|  |  |
| --- | --- |
| **Logo  Description automatically generated** | **BÀI THỰC HÀNH ĐỒ HỌA MÁY TÍNH** |
|  | **MA TRẬN CÁC PHÉP BIẾN HÌNH** |

**Họ và tên Sinh viên:**Phan Trần Nhật Hạ  

**Mã Sinh viên:**102210159     **Nhóm:** 21Nh12

[**1.**    **lab03AffineMatrix.cpp**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-MaTranCacPhepBienHinh.htm#_Toc129268324)

[**2.**    ***labmatrixCG* - Thực hiện các phép biến hình thông qua tính toán ma trận**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-MaTranCacPhepBienHinh.htm#_Toc129268325)

[2.1        Vectors.h](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-MaTranCacPhepBienHinh.htm#_Toc129268326)

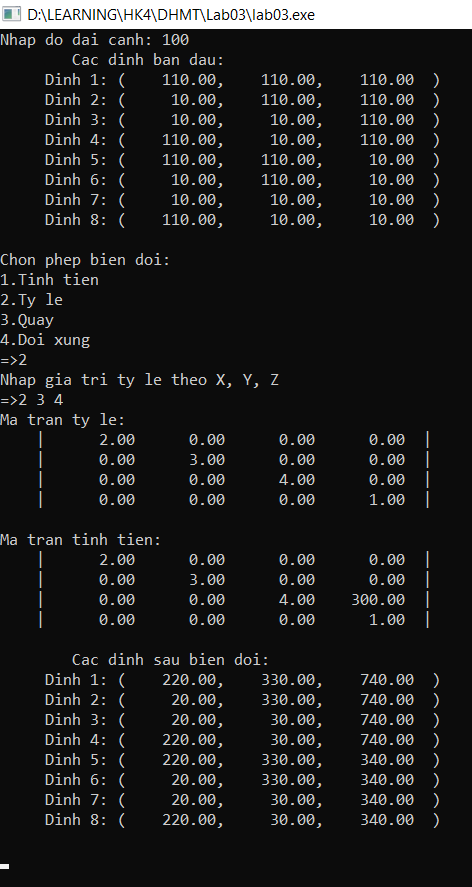
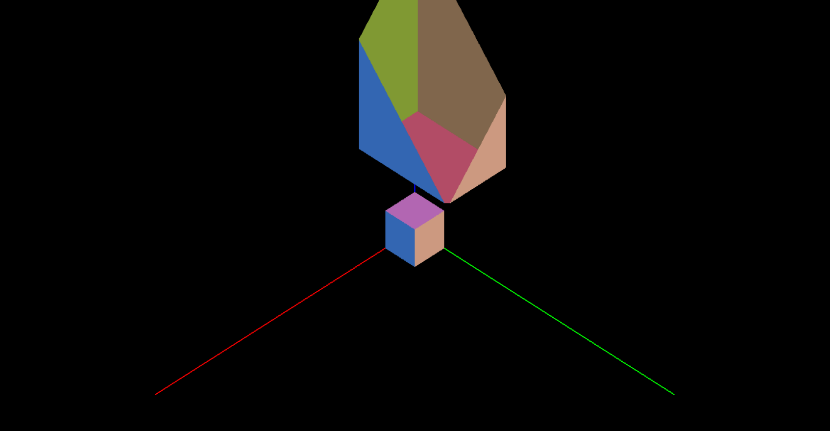
[2.2        Matrices.h](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-MaTranCacPhepBienHinh.htm#_Toc129268327)

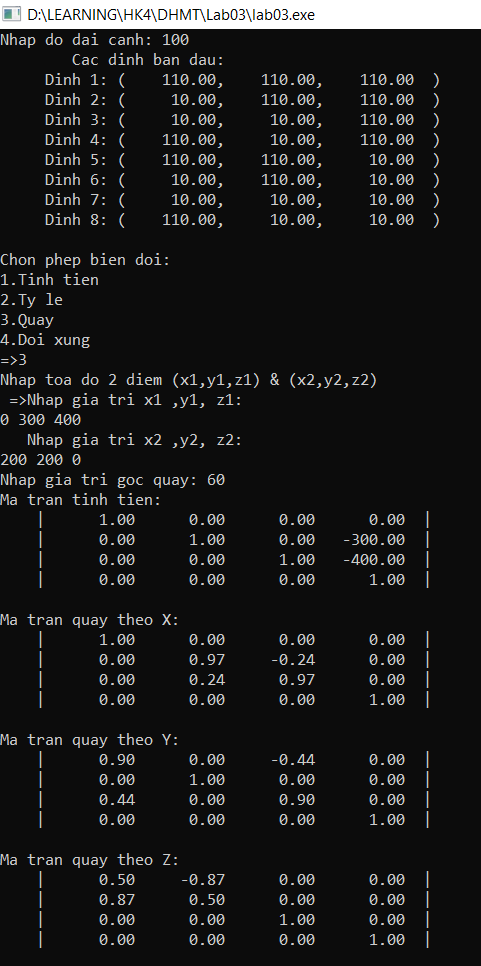
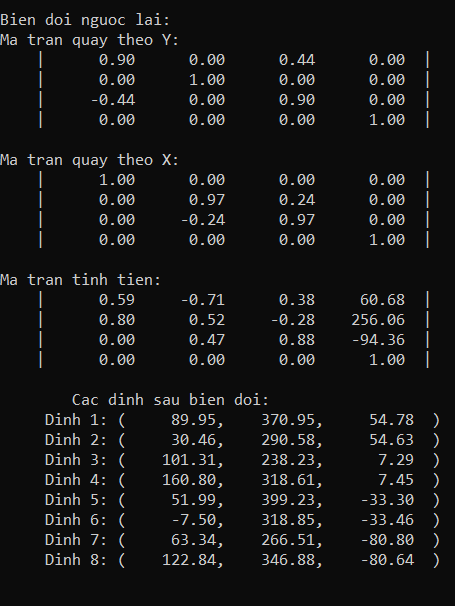
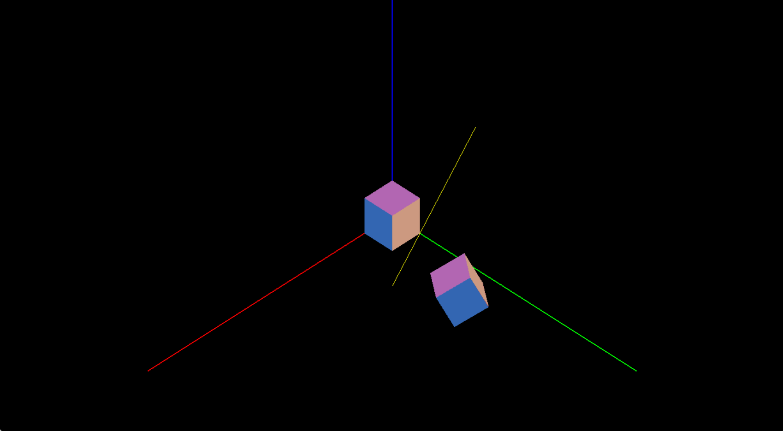
[2.3        main.cpp](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-MaTranCacPhepBienHinh.htm#_Toc129268328)

[**3.**    **BÀI TẬP**](http://itfdut.ddns.net/mod/resource/tknetlab/LabDHMT-MaTranCacPhepBienHinh.htm#_Toc129268329)

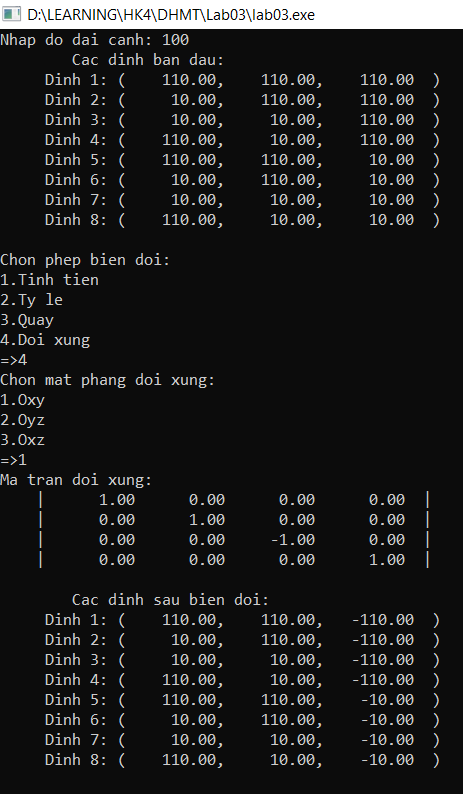
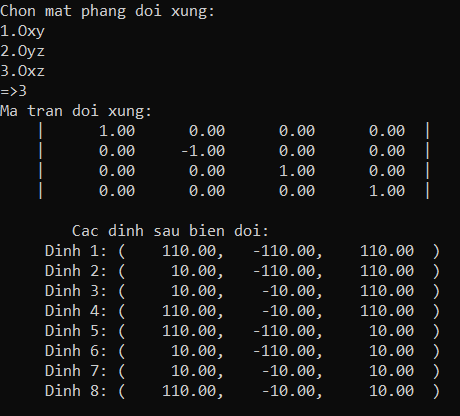
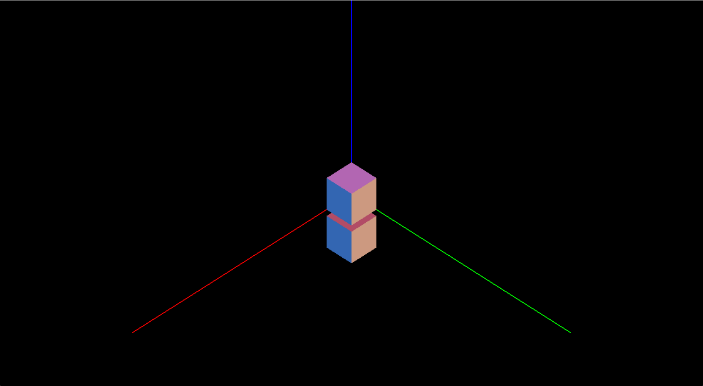
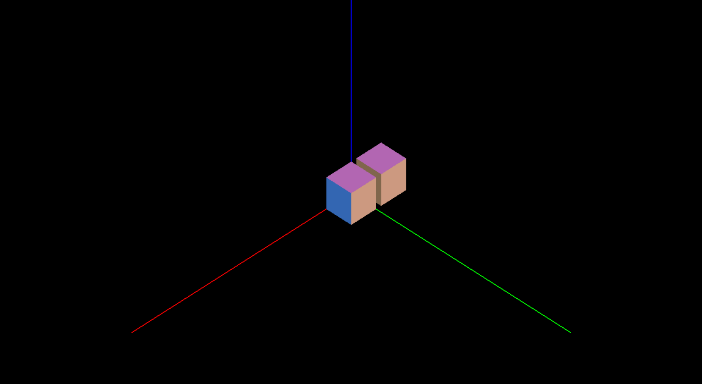
**1.   lab03AffineMatrix.cpp**

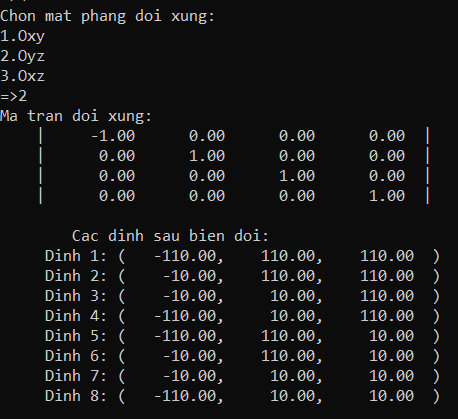
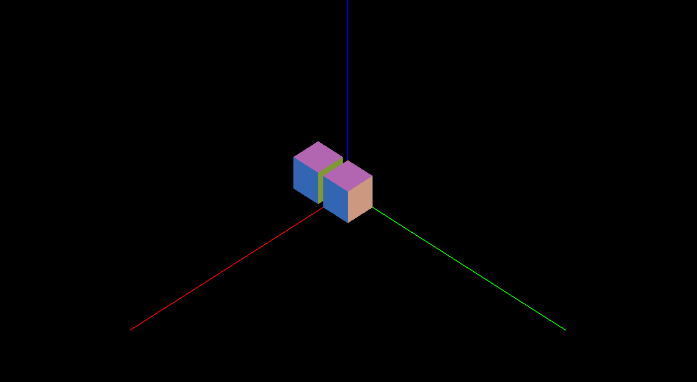
|  |  |
| --- | --- |
|  |  |

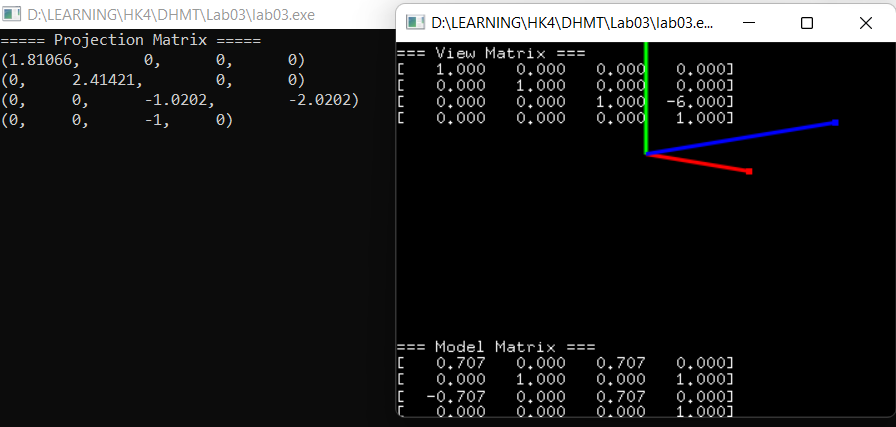
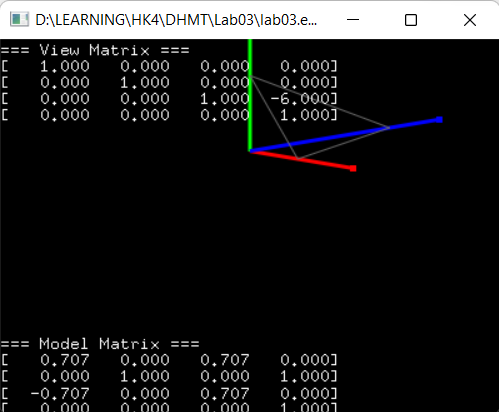
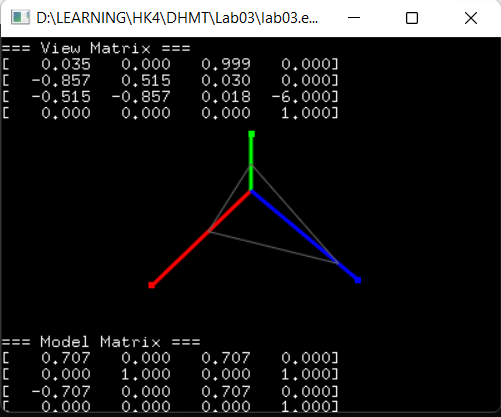


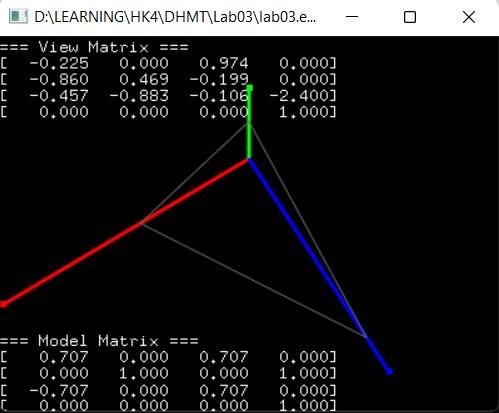


|  |  |
| --- | --- |
|  |  |





**2.   *labmatrixCG* - Thực hiện các phép biến hình thông qua tính toán ma trận**



**3.   BÀI TẬP**

Viết chương trình thực hiện các yêu cầu sau:

1) Cho điểm Pobj(xobj, yobj, zobj, 1) trong hệ tọa độ thế giới thực (world wpace), với xobj, yobj, zobj là các giá trị được trích từ mã sinh viên.

Xác định các ma trận biến đổi và điểm Pw(xw, yw, zw, 1) khi thực hiện các phép biến đổi sau:

§  Phép biến đổi Tịnh tiến với dx, dy, dz

§  Quay quanh trục Ox, Oy, Oz một góc tùy chọn

§  Quay quanh một trục PQ có P(xp, yp, zp), Q(xq, yq, zq)

2) Xét điểm Pw(xw, yw, zw, 1) trong hệ tọa độ thế giới thực (World Space) được tính toán từ câu 1.

Xác định các ma trận biến đổi và và xác định điểm Pcam(xcam, ycam, zcam, 1) trong hệ tọa độ camera space khi thực hiện:

§  Phép biến đổi camera đặt tại eye = (xeye, yeye, zeye), nhìn vào tâm center = (x2, y2, z2), hướng lên k = (0, 1, 0)]

3) Cho điểm Pcam(xcam, ycam, zcam, 1) trong hệ tọa độ camera space là kết quả của câu 2.

Xác định các ma trận biến đổi và và xác định điểm Pclip(xclip, yclip, zclip, wclip) trong clipping/projection space khi thực hiện:

§  Phép chiếu song song

§  Phép chiếu xiên

§  Phép chiếu trực giao

§  Phép chiếu phối cảnh

**CODE**

#include <iostream>

#include <iomanip>

#include <GL/glut.h>

#include <math.h>

using namespace std;

void printmatrix(GLdouble thematrix[16])

{

int i, j;

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

{

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

{

cout << fixed << setprecision(4) << setw(7) << thematrix[j \* 4 + i] << " ";

}

cout << endl;

}

}

void printCurrentModelViewMatrix()

{

GLdouble matrixd[16];

glGetDoublev(GL\_MODELVIEW\_MATRIX, matrixd);

printmatrix(matrixd);

}

float\* mulMatrix(GLdouble matrix[16], float x, float y, float z)

{

float\* res = new float[3];

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

{

res[i] = matrix[0 \* 4 + i] \* x + matrix[1 \* 4 + i] \* y + matrix[2 \* 4 + i] \* z + matrix[3 \* 4 + i];

}

return res;

}

float\* mul(float x, float y, float z, int mode = GL\_MODELVIEW\_MATRIX)

{

GLdouble matrixd[16];

glGetDoublev(GL\_VIEWPORT, matrixd);

printmatrix(matrixd);

return mulMatrix(matrixd, x, y, z);

}

void display2()

{

printf("Print the current modelview matrix\n");

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

gluLookAt(0, 1.0, 3.0, 0, 0, 0, 0, 1.0, 0);

printCurrentModelViewMatrix();

glLoadIdentity();

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 1.0, 1.0);

glLoadIdentity();

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

glScalef(1.0, 2.0, 1.0);

glutWireCube(1.0);

glFlush();

}

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

{

glutInit(&argc, argv);

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

glutInitWindowSize(20, 20);

glutCreateWindow("Viewport Example");

glutDisplayFunc(display2);

glMatrixMode(GL\_MODELVIEW);

float x0, y0, z0;

cout << "Nhap diem ban dau (x,y,z): " << endl;

cin >> x0 >> y0 >> z0;

// translate

glLoadIdentity();

float dx, dy, dz;

cout << "Nhap vector tinh tien (x,y,z): " << endl;

cin >> dx >> dy >> dz;

glTranslatef(dx, dy, dz);

// rotate

float angleX, angleY, angleZ;

cout << "Nhap goc quay quanh Ox: " << endl;

cin >> angleX;

glRotatef(angleX, 1, 0, 0);

cout << "Nhap goc quay quanh Oy: " << endl;

cin >> angleY;

glRotatef(angleY, 0, 1, 0);

cout << "Nhap goc quay quanh Oz: " << endl;

cin >> angleZ;

glRotatef(angleZ, 0, 0, 1);

// PQ

float xp, yp, zp, xq, yq, zq;

cout << "Nhap toa do 2 diem P, Q:" << endl;

cout << "Toa do P (x,y,z):" << endl;

cin >> xp >> yp >> zp;

cout << "Toa do Q (x,y,z):" << endl;

cin >> xq >> yq >> zq;

float angle;

cout << "Nhap goc quay quanh PQ:" << endl;

cin >> angle;

float x = xq - xp;

float y = yq - yp;

float z = zq - zp;

float thetaX = acos(z / sqrt(y \* y + z \* z));

float thetaY = atan(x / z);

glTranslatef(-xp, -yp, -zp);

glRotatef(thetaX, 1, 0, 0);

glRotatef(-thetaY, 0, 1, 0);

glRotatef(angle, 0, 0, 1);

glRotatef(thetaY, 0, 1, 0);

glRotatef(-thetaX, 1, 0, 0);

glTranslatef(xp, yp, zp);

float\* re = mul(x0, y0, z0);

cout << "Point World\n";

cout << re[0] << " " << re[1] << " " << re[2] << endl;

float e1, e2, e3;

float c1, c2, c3;

float u1, u2, u3;

cout << "Nhap diem nhin (x,y,z):" << endl;

cin >> e1 >> e2 >> e3;

cout << "Nhap center (x,y,z):" << endl;

cin >> c1 >> c2 >> c3;

cout << "nhap up (recommend: 0,1,0):" << endl;

cin >> u1 >> u2 >> u3; //(0, 1, 0)

glMatrixMode(GL\_MODELVIEW);

gluLookAt(e1, e2, e3, c1, c2, c3, u1, u2, u3);

re = mul(x0, y0, z0);

cout << "\n Point Camera \n" << endl;

x0 = re[0], y0 = re[1], z0 = re[2];

cout << x0 << " " << y0 << " " << z0 << endl;

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45, 1, 0.1, 100);

re = mul(x0, y0, z0, GL\_PROJECTION\_MATRIX);

x0 = re[0], y0 = re[1], z0 = re[2];

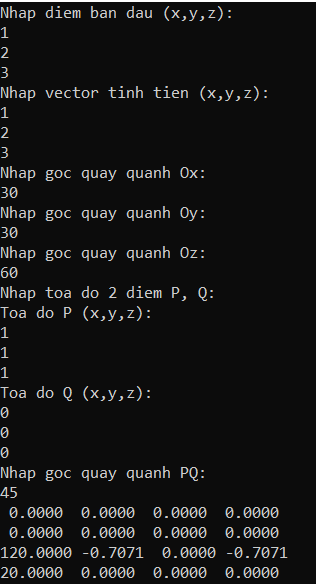
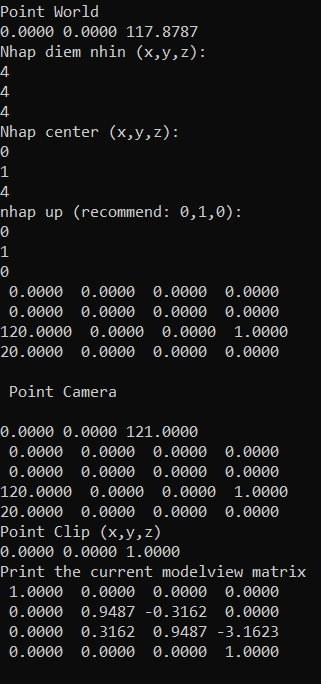
cout << "Point Clip (x,y,z)" << endl;

cout << x0 << " " << y0 << " " << z0 << endl;

glutMainLoop();

return 0;

}



------------------------------------------------