Volume Illumination

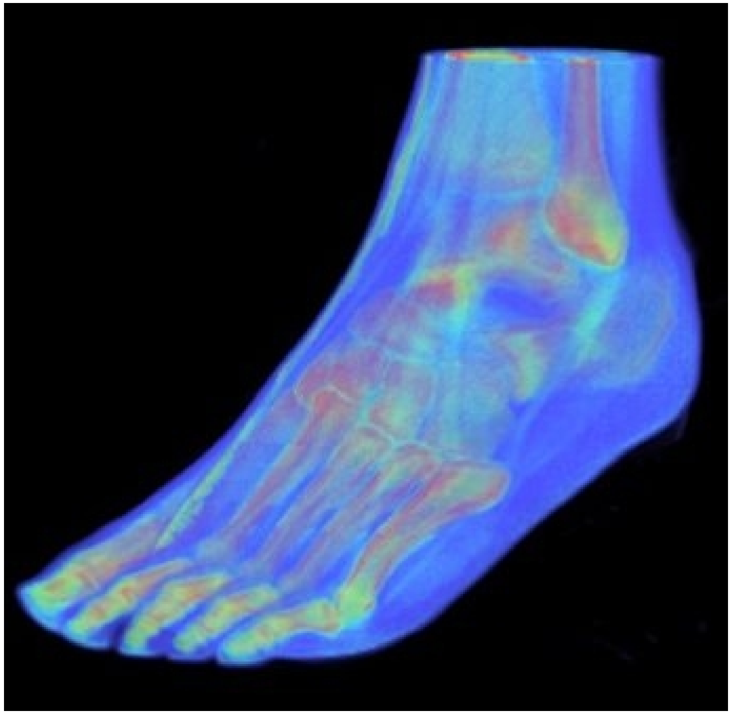
Visualisation – Lecture 11

Taku Komura

Institute for Perception, Action & Behaviour
School of Informatics

Previously : Volume Rendering

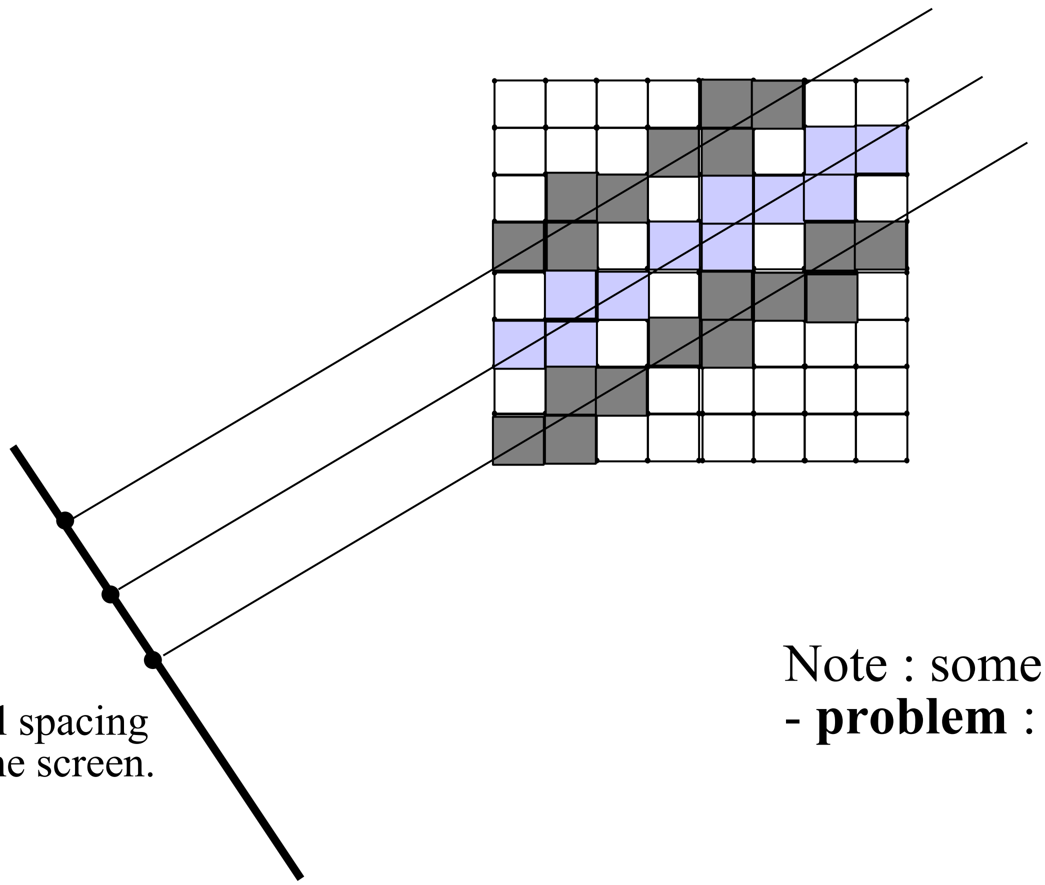
- **Image Order** Volume Rendering
  - **ray casting** / **intensity transfer** function / opacity transfer function
- Artifacts caused by the sampling method, step size etc.



Shear-warp Algorithm

- An efficient method to traverse the volume data [Lacroute '94]
- Assumes a **orthographic camera model**
  - projection perpendicular to image plane

Ray casting with templates

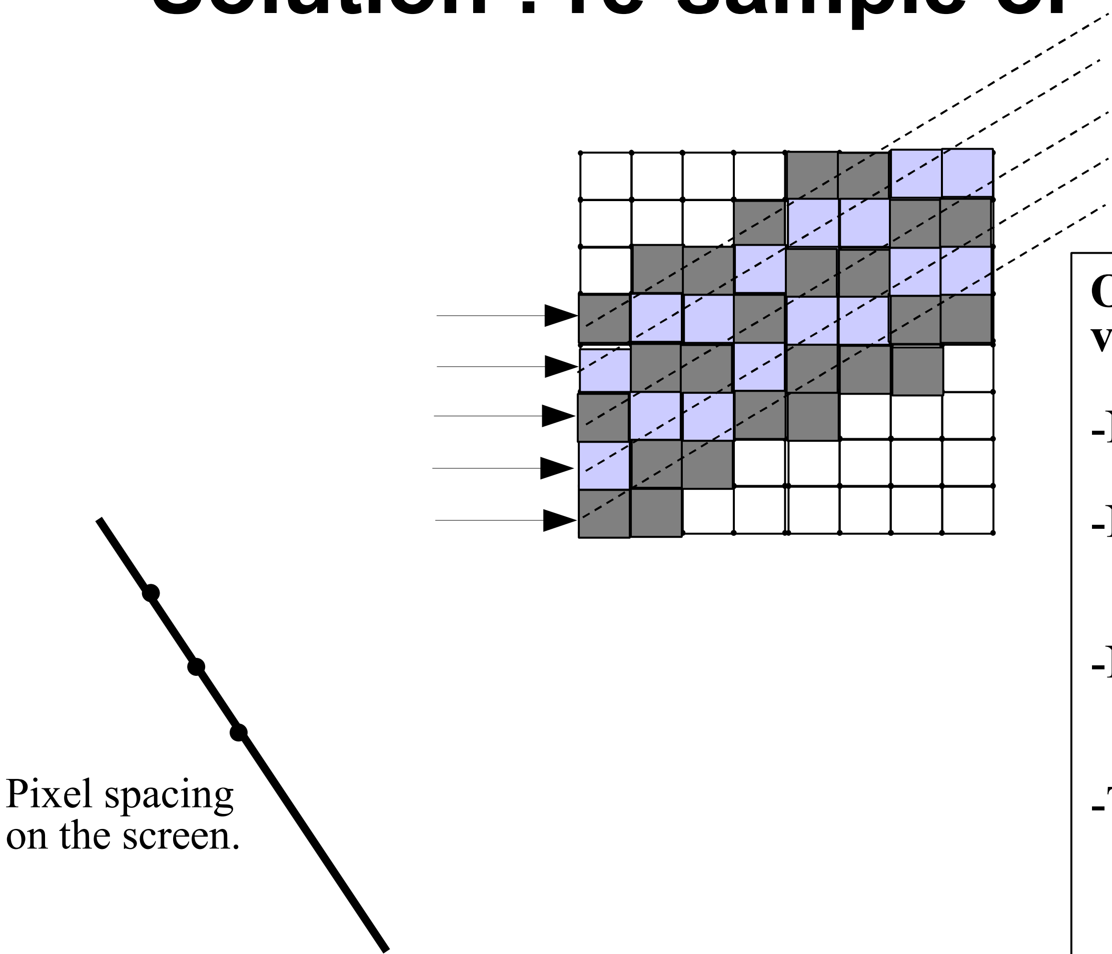


The rays are parallel:

- Each ray forms an identical path of voxels through the grid
- Path known as a *template*
- Save computation

Note : some voxels are missed
- **problem** : artefacts will be produced.

Solution : re-sample or *warp* template



Originate templated rays in each voxel at the edge of the grid

-No gaps.

-Need to **resample the samples onto the screen.**

-Note some voxels visited many times.

-This is **re-sampling in 3D, but is *warping* in the 2D screen space.**

- Perform warping to get samples back into 2D image grid correctly
  - amount of warp required dependant on viewing angle to volume

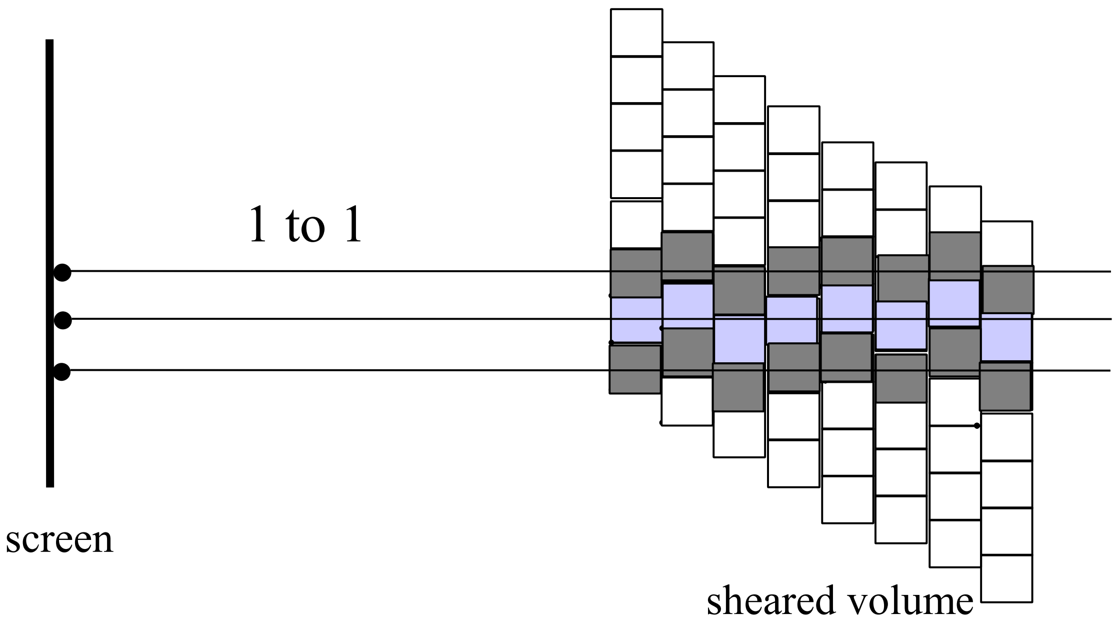
Shear-warp factorisation

- Instead of traversing along rays, **visit voxels in a plane**
  - regardless of camera viewing angle
- Use **front-to-back ordering**
  - support early termination (last lecture)

- Perform final warp on the image due to the shear process

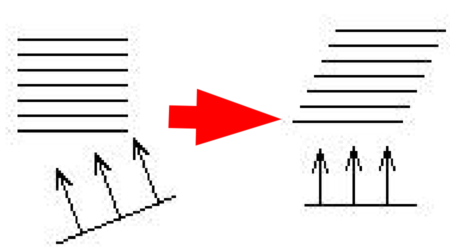
- Improve efficiency using run-length encoding of voxels
  - Skipping the transparent voxels

Transform to sheared space

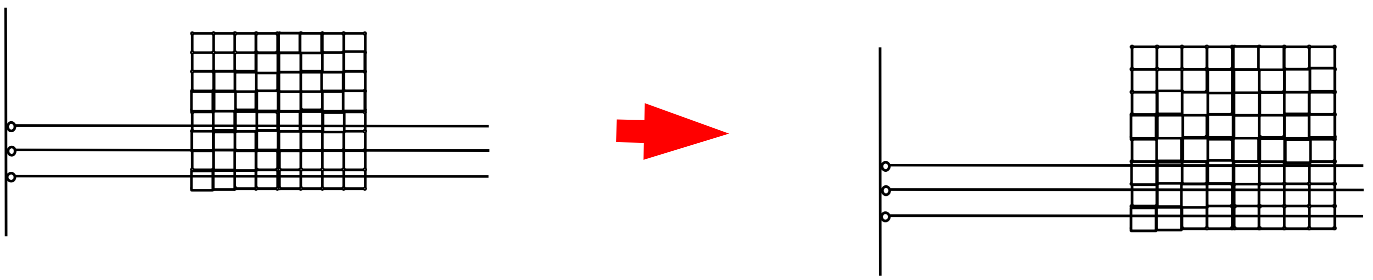


- Rays are cast from the base plane voxels at the same place.
- They **intersect voxels on subsequent planes in the same location.**
- Only **one set of interpolation weights needs to be computed for all the voxels in a plane.**

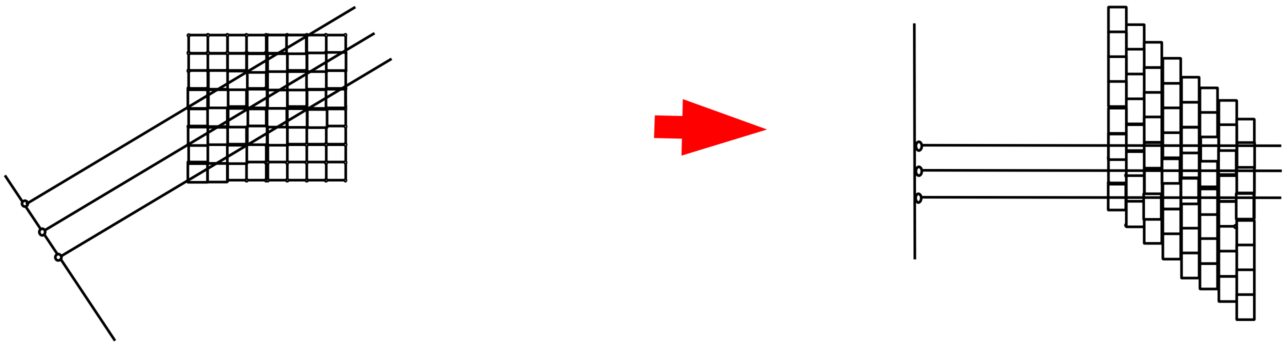
- **volume is sheared** so that rays remain perpendicular to base plane
  - lead to **efficient ray computation**



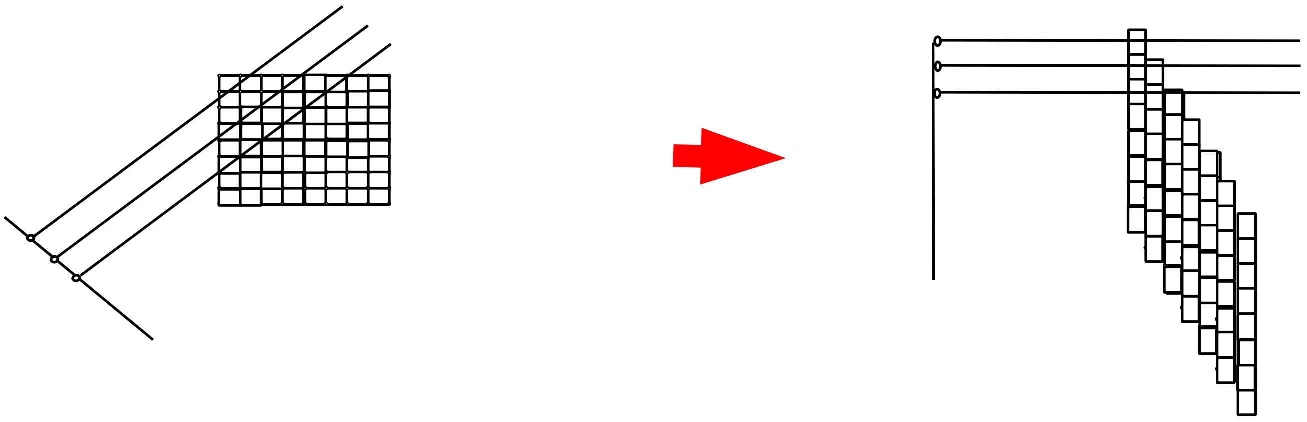
Rotation of camera with *shear-warp*



No rotation



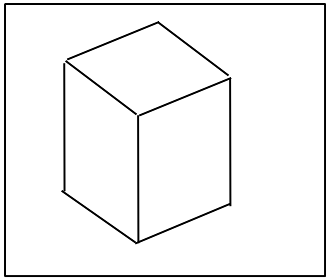
Small angle



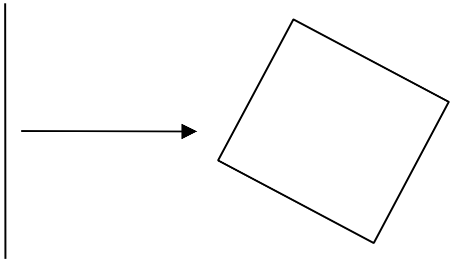
Large angle

Final stage of shear-warp

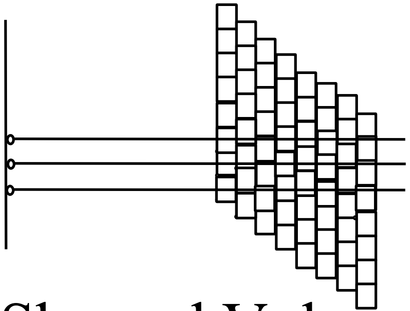
- Final stage of re-sampling (**warp**) required
  - transform resulting image from sheared space to regular Cartesian space (image plane).



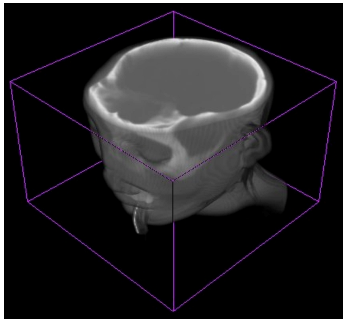
View of Volume

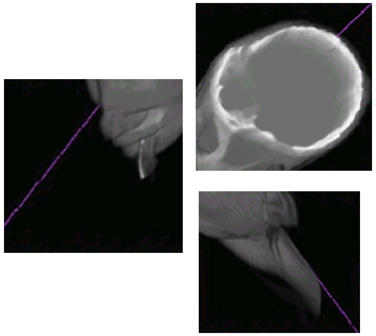


Top down view

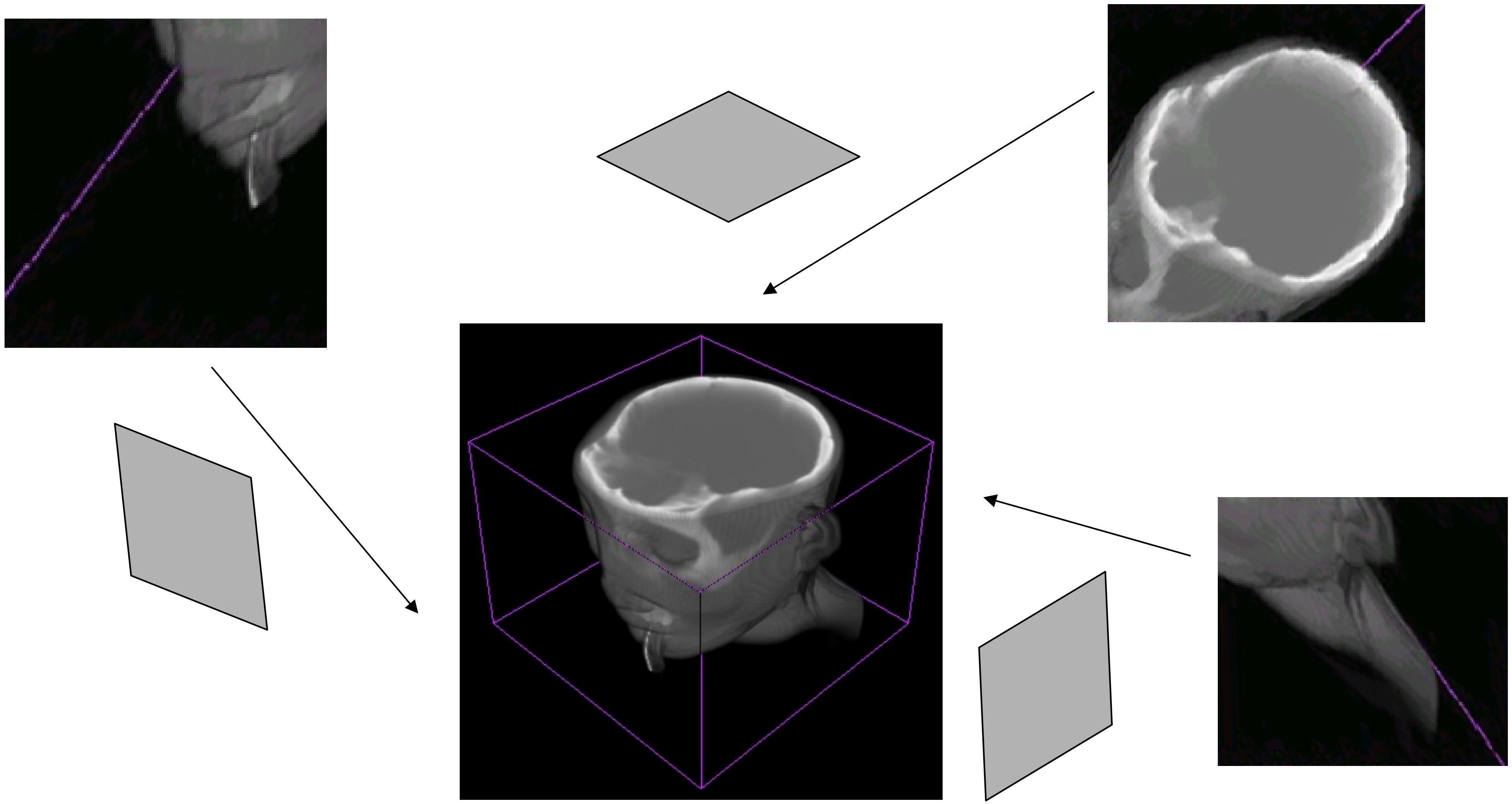


Sheared Volume





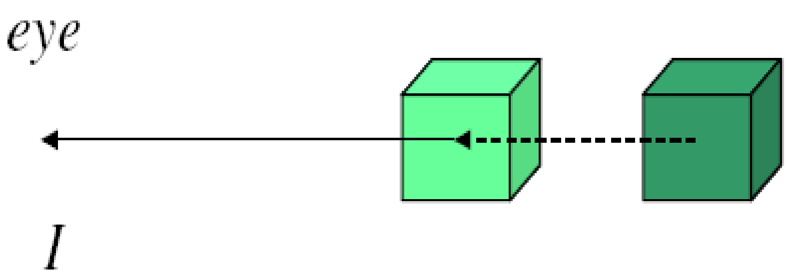
Warping the images



Light Propagation in Volumes

- **Lighting in volume**
  - only **transmission and emission** considered (until now)
  - can also:
    - **reflect light**
    - **scatter light** into different directions

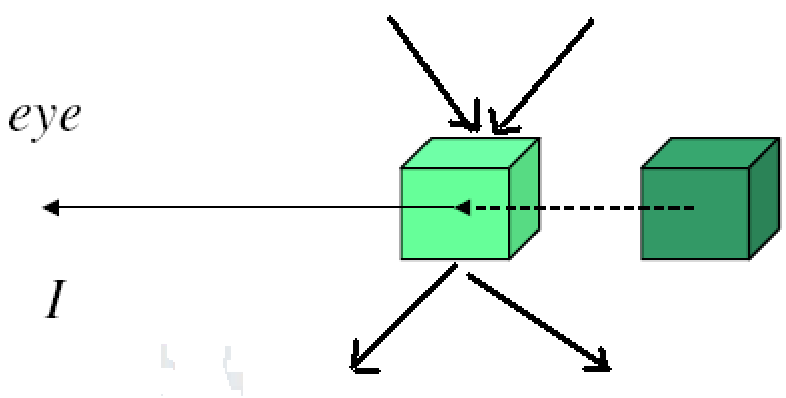
Global Illumination of Volumes

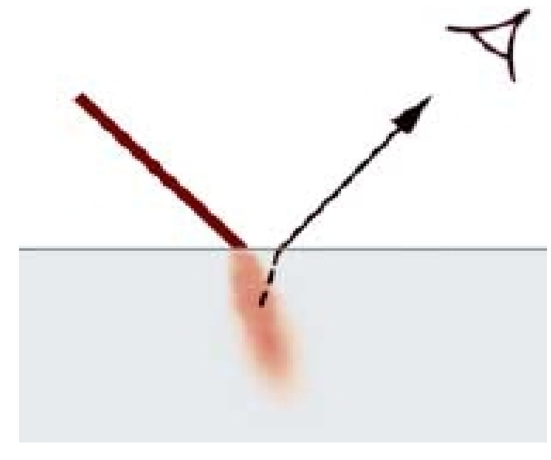


For every voxel ray intersects, need to consider:

- **Light absorbed.**
- **Light emitted.**
- Light scattered out of the ray.
- Light scattered into the ray.

Global Illumination of Volumes





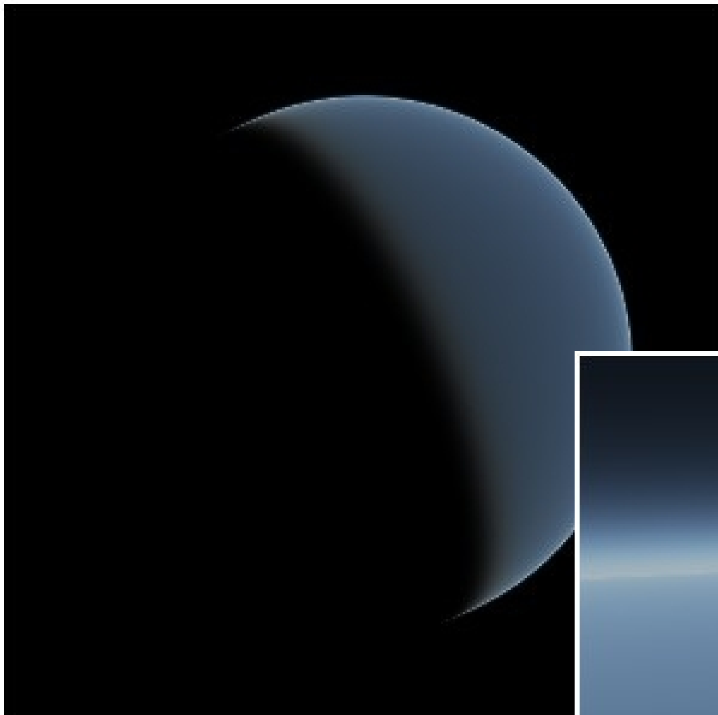
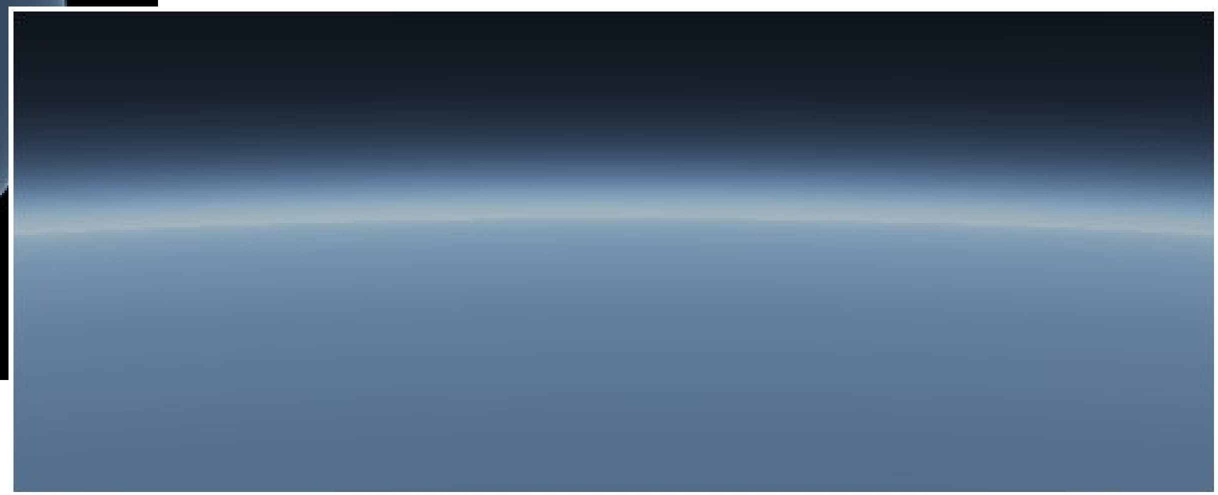
For every voxel ray intersects, need to consider:

- Light absorbed.
- Light emitted.
- **Light scattered out of the ray.**
- **Light scattered into the ray.**

Normally ignore scattering in volumetric illumination !

Why ? : computational cost

Example: single scatter

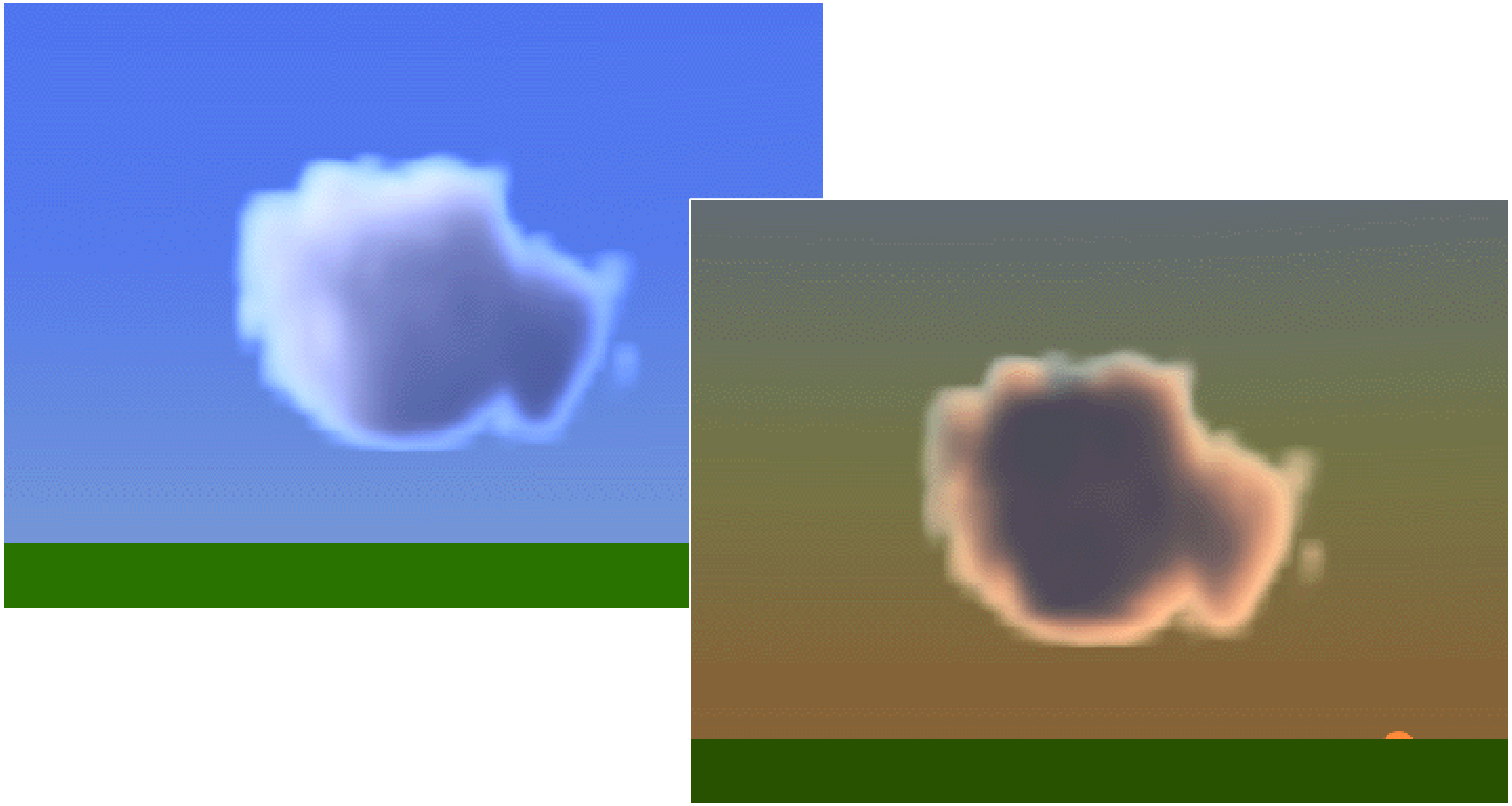



Synthetic images of the Earth's atmosphere produced by Rayleigh scattering.

Irwin 95.

- Light scattered to produce atmospheric haze effect

Example : multiple scattering



- Light scattered multiple times to produce simulation of a cloud

Example : sub-surface scattering


Very large statue


Medium sized


Small statue.

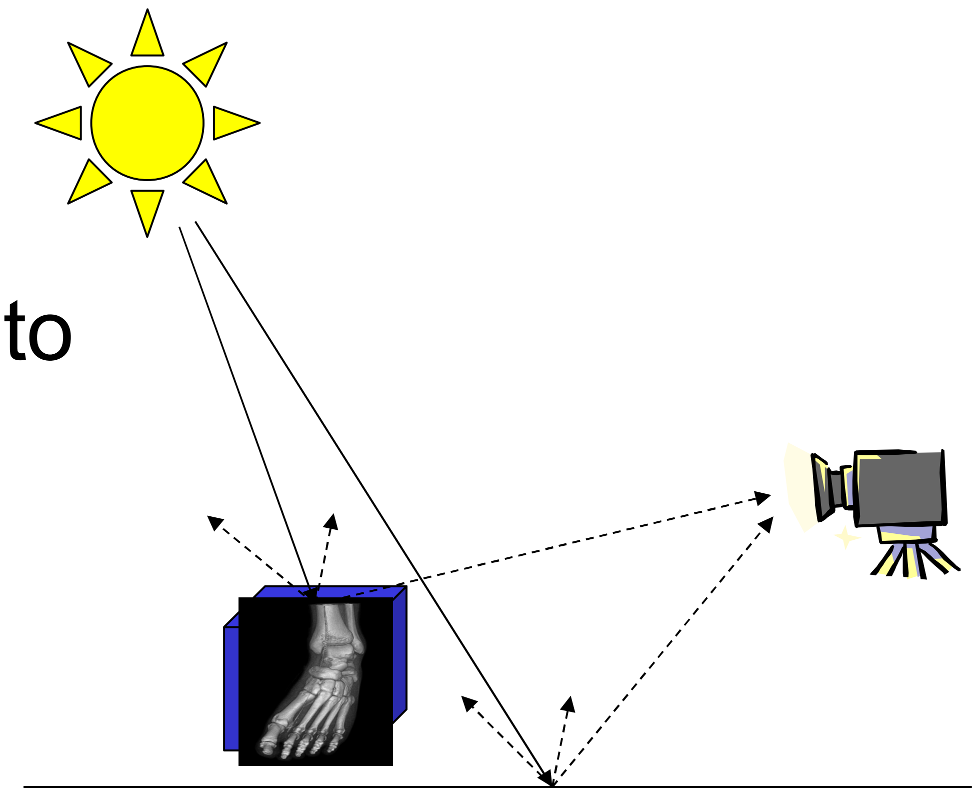
- Varying **how light is scattered inside a surface affects perception**
  - hence **useful in visualisation**
  - **why ? -** prior visual experience, perception of distance ?

Volume Illumination - ?

- Scattering is too costly so we usually do not take them into account when doing volume rendering
- But we still can add slight shadows to the volume by illuminating them

Volume Illumination

- Why do we want to illuminate volumes?
- **illumination helps us to better understand 3D structure**
  - displays visual cues to **surface orientation**
  - highlight significant **gradients within volume**



What are we illuminating ?

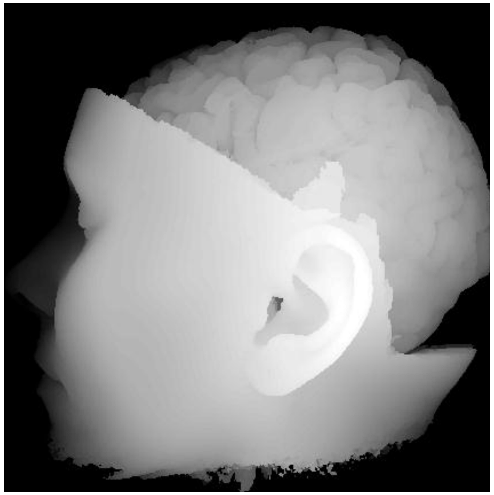
- **embedded (iso-) surface**
- sharp **gradients in opacity**

Shading an Embedded iso-surface

- classify volume with a step function
- use regular specular / diffuse surface shading
- **Remember** for lighting equations of lecture 2 require
  - illumination direction
  - camera model (position)
  - surface orientation
  - **need** to calculate and store **surface normal**

Estimating the surface normal from distance map

- Use distance map to the iso-surface value



1. **Determine the threshold value**
2. **Determine the surface voxels based on the threshold**
3. **Compute the normal vectors based on centred difference method**

For example, if we sample the centre of the voxels,

$$\delta z_x = \frac{1}{2}\left(z(x+1, y) - z(x-1, y)\right)$$

$$\delta z_y = \frac{1}{2}\left(z(x, y+1) - z(x, y-1)\right)$$

$$N = \left(\frac{-\delta z_x}{\delta z_x^2 + \delta z_y^2 + 1}, \frac{-\delta z_y}{\delta z_x^2 + \delta z_y^2 + 1}, \frac{1}{\delta z_x^2 + \delta z_y^2 + 1}\right)$$

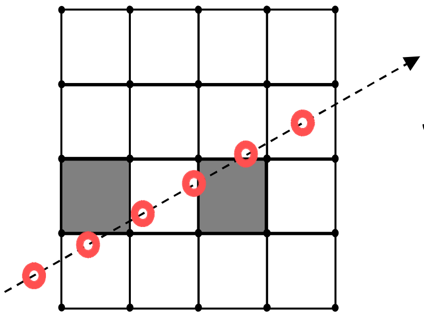
Result : illuminated iso-surface

MIP technique

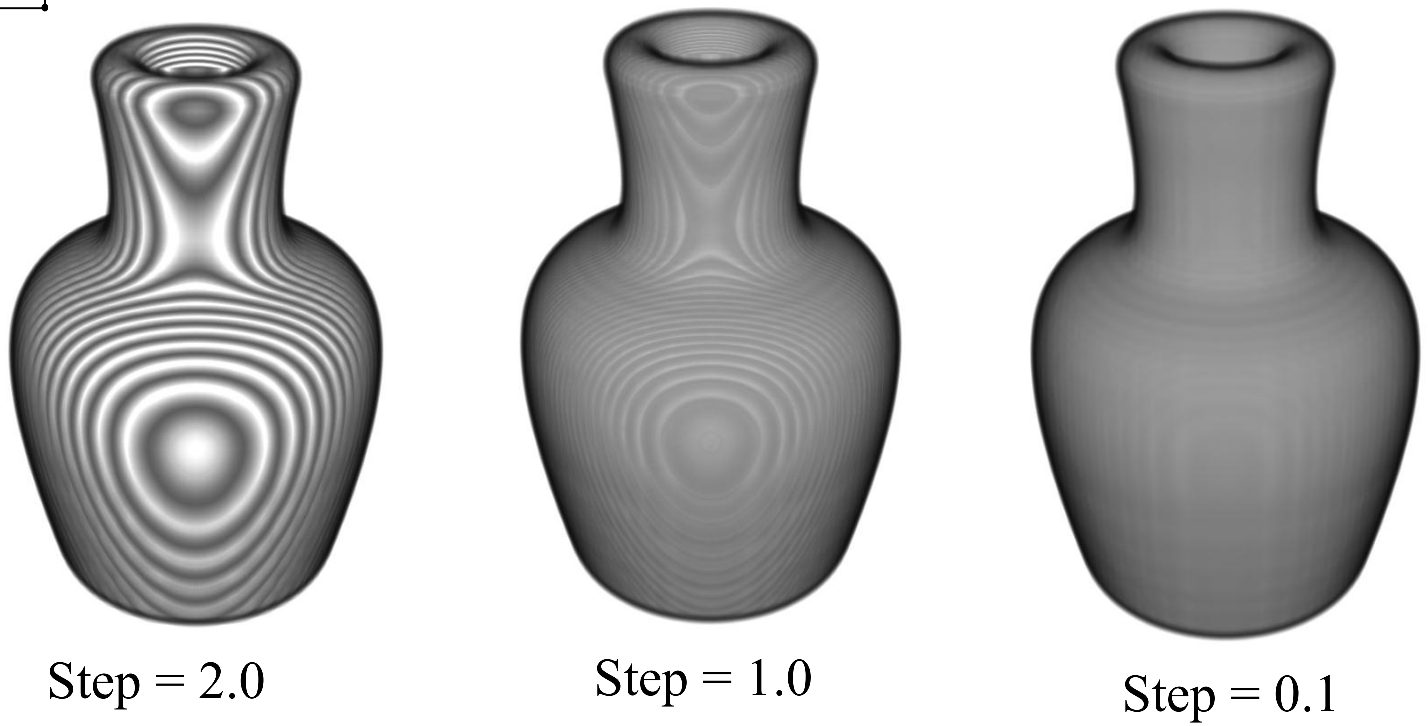
Shaded embedded iso-surface.

- Surface normals recovered from depth map of surface

Normals are sensitive to step size



Step size through the volume / over the depth map



Artefacts with larger step sizes under standard lighting model

Illuminating Opacity (Scalar) Gradient

- **Illuminate "scalar gradient"** instead of iso-surface
  - requirement : estimate and store gradient at every voxel

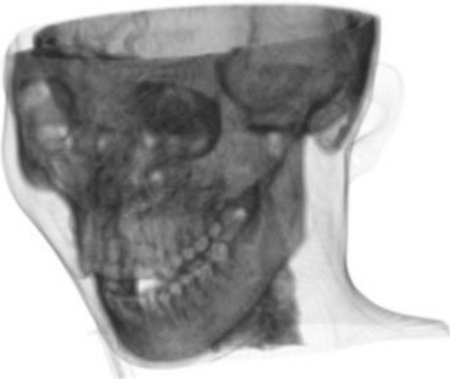
Composite

Shaded Composite

Shaded opacity gradient (shades changes in opacity)

Estimating Opacity Gradient

- Use **3D centred difference operator**



$$\nabla I = (I_x, I_y, I_z) = \left(\frac{\delta}{\delta x} I, \frac{\delta}{\delta y} I, \frac{\delta}{\delta z} I\right)$$

$$\frac{\delta}{\delta x} I = \frac{I(x+1, y, z) - I(x-1, y, z)}{2}$$

We can extract the normal vectors of the region where the scalar values are changing significantly, i.e. boundary of tissues

- **Evaluate at each voxel and interpolate**

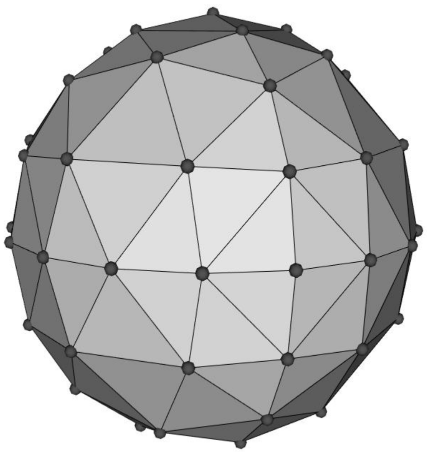
Illumination : storing normal vectors

- Visualisation is **interactive**
  - **compute normal vectors for surface/gradient once**
  - **store normal**
  - **perform interactive shading calculations**

- **Storage :**
  - 256^3 data set of 1-byte scalars ~16Mb
  - normal vector (stored as floating point(4-byte)) ~ 200Mb!
  - **Solution** : quantise direction & magnitude as small number of bits

Illumination : storing normal vectors

- Quantize vector direction into one of N directions on a sub-divided sphere

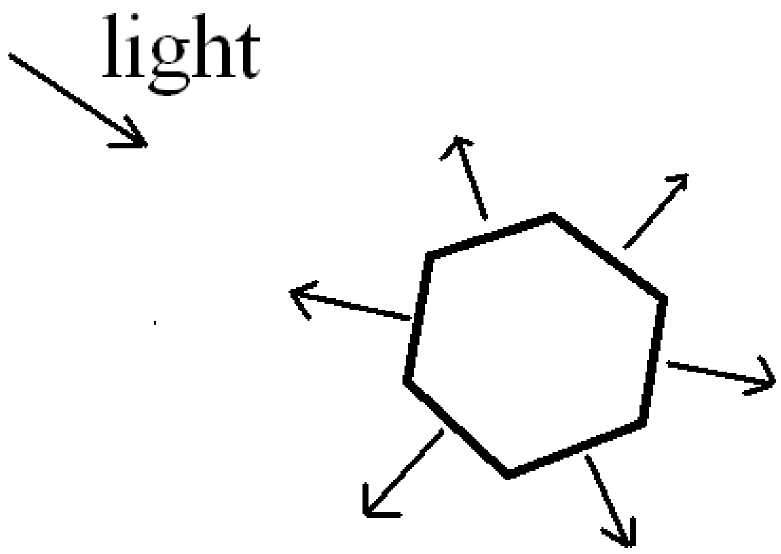


Subdivide an octahedron into a sphere.

Number the vertices.

Encode the direction according the nearest vertex that the vector passes through.

For infinite light sources, only need to calculate the shading values once and store these in a table.



Summary

- Shear-Warping
- Light scattering
- Volume illumination