Obraz zawierający tekst, Czcionka, logo, Grafika

Opis wygenerowany automatycznie

Sprawozdanie 2

Ćwiczenie 2. **Modelowanie obiektów 3D**

Autor: Krzysztof Zalewa

Spis treści

[1.Wstęp teoretyczny 3](#_Toc182249348)

[1.1 Utah teapot 3](#_Toc182249349)

[1.2 Jajko 3](#_Toc182249350)

[1.2.1 Równanie jajka 3](#_Toc182249351)

[1.3 Renderowanie przy pomocy prymitywów 4](#_Toc182249352)

[1.3.1 GL\_POINTS 4](#_Toc182249353)

[1.3.2 GL\_LINES 5](#_Toc182249354)

[1.3.3 GL\_TRIANGLES 6](#_Toc182249355)

[2.Zadania laboratoryjne 7](#_Toc182249356)

[2.1 Treść zadania 7](#_Toc182249357)

[2.2 Opis działania programu 7](#_Toc182249358)

[2.3 Kod programu 7](#_Toc182249359)

[3.Wnioski 14](#_Toc182249360)

[4.Źródła 14](#_Toc182249361)

# 1.Wstęp teoretyczny

## 1.1 Utah teapot

Obraz zawierający garnek, naczynia kuchenne, czajniczek, czajnik

Opis wygenerowany automatycznie

*Rysunek 1. Utah teapot wygenerowany przy pomocy mojego programu*

Czajnik z Utah to jeden z podstawowych modeli 3d często używany jako punkt odniesienia w testach. Jest to model matematyczny czajnika marki Melitta. Model ten został stworzony przez Martina Newella w 1975. Jako jeden z pierwszych obiektów 3d został opisany przy pomocy krzywych Béziera.

## 1.2 Jajko

### 1.2.1 Równanie jajka

By narysować jajko należało wyliczyć współrzędne punktów z których składa się to jajko. Wzory na punkty które należą do jajka zostały podane na prezentacji ze strony Dr. Gniewkowskiego

**x (u, v ) =(︀ −90 · u5 + 225 · u4 − 270 · u3 + 180 · u2 − 45 · u)︀ · cos (𝜋 · v )**

**y (u, v ) = 160 · u4 − 320 · u3 + 160 · u2 − 5**

**z(u, v ) =(︀ −90 · u5 + 225 · u4 − 270 · u3 + 180 · u2 − 45 · u)︀ · sin (𝜋 · v )**

By poprawnie narysować jajko u powinno być w przedziale 0<=u<=1 a v w przedziale 0<=v<=2𝜋

## 1.3 Renderowanie przy pomocy prymitywów

### 1.3.1 GL\_POINTS

Obraz zawierający rysowanie, szkic, biały, design

Opis wygenerowany automatycznie

*Rysunek 2. Jajko składające się z samych punktów (wygenerowane w moim programie)*

Do wykonania tej części zadania wystarczyło narysować wszystkie wygenerowane punkty.

### 1.3.2 GL\_LINES

Obraz zawierający szkic, bezkręgowiec, rysowanie, sztuka

Opis wygenerowany automatycznie

*Rysunek 3. Jajko składające się z linii (wygenerowane w moim programie)*

By osiągnąć taki efekt należało połączyć ze sobą dwie pary punktów:

1. Punkt (u,v) z (u+1,v) dla otrzymania linii poziomej
2. Punkt (u,v) z (u,v+1) dla otrzymania linii poionowej

Gdzie u to numer linii poziomej a v linii pionowej

### 1.3.3 GL\_TRIANGLES

Obraz zawierający sztuka, Wielobarwność, kula, kreatywność

Opis wygenerowany automatycznie

*Rysunek 4. Jajko składające się z trójkątów (wygenerowane w moim programie)*

W przeciwieństwie do pozostałych opcji renderowania dla jajka złożonego z trójkątów wystarczyło przejść przez 1/3 wierzchołków. Proces generowania polegał na narysowaniu dwóch trójkątów

1. Trójkąt (u,v),(u+1,v) i (u+1,v-1)
2. Trójkąt (u,v),(u+1,v) i (u,v+1)

# 2.Zadania laboratoryjne

## 2.1 Treść zadania

W ramach zadania należało napisać program który w przestrzeni 3d wyświetli :

1. Czajnik (Utah teapot)
2. Jajko (Za pomocą punktów)
3. Jajko (Za pomocą linii)
4. Jajko (Za pomocą trójkątów)

## 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)

## 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;

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;

float totalRotationY = 0.0f;

float totalRotationZ = 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\*2));

    }

    vector<vector<pointsRgb>> getPointsMatrix(){

        return pointsMatrix;

    }

    void generateMatrix(float scale){

        for(int u=0;u<(density/2)-6;u++){

            float \_u = u/((float)density-1);

            for(int v=0;v<density;v++){

                float \_v = v/((float)density-1);

                \_v \*= 2.0f \* M\_PI;

                pointsMatrix[u][v].x = scale\*((-90\*pow(\_u,5)) + (255\*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)) + (255\*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:

            glBegin(GL\_POINTS);

            for(int u=0;u<(density/2)-6;u++){

                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/2)-7;u++){

                for(int v=0;v<density-1;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);

                    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][v+1].r, pointsMatrix[u][v+1].g, pointsMatrix[u][v+1].b);

                    glVertex3f(pointsMatrix[u][v+1].x, pointsMatrix[u][v+1].y, pointsMatrix[u][v+1].z);

                }

            }

            glEnd();

            break;

        case 3:

            glBegin(GL\_TRIANGLES);

            for(int u=0;u<(density/2)-7;u++){

                for(int v=1;v<density/3;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);

                    glVertex3f(pointsMatrix[u+1][v].x, pointsMatrix[u+1][v].y, pointsMatrix[u+1][v].z);

                    glVertex3f(pointsMatrix[u+1][v-1].x, pointsMatrix[u+1][v-1].y, pointsMatrix[u+1][v-1].z);

                    glColor3f(pointsMatrix[u+1][v].r,pointsMatrix[u+1][v].g,pointsMatrix[u+1][v].b);

                    glVertex3f(pointsMatrix[u][v].x,pointsMatrix[u][v].y,pointsMatrix[u][v].z);

                    glVertex3f(pointsMatrix[u+1][v].x, pointsMatrix[u+1][v].y, pointsMatrix[u+1][v].z);

                    glVertex3f(pointsMatrix[u][v+1].x, pointsMatrix[u][v+1].y, pointsMatrix[u][v+1].z);

                }

            }

            glEnd();

            break;

        }

    }

    ~Egg(){}

};

Egg egg(100);

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;

}

string bool\_to\_string(bool convert){

    if(convert){

        return "true";

    }else{

        return "false";

    }

}

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. 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:

        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 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(0,0,6,0,0,0,0,1,0);//Ustawienie kamery

    //Pierwsze trzy lokalizacja

    //Gdzie patrzy

    //Tilt 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{

        glPushMatrix();

            egg.draw(eggMode);

        glPopMatrix();

    }

    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);

}

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

    consoleWindow = GetConsoleWindow();

    glutInit(&argc, argv);

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

    glutInitWindowSize(800,800);

    glutCreateWindow("Lab 2 - Czajnik i Jajko");

    glutWindow = FindWindowW(NULL,L"Lab 2 - Czajnik i Jajko");

    Init();

    glutDisplayFunc(display);

    glutIdleFunc(nullptr);

    glutKeyboardFunc(keyDown);

    glutKeyboardUpFunc(keyUp);

    menu();

    glutMainLoop();

    system("pause");

    return 0;

}

# 3.Wnioski

Implementacja przebiegła pomyślnie. Program poprawnie generuje jajka i czajnik.

# 4.Źródła

[1] <https://graphics.cs.utah.edu/teapot/>

[2] slajd 6 https://gniewkowski.wroclaw.pl/gk/lab3.pdf