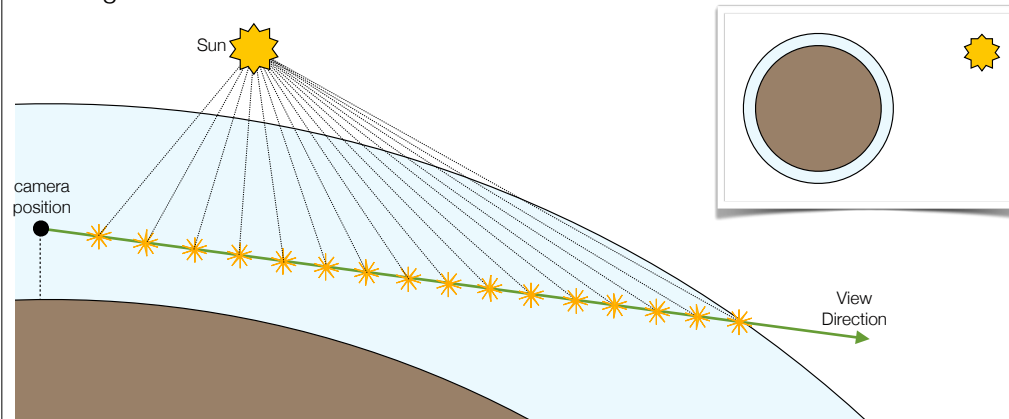


Material prepared from this chapter of GPU Gems 2:  
[http://http.developer.nvidia.com/GPUGems2/gpugems2\\_chapter16.html](http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter16.html)

# Atmospheric Model



Particles along the view ray reflect a **portion** (not 100% of it.. thus “scattering”) of the light toward the camera.



planet: earth radius + atmosphere radius

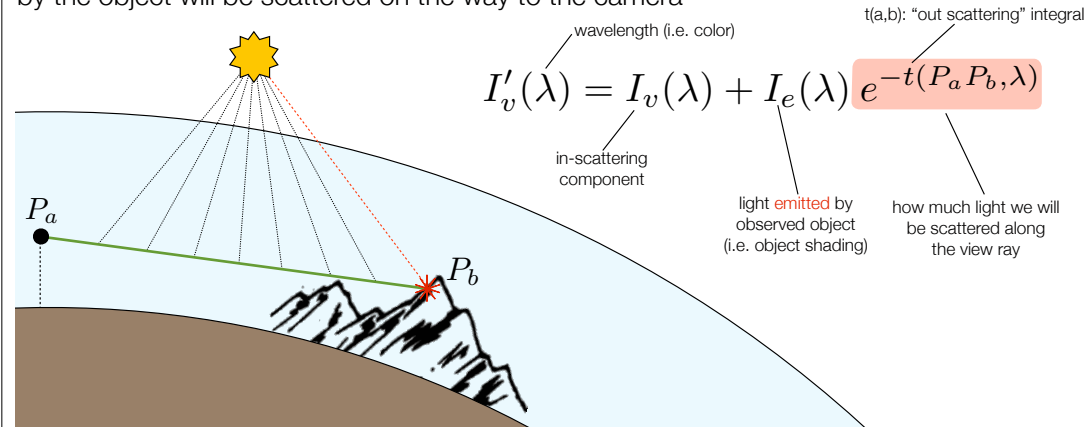
observer: height + view direction (angle)

sun: position (or direction assuming far-field approximation)

# Are you looking at an object?



we must also take into account that some of the light reflected by the object will be scattered on the way to the camera

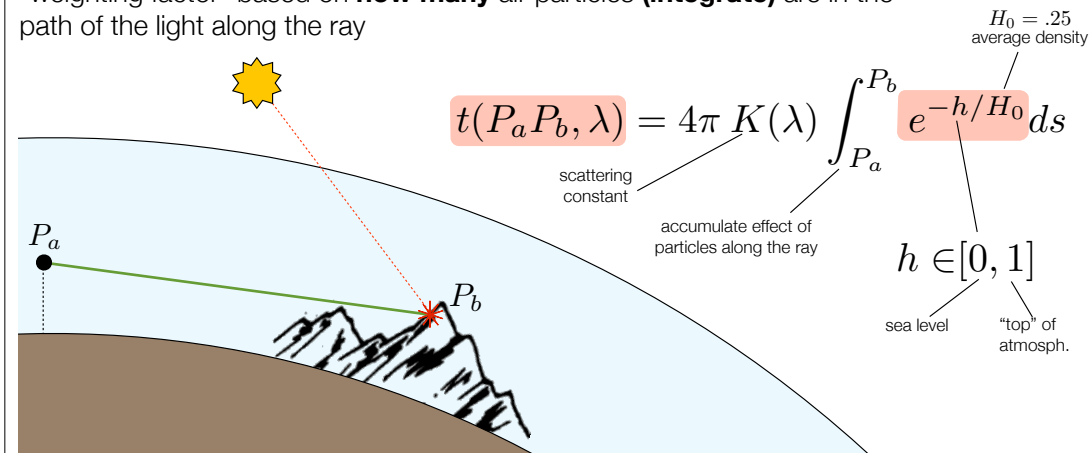


$$I'_v(\lambda) = I_v(\lambda) + I_e(\lambda) e^{-t(P_a P_b, \lambda)}$$

# “Out Scattering” Equation



“weighting factor” based on **how many** air particles (**integrate**) are in the path of the light along the ray



$$t(P_a P_b, \lambda) = 4\pi K(\lambda) \int_{P_a}^{P_b} e^{-h/H_0} ds$$

scattering constant

accumulate effect of particles along the ray

$H_0 = .25$   
average density

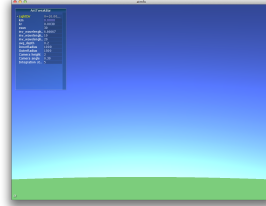
$h \in [0, 1]$   
sea level “top” of atmosph.

Essentially density of particles is assumed proportional to the height  
The scattering constant “K” is different for {Mie, Rayleigh} scattering

# Rayleigh v.s. Mie Scattering



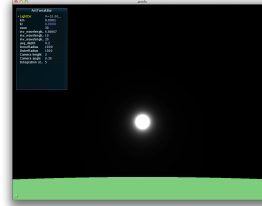
## Rayleigh Scattering



Scattering of small molecules in the air  
Scatters more short wavelengths

Effect: blue daylight sky  
Effect: red sunset


## Mie Scattering



Caused by aerosol particles (dust)  
Scatters equally across wavelengths


Effect: gray sky on rainy days  
Effect: sun halo on cloudy days

Rayleigh (blue → green → red)



ÉCOLE POLYTECHNIQUE  
FÉDÉRALE DE LAUSANNE

# "In Scattering" Equation



$$I_v(\lambda) = I_{\text{sun}}(\lambda) K(\lambda) \int_{P_a}^{P_b} F(\theta, g) e^{-h/H_0} e^{-t(PP_c, \lambda)} e^{-t(PP_a, \lambda)} ds$$

$I_v(\lambda)$   
final color

$I_{\text{sun}}(\lambda)$   
sunlight

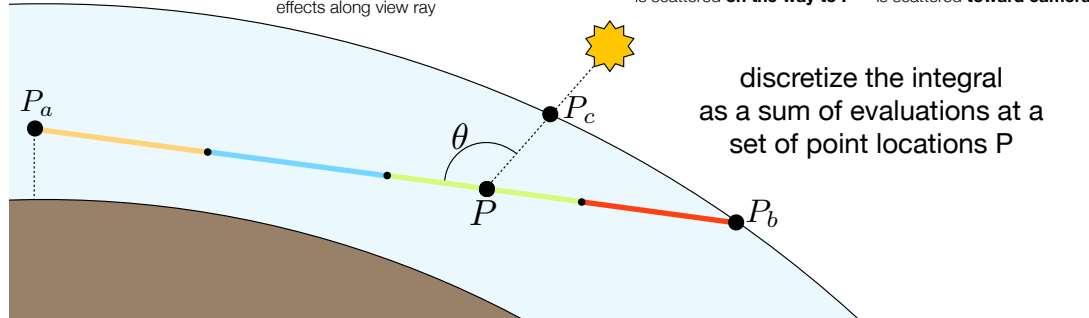
$K(\lambda)$   
integrate scattering  
effects along view ray

$F(\theta, g)$   
"phase function"  
(Mie v.s. Rayleigh)

atm\_density  
 $e^{-h/H_0}$   
how much of the **sunlight**  
is scattered **on the way to P**

$e^{-t(PP_c, \lambda)}$   
how much of the **light at P**  
is scattered **toward camera**

$e^{-t(PP_a, \lambda)}$   
how much of the **light at P**  
is scattered **toward camera**



discretize the integral  
as a sum of evaluations at a  
set of point locations P

Introduction to Computer Graphics

6

Atmospheric Scattering

The parameter "g" of the phase function changes for {Mie, Rayleigh} (see next slide)

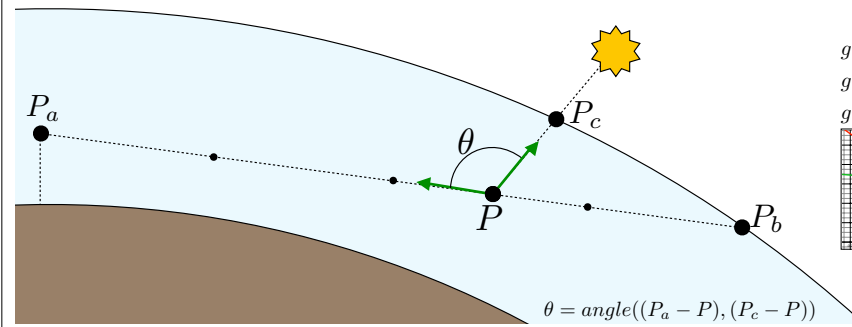
# The Phase Function “F”



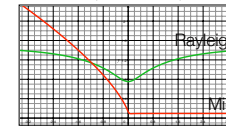
how much light is scattered in the direction of the camera (w.r.t. **angle**)

$$I_v(\lambda) = I_{\text{sun}}(\lambda) K(\lambda) \int_{P_a}^{P_b} F(\theta, g) e^{-h/H_0} e^{-t(P_c, \lambda)} e^{-t(P_a, \lambda)} ds$$

$$F(\theta, g) = \frac{3(1 - g^2)}{2(2 + g^2)} \frac{1 + \cos^2(\theta)}{(1 + g^2 - 2g \cos(\theta))^{\frac{3}{2}}}$$



$g \neq \{-1, +1\}$   
 $g = [-.99, -.75]$  (Mie)  
 $g = 0$  (Rayleigh)

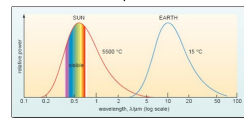




$$I_v(\lambda) = I_{\text{sun}}(\lambda) K(\lambda) \int_{P_a}^{P_b} F(\theta, g) e^{-h/H_0} e^{-t(P P_c, \lambda)} e^{-t(P P_a, \lambda)} ds$$

Assume parallel sunlight!!!

$$I_v(\lambda) = I_{\text{sun}}(\lambda) K(\lambda) F(\theta, g) e^{-t(P P_c, \lambda)} \int_{P_a}^{P_b} e^{-h/H_0} e^{-t(P P_a, \lambda)} ds$$



Assuming (1,1,1) is ok! :)

almost a constant! ;)  
(precompute and store in lookup table)

Inner integral :(

[Nishita et al. '93]

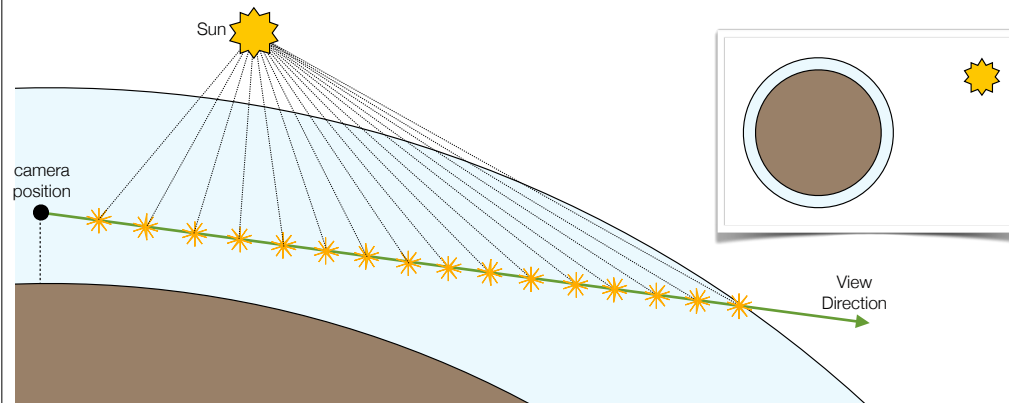
The evaluation of the outer integral is “ok” (...necessary).  
But can we do something about the inner integral?  
Otherwise the cost is 300 computations per-fragment  
(and that just discretizing integral with 5 samples!).



# Implementation task



Start implementing the “slow” version (aka what you have seen in the previous slide).  
We provide a **simple C++ framework** to get you started on this task.



You'll have to setup the camera, compute ray-sphere intersections, shade the terrain, etc...

# Using a Lookup Table

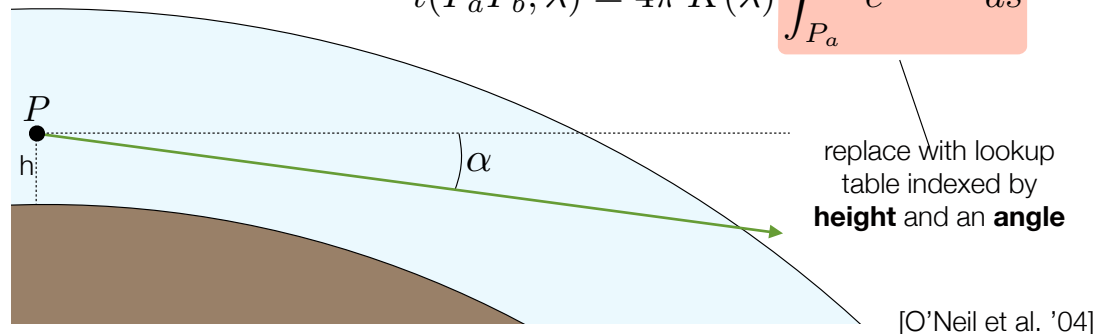


$$I_v(\lambda) = I_{\text{sun}}(\lambda) K(\lambda) F(\theta, g) e^{-t(P P_e, \lambda)} \int_{P_a}^{P_b} e^{-h/H_0} e^{-t(P P_a, \lambda)} ds$$

likely the reasons for  
which this was named "t"  
(as in lookup-table)

$$t(P_a P_b, \lambda) = 4\pi K(\lambda)$$

$$\int_{P_a}^{P_b} e^{-h/H_0} ds$$

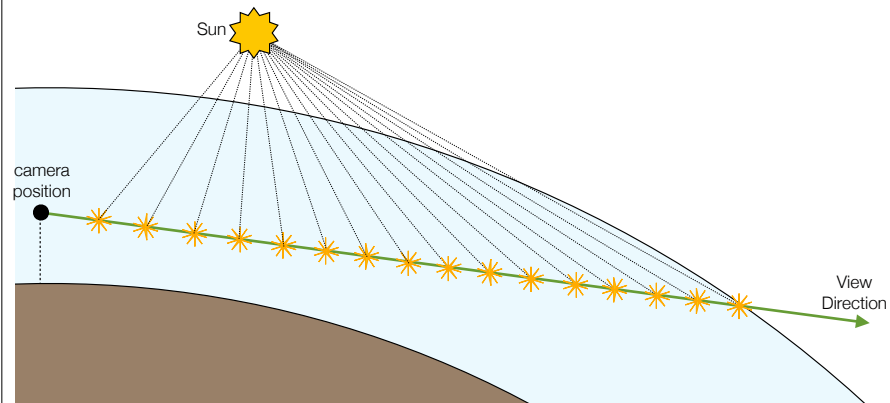


Read VERY carefully the 2nd part of Section 16.3.  
You have to make a few important tweaks to use the lookup table

# Implementation task (optional)



Perform the scattering integration using a lookup table.  
(optional because you might want to try the close form by O'Neil directly)



# Using a Close Form



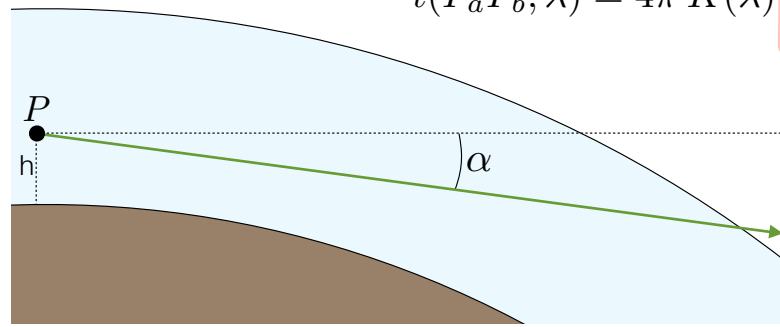
$$I_v(\lambda) = I_{\text{sun}}(\lambda) K(\lambda) F(\theta, g) e^{-t(P, \lambda)} \int_{P_a}^{P_b} e^{-h/H_0} e^{-t(P, \lambda)} ds$$

likely the reasons for  
which this was named "t"  
(as in lookup-table)

$$t(P_a P_b, \lambda) = 4\pi K(\lambda)$$

$$\int_{P_a}^{P_b} e^{-h/H_0} ds$$

replace with close  
form expressions



[O'Neil et al. '04]

Read VERY carefully section 16.4.1 of the tutorial

