POLITECHNIKA WROCŁAWSKA
Instytut Informatyki, Automatyki i Robotyki
Zakład Systemów Komputerowych

Grafika komputerowa i komunikacja człowiek - komputer

Kurs: INEK00012L

Sprawozdanie z ćwiczenia nr 3

TEMAT ĆWICZENIA: OpenGL – Modelowanie obiektów 3D

Wykonał:	**Bartosz Szymczak, nr indeksu 252734**
Termin:	**Poniedziałek TN 7:30-10:30**
Data wykonania ćwiczenia:	**22.11.2021r.**
Data oddania sprawozdania:	
Ocena:	

Uwagi prowadzącego:

Spis treści

1. Cel ćwiczenia

Celem ćwiczenia jest wprowadzenie w zagadnienia modelowania i wizualizacji scen 3D z wykorzystaniem biblioteki OpenGL z rozszerzeniem GLUT. Na podstawie przykładów ze strony ćwiczenia oraz na podstawie równań parametrycznych będziemy w stanie stworzyć model obiektu jajka. Dodatkowo dodamy prosty sposób sterowania modelem za pomocą klawiatury.

2. Wstęp teoretyczny

Biblioteki OpenGl wraz z rozszerzeniami są zazwyczaj używane do modelowania 3D. Modele 3D obiektów uzyskujemy poprzez odpowiednią budowę powierzchni opisanej równaniami parametrycznymi. Dodatkowo istnieją możliwości obracania wytworzonym obrazem wokół różnych osi. Do opisania jajka służą następujące równania parametryczne:

$$x(u,v) = (-90u^5 + 225u^4 - 270u^3 + 180u^2 - 45u)\cos(\pi v),$$
$$y(u,v) = 160u^4 - 320u^3 + 160u^2,$$
$$z(u,v) = (-90u^5 + 225u^4 - 270u^3 + 180u^2 - 45u)\sin(\pi v),$$

gdzie u i v należą do zbioru [0;1].

3. Opis programu

Do stworzenia programu wykorzystano język C++, biblioteki OpenGL i GLUT oraz środowisko Visual Studio Community 2019. Szkielet programu bazuje na przykładowych programach opisanych w instrukcji do ćwiczenia.

3.1 Biblioteki oraz zmienne globalne

```
#define _USE_MATH_DEFINES
#include <windows.h>
#include <iostream>
#include <gl/gl.h>
#include <gl/glut.h>
#include <cmath>
using namespace std;
```

```
//Definicja zmiennej przechowującej współrzędne (x,y,z)
typedef float point3[3];

//Definicja zmiennej przechowującej ilość przedziałów
//boku kwadratu jednostkowego dziedziny parametrycznej
int n;

//Definicja i inicjalizacja zmiennej odpowiedzialnej za rodzaj wyświetlania obrazu
int model = 1;

//Macierze zmiennych point3
point3** points;
point3** colors;

//Trzy kąty obrotu, theta[0] odpowiada za obrót wokół osi x, theta [1] wokół osi y, a theta [2] wokół osi z
static GLfloat theta[] = { 20.0, 20.0, 20.0 };

//Definicja i inicjalizacja zmiennej przechowującej informację o obracaniu obrazu
//początkowo ustawiona na false
bool spin = false;
```

3.2 Poszczególne funkcje

3.2.1 Funkcja main()

Funkcja main() jest punktem wejścia programu. Wywołuje menu, które komunikuje się z użytkownikiem po czym generuje macierze, przechowującą współrzędne punktów jajka oraz kolory potrzebne do reprezentacji graficznych. Dalej następuje szereg funkcji związanych z biblioteką OpenGL oraz rozszerzeniem GLUT. Następuje wybranie trybów wyświetlania, buforów, ustawień okna aplikacji oraz funkcji zwrotnych (callback), które biorą udział w renderowaniu obrazu.

```
// Główny punkt wejścia programu.
void main(void)
{
   //Wyświetlenie menu w oknie konsolowym
   menu();
   //Wygenerowanie macierzy, przechowującej współrzędne punktów jajka
   points = new point3 * [n];
   for (int i = 0; i < n; ++i)
   {
      points[i] = new point3[n];
   }
   //Wygenerowanie macierzy, przechowującej kolory
   colors = new point3 * [n];
   for (int i = 0; i < n; ++i)
   {
      colors[i] = new point3[n];
      for (int j = 0; j < n; ++j)
      {
         colors[i][j][0] = (float)rand() / RAND_MAX;
         colors[i][j][1] = (float)rand() / RAND_MAX;
         colors[i][j][2] = (float)rand() / RAND_MAX;
      }
   }
```

```
   //Funkcja ustawia tryby wyświetlania:
   // GLUT_DOUBLE - używanie podwójnego buforu przestrzeni renderowania,
   // GLUT_RGB - używanie kolorów w konwencji RGB
   // GLUT_DEPTH - użycie buforu do stworzenia głębi (istotny dla grafiki 3D)
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
   //Ustawienie początkowej wielkości okna aplikacji
   glutInitWindowSize(300, 300);
   //Tworzenie okna aplikacji o podanym tytule
   glutCreateWindow("Jajko");
   //Metoda glutDisplayFunc ustawia funkcję,
   //która będzie wywoływana w pętli w celu renderowania obrazu
   glutDisplayFunc(RenderScene);
   //Metoda glutReshapeFunc ustawia funkcję,
   //która będzie wywoływana w sytuacji zmiany rozmiaru wytworzonego okna
   glutReshapeFunc(ChangeSize);
   //Metoda glutIdleFunc ustawia funkcję,
   //która będzie wywoływana w pętli w celu modyfikacji obrazu
   if (spin) glutIdleFunc(spinEgg);
   //Metoda glutKeyboardFunc ustawia funkcję,
   //która będzie wywoływana w sytuacji wykrycia wciśnięcia klawisza
   glutKeyboardFunc(keys);
   //Wywołanie funkcji ustalającej stan renderowania
   MyInit();
   //Funkcja glEnable włącza aktualizowanie buforu głębi
   glEnable(GL_DEPTH_TEST);
   //Funkcja glutMainLoop zapętla powyższe procesy
   glutMainLoop();
}
```

3.2.2 Funkcja eggPoints()

Funkcja eggPoints() odpowiedzialna jest za generowanie punktów reprezentujących strukturę jajka. Bazuje ona na obliczaniu wartości współrzędnych punktów jajka na podstawie gotowych wzorów. Funkcja wypełnia w ten sposób tablicę points.

```
//Metoda generująca punkty reprezentujące strukturę jajka
void eggPoints()
{
   float x, y, z, fu, fv;
   for (int u = 0; u < n; ++u)
   {
      for(int v = 0; v < n; ++v)
      {
         //Wyznaczenie wartości "u" i "v"
         //Z definicji "u" i "v" to liczby z zakresu [0;1]
         fu = (float)u / ((float)n - 1);
         fv = (float)v / ((float)n - 1);
         //Obliczenie odpowiednich wartości współrzędnych punktu na podstawie gotowych wzorów
         x = (-90 * pow(fu, 5) + 225 * pow(fu, 4) - 270 * pow(fu, 3) + 180 * pow(fu, 2) - (45 * fu)) * cos(M_PI *
fv);
         y = 160 * pow(fu, 4) - 320 * pow(fu, 3) + 160 * pow(fu, 2);
         z = (-90 * pow(fu, 5) + 225 * pow(fu, 4) - 270 * pow(fu, 3) + 180 * pow(fu, 2) - (45 * fu)) * sin(M_PI *
fv);
         //Przypisanie obliczonych wartości do odpowiednich zmiennych
         points[u][v][0] = x;
         points[u][v][1] = y;
         points[u][v][2] = z;
      }
   }
```

}

3.2.3 Funkcja Egg()

Funkcja Egg() jest funkcją biorącą udział w renderowaniu obrazu. Odpowiada za wyświetlanie jajka w podany sposób. Pobiera dane z wygenerowanych punktów przez funkcję eggPoints(). Dodatkowo przy tworzeniu jajka z trójkątów, udział bierze tablica colors, która wypełniona jest pseudolosowymi wartościami kolorów. We wszystkich rodzajach wyświetlania jajka istnieją zabezpieczenia na wypadek wyjścia poza tablicę danych.

```
//Funkcja rysująca jajko w podany sposób na podstawie wygenerowanych punktów
void Egg()
{
   //Generacja punktów
   eggPoints();
   //Model jajka stworzony z siatki punktów
   if (model == 1)
   {
      glPointSize(2.0f);
      glBegin(GL_POINTS);
      for (int u = 0; u < n; ++u)
      {
         for (int v = 0; v < n; ++v)
         {
            glVertex3fv(points[u][v]);
         }
      }
      glEnd();
   }
   //Model jajka stworzony z punktów połączonych w linie
   else if (model == 2)
   {
      //Połaczenie punktów poziomo
      glBegin(GL_LINES);
      for (int u = 0; u < n; ++u) {
         for (int v = 0; v < n; ++v) {
            glVertex3fv(points[u][v]);
            if (v != n - 1) {
               glVertex3fv(points[u][v + 1]);
            }
            else {
               glVertex3fv(points[u][0]);
            }
         }
      }
      glEnd();
      //Połaczenie punktów pionowo
      glBegin(GL_LINES);
      for (int u = 0; u < n; ++u) {
         for (int v = 0; v < n; ++v) {
            glVertex3fv(points[u][v]);
            if (u != n - 1) {
               glVertex3fv(points[u + 1][v]);
            }
            else {
               glVertex3fv(points[0][v]);
            }
         }
      }
```

```
      glEnd();
   }
   //Model jajka stworzony z punktów połączonych w trójkąty
   else if (model == 3)
   {
      for (int u = 0; u < n; ++u)
      {
         for (int v = 0; v < n; ++v)
         {
            //Zabezpieczenie przed wyjściem poza tablice
            int vdiff = 0, udiff = 0;
            if (v == n - 1) {
               vdiff = 1;
            }
            if (u == n - 1) {
               udiff = 1;
            }

            glBegin(GL_TRIANGLES);
            glColor3fv(colors[u][v + 1 - n * vdiff]);
            glVertex3fv(points[u][(v + 1 - n * vdiff)]);
            glColor3fv(colors[u + 1 - n * udiff][v]);
            glVertex3fv(points[(u + 1 - n * udiff)][v]);
            glColor3fv(colors[u][v]);
            glVertex3fv(points[u][v]);
            glEnd();

            glBegin(GL_TRIANGLES);
            glColor3fv(colors[u][v + 1 - n * vdiff]);
            glVertex3fv(points[u][v + 1 - n * vdiff]);

            glColor3fv(colors[u + 1 - n * udiff][v]);
            glVertex3fv(points[(u + 1 - n * udiff)][v]);

            glColor3fv(colors[u + 1 - n * udiff][v + 1 - n * vdiff]);
            glVertex3fv(points[(u + 1 - n * udiff)][(v + 1 - n * vdiff)]);
            glEnd();
         }
      }
   }
}
```

3.2.4 Funkcja RenderScene()

Funkcja RenderScene() jest funkcją zwrotną, która uczestniczy w renderowaniu obrazu. Składa się z funkcji czyszczących okno i macierze, odpowiedzialnych za obracanie modelu, tworzenia jajka w opisany wyżej sposób, przekazania poleceń do rysowania oraz aktualizowanie buforu.

```
// Funkcja określająca co ma być rysowane (zawsze wywoływana gdy trzeba
// przerysować scenę)
void RenderScene(void)
{

   // Czyszczenie okna aktualnym kolorem czyszczącym
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

   // Czyszczenie macierzy bieżącej
   glLoadIdentity();
```

```
    //Kolejne 3 funkcje odpowiedzialne za obracanie modelu
    glRotatef(theta[0], 1.0, 0.0, 0.0);

    glRotatef(theta[1], 0.0, 1.0, 0.0);

    glRotatef(theta[2], 0.0, 0.0, 1.0);

    //Funkcja tworzy jajko w określony sposób
    Egg();

    // Przekazanie poleceń rysujących do wykonania
    glFlush();

    // Aktualizowanie buforów (zamiana buforu z aktualnymi danymi z buforem wyświetlającym obraz)
    glutSwapBuffers();
}
```

3.2.5 Funkcja spinEgg()

Funkcja spinEgg() zmienia aktualny kąt obrócenia jajka. Umożliwia animację obracania modelu.

```
//Funkcja zmienia aktualny kąt obrócenia jajka
void spinEgg()
{

    theta[0] -= 0.01;
    if (theta[0] > 360.0) theta[0] -= 360.0;

    theta[1] -= 0.01;
    if (theta[1] > 360.0) theta[1] -= 360.0;

    theta[2] -= 0.01;
    if (theta[2] > 360.0) theta[2] -= 360.0;

    //odświeżenie zawartości aktualnego okna aplikacji
    glutPostRedisplay();
}
```

3.2.6 Funkcja keys()

Funkcja keys() reaguje w podany sposób na sygnały z klawiatury. Jej atrybuty to wciśnięty klawisz oraz współrzędne kursora myszki (niepotrzebne w tym programie).

```
//Funkcja reagująca na naciśnięty klawisz
void keys(unsigned char key, int x, int y)
{
    if (key == 'p') model = 1;
    if (key == 'w') model = 2;
    if (key == 's') model = 3;

    // przerysowanie obrazu sceny
    RenderScene();
}
```

Pozostałe funkcje zostały analogicznie użyte jak w poprzednim ćwiczeniu.

3.3 Działanie programu

Program początkowo w oknie konsolowym wyświetla menu, które prosi użytkownika o podanie liczby N oraz informacji na temat obracania obrazu. Dodatkowo wyświetlany jest sposób modyfikacji reprezentacji jajka.

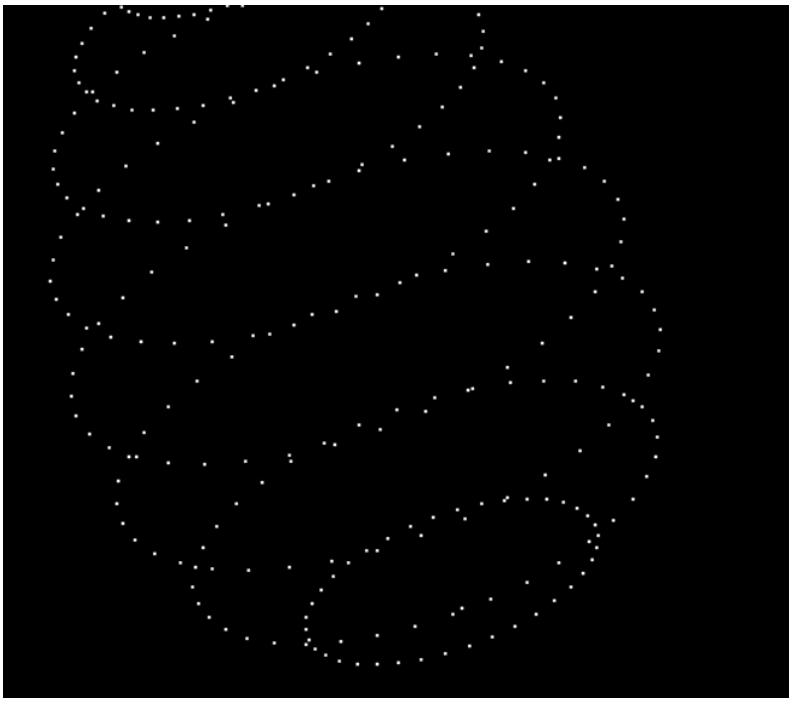
```
Podaj liczbe calkowita N, bedaca iloscia przedzialow
boku kwadratu jednostkowego dziedziny parametrycznej (najlepiej z zakresu [0;100]) :
> 20
Czy obraz ma sie obracac? (0 - NIE, 1 - TAK)
> 0

W celu zmiany trybu wyswietlania obrazu nalezy nacisnac:
'p' - siatka punktow
'w' - siatka punktow polaczona liniami
's' - siatka punktow polaczona w trojkaty
```

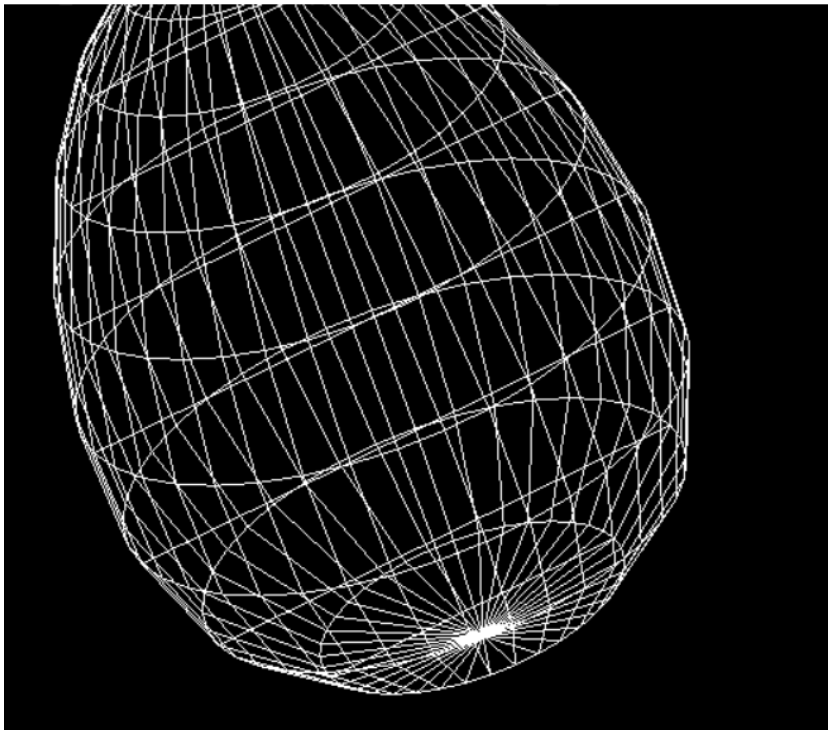
Rysunek 1 – Przykładowe działanie programu w oknie konsoli

Dalej program wyświetla okno aplikacji, na którym widoczny jest obraz, zgodny z podanymi danymi, zdolny do aktualizowania w określony przez użytkownika sposób.

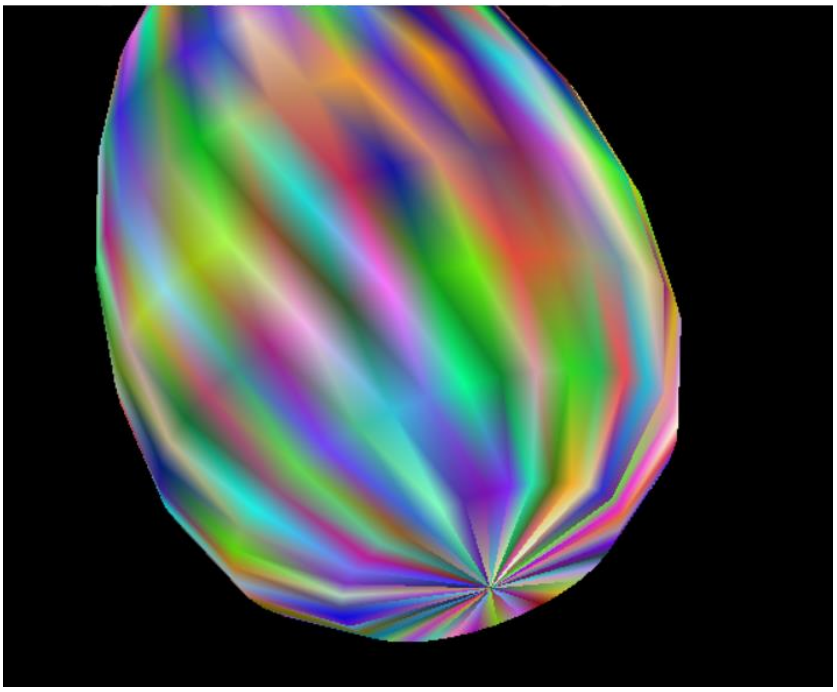
4. Wyniki



Rysunek 2 – Wyświetlenie obrazu jajka, gdzie n = 20 oraz spin = 0.



Rysunek 3 – Wyświetlenie obrazu jajka, po wciśnięciu klawisza ‘w’ gdzie n = 20 oraz spin = 0.



Rysunek 4 – Wyświetlenie obrazu jajka, po wciśnięciu klawisza ‘s’ gdzie n = 20 oraz spin = 0.

5. Wnioski

Dzięki dokumentacji oraz instrukcji ze strony prowadzącego byłem w stanie stworzyć symulację określającą zadany rodzaj jajka. Symulacja zwróciła oczekiwane wyniki, program został wykonany poprawnie.

6. Źródła

Instrukcja do ćwiczenia ze stronu kursu:
http://www.zsk.ict.pwr.wroc.pl/zsk/dyd/intinz/gk/lab/cw_3_dz/

Instrukcja do analogicznego ćwiczenia mgr inż. Szymona Datko:
https://datko.pl/GK/lab3.pdf

Dokumentacje OpenGL i GLUT:
https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/
https://www.opengl.org/resources/libraries/glut/spec3/spec3.html

Źródła były dostępne w dniu wykonywana ćwiczenia.