Introduction

Computer Graphics vs. Visualization

“Visualization methods transform data into primitives that are rendered using CG techniques”

Today’s objective: understand basic concepts of CG necessary to use a visualization library like VTK
Outline

Graphics Primitives
Colors
Shading
Rasterization
Cameras
Fundamental Techniques
Graphics Primitives

Points

Positions in space

2D($x, y$), 3D($x, y, z$) 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

Triangle mesh (piecewise linear)
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

Cameras

Fundamental Techniques
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

RGB system encodes colors with 3 scalars

Simplified models of human color perception

More on the topic next week!
Outline

Graphics Primitives
Colors
Shading
Rasterization
Cameras
Fundamental Techniques
Shading reveals the shape of 3D objects through their interaction with light.

Shading creates colors as a function of:

- surface properties
- surface normals
- lights

Only covering basics here.

Surfaces show information, lights show surfaces, shading controls how.
Shading

Phong lighting model (1975): Light reflected by surface is combination of:

- Specular reflection
- Diffuse reflection
- Ambient reflection
Shading

Phong lighting model (1975): Light reflected by surface is combination of:

Specular reflection

Diffuse reflection

Ambient reflection
Shading

Phong lighting model (1975): Light reflected by surface is combination of:

Specular reflection
Diffuse reflection
Ambient reflection
Phong lighting model (1975):

- Light reflected by surface is combination of:
  - Specular reflection
  - Diffuse reflection
  - Ambient reflection

Shading

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

Phong lighting model: Ambient reflection
Shading

Phong lighting model: Ambient reflection

Light intensity per light source and per color channel

\[ I = k_s I_s \]
Shading

Phong lighting model: Ambient reflection

Light intensity per light source and per color channel

\[ I = k_s I_s \]

relative contributions (material specific)
Shading

Phong lighting model: 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

\[
\vec{l} \cdot \vec{n} \cdot \vec{r} \cdot \vec{v}
\]

\[\theta \quad \theta\]
Shading

Phong lighting model: Specular Reflection

\[ I_s = I_i \cos^n \gamma = I_i \cos^n \langle \vec{r}, \vec{v} \rangle \]
Shading

Phong lighting model: **Diffuse Reflection**

Non-shiny surfaces

Diffuse reflection depends only on relative position of light source and surface normal.

![Diagram](image)
Shading

Phong lighting model: **Diffuse Reflection**

\[ I_d = I_i \cos \theta = I_i (\vec{l} \cdot \vec{n}) \]
Shading

Phong lighting model

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

- ambient light
- sum over all light sources
- diffuse
- specular
Outline

Graphics Primitives
Colors
Shading
Rasterization
Cameras
Fundamental Techniques
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*

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)
Back to Shading

Flat shading

(Color constant per polygon)
Gouraud shading (1971)

Shade vertices first, then interpolate* colors

(*) More on interpolation in the coming weeks...
Phong shading

Interpolate normals and shade every pixel separately

Note: Phong lighting model ≠ Phong shading
Outline

Graphics Primitives
Colors
Shading
Rasterization
Cameras
Fundamental Techniques
Projections

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

Camera position

Front clipping plane

Back clipping plane

Image plane

Only primitives located between front and back clipping planes will be rendered.
Camera

Degrees of freedom

- Elevation
- Roll
- Azimuth
Outline

Graphics Primitives
Colors
Shading
Rasterization
Cameras

Fundamental Techniques
Fundamental Techniques

Backface culling
Z-buffer
Alpha blending
Depth peeling
Fundamental Techniques

Back-face culling

Determine whether a polygon is visible (i.e., should be rendered)

Check polygon normal against camera viewing direction: remove polygons that are
Fundamental Techniques

Back-face culling

Definition: a polygon is visible if its normal is in the viewing direction of the camera, otherwise it is not visible (referred to as back-face culling).

Check: The normal of a polygon is compared against the camera viewing direction. Polygons with normals pointing away from the camera are not visible and can be removed from the rendering process.
Fundamental Techniques

Back-face culling

Definition:
Visible

Checking polygon normal against camera viewing direction: remove polygons that are

\[ \text{visible} \quad \text{not visible} \]
Fundamental Techniques

Back-face culling

Definition: When a polygon is visible (i.e., should be rendered), check its normal against the camera viewing direction.

Check to remove polygons that have a normal vector:

\[
\nabla \cdot \nabla > 0
\]

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

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
Fundamental Techniques

Z-buffer algorithm

gray value indicates distance to image plane
Fundamental Techniques

Alpha blending / compositing

Approximate visual appearance of semi-transparent object in front of another object

Implemented with OVER operator
Fundamental Techniques

Alpha blending / compositing

Approximate visual appearance of semi-transparent object in front of another object

Implemented with OVER operator

\[ c = af \cdot cf + (1 - af) \cdot ab \cdot cb \]

\[ a = af + (1 - af) \cdot ab \]

\[ c = (0.54, 0.4, 0) \]

\[ a = 0.94 \]

\[ cb = (1, 0, 0) \]

\[ ab = 0.9 \]

\[ cf = (0, 1, 0) \]

\[ af = 0.4 \]
Alpha blending / compositing

Approximate visual appearance of semi-transparent object in front of another object

Implemented with OVER operator

\[
\begin{align*}
\mathbf{c} &= \mathbf{a} + (1 - \alpha) \mathbf{b} \\
\mathbf{c} &= \mathbf{a} + (1 - \alpha) \mathbf{b} \\
\mathbf{c} &= \mathbf{a} + (1 - \alpha) \mathbf{b} \\
\mathbf{c} &= \mathbf{a} + (1 - \alpha) \mathbf{b}
\end{align*}
\]

Alpha blending is used in volume rendering and depth peeling. More on this later...
Fundamental Techniques

Depth Peeling

Order-independent transparency

Use 2 z-buffers and multiple rendering passes
Fundamental Techniques

Depth Peeling
Fundamental Techniques

Depth Peeling

![Depth Peeling Diagram]
Fundamental Techniques

Depth Peeling
Fundamental Techniques

Depth Peeling
Depth peeling is the only way to get correct transparency results in VTK. Make sure to use it!
Fundamental Algorithms

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

- Install VTK

https://www.cs.purdue.edu/homes/cs530/projects/project0.html