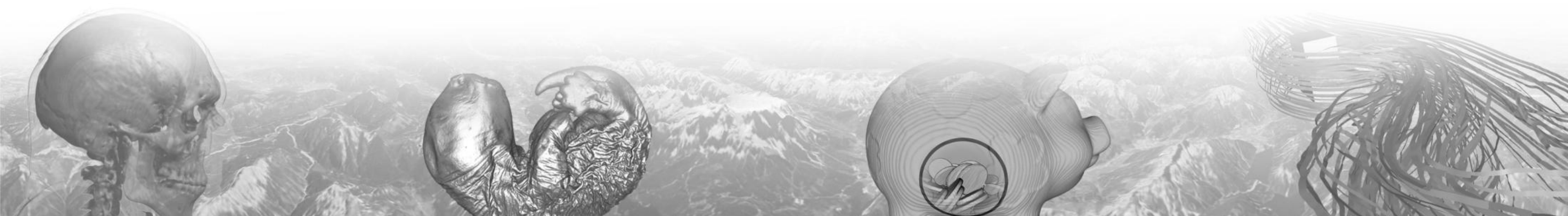
Interactive Methods in Scientific Visualization

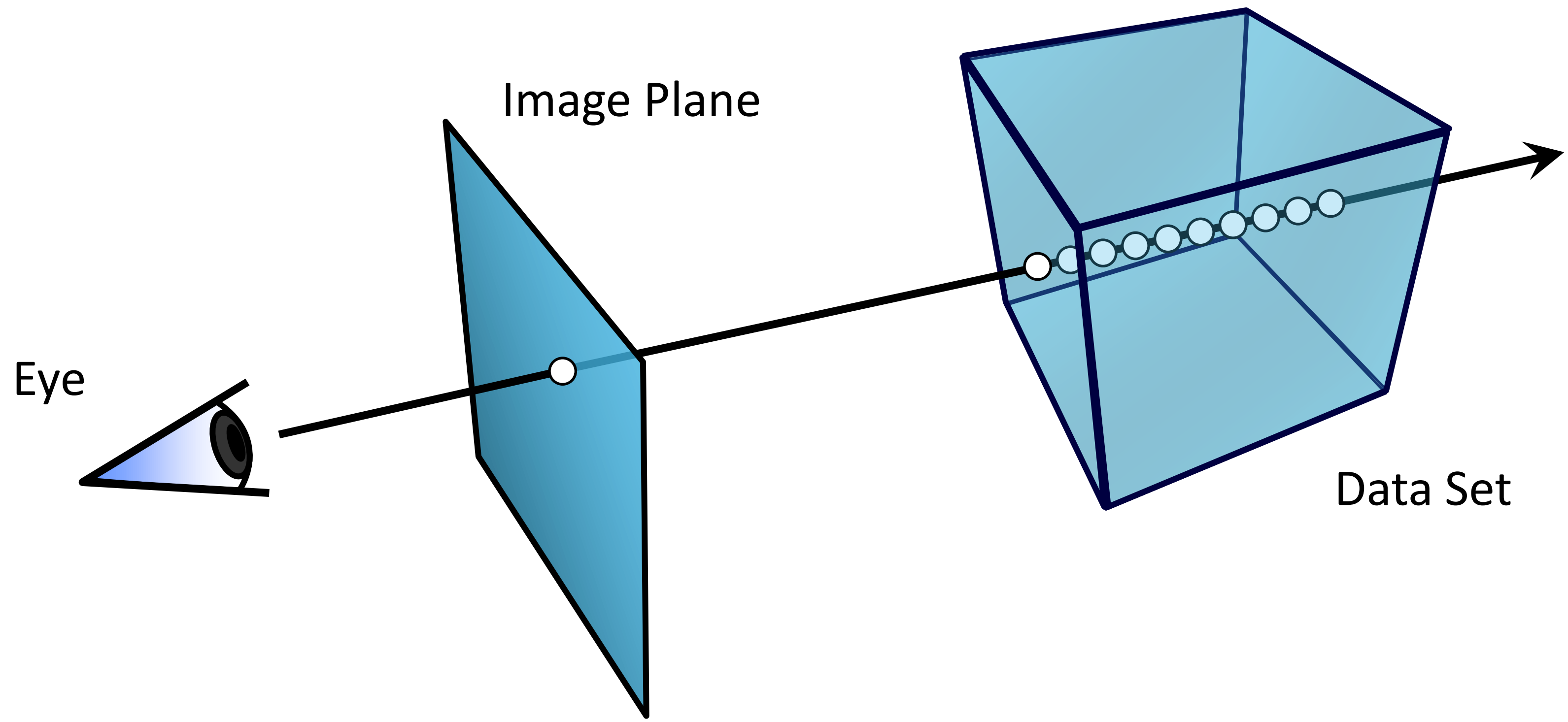
GPU Volume Raycasting

Christof Rezk-Salama

University of Siegen, Germany



Volume Rendering in a Nutshell



Back-to-front iteration

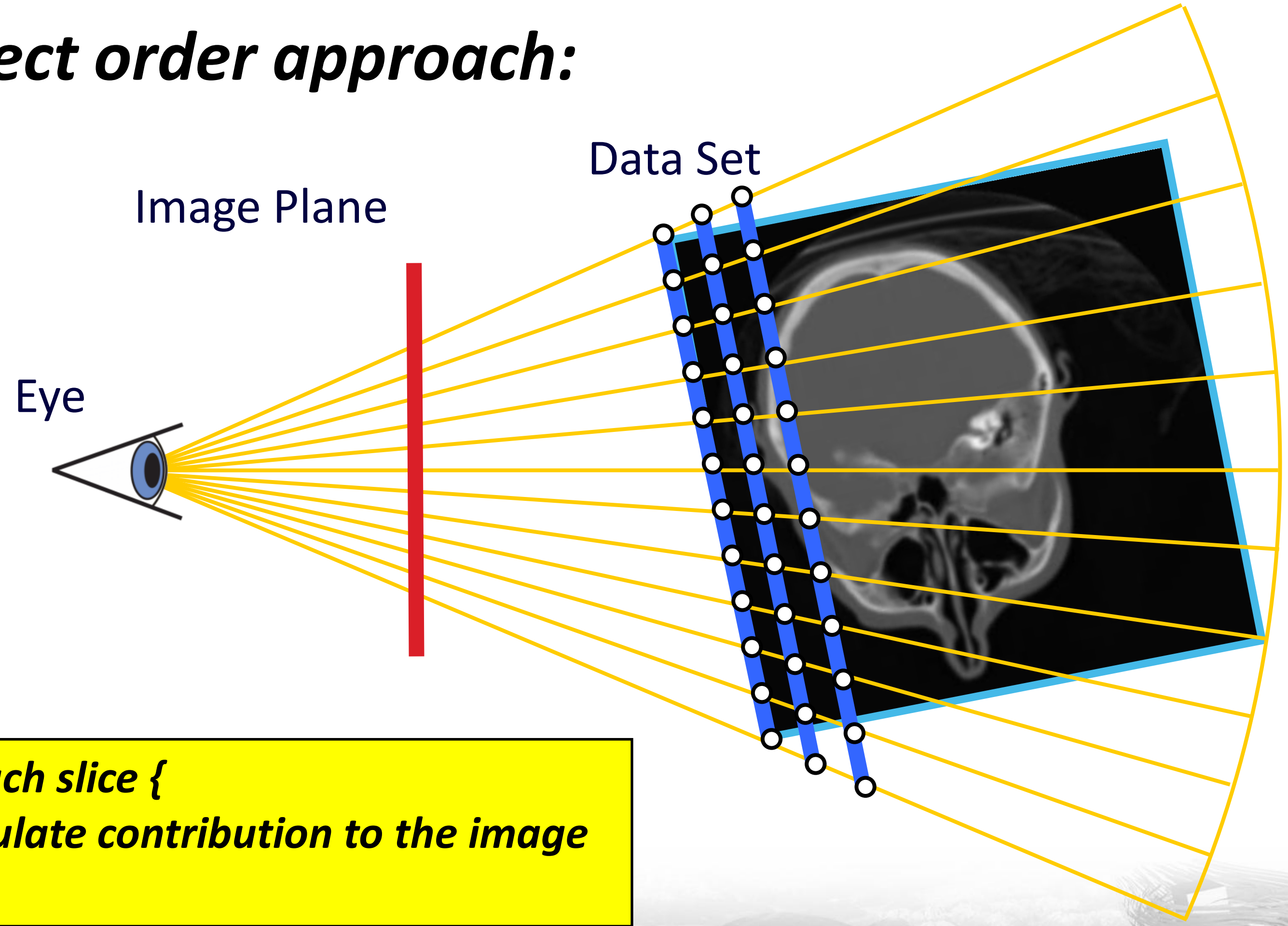
$$C'_i \;=\; C_i + (1 - A_i)C'_{i-1}$$

Front-to-back iteration

$$\begin{aligned} C'_i &= C'_{i+1} + (1 - A'_{i+1})C_i \\ A'_i &= A'_{i+1} + (1 - A'_{i+1})A_i \end{aligned}$$

Volume Rendering

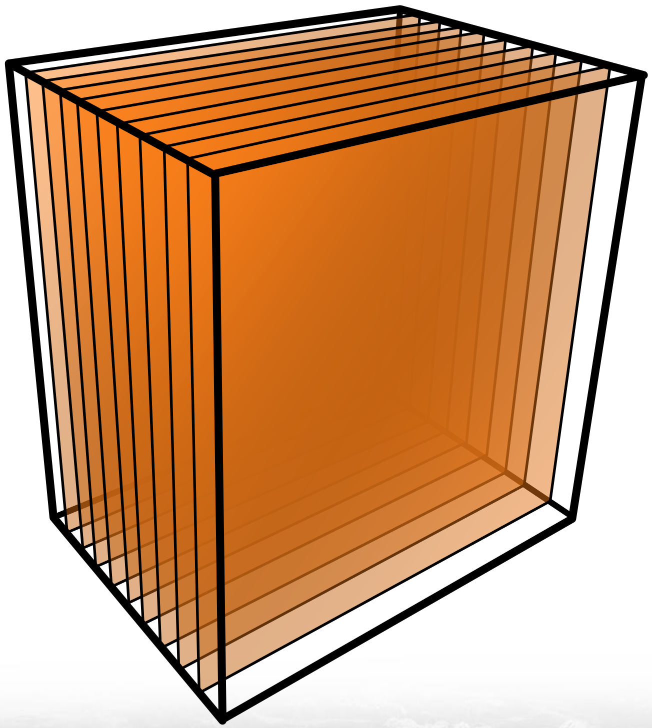
Object order approach:

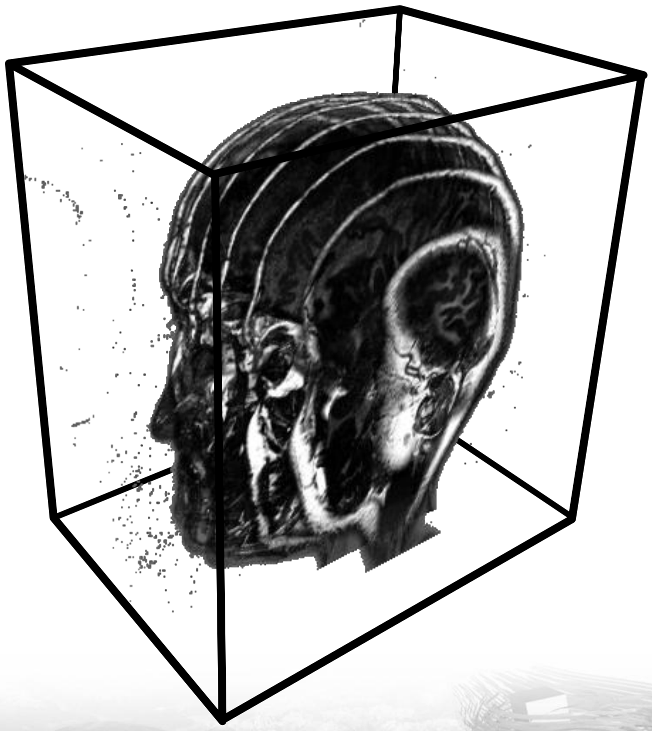



```
For each slice {
  calculate contribution to the image
}
```

Object Order Approach

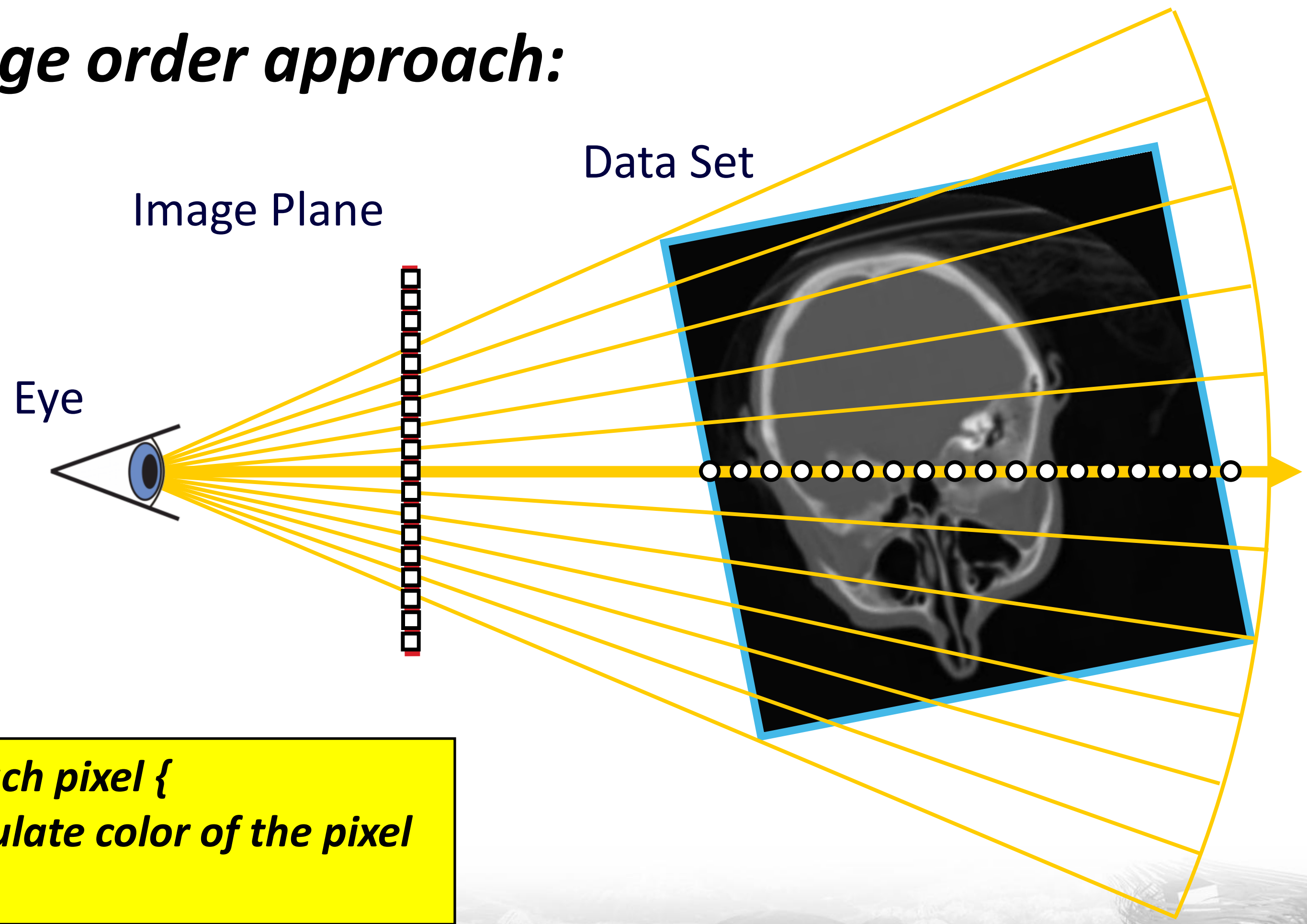
Proxy geometry (Polygonal Slices)





Volume Rendering

Image order approach:



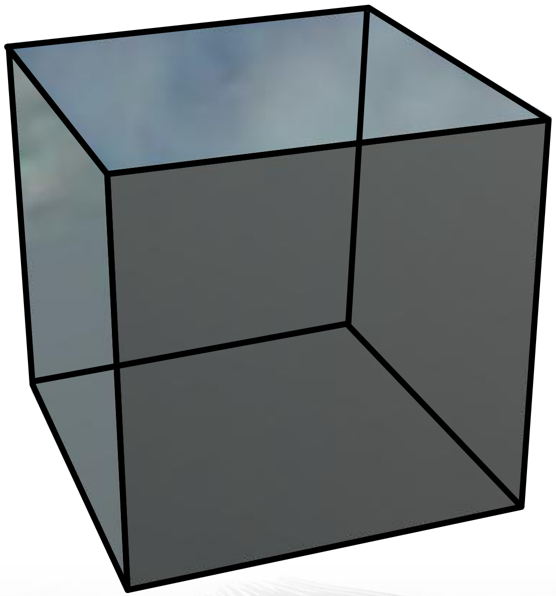
For each pixel {
calculate color of the pixel
}

GPU Volume Raycasting

- Easy to implement
- Everybody knows how to draw a box:

```
glBegin(GL_QUADS);
{
  glVertex3d(0.0,0.0,0.0);
  glVertex3d(0.0,0.0,1.0);
  glVertex3d(0.0,1.0,1.0);
  glVertex3d(0.0,1.0,0.0);

  …
}
glEnd();
```



GPU Volume Raycasting

- Simple Shader for Texture Mapping

```
float3 main(      float3 worldPos   : TEXCOORD0,
                  float3 viewVec    : TEXCOORD1,

                  uniform sampler3D volumeTex
           ) : COLOR
{
    float4 color = float4(0, 0, 0, 0);
    for( int i = 0; i < NUM_STEPS; i++) {
        float4 sample = tex3D(volumeTex, worldPos);
        color += (1 - color.a) * sample;
        if (color.a > 0.99) break;
        worldPos += STEPSIZE * viewVec;
    }

    return color;

}
```

Original Vertex Position in World Space

Viewing Vector in World Space

3D Texture

GPU Raycasting vs. Texture Slicing

GPU Raycasting is easier to impoment

and also more efficient, because:

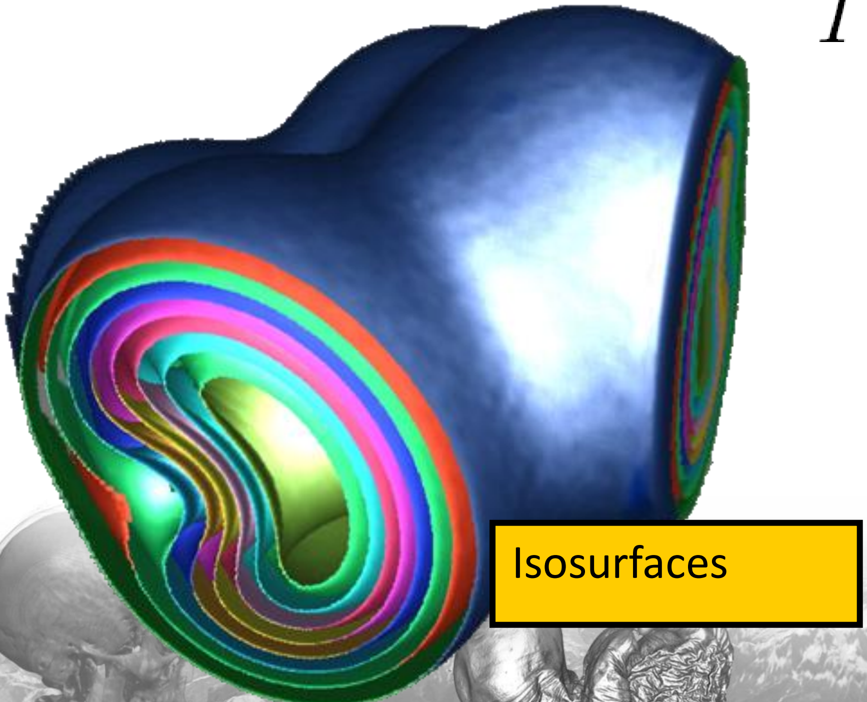
- No Alpha Blending!
  avoid successive read/write operations
  to the frame buffer
- Early Ray Termination

… provided you have an up-to-date GPU

Local Illumination

- Surface lighting: Light is reflected at surfaces
- Volume lighting: Light is scattered at ***isosurfaces***

$$I(\mathbf{p}) = \{\, \mathbf{x} \mid f(\mathbf{x}) = f(\mathbf{p}) \,\}$$



- Isosurface extraction not required!
- We only need the normal vector
- Normal vector of isosurface is equal to (normalized) gradient vector

Gradient Estimation

- The gradient vector is the first-order derivative of the scalar field

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \dfrac{\partial f(\mathbf{x})}{\partial x} \\ \dfrac{\partial f(\mathbf{x})}{\partial y} \\ \dfrac{\partial f(\mathbf{x})}{\partial z} \end{pmatrix}$$

partial derivative ***in x-direction***

partial derivative ***in y-direction***

partial derivative ***in z-direction***

- We can estimate the gradient vector using finite differencing schemes

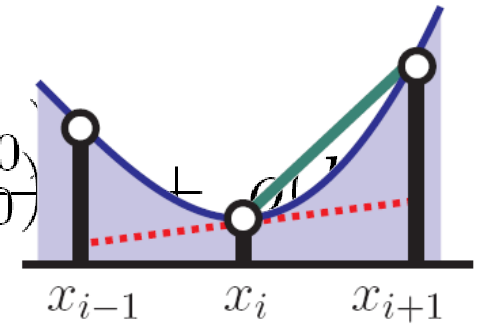
Finite Differences

- Taylor expansion:

Forward Difference:

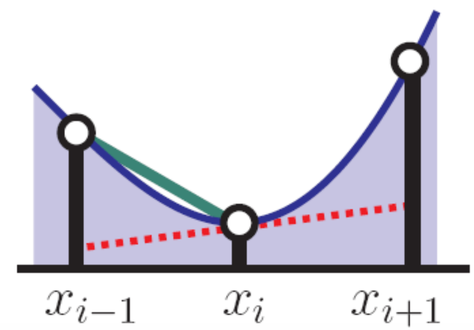
$$f(x_0 + h) = f(x_0) + \frac{f'(x_0)}{1!} h + O(h^2)$$

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h}$$



Backward Difference:

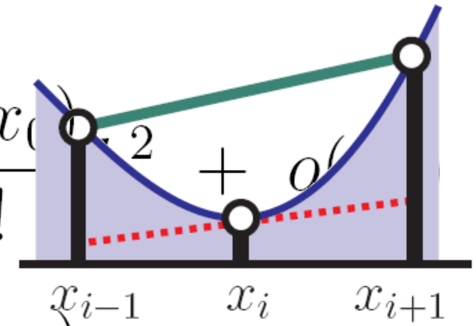
$$f'(x_0) = \frac{f(x_0) - f(x_0 - h)}{h}$$



Finite Differences

Central Difference:

$$f(x_0 + h) = f(x_0) + \frac{f'(x_0)}{1!}h + \frac{f''(x_0)}{2!}h^2 + o(h^3)$$



$$f(x_0 - h) = f(x_0) - \frac{f'(x_0)}{1!}h + \frac{f''(x_0)}{2!}h^2 + o(h^3)$$

Gradient Approximation using Central Differences:

$$f(x_0 + h) - f(x_0 - h) = 2f'(x_0)h$$

$$\frac{f(x_0 + h) - f(x_0 - h)}{2h}$$

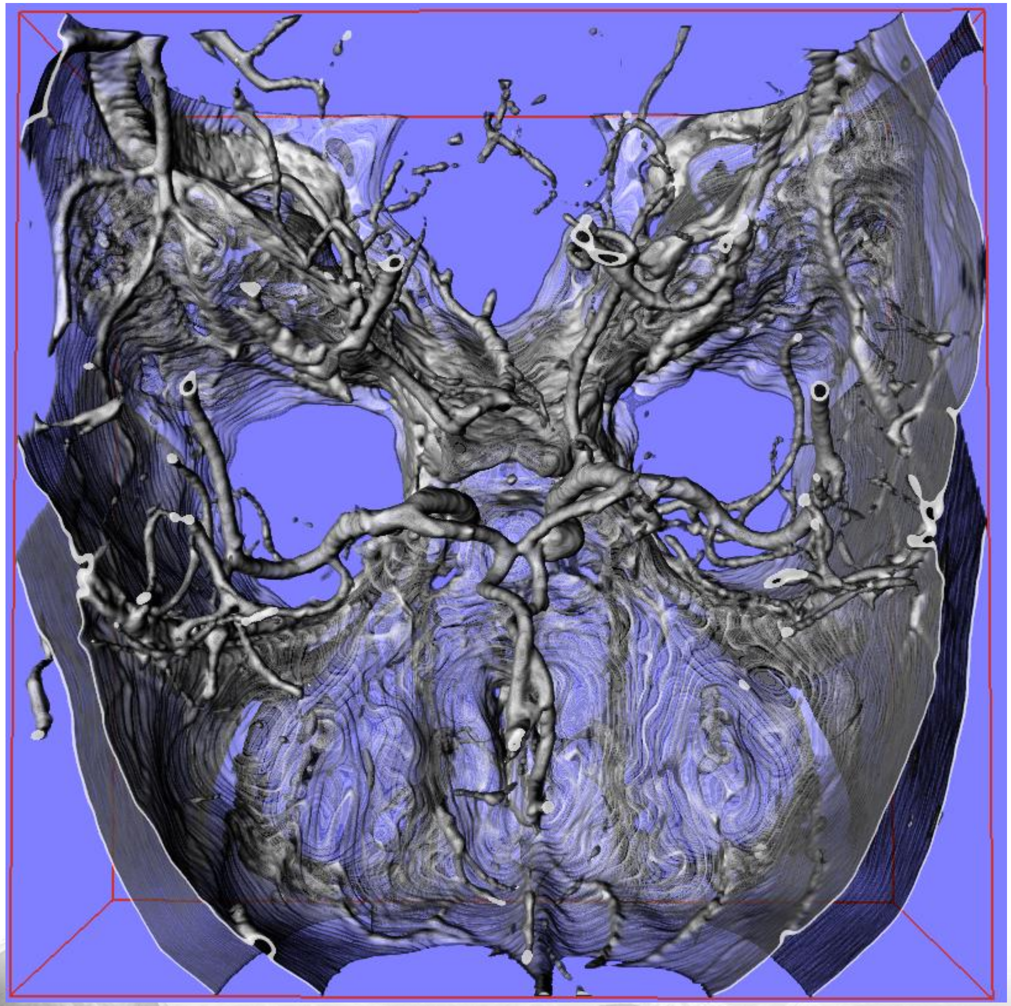
$$\nabla f(x,y,z) \approx \frac{1}{2h}\begin{pmatrix} f(x+h,y,z) - f(x-h,y,z) \\ f(x,y+h,z) - f(x,y-h,z) \\ f(x,y,z+h) - f(x,y,z-h) \end{pmatrix}$$

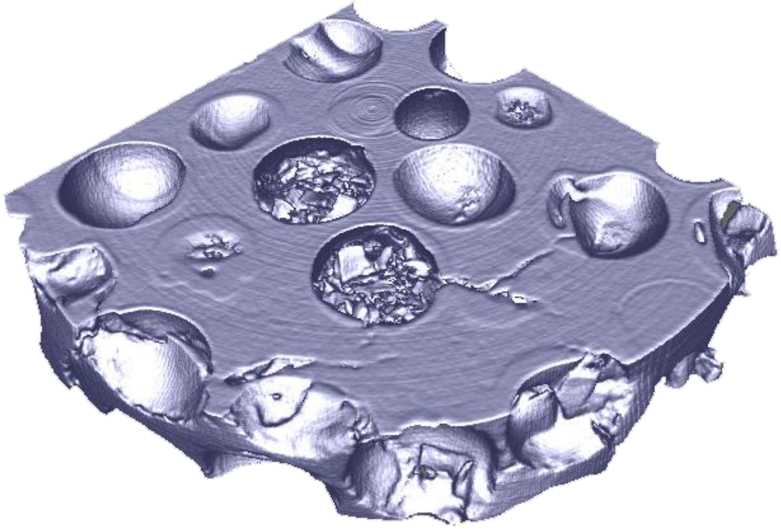
On-the-fly Gradient Estimation

$$\nabla f(x,y,z) \approx \frac{1}{2h} \begin{pmatrix} f(x+h,y,z) - f(x-h,y,z) \\ f(x,y+h,z) - f(x,y-h,z) \\ f(x,y,z+h) - f(x,y,z-h) \end{pmatrix}$$

```
float3 sample1, sample2;
// six texture samples for the gradient
sample1.x = tex3D(texture,uvw-half3(DELTA,0.0,0.0)).x;
sample2.x = tex3D(texture,uvw+half3(DELTA,0.0,0.0)).x;
sample1.y = tex3D(texture,uvw-half3(0.0,DELTA,0.0)).x;
sample2.y = tex3D(texture,uvw+half3(0.0,DELTA,0.0)).x;
sample1.z = tex3D(texture,uvw-half3(0.0,0.0,DELTA)).x;
sample2.z = tex3D(texture,uvw+half3(0.0,0.0,DELTA)).x;
// central difference and normalization
float3 N = normalize(sample2-sample1);
```

Examples Local Illumination





On-the-fly Gradient Estimation

Drawbacks:

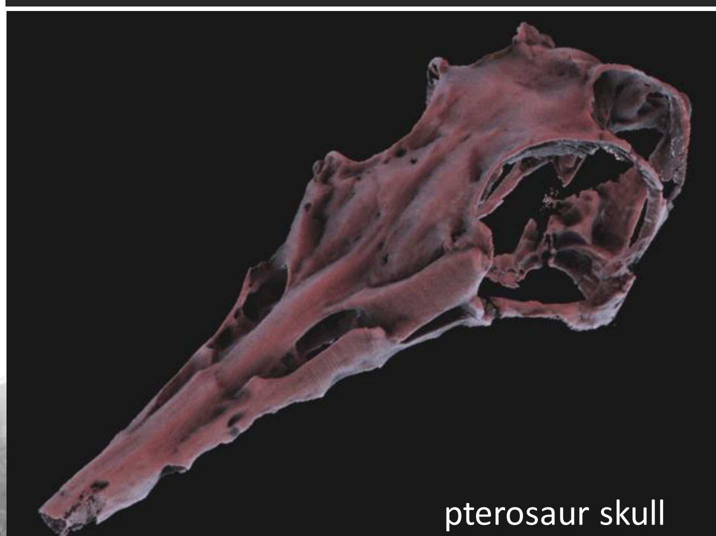
- Each additional texture sample is expensive!
  - ***Central Differences:*** 7 Texture Samples
  - ***Forward/Backward Differences:*** 4 Texture Samples

Advantages:

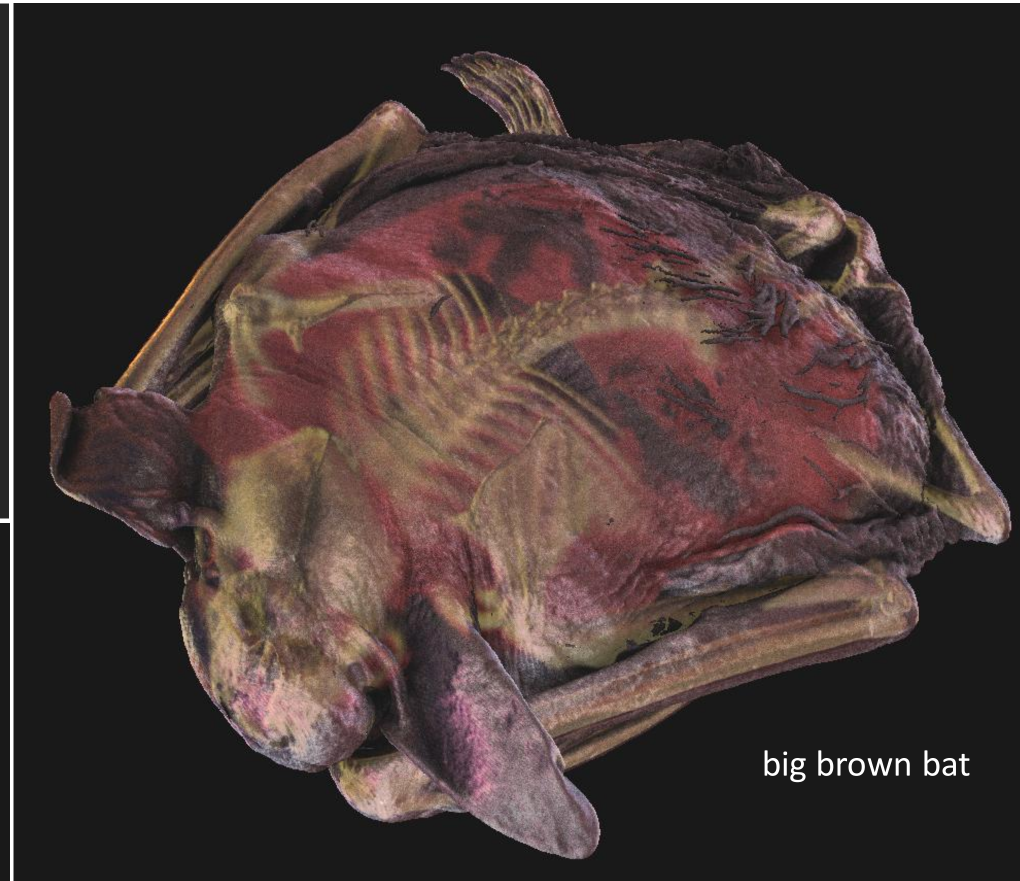
- Low memory requirements
- Gradient estimation at floating point precision!
- Gradient estimation can be omitted in FP (using a conditional branch)

Advanced Illumination

cheetah skull



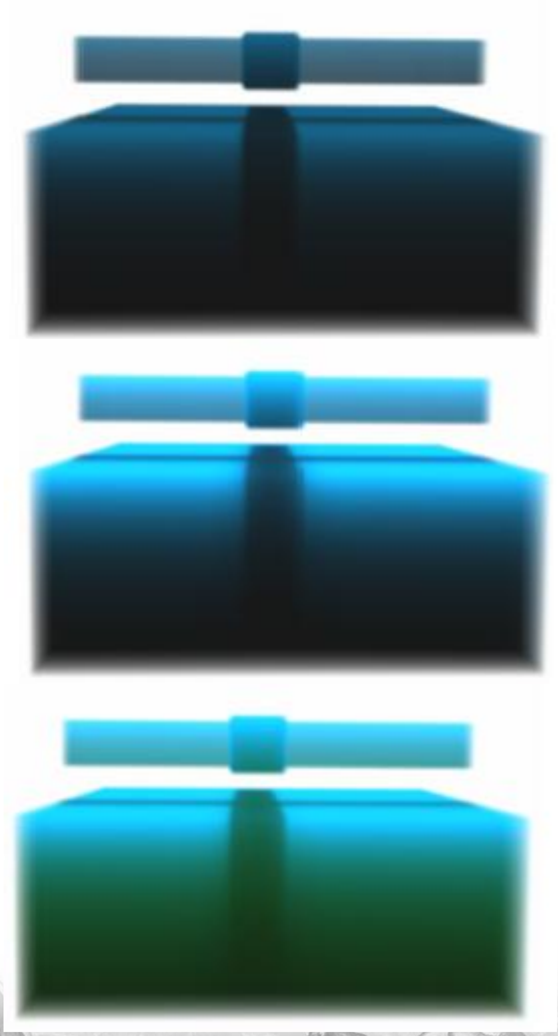
pterosaur skull

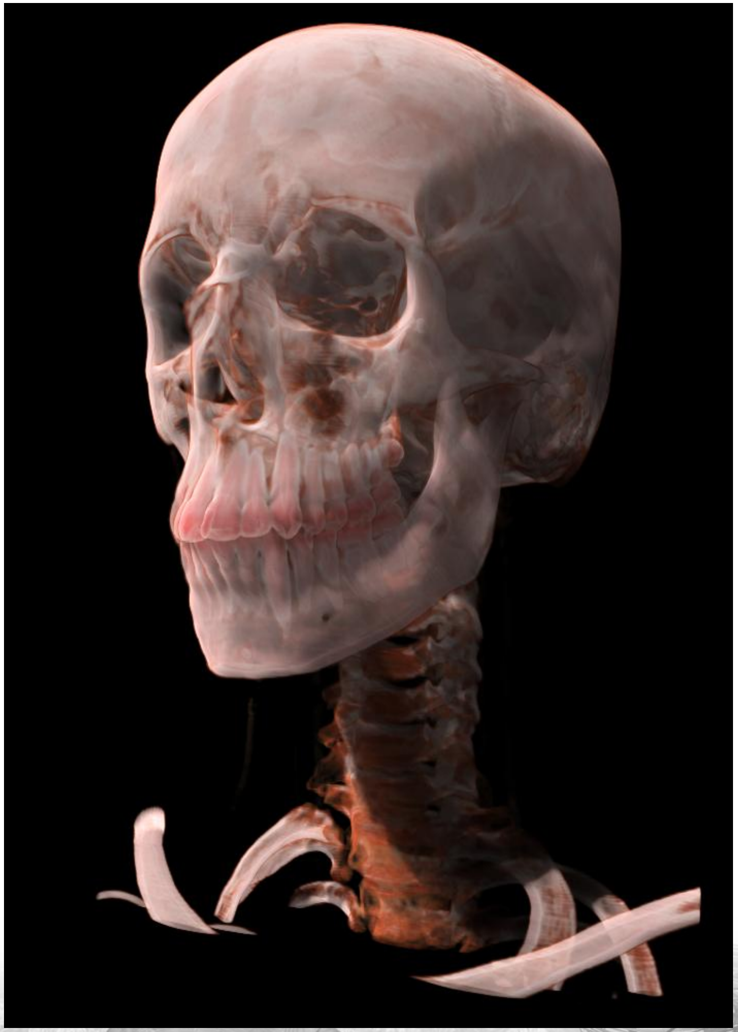


big brown bat

Data sets available at the
UTCT data archive, DIGIMORPH
http://utct.tacc.utexas.edu

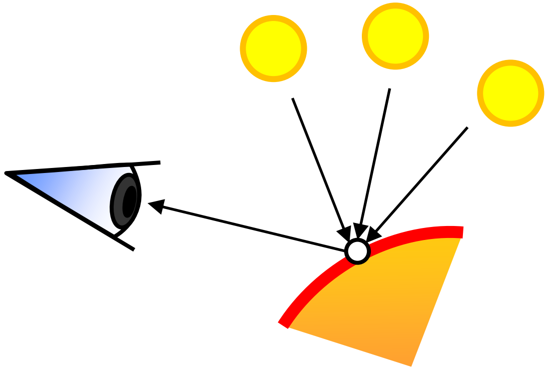
Translucency



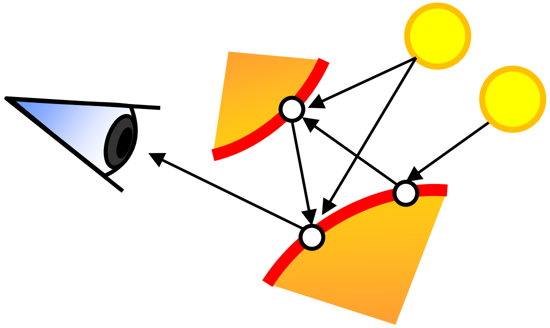


Scattering Effects

When a photon hits a surface, it changes both direction and energy



- ***Single Scattering:***
  - Light is scattered ***once*** before it reaches the eye
  - Local illumination model



- ***Multiple Scattering***
  - Soft shadows
  - Translucency
  - Color bleeding

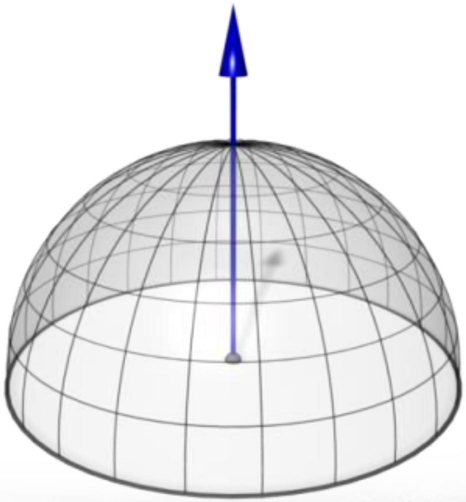
Surface-Like Reflection: The BRDF

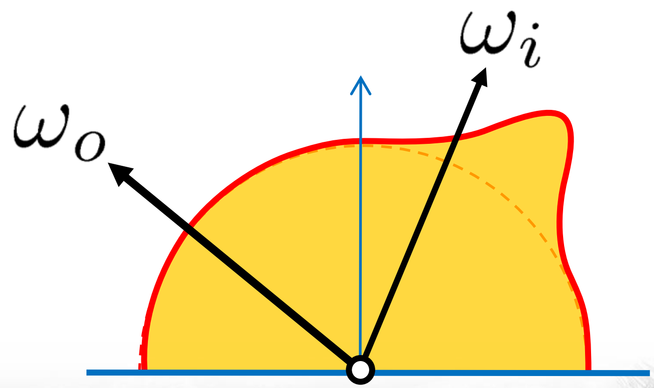
$$L(\mathbf{x},\omega_o) = \int_{\Omega+} \boxed{f_r(\mathbf{x},\omega_o \to \omega_i)}\, L(\mathbf{x},\omega_i)\cos\theta_i \, d\omega_i$$

Hemisphere

BRDF

Elevation Angle





Volume Scattering: The Phase Function

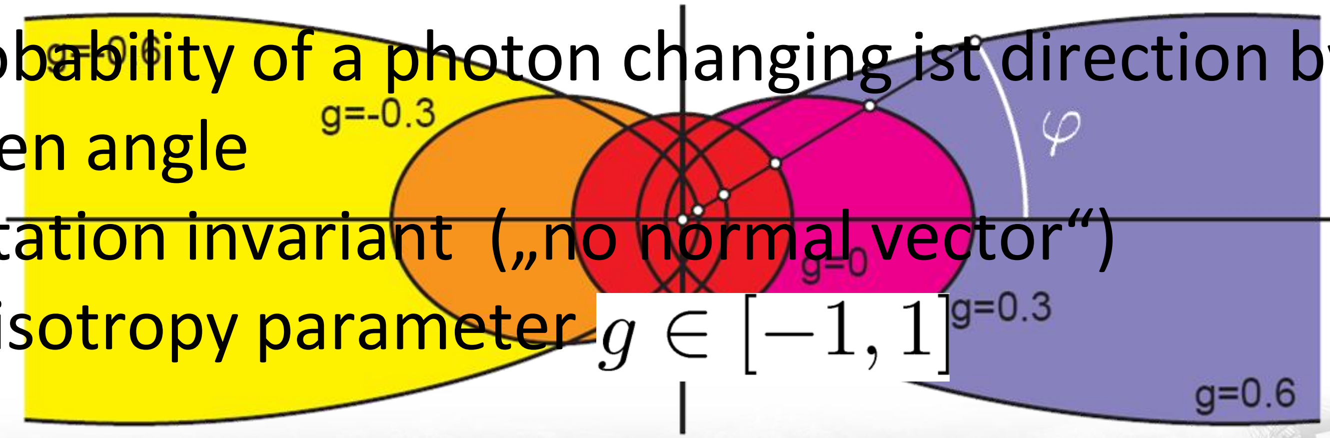
$$L(\mathbf{x},\omega_o) = \int_\Omega \boxed{h(\mathbf{x},\omega_o \rightarrow \omega_i)} L(\mathbf{x},\omega_i)\, d\omega_i$$

Sphere

Phase Function

- Example: Henyey Greenstein

$$h_{\mathrm{HG}}(\mathbf{x},\omega_i \rightarrow \omega_o) = \frac{1-g^2}{4\pi(1+g^2-2g(\omega_i\cdot\omega_o))^{\frac{3}{2}}}$$



- Probability of a photon changing ist direction by a given angle
- Rotation invariant („no normal vector“)
- Anisotropy parameter $g \in [-1,1]$

The BSDF

$$L(\mathbf{x}, \omega_o) = \int_\Omega \boxed{f(\mathbf{x}, \omega_o \to \omega_i)} L(\mathbf{x}, \omega_i) |\cos\theta_i| \, d\omega_i$$

Sphere

BSDF

Elevation Angle

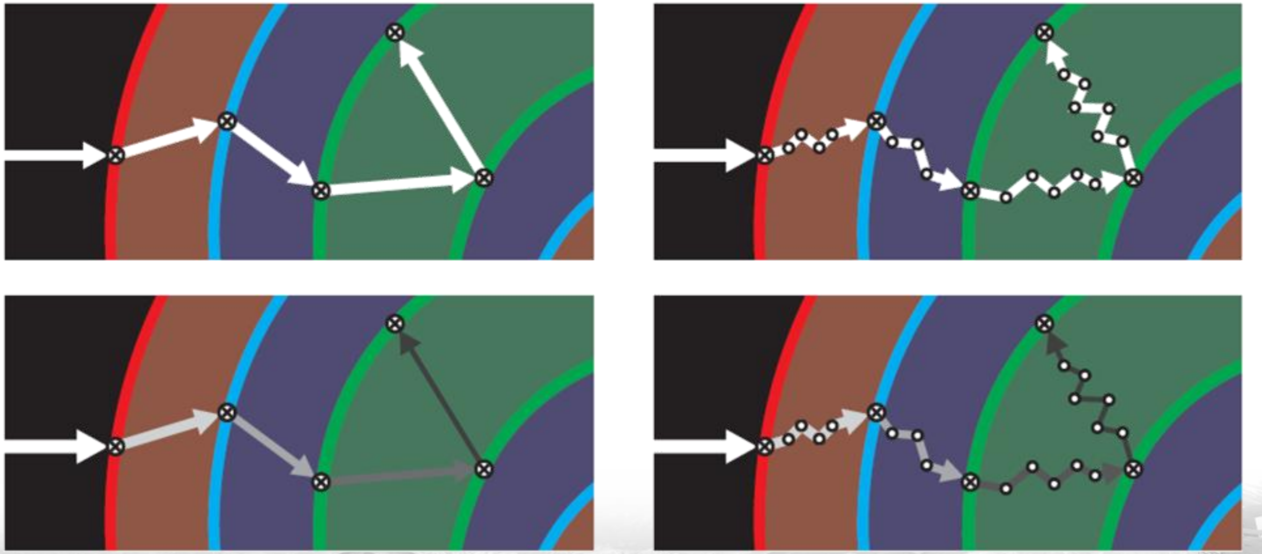
- Scattering with orientation (surface-like)
- Phase function notation:

$$h_{\mathrm{BSDF}}(\mathbf{x}, \omega_i \to \omega_o) = f(\mathbf{x}, \omega_i \to \omega_o) \, |\cos\theta_i|$$

A Practical Model

- Surface-like scattering (BSDF) at high gradients

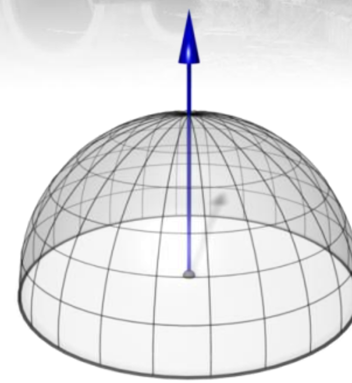
$$h(\mathbf{x},\omega_i,\omega_o) = \begin{cases} h_{\mathrm{BSDF}}(\mathbf{x},\omega_i,\omega_o) \quad \text{with } \mathbf{n} = \frac{\nabla s(\mathbf{x})}{|\nabla s(\mathbf{x})|}, & \text{if } |\nabla s(\mathbf{x})| > \psi \\ h_{\mathrm{HG}}(\mathbf{x},\omega_i,\omega_o), & \text{otherwise} \end{cases}$$



A Practical Model

Math Notation

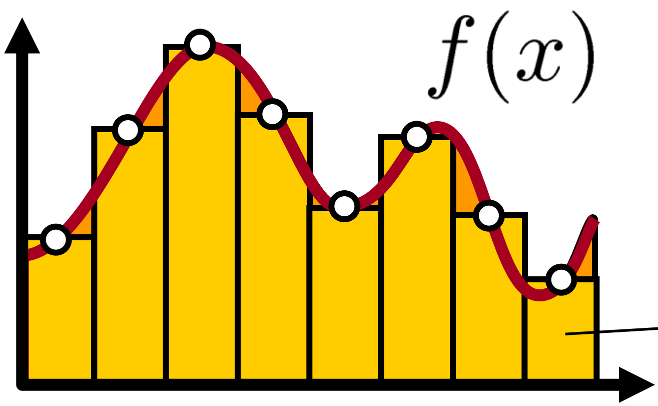
$$L(\mathbf{x}, \omega_o) = \int_{\Omega} h(\mathbf{x}, \omega_o \rightarrow \omega_i) \, L(\mathbf{x}, \omega_i) \, d\omega_i$$



integrates over the entire sphere/hemisphere

- Integral must be solved for every intersection point
- ***Fredholm Equation*** (cannot be solved analytically)

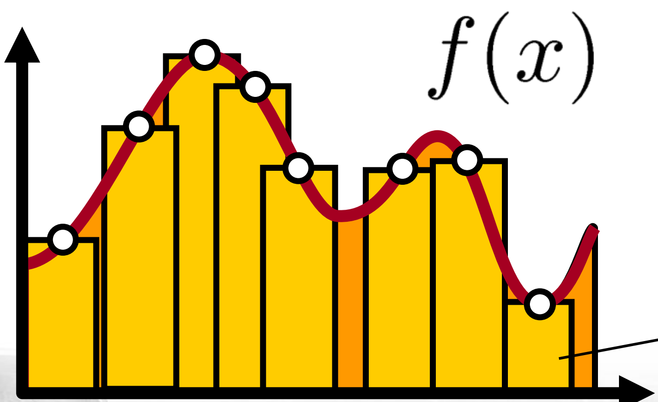
Numerical Integration



Equidistant Sampling

- Approximation integral by a Riemann sum

$$\int_a^b f(x)dx \approx \sum_{i=0}^{N} f(x_i)\frac{b-a}{N}$$

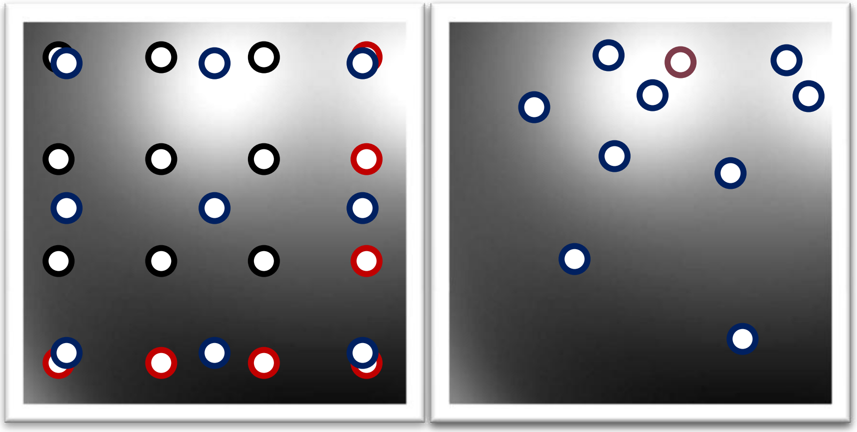


Stochastic Sampling

- Uniformly distributed samples
- Approximation by sum

$$\int_a^b f(x)dx \approx \sum_{i=0}^{N} f(x_i)\frac{b-a}{N}$$

Stochastic Sampling



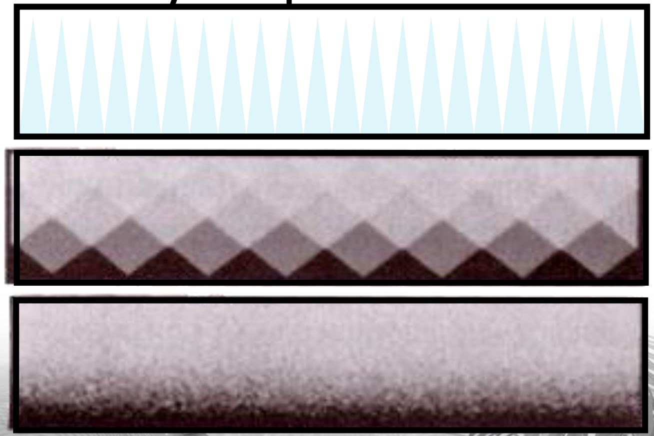
Cons:

– Slower convergence than Riemann sum

Pros:

– ***Better Scalability for multidimensional functions:*** increase number of samples in arbitrary steps

– ***Noise*** instead of Aliasing



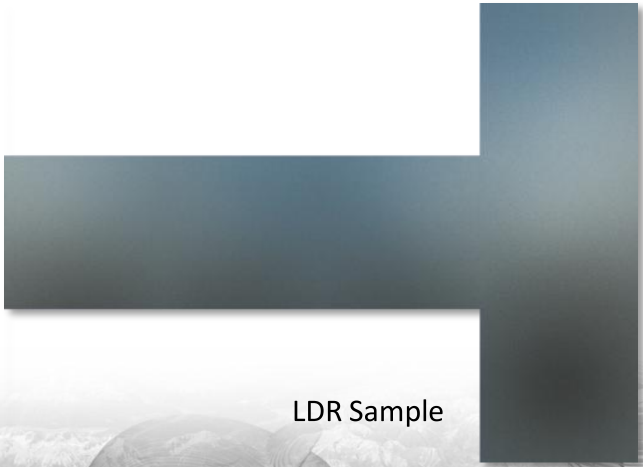
– ***Independent of sampling grid:*** Clever placement of samples will improve the convergence!

Blind Monte-Carlo Sampling

- **Example: Filtering an Environment Map**

Given an Environment Map
(i.e. photograph: fisheye or mirror ball)

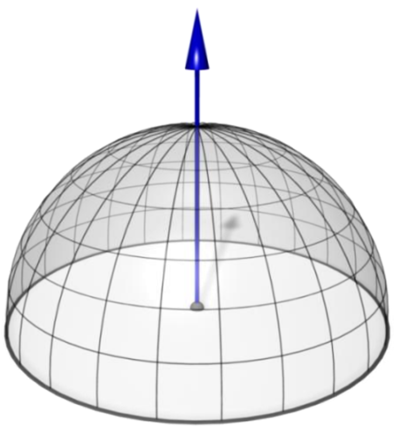
Calculate an Irradiance Map
For each pixel of the irradiance map:
- Determine n random directions on the hemisphere
- Sample the Environment Map and
- Average the results (incl. cos-term)



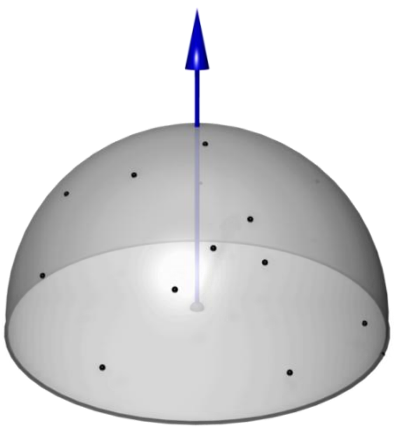
LDR Sample

Rendering

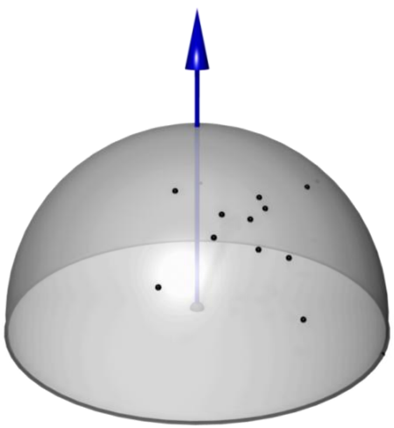
- Calculate the radiance of a point
  - depending on the incoming light on the sphere/hemisphere
  - depending on the phase function/BRDF



Deterministic
Uniform sampling of the sphere/hemisphere.
High computational load
good approximation

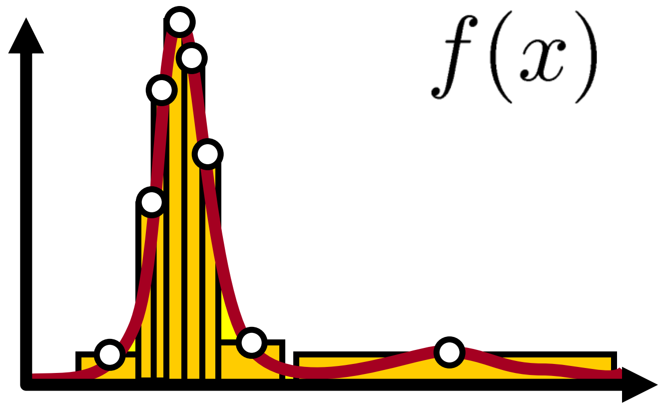


Blind Monte-Carlo
Randomized sampling of the sphere/hemisphere.
Visually better images for fewer samples, slow convergence



Importance Sampling
Place samples where contribution is high
Faster!

Importance Sampling



Stochastic Sampling

- Non-uniformly distributed samples
- Approximation by sum

$$\int_a^b f(x)dx \approx \frac{1}{N}\sum_{i=0}^{N}\frac{f(x_i)}{p(x_i)}$$

Probability Distribution Function (PDF)

Clever placement of samples

- Many samples where function is high
- Few samples where function is low

Sampling a Specular Lobe

- Simple Approach

$$f(\varphi) = \cos^s(\varphi) = (\mathbf{r} \cdot \mathbf{v})^s$$

Specular term

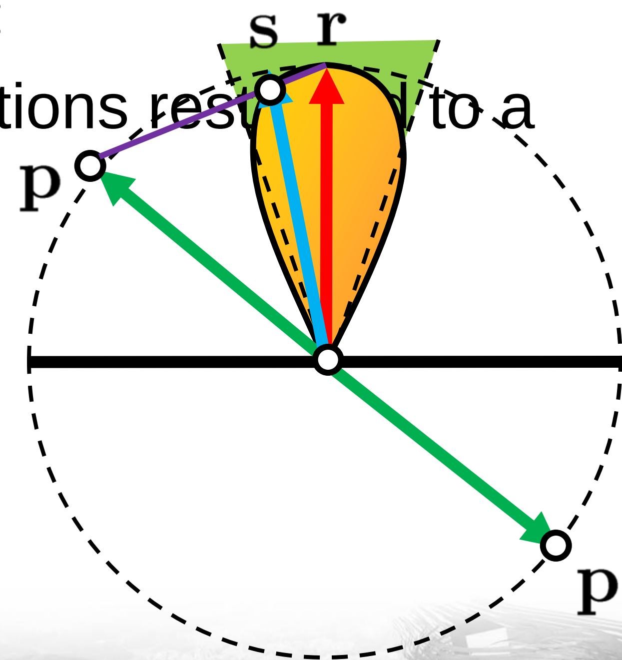
Non-optimal, but easy to implement

Idea: uniform distribution of directions restricted to a cone

- Precompute random unit vectors with uniform PDF
- Randomly pick one vector $\mathbf{p}$
- Negate vector, if $(\mathbf{r} \cdot \mathbf{p}) < 0$
- Blend with vector $\mathbf{r}$ and normalize

$$\mathbf{s} = \alpha\,\mathbf{r} + (1 - \alpha)\mathbf{p}$$

- Blend weight α controls the size of the specular highlight and can be calculated from shininess s



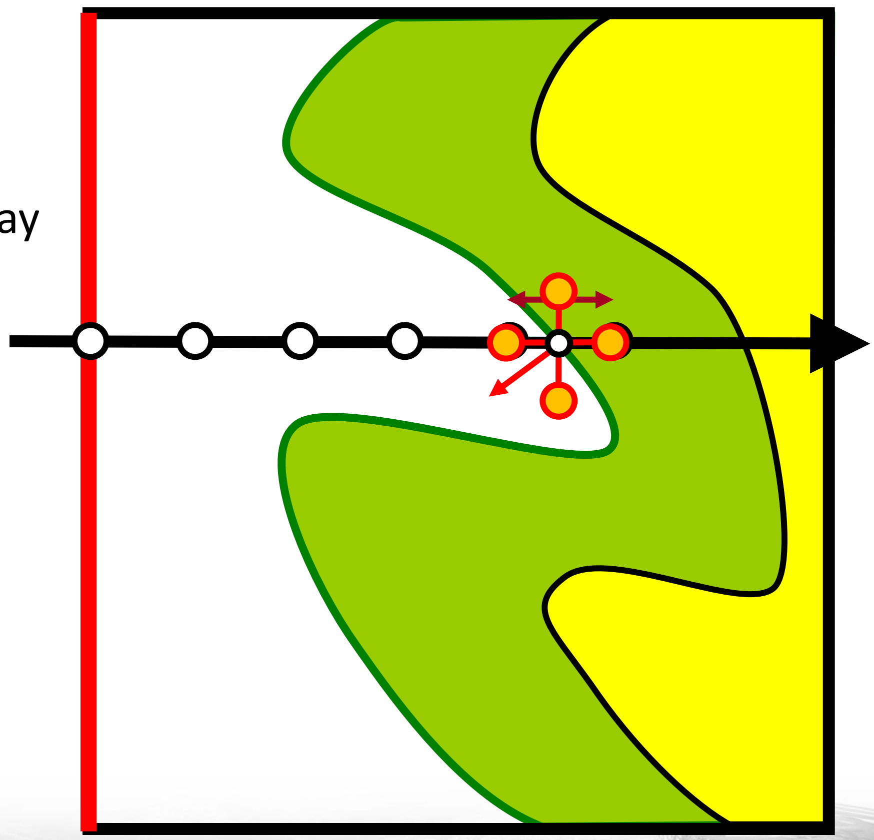
Importance Sampling

Literature:

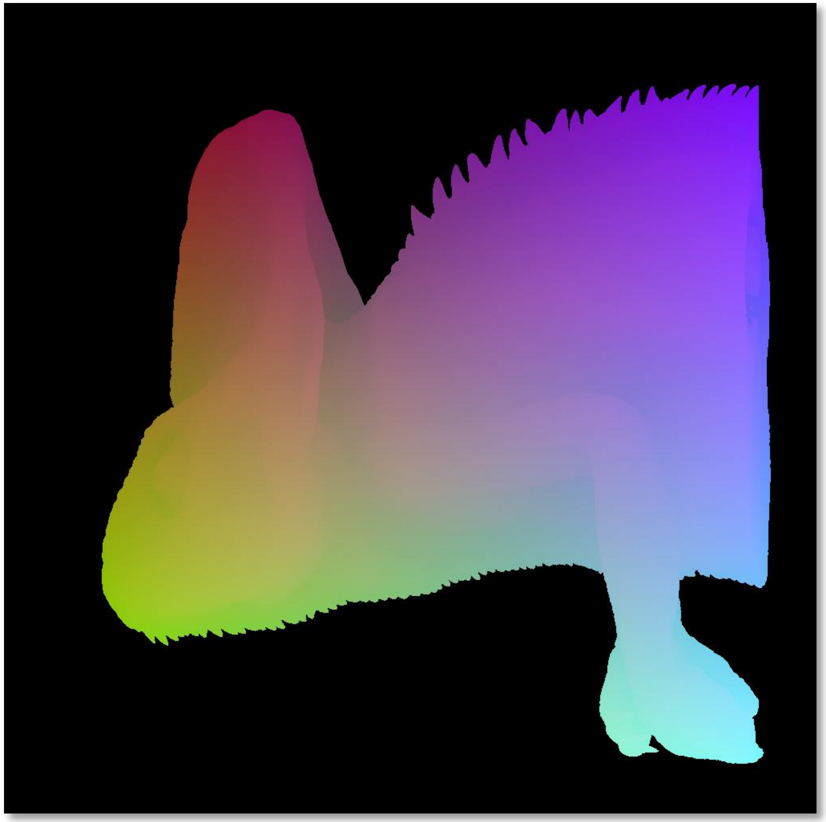
- M. Pharr, G. Humphries: **Physically Based Rendering**, Morgan Kauffman (Elsevier), 2004
- M. Colbert, J. Křivánek, **GPU-Based Importance Sampling** *in H.Nguyen (edt.):* **GPU Gems 3**, *Addison-Wesley, 2008*

GPU Ray-Casting

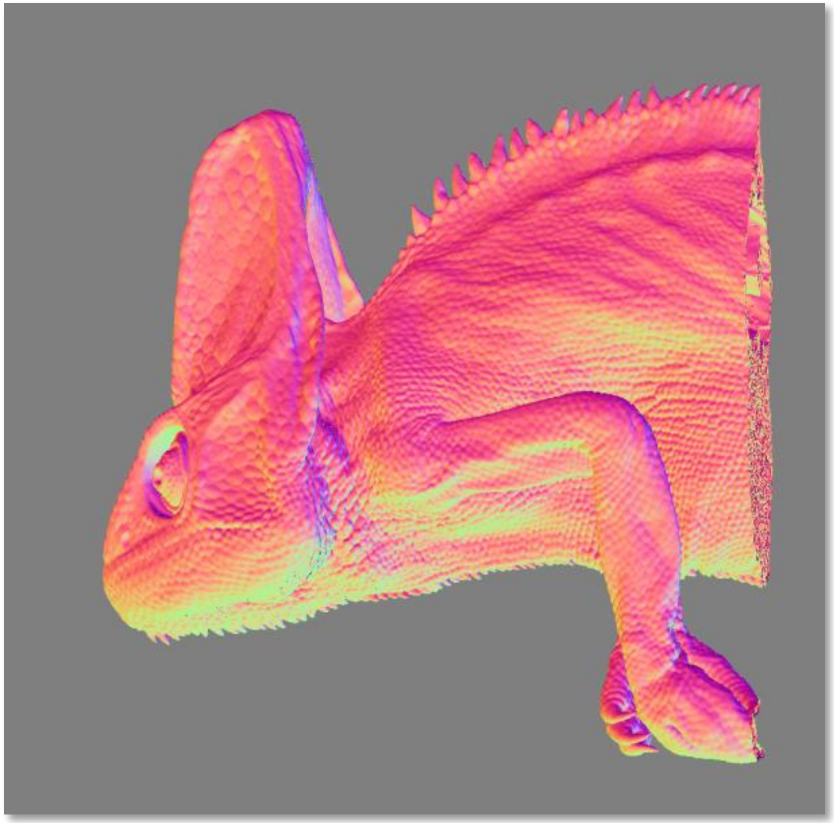
- Calculate First Intersection with Isosurface
  - Rasterize the front faces of the bounding box
  - For each fragment, cast a ray
  - Find first intersection point with isosurface by sampling along the ray
    - interval bisection
  - Store the intersection point in render target 0
  - Estimate the gradient vector using central differences
  - Store the gradient vector in render target 1



First Render Pass



MRT0: xyz-coordinates of first intersection point with isosurface



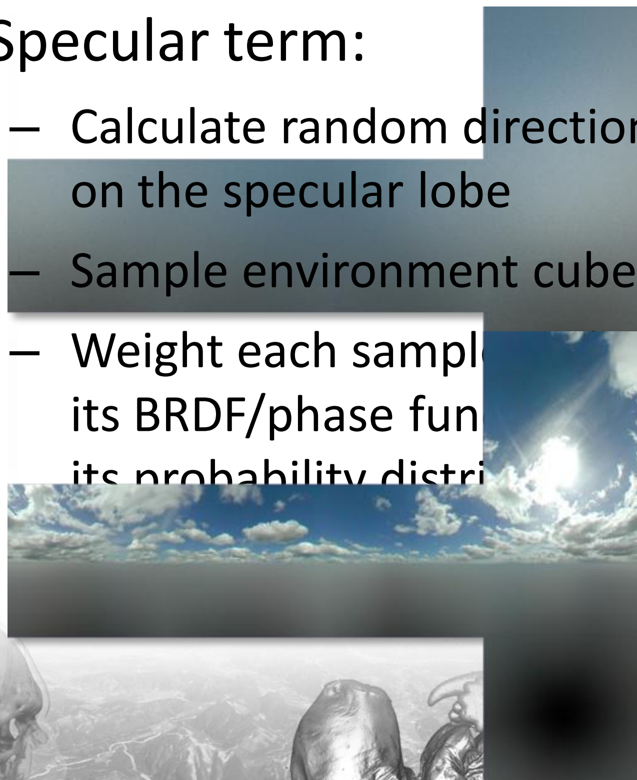
MRT1: xyz-components of gradient vector (color coded)

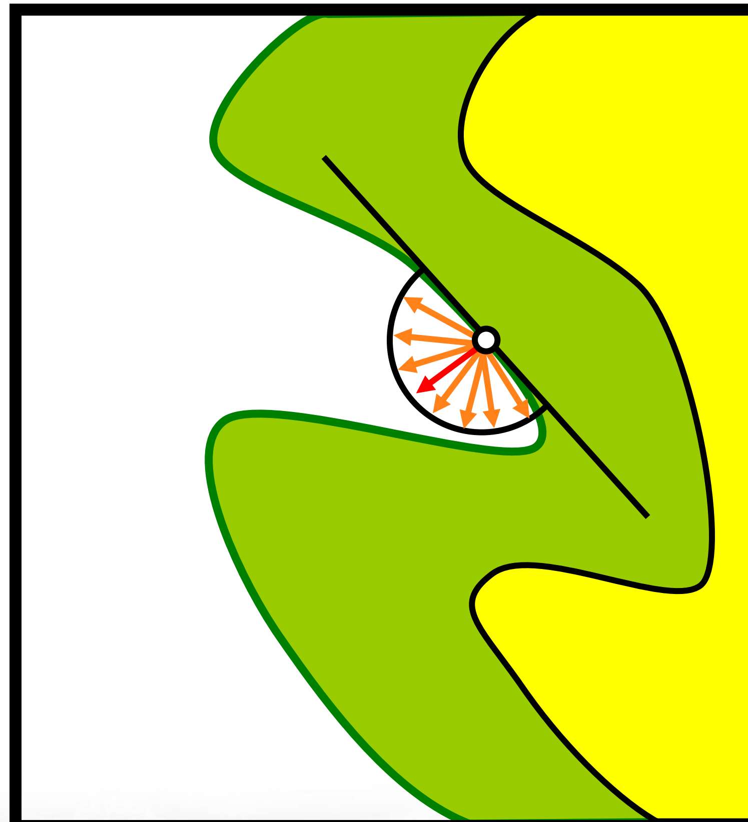
High Quality Isosurface

Deferred Shading

Single Scattering (no shadows)

- Diffuse term:
  - Sample irradiance cube using gradient direction
- Specular term:
  - Calculate random directions on the specular lobe
  - Sample environment cube
  - Weight each sampl
    its BRDF/phase fun nd
    its probability distri





High Quality Isosurface

Why not use a pre-filtered environment map for the specular term as well?

You can, but

- it only works for ***one*** specular exponent per object
- Variable shininess may be used to ***visualize additional surface properties*** (e.g. gradient magnitude)

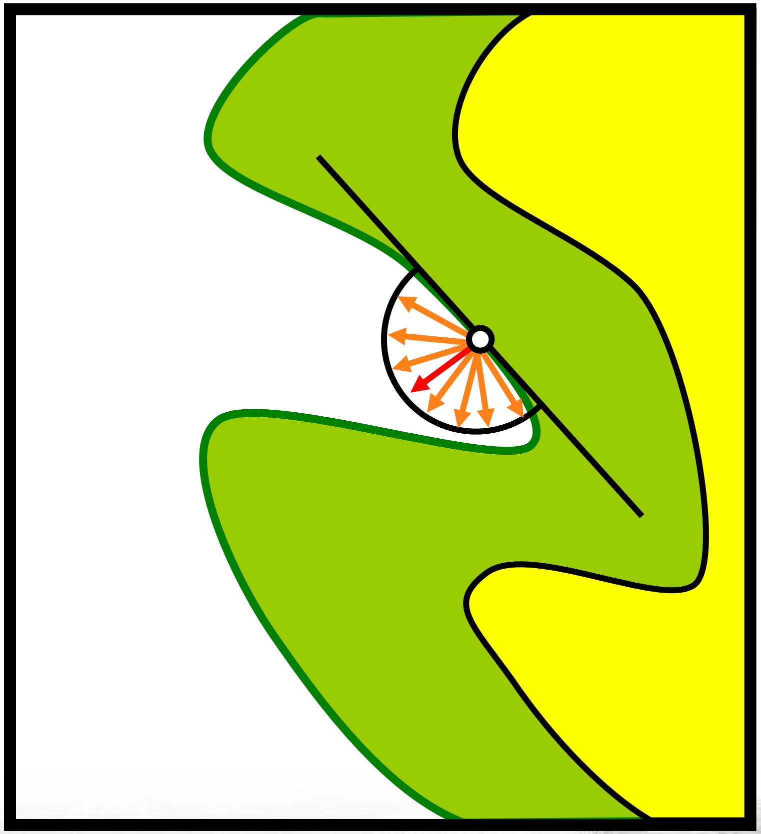
Ambient Occlusion

Approximate indirect light

Accessibility term:

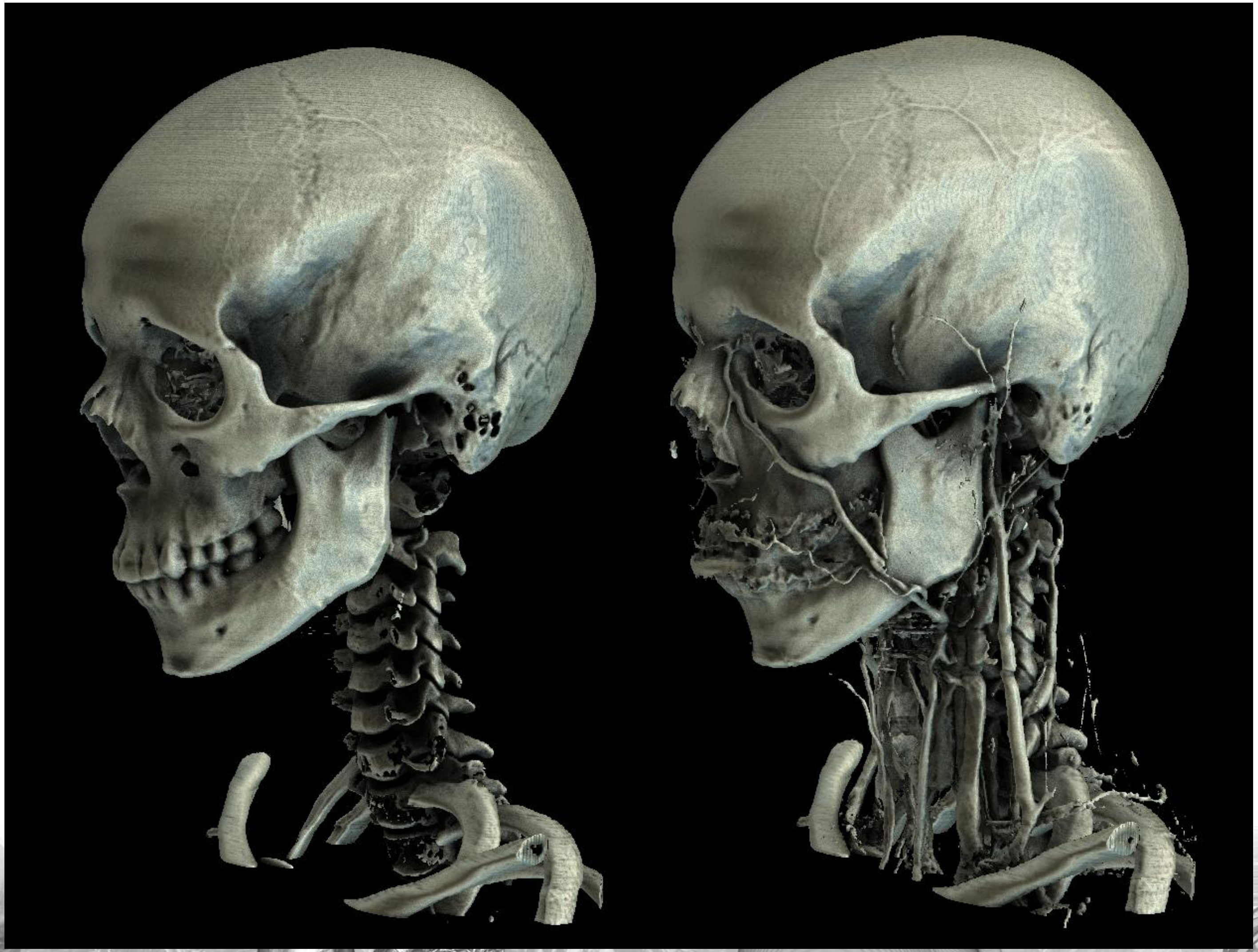
- How easy can a surface point be reached?

- Ambient Occlusion:
  - Calculate random directions on the specular lobe
  - Cast a ray and sample only a few steps in each direction
  - Count the rays that do not intersect the isosurface again

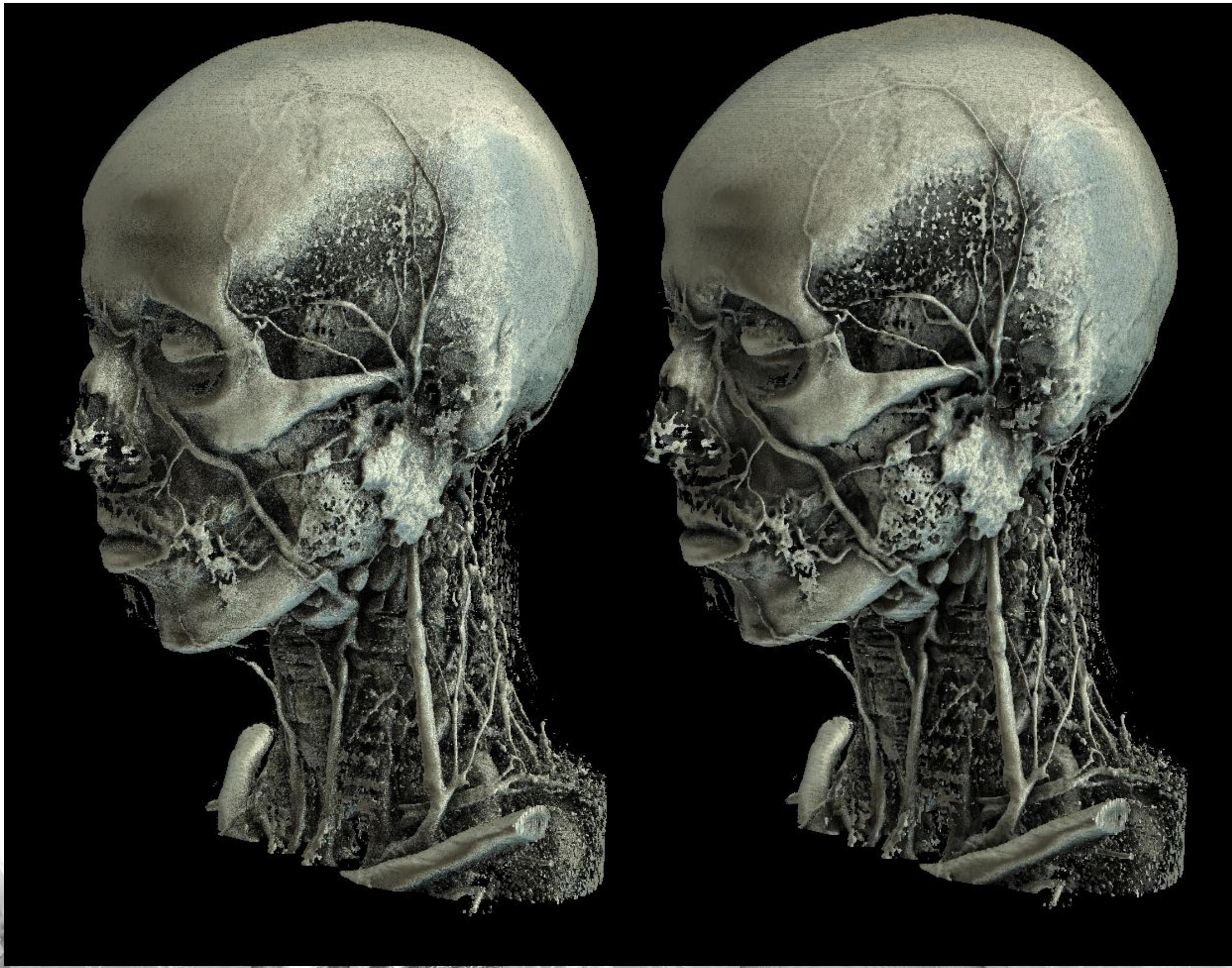


High Quality Isosurface

Single Scattering Example

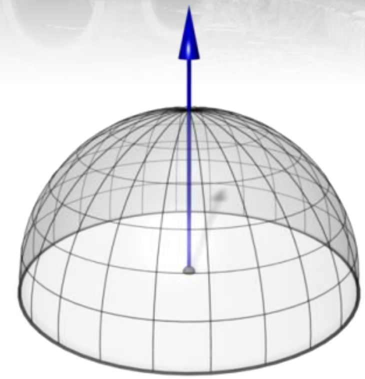


Single Scattering Example



Multiple Scattering

$$L(\mathbf{x}, \omega_o) = \int_{\Omega} p(\mathbf{x}, \omega_o \rightarrow \omega_i)\, L(\mathbf{x}, \omega_i)\, d\omega_i$$

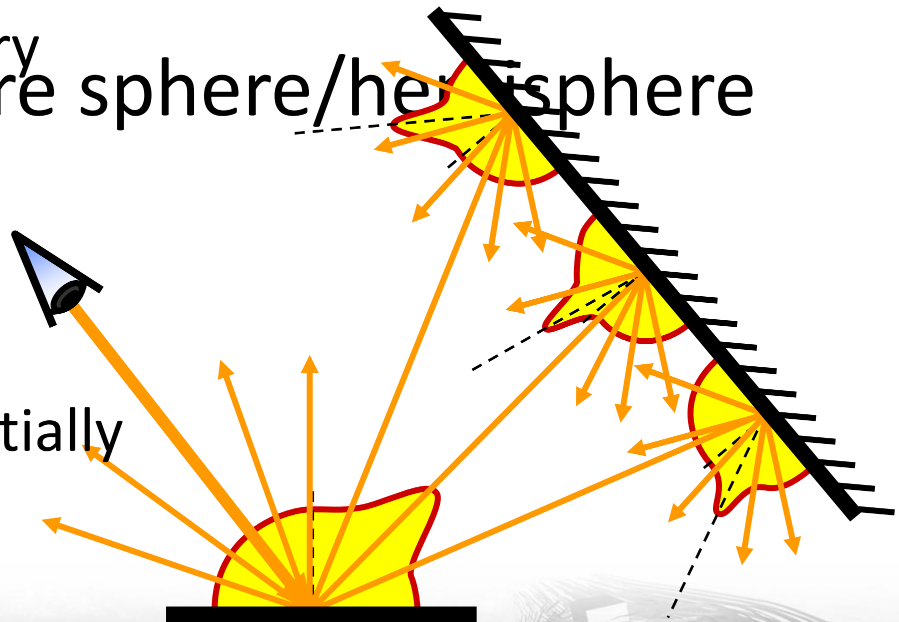


integrates over the entire sphere/hemisphere

- Integral must be solved for every intersection point
- ***Fredholm Equation*** (cannot be solved analytically)

Numerical Solution:

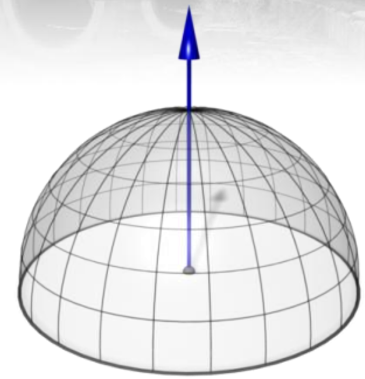
- Number of rays grows exponentially
- Much workload spent for little contribution



Multiple Scattering

Mathematical Model

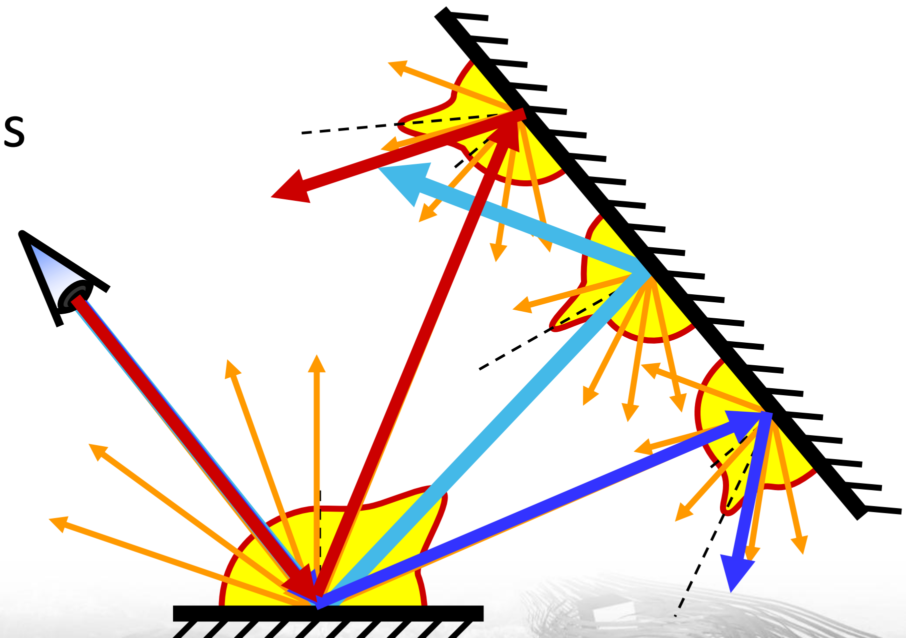
$$L(\mathbf{x}, \omega_o) = \int_{\Omega} p(\mathbf{x}, \omega_o \rightarrow \omega_i)\, L(\mathbf{x}, \omega_i)\, d\omega_i$$



integrates over the entire sphere/hemisphere

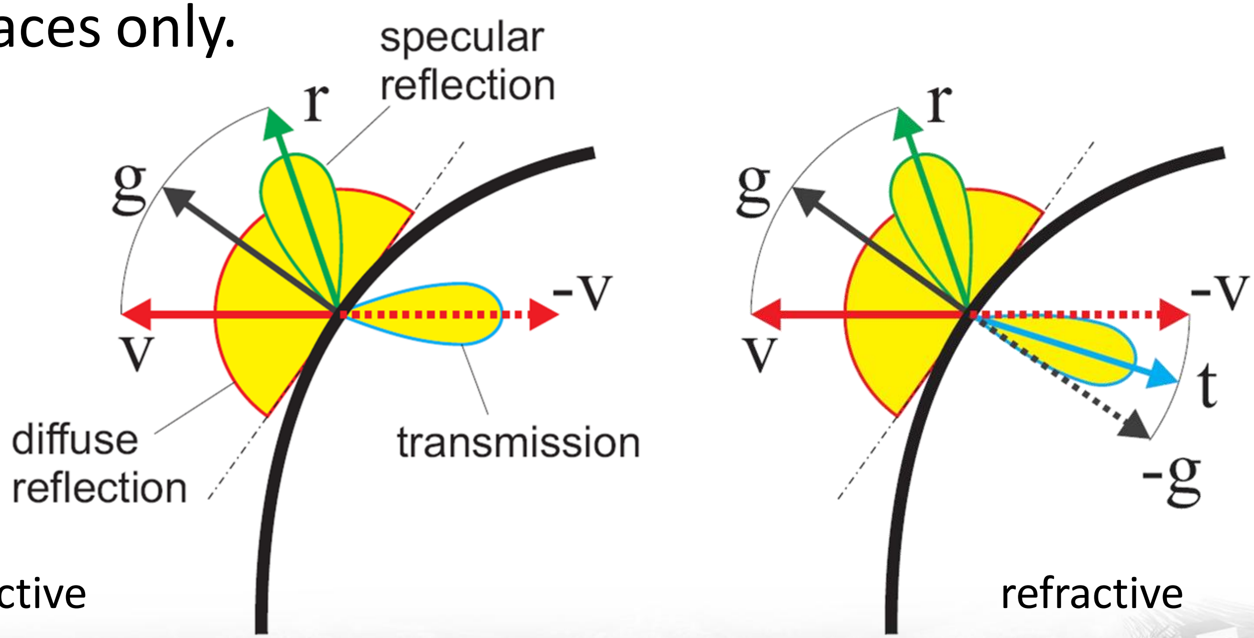
Quantum Optics

- Trace the path of single photons
- Photons are scattered randomly
- Probability of scattering direction given by BRDF/phase function
- ***Monte Carlo path tracing***



Phase Function Model

- Scattering of light at every point inside the volume
  - Too expensive (extremely slow convergence)
  - Not practicable. Controlling the visual appearance is difficult
- ***Idea:*** Restrict scattering events to a fixed number of isosurfaces only.



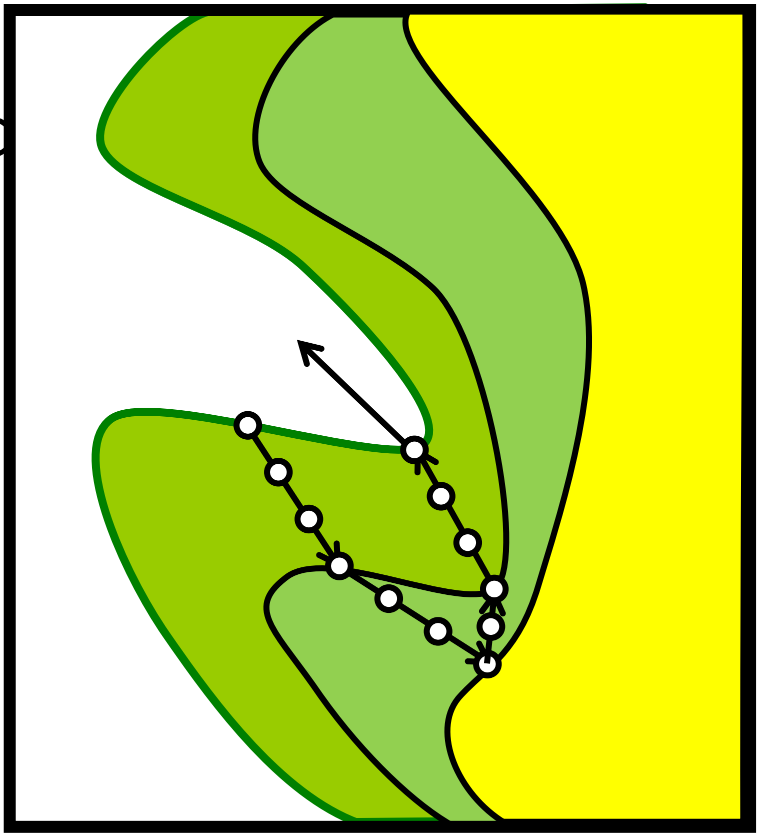
non-refractive

refractive

GPU Ray-Casting

Scattering Pass

- Start at first isosurface and trace inwards
- Account for absorption along the rays
- Proceed until next isosurface
- Calculate scattering event
- Sample the environment on exit

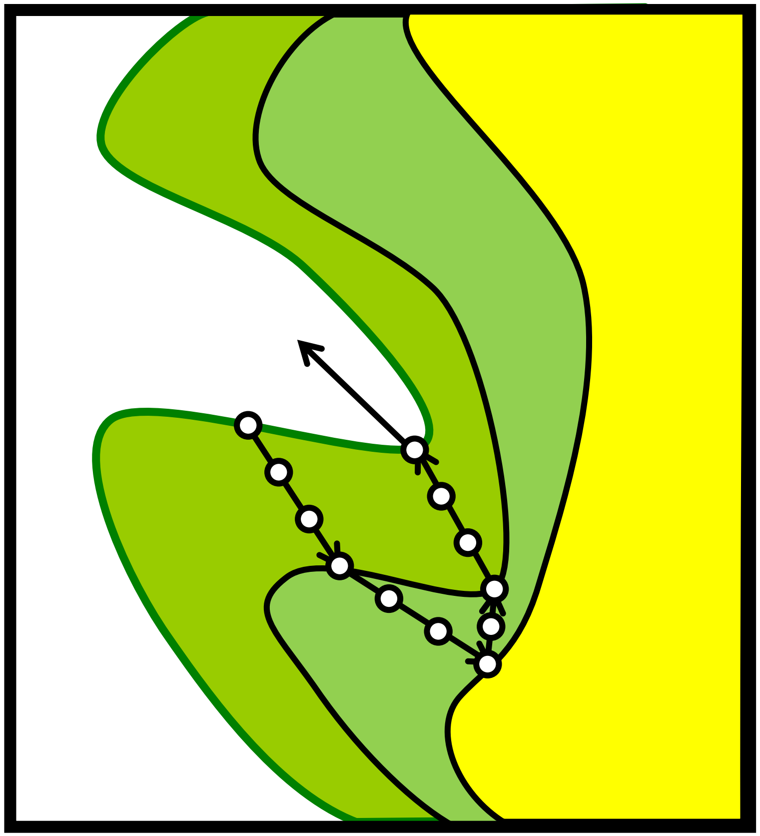


GPU Ray-Casting

Scattering Pass

Simplifying Assumption:

- Absorption on the „way in“ is same as on the „way out“
- Abort the ray inside the volume square the absorption and sample irradiance map
- ***Not very accurate but good visual results***

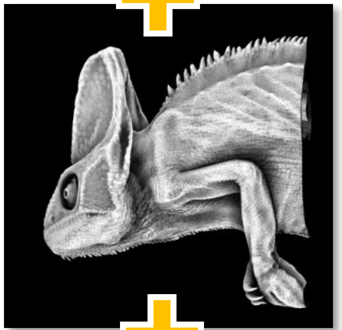


Scattering Pass

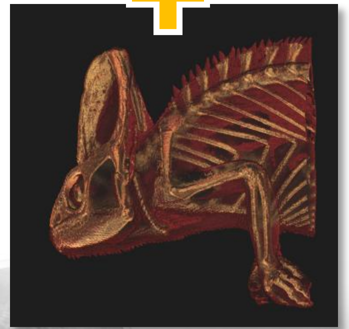
preview in real-time

final version in ½-1 seconds

Final Composite



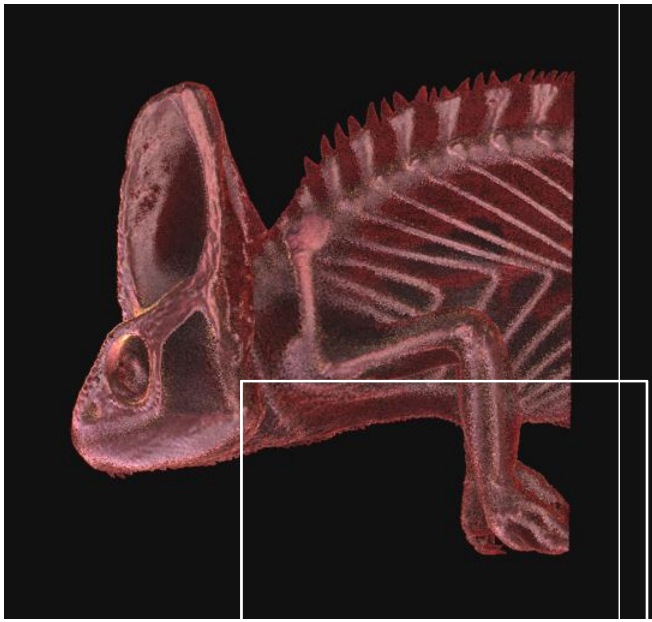
Multiply

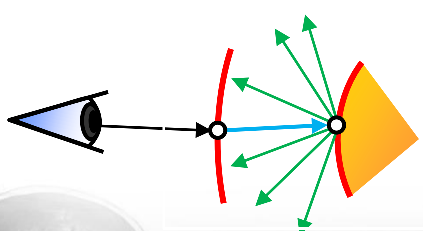


Blend
using
Fresnel term

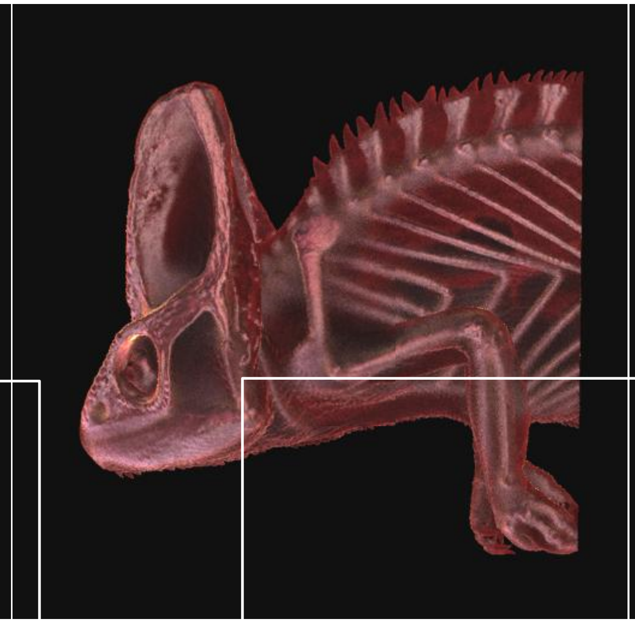
Path Tracing

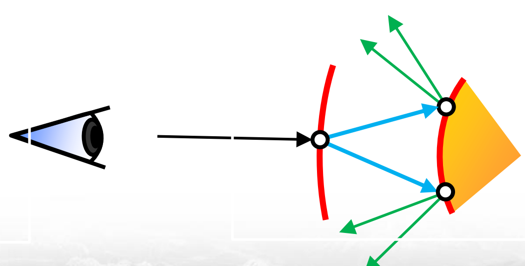
Primary rays: 1
Secondary rays: 64



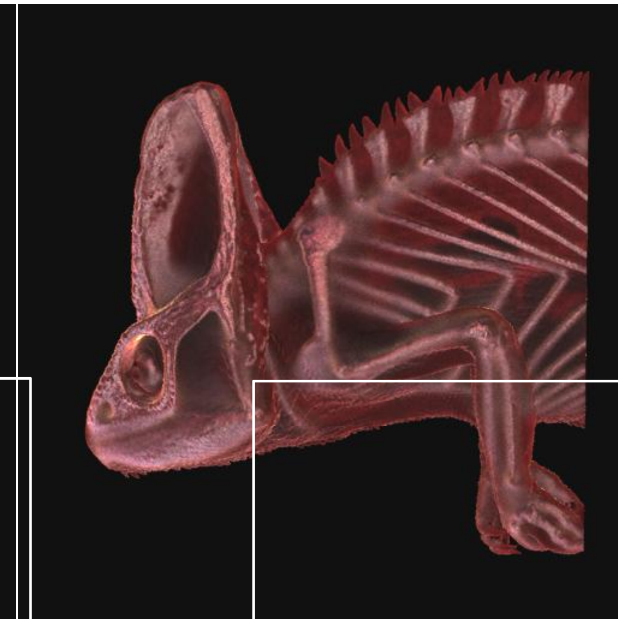


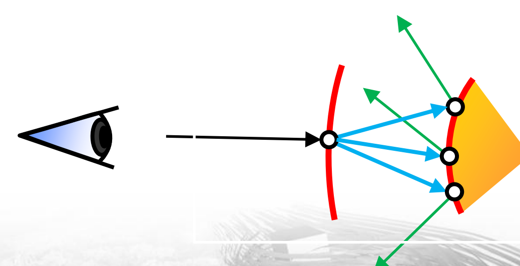
Primary rays: 8
Secondary rays: 8





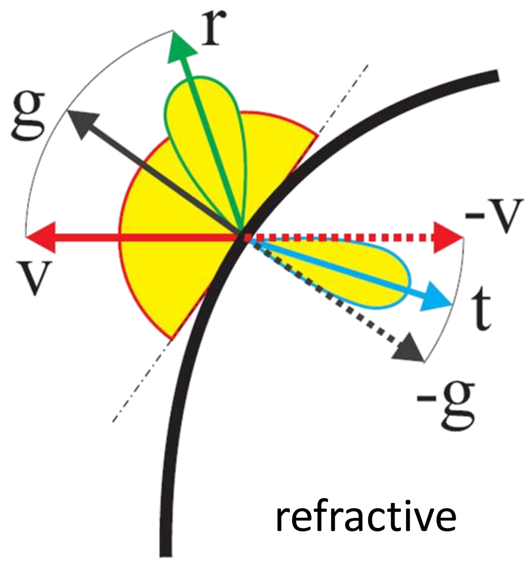
Primary rays: 64
Secondary rays: 1





Examples

Different scattering cone angles for the „inward-looking“ (transmissive) Phong-lobe



refractive