

Uniwersytet Bielsko-Bialski

Obraz zawierający tekst, Czcionka, logo, Grafika

Opis wygenerowany automatycznie

***Grafika Komputerowa***

# (ćwiczenia laboratoryjne)

**Laboratoria numer: 5.**

# Temat ćwiczenia: Geometria trójwymiarowa OPEN GL

Data wykonania ćwiczenia: 10.04.2024

Wykonała:

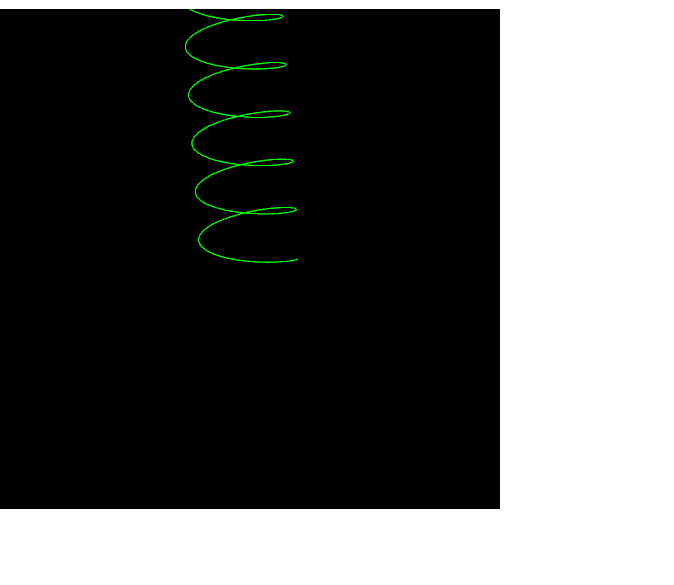
Krzysztof Wierzbicki

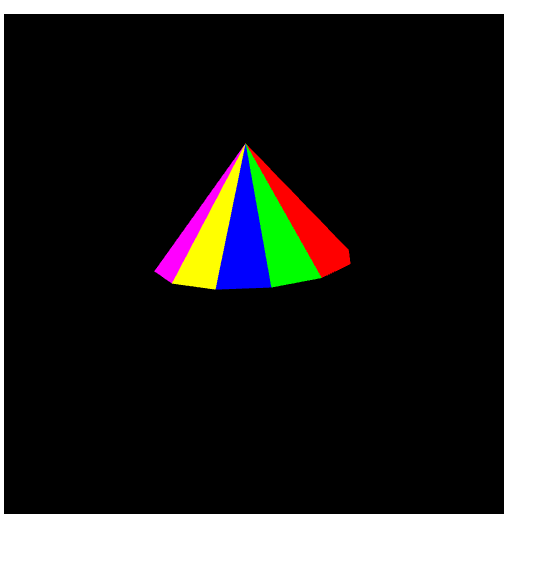
<https://github.com/ChrisGH00/GK_LAB5>

# Cel zadania

1. Stworzyć dwa obiekty przy użyciu OpenGL (w języku JavaScript). Po uruchomieniu zakończonego programu naciśnięcie jednego z klawiszy numerycznych 1 lub 2 spowoduje wybranie wyświetlanego obiektu. Program ustawia wartość zmiennej globalnej, objectNumber, aby powiedzieć, który obiekt ma zostać narysowany. Użytkownik może obracać obiekt za pomocą klawiszy strzałek, PageUp, PageDown i Home. Podprogram display() jest wywoływany, aby narysować obiekt.
2. Obiekt 1. Korkociąg wokół osi {x | y | z} zawierający N obrotów. Punkty są stopniowo powiększane. Ustalić aktualny kolor rysujący na {zielony | niebieski | brązowy | … }.
3. Obiekt 2. Pyramida, wykorzystując dwa wachlarze trójkątów oraz modelowanie hierarchiczne (najpierw tworzymy podprogramę rysowania jednego trójkonta; dalej wykorzystując przekształcenia geometryczne tworzymy pyramidę). Podstawą pyramidy jest wielokąt o N wierzchówkach.

# 2. Wynik działania





<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Unlit Cube Example</title>

<script src="glsim.js"></script>  <!-- load the OpenGL 1.1 simulation code -->

<script>

    let rotateX = 15;

    let rotateY = -15;

    let rotateZ = 0;

    var objectNumber = 1;

    function drawCorkscrew() {

    var turns = 11; // Zmieniona liczba obrotów dla efektu sprężyny

    var angleStep = Math.PI / 20; // Krok kątowy dla płynniejszego zakrętu (zmniejszony)

    var radius = 0.2; // Promień spirali

    var heightStep = 0.005; // Krok wysokości między kolejnymi obręczami

    glColor3f(0, 1, 0); // Zielony kolor dla odmiany

    glBegin(GL\_LINE\_STRIP);

    for (var i = 0; i < turns \* 40; i++) { // Więcej iteracji dla dłuższej sprężyny

        var angle = angleStep \* i;

        var x = radius \* Math.cos(angle);

        var y = heightStep \* i;

        var z = radius \* Math.sin(angle);

        glVertex3f(x, y, z);

    }

    glEnd();

}

function drawTriangle() {

    glBegin(GL\_TRIANGLES);

    glVertex3f(0, 1, 0);

    glVertex3f(-1, -1, 1);

    glVertex3f(1, -1, 1);

    glEnd();

}

function drawHexagonalPyramid() {

    var baseRadius = 0.4; // Promień podstawy piramidy

    var height = 0.5; // Wysokość piramidy

    // Tablica kolorów dla każdej ściany piramidy

    var colors = [

        [1, 0, 0], // Czerwony

        [0, 1, 0], // Zielony

        [0, 0, 1], // Niebieski

        [1, 1, 0], // Żółty

        [1, 0, 1], // Fioletowy

        [0, 1, 1], // Cyjan

        [1, 0.5, 0], // Pomarańczowy

        [0.5, 0, 1], // Indygo

        [0.5, 0.5, 0.5], // Szary

        [0, 0, 0] // Czarny

    ];

    // Rysowanie boków piramidy

    glBegin(GL\_TRIANGLES);

    for (let i = 0; i < 11; i++) { // Zmieniona liczba boków piramidy na 11

        glColor3fv(colors[i % colors.length]); // Ustawienie koloru dla danej ściany

        // Wierzchołek piramidy

        glVertex3f(0, height, 0);

        // Lewy dolny wierzchołek podstawy

        glVertex3f(baseRadius \* Math.cos((i + 1) \* 2 \* Math.PI / 11), 0, baseRadius \* Math.sin((i + 1) \* 2 \* Math.PI / 11));

        // Prawy dolny wierzchołek podstawy

        glVertex3f(baseRadius \* Math.cos(i \* 2 \* Math.PI / 11), 0, baseRadius \* Math.sin(i \* 2 \* Math.PI / 11));

    }

    glEnd();

    // Rysowanie podstawy piramidy

    glColor3f(0, 1, 0); // Ustawienie koloru niebieskiego dla podstawy piramidy

    glBegin(GL\_TRIANGLE\_FAN);

    glVertex3f(0, 0, 0); // Środek podstawy

    for (let i = 0; i <= 11; i++) { // Zmieniona liczba boków piramidy na 11

        glVertex3f(baseRadius \* Math.cos(i \* 2 \* Math.PI / 11), 0, baseRadius \* Math.sin(i \* 2 \* Math.PI / 11));

    }

    glEnd();

}

    function square(r, g, b) {

        glColor3f(r,g,b);

        glBegin(GL\_TRIANGLE\_FAN);

        glVertex3f(-0.5, -0.5, 0.5);

        glVertex3f(0.5, -0.5, 0.5);

        glVertex3f(0.5, 0.5, 0.5);

        glVertex3f(-0.5, 0.5, 0.5);

        glEnd();

    }

    function cube(size) {

        glPushMatrix();

        glScalef(size,size,size);

        square(1, 0, 0);

        glPushMatrix();

        glRotatef(90, 0, 1, 0);

        square(0, 1, 0);

        glPopMatrix();

        glPushMatrix();

        glRotatef(-90, 1, 0, 0);

        square(0, 0, 1);

        glPopMatrix();

        glPushMatrix();

        glRotatef(180, 0, 1, 0);

        square(0, 1, 1);

        glPopMatrix();

        glPushMatrix();

        glRotatef(-90, 0, 1, 0);

        square(1, 0, 1);

        glPopMatrix();

        glPushMatrix();

        glRotatef(90, 1, 0, 0);

        square(1, 1, 0);

        glPopMatrix();

        glPopMatrix();

    }

    function display() {

        glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

    glLoadIdentity();

    glRotatef(rotateZ, 0, 0, 1);

    glRotatef(rotateY, 0, 1, 0);

    glRotatef(rotateX, 1, 0, 0);

    switch (objectNumber) {

        case 1:

        drawCorkscrew(); // Rysowanie sprężyny stożkowej

            break;

        case 2:

            drawHexagonalPyramid(); // Rysowanie piramidy sześciokątnej

            break;

        // Możesz dodać więcej obiektów tutaj

    }

    }

    function initGL() {

        glMatrixMode(GL\_PROJECTION);

        glOrtho(-1, 1, -1, 1, -1, 1);

        glMatrixMode(GL\_MODELVIEW);

        glEnable(GL\_DEPTH\_TEST);

        glClearColor(0, 0, 0, 1);

    }

    function doKeyDown(evt) {

        let key = evt.keyCode;

        if (key == 49) { // Klawisz '1'

        objectNumber = 1;

    } else if (key == 50) { // Klawisz '2'

        objectNumber = 2;

    }

        if ( key == 37 ) // left

           rotateY -= 15;

        else if ( key == 39 ) // right

           rotateY += 15;

        else if ( key == 40) // down

           rotateX += 15;

        else if ( key == 38 ) // up

           rotateX -= 15;

        else if ( key == 33 ) // page up

           rotateZ += 15;

        else if ( key == 34 ) // page down

           rotateZ -= 15;

        else if ( key == 36 )  // home

           rotateX = rotateY = rotateZ = 0;

        if (key >=34 && key <= 40) {

            evt.preventDefault();

        }

        display();

    }

    function init() {

        try {

            glsimUse("glcanvas"); // OpenGL will draw to the canvas with id="glcanvas".

        }

        catch (e) {

            document.getElementById("canvas-holder").innerHTML =

                "Sorry, an error occured:<br>" + e;

            return;

        }

        document.onkeydown = doKeyDown;

        initGL();

        display();

    }

</script>

</head>

<body onload="init()">

<h3>A Simple Unlit Cube in OpenGL 1.1</h3>

<p>(Rotate using arrow keys, page up, page down, and home keys, switch between 1(Corkscrew) and 2(Pyramid))</p>

<noscript>

<p><b>Sorry, this page requires JavaScript!</b></p>

</noscript>

<div id="canvas-holder">

<canvas id="glcanvas" width="500" height="500"></canvas>

</div>

</body>

</html>

# 3. Wnioski

Interaktywność i sterowanie: Ćwiczenie pozwala na naukę interaktywnego programowania w OpenGL przy użyciu JavaScript. Możliwość sterowania obiektem za pomocą klawiatury (strzałki, PageUp, PageDown) oraz zmiana obiektów poprzez klawisze numeryczne (1 i 2) umożliwiają eksperymentowanie z różnymi ustawieniami i parametrami.

Rysowanie geometrii 3D: Praktyczna implementacja funkcji do rysowania spiral stożkowych i piramid sześciokątnych pozwala na zrozumienie jak używać podstawowych elementów OpenGL, takich jak glBegin(), glEnd(), oraz jak manipulować transformacjami geometrii przy użyciu glRotatef(), glScalef() i glPushMatrix(), glPopMatrix().

Kolorowanie i estetyka: Nauka o kolorowaniu ścian piramid oraz spiral stożkowych zwiększa estetykę i czytelność renderowanych obiektów. Zastosowanie różnych kolorów dla każdej ściany piramidy pozwala na wizualne wyróżnienie poszczególnych elementów geometrycznych.