Sprawozdanie z ćwiczenia 5
OpenGL – oświetlanie scen 3D
Maksymilian Iwanow
209946
Poniedziałek 10.15

Celem ćwiczenia była ilustracja możliwości oświetlania obiektów na scenach 3D z wykorzystaniem biblioteki OpenGL oraz rozszerzeniem GLUT.

Do wykonania były następujące dwa zadania:

Zadanie 1

Należy zmodyfikować ostatni program wykonany w ćwiczeniu 2 (obracające się jajko) przez:

- Wprowadzenie na scenę jednego źródła światła (np. w kolorze żółtym).
- Zdefiniowane materiału z jakiego wykonane jest jajko (np. białe, połyskujące).
- Dodanie dla poszczególnych wierzchołków modelu jajka informacji o wektorach normalnych wyliczonych na podstawie zależności podanych w poprzednim punkcie.

Rysunek jaki pojawi się na ekranie po wprowadzeniu tych zmian powinien wyglądać mniej więcej tak:

Zadanie 2

Zadanie polega na napisaniu programu, pozwalającego na oświetlanie modelu jajka przy pomocy dwóch źródeł barwnego światła i umożliwieniu manipulowania położeniem źródeł przy pomocy myszy. Wizualizacja modelu i sterowanie położeniem źródeł światła powinno odbywać się przy następujących założeniach:

- Jajko znajduje się jest w środku układu współrzędnych.
- Punkt, w którym umieszczone jest źródło może poruszać się po powierzchni sfery o promieniu ***R*** i środku leżącym w środku układu współrzędnych.
- Sterowanie położeniem źródła światła odbywać się (tak jak w ćwiczeniu 4) przy pomocy dwóch kątów. Pierwszy z nich określa kierunek świecenia na obiekt i nosi nazwę **azymutu** i oznaczany będzie jako ***Θ***. Drugi oznaczony przez ***Φ***, określa pośrednio wysokość położenia źródła światła nad hipotetycznym horyzontem i nazywa się kątem **elewacji**. Właściwy układ geometryczny został zilustrowany na rysunku 5.

Wprowadzenie na scenę 3D źródła światła:

```
void MyInit(void)
{
       glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
// Definicja materiału z jakiego zrobiony jest czajnik

    GLfloat mat_ambient[]  = {1.0, 1.0, 1.0, 1.0};
    // współczynniki ka =[kar,kag,kab] dla światła otoczenia

    GLfloat mat_diffuse[]  = {1.0, 1.0, 1.0, 1.0};
    // współczynniki kd =[kdr,kdg,kdb] światła rozproszonego

    GLfloat mat_specular[] = {1.0, 1.0, 1.0, 1.0};
    // współczynniki ks =[ksr,ksg,ksb] dla światła odbitego

    GLfloat mat_shininess  = {20.0};
    // współczynnik n opisujący połysk powierzchni

/*************************************************************************************/
// Definicja źródła światła
    GLfloat light_position[] = {0.0, 10.0, 0.0, 1.0};
    GLfloat light_position1[] = {10.0, 0.0, 0.0, 1.0};
    // położenie źródła

    GLfloat light_ambient[] = {0.0, 1.0, 0.0, 1.0};
    // składowe intensywności świecenia źródła światła otoczenia
    // Ia = [Iar,Iag,Iab]

    GLfloat light_diffuse[] = {0.4, 1.0, 0.5, 1.0};
    // składowe intensywności świecenia źródła światła powodującego
    // odbicie dyfuzyjne Id = [Idr,Idg,Idb]

    GLfloat light_specular[]= {1.0, 1.0, 1.0, 1.0};
```

```
    // składowe intensywności świecenia źródła światła powodującego
    // odbicie kierunkowe Is = [Isr,Isg,Isb]

    GLfloat att_constant  = {0.5};
    // składowa stała ds dla modelu zmian oświetlenia w funkcji
    // odległości od źródła

    GLfloat att_linear    = {0.05};
    // składowa liniowa dl dla modelu zmian oświetlenia w funkcji
    // odległości od źródła

    GLfloat att_quadratic  = {0.001};
    // składowa kwadratowa dq dla modelu zmian oświetlenia w funkcji
    // odległości od źródła

    // położenie źródła
    GLfloat light_ambient1[] = {1.0, 0.0, 0.0, 1.0};
    // składowe intensywności świecenia źródła światła otoczenia
    // Ia = [Iar,Iag,Iab]

    GLfloat light_diffuse1[] = {0.5, 0.5, 0.5, 1.0};
    // składowe intensywności świecenia źródła światła powodującego
    // odbicie dyfuzyjne Id = [Idr,Idg,Idb]

    GLfloat light_specular1[]= {0.2, 0.2, 0.3, 1.0};
    // składowe intensywności świecenia źródła światła powodującego
    // odbicie kierunkowe Is = [Isr,Isg,Isb]

// Ustawienie parametrów materiału i źródła światła
/*****************************************************************************/
// Ustawienie patrametrów materiału

    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);

// Ustawienie parametrów źródła

    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);

       glLightfv(GL_LIGHT1, GL_AMBIENT, light_ambient1);
       glLightfv(GL_LIGHT1, GL_DIFFUSE, light_diffuse1);
       glLightfv(GL_LIGHT1, GL_SPECULAR, light_specular1);
       glLightfv(GL_LIGHT1, GL_POSITION, light_position1);

       glLightf(GL_LIGHT1, GL_CONSTANT_ATTENUATION, att_constant);
       glLightf(GL_LIGHT1, GL_LINEAR_ATTENUATION, att_linear);
       glLightf(GL_LIGHT1, GL_QUADRATIC_ATTENUATION, att_quadratic);

    glShadeModel(GL_SMOOTH); // właczenie łagodnego cieniowania
    glEnable(GL_LIGHTING);   // właczenie systemu oświetlenia sceny
    glEnable(GL_LIGHT0);     // włączenie źródła o numerze 0
    glEnable(GL_LIGHT1);     // włączenie źródła o numerze 1
    glEnable(GL_DEPTH_TEST); // włączenie mechanizmu z-bufora
}
```

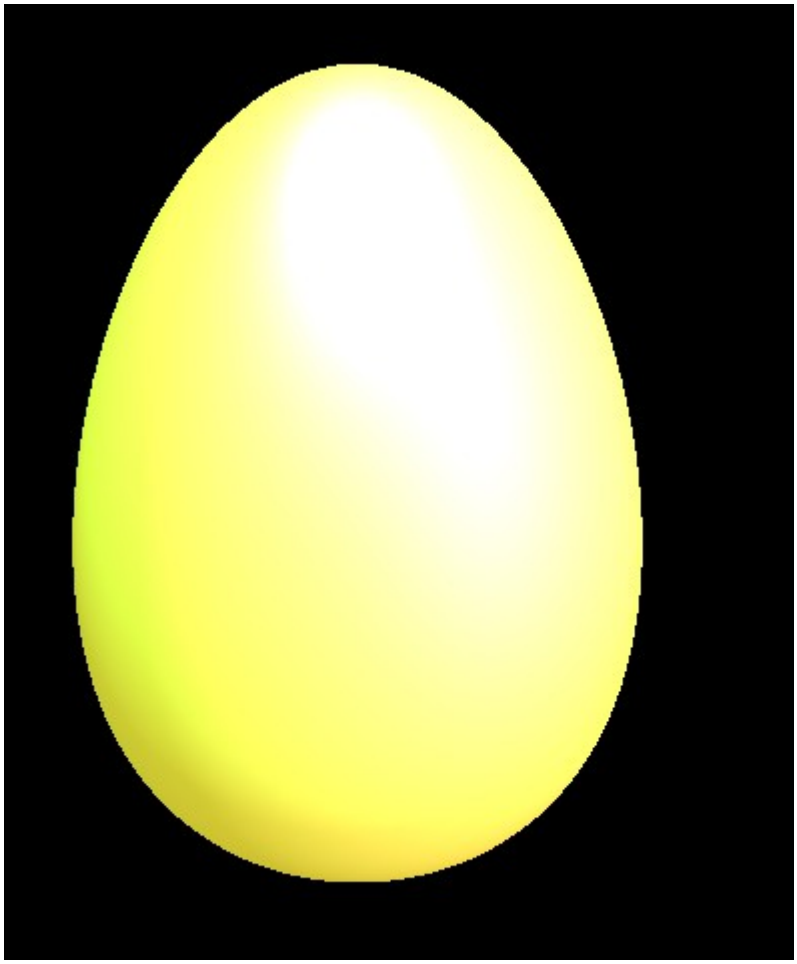
Obliczanie wektora normalnego, rysowanie jajka:

```
#define N 150
float wikt[N][N][3];
void jajo3()
{
	float u = 0, v=0;
	float tp[N][N][3];
	for(int i=0; i<N; ++i)
	{
		u = (float)i/(N-1);
		for(int j = 0; j<N; ++j)
		{
			v = (float)j/(N-1);
tp[i][j][0] = (-90*potg(u, 5)+225*potg(u, 4)-270*potg(u, 3)+180*potg(u, 2)-45*u)*cos(pi*v);
tp[i][j][1] = 160*potg(u, 4)-320*potg(u, 3)+160*potg(u, 2)-5.0;
tp[i][j][2] = (-90*potg(u, 5)+225*potg(u, 4)-270*potg(u, 3)+180*potg(u, 2)-45*u)*sin(pi*v);

float xu = (-450*potg(u,4)+900*potg(u,3) - 810*potg(u,2) + 360*u -45)*cos(3.14 * v);
float xv = 3.14 * (90*potg(u,5)-225*potg(u,4)+270*potg(u,3)-180*potg(u,2)+45*u)*sin(3.14*v);
float yu = 640*potg(u,3)-960*potg(u,2) + 320*u;
float yv = 0.0;
float zu = (-450*potg(u,4)+900*potg(u,3) - 810*potg(u,2) + 360*u -45)*sin(3.14 * v);
float zv = -3.14*(90*potg(u,5)-225*potg(u,4)+270*potg(u,3)-180*potg(u,2) +45*u)*cos(3.14*v);

float dlugosc;
wikt[i][j][0] = yu*zv-zu*yv;
wikt[i][j][1] = zu*xv-xu*zv;
wikt[i][j][2] = xu*yv-yu*xv;
dlugosc = sqrt(potg(wikt[i][j][0],2) + potg(wikt[i][j][1],2) + potg(wikt[i][j][2],2));
wikt[i][j][0] = wikt[i][j][0]/dlugosc;
wikt[i][j][1] =wikt[i][j][1]/ dlugosc;
wikt[i][j][2] = wikt[i][j][2]/dlugosc;
if(i >= N/2)
			{
				wikt[i][j][0] = wikt[i][j][0]* (-1);
				wikt[i][j][1] = wikt[i][j][1]* (-1);
				wikt[i][j][2] = wikt[i][j][2]* (-1);
			}
		}
	}
glBegin(GL_TRIANGLES);
for(int i=0; i<N-1; i++)
	{
		for(int j=0; j<N-1; j++)
		{
			glBegin(GL_TRIANGLES);
			glNormal3fv(wikt[i][j]);
			glVertex3fv(tp[i][j]);
			glNormal3fv(wikt[i+1][j]);
			glVertex3fv(tp[i+1][j]);
			glNormal3fv(wikt[i][j+1]);
			glVertex3fv(tp[i][j+1]);
			glEnd();
			glBegin(GL_TRIANGLES);
			glNormal3fv(wikt[i+1][j+1]);
			glVertex3fv(tp[i+1][j+1]);
			glNormal3fv(wikt[i+1][j]);
			glVertex3fv(tp[i+1][j]);
			glNormal3fv(wikt[i][j+1]);
			glVertex3fv(tp[i][j+1]);
			glEnd();
		}
	}
	glEnd();
}
```

Funkcja RenderScene() :

```
void RenderScene(void)
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glutReshapeFunc(ChangeSize);
	glLoadIdentity();
	gluLookAt(viewer[0],viewer[1],viewer[2], 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
	//Axes();
	if(status == 1)
	{
		theta += delta_x*pix2angle/50;
		theta2 += delta_y*pix2angle/50;
	}
	else if (statusp==1)
	{
		if(delta_x>=delta_y) zoom=zoom+0.01;
		else zoom=zoom-0.01;

		if (zoom >=-0.01 && zoom <= 4.0) glScalef(zoom, zoom, zoom );
		else zoom = 1.0;

	}

	viewer[0]=Er*cos(theta)*cos(theta2);
	viewer[1]=Er*sin(theta2);
	viewer[2]=Er*sin(theta)*cos(theta2);
	glColor3f(1.0f, 1.0f, 1.0f);
	jajo3();
	glFlush();
	glutSwapBuffers();


}
```

Wynik działania programu:



Wnioski, problemy, podsumowanie:

1. Największą trudnością było wyznaczanie wektora normalnego :

W ćwiczeniu 3 budowano model jajka na postawie równań parametrycznych powierzchni w postaci:

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

Wektor normalny do punktu leżącego na tak opisanej powierzchni można znaleźć posługując się wzorami:

$$N(u,v) = \left[\begin{vmatrix} y_u & z_u \\ y_v & z_v \end{vmatrix}, \begin{vmatrix} z_u & x_u \\ z_v & x_v \end{vmatrix}, \begin{vmatrix} x_u & y_u \\ x_v & y_v \end{vmatrix}\right] =$$
$$= \left[y_u \cdot z_v - z_u y_v \ , \quad z_u \cdot x_v - x_u z_v \ , \quad x_u \cdot y_v - y_u x_v\right] \neq 0$$

przy czym

$$x_u = \frac{\partial x(u,v)}{\partial u}, \quad x_v = \frac{\partial x(u,v)}{\partial v}$$
$$y_u = \frac{\partial y(u,v)}{\partial u}, \quad y_v = \frac{\partial y(u,v)}{\partial v}$$
$$z_u = \frac{\partial z(u,v)}{\partial u}, \quad z_v = \frac{\partial z(u,v)}{\partial v}$$

Po wykonaniu prostych operacji różniczkowania otrzymuje się wzory pozwalające na łatwe obliczenie wektora normalnego.

$$x_u = \frac{\partial x(u,v)}{\partial u} = \left(-450u^4 + 900u^3 - 810u^2 + 360u - 45\right)\cdot\cos(\pi v)$$
$$x_v = \frac{\partial x(u,v)}{\partial v} = \pi\cdot\left(90u^5 - 225u^4 + 270u^3 - 180u^2 + 45u\right)\cdot\sin(\pi v)$$
$$y_u = \frac{\partial y(u,v)}{\partial u} = 640u^3 - 960u^2 + 320u$$
$$y_v = \frac{\partial y(u,v)}{\partial v} = 0$$
$$z_u = \frac{\partial z(u,v)}{\partial u} = \left(-450u^4 + 900u^3 - 810u^2 + 360u - 45\right)\cdot\sin(\pi v)$$
$$z_v = \frac{\partial z(u,v)}{\partial v} = -\pi\cdot\left(90u^5 - 225u^4 + 270u^3 - 180u^2 + 45u\right)\cdot\cos(\pi v)$$

2. Niestety, nie udało się umożliwić sterowania położeniem źródeł przy pomocy myszy – za pomocą myszy manipulowane było jajko.