2D Graphics Primitives III

Clipping

Example: user change in window size and Expose event

Redraw entire image?

Redraw clipped image

Specifications of entire scene as lines, circles etc (OpenGL calls)
Scan convert entire scene into a pixmap
Copy portion of pixmap as specified by clipping rectangle

Versus

Specification of entire scene as above
Scan convert entire scene, but write only visible pixels
(Scissoring)

Versus

Specifications of entire scene as above
Specifications of clipped scene
Scan convert from clipped specifications

First is easy, but wastes time and space

Second may be quite efficient if done in microcode or hardware
Generalizes to arbitrary shape clip regions

Third is often best for points, lines, polygons (simple algorithms)
Define Clip Rectangle (Region)

Start with rectangular clip region (clip-box)

defined by \( x_l, x_r, y_t, y_b \)

Clipping Points

How determine if point should be displayed?

Clipping Lines

More difficult

Which lines easy to clip?
Cohen-Sutherland Line Clipping Algorithm

Based on idea that some lines are trivially accepted (entire line drawn)
others trivially rejected (none of line drawn)
others more difficult (maybe clip some, maybe draw nothing)

Look at nine regions of space as divided by the clip-box

Assign 4 bit region code to each region:

\[ b_4 \ b_3 \ b_2 \ b_1 \]

\[ b_1 = 1 \text{ if point is to the left of the left boundary} \]
\[ b_2 = 1 \text{ if point is to the right of the right boundary} \]
\[ b_3 = 1 \text{ if point is below bottom boundary} \]
\[ b_4 = 1 \text{ if point is above top boundary} \]

Find region code of each end of line \((C_1, C_2)\)
Use to accept or reject line

eg if both ends are 0000?

what else is easy case?
Look at logical operations on the region codes (AND, OR)

A) If C1 OR C2 = 0000, then trivially accept line

B) How trivially reject a line that has both points above top?
   C1 AND C2 = 1xxx

How trivially reject a line below, to right and to left?
   C1 AND C2 = ?
   C1 AND C2 = ?
   C1 AND C2 = ?

How generalize these four cases?
   C1 AND C2 not equal 0000

C) Rest are difficult
   C1 AND C2 = 0000

Cohen-Sutherland Algorithm
   Start with input list of lines (endpoints)
   M: While input list is not empty
      Find Region codes (C1 and C2) for line
      Remove line from input list
      If C1 OR C2 = 0000, then add line to output list
      Else if C1 AND C2 = 0000, find intersection of line
         with an edge (top, bottom, left, right order)
         Add intersection point and interior point to input list
   End
Given the order of testing for intersections, what is a worst case input line for Cohen-Sutherland algorithm?

This required two clips
First required two tests
Second required four tests

Can a line require 4 clips? (How many regions can a line pass through?)

If four clips required, then how many tests on first clip?
on second?
on third?
on forth?

Cohen-Sutherland not the most efficient algorithm as it can end up doing needless clipping.
Still used widely, since widely known
Cyrus-Beck Parametric Line Clipping Algorithm
more efficient
can clip against convex polygon clip region
can clip in 3D as well as 2D

Liang-Barsky
like above, but faster for upright rectangular 2D and 3D regions

Derivatons of Liang-Barsky

Based on parametric representation of line 
\((x_1, y_1)\) 
\[ \Delta x = x_2 - x_1 \]
\[ \Delta y = y_2 - y_1 \]
\((x_2, y_2)\)

\[ x = x_1 + \Delta x \cdot u \]
\[ y = y_1 + \Delta y \cdot u \]
\[ 0 \leq u \leq 1 \]

Write clipping equations in parametric form 

\[ x_L \leq x_1 + \Delta x u \leq x_r \]
\[ y_b \leq y_1 + \Delta y u \leq y_t \]

Rewrite as four inequalities 

\[ u \cdot p_k \leq q_k , \text{ where } k = 1, 2, 3, 4 \]

\[ p_1 = -\Delta x \quad q_1 = x_1 - x_L \]
\[ p_2 = \Delta x \quad q_2 = x_r - x_1 \]
\[ p_3 = -\Delta y \quad q_3 = y_1 - y_b \]
\[ p_4 = \Delta y \quad q_4 = y_t - y_1 \]
Each value of $k$ corresponds to one boundary:
- $k = 1$ corresponds to left boundary
- $k = 2$ corresponds to the right boundary
- $k = 3$ ?
- $k = 4$ ?

If line is parallel to the $k$th boundary, then
$p_k = ?$

The values of $q_k$ indicate which side of the $k$th boundary the start point is on:
- if $q_k < 0$, then $p_1$ is outside $k$th boundary
- if $q_k >= 0$, the $p_1$ is inside or on the $k$th boundary

if $p_k < 0$, then line goes from outside to inside the $k$th boundary
if $p_k > 0$, then line goes from inside to outside the $k$th boundary

if $p_k - 0$, then the intersection of the line with the $k$th boundary is at

$$r_k = q_k / p_k$$

For each line we want to find $u_1$ and $u_2$ that lie in clip region
Liang-Barsky Algorithm
For each line segment
\( u_1 = 0; \)
\( u_2 = 1; \)  (We are starting with the original endpoints)
\( k =1; \)
while still need to clip and \( k \leq 4 \)
compute \( p_k \) and \( q_k \)
if \( p_k = 0 \) and \( q_k < 0 \), then reject line and stop clipping
else
  if \( p_k < 0 \),
    \( u_1 = \) maximum of \( u_1 \) and \( r_k \)
  else
    \( u_2 = \) minimum of \( u_2 \) and \( r_k \)
if \( u_1 > u_2 \)
  reject line and stop clipping
  \( k = k + 1; \)
end
if line not rejected, \( u_1 \) and \( u_2 \) are end points of clipped line
end
Example:

\[ \begin{align*}
\text{Line AB} & \\
p_1 &= \Delta x & q_1 &= x_1 - x_L \\
p_2 &= \Delta x & q_2 &= x_R - x_1 \\
p_3 &= -\Delta y & q_3 &= y_1 - y_B \\
p_4 &= \Delta y & q_4 &= y_T - y_1 \\
\end{align*} \]

\[ \begin{align*}
u_1 &= 0 & u_2 &= 1 & p_1 &< 0 & q_1 &< 0 \\
u_1 &= r_1 & u_2 &= 1 & p_2 &> 0 & q_2 &> 0 \\
u_1 &= r_1 & u_2 &= 1 & p_3 &> 0 & q_3 &> 0 \\
u_1 &= r_1 & u_2 &= r_3 & p_4 &< 0 & q_4 &> 0 \\
\end{align*} \]
Liang-Barsky versus Cohen-Sutherland
Liang-Barsky computes fewer intersections for a line needing clipping
But doesn't have a trivial accept

If most lines can be trivially accepted or rejected,
  Use Cohen-Sutherland
else
  Use Liang-Barsky

Clipping Circles
Can approximate with 2 rectangles for trivial accept and reject

Outer used for?
Inner used for?

Can make better approximations using polygons