**COMPUTER GRAPHICS FINAL LAB FILE**

**CSE DEVOPS SEM-V (2018-22)**

**Submitted by: Akansha Bansiya**

**SAP: 500067027**

**Rno: R171218009**

**1. GREEN TRIANGLE**

#include<GL/glut.h>

#include<GL/gl.h>

void triangle(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_TRIANGLES);

glColor3f(0,1,0);

glVertex2f(0,1);

glVertex2f(-1,0);

glVertex2f(1,0);

glEnd();

glFlush();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

glutCreateWindow("Triangle");

//init();

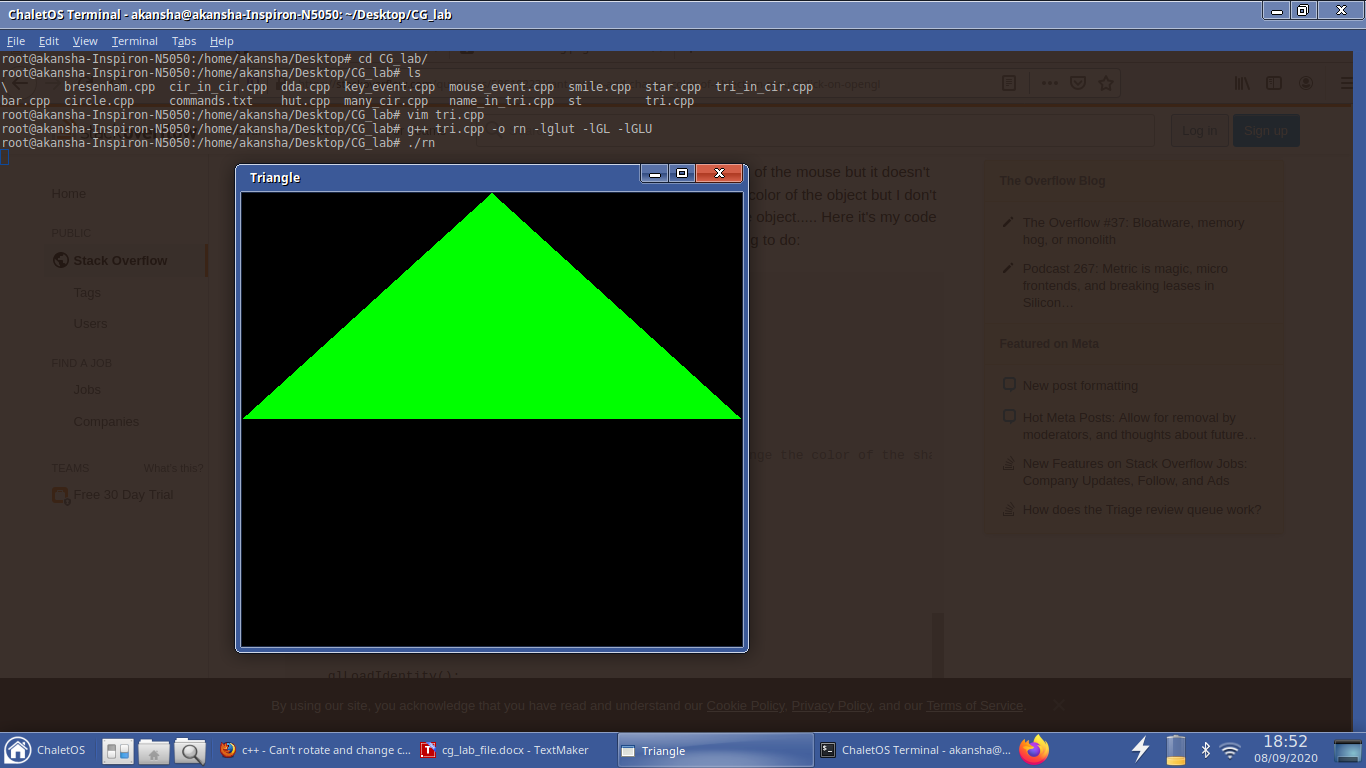
//drawstring();

glutDisplayFunc(triangle);

glutMainLoop();

return 0;

}



**2. DISPLAY NAME IN TRIANGLE**

#include<GL/glut.h>

#include<GL/gl.h>

void drawstring()

{

char d[7]={'A','k','a','n','s','h','a'};

char \*s=d;

char \*c;

glRasterPos3f(-0.1,0.4,0);

for(c=s;\*c!='\0';c++)

{ glColor3f(1,0,0);

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15,\*c);

}

}

void triangle(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_LINES);

glColor3f(0,1,0);

glVertex2f(0,1);

glVertex2f(-1,0);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,0);

glVertex2f(-1,0);

glVertex2f(1,0);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,0);

glVertex2f(1,0);

glVertex2f(0,1);

glEnd();

drawstring();

glFlush();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

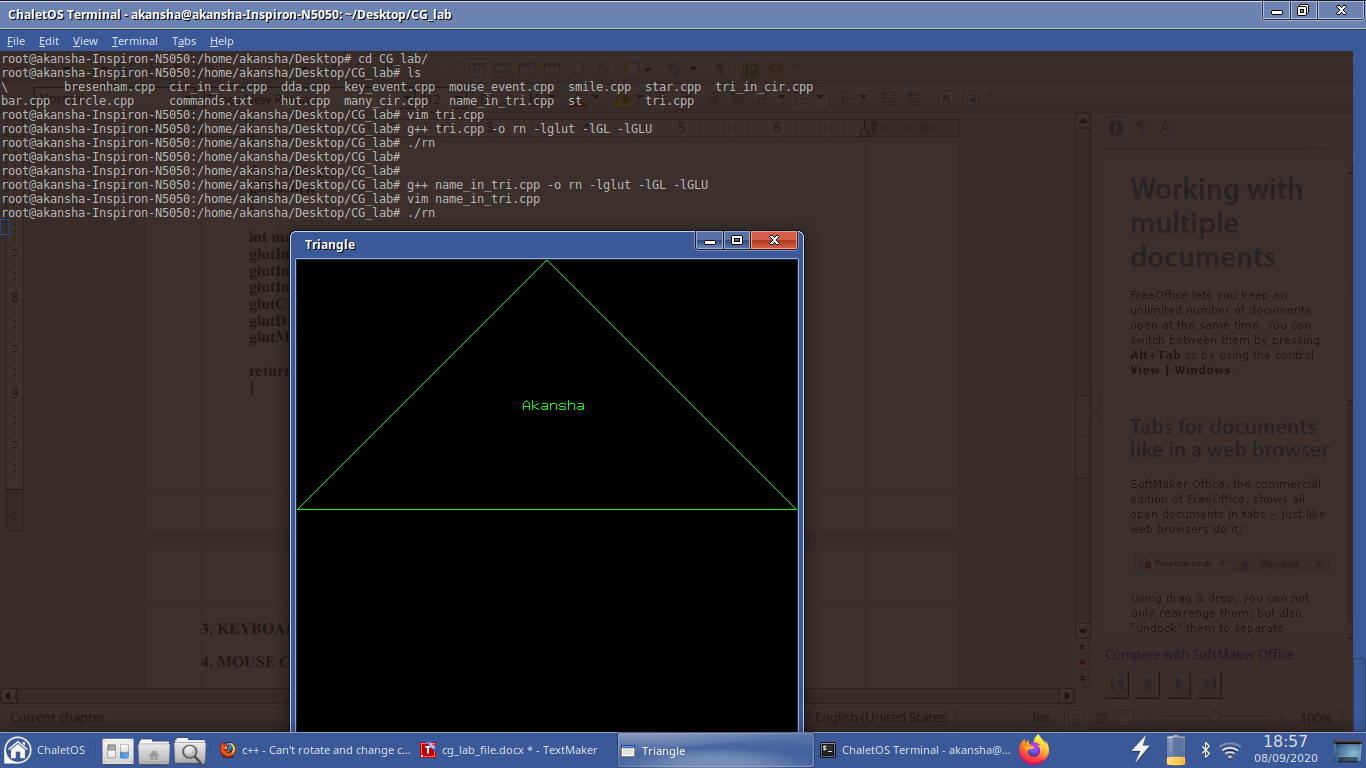
glutCreateWindow("Triangle");

glutDisplayFunc(triangle);

glutMainLoop();

return 0;

}



**3. KEYBOARD KEY PRESS EVENT**

#include<GL/glut.h>

#include<GL/gl.h>

int r=1,g=1,b=1;

void ch\_color(unsigned char key,int x,int y)

{

switch (key) {

case 'b':

r = 0;

g = 0;

b = 1;

break;

case 'r':

r = 1;

g = 0;

b = 0;

break;

case 'g':

r = 0;

g = 1;

b = 0;

break;

}

glutPostRedisplay();

}

void triangle(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_TRIANGLES);

glColor3f(r,g,b);

glVertex2f(0,1);

glVertex2f(-1,0);

glVertex2f(1,0);

glEnd();

glFlush();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

glutCreateWindow("Triangle");

glutDisplayFunc(triangle);

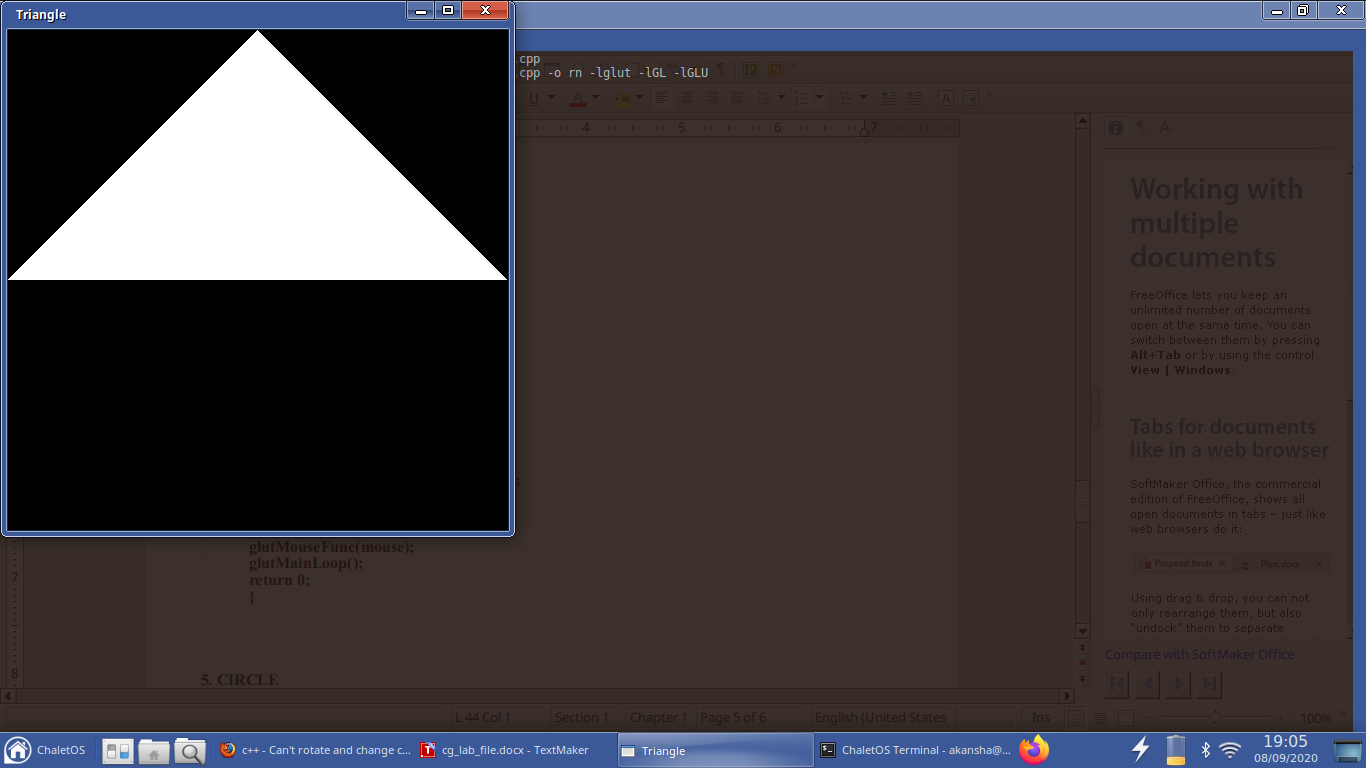
glutKeyboardFunc(ch\_color);

glutMainLoop();

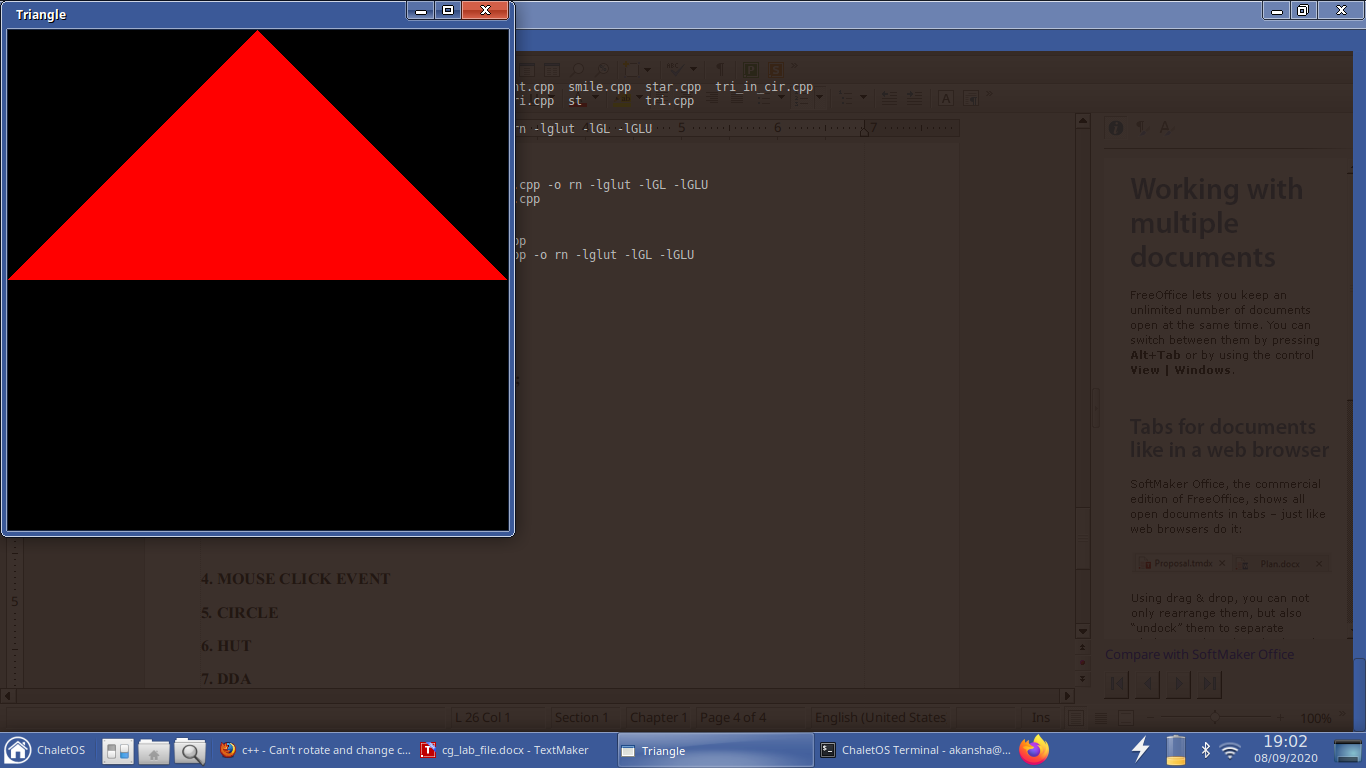
return 0;

}

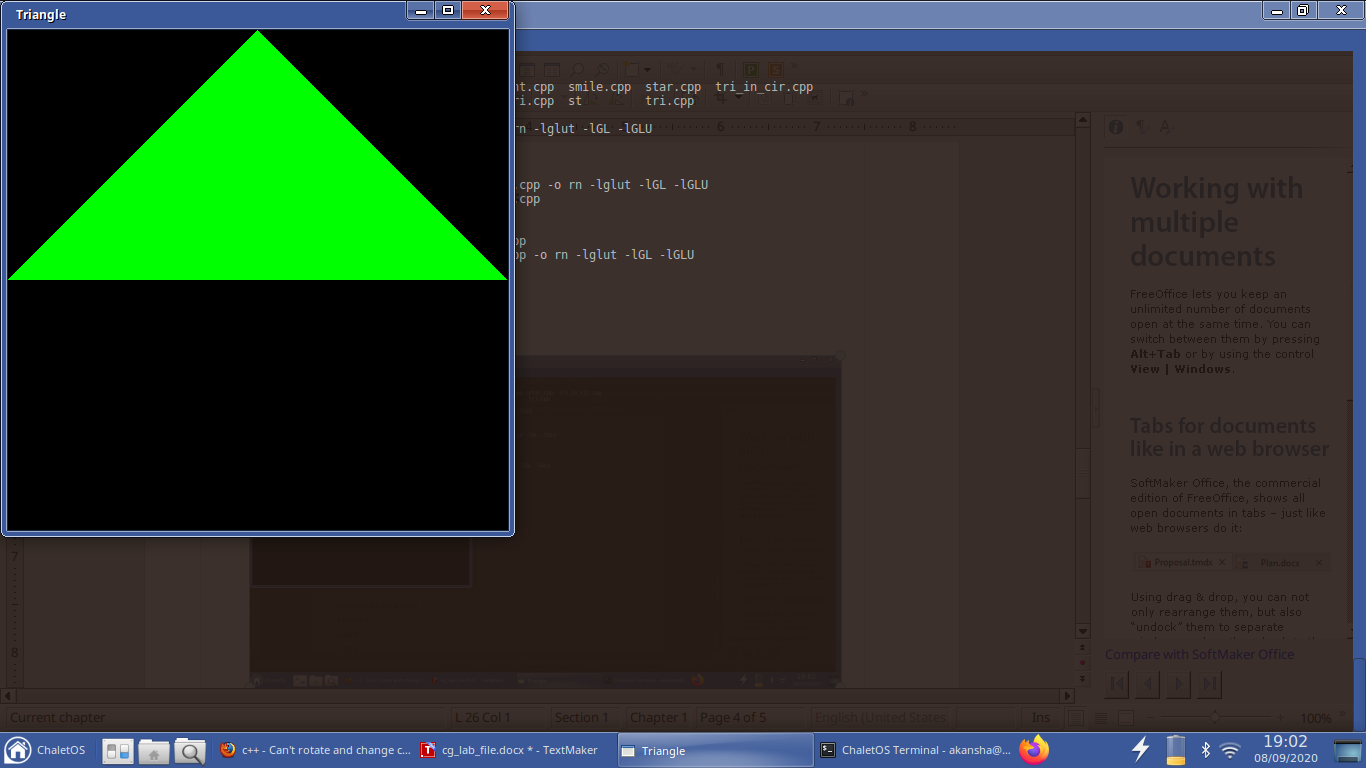
**Initially:**



**on pressing ‘r’ key:**



**on pressing ‘g’ key:**



**4. MOUSE CLICK EVENT**

#include<GL/glut.h>

#include<GL/gl.h>

int r=1,g=1,b=1;

void mouse(int button,int state,int x,int y)

{ switch (button) {

case GLUT\_LEFT\_BUTTON:

{ r=0;g=1;b=0;

break;}

case GLUT\_RIGHT\_BUTTON:

{ r=1;g=0;b=0;

break;}

}

glutPostRedisplay();

}

void triangle(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_TRIANGLES);

glColor3f(r,g,b);

glVertex2f(0,1);

glVertex2f(-1,0);

glVertex2f(1,0);

glEnd();

glFlush();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

glutCreateWindow("Triangle");

glutDisplayFunc(triangle);

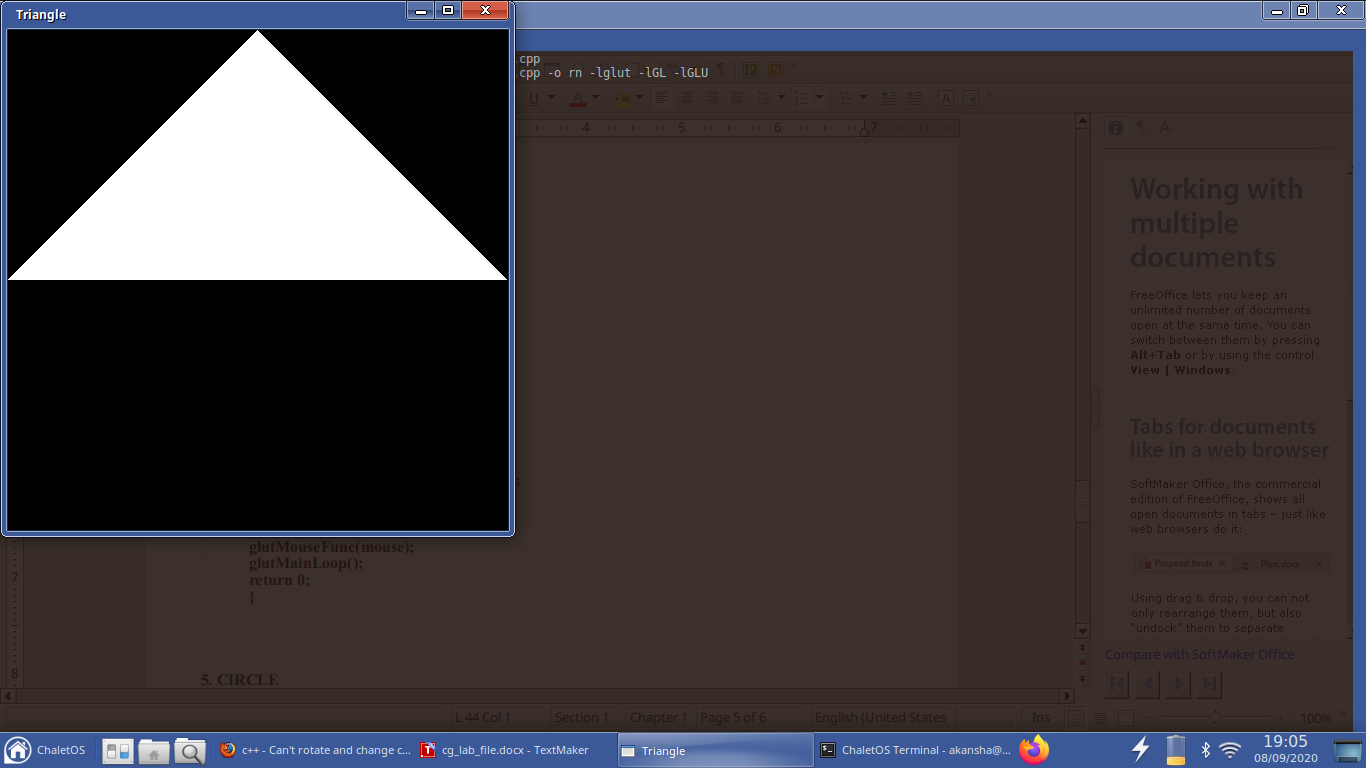
glutMouseFunc(mouse);

glutMainLoop();

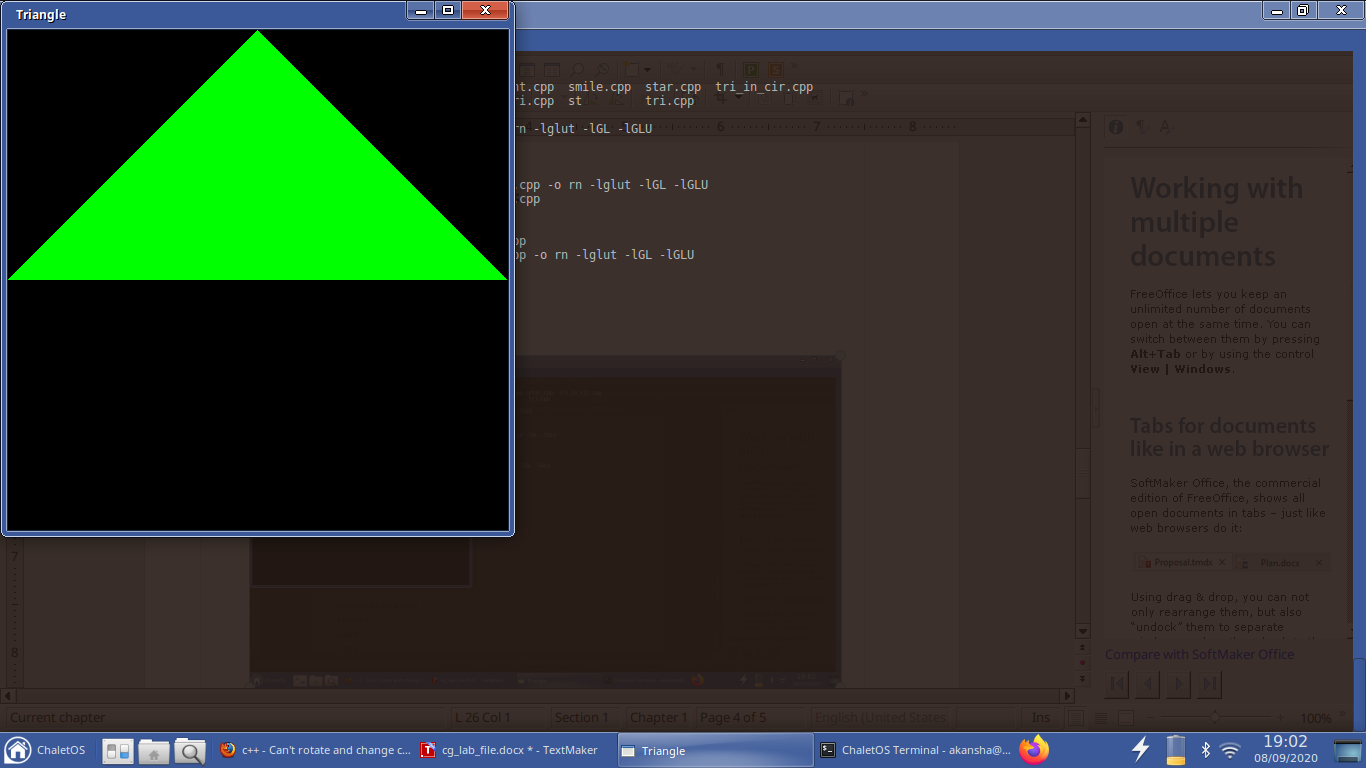
return 0;

}

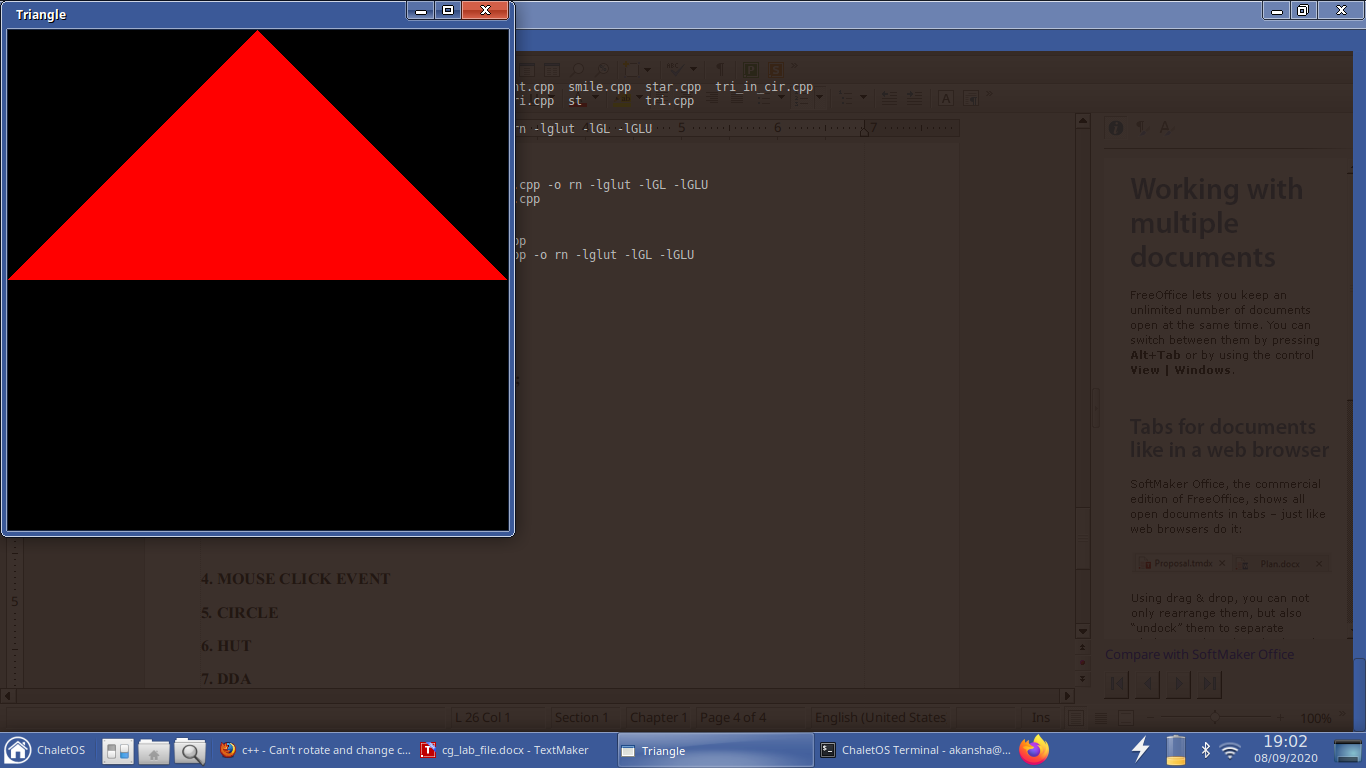
**Initially:**



**on left click:**



**on right click:**



**5. CIRCLE**

#include<GL/glut.h>

#include<GL/gl.h>

#include<cmath>

void circle(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_POINTS);

for(int i=0;i<180;++i)

{

glVertex3f(cos(2\*3.14159\*i/180),sin(2\*3.14159\*i/180),0);

}

glEnd();

glFlush();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(200,200);

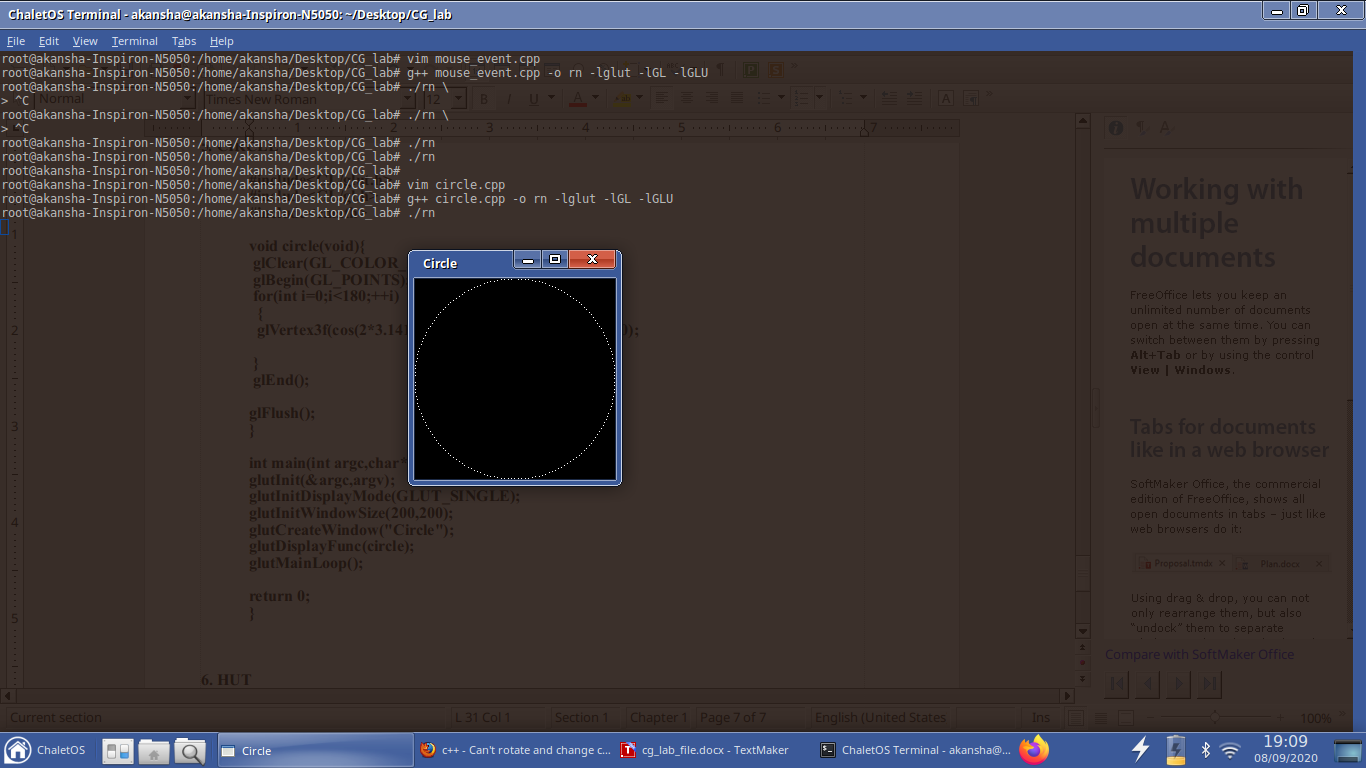
glutCreateWindow("Circle");

glutDisplayFunc(circle);

glutMainLoop();

return 0;

}



**6. HUT**

#include<GL/glut.h>

#include<GL/gl.h>

void hut(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_LINES);

glColor3f(0,1,0);

glVertex2f(20,20);

glVertex2f(20,200);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,0);

glVertex2f(20,200);

glVertex2f(280,200);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,0);

glVertex2f(280,200);

glVertex2f(280,20);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,0);

glVertex2f(150,200);

glVertex2f(150,20);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,0);

glVertex2f(280,20);

glVertex2f(20,20);

glEnd();

glBegin(GL\_LINES);

glColor3f(1,0,0);

glVertex2f(150,200);

glVertex2f(215,250);

glEnd();

glBegin(GL\_LINES);

glColor3f(1,0,0);

glVertex2f(215,250);

glVertex2f(280,200);

glEnd();

glBegin(GL\_LINES);

glColor3f(1,0,0);

glVertex2f(20,200);

glVertex2f(85,250);

glEnd();

glBegin(GL\_LINES);

glColor3f(1,0,0);

glVertex2f(85,250);

glVertex2f(150,200);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,1);

glVertex2f(70,70);

glVertex2f(70,110);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,1);

glVertex2f(70,110);

glVertex2f(100,110);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,1);

glVertex2f(100,110);

glVertex2f(100,70);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,1);

glVertex2f(100,70);

glVertex2f(70,70);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,1);

glVertex2f(210,20);

glVertex2f(210,100);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,1);

glVertex2f(210,100);

glVertex2f(230,100);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,1);

glVertex2f(230,100);

glVertex2f(230,20);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,1);

glVertex2f(230,20);

glVertex2f(210,20);

glEnd();

glFlush();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(1000,1000);

glutCreateWindow("HUT");

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

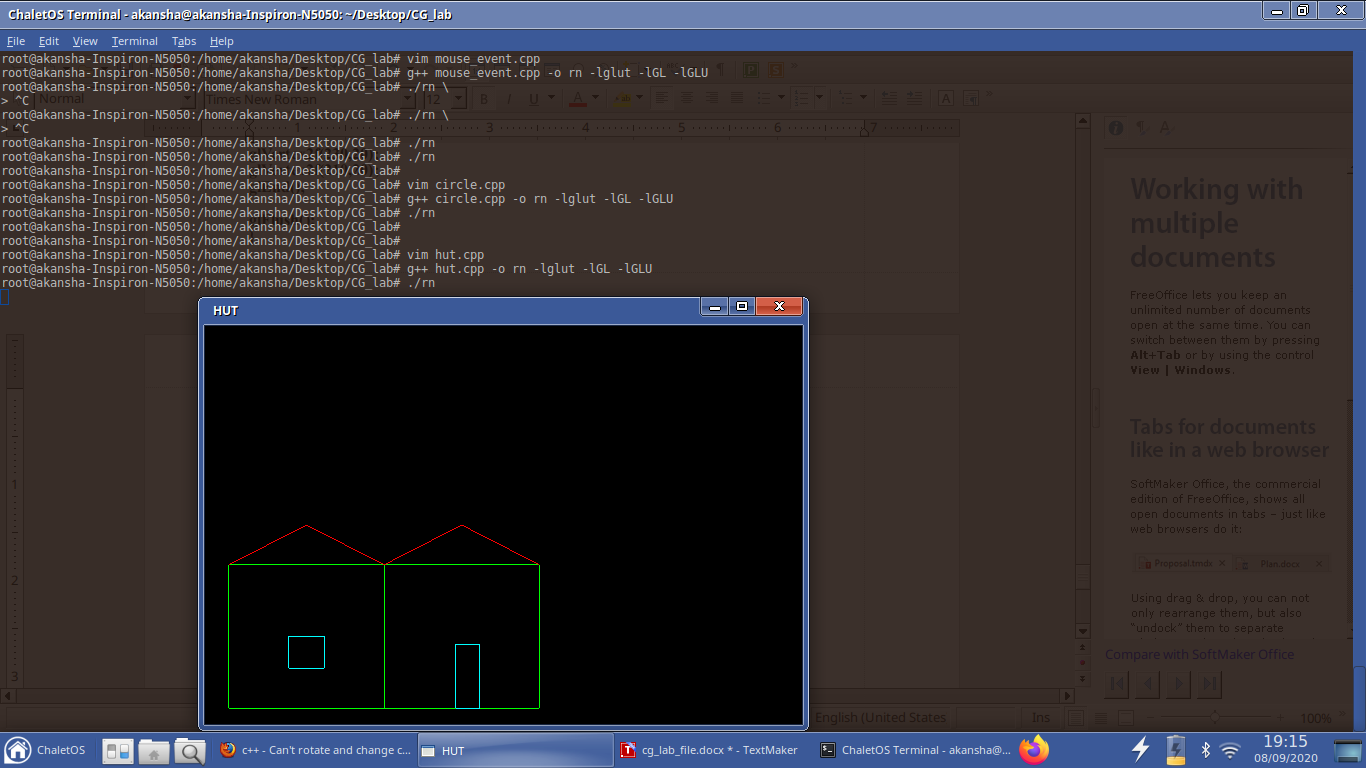
gluOrtho2D(0,500,0,500);

glutDisplayFunc(hut);

glutMainLoop();

return 0;

}



**7. DDA**

#include<GL/glut.h>

#include<GL/gl.h>

#include<cmath>

#include<iostream>

using namespace std;

float x1,yp1,x2,y2;

void dda\_line(){

float dx,dy;

dx=x2-x1;

dy=y2-yp1;

int step;

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

{step=abs(dx);}

else

{step=abs(dy);}

dx=dx/step;

dy=dy/step;

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,200,0,200);

glPointSize(1.0);

glBegin(GL\_POINTS);

glColor3f(0,1,1);

glVertex2f(round(x1),round(yp1));

for(int i=1;i<=step;i++)

{

//glBegin(GL\_POINTS);

x1=x1+dx;

yp1=yp1+dy;

glVertex2f(round(x1),round(yp1));

}

glEnd();

glFlush();

}

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

glutInit(&argc,argv);

cout<<"x1: ";

cin>>x1;

cout<<"y1: ";

cin>>yp1;

cout<<"x2: ";

cin>>x2;

cout<<"y2: ";

cin>>y2;

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

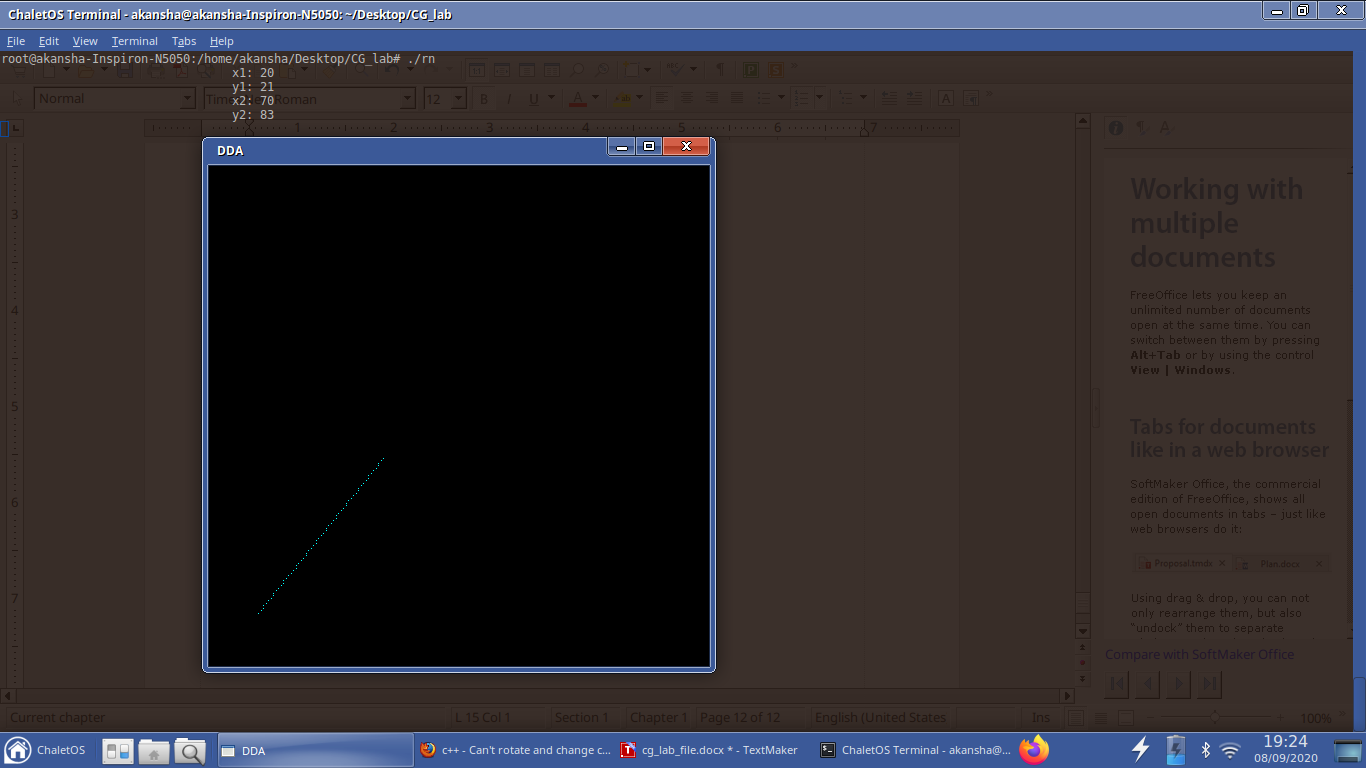
glutCreateWindow("DDA");

glutDisplayFunc(dda\_line);

glutMainLoop();

return 0;

}



**8. BRESENHAM**

#include<GL/glut.h>

#include<GL/gl.h>

#include<cmath>

#include<iostream>

using namespace std;

int x1,yp1,x2,y2;

void br\_line(){

int dx,dy;

dx=x2-x1;

dy=y2-yp1;

int p;

p=2\*dy-dx;

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,200,0,200);

glPointSize(1.0);

glBegin(GL\_POINTS);

glColor3f(0,1,1);

while(x1<=x2)

{

glVertex2f(x1,yp1);

x1=x1+1;

if(p>0)

{

yp1=yp1+1;

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

}

else{

p=p+2\*dy;

}

}

glEnd();

glFlush();

}

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

glutInit(&argc,argv);

//double x1,y1,x2,y2;

cout<<"x1: ";

cin>>x1;

cout<<"y1: ";

cin>>yp1;

cout<<"x2: ";

cin>>x2;

cout<<"y2: ";

cin>>y2;

if(x1>x2 || yp1>y2){

cout<<"Invalid Points \n";

return 0;

}

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

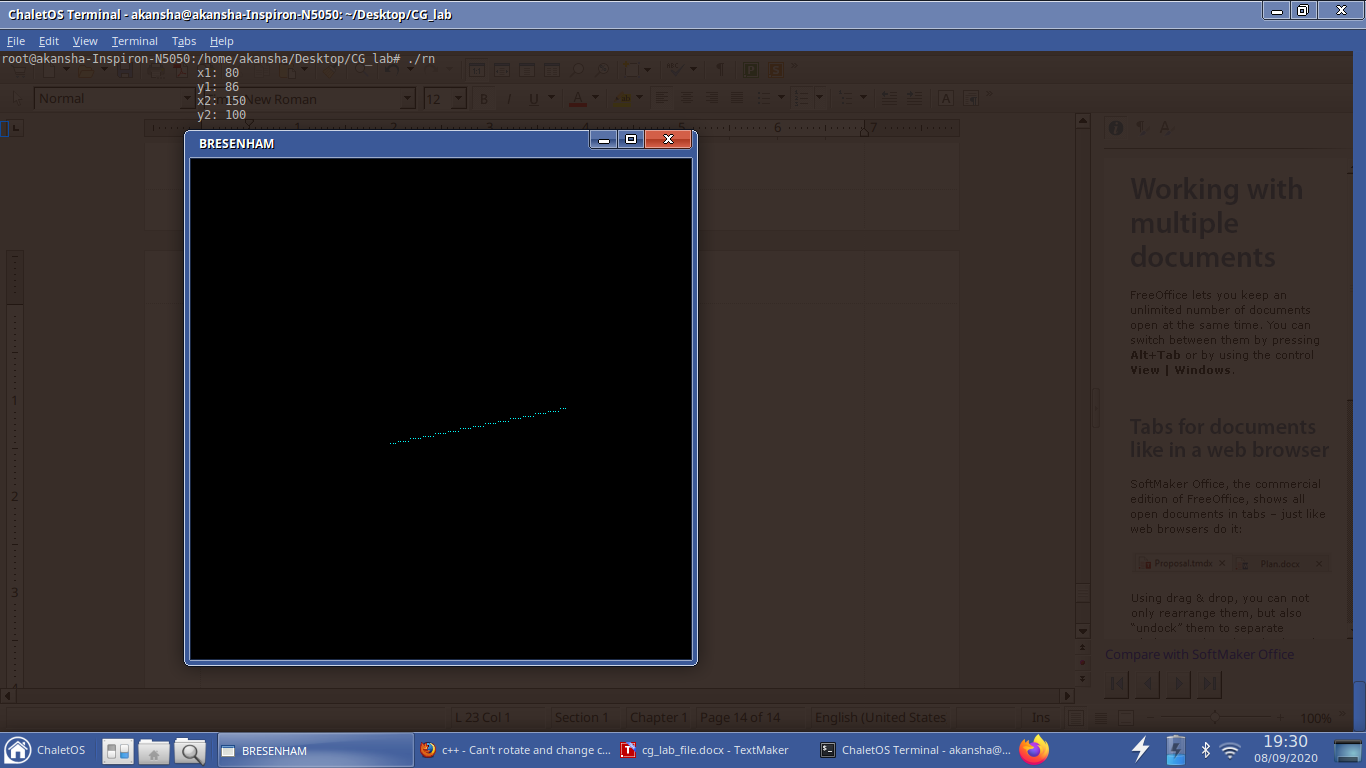
glutCreateWindow("BRESENHAM");

glutDisplayFunc(br\_line);

glutMainLoop();

return 0;

}



**9. STAR**

#include<GL/glut.h>

#include<GL/gl.h>

void star(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_LINES);

glColor3f(0,1,0);

glVertex2f(0,1);

glVertex2f(0.5,-0.25);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,0);

glVertex2f(0.5,-0.25);

glVertex2f(-0.5,-0.25);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,1,0);

glVertex2f(-0.5,-0.25);

glVertex2f(0,1);

glEnd();

glBegin(GL\_LINES);

glColor3f(1,0,0);

glVertex2f(-0.5,0.5);

glVertex2f(0.5,0.5);

glEnd();

glBegin(GL\_LINES);

glColor3f(1,0,0);

glVertex2f(0.5,0.5);

glVertex2f(0,-0.5);

glEnd();

glBegin(GL\_LINES);

glColor3f(1,0,0);

glVertex2f(0,-0.5);

glVertex2f(-0.5,0.5);

glEnd();

glBegin(GL\_LINES);

glColor3f(0,0,1);

glVertex2f(0,0.5);

glVertex2f(0.25,0);

glEnd();

//drawstring();

glBegin(GL\_LINES);

glColor3f(0,0,1);

glVertex2f(0.25,0);

glVertex2f(-0.25,0);

glEnd();

//drawstring();

glBegin(GL\_LINES);

glColor3f(0,0,1);

glVertex2f(-0.25,0);

glVertex2f(0,0.5);

glEnd();

//drawstring();

glBegin(GL\_LINES);

glColor3f(0,1,1);

glVertex2f(0.125,0.25);

glVertex2f(-0.125,0.25);

glEnd();

//drawstring();

glBegin(GL\_LINES);

glColor3f(0,1,1);

glVertex2f(0.125,0.25);

glVertex2f(0,0);

glEnd();

//drawstring();

glBegin(GL\_LINES);

glColor3f(0,1,1);

glVertex2f(0,0);

glVertex2f(-0.125,0.25);

glEnd();

//drawstring();

glFlush();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

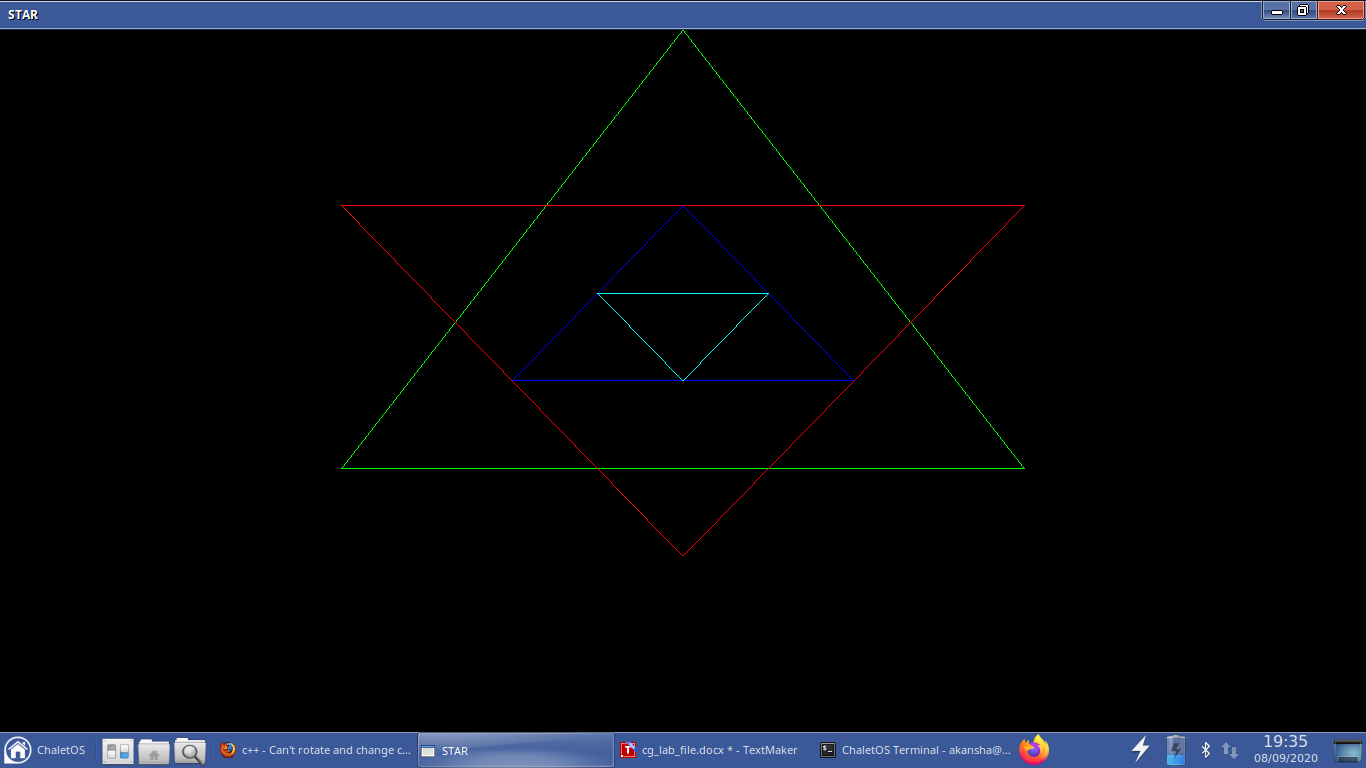
glutInitWindowSize(2000,2000);

glutCreateWindow("STAR");

glutDisplayFunc(star);

return 0;

}



**10. TRIANGLE (FILLED WITH LINES) IN CIRCLE**

#include<GL/glut.h>

#include<GL/gl.h>

#include<cmath>

#include<iostream>

using namespace std;

//float x1,yp1,x2,y2;

float arr1[8][2];

float arr2[8][2];

void dda\_line(int c,float x1,float yp1,float x2,float y2){

float dx,dy;

dx=x2-x1;

dy=y2-yp1;

int step;

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

{step=abs(dx);}

else

{step=abs(dy);}

dx=dx/step;

dy=dy/step;

//glClear(GL\_COLOR\_BUFFER\_BIT);

int temp;

if(c==3){

temp=step/10;

c=1;

}

else{

temp=step/8;

}

int tc=0;

glPointSize(1.0);

glBegin(GL\_POINTS);

glColor3f(0,1,1);

glVertex2f(round(x1),round(yp1));

for(int i=1;i<=step;i++)

{

//glBegin(GL\_POINTS);

x1=x1+dx;

yp1=yp1+dy;

glVertex2f(round(x1),round(yp1));

if(c!=0 && tc!=8 && i%temp==0){

if(c==1)

{arr1[tc][0]=round(x1);

arr1[tc][1]=round(yp1);

tc++;

}

else if(c==2)

{arr2[tc][0]=round(x1);

arr2[tc][1]=round(yp1);

tc++;

}

}

}

glEnd();

glFlush();

}

void d(){

glClear(GL\_COLOR\_BUFFER\_BIT);

float t1,t2,t3,t4,t5,t6;

glBegin(GL\_POINTS);

for(int i=0;i<360;i++)

{

glVertex3f(200+150\*cos(3.14159\*i/180),200+150\*sin(3.14159\*i/180),0);

if(i==90)

{

t1=200+150\*cos(3.14159\*i/180.0);

t2=200+150\*sin(3.14159\*i/180.0);

}

else if(i==225)

{

t3=200+150\*cos(3.14159\*i/180.0);

t4=200+150\*sin(3.14159\*i/180.0);

}

else if(i==315)

{

t5=200+150\*cos(3.14159\*i/180.0);

t6=200+150\*sin(3.14159\*i/180.0);

t6=200+150\*sin(3.14159\*i/180.0);

}

}

dda\_line(1,t1,t2,t3,t4);

dda\_line(0,t5,t6,t3,t4);

float x1,yp1,x2,y2;

x1=t1;

yp1=t2;

x2=(t3+t5)/2;

y2=(t4+t6)/2;

dda\_line(2,x1,yp1,x2,y2);

for(int j=0;j<8;j++){

x1=arr1[j][0];

yp1=arr1[j][1];

x2=arr2[j][0];

y2=arr2[j][1];

glColor3f(0,1,1);

glBegin(GL\_LINES);

glVertex2f(x1,yp1);

glVertex2f(x2,y2);

glEnd();

}

dda\_line(3,t1,t2,t5,t6);

for(int j=0;j<8;j++){

x1=arr1[j][0];

yp1=arr1[j][1];

x2=arr2[j][0];

y2=arr2[j][1];

glColor3f(1,0,0);

glBegin(GL\_LINES);

glVertex2f(x1,yp1);

glVertex2f(x2,y2);

glEnd();

}

glFlush();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

glutCreateWindow("tri in circle");

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

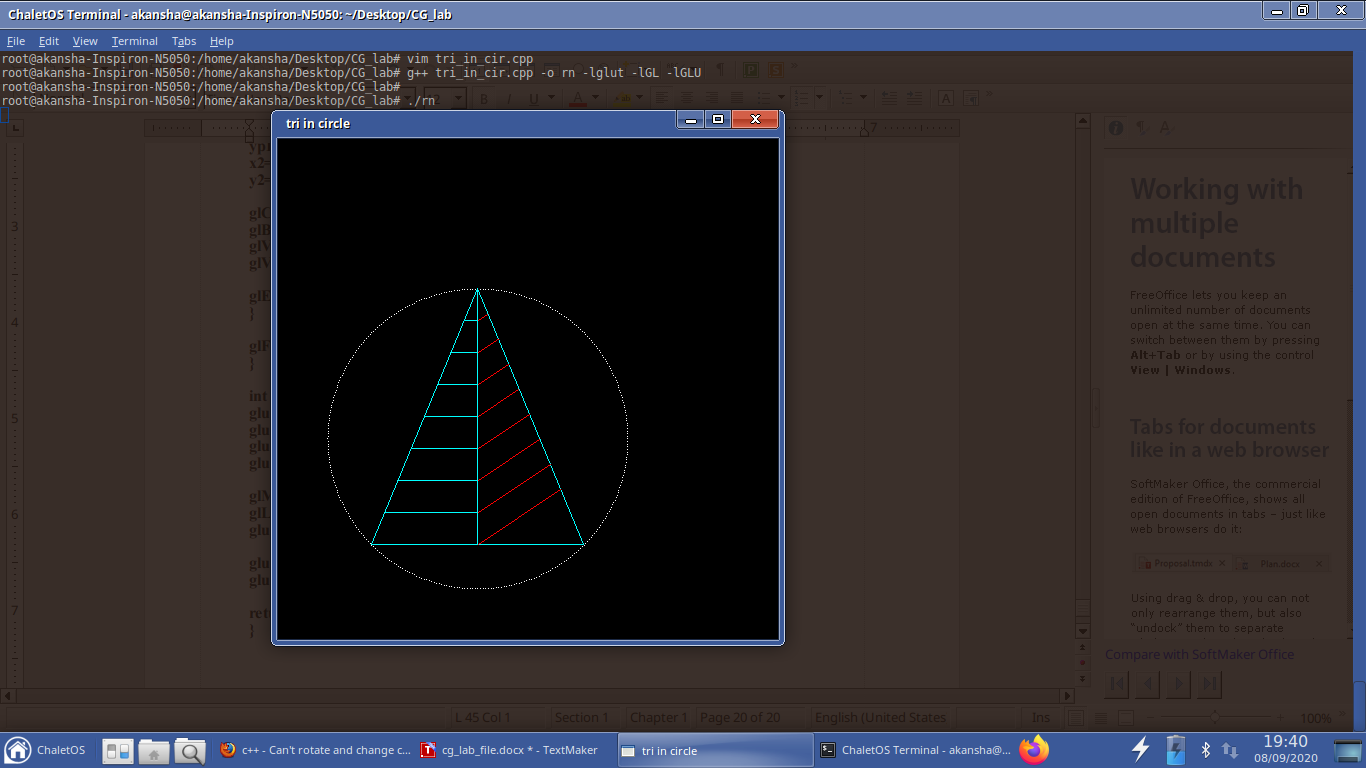
gluOrtho2D(0,500,0,500);

glutDisplayFunc(d);

glutMainLoop();

return 0;

}



**11. SMILE FACE**

#include<GL/glut.h>

#include<GL/gl.h>

#include<cmath>

void draw(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

float t1,t2,t3,t4,t5,t6;

glPointSize(2);

glBegin(GL\_POINTS);

for(int i=0;i<360;i++)

{

glVertex3f(200+150\*cos(3.14159\*i/180),200+150\*sin(3.14159\*i/180),0);

}

glEnd();

glPointSize(2);

glBegin(GL\_POINTS);

for(int i=0;i<360;i++)

{

glVertex3f(240+10\*cos(3.14159\*i/180),290+10\*sin(3.14159\*i/180),0);

}

glEnd();

glPointSize(2);

glBegin(GL\_POINTS);

for(int i=0;i<360;i++)

{

glVertex3f(155+10\*cos(3.14159\*i/180),290+10\*sin(3.14159\*i/180),0);

}

glEnd();

glPointSize(2);

glBegin(GL\_POINTS);

for(int i=225;i<315;i++)

{

glVertex3f(200+50\*cos(3.14159\*i/180),200+50\*sin(3.14159\*i/180),0);

}

glEnd();

glBegin(GL\_LINES);

glVertex2f(200,250);

glVertex2f(200,220);

glEnd();

glFlush();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

glutCreateWindow("SMILE");

//init();

//glClearColor(0,0,0,1);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,500,0,500);

glutDisplayFunc(draw);

glutMainLoop();

return 0;

}



**12. CIRCLES INSIDE CIRCLE (FILL SPACE WITH LINES)**

#include<GL/glut.h>

#include<GL/gl.h>

#include<cmath>

int x1,yp1,x2,y2;

void dda\_line();

float dx,dy;

dx=x2-x1;

dy=y2-yp1;

int step;

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

{step=abs(dx);}

else

{step=abs(dy);}

dx=dx/step;

dy=dy/step;

//glPointSize(2.0);

glBegin(GL\_POINTS);

//glColor3f(0,1,0);

glVertex2f(round(x1),round(yp1));

for(int i=1;i<=step;i++)

{

//glBegin(GL\_POINTS);

x1=x1+dx;

yp1=yp1+dy;

glVertex2f(round(x1),round(yp1));

}

glEnd();

glFlush();

}

void draw(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

for(int i=0;i<360;i++)

{

x1=200+150\*cos(3.14159\*i/180);

yp1=200+150\*sin(3.14159\*i/180);

x2=200+130\*cos(3.14159\*i/180);

y2=200+130\*sin(3.14159\*i/180);

if(i%2==0)

{

glColor3f(0,1,0);

glBegin(GL\_LINES);

glVertex2f(x1,yp1);

glVertex2f(x2,y2);

glEnd();

}

x1=200+110\*cos(3.14159\*(i+4)/180);

yp1=200+110\*sin(3.14159\*(i+4)/180);

if(i%2==0){

glColor3f(0,0,1);

glBegin(GL\_LINES);

glVertex2f(x1,yp1);

glVertex2f(x2,y2);

glEnd();}

//dda\_line();

}

for(int i=0;i<360;i++)

{

x1=200+110\*cos(3.14159\*i/180);

yp1=200+110\*sin(3.14159\*i/180);

x2=200+90\*cos(3.14159\*i/180);

y2=200+90\*sin(3.14159\*i/180);

if(i%4==0)

{

glColor3f(1,1,1);

glBegin(GL\_LINES);

glVertex2f(x1,yp1);

glVertex2f(x2,y2);

glEnd();

}

}

for(int i=0;i<360;i++)

{

x1=200+110\*cos(3.14159\*i/180);

yp1=200+110\*sin(3.14159\*i/180);

x2=200+90\*cos(3.14159\*(i+40)/180);

y2=200+90\*sin(3.14159\*(i+40)/180);

if(i%4==0)

{

glColor3f(1,1,1);

glBegin(GL\_LINES);

glVertex2f(x1,yp1);

glVertex2f(x2,y2);

glEnd();

}

}

for(int i=0;i<360;i++)

{

x1=200+90\*cos(3.14159\*i/180);

yp1=200+90\*sin(3.14159\*i/180);

x2=200+70\*cos(3.14159\*i/180);

y2=200+70\*sin(3.14159\*i/180);

if(i%2==0)

{

glColor3f(1,0,0);

glBegin(GL\_LINES);

glVertex2f(x1,yp1);

glVertex2f(x2,y2);

glEnd();

}

x1=200+40\*cos(3.14159\*(i+4)/180);

yp1=200+40\*sin(3.14159\*(i+4)/180);

if(i%4==0){

glColor3f(0,1,1);

glBegin(GL\_LINES);

glVertex2f(x1,yp1);

glVertex2f(x2,y2);

glEnd();}

//dda\_line();

}

for(int i=0;i<360;i++)

{

x1=200+70\*cos(3.14159\*i/180);

yp1=200+70\*sin(3.14159\*i/180);

x2=200+40\*cos(3.14159\*(i+40)/180);

y2=200+40\*sin(3.14159\*(i+40)/180);

if(i%4==0)

{

glColor3f(0,1,1);

glBegin(GL\_LINES);

glVertex2f(x1,yp1);

glVertex2f(x2,y2);

glEnd();

}

x1=200+10\*cos(3.14159\*(i+20)/180);

yp1=200+10\*sin(3.14159\*(i+20)/180);

if(i%4==0)

{

glColor3f(1,1,0);

glBegin(GL\_LINES);

glVertex2f(x1,yp1);

glVertex2f(x2,y2);

glEnd();

}

}

glFlush();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

glutCreateWindow("CIRCLES");

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,500,0,500);

glutDisplayFunc(draw);

glutMainLoop();

return 0;

}



**13. BAR GRAPH**

#include<GL/glut.h>

#include<GL/gl.h>

#include<cmath>

void drawstring()

{

//glClear(GL\_COLOR\_BUFFER\_BIT);

char v='x';

glRasterPos3f(405,20,0);

glColor3f(1,1,1);

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15,v);

v='y';

glRasterPos3f(20,409,0);

glColor3f(1,1,1);

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15,v);

v='o';

glRasterPos3f(12,12,0);

glColor3f(1,1,1);

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15,v);

v='>';

glRasterPos3f(395,16,0);

glColor3f(1,1,1);

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15,v);

v='^';

glRasterPos3f(16,395,0);

glColor3f(1,1,1);

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15,v);

}

void bar(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

drawstring();

glBegin(GL\_LINES);

glColor3f(1,1,1);

glVertex2f(20,20);

glVertex2f(20,400);

glEnd();

glBegin(GL\_LINES);

glColor3f(1,1,1);

glVertex2f(20,20);

glVertex2f(400,20);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(1,0,0);

glVertex2f(40,20);

glVertex2f(40,300);

glVertex2f(60,300);

glVertex2f(60,20);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(1,1,0);

glVertex2f(80,20);

glVertex2f(80,250);

glVertex2f(100,250);

glVertex2f(100,20);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0,0,1);

glVertex2f(120,20);

glVertex2f(120,350);

glVertex2f(140,350);

glVertex2f(140,20);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0,1,1);

glVertex2f(160,20);

glVertex2f(160,200);

glVertex2f(180,200);

glVertex2f(180,20);

glEnd();

glFlush();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

glutCreateWindow("bar graph");

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,500,0,500);

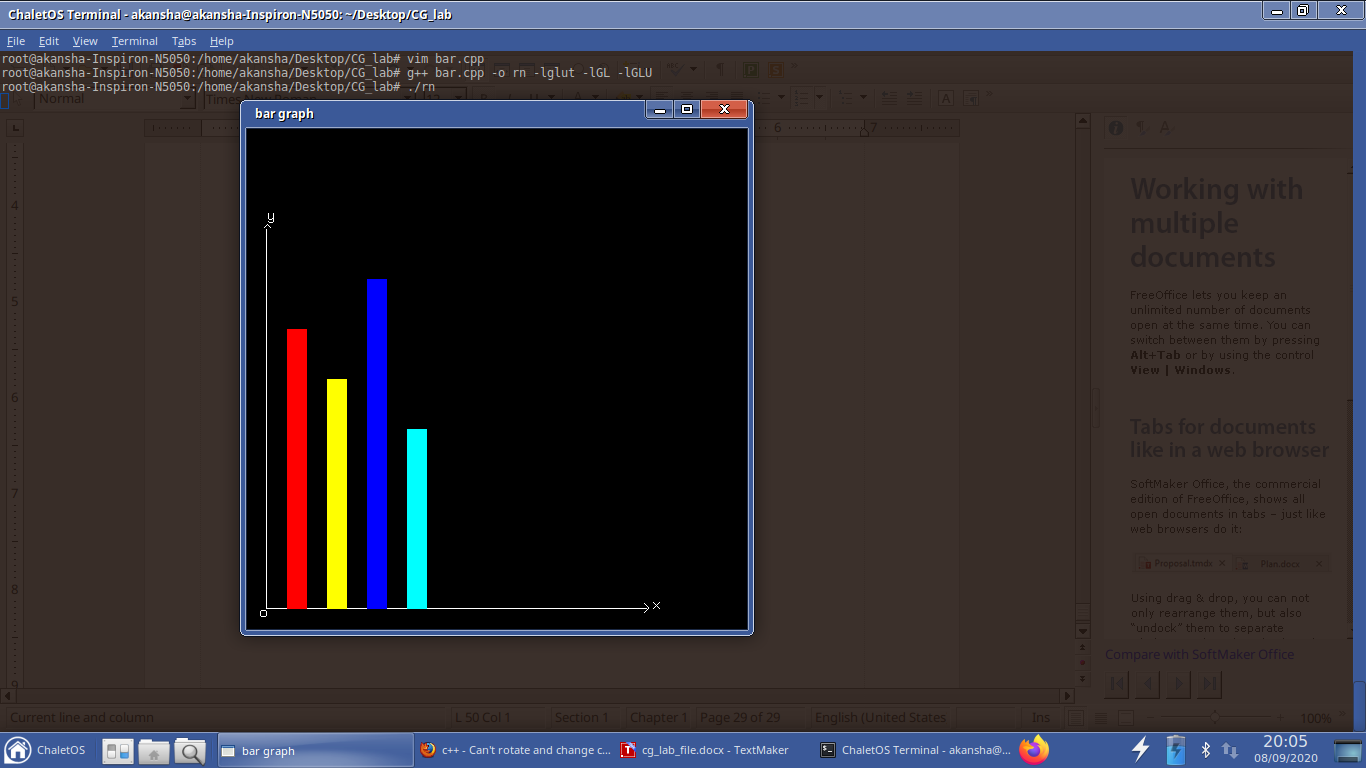
glutDisplayFunc(bar);

glutMainLoop();

return 0;

}

output:



**14. TRANSFORMATIONS (SCALING, ROTATION AND REFLECTION)**

#include<GL/glut.h>

#include<GL/gl.h>

#include<cmath>

#include<iostream>

using namespace std;

void scale\_draw(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1,1,1);

glBegin(GL\_TRIANGLES);

glVertex2f(100,100);

glVertex2f(200,150);

glVertex2f(100,200);

glEnd();

glScalef(2,2,0);

glColor3f(1,0,0);

glBegin(GL\_TRIANGLES);

glVertex2f(100,100);

glVertex2f(200,150);

glVertex2f(100,200);

glEnd();

glFlush();

}

void rotate\_draw(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1,1,1);

glBegin(GL\_TRIANGLES);

glVertex2f(100,100);

glVertex2f(200,150);

glVertex2f(100,200);

glEnd();

glTranslatef(100, 100, 0);

glRotatef(45, 0, 0, 1);

glTranslatef(-100, -100, 0);

glColor3f(1,0,0);

glBegin(GL\_TRIANGLES);

glVertex2f(100,100);

glVertex2f(200,150);

glVertex2f(100,200);

glEnd();

glFlush();

}

void reflection\_draw(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1,1,1);

glBegin(GL\_TRIANGLES);

glVertex2f(300,-200);

glVertex2f(400,-250);

glVertex2f(200,-300);

glEnd();

glPushMatrix();

glScalef(1, -1, 1);

glColor3f(1,0,0);

glBegin(GL\_TRIANGLES);

glVertex2f(300,-200);

glVertex2f(400,-250);

glVertex2f(200,-300);

glEnd();

glPopMatrix();

glFlush();

}

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

int c;

cout<<"\n1 - scale\n2 - rotate\n3 - reflection\n";

cin>>c;

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

glutCreateWindow("PATTERN");

glLoadIdentity();

gluOrtho2D(500,0,500,0);

switch(c){

case 1:

{glutDisplayFunc(scale\_draw);

break;}

case 2:

{glutDisplayFunc(rotate\_draw);

break;}

case 3:

{glutDisplayFunc(reflection\_draw);

break;}

}

glutMainLoop();

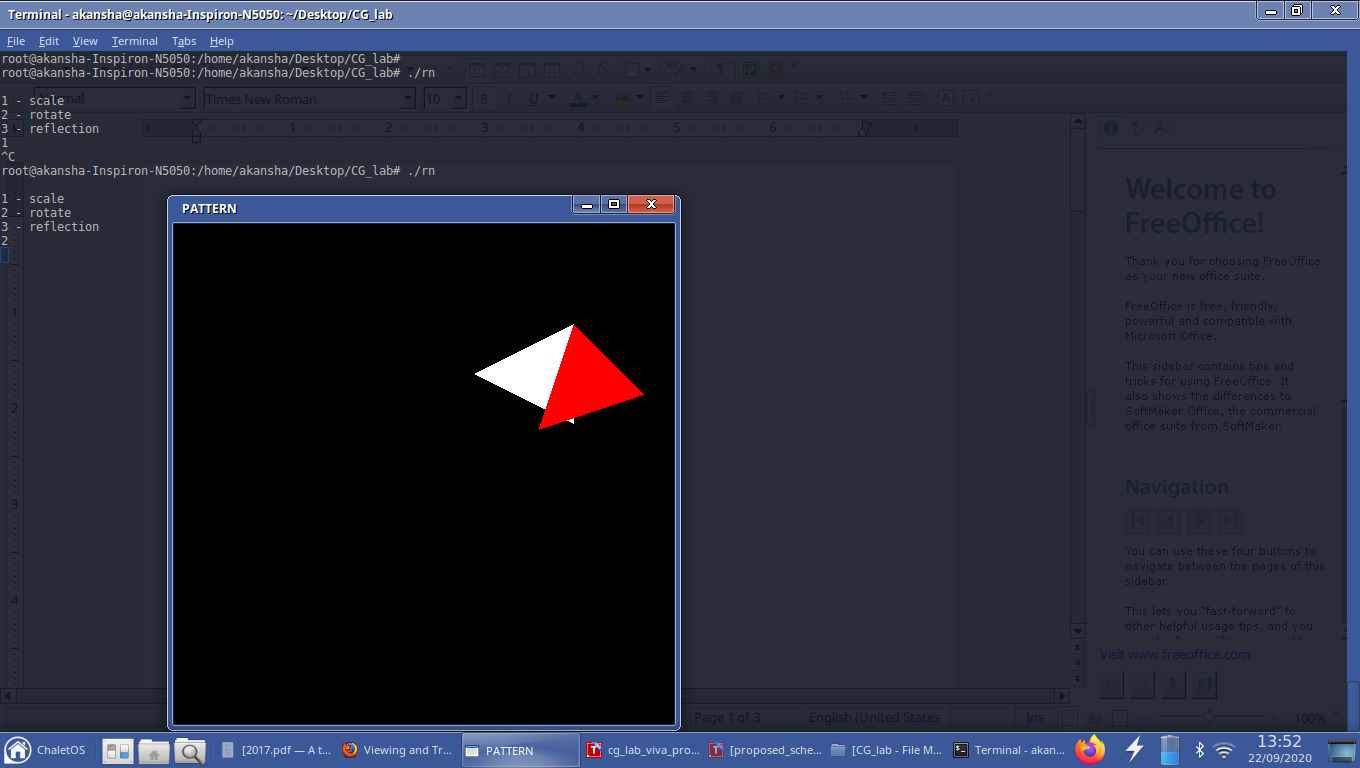
return 0;

}

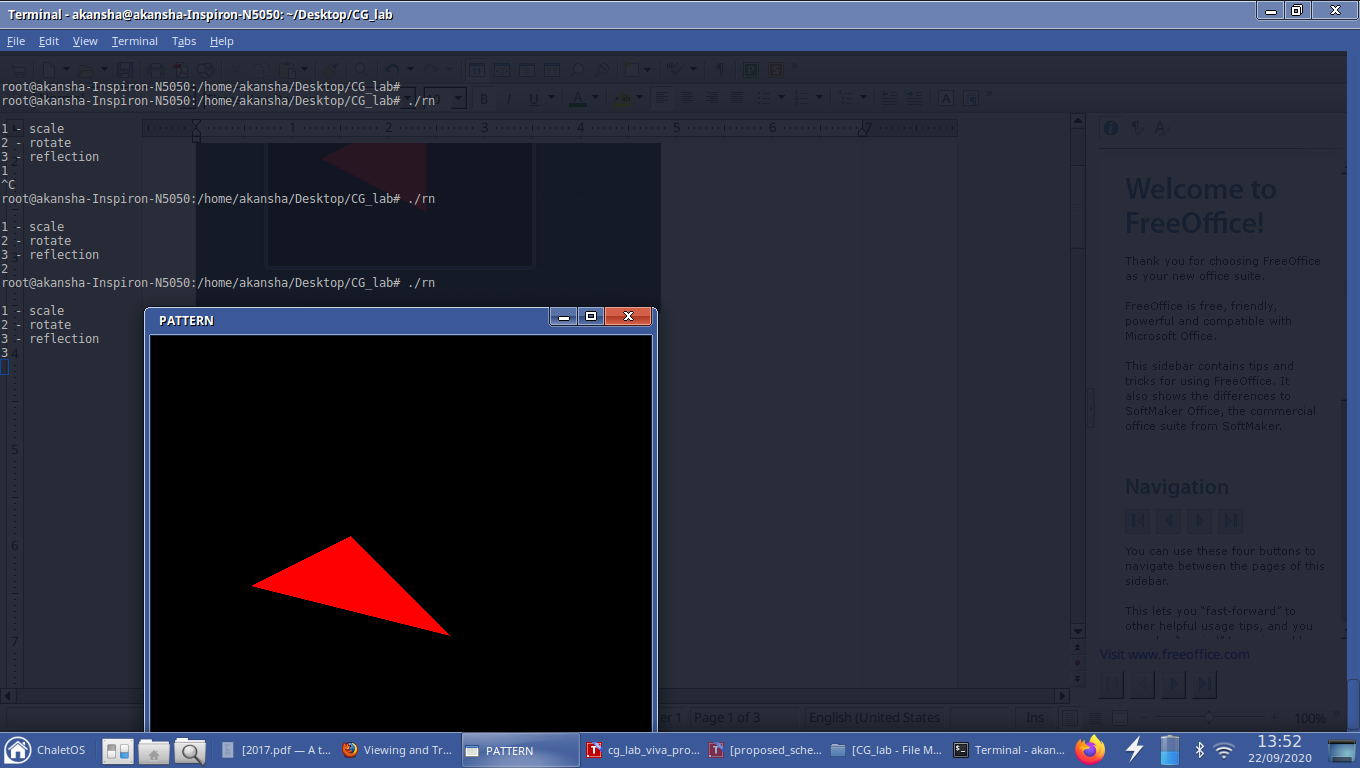
**scale**



**rotate**



**reflection**



**15. PATTERN**

#include<GL/glut.h>

#include<GL/gl.h>

#include<cmath>

void draw(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(2);

glBegin(GL\_POINTS);

for(int i=0;i<360;i++)

{

glVertex3f(150+50\*cos(3.14159\*i/180),100+50\*sin(3.14159\*i/180),0);

}

glEnd();

glPointSize(2);

glBegin(GL\_POINTS);

for(int i=0;i<360;i++)

{

glVertex3f(350+50\*cos(3.14159\*i/180),100+50\*sin(3.14159\*i/180),0);

}

glEnd();

glBegin(GL\_LINES);

glVertex2f(100,50);

glVertex2f(100,150);

glEnd();

glBegin(GL\_LINES);

glVertex2f(0,150);

glVertex2f(500,150);

glEnd();

glBegin(GL\_LINES);

glVertex2f(200,150);

glVertex2f(200,50);

glEnd();

glBegin(GL\_LINES);

glVertex2f(200,50);

glVertex2f(100,50);

glEnd();

glBegin(GL\_LINES);

glVertex2f(300,50);

glVertex2f(300,150);

glEnd();

glBegin(GL\_LINES);

glVertex2f(300,50);

glVertex2f(400,50);

glEnd();

glBegin(GL\_LINES);

glVertex2f(400,50);

glVertex2f(400,150);

glEnd();

glBegin(GL\_LINES);

glVertex2f(0,150);

glVertex2f(50,250);

glEnd();

glBegin(GL\_LINES);

glVertex2f(50,250);

glVertex2f(100,150);

glEnd();

glBegin(GL\_LINES);

glVertex2f(200,150);

glVertex2f(250,250);

glEnd();

glBegin(GL\_LINES);

glVertex2f(250,250);

glVertex2f(300,150);

glEnd();

glBegin(GL\_LINES);

glVertex2f(400,150);

glVertex2f(450,250);

glEnd();

glBegin(GL\_LINES);

glVertex2f(450,250);

glVertex2f(500,150);

glEnd();

glFlush();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

glutCreateWindow("PATTERN");

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

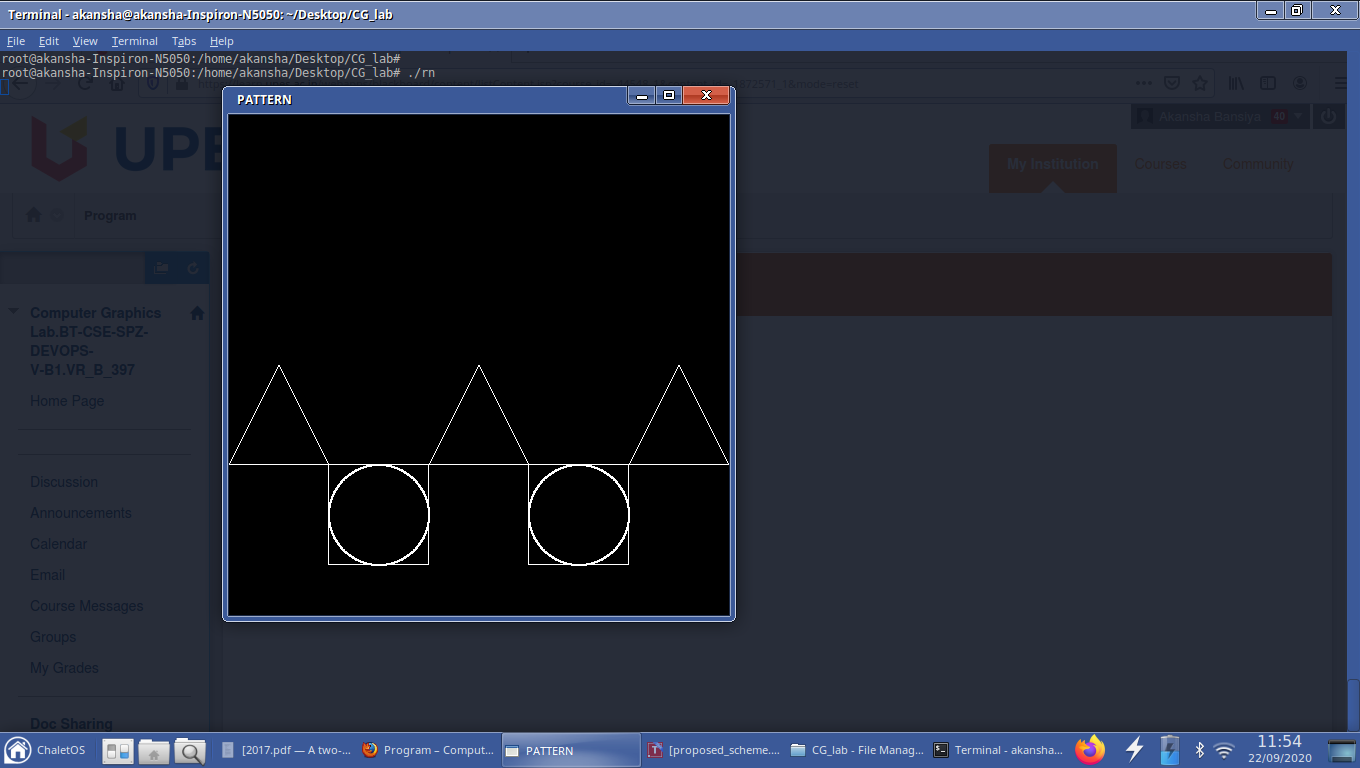
gluOrtho2D(0,500,0,500);

glutDisplayFunc(draw);

glutMainLoop();

return 0;

}



**16. CONTINUOUSLY MOVING TRIANGLE**

#include<GL/glut.h>

#include<GL/gl.h>

#include<time.h>

#include <unistd.h>

void delay(int number\_of\_seconds)

{

int milli\_seconds = 1000 \* number\_of\_seconds;

clock\_t start\_time = clock();

while (clock() < start\_time + milli\_seconds)

;

}

int x=0,y=0;

void triangle(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_TRIANGLES);

//glColor3f(r,g,b);

glVertex2f(10+x,10+y);

glVertex2f(40+x,40+y);

glVertex2f(70+x,10+y);

glEnd();

glFlush();

}

void m(){

for(int i=0;i<=20;i++)

{

//delay(100); //or sleep or usleep

usleep(100000);

x=x+5;

y=y+0;

triangle();

}

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

glutCreateWindow("Moving Triangle");

//init();

//drawstring();

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,500,0,500);

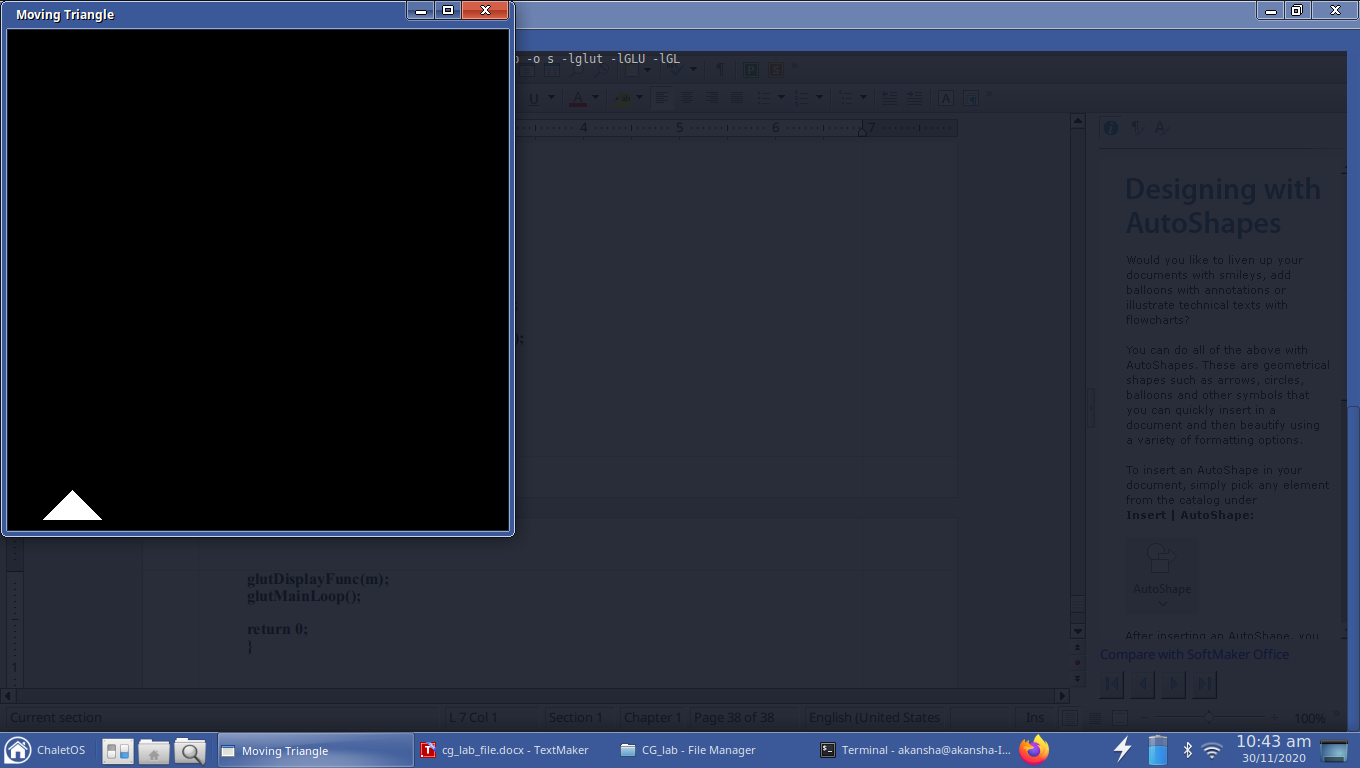
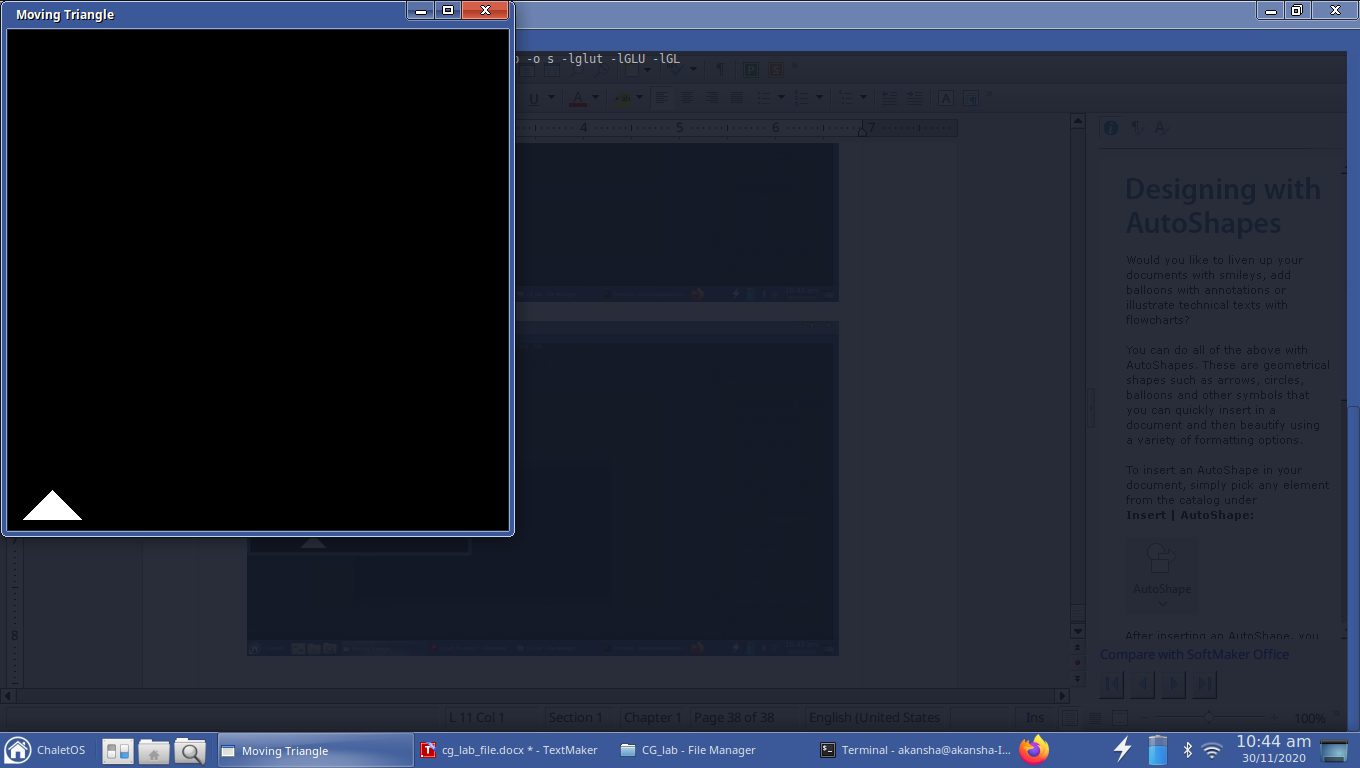
glutDisplayFunc(m);

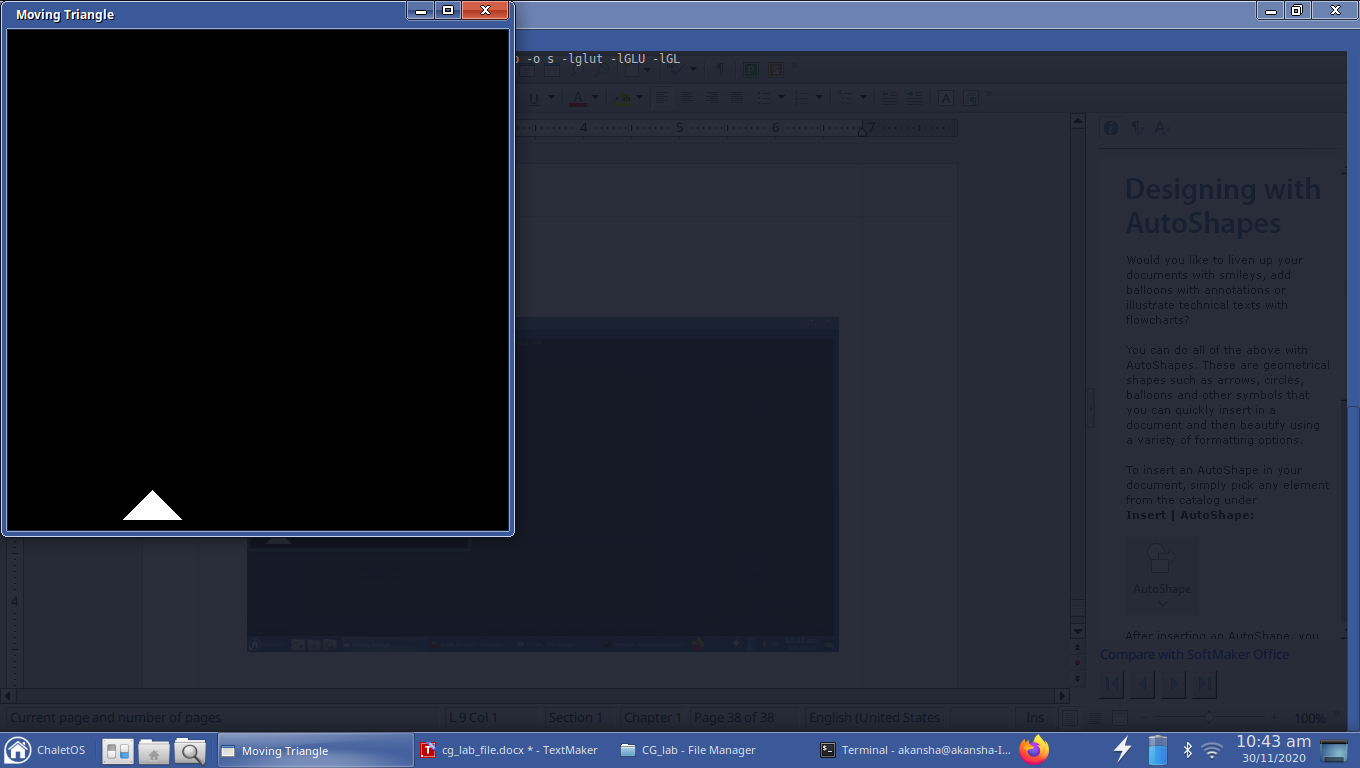
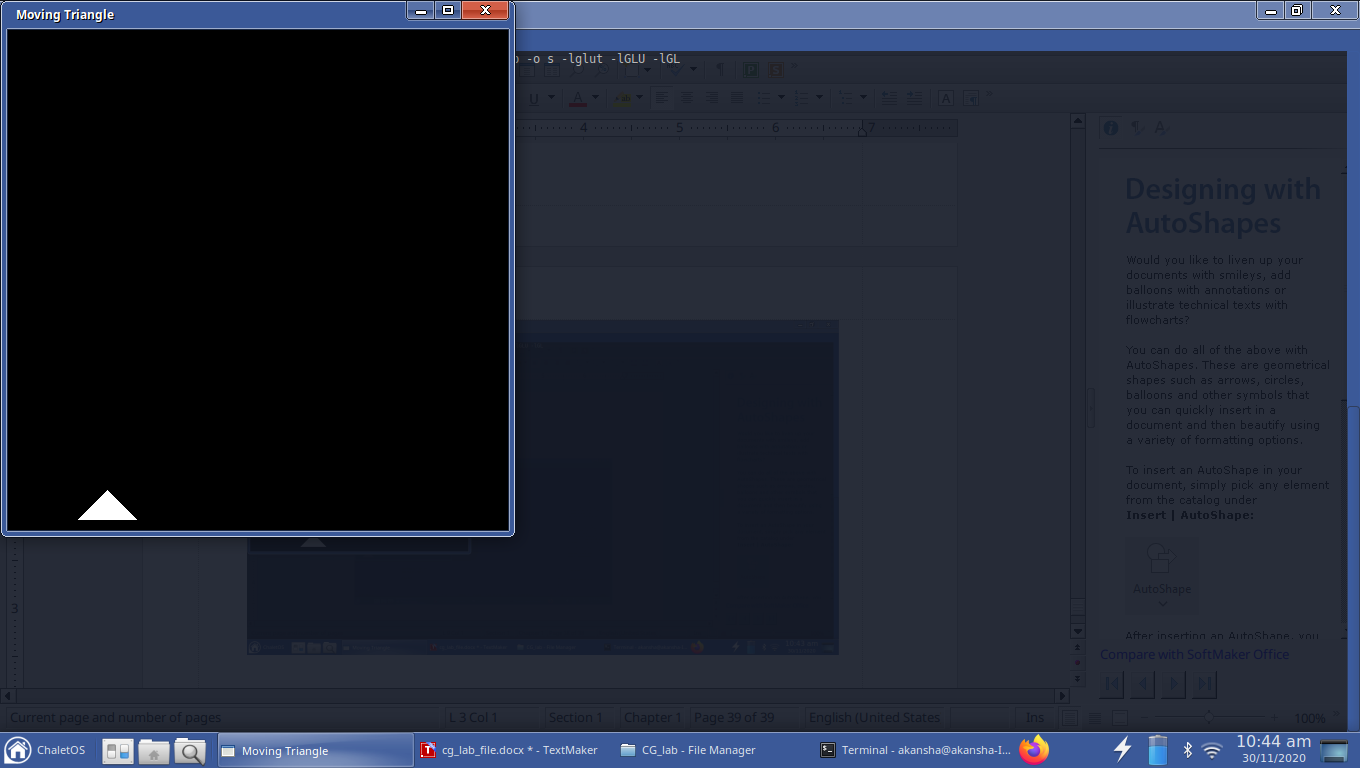
glutMainLoop();

return 0;

}

**Snapshots of moving triangle at different positions:**





**17. SIERPINSKI TRIANGLE**

#include<GL/glut.h>

#include<GL/gl.h>

#include<cmath>

void draw(float x1, float x2, float x3, float y1, float y2, float y3,int c1,int c2)

{

glColor3f(c1, c2, 1);

glBegin(GL\_TRIANGLES);

{

glVertex3f( (x1+x3)/2, (y1+y3)/2, 0);

glVertex3f( (x2+x3)/2, (y2+y3)/2, 0);

glVertex3f( (x2+x1)/2, (y2+y1)/2, 0);

}

glEnd();

}

void triangle(float x1, float x2, float x3, float y1, float y2, float y3,int c1,int c2){

draw(x1, x2, x3, y1, y2, y3,c1,c2);

if((x2-x1)>2 && (x3-x1)>2)

{

triangle((x1+x3)/2, (x3+x2)/2, x3, (y1+y3)/2, (y3+y2)/2, y3,0,1);

triangle(x1, (x1+x2)/2, (x1+x3)/2, y1, (y1+y2)/2, (y1+y3)/2,1,0);

triangle((x1+x2)/2, x2, (x3+x2)/2, (y1+y2)/2, y2, (y3+y2)/2,0,0);

}

}

void st(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

//triangle(0,200,400,0,200,0,1,1);

glBegin(GL\_TRIANGLES);

glColor3f(1,1,1);

glVertex2f(0,0);

glVertex2f(200,200);

glVertex2f(400,0);

glEnd();

triangle(0,200,400,0,200,0,1,1);

glFlush();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(1000,1000);

glutCreateWindow("akansha sierpinski triangle");

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

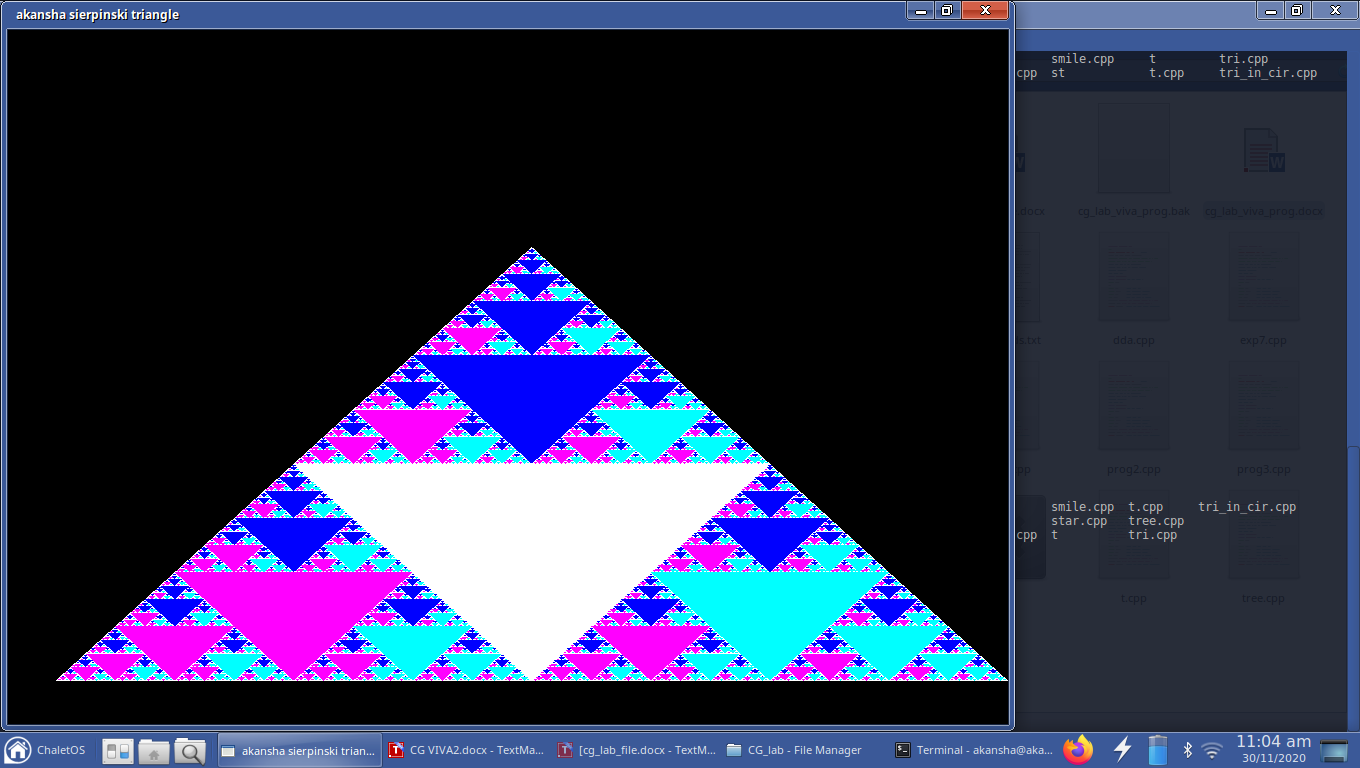
gluOrtho2D(-20,500,-20,500);

glutDisplayFunc(st);

glutMainLoop();

return 0;

}



**18. PATTERN (CIRCLES)**

#include<GL/glut.h>

#include<GL/gl.h>

#include<cmath>

void draw(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

float x=250,r=5;

glPointSize(1);

for(int j=10;j>0;j--)

{

glBegin(GL\_POINTS);

for(int i=0;i<360;i++)

{

glVertex3f(x+r\*cos(3.14159\*i/180),250+r\*sin(3.14159\*i/180),0);

}

glEnd();

if(j>0){

x=x-j;

r=r+j;

}

else{

x=x+j;

r=r+j;

}

}

for(int j=1;j<10;j++)

{

glBegin(GL\_POINTS);

for(int i=0;i<360;i++)

{

glVertex3f(x+r\*cos(3.14159\*i/180),250+r\*sin(3.14159\*i/180),0);

}

glEnd();

if(j>0){

x=x+j;

r=r+j;

}

else{

x=x+j;

r=r+j;

}

}

glFlush();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

glutCreateWindow("Akansha");

//init();

//glClearColor(0,0,0,1);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

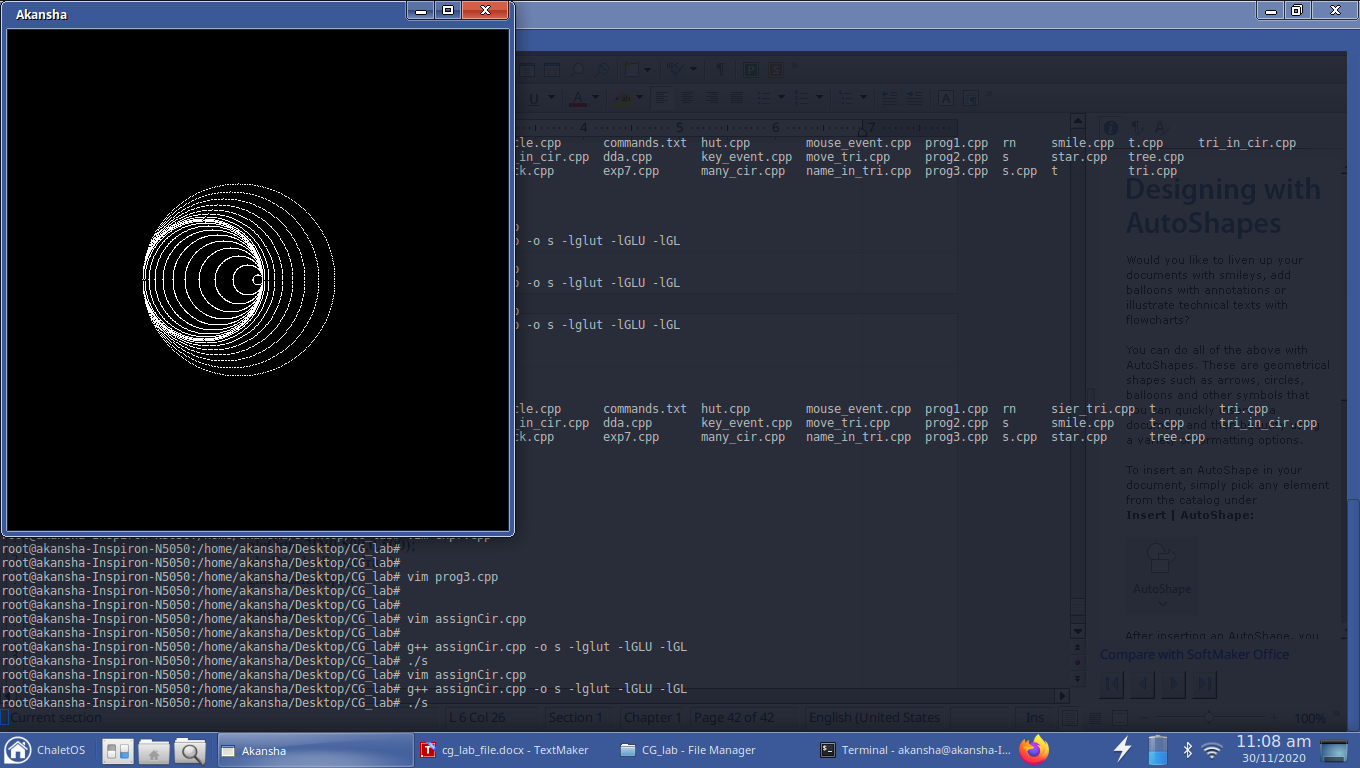
gluOrtho2D(0,500,0,500);

glutDisplayFunc(draw);

glutMainLoop();

return 0;

}



**19. MOVING CHARACTER WITH KEYBOARD ARROWS**

#include<GL/glut.h>

#include<GL/gl.h>

#include<iostream>

#include<cmath>

#include<string>

int x1=0,yy=0;

void characterDraw(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

glTranslatef(x1, yy, 0);

glColor3f(1,1,0);

glBegin(GL\_POLYGON);

for(int i=0;i<360;i++)

{

glVertex3f(15\*cos(3.14159\*i/180),65+15\*sin(3.14159\*i/180),0);

}

glEnd();

glBegin(GL\_TRIANGLES);

glColor3f(1,0,0);

glVertex2f(0,50);

glVertex2f(-30,0);

glVertex2f(30,0);

glEnd();

glBegin(GL\_LINES);

glColor3f(1,1,0);

glVertex2f(-20,-20);

glVertex2f(-5,0);

glEnd();

glBegin(GL\_LINES);

glColor3f(1,1,0);

glVertex2f(20,-20);

glVertex2f(5,0);

glEnd();

glFlush();

}

void ch\_move(int key,int x,int y)

{

switch (key) {

case GLUT\_KEY\_LEFT:

x1 = x1-10;yy=0;

break;

case GLUT\_KEY\_RIGHT:

x1 = x1+10;yy=0;

break;

case GLUT\_KEY\_UP:

yy = yy+10;x1=0;

break;

case GLUT\_KEY\_DOWN:

yy = yy-10;x1=0;

break;

}

glutPostRedisplay();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

glutCreateWindow("CHARACTER MOTION");

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(-300,300,-300,300);

glutDisplayFunc(characterDraw);

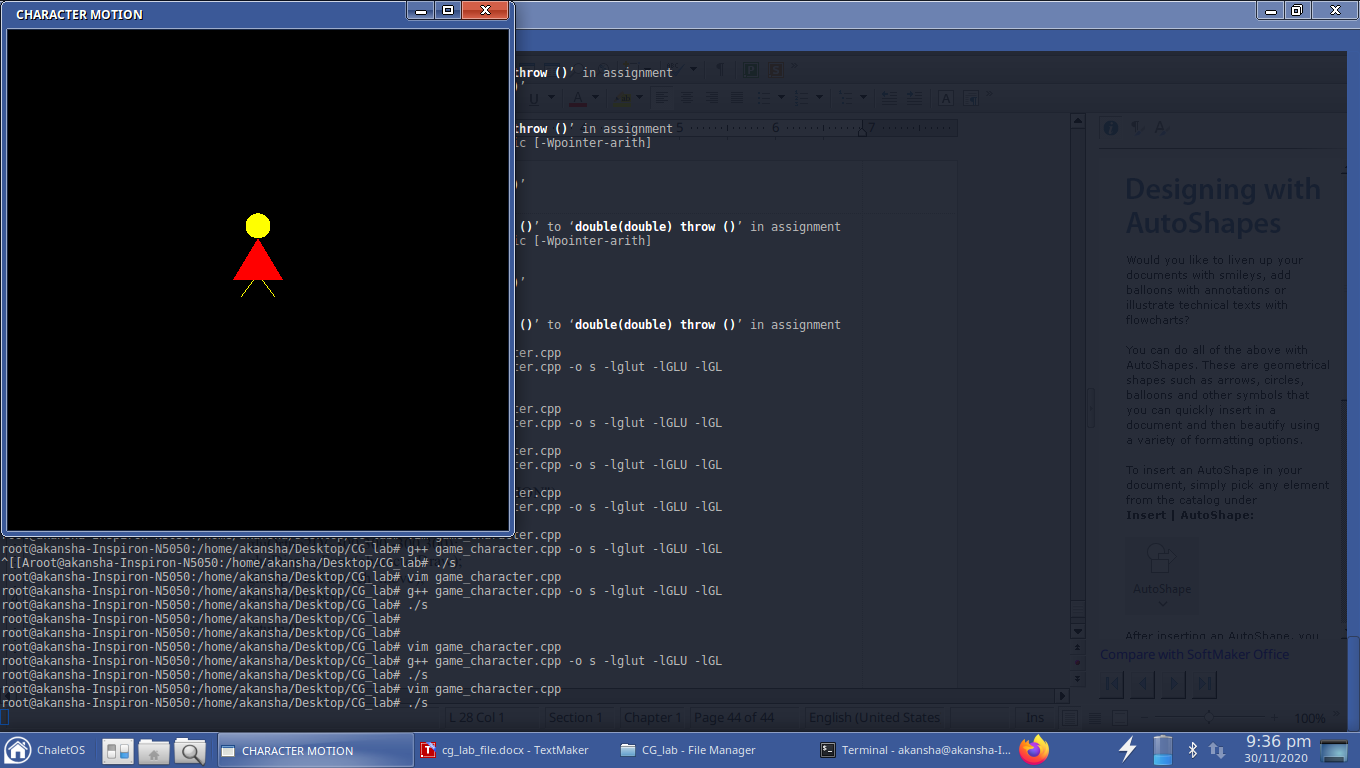
glutSpecialFunc(ch\_move);

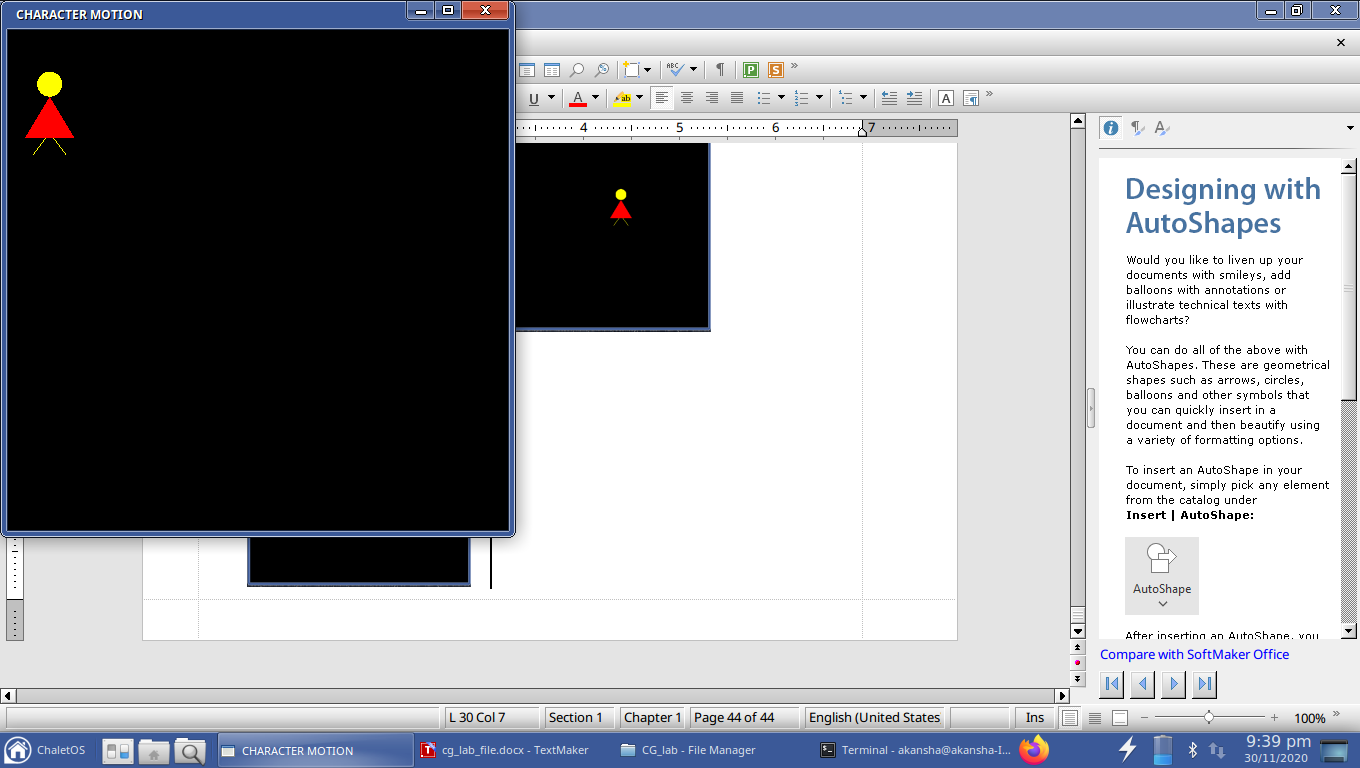
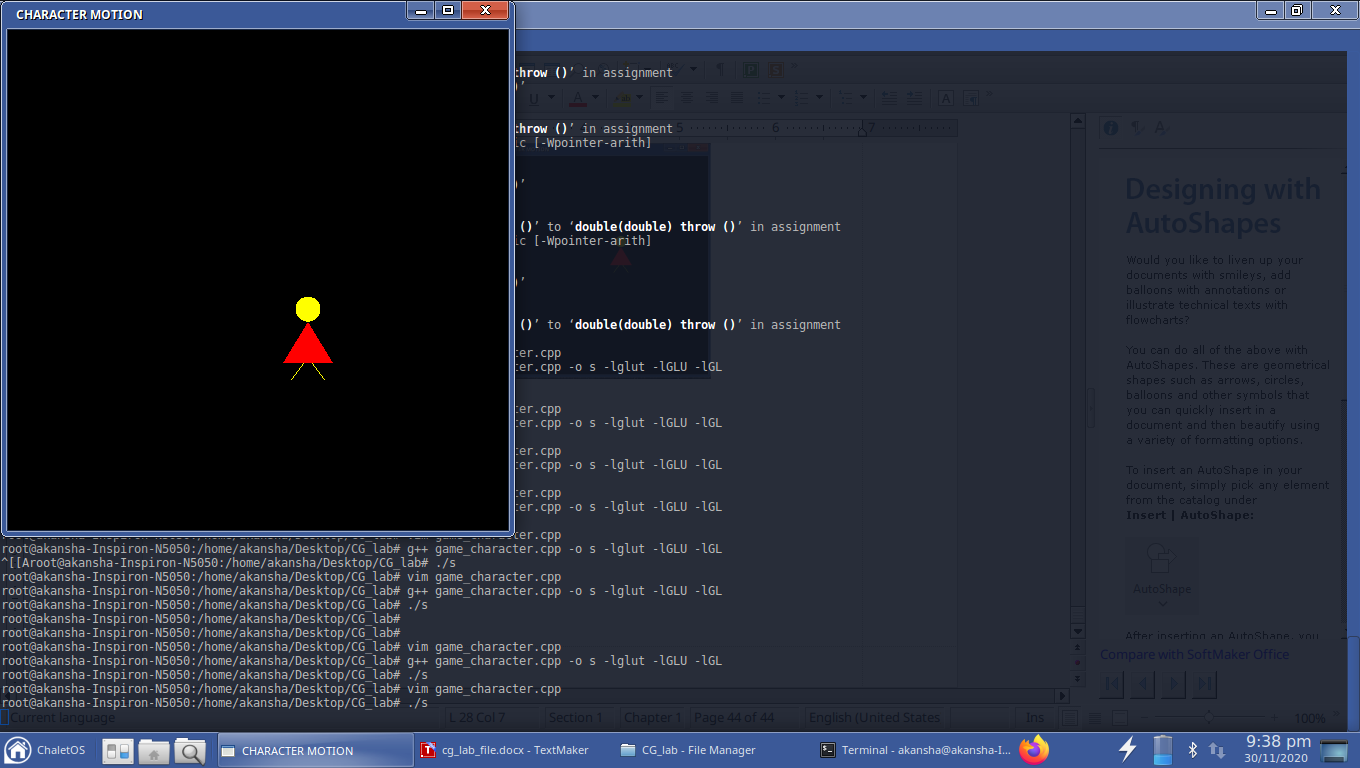
glutMainLoop();

return 0;

}

**Character is shifted to different positions by pressing keyboard arrow keys:**





**20. CLOCK**

#include<GL/glut.h>

#include<GL/gl.h>

#include<cmath>

#include <unistd.h>

int x=200,y=120;

void line(void){

glBegin(GL\_LINES);

glVertex2f(200,200);

glVertex2f(x,y);

glEnd();

}

void drawstring(char v,float px,float py)

{

glRasterPos3f(px,py,0);

glColor3f(1,1,0);

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15,v);

}

void drawstr2(char \*a,float px,float py)

{

char\* c=a;

glRasterPos3f(px,py,0);

for(int i=0;i<=1;c++)

{ i++;glColor3f(1,1,0);

glutBitmapCharacter(GLUT\_BITMAP\_9\_BY\_15,\*c);

}

}

void draw2(void){

glClear(GL\_COLOR\_BUFFER\_BIT);

//glPointSize(2);

glColor3f(1,1,0);

glBegin(GL\_POINTS);

for(int i=0;i<360;i++)

{

glVertex3f(200+150\*cos(3.14159\*i/180),200+150\*sin(3.14159\*i/180),0);

}

glEnd();

char tm[9]={'3','2','1','9','8','7','6','5','4'};

char ar[3][2]={{'1','2'},{'1','1'},{'1','0'}};

int j=0,k=-1;

for(int i=0;i<360;i++)

{

if(i%30==0){

if(k==0 || k==1 || k==2)

{

drawstr2(ar[k],200+125\*cos(3.14159\*i/180),200+125\*sin(3.14159\*i/180));

k++; }else{ drawstring(tm[j],200+125\*cos(3.14159\*i/180),200+125\*sin(3.14159\*i/180));

if(tm[j]=='1')

{k++;}

j++;

}

}

}

line();

glFlush();

}

void m(){

while(1){

for(int i=360;i>0;i--)

{

x=200+100\*cos(3.14159\*i/180);y=200+100\*sin(3.14159\*i/180);

draw2();

usleep(100000);

}

}

}

void draw()

{

draw2();

m();

}

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

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE);

glutInitWindowSize(500,500);

glutCreateWindow("CLOCK");

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,500,0,500);

glutDisplayFunc(draw);

glutMainLoop();

return 0;

}

**The clock needle is rotating in clockwise direction.**

**Different positions of moving needle:**

