**SPRAWOZDANIE**

Zajęcia: Grafika komputerowa

Prowadzący: mgr Mikołaj Grygiel

**Laboratorium 6**

28.04.2024

**Temat:** Geometria trójwymiarowa biblioteki OpenGL

**Wariant**: 1

Radosław Skrzypczyński

Informatyka I stopień,

niestacjonarne,

4 semestr,

Gr. 1B

1. **Polecenie:**

Stworzyć dwa obiekty przy użzyciu OpenGL. Po uruchomieniu 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 możze obracać obiekt za pomocą klawiszy strzałek, PageUp, PageDown i Home. Podprogram display() jest wywoływany, aby narysować obiekt.

Obiekt 1. Korkociąg wokół osi x | y | z zawierający N obrotów. Punkty są stopniowo powiększane. Ustalić aktualny kolor rysujący na zielony | niebieski | brązowy | . . . .

Obiekt 2. Pyramida, wykorzystuja˛c dwa wachlarze trójkątów oraz modelowanie hierarchiczne (najpierw tworzymy podprogram rysowania jednego trójkąta; dalej wykorzystując przekształcenia geometryczne tworzymy pyramidę). Podstawą pyramidy jest wielokąt o N wierzchołkach.

**2. Wprowadzane dane:**

Klawisze 1 i 2 służące do zmiany obiektów

**3. Wykorzystane komendy:**

*<!DOCTYPE html>*

*<html>*

*<!--*

*Use OpenGL and GLUT to draw a simple cube*

*with each face being a different color. Rotations*

*can be applied with the arrow keys, the page up*

*key, and the page down key. The home key will set*

*all rotations to 0. Initial rotations about the*

*x, y, and z axes are 15, -15, and 0.*

*This program is meant as an example of using modeling*

*transforms, with glPushMatrix and glPopMatrix.*

*Note that this program does not use lighting.*

*-->*

*<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; // rotations of cube about the axes*

*let rotateY = -15;*

*let rotateZ = 0;*

*let objectNumber = 1;*

*function corkscrew(r, g, b, spiralCount, radius, distance) {*

*let startSize = 0.01;*

*let endSize = 0.02;*

*let rn = r;*

*let gn = g;*

*let bn = b;*

*let heightStep = distance / (360 \* spiralCount);*

*let height = -distance/2;*

*let sizePoint = (endSize - startSize) / (360 \* spiralCount);*

*for (let i =0; i <= 360 \* spiralCount; i++) {*

*let angle = i \* ((2 \* Math.PI) / 360);*

*let x = (radius - (radius \* height / distance)) \* Math.cos(angle);*

*let y = (radius - (radius \* height / distance)) \* Math.sin(angle);*

*let scale = startSize + sizePoint \* i;*

*if (i % 60 == 0){*

*rn+= r > 125 ? -2 : 2;*

*gn+= g > 125 ? -2 : 2;*

*bn+= b > 125 ? -2 : 2;*

*}*

*glColor3ub(rn, gn, bn);*

*glPushMatrix();*

*glTranslatef(x, y, height);*

*glScalef(scale, scale, scale);*

*glBegin(GL\_LINE\_STRIP);*

*glVertex3f(-0.5, -0.5, 0);*

*glVertex3f(0.5, -0.5, 0);*

*glVertex3f(0.5, 0.5, 0);*

*glVertex3f(-0.5, 0.5, 0);*

*glEnd();*

*glPopMatrix();*

*height += heightStep;*

*}*

*}*

*function pyramid(size, sides, height){*

*glPushMatrix();*

*glScalef(size,size,size);*

*glRotatef(90, 0, 1, 0);*

*polygon(sides,1,0,0,1);*

*var angle = (Math.PI \* 2) / sides;*

*let prevX = 0;*

*let prevY = 0;*

*for (var i = 0; i <= sides; i++) {*

*var x = (Math.cos(i \* angle)/2);//\*size;*

*var y = (Math.sin(i \* angle)/2);//\*size;*

*if(i>0){*

*let r = Math.floor(Math.random() \* 255);*

*let g = Math.floor(Math.random() \* 255);*

*let b = Math.floor(Math.random() \* 255);*

*triangle(height,x,y,prevX,prevY,r,g,b,size);*

*}*

*prevX = x;*

*prevY = y;*

*}*

*glPopMatrix();*

*}*

*function triangle(height,x1,y1,x2,y2,r,g,b,size){*

*glColor3ub(r,g,b);*

*glPushMatrix();*

*glRotatef(90, 0, 1, 0);*

*glRotatef(18, 0, 0, 1);*

*glBegin(GL\_TRIANGLE\_FAN);*

*glVertex3f(0,0,height/2);*

*glVertex3f(x1,y1,0);*

*glVertex3f(x2,y2,0);*

*glEnd();*

*glPopMatrix();*

*}*

*function polygon(sides,r,g,b,scale){*

*glColor3f(r,g,b);*

*glBegin(GL\_TRIANGLE\_FAN);*

*var angle = (Math.PI \* 2) / sides;*

*for (var i = 0; i <= sides; i++) {*

*var x = (Math.cos(i \* angle)/2);//\*scale;*

*var y = (Math.sin(i \* angle)/2);//\*scale;*

*glVertex3f(0, x, y);*

*}*

*glEnd();*

*}*

*function tube(sides,r,g,b,scale,startX,startY,startZ,offsetX,offsetY,offsetZ){*

*glColor3f(r,g,b);*

*glBegin(GL\_TRIANGLE\_FAN);*

*var angle = (Math.PI \* 2) / sides;*

*for (var i = 0; i <= sides; i++) {*

*var x = (Math.cos(i \* angle)/2);//\*scale;*

*var y = (Math.sin(i \* angle)/2);//\*scale;*

*glVertex3f(startZ, startX, startY);*

*glVertex3f(startZ+offsetZ,startX+x+offsetX,startY+y+offsetY);*

*}*

*glEnd();*

*}*

*function pentagon(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 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) { // draws a cube with side length = size*

*glPushMatrix();*

*glScalef(size,size,size); // scale unit cube to desired size*

*square(1, 0, 0); // red front face*

*glPushMatrix();*

*glRotatef(90, 0, 1, 0);*

*square(0, 1, 0); // green right face*

*glPopMatrix();*

*glPushMatrix();*

*glRotatef(-90, 1, 0, 0);*

*square(0, 0, 1); // blue top face*

*glPopMatrix();*

*glPushMatrix();*

*glRotatef(180, 0, 1, 0);*

*square(0, 1, 1); // cyan back face*

*glPopMatrix();*

*glPushMatrix();*

*glRotatef(-90, 0, 1, 0);*

*square(1, 0, 1); // magenta left face*

*glPopMatrix();*

*glPushMatrix();*

*glRotatef(90, 1, 0, 0);*

*square(1, 1, 0); // yellow bottom face*

*glPopMatrix();*

*glPopMatrix(); // Restore matrix to its state before cube() was called.*

*}*

*// ----------------------------------------------------------------------*

*function display() { // Draws the image.*

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

*glLoadIdentity(); // Set up modelview transform.*

*glRotatef(rotateZ,0,0,1);*

*glRotatef(rotateY,0,1,0);*

*glRotatef(rotateX,1,0,0);*

*if (objectNumber == 1){*

*corkscrew(50, 50, 120, 5, 0.3, 1.5);*

*}*

*else{*

*pyramid(1, 5, 1);*

*}*

*}*

*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 == 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;*

*else if (key == 49)*

*objectNumber = 1;*

*else if (key == 50)*

*objectNumber = 2;*

*if (key >=34 && key <= 40) {*

*evt.preventDefault();*

*}*

*display();*

*}*

*function init() { // Called by <body onload="init()">, when the page has loaded.*

*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.)</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>*

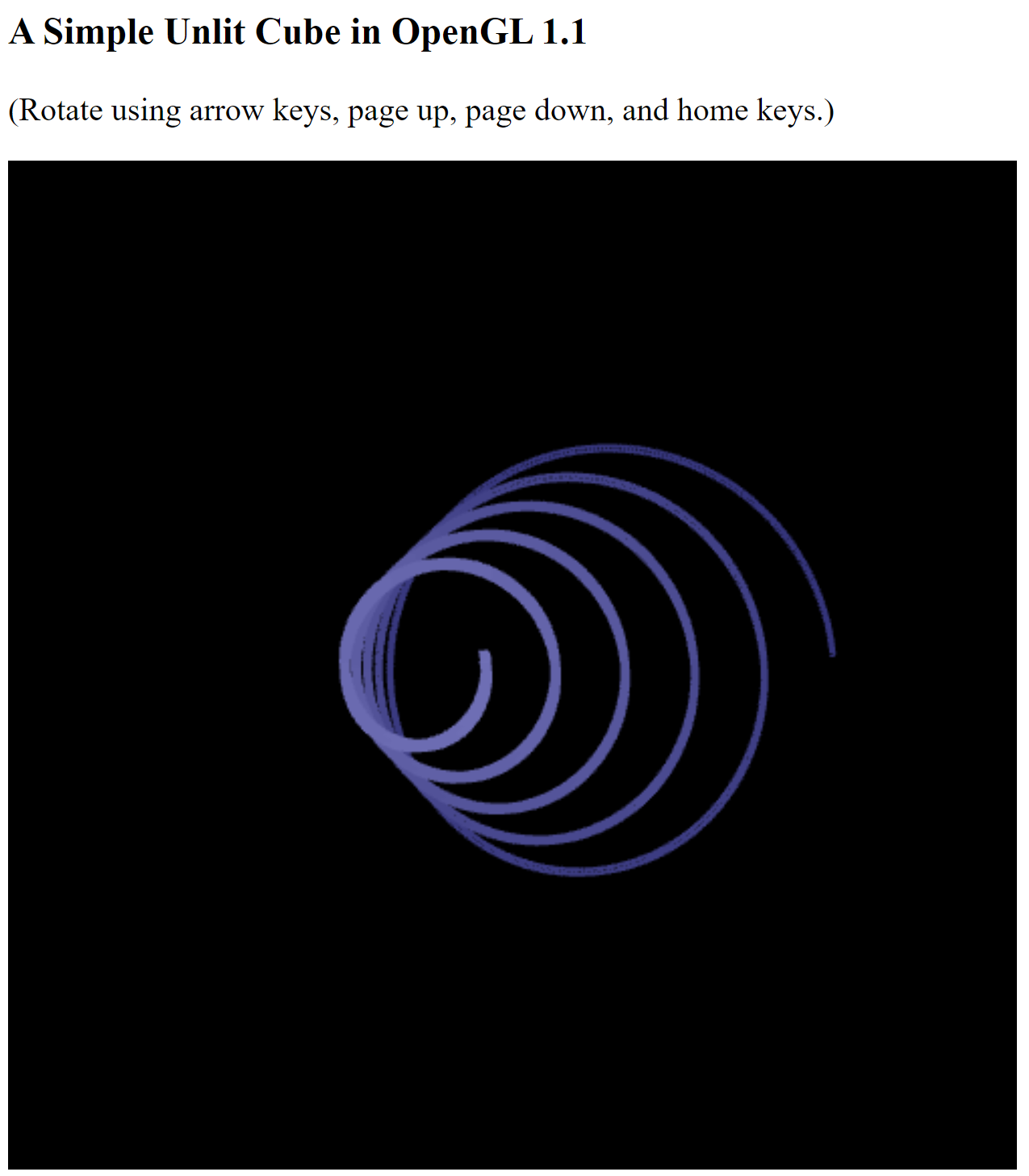
*</body>*

*</html>*

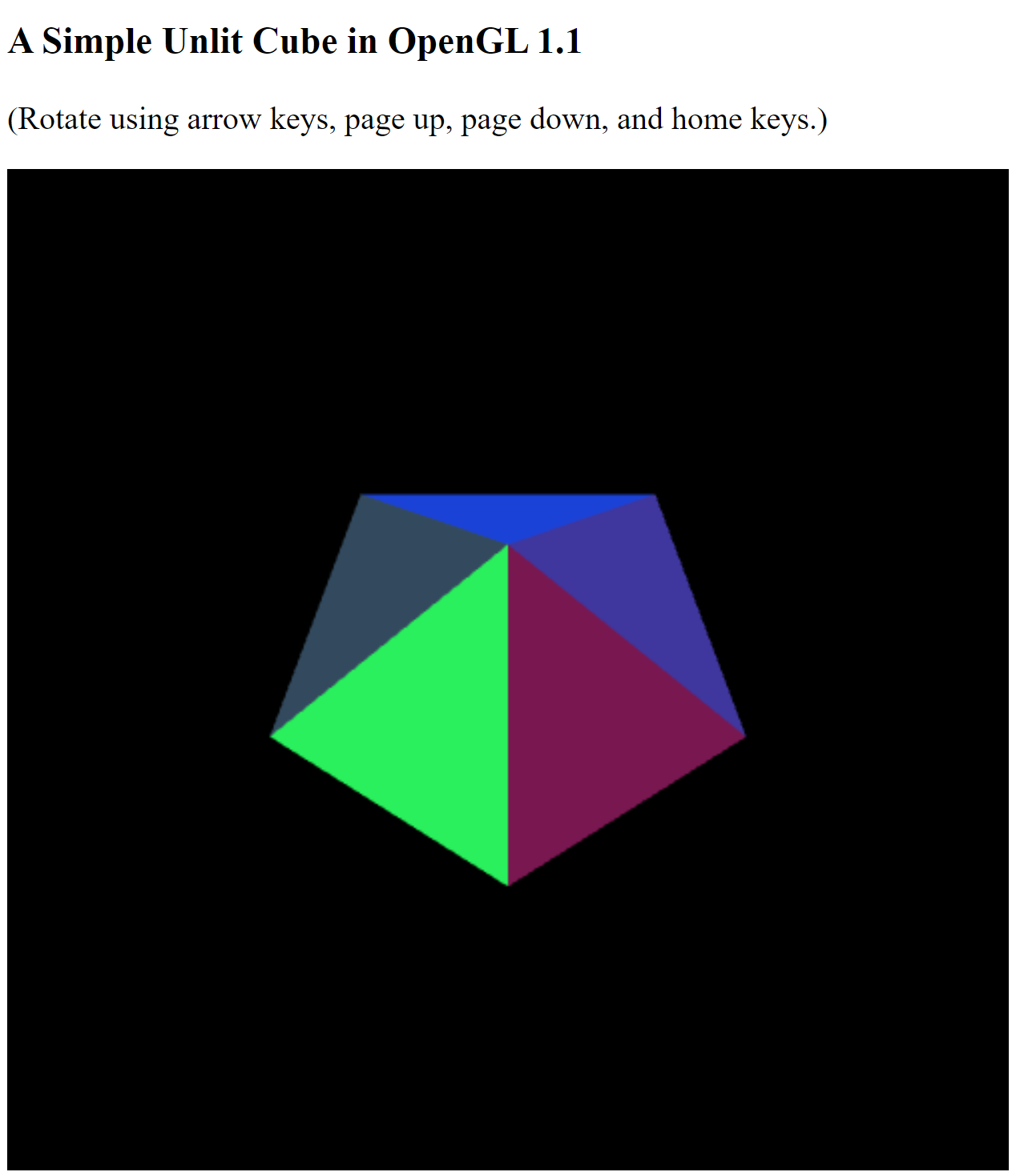
Link do zdalnego repozytorium: https://github.com/Slayzerus/UBB\_GrafikaKomputerowa/tree/main/Lab%206

**4. Wynik działania:**

Obiekt 1:



Obiekt 2:



**5. Wnioski:**

W zadaniu dotyczącym geometrii 3D przy użyciu OpenGL, głównym celem było zastosowanie podstawowych prymitywów i transformacji do stworzenia złożonych obiektów. OpenGL używa domyślnego układu współrzędnych z zakresem wartości od -1 do 1, gdzie dodatni kierunek osi z jest prostopadły do obrazu. Biblioteka ta pozwala na rysowanie prostych kształtów, takich jak punkty, linie i trójkąty, definiowanych przez wierzchołki. Kolory w OpenGL ustawia się za pomocą funkcji glColor\*, z możliwością dodania przezroczystości przy użyciu komponentu alfa. Test głębi jest kluczowy dla poprawnego renderowania scen 3D, umożliwiając określenie, które obiekty są widoczne, a które ukryte. Transformacje 3D, takie jak skalowanie, translacja i rotacja, są stosowane za pomocą macierzy transformacji, które można zarządzać stosując funkcje glPushMatrix i glPopMatrix. Modelowanie hierarchiczne pozwala na definiowanie obiektów w lokalnych układach współrzędnych i umieszczanie ich w globalnym układzie za pomocą transformacji. Zadanie polegało na stworzeniu dwóch obiektów: korkociągu wokół jednej z osi z punktami stopniowo powiększanymi oraz piramidy zbudowanej z wachlarzy trójkątów. Kluczowe było zastosowanie bufora głębi oraz odpowiednie zarządzanie stanem rysowania, aby uzyskać poprawnie renderowane obiekty 3D. Poprzez praktyczne zastosowanie tych koncepcji, zadanie umożliwiło głębsze zrozumienie geometrii 3D w OpenGL.