**Trần Hà Duy – 22IT054 – Lab 06**

#include <GL/glut.h>

#include <cmath>

#include <vector>

#include <algorithm>

// Back-face Culling

bool isBackFace(float v1[3], float v2[3], float v3[3], float cameraPos[3]) {

float u[3], v[3], n[3], view[3];

// Tính vector canh

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

u[i] = v2[i] - v1[i];

v[i] = v3[i] - v1[i];

view[i] = cameraPos[i] - v1[i];

}

// Tính pháp tuyen

n[0] = u[1] \* v[2] - u[2] \* v[1];

n[1] = u[2] \* v[0] - u[0] \* v[2];

n[2] = u[0] \* v[1] - u[1] \* v[0];

// Dot product

float dot = n[0] \* view[0] + n[1] \* view[1] + n[2] \* view[2];

return dot < 0;

}

// Z-buffer (Depth-buffer)

void init() {

glEnable(GL\_DEPTH\_TEST);

glClearColor(0.0, 0.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, 1.0, 1.0, 100.0);

glMatrixMode(GL\_MODELVIEW);

}

// Depth-sorting (Painter's Algorithm)

struct Face {

float vertices[4][3];

float zDepth;

};

// Tính do sâu trung b?nh (z)

float calculateZDepth(Face& face) {

float zSum = 0.0f;

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

zSum += face.vertices[i][2];

return zSum / 4.0f;

}

void drawCube() {

float cameraPos[3] = {0.0f, 0.0f, 5.0f};

float vertices[8][3] = {

{-1.0, -1.0, -1.0}, {1.0, -1.0, -1.0}, {1.0, 1.0, -1.0}, {-1.0, 1.0, -1.0},

{-1.0, -1.0, 1.0}, {1.0, -1.0, 1.0}, {1.0, 1.0, 1.0}, {-1.0, 1.0, 1.0}

};

int faces[6][4] = {

{0, 1, 2, 3}, {4, 5, 6, 7}, {0, 4, 7, 3},

{1, 5, 6, 2}, {0, 1, 5, 4}, {3, 2, 6, 7}

};

std::vector<Face> faceList;

for (int i = 0; i < 6; ++i) {

Face face;

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

face.vertices[j][0] = vertices[faces[i][j]][0];

face.vertices[j][1] = vertices[faces[i][j]][1];

face.vertices[j][2] = vertices[faces[i][j]][2];

}

face.zDepth = calculateZDepth(face);

faceList.push\_back(face);

}

// Sap xep và ve mat theo Depth-sorting

std::sort(faceList.begin(), faceList.end(), [](Face& f1, Face& f2) {

return f1.zDepth > f2.zDepth;

});

glBegin(GL\_QUADS);

for (auto& face : faceList) {

if (!isBackFace(face.vertices[0], face.vertices[1], face.vertices[2], cameraPos)) {

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

glVertex3fv(face.vertices[j]);

}

}

}

glEnd();

}

// Display function

void display() {

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

glLoadIdentity();

gluLookAt(3.0, 3.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

// Ve hinh lap phuong

glColor3f(1.0f, 0.5f, 0.0f);

drawCube();

glFlush();

glutSwapBuffers();

}

// Main function

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

glutInit(&argc, argv);

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

glutInitWindowSize(800, 600);

glutCreateWindow("OpenGL - Back-face Culling, Z-buffer and Depth-sorting");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}