Interactive Rendering of Atmospheric Scattering Effects Using Graphics Hardware

Yoshinori Dobashi *Hokkaido University*

Tsuyoshi Yamamoto *Hokkaido University*

Tomoyuki Nishita *Tokyo University*

Overview

- Introduction
  - motivation
  - previous work
- Rendering Light Beams
  - basic Idea
  - problems
  - high quality rendering
- Rendering the Earth’s Atmosphere
  - rendering sky
  - rendering the earth viewed from space
- Results
- Conclusion

Overview

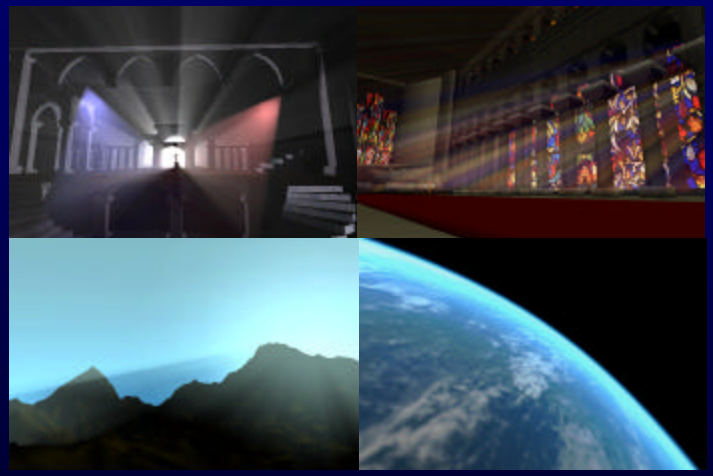
- Introduction
  - motivation
  - previous work
- Rendering Light Beams
  - basic Idea
  - problems
  - high quality rendering
- Rendering the Earth’s Atmosphere
  - rendering sky
  - rendering the earth viewed from space
- Results
- Conclusion

Overview

- **Introduction**
  - **motivation**
  - **previous work**
- **Rendering Light Beams**
  - **basic Idea**
  - **problems**
  - **high quality rendering**
- **Rendering the Earth's Atmosphere**
  - **rendering sky**
  - **rendering the earth viewed from space**
- **Results**
- **Conclusion**

Motivation

- **Real-time rendering of realistic images**
- **Atmospheric Scattering Effects**
  - scattering and absorption of light due to small particles
  - spotlights
  - sunlight through windows
  - earth's atmosphere



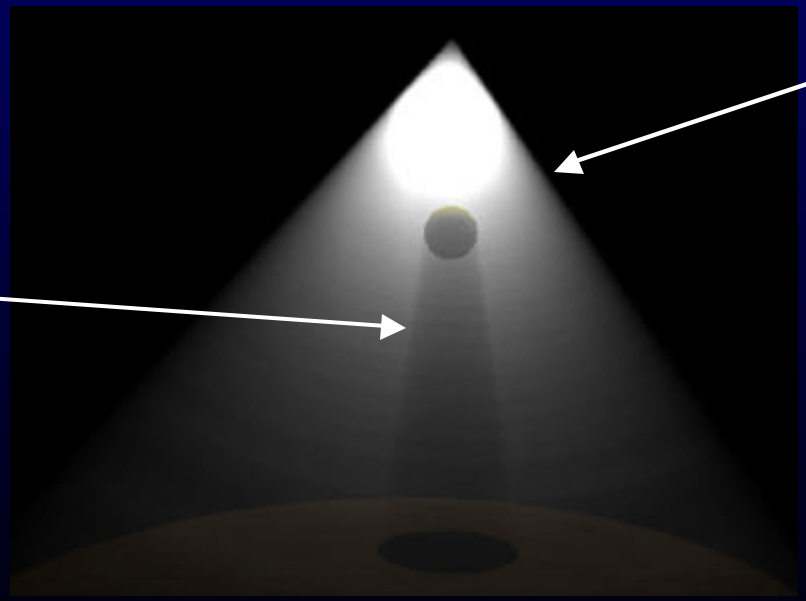
requires long computation time

- **Our Goal:**

Real-time rendering of atmospheric effects

Previous Work

- **Volume rendering** [e.g. Behrens98, Westermann98]
  - use of voxels to store the intensity of light
  - consuming texture memory for volume data
  - difficult to capture the edges of shafts of light



Previous Work

- **2D texture based approach** [Dobashi01, Everitt99]
  - use of shadow and projective texture mapping
  - artifacts due to sampling errors
- **Interleaved sampling** [Keller01]
  - a general and efficient solution to the sampling problem
  - not optimal for rendering atmospheric scattering
- **Earth's atmosphere**
  - no methods for real-time rendering of realistic images

Proposed Method

- Precise and efficient rendering of atmospheric effects
- Rendering light beams
  - point/infinite light sources
  - uniform density of atmospheric particles
- Rendering earth's atmosphere
  - sky
  - the earth viewed from space
  - density decreases exponentially according to the height from the ground

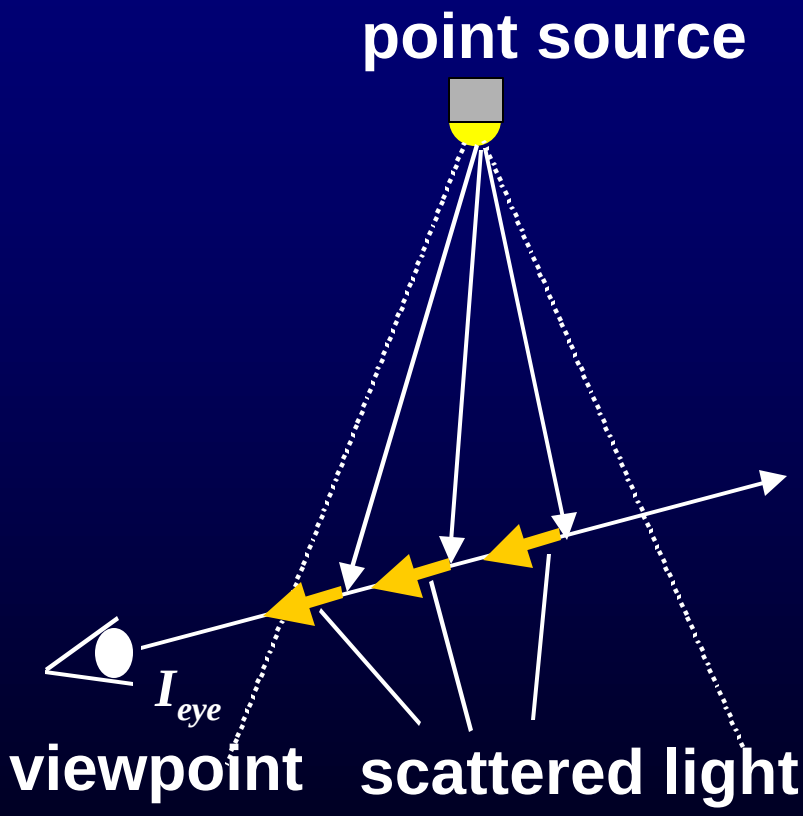
Overview

- Introduction
  - motivation
  - previous work
- **Rendering Light Beams**
  - **basic Idea**
  - **problems**
  - **high quality rendering**
- Rendering the Earth's Atmosphere
  - rendering sky
  - rendering the earth viewed from space
- Results
- Conclusion

Shading Model for Light Beams

Intensity at viewpoint

$$I_s(l) = \int_0^T I_l(t, l) H(t) g(t, l) dt$$



Shading Model for Light Beams

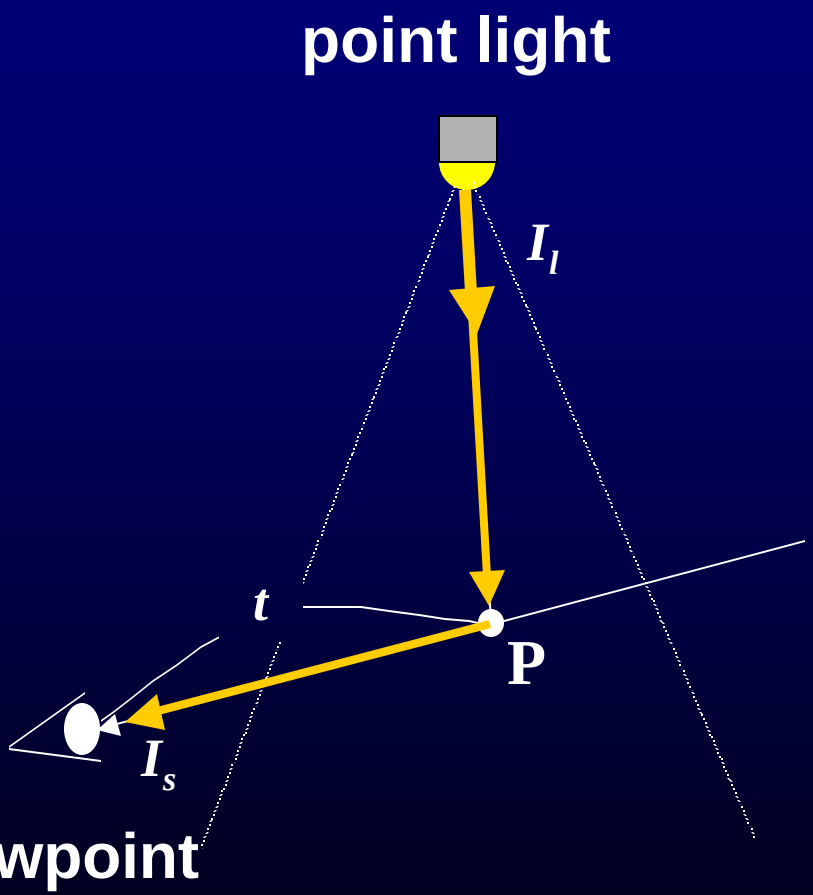
Intensity at viewpoint

$$I_s(\lambda) = \int_0^T I_l(t, \lambda) H(t) g(t, \lambda) dt$$

$I_l(t, \lambda)$ intensity of light

$H(t)$ visibility function

$g(t, \lambda)$ attenuation function

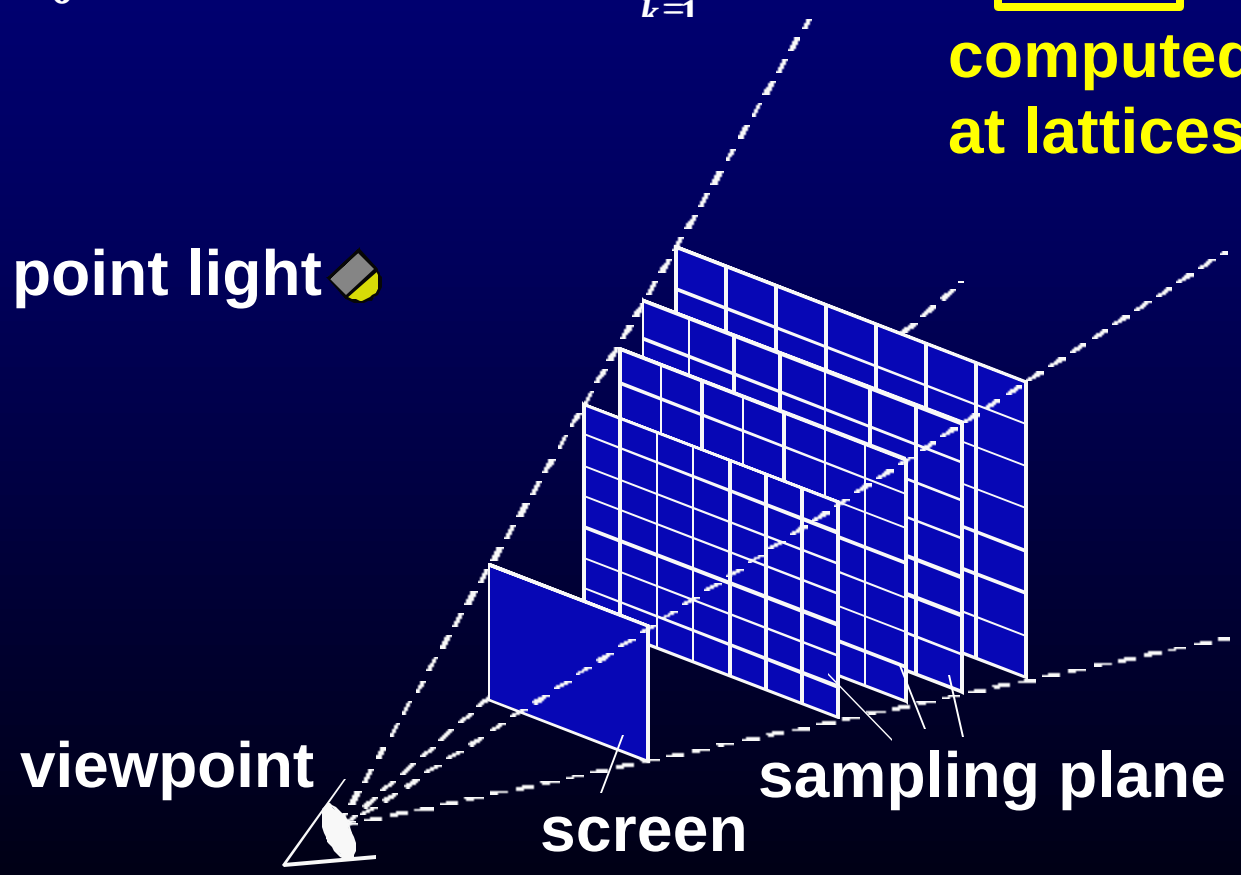


Basic Idea [Dobashi00]

Intensity at viewpoint

$$I_s(\lambda) = \int_0^T I_l(t,\lambda) H(t) g(t,\lambda)\, dt = \sum_{k=1}^{n} I_l(t,\lambda) H(t) \boxed{g(t,\lambda)}$$

computed at lattices

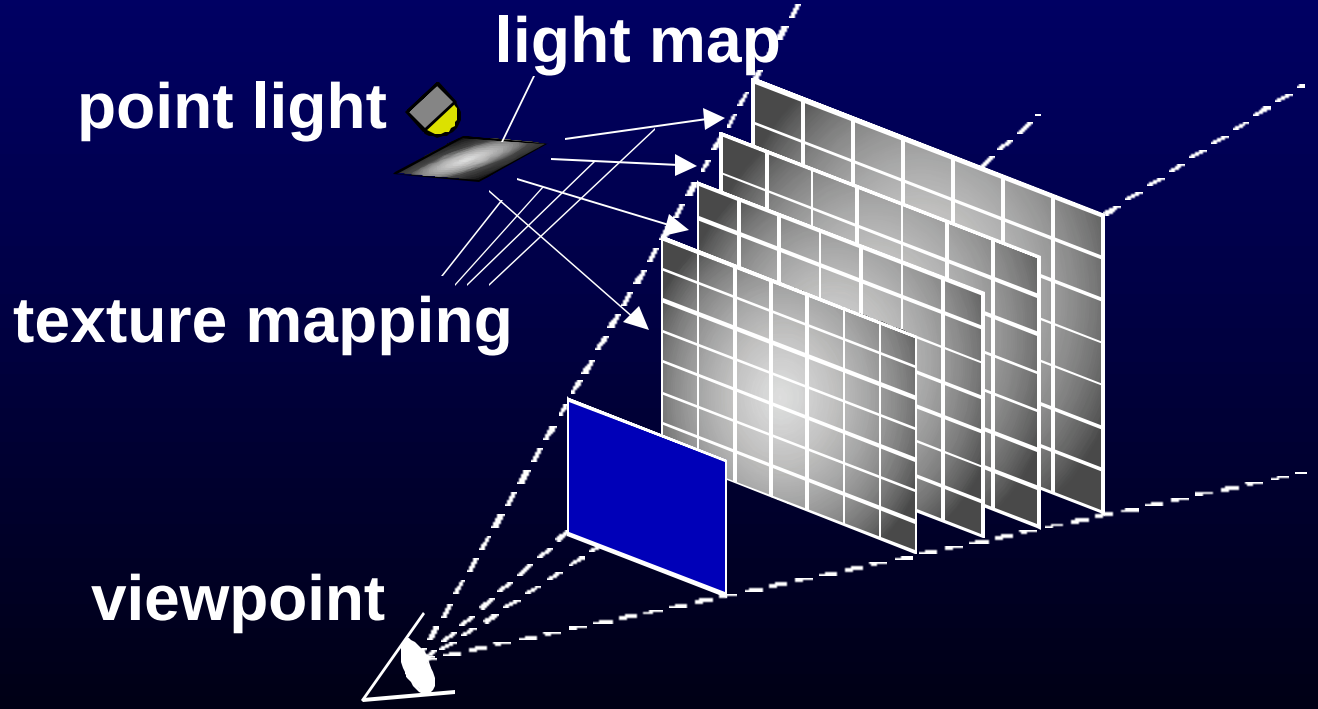


Basic Idea [Dobashi00]

Intensity at viewpoint

$$I_s(l) = \int_0^T I_l(t, l) H(t) g(t, l) dt = \sum_{k=1}^{n} I_l(t, l) H(t) g(t, l)$$

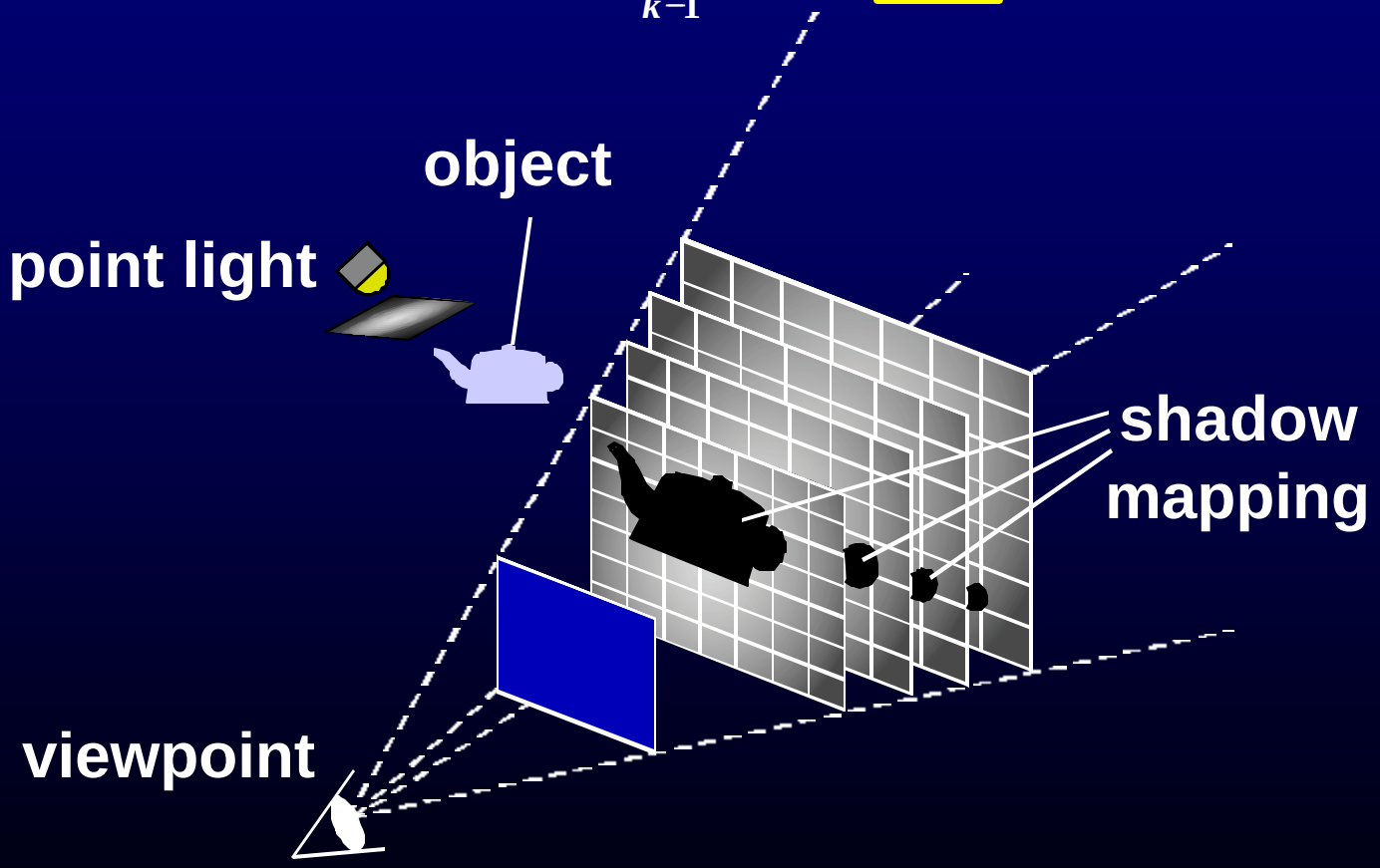
ight map



Basic Idea [Dobashi00]

Intensity at viewpoint

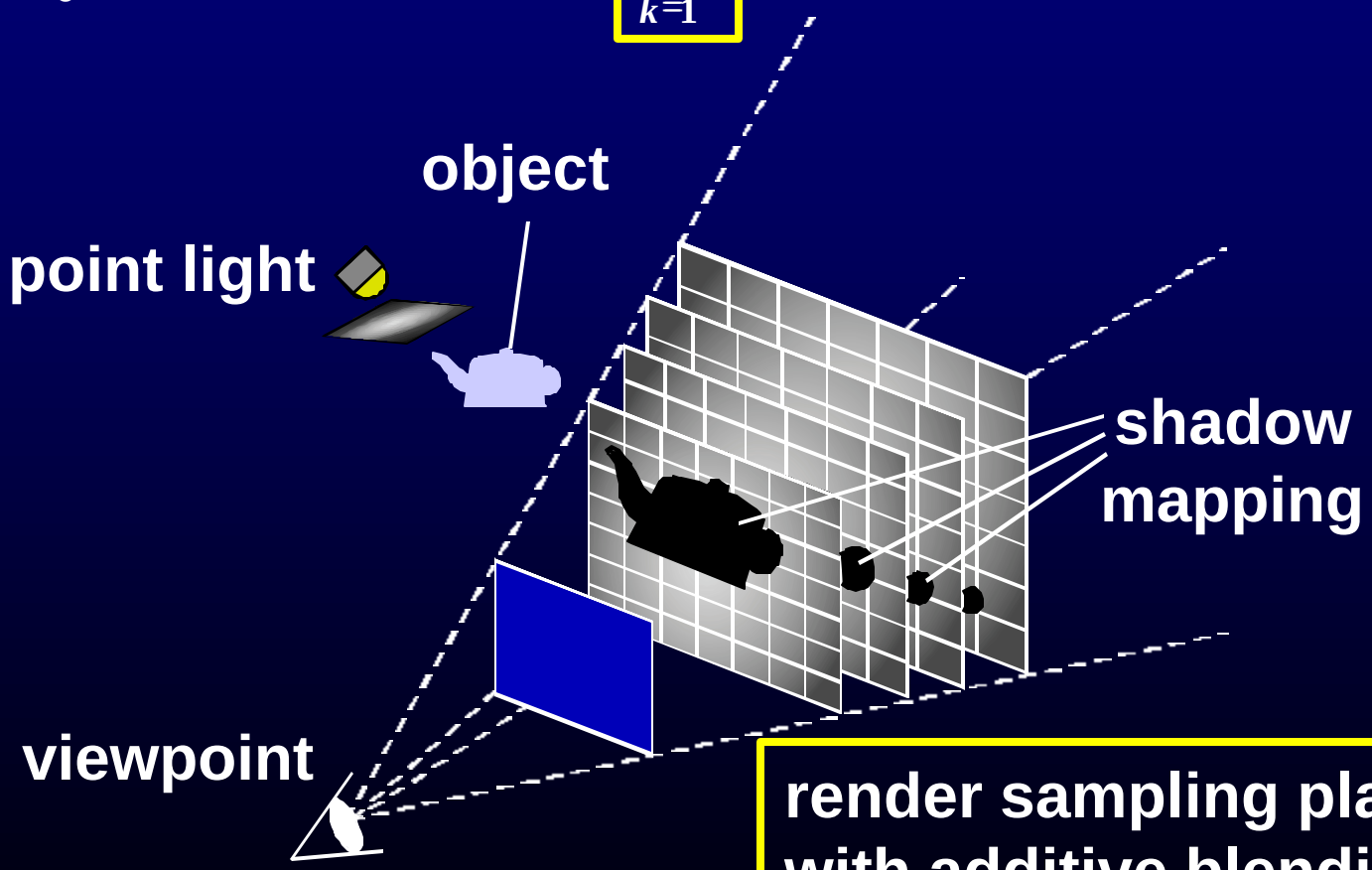
$$I_s(l) = \int_0^T I_l(t,l)H(t)g(t,l)dt = \sum_{k=1}^{n} I_l(t,l)\boxed{H(t)}g(t,l)$$



Basic Idea [Dobashi00]

Intensity at viewpoint

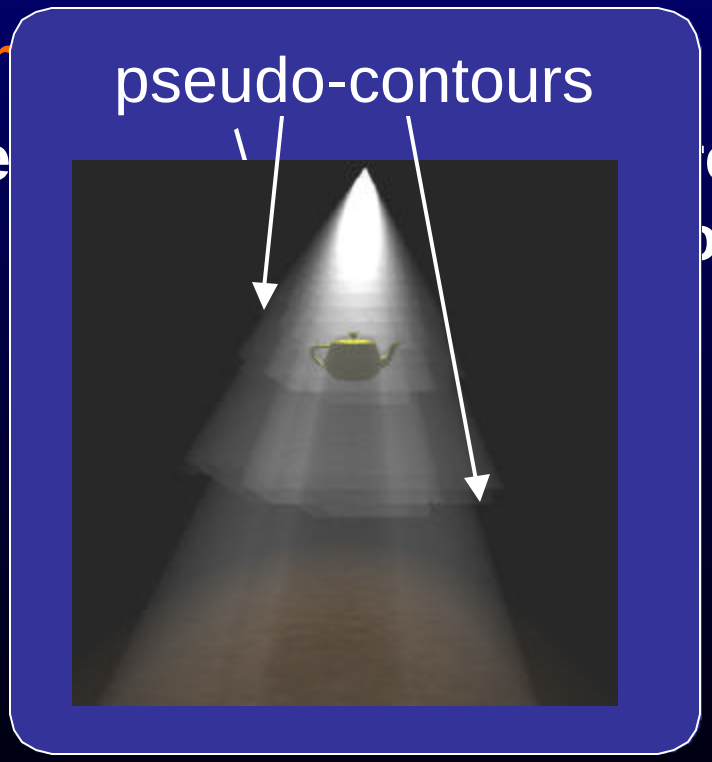
$$I_s(\lambda) = \int_0^T I_l(t,\lambda) H(t) g(t,\lambda) dt = \sum_{k=1}^{n} I_l(t,\lambda) H(t) g(t,\lambda)$$



render sampling planes with additive blending

Problems

- Accuracy μ { number of planes / number of lattice points
- Many planes/lattices for high quality image
  - artifacts due to quantization
    - quantization with 8 bit pre … e
    - accumulation of errors in … of sampling planes
  - increase in rendering time
    - rendering time μ { number … / number …



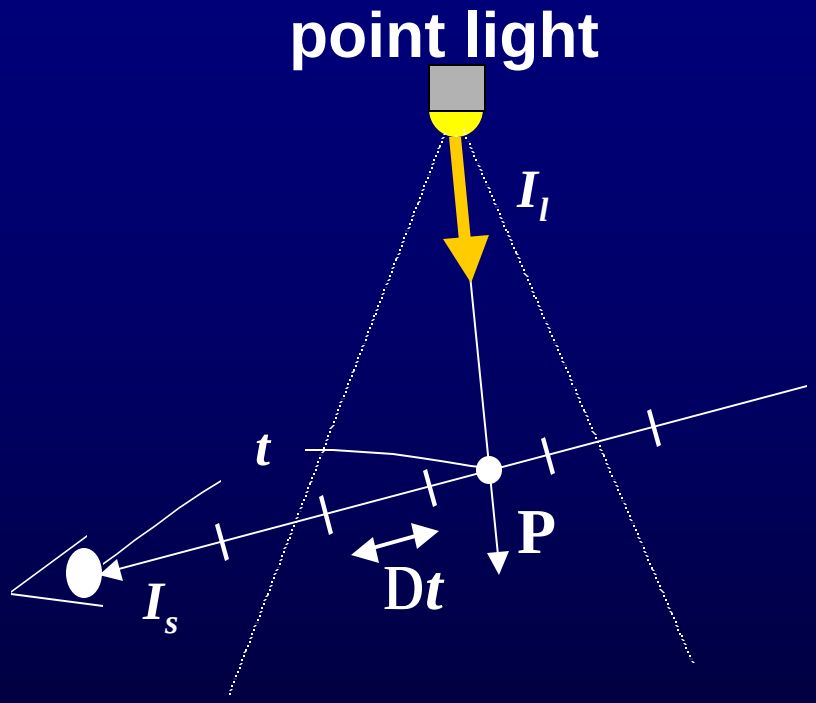
High Quality Rendering

Intensity at viewpoint

$$I_s(\lambda) = \int_0^T I_l(t,\lambda) H(t) g(t,\lambda)\, dt$$

$$= \sum_{k=1}^{n} \Delta I_s(t,\lambda)$$

$$\Delta I_s(t_k,\lambda) = \int_{t_k}^{t_k+\Delta t} I_l(t,\lambda) H(t) g(t,\lambda)\, dt$$



High Quality Rendering

Intensity at viewpoint

$$I_s(\lambda) = \int_0^T I_l(t,\lambda) H(t) g(t,\lambda)\, dt$$

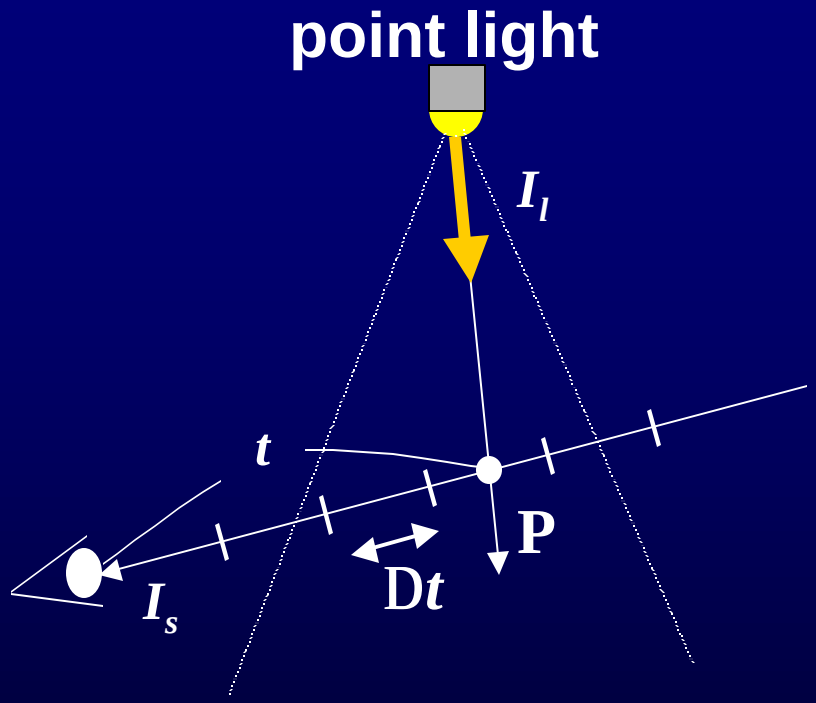
$$= \sum_{k=1}^{n} \Delta I_s(t,\lambda)$$

$$\Delta I_s(t_k,\lambda) = \int_{t_k}^{t_k+\Delta t} I_l(t,\lambda) H(t) g(t,\lambda)\, dt$$

changes severely | changes smoothly

=

const. in Δt



High Quality Rendering

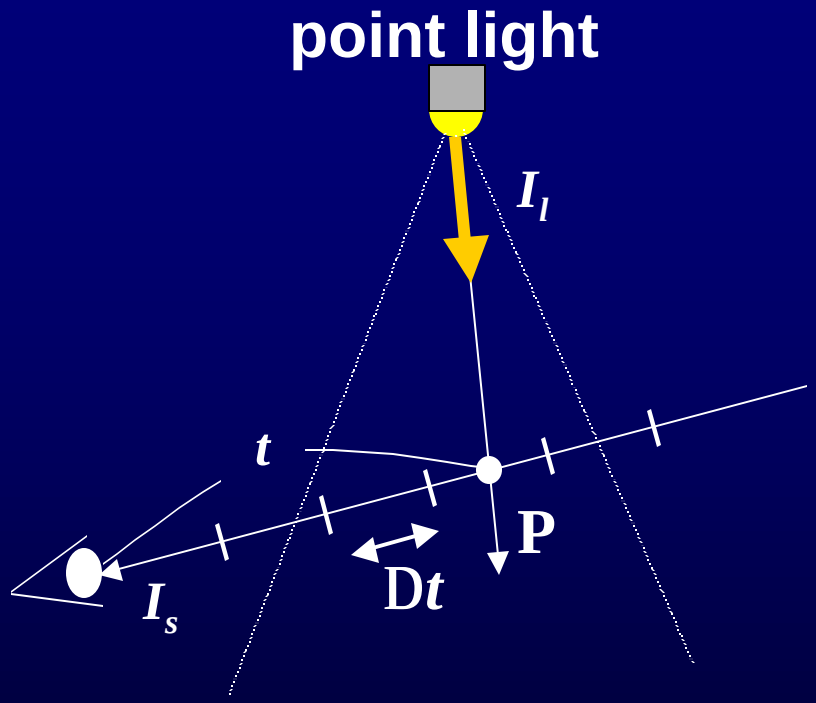
Intensity at viewpoint

$$I_s(\lambda) = \int_0^T I_l(t,\lambda) H(t) g(t,\lambda)\, dt$$

$$= \sum_{k=1}^{n} \Delta I_s(t,\lambda)$$

$$\Delta I_s(t_k,\lambda) = \int_{t_k}^{t_k+\Delta t} I_l(t,\lambda) H(t) g(t,\lambda)\, dt$$

$$\approx \underbrace{\left(\int_{t_k}^{t_k+\Delta t} I_l(t,\lambda) H(t)\, dt \right)}_{f_h} \times \underbrace{\left(\int_{t_k}^{t_k+\Delta t} g(t,\lambda)\, dt \right)}_{f_l}$$



High Quality Rendering

Intensity at viewpoint

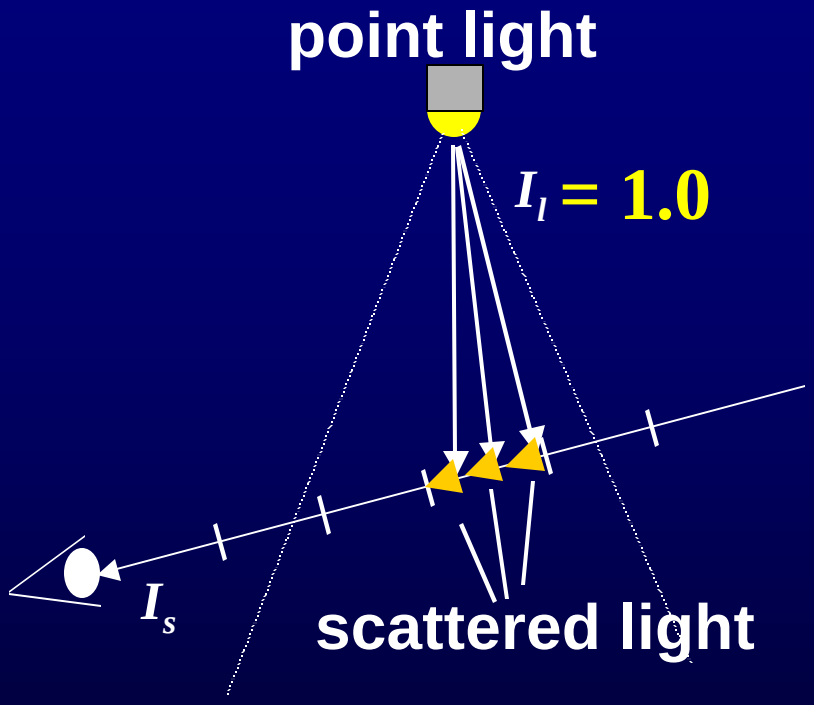
$$I_s(\lambda) = \int_0^T I_l(t,\lambda) H(t) g(t,\lambda)\, dt$$

$$= \sum_{k=1}^{n} \Delta I_s(t,\lambda)$$

$$\Delta I_s(t_k,\lambda) = \int_{t_k}^{t_k+\Delta t} I_l(t,\lambda) H(t) g(t,\lambda)\, dt$$

$$\approx \left(\int_{t_k}^{t_k+\Delta t} I_l(t,\lambda) H(t)\, dt \right) \times \left(\int_{t_k}^{t_k+\Delta t} g(t,\lambda)\, dt \right)$$

f_h f_l



scattering component

High Quality Rendering

Intensity at viewpoint

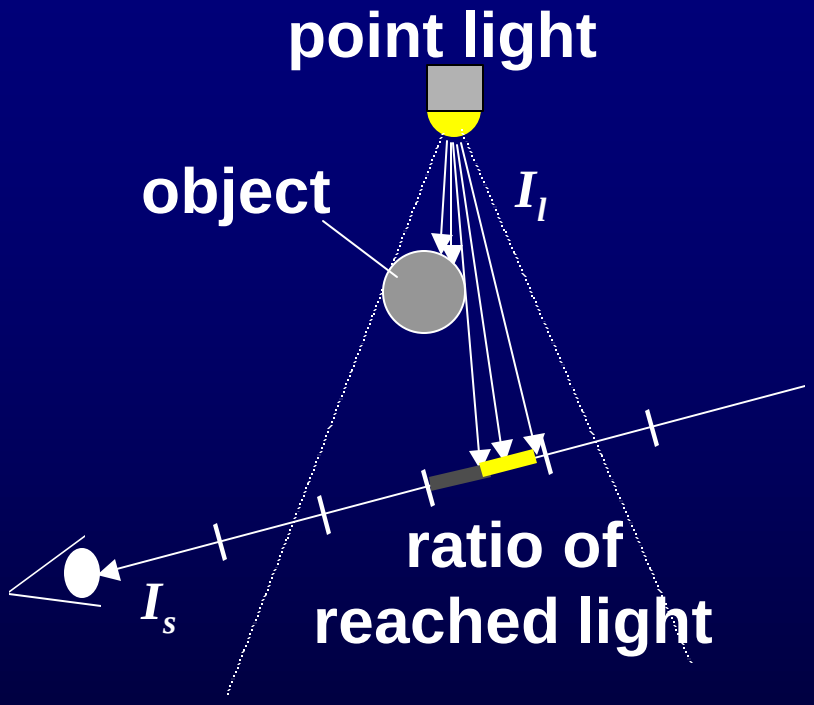
$$I_s(\lambda) = \int_0^T I_l(t,\lambda) H(t) g(t,\lambda)\, dt$$

$$= \sum_{k=1}^{n} \Delta I_s(t,\lambda)$$

$$\Delta I_s(t_k,\lambda) = \int_{t_k}^{t_k+\Delta t} I_l(t,\lambda) H(t) g(t,\lambda)\, dt$$

$$\approx \left(\int_{t_k}^{t_k+\Delta t} I_l(t,\lambda) H(t)\, dt \right) \times \left(\int_{t_k}^{t_k+\Delta t} g(t,\lambda)\, dt \right)$$

f_h f_l



Illumination component	scattering component

High Quality Rendering

- **Scattering component** $\left[f_l : \int_{t_k}^{t_k + \Delta t} g(t, l)\, dt \right]$
  - changes smoothly
  - can be sampled at a large interval

  use of texture to store pre-integrated values

- **Illumination component** $\left[f_h : \int_{t_k}^{t_k + \Delta t} I_l(t, l) H(t)\, dt \right]$
  - includes visibility H and intensity distribution I_l
  - must be sampled at a short interval

  sub-planes for accurate sampling

High Quality Rendering

- **Scattering component** $\left[f_l : \int_{t_k}^{t_k + \Delta t} g(t, l)\, dt \right]$

  - changes smoothly
  - can be sampled at a large interval

  use of texture to store pre-integrated values

- **Illumination component** $\left[f_h : \int_{t_k}^{t_k + \Delta t} I_l(t, l) H(t)\, dt \right]$

  - includes visibility H and intensity distribution I_l
  - must be sampled at a short interval

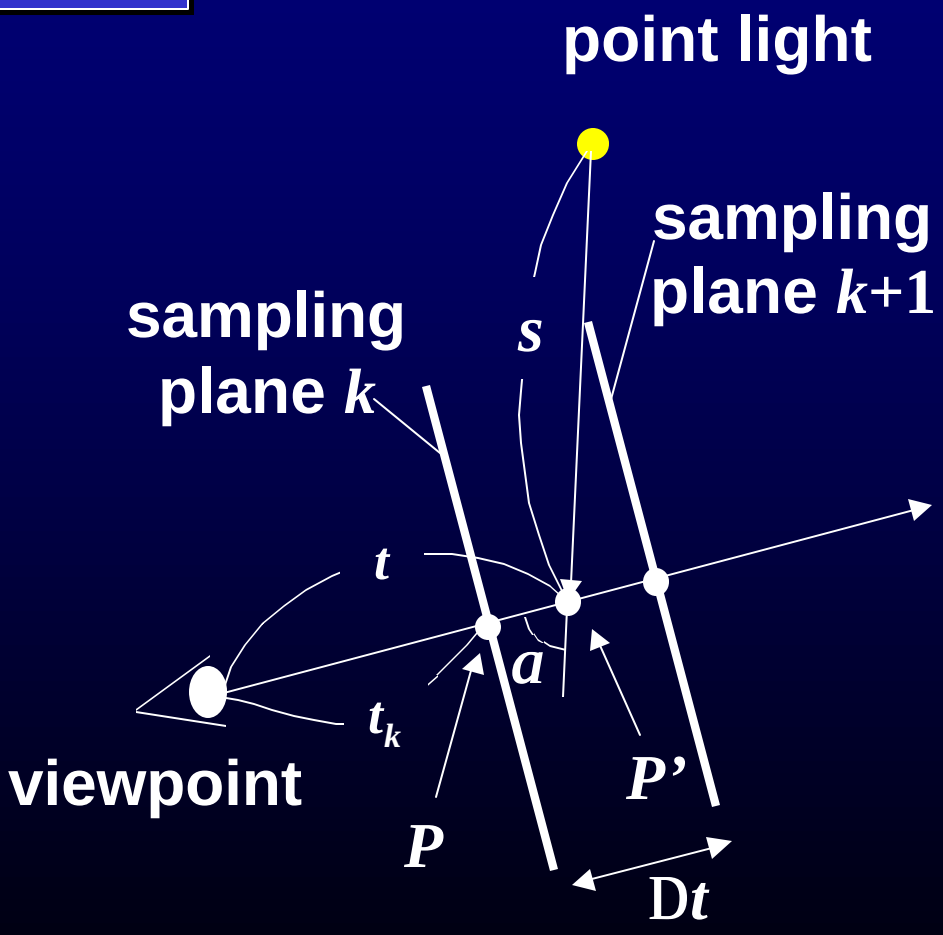
  sub-planes for accurate sampling

Textures for Scattering Component

$$f_l(t_k, l) = \int_{t_k}^{t_k + \Delta t} r_c F(a, l) \frac{\exp(-br_c(s+t))}{s^2} dt$$

(point light)

- r_c density
- b attenuation ratio
- t_k distance between viewpoint and P
- s distance between light and P'
- t distance between viewpoint and P'
- $F(a, l)$ phase function
- a phase angle



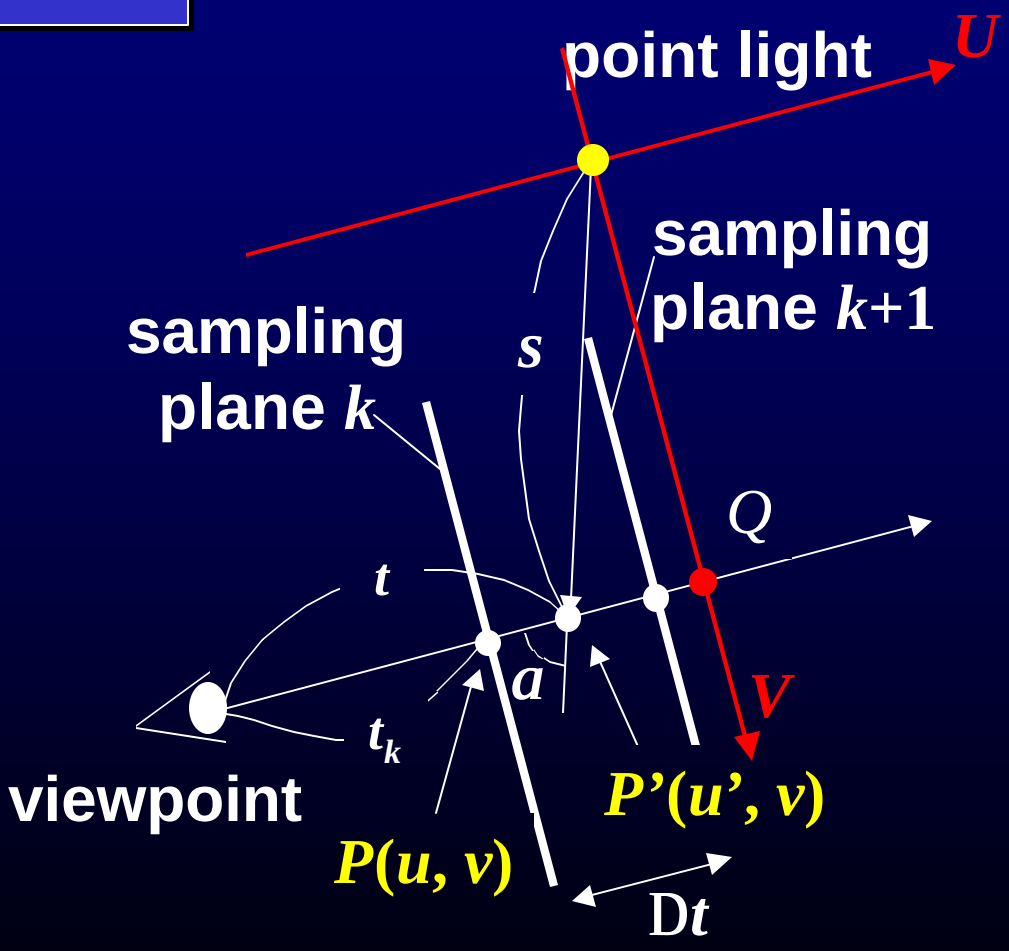
Textures for Scattering Component

$$f_l(t_k, l) = \int_{t_k}^{t_k + \Delta t} r_c F(a, l) \frac{\exp(-br_c(s+t))}{s^2} dt$$

(point light)

- local coordinate UV

$$\begin{cases} t = u' - u + t_k \\ \cos a = -u'/s \\ s^2 = u'^2 + v^2 \end{cases}$$



Textures for Scattering Component

$$f_l(u,v,l) = c_k q(u,v,l)$$

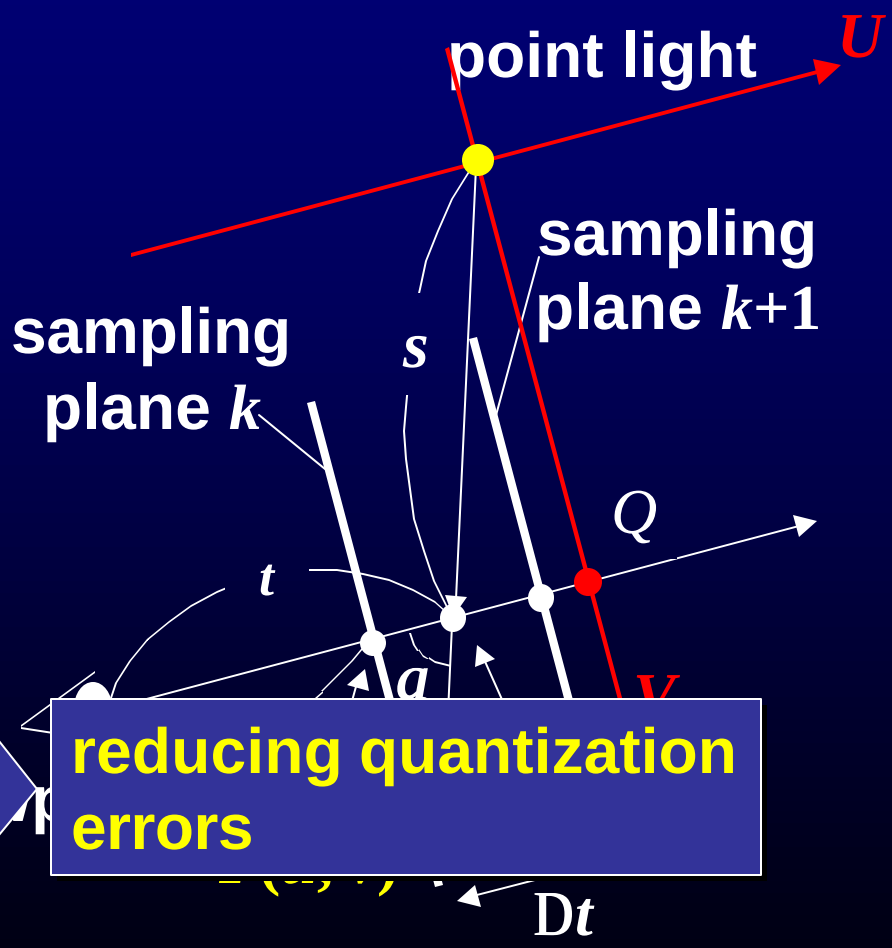
$$c_k = \exp(-br_c t_k)$$

$$q(u,v,l) = \int_u^{u+\Delta t} r_c F(\cos^{-1}(-u'/\sqrt{u'^2+v^2}),l) \times \frac{\exp(-br_c(\sqrt{u'^2+v^2}+u'-u))}{u'^2+v^2} du'$$

c_k : constant for each sampling plane

$q(u,v,l)$: 2D texture

f_l is evaluated precisely since texture stores integrated values



reducing quantization errors

High Quality Rendering

- **Scattering component** $\left[f_l : \int_{t_k}^{t_k + \Delta t} g(t, l)\, dt \right]$
  - changes smoothly
  - can be sampled at a large interval

  use of texture to store pre-integrated values

- **Illumination component** $\left[f_h : \int_{t_k}^{t_k + \Delta t} I_l(t, l) H(t)\, dt \right]$
  - includes visibility H and intensity distribution I_l
  - must be sampled at a short interval

  sub-planes for accurate sampling

Computation of Illumination Component

$$f_h(t_k, l) = \int_{t_k}^{t_k + \Delta t} I_l(t, l) H(t) dt$$

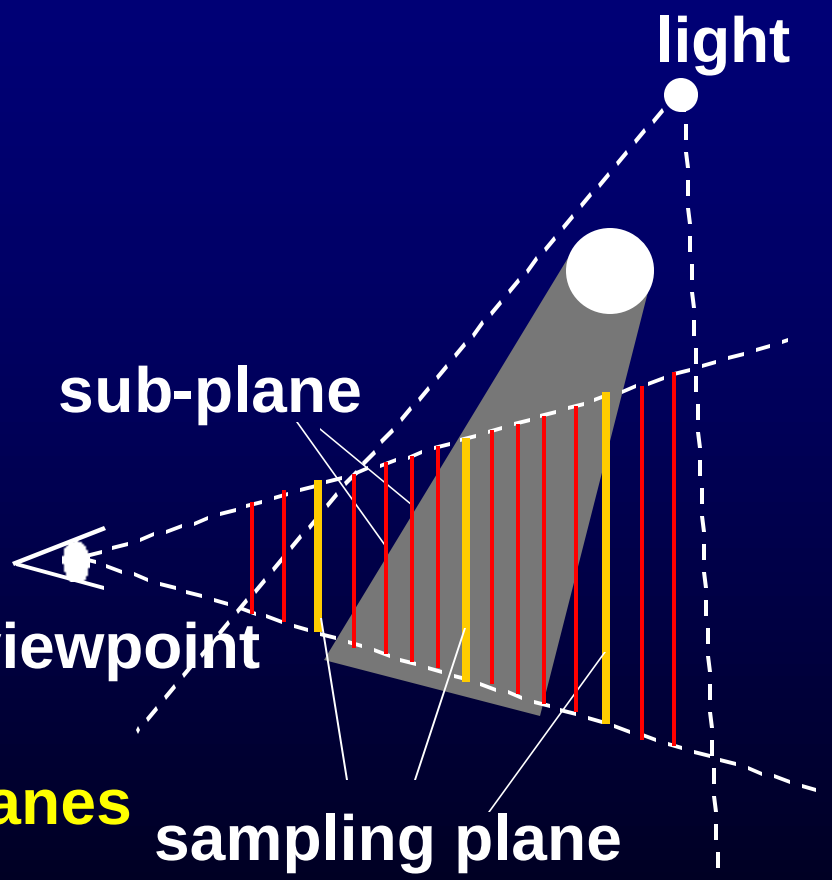
- m_k sub-planes between sampling planes k and $k+1$

$$f_h(t_k, l) = \frac{1}{m_k} \sum_{j=0}^{m_k - 1} I_l(r_j, l) H(r_j)$$

r_j : distance between viewpoint and sub-plane j

- m_k is determined adaptively

strong intensity → many sub-planes



Computation of Illumination Component

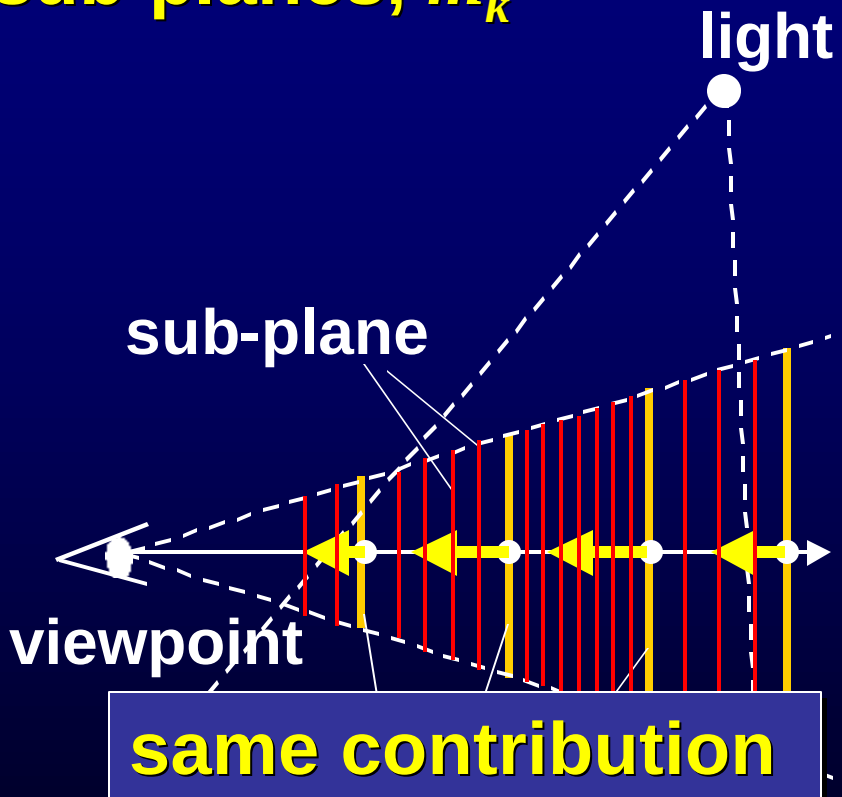
- **Determining number of sub-planes, m_k**

1. shoot a ray
2. compute intensity of scattered light

   (use textures for scattering component)
3. generate sub-planes in proportion to intensity

$$\frac{\text{Intensity of scattered light}}{m_k} \pounds e$$

(e :user-specified threshold)



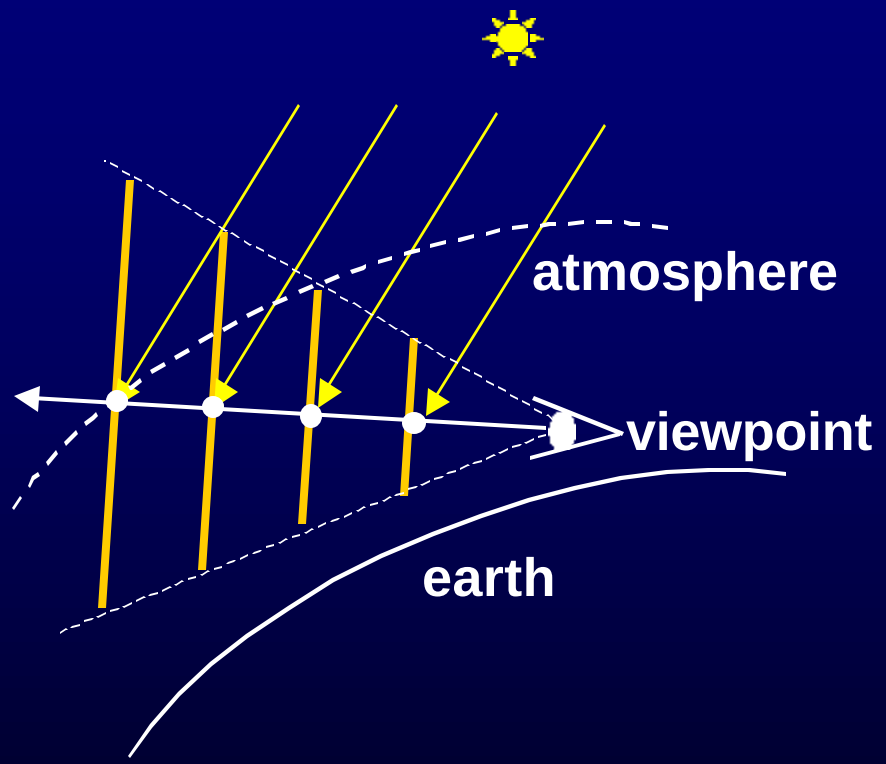
same contribution of each sub-plane to pixel intensity

Overview

- Introduction
  - motivation
  - previous work
- Rendering Light Beams
  - basic Idea
  - problems
  - high quality rendering
- **Rendering the Earth's Atmosphere**
  - **rendering sky**
  - **rendering the earth viewed from space**
- Results
- Conclusion

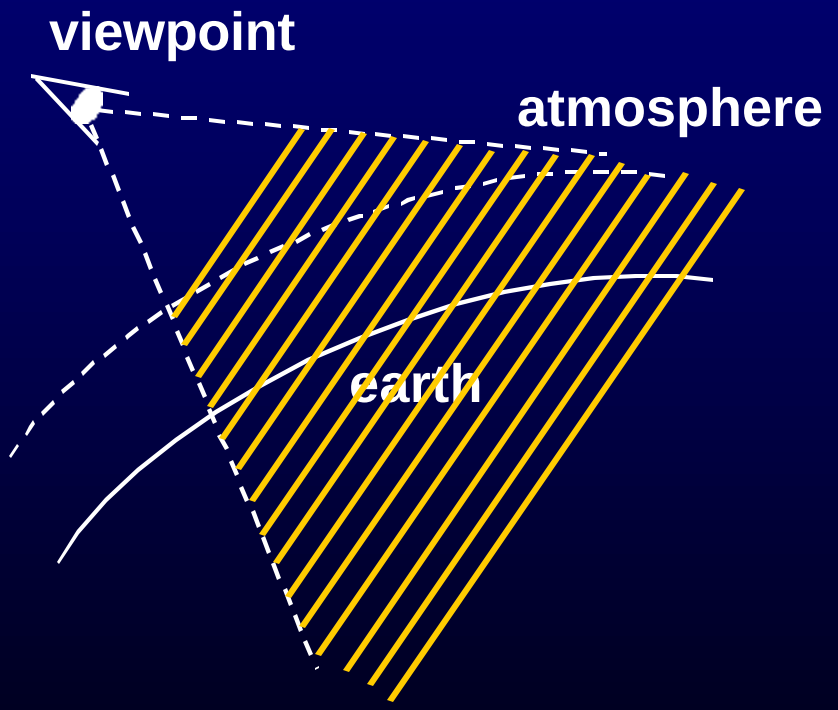
Rendering Earth's Atmosphere

- No shadows of objects
- Rendering Sky
  - extending method for light beams
- Rendering the earth viewed from space



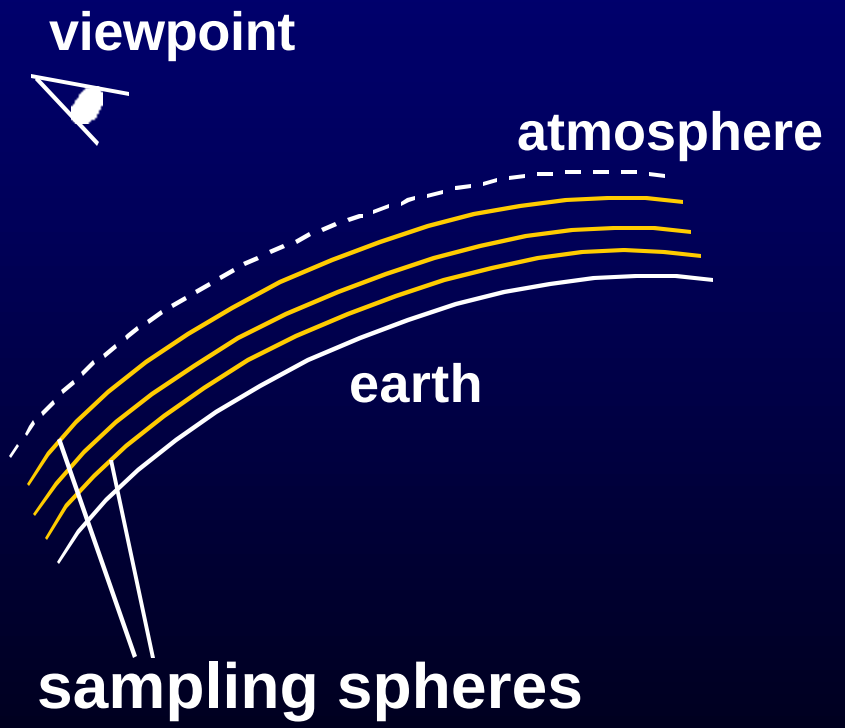
Rendering Earth's Atmosphere

- No shadows of objects
- Rendering Sky
  - extending method for light beams
- Rendering the earth viewed from space
  - atmosphere is very thin layer covering the earth



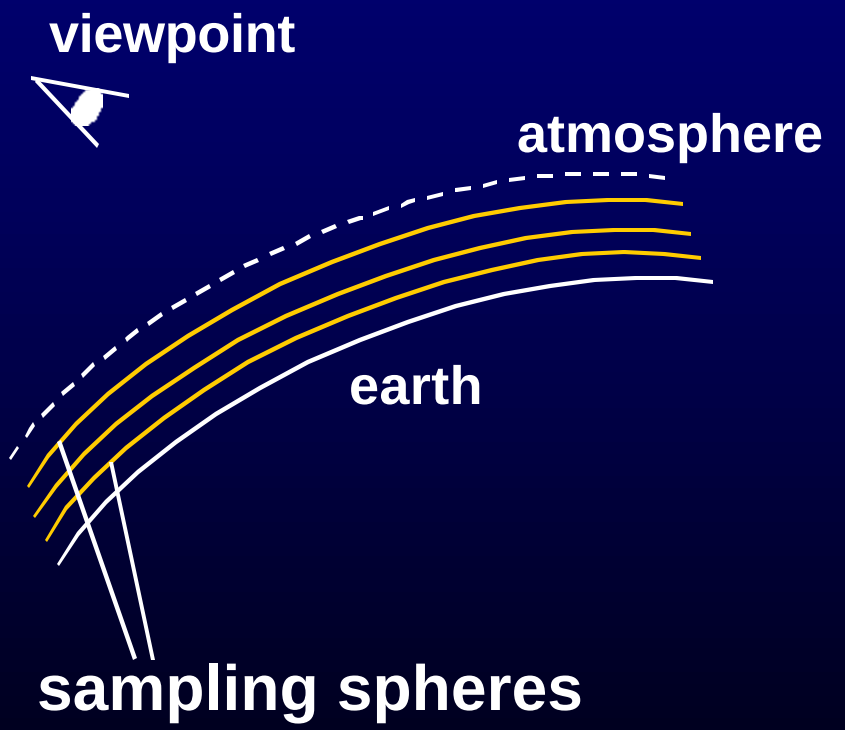
Rendering Earth's Atmosphere

- No shadows of objects
- Rendering Sky
  - extending method for light beams
- Rendering the earth viewed from space
  - atmosphere is very thin layer covering the earth
  - use of *sampling spheres* instead of sampling plane



Rendering Earth's Atmosphere

- **No shadows of objects**
- **Rendering Sky**
  - extending method for light beams
- **Rendering the earth viewed from space**
  - atmosphere is very thin layer covering the earth
  - use of *sampling spheres* instead of sampling plane



Rendering Sky

- **Intensity at viewpoint**

$$I_v(\lambda) = I_{sun}(\lambda)\int_0^T F(a,\lambda)\,r(t,\lambda)\,g_l(s,\lambda)\,g_v(t,\lambda)\,dt$$

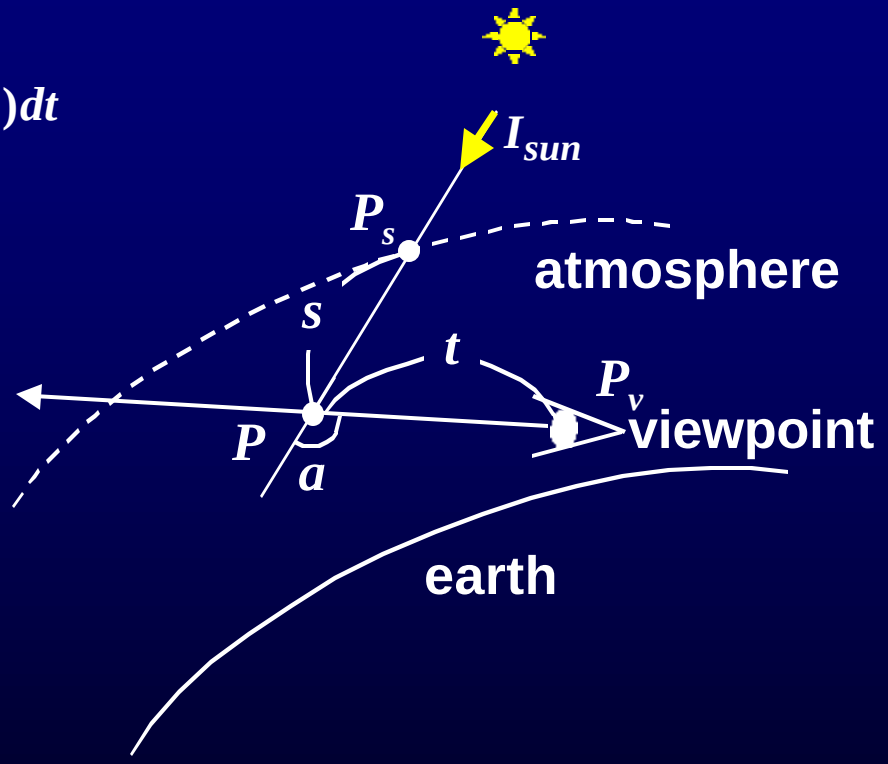
$I_{sun}(\lambda)$: intensity of sunlight

$F(a,\lambda)$: phase function

$r(t,\lambda)$: density of particles

$g_l(s,\lambda)$: attenuation ratio between P_s and P

$g_v(t,\lambda)$: attenuation ratio between P and P_v



Rendering Sky

- **Intensity at viewpoint**

$$I_v(\lambda) = I_{sun}(\lambda)\int_0^T F(a,\lambda)\,r(t,\lambda)\,g_\lambda(s,\lambda)\,g_v(t,\lambda)\,dt$$

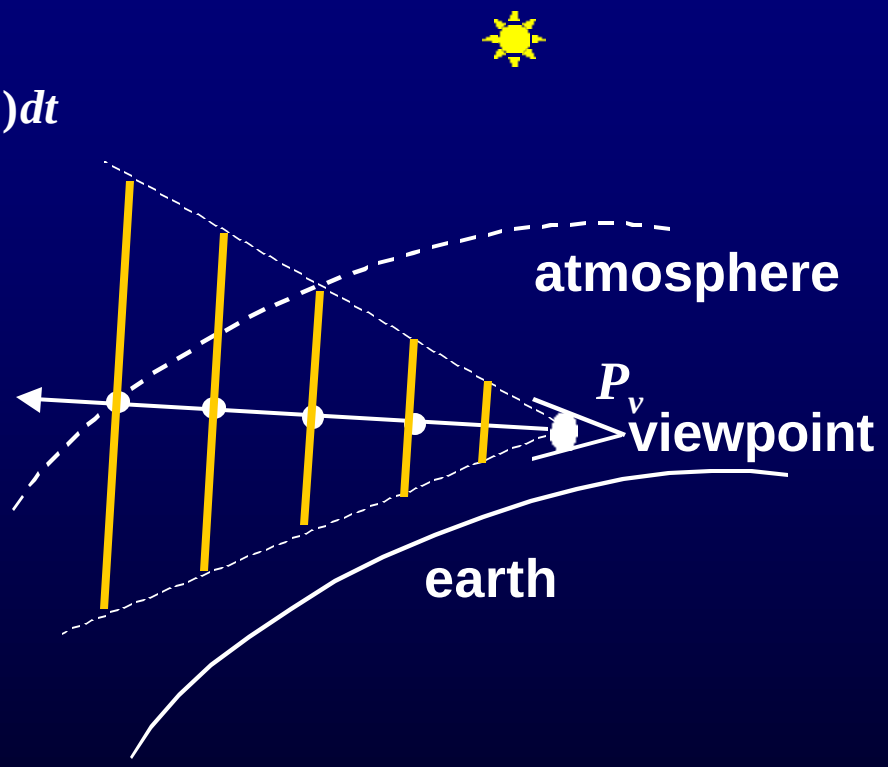
$I_{sun}(\lambda)$: intensity of sunlight

$F(a,\lambda)$: phase function

$r(t,\lambda)$: density of particles

$g_v(t,\lambda)$: attenuation ratio between P_s and P

$g_\lambda(s,\lambda)$: attenuation ratio between P and P_v



Rendering Sky

- **Intensity at viewpoint**

$$I_v(\lambda) = I_{sun}(\lambda) \sum_{k=1}^{n} F(a,\lambda) R(t) g_\lambda(s,\lambda) \prod_{j=1}^{k-1} \Delta g_v(t_j,\lambda)$$

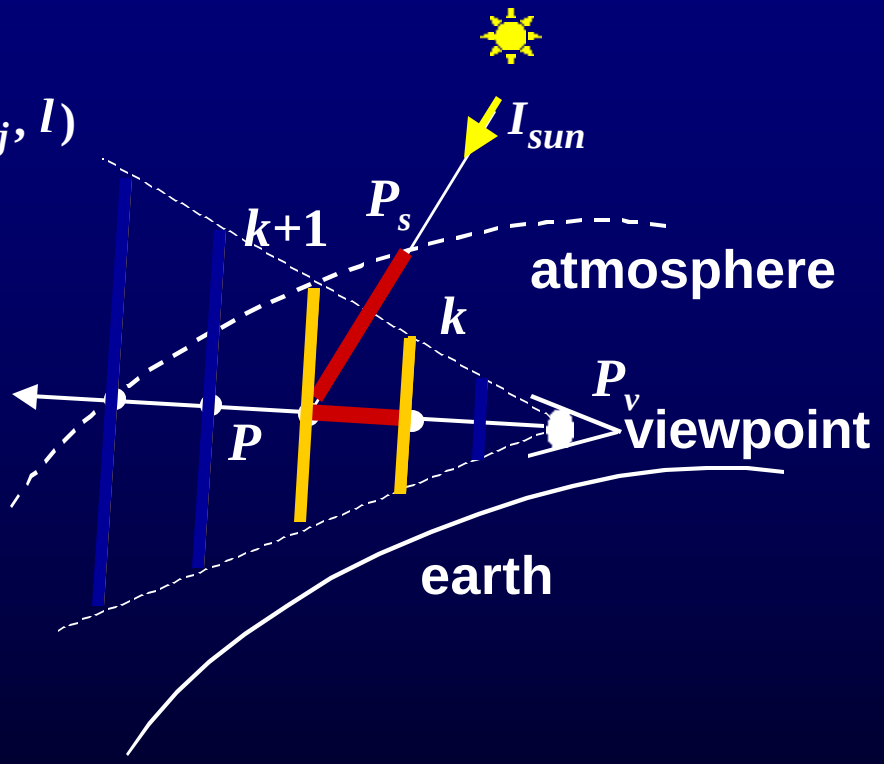
$I_{sun}(\lambda)$: intensity of sunlight

$g_\lambda(s,\lambda)$: attenuation ratio between P_s and P

$F(a,\lambda)$: phase function

$\Delta g_v(t_k,\lambda)$: attenuation between sampling planes k and k+1

$R(t)$: cumulative density of particles between sampling planes k and k+1

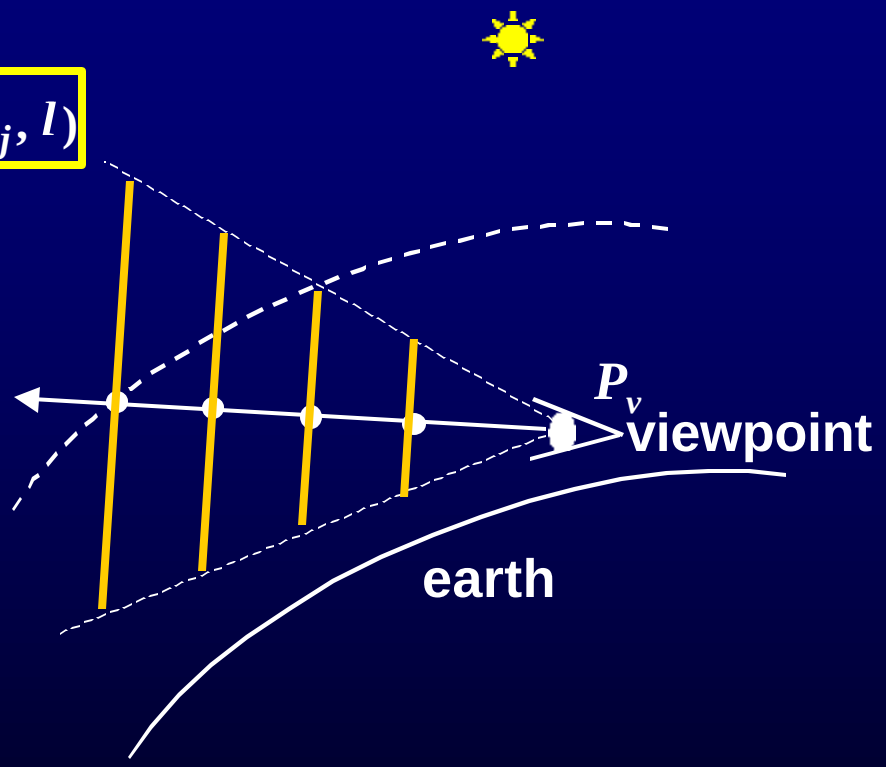


Rendering Sky

- **Intensity at viewpoint**

$$I_v(\lambda) = I_{sun}(\lambda) \sum_{k=1}^{n} F(a,\lambda) R(t) g_\lambda(s,\lambda) \prod_{j=1}^{k-1} \Delta g_v(t_j,\lambda)$$

- **Algorithm**
  - create textures of g_λ, Δg_v, R
  - for k = n to 1, repeat:
    - map g_λ, Δg_v, R textures onto sampling plane k

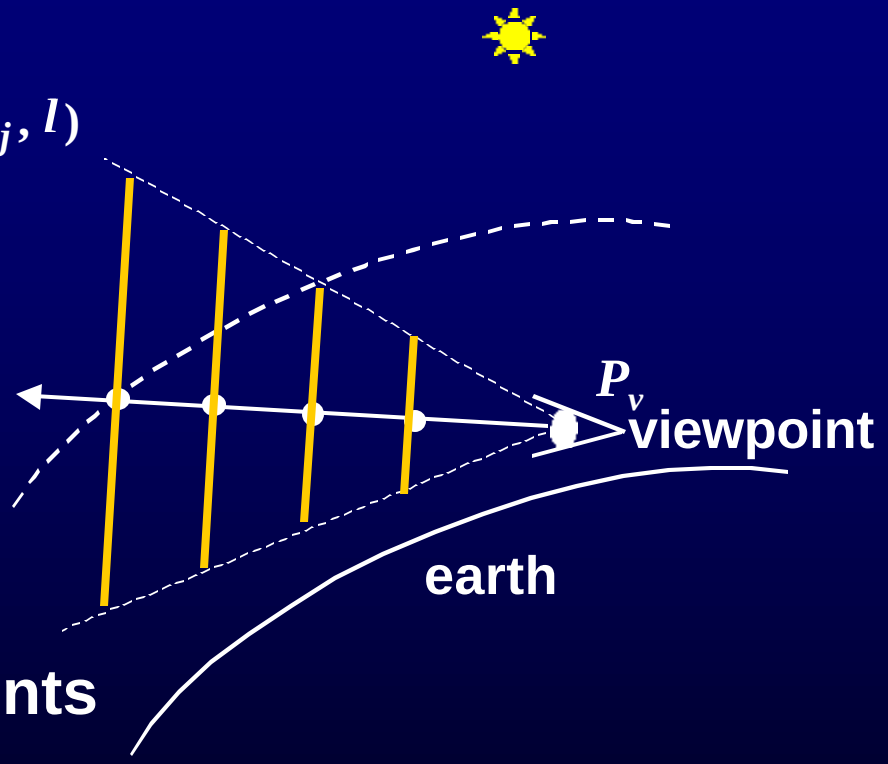


Rendering Sky

- **Intensity at viewpoint**

$$I_v(l) = I_{sun}(l) \sum_{k=1}^{n} F(a, l) R(t) g_l(s, l) \prod_{j=1}^{k-1} \Delta g_v(t_j, l)$$

- **Algorithm**
  - create textures of g_l, Δg_v, R
  - for k = n to 1, repeat:
    - map g_l, Δg_v, R textures onto sampling plane k
    - compute $I_{sun}F$ at lattice points



Rendering Sky

- **Intensity at viewpoint**

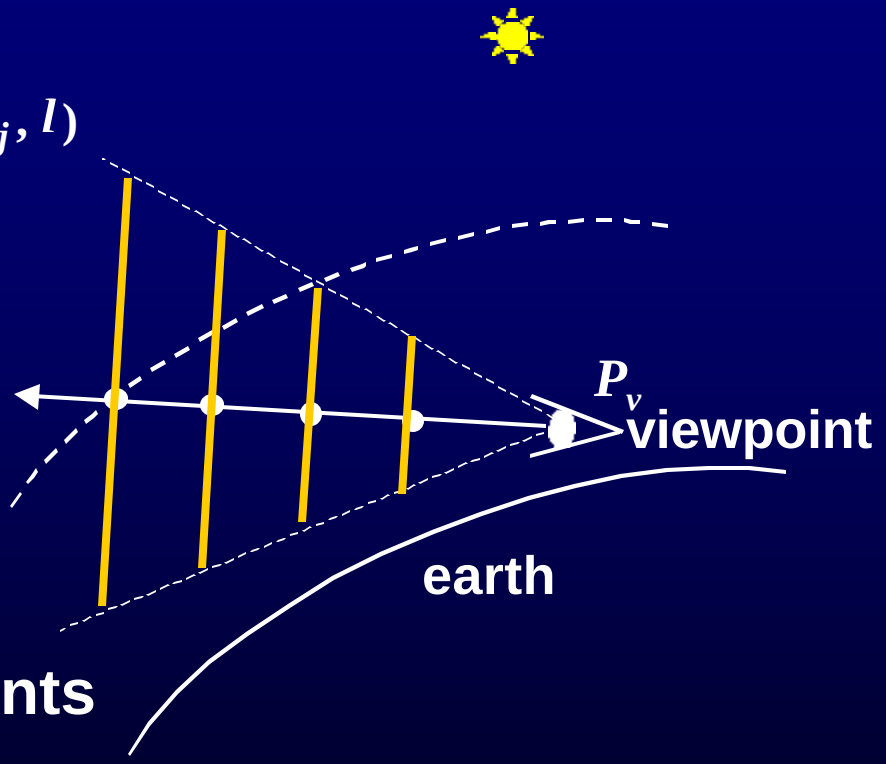
$$I_v(l) = I_{sun}(l) \sum_{k=1}^{n} F(a,l)\,R(t)\,g_l(s,l) \prod_{j=k-1}^{1} \Delta g_v(t_j,l)$$

- **Algorithm**
  - create textures of g_l, Δg_v, R
  - for k = n to 1, repeat:
    - map g_l, Δg_v, R textures onto sampling plane k
    - compute $I_{sun}F$ at lattice points
    - draw plane with blending:

$$FB = \underline{FB} \times \Delta g_v + \underline{F} \times R \times g_l \quad (FB\text{: frame buffer})$$

attenuation

intensity of scattered light



Rendering Sky

- **Intensity at viewpoint**

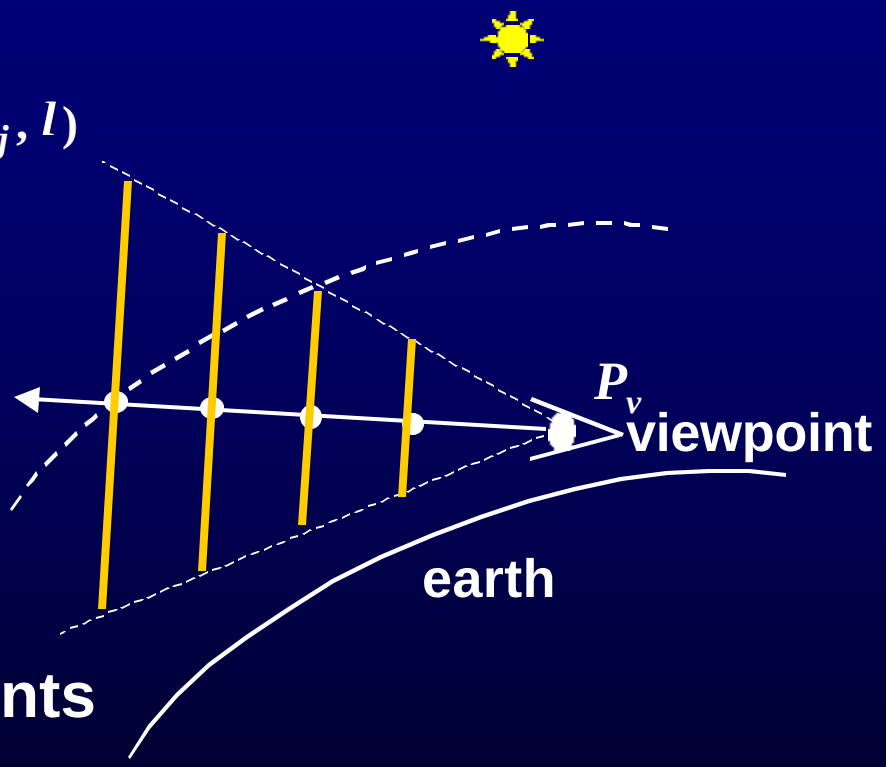
$$I_v(\lambda) = I_{sun}(\lambda) \sum_{k=1}^{n} F(a,\lambda) R(t) g_\lambda(s,\lambda) \prod_{j=1}^{k-1} Dg_v(t_j,\lambda)$$

- **Algorithm**
  - create textures of g_λ, Dg_v, R
  - for k = n to 1, repeat:
    - map g_λ, Dg_v, R textures onto sampling plane k
    - compute $I_{sun}F$ at lattice points
    - draw plane with blending:

$$FB = \underline{FB} \times Dg_v + \underline{F} \times R \times \underline{g_\lambda} \quad (FB\text{: frame buffer})$$

attenuation

intensity of scattered light



Rendering Sky

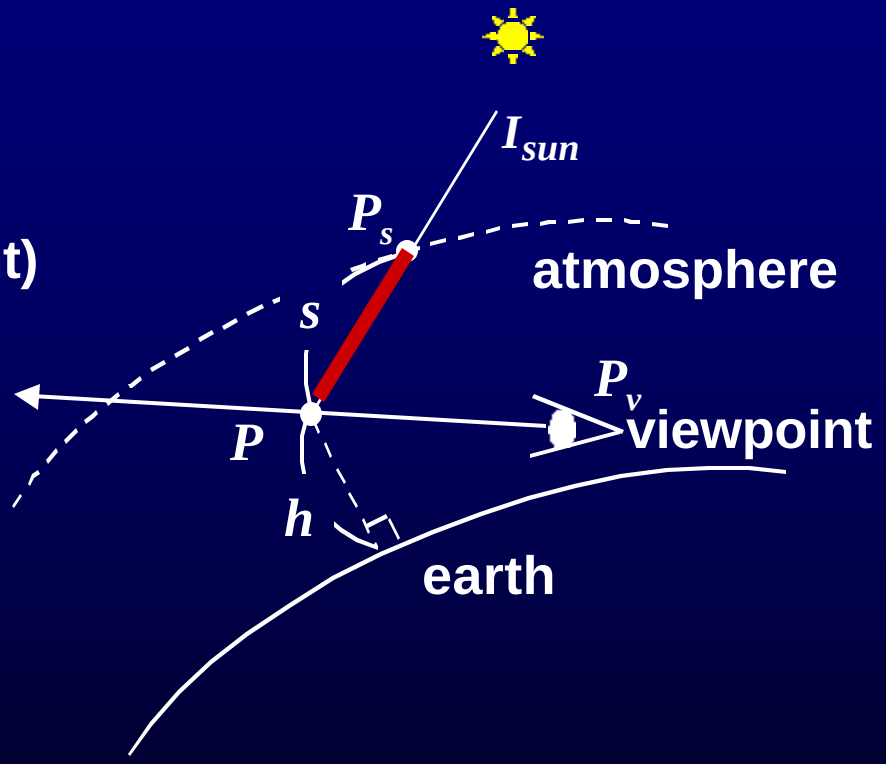
- Textures of g_l, Dg_v, R

$$g_l(PP_s, l) = \exp\left(-\int_0^{\overline{PP_s}} b(l)\, r(l)\, dl\right)$$

b : extinction coefficient (constant)

r : density of particles

$r(h) = \exp(-ah)$ (a: const)



Rendering Sky

- **Textures of g_l, Dg_v, R**

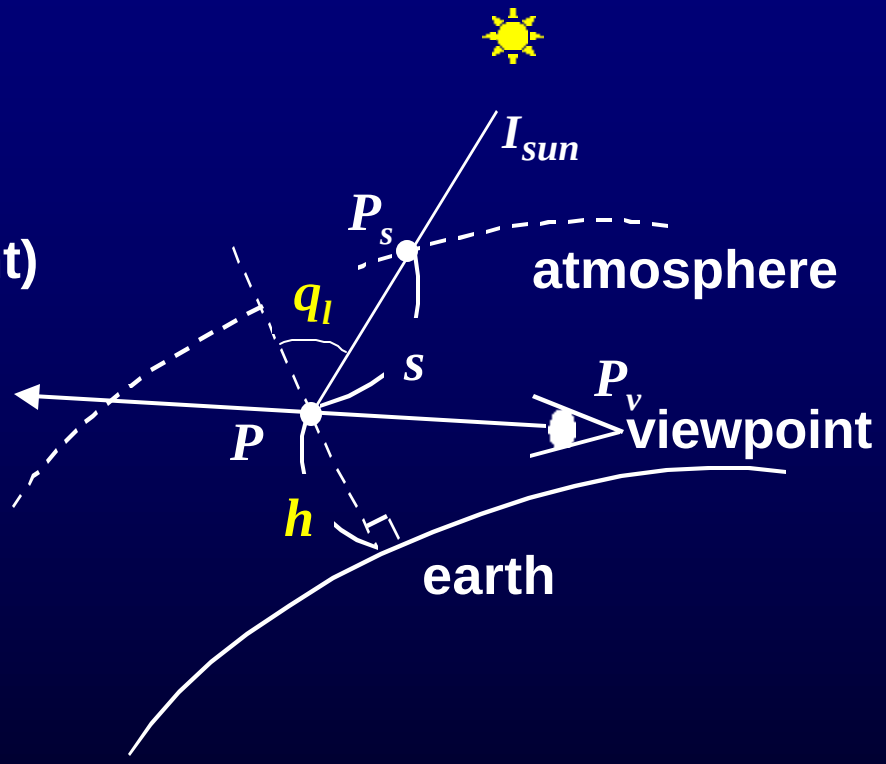
$$g_l(PP_s, l) = \exp\left(-\int_0^{\overline{PP_s}} b(l)\, r(l)\, dl\right)$$

b : extinction coefficient (constant)

r : density of particles

$r(h) = \exp(-ah)$ (a: const)

$$g_l(PP_s, l) \approx g_l(h, q_l, l)$$



Rendering Sky

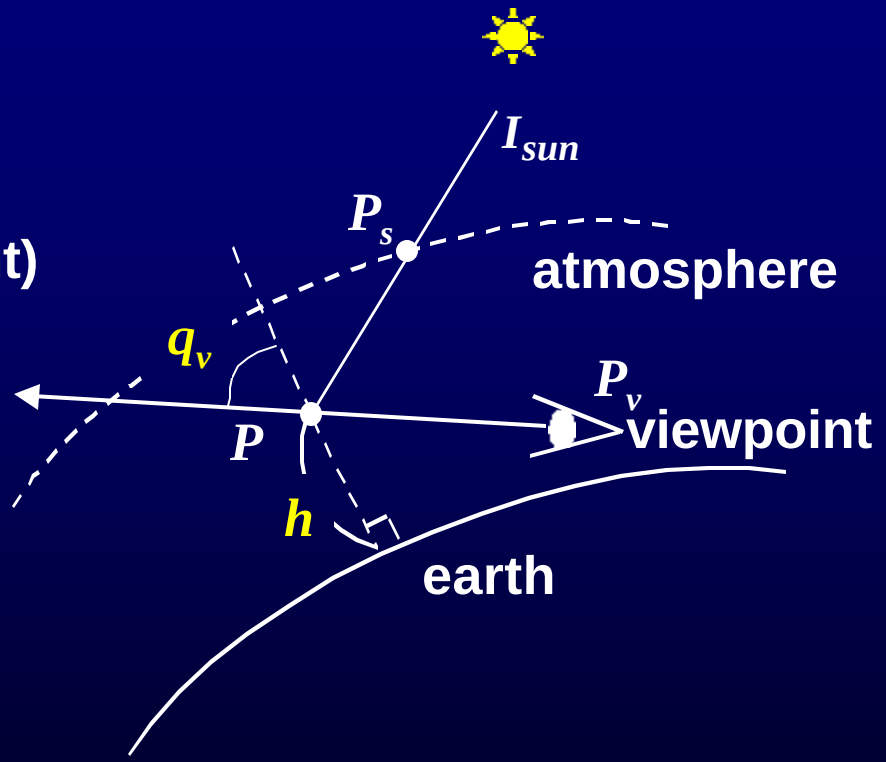
- **Textures of g_l, Δg_v, R**

$$g_l(PP_s, l) = \exp\left(-\int_0^{\overline{PP_s}} b(l)\, r(l)\, dl\right)$$

b : extinction coefficient (constant)

r : density of particles

$r(h) = \exp(-ah)$ (a: const)



$g_l(PP_s, l) \Rightarrow g_l(h, q_l, l)$

$\Delta g_v(PP_v, l) \Rightarrow \Delta g_v(h, q_v, l)$

$R(PP_v, l) \Rightarrow R(h, q_v, l)$

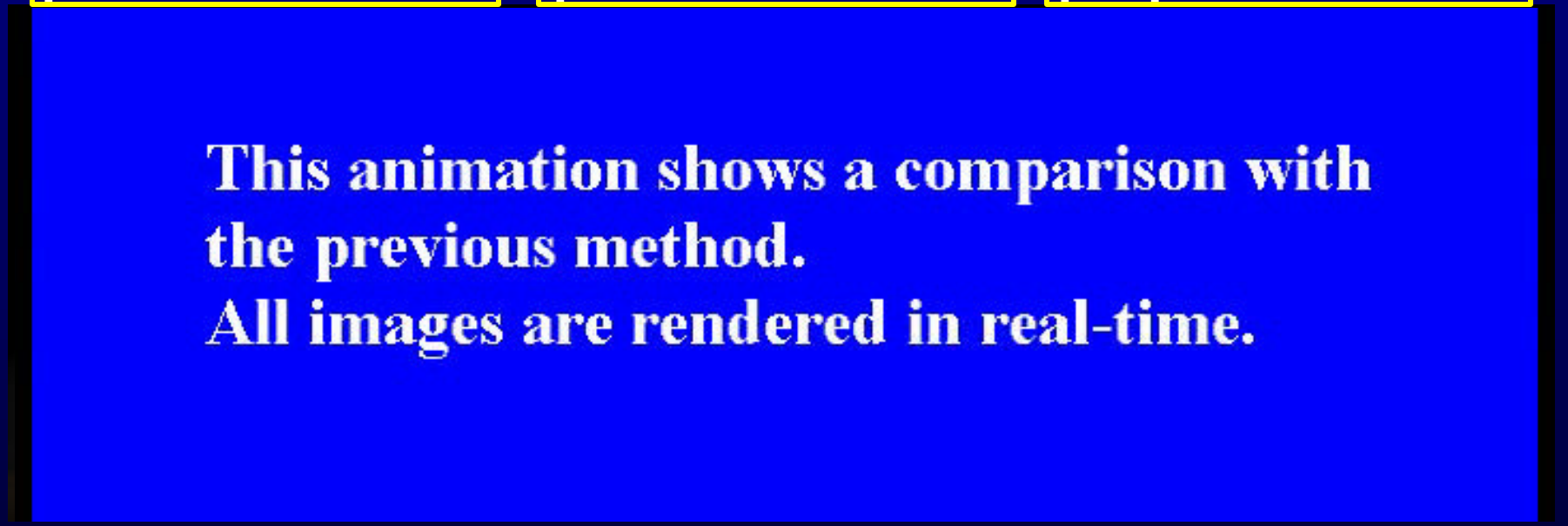
store as 2D textures

Overview

- Introduction
  - motivation
  - previous work
- Rendering Light Beams
  - basic Idea
  - problems
  - high quality rendering
- Rendering the Earth's Atmosphere
  - rendering sky
  - rendering the earth viewed from space
- **Results**
- Conclusion

Experimental Results

previous method | previous method | proposed method



#planes: 40
mesh: 60x60

#planes: 160
mesh: 60x60

#planes: 30
#sub-planes: 120 (ave.)
mesh: 10x10

computer: Athlon 1.7GHz, GeForce3

Experimental Results

method	previous	interleaved [Keller01]	proposed	ray-tracing
result				
# of planes (sub-planes)	40	160	23 (174)	-
mesh	60x90	10x15	10x15	-
time	0.13 [sec.]	0.12 [sec.]	0.08 [sec.]	29 [sec.]

Image size: 300x450 computer: Athlon 1.7GHz, GeForce3

Experimental Results

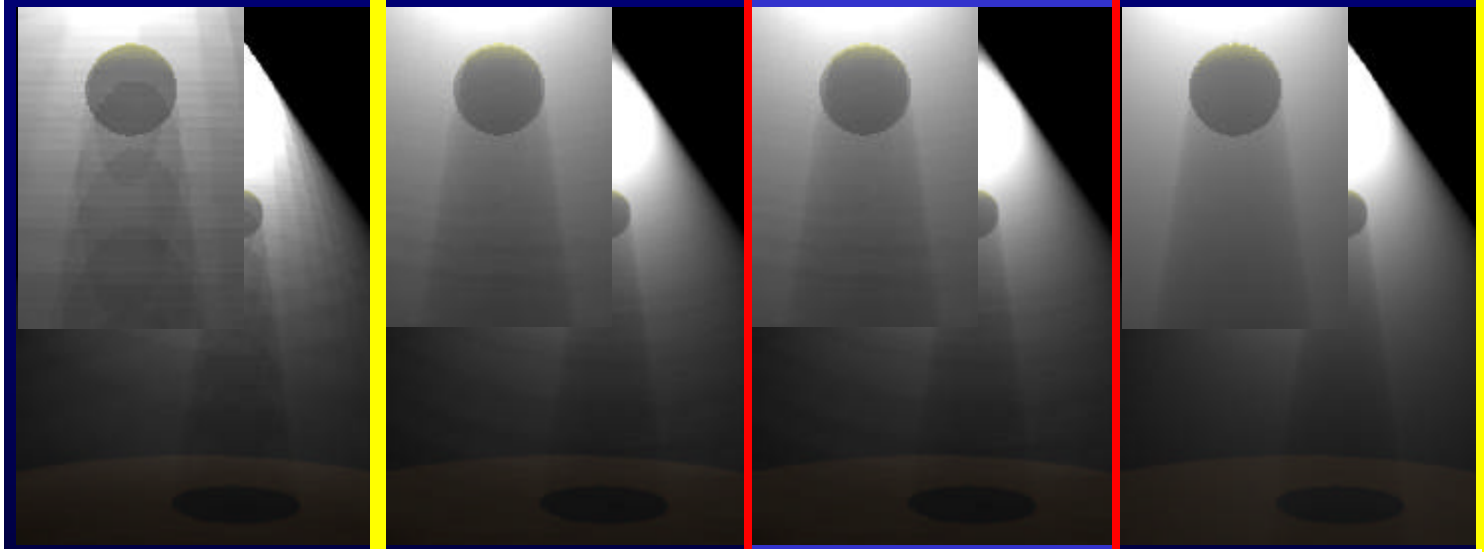
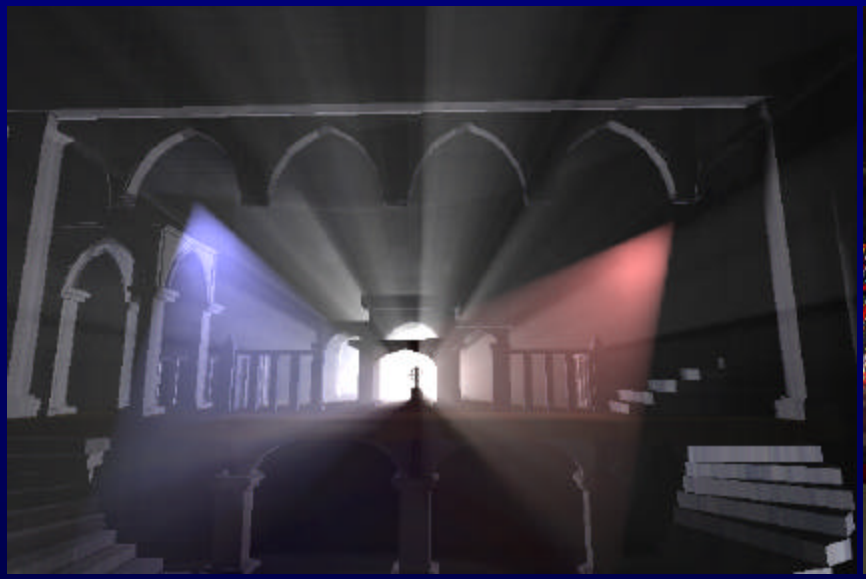
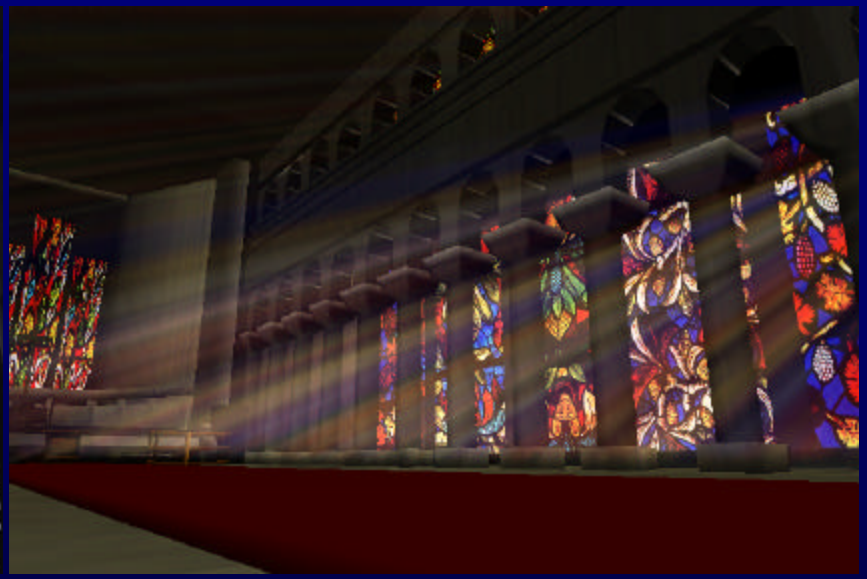
method	previous	interleaved [Keller01]	proposed	ray-tracing
result				
# of planes (sub-planes)	40	160	23 (174)	-
mesh	60x90	10x15	10x15	-
time	0.13 [sec.]	0.12 [sec.]	0.08 [sec.]	29 [sec.]

Image size: 300x450 computer: Athlon 1.7GHz, GeForce3

Results



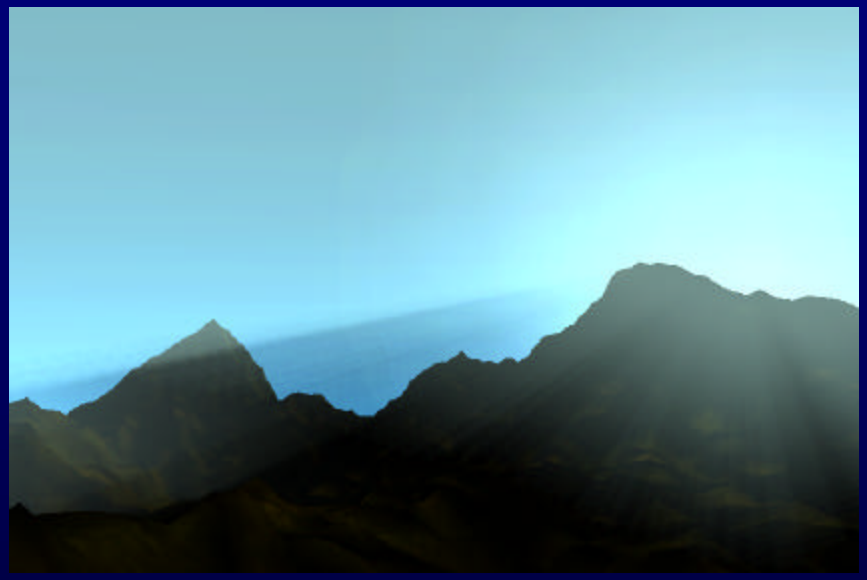
sampling plane: 9
sub-plane:309
time: 0.32 [sec.]



sampling plane: 30
sub-plane:263
time: 0.12 [sec.]

Image size: 720x480 computer: Athlon 1.7GHz, GeForce3

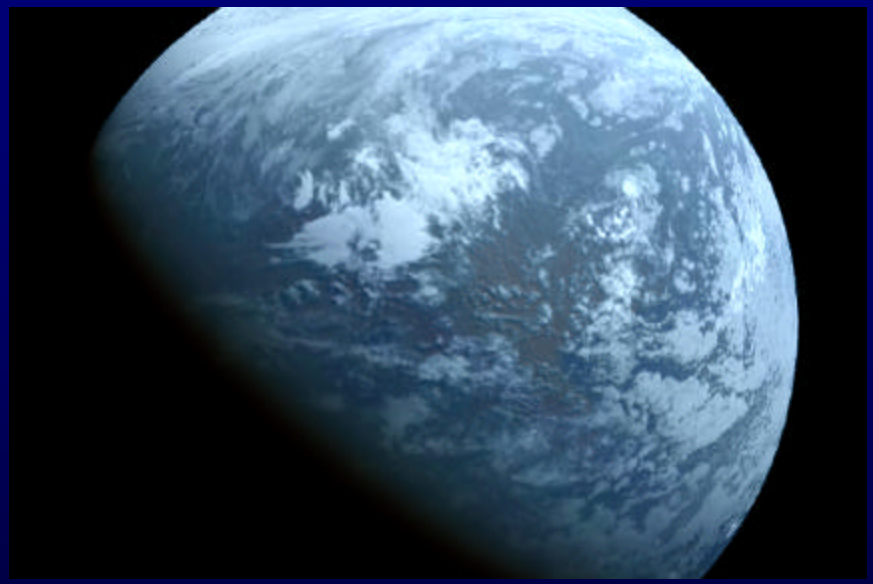
Results



sampling plane: 100(sky)
30(shaft)

sub-plane:135

time: 0.16 [sec.]



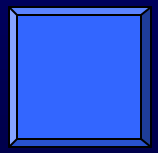
sampling sphere: 10

time: 0.06 [sec.]

Image size: 720x480 computer: Athlon 1.7GHz, GeForce3

DEMO

- **Real-time Animation Using Note PC**
  - **Pentium III (1.2 GHz)**
  - **Nvidia GeForce 2 Go**



Conclusion

- **Fast rendering of atmospheric effects**
- **Rendering of light beams**
  - 2D textures to store intensities of scattered light
  - sub-planes for precise sampling of shadows
- **Extension to rendering earth's atmosphere**
  - rendering of sky
  - rendering of earth viewed from space