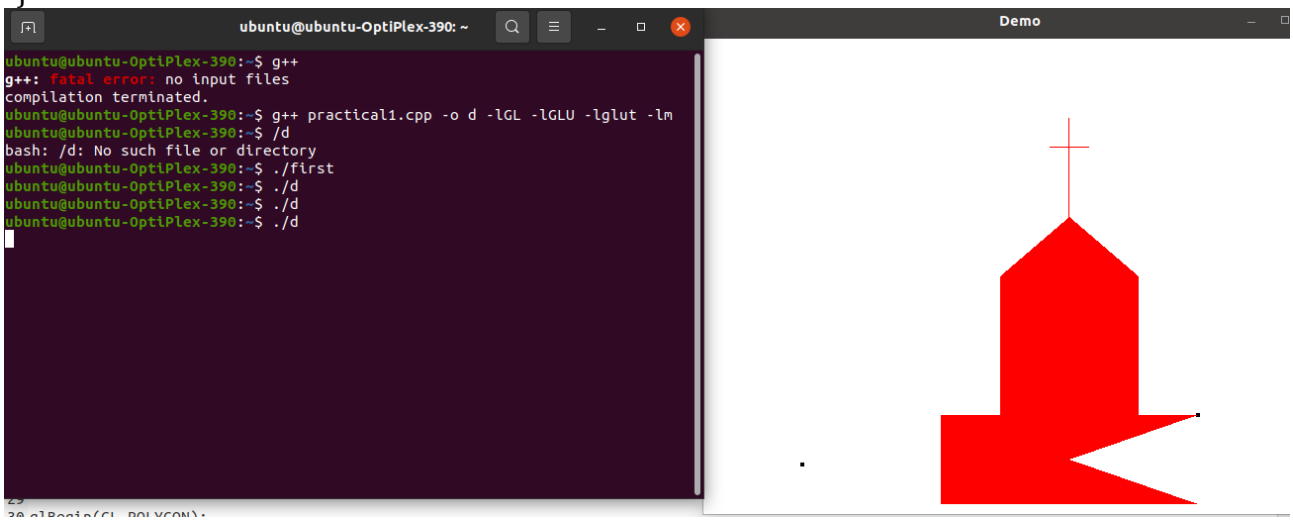
Experiment 1:

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

void init()
{
glClearColor(1,1,1,0);
glColor3f(0.0f,0.0f,1.0f);
glPointSize(12);
gluOrtho2D(0,640,0,800);
}
void pixeldisplay()
{
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_POINTS);
  glVertex2i(600,200);
  glEnd();
 glBegin(GL_POLYGON);
 glColor3f(0.0f,1.0f,0.0f);
 glPolygonMode(GL_FRONT,GL_LINE);
  glVertex2i(100,100);
  glVertex2i(100,700);
  glVertex2i(200,700);
  glVertex2i(200,100);
  glEnd();
  glFlush();
  glBegin(GL_LINES);
glColor3f(1.0f,0.0f,0.0f);
glVertex2i(300,300);
    glVertex2i(500,500);
    glVertex2i(500,500);
    glVertex2i(500,300);
    glVertex2i(300,300);
    glVertex2i(500,300);
    glEnd();

glBegin(GL_TRIANGLES);
  glColor3f(1.0f,0.0f,1.0f);

  glVertex2i(100,100);
    glVertex2i(300,100);
    glVertex2i(200,300);
    //glVertex2i(80,30);
  glEnd();
  glFlush();
  }
int main(int argc,char **argv)
  {
  glutInit(&argc,argv);
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
```

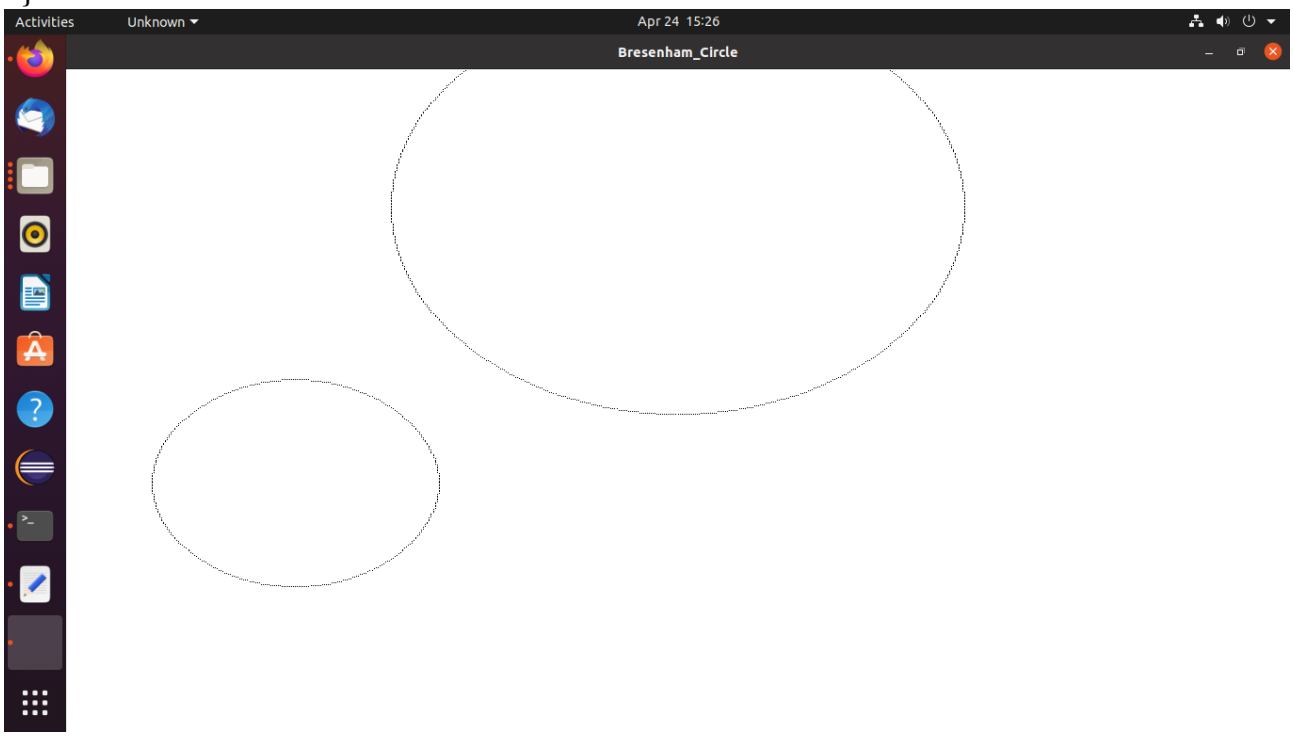
```
 glutInitWindowSize(640,800);
 glutInitWindowPosition(0,0);
 glutCreateWindow("First Program to draw Pixel");
 init();
 glutDisplayFunc(pixeldisplay);
 glutMainLoop();
}
```



Experiment 3:

```
#include<stdio.h>
#include<math.h>
#include<GL/glut.h>
void init()
{
glClearColor(1,1,1,0);
glColor3f(0.0f,0.0f,1.0f);
glPointSize(12);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0,640,0,800);
}
void pixeldisplay()
{
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_POINTS);
  glVertex2i(600,200);
  glEnd();
 glBegin(GL_POLYGON);
 glColor3f(0.0f,1.0f,0.0f);
 glPolygonMode(GL_FRONT,GL_LINE);
  glVertex2i(100,450);
  glVertex2i(200,450);
  glVertex2i(100,500);
  glVertex2i(200,500);
  glVertex2i(200,500);
  glVertex2i(200,450);
  glEnd();
  glFlush();
glBegin(GL_TRIANGLES);
  glColor3f(1.0f,0.0f,1.0f);
  glVertex2i(100,500);
    glVertex2i(200,500);
    glVertex2i(150,600);
    //glVertex2i(80,30);
  glEnd();
  glFlush();
  glBegin(GL_POLYGON);
 glColor3f(1.0f,1.0f,0.0f);
 glPolygonMode(GL_FRONT,GL_LINE);
  glVertex2i(50,350);
  glVertex2i(250,350);
  glVertex2i(50,450);
  glVertex2i(250,450);
  glVertex2i(250,450);
  glVertex2i(250,350);
  glEnd();
  glFlush();
```

```
  }
int main(int argc,char **argv)
  {
  glutInit(&argc,argv);
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
  glutInitWindowSize(640,800);
  glutInitWindowPosition(0,0);
  glutCreateWindow("First Program to draw Pixel");
  init();
  glutDisplayFunc(pixeldisplay);
  glutMainLoop();
  }
```

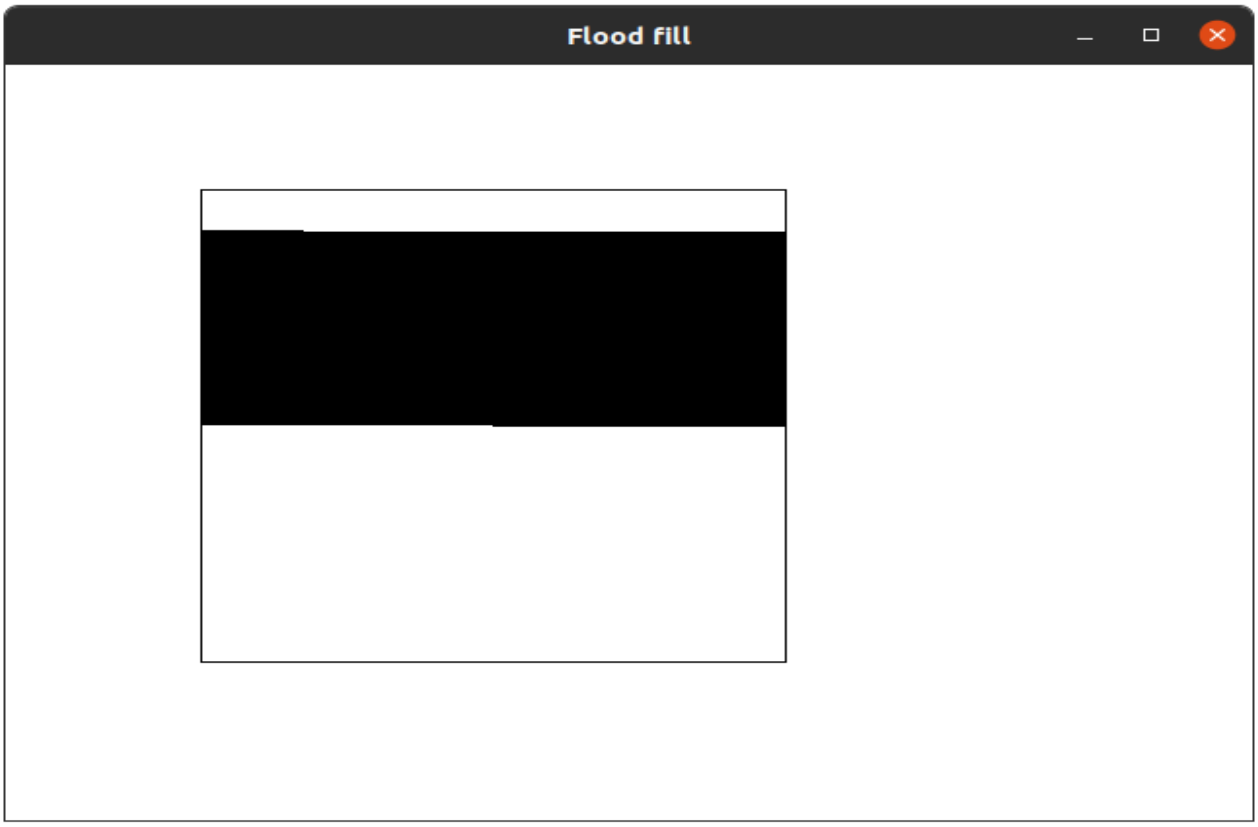


Experiment 4:

```
#include<iostream>
#include<stdio.h>
#include<math.h>
#include<time.h>
#include<GL/glut.h>
using namespace std;
void delay(float ms)
{
clock_t goal=ms+clock();
while(goal>clock());
}
int option;
/*struct Point
{
   GLint x;
   GLint y;
};
struct Color
{
   GLfloat r;
   GLfloat g;
   GLfloat b;
};*/

void init()
{
   glClearColor(1.0,1.0,1.0,0.0);
   glColor3f(1.0,1.0,1.0);
   glPointSize(1.0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(0,640,0,480);
}

/*Color getPixelColor(GLint x, GLint y)
{
   Color color;
   glReadPixels(x,y,1,1, GL_RGB, GL_FLOAT, &color);
   return color;
}

void setPixelColor(GLint x, GLint y, Color color)
{
   glColor3f(color.r, color.g, color.b);
   glBegin(GL_POINTS);
      glVertex2i(x,y);
   glEnd();
```

```
    glFlush();
}
*/
/*void floodFill(int x, int y, Color oldColor, Color newColor)
{
    Color color;
    color=getPixelColor(x,y);
    if(color.r==oldColor.r && color.g==oldColor.g && color.b==oldColor.b)
    {
        setPixelColor(x,y,newColor);
        floodFill(x+1,y,oldColor,newColor);
        floodFill(x,y+1,oldColor,newColor);
        floodFill(x-1,y,oldColor,newColor);
        floodFill(x,y-1,oldColor,newColor);
    }
    return;
}*/
void bound_it(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();
bound_it(x+1,y,fillColor,bc);
bound_it(x-2,y,fillColor,bc);
bound_it(x,y+2,fillColor,bc);
bound_it(x,y-2,fillColor,bc);
}
    return;
}
void onMouseClick(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 color[]={0,0,1};
    if(option==1)
        /*floodFill(x,500-y,oldColor,newColor);*/

        bound_it(x,y,color,bCol);
        }
        }
}
```

```
void display()
{glLineWidth(3);
glPointSize(2);

        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1,0,0);
        glBegin(GL_LINE_LOOP);
    glVertex2i(150,100);
    glVertex2i(300,300);
    glVertex2i(450,100);
    glEnd();
        glFlush();
}
int main(int argc, char**argv)
{
    cout<<"1: floodfill"<<endl;
    cout<<"2: boundaryfill"<<endl;
    cout<<"enter option:";
    cin>>option;
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (640, 480);
    glutInitWindowPosition (200,200);
    glutCreateWindow ("polygon filling");
    init();
    glutDisplayFunc(display);
    glutMouseFunc(onMouseClick);
    glutMainLoop();
    return 0;
}
```


Flood fill

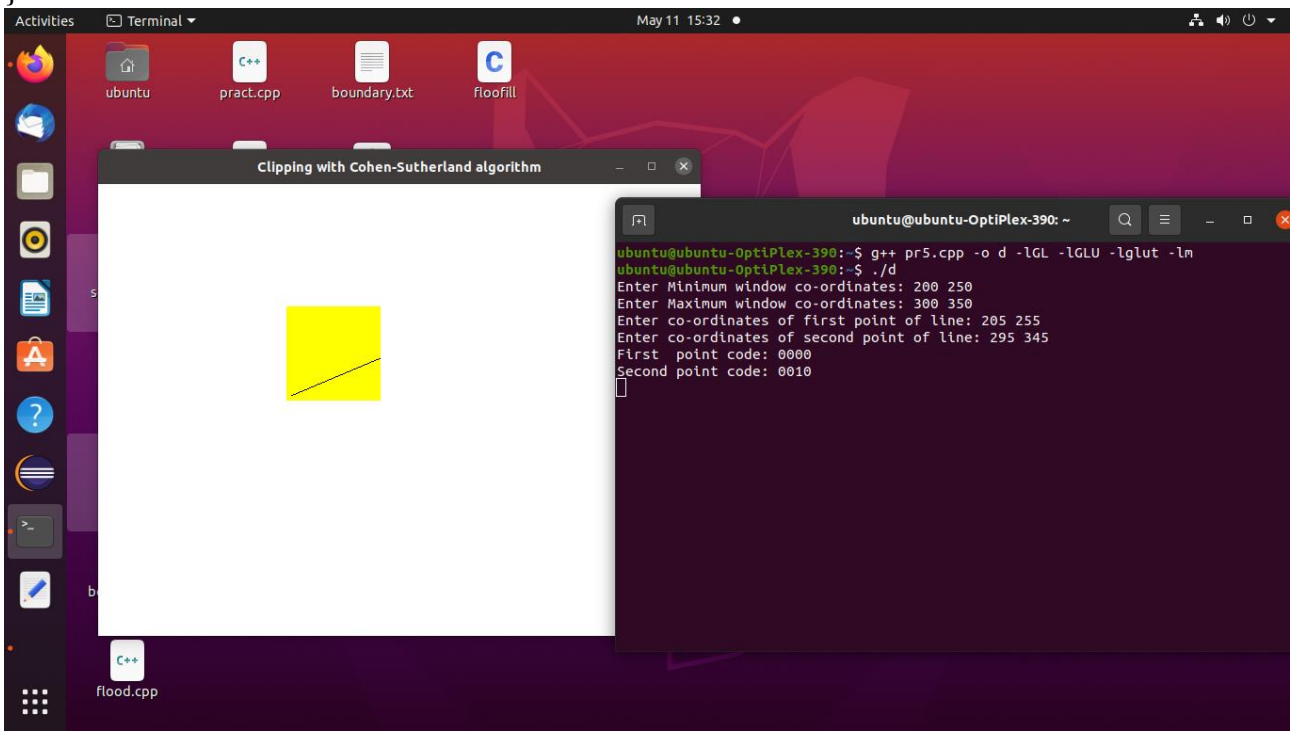
Experiment 5:

```
#include<iostream>
#include<math.h>
#include<GL/glut.h>
#include<stdio.h>
using namespace std;
int minx,miny,maxx,maxy;
int fstx,fsty,sndx,sndy;
int code1[4]={0,0,0,0};
int code2[4]={0,0,0,0};
bool isreject=false;
int getcode1(int x,int y)
{
if(y>maxy)
{
code1[0]=1;
}
if(y<miny)
{
code1[1]=1;
}
if(x<minx)
{
code1[3]=1;
}
if(x>maxx)
{
code1[2]=1;
}
int codeRes1=code1[0]*1000+code1[1]*100+code1[2]*10+code1[3];
return codeRes1;
}
int getcode2(int x,int y)
{
if(y>maxy)
{
code2[0]=1;
}
if(y<miny)
{
code2[1]=1;
}
if(x<minx)
{
code2[3]=1;
}
if(x>maxx)
{
code2[2]=1;
}
int codeRes2=code2[0]*1000+code2[1]*100+code2[2]*10+code2[3];
```

```
return codeRes2;
}
void generateCodeForPoints()
{
getcode1(fstx,fsty);
getcode2(sndx,sndy);
}
void draw_Line(int x0,int y0, int x1,int y1)
{
glBegin(GL_LINES);
glVertex2i(x0,y0);
glVertex2i(x1,y1);
glEnd();
}
void cohenSuth()
{
if(getcode1(fstx,fsty)==0 && getcode2(sndx,sndy)==0 &&isreject==false)
{
draw_Line(fstx,fsty,sndx,sndy);
}
else{
for(int i=0;i<4;i++)
{
if(code1[i]==code2[i] &&code1[i]==1)
{
isreject=true;
break;
}
}
if(isreject)
{
cout<<"both points rejected"<<endl;
}
else
{
for(int i=0;i<4;i++)
{
if(code1[i]==1)
{
switch(i)
{
case 0:fsty=maxy;
break;
case 1:fsty==miny;
break;
case 2:fstx=maxx;
break;
case 3:fstx=minx;
break;
}
}
if(code2[i]==1)
```

```
{
switch(i)
{
case 0:sndy=maxy;
break;
case 1:sndy==miny;
break;
case 2:sndx=maxx;
break;
case 3:sndx=minx;
break;
}
}
}
draw_Line(fstx,fsty,sndx,sndy);
}
}
}
void myInit(void)
{
glClearColor(1.0,1.0,1.0,0);
glColor3f(0.0,0.0,0.0);
glPointSize(1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0 , 640 , 0 , 480);
}
void myDisplay(void)
{
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0,1.0,0.0);
glRecti(minx,miny,maxx,maxy);
glColor3f(0.0,0.0,0.0);
cohenSuth();
glFlush();
}
int main(int argc, char **argv)
{
cout<<"enter maximum window coordinates:";
cin>>maxx>>maxy;
cout<<"enter minimum window coordinates:";
cin>>minx>>miny;
cout<<"enter coordinates of first point of line:";
cin>>fstx>>fsty;
cout<<"enter coordinates of second point of line:";
cin>>sndx>>sndy;
generateCodeForPoints();
cout<<"first point code:"<<code1[0]<<code1[1]<<code1[2]<<code1[3]<<endl;
cout<<"second point code:"<<code2[0]<<code2[1]<<code2[2]<<code2[3]<<endl;
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(640, 480);
```

```
glutInitWindowPosition(100, 150);
glutCreateWindow("line clipping");
glutDisplayFunc(myDisplay);
myInit();
glutMainLoop();
}
```



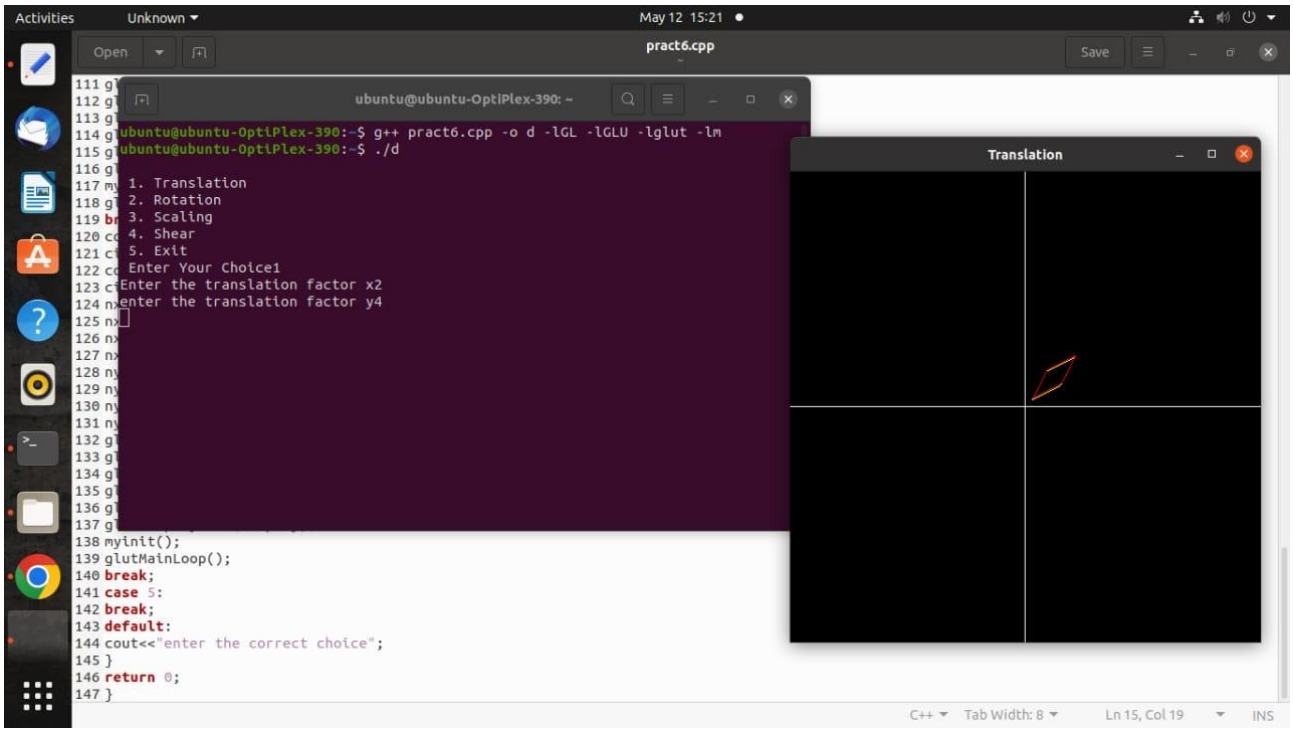
Experiment 6:

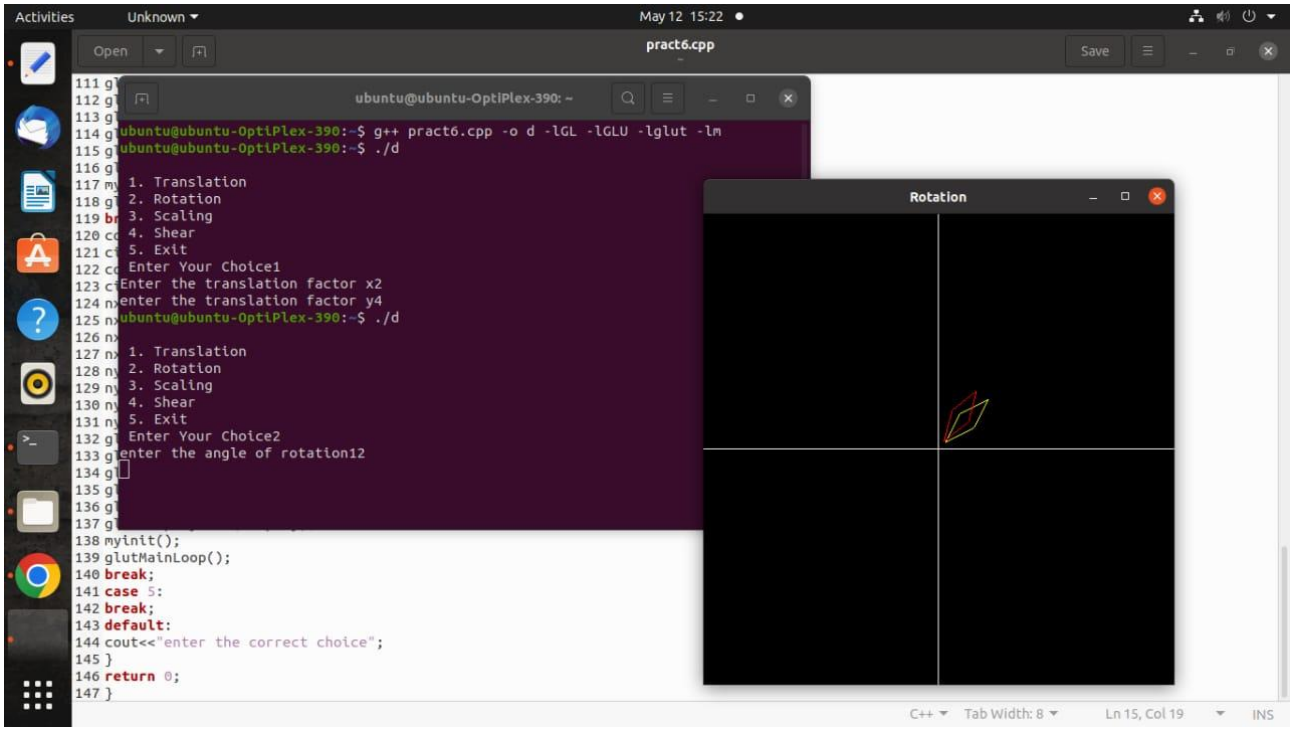
```
#include<iostream>
#include<math.h>
#include<GL/glut.h>
using namespace std;
int choice;
int x1,x2,x3,x4,yy1,y2,y3,y4,nx1,nx2,nx3,nx4,ny1,ny2,ny3,ny4,c,shx,shy;
float sx,sy,xt,yt,r;
double t;
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0,1.0,1.0);
glBegin(GL_LINES);
glVertex2i(-500,0);
glVertex2i(500,0);
glVertex2i(0,-500);
glVertex2i(0,500);
glEnd();
glColor3f(1.0,1.0,0.0);
glBegin(GL_LINE_LOOP);
glVertex2f(x1,yy1);
glVertex2f(x2,y2);
glVertex2f(x3,y3);
glVertex2f(x4,y4);
glEnd();
glColor3f(1.0,0.0,0.0);
glBegin(GL_LINE_LOOP);
glVertex2f(nx1,ny1);
glVertex2f(nx2,ny2);
glVertex2f(nx3,ny3);
glVertex2f(nx4,ny4);
glEnd();
glFlush();
}
void myinit()
{
glClearColor(0.0,0.0,0.0,1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-500.0,500.0,-500.0,500.0);
}
int main(int argc, char **argv)
{
x1=15;
yy1=15;
x2=75;
y2=45;
x3=105;
```

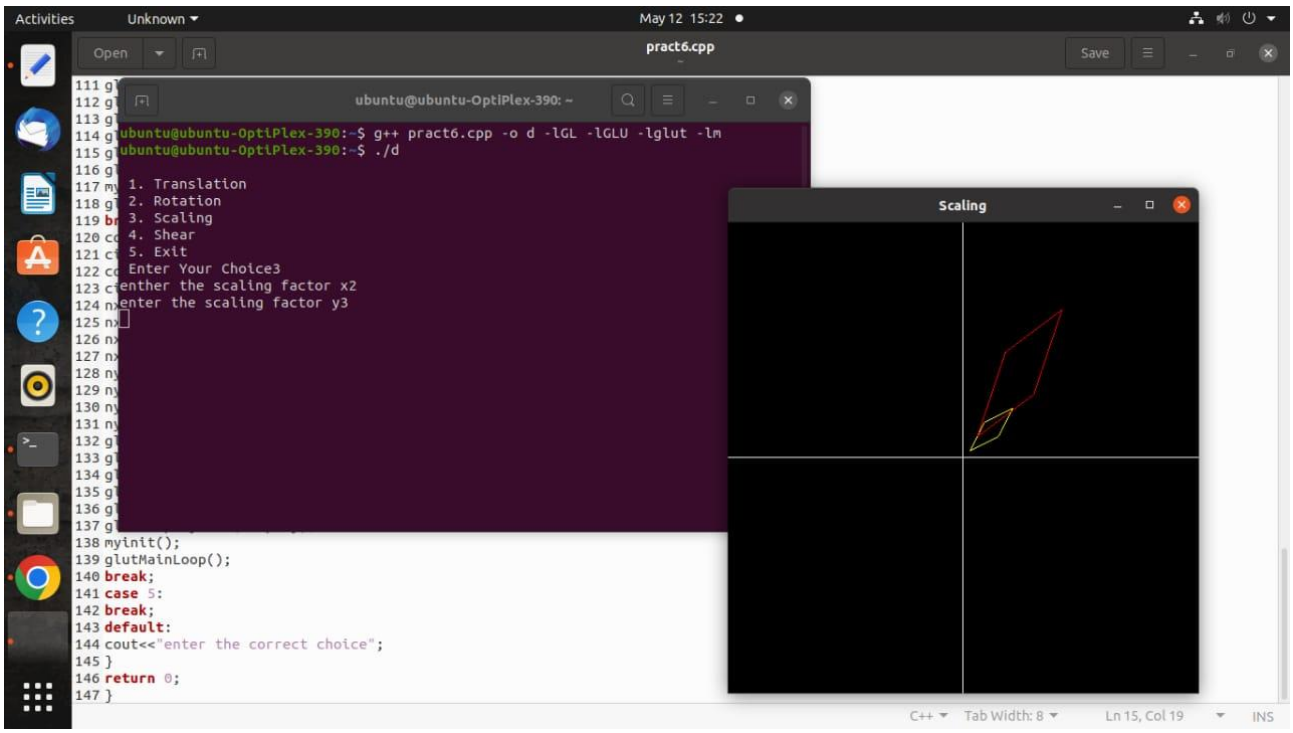
```
y3=105;
x4=45;
y4=75;
cout<<"\n 1. Translation \n 2. Rotation \n 3. Scaling \n 4. Shear \n 5. Exit \n Enter Your Choice";
cin>>c;
switch(c)
{
case 1:
cout<<"Enter the translation factor x";
cin>>xt;
cout<<"enter the translation factor y";
cin>>yt;
nx1=x1+xt;
ny1=yy1+yt;
nx2=x2+xt;
ny2=y2+yt;
nx3=x3+xt;
ny3=y3+yt;
nx4=x4+xt;
ny4=y4+yt;
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
glutCreateWindow("Translation");
glutDisplayFunc(display);
myinit();
glutMainLoop();
break;
case 2:
cout<<"enter the angle of rotation";
cin>>r;
t=3.14*r/180;nx1=(x1*cos(t)-yy1*sin(t));
ny1=(x1*sin(t)+yy1*cos(t));
nx2=(x2*cos(t)-y2*sin(t));
ny2=(x2*sin(t)+y2*cos(t));
nx3=(x3*cos(t)-y3*sin(t));
ny3=(x3*sin(t)+y3*cos(t));
nx4=(x4*cos(t)-y4*sin(t));
ny4=(x4*sin(t)+y4*cos(t));
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
glutCreateWindow("Rotation");
glutDisplayFunc(display);
myinit();
glutMainLoop();
break;
case 3:
cout<<"enther the scaling factor x";
cin>>sx;
```

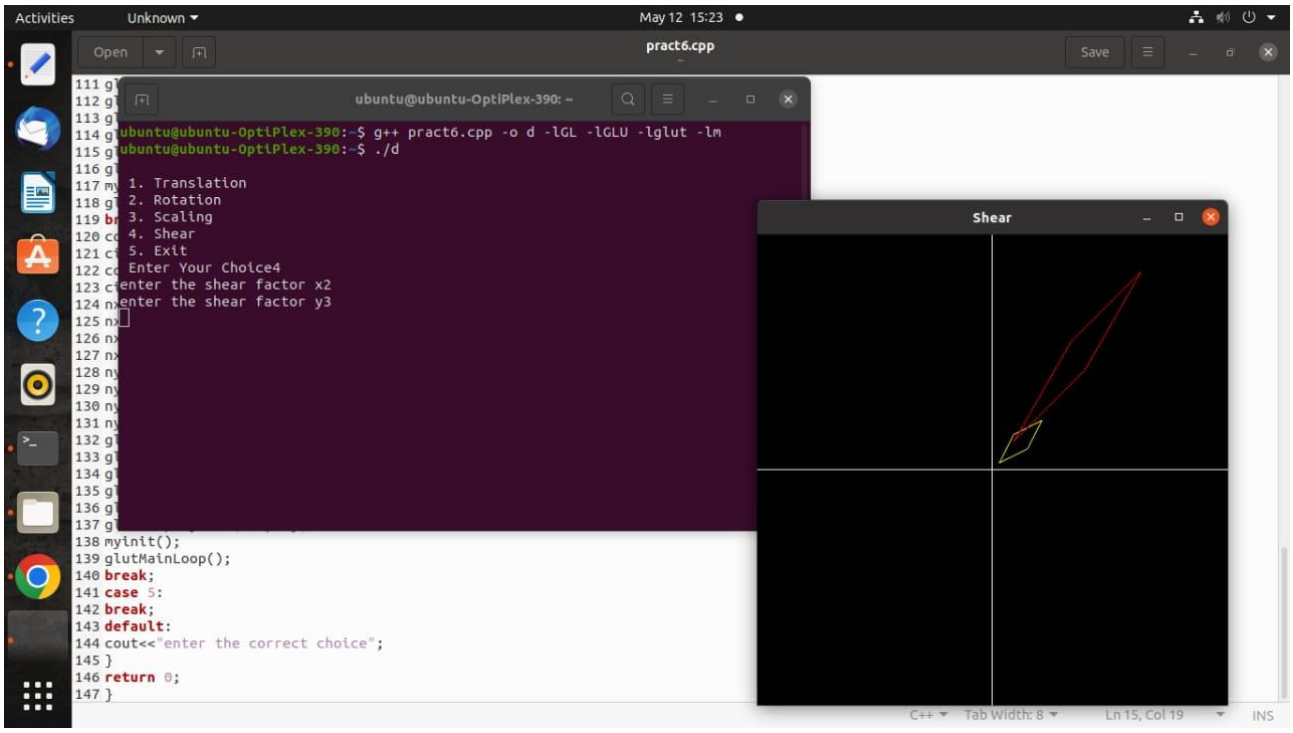
```
cout<<"enter the scaling factor y";
cin>>sy;
nx1=x1*sx;
ny1=yy1*sy;
nx2=x2*sx;
ny2=y2*sy;
nx3=x3*sx;
ny3=y3*sy;
nx4=x4*sx;
ny4=y4*sy;
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
glutCreateWindow("Scaling");
glutDisplayFunc(display);
myinit();
glutMainLoop();
break;case 4:
cout<<"enter the shear factor x";
cin>>shx;
cout<<"enter the shear factor y";
cin>>shy;
nx1=(x1+shx*yy1);
nx2=(x2+shx*y2);
nx3=(x3+shx*y3);
nx4=(x4+shx*y4);
ny1=(yy1+shy*x1);
ny2=(y2+shy*x2);
ny3=(y3+shy*x3);
ny4=(y4+shy*x4);
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(0,0);
glutCreateWindow("Shear");
glutDisplayFunc(display);
myinit();
glutMainLoop();
break;
case 5:
break;
default:
cout<<"enter the correct choice";
}
return 0;
```

}


Activities
Unknown
May 12 15:21
pract6.cpp
Open
Save
ubuntu@ubuntu-OptiPlex-390: ~
ubuntu@ubuntu-OptiPlex-390:~$ g++ pract6.cpp -o d -lGL -lGLU -lglut -lm
ubuntu@ubuntu-OptiPlex-390:~$ ./d
1. Translation
2. Rotation
3. Scaling
4. Shear
5. Exit
Enter Your Choice1
Enter the translation factor x2
enter the translation factor y4
Translation
myinit();
glutMainLoop();
break;
case 5:
break;
default:
cout<<"enter the correct choice";
}
return 0;
}
C++
Tab Width: 8
Ln 15, Col 19
INS

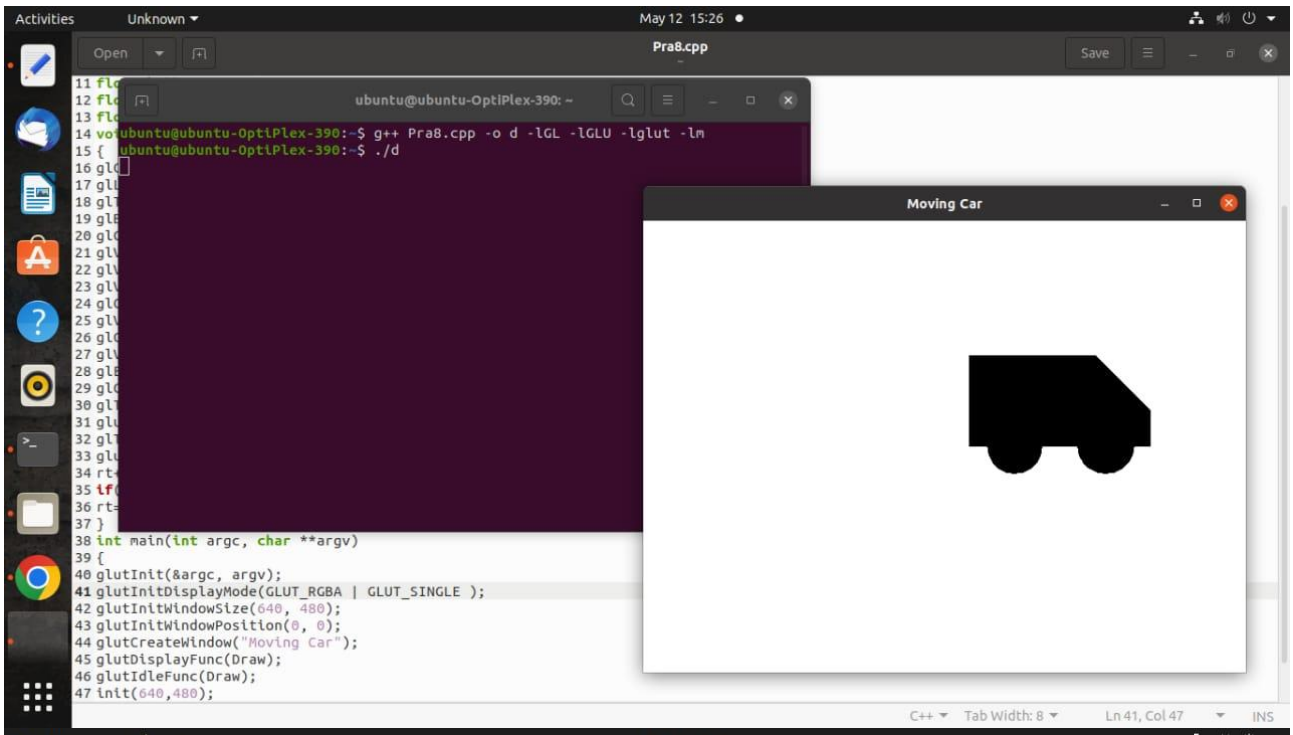

Activities
Unknown
May 12 15:22
pract6.cpp
Open
Save
ubuntu@ubuntu-OptiPlex-390: ~
ubuntu@ubuntu-OptiPlex-390:~$ g++ pract6.cpp -o d -lGL -lGLU -lglut -lm
ubuntu@ubuntu-OptiPlex-390:~$ ./d
1. Translation
2. Rotation
3. Scaling
4. Shear
5. Exit
Enter Your Choice1
Enter the translation factor x2
enter the translation factor y4
ubuntu@ubuntu-OptiPlex-390:~$ ./d
1. Translation
2. Rotation
3. Scaling
4. Shear
5. Exit
Enter Your Choice2
enter the angle of rotation12
Rotation
myinit();
glutMainLoop();
break;
case 5:
break;
default:
cout<<"enter the correct choice";
}
return 0;
}
C++
Tab Width: 8
Ln 15, Col 19
INS

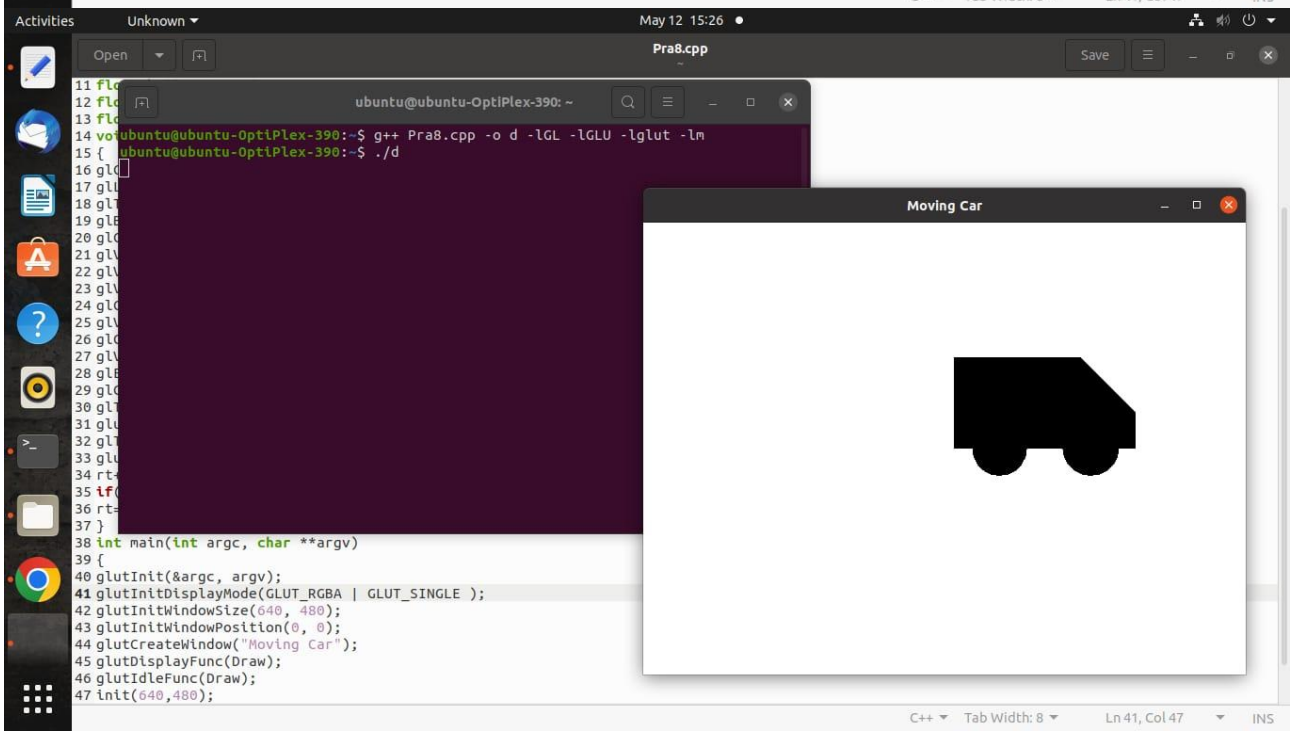

Activities
Unknown
May 12 15:22
Open
pract6.cpp
Save
ubuntu@ubuntu-OptiPlex-390: ~
ubuntu@ubuntu-OptiPlex-390:~$ g++ pract6.cpp -o d -lGL -lGLU -lglut -lm
ubuntu@ubuntu-OptiPlex-390:~$ ./d
1. Translation
2. Rotation
3. Scaling
4. Shear
5. Exit
Enter Your Choice3
enther the scaling factor x2
enter the scaling factor y3
Scaling
myinit();
glutMainLoop();
break;
case 5:
break;
default:
cout<<"enter the correct choice";
}
return 0;
}
C++
Tab Width: 8
Ln 15, Col 19
INS


Activities
Unknown
May 12 15:23
Open
pract6.cpp
Save
ubuntu@ubuntu-OptiPlex-390: ~
ubuntu@ubuntu-OptiPlex-390:~$ g++ pract6.cpp -o d -lGL -lGLU -lglut -lm
ubuntu@ubuntu-OptiPlex-390:~$ ./d
1. Translation
2. Rotation
3. Scaling
4. Shear
5. Exit
Enter Your Choice4
enter the shear factor x2
enter the shear factor y3
Shear
myinit();
glutMainLoop();
break;
case 5:
break;
default:
cout<<"enter the correct choice";
}
return 0;
}
C++
Tab Width: 8
Ln 15, Col 19
INS

Experiment 8(car):

```
#include<iostream>
#include <GL/glut.h>
float rt = 0.0f;
void init(int Width, int Height)
{
glClearColor(1.1, 1.1, 1.1, 1.1);
glMatrixMode(GL_PROJECTION);
gluPerspective(45.0f,(GLfloat)Width/(GLfloat)Height,0.1f,50.0f);
glMatrixMode(GL_MODELVIEW);
}
float ballX = -0.5f;
float ballY = 0.0f;
float ballZ = 0.0f;
void Draw()
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
glTranslatef(rt,0.0f,-6.0f);
glBegin(GL_POLYGON);
glColor3f(0.0,0.0,0.0);
glVertex3f(-1.0f, 1.0f, 0.0f);
glVertex3f(0.4f, 1.0f, 0.0f);
glVertex3f(1.0f, 0.4f, 0.0f);
glColor3f(0.0,0.0,0.0);
glVertex3f( 1.0f,0.0f, 0.0f);
glColor3f(0.0,0.0,0.0);
glVertex3f(-1.0f,0.0f, 0.0f);
glEnd();
glColor3f(0.0, 0.0, 0.0);
glTranslatef(ballX,ballY,ballZ);
glutSolidSphere (0.3, 20, 20);
glTranslatef(ballX+1.5,ballY,ballZ);
glutSolidSphere (0.3, 20, 20);
rt+=0.005f;
if(rt>2)
rt=-2.0f;glutSwapBuffers();
}
int main(int argc, char **argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_RGBA | GLUT_SINGLE );
glutInitWindowSize(640, 480);
glutInitWindowPosition(0, 0);
glutCreateWindow("Moving Car");
glutDisplayFunc(Draw);
glutIdleFunc(Draw);
init(640,480);
glutMainLoop();
```


Activities
Unknown
May 12 15:26
Open
Pra8.cpp
Save
ubuntu@ubuntu-OptiPlex-390: ~
ubuntu@ubuntu-OptiPlex-390:~$ g++ Pra8.cpp -o d -lGL -lGLU -lglut -lm
ubuntu@ubuntu-OptiPlex-390:~$ ./d
Moving Car
int main(int argc, char **argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_RGBA | GLUT_SINGLE);
glutInitWindowSize(640, 480);
glutInitWindowPosition(0, 0);
glutCreateWindow("Moving Car");
glutDisplayFunc(Draw);
glutIdleFunc(Draw);
init(640,480);
C++
Tab Width: 8
Ln 41, Col 47
INS


Activities
Unknown
May 12 15:26
Open
Pra8.cpp
Save
ubuntu@ubuntu-OptiPlex-390: ~
ubuntu@ubuntu-OptiPlex-390:~$ g++ Pra8.cpp -o d -lGL -lGLU -lglut -lm
ubuntu@ubuntu-OptiPlex-390:~$ ./d
Moving Car
int main(int argc, char **argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_RGBA | GLUT_SINGLE);
glutInitWindowSize(640, 480);
glutInitWindowPosition(0, 0);
glutCreateWindow("Moving Car");
glutDisplayFunc(Draw);
glutIdleFunc(Draw);
init(640,480);
C++
Tab Width: 8
Ln 41, Col 47
INS

```
return 0;
}
```

Experiment 5: Bezeir Curve

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <stdlib.h>

int i;

// Function to initialize the Beizer
// curve pointer
void init(void)
{
	glClearColor(1.0, 1.0, 1.0, 1.0);
}

// Function to draw the Bitmap Text
void drawBitmapText(char* string, float x,
                                        float y, float z)
{
	char* c;
	glRasterPos2f(x, y);

	// Traverse the string
	for (c = string; *c != '\0'; c++) {
		glutBitmapCharacter(
			GLUT_BITMAP_TIMES_ROMAN_24, *c);
	}
}

// Function to draw the shapes
void draw(GLfloat ctrlpoints[4][3])
{
```

```
	glShadeModel(GL_FLAT);
	glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 4,
			&ctrlpoints[0][0]);

	glEnable(GL_MAP1_VERTEX_3);

	// Fill the color
	glColor3f(1.0, 1.0, 1.0);
	glBegin(GL_LINE_STRIP);

	// Find the coordinates
	for (i = 0; i <= 30; i++)
		glEvalCoord1f((GLfloat)i / 30.0);

	glEnd();
	glFlush();
}

// Function to display the curved
// drawn using the Beizer Curve
void display(void)
{
	int i;

	// Specifying all the control
	// points through which the
	// curve will pass
	GLfloat ctrlpoints[4][3]
		= { { -0.00, 2.00, 0.0 },
			{ -2.00, 2.00, 0.0 },
			{ -2.00, -1.00, 0.0 },
			{ -0.00, -1.00, 0.0 } };
```

```
draw(ctrlpoints);

GLfloat ctrlpoints2[4][3]
        = { { 0.0, -1.00, 0.0 },
                { 0.55, -0.65, 0.0 },
                { 0.65, -0.25, 0.0 },
                { 0.00, 0.70, 0.0 } };

draw(ctrlpoints2);
GLfloat ctrlpoints3[4][3]
        = { { 0.0, 0.70, 0.0 },
                { 0.15, 0.70, 0.0 },
                { 0.25, 0.70, 0.0 },
                { 0.65, 0.700, 0.0 } };

draw(ctrlpoints3);
GLfloat ctrlpoints4[4][3]
        = { { 0.65, 0.70, 0.0 },
                { 0.65, -0.90, 0.0 },
                { 0.65, -0.70, 0.0 },
                { 0.65, -1.00, 0.0 } };

draw(ctrlpoints4);

GLfloat ctrlpoints5[4][3]
        = { { 1.00, -1.00, 0.0 },
                { 1.00, -0.5, 0.0 },
                { 1.00, -0.20, 0.0 },
                { 1.00, 1.35, 0.0 } };

draw(ctrlpoints5);
GLfloat ctrlpoints6[4][3]
```

```
        = { { 1.00, 1.35 },
                { 1.10, 1.35, 0.0 },
                { 1.10, 1.35, 0.0 },
                { 1.90, 1.35, 0.0 } };

draw(ctrlpoints6);
GLfloat ctrlpoints7[4][3]
        = { { 1.00, 0.50, 0.0 },
                { 1.10, 0.5, 0.0 },
                { 1.10, 0.5, 0.0 },
                { 1.90, 0.5, 0.0 } };

draw(ctrlpoints7);
GLfloat ctrlpoints8[4][3]
        = { { 3.50, 2.00, 0.0 },
                { 1.50, 2.00, 0.0 },
                { 1.50, -1.00, 0.0 },
                { 3.50, -1.00, 0.0 } };
draw(ctrlpoints8);

GLfloat ctrlpoints9[4][3]
        = { { 3.50, -1.00, 0.0 },
                { 4.05, -0.65, 0.0 },
                { 4.15, -0.25, 0.0 },
                { 3.50, 0.70, 0.0 } };
draw(ctrlpoints9);

GLfloat ctrlpoints10[4][3]
        = { { 3.50, 0.70, 0.0 },
                { 3.65, 0.70, 0.0 },
                { 3.75, 0.70, 0.0 },
                { 4.15, 0.700, 0.0 } };
```

```
draw(ctrlpoints10);
GLfloat ctrlpoints11[4][3]
        = { { 4.15, 0.70, 0.0 },
                { 4.15, -0.90, 0.0 },
                { 4.15, -0.70, 0.0 },
                { 4.15, -1.00, 0.0 } };

draw(ctrlpoints11);

GLfloat ctrlpoints12[4][3]
        = { { -2.0, 2.50, 0.0 },
                { 2.05, 2.50, 0.0 },
                { 3.15, 2.50, 0.0 },
                { 4.65, 2.50, 0.0 } };

draw(ctrlpoints12);

GLfloat ctrlpoints13[4][3]
        = { { -2.0, -1.80, 0.0 },
                { 2.05, -1.80, 0.0 },
                { 3.15, -1.80, 0.0 },
                { 4.65, -1.80, 0.0 } };

draw(ctrlpoints13);

GLfloat ctrlpoints14[4][3]
        = { { -2.0, -1.80, 0.0 },
                { -2.0, 1.80, 0.0 },
                { -2.0, 1.90, 0.0 },
                { -2.0, 2.50, 0.0 } };
```

```
	draw(ctrlpoints14);
	GLfloat ctrlpoints15[4][3]
		= { { 4.650, -1.80, 0.0 },
			{ 4.65, 1.80, 0.0 },
			{ 4.65, 1.90, 0.0 },
			{ 4.65, 2.50, 0.0 } };

	draw(ctrlpoints15);

	// Specifying the colour of
	// text to be displayed
	glColor3f(1, 0, 0);
	drawBitmapText("Bezier Curves "
				"Implementation",
				-1.00, -3.0, 0);
	glFlush();
}

// Function perform the reshaping
// of the curve
void reshape(int w, int h)
{
	glViewport(0, 0, (GLsizei)w,
			(GLsizei)h);

	// Matrix mode
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();

	if (w <= h)
		glOrtho(-5.0, 5.0, -5.0
								* (GLfloat)h / (GLfloat)w,
```

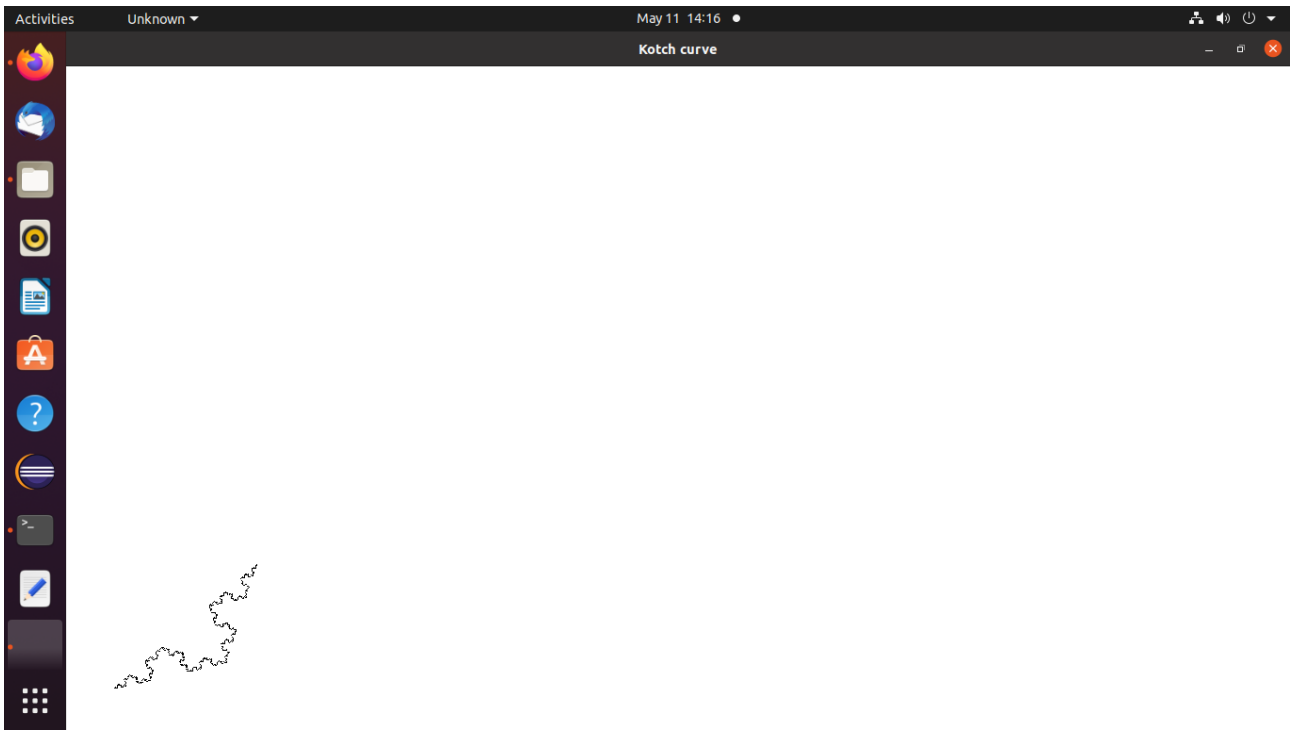
```
                        5.0 * (GLfloat)h / (GLfloat)w, -5.0, 5.0);
        else
                glOrtho(-5.0 * (GLfloat)w / (GLfloat)h,
                        5.0 * (GLfloat)w / (GLfloat)h,
                        -5.0, 5.0,
                        -5.0, 5.0);

        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
}

// Driver Code
int main(int argc, char** argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(
                GLUT_SINGLE | GLUT_RGB);

        // Specifies the window size
        glutInitWindowSize(500, 500);
        glutInitWindowPosition(100, 100);

        // Creates the window as
        // specified by the user
        glutCreateWindow(argv[0]);
        init();

        // Links display event with the
        // display event handler(display)
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
```

```
    // Loops the current event
    glutMainLoop();

    return 0;
}
```

Experiment 5: Koch curve

```
#include<iostream>
#include<GL/glut.h>
#include<stdio.h>
//#include<math.h>
using namespace std;
float x1,x2,y1,y2,n;
void getdata()
{
cout<<"enter start & end points of line";
cin>>x1>>y1>>x2>>y2;
cout<<"enter nos interation";
cin>>n;
}

void koch(float x1,float y1,float x2,float y2,float n)
{
float ang=60;ang=ang*3.14/180;
float x3=(2*x1+x2)/3;
float y3=(2*y1+y2)/3;
float x4=(x1+2*x2)/3;
float y4=(y1+2*y2)/3;
float x=x3+(x4-x3)*0.5+(y4-y3)*0.8660;
float y=y3-(x4-x3)*0.8660+(y4-y3)*0.5;
if(n>0)
{
koch(x1,y1,x3,y3,n-1);
koch(x3,y3,x,y,n-1);
koch(x,y,x4,y4,n-1);
koch(x4,y4,x2,y2,n-1);
}
else
{
glBegin(GL_LINE_STRIP);
glClearColor(1.0,1.0,1.0,0.0);
glColor3f(0.0,1.0,1.0);
glVertex2f(x1,y1);
glColor3f(0.0,1.0,1.0);
glVertex2f(x3,y3);
glColor3f(1.0,1.0,0.0);
glVertex2f(x,y);
glColor3f(1.0,0.0,1.0);
glVertex2f(x4,y4);
glColor3f(1.0,1.0,1.0);
glVertex2f(x2,y2);
glEnd();

}
}
void Init()
```

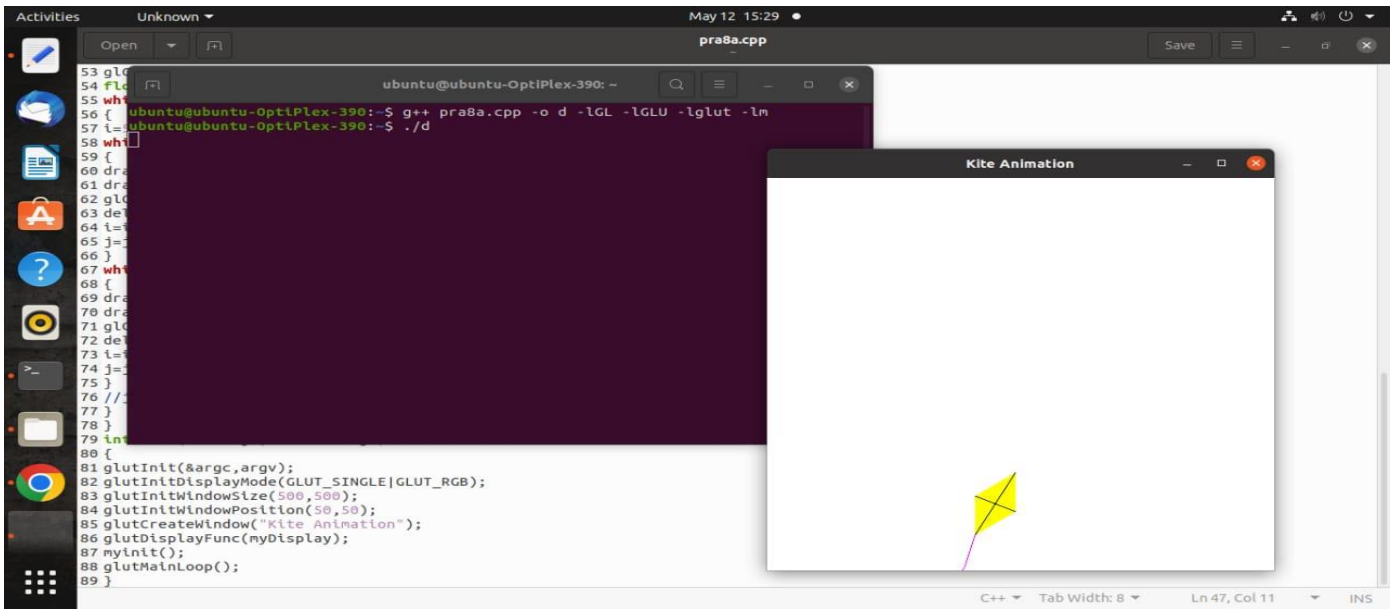
```
{
glClearColor(0.0,0.0,0.0,0.0);
glColor3f(0.0,0.0,0.0);
gluOrtho2D(0.0,640.0,480.0,0.0);
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
koch(x1,y1,x2,y2,n);
glFlush();
}
int main(int argv,char **argc)
{

getdata();

glutInit(&argv,argc);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowPosition(100,100);
glutInitWindowSize(640,480);
glutCreateWindow("KOCH");
Init();
glutDisplayFunc(display);
glutMainLoop();

return 0;
}
```

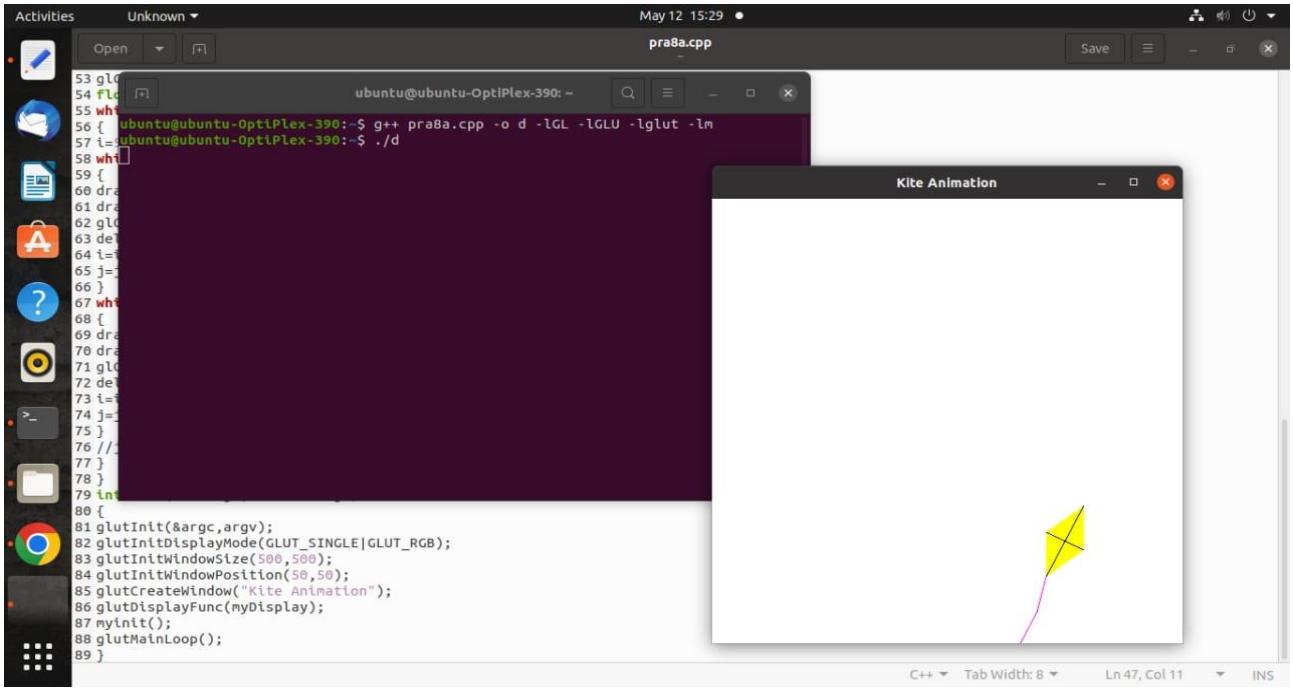


Experiment 8(kite):

```
 #include<iostream>
#include <GL/glut.h>
#include<math.h>
GLsizei wh=500,ww=500;
void myinit()
{
glClearColor(1.0,1.0,1.0,0.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0,(GLdouble)ww,0.0,(GLdouble)wh);
glMatrixMode(GL_MODELVIEW);
//glPointSize(4);
}
void drawkite(float x,float y)
{
glColor3f(1.0,1.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(x-20,y+10);
glVertex2f(x-20,y-40);
glVertex2f(x+20,y-10);
glVertex2f(x+20,y+40);
glEnd();
glColor3f(0.0,0.0,0.0);
glBegin(GL_LINES);
glVertex2f(x-20,y+10);
glVertex2f(x+20,y-10);
glVertex2f(x-20,y-40);
glVertex2f(x+20,y+40);
glEnd();
glFlush();
}
void drawstring(float x,float y)
{
glColor3f(1.0,0.0,1.0);
glBegin(GL_LINES);
glVertex2f(x-20,y-40);
glVertex2f(x-30,y-80);
glVertex2f(x-30,y-80);
glVertex2f(x-50,y-120);
glVertex2f(x-50,y-120);
glVertex2f(x-80,y-150);
glEnd();
glFlush();
}
void delay()
{
int i,j,r;
```

```
for(i=0;i<1000;i++)
for(j=0;j<60000;j++)
r=i*j*10;}
void myDisplay()
{
glClear(GL_COLOR_BUFFER_BIT);
float i=50.0,j=50.0;
while(j<=450.0)
{
i=50.0;
while(i<400.0)
{
drawkite(i,j);
drawstring(i,j);
glClear(GL_COLOR_BUFFER_BIT);
delay();
i=i+5.0;
j=j+1.0;
}
while(i>100.0)
{
drawkite(i,j);
drawstring(i,j);
glClear(GL_COLOR_BUFFER_BIT);
delay();
i=i-5.0;
j=j+1.0;
}
//j=j+30.0;
}
}
int main(int argc, char** argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(50,50);
glutCreateWindow("Kite Animation");
glutDisplayFunc(myDisplay);
myinit();
glutMainLoop();
}
```


Activities
Unknown
May 12 15:29
Open
pra8a.cpp
Save
ubuntu@ubuntu-OptiPlex-390: ~
ubuntu@ubuntu-OptiPlex-390:~$ g++ pra8a.cpp -o d -lGL -lGLU -lglut -lm
ubuntu@ubuntu-OptiPlex-390:~$ ./d
Kite Animation
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(50,50);
glutCreateWindow("Kite Animation");
glutDisplayFunc(myDisplay);
myinit();
glutMainLoop();
C++
Tab Width: 8
Ln 47, Col 11
INS


Activities
Unknown
May 12 15:29
Open
pra8a.cpp
Save
ubuntu@ubuntu-OptiPlex-390: ~
ubuntu@ubuntu-OptiPlex-390:~$ g++ pra8a.cpp -o d -lGL -lGLU -lglut -lm
ubuntu@ubuntu-OptiPlex-390:~$ ./d
Kite Animation
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500);
glutInitWindowPosition(50,50);
glutCreateWindow("Kite Animation");
glutDisplayFunc(myDisplay);
myinit();
glutMainLoop();
C++
Tab Width: 8
Ln 47, Col 11
INS

Experiment 2:

```
#include<GL/glut.h>
#include<stdlib.h>
#include<stdio.h>
float x1,x2,y1,y2;

void display(void)
{
float dy,dx,step,x,y,k,Xin,Yin;
dx=x2-x1;
dy=y2-y1;

if(abs(dx)> abs(dy))
{
step = abs(dx);
}
else
step = abs(dy);

Xin = dx/step;
Yin = dy/step;

x= x1;
y=y1;
glBegin(GL_POINTS);
glVertex2i(x,y);
glEnd();

for (k=1 ;k<=step;k++)
{
x= x + Xin;
y= y + Yin;

glBegin(GL_POINTS);
glVertex2i(x,y);
glEnd();
}


glFlush();
}

void init(void)
{
glClearColor(0.7,0.7,0.7,0.7);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-100,100,-100,100);
}

int main(int argc, char** argv) {
```

```
printf("Enter the value of x1 : ");
scanf("%f",&x1);
printf("Enter the value of y1 : ");
scanf("%f",&y1);
printf("Enter the value of x2 : ");
scanf("%f",&x2);
printf("Enter the value of y2 : ");
scanf("%f",&y2);

glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (500, 500);
glutInitWindowPosition (100,100);
glutCreateWindow ("DDA Line Algo");
init();
glutDisplayFunc(display);
glutMainLoop();

return 0;
}
```