ASSIGNMENT 1

#include<iostream>

#include<GL/glut.h>

using namespace std;

int c=100,d=100,a=10,b=10;

int term =1;

void inti()

{

glPointSize(2);

cout<<"\n\n\n..::MENU::..";

cout<<"\n\n1. BRESALHAM's ALGORITHM\n2.DDA ALGORITHM\n\n";

cin>>term;

cout<<"ENTER THE BOTTOM MOST POINT ";

cin>>a>>b;

cout<<"ENTER THE TOP MOST POINT ";

cin>>c>>d;

glClearColor(0.5f,0.5,0,1);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0, 500, 0, 500);

}

int maximum(int a,int b)

{

if(a>b)

return a;

return b;

}

void pixelizer(int x,int y)

{

glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();

}

void line2(int x1,int y1,int x2,int y2)

{ int dx = x2-x1;

int dy = y2-y1;

float xinc = (float)dx / maximum(abs(dx),abs(dy));

float yinc = (float)dy / maximum(abs(dx),abs(dy));

float x = x1;

float y = y1;

pixelizer(x1,y1);

for(int i=1 ; i < maximum(abs(dx),abs(dy)) ; i++)

{ x=x+xinc;

y=y+yinc;

x1=x+0.5;

y1=y+0.5;

pixelizer(x1,y1);

}

}

void line(int x1,int y1,int x2,int y2)

{ int x = x1;

int y = y1;

int dx = abs(x2-x1);

int dy = abs(y2 - y1);

int sx=1,sy=1;

if(x2-x1 < 0 )

sx=-1;

if(y2-y1 < 0)

sy = -1;

int g =0,i=1;

if(dx > dy)

{ pixelizer(x,y);

g = (2\*dy) - dx;

while(i<dx)

{if(g>=0)

{ x=x+sx;

y=y+sy;

g=g+ 2\*(dy-dx);}

else

{

x=x+sx;

g=g+(2\*dy);

}

i++;

pixelizer(x,y);

}

}

else

{ pixelizer(x,y);

g = (2\*dx) - dy;

while(i<dy)

{

if(g>=0)

{

y=y+sy;

x=x+sx;

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

}

else

{

y=y+sy;

g=g+(2\*dx);

}

i++;

pixelizer(x,y);

}

}

void rectangled(int a,int b,int c,int d)

{system("color f0");

if(term==1)

{

line(a,b,a,d);line(a,d,c,d);line(c,d,c,b); line(c,b,a,b);

line(a,(b+d)/2,(a+c)/2,d);line((a+c)/2,d,c,(d+b)/2);

line(c,(d+b)/2,(c+a)/2,b);line((c+a)/2,b,a,(b+d)/2);

line((3\*a+c)/4 , (3\*d+b)/4 ,(3\*c+a)/4 , (3\*d+b)/4);

line((3\*c+a)/4 , (3\*d+b)/4 , (3\*c+a)/4 , (3\*b+d)/4);

line( (3\*c+a)/4 ,(3\*b+d)/4 , (3\*a+c)/4 , (3\*b+d)/4);

line ((3\*a+c)/4 , (3\*b+d)/4 ,(3\*a+c)/4 , (3\*d+b)/4);

}

if(term==2)

{line2(a,b,a,d);line2(a,d,c,d);line2(c,d,c,b); line2(c,b,a,b);

line2(a,(b+d)/2,(a+c)/2,d);line2((a+c)/2,d,c,(d+b)/2);

line2(c,(d+b)/2,(c+a)/2,b);line2((c+a)/2,b,a,(b+d)/2);

line2((3\*a+c)/4 , (3\*d+b)/4 ,(3\*c+a)/4 , (3\*d+b)/4);

line2((3\*c+a)/4 , (3\*d+b)/4 , (3\*c+a)/4 , (3\*b+d)/4);

line2( (3\*c+a)/4 ,(3\*b+d)/4 , (3\*a+c)/4 , (3\*b+d)/4);

line2 ((3\*a+c)/4 , (3\*b+d)/4 ,(3\*a+c)/4 , (3\*d+b)/4)}}

void abhimanyu()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

rectangled(a,b,c,d);

glFlush();

}

int main(int a,char \*b[])

{

system("Color f0");

glutInit(&a,b);

glutInitWindowSize(720,720);

glutInitWindowPosition(1,1);

glutInitDisplayMode(GLUT\_RGB|GLUT\_SINGLE);

glutCreateWindow("ASSIGNMENT 1");

inti();

glutDisplayFunc(abhimanyu);

glutMainLoop();

return 0;

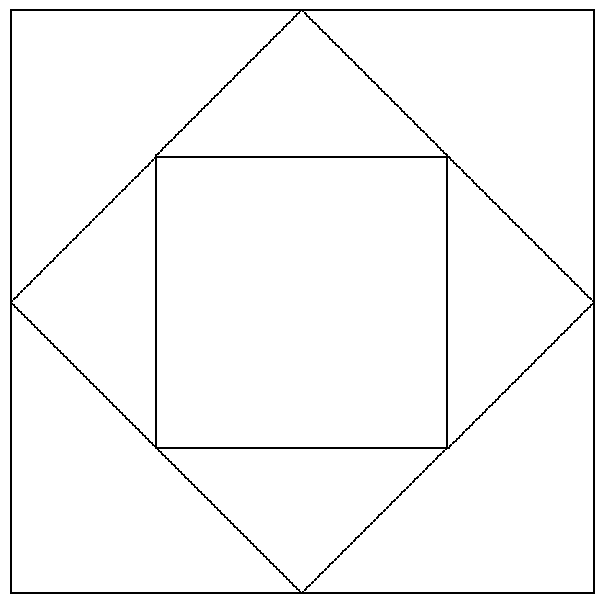
}

INPUT:

ENTER BOTTOM POINT – 45 45

ENTER TOP MOST POINT -450 450

OUTPUT:



ASSIGNMENT 2

#include<iostream>

#include<conio.h>

#include<stdlib.h>

#include<math.h>

#include<GL/glut.h>

using namespace std;

int tx,ty,angle,length;

int init()

{ cout<<"ENTER THE INITAL POINT x - ";

cin>>tx;

cout<<"\nENTER THE INITAL POINT Y - ";

cin>>ty;

cout<<"\nENTER THE LENGTH OF TRRIANGLE's SIDE - ";

cin>>length;

cout<<"\nENTER THE ANGLE OF ROTATED TRIANGLE - ";

cin>>angle;

glClearColor(1,1,1,1);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,720,0,720);

glMatrixMode(GL\_MODELVIEW);

glViewport(0,0,720,720);

return 0;

}

int pixelizer(int x,int y)

{ glColor3f(0,0,0);

glBegin(GL\_POINTS);

glVertex2f(x,y);

glEnd();

return 0;

}

int DDA(int x1,int y1,int x2,int y2)

{ float dx = x2-x1;

float dy = y2-y1;

float step = max (abs(dx),abs(dy));

dx=dx/step;

dy=dy/step;

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

pixelizer(x1+(dx)\*i,y1+(dy)\*i);

return 0;}

int breselham(int radius,int cx,int cy)

{int x=0;

int y=radius;

int s = 3 - (2\*radius);

while(y>x)

{if(s>=0)

{ x=x+1;

s = s+(4\*(x-y))+10;

y=y-1;}

else

{ x=x+1;

s = s+(4\*x)+6;}

pixelizer(x+cx,y+cy); pixelizer(-x+cx,y+cy); pixelizer(-x+cx,-y+cy);

pixelizer(x+cx,-y+cy);pixelizer(y+cx,x+cy); pixelizer(y+cx,-x+cy);

pixelizer(-y+cx,-x+cy); pixelizer(-y+cx,x+cy);

}

return 0;

}

void abhimanyu()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

DDA(tx,ty,(length)\*(cos(angle\*3.14/180))+tx,length\*(sin(angle\*3.14/180))+ty);

DDA((length)\*(cos(angle\*3.14/180))+tx,length\*(sin(angle\*3.14/180))+ty,(length)\*(cos((60+angle)\*3.14/180))+tx,length\*(sin((angle+60)\*3.14/180))+ty);

DDA((length)\*(cos((60+angle)\*3.14/180))+tx,length\*(sin((angle+60)\*3.14/180))+ty,tx,ty);

int centerx,centery;

centerx = ((length/2)/(cos(30\*3.14/180)))\*(cos((angle+30)\*3.14/180))+tx;

centery = ((length/2)/(cos(30\*3.14/180)))\*(sin((angle+30)\*3.14/180))+ty;

breselham((length/2)/(cos(30\*3.14/180)),centerx,centery);

breselham((length/4)/(cos(30\*3.14/180)),centerx,centery);

glutSwapBuffers();

}

int main(int b,char \*c[])

{

system("color f0");

glutInit(&b,c);

glutInitDisplayMode(GLUT\_DOUBLE|GLUT\_RGBA|GLUT\_DEPTH);

glutInitWindowSize(720,720);

glutInitWindowPosition(1,1);

glutCreateWindow("Abhimanyu' window");

init();

glutDisplayFunc(abhimanyu);

glutMainLoop();

return 0;

}

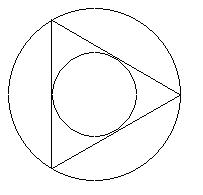
INPUT:-

ENTER ONE POINT OF TRIANGLE: - 55 55

ENTER THE LENGTH OF TRIANGLE:- 120

ENTER THE ANGLE OF ROTATION:- 30

OUTPUT:-



ASSIGNMENT 3

#include<iostream>

#include<list>

#include<GL/glut.h>

using namespace std;

list <int>newaxis;

int a=-1,b=-1,c=-1,d=-1,e=-1,f=-1;

int r=0;

float color[3] = {1,1,1};

void inti()

{

glPointSize(2);

glClearColor(0.5f,0.5,0,1);

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0, 720, 0, 720);

}

void pixelizer(int x,int y)

{ glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();}

int maximum(int a,int b)

{if(a>b)

return a;

return b;

}

void line(int x1,int y1,int x2,int y2)

{ int dx = x2-x1;

int dy = y2-y1;

float xinc = (float)dx / maximum(abs(dx),abs(dy));

float yinc = (float)dy / maximum(abs(dx),abs(dy));

float x = x1;

float y = y1;

pixelizer(x1,y1);

for(int i=1 ; i < maximum(abs(dx),abs(dy)) ; i++)

{ x=x+xinc;

y=y+yinc;

x1=x+0.5;

y1=y+0.5;

pixelizer(x1,y1);

}

}

void abhimanyu()

{glBegin(GL\_TRIANGLES);

glColor3f(1,0,0);

glVertex2i(0,0);

glColor3f(0,1,0);

glVertex2i(50,50);

glColor3f(0,0,1);

glVertex2i(100,0);

glEnd();

glColor3f(color[0],color[1],color[2]);

if(r==1)

{ a=newaxis.front();

newaxis.pop\_front();

b=newaxis.front();

newaxis.pop\_front();

system("cls");

cout<<a<<" "<<b<<endl;

e=a;

f=b;

while(!newaxis.empty())

{ c= newaxis.front();

newaxis.pop\_front();

d=newaxis.front();

newaxis.pop\_front();

cout<<c<<" "<<d<<endl;

line(a,b,c,d);

a=c;

b=d;

}

line(a,b,e,f);

glFlush();

}

void colorchoser(int x,int y)

{

glReadPixels(x,720-y,1,1,GL\_RGB,GL\_FLOAT,color);

cout<<x<<" "<<y<<" "<<color[0]<<" "<<color[1]<<" "<<color[2];

}

void clickings(int ButtonNo , int ON\_OR\_OFF ,int xaxis , int yaxis )

{

if(ON\_OR\_OFF == GLUT\_DOWN )

{

if( ButtonNo == GLUT\_RIGHT\_BUTTON)

{

if(xaxis<=100 && 720-yaxis <=100)

{

colorchoser(xaxis,yaxis);

goto ab;

}

r=0;

a=xaxis;

b=yaxis;

cout<<"ENTER - "<<a<<" "<<720-b<<endl;

pixelizer(a,720-b);

c=a;

d=720-b;

newaxis.push\_back(a);

newaxis.push\_back(720-b);

cout<<newaxis.size();

}

}

if(ON\_OR\_OFF == GLUT\_DOWN )

{

if(ButtonNo == GLUT\_LEFT\_BUTTON)

r=1;

}

ab:

glutPostRedisplay();

}

int main(int a,char \*b[])

{

system("Color f0");

glutInit(&a,b);

glutInitWindowSize(720,720);

glutInitWindowPosition(1,1);

glutInitDisplayMode(GLUT\_RGB|GLUT\_SINGLE);

glutCreateWindow("ASSIGNMENT 3");

inti();

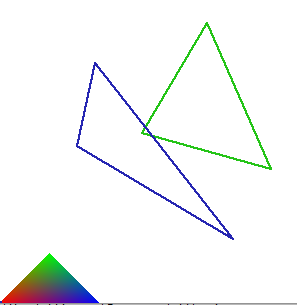
glutDisplayFunc(abhimanyu);

glutMouseFunc (clickings);

glutMainLoop();

return 0;

}



ASSIGNMENT 04

#include<iostream>

#include<GL/glut.h>

#include<math.h>

using namespace std;

int putpixel(int x,int y)

{ glColor3f(0,0,0);

glPointSize(2);

glBegin(GL\_POINTS);

glVertex2f(x,y);

glEnd();

return 0;

}

int bresalham(int x1,int y1,int x2,int y2)

{ cout<<"ENTERD";

int dx = abs(x2-x1);

int dy = abs(y2-y1);

int signx,signy;

if(x2>=x1)

signx =1;

else

signx =-1;

if(y2>=y1)

signy =1;

else

signy =-1;

int s=0;

if(dx>=dy)

{

s = 2\*dy-dx;

putpixel(x1,y1);

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

{

if(s>=0)

{

x1=x1+signx;

y1 = y1+signy;

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

}

else

{

x1=x1+signx;

s = s+2\*(dy);

}

putpixel(x1,y1);}}

if(dx<dy)

{s = 2\*dx-dy;

putpixel(x1,y1);

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

{if(s>=0)

{ y1=y1+signy;

x1 = x1+signx;

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

}

else

{

y1=y1+signy;

s = s+2\*(dx);} putpixel(x1,y1);}}

return 0;

}

int color[3];

int floodfiller(int x,int y)

{

glReadPixels(x,y,1,1,GL\_RGB,GL\_INT,color);

if(color[0]==color[1]&&color[1]==color[2]&&color[2]==0)

return 0;

putpixel(x,y);

cout<<color[0]<<endl;

floodfiller(x+1,y);

floodfiller(x-2,y);

floodfiller(x,y+1);

floodfiller(x,y-2);

return 0;

}

void abhimanyu()

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

glColor3f(0,0,0);

int a=0;

for(int i=0;i<=4;i++)

{

bresalham(75\*i\*cos(45\*3.14/180)-a\*sin(45\*3.14/180)+250,75\*i\*cos(45\*3.14/180)+a\*sin(45\*3.14/180),75\*i\*cos(45\*3.14/180)-75\*4\*sin(45\*3.14/180)+250,75\*i\*cos(45\*3.14/180)+75\*4\*sin(45\*3.14/180));

bresalham(a\*cos(45\*3.14/180)-75\*i\*sin(45\*3.14/180)+250,a\*cos(45\*3.14/180)+75\*i\*sin(45\*3.14/180),75\*4\*cos(45\*3.14/180)-75\*i\*sin(45\*3.14/180)+250,75\*4\*cos(45\*3.14/180)+75\*i\*sin(45\*3.14/180));

}

floodfiller(250,250); floodfiller(350,250); floodfiller(150,250);

floodfiller(250,150);floodfiller(250,50); floodfiller(350,250);

floodfiller(250,350);floodfiller(150,150);floodfiller(350,150);

glutSwapBuffers();

void inti()

{ glClearColor(1,1,1,0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,720,0,720);

glMatrixMode(GL\_MODELVIEW);

glViewport(0,0,720,720);

}

int main(int a,char\*b[])

{

system("color f0");

glutInit(&a,b);

glutInitDisplayMode(GLUT\_RGBA|GLUT\_DOUBLE|GLUT\_DEPTH);

glutInitWindowSize(720,720);

glutInitWindowPosition(1,1);

glutCreateWindow("abhimanyu");

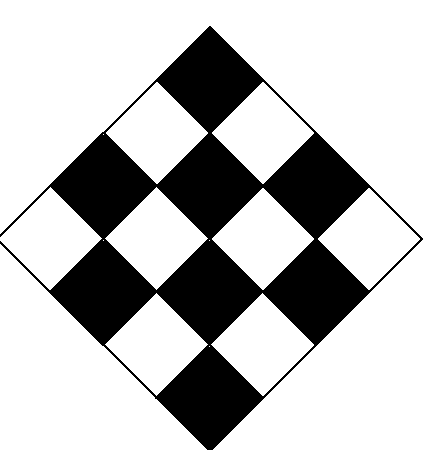
inti();

glutDisplayFunc(abhimanyu);

glutMainLoop();

return 0;

}



ASSIGNMENT 7

#include <iostream>

#include<GL/glut.h>

using namespace std;

const int w=500,h=500;

int flag=1,wi[4][2],wo[4][2],n=0,m=0;

int ymax, ymin, xmax, xmin;

void \*font[]={GLUT\_BITMAP\_8\_BY\_13,GLUT\_BITMAP\_HELVETICA\_18 };

typedef struct point{

int xc,yc;

}point;

point e[20],out[20] ;

void set(int x, int y)

{

glColor3f(1,0,0);

glPointSize(4);

glBegin(GL\_POINTS);

glVertex2f(x,y);

glEnd();

glFlush();

}

void abc()

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,w,0,h);

}

void menu()

{

glColor3f(1,1,1);

glRasterPos2f(100,480);

char m[]="Pls read carefully Instruction Before Starting:::";

for(int i=0;i<50;i++)

glutBitmapCharacter(font[1],m[i]);

glEnd();

glRasterPos2f(10,400);

char m1[]="1...First click by left button in upper window twice";

char m2[]="2...right click for window creating";

for(int i=0;i<55;i++)

glutBitmapCharacter(font[0],m1[i]);

glEnd();

glRasterPos2f(10,350);

for(int i=0;i<36;i++)

glutBitmapCharacter(font[0],m2[i]);

glEnd();

char m3[]="3.....Now draw polygon with left click...";

glRasterPos2f(10,300);

for(int i=0;i<45;i++)

glutBitmapCharacter(font[0],m3[i]);

glEnd();

char m4[]="START";

glRasterPos2f(253,255);

for(int i=0;i<6;i++)

glutBitmapCharacter(font[0],m4[i]);

glEnd();

}

void desplay()

{

glColor3f(1,1,0);

glPointSize(4);

glClearColor(0.25,.13,.16,0);

glClear(GL\_COLOR\_BUFFER\_BIT);

menu();

glColor3f(1,1,1);

glBegin(GL\_LINE\_LOOP);

glVertex2d(250,250);

glVertex2d(250,270);

glVertex2d(300,270);

glVertex2d(300,250);

glEnd();

}

void clip()

{float s;

int c=0;

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

{

if(e[i].xc<xmin&&e[i+1].xc>xmin)

{

s=(float)(e[i+1].yc-e[i].yc)/(e[i+1].xc-e[i].xc);

out[c].yc=e[i].yc+(int)(s\*(xmin-e[i].xc));

out[c].xc=xmin;

c++;

out[c].xc=e[i+1].xc;

out[c].yc=e[i+1].yc;

c++;

}

else if(e[i].xc>xmin&&e[i+1].xc>xmin)

{

out[c].xc=e[i+1].xc;

out[c].yc=e[i+1].yc;

c++;

}

else if(e[i].xc>xmin&&e[i+1].xc<xmin)

{

s=(float)(e[i+1].yc-e[i].yc)/(e[i+1].xc-e[i].xc);

out[c].yc=e[i].yc+(int)(s\*(xmin-e[i].xc));

out[c].xc=xmin;

c++;

}

}

out[c]=out[0];

m=c;

c=0;

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

{if(out[i].yc<ymin&&out[i+1].yc>ymin)

{

s=(float)(out[i+1].yc-out[i].yc)/(out[i+1].xc-out[i].xc);

e[c].xc=out[i].xc+(int)((ymin-out[i].yc)/s);

e[c].yc=ymin;

c++;

e[c].xc=out[i+1].xc;

e[c].yc=out[i+1].yc;

c++;

}

else if(out[i].yc>ymin&&out[i+1].yc>ymin)

{

e[c].xc=out[i+1].xc;

e[c].yc=out[i+1].yc;

c++;

}

else if(out[i].yc>ymin&&out[i+1].yc<ymin)

{

s=(float)(out[i+1].yc-out[i].yc)/(out[i+1].xc-out[i].xc);

e[c].xc=out[i].xc+(int)((ymin-out[i].yc)/s);

e[c].yc=ymin;

c++;

e[c]=e[0];

m=c;

c=0;

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

{

if(e[i].xc>xmax&&e[i+1].xc<xmax)

{

s=(float)(e[i+1].yc-e[i].yc)/(e[i+1].xc-e[i].xc);

out[c].yc=e[i].yc+(int)(s\*(xmax-e[i].xc));

out[c].xc=xmax;

c++;

out[c].xc=e[i+1].xc;

out[c].yc=e[i+1].yc;

c++;

}

else if(e[i].xc<xmax&&e[i+1].xc<xmax)

{

out[c].xc=e[i+1].xc;

out[c].yc=e[i+1].yc;

c++;

}

else if(e[i].xc<xmax&&e[i+1].xc>xmax)

{

s=(float)(e[i+1].yc-e[i].yc)/(e[i+1].xc-e[i].xc);

out[c].yc=e[i].yc+(int)(s\*(xmax-e[i].xc));

out[c].xc=xmax;

c++;

}

}

out[c]=out[0];

m=c;

c=0;

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

{

if(out[i].yc>ymax&&out[i+1].yc<ymax)

{

s=(float)(out[i+1].yc-out[i].yc)/(out[i+1].xc-out[i].xc);

e[c].xc=out[i].xc+(int)((ymax-out[i].yc)/s);

e[c].yc=ymax;

c++;

e[c].xc=out[i+1].xc;

e[c].yc=out[i+1].yc;

c++;

}

else if(out[i].yc<ymax&&out[i+1].yc<ymax)

{

e[c].xc=out[i+1].xc;

e[c].yc=out[i+1].yc;

c++;

}

else if(out[i].yc<ymax&&out[i+1].yc>ymax)

{

s=(float)(out[i+1].yc-out[i].yc)/(out[i+1].xc-out[i].xc);

e[c].xc=out[i].xc+(int)((ymax-out[i].yc)/s);

e[c].yc=ymax;

c++;

}

}

e[c]=e[0];

glColor3f(0,1,0);

glBegin(GL\_LINE\_LOOP);

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

glVertex2f(e[i].xc,e[i].yc-250);

glEnd();

glFlush();

}

void draw()

{

wi[1][0]=wi[2][0];

wi[1][1]=wi[0][1];

wi[3][0]=wi[0][0];

wi[3][1]=wi[2][1];

if(wi[0][0]>wi[1][0])

{

xmax=wi[0][0];

xmin=wi[1][0];

}

else

{xmax=wi[1][0];

xmin=wi[0][0];

}

void drawoutput()

{wo[0][0]=wi[0][0];

wo[1][0]=wi[1][0];

wo[2][0]=wi[2][0];

wo[3][0]=wi[3][0];

wo[0][1]=wi[0][1]-250;

wo[1][1]=wi[1][1]-250;

wo[2][1]=wi[2][1]-250;

wo[3][1]=wi[3][1]-250;

glRasterPos2f(180,480-250);

glColor3f(0,0,1);

glBegin(GL\_LINE\_LOOP);

for(int i=0; i<4;i++)

glVertex2f(wo[i][0],wo[i][1]);

glEnd();

glFlush();

}

void mine(int b,int s, int x, int y)

{if(s==GLUT\_DOWN)

{if(b==GLUT\_LEFT\_BUTTON)

{if(flag==1)

{wi[n][0]=x;

wi[n][1]=500-y;

n+=2;

set(x,500-y);

}

else

{e[m].xc=x;

e[m].yc=500-y;

set(x, 500-y);

glColor3f(1,1,1);

if(m>0)

{glBegin(GL\_LINES);

glVertex2f(e[m-1].xc,e[m-1].yc);

glVertex2f(e[m].xc, e[m].yc);

glEnd();

glFlush();

}

m++;

}

}

else if(b==GLUT\_RIGHT\_BUTTON)

{

if(flag==1)

{

draw();

flag=0;

}

else

{

glColor3f(1,1,1);

e[m].xc=e[0].xc;

e[m].yc=e[0].yc;

glBegin(GL\_LINES);

glVertex2f(e[m-1].xc, e[m-1].yc);

glVertex2f(e[m].xc,e[m].yc);

glEnd();

glFlush();

drawoutput();

clip();

}

}

}

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(w,h);

glutCreateWindow("rohit chahar: poly clip");\

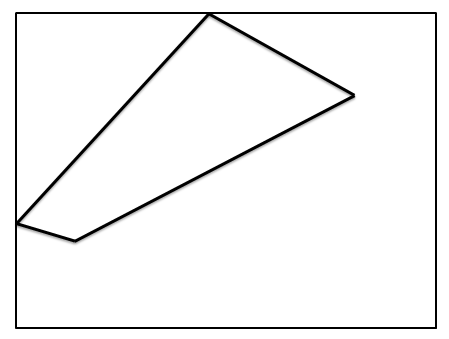
abc();

glutDisplayFunc(desplay);

glutMouseFunc(mine1);

glutMainLoop();

return 0;}



ASSIGNMENT 6

#include<iostream>

#include<math.h>

#include<GL/glut.h>

using namespace std;

int triangle =1;

int quad =0;

int active=0;

int input=0;

int xaxis[4],yaxis[4];

int xax[4],yax[4];

void shear(int x,int y)

{

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

{ xax[i] = xaxis[i] + x\*yaxis[i];

yax[i] = yaxis[i];

xax[i]=xax[i];

yax[i] =xax[i]\*y +yax[i];}}

void resize(int x,int y)

{

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

{ xax[i] = xaxis[i]\*x;

yax[i] = yaxis[i]\*y;

cout<<xax[i]<<endl;}}

void tran(int x,int y)

{for(int i=0;i<input;i++)

{xax[i] = xaxis[i]+x;

yax[i] = yaxis[i]+y;}}

void rota(int angle)

{float r = (angle\*3.14)/180;

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

{ xax[i] = xaxis[i]\*cos(r) - yaxis[i]\*sin(r) ;

yax[i] = xaxis[i]\*cos(r) + yaxis[i]\*sin(r) ;

}

}

void init()

{cout<<"\n\n\n..:: MENU::..\n\n1.TRIANGLE \n2.QUADRRILATERAL\n\n";

switch(getch())

{case '1':

triangle=1;

break;

case'2':

triangle=0;

quad=1;

break;

}

glClearColor(1,1,1,1);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,720,0,720);

glMatrixMode(GL\_MODELVIEW);

}

void menu()

{ cout<<"1. TRANSLATE\n2.ROTATE\n3.SHEAR\n4.RESIZE";

switch(getch())

{case'1':

cout<<"ENTER X ,y TO TRANSLATE - ";

int xx,yy;

cin>>xx>>yy;

tran(xx,yy);

break;

case'2':

cout<<"ENTER ANGLE OF ROTATION - ";

int r;

cin>>r;

rota(r);

break;

case'3':

cout<<"ENTER X ,y TO SHEAR - ";

int xx2,yy2;

cin>>xx2>>yy2;

shear(xx2,yy2);

break;

case'4':

cout<<"ENTER X ,y TO RESIZE - ";

int xx22,yy22;

cin>>xx22>>yy22;

resize(xx22,yy22);

break;

break;

}

cout<<endl;

}

void abhimanyu()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

if((quad==1 && input==4) || (input==3 &&triangle==1))

{glBegin(GL\_LINE\_LOOP);

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

glVertex2i(xaxis[i],yaxis[i]);

glEnd();

if(active==0)

{menu();

active=1;

}

}

if(active==1)

{glColor3f(1,0,0);

glLineWidth(3);

glBegin(GL\_LINE\_LOOP);

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

{glVertex2i(xax[i],yax[i]);}

glEnd();

}

glFlush();

}

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

{

y=720-y;

if(input==3 && triangle==1)

{

goto ab;

}

if(input==4 && quad==1)

{

goto ab;

}

if(button==GLUT\_RIGHT\_BUTTON && state==GLUT\_UP)

{

xaxis[input]=x;

yaxis[input++]=y;

}

ab:

cout<<endl;

}

int main(int b,char\* arg[])

{

system("color f0");

glutInit(&b,arg);

glutInitWindowSize(720,720);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGBA);

glutInitWindowPosition(150,1);

glutCreateWindow("DISPLAY CHART");

init();

glutMouseFunc(mouse);

glutDisplayFunc(abhimanyu);

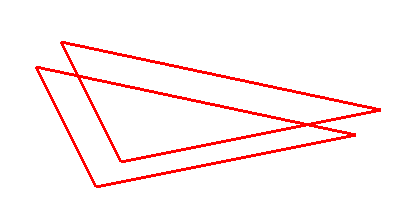
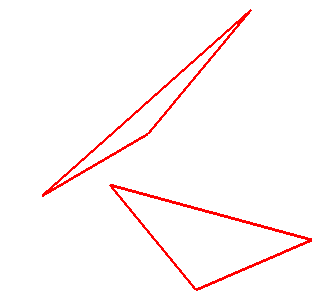
glutMainLoop();

return 0;

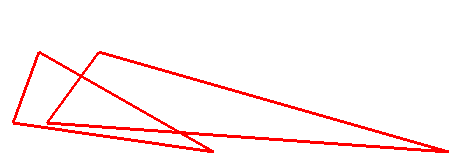
}

**OUTPUTS:**

1. **ROTATION ABOUT 15: 2.TRANSLATION by 20 20**

****

**3.RESIZE 2x**

****

**ASSIGNMENT 7**

#include <GL/glut.h>

#include <bits/stdc++.h>

using namespace std;

GLfloat angle= 0.0;

void spin (void)

{angle+= 1.0;

glutPostRedisplay();

}

void display(void)

{

glClearColor(0.3,0.7,0.3,0.0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glLoadIdentity();

gluLookAt(0.0,0.0,5.0,0.0,0.0,0.0,0.0,1.0,0.0);

glRotatef(angle, 0, 1, 0);

glutWireCube(2.0);

glutSwapBuffers();

}

void reshape(int width,int height)

{

glViewport(0,0,(GLsizei) width,(GLsizei) height);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60,(GLfloat) width/(GLfloat) height,1.0,100.0);

glMatrixMode(GL\_MODELVIEW);

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_DOUBLE);

glutInitWindowSize(500,500);

glutInitWindowPosition(200,100);

glutCreateWindow("assign7");

glutDisplayFunc(display);

glutKeyboardFunc(keyboard);

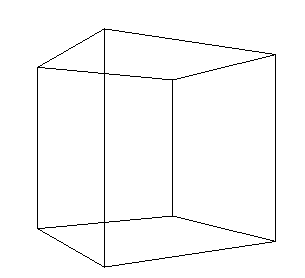
glutReshapeFunc(reshape);

glutIdleFunc(spin);

glutMainLoop();

return 0;

}



ASSIGNMENT 8

**PENDULAM**

#include<iostream>

#include<GL/glut.h>

using namespace std;

void init()

{ glClearColor(0,0.7,1,1);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(1,720,1,720);

glMatrixMode(GL\_MODELVIEW);

}float inc=0.041;

float angle=0;

void abhimanyu()

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

glLoadIdentity();

glColor3f(1,0.6,0);

glBegin(GL\_POLYGON);

glVertex2f(270,480-70);

glVertex2f(270,190-70);

glVertex2f(360,100-70);

glVertex2f(450,190-70);

glVertex2f(450,480-70);

glEnd();

glColor3f(0,0,0);

glLineWidth(10);

glBegin(GL\_LINE\_LOOP);

glVertex2f(270,480-70);

glVertex2f(270,190-70);

glVertex2f(360,100-70);

glVertex2f(450,190-70);

glVertex2f(450,480-70);

glEnd();

glLoadIdentity();

if(angle>225)

{ angle=225;

inc=-inc; }

if(angle<135)

{ angle=135;

inc=-inc; }

angle+=inc;

double radian=angle\*3.14/180;

float y2=75\*cos((double) radian);

float x2=75\*sin((double) radian);

glColor3f(0.0,0.0,0.0);

glLineWidth(2);

glBegin(GL\_POLYGON);

{ glVertex2f((720/2)-5,(720/2)+(720/4)-25);

glVertex2f((720/2)-5+x2,(720/2)+(720/4)-450+y2);

glVertex2f((720/2)+5+x2,(720/2)+(720/4)-450+y2);

glVertex2f((720/2)+5,(720/2)+(720/4)-25);

}

glEnd();

glBegin(GL\_POLYGON);

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

{ float degInRad = i\*(3.14/180);

glVertex2f(cos(degInRad)\*15+ 360+x2,sin(degInRad)\*15+110-25+y2);

}

glEnd();

glLoadIdentity();

glColor3f(0.5,0.0,0.1);

glBegin(GL\_POLYGON);

glVertex2f(360,690-150);

glVertex2f(200,590);

glVertex2f(200,420);

glVertex2f(360,340);

glVertex2f(360+160,420);

glVertex2f(360+160,590);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(1,1,0.99);

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

{ float degInRad = i\*(3.14/180);

glVertex2f(cos(degInRad)\*150+ (720/2),sin(degInRad)\*150+(720/2)+(720/4)-25);}

glEnd();

glLineWidth(10);

glBegin(GL\_LINE\_LOOP);

glColor3f(1,0.51,0.99);

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

{ glColor3f(0,0,0);

float degInRad = i\*(3.14/180);

glVertex2f(cos(degInRad)\*148+ (720/2),sin(degInRad)\*148+(720/2)+(720/4)-25);

}

glEnd();

glBegin(GL\_LINES);

glVertex2i(720/2,(720/2)+(720/4)-25);

glVertex2i((720\*3)/4.6,(720/2)+(720/4)-25);

glEnd();

glBegin(GL\_LINE\_LOOP);

glVertex2i(720/2,(720/2)+(720/4)-25);

glVertex2i((720\*3)/4.6,(720/1.58));

glEnd();

glBegin(GL\_LINE\_LOOP);

glColor3f(1,1,0.99);

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

{ glColor3f(0,0,0);

float degInRad = i\*(3.14/180);

glVertex2f(cos(degInRad)\*128+ (720/2),sin(degInRad)\*128+(720/2)+(720/4)-25);

}

glEnd();

glBegin(GL\_LINES);

glVertex2f(150+ (720/2),(720/2)+(720/4)-25);

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

{

float degInRad = i\*(3.14/180);

glVertex2f(cos(degInRad)\*130+ (720/2),sin(degInRad)\*130+(720/2)+(720/4)-25);

}

glEnd();

glutSwapBuffers();

}

int main(int a,char \*b[])

{

system("color f0");

glutInit(&a,b);

glutInitDisplayMode(GLUT\_RGBA|GLUT\_DOUBLE);

glutInitWindowSize(720,720);

glutInitWindowPosition(1,1);

glutCreateWindow("ABHIMANYU");

init();

glutDisplayFunc(abhimanyu);

glutIdleFunc(abhimanyu);

glutMainLoop();

return 0;

}

**FLAG:**

#include<iostream>

#include<GL/glut.h>

using namespace std;

float p=-100,flag=0;

void init()

{ glClearColor(0,0.7,1,1);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(1,720,1,720);

glMatrixMode(GL\_MODELVIEW)}

void abhimanyu()

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

glLoadIdentity();

glColor3f(0,0.5,0);

glBegin(GL\_POLYGON);

glColor3f(0,0,0);

glVertex2f(139+35,0);

glVertex2f(139+35,354);

glVertex2f(142+35,357);

glVertex2f(142+35,0);

glEnd();

glLoadIdentity();

glTranslatef(0,p,0);

glBegin(GL\_POLYGON);

glColor3f(0.9,0.59,0);

glVertex2f(100+74,200+p);

glVertex2f(100+74,230+p);

glVertex2f(250+74,230+p);

glVertex2f(250+74,200+p);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(1,1,1);

glVertex2f(100+74,200-30+p);

glVertex2f(100+74,230-30+p);

glVertex2f(250+74,230-30+p);

glVertex2f(250+74,200-30+p);

glEnd();

glBegin(GL\_POLYGON);

glColor3f(0,1,0);

glVertex2f(100+74,200-30\*2+p);

glVertex2f(100+74,230-30\*2+p);

glVertex2f(250+74,230-30\*2+p);

glVertex2f(250+74,200-30\*2+p);

glEnd();

glBegin(GL\_LINE\_LOOP);

glColor3f(0,0,1);

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

{

float degInRad = i\*(3.14/180);

glVertex2f(cos(degInRad)\*15+245,sin(degInRad)\*15+185+p);

}

glEnd();

glBegin(GL\_LINES);

glVertex2i(230,185+p);

glVertex2i(260,185+p);

glVertex2i(245,170+p);

glVertex2i(245,200+p);

glVertex2i(236,174+p);

glVertex2i(256,198+p);

glVertex2i(255,174+p);

glVertex2i(236,198+p);

glEnd();

if(230+p >=286)

flag=1;

if(flag==0)

p=p+0.1;

glutSwapBuffers();

glutPostRedisplay();

}

int main(int a,char \*b[])

{

system("color f0");

glutInit(&a,b);

glutInitDisplayMode(GLUT\_RGBA|GLUT\_DOUBLE);

glutInitWindowSize(720,720);

glutInitWindowPosition(1,1);

glutCreateWindow("ABHIMANYU");

init();

glutDisplayFunc(abhimanyu);

glutIdleFunc(abhimanyu);

glutMainLoop();

return 0;

}

**VEHICLE LOCOMOTION:**

#include<iostream>

#include<GL/glut.h>

#include<math.h>

using namespace std;

void reshape(int w,int h)

{

glViewport(0,0,w,h);

glutPostRedisplay();

}

void init()

{

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LEQUAL);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,720,0,720);

glMatrixMode(GL\_MODELVIEW);

}

float x=-80;

float p=720,Q=50;

float P=720,q=30;

float n=-80,r=0;

float j=0;

void abhimanyu()

{ glClearColor(0,0.8-j/250,1-j/250,1);

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

r=r+0.1;

n=n+0.6;

glTranslatef(-n,sin(r\*3.14/180)\*200,0);

glColor3f(1,0,0);

glBegin(GL\_POLYGON);

glVertex3f(15+P,4+Q,0);

glVertex3f(15+P,33+Q,0);

glVertex3f(16+P,33+Q,0);

glVertex3f(17+P,33.5+Q,0);

glVertex3f(18+P,34+Q,0);glVertex3f(19+P,34.5+Q,0);glVertex3f(20+P,35+Q,0);

glVertex3f(55+P,35+Q,0);

glVertex3f(85+p,63+q,0);

glVertex3f(145+p,63+q,0);

glColor3f(0.6,0.6,0.6);

glVertex3f(195+p,35+q,0);

glVertex3f(222+p,34.5+q,0);

glVertex3f(223+p,34+q,0);

glVertex3f(224+p,33.5+q,0);

glVertex3f(225+p,33+q,0);

glColor3f(1,0,0);

glVertex3f(226+p,4+q,0);

glEnd();

glColor3f(0,0,0);

glBegin(GL\_POLYGON);

glVertex3f(56+p,30+Q-8,0);

glVertex3f(136+p-5,30+q-1,0);

glVertex3f(136+p-5,57+q+3,0);

glColor3f(1,1,1);

glVertex3f(86+p,57+q+3,0);

glEnd();

glColor3f(0,0,0);

glBegin(GL\_POLYGON);

glVertex3f(141+P,30+Q+6,0);

glVertex3f(141+p,57+q-25,0);

glColor3f(1,1,1);

glVertex3f(188+p,30+q-3,0);

glEnd();

glBegin(GL\_POLYGON);

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

{

float degInRad = i\*(3.14/180);

glColor3f(0,0,0);

glVertex3f(cos(degInRad)\*20+ p+75,sin(degInRad)\*20+q+5,0);

glColor3f(0.3,0.3,0.3);

glVertex3f(cos(degInRad)\*14+ p+75,sin(degInRad)\*14+q+5,0);

}

glEnd();

glEnd();

glLoadIdentity();

glTranslatef(j,-j-50,0);

glBegin(GL\_POLYGON);

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

{

float degInRad = i\*(3.14/180);

glColor3f(1,1,0);

glVertex3f(cos(degInRad)\*30+j+358,sin(degInRad)\*30+350,-0.5);

}

glEnd();

j=j+0.03;

x=-80;

glutSwapBuffers();

glutPostRedisplay();

}

int main(int b,char \*c[])

{

glutInit(&b,c);

glutInitWindowSize(720,720);

glutInitWindowPosition(1,1);

glutInitDisplayMode(GLUT\_RGBA|GLUT\_DOUBLE|GLUT\_DEPTH);

glutCreateWindow("abhimanyu'animation");

init();

glutReshapeFunc(reshape);

glutDisplayFunc(abhimanyu);

glutMainLoop();

return 0;

}

