POLITECHNIKA WROCŁAWSKA
Instytut Informatyki, Automatyki i Robotyki
Zakład Systemów Komputerowych

Wprowadzenie do grafiki komputerowej

Kurs: INE4234L

Sprawozdanie z ćwiczenia nr 5

OpenGL - oświetlanie scen 3-D

Wykonał:	**Kamil Kamyszek**
Termin:	**PT/NP 11.00-14.00**
Data wykonania ćwiczenia:	**07.12.18r.**
Data oddania sprawozdania:	**21.12.18r.**
Ocena:	

Uwagi prowadzącego:

1 Wstęp

Na piątym laboratorium z grafiki komputerowej studenci mieli za zadanie zapoznać się z oświetlaniem scen w 3D. Do zrozumienia tego tematu potrzebne było uważne przestudiowanie instrukcji zamieszczonej przez prowadzącego na stronie ZSK, jak również zaznajomienie się z nowo poznanymi funkcjami bibliotek OpenGL i GLUT.

2 Przebieg Laboratorium

Laboratorium rozpoczęto od oświetlenia czajnika z poprzednich zajęć światłem białym. To ćwiczenie umożliwiło zapoznanie się z możliwościami bibliotek graficznych używanych przez studentów jak również z funkcjami potrzebnymi do uzyskania zadowalających efektów. Drugim zadaniem było oświetlenie jajka wykonanego na poprzednich zajęciach. Efekt nie był zadowalający i dlatego kolejnym co trzeba było zrobić tym razem samodzielnie było stworzenie jednego źródła światła dla jajka, które będzie wyglądać bardziej realistycznie.

3 Zadania do samodzielnego wykonania

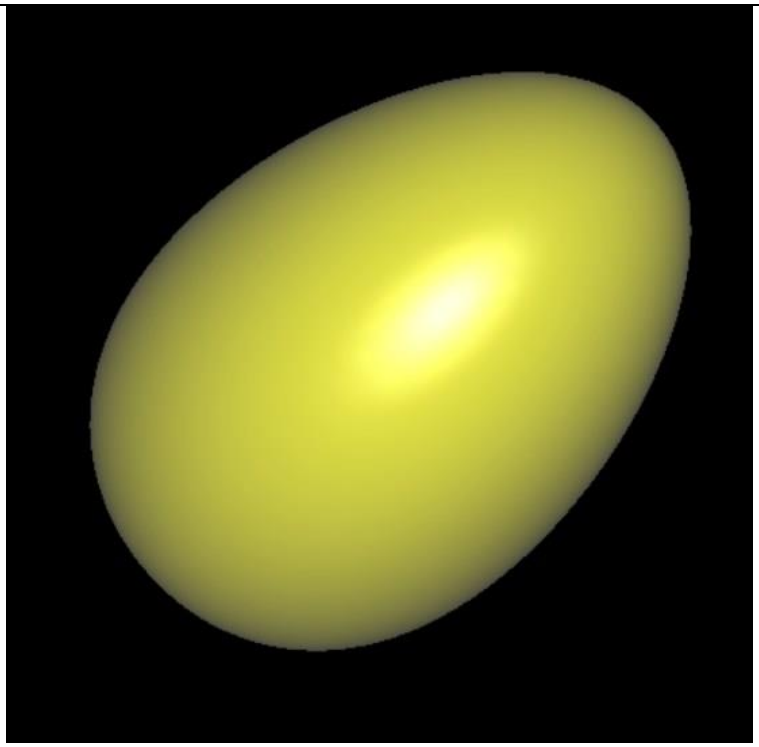
Oświetlenie jajka bardziej realistycznie wymagało zapoznania się z definicja wektorów normalnych i instrukcji zamieszczonej na stronie zsk. Umożliwiło to stworzenie tak wyglądającego modelu: **Jajko z jednym źródłem światła**

Do funkcji tworzącej jajko trzeba było dodać wektory normalne (według instrukcji na stronie zsk):

```
float ux, uz, uy, vz, vy, vx, length;

// obliczenie wartosci wektorów normalnych z instrukcji
ux = (-450 * pow(u, 4) + 900 * pow(u, 3) - 810 * pow(u, 2) + 360 * u - 45)*cos(3.14*v);uy = (640 *
pow(u, 3) - 960 * pow(u, 2) + 320 * u);
uz = (-450 * pow(u, 4) + 900 * pow(u, 3) - 810 * pow(u, 2) + 360 * u - 45)*sin(3.14*v);

vx = 3.14*(90 * pow(u, 5) - 225 * pow(u, 4) + 270 * pow(u, 3) - 180 * pow(u, 2) + 45 * u)*sin(3.14*v);
                vy = 0;
                vz = -3.14*(90 * pow(u, 5) - 225 * pow(u, 4) + 270 * pow(u, 3) - 180 * pow(u, 2) + 45 *
u)*cos(3.14*v);

vectorNorm[i][j][0] = uy * vz - uz * vy; //opisanie wektorów normalnych według instrukcji
vectorNorm[i][j][1] = uz * vx - ux * vz;
vectorNorm[i][j][2] = ux * vy - uy * vx;

length = sqrt(vectorNorm[i][j][0] * vectorNorm[i][j][0] + vectorNorm[i][j][1] * vectorNorm[i][j][1] +
vectorNorm[i][j][2] * vectorNorm[i][j][2]);

//Warunki ustalające oswietlenie jajka:
 if (i < N / 2)                          //jeśli jesteśmy w pierwszej połowie jajka
{
        vectorNorm[i][j][0] = (uy*vz - uz * vy) / length;
        vectorNorm[i][j][1] = (uz*vx - ux * vz) / length;
        vectorNorm[i][j][2] = (ux*vy - uy * vx) / length;
}
else if (i > N / 2)                     //jeśli jesteśmy w drugiej połowie jajka
{
        vectorNorm[i][j][0] = -1 * (uy*vz - uz * vy) / length;
        vectorNorm[i][j][1] = -1 * (uz*vx - ux * vz) / length;
        vectorNorm[i][j][2] = -1 * (ux*vy - uy * vx) / length;
}
else if (i == 0 || i == N)              //Warunki graniczne
{
        vectorNorm[i][j][0] = 0;
        vectorNorm[i][j][1] = -1;
        vectorNorm[i][j][2] = 0;
}
else
{
        vectorNorm[i][j][0] = 0;
        vectorNorm[i][j][1] = 1;
        vectorNorm[i][j][2] = 0;
}
```

Aby umożliwić stworzenie jajka trzeba było przerobić funkcję **MyInit()**. Poniższy kod nie jest skomentowany, gdyż został opisany dokładnie na stronie instrukcji.

```
void MyInit(void)
{
	glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

	GLfloat mat_ambient[] = { 1.0,1.0, 1.0, 1 };
	GLfloat mat_diffuse[] = { 1.0, 1.0, 1.0, 1 };
	GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
	GLfloat mat_shininess = { 100.0 };
	GLfloat light_position[] = { 0.0, 0.0, 0.0, 1.0 };
	GLfloat light_ambient[] = {0.1, 0.1, 0.1, 1.0 };
	GLfloat light_diffuse[] = { 1.0, 1.0, 0.0, 1.0 };
	GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
	GLfloat att_constant = { 1.0 };
	GLfloat att_linear = { (GLfloat) 0.05 };
	GLfloat att_quadratic = { (GLfloat) 0.001 };

	glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
	glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
	glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
	glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);

	glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
	glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
	glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
	glLightfv(GL_LIGHT0, GL_POSITION, light_position);

	glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, att_constant);
	glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, att_linear);
	glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, att_quadratic);

	glShadeModel(GL_SMOOTH);
	glEnable(GL_LIGHTING);
	glEnable(GL_LIGHT0);
	glEnable(GL_DEPTH_TEST);
}
```



1 Jajko z jednym źródłem światła

Kolejnym zadaniem do wykonania było stworzenie drugiego źródła światła i umożliwienie poruszania poszczególnymi źródłami za pomocą myszki. Na dodatek jajko miało znajdować się w środku układu współrzędnych. Efekt: **Rysunek 2 Jajko z dwoma** źródłami światła

Aby umożliwić ruszanie dwoma źródłami trzeba było przedefiniować metodę **mouse()** i **motion():**

```
void Mouse(int btn, int state, int x, int y)
{
	if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
	{
		x_pos_old = x;		// przypisanie aktualnie odczytanej pozycji kursora jako pozycji poprzedniej
		y_pos_old = y;
		status = 1;		// Wciśniety lewy przycisk myszy
	}

	else if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
	{
		x_pos_old = x;		// przypisanie aktualnie odczytanej pozycji kursora jako pozycji poprzedniej
		y_pos_old = y;
		status = 2;		 //Wciśnięty prawy przycisk myszy
	}
	else

		status = 0;		// nie został wciśnięty żaden przycisk myszy
}
```

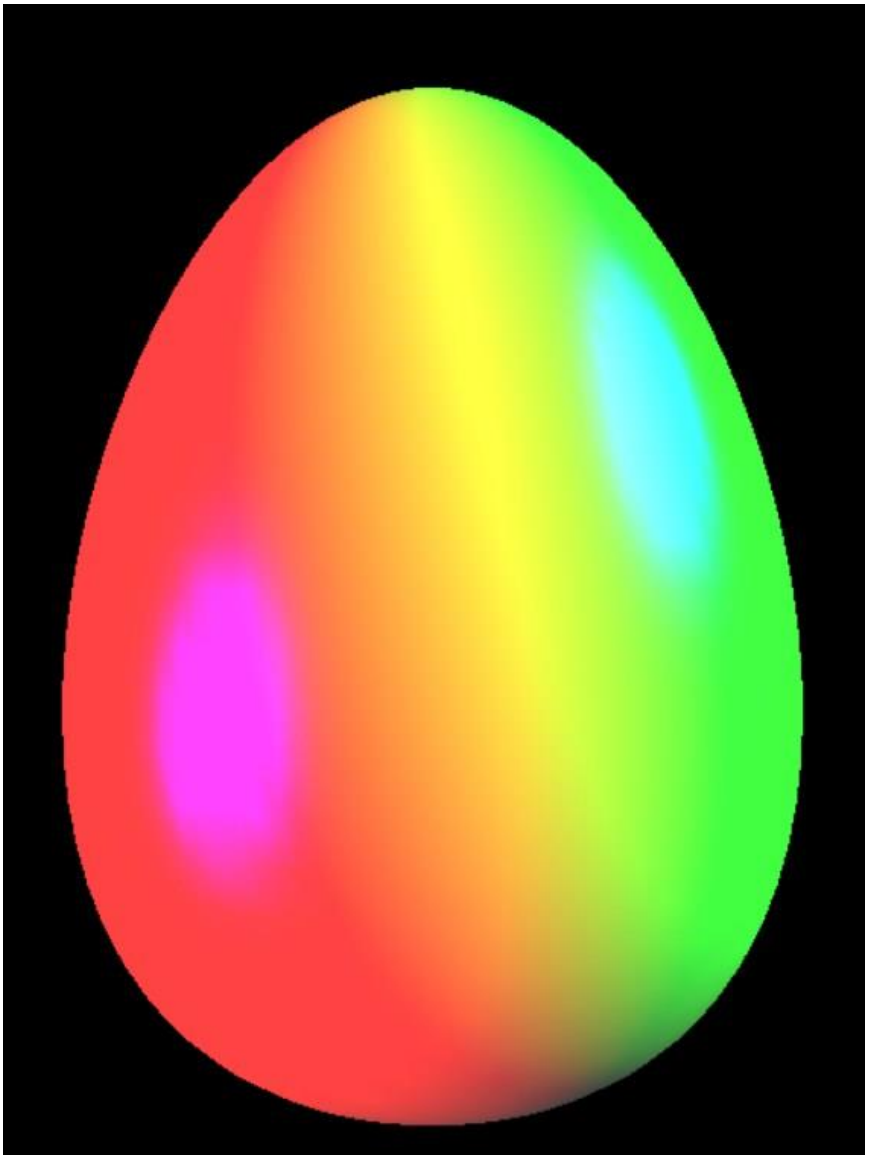
```
void Motion(GLsizei x, GLsizei y)
{
	delta_x = x - x_pos_old;		// obliczenie różnicy położenia kursora myszy
	delta_y = y - y_pos_old;
	x_pos_old = x;  // przypisanie aktualnie odczytanej pozycji kursora jako pozycji poprzedniej
	y_pos_old = y;
	glutPostRedisplay();		// przerysowanie obrazu sceny
}
```

Dodany kod do metody **RenderScene()** umożliwiający zmianę położenia światła:

```
	if(status==1){
		theta += delta_x * pix2angle;	 // modyfikacja kąta obrotu o kat proporcjonalny
		fi += delta_y * pix2angle;
	}

	if (status == 2) {
		theta2 += delta_x * pix2angle;	 // modyfikacja kąta obrotu o kat proporcjonalny
		fi2 += delta_y * pix2angle;
	}
	float pi = 3.14;
	if (fi > 2 * pi) fi = 2 * pi;		//Dziedzina dla pi i fi
	if (theta > 2 * pi) theta = 2 * pi;
	if (fi < 0) fi = 0;
	if (theta < 0) theta = 0;
	if (fi2 > 2 * pi) fi2 = 2 * pi;
	if (theta2 > 2 * pi) theta2 = 2 * pi;
	if (fi2 < 0) fi2 = 0;
	if (theta2 < 0) theta2 = 0;

	//Ustalenie położenia źródła swiatła pierwszego według instrukcji
	light_position[0] = zoom * cos(theta2) * cos(fi2);
	light_position[1] = zoom * sin(fi2);
	light_position[2] = zoom * sin(theta2) * cos(fi2);
	//Ustalenie położenia źródła swiatła drugiego według instrukcji
	light_positionSecond[0] = zoom * cos(theta) * cos(fi);
	light_positionSecond[1] = zoom * sin(fi);
	light_positionSecond[2] = zoom * sin(theta) * cos(fi);
	glLightfv(GL_LIGHT0, GL_POSITION, light_position);
	glLightfv(GL_LIGHT1, GL_POSITION, light_positionSecond);
```

Zmodyfikowaniu musiała ulec również metoda **MyInit()**, gdyż trzeba było dodać drugie źródło światła:

```
void MyInit(void)
{
	glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
	GLfloat mat_ambient[] = { 1.0,1.0, 1.0, 1.0 };
	GLfloat mat_diffuse[] = { 5.0, 5.0, 5.0, 1.0 };
	GLfloat mat_specular[] = { 10.0, 10.0, 10.0, 1.0 };
	GLfloat mat_shininess = { 200.0 };
	//Definicja pierwszego źródła światła
	GLfloat light_position[] = { 0.0, 0.0, -10.0,1.0 };
	GLfloat light_ambient[] = { 0.1, 0.1, 0.1, 1.0 };
	GLfloat light_diffuse[] = { 0.0, 1.0, 0.0, 1.0 };
	GLfloat light_specular[] = {0.0, 1.0, 1.0, 0.0 };
	GLfloat att_constant = { 1.0 };
	GLfloat att_linear = { (GLfloat) 0.05 };
	GLfloat att_quadratic = { (GLfloat) 0.001 };
	//Definicja drugiego źródła światła
	GLfloat light_position2[] = { 0.0, 0.0, 0.0,1.0 };
	GLfloat light_ambient2[] = { 0.1, 0.1, 0.1, 1.0 };
	GLfloat light_diffuse2[] = { 1.0, 0.0,0.0, 10.0 };
	GLfloat light_specular2[] = { 1.0, 0.0,1.0, 10.0 };
	GLfloat att_constant2 = { 1.0 };
	GLfloat att_linear2 = { (GLfloat) 0.05 };
	GLfloat att_quadratic2 = { (GLfloat) 0.001 };
	glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
	glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
	glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
	glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);

	glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
	glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
	glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
	glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, att_constant);
	glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, att_linear);
	glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, att_quadratic);
	glLightfv(GL_LIGHT0, GL_POSITION, light_position);
	glLightfv(GL_LIGHT1, GL_AMBIENT, light_ambient2);
	glLightfv(GL_LIGHT1, GL_DIFFUSE, light_diffuse2);
	glLightfv(GL_LIGHT1, GL_SPECULAR, light_specular2);
	glLightf(GL_LIGHT1, GL_CONSTANT_ATTENUATION, att_constant2);
	glLightf(GL_LIGHT1, GL_LINEAR_ATTENUATION, att_linear2);
	glLightf(GL_LIGHT1, GL_QUADRATIC_ATTENUATION, att_quadratic2);
	glLightfv(GL_LIGHT1, GL_POSITION, light_position2);

	glShadeModel(GL_SMOOTH);
	glEnable(GL_LIGHTING
	glEnable(GL_LIGHT0);
	glEnable(GL_LIGHT1);
	glEnable(GL_DEPTH_TEST);
}
```



Rysunek 2 Jajko z dwoma źródłami światła

4 Wnioski

Dzięki zamieszczonej instrukcji na stronie zsk, wykonanie zadania nie sprawiło większych trudności i pozwoliło oswoić się z zagadnieniem oświetlenia scen 3D. Można było to przetestować w przypadku obiektów wygenerowanych przez studentów na zajęciach. Biblioteki graficzne OpenGL i GLUT w dużym stopniu ułatwiają użytkownikowi zaprogramowanie oświetlenia, dzięki czemu można w krótkim czasie oświetlić wykonany obiekt. Nauczenie się oświetlania obiektów ma bardzo wiele zastosowań w grafice 3D i pozwala na zwiększanie realizmu tworzonych struktur.