}\cos\gamma$

$$= \frac{\beta^2 I_0 e^{-T_{sv}\cos\gamma}}{4\pi} \int_{-T_{sv}\cos\gamma}^{T_{vp} - T_{sv}\cos\gamma} \frac{e^{-\sqrt{z^2 + T_{sv}^2 \sin^2\gamma}}}{z^2 + T_{sv}^2 \sin^2\gamma} \cdot e^{-z}\, dz$$

——>substitute $z = T_{sv}\sin\gamma\tan\eta$

$$= \frac{\beta^2 I_0 e^{-T_{sv}\cos\gamma}}{4\pi T_{sv}\sin\gamma} \int_{\gamma - \frac{\pi}{2}}^{\arctan\frac{T_{vp} - T_{sv}\cos\gamma}{T_{sv}\sin\gamma}} e^{-T_{sv}\sin\gamma\frac{1+\sin\eta}{\cos\eta}}\, d\eta$$

——>substitute $\eta = 2\xi - \frac{\pi}{2}$

$$= \frac{\beta^2 I_0 e^{-T_{sv}\cos\gamma}}{2\pi T_{sv}\sin\gamma} \int_{\gamma/2}^{\frac{\pi}{4} + \frac{1}{2}\arctan\frac{T_{vp} - T_{sv}\cos\gamma}{T_{sv}\sin\gamma}} \exp[-T_{sv}\sin\gamma\tan\xi]\, d\xi$$

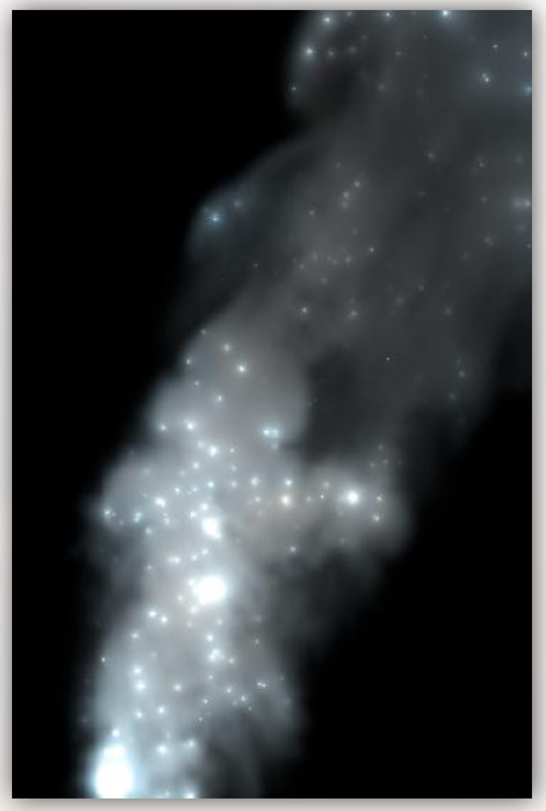
```
frag2app fmain(
    float4 objPos : TEXCOORD3,            // 2D texture coords
    ...
    uniform samplerRECT F,                // 2D special functions
    uniform samplerRECT G0,
    uniform samplerRECT Gn)
{
    frag2app OUT;                         // output radiance
    // Set up and calculate Tsv, γ, Dsv, Tvp, θs and θ's

    /********** Compute La from equation 11 ******/
    A0 = (β * I0 * exp[-Tsv * cos γ])/(2π * Dsv * sin γ);// equation 7
    A1 = Tsv * sin γ;                     // equation 8
    v = π/4 + (1/2) arctan [(Tvp - Tsv * cos γ)/(Tsv * sin γ)];
                                          // v is one of texture coords
    f1 = texRECT(F, float2(A1, v));       // 2D texture lookup
    f2 = texRECT(F, float2(A1, γ/2));
    airlight = A0 * (f1 - f2);            // equation 11

    /********** Diffuse surface radiance from equation 17 ******/
    d1 = kd * exp[-Tsp] * cos θs * I0/(Dsp * Dsp);
    d2 = (kd * I0 * β * β)/(2π * Tsp) * texRECT(G0, float2(Tsp, θs));
    diffuse = d1 + d2;

    /********** Specular surface radiance from equation 18 ******/
    s1 = ks * exp[-Tsp] * cos^n θ's * I0/(Dsp * Dsp);
    s2 = (ks * I0 * β * β)/(2π * Tsp) * texRECT(Gn, float2(Tsp, θ's));
    specular = s1 + s2;

    /********** Final Color (equation 19) ******/
    OUT.color = airlight + (diffuse + specular) * exp[-Tvp];
    return OUT;
}
```

- Under some specific conditions, scattering might be analytically approximated
- For instance, let's assume (Sun et al.):
  - Homogeneous medium, spanning the entire visible scene
  - Only single scattering
  - Isotropic point light sources

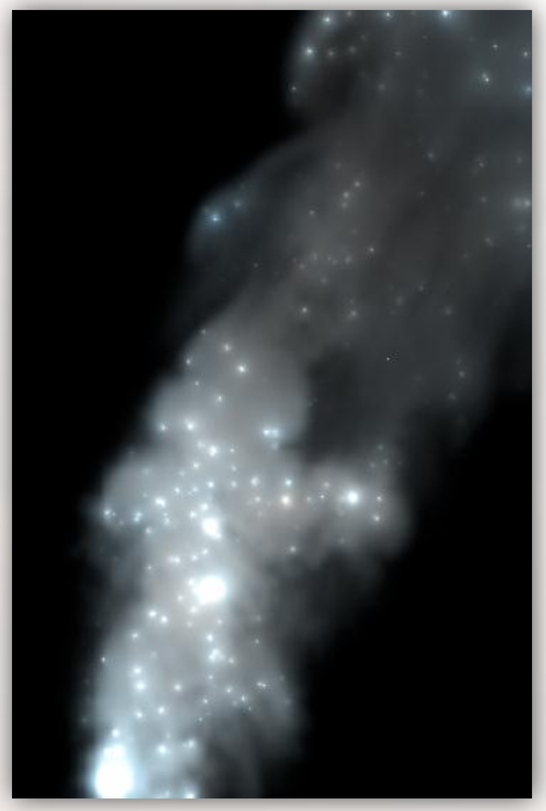
- Under some specific conditions, scattering might be analytically approximated
- For instance, let's assume (Sun et al.):
  - Homogeneous medium, spanning the entire visible scene
  - Only single scattering
  - Isotropic point light sources

- Under some specific conditions, scattering might be analytically approximated
- For instance, let's assume (Sun et al.):
  - Homogeneous medium, spanning the entire visible scene
  - Only single scattering
  - Isotropic point light sources

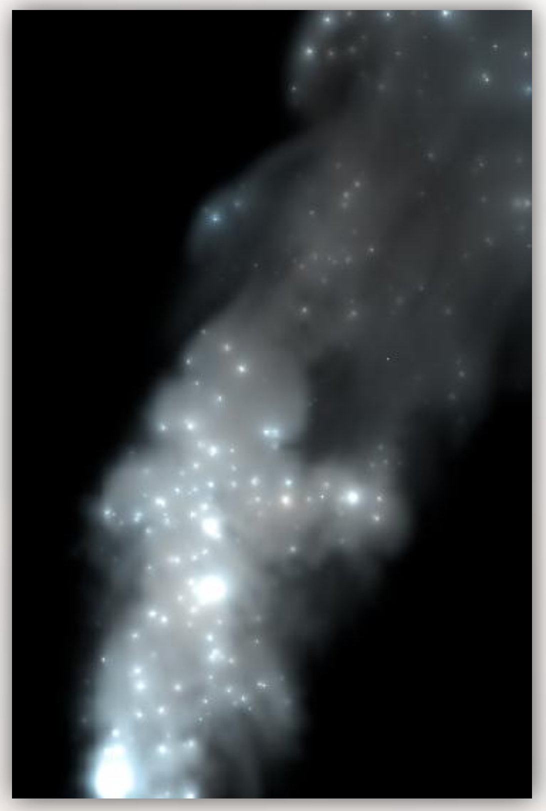
- Extension of IR to participating media



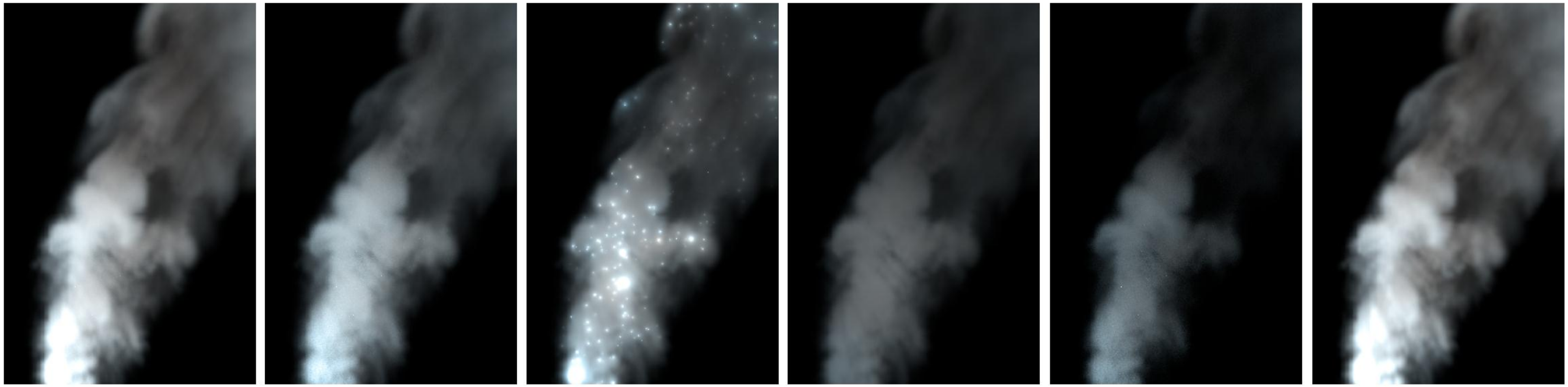
- Extension of IR to participating media
- As in areal IR, singularities appear
- Solution – bias compensation
  - Exact – slow



- Extension of IR to participating media
- As in areal IR, singularities appear
- Solution – bias compensation
  - Exact – slow
  - Approximations:
    - using other VPLs
    - sub-sampling random walks
    - local visibility reuse
    - local vertices generation
    - limited recursion depth



- Extension of IR to participating media
- As in areal IR, singularities appear
- Solution – bias compensation



a) unbiased, 8h50min b) multiple scatt. only c) VPL rendering d) VPLs with clamping e) clamping energy loss f) our method, 5 fps

- Adaptation of Discrete Ordinates method (VRT variant)

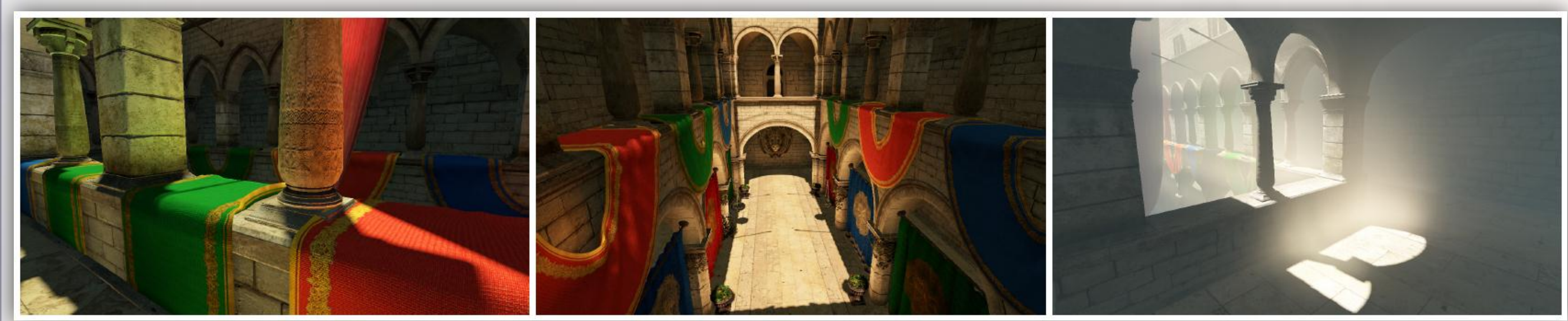
- Adaptation of Discrete Ordinates method (VRT variant)
- Lattice-based – uses light propagation volume (LPV)
- Only used for low-frequency (indirect) lighting

- Adaptation of Discrete Ordinates method (VRT variant)
- Lattice-based – uses light propagation volume (LPV)
- Only used for low-frequency (indirect) lighting
- Basic steps (per frame!):
    1. LPV initialization with area lights & surfaces causing indirect lighting
    2. Creation of volumetric representation of blocker geometry
    3. Light propagation simulation inside LPV
    4. Using LPV for lighting scene geometry

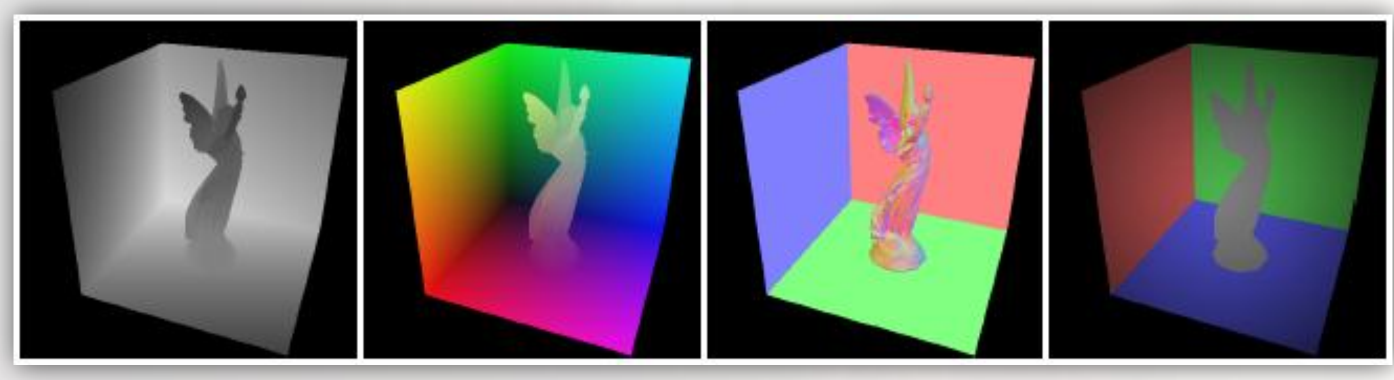
- Adaptation of Discrete Ordinates method (VRT variant)
- Lattice-based – uses light propagation volume (LPV)
- Only used for low-frequency (indirect) lighting
- Basic steps (per frame!):
    1. LPV initialization with area lights & surfaces causing indirect lighting
    2. Creation of volumetric representation of blocker geometry
    3. Light propagation simulation inside LPV
    4. Using LPV for lighting scene geometry

- Adaptation of Discrete Ordinates method (VRT variant)
- Lattice-based – uses light propagation volume (LPV)
- Only used for low-frequency (indirect) lighting
- Basic steps (per frame!):
    1. LPV initialization with area lights & surfaces causing indirect lighting
    2. Creation of volumetric representation of blocker geometry
    3. Light propagation simulation inside LPV
    4. Using LPV for lighting scene geometry

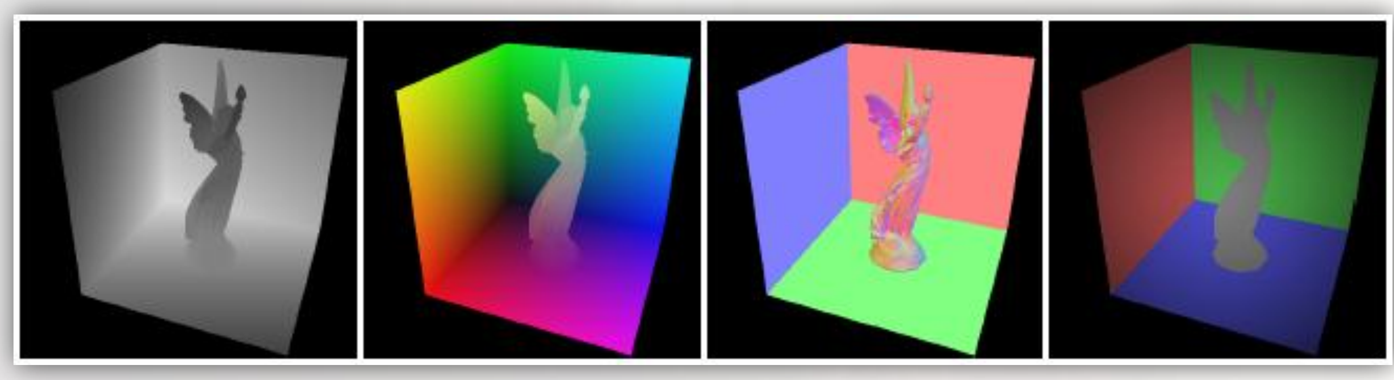
- Adaptation of Discrete Ordinates method (VRT variant)
- Lattice-based – uses light propagation volume (LPV)
- Only used for low-frequency (indirect) lighting
- Basic steps (per frame!):
  1. LPV initialization with area lights & surfaces causing indirect lighting
  2. Creation of volumetric representation of blocker geometry
  3. Light propagation simulation inside LPV
  4. Using LPV for lighting scene geometry



- Every (point) light yields one reflective shadow map (RSM)

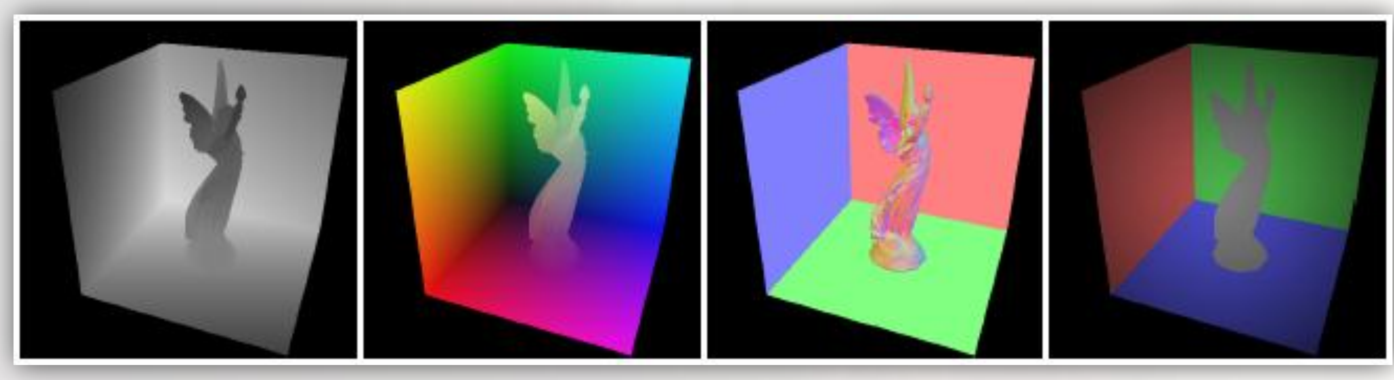


- Every (point) light yields one reflective shadow map (RSM)

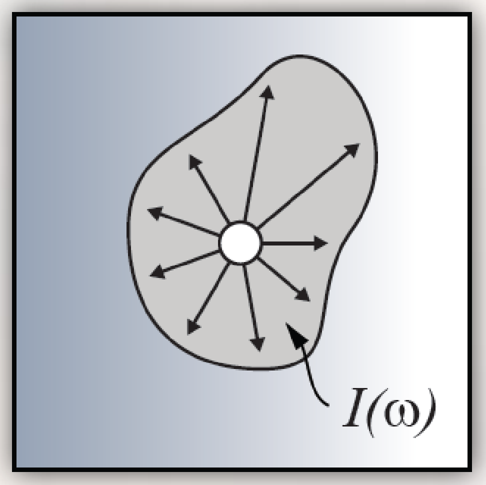


- Every texel of a RSM is treated as VPL
- Low-frequency lights (area lights, env. map, fuzzy lights) treated as VPLs

- Every (point) light yields one reflective shadow map (RSM)

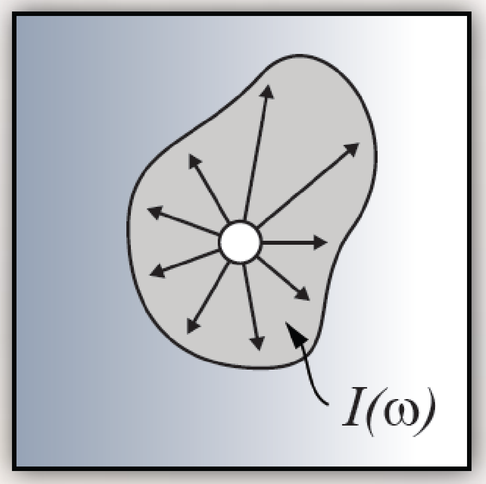


- Every texel of a RSM is treated as VPL
- Low-frequency lights (area lights, env. map, fuzzy lights) treated as VPLs
- VPLs are injected into LPV using spherical harmonic (SH) projection
- **Result – initial energy state of the scene in a single LPV**

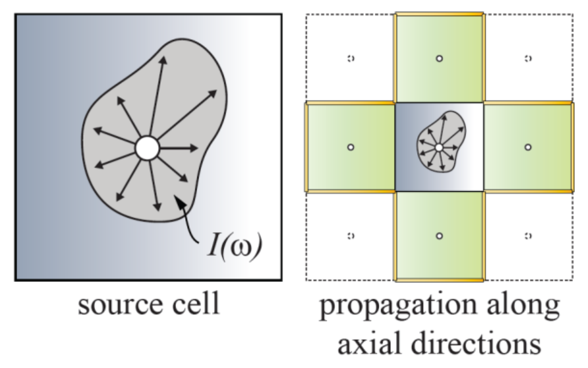


- Surfaces are sampled from camera position and multiple RSMs (not the lighting ones!)
- Temporal coherence

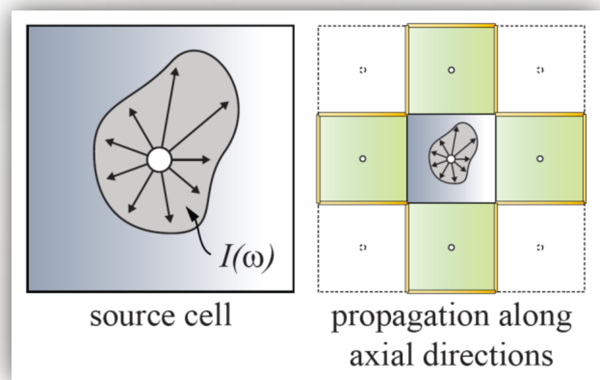
- Surfaces are sampled from camera position and multiple RSMs (not the lighting ones!)
- Temporal coherence
- Surfels are inserted into geometry volumes (GV), again using SHs
- **Result – multiple GVs, each corresponding to one surfels source**
- These are merged in to one GV (max)



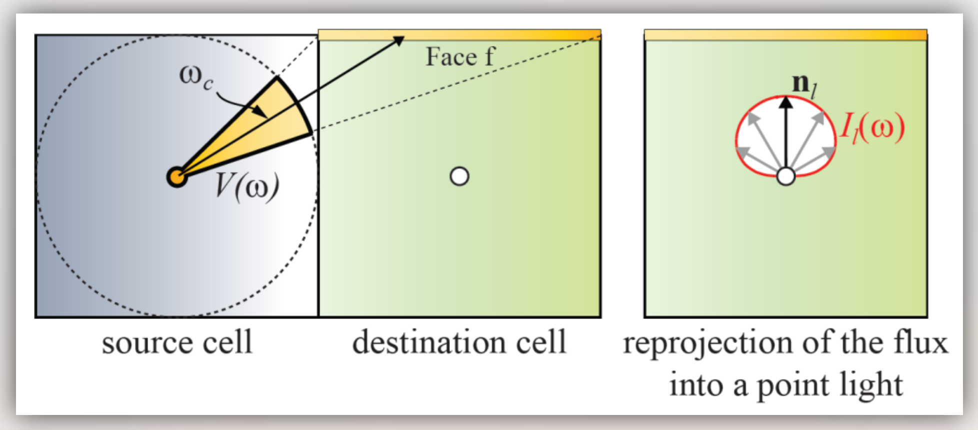
- Each source cell propagates light to its 6 adjacent cells (instead of 26)



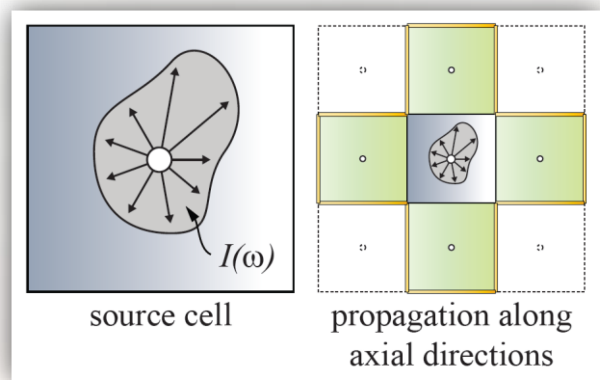
- Each source cell propagates light to its 6 adjacent cells (instead of 26)
- Each destination cell reprojects the received light into its centre

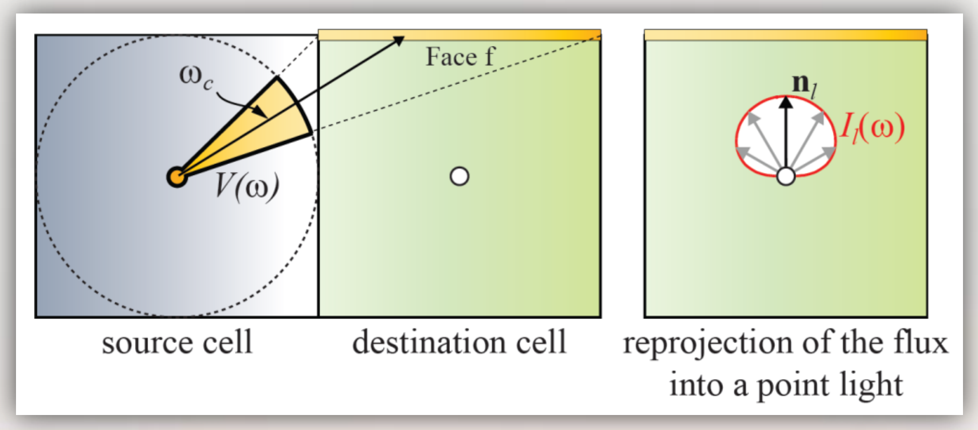


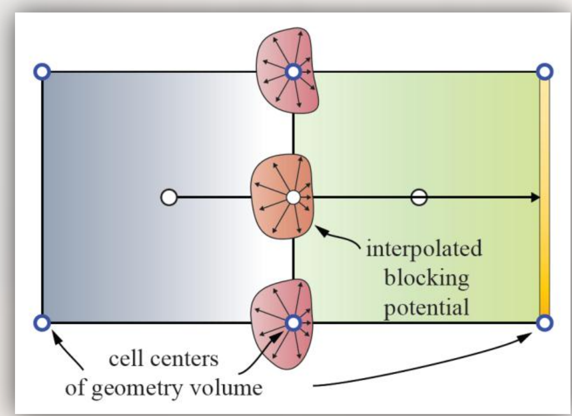
$$\Phi_f = \int_\Omega \Phi_l \langle \mathbf{n}_l, \omega \rangle_+ d\omega \qquad \Phi_l = \Phi_f / \pi$$



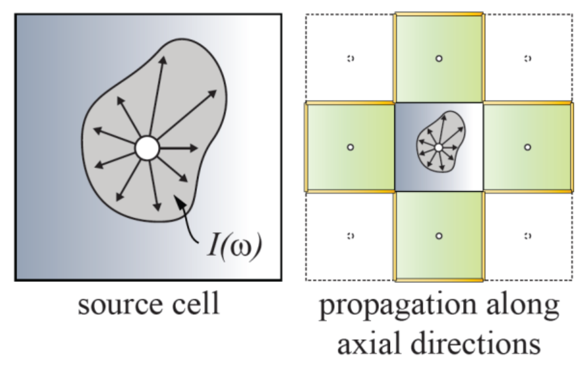
- Each source cell propagates light to its 6 adjacent cells (instead of 26)
- Each destination cell reprojects the received light into its centre
- Each propagation step accounts for blocking from GV

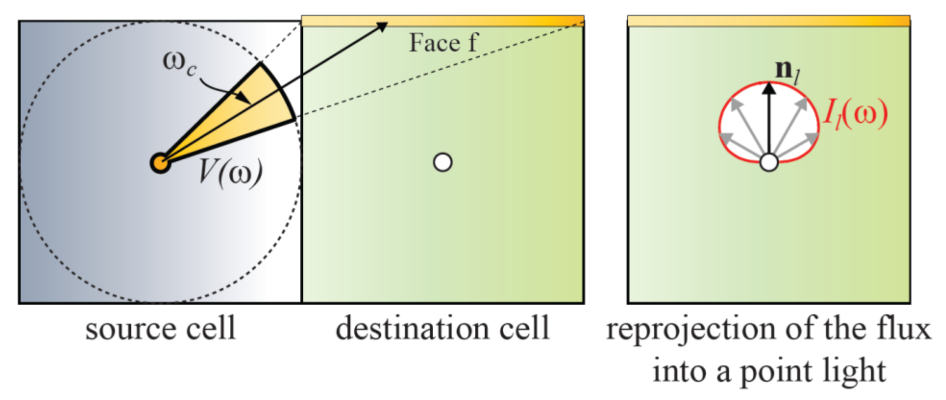


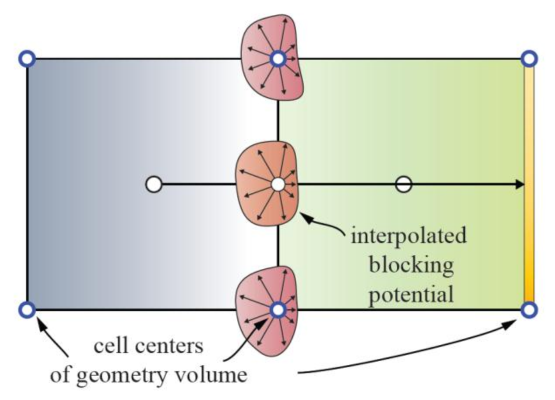




- Each source cell propagates light to its 6 adjacent cells (instead of 26)
- Each destination cell reprojects the received light into its centre
- Each propagation step accounts for blocking from GV
- Iteration count (∑)
- **Result – scene energy state**

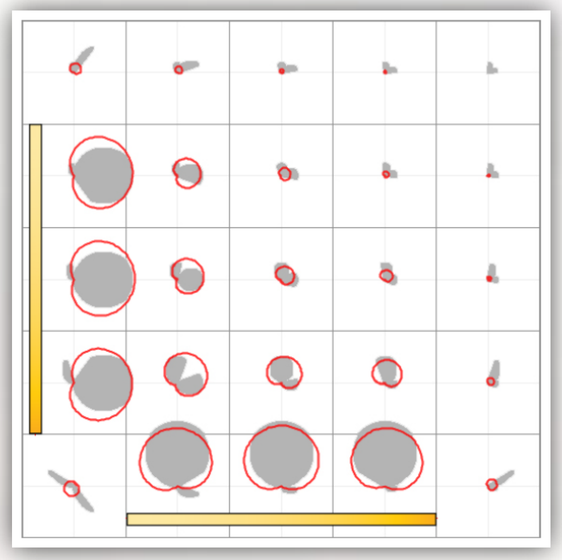


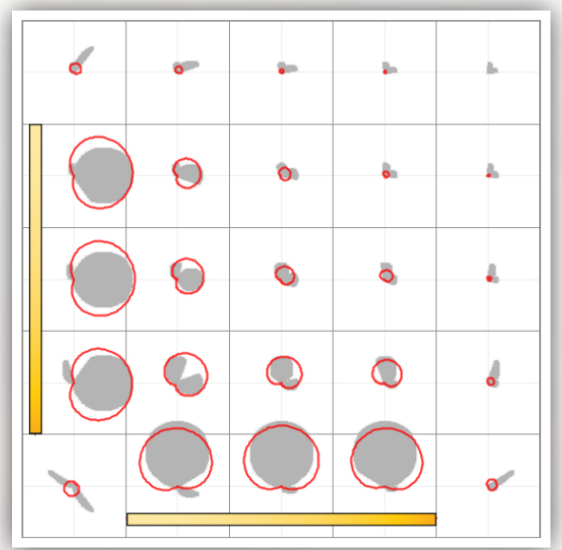




Computer Graphics Charles University

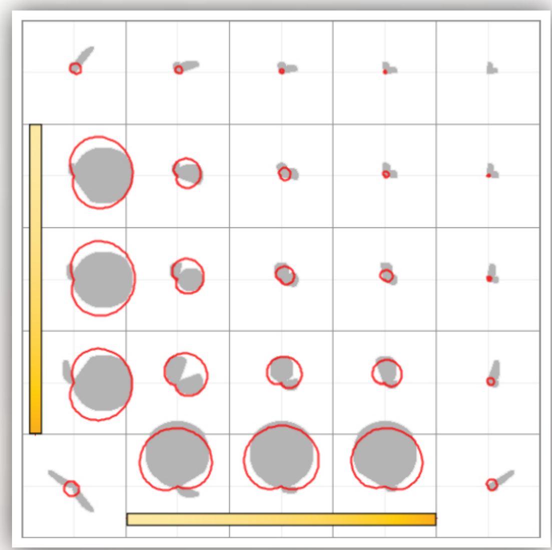
- Diffuse surfaces – simply fetch the LPV

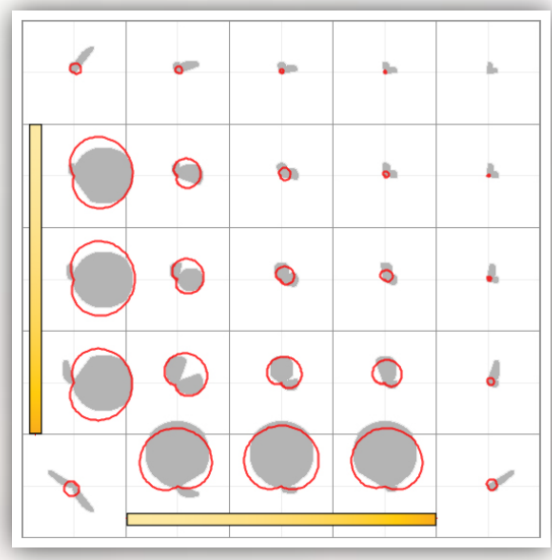




- Diffuse surfaces – simply fetch the LPV
- Glossy surfaces – perform ray marching along reflected vector

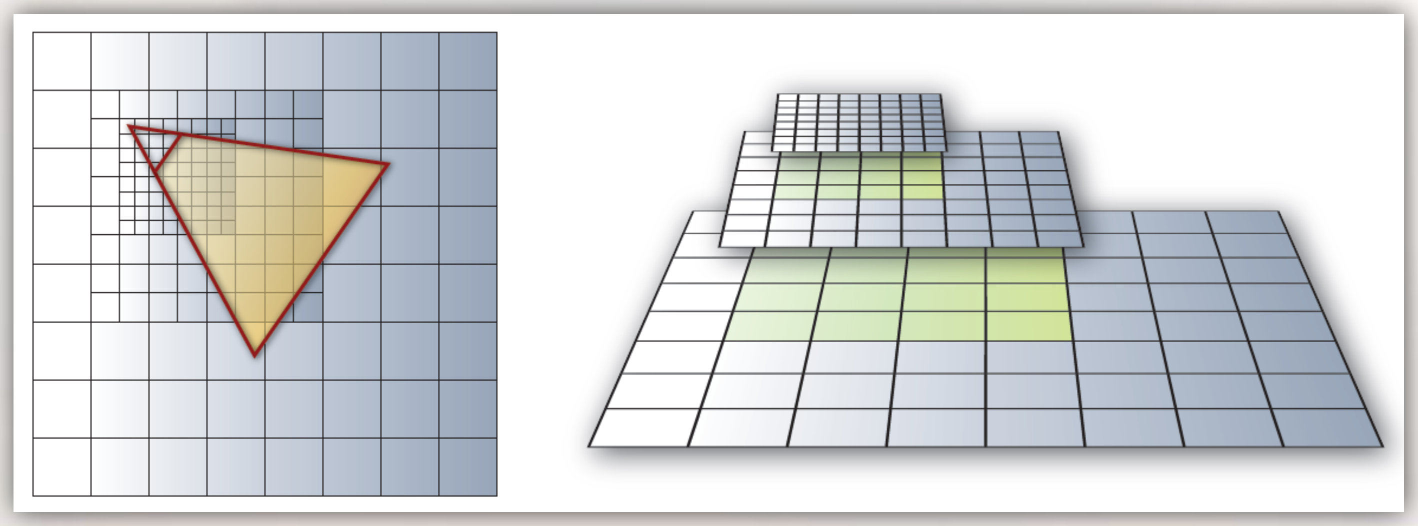
- Diffuse surfaces – simply fetch the LPV
- Glossy surfaces – perform ray marching along reflected vector
- Participating media – ray-march through the LPV along view ray



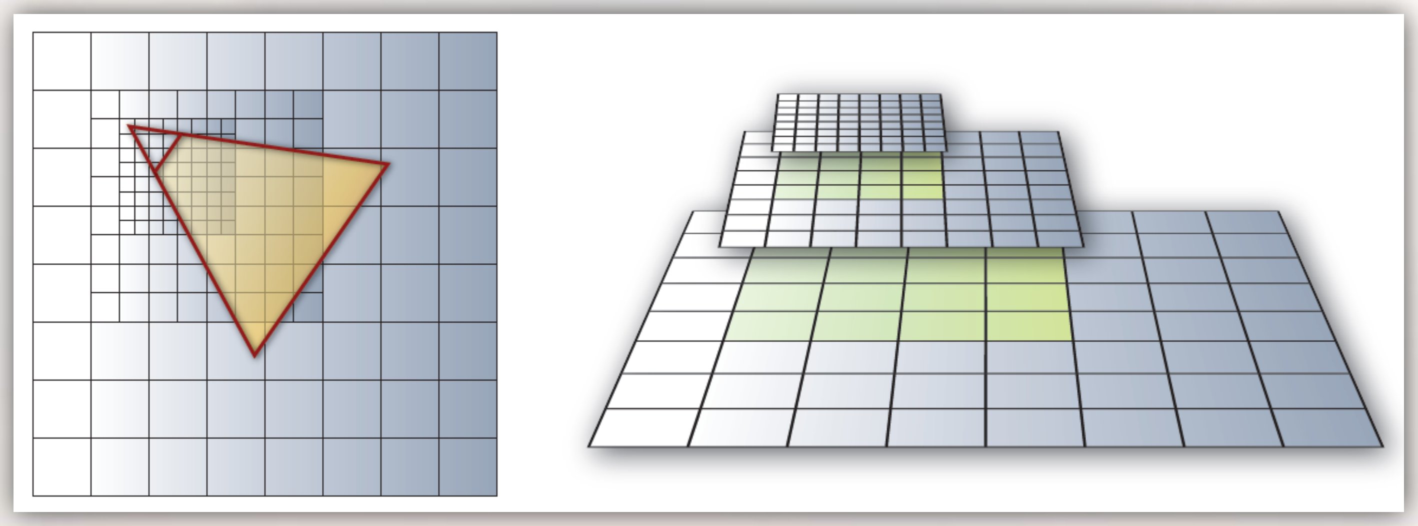


- Diffuse surfaces – simply fetch the LPV
- Glossy surfaces – perform ray marching along reflected vector
- Participating media – ray-march through the LPV along view ray
- Limitations:
  - Isotropic PF
  - Low-frequency light
  - Homogeneous medium (unless density volume is used)

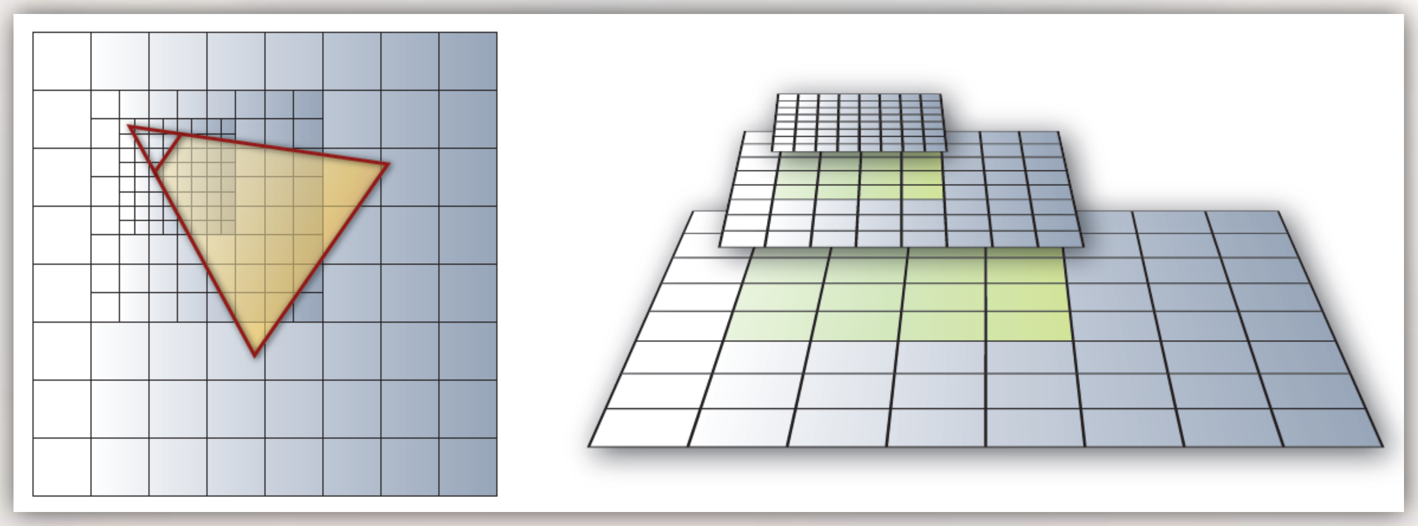
- Instead of one large LPV, use several nested smaller ones (3)
- Centred around observer, displaced along view direction



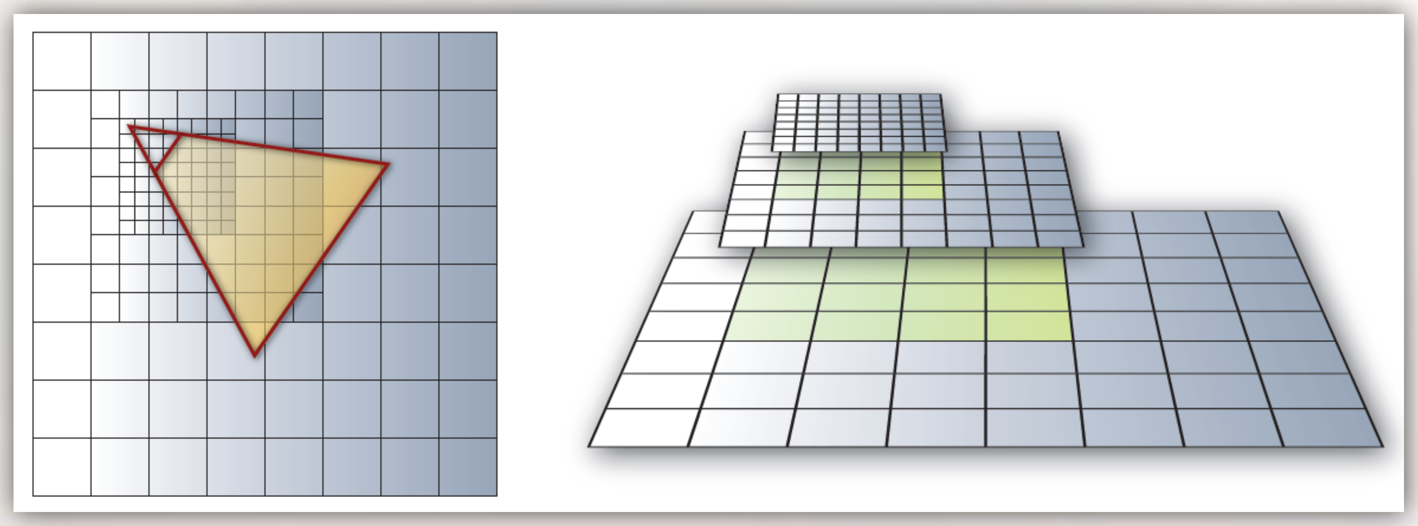
- Instead of one large LPV, use several nested smaller ones (3)
- Centred around observer, displaced along view direction
- **Injection** – inject VPLs and surfels into all LPV levels
- **Propagation** – simulate all levels separately
- **Fetching** – fetch the finest available level, interpolate at boundaries



- Instead of one large LPV, use several nested smaller ones (3)
- Centred around observer, displaced along view direction
- **Injection** – inject VPLs and surfels into all LPV levels
- **Propagation** – simulate all levels separately
- **Fetching** – fetch the finest available level, interpolate at boundaries



- Instead of one large LPV, use several nested smaller ones (3)
- Centred around observer, displaced along view direction
- **Injection** – inject VPLs and surfels into all LPV levels
- **Propagation** – simulate all levels separately
- **Fetching** – fetch the finest available level, interpolate at boundaries



- Statistics:
  - 2^{16} VPLs per primary light source
  - 3.75MB for cascaded LPV ($3x32^3$ cells) and 0.75MB per GV
  - 8 propagation iterations (!)
  - NV GTX 285: ~100 FPS (diffuse only), ~35 FPS (participating medium)

- Statistics:
  - 2^{16} VPLs per primary light source
  - 3.75MB for cascaded LPV ($3x32^3$ cells) and 0.75MB per GV
  - 8 propagation iterations (!)
  - NV GTX 285: ~100 FPS (diffuse only), ~35 FPS (participating medium)
- Evaluation:
  - Pros: very fast, physically-based, obtains energy state of the entire scene, temporal coherence, allows fully dynamic scenes, flexible
  - Cons: lots of ‘hacks’ and potential sources of visual artefacts

Outline

- Motivation
- Introduction
- Properties of participating media
- Rendering equation
- Storage strategies
- Non-interactive rendering strategies
- Part I revision
- Interactive rendering strategies
- **Atmospheric rendering**
- Cloud rendering
- (References)

- Specifics
  - **Very** sparse medium
  - Spatially large and symmetrical
  - Very little absorption (mostly urban areas)
  - Combined Rayleigh and Mie scattering
  - Well defined density (exponential w/r to altitude)
  - Density may vary w/r to latitude and longitude
  - Special phenomena (sundogs, parhelia)
  - Stable, slowly changing

$$\rho_{\{R,M\}}(P) = \exp(-\frac{h}{H_{\{R,M\}}})$$

- Specifics
  - **Very** sparse medium
  - Spatially large and symmetrical
  - Very little absorption (mostly urban areas)
  - Combined Rayleigh and Mie scattering
  - Well defined density (exponential w/r to altitude)
  - Density may vary w/r to latitude and longitude
  - Special phenomena (sundogs, parhelia)
  - Stable, slowly changing

$$\rho_{\{R,M\}}(P) = \exp(-\frac{h}{H_{\{R,M\}}})$$

- Classical methods
  - Path tracing
  - Volumetric radiance transfer
  - Photon mapping

- Most notable – Preetham's model
- Sky luminance $Y(T,\theta,\theta_s,\delta)$ given as

$$Y = Y_z \frac{\mathcal{F}(\theta,\gamma)}{\mathcal{F}(0,\theta_s)} \qquad \mathcal{F}(\theta,\gamma) = \left(1 + Ae^{\frac{B}{\cos\theta}}\right)\left(1 + Ce^{D\gamma} + E\cos^2\gamma\right)$$

- Most notable – Preetham's model
- Sky luminance $Y(T,\theta,\theta_s,\delta)$ given as

$$Y = Y_z \frac{\mathcal{F}(\theta,\gamma)}{\mathcal{F}(0,\theta_s)} \qquad \mathcal{F}(\theta,\gamma) = \left(1 + Ae^{\frac{B}{\cos\theta}}\right)\left(1 + Ce^{D\gamma} + E\cos^2\gamma\right)$$

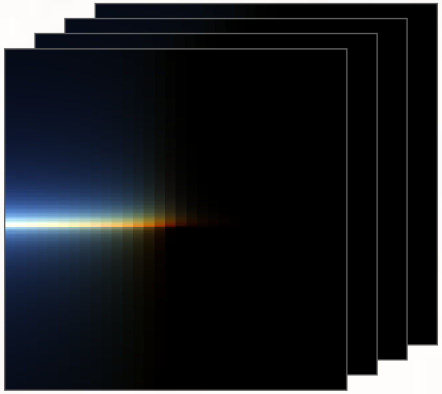
- T – turbidity (loosely "how strong overcast it is")

- Most notable – Preetham's model
- Sky luminance $Y(T,\theta,\theta_s,\delta)$ given as

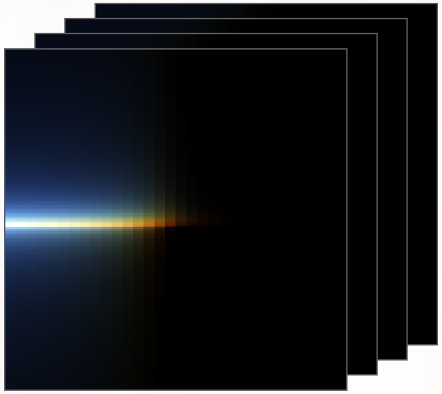
$$Y = Y_z \frac{\mathcal{F}(\theta,\gamma)}{\mathcal{F}(0,\theta_s)} \qquad \mathcal{F}(\theta,\gamma) = \left(1 + Ae^{\frac{B}{\cos\theta}}\right)\left(1 + Ce^{D\gamma} + E\cos^2\gamma\right)$$

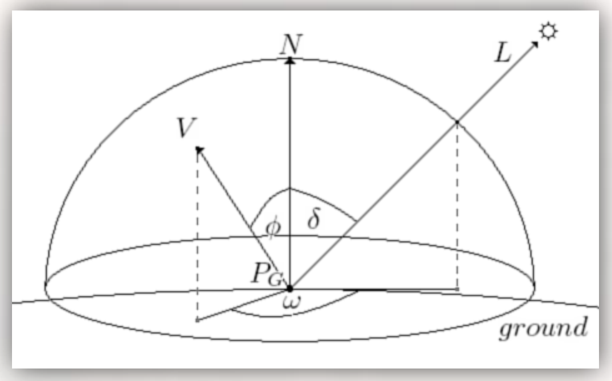
- T – turbidity (loosely "how strong overcast it is")
- Evaluation
  - Pros: simple (to use), fast
  - Cons: fixed to Earth's atmosphere, numerically unstable for $T<2$ and $T>10$, limited to zero altitude, limited to clear sky

- Basic idea
    1. Precompute scattering into table of colour values
    2. Fetch this table during rendering to obtain sky colour



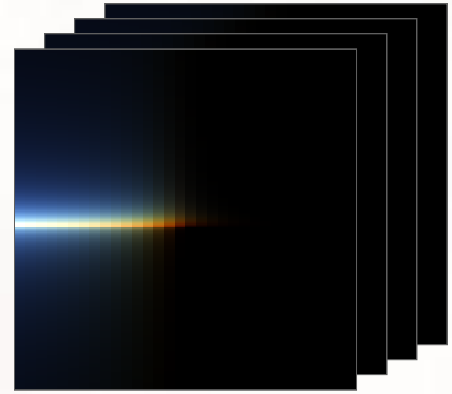
- Basic idea
  1. Precompute scattering into table of colour values
  2. Fetch this table during rendering to obtain sky colour
- Table dimensions
  - Sun zenith angle δ
  - View zenith angle ϕ
  - Sun azimuth ω
  - Observer altitude h

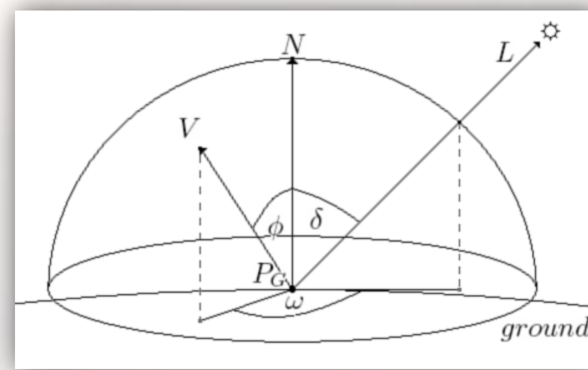


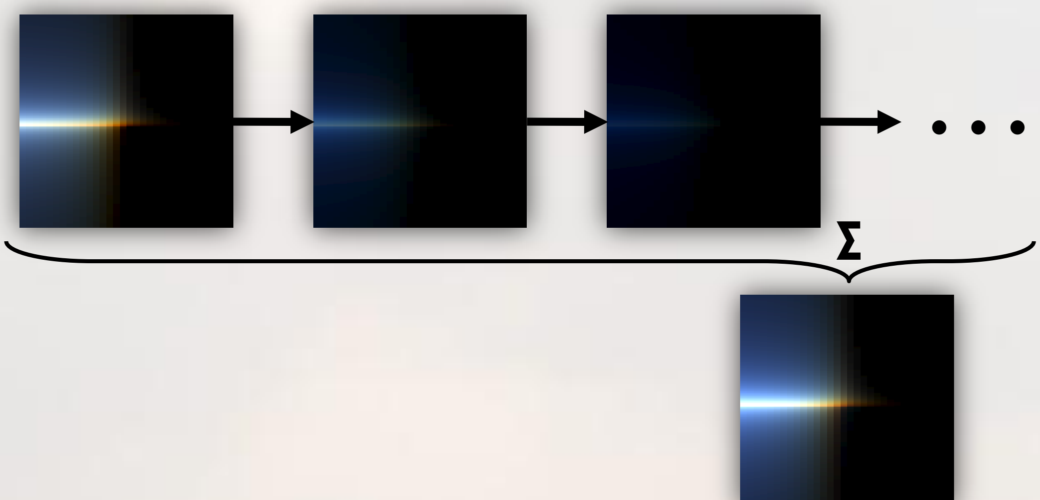


Atmospheric rendering – precomputed scattering

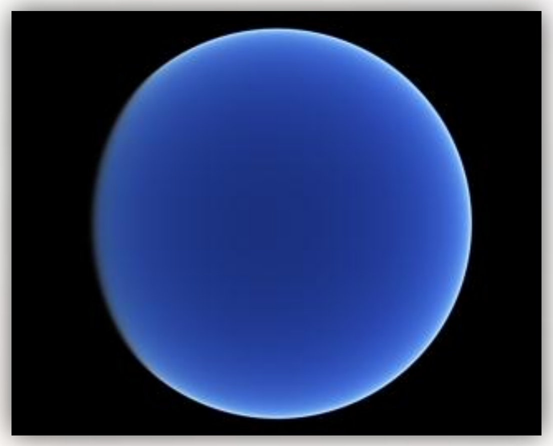
- Basic idea
  1. Precompute scattering into table of colour values
  2. Fetch this table during rendering to obtain sky colour
- Table dimensions
- Incremental multiple scattering computation



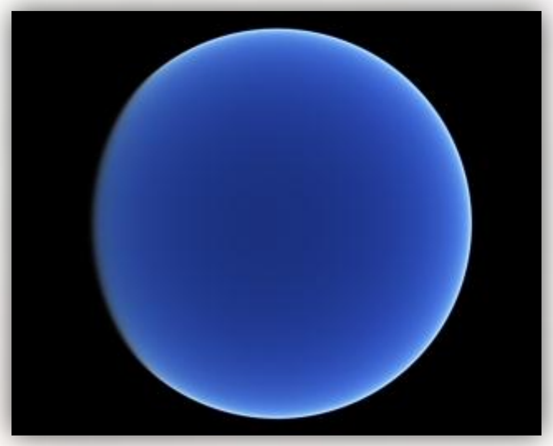


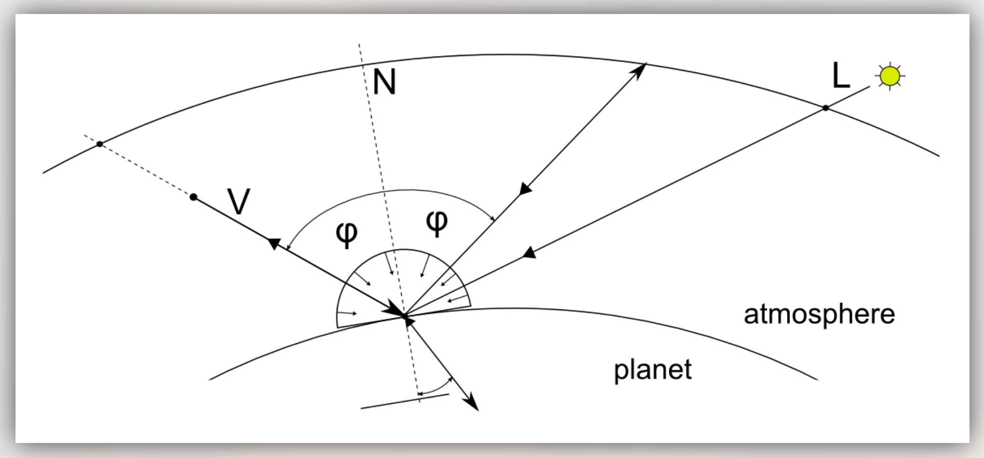


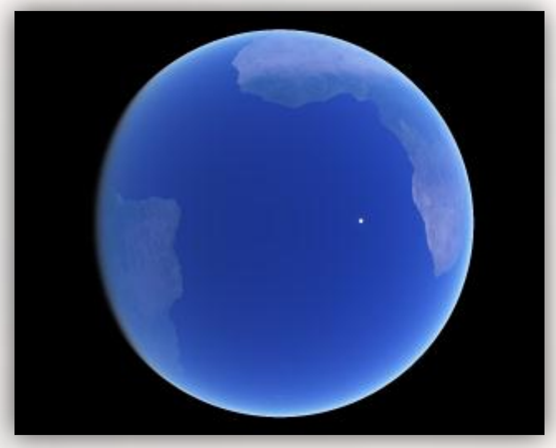
- Atmosphere
  - Plain sphere
  - 4D texture lookup (emulated)



- Atmosphere
  - Plain sphere
  - 4D texture lookup (emulated)
- Planetary surface
  - Atmospheric scattering
  - Ambient light or surface reflection
  - Water scattering (if present)





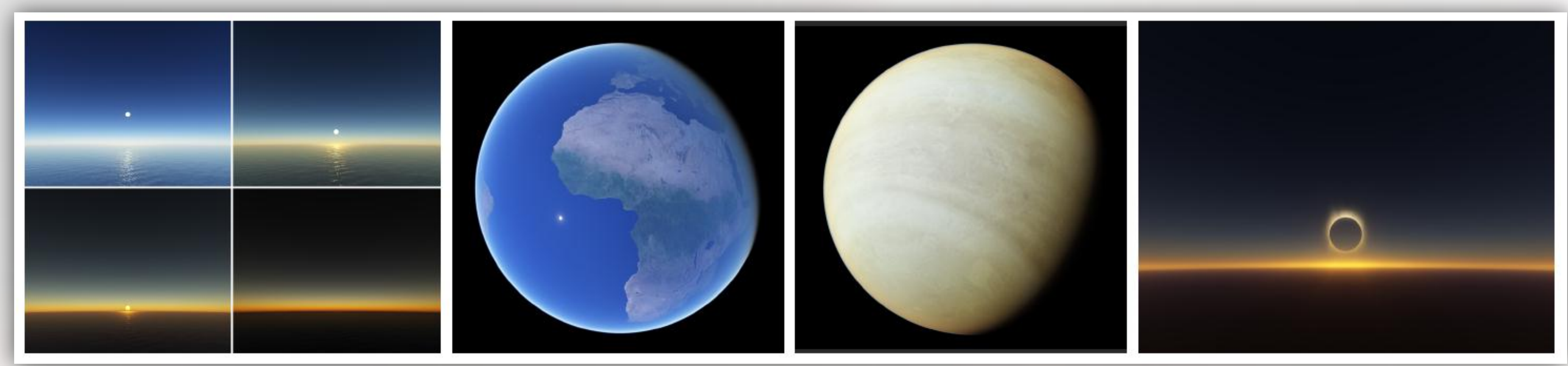


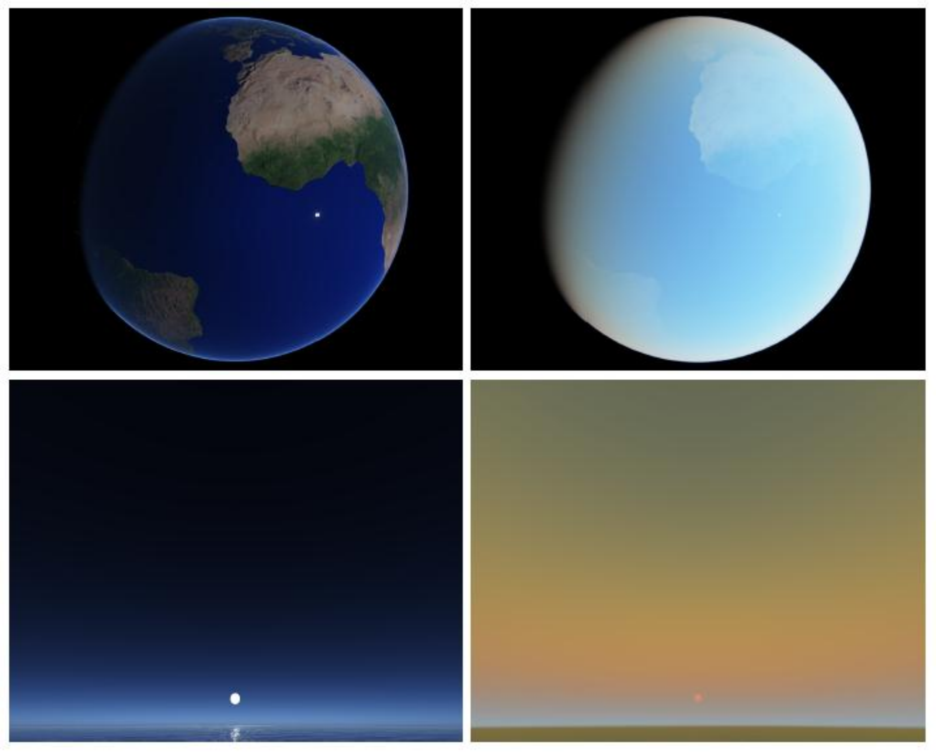
- Statistics
  - Precomputation - ~1 hour (CPU) / ~10s seconds (GPU)
  - Dataset ~10MB
  - NV 8800GT: ~100 FPS

- Statistics
  - Precomputation - ~1 hour (CPU) / ~10s seconds (GPU)
  - Dataset ~10MB
  - NV 8800GT: ~100 FPS

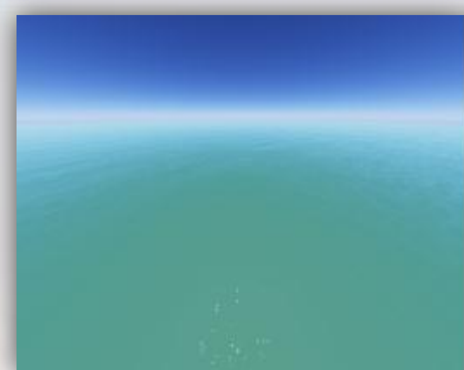
- Evaluation
  - Pros: very fast, directly usable in real-time engines, good looking results, supports multiple scattering, applicable to other media (water)
  - Cons: fixed atmospheric parameters, doesn't account for lat/long density variations

Precomputed scattering - results

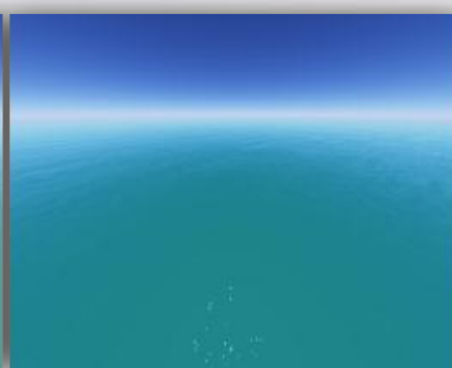




Precomputed scattering - results



1 meter



4 meters

10 meters

100 meters

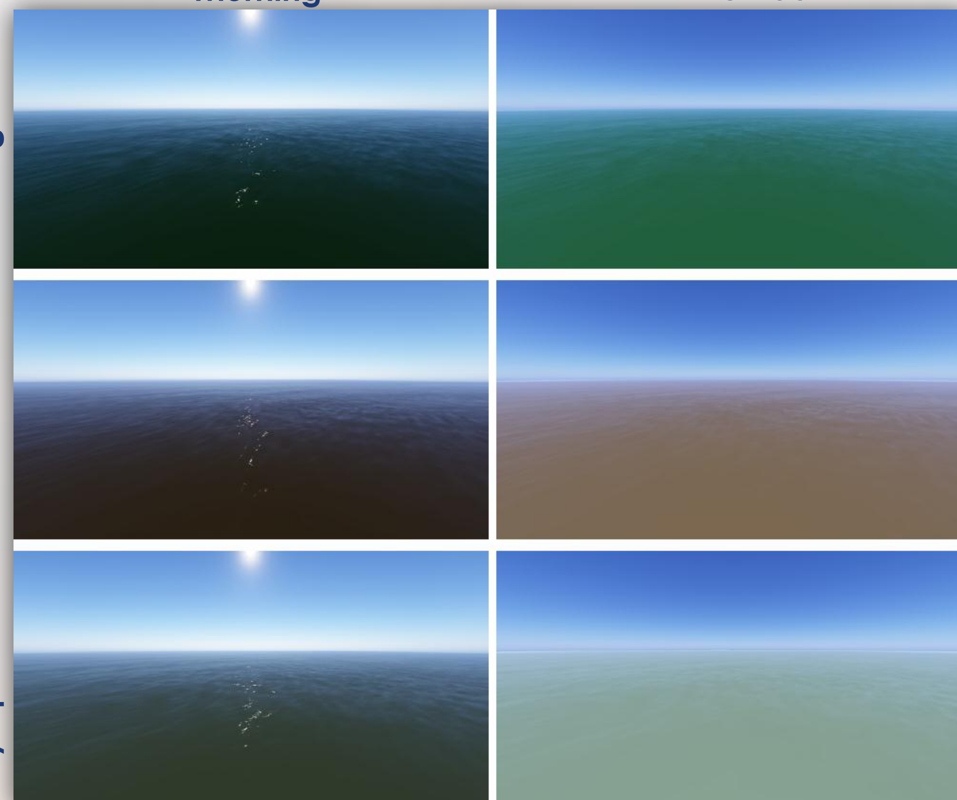
Pure seawater

Morning | Afternoon

Algae

Mud

Phytoplankton



Outline

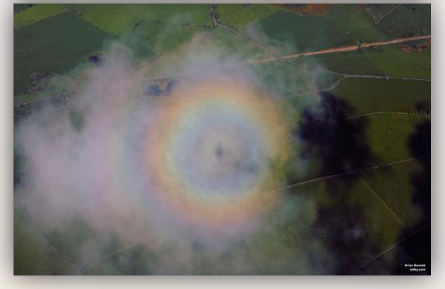
- Motivation
- Introduction
- Properties of participating media
- Rendering equation
- Storage strategies
- Non-interactive rendering strategies
- Part I revision
- Interactive rendering strategies
- Atmospheric rendering
- **Cloud rendering**
- (References)

- Specifics
  - Mediocre density
  - Large and asymmetrical shape
  - **No absorption – 100% albedo**
  - **High scattering anisotropy**
  - Mie scattering only
  - Potentially strong density fluctuation
  - Special phenomena (glory)
  - Mediocre evolution speed

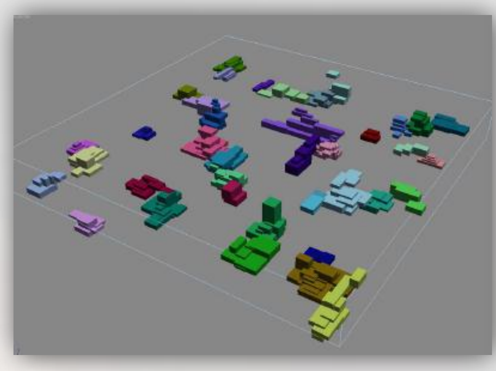
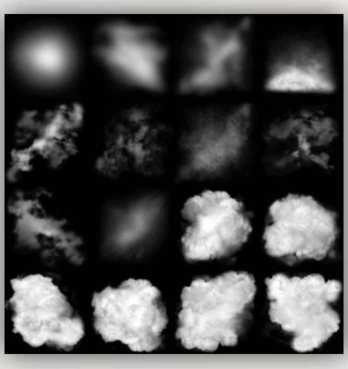
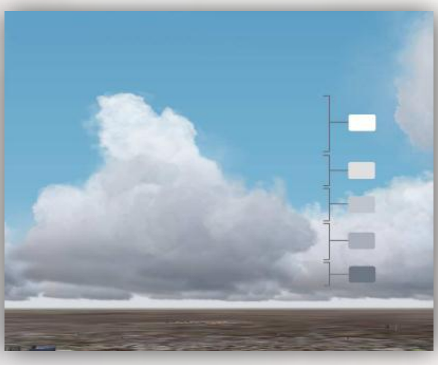



- Specifics
  - Mediocre density
  - Large and asymmetrical shape
  - **No absorption – 100% albedo**
  - **High scattering anisotropy**
  - Mie scattering only
  - Potentially strong density fluctuation
  - Special phenomena (glory)
  - Mediocre evolution speed
- Classical methods
  - Path tracing
  - Volumetric radiance transfer
  - Photon mapping

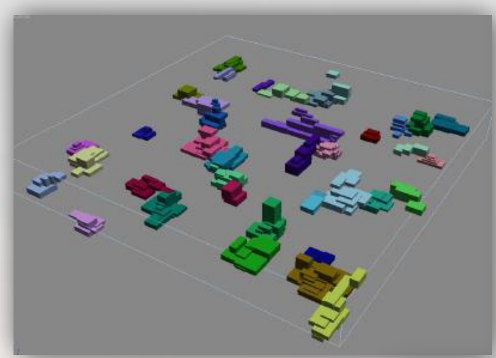


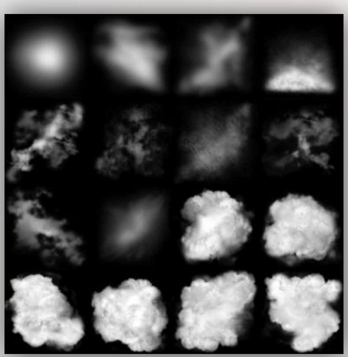



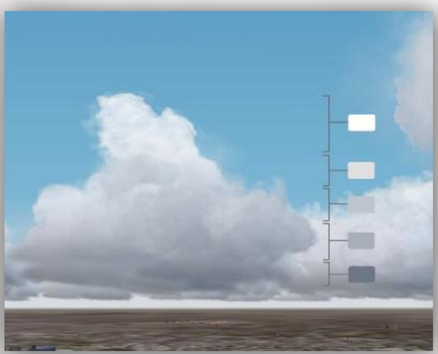
- Wang

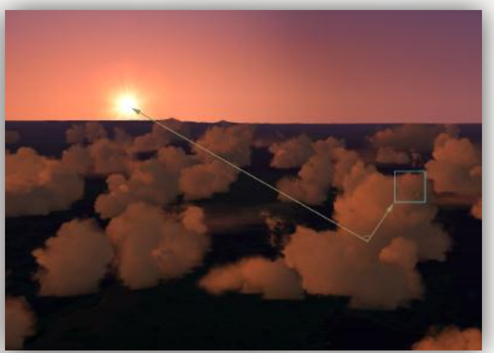





- Wang



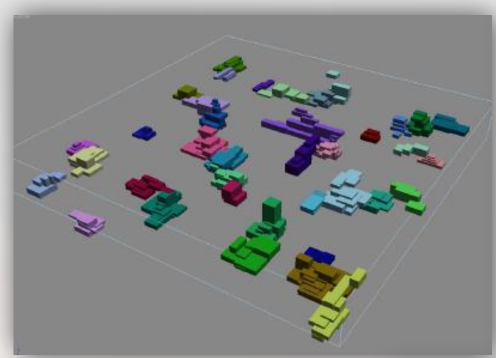
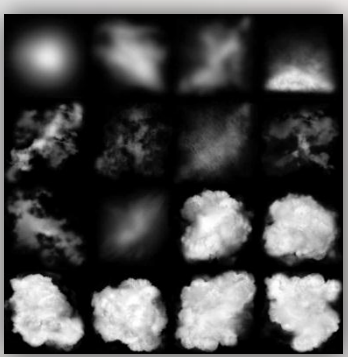
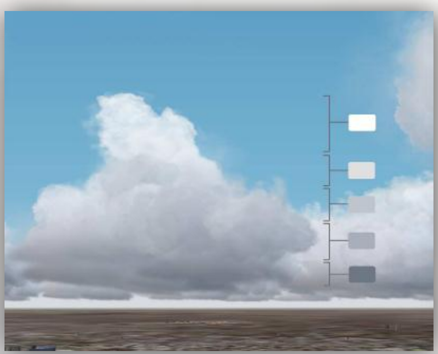
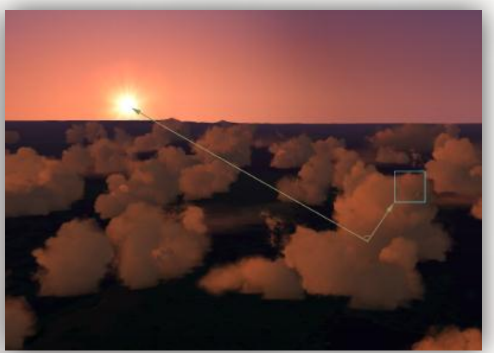






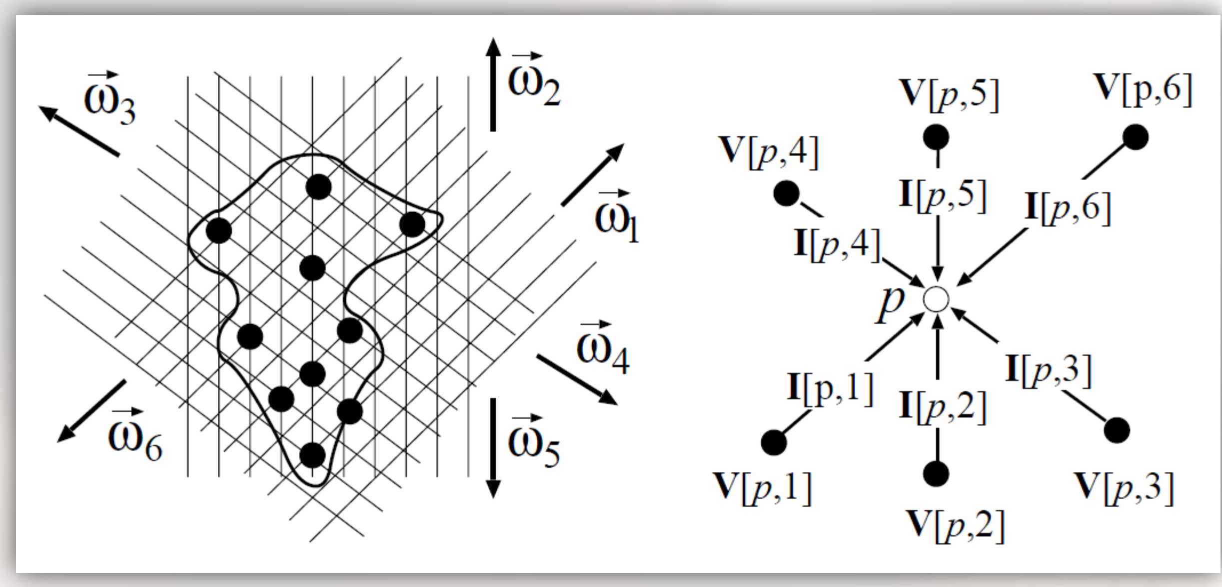
Cloud rendering – billboard-based methods

- Wang

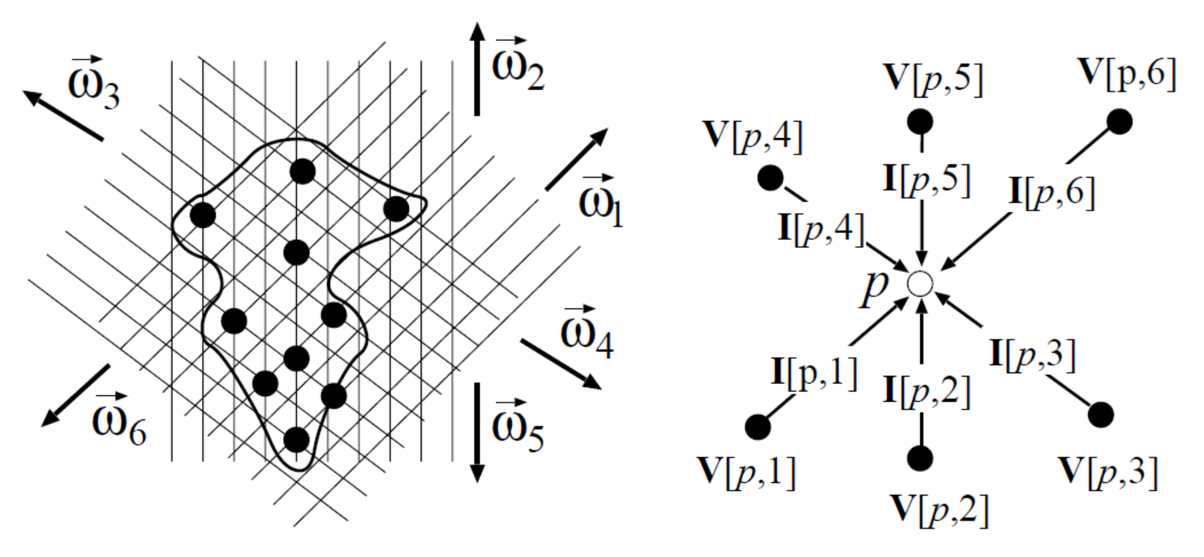





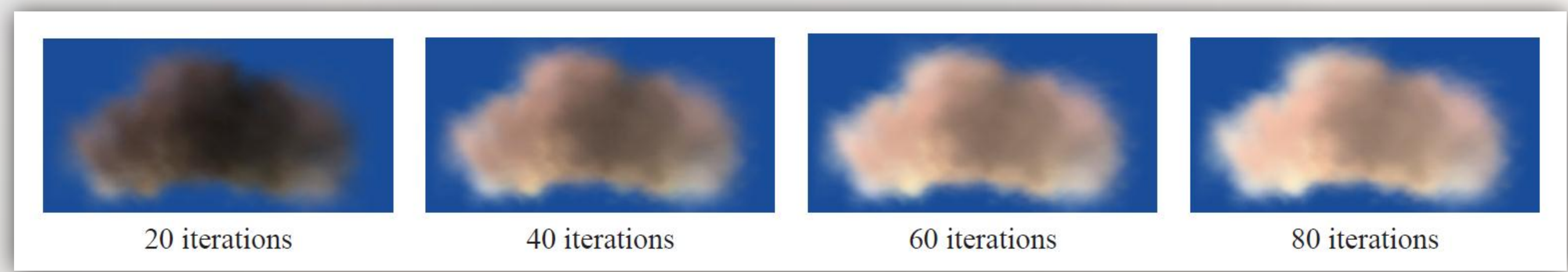
- Evaluation
    - Pros: fast, maps well to gaming studios pipeline
    - Cons: purely empirical, lengthy modelling phase

- Szirmay-Kalos et al.
- Idea: discretize and reuse light paths for every particle

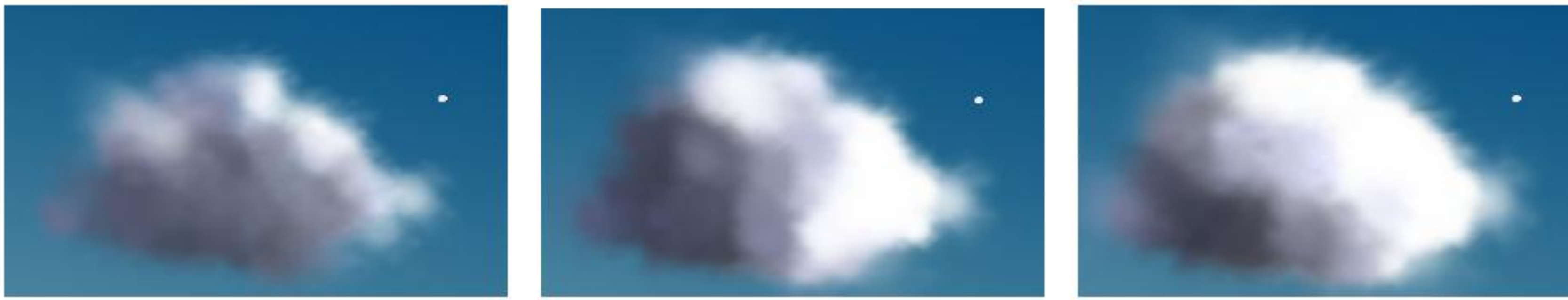


- Szirmay-Kalos et al.
- Idea: discretize and reuse light paths for every particle

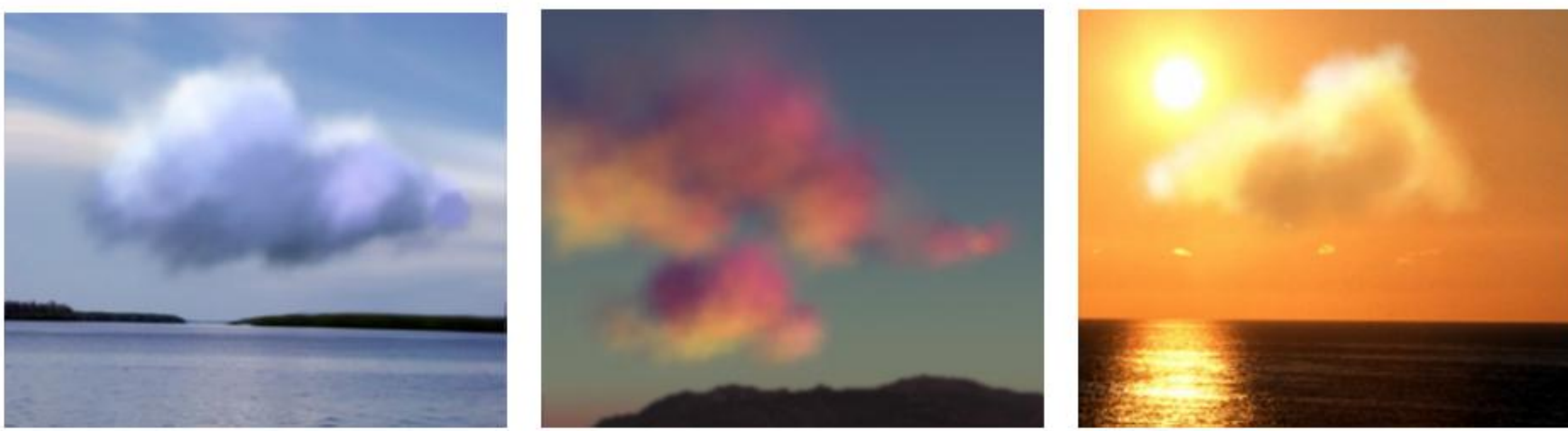




- Evaluation
  - Pros: maps well to GPU
  - Cons: doesn't allow 'frameless' behaviour, fuzzy results

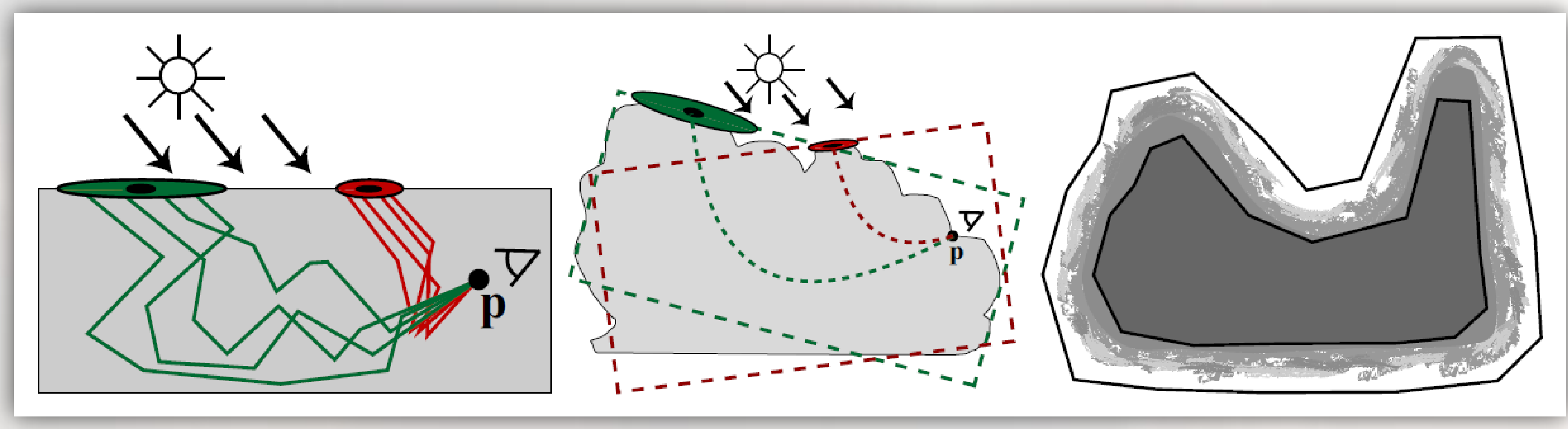


$D = 128, N = 512$ (45 FPS) | $D = 128, N = 2048$ (15 FPS) | $D = 512, N = 4096$ (reference)



- And now for something completely different…

- And now for something completely different…
- Method based on partial scattering precomputation



- And now for something completely different…
- Method based on partial scattering precomputation
- Evaluation:
  - Pros: physically-based, accounts for multiple anisotropic scattering
  - Cons: very complicated, limiting assumptions, slow, parts of the method a bit shady

Conclusion

Outline

- Motivation
- Introduction
- Properties of participating media
- Rendering equation
- Storage strategies
- Non-interactive rendering strategies
- Part I revision
- Interactive rendering strategies
- Atmospheric rendering
- Cloud rendering
- **(References)**

References

- Airlieau B. et al.: **Photon streaming for interactive global illumination in dynamic**
- **scenes**, 2010
- **Beer-Lambert-Bouguer law**: http://en.wikipedia.org/wiki/Beer%E2%80%93Lambert_law
- Born M. and Wolf E.: **Principles of Optics** (7th edition, corrected reprint) , 2003
- Bouthors A. et al.: **Interactive Multiple Anisotropic Scattering in Clouds**, 2008
- Bruneton E. and Neyret F.: **Precomputed Atmospheric Scattering**, 2008
- Chandrasekhar S.: **Radiative Transfer**, 1960
- Elek O. and Kmoch P.: **Real-Time Spectral Scattering in Large-Scale Natural Participating Media**, 2010; http://www.oskee.wz.cz/stranka/uploads/SCCG10ElekKmoch.pdf
- Engelhardt T. et al.: **Instant Multiple Scattering for Interactive Rendering of**
- **Heterogeneous Participating Media**, 2010
- Haber J. et al.: **Physically based Simulation of Twilight Phenomena**, 2005
- van de Hulst H. C.: **Light Scattering by Small Particles** (corrected reprint), 1981
- Jarosz W. et al.: **Radiance Caching for Participating Media**, 2008
- Jarosz W. et al.: **The Beam Radiance Estimate for Volumetric Photon Mapping**, 2008
- Jensen H. W. and Christensen P. W.: **Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps**, 1998
- Jensen H. W.: **Realistic Image Synthesis Using Photon Mapping**, 2001
- Kaplanyan A. and Dachsbacher C.: **Cascaded Light Propagation Volumes for Real-Time Indirect Simulation**, 2010

References

- Keller A.: **Instant Radiosity**, 1997
- Keller A.: **Quasi Monte Carlo Methods for Realistic Image Synthesis** (PhD thesis), 1998
- Kniss J. et al.: **Interactive Translucent Volume Rendering and Procedural Modelling**, 2002
- Lafortune E. P. and Willems Y. D.: **Rendering Participating Media with Bidirectional Path Tracing**, 1996
- Pauly M. et al.: **Metropolis Light Transport for Participating Media**, 2000
- Preetham A. J. et al.: **A Practical Analytic Model for Daylight**, 1999
- Purcell T. J. et al.: **Photon Mapping on Programmable Graphics Hardware**, 2003
- Raab M. et al.: **Unbiased Global Illumination with Participating Media**, 2006
- Riley K. et al.: **Efficient Rendering of Atmospheric Phenomena**, 2004
- Sun B. et al.: **A Practical Analytic Single Scattering Model for Real Time Rendering**, 2005
- Szirmay-Kalos L. et al.: **Real-Time Multiple Scattering in Participating Media with Illumination Networks**, 2005
- Tristan L.: **Soft Particles**, 2007
- Veach E.: **Robust Monte Carlo Methods for Light Transport Simulation** (PhD thesis), 1998
- Wang N.: **Realistic and Fast Cloud Rendering**, 2003
- Woodcock E. et al.: **Techniques Used in the GEM Code for Monte Carlo Neutronics Calculations in Reactors and Other Systems of Complex Geometry**, 1965
- Yue Y. et al.: **Unbiased Adaptive Stochastic Sampling for Rendering Inhomogeneous Participating Media**, 2010
- Zotti G. et al.: **A Critical Review of the Preetham Skylight Model**, 2007