Experiment No : 10

Aim : To draw a polygon and clip it using Sutherland Hodgeman Polygon Clipping Algorithm

Theory :
Polygon clipping is the process of finding the portion of a give convex polygon inside of a given convex polygon. It is widely used in 2D and 3D animation. The Sutherland Hodgeman Polygon Clipping Algorithm is one such algorithm.

Algorithm :
1. Begin with an input list of all vertices in the subject polygon
2. One side of the clipping boundary is extended infinitely in both directions, and the path of the subject polygon is traversed.
3. Vertices from the input list are inserted into an output list if they lie on the visible side of the extended clip polygon line, and new vertices are added to the output list where the subject polygon path crosses the extended clip polygon line.

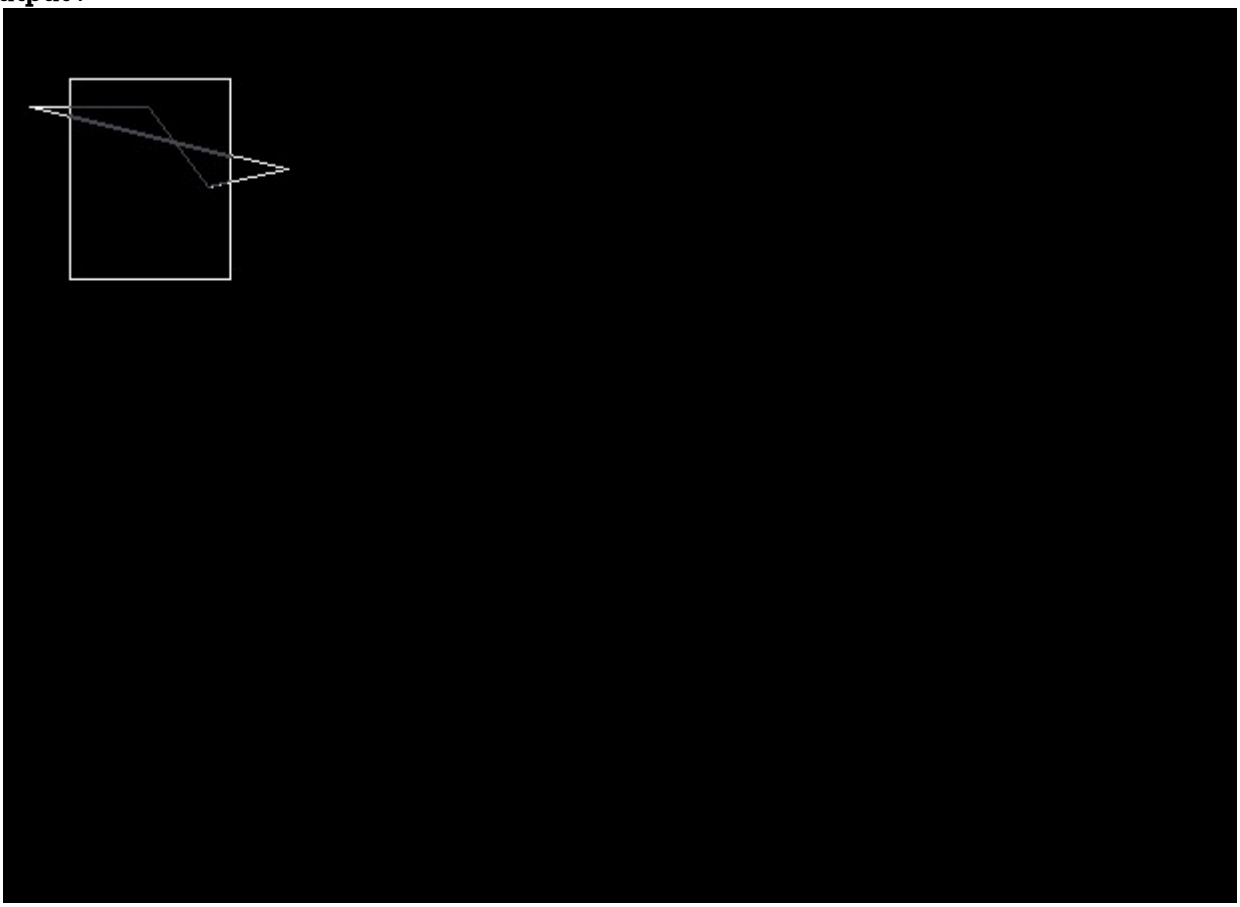
Code & Output :
```
#include <math.h>
#include<iostream>
#include<graphics.h>
#define ROUND(a) ((int)(a+0.5))
#define n 4

#define LEFT_EDGE 0x1
#define RIGHT_EDGE 0x2
#define BOTTOM_EDGE 0x4
#define TOP_EDGE 0x8

#define INSIDE(a)  (!a)
#define REJECT(a,b) (a&b)
#define ACCEPT(a,b) (!(a|b))

typedef struct wcpt2
{
   int x,y;
}wcpt2;

typedef struct dcpt
{
   int x,y;
}dcpt;

int main()
{
   int gd=DETECT,gm;
   int left,top,right,bottom;
   int x1,x2,y1,y2;
   int maxx, maxy;

   int poly[10];
   void clipline(dcpt,dcpt,wcpt2,wcpt2);

   initgraph(&gd,&gm,NULL);
   maxx = getmaxx()/4;
   maxy = getmaxy()/4;
```

```
    poly[0] = 20;
    poly[1] = maxy / 2;

    poly[2] = maxx - 10;
    poly[3] = 90;

    poly[4] = maxx - 50;
    poly[5] = maxy - 20;

    poly[6] = maxx / 2;
    poly[7] = maxy / 2;


    poly[8] = poly[0];
    poly[9] = poly[1];

    drawpoly(5, poly);

    rectangle(40,45,120,145);
    wcpt2 pt1,pt2;
    dcpt winmin,winmax;

    winmin.x=40;
    winmin.y=45;
    winmax.x=120;
    winmax.y=145;

    pt1.x=20;
    pt1.y=maxy/2;
    pt2.x=maxx-10;
    pt2.y=10;

    int i=0;
    for(int index=0;index<n;index++)
    {
       if(index==n-1)
       {
          pt1.x=poly[i];
          pt1.y=poly[i+1];
          i=0;
          pt2.x=poly[i];
          pt2.y=poly[i+1];
          clipline(winmin,winmax,pt1,pt2);
       }
       else
       {
          pt1.x=poly[i];
          pt1.y=poly[i+1];
          pt2.x=poly[i+2];
          pt2.y=poly[i+3];
          clipline(winmin,winmax,pt1,pt2);
       }
       i+=2;
    }
    pt1.x=poly[i];
    pt1.y=poly[i+1];
    clipline(winmin,winmax,pt1,pt2);
    getch();
}
```

```
unsigned char encode(wcpt2 pt,dcpt winmin,dcpt winmax)
{
   unsigned char code=0x00;
   if(pt.x < winmin.x)
      code=code | LEFT_EDGE;
   if(pt.x > winmax.x)
      code=code | RIGHT_EDGE;
   if(pt.y < winmin.y)
      code=code | TOP_EDGE;
   if(pt.y > winmax.y)
      code=code | BOTTOM_EDGE;
   return code;
}


void swappts(wcpt2 *p1,wcpt2 *p2)
{
   wcpt2 tmp;
   tmp = *p1;
   *p1 = *p2;
   *p2 = tmp;
}


void swapcode(unsigned char *c1,unsigned char *c2)
{
   unsigned char tmp;
   tmp = *c1;
   *c1 = *c2;
   *c2 = tmp;
}


void clipline(dcpt winmin,dcpt winmax,wcpt2 p1,wcpt2 p2)
{
   unsigned char encode(wcpt2,dcpt,dcpt);
   unsigned char code1,code2;
   int done = 0 , draw = 0;
   float m;
   void swapcode(unsigned char *c1,unsigned char *c2);
   void swappts(wcpt2 *p1,wcpt2 *p2);

   while(!done)
   {
      code1 = encode(p1,winmin,winmax);
      code2 = encode(p2,winmin,winmax);
      if(ACCEPT(code1,code2))
      {
         draw = 1;
         done = 1;
      }
      else if(REJECT(code1,code2))
         done = 1;
      else if(INSIDE(code1))
      {
            swappts(&p1,&p2);
            swapcode(&code1,&code2);
      }
      if(code1 & LEFT_EDGE)
```

```
        {
            p1.y += (winmin.x - p1.x) *  (p2.y - p1.y) / (p2.x - p1.x);
            p1.x = winmin.x;
        }
        else if(code1 & RIGHT_EDGE)
        {
            p1.y += (winmax.x - p1.x) *  (p2.y - p1.y) / (p2.x - p1.x);
            p1.x = winmax.x;
        }
        else if(code1 & TOP_EDGE)
        {
           if(p2.x != p1.x)
            p1.x += (winmin.y - p1.y) *  (p2.x - p1.x) / (p2.y - p1.y);
            p1.y = winmin.y;
        }
        else if(code1 & BOTTOM_EDGE)
        {
           if(p2.x != p1.x)
            p1.x += (winmax.y - p1.y) *  (p2.x - p1.x) / (p2.y - p1.y);
            p1.y = winmax.y;
        }
    }
    if(draw)
    {
    setcolor(8);
    line(p1.x,p1.y,p2.x,p2.y);
    }
}
```

Output :



Conclusion : Program to draw a polygon and clip it using Sutherland Hodgeman Polygon Clipping Algorithm was successfully written and executed

Deepraj Bhosale
181105016