Computer Graphics

- Participating Media & Volume Rendering -

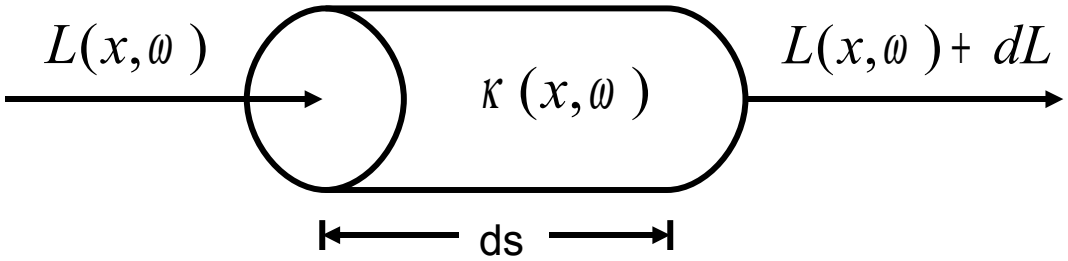
Philipp Slusallek

Motivation

- **Applications**
  - Fog, smoke, clouds, fire, water, …
  - Scientific/medical visualization: CT, MRI
  - Simulations: Fluid flow, temperature, weather, ...
  - Subsurface scattering
- **Effects in Participating Media**
  - Absorption
  - Emission
  - Scattering
    - Out-scattering
    - In-scattering
- **Literature**
  - Klaus Engel et al., *Real-time Volume Graphics*, AK Peters
  - Paul Suetens, *Fundamentals of Medical Imaging*, Cambridge University Press

Absorption

- **Absorption Coefficient κ(x,ω)**
  - Probability of a photon being absorbed at x in direction ω per unit length



$$dL(x,\omega) = -\kappa(x,\omega)L(x,\omega)ds$$

$$\frac{dL}{ds}(x,\omega) = -\kappa(x,\omega)L(x,\omega)$$

  - Optical depth τ of a material of thickness s
    - Physical interpretation:
      - Measure for how far light travels before being absorbed

$$\tau(s) = \int_0^s \kappa(x + t\omega, \omega)dt \quad [= \kappa s, \quad iff \ \kappa = const]$$

Transparency and Opacity

- **Integration Along Ray**

$$\frac{dL}{ds}(x,\omega) = -\kappa(x,\omega)L(x,\omega) \quad and \quad \tau(s) = \int_0^s \kappa(x+t\omega,\omega)dt$$

$$L(x+s\omega,\omega) = e^{-\tau(s)}L(x,\omega) = T(s)L(x,\omega)$$

- **Transparency (or Transmittance)**

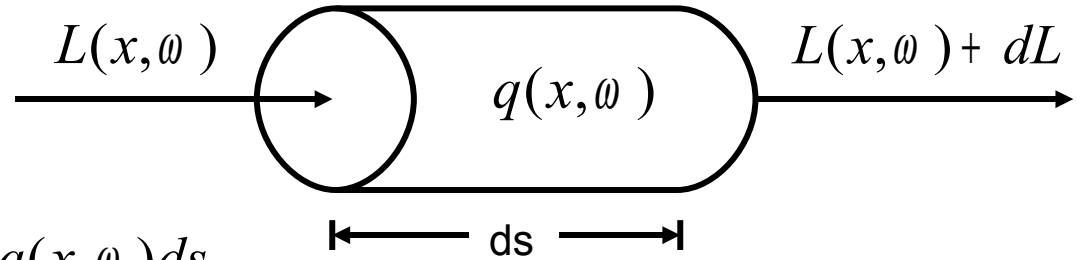
$$T(s) = e^{-\tau(s)} = e^{-\int_0^s \kappa(x+t\omega)dt}$$

- **Opacity**

$$O(s) = 1 - T(s)$$

Emission

- **Emission Coefficient q(x,ω)**
  - Number of photons being emitted at x in direction ω per unit length



$$dL(x,\omega) = q(x,\omega)ds$$

$$\frac{dL}{ds}(x,\omega) = q(x,\omega)$$

Emission-Absorption Model

- **Emission-Absorption Model**
  - Kombines absorption and emission only
- **Volume Rendering Equation**
  - In differential form

$$\frac{dL}{ds}(x,\omega) = -\kappa(x,\omega)L(x,\omega) + q(x,\omega)$$

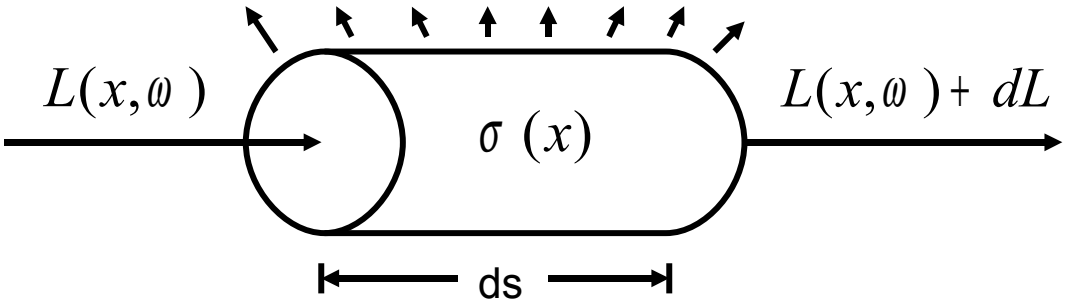
- **Volume Rendering Integral**

$$L(x + s\omega, \omega) = L(x,\omega)e^{-\int_0^s \kappa(t)dt} + \int_0^s q(s')e^{-\int_{s'}^s \kappa(t)dt} ds'$$

  - Incoming light is absorbed along the entire segment
  - Emitted light is only absorbed along the remaining segment
  - Must integrate over emission along the entire segment

Out-Scattering

- **Scattering cross-section σ(x,ω)**
  - Probability of a photon being scattered out of direction per unit length



  - Total absorption (extinction): true absorption plus out-scattering

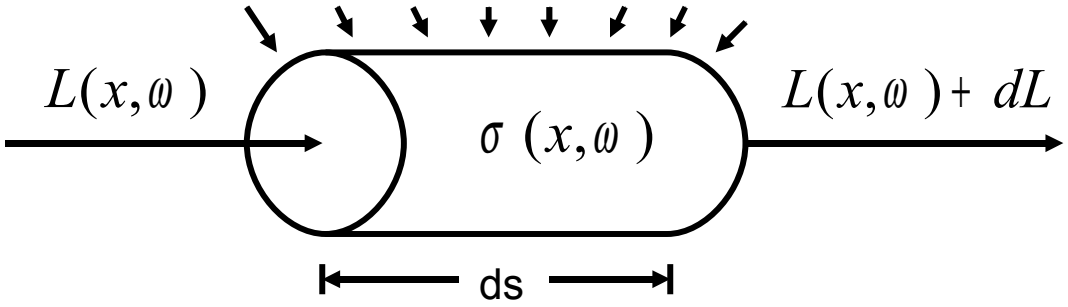
$$\chi = \kappa + \sigma$$

  - Albedo (“Weißheit”, measure for reflectivity or ability to scatter)

$$W = \frac{\sigma}{\chi} = \frac{\sigma}{\kappa + \sigma}$$

In-Scattering

- **Scattering cross-section σ(x,ω)**
  - Number of photons being scattered into path per unit length
  - Depend on scattering coefficient (probability of being scattered) and the phase function (directional distribution of out-scattering events)



$$j(x,\omega) = \int_{S^2} \sigma(x,\omega_i)\, p(x,\omega_i,\omega)\, L(x,\omega_i)\, d\omega_i$$

  - **Total Emission**: true emission q plus in-scattering j

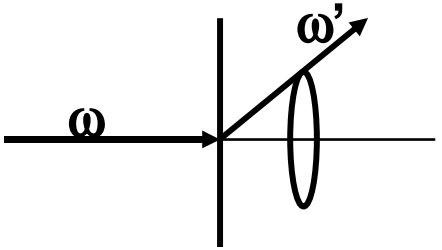
$$\eta(x,\omega) = q(x,\omega) + j(x,\omega)$$

  - **Phase function** (essentially the BRDF for volumes)

$$p(x,\omega_i,\omega)$$

Phase Functions

- **Phase angle is often only relative to incident direction**
  - $\cos\theta = \omega \cdot \omega'$
- **Reciprocity and energy conservation**

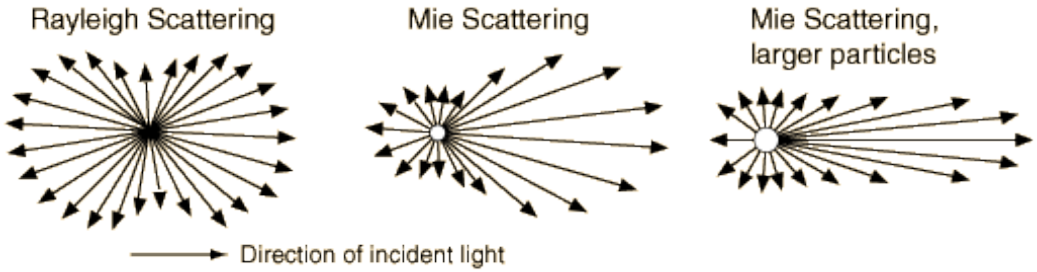


$$p(x,\omega_i,\omega) = p(x,\omega,\omega_i)$$

$$\frac{1}{4\pi}\int_{S^2} p(x,\omega_i,\omega)\,d\omega = 1$$

- **Phase functions**
  - Isotropic

$$p(\cos\theta) = 1$$

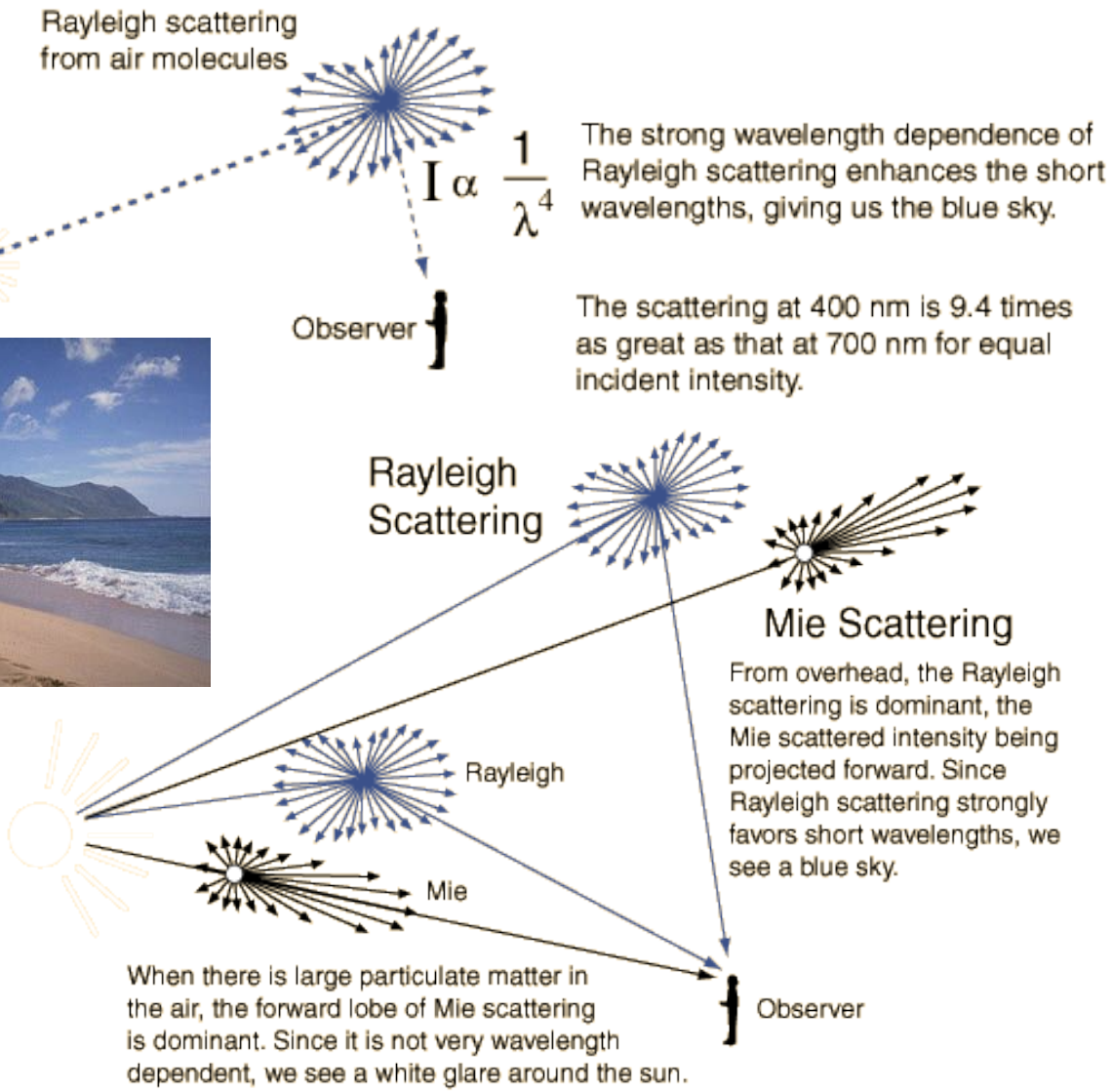


  - Rayleigh (small molecules)
    - Strong wavelength dependence

$$p(\cos\theta) = \frac{3}{4}\frac{1+\cos^2\theta}{\lambda^4}$$

  - Mie scattering (larger spherical particles)

Rayleigh and Mie Scattering



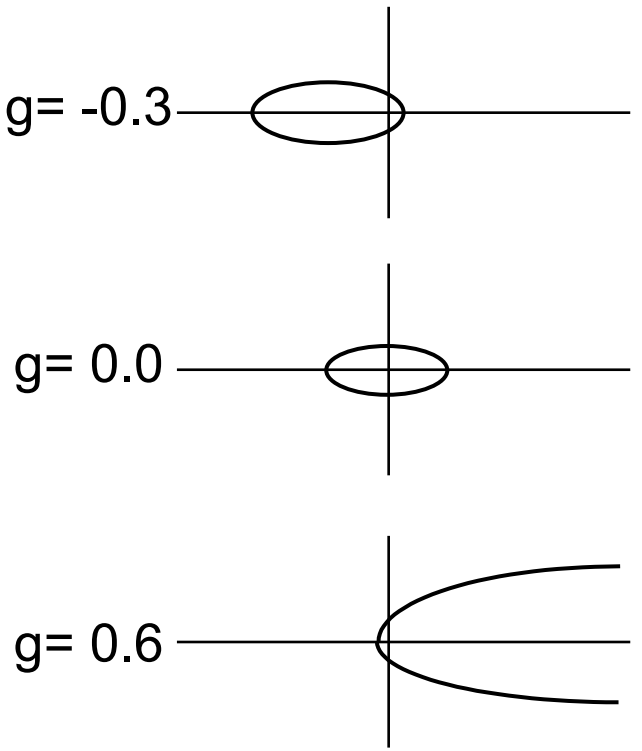
Henyey-Greenstein Phase Function

- **Empirical Phase Function**
  - Often used for interstellar clouds, tissue, and similar material

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

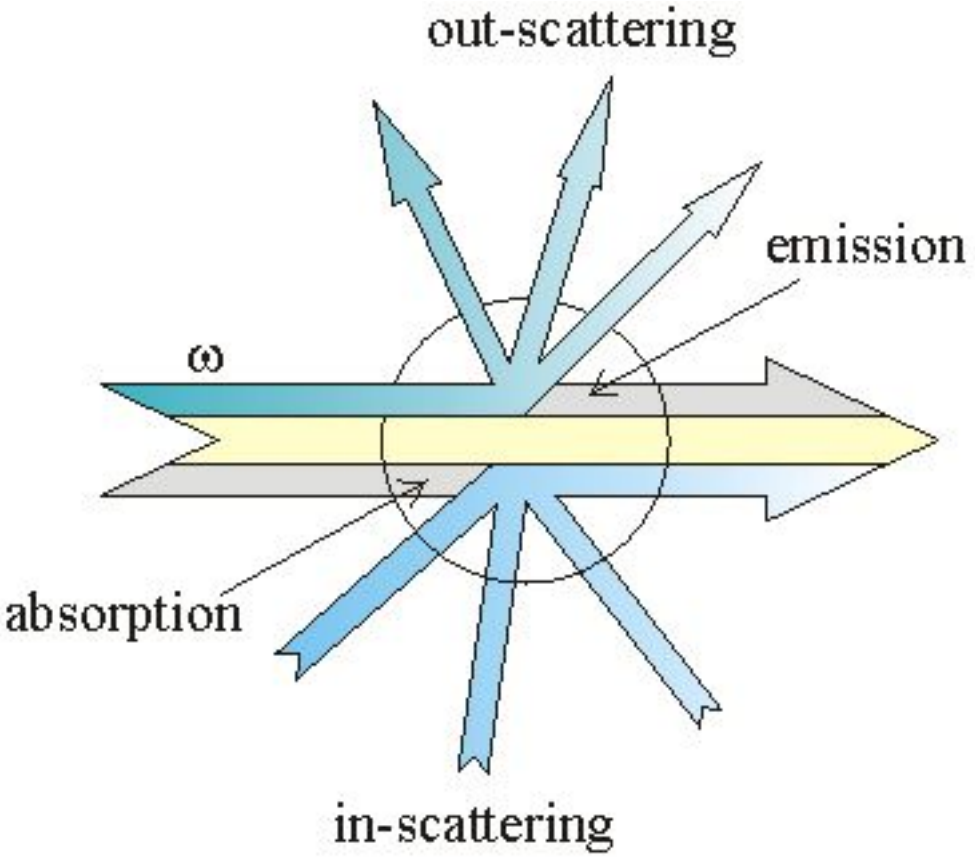
  - Average cosine of phase angle

$$g = 2\pi \int_0^{\pi} p(\cos\theta)\cos\theta \; d\theta$$



Summary

- **Scattering in a volume**



Full Volume Rendering

- **Full Volume Rendering Equation**

$$\omega \cdot \nabla_x L(x,\omega) = \frac{\partial L(x,\omega)}{\partial s} = -\chi(x,\omega)L(x,\omega) + q(x,\omega)$$
$$+ \int_{S^2} \sigma(x,\omega_i) p(x,\omega_i,\omega) L(x,\omega_i) d\omega_i$$

$$\nabla_x = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right) \quad \text{at point x}$$

- **Full Volume Rendering Integral**

$$L(x + s\omega, \omega) = \int_0^s e^{-\int_{s'}^{s} \chi(x+t\omega,\omega)dt} \eta(x + s'\omega, \omega) ds'$$

Attenuation: (absorption & out-scattering)

Source Term: in-scattering,emission, and background ($\eta(0,\omega) = L(x,\omega)\delta(x)$)

Simple Atmosphere Model

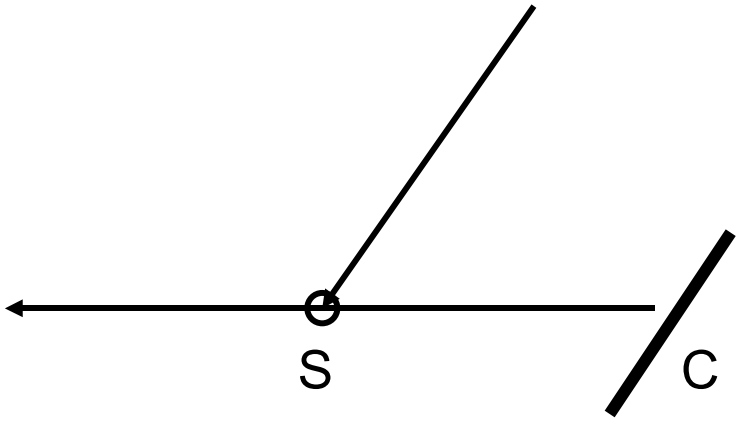
- **Assumptions**
  - Homogeneous media (κ = const)
  - Constant source term q (ambient illumination)

$$\frac{\partial L(s)}{\partial s} = -\kappa L(s) + q$$

$$L(s) = e^{-\kappa s} C + \int_0^s e^{-\kappa s'} q \, ds'$$

$$L(s) = e^{-\kappa s} C + \left(1 - e^{-\kappa s}\right) q$$

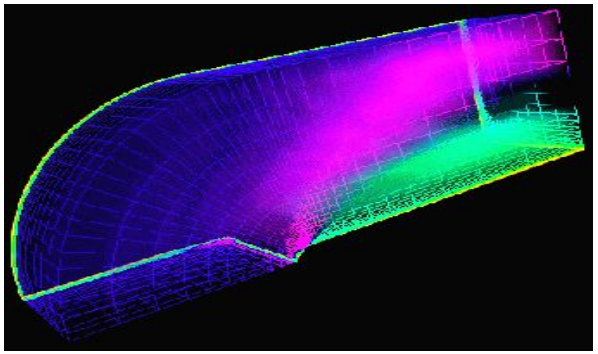
$$L(s) = T(s) C + (1 - T(s)) q$$



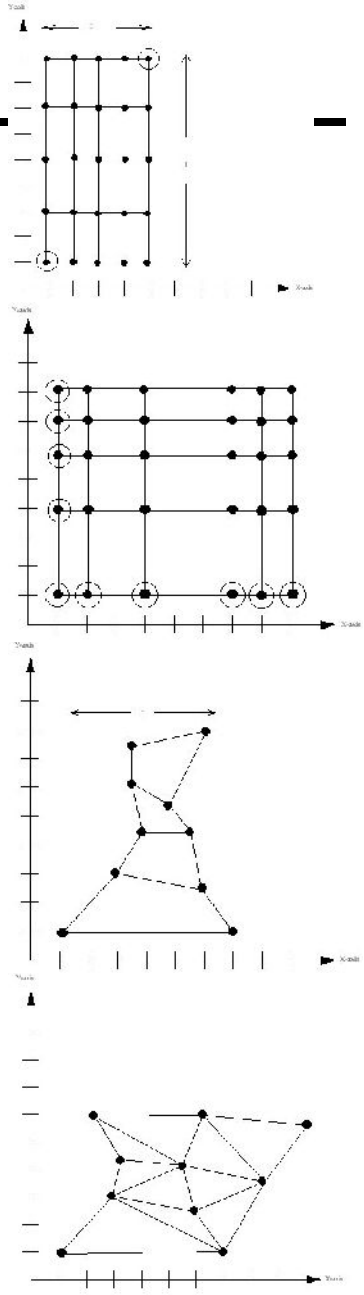
- **Fog and Haze (in OpenGL)**
  - Affine combination of background and fog color
  - Depending on distance

Volume Representations

- **Simple shapes with procedural solid texture**
  - Ellipsoidal clouds with sum-of-sines densities
  - Hypertextures [Perlin]
- **3D array**
  - Regular (uniform) or rectilinear (rectangular)
  - CT, MRI
- **3D meshes**
  - Curvilinear grid (mapping of regular grid to 3D)
    - “Computational space” is uniform grid
    - “Physical space“ is distorted
    - Must map between them (through Jacobian)
  - Unstructured meshes
    - Point clouds
    - Often tesselated into tetrahedral mesh)

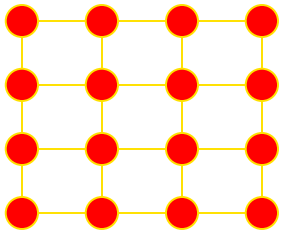


Curvilinear grid

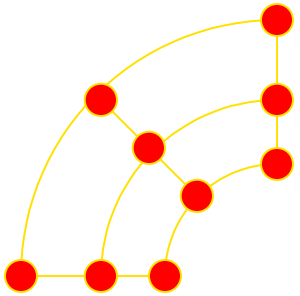


Volume Organization

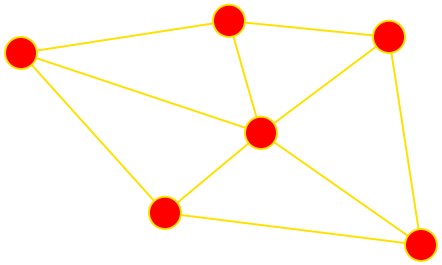
- **Rectilinear Grid:**
  - Wald et al.
  - Implicit kd-trees



- **Curvilinear Grid:**
  - Warped Rectilinear Grid
  - Hexahedral cells

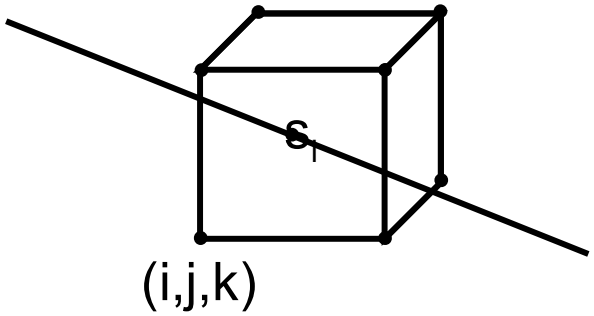


- **Unstructured Mesh:**
  - Tetrahedral cells



Discrete Volume Data

- **Interpolation from scalar grid data (v)**
  - Data given only at grid locations
  - Must derive smooth function in between by interpolation



- **Typically trilinear interpolation from corner values**

$$v(s_l) = trilinear(v, i, j, k, x(s_l))$$

- **Higher order interpolation**
  - Gives smoother results
- **Transfer Functions (e.g. $\kappa(v), q(v)$)**
  - Map scalar value to optical properties
  - Before (pre-) or after (post-classification) interpolation

Piecewise Integration

- **Split of entire segment into pieces (s_0, s_1, ..., s_n)**
- **Consider sub-segment [s_{i-1}, s_i]**

$$L(s_i) = L(s_{i-1})T(s_{i-1}, s_i) + \int_{s_{i-1}}^{s_i} q(t)T(t, s_i)dt$$

  - New notation

$$T_i = T(s_{i-1}, s_i), \quad c_i = \int_{s_{i-1}}^{s_i} q(t)T(t, s_i)dt$$

  - Yields

$$L(s) = L(s_n) = L(s_{n-1})T_n + c_n = (L(s_{n-2})T_{n-1} + c_{n-1})T_n + c_n = \cdots$$

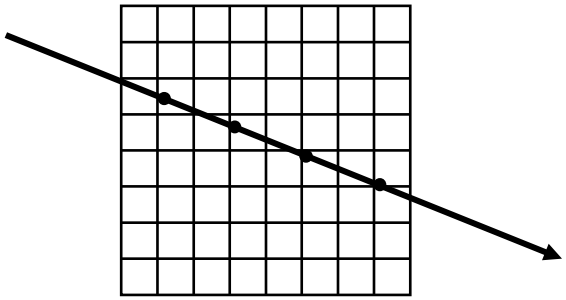
$$L(s) = \sum_{i=0}^{n} c_i \prod_{j=i+1}^{n} T_j \quad \text{with } c_0 = L(s_0)$$

  - Often simplification over equidistant segments

$$T_i \approx e^{-\kappa(s_i)\Delta x}, \quad c_i \approx q(t)\Delta x$$

Compositing Along a Ray

- **Incremental compositing algorithm**
  - As seen from the viewer (s_n is at front)
- **Two Approaches**
  - Front to back (start at s_n) and back to front (start at s_0)
  - Accumulate color and opacity

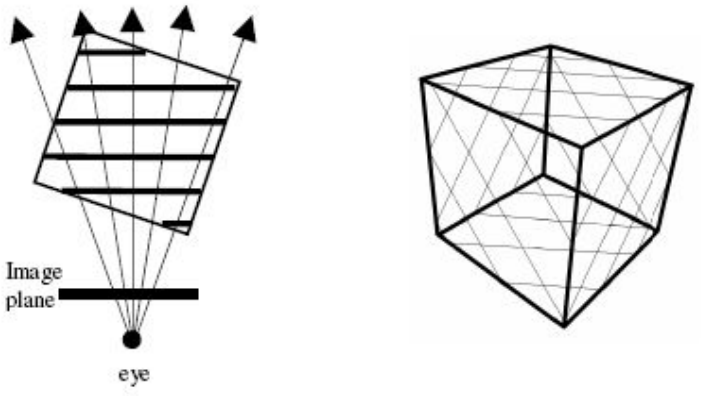


- **Algorithm (front to back)**
  - Allows for early ray termination
  - $C = C_n$, $\alpha=0$ (Opacity)
  - for (i=n-1; i >= 0; i--)
  - $C \mathrel{+}= (1-\alpha)*c_i$
  - $\alpha \mathrel{+}= \alpha + (1-\alpha)(1-T_i)$
  - if (α > threshold) break
  - $C= (1-\alpha)C_{background}$

- **Algorithm (back to front)**
  - Does not allow for termination
  - $C = C_{background}$
  - for (i=0; i <= n; i++)
  - $C= (1-T_i)C + c_i$

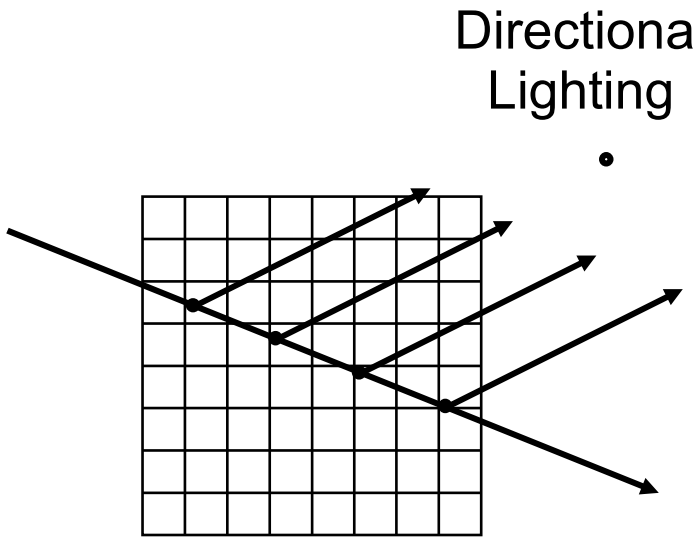
Volume Rendering on GPU

- **Volume Rendering with 3D-Textures**
  - Given volume data set as 3D texture
  - Slice bounding box of 3D texture with planes parallel to viewing plane
  - Render with back to front approach
    - With compositing set appropriately (does not need Alpha buffer)
    - FB_color = FB_color * (1-fragment_alpha) + fragment_color
- **Using 2D Texture**
  - Same technique but use 2D slices of of volume directly
  - Needs three copies (xy, xz, yz) to always use best orientation



Single Scattering

- **Single scattering approximation**
  - Compute illumination via shadow ray
    - Accumulate transparency along the way
    - Multiply with scattering coefficient, phase function, and light radiance
  - Accumulate front to back
    - Illumination from light source
    - Weight with transparency
    - Accumulate transparency
  - Add background illumination times transparency



$T=1$
$L=0$
for ($s=0$; $s < 1$; $s+= ds$)
  $j= \sigma(s) * p(\omega, \omega_L) * L_s * T_S$
  $L \mathrel{+}= T*j*ds$
  $T \mathrel{*}= (1- \mathrm{T}(v(s)))$
$L \mathrel{+}= T * L_0$

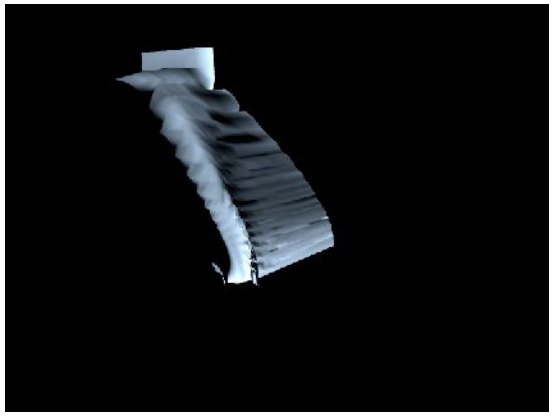
Shadow Ray:
$T_S=1$
for ($t=0$; $t < 1$; $t+= dt$)
  $T_S \mathrel{*}= (1- \mathrm{T}(v(t)))*dt$

Multiple Scattering

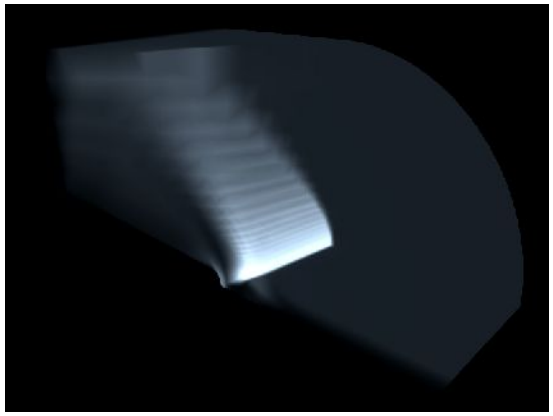
- **Highly computationally demanding**
  - Zonal method (FE-Technique) [Rushmeier'87]
    - Assume constant, isotropic scattering in voxels
    - Set up linear system (a la radiosity) and solve numerically
    - Also includes surface interactions (SS, SV, VS, VV)
  - P-N (P_N) method [Kajiya'84]
    - Represent light distribution at each point in Spherical Harmonics (SH)
    - Compute interactions of SH-coefficients an solve numerically
  - Discrete Ordinate method [Languénou'95]
    - Choose M fixed directions to redistribute energy in
    - Can generate "ray effects" due to fixed directions
      - Should distribute in solid angle
  - Diffusion process [Stam'95]
    - Assumes optically dense medium → much scattering → uniform diffusion
    - Recently also used for sub-surface scattering approximation
    - E.g. computes Point Spread Function (PSF)

Volume Visualization Techniques

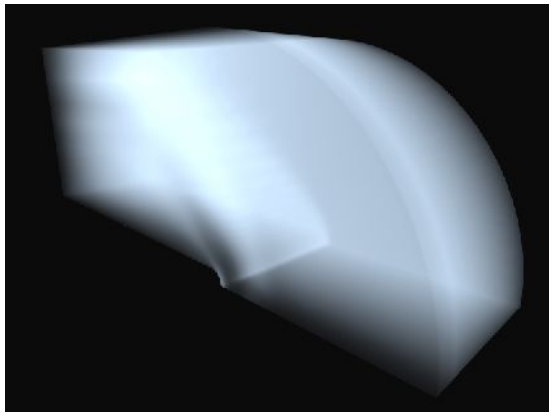
- **Rendering Volume Data**
  - Isosurface Rendering (implicit surface)
  - Maximum-Intensity-Projection
    - Render the larges volume value along a ray
  - Direct or Emission-Absorption Volume Rendering (x-ray)



Isosurface Rendering



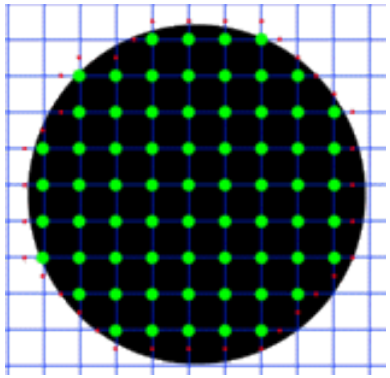
Maximum-Intensity-P.

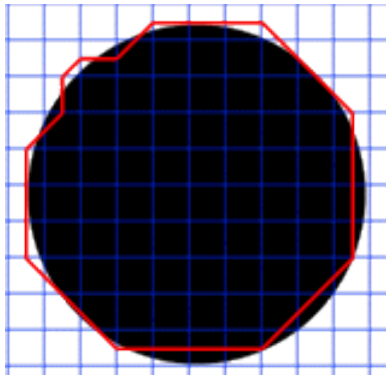


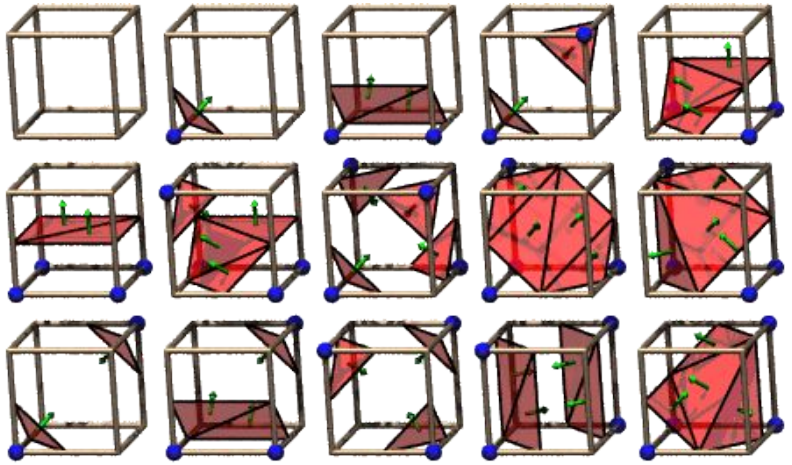
E-A Volume Rendering

Indirect Volume Rendering

- **Iso-Surfaces**
  - Compute iso-surface for v(x,y,z)= C and shade as normal
- **Ray Tracing**
  - Intersect ray with cubic surface defined by values at vertices
  - Several accurate and/or fast algorithms
- **Marching Cubes algorithm**
  - Iterate over all voxels
  - Classify voxel into 15 classes (by symmetry) ➔ surface topology
  - Compute vertex location by interpolation
  - Render as triangle mesh





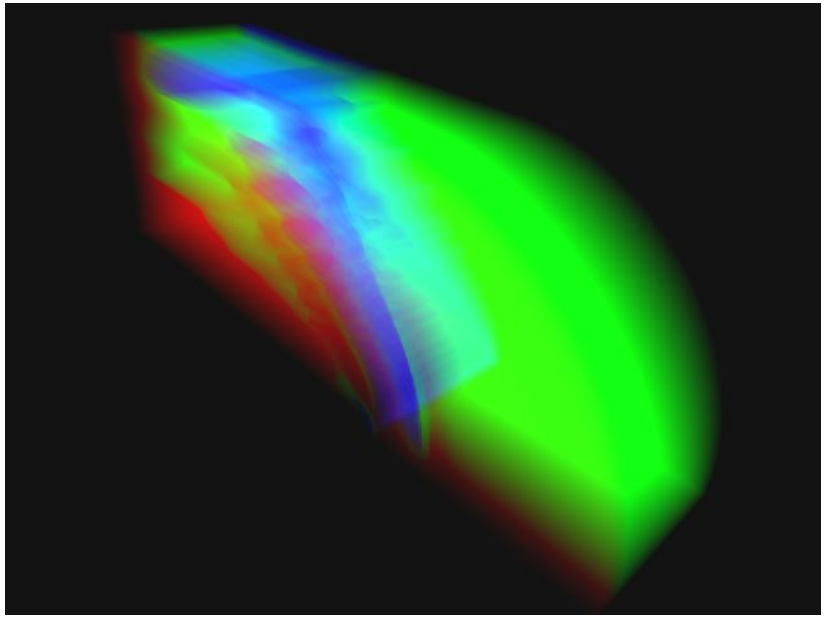


The 15 Cube Combinations

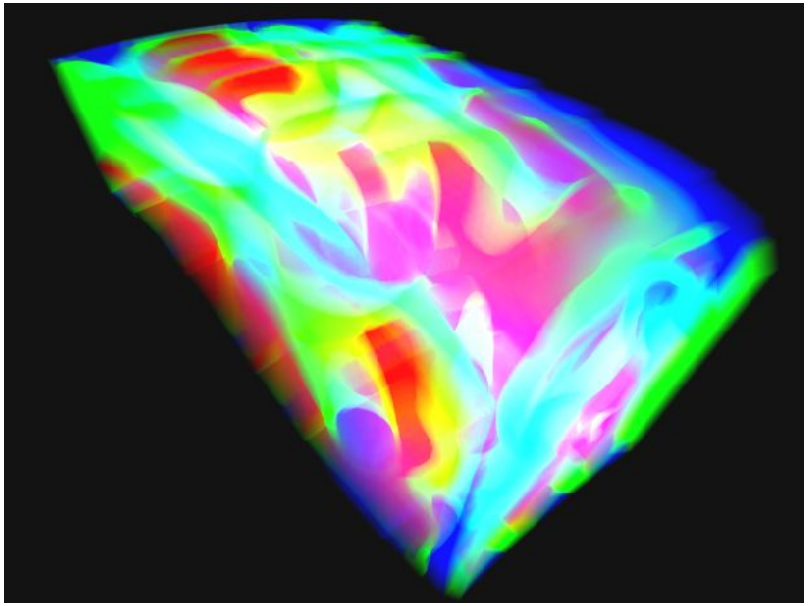
Volumes and Surfaces

- **Interactions**
  - Surface/Volume
    - Intersect with surfaces → ray segment
    - Perform volume rendering along segment
    - Add contribution from surface
    - Must handle surfaces within volumes correctly
  - Volume/Volume
    - Parallel traversal necessary if volumes overlap
      - Opacity combines from both volumes
- **Comparison**
  - Surfaces:
    - Complex traversal operations
    - Single intersection per ray → few complex shading operations
  - Volumes
    - Often simple traversal
    - Constantly shading but often simple shading algorithms

Results [Marmitt]

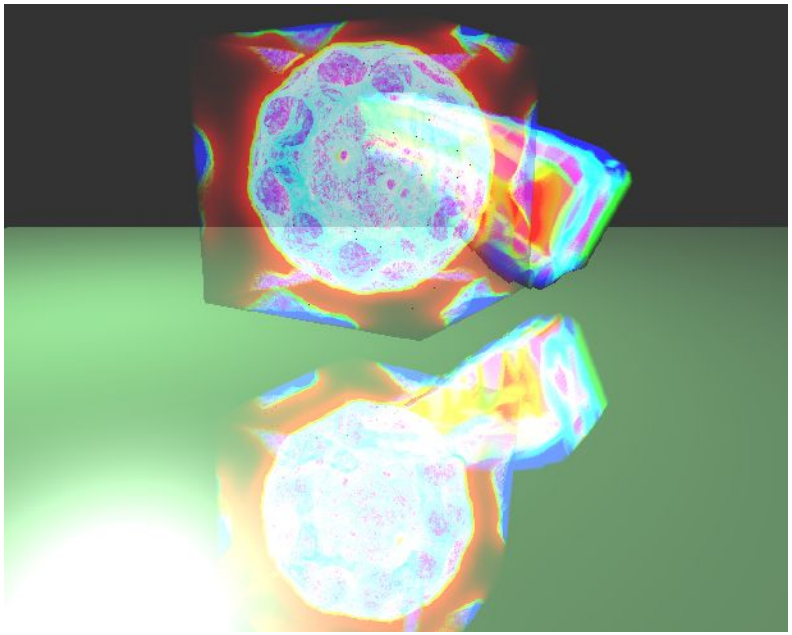


left: Blunt-fin (tetra),
high-quality rendering
(~ 4 fps, 640x480, 16 Cores)

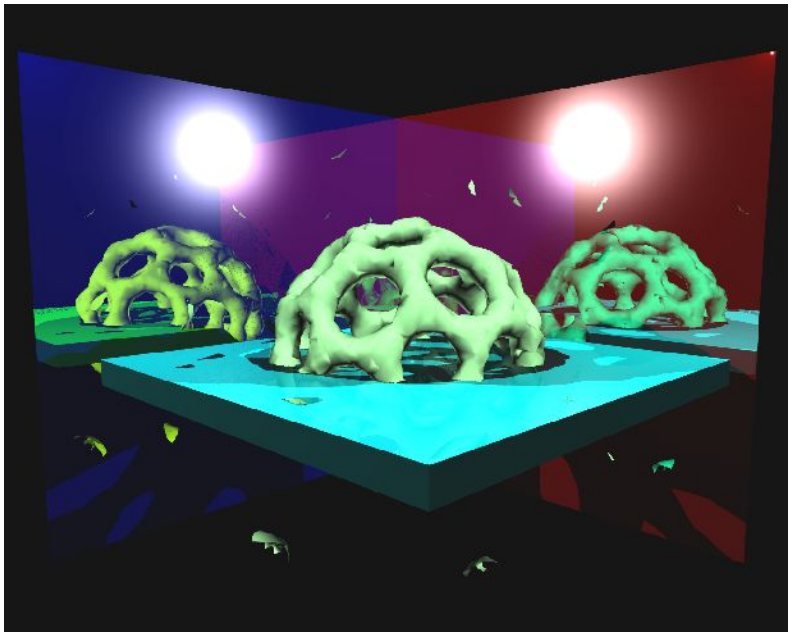


right: Combustion chamber,
high-quality rendering
(~ 4 fps , 640x480, 16 Cores)

Results [Marmitt]



left: Buckyball and Combustion blended together (~ 2 fps , 640x480, 16 Cores)



right: Buckminster Fulleren (Buckyball) iso-surface in a polygonal environment (~ 3 fps , 640x480, 16 Cores)