Projective Rendering Pipeline

Lines and Curves

Explicit
- Line: \( y = mx + b \)
- Circle: \( y = \pm \sqrt{r^2 - x^2} \)

Parametric
- Line: \( x(t) = x_0 + t(x_1 - x_0), y(t) = y_0 + t(y_1 - y_0) \)
  \( t \in [0,1] \)
  \( P(t) = P_0 + t(P_1 - P_0) \)
  \( P(t) = (1-t)P_0 + tP_1 \)
- Circle: \( x(\theta) = r \cos(\theta), y(\theta) = r \sin(\theta), \) \( \theta \in [0,2\pi] \)

Implicit
- Line: \( F(x,y) = (x-x_0)dy - (y-y_0)dx \)
  \( F(x,y) = 0 \) (on line)
  \( F(x,y) > 0 \) (on line)\( \) (on line)\( \) line
  \( F(x,y) < 0 \) (on line)\( \) (on line)\( \) line
- Circle: \( F(x,y) = x^2 + y^2 - r^2 \)
  \( F(x,y) = 0 \) (on circle)
  \( F(x,y) > 0 \) (on circle)
  \( F(x,y) < 0 \) (on circle)

Polygons

Basic Types
- Simple convex
- Simple concave
- Non-simple (self-intersection)

Polygons: Vertex List

- Ordered list of references to a vertex list
**Triangle Meshes: Winged Edge Data Structure**

**Mesh Operations**
- Find boundary edges
- Edge collapse

---

**Winged Edge Data Structure**

**Features**
- Data structure for polygonal meshes
- Holds required topological information
- Relatively storage efficient
- Works for arbitrary polygons without holes
  - not just triangles
- Represents only manifold surfaces

---

**Triangle Meshes: Winged Edge Data Structure**

**Edge:**
- Vertex *start; (x)
- Vertex *end; (y)
- Face *left; (F_l)
- Face *right; (F_r)
- Edge *leftPred; (P_l)
- Edge *leftSucc; (S_l)
- Edge *rightPred; (P_r)
- Edge *right Succ; (S_r)

---

**Scan Conversion**

**Basic Algorithm**
- Intersect each scanline with all edges
- Sort intersections in x
- Calculate parity to determine in/out
- Fill the 'in' pixels

---

**Scan Conversion**

- Exclude horizontal edges
- Vertices lying on scanlines
  - Local extrema: leave as is (count twice)
  - Otherwise: shorten edge (count once)
**Scan Conversion**

**Exploit Spatial Coherence**
- neighboring scanlines likely to cross same edges

**Building the Edge Table**
- for each edge
  - skip if horizontal
  - if not local extrema, shorten upper vertex
    - add edge to linked list for scanline corresponding to lower vertex, storing
      - end y: last scanline to consider
      - start x: starting x coord for edge
      - 1/m: use for incrementing x

**Using the Edge Table**
- maintain an Active Edge List (AEL)
  
  for each scanline {
    add new edges to AEL from edge table
    if (AEL != NIL) {
      sort AEL by x
      fill pixels between edge pairs
      delete finished edges
      update edge x values
    }
  }

**Scan Conversion of Triangles**
*(to be completed)*