Experiment No 5

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

An algorithm that clips a polygon must deal with many different cases. The case is particularly note worthy in that the concave polygon is clipped into two separate polygons. All in all, the task of clipping seems rather complex. Each edge of the polygon must be tested against each edge of the clip rectangle; new edges must be added, and existing edges must be discarded, retained, or divided.

The following example illustrate a simple case of polygon clipping

Sutherland and Hodgman's polygon-clipping algorithm uses a divide-and-conquer strategy:

It solves a series of simple and identical problems that, when combined, solve the overall problem. The simple problem is to clip a polygon against a single infinite clip edge. Four clip edges, each defining one boundary of the clip rectangle, successively clip a polygon against a clip rectangle.

Note the difference between this strategy for a polygon and the Cohen-Sutherland algorithm for clipping a line: The polygon clipper clips against four edges in succession, whereas the line clipper tests the outcode to see which edge is crossed, and clips only when necessary.

Steps of Sutherland-Hodgman's polygon-clipping algorithm

• Polygons can be clipped against each edge of the window one at a time.

Windows/edge intersections, if any, are easy to find since the X or Y coordinates are already known.

• Vertices which are kept after clipping against one window edge are saved for clipping against the remaining edges.

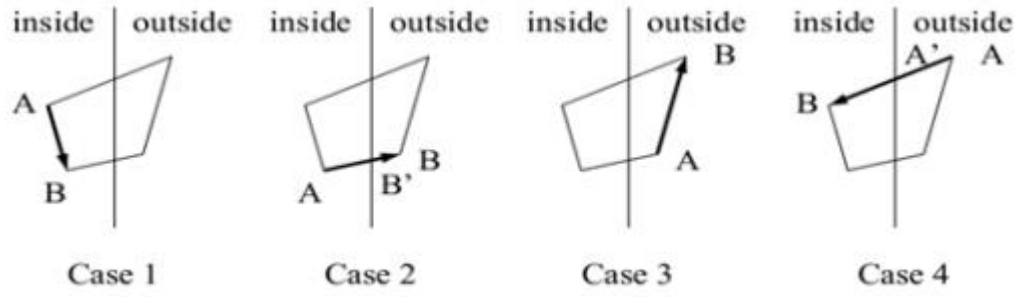
• Note that the number of vertices usually changes and will often increases.

• We are using the Divide and Conquer approach

• After clipped by the right and bottom clip boundaries.

The original polygon and the clip rectangle.

Clipping polygons would seem to be quite complex. A single polygon can actually be split into multiple polygons .The Sutherland-Hodgman algorithm clips a polygon against all edges of the clipping region in turn. The algorithm steps from vertex to vertex, adding 0, 1, or 2 vertices to the output list at each step.



The Sutherland-Hodgeman Polygon-Clipping Algorithms clips a given polygon successively against the edges of the given clip-rectangle. These clip edges are denoted with e1, e2, e3, and e4, here. The closed polygon is represented by a list of its verteces (v1 to vn; Since we got 15 verteces in the example shown above, vn = v15.).

Clipping is computed successively for each edge. The output list of the previous clipping run is used as the inputlist for the next clipping run. 1st run: Clip edge: e1; inputlist = {v1, v2, ..., v14, v15}, the given polygon

```
#include <iostream>
#include <math.h>
#include <time.h>
#include <GL/glut.h>
using namespace std;
int wxmin = 200,wxmax=500,wymax=350, wymin=100;
int points[10][2];
int edge;
void init(){
 glClearColor(1.0,1.0,1.0,0.0);
 glMatrixMode(GL_PROJECTION);
 gluOrtho2D(0,640,0,480);
 glClear(GL_COLOR_BUFFER_BIT);
}
void Draw(){
 glClearColor(1.0,1.0,1.0,0.0);
 glClear(GL_COLOR_BUFFER_BIT);
```

```
 glColor3f(0.2,0.2,1);
 glBegin(GL_POLYGON);
 for(int i=0; i<edge; i++)
 {
 glVertex2i(points[i][0],points[i][1]);
 }
 glEnd();
 glFlush();

 glColor3f(0,1,0);
 glBegin(GL_LINE_LOOP);
 glVertex2i(200,100);
 glVertex2i(500,100);
 glVertex2i(500,350);
 glVertex2i(200,350);
 glEnd();
 glFlush();
}
int BottomCliping(int e){
float m=0;
int x=0,k=0;
int t[10][2];
 for(int i=0; i<e; i++){
 if(points[i][1] < wymin){

 if(points[i+1][1] < wymin){

 }
 else if(points[i+1][1] > wymin){
 float x1,x2;
 float y1,y2;
```

```
x1 = points[i][0];
y1 = points[i][1];
x2 = points[i+1][0];
y2 = points[i+1][1];
x = ((1/((y2-y1)/(x2-x1))) * (wymin - y1) )+ x1;
t[k][0] = x;
t[k][1] = wymin;
k++;

}

}
else if(points[i][1]>wymin){

if(points[i+1][1] > wymin){
t[k][0] = points[i][0];
t[k][1] = points[i][1];
k++;
}
else if(points[i+1][1] < wymin){
float x1,x2;
float y1,y2;
x1 = points[i][0];
y1 = points[i][1];
x2 = points[i+1][0];
y2 = points[i+1][1];

x = ((1/((y2-y1)/(x2-x1))) * (wymin - y1) )+ x1;

t[k][0] = x1;
t[k][1] = y1;
```

```
 k++;
 t[k][0] = x;
 t[k][1] = wymin;
 k++;

 }

 }

 }
 cout<<"k = "<<k;
 for(int i=0; i<10;i++)
 {
 points[i][0] = 0;
 points[i][1] = 0;

 }

 for(int i=0; i<k;i++)
 {
 cout<<"\n"<<t[i][0]<<" "<<t[i][1];
 points[i][0] = t[i][0];
 points[i][1] = t[i][1];

 }
 points[k][0] = points[0][0];
 points[k][1] = points[0][1];
 return k;
}
int TopCliping(int e){
float m=0;
```

```
int x=0,k=0;
int t[10][2];
 for(int i=0; i<e; i++){
 if(points[i][1] > wymax){

 if(points[i+1][1] > wymax){

 }
 else if(points[i+1][1] < wymax){
 float x1,x2;
 float y1,y2;
 x1 = points[i][0];
 y1 = points[i][1];
 x2 = points[i+1][0];
 y2 = points[i+1][1];
 x = ((1/((y2-y1)/(x2-x1))) * (wymax - y1) )+ x1;
 t[k][0] = x;
 t[k][1] = wymax;
 k++;

 }

 }
 else if(points[i][1]<wymax){

 if(points[i+1][1] < wymax){
 t[k][0] = points[i][0];
 t[k][1] = points[i][1];
 k++;
 }
 else if(points[i+1][1] > wymax){
```

```
float x1,x2;
float y1,y2;
x1 = points[i][0];
y1 = points[i][1];
x2 = points[i+1][0];
y2 = points[i+1][1];

x = ((1/((y2-y1)/(x2-x1))) * (wymax - y1) )+ x1;

t[k][0] = x1;
t[k][1] = y1;
k++;
t[k][0] = x;
t[k][1] = wymax;
k++;



}

}

}
cout<<"k = "<<k;
for(int i=0; i<10;i++)
{
points[i][0] = 0;
points[i][1] = 0;

}

for(int i=0; i<k;i++)
```

```
{
 cout<<"\n"<<t[i][0]<<" "<<t[i][1];
 points[i][0] = t[i][0];
 points[i][1] = t[i][1];

 }
 points[k][0] = points[0][0];
 points[k][1] = points[0][1];
 return k;
}
int leftCliping(int e){
float m=0;
int y=0, k = 0;
int t[10][2];
 for(int i=0;i<e;i++)
 {

 if(points[i][0] < wxmin){

 if(points[i+1][0] < wxmin){
 cout<<"\n Test 1";

 }
 else if (points[i+1][0] > wxmin){
 cout<<"\n Test 2";
 float x1,x2;
 float y1,y2;
 x1 = points[i][0];
 y1 = points[i][1];
 x2 = points[i+1][0];
 y2 = points[i+1][1];
```

```
y = (((y2-y1)/(x2-x1)) * (wxmin - x1) )+ y1;
t[k][0] = wxmin;
t[k][1] = y;
k++;
}
}
else if(points[i][0] > wxmin){

if(points[i+1][0] > wxmin){

t[k][0] = points[i][0];
t[k][1] = points[i][1];
k++;
}
else if(points[i+1][0] < wxmin){



float x1,x2;
float y1,y2;
x1 = points[i][0];
y1 = points[i][1];
x2 = points[i+1][0];
y2 = points[i+1][1];

y = ((y2-y1)/(x2-x1)*(wxmin - x1)) + y1;

t[k][0] = x1;
t[k][1] = y1;
k++;
t[k][0] = wxmin;
t[k][1] = y;
```

```
k++;
}


}
}
cout<<"k = "<<k;
for(int i=0; i<10;i++)
{
points[i][0] = 0;
points[i][1] = 0;

}

for(int i=0; i<k;i++)
{
cout<<"\n"<<t[i][0]<<" "<<t[i][1];
points[i][0] = t[i][0];
points[i][1] = t[i][1];

}
points[k][0] = points[0][0];
points[k][1] = points[0][1];
return k;
}
int RightCliping(int e){
float m=0;
int y=0, k = 0;
int t[10][2];
for(int i=0;i<e;i++)
{
```

```
if(points[i][0] > wxmax){

if(points[i+1][0] > wxmax){



}
else if(points[i+1][0] < wxmax){

float x1,x2;
float y1,y2;
x1 = points[i][0];
y1 = points[i][1];
x2 = points[i+1][0];
y2 = points[i+1][1];
y = (((y2-y1)/(x2-x1)) * (wxmax - x1) )+ y1;
t[k][0] = wxmax;
t[k][1] = y;
k++;
}

}
else if(points[i][0] < wxmax){

if(points[i+1][0] < wxmax){

t[k][0] = points[i][0];
t[k][1] = points[i][1];
k++;
}
else if(points[i+1][0] > wxmax){
```

```
float x1,x2;
float y1,y2;
x1 = points[i][0];
y1 = points[i][1];
x2 = points[i+1][0];
y2 = points[i+1][1];

y = ((y2-y1)/(x2-x1)*(wxmax - x1)) + y1;
t[k][0] = x1;
t[k][1] = y1;
k++;
t[k][0] = wxmax;
t[k][1] = y;
k++;
}
}
}
cout<<"k = "<<k;
for(int i=0; i<10;i++)
{
points[i][0] = 0;
points[i][1] = 0;

}
for(int i=0; i<k;i++)
{
cout<<"\n"<<t[i][0]<<" "<<t[i][1];
points[i][0] = t[i][0];
points[i][1] = t[i][1];
```

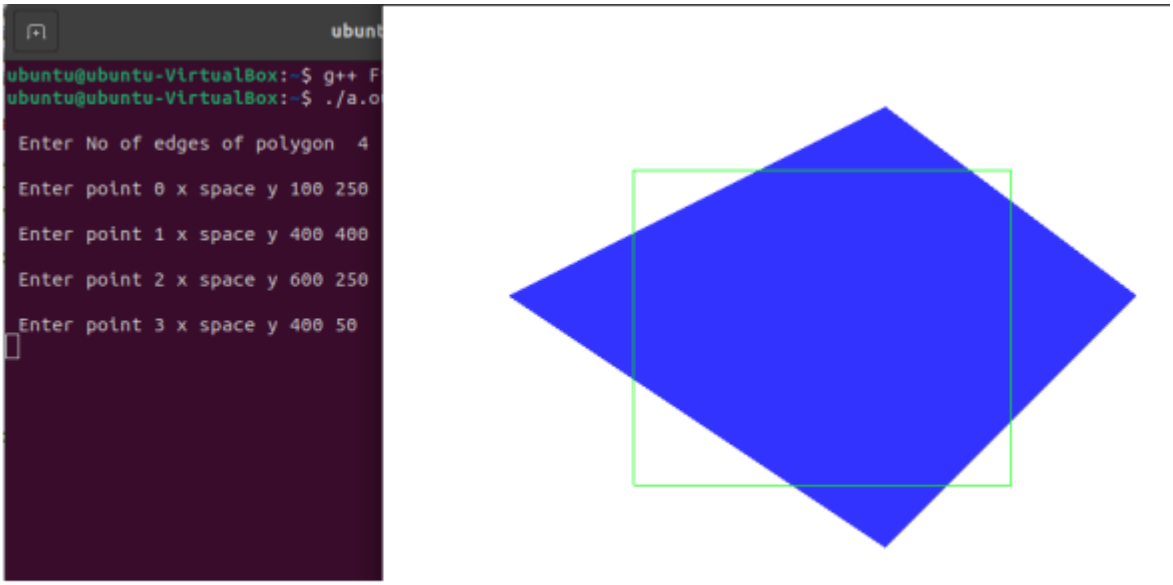
```
 }
 points[k][0] = points[0][0];
 points[k][1] = points[0][1];
 return k;
}
void P_C(){
 Draw();
}
void goMenu(int value){
 switch(value){

 case 1:
 edge = leftCliping(edge);
 Draw();
 break;
 case 2:
 edge = RightCliping(edge);
 Draw();
 break;
 case 3:
 edge = TopCliping(edge);
 Draw();
 break;
 case 4:
 edge = BottomCliping(edge);
 Draw();
 break;

 }
 glutPostRedisplay();
}
```

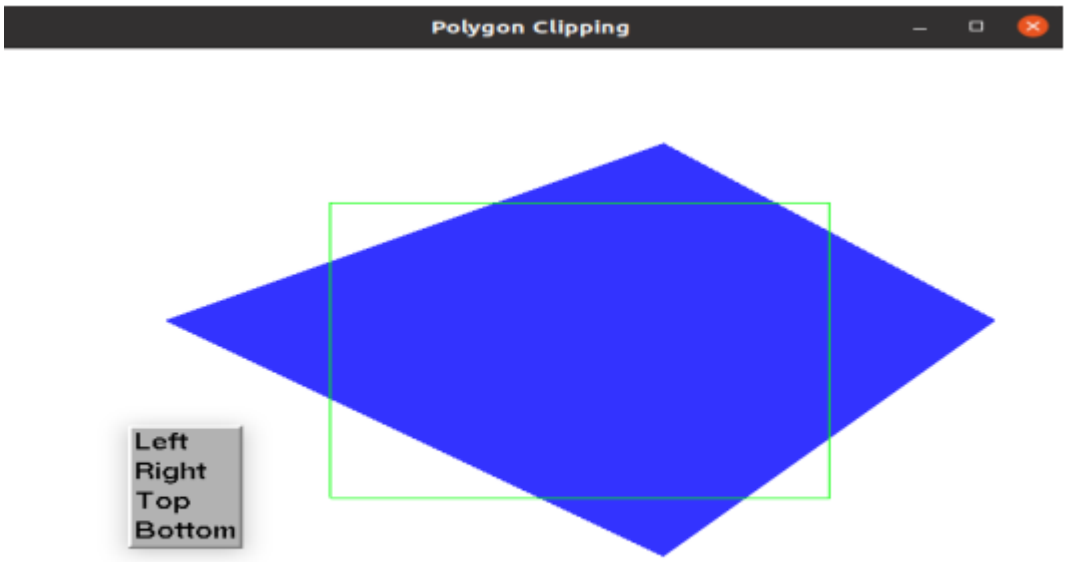
```
int main(int argc, char** argv){
 cout<<"\n Enter No of edges of polygon ";
 cin>>edge;

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

 cout<<"\n Enter point "<<i<<" x space y ";
 cin>>points[i][0]>>points[i][1];

 }
 points[edge][0] = points[0][0];
 points[edge][1] = points[0][1];
 glutInit(&argc, argv);
 glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
 glutInitWindowSize(640,480);
 glutInitWindowPosition(200,200);
 glutCreateWindow("Polygon Clipping");
 init();

 glutCreateMenu(goMenu);
 glutAddMenuEntry("Left",1);
 glutAddMenuEntry("Right",2);
 glutAddMenuEntry("Top",3);
 glutAddMenuEntry("Bottom",4);
 glutAttachMenu(GLUT_RIGHT_BUTTON);

 glutDisplayFunc(P_C);

 glutMainLoop();
 return 0;
}
```

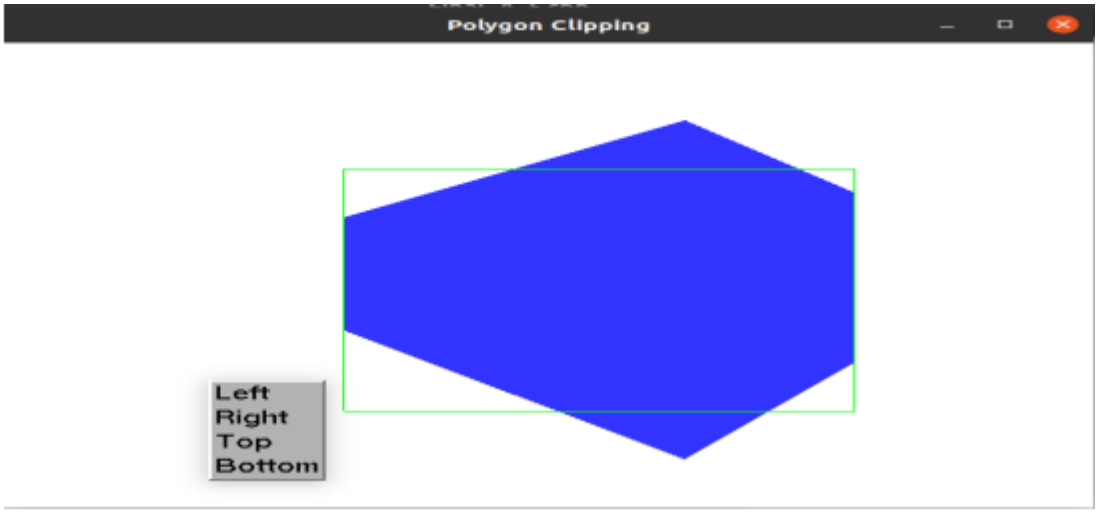
Out put

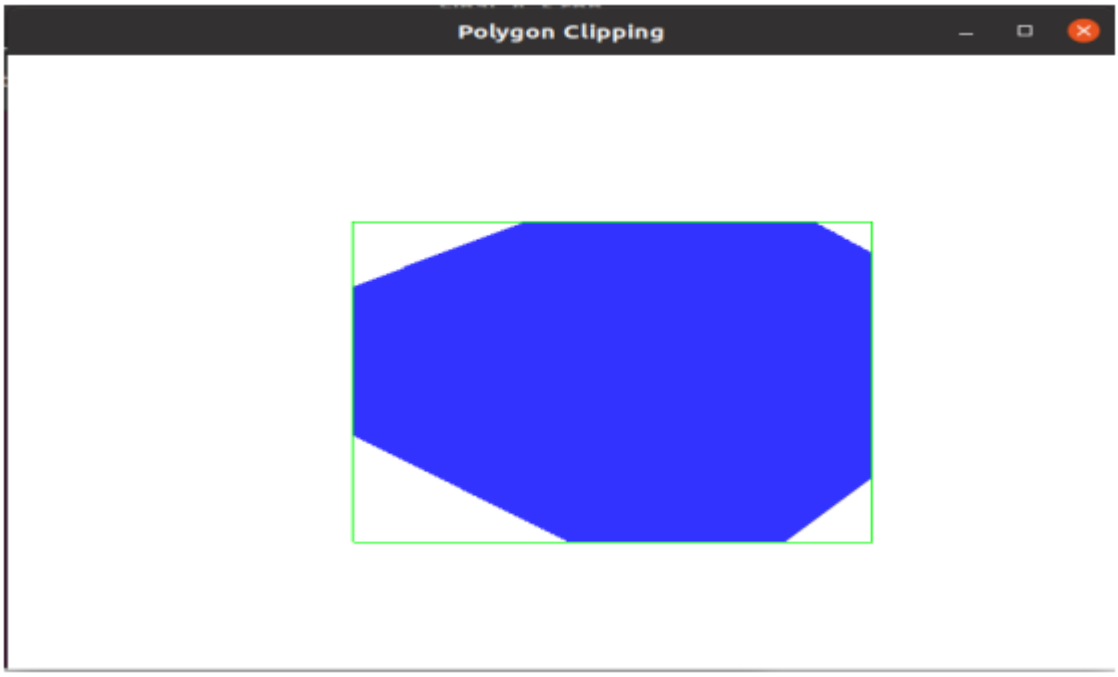
g++ filename.cpp -lGL -lGLU -lglut

./a.out






Polygon Clipping
Left
Right
Top
Bottom


Polygon Clipping