EECS 487: Interactive Computer Graphics

Lecture 5:
- Finish up line rasterization
- Line clipping

Clipping

Clip against viewing window/viewport

Why clip?
- avoids rasterizing outside window
- speeds up rasterization
- prevents memory errors
- avoids divide by 0 and overflows
- in 3D, clip against view volume
- polygons that are too close can obscure view
- those too far shouldn’t be visible and could mess up depth buffer

Clipping Line Segments

How to clip?
- preprocessing to exclude/include trivial cases
  - accept/reject bitmask test: Cohen-Sutherland
- clip the intersecting cases:
  - parametric line trimming: Cyrus-Beck

Clipping Cohen-Sutherland

Trivial Accept/Reject test: compute a 4-bit “outcode” ($L_i$) for each end point ($p_i$):

Before clipping

after clipping

Cohen-O'Sullivan

Accept line if $L_0 \& L_1 = 0$ (bitwise or) both points are inside window
Reject line if $L_0 \& L_1 \neq 0$ (bitwise and) line is outside window
Else may need clipping
### Line Clipping Examples

- **A**
  - \( A_0 = 1001 \)
  - \( A_1 = 1000 \)
  - \( A_0 \land A_1 = 1000 \) - A is Rejected

- **F**
  - \( F_0 = 0000 \)
  - \( F_1 = 0100 \)
  - \( F_0 \lor F_1 = 0100 \) - F needs clipping

- **E**
  - \( E_0 = 0000 \)
  - \( E_1 = 0000 \)
  - \( E_0 \lor E_1 = 0000 \) - E is Accepted

- **H**
  - \( H_0 = 0100 \)
  - \( H_1 = 0010 \)
  - \( H_0 \land H_1 = 0100 \) - H needs clipping?

### Line Clipping Cohen-Sutherland

- Each ‘1’ in the outcode indicates the line intersecting an edge, e.g., \( 0100 \) means intersection with y-min.
- Starting from the **left most outside point** (A in example), going left to right (e.g.,) on the outcode, compute the intersection with the window edge that cuts the line into two segments.
- Test the outcodes of each segment, clip the segment(s) **outside** the window.
- **Recurse** until all segments are checked.

### Normal Vectors

- **What is the normal vector** of a line? A vector perpendicular to the line.

- **What is a unit normal?**
  - Normal vector of magnitude one: \( \mathbf{n}/||\mathbf{n}|| \)

- Normal vectors are important to many graphics calculations.

### Implicit Line Eqn. Using Vectors

- Let \( \mathbf{n} \) be a normal vector of the line and \( \mathbf{q} \) a point on the line.
  - the point \( \mathbf{p} \) is on the line iff \( f(\mathbf{p}) = \mathbf{n} \cdot (\mathbf{p} - \mathbf{q}) = 0 \)
  - the point \( \mathbf{p}' \) is above the line iff \( f(\mathbf{p}') > 0 \) \((\theta' < 90')\)
  - the point \( \mathbf{p}'' \) is below the line iff \( f(\mathbf{p}'') < 0 \) \((\theta'' > 90')\)
  - if \( f(\mathbf{p}) \neq 0 \), \( \mathbf{p}' \)’s projection onto \( \mathbf{n} \) has a non-zero length.

\[ \theta \text{ measured in the direction of travel} \]
Cyrus-Beck Line Clipping

Compute the intersection between line $u$ and edge $i$

Let:
• $u$ the vector from $p_0$ to $p_1$: $u = (p_1 - p_0)$
• $n_i$ be the normal of edge $i$, pointing away from the clipping window
• $p_e$ an arbitrary point on edge $i$

then:
• if $n_i \cdot u = 0$ the line is parallel to the edge $i$
• otherwise, let $p_e(t)$ be the intersection of $u$ and edge $i$
• solve for $t_i$ (repeat for each of the four edges):

$$n_i \cdot (p_e(t_i) - p_e) = 0$$
$$n_i \cdot [p_0 + t_i(p_1 - p_0) - p_e] = 0$$
$$n_i \cdot [p_0 - p_e] + n_i \cdot t_i u = 0$$

$$t_i = \frac{n_i \cdot [p_0 - p_e]}{-n_i \cdot u} = \frac{n_i \cdot [p_e - p_0]}{n_i \cdot u}$$

$u$ $p_0$ $p_1$ $n_i$ $p_e$ $t_i$

Foley et al. ’94

Cyrus-Beck Line Clipping

Now classify each intersection point as Potentially Entering (PE) or Potentially Leaving (PL) at edge $i$:  
• if $n_i \cdot u < 0$, intersection is PE (why?)
• if $n_i \cdot u > 0$, intersection is PL (why?)

Let $t_L$ be the MIN of the $t_i$’s that are PL and $t_E$ the MAX of the $t_i$’s that are PE
• if $t_L < t_E$, the line is outside the clipping window (Line 2)
• otherwise ($t_L$, $t_E$) are the clipped line’s bounding points
• in case actual line segment starts or ends inside window, $t_L < 0$ or $t_E > 1$ respectively, we let $\max(0, t_L)$ and $\min(1, t_E)$ be the clipped line’s bounding points

Foley et al. ’94

Cyrus-Beck Line Clipping

Let two sides of the clipping window define a region $E(nter)$ that the line enters and never leaves

Let the other two sides define a region $L(ave)$ that the line starts in and eventually leaves

(Algorithm determines $E$ and $L$ automatically!)

The dot products of the line and the normal ($n$) of the boundary edges determine the parameters (the $t$’s) at the intersection points

Whereas Cohen-Sutherland is limited to upright rectangle, Cyrus-Beck works well with arbitrary convex polygon as clipping area