**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

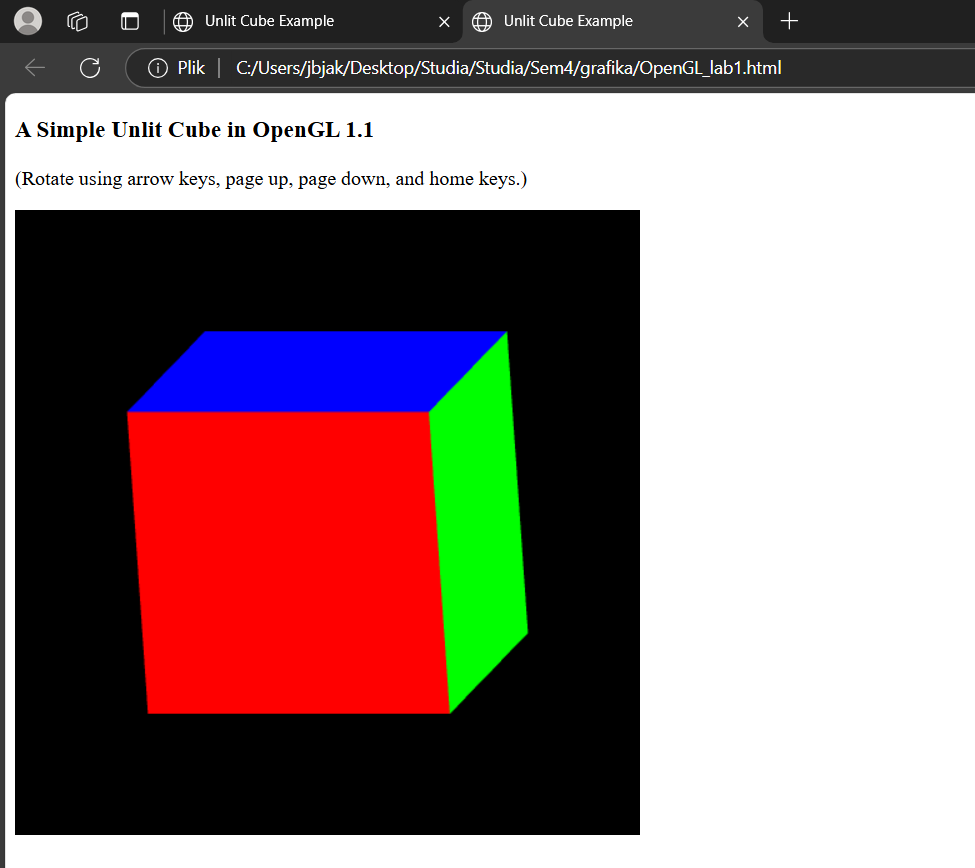
# Polecenie

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.

# Wprowadzane dane:

****

Na podstawie podanych plików należało wykonać zadanie z polecenia

# Wykorzystane komendy:

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

Kod Programu:

1. <!DOCTYPE html>

2. <html>

3. <head>

4. <meta charset="UTF-8">

5. <title>Unlit Cube Example</title>

6. <script src="glsim.js"></script>

7. <script>

8.     let rotateX = 15;

9.     let rotateY = -15;

10.     let rotateZ = 0;

11.     let corkscrewColor = [0, 1, 0];

12.

13.      function triangle(r,g,b){

14.          glColor3f(r,g,b);

15.          glBegin(GL\_TRIANGLE\_FAN);

16.          glVertex3f(0, 0, z);

17.          const egdes = 10;

18.          for (let i = 0; i < 2; i++) {

19.              let currDeg = (((Math.PI \* 2) / egdes) \* i);

20.              glVertex3f(Math.cos(currDeg), Math.sin(currDeg), 0);

21.          }

22.          glEnd();

23.      }

24.

25.      function corkscrew(turns, radius, height, axis, color) {

26.         glColor3f(color[0], color[1], color[2]);

27.         glBegin(GL\_LINE\_STRIP);

28.         const points = 200;

29.         for (let i = 0; i <= points; i++) {

30.             let angle = (i / points) \* (Math.PI \* 2 \* turns);

31.             let scale = 1 + (i / points);

32.             let x = Math.cos(angle) \* radius \* scale;

33.             let y = Math.sin(angle) \* radius \* scale;

34.             let z = height \* (i / points);

35.             if (axis === 'x') {

36.                 glVertex3f(z, x, y);

37.             } else if (axis === 'y') {

38.                 glVertex3f(x, z, y);

39.             } else {

40.                 glVertex3f(x, y, z);

41.             }

42.         }

43.         glEnd();

44.     }

45.

46.      function pyramid(size) {

47.          glPushMatrix();

48.          glScalef(size, size, size);

49.          const sides = 12;

50.          let angleStep = (Math.PI \* 2) / sides;

51.          const colors = [

52.              [1, 0, 0], [0, 1, 0], [0, 0, 1], [1, 1, 0], [1, 0, 1], [0, 1, 1],

53.              [0.5, 0.5, 0], [0.5, 0, 0.5], [0, 0.5, 0.5], [0.5, 0.5, 0.5],

54.              [1, 0.5, 0], [0, 1, 0.5]

55.          ];

56.          glBegin(GL\_TRIANGLES);

57.          for (let i = 0; i < sides; i++) {

58.              let angle1 = i \* angleStep;

59.              let angle2 = (i + 1) \* angleStep;

60.              let color = colors[i % colors.length];

61.              glColor3f(color[0], color[1], color[2]);

62.              glVertex3f(0, 0, 1);

63.              glVertex3f(Math.cos(angle1), Math.sin(angle1), 0);

64.              glVertex3f(Math.cos(angle2), Math.sin(angle2), 0);

65.          }

66.          glEnd();

67.          glColor3f(0.5, 0.25, 0);

68.          glBegin(GL\_POLYGON);

69.          for (let i = 0; i < sides; i++) {

70.              let angle = i \* angleStep;

71.              glVertex3f(Math.cos(angle), Math.sin(angle), 0);

72.          }

73.          glEnd();

74.          glPopMatrix();

75.      }

76.

77.      function display() {

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

79.          glLoadIdentity();

80.          glRotatef(rotateZ,0,0,1);

81.          glRotatef(rotateY,0,1,0);

82.          glRotatef(rotateX,1,0,0);

83.          if (drawMode === "corkscrew") {

84.             corkscrew(12, 0.2, 1, 'z', corkscrewColor);

85.          } else if (drawMode === "pyramid") {

86.              glPushMatrix();

87.              glTranslatef(0, 0, -0.5);

88.              pyramid(0.5);

89.              glPopMatrix();

90.          }

91.      }

92.

93.     function initGL() {

94.         glMatrixMode(GL\_PROJECTION);

95.         glOrtho(-1, 1, -1, 1, -1, 1);

96.         glMatrixMode(GL\_MODELVIEW);

97.         glEnable(GL\_DEPTH\_TEST);

98.         glClearColor(0, 0, 0, 1);

99.     }

100.

101.     function doKeyDown(evt) {

102.         let key = evt.keyCode;

103.         if (key == 49) {

104.             drawMode = "corkscrew";

105.         } else if (key == 50) {

106.             drawMode = "pyramid";

107.         } else if (key == 90) {

108.             corkscrewColor = [0, 1, 0];

109.         } else if (key == 66) {

110.             corkscrewColor = [0.6, 0.3, 0];

111.         } else if (key == 78) {

112.             corkscrewColor = [0, 0, 1];

113.         } else {

114.             if ( key == 37 )

115.             rotateY -= 15;

116.             else if ( key == 39 )

117.             rotateY += 15;

118.             else if ( key == 40)

119.             rotateX += 15;

120.             else if ( key == 38 )

121.             rotateX -= 15;

122.             else if ( key == 33 )

123.             rotateZ += 15;

124.             else if ( key == 34 )

125.             rotateZ -= 15;

126.             else if ( key == 36 )

127.             rotateX = rotateY = rotateZ = 0;

128.             if (key >=34 && key <= 40) {

129.                 evt.preventDefault();

130.             }

131.         }

132.         display();

133.     }

134.

135.     function init() {

136.         try {

137.             glsimUse("glcanvas");

138.         }

139.         catch (e) {

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

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

142.             return;

143.         }

144.         document.onkeydown = doKeyDown;

145.         initGL();

146.         display();

147.     }

148.

149. </script>

150. </head>

151. <body onload="init()">

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

153. <p>(Rotate using arrow keys, page up, page down, and home keys.)</p>

154. <noscript>

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

156. </noscript>

157. <div id="canvas-holder">

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

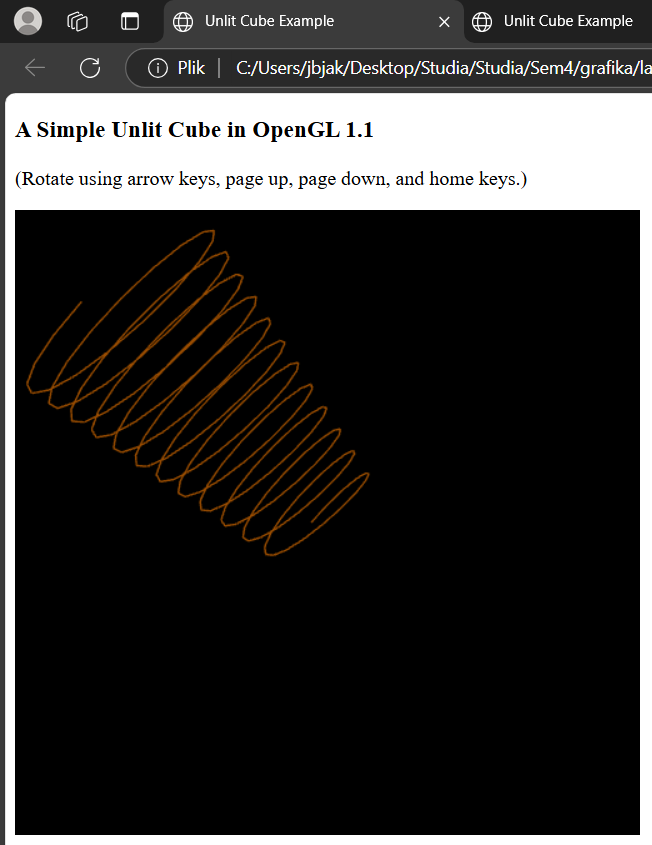
159. </div>

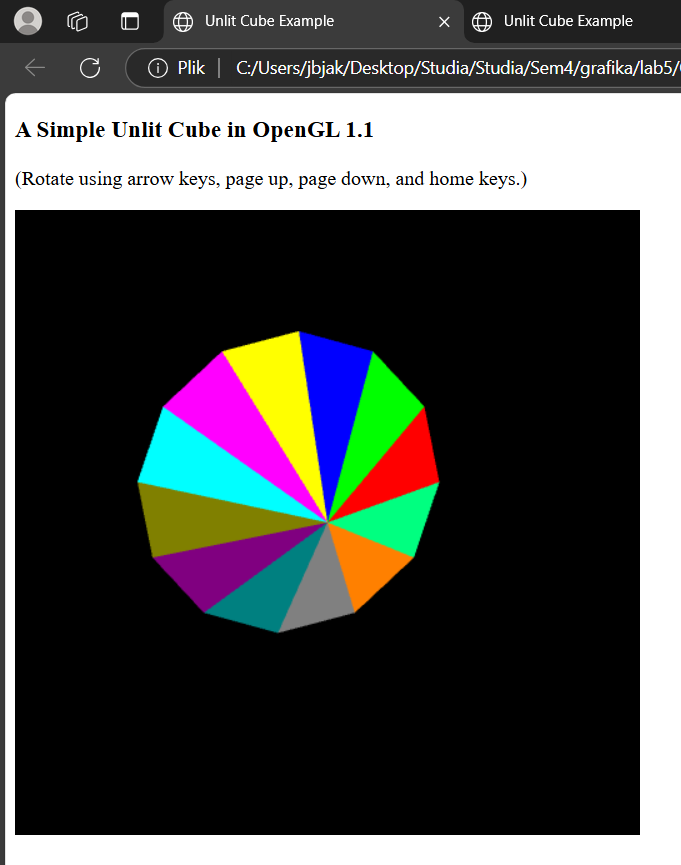
160. </body>

161. </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>.

# 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>`.

# Wnioski:

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