SPRAWOZDANIE

Zajęcia: Grafika komputerowa
Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium: Grafika Komputerowa
06.04.2020
Temat: „Światło i materiały w OpenGL”
Wariant: -

Michał Krzyżowski
Informatyka I stopień,
stacjonarne,
4 semestr,
Gr.1b

1. Polecenie:

W obecnych laboratoriach należało stworzyć piramidę (w poleceniu nie jest sprecyzowane jakim ostrosłupem ma ona być) z użyciem różnych materiałów i umieszczeniem jej na „podstawie”. Po narysowaniu widać że włączone jest jedynie oświetlenie podstawowe. W ramach laboratorium należało poprawić te oświetlenie. Do wyboru był języka Java i C.

2. Wprowadzane dane:

Dane w zadaniu ustawione są na stałe (znajdują się bezpośrednio w kodzie) są to choćby materiały jakie są używane do budowy piramidy, ustawienia kolorów i światła. Po uruchomieniu samego programu już nie wprowadzamy żadnych danych, poza tym iż możliwe jest poinformowanie aplikacji że chcemy przesunąć obraz, na co sama aplikacja zareaguje.

3. Wykorzystane komendy:

Komendy wykorzystywane podczas pracy nad projektem to instrukcje: glVertex3f która służy do ustawienia punktu w przestrzeni 3d. Dzięki tej komendzie możliwe było rysowanie brył 3d gdyż składają się one z wierzchołków które połączone są ze sobą tworząc całość. Kolejna komenda to glMaterialfv(), przyjmuje ona 3 lub 4 parametry. W moim kodzie przeładowanie przyjmujące 4-ry parametry jest wykorzystywane. Pierwszy parametr informuje o stronie bryły na którą ma to działać – frontowa/ tylnia/ tylnia i fronotwa, drugi parametr mówi o właściwościach materiału dostępne opcje to GL_Ambient/ gl_diffuse, gl_specular, gl_emission albo też połączenie dwóch z nich na raz gl_ambient_and_diffuse, trzeci parametr to tablica dla kolorowych komponentów, która jeżeli dobrze zrozumiałem jest także używana przy obliczaniu oświetlenia. W moim kodzie rozbiłem to na 4 osobne komendy, które kolejno ustawiają poprzez te tablice kolejno kolor otoczenia, kolor rozproszony, kolor otoczenia i ostatnia pojedyncza wartość odpowiadająca za połysk. Kolejna komenda to glenable która włącza dane światło, podając parametr np. gl_light0/ gl_light1. Kolejna komenda tyczy się wcześniej aktywowanego światła, zmiany jego właściwości jest to glLightfv(). Funkcja ta przyjmuje kolejno jako parametr pierwszy światło które ma być skonfigurowane, drugi informuje która z właściwości światła ma ulec konfiguracji (tutaj możemy ustawić pozycje danego światła), 3-ci parametr tyczy się koloru. Komenda gllightmodelfv() odpowiada za ustawienia globalne światła.

Poniższy kod prezentuje rysowanie piramidy, próby konfigurowania ustawień światła:

```
GL2 gl2 = drawable.getGL().getGL2(); // The object that contains all the OpenGL methods.

gl2.glClear( GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT );

gl2.glLoadIdentity();
glu.gluLookAt( 0,8,40, 0,1,0, 0,1,0 );  // viewing transform
float[] color= {1.0f,1.0f,0.0f,1f,
        0.0f,1.0f,1.0f,1f,
        0.0f,1.0f,0.0f,1f,
        50f};
float[] gcolor= {0.6f, 0, 0.6f, 1 };
gl2.glRotated( rotateY, 0, 1, 0 );  // modeling transform: rotation of the scene about y-axis
gl2.glLightModelfv(GL2.GL_LIGHT_MODEL_TWO_SIDE,gcolor,1);

float[] gray = { 0.6f, 0.6f, 0.6f, 1 };
float[] zero = { 0, 0, 0, 1 };

gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_AMBIENT_AND_DIFFUSE, gray, 0);
gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_SPECULAR, zero, 0);
gl2.glPushMatrix();
gl2.glTranslated(0,-1.5,0); // Move top of stage down to y = 0
gl2.glScaled(1, 0.05, 1); // Stage will be one unit thick,
glut.glutSolidCube(20);
gl2.glPopMatrix();

// TODO draw some shapes!
gl2.glEnable(gl2.GL_LIGHTING);
gl2.glEnable(gl2.GL_COLOR_MATERIAL);
gl2.glEnable(gl2.GL_LIGHT0);
```

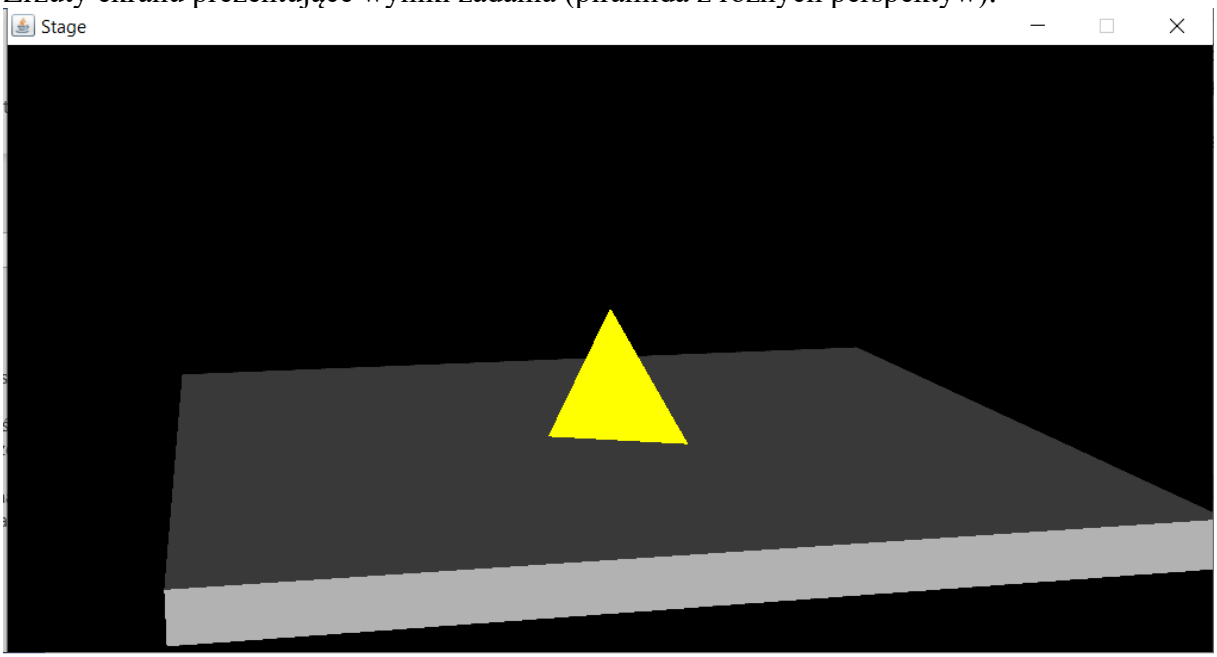
```
// TODO draw some shapes!
gl2.glEnable(gl2.GL_LIGHTING);
gl2.glEnable(gl2.GL_COLOR_MATERIAL);
gl2.glEnable(gl2.GL_LIGHT0);
gl2.glEnable(gl2.GL_LIGHT1);
gl2.glEnable(gl2.GL_LIGHT4);
gl2.glEnable(gl2.GL_LIGHT7);
float[] position = { 1,1,3,1 };
gl2.glLightfv(gl2.GL_LIGHT1, gl2.GL_POSITION, position,0);
gl2.glBegin( gl2.GL_TRIANGLES );

gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_AMBIENT_AND_DIFFUSE,color , 0);
gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_SPECULAR,color , 8);
gl2.glMaterialf(gl2.GL_FRONT_AND_BACK, gl2.GL_SHININESS, color[12]);
float[] blue1  = {0,4, 0,4, 0,6, 1};
float[] blue2  = {0,0, 0, 0,8, 1};
float[] blue3  = {0,0, 0, 0,15, 1};
gl2.glLightfv (gl2.GL_LIGHT1, gl2.GL_DIFFUSE, blue1,0);
gl2.glLightfv (gl2.GL_LIGHT1, gl2.GL_SPECULAR, blue2,0);
gl2.glLightfv (gl2.GL_LIGHT1, gl2.GL_AMBIENT, blue3,0);
gl2.glLightfv (gl2.GL_LIGHT4, gl2.GL_DIFFUSE, blue1,0);
gl2.glLightfv (gl2.GL_LIGHT4, gl2.GL_SPECULAR, blue2,0);
gl2.glLightfv (gl2.GL_LIGHT4, gl2.GL_AMBIENT, blue3,0);
gl2.glLightfv (gl2.GL_LIGHT7, gl2.GL_DIFFUSE, blue1,0);
gl2.glLightfv (gl2.GL_LIGHT7, gl2.GL_SPECULAR, blue2,0);
gl2.glLightfv (gl2.GL_LIGHT7, gl2.GL_AMBIENT, blue3,0);
gl2.glColor3f(1f, 1f, 0f);
gl2.glVertex3f( 0.0f, 2.f, 0.0f ); //czubek
gl2.glVertex3f( -2.0f, -2.0f, 2.0f );
gl2.glVertex3f( 2.0f, -2.0f, 2.0f);//czemu 2ka działa skoro od -1 do 1 miało być?
//green
gl2.glVertex3f( 0.0f, 2.0f, 0.0f);
gl2.glVertex3f( -2.0f, -2.0f, 2.0f);
```

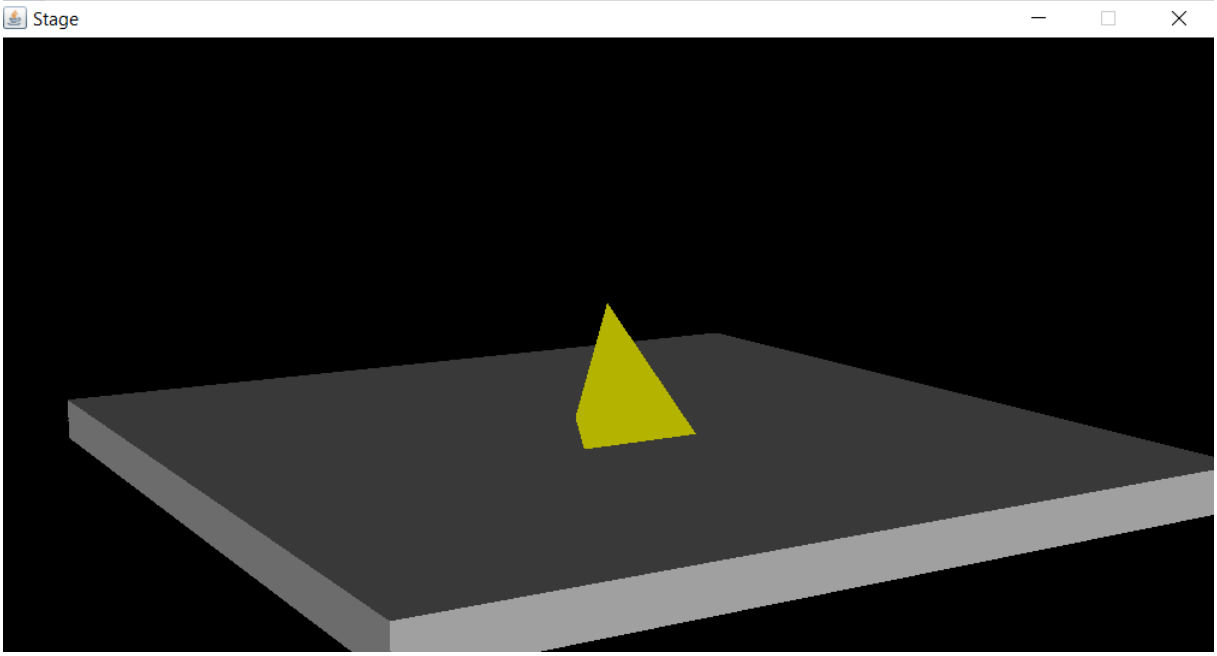
```
    //green
    gl2.glVertex3f( 0.0f, 2.0f, 0.0f);
    gl2.glVertex3f( -2.0f, -2.0f, 2.0f);
    gl2.glVertex3f( 0.0f, -2.0f, -2.0f);

    float[] color2= {1.0f,0.0f,0.0f,1f,
            0.0f,0.0f,1.0f,1f,
            1.0f,1.0f,1.0f,1f,
            25f};
    gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_AMBIENT,color , 0);
    gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_DIFFUSE,color , 4);
    gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_SPECULAR,color , 8);
    gl2.glMaterialf(gl2.GL_FRONT_AND_BACK, gl2.GL_SHININESS, color[12]);

    gl2.glVertex3f( 0.0f, 2.0f, 0.0f);
    gl2.glVertex3f( 0.0f, -2.0f, -2.0f);
    gl2.glVertex3f( 2.0f, -2.0f, 2.0f);
    //red
    gl2.glVertex3f( -2.0f, -2.0f, 2.0f);
    gl2.glVertex3f( 0.0f, -2.0f, -2.0f);
    gl2.glVertex3f( 2.0f, -2.0f, 2.0f);
    gl2.glColor3f(0.6f, 0.6f, 0.6f);
    gl2.glEnd();
} // end display()
```

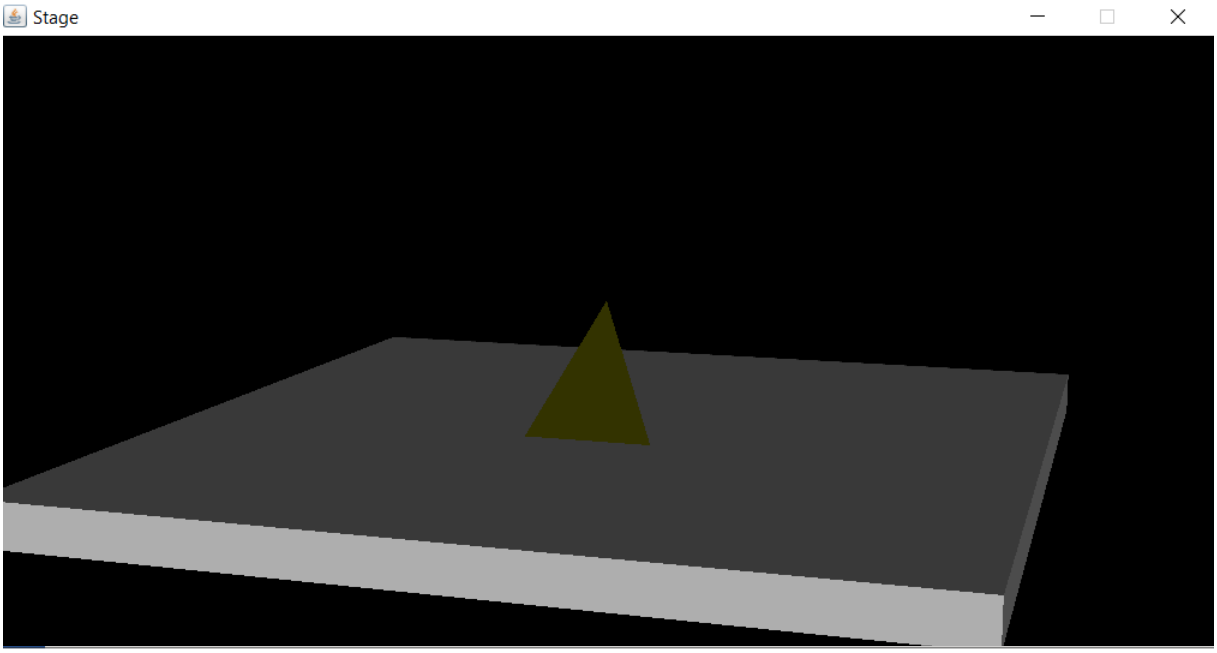
Link do zdalnego repozytorium zawierającego projekt:
https://github.com/Jarverr/GrafikaKomputerowaLab6

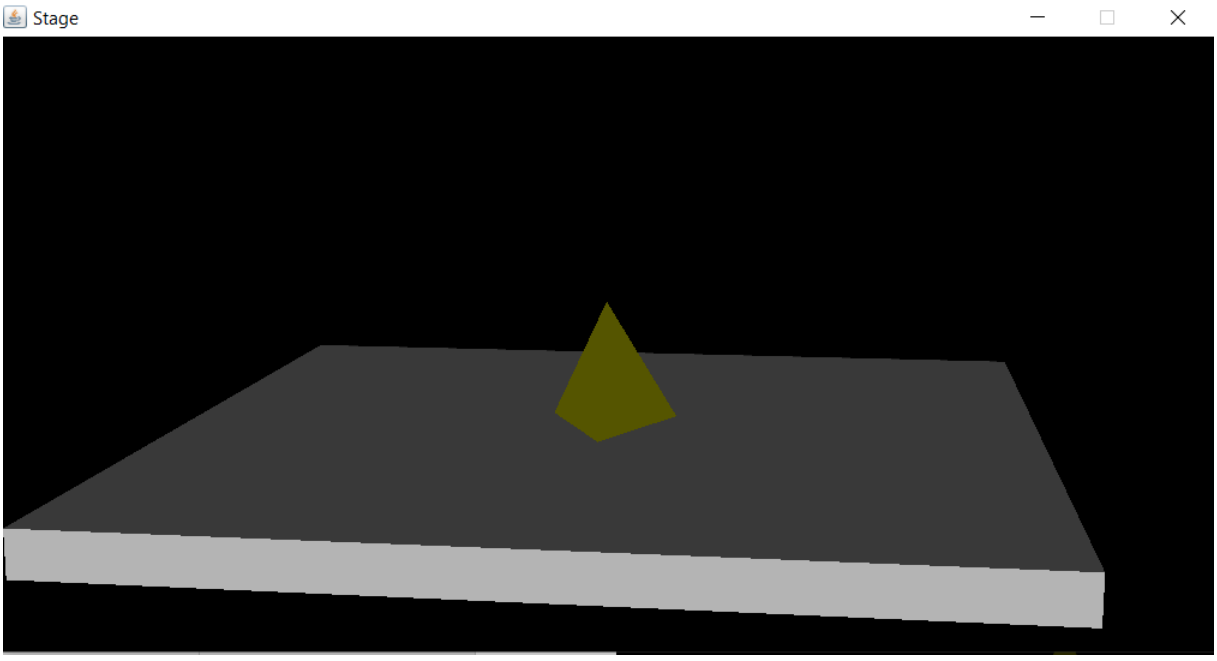
4. Wynik działania:

Zrzuty ekranu prezentujące wyniki zadania (piramida z różnych perspektyw):









5. Wnioski:

Dzięki możliwością jakie daje biblioteka OpenGL, jestem w stanie w stosunkowo prostu sposób stworzyć trójwymiarową bryłę a następnie sformatować światło, które na nią pada. Niestety z własnych doświadczeń, mam wrażenie, że operowanie tymi światłami jest mało intuicyjne, choć może to wynikać z braku doświadczenia. Na pewno możliwości modyfikacji światła jest wiele, tako samo ustawień właściwości obiektu/ bryły jak powinna się zachowywać gdy spotyka się ze światłem. Czyni to całkiem solidne narzędzie do operowania zachowaniem światła w grafice.