Lecture Notes #2 - Polygon scan-conversion

Reading: H&B Sec 3.11 (Foley 3.5, 3.6, 19.2.8 optional)

Overview:

- Triangles
- Scan-conversion
- Gouraud shading
- Texture mapping
- Simple polygons

Problem

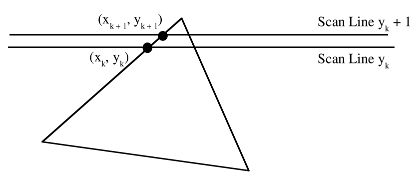
Given: A triangle *T* specified by vertices
$v_1 = (x_1, y_1)$, $v_2 = (x_2, y_2)$, $v_3 = (x_3, y_3)$ in pixel coordinates.

Find: All pixel centers inside *T*.

For now, we'll consider *T* to be a closed set of points -- that is, *T* contains its boundaries.

Sweep-line algorithm

Observation: If $p \in T$, then neighbors of p probably are too.



Successive scan lines of a triangle

This is another example of "coherence."

Leads to the "sweep-line algorithm":

1. Sweep from top to bottom
2. Maintain intersection of *T* and sweep line -- called a "span."
3. Paint pixels in the span.

Sweep-line algorithm, cont.

Here it is in pseudocode:

Initialize x_L and x_R

for each scan line y covered by T **do**

  Paint pixels (x_L, y), ... , (x_R, y) on current span

  Incrementally update x_L and x_R

end for

The edge pointers can be updated with Bresenham's algorithm.

Shading the triangle

The triangle can be "shaded" in a number of ways:

- "Flat shading": Use a uniform color.
- "Gouraud shading": Linearly interpolate between different vertex colors.
- "Texture mapping": Fill with colors from an arpitrary image.

Gouraud shading

Given: A triangle $T = (v_1, v_2, v_3)$ with colors c_1, c_2, c_3 specified at the vertices.

Problem: Scan-convert the triangle while linearly interpolating the colors.

Solution:

Polygons

Polygons can be classified as:

- "Simple"
- "Star-shaped"
- "Convex"

Polygon scan conversion

Given: A simple polygon P with vertices $(x_1, y_1), \ldots, (x_n, y_n)$.

Find: All pixel centers inside P.

Method 1

Simplest approach:

> *B* = *BoundingBox*(*P*)
> **for** each pixel $p \in B$ **do**
>   **if** $p \in P$ **then**
>   *Paint*(*p*)
>   **end if**
> **end for**

Testing point inclusion

The previous algorithm suggests an important subproblem:

Given: A polygon P and point $p = (p_x, p_y)$.

Determine: $p \in P$?

Ray firing

Here's a simple method to test $p \in P$:

1. Draw a ray from p outward in any direction.
2. Count number of intersections of ray with boundary of P.
3. If odd, $p \in P$; otherwise $p \notin P$.

Question: What happens if the ray crosses at a vertex?

Implicit line formulas

Here's a slightly easier method. Consider case of triangle:

1. Consider the edge of v_1v_2.
2. Write down the implicit form of the line:

$$l_{12}(x, y) = A_{12}x + B_{12}y + C_{12}$$

and pick sign of l_{12} so that $l_{12}(x_3, y_3) < 0$.

3. This defines a half-plane H_{12}:

$$H_{12} = \{(x, y) : l_{12}(x, y) <= 0\}$$

Implicit line formulas, cont.

4. Do the same for l_{13} and l_{23} to give half-planes H_{13} and H_{23}.

5. Notice:

$$T = H_{12} \cap H_{13} \cap H_{23}$$

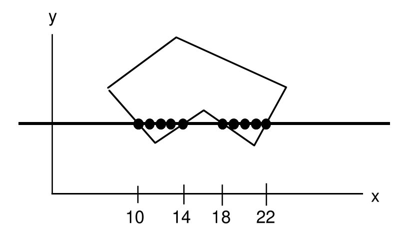
Therefore,

$$p \in T \Leftrightarrow (p \in H_{12}) \textit{ and } (p \in H_{13}) \textit{ and } (p \in H_{23})$$
$$\Leftrightarrow (l_{12}(p_x, p_y) \leq 0) \textit{ and } (l_{13}(p_x, p_y) \leq 0) \textit{ and } (l_{23}(p_x, p_y) \leq 0)$$

Question: Does this algorithm work for concave polygons?

Sweep-line algorithm

Method 1 doesn't take advantage of coherence.
A more efficient algorithm would identify where inclusion changes...



A scan line with two spans

All pixels in a "span" are inside and should be painted.

Sweep-line algorithm, cont.

Here is the algorithm in more detail:

for each scan line y covered by P **do**
  Identify all intersecions $x_0, \ldots, x_{k-1}$, sorted left-to-right
  for each pair of intersections x_{2i}, x_{2i+1} **do**
    Paint all pixels in $[(x_{2i}, y), (x_{2i+1}, y)]$
  end for
end for

Q: How should vertices be handled?

Active Edge List

For efficiency, the algorithm maintains a "sorted edge table" of "edge records":

The edge records intersecting the current scan line are sorted in the "active edge list."

Practical polygon scan conversion

Many implementations just <u>triangulate</u> the polygon, and then scan convert the triangles!

This is particularly easy to do for <u>convex polygons</u>

Gouraud-shaded polygons

<u>Method:</u>

- Use sweep-line algorithm to determine interior pixels.
- Linearly interpolate colors along edges.
- Linearly interpolate colors along spans.

Q: What is not so good about this algorithm?