Department of Computer Science and Engineering

Shivanirudh S G, 185001146, Semester VII

23 September 2021

UCS1712 - Graphics and Multimedia Lab

Exercise 8: 3-Dimensional Transformations in C++ using OpenGL

Aim:

Perform basic 3D Transformations on any 3D Object.

Code:

```
#ifndef LOPENGL_H
#define LOPENGL_H

#include <GL/freeglut.h>
#include <GL/gl.h>
#include <GL/glu.h>
#include <math.h>
#include <stdio.h>
#include<stdlib.h>
#include<iostream>
#include<vector>
#include<ctime>
#include<tuple>
#include<unistd.h>
using namespace std;

#endif
```

```c
#ifndef LUTIL_H
#define LUTIL_H

#include "Headers.h"

//Screen Constants
const int SCREEN_WIDTH = 1362;
const int SCREEN_HEIGHT = 750;
const int SCREEN_FPS = 60;
const int POINT_SIZE=3;

typedef float MatrixDim [4][4];
MatrixDim transformation_matrix;

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 init();

void render();

void setIdentityM(MatrixDim m);

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

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

void RotateX(float angle);

void RotateY(float angle);

void RotateZ(float angle);

void multiplyMatrices();

void Axes(void);

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

#endif
```

```c
#include "Signatures.h"

void init(){
    glClearColor(0.0,0.0,0.0,1.0);
    glOrtho(-454.0,454.0,-250.0,250.0,-250.0,250.0);
    glEnable(GL_DEPTH_TEST);
}
```

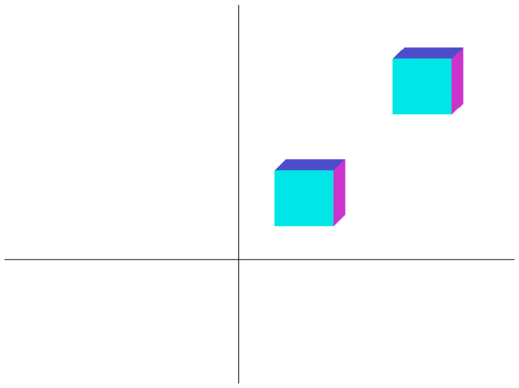
```cpp

void render(){


    while(true){
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        Axes();
        glColor3f(1.0,0.0,0.0);
        draw(input);
        setIdentityM(transformation_matrix);

        cout<<"Choose transformation: "<<endl;
        cout<<"1 for Translation"<<endl<<"2 for Rotation"<<endl;
        cout<<"3 for Scaling"<<endl<<"0 to Exit"<<endl;
        cout<<"Enter your choice: ";cin>>choice;

        cout<<"transformation: "<<choice<<endl;

        if(choice == 1){
            cout<<"Enter the translation factor for X, Y and Z: ";
            cin>>tx >> ty >> tz;
        }
        else if(choice == 2){
            cout<<"Enter the rotation angle: ";
            cin>>angle;

            cout<<"Choose axis to rotate around: "<<endl;
            cout<<"1 for around X axis"<<endl<<"2 for around Y axis
    "<<endl;
            cout<<"3 for around Z axis"<<endl;
            cout<<"Enter your choice: ";cin>>choiceRot;

        }
        else if(choice == 3){
            cout<<"Enter the scaling factor for X, Y and Z: ";
            cin>>sx >> sy >> sz;
        }
        else if(choice){
            cout<<"Invalid option"<<endl;
        }
        else;

        switch(choice){
            case 1:
                translate(tx,ty,tz);
                break;
            case 2:
                switch (choiceRot) {
                    case 1:
                        RotateX(angle);
                        break;
                    case 2:
                        RotateY(angle);
                        break;
                    case 3:
                        RotateZ(angle);
                        break;
                    default:
                        break;
                }
                multiplyMatrices();
                for (int i = 0; i < 8;i++){
                    for (int j = 0; j < 3;j++){
                        cout << output[i][j] << " ";
                    }
                    cout << endl;
                }
                break;
            case 3:
                scale(sx,sy,sz);
                multiplyMatrices();
                break;
        }

        draw(output);
        glFlush();
    }
}

void setIdentityM(MatrixDim 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){
    transformation_matrix[0][0]=sx;
    transformation_matrix[1][1]=sy;
    transformation_matrix[2][2]=sz;
}

void RotateX(float angle){
    cout << angle << endl;
    angle = angle * 3.1416 / 180;
    cout << angle << endl;
    transformation_matrix[1][1] = cos(angle);
    transformation_matrix[1][2] = -sin(angle);
    transformation_matrix[2][1] = sin(angle);
    transformation_matrix[2][2] = cos(angle);
}

void RotateY(float angle){
    angle = angle*3.1416/180;
    transformation_matrix[0][0] = cos(angle);
    transformation_matrix[0][2] = -sin(angle);
    transformation_matrix[2][0] = sin(angle);
    transformation_matrix[2][2] = cos(angle);

}

void RotateZ(float angle){
    angle = angle*3.1416/180;
    transformation_matrix[0][0] = cos(angle);
    transformation_matrix[0][1] = sin(angle);
    transformation_matrix[1][0] = -sin(angle);
    transformation_matrix[1][1] = cos(angle);
}

void multiplyMatrices(){
    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]+=(input[i][k]*transformation_matrix[k
    ][j]);
            }
        }
    }

}
void Axes(void){
    glColor3f (1.0, 1.0, 1.0);
    glBegin(GL_LINES);
        glVertex2s(-1000 ,0);
        glVertex2s( 1000 ,0);
    glEnd();
    glBegin(GL_LINES);
        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();
}
```

```
#include "Helpers.h"

int main( int argc, char* args[] ){

    glutInit(&argc,args);
    glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
    glutInitWindowSize(1362,750);
    glutInitWindowPosition(0,0);
    glutCreateWindow( "OpenGL" );

    init();

    glutDisplayFunc(render);

    glutMainLoop();

    return 0;
}
```

Output:

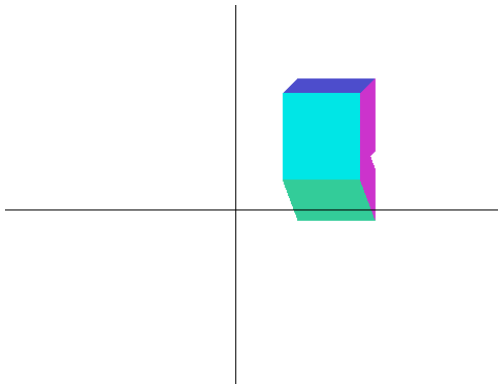
Translation:

```
    Choose transformation:
1 for Translation
2 for Rotation
3 for Scaling
0 to Exit
Enter your choice: 1
Enter the translation factor for X, Y and Z: 100 100 100
```



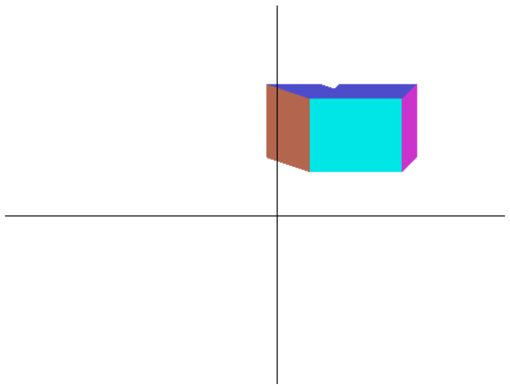
Rotation - X axis:

```
Choose transformation:
1 for Translation
2 for Rotation
3 for Scaling
0 to Exit
Enter your choice: 2

Enter the rotation angle: 45
Choose axis to rotate around:
1 for around X axis
2 for around Y axis
3 for around Z axis
Enter your choice: 1
```



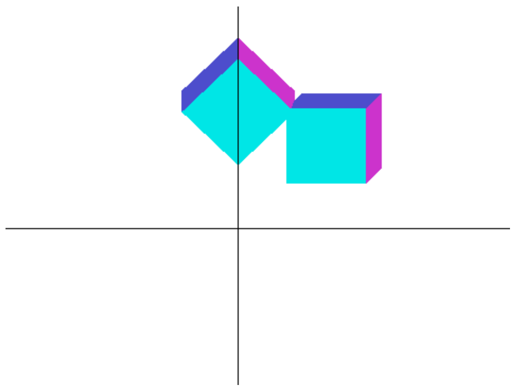
Rotation - Y axis:

```
Choose transformation:
1 for Translation
2 for Rotation
3 for Scaling
0 to Exit
Enter your choice: 2

Enter the rotation angle: 45
Choose axis to rotate around:
1 for around X axis
2 for around Y axis
3 for around Z axis
Enter your choice: 2
```



Rotation - Z axis:

```
Choose transformation:
1 for Translation
2 for Rotation
3 for Scaling
0 to Exit
Enter your choice: 2

Enter the rotation angle: 45
Choose axis to rotate around:
1 for around X axis
2 for around Y axis
3 for around Z axis
Enter your choice: 3
```



Scaling:

```
Choose transformation:
1 for Translation
2 for Rotation
3 for Scaling
0 to Exit
Enter your choice: 3
Enter the translation factor for X, Y and Z: 2 2 2
```