Politechnika Wrocławska

Sprawozdanie 2

Ćwiczenie 4.Oświetlenie scen

Krzysztof Zalewa

2.12.2024

Spis treści

1 Wstęp teoretyczny

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortisfacilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdietmi nec ante. Donec ullamcorper, felis non sodales...

1.1 Temat1

1.1.1 Temat 1.1

2 Zadanie laboratoryjne

2.1 Treść zadania

2.2 Opis działania programu

Zgodnie z treścią zadania program rysuje 4 obiekty. Domyślnie jajko i czajnik rysowane są w kolorze czarnym. Jednakże jest możliwość zmiany koloru na losowy. Wyświetlone obiekty można obracać za pomocą klawiatury (Przycisk musi być wciśnięty i przytrzymany).
Kontrola obrotu:
A D -obrót po osi Y
W S - obrót po osi X
Q E - obrót po osi Z
ESC - Powrót do menu (okno konsolowe)
Ruch myszy w osi X - Obrót kamery w osi X
Ruch myszy w osi Y - Obrót kamery w osi Y
Scroll up - Przybiliżenie obiektu
Scroll down - Oddalenie obiektu

2.3 Kod programu

.

```
#include <windows.h>
#include <iostream>
#include <GL/glu.h>
#include <vector>
#include <math.h>
#define FREEGLUT_STATIC
#include <GL/freeglut.h>
using namespace std;
```

```cpp
HWND consoleWindow;
HWND glutWindow;

GLfloat deg = 0;
int sx =0,sy = 0,sz = 0;
bool spin = false;
bool drawTeapot = true;
bool color = false;
int eggMode = 0;
float totalRotationX = 0.0f,totalRotationY = 0.0f,totalRotationZ = 0.0f;
int radius = 6,lastX = 0,lastY = 0;
float cameraRotationX = 0.0f,cameraRotationY = 0.0f,cameraRotationZ = radius;
float phi = 0.0f;
float theta = 0.0f;
struct pointsRgb{
    //Pozycja
    float x = 0.0;
    float y = 0.0;
    float z = 0.0;
    //Kolor
    float r = 0.0;
    float g = 0.0;
    float b = 0.0;
}typedef pointsRgb;

class Egg{
    private:
    int density;
    vector<vector<pointsRgb>> pointsMatrix;
    float randFloat(){
        return (float)rand()/(float)(RAND_MAX);
    }
    public:
    Egg(int density ) : density(density){
        pointsMatrix.resize(density,vector<pointsRgb>(density));
    }
    vector<vector<pointsRgb>> getPointsMatrix(){
        return pointsMatrix;
    }
    void generateMatrix(float scale){
        for(int u=0;u<(density);u++){
            float _u = 0.5/((float)density-1);
            _u *= u;
            if(u==density-1){
                pointsMatrix[u][0].y = scale*((160*pow(_u,4)) - (320*pow(_u,3)) +
                ↪  (160 * pow(_u,2)) - 5);
                if(color){
                    pointsMatrix[u][0].r = randFloat();
                    pointsMatrix[u][0].g = randFloat();
                    pointsMatrix[u][0].b = randFloat();
                }else{
                    pointsMatrix[u][0].r = 0.0f;
                    pointsMatrix[u][0].g = 0.0f;
                    pointsMatrix[u][0].b = 0.0f;
                }
                break;
            }
            for(int v=0;v<density;v++){
                float _v = v/((float)density);
                _v *= 2.0f;
                pointsMatrix[u][v].x = scale*((-90*pow(_u,5) + 225*pow(_u,4) - 270*pow(_u,3) + 180*pow(_u,2) - 45*_u) * cos(M_PI*_v));
                pointsMatrix[u][v].y = scale*(160*pow(_u,4) - 320*pow(_u,3) + 160 * pow(_u,2) - 5);
                pointsMatrix[u][v].z = scale*((-90*pow(_u,5) + 225*pow(_u,4) - 270*pow(_u,3) + 180*pow(_u,2) - 45*_u) * sin(M_PI*_v));
                if(color){
                    pointsMatrix[u][v].r = randFloat();
                    pointsMatrix[u][v].g = randFloat();
                    pointsMatrix[u][v].b = randFloat();
                }else{
                    pointsMatrix[u][v].r = 0.0f;
                    pointsMatrix[u][v].g = 0.0f;
                    pointsMatrix[u][v].b = 0.0f;
                }
            }
        }
    }
    void draw(int model){
        switch (model)
        {
        case 1:
            glPointSize(5.0f);
            glBegin(GL_POINTS);
            for(int u=0;u<density-1;u++){
                if(u==0){
                    glColor3f(pointsMatrix[u][0].r,pointsMatrix[u][0].g,pointsMatrix[u][0].b);
                    glVertex3f(pointsMatrix[u][0].x,pointsMatrix[u][0].y,pointsMatrix[u][0].z);
                    continue;
                }
                if(u==density-2){
                    glColor3f(pointsMatrix[u+1][0].r,pointsMatrix[u+1][0].g,pointsMatrix[u+1][0].b);
                    glVertex3f(pointsMatrix[u+1][0].x,pointsMatrix[u+1][0].y,pointsMatrix[u+1][0].z);
                    break;
                }
                for(int v=0;v<density;v++){
                    glColor3f(pointsMatrix[u][v].r,pointsMatrix[u][v].g,pointsMatrix[u][v].b);
                    glVertex3f(pointsMatrix[u][v].x,pointsMatrix[u][v].y,pointsMatrix[u][v].z);
                }
            }
            glEnd();
            break;
        case 2:
            glBegin(GL_LINES);
            for(int u=0;u<density-1;u++){
                if(u==0){
                    for(int v=0;v<density;v++){
                        glColor3f(pointsMatrix[u][0].r,pointsMatrix[u][0].g,pointsMatrix[u][0].b);
                        glVertex3f(pointsMatrix[u][0].x,pointsMatrix[u][0].y,pointsMatrix[u][0].z);
                        glColor3f(pointsMatrix[u+1][v].r, pointsMatrix[u+1][v].g, pointsMatrix[u+1][v].b);
                        glVertex3f(pointsMatrix[u+1][v].x, pointsMatrix[u+1][v].y, pointsMatrix[u+1][v].z);
                    }
                    continue;
                }
                if(u==density-2){
                    for(int v=0;v<density;v++){
                        glColor3f(pointsMatrix[u+1][0].r,pointsMatrix[u+1][0].g,pointsMatrix[u+1][0].b);
                        glVertex3f(pointsMatrix[u+1][0].x,pointsMatrix[u+1][0].y,pointsMatrix[u+1][0].z);
                        glColor3f(pointsMatrix[u][v].r, pointsMatrix[u][v].g, pointsMatrix[u][v].b);
                        glVertex3f(pointsMatrix[u][v].x, pointsMatrix[u][v].y, pointsMatrix[u][v].z);
                    }
                    break;
                }
                for(int v=0;v<density;v++){
                    int nextV = (v + 1) % density;
                    glColor3f(pointsMatrix[u][v].r,pointsMatrix[u][v].g,pointsMatrix[u][v].b);
                    glVertex3f(pointsMatrix[u][v].x,pointsMatrix[u][v].y,pointsMatrix[u][v].z);
                    glColor3f(pointsMatrix[u+1][v].r, pointsMatrix[u+1][v].g, pointsMatrix[u+1][v].b);
                    glVertex3f(pointsMatrix[u+1][v].x, pointsMatrix[u+1][v].y, pointsMatrix[u+1][v].z);

                    glColor3f(pointsMatrix[u][v].r,pointsMatrix[u][v].g,pointsMatrix[u][v].b);
                    glVertex3f(pointsMatrix[u][v].x,pointsMatrix[u][v].y,pointsMatrix[u][v].z);
                    glColor3f(pointsMatrix[u][nextV].r, pointsMatrix[u][nextV].g, pointsMatrix[u][nextV].b);
                    glVertex3f(pointsMatrix[u][nextV].x, pointsMatrix[u][nextV].y, pointsMatrix[u][nextV].z);

                }
            }
            glEnd();
            break;
        case 3:
            glBegin(GL_TRIANGLES);
            for(int u=0;u<density-1;u++){
                if(u==0){
                    for(int v=0;v<density;v++){
                        int nextV = (v + 1) % density;
                        glColor3f(pointsMatrix[u][0].r,pointsMatrix[u][0].g,pointsMatrix[u][0].b);
                        glVertex3f(pointsMatrix[u][0].x,pointsMatrix[u][0].y,pointsMatrix[u][0].z);
                        glColor3f(pointsMatrix[u+1][nextV].r,pointsMatrix[u+1][nextV].g,pointsMatrix[u+1][nextV].b);
                        glVertex3f(pointsMatrix[u+1][nextV].x,pointsMatrix[u+1][nextV].y,pointsMatrix[u+1][nextV].z);
                        glColor3f(pointsMatrix[u+1][v].r,pointsMatrix[u+1][v].g,pointsMatrix[u+1][v].b);
                        glVertex3f(pointsMatrix[u+1][v].x,pointsMatrix[u+1][v].y,pointsMatrix[u+1][v].z);
                    }
                    continue;
                }
                if(u==density-2){
                    for(int v=0;v<density;v++){
                        int nextV = (v + 1) % density;
                        glColor3f(pointsMatrix[u+1][0].r,pointsMatrix[u+1][0].g,pointsMatrix[u+1][0].b);
                        glVertex3f(pointsMatrix[u+1][0].x,pointsMatrix[u+1][0].y,pointsMatrix[u+1][0].z);
                        glColor3f(pointsMatrix[u][v].r,pointsMatrix[u][v].g,pointsMatrix[u][v].b);
                        glVertex3f(pointsMatrix[u][v].x,pointsMatrix[u][v].y,pointsMatrix[u][v].z);
                        glColor3f(pointsMatrix[u][nextV].r,pointsMatrix[u][nextV].g,pointsMatrix[u][nextV].b);
                        glVertex3f(pointsMatrix[u][nextV].x,pointsMatrix[u][nextV].y,pointsMatrix[u][nextV].z);
                    }
                    break;
                }
                for(int v=0;v<density;v++){
                    int nextV = (v + 1) % density;
                    glColor3f(pointsMatrix[u][v].r,pointsMatrix[u][v].g,pointsMatrix[u][v].b);
                    glVertex3f(pointsMatrix[u][v].x,pointsMatrix[u][v].y,pointsMatrix[u][v].z);
                    glColor3f(pointsMatrix[u+1][nextV].r,pointsMatrix[u+1][nextV].g,pointsMatrix[u+1][nextV].b);
                    glVertex3f(pointsMatrix[u+1][nextV].x, pointsMatrix[u+1][nextV].y, pointsMatrix[u+1][nextV].z);
                    glColor3f(pointsMatrix[u+1][v].r,pointsMatrix[u+1][v].g,pointsMatrix[u+1][v].b);
                    glVertex3f(pointsMatrix[u+1][v].x, pointsMatrix[u+1][v].y, pointsMatrix[u+1][v].z);

                    glColor3f(pointsMatrix[u+1][nextV].r,pointsMatrix[u+1][nextV].g,pointsMatrix[u+1][nextV].b);
                    glVertex3f(pointsMatrix[u+1][nextV].x, pointsMatrix[u+1][nextV].y, pointsMatrix[u+1][nextV].z);
                    glColor3f(pointsMatrix[u][v].r,pointsMatrix[u][v].g,pointsMatrix[u][v].b);
```

```
                    glVertex3f(pointsMatrix[u][v].x,pointsMatrix[u][v].y,pointsMatrix[u][v].z);
                    glColor3f(pointsMatrix[u][nextV].r,pointsMatrix[u][nextV].g,pointsMatrix[u][nextV].b);
                    glVertex3f(pointsMatrix[u][nextV].x, pointsMatrix[u][nextV].y, pointsMatrix[u][nextV].z);
                }
            }
            glEnd();
            break;
        }
    }
    ~Egg(){

    }
};
Egg egg(20);
void toggleFocusToConsole() {
    ShowWindow(glutWindow, SW_HIDE);
    ShowWindow(consoleWindow, SW_SHOWNORMAL);
    SetForegroundWindow(consoleWindow);
}

void toggleFocusToGLUT() {
    ShowWindow(consoleWindow, SW_HIDE);
    ShowWindow(glutWindow, SW_SHOWNORMAL);
    SetForegroundWindow(glutWindow);
}
void animate(){
    float rotationSpeed = 0.5f;
    totalRotationX += rotationSpeed * sx;
    totalRotationY += rotationSpeed * sy;
    totalRotationZ += rotationSpeed * sz;
    glutPostRedisplay();
}
void reset_rotation(){
    totalRotationX = 0.0f;
    totalRotationY = 0.0f;
    totalRotationZ = 0.0f;
    radius = 6;
    cameraRotationX = 0.0f;
    cameraRotationY = 0.0f;
    cameraRotationZ = radius;
    lastX = 0;
    lastY = 0;
}
string bool_to_string(bool convert){
    if(convert){
        return "true";
    }else{
        return "false";
    }
}
void printControls(){
    cout<<"A D - obrot po osi Y\n";
    cout<<"W S - obrot po osi X\n";
    cout<<"Q E - obrot po osi Z\n";
    cout<<"ESC - Powrot do menu (okno konsolowe)\n";
    cout<<"Nalezy nacisnac i przytrzymac PPM\n";
    cout<<"Ruch myszy w osi X - Obrot kamery w osi X\n";
    cout<<"Ruch myszy w osi Y - Obrot kamery w osi Y\n";
    cout<<"Scroll up - Przybilizenie obiektu\n";
    cout<<"Scroll down - Oddalenie obiektu\n";
    cout<<"Nacisnij Enter zeby kontynuowac\n"<<flush;
    cin.get();
    cin.get();
}
void menu(){
    toggleFocusToConsole();
    reset_rotation();
    cout<<"==============================\n";
    cout<<"1. Narysuj czajnik\n";
    cout<<"2. Narysuj jajko (punkty)\n";
    cout<<"3. Narysuj jajko (linie)\n";
    cout<<"4. Narysuj jajko (trojkaty) \n";
    cout<<"5. Rysowanie w kolorze: "<<bool_to_string(color)<<"\n";
    cout<<"6. Kontrola\n";
    cout<<"7. Zakoncz program\n";
    cout<<"> ";
    int x;
    cin>> x;
    switch (x)
    {
    case 1:
        drawTeapot = true;
        break;
    case 2:
        drawTeapot = false;
        eggMode = 1;
        break;
    case 3:
        drawTeapot = false;
        eggMode = 2;
        break;
    case 4:
        drawTeapot = false;
        eggMode = 3;
        break;
    case 5:
        color=!color;
        egg.generateMatrix(0.5f);
        menu();
        break;
    case 6:
        printControls();
        menu();
        break;
    case 7:
        exit(0);
        break;
    default:
        cout<<"Podano nieporawny znak\n";
        menu();
        break;
    }
    toggleFocusToGLUT();
    glutPostRedisplay();
}
void keyDown(u_char key,int x,int y){
    switch (key)
    {
    case 'Q':
    case 'q':
        sz=1;
        glutIdleFunc(animate);
        break;
    case 'E':
    case 'e':
        sz=-1;
        glutIdleFunc(animate);
        break;
    case 'W':
    case 'w':
        sx=-1;
        glutIdleFunc(animate);
        break;
    case 'S':
    case 's':
        sx=1;
        glutIdleFunc(animate);
        break;
    case 'A':
    case 'a':
        sy=-1;
        glutIdleFunc(animate);
        break;
    case 'D':
    case 'd':
        sy=1;
        glutIdleFunc(animate);
        break;
    default:
        break;
    }
}
void keyUp(u_char key,int x,int y){
    switch (key)
    {
    case 'E':
    case 'Q':
    case 'e':
    case 'q':
        sz=0;
        break;
    case 'W':
    case 'S':
    case 'w':
    case 's':
        sx=0;
        break;
    case 'A':
    case 'D':
    case 'd':
    case 'a':
        sy=0;
        break;
    case 27:
        menu();
        break;
    default:
        break;
    }
    if (sx == 0 && sy == 0 && sz == 0) {
        glutIdleFunc(nullptr);
    }
}
void mouse(int x, int y){
    float sensitivity =0.75f;
    float phi = sensitivity*((2.0f * y / 400) - 1.0f);
    float theta = sensitivity*((2.0f * (400 - x) / 400) - 1.0f);
    float maxPhi = 1.75f;  // Restrict phi range to avoid gimbal lock (approx ±85
    ↪  degrees)
    if (phi > maxPhi){
        phi = maxPhi;
    }
    if (phi < -maxPhi){
        phi = -maxPhi;
    }
    cameraRotationX = radius*cos(theta)*cos(phi);
    cameraRotationY = radius*sin(phi);
    cameraRotationZ = radius*sin(theta)*cos(phi);
    lastX = x;
    lastY = y;
    glutPostRedisplay();
}
void mouseWheel(int button, int dir, int x, int y){
    if (dir > 0){
        radius -= 1;
    }else{
        radius += 1;
    }
    if(radius>=10){
        radius=10;
    }
    if(radius<=1){
        radius=1;
    }
    glutPostRedisplay();
}
void display() {
    GLfloat lPos[] = {0,4,0,1};//x,y,z,czy światło jest odległe
    GLfloat col[] = {1,0,0,1};
    glLoadIdentity();
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    //glLightfv(GL_LIGHT0,GL_POSITION,lPos);
    gluLookAt(cameraRotationX,cameraRotationY,cameraRotationZ,0,0,0,0,1,0);//Ustawienie kamery
    glRotatef(totalRotationX, 1.0f, 0.0f, 0.0f);
    glRotatef(totalRotationY, 0.0f, 1.0f, 0.0f);
    glRotatef(totalRotationZ, 0.0f, 0.0f, 1.0f);
    if(drawTeapot){
        glutWireTeapot(1);
    }else{
        glShadeModel(GL_FLAT);
        egg.draw(eggMode);
    }
    glutSwapBuffers();
}
void Init() {
    egg.generateMatrix(0.5f);
    glEnable(GL_DEPTH_TEST); //bez tego frontalna sciana nadpisuje tylnią
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glFrustum(-1,1,-1,1,2,10);
    glMatrixMode(GL_MODELVIEW);
    // // Włącza culling, czyli pomijanie tylnych ścianek
    // glEnable(GL_CULL_FACE);
    // // Ustawia kierunek frontowych ścianek jako przeciwny do ruchu wskazówek zegara
    // glFrontFace(GL_CW);
    // // Ustawia pomijanie tylnych ścianek
    glCullFace(GL_BACK);
    //glEnable(GL_LIGHTING); //Włączenie oświetlenia
    //glEnable(GL_LIGHT0); //Dodanie źródła światła
}
int main(int argc, char** argv){
    consoleWindow = GetConsoleWindow();
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(800,800);
    glutCreateWindow("Lab 3 - Czajnik i Jajko");
    glutWindow = FindWindowW(NULL,L"Lab 3 - Czajnik i Jajko");
    Init();
    glutDisplayFunc(display);
    glutIdleFunc(nullptr);
    glutKeyboardFunc(keyDown);
    glutKeyboardUpFunc(keyUp);
    glutMotionFunc(mouse);
    glutMouseWheelFunc(mouseWheel);
    menu();

    glutMainLoop();
    system("pause");
    return 0;
}
```

Fragment kodu 1: Fragment kodu z programu

3 Wnioski

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortisfacilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdietmi nec ante. Donec ullamcorper, felis non sodales...

4 Źródła

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortisfacilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdietmi nec ante. Donec ullamcorper, felis non sodales...