Trabajo Fin de Grado

Universidad Zaragoza
1542

Modelado y simulación del transporte de la luz en la atmósfera

Modeling and Simulation of Light Transport in the Atmosphere

Autor: Pedro Andrés Gavín Murillo

Director: Adrián Jarabo Torrijos

Graphics and Imaging Lab


NASA


VozPopuli


NASA

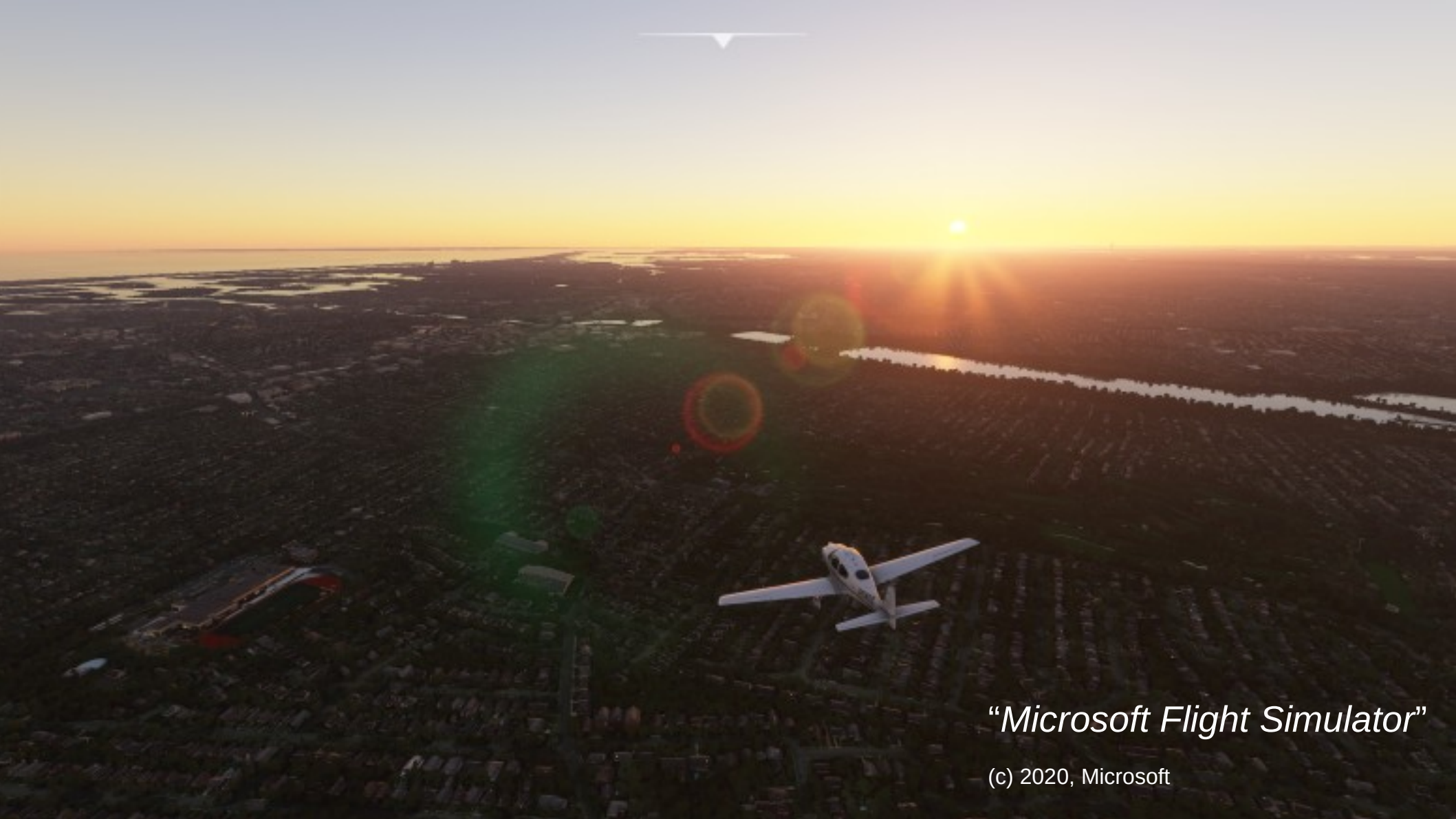
¿Por qué esto es importante?


“The Good Dinosaur”
(c) 2015, Pixar/ Disney

“*The Good Dinosaur*”


“*The Lion King*”
(c) 2019, Disney

“*Assassin's Creed: Valhalla*”



“*Microsoft Flight Simulator*”


NASA


Architectural Visualization
Image courtesy of Groupe Legendre

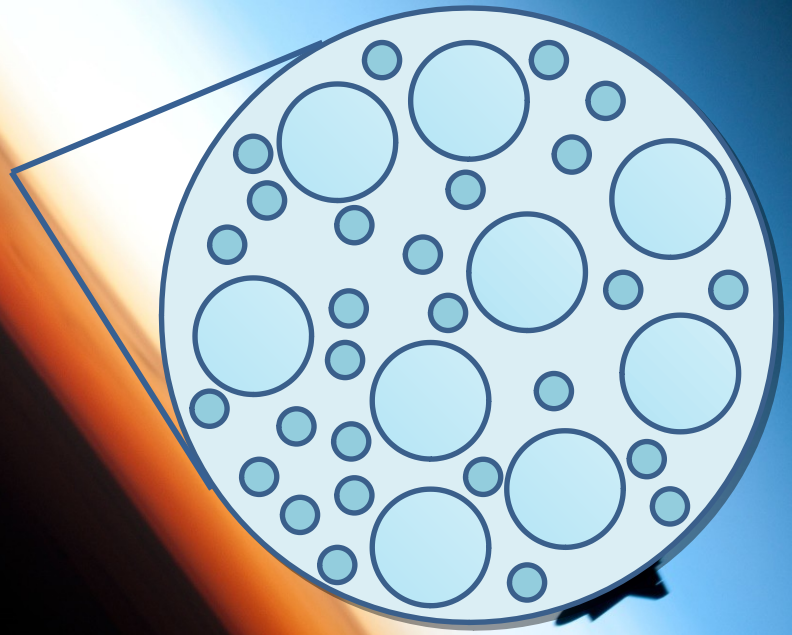
Architectural Visualization

Zaha Hadid Architects | ZHVR Group

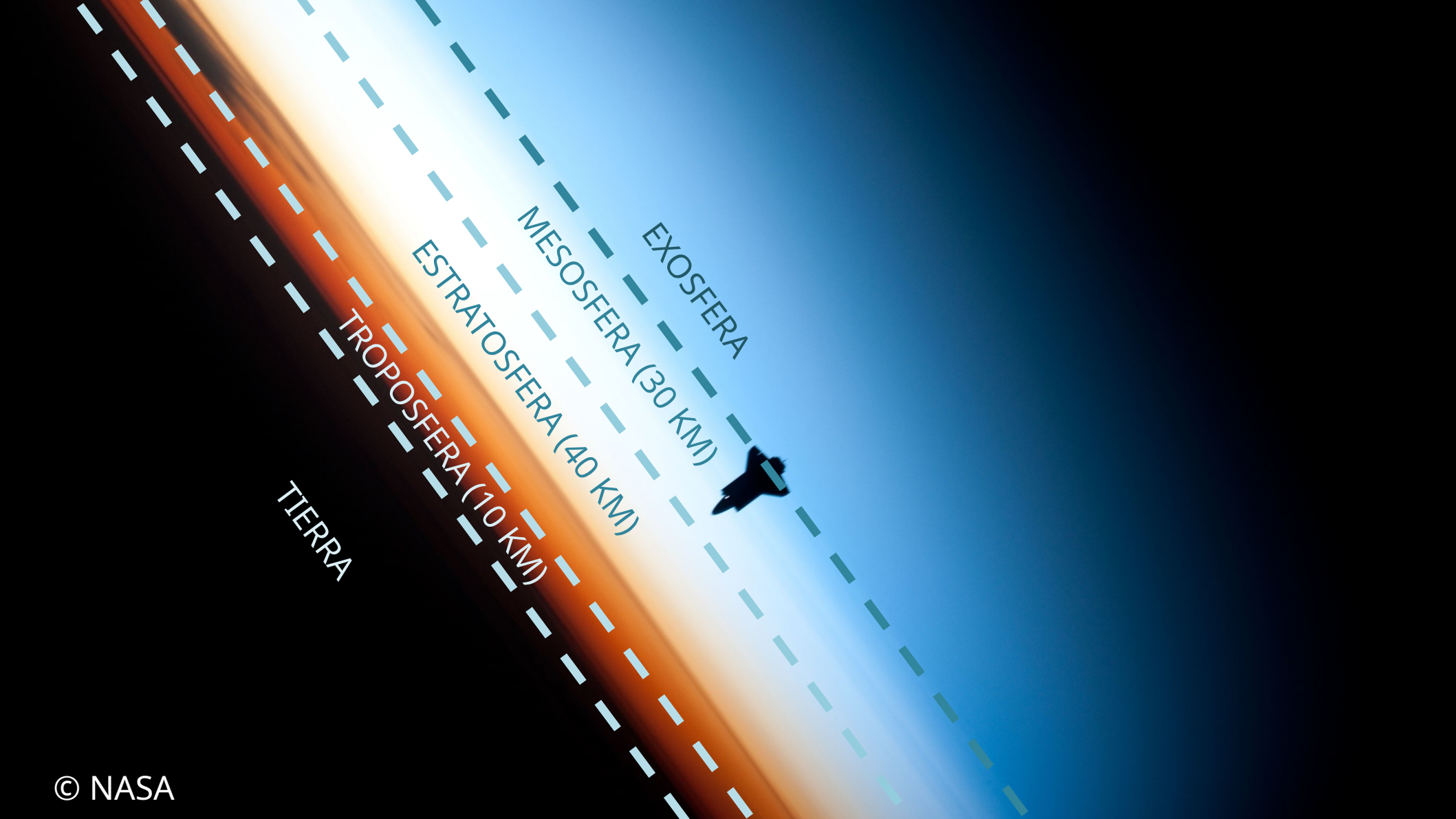
Architectural Visualization

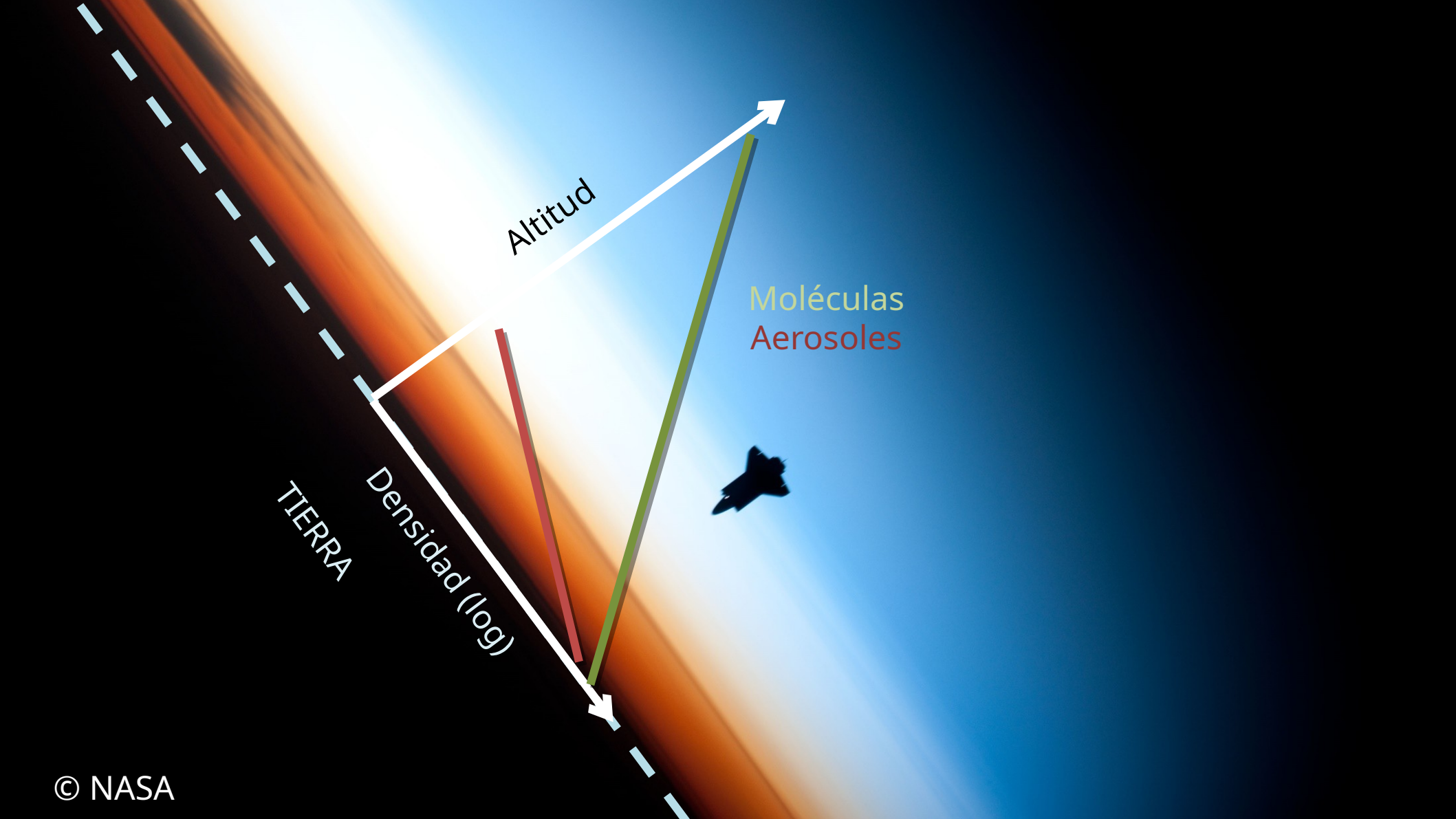
Rag3DVIZ

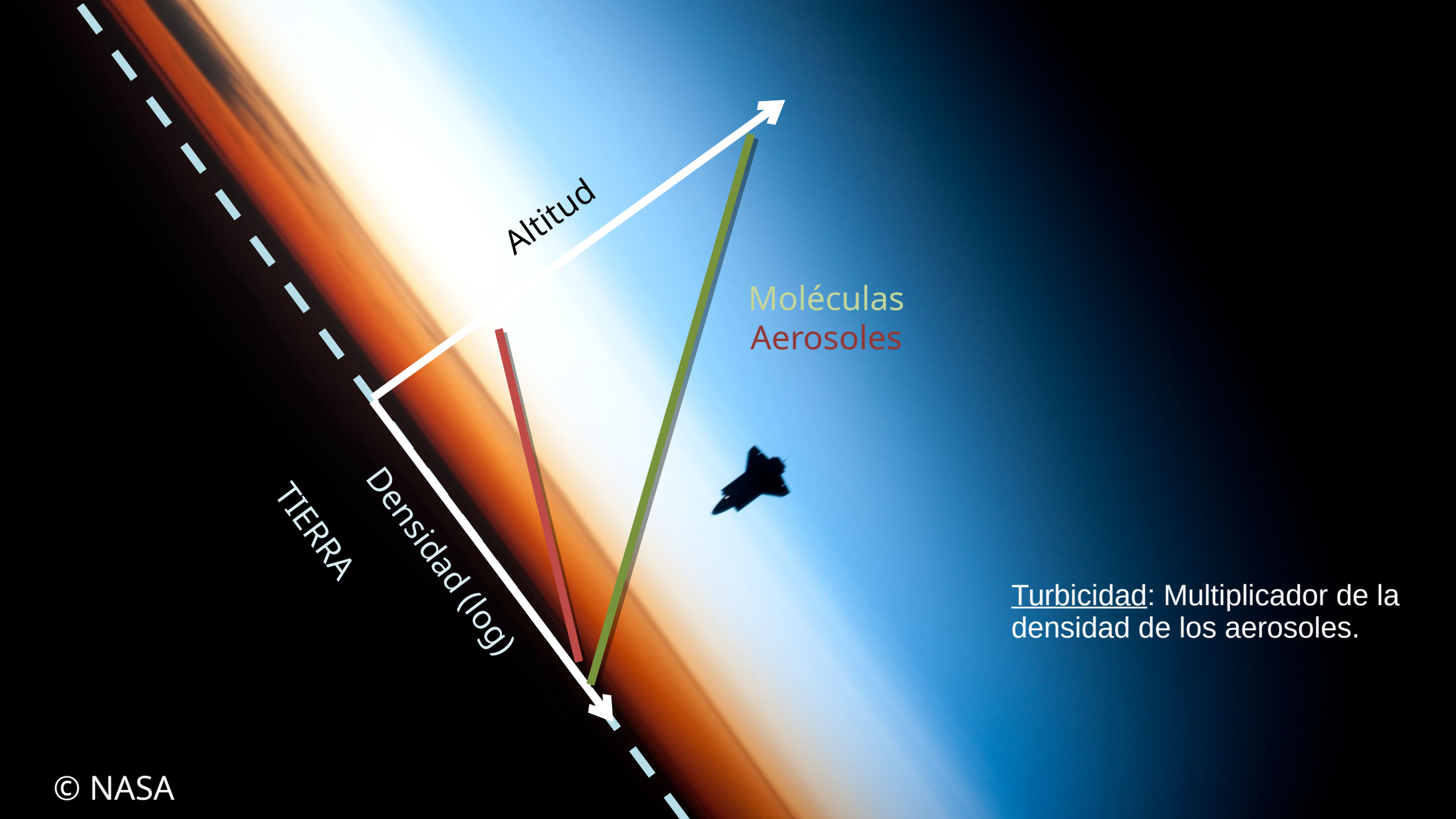
¿Qué es la atmósfera?



1. Moléculas
   - N, O2…
   - Ozono (O3)
2. Aerosoles
   - Agua
   - Arena, Sal
   - Humo


EXOSFERA
MESOSFERA (30 KM)
ESTRATOSFERA (40 KM)
TROPOSFERA (10 KM)
TIERRA


Altitud
Densidad (log)
TIERRA
Moléculas
Aerosoles
© NASA



Turbicidad: Multiplicador de la densidad de los aerosoles.

Transporte de la luz en la atmósfera

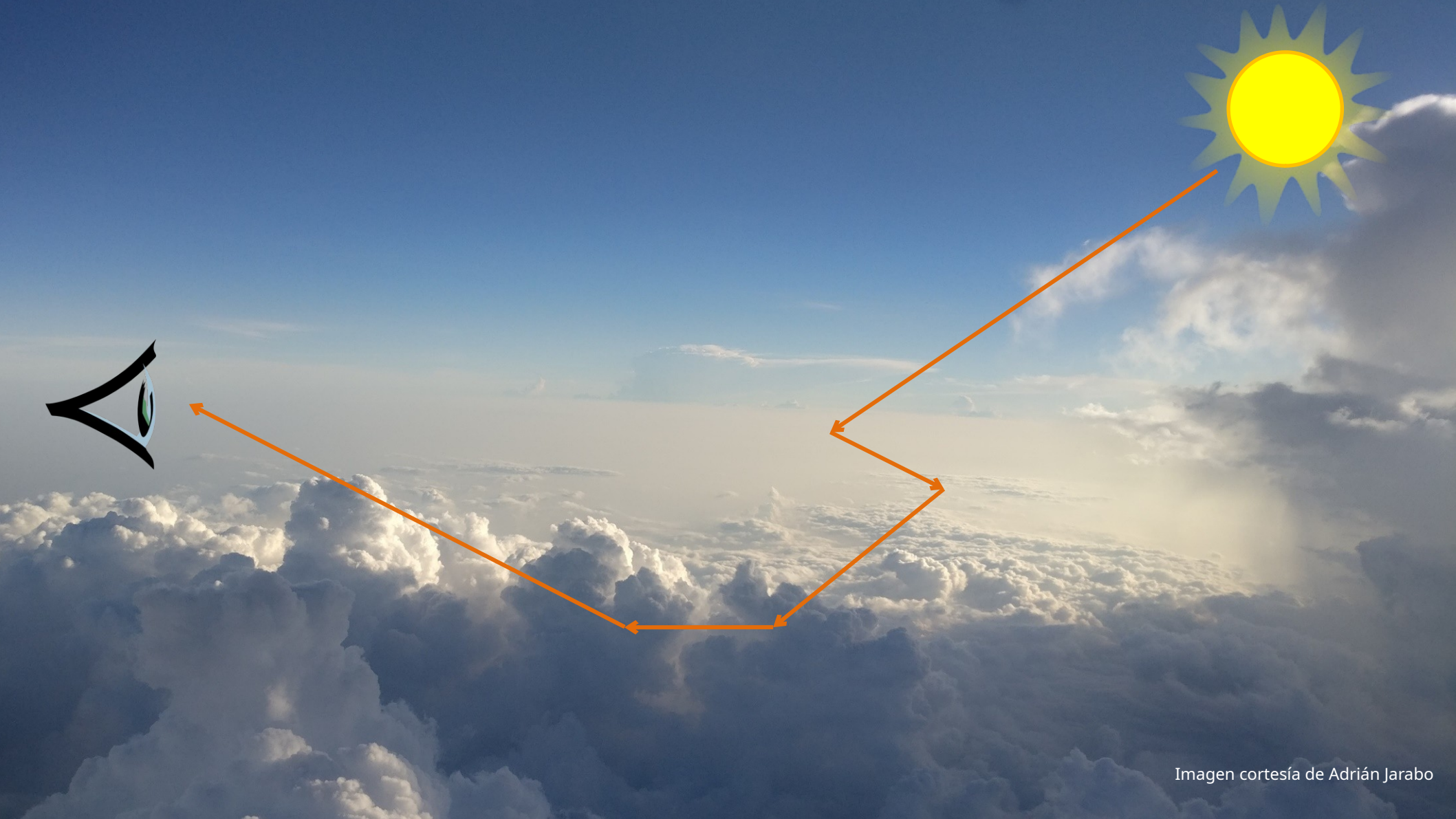


Imagen cortesía de Adrián Jarabo

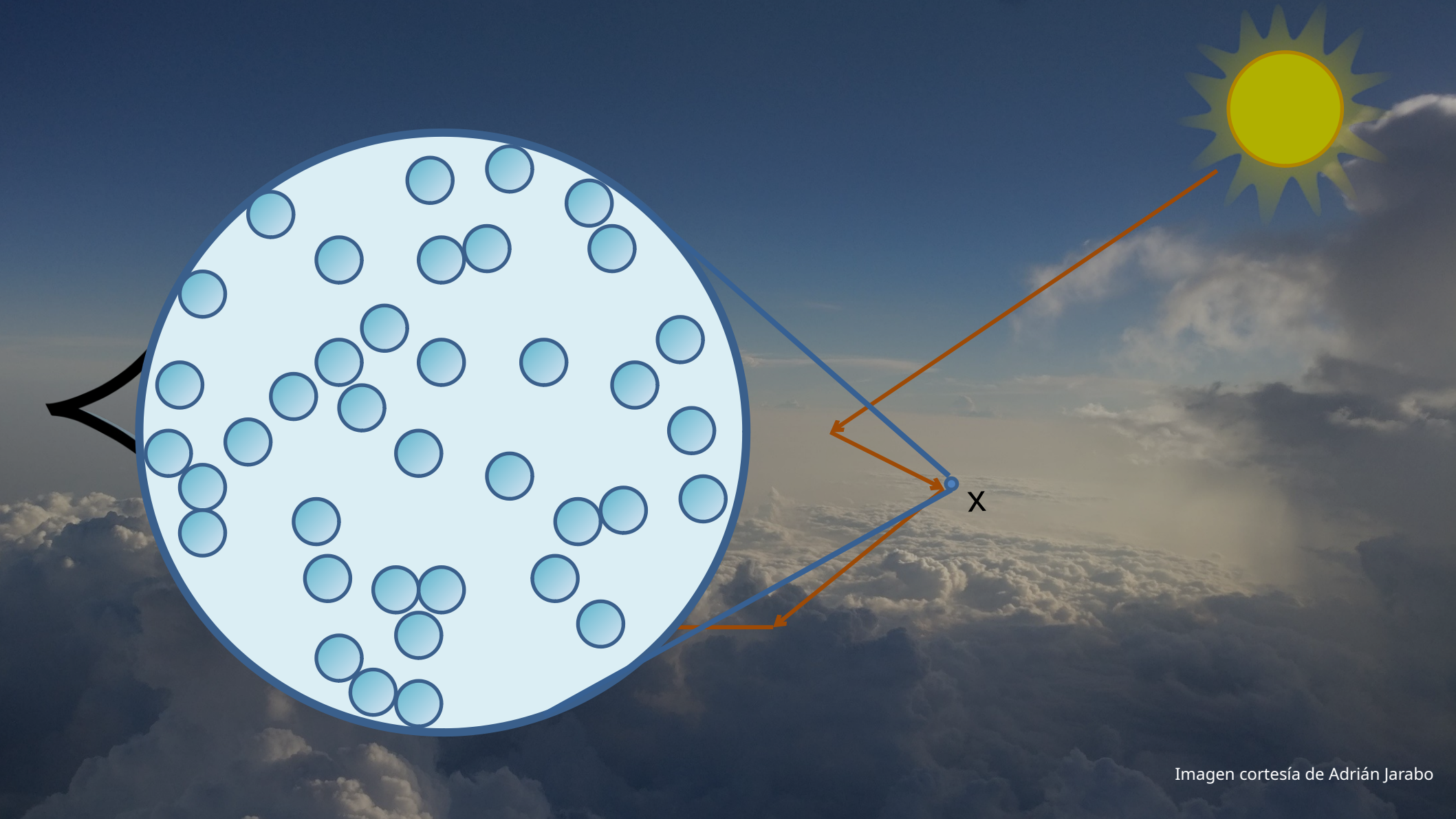


Imagen cortesía de Adrián Jarabo

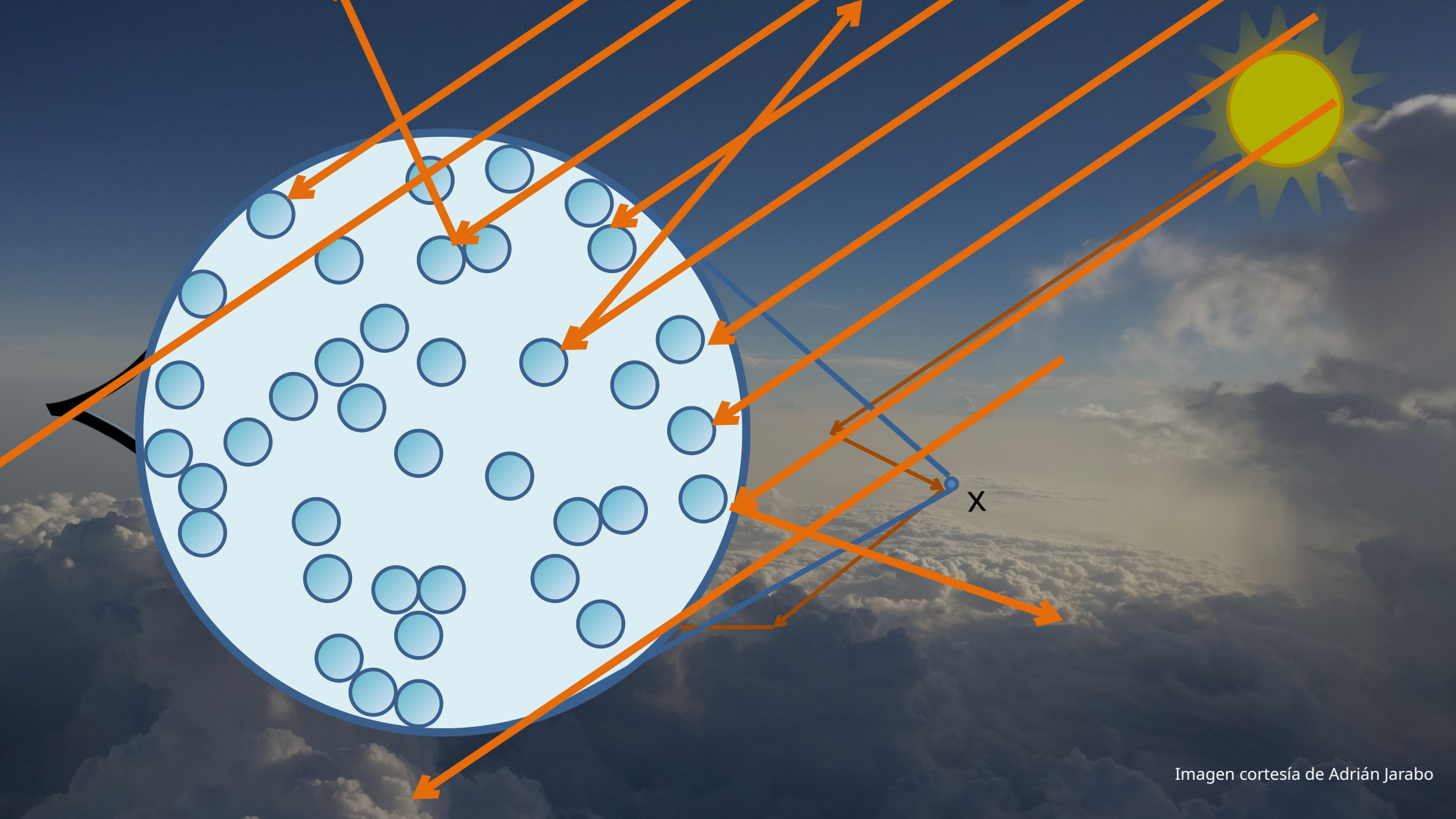
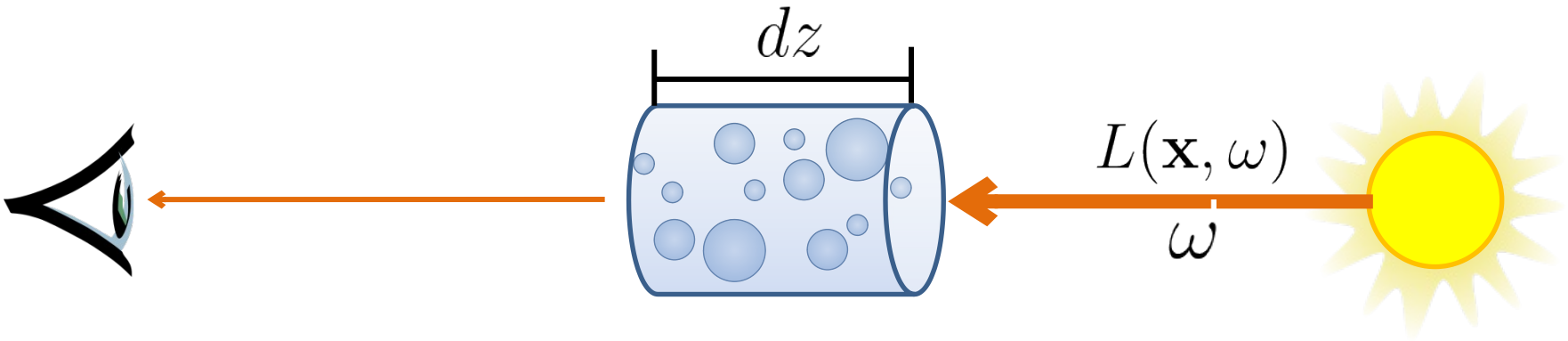


Imagen cortesía de Adrián Jarabo

Evento de absorción

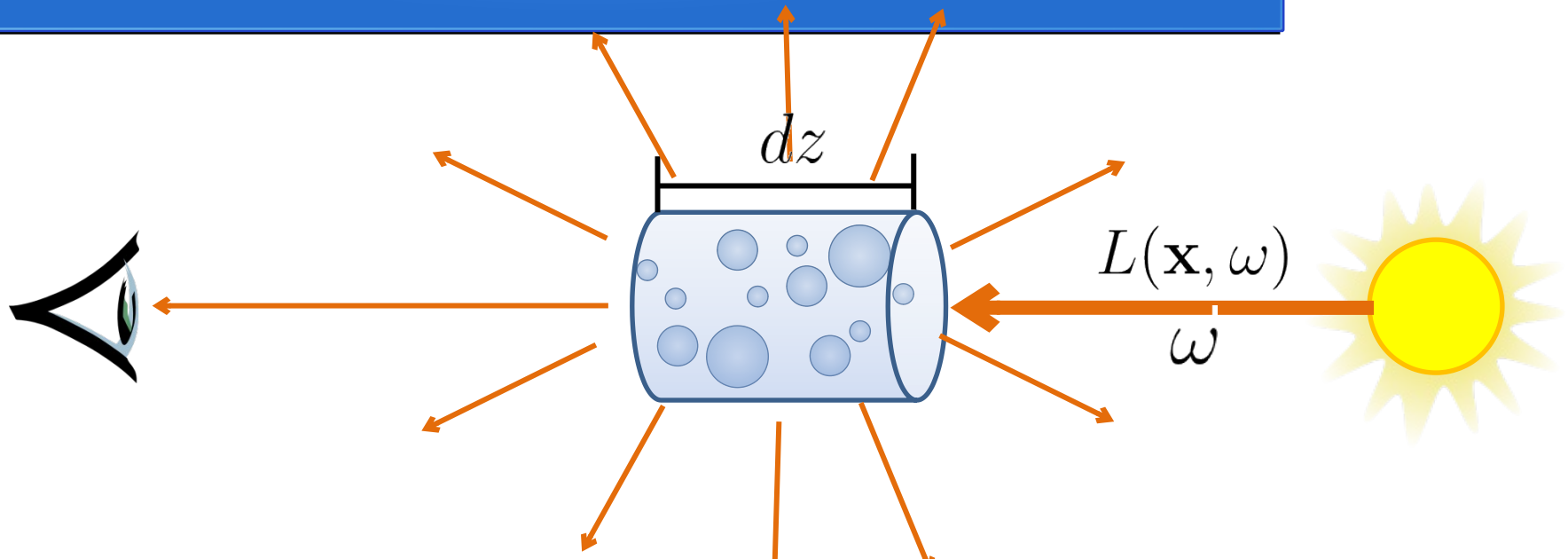


$$dL(\mathbf{x}, \omega) = -\mu_a(\mathbf{x}) L(\mathbf{x}, \omega) dz$$

$\mu_a(\mathbf{x})$: Coeficiente de absorción [m^{-1}]

Sección transversal
×
Concentración de partículas

Evento de out-scattering

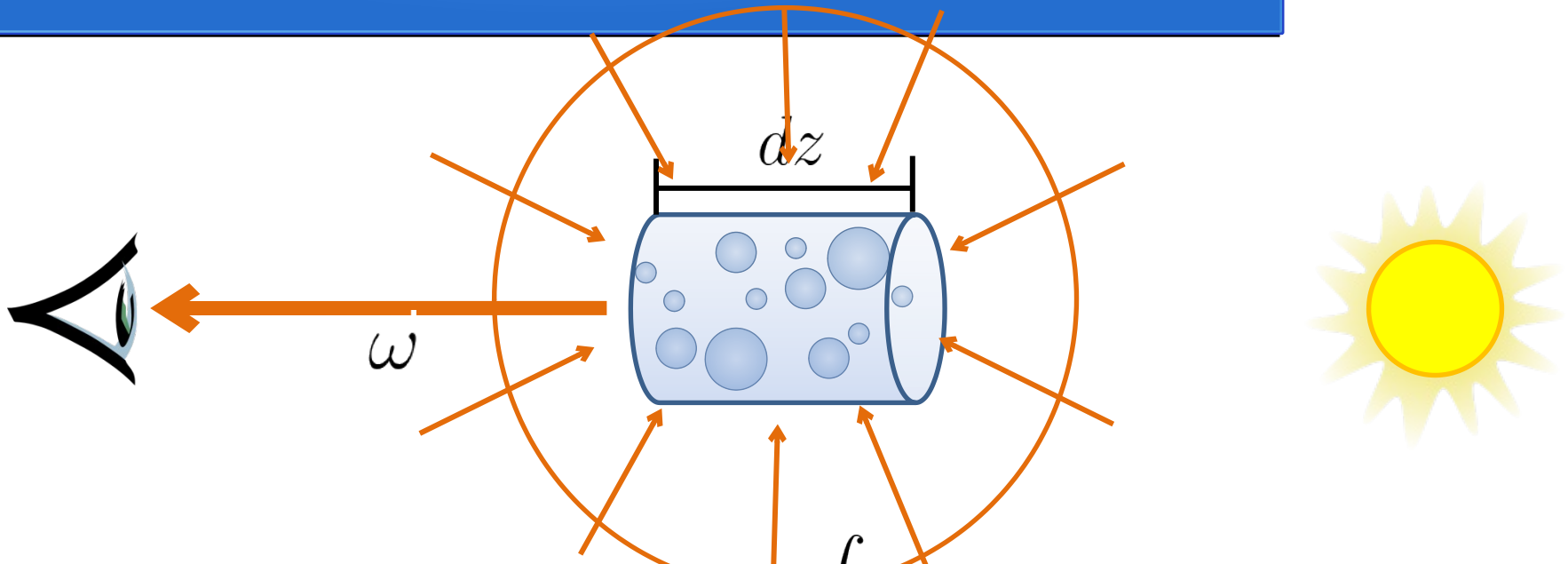


$$dL(\mathbf{x}, \omega) = -\mu_s(\mathbf{x}) L(\mathbf{x}, \omega) dz$$

$\mu_s(\mathbf{x})$: Coeficiente de scattering [m^{-1}]

Sección transversal
×
Concentración de partículas

Evento de in-scattering

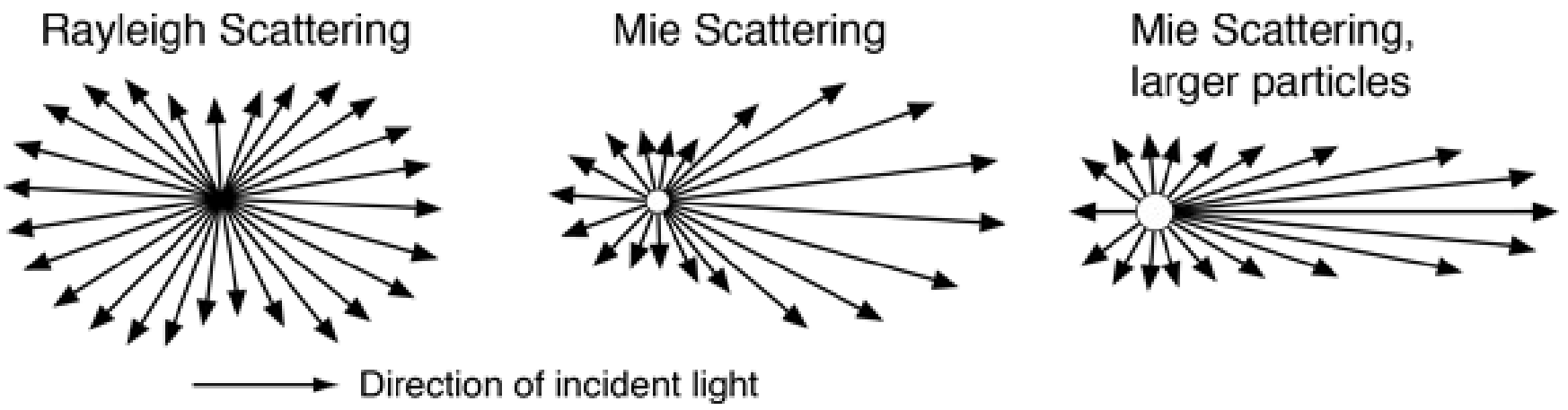


$$dL(\mathbf{x},\omega) = \mu_s \underbrace{\int_\Omega L(\mathbf{x},\omega_i) f_s(\mathbf{x},\omega_i,\omega) d\omega_i}_{L_i(\mathbf{x},\omega)} \; dz$$

$f_s(\mathbf{x},\omega_i,\omega)$: Función de fase [sr^{-1}]

Funciones de fase

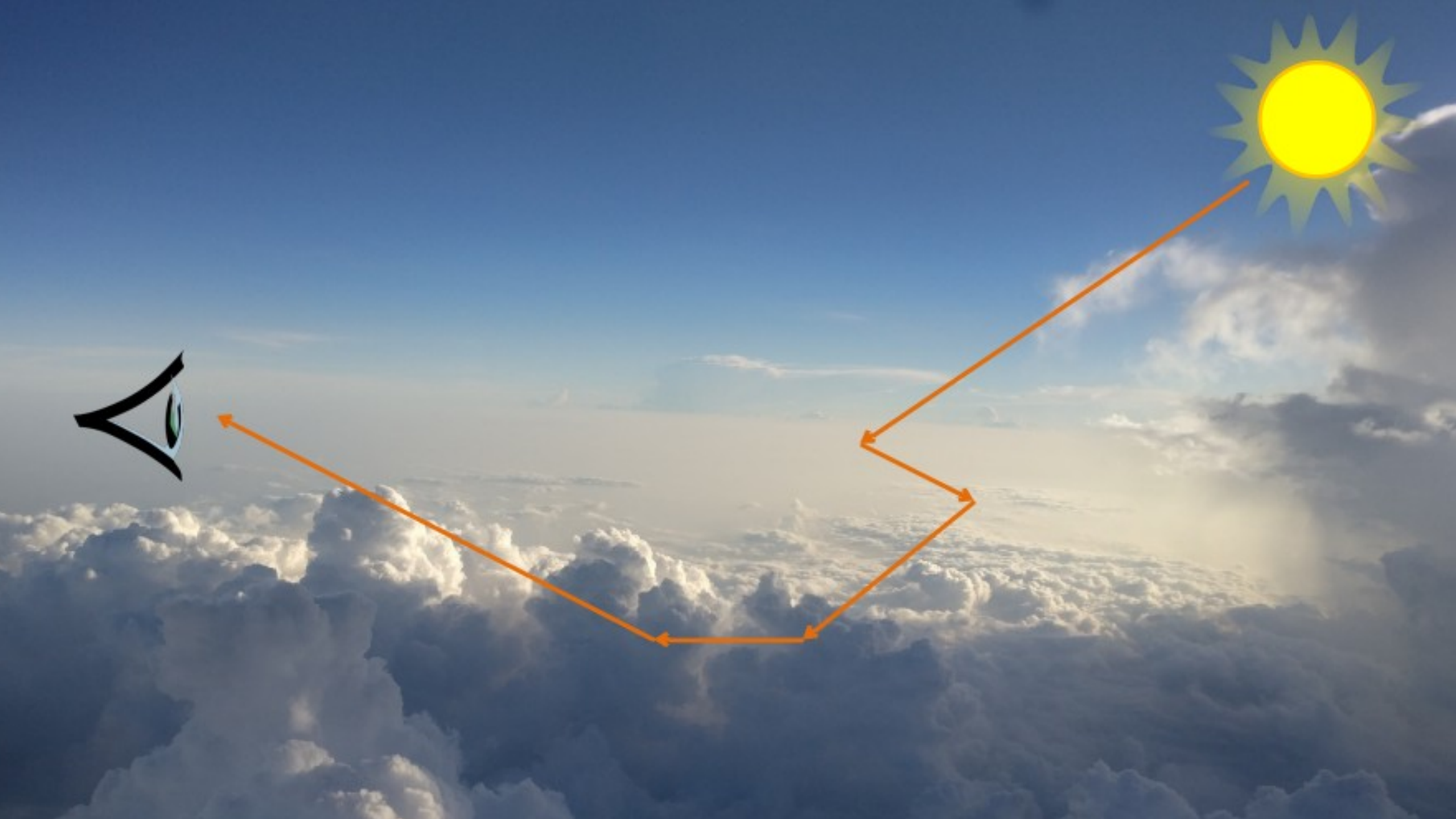
- Moléculas
- Aerosoles

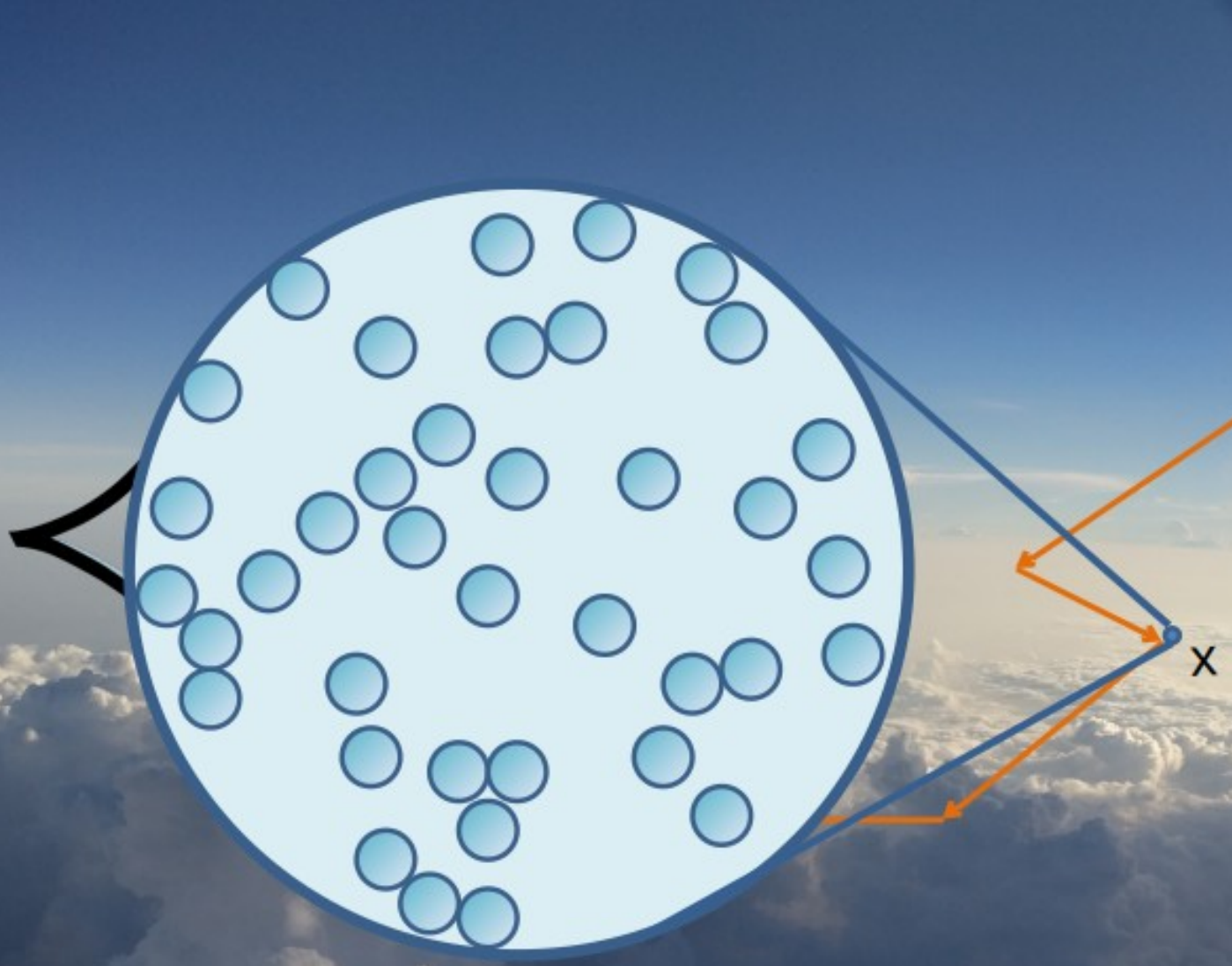


Calculando la radiancia espectral

$$(\overrightarrow{\omega_o} \cdot \nabla) L(\mathbf{x}, \overrightarrow{\omega_o}, \lambda) = -\mu_t(\mathbf{x}, \lambda) L(\mathbf{x}, \overrightarrow{\omega_o}, \lambda) + \mu_s(\mathbf{x}, \lambda) \int_{S^2} f(\mathbf{x}, \overrightarrow{\omega_i} \rightarrow \overrightarrow{\omega_o}, \lambda) L(\mathbf{x}, \overrightarrow{\omega_i}, \lambda) \mathrm{d}\overrightarrow{\omega_i}$$

Recapitulando





1. Moléculas
   - N, O2…
   - Ozono (O3)
2. Aerosoles
   - Agua
   - Arena, Sal
   - Humo

Atmósfera terrestre

- Medio participativo heterogéneo compuesto por:
  - Moléculas (N, O2, Ozono...)
  - Aerosoles (Agua, Arena, Sal, Humo…)
- Existe una dependencia espacio-temporal.
- Transporte de la luz modelado mediante la RTE (Radiative Transfer Equation).
- Partículas definidas a partir de sus parámetros ópticos.

Implementación

Modelos existentes

- Modelos explícitos: Nishita y col. [93,95], O'Neil [05], Jensen y col. [01], Haber y col. [05], Gutierrez y col. [04], Kutz [12].
  - Radiancia precalculada: Bruneton y Neyret [08], Elek y Kmoch [10].
  - Dependiente de la fecha, posición geográfica y bioma: **Guimera y col. [18]**.
- Modelos analíticos: Perez y col. [93], Preetham y col. [99], Wilkie y col. [04], Hosek y Wilkie [12, 13].

Modelo elegido

- Modelo de Guimera y col. [18]
  - Dispersión molecular de Rayleigh en función de la altitud.
  - Absorción molecular del ozono en función de la altitud y de la fecha.
  - Modelo de aerosoles en función de la altitud y del entorno bioclimático (*Background*, *Desert-Dust*, *Maritime-Clean*, *Maritime-Mineral*, *Polar-Antartic*, *Polar-Artic*, *Remote-Continental*, *Rural* y *Urban*).

Mitsuba 2

- Motor de render open-source.
- Implementa algoritmos de transporte de luz utilizando radiancia espectral.
- Escrito en C++20 bajo un paradigma de orientación a objetos.
- Compuesto por una estructura de clases genéricas de las que heredan diferentes plugins.

¿Por qué Mitsuba 2?

- Porque es eficiente y vectorizable.
- Porque es un render espectral.
- Porque utiliza rendering diferenciable.
- Porque permite rendering polarizable.

Volume reconstruction

Reconstruct volume density from multiview setup (accounting for scattering)

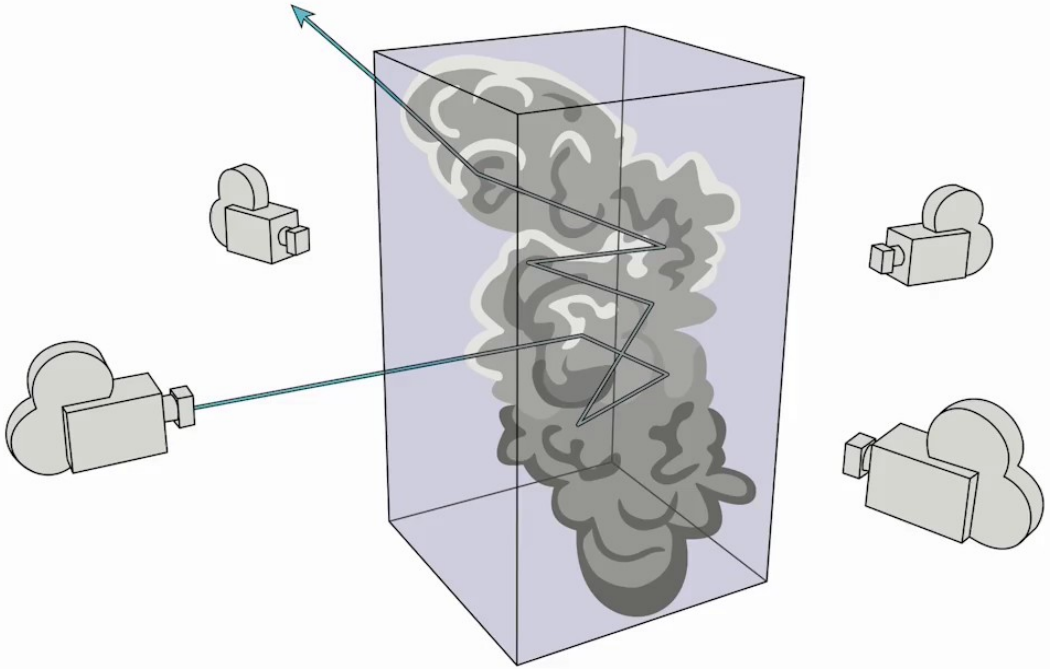


Diagrama de clases

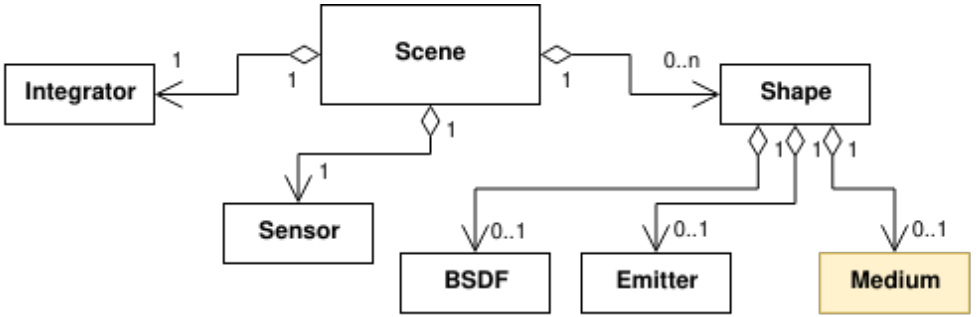
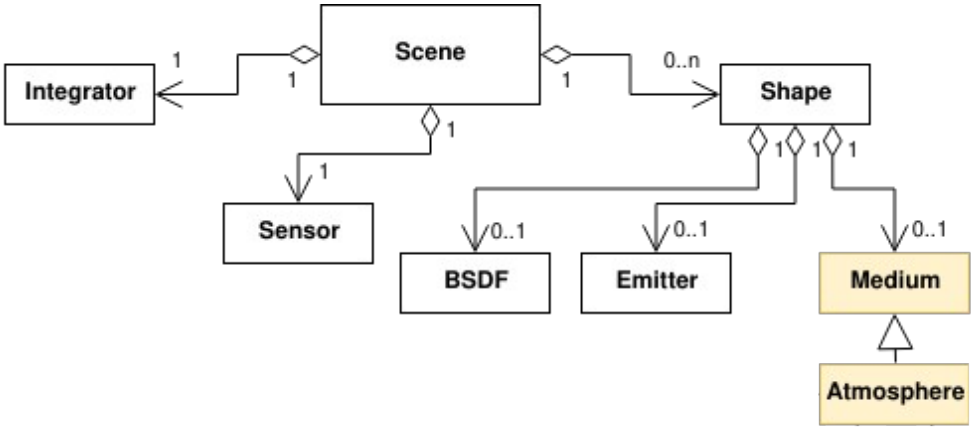
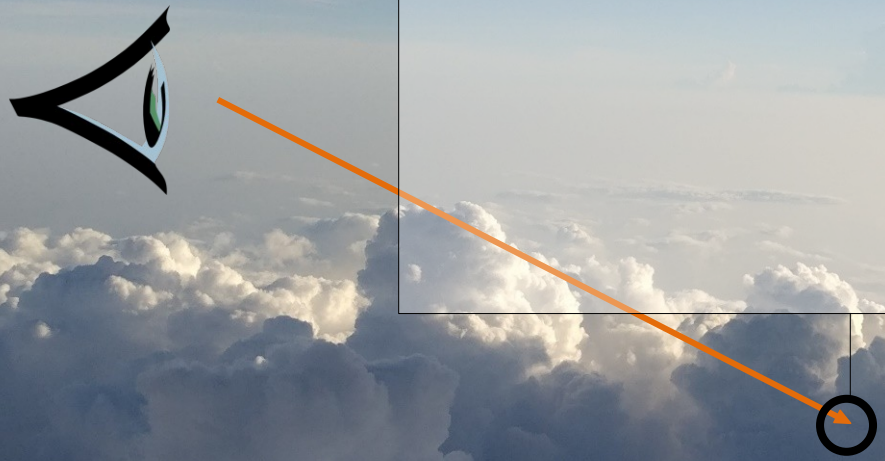


Diagrama de clases



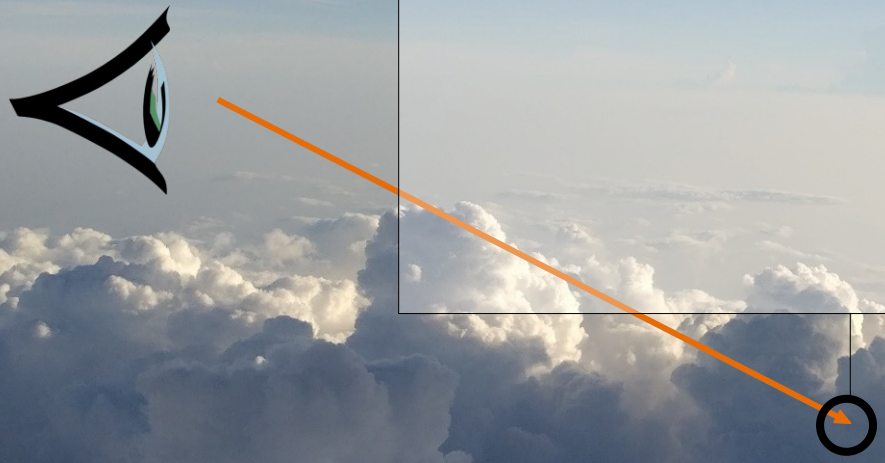
Path tracing volumétrico

1. Calcular la distancia del rayo en base a la extinción hasta que se produzca un evento.



1. Calcular la distancia del rayo en base a la extinción hasta que se produzca un evento.

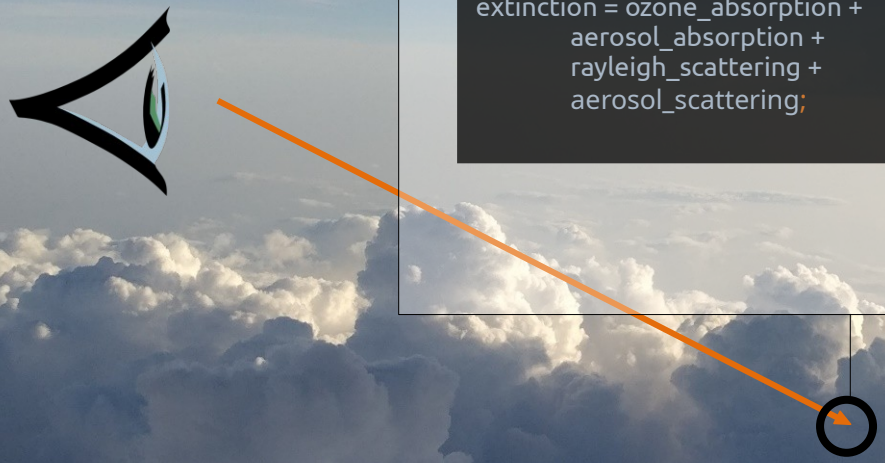
Cuanto mayor sea la extinción, mayor será la probabilidad de que ocurra un evento en una distancia menor.



1. Calcular la distancia del rayo en base a la extinción hasta que se produzca un evento.

Cuanto mayor sea la extinción, mayor será la probabilidad de que ocurra un evento en una distancia menor.

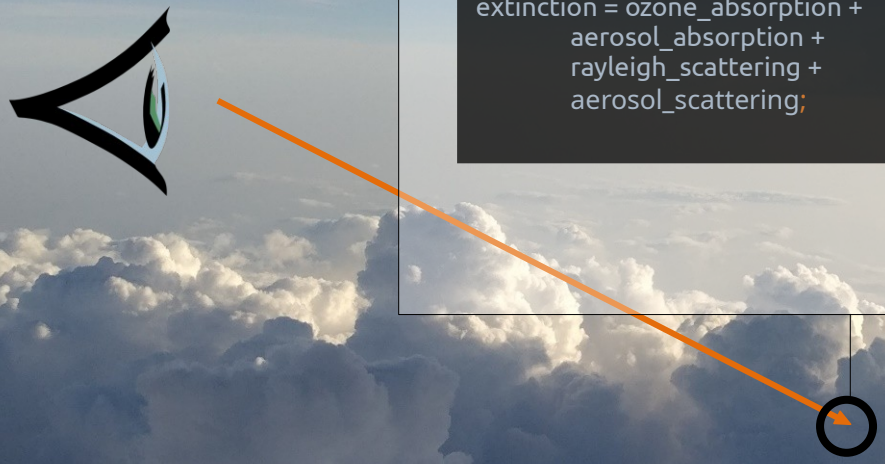
```
extinction = ozone_absorption +
             aerosol_absorption +
             rayleigh_scattering +
             aerosol_scattering;
```



1. Calcular la distancia del rayo en base a la extinción hasta que se produzca un evento.

Cuanto mayor sea la extinción, mayor será la probabilidad de que ocurra un evento en una distancia menor.

```
extinction = ozone_absorption +
             aerosol_absorption +
             rayleigh_scattering +
             aerosol_scattering;
```



Coeficientes de aerosoles, calculados con MiePlot a partir de “Guide to Global Aerosol Models”.

```
// row[0] = wavelength
// row[1] = cross section absorption coefficient
// row[2] = cross section scattering coefficient
const static std::array<std::array<float, 1001>, 3> tabulatedValues;
```

1. Calcular la distancia del rayo en base a la extinción hasta que se produzca un evento.

Cuanto mayor sea la extinción, mayor será la probabilidad de que ocurra un evento en una distancia menor.

```
extinction = ozone_absorption +
             aerosol_absorption +
             rayleigh_scattering +
             aerosol_scattering;
```

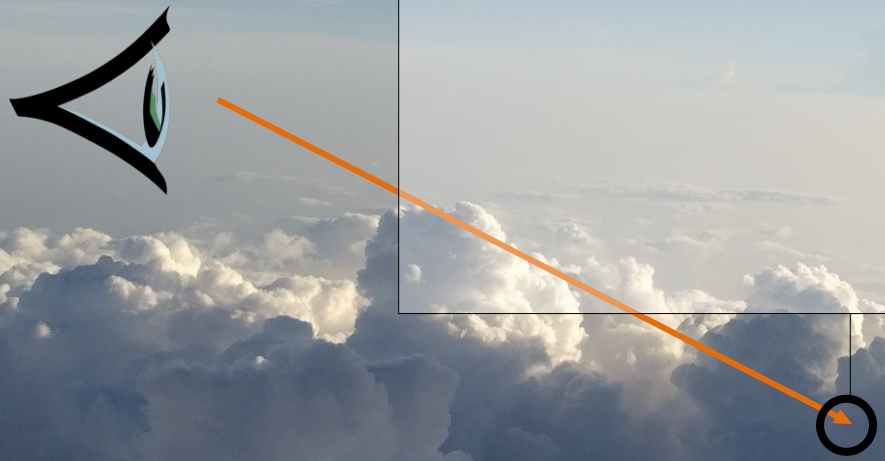
Coeficientes de moléculas, extraídos de:

- Rayleigh: U.S. Standard Atmosphere [76], Bucholtz [95].
- Ozono: Ramanathan y Kulkarni [53], Dütsch [74], Gorshelev y col. [14].

Coeficientes de aerosoles, calculados con MiePlot a partir de “Guide to Global Aerosol Models”.

```
// row[0] = wavelength
// row[1] = cross section absorption coefficient
// row[2] = cross section scattering coefficient
const static std::array<std::array<float, 1001>, 3> tabulatedValues;
```

1. Calcular la distancia del rayo en base a la extinción hasta que se produzca un evento.
2. Calcular la nueva dirección del rayo en base a la función de fase aplicada.

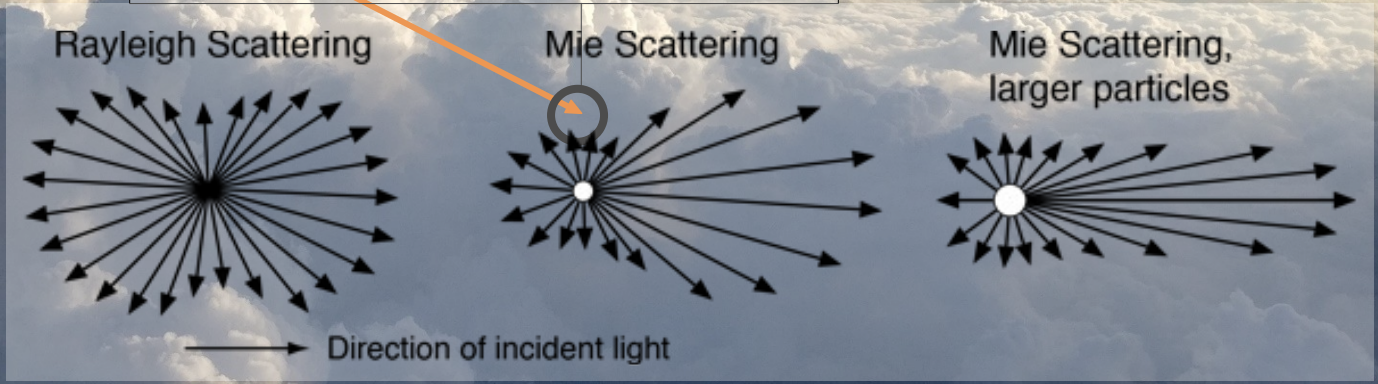


1. Calcular la distancia del rayo en base a la extinción hasta que se produzca un evento.
2. Calcular la nueva dirección del rayo en base a la función de fase aplicada.

```
rayleigh_scattering_prob = rayleigh_scattering / (rayleigh_scattering + aerosol_scattering);

// Russian roulette
if (random < rayleigh_scattering_prob)
    return m_phase_function.get(); // P(molecular) = [0, rayleigh_scattering_prob)
else
    return m_aerosol_phase_function.get(); // P(aerosol) = [rayleigh_scattering_prob,  1)
```

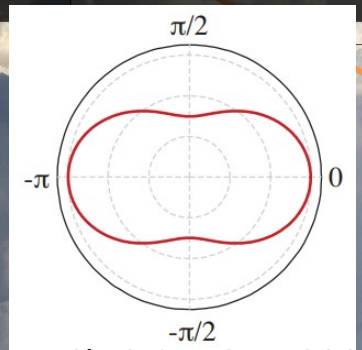
1. Calcular la distancia del rayo en base a la extinción hasta que se produzca un evento.
2. Calcular la nueva dirección del rayo en base a la función de fase aplicada.

```
rayleigh_scattering_prob = rayleigh_scattering / (rayleigh_scattering + aerosol_scattering);

// Russian roulette
if (random < rayleigh_scattering_prob)
    return m_phase_function.get(); // P(molecular) = [0, rayleigh_scattering_prob)
else
    return m_aerosol_phase_function.get(); // P(aerosol) = [rayleigh_scattering_prob,  1)
```

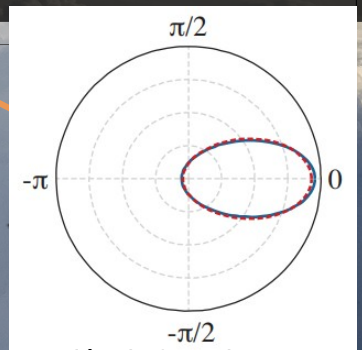


1. Calcular la distancia del rayo en base a la extinción hasta que se produzca un evento.
2. Calcular la nueva dirección del rayo en base a la función de fase aplicada.

```
rayleigh_scattering_prob = rayleigh_scattering / (rayleigh_scattering + aerosol_scattering);

// Russian roulette
if (random < rayleigh_scattering_prob)
    return m_phase_function.get(); // P(molecular) = [0, rayleigh_scattering_prob)
else
    return m_aerosol_phase_function.get(); // P(aerosol) = [rayleigh_scattering_prob,  1)
```



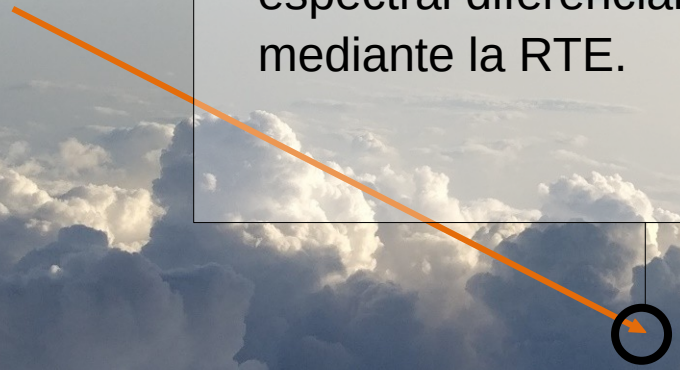
Función de fase de Rayleigh



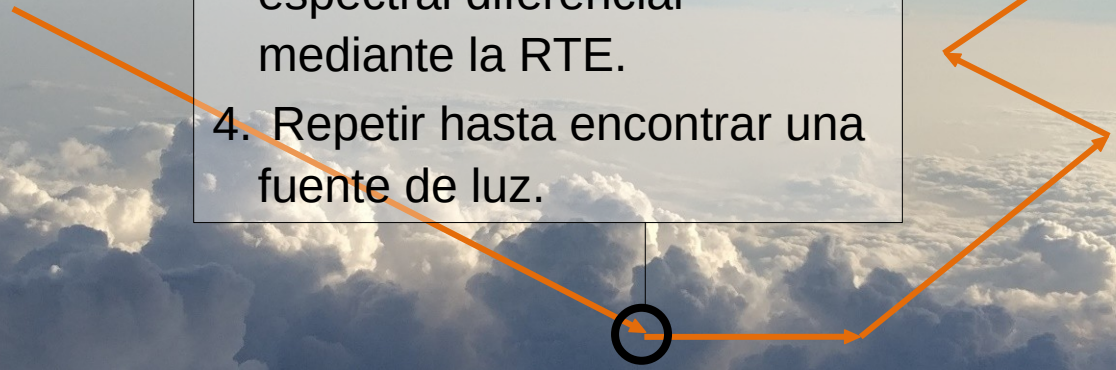
Función de fase de Henyey-Greenstein

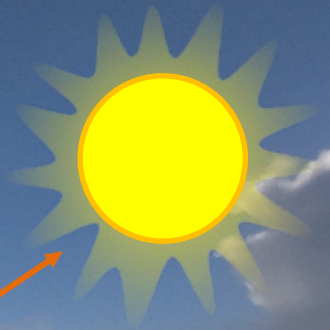
Direction of incident light

1. Calcular la distancia del rayo en base a la extinción hasta que se produzca un evento.
2. Calcular la nueva dirección del rayo en base a la función de fase aplicada.
3. Calcular la radiancia espectral diferencial mediante la RTE.



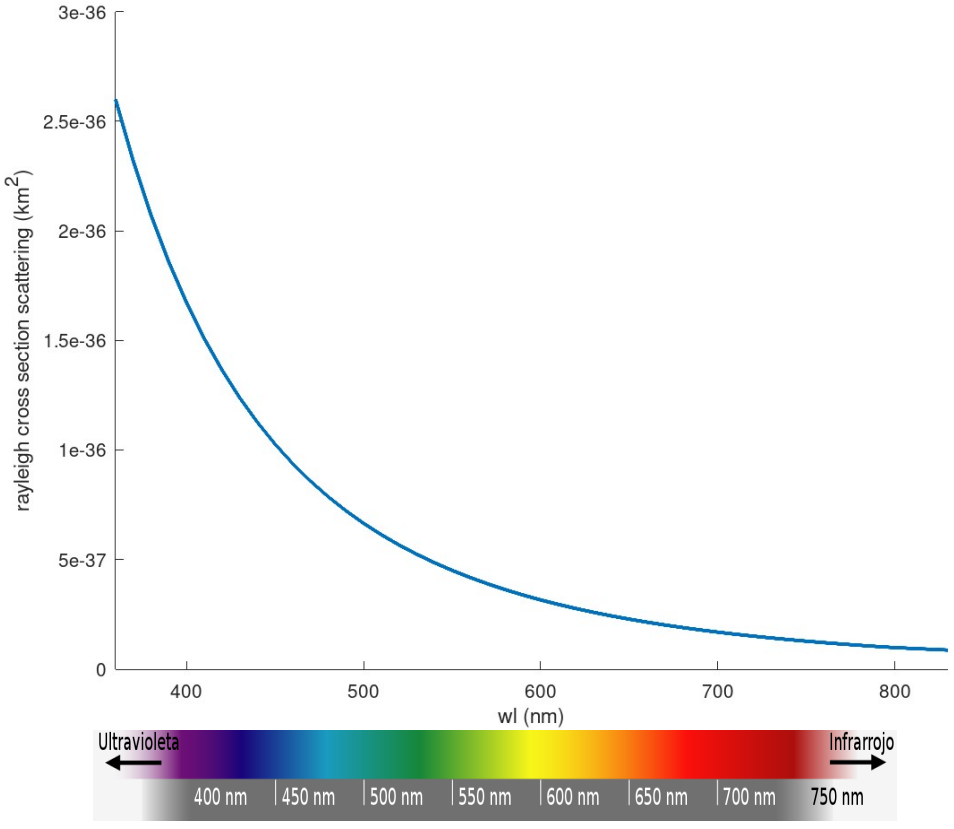
1. Calcular la distancia del rayo en base a la extinción hasta que se produzca un evento.
2. Calcular la nueva dirección del rayo en base a la función de fase aplicada.
3. Calcular la radiancia espectral diferencial mediante la RTE.
4. Repetir hasta encontrar una fuente de luz.

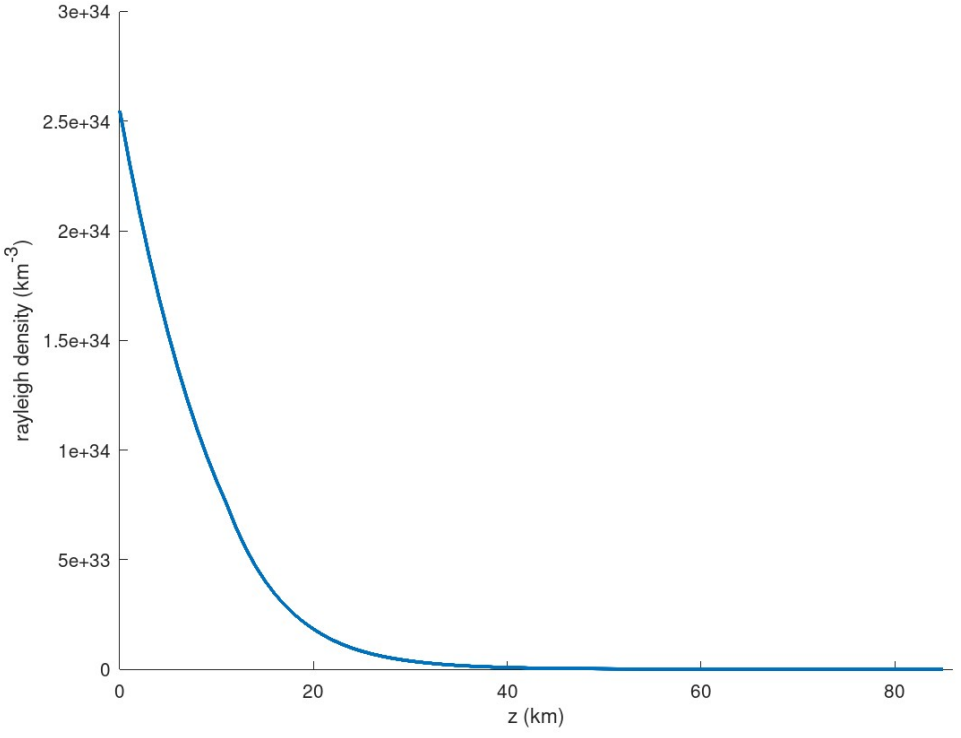




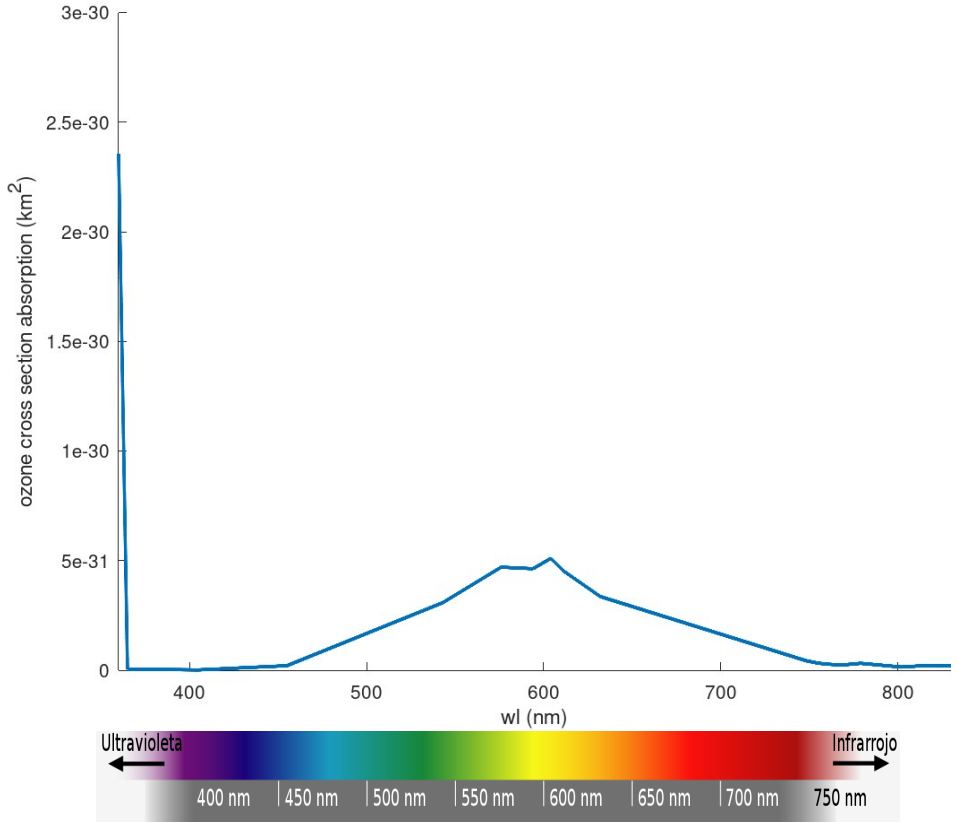
Análisis del modelo

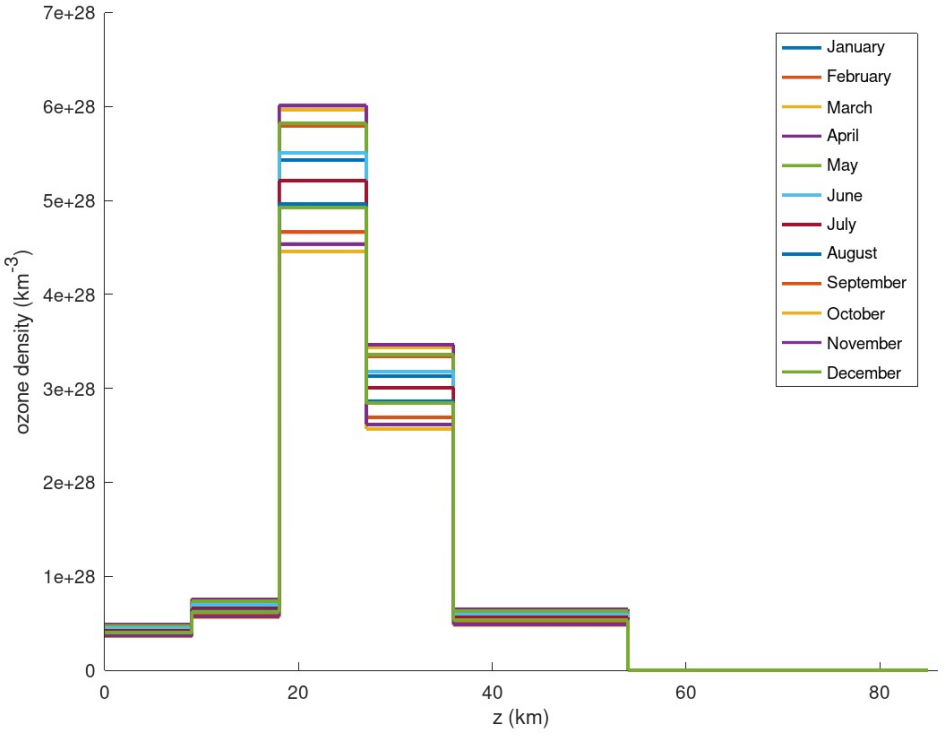
Rayleigh Scattering



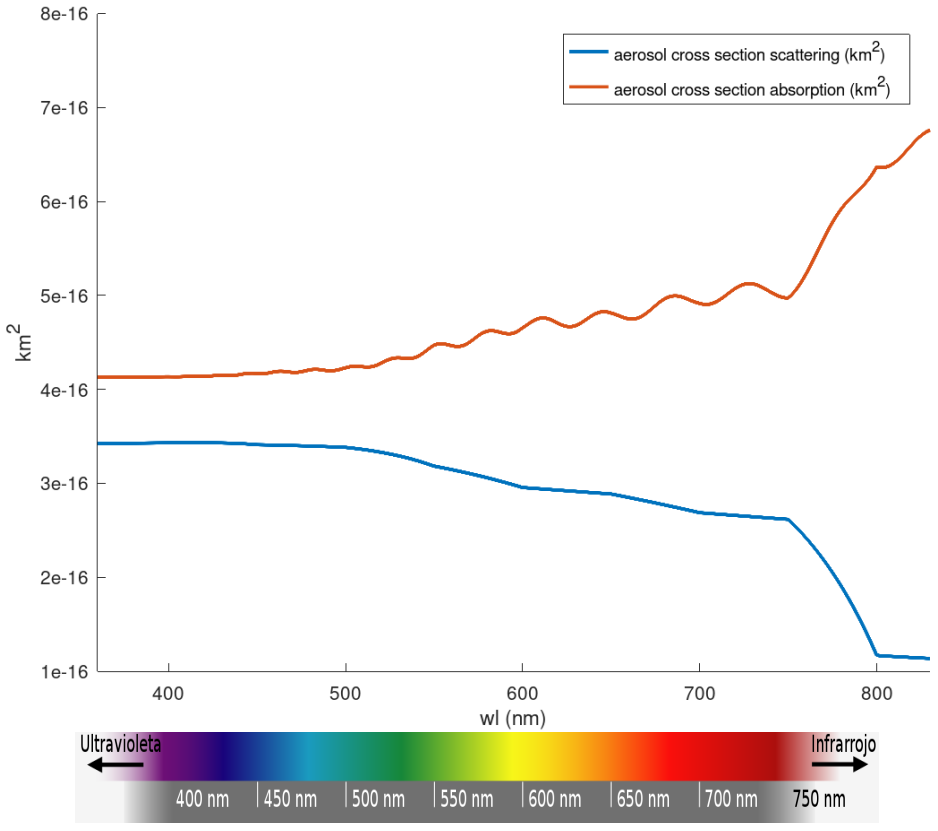


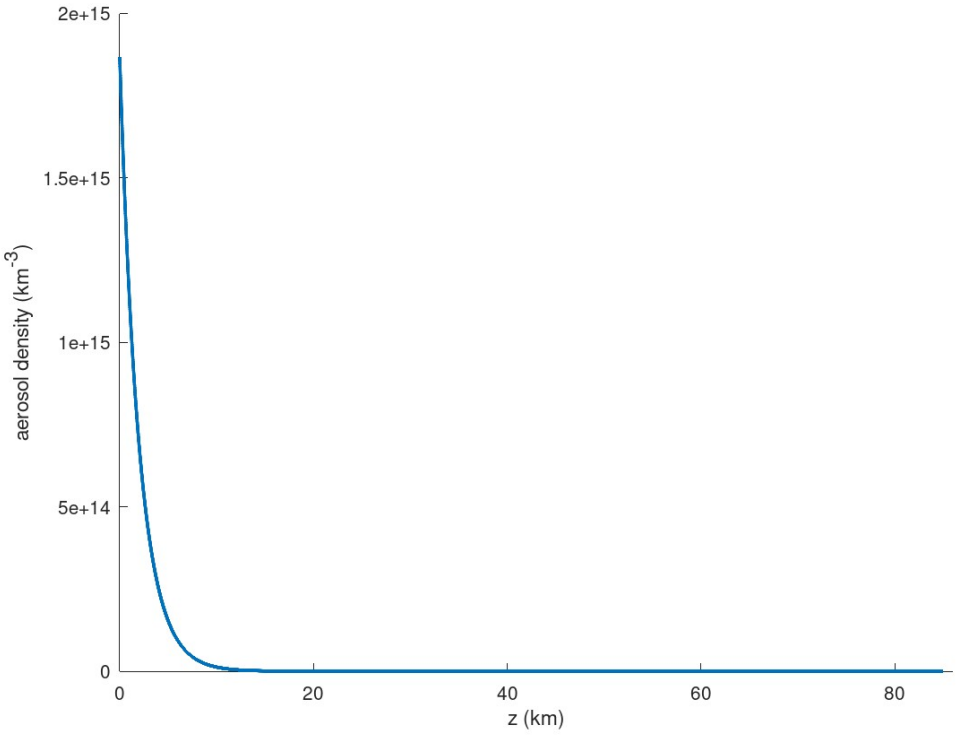
Absorción del ozono



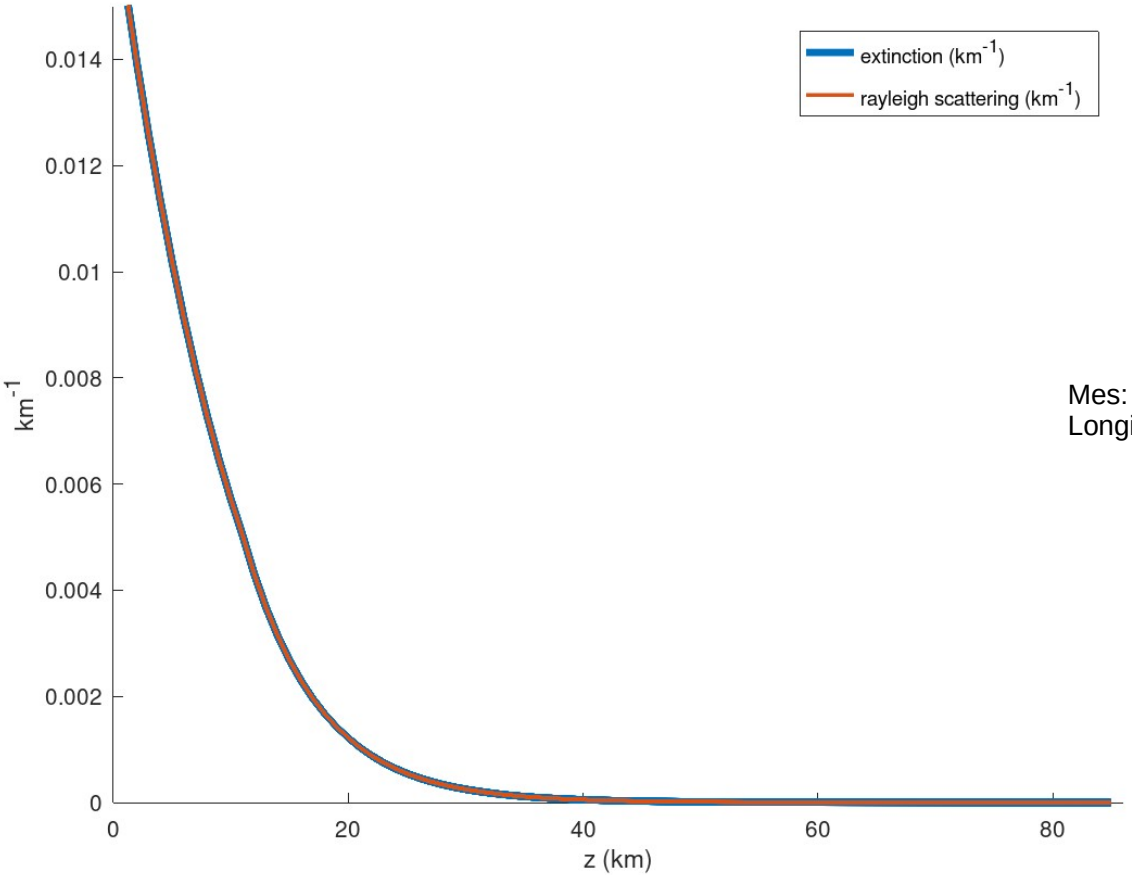


Aerosol Desert-Dust



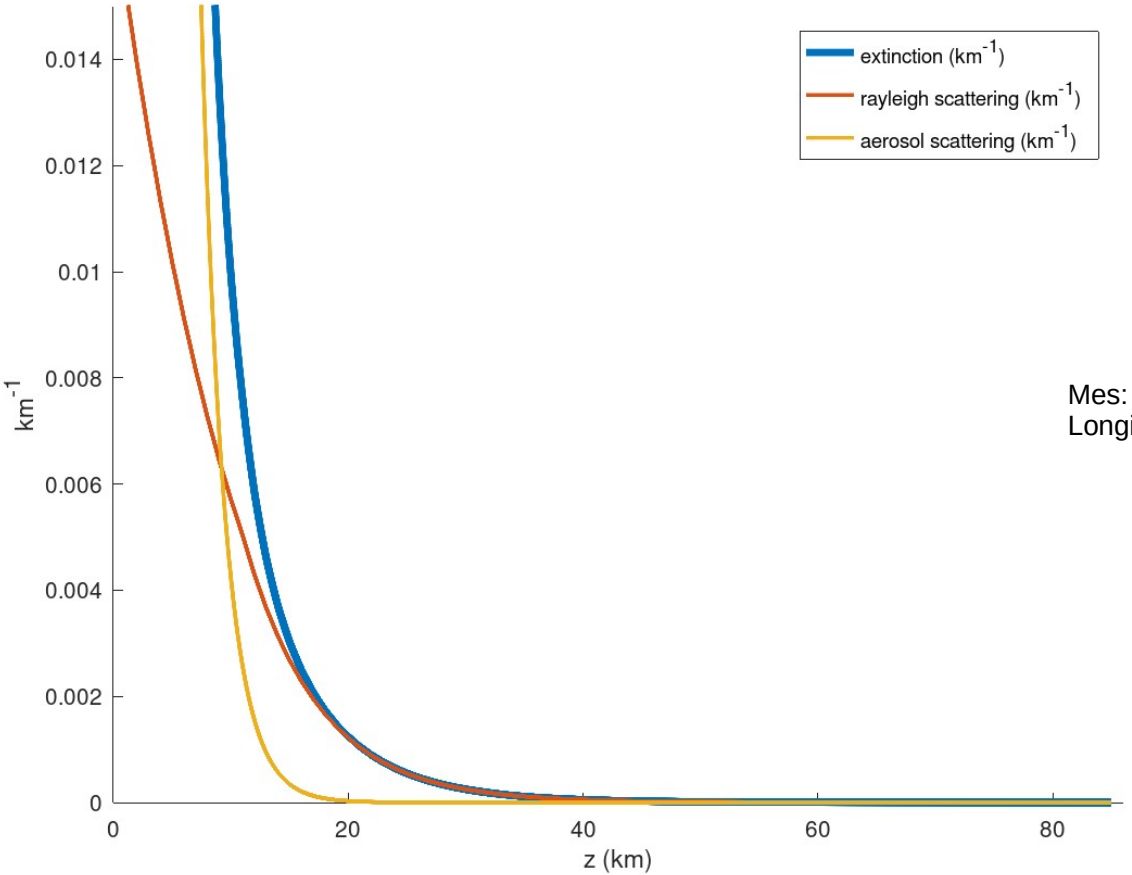


Rayleigh + Ozono + Desert-Dust



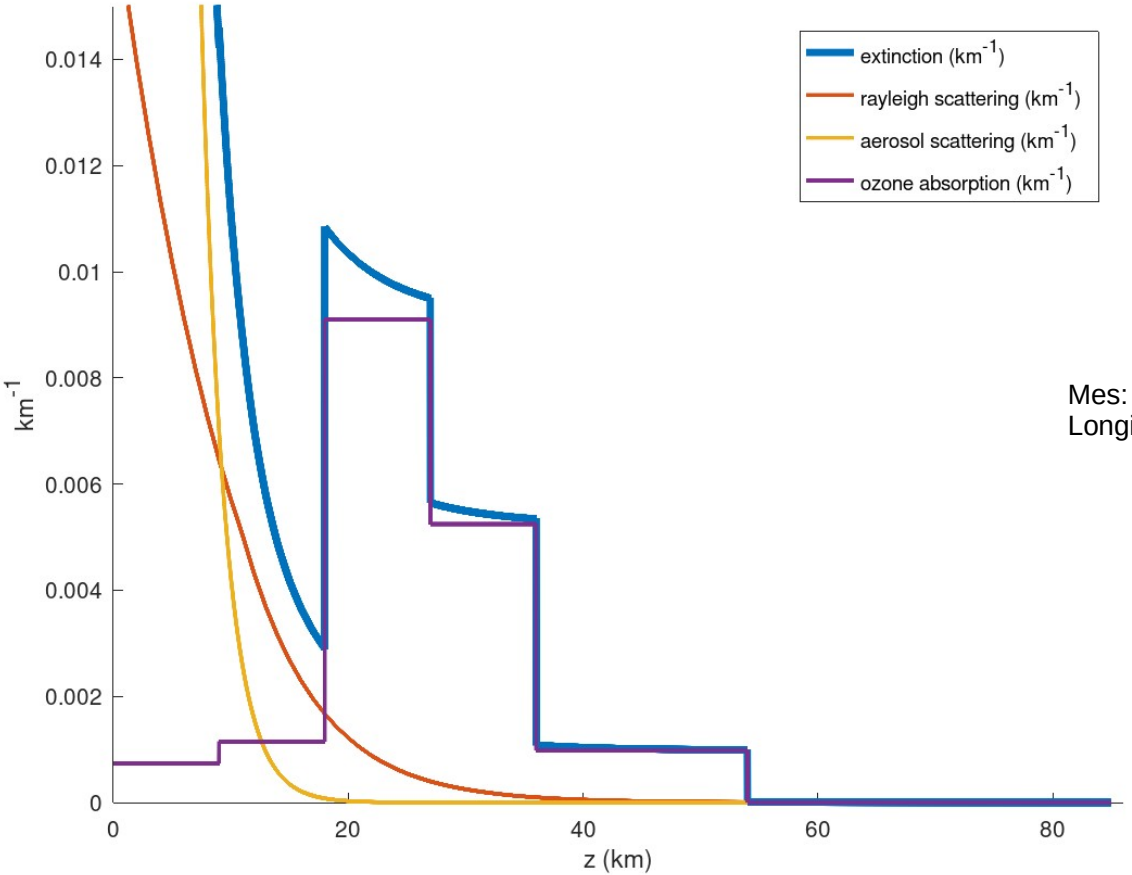
Mes: Enero
Longitud de onda: 500 nm

Rayleigh + Ozono + Desert-Dust



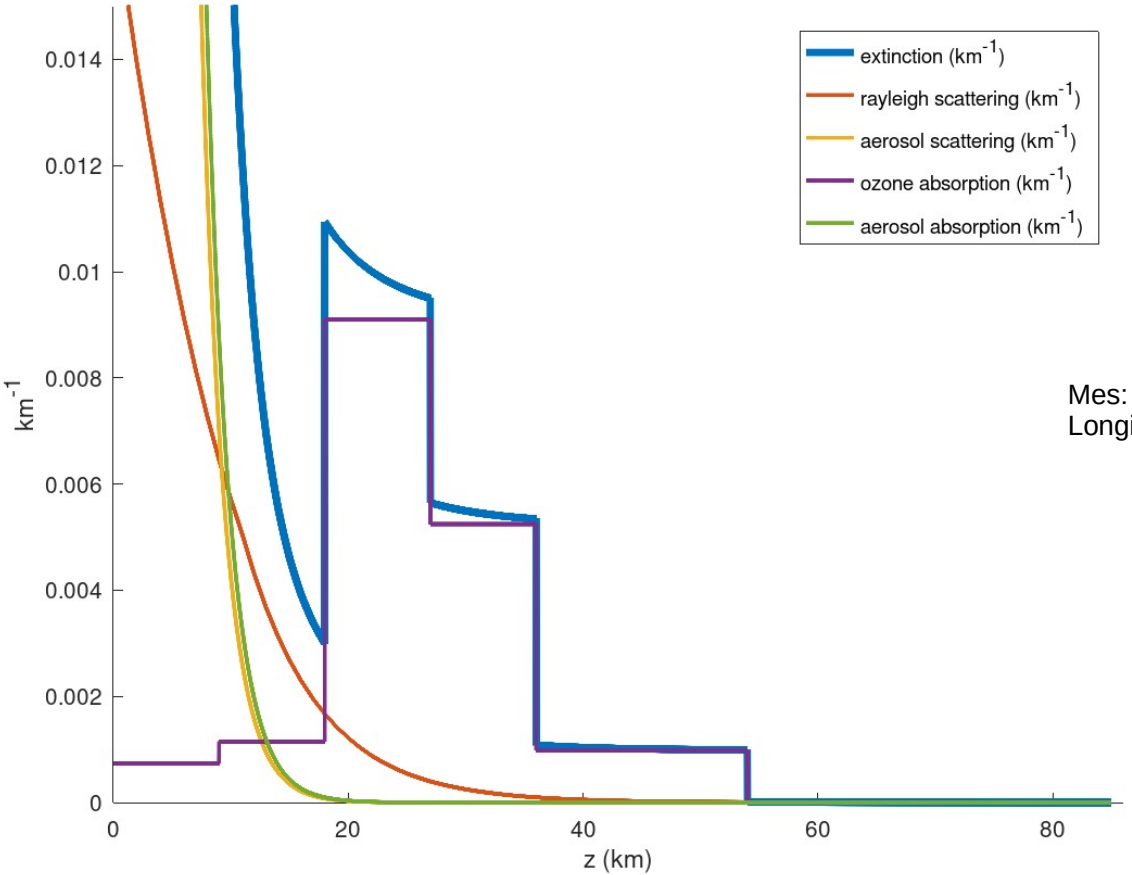
Mes: Enero
Longitud de onda: 500 nm

Rayleigh + Ozono + Desert-Dust



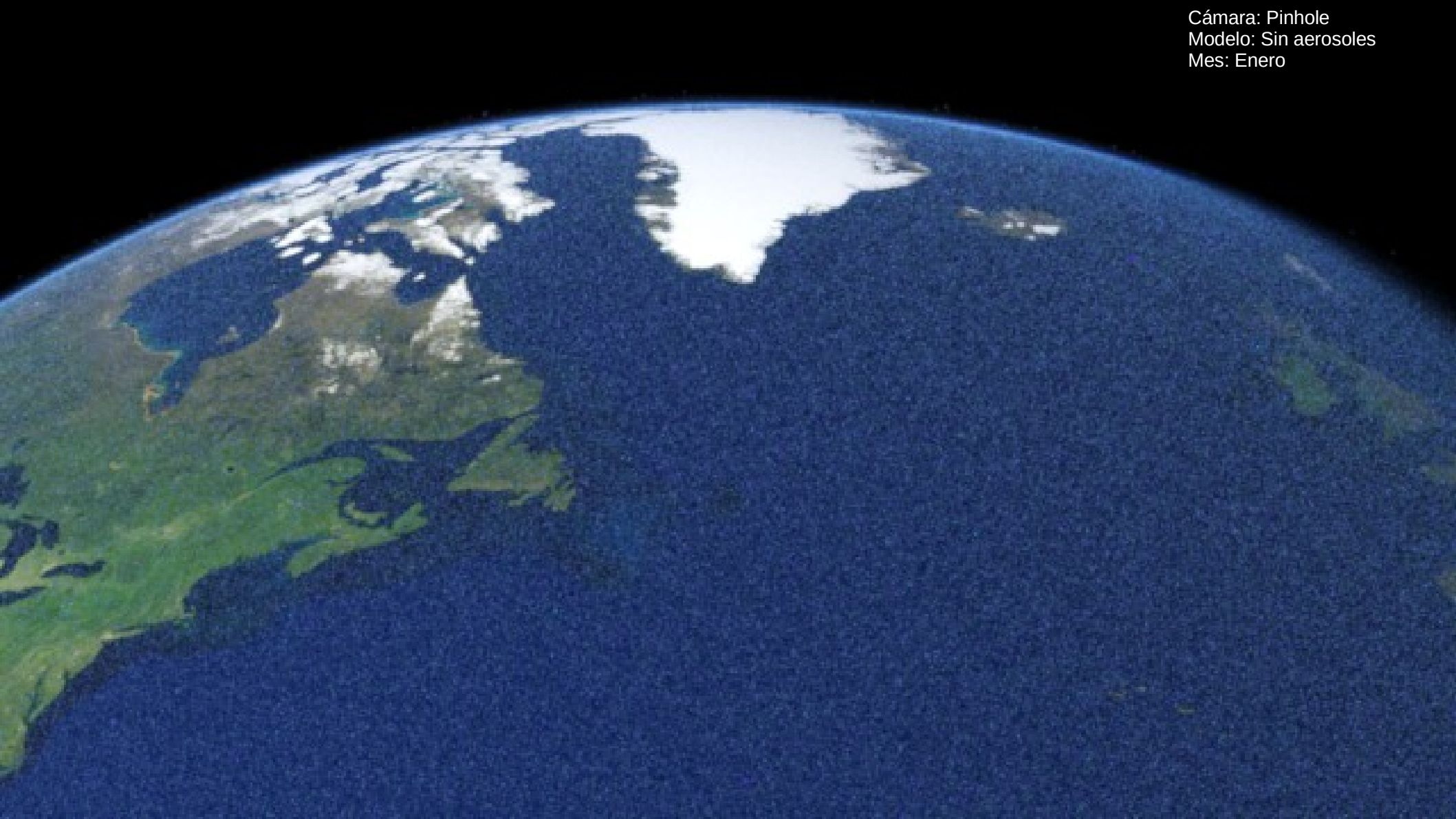
Mes: Enero
Longitud de onda: 500 nm

Rayleigh + Ozono + Desert-Dust



Mes: Enero
Longitud de onda: 500 nm

Resultados


Cámara: Pinhole
Modelo: Sin aerosoles
Mes: Enero

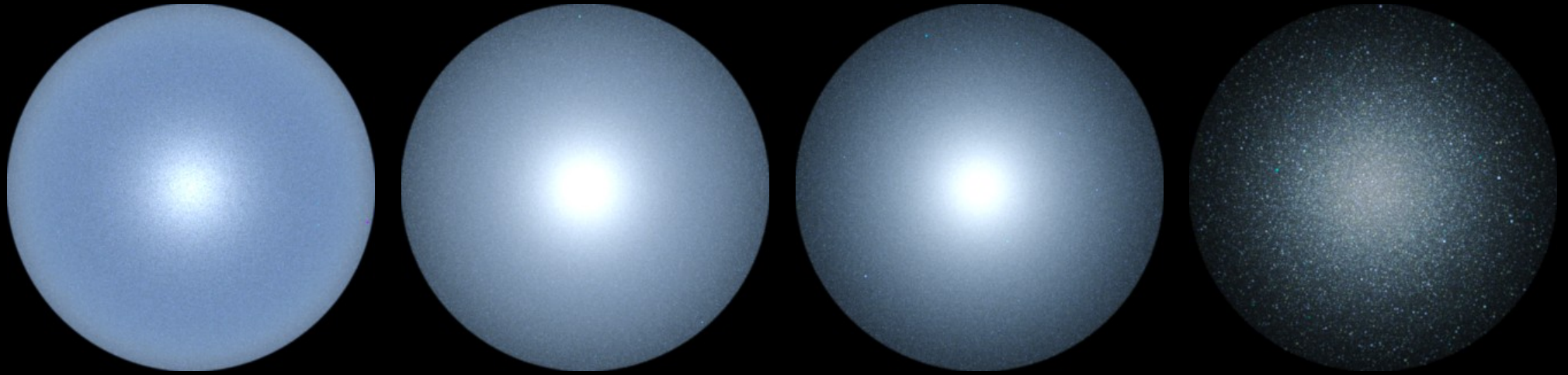

Cámara: Pinhole
Modelo: Sin aerosoles
Mes: Febrero
Sol: 90º sobre el horizonte


Cámara: Pinhole
Modelo: Sin aerosoles
Mes: Febrero
Sol: 180º sobre el horizonte

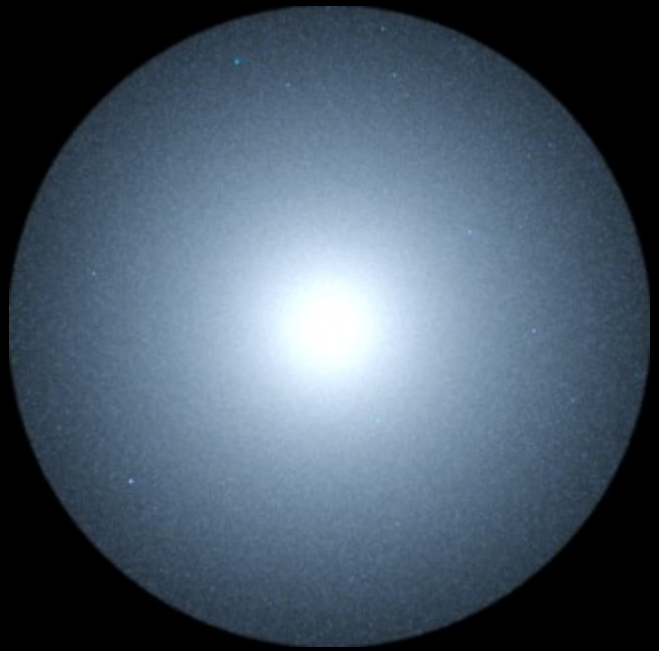
Cámara: Fisheye
Modelo: Desert-Dust
Mes: Enero
Turbicidad: Aumenta de izq. a der.

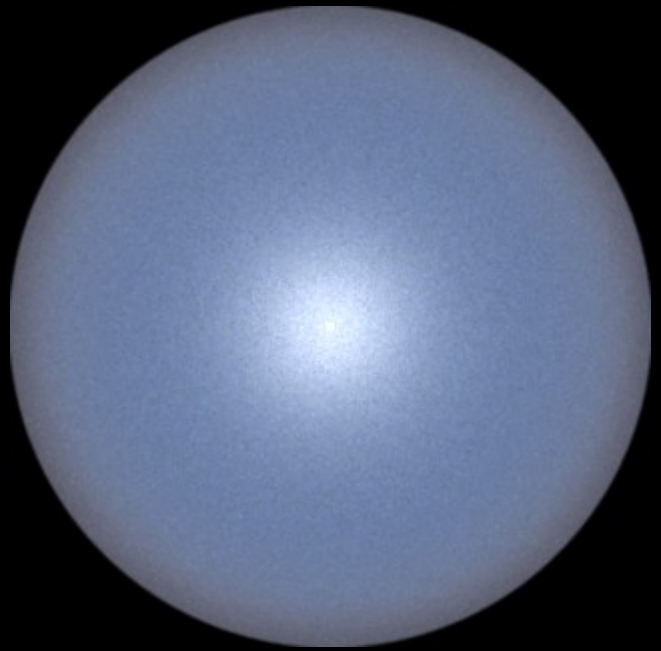
Cámara: Fisheye
Modelo: Sin aerosoles
Mes: Enero
Sol: 90º, 5º y -5º (izq. a der.) sobre el horizonte

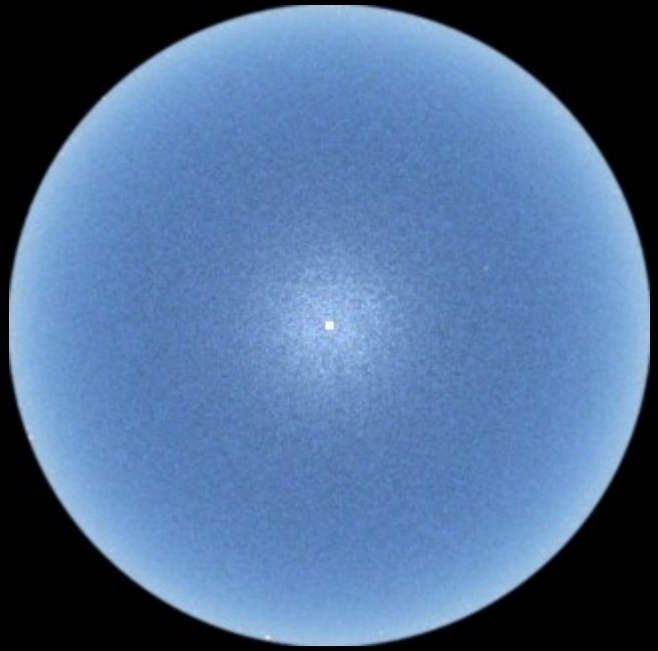
Cámara: Fisheye
Modelo: Desert-Dust
Mes: Enero
Sol: 90º sobre el horizonte
Turbicidad: Aumenta de izq. a der.



Cámara: Fisheye
Mes: Enero
Sol: 90º sobre el horizonte
Turbicidad: Media

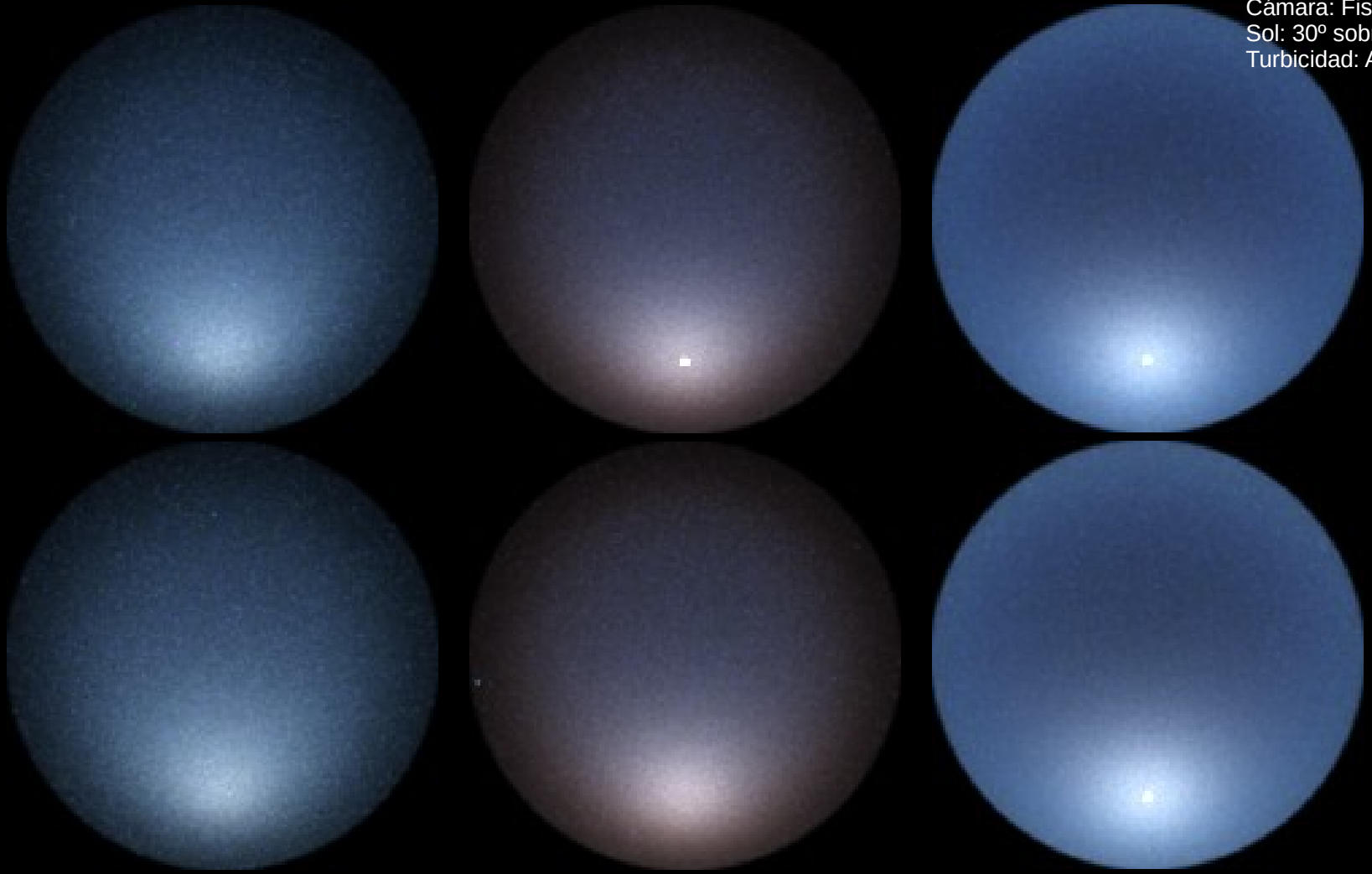






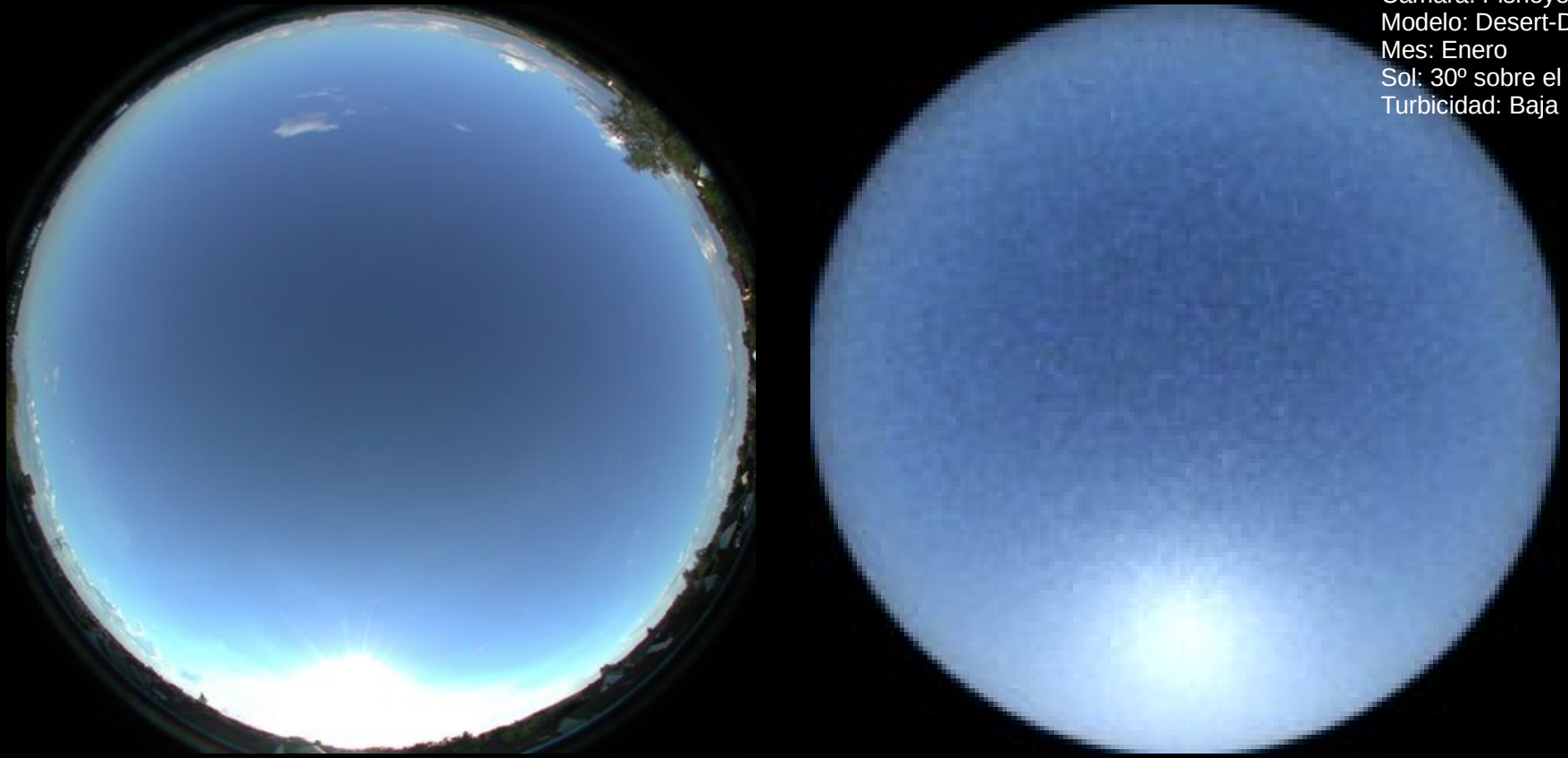
- Desert-Dust
- Maritime-Mineral
- Polar-Artic

Cámara: Fisheye
Sol: 30º sobre el horizonte
Turbicidad: Alta



- Enero
- Noviembre

- Desert-Dust
- Maritime-Mineral
- Polar-Artic

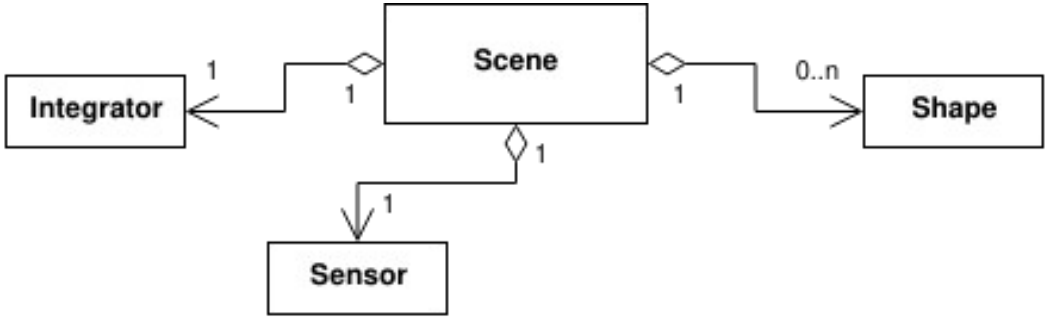


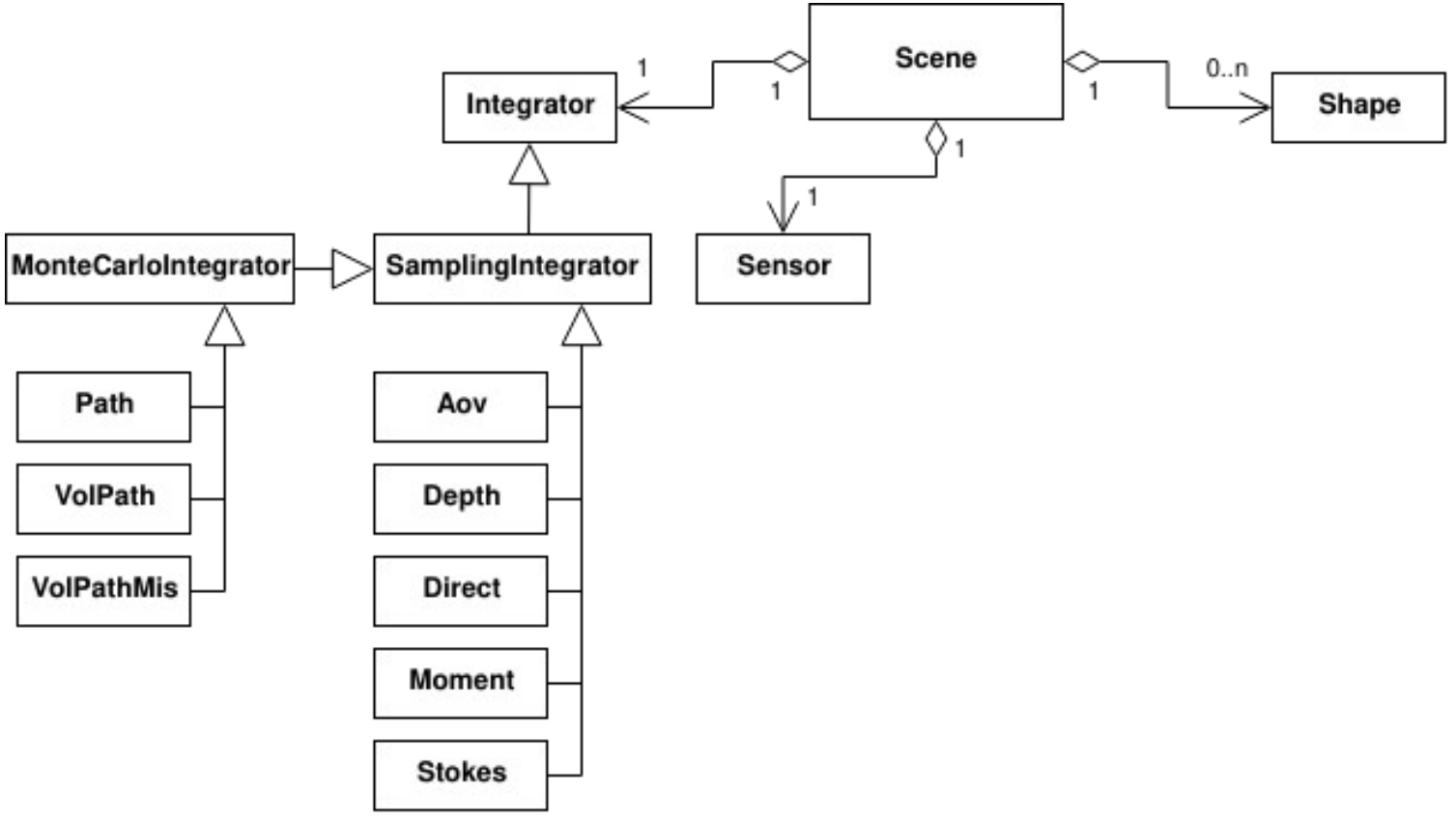
- Fotografía real de Hosek-Wilkie [12] (izq.) y render (der.)

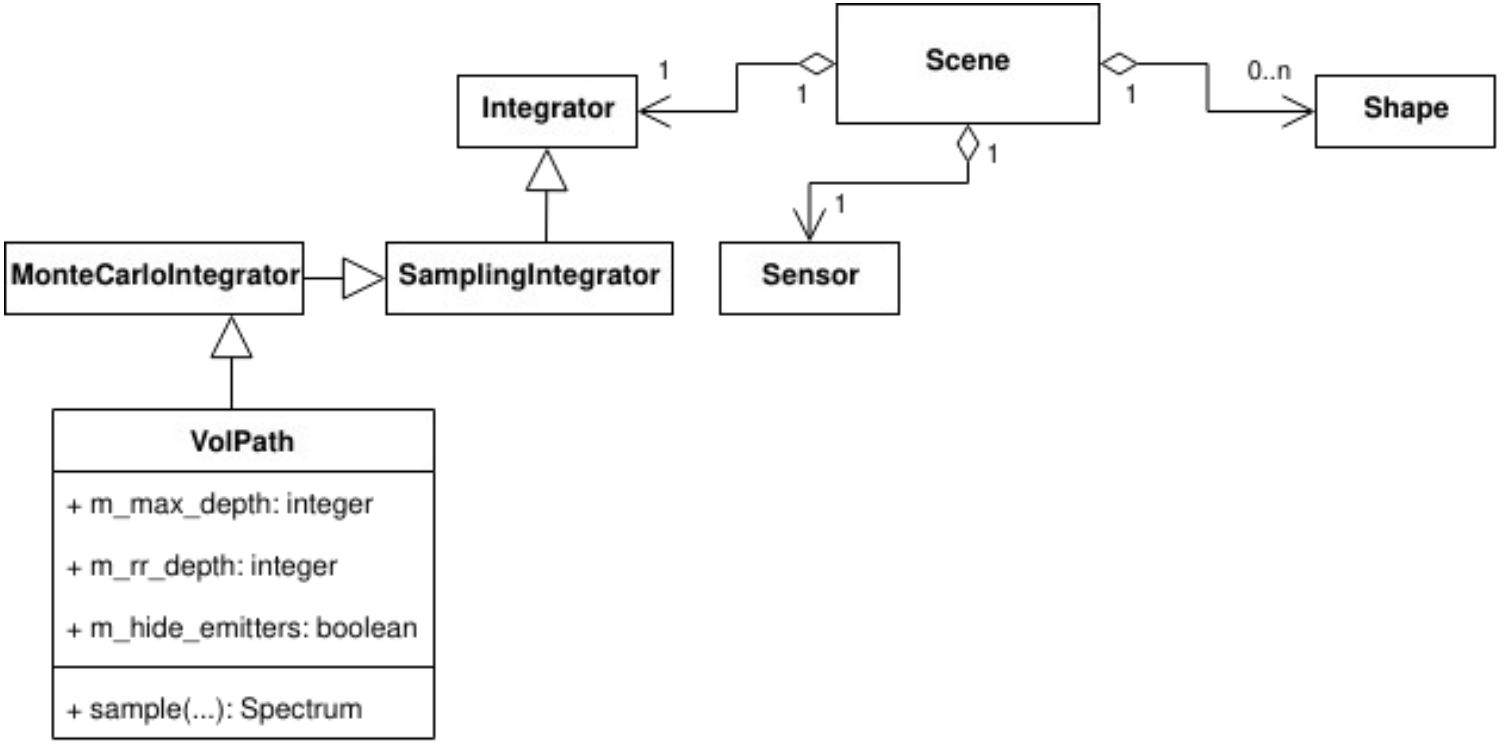
Conclusiones

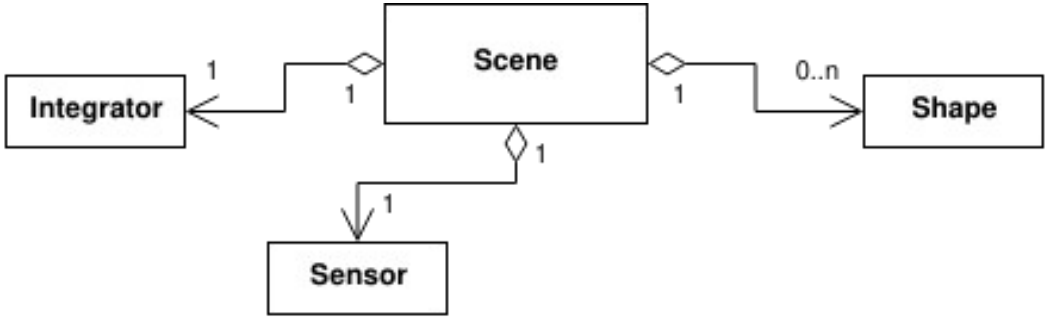
- Dificultad al comprender los modelos físicos por primera vez.
- Gran curva de aprendizaje inicial de Mitsuba 2.
- Ha sido realmente interesante implementar los modelos físicos sobre un proyecto open-source de rendering profesional.
- Satisfacción al obtener los resultados de manera visual.
- Posibilidad de extender el trabajo realizado a otros medios atmosféricos diferentes y añadir efectos atmosféricos como los arcoíris o las auroras boreales.

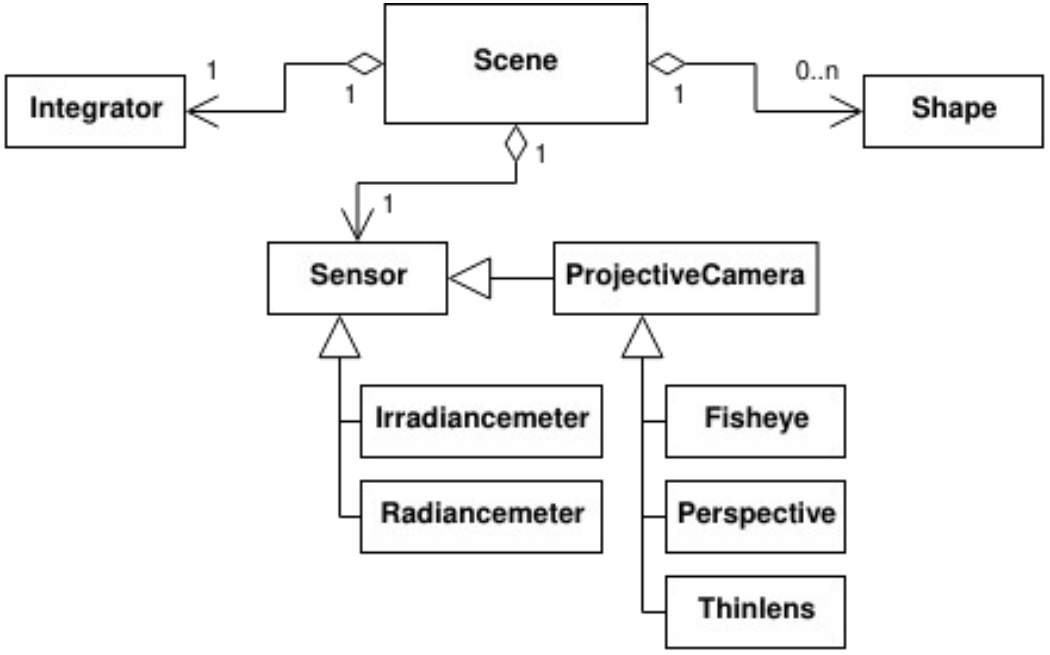
Gracias por la atención

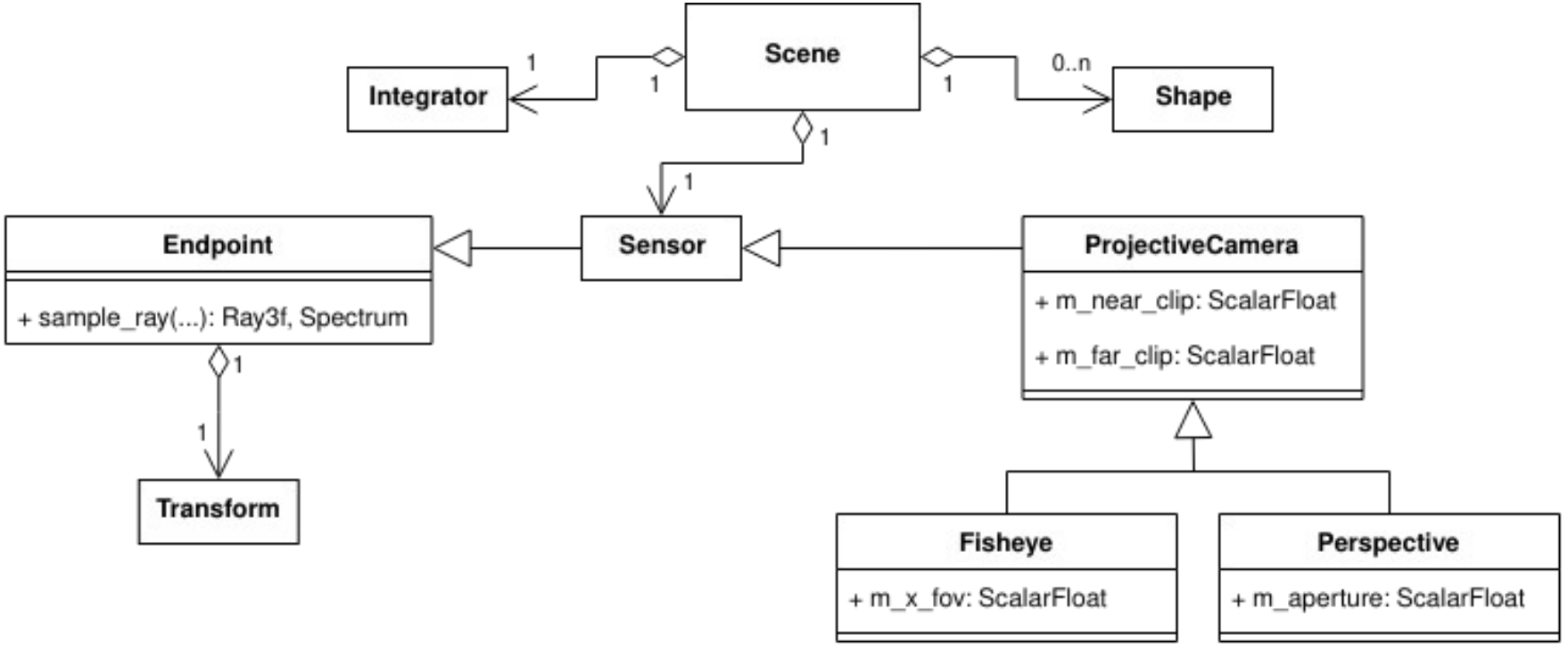

Scene
Integrator
Shape
Sensor
1
1
1
0..n
1
1

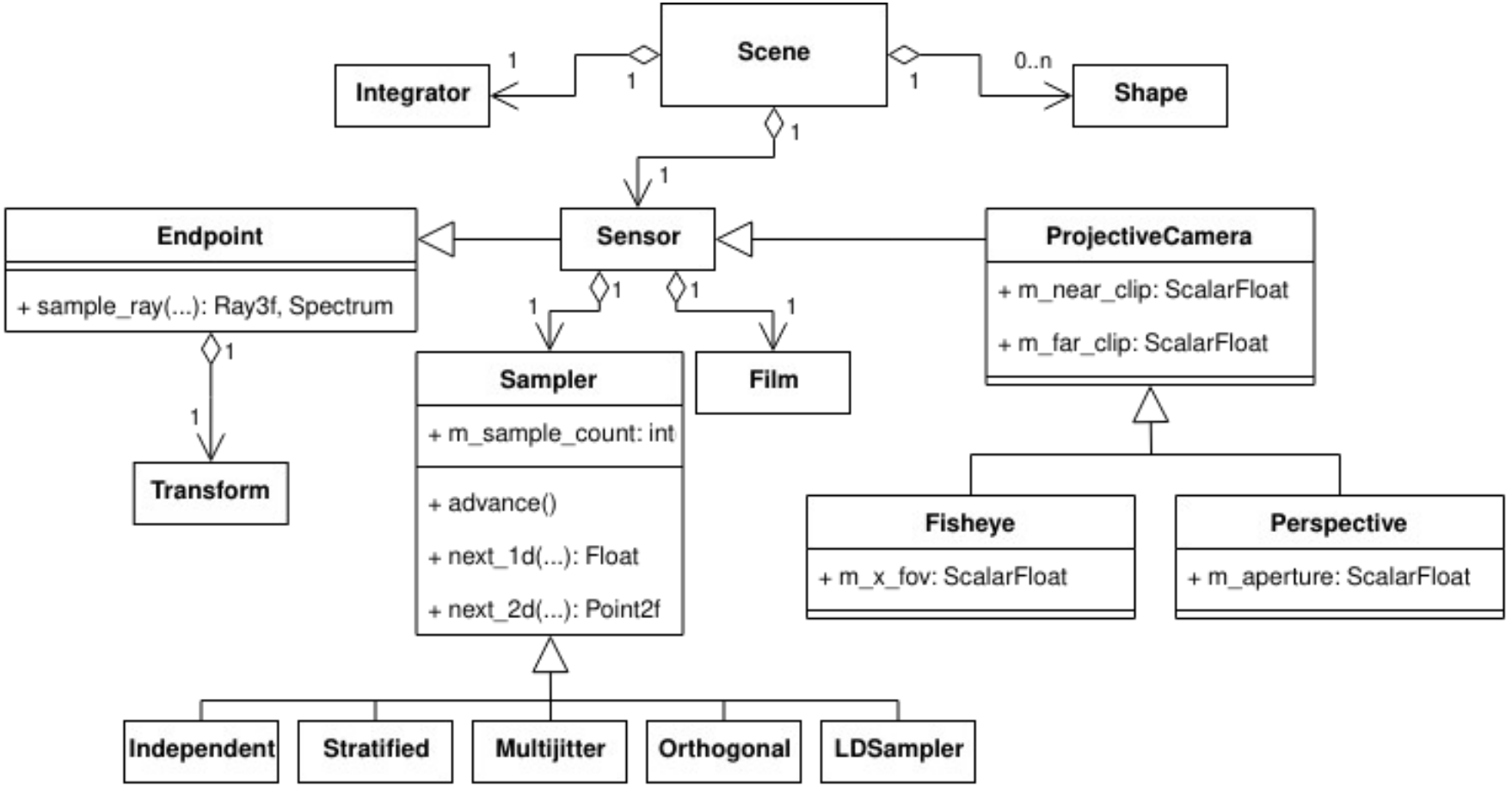

Scene
Integrator
Shape
Sensor
MonteCarloIntegrator
SamplingIntegrator
Path
VolPath
VolPathMis
Aov
Depth
Direct
Moment
Stokes
1
1
1
1
1
1
0..n

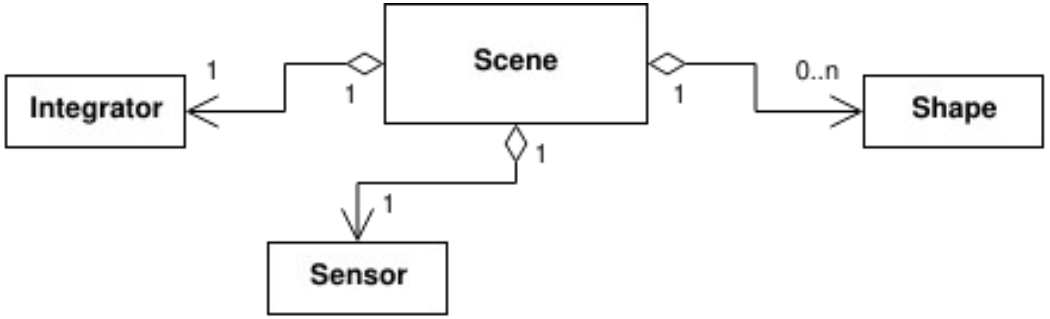

Scene
Integrator
Shape
Sensor
MonteCarloIntegrator
SamplingIntegrator
VolPath
+ m_max_depth: integer
+ m_rr_depth: integer
+ m_hide_emitters: boolean
+ sample(...): Spectrum
1
1
1
1
1
1
0..n

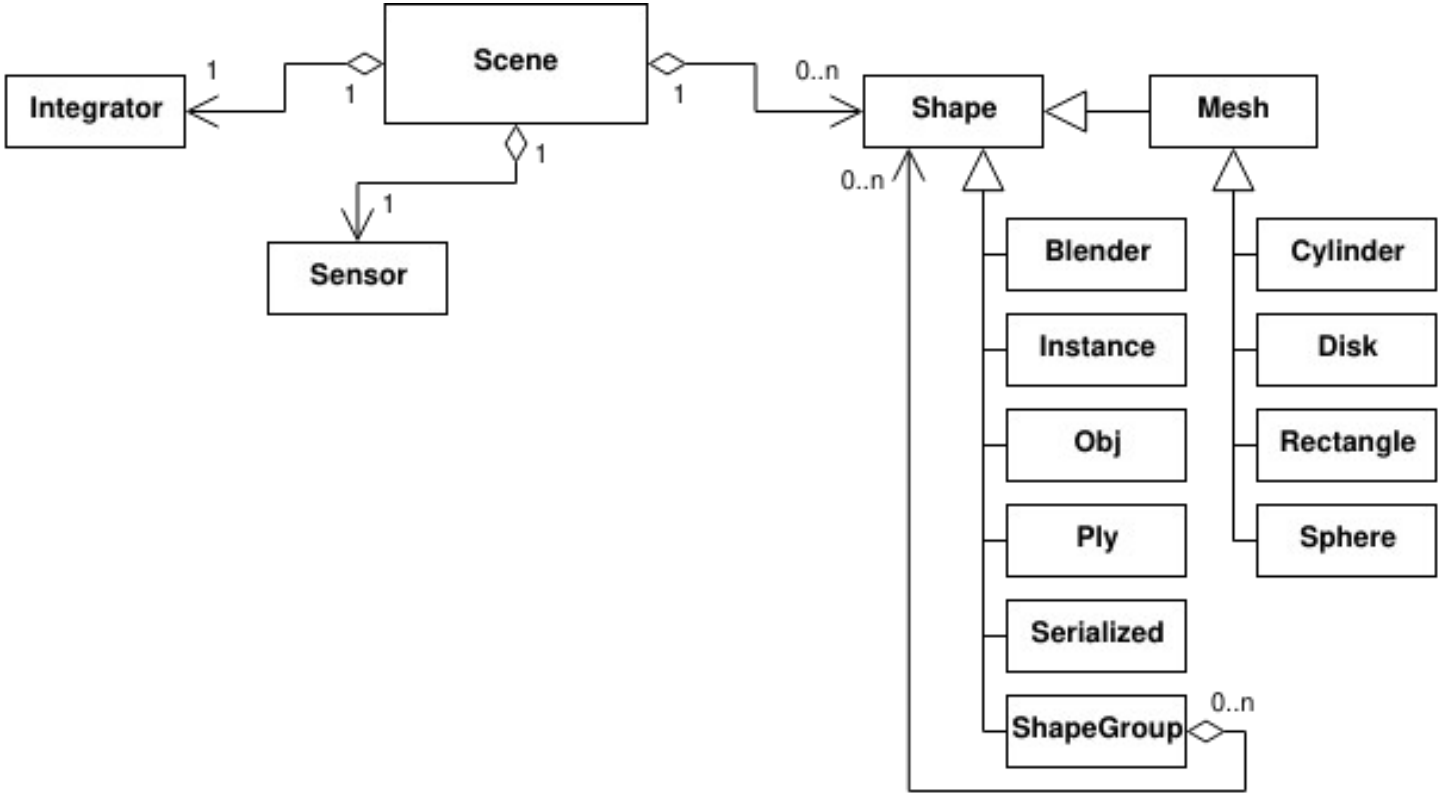

Scene
Integrator
Shape
Sensor
1
1
1
0..n
1
1

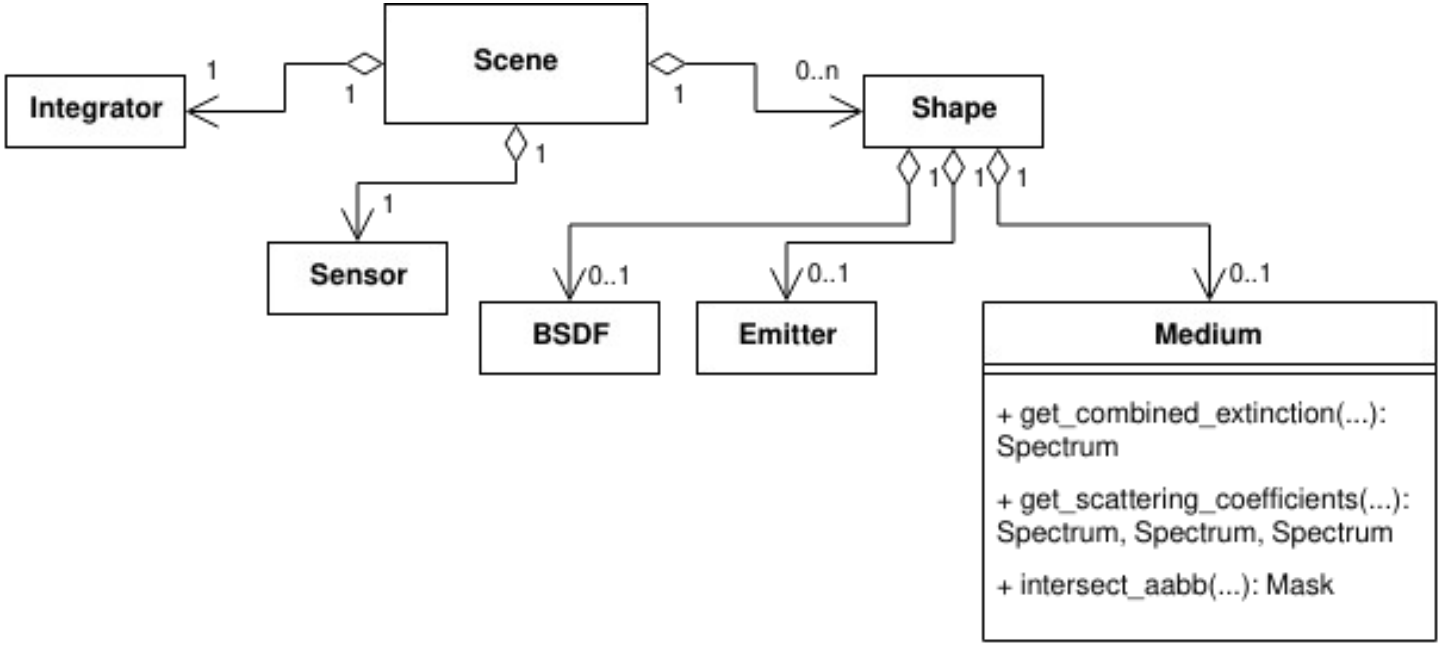

Scene
Integrator
Shape
Sensor
ProjectiveCamera
Irradiancemeter
Radiancemeter
Fisheye
Perspective
Thinlens
1
1
1
1
1
1
0..n

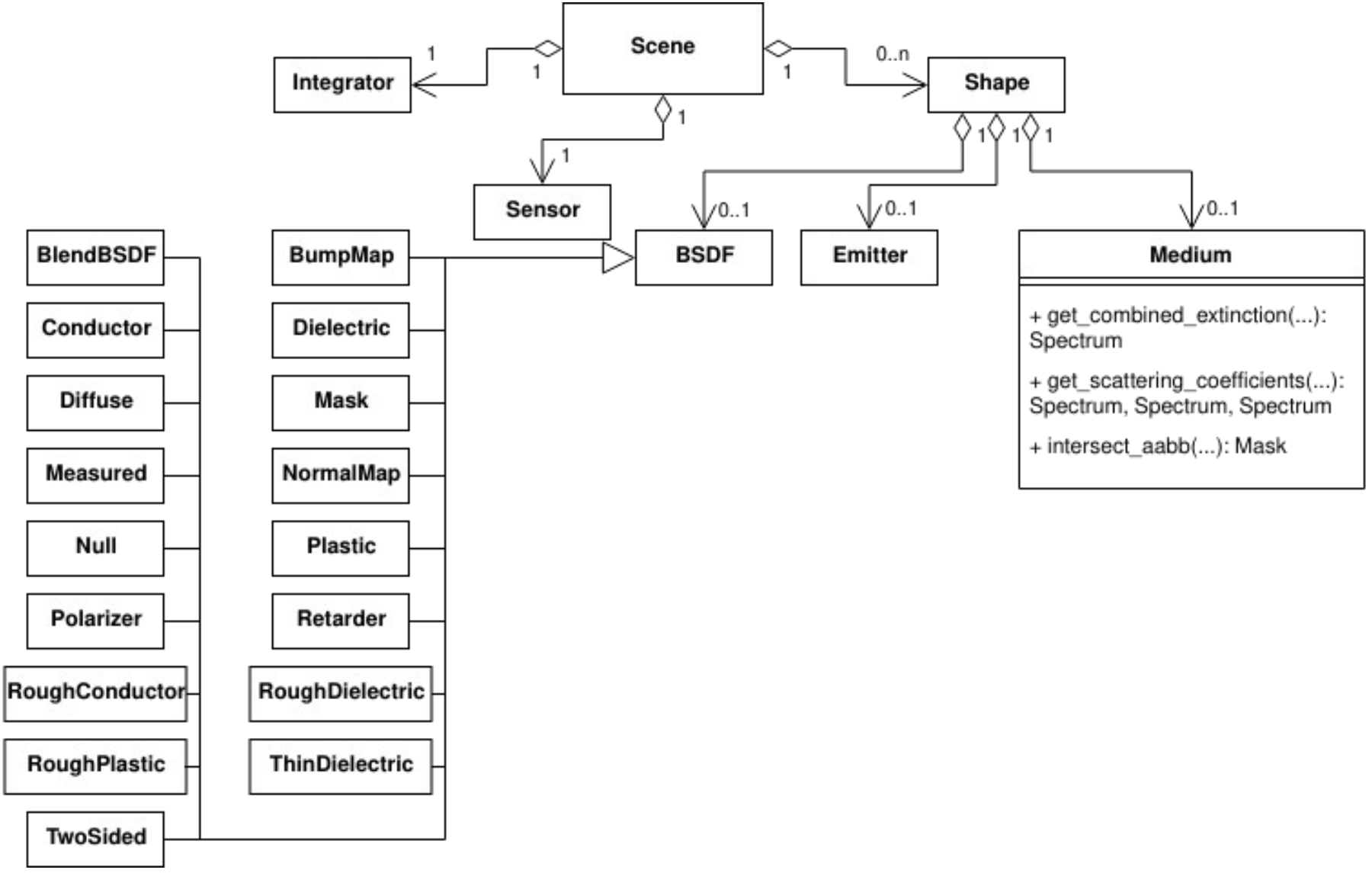

Scene
Integrator
Shape
1
1
1
1
0..n
1
1
Endpoint
+ sample_ray(...): Ray3f, Spectrum
Sensor
ProjectiveCamera
+ m_near_clip: ScalarFloat
+ m_far_clip: ScalarFloat
1
1
Transform
Fisheye
+ m_x_fov: ScalarFloat
Perspective
+ m_aperture: ScalarFloat

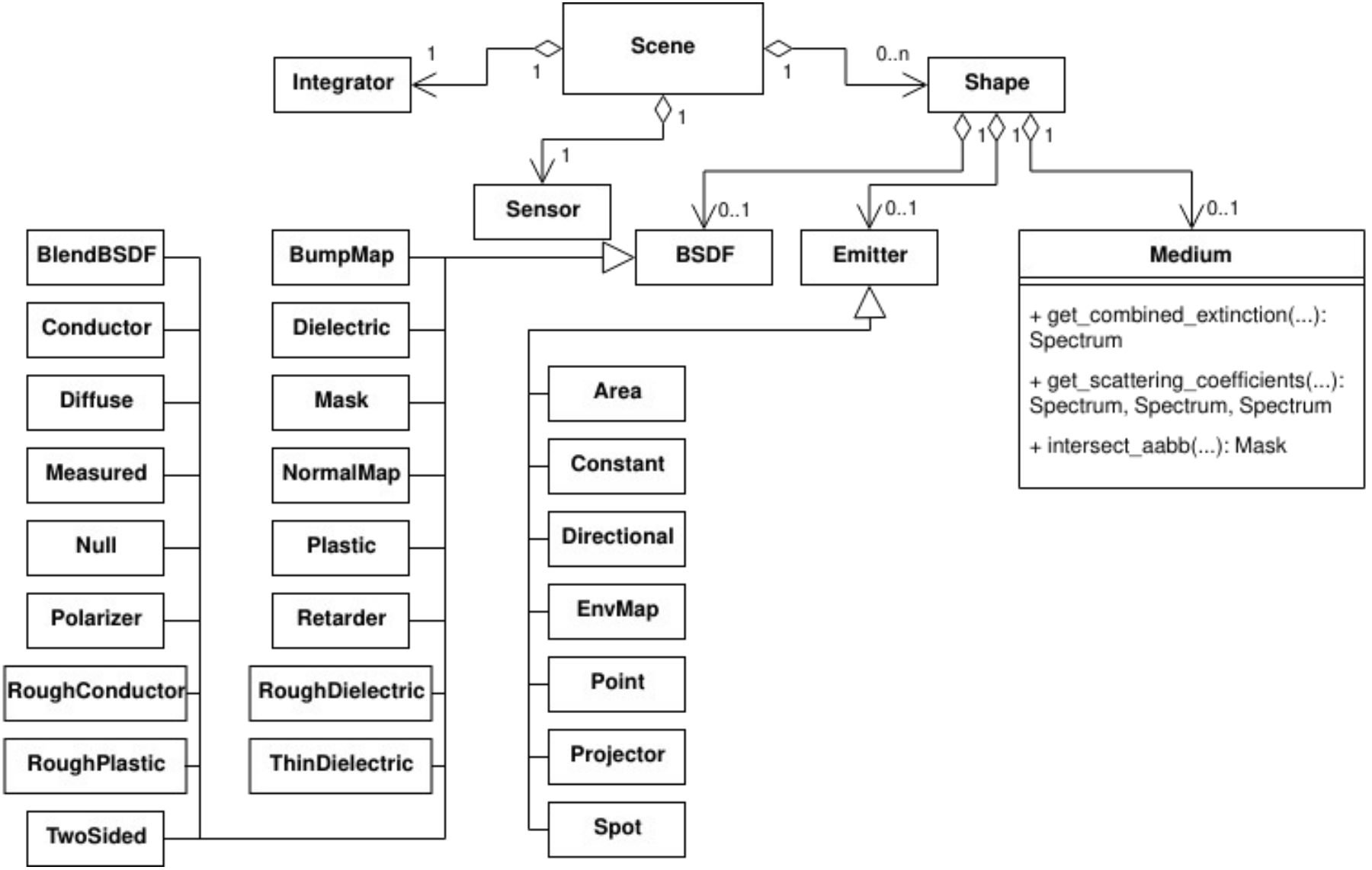

Scene
Integrator
Shape
1
1
1
1
0..n
1
1
Endpoint
+ sample_ray(...): Ray3f, Spectrum
Sensor
ProjectiveCamera
+ m_near_clip: ScalarFloat
+ m_far_clip: ScalarFloat
1
1
1
1
1
1
Transform
Sampler
+ m_sample_count: int
+ advance()
+ next_1d(...): Float
+ next_2d(...): Point2f
Film
Fisheye
+ m_x_fov: ScalarFloat
Perspective
+ m_aperture: ScalarFloat
Independent
Stratified
Multijitter
Orthogonal
LDSampler

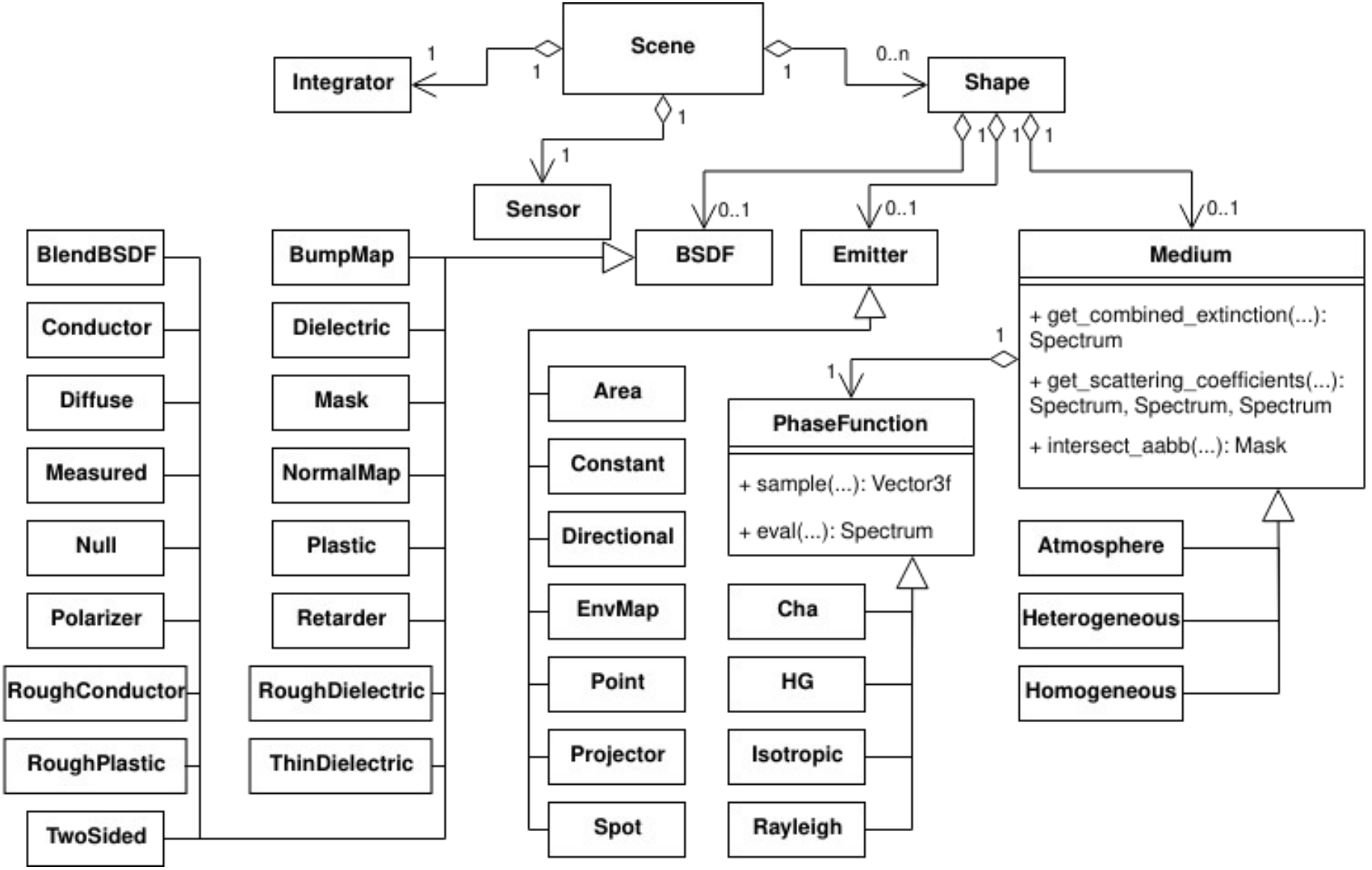

Scene
Integrator
Shape
Sensor
1
1
1
0..n
1
1

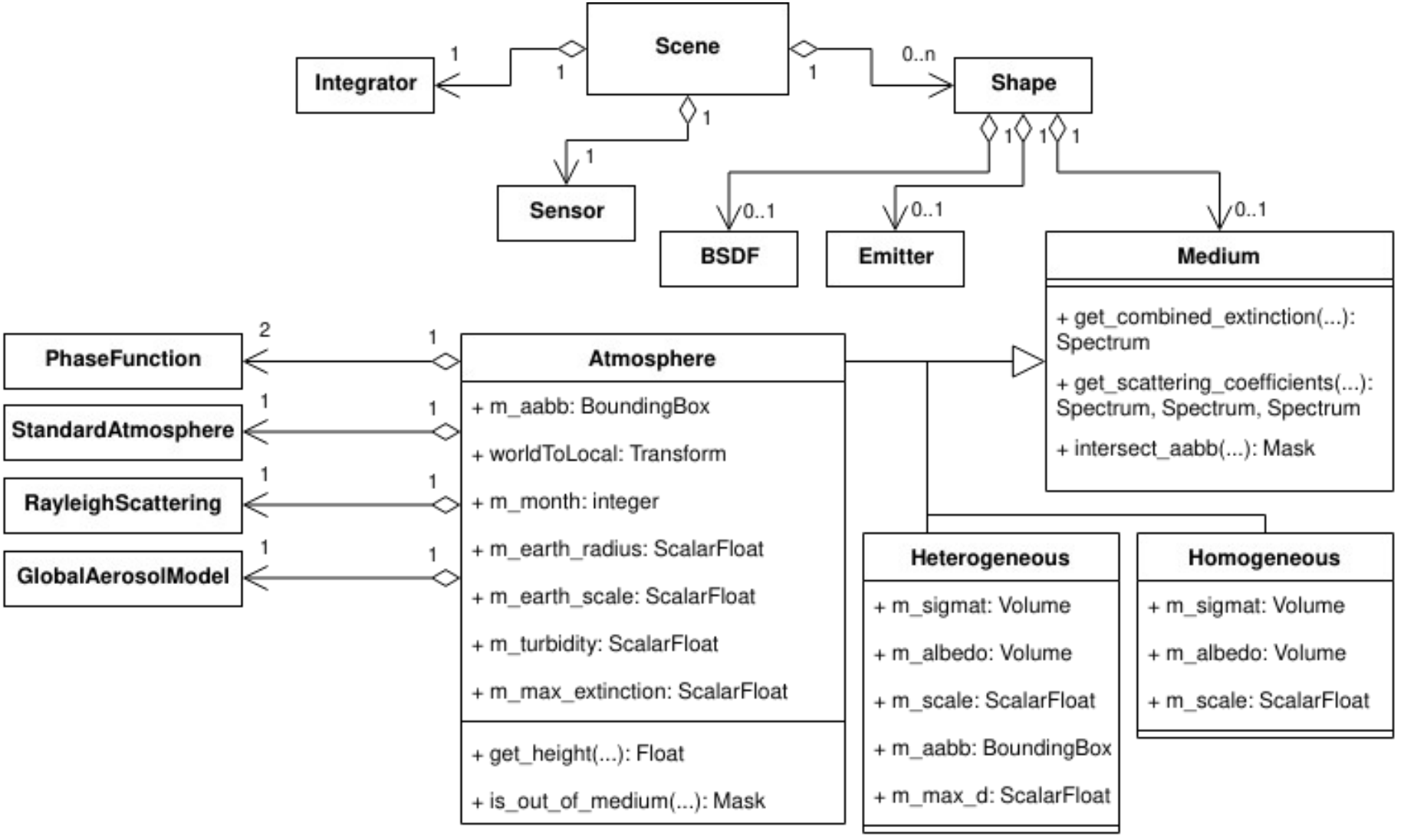

Scene
Integrator
Sensor
Shape
Mesh
1
1
1
1
1
1
0..n
0..n
Blender
Instance
Obj
Ply
Serialized
ShapeGroup
0..n
Cylinder
Disk
Rectangle
Sphere

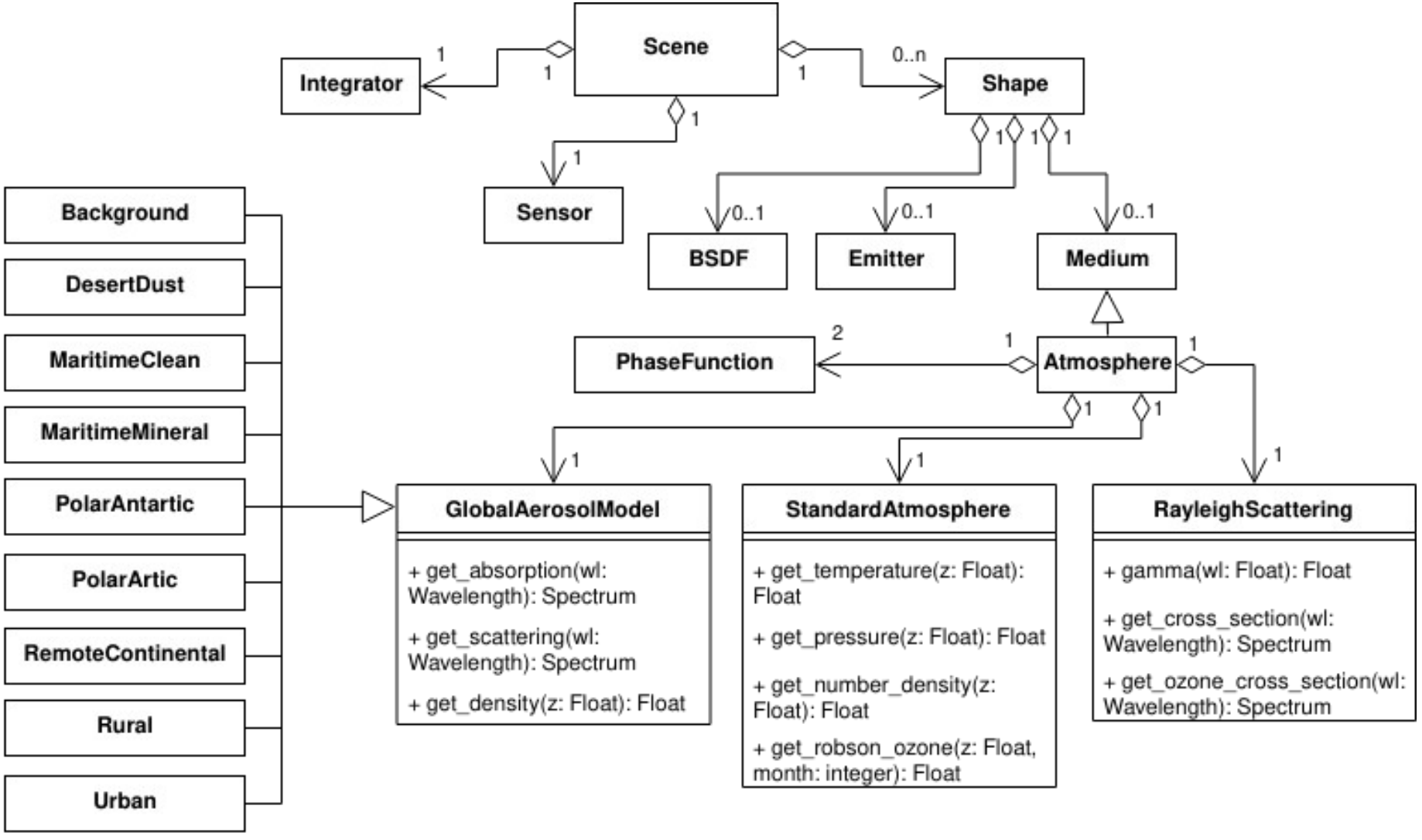

Scene
Integrator
Sensor
Shape
BSDF
Emitter
Medium
1
1
1
1
1
0..n
1
1
1
1
0..1
0..1
0..1
+ get_combined_extinction(...): Spectrum
+ get_scattering_coefficients(...): Spectrum, Spectrum, Spectrum
+ intersect_aabb(...): Mask


Scene
Integrator
Sensor
Shape
BSDF
Emitter
Medium
1
0..n
0..1
+ get_combined_extinction(...): Spectrum
+ get_scattering_coefficients(...): Spectrum, Spectrum, Spectrum
+ intersect_aabb(...): Mask
BlendBSDF
Conductor
Diffuse
Measured
Null
Polarizer
RoughConductor
RoughPlastic
TwoSided
BumpMap
Dielectric
Mask
NormalMap
Plastic
Retarder
RoughDielectric
ThinDielectric


Scene
Integrator
Sensor
Shape
BSDF
Emitter
Medium
1
0..n
0..1
BlendBSDF
Conductor
Diffuse
Measured
Null
Polarizer
RoughConductor
RoughPlastic
TwoSided
BumpMap
Dielectric
Mask
NormalMap
Plastic
Retarder
RoughDielectric
ThinDielectric
Area
Constant
Directional
EnvMap
Point
Projector
Spot
+ get_combined_extinction(...): Spectrum
+ get_scattering_coefficients(...): Spectrum, Spectrum, Spectrum
+ intersect_aabb(...): Mask


Scene
Integrator
Sensor
Shape
BSDF
Emitter
Medium
BlendBSDF
Conductor
Diffuse
Measured
Null
Polarizer
RoughConductor
RoughPlastic
TwoSided
BumpMap
Dielectric
Mask
NormalMap
Plastic
Retarder
RoughDielectric
ThinDielectric
Area
Constant
Directional
EnvMap
Point
Projector
Spot
PhaseFunction
+ sample(...): Vector3f
+ eval(...): Spectrum
Cha
HG
Isotropic
Rayleigh
+ get_combined_extinction(...): Spectrum
+ get_scattering_coefficients(...): Spectrum, Spectrum, Spectrum
+ intersect_aabb(...): Mask
Atmosphere
Heterogeneous
Homogeneous
1
0..n
0..1


Scene
Integrator
Sensor
Shape
BSDF
Emitter
Medium
+ get_combined_extinction(...): Spectrum
+ get_scattering_coefficients(...): Spectrum, Spectrum, Spectrum
+ intersect_aabb(...): Mask
PhaseFunction
StandardAtmosphere
RayleighScattering
GlobalAerosolModel
Atmosphere
+ m_aabb: BoundingBox
+ worldToLocal: Transform
+ m_month: integer
+ m_earth_radius: ScalarFloat
+ m_earth_scale: ScalarFloat
+ m_turbidity: ScalarFloat
+ m_max_extinction: ScalarFloat
+ get_height(...): Float
+ is_out_of_medium(...): Mask
Heterogeneous
+ m_sigmat: Volume
+ m_albedo: Volume
+ m_scale: ScalarFloat
+ m_aabb: BoundingBox
+ m_max_d: ScalarFloat
Homogeneous
+ m_sigmat: Volume
+ m_albedo: Volume
+ m_scale: ScalarFloat


Scene
Integrator
Sensor
Shape
BSDF
Emitter
Medium
PhaseFunction
Atmosphere
Background
DesertDust
MaritimeClean
MaritimeMineral
PolarAntartic
PolarArtic
RemoteContinental
Rural
Urban
GlobalAerosolModel
+ get_absorption(wl: Wavelength): Spectrum
+ get_scattering(wl: Wavelength): Spectrum
+ get_density(z: Float): Float
StandardAtmosphere
+ get_temperature(z: Float): Float
+ get_pressure(z: Float): Float
+ get_number_density(z: Float): Float
+ get_robson_ozone(z: Float, month: integer): Float
RayleighScattering
+ gamma(wl: Float): Float
+ get_cross_section(wl: Wavelength): Spectrum
+ get_ozone_cross_section(wl: Wavelength): Spectrum