Binghamton University

EngiNet™

State University of New York

Thomas J. Watson

School of Engineering
and Applied Science

EngiNet™

WARNING

CS 460/560

Computer Graphics

Professor Richard Eckert

Lecture # 15

March 26, 2001

Lecture 14 -- Clipping

Clipping

- Elimination of parts of scene outside a window or viewport
- Clipping with respect to a window (xwmin, ywmin, xwmax, ywmax given)
  - Clip at this level ==> fewer points go through viewing transformation
- Clipping with respect to a viewport (xvmin, yvmin, xvmax, yvmax given)

Clipping

- Points
- Lines
  - Cohen-Sutherland Line Clipper
- Polygons
  - Sutherland-Hodgeman Polygon Clipper
  - Weiler-Atherton Polygon Clipper
- Other Curves
- Text

Point Clipping

- Given:
  - point (x,y)
  - clipping rectangle (window or viewport) (xmin,ymin,xmax,ymax)
- Point test:

```
if ((x<=xmax) && (x>=xmin)
    && (y<=ymax) && (y>=ymin)
        the point x,y lies inside the clip area
        so keep it!
```

Line Clipping

- Could apply point test to all points on line
  - Too much work
- Need a simple test involving line's endpoint coordinates

Cohen-Sutherland Line Clipper

- Observation-- All lines fall into one of three categories
  1. Both endpoints inside clip rectangle
     - (Trivially accept entire line)
  2. Both endpoints outside clip rectangle on the same side of one of its borders
     - (Trivially reject entire line)
  3. Neither 1 or 2
     - (Chop off part of line outside one of borders and repeat)

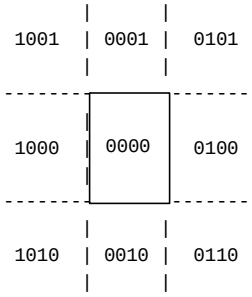
Region Code

- A tool in assigning lines to Category 1 or 2
- 4-bit region code number assigned to an endpoint (x,y)
- Any set bit means endpoint is outside of one of the 4 borders of the clip rectangle
- Each bit position corresponds to a different border

Region Code RC = LRBT

- L=left (if x<xmin, L=1, else L=0)
- R=Right (if x>xmax, R=1, else R=0)
- B=Bottom (if y<ymin, B=1, else B=0)
- T=Top (if y>ymax, T=1, else B=0)
- Divides entire x-y plane 9 regions

Region Codes (LRBT)



Category 1 Lines

- Assume region codes for the line's endpoints are RC1 and RC2

if (RC1 | RC2 == 0)
both RCs are 0000
both endpoints are inside
so Category 1 (trivial accept)

Category 2 Lines

- If both endpoints are outside same border
  - (Category 2 line)
- then both region codes will have same bit set in one of the four bit positions--i.e.:

if (RC1 & RC2 != 0)
both endpoints outside same border
so Category 2 (trivial reject)

Category 3 Lines

- Want to chop off outside part of line
- May have both endpoints (P1 & P2) outside different borders of clip region
  - So it's not important which end is chopped off first
- But if one endpoint's in and other's out:
  - Want to chop off outside end
  - So Arrange things so P1 is the outside point
    - (swap P1 & P2 if necessary)

How to do the Chopping

- Want to determine the new endpoint
- P1's endpoint coordinates (x1,y1), (x2,y2) are known
- Slope m can be computed from them
- So y = m*(x-x2) + y2 (point slope form)
- Or x = (y-y2)/m + x2
- Look at P1's region code (RC1)
- Four possible cases:

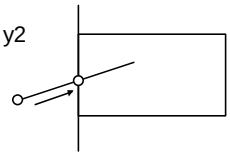
If RC1 == 1xxx (P1 to left of xmin)

- New endpoint should be on the left boundary:

  x1 <-- xmin

  y1 <-- m*(xmin-x2) + y2

  Reset RC's L bit



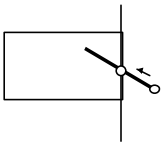
If RC1 == x1xx (P1 right of xmax)

- New endpoint should be on the right boundary:

  x1 <---xmax

  y1 <---m*(xmax-x2) +y2

  Reset RC's R bit



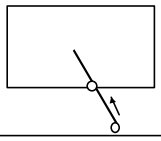
If RC1 == xx1x (P1 below ymin)

- New endpoint should be on the bottom boundary:

  y1 <---ymin

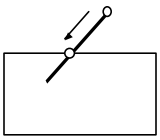
  x1 <---(ymin-y2)/m + x2

  Reset RC's B bit



If RC == xxx1 (P1 above ymax)

- New endpoint should be on the top boundary:
  - y1 <---ymax
  - x1 <---(ymax-y2)/m + y2
  - Reset RC's T bit



The C-S Line Clipping Algorithm

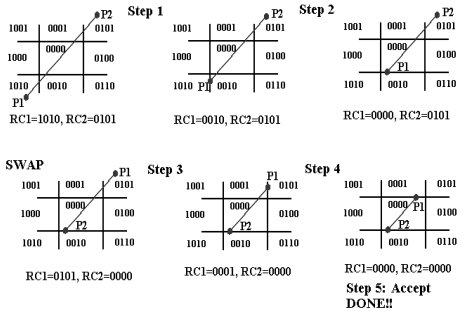
- Input:
  - Original endpoints (x1,y1,x2,y2)
  - Clip region boundaries (xmin,ymin,xmax,ymax)
- Output:
  - Accept Code (AC)
    - AC==TRUE ==> part of line was inside
    - AC==FALSE ==> no part of line was inside
  - Clipped Line endpoints (x1,y1,x2,y2)
    - (if AC==TRUE)

C-S Algorithm Pseudo-code:

```
CS_LineClip(xmin,ymin,xmax,ymax,x1,y1,x2,y2,AC)
   done = FALSE
   While (!done)
      Calculate endpoint codes rc1, rc2
      If ((rc1 | rc2) == 0 )    // Category 1
         done = TRUE
         AC = TRUE
      Else
         If ((rc1 & rc2) != 0)  // Category 2
            done = TRUE
            AC = FALSE
         Else
            If (P1 is inside)
              Swap (x1,x2), (y1,y2); and rc1,rc2
```

```
If (L-bit of rc1 is set)     // 1xxx
  x1 = xmin
  y1 = m*(xmin-x2) + y2
Else
  If (R-bit of rc1 is set)   // x1xx
    x1 = xmax
    y1 = m*(xmax-x2) + y2
  Else
    If (B-bit of rc1 is set) // xx1x
      y1 = ymin
      x1 = (ymin-y2)/m + x2
    Else                     // xxx1
        y1 = ymax
        x1 = (ymax-y2)/m + x2
```

Cohen-Sutherland Clipping Example



Polygon Clipping

- Clip a polygon to a rectangular clip area
- Input
  - Ordered list of polygon vertices (nin, vin[])
  - Clip rectangle boundary coordinates (xmin, ymin, xmax, ymax).
- Output:
  - An ordered list of clipped polygon vertices (nout, vout[]).
  - vin[] and vout[] could be arrays of POINTs

Approaches to Polygon Clipping

- Use a line clipper on each polygon edge???
- But we usually won't get back a polygon
  - Parts of the clip rectangle will be edges of the clipped polygon
- Really need new list of edges (or vertices)

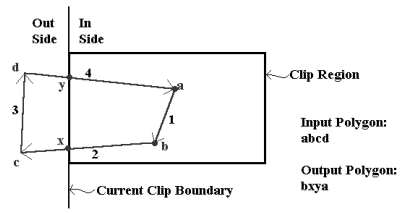
Sutherland-Hodgeman Polygon Clipper

- Approach:
  - Clip all polygon edges with respect to each clipping boundary
  - Do four passes; on each pass:
    - Traverse current polygon and clip with respect to one of the four boundaries
    - Assemble output polygon edges as you go
    - vin[] --> Clip Left --> vtemp1[] --> Clip Right --> vtemp2[] --> Clip Bottom --> vtemp3[] --> Clip Top --> vout[]

- On any polygon traversal the clip boundary divides plane into "in" side and "out" side
- For any given edge (vertices i and i+1),
  - during traversal, there are four possibilities:
  - (Assume vertex i has already been processed)

```
VERTEX i VERTEX i+1  ACTION
 in       in       Add Vertex i+1 to output list
 out      out      Add no vertex to output list
 in       out      Add intersection point with
                    edge to output list
 out      in       Add intersection point with edge
                    and vertex i+1 to output list
```

Sample Traversal



Traversal	Type	Action
1 a → b	in-in	Add point b
2 b → c	in-out	Add intersection point x
3 c → d	out-out	Add nothing
4 d → a	out-in	Add intersection point y and point a

Implementation

- Function sh_clip()
  - Will clip an input polygon (ni, vi[])
  - With respect to a given boundary (bndry)
  - Generating an output polygon (no, vo[])
- Enumerate the boundaries as:
  - LEFT, RIGHT, BOTTOM, and TOP

sh_clip(ni, vi[], no, vo[], xmin, ymin, xmax, ymax, bndry);

vi[] and vo[]: could be arrays of POINTs

ni, no: number of points in each array

xmin, ymin, xmax, ymax: clip region boundaries

Using sh_clip() to clip a polygon

- Make four calls to sh_clip():

sh_clip(nin, vin[], ntemp1, vtemp1[], xmin, ymin, xmax, ymax, LEFT);

sh_clip(ntemp1, vtemp1[], ntemp2, vtemp2[], xmin, ymin, xmax, ymax, RIGHT);

sh_clip(ntemp2, vtemp2[], ntemp3, vtemp3[], xmin, ymin, xmax, ymax, BOTTOM);

sh_clip(ntemp3, vtemp3[], nout, vout[], xmin, ymin, xmax, ymax, TOP);

Three Helper Functions

BOOL inside(V, xmin, ymin, xmax, ymax, Bndry)

- Returns TRUE if vertex point V is on the "in" side of boundary Bndry

intersect(V1, V2, xmin, ymin, xmax, ymax, Bndry, Vnew)

- Computes intersection point of edge whose endpoints are V1 and V2 with boundary Bndry
- Returns the resulting point in Vnew

output(V, n, vout[])

- Adds vertex point V to the polygon (n, v[])
  - n will be incremented by 1
  - vertex V added to end of polygon's vertex list v[]

```
sh_clip (ni, vi[], no, vo[], bndry)
no = 0                          // output list begins empty
First_V = vi[0]                 // first vertex (i)
For (j=0 to ni-1)               // traverse polygon
  Second_V = v[(j+1) % ni]      // second vertex (i+1)
  If (inside(first_V, bndry)
    If (inside(Second_V, bndry)  // "in-in" case
       output(Second_V, no, vo)
    Else                         // "in-out" case
       intersect(First_V, Second_V, bndry, Vtemp)
       output (Vtemp, no, vo)
   Else
     If (inside(Second_V, bndry)  // "out-in" case
       intersect(First_V, Second_V, bndry, Vtemp)
       output(Vtemp, no, vo)
       output(Second_V, no, vo)   // no "out-out" case
  First_V = Second_V             // prepare for next edge
```

Example of S-H Clipping