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

Laboratorium 9
29 IV 2021 r.
Temat: „Konstruowanie obiektów z użyciem Three.js”
Wariant:
Figura: 11

Robert Laszczak
Informatyka I stopień
Stacjonarne, 4 semestr
Grupa 2B

1. Polecenie

Celem jest konstruowanie modelu figury szachowej zgodnie z wariantem zadania (patrz rysunek) używając three.js w oparciu na omówione na zajęcie metody konstruowania obiektów

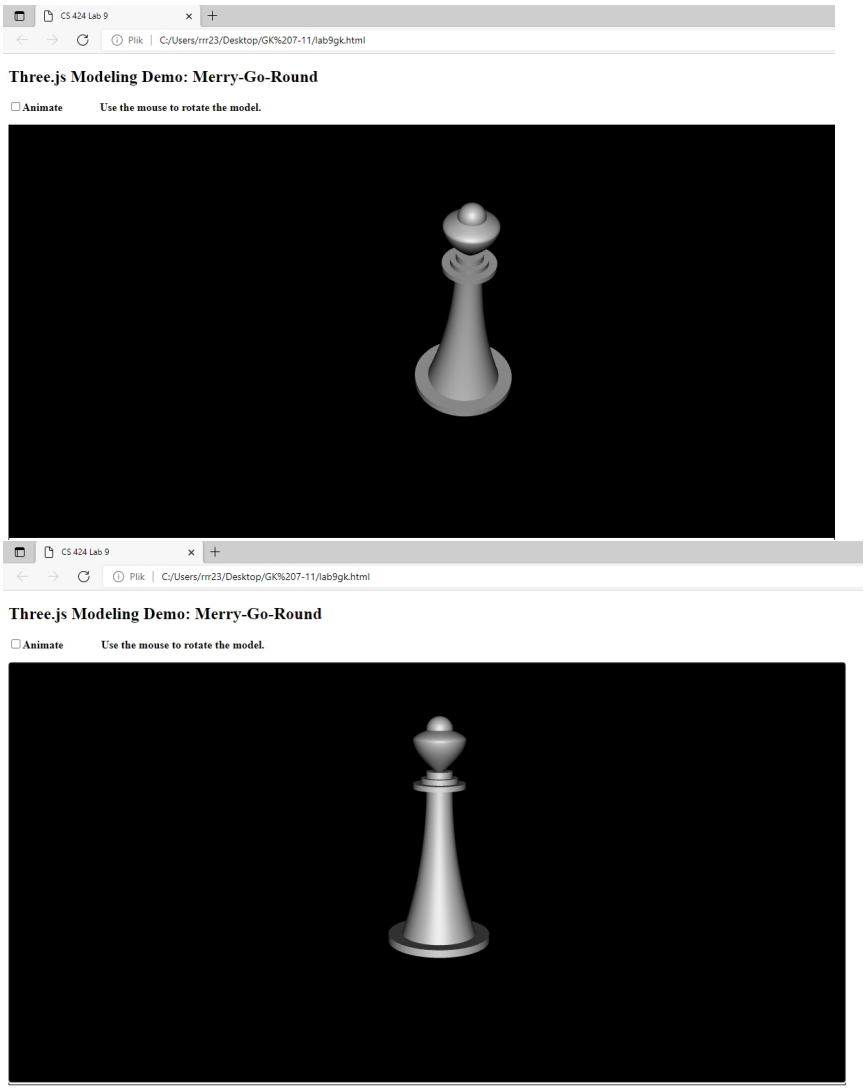
2. Wprowadzam dane:

Numer figury - 11

3. Wykorzystane komendy:

Link do zdalnego repozytorium:
https://github.com/RLaszczak/Lab_GK

4.Wyniki działania:



5. Wnioski

Na podstawie otrzymanego wyniku można stwierdzić, że:

- Możemy sterować oświetleniem, odbiciami światła i innymi jego aspektami.
- Możemy łatwo tworzyć trójwymiarowe obiekty graficzne i obracać je.

Kod:

```
<!DOCTYPE html>



<head>

    <meta charset="UTF-8">
    <title>CS 424 Lab 9</title>
    <script src="https://cdn.jsdelivr.net/npm/three@0.115/build/three.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/three@0.115/examples/js/controls/OrbitControls.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/three@0.115/examples/js/loaders/GLTFLoader.js"></script>
    <script>
        "use strict";
        var canvas, renderer, scene, camera; // Standard three.js requirements.
        var controls; // An OrbitControls object that is used to implement
        // rotation of the scene using the mouse.  (It actually rotates
        // the camera around the scene.)
        var animating = false; // Set to true when an animation is in progress.
        var frameNumber = 0; // Frame number is advanced by 1 for each frame while animating.
        var tempObject; // A temporary animated object.  DELETE IT.

        var n = 14;
        var R = 11;
        var horses = new THREE.Group();
        var pole;
        var roof;
        var pivot1 = new THREE.Group();
        /**
         *  The render function draws the scene.
         */
        function render() {
            renderer.render(scene, camera);
        }
        /**
         * This function is called by the init() method to create the world.
         */
        function createWorld() {
            renderer.setClearColor("black"); // Background color for scene.
            scene = new THREE.Scene();
            // -------------------- Make a camera with viewpoint light ----------------------
            camera = new THREE.PerspectiveCamera(30, canvas.width / canvas.height, 0.1, 100);
            camera.position.z = 30;
            var light; // A light shining from the direction of the camera; moves with the camera.
            light = new THREE.DirectionalLight();
            light.position.set(0, 0, 1);
            camera.add(light);
            scene.add(camera);
```

```
//-------------------- Create the scene's visible objects ----------------------
var material = new THREE.MeshPhongMaterial({
    color: "silver",
    shininess: 20,
    reflectivity: 10,
    shading: THREE.FlatShading
});
pole = new THREE.Mesh(
    new THREE.SphereGeometry(0.5, 100, 100, 0, Math.PI),
    material
);
pole.position.y = 5.38;
                                pole.rotation.x = -Math.PI/2;
scene.add(pole);
var head1 = new THREE.Curve();
head1.getPoint = function(t) {
    return new THREE.Vector2(Math.cos(t * 1.5), t / 2);
}

var head2 = new THREE.Curve();
head2.getPoint = function(t) {
    return new THREE.Vector2(Math.sin(-t * 1.5), t * 1.3);
}
var pointsHead1 = head1.getPoints(256);
var geometryHead1 = new THREE.LatheGeometry(pointsHead1, 150);
var pawnHead1 = new THREE.Mesh(geometryHead1, material);
pawnHead1.position.y = 5;
scene.add(pawnHead1);
var pointsHead2 = head2.getPoints(256);
var geometryHead2 = new THREE.LatheGeometry(pointsHead2, 150);
var pawnHead2 = new THREE.Mesh(geometryHead2, material);
pawnHead2.position.y = 3.7;
scene.add(pawnHead2);
var geometry = new THREE.CylinderGeometry(0.5, 0.5, 0.3, 100);
geometry.computeVertexNormals();
var cylinder = new THREE.Mesh(geometry, material);
cylinder.position.y = 3.7;
scene.add(cylinder);
var geometry = new THREE.CylinderGeometry(0.7, 0.7, 0.2, 100);
geometry.computeVertexNormals();
var cylinder = new THREE.Mesh(geometry, material);
cylinder.position.y = 3.5;
scene.add(cylinder);
var geometry = new THREE.CylinderGeometry(1, 1, 0.2, 100);
geometry.computeVertexNormals();
var cylinder = new THREE.Mesh(geometry, material);
cylinder.position.y = 3.3;
scene.add(cylinder);
var body = new THREE.Curve();
body.getPoint = function(t) {
    return new THREE.Vector2(t * t + 0.5, t * 6);
}
```

```
    var points = body.getPoints(256);
    var geometry = new THREE.LatheGeometry(points, 150);
    var pawnBody = new THREE.Mesh(geometry, material);
    pawnBody.scale.set(1, -1, 1);
    pawnBody.position.y = 3.3;
    scene.add(pawnBody);
    var geometry = new THREE.CylinderGeometry(2, 2, 0.3, 100);
    geometry.computeVertexNormals();
    var cylinder = new THREE.Mesh(geometry, material);
    cylinder.position.y = -2.7;
    scene.add(cylinder);
} // end function createWorld()
/**
 *  This function is called once for each frame of the animation, before
 *  the render() function is called for that frame.  It updates any
 *  animated properties.  The value of the global variable frameNumber
 *  is incrementd 1 before this function is called.
 */
function updateForFrame() {
    // Update size and rotation of tempObject.  DELETE THIS!
    var loopFrame = frameNumber % 240;
    if (loopFrame > 120) {
        loopFrame = 240 - loopFrame;
    }
    var scaleFactor = 1 + loopFrame / 120;
    tempObject.scale.set(scaleFactor, scaleFactor, scaleFactor);
    tempObject.rotation.y += 0.01;
}
/* ---------------------------- MOUSE AND ANIMATION SUPPORT ------------------
/**
 *  This page uses THREE.OrbitControls to let the user use the mouse to rotate
 *  the view.  OrbitControls are designed to be used during an animation, where
 *  the rotation is updated as part of preparing for the next frame.  The scene
 *  is not automatically updated just because the user drags the mouse.  To get
 *  the rotation to work without animation, I add another mouse listener to the
 *  canvas, just to call the render() function when the user drags the mouse.
 *  The same thing holds for touch events -- I call render for any mouse move
 *  event with one touch.
 */
function installOrbitControls() {
    controls = new THREE.OrbitControls(camera, canvas);
    controls.noPan = true;
    controls.noZoom = true;
    controls.staticMoving = true;
    function move() {
        controls.update();
        if (!animating) {
            render();
        }
    }
    function down() {
        document.addEventListener("mousemove", move, false);
```

```
    }
    function up() {
        document.removeEventListener("mousemove", move, false);
    }
    function touch(event) {
        if (event.touches.length == 1) {
            move();
        }
    }
    canvas.addEventListener("mousedown", down, false);
    canvas.addEventListener("touchmove", touch, false);
}
/*  Called when user changes setting of the Animate checkbox. */
function doAnimateCheckbox() {
    var run = document.getElementById("animateCheckbox").checked;
    if (run != animating) {
        animating = run;
        if (animating) {
            requestAnimationFrame(doFrame);
        }
    }
}
/*  Drives the animation, called by system through requestAnimationFrame() */
function doFrame() {
    if (animating) {
        frameNumber++;
        updateForFrame();
        render();
        requestAnimationFrame(doFrame);
    }
}
/*----------------------------- INITIALIZATION ----------------------------------------
/**
 *  This function is called by the onload event so it will run after the
 *  page has loaded.  It creates the renderer, canvas, and scene objects,
 *  calls createWorld() to add objects to the scene, and renders the
 *  initial view of the scene.  If an error occurs, it is reported.
 */
function init() {
    try {
        canvas = document.getElementById("glcanvas");
        renderer = new THREE.WebGLRenderer({
            canvas: canvas,
            antialias: true,
            alpha: false
        });
    } catch (e) {
        document.getElementById("message").innerHTML = "<b>Sorry, an error occurred:<br>" +
            e + "</b>";
        return;
    }
    document.getElementById("animateCheckbox").checked = false;
```

```
        document.getElementById("animateCheckbox").onchange = doAnimateCheckbox;
        createWorld();
        installOrbitControls();
        render();
      }
  </script>
</head>
<body onload="init()">
  <h2>Three.js Modeling Demo: Merry-Go-Round</h2>
  <noscript>
 <p style="color: #AA0000; font-weight: bold">Sorry, but this page requires JavaScript!</p>
</noscript>
  <p style="color:#AA0000; font-weight: bold" id="message">
  </p>
  <p>
    <label><input type="checkbox" id="animateCheckbox"><b>Animate</b></label>
    <b style="margin-left:50px">Use the mouse to rotate the model.</b>
  </p>
  <div id="canvas-holder" style="float:left; border: thin solid black; background-color: white">
    <canvas width=1200 height=600 id="glcanvas"></canvas>
  </div>
</body>
</html>
```