МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра інформатики та програмної інженерії

Звіт

З лабораторної роботи № 6 з дисципліни
«Комп’ютерна графіка та обробка зображень»

Тема: «Проектування у тривимірному просторі»

Виконав(ла) *ІП-15 Мєшков Андрій* _____

(шифр, прізвище, ім'я, по батькові)

Перевірив *Щебланін Ю. М.* _____

(шифр, прізвище, ім'я, по батькові)

Київ 2024

ЗМІСТ

ВСТУП

У сучасному світі комп'ютерна графіка відіграє ключову роль у багатьох сферах діяльності, включаючи дизайн, інженерію, розробку ігор та анімацію. Однією з важливих складових комп'ютерної графіки є проектування у тривимірному просторі, що дозволяє створювати реалістичні моделі та сцени. Ця лабораторна робота присвячена вивченню основних методів та алгоритмів, які використовуються для побудови та відображення тривимірних об'єктів на екрані.

Метою даної лабораторної роботи є ознайомлення з різними видами проекцій у тривимірному просторі, а також з методами налаштування камери для отримання різних видів зображень. У ході виконання роботи буде реалізовано декілька типів проекцій, включаючи ортографічні та перспективні проекції, а також побудову аксонометричних проекцій. В результаті роботи студенти отримають практичні навички створення та маніпуляції тривимірними об'єктами за допомогою бібліотеки Three.js.

ХІД РОБОТИ

Завдання 5.1

Побудова чотирьох проекцій куба в різних областях одного вікна.

Лістинг програмного коду:

Index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <title>Lab 6</title>
        <style>
            * {
                box-sizing: border-box;
            }
            body {
                margin: 0;
                padding: 0;
            }
        </style>
    </head>
    <body>
        <script type="module" src="./task2.js"></script>
    </body>
</html>
```

Task1.js

```
import * as THREE from 'three';
import { OrbitControls } from 'three/addons/controls/OrbitControls.js';

// Create the scene
const scene = new THREE.Scene();

// Create a cube with different colored faces
const geometry = new THREE.BoxGeometry();
const materials = [
    new THREE.MeshBasicMaterial({ color: 0x00ff00 }), // green
    new THREE.MeshBasicMaterial({ color: 0xffff00 }), // yellow
    new THREE.MeshBasicMaterial({ color: 0xff00ff }), // magenta
    new THREE.MeshBasicMaterial({ color: 0xff00ff }), // magenta
    new THREE.MeshBasicMaterial({ color: 0xff0000 }), // red
    new THREE.MeshBasicMaterial({ color: 0x0000ff }), // blue
];

const cube = new THREE.Mesh(geometry, materials);
scene.add(cube);
```

```
// Add coordinate axes to the scene
const axesHelper = new THREE.AxesHelper(5); // 5 units long
scene.add(axesHelper);

// Create the renderer
const renderer = new THREE.WebGLRenderer();
renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);

const aspectRatio = window.innerWidth / window.innerHeight;
const cameras = [];

// Camera 1: Front view (XOY)
const camera1 = new THREE.OrthographicCamera(-aspectRatio, aspectRatio, 1, -1, 0.1, 100);
camera1.position.set(0, 0, 5);
camera1.lookAt(0, 0, 0);
cameras.push(camera1);

// Camera 2: Top view (XOZ)
const camera2 = new THREE.OrthographicCamera(-aspectRatio, aspectRatio, 1, -1, 0.1, 100);
camera2.position.set(0, 5, 0);
camera2.lookAt(0, 0, 0);
camera2.up.set(0, 0, -1);
cameras.push(camera2);

// Camera 3: Side view (YOZ)
const camera3 = new THREE.OrthographicCamera(-aspectRatio, aspectRatio, 1, -1, 0.1, 100);
camera3.position.set(5, 0, 0);
camera3.lookAt(0, 0, 0);
cameras.push(camera3);

// Camera 4: Isometric view
const camera4 = new THREE.OrthographicCamera(-aspectRatio, aspectRatio, 1, -1, 0.1, 100);
camera4.position.set(5, 5, 5);
camera4.lookAt(0, 0, 0);
cameras.push(camera4);

function renderView(camera, x, y, width, height) {
    const windowWidth = window.innerWidth;
    const windowHeight = window.innerHeight;

    renderer.setViewport(x, y, width, height);
    renderer.setScissor(x, y, width, height);
    renderer.setScissorTest(true);

    camera.aspect = width / height;
    camera.updateProjectionMatrix();

    renderer.render(scene, camera);
}

function animate() {
    requestAnimationFrame(animate);
```

```
    // Clear the entire canvas
    renderer.clear();

    // Calculate viewports and render each view
    const width = window.innerWidth / 2;
    const height = window.innerHeight / 2;

    // Top-left: Front view (XOY)
    renderView(cameras[0], 0, height, width, height);

    // Bottom-left: Top view (XOZ)
    renderView(cameras[1], 0, 0, width, height);

    // Top-right: Side view (YOZ)
    renderView(cameras[2], width, height, width, height);

    // Bottom-right: Isometric view
    renderView(cameras[3], width, 0, width, height);
}

animate();
```

Результати виконання програми:

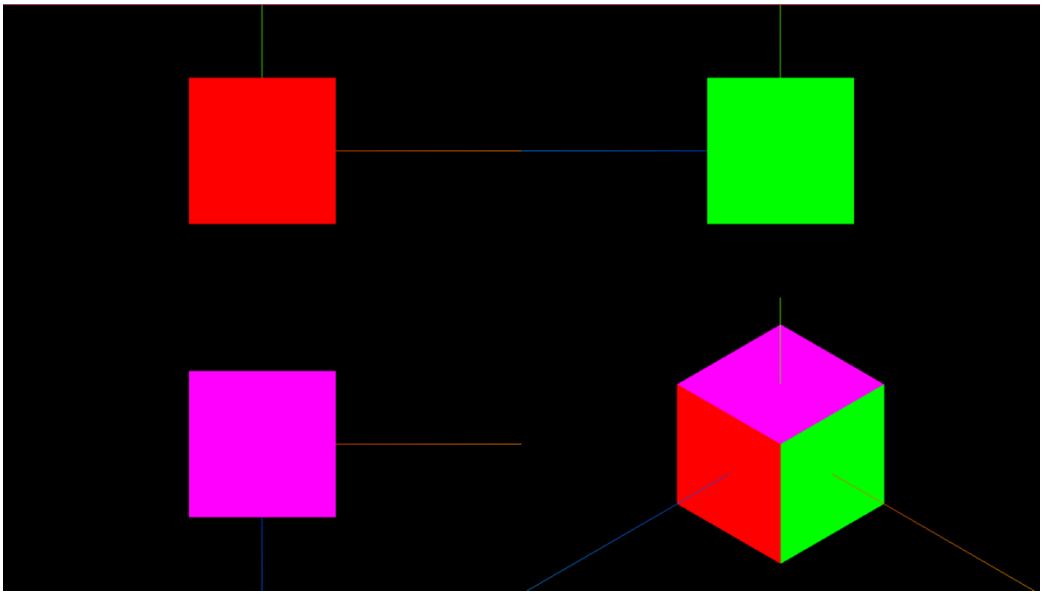

Рисунок 5.1 — Результат виконання програми

Завдання 5.2

Побудова чотирьох проекцій чайника в різних областях одного вікна.

Лістинг програмного коду:

Task2.js

```
import * as THREE from 'three';
import { OrbitControls } from 'three/addons/controls/OrbitControls.js';
import { TeapotGeometry } from 'three/addons/geometries/TeapotGeometry.js';

// Color constants
const COLORS = {
    C_White: new THREE.Color(1.0, 1.0, 1.0),
    C_Black: new THREE.Color(0.0, 0.0, 0.0),
    C_Grey: new THREE.Color(0.5, 0.5, 0.5),
    C_DarcGrey: new THREE.Color(0.2, 0.2, 0.2),
    C_Red: new THREE.Color(1.0, 0.0, 0.0),
    C_Green: new THREE.Color(0.0, 1.0, 0.0),
    C_Blue: new THREE.Color(0.0, 0.0, 1.0),
    C_DarcBlue: new THREE.Color(0.0, 0.0, 0.5),
    C_Cyan: new THREE.Color(0.0, 1.0, 1.0),
    C_Magenta: new THREE.Color(1.0, 0.0, 1.0),
    C_Yellow: new THREE.Color(1.0, 1.0, 0.0),
    C_Orange: new THREE.Color(0.1, 0.5, 0.0),
    C_Lemon: new THREE.Color(0.8, 1.0, 0.0),
    C_Brown: new THREE.Color(0.5, 0.3, 0.0),
    C_Navy: new THREE.Color(0.0, 0.4, 0.8),
    C_Aqua: new THREE.Color(0.4, 0.7, 1.0),
    C_Cherry: new THREE.Color(1.0, 0.0, 0.5)
};

// Create the scene
const scene = new THREE.Scene();

// Create a teapot
const geometry = new TeapotGeometry(2);
const material = new THREE.MeshPhongMaterial({ color: COLORS.C_Cherry });
const teapot = new THREE.Mesh(geometry, material);
scene.add(teapot);

// Add coordinate axes to the scene
const axesHelper = new THREE.AxesHelper(5); // 5 units long
scene.add(axesHelper);

// Setup lighting
const light = new THREE.DirectionalLight(0xffffff, 1);
light.position.set(1, 1, 1).normalize();
scene.add(light);

// Create the renderer
const renderer = new THREE.WebGLRenderer();
renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);

const aspectRatio = window.innerWidth / window.innerHeight;
const cameras = [];
```

```
// Camera 1: Front view (XOY)
const camera1 = new THREE.OrthographicCamera(-aspectRatio * 5, aspectRatio * 5, 5, -5, 0.1,
100);
camera1.position.set(0, 0, 5);
camera1.lookAt(0, 0, 0);
cameras.push(camera1);

// Camera 2: Top view (XOZ)
const camera2 = new THREE.OrthographicCamera(-aspectRatio * 5, aspectRatio * 5, 5, -5, 0.1,
100);
camera2.position.set(0, 5, 0);
camera2.lookAt(0, 0, 0);
camera2.up.set(0, 0, -1);
cameras.push(camera2);

// Camera 3: Side view (YOZ)
const camera3 = new THREE.OrthographicCamera(-aspectRatio * 5, aspectRatio * 5, 5, -5, 0.1,
100);
camera3.position.set(5, 0, 0);
camera3.lookAt(0, 0, 0);
cameras.push(camera3);

// Camera 4: Isometric view
const camera4 = new THREE.OrthographicCamera(-aspectRatio * 5, aspectRatio * 5, 5, -5, 0.1,
100);
camera4.position.set(5, 5, 5);
camera4.lookAt(0, 0, 0);
cameras.push(camera4);

function renderView(camera, x, y, width, height) {
    const windowWidth = window.innerWidth;
    const windowHeight = window.innerHeight;

    renderer.setViewport(x, y, width, height);
    renderer.setScissor(x, y, width, height);
    renderer.setScissorTest(true);

    camera.aspect = width / height;
    camera.updateProjectionMatrix();

    renderer.render(scene, camera);
}

function animate() {
    requestAnimationFrame(animate);

    renderer.clear();

    const width = window.innerWidth / 2;
    const height = window.innerHeight / 2;

    renderView(cameras[0], 0, height, width, height);
    renderView(cameras[1], 0, 0, width, height);
```

```
    renderView(cameras[2], width, height, width, height);
    renderView(cameras[3], width, 0, width, height);
}

animate();
```

Результати виконання програми:

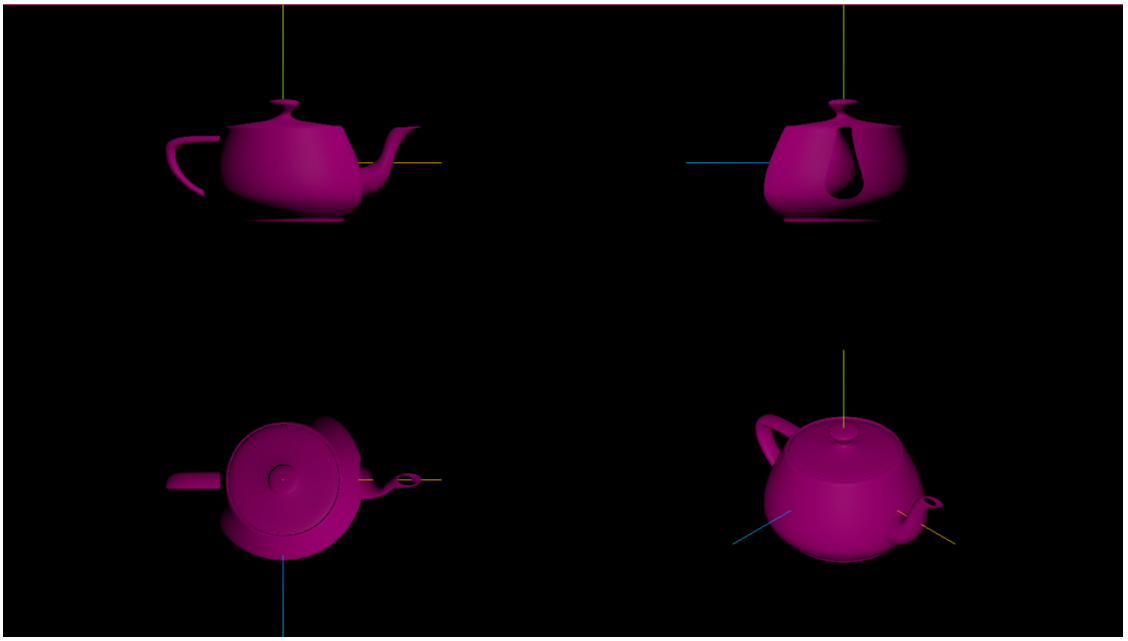

Рисунок 5.2 — Результат виконання програми

Завдання 5.3, 5.4, 5.5

Скопіювати файли проекту із завдання 2.2 в нову папку.

5.3 Для двох верхніх проекцій задати точки спостереження для одержання ортографічних проекцій згідно варіанту.

Варіант 7. Вид зліва і знизу.

5.4 На нижній лівій проекції побудувати аксонометричну проекцію згідно варіанту.

Варіант 7. $\varphi=45°, \psi= 75°$

На нижній правій проекції побудувати перспективну проекцію з кутом зору згідно варіанту.

Варіант 7. $\varphi=60°$

Підібрати інші параметри так, щоб видимий розмір куба приблизно дорівнював розмірам в інших проекціях.

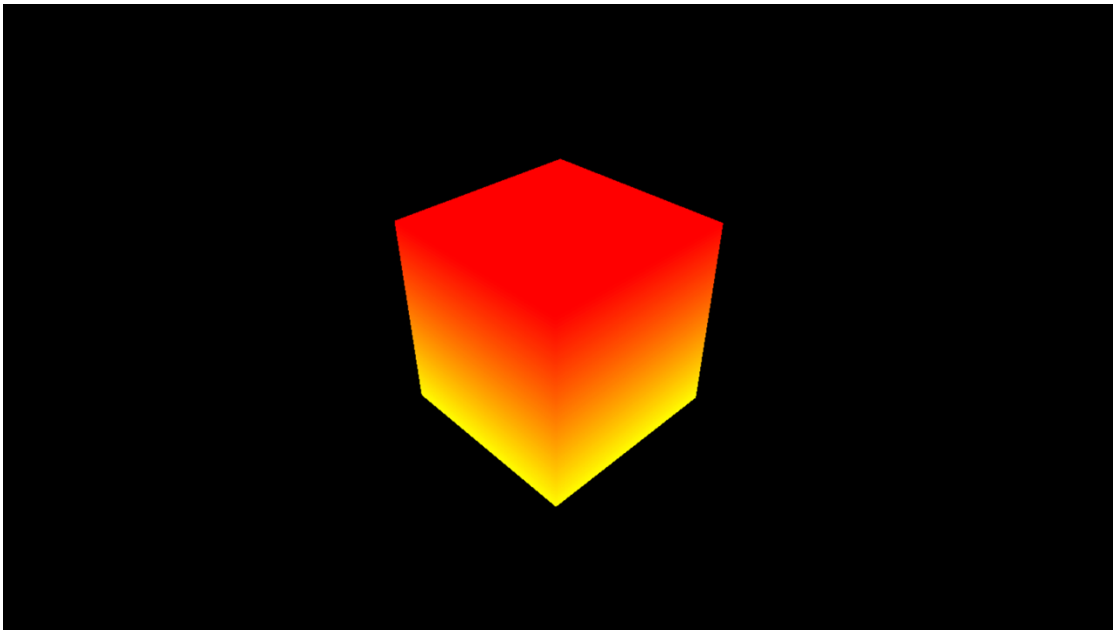

Рисунок 5.3 — Куб з завдання 2.2

Лістинг програмного коду:

Task3,4,5.js

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

const scene = new THREE.Scene();

const geometry = new THREE.BoxGeometry(2, 2, 2);
const gradient = new THREE.ShaderMaterial({
    uniforms: {
        color1: { value: new THREE.Color("yellow") },
        color2: { value: new THREE.Color("red") }
    },
    vertexShader: `
        varying vec2 vUv;
        void main() {
            vUv = uv;
            gl_Position = projectionMatrix * modelViewMatrix * vec4(position, 1.0);
        }
    `,
    fragmentShader: `
        uniform vec3 color1;
        uniform vec3 color2;
        varying vec2 vUv;
        void main() {
            gl_FragColor = vec4(mix(color1, color2, vUv.y), 1.0);
        }
    `
});

const materials = [
    gradient,
    gradient,
```

```
    new THREE.MeshBasicMaterial({ color: 0xFF0000 }),
    new THREE.MeshBasicMaterial({ color: 0xFFFF00 }),
    gradient,
    gradient
];

const cube = new THREE.Mesh(geometry, materials);
scene.add(cube);

const axesHelper = new THREE.AxesHelper(5);
scene.add(axesHelper);

const renderer = new THREE.WebGLRenderer();
renderer.setSize(window.innerWidth, window.innerHeight);
document.body.appendChild(renderer.domElement);

const aspectRatio = window.innerWidth / window.innerHeight;
const cameras = [];

// Camera 1: Orthographic view from the left (XOY)
const camera1 = new THREE.OrthographicCamera(-aspectRatio * 3, aspectRatio * 3, 3, -3, 0.1,
100);
camera1.position.set(-4, 0, 0);
camera1.lookAt(0, 0, 0);
cameras.push(camera1);

// Camera 2: Orthographic view from the bottom (XOZ)
const camera2 = new THREE.OrthographicCamera(-aspectRatio * 3, aspectRatio * 3, 3, -3, 0.1,
100);
camera2.position.set(0, -4, 0);
camera2.lookAt(0, 0, 0);
camera2.up.set(0, 0, -1);
cameras.push(camera2);

// Camera 3: Axonometric view (fi=45, psi=75)
const camera3 = new THREE.OrthographicCamera(-aspectRatio * 3, aspectRatio * 3, 3, -3, 0.1,
100);
camera3.position.set(-2, 3, 0);
camera3.lookAt(0, 0, 0);
camera3.rotation.order = 'YXZ'; // Rotate in the correct order
camera3.rotateY(THREE.MathUtils.degToRad(75));
camera3.rotateX(THREE.MathUtils.degToRad(45));
cameras.push(camera3);

// Camera 4: Perspective view (fi=60)
const camera4 = new THREE.PerspectiveCamera(60, aspectRatio, 0.1, 100);
camera4.position.set(4, 4, 4);
camera4.lookAt(0, 0, 0);
cameras.push(camera4);

function renderView(camera, x, y, width, height) {
    const windowWidth = window.innerWidth;
    const windowHeight = window.innerHeight;
```

```
    renderer.setViewport(x, y, width, height);
    renderer.setScissor(x, y, width, height);
    renderer.setScissorTest(true);

    camera.aspect = width / height;
    camera.updateProjectionMatrix();

    renderer.render(scene, camera);
}

function animate() {
    requestAnimationFrame(animate);

    renderer.clear();

    const width = window.innerWidth / 2;
    const height = window.innerHeight / 2;

    renderView(cameras[0], 0, height, width, height);
    renderView(cameras[2], 0, 0, width, height);
    renderView(cameras[1], width, height, width, height);
    renderView(cameras[3], width, 0, width, height);
}

animate();
```

Результат виконання програми:

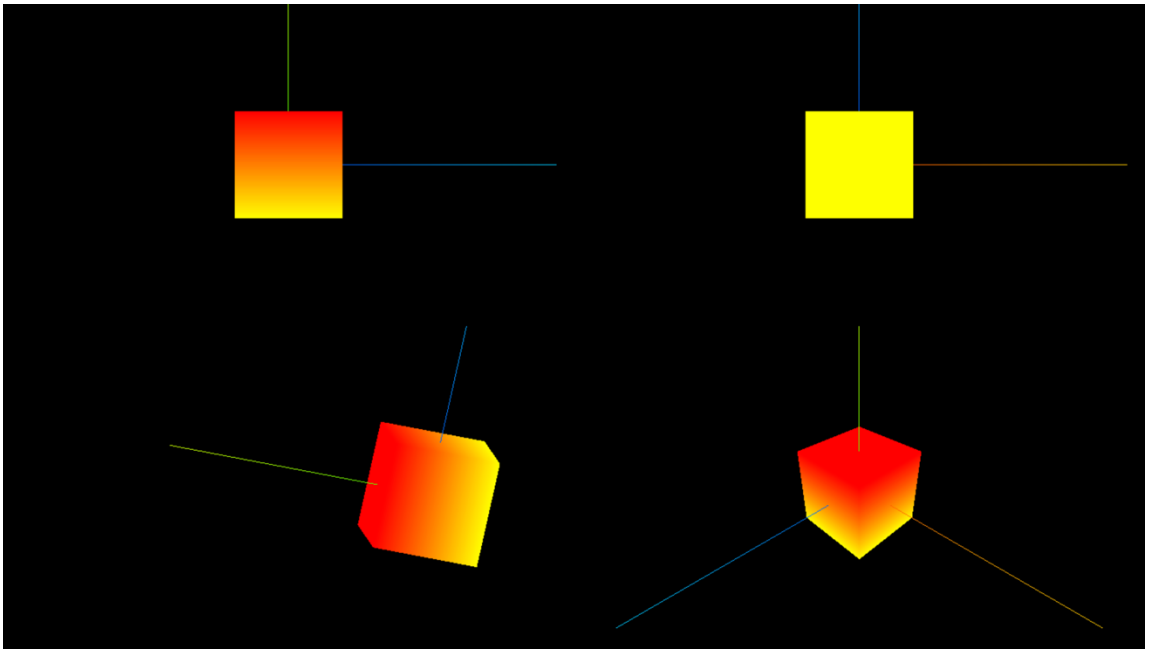

Рисунок 5.4 — Результат виконання програми

ВИСНОВКИ

Виконання лабораторної роботи дозволило отримати глибше розуміння принципів проектування у тривимірному просторі. У процесі роботи було реалізовано різні види проекцій, що дозволило побачити, як змінюється вигляд об'єктів при зміні точок спостереження та параметрів камери. Зокрема, ми навчилися створювати ортографічні проекції з різних точок зору, будувати аксонометричні проекції з заданими кутами та налаштовувати перспективні проекції з різними кутами огляду.

Практичні навички, отримані в ході виконання цієї лабораторної роботи, є фундаментальними для подальшого вивчення комп'ютерної графіки та розробки програмного забезпечення, яке використовує тривимірну графіку. Знання, здобуті під час лабораторної роботи, можуть бути застосовані у різних галузях, включаючи розробку ігор, візуалізацію даних, анімацію та інженерне проектування.

Таким чином, лабораторна робота з проектування у тривимірному просторі не лише ознайомила нас із основними поняттями та техніками, але й заклала основу для подальшого розвитку у сфері комп'ютерної графіки.

.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Three.js [Електронний ресурс] — Режим доступу: https://threejs.org/docs/