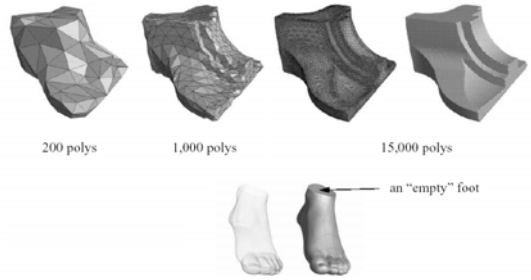


Visualization

Surface

- a mesh of polygons:



Surface - Pros and Cons

- Pros:
 - fast rendering algorithms are available
 - acceleration in special hardware is relatively easy and cheap (many \$100 game boards)
 - use OpenGL to specify rendering parameters
 - surface realism can be added via texture mapping
-

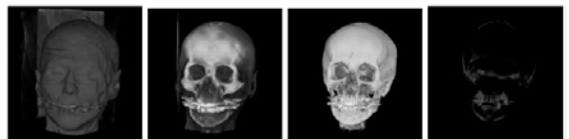
Surface - Pros and Cons

- Cons:
 - discards the interior of the object and just maintains the object's shell
 - does not facilitate real-world operations such as cutting, slicing, direction
 - does not enable artificial viewing modes such as semi-transparencies, X-ray
 - surface-less phenomena such as clouds, fog, gas are hard to model and represent
-

Volumes with direct rendering

- Maintains a representation that is close to the underlying fully-3D object (but discrete)
 - Models the object as a magic gel that can change its properties at any time.
 - Different aspects of the dataset can be emphasized via changes in the functions that translate raw densities into colors and transparencies
 - Volume rendering is a formidable technique for the *exploration* of datasets, since when the nature of the data is not known, it is difficult to create the right polygonal mesh
-

Direct Volume Rendering

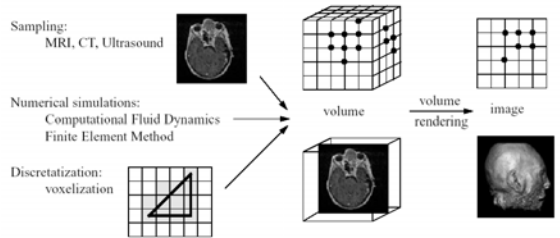


Volumes with direct rendering

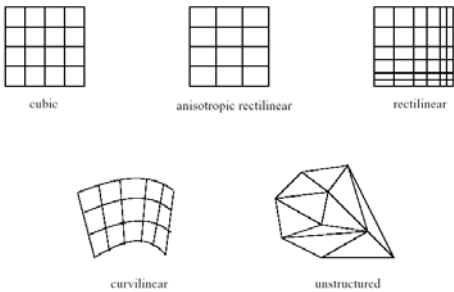
- Pros:
 - maintains a representation that is close to the underlying fully-3D object (but discrete)
 - can achieve a level of realism (and 'hyper-realism') that is unmatched by surface graphics.
 - allows easy and natural exploration of volumetric datasets
- Cons:
 - has high rendering complexity
 - hardware acceleration is complex and expensive (a commodity board costs over \$3,000)

Volume Rendering

- The process of generating a 2D image from the 3D volume is called *volume rendering*

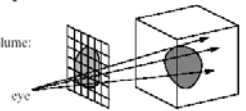


Volume Grid Types

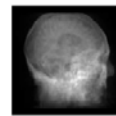


Volume Rendering Modes

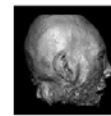
- For each pixel in the image, a ray is cast into the volume:



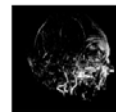
- Four main volume rendering modes exist:



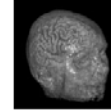
X-ray:
rays sum volume contributions along their linear paths



Iso-surface:
rays look for the object surfaces, defined by a certain volume value

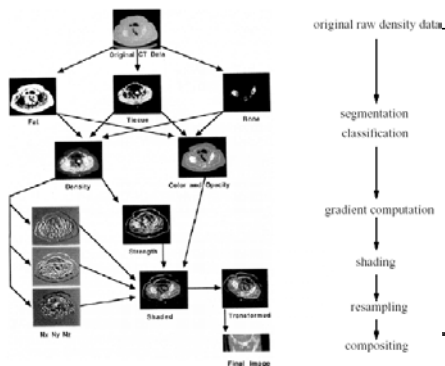


Maximum Intensity Projection (MIP):
a pixel value stores the largest volume value along its ray

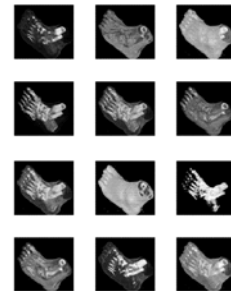


Full volume rendering:
rays composite volume contributions along their linear paths

Volume Rendering Pipeline

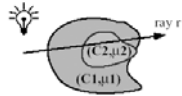


Full Volume Rendering



Raycasting

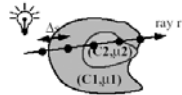
- Consider a volume consisting of particles:
 - each has color C and light attenuating density μ
- A rendering ray accumulates attenuated colors
- We write the continuous volume rendering integral:



$$I_\lambda(x, r) = \int_0^L C_\lambda(s) \mu(s) e^{-\int_0^s \mu(t) dt} ds \quad (\text{this is generally not solvable analytically})$$

- We can approximate it by discretizing it into sampling intervals of width Δs :

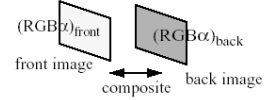
$$I_\lambda(x, r) = \sum_{i=0}^{L/\Delta s} C_\lambda(i\Delta s) \mu(i\Delta s) \Delta s \cdot \prod_{j=0}^{i-1} e^{-\mu(j\Delta s) \Delta s}$$



Raycasting

- It is the accumulation of colors weighted by opacities
- Colors and opacities of back pixels are attenuated by opacities of front pixels:

$$rgb = RGB_{back} \cdot \alpha_{back} (1 - \alpha_{front}) + RGB_{front} \cdot \alpha_{front}$$



- Volume rendering uses this recursive expression to combine (=composite) the samples taken along the ray

Chalk Talk