MAE 473-573: Lecture #13 - Hidden Lines / Pixel Tests

Lecture Overview:

• Motivation
• Identifying hidden lines/polygons
• Shaded Image concepts
• Convexity of geometry
• Pixel tests (Is a pixel inside a polygon?)
• Area-subdivision algorithms (Warnock)
• An intersection approach - Appel’s Method
• Motivation

“The Quest for Visual Realism”

We’re transitioning from 2D to 3D graphics. Once the 3D object is projected onto the 2D display plane, we must extract information to enhance the visual quality of the screen image.

Transition from wireframe to “filled” solid (hidden lines removed)

Eventually: Shaded (colored) images, texturing, etc.
• Identifying hidden lines/polygons

**Background**

• Ultimately, hidden line removal translates into hidden surface removal….

• For this reason, the concept of “shading” must be introduced

• Shading implementation concepts will be introduced later

**Concept (trivial)**

• Polygons are shaded proportional to “light” falling on them

• Polygons which can’t be seen (from viewpoint) **should not appear**
• Shaded Image concepts

Simplest test for hidden lines………………DOT PRODUCT

\[ \theta = \cos^{-1}\left(\frac{\mathbf{L} \cdot \mathbf{N}}{||\mathbf{L}|| \cdot ||\mathbf{N}||}\right) \]

If \( |\theta| \leq \frac{\pi}{2} \) - the polygon MAY be visible
If \( |\theta| > \frac{\pi}{2} \) - the polygon is not visible
• Convexity of shapes

• The “hidden line” test is always good for:
  a. Ruling out faces that are definitely out of view

• But: Problem with non-convex shapes……….. (MAE 550)
  Test can fail if shape has many geometrical intricacies

• Gross test for convexity - draw straight lines between interior points of a geometry - do they remain inside polygon?
Examples of Convex polygons

Examples of Concave polygons
• Painter’s Algorithm

• Sort object polygons by distance from camera, farthest to nearest (Based on z-coordinate data of polygon vertices)

• resolve any *ambiguities* that sorting may cause when the z-extents of numerous polygons overlap (can split polygons if need be)

• Shade (“paint”) the farthest polygons first, gradually working closer to the viewpoint

*Figure 13.21* Some cases in which z extents of polygons overlap.
Testing for depth ambiguities (5 checks):

• Compare x-extents - overlap?

• Compare y-extents - overlap?

• Is P entirely on the opposite side of Q’s plane from viewpoint?

• Is Q entirely on the same side of P’s plane as the viewpoint?

• Do the projections of the polygons onto the display plane NOT overlap?

TESTS NOT TRIVIAL……..
MAE 473-573: Lecture #13 - Hidden Lines / Pixel Tests

• Related concern (for shading):

Once you’ve determined which lines/surfaces are visible, how does one fill the appropriate pixels that comprise those entities?

i.e. How to determine if A Pixel is Inside a Polygon

• First, project vertices of 3D object to display plane
• Construct 2D polygon by connecting lines between vertices
• Then:
  a. Gross check (eliminates “clearly outside” pixels)
  b. Fine check
• Is a pixel inside a polygon? (cont.)
a. Gross check - extents
b. Fine check - “sum of angles” criteria (regard SIGN of angle...)
If sum = 0: exterior
If sum = 360: interior

Notes - Important!

For magnitude of each angle: use dot product test!

\[
\theta = \cos^{-1}\left( \frac{\mathbf{L} \cdot \mathbf{N}}{\|\mathbf{L}\| \|\mathbf{N}\|} \right)
\]

For sign of each angle: use cross product test!
Also: apply vertices in ascending order!

- Is a pixel in a polygon? (handout)
Area-subdivision Algorithms:

• Used for “visible surface determination”

• Uses a “divide and conquer” strategy to determine visibility

• Implements successive spatial partitioning in the projection plane

• As areas become smaller, fewer polygons overlap each area, rendering a visibility decision possible
Warnock’s algorithm (1969):

- Subdivides projection plane into 4 equal squares
- Projection of each polygon has one of four relationships to the area of interest:
  
  a. Surrounding polygons  
  b. Intersecting polygons  
  c. Contained polygons  
  d. Disjoint polygons

![Image of four relationships](image-url)
Warnock’s algorithm (cont.)

In these 4 situations, subdivision need not occur:

1. All polygons are disjoint from area.
   Result: Display background color.

2. There is only 1 intersecting OR contained polygon.
   Result: Background color is displayed, then polygon is filled.

3. There is a single surrounding polygon.
   Result: Area is filled with polygon color.
MAE 473-573: Lecture #13 - Hidden Lines / Pixel Tests

Warnock’s algorithm (cont.)

4. More than 1 polygon is intersecting, contained in, or surrounding the area, but one is a surrounding polygon afront all other polygons.

Result: Compute the z-coordinates of the planes of all surrounding, intersecting, and contained polygons. If there exists a surrounding polygon whose 4 corner z-coords are closest to the viewpoint than those of any other polygons, then fill that area with the color of the surrounding polygon.
Figure 13.24  Two examples of case 4 in recursive subdivision. (a) Surrounding polygon is closest at all corners of area of interest. (b) Intersecting polygon plane is closest at left side of area of interest. × marks the intersection of surrounding polygon plane; ○ marks the intersection of intersecting polygon plane; * marks the intersection of contained polygon plane.
Warnock’s algorithm (cont.)

- If no decision can be made - must subdivide square into 4 more squares
- Repeat process
Appel’s Method (1963):

A boolean intersection approach. Procedure: divide all the faces into back and front facing regions, and then propagate the visibility through the model.

Overall Method:

1. Eliminate all invisible edges.
2. Calculate the “Quantitative Invisibility” of all remaining segments of the remaining edges.
3. Project into 2D and draw all line-segments with a Q.I. of zero (i.e. all visible segments).
Appel’s Method (cont.)

Invisible edges
Contour edges
Material edges

Appel’s Method: types of edges

Appel’s implementation (handout)
MAE 473-573: Lectures #15 & 17

C - dynamic memory - examples
Review homogeneous coordinates
Project 1 - discuss
HW3 pseudo code
Go over transform examples in greater depth
Double buffering
Return HW1

Rotation transform actual code
Exam #1 Review