**SPRAWOZDANIE**

Zajęcia: Grafika komputerowa

Prowadzący: prof. dr hab. Vasyl Martsenyuk

# **Laboratorium**

Data: 29.04.2024

**Temat:**

"Podstawy Three.js"

# **Wariant:**

# Jedenastokąt

Mateusz Żelazo

Informatyka I stopnia

stacjonarne, 4 semestr

Gr.3a

### **Zadanie nr. 1**

1. **Polecenie:**

Celem jest konstruowanie złożonego modelu za pomocą three.js - animowanej karuzeli (podstawa karuzeli jest jedenastokąt) i co najmniej jednego innego wybranego modelu

1. **Wprowadzane dane:**

**earth = new THREE.Mesh(**

**new THREE.SphereGeometry(1.5, 16, 8),**

**new THREE.MeshLambertMaterial({**

**color: "white",**

**map: makeTexture("resources/earth.jpg")**

**})**

**);**

**earth.position.y = 1.5**

**dach = new THREE.Mesh(**

**new THREE.CylinderGeometry(0, 4, 1, 11, 1),**

**new THREE.MeshPhongMaterial({**

**color: 0x66BBFF,**

**specular: 0x222222,**

**shininess: 16,**

**shading: THREE.FlatShading**

**})**

**);**

***// porecz.position.y = 1.5***

**dach.position.y = 3.5**

**podstawa = new THREE.Mesh(**

**new THREE.CylinderGeometry(4, 4, 0.1, 11, 1),**

**new THREE.MeshPhongMaterial({**

**color: 0x66BBFF,**

**specular: 0x222222,**

**shininess: 16,**

**shading: THREE.FlatShading**

**})**

**);**

**podstawa.rotation.y = Math.PI / 12;**

**podstawa.position.y = -1.5**

**scene.add(podstawa);**

**podstawa.add(earth);**

**podstawa.add(dach);**

**let w = 11;**

**let r = 3.5;**

**let kat = 360 / w;**

**for (let i = 1; i <= w; i++) {**

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

**let x = Math.sin(rad) \* r;**

**let y = Math.cos(rad) \* r;**

**porecz[i] = new THREE.Mesh(**

**new THREE.CylinderGeometry(0.1, 0.1, 3, 99, 1),**

**new THREE.MeshLambertMaterial({**

**color: "white",**

**map: makeTexture("resources/paisley.jpg")**

**})**

**);**

***// porecz.position.y = 1.5***

***// porecz[i].position.y = i \* 3***

**porecz[i].position.x = x**

**porecz[i].position.y = 1.5**

**porecz[i].position.z = y**

**podstawa.add(porecz[i]);**

**let teapotGeom = new THREE.BufferGeometry();**

**teapotGeom.addAttribute("position", new THREE.BufferAttribute(teapotModel.vertexPositions, 3));**

**teapotGeom.addAttribute("normal", new THREE.BufferAttribute(teapotModel.vertexNormals, 3));**

**teapotGeom.setIndex(new THREE.BufferAttribute(teapotModel.indices, 1));**

**let teapotMaterial = new THREE.MeshPhongMaterial({**

**color: new THREE.Color(0.780392, 0.568627, 0.113725),**

**specular: 0x171615,**

**shininess: 28,**

**side: THREE.DoubleSide *// required since some parts of the inside are visible***

**});**

**let czajnik = new THREE.Mesh(teapotGeom, teapotMaterial);**

**czajnik.scale.set(0.05, 0.05, 0.05);**

**porecz[i].add(czajnik);**

1. **Wykorzystane komendy:**
   1. **kod źródłowy**

**<!DOCTYPE html>**

**<head>**

**<meta charset="UTF-8">**

**<script src="build/three.js"></script>**

**<script src="resources/OrbitControls.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.***

***/\*\****

***\* 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 ----------------------***

**tempObject = new THREE.Mesh( *// DELETE THIS !***

**new THREE.CylinderGeometry(2,4,8,6,1),**

**new THREE.MeshPhongMaterial({**

**color: 0x66BBFF,**

**specular: 0x222222,**

**shininess: 16,**

**shading: THREE.FlatShading**

**})**

**);**

**tempObject.rotation.y = Math.PI/12;**

**scene.add(tempObject);**

**} *// 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">**

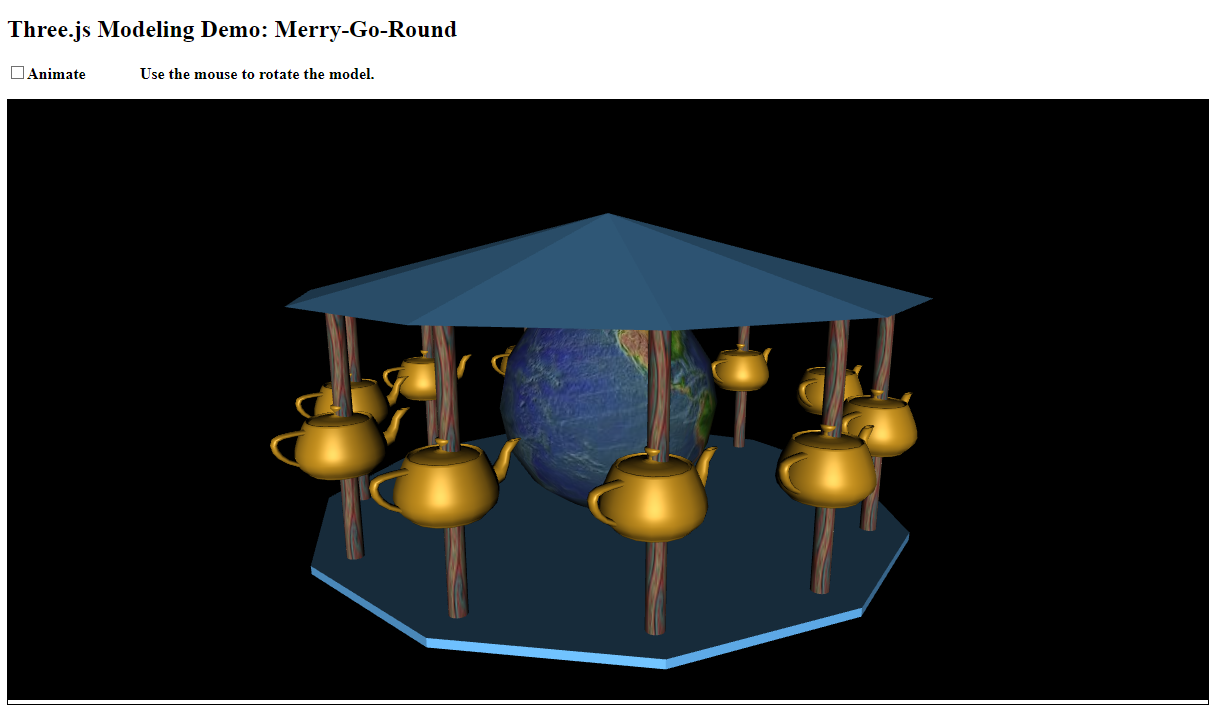
**<canvas width=1200 height=600 id="glcanvas"></canvas>**

**</div>**

**</body>**

**</html>**

1. **Link do zdalnego repozytorium:** 
   1. <https://github.com/Terminalk/GKLab>
2. **Wynik działania:** 
   1. **Wynik działania**

****

1. **Wnioski:**

Za pomocą tree.js jesteśmy w stanie zaimplementować na stronie internetowej obiekty 3d oraz niemal je edytować / przekształcać. Tree.js jest bardzo dobrym oraz użytecznym rozszerzeniem możliwość JavaScriptu

### 

### 

### 