Grafika komputerowa - oświetlanie scen 3D

Kacper Małkowski 252724

prowadzący - dr. inż. Jan Nikodem

1 Wstęp

Celem ćwiczenia jest ilustracja możliwości oświetlania obiektów na scenach 3D z wykorzystaniem biblioteki OpenGL z rozszerzeniem GLUT. Poznaliśmy opis własności materiału, z którego jest wykonany oświetlany obiekt, jak na scenie zdefiniować źródło światła i jak dobrać jego parametry. Użyliśmy po raz pierwszy modelu Phonga, oraz poznaliśmy sposób na wyliczanie wektorów normalnych potrzebnych do użycia modelu.

2 Plan działania

Program ma wyświetlać model (jajko lub czajnik), który jest oświetlany przez 2 ruchome źródła światła. Obracanie obiektu odbywa się przy przyciśniętym środkowym przycisku myszy. Czerwone światło jest przesuwane przy wciśniętym lewym przycisku myszy, a niebieskie przy prawym. Model Phonga pozwala na oświetlenie modelów 3D używając do tego jedynie wektorów normalnych powierzchni odbijającej swiatło oprócz definicji materiału, oraz oświetlenia. W ćwiczeniu laboratoryjnym został podany kod pokazujący jak zdefiniować materiał i utworzyć źródło światła, co wykorzystałem w programie w funkcji myInit().

```
1  void MyInit(void)
2  {
3    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
4
5
6    GLfloat mat_ambient[] = { 1.0, 1.0, 1.0, 1.0 };//okreslenie parametrow materialu
7    GLfloat mat_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
8    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
9    GLfloat mat_shininess = { 20.0 };
10
11   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);//ustawienie parametrow materialu
12   glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
13   glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
14   glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);
15
16
17   GLfloat light_position[] = { 0.0, 0.0, 10.0, 1.0 };//pocztkoawa pozycja punktow
         swietlnych
18
19   GLfloat att_constant = { 1.0 };//okreslenia parametrow swiatla
20   GLfloat att_linear = { 0.05 };
21   GLfloat att_quadratic = { 0.001 };
22   GLfloat light_ambient[] = { 0.1, 0.1, 0.1, 1.0 };
23
24
25   GLfloat redlight_diffuse[] = { 1.0, 0.0, 0.0, 1.0 };//parametry swiatla czerwonego
26   GLfloat redlight_specular[] = { 1.0, 1.0, 1.0, 1.0 };
27   GLfloat bluelight_diffuse[] = { 0.0, 0.0, 1.0, 1.0 };//parametry swiatla
         niebieskiego
28   GLfloat bluelight_specular[] = { 1.0, 1.0, 1.0, 1.0 };
29
```

```
30    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);//ustawienie GL_LIGHT0 na czerwone
          zrodlo swiatla
31    glLightfv(GL_LIGHT0, GL_DIFFUSE, redlight_diffuse);
32    glLightfv(GL_LIGHT0, GL_SPECULAR, redlight_specular);
33    glLightfv(GL_LIGHT0, GL_POSITION, redLightPosition);
34    glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, att_constant);
35    glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, att_linear);
36    glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, att_quadratic);
37
38    glLightfv(GL_LIGHT1, GL_AMBIENT, light_ambient);//ustawienie GL_LIGHT0 na
          niebieskie zrodlo swiatla
39    glLightfv(GL_LIGHT1, GL_DIFFUSE, bluelight_diffuse);
40    glLightfv(GL_LIGHT1, GL_SPECULAR, bluelight_specular);
41    glLightfv(GL_LIGHT1, GL_POSITION, blueLightPosition);
42    glLightf(GL_LIGHT1, GL_CONSTANT_ATTENUATION, att_constant);
43    glLightf(GL_LIGHT1, GL_LINEAR_ATTENUATION, att_linear);
44    glLightf(GL_LIGHT1, GL_QUADRATIC_ATTENUATION, att_quadratic);
45
46
47    //uruchomienie oswietlenia
48    glShadeModel(GL_SMOOTH); // wlaczenie lagodnego cieniowania
49    glEnable(GL_LIGHTING);   // wlaczenie systemu o wietlenia sceny
50    glEnable(GL_LIGHT0);     // wlaczenie zrodla czerwonego
51    glEnable(GL_LIGHT1);  // wlaczenie zrodla niebieskiego
52    glEnable(GL_DEPTH_TEST); // wlaczenie mechanizmu z-bufora
53
54  }
```

Najpierw zdefiniowałem materiał za pomocą tablic z parametrami i funkcji definiującej materiał glMaterialfv(). Następnie zrobiłem to samo z parametrami źródła światła i ustawiłem je przy pomocy funkcji glLightfv(). W taki sposób stworzyłem dwa źródła światła GL_LIGHT0 i GL_LIGHT1. Na końcu użyłem funkcji glShadeModel(), która wygładza rysowanie światła na modelu, oraz glEnable() do aktywacji modelu oświetlania oraz do obu źródeł światła.

3 Program

Jako szkielet programu użyłem kodu z poprzedniego laboratorium. Dodałem wyżej opisaną funkcję do stworzenia światła i nadania materiału. Następne zmiany opiszę po kolei.

3.1 Zmienne globalne

```
1  //zmienne ogolne
2  typedef float point3[3];
3  typedef float point9[9];
4  int testedObject = 2; //rysowany obiekt
5  static int x_pos_old = 0;       // pozycje kursora myszy
6  static int delta_x = 0;
7  static int y_pos_old = 0;
8  static int delta_y = 0;
9  static GLfloat pix2angle;     // przelicznik pikseli na stopnie
10
11 //zmienne obiektu
12 int n = 100;//ilosc punktow jajka
13 float scale = 3.0;//wielkosc obiektu
14 point3** points;//siatka punktow
15 point3** vectors;//wektory punktow powierzchni jajka
16
17 //zmienne obrotu obiektu
18 static GLint statusMiddle = 0;//stan srdkowego klawisza
19 float rViewer = 10;//R wokol punktu obserwowanego
20 static GLfloat viewer[] = { 0.0, 0.0, 10.0 };//pozycja punktu widzenia
21 static GLfloat azymuth = 0;    //katy obserwacji punktu obserwowanego
22 static GLfloat elevation = 0;
```

```
23
24 //punkty swietlne
25 static GLint statusLeft = 0;         // stan klawisza lewego
26 static GLint statusRight = 0;      // stan klawisza prawego
27 float rLight = 10;              //odleglosc zrodla swiatla
28 static GLfloat lightRedAngles[] = { 0.0, 0.0 };//dane swiatla czerwonego
29 GLfloat redLightPosition[] = { 10.0, 10.0, 10.0, 1.0 };
30 static GLfloat lightBlueAngles[] = { 0.0, 0.0 };//dane swiatla niebieskiego
31 GLfloat blueLightPosition[] = { -10.0, 10.0, 10.0, 1.0 };
```

Najpierw zmieniłem zmienne globalne używane w poprzednim zadaniu. Usunąłem niepotrzebne zmienne, oraz dodałem zmienne do obsługi środkowego przycisku myszki, oraz zmienne przechowujące pozycje źródeł światła. Każda zmienna jest opisana w kodzie.

3.2 main()

```
1  void main(void)
2  {
3    srand(time(NULL));
4    points = new  point3 * [n + 1];//tablica punktow
5    vectors = new point3 * [n + 1];//tablice
6    for (int i = 0; i <= n; i++) {
7      points[i] = new point3[n + 1];
8      vectors[i] = new point3[n + 1];
9    }
10
11   cout << "Obsluga programu:\nc - czajnik\nj - jajko\nlewy przycisk myszy - aktywuje
         ruch czerwonego swiatla\nprawy przycisk myszy - altywuje ruch niebieskiego
         swiatla\nsrodkowy przycisk myszy - ruch obiektu" << endl;
12   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
13   glutInitWindowSize(1000, 1000);
14   glutCreateWindow("Rzutowanie perspektywiczne");
15   glutDisplayFunc(RenderScene);
16   glutReshapeFunc(ChangeSize);
17   MyInit();
18   glutMouseFunc(Mouse);//"lapanie" akcji na przyciskach myszy
19   glutMotionFunc(Motion);//"lapanie" ruchu myszki
20   glutKeyboardFunc(keys);//"lapanie" akcji na klawiaturze
21   glEnable(GL_DEPTH_TEST);
22   glutMainLoop();
23 }
```

Najpierw alokowana jest tablica punktów jajka, oraz jego wektorów normalnych. Następnie wyświetlana w terminalu jest instrukcja obsługi programu. Na końcu jest blok funkcji z biblioteki GLUT obsługujące funkcje rysowania, myszy, klawiatury i pozostałych implementowanych na zajęciach funkcjonalności. Wszystkie te funkcje były wcześniej użyte oprócz funkcji glEnable(), która uruchamia oświetlenie.

3.3 mouse() i keys()

```
1  void Mouse(int btn, int state, int x, int y)
2  {
3    if (btn == GLUT_RIGHT_BUTTON && state == GLUT_DOWN) {//sprawdzenie czy
         przycisniety zostal prawy klawisz
4      x_pos_old = x;
5      y_pos_old = y;
6      statusRight = 1;         //ustawienie flagi przycisku
7    }
8    else {
9      statusRight = 0;     //ustawienie flagi przycisku
10   }
11
12   if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN) {//sprawdzenie czy przycisniety
         zostal lewy klawisz
```

```
    x_pos_old = x;
    y_pos_old = y;
    statusLeft = 1;     //ustawienie flagi przycisku
  }
  else {
    statusLeft = 0;    //ustawienie flagi przycisku
  }

  if (btn == GLUT_MIDDLE_BUTTON && state == GLUT_DOWN) {//sprawdzenie czy
      przycisniety zostal lewy klawisz
    x_pos_old = x;
    y_pos_old = y;
    statusMiddle = 1;     //ustawienie flagi przycisku
  }
  else {
    statusMiddle = 0;    //ustawienie flagi przycisku
  }

  RenderScene();
}
```

Zmianom uległa także z funkcje obsługujące mysz i klawiaturę - została dodana funkcjonalność wykrywająca środkowy przycisk myszy, oraz dodana obsługa klawiszy 'c" i "j" odpowiadająca za zmianę wyświetlanego obiektu c - czajnika, j - jajka.

```
void keys(unsigned char key, int x, int y)
{
  if (key == 'c') {
    testedObject = 1;
  }
  if (key == 'j') {
    testedObject = 2;
  }

  RenderScene();
}
```

3.4 egg()

```
void egg() {
  float u = 0, v = 0;
  float udiff = 1.0 / n, vdiff = 1.0 / n; //n - liczba punktow na powierzchni jajka
  glTranslated(0, (-(160 * pow(0.5, 4) - 320 * pow(0.5, 3) + 160 * pow(0.5, 2)) / 2)
      * (scale + 7) / 10, 0);//obnizenie  rodka  figury do centrum ukladu
      wspolrzednych

  for (int i = 0; i <= n; i++) {//punkty powierzchni
    v = 0;//obliczenie poteg w celu ulatwienia kodu
    float u2 = pow(u, 2);
    float u3 = pow(u, 3);
    float u4 = pow(u, 4);
    float u5 = pow(u, 5);

    for (int ii = 0; ii <= n; ii++) {//obliczenie wspolrzednych punktow
      points[i][ii][0] = ((-90 * u5 + 225 * u4 - 270 * u3 + 180 * u2 - 45 * u) *
          cos(M_PI * v)) * (scale + 7) / 10;
      points[i][ii][1] = (160 * u4 - 320 * u3 + 160 * u2) * (scale + 7) / 10;
      points[i][ii][2] = ((-90 * u5 + 225 * u4 - 270 * u3 + 180 * u2 - 45 * u) *
          sin(M_PI * v)) * (scale + 7) / 10;
      v = v + vdiff;
    }
    u = u + udiff;
  }

  u = 0;
```

```
23    for (int i = 0; i <= n; i++) {//wektory normalne
24      v = 0;//obliczenie poteg w celu ulatwienia kodu
25      float u2 = pow(u, 2);
26      float u3 = pow(u, 3);
27      float u4 = pow(u, 4);
28      float u5 = pow(u, 5);
29
30      for (int ii = 0; ii <= n; ii++) {
31        point9 vector;
32        vector[0] = (-450 * u4 + 900 * u3 - 810 * u2 + 360 * u - 45) * cos(M_PI * v);
33        vector[1] = M_PI * (90 * u5 - 225 * u4 + 270 * u3 - 180 * u2 + 45) * sin(M_PI
              * v);
34        vector[2] = 640 * u3 - 960 * u2 + 320 * u;
35        vector[3] = 0;
36        vector[4] = (-450 * u4 + 900 * u3 - 810 * u2 + 360 * u - 45) * sin(M_PI * v);
37        vector[5] = -1 * M_PI * (90 * u5 - 225 * u4 + 270 * u3 - 180 * u2 + 45) *
              cos(M_PI * v);
38
39        vector[6] = vector[2] * vector[5] - vector[4] * vector[3];//wektory
40        vector[7] = vector[4] * vector[1] - vector[0] * vector[5];
41        vector[8] = vector[0] * vector[3] - vector[2] * vector[1];
42        float vectorSize = sqrt(pow(vector[6], 2) + pow(vector[7], 2) + pow(vector[8],
              2));//normalizacja wektora
43        if (vectorSize == 0) {
44          vectorSize = 1;
45        }
46
47        vectors[i][ii][0] = vector[6] / vectorSize;
48        vectors[i][ii][1] = vector[7] / vectorSize;
49        vectors[i][ii][2] = vector[8] / vectorSize;
50
51        v = v + vdiff;
52      }
53      u = u + udiff;
54    }
55
56    for (int i = 0; i < n; i++) {//rysowanie
57      for (int ii = 0; ii < n; ii++) {
58        glBegin(GL_TRIANGLES);//rysowanie pierwszego trojkata
59        glNormal3fv(vectors[i][ii]);
60        glVertex3f(points[i][ii][0], points[i][ii][1], points[i][ii][2]);
61        glNormal3fv(vectors[i + 1][ii]);
62        glVertex3f(points[i + 1][ii][0], points[i + 1][ii][1], points[i + 1][ii][2]);
63        glNormal3fv(vectors[i + 1][ii + 1]);
64        glVertex3f(points[i + 1][ii + 1][0], points[i + 1][ii + 1][1], points[i +
              1][ii + 1][2]);
65        glEnd();
66
67        glBegin(GL_TRIANGLES);//rysowanie drugiego trojkata
68        glNormal3fv(vectors[i][ii]);
69        glVertex3f(points[i][ii][0], points[i][ii][1], points[i][ii][2]);
70        glNormal3fv(vectors[i][ii + 1]);
71        glVertex3f(points[i][ii + 1][0], points[i][ii + 1][1], points[i][ii + 1][2]);
72        glNormal3fv(vectors[i + 1][ii + 1]);
73        glVertex3f(points[i + 1][ii + 1][0], points[i + 1][ii + 1][1], points[i +
              1][ii + 1][2]);
74        glEnd();
75      }
76    }
77  }
```

Największą zmianą w kodzie generowania modelu jajka jest dodanie pętli liczącej wektory normalne dla kolejnych punktów jajka (linijki 23-54). Wektory normalne liczone są ze wzorów z instrukcji.

3.5 zadanie()

```
1  void zadanie() {
2    if (statusMiddle == 1) {//obracanie obiektu
3      elevation += 0.01 * delta_y * pix2angle;
4      azymuth += 0.01 * delta_x * pix2angle;
5    }
6
7    viewer[0] = rViewer * cos(azymuth) * cos(elevation);//obliczenie pozycji punktu
         widzenia
8    viewer[1] = rViewer * sin(elevation);
9    viewer[2] = rViewer * sin(azymuth) * cos(elevation);
10
11   if (statusLeft == 1) {//przesuwanie czerwonego swiatla
12     lightRedAngles[0] += delta_x * pix2angle / 100;
13     lightRedAngles[1] += delta_y * pix2angle / 100;
14   }
15
16   if (statusRight == 1) {//przesuwanie niebieskieg swiatla
17     lightBlueAngles[0] += delta_x * pix2angle / 100;
18     lightBlueAngles[1] += delta_y * pix2angle / 100;
19   }
20
21   redLightPosition[0] = rLight * cos(lightRedAngles[0]) *
         cos(lightRedAngles[1]);//obliczenie pozycji czerwonego swiatla
22   redLightPosition[1] = rLight * sin(lightRedAngles[1]);
23   redLightPosition[2] = rLight * sin(lightRedAngles[0]) * cos(lightRedAngles[1]);
24
25   blueLightPosition[0] = rLight * cos(lightBlueAngles[0]) *
         cos(lightBlueAngles[1]);//obliczenie pozycji niebieskiego swiatla
26   blueLightPosition[1] = rLight * sin(lightBlueAngles[1]);
27   blueLightPosition[2] = rLight * sin(lightBlueAngles[0]) * cos(lightBlueAngles[1]);
28
29
30
31   glLightfv(GL_LIGHT0, GL_POSITION, redLightPosition);//aktualizacja pozycji zrodla
         swiatla
32   glLightfv(GL_LIGHT1, GL_POSITION, blueLightPosition);
33
34   switch (testedObject) {//wybranie obiektu do wyswietlania
35   case 1:
36     glutSolidTeapot(scale);
37     break;
38   case 2:
39     egg();
40     break;
41   }
42 }
```

Ostanie zmiany wystąpiły w funkcji zadanie, która wywoływana jest w RenderScene(). Zostały w niej dodane obrót kamery (linijki 2-9), oraz obracanie źródeł światła (11-32). Zmiana położenia źródeł światła odbywa się przy pomocy wzorów jak na obrót kamery wokół obiektu, a pozycja źródła jest aktualizowana przy pomocy funkcji glLightfv(), w której argumentem jest tablica z nową pozycją źródła.

4 Wnioski

System oświetlenia w bibliotece GLUT jest łatwy w implementacji. Model Phonga pozwala na skuteczne oświetlenie sceny 3D. Efekt programu jest satysfakcjonujący.