МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **КНІТ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 1
З дисципліни: *"Віртуальна реальність"*
На тему: *"Організація середовища розробки застосунків віртуальної реальності з використанням веб API WEBXR"*

Лектор:
асист. каф. ПЗ
Задорожний І.М.

Виконав:
ст. гр. ПЗ-43
Лесневич Є.Є.

Прийняв:
асист. каф. ПЗ
Бауск О.Є.

« ____ » _____ 2025 р.

∑= _____ .....

Львів – 2025

Тема роботи: Основи роботи з XebXR -- API доповненої реальністі на веб платформі.

Мета роботи: Налаштувати середовище розробки для роботи з веб API доповненої реальністі, ознайомитись з інструментами веб-розробки, Typescript, Three.js, WebXR API, імплементувати створення простого 3D об'єкту в WebXR, налаштувати remote debugging для мобільного пристрою.

ТЕОРЕТИЧНІ ВІДОМОСТІ

WebXR - це набір стандартів, які використовуються разом для підтримки рендерингу 3D сцен на пристроях, призначених для представлення віртуальних світів (віртуальна реальність, або VR), або для додавання графічних зображень до реального світу (доповнена реальність, або AR). WebXR Device API реалізує основний набір функцій WebXR, керуючи вибором вихідних пристроїв, рендерингом 3D сцени на обраному пристрої з відповідною частотою кадрів та керуванням векторами руху, створеними за допомогою контролерів введення.

Пристрої, сумісні з WebXR, включають повністю імерсивні 3D-гарнітури з відстеженням руху та орієнтації, окуляри, які накладають графіку поверх сцени реального світу, що проходить через кадри, та мобільні телефони, які доповнюють реальність, захоплюючи світ камерою та доповнюючи цю сцену комп'ютерно-згенерованими зображеннями.

Для досягнення цих цілей WebXR Device API надає такі ключові можливості: пошук сумісних пристроїв виводу VR або AR Рендеринг 3D сцени на пристрої з відповідною частотою кадрів (Опціонально) віддзеркалення виводу на 2D дисплей Створення векторів, що представляють рухи елементів керування введенням

На найбільш базовому рівні сцена представляється в 3D шляхом обчислення перспективи, яку потрібно застосувати до сцени, щоб відрендерити її з точки зору кожного ока користувача, обчислюючи положення кожного ока і рендерячи сцену з цієї позиції, дивлячись у напрямку, в якому користувач зараз дивиться. Кожне з цих двох зображень рендериться в єдиний фреймбуфер, де зображення лівого ока

рендериться зліва, а точка зору правого ока рендериться в праву половину буфера. Після того, як обидві перспективи сцени були відрендерені, отриманий фреймбуфер передається на пристрій WebXR для представлення користувачеві через його гарнітуру або інший відповідний пристрій відображення.

У той час як старіший API WebVR був розроблений виключно для підтримки віртуальної реальності (VR), WebXR забезпечує підтримку як VR, так і доповненої реальності (AR) в інтернеті. Підтримка функціональності AR додається модулем WebXR Augmented Reality.

Типовий XR-пристрій може мати 3 або 6 ступенів свободи і може мати або не мати зовнішній датчик положення. Обладнання також може включати акселерометр, барометр або інші датчики, які використовуються для визначення, коли користувач рухається в просторі, повертає голову тощо

ЗАВДАННЯ

1. Налаштувати середовище розробки Cursor та інструменти для роботи з Node.js (NVM, PNPM).

2. Встановити та налаштувати ngrok для створення HTTPS ендпойнту.

3. Створити базовий веб-проект з використанням Three.js та WebXR API.

4. Імплементувати відображення простого 3D об'єкту в середовищі доповненої реальності.

5. Дослідити базові методи і функції WebXR API.

6. Налаштувати remote debugging для тестування на мобільному пристрої.

ІНДИВІДУАЛЬНИЙ ВАРІАНТ ЗАВДАННЯ

10. Варіант 3D об'єкту: куб з кольором нижньої грані "червоний", верхньої "зелений", інших "синій"

ХІД ВИКОНАННЯ

Вміст файлу package.json

```
{
    "name": "webxr-project",
    "version": "1.0.0",
    "description": "WebXR demo project",
    "scripts": {
        "start": "concurrently \"pnpm build --watch=forever\" \"pnpm serve
.\"",
        "build": "esbuild main.ts --bundle --outfile=dist/main.js --
format=esm"
    },
    "devDependencies": {
        "@types/three": "^0.172.0",
        "@types/webxr": "^0.5.10",
        "concurrently": "^9.1.2",
        "esbuild": "^0.20.2",
        "serve": "^14.2.4",
        "typescript": "^5.3.3"
    },
    "packageManager":
"pnpm@10.5.2+sha512.da9dc28cd3ff40d0592188235ab25d3202add8a207afbedc682220e4a0029f
fbff4562102b9e6e46b4e3f9e8bd53e6d05de48544b0c57d4b0179e22c76d1199b",
    "dependencies": {
        "three": "^0.172.0"
    }
}
```

Вміст файлу main.ts

```
import * as THREE from 'three';
const MODE = 'immersive-ar';
async function activateXR(): Promise<void> {
    const canvas = document.createElement("canvas");
    document.body.appendChild(canvas);

    const gl = canvas.getContext("webgl2", {xrCompatible: true});
    if (!gl) throw new Error("WebGL not supported");

    // FIX THIS:
    const scene = new THREE.Scene();

    const redMaterial = new THREE.MeshBasicMaterial({ color: 0xff0000 }); //
red for bottom face
    const greenMaterial = new THREE.MeshBasicMaterial({ color: 0x00ff00 });
// green for top face
    const blueMaterial = new THREE.MeshBasicMaterial({ color: 0x0000ff });
// blue for other faces

    // initialize materials
    const materials = [
        blueMaterial, // front face
        blueMaterial, // back face
        greenMaterial, // top face
        redMaterial,   // bottom face
        blueMaterial, // left face
        blueMaterial  // right face
    ];
```

```
        const cube = new THREE.Mesh(new THREE.BoxGeometry(0.5, 0.5, 0.5),
materials);
        // set cube position
        cube.position.set(1, 0, 1);
        // add cube to scene
        scene.add(cube);

        const directionalLight = new THREE.DirectionalLight(0xffffff, 0.3);
        directionalLight.position.set(10, 15, 10);
        scene.add(directionalLight);

        const renderer = new THREE.WebGLRenderer({
            alpha: true,
            preserveDrawingBuffer: true,
            canvas: canvas,
            context: gl
        });
        renderer.autoClear = false;

        // FIX THIS:
        const camera = new THREE.PerspectiveCamera();;
        camera.matrixAutoUpdate = false;

        if (!navigator.xr) {
            throw new Error("WebXR is not supported by your browser");
        }

        try {
            const supported = await navigator.xr.isSessionSupported(MODE);
            if (!supported) {
                throw new Error(`${MODE} mode is not supported by your
browser/device`);
            }
        } catch (e) {
            throw new Error('Error checking WebXR support: ' + e);
        }

        const session = await navigator.xr.requestSession(
            MODE,
            {
                requiredFeatures: ['local']
            }
        );

        // FIX THIS:
        const baseLayer = new XRWebGLLayer(session, gl);
        session.updateRenderState({
            baseLayer
        });

        const referenceSpaceTypes: XRReferenceSpaceType[] = [
            'local',
            'local-floor',
            'bounded-floor',
            'unbounded',
            'viewer'
        ];

        let referenceSpace: XRReferenceSpace | null = null;

        // observe how reference space types and request reference space
        // are applied to the scene
        for (const spaceType of referenceSpaceTypes) {
```

```
            try {
                // request reference space
                // FIX THIS:
                referenceSpace = await session.requestReferenceSpace(spaceType);
                console.log('Reference space established:', spaceType);
                break;
            } catch(e) {
                console.log(e);
                console.log('Reference space failed:', spaceType);
                continue;
            }
        }

        if (!referenceSpace) {
            throw new Error('No reference space could be established');
        }

        // Create a render loop that allows us to draw on the AR view.
        const onXRFrame = (time: number, frame: XRFrame) => {
            // Queue up the next draw request.
            session.requestAnimationFrame(onXRFrame);

            const baseLayer = session.renderState.baseLayer;
            if (!baseLayer) return;

            // Bind the framebuffer and clear it
            gl.bindFramebuffer(gl.FRAMEBUFFER, baseLayer.framebuffer);
            gl.clearColor(0, 0, 0, 0);
            gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);

            const pose = frame.getViewerPose(referenceSpace);
            if (pose) {
                const view = pose.views[0];
                const viewport = baseLayer.getViewport(view);
                if (!viewport) return;
                renderer.setSize(viewport.width, viewport.height);

                // Update the camera with the XR view's transform and projection
                camera.matrix.fromArray(view.transform.matrix);
                camera.projectionMatrix.fromArray(view.projectionMatrix);
                camera.updateMatrixWorld(true);

                // Render the scene into the cleared framebuffer
                renderer.render(scene, camera);
            }
        };

        session.requestAnimationFrame(onXRFrame);
    }

    // Make the function available globally
    (window as any).activateXR = activateXR;
```

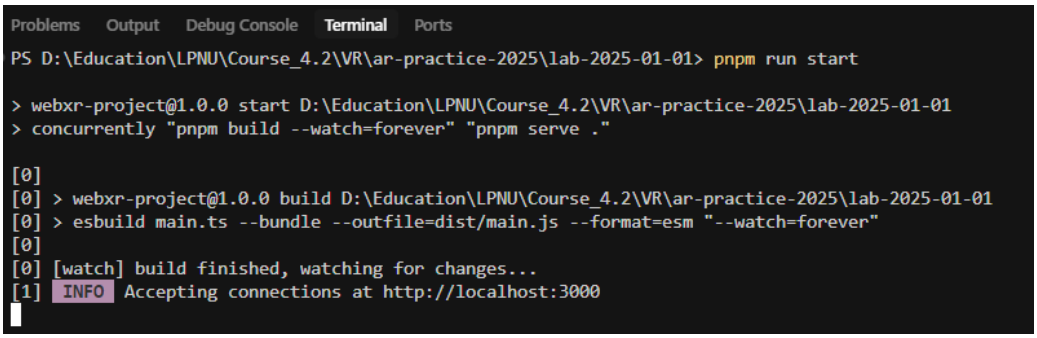

Рис. 1. Результат виконання команди pnpm run start

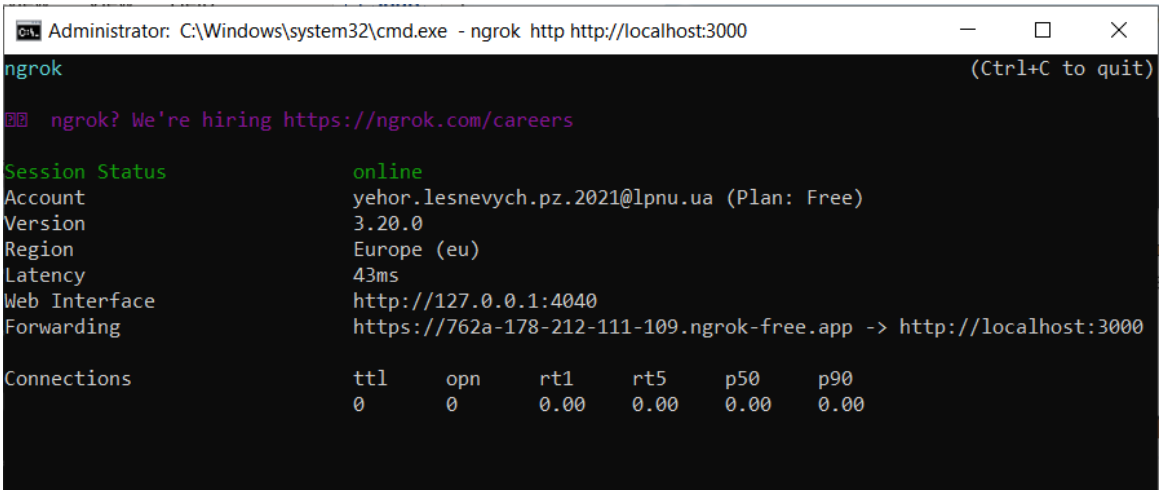

Рис. 3. Результат виконання команди ngrok http http://localhost:3000

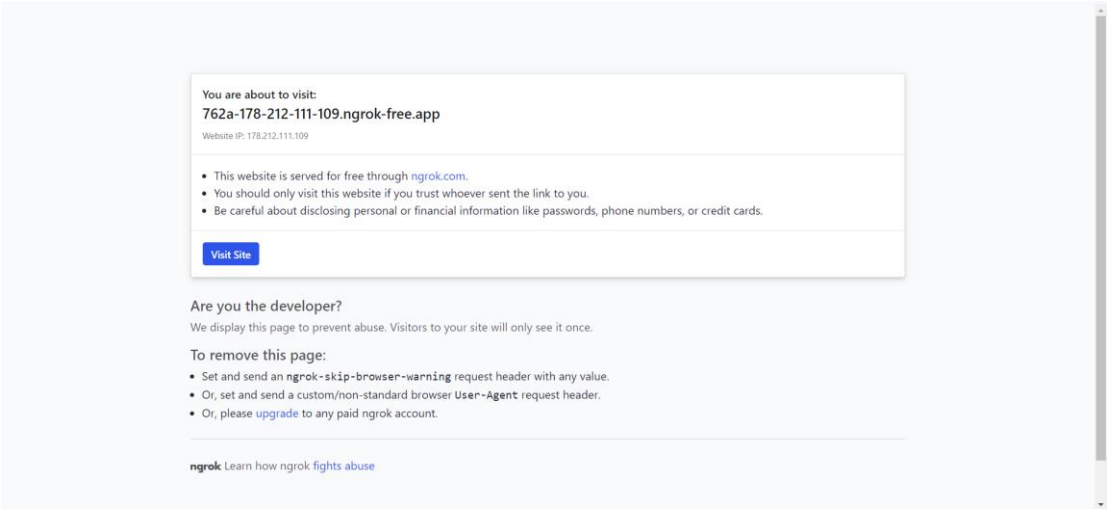

Рис. 4. Початкове вікно при переході за створеним за допомогою ngrok HTTPS посиланням

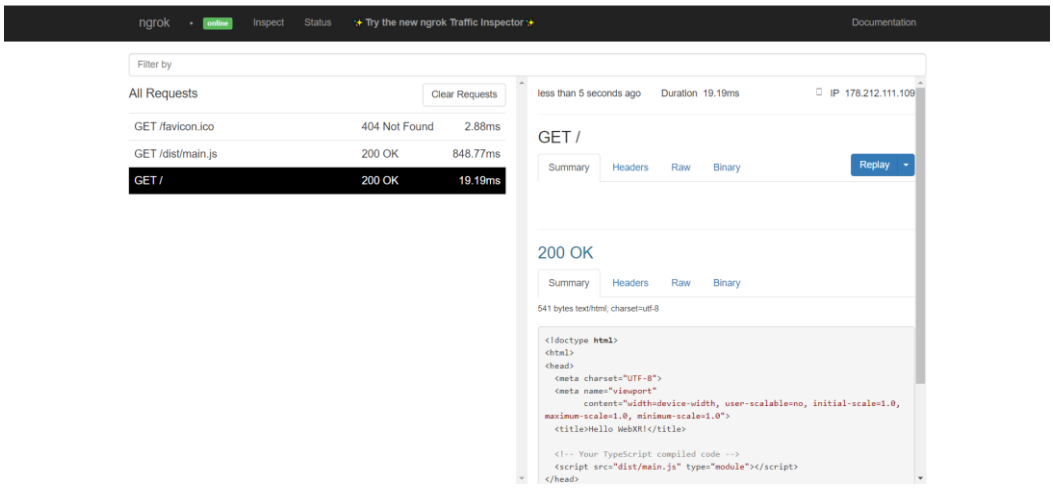

Рис. 5. Веб-інтерфейс ngrok

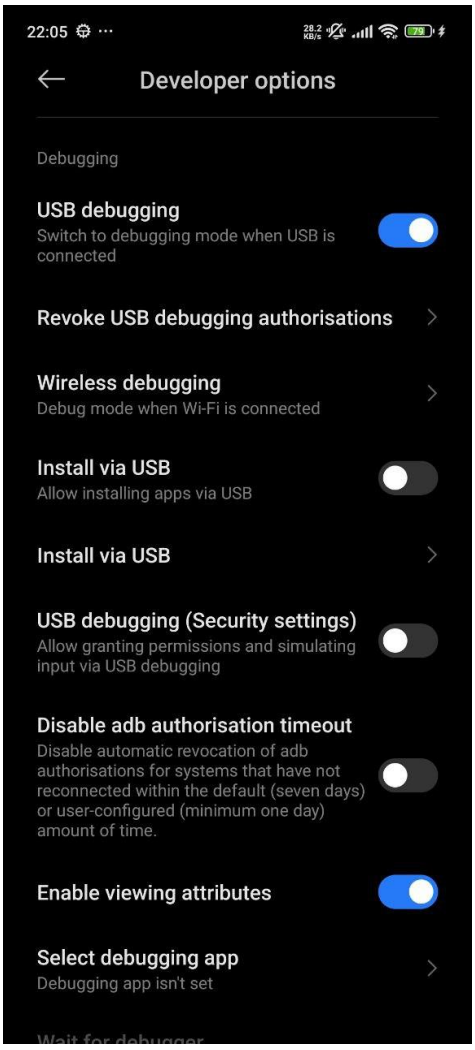

Рис. 6. Увімкнення функції відлагодження за допомогою USB на мобільному пристрої

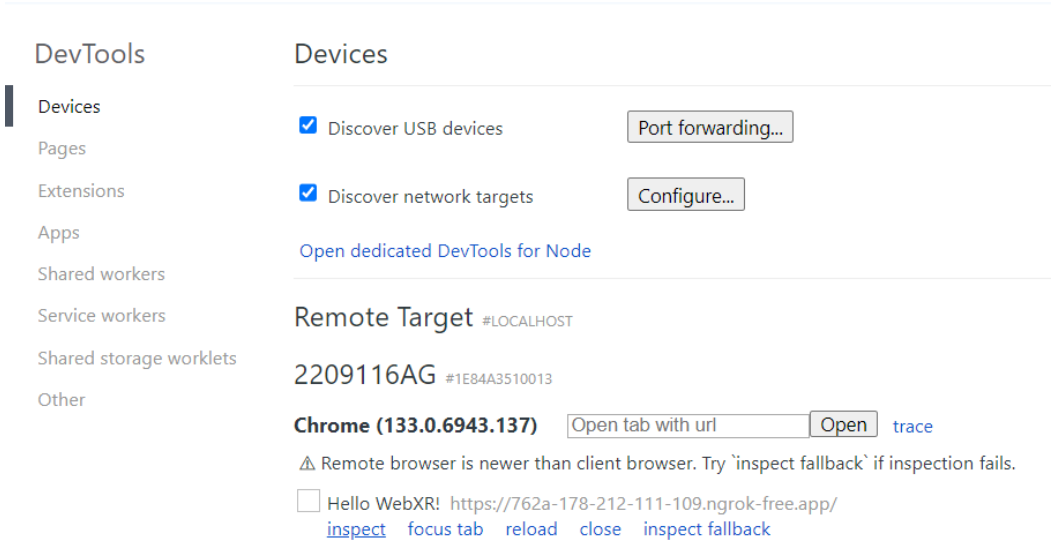

Рис. 7. Вікно під’єднання до мобільного пристрою для відлагодження

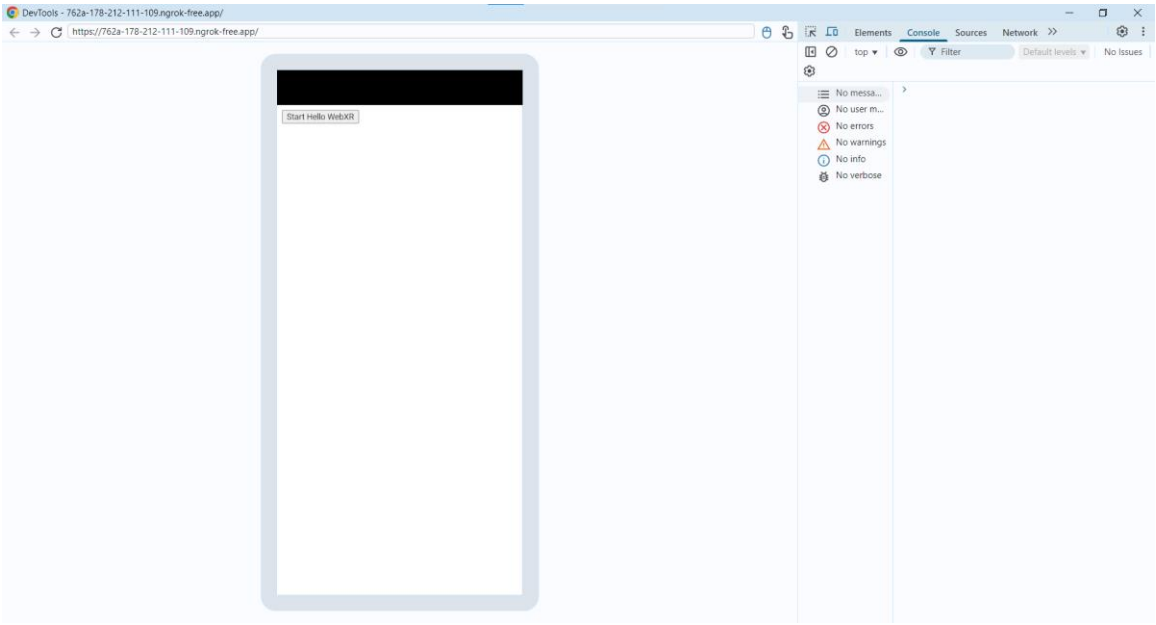

Рис. 8. Вікно сесії відлагодження веб-сайту на мобільному пристрої з комп’ютера

РЕЗУЛЬТАТИ

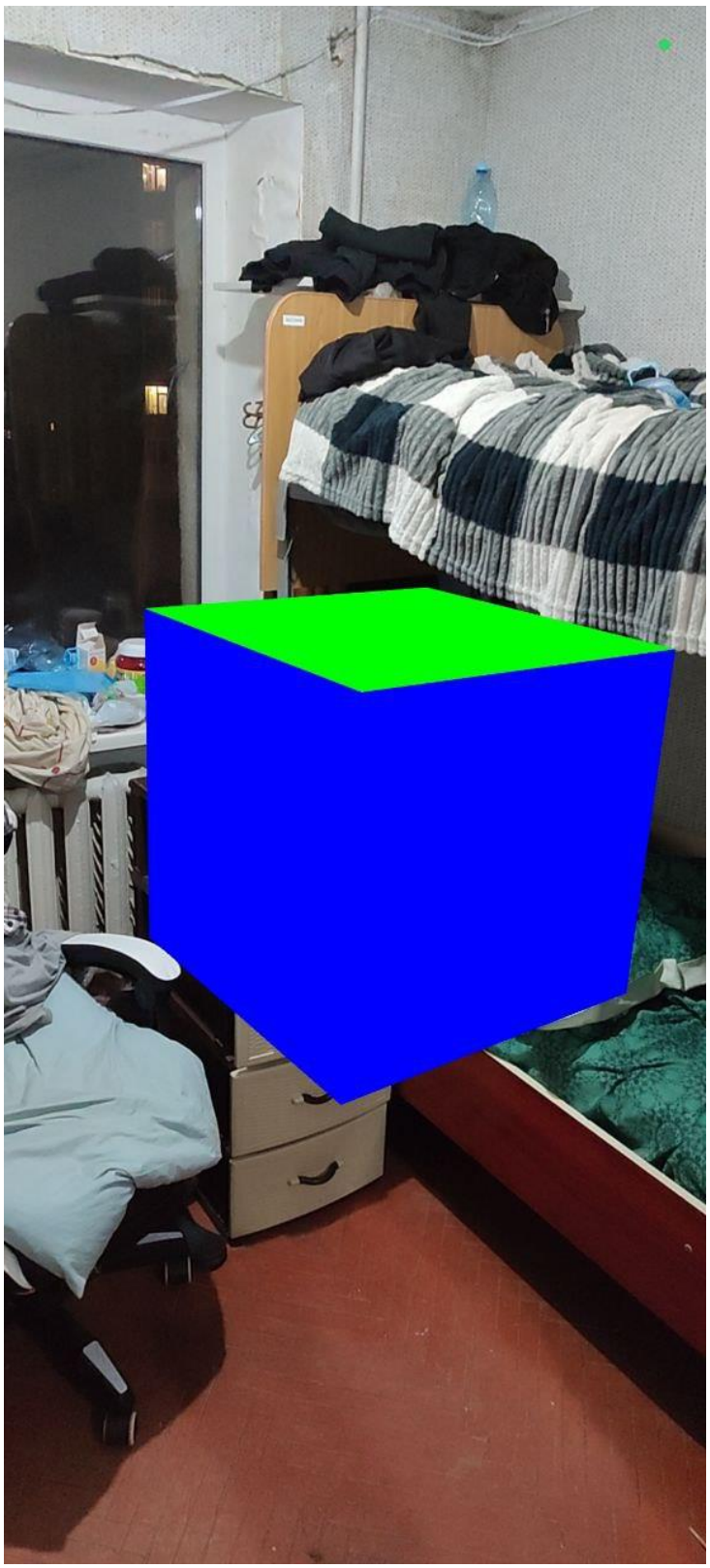

Рис. 8. Результат виконання програми – відмальований на сцені куб, вигляд зверху

Рис. 9. Результат виконання програми – відмальований на сцені куб, вигляд знизу

ВИСНОВКИ

Отже, під час виконання даної лабораторної роботи було налаштовано середовище розробки для роботи з веб API доповненої реальності. Ознайомився з інструментами веб-розробки, Typescript, Three.js, WebXR API. Було імплементовано створення простого 3D об'єкту в WebXR, а саме: куб з кольором нижньої грані "червоний", верхньої "зелений", інших "синій". Налаштовано remote debugging для мобільного пристрою.