POLITECHNIKA WROCŁAWSKA

Instytut Informatyki, Automatyki i Robotyki

Zakład Systemów Komputerowych

Wprowadzenie do grafiki komputerowej

Kurs: INE4234L

Sprawozdanie z ćwiczenia nr 3

Modelowanie i wizualizacja obiektu 3-D opisanego równaniem parametrycznym

Wykonał:	**Kamil Kamyszek**
Termin:	**PT/NP 11.00-14.00**
Data wykonania ćwiczenia:	**26.10.2018r.**
Data oddania sprawozdania:	**09.11.2018r.**
Ocena:	

Uwagi prowadzącego:

1 Wstęp

Na trzecim laboratorium z grafiki studenci mieli zaznajomić się z prostym modelowaniem 3D opisanym równaniem parametrycznym. Do zrozumienia modelowania 3D trzeba było zapoznać się z nowymi funkcjami bibliotek OpenGL i GLUT przedstawionymi w instrukcji.

2 Przebieg Laboratorium

Najpierw narysowany został układ współrzędnych, a na nim wygenerowany imbryk do parzenia herbaty. To ćwiczenie pozwoliło poznać podstawowe funkcje i genezę tworzenia obiektów trójwymiarowych. Po zapoznaniu się z kolejnymi funkcjami bibliotek graficznych, możliwe było obrócenie obiektu na układzie współrzędnych o zadaną ilość stopni. Kolejne polecenie dotyczyło stworzenia trójwymiarowego jajka obracającego się wokoło układu współrzędnych. Na stronie ZSK były podane trzy równania parametryczne odpowiadającego trzem osiom układu dzięki czemu możliwe było wygenerowanie podanej w poleceniu bryły.

3 Zadanie do samodzielnego wykonania

Model jajka, został opisany równaniem parametrycznym (Rysunek 1):

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

Rysunek 1 Osie x,y,z jajka opisane równaniem parametrycznym

Oto kod opisujący najważniejsze funkcje programu:

//Do stworzonej tablicy trójwymiarowej NxN muszę wprowadzić wartości dla osi x, y i z:

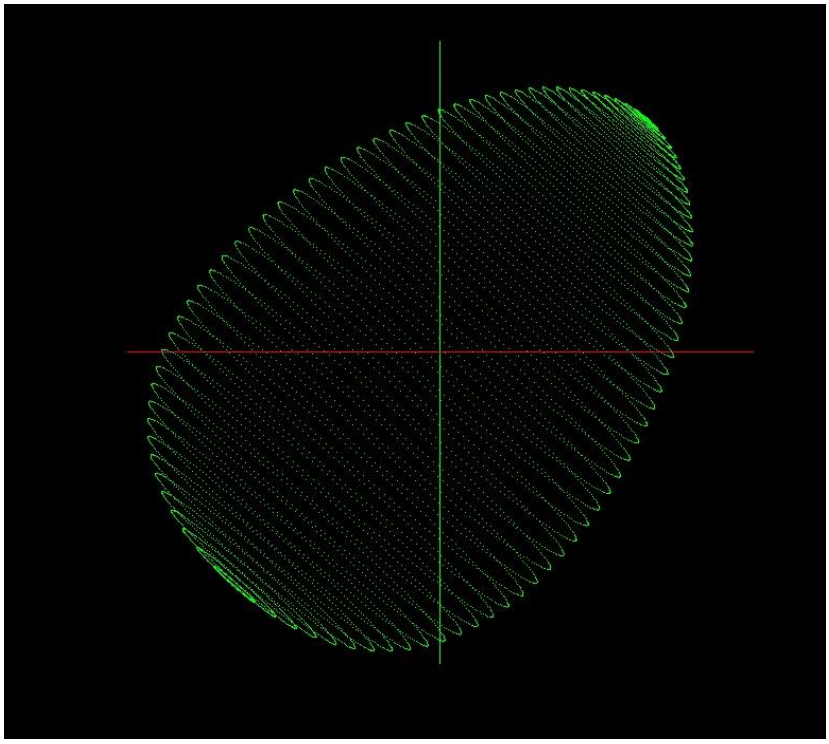
```
for (int i = 0; i<N; i++)
              for (int j = 0; j<N; j++)
{

       u = (float)i / (N - 1); // Wartość u musi zawierać się między 0 a 1
       v = (float)j / (N - 1); // Wartość v musi zawierać się między 0 a 1

tab[i][j][0] = (-90 * pow(u, 5) + 225 * pow(u, 4) - 270 * pow(u, 3) + 180 *pow(u, 2) - 45 *
u)*cos(M_PI * v);           //Wartość dla x wygenerowana z równania podanego na Rysunku 1

tab[i][j][1] = 160 * pow(u, 4) - 320 * pow(u, 3) + 160 * pow(u, 2) - 5;
                            //Wartość dla y wygenerowana z równania podanego na Rysunku 1
tab[i][j][2] = (-90 * pow(u, 5) + 225 * pow(u, 4) - 270 * pow(u, 3) + 180 * pow(u, 2) - 45
* u)*sin(M_PI * v);         //Wartość dla z wygenerowania z równania podanego na Rysunku 1
}
```

Podając u i v z funkcji rand() generującej losową liczbę od 1->100 i dzieląc przez 100 niestety nie generowało na końcowym etapie pożądanej figury i musiało zostać zastąpione generowaniem wartości 0 do 1 jak powyżej.

Pętla generująca ciąg punktów tworzących obrys jajka w przestrzeni. Efekt: **Jajko wygenerowane z punktów**

```
if (model == 0) // model wyświetlania jajka
	{
		glBegin(GL_POINTS); //Funkcja generująca punkt
		for (int i = 0; i < N - 1; ++i) {
	for (int j = 0; j < N - 1; ++j)
		{
			glVertex3fv(tab[i][j]); //Generowanie punktów na osi dla x, y i z
		}
	}
		glEnd();
	}
```

Funkcja glBegin(GL_POINTS) pozwala na wygenerowanie punktu w przestrzeni podając jego wartość x, y i z
Funkcja glVertex3fv pozwala na stworzenie wskaźnika do tablicy 3 elementów (tutaj x, y, z)
Efekt:

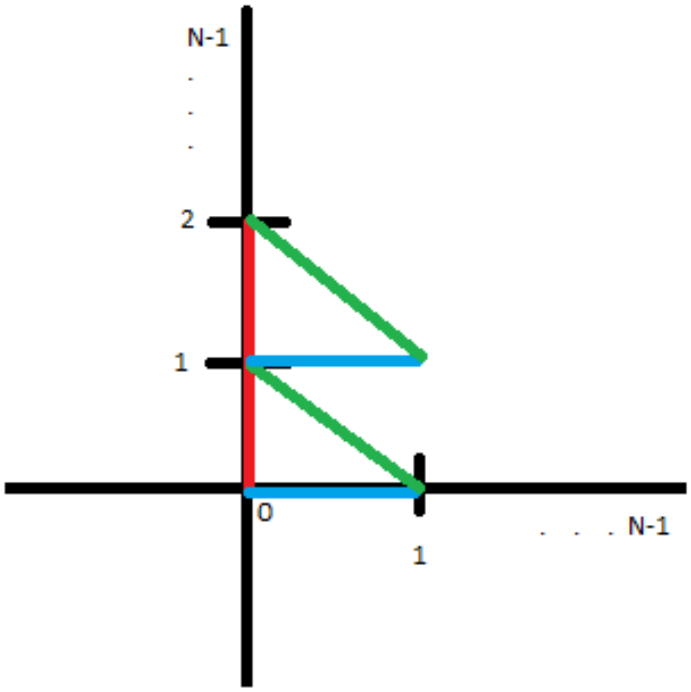


Rysunek 2 Jajko wygenerowane z punktów

Pętla generująca linie tworzące jajko z trójkątnej siatki. Efekt: **Jajko złożone z siatki trójkątów**

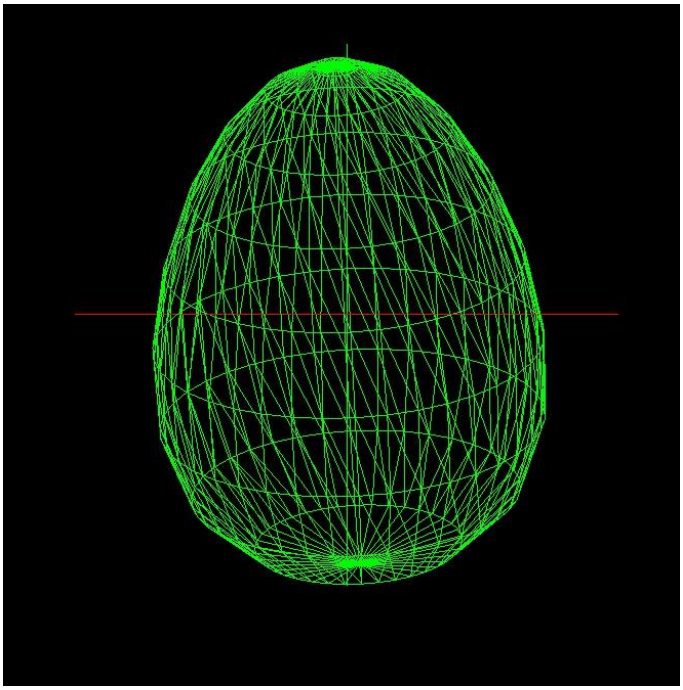
```
if (model == 1) // model wyświetlania jajka
	{

	for (int i = 0; i < N-1 ; i++) {
		for (int j = 0; j < N-1 ; j++)//N-1 żeby nie wyjść poza tablice w pętli
			{
			glBegin(GL_LINES);

			glVertex3fv(tab[i][j]);		//np. P (0,0)
			glVertex3fv(tab[i][j+1]);	//np. P (0,1)

			glVertex3fv(tab[i][j]);		//np. P (0,0)
			glVertex3fv(tab[i+1][j]);	//np. P (1,0)

			glVertex3fv(tab[i+1][j]);	//np. P (1,0)
			glVertex3fv(tab[i][j+1]);	//np. P (0,1)
			}
		}
		glEnd();
	}
```

Funkcja glBegin(GL_LINES) pozwala na wygenerowanie linii podając punkt początkowy i końcowy. Na rysunku schematyczny opis dla dwóch wymiarów (**Generowanie siatki trójkątów**):



Rysunek 3 Generowanie siatki trójkątów

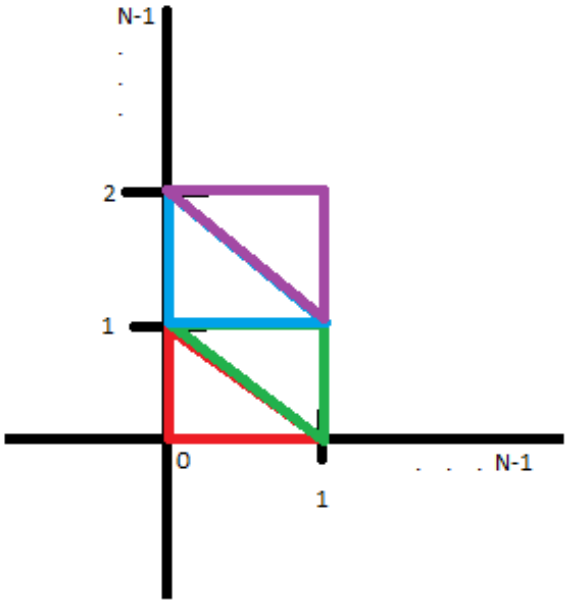
Efekt:



Rysunek 4 Jajko złożone z siatki trójkątów

Pętla generująca i kolorująca jajko z trójkątów wypełnionych kolorami. Poniżej rysunek obrazujący działanie pętli na modelu 2D (**Generowanie trójkątów**). Efekt: **Pokolorowane jajko**

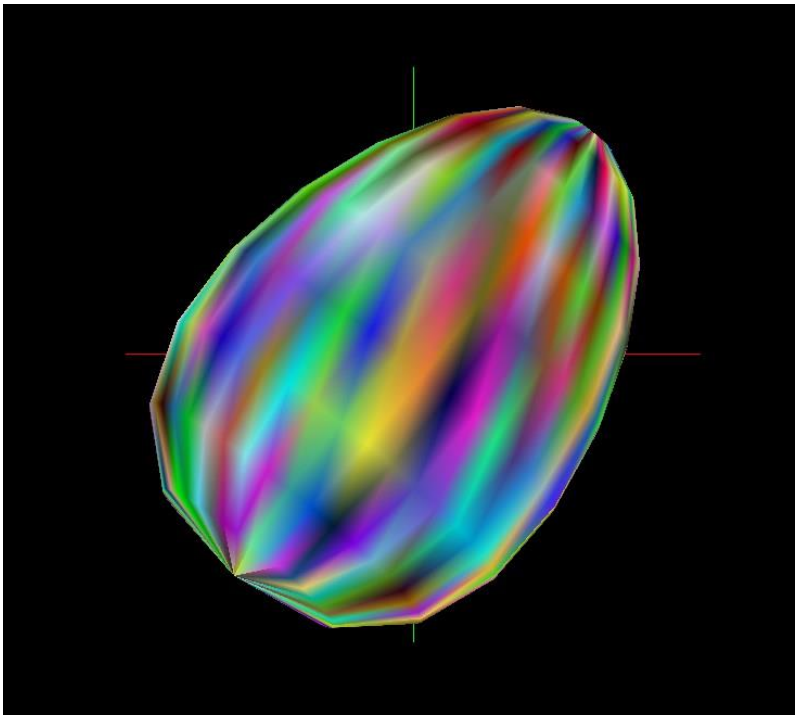
```
if (model == 2)  // model wyświetlania jajka
	{
		for (int i = 0; i < N-1 ; i++) {
			for (int j = 0; j < N-1 ; j++)
			{
				glBegin(GL_TRIANGLES);			//Generuje trojkąt

				glColor3fv(colors_tab[i][j+1]);	//Tablica NxN kolorów
				glVertex3fv(tab[i][j + 1]);		//np. P (0,1)

				glColor3fv(colors_tab[i+1][j]);
				glVertex3fv(tab[i + 1][j]);		//np. P (1,0)

				glColor3fv(colors_tab[i+1][j+1]);
				glVertex3fv(tab[i + 1][j + 1]);	//np. P (1,1)



				glColor3fv(colors_tab[i+1][j]);
				glVertex3fv(tab[i + 1][j]);		//np. P (1,0)


				glColor3fv(colors_tab[i][j+1]);
				glVertex3fv(tab[i][j + 1]);		//np. P (0,1)


				glColor3fv(colors_tab[i][j]);
				glVertex3fv(tab[i][j]);			//np. P (0,0)

				glEnd();
			}
		}}
```



Rysunek 5 Generowanie trójkątów

W przypadku generowania całej struktury wypełnionej kolorem musiała zostać stworzona dodatkowa **tabela przechowująca losowe wygenerowane kolory**. Aby przechodziły one gładko między trójkątami trzeba było wygenerować te same odcienie na końcach wierzchołków, które się ze sobą stykają.

Efekt:



Rysunek 6 Pokolorowane jajko

```
float createRandom ()
{
	float randomNumber;
	double const generateRandom = rand () % 10;	//liczba od 1 do 10

	randomNumber = generateRandom / 10;		//zwraca liczbę od 0 do 1

	return randomNumber;
}
void randomcolors () {
	for (int i = 0; i < N; i++)
	{
		for (int j = 0; j < N; j++)
		{

			for (int l = 0; l < 3; l++)
			{
				colors_tab[i][j][l] = createRandom ();
			}

		}
	}
	}
```

Rysunek 7 tabela przechowująca losowe wygenerowane kolory

Jajko obracało się wokół osi według funkcji podanej w instrukcji.

3 Wnioski

Dzięki zamieszczonej instrukcji na stronie zsk, wykonanie zadania nie sprawiło większych trudności i pozwoliło oswoić się z modelowaniem 3D wykorzystując biblioteki OpenGL i GLUT. Problemem, którego niestety nie udało się rozwiązać było usunięcie bądź zamaskowanie „szwa” w miejscu, w którym jajko „składa” się, gdyż tam kolory nie przechodzą łagodnie i powstaje kreska. Efekt ten psuje założenia grafiki komputerowej, która ma za zadanie oszukanie ludzkiego oka i pokazanie jak najbardziej zbliżonego do rzeczywistego wyglądu obiektu. Niemniej jednak efekt końcowy ma zastosowanie chociażby przy tworzeniu animacji obrotu planet wokół Słońca. Pokolorowane jajko można również wydrukować i powiesić jako dekoracja na Wielkanoc.