GameDevelopers Conference

Rendering Outdoor Light Scattering in Real Time

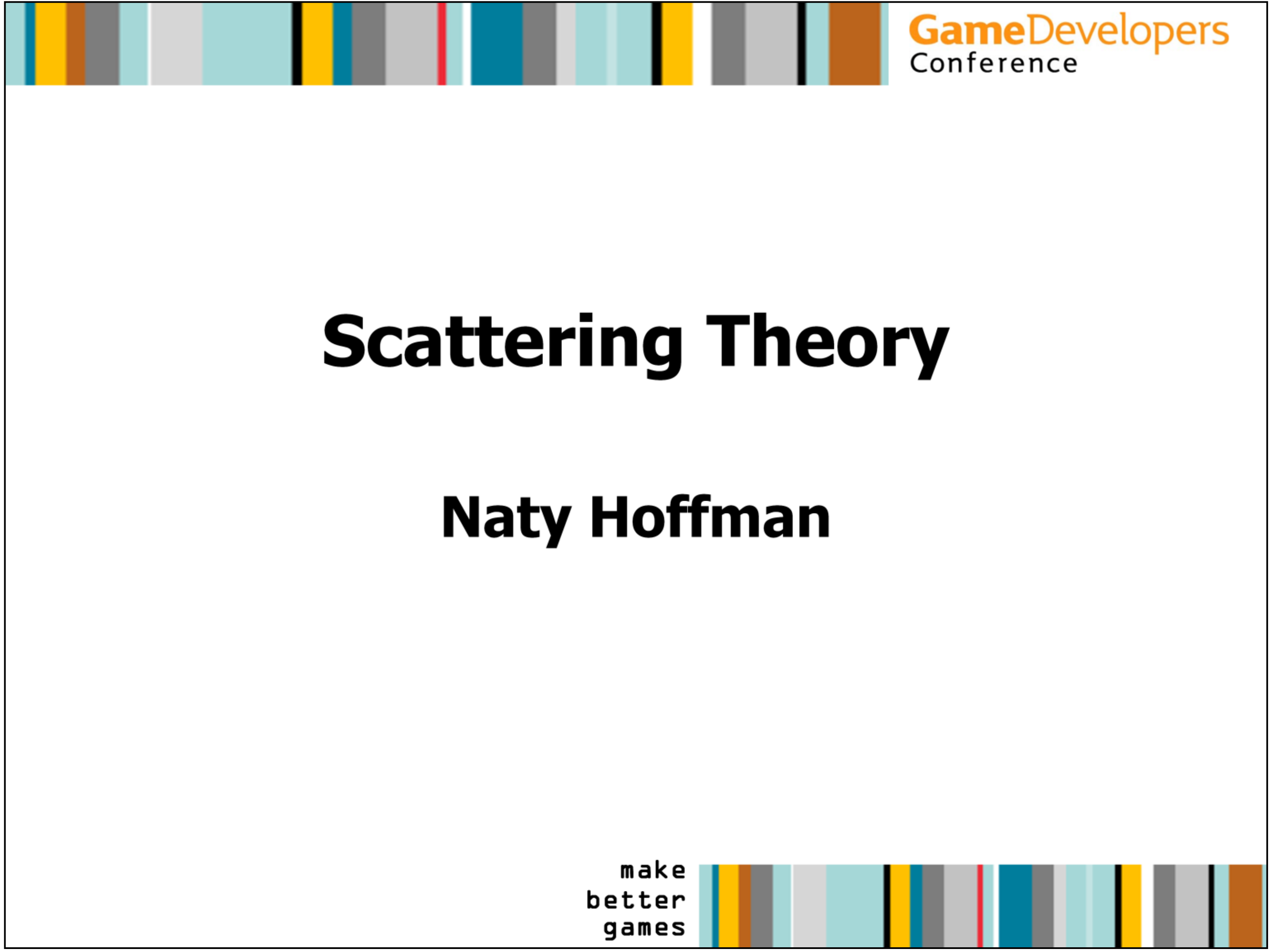
Naty Hoffman
Westwood Studios
naty@westwood.com

Arcot J Preetham
ATI Research
preetham@ati.com

make
better
games

Outline

- Basics (Naty)
  - Atmospheric Light Scattering
  - Radiometric Quantities
  - From Radiance to Pixels
- Scattering Theory (Naty)
  - Absorption, Out-Scattering, In-Scattering
  - Rayleigh and Mie Scattering
- Implementation (Preetham)
  - Aerial Perspective, Sunlight, Skylight
  - Vertex Shader
- Future Work (Preetham)

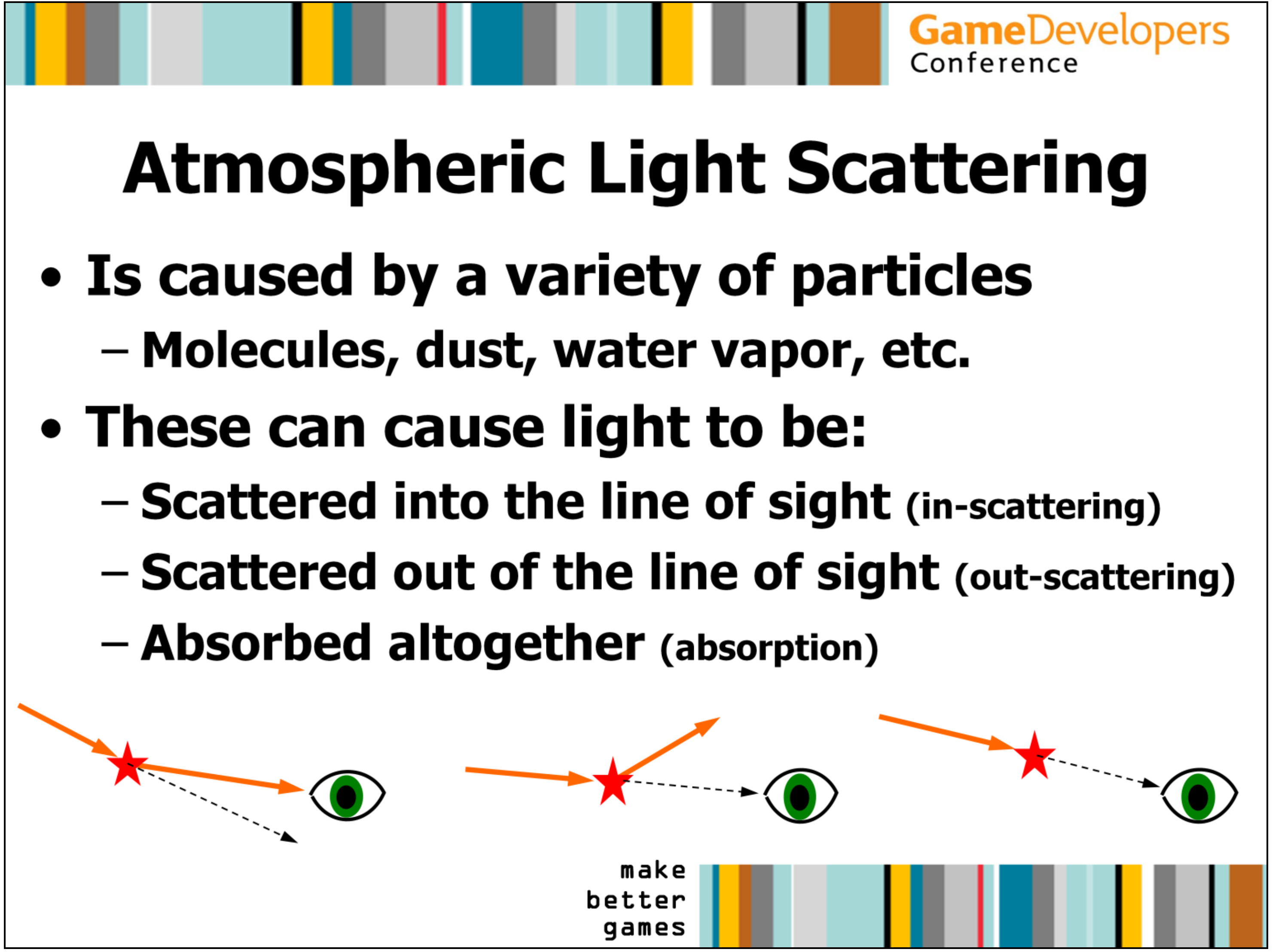


GameDevelopers Conference

Scattering Theory

Naty Hoffman

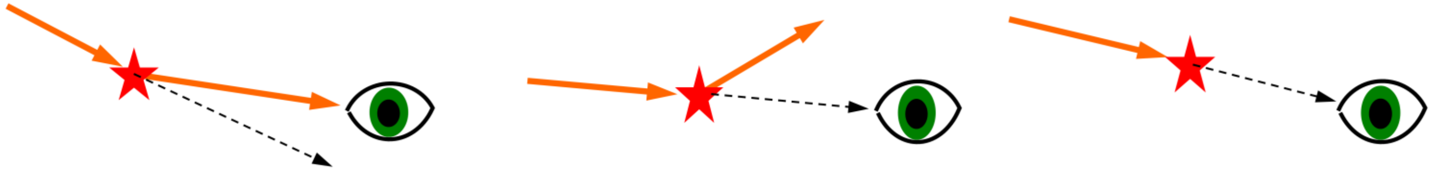
make better games



GameDevelopers Conference

Atmospheric Light Scattering

- Is caused by a variety of particles
  - Molecules, dust, water vapor, etc.
- These can cause light to be:
  - Scattered into the line of sight (in-scattering)
  - Scattered out of the line of sight (out-scattering)
  - Absorbed altogether (absorption)

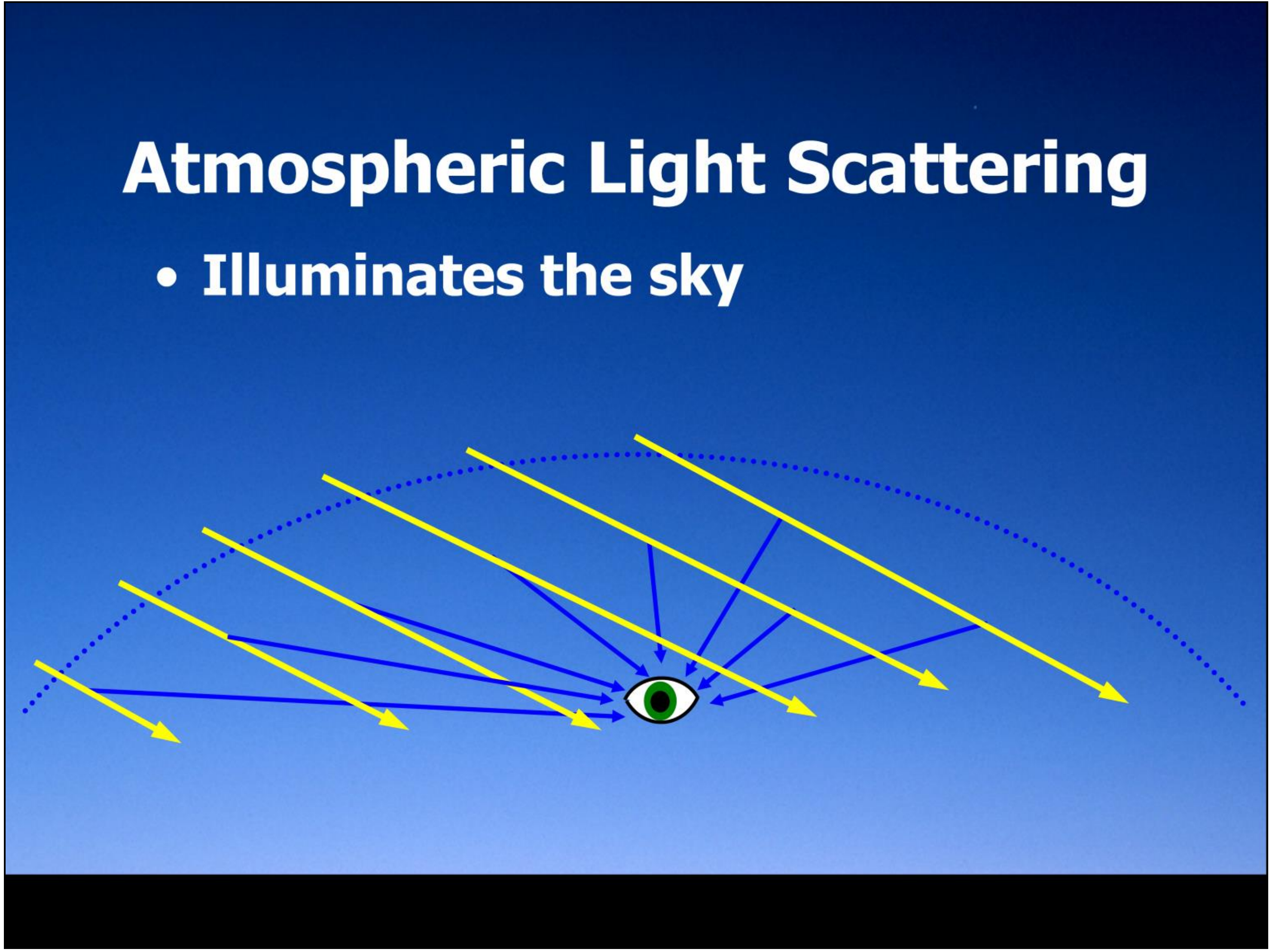


make better games

There are three main phenomena relating to atmospheric light scattering: in-scattering, out-scattering and absorption.

(the word "scattering" is not 100% correct as a group name for all three but it makes for convenient shorthand).

The scattering /absorption events are marked by red stars.

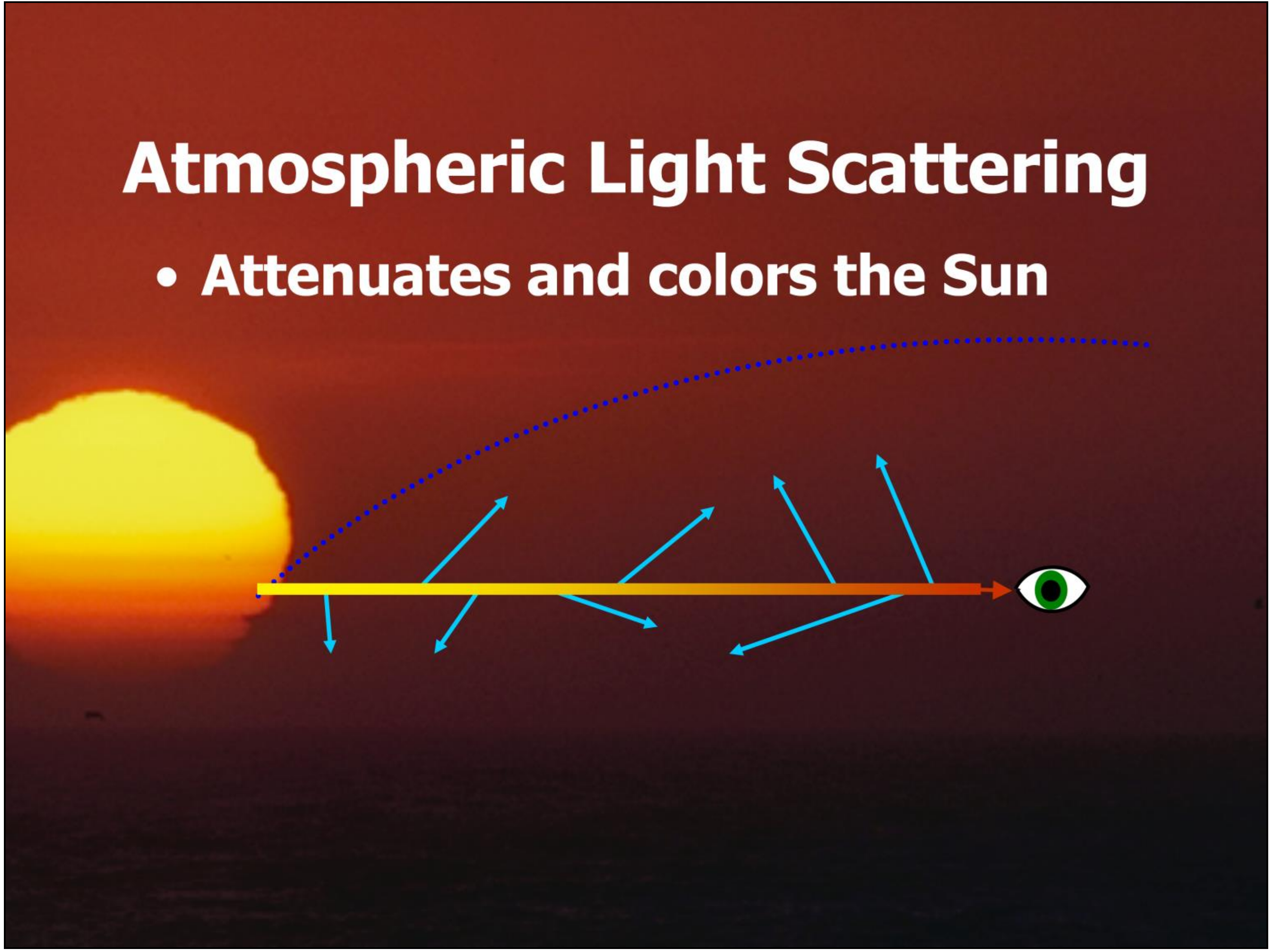


Atmospheric Light Scattering

- Illuminates the sky

The dotted blue curve is the edge of the atmosphere, the yellow arrows are sunlight.

In-scattering causes part of the sunlight to be scattered towards the eye from all directions. Note that blue light is preferentially scattered – we will expand upon this at a later point.

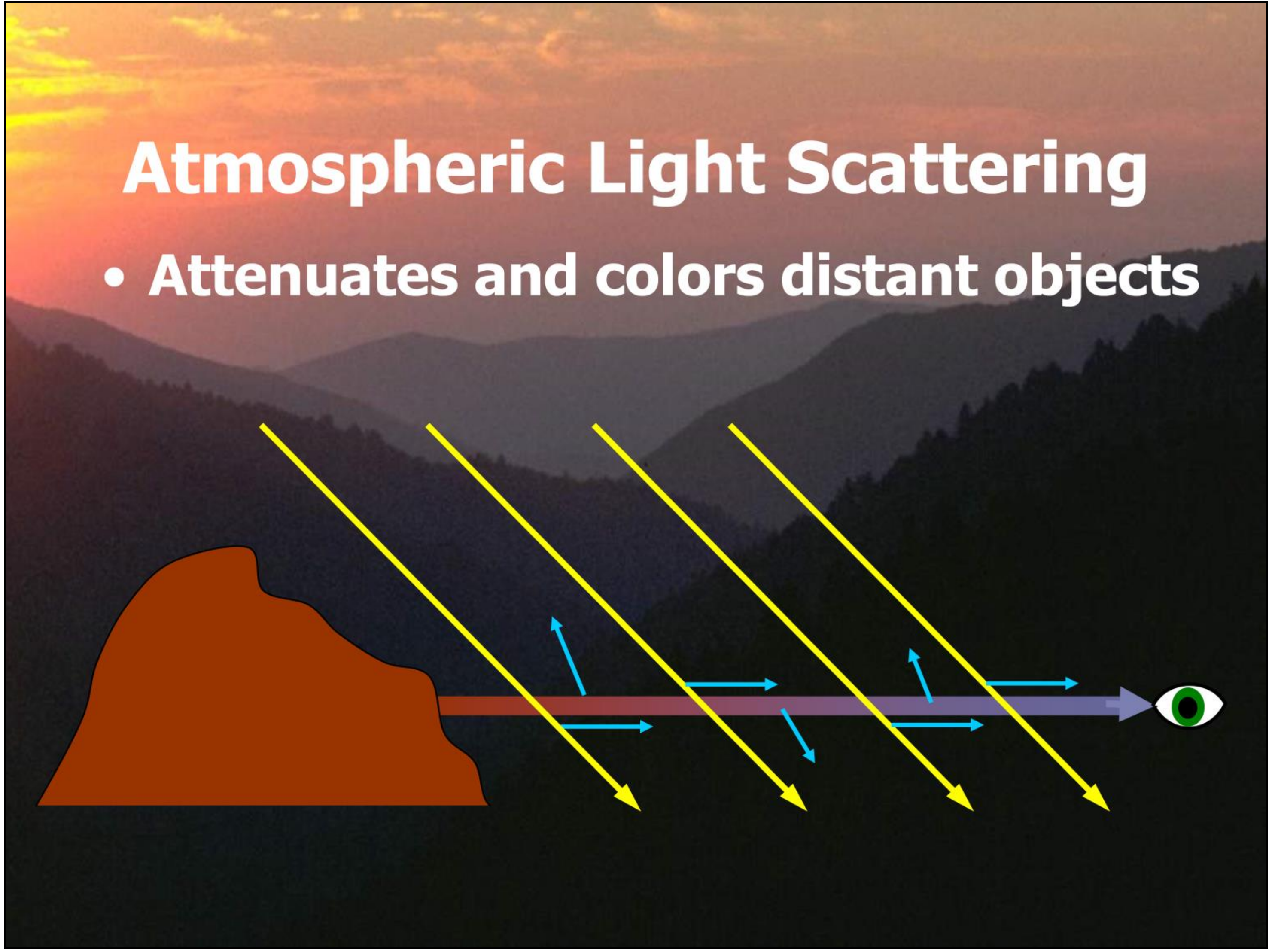


Atmospheric Light Scattering

- Attenuates and colors the Sun

Out-scattering and absorption causes part of the light from the sun to be lost before it reaches the ground. Again mostly blue light is lost, which causes the sun's color to shift towards yellow / red.

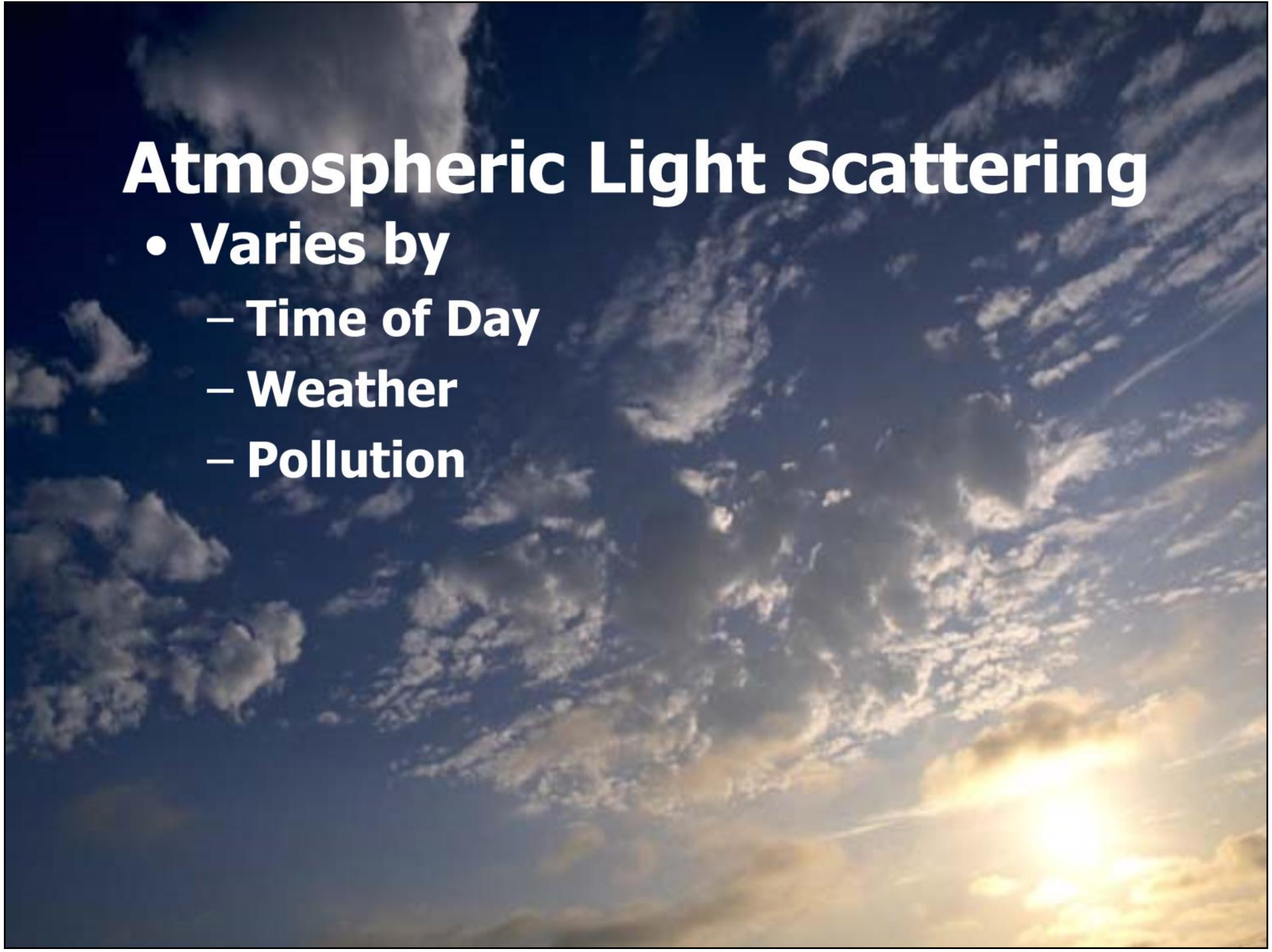
When the sun is near the horizon its light travels a much farther distance through the atmosphere then when it is at the zenith, so there is much more scattering and the sun's light is weaker and redder.



Atmospheric Light Scattering

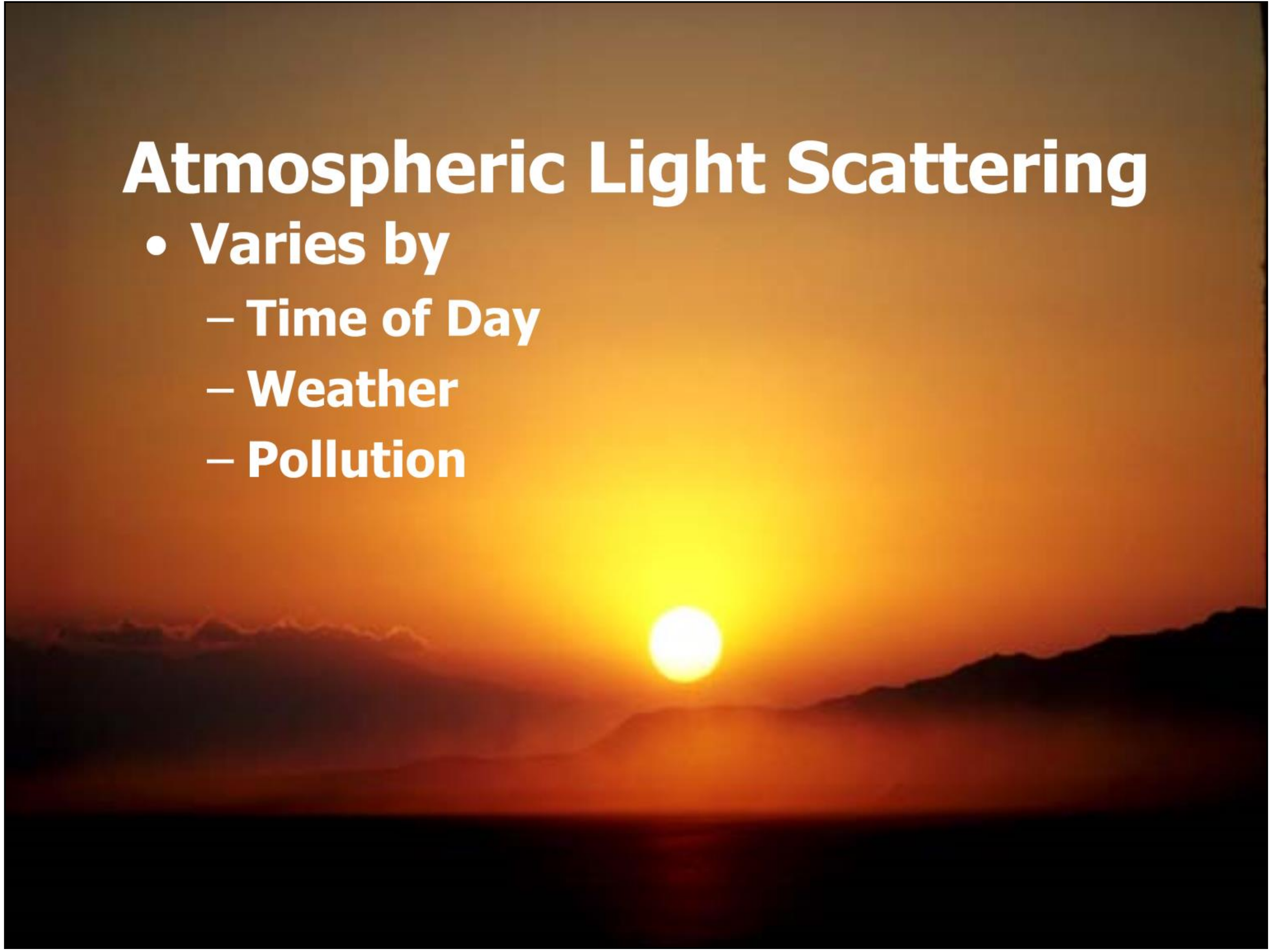
- Attenuates and colors distant objects

Light from distant objects is both attenuated by out-scattering / absorption and added to by in-scattering.



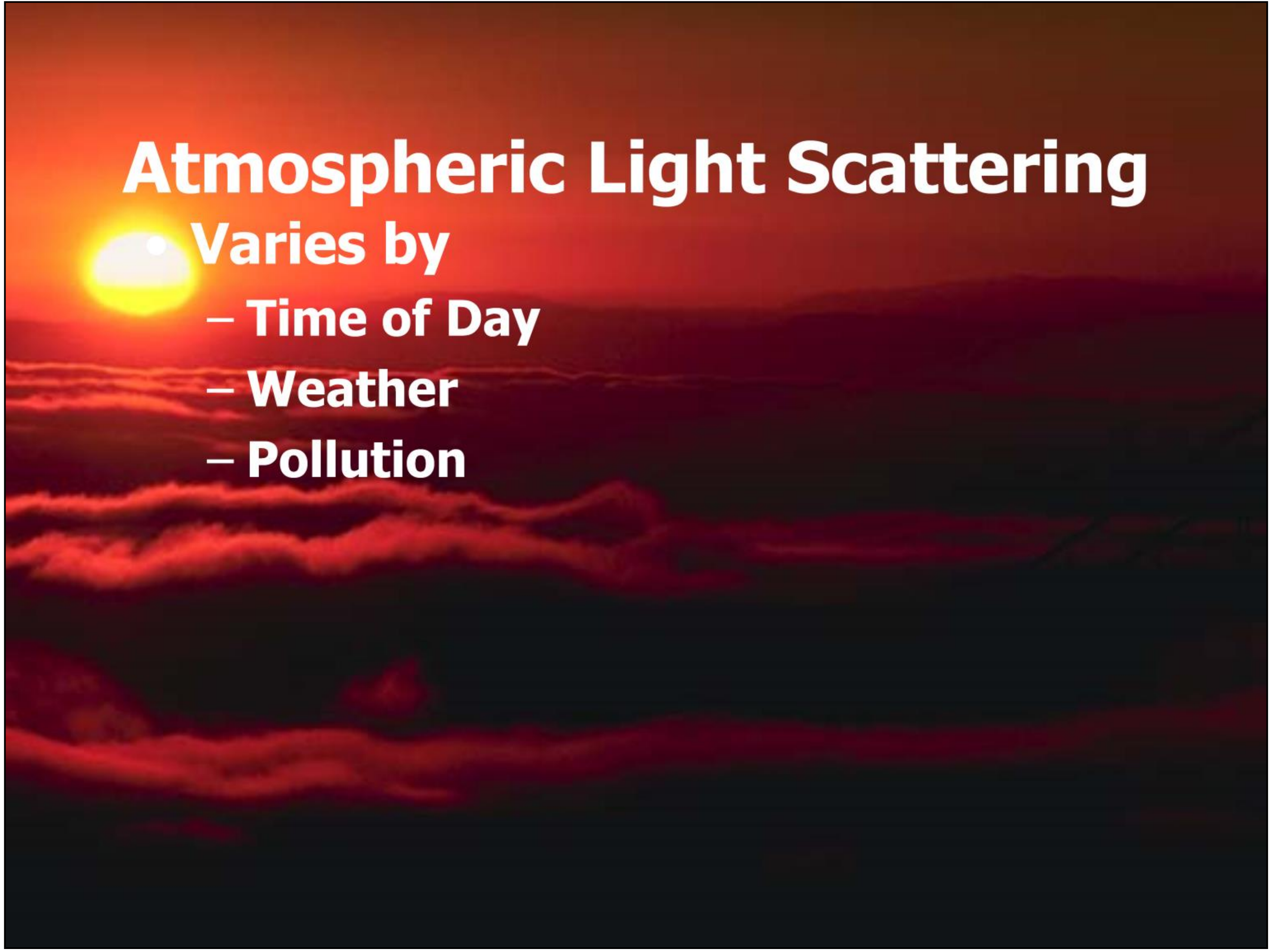
Atmospheric Light Scattering

- Varies by
  - Time of Day
  - Weather
  - Pollution



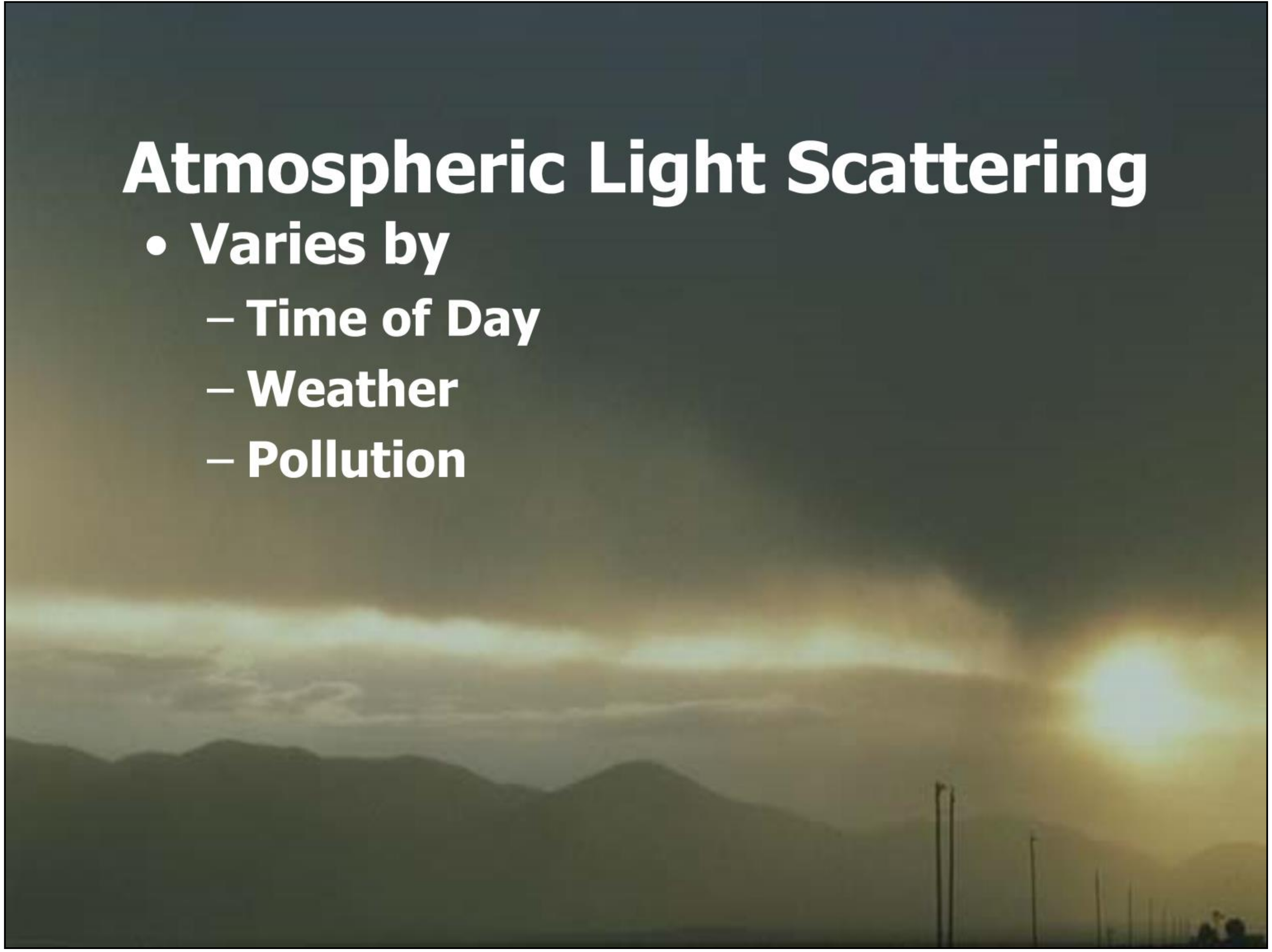
Atmospheric Light Scattering

- Varies by
  - Time of Day
  - Weather
  - Pollution



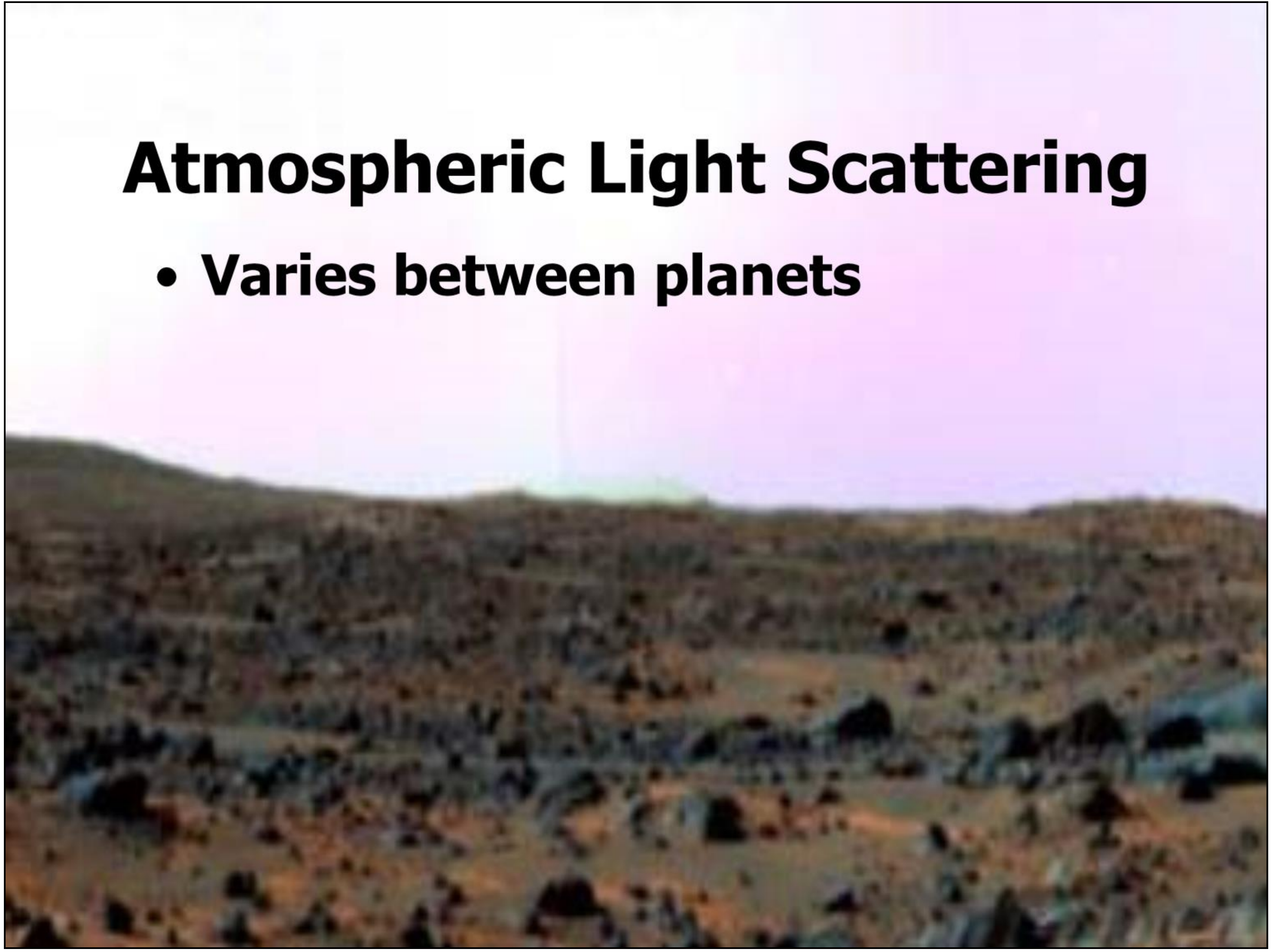
Atmospheric Light Scattering

- Varies by
  - Time of Day
  - Weather
  - Pollution



Atmospheric Light Scattering

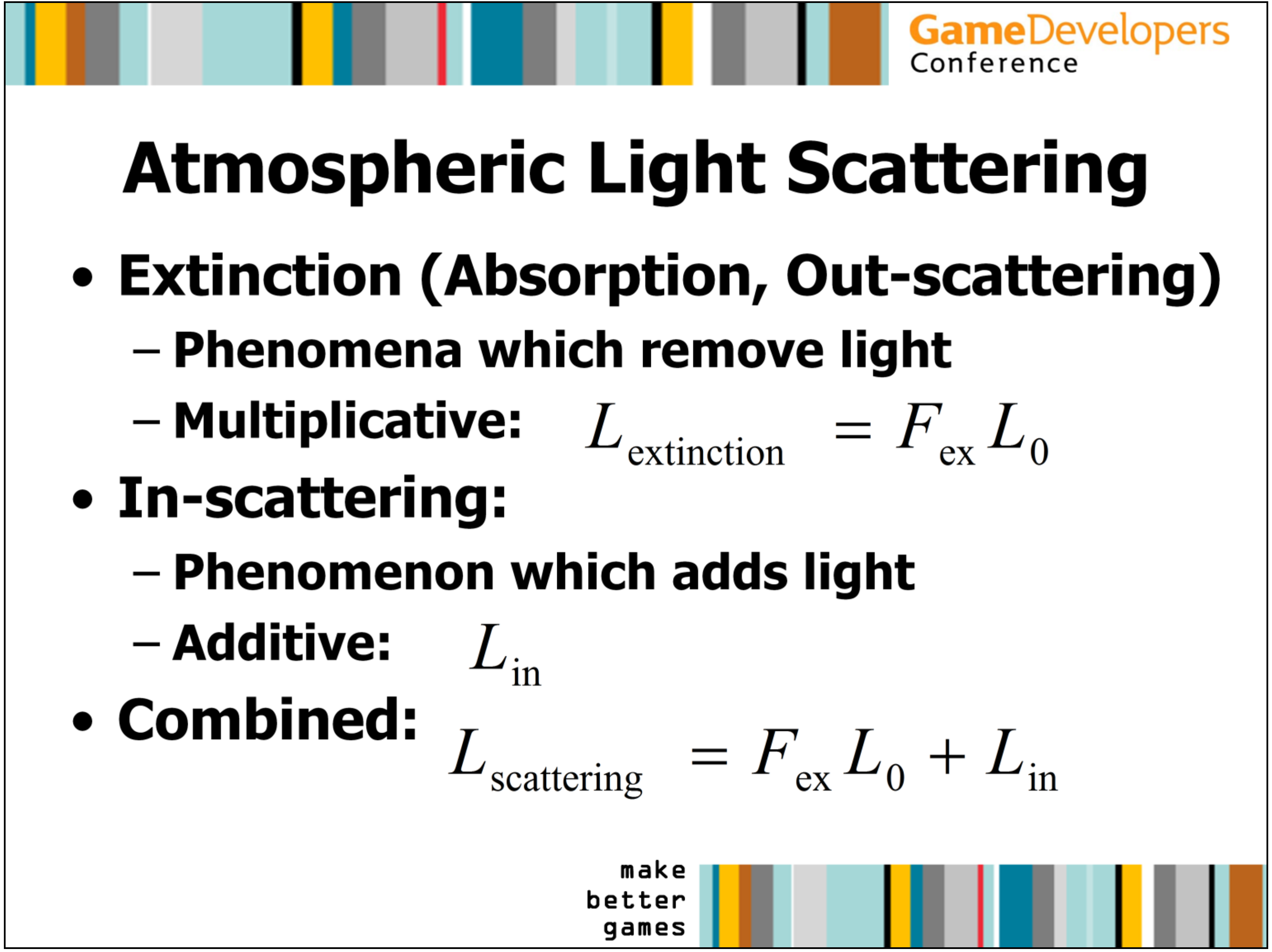
- Varies by
  - Time of Day
  - Weather
  - Pollution



Atmospheric Light Scattering

- Varies between planets

This is an image from the Mars Pathfinder mission.



Game Developers Conference

Atmospheric Light Scattering

- Extinction (Absorption, Out-scattering)
  - Phenomena which remove light
  - Multiplicative: $L_{\text{extinction}} = F_{\text{ex}} L_0$
- In-scattering:
  - Phenomenon which adds light
  - Additive: L_{in}
- Combined: $L_{\text{scattering}} = F_{\text{ex}} L_0 + L_{\text{in}}$

make better games

There are two types of phenomena which occur when light interacts with the atmosphere. Extinction phenomena (absorption, out-scattering) reduce the intensity of the light, and do so in a <u>multiplicative</u> fashion. In-scattering <u>adds</u> to the light intensity (another additive phenomenon—emission—is rare in the atmosphere and will not be discussed in this lecture). All these phenomena can be summarized as two factors: one which is multiplied with the light intensity (or color), and one which is added to it. This lecture will show you how to compute these factors and apply them to complex outdoor scenes – in real-time.

This is a screenshot from our real-time demo.

GameDevelopers Conference

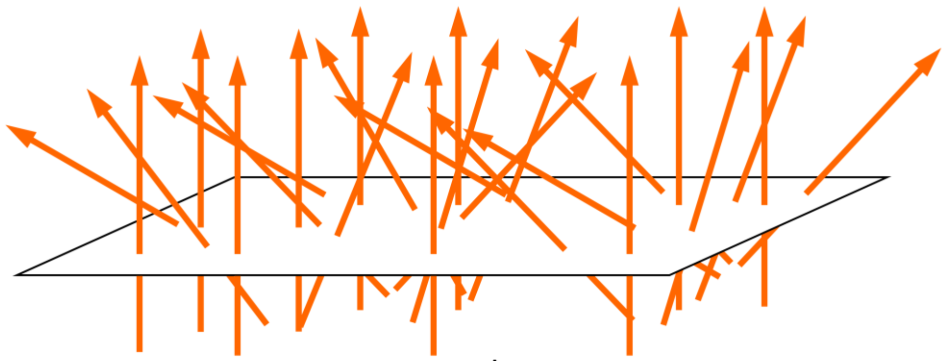
Radiometric Quantities

- Radiant Flux
- Radiance
- Irradiance

make
better
games

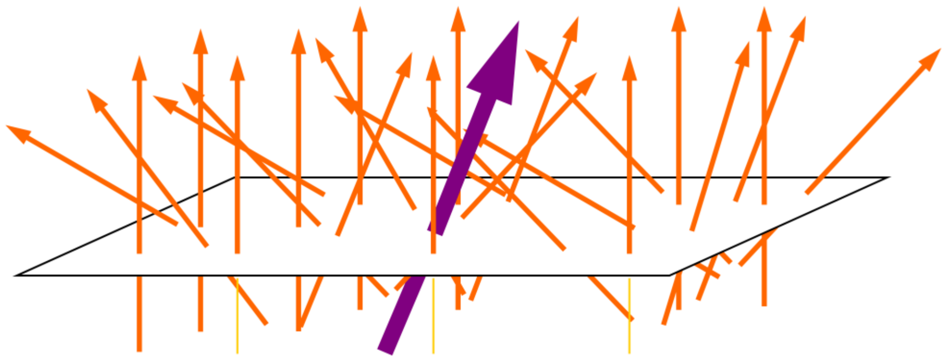
Radiometric Quantities

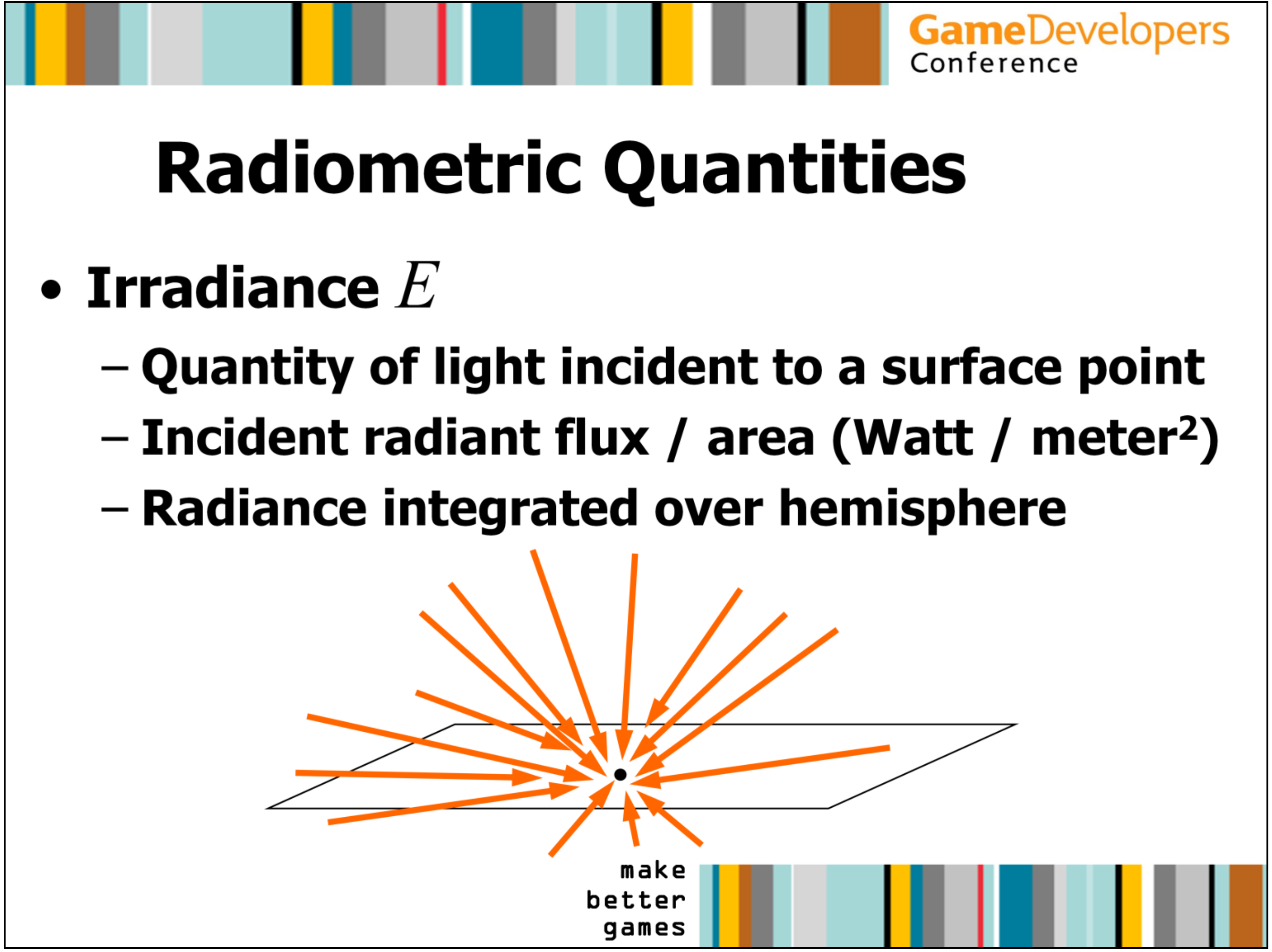
- Radiant Flux Φ
  - Quantity of light through a surface
  - Radiant power (energy / time)
  - Watt



Radiometric Quantities

- Radiance L
  - Quantity of light in a single ray
  - Radiant flux / area / solid angle
  - Watt / (meter^2 * steradian)

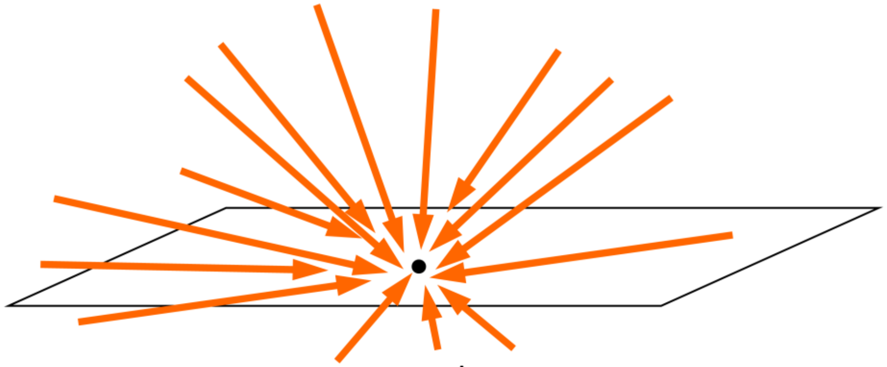




GameDevelopers Conference

Radiometric Quantities

- Irradiance E
  - Quantity of light incident to a surface point
  - Incident radiant flux / area (Watt / meter^2)
  - Radiance integrated over hemisphere

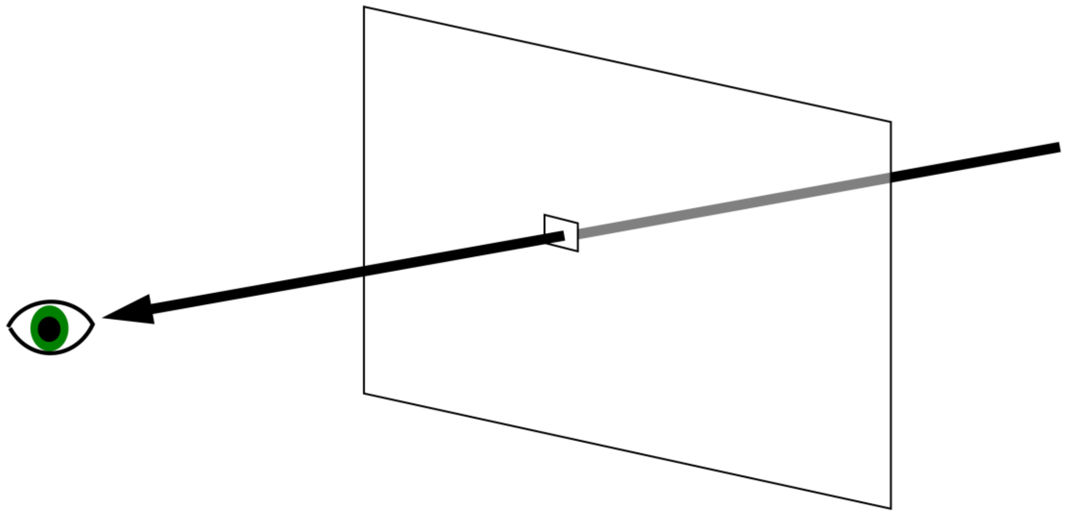


make better games

Irradiance can be viewed as radiance projected to a surface and multiplied by or integrated over a solid angle. Irradiance is needed to illuminate something – a single ray of finite radiance alone will not light a surface. Point lights are nonphysical – the math only works out if they pack a nonzero irradiance into a single ray (zero solid angle), which requires infinite radiance. For example, the sun illuminates due to its irradiance contribution, which is proportional to the product of its (very large) radiance and its (very small) solid angle.

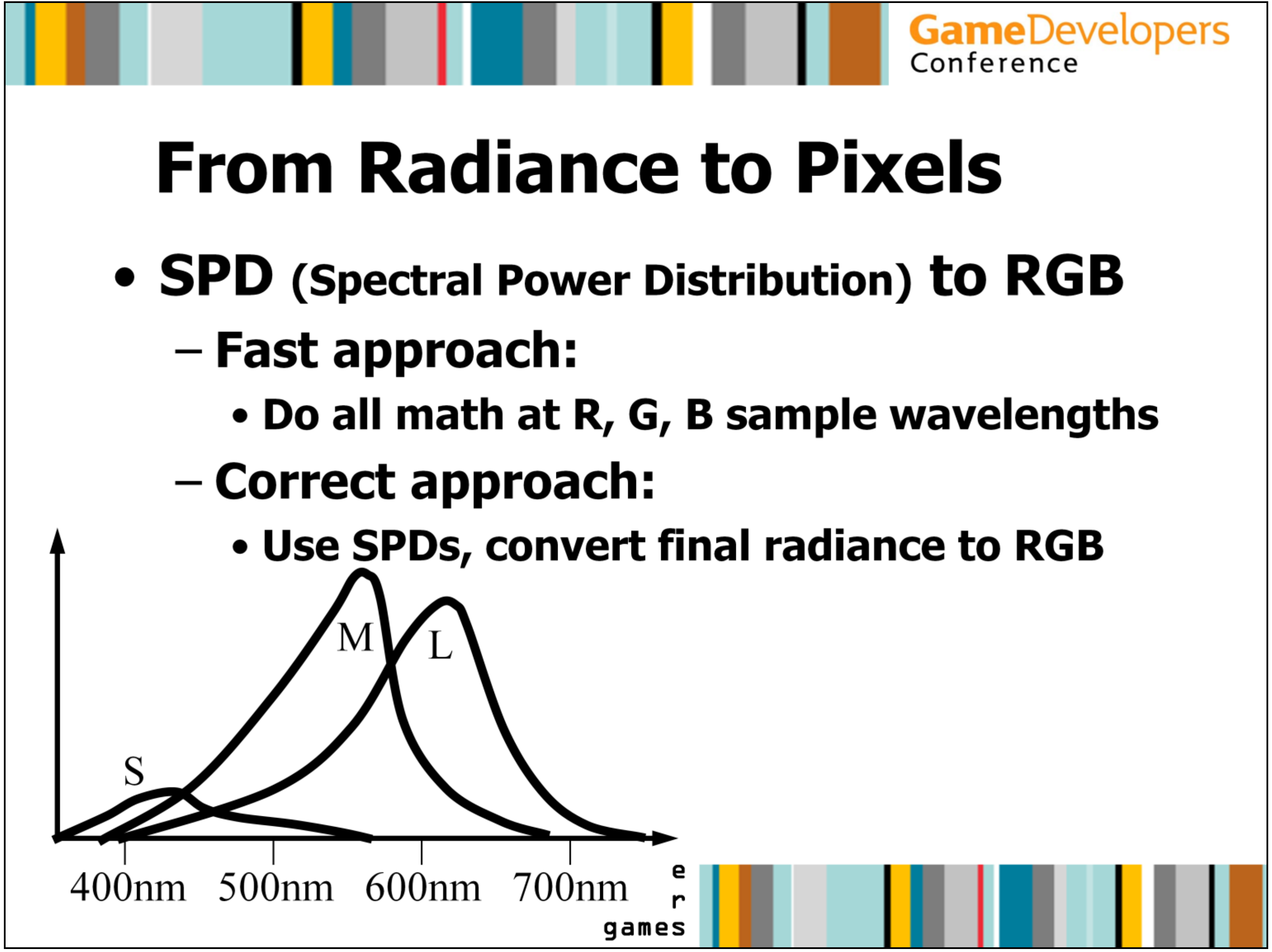
From Radiance to Pixels

- Compute radiance incident to eye through each screen pixel



From Radiance to Pixels

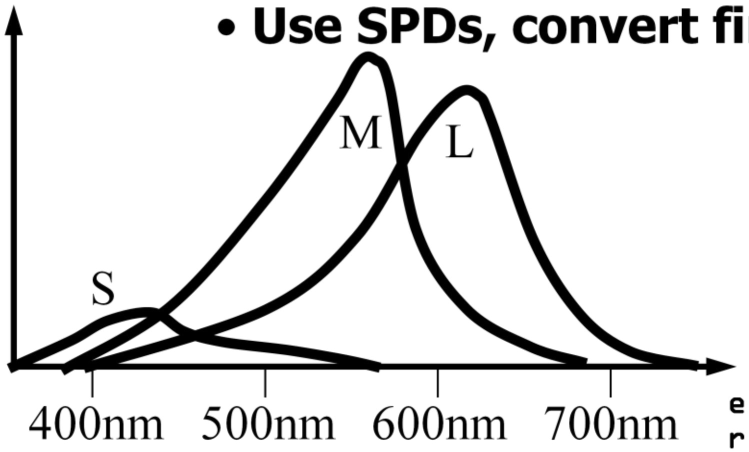
- Pixel value based on radiance
- But radiance is distributed continuously along the spectrum
  - We need three numbers: R, G, B



GameDevelopers Conference

From Radiance to Pixels

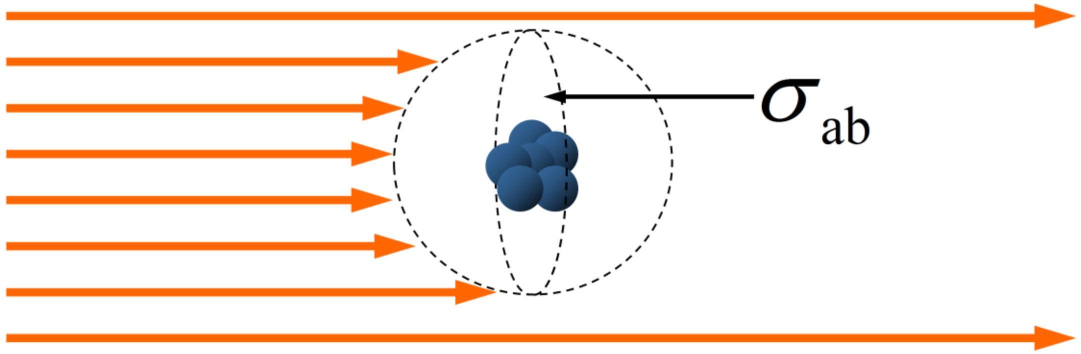
- **SPD** (Spectral Power Distribution) **to RGB**
  - Fast approach:
    - Do all math at R, G, B sample wavelengths
  - Correct approach:
    - Use SPDs, convert final radiance to RGB



When computing with SPDs, typically a few dozen samples along the visible part of the spectrum are used. The final conversion to RGB is done by multiplying the SPD by perceptual weighting functions and integrating. Using three sample frequencies is much faster and amenable to hardware implementation. However, it loses a lot of information—all the spectral structure between the samples. However to some degree this problem exists throughout real-time graphics. We will use the fast approach.

Absorption

- Absorption cross section σ_{ab}
  - Absorbed radiant flux per unit incident irradiance Φ / E
  - Units of area (meter²)

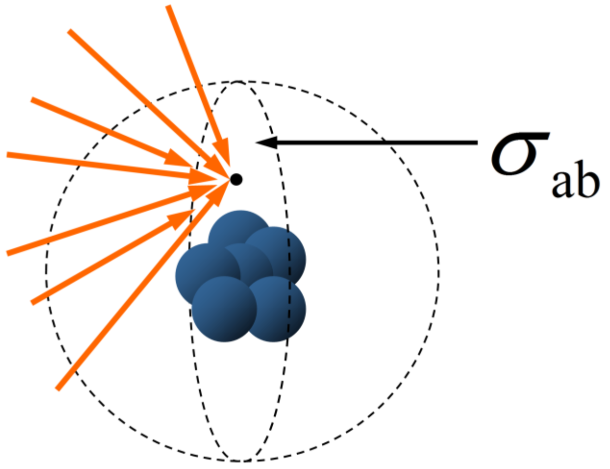


The efficiency with which a single particle absorbs light in its vicinity can be quantified by its absorption cross section, defined as absorbed radiant flux (Watts) per unit incident irradiance (Watts over meters squared). So we see that its units are meters squared, in other words area, which matches the “cross section” name. To understand this quantity a bit more intuitively, let us assume (for simplification) that its absorption is characterized by absorbing all photons which enter a sphere around the particle. We can look at a cross section of this sphere as a surface area which absorbs the incident irradiance at each point.

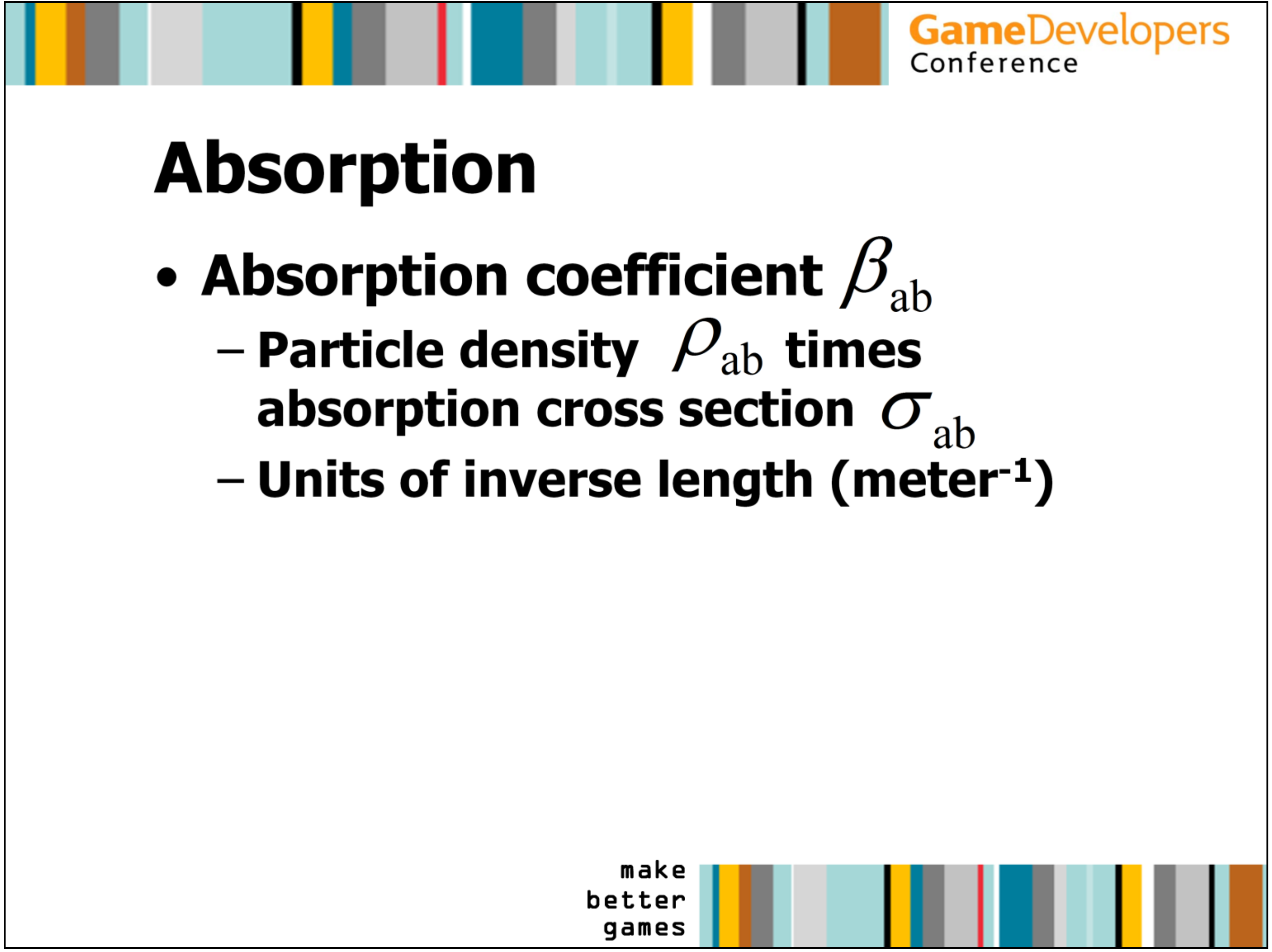
Absorption

- Absorption cross section σ_{ab}

$$\Phi = E\sigma_{ab} \Longrightarrow \sigma_{ab} = \Phi / E$$



If we look at the absorbed irradiance at a single point it becomes clear that the total flux absorbed by the cross-section is incident irradiance times the cross-section area. This quickly yields the original absorption cross-section definition. Note that a particle can have different absorption cross-sections for photons of different wavelengths.



Absorption

- **Absorption coefficient** β_{ab}
  - **Particle density** ρ_{ab} **times absorption cross section** σ_{ab}
  - **Units of inverse length (meter^{-1})**

As opposed to a single particle, we characterize the absorptive efficiency of an absorbing medium by its absorption coefficient. This is defined as particle density (inverse cubic meters) times the absorption cross section (square meters). So we see that its units are inverse length.

GameDevelopers Conference

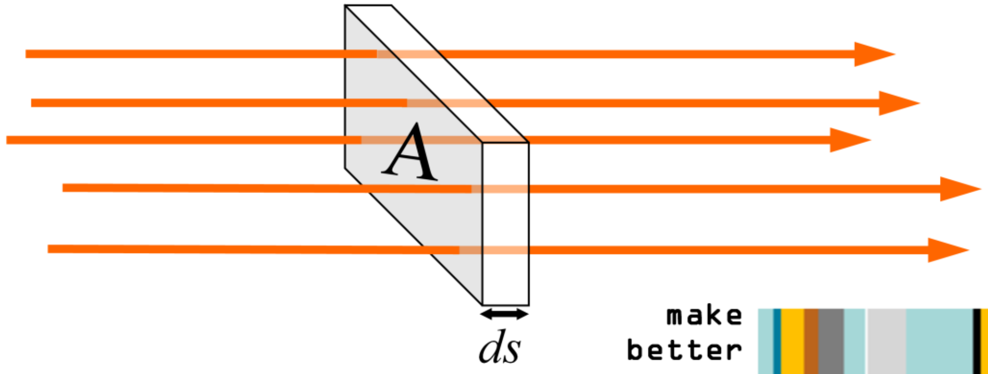
Absorption

- **Total absorption cross section:**

$$A_{ab} = \sigma_{ab} \rho_{ab} A \, ds$$

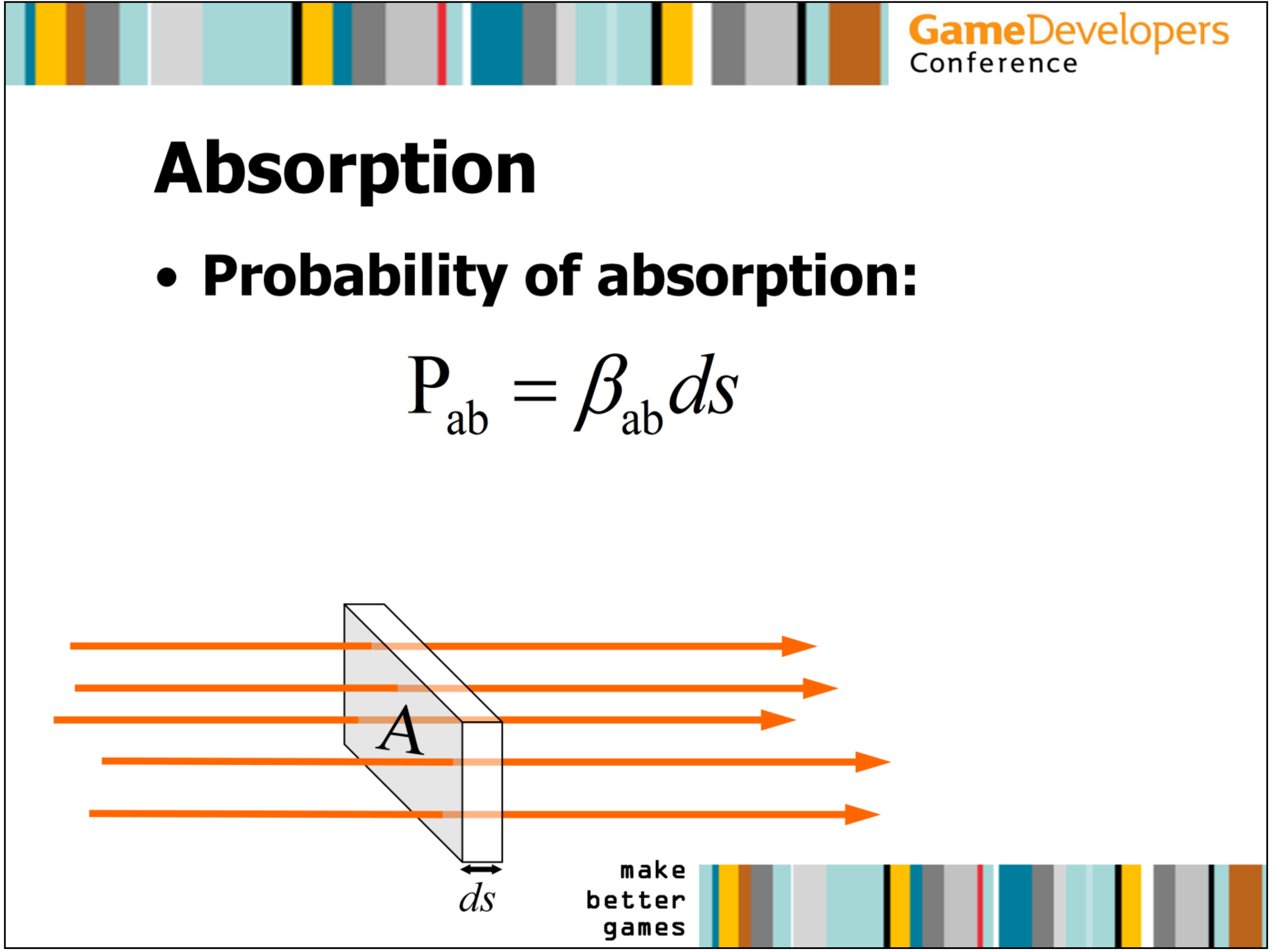
- **Probability of absorption:**

$$P_{ab} = A_{ab} / A = \sigma_{ab} \rho_{ab} \; ds = \beta_{ab} ds$$



make better games

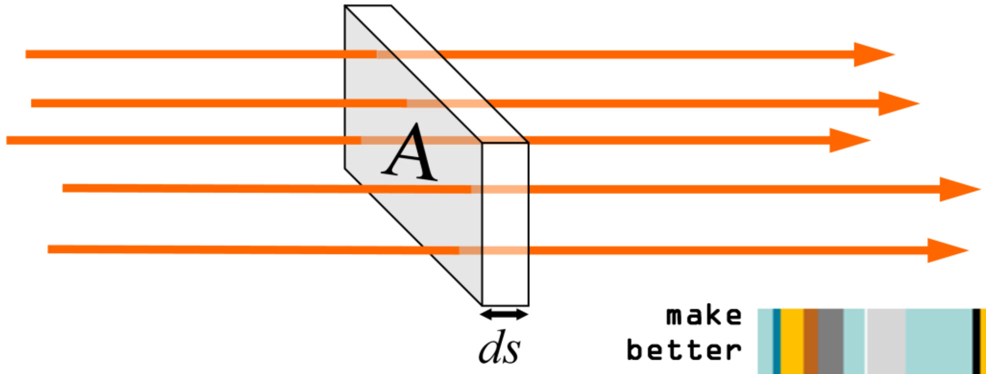
To understand this quantity more intuitively, look at a very thin slab of absorbing medium with depth ds and area A. Photons are traversing it in a perpendicular direction. The total absorption area of the slab is the particle cross section times the number of particles, or the cross section times the particle density times the volume. The probability that a photon is absorbed equals the absorption area over the slab area, which equals the absorption coefficient times *ds*. Note that the absorption coefficient of a mix of different absorbing particles is just the sum of their absorption coefficients.



Absorption

- **Probability of absorption:**

$$P_{ab} = \beta_{ab} ds$$



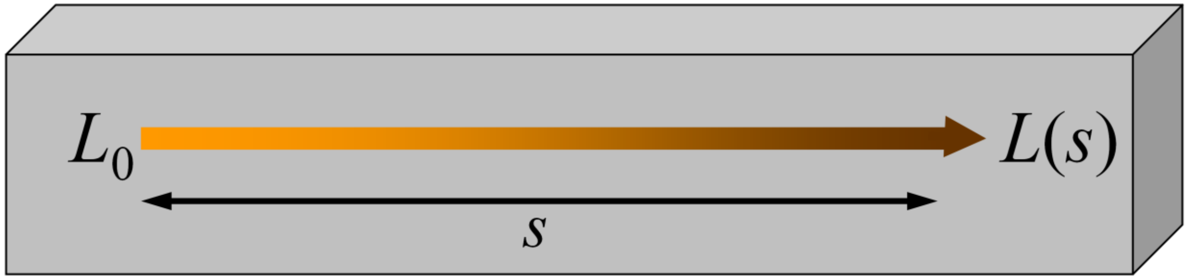
make better games

So the significance of the absorption coefficient is that it relates the distance a photon travels through the medium to its chance of being absorbed. Another way to look at it is that it relates the distance a ray of light travels through the medium to the degree by which the light is attenuated by the medium. Since ds is a differential distance, this is a differential equation.

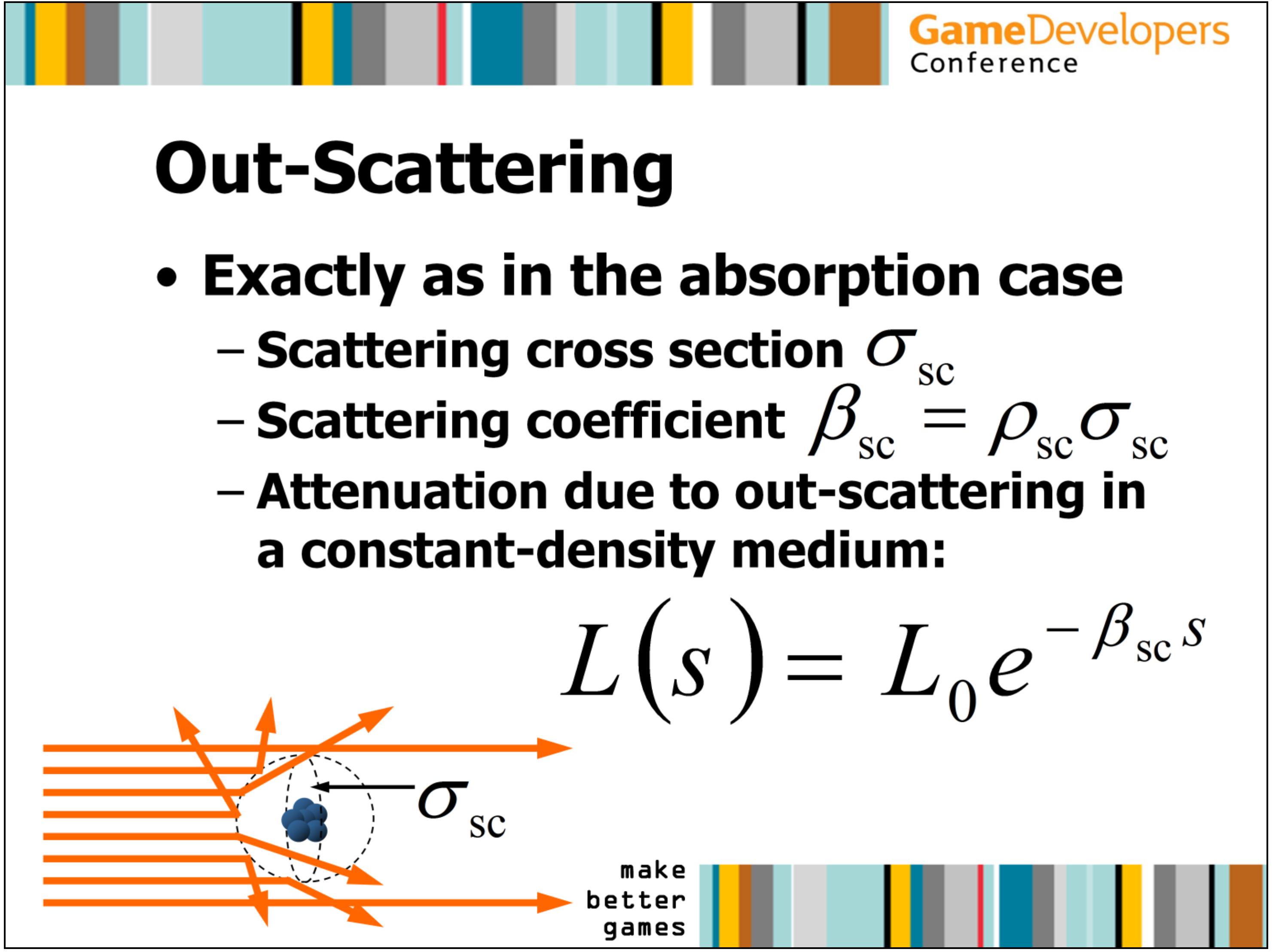
Absorption

- Attenuation of radiance from travel through a <u>constant-density</u> absorptive medium:

$$L(s) = L_0 e^{-\beta_{\mathrm{ab}} s}$$



We can solve this equation to find how radiance is attenuated by traveling through an absorptive medium. The solution for a constant-density medium is an exponential decay function (the proceedings paper has derivations for variable-density media). Similarly to the absorption cross-section, the absorption coefficient can be wavelength dependent (e.g. an RGB triple given we are using the “fast approach” to spectral variation). The magnitude and color of this triple will depend on the exact mix of absorptive particles in the air—tables with real-world values are available.

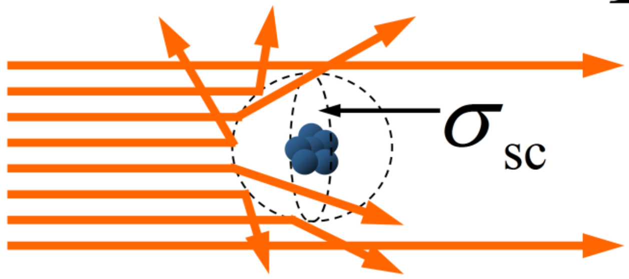


Game Developers Conference

Out-Scattering

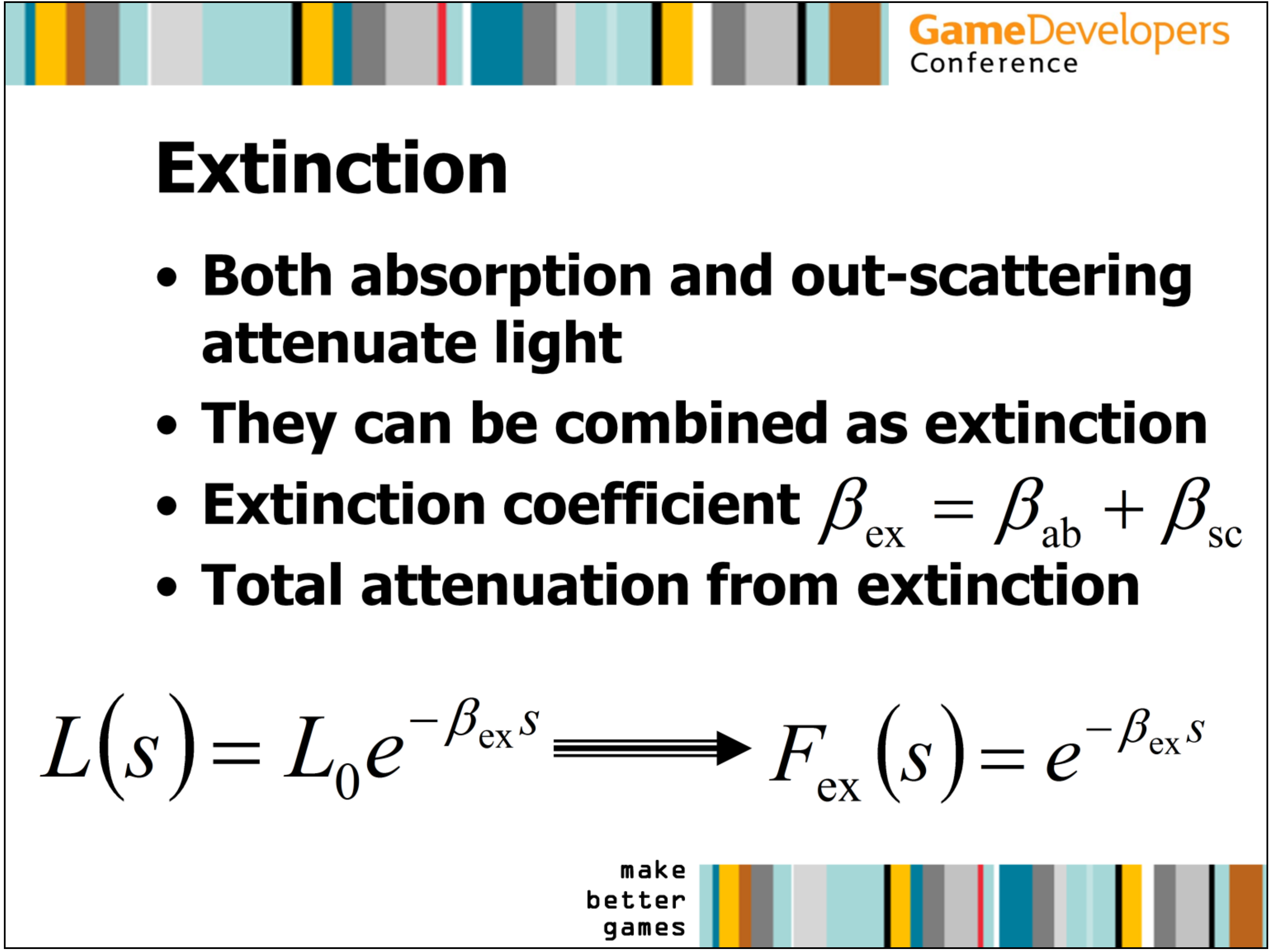
- Exactly as in the absorption case
  - Scattering cross section σ_{sc}
  - Scattering coefficient $\beta_{sc} = \rho_{sc}\sigma_{sc}$
  - Attenuation due to out-scattering in a constant-density medium:

$$L(s) = L_0 e^{-\beta_{sc} s}$$



make better games

As in the absorption case, we define the scattering cross section as the scattered radiant flux per unit incident irradiance, and the scattering coefficient as the density of scattering particles times the scattering cross section. The attenuation formula is basically the same—here attenuation is caused not by photons being absorbed but by them being scattered out of the path of the ray. Note that like absorption, if we have multiple types of scattering particles in a medium we can sum up their coefficients to get the total scattering coefficient of the medium.



GameDevelopers Conference

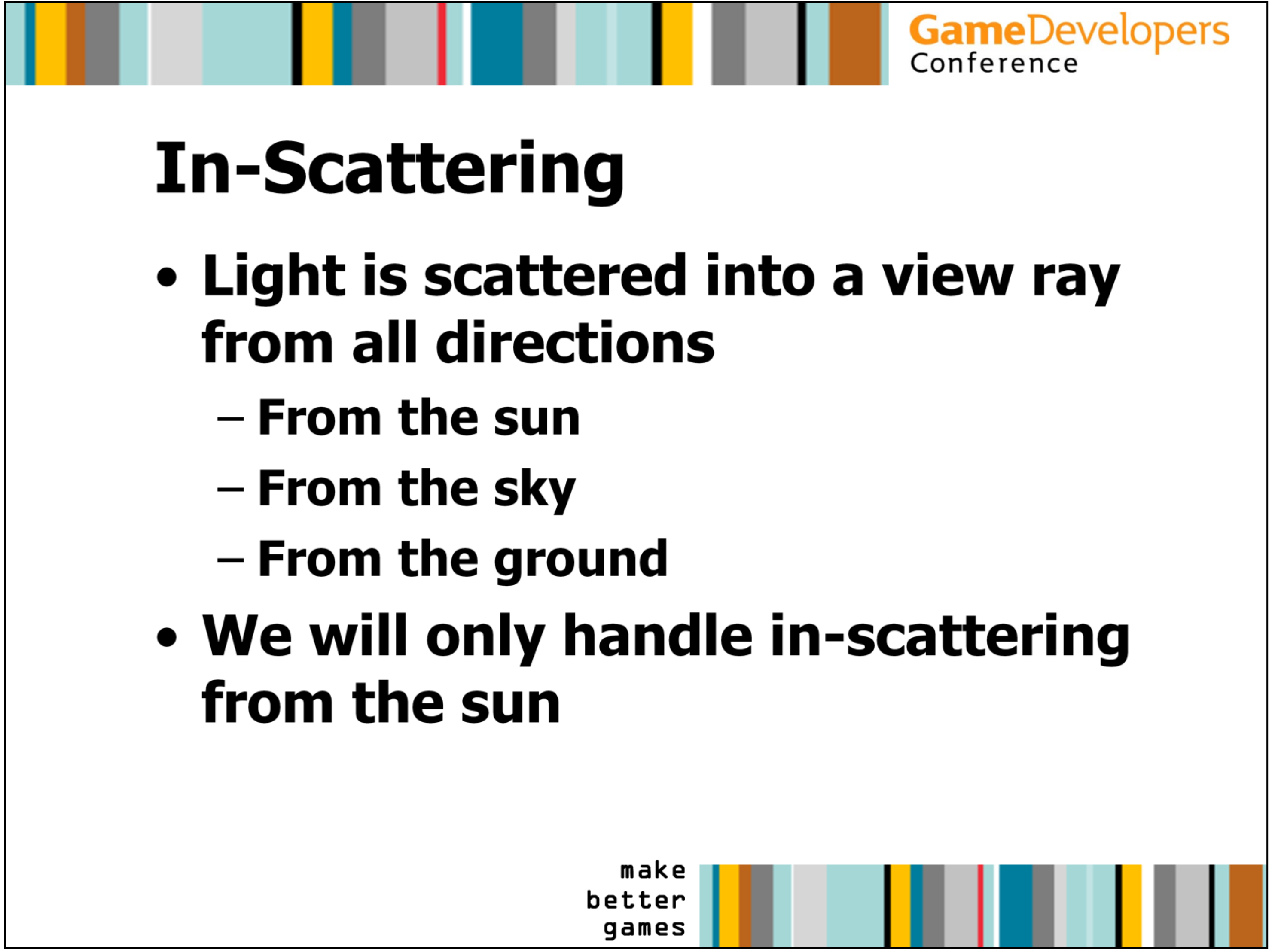
Extinction

- Both absorption and out-scattering attenuate light
- They can be combined as extinction
- Extinction coefficient $\beta_{\text{ex}} = \beta_{\text{ab}} + \beta_{\text{sc}}$
- Total attenuation from extinction

$$L(s) = L_0 e^{-\beta_{\text{ex}} s} \Longrightarrow F_{\text{ex}}(s) = e^{-\beta_{\text{ex}} s}$$

make better games

We can take another step – we can add the total absorption and scattering coefficients to get the extinction coefficient. Note that this is a factor which multiplies the original, unattenuated light.



GameDevelopers Conference

In-Scattering

- Light is scattered into a view ray from all directions
  - From the sun
  - From the sky
  - From the ground
- We will only handle in-scattering from the sun

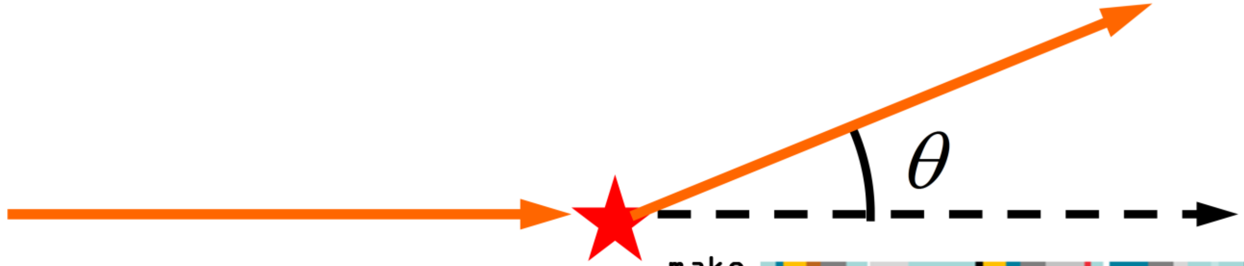
make better games

For this talk, out of all the possible sources for in-scattering, we will only handle in-scattering from the sun. This is the simplest case and the sun is usually the primary contributor to in-scattering.

GameDevelopers Conference

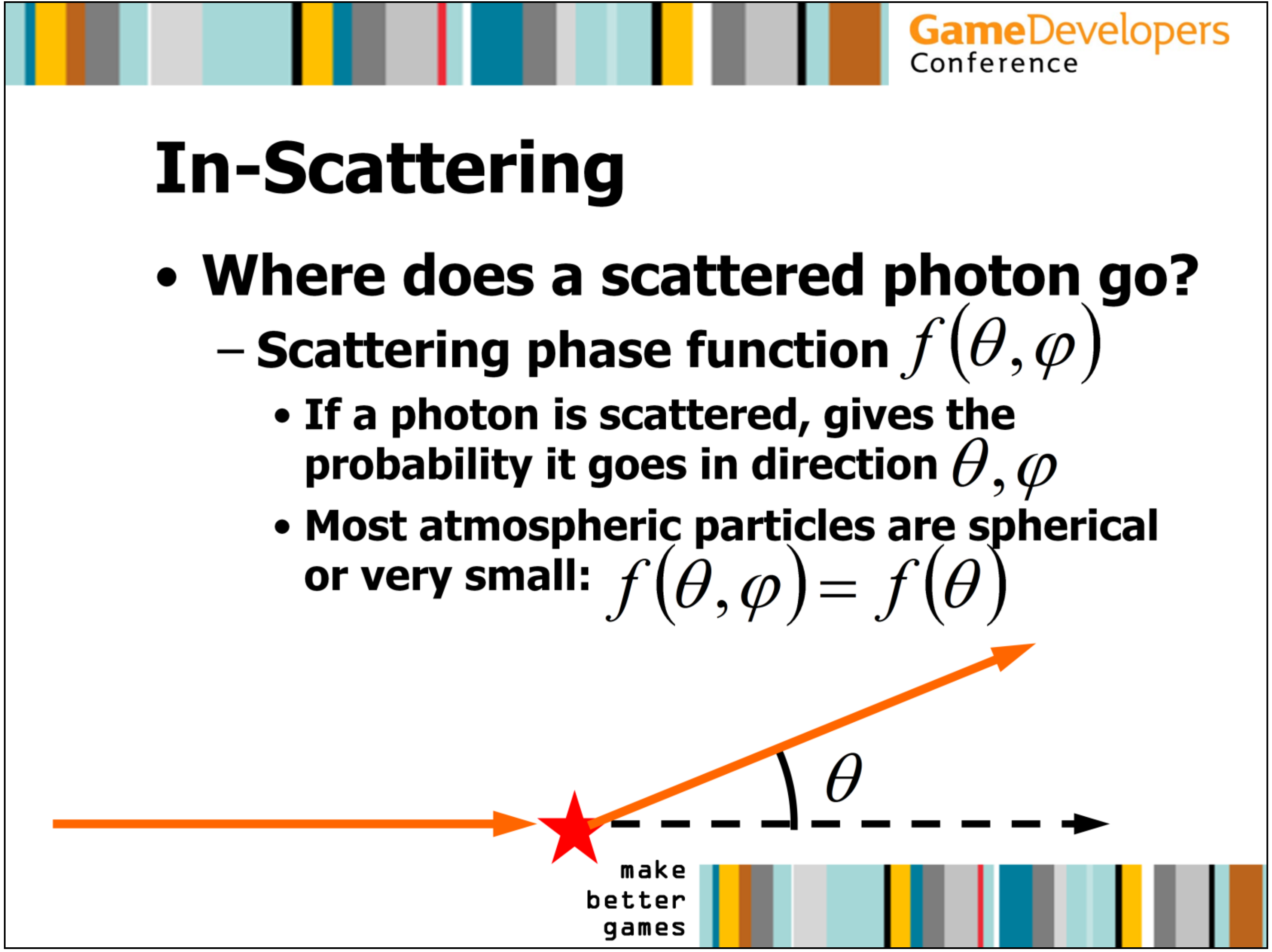
In-Scattering

- Where does a scattered photon go?
  - Scattering phase function $f(\theta, \varphi)$
    - If a photon is scattered, gives the probability it goes in direction θ, φ
    - Most atmospheric particles are spherical or very small: $f(\theta, \varphi) = f(\theta)$



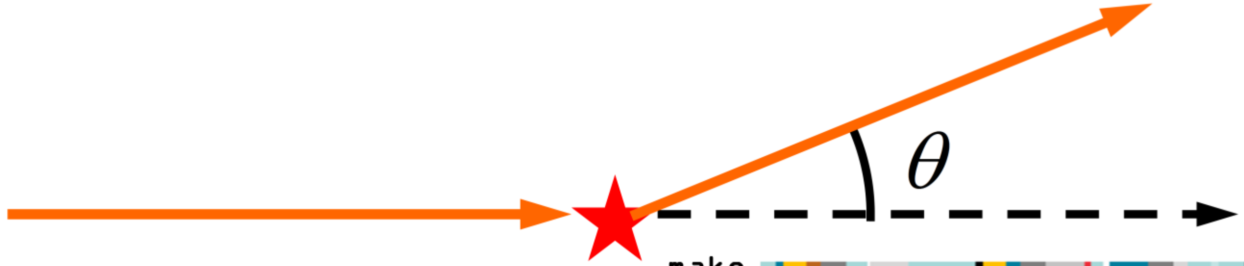
make better games

Unlike extinction, in-scattering adds light to a ray rather than attenuating it. In-scattering and out-scattering are two sides of the same phenomenon. The scattering coefficient tells us the amount of light scattered for both, but not in what direction it was scattered. The phase function is in polar coordinates relative to the original direction. If the particle is small enough (relative to light wavelength) to be considered a point dipole, or if it is spherical (as are most larger particles, like water vapor droplets) the scattering phase function depends only on the angle θ between the original path of the light and the new path.



In-Scattering

- Where does a scattered photon go?
  - Scattering phase function $f(\theta,\varphi)$
    - If a photon is scattered, gives the probability it goes in direction θ,φ
    - Most atmospheric particles are spherical or very small: $f(\theta,\varphi) = f(\theta)$

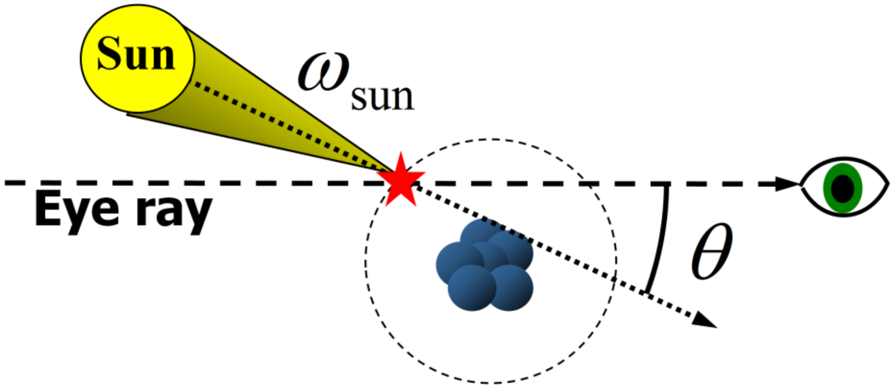


In this talk, we will assume that the phase functions are wavelength-independent (unlike the scattering coefficient). This is reasonably accurate for most classes of atmospheric particles. The phase function is probability per solid angle—its units are inverse solid angle. Integrating it over the sphere gives us 1, a dimensionless quantity.

GameDevelopers Conference

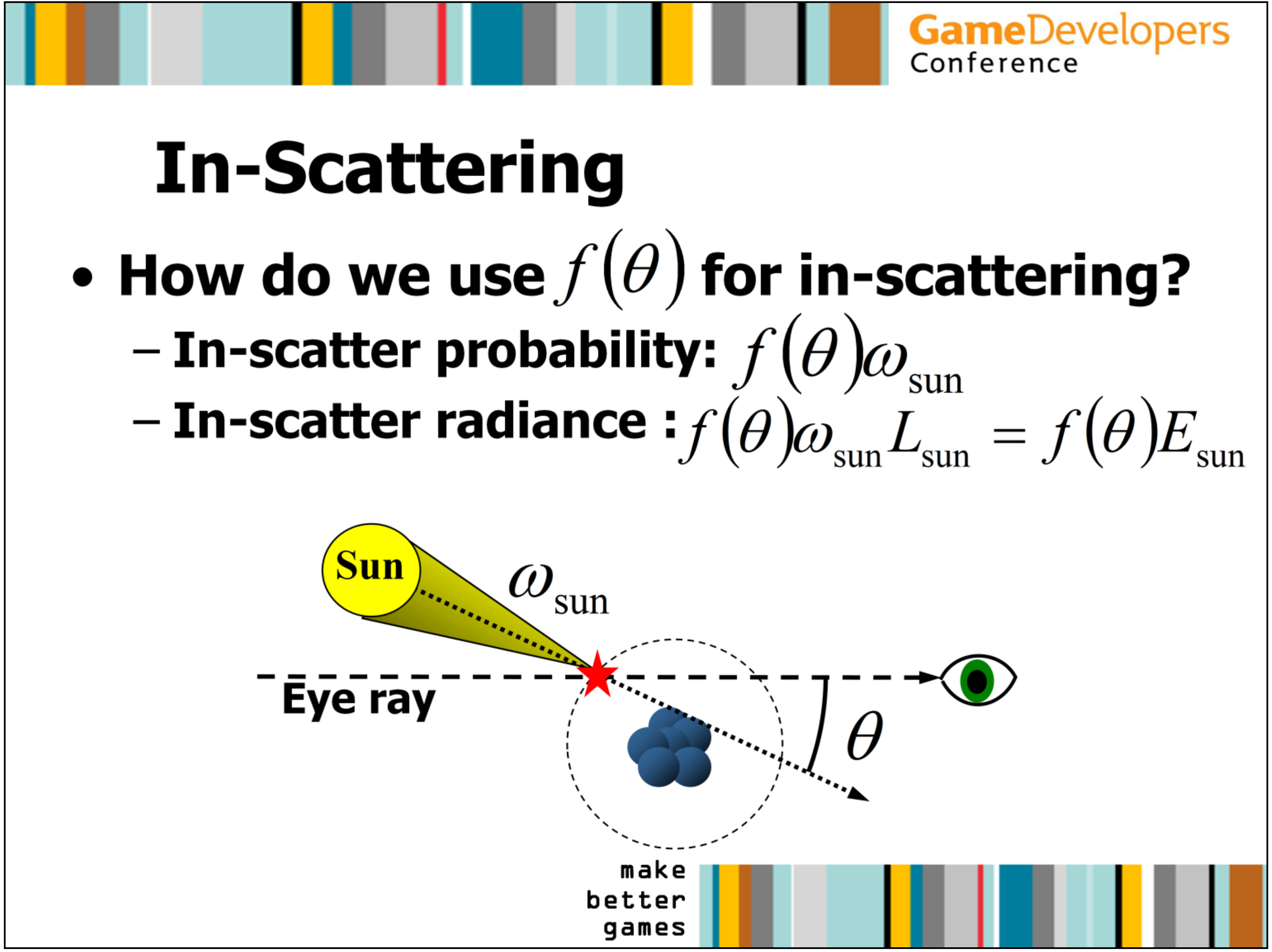
In-Scattering

- How do we use $f(\theta)$ for in-scattering?
  - In-scatter probability: $f(\theta)\omega_{\text{sun}}$



make better games

Given a scattering event (e.g. the viewing ray hit the scattering cross-section of a particle), how much radiance does the event add to the ray by in-scattering? Since we only handle in-scattering from the sun, let us look at the sun's solid angle. This is quite small (a cone about half a degree across) so we can assume a constant theta. Since the scattering phase function is defined as scattering probability per solid angle, the probability of this event causing light to scatter from the sun into the ray is equal to the phase function (evaluated at theta) times the sun's solid angle.

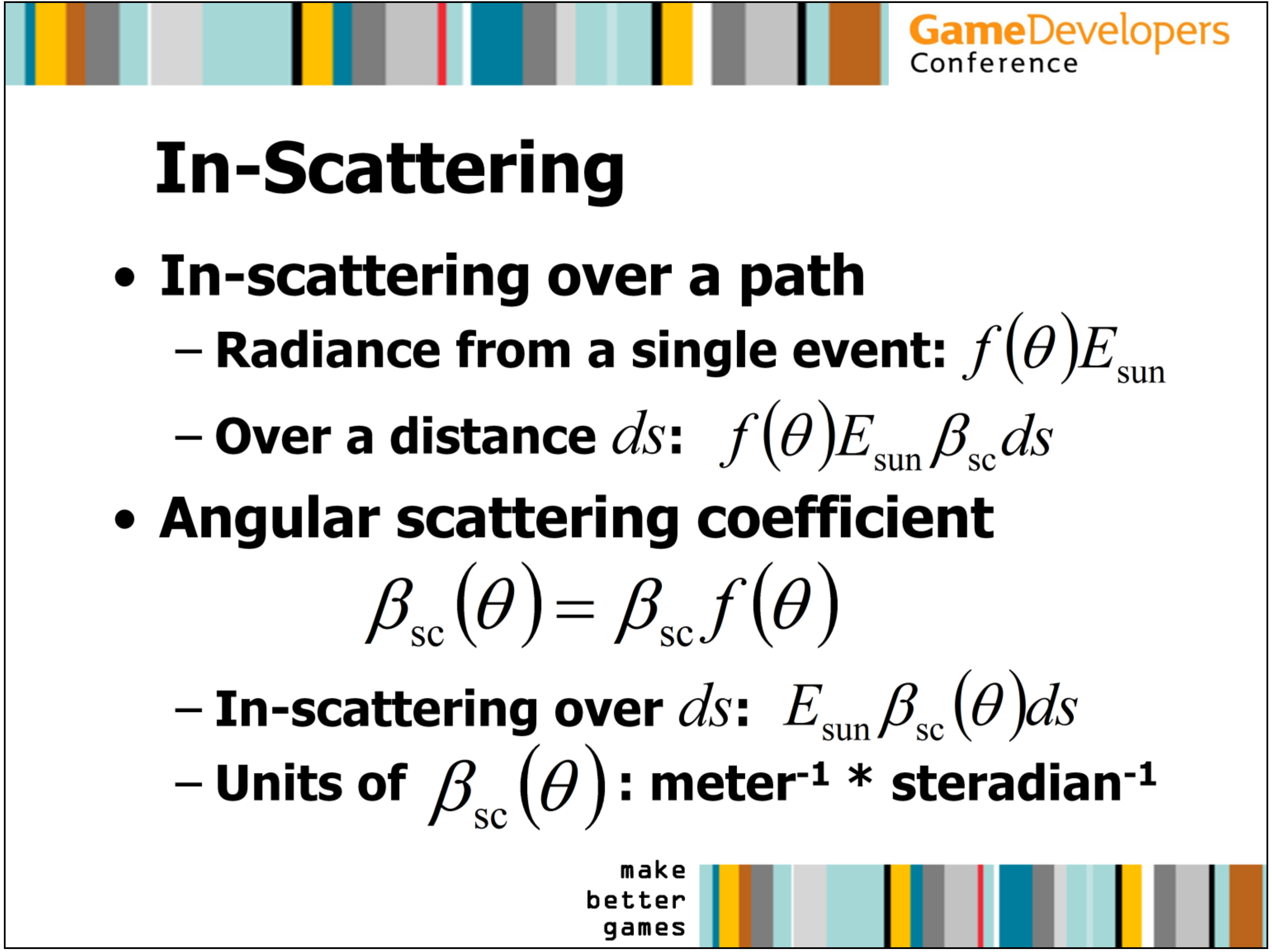


In-Scattering

- How do we use $f(\theta)$ for in-scattering?
  - In-scatter probability: $f(\theta)\omega_{\text{sun}}$
  - In-scatter radiance : $f(\theta)\omega_{\text{sun}}L_{\text{sun}} = f(\theta)E_{\text{sun}}$

Sun
ω_{sun}
Eye ray
θ
GameDevelopers Conference
make better games

Therefore the in-scattered radiance from the sun equals the phase function times the sun's solid angle times its radiance. Since the sun's solid angle times its radiance is a constant and shows up a lot, we can define a handy constant called E_{sun} (it has the same units as irradiance). Then the in-scattered radiance is simply the phase function times this constant. Remember that this is the in-scattered radiance from a single scattering event.



GameDevelopers Conference

In-Scattering

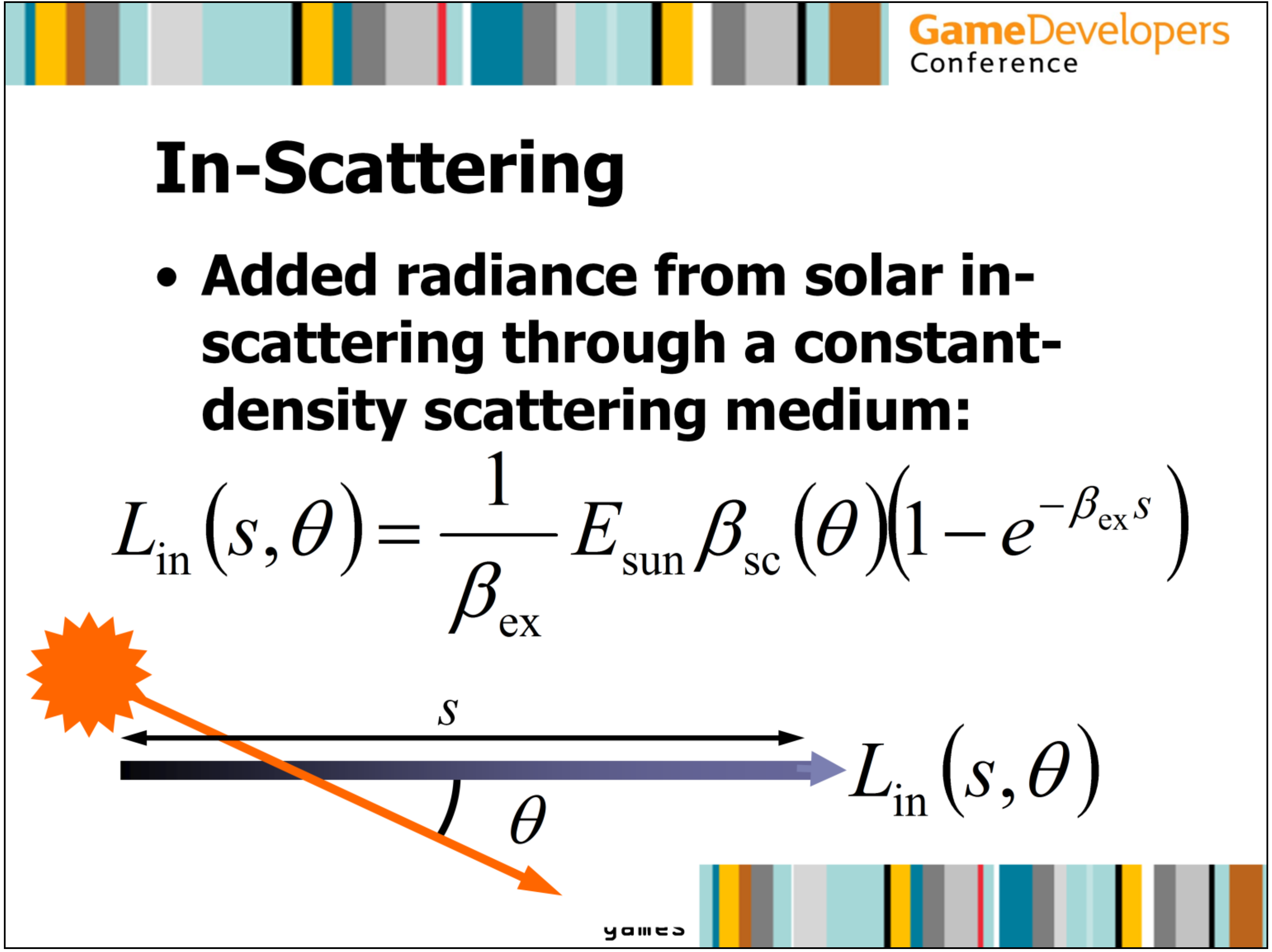
- **In-scattering over a path**
  - **Radiance from a single event:** $f(\theta)E_{\mathrm{sun}}$
  - **Over a distance** ds**:** $f(\theta)E_{\mathrm{sun}}\beta_{\mathrm{sc}}ds$
- **Angular scattering coefficient**

$$\beta_{\mathrm{sc}}(\theta)=\beta_{\mathrm{sc}}f(\theta)$$

  - **In-scattering over** ds**:** $E_{\mathrm{sun}}\beta_{\mathrm{sc}}(\theta)ds$
  - **Units of** $\beta_{\mathrm{sc}}(\theta)$ **: meter^{-1} * steradian^{-1}**

make better games

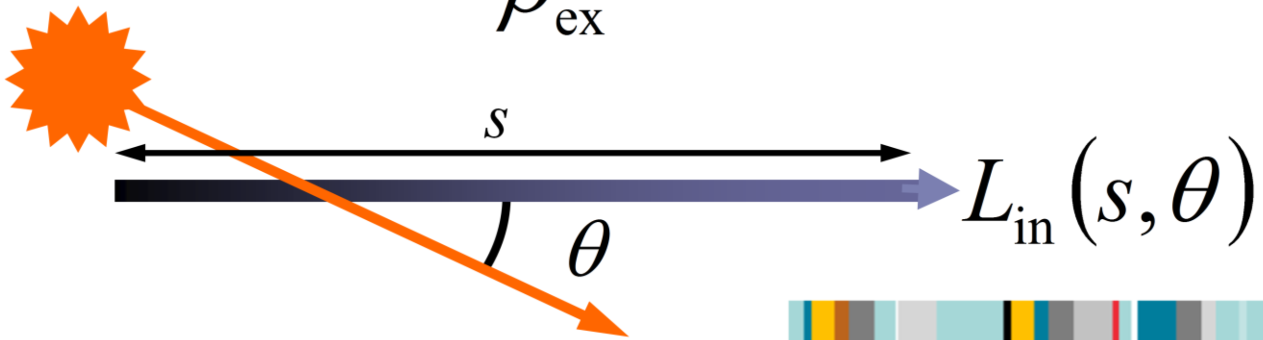
To get the total added radiance over an infinitesimal distance *ds*, we must multiply the in-scattered radiance from a single scattering event by the probability of such an event, which is the scattering coefficient times *ds*. We will define the angular scattering coefficient as the phase function times the scattering coefficient. Then the added radiance in-scattered over *ds* is the angular scattering coefficient times E_{sun} times *ds*.



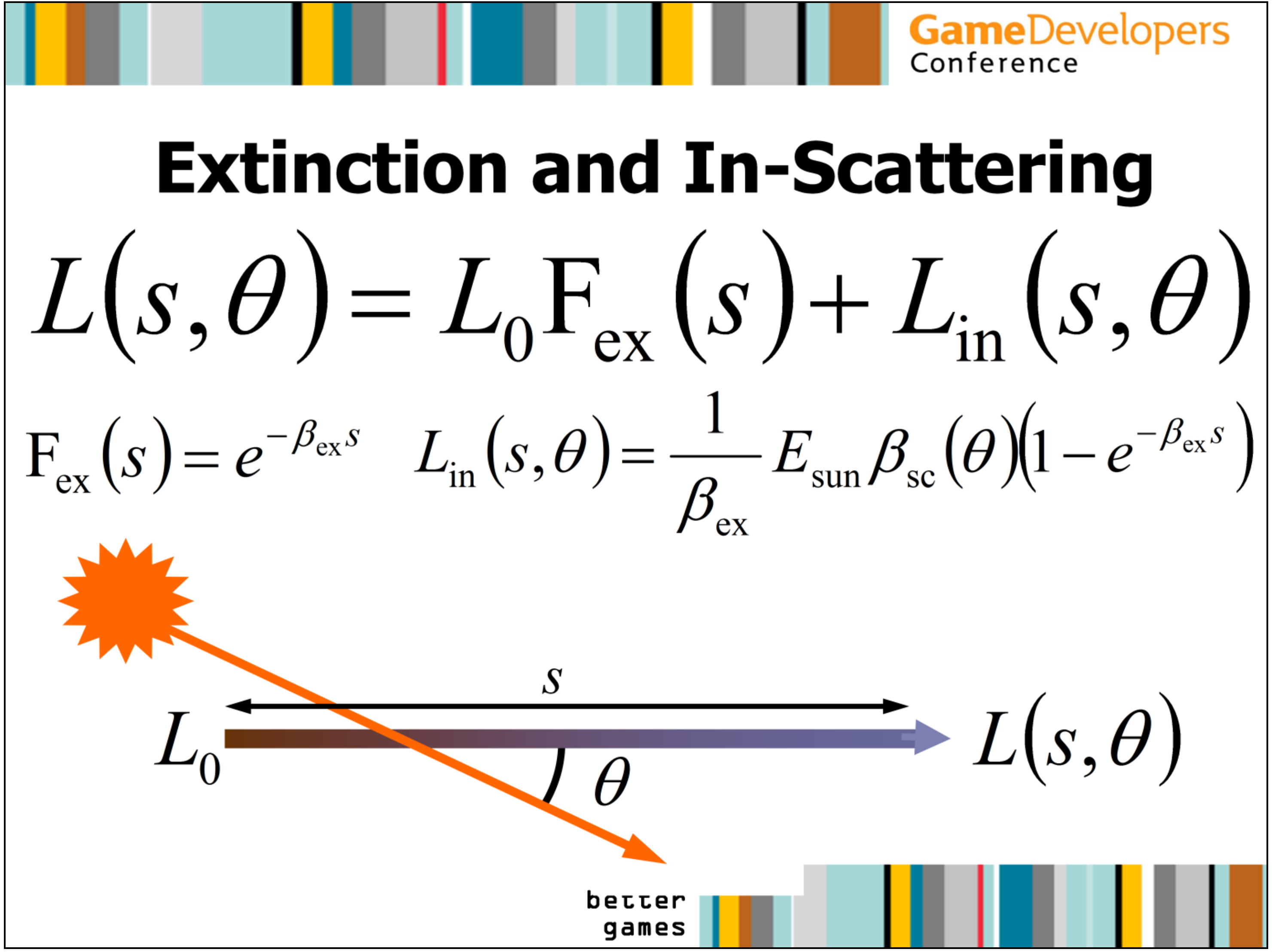
In-Scattering

- **Added radiance from solar in-scattering through a constant-density scattering medium:**

$$L_{\mathrm{in}}(s,\theta)=\frac{1}{\beta_{\mathrm{ex}}}E_{\mathrm{sun}}\beta_{\mathrm{sc}}(\theta)\left(1-e^{-\beta_{\mathrm{ex}}s}\right)$$



Getting from the added differential radiance to a closed-form expression is not as simple as for extinction. The reason is that extinction can be considered in isolation, but in-scattering cannot. This is because in-scattered light undergoes extinction on its way to the eye. The resulting derivation is a little complex, especially if the coefficients vary over the path (see proceedings paper for details). However, with our assumptions the resulting formula is a reasonably simple function of the travel distance and theta (the angle between the viewing ray and the sun direction).

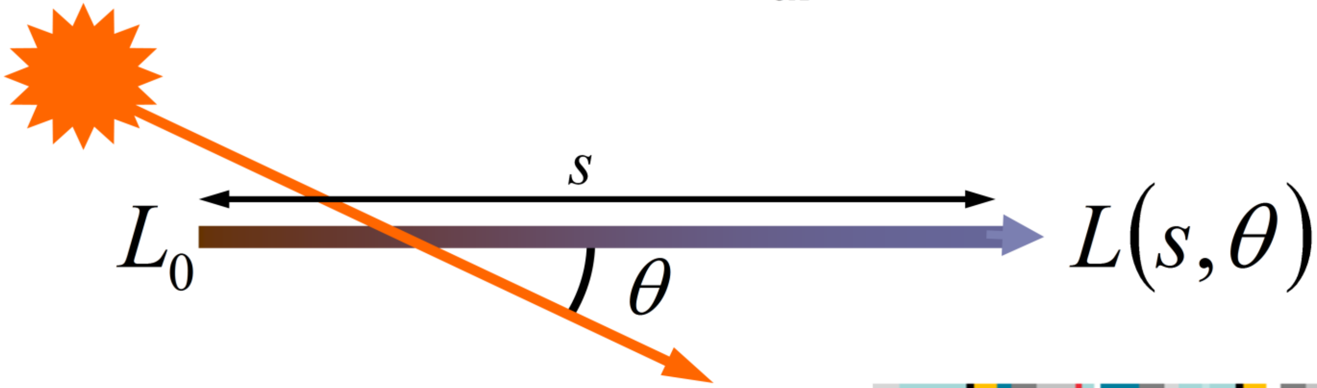


GameDevelopers Conference

Extinction and In-Scattering

$$L(s,\theta) = L_0 \mathrm{F}_{\mathrm{ex}}(s) + L_{\mathrm{in}}(s,\theta)$$

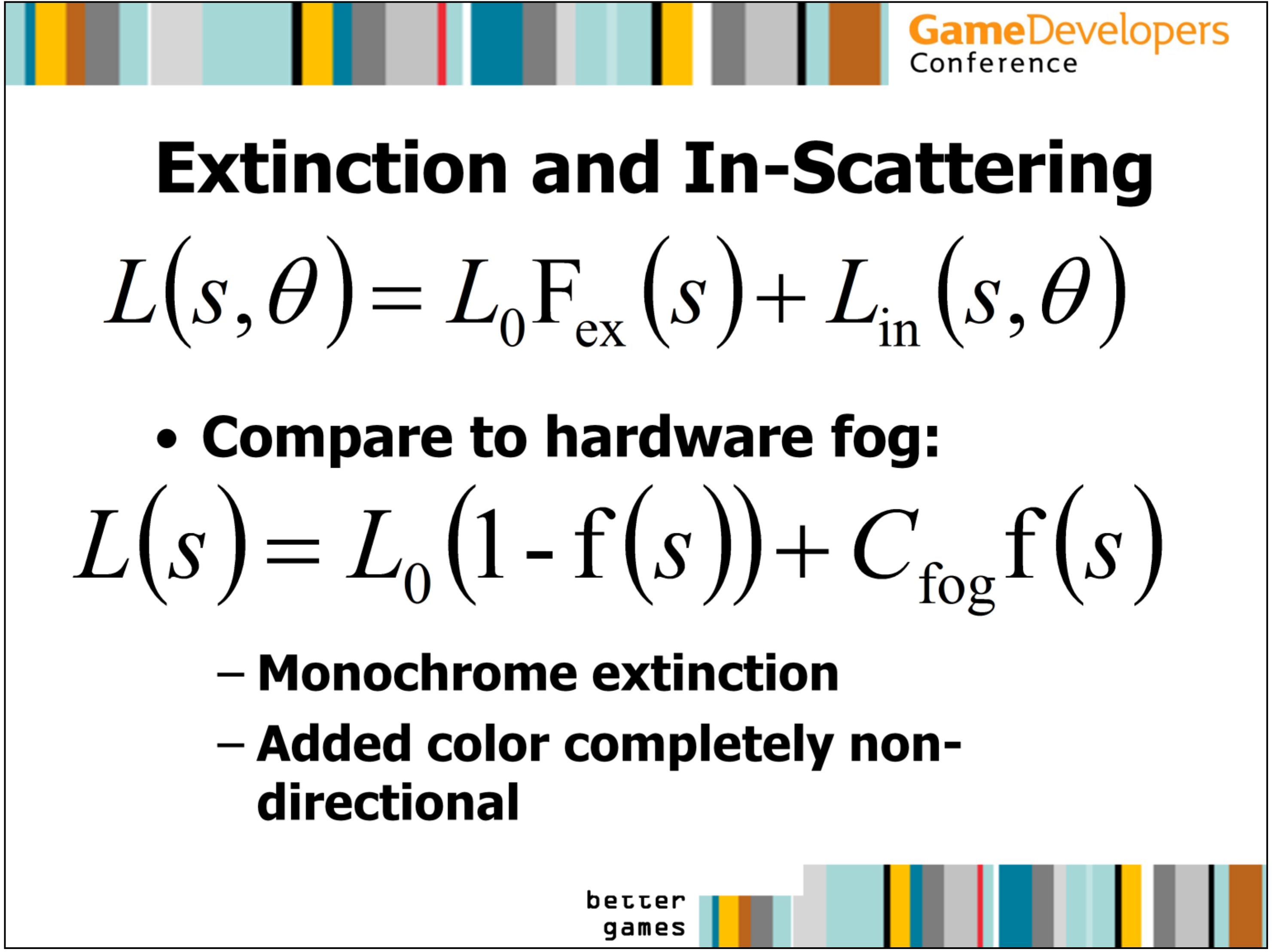
$$\mathrm{F}_{\mathrm{ex}}(s) = e^{-\beta_{\mathrm{ex}} s} \qquad L_{\mathrm{in}}(s,\theta) = \frac{1}{\beta_{\mathrm{ex}}} E_{\mathrm{sun}} \beta_{\mathrm{sc}}(\theta)\left(1 - e^{-\beta_{\mathrm{ex}} s}\right)$$



better games

Summarizing our results so far:

The final radiance (or pixel) value is the original value times an extinction factor (function of distance) plus an in-scattering value (function of distance and theta).



Extinction and In-Scattering

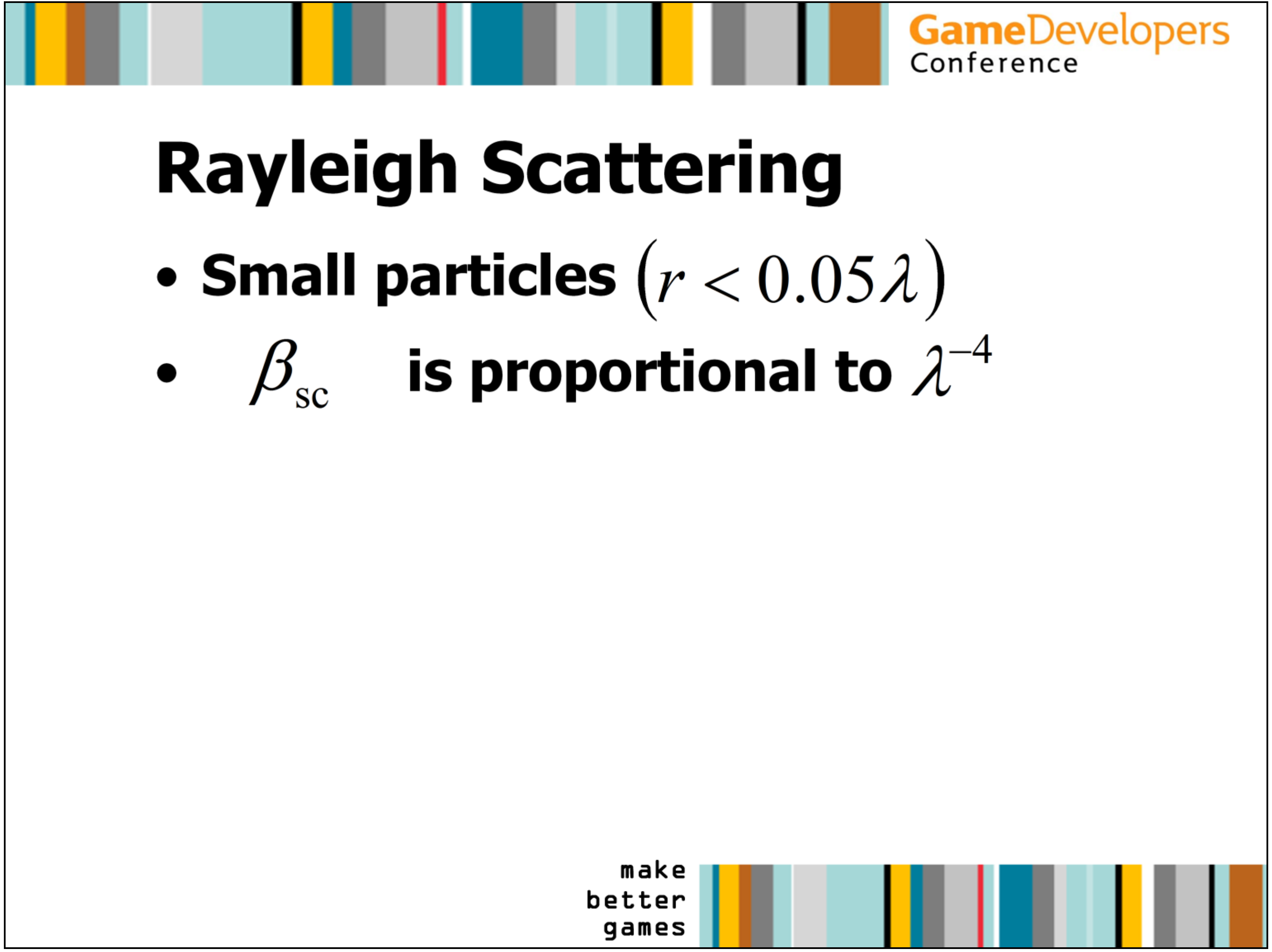
$$L(s,\theta) = L_0 \mathrm{F}_{\mathrm{ex}}(s) + L_{\mathrm{in}}(s,\theta)$$

- **Compare to hardware fog:**

$$L(s) = L_0(1 - \mathrm{f}(s)) + C_{\mathrm{fog}}\mathrm{f}(s)$$

  - **Monochrome extinction**
  - **Added color completely non-directional**

Let us compare this to hardware fog. In our model, the extinction and added light are two independent RGB quantities (for example, changing sunlight affects the amount of in-scattering without affecting extinction). Usually extinction is reddish. The added light color can be strongly directional. The whole thing is based on real physical principles. With hardware fog, the extinction is monochrome. The added light color is non-directional, so looking away from the sun or towards the sun will produce the same addition. Fog does not map to actual physical phenomena.



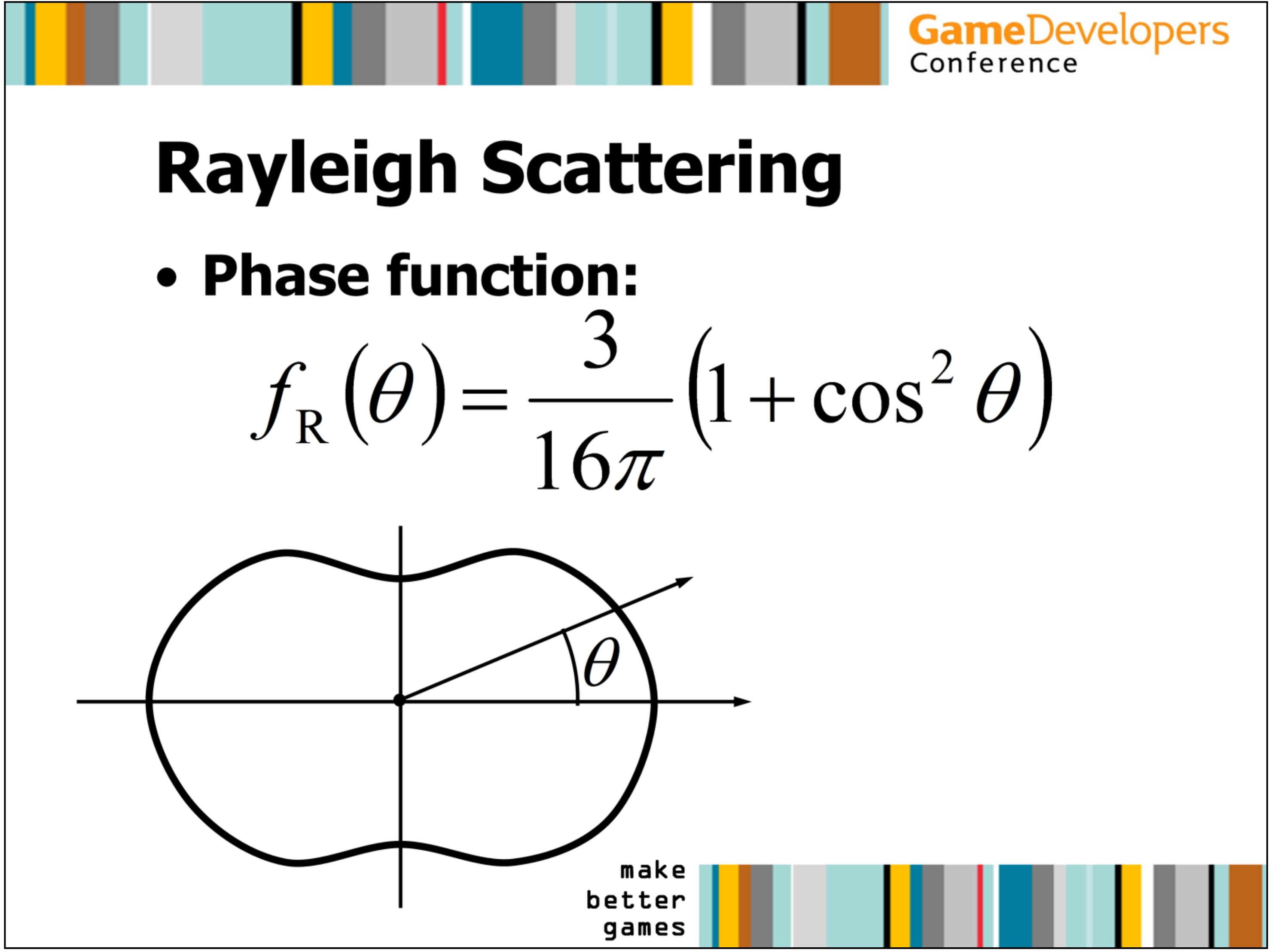
GameDevelopers Conference

Rayleigh Scattering

- Small particles $(r < 0.05\lambda)$
- β_{sc} is proportional to λ^{-4}

make better games

The form of the scattering coefficient – how it depends on angle and wavelength—depends on the <u>type</u> of particle doing the scattering (density affects magnitude but not form). For particles much smaller than the wavelength of light (e.g. air molecules), the form is that of Rayleigh scattering. Exact expressions can be found in the literature, but the important detail is the 1 / lambda^4 wavelength dependence—blue (shorter) wavelengths are scattered <u>much</u> more than red (longer) wavelengths. This explains the blue color of the sky, the red color of the sun during sunset, and many other phenomena.

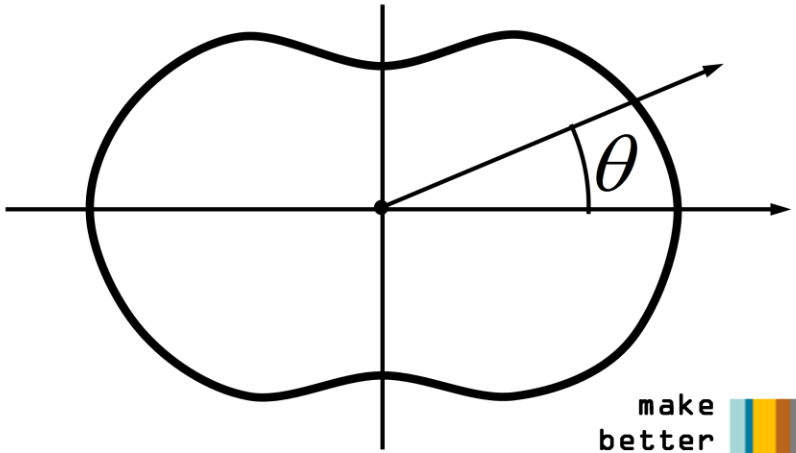


GameDevelopers Conference

Rayleigh Scattering

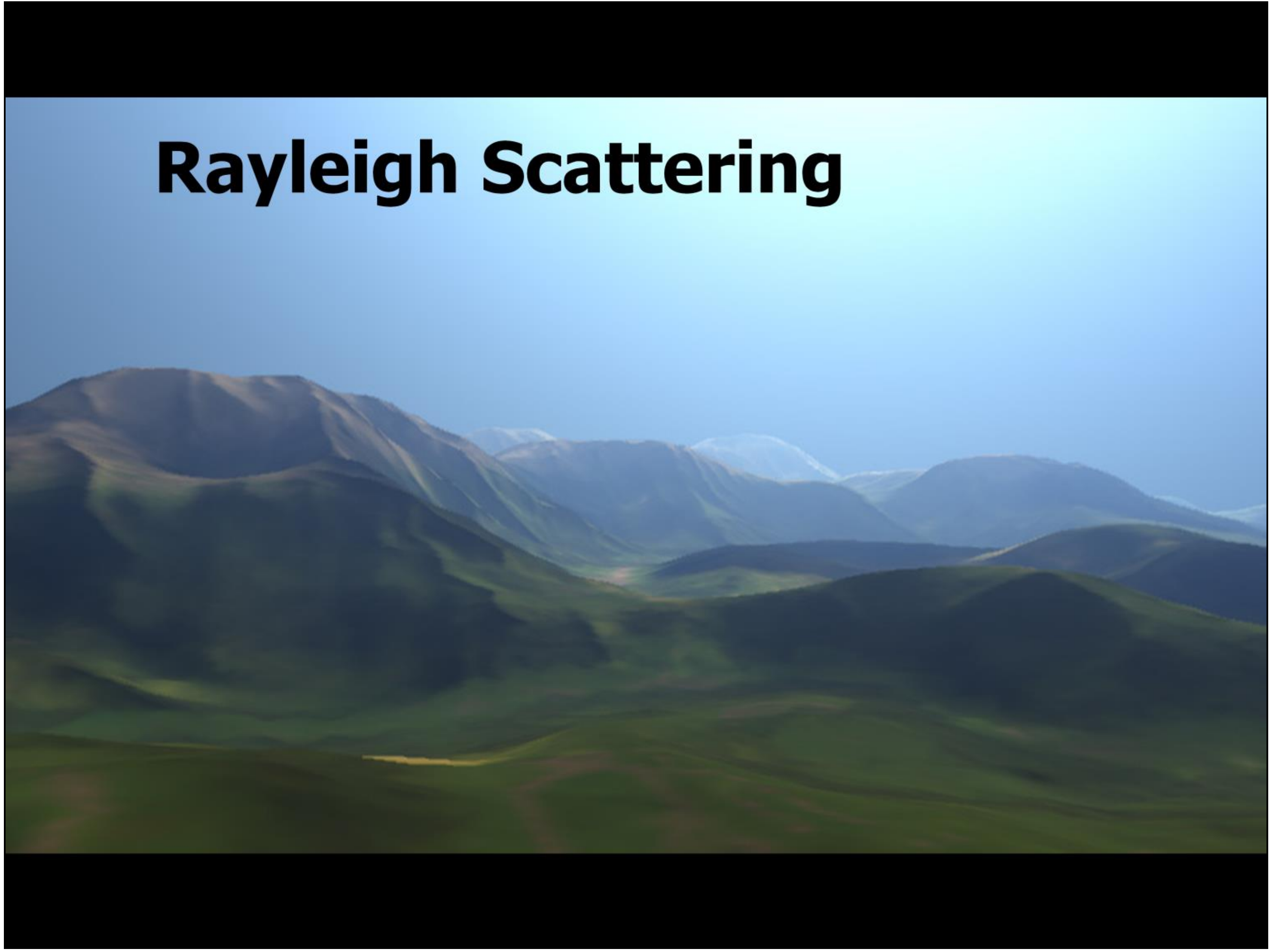
- **Phase function:**

$$f_R(\theta) = \frac{3}{16\pi}\left(1 + \cos^2\theta\right)$$



make better games

Let us look at the Rayleigh scattering phase function plot, a polar plot where the distance from the origin in each direction corresponds to the scattering probability in that direction. The phase function is symmetrical – Rayleigh particles scatter light equally in the forward and backward directions (less sideways). The phase function is fairly simple to compute, especially since it uses the cosine of the angle between the view direction and the sun direction, which is easily computed via a dot product.

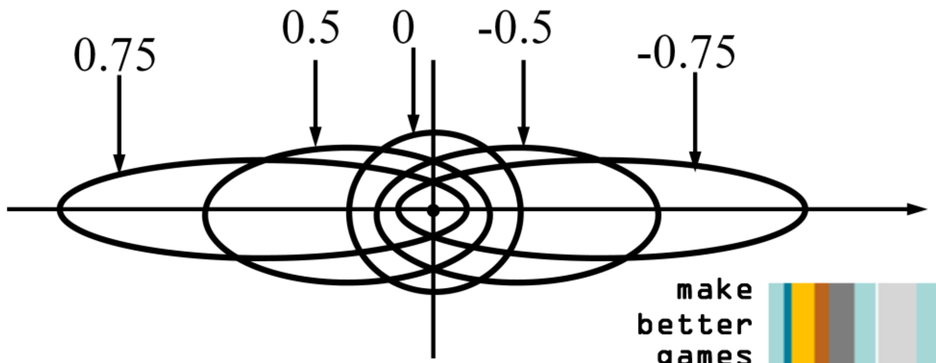


A screenshot from our real-time demo, with Rayleigh scattering predominating.

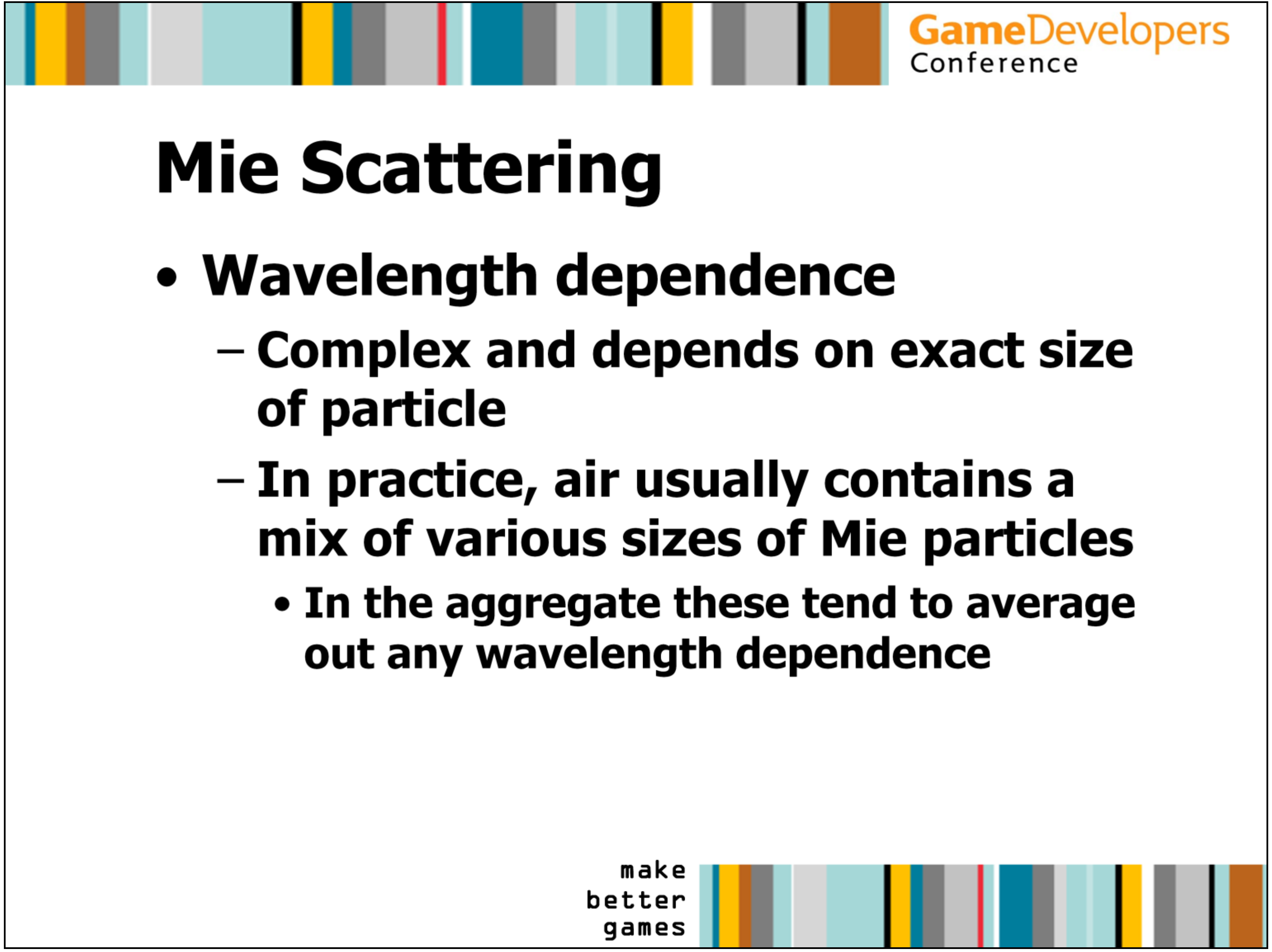
Mie Scattering

- **Larger, spherical particles**
- **Phase function approximation:**
  - **Henyey-Greenstein**

$$f_{\mathrm{HG}}(\theta) = \frac{(1-g)^2}{4\pi\left(1+g^2-2g\cos(\theta)\right)^{3/2}}$$



Mie scattering theory applies to larger spherical particles. The Mie phase function is complex, but usually well approximated by the Henyey-Greenstein phase function. Which is simply the polar form of the equation for an ellipse, with the origin at one of its foci. The magnitude of g determines the ellipse's eccentricity and its sign determines the ellipse's direction—if negative, the ellipse points forwards (forward-scattering), and if positive the ellipse points backwards (back-scattering). For most particles, g is negative and increases in magnitude with particle size. This function also uses the angle cosine (dot product).



GameDevelopers Conference

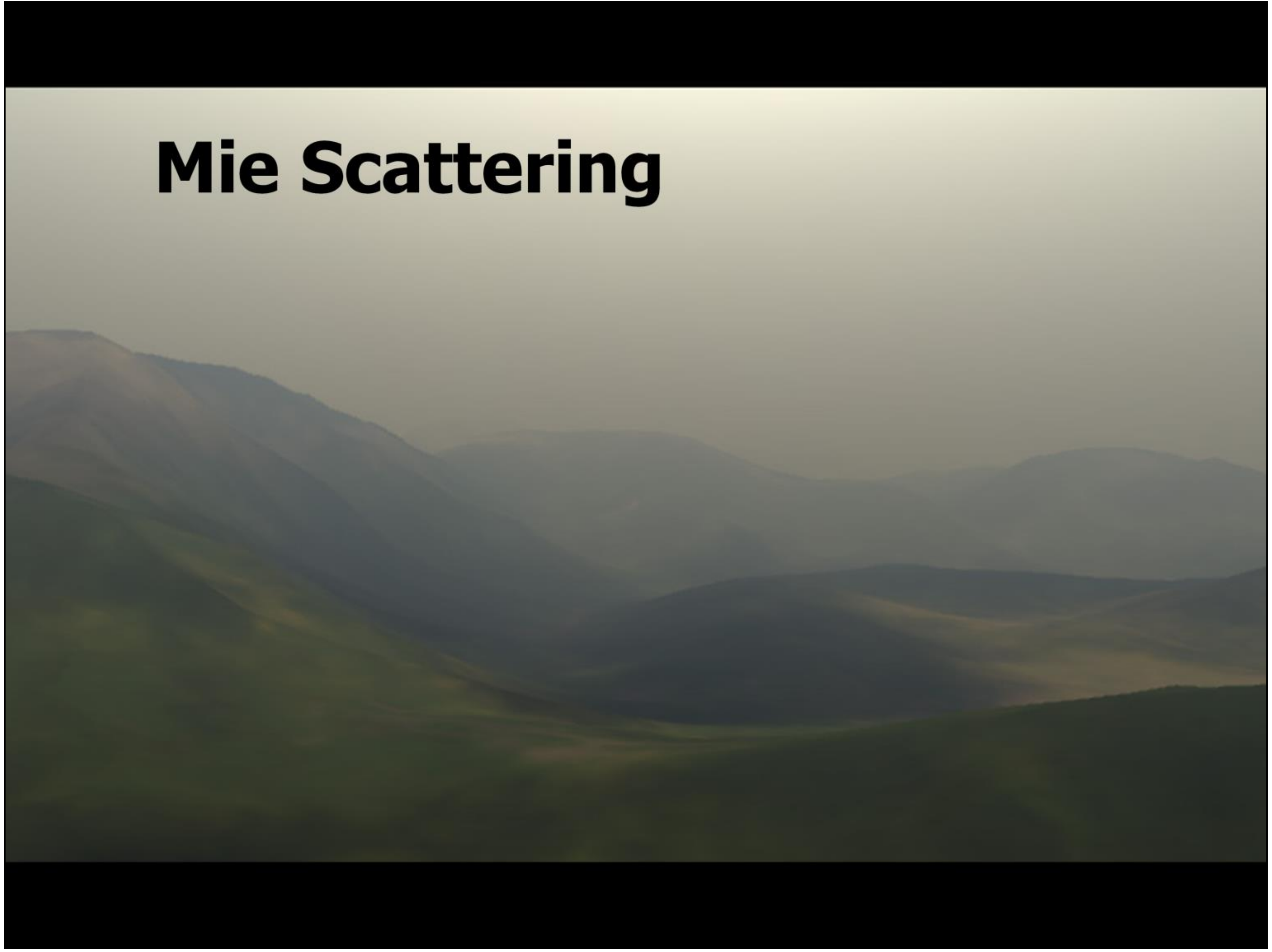
Mie Scattering

- Wavelength dependence
  - Complex and depends on exact size of particle
  - In practice, air usually contains a mix of various sizes of Mie particles
    - In the aggregate these tend to average out any wavelength dependence

make better games

So we will assume no wavelength dependence for Mie scattering. This means that Mie scattering tends to preserve the color of the scattered light.

This can be seen in overcast days, which tend to be gray rather than blue. This is because the scattering is predominated by Mie-scattering water droplets rather than the Rayleigh-scattering air molecules which characterize scattering on clear days.



A screenshot from our real-time demo, with Mie scattering predominating.

GameDevelopers Conference

Combined Scattering

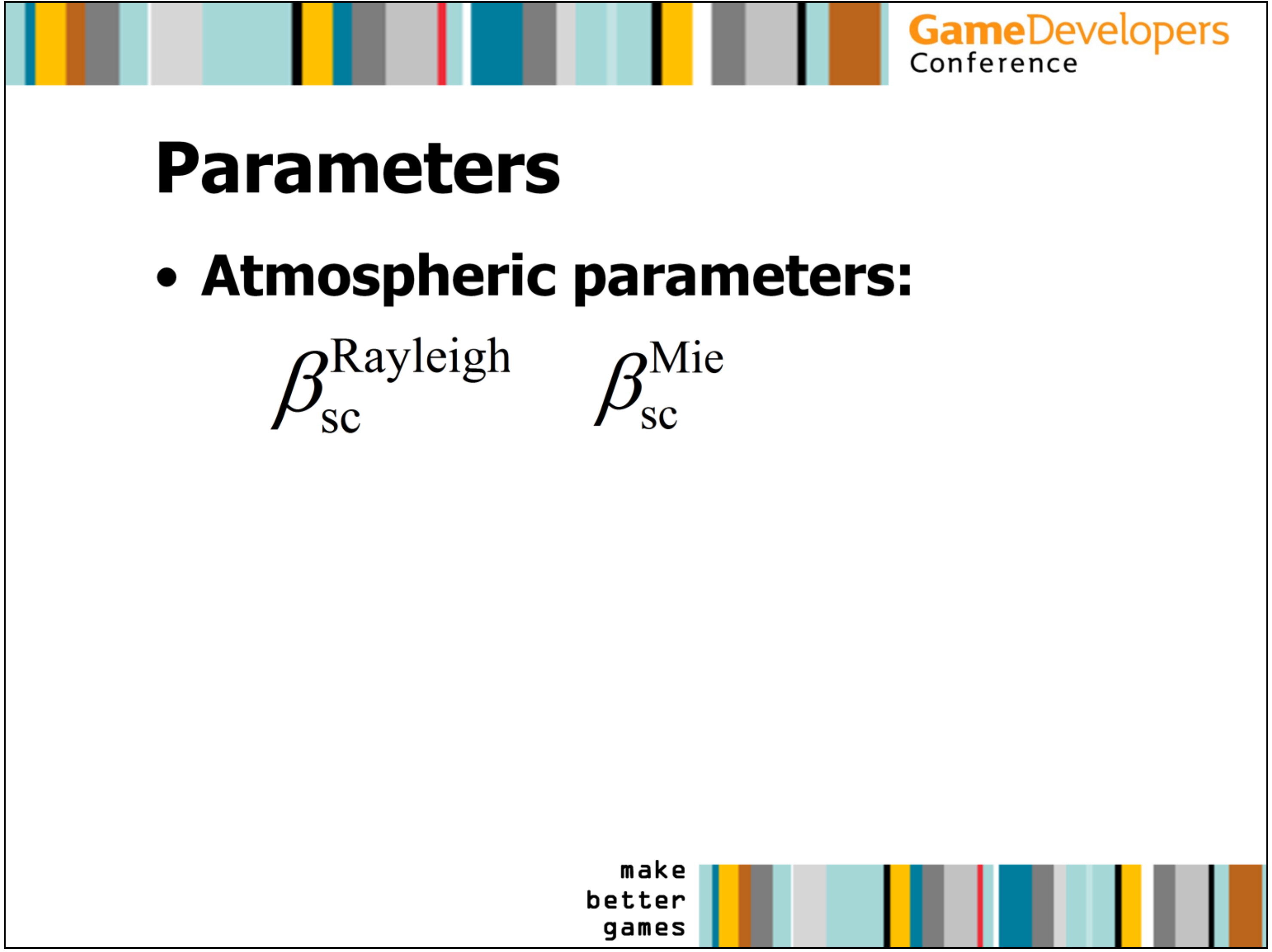
- In practice, air contains both Rayleigh and Mie scatterers
- Absorption is usually slight
- We will use:

$$\beta_{\text{ex}} = \beta_{\text{sc}}^{\text{Rayleigh}} + \beta_{\text{sc}}^{\text{Mie}}$$

$$\beta_{\text{sc}}(\theta) = \beta_{\text{sc}}^{\text{Rayleigh}} f_{\text{R}}(\theta) + \beta_{\text{sc}}^{\text{Mie}} f_{\text{HG}}(\theta)$$

make better games

We will assume no absorption (a reasonable approximation unless the air is very polluted), so the extinction coefficient is equal to the scattering coefficient, which is the sum of the Rayleigh and Mie scattering coefficients. The angular scattering coefficient is similarly equal to the sum of the Rayleigh and Mie angular scattering coefficients, each of which is equal to the product of the appropriate scattering coefficient and phase function. These equations combined with the extinction and in-scattering equations from a few slides ago give us all the math we need to compute the interaction of light with the atmosphere.



GameDevelopers Conference

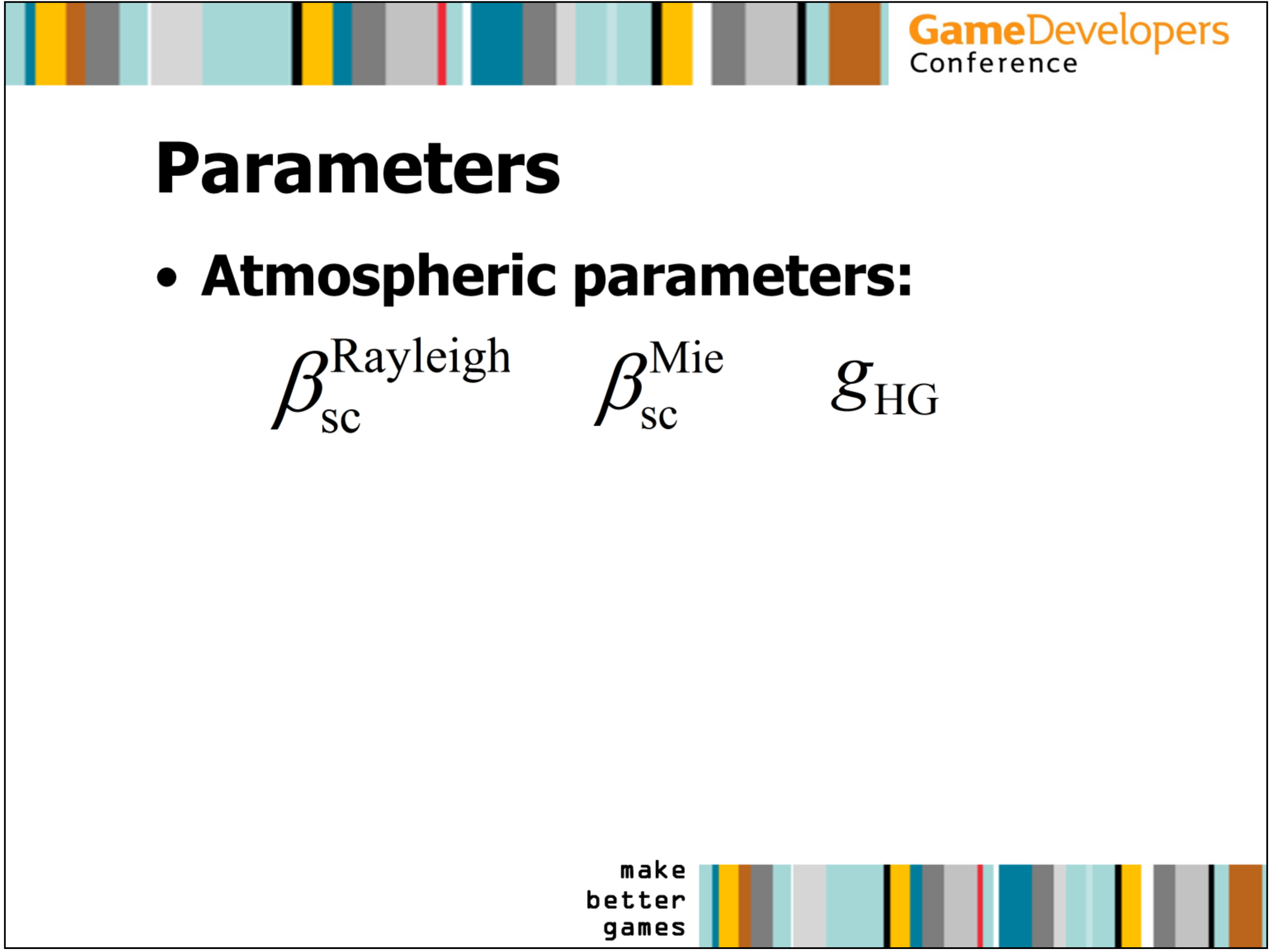
Parameters

- **Atmospheric parameters:**

$$\beta_{\mathrm{sc}}^{\mathrm{Rayleigh}} \qquad \beta_{\mathrm{sc}}^{\mathrm{Mie}}$$

make better games

Now we have all the equations of our model; what are its parameters? First we have the Rayleigh and Mie scattering coefficients, which are properties of the atmosphere. These are functions of the density of air molecules (which depends on elevation) and aerosols (which depend on pollution and humidity) respectively. The coefficients are wavelength-dependent, so they are RGB triples. The color of the Rayleigh coefficient is determined by the inverse fourth power of the wavelengths we are using for R, G and B. The Mie coefficient will usually be monochromatic.

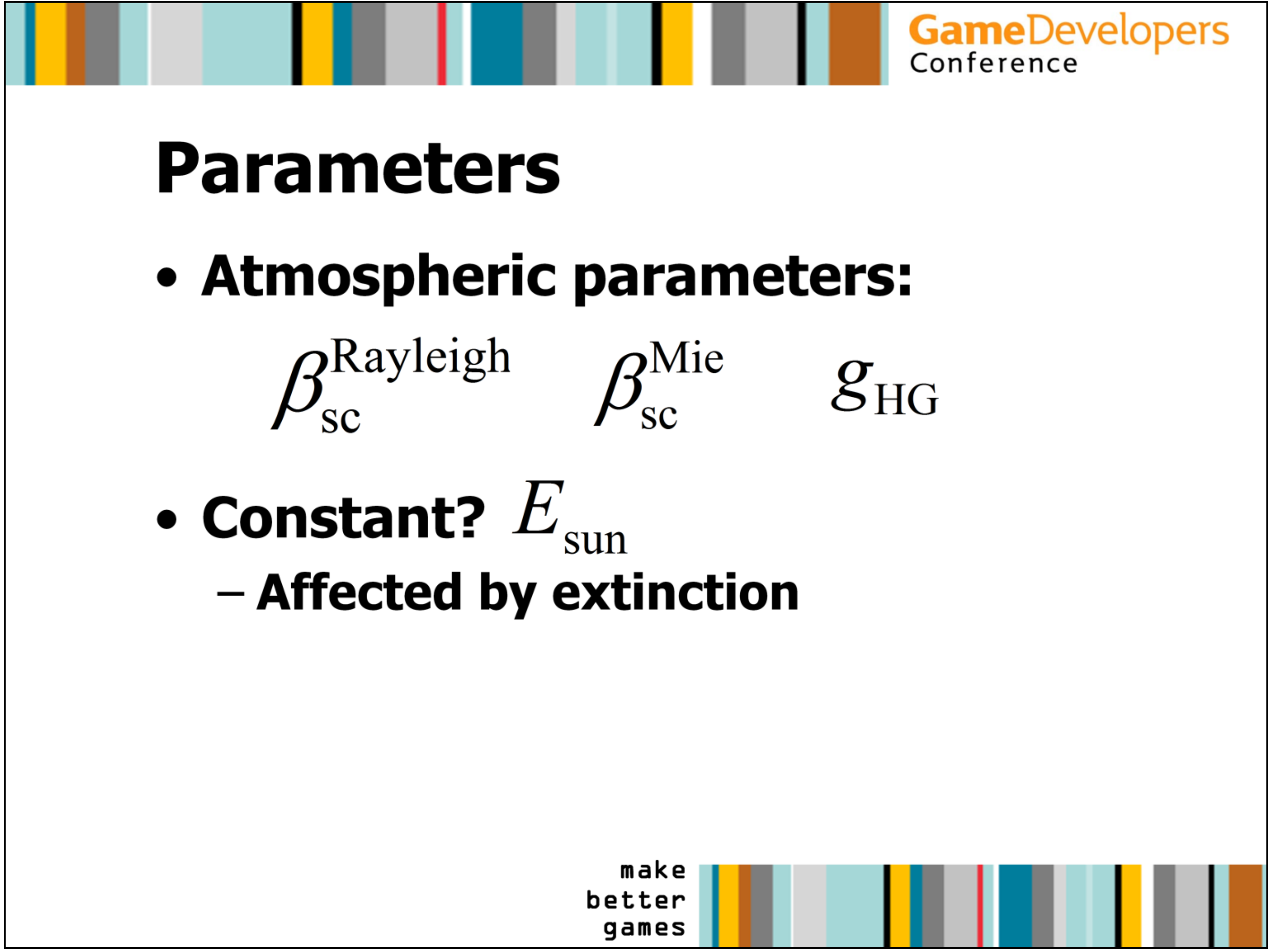


Parameters

- **Atmospheric parameters:**

$$\beta_{\text{sc}}^{\text{Rayleigh}} \quad \beta_{\text{sc}}^{\text{Mie}} \quad g_{\text{HG}}$$

Another atmospheric property is the Henyey-Greenstein *g* parameter. This will usually be negative (forward scattering) – it's exact value depends on the mix of aerosols in the air. Larger particles will tend to increase the absolute value of *g*.



Parameters

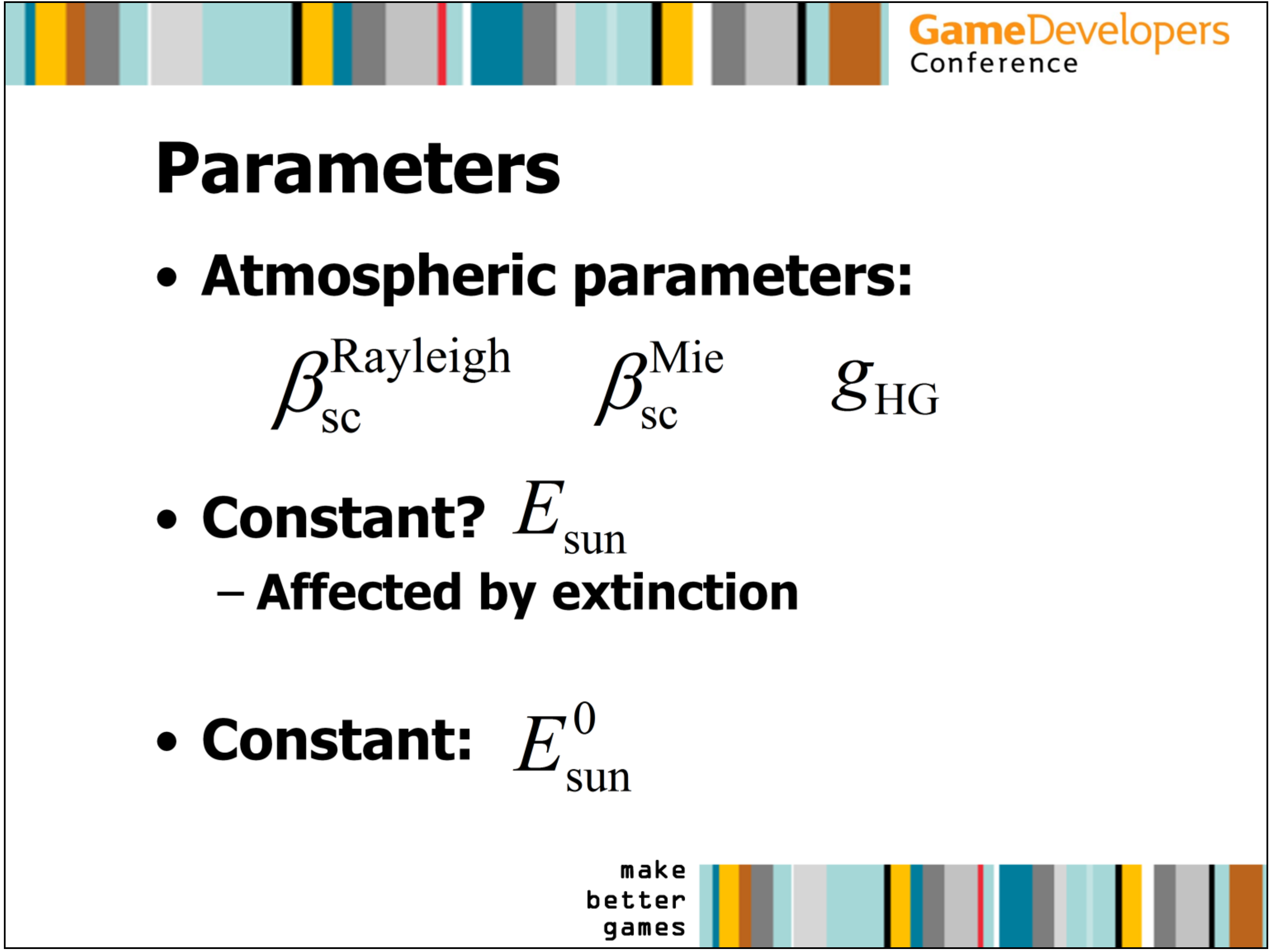
- **Atmospheric parameters:**

$$\beta_{\text{sc}}^{\text{Rayleigh}} \qquad \beta_{\text{sc}}^{\text{Mie}} \qquad g_{\text{HG}}$$

- **Constant?** E_{sun}
  - **Affected by extinction**

GameDevelopers Conference
make better games

Esun is used in the in-scattering equations, but it is not truly a constant – it is itself affected by extinction. To simplify calculations we will use the value of Esun at ground level in all our equations – this needs to be calculated as a function of the extinction coefficient and the optical depth of the atmosphere between the sun and the ground, which itself depends on the angle of the sun.



GameDevelopers Conference

Parameters

- **Atmospheric parameters:**

  $\beta_{\mathrm{sc}}^{\mathrm{Rayleigh}} \quad \beta_{\mathrm{sc}}^{\mathrm{Mie}} \quad g_{\mathrm{HG}}$

- **Constant?** E_{sun}
  - **Affected by extinction**
- **Constant:** E_{sun}^{0}

make better games

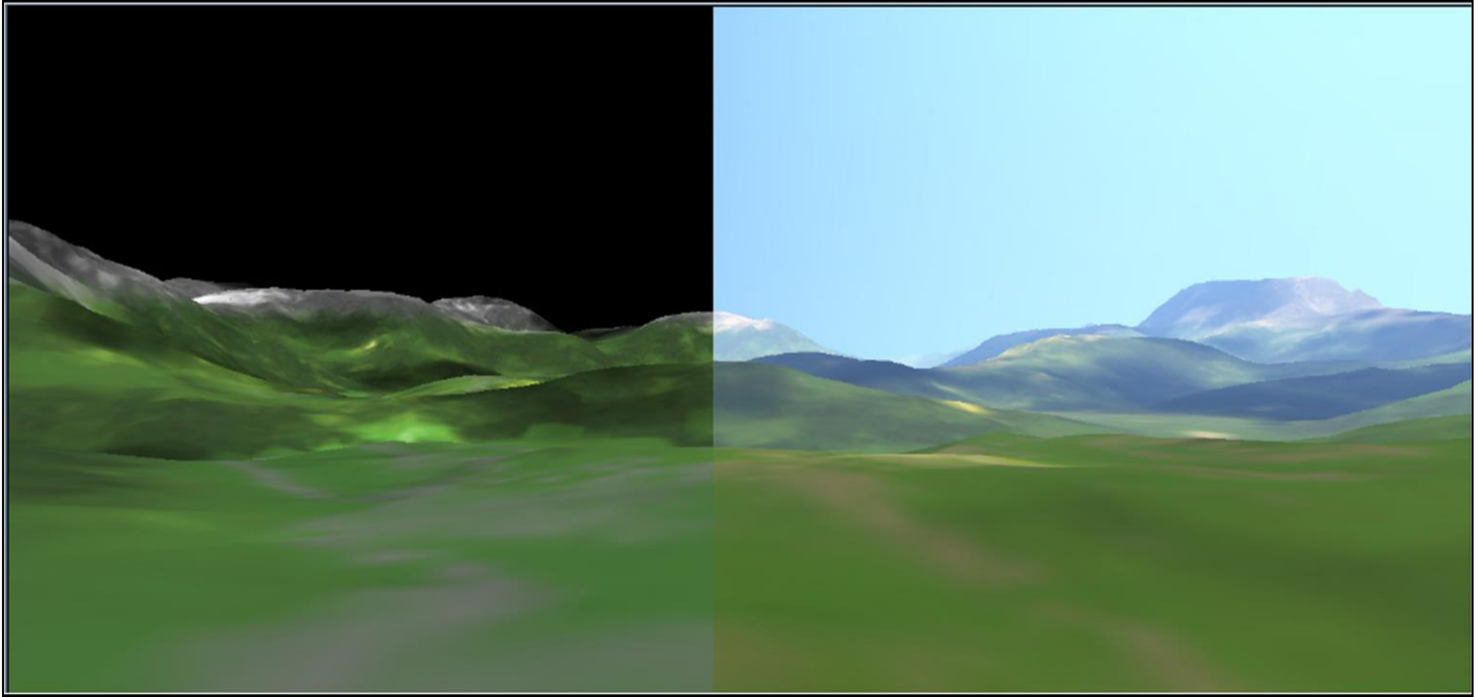
The value of Esun before extinction – Esun0 - is a constant, and represents the intensity of the sunlight in outer space before it enters the atmosphere.

Implementation

AJ Preetham

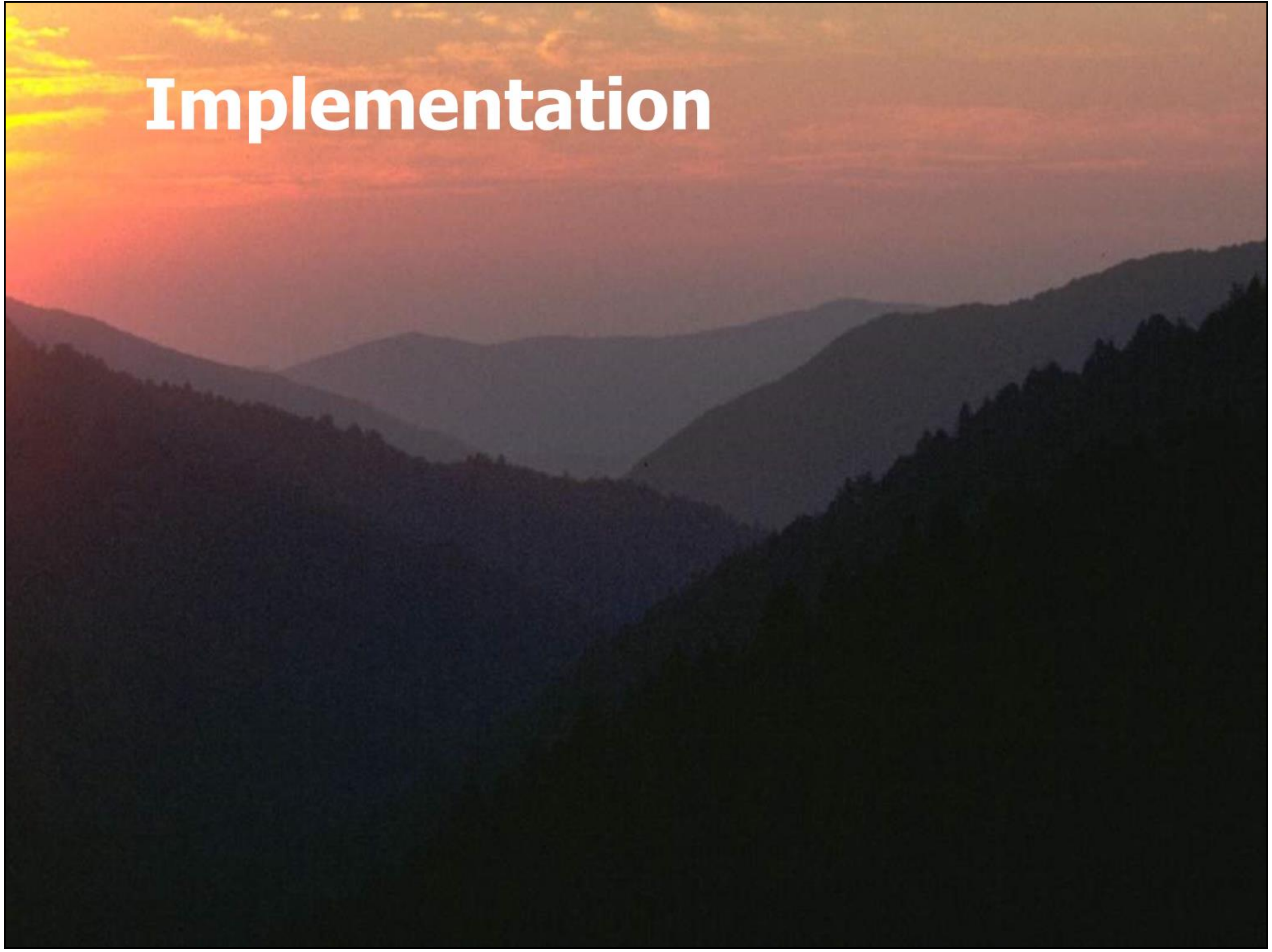
Implementation

How ?



Now that Naty has given you a good introduction on scattering theory, lets see how we can use this to create good-looking images. How do we take a scene and add scattering effects to it?

There are three issues involved in rendering this picture on the right: Aerial perspective, sky, and sunlight. Lets look at them one by one.



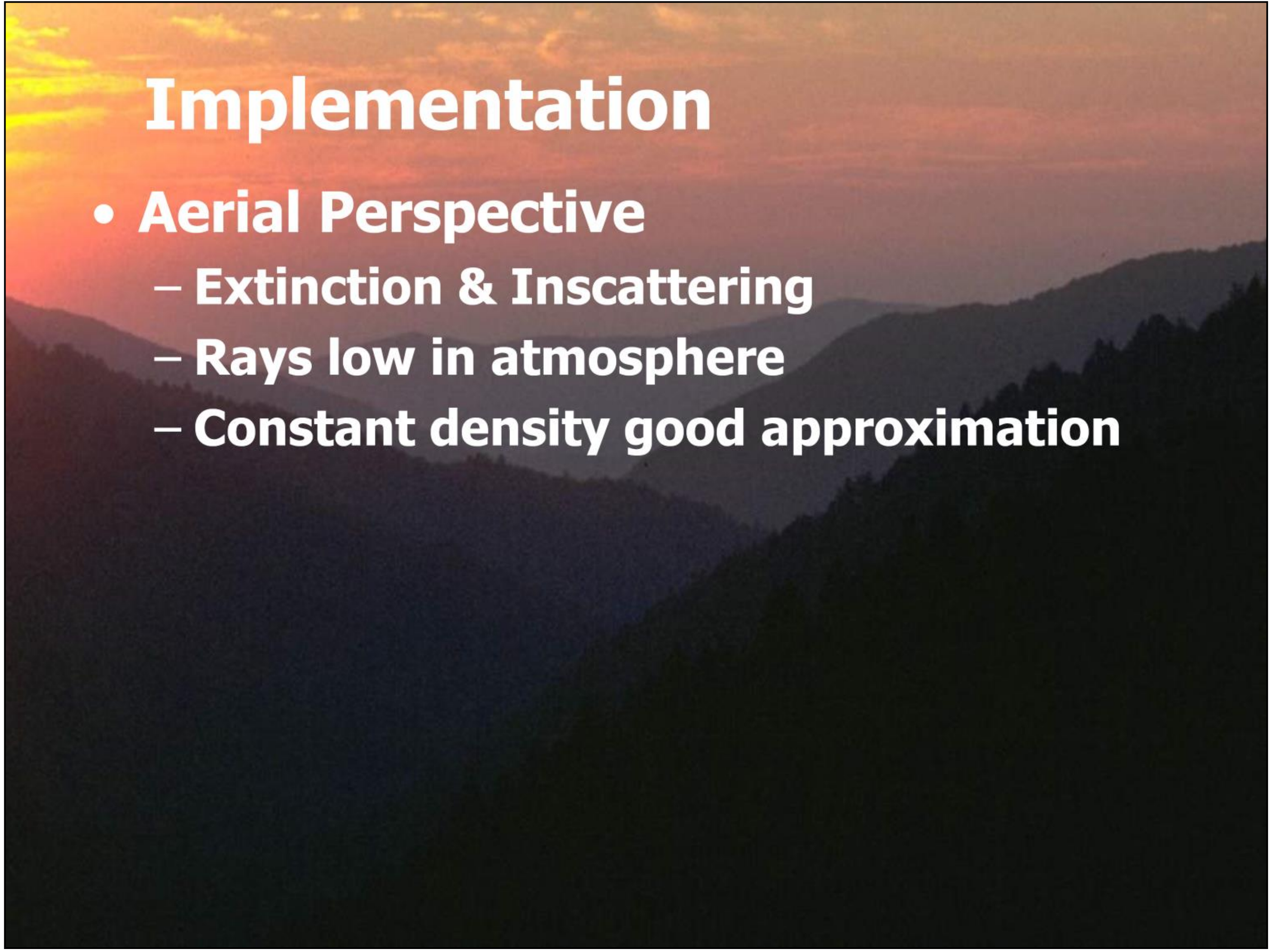
Implementation

Look at this picture and you will notice that the color changes with distance and gives us a visual cue that an object is far away.

Implementation

- Aerial Perspective
  - Extinction & Inscattering

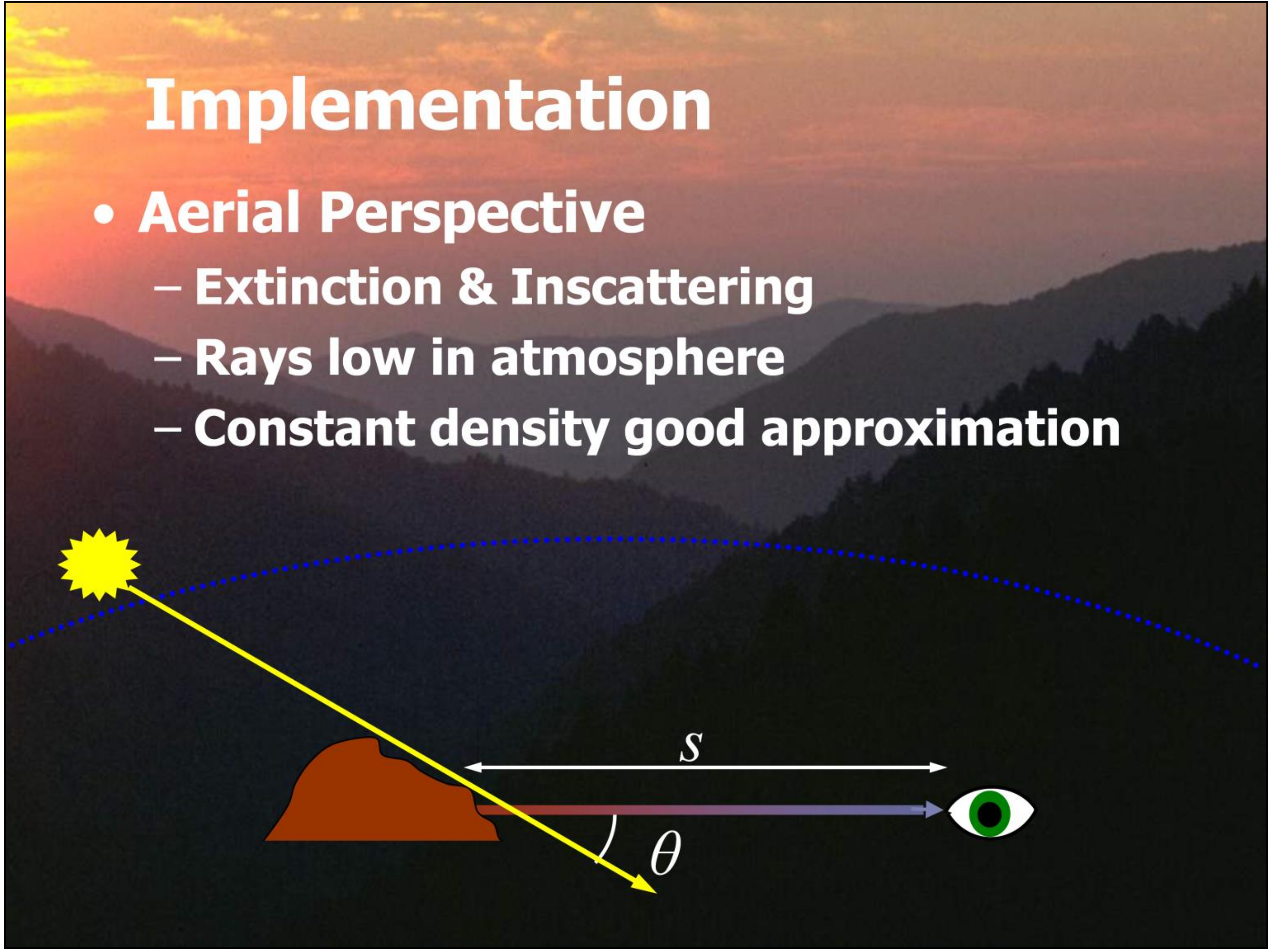
Aerial perspective is the change in color which gives us a visual cue that an object is far away. It is the result of both extinction and in-scattering.



Implementation

- Aerial Perspective
  - Extinction & Inscattering
  - Rays low in atmosphere
  - Constant density good approximation

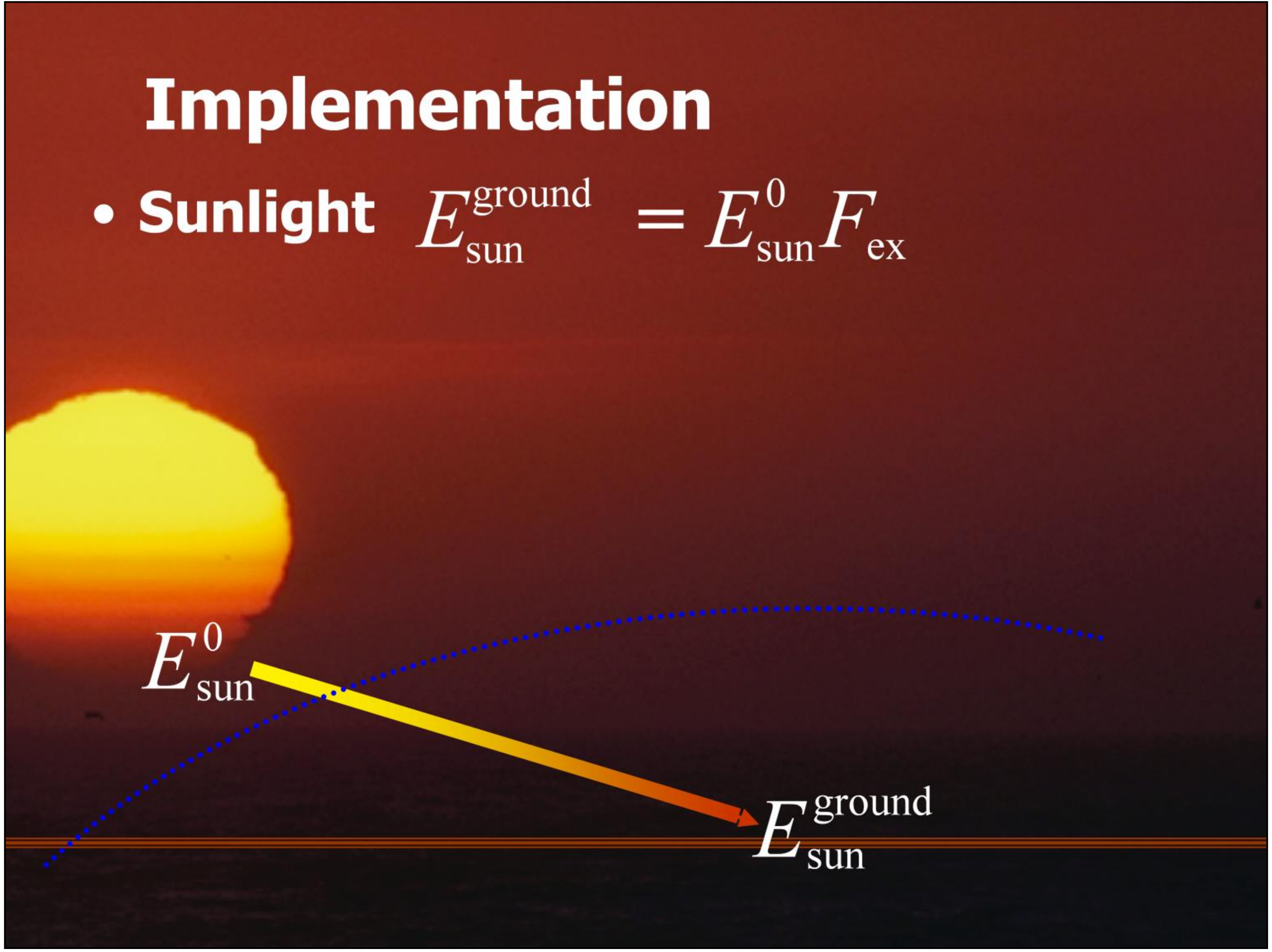
Our equations assume a constant density of particles in the atmosphere. In reality, the density of particles varies with elevation. However, for the case of aerial perspective, the eye position and the object being looked at are usually at a similar elevation, hence constant density is a good approximation.



Implementation

- Aerial Perspective
  - Extinction & Inscattering
  - Rays low in atmosphere
  - Constant density good approximation

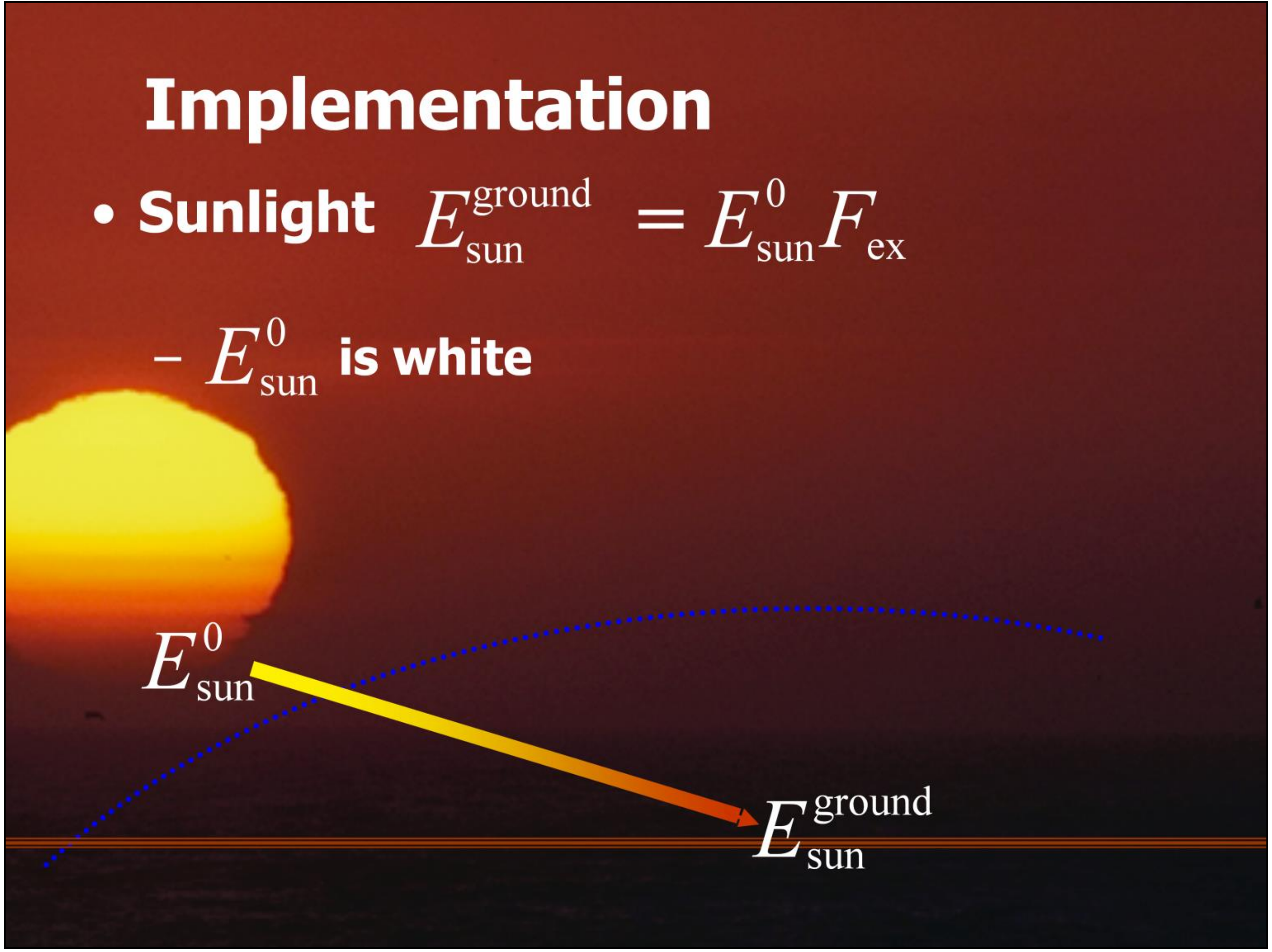
In this case we can simply plug the distance s and the angle theta into our equations, to get the extinction and in-scattering factors to apply to the color of the object for aerial perspective.



Implementation

- **Sunlight** $E_{\text{sun}}^{\text{ground}} = E_{\text{sun}}^{0} F_{\text{ex}}$

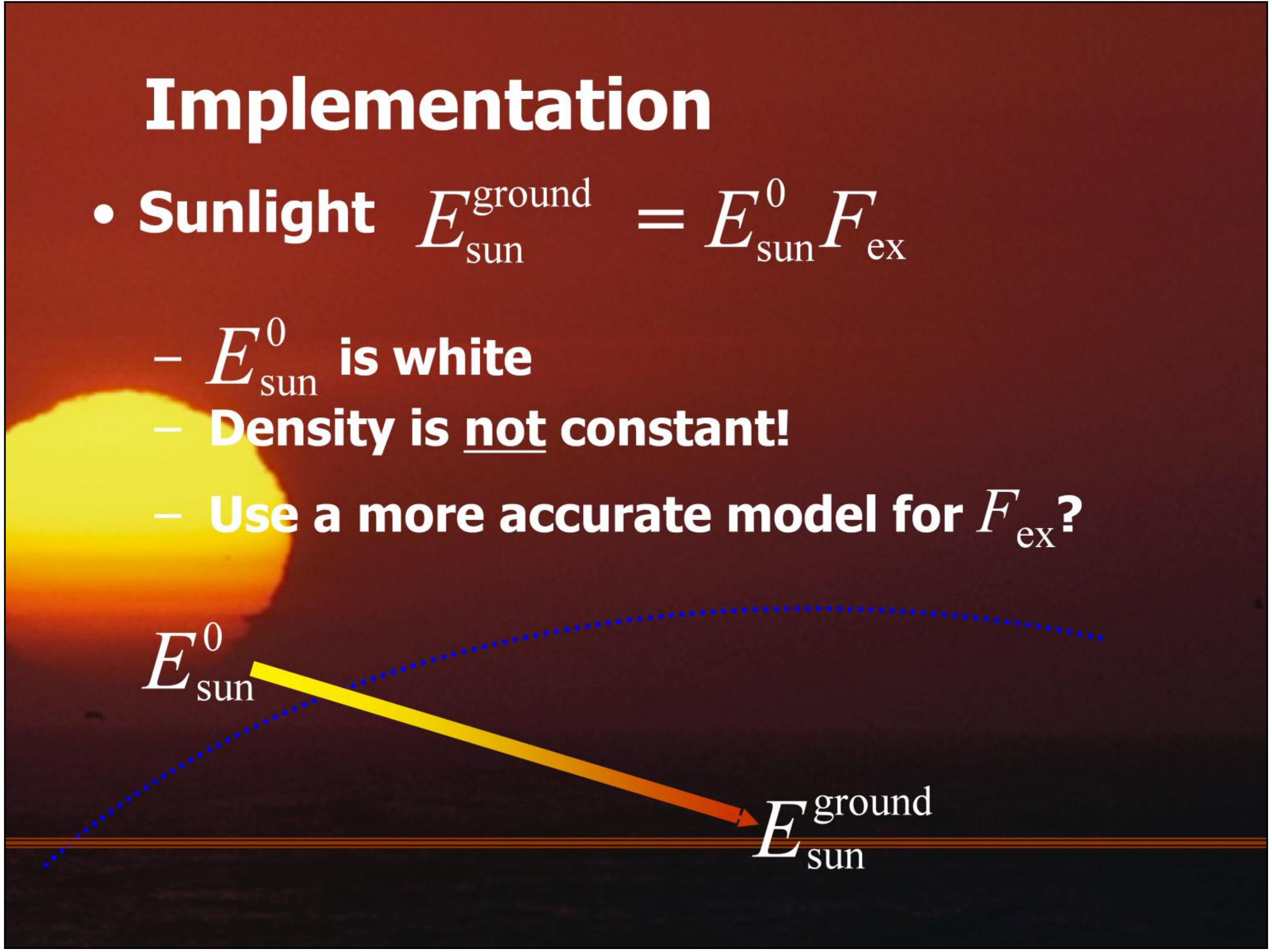
We need to calculate the sunlight received at ground level for in-scattering calculations (and lighting!). This is simply Esun0 (sun's intensity outside earth's atmosphere) times an extinction factor.



Implementation

- **Sunlight** $E_{\text{sun}}^{\text{ground}} = E_{\text{sun}}^{0} F_{\text{ex}}$
  - E_{sun}^{0} **is white**

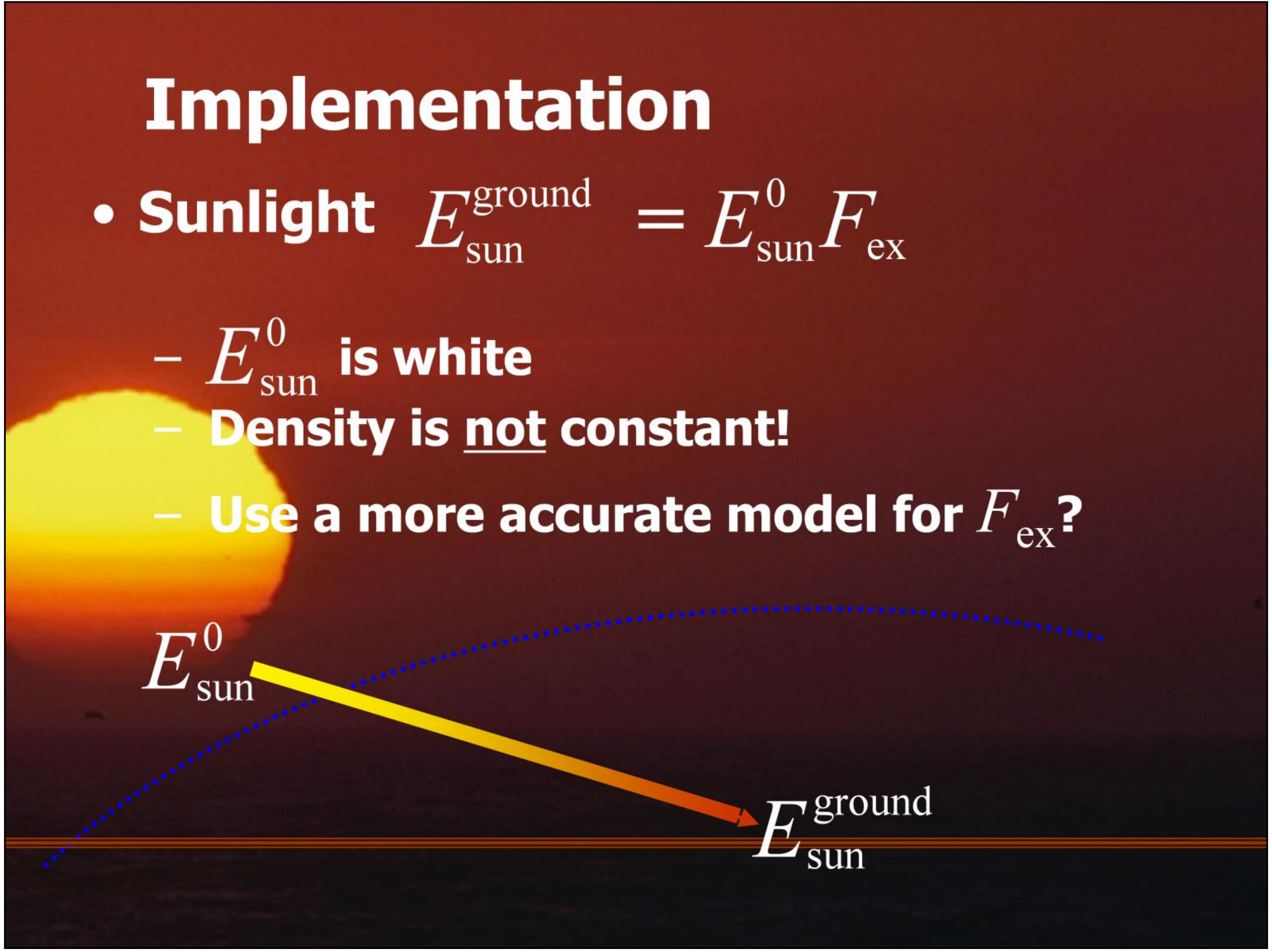
A good value to use for Esun0 is RGB triple (1,1,1). This value scales everything else, and represents an upper bound on sun illumination. The sun's spectrum outside the atmosphere is mostly white. The sun is called a yellow star, but the yellow tinge is barely noticeable to the eye, and is a result of extinction.



Implementation

- Sunlight $E_{\text{sun}}^{\text{ground}} = E_{\text{sun}}^{0} F_{\text{ex}}$
  - E_{sun}^{0} is white
  - Density is <u>not</u> constant!
  - Use a more accurate model for F_{ex}?

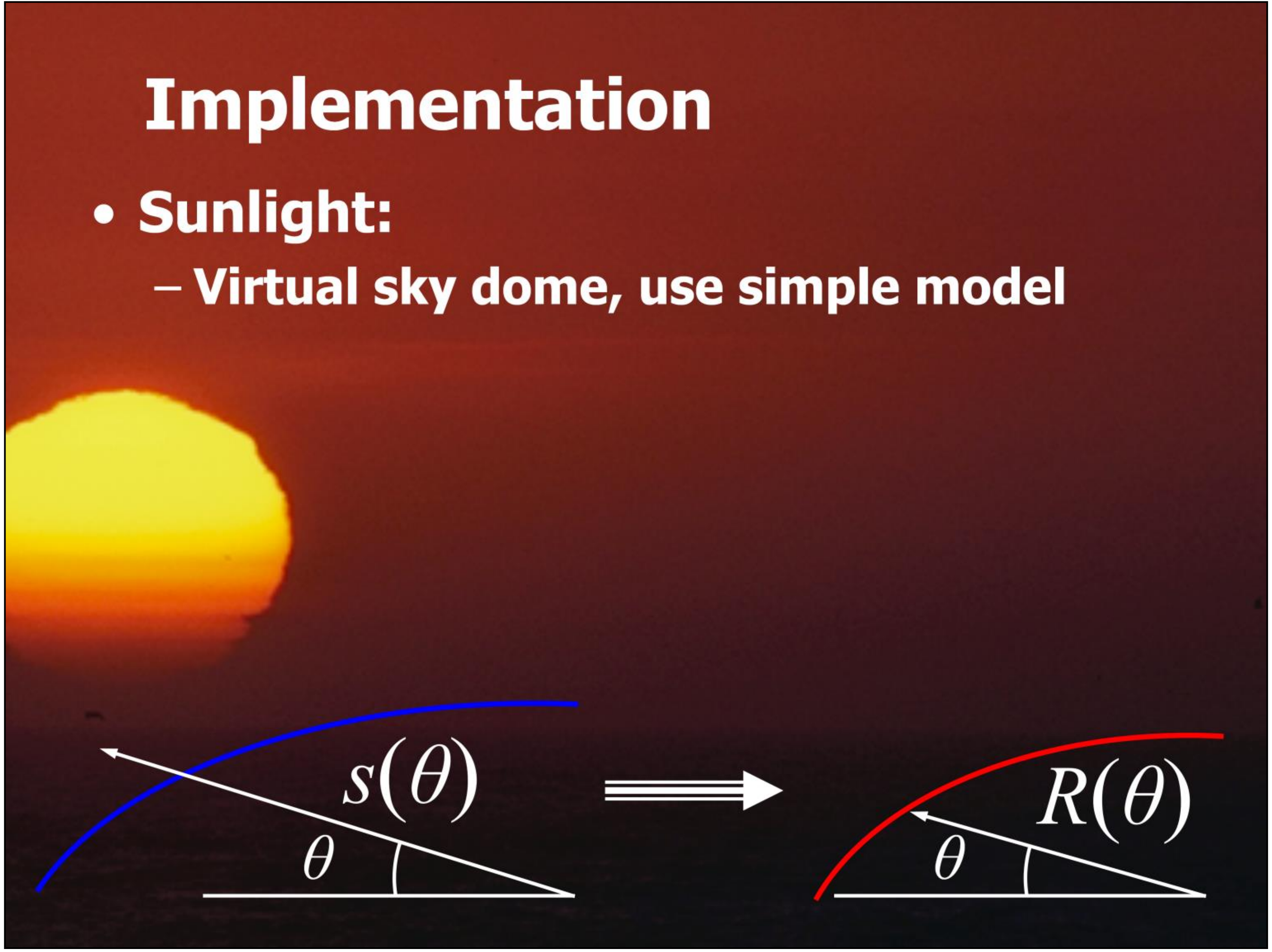
There is a problem however - our extinction equations assume constant density, and this assumption is most definitely not correct along the path traveled by the sunlight from outer space to the eye. One solution is to use a more complex, and accurate model to calculate extinction factor. We only need to do this calculation once per frame at most, and perhaps much less often than that because usually the sun does not move much from frame to frame. Such accurate models may be found in the references to this talk. This is what we do in our implementation.



Implementation

- **Sunlight** $E_{\text{sun}}^{\text{ground}} = E_{\text{sun}}^{0} F_{\text{ex}}$
  - E_{sun}^{0} **is white**
  - **Density is <u>not</u> constant!**
  - **Use a more accurate model for F_{ex}?**

This yields the most accurate results, which is important since Esun affects many other calculations. However, now we have two different models with different parameters, which complicates the implementation and might yield inconsistencies.

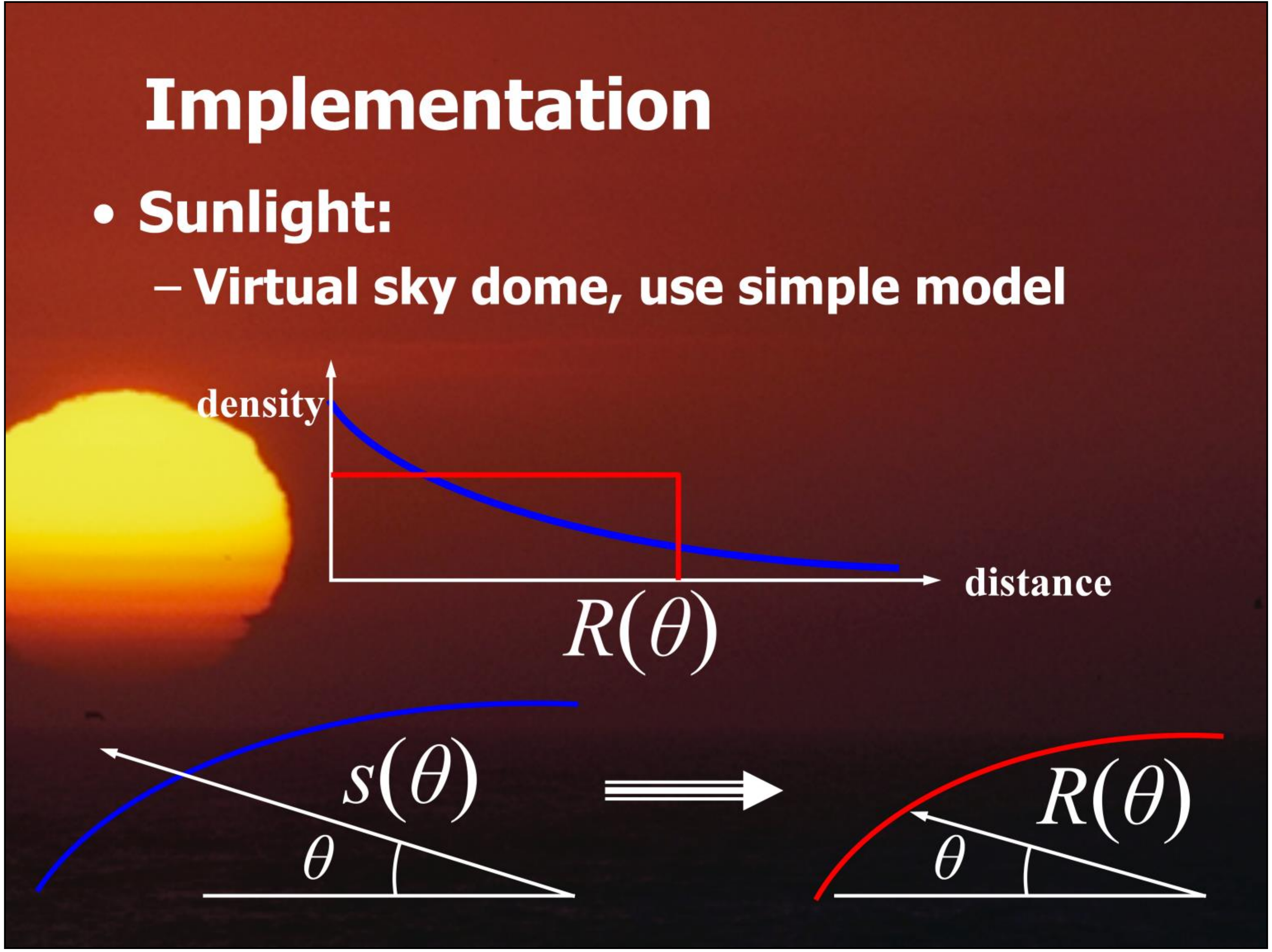


Implementation

- **Sunlight:**
  - **Virtual sky dome, use simple model**

$s(\theta)$
θ
$R(\theta)$
θ

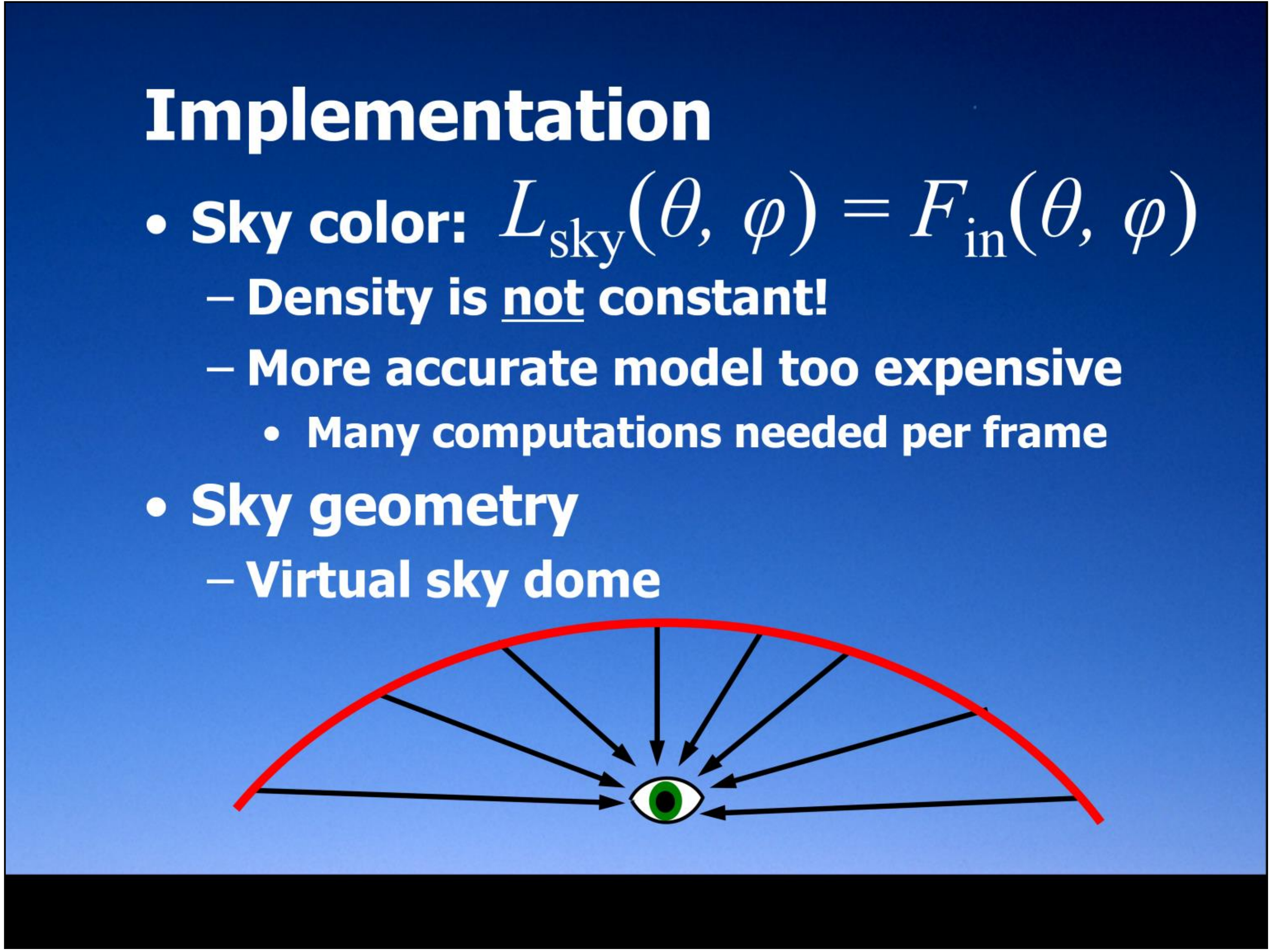
Another approach is to do a preprocessing step, where we define a virtual sky dome by integrating the density along various paths in a more complex and accurate sky density distribution. The resulting integrated density is used to define distances to the virtual sky dome in those directions. Each distance R is defined so that R times the constant density of our constant density model equals the integrated density from the real sky density distribution.



Implementation

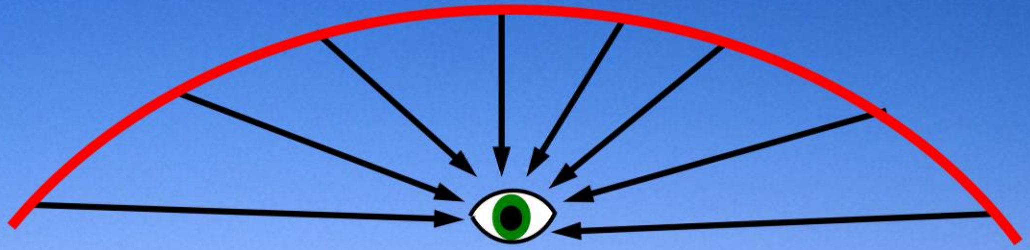
- **Sunlight:**
  - Virtual sky dome, use simple model

More intuitively: the blue line is the density vs. distance curve of the real atmosphere for given direction θ. The red line represents a constant-density atmosphere which ends abruptly at a distance R(θ), computed so that the area under both curves is the same. R(θ) can be pre-computed for various sample directions and a smooth surface fitted to the results; this is used at runtime to quickly generate a value of R for any θ that is then used to compute the sunlight extinction factor. This has the advantage that everything uses the same model, but is less accurate.

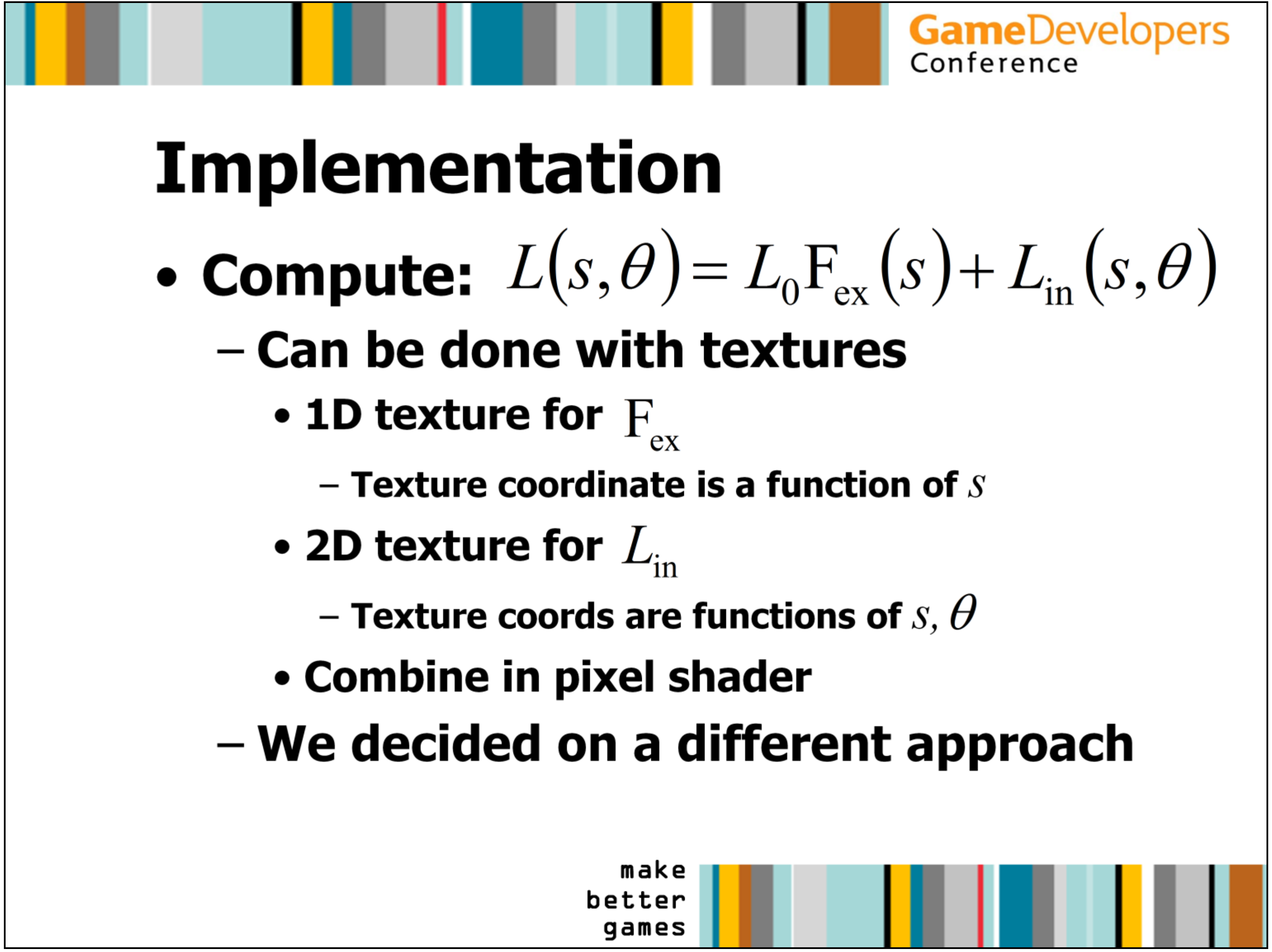


Implementation

- **Sky color:** $L_{\text{sky}}(\theta,\ \varphi) = F_{\text{in}}(\theta,\ \varphi)$
  - Density is <u>not</u> constant!
  - More accurate model too expensive
    - Many computations needed per frame
- **Sky geometry**
  - Virtual sky dome



Sunlight is scattered by the atmosphere multiple times in different directions and is received at the surface as skylight. The skylight color in each direction is the result of in-scattering from that direction. Similar to the sunlight case, the travel path extends all the way to the edge of the atmosphere. Applying an expensive model is less attractive in this case since we need to do the computation for many directions each frame. In this case, we render the mesh in the shape and size of the virtual sky dome and applying the in-scattering computations to it.

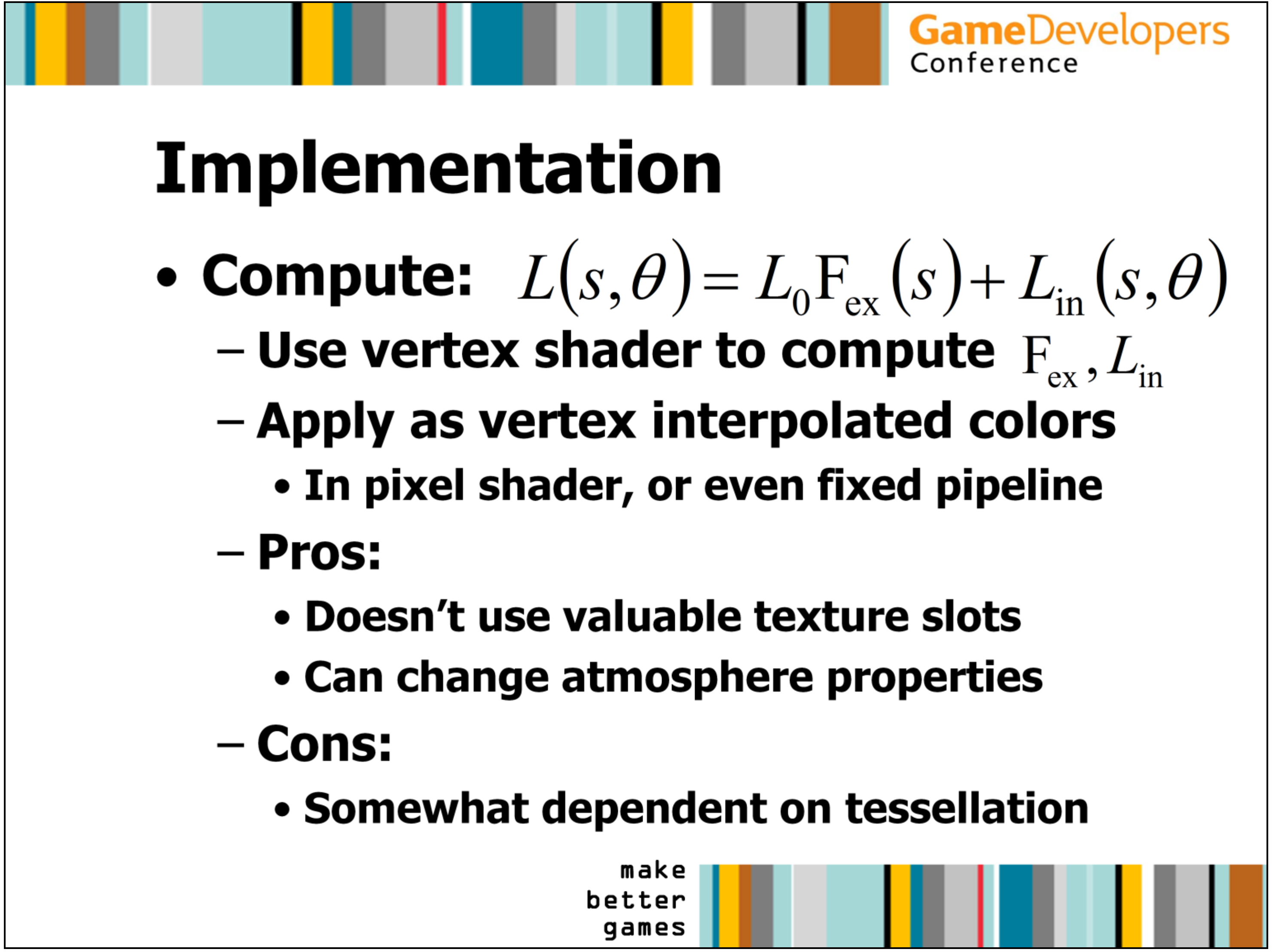

GameDevelopers Conference

Implementation

- **Compute:** $L(s,\theta) = L_0 \mathrm{F}_{\mathrm{ex}}(s) + L_{\mathrm{in}}(s,\theta)$
  - **Can be done with textures**
    - **1D texture for** F_{ex}
      - **Texture coordinate is a function of** s
    - **2D texture for** L_{in}
      - **Texture coords are functions of** s, θ
    - **Combine in pixel shader**
  - **We decided on a different approach**

make better games

The effective color of an object or sky at a distance s and scattering angle theta is the color of the object times extinction factor added to the inscattering. Since the extinction factor and the inscattering term are functions of s and theta, they could be represented as precomputed textures and the result can be obtained by combining them in a pixel shader. We decided to take another approach.


GameDevelopers Conference

Implementation

- **Compute:** $L(s,\theta) = L_0 F_{ex}(s) + L_{in}(s,\theta)$
  - **Use vertex shader to compute** F_{ex}, L_{in}
  - **Apply as vertex interpolated colors**
    - **In pixel shader, or even fixed pipeline**
  - **Pros:**
    - **Doesn't use valuable texture slots**
    - **Can change atmosphere properties**
  - **Cons:**
    - **Somewhat dependent on tessellation**

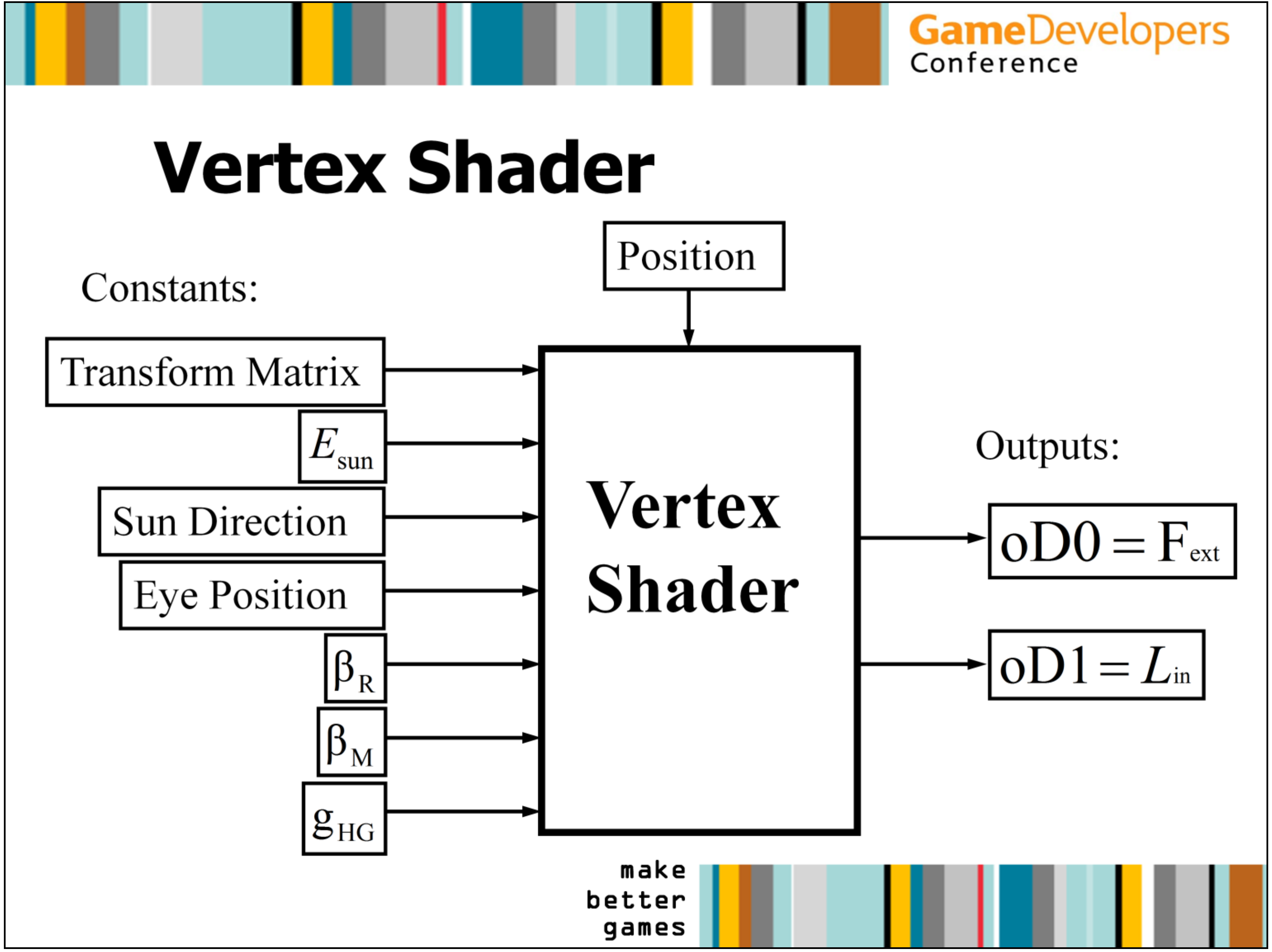
make better games

We decided to use a vertex shader instead to compute the extinction and inscattering term, and use the rasterizer to interpolate these values. Advantages:

(1) not use valuable texture slots

(2) can vary the atmospheric parameters and not bother about recreating textures.

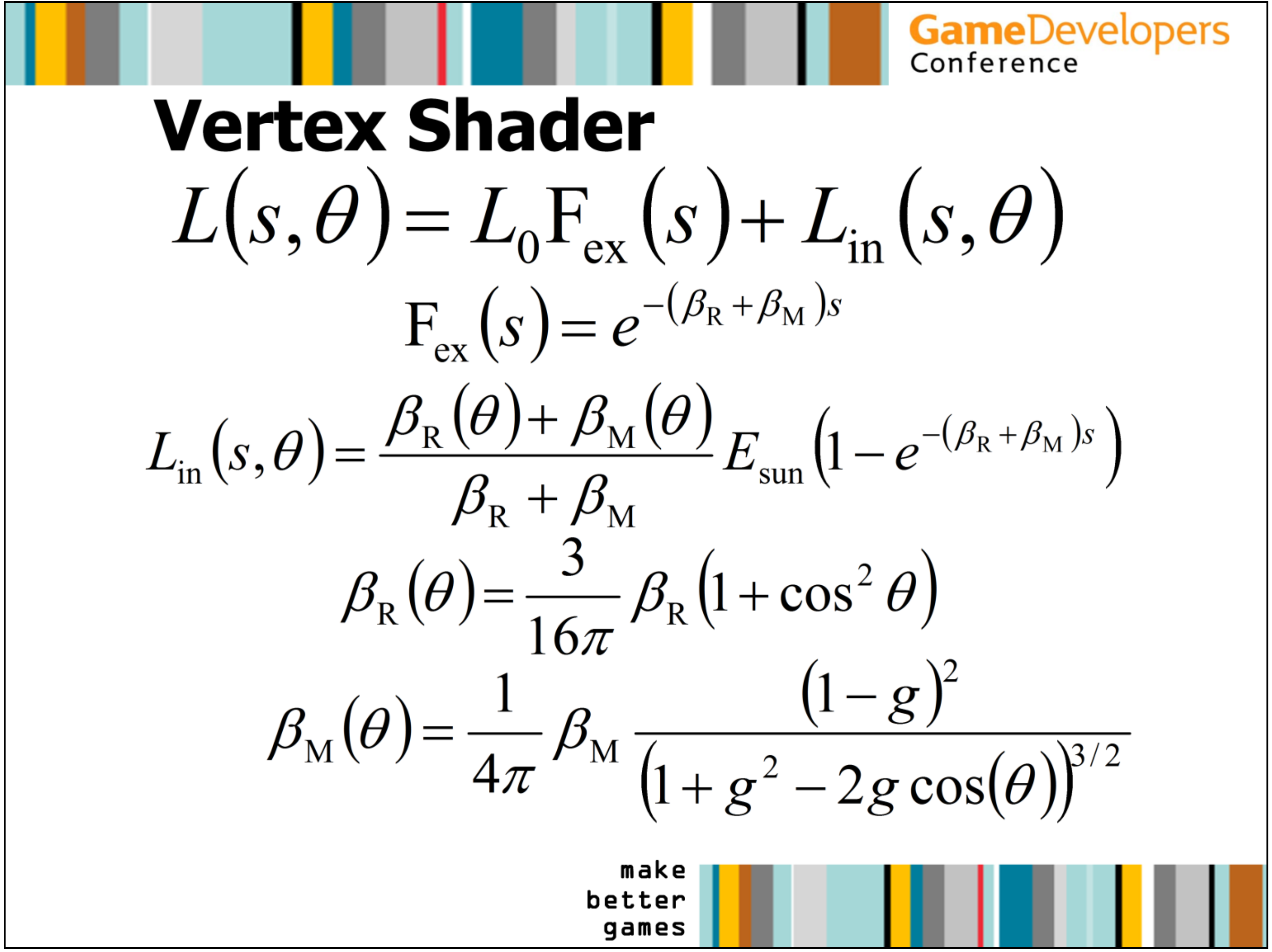
Disadvantage: since the functions are computed at the vertex level they are dependent on tessellation.



Vertex Shader

This block diagram is self explanatory. We have a whole bunch of constants that are sent to the vertex shader along with the position of the object.

The vertex shader computes the extinction and inscattering term from the math in the previous slide and writes them onto the color registers.


Vertex Shader

$$L(s,\theta) = L_0 \mathrm{F}_{\mathrm{ex}}(s) + L_{\mathrm{in}}(s,\theta)$$

$$\mathrm{F}_{\mathrm{ex}}(s) = e^{-(\beta_{\mathrm{R}} + \beta_{\mathrm{M}})s}$$

$$L_{\mathrm{in}}(s,\theta) = \frac{\beta_{\mathrm{R}}(\theta) + \beta_{\mathrm{M}}(\theta)}{\beta_{\mathrm{R}} + \beta_{\mathrm{M}}} E_{\mathrm{sun}} \left(1 - e^{-(\beta_{\mathrm{R}} + \beta_{\mathrm{M}})s}\right)$$

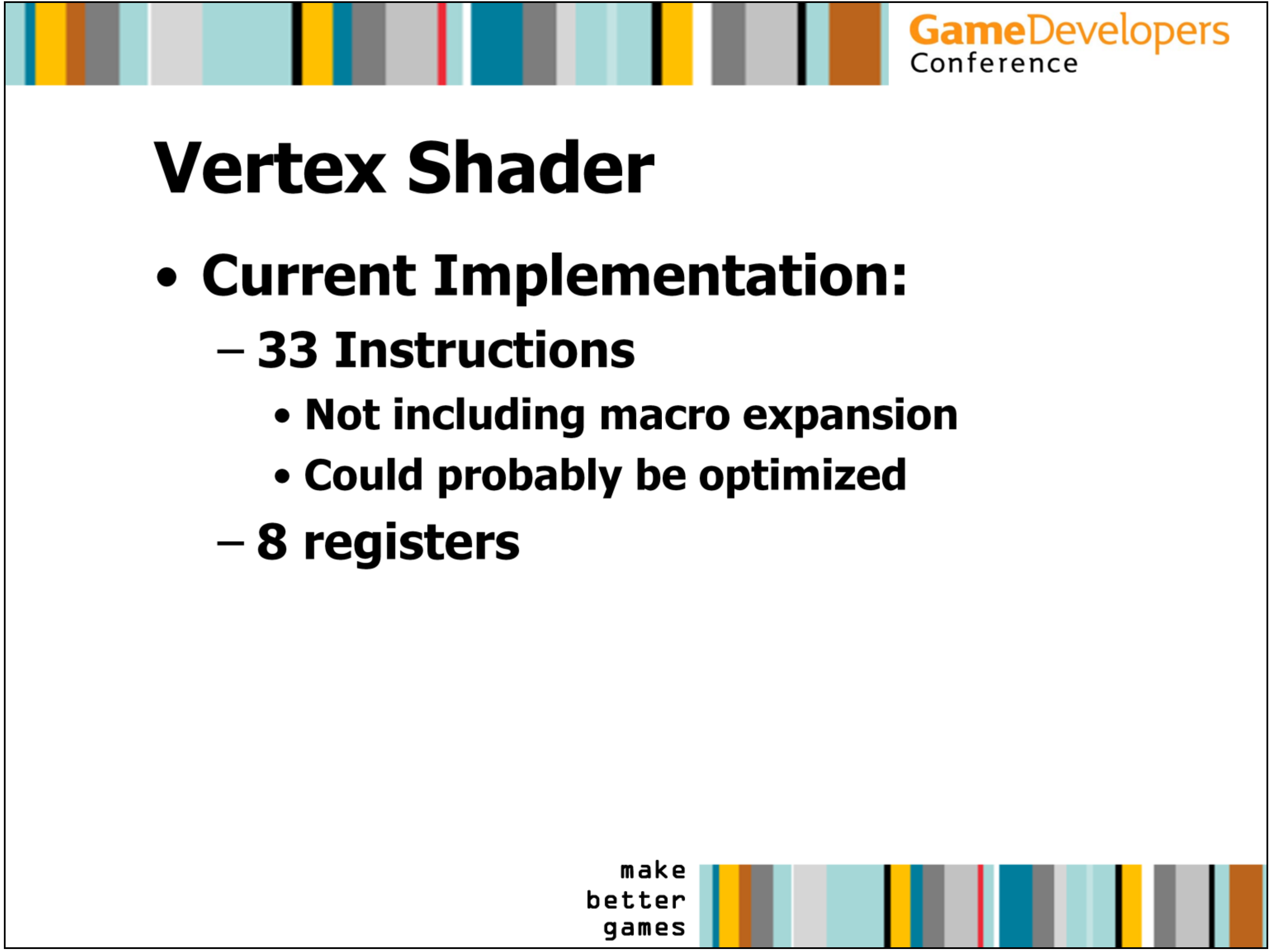
$$\beta_{\mathrm{R}}(\theta) = \frac{3}{16\pi} \beta_{\mathrm{R}} \left(1 + \cos^2\theta\right)$$

$$\beta_{\mathrm{M}}(\theta) = \frac{1}{4\pi} \beta_{\mathrm{M}} \frac{(1-g)^2}{\left(1 + g^2 - 2g\cos(\theta)\right)^{3/2}}$$

GameDevelopers Conference

make better games

All the math the vertex shader needs to do is on this page. The first equation is the scattering equation that gives the final color of an object in particle medium. The next two equations show how the extinction and inscattering terms are computed from scattering coefficients. Look at the paper for how these were arrived at. Also, note that the angular scattering coefficient depends on the cosine of the angle, which is simply a dot product. The only expensive operation is the exponential function.


GameDevelopers Conference

Vertex Shader

- **Current Implementation:**
  - 33 Instructions
    - Not including macro expansion
    - Could probably be optimized
  - 8 registers

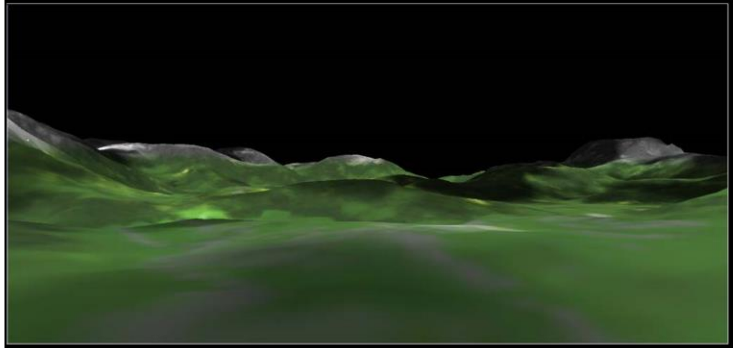
make better games

The current implementation is far from being fully optimized – basically we concentrated on functionality.

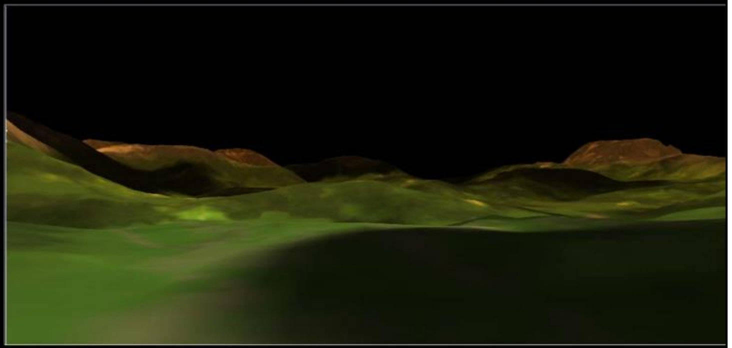
8 registers are used explicitly, the macros are probably using more.

Pixel Shader

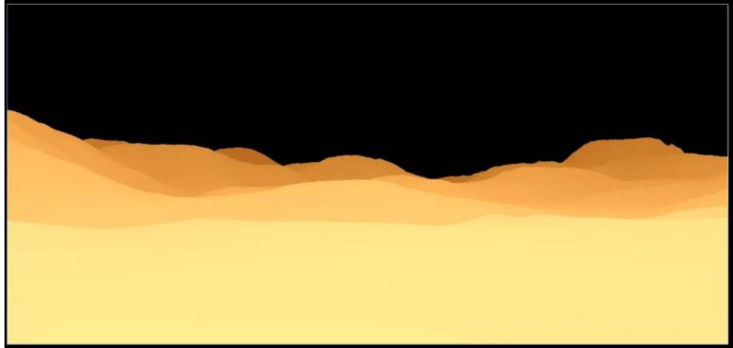
$$L = L_0 * \mathrm{F}_{ex} + L_{in}$$



L_0 $\times$ $=$



$L_0 * \mathrm{F}_{ex}$



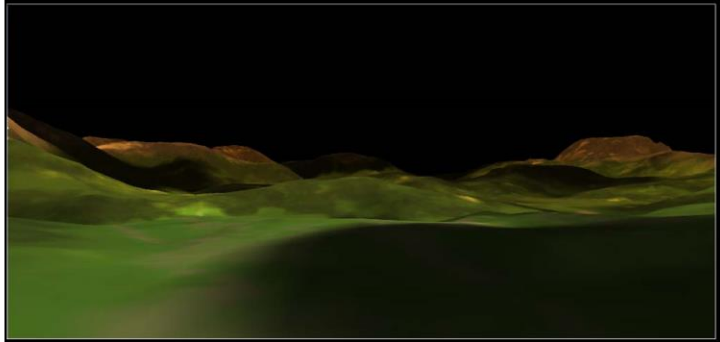
F_{ex}

This is a pictorial representation of the fundamental equation that the original radiance is attenuated by the extinction factor & added to the inscattering term.

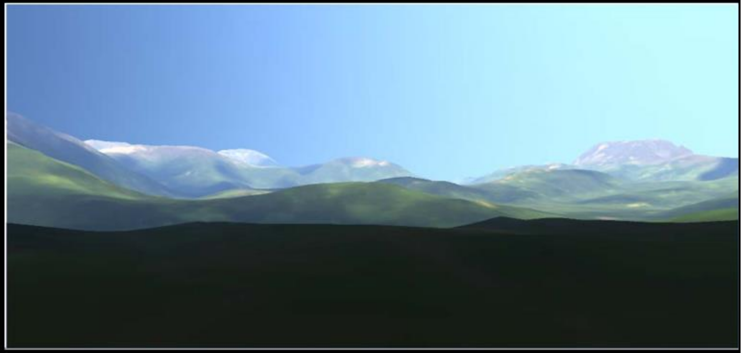
On the top left is the original color of the scene and on the bottom left is the extinction term. Note that extinction increases with distance. Also, the color is yellowish red, implying that the blue light is scattered most. On your right is the result of the original color times extinction.

Pixel Shader

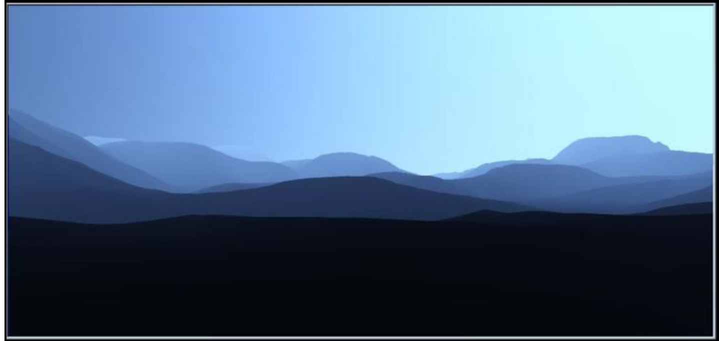
$$L = L_0 * \mathrm{F}_{ex} + L_{in}$$



$L_0 * \mathrm{F}_{ex}$ + =



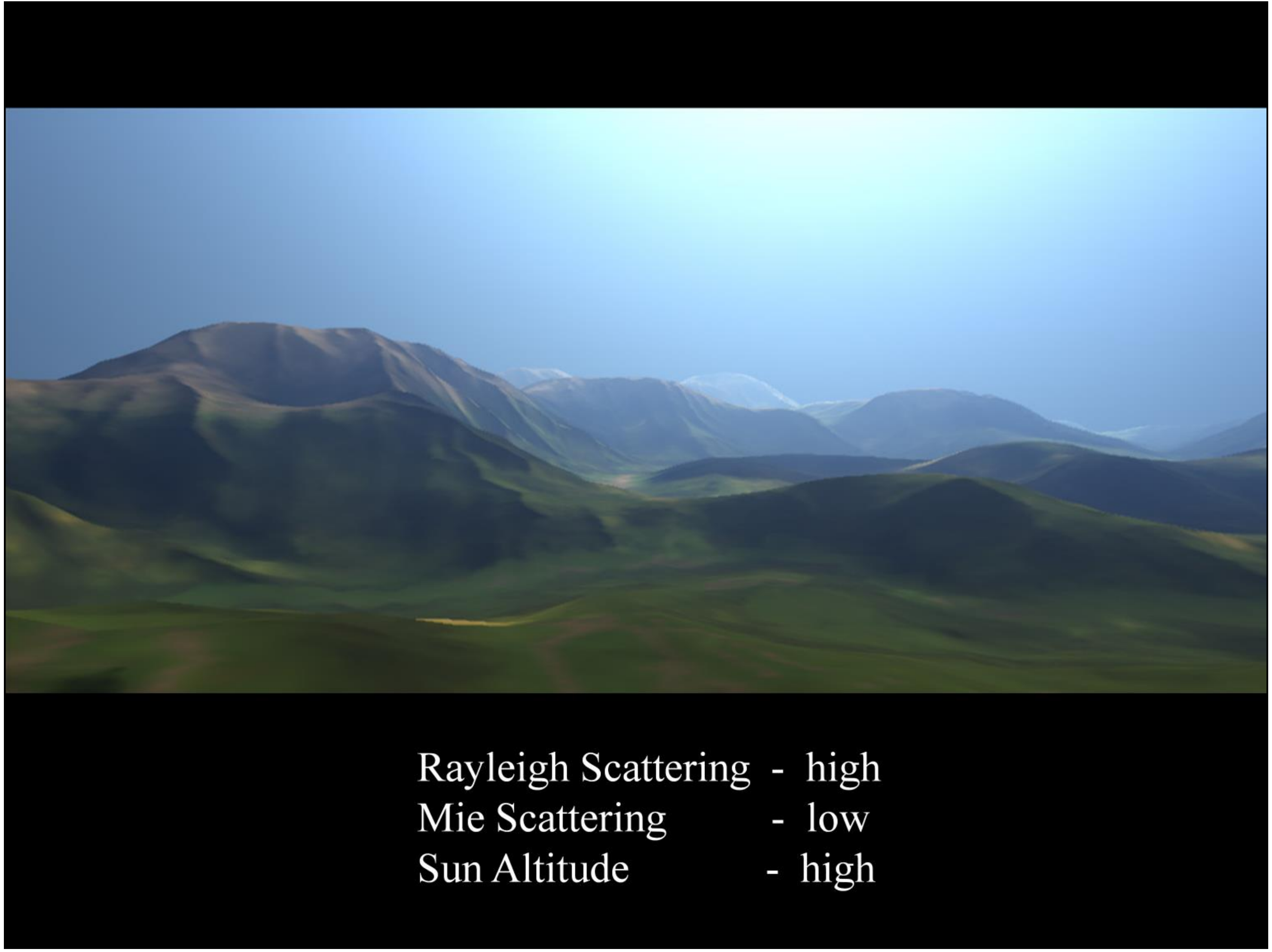
$$L = L_0 * \mathrm{F}_{ex} + L_{in}$$



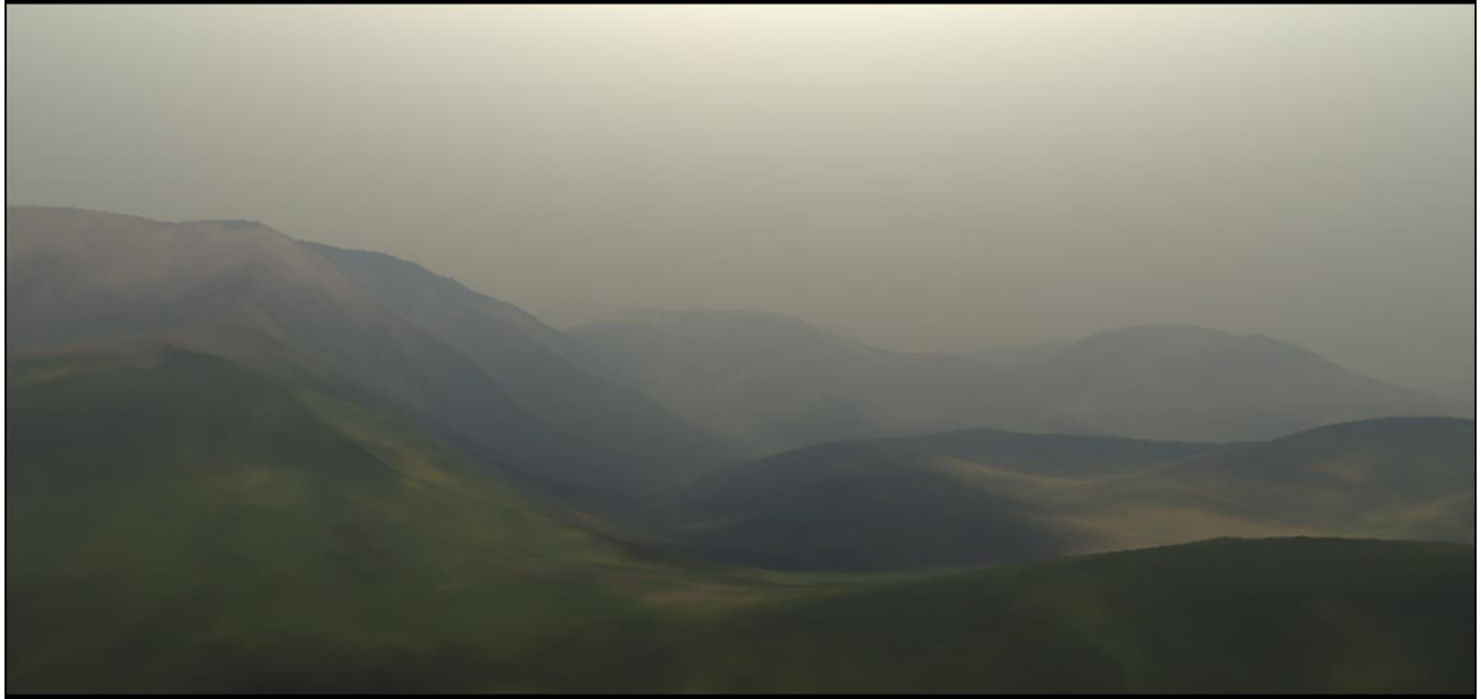
L_{in}

On the top left is the result from the previous computation. On the bottom left is the inscattering term. Notice that the amount of iscattering increases with depth and is also strongly blue in color which comes as no surprise because blue light is scattered the most. There is also a variation in the blue color depending on the orientation of the sun wrt to the viewing direction. You get the final result when you add them up both.

Results

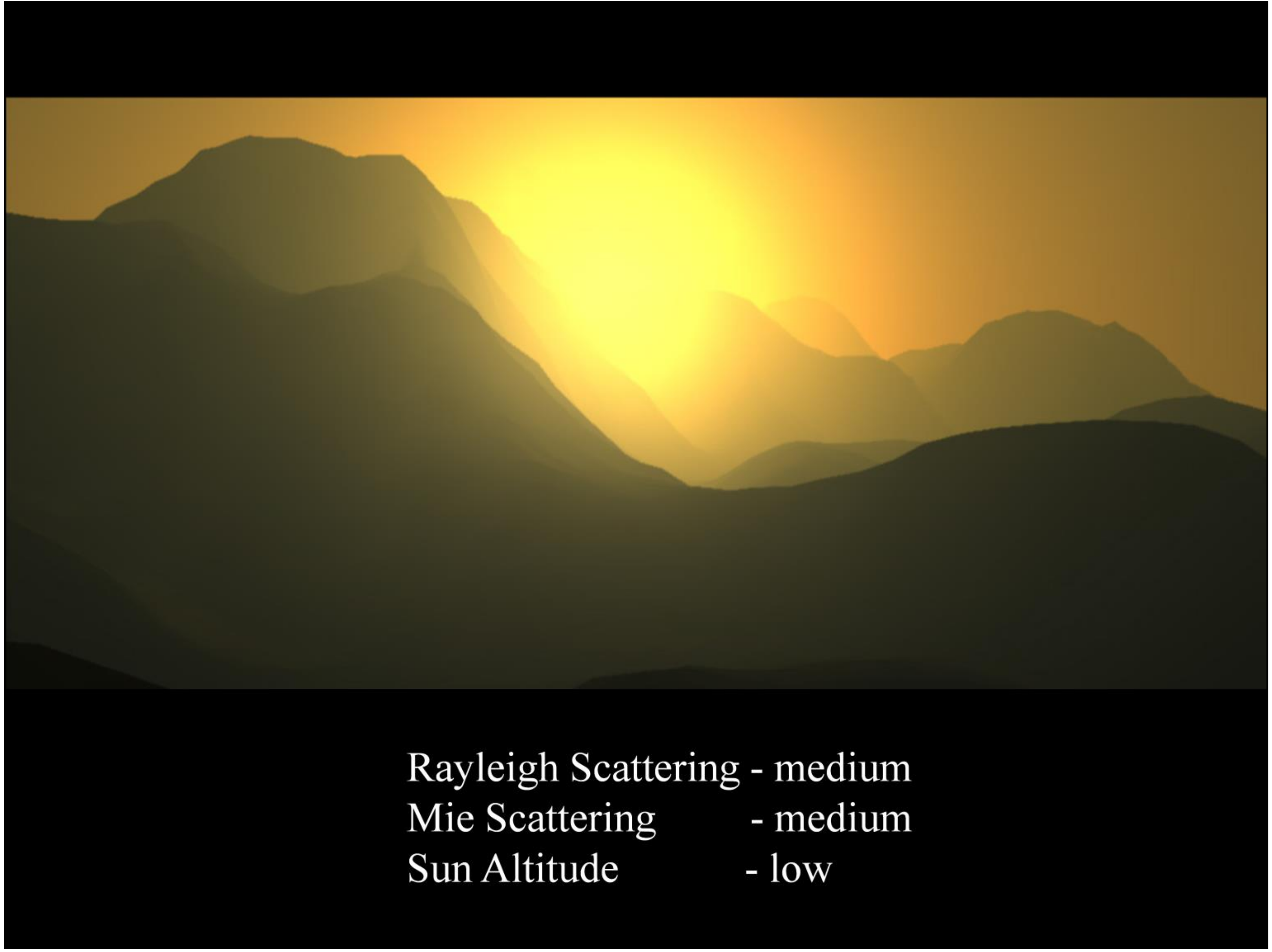

Rayleigh Scattering - high
Mie Scattering - low
Sun Altitude - high

This is a screenshot from our real-time demo.



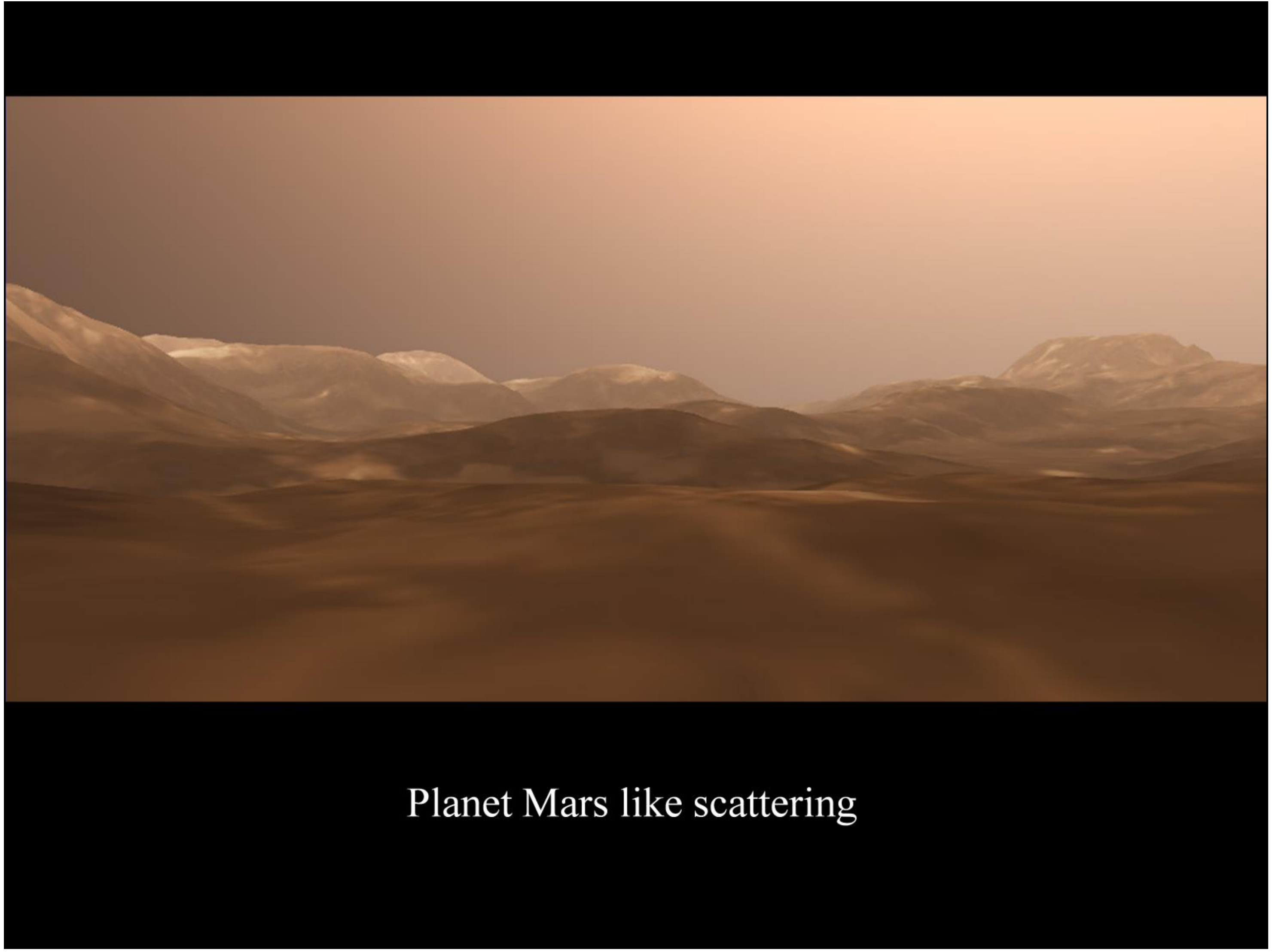
Rayleigh Scattering - low
Mie Scattering - high
Sun Altitude - high

You could simulate a very cloudy/overcast scene with high mie scattering coefficients.



When you have sun low in the picture, you get a strong forward scattering of yellow light, which gives you a sunset effect.

Notice that the shift towards yellow color depends on the orientation of the sun wrt to viewing direction.



Here we tried using Mars like scattering coefficients to generate this pinkish hue sky.

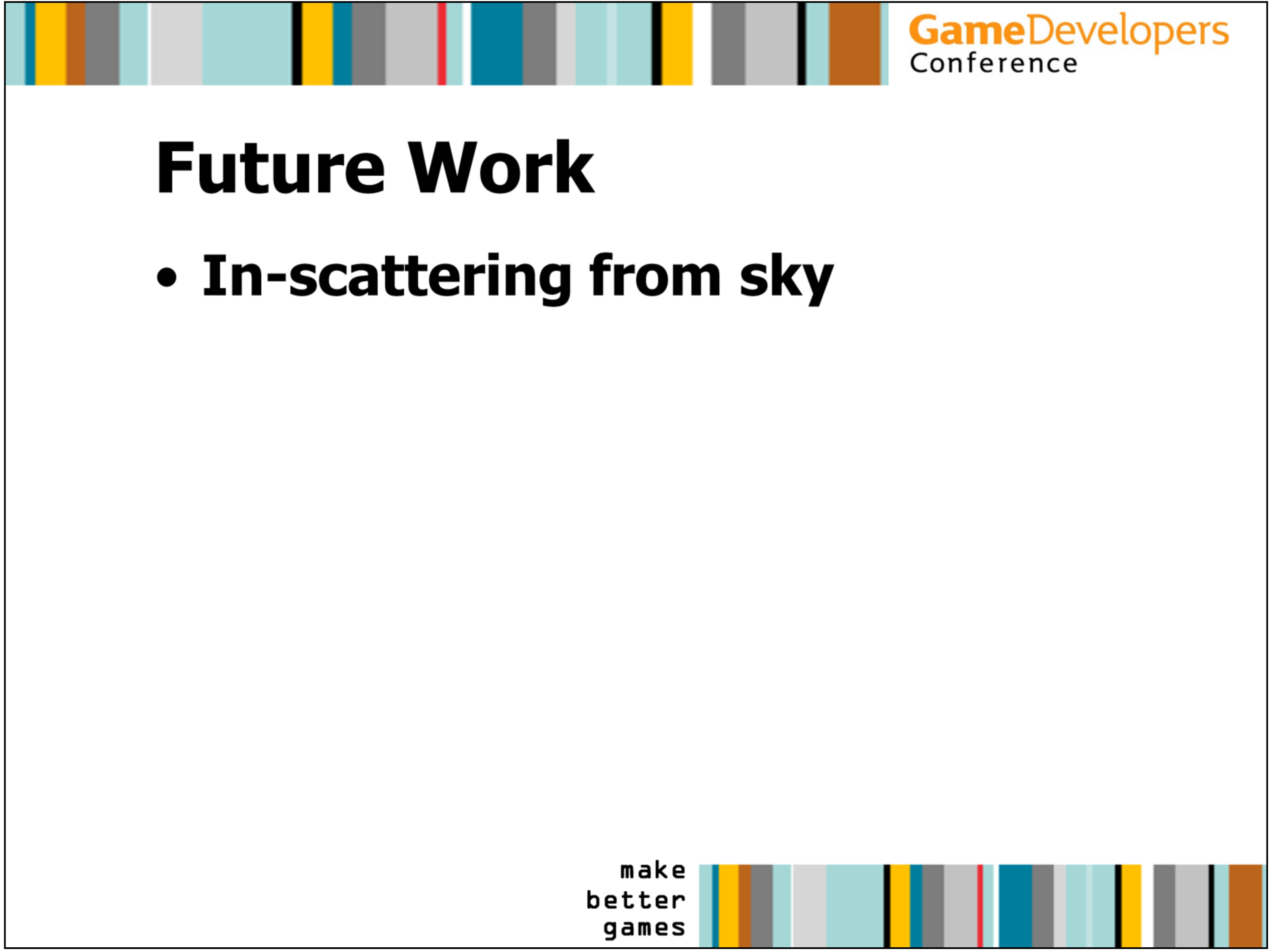
Demo

GameDevelopers Conference

Conclusion

- Scattering is easy to implement.
- Easy to add to an existing rendering framework
  - compute F_{ex} and L_{in}
  - apply these to existing color to get final color

make
better
games



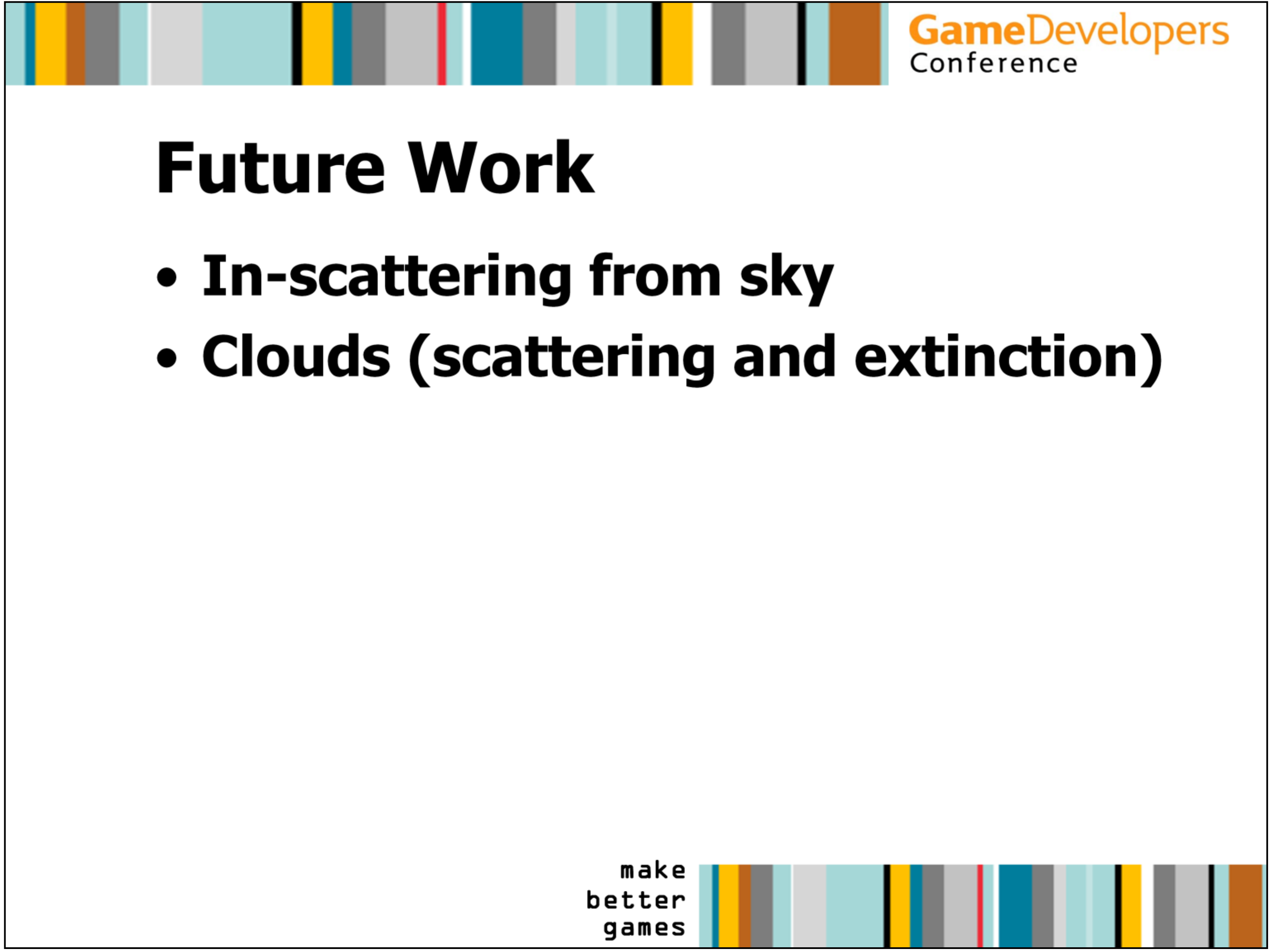
GameDevelopers Conference

Future Work

- In-scattering from sky

make better games

There is a lot of scope for scattering, now that the graphics pipeline is moving more towards a programmable model. In our current simulations, the lighting calculations on the terrain are not accurate because we have not accounted for the light coming from the sky. This is the light that illuminates shadowed regions.



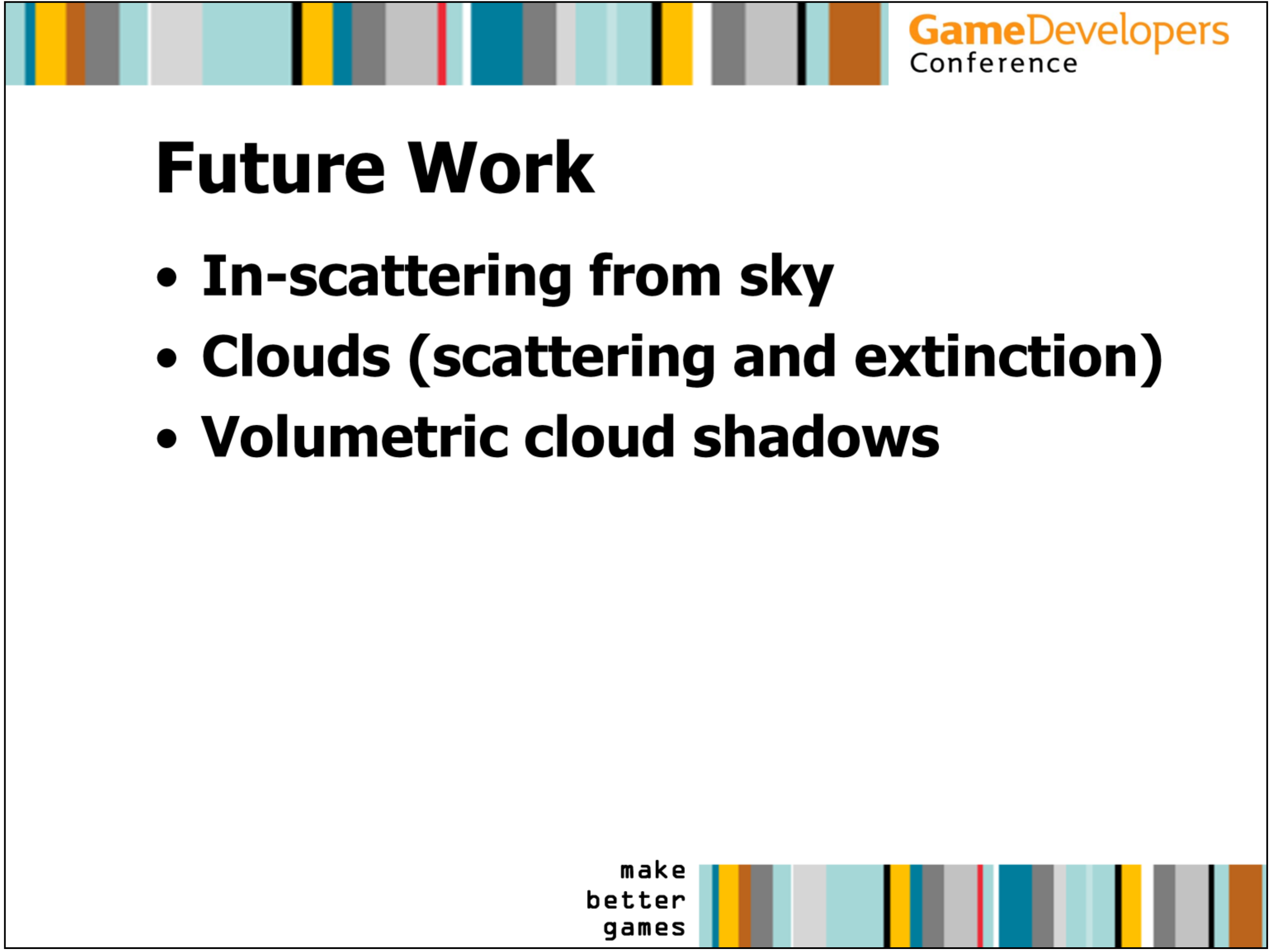
GameDevelopers Conference

Future Work

- In-scattering from sky
- Clouds (scattering and extinction)

make better games

Mark Harris gave a lecture on rendering clouds, but it would be interesting to find out if it could be done using scattering theory in real time.

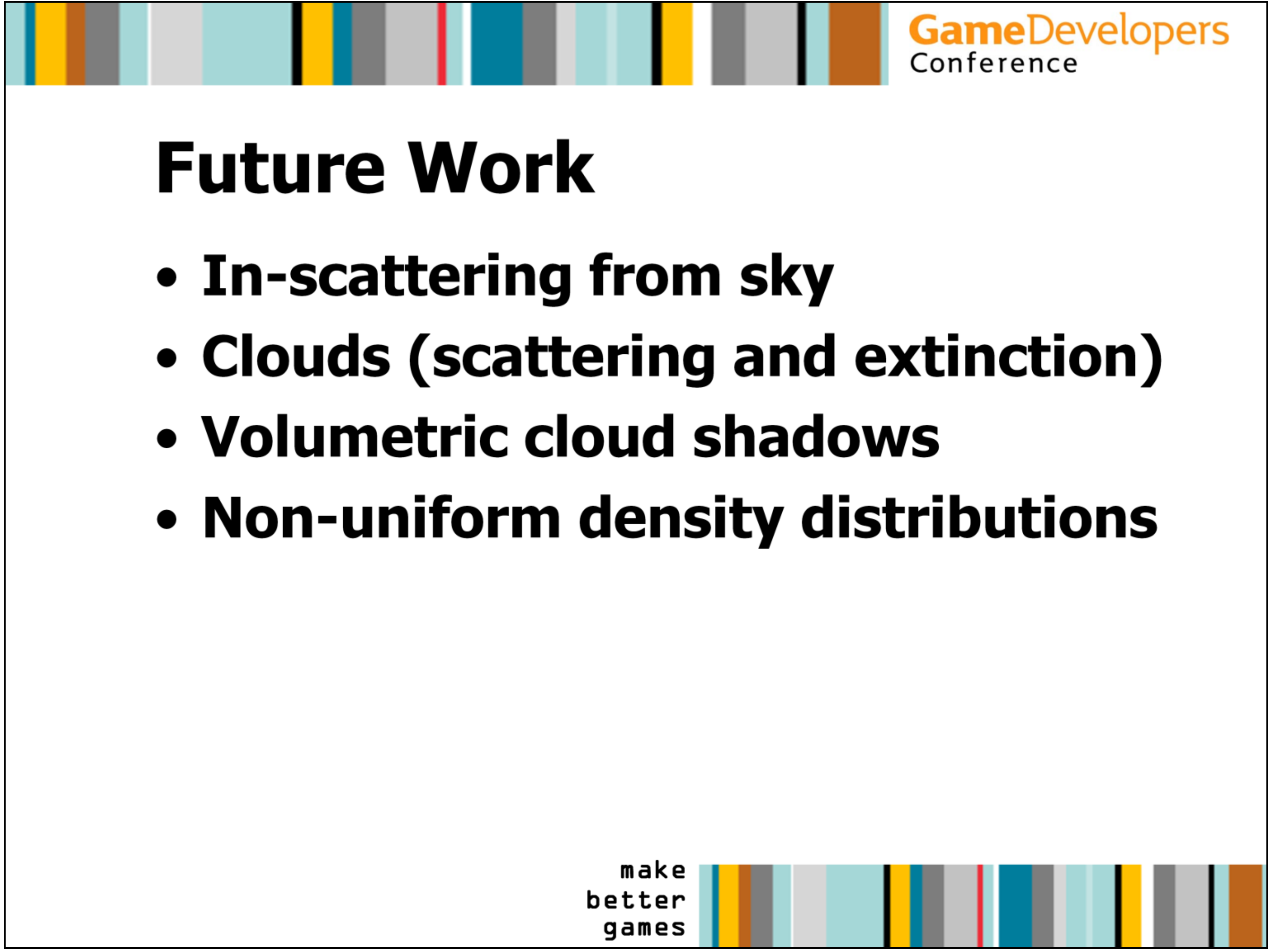

GameDevelopers Conference

Future Work

- In-scattering from sky
- Clouds (scattering and extinction)
- Volumetric cloud shadows

make better games

Volumetric cloud shadows is another interesting effect in atmosphere where rays of sunlight shine through patchy clouds.

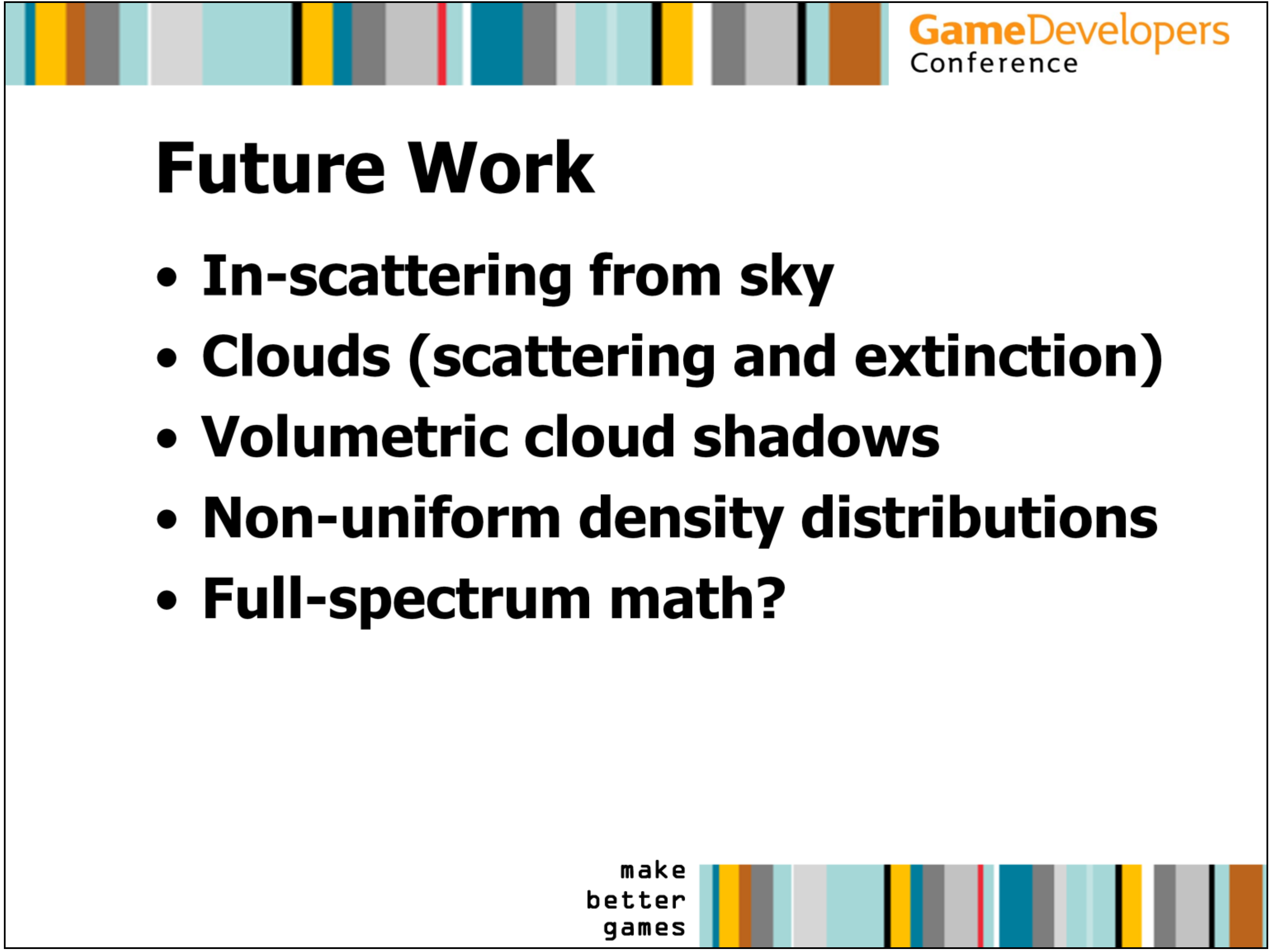


Future Work

- In-scattering from sky
- Clouds (scattering and extinction)
- Volumetric cloud shadows
- Non-uniform density distributions

GameDevelopers
Conference
make
better
games

Trying to do a more accurate density distribution.


GameDevelopers Conference

Future Work

- In-scattering from sky
- Clouds (scattering and extinction)
- Volumetric cloud shadows
- Non-uniform density distributions
- Full-spectrum math?

make better games

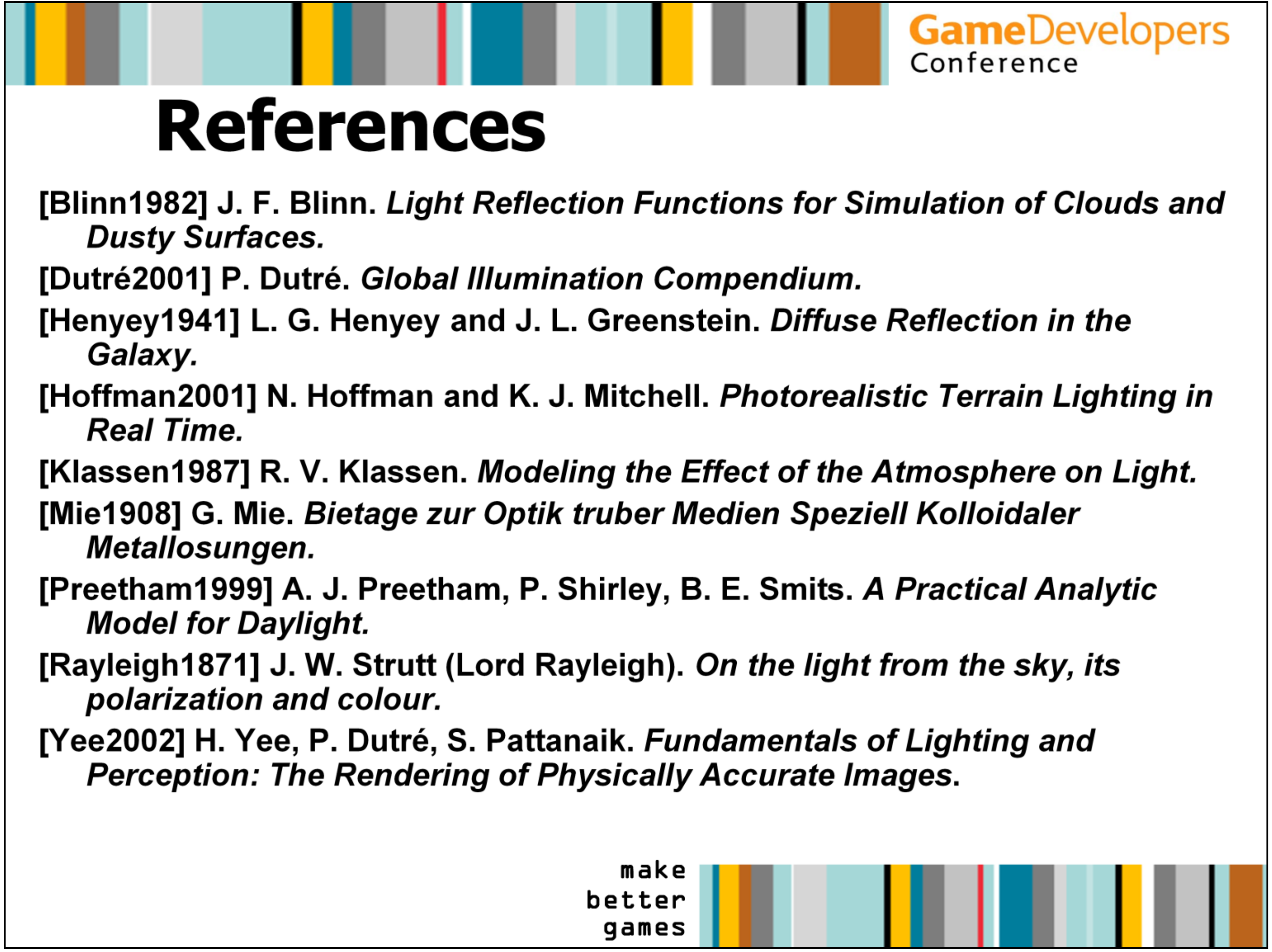
And explore the possibility of doing a more full spectrum math for more accuracy.

GameDevelopers Conference

Acknowledgements

- We would like to thank
  - Kenny Mitchell for the terrain engine used in our demo
  - Solomon Srinivasan for help with the demo movie

make better games


Game Developers Conference

References

[Blinn1982] J. F. Blinn. *Light Reflection Functions for Simulation of Clouds and Dusty Surfaces.*

[Dutré2001] P. Dutré. *Global Illumination Compendium.*

[Henyey1941] L. G. Henyey and J. L. Greenstein. *Diffuse Reflection in the Galaxy.*

[Hoffman2001] N. Hoffman and K. J. Mitchell. *Photorealistic Terrain Lighting in Real Time.*

[Klassen1987] R. V. Klassen. *Modeling the Effect of the Atmosphere on Light.*

[Mie1908] G. Mie. *Bietage zur Optik truber Medien Speziell Kolloidaler Metallosungen.*

[Preetham1999] A. J. Preetham, P. Shirley, B. E. Smits. *A Practical Analytic Model for Daylight.*

[Rayleigh1871] J. W. Strutt (Lord Rayleigh). *On the light from the sky, its polarization and colour.*

[Yee2002] H. Yee, P. Dutré, S. Pattanaik. *Fundamentals of Lighting and Perception: The Rendering of Physically Accurate Images*.

make better games

The full references are in the proceedings paper.

GameDevelopers
Conference

THANK YOU

make
better
games