**PROGRAM 1:**

**AIM: Program to draw a point on the screen.**

#include<GL/gl.h>

#include<GL/glut.h>

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(6.0);

glBegin(GL\_POINTS);

glVertex2i(50,50);

glColor3f(0.5,0.0,0.0);

glVertex2i(10,25);

glColor3f(0.1,0.2,0.3);

glEnd();

glFlush();

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("point plot");

glClearColor(0.0,0.0,0.0,0.0);

glOrtho(-100,100,-100,100,-10,10);

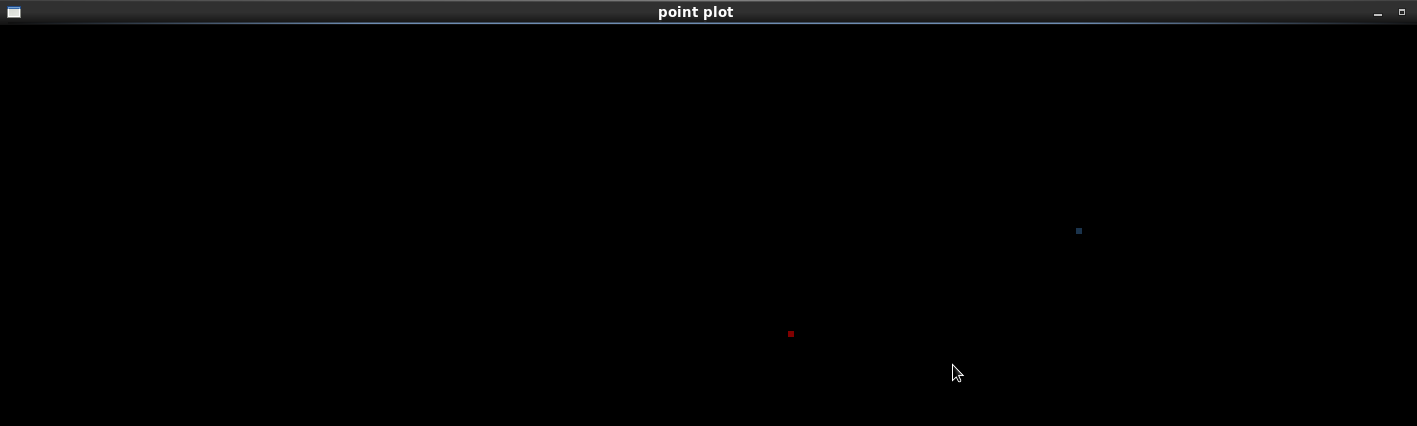
glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**OUTPUT:**



**PROGRAM 2:**

**AIM:Program to draw a line on the screen.**

#include<GL/gl.h>

#include<GL/glut.h>

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(6.0);

glBegin(GL\_LINES);

glVertex2i(50,50);

glVertex2i(10,10);

glColor3f(0.5,0.0,0.0);

glVertex2i(30,30);

glVertex2i(20,20);

glColor3f(0.0,1.0,0.0);

glEnd();

glFlush();

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("Line Ploting");

glClearColor(0.0,0.0,0.0,0.0);

glOrtho(-100,100,-100,100,-10,10);

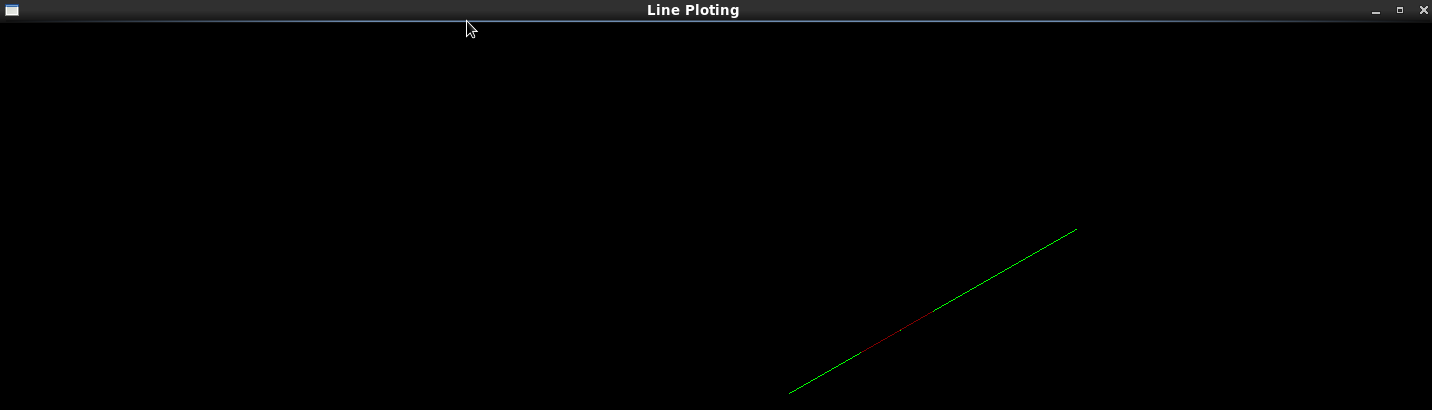
glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**OUTPUT:**



**PROGRAM 3:**

**AIM : Program to draw a line using DDA algorithm.**

#include<stdio.h>

#include<GL/gl.h>

#include<GL/glut.h>

float x,y,x1,z1,x2,y2,dx,dy,step;

void dda()

{

int xinc,yinc,k;

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(6.0);

glColor3f(0.0,0.0,1.0);

dx=x2-x1;

dy=y2-z1;

if(abs(dx)>abs(dy))

step=abs(dx);

else

step=abs(dy);

xinc=dx/(float)step;

yinc=dy/(float)step;

x=x1;

y=z1;

for(k=0;k<=step;k++)

{

glBegin(GL\_LINES);

glVertex2f(x,y);

x=x+xinc;

y=y+yinc;

glVertex2f(x,y);

}

glEnd();

glFlush();

}

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

{

printf("\n enter the coordinate of x1 and y1\n");

scanf("%f%f",&x1,&z1);

printf("\n enter the coordinates of x2 and y2\n");

scanf("%f%f",&x2,&y2);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("DDA line drawing");

glClearColor(0.0,0.0,0.0,0.0);

glOrtho(-100,100,-100,100,-10,10);

glutDisplayFunc(dda);

glutMainLoop();

return 0;

}

**OUTPUT:**

[Nidhin@DBCPC12 dda\_line]$ g++ dda\_line.cpp -lglut

[Nidhin@DBCPC12 dda\_line]$ ./a.out

enter the coordinate of x1 and y1

30

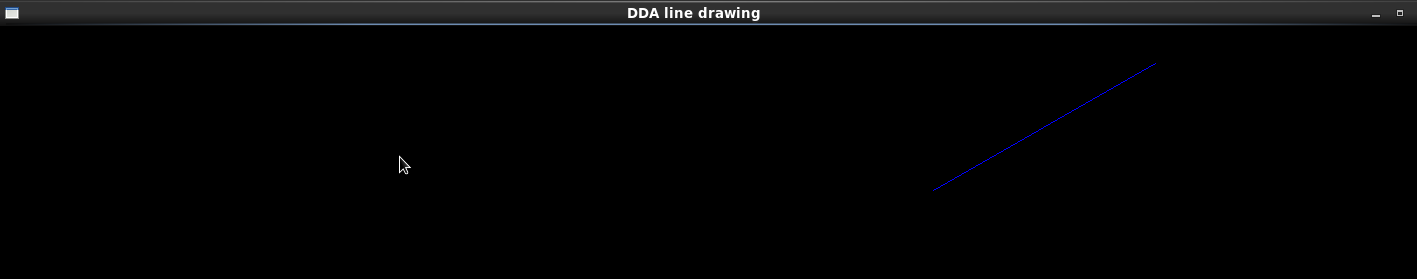
60

enter the coordinates of x2 and y2

60

90

[Nidhin@DBCPC12 dda\_line]$



**PROGRAM 4:**

**AIM: Program to draw a line using Bresenham Line drawing algorithm.**

#include<stdio.h>

#include<GL/gl.h>

#include<GL/glut.h>

float x1,x2,z1,y2,x,y,step,p,dx,dy;

void bresenham()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(9.0);

glColor3f(1.0,0.0,0.0);

int k;

dx=x2-x1;

dy=y2-z1;

step=dx-1;

p=2\*(dy-dx);

x=x1;

y=z1;

for(k=0;k<=step;k++)

{

glBegin(GL\_POINTS);

glVertex2f(x,y);

if(p<0)

{

x=x+1;

p=p+(2\*dy);

}

else

{

x=x+1;

y=y+1;

p=p+2\*(dy-dx);

}

glVertex2f(x,y);

}

glEnd();

glFlush();

}

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

{

printf("enter the coordinate of x1 and y1");

scanf("%f%f",&x1,&z1);

printf("enter the coordinates of x2 and y2");

scanf("%f%f",&x2,&y2);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("Breseham Line drawing");

glClearColor(0.0,0.0,0.0,0.0);

glOrtho(-100,100,-100,100,-10,10);

glutDisplayFunc(bresenham);

glutMainLoop();

return 0;

}

**OUTPUT:**

[Nidhin@DBCPC12 bresenham]$ g++ bresenham\_line.cpp -lglut

[Nidhin@DBCPC12 bresenham]$ ./a.out

enter the coordinate of x1 and y1

30

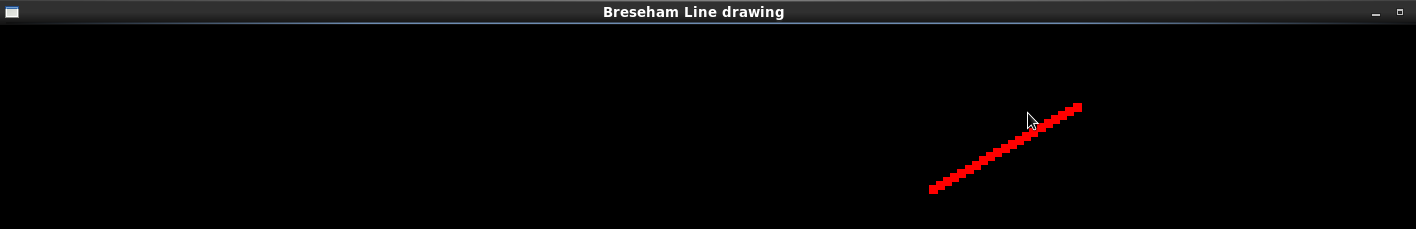
60

enter the coordinates of x2 and y2

50

100

[Nidhin@DBCPC12 bresenham]$



**PROGRAM 5:**

**AIM: Program to draw a circle.**

#include<GL/glut.h>

#include<stdio.h>

void circle(float xx,float yy);

float r,xc,yc,x,y,p;

void circle\_alg()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,0.0,0.0);

glPointSize(8.0);

x=0;

y=r;

glBegin(GL\_POINTS);

circle(x,y);

while(x<=y)

{

if(p<0)

{

x=x+1;

p=p+(2\*x)+1;

}

else

{

x=x+1;

y=y-1;

p=p+(2\*(x-y))+1;

}

circle(x,y);

}

glEnd();

glFlush();

}

void circle(float xx,float yy)

{

glVertex2f(xc+xx,yc+yy);

glVertex2f(xc+xx,yc-yy);

glVertex2f(xc-xx,yc+yy);

glVertex2f(xc-xx,yc-yy);

glVertex2f(xc+yy,yc+xx);

glVertex2f(xc+yy,yc-xx);

glVertex2f(xc-yy,yc-xx);

glVertex2f(xc-yy,yc+xx);

}

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

{

printf("enter the values for xc and yc\n");

scanf("%f%f",&xc,&yc);

printf("enter the value for radius r\n");

scanf("%f",&r);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("Circle Drawing");

glClearColor(0.0,0.0,0.0,0.0);

glOrtho(-100,100,-100,100,-10,10);

glutDisplayFunc(circle\_alg);

glutMainLoop();

return 0;

}

**OUTPUT:**

[Nidhin@DBCPC12 circle\_drawing]$ ./a.out

enter the values for xc and yc

10

20

enter the value for radius r

20

[Nidhin@DBCPC12 circle\_drawing]$



**PROGRAM 6:**

**AIM: Program to illustrate boundary fill algorithm.**

#include<GL/glut.h>

void boundaryfill(float,float,float[],float[]);

float fill[3]={1.0,0.0,0.0},old[3]={0.0,1.0,0.0};

void display()

{

glClearColor(0.0,0.0,0.0,0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3fv(old);

glBegin(GL\_LINE\_LOOP);

glVertex2i(100,150);

glVertex2i(400,150);

glVertex2i(400,350);

glVertex2i(100,350);

glEnd();

boundaryfill(200.5,160.3,fill,old);

glFlush();

}

void boundaryfill(float x,float y,float fill[3],float old[3])

{

float pix[3];

glReadPixels(x,y,1.0,1.0,GL\_RGB,GL\_FLOAT,pix);

if((!((pix[0]==old[0])&&(pix[1]==old[1])&&(pix[2]==old[2])))&&(!((pix[0]==fill[0])&&(pix[1]==fill[1])&&(pix[2]==fill[2]))))

{

glBegin(GL\_POINTS);

glColor3fv(fill);

glVertex2f(x,y);

glEnd();

glFlush();

boundaryfill(x-1,y,fill,old);

boundaryfill(x+1,y,fill,old);

boundaryfill(x,y+1,fill,old);

boundaryfill(x,y-1,fill,old);

}

}

int main(int argc,char \* argv[])

{

glutInit(&argc,argv);

glutInitWindowSize(640,480);

glutInitWindowPosition(0,0);

glutCreateWindow("Boundary fill");

glutDisplayFunc(display);

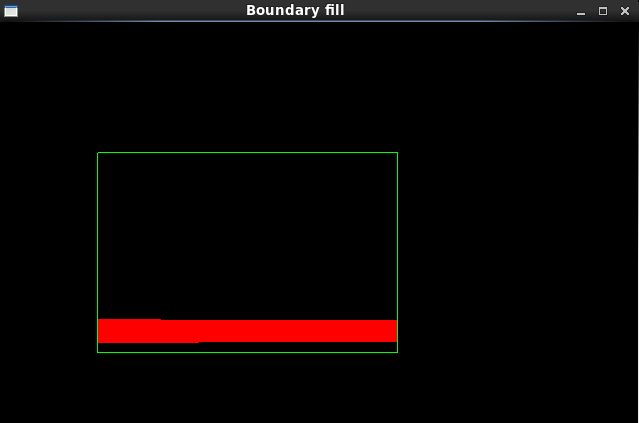
glOrtho(0.0,640.0,0.0,480.0,1.0,-1.0);

glutMainLoop();

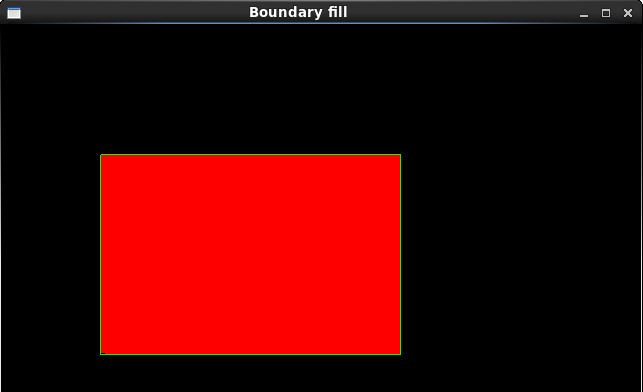
return 0;

}

**OUTPUT:**







**PROGRAM 7:**

**AIM: Program to illustrate flood fill algorithm.**

#include<GL/glut.h>

void floodfill(float,float,float[],float[]);

float fill[3]={1.0,0.0,0.0},old[3]={0.0,1.0,0.0};

void display()

{

glClearColor(0,0,0,0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3fv(old);

glBegin(GL\_POLYGON);

glVertex2i(100,150);

glVertex2i(400,150);

glVertex2i(400,350);

glVertex2i(100,350);

glEnd();

glFlush();

floodfill(200.0,160.0,fill,old);

glFlush();

}

void floodfill(float x,float y,float fill[3],float old[3])

{

float pix[3];

glReadPixels(x,y,1.0,1.0,GL\_RGB,GL\_FLOAT,pix);

if(pix[0]==old[0]&&pix[1]==old[1]&&pix[2]==old[2])

{

glBegin(GL\_POINTS);

glColor3fv(fill);

glVertex2f(x,y);

glEnd();

glFlush();

floodfill(x-1,y,fill,old);

floodfill(x+1,y,fill,old);

floodfill(x,y+1,fill,old);

floodfill(x,y-1,fill,old);

}

}

int main(int argc,char \*argv[])

{

glutInit(&argc,argv);

glutInitWindowSize(640,480);

glutCreateWindow("flood fill");

glutDisplayFunc(display);

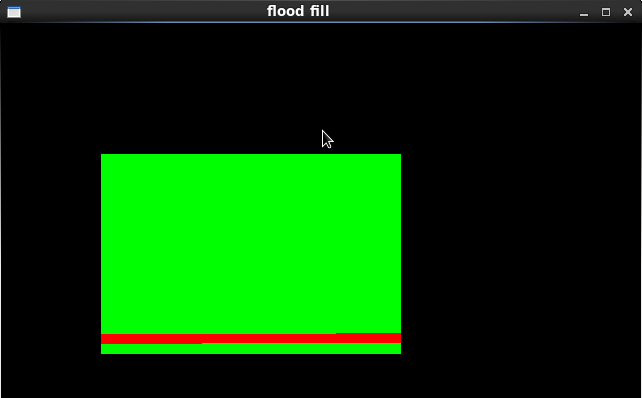
glOrtho(0.0,640.0,0.0,480.0,1.0,-1.0);

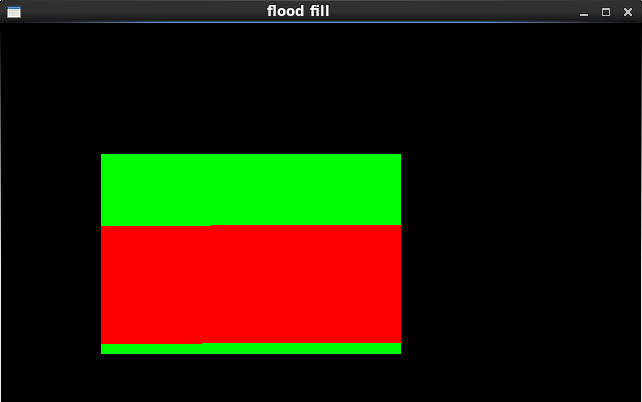
glutMainLoop();

return 0;

}

**OUTPUT:**

****

****

**PROGRAM 8:**

**AIM: Program to perform 2D transformations.**

#include<stdio.h>

#include<math.h>

#include<GL/glut.h>

int ch;

float x1=0.5,x2=0.8,x3=0.8,x4=0.5,y=0.5,y2=0.5,y3=0.8,y4=0.8;

float X1,X2,X3,X4,Y1,Y2,Y3,Y4;

void display(void)

{

float tx,ty;

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.7,0.3,0.3);

glPointSize(10.0);

glBegin(GL\_POLYGON);

glVertex2f(x1,y);

glVertex2f(x2,y2);

glVertex2f(x3,y3);

glVertex2f(x4,y4);

glEnd();

glColor3f(0.8,0.0,0.0);

glBegin(GL\_POLYGON);

glVertex2f(X1,Y1);

glVertex2f(X2,Y2);

glVertex2f(X3,Y3);

glVertex2f(X4,Y4);

glEnd();

glFlush();

}

void translate()

{

float tx,ty;

printf("enter value for tx and ty\n");

scanf("%f%f",&tx,&ty);

X1=x1+tx;

X2=x2+tx;

X3=x3+tx;

Y1=y+ty;

Y2=y2+ty;

Y3=y3+ty;

X4=x4+tx;

Y4=y4+ty;

}

void rotate()

{

int theta;

printf("enter an angle\n");

scanf("%d",&theta);

X1=x1\*cos(theta)-y\*sin(theta);

X2=x2\*cos(theta)-y2\*sin(theta);

X3=x3\*cos(theta)-y3\*sin(theta);

X4=x4\*cos(theta)-y4\*sin(theta);

Y1=x1\*sin(theta)+y\*cos(theta);

Y2=x2\*sin(theta)+y2\*cos(theta);

Y3=x3\*sin(theta)+y3\*cos(theta);

Y4=x4\*sin(theta)+y4\*cos(theta);

}

void scale()

{

float sx,sy;

printf("enter values for sx and sy\n");

scanf("%f%f",&sx,&sy);

X1=x1\*sx;

X2=x2\*sx;

X3=x3\*sx;

X4=x4\*sx;

Y1=y\*sy;

Y2=y2\*sy;

Y3=y3\*sy;

Y4=y4\*sy;

}

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

{

printf("2D TRANSFORMATION OPERATIONS ARE\n");

printf("1:TRANSLATION\n");

printf("2:ROTATION\n");

printf("3:SCALING\n");

printf("ENTER UR CHOICE\n");

scanf("%d",&ch);

switch(ch)

{

case 1:

translate();

break;

case 2:

rotate();

break;

case 3:

scale();

break;

default:

printf("wrong choice\n");

break;

}

glutInit(&argc,argv);

glutInitWindowSize(640,480);

glutInitWindowPosition(0,0);

glutCreateWindow("2D TRANSFORMATIONS");

glutDisplayFunc(display);

//glOrtho(0.0,640.0,0.0,480.0,1.0,-1.0);

glutMainLoop();

return 0;

}

**OUTPUT:**

[Nidhin@DBCPC12 2Dtransformations]$ g++ 2d\_transformations.cpp -lglut

[Nidhin@DBCPC12 2Dtransformations]$ ./a.out

2D TRANSFORMATION OPERATIONS ARE

1:TRANSLATION

2:ROTATION

3:SCALING

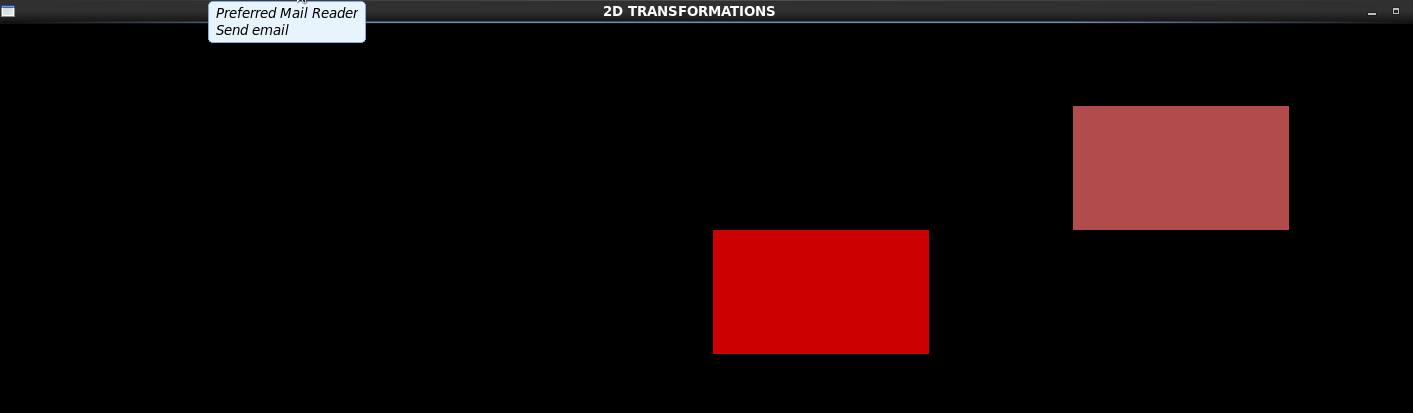
ENTER UR CHOICE

1

enter value for tx and ty

-0.5

-0.3



[Nidhin@DBCPC12 2Dtransformations]$ ./a.out

2D TRANSFORMATION OPERATIONS ARE

1:TRANSLATION

2:ROTATION

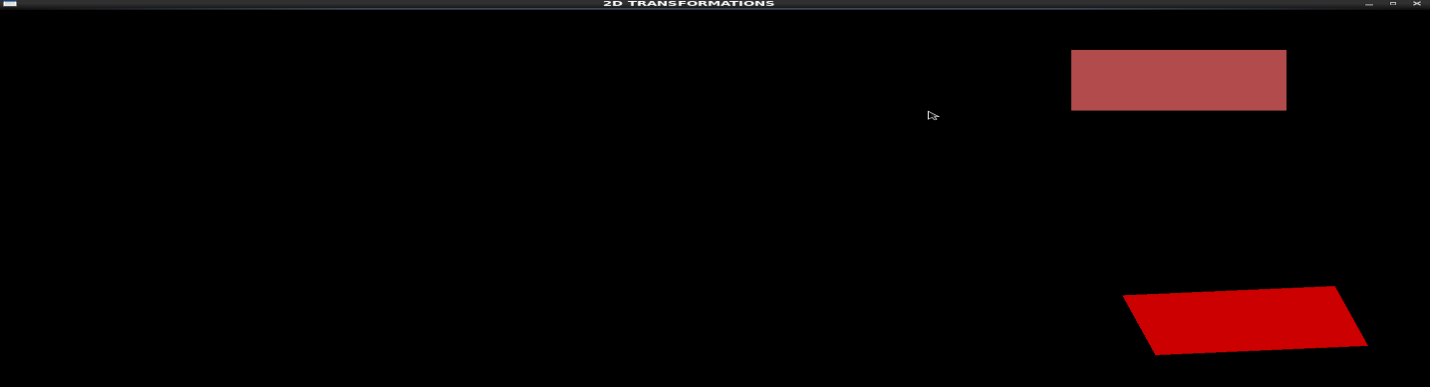
3:SCALING

ENTER UR CHOICE

2

enter an angle

30



[Nidhin@DBCPC12 2Dtransformations]$ ./a.out

2D TRANSFORMATION OPERATIONS ARE

1:TRANSLATION

2:ROTATION

3:SCALING

ENTER UR CHOICE

3

enter values for sx and sy

0.5

0.3



**PROGRAM 9:**

**AIM: Program to perform 3D transformations.**

#include<stdio.h>

#include<math.h>

#include<GL/glut.h>

int ch;

float x1=0.5,x2=0.8,x3=0.8,x4=0.5,y=0.5,y2=0.5,y3=0.8,y4=0.8,z1=0.6,z2=0.4,z3=0.7,z4=0.2;

float X1,X2,X3,X4,Y,Y2,Y3,Y4,Z1,Z2,Z3,Z4;

void display(void)

{

float tx,ty;

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.7,0.3,0.3);

glPointSize(10.0);

glBegin(GL\_POLYGON);

glVertex3f(x1,y,z1);

glVertex3f(x2,y2,z2);

glVertex3f(x3,y3,z3);

glVertex3f(x4,y4,z4);

glEnd();

glColor3f(0.5,0.0,0.0);

glBegin(GL\_POLYGON);

glVertex3f(X1,Y,Z1);

glVertex3f(X2,Y2,Z2);

glVertex3f(X3,Y3,Z3);

glVertex3f(X4,Y4,Z4);

glEnd();

glFlush();

}

void translate()

{

float tx,ty,tz;

printf("Enter values for tx,ty and tz\n");

scanf("%f%f%f",&tx,&ty,&tz);

X1=x1+tx;

X2=x2+tx;

X3=x3+tx;

X4=x4+tx;

Y=y+ty;

Y2=y2+ty;

Y3=y3+ty;

Y4=y4+ty;

Z1=z1+tz;

Z2=z2+tz;

Z3=z3+tz;

Z4=z4+tz;

}

void rotate()

{

int theta;

printf("Enter an angle\n");

scanf("%d",&theta);

X1=x1\*cos(theta)-y\*sin(theta);

X2=x2\*cos(theta)-y2\*sin(theta);

X3=x3\*cos(theta)-y3\*sin(theta);

X4=x4\*cos(theta)-y4\*sin(theta);

Y=x1\*sin(theta)+y\*cos(theta);

Y2=x2\*sin(theta)+y2\*cos(theta);

Y3=x3\*sin(theta)+y3\*cos(theta);

Y4=x4\*sin(theta)+y4\*cos(theta);

Z1=z1\*cos(theta)-z1\*sin(theta);

Z2=z2\*cos(theta)-z2\*sin(theta);

Z3=z3\*cos(theta)-z3\*sin(theta);

Z4=z4\*cos(theta)-z4\*sin(theta);

}

void scale()

{

float sx,sy,sz;

printf("Enter values for sx,sy and sz\n");

scanf("%f%f%f",&sx,&sy,&sz);

X1=x1\*sx;

X2=x2\*sx;

X3=x3\*sx;

X4=x4\*sx;

Y=y\*sy;

Y2=y2\*sy;

Y3=y3\*sy;

Y4=y4\*sy;

Z1=z1\*sz;

Z2=z2\*sz;

Z3=z3\*sz;

Z4=z4\*sz;

}

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

{

printf("3D TRANSFORMATION OPERATIONS\n");

printf("1:TRANSLATION\n");

printf("2:ROTATION\n");

printf("3:SCALING\n");

printf("ENTER UR CHOICE\n");

scanf("%d",&ch);

switch(ch)

{

case 1:

translate();

break;

case 2:

rotate();

break;

case 3:

scale();

break;

}

glutInit(&argc,argv);

glutInitWindowSize(640,480);

glutInitWindowPosition(0,0);

glutCreateWindow("3D TRANSFORMATION\n");

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

**OUTPUT:**

[Nidhin@DBCPC12 3Dtransformations]$ g++ 3d\_transformations.cpp -lglut

[Nidhin@DBCPC12 3Dtransformations]$ ./a.out

3D TRANSFORMATION OPERATIONS

1:TRANSLATION

2:ROTATION

3:SCALING

ENTER UR CHOICE

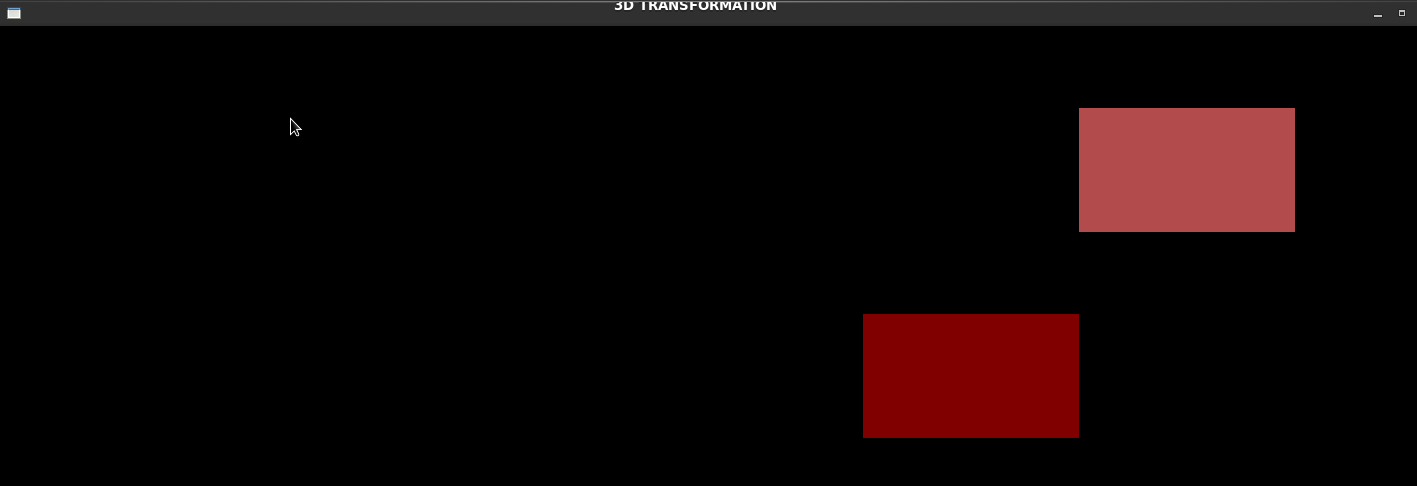
1

Enter values for tx,ty and tz

-0.3

-0.5

-0.7



[Nidhin@DBCPC12 3Dtransformations]$ ./a.out

3D TRANSFORMATION OPERATIONS

1:TRANSLATION

2:ROTATION

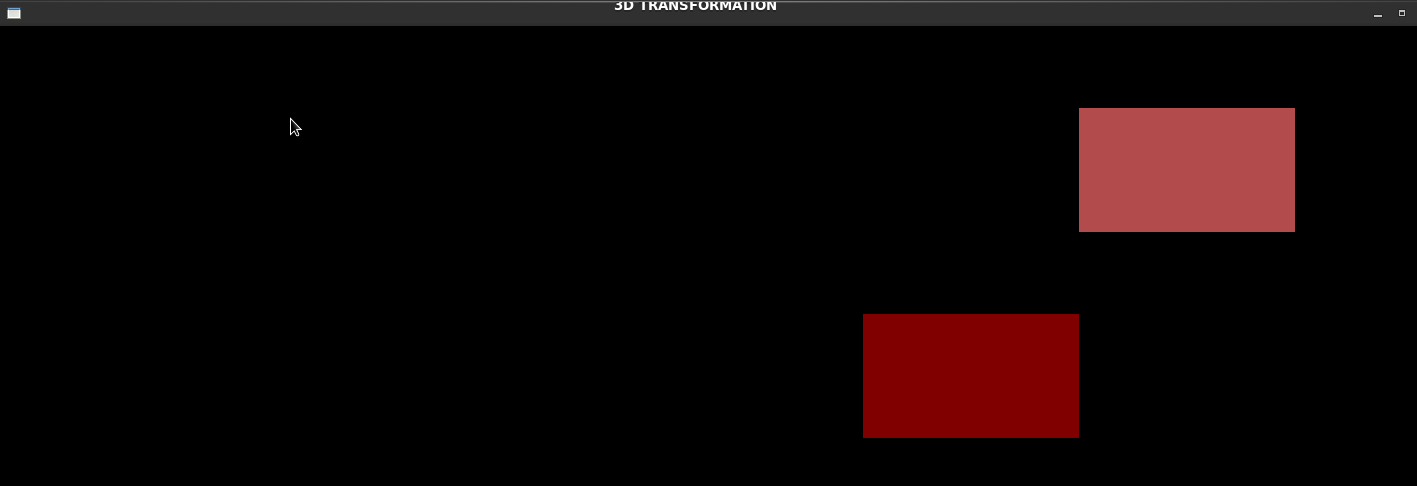
3:SCALING

ENTER UR CHOICE

2

Enter an angle

30



[Nidhin@DBCPC12 3Dtransformations]$ ./a.out

3D TRANSFORMATION OPERATIONS

1:TRANSLATION

2:ROTATION

3:SCALING

ENTER UR CHOICE

3

Enter values for sx,sy and sz

0.3

0.5

0.7

