CS 530 Visualization

Computer Graphics Primer

January 9, 2013

Introduction

- Computer Graphics & Visualization
  - Visualization methods transform data into *primitives* that are *rendered* using CG techniques
- This is not a CG class!
- Today's objective: understand basic principles of CG necessary to properly manipulate a visualization library like VTK

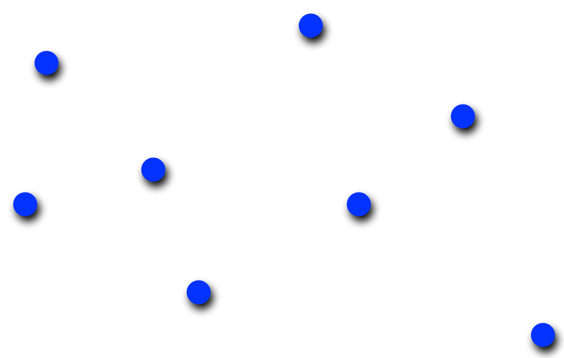
Outline

- Graphics Primitives
- Colors
- Shading
- Rasterization
- Projections and Cameras
- Fundamental Algorithms

Graphics Primitives

Points

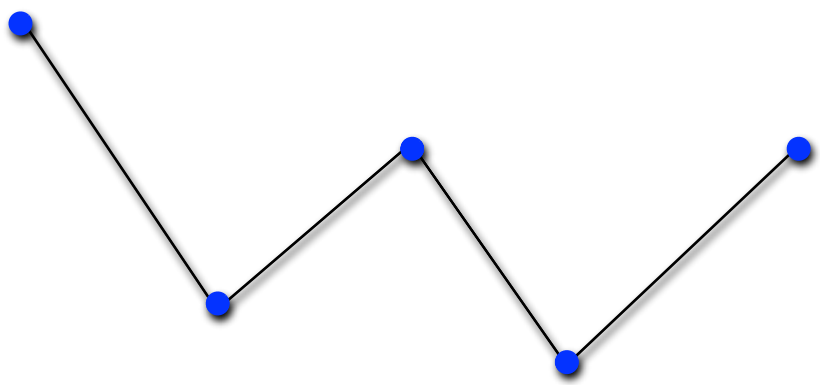
- 2D, 3D coordinates
- 0-dimensional objects



Graphics Primitives

Lines

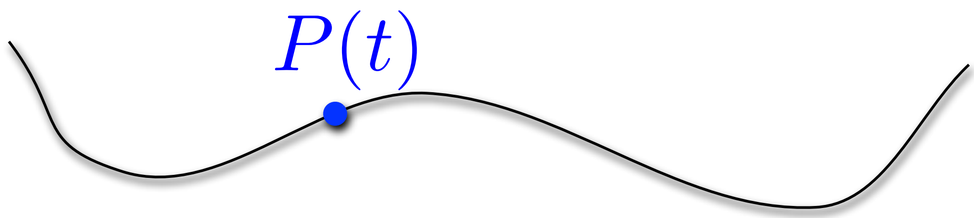
- 1-dimensional objects
- Polygonal description (*"polyline"*)
  - piecewise linear



Graphics Primitives

Lines

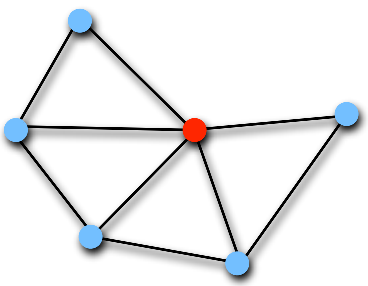
- 1-dimensional objects
- Parametric description
  - $P : t \in I \mapsto P(t) \in \mathbb{R}^n$
  - E.g. splines



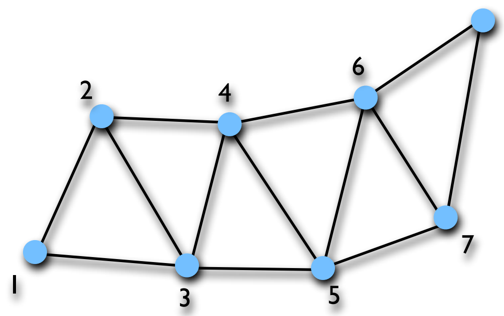
Graphics Primitives

Surfaces

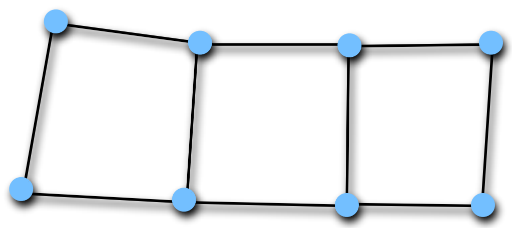
- 2-dimensional objects
- Polygonal description
    - Important topologies (special cases)



triangle fan



triangle strip

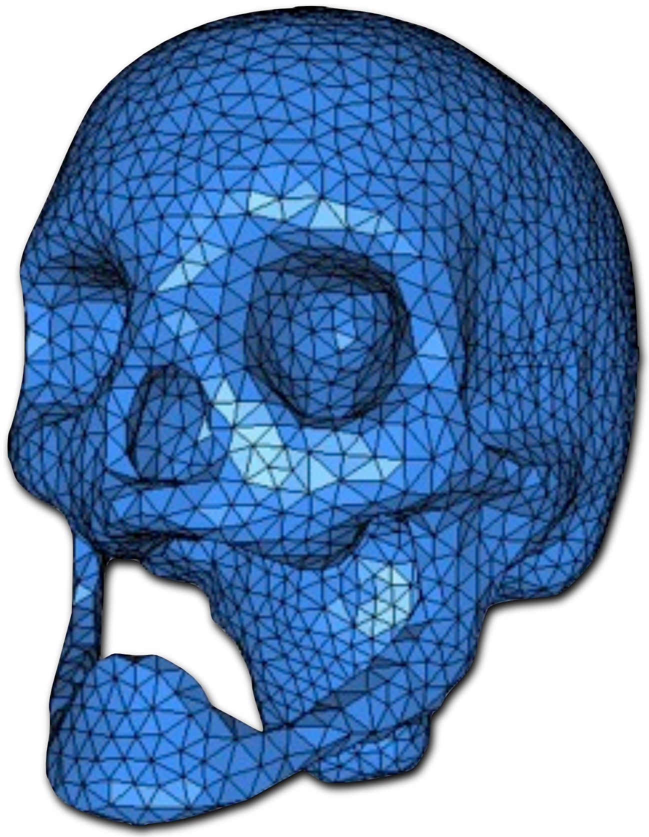


quad strip

Graphics Primitives

Surfaces

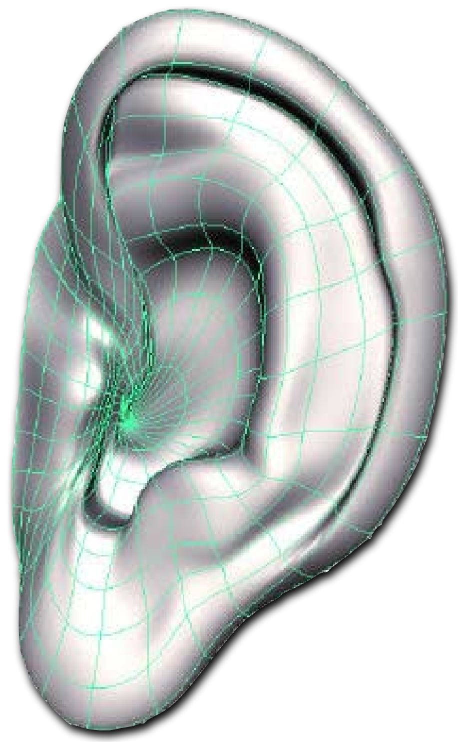
- 2-dimensional objects
- Polygonal description
  - typically: triangle mesh (piecewise linear)



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
- ...

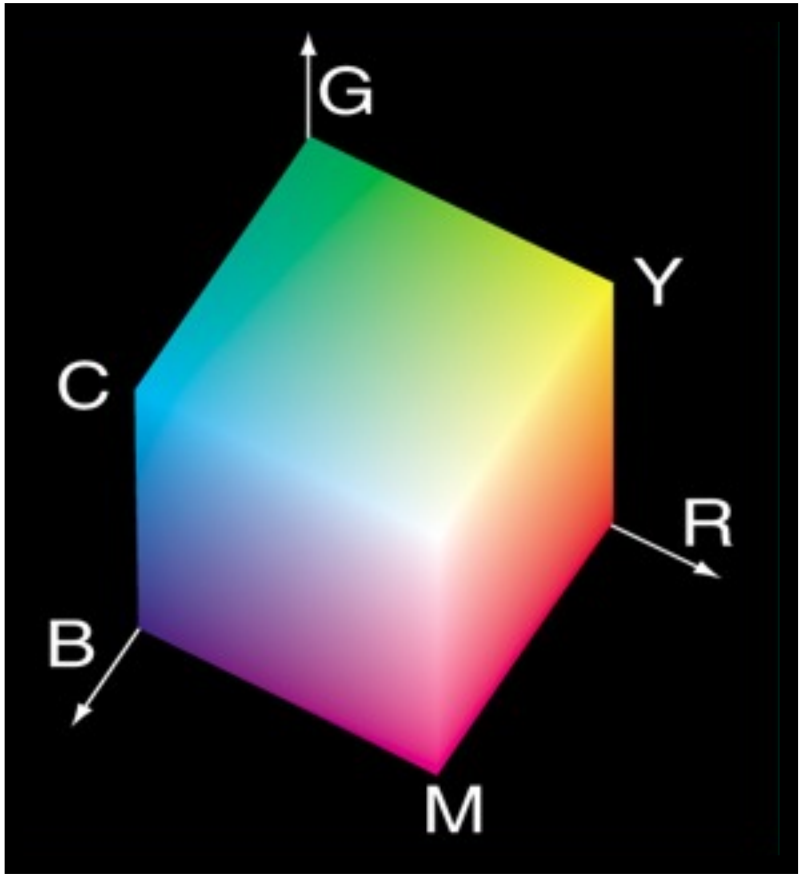
Outline

- Graphics Primitives
- Colors
- Shading
- Rasterization
- Projections and Cameras
- Fundamental Algorithms

Colors

- RGB system

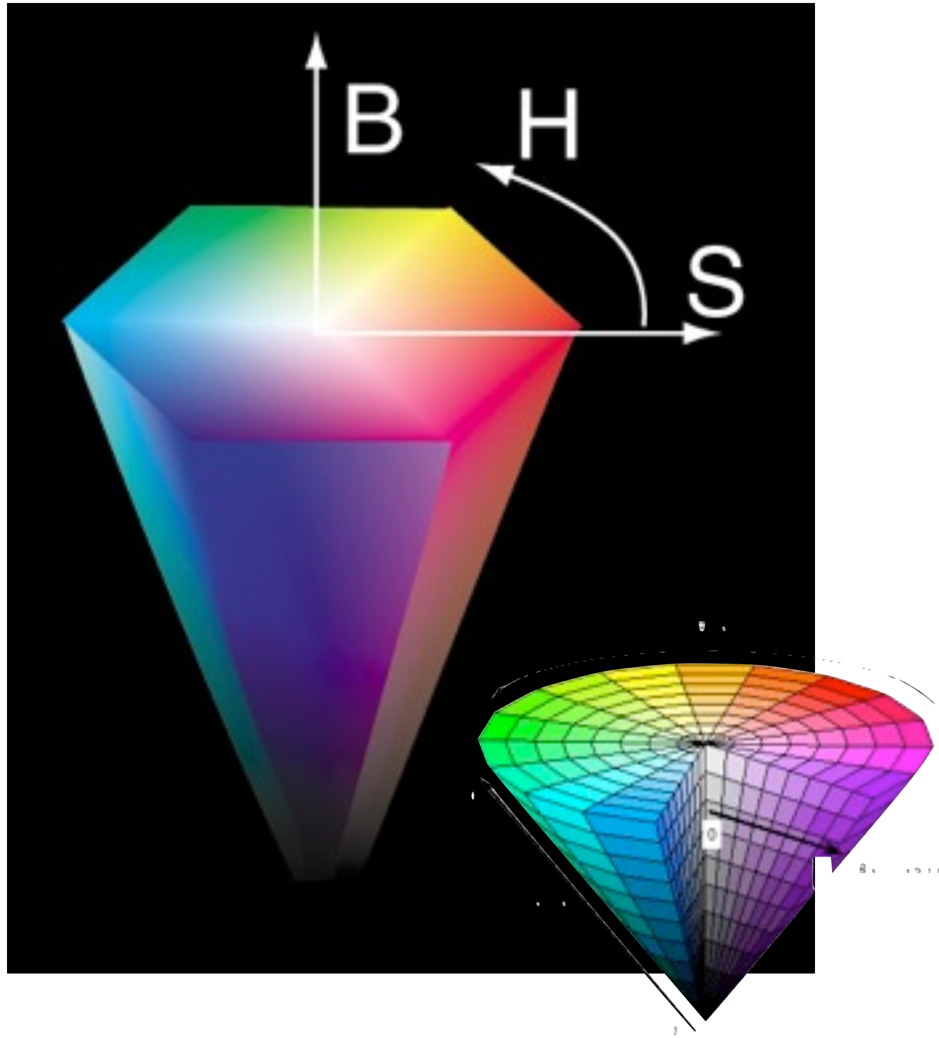
Black	0,0,0
White	1,1,1
Red	1,0,0
Green	0,1,0
Blue	0,0,1
Yellow	1,1,0
Cyan	0,1,1
Magenta	1,0,1
Sky Blue	1/2,1/2, 1



Colors

- HSV system

Black	*,*,0
White	*,0,1
Red	0,1,1
Green	1/3, 1,1
Blue	2/3,1,1
Yellow	1/6,1,1
Cyan	1/2,1,1
Magenta	5/6,1,1
Sky Blue	2/3,1/2,1



Colors

- Both RGB and HSV encode colors with 3 scalars
- Simplified models of human color perception
- More on the topic in the following weeks!

Outline

- Graphics Primitives
- Colors
- Shading
- Rasterization
- Projections and Cameras
- Fundamental Algorithms

Shading

- How we show the shape of things
- Role of shading: create colors as a function of:
  - surface properties
  - surface normals
  - lights
- Rich subject (but we are doing Vis!)
- Surfaces show information, lights show surfaces, shading controls how

Shading

Phong lighting model (1975)

- Specular reflection
- Diffuse reflection
- Ambient reflection

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

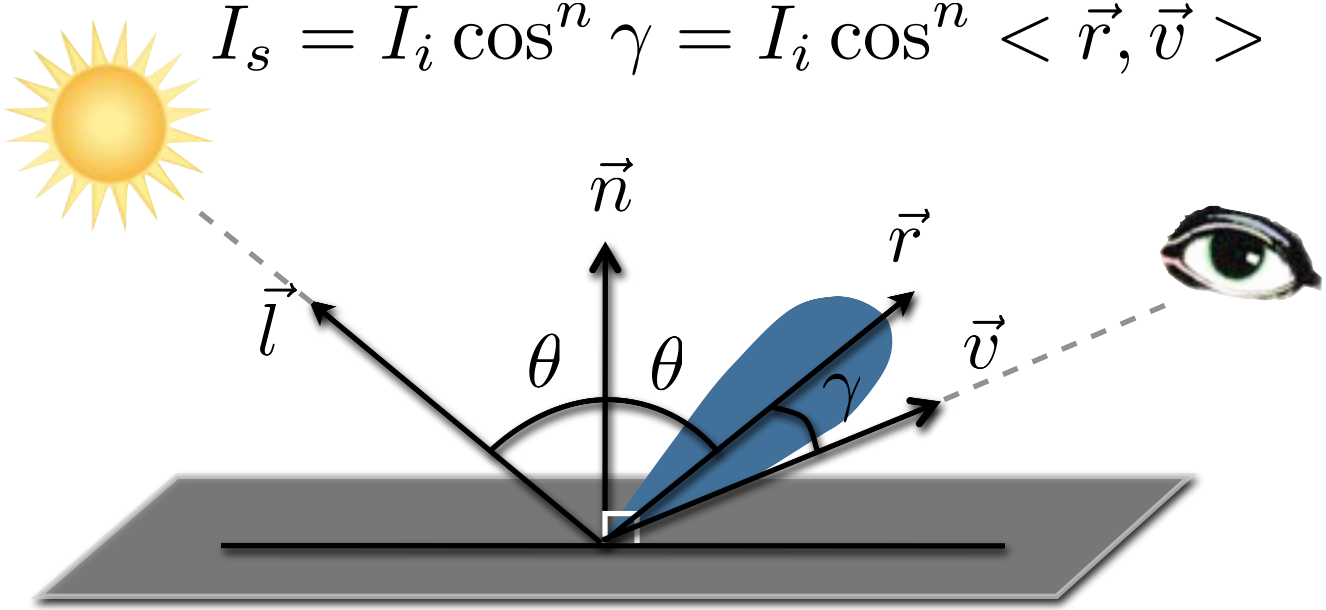
relative contributions

Shading

Phong lighting model (1975)

- Specular reflection : mirror-like surfaces

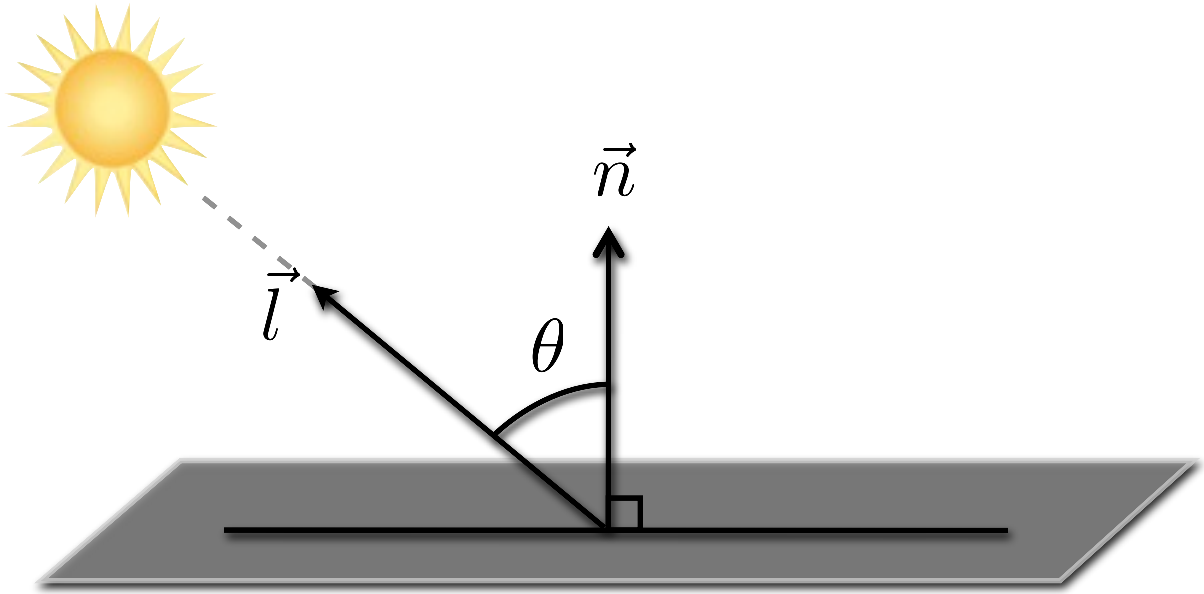
$$I_s = I_i \cos^n \gamma = I_i \cos^n < \vec{r}, \vec{v} >$$

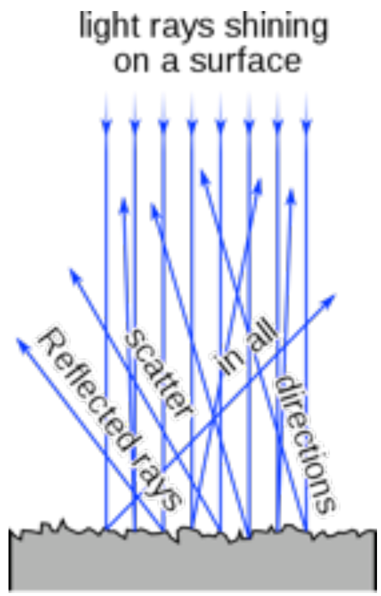


Shading

Phong lighting model (1975)

- Diffuse reflection : non-shiny surfaces

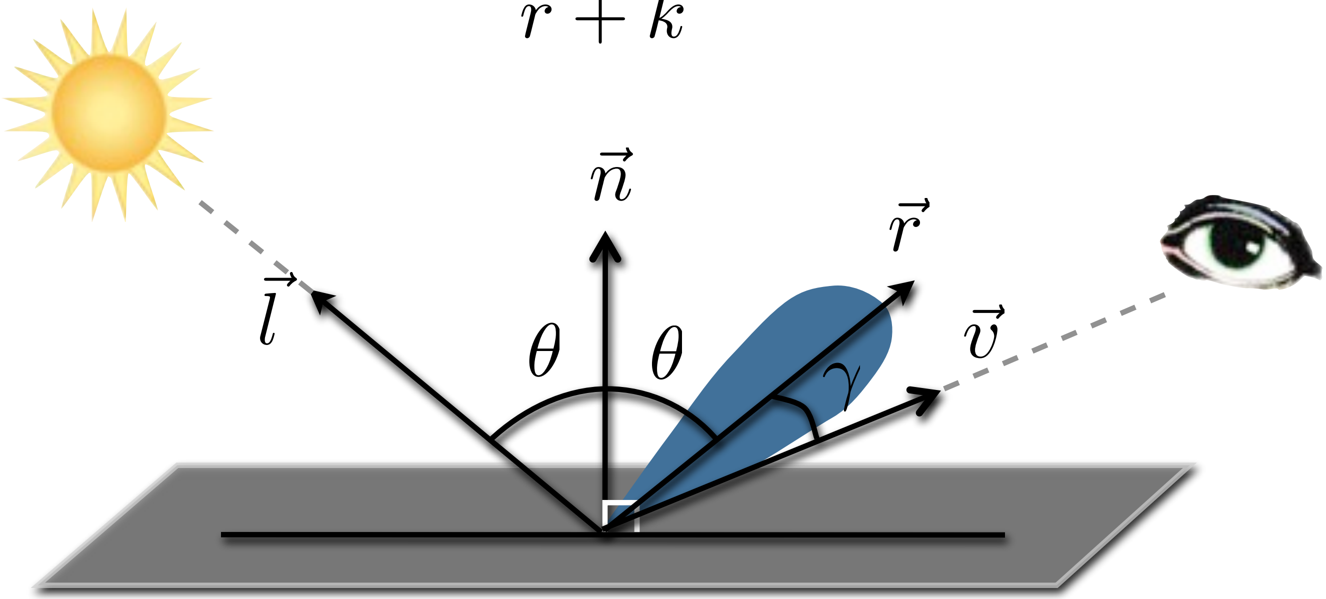




Shading

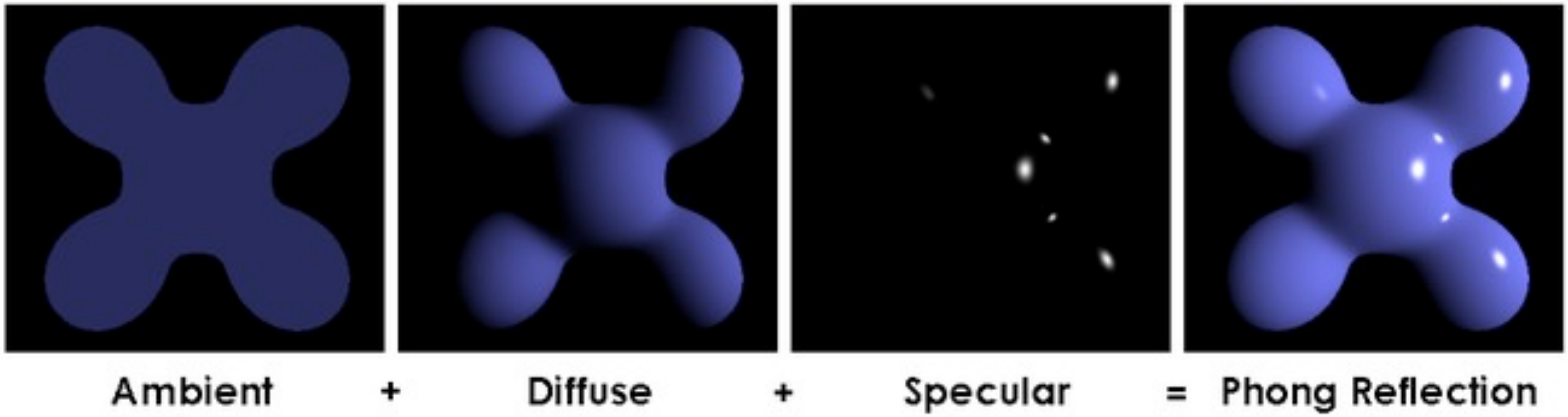
Phong lighting model (1975)

$$I = I_i \frac{(k_{s\alpha} \cos^n < \vec{r}, \vec{v} > + k_{d\alpha} \cos < \vec{r}, \vec{n} >)}{r + k} + k_{a\alpha} I_a$$



Shading

Phong shading model



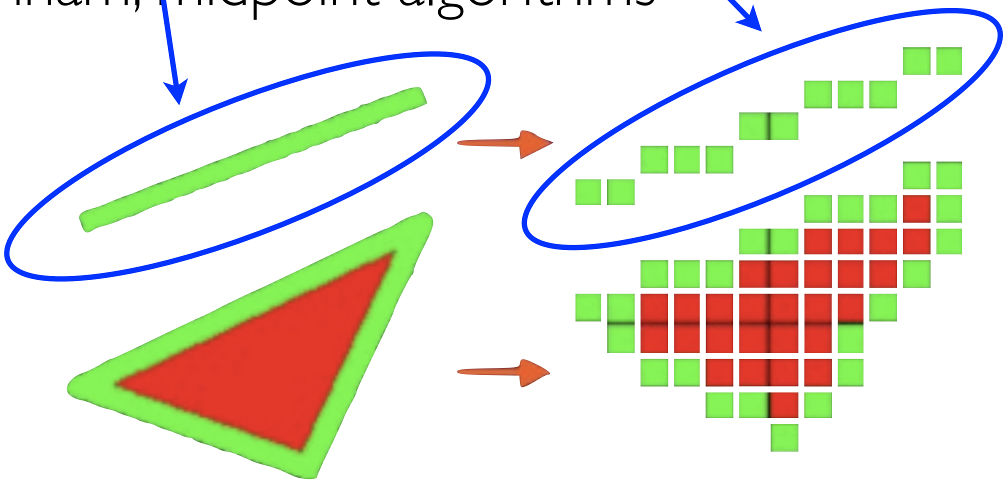
http://en.wikipedia.org/wiki/Phong_shading

Outline

- Graphics Primitives
- Colors
- Shading
- Rasterization
- Projections and Cameras
- Fundamental Algorithms

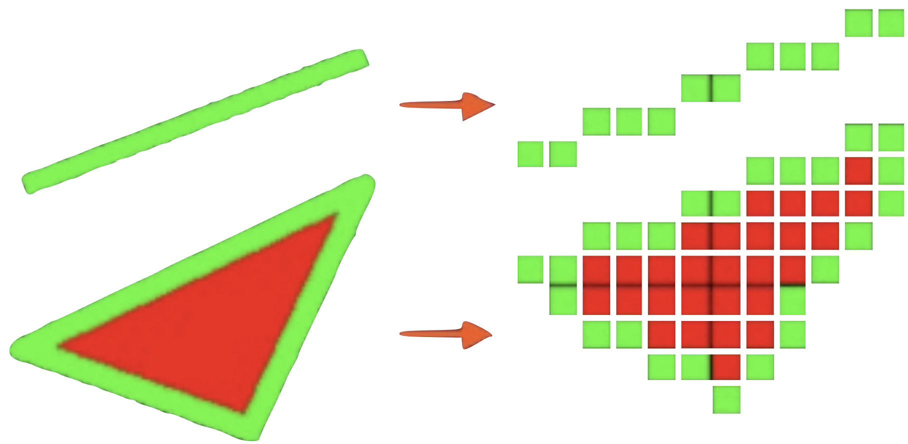
Rasterization

- Putting shaded polygons on screen
  - from primitives to *pixels*
- Lowest level: scan conversion
  - Bresenham, midpoint algorithms



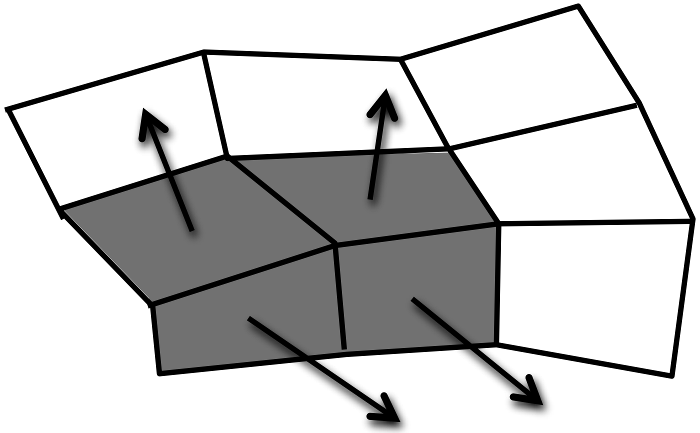
Rasterization

- Putting shaded polygons on screen
  - from primitives to *pixels*
- OpenGL (graphics library) takes care of such operations (under the hood in VTK)



Back to Shading

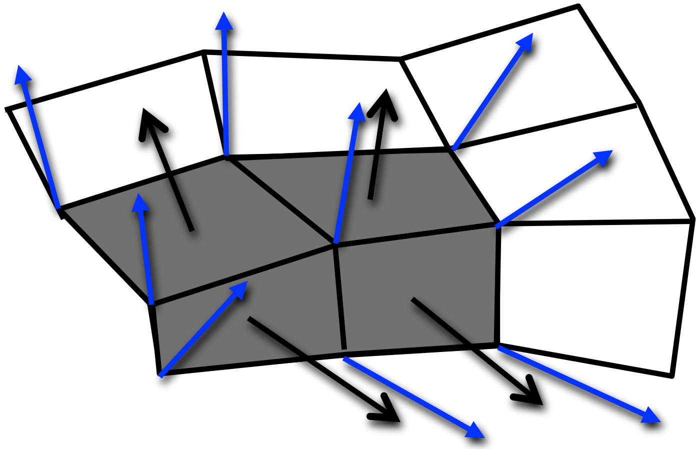
Flat shading

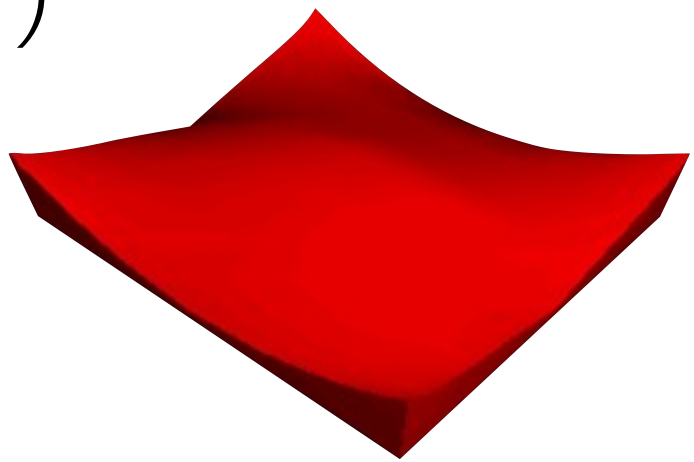


Back to Shading

Gouraud shading (1971)

- Shade vertices first, then interpolate

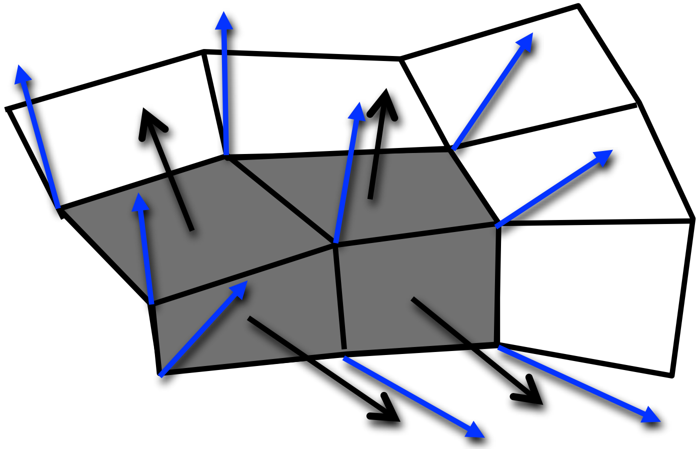


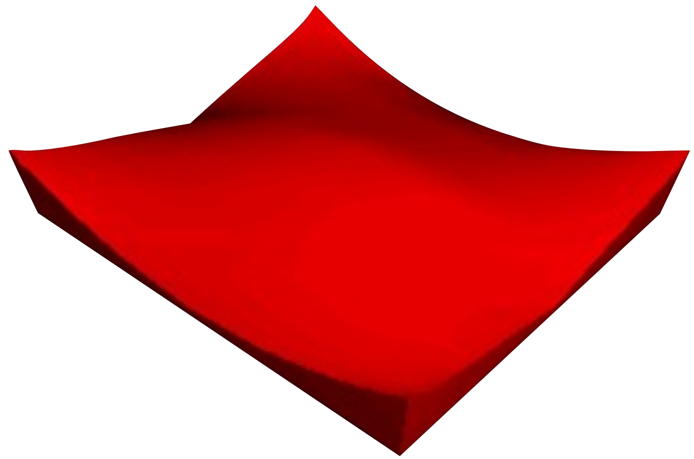


Back to Shading

Phong shading

- Interpolate normals and shade every pixel





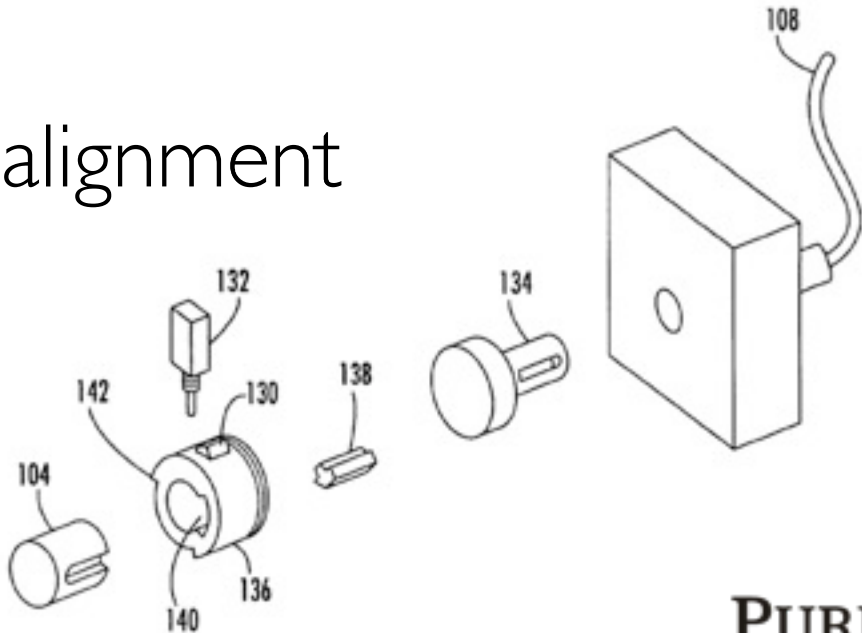
Outline

- Graphics Primitives
- Colors
- Shading
- Rasterization
- Projections and Cameras
- Fundamental Algorithms

Projections

Orthogonal projection

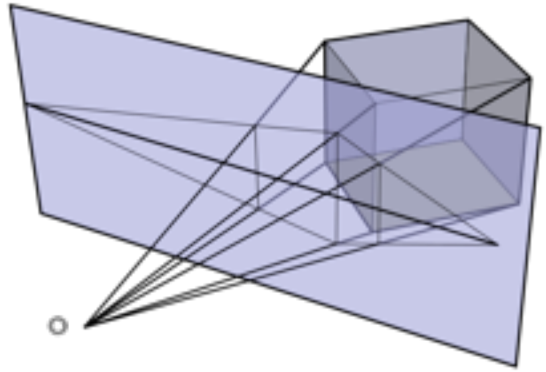
- parallel lines remain parallel
- objects do not change size as they get closer
- good for checking alignment



PURDUE
UNIVERSITY

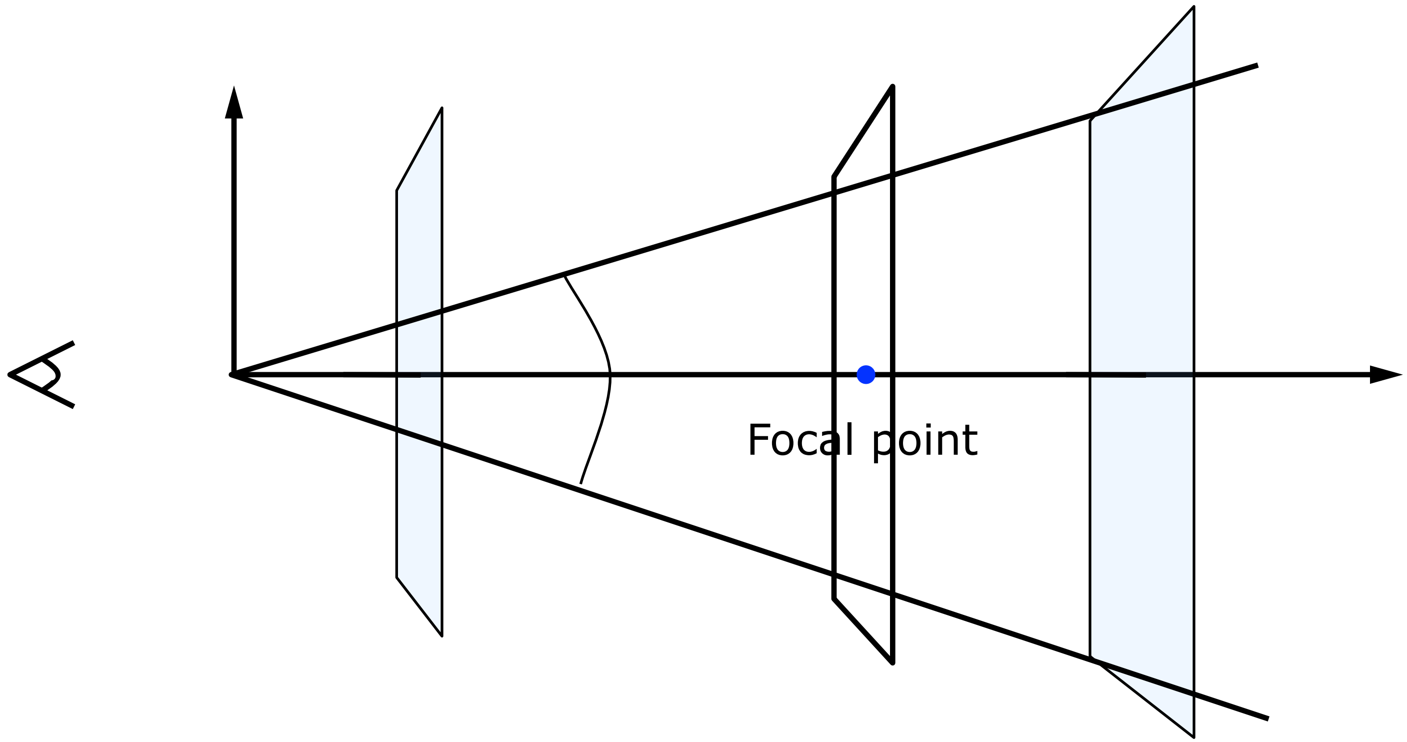
Projections

Perspective projection



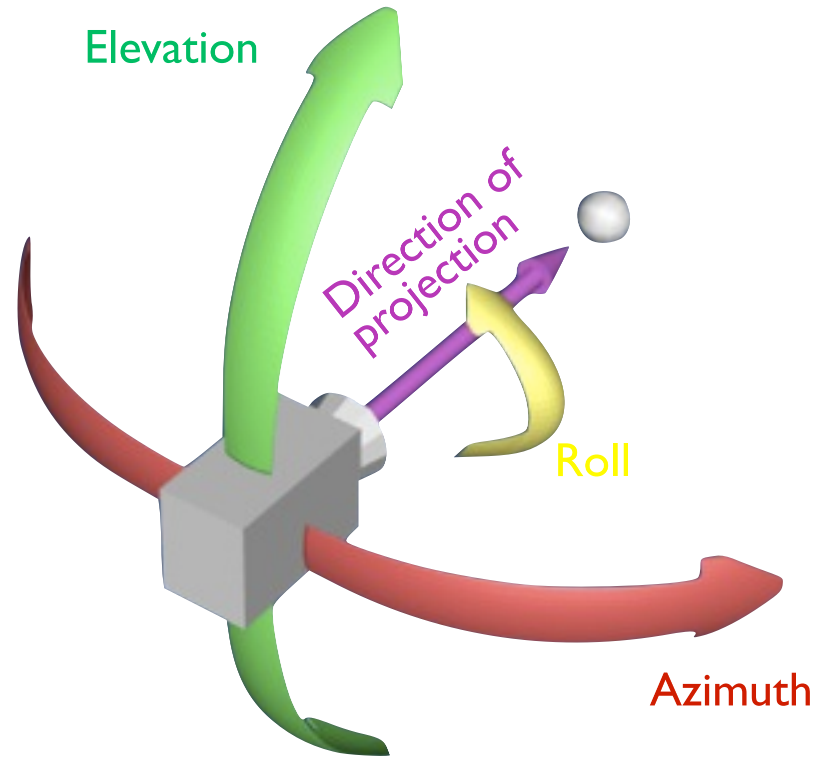
- parallel lines do not necessarily remain parallel
- objects get larger as get closer
- fly-through realism

Camera



Camera

- Degrees of freedom



Outline

- Graphics Primitives
- Colors
- Shading
- Rasterization
- Projections and Cameras
- Fundamental Algorithms

Fundamental Algorithms

- Back-face culling
- Depth sort
- Z-buffer
- Ray tracing

Fundamental Algorithms

Back-face culling

- Determine whether a polygon is visible (and should be rendered)
- Check polygon normal against viewing direction of the camera

Fundamentals Algorithms

Depth-sort algorithm

1. Sort all polygons from near to far (z-coordinates)
2. Resolve ambiguities (overlapping in z)
3. Scan convert each polygon in ascending order of z coord. (back to front)

Fundamentals 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

Fundamentals Algorithms

Z-buffer algorithm

Fundamental Algorithms

Ray-tracing algorithm

- Send ray from pixel into view volume
- Determine which surface is struck
- Combined with lighting algorithm