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

```
#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;
```

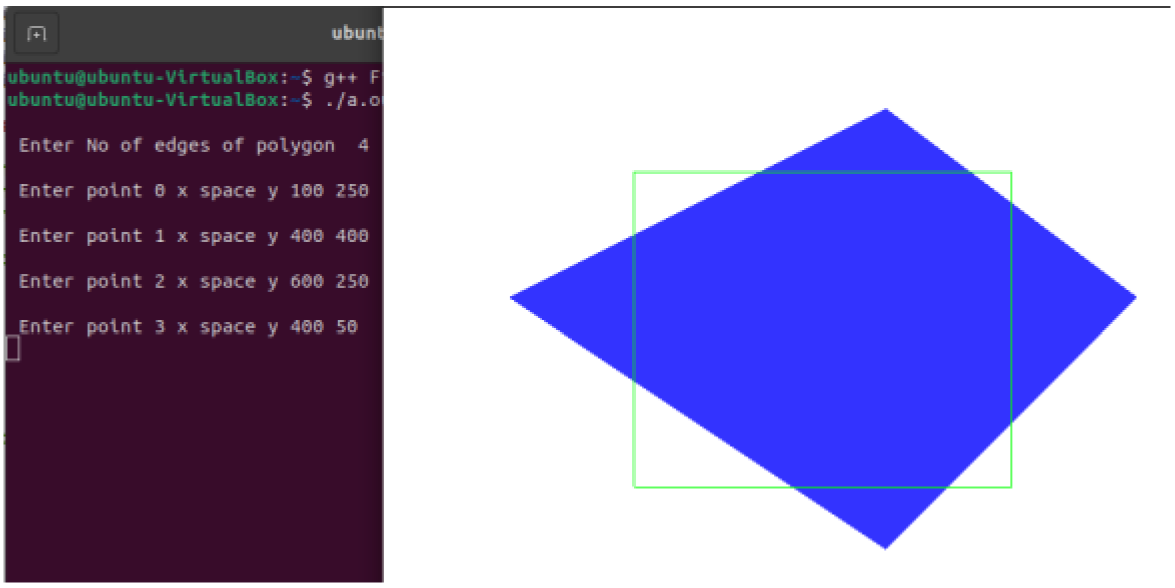
```
 points[i][1] = 0;


 }


 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){
```

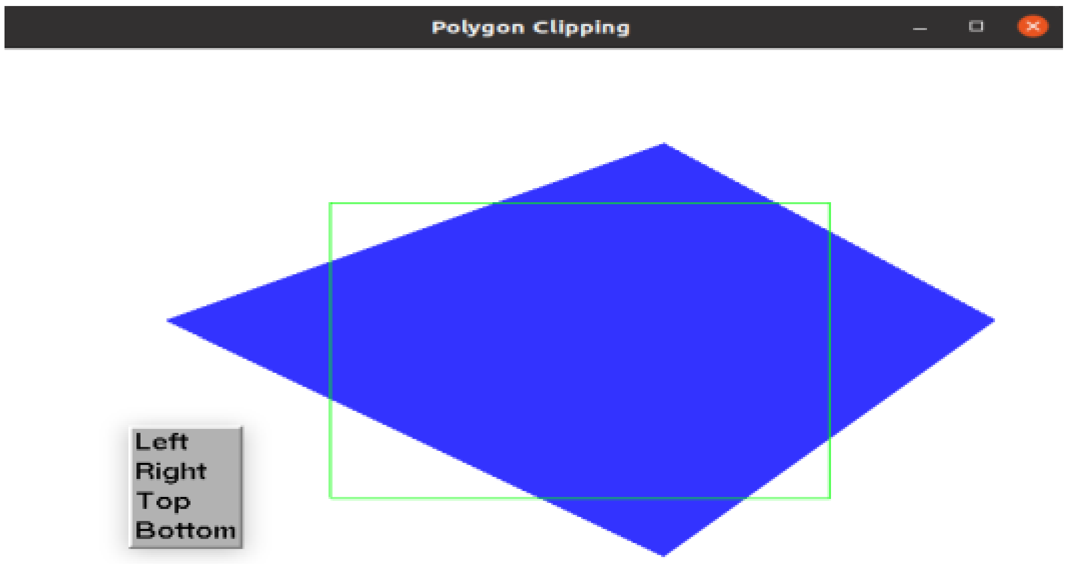
```
 case 1:
 edge = leftCliping(edge);
 Draw();
 break;
 case 2:
 edge = RightCliping(edge);
 Draw();
 break;
 case 3:
 edge = TopCliping(edge);
 Draw();
 break;
 case 4:
 edge = BottomCliping(edge);
 Draw();
 break;

 }
 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;
}
```

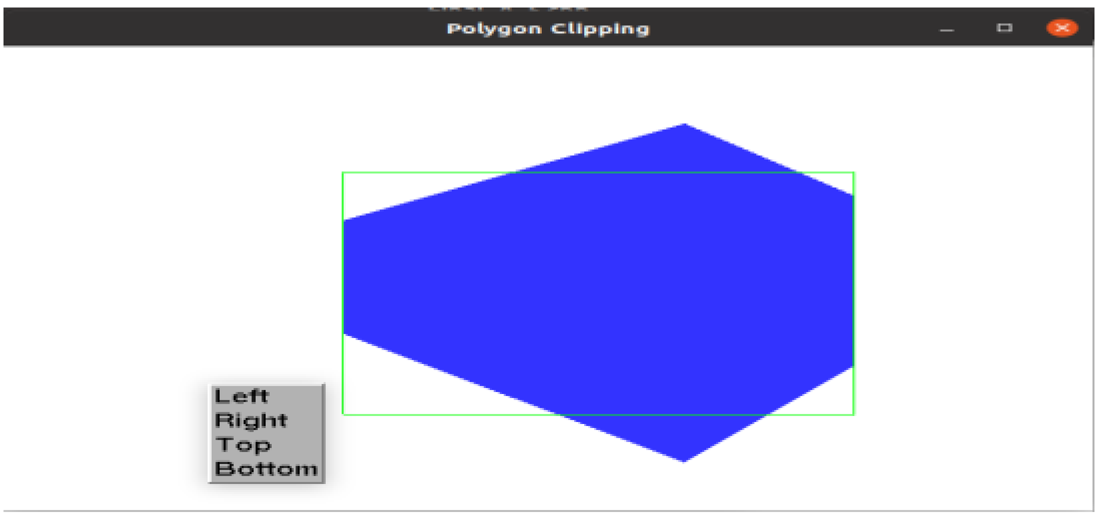
Out put

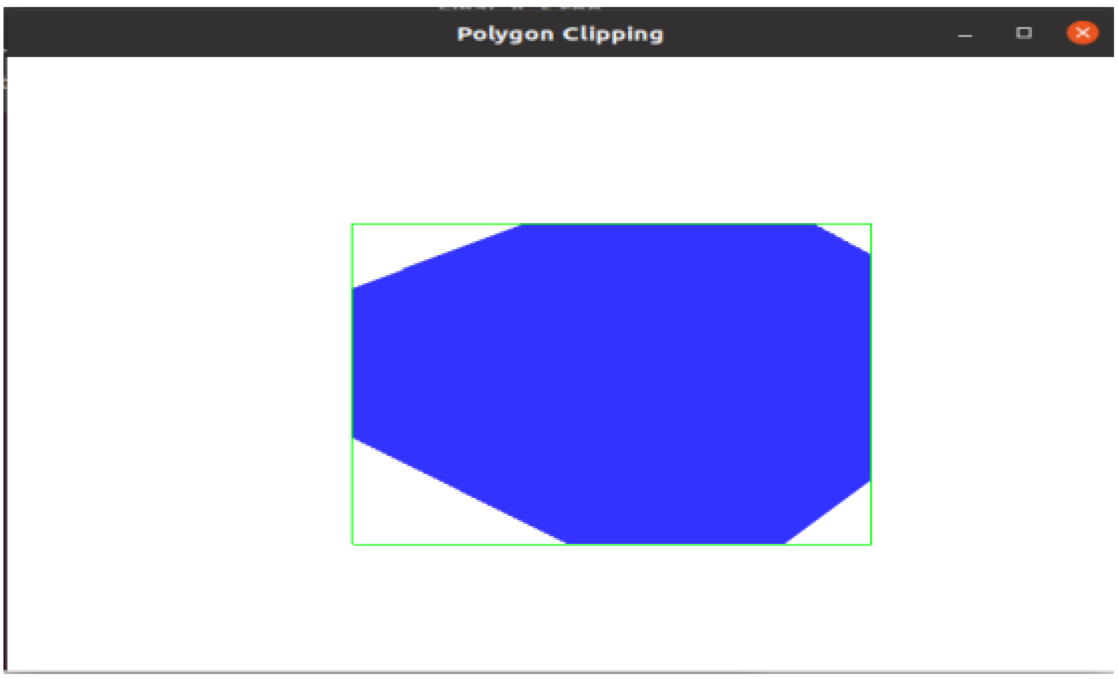
```
g++ filename.cpp -lGL -lGLU -lglut
./a.out
```



```
ubuntu@ubuntu-VirtualBox:~$ g++ F
ubuntu@ubuntu-VirtualBox:~$ ./a.o

 Enter No of edges of polygon  4

 Enter point 0 x space y 100 250

 Enter point 1 x space y 400 400

 Enter point 2 x space y 600 250

 Enter point 3 x space y 400 50
```




Polygon Clipping
Left
Right
Top
Bottom


Polygon Clipping