Grafika komputerowa i komunikacja człowiek-komputer

Sprawozdanie z laboratorium

Data	Tytuł zajęć	Uczestnicy
04.12.2017 8:00	OpenGL - teksturowanie powierzchni obiektów	Iwo Bujkiewicz (226203)

Zadania

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

1. wyświetlanie oświetlonego, wypełnionego trójkąta z możliwością jego obracania,
2. nakładanie tekstury na wypełniony trójkąt,
3. wyświetlanie oświetlonego, oteksturowanego wielościanu.

Kolejne etapy realizacji

Zadanie 1

Pierwsze zadanie polegało na trywialnej modyfikacji programu z poprzedniego laboratorium w taki sposób, aby na scenie rysowany był pojedynczy, biały, oświetlony trójkąt.

```
void draw_triangle() {
    flat_color(1.0f, 1.0f, 1.0f, 1.0f);
    glBegin(GL_TRIANGLES);
    glVertex3f(-4.5f, -3.0f, 1.0f);
    glVertex3f(0.0f, 4.6f, -1.0f);
    glVertex3f(4.5f, -3.0f, 1.0f);
    glEnd();
}
```

Rezultat prezentuje poniższy zrzut ekranu.

Wypełniony trójkąt

Zadanie 2

W ramach drugiego zadania należalo na przygotowany wcześniej trójkąt nałożyć teksturę załadowaną z pliku graficznego. Do odczytu pliku użyto kodu z instrukcji laboratoryjnej.

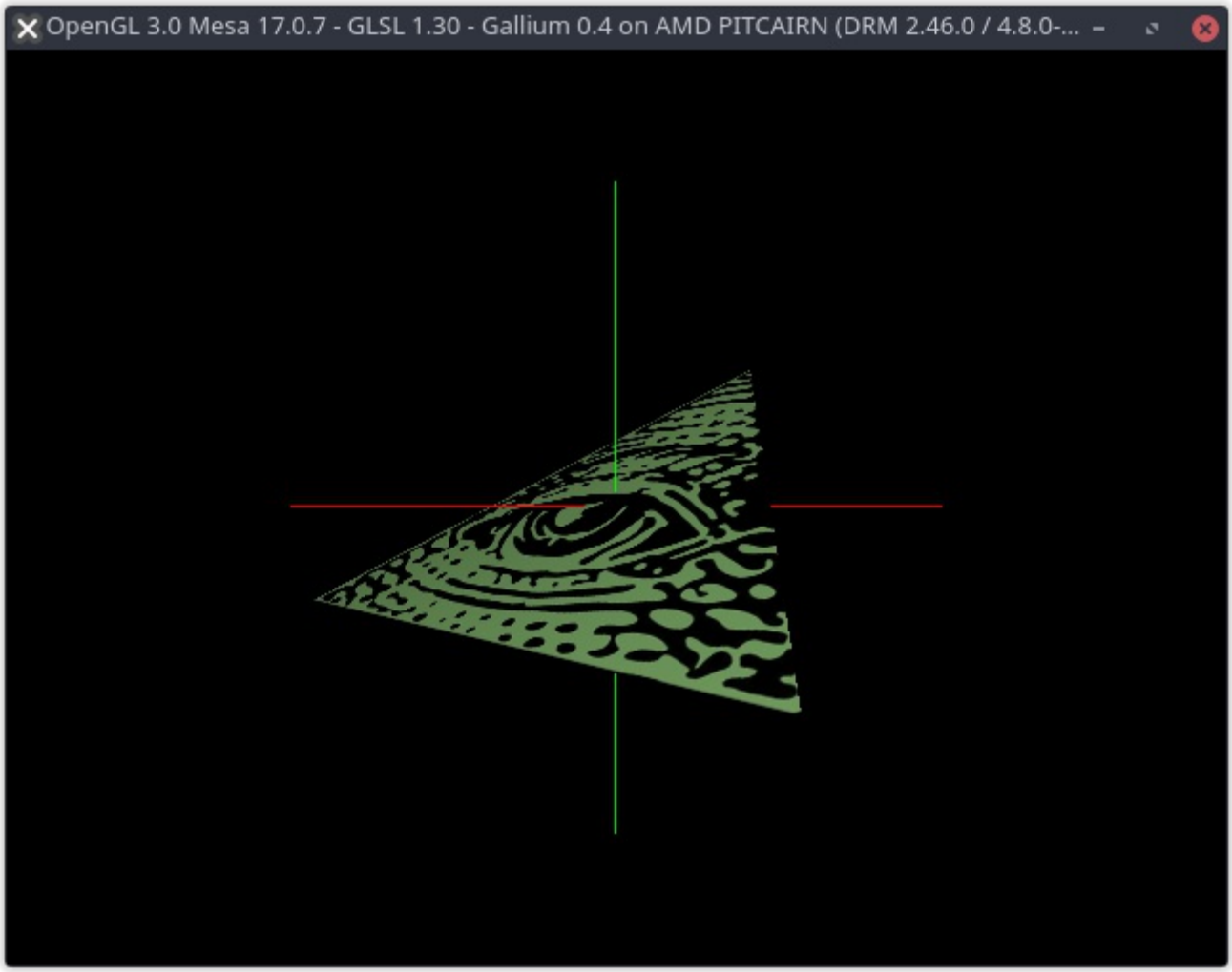
Tekstura ładowana była na początku działania programu, wewnątrz funkcji `init_render()`. Oprócz tego warto wspomnieć, że wywołaniem `glEnable(GL_CULL_FACE)` włączone zostało obcinanie tylnych powierzchni ścian obiektów, dzięki czemu każda ściana była od tego momentu rysowana tylko raz.

```
int 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);
    glEnable(GL_CULL_FACE);

    TGAimage* texture_ptr = read_tga("sample.tga");
    if (texture_ptr != NULL) {
        TGAimage texture = *texture_ptr;
        glTexImage2D(GL_TEXTURE_2D,
                     0,
                     texture.image_components,
                     texture.header.image_width,
                     texture.header.image_height,
                     0, texture.image_format,
                     GL_UNSIGNED_BYTE,
                     texture.bytes);
        free(texture.bytes);

        glEnable(GL_TEXTURE_2D);
        glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);

        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    }
    else {
        fprintf(stderr, "Could not read texture data\n");
        return 1;
    }
    return 0;
}
```

Do rysowanego na scenie trójkąta trzeba było dodać współrzędne mapowania tekstury, aby tekstura była poprawnie 'rozpięta' na jego powierzchni.

```
void draw_triangle() {
    flat_color(1.0f, 1.0f, 1.0f, 1.0f);
    glBegin(GL_TRIANGLES);
    glTexCoord2f(0.0f, 1.0f);
    glVertex3f(-4.5f, -3.0f, 1.0f);
    glTexCoord2f(1.0f, 1.0f);
    glVertex3f(4.5f, -3.0f, 1.0f);
    glTexCoord2f(0.5f, 0.0f);
    glVertex3f(0.0f, 4.6f, -1.0f);
    glEnd();
}
```

Rezultat działania programu prezentuje poniższy zrzut ekranu.



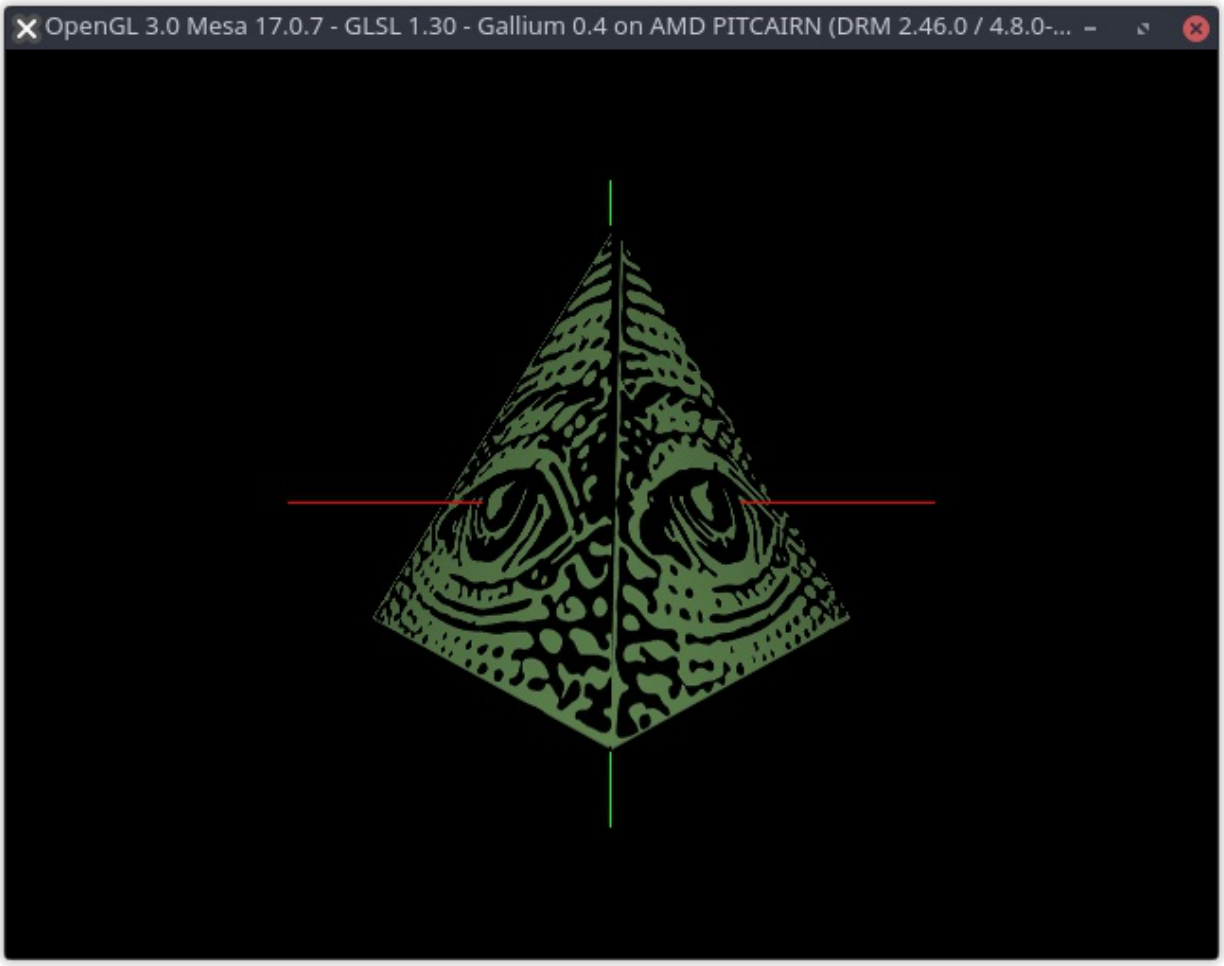
Oteksturowany trójkąt

Zadanie 3

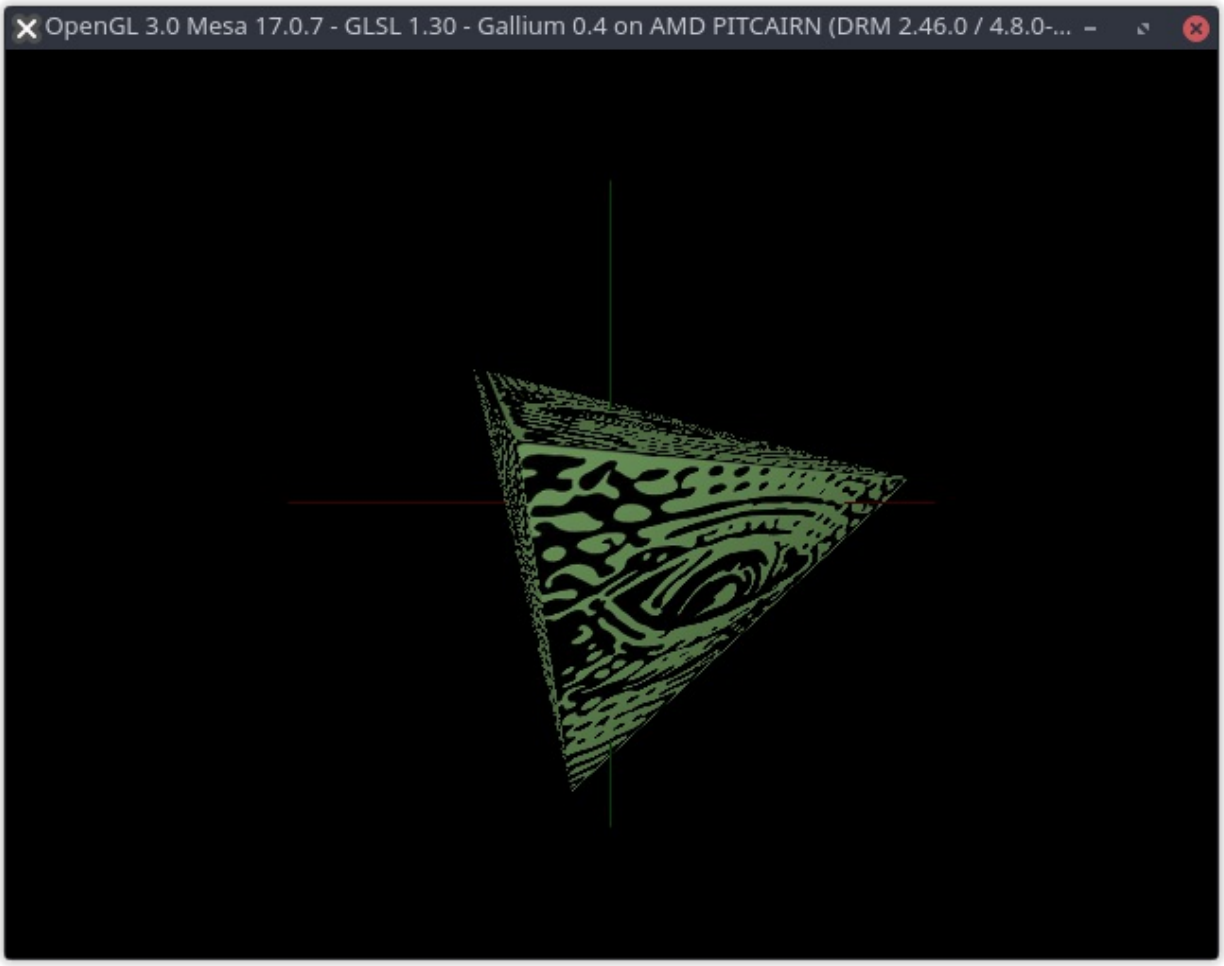
Trzecim zadaniem było oteksturowanie wielościanu. Wybrano do tego celu ostrosłup trójkątny, zbliżony wymiarami do czworościanu foremnego. Oprócz współrzędnych mapowania tekstury, dodano również wektory normalne odpowiednie dla poszczególnych ścian bryły, aby były one poprawnie cieniowane.

```
void draw_tetrahedron() {
    flat_color(1.0f, 1.0f, 1.0f, 1.0f);

    glBegin(GL_TRIANGLES);

    glTexCoord2f(0.0f, 1.0f);
    glNormal3f(0.0f, -1.0f, 0.0f);
    glVertex3f(-4.5f, -2.1666f, -2.1666f);
    glTexCoord2f(1.0f, 1.0f);
    glNormal3f(0.0f, -1.0f, 0.0f);
    glVertex3f(4.5f, -2.1666f, -2.1666f);
    glTexCoord2f(0.5f, 0.0f);
    glNormal3f(0.0f, -1.0f, 0.0f);
    glVertex3f(0.0f, -2.1666f, 4.3334f);

    glTexCoord2f(0.0f, 1.0f);
    glNormal3f(-4.5f, 2.1666f, 2.1666f);
    glVertex3f(-4.5f, -2.1666f, -2.1666f);
    glTexCoord2f(1.0f, 1.0f);
    glNormal3f(-4.5f, 2.1666f, 2.1666f);
    glVertex3f(0.0f, -2.1666f, 4.3334f);
    glTexCoord2f(0.5f, 0.0f);
    glNormal3f(-4.5f, 2.1666f, 2.1666f);
    glVertex3f(0.0f, 4.3334f, 0.0f);

    glTexCoord2f(0.0f, 1.0f);
    glNormal3f(4.5f, 2.1666f, 2.1666f);
    glVertex3f(0.0f, -2.1666f, 4.3334f);
    glTexCoord2f(1.0f, 1.0f);
    glNormal3f(4.5f, 2.1666f, 2.1666f);
    glVertex3f(4.5f, -2.1666f, -2.1666f);
    glTexCoord2f(0.5f, 0.0f);
    glNormal3f(4.5f, 2.1666f, 2.1666f);
    glVertex3f(0.0f, 4.3334f, 0.0f);

    glTexCoord2f(0.0f, 1.0f);
    glNormal3f(0.0f, 2.1666f, -4.3334f);
    glVertex3f(4.5f, -2.1666f, -2.1666f);
    glTexCoord2f(1.0f, 1.0f);
    glNormal3f(0.0f, 2.1666f, -4.3334f);
    glVertex3f(-4.5f, -2.1666f, -2.1666f);
    glTexCoord2f(0.5f, 0.0f);
    glNormal3f(0.0f, 2.1666f, -4.3334f);
    glVertex3f(0.0f, 4.3334f, 0.0f);

    glEnd();
}
```

Rezultat działania programu prezentują poniższe zrzuty ekranu.



Oteksturowany czworościan



Oteksturowany czworościan

Kod źródłowy

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