SPRAWOZDANIE

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

Laboratorium II
Data
Temat: Zadanie_OpenGL1

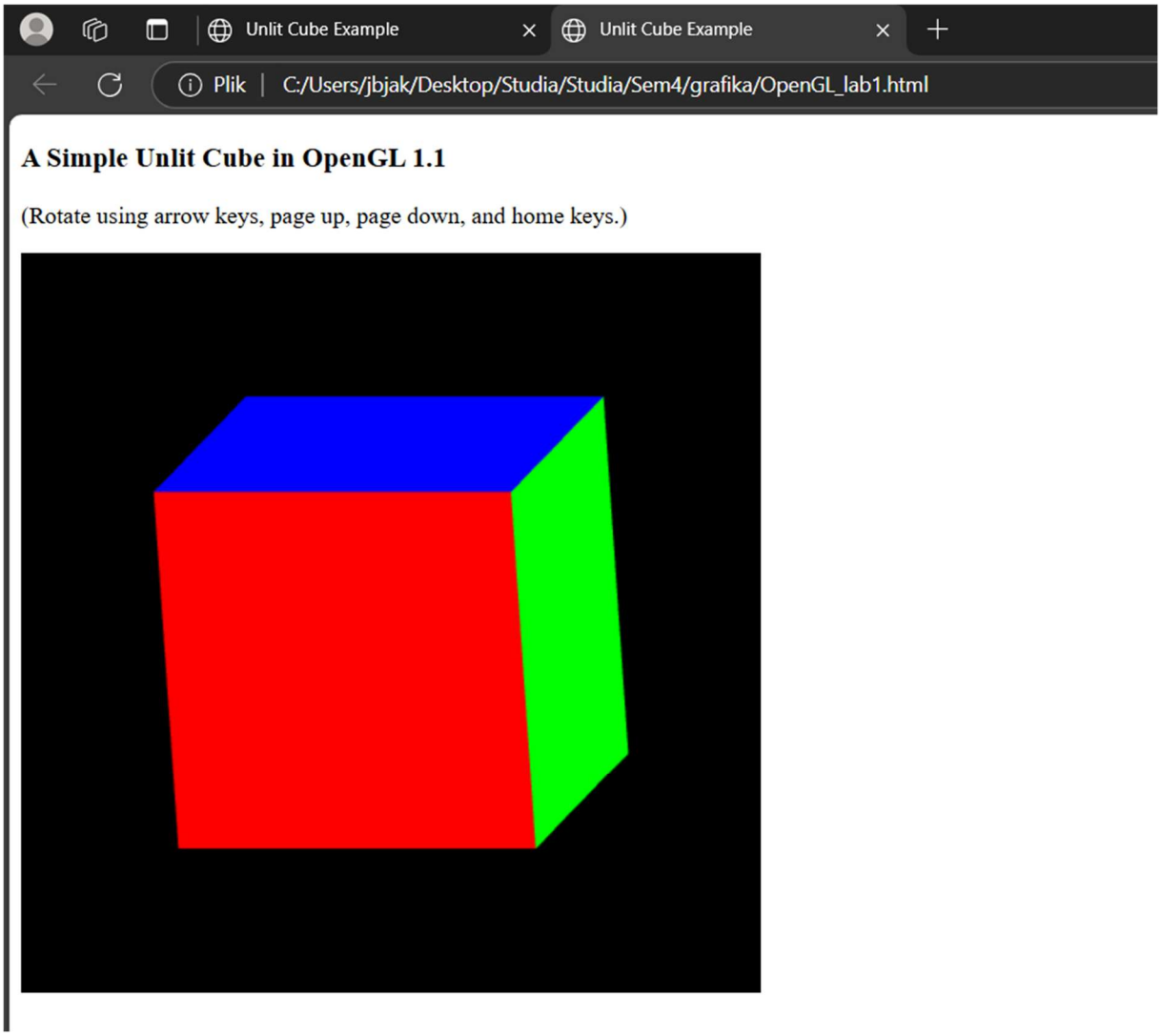
Wariant 8 + 4

Jakub Bąk
Informatyka I stopień,
stacjonarne,
4 semestr,
Gr.3b

1. Polecenie

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.
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, 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. Wprowadzane dane:



Na podstawie podanych plików należało wykonać zadanie z polecenia

3. Wykorzystane komendy:

Link do github: https://github.com/Szeladin/grafika.git

Kod Programu:

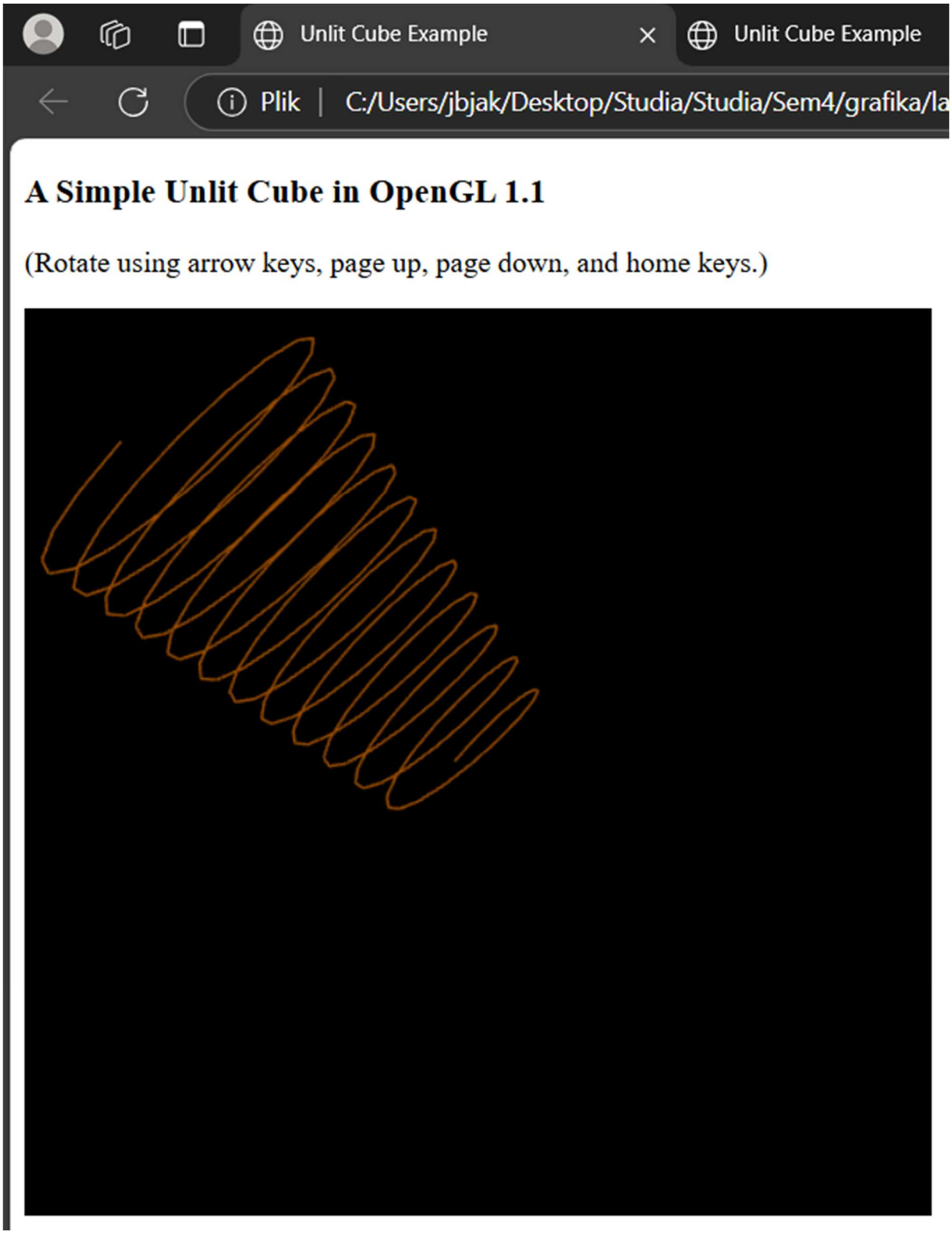
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Unlit Cube Example</title>
<script src="glsim.js"></script>
<script>
    let rotateX = 15;
    let rotateY = -15;
    let rotateZ = 0;
    let corkscrewColor = [0, 1, 0];

     function triangle(r,g,b){
         glColor3f(r,g,b);
         glBegin(GL_TRIANGLE_FAN);
         glVertex3f(0, 0, z);
         const egdes = 10;
         for (let i = 0; i < 2; i++) {
             let currDeg = (((Math.PI * 2) / egdes) * i);
             glVertex3f(Math.cos(currDeg), Math.sin(currDeg), 0);
         }
         glEnd();
     }

     function corkscrew(turns, radius, height, axis, color) {
        glColor3f(color[0], color[1], color[2]);
        glBegin(GL_LINE_STRIP);
        const points = 200;
        for (let i = 0; i <= points; i++) {
            let angle = (i / points) * (Math.PI * 2 * turns);
            let scale = 1 + (i / points);
            let x = Math.cos(angle) * radius * scale;
            let y = Math.sin(angle) * radius * scale;
            let z = height * (i / points);
            if (axis === 'x') {
                glVertex3f(z, x, y);
            } else if (axis === 'y') {
                glVertex3f(x, z, y);
            } else {
                glVertex3f(x, y, z);
            }
        }
        glEnd();
    }

     function pyramid(size) {
         glPushMatrix();
         glScalef(size, size, size);
         const sides = 12;
         let angleStep = (Math.PI * 2) / sides;
         const colors = [
             [1, 0, 0], [0, 1, 0], [0, 0, 1], [1, 1, 0], [1, 0, 1], [0, 1, 1],
             [0.5, 0.5, 0], [0.5, 0, 0.5], [0, 0.5, 0.5], [0.5, 0.5, 0.5],
             [1, 0.5, 0], [0, 1, 0.5]
         ];
         glBegin(GL_TRIANGLES);
```

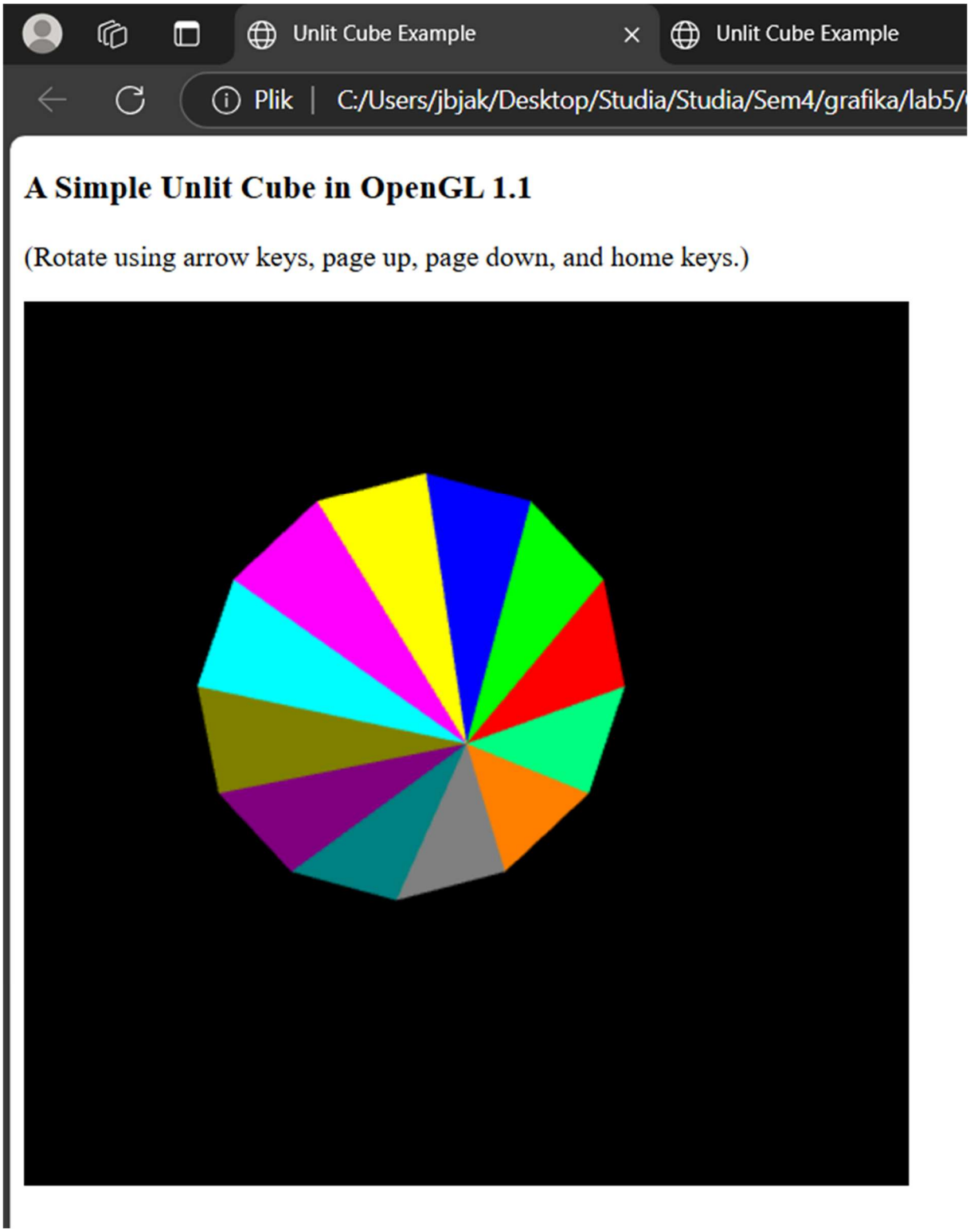
```
        for (let i = 0; i < sides; i++) {
            let angle1 = i * angleStep;
            let angle2 = (i + 1) * angleStep;
            let color = colors[i % colors.length];
            glColor3f(color[0], color[1], color[2]);
            glVertex3f(0, 0, 1);
            glVertex3f(Math.cos(angle1), Math.sin(angle1), 0);
            glVertex3f(Math.cos(angle2), Math.sin(angle2), 0);
        }
        glEnd();
        glColor3f(0.5, 0.25, 0);
        glBegin(GL_POLYGON);
        for (let i = 0; i < sides; i++) {
            let angle = i * angleStep;
            glVertex3f(Math.cos(angle), Math.sin(angle), 0);
        }
        glEnd();
        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);
        if (drawMode === "corkscrew") {
           corkscrew(12, 0.2, 1, 'z', corkscrewColor);
        } else if (drawMode === "pyramid") {
            glPushMatrix();
            glTranslatef(0, 0, -0.5);
            pyramid(0.5);
            glPopMatrix();
        }
    }

    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) {
            drawMode = "corkscrew";
        } else if (key == 50) {
            drawMode = "pyramid";
        } else if (key == 90) {
            corkscrewColor = [0, 1, 0];
        } else if (key == 66) {
            corkscrewColor = [0.6, 0.3, 0];
        } else if (key == 78) {
            corkscrewColor = [0, 0, 1];
        } else {
            if ( key == 37 )
            rotateY -= 15;
            else if ( key == 39 )
            rotateY += 15;
            else if ( key == 40)
            rotateX += 15;
            else if ( key == 38 )
            rotateX -= 15;
            else if ( key == 33 )
            rotateZ += 15;
            else if ( key == 34 )
            rotateZ -= 15;
            else if ( key == 36 )
```

```
            rotateX = rotateY = rotateZ = 0;
            if (key >=34 && key <= 40) {
                evt.preventDefault();
            }
        }
        display();
    }

    function init() {
        try {
            glsimUse("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>
```

Program rysuje hierarchiczny model wiatraka w SVG, składający się z prostokąta, trójkąta i obracających się dwunastokątów. Wiatraki są skalowane i rozmieszczane w różnych pozycjach za pomocą transformacji. Animacja obracania dwunastokątów jest realizowana za pomocą elementu <animateTransform>.

4. Wyniki działania





Program rysuje obiekty 3D w **WebGL**, takie jak piramida i spiralny kształt (corkscrew), które można obracać za pomocą klawiatury. Użytkownik może zmieniać tryb rysowania i kolor spirali, a animacja jest w HTML `<canvas>`.

5. Wnioski:

Podsumowując, OpenGL oferuje podstawowe narzędzia do tworzenia geometrii 3D, ale wymaga dodatkowych technik do renderowania bardziej złożonych kształtów i efektów wizualnych.