Grafika komputerowa i komunikacja człowiek-komputer

Sprawozdanie z laboratorium

Data	Tytuł zajęć	Uczestnicy
20.11.2017 8:00	OpenGL - oświetlanie scen 3D	Iwo Bujkiewicz (226203)

Zadania

Na zajęciach należało napisać, na podstawie instrukcji laboratoryjnej, programy realizujące trójwymiarowo, w rzucie perspektywicznym:

1. wyświetlanie wypełnionych trójkątów powierzchni oświetlonego czajnika Newella,
2. wyświetlanie oświetlonego modelu jajka,
3. wyświetlanie sceny z trzech różnych pozycji kamery,
4. używanie poprawnie wygenerowanych wektorów normalnych do cieniowania oświetlonych obiektów.

Kolejne etapy realizacji

Zadania 1, 2, 3, 4

Wszystkie zadania zrealizowano w ramach jednego programu.

Do rysowania jajka użyto zmodyfikowanego kodu z wcześniejszych laboratoriów. Istotne było dodanie macierzy wektorów normalnych oraz obsługi materiałów (zamiast bezpośredniego użycia kolorów do rysowania).

```
void vert(point3f * vertex) {
    memcpy(mat_ambient, get_vertex_color(vertex), sizeof(point3f));
    memcpy(mat_diffuse, get_vertex_color(vertex), sizeof(point3f));
    memcpy(mat_specular, get_vertex_color(vertex), sizeof(point3f));
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);

    point3f * normal = get_vertex_normal(vertex);
    glNormal3f((*normal)[0], (*normal)[1], (*normal)[2]);
    glVertex3f((*vertex)[0], (*vertex)[1], (*vertex)[2]);
}
```

```
void edge(point3f * vertex, point3f * next_vertex) {
    vert(vertex);
    vert(next_vertex);
}
```

```
void face(point3f * v1, point3f * v2, point3f * v3) {
    vert(v1);
    vert(v2);
    vert(v3);
}
```

Czajnik Newella rysowany był jednolitym, białym materiałem.

```
    if (!viewmode) {
        GLfloat mat_ambient[] = {1.0f, 1.0f, 1.0f, 1.0f};
        GLfloat mat_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
        GLfloat mat_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
        GLfloat mat_shininess = 20.0f;

        glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
        glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
        glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);
        glutSolidTeapot(3.0);
    }
```

Ważnym elementem tego programu była zmodyfikowana funkcja `init_render()`, w której tworzone było dynamiczne źródło światła, a także włączane było wygładzone cieniowanie obiektów na scenie.

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

    GLfloat light0_ambient[] = {0.1f, 0.1f, 0.1f, 1.0f};
    GLfloat light0_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
    GLfloat light0_specular[] = {1.0f, 1.0f, 1.0f, 1.0f};
    GLfloat light0_position[] = {0.0f, 0.0f, 10.0f, 1.0f};

    glLightfv(GL_LIGHT0, GL_AMBIENT, light0_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light0_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light0_specular);
    glLightfv(GL_LIGHT0, GL_POSITION, light0_position);

    glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 1.0f);
    glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, 0.05f);
    glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.001f);

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

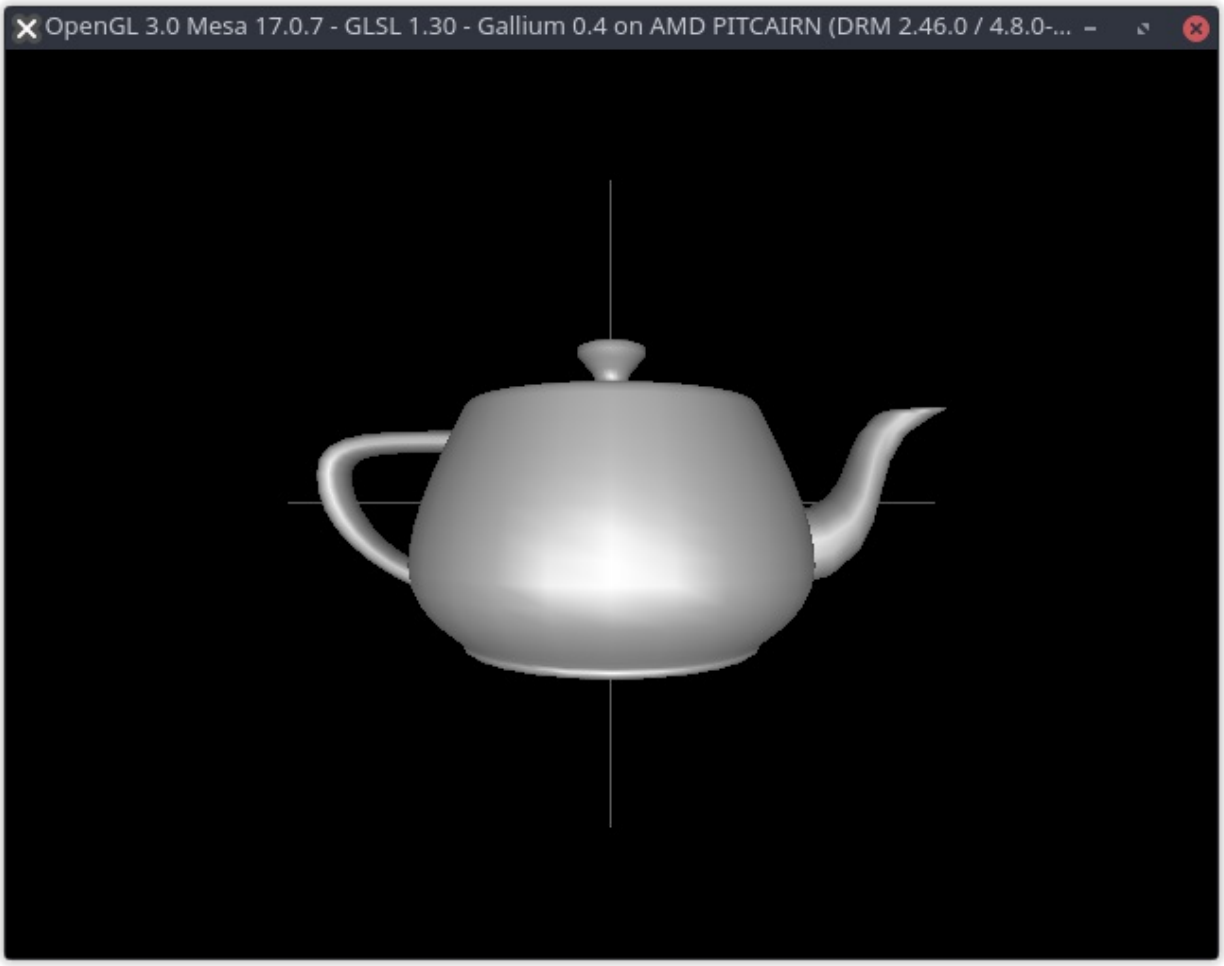
Zmiany punktu widzenia realizowała funkcja `key_pressed()`, która po naciśnięciu odpowiednich klawiszy nadpisywała współrzędne pozycji kamery, z których korzystała funkcja `render_scene()`.

```
void key_pressed(unsigned char key, int x, int y) {
    if (key == 'p')
        viewmode ^= 1<<0;
    else if (key == 'w')
        viewmode ^= 1<<1;
    else if (key == 's')
        viewmode ^= 1<<2;
    else if (key == '1') {
        GLdouble new_camera[] = {0.0f, 0.0f, 10.0f};
        memcpy(camera, new_camera, 3 * sizeof(GLdouble));
    }
    else if (key == '2') {
        GLdouble new_camera[] = {15.0f, 15.0f, 15.0f};
        memcpy(camera, new_camera, 3 * sizeof(GLdouble));
    }
    else if (key == '3') {
        GLdouble new_camera[] = {0.0f, -15.0f, -15.0f};
        memcpy(camera, new_camera, 3 * sizeof(GLdouble));
    }

    glutPostRedisplay();
}
```

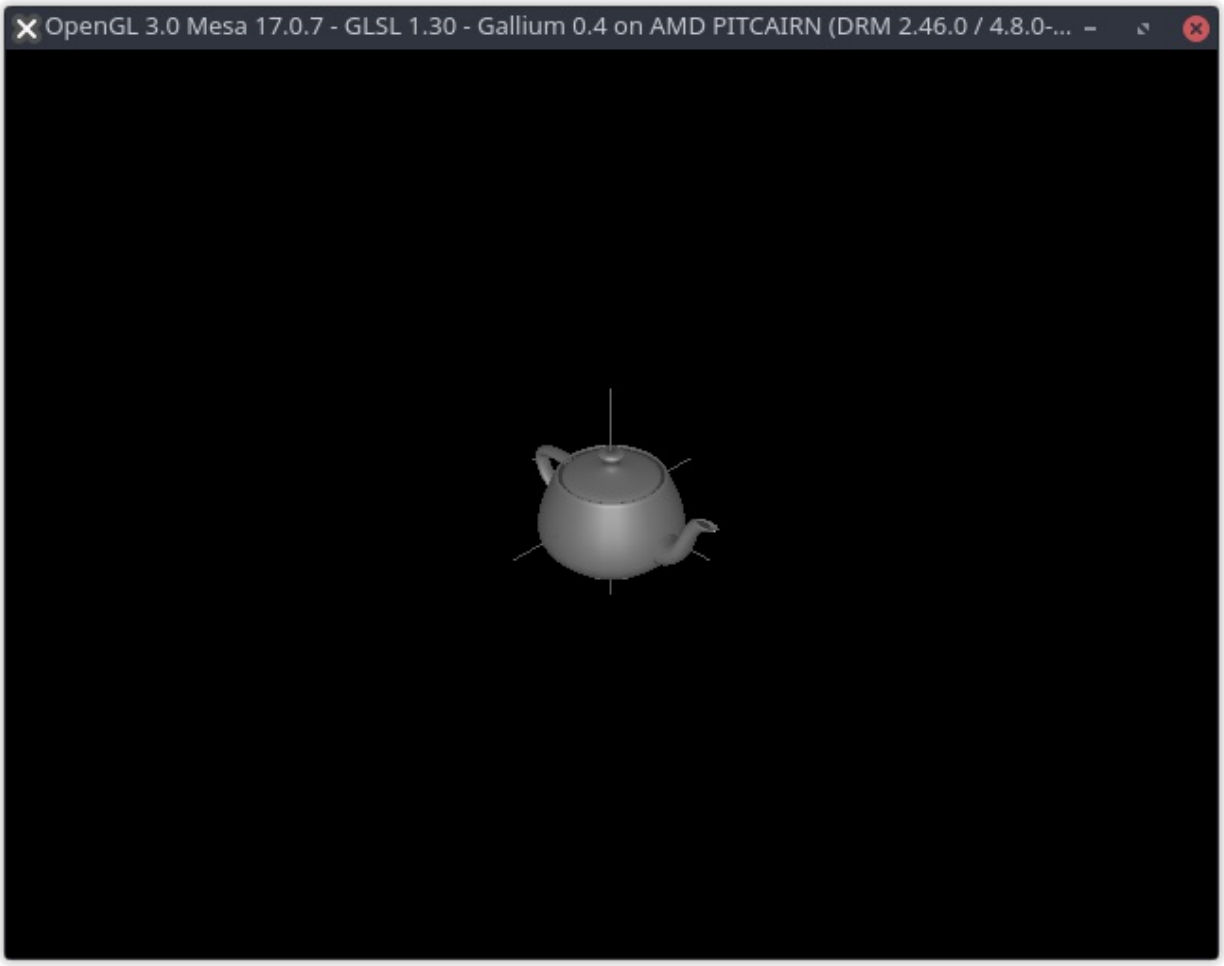
Aby oświetlone jajko było odpowiednio cieniowane, należało wygenerować dla jego wierzchołków wektory normalne. W tym celu zmodyfikowana została funkcja `create_vertex()`. Wektory generowane były zgodnie ze wzorami przedstawionymi w instrukcji laboratoryjnej, a ich normalizacja realizowana była automatycznie dzięki wywołaniu `glEnable(GL_NORMALIZE)` w `init_render()`.

```
void create_vertex(float u, float v, point3f * result, point3f * normal) {
    (*result)[0] = (-  90.0f * powf(u, 5.0f)
            + 225.0f * powf(u, 4.0f)
            - 270.0f * powf(u, 3.0f)
            + 180.0f * powf(u, 2.0f)
            -  45.0f * u) * cosf (M_PI * v);

    (*result)[1] =    160.0f * powf(u, 4.0f)
            - 320.0f * powf(u, 3.0f)
            + 160.0f * powf(u, 2.0f)
            - 5.0f;

    (*result)[2] = (-  90.0f * powf(u, 5.0f)
            + 225.0f * powf(u, 4.0f)
            - 270.0f * powf(u, 3.0f)
            + 180.0f * powf(u, 2.0f)
            -  45.0f * u) * sinf(M_PI * v);

    float xu = (- 450.0f * powf(u, 4.0f)
            + 900.0f * powf(u, 3.0f)
            - 810.0f * powf(u, 2.0f)
            + 360.0f * u
            -  45.0f) * cosf(M_PI * v);

    float xv = (   90.0f * powf(u, 5.0f)
            - 225.0f * powf(u, 4.0f)
            + 270.0f * powf(u, 3.0f)
            - 180.0f * powf(u, 2.0f)
            +  45.0f * u) * M_PI * sinf(M_PI * v);

    float yu =   640.0f * powf(u, 3.0f)
           - 960.0f * powf(u, 2.0f)
           + 320.0f * u;

    float yv = 0.0f;

    float zu = (- 450.0f * powf(u, 4.0f)
            + 900.0f * powf(u, 3.0f)
            - 810.0f * powf(u, 2.0f)
            + 360.0f * u
            -  45.0f) * sinf(M_PI * v);

    float zv = (   90.0f * powf(u, 5.0f)
            - 225.0f * powf(u, 4.0f)
            + 270.0f * powf(u, 3.0f)
            - 180.0f * powf(u, 2.0f)
            +  45.0f * u) * (- M_PI) * cosf(M_PI * v);

    (*normal)[0] = yu * zv - zu * yv;
    (*normal)[1] = zu * xv - xu * zv;
    (*normal)[2] = xu * yv - yu * xv;
    if (u > 0.5f) {
        (*normal)[0] *= -1.0f;
        (*normal)[1] *= -1.0f;
        (*normal)[2] *= -1.0f;
    }
}
```

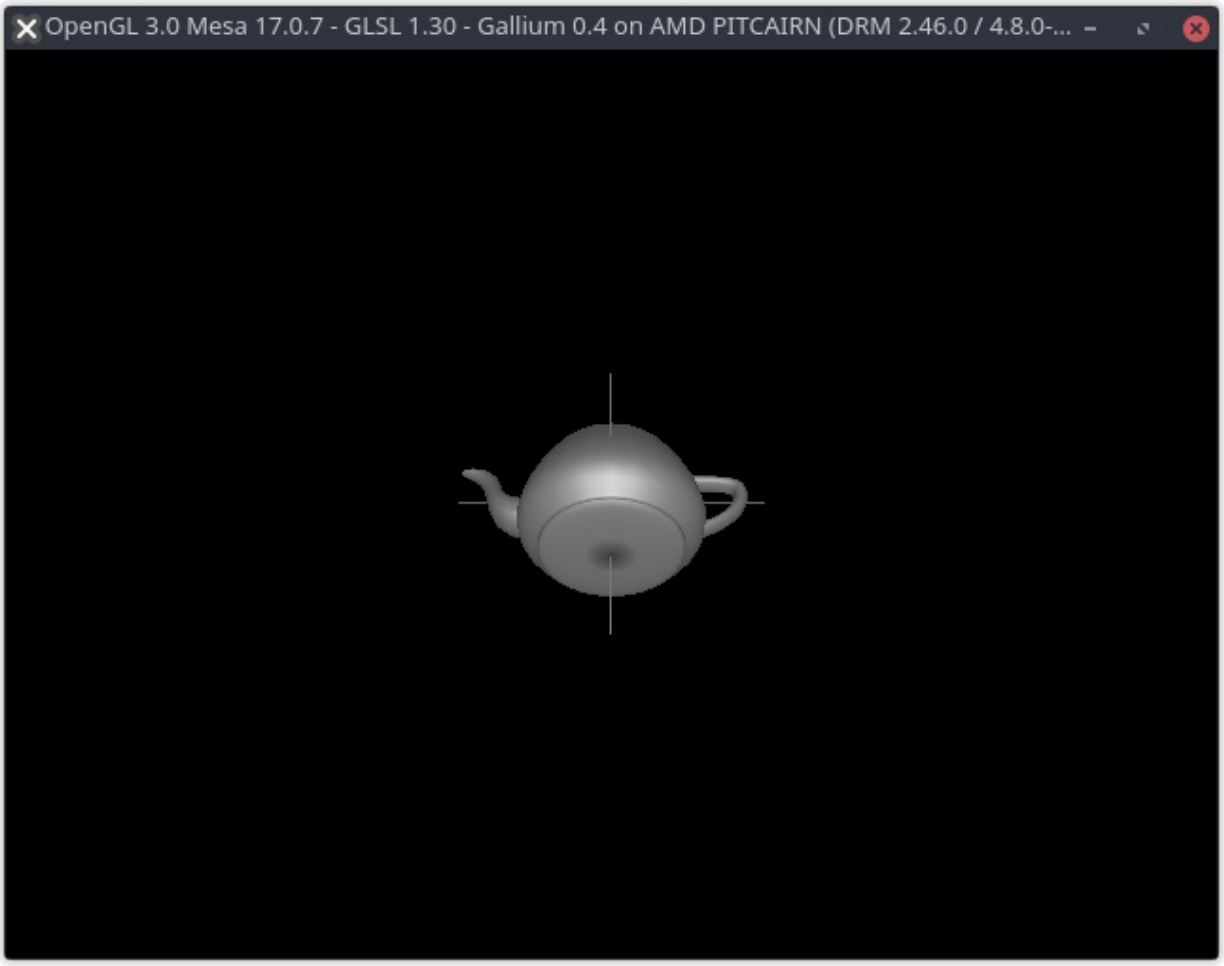
Rezultat działania programu prezentują poniższe zrzuty ekranu.



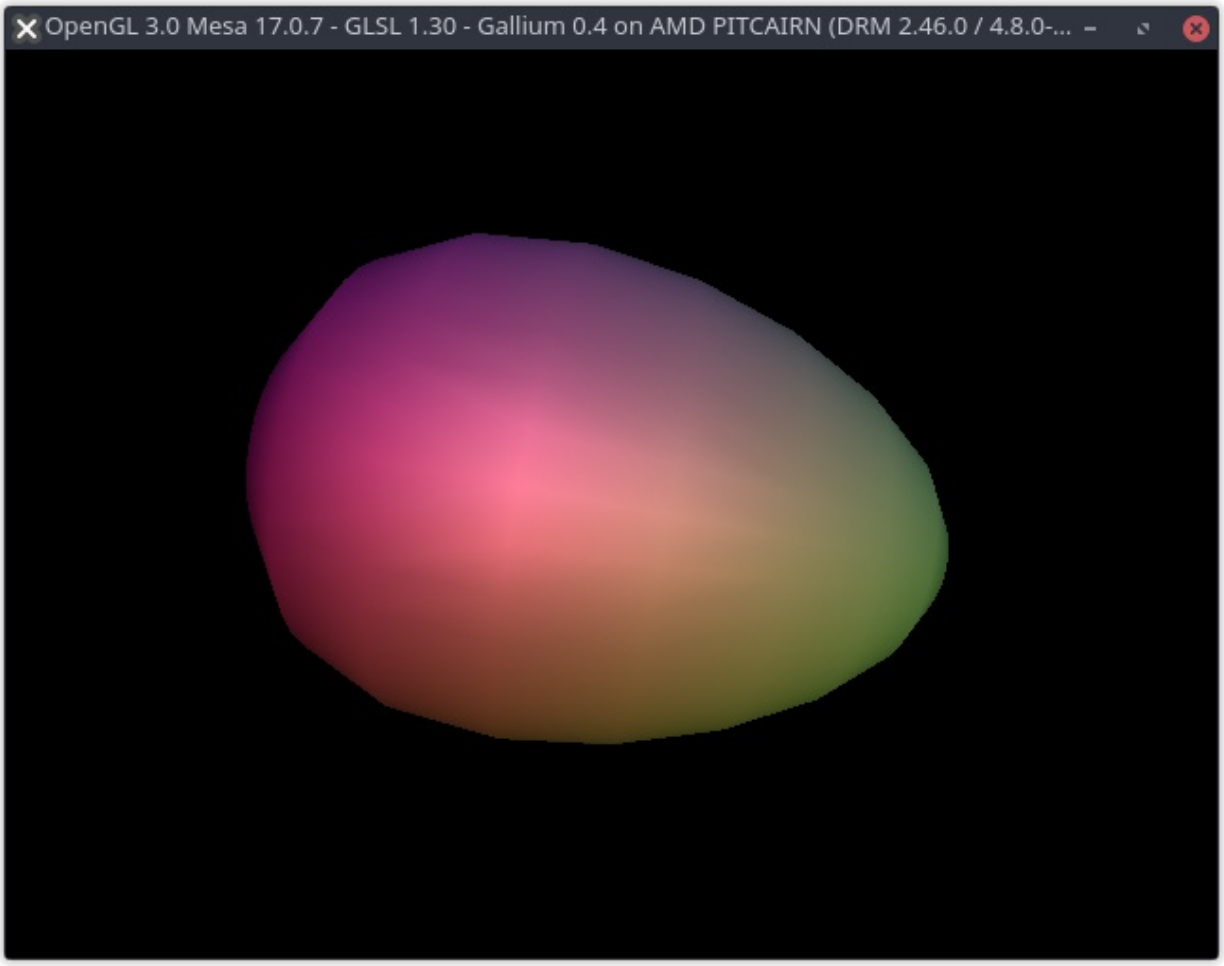
Oświetlony czajnik Newella



Widok z perspektywy drugiej kamery



Widok z perspektywy trzeciej kamery



Oświetlony model jajka

Kod źródłowy

Kompletny kod opisanych tu programów został załączony do sprawozdania w osobnych plikach.