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

Laboratorium 8
22 IV 2021 r.
Temat: „Podstawy Three.js”
Wariant:
Liczba kątów:5

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

1. Polecenie

Celem jest konstruowanie złożonego modelu za pomocą three.js - animowanej karuzeli (podstawa karuzeli jest wielokątem odpowiednio z konfiguracją zadania) i co najmniej jednego innego wybranego modelu (patrz Fig.). Pliki do pobrania znajdują się poniżej. Głównym plikiem jest lab9.html. Podfolder zasobów resources zawiera dwa pliki JavaScript używane przez program oraz model konia, którego używamy w karuzeli.
Zawiera również kilka plików graficznych, które można wykorzystać jako tekstury...

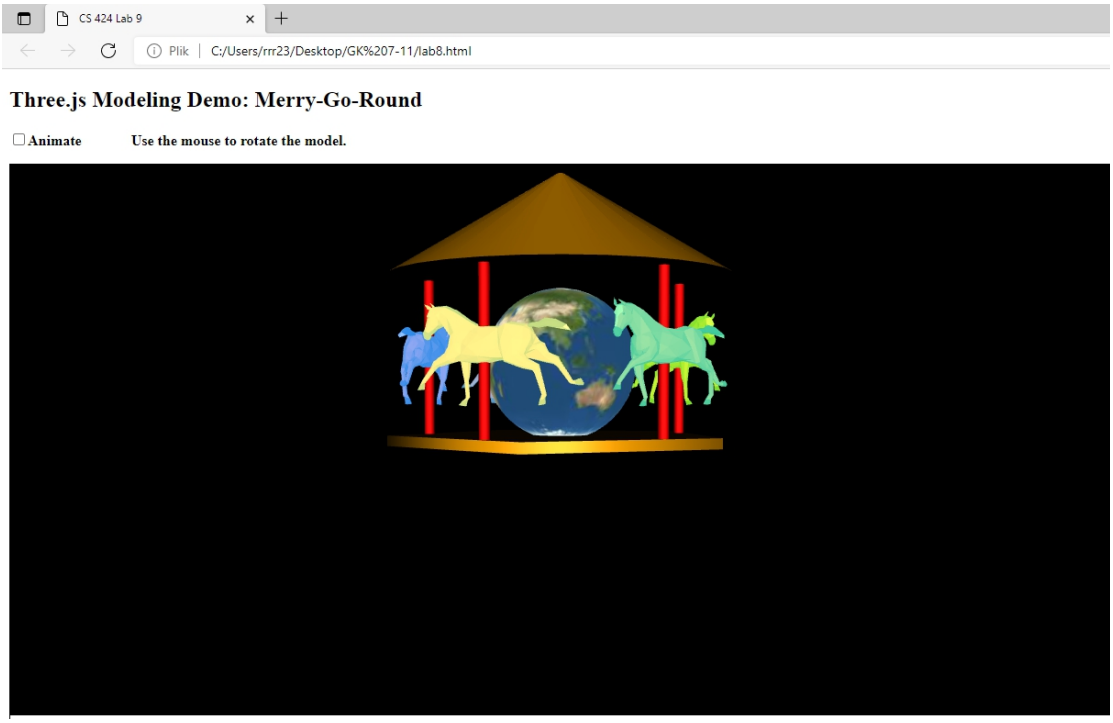
2. Wprowadzam dane:

Liczba kątów n = 5

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 łatwo tworzyć trójwymiarowe obiekty graficzne i obracać je.
- Możemy sterować oświetleniem.

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 = true; // 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 ----------------------
   roof = new THREE.Mesh(
      new THREE.CylinderGeometry(0.1, n, 3, 30, 1),
      new THREE.MeshPhongMaterial({
         color: "orange",
      })
   );
   roof.position.y = 6.2;
   scene.add(roof);
   var geometry = new THREE.SphereGeometry(2.2, 25, 25);
   var material = new THREE.MeshBasicMaterial({
      map: new THREE.TextureLoader().load('https://encrypted-
tbn0.gstatic.com/images?q=tbn%3AANd9GcS9Uv3AxsQE88_-z0Njxqephw8NMTi8odMGReBqPH-m9-fsfjde&usqp=CAU')
   });
   var sphere = new THREE.Mesh(geometry, material);
   sphere.position.y = 2.2;
   scene.add(sphere);
   var floor = new THREE.Mesh(
      new THREE.CylinderGeometry(14, 14, 0.3, 14, 1),
      new THREE.MeshPhongMaterial({
         color: "orange"
      })
   );
   floor.rotation.y = Math.PI / n;
   scene.add(floor);
   var newMaterial;
   var loader = new THREE.GLTFLoader();
   for (let i = 0; i < n; i++) {
      horses.add(loader.load('https://threejs.org/examples/models/gltf/Horse.glb', function(horse) {
         let hScene = horse.scene;
         hScene.scale.multiplyScalar(0.015);
         hScene.position.x = R * Math.cos((2 * Math.PI * i) / n);
         hScene.position.y = 1;
         hScene.position.z = R * Math.sin((2 * Math.PI * i) / n);
         hScene.rotation.y = -(2 * Math.PI * i) / n;
         hScene.traverse((o) => {
            if (o.isMesh) o.material.emissive = new THREE.Color("#" + Math.floor(Math.random() * 16777215).toString(16));
         });
         scene.add(hScene);
      }));
      pole = new THREE.Mesh(
         new THREE.CylinderGeometry(0.15, 0.15, 5, 30, 1),
         new THREE.MeshPhongMaterial({
            color: "red",
            shininess: 8
```

```
        })
      );
      pole.position.x = R * Math.cos((2 * Math.PI * i) / n);
      pole.position.y = 2.5;
      pole.position.z = R * Math.sin((2 * Math.PI * i) / n);
      scene.add(pole);
    }
} // 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>
```