POLITECHNIKA WROCŁAWSKA

Grafika komputerowa i komunikacja człowiek-komputer

Kurs: INEK00012L

Sprawozdanie z ćwiczenia nr 3

OpenGL - modelowanie obiektów 3-D

Wykonał:	**Janusz Pelc 252799**
Termin:	**Np. PN/TP 7:30-10:30**
Data wykonania ćwiczenia:	**25 Października 2021**
Data oddania sprawozdania:	**3 Listopada 2021**
Ocena:	

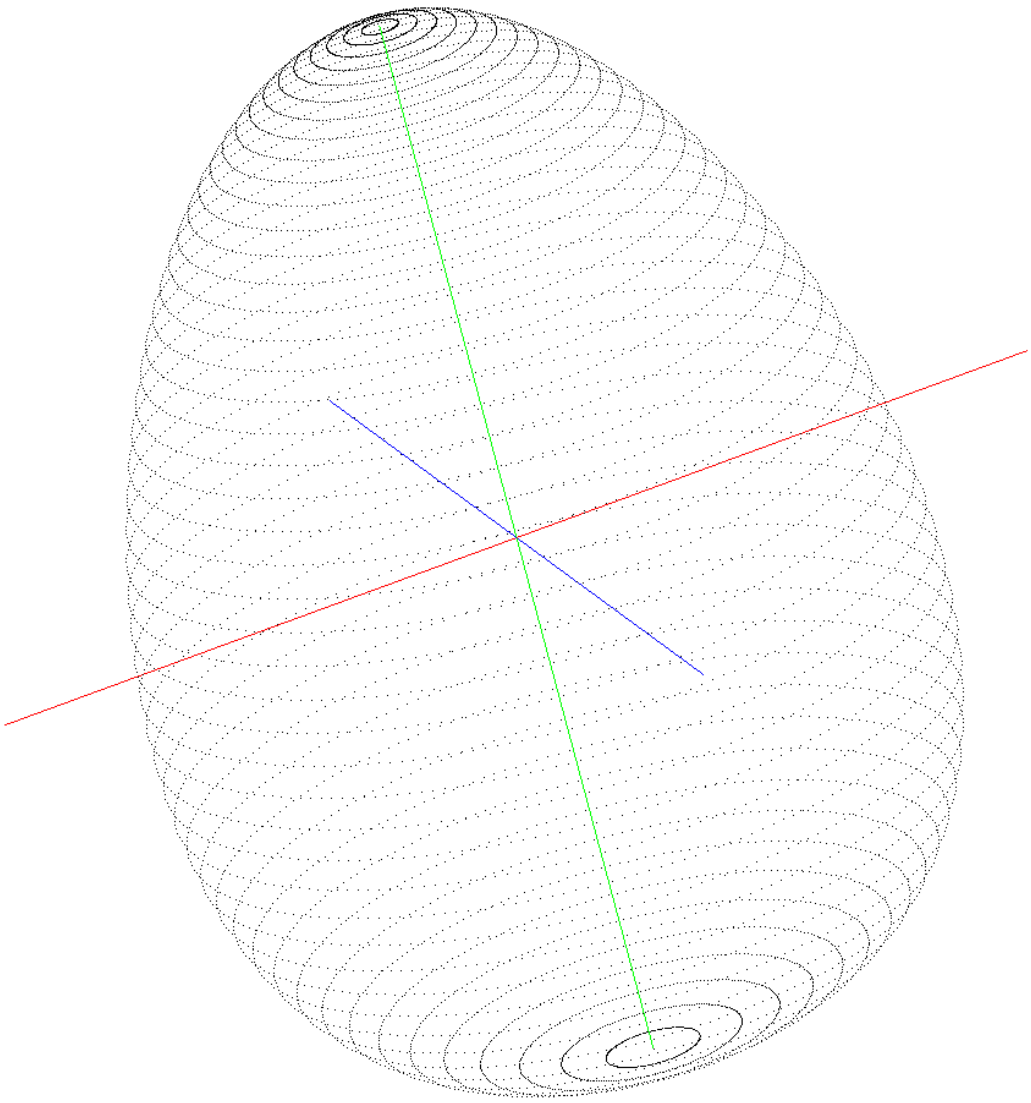
Uwagi prowadzącego:

1. Wstęp

Ćwiczenie polegało na narysowaniu modelu 3d jajka za pomocą biblioteki OpenGL. Jajko składa się z wierzchołków znajdujących się w przestrzeni trójwymiarowej. Współrzędne wierzchołków zapisane są w 3 tablicach dwuwymiarowych NxN. Zawierają one współrzędne x, y i z każdego wierzchołka. Współrzędne są wyliczane za pomocą funkcji:

$$x(u,v) = (-90u^5 + 225u^4 - 270u^3 + 180u^2 - 45u)\cos(\pi v)$$
$$y(u,v) = 160u^4 - 320u^3 + 160u^2$$
$$z(u,v) = (-90u^5 + 225u^4 - 270u^3 + 180u^2 - 45u)\sin(\pi v)$$
$$0 \le u \le 1$$
$$0 \le v \le 1$$

Argumenty u i v to współrzędne wierzchołka w tablicach podzielone przez N, aby znajdowały się w zakresie [0,1]. Ostatecznie otrzymujemy wierzchołki w następującym układzie

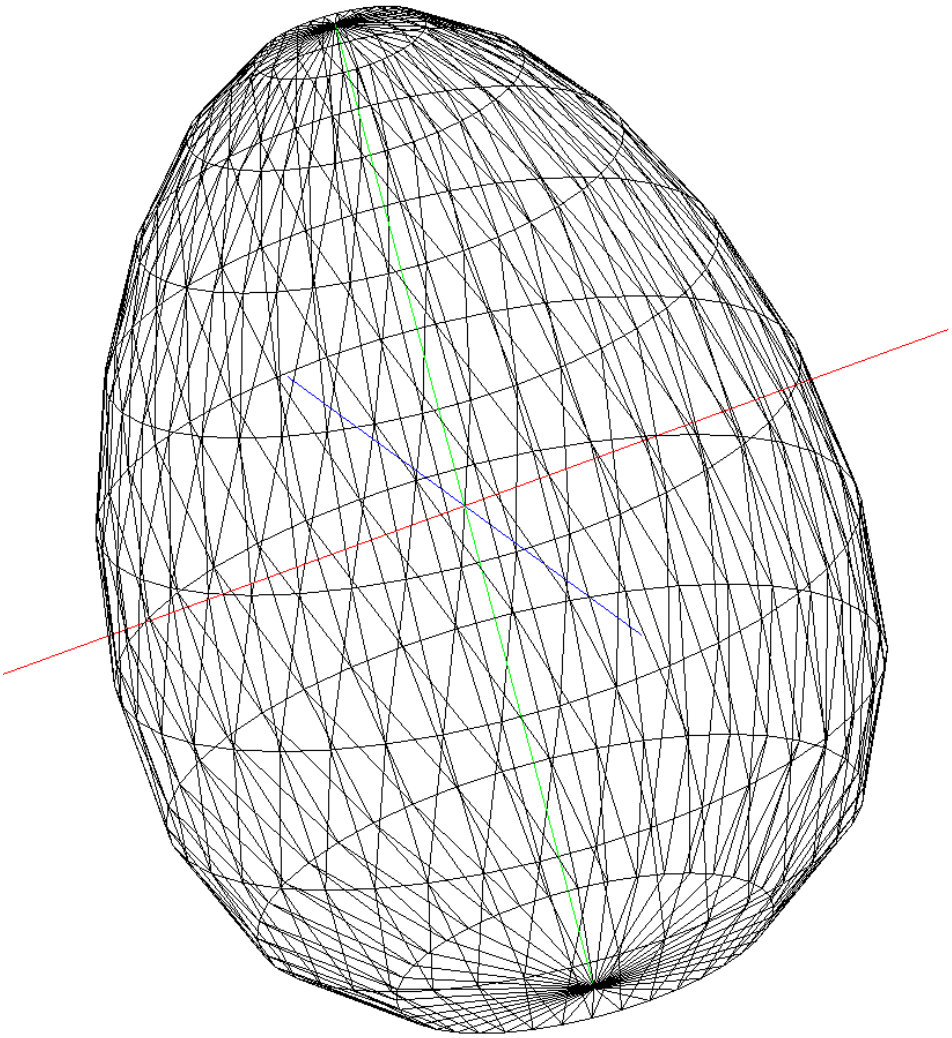


Rys1. Układ dla N=100 jako punkty

2. Program

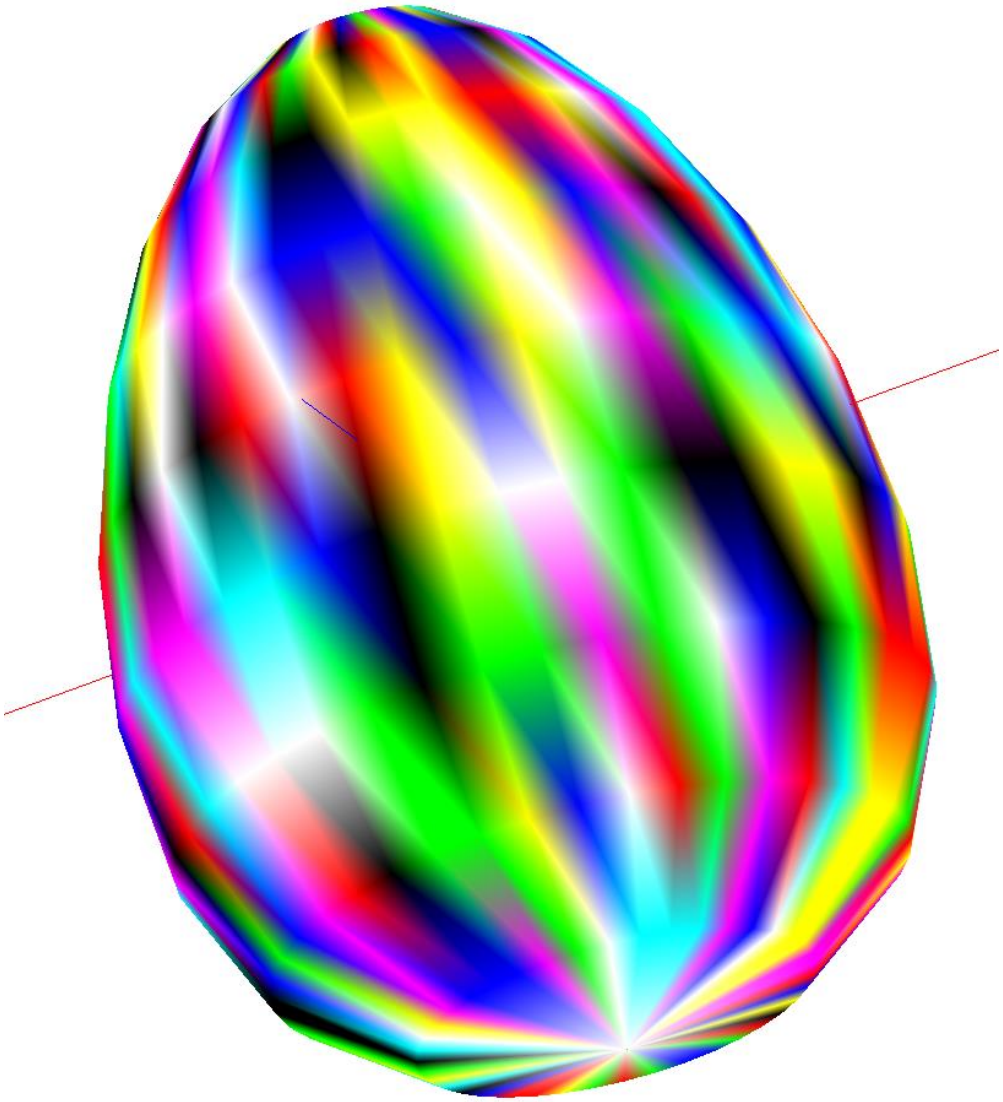
Program jest napisy w języku c++ za pomocą kompilatora Visual Studio 19 i jest bazowany na kodzie z instrukcji ćwiczenia. Główną funkcją programu jest ChangeSize(), tworzy ona i określa parametry środowiska, w którym rysowany będzie jajko. Figura powstaje w funkcji initEgg(), która wyznacza współrzędne wierzchołków oraz przypisuje im losowe kolory. RenderScene() wywołuje funkcję Egg(). , która może rysować jajko na 3 różne sposoby, zależne od wciśniętych klawiszy:

- Punkty (klawisz „p") – we współrzędnych każdego wierzchołka rysowany jest jeden punkt za pomocą GL_POINTS (patrz Rys1.)
- Siatka (klawisz „s") – miedzy sąsiadującymi wierzchołkami rysowane są linie za pomocą GL_LINES, które tworzą na powierzchni jajka siatkę z trójkątów



Rys2. Układ dla N=20 jako siatka

- Wypełnione trójkąty (klawisz „w”) – trójkąty, z których składa się siatka są wypełniane gradientami za pomocą GL_POLYGON. W funkcji Colors() każdemu wierzchołkowi przypisane jest losowy kolor i gradienty składają się z kolorów wierzchołków danego trójkąta. W efekcie otrzymujemy pomalowane jajko z płynnymi przejściami kolorów między trójkątami



Rys3. Układ dla N=20 jako wypełnione trójkąty

```
//Funkcja rysująca jajko na 3 różne sposoby
void Egg(void) {

	if (model == 1) { //punkty
		glColor3f(1, 1, 0); //kolor rysowania jest żółty
		for (int i = 0; i < N; i++) {
			for (int k = 0; k < N; k++) {
				glBegin(GL_POINTS);
				glVertex3f(x[i][k], y[i][k], z[i][k]);
				glEnd();
			}
		}
	}
```

```
	else if (model == 2) { //siatka
		glColor3f(1, 1, 0); //kolor rysowania jest żółty

		//rysowane są pionowe linie siatki jajka
		for (int i = 0; i < N; i++) {
			for (int k = 0; k < N; k++) {
				glBegin(GL_LINES); //funkcja rysuje linie
				glVertex3f(x[i][k], y[i][k], z[i][k]);
				glVertex3f(x[(i + 1) % N][k], y[(i + 1) % N][k], z[(i + 1) % N][k]);
				glEnd();

			}
		}

		//rysowane są ukośne linie siatki jajka
		for (int k = 0; k < N - 1; k++) {
			int i;
			for (i = 0; i < (N - 1) / 2; i++) {
				glBegin(GL_LINES);
				glVertex3f(x[i + 1][k], y[i + 1][k], z[i + 1][k]);
				glVertex3f(x[i][k + 1], y[i][k + 1], z[i][k + 1]);
				glEnd();
			}
			for (; i < N - 1; i++) {
				glBegin(GL_LINES);
				glVertex3f(x[i][k], y[i][k], z[i][k]);
				glVertex3f(x[i + 1][k + 1], y[i + 1][k + 1], z[i + 1][k + 1]);
				glEnd();
			}
		}

		//rysowane są ukośne linie łączące krawędzie siatki v=0 i v=N-1
		//kierunek skośnych lini jest zamieniany w połowie współrzędnej "u" tablicy
wierzchołków,
		//aby kiernek wszystkich ukośnych lini na jajku był taki sam
		int i;
		for (i = 0; i < N / 2; i++) {
			glBegin(GL_LINES);
			glVertex3f(x[i + 2][N - 1], y[i + 2][N - 1], z[i + 2][N - 1]);
			glVertex3f(x[N - 1 - i][0], y[N - 1 - i][0], z[N - 1 - i][0]);
			glEnd();
		}
		for (i; i < N; i++) {
			glBegin(GL_LINES);
			glVertex3f(x[i + 1][N - 1], y[i + 1][N - 1], z[i + 1][N - 1]);
			glVertex3f(x[N - 2 - i][0], y[N - 2 - i][0], z[N - 2 - i][0]);
			glEnd();
		}

		//rysowane są poziome linie siatki jajka
		for (int i = 0; i < N; i++) {
			for (int k = 0; k < N-1; k++) {
				glBegin(GL_LINES);
				glVertex3f(x[i][k], y[i][k], z[i][k]);
				glVertex3f(x[i][k + 1], y[i][k + 1], z[i][k + 1]);
				glEnd();
			}
		}

		//rysowane są poziome linie łączące krawędzie siatki v=0 i v=N-1
		for (int i = 0; i < N-1; i++) {
			glBegin(GL_LINES);
			glVertex3f(x[i + 1][N - 1], y[i + 1][N - 1], z[i + 1][N - 1]);
			glVertex3f(x[N - 1 - i][0], y[N - 1 - i][0], z[N - 1 - i][0]);
			glEnd();
		}

	}




	else if (model == 3) { //wypełnione trójkąty
```

```
                for (int i = 0; i < N-1 ; i++) {
                        for (int k = 0; k < N- 1; k++) {
                                //rysowane są 2 trójkąty o wspólnym boku pomiędzy 4 sąsiadującymi w
tablicy wierzchołkami
                                glBegin(GL_POLYGON);  //funkcja rysuje wielokąt
                                glColor3f(R[i][k],G[i][k],B[i][k]);
                                glVertex3f(x[i][k], y[i][k], z[i][k]);

                                glColor3f(R[i + 1][k], G[i + 1][k], B[i + 1][k]);
                                glVertex3f(x[i + 1][k], y[i + 1][k], z[i + 1][k]);

                                glColor3f(R[i][k + 1], G[i][k + 1], B[i][k + 1]);
                                glVertex3f(x[i][k + 1], y[i][k + 1], z[i][k + 1]);
                                glEnd();

                                glBegin(GL_POLYGON);
                                glColor3f(R[i + 1][k + 1], G[i + 1][k + 1], B[i + 1][k + 1]);
                                glVertex3f(x[i + 1][k + 1], y[i + 1][k + 1], z[i + 1][k + 1]);

                                glColor3f(R[i + 1][k], G[i + 1][k], B[i + 1][k]);
                                glVertex3f(x[i + 1][k], y[i + 1][k], z[i + 1][k]);

                                glColor3f(R[i][k + 1], G[i][k + 1], B[i][k + 1]);
                                glVertex3f(x[i][k + 1], y[i][k + 1], z[i][k + 1]);
                                glEnd();
                        }
                }

                //Rysowane są trójkąty pomiędzy krawędziami u=0 i u=N-1
                for (int i = 0; i < N - 1; i++) {
                        glBegin(GL_POLYGON);  //funkcja rysuje wielokąt
                        glColor3f(R[0][i], G[0][i], B[0][i]);
                        glVertex3f(x[0][i], y[0][i], z[0][i]);

                        glColor3f(R[N - 1][i + 1], G[N - 1][i + 1], B[N - 1][i+1]);
                        glVertex3f(x[N - 1][i + 1], y[N - 1][i + 1], z[N - 1][i+1]);

                        glColor3f(R[N - 1][i], G[N - 1][i], B[N - 1][i]);
                        glVertex3f(x[N - 1][i], y[N - 1][i], z[N - 1][i]);
                        glEnd();
                }

                // rysowane są trójkąty między krawędziami v=0 i v=N-1
                for (int i = 0; i < N-1 ; i++) {
                        glBegin(GL_POLYGON);  //funkcja rysuje wielokąt
                        glColor3f(R[i][N - 1], G[i][N - 1], B[i][N - 1]);
                        glVertex3f(x[i][N - 1], y[i][N - 1], z[i][N - 1]);

                        glColor3f(R[i + 1][N - 1], G[i + 1][N - 1], B[i + 1][N - 1]);
                        glVertex3f(x[i + 1][N - 1], y[i + 1][N - 1], z[i + 1][N - 1]);

                        glColor3f(R[N - 1 - i][0], G[N - 1 - i][0], B[N - 1 - i][0]);
                        glVertex3f(x[N - 1 - i][0], y[N - 1 - i][0], z[N - 1 - i][0]);
                        glEnd();
                }
                for (int i = 0; i < N - 1; i++) {
                        glBegin(GL_POLYGON);  //funkcja rysuje wielokąt
                        glColor3f(R[i+1][N - 1], G[i+1][N - 1], B[i+1][N - 1]);
                        glVertex3f(x[i+1][N - 1], y[i+1][N - 1], z[i+1][N - 1]);

                        glColor3f(R[N  -1- i][0], G[N  -1- i][0], B[N  -1- i][0]);
                        glVertex3f(x[N  -1- i][0], y[N  -1- i][0], z[N  -1- i][0]);

                        glColor3f(R[N - 2 - i][0], G[N - 2 - i][0], B[N - 2 - i][0]);
                        glVertex3f(x[N - 2 - i][0], y[N - 2 - i][0], z[N - 2 - i][0]);
                        glEnd();
                }
        }
}
```

Wciśnięte klawisze są rejestrowane przez funkcje zwrotną keys(), która wtedy zmienia model na wybrany.

```
//funkcja zwrotna wyznaczają model rysowania jajka
void keys(unsigned char key, int x, int y)
{
	if (key == 'p') model = 1; //punkty
	if (key == 's') model = 2; //siatka
	if (key == 'w') model = 3; //wypełnione trójkąty
	RenderScene(); // przerysowanie obrazu sceny
}
```

Jest ona wywoływana w funkcji main(), przez funkcję glutKeyboardFunc(keys).

Ostatecznie dzięki funkcji spinEgg(), jajko jest wprawiane w ruch obrotowy we wszystkich kierunkach wokół własnego środka. Jest ona wywoływana w funkcji main() przez funkcje glutIdleFunc(spinEgg), która bez przerwy ją wywołuje. Przy każdym wywołaniu zmieniane są kąty kolejnego obrotu. Następnie obraz zostaje przerysowany, przez funkcje RenderScene(), z jajkiem obróconym o nowy kąt.

```
// funkcja określa kąty obrotu jajka
void spinEgg()
{
	theta[0] -= 0.5; //wartość o jaką zmieniany jest kąt obrotu przy każdym wywołaniu funkcji
	if (theta[0] > 360.0) theta[0] -= 360.0; //kąt obrotu to mod[360]

	theta[1] -= 0.5;
	if (theta[1] > 360.0) theta[1] -= 360.0;

	theta[2] -= 0.5;
	if (theta[2] > 360.0) theta[2] -= 360.0;

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

3. Wnioski

Największym wyzwaniem było znalezienie sposobu na połącznie liniami wierzchołków z przeciwnych krawędzi u i v tablicy. Wymagało to zrozumienia sposobu w jaki funkcje określające współrzędne wierzchołków wyliczają ich wartości. Na początku w modelu siatki i trójkątów pojawiały się luki, gdyż nic nie było rysowane pomiędzy wierzchołkami z przeciwnych krawędzi tablicy. Wymagane było wymyślenie algorytmów wyznaczających wierzchołki między którymi rysowane będą linie i kolorowe trójkąty.

Kolejnym problemem było stworzenie płynnych przejść kolorów pomiędzy trójkątami. Wierzchołki za współrzędnymi tablicy u=0, jak i u=N/2 znajdowały się w jednym punkcie, więc przy losowaniu dla każdego wierzchołka innego koloru przejścia kolorów między trójkątami na biegunach jajka nie były płynne. Zostało to rozwiązane przez przypisanie wszystkim tym wierzchołkom koloru białego.

4. Kod źródłowy

```
/*************************************************************************************/

//  Szkielet programu do tworzenia modelu sceny 3-D z wizualizacją osi
//  układu współrzednych

/*************************************************************************************/

#define _USE_MATH_DEFINES

#include <windows.h>
#include <gl/gl.h>
#include <gl/glut.h>
#include <math.h>
#include <iostream>

using namespace std;

typedef float point3[3];

/*************************************************************************************/

const int N = 20; //rozmiar tablicy wierzchołków NxN
static GLfloat theta[] = { 0.0, 0.0, 0.0 }; // trzy kąty obrotu
int model = 2;  // 1- punkty, 2- siatka, 3 - wypełnione trójkąty

//tablice współrzędnych wierzchołków
float x[N][N];
float y[N][N];
float z[N][N];

//tablice wartości RGB wierzchołków
int R[N][N];
int G[N][N];
int B[N][N];

// Funkcja rysująca osie układu współrzędnych
void Axes(void)
{

        point3  x_min = { -5, 0.0, 0.0 };
        point3  x_max = { 5, 0.0, 0.0 };
        // początek i koniec obrazu osi x

        point3  y_min = { 0.0, -5, 0.0 };
        point3  y_max = { 0.0,  5, 0.0 };
        // początek i koniec obrazu osi y

        point3  z_min = { 0.0, 0.0, -5 };
        point3  z_max = { 0.0, 0.0,  5 };
        //  początek i koniec obrazu osi y
        glColor3f(1.0f, 0.0f, 0.0f);  // kolor rysowania osi - czerwony
        glBegin(GL_LINES); // rysowanie osi x
        glVertex3fv(x_min);
        glVertex3fv(x_max);
        glEnd();

        glColor3f(0.0f, 1.0f, 0.0f);  // kolor rysowania - zielony
        glBegin(GL_LINES);  // rysowanie osi y

        glVertex3fv(y_min);
        glVertex3fv(y_max);
        glEnd();

        glColor3f(0.0f, 0.0f, 1.0f);  // kolor rysowania - niebieski
        glBegin(GL_LINES); // rysowanie osi z

        glVertex3fv(z_min);
        glVertex3fv(z_max);
        glEnd();

}
```

```
/*************************************************************************************/
//Funkcja losująca kolor dla każdego wierzchołka
void Colors() {
	for (int i = 0; i < N; i++) {
		for (int k = 0; k < N; k++) {
			//dla wartości RGB losuje 0 lub 1
			if (x[i][k]==0 && z[i][k] == 0) {
				R[i][k] = 1;
				G[i][k] = 1;
				B[i][k] = 1;
			}
			else {
				R[i][k] = rand() % 2;
				G[i][k] = rand() % 2;
				B[i][k] = rand() % 2;
			}
		}
	}
}

void initEgg() {
	float u = 0, v = 0; //zmienne u i v wykorzystywane w funkcjach określających wierzchołki jajka;
	float fN = N; //wartość stałej N jako float
	for (int i = 0; i < N; i++) {
		u = float(i / (fN)); //zmienna zmniejszana proporcjonalnie do stałej N, aby znajdowała
się w zakresie użytej funkcji [0,1]
		for (int k = 0; k < N; k++) {
			v = float(k / (fN)); //zmienna jest zmniejszana proporcjonalnie do stałej N,
aby znajdowała się w zakresie użytej funkcji [0,1]
			//na podstawie użytej funkcji wyznaczane są wierzchołki
			x[i][k] = (-90 * pow(u, 5) + 225 * pow(u, 4) - 270 * pow(u, 3) + 180 * pow(u,
2) - 45 * u) * cos(M_PI * v);
			y[i][k] = 160 * pow(u, 4) - 320 * pow(u, 3) + 160 * pow(u, 2) - 5;
			z[i][k] = (-90 * pow(u, 5) + 225 * pow(u, 4) - 270 * pow(u, 3) + 180 * pow(u,
2) - 45 * u) * sin(M_PI * v);
		}
	}
	Colors();
	//wyznaczenie kolorów wierzchołków
}

//Funkcja rysująca jajko na 3 różne sposoby
void Egg(void) {

	if (model == 1) { //punkty
		glColor3f(1, 1, 0); //kolor rysowania jest żółty
		for (int i = 0; i < N; i++) {
			for (int k = 0; k < N; k++) {
				glBegin(GL_POINTS);
				glVertex3f(x[i][k], y[i][k], z[i][k]);
				glEnd();
			}
		}
	}

	else if (model == 2) { //siatka
		glColor3f(1, 1, 0); //kolor rysowania jest żółty

		//rysowane są pionowe linie siatki jajka
		for (int i = 0; i < N; i++) {
			for (int k = 0; k < N; k++) {
				glBegin(GL_LINES); //funkcja rysuje linie
				glVertex3f(x[i][k], y[i][k], z[i][k]);
				glVertex3f(x[(i + 1) % N][k], y[(i + 1) % N][k], z[(i + 1) % N][k]);
				glEnd();

			}
		}

		//rysowane są ukośne linie siatki jajka
		for (int k = 0; k < N - 1; k++) {
			int i;
			for (i = 0; i < (N - 1) / 2; i++) {
				glBegin(GL_LINES);
```

```
                                glVertex3f(x[i + 1][k], y[i + 1][k], z[i + 1][k]);
                                glVertex3f(x[i][k + 1], y[i][k + 1], z[i][k + 1]);
                                glEnd();
                        }
                        for (; i < N - 1; i++) {
                                glBegin(GL_LINES);
                                glVertex3f(x[i][k], y[i][k], z[i][k]);
                                glVertex3f(x[i + 1][k + 1], y[i + 1][k + 1], z[i + 1][k + 1]);
                                glEnd();
                        }
                }

                //rysowane są ukośne linie łączące krawędzie siatki v=0 i v=N-1
                //kierunek skośnych lini jest zamieniany w połowie współrzędnej "u" tablicy
wierzchołków,
                //aby kierunek wszystkich ukośnych lini na jajku był taki sam
                int i;
                for (i = 0; i < N / 2; i++) {
                        glBegin(GL_LINES);
                        glVertex3f(x[i + 2][N - 1], y[i + 2][N - 1], z[i + 2][N - 1]);
                        glVertex3f(x[N - 1 - i][0], y[N - 1 - i][0], z[N - 1 - i][0]);
                        glEnd();
                }
                for (i; i < N; i++) {
                        glBegin(GL_LINES);
                        glVertex3f(x[i + 1][N - 1], y[i + 1][N - 1], z[i + 1][N - 1]);
                        glVertex3f(x[N - 2 - i][0], y[N - 2 - i][0], z[N - 2 - i][0]);
                        glEnd();
                }

                //rysowane są poziome linie siatki jajka
                for (int i = 0; i < N; i++) {
                        for (int k = 0; k < N-1; k++) {
                                glBegin(GL_LINES);
                                glVertex3f(x[i][k], y[i][k], z[i][k]);
                                glVertex3f(x[i][k + 1], y[i][k + 1], z[i][k + 1]);
                                glEnd();
                        }
                }

                //rysowane są poziome linie łączące krawędzie siatki v=0 i v=N-1
                for (int i = 0; i < N-1; i++) {
                        glBegin(GL_LINES);
                        glVertex3f(x[i + 1][N - 1], y[i + 1][N - 1], z[i + 1][N - 1]);
                        glVertex3f(x[N - 1 - i][0], y[N - 1 - i][0], z[N - 1 - i][0]);
                        glEnd();
                }

        }

        else if (model == 3) { //wypełnione trójkąty

                for (int i = 0; i < N-1 ; i++) {
                        for (int k = 0; k < N- 1; k++) {
                                //rysowane są 2 trójkąty o wspólnym boku pomiędzy 4 sąsiadującymi w
tablicy wierzchołkami
                                glBegin(GL_POLYGON);  //funkcja rysuje wielokąt
                                glColor3f(R[i][k],G[i][k],B[i][k]);
                                glVertex3f(x[i][k], y[i][k], z[i][k]);

                                glColor3f(R[i + 1][k], G[i + 1][k], B[i + 1][k]);
                                glVertex3f(x[i + 1][k], y[i + 1][k], z[i + 1][k]);

                                glColor3f(R[i][k + 1], G[i][k + 1], B[i][k + 1]);
                                glVertex3f(x[i][k + 1], y[i][k + 1], z[i][k + 1]);
                                glEnd();

                                glBegin(GL_POLYGON);
                                glColor3f(R[i + 1][k + 1], G[i + 1][k + 1], B[i + 1][k + 1]);
                                glVertex3f(x[i + 1][k + 1], y[i + 1][k + 1], z[i + 1][k + 1]);

                                glColor3f(R[i + 1][k], G[i + 1][k], B[i + 1][k]);
                                glVertex3f(x[i + 1][k], y[i + 1][k], z[i + 1][k]);
```

```
                                glColor3f(R[i][k + 1], G[i][k + 1], B[i][k + 1]);
                                glVertex3f(x[i][k + 1], y[i][k + 1], z[i][k + 1]);
                                glEnd();
                        }
                }

                //Rysowane są trójkąty pomiędzy krawędziami u=0 i u=N-1
                for (int i = 0; i < N - 1; i++) {
                        glBegin(GL_POLYGON);  //funkcja rysuje wielokąt
                        glColor3f(R[0][i], G[0][i], B[0][i]);
                        glVertex3f(x[0][i], y[0][i], z[0][i]);

                        glColor3f(R[N - 1][i + 1], G[N - 1][i + 1], B[N - 1][i+1]);
                        glVertex3f(x[N - 1][i + 1], y[N - 1][i + 1], z[N - 1][i+1]);

                        glColor3f(R[N - 1][i], G[N - 1][i], B[N - 1][i]);
                        glVertex3f(x[N - 1][i], y[N - 1][i], z[N - 1][i]);
                        glEnd();
                }

                // rysowane są trójkąty między krawędziami v=0 i v=N-1
                for (int i = 0; i < N-1 ; i++) {
                        glBegin(GL_POLYGON);  //funkcja rysuje wielokąt
                        glColor3f(R[i][N - 1], G[i][N - 1], B[i][N - 1]);
                        glVertex3f(x[i][N - 1], y[i][N - 1], z[i][N - 1]);

                        glColor3f(R[i + 1][N - 1], G[i + 1][N - 1], B[i + 1][N - 1]);
                        glVertex3f(x[i + 1][N - 1], y[i + 1][N - 1], z[i + 1][N - 1]);

                        glColor3f(R[N - 1 - i][0], G[N - 1 - i][0], B[N - 1 - i][0]);
                        glVertex3f(x[N - 1 - i][0], y[N - 1 - i][0], z[N - 1 - i][0]);
                        glEnd();
                }
                for (int i = 0; i < N - 1; i++) {
                        glBegin(GL_POLYGON);  //funkcja rysuje wielokąt
                        glColor3f(R[i+1][N - 1], G[i+1][N - 1], B[i+1][N - 1]);
                        glVertex3f(x[i+1][N - 1], y[i+1][N - 1], z[i+1][N - 1]);

                        glColor3f(R[N  -1- i][0], G[N  -1- i][0], B[N  -1- i][0]);
                        glVertex3f(x[N  -1- i][0], y[N  -1- i][0], z[N  -1- i][0]);

                        glColor3f(R[N - 2 - i][0], G[N - 2 - i][0], B[N - 2 - i][0]);
                        glVertex3f(x[N - 2 - i][0], y[N - 2 - i][0], z[N - 2 - i][0]);
                        glEnd();
                }
        }
}

// funkcja określa kąty obrotu jajka
void spinEgg()
{
        theta[0] -= 0.5; //wartość o jaką zmieniany jest kąt obrotu przy każdym wywołaniu funkcji
        if (theta[0] > 360.0) theta[0] -= 360.0; //kąt obrotu to mod[360]

        theta[1] -= 0.5;
        if (theta[1] > 360.0) theta[1] -= 360.0;

        theta[2] -= 0.5;
        if (theta[2] > 360.0) theta[2] -= 360.0;

        glutPostRedisplay(); //odświeżenie zawartości aktualnego okna
}


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

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

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

```
        Axes();
        // Narysowanie osi przy pomocy funkcji zdefiniowanej wyżej

        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);
        //funkcje obracają obraz o wybrany kąt, wokół wybranych osi

        Egg();
        //funkcja rysuje jajko

        glFlush();
        // Przekazanie poleceń rysujących do wykonania

        glutSwapBuffers();
}

/*************************************************************************************/

// Funkcja ustalająca stan renderowania
void MyInit(void)
{

        glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        // Kolor czyszczący (wypełnienia okna) ustawiono na czarny, nieprzezroczysty
        initEgg();

}

/*************************************************************************************/

// Funkcja ma za zadanie utrzymanie stałych proporcji rysowanych
// w przypadku zmiany rozmiarów okna.
// Parametry vertical i horizontal (wysokość i szerokość okna) są
// przekazywane do funkcji za każdym razem gdy zmieni się rozmiar okna.
void ChangeSize(GLsizei horizontal, GLsizei vertical)
{

        GLfloat AspectRatio;
        // Deklaracja zmiennej AspectRatio  określającej proporcję
        // wymiarów okna
        if (vertical == 0)  // Zabezpieczenie przed dzieleniem przez 0
                vertical = 1;
        glViewport(0, 0, horizontal, vertical);
        // Ustawienie wielkości okna okna widoku (viewport)
        // W tym przypadku od (0,0) do (horizontal, vertical)
        glMatrixMode(GL_PROJECTION);
        // Przełączenie macierzy bieżącej na macierz projekcji
        glLoadIdentity();
        // Czyszcznie macierzy bieżącej
        AspectRatio = (GLfloat)horizontal / (GLfloat)vertical;
        // Wyznaczenie współczynnika  proporcji okna
        // Gdy okno nie jest kwadratem wymagane jest określenie tak zwanej
        // przestrzeni ograniczającej pozwalającej zachować właściwe
        // proporcje rysowanego obiektu.
        // Do określenia przestrzeni ograniczającej służy funkcja
        // glOrtho(...)
        if (horizontal <= vertical)

                glOrtho(-7.5, 7.5, -7.5 / AspectRatio, 7.5 / AspectRatio, 10.0, -10.0);
        else

                glOrtho(-7.5*AspectRatio, 7.5*AspectRatio, -7.5, 7.5, 10.0, -10.0);
        glRotated(30, 1, 1, 1);
        glMatrixMode(GL_MODELVIEW);

        // Przełączenie macierzy bieżącej na macierz widoku modelu
        glLoadIdentity();
        // Czyszczenie macierzy bieżącej
}

//funkcja zwrotna wyznaczają model rysowania jajka
```

```
void keys(unsigned char key, int x, int y)
{
        if (key == 'p') model = 1; //punkty
        if (key == 's') model = 2; //siatka
        if (key == 'w') model = 3; //wypoełnione trójkąty
        RenderScene(); // przerysowanie obrazu sceny
}

/*************************************************************************************/

// Główny punkt wejścia programu. Program działa w trybie konsoli
void main(void)
{

        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

        glutInitWindowSize(600, 600);
        //wyznaczenie wstepnej wielkości okna w pikselach

        glutCreateWindow("Układ współrzędnych 3-D");

        glutDisplayFunc(RenderScene);
        // Określenie, że funkcja RenderScene będzie funkcją zwrotną
        // (callback function).  Bedzie ona wywoływana za każdym razem
        // gdy zajdzie potrzba przeryswania okna
        glutReshapeFunc(ChangeSize);
        // Dla aktualnego okna ustala funkcję zwrotną odpowiedzialną
        // zazmiany rozmiaru okna
        MyInit();
        // Funkcja MyInit() (zdefiniowana powyżej) wykonuje wszelkie
        // inicjalizacje konieczne  przed przystąpieniem do renderowania
        glEnable(GL_DEPTH_TEST);
        // Włączenie mechanizmu usuwania powierzchni niewidocznych
        glutKeyboardFunc(keys);
        glutIdleFunc(spinEgg);
        glutMainLoop();
        // Funkcja uruchamia szkielet biblioteki GLUT
}

/*************************************************************************************/
```