**Practical No. 15**

***Title****:-*

|  |
| --- |
| Write a C program for Sutherland Hodgeman Polygon Clipping. |

***Relevant Course Outcome(s):-***

Implement Various Clipping algorithms and given curve generating algorithms.

***Resources Required (Hardware & Softwares):-***

A Desktop PC/ Laptop

Ansi C/ Turbo C/ (Any distribution) installed

***Theory****:-*

***Polygon Clipping***

****

***Sutherland-Hodgeman Polygon Clipping***

Processing the polygon boundary as a whole against each window edge

Processing all polygon vertices against each clip rectangle boundary in turn

****

Pass each pair of adjacent polygon vertices to a window boundary clipper

There are four cases:

****

**V1-outside V2-Inside V1 & V2 – Inside V1-Inside V2-Outside V1& V2- Outside**

Intermediate output vertex list

Once all vertices have been processed for one clip window boundary, it is generated.

The output list of vertices is clipped against the next window boundary.

It can be eliminated by a pipeline of clipping routine.

***Algorithm :-***

1. Read coordinates of all vertices of the polygon

2. Read coordinates of the clipping window

3. Consider the left edge of the window

4. Compare the vertices of each edge of the polygon, individually with the clipping plane.

5. Save the resulting intersections and vertices in the new list of vertices according to four possible relationships between the edge and the clipping boundary discussed earlier.

6. Repeat the steps 4 and 5 for remaining edges of the clipping window. Each time the resultant list of vertices is successively passed to process the next edge of the clipping window.

7. Stop.

**Program for Sutherland-Hodgeman Algorithm**

#include<stdio.h>

#include<graphics.h>

#include<math.h>

typedef struct

{

float x;

float y;

}PT;

int n;

main()

{

int i,j,gd=DETECT,gm;

PT d,p1,p2,p[20],pi1,pi2,pp[20];

detectgraph(&gd,&gm);

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

/\* Read coordinates of clipping window

----------------------------------------- \*/

printf("Enter coordinates (left,top) of point1 : ");

scanf("%f %f",&p1.x,&p1.y);

printf("Enter coordinates (right,bottom) of point2 : ");

scanf("%f %f",&p2.x,&p2.y);

/\* Enter the number of vertex

------------------------------ \*/

printf("Enter the number of vertex : ");

scanf("%d",&n);

/\* Read vertex coordinates of clipping window

----------------------------------------- \*/

for(i=0;i<n;i++)

{

printf("Enter coordinates of vertex%d : ",i+1);

scanf("%f %f",&p[i].x,&p[i].y);

}

p[i].x = p[0].x;

p[i].y = p[0].y;

cleardevice();

drawpolygon(p,n);

rectangle(p1.x,p1.y,p2.x,p2.y);

getch();

left(p1,p,pp);

right(p2,p,pp);

top(p1,p,pp);

bottom(p2,p,pp);

cleardevice();

rectangle(p1.x,p1.y,p2.x,p2.y);

drawpolygon(p,n);

getch();

closegraph();

}

left(PT p1,PT p[20],PT pp[20])

{

int i,j=0;

for(i=0;i<n;i++)

{

if(p[i].x < p1.x && p[i+1].x >= p1.x)

{

if(p[i+1].x-p[i].x!=0)

{

pp[j].y = (p[i+1].y-p[i].y)/(p[i+1].x-p[i].x)\* (p1.x-p[i].x)+p[i].y;

}

else

{

pp[j].y = p[i].y;

}

pp[j].x = p1.x;

j++;

pp[j].x=p[i+1].x;

pp[j].y=p[i+1].y;

j++;

}

if(p[i].x > p1.x && p[i+1].x >= p1.x)

{

pp[j].y = p[i+1].y;

pp[j].x = p[i+1].x;

j++;

}

if(p[i].x > p1.x && p[i+1].x <= p1.x)

{

if(p[i+1].x-p[i].x!=0)

{

pp[j].y = (p[i+1].y-p[i].y)/(p[i+1].x-p[i].x)\* (p1.x-p[i].x)+p[i].y;

}

else

{

pp[j].y = p[i].y;

}

pp[j].x = p1.x;

j++;

}

}

for(i=0;i<j;i++)

{

p[i].x = pp[i].x;

p[i].y = pp[i].y;

}

p[i].x = pp[0].x;

p[i].y = pp[0].y;

n=j;

}

right(PT p2,PT p[20],PT pp[20])

{

int i,j=0;

for(i=0;i<n;i++)

{

if(p[i].x > p2.x && p[i+1].x <= p2.x)

{

if(p[i+1].x-p[i].x!=0)

{

pp[j].y = (p[i+1].y-p[i].y)/(p[i+1].x-p[i].x)\* (p2.x-p[i].x)+p[i].y;

}

else

{

pp[j].y = p[i].y;

}

pp[j].x = p2.x;

j++;

pp[j].x=p[i+1].x;

pp[j].y=p[i+1].y;

j++;

}

if(p[i].x < p2.x && p[i+1].x <= p2.x)

{

pp[j].y = p[i+1].y;

pp[j].x = p[i+1].x;

j++;

}

if(p[i].x < p2.x && p[i+1].x >= p2.x)

{

if(p[i+1].x-p[i].x!=0)

{

pp[j].y = (p[i+1].y-p[i].y)/(p[i+1].x-p[i].x)\* (p2.x-p[i].x)+p[i].y;

}

else

{

pp[j].y = p[i].y;

}

pp[j].x = p2.x;

j++;

}

}

for(i=0;i<j;i++)

{

p[i].x = pp[i].x;

p[i].y = pp[i].y;

}

p[i].x = pp[0].x;

p[i].y = pp[0].y;

n=j;

}

top(PT p1,PT p[20],PT pp[20])

{

int i,j=0;

for(i=0;i<n;i++)

{

if(p[i].y < p1.y && p[i+1].y >= p1.y)

{

if(p[i+1].y-p[i].y!=0)

{

pp[j].x = (p[i+1].x-p[i].x)/(p[i+1].y-p[i].y)\* (p1.y-p[i].y)+p[i].x;

}

else

{

pp[j].x = p[i].x;

}

pp[j].y = p1.y;

j++;

pp[j].x=p[i+1].x;

pp[j].y=p[i+1].y;

j++;

}

if(p[i].y > p1.y && p[i+1].y >= p1.y)

{

pp[j].y = p[i+1].y;

pp[j].x = p[i+1].x;

j++;

}

if(p[i].y > p1.y && p[i+1].y <= p1.y)

{

if(p[i+1].y-p[i].y!=0)

{

pp[j].x = (p[i+1].x-p[i].x)/(p[i+1].y-p[i].y)\* (p1.y-p[i].y)+p[i].x;

}

else

{

pp[j].x = p[i].x;

}

pp[j].y = p1.y;

j++;

}

}

for(i=0;i<j;i++)

{

p[i].x = pp[i].x;

p[i].y = pp[i].y;

}

p[i].x = pp[0].x;

p[i].y = pp[0].y;

n=j;

}

bottom(PT p2,PT p[20],PT pp[20])

{

int i,j=0;

for(i=0;i<n;i++)

{

if(p[i].y > p2.y && p[i+1].y <= p2.y)

{

if(p[i+1].y-p[i].y!=0)

{

pp[j].x = (p[i+1].x-p[i].x)/(p[i+1].y-p[i].y)\* (p2.y-p[i].y)+p[i].x;

}

else

{

pp[j].x = p[i].x;

}

pp[j].y = p2.y;

j++;

pp[j].x=p[i+1].x;

pp[j].y=p[i+1].y;

j++;

}

if(p[i].y < p2.y && p[i+1].y <= p2.y)

{

pp[j].y = p[i+1].y;

pp[j].x = p[i+1].x;

j++;

}

if(p[i].y < p2.y && p[i+1].y >= p2.y)

{

if(p[i+1].y-p[i].y!=0)

{

pp[j].x = (p[i+1].x-p[i].x)/(p[i+1].y-p[i].y)\* (p2.y-p[i].y)+p[i].x;

}

else

{

pp[j].x = p[i].x;

}

pp[j].y = p2.y;

j++;

}

}

for(i=0;i<j;i++)

{

p[i].x = pp[i].x;

p[i].y = pp[i].y;

}

p[i].x = pp[0].x;

p[i].y = pp[0].y;

n=j;

}

drawpolygon(PT x[20],int n)

{

int i;

for(i=0;i<n-1;i++)

{

line(x[i].x,x[i].y,x[i+1].x,x[i+1].y);

}

line(x[i].x,x[i].y,x[0].x,x[0].y);

}

**Output:- ( Paste your own Output )**

***Conclusion:-*** Thus, we have Implemented a C program for Line Clipping using Cohen-Sutherland algorithm.