SPRAWOZDANIE

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

Laboratorium 6

Data: 2024

Temat: Światło i materiały w OpenGL

Wariant:

Dwunastokąt, red plastic

Bartosz
Gruszeczka
Informatyka I stopień,
stacjonarne,
4 semestr
Gr. 2b

1. Polecenie:

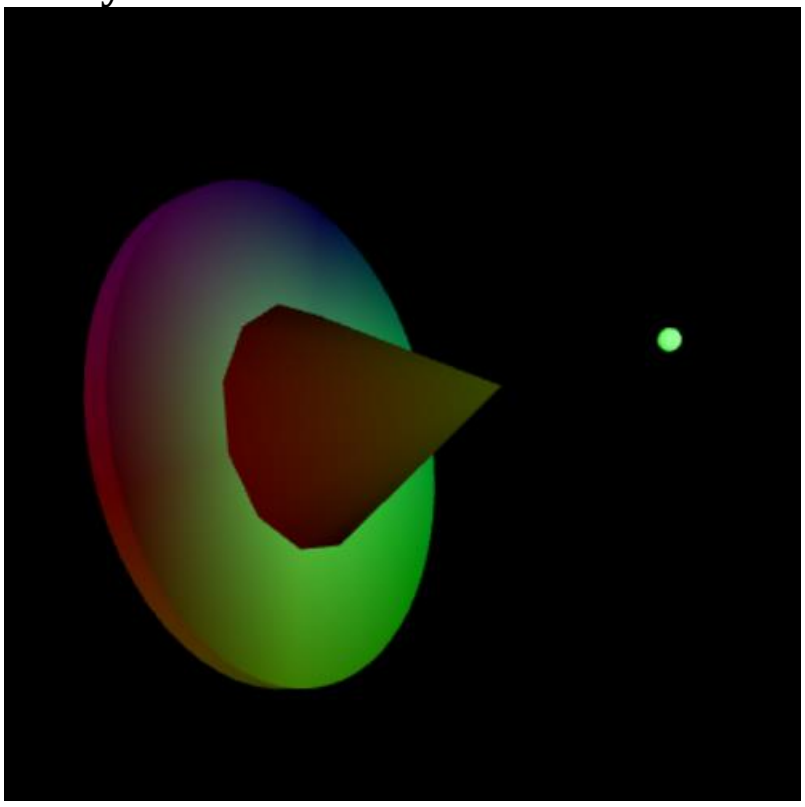
Celem jest stworzenie pyramidy z użyciem różnych materiałów okrelonych wariantem zadania i umieszczenie jej na „podstawie”. Użytkownik może obracać podstawę wokół osi Y, przeciągając mysz w poziomie. Scena wykorzystuje globalne światło otoczenia (ambient) oraz źródło światła o kształcie kuli z możliwością animacji obrotu wokół pyramidy.

2. Wykorzystane komendy:

```
var material = [ /* 8, 19 - "red plastic" */   0.0, 0.0, 0.0, 1.0, 0.5, 0.0, 0.0, 1.0, 0.7, 0.6, 0.6, 1.0, .25 * 128];
var pyramidDiffuse = [material[4], material[5], material[6], 1];
var pyramidSpecular = [material[8], material[9], material[10], 1];
var pyramidShininess = material[12];
 function pyramid() {
    glBegin(GL_TRIANGLES);
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, pyramidDiffuse);
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, pyramidSpecular);
    glMateriali(GL_FRONT_AND_BACK, GL_SHININESS, pyramidShininess);

    var baseRadius = 1;
    var numSides = 12;
    var baseAngleIncrement = (2 * Math.PI) / numSides;
    var height = 2;

    for (var i = 0; i < numSides; i++) {
        var angle1 = i * baseAngleIncrement;
        var x1 = baseRadius * Math.cos(angle1);
        var z1 = baseRadius * Math.sin(angle1);

        var angle2 = (i + 1) * baseAngleIncrement;
        var x2 = baseRadius * Math.cos(angle2);
        var z2 = baseRadius * Math.sin(angle2);

        // Ściana boczna
        glNormal3f(x1, 0.3, z1);
        glVertex3d(x1, 0.3, z1);
        glNormal3f(x2, 0.3, z2);
        glVertex3d(x2, 0.3, z2);
        glNormal3f(0, 1, 0);
        glVertex3d(0, height, 0);
    }
    glEnd();
 }
function display() {

    glClearColor(0,0,0,1);
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
    camera.apply();
    lights();

    if (ambientLight.checked) glLightModelfv(GL_LIGHT_MODEL_AMBIENT, [0.15, 0.15, 0.15, 1] );
    else glLightModelfv(GL_LIGHT_MODEL_AMBIENT, [0, 0, 0, 1] );

    if (drawBase.checked) {
        glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, [0, 0, 0, 1] );
        glPushMatrix();
        glTranslated(0, -5, 0);
        glRotated(-90, 1, 0, 0);
        glScaled(10,10,0.5);
        drawCylinder();
        glPopMatrix();
    }

    glColor3d(1,1,1);  // sets diffuse and ambient color for teapot
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, [0.2, 0.2, 0.2, 1]);

    glPushMatrix();
    glTranslatef(0,-6,0);
    glScalef(5,5,5);
    pyramid();
    glPopMatrix();
}
```


Link do repozytorium Github

3. Wynik działania:



4. Wnioski:

Dzięki zaawansowanym modelom oświetlenia można uzyskać bardzo realistyczne efekty świetlne. OpenGL pozwala na różne typy świateł i techniki oświetleniowe. Dzięki wsparciu sprzętowemu, operacje związane z oświetleniem mogą być wykonywane bardzo szybko, co jest kluczowe w aplikacjach wymagających wysokiej wydajności, takich jak gry komputerowe.
Jednakże mplementacja zaawansowanych efektów świetlnych może być skomplikowana i wymagać dużej wiedzy z zakresu grafiki komputerowej. Zaawansowane techniki oświetlenia mogą być obciążające dla procesora graficznego, co może negatywnie wpływać na wydajność w mniej wydajnych systemach.