**Pune Institute Of Computer Technology Dhankawadi,**

**Pune – 43.**

Assignment No. 5

Computer Graphics

**SE-IT-10 ACADEMIC YEAR :- 2020-2021**

**Name :- Diptesh Ravindra Varule Roll No :- 23277**

**Topic Name**:

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

Source Code:

/\*\* Sutherland Hodgeman Polygon Clipping

\* Usage - click on the canvas to seed the points of the polygon

\* press d - to draw the seeded polygon

\* drag the mouse cursor to draw the required clip rectangle

\* press c - to clip the required polygon section

\* NOTE - Clip rectangle must be drawn from top left to bottom right

\* Contains Debug Code to reconstruct the clipping in comments in SHPC function

\*/

#include <iostream>

#include <math.h>

#include <time.h>

#include <GL/glut.h>

#include <list>

using namespace std;

void init(){

glClearColor(1.0,1.0,1.0,1.0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,640,0,480);

}

int xmin = 0,ymin = 0,xmax = 0,ymax = 0;

int enter = 1,sz,st\_flag=1;

float\*\* pts;

class points{

int x;

int y;

public:

points(int x,int y){

this->x = x;

this->y = y;

}

int getx(){

return x;

}

int gety(){

return y;

}

};

class tryo{

int x;

int y;

public:

void setx(int x){this->x = x;}

int getx(){return x;}

};

points \*s, \*p;

list <points\*> in;

list <points\*> outer;

void delay(float ms){

clock\_t goal = ms + clock();

while(goal>clock());

}

void drawPolygon(){

//draw polygon and create the points array

glBegin(GL\_LINE\_LOOP);

pts = new float\*[in.size()];

for(int i=0; i<in.size(); i++){

pts[i] = new float[2];

}

sz = in.size();

while(in.size()>0){

points\* temp = in.front();

pts[in.size()-1][0] = temp->getx();

pts[in.size()-1][1] = temp->gety();

glVertex2i(temp->getx(),temp->gety());

in.pop\_front();

}

glEnd();

glFlush();

}

void redraw(){

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_LINE\_LOOP);

for(int i=0; i<sz; i++){

glVertex2i(pts[i][0],pts[i][1]);

}

glEnd();

glFlush();

glColor3f(0,0,0);

glBegin(GL\_LINE\_LOOP);

glVertex2i(xmin,ymin);

glVertex2i(xmin,ymax);

glVertex2i(xmax,ymax);

glVertex2i(xmax,ymin);

glEnd();

glFlush();

glColor3f(1,0,0);

glLineWidth(1.0);

}

void draw\_pixel(int x,int y)

{

glColor3f(1.0,0.0,0.0);

glPointSize(6.0);

glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();

}

// DEFINE

// 1 -> right edge

// 2 -> bottom edge

// 3 -> left edge

// 4 -> top edge

// xmin,ymin = top left

// xmax,ymax = bottom right

int inside(int x, int y, int clip\_edge){

switch(clip\_edge){

case 1: if(x<xmax) return 1; else return 0;break;

case 2: if(y>ymax) return 1; else return 0;break;

case 3: if(x>xmin) return 1; else return 0;break;

case 4: if(y<ymin) return 1; else return 0;break;

default: return 0;break;

}

}

points\* intersect(points\* S, points\* P, int clip\_edge){

float m; //div by zero error earlier

if((P->getx()-S->getx())==0)

m = 0;

else

m = (float) (P->gety()-S->gety())/(P->getx()-S->getx());

float c = (float) (S->gety()) - (m\*S->getx());

if(clip\_edge==1){int x = xmax; int y = (m\*x)+c;return (new points(x,y));} //bug was because of m=0 thing again

if(clip\_edge==2){int y = ymax; int x; if(m==0) x = P->getx(); else x = (y-c)/m;return (new points(x,y));}

if(clip\_edge==3){int x = xmin; int y = (m\*x)+c;return (new points(x,y));}

if(clip\_edge==4){int y = ymin; int x; if(m==0) x = P->getx(); else x = (y-c)/m;return (new points(x,y));}

}

float\*\* out\_to\_in(float\*\* inner, list<points\*> out){

inner = new float\*[out.size()];

for(int i=0; i<out.size(); i++){

inner[i] = new float[2];

}

sz = out.size();

while(out.size()>0){

points\* temp = out.front();

inner[out.size()-1][0] = temp->getx();

inner[out.size()-1][1] = temp->gety();

out.pop\_front();

}

out.empty();

return inner;

}

//Contains debugging statements to re-create clipping as needed.

float\*\* SHPC(float\*\* inva, list<points\*> out,int clip\_edge){

/\*cout<<"SHPC"<<endl;

for(int i=0; i<sz; i++)

cout<<"\n"<<inva[i][0]<<" "<<inva[i][1];

cout<<"\nxmin - "<<xmin<<" ymin - "<<ymin;

cout<<"\nxmax - "<<xmax<<" ymax - "<<ymax<<endl;\*/

s = new points(inva[sz-1][0],inva[sz-1][1]);

for(int j=0; j<sz; j++){

p = new points(inva[j][0],inva[j][1]);

//cout<<"\n Sx - "<<s->getx()<<" Sy - "<<s->gety();

//cout<<"\n Py - "<<p->getx()<<" Py - "<<p->gety();

if(inside(p->getx(),p->gety(),clip\_edge)) // case 1 & 4

{

//cout<<"\nCASE14"<<endl;

if(inside(s->getx(),s->gety(),clip\_edge)){ // case 1

out.push\_front(new points(p->getx(),p->gety()));

//cout<<"\nCASE1 - "<<p->getx()<<" "<<p->gety()<<endl;

}

else{ // case 4

points\* temp = intersect(s,p,clip\_edge);

//cout<<"\nCASE4 1 - "<<temp->getx()<<" "<<temp->gety()<<endl;

//cout<<"\nCASE4 2 - "<<p->getx()<<" "<<p->gety()<<endl;

out.push\_front(temp);

out.push\_front(p);

}

}

else if(inside(s->getx(),s->gety(),clip\_edge)){ //case 2

points\* temp = intersect(s,p,clip\_edge);

//cout<<"\nCASE2 - "<<temp->getx()<<" "<<temp->gety()<<endl;

out.push\_front(temp);

}

else{

//cout<<"\nCASE3";

}

s = p;

}

inva = out\_to\_in(inva,out);

return inva;

}

void key(unsigned char key\_t, int x, int y){

if((key\_t=='d')||(key\_t=='D'))

{

enter = -1;

glColor3f(0.0,0.0,1.0);

drawPolygon();

in.empty();

}

if((key\_t=='c')||(key\_t=='C'))

{

pts = SHPC(pts,outer,1);

pts = SHPC(pts,outer,2);

pts = SHPC(pts,outer,3);

pts = SHPC(pts,outer,4);

redraw();

}

}

void mouse(int btn, int state, int x, int y){

y = 480-y;

if(btn==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN && enter)

{

points\* temp = new points(x,y);

in.push\_front(temp);

draw\_pixel(x,y);

}

}

void drag\_start(GLint x, GLint y){

y = 480-y;

if(enter==-1&&st\_flag){

xmin = x;

ymin = y;

st\_flag = 0;

}

else{

xmax = x;

ymax = y;

}

redraw();

}

void drag\_end(GLint x, GLint y){

y = 480-y;

if(enter==-1&&st\_flag==0){

xmax = x;

ymax = y;

st\_flag = 1;

redraw();

}

}

void world(){

glPointSize(2);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1,0,0);

}

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

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(640,480);

glutInitWindowPosition(200,200);

glutCreateWindow("Polygon Clipping");

glClear(GL\_COLOR\_BUFFER\_BIT);

glClearColor(1.0,1.0,1.0,0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glutDisplayFunc(world);

glutMouseFunc(mouse);

glutMotionFunc(drag\_start);

glutPassiveMotionFunc(drag\_end);

glutKeyboardFunc(key);

init();

glutMainLoop();

return 0;

}

Output:

Text

Description automatically generated

A picture containing shape

Description automatically generated