EECS 487: Interactive Computer Graphics

Lecture 5:

- Finish up line rasterization
- Line clipping

Clipping

Clip against viewing window/viewport

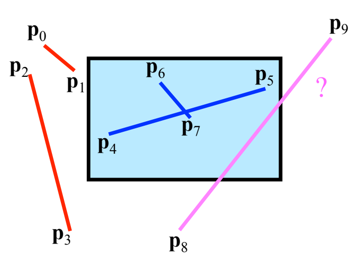
Why clip?

- 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

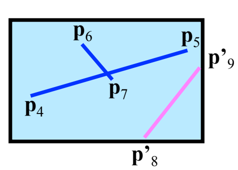
Clipping Line Segments

How to clip?

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



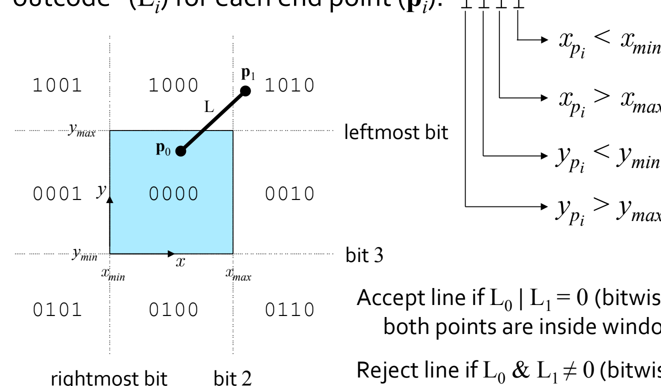
before clipping



after clipping

Cohen-Sutherland

Trivial Accept/Reject test: compute a 4-bit "outcode" (L_i) for each end point ($\mathbf{p}_i$): 1111



Accept line if $L_0 \mid L_1 = 0$ (bitwise or)
both points are inside window

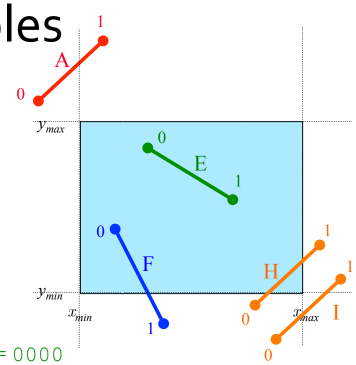
Reject line if $L_0 \,\&\, L_1 \neq 0$ (bitwise and)
line is outside window

Else may need clipping

Line Clipping Examples

1111
- $x_{p_i} < x_{min}$
- $x_{p_i} > x_{max}$
- $y_{p_i} < y_{min}$
- $y_{p_i} > y_{max}$

Accept if $L_0 \mid L_1 = 0$
Reject if $L_0 \,\&\, L_1 \neq 0$



A_0 = 1001
A_1 = 1000
$A_0 \mid A_1$ = 1001
$A_0 \,\&\, A_1$ = 1000 A is Rejected

F_0 = 0000
F_1 = 0100
$F_0 \mid F_1$ = 0100
$F_0 \,\&\, F_1$ = 0000 F needs clipping

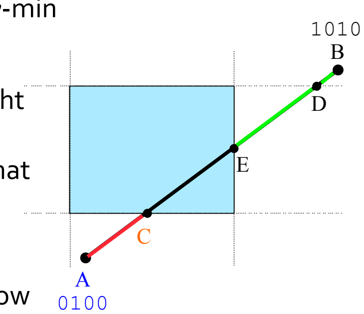
E_0 = 0000
E_1 = 0000
$E_0 \mid E_1$ = 0000 E is Accepted

H_0 = 0100
H_1 = 0010
$H_0 \mid H_1$ = 0110
$H_0 \,\&\, H_1$ = 0000 H needs clipping?
I 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

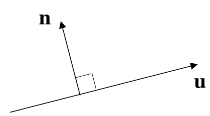


Test the outcodes of each segment, clip the segment(s) outside the window

Recurse until all segments are checked

Normal Vectors

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



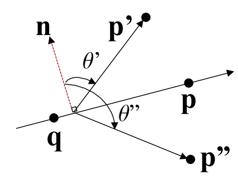
What is a unit normal?
Normal vector of magnitude one: $\mathbf{n}/||\mathbf{n}||$

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 $f(\mathbf{p}') > 0$ ($\theta' < 90^\circ$)
- the point $\mathbf{p}''$ is below the line iff $f(\mathbf{p}'') < 0$ ($\theta'' > 90^\circ$)
- if $f(\mathbf{p}) \neq 0$, $\mathbf{p}$'s projection onto $\mathbf{n}$ has a non-zero length



θ measured in the direction of travel

Cyrus-Beck Line Clipping

Compute the intersection between line $\mathbf{u}$ and edge i

Let:

- $\mathbf{u}$ the vector from $\mathbf{p}_0$ to $\mathbf{p}_1$: $\mathbf{u} = (\mathbf{p}_1 - \mathbf{p}_0)$
- $\mathbf{n}_i$ be the normal of edge i, pointing away from the clipping window
- $\mathbf{p}_{e_i}$ an arbitrary point on edge i

then:

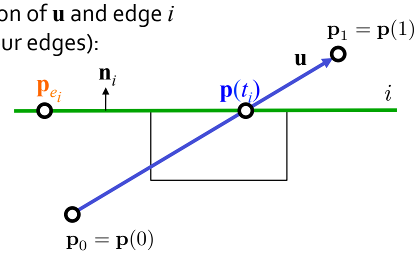
- if $\mathbf{n}_i \cdot \mathbf{u} = 0$ the line is parallel to the edge i
- otherwise, let $\mathbf{p}(t_i)$ be the intersection of $\mathbf{u}$ and edge i
- solve for t_i (repeat for each of the four edges):

$$\mathbf{n}_i \cdot [\mathbf{p}(t_i) - \mathbf{p}_{e_i}] = 0$$

$$\mathbf{n}_i \cdot [\mathbf{p}_0 + t_i(\mathbf{p}_1 - \mathbf{p}_0) - \mathbf{p}_{e_i}] = 0$$

$$\mathbf{n}_i \cdot \left[\mathbf{p}_0 - \mathbf{p}_{e_i}\right] + \mathbf{n}_i \cdot t_i \mathbf{u} = 0$$

$$t_i = \frac{\mathbf{n}_i \cdot [\mathbf{p}_0 - \mathbf{p}_{e_i}]}{-\mathbf{n}_i \cdot \mathbf{u}} = \frac{\mathbf{n}_i \cdot [\mathbf{p}_{e_i} - \mathbf{p}_0]}{\mathbf{n}_i \cdot \mathbf{u}}$$



Foley et al.94

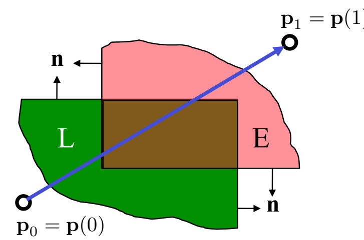
Cyrus-Beck Line Clipping

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(eave) that the line starts in and eventually leaves

(Algorithm determines E and L automatically!)

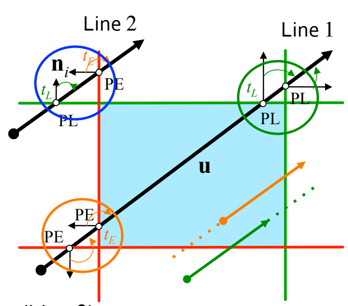
The dot products of the line and the normal ($\mathbf{n}$) of the boundary edges determine the parameters (the t's) at the intersection points



Cyrus-Beck Line Clipping

Now classify each intersection point as Potentially Entering (PE) or Potentially Leaving (PL) at edge i:

- if $\mathbf{n}_i \cdot \mathbf{u} < 0$, intersection is PE (why?)
- if $\mathbf{n}_i \cdot \mathbf{u} > 0$, 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_E, t_L) are the clipped line's bounding points
- in case actual line segment starts or ends inside window, $t_E < 0$ or $t_L > 1$ respectively, we let $\max(0, t_E)$ and $\min(1, t_L)$ be the clipped line's bounding points

Foley et al.94

Cyrus-Beck Line Clipping

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