Computer Graphics

Week 4
Lecture 2

Line Clipping II

REF:

Hearn and Baker book

Line basics

Parametric Equation of line:

$x = x1 + u * dx$

$y = y1 + u * dy$ for $0 \leq u \leq 1$

where

$dx = x2 - x1$

$dy = y2 - y1$

Line basics

A point on line is inside clipping window if :

$$xw_{min} \le x_1 + u\Delta x \le xw_{max}$$

$$yw_{min} \le y_1 + u\Delta y \le yw_{max}$$

This can also be expressed as:

$$up_k \le q_k, \qquad k = 1, 2, 3, 4$$

where

$p_1 = -\Delta x, \quad q_1 = x_1 - xw_{min}$ ....(Left edge eq.)

$p_2 = \Delta x, \quad q_2 = xw_{max} - x_1$ ....(Right edge eq.)

$p_3 = -\Delta y, \quad q_3 = y_1 - yw_{min}$ ....(Bottom edge eq.)

$p_4 = \Delta y, \quad q_4 = yw_{max} - y_1$ ....(Top edge eq.)

we will use k=1,2,3 and 4 for left, right , bottom and top edge respectively

Liang Barsky Algo

<u>Observations:</u>

- ➔ If $p_k = 0$ then line is parallel to corresponding edge
    - If for that corresponding k , $q_k < 0$ then line is outside clipping window
    - If for that corresponding k , $q_k >= 0$ then line is inside the boundary
- ➔ If $p_k < 0$, then infinite extension of line travels from outside to inside of infinite extension of corresponding clipping boundary
- ➔ If $p_k > 0$, then infinite extension of line travels from inside to outside of infinite extension of corresponding clipping boundary

Liang Barsky Algo: intro

For clipping candidate lines we can find u that defines the intersection point of line with clipping boundary

$$u = q_k/p_k$$

For each line we can find 2 parameters u1 and u2 that define the part of line that lies inside the clipping window

Finding u1 and u2

Finding u1	Finding u2
Condition: Use only when $p_k<0$ only (boundaries where line is coming from outside to inside)	Condition: Use only when $p_k>0$ only (boundaries where line is going from inside to outside)
Compute $r_k = q_k/p_k$ for each edge for which $p_k <0$, then $u1 = \max [0,r_k]$	Compute $r_k = q_k/p_k$ for each edge for which $p_k >0$, then $u2 = \min [1,r_k]$

- If u1 > u2 then line segment is outside clipping window → no need to draw
- else calculate points from values of u1 and u2 and draw them

Summary of Algo

1. Initialize u1=0 and u2=1
2. Calculate p_k and q_k for each clipping boundary (k =1,2,3,4)
3. If $p_k<0$ then calculate $r_k = q_k/p_k$, and use it to update u1
4. If $p_k>0$ then calculate $r_k = q_k/p_k$, and use it to update u2
   a. If at any point u1 > u2 reject the line
5. If $p_k = 0$ and $q_k < 0$, reject the line
6. Lastly, if the line is not rejected then plot line based on points calculated from u1 and u2

Pseudo code

```c
#include "graphics.h"

#define ROUND(a) ((int)(a+0.5))

int clipTest (float p, float q, float * u1, float * u2)

{
  float r;
  int retVal = TRUE;

  if (p < 0.0) {
    r = q / p;
    if (r > *u2)
      retVal = FALSE;
    else
      if (r > *u1)
        *u1 = r;
  }
  else
    if (p > 0.0) {
      r = q / p;
      if (r < *u1)
        retVal = FALSE;
      else if (r < *u2)
        *u2 = r;
    }
    else
      /* p = 0, so line is parallel to this clipping edge */
      if (q < 0.0)
        /* Line is outside clipping edge */
        retVal = FALSE;

  return (retVal);
}
```

```
void clipLine (dcPt winMin, dcPt winMax, wcPt2 p1, wcPt2 p2)
{
  float u1 = 0.0, u2 = 1.0, dx = p2.x - p1.x, dy;

  if (clipTest (-dx, p1.x - winMin.x, &u1, &u2))
    if (clipTest (dx, winMax.x - p1.x, &u1, &u2)) {
      dy = p2.y - p1.y;
      if (clipTest (-dy, p1.y - winMin.y, &u1, &u2))
        if (clipTest (dy, winMax.y - p1.y, &u1, &u2)) {
          if (u2 < 1.0) {
            p2.x = p1.x + u2 * dx;
            p2.y = p1.y + u2 * dy;
          }
          if (u1 > 0.0) {
            p1.x += u1 * dx;
            p1.y += u1 * dy;
          }
          lineDDA (ROUND(p1.x), ROUND(p1.y), ROUND(p2.x), ROUND(p2.y));
        }
    }
}
```

Examples on board

The End

Text

P

Text

Text

The End