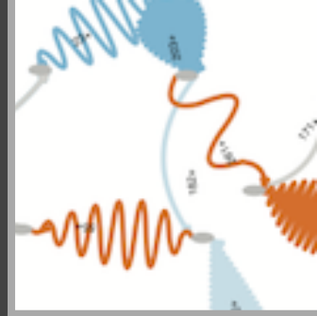
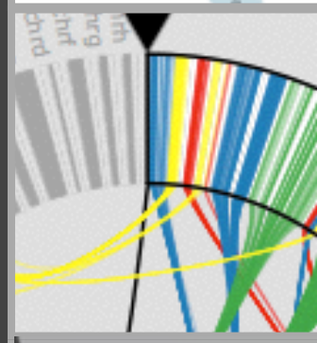
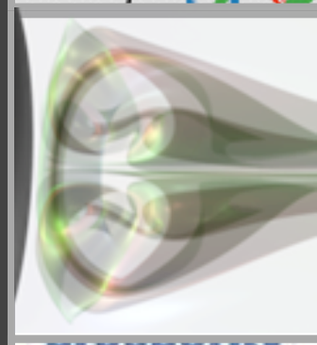
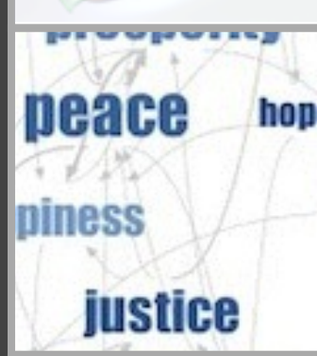
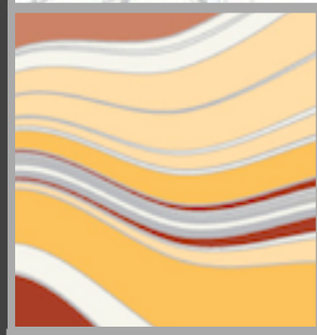
cs6630 | November 6 2014

3D GRAPHICS

Miriah Meyer
University of Utah

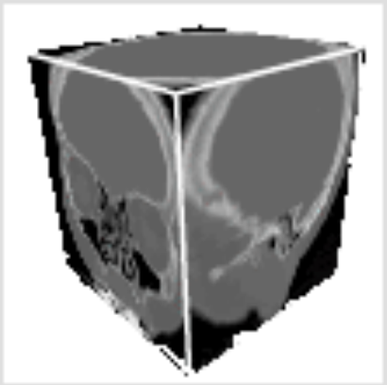





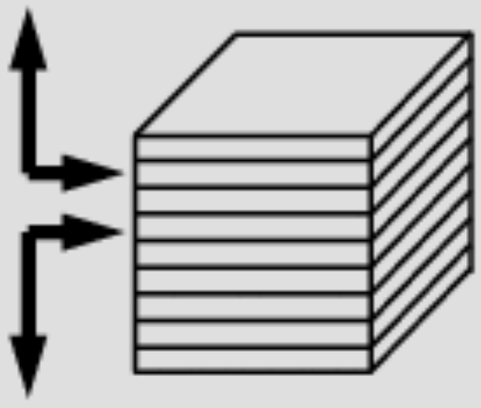


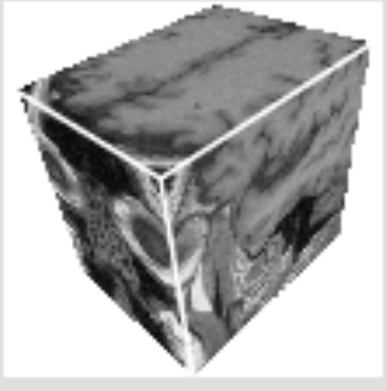

administrivia . . .

- lectures and office hours next week
- new due date for final assignment

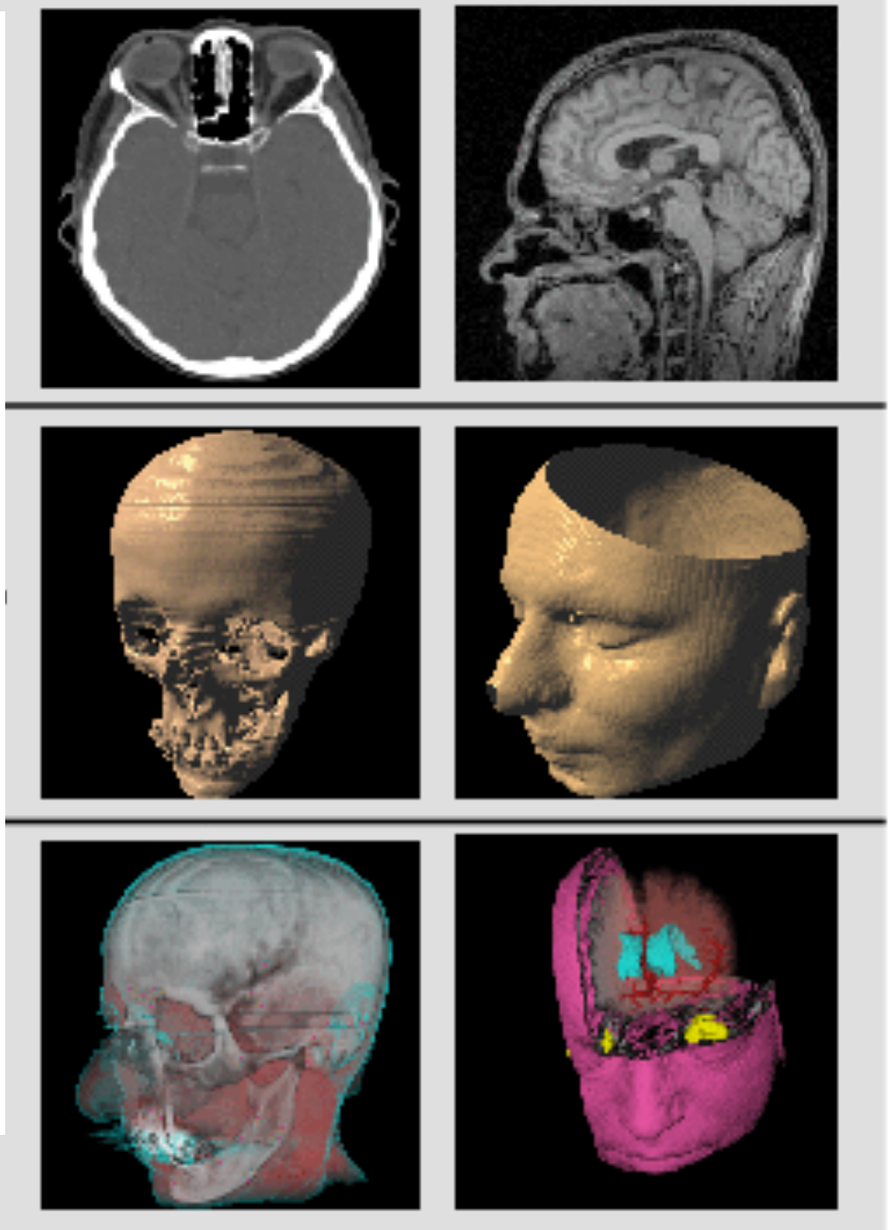
last time . . .

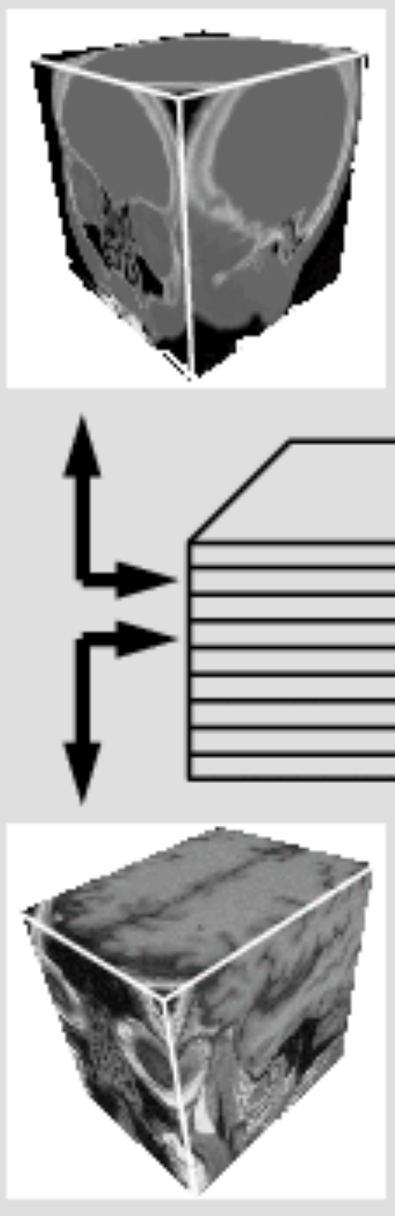




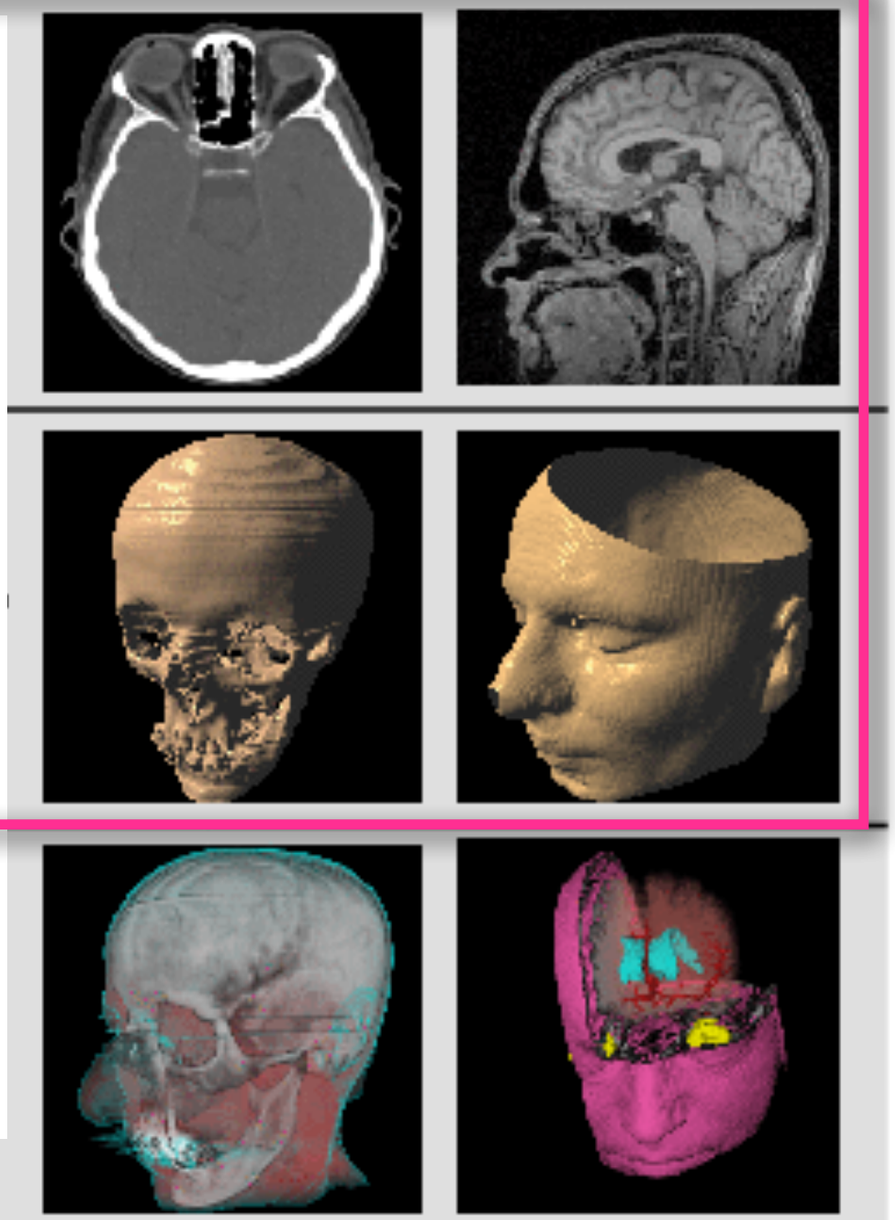


- 2D visualization
  slice images
  (or multi-planar
  reformating MPR)
- *Indirect*
  3D visualization
  isosurfaces
  (or surface-shaded
  display SSD)
- *Direct*
  3D visualization
  (direct volume
  rendering DVR)

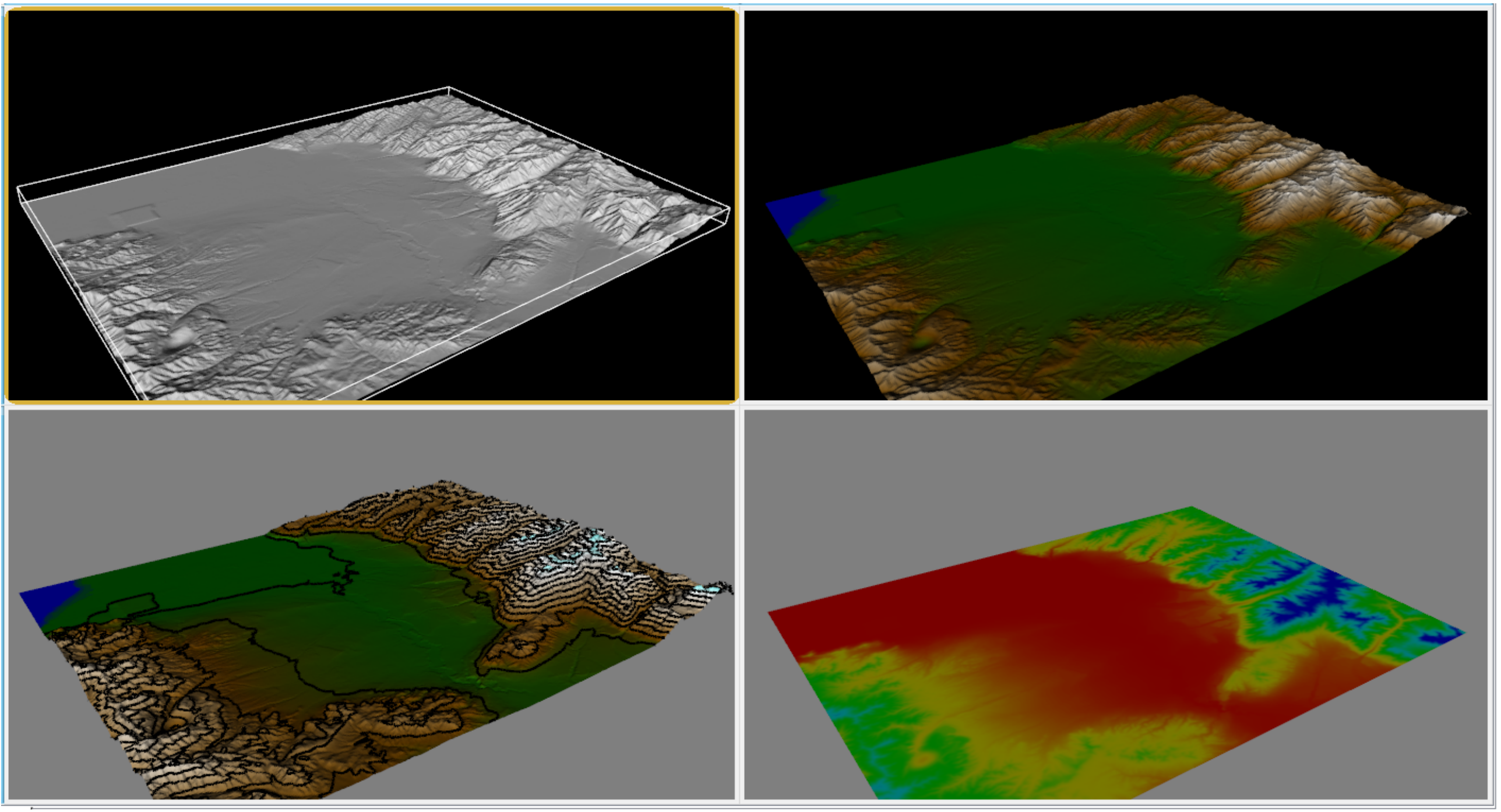




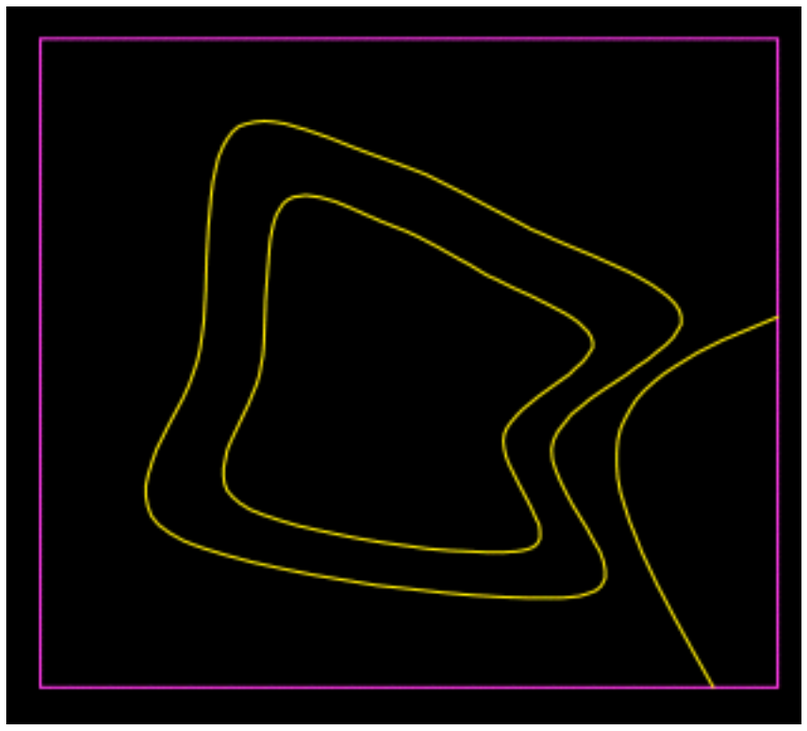
- 2D visualization
  slice images
  (or multi-planar
  reformating MPR)
- *Indirect*
  3D visualization
  isosurfaces
  (or surface-shaded
  display SSD)
- *Direct*
  3D visualization
  (direct volume
  rendering DVR)



Compare



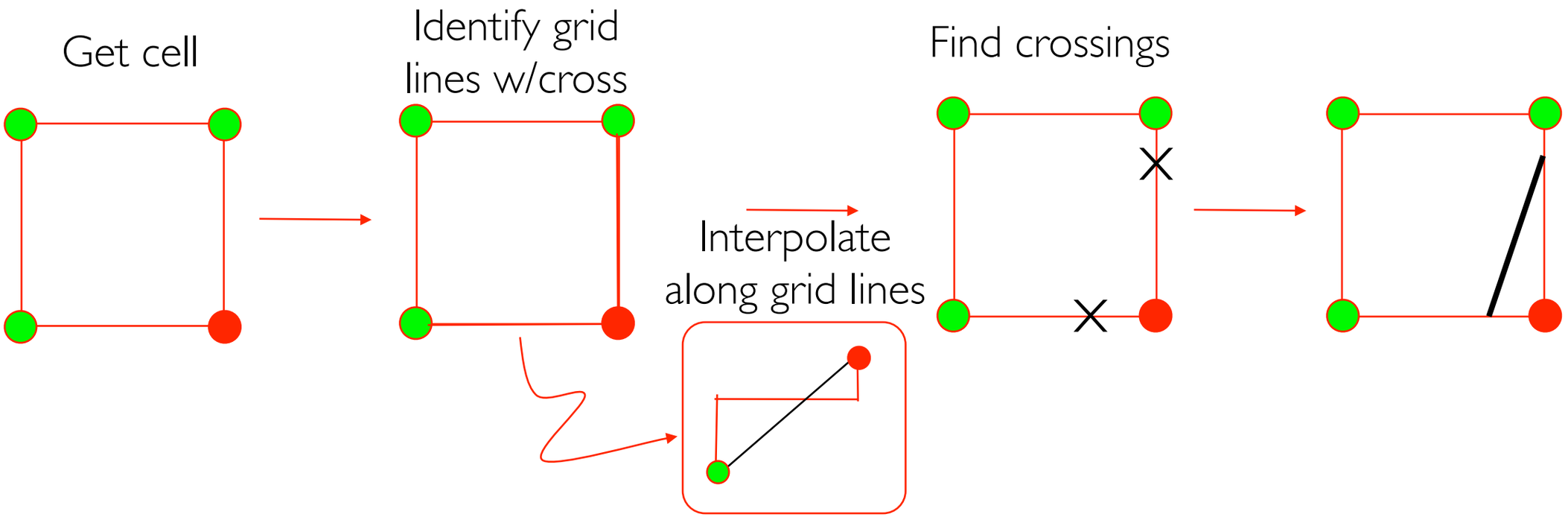
Properties



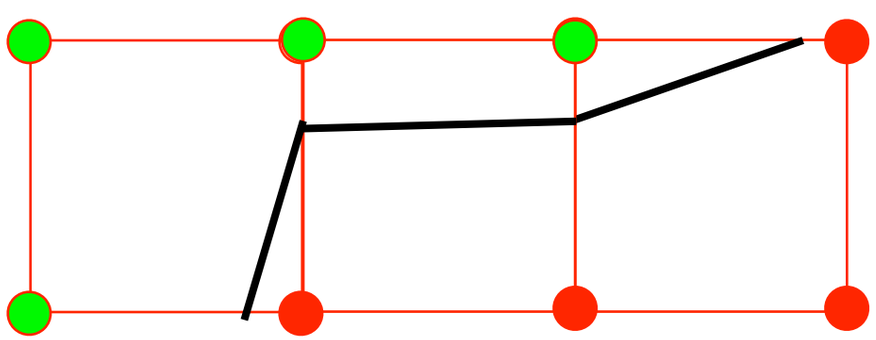
- **Preimage** of a single scalar value:
  - Concept generalizes to any dimension
- **Closed** except at boundaries
- **Nested**: isocontours of different isovalues do not cross
  - Can consider the zero-set case (generalizes)
  - $f(x,y) = v \Leftrightarrow f(x,y) - v = 0$
- Normals given by gradient vector of f()

Approach to Contouring in 2D

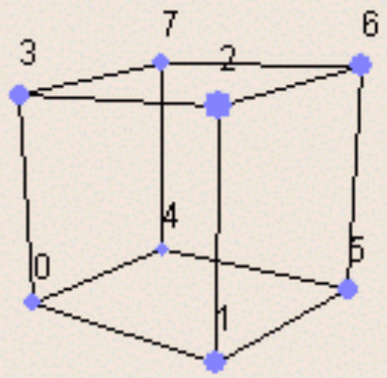
- Contour must cross every grid line connecting two grid points of opposite sign



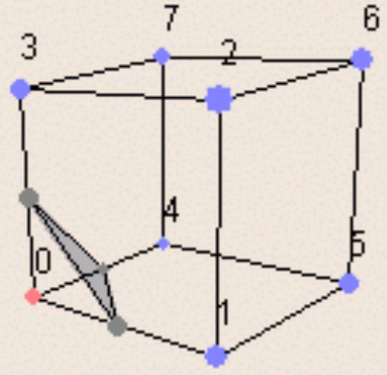
Primitives naturally chain together



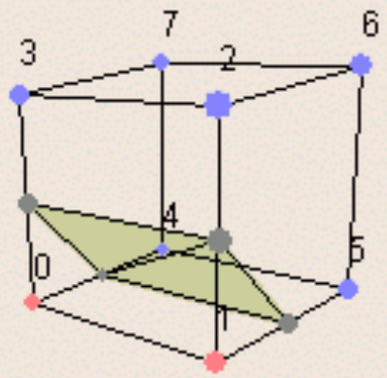
Case Table



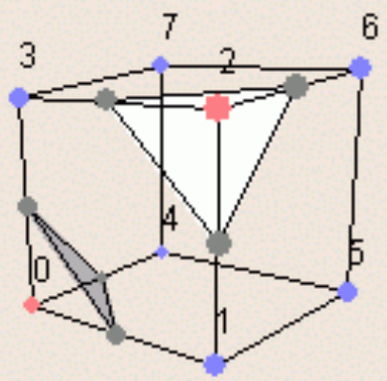
Case 0



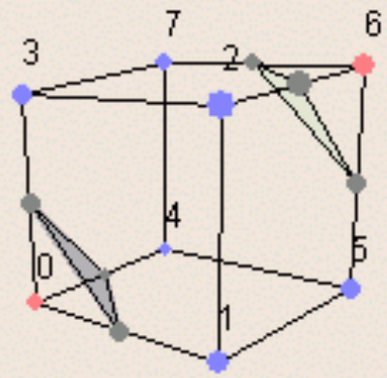
Case 1



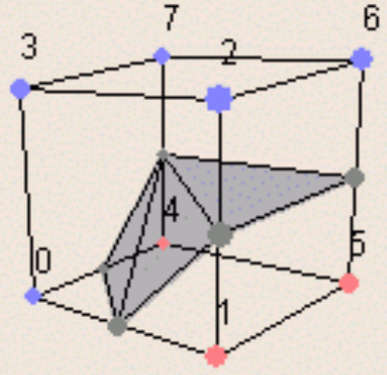
Case 2



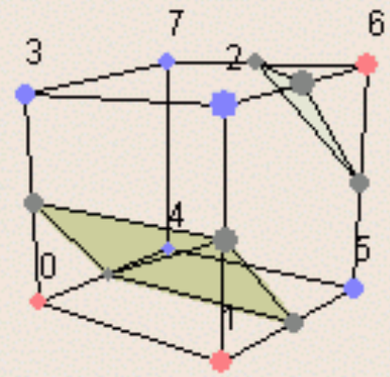
Case 3



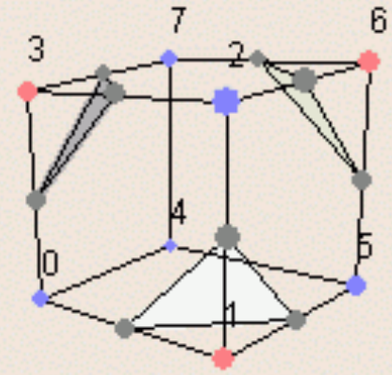
Case 4



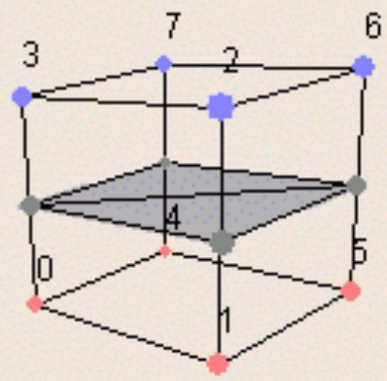
Case 5



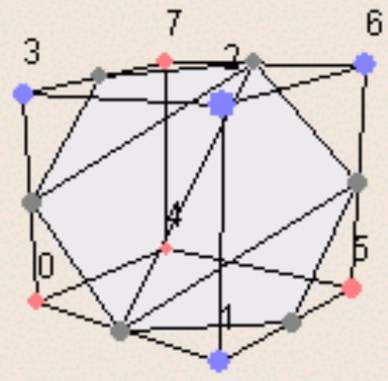
Case 6



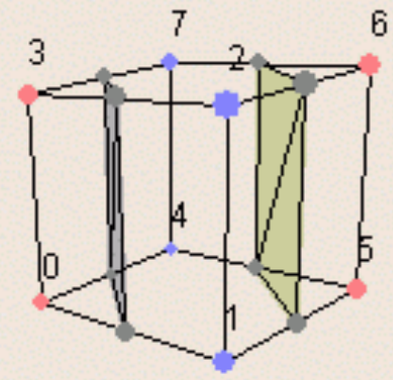
Case 7



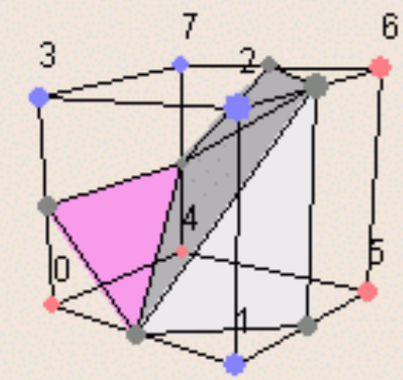
Case 8



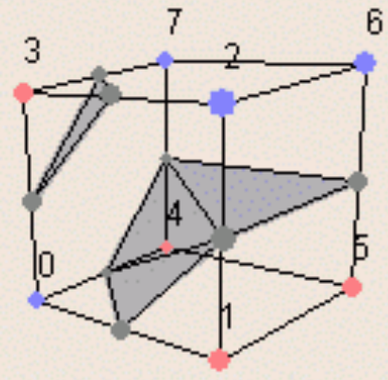
Case 9



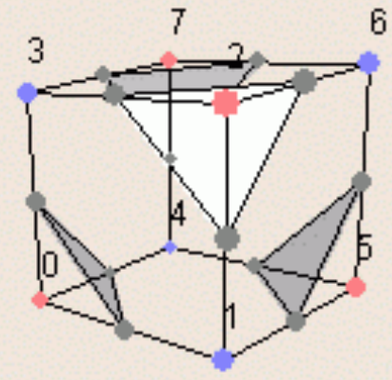
Case 10



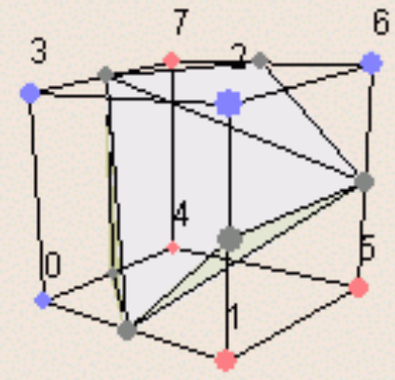
Case 11



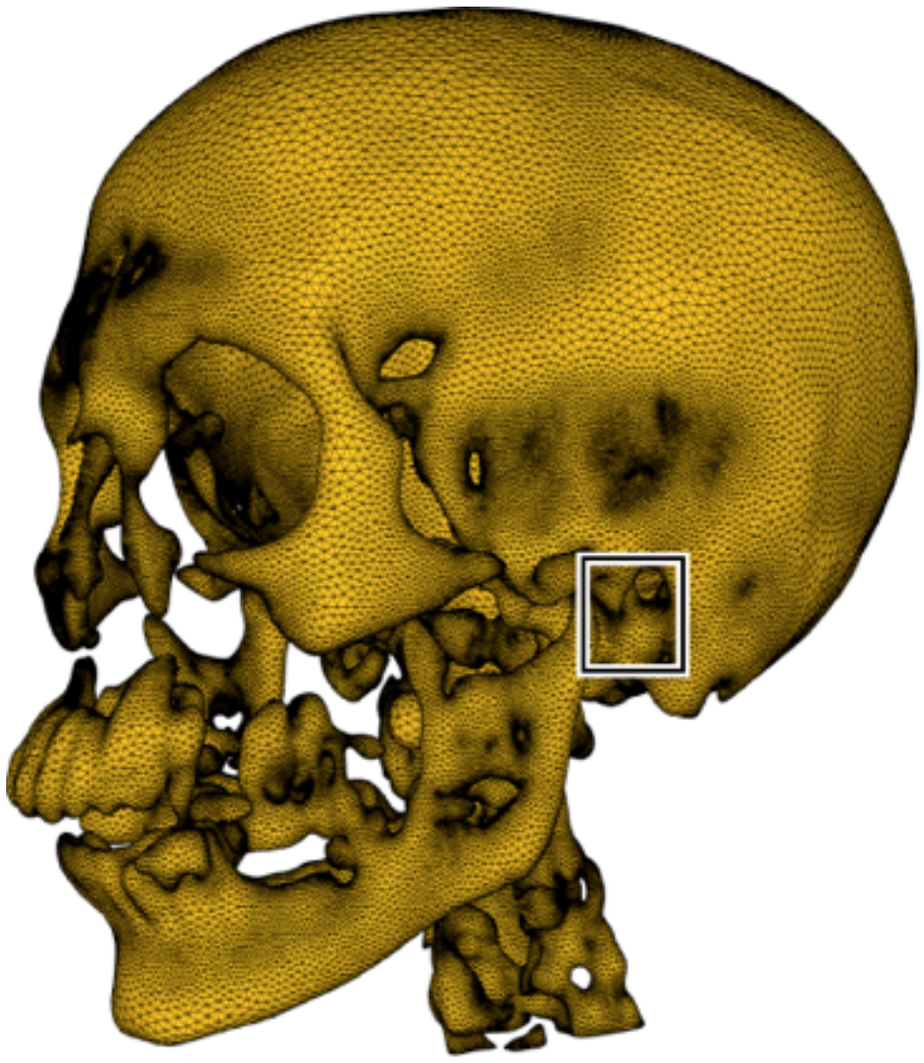
Case 12

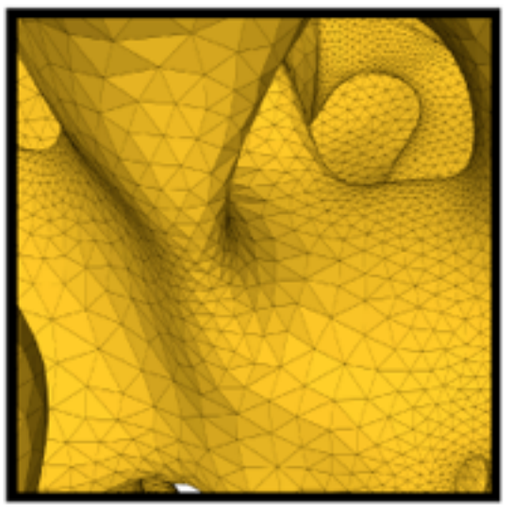


Case 13

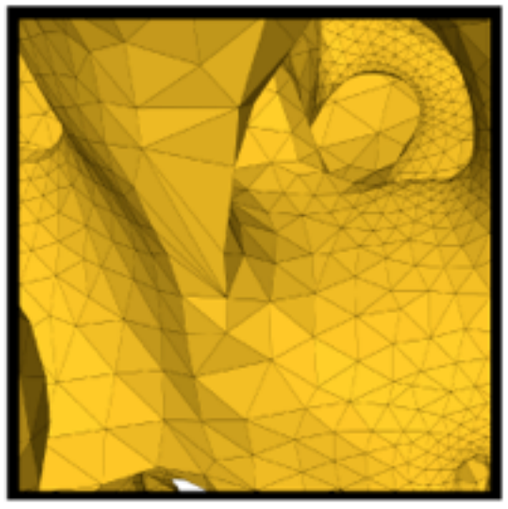


Case 14

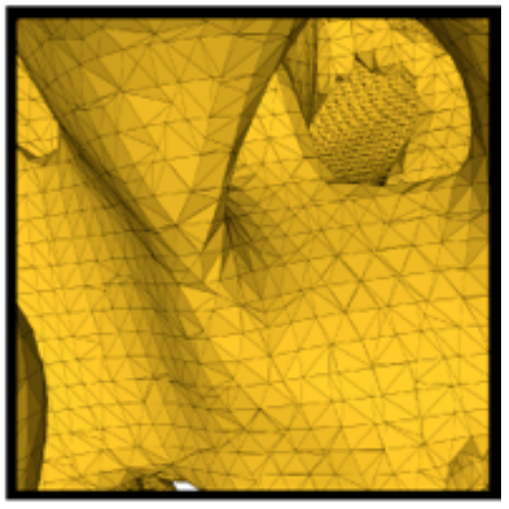




particle system



advancing front



marching cubes

today . . .

3D Graphics

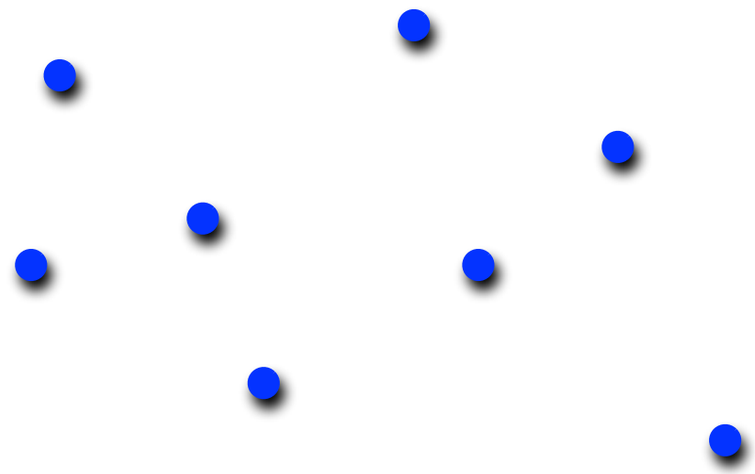
- The key idea for today:
  - Visualization transforms data into **primitives** that are **rendered** using (three-dimensional) **computer graphics** techniques

Graphics Primitives

Graphics Primitives

Points

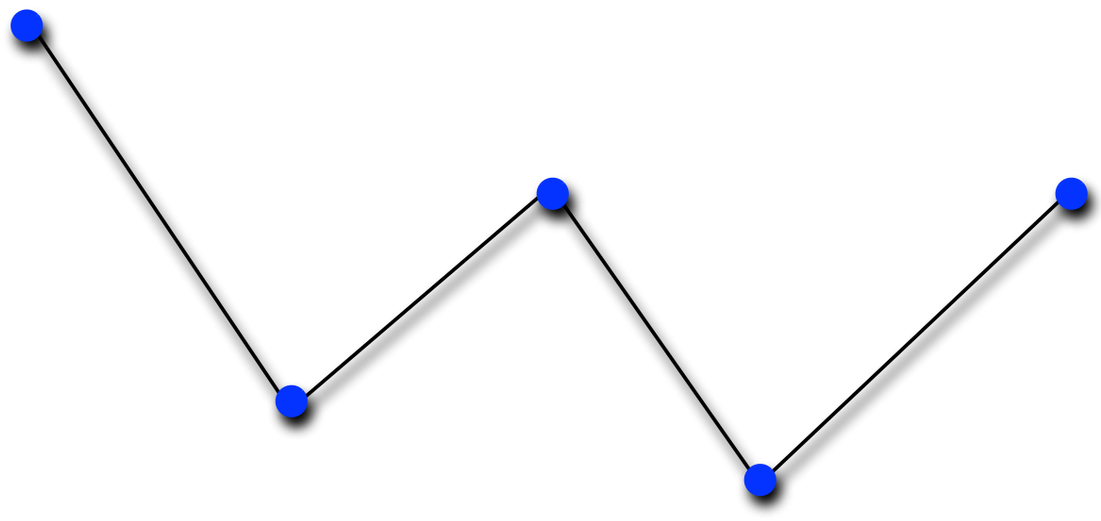
- Positions in space
- 2D, 3D coordinates
- 0-dimensional objects



Graphics Primitives

Lines

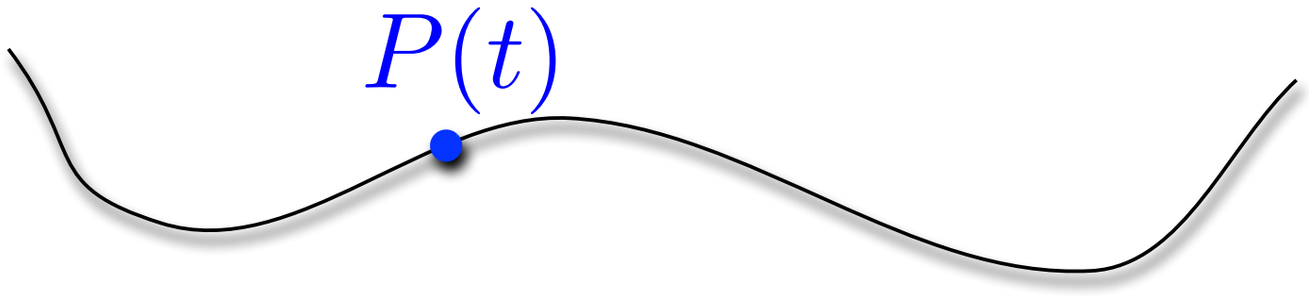
- 1-dimensional objects
- Polygonal description (*"polyline"*)
  - piecewise linear



Graphics Primitives

Lines

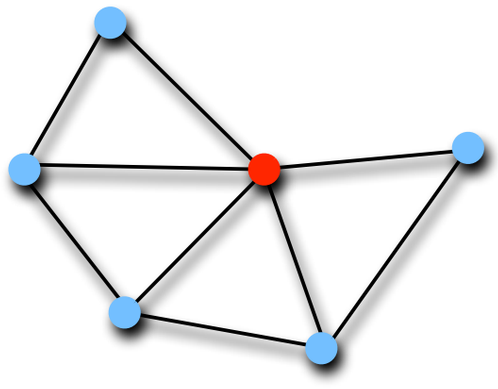
- 1-dimensional objects
- Parametric description
  - $P : t \in I \mapsto P(t) \in \mathbb{R}^n$
  - e.g., splines



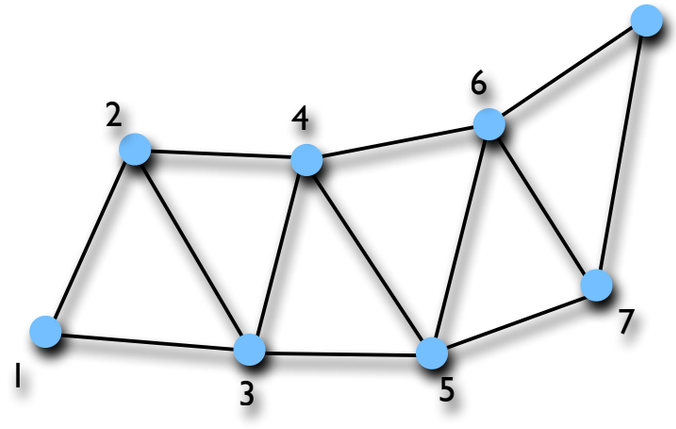
Graphics Primitives

Surfaces

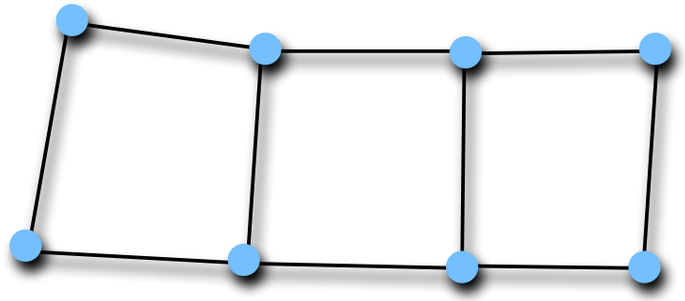
- 2-dimensional objects (in 3D)
- Polygonal description
- Important topologies (special cases)



triangle fan



triangle strip

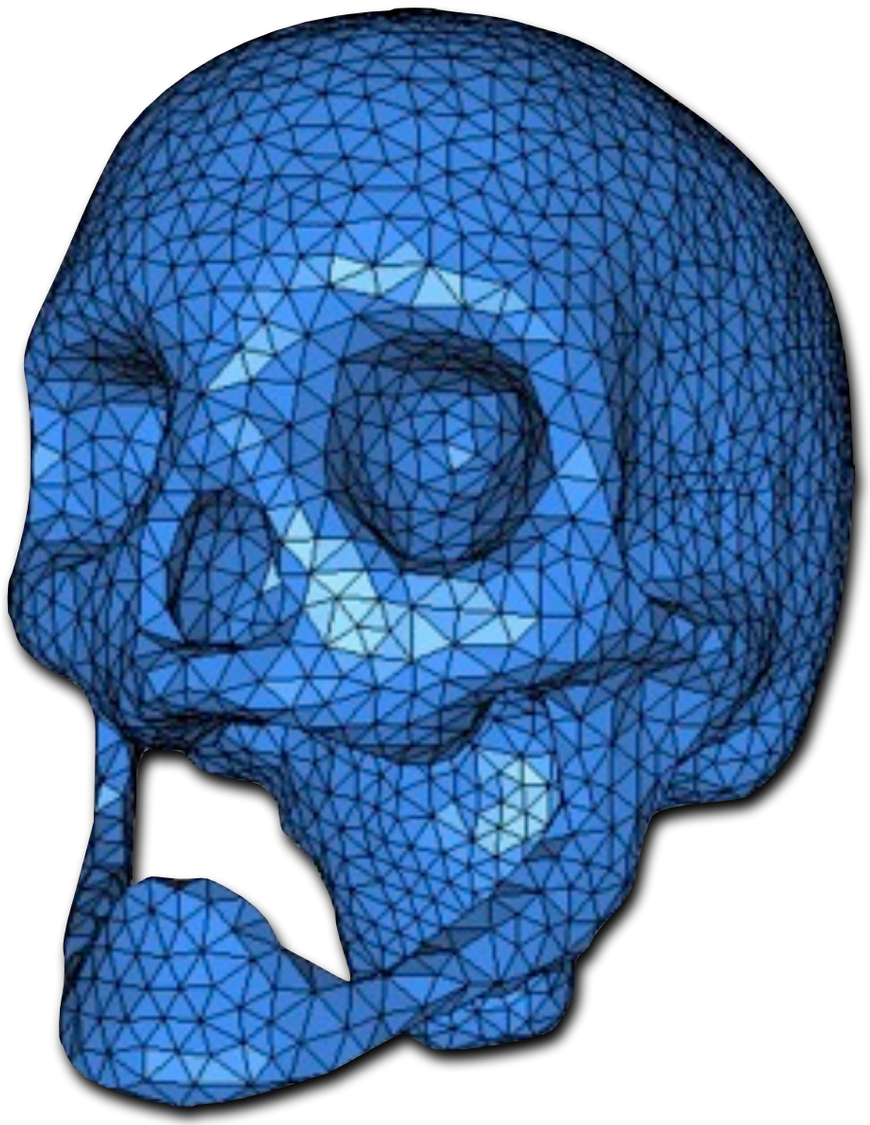


quad strip

Graphics Primitives

Surfaces

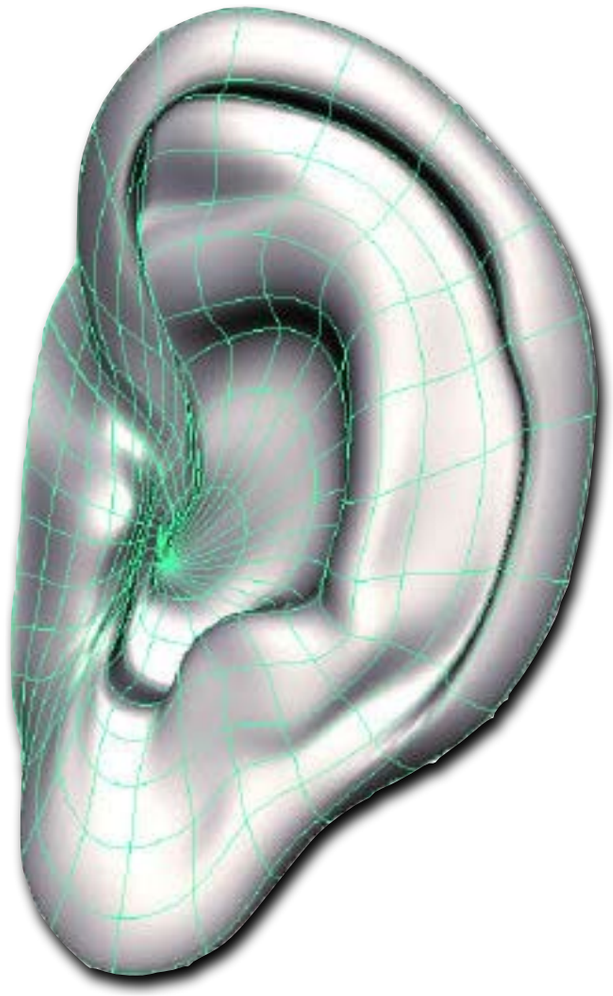
- 2-dimensional objects
- Polygonal description
  - typically: triangle mesh (piecewise linear)



Graphics Primitives

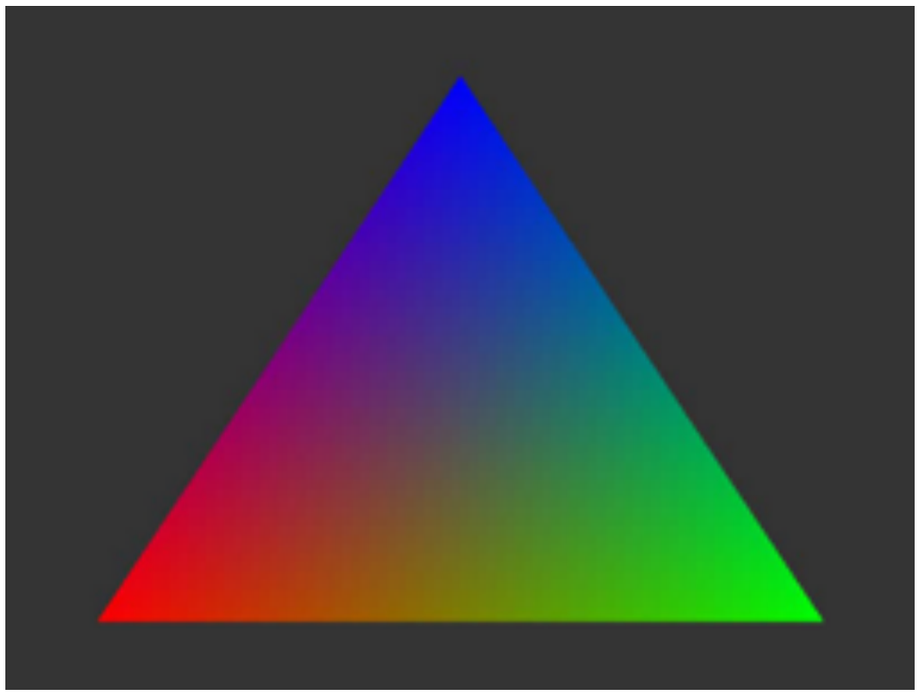
Surfaces

- 2-dimensional objects
- Parametric description
  - $P : (u, v) \in I \times J \mapsto P(u, v) \in \mathbb{R}^n$
  - bi-quadratic, bi-cubic, Bezier, splines, NURBS...

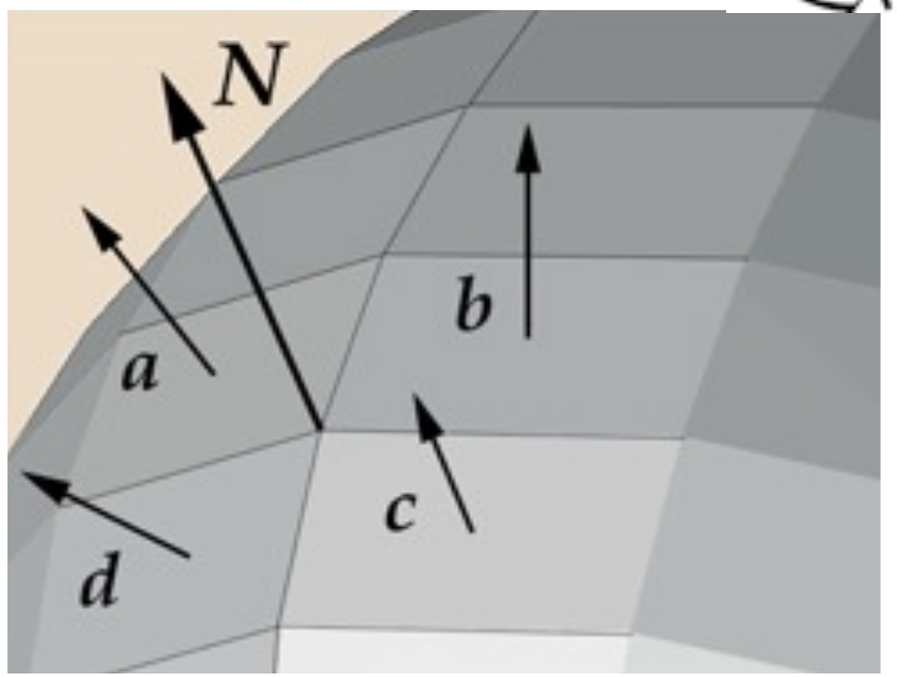


Primitive Attributes

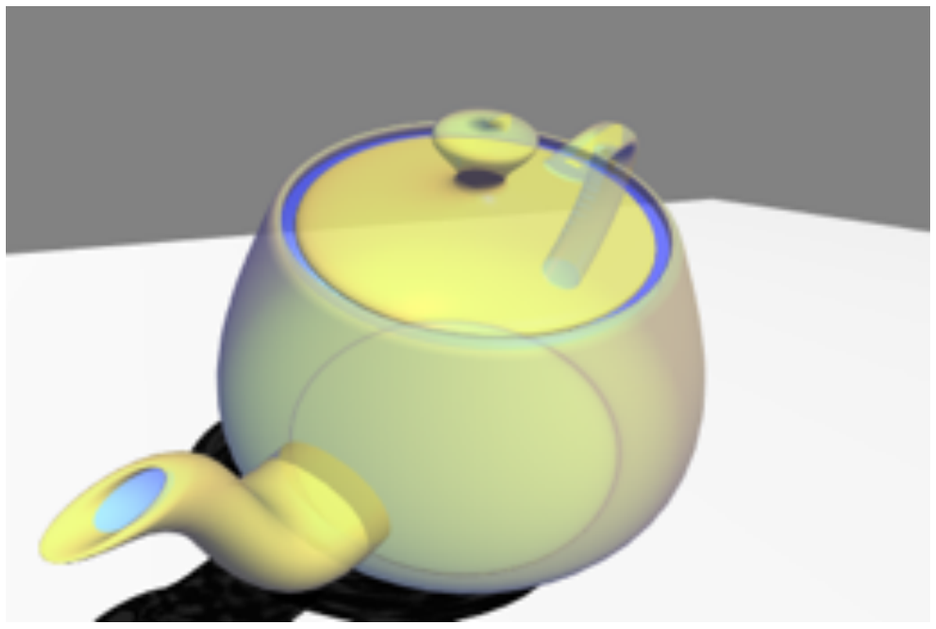
Color

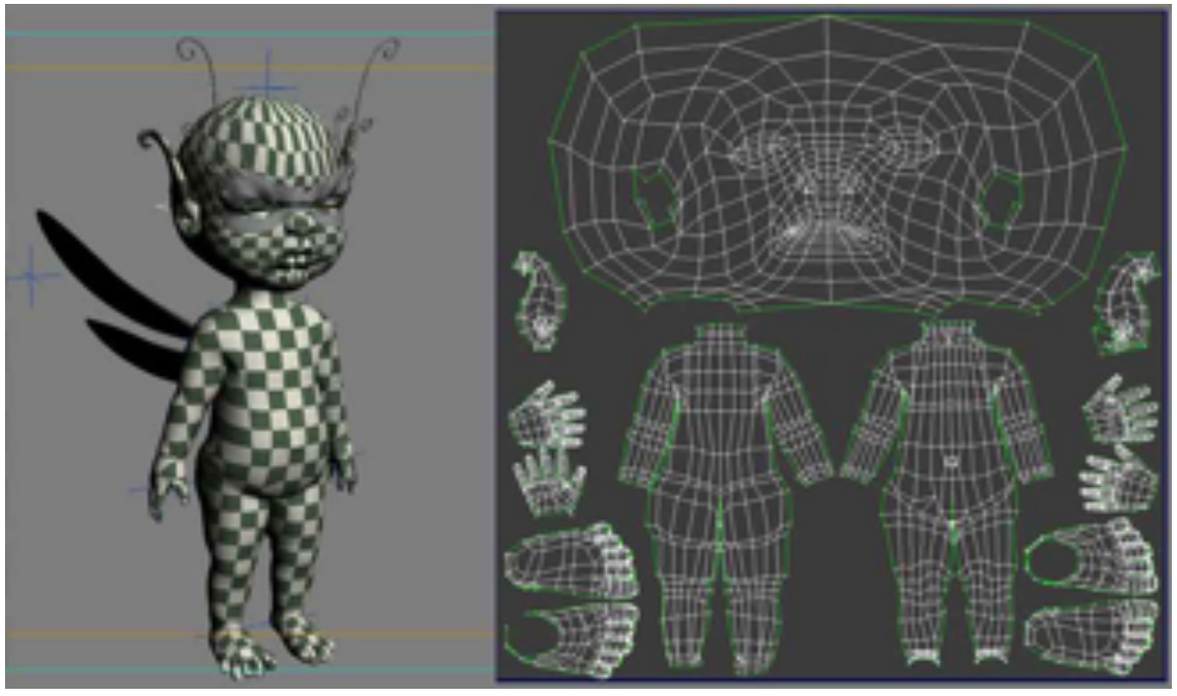


Normal



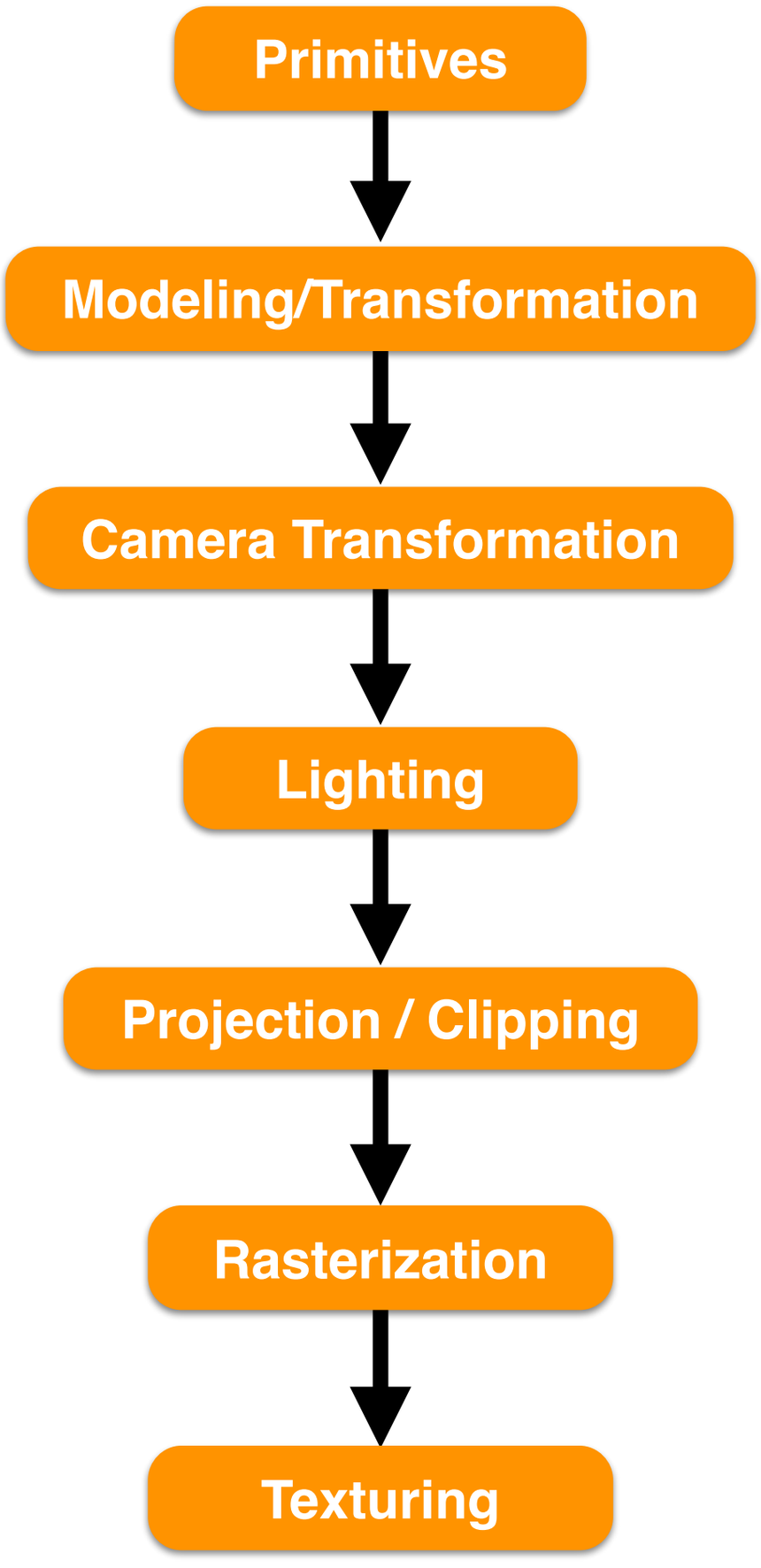
Opacity





Texture coordinates

The Graphics Pipeline



The Pipeline

- Objects are kept and manipulated in 3D until the projection step

- After converting to screen space coordinates, additional coloring and texturing can be applied more rapidly

http://en.wikipedia.org/wiki/Graphics_pipeline

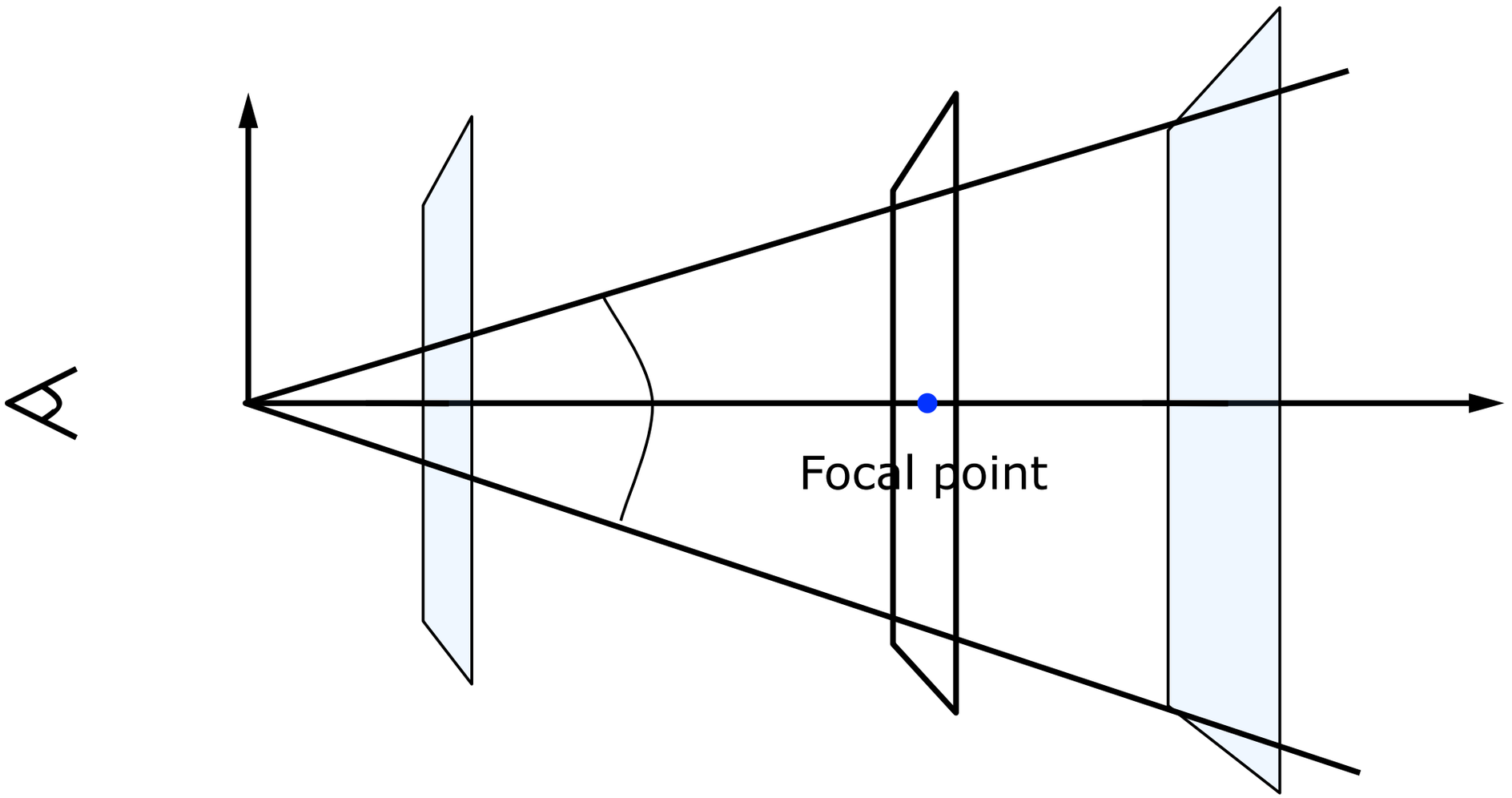
Modeling/Transformation

- Each 3D primitive is assigned locations in the world coordinate system.

- Transformations can be applied globally or locally to individual objects in this space

- Homogeneous coordinate spaces are used so that we can express translations in 3D. Transformations stored as 4x4 matrices

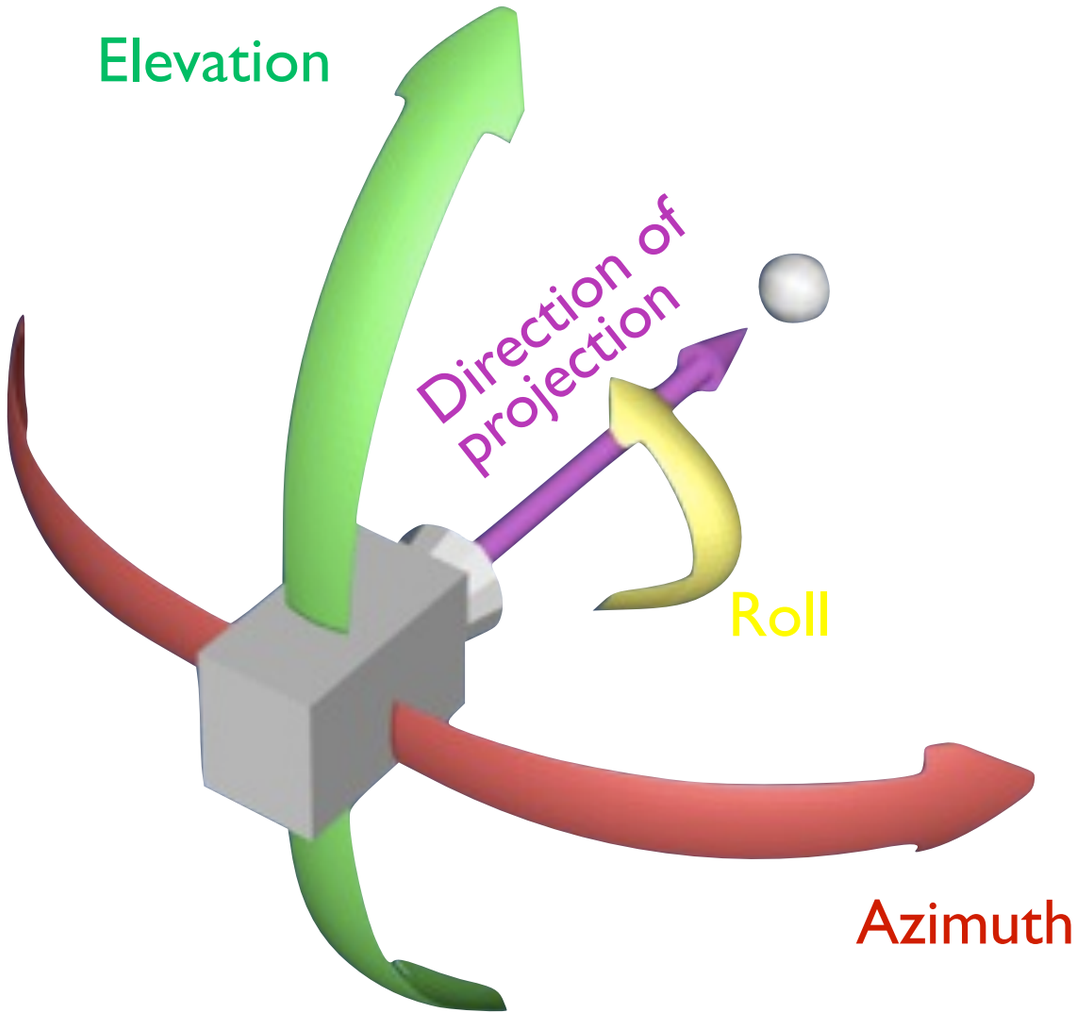
Camera



- Need to specify eye position, eye direction, and eye orientation (or “up” vector)
- This information defines a transformation from world coordinates to camera coordinates

Camera

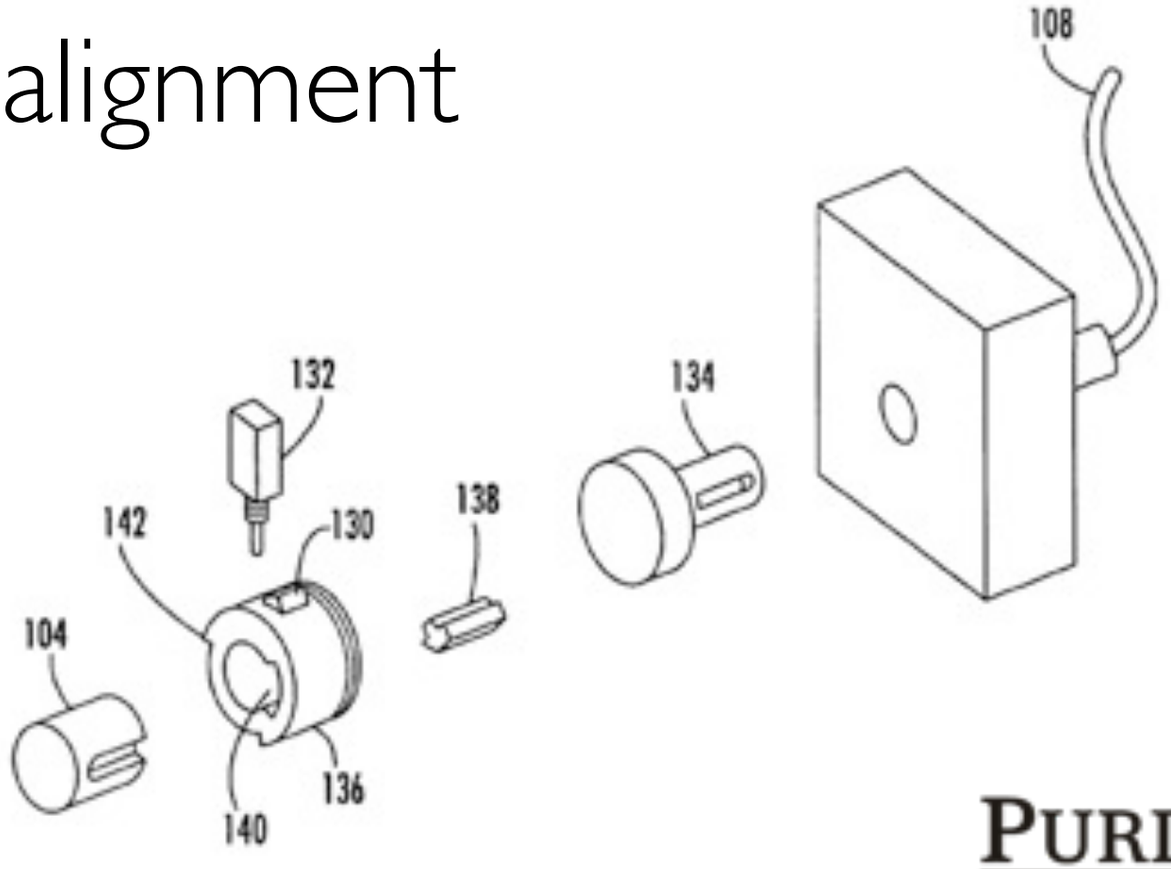
- Degrees of freedom



Projections

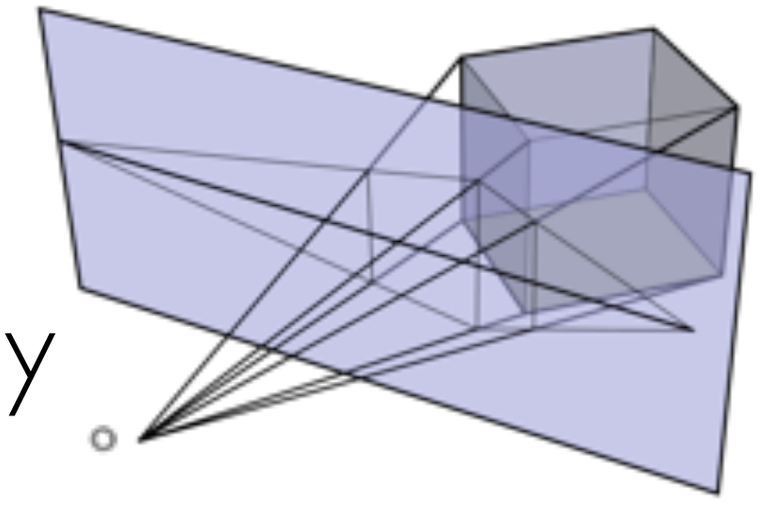
Orthogonal projection

- parallel lines remain parallel
- objects do not change size as they get closer
- good for checking alignment



Projections

Perspective projection



- parallel lines do not necessarily remain parallel
- objects get larger as they get closer
- fly-through realism

Orthographic vs. Perspective Projections

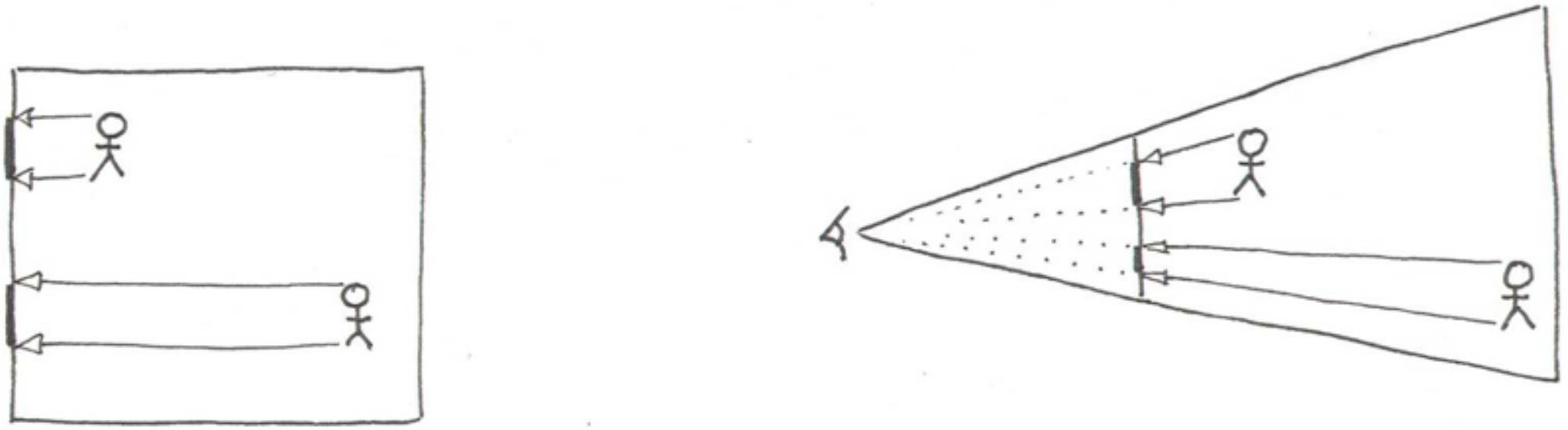
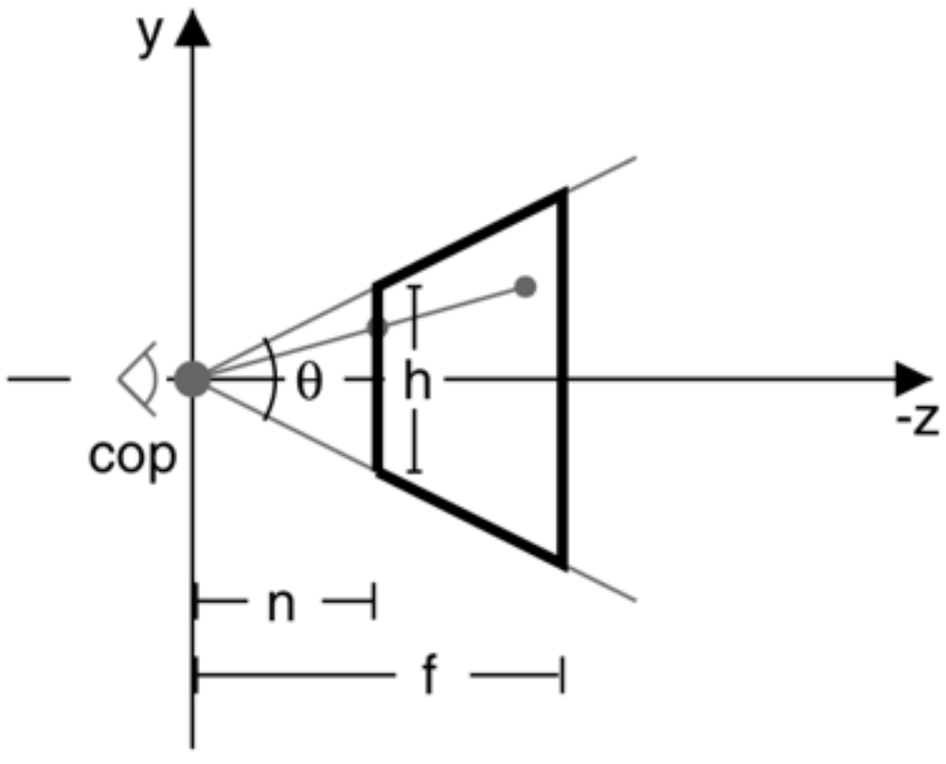


Figure 1: Orthographic and perspective projections: Left, the projectors of an orthographic projection are perpendicular to the viewing plane. Right, the projectors of a perspective projection pass through the center of projection. Foreshortening occurs with perspective.

Dianne Hansford

Perspective Projection

- Maps points in 4D (where it is easier to define the view volume and clipping planes) to positions on the 2D display through multiplication and *homogenization*



$$M_p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & (n+f)/n & -f \\ 0 & 0 & 1/n & 0 \end{bmatrix}.$$

$$M_p\bar{\mathbf{x}} = \begin{bmatrix} x \\ y \\ z(\frac{n+f}{n}) - f \\ z/n \end{bmatrix}$$

Shading

Shading

- Shading reveals the shape of 3D objects through their interaction with light
- Shading creates colors as a function of:
  - surface properties
  - surface normals
  - lights
- Rich subject (*we are only interested in basics here*)
- Surfaces show information, lights show surfaces, shading controls how

Shading

Phong lighting model (1975)

- Specular reflection
- Diffuse reflection
- Ambient reflection

Shading

Phong lighting model (1975)

- Specular reflection
- Diffuse reflection
- Ambient reflection

Light intensity per light source and per color channel

$$I = k_s I_s$$

Shading

Phong lighting model (1975)

- Specular reflection
- Diffuse reflection
- Ambient reflection

Light intensity per light source and per color channel

$$I = k_s I_s$$

relative contributions
(material specific)

Shading

Phong lighting model (1975)

- Specular reflection
- Diffuse reflection
- Ambient reflection

Light intensity per light source and per color channel

$$I = k_s I_s + k_d I_d$$

relative contributions (material specific)

Shading

Phong lighting model (1975)

- Specular reflection
- Diffuse reflection
- Ambient reflection

Light intensity per light source and per color channel

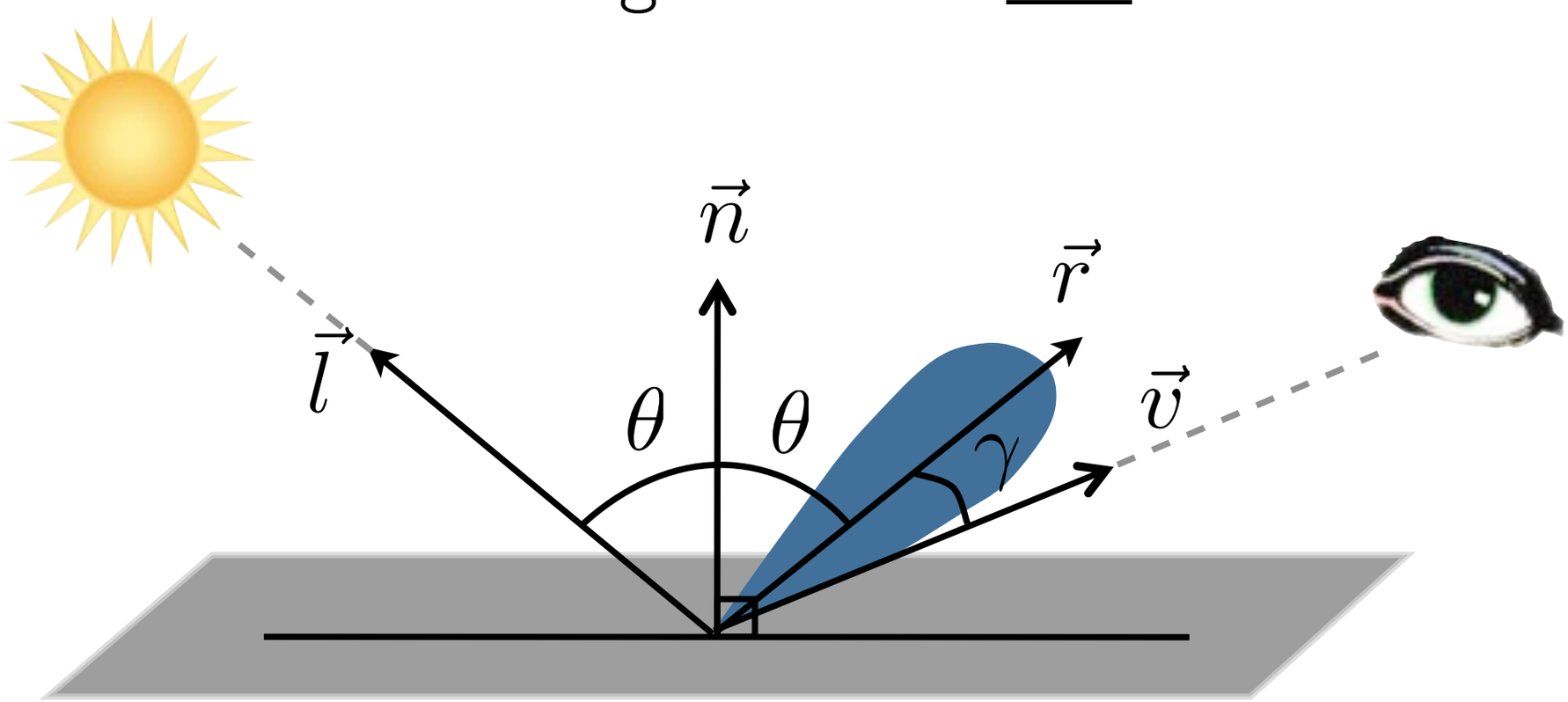
$$I = k_s I_s + k_d I_d + k_a I_a$$

relative contributions
(material specific)

Shading

Phong lighting model: Specular Reflection

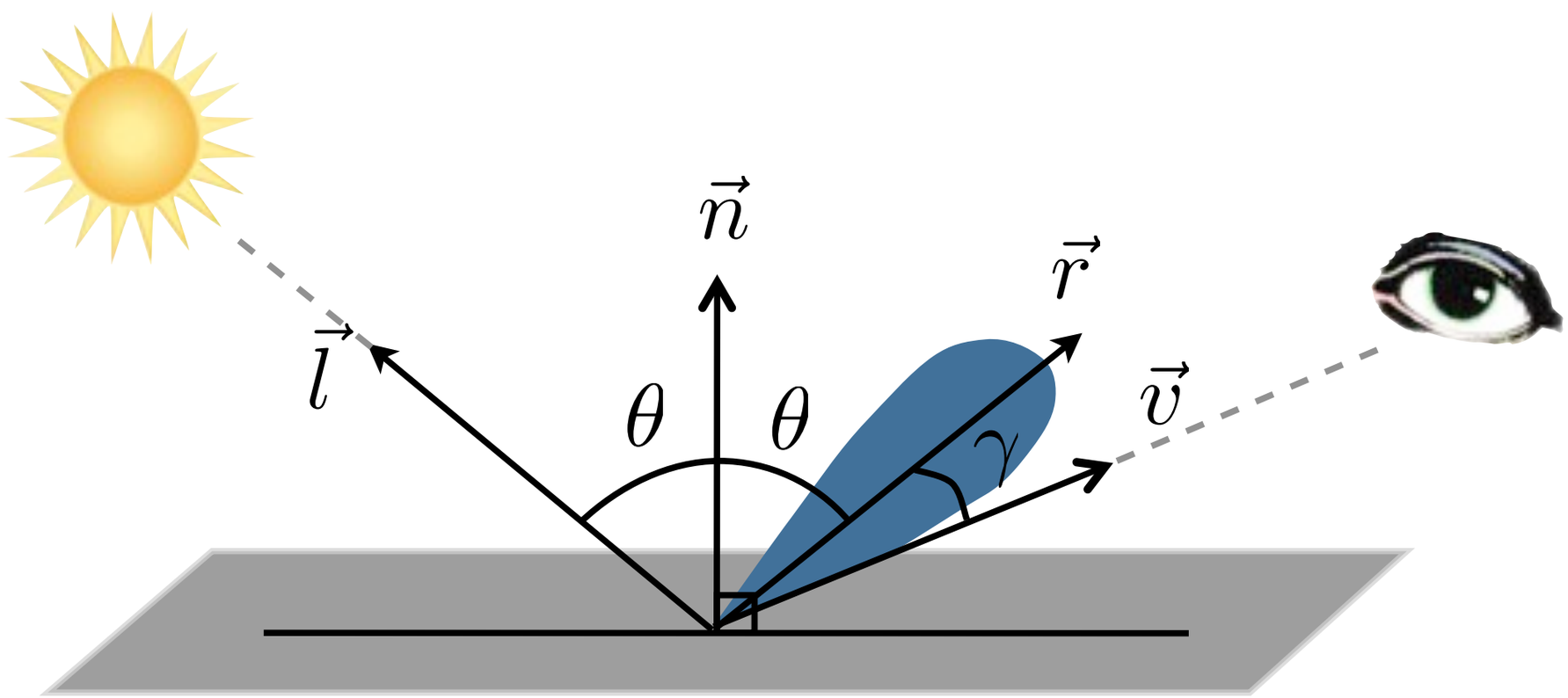
- mirror-like surfaces
- Specular reflection depends on position of the observer relative to light source and surface normal



Shading

Phong lighting model: Specular Reflection

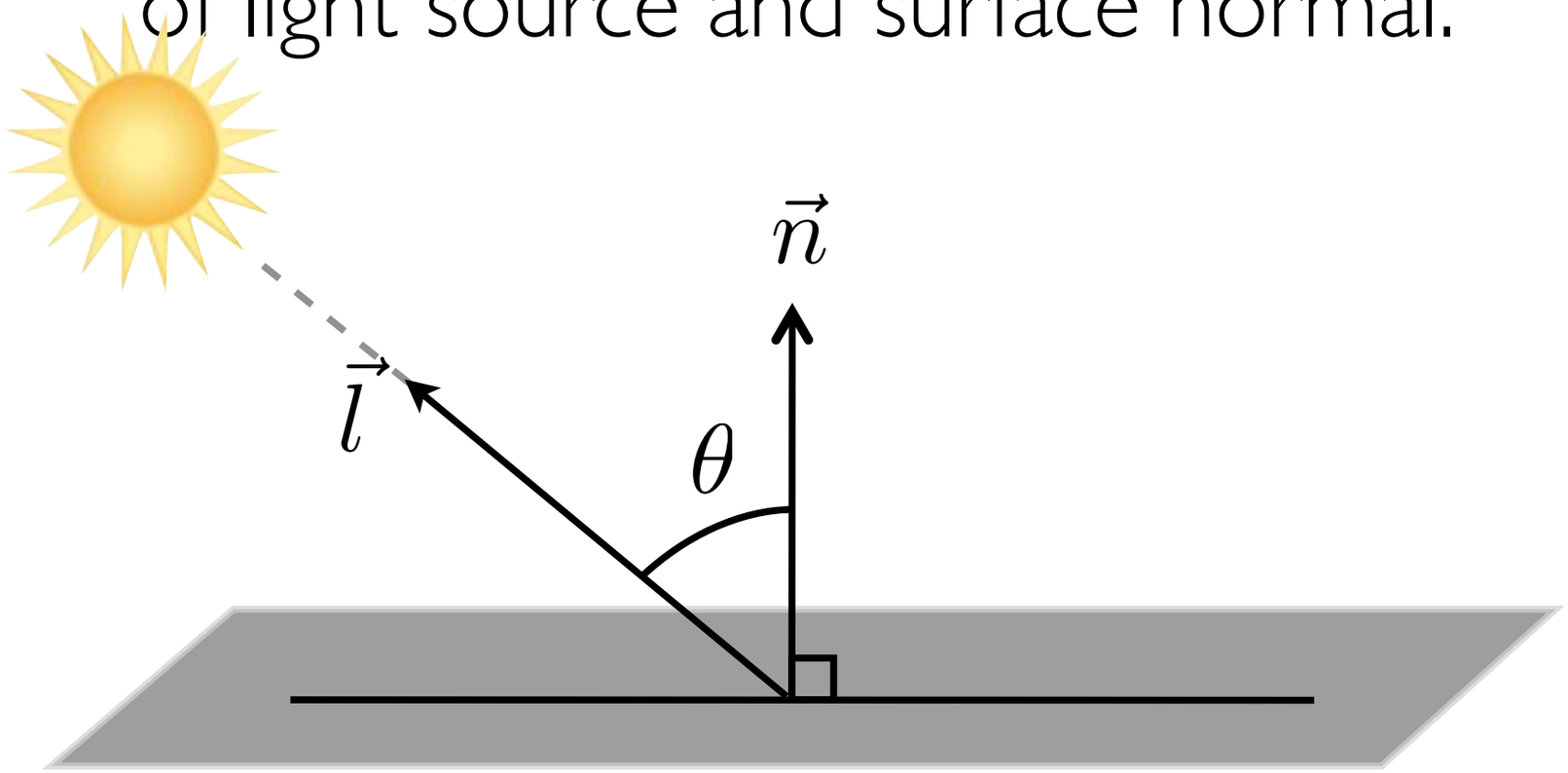
$$I_s = I_i \cos^n \gamma = I_i \cos^n < \vec{r}, \vec{v} >$$

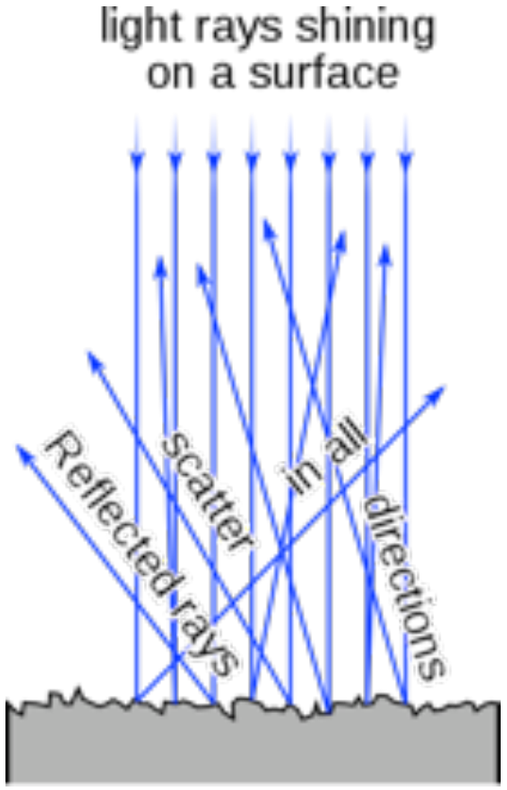


Shading

Phong lighting model: Diffuse Reflection

- Non-shiny surfaces
- Diffuse reflection depends only on relative position of light source and surface normal.

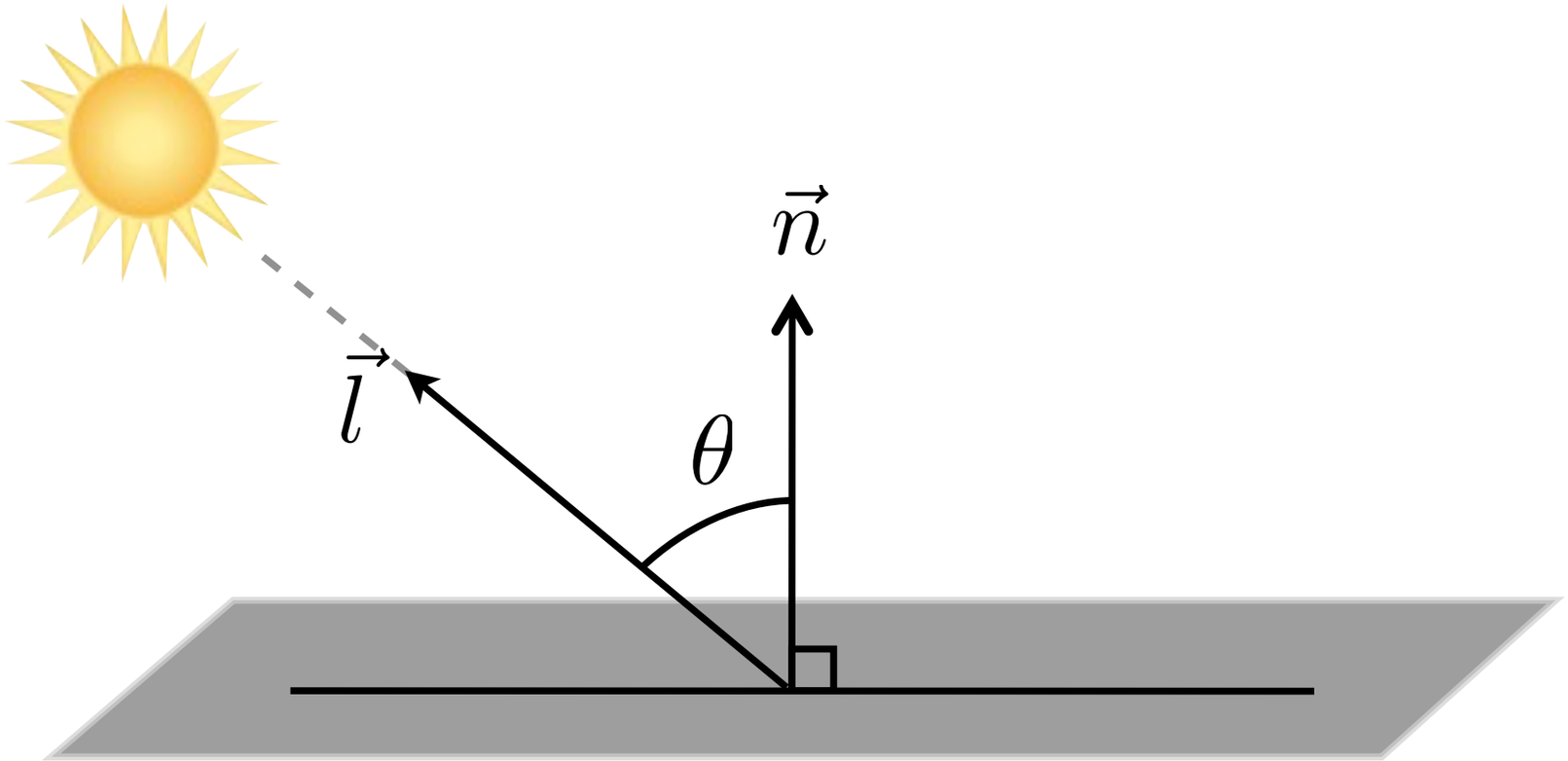


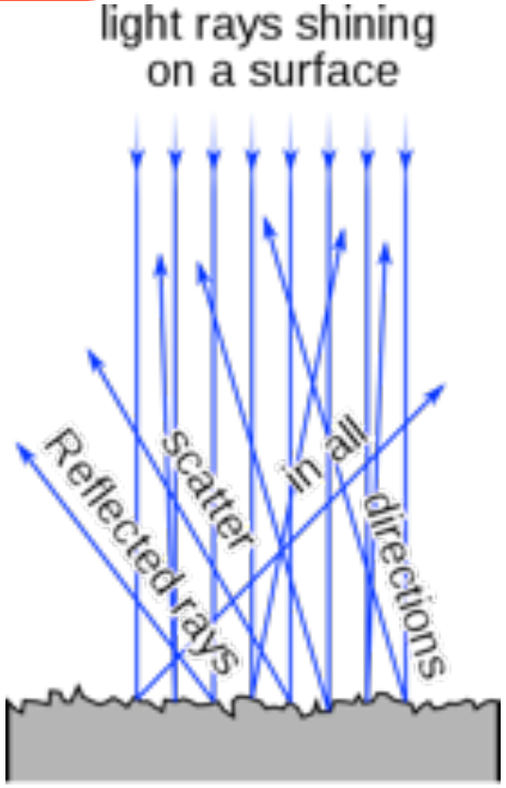


Shading

Phong lighting model: Diffuse Reflection

$$I_d = I_i \cos\theta = I_i \ (\vec{l} \cdot \vec{n})$$





Shading

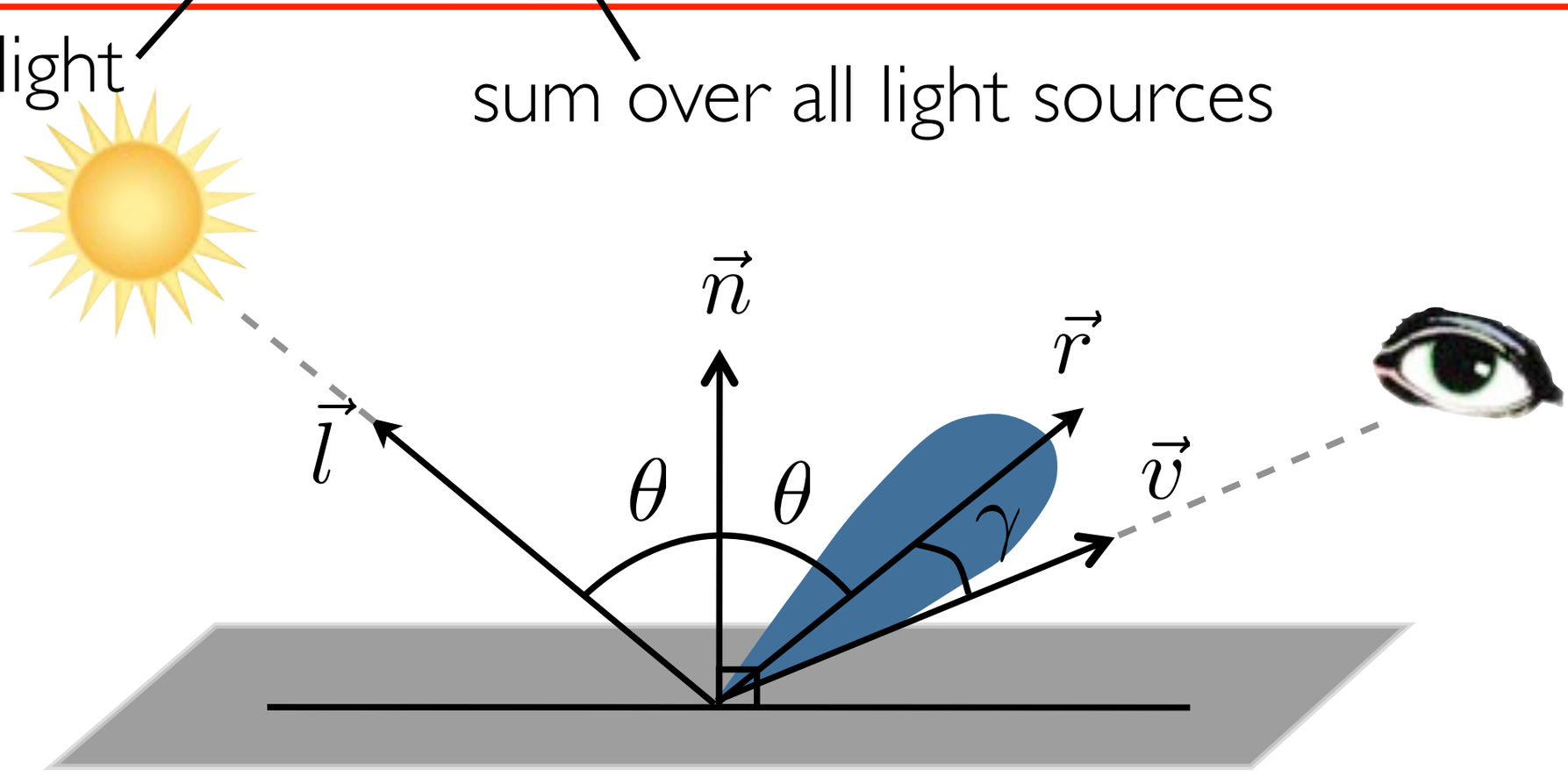
Phong lighting model

per color channel

$$I = k_a I_a + \sum_{i=1}^{N} I_i \left(k_d \cos(\theta) + k_s \cos^n(\gamma) \right)$$

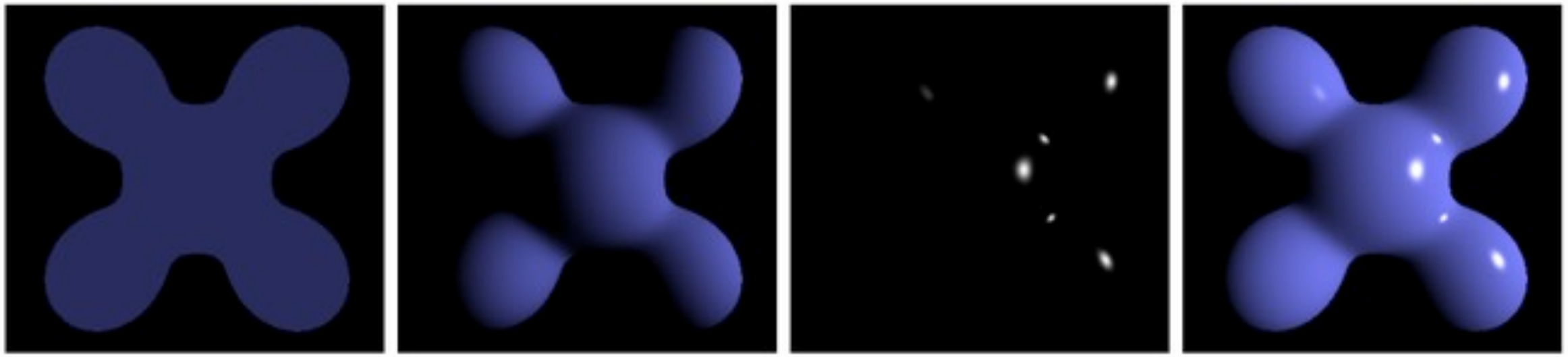
ambient light

sum over all light sources



Shading

Phong lighting model



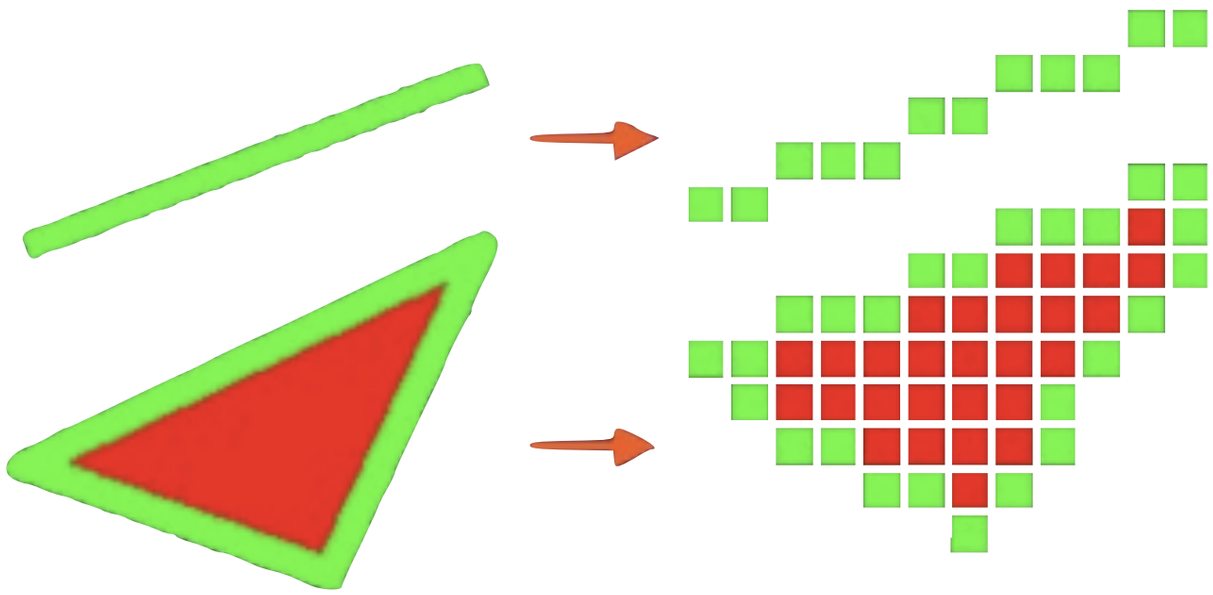
Ambient + Diffuse + Specular = Phong Reflection

http://en.wikipedia.org/wiki/Phong_shading

Rasterization

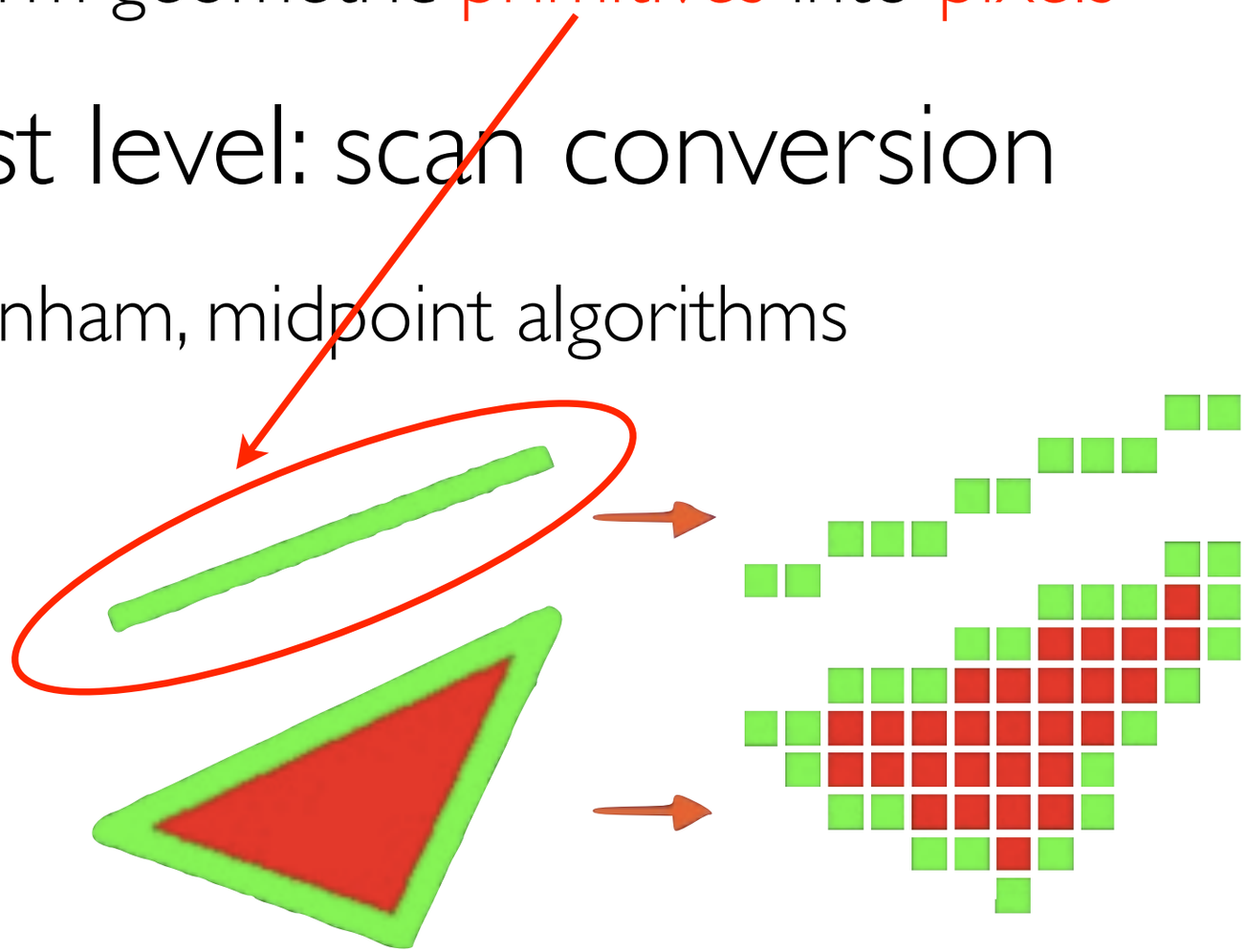
Rasterization

- Putting shaded polygons on screen
  - Transform geometric primitives into pixels
- Lowest level: scan conversion
  - Bresenham, midpoint algorithms



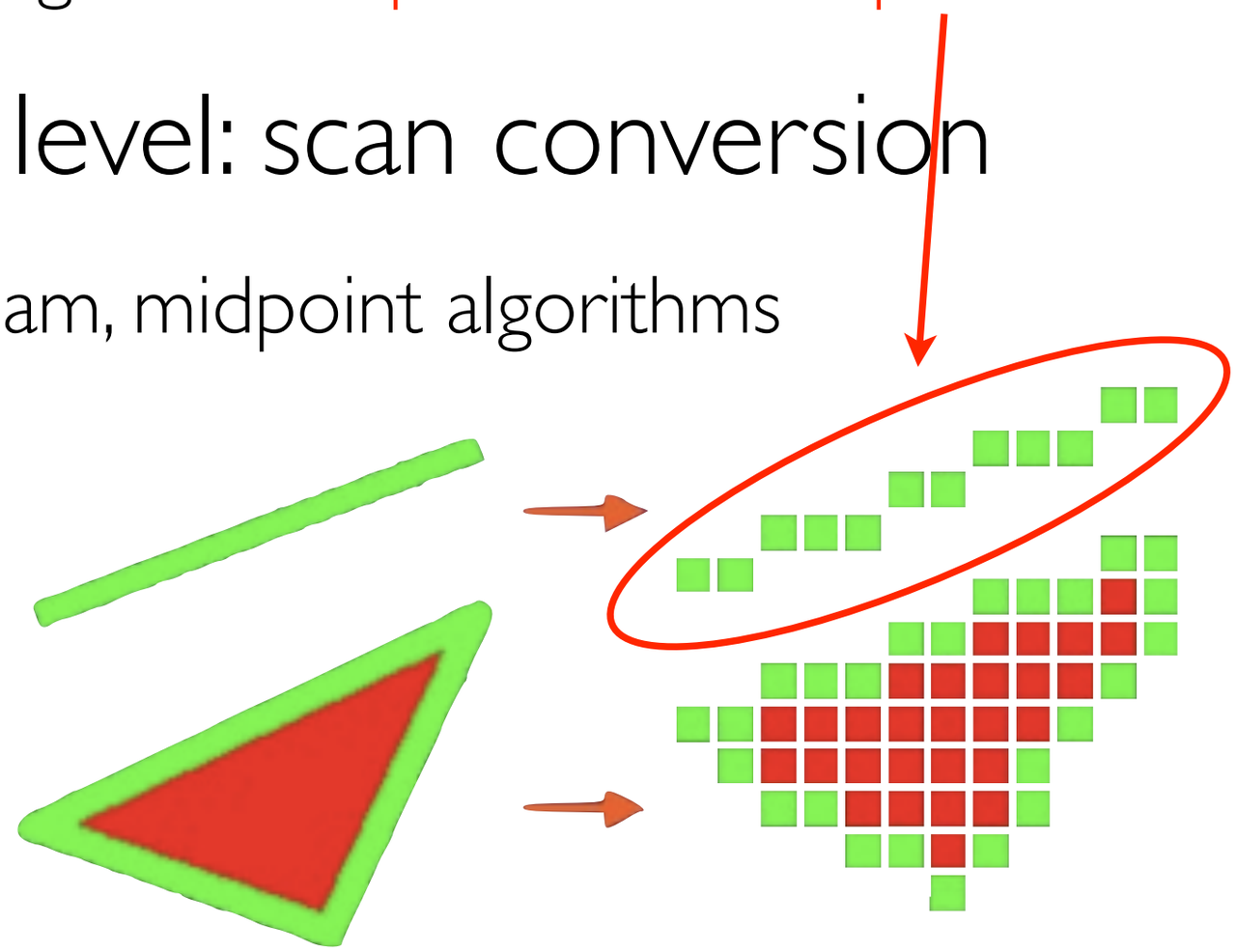
Rasterization

- Putting shaded polygons on screen
  - Transform geometric primitives into pixels
- Lowest level: scan conversion
  - Bresenham, midpoint algorithms



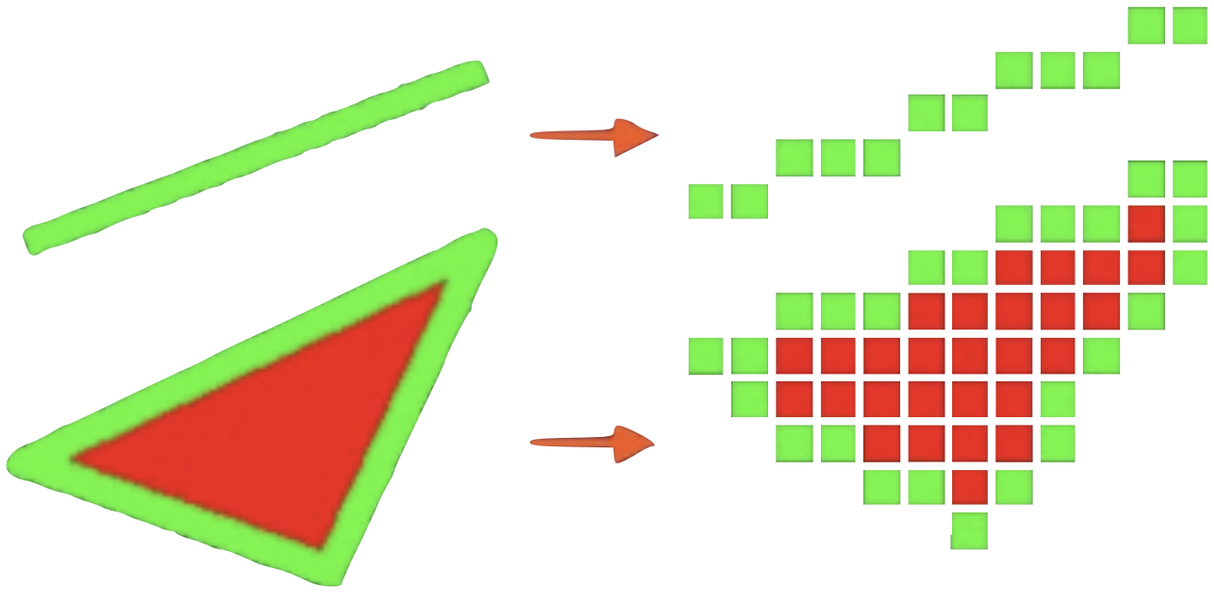
Rasterization

- Putting shaded polygons on screen
  - Transform geometric primitives into pixels
- Lowest level: scan conversion
  - Bresenham, midpoint algorithms



Rasterization

- Putting shaded polygons on screen
  - Transform geometric primitives into pixels
- OpenGL (graphics library) takes care of such operations (under the hood in VTK)



Hidden Surface Removal

- Process of determining which primitives are not visible from a given viewpoint and then removing them from the display pipeline

Fundamental Algorithms

Back-face culling

- Determine whether a polygon is visible (and should be rendered)
- Check polygon normal against camera viewing direction: remove polygons that are pointing away!

Fundamental Algorithms

Back-face culling

- Determine whether a polygon is visible (and should be rendered)
- Check polygon normal against camera viewing direction: remove polygons that are pointing away!

If some objects do not show up/are black in your visualization: check your normals!

Fundamental Algorithms

Z-buffer algorithm

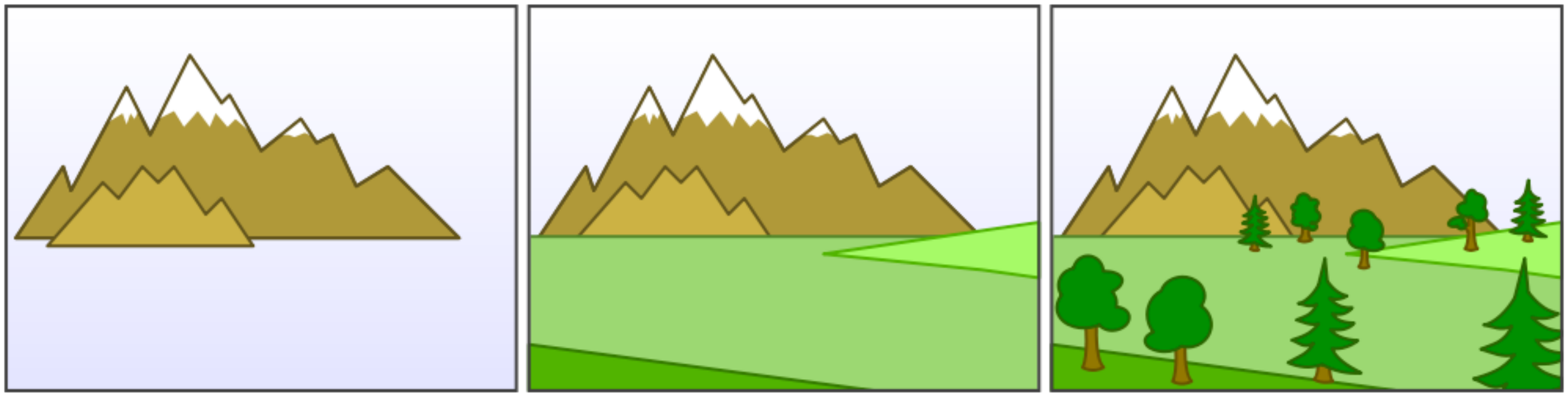
- Maintain a z-buffer of same resolution as frame buffer
- During scan conversion compare z-coord of pixel to be stored with z-coord of current pixel in Z-buffer
- If new pixel is closer, store value in frame buffer and new z-coord in Z-buffer

Fundamental Algorithms

Z-buffer algorithm

Painter’s Algorithm

- Does not require rasterization, instead sorts primitives from back to front and draws them in order



http://en.wikipedia.org/wiki/Painter%27s_algorithm

Fundamental Algorithms

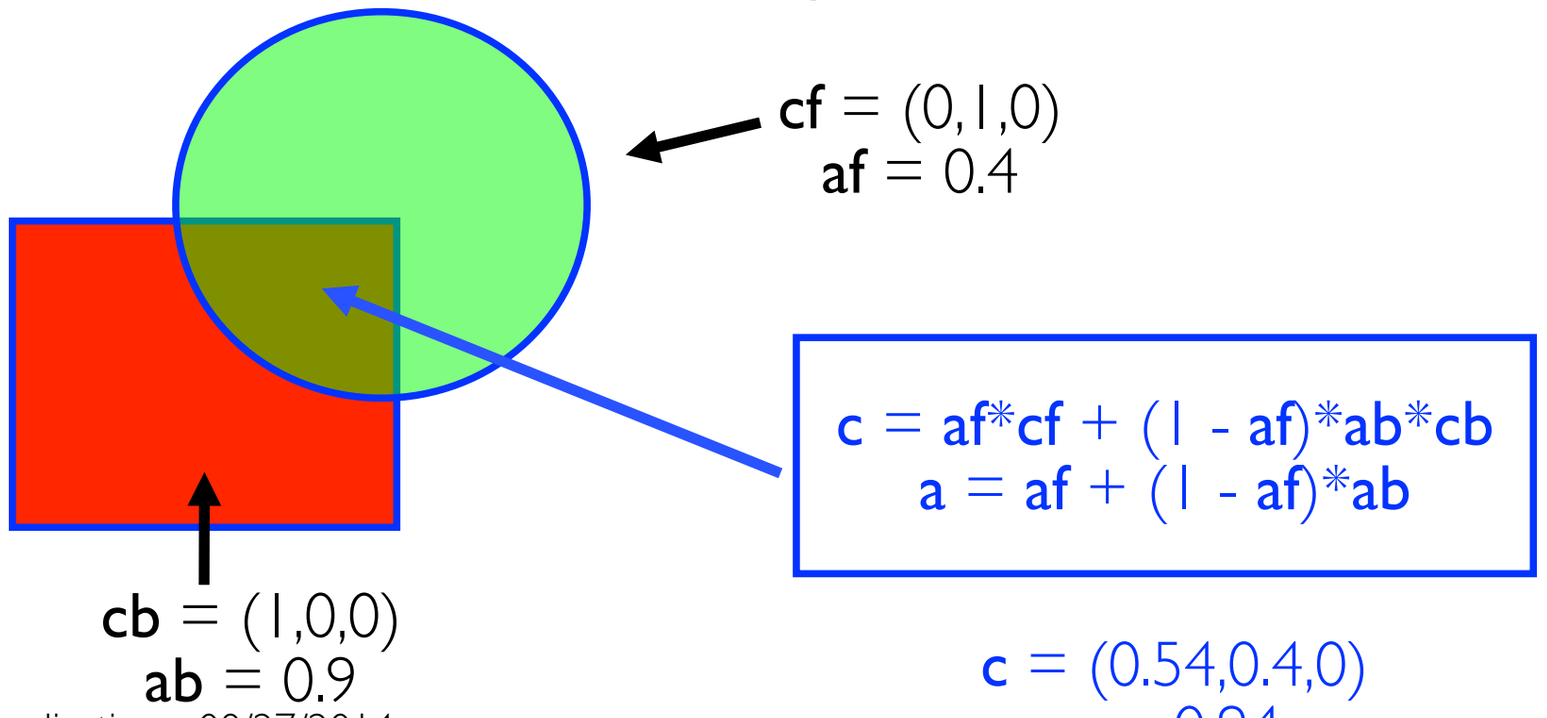
Alpha blending / compositing

- Approximate visual appearance of semi-transparent object in front of another object
- Implemented with OVER operator

Fundamental Algorithms

Alpha blending / compositing

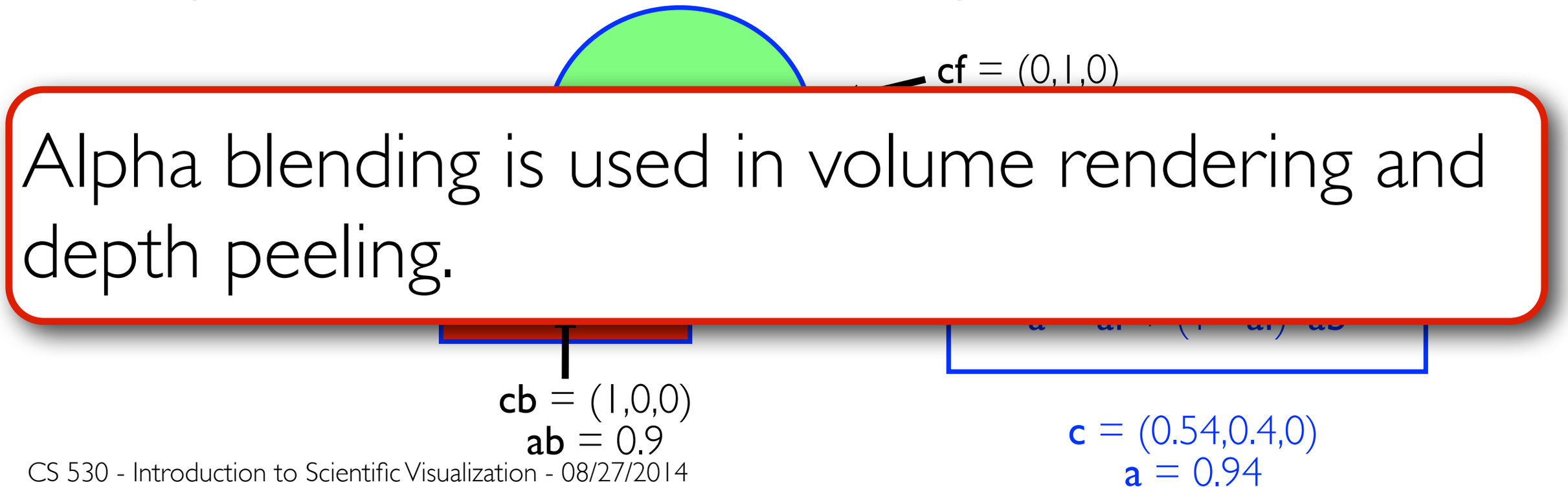
- Approximate visual appearance of semi-transparent object in front of another object
- Implemented with OVER operator



Fundamental Algorithms

Alpha blending / compositing

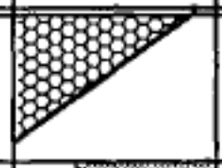
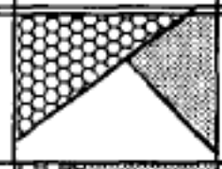
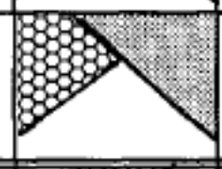
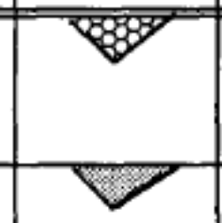
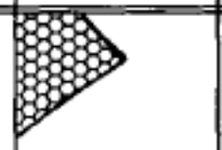
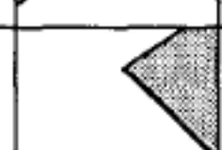
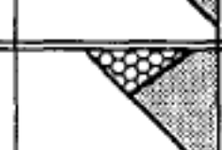
- Approximate visual appearance of semi-transparent object in front of another object
- Implemented with OVER operator



Alpha blending is used in volume rendering and depth peeling.

12 Operators In Total

- $c_P = F_A c_A + F_B c_B$
- Various F functions dictate how to blend
- Always assumes color-associated values (alway is premultiplied into the color)

operation	quadruple	diagram	F_A	F_B
clear	(0,0,0,0)		0	0
A	(0,A,0,A)		1	0
B	(0,0,B,B)		0	1
A **over** *B*	(0,A,B,A)		1	$1-\alpha_A$
B **over** *A*	(0,A,B,B)		$1-\alpha_B$	1
A **in** *B*	(0,0,0,A)		α_B	0
B **in** *A*	(0,0,0,B)		0	α_A
A **out** *B*	(0,A,0,0)		$1-\alpha_B$	0
B **out** *A*	(0,0,B,0)		0	$1-\alpha_A$
A **atop** *B*	(0,0,B,A)		α_B	$1-\alpha_A$
B **atop** *A*	(0,A,0,B)		$1-\alpha_B$	α_A
A **xor** *B*	(0,A,B,0)		$1-\alpha_B$	$1-\alpha_A$

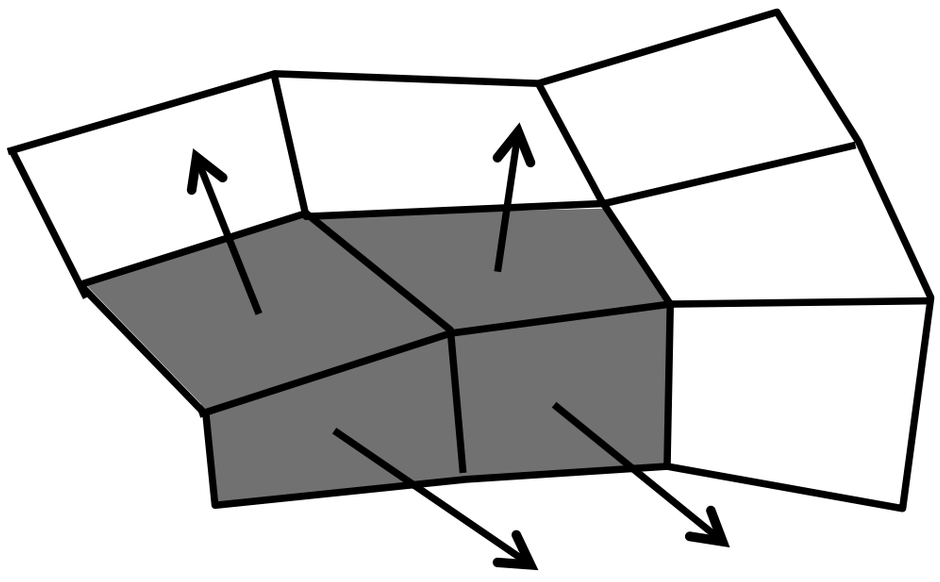
Porter & Duff, 1984

Shading and Illumination

Back to Shading

Flat shading

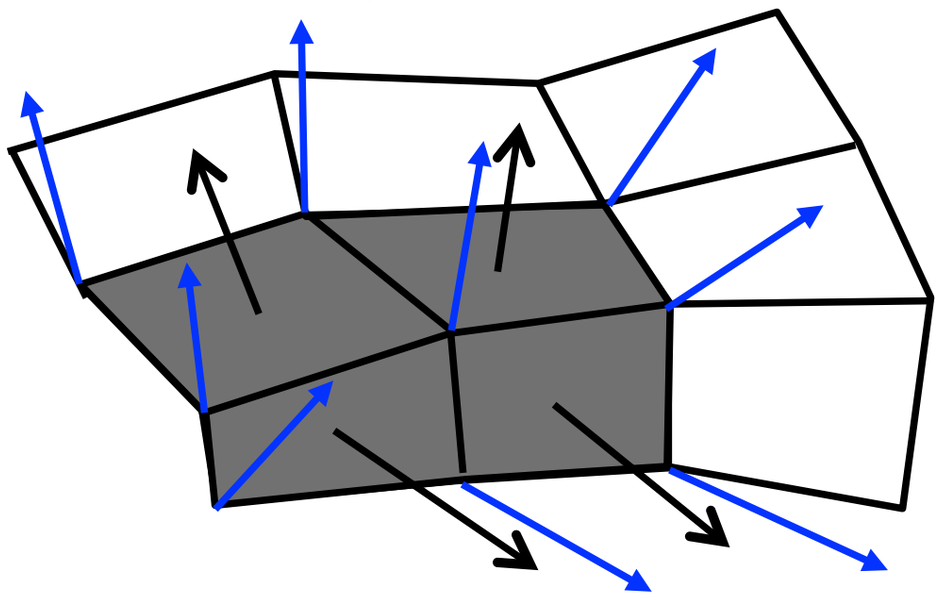
(Color constant per polygon)

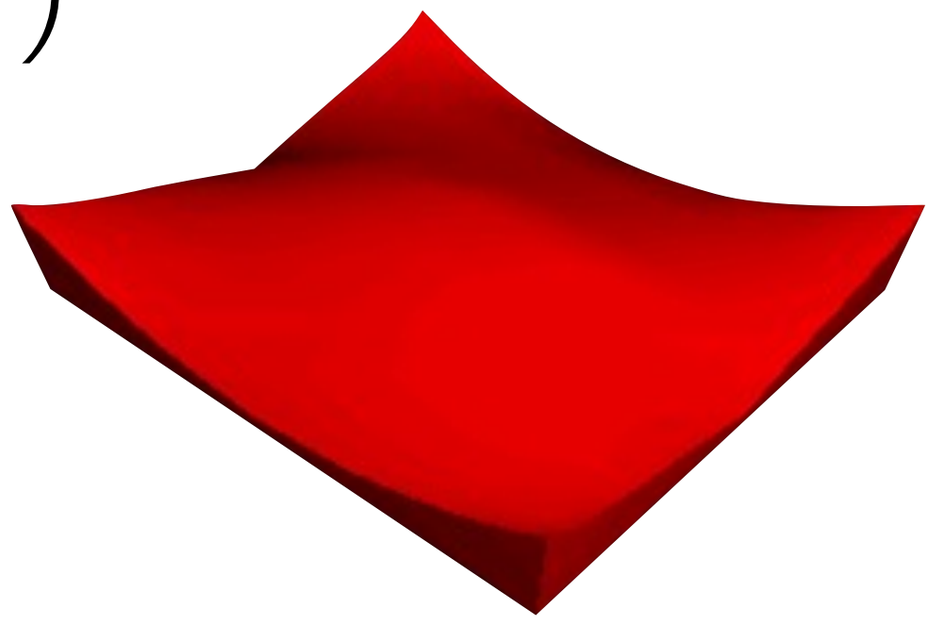


Back to Shading

Gouraud shading (1971)

- Shade vertices first, then interpolate* colors



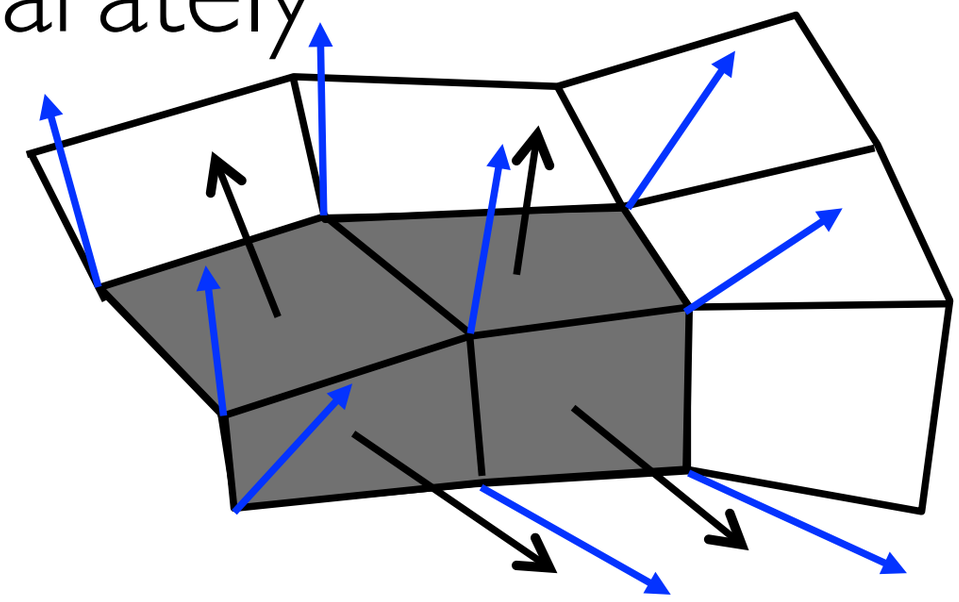


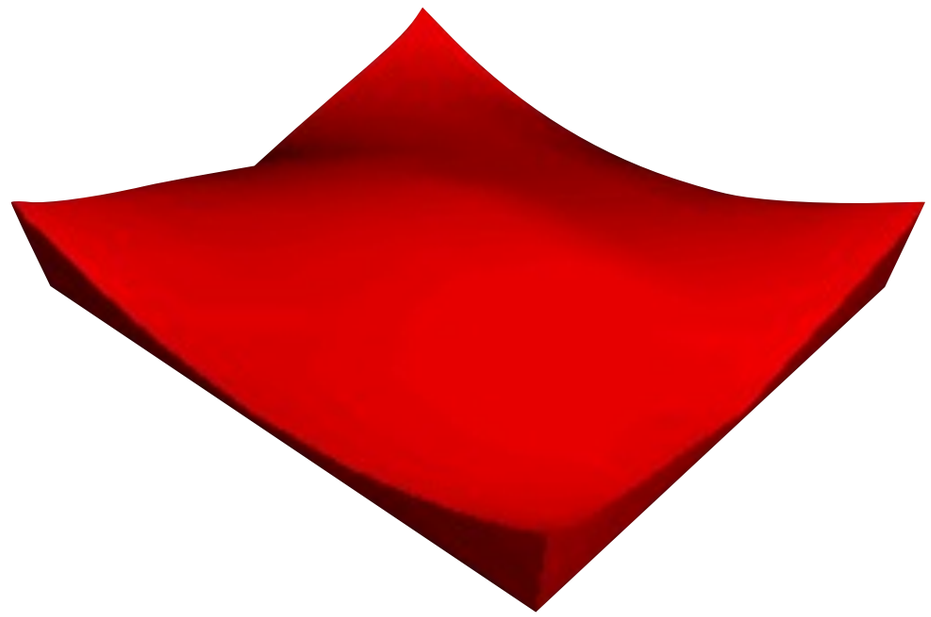
(*) More on interpolation in coming weeks...

Back to Shading

Phong shading

- Interpolate normals and shade every pixel separately





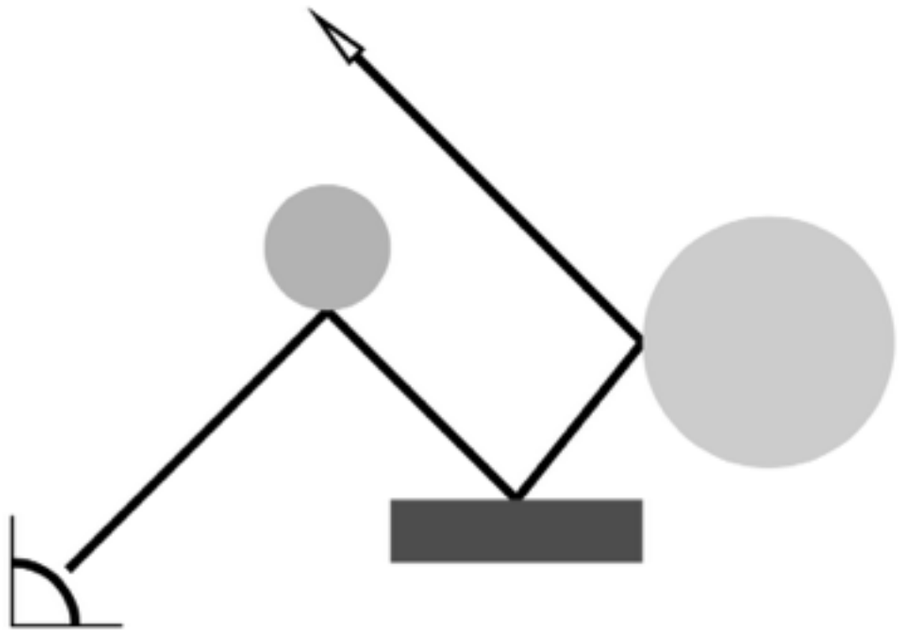
Note: Phong lighting model ≠ Phong shading

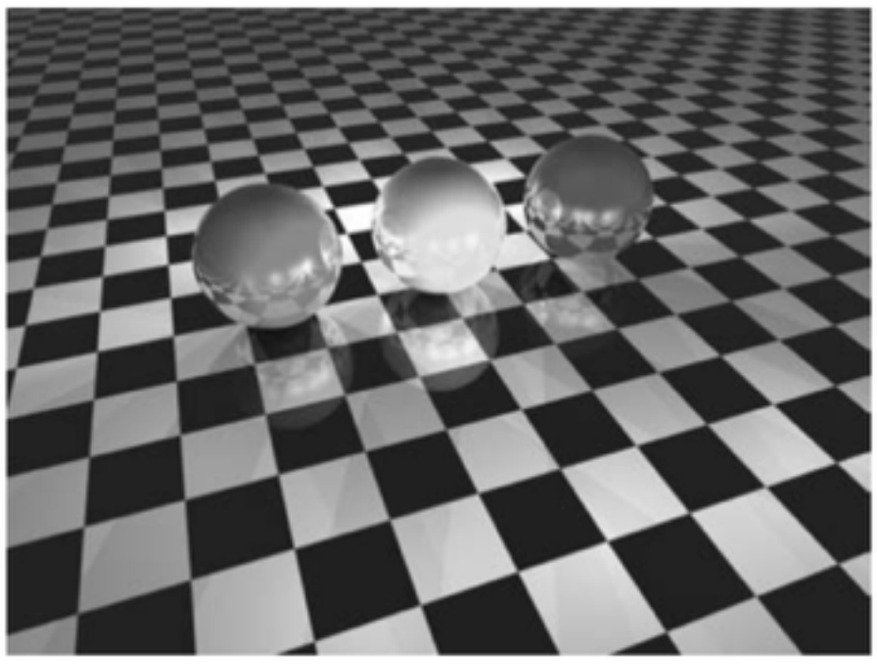
Global Illumination

- Phong shading is quick, but only has single interactions and doesn't really model light

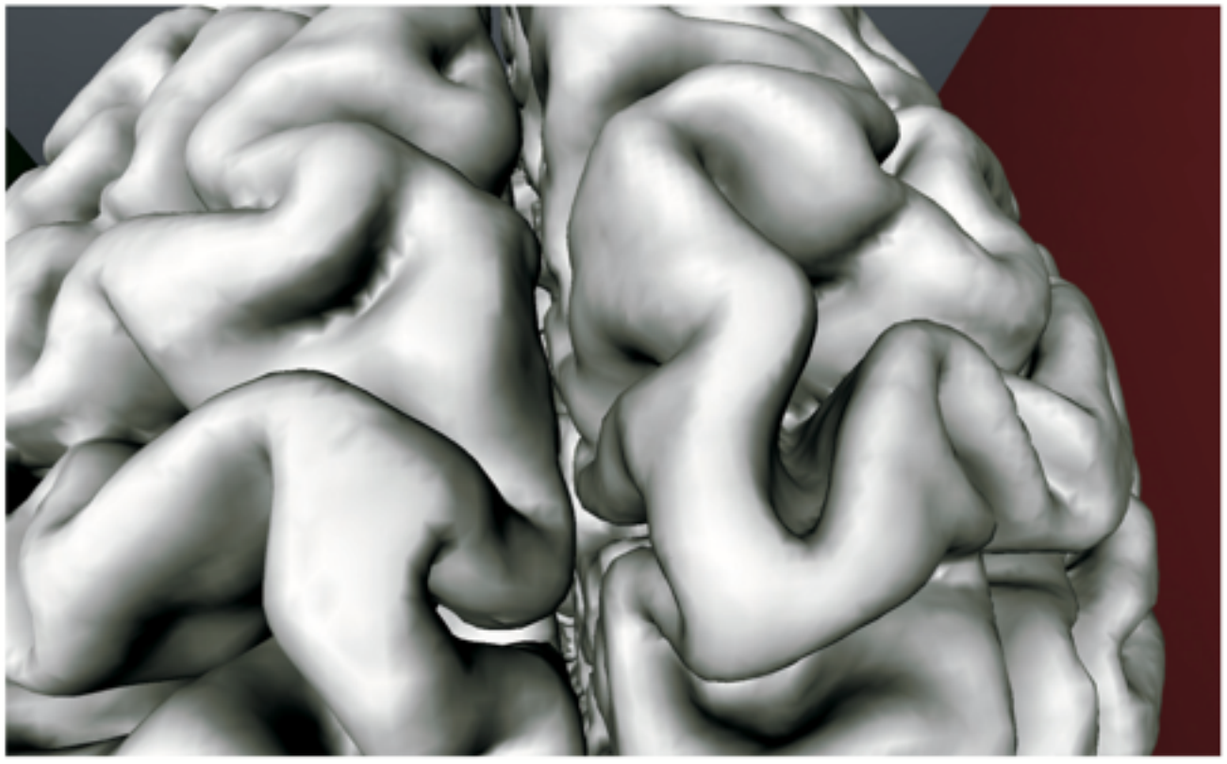
- Global illumination can be achieved ray tracing — casting rays from the eye that reflect, create shadows, scatter, etc.

- Can also cast rays from light sources (photon mapping)

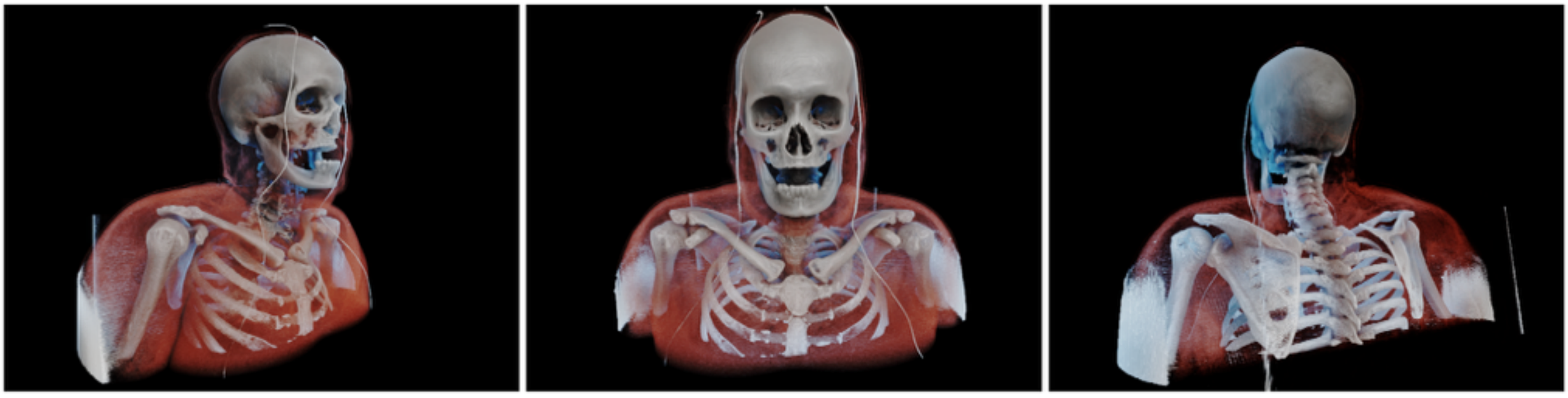




See Kajiya, Blinn, and many more…

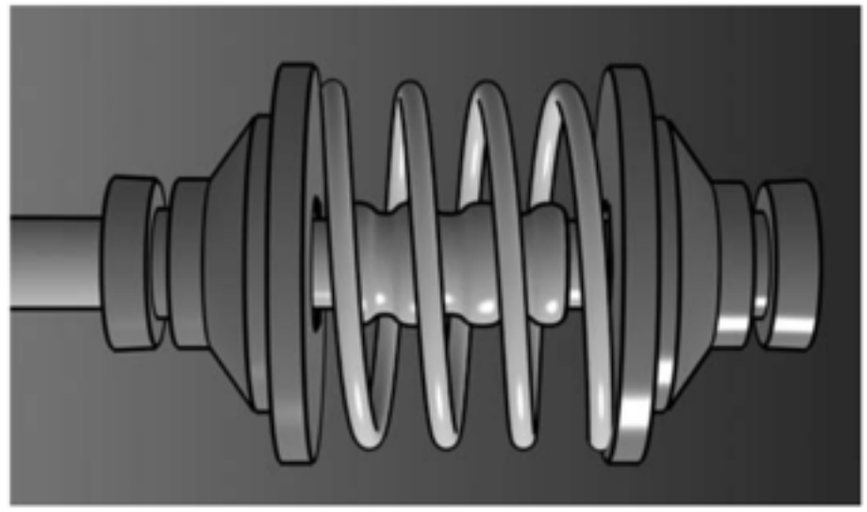


Banks and Beason, 2007



Zhang and Ma, 2013

Nonphotorealistic Rendering (NPR)



Gooch and Gooch

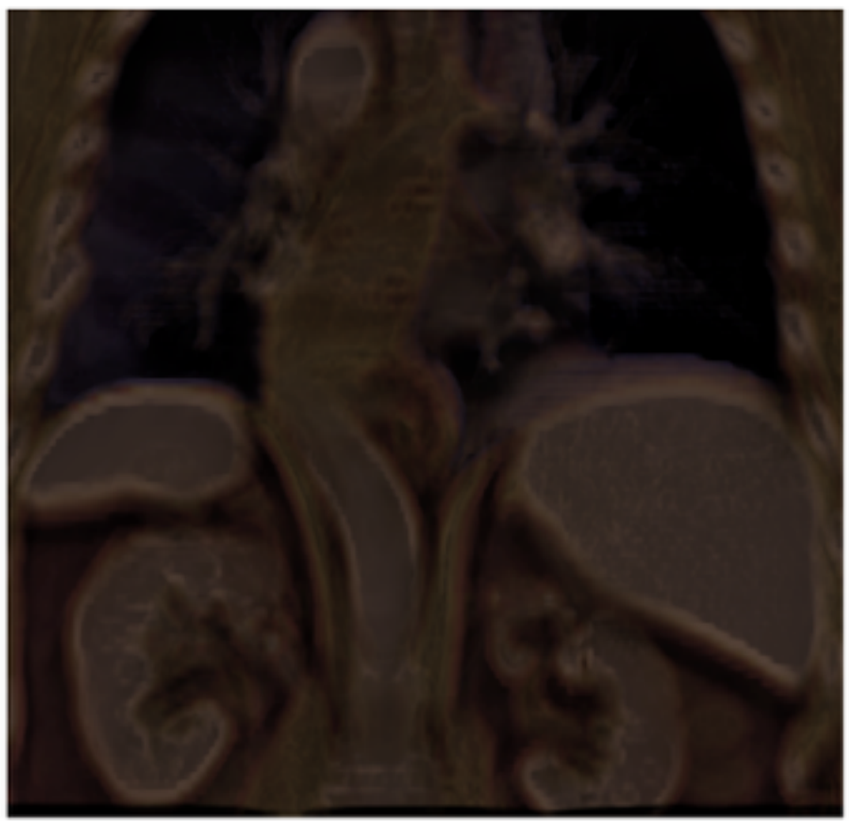


Figure 6. Distance color blending and halos around features of CT volume.

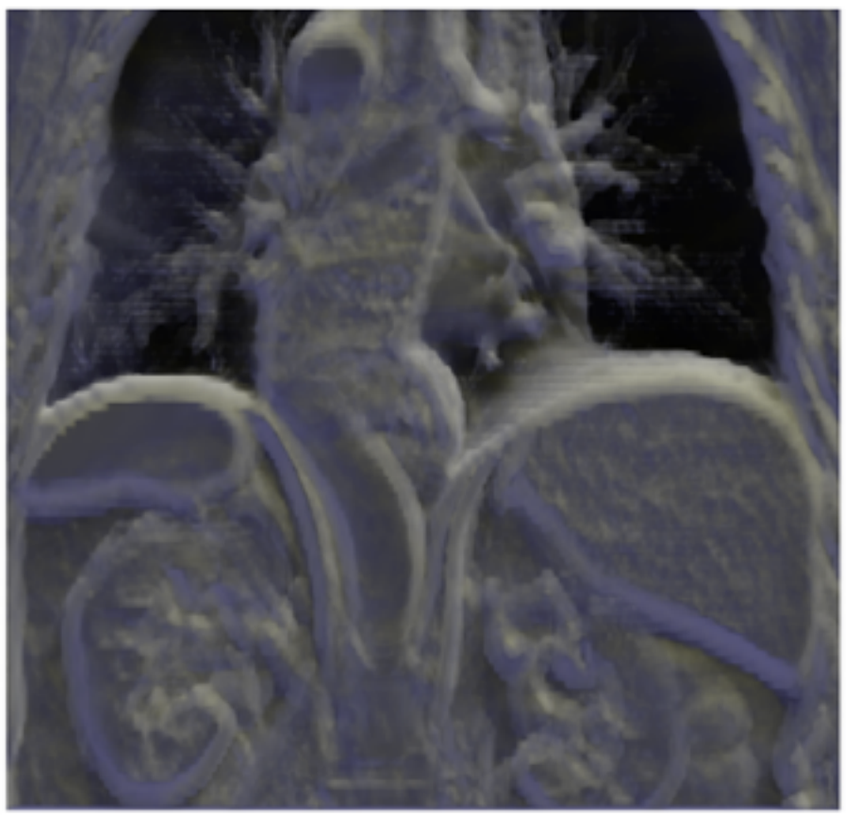


Figure 7. Tone shading in CT volume. Surfaces toward light receive warm color.

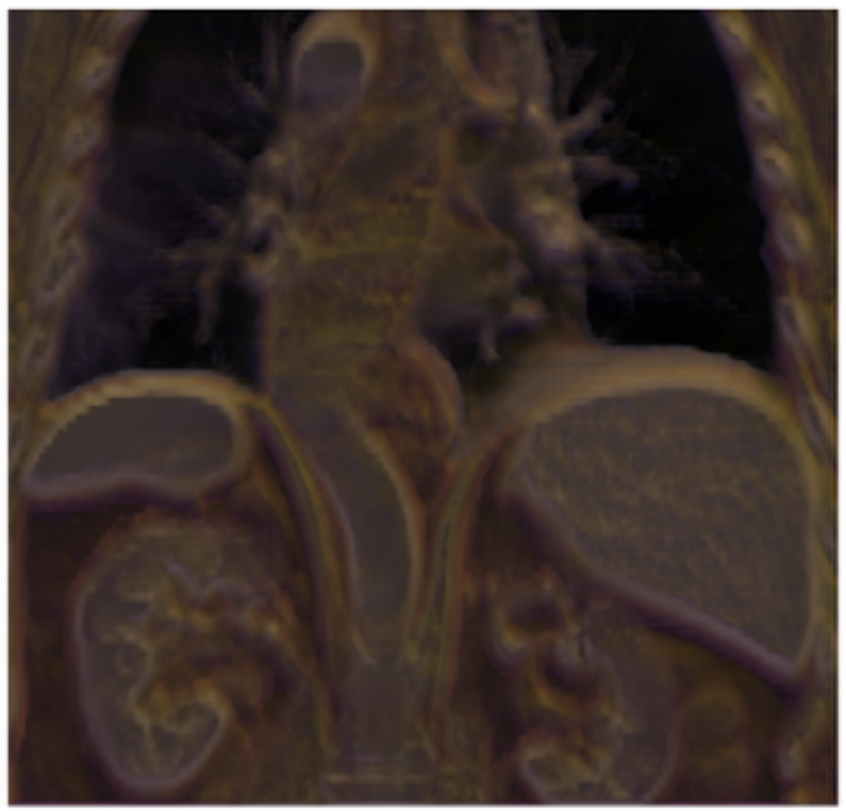
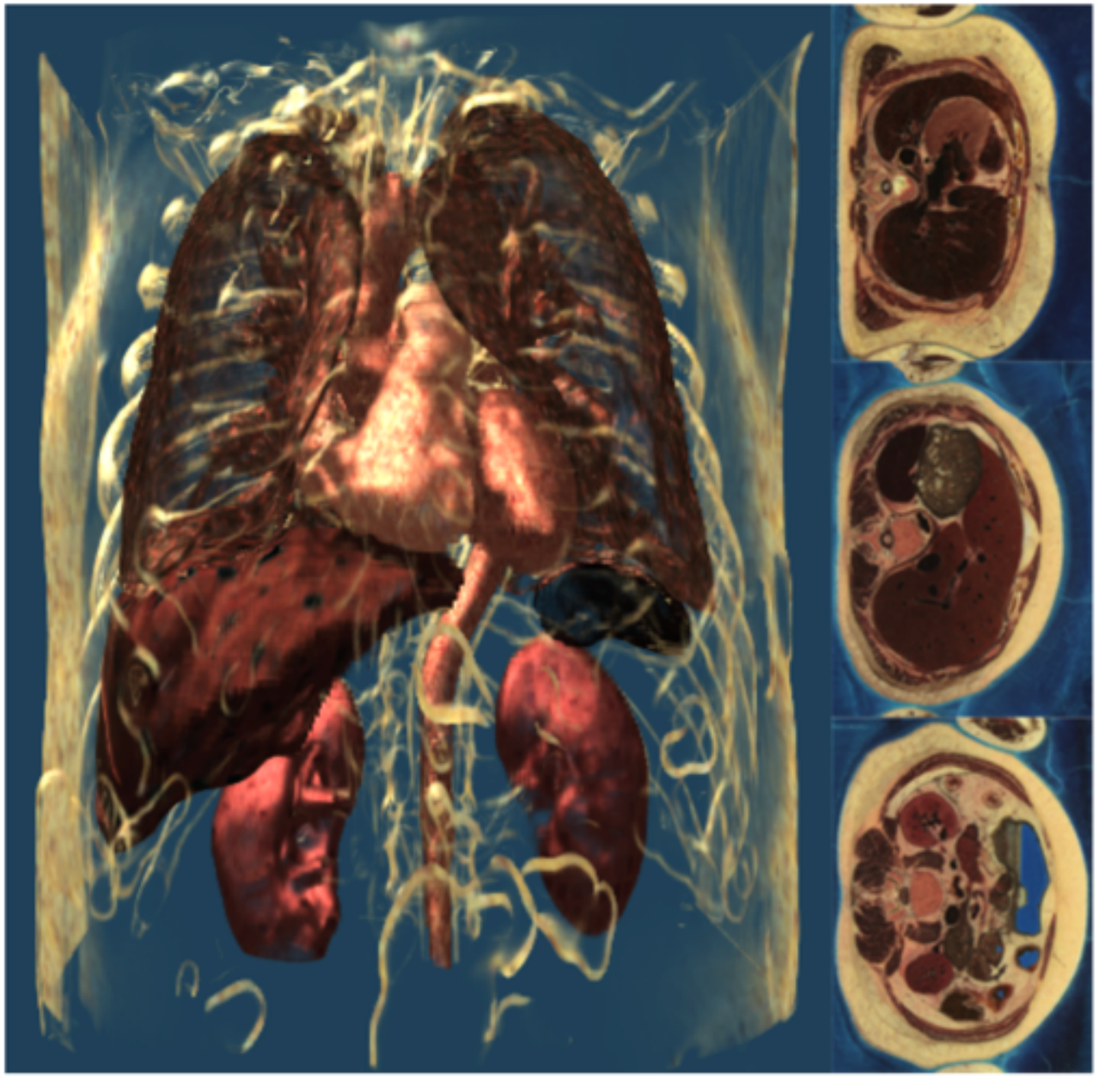
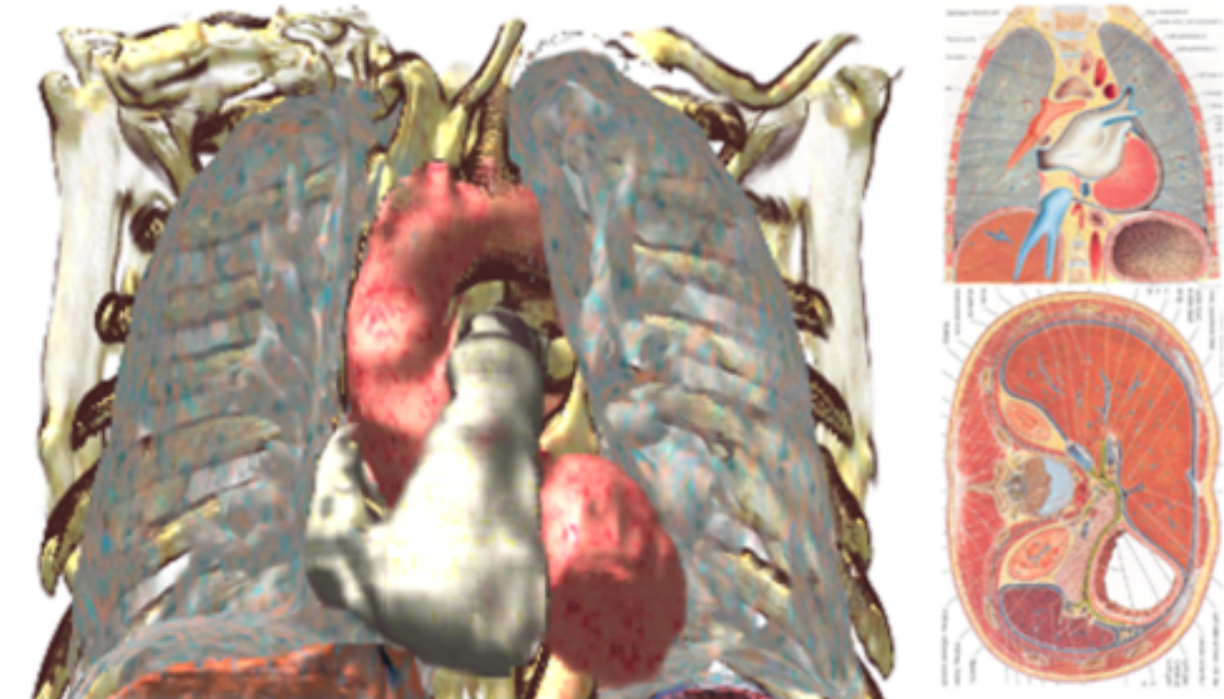


Figure 8. Tone shading in colored volume. Surfaces toward light receive warm color.

Ebert and Rheingans, 2000

Nonphotorealistic Rendering (NPR)





Lu and Ebert, 2005

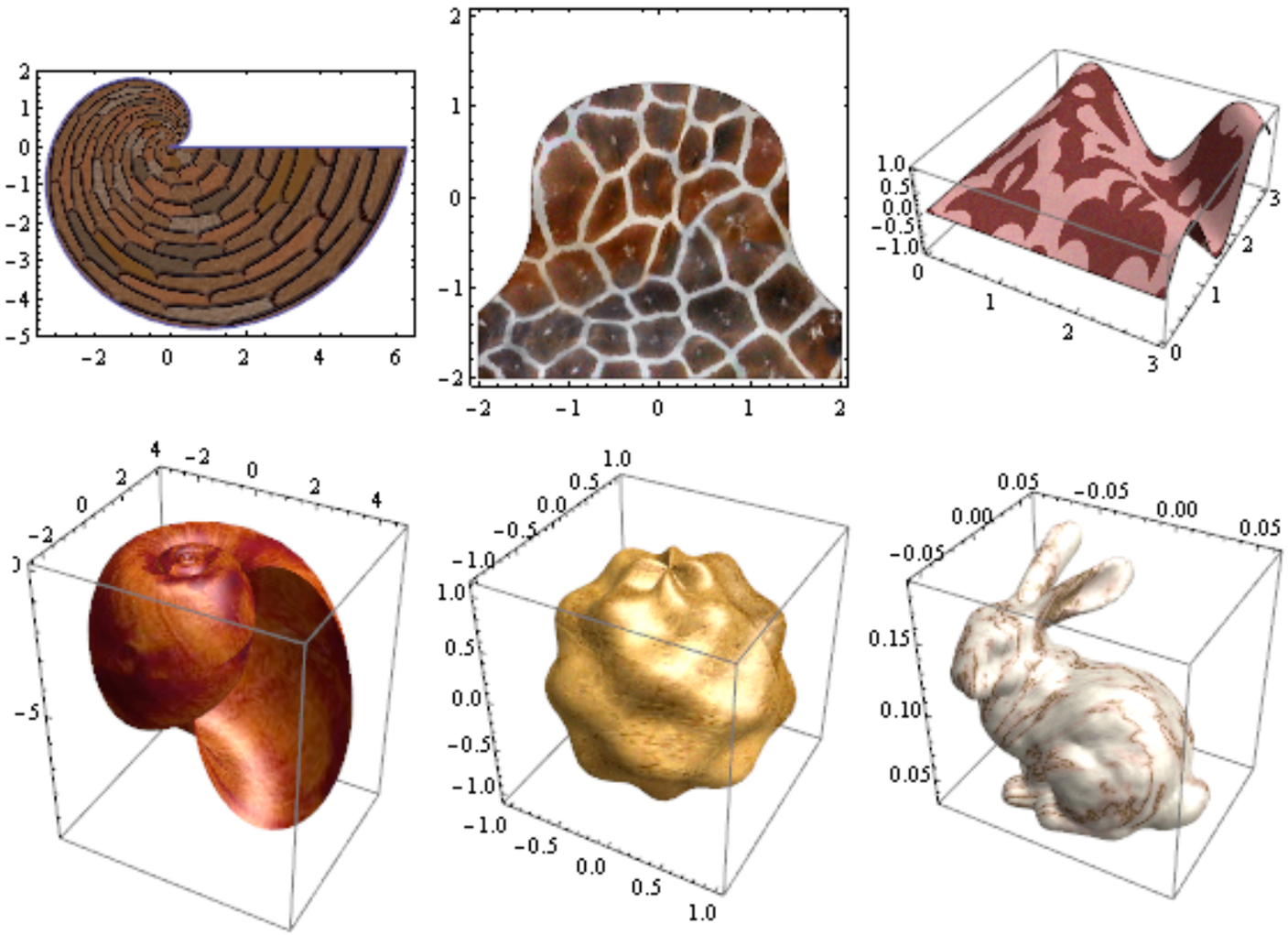
Texture Mapping

- Idea: Use image data to make up for lack of complexity in the modeling process

- Pixels get additional coordinate information (texture coordinates) which index an image.

- Graphics pipelines are specifically designed to take of advantage of this concept, with fast access to memory and builtin interpolation, texture mapping units

- 3D textures are a common way to accelerate volume rendering (more on this soon)



http://www.wolfram.com/mathematica/new-in-8/enhanced-2d-and-3d-graphics/

L19: Volume Rendering

REQUIRED READING

Chapter 8

Arrange Spatial Data

8.1 The Big Picture

For datasets with spatial semantics, the usual choice for *arrange* is to *use* the given spatial information to guide the layout. In this case, the choices of *express*, *separate*, *order*, and *align* do not apply because the position channel is not available for directly encoding attributes. The two main spatial data types are geometry, where shape information is directly conveyed by spatial elements that do not necessarily have associated attributes, and spatial fields, where attributes are associated with each cell in the field. (See Figure 8.1.) For scalar fields with one attribute at each field cell, the two main visual encoding idiom families are isocontours and direct volume rendering. For both vector and tensor fields, with multiple attributes at each cell, there are four families of encoding idioms: flow glyphs that show local information, geometric approaches that compute derived geometry from a sparse set of seed points, texture

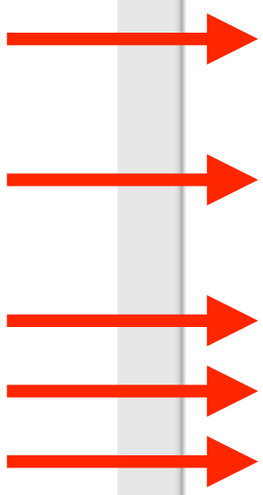
Real-Time Volume Graphics

GLU-based volume rendering for scientific visualization and visual arts

Tutorials

Real-Time Volume Graphics Tutorial

Here are the slides form the original course, the book is based upon. The latest revision are the powerpoint slides of the Tutorial held at the Eurographics 2006 conference in Vienna, Austria, September/5/2006. The same course was also held at SIGGRAPH



» Part 01: Introduction and Theory (ppt) (pdf)

» Part 02: GPU Programming (ppt) (pdf)

» Part 03: Texture-Based Volume Rendering (ppt) (pdf)

» Part 04: GPU-Based Ray Casting (ppt) (pdf)

» Part 05: Transfer Functions (ppt) (pdf)

» Part 06: Local Illumination (ppt) (pdf)

» Part 07: Global Illumination (ppt) (pdf)

» Part 08: Improving Performance (ppt) (pdf)

» Part 09: Improving Quality (ppt) (pdf)

» Part 10: Transfer Functions Reloaded (ppt) (pdf)

» Part 11: Game Developers Guide to Volume Graphics (ppt) (pdf)

» Part 12: Volume Modeling, Deformation and Animation (ppt) (pdf)

Topics

»

» About the Book
» Code Samples
» Datasets
» Errata
» Table of Contents
» Tutorials
» Volume Graphics Tutorials
» What is Volume Graphics?

Topics

» News and Events
» Software

People

» Christof Rezk-Salama
» Daniel Weiskopf
» Joe M. Kniss
» Klaus Engel
» Markus Hadwiger

Recent Comments

» Priscilla on Voreen Framework
» is quibids legit on Voreen Framework
» Prweb.Com on Voreen Framework
» musculo on Voreen Framework
» Антон Павлович on Voreen