
CS770/870 Fall 2008

Clipping

Related material in Hill and Kelley: Ch. 3.3

10:35

CS770/870 Fall 2008 Bergeron/Plumlee

1

Preview

- 2D clipping
 - line clipping
 - polygon clipping
 - Canonical 2D clipping coordinates
- 3D clipping
 - to a parallel projection clipping volume
 - to a perspective projection clipping volume
 - canonical 3D clipping coordinates

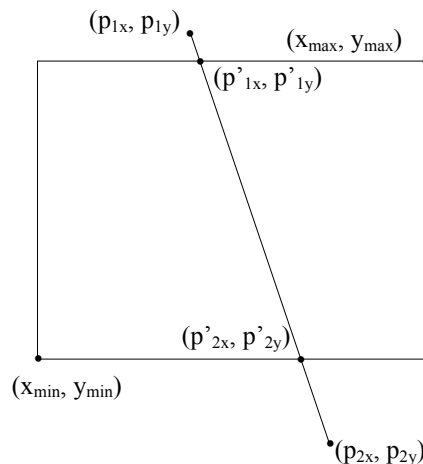
10:35

CS770/870 Fall 2008 Bergeron/Plumlee

2

2D Line Clipping

- Given a *window* in world coordinates
 - (x_{\min}, y_{\min}) to (x_{\max}, y_{\max})
- and a line in world coordinates:
 - (p_{1x}, p_{1y}) to (p_{2x}, p_{2y})
- identify what portion of the line (if any) lies within the window



10:35

CS770/870 Fall 2008 Bergeron/Plumlee

3

Cohen-Sutherland Line Clipping

- Classic algorithm
 - Fast rejection of lines entirely outside window
 - Efficient intersection of lines with window edges
- Steps
 1. Classify each line endpoint by an “out” code that identifies the point’s location wrt all 4 window edges
 2. Reject any lines, both of whose endpoints are “outside” at least 1 window edge
 3. Intersect line with each window edge

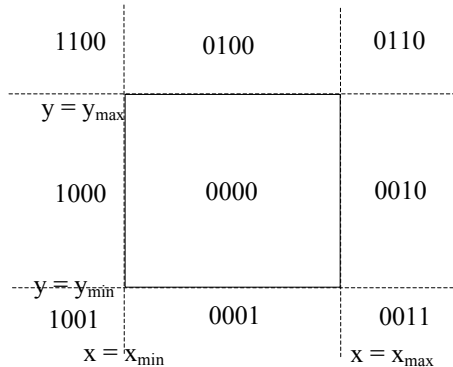
10:35

CS770/870 Fall 2008 Bergeron/Plumlee

4

Out codes

- “out” code is a 4-bit value: LTRB
 - L: $x < x_{\min}$ (x,y) to left of window
 - T: $y > y_{\max}$ (x,y) above window
 - R: $x > x_{\max}$ (x,y) to right of window
 - B: $y < y_{\min}$ (x,y) below window
- Each region in the world has a unique out code
- For a line with end points P_1 and P_2 , compute the corresponding out codes, out_1 and out_2
- if $((out_1 \& out_2) \neq 0000)$
 - trivial reject
- else if $(out_1 == out_2 == 0000)$
 - trivial accept
- else
 - intersect window edges with line



10:35

CS770/870 Fall 2008 Bergeron/Plumlee

5

Line Equation in slope/intercept form

- Equation of a line:

$$y = mx + b$$
 where m is slope of the line and b is 0-intercept
- Given a line defined by points (x_1, y_1) and (x_2, y_2)

$$m = (y_2 - y_1) / (x_2 - x_1)$$
- Can find b by plugging either end point into $y=mx+b$

$$b = y_1 - mx_1$$
- So, the line equation is

$$y = mx + y_1 - mx_1$$

10:35

CS770/870 Fall 2008 Bergeron/Plumlee

6

Line / window edge intersection

- Solve 2 (simple) simultaneous equations
- Left edge intersection ($x = x_{\min}$)

$$y = mx_{\min} + b$$
- Right edge intersection ($x = x_{\max}$)

$$y = mx_{\max} + b$$
- Top edge intersection ($y = y_{\max}$)

$$y_{\max} = mx + b, \text{ or } x = (y_{\max} - b) / m$$
- Bottom edge intersection ($y = y_{\min}$)

$$y_{\min} = mx + b, \text{ or } x = (y_{\min} - b) / m$$

10:35

CS770/870 Fall 2008 Bergeron/Plumlee

7

High-level Algorithm

```

boolean lineClip( Point p1, Point p2 )
    out1 = outcode( p1 ); out2 = outcode( p2 );
    while ( true ) // ugly!
        if out1 == 0 && out2 == 0
            return true // trivial accept
        if out1 & out2 == 0 // boolean and not a conditional evaluation
            return false // trivial reject
        if out1 == 0
            swap( p1, p2 ); swap( out1, out2 ); // guarantee that p1 is outside window
        if ( out1 & leftBit ) // leftBit = 1000
            intersectLine with left boundary; update p1 and out1
        else if ( out1 & topBit ) // topBit = 0100
            intersectLine with top boundary; update p1 and out1
        else if ( out1 & rightBit ) // rightBit = 0010
            intersectLine with right boundary; update p1 and out1
        else if ( out1 & bottomBit ) // bottomBit = 0001
            intersectLine with bottom boundary; update p1 and out1
    
```

10:35

CS770/870 Fall 2008 Bergeron/Plumlee

8