**SPRAWOZDANIE**

Zajęcia: Grafika komputerowa

Prowadzący: mgr Mikołaj Grygiel

**Laboratorium 9**

23.06.2024

**Temat:** Podstawy Three.js

**Wariant**: 1

Radosław Skrzypczyński

Informatyka I stopień,

niestacjonarne,

4 semestr,

Gr. 1B

1. **Polecenie:**

**2. Wprowadzane dane:**

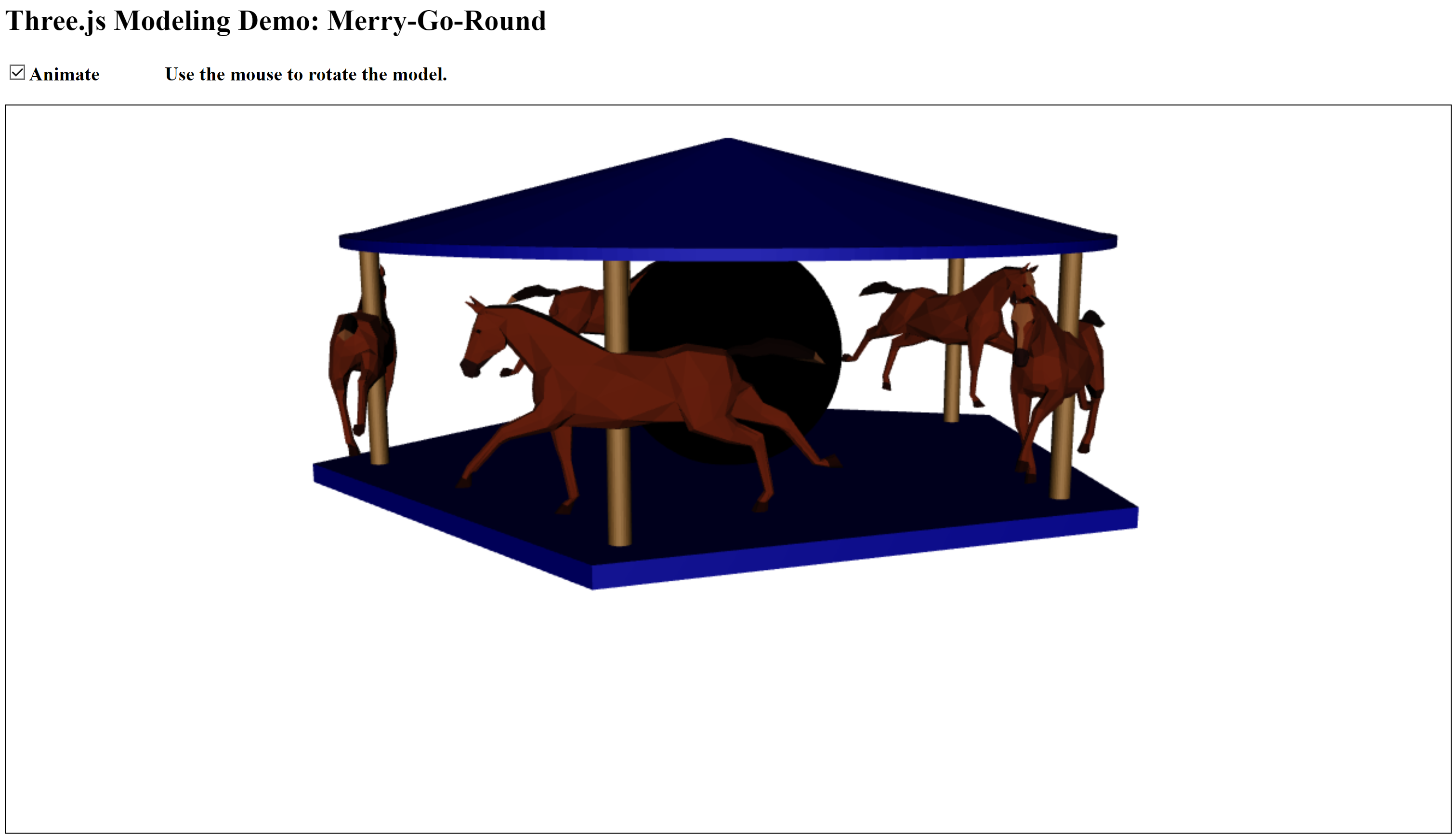
Przycisk włączający animację

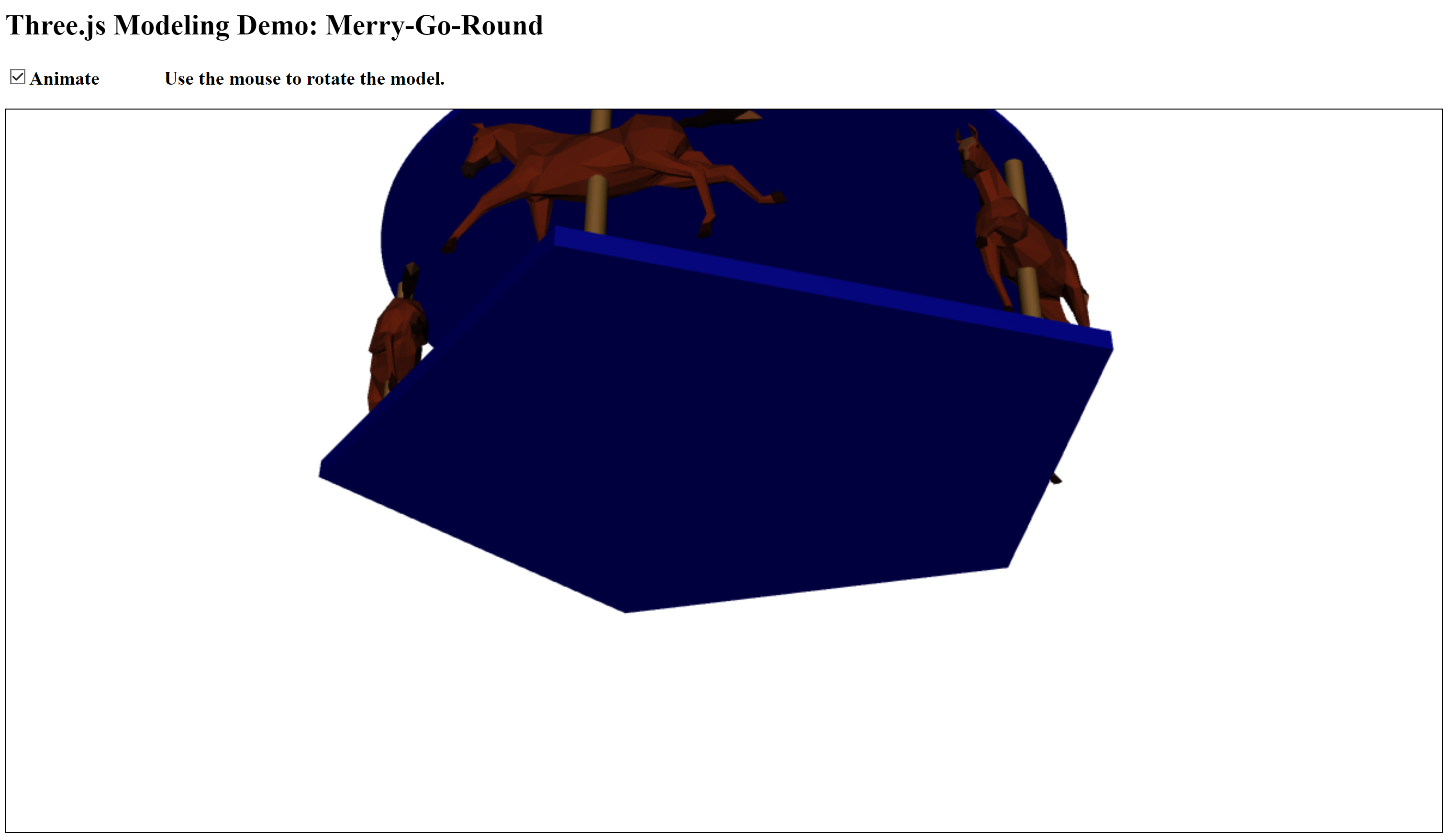
**3. Wykorzystane komendy:**

<!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 *floor*;  
 var *pole1*,*pole2*,*pole3*,*pole4*,*pole5*;  
 var *pivot1*,*pivot2*,*pivot3*,*pivot4*,*pivot5*;  
 var *roof*;  
 var *roof2*;  
 var *k1*,*k2*,*k3*,*k4*,*k5*,*k6*;  
 */\*\*  
 \* 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("white"); // Background color for scene.  
 *scene* = new THREE.Scene();  
  
 *camera* = new THREE.PerspectiveCamera(30, *canvas*.width/*canvas*.height, 0.1, 100);  
 *camera*.position.z = 40;  
 *camera*.position.y = 20;  
 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 ----------------------  
 *floor* = new THREE.Mesh(  
 new THREE.CylinderGeometry(13.5,13.5,0.6,7,1),  
 new THREE.MeshPhongMaterial({  
 color: 0x441c84,  
 specular: 0x222222,  
 shininess: 8,  
 shading: THREE.FlatShading  
 })  
 );  
 *floor*.rotation.y = *Math*.PI/12;  
 *scene*.add(*floor*);  
  
 var geometry = new THREE.SphereGeometry( 3.7, 32, 32 );  
 var material = new THREE.MeshBasicMaterial( { map: new THREE.TextureLoader().load('resources/earth.jpg') });  
 var sphere = new THREE.Mesh(geometry, material);  
 sphere.position.y=3.8;  
 *scene*.add(sphere);  
  
 *pole1* = new THREE.Mesh(  
 new THREE.CylinderGeometry(0.3,0.3,7.5,30,1),  
 new THREE.MeshPhongMaterial({  
 color: 0x7c5426,  
 specular: 0x222222,  
 shininess: 8,  
 shading: THREE.FlatShading  
 })  
 );  
 *pole1*.position.x=11.2;  
 *pole1*.position.y=3.9;  
 *pole1*.position.z=0.55;  
 *pole1*.rotation.y = *Math*.PI/12;  
 *scene*.add(*pole1*);  
  
 *pole2* = new THREE.Mesh(  
 new THREE.CylinderGeometry(0.3,0.3,7.5,30,1),  
 new THREE.MeshPhongMaterial({  
 color: 0x7c5426,  
 specular: 0x222222,  
 shininess: 8,  
 shading: THREE.FlatShading  
 })  
 );  
 *pole2*.position.x=-9.5;  
 *pole2*.position.y=3.9;  
 *pole2*.position.z=6.2;  
 *pole2*.rotation.y = *Math*.PI/12;  
 *scene*.add(*pole2*);  
  
 *pole3* = new THREE.Mesh(  
 new THREE.CylinderGeometry(0.3,0.3,7.5,30,1),  
 new THREE.MeshPhongMaterial({  
 color: 0x7c5426,  
 specular: 0x222222,  
 shininess: 8,  
 shading: THREE.FlatShading  
 })  
 );  
 *pole3*.position.x=2.95;  
 *pole3*.position.y=3.9;  
 *pole3*.position.z=11;  
 *pole3*.rotation.y = *Math*.PI/12;  
 *scene*.add(*pole3*);  
  
 *pole4* = new THREE.Mesh(  
 new THREE.CylinderGeometry(0.3,0.3,7.5,30,1),  
 new THREE.MeshPhongMaterial({  
 color: 0x7c5426,  
 specular: 0x222222,  
 shininess: 8,  
 shading: THREE.FlatShading  
 })  
 );  
 *pole4*.position.x=4;  
 *pole4*.position.y=3.9;  
 *pole4*.position.z=-10.5;  
 *pole4*.rotation.y = *Math*.PI/12;  
 *scene*.add(*pole4*);  
  
  
 *pole5* = new THREE.Mesh(  
 new THREE.CylinderGeometry(0.3,0.3,7.5,30,1),  
 new THREE.MeshPhongMaterial({  
 color: 0x7c5426,  
 specular: 0x222222,  
 shininess: 8,  
 shading: THREE.FlatShading  
 })  
 );  
 *pole5*.position.x=-8.7;  
 *pole5*.position.y=3.9;  
 *pole5*.position.z=-7.1;  
 *pole5*.rotation.y = *Math*.PI/12;  
 *scene*.add(*pole5*);  
  
 *roof* = new THREE.Mesh(  
 new THREE.CylinderGeometry(0.1,12,3,30,1),  
 new THREE.MeshPhongMaterial({  
 color: 0x441c84,  
 specular: 0x222222,  
 shininess: 8,  
 shading: THREE.FlatShading  
 })  
 );  
 *roof*.position.y=9.1;  
 *scene*.add(*roof*);  
  
 *roof2* = new THREE.Mesh(  
 new THREE.CylinderGeometry(12,12,0.3,200,1),  
 new THREE.MeshPhongMaterial({  
 color: 0x441c84,  
 specular: 0x222222,  
 shininess: 8,  
 shading: THREE.FlatShading  
 })  
 );  
 *roof2*.position.y=7.5;  
 *scene*.add(*roof2*);  
  
 var loader = new THREE.GLTFLoader();  
  
 var horse1 = loader.load( 'https://threejs.org/examples/models/gltf/Horse.glb', function ( gltf ) {  
 gltf.scene.scale.multiplyScalar( 0.03 );  
 gltf.scene.position.x = 11;  
 gltf.scene.position.z = 1;  
 gltf.scene.position.y = 1;  
 *pivot1*.add( gltf.scene ); // Add horse to pivot1  
 });  
  
 var horse2 = loader.load( 'https://threejs.org/examples/models/gltf/Horse.glb', function ( gltf ) {  
 gltf.scene.scale.multiplyScalar( 0.03 );  
 gltf.scene.position.x = 3;  
 gltf.scene.position.z = 11;  
 gltf.scene.position.y = 1;  
 gltf.scene.rotation.y = -1;  
 *pivot2*.add( gltf.scene ); // Add horse to pivot2  
 });  
  
 var horse3 = loader.load( 'https://threejs.org/examples/models/gltf/Horse.glb', function ( gltf ) {  
 gltf.scene.scale.multiplyScalar( 0.03 );  
 gltf.scene.position.x = 4;  
 gltf.scene.position.z = -10.5;  
 gltf.scene.position.y = 1;  
 gltf.scene.rotation.y = 1.2;  
 *pivot3*.add( gltf.scene ); // Add horse to pivot3  
 });  
  
 var horse4 = loader.load( 'https://threejs.org/examples/models/gltf/Horse.glb', function ( gltf ) {  
 gltf.scene.scale.multiplyScalar( 0.03 );  
 gltf.scene.position.x = -10;  
 gltf.scene.position.z = 6;  
 gltf.scene.position.y = 1;  
 gltf.scene.rotation.y = -2.5;  
 *pivot4*.add( gltf.scene ); // Add horse to pivot4  
 });  
  
 var horse5 = loader.load( 'https://threejs.org/examples/models/gltf/Horse.glb', function ( gltf ) {  
 gltf.scene.scale.multiplyScalar( 0.03 );  
 gltf.scene.position.x = -9;  
 gltf.scene.position.z = -7;  
 gltf.scene.position.y = 1;  
 gltf.scene.rotation.y = -3.5;  
 *pivot5*.add( gltf.scene ); // Add horse to pivot5  
 });  
  
 var box1 = new THREE.Box3().setFromObject(*pole1*,horse1);  
 var box2 = new THREE.Box3().setFromObject(*pole2*);  
 var box3 = new THREE.Box3().setFromObject(*pole3*);  
 var box4 = new THREE.Box3().setFromObject(*pole4*);  
 var box5 = new THREE.Box3().setFromObject(*pole5*);  
  
 box1.center( *pole1*.position );  
 box2.center( *pole2*.position );  
 box3.center( *pole3*.position );  
 box4.center( *pole4*.position );  
 box5.center( *pole5*.position );  
  
 *pivot1* = new THREE.Group();  
 *pivot2* = new THREE.Group();  
 *pivot3* = new THREE.Group();  
 *pivot4* = new THREE.Group();  
 *pivot5* = new THREE.Group();  
  
 *scene*.add(*pivot1*);  
 *scene*.add(*pivot2*);  
 *scene*.add(*pivot3*);  
 *scene*.add(*pivot4*);  
 *scene*.add(*pivot5*);  
  
 *pivot1*.add(*pole1*);  
 *pivot2*.add(*pole2*);  
 *pivot3*.add(*pole3*);  
 *pivot4*.add(*pole4*);  
 *pivot5*.add(*pole5*);  
  
 } // 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 incremented 1 before this function is called.  
 \*/* function updateForFrame() {  
 let rotation = -0.01;  
 *floor*.rotation.y += rotation;  
 *roof*.rotation.y += rotation;  
 *roof2*.rotation.y += rotation;  
  
 *pivot1*.rotation.y += rotation;  
 *pivot2*.rotation.y += rotation;  
 *pivot3*.rotation.y += rotation;  
 *pivot4*.rotation.y += rotation;  
 *pivot5*.rotation.y += rotation;  
  
 }  
  
  
 /\* ---------------------------- 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>

Link do zdalnego repozytorium: https://github.com/Slayzerus/UBB\_GrafikaKomputerowa/tree/main/Lab%209

**4. Wynik działania:**

****

****

**5. Wnioski:**

Wprowadzenie do podstaw biblioteki Three.js pozwoliło na zrozumienie kluczowych elementów tworzenia grafiki 3D w przeglądarkach internetowych. Three.js umożliwia tworzenie, renderowanie oraz animowanie scen 3D za pomocą HTML i JavaScript, co stanowi potężne narzędzie dla programistów zajmujących się grafiką komputerową. Biblioteka ta korzysta z WebGL, co umożliwia wykorzystanie sprzętowej akceleracji grafiki w nowoczesnych przeglądarkach.

W trakcie pracy z Three.js, kluczowymi elementami okazały się sceny, kamery oraz mechanizmy renderujące. Scena jest głównym obiektem, który przechowuje wszystkie elementy tworzące świat 3D, w tym oświetlenie, obiekty graficzne i kamery. Kamera reprezentuje punkt widzenia, z którego scena jest oglądana, a WebGLRenderer jest odpowiedzialny za renderowanie sceny na elemencie canvas. W celu stworzenia realistycznego obrazu, konieczne było również zrozumienie różnych typów kamer oraz ich konfiguracji.

Obiekty Three.js, takie jak geometrie, materiały i oświetlenie, pozwalają na dokładne dostosowanie wyglądu i zachowania sceny 3D. Geometrie określają kształt obiektów, materiały definiują ich powierzchnię, a oświetlenie wpływa na to, jak obiekty są postrzegane. Biblioteka oferuje szeroką gamę gotowych klas do tworzenia punktów, linii i trójkątów, co znacząco upraszcza proces tworzenia złożonych modeli.

Three.js także ułatwia zarządzanie hierarchią obiektów w scenie poprzez mechanizm grafu sceny. Obiekty mogą być organizowane w strukturę drzewiastą, co pozwala na łatwe grupowanie i manipulowanie wieloma obiektami jednocześnie. Metody takie jak add() i remove() umożliwiają dynamiczne dodawanie i usuwanie obiektów z grafu sceny.

Transformacje obiektów, takie jak skalowanie, obracanie i przesuwanie, są integralną częścią Three.js. Każdy obiekt posiada właściwości takie jak position, rotation i scale, które pozwalają na precyzyjne kontrolowanie jego położenia i orientacji w przestrzeni 3D. Te transformacje są kluczowe dla tworzenia realistycznych animacji i interakcji w scenie.

Praca z Three.js pokazała również znaczenie oświetlenia w tworzeniu realistycznych scen 3D. Różne typy świateł, takie jak światła kierunkowe, punktowe, otoczenia i reflektory, pozwalają na dokładne modelowanie efektów świetlnych. Światła wpływają na wygląd obiektów, podkreślając ich kształty i materiały.

Three.js oferuje również zaawansowane funkcje, takie jak tekstury, które pozwalają na dodawanie szczegółów powierzchni obiektów. Tekstury mogą być ładowane z plików i stosowane do obiektów, co znacząco zwiększa realizm sceny. Dzięki temu możliwe jest tworzenie skomplikowanych i realistycznych modeli 3D.

Podsumowując, Three.js jest potężnym narzędziem do tworzenia grafiki 3D w przeglądarkach internetowych. Pozwala na tworzenie złożonych scen, zarządzanie hierarchią obiektów, stosowanie różnych rodzajów oświetlenia oraz dodawanie tekstur. Praca z Three.js wymaga zrozumienia podstawowych pojęć z zakresu grafiki komputerowej, ale oferuje ogromne możliwości dla twórców interaktywnych aplikacji 3D.