**Three.js** ‑це кросбраузерна JavaScript бібліотека, яка використовується для створення та відображення анімованої тривимірної комп'ютерної графіки у веб-браузері. Three.js використовує WebGl.

***Слайд 1. WebGl***– це програмна бібліотека для JavaScript, яка дозволяє створювати 3D графіку, яка функціонує у браузерах.

[Three.js – JavaScript 3D Library](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjEpeqo5vv5AhX8GVkFHdghDj4QFnoECAQQAQ&url=https%3A%2F%2Fthreejs.org%2F&usg=AOvVaw10Toldt3Qs_8oAPRHb0CA6)

[https://threejs.org](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjEpeqo5vv5AhX8GVkFHdghDj4QFnoECAQQAQ&url=https%3A%2F%2Fthreejs.org%2F&usg=AOvVaw10Toldt3Qs_8oAPRHb0CA6)

Зараз перейдіть за посиланням.

***Слайд 2,3.*** На сайті є документація, приклади та редактор.

***Слайд 4.*** Моделювання графіки з використанням Three.js дає можливість оперувати такими поняттями як сцена, світло, камера, об'єкти та їх матеріали.

У цій лекції ми розглянемо основи використання Three.js, структуру необхідного HTML, фундаментальні об'єкти бібліотеки та як створити просту сцену.

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>Приклад 01.01 – Порожня сторінка</title>

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

</head>

<body>

<script>

init();

function init()

{

}

</script>

</body>

</html>

***Створення сцени***

***Слайд 5.*** Для того, щоб була можливість щось відображати з використанням Three.js, потрібні три речі: сцена (***scene***), камера (***camera***) і візуалізатор (***renderer***) рендерер, щоб була можливість показувати сцену, зняту камерою.

* ***Scene*** своєрідна платформа, де розміщуються всі об'єкти, які ми створюємо;
* ***Camera*** *‑*по суті - це "око", яке буде спрямовано на сцену. Камера знімає та відображає об'єкти, які розташовані на сцені;
* ***Renderer****‑*візуалізатор, який дозволяє відображати сцену, зняту камерою.

//Ініціалізації рендерера

var renderer = new THREE.WebGLRenderer();

renderer.setClearColor(new THREE.Color(0xEEEEEE));

// Встановлення розміру

renderer.setSize( window.innerWidth, window.innerHeight );

//Додавання елемента рендерера на сторінку. Робиться це так і однаково для всіх типів рендерерів:

document.body.appendChild( renderer.domElement );

var camera = new THREE.PerspectiveCamera(45,window.innerWidth / window.innerHeight, 0.1,1000);

var scene = new THREE.Scene();

У даному прикладі використовується перспективна камера.

Перспективна камера приймає 4 аргументи:

* ***Fov***або Field Of View (поле/кут зору)‑визначає кут, який можна бачити навколо центру камери.
* ***Aspect ratio*** ‑пропорція, або співвідношення ширини до висоти екрана. При великих значеннях поля зору видимий розмір об'єктів швидко зменшується на відстані. При невеликих значеннях, навпаки, видимий розмір об'єктів слабо залежить від відстані.
* ***Near & Far*** ‑мінімальна та максимальна відстань від камери, яка потрапляє у рендеринг. Так, дуже далекі точки не будуть рисуватись взагалі, як і точки, які знаходяться дуже близько.

Таким чином ми створилисцену (scene), камеру (camera) і візуалізатор але сцена в нас порожня.

***Слайд 6. Створення об'єктів на сцені***

Об'єкт, який створюється на сцені, називається ***Mesh***.

***Mesh***- це клас, який представляє об'єкти на основі трикутної полігональної сітки.

Цей клас приймає 2 аргументи:

* ***Geometry*** *‑*описує форму (положення вершин, грані, радіус тощо)
* ***Material****‑* описує зовнішній вигляд об'єктів (колір, текстура, прозорість тощо)

Спробуємо створити найпростіші об'єкти – осі координат, площину та куб.

var axes = new THREE.AxisHelper( 20 );

scene.add(axes);

var planeGeometry = new THREE.PlaneGeometry(60,20,1,1);

var planeMaterial = new THREE.MeshBasicMaterial({color: 0xcccccc});

var plane = new THREE.Mesh(planeGeometry,planeMaterial);

plane.rotation.x=-0.5\*Math.PI;

plane.position.x = 15;

plane.position.y = 0;

plane.position.z = 0;

scene.add(plane);

var cubeGeometry = new THREE.CubeGeometry(4,4,4);

var cubeMaterial = new THREE.MeshBasicMaterial({color: 0xff0000, wireframe: true});

var cube = new THREE.Mesh(cubeGeometry, cubeMaterial);

cube.position.x = -4;

cube.position.y = 3;

cube.position.z = 0;

scene.add(cube);

Three.js надає кілька типів матеріалів. Вони визначають, як об'єкти з'являтимуться на сцені. Які матеріали ви використовуєте залежить від того, чого ви намагаєтеся досягти.

В даному прикладі використовується матеріал [***MeshBasicMaterial***](https://threejs.org/docs/#api/en/materials/MeshBasicMaterial), який не залежить від світла.

Основні властивості якого:

color: колір матеріалу (за замовчуванням встановлений білий (0xffffff)).

wireframe (каркас): візуалізація геометрії як каркаса, тобто вершин та ребер. За промовчанням -false.

***Слайд 7.*** В наведеному прикладі ми зустріли не розглянуті функції position та rotation.

Для розташування фігур у просторі використовується Декартова система координат:

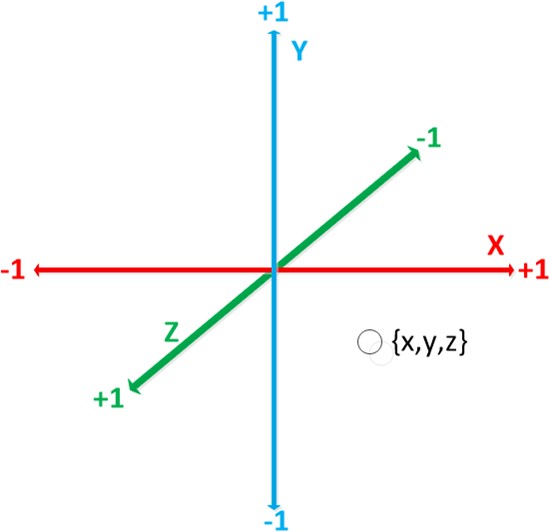


Рисунок 1‑Декартова система координат у WebGL

Як завжди, для програм, які працюють із ***3D***, вгору спрямована вісь ***Y***. На рисунку також вказані правила зміни знаків. Наприклад, при наближенні до нас значення координати ***Z*** збільшується.

При додаванні об'єкта на сцену ***Three.js*** можна вказати його координати:

object.position.x = -50; object.position.y = 20; object.position.z = 60;

або одним рядком:

object.position.set(-50, 20, 60);

або за допомогою класу тривимірних векторів ***Vector3***:

object.position = new THREE.Vector3( -50, 20, 60 );

При цьому у зазначеній точці зазвичай розташований геометричний центр тіла. Якщо ж координати об'єкта не вказані, всі вони рівні нулю.

Можна встановити координати об'єкта, прирівнявши їх до координат іншого:

object2.position = object1.position;

Для завдання кутів повороту тіла використовують властивість rotation. Кути вказуються у радіанах. Наприклад, команда

object.rotation.y = Math.PI/2;

означає, що об'єкт обертається на 90 градусів проти годинникової стрілки, якщо дивитися "зверху" - з боку позитивного напрямку вісі ***OY***, при цьому віссю обертання служить вісь ординат ***OY***. Синтаксис rotation такий же, як і в position.

Останньою дією нам необхідно запустити рендеринг.

Далі йдевізуалізатор (рендерер).

Тут можна пояснити, що таке рендеринг (rendering), рендер (render) і рендерер (renderer).

***Рендеринг*** (Візуалізація) це процес перерахунку тривимірної сцени, тобто. всіх об'єктів, які знаходяться в полі видимості камери, двовимірне зображення. Ось це підсумкове зображення, яке відображається на екрані монітора, називається рендером. А програма, яка виконує рендеринг, відповідно називається рендерером або візуалізатором.

Крім створення екземпляра візуалізатора (рендерера), також потрібно встановити розміри, в яких буде відображатися наша програма. Хорошою ідеєю буде використання ширини та висоти області, яку нам хочеться зайняти нашим додатком – у даному випадку, це ширина та висота вікна браузера. При виконанні ресурсоємних програм також можна задати параметру setSize менші значення, window.innerWidth/2 і window.innerHeight/2, які зроблять візуалізацію програми у половинному розмірі.

Якщо потрібно зберегти розміри програми, але відобразити її з нижчою роздільною здатністю, можна зробити це, викликавши параметр setSize зі значенням false як параметр updateStyle.

Наприклад, код

setSize(window.innerWidth/2, window.innerHeight/2, false)

зробить візуалізацію програми у половинному розмірі, з урахуванням того, що наш <canvas> має100%. -ву ширину і висоту від розмірів вікна браузера.

І нарешті, що не менш важливо, до HTML-документу додаємо елемент візуалізатора. Це елемент<canvas>, де рендерер і відображає сцену.

***Слайд 8*** Приклад prime01\_02

***Слайд 9 АНІМАЦІЯ***

Анімація – послідовний неодноразовий рендеринг сцени (для відображення динаміки). Таке послідовне відображення здійснюється функцією requestAnimationFrame.

приклад (***prime01-03.html***):

function renderScene()

{

cube.rotation.x += 0.02;

cube.rotation.y+=0.02;

cube.rotation.z += 0.02;

requestAnimationFrame(renderScene);

renderer.render (scene, camera);

}

***керування оглядом сцени***

З використанням перспективної камери важко досить повно оцінити тривимірність об'єктів на сцені. Хочеться мати можливість розглядати об'єкти з різних боків.

Для цього Three.js є можливість керування оглядом сцени за допомогою спеціальних «контролів», додавання яких дозволяє змінювати точку огляду камери за допомогою мишки, наближатися або віддалятися від сцени.

Для керування оглядом необхідно завантажити файл TrackballControls.js сайту <http://Threejs.org> у папку js і для його використання додати посилання (***prime01-04.html***):

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>Приклад 01.04</title>

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

<script src="./js/TrackballControls.js"></script>

</head>

Тепер, після створення камери, можна додати контрол:

var camera = new THREE.PerspectiveCamera(45,window.innerWidth / window.innerHeight, 0.1,1000);

var scene = new THREE.Scene();

var controls = new THREE.TrackballControls( camera );

Перший параметр Camera.

Після створення контролу можна задати, наприклад, швидкість обертання камери під час руху миші:

controls.rotateSpeed=2; (1)

чи змінюватиметься положення камери (наближатися або віддалятися від сцени) при крученні коліщатка миші (за замовчуванням false - змінюється), і швидкість такої зміни:

controls.noZoom=false; (2)

controls.zoomSpeed=1.2; (3)

Також можна вказати, чи камера після зупинки миші трохи рухатиметься за інерцією чи ні, за замовчуванням false (рухається):

controls.staticMoving=true; (4)

Також є інші контроли. Наприклад, FirstPersonConrols дозволяє "бачити" сцену від першої особи, і пересуватися по ній за допомогою стрілок (або кнопок WASD) і миші.