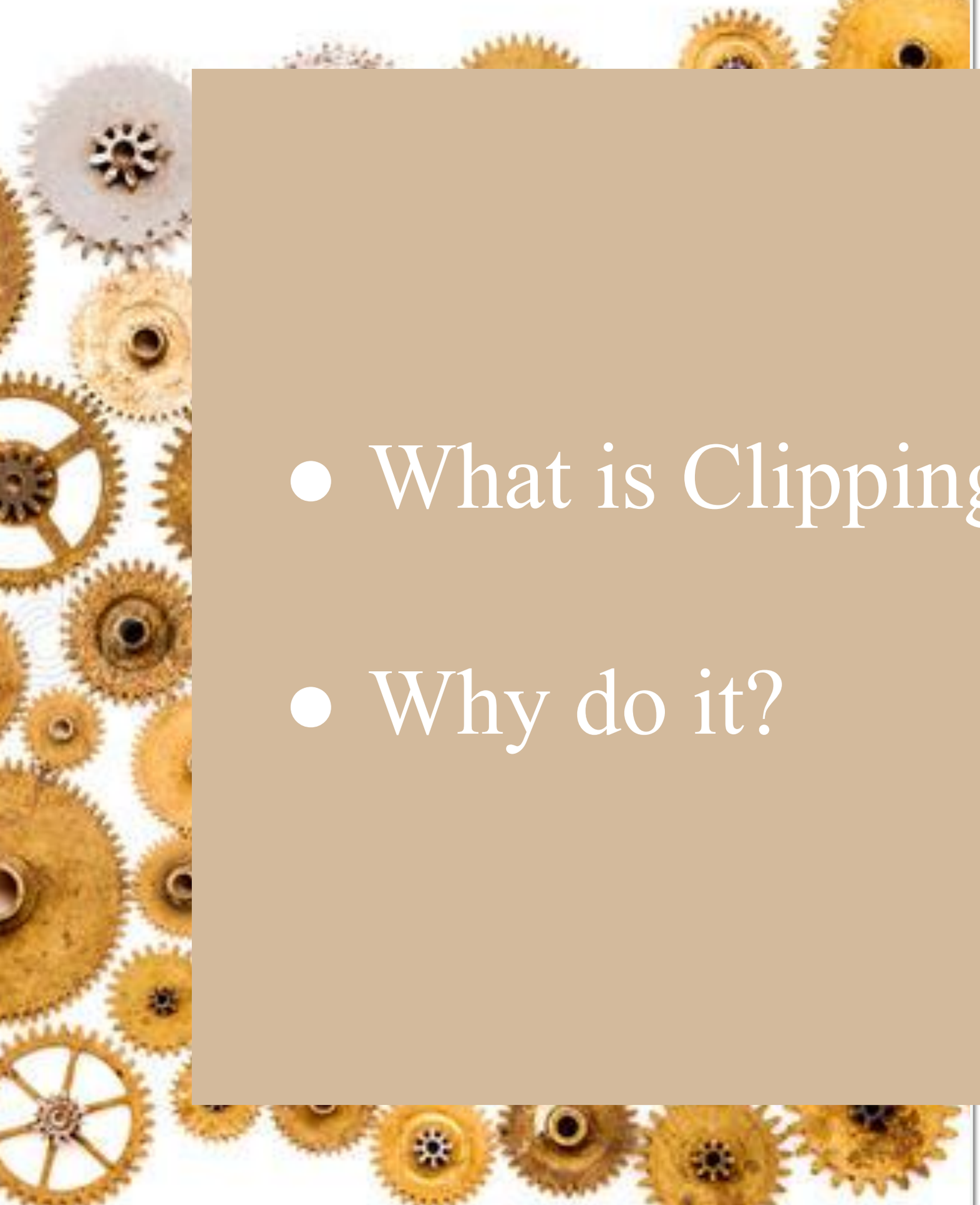
Computer Graphics

Week 4

Lecture 1

Line Clipping I



- What is Clipping?
- Why do it?

Point Clipping

A point (x,y)is **inside** the clipping window if it satisfies

$$x_{min} \leq x \leq x_{max}$$
$$y_{min} \leq y \leq y_{max}$$

Line Clipping

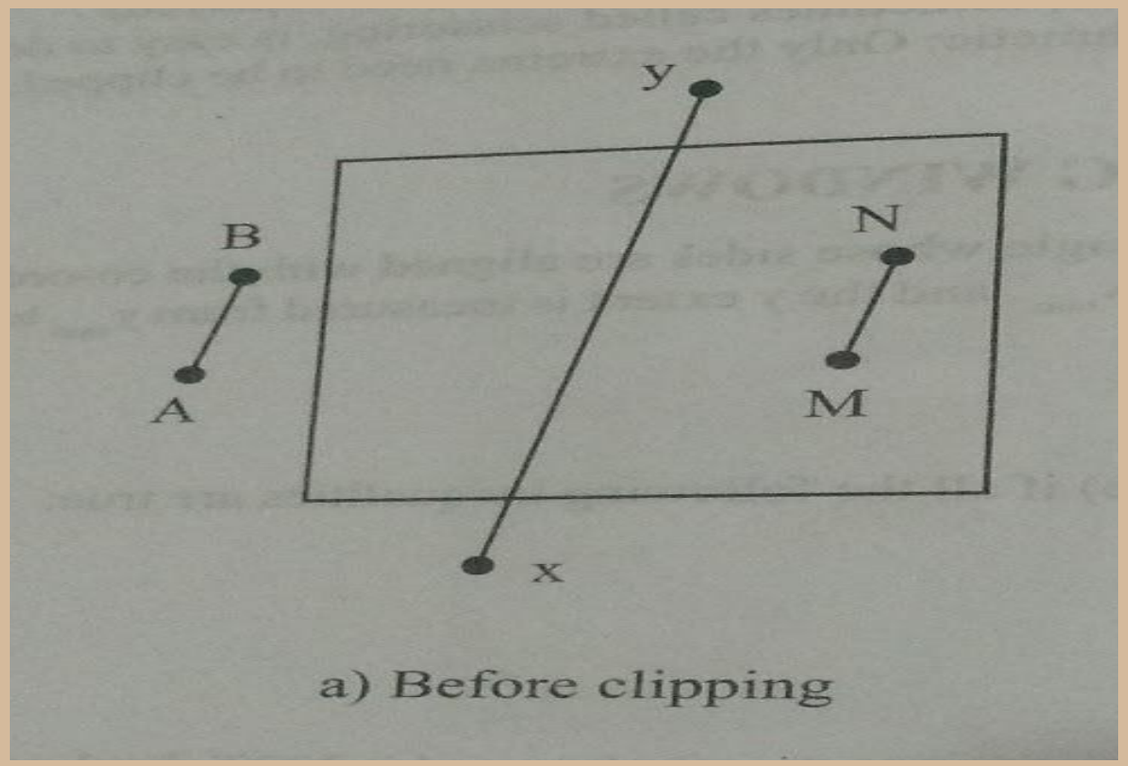
Two Phases:

- Identify the lines that intersect the window and so needs to be clipped
- Perform the clipping

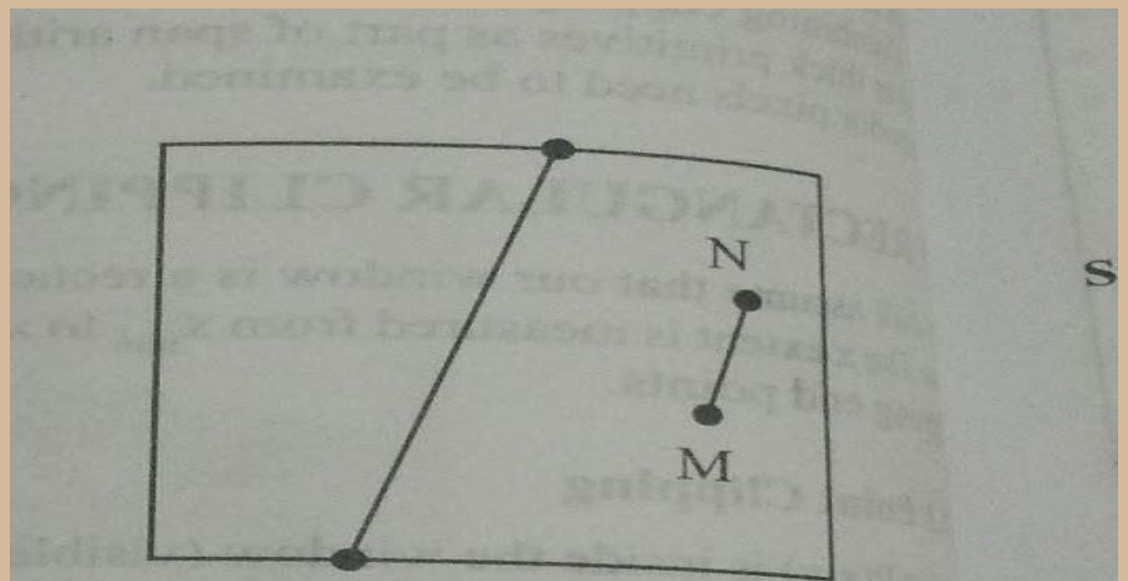
3 Clipping Categories

All lines segments fall into 3 categories:

1. Trivially accepted (Visible)
2. Trivially rejected (Not Visible)
3. Clipping Candidate



a) Before clipping



b) After clipping

Cohen-Sutherland Algo

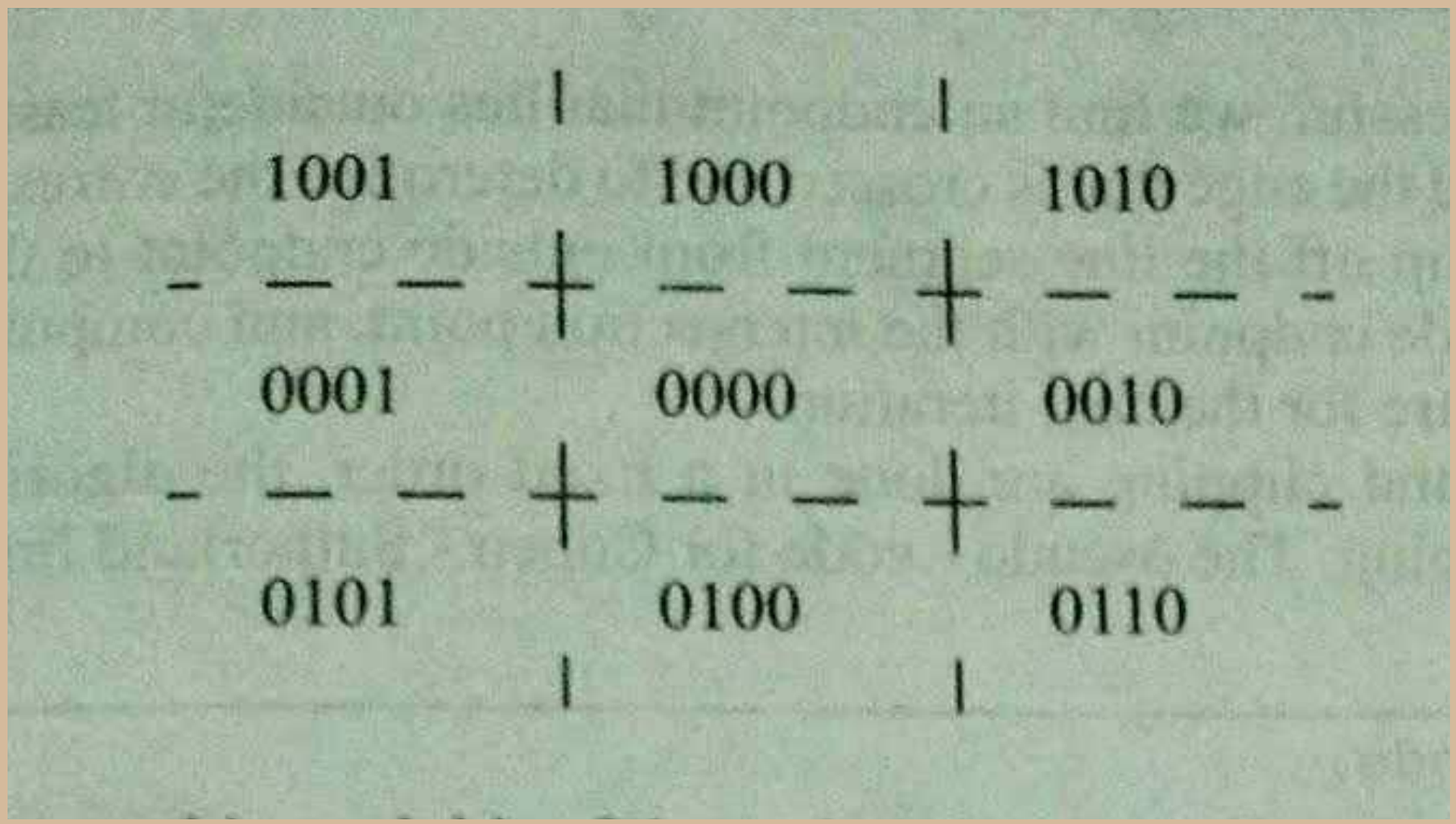
Step 1:

Assign a 4 bit code (outcode) to the end-points of the line.

If the endpoint is :

above top edge of window (y>ymax) put Bit1 = 1
below bottom edge of window (y<ymin) put Bit2 = 1
to right of right edge of window (x>xmax) put Bit3 = 1
to left of left edge of window (x>xmin) put Bit4 = 1

	Top	Bottom	Right	Left
Bit	1	2	3	4


1001
1000
1010
0001
0000
0010
0101
0100
0110

Cohen-Sutherland Algo

Step 2:

Classify the line

Line is:

Visible	if both point's code is 0000
NOT Visible	if logical AND of both points is **not 0000**
Clipping Candidate	if logical AND of both points is 0000

Cohen-Sutherland Algo

Step 3:

```
If clipping Candidate find the intersection
 points with the window
```

If :

Bit1 = 1 → intersects with line y=ymax

Bit2 = 1 → intersects with line y=ymin

Bit3 = 1 → intersects with line x=xmax

Bit4 = 1 → intersects with line x=xmin

if value of x (Case 3 & 4)is known find y by $y=y0 + m(x - x0)$

if value of y (Case 1 & 2)is known find x by $x=x0 + (1/m)(y - y0)$

Pseudo Code

```
typedef int OutCode;
enum {TOP = 0x8, BOTTOM = 0x4, RIGHT = 0x2, LEFT = 0x1};
// Compute the bit code for a point (x, y) using the clip rectangle
// bounded diagonally by (xmin, ymin), and (xmax, ymax)

// ASSUME THAT xmax, xmin, ymax and ymin are global constants.

OutCode ComputeOutCode(double x, double y)

{
OutCode code = 0;
// initialised as being inside of clip window

        if (x < xmin)                   // to the left of clip window
                code |= LEFT;
        else if (x > xmax)              // to the right of clip window
                code |= RIGHT;
        if (y < ymin)                   // below the clip window
                code |= BOTTOM;
        else if (y > ymax)              // above the clip window
                code |= TOP;
        return code;
}
// Cohen-Sutherland clipping algorithm clips a line from
// P0 = (x0, y0) to P1 = (x1, y1) against a rectangle with
// diagonal from (xmin, ymin) to (xmax, ymax).
void CohenSutherlandLineClipAndDraw(double x0, double y0, double x1,
double y1)

{
/* compute outcodes for P0, P1, and whichever point lies outside the
clip rectangle */
Outcode outcode0, outcode1, outcodeOut ;
        OutCode outcode0 = ComputeOutCode(x0, y0);
        OutCode outcode1 = ComputeOutCode(x1, y1);
        bool accept = FALSE, done = FALSE;

        do {
if (!(outcode0 | outcode1)) {   //Trivially accept and exit
                accept = TRUE;
```

```
                    done = TRUE ;

                } else if (outcode0 & outcode1) {
                    done = TRUE ;
                } else {
// failed both tests, so calculate the line segment to clip
// from an outside point to an intersection with clip edge
                        double x, y;
// At least one endpoint is outside the clip rectangle; pick it.

  OutCode outcodeOut = outcode0? outcode0 : outcode1;

// Now find the intersection point;use formulas
// y = y0 + slope * (x - x0), x = x0 + (1 / slope) * (y - y0)
if (outcodeOut & TOP) {                    .// point is above the clip rectangle
x = x0 + (x1 - x0) * (ymax - y0) / (y1 - y0);
y = ymax;
                            }
    else if (outcodeOut & BOTTOM) {
// point is below the clip rectangle
x = x0 + (x1 - x0) * (ymin - y0) / (y1 - y0);
y = ymin;                           }
    else if (outcodeOut & RIGHT) {
// point is to the right of clip rectangle
y = y0 + (y1 - y0) * (xmax - x0) / (x1 - x0);
x = xmax;                           }
    else if (outcodeOut & LEFT) {
// point is to the left of clip rectangle
y = y0 + (y1 - y0) * (xmin - x0) / (x1 - x0);
x = xmin;                           }
// Now we move outside point to intersection point to clip
// and get ready for next pass.
                            if (outcodeOut == outcode0) {
                                x0 = x;
                                y0 = y;
                            outcode0 = ComputeOutCode(x0, y0);
                            } else {
                                x1 = x;
                                y1 = y;
                            outcode1 = ComputeOutCode(x1, y1);
                            }
                }
        }while(done == FALSE);
```

Example

For the window extending from L(-3,-2) and R(2,3), find intersection for line extending from:

- S(-4,-1) and T(3,-2)

Deficiencies ??

The End

Slide 1

Text

TEX

T

[TEXT]

Text

The End