Лабораторна робота №1: Ініціалізація ядра Three.js та створення першої сцени

1. Вступна частина

Мета роботи

Ознайомлення з архітектурою бібліотеки Three.js, налаштування середовища розробки та опанування базових принципів візуалізації 3D-контенту у веб-середовищі.

Перелік компетенцій

- Розуміння архітектури рендерингу в браузері (WebGL/GPU).
- Вміння проектувати ієрархічні структури даних (Scene Graph).
- Навички роботи з сучасними інструментами збірки (Vite, ES6 модулі).
- Здатність створювати адаптивні графічні інтерфейси для імерсивних застосунків.

2. Теоретична частина

2.1. Архітектура Three.js та Scene Graph

Three.js базується на концепції **Scene Graph** (Граф сцени). Це деревоподібна структура даних, де кожен вузол (Node) представляє об'єкт у просторі.

- **Scene (Сцена):** Це кореневий контейнер. Все, що ви бачите (меші, світло, камери), має бути додане до сцени.
- Ієрархія дозволяє успадковувати трансформації (позиція, поворот, масштаб) від батьківських об'єктів до дочірніх.

2.2. WebGLRenderer: Міст між CPU та GPU

WebGLRenderer — це компонент, який перетворює математичний опис сцени в пікселі на екрані.

- Він використовує API WebGL для взаємодії з відеокартою.
- Рендерер створює елемент <canvas>, у якому відбувається малювання.
- Основна функція рендерера — метод .render(scene, camera), який виконує проекцію 3D-простору на 2D-площину екрана.

2.3. Камери та PerspectiveCamera

У Three.js камера визначає, що саме потрапить у кадр. Найчастіше використовується PerspectiveCamera, яка імітує людське око. Параметри конструктора:

1. **FOV (Field of View):** Кут огляду по вертикалі (у градусах). Чим більший кут, тим більше об'єктів потрапляє в кадр, але виникають дисторсії.
2. **Aspect Ratio:** Співвідношення сторін (ширина / висота). Зазвичай відповідає розміру контейнера.
3. **Near (Ближня площина відсікання):** Об'єкти ближче цього значення не відображаються.
4. **Far (Дальня площина відсікання):** Об'єкти далі цього значення не відображаються.

*Область між Near та Far називається **Frustum** (зрізана піраміда видимості).*

2.4. Життєвий цикл: Render Loop

Для створення ілюзії руху необхідно перемальовувати сцену з частотою 60+ кадрів на секунду (FPS). Ми використовуємо requestAnimationFrame. Це API браузера, яке:

- Призупиняє рендеринг, якщо вкладка неактивна (економія ресурсів).
- Синхронізує оновлення з частотою оновлення монітора.

2.5. Система координат

Three.js використовує **праву систему координат** (Right-handed):

- **X:** спрямована праворуч.
- **Y:** спрямована вгору.
- **Z:** спрямована "на глядача" (позитивні значення виходять з екрана).

3. Практичне завдання

Крок 1: Підготовка середовища

Для швидкої збірки використаємо **Vite**. У терміналі виконайте:

```
npm create vite@latest my-three-app -- --template vanilla
cd my-three-app
npm install three
npm run dev
```

Крок 2: Базова реалізація

Створіть файл main.js та ініціалізуйте ядро системи:

```
import * as THREE from 'three';

// 1. Створення сцени
const scene = new THREE.Scene();
scene.background = new THREE.Color(0x111111);

// 2. Налаштування камери
const camera = new THREE.PerspectiveCamera(
    75,
    window.innerWidth / window.innerHeight,
    0.1,
    1000
);
camera.position.z = 5;

// 3. Створення рендерера
const renderer = new THREE.WebGLRenderer({ antialias: true });
renderer.setSize(window.innerWidth, window.innerHeight);
renderer.setPixelRatio(window.devicePixelRatio);
document.body.appendChild(renderer.domElement);

// 4. Додавання об'єкта (Mesh)
const geometry = new THREE.BoxGeometry(1, 1, 1);
const material = new THREE.MeshNormalMaterial();
const cube = new THREE.Mesh(geometry, material);
scene.add(cube);

// 5. Обробка адаптивності (Resize)
window.addEventListener('resize', () => {
    // Оновлення параметрів камери
    camera.aspect = window.innerWidth / window.innerHeight;
    camera.updateProjectionMatrix();

    // Оновлення розміру рендерера
    renderer.setSize(window.innerWidth, window.innerHeight);
});

// 6. Цикл анімації (Render Loop)
function animate() {
    requestAnimationFrame(animate);

    // Обертання об'єкта
```

```
    cube.rotation.x += 0.01;
    cube.rotation.y += 0.01;

    renderer.render(scene, camera);
}

animate();
```

4. Завдання для самостійної роботи

Рівень: Базовий

1. **Дисторсія камери:** Змініть параметр FOV на 120 та 30. Проаналізуйте, як змінюється сприйняття об'єкта.
2. **Позиціонування:** Створіть три різні куби та розмістіть їх у ряд вздовж осі X (наприклад, на позиціях -2, 0, 2).

Рівень: Просунутий

1. **Геометрія:** Замініть BoxGeometry на іншу стандартну геометрію (наприклад, TorusKnotGeometry або IcosahedronGeometry).
2. **Творче завдання:** Реалізуйте зміну кольору фону сцени залежно від положення куба або за таймером.
3. **Анімація за умовою:** Зробіть так, щоб швидкість обертання об'єкта залежала від його віддаленості від центру координат по осі Z.

5. Контрольні питання та звітність

Питання для самоперевірки:

1. Чим відрізняється PerspectiveCamera від OrthographicCamera? У яких випадках в AR/VR краще використовувати кожну з них?
2. Що станеться, якщо параметр near встановити більшим за far?
3. Яка роль методу updateProjectionMatrix() при зміні розміру вікна? Що буде, якщо його не викликати?
4. Чому використання setInterval для анімації є поганою практикою порівняно з requestAnimationFrame?