CSC 544
Data Visualization

Joshua Levine
josh@arizona.edu

Lecture 18
Volume Data

March 22, 2023

Today's Agenda

- Reminders:
  - A04 questions (due Mar. 27)? P02 (due Mar. 29)?

- Goals for today:
  - Discuss representation of volumetric scalar data for visualization
  - Discuss basic graphics primitives used in volume rendering

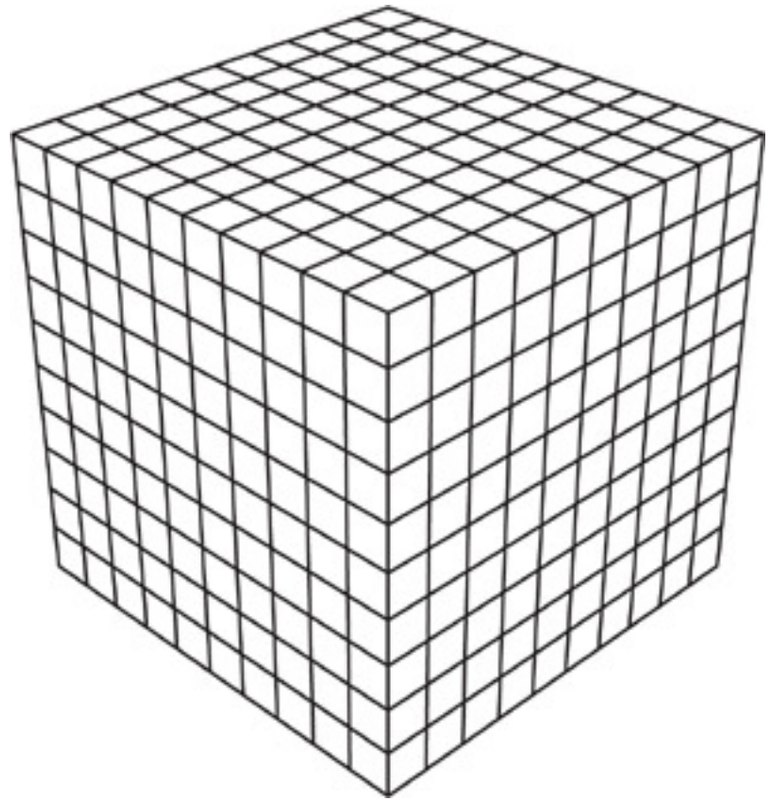
Volume Data

- Visualization of 1D, 2D, or 3D scalar fields
  - 1D scalar field: $\Omega \in R \rightarrow R$
  - 2D scalar field: $\Omega \in R^2 \rightarrow R$
  - 3D scalar field: $\Omega \in R^3 \rightarrow R$

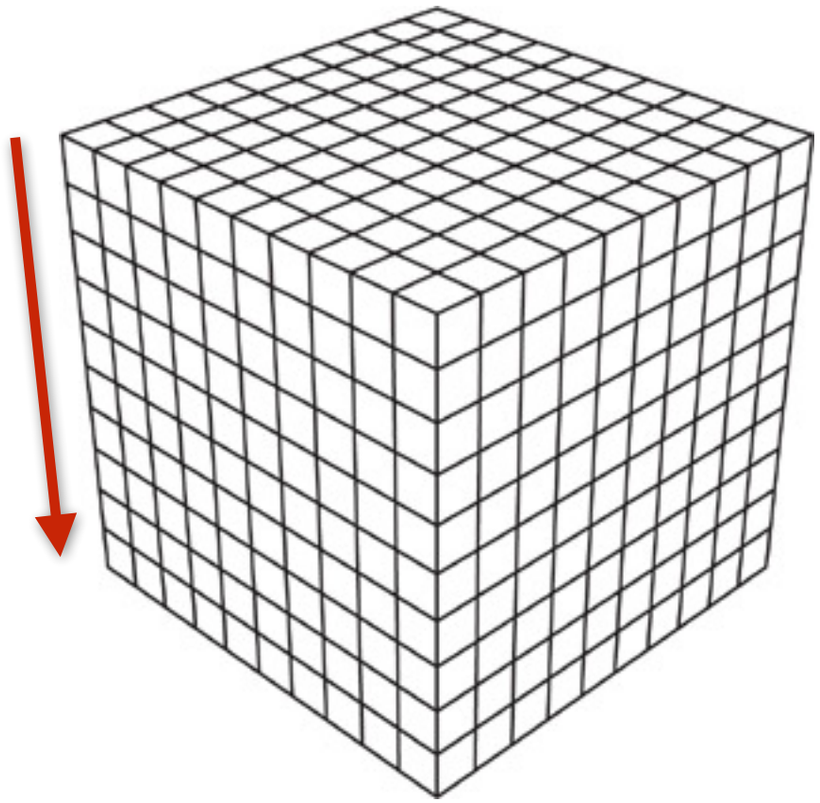
    $\rightarrow$ **Volume visualization!**

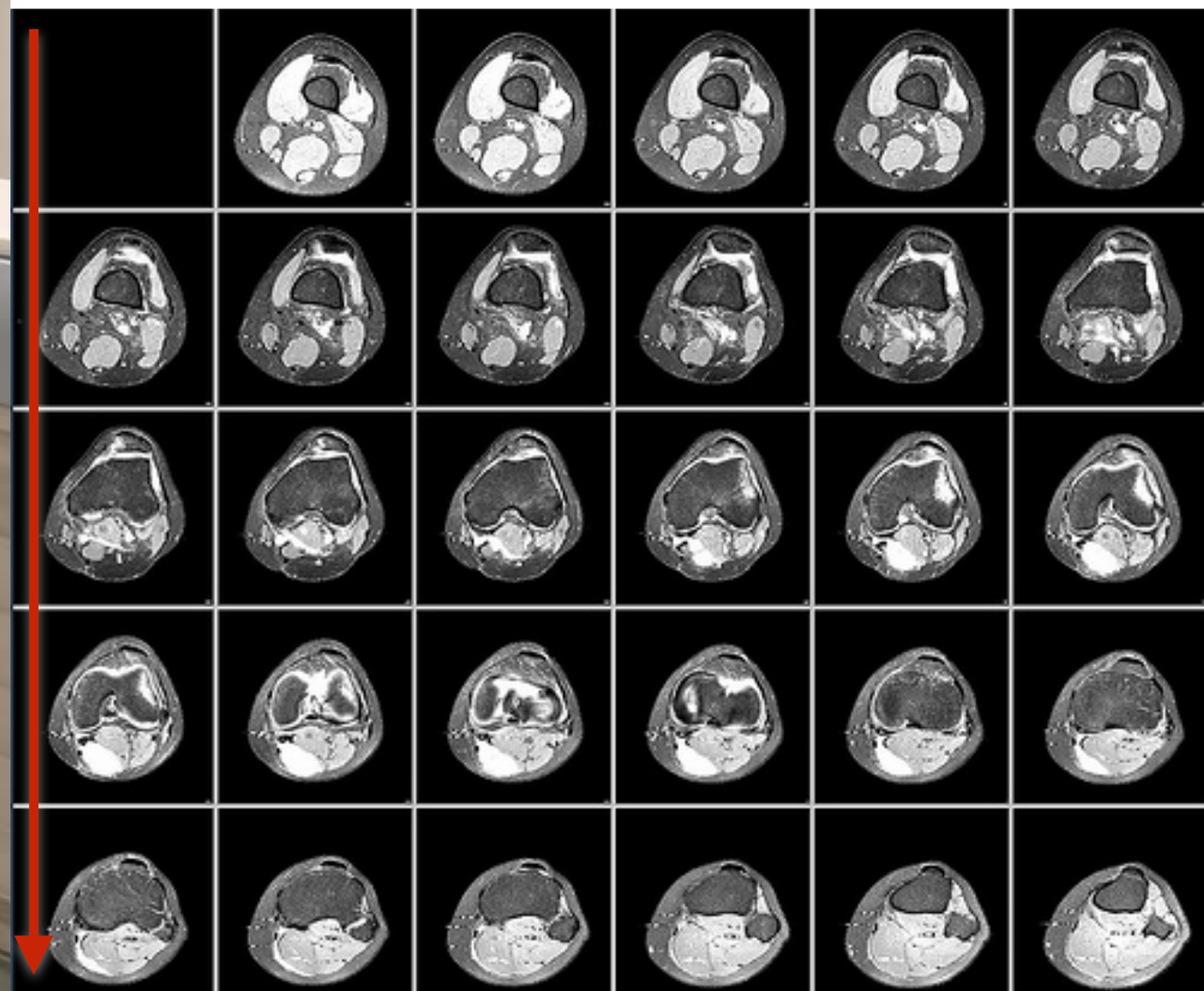

TOSHIBA


TOSHIBA

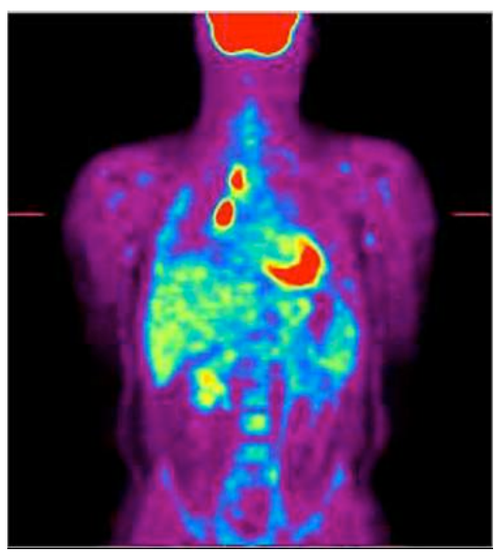



TOSHIBA

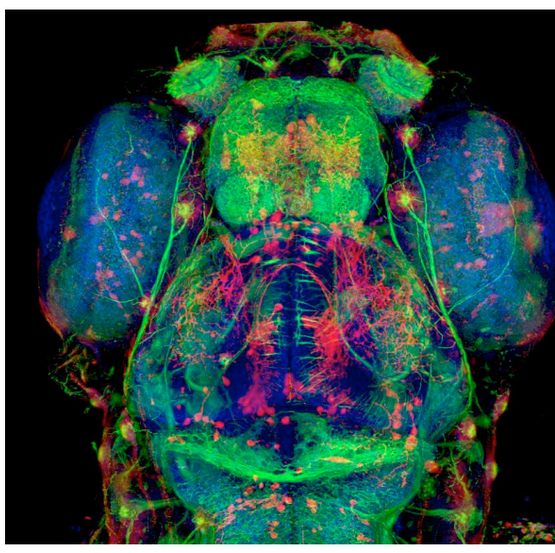




- Measured sources of volume data
  - CT (computed tomography)
  - PET (positron emission tomography)
  - MRI (magnetic resonance imaging)
  - Ultrasound
  - Confocal Microscopy

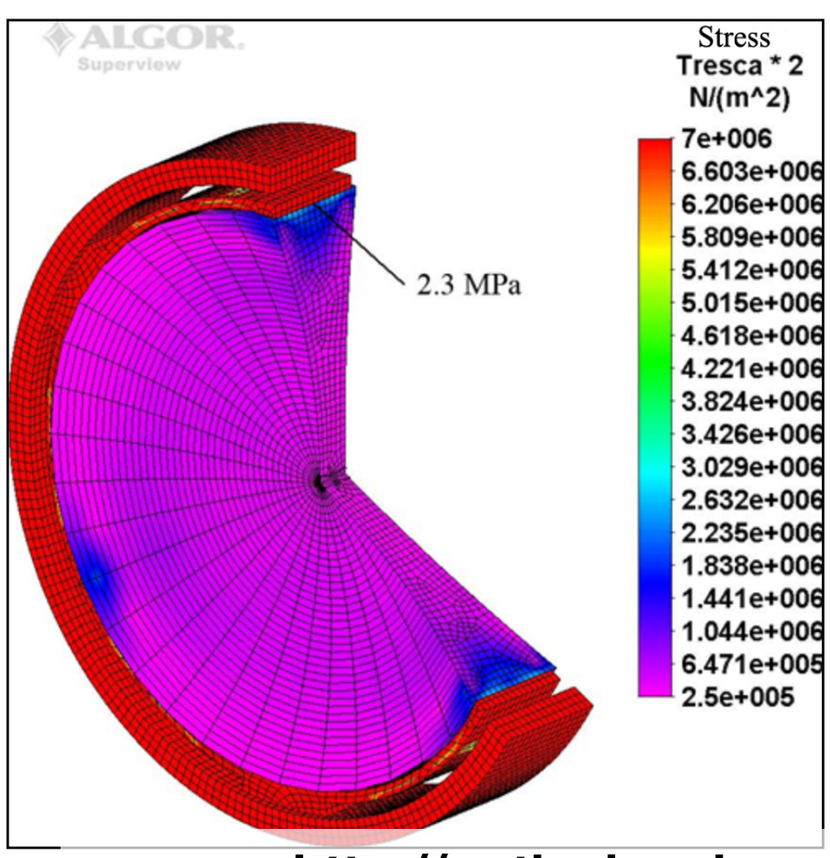



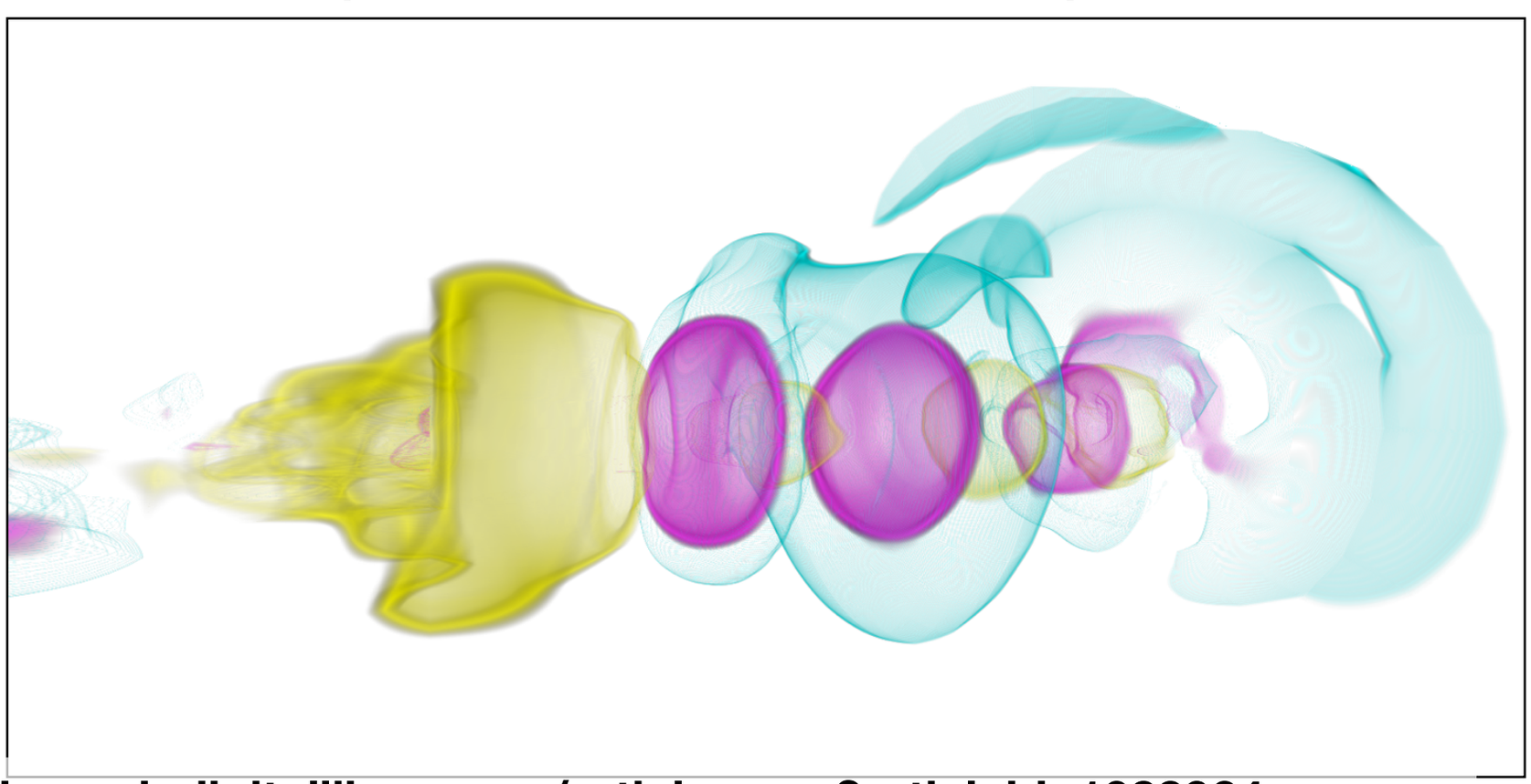




- Synthetic sources of volume data
  - CFD (computational fluid dynamics)
  - Voxelization of discrete geometry

http://www.cs.utah.edu/~bnelson/publications.html





http://opticalengineering.spiedigitallibrary.org/article.aspx?articleid=1088924

T. Fogal, J. Krueger. Size Matters - Revealing Small Scale Structures in Large Datasets, In IFMBE Proceedings, Vol. 25/13, Springer Berlin Heidelberg, pp. 41--44. 2009.

3D Graphics

- The key idea for today:
    - We’ve learned: Visualization transforms data into **primitives** that are displayed.

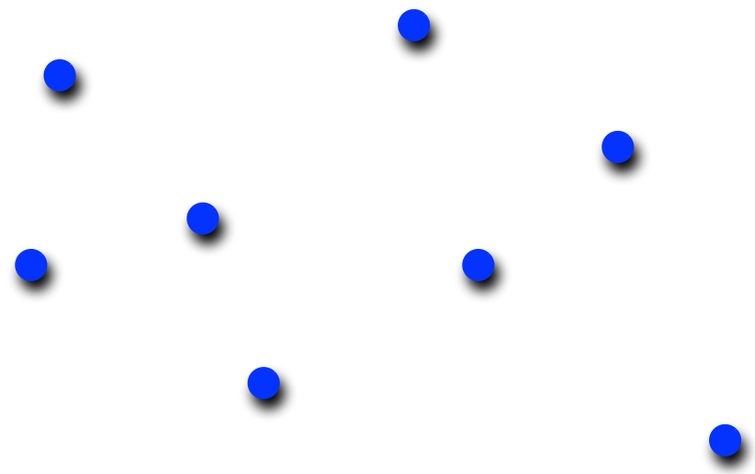
- The key idea for today:
    - We’ve learned: Visualization transforms data into **primitives** that are displayed.
    - Alternatively: Visualization transforms data into **primitives** that are **rendered** using (three-dimensional) **computer graphics** techniques

Graphics Primitives

Graphics Primitives

Points

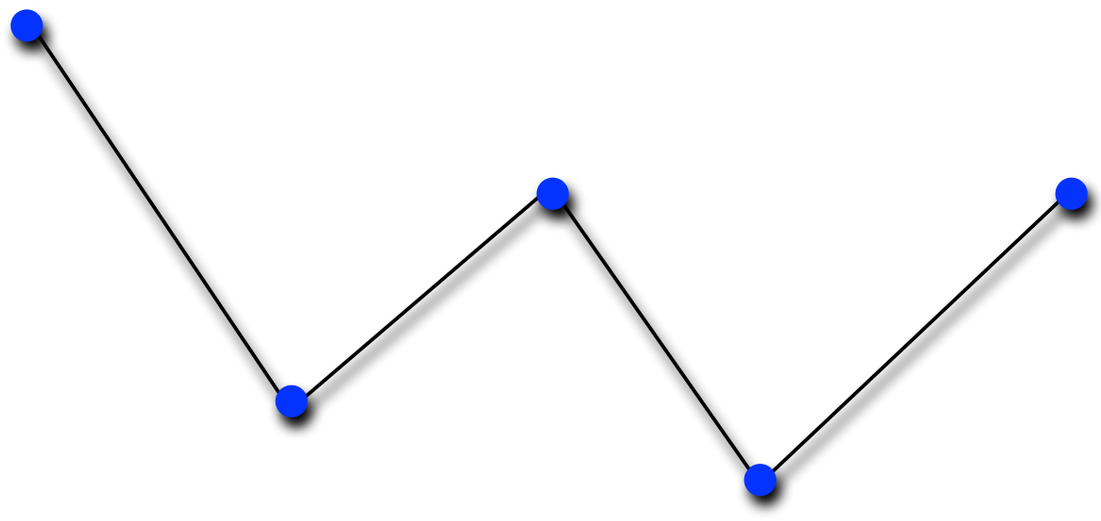
- Positions in space
- 2D, 3D coordinates
- 0-dimensional objects



Graphics Primitives

Lines

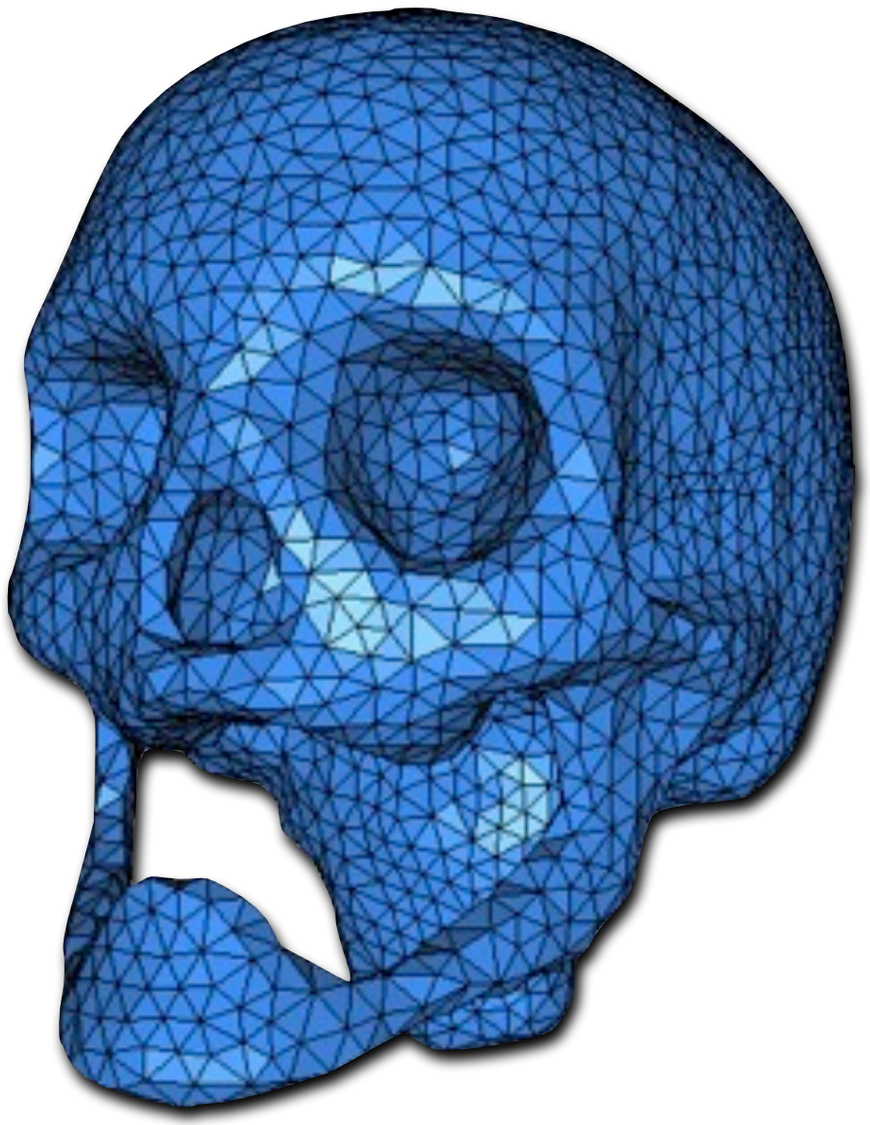
- 1-dimensional objects
- Polygonal description (*"polyline"*)
  - piecewise linear



Graphics Primitives

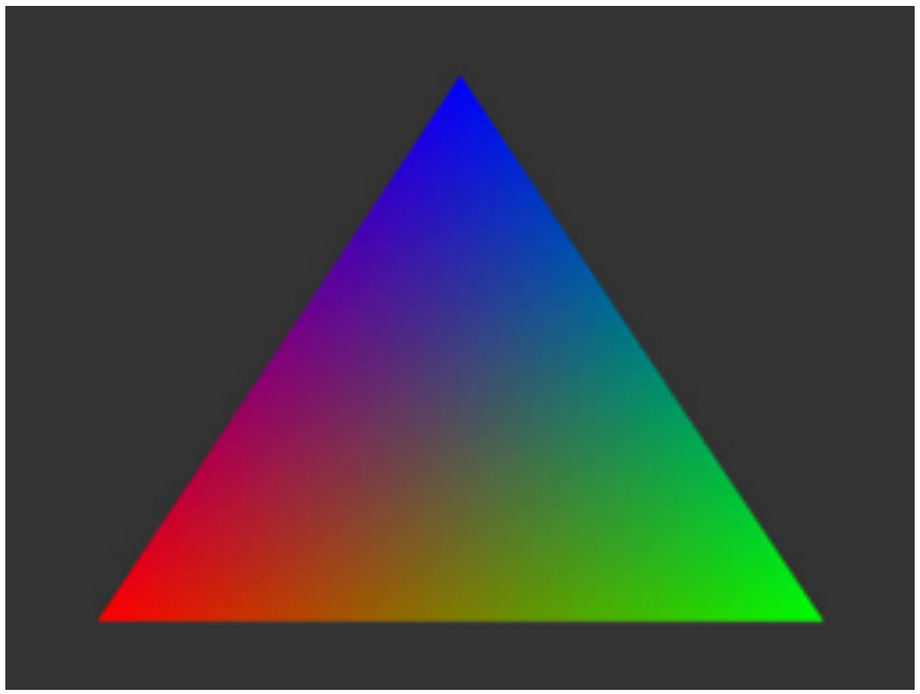
Surfaces

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

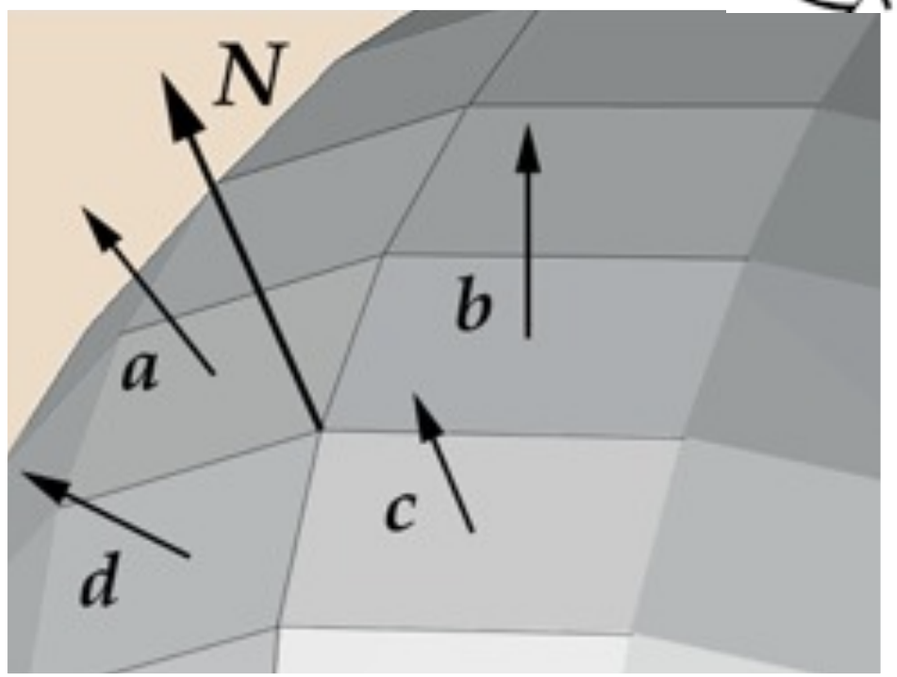


Primitive Attributes

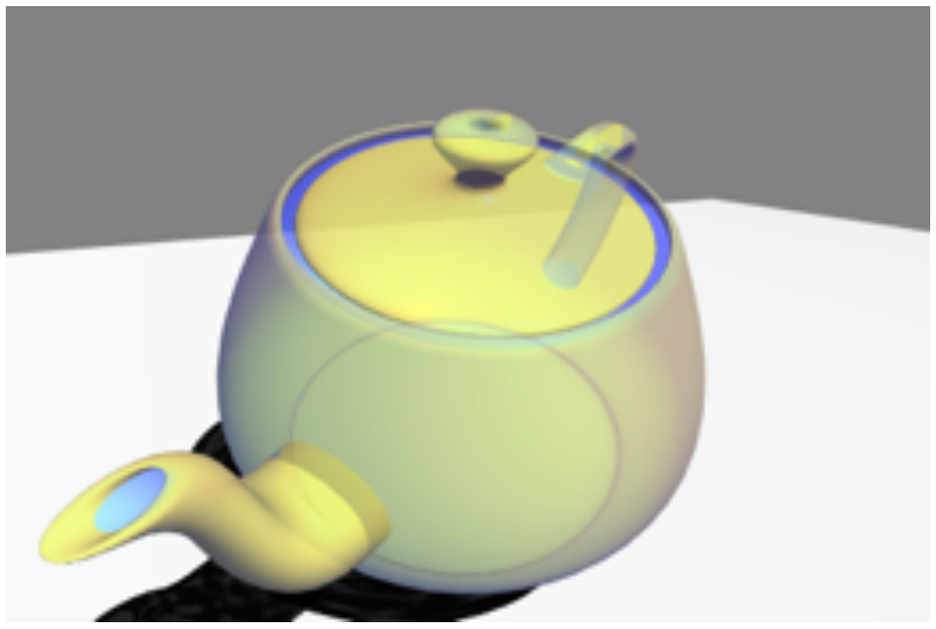
Color

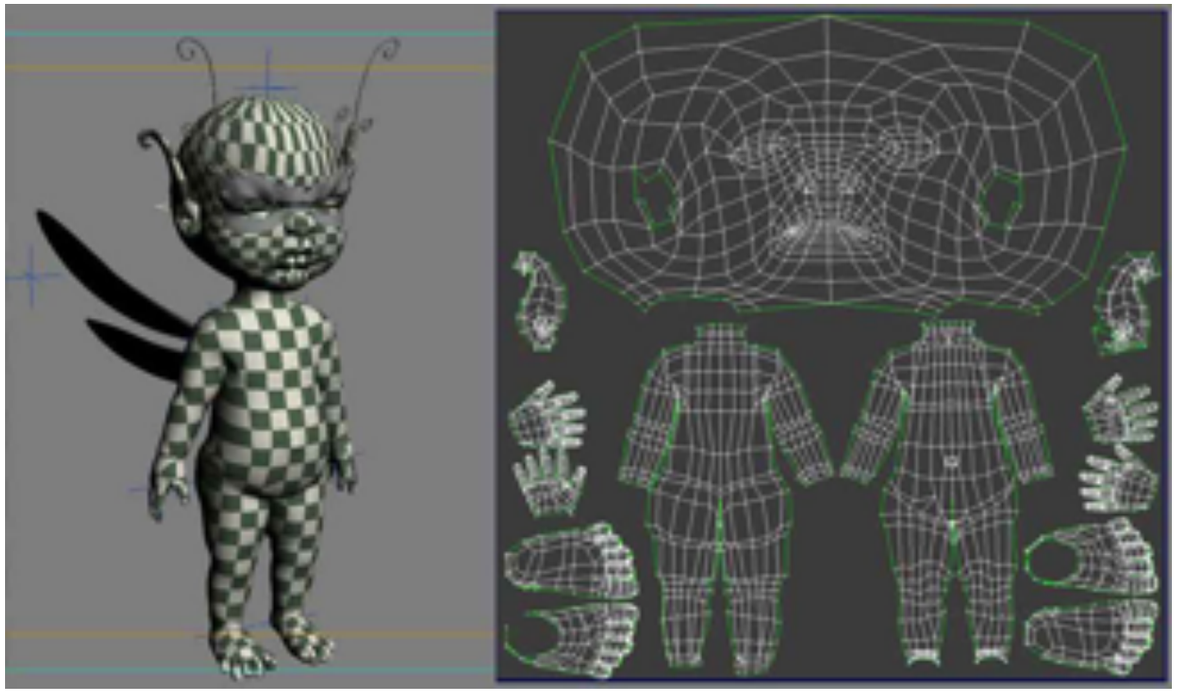


Normal



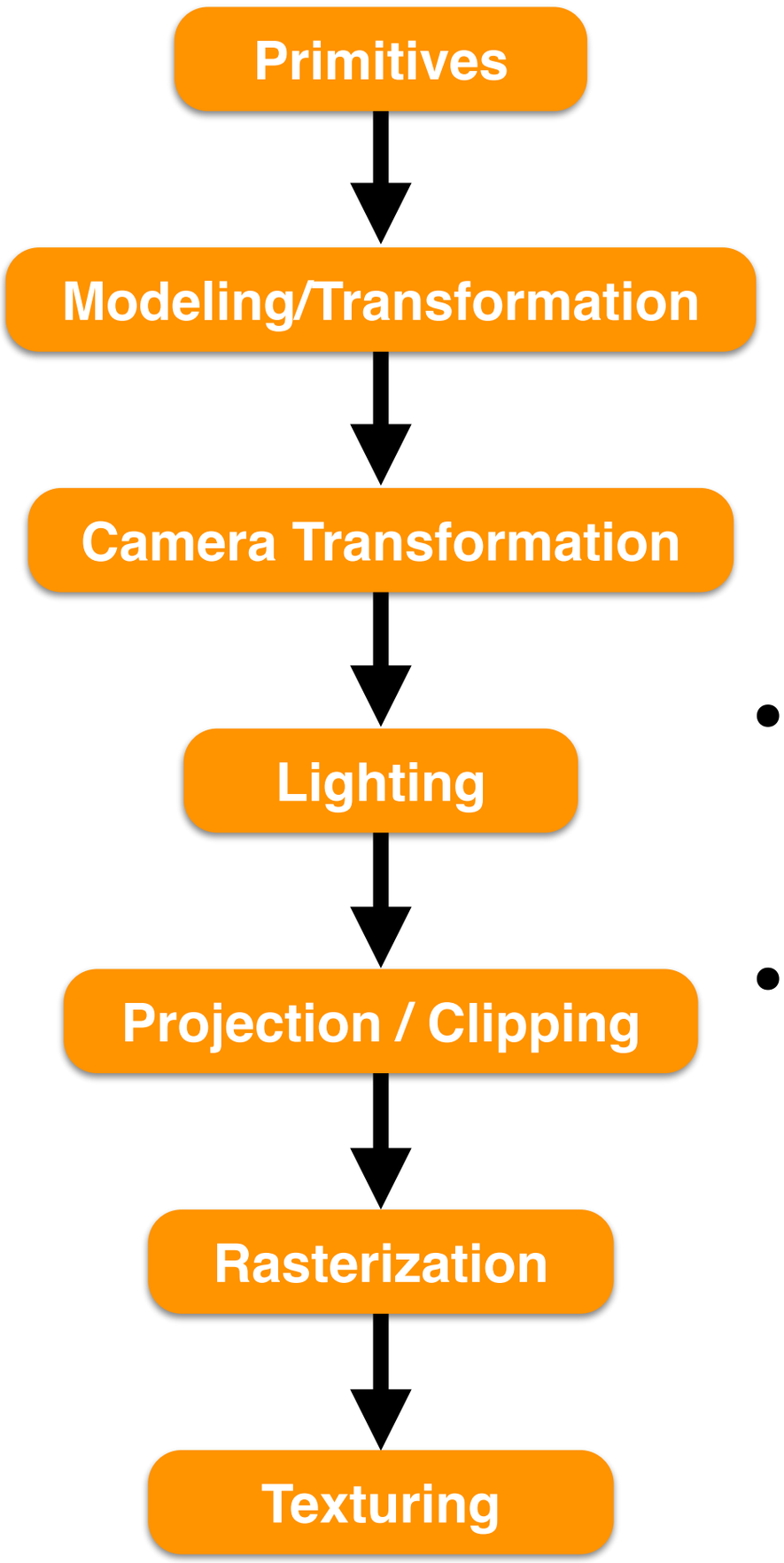
Opacity





Texture coordinates

The Graphics Pipeline

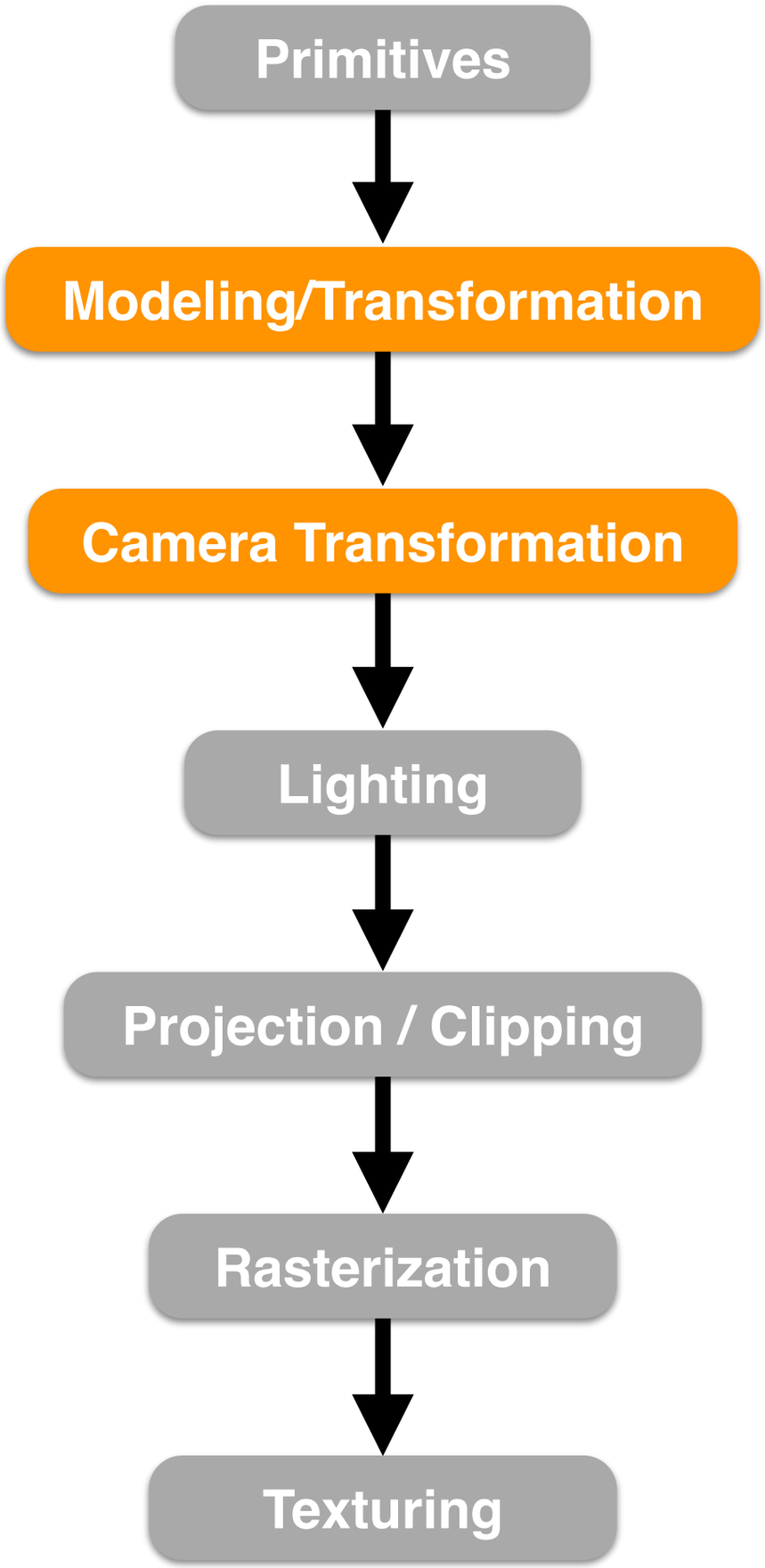


The Pipeline

- 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



Transformations

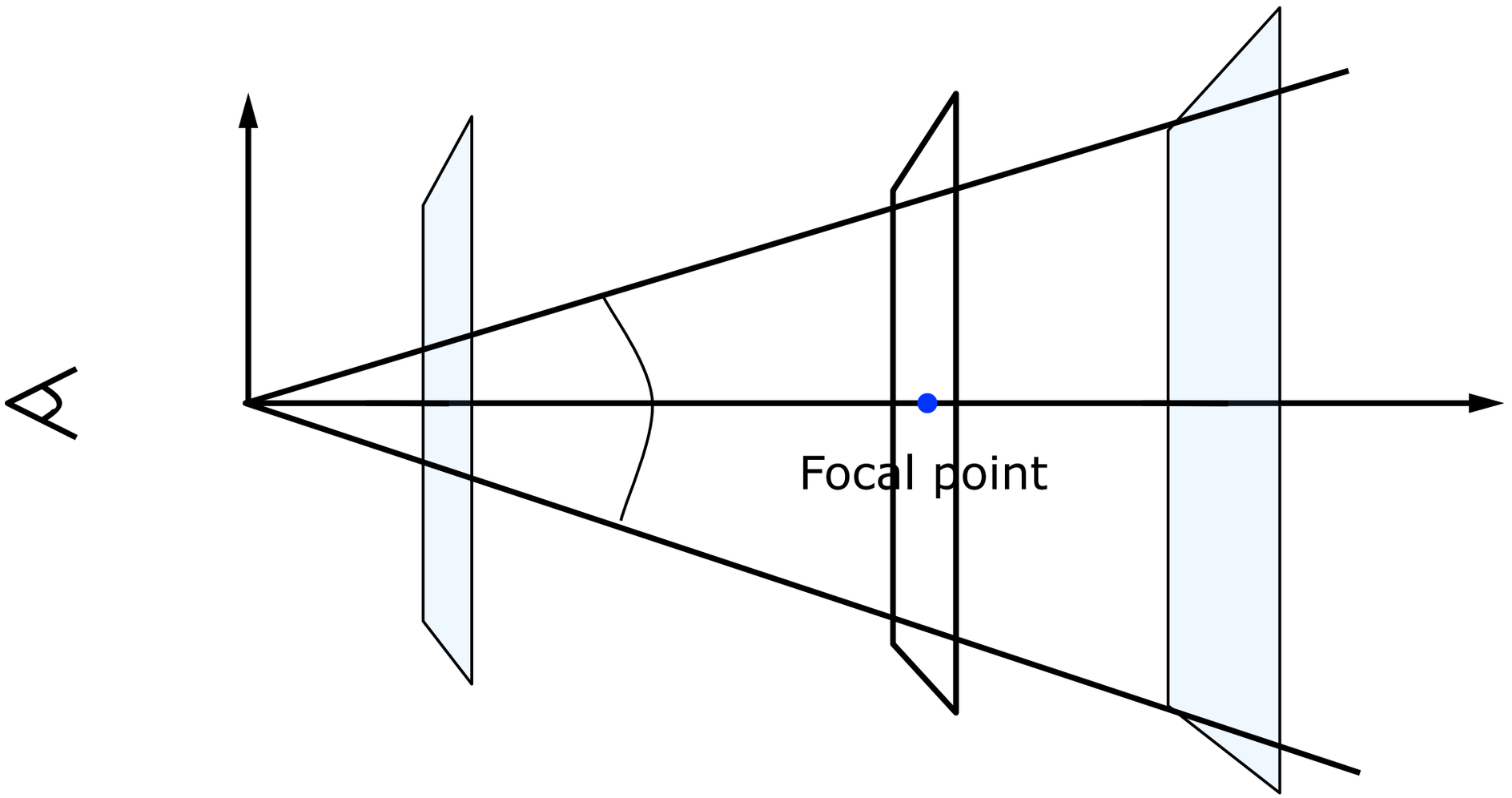
Modeling/Transformation

- 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

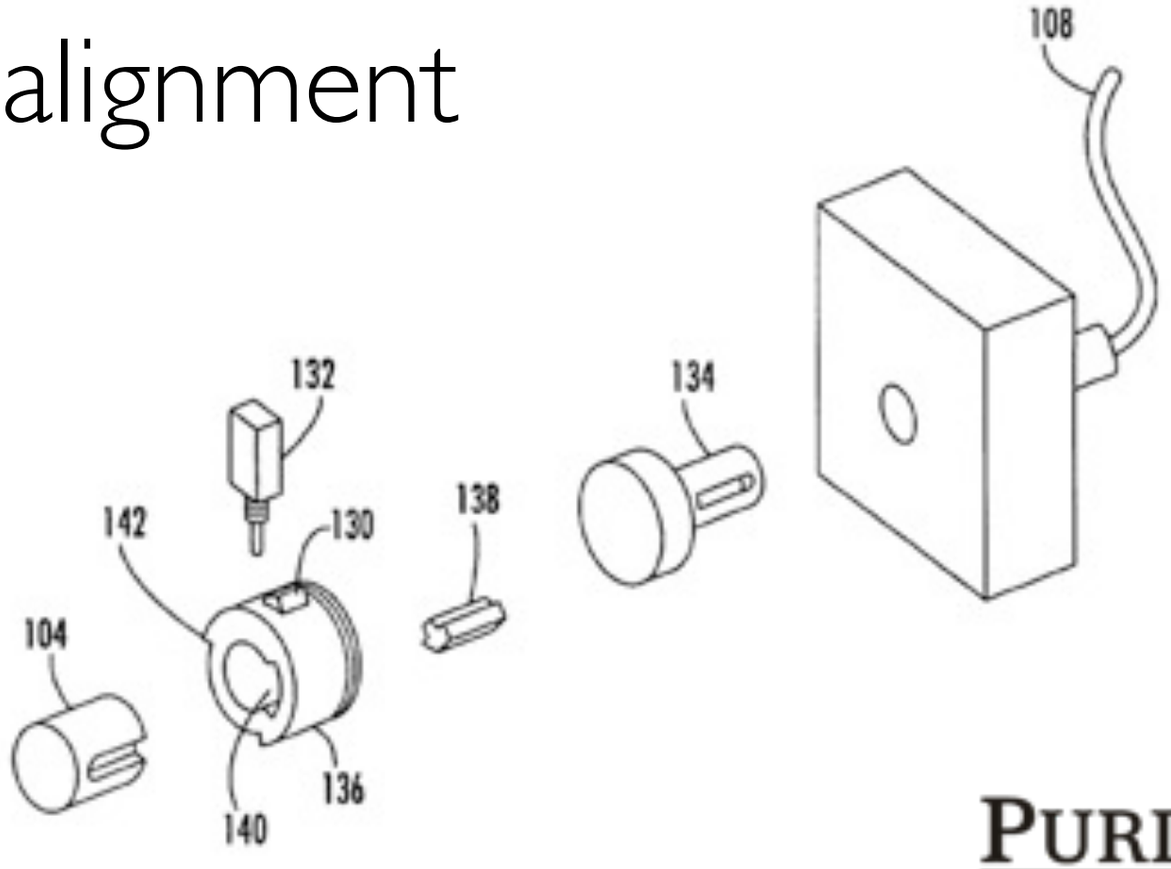


- Need to specify eye position, eye direction, and eye orientation (or “up” vector)
- This information defines a transformation from world coordinates to camera coordinates

Projections

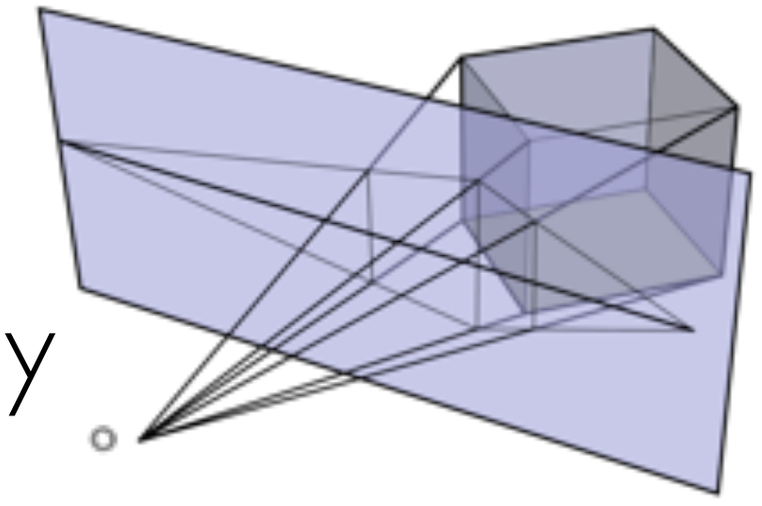
Orthogonal projection

- 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

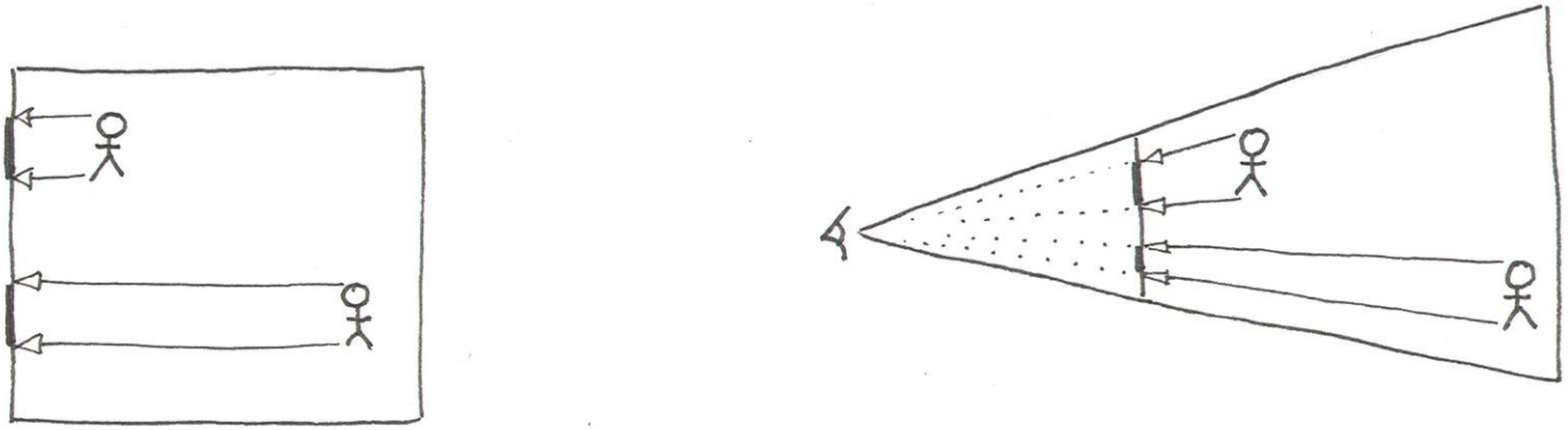
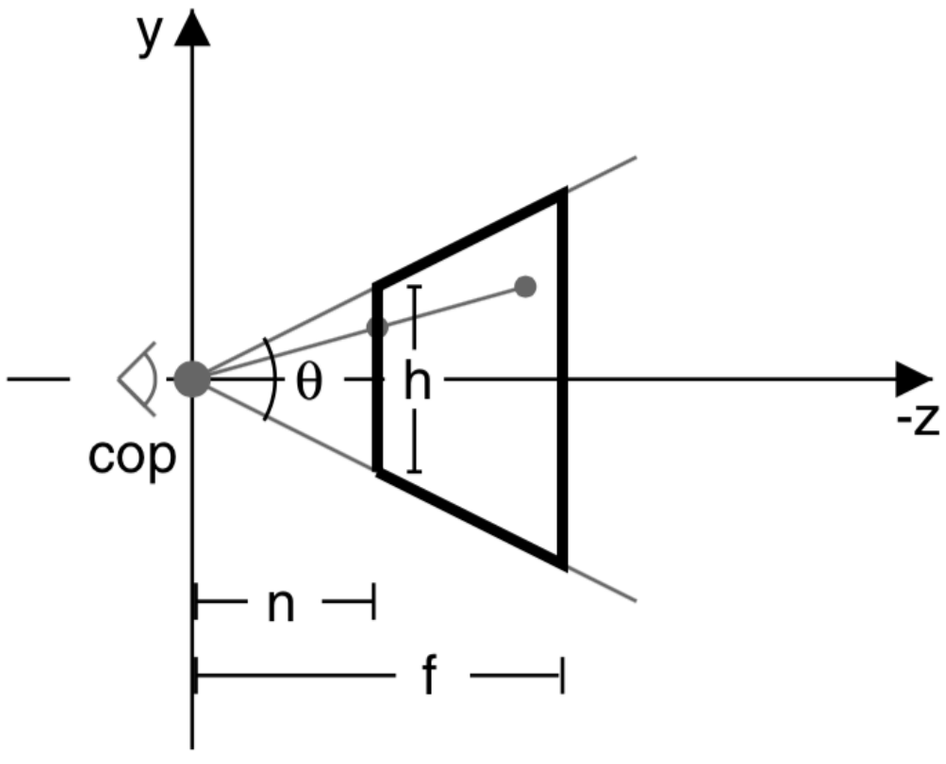


Figure 1: Orthographic and perspective projections: Left, the projectors of an orthographic projection are perpendicular to the viewing plane. Right, the projectors of a perspective projection pass through the center of projection. Foreshortening occurs with perspective.

Dianne Hansford

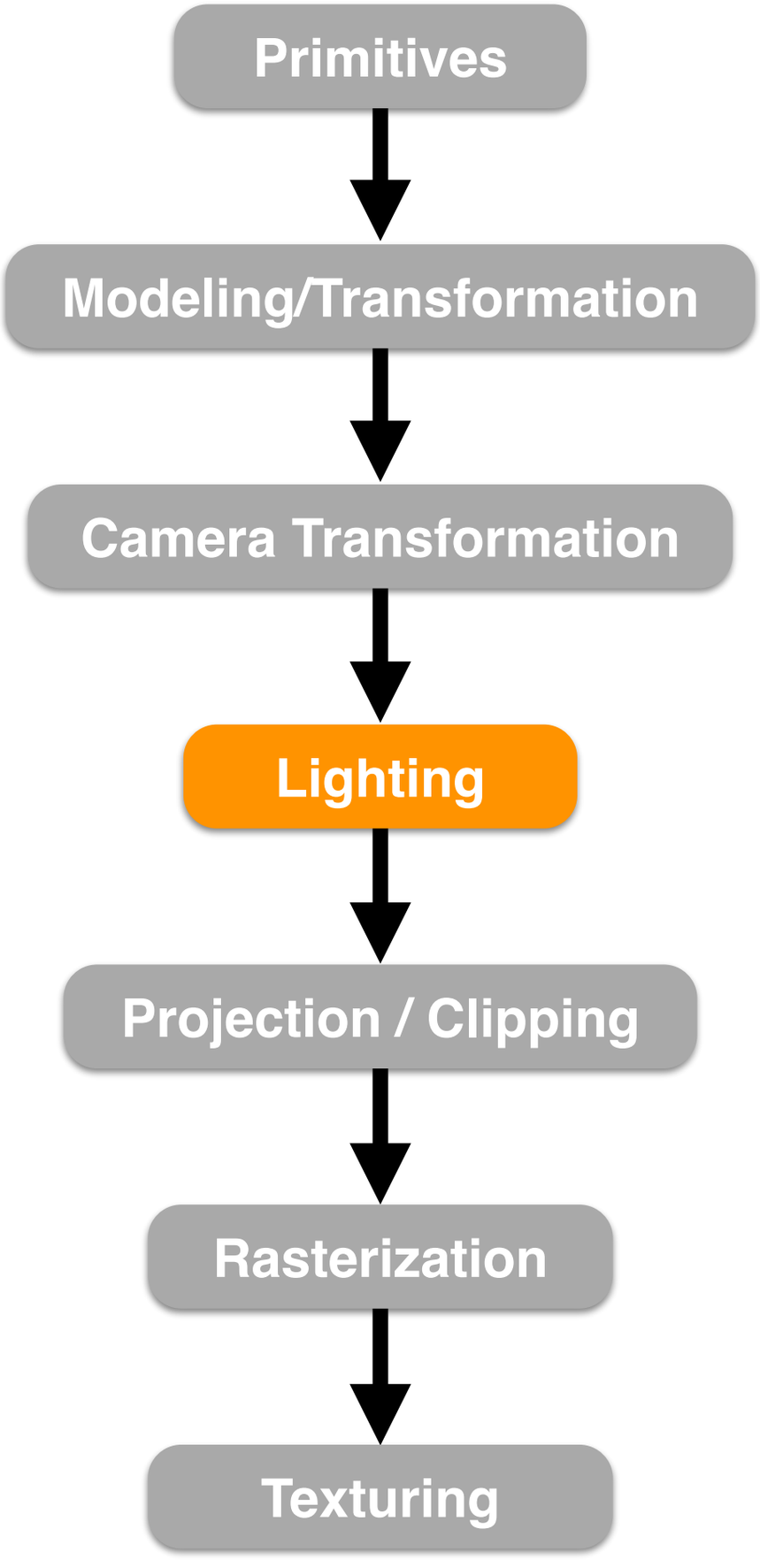
Perspective Projection

- 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



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

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



Lighting and 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

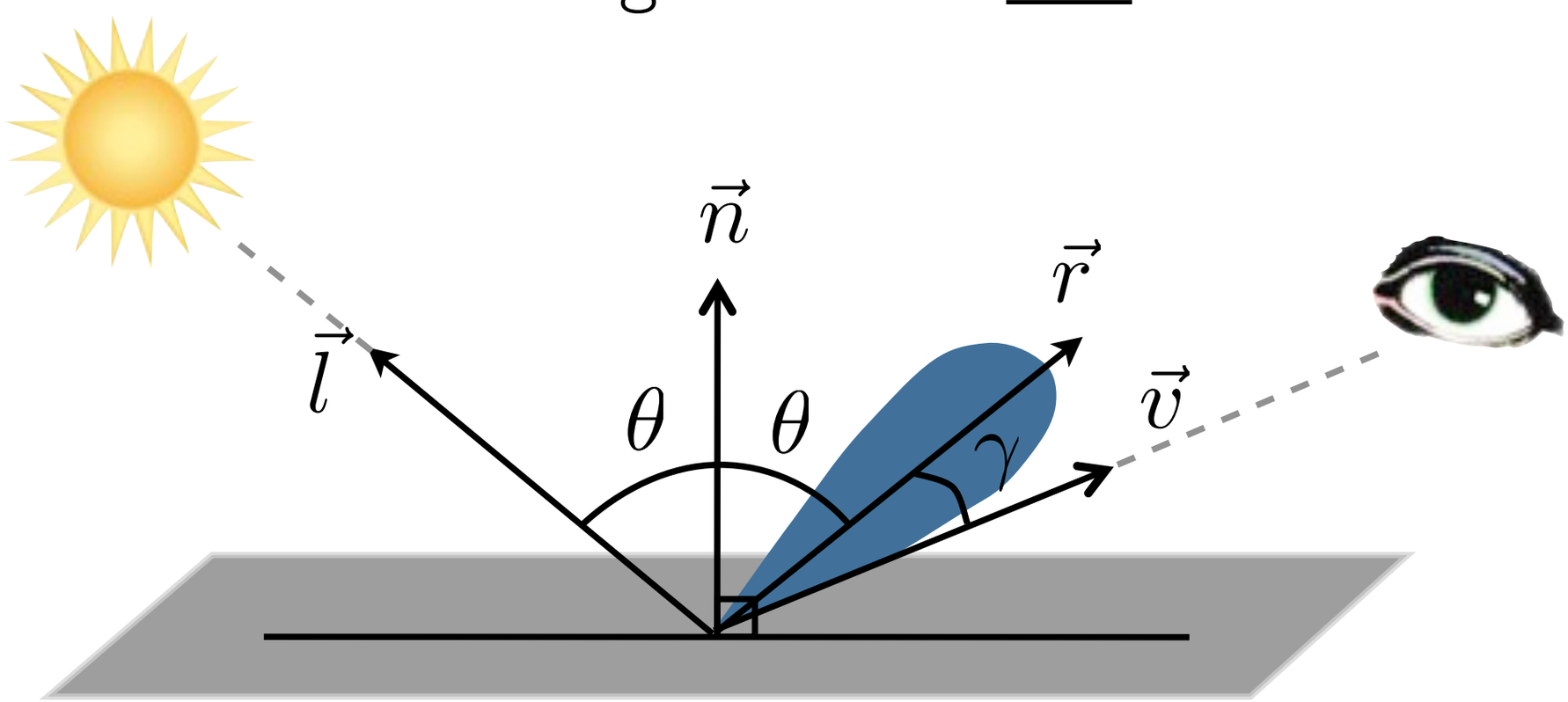
$$I = k_s I_s + k_d I_d + k_a I_a$$

relative contributions
(material specific)

Shading

Phong lighting model: Specular Reflection

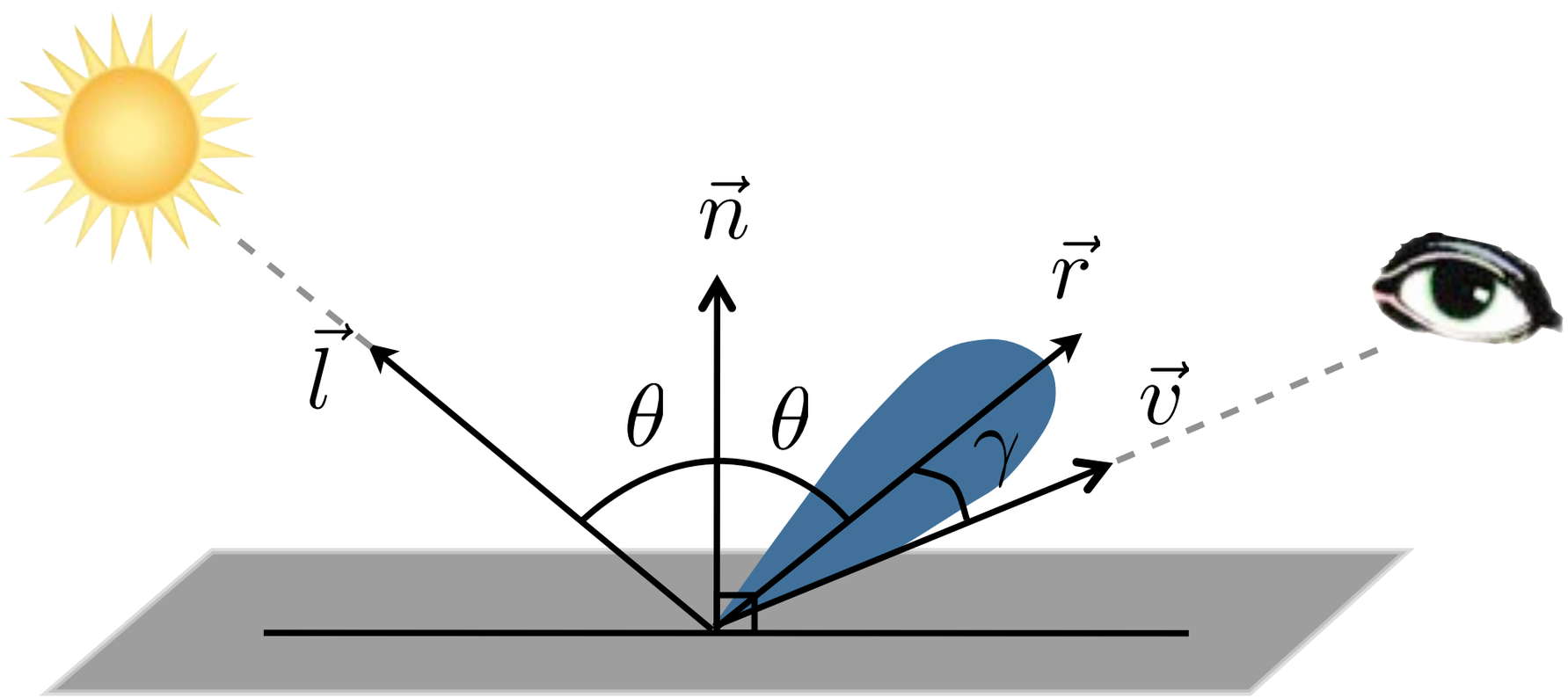
- mirror-like surfaces
- Specular reflection depends on position of the observer relative to light source and surface normal



Shading

Phong lighting model: Specular Reflection

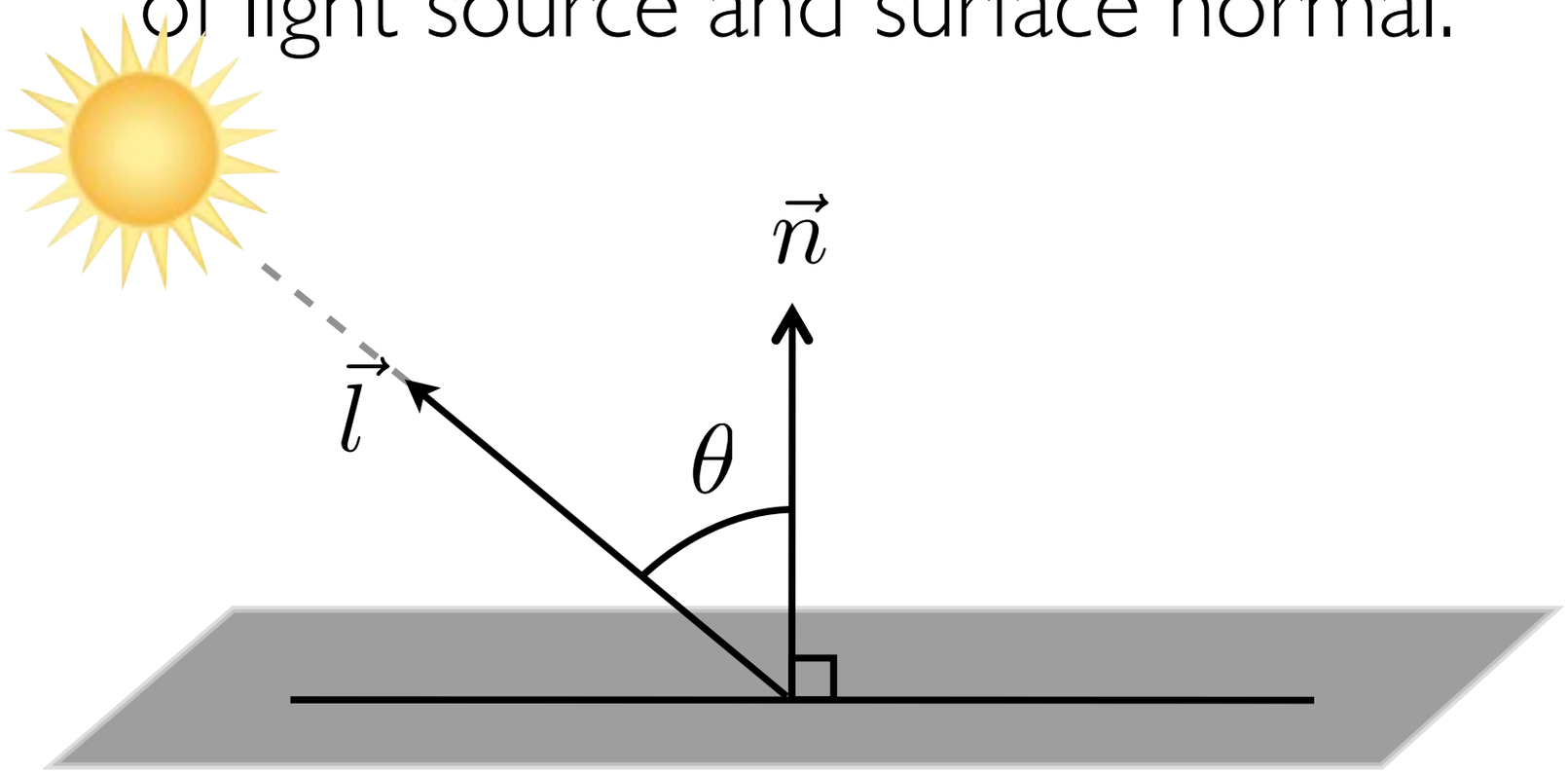
$$I_s = I_i \cos^n \gamma = I_i \cos^n < \vec{r}, \vec{v} >$$

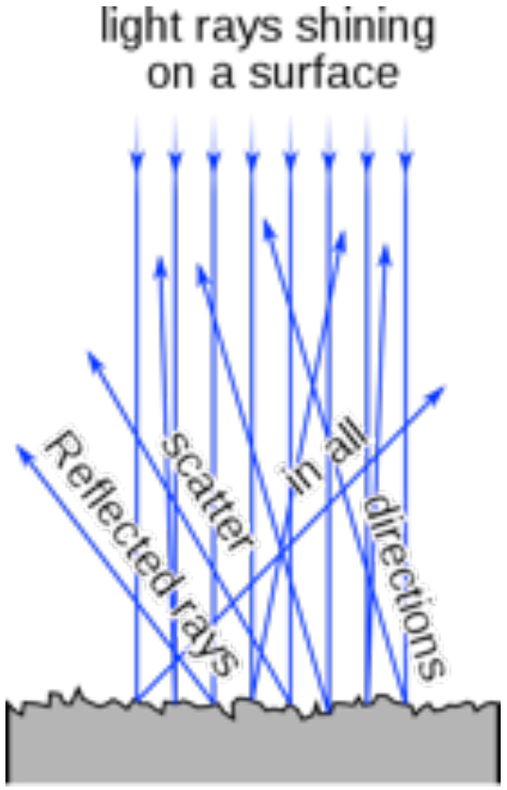


Shading

Phong lighting model: Diffuse Reflection

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

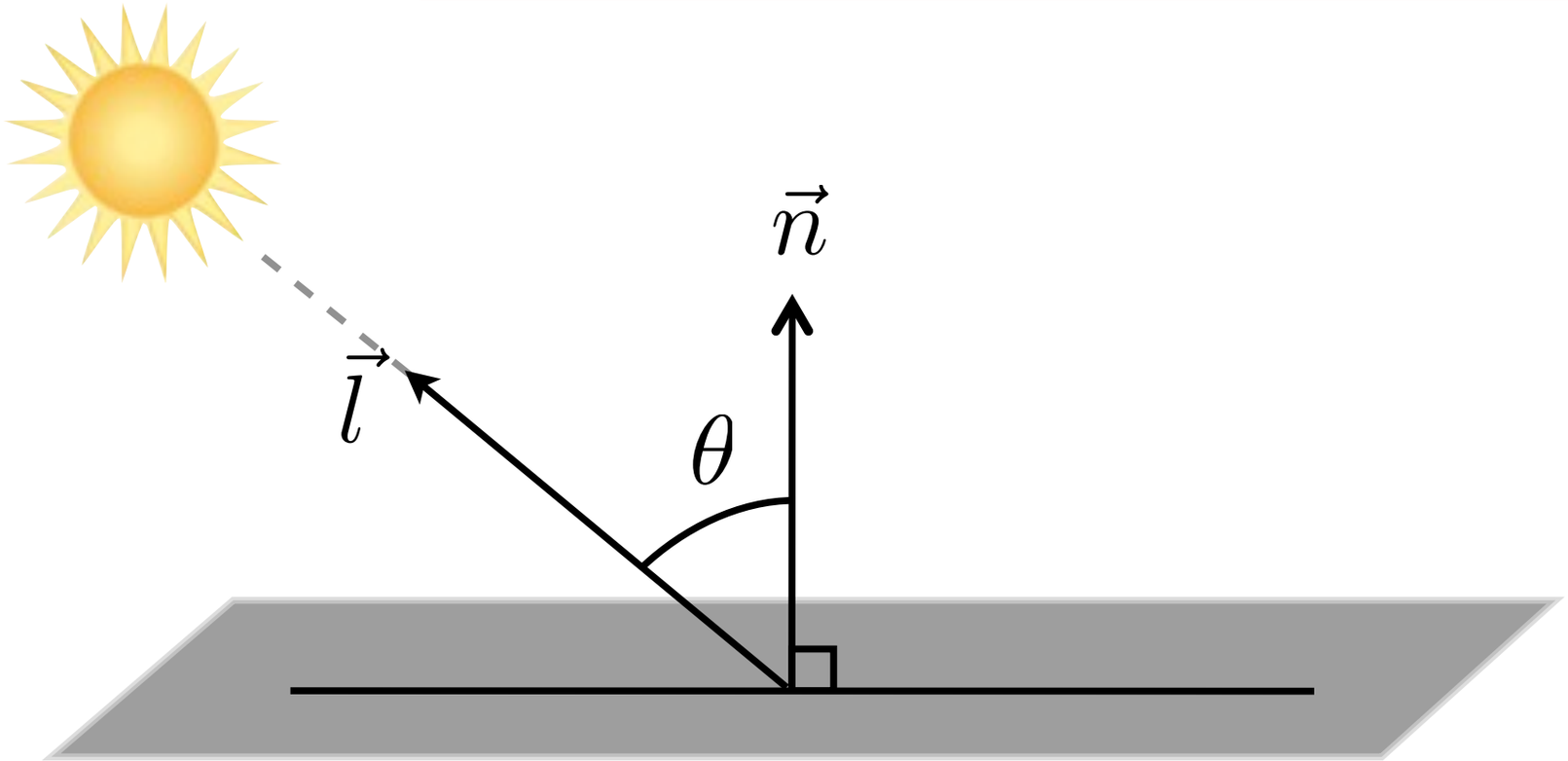


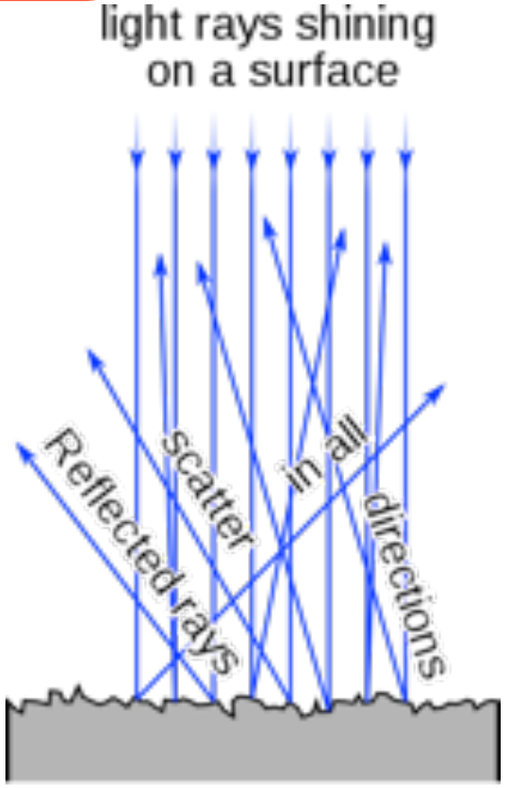


Shading

Phong lighting model: Diffuse Reflection

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





Shading

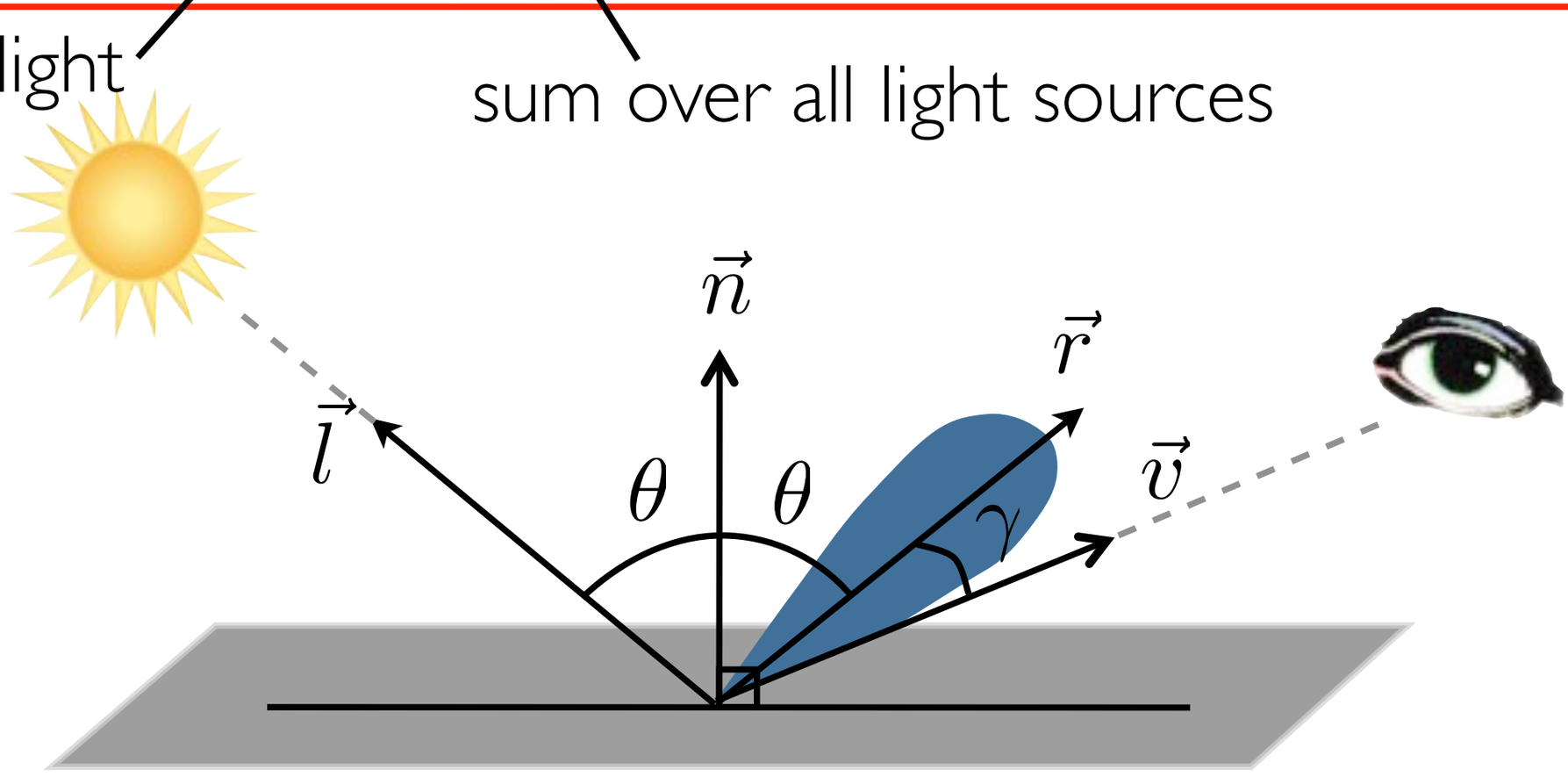
Phong lighting model

per color channel

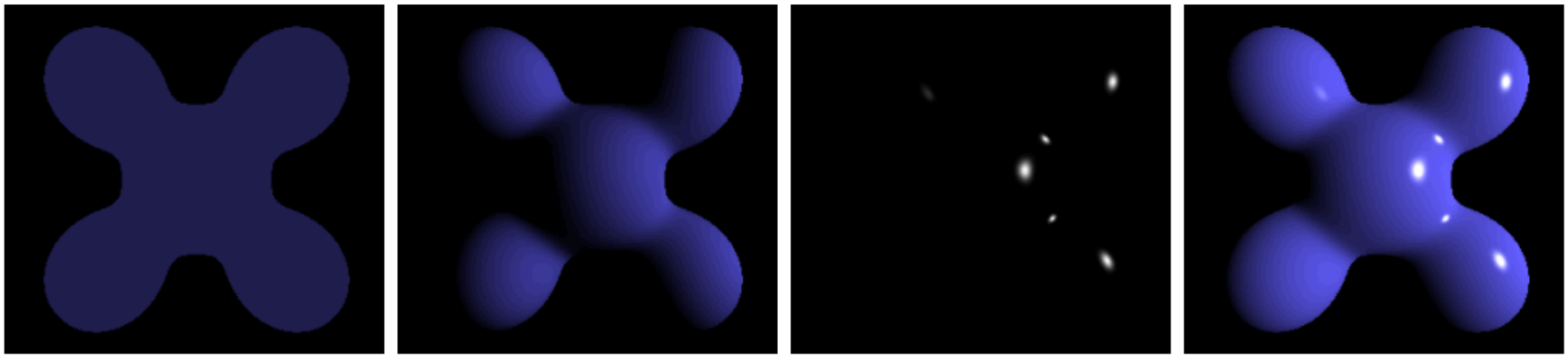
$$I = k_a I_a + \sum_{i=1}^{N} I_i \left(k_d \cos(\theta) + k_s \cos^n(\gamma) \right)$$

ambient light

sum over all light sources

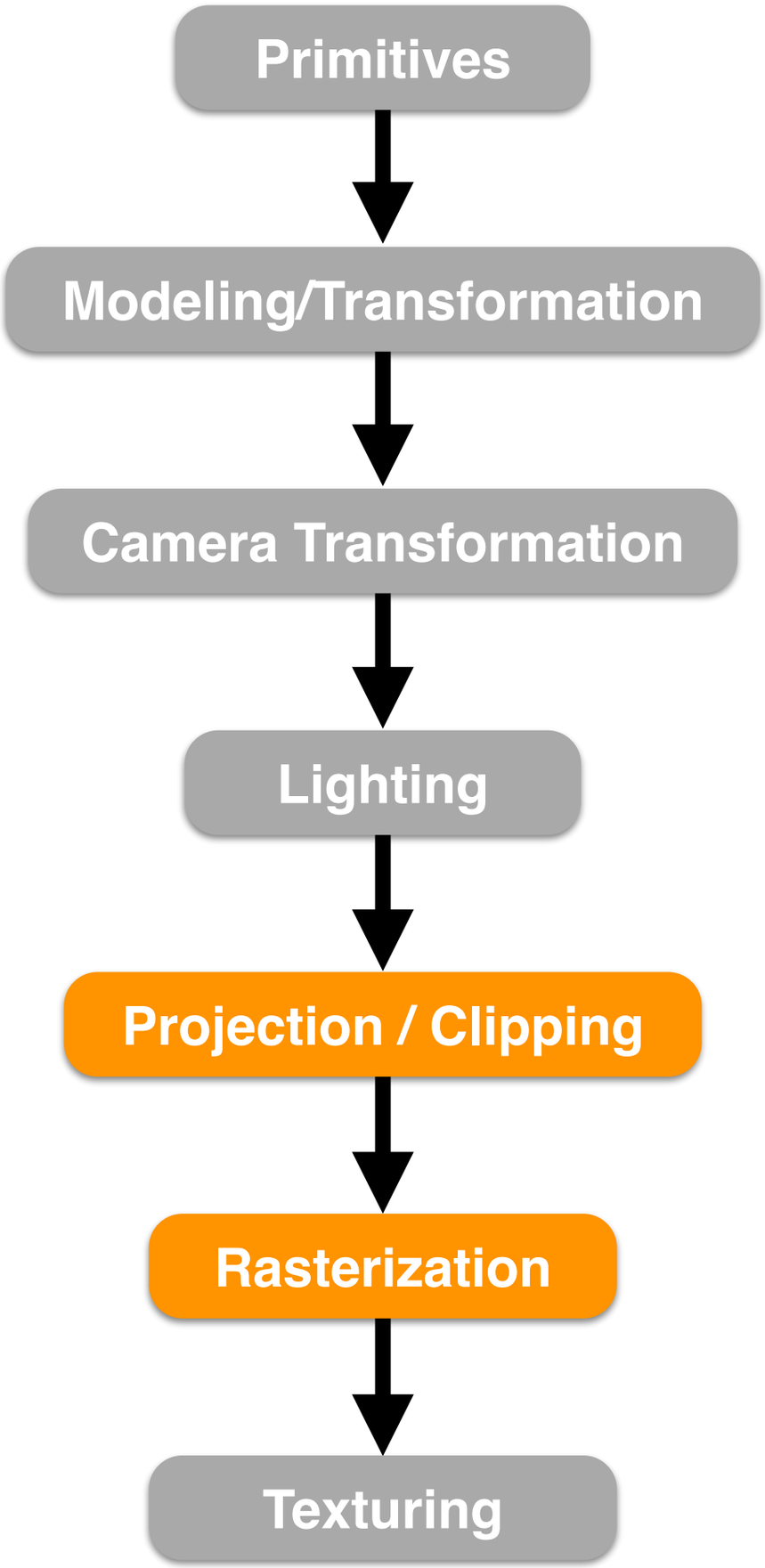


Blinn-Phong Decomposed



Ambient + Diffuse + Specular = Phong Reflection

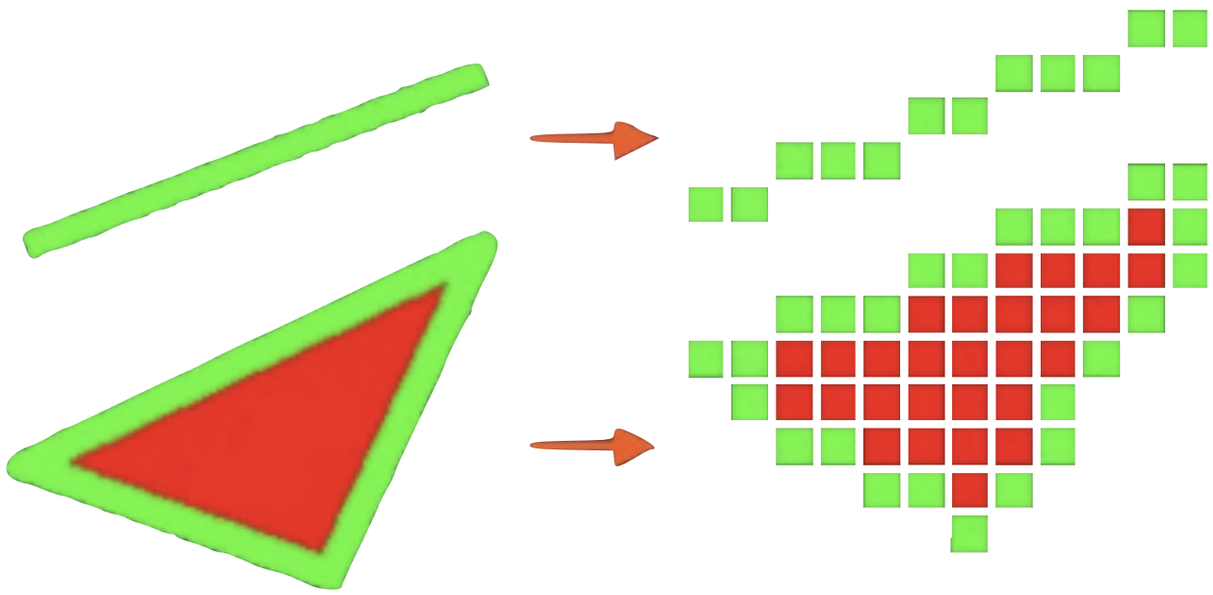
https://en.wikipedia.org/wiki/Phong_shading



Rasterization

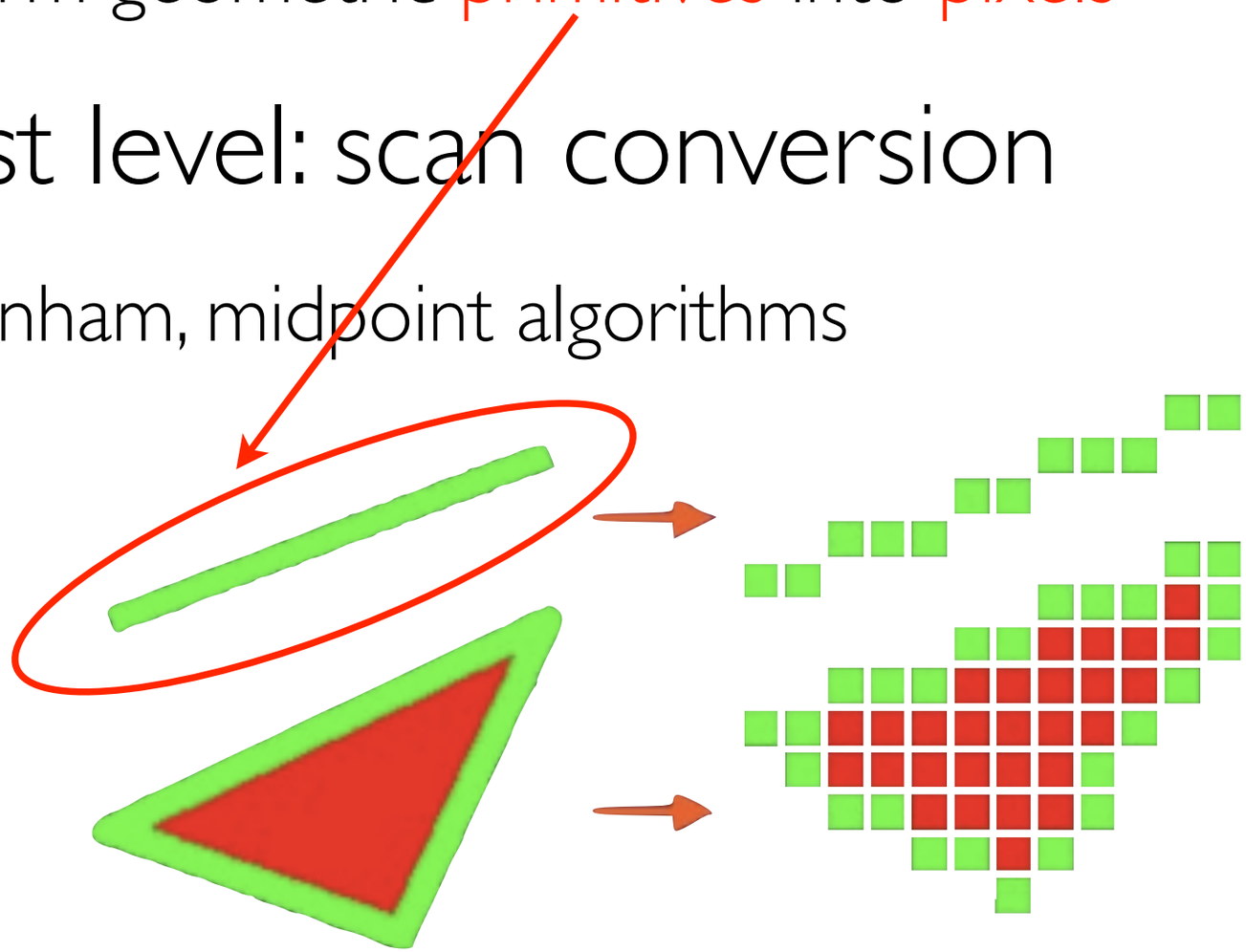
Rasterization

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



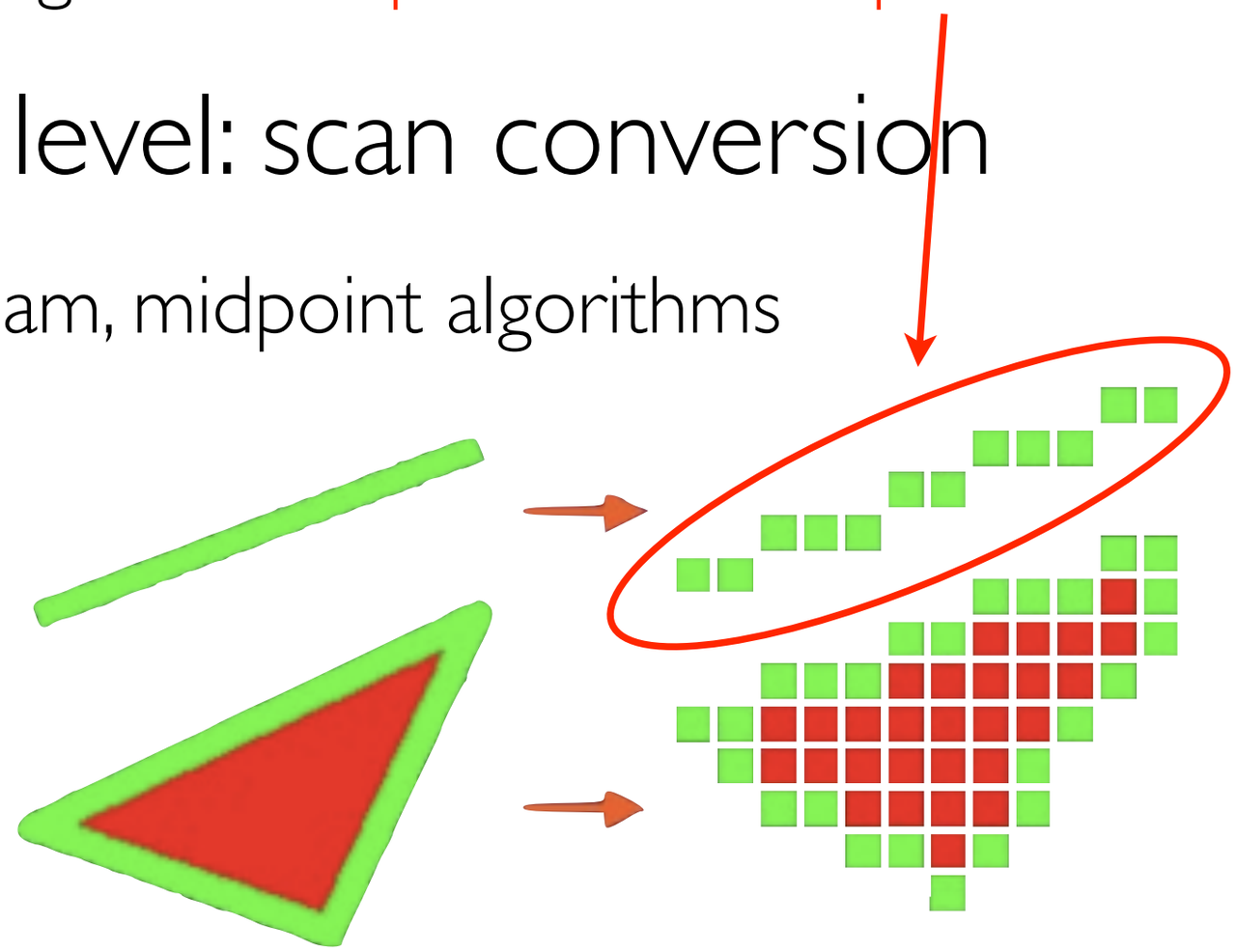
Rasterization

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



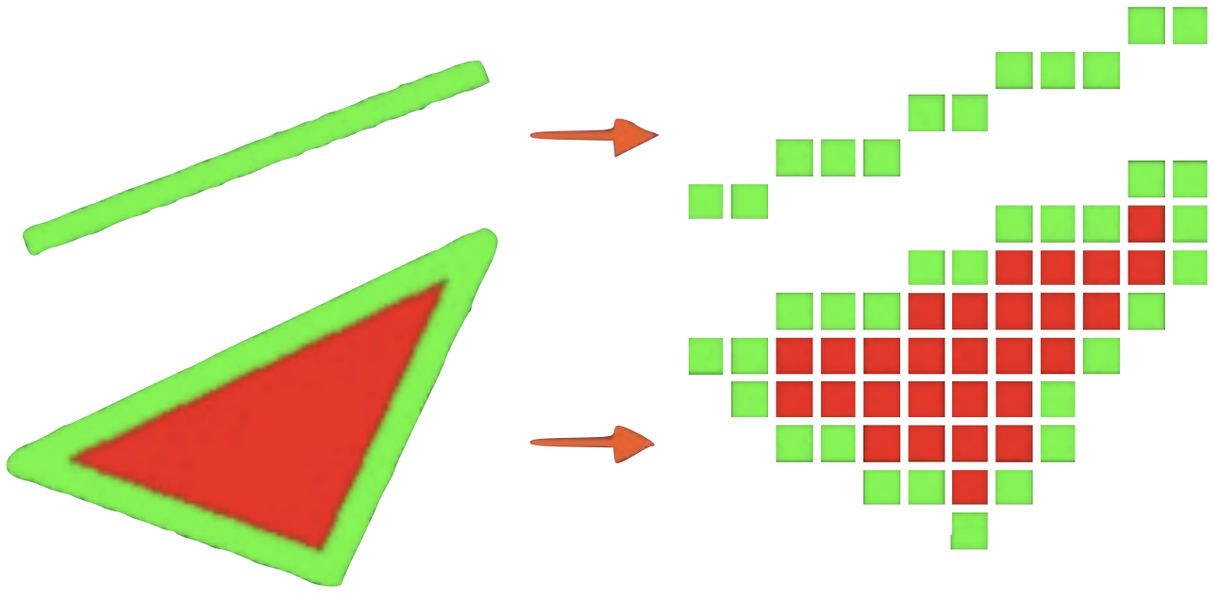
Rasterization

- 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

Z-buffer algorithm

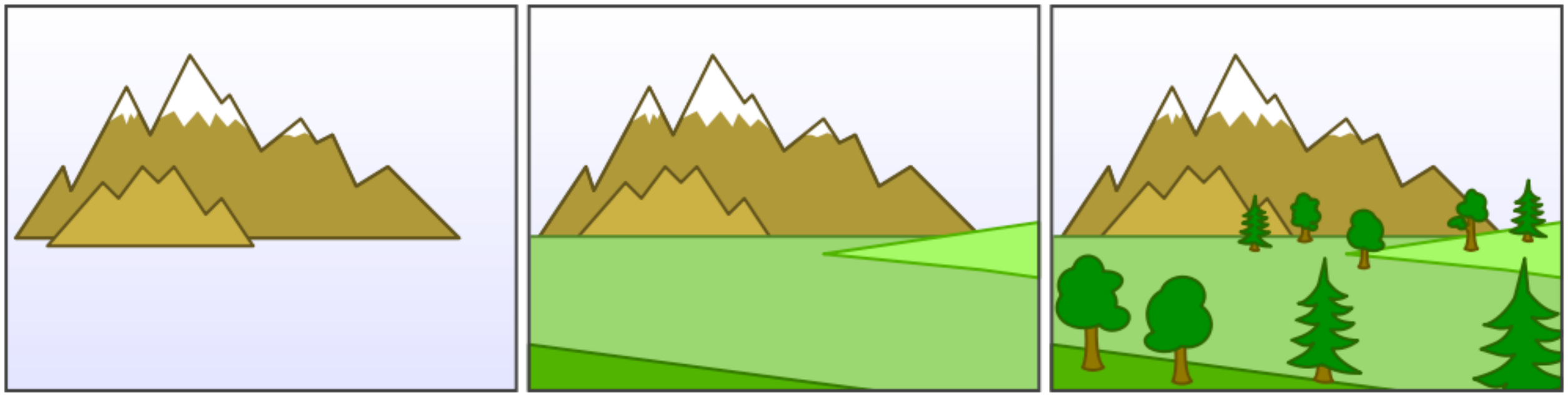
- Maintain a z-buffer of same resolution as frame buffer
- During scan conversion compare z-coord of pixel to be stored with z-coord of current pixel in Z-buffer
- If new pixel is closer, store value in frame buffer and new z-coord in Z-buffer

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

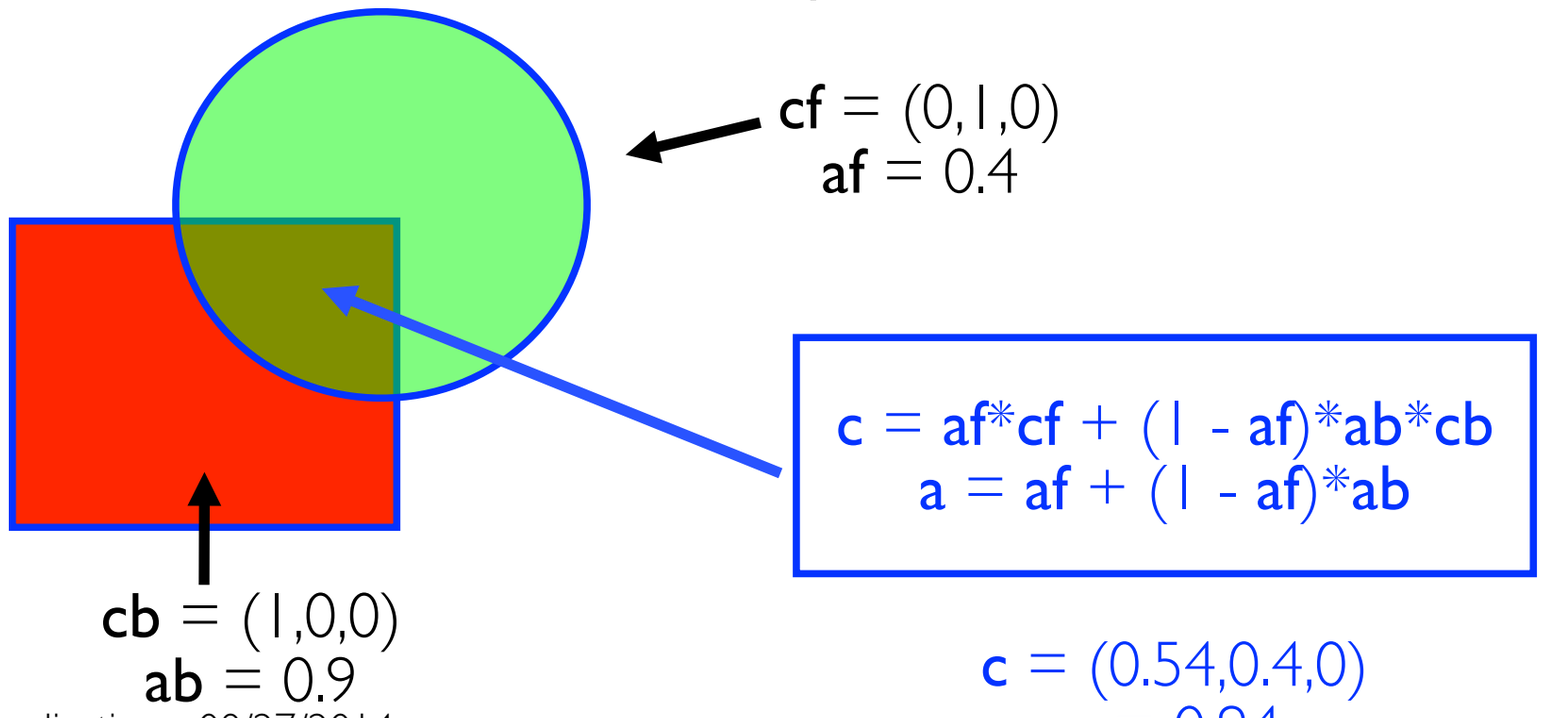
Alpha blending / compositing

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

Fundamental Algorithms

Alpha blending / compositing

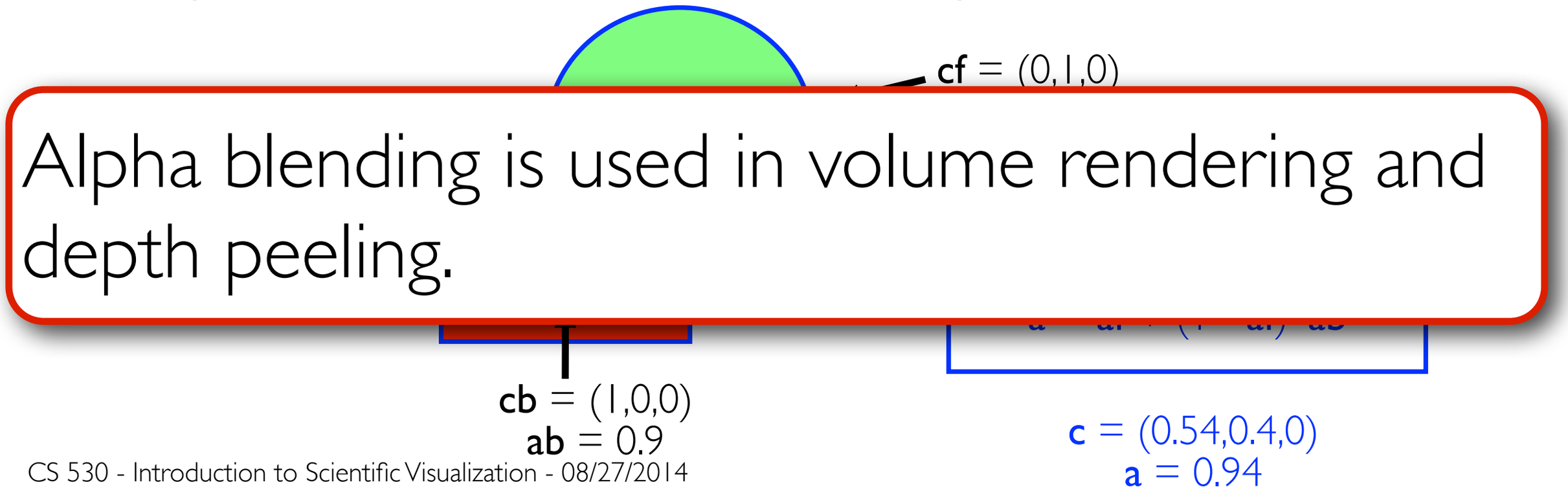
- Approximate visual appearance of semi-transparent object in front of another object
- Implemented with OVER operator



Fundamental Algorithms

Alpha blending / compositing

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



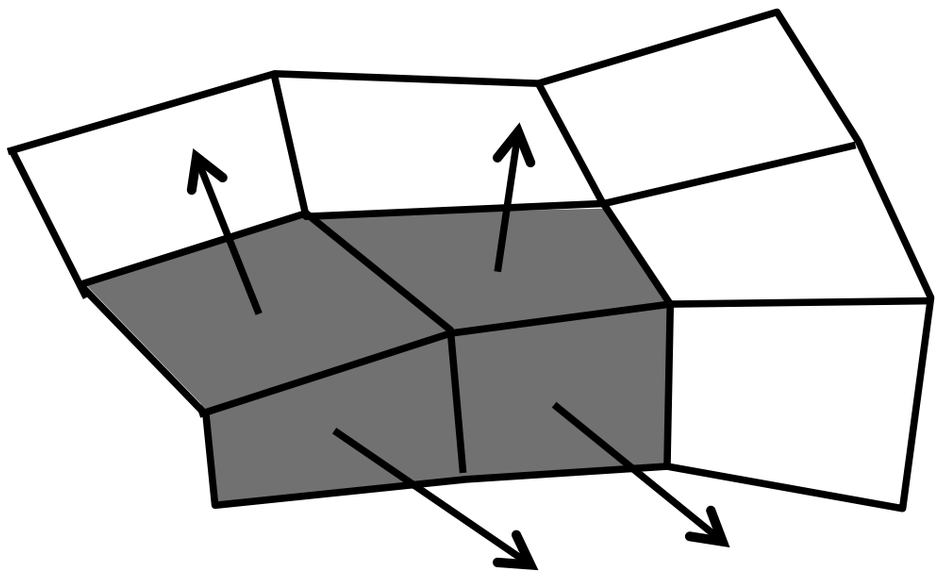
Alpha blending is used in volume rendering and depth peeling.

Shading and Illumination

Back to Shading

Flat shading

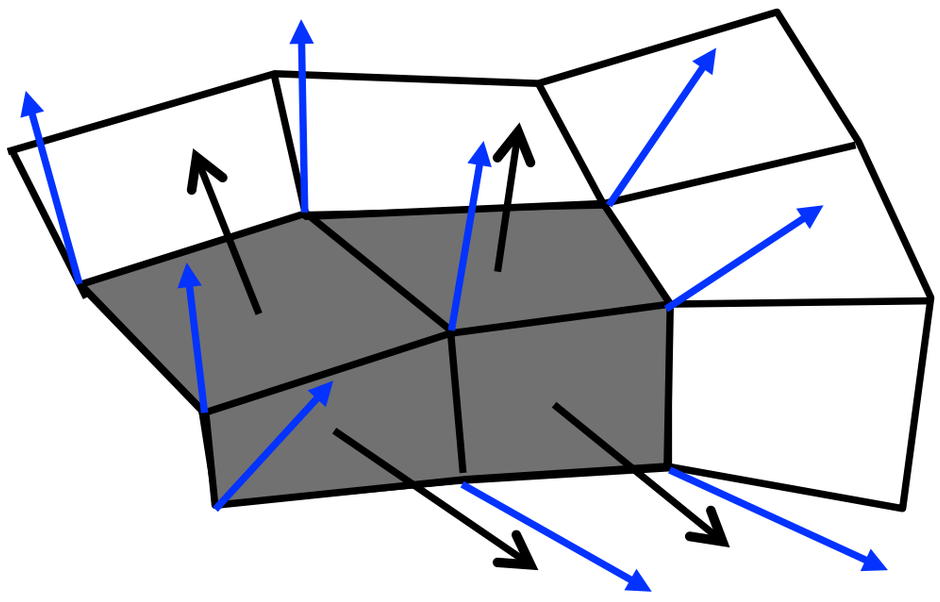
(Color constant per polygon)

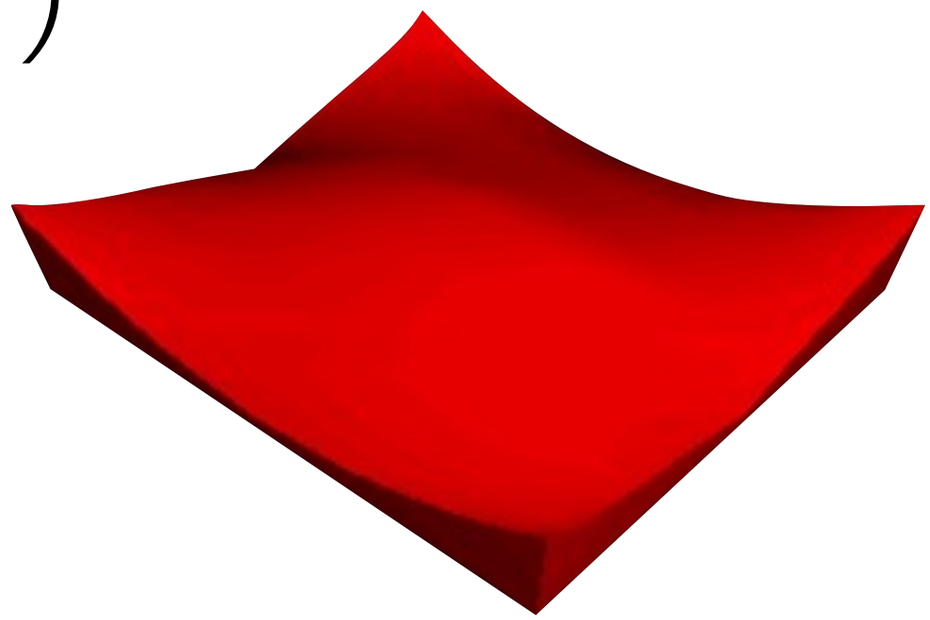


Back to Shading

Gouraud shading (1971)

- Shade vertices first, then interpolate* colors



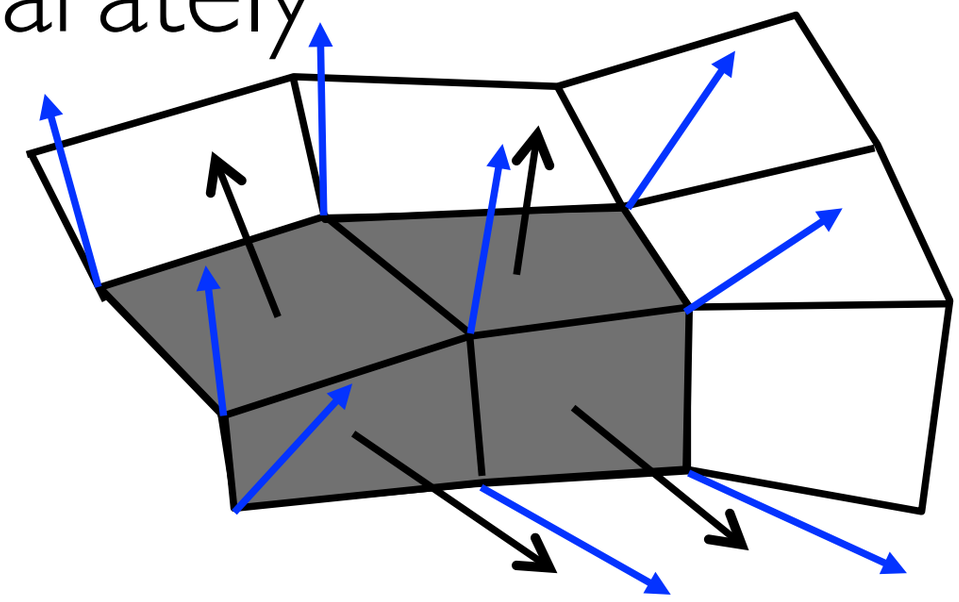


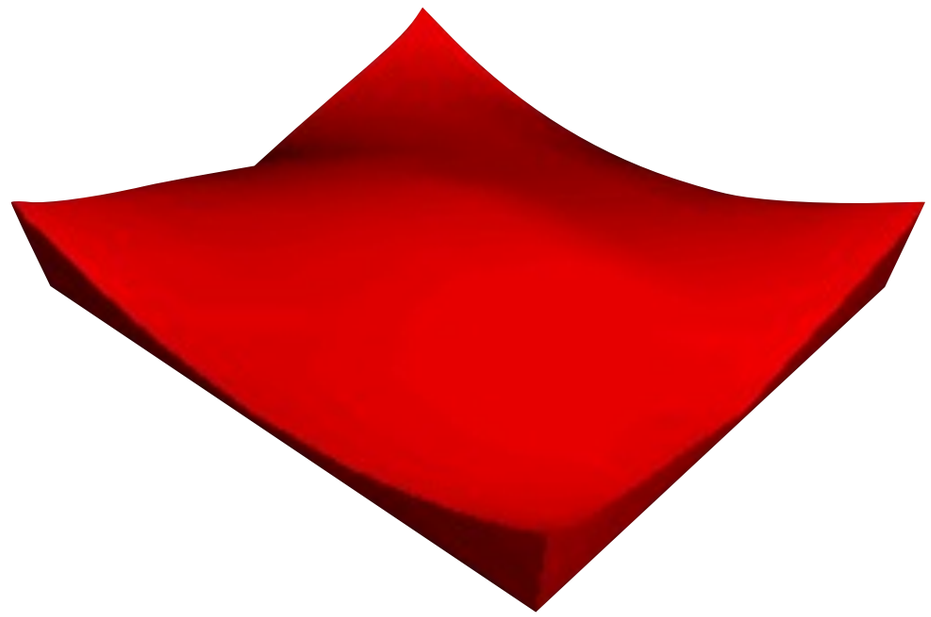
(*) More on interpolation in coming weeks...

Back to Shading

Phong shading

- Interpolate normals and shade every pixel separately





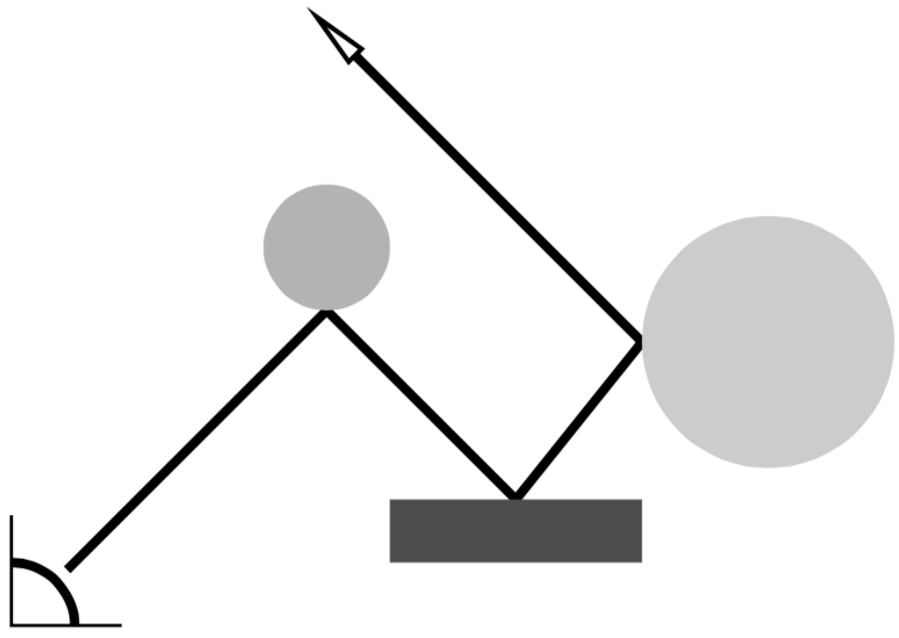
Note: Phong lighting model ≠ Phong shading

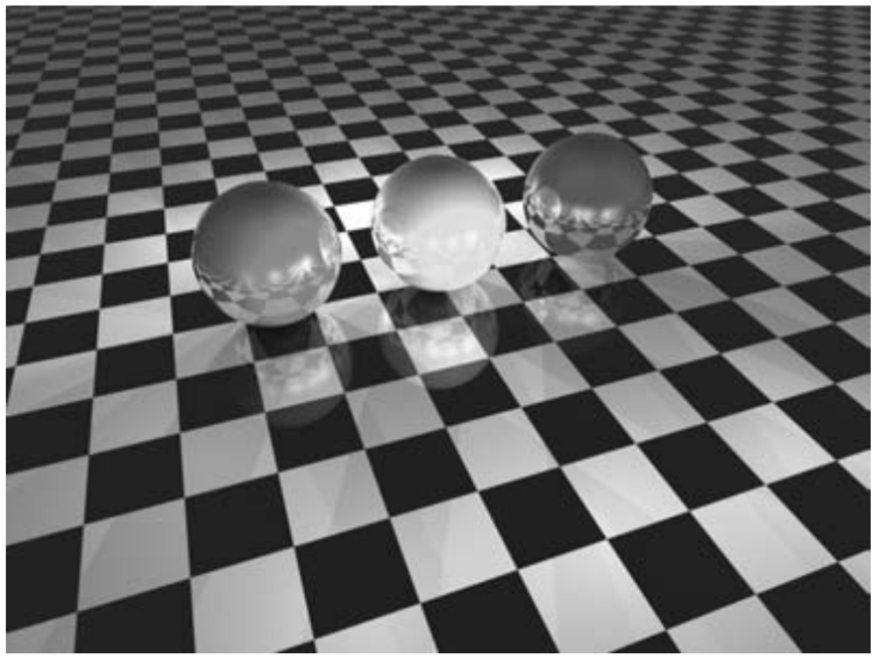
Global Illumination

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

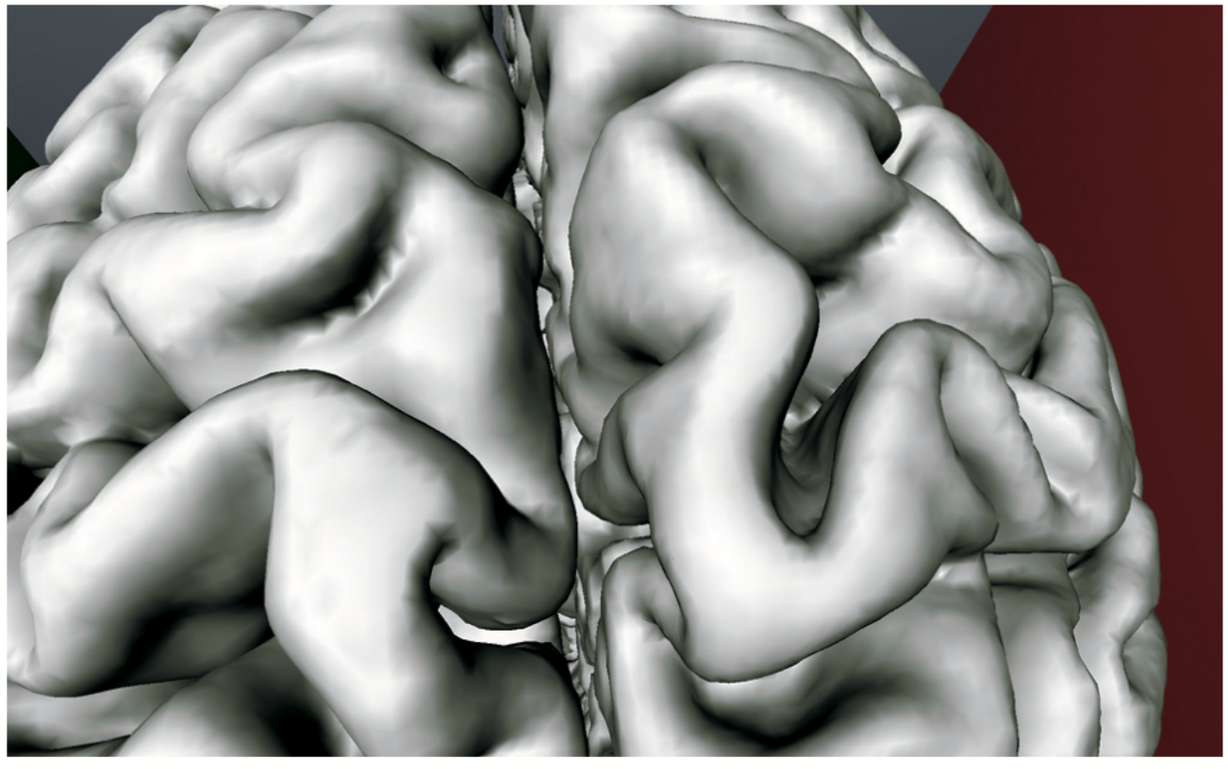
- Global illumination can be achieved by ray tracing — casting rays from the eye that reflect, create shadows, scatter, etc.

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

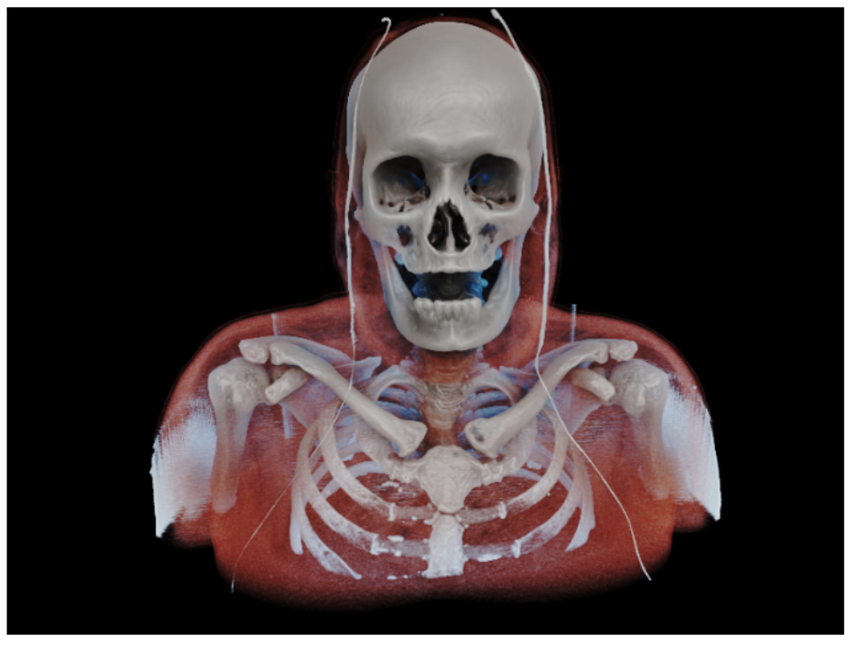


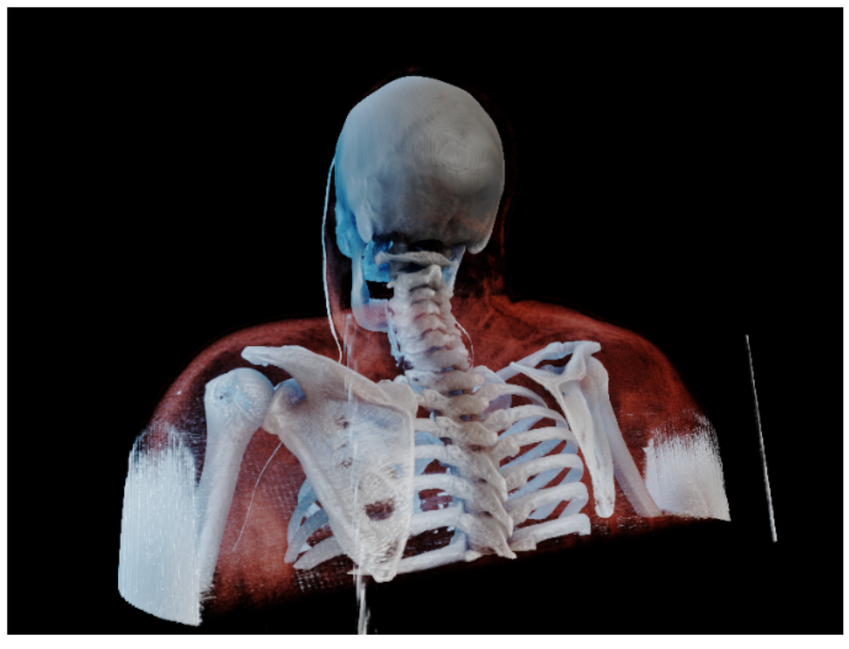


See Kajiya, Blinn, and many more…



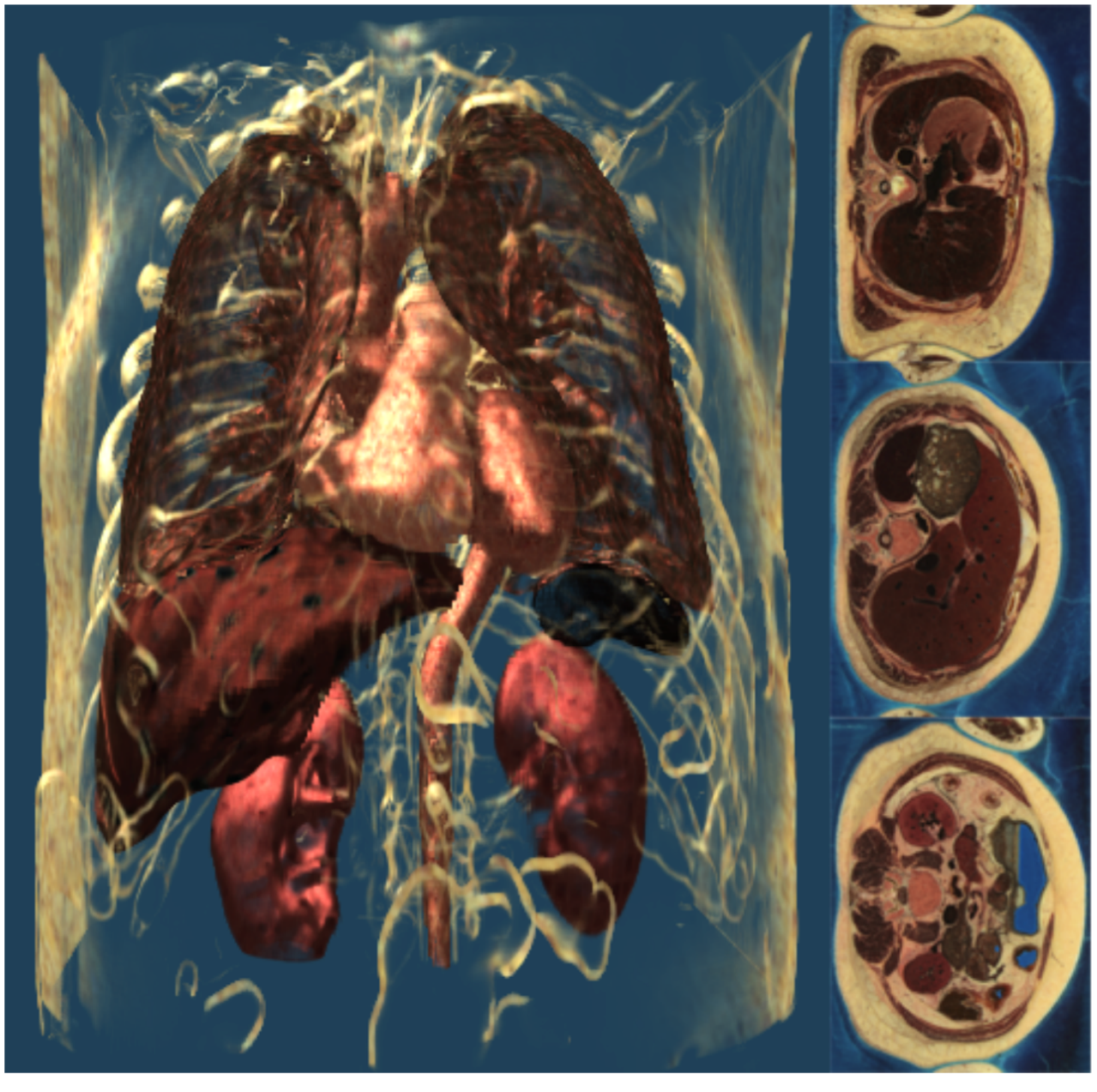
Banks and Beason, 2007

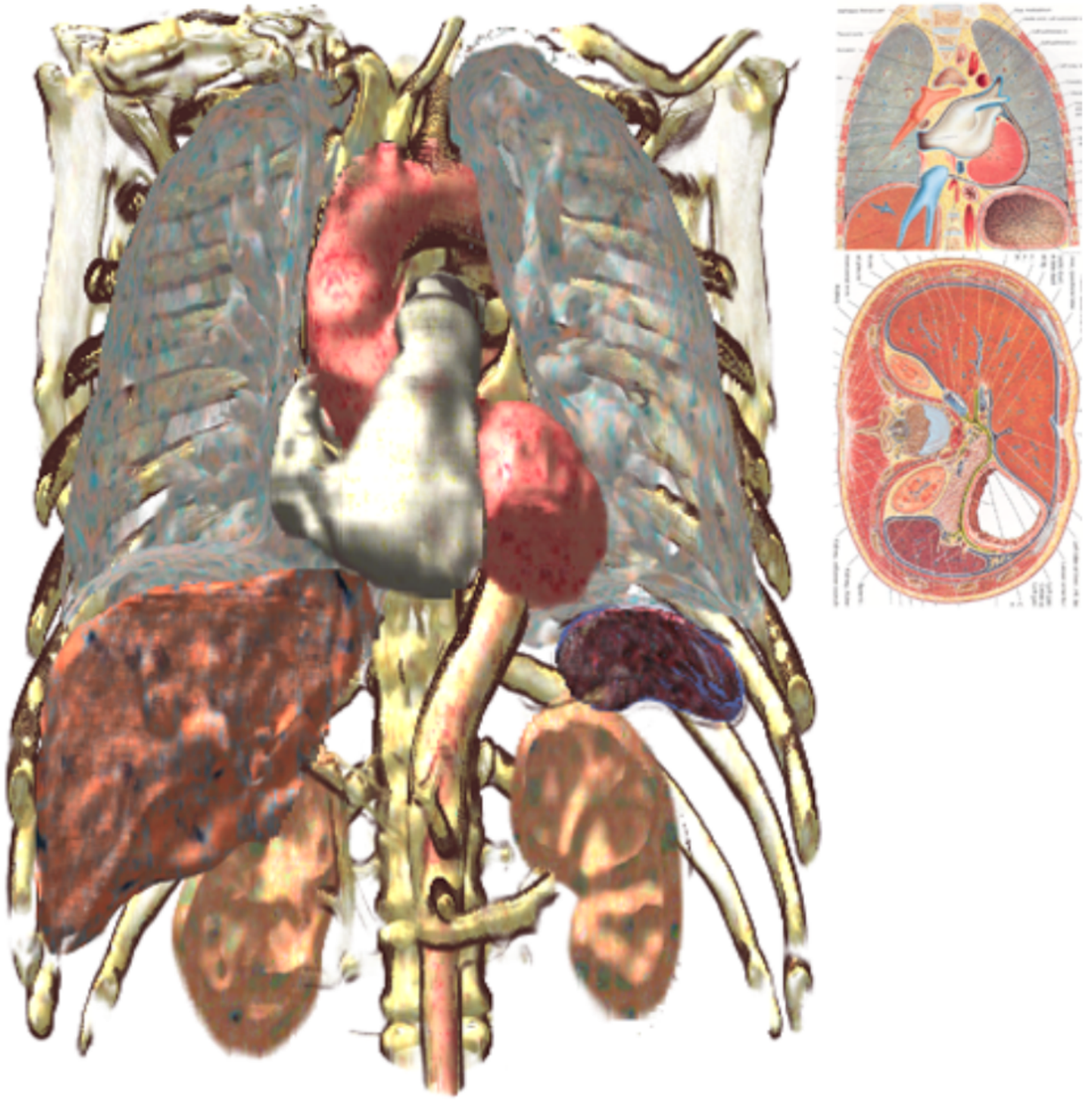




Zhang and Ma, 2013

Nonphotorealistic Rendering (NPR)





Lu and Ebert, 2005

Lec19 Reading

- Munzner, 8.4.2

- Display of Surfaces from Volume Data. Marc Levoy, IEEE Computer Graphics and Applications, 8(3), 29-37, 1988.

Reminder
Assignment 04

Assigned: Monday, March 13
Due: Monday, March 27, 4:59:59 pm

Reminder
Project Milestone 02

Assigned: Wednesday, February 22
Due: Wednesday, March 29, 4:59:59 pm