gradeup

Computer Graphics Part -3

Prep Smart. Stay Safe. go Gradeup

gradeup.co

Computer Graph Part-3

Content:

1. **Viewing and Clipping**
2. **Breserhens Circle Algorithm**
3. **Mid Point Circle**

VIEWING AND CLIPPING

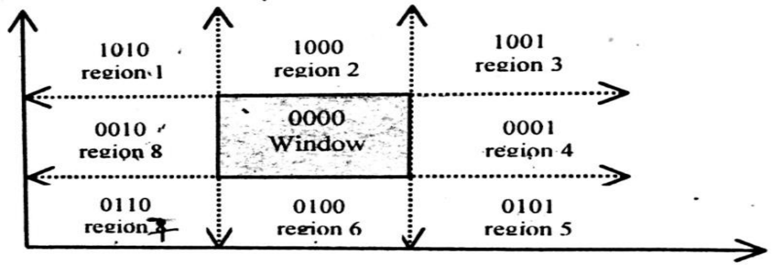
LINE CLIPPING

Line is a series of infinite number of points, where no two points have space in between them. So, the above said inequality also holds for every point on the line to be clipped. A variety of line clopping algorithms are available in the world of computer graphics, but we restrict our discussion to the following Line clipping algorithms, name after their respective developers:

1. Cohen Sutherland algorithm,
2. Cyrus-Back of algorithm

CHONE SUTHERLAND ALGORITHM

This algorithm is quite interesting. The clipping problem is simplified by dividing the area surrounding the window region into four segments Up, Down, Left, Right (U, D, L, R) and assignment of number 1 and 0 to respective segments helps in positioning the region surrounding the window. How this positioning of regions is performed can be well understood by considering below figure.



In above figure, you might have noticed, that all coding or regions U, D, L, R is done with respect to window region. As window is neither Up or Down, neither Left nor Right, so, the respective bits UDLR are 0000; now see region 1 of above figure. The positioning cod e UDLR is 1010, i.e., the region 1 lying on the position which is upper left side of the window. Region 1 has UDLR code 1010 (Up so U =1, not Down so D=0, Left so L = 1, not Right so R = 0).

The meaning of the UDLR code to state the location of region with respect to window is:

1st bit → Up(U); 2nd bit → Down(D); 3rd bit → Left(L); 4th bit → Right(R),

Now, to perform Line clipping for various line segment which may reside inside the window region fully or partially, or may not even lie in the widow region; we use the tool of logical ANDing between UDLR codes of the points lying on the line.

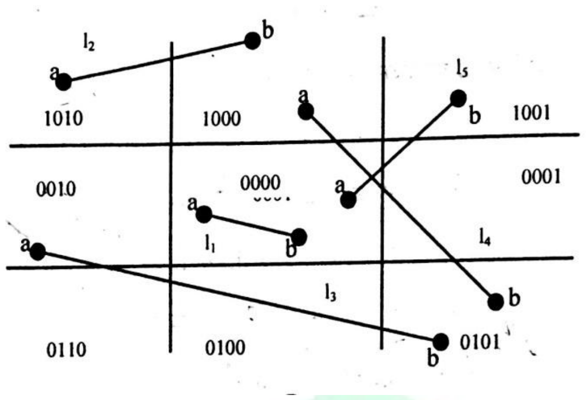
Logical ANDing ($\wedge$) operation

Between respective bits implies

Note:

- UDLR code of window is 0000 always and w.r.t this will create bit codes of other regions.
- A line segment is visible if both the UDLR codes of the end points of the line. Segment equal to 0000 i.e. UDLR code of window region. If the resulting code is not 0000 then, that line segment or section of line segment may or may not be visible.

Let us examine how this clipping algorithm works. For the sake of simplicity we will tackle all the cases with the help of example lines l_1 to l_5 shows in below figure. Each line segment represents a case.



Now, this in above figure, line l_1 is completely visible, l_2 and l_3 are completely invisible, l_4 and l_5 are partially visible. We will discuss these out comings as three separate cases.

Case 1: $l_1 \rightarrow$ Completely visible, i.e., Trival acceptance

($\therefore$ both points lie inside the window)

Case 2: l_2 and $l_3 \rightarrow$ Invisible, i.e., Trival acceptance rejection

Case 3: l_4 and $l_5 \rightarrow$ partially visible ($\therefore$ partially inside the window)

Now let us examine there three cases with the help of this algorithm:

Case 1: (Trival acceptance case) if the UDLR bit codes of the end points P, Q, of a given line is 0000 then line is completely visible. Here this is the case as the end points a an b of line l_1 are; a(0000), b (0000). If this trival acceptance test is failed then, the line segment PQ is passed onto Case 2.

Case 2: (Trival Rejection Case) if the logical intersection (AND) of the bit codes of the end points P, Q of the line segment is $\neq$ 0000 then line segment is not visible or is rejected.

Note that, in above figure, line 2 is completely on the top of the window but line 3 is neither on top or not at the in bottom plus, either on the LHS nor on the RHS of the window. We choose the standard formula of logical ANDing to test the non visibility of the line segment.

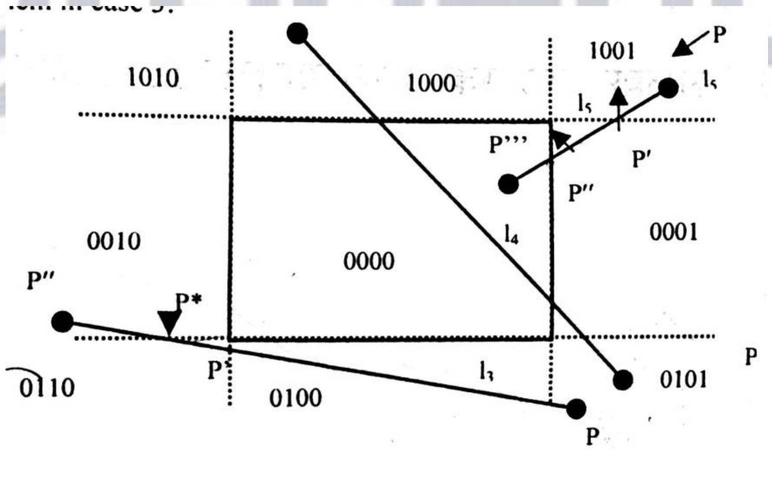
To test the visibility of line 2 and 3 we need to compute the logical intersection of end points for line 2 and line 3.

Line l2: bit code of end points are 1010 and 1000

Logical intersection of end points = (1010) ∧ (1000) = 1000

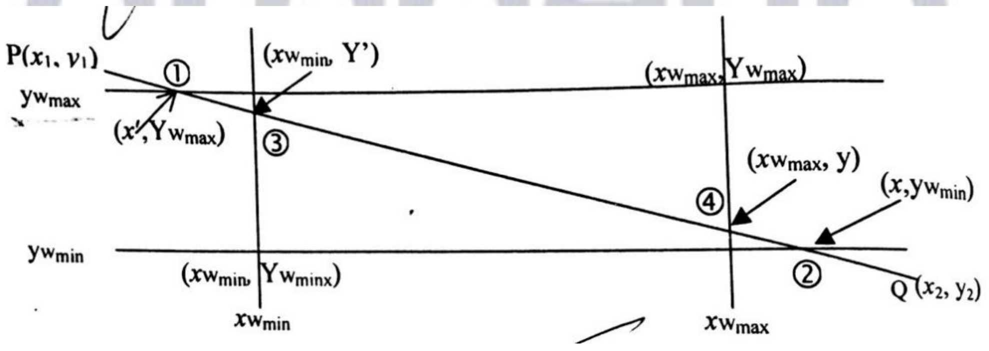
As logical intersection ≠ 0000. So line 2 will be invisible.

Line l3: end point have bit codes 0010 and 0101 now logical intersection = 0000, i.e., 0010 ∧ 0101 = 0000 from the above figure, the line is invisible. In line 4 one end point is on top and the other on the bottom so, logical intersection is 0000 but then it is partially visible, same is true with line 5. These are special cases and we will discuss them in case 3.



Case 3: Suppose for the line segment PQ, both the trivial acceptance and rejection tests failed (i.e., Case 1 and Case 2 conditions do not hold, this is the case for l_3, l_4, and l_5 line segments) shown in above figure. For such non-trivial Cases the algorithm is processed, as follows.

Both bitcodes for the end points P, Q of the line segment cannot be equal to 0000. Let us assume that the starting point of the line segment is P whose bit code is not equal to 0000. For example, for the line segment l_5 we choose P to be the bitcodes 1001. Now, scan the bitcodes of P from the first bit to the fourth bit and find the position of the bit at which the bit value 1 appears at the first time. For the line segment l_5 it appears at the very first position. If the bit value 1 occurs at the first position then proceed to intersect the line segment with the UP edge of the window and assign the first bit value to its point of intersection as 0. Similarly, if the bit value 1 occurs at the second position while scanning the bit values at the first time then intersect the line segment PQ with the Down edge of the window and so on. This point of intersection may be labelled as P'. Clearly the line segment PP' is outside the window and therefore rejected and the new line segment considered for dipping will be P'Q. The coordinates of P' and its remaining new bit values are computed. Now, by taking P as P', again we have the new line segment PQ which will again be referred to case 1 for clipping.



Geometrical study of the above type of clipping (it helps to find point of intersection of line PQ with any edge).

Let (x_1, y_1) and (x_2, y_2) be the coordinates of P and Q respectively.

1. Top case/above

   If $y_1 > y^{w}_{max}$ then 1st bit of code = 1 (signifying above) else bit code = 0

2. Bottom case/below case

   If $y_1 > y^{w}_{max}$ then 2nd bit = 1 (i.e. below) else bit = 0

3. Left case: if $x_1 < x^{w}_{max}$ then 3rd bit = 1 (i.e., left) else 0

4. Right case: if $x_1 > x^{w}_{max}$ then 4th bit = 1 (i.e. right) else 0

   Similarly, the bit codes of the point Q will also be assigned.

Breserhens Circle Algorithm

1. Get the coordinate of the centre of the circle & radius & store then in

   x,y & R, Set P=0 & Q=R

2. Set decision parameter

   D= 3-2R

3. If D<0

   D=D+4P+6

   $Y_{K+1}=y$

   $X_{k+1}=x_k+1$

4. If D>0

   $X_{k+1}=x+1$

   Set $Y_{k+1} = Y-1$

   & D=D+4(P-Q)+10

5. Slop if x>y.

Midpoint Circle Algorithm

a. Input radius r and circle, center (x_c , y_c) and obtain the first point on the circumference of the circle centered on origin as

$$(x_o , y_o) = (0,r)$$

b. Calculate initial value of decision parameter as

$$p_o = 5/4 - r - l - r$$

c. At each x_k position starting at k = 0 perform following test.

1. If $P_k < 0$ then next point along the circle centered on(0,0) is (x_{k+1}, y_k) and $P_{k+1} = P_k + 2 x_{K+1} + 1$
2. Else the next point along circle is $(x_{k+1}, Y_k - 1)$ and

   $P_{K+1} = P_k + 2 X_{K+1} - 2 Y_{k-1}$

   Where $2 X_{k+1} = 2 (X_k + 1) = 2 X_k + 2$ and $2 (Y_k - 1) = 2Y_k - 2$

d. Determine symmetry points in the other seven octants

e. Move each calculated pixel position(x,y) onto the circular path centred on (X_c, Y_c) and plot coordinate values $x = x + x_c$, $Y = Y + Y_c$

f. Repeat step (c) and (e) until $x \geq y$.

Example: Given a circle radius r =10 determine positions along the circle octants in 1st Quadrant from x= 0 to x = y

Solution: An initial decision parameter $p_0 = 1-r = 1-10 = -9$.

For circle centerded on coordinate origin the initial point$(X_0, Y_0) = (0,10)$ and initial increment for calculating decision parameter are :

$2 X_0 = 0$, $2Y_0 = 20$

Using mid point algorithm point are:

Q_0	P_k	(X_{k+1}, Y_{k+1})	$2\ X_{k+1}$	$2\ Y_{K-1}$
0	-9	(1,10)	2	20
1	-6	(2,10)	4	20
2	-1	(3,10)	6	20
3	6	(4,9)	8	18
4	-3	(5,9)	10	18
5	8	(6,8)	12	16
6	5	(7,7)	14	14