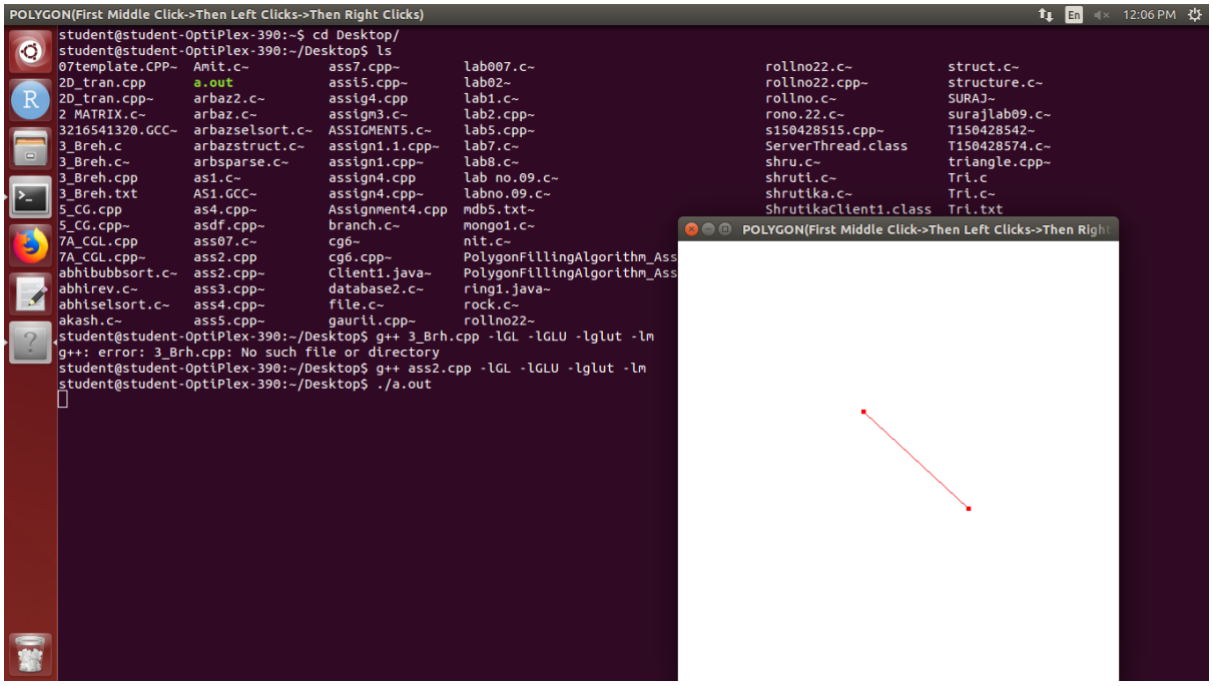
Opengl program to draw a solid line using DDA algorithm

Assignment 2

```
#include<iostream>
#include<GL/glut.h>
using namespace std;
int Algo,type;
void Init()
{
   glClearColor(0,0,0,0);
   glColor3f(1,0,0);
   gluOrtho2D(0,640,0,480);
   glClear(GL_COLOR_BUFFER_BIT);
}sss
int sign(float a){
   if(a==0){
      return 0;
   }
   if(a>0){
      return 1;
   }
   return -1;
}
void B_Line(int x_1,int y_1,int x_2,int y_2,int t){

   float dy, dx, m , P;
   dy = y_2 - y_1;
   dx = x_2 - x_1;
   m = dy/dx;
   P = 2*dy - dx;
   int x = x_1, y = y_1;
   cout<<"\n x1 = "<<x<<" y1 = "<<y;
```

```
if(m<1){
   int cnt=1;
   for(int i=0; i<=dx;i++){
      if(t == 1){
         glBegin(GL_POINTS);
            glVertex2i(x,y);
         glEnd();
      }
      if(t == 2){
         if(i%2==0){
            glBegin(GL_POINTS);
               glVertex2i(x,y);
            glEnd();
         }
      }
      if(t == 3){
         if(cnt <= 20){
            glBegin(GL_POINTS);
               glVertex2i(x,y);
            glEnd();
         }
         cnt++;
         if(cnt == 25){
            cnt =1;
         }
```


POLYGON(First Middle Click->Then Left Clicks->Then Right Clicks)
12:06 PM
student@student-OptiPlex-390:~$ cd Desktop/
student@student-OptiPlex-390:~/Desktop$ ls
07template.CPP~ Amit.c~ ass7.cpp~ lab007.c~ rollno22.c~ struct.c~
2D_tran.cpp a.out assi5.cpp~ lab02~ rollno22.cpp~ structure.c~
2D_tran.cpp~ arbaz2.c~ assig4.cpp lab1.c~ rollno.c~ SURAJ~
2 MATRIX.c~ arbaz.c~ assigm3.c~ lab2.cpp~ rono.22.c~ surajlab09.c~
3216541320.GCC~ arbazselsort.c~ ASSIGMENT5.c~ lab5.cpp~ s150428515.cpp~ T150428542~
3_Breh.c arbazstruct.c~ assign1.1.cpp~ lab7.c~ ServerThread.class T150428574.c~
3_Breh.c~ arbsparse.c~ assign1.cpp~ lab8.c~ shru.c~ triangle.cpp~
3_Breh.cpp as1.c~ assign4.cpp lab no.09.c~ shruti.c~ Tri.c
3_Breh.txt AS1.GCC~ assign4.cpp~ labno.09.c~ shrutika.c~ Tri.c~
5_CG.cpp as4.cpp~ Assignment4.cpp mdb5.txt~ ShrutikaClient1.class Tri.txt
5_CG.cpp~ asdf.cpp~ branch.c~ mongo1.c~
7A_CGL.cpp ass07.c~ cg6~ nit.c~
7A_CGL.cpp~ ass2.cpp cg6.cpp~ PolygonFillingAlgorithm_Ass
abhibubbsort.c~ ass2.cpp~ Client1.java~ PolygonFillingAlgorithm_Ass
abhirev.c~ ass3.cpp~ database2.c~ ring1.java~
abhiselsort.c~ ass4.cpp~ file.c~ rock.c~
akash.c~ ass5.cpp~ gaurii.cpp~ rollno22~
student@student-OptiPlex-390:~/Desktop$ g++ 3_Brh.cpp -lGL -lGLU -lglut -lm
g++: error: 3_Brh.cpp: No such file or directory
student@student-OptiPlex-390:~/Desktop$ g++ ass2.cpp -lGL -lGLU -lglut -lm
student@student-OptiPlex-390:~/Desktop$ ./a.out
POLYGON(First Middle Click->Then Left Clicks->Then Righ

Opengl program to draw a circle using bresenham's circle drawing algorithm

ASSIGNMENT 3

Implement Bresenham circle drawing algorithm to draw any object. The object should be displayed in all the quadrants with respect to center and radius

Code

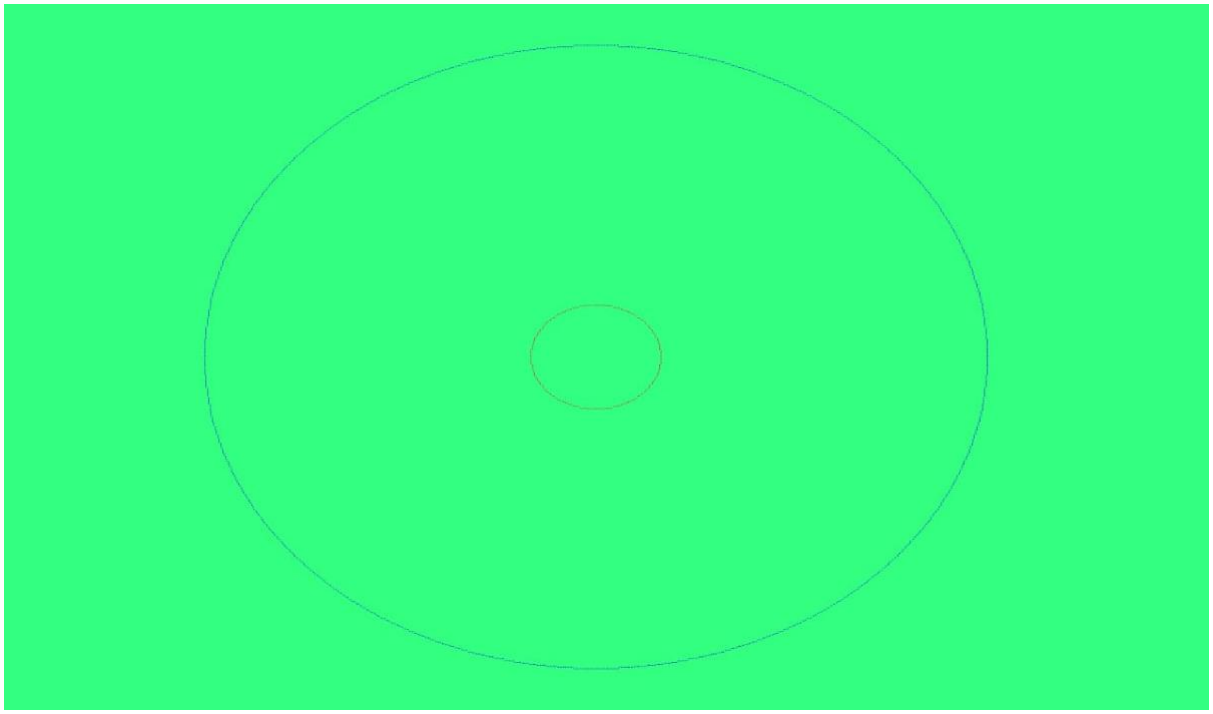
```
#include<GL/glut.h>
#include<iostream>
using namespace std;
int r;
void E_way(int x, int y){
      glBegin(GL_POINTS);
         glVertex2i(x+320,y+240);
         glVertex2i(y+320,x+240);
         glVertex2i(y+320, -x+240);
         glVertex2i(x+320, -y+240);
         glVertex2i(-x+320,-y+240);
         glVertex2i(-y+320,-x+240);
         glVertex2i(-y+320,x+240);
         glVertex2i(-x+320,y+240);
      glEnd();
      glFlush();
}
void B_circle(){
   float d;
   d = 3 - 2*r;

   int x,y;
   x = 0 ;
   y = r ;
```

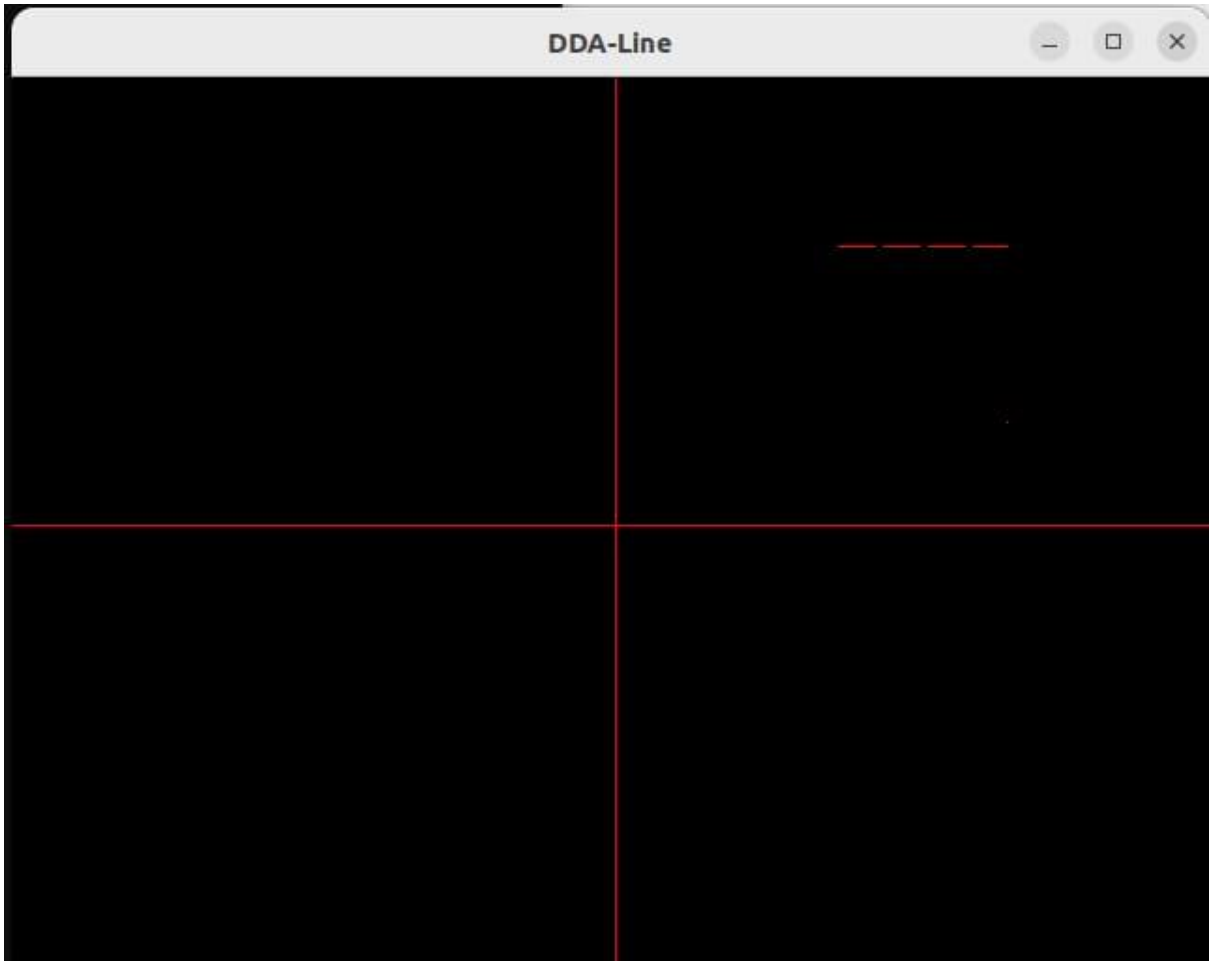
```
    do{
        E_way(x,y);
        if(d<0){
            d=d+4*x+6;
        }
        else{
            d= d+4*(x-y)+10;
            y=y-1;
        }
        x=x+1;
    }
while(x<y);
}
void init(){
    glClearColor(1,1,1,0);
    glColor3f(1,0,0);
    gluOrtho2D(0,640,0,480);
    glClear(GL_COLOR_BUFFER_BIT);
}

int main(int argc, char **argv){
    cout<<"\n Enter Radius \t ";
    cin>>r;
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowPosition(100,100);
    glutInitWindowSize(640,480);
    glutCreateWindow("Circle");
    init();
    glutDisplayFunc(B_circle);
    glutMainLoop();
```

```
    return 0;
}
```

OUTPUT

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





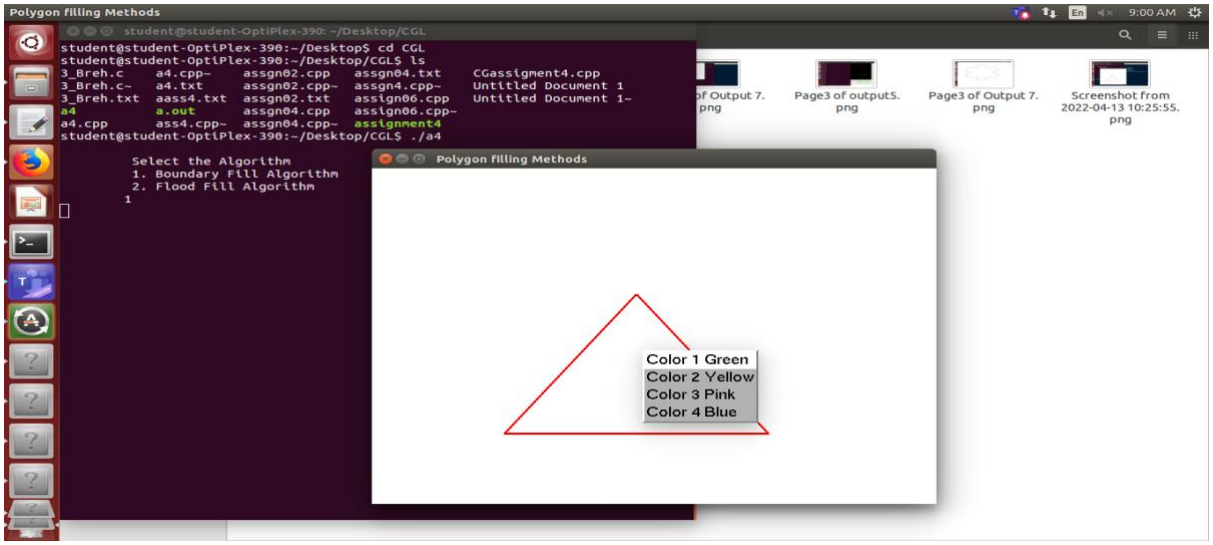
Write a program to create menu and attach it with right mouse button

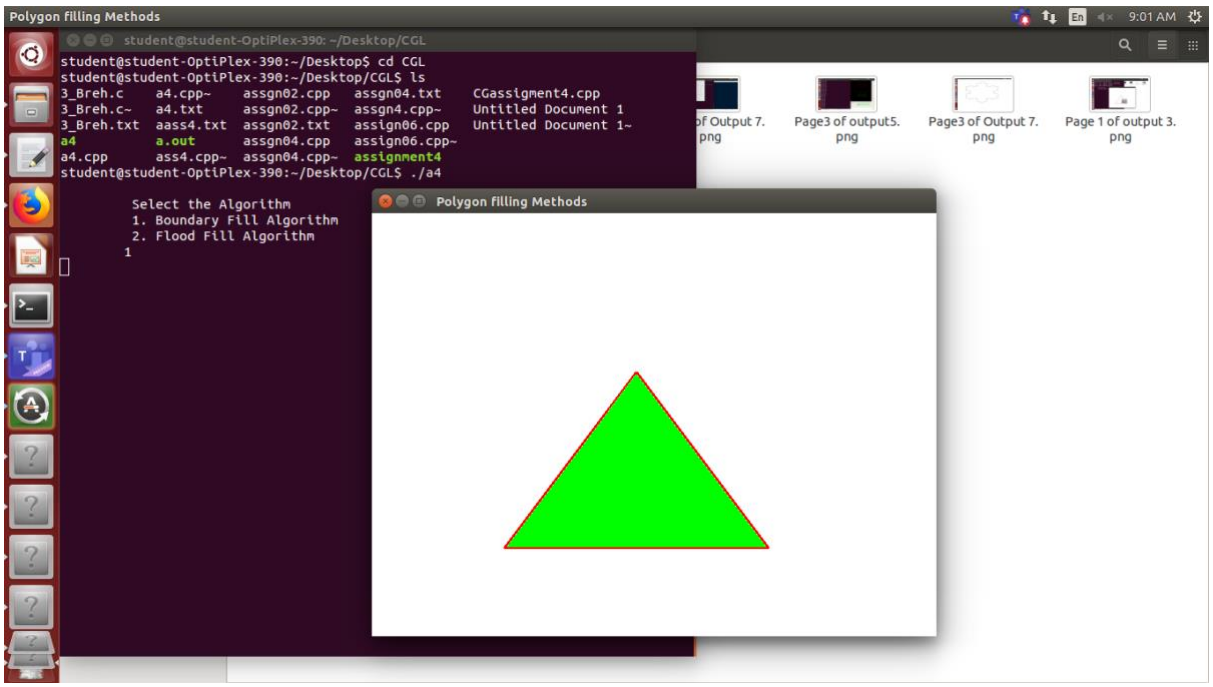
ASSIGNMENT 4

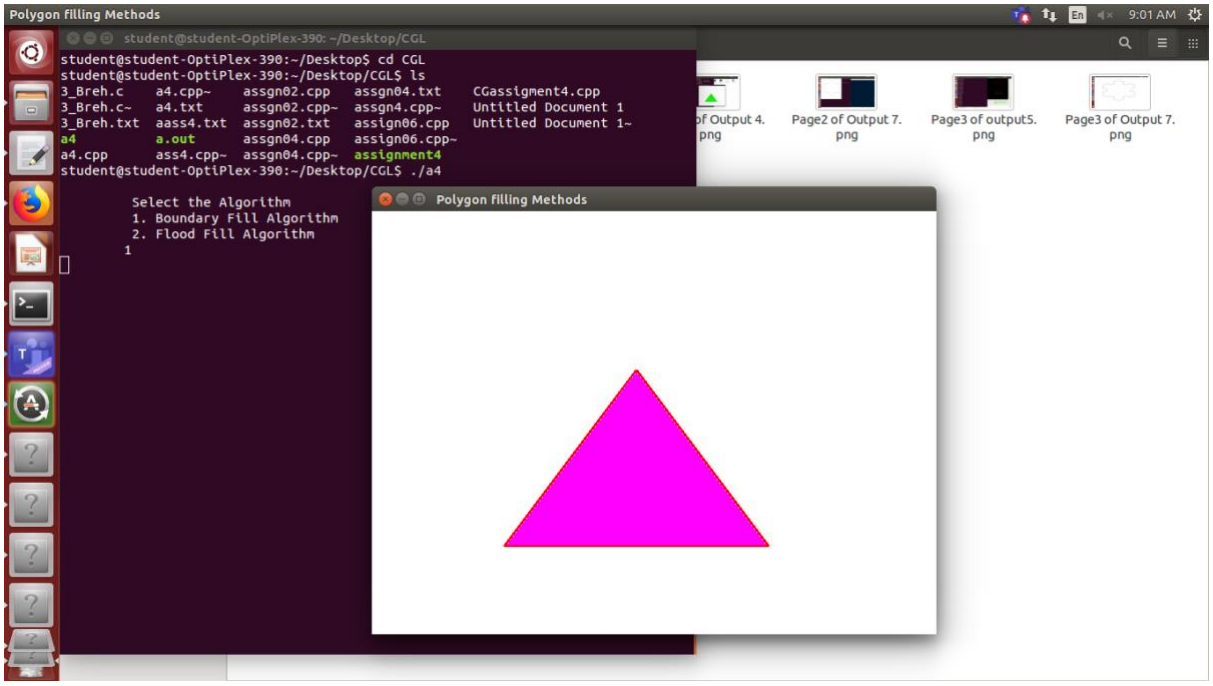
```
#include <iostream>
#include <math.h>
#include <time.h>
#include <GL/glut.h>
using namespace std;
float R=0,G=0,B=0;
int Algo;
void delay(float ms){
   clock_t goal = ms + clock();
   while(goal>clock());
```

```
}
void init(){
   glClearColor(1.0,1.0,1.0,0.0);
   glMatrixMode(GL_PROJECTION);
   gluOrtho2D(0,640,0,480);
}
void boundaryFill(int x, int y, float* fillColor, float* bc){
   float color[3];
   glReadPixels(x,y,1.0,1.0,GL_RGB,GL_FLOAT,color);
   if((color[0]!=bc[0] || color[1]!=bc[1] || color[2]!=bc[2])&&(
    color[0]!=fillColor[0] || color[1]!=fillColor[1] || color[2]!=fillColor[2])){
      glColor3f(fillColor[0],fillColor[1],fillColor[2]);
      glBegin(GL_POINTS);
         glVertex2i(x,y);
      glEnd();
      glFlush();
      boundaryFill(x+1,y,fillColor,bc);
      boundaryFill(x-2,y,fillColor,bc);
      boundaryFill(x,y+2,fillColor,bc);
      boundaryFill(x,y-2,fillColor,bc);
   }
}
void floodFill(int x, int y, float *newCol, float *oldcol)
{
float pixel[3];
glReadPixels(x,y,1,1,GL_RGB,GL_FLOAT,pixel);
if(oldcol[0]==pixel[0] && oldcol[1]==pixel[1] && oldcol[2]==pixel[2]){
 glBegin(GL_POINTS);
 glColor3f(newCol[0],newCol[1],newCol[2]);
 glVertex2i(x,y);
 glEnd();
```

```
 glFlush();
 floodFill(x,y+1,newCol,oldcol);
 floodFill(x+1,y,newCol,oldcol);
 floodFill(x,y-1,newCol,oldcol);
 floodFill(x-1,y,newCol,oldcol);
}
}
void mouse(int btn, int state, int x, int y){
   y = 480-y;
   if(btn==GLUT_LEFT_BUTTON)
   {
      if(state==GLUT_DOWN)
      {
         float bcol[] = {1,0,0};
         float newCol[] = {R,G,B};
        float oldcol[] = {1,1,1};
           if(Algo==1)
       {
                    boundaryFill(x,y,newCol,bcol);
            }
           if(Algo==2)
        {
              floodFill(x,y,newCol,oldcol);
        }
      }
   }
}
```


Polygon filling Methods
9:00 AM
student@student-OptiPlex-390: ~/Desktop/CGL
student@student-OptiPlex-390:~/Desktop$ cd CGL
student@student-OptiPlex-390:~/Desktop/CGL$ ls
3_Breh.c a4.cpp~ assgn02.cpp assgn04.txt CGassigment4.cpp
3_Breh.c~ a4.txt assgn02.cpp~ assgn4.cpp~ Untitled Document 1
3_Breh.txt aass4.txt assgn02.txt assign06.cpp Untitled Document 1~
a4 a.out assgn04.cpp assign06.cpp~
a4.cpp ass4.cpp~ assgn04.cpp~ assignment4
student@student-OptiPlex-390:~/Desktop/CGL$ ./a4
Select the Algorithm
1. Boundary Fill Algorithm
2. Flood Fill Algorithm
1
Polygon filling Methods
Color 1 Green
Color 2 Yellow
Color 3 Pink
Color 4 Blue
Page3 of output5. png
Page3 of Output 7. png
Screenshot from 2022-04-13 10:25:55. png


Polygon filling Methods
9:01 AM
student@student-OptiPlex-390: ~/Desktop/CGL
student@student-OptiPlex-390:~/Desktop$ cd CGL
student@student-OptiPlex-390:~/Desktop/CGL$ ls
3_Breh.c a4.cpp~ assgn02.cpp assgn04.txt CGassigment4.cpp
3_Breh.c~ a4.txt assgn02.cpp~ assgn4.cpp~ Untitled Document 1
3_Breh.txt aass4.txt assgn02.txt assign06.cpp Untitled Document 1~
a4 a.out assgn04.cpp assign06.cpp~
a4.cpp ass4.cpp~ assgn04.cpp~ assignment4
student@student-OptiPlex-390:~/Desktop/CGL$ ./a4
Select the Algorithm
1. Boundary Fill Algorithm
2. Flood Fill Algorithm
1
Polygon filling Methods
Page3 of output5. png
Page3 of Output 7. png
Page 1 of output 3. png



ASSIGNMENT 5

```
#include<iostream>
#include<GL/glut.h>
#include<math.h>
using namespace std;
#define w 500
#define h 500
#define max 50
int win[4][2],k=0,flag=0,m=0,n=0,flag1=0,wino[4][2],flag2=0;
int xmin,ymin,xmax,ymax;
class edge
{
```

```
    public:
    int xc,yc;
};
 edge e[max];   //maximum edges for polygon
 edge out[max]; // store no of vertices changed after clipping boundary
void setpixel(GLint x, GLint y) // Used to draw x-axis
{
        glColor3f(1.0,0.0,0.0);
        glPointSize(2.0);
        glBegin (GL_POINTS);
                glVertex2f(x,y);
        glEnd();
        glFlush();
}
class clipping
{
    public:
     int left_below(int val,int min )
     {
        if(val>min) //val is entered vertex compared  with left edge of window
        return 1;
        else return 0;
     }
    int right_above(int val,int min )
     {
        if(val>=min)
        return 0;
        else return 1;
     }
    void suth_hodge()
    {
```

```
int k=0,j,i;
float s;
cout<<"\nxmin ="<<xmin;
cout<<"\nymin ="<<ymin;
cout<<"\nxmax ="<<xmax;
cout<<"\nymax ="<<ymax;
for(int index=0;index<4;index++)
{
    if(index==0) //left clipping
    {
        for(j=0;j<m;j++)
        {
            if(!(left_below(e[j].xc,xmin))&&(left_below(e[j+1].xc,xmin))) //outside to inside
            {
                cout<<"\nj:  "<<j;
            //    cout<<"\n e[j+1].yc ="<<e[j+1].yc<<" e[j].yc ="<< e[j].yc;
                s=(float)(e[j+1].yc-e[j].yc)/(e[j+1].xc-e[j].xc);
              //
                out[k].yc =e[j].yc+(int)(s*(xmin-e[j].xc));
                out[k].xc=xmin;
                k++;
                out[k].yc =e[j+1].yc;
                out[k].xc =e[j+1].xc;
                k++;
            }
            else if(!(left_below(e[j].xc,xmin))&&!(left_below(e[j+1].xc,xmin)))//both points are
outside
            {cout<<"\nj:  "<<j;
            }
            else if((left_below(e[j].xc,xmin))&&!(left_below(e[j+1].xc,xmin)))//inside to outside
            {
```

```
                //  cout<<"\n e[j+1].yc ="<<e[j+1].yc<<" e[j].yc ="<< e[j].yc;
                  s=(float)(e[j+1].yc-e[j].yc)/(e[j+1].xc-e[j].xc);
                  out[k].yc =e[j].yc+(int)(s*(xmin-e[j].xc));
                  out[k].xc=xmin; cout<<"\nj:  "<<j;
                  k++;
                }
                else if((left_below(e[j].xc,xmin))&&(left_below(e[j+1].xc,xmin))) //inside to inside
                {cout<<"\nj:  "<<j;
                  out[k].yc =e[j+1].yc;
                  out[k].xc =e[j+1].xc;
                  k++;
                }

              }
              out[k].yc=out[0].yc;
              out[k].xc=out[0].xc;
              k++;
              m=k;
           for(i=0;i<m;i++)
                {
                        cout<<"\nX: "<<out[i].xc<<" Y: "<<out[i].yc;
                }
          }
        if(index==1)
          {
              k=0;
              for(j=0;j<m-1;j++)
              {
                  if(!(left_below(out[j].yc,ymin))&&(left_below(out[j+1].yc,ymin))) // bottom
clipping
                  {
```

```
        cout<<"\n o[j+1].yc ="<<out[j+1].yc<<" o[j].yc ="<< out[j].yc;
        s=(float)(out[j+1].yc-out[j].yc)/(out[j+1].xc-out[j].xc);
       cout<<"\nslope is :"<<s;
        e[k].xc =out[j].xc+(int)((ymin-out[j].yc)/s);
        e[k].yc=ymin;
        k++;
        e[k].yc=out[j+1].yc;
        e[k].xc=out[j+1].xc ;
        k++;
    }
    else if(!(left_below(out[j].yc,ymin))&&!(left_below(out[j+1].yc,ymin)))
    {
    }
    else if((left_below(out[j].yc,ymin))&&!(left_below(out[j+1].yc,ymin)))
    {
        cout<<"\n o[j+1].yc ="<<out[j+1].yc<<" o[j].yc ="<< out[j].yc;
        s=(float)(out[j+1].yc-out[j].yc)/(out[j+1].xc-out[j].xc);
        cout<<"\nslope is :"<<s;
        e[k].xc =out[j].xc+(int)((ymin-out[j].yc)/s);
        e[k].yc=ymin;
        k++;
    }
    else if((left_below(out[j].yc,ymin))&&(left_below(out[j+1].yc,ymin)))
    {
        e[k].yc=out[j+1].yc;
        e[k].xc=out[j+1].xc ;
        k++;
    }
    }
e[k].yc=e[0].yc;
 e[k].xc=e[0].xc;
```

```
        k++;
        m=k;
    for(i=0;i<m;i++)
        {
            cout<<"\neX: "<<e[i].xc<<" eY: "<<e[i].yc;
        }
  }

if(index==2)    // Right clipping
  {
        k=0;
        for(j=0;j<m-1;j++)
        {
            if(!(right_above(e[j].xc,xmax))&&(right_above(e[j+1].xc,xmax)))
            {
                s=(float)(e[j+1].yc-e[j].yc)/(e[j+1].xc-e[j].xc);
                out[k].yc =e[j].yc+(int)(s*(xmax-e[j].xc));
                out[k].xc=xmax;
                k++;
                out[k].yc=e[j+1].yc;
                out[k].xc=e[j+1].xc ;
                k++;
            }
            if(!(right_above(e[j].xc,xmax))&&!(right_above(e[j+1].xc,xmax)))
            {
            }
            if((right_above(e[j].xc,xmax))&&!(right_above(e[j+1].xc,xmax)))
            {
                s=(float)(e[j+1].yc-e[j].yc)/(e[j+1].xc-e[j].xc);
                out[k].yc =e[j].yc+(int)(s*(xmax-e[j].xc));
                cout<<"\nslope"<<s;
```

```
            out[k].xc=xmax;
            k++;
        }
        if((right_above(e[j].xc,xmax))&&(right_above(e[j+1].xc,xmax)))
        {
            out[k].yc=e[j+1].yc;
            out[k].xc=e[j+1].xc ;
            k++;
        }
    }
    out[k].yc=out[0].yc;
    out[k].xc=out[0].xc;
    k++;

    m=k;
    for(i=0;i<m;i++)
        {
                cout<<"\noX1: "<<out[i].xc<<" oY1: "<<out[i].yc;
        }
 }
if(index==3) //top clipping
 {
    k=0;
    for(j=0;j<m-1;j++)
    {
        if(!(right_above(out[j].yc,ymax))&&(right_above(out[j+1].yc,ymax)))
        {
            s=(float)(out[j+1].yc-out[j].yc)/(out[j+1].xc-out[j].xc);
            e[k].xc = out[j].xc+(int)((ymax-out[j].yc)/s);
            e[k].yc=ymax;
            k++;
```

```
                e[k].yc=out[j+1].yc;
                e[k].xc=out[j+1].xc ;
                k++;
            }
            if(!(right_above(out[j].yc,ymax))&&!(right_above(out[j+1].yc,ymax)))
            {
            }
           if((right_above(out[j].yc,ymax))&&!(right_above(out[j+1].yc,ymax)))
            {
                s=(float)(out[j+1].yc-out[j].yc)/(out[j+1].xc-out[j].xc);
               e[k].xc = out[j].xc+(int)((ymax-out[j].yc)/s);
                e[k].yc=ymax;
                k++;
            }
            if((right_above(out[j].yc,ymax))&&(right_above(out[j+1].yc,ymax)))
            {
                e[k].yc=out[j+1].yc;
                e[k].xc=out[j+1].xc ;
                k++;
            }
          }
           e[k].yc=e[0].yc; //complete polygon with first vertex
           e[k].xc=e[0].xc;
           k++;
           m=k;
           for(i=0;i<m;i++)
            {
                    cout<<"\neX: "<<e[i].xc<<" eY: "<<e[i].yc;
            }
      }
}
```

```
 glColor3f(0.0,1.0,0.0);
   for(i=0;i<k-1;i++)
        {
             cout<<"\nx: "<<e[i].xc<<"y: "<<e[i].yc;  //polygon has drawn line by line after clipping
          glBegin(GL_LINES);
          glVertex2i(e[i].xc,e[i].yc-250);
          glVertex2i(e[i+1].xc,e[i+1].yc-250);
          glEnd();
        }
  glFlush();
      }
};
void choice (void)
{
    int i,ch;
           glPointSize(4.0);
           glFlush();
           for(i=-w;i<=w;i++)
           {
                    setpixel(i,0);
           //       setpixel(0,i);
           }
}
void init()
{
  glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(1.0,1.0,1.0,0.0);
    glPointSize(2.0);
    glMatrixMode(GL_PROJECTION);
           glLoadIdentity();
           gluOrtho2D(-w/2,w/2,-h/2,h/2);
```

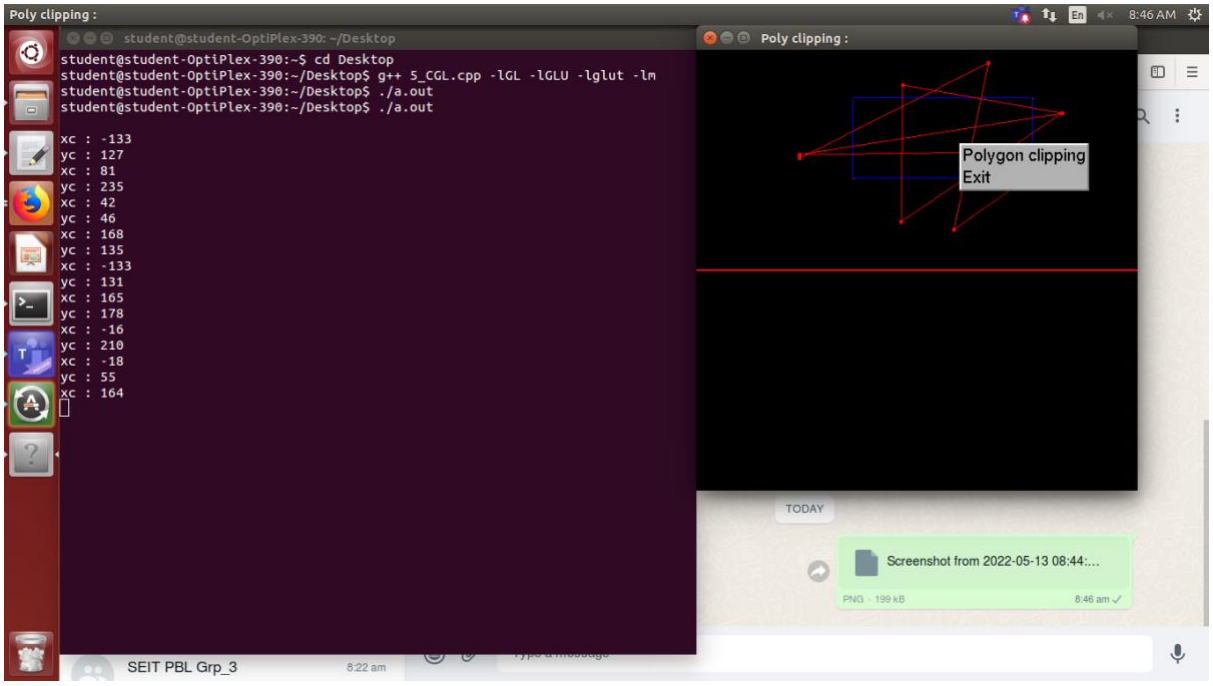
```
            glFlush();
}
void max_min()
    {
 if(win[0][0]<win[2][0])  //setting xmin, xmax either lower left corner or upper right corner
        {
            xmin=win[0][0];
            xmax=win[2][0];
        }
        else
        {
            xmin=win[2][0];
            xmax=win[0][0];
        }
        if(win[0][1]<win[2][1])
        {
            ymin=win[0][1];
            ymax=win[2][1];
        }
        else
        {
            ymin=win[2][1];
            ymax=win[0][1];
        }
}
void draw_window()
{
        win[1][0]=win[0][0]; //window is drawn
        win[1][1]=win[2][1];
        win[3][0]=win[2][0];
        win[3][1]=win[0][1];
```

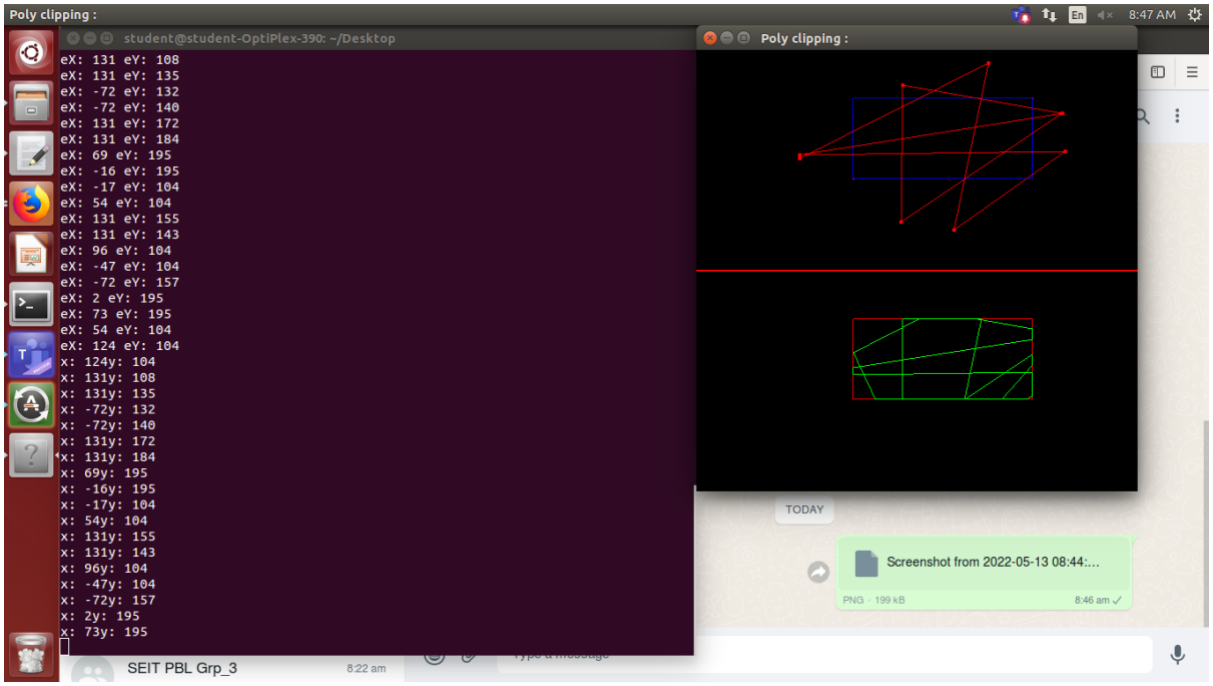
```
        glPointSize(4);
        glColor3f(0.0f,0.0f,1.0f);
        glBegin(GL_LINE_LOOP);
        for(int i=0;i<4;i++)
                                glVertex2f(win[i][0],win[i][1]);
                               glEnd();
                               glFlush();
}
 void draw_winoutput()
{
                wino[0][0]=win[0][0]; //before clipping window drawing
         wino[1][0]=win[1][0];
         wino[2][0]=win[2][0];
         wino[3][0]=win[3][0];
        wino[0][1]=win[0][1]-250; //after clipping window drawing
         wino[1][1]=win[1][1]-250;
         wino[2][1]=win[2][1]-250;
         wino[3][1]=win[3][1]-250;
         glColor3f(1.0f,0.0f,0.0f);
         glPointSize(4);
         glBegin(GL_LINE_LOOP);
         for(int i=0;i<4;i++)
                                glVertex2f(wino[i][0],wino[i][1]);
                               glEnd();
                               glFlush();
}
void menu(int item)
{ /* Callback called when the user clicks the middle mouse button */
   clipping c;
   cout<<"menu: you clicked item:\t" <<item;
   if(item==1)
```

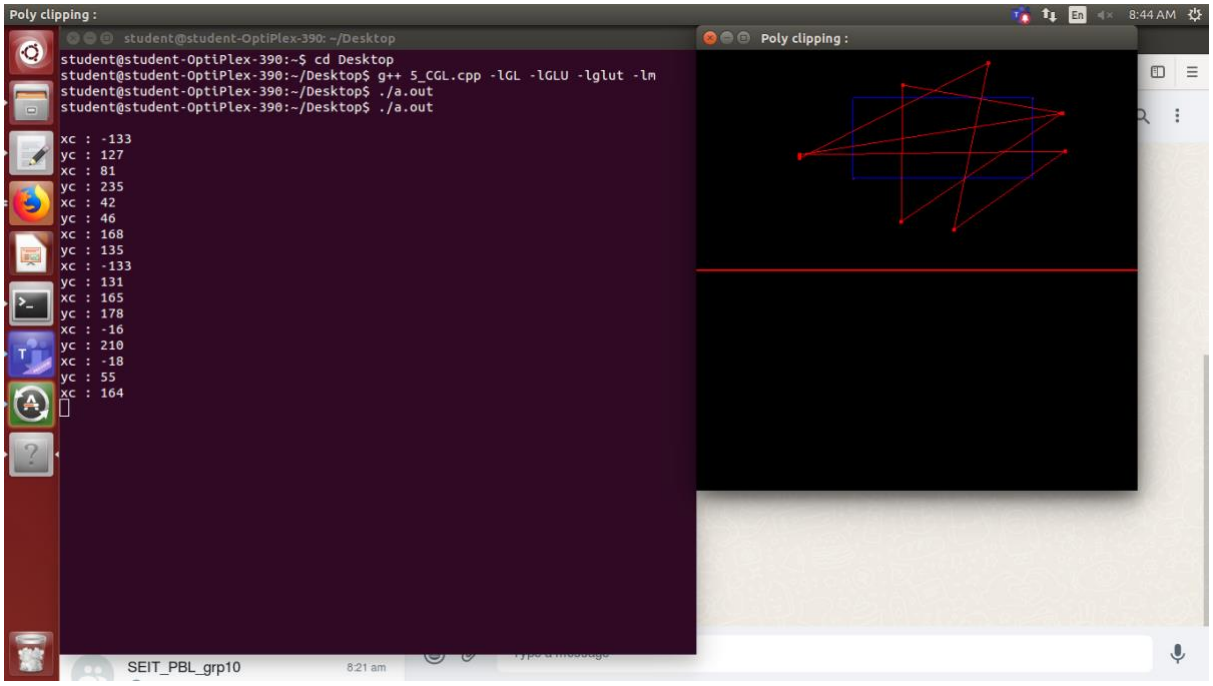
```
   {
        draw_winoutput();
        c.suth_hodge();
   }
  if(item==2)
            exit(0);
}
void mouseClick(int button, int state, int x, int y)
{
clipping c;
   if(state== GLUT_DOWN)
            {
if(button==GLUT_LEFT_BUTTON)
                        {
          if(flag==1)
          {
                    e[m].xc=(x-250);
                                    e[m].yc=(250-y);
                cout<<"\nxc : "<<e[m].xc;
                cout<<"\nyc : "<<e[m].yc;
                glColor3f(1.0,0.0,0.0);
                glPointSize(4);
                glBegin(GL_POINTS);
                                      glVertex2f(e[m].xc,e[m].yc);
                        glEnd();
                if(m>0)
                {
                                    glBegin(GL_LINES);
                                      glVertex2f(e[m-1].xc,e[m-1].yc);
                                     glVertex2f(e[m].xc,e[m].yc);
                        glEnd();
```

```
            }
                    m++;
                        glFlush();
                        flag1=1;
        }
        else
        {
                        win[k][0]=(x-250);
                        win[k][1]=(250-y);
                        glColor3f(1,0,0);
                        glBegin(GL_POINTS);
                        glVertex2f(win[k][0],win[k][1]);
                        glEnd();
                        k=k+2;
                        glFlush();
       }
                }
                if(button==GLUT_RIGHT_BUTTON)
                {
                     flag=1;
        if(flag1==1)
        {
            glColor3f(1.0,0.0,0.0);
            glPointSize(4);
            glBegin(GL_LINES);
glVertex2f(e[m-1].xc,e[m-1].yc);
glVertex2f(e[0].xc,e[0].yc);

                            glEnd();
                            e[m].yc=e[0].yc;
                            e[m].xc=e[0].xc;
```

```
            glFlush();
            n++;
          }
          else
          {
          max_min();
          draw_window();
         }
       }
          }
}
int main(int argc, char **argv)
{
   glutInit (&argc, argv); //initialise the device(glut window) by 'init' routine to client window
   glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
   glutInitWindowSize(w,h);
   glutCreateWindow ("Poly clipping :");
   glutDisplayFunc (choice); //call backs are function pointers
   init();
  glutMouseFunc(mouseClick);
   glutCreateMenu (menu); /* Create the first menu & add items */
   glutAddMenuEntry ("Polygon clipping", 1);
   glutAddMenuEntry ("Exit", 2);
   glutAttachMenu (GLUT_MIDDLE_BUTTON); // attach menu to middle button of mouse
   glutMainLoop();
   return 0;
}
```


Poly clipping :
8:46 AM
student@student-OptiPlex-390: ~/Desktop
student@student-OptiPlex-390:~$ cd Desktop
student@student-OptiPlex-390:~/Desktop$ g++ 5_CGL.cpp -lGL -lGLU -lglut -lm
student@student-OptiPlex-390:~/Desktop$ ./a.out
student@student-OptiPlex-390:~/Desktop$ ./a.out
xc : -133
yc : 127
xc : 81
yc : 235
xc : 42
yc : 46
xc : 168
yc : 135
xc : -133
yc : 131
xc : 165
yc : 178
xc : -16
yc : 210
xc : -18
yc : 55
xc : 164
Poly clipping :
Polygon clipping
Exit
TODAY
Screenshot from 2022-05-13 08:44:...
PNG · 199 kB
8:46 am
SEIT PBL Grp_3
8:22 am


Poly clipping :
8:47 AM
student@student-OptiPlex-390: ~/Desktop
eX: 131 eY: 108
eX: 131 eY: 135
eX: -72 eY: 132
eX: -72 eY: 140
eX: 131 eY: 172
eX: 131 eY: 184
eX: 69 eY: 195
eX: -16 eY: 195
eX: -17 eY: 104
eX: 54 eY: 104
eX: 131 eY: 155
eX: 131 eY: 143
eX: 96 eY: 104
eX: -47 eY: 104
eX: -72 eY: 157
eX: 2 eY: 195
eX: 73 eY: 195
eX: 54 eY: 104
eX: 124 eY: 104
x: 124y: 104
x: 131y: 108
x: 131y: 135
x: -72y: 132
x: -72y: 140
x: 131y: 172
x: 131y: 184
x: 69y: 195
x: -16y: 195
x: -17y: 104
x: 54y: 104
x: 131y: 155
x: 131y: 143
x: 96y: 104
x: -47y: 104
x: -72y: 157
x: 2y: 195
x: 73y: 195
Poly clipping :
TODAY
Screenshot from 2022-05-13 08:44:...
PNG · 199 kB
8:46 am
SEIT PBL Grp_3
8:22 am



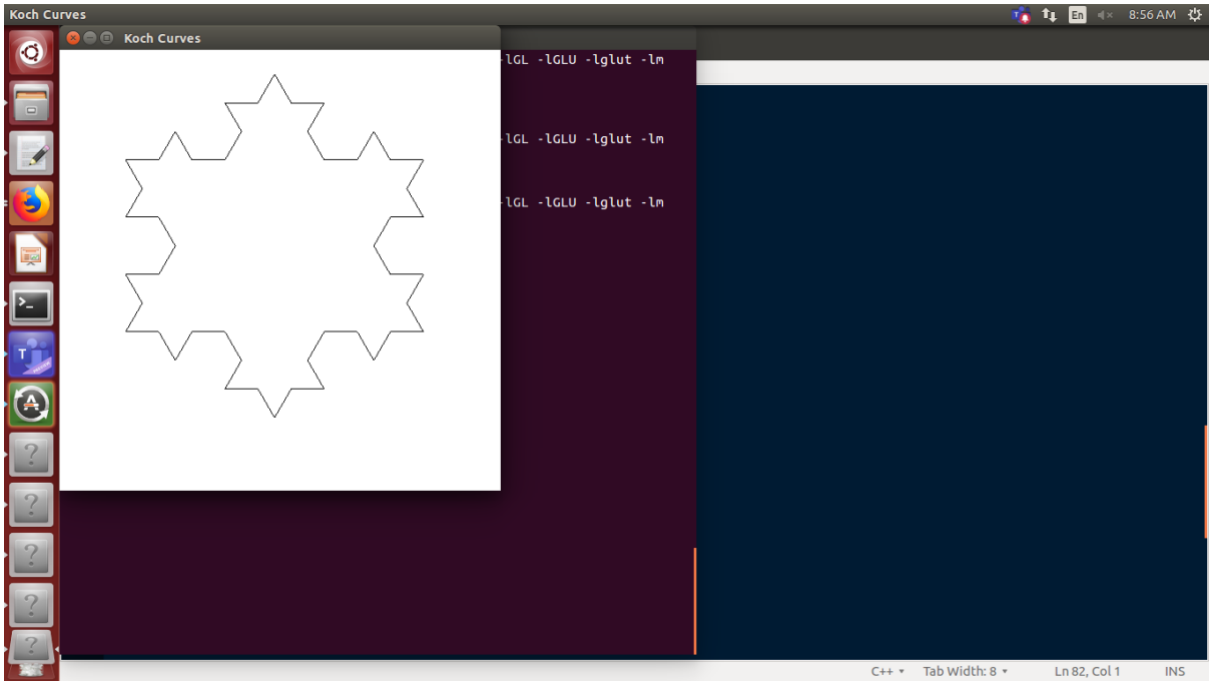
Write a program to generate fractals using Koch curve

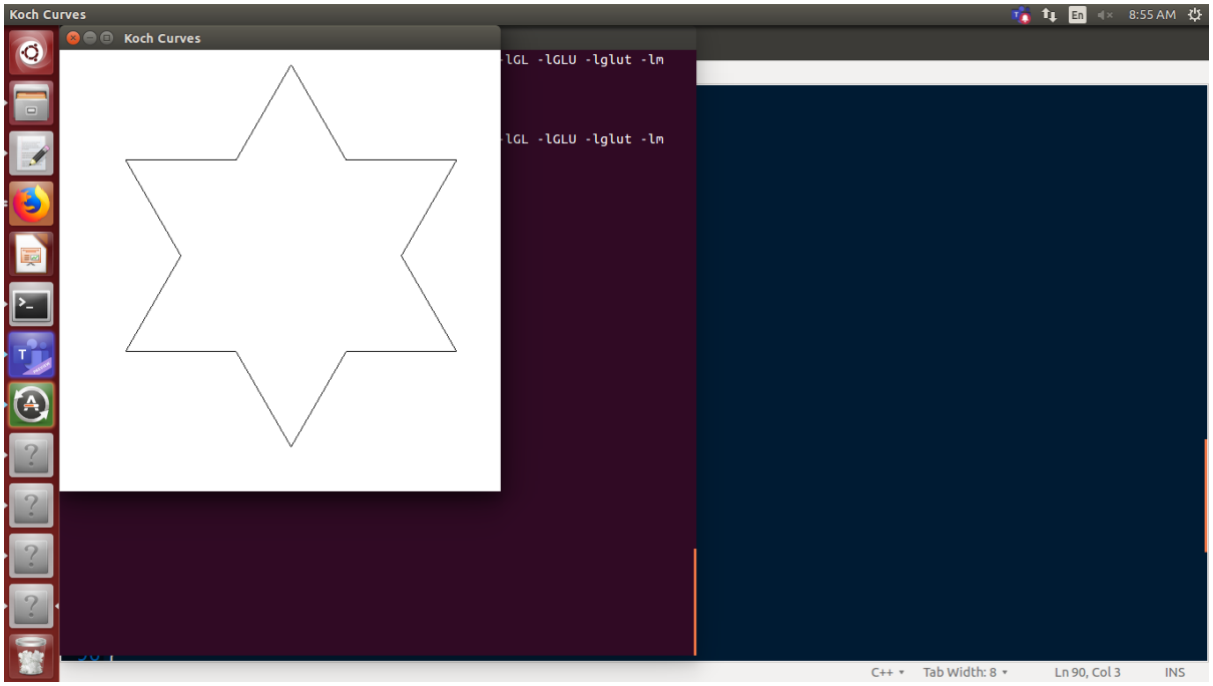
ASSIGNMENT 7

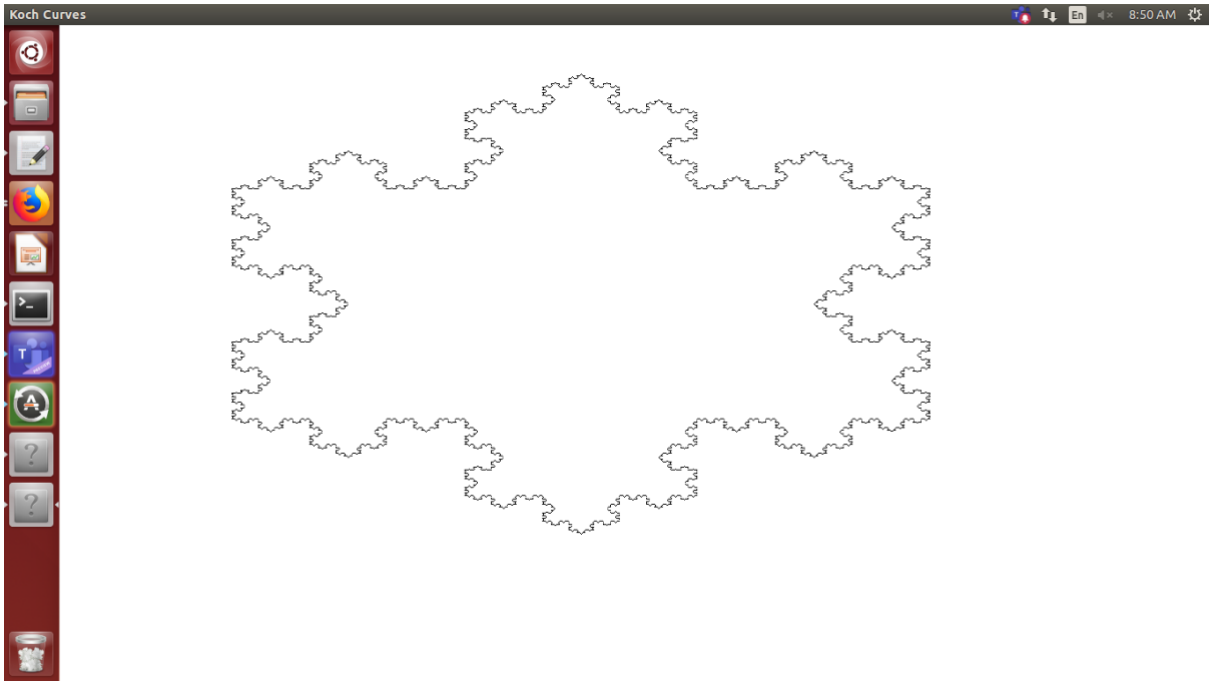
```
#include <GL/glut.h>

#include <math.h>

GLfloat oldx=-0.7,oldy=0.5;

void drawkoch(GLfloat dir,GLfloat len,GLint iter)

{


GLdouble dirRad = 0.0174533 * dir ;


GLfloat newX = oldx + len * cos(dirRad);

        GLfloat newY = oldy + len * sin(dirRad);

        if (iter==0)
```

```
        {
            glVertex2f(oldx, oldy);
            glVertex2f(newX, newY);
            oldx = newX;
            oldy = newY;
       }
Else
 {
            iter--;
            //draw the four parts of the side   _ /  \  _
            drawkoch(dir, len, iter);
           dir += 60.0;
          drawkoch(dir, len, iter);
         dir -= 120.0;
         drawkoch(dir, len, iter);
        dir += 60.0;
 drawkoch(dir, len, iter);
}
}
void display()
{
 glClearColor(1.0,1.0,1.0,0);
        glColor3f(0.0, 0.0, 0.0);
 glClear( GL_COLOR_BUFFER_BIT );
 glBegin(GL_LINES);
 /*
             drawkoch(0.0,0.5,1);
     drawkoch(-120.0, 0.5, 1);
  drawkoch(120.0,0.5,1);
 //
 drawkoch(0.0,0.15,2);
```

```
 drawkoch(-120.0, 0.15, 2);
drawkoch(120.0,0.15,2);
            */
 drawkoch(0.0,0.05,3);
 drawkoch(-120.0, 0.05, 3);
 drawkoch(120.0,0.05,3);
 glEnd();
 glFlush();
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
glutCreateWindow("Koch Curves");
glutDisplayFunc(display);
glutMainLoop();
}
```


Koch Curves
8:56 AM
Koch Curves
lGL -lGLU -lglut -lm
lGL -lGLU -lglut -lm
lGL -lGLU -lglut -lm
C++ Tab Width: 8 Ln 82, Col 1 INS


Koch Curves
8:55 AM
Koch Curves
lGL -lGLU -lglut -lm
lGL -lGLU -lglut -lm
C++ Tab Width: 8 Ln 90, Col 3 INS


Koch Curves
En
8:50 AM

Write a program to perform 2D translation

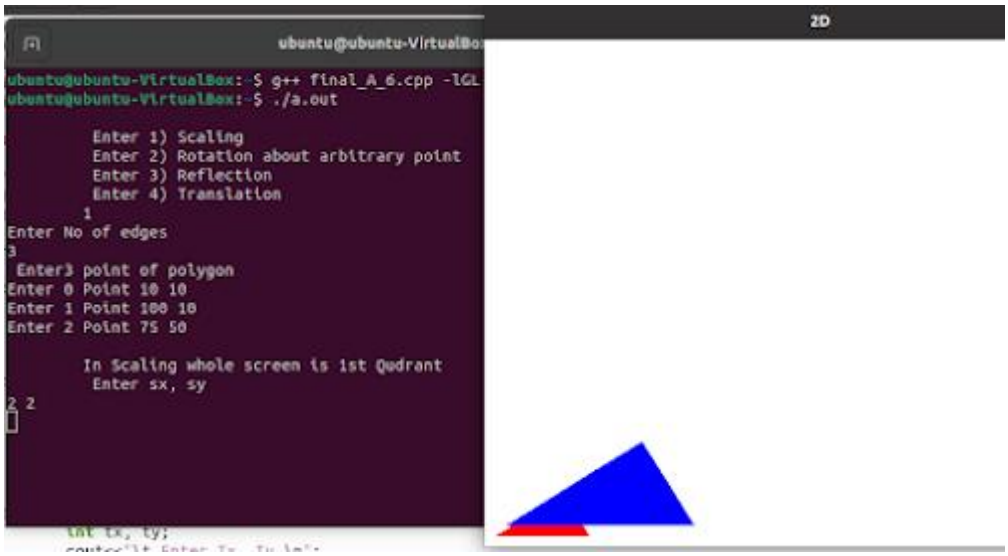
ASSIGNMENT 6

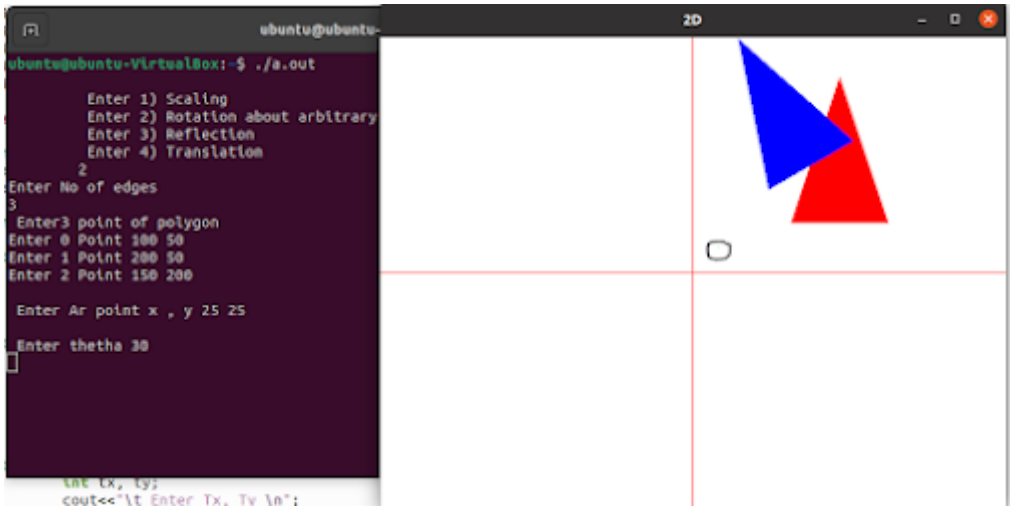
```
#include <iostream>
#include <math.h>
#include <time.h>
#include <GL/glut.h>
#include <vector>

using namespace std;

int edge;
vector<int> xpoint;
vector<int> ypoint;

int ch;

double round(double d){

    return floor(d + 0.5);
}

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 translation(){
    int tx, ty;
    cout<<"\t Enter Tx, Ty \n";
    cin>> tx>> ty;

    //Translate the point
    for(int i=0;i<edge;i++){

        xpoint[i] = xpoint[i] + tx;
        ypoint[i] = ypoint[i] + ty;

    }


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

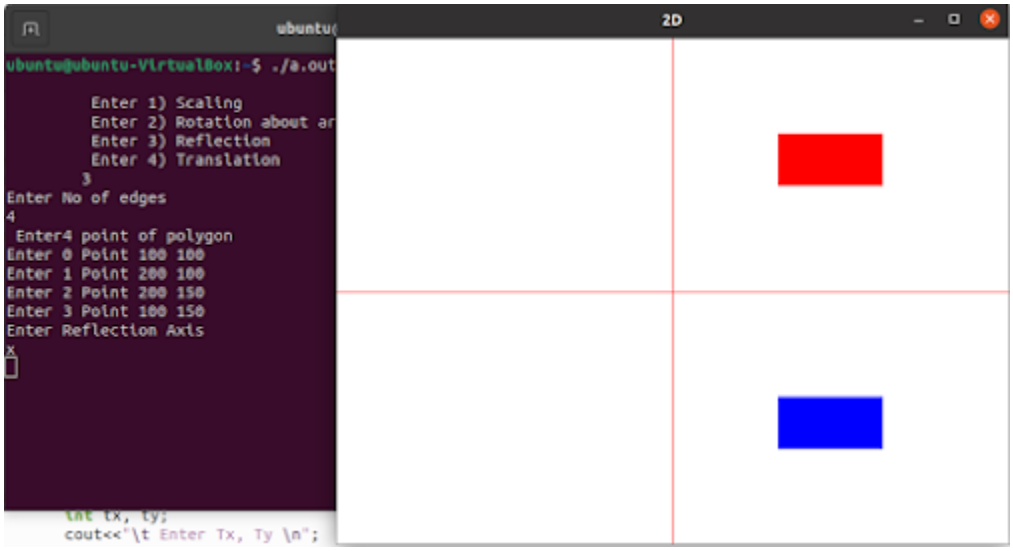
```
void rotaion(){
    int cx, cy;
    cout<<"\n Enter Ar point x , y ";
    cin >> cx >> cy;

    cx = cx+320;
    cy = cy+240;
    glColor3f(0.0, 1.0, 0.0);
    glBegin(GL_POINTS);
        glVertex2i(cx,cy);
    glEnd();
    glFlush();

    double the;
    cout<<"\n Enter thetha ";
    cin>>the;
    the = the * 3.14/180;

    glColor3f(0,0,1.0);
    glBegin(GL_POLYGON);
        for(int i=0;i<edge;i++){
            glVertex2i(round(((xpoint[i] - cx)*cos(the) -
((ypoint[i]-cy)*sin(the))) + cx),
                    round((xpoint[i] - cx)*sin(the) + ((ypoint[i]-
cy)*cos(the))) + cy));
        }
    glEnd();
    glFlush();
}

void scale(){

    glColor3f(1.0,0,0);
    glBegin(GL_POLYGON);
        for(int i=0;i<edge;i++){
            glVertex2i(xpoint[i]-320,ypoint[i]-240);
        }
    glEnd();
    glFlush();
    cout<<"\n\tIn Scaling whole screen is 1st Qudrant \n";
    int sx, sy;
    cout<<"\t Enter sx, sy \n";
    cin>> sx>> sy;

    //scale the point
    for(int i=0;i<edge;i++){

        xpoint[i] = (xpoint[i]-320) * sx;
        ypoint[i] = (ypoint[i]-240) * sy;
    }

    glColor3f(0,0,1.0);
    glBegin(GL_POLYGON);
        for(int i=0;i<edge;i++){
            glVertex2i(xpoint[i],ypoint[i]);
        }
```

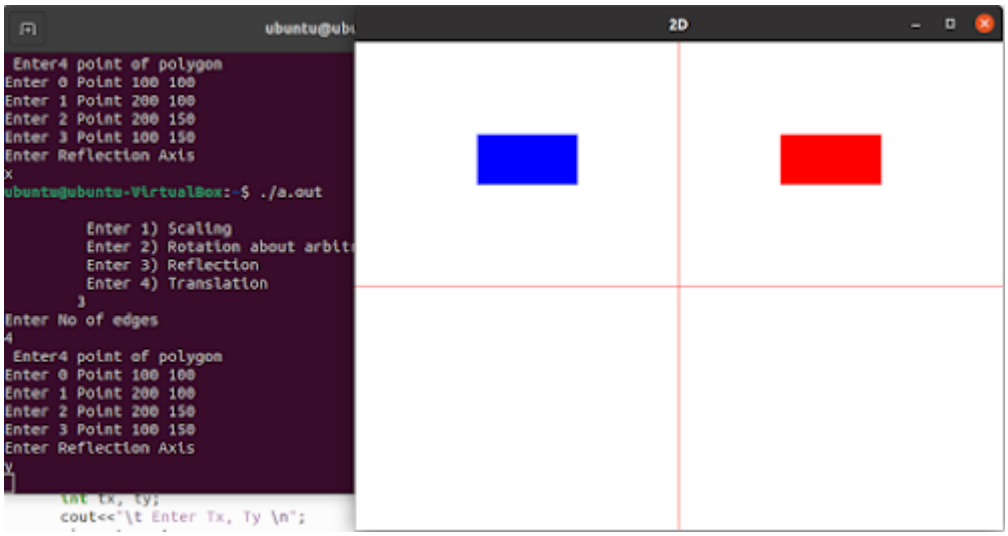
```
    glEnd();
    glFlush();
}

void reflection(){
    char reflection;
    cout<<"Enter Reflection Axis \n";
    cin>> reflection;

    if(reflection == 'x' || reflection == 'X'){

        glColor3f(0.0,0.0,1.0);
        glBegin(GL_POLYGON);
            for(int i=0;i<edge;i++){
                glVertex2i(xpoint[i], (ypoint[i] * -1)+480);
            }
        glEnd();
        glFlush();

    }
    else if(reflection == 'y' || reflection == 'Y'){
        glColor3f(0.0,0.0,1.0);
        glBegin(GL_POLYGON);
            for(int i=0;i<edge;i++){
                glVertex2i((xpoint[i] * -1)+640,(ypoint[i]));
            }
        glEnd();
        glFlush();
    }
}

void Draw(){

    if(ch==2 || ch==3 || ch==4){
        glColor3f(1.0,0,0);
        glBegin(GL_LINES);
            glVertex2i(0,240);
            glVertex2i(640,240);
        glEnd();
        glColor3f(1.0,0,0);
        glBegin(GL_LINES);
            glVertex2i(320,0);
            glVertex2i(320,480);
        glEnd();
        glFlush();

        glColor3f(1.0,0,0);
        glBegin(GL_POLYGON);
            for(int i=0;i<edge;i++){
                glVertex2i(xpoint[i],ypoint[i]);
            }
        glEnd();
        glFlush();
    }
    if(ch==1){
        scale();
```

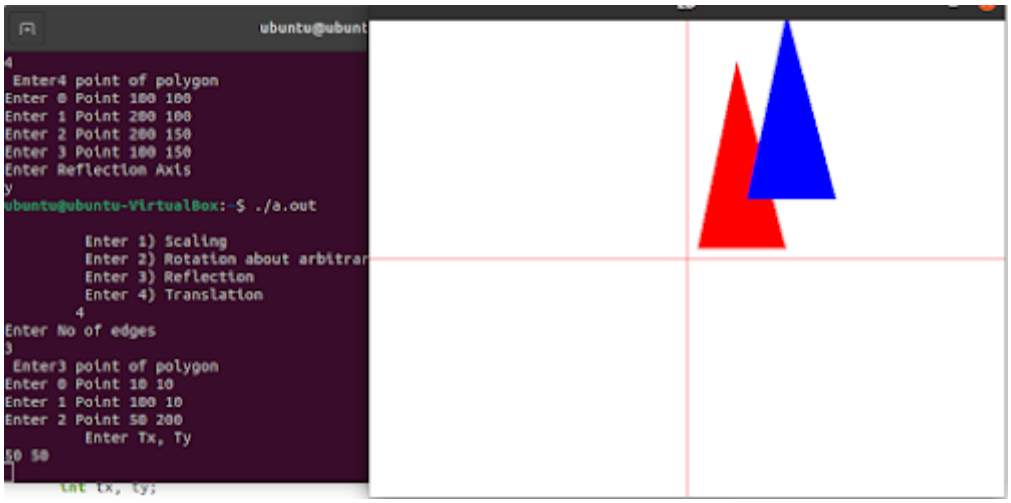
```
    }
    else if(ch == 2){
        rotaion();
    }
    else if( ch == 3){
        reflection();
    }
    else if (ch == 4){
        translation();
    }
}
int main(int argc, char** argv){

    cout<<"\n \t Enter 1) Scaling ";
    cout<<"\n \t Enter 2) Rotation about arbitrary point";
    cout<<"\n \t Enter 3) Reflection";
    cout<<"\n \t Enter 4) Translation  \n \t";

    cin>>ch;

    if(ch==1 || ch==2 || ch==3 || ch==4){

        cout<<"Enter No of edges \n";
        cin>> edge;

        int xpointnew, ypointnew;
        cout<<" Enter"<< edge <<" point of polygon \n";
        for(int i=0;i<edge;i++){

            cout<<"Enter "<< i << " Point ";
            cin>>xpointnew>>ypointnew;

            xpoint.push_back(xpointnew+320);
            ypoint.push_back(ypointnew+240);

        }
            glutInit(&argc, argv);
            glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
            glutInitWindowSize(640,480);
            glutInitWindowPosition(200,200);
            glutCreateWindow("2D");
            init();
            glutDisplayFunc(Draw);

        glutMainLoop();
            return 0;
        }
        else{
            cout<<"\n \t Check Input run again";
        return 0;
    }
}

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


2D
ubuntu@ubuntu-VirtualBox:~$ g++ final_A_6.cpp -lGL
ubuntu@ubuntu-VirtualBox:~$ ./a.out
Enter 1) Scaling
Enter 2) Rotation about arbitrary point
Enter 3) Reflection
Enter 4) Translation
1
Enter No of edges
3
Enter3 point of polygon
Enter 0 Point 10 10
Enter 1 Point 100 10
Enter 2 Point 75 50
In Scaling whole screen is 1st Qudrant
Enter sx, sy
2 2


2D
ubuntu@ubuntu-VirtualBox:~$ ./a.out
Enter 1) Scaling
Enter 2) Rotation about arbitrary
Enter 3) Reflection
Enter 4) Translation
2
Enter No of edges
3
Enter3 point of polygon
Enter 0 Point 100 50
Enter 1 Point 200 50
Enter 2 Point 150 200
Enter Ar point x , y 25 25
Enter thetha 30


2D
ubuntu@ubuntu-VirtualBox:~$ ./a.out
Enter 1) Scaling
Enter 2) Rotation about ar
Enter 3) Reflection
Enter 4) Translation
3
Enter No of edges
4
Enter4 point of polygon
Enter 0 Point 100 100
Enter 1 Point 200 100
Enter 2 Point 200 150
Enter 3 Point 100 150
Enter Reflection Axis
x


2D
Enter4 point of polygon
Enter 0 Point 100 100
Enter 1 Point 200 100
Enter 2 Point 200 150
Enter 3 Point 100 150
Enter Reflection Axis
x
ubuntu@ubuntu-VirtualBox:~$ ./a.out
Enter 1) Scaling
Enter 2) Rotation about arbit
Enter 3) Reflection
Enter 4) Translation
3
Enter No of edges
4
Enter4 point of polygon
Enter 0 Point 100 100
Enter 1 Point 200 100
Enter 2 Point 200 150
Enter 3 Point 100 150
Enter Reflection Axis
y
cout<<"\t Enter Tx, Ty \n";


4
Enter4 point of polygon
Enter 0 Point 100 100
Enter 1 Point 200 100
Enter 2 Point 200 150
Enter 3 Point 100 150
Enter Reflection Axis
y
ubuntu@ubuntu-VirtualBox:~$ ./a.out
Enter 1) Scaling
Enter 2) Rotation about arbitrar
Enter 3) Reflection
Enter 4) Translation
4
Enter No of edges
3
Enter3 point of polygon
Enter 0 Point 10 10
Enter 1 Point 100 10
Enter 2 Point 50 200
Enter Tx, Ty
50 50