Scan Conversion & Shading

Overview

- Scan conversion
  - Figure out which pixels to fill
- Shading
  - Determine a color for each filled pixel

Scan Conversion

- Render an image of a geometric primitive by setting pixel colors
  ```
  void SetPixel(int x, int y, Color rgba)
  ```
- Example: Filling the inside of a triangle

Triangle Scan Conversion

- Properties of a good algorithm
  - Symmetric
  - Straight edges
  - Antialiased edges
  - No cracks between adjacent primitives
  - MUST BE FAST!
**Triangle Scan Conversion**

- Properties of a good algorithm
  - Symmetric
  - Straight edges
  - Antialiased edges
  - No cracks between adjacent primitives
  - MUST BE FAST!

**Simple Algorithm**

- Color all pixels inside triangle

```c
void ScanTriangle(Triangle T, Color rgba)
{
    for each pixel P at (x, y)
    {
        if (Inside(T, P))
            SetPixel(x, y, rgba);
    }
}
```

**Inside Triangle Test**

- A point is inside a triangle if it is in the positive halfspace of all three boundary lines
  - Triangle vertices are ordered counter-clockwise
  - Point must be on the left side of every boundary line

**Inside Triangle Test**

```c
Boolean Inside(Triangle T, Point P)
{
    for each boundary line L of T {
        Scalar d = L.a*P.x + L.b*P.y + L.c;
        if (d < 0.0) return FALSE;
    }
    return TRUE;
}
```

**Simple Algorithm**

- What is bad about this algorithm?

```c
void ScanTriangle(Triangle T, Color rgba)
{
    for each pixel P at (x, y)
    {
        if (Inside(T, P))
            SetPixel(x, y, rgba);
    }
}
```

**Triangle Sweep-Line Algorithm**

- Take advantage of spatial coherence
  - Compute which pixels are inside using horizontal spans
  - Process horizontal spans in scan-line order
- Take advantage of edge linearity
  - Use edge slopes to update coordinates incrementally
**Triangle Sweep-Line Algorithm**

```c
void ScanTriangle(Triangle T, Color rgba)
{
    for each edge pair {
        initialize x_L, x_R;
        compute dx_L/dy_L and dx_R/dy_R;
        for each scanline at y {
            for (int x = x_L; x <= x_R; x++)
                SetPixel(x, y, rgba);
            x_L += dx_L/dy_L;
            x_R += dx_R/dy_R;
        }
    }
}
```

**Polygon Scan Conversion**

- Fill pixels inside a polygon
  - Triangle
  - Quadrilateral
  - Convex
  - Star-shaped
  - Concave
  - Self-intersecting
  - Holes

What problems do we encounter with arbitrary polygons?

**Inside Polygon Rule**

- Odd-parity rule
  - Any ray from P to infinity crosses odd number of edges

**Polygon Sweep-Line Algorithm**

- Incremental algorithm to find spans, and determine insideness with odd parity rule
  - Takes advantage of scanline coherence
**Polygon Sweep-Line Algorithm**

```c
void ScanPolygon(Triangle T, Color rgba) {
    sort edges by maxy
    make empty "active edge list"
    for each scanline (top-to-bottom) {
        insert/remove edges from "active edge list"
        update x coordinate of every active edge
        sort active edges by x coordinate
        for each pair of active edges (left-to-right)
            SetPixels(x_i, x_{i+1}, y, rgba);
    }
}
```

**Hardware Scan Conversion**

- Convert everything into triangles
  - Scan convert the triangles

**Hardware Antialiasing**

- Supersample pixels
  - Multiple samples per pixel
  - Average subpixel intensities (box filter)
  - Trades intensity resolution for spatial resolution