**CGG ASSIGNMENT 6**

Name: Kedar Kale

Roll No: 129

Block B1

Topic: 3D Transformation

Code:

#include<iostream>

#include<math.h>

#include<GL/glut.h>

using namespace std;

typedef float Matrix4 [4][4];

Matrix4 theMatrix;

static GLfloat input[8][3]=

{

{40,40,-50},{90,40,-50},{90,90,-50},{40,90,-50},

{30,30,0},{80,30,0},{80,80,0},{30,80,0}

};

float output[8][3];

float tx,ty,tz;

float sx,sy,sz;

float angle;

int choice,choiceRot;

void setIdentityM(Matrix4 m)

{

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

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

m[i][j]=(i==j);

}

void translate(int tx,int ty,int tz)

{

for(int i=0;i<8;i++)

{

output[i][0]=input[i][0]+tx;

output[i][1]=input[i][1]+ty;

output[i][2]=input[i][2]+tz;

}

}

void scale(int sx,int sy,int sz)

{

theMatrix[0][0]=sx;

theMatrix[1][1]=sy;

theMatrix[2][2]=sz;

}

void RotateX(float angle) //Parallel to x

{

angle = angle\*3.142/180;

theMatrix[1][1] = cos(angle);

theMatrix[1][2] = -sin(angle);

theMatrix[2][1] = sin(angle);

theMatrix[2][2] = cos(angle);

}

void RotateY(float angle) //parallel to y

{

angle = angle\*3.14/180;

theMatrix[0][0] = cos(angle);

theMatrix[0][2] = -sin(angle);

theMatrix[2][0] = sin(angle);

theMatrix[2][2] = cos(angle);

}

void RotateZ(float angle) //parallel to z

{

angle = angle\*3.14/180;

theMatrix[0][0] = cos(angle);

theMatrix[0][1] = sin(angle);

theMatrix[1][0] = -sin(angle);

theMatrix[1][1] = cos(angle);

}

void multiplyM()

{

//We Don't require 4th row and column in scaling and rotation

//[8][3]=[8][3]\*[3][3] //4th not used

for(int i=0;i<8;i++)

{

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

{

output[i][j]=0;

for(int k=0;k<3;k++)

{

output[i][j]=output[i][j]+input[i][k]\*theMatrix[k][j];

}

}

}

}

void Axes(void)

{

glColor3f (0.0, 0.0, 0.0); // Set the color to BLACK

glBegin(GL\_LINES); // Plotting X-Axis

glVertex2s(-1000 ,0);

glVertex2s( 1000 ,0);

glEnd();

glBegin(GL\_LINES); // Plotting Y-Axis

glVertex2s(0 ,-1000);

glVertex2s(0 , 1000);

glEnd();

}

void draw(float a[8][3])

{

glBegin(GL\_QUADS);

glColor3f(0.7,0.4,0.5); //behind

glVertex3fv(a[0]);

glVertex3fv(a[1]);

glVertex3fv(a[2]);

glVertex3fv(a[3]);

glColor3f(0.8,0.2,0.4); //bottom

glVertex3fv(a[0]);

glVertex3fv(a[1]);

glVertex3fv(a[5]);

glVertex3fv(a[4]);

glColor3f(0.3,0.6,0.7); //left

glVertex3fv(a[0]);

glVertex3fv(a[4]);

glVertex3fv(a[7]);

glVertex3fv(a[3]);

glColor3f(0.2,0.8,0.2); //right

glVertex3fv(a[1]);

glVertex3fv(a[2]);

glVertex3fv(a[6]);

glVertex3fv(a[5]);

glColor3f(0.7,0.7,0.2); //up

glVertex3fv(a[2]);

glVertex3fv(a[3]);

glVertex3fv(a[7]);

glVertex3fv(a[6]);

glColor3f(1.0,0.1,0.1);

glVertex3fv(a[4]);

glVertex3fv(a[5]);

glVertex3fv(a[6]);

glVertex3fv(a[7]);

glEnd();

}

void init()

{

glClearColor(1.0,1.0,1.0,1.0); //set backgrond color to white

glOrtho(-454.0,454.0,-250.0,250.0,-250.0,250.0);

// Set the no. of Co-ordinates along X & Y axes and their gappings

glEnable(GL\_DEPTH\_TEST);

// To Render the surfaces Properly according to their depths

}

void display()

{

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

Axes();

glColor3f(1.0,0.0,0.0);

draw(input);

setIdentityM(theMatrix);

switch(choice)

{

case 1:

translate(tx,ty,tz);

break;

case 2:

scale(sx,sy,sz);

multiplyM();

break;

case 3:

switch (choiceRot) {

case 1:

RotateX(angle);

break;

case 2: RotateY(angle);

break;

case 3:

RotateZ(angle);

break;

default:

break;

}

multiplyM();

break;

}

draw(output);

glFlush();

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB|GLUT\_DEPTH);

glutInitWindowSize(1362,750);

glutInitWindowPosition(0,0);

glutCreateWindow("3D TRANSFORMATIONS");

init();

cout<<"Enter your choice number:\n1.Translation\n2.Scaling\n3.Rotation\n=>";

cin>>choice;

switch (choice) {

case 1:

cout<<"\nEnter Tx,Ty &Tz: \n";

cin>>tx>>ty>>tz;

break;

case 2:

cout<<"\nEnter Sx,Sy & Sz: \n";

cin>>sx>>sy>>sz;

break;

case 3:

cout<<"Enter your choice for Rotation about axis:\n1.parallel to X-axis."

<<"(y& z)\n2.parallel to Y-axis.(x& z)\n3.parallel to Z-axis."

<<"(x& y)\n =>";

cin>>choiceRot;

switch (choiceRot) {

case 1:

cout<<"\nENter Rotation angle: ";

cin>>angle;

break;

case 2:

cout<<"\nENter Rotation angle: ";

cin>>angle;

break;

case 3:

cout<<"\nENter Rotation angle: ";

cin>>angle;

break;

default:

break;

}

break;

default:

break;

}

glutDisplayFunc(display);

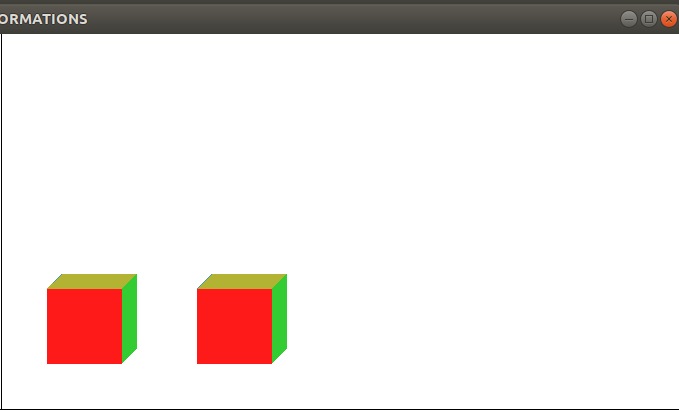
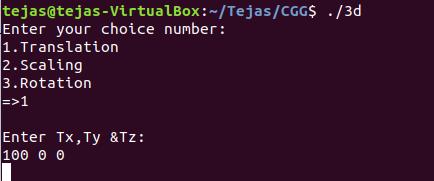
glutMainLoop();

return 0;

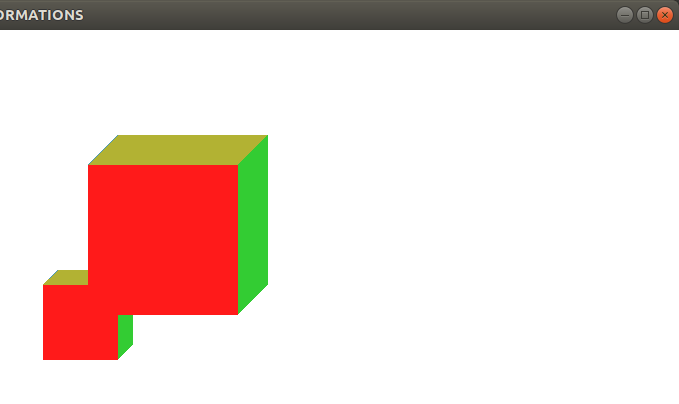
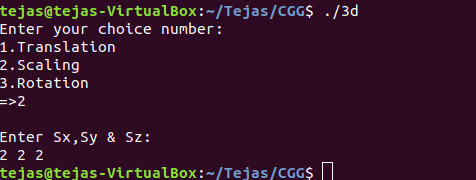
}

Output:

1) Transformation:



2) Scaling:



3) Rotation:

