**DEERWALK INSTITUTE OF TECHNOLOGY**

**Tribhuvan University**

**Faculties of Computer Science**

**A logo of a sea creature

Description automatically generated**

**Bachelors of Science in Computer Science and Information Technology (BSc. CSIT)**

**Course: Computer Graphics (CSC214)**

**Year/Semester: II/III**

**A Lab report on:**

**Implementation of 3D transformation Algorithm in C++**

Submitted by:

Name: Parth Poudyal

Roll: 1317

Submitted to:

Binod Sitaula

Department of Computer Science

**Theory**

Like 2D transformation, 3D transformation too is the conversion of the vertices into something else but in a 3D space.

We can perform transformations such as translation, rotation, scaling etc. in a 3D space.

Translation

Translation is the shifting of position of objects in a screen.

Translation can be performed as follows:

X’ = X + tx

Y’ = Y + ty

Z’ = Z + tz

The pair (tx, ty, tz) is called the translation vector or shift vector. The above equations can also be represented using the column vectors.

Rotation

Rotation is the process of rotating an object about a pivot point (typically the origin) by a specified angle . For the new coordinates for X’, Y’, Z’ on rotation about theta along respective axis can be represented as:

Rotation about X - axis:

Rotation about Y - axis:

Rotation about Z - axis:

Scaling

Scaling is a process of increment or decrement of dimensions of an object by a certain ratio. Scaling can be achieved by multiplying the coordinates with a scaling factor to get the desired result. Scaling changes the size of an object by multiplying the original coordinates with scaling factors

For Scaling:

**Program**

#include <iostream>

#include <graphics.h>

#include <cmath>

#include <conio.h>

#define x 250

#define y 100

#define z 100

using namespace std;

int ori[24];

int s, ch;

void wtor(int pts3[24]);

void cube(int pts1[16]);

void tran();

void rot();

void scale();

void cube(int pts1[16])

{

    setcolor(15);

    setlinestyle(DOTTED\_LINE, 1, 1);

    line(pts1[0], pts1[1], pts1[2], pts1[3]);

    line(pts1[2], pts1[3], pts1[4], pts1[5]);

    line(pts1[4], pts1[5], pts1[6], pts1[7]);

    line(pts1[6], pts1[7], pts1[0], pts1[1]);

    line(pts1[0], pts1[1], pts1[8], pts1[9]);

    line(pts1[2], pts1[3], pts1[10], pts1[11]);

    line(pts1[4], pts1[5], pts1[12], pts1[13]);

    line(pts1[6], pts1[7], pts1[14], pts1[15]);

    line(pts1[8], pts1[9], pts1[10], pts1[11]);

    line(pts1[10], pts1[11], pts1[12], pts1[13]);

    line(pts1[12], pts1[13], pts1[14], pts1[15]);

    line(pts1[14], pts1[15], pts1[8], pts1[9]);

}

void wtor(int pts3[24])

{

    int pts2[16];

    float a1 = M\_PI \* 60 / 180;

    float a2 = M\_PI \* 60 / 180;

    for (int i = 0, j = 0; i < 24; i += 3, j += 2)

    {

        pts2[j] = pts3[i] + pts3[i + 2] \* (1 / tan(a1)) \* cos(a2);

        pts2[j + 1] = pts3[i + 1] + pts3[i + 2] \* (1 / tan(a1)) \* sin(a2);

    }

    cube(pts2);

}

void tran()

{

    int tr3[24], tx, ty, tz;

    cout << "\nEnter the translation vectors (tx, ty, tz): ";

    cin >> tx >> ty >> tz;

    for (int i = 0; i < 24; i += 3)

    {

        tr3[i] = ori[i] + tx;

        tr3[i + 1] = ori[i + 1] + ty;

        tr3[i + 2] = ori[i + 2] + tz;

    }

    wtor(tr3);

}

void rot()

{

    int r3[24], axis;

    float deg, rad;

    cout << "\nEnter the angle of rotation: ";

    cin >> deg;

    rad = M\_PI \* deg / 180;

    cout << "\nChoose axis of rotation (1: X, 2: Y, 3: Z): ";

    cin >> axis;

    for (int i = 0; i < 24; i += 3)

    {

        switch (axis)

        {

        case 1:

            r3[i] = ori[i];

            r3[i + 1] = ori[i + 1] \* cos(rad) - ori[i + 2] \* sin(rad);

            r3[i + 2] = ori[i + 1] \* sin(rad) + ori[i + 2] \* cos(rad);

            break;

        case 2:

            r3[i] = ori[i] \* cos(rad) + ori[i + 2] \* sin(rad);

            r3[i + 1] = ori[i + 1];

            r3[i + 2] = -ori[i] \* sin(rad) + ori[i + 2] \* cos(rad);

            break;

        case 3:

            r3[i] = ori[i] \* cos(rad) - ori[i + 1] \* sin(rad);

            r3[i + 1] = ori[i] \* sin(rad) + ori[i + 1] \* cos(rad);

            r3[i + 2] = ori[i + 2];

            break;

        default:

            cout << "\nInvalid choice!";

            return;

        }

    }

    wtor(r3);

}

void scale()

{

    float sx, sy, sz;

    int s3[24];

    cout << "\nEnter the scaling factors (sx, sy, sz): ";

    cin >> sx >> sy >> sz;

    for (int i = 0; i < 24; i += 3)

    {

        s3[i] = ori[i] \* sx;

        s3[i + 1] = ori[i + 1] \* sy;

        s3[i + 2] = ori[i + 2] \* sz;

    }

    wtor(s3);

}

int main()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, (char\*)"");

    cout << "\nEnter the side of the cube: ";

    cin >> s;

    // Original cube coordinates

    int p = 0;

    for (int i = 0; i <= 1; i++)

    {

        for (int j = 0; j <= 1; j++)

        {

            for (int k = 0; k <= 1; k++)

            {

                ori[p++] = x + i \* s;

                ori[p++] = y - j \* s;

                ori[p++] = z + k \* s;

            }

        }

    }

    wtor(ori);

    getch();

    do

    {

        cleardevice();

        wtor(ori);

        cout << "\n1. Translation\n2. Rotation\n3. Scaling\n4. Exit\nEnter your choice: ";

        cin >> ch;

        switch (ch)

        {

        case 1: tran(); break;

        case 2: rot(); break;

        case 3: scale(); break;

        case 4: break;

        default: cout << "\nInvalid choice!";

        }

        getch();

    } while (ch != 4);

    closegraph();

    return 0;

}

**Conclusion**

From this project I got to understand how 3D transformations can be performed in C++. I would also like to thank our Computer Graphics teacher for incredible support during this process.