Name – Isha Kishor Katariya

Class - SEIT Div - A

Roll No – 207A056

Experiment - Implement Cohen Sutherland polygon clipping method to clip the polygon with respect the viewport and window. Use mouse click, keyboard interface.

#include<iostream>

#include<GL/glut.h> #include<math.h> using namespace std; int xl=50,xh=200,yl=50,yh=200; int flag=0; float u1,v1,u2,v2;

//defining the structure code to get - opcodes. struct code

{ int t,b,r,l;

}; void init() {

//set the background color.

glClearColor(1,1,1,0);

//activate the color\_buffer\_bit and assign the background color mentioned in glClearColor.

glClear(GL\_COLOR\_BUFFER\_BIT);

//set the pixel color as black. glColor3f(0,0,0); }

//get the opcodes (tbrl). code get\_code(int u,int v) { code c={0,0,0,0}; if(u<xl) c.l=1; if(u>xh)

c.r=1; if(v<yl)

c.b=1; if(v>yh)

c.t=1; return c;

}

/\*

// #BRESENHEM LINE DRAWING ALOGORITHM void line(int u1,int v1,int u2,int v2)

{ int dx,dy,p,xi=1,yi=1;

dx=u2-u1; dy=v2v1;

if(dx<0)

{

dx=-dx; xi=-1; }

if(dy<0)

{

dy=-dy; yi=-1; }

glBegin(GL\_POINTS); glVertex2i(u1,v1); if(dx>dy)

{ p=(2\*dy)-dx; while(u1!=u2)

{ if(p<=0) { p+=2\*dy; } else { p+=2\*(dy-dx); v1+=yi;

}

u1+=xi; glVertex2i(u1,v1);

}

} else {

p=(2\*dx)-dy; while(v1!=v2)

{ if(p<=0) { p+=2\*dx; } else { p+=2\*(dx-dy); u1+=xi;

}

v1+=yi; glVertex2i(u1,v1);

}

}

glEnd(); glFlush(); }

\*/

// #DDA-LINE DRAWING ALGORITHM

void line(float u1,float v1,float u2,float v2)

{ float dx,dy,x=u1,y=v1,xi,yi; int

steps,i;

dx=u2-u1; dy=v2v1;

steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);

xi=dx/(float)steps;

yi=dy/(float)steps;

glBegin(GL\_POINTS);

glVertex2f(x,y);

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

{

x+=xi; y+=yi;

glVertex2f(x,y); }

glEnd();

glFlush();

}

//draw the window void draw\_window() { line(50,50,200,50); line(50,50,50,200); line(200,50,200,200); line(50,200,200,200); }

//mouse function to draw the line. void mymouse(int button,int state,int x,int y)

{ glColor3f(0,0,0);

if(state==GLUT\_DOWN && flag==0)

{ u1=x; v1=480y; flag=1;

}

else if(state==GLUT\_DOWN && flag==1)

{ u2=x; v2=480y; flag=2; line(u1,v1,u2,v2);

}

}

//cohen-sutherland algorithm to clip the line void cohen()

{ code c1,c2,c; float m; int xi,yi,flag;

//get the slope of line m=(v2-v1)/(u2u1);

//get the opcodes of both the co-ordinates in c1 and c2. c1=get\_code(u1,v1); c2=get\_code(u2,v2); while(1)

{

//if line inside the window , draw the line as it is.

if( c1.t==0 && c2.t==0 && c1.b==0 && c2.b==0 && c1.r==0 && c2.r==0

&& c1.l==0 && c2.l==0 ) break;

//if the ANDING of opcodes is non-zero then don't draw the line. else if( ( (c1.t && c2.t) || (c1.b && c2.b) || (c1.r && c2.r) || (c1.l && c2.l) ) !

=0)

{ u1=v1=u2=v2=0; break;

}

//if line partially inside the window changing the co-ordinates as per following

conditions. else

{ if( c1.l==1 || c2.l==1)

{ xi=xl; yi=v1+m\*(xlu1)

; if(c1.l==1) flag=0; else flag=1;

} else if( c1.r==1 || c2.r==1 )

{ xi=xh; yi=v1+m\*(xhu1);

if(c1.r==1) flag=0; else flag=1; } else if( c1.b==1 || c2.b==1 )

{ xi=u1+((1/m)\*(yl-v1)); yi=yl; if(c1.b==1)

flag=0;

else

flag=1; } else if( c1.t==1 || c2.t==1 )

{ xi=u1+((1/m)\*(yh-v1)); yi=yh; if(c1.t==1) flag=0; else flag=1;

}

//get the code of xi and yi. c=get\_code(xi,yi); if(flag==0)

{ u1=xi; v1=yi; c1=c; } else if(flag==1)

{ u2=xi; v2=yi

;

c2=c;

}

}//end\_else

}//end\_while

//draw\_the window and clipped line.

draw\_window(); line(u1,v1,u2,v2);

} void mykey(char unsigned key,int x,int y)

{

//press 'c' to clip the line.

if(key=='c') {

init();

cohen(); }

//press 'r' to reset the window. if(key=='r')

{ init();

draw\_window();

flag=0;

} } int main(int argc,char \*\*argv) {

//initializing the glut-library.

glutInit(&argc, argv);

//set the display mode as GLUT\_SINGLE for single buffer window.

glutInitDisplayMode(GLUT\_SINGLE);

//set the size of window.

glutInitWindowSize(640,480);

//set the window position. glutInitWindowPosition(0,0); //creating the window and assigning the name. glutCreateWindow("Line\_Clipping");

//declaring the co-ordinates of ortho-2d function i.e getting the orthographic projection.

gluOrtho2D(0,640,0,480);

//initialize the window created with background color,set the pixel color.

init(); glFlush();

//draw the window. draw\_window();

//get the line using mouse. glutMouseFunc(mymouse);

//clip the line by pressing 'c' and also the reset the window by pressing 'r'.

glutKeyboardFunc(mykey);

//keeping the window open.

glutMainLoop();

return 0;

}//end\_main