

ОРГАНІЗАЦІЯ СЕРЕДОВИЩА РОЗРОБКИ ЗАСТОСУНКІВ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ З ВИКОРИСТАННЯМ ВЕБ API WEBXR

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторної роботи № 1
з дисципліни «Віртуальна реальність»
для студентів першого (бакалаврського) рівня вищої освіти спеціальності
121 “Інженерія програмного забезпечення”

Львів -- 2025

Організація середовища розробки застосунків віртуальної реальності з використанням веб API

WebXR: методичні вказівки до виконання лабораторної роботи №1 з дисципліни "Віртуальна реальність" для студентів першого (бакалаврського) рівня вищої освіти спеціальності 121 "Інженерія програмного забезпечення". Укл.: О.Є. Бауск. -- Львів: Видавництво Національного університету "Львівська політехніка", 2025. -- 16 с.

Укладач: Бауск О.Є., к.т.н., асистент кафедри ПЗ

Відповідальний за випуск: Федасюк Д.В., доктор техн. наук, професор

Рецензенти: Федасюк Д.В., доктор техн. наук, професор

Задорожний І.М., асистент кафедри ПЗ

Тема роботи: Основи роботи з WebXR -- 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 пристроя.

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 пристройів, рендерингом 3D сцени на обраному пристройі 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 пристройів виводу VR або AR Рендеринг 3D сцени на пристройі з відповідною частотою кадрів (Опціонально) відзеркалення 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 пристрой WebXR для представлення користувачеві через його гарнітуру або інший відповідний пристрой 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-пристрой 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.

Основні компоненти WebXR

1. WebXR Device API

- Забезпечує доступ до VR/AR пристройів

- Керує сесіями та просторовим відстеженням
- Обробляє введення від контролерів

2. Three.js інтеграція

- Популярна JavaScript бібліотека для 3D графіки
- Спрощує створення та рендеринг 3D об'єктів
- Надає готові компоненти для роботи з WebXR

Технічні вимоги

1. Для розробки:

- Сучасний веб-браузер з підтримкою WebXR
- Node.js та пакетний менеджер (npm/rpm)
- Інструменти збірки (наприклад, esbuild)
- HTTPS з'єднання (необхідне для WebXR)

2. Для мобільних пристрій:

- Android 8.0 або новіше
- ARCore-сумісний пристрій
- Chrome 79+ або інший сумісний браузер

WebXR та ARCore

WebXR Device API ([MDN Documentation](#)) надає стандартизований інтерфейс для доступу до VR та AR функціональності через веб-браузер. На Android пристроях WebXR працює поверх ARCore ([Google ARCore Documentation](#)) - платформи Google для створення AR досвіду.

ARCore забезпечує основні можливості, необхідні для AR:

- Відстеження руху (motion tracking) - дозволяє телефону розуміти своє положення відносно світу
- Розуміння навколишнього середовища (environmental understanding) - виявлення розміру та розташування поверхонь
- Оцінка освітлення (light estimation) - дозволяє правильно освітлювати віртуальні об'єкти

WebXR використовує ці можливості ARCore через абстрактний шар, що дозволяє писати крос-платформний код. Коли ви використовуєте WebXR API на Android пристрії, браузер автоматично взаємодіє з ARCore для забезпечення AR функціональності.

Стандарт WebXR забезпечує можливості доповненої реальності у веб-браузері. У Google Chrome WebXR використовує ARCore для забезпечення функціональності AR на підтримуваних пристроях Android.

При виборі між WebXR та ARCore врахуйте цільовий користувачкий досвід. Android-додаток з використанням ARCore забезпечить більш плавний користувачкий досвід, ніж відкриття веб-браузера для відображення WebXR-контенту. Аналогічно, веб-сторінка зможе розпочати безперебійну WebXR-сесію в браузері. Це дозволяє вашим користувачам залишатися у звичному середовищі.

Порівняння функцій При виборі розробки AR-досвіду з використанням ARCore або WebXR, майте на увазі, що не всі функції ARCore доступні у WebXR.

Наступна таблиця описує функції ARCore та їх відповідний статус у Chrome.

Функція ARCore	Статус у Chrome	Пояснення WebXR
Тест зіткнень	Випущено	WebXR Device API - Тестування зіткнень
Якорі	Випущено	Що таке якір
API глибини	Випущено	Визначення глибини WebXR
Виявлення площин	Тестування розробниками	Пояснення виявлення площин
Оцінка освітлення	Випущено	WebXR Device API - Оцінка освітлення
Доповнені обличчя	✗	✗
Доповнені зображення	✗	✗
Миттєве розміщення	✗	✗

Особливості розробки з WebXR

При розробці WebXR додатків важливо враховувати:

1. Безпека та дозволи

- WebXR вимагає HTTPS з'єднання
- Користувач повинен явно надати дозвіл на доступ до камери
- Сесія AR починається тільки після взаємодії користувача (наприклад, натискання кнопки)

2. Життєвий цикл AR сесії

- Перевірка підтримки AR функціональності
- Ініціалізація сесії
- Обробка подій входу/виходу з сесії
- Коректне завершення сесії

3. Оптимізація продуктивності

- Мінімізація використання ресурсів
- Ефективне управління 3D об'єктами
- Правильна обробка втрати відстеження

Детальніше про розробку можна дізнатись у:

- [WebXR Samples](#)
- [Google ARCore WebXR](#)
- [Three.js WebXR Documentation](#)

Хід роботи

1. Необхідні інструменти розробки

1.1. Установити середовище розробки Cursor: <https://www.cursor.com/>

1.2. Для управління версіями Node.js -- Установити NVM: <https://github.com/nvm-sh/nvm>

1.2.1 Версія Node.js для виконання лабораторних робіт - LTS 22.13.1.

1.2.2 У випадку проблеми зі зміною версії Node.js за допомогою NVM, наприклад неможливості перейти з system на LTS, реініціалізацію можна виконати наступним чином:

```
nvm deactivate && nvm unload && source ~/.nvm/nvm.sh && nvm use --lts
```

1.3. Установити Node.js, останню LTS версію, за допомогою NVM.

```
nvm install --lts
nvm use --lts
```

1.4. Для управління Node.js залежностями в проекті -- Установити PNPM: <https://www.npmjs.com/get-npm>

1.4.1. Варіант 1. Використовувати corepack для встановлення PNPM.

```
npm install --global corepack@latest
corepack enable pnpm
```

1.4.2. Варіант 2. Використовувати pnpm для встановлення PNPM.

```
npm install -g pnpm
```

1.5. Для створення HTTPS ендпоїнту онлайн -- зареєструватись в сервісі ngrok та встановити ngrok CLI: <https://ngrok.com/>

```
curl -sSL https://ngrok-agent.s3.amazonaws.com/ngrok.asc \
| sudo tee /etc/apt/trusted.gpg.d/ngrok.asc >/dev/null \
&& echo "deb https://ngrok-agent.s3.amazonaws.com buster main" \
| sudo tee /etc/apt/sources.list.d/ngrok.list \
&& sudo apt update \
&& sudo apt install ngrok

ngrok config add-authtoken <YOUR_AUTH_TOKEN>
# Test ngrok
ngrok http localhost:3000
```

1.6 Склонувати локально цей репозиторій і відкрити в Cursor перше завдання ЛР-01-01 ar-practice-2025/lab-2025-01-01 .

2. Нотатки до виконання.

2.1. NVM - Node Version Manager, підсистема для управління версіями Node.js. NVM необхідно для того, щоб можна було легко перемикатися між різними версіями Node.js. В рамках цього та інших курсів та лабораторних робіт може виникнути потреба мати доступ до різних версій Node.js. NVM дозволяє легко перемикатися між різними версіями рантайму. Наша робоча версія Node.js в рамках цього курсу - LTS 22.13.1.

2.2. PNPM - це аналог npm, ефективний пакет-менеджер, який використовується для управління залежностями в проектах. PNPM є альтернативою npm, яка має більш швидкі та ефективні алгоритми для керування залежностями. За бажанням можна використовувати інші пакет-менеджери, такі як Yarn, npm, vite тощо, але методичні вказівки та завдання будуть розроблені з урахуванням PNPM.

2.3. Cursor - це IDE, яка інтегрує VSCode та AI, і дозволяє використовувати промпти штучного інтелекту для написання коду. Cursor є альтернативою VSCode, методичні вказівки не роблять особливих вимог до використання Cursor, але він може бути корисним для написання коду. Як альтернатива може використовуватись VSCode.

3. Виконання завдання.

3.1. Відкрити в Cursor проект ar-practice-2025/lab-2025-01-01 .

Ця директорія має бути робочою для наступних команд, якщо не вказано інше.

3.2. Переконатись, що версії інструментів встановлені і відповідають очікуваним. В терміналі виконати команди:

```
# Node.js
node --version
# v22.13.1
# У випадку проблеми з версією:
nvm install --lts
nvm use --lts

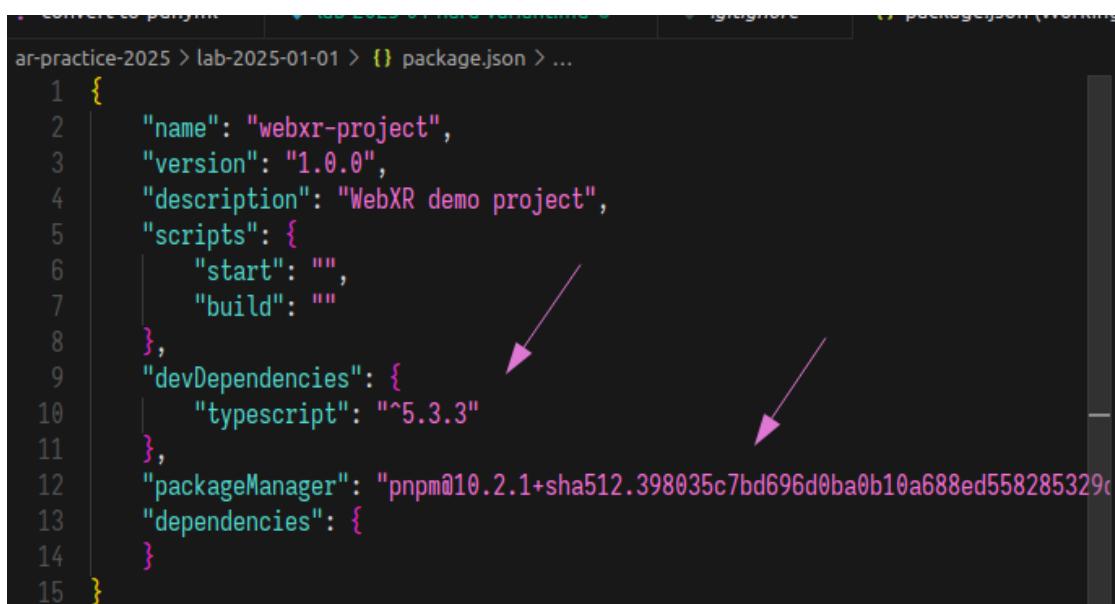
# PNPM
pnpm --version
# v10.2.1
# У випадку проблеми з версією:
corepack enable pnpm
corepack use pnpm@latest-10
```

3.3. Команди в п 3.2. прописують в файлі конфігурації проекта (package.json) версію пакетного менеджера (ми використовуємо PNPM).

Також в файлі конфігурації прописані всі залежності, які потрібно встановити для розробки проекту. Залежності встановлюються командою pnpm install . Існують залежності часу виконання, які потрібні для production версії проекту, тобто для виконання коду проекту, і залежності часу розробки, які потрібні тільки для розробки проекту. Приклад залежностей часу розробки -- це Typescript компілятор, який

транслює код з TypeScript в JavaScript що виконується в браузері (і тому TypeScript не потрібний для виконання коду проекту), і декларації типів для TypeScript, які потрібні для правильної компіляції коду і також не використовуються в готовому скомпільованому коді.

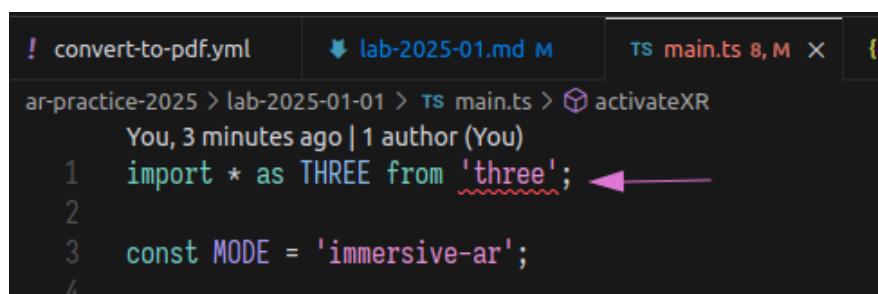
Слід відкрити файл `package.json` і переконатись, що corepack коректно встановив "packageManager" та що ми маємо в залежностях часу розробки правильну версію TypeScript "devDependencies"->"typescript" :



```
ar-practice-2025 > lab-2025-01-01 > {} package.json > ...
1 {
2   "name": "webxr-project",
3   "version": "1.0.0",
4   "description": "WebXR demo project",
5   "scripts": {
6     "start": "",
7     "build": ""
8   },
9   "devDependencies": {
10     "typescript": "^5.3.3"
11   },
12   "packageManager": "pnpm@10.2.1+sha512.398035c7bd696d0ba0b10a688ed558285329c
13   "dependencies": {
14   }
15 }
```

3.4. Відкрити файл `main.ts` і переконатись, що він не містить помилок.

3.4.1. Перша помилка, яку ви маєте побачити -- це помилка з імпортом Three.js:



```
! convert-to-pdf.yml      ↓ lab-2025-01.md M      ts main.ts 8, M × {}
ar-practice-2025 > lab-2025-01-01 > ts main.ts > activateXR
You, 3 minutes ago | 1 author (You)
1 import * as THREE from 'three'; ←
2
3 const MODE = 'immersive-ar';
4
```

Вирішить проблеми з імпортом і типізацією Three.js та WebXR API. Встановіть наступні залежності:

```
pnpm install three@0.172.0
pnpm install -D @types/three@0.172.0
pnpm install -D @types/webxr
```

3.5. Встановити залежності для запуску проекту.

В роботі використовуються наступні команди, які треба запускати в терміналі, знаходячись в робочій директорії проекту, `ar-practice-2025/lab-2025-01-01`.

Зараз вони наведені тут просто для ознайомлення, запускати їх не потрібно.

```
pnpm run start # запуск проекту для розробки з перебудуванням при зміні файлів
pnpm run build # білд проекту
```

Необхідно інсталювати залежності `esbuild` та `serve` для того, щоб робити білд проекту та запустити локальний сервер для розробки та відлагодження:

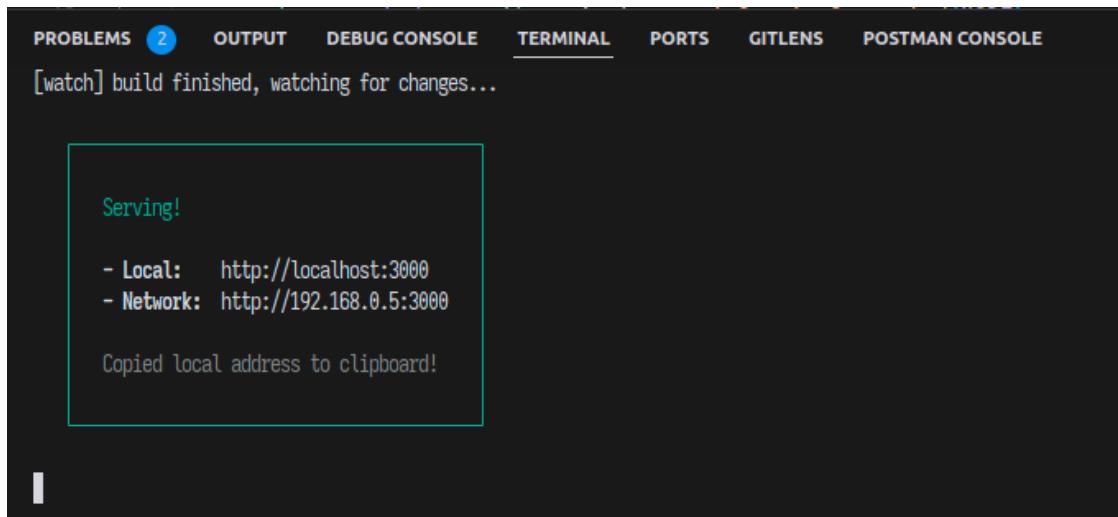
```
pnpm install -D esbuild serve
```

Перевірте, чи працює команда `pnpm run start`. Успішне виконання команди має відкрити локальний сервер на порту 3000.

- Увага! Команди треба виконувати в терміналі, знаходячись в робочій директорії проекту, `arg-practice-2025/lab-2025-01-01`.*

```
pnpm run start
```

Успішне виконання команди має відкрити локальний сервер на порту 3000. Оскільки проект ще треба імплементувати, команда може не працювати і видати помилку, на цьому кроці це нормально.



3.6. Використати ngrok для створення HTTPS ендпоїнту, доступного з мобільного пристрою.

```
ngrok http http://localhost:3000
```

Команда повинна відкрити ендпоїнт, який можна буде використовувати для доступу до проекту з мобільного пристрою.

```

ngrok

Found a bug? Let us know: https://github.com/ngrok/ngrok

Session Status           online
Account                  bauskas@gmail.com (Plan: Free)
Version                 3.19.1
Region                  Europe (eu)
Latency                22ms
Web Interface          http://127.0.0.1:4040
Forwarding              https://a952-92-253-236-134.ngrok-free.app -> http://localhost:3000

Connections             ttl     opn      rt1      rt5      p50      p90
                        2        0       0.03    0.01    6.14    6.28

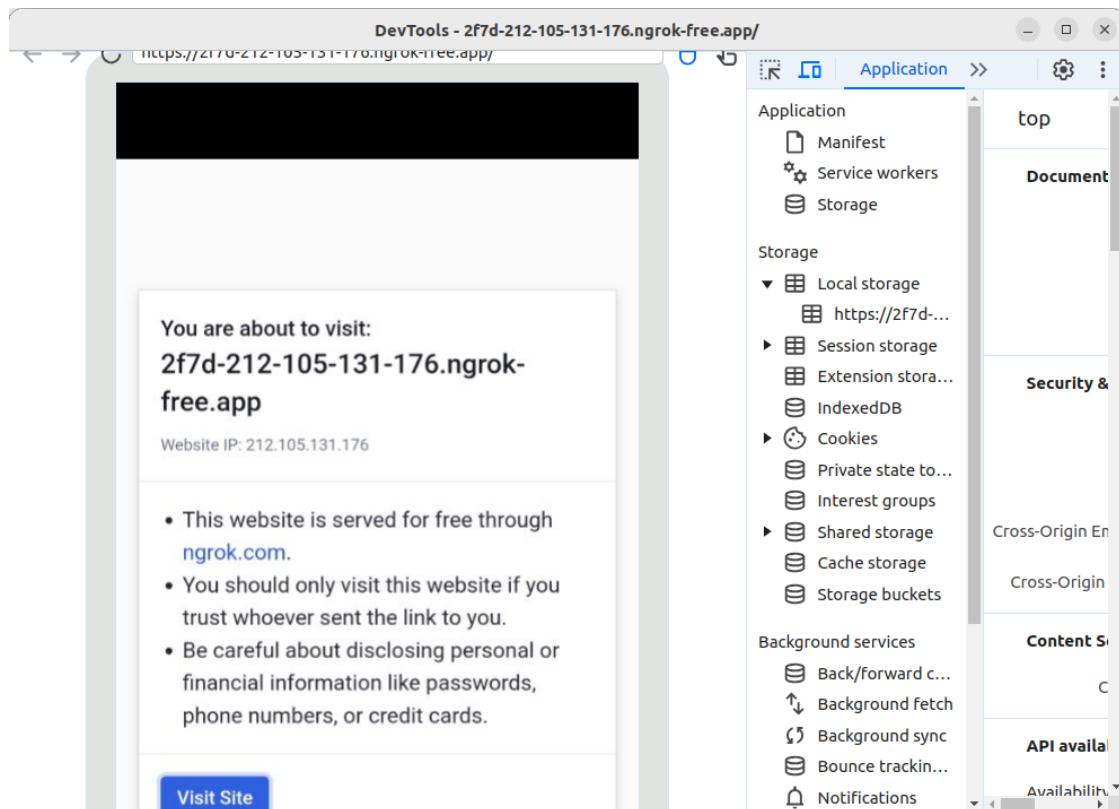
HTTP Requests
-----
01:46:01.837 EET GET /favicon.ico           404 Not Found
01:46:01.661 EET GET /dist/main.js          200 OK
01:46:01.872 EET GET /dist/main.js          200 OK
01:46:01.559 EET GET /                   200 OK

```

На скріншоті вище показано, що ngrok відкрив ендпойнт (строка `https://a952-92-253-236-134.ngrok-free.app` після слова `Forwarding`). Скопіюйте цю строку в браузер. Зараз проект ще не в робочому стані, але після виконання роботи за цією адресою буде доступний доступ до працюючого локально сервера з вашого мобільного пристрою.

3.7. Відкрити ендпойнт в браузері на мобільному пристрої

3.7.1 Відкрити отриманий вище від ngrok HTTPS ендпойнт в браузері на мобільному пристрої і переконатись, що він відкривається.



Зараз, оскільки проект ще не в робочому стані, треба просто переконатись що сторінка як на скріншоті вище відкривається.

3.8. Імплементувати створення простого 3D об'єкту в WebXR.

3.8.1. Дослідіть приклад створення простого 3D об'єкту в WebXR в документації. Задача Лабораторної роботи -- відтворити приклад з документації до розділу "Run Hello WebXR", на який подано посилання нижче.

https://developers.google.com/ar/develop/webxr/hello-webxr#run_hello_webxr

3.8.2. Відкрийте файл `main.ts`, дослідіть існуючий код. Це непрацюючий шаблон коду, який треба наповнити імплементацією відповідно до прикладу вище. Зверніть увагу на наступні речі:

- Структура коду інша, ніж у прикладі. Проаналізуйте приклад крок за кроком і використайте відповідні фрагменти в `main.ts` там, де це потрібно.
- Ваш код використовує Typescript, код з документації потрібно оздобити відповідними типами.

3.8.3. Імплементуйте код відповідно до прикладу з документації.

3.8.3.1. Знайдіть в коді `main.ts` наступний фрагмент коду:

```
// FIX THIS:  
const scene = null;
```

Імплементуйте правильну ініціалізацію сцени (є в документації по посиланню вище). Поміняйте код вище на наступний:

```
const scene = new THREE.Scene();
```

3.8.3.2. Знайдіть в коді `main.ts` наступний фрагмент коду:

```
// FIX THIS:  
const camera = null;
```

Імплементуйте правильну ініціалізацію камери (є в документації по посиланню вище). Поміняйте код вище на наступний:

```
const camera = new THREE.PerspectiveCamera();
```

3.8.3.3. Знайдіть в коді `main.ts` наступний фрагмент коду:

```
// FIX THIS:  
const baseLayer = null;
```

Імплементуйте правильну ініціалізацію базового шару (є в документації по посиланню вище). Поміняйте код вище на наступний:

```
const baseLayer = new XRWebGLLayer(session, gl);
```

3.8.3.4. Знайдіть в коді `main.ts` наступний фрагмент коду:

```
// request reference space  
referenceSpace = null;
```

Імплементуйте правильну ініціалізацію простору (є в документації по посиланню вище). Поміняйте код вище на наступний:

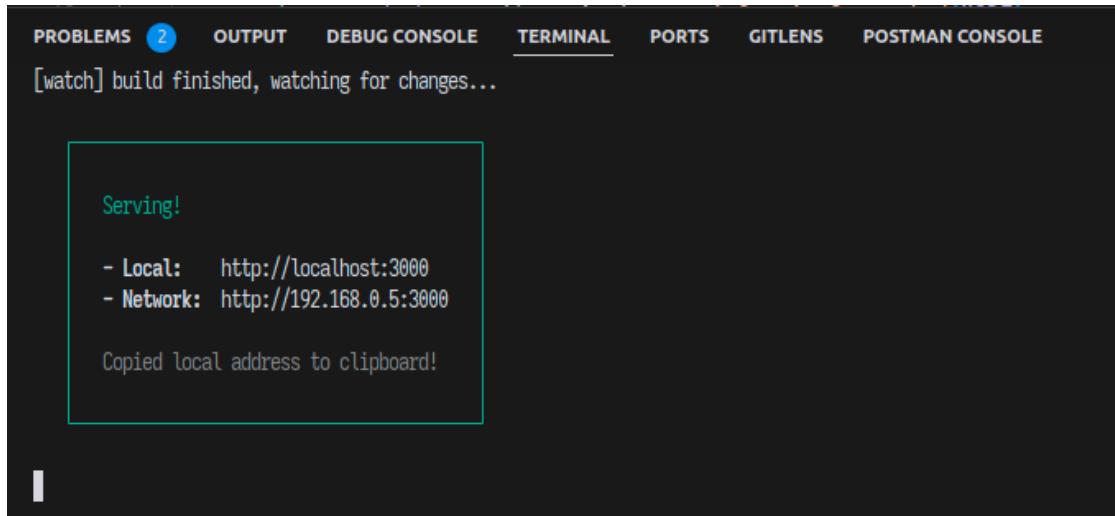
```
referenceSpace = await session.requestReferenceSpace(spaceType);
```

3.9. Запустіть проект і переконайтесь, що він працює.

3.9.1. Виконайте команду побудови проекту для розробки:

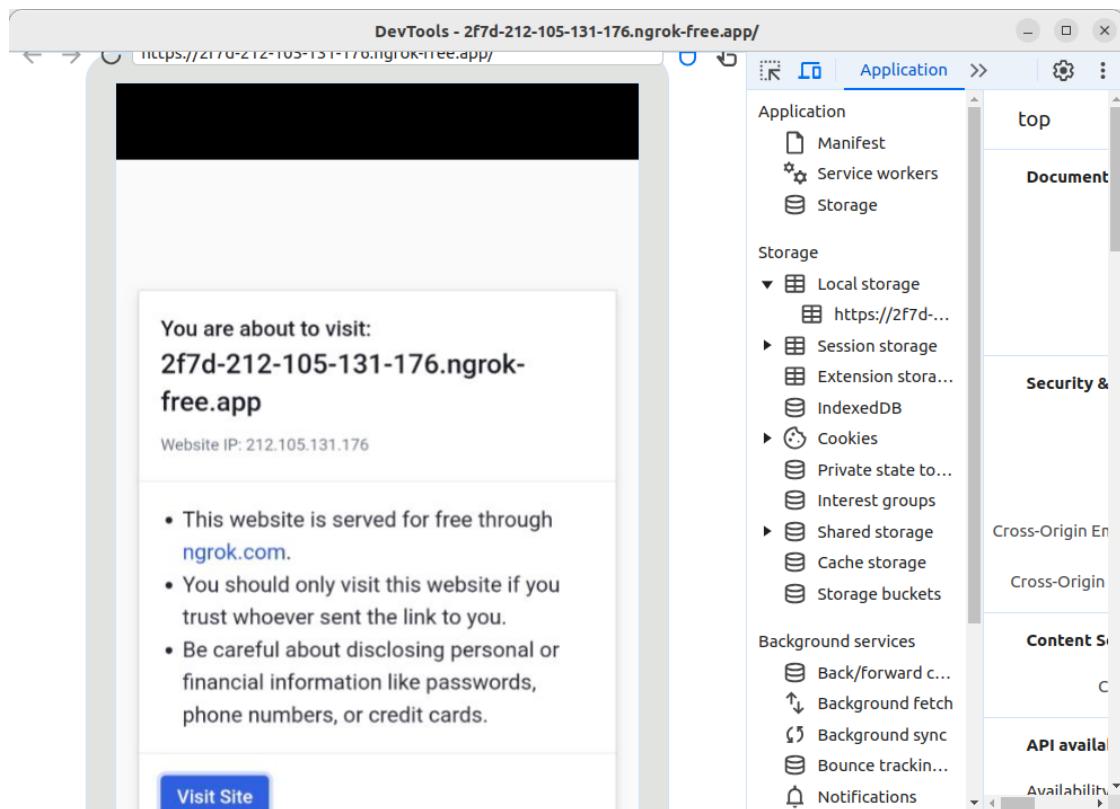
```
pnpm run start
```

Перевірте, чи запустився локальний сервер на порту 3000, як на скріншоті нижче.



3.9.2. Якщо ngrok не запустився, запустіть його командою з пункту 3.6.

3.9.3. Відкрийте ендпойнт в браузері на мобільному пристрої і переконайтесь, що він відкривається.



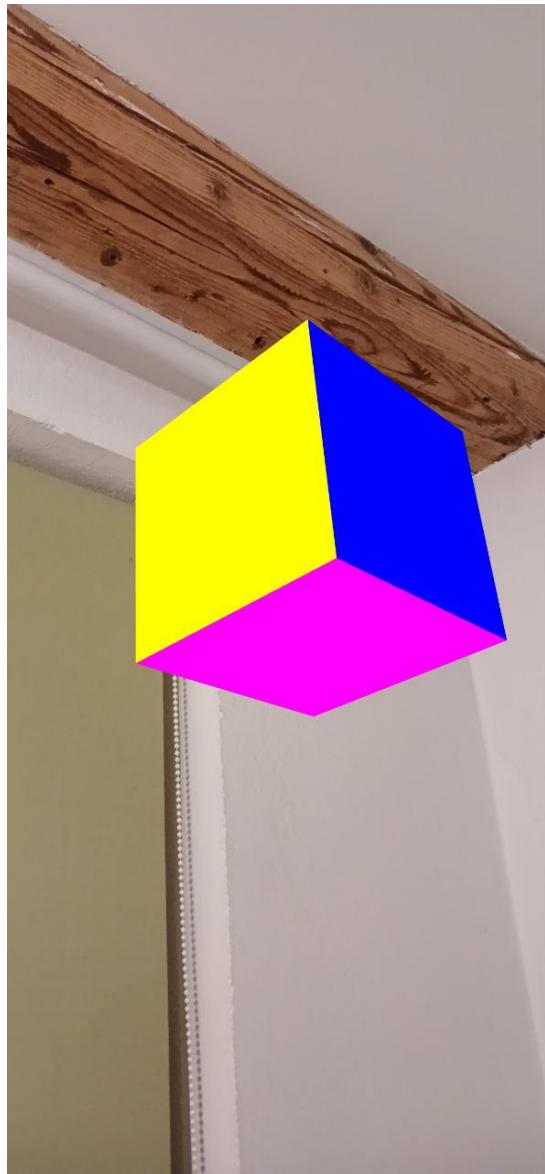
Нажміть "Visit Site" і переконайтесь, що відкривається сторінка з проектом.

3.9.4. У вас має відкритися сторінка проекту:

[Start Hello WebXR](#)

Нажміть кнопку "Start Hello WebXR", надайте всі необхідні дозволи для доступу до камери і т.д.

Ви можете побачити відео з камери телефону. Покрутіть телефоном і знайдіть в просторі 3D об'єкт, який був створений в коді. Дослідіть, як він відображається в просторі.



УМОВА ЗАВДАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

1. Налаштuvати 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 дополненої реальності.
5. Дослідити базові методи і функції WebXR API.
6. Налаштувати remote debugging для тестування на мобільному пристрої.

ІНДІВІДУАЛЬНІ ВАРИАНТИ ЗАВДАННЯ

1. Варіант довільного вибору: як додаткове завдання на додані бали, виконати налаштування remote debugging для тестування на мобільному пристрої.
2. Варіант розмірів 3D об'єкту: куб 1.5x1.5x1.5 м.

3. Варіант розмірів 3D об'єкту: куб 1x1x1 м.
4. Варіант розмірів 3D об'єкту: куб 0.5x0.5x0.5 м.
5. Варіант 3D об'єкту: куб з кольором нижньої грані "зелений", верхньої "червоний", інших "синій".
6. Варіант 3D об'єкту: куб з кольором нижньої грані "червоний", верхньої "зелений", інших "синій".
7. Варіант 3D об'єкту: куб з кольором нижньої грані "синій", верхньої "червоний", інших "зелений".
8. Варіант 3D об'єкту: куб з кольором нижньої грані "синій", верхньої "зелений", інших "червоний".
9. Варіант 3D об'єкту: куб з кольором нижньої грані "червоний", верхньої "синій", інших "зелений".
10. Варіант 3D об'єкту: куб з кольором нижньої грані "червоний", верхньої "зелений", інших "синій".
11. Варіант 3D об'єкту: куб з кольором нижньої грані "червоний", верхньої "синій", інших "фіолетовий".
12. Варіант 3D об'єкту: куб з кольором нижньої грані "синій", верхньої "червоний", інших "фіолетовий".
13. Варіант 3D об'єкту: куб з кольором нижньої грані "синій", верхньої "фіолетовий", інших "червоний".
14. Варіант 3D об'єкту: куб з кольором нижньої грані "фіолетовий", верхньої "синій", інших "червоний".
15. Варіант 3D об'єкту: куб з кольором нижньої грані "фіолетовий", верхньої "червоний", інших "синій".
16. Варіант 3D об'єкту: куб з кольором нижньої грані "фіолетовий", верхньої "синій", інших "червоний".

ЗМІСТ ЗВІТУ

1. Тема та мета роботи.
2. Теоретичні відомості.
3. Постановка завдання.
4. Хід виконання, який включатиме скріншоти процесу білду та запуску проекту, скріншоти імплементації 3D сцени відповідно до прикладу з документації.
5. Результати виконання у вигляді скріншотів з мобільного пристроя.
6. Висновки.

КОНТРОЛЬНІ ПИТАННЯ

1. Що таке WebXR API?
2. Які основні компоненти необхідні для розробки WebXR додатків?
3. Як налаштовувати середовище розробки для WebXR?
4. Що таке remote debugging і для чого він використовується в контексті WebXR?
5. Які інструменти потрібні для тестування WebXR додатків на мобільних пристроях?
6. Яка роль ngrok у розробці веб-додатків для мобільних пристройів?

7. Які основні етапи створення простого 3D об'єкту в WebXR?
8. Як працює Three.js з WebXR API?
9. Опишіть свій досвід з використанням TypeScript в лабораторній роботі.
10. Які вимоги до мобільних пристройів для роботи з WebXR?
11. Що таке AR сесія в контексті WebXR і як вона ініціалізується?
12. Як забезпечити кросбраузерну підтримку WebXR додатків?
13. Які інструменти для збірки проекту використовуються в даній лабораторній роботі?
14. Які основні проблеми можуть виникнути при розробці WebXR додатків та як їх вирішувати?
15. Як оптимізувати продуктивність WebXR додатків?

СПИСОК ЛІТЕРАТУРИ

1. WebXR Device API [Електронний ресурс]. -- Режим доступу: <https://www.w3.org/TR/webxr/>
2. WebXR Samples [Електронний ресурс]. -- Режим доступу: <https://immersive-web.github.io/webxr-samples/>
3. Google ARCore WebXR [Електронний ресурс]. -- Режим доступу:
<https://developers.google.com/ar/develop/webxr/hello-webxr>
4. Three.js WebXR Documentation [Електронний ресурс]. -- Режим доступу: <https://threejs.org/docs/?q=webxr#api/en/renderers/webxr/WebXRManager>
5. MDN WebXR Device API Documentation [Електронний ресурс]. -- Режим доступу:
https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API
6. Google ARCore Documentation [Електронний ресурс]. -- Режим доступу:
<https://developers.google.com/ar>
7. Parisi T. Learning Virtual Reality: Developing Immersive Experiences and Applications for Desktop, Web, and Mobile. -- Amazon Publishing, 2015. -- 172 с.