Name: Vinayak Madan Shete

Roll No.: TYCOC303

Div: C Batch: C4

Course Name: PBL3- Computer Graphics & Gaming

Course Code: BCE5504

==

Problem Definition:

Write C++/Java program to implement Cohen-Sutherland line clipping algorithm. OR Write C++/Java program to implement Cohen Sutherland Hodgman algorithm to clip any polygon

==

Input:

```
#include <bits/stdc++.h>
#include <graphics.h>
using namespace std;
int xmin, xmax, ymin, ymax;
struct lines {
int x1, y1, x2, y2;
};
int sign(int x) {
return (x > 0) ? 1 : 0;
}
void clip(struct lines mylines) {
int bits[4], byte[4], i, var;
setcolor(RED);
bits[0] = sign(xmin - mylines.x1);
byte[0] = sign(xmin - mylines.x2);
bits[1] = sign(mylines.x1 - xmax);
byte[1] = sign(mylines.x2 - xmax);
```

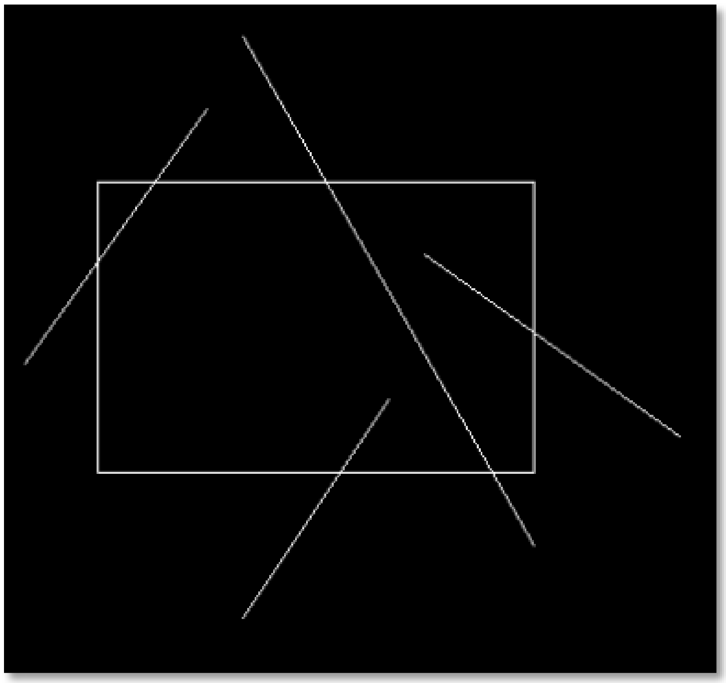
```
bits[2] = sign(ymin - mylines.y1);
byte[2] = sign(ymin - mylines.y2);
bits[3] = sign(mylines.y1 - ymax);
byte[3] = sign(mylines.y2 - ymax);
string initial = "", end = "", temp = "";
for (i = 0; i < 4; i++) {
initial += (bits[i] == 0) ? '0' : '1';
end += (byte[i] == 0) ? '0' : '1';
}
float m = (mylines.y2 - mylines.y1) / (float)(mylines.x2 - mylines.x1);
float c = mylines.y1 - m * mylines.x1;
if (initial == end && end == "0000") {
line(mylines.x1, mylines.y1, mylines.x2, mylines.y2);
return;
} else {
for (i = 0; i < 4; i++) {
Pranav Kulkarni TYCOA288
int val = (bits[i] & byte[i]);
temp += (val == 0) ? '0' : '1';
}
if (temp != "0000")
return;
for (i = 0; i < 4; i++) {
if (bits[i] == byte[i])
continue;
if (i == 0 && bits[i] == 1) {
var = round(m * xmin + c);
mylines.y1 = var;
mylines.x1 = xmin;
}
if (i == 0 && byte[i] == 1) {
var = round(m * xmin + c);
mylines.y2 = var;
mylines.x2 = xmin;
}
if (i == 1 && bits[i] == 1) {
var = round(m * xmax + c);
mylines.y1 = var;
mylines.x1 = xmax;
}
if (i == 1 && byte[i] == 1) {
var = round(m * xmax + c);
mylines.y2 = var;
mylines.x2 = xmax;
}
if (i == 2 && bits[i] == 1) {
var = round((float)(ymin - c) / m);
mylines.y1 = ymin;
mylines.x1 = var;
}
if (i == 2 && byte[i] == 1) {
var = round((float)(ymin - c) / m);
mylines.y2 = ymin;
mylines.x2 = var;
}
if (i == 3 && bits[i] == 1) {
var = round((float)(ymax - c) / m);
Pranav Kulkarni TYCOA288
mylines.y1 = ymax;
Pranav Kulkarni TYCOA288
mylines.x1 = var;
}
if (i == 3 && byte[i] == 1) {
var = round((float)(ymax - c) / m);
mylines.y2 = ymax;
mylines.x2 = var;
}
bits[0] = sign(xmin - mylines.x1);
```

```
byte[0] = sign(xmin - mylines.x2);
bits[1] = sign(mylines.x1 - xmax);
byte[1] = sign(mylines.x2 - xmax);
bits[2] = sign(ymin - mylines.y1);
byte[2] = sign(ymin - mylines.y2);
bits[3] = sign(mylines.y1 - ymax);
byte[3] = sign(mylines.y2 - ymax);
}
initial = "";
end = "";
for (i = 0; i < 4; i++) {
initial += (bits[i] == 0) ? '0' : '1';
end += (byte[i] == 0) ? '0' : '1';
}
if (initial == end && end == "0000") {
line(mylines.x1, mylines.y1, mylines.x2, mylines.y2);
return;
} else
return;
}
}
int main() {
int gd = DETECT, gm;
xmin = 160;
xmax = 400;
ymin = 160;
ymax = 320;
initgraph(&gd, &gm, NULL);
line(xmin, ymin, xmax, ymin);
line(xmax, ymin, xmax, ymax);
line(xmax, ymax, xmin, ymax);
line(xmin, ymax, xmin, ymin);
struct lines mylines[4];
mylines[0].x1 = 120;
```

```
mylines[0].y1 = 260;
mylines[0].x2 = 220;
Pranav Kulkarni TYCOA288
mylines[0].y2 = 120;
mylines[1].x1 = 240;
mylines[1].y1 = 80;
mylines[1].x2 = 400;
mylines[1].y2 = 360;
mylines[2].x1 = 240;
mylines[2].y1 = 400;
mylines[2].x2 = 320;
mylines[2].y2 = 280;
mylines[3].x1 = 340;
mylines[3].y1 = 200;
mylines[3].x2 = 480;
mylines[3].y2 = 300;
for (int i = 0; i < 4; i++) {
line(mylines[i].x1, mylines[i].y1, mylines[i].x2, mylines[i].y2);
delay(1000);
}
for (int i = 0; i < 4; i++) {
clip(mylines[i]);
delay(1000);
}
delay(4000);
getch();
closegraph();
return 0;
}
```

Output:

Before Clipping:



After Clipping: