**/\*WAP to implement 2D Transformation (Translation,Scaling,Rotation,Shearing & Reflection) in Cpp\*/**

**#include <iostream>**

**#include <cmath>**

**#include <graphics.h>**

**#define pi 3.141592654**

**using namespace std;**

**int i,choice,plot\_count=0;**

**float x,y,z=1.0,x\_n,y\_n;**

**float x\_p\_1,y\_p\_1,x\_p\_2,y\_p\_2,x\_p\_3,y\_p\_3;**

**float t\_x,t\_y;//translation**

**float x\_f,y\_f,s\_x,s\_y;//Scaling**

**float x\_r,y\_r,thetha;//Rotation**

**float x\_ref,y\_ref,sh\_x,sh\_y;//Shearing**

**float m,c;//Reflection**

**float a[3][3];**

**void matrix\_multiplication();**

**void create\_graphics()**

**{**

**initwindow(1366,768);**

**setlinestyle(0,0,1);**

**for(i=0; i<=1366; i++)**

**{**

**line(0,i,1366,i);**

**}**

**setcolor(LIGHTBLUE);**

**line(0,384,1365,384);**

**line(683,0,683,1365);**

**setcolor(LIGHTGRAY);**

**for(i=633; 0<=i; i=i-50)//633=683-50**

**{**

**line(i,0,i,768);**

**}**

**for(i=733; i<=1366; i=i+50)//733=683+50**

**{**

**line(i,0,i,768);**

**}**

**for(i=334; 0<=i; i=i-50)//334=384-50**

**{**

**line(0,i,1366,i);**

**}**

**for(i=434; i<=1365; i=i+50)//434=384+50**

**{**

**line(0,i,1365,i);**

**}**

**setcolor(GREEN);**

**setlinestyle(0,0,3);//x\_user=682+x\_user\*50, y\_user=383-y\_user\*50**

**line(782,183,982,33);//(682+50\*2,383-50\*4)&(682+50\*6,383-50\*7) or (2,4)&(6,7)**

**line(982,33,632,533);//(682+50\*6,383-50\*7)&(682-50\*1,383+50\*3) or (6,7)&(-1,-3)**

**line(632,533,782,183);//(682-50\*1,383+50\*3)&(682+50\*2,383-50\*4) or (-1,-3)&(2,4)**

**setcolor(BLUE);**

**}**

**void plotcount()**

**{**

**plot\_count++;**

**if (plot\_count==1)**

**{**

**x\_p\_1=x\_n;**

**y\_p\_1=y\_n;**

**}**

**if (plot\_count==2)**

**{**

**x\_p\_2=x\_n;**

**y\_p\_2=y\_n;**

**}**

**if (plot\_count==3)**

**{**

**x\_p\_3=x\_n;// (2,4),(6,7),(-1,-3)**

**y\_p\_3=y\_n;//(782,183),(982,33),(632,533)**

**cout<<"\n\n\t\t(2,4) maps to ("<<x\_p\_1<<","<<y\_p\_1<<")";**

**cout<<"\n\n\t\t(6,7) maps to ("<<x\_p\_2<<","<<y\_p\_2<<")";**

**cout<<"\n\n\t\t(-1,-3) maps to ("<<x\_p\_3<<","<<y\_p\_3<<")";**

**x\_p\_1=682+x\_p\_1\*50;**

**y\_p\_1=383-y\_p\_1\*50;**

**x\_p\_2=682+x\_p\_2\*50;**

**y\_p\_2=383-y\_p\_2\*50;**

**x\_p\_3=682+x\_p\_3\*50;**

**y\_p\_3=383-y\_p\_3\*50;**

**setlinestyle(0,0,3);**

**line(x\_p\_1,y\_p\_1,x\_p\_2,y\_p\_2);**

**line(x\_p\_2,y\_p\_2,x\_p\_3,y\_p\_3);**

**line(x\_p\_3,y\_p\_3,x\_p\_1,y\_p\_1);**

**setcolor(RED);**

**setlinestyle(0,0,1);**

**line(x\_p\_1,y\_p\_1,782,183);**

**line(x\_p\_2,y\_p\_2,982,33);**

**line(x\_p\_3,y\_p\_3,632,533);**

**plot\_count=0;**

**}**

**}**

**void translation()**

**{**

**x\_n=x+t\_x;**

**y\_n=y+t\_y;**

**plotcount();**

**}**

**void scaling()**

**{**

**a[0][0]=s\_x;**

**a[0][1]=0;**

**a[0][2]=x\_f\*(1-s\_x);**

**a[1][0]=0;**

**a[1][1]=s\_y;**

**a[1][2]=y\_f\*(1-s\_y);**

**a[2][0]=0;**

**a[2][1]=0;**

**a[2][2]=1;**

**matrix\_multiplication();**

**plotcount();**

**}**

**void rotation()**

**{**

**a[0][0]=cos(thetha);**

**a[0][1]=-sin(thetha);**

**a[0][2]=x\_r\*(1-cos(thetha))+y\_r\*sin(thetha);**

**a[1][0]=sin(thetha);**

**a[1][1]=cos(thetha);**

**a[1][2]=y\_r\*(1-cos(thetha))-x\_r\*sin(thetha);**

**a[2][0]=0;**

**a[2][1]=0;**

**a[2][2]=1;**

**matrix\_multiplication();**

**plotcount();**

**}**

**void shearing()**

**{**

**a[0][0]=1;**

**a[0][1]=sh\_x;**

**a[0][2]=-sh\_x\*y\_ref;**

**a[1][0]=sh\_y;**

**a[1][1]=1;**

**a[1][2]=-sh\_y\*x\_ref;**

**a[2][0]=0;**

**a[2][1]=0;**

**a[2][2]=1;**

**matrix\_multiplication();**

**plotcount();**

**}**

**void reflection()**

**{**

**a[0][0]=(1-m\*m)/(1+m\*m);**

**a[0][1]=2\*m/(1+m\*m);**

**a[0][2]=-2\*c\*m/(1+m\*m);**

**a[1][0]=2\*m/(1+m\*m);**

**a[1][1]=(m\*m-1)/(1+m\*m);**

**a[1][2]=2\*c/(1+m\*m);**

**a[2][0]=0;**

**a[2][1]=0;**

**a[2][2]=1;**

**matrix\_multiplication();**

**plotcount();**

**}**

**void matrix\_multiplication()**

**{**

**x\_n=x\*a[0][0]+y\*a[0][1]+z\*a[0][2];**

**y\_n=x\*a[1][0]+y\*a[1][1]+z\*a[1][2];**

**//z\_n=x\*a[2][0]+y\*a[2][1]+z\*a[2][2];**

**}**

**int main()**

**{**

**int i;**

**while(1)**

**{**

**//cout<<"\n\n\n\t\t\t\t\t1366\*768 ";//origin = (682,383)**

**cout<<"\n\n\n\t\t\t\t\t(-13.66,0),(13.66,0),(0,-7.66),(0,7.66) ";**

**cout<<"\n\n\n\t\t Enter type of 2D Transformation.";**

**cout<<"\n\n\t\t 1. Translation";**

**cout<<"\n\n\t\t 2. Scaling";**

**cout<<"\n\n\t\t 3. Rotation";**

**cout<<"\n\n\t\t 4. Shearing";**

**cout<<"\n\n\t\t 5. Reflection\n\n";**

**cout<<"\n\n\t\t Enter your choice: ";**

**cin>>choice;**

**if (choice==1)**

**{**

**cout<<"\n\n\t\t Translation ";**

**cout<<"\n\n\t\t Enter translation in x direction t\_x : ";**

**cin>>t\_x;**

**cout<<"\n\n\t\t Enter translation in y direction t\_y : ";**

**cin>>t\_y;**

**create\_graphics();**

**x=2;**

**y=4;**

**translation();**

**x=6;**

**y=7;**

**translation();**

**x=-1;**

**y=-3;**

**translation();**

**}**

**if (choice==2)**

**{**

**cout<<"\n\n\t\t Scaling ";**

**cout<<"\n\n\t\t Enter center of scaling x\_f : ";**

**cin>>x\_f;**

**cout<<"\n\n\t\t Enter center of scaling y\_f : ";**

**cin>>y\_f;**

**cout<<"\n\n\t\t Enter s\_x : ";**

**cin>>s\_x;**

**cout<<"\n\n\t\t Enter s\_y : ";**

**cin>>s\_y;**

**create\_graphics();**

**setcolor(RED);//mark new center as red**

**setlinestyle(0,0,8);**

**line(682+x\_f\*50,383-y\_f\*50,682+x\_f\*50,383-y\_f\*50);**

**setcolor(BLUE);**

**setlinestyle(0,0,2);**

**x=2;**

**y=4;**

**scaling();**

**x=6;**

**y=7;**

**scaling();**

**x=-1;**

**y=-3;**

**scaling();**

**}**

**if (choice==3)**

**{**

**cout<<"\n\n\t\t Rotation ";**

**cout<<"\n\n\t\t Enter center of rotation x\_r : ";**

**cin>>x\_r;**

**cout<<"\n\n\t\t Enter center of rotation y\_r : ";**

**cin>>y\_r;**

**cout<<"\n\n\t\t Enter angle of rotation in degree : ";**

**cin>>thetha;**

**thetha=thetha\*pi/180;**

**create\_graphics();**

**setcolor(RED);//mark new center as red**

**setlinestyle(0,0,8);**

**line(682+x\_r\*50,383-y\_r\*50,682+x\_r\*50,383-y\_r\*50);**

**setcolor(BLUE);**

**setlinestyle(0,0,2);**

**x=2;**

**y=4;**

**rotation();**

**x=6;**

**y=7;**

**rotation();**

**x=-1;**

**y=-3;**

**rotation();**

**}**

**if (choice==4)**

**{**

**cout<<"\n\n\t\t Shearing ";**

**cout<<"\n\n\t\t Enter shearing center x\_ref : ";**

**cin>>x\_ref;**

**cout<<"\n\n\t\t Enter shearing center y\_ref : ";**

**cin>>y\_ref;**

**cout<<"\n\n\t\t Enter sh\_x : ";**

**cin>>sh\_x;**

**cout<<"\n\n\t\t Enter sh\_y : ";**

**cin>>sh\_y;**

**create\_graphics();**

**setcolor(RED);//mark new center as red**

**setlinestyle(0,0,8);**

**line(682+x\_ref\*50,383-y\_ref\*50,682+x\_ref\*50,383-y\_ref\*50);**

**setlinestyle(0,0,2);**

**setcolor(BLUE);**

**x=2;**

**y=4;**

**shearing();**

**x=6;**

**y=7;**

**shearing();**

**x=-1;**

**y=-3;**

**shearing();**

**}**

**if (choice==5)**

**{**

**cout<<"\n\n\t\t Reflection ";**

**cout<<"\n\n\t\t Enter m: ";**

**cin>>m;**

**cout<<"\n\n\t\t Enter c : ";**

**cin>>c;**

**create\_graphics();**

**setlinestyle(0,0,8);**

**setcolor(RED);**

**line(682,383-50\*c,682,383-50\*c);//y=mx+c, y intercept**

**line(682+50\*-c/m,383,682+50\*-c/m,383);//y=mx+c, x intercept**

**setlinestyle(0,0,2);**

**setcolor(BLUE);//(2,4),(6,7),(-1,-3)**

**x=2;**

**y=4;**

**reflection();**

**x=6;**

**y=7;**

**reflection();**

**x=-1;**

**y=-3;**

**reflection();**

**}**

**getch();**

**closegraph();**

**}**

**return 0;**

**}**

**/\*WAP to implement 2D Transformation (Translation,Scaling,Rotation,Shearing & Reflection) in Cpp\*/**

**#include<GL/gl.h>**

**#include<GL/glu.h>**

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

**#include<iostream>**

**#include<vector>**

**#include<math.h>**

**#define PI 3.14159265358979323846**

**using namespace std;**

**void display(); //display function**

**void reshape(int,int); //reshape the viewport**

**void timer(int); //for displaying no of frames in a sec**

**void getinfo(); //info from user**

**float AxB3[3][3]= {1,0,0,**

**0,1,0,**

**0,0,1**

**};**

**float AxB1[3];**

**// matrix multiply**

**void matrix3x3(float A[3][3],float B[3][3])**

**{**

**for(int i=0; i<3; i++)**

**{**

**for(int j=0; j<3; j++)**

**{**

**AxB3[i][j]=A[i][0]\*B[0][j]+A[i][1]\*B[1][j]+A[i][2]\*B[2][j];**

**}**

**}**

**}**

**void matrix3x1(float A[3][3],float B[3])**

**{**

**for(int i=0; i<3; i++)**

**{**

**AxB1[i]=A[i][0]\*B[0]+A[i][1]\*B[1]+A[i][2]\*B[2];**

**}**

**}**

**void drawTranslate();**

**void drawScale();**

**void drawRotate();**

**void drawReflect();**

**void drawShear();**

**int selector;**

**float xr=0,yr=0; //reference**

**float tx,ty; //translate**

**float sx,sy; //scaling**

**float ang; //rotation**

**int tor; //reflection**

**float m=0,c=0;**

**float shx,shy; //shearing**

**float x[4],y[4];**

**float nx[4],ny[4];**

**void init()**

**{**

**glClearColor(0.1,0.1,0.1,1.0); //background color**

**}**

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

**{**

**getinfo();**

**if(selector>5||selector<0)**

**return 0;**

**glutInit(&argc,argv);**

**glutInitDisplayMode(GLUT\_RGB|GLUT\_DOUBLE);**

**glutInitWindowSize(500,500);**

**glutInitWindowPosition(200,200);**

**glutCreateWindow("2D Transformation");**

**glutReshapeFunc(reshape);**

**glutDisplayFunc(display);**

**glutSetKeyRepeat(GLUT\_KEY\_REPEAT\_OFF);**

**glutTimerFunc(0,timer,0);**

**init();**

**glutMainLoop();**

**return 0;**

**}**

**void display()**

**{**

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

**glLoadIdentity();**

**glColor3f(.7,.7,.7);//axis line color**

**glBegin(GL\_LINES);**

**glVertex2f(250,0);**

**glVertex2f(-250,0);**

**glVertex2f(0,250);**

**glVertex2f(0,-250);**

**glEnd();**

**glColor3f(0,.8,.8);**

**glBegin(GL\_QUADS);**

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

**{**

**glVertex2f(x[i],y[i]);**

**}**

**glEnd();**

**switch(selector)**

**{**

**case 1:**

**drawTranslate();**

**break;**

**case 2:**

**drawScale();**

**break;**

**case 3:**

**drawRotate();**

**break;**

**case 4:**

**drawReflect();**

**break;**

**case 5:**

**drawShear();**

**break;**

**}**

**glutSwapBuffers();**

**}**

**void reshape(int w,int h)**

**{**

**glViewport(0,0,w,h);**

**glMatrixMode(GL\_PROJECTION);**

**glLoadIdentity();**

**gluOrtho2D(-250,250,-250,250);**

**glMatrixMode(GL\_MODELVIEW);**

**}**

**void timer(int)**

**{**

**glutPostRedisplay();**

**glutTimerFunc(1000/30,timer,0);**

**}**

**void getinfo()**

**{**

**int temp;**

**cout<<"Enter the 4 points: "<<endl;**

**cout<<"\t 1st point(x1,y1): ";**

**cin>>x[0]>>y[0];**

**cout<<"\t 2nd point(x2,y2): ";**

**cin>>x[1]>>y[1];**

**cout<<"\t 3rd point(x3,y3): ";**

**cin>>x[2]>>y[2];**

**cout<<"\t 4th point(x4,y4): ";**

**cin>>x[3]>>y[3];**

**cout<<"\nRemember Cyan Quad is original \nAND dark Yellow Quad is translated"<<endl;**

**cout<<"\n======================================="<<endl;**

**cout<<"\t1. Translate"<<endl;**

**cout<<"\t2. Scale"<<endl;**

**cout<<"\t3. rotate"<<endl;**

**cout<<"\t4. reflect"<<endl;**

**cout<<"\t5. shear"<<endl;**

**cout<<endl<<"\tWhat do you want to do: ";**

**cin>>selector;**

**cout<<endl;**

**switch(selector)**

**{**

**case 1:**

**cout<<"Enter the following to tanslate:"<<endl;**

**cout<<"\tTx: ";**

**cin>>tx;**

**cout<<"\tTy: ";**

**cin>>ty;**

**break;**

**case 2:**

**cout<<"Enter the following to scale:"<<endl;**

**cout<<"\t1:origin scaling\n\t2:fixed point scaling \n \t Enter your choice: ";**

**cin>>temp;**

**if(temp==2)**

**{**

**cout<<"reference point:"<<endl;**

**cout<<"\txRef: ";**

**cin>>xr;**

**cout<<"\tyRef: ";**

**cin>>yr;**

**}**

**cout<<"\tSx: ";**

**cin>>sx;**

**cout<<"\tSy: ";**

**cin>>sy;**

**break;**

**case 3:**

**cout<<"Enter the following to rotate:"<<endl;**

**cout<<"\t1:origin rotating\n\t2:fixed point rotate\n \t Enter your choice: ";**

**cin>>temp;**

**if(temp==2)**

**{**

**cout<<"reference point:"<<endl;**

**cout<<"\txRef: ";**

**cin>>xr;**

**cout<<"\tyRef: ";**

**cin>>yr;**

**}**

**cout<<"\tAngle: ";**

**cin>>ang;**

**ang=ang\*PI/180;**

**break;**

**case 4:**

**cout<<"Enter the type of reflection:"<<endl;**

**cout<<"\t1.x-axis reflection."<<endl;**

**cout<<"\t2.y-axis reflection."<<endl;**

**cout<<"\t3.origin reflection."<<endl;**

**cout<<"\t4.x=y reflection."<<endl;**

**cout<<"\t5.x=-y reflection."<<endl;**

**cout<<"\t6.y=mx+c reflection."<<endl;**

**cout<<"\t Enter your choice: ";**

**cin>>tor;**

**if(tor==6)**

**{**

**cout<<"\t\t m: ";**

**cin>>m;**

**cout<<"\t\t c: ";**

**cin>>c;**

**}**

**break;**

**case 5:**

**cout<<"Enter type of shear:"<<endl;**

**cout<<"\t1.x-shear."<<endl;**

**cout<<"\t2.y-shear."<<endl;**

**cout<<"\t3.x-y-shear."<<endl;**

**cout<<"\t4.reference x-y-shear."<<endl;**

**cout<<"\t Enter your choice: ";**

**cin>>temp;**

**cout<<"\t Enter the following: "<<endl;**

**if(temp==4)**

**{**

**cout<<"reference point:"<<endl;**

**cout<<"\txRef: ";**

**cin>>xr;**

**cout<<"\tyRef: ";**

**cin>>yr;**

**}**

**if(temp!=2)**

**{**

**cout<<"\t\t Shx: ";**

**cin>>shx;**

**}**

**if(temp!=1)**

**{**

**cout<<"\t\t Shy: ";**

**cin>>shy;**

**}**

**break;**

**default:**

**cout<<"\n\t Did you see "<<selector<<" in the list?\n\t Please go to hospital to check up your eyes."<<endl;**

**break;**

**}**

**}**

**//translate**

**void drawTranslate()**

**{**

**float a[3][3]= {1,0,tx,**

**0,1,ty,**

**0,0,1**

**};**

**glColor3f(.5,.5,0);**

**glBegin(GL\_QUADS);**

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

**{**

**float b[3]= {x[i],y[i],1};**

**matrix3x1(a,b);**

**glVertex2f(AxB1[0],AxB1[1]);**

**}**

**glEnd();**

**glColor3f(1,0,0);**

**glBegin(GL\_LINES);**

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

**{**

**float b[3]= {x[i],y[i],1};**

**matrix3x1(a,b);**

**glVertex2f(x[i],y[i]);**

**glVertex2f(AxB1[0],AxB1[1]);**

**}**

**glEnd();**

**}**

**//Scaling**

**void drawScale()**

**{**

**float a[3][3]= {1,0,xr,**

**0,1,yr,**

**0,0,1**

**};**

**float b[3][3]= {sx,0,0,**

**0,sy,0,**

**0,0,1**

**};**

**float c[3][3]= {1,0,-xr,**

**0,1,-yr,**

**0,0,1**

**};**

**matrix3x3(a,b);**

**matrix3x3(AxB3,c);**

**glColor3f(.5,.5,0);**

**glBegin(GL\_QUADS);**

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

**{**

**float d[3]= {x[i],y[i],1};**

**matrix3x1(AxB3,d);**

**glVertex2f(AxB1[0],AxB1[1]);**

**}**

**glEnd();**

**glColor3f(1,0,0);**

**glBegin(GL\_LINES);**

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

**{**

**float d[3]= {x[i],y[i],1};**

**matrix3x1(AxB3,d);**

**glVertex2f(xr,yr);**

**//cout<<AxB1[0]<<" "<<AxB1[1]<<endl;**

**glVertex2f(AxB1[0],AxB1[1]);**

**}**

**glEnd();**

**}**

**//rotation**

**void drawRotate()**

**{**

**float a[3][3]= {1,0,xr,**

**0,1,yr,**

**0,0,1**

**};**

**float b[3][3]= {cos(ang),-sin(ang),0,**

**sin(ang),cos(ang),0,**

**0,0,1**

**};**

**float c[3][3]= {1,0,-xr,**

**0,1,-yr,**

**0,0,1**

**};**

**matrix3x3(a,b);**

**matrix3x3(AxB3,c);**

**glColor3f(.5,.5,0);**

**glBegin(GL\_QUADS);**

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

**{**

**float d[3]= {x[i],y[i],1};**

**matrix3x1(AxB3,d);**

**glVertex2f(AxB1[0],AxB1[1]);**

**}**

**glEnd();**

**glColor3f(1,0,0);**

**glBegin(GL\_LINES);**

**glVertex2f(xr,yr);**

**glVertex2f((x[0]+x[2])/2,(y[0]+y[2])/2);**

**float d[3]= {(x[0]+x[2])/2,(y[0]+y[2])/2,1};**

**matrix3x1(AxB3,d);**

**glVertex2f(xr,yr);**

**glVertex2f(AxB1[0],AxB1[1]);**

**glEnd();**

**}**

**//reflection**

**void drawReflect()**

**{**

**float a[3][3]= {1,0,0,**

**0,1,0,**

**0,0,1**

**};**

**switch(tor)**

**{**

**case 1:**

**glColor3f(0,1,0);**

**glBegin(GL\_LINES);**

**glVertex2f(250,0);**

**glVertex2f(-250,0);**

**glEnd();**

**break;**

**case 2:**

**glColor3f(0,1,0);**

**glBegin(GL\_LINES);**

**glVertex2f(0,250);**

**glVertex2f(0,-250);**

**glEnd();**

**a[0][0]=-1;**

**break;**

**case 3:**

**a[0][0]=-1;**

**a[1][1]=-1;**

**break;**

**case 4:**

**glColor3f(0,1,0);**

**glBegin(GL\_LINES);**

**glVertex2f(250,250);**

**glVertex2f(-250,-250);**

**glEnd();**

**m=1;**

**c=0;**

**break;**

**case 5:**

**glColor3f(0,1,0);**

**glBegin(GL\_LINES);**

**glVertex2f(250,-250);**

**glVertex2f(-250,250);**

**glEnd();**

**m=-1;**

**c=0;**

**break;**

**case 6:**

**glColor3f(0,1,0);**

**glBegin(GL\_LINES);**

**glVertex2f(250,m\*250+c);**

**glVertex2f(-250,-m\*250+c);**

**glEnd();**

**break;**

**}**

**if(tor!=2&&tor!=3)**

**{**

**a[0][0]=(1-m\*m)/(1+m\*m);**

**a[0][1]=(2\*m)/(1+m\*m);**

**a[0][2]=(-2\*c\*m)/(1+m\*m);**

**a[1][0]=(2\*m)/(1+m\*m);**

**a[1][1]=(m\*m-1)/(1+m\*m);**

**a[1][2]=(2\*c)/(1+m\*m);**

**}**

**glColor3f(.5,.5,0);**

**glBegin(GL\_QUADS);**

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

**{**

**float d[3]= {x[i],y[i],1};**

**matrix3x1(a,d);**

**glVertex2f(AxB1[0],AxB1[1]);**

**}**

**glEnd();**

**glColor3f(1,0,0);**

**glBegin(GL\_LINES);**

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

**{**

**float b[3]= {x[i],y[i],1};**

**matrix3x1(a,b);**

**glVertex2f(x[i],y[i]);**

**glVertex2f(AxB1[0],AxB1[1]);**

**}**

**glEnd();**

**}**

**//shearing**

**void drawShear()**

**{**

**float a[3][3]= {1,shx,-shx\*yr,**

**shy,1,-shy\*xr,**

**0,0,1**

**};**

**glColor3f(.5,.5,0);**

**glBegin(GL\_QUADS);**

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

**{**

**float d[3]= {x[i],y[i],1};**

**matrix3x1(a,d);**

**glVertex2f(AxB1[0],AxB1[1]);**

**}**

**glEnd();**

**glColor3f(1,0,0);**

**glBegin(GL\_LINES);**

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

**{**

**float b[3]= {x[i],y[i],1};**

**matrix3x1(a,b);**

**glVertex2f(x[i],y[i]);**

**glVertex2f(AxB1[0],AxB1[1]);**

**}**

**glEnd();**

**}**