МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ
Фізико-математичний факультет
Кафедра інформатики та прикладної математики

«Допущено до захисту»

Завідувач кафедри

_____ Соловйов В. М.

«___» _____ 2020 р.

Реєстраційний № _____

«___» _____ 2020 р.

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІЗ ДОПОВНЕНОЮ РЕАЛЬНІСТЮ ДЛЯ WEB

Кваліфікаційна робота студента групи І-16
ступінь вищої освіти «бакалавр»
спеціальності 014.09 Середня освіта (інформатика)
Шепілєва Дмитра Сергійовича

Керівник доктор педагогічних наук, професор
Семеріков Сергій Олексійович

Оцінка:
Національна шкала _____
Шкала ECTS _____ Кількість балів _____
Голова ЕК _____ _____
(підпис) (прізвище, ініціали)

Члени ЕК _____ _____
(підпис) (прізвище, ініціали)

_____ _____
(підпис) (прізвище, ініціали)

_____ _____
(підпис) (прізвище, ініціали)

_____ _____
(підпис) (прізвище, ініціали)

Кривий Ріг – 2020

ЗМІСТ

ВСТУП

Актуальність теми. Доповнена реальність (augmented reality, також відома як mixed reality) є сьогодні досить популярною технологією, що поступово набуває поширення в системі освіти. Основним напрямом застосуваннями доповненої реальності в освіті відносяться є забезпечення наочності навчання шляхом візуалізації комп'ютерних моделей об'єктів та систем за допомогою пристроїв загального (комп'ютер, обладнаний убудованою або зовнішньою веб-камерою), спеціального (окуляри доповненої реальності, шоломи віртуальної реальності) призначення та пристованих пристроїв (мобільні телефони).

Саме останній клас пристроїв є найбільш поширеним серед всіх учасників освітнього процесу. Станом на травень 2020 року в Україні нараховується 8,3 млн. користувачів мобільних ігор, 26,2 % яких – у віці 18-24 роки [17]. За даними 2019 року, українські мобільні користувачі у віці 16-25 років найчастіше використовують соціальні мережі (92 %), засоби для обміну повідомленнями (66 %), відеозастосунки (64 %) та мобільні ігри (50 %) [25]. Порівняння останніх двох джерел надає можливість зробити висновок про те, що в Україні від 16 до 16,6 млн. мобільних користувачів, що є учнями старших класів, студентами всіх рівнів вищої освіти або особами студентського віку. За даними Держстату України, на початок 2019/2020 року в коледжах, технікумах, училищах, університетах, академіях та інститутах навчалось 1,440 млн. студентів [32], при цьому у віці 16-25 років перебувало 4,163 млн. осіб [33, с. 26].

Порівняння наведених статистичних даних вказує на те, що кожна особа студентського віку має у середньому 2 номери мобільних пристроїв, прив'язаних до одного з поширених магазинів мобільних додатків: це може бути як один мобільний пристрій із двома SIM-картами, так й два окремих мобільних телефони. Станом на квітень 2020 року, на 82,1 % мобільних пристроїв в Україні встановлена операційна система Android [18], а провідним

веб-браузером є Chrome (використовується на 72,9 % пристроїв) [15]. Ураховуючи, що третій за популярністю веб-браузер – Opera – побудований на ядрі Chrome [10], частка Chrome досягає 80 %.

Незважаючи на наявність окремих клієнтів для соціальних мереж, обміну повідомленнями, перегляду відео та мобільних ігор, веб-браузер є універсальним засобом для виконання відповідних задач та швидкого створення нових програм для роботи у веб-середовищі. Головною перевагою веб-клієнтів є їх універсальність (версії Chrome існують для всіх мобільних платформ), а головним недоліком донедавна була низька швидкість виконання програм. Поява у березні 2017 року WebAssembly надала можливість не лише суттєво прискорити виконання веб-програм, а й перенести у веб-середовище програмне забезпечення, створене для інших платформ.

Ключовим для створення програмного забезпечення із доповненою реальністю є доступ до якісних бібліотек машинного зору, розпізнавання та відслідковування об’єктів, таких як ARToolKit та її веб-версії JSARToolKit5 [4], застосування якої надає можливість розробки програмного забезпечення із доповненою реальністю для Web. Водночас досвід спільного застосування JSARToolKit5 та WebGL виявив, що:

– маркери, що використовує JSARToolKit5, є двовимірним різновидом штрих-коду, що має низький рівень наочності;

– візуалізація об’єктів за допомогою WebGL потребує глибокого розуміння принципів роботи 3D-графіки та значних обсягів програмного коду.

Таким чином, дані засоби розробки доповненої реальності для Web, призначені для підвищення рівня наочності навчання, самі є далекими від наочності та доступності, що породжує *проблему дослідження* – добір та апробація засобів розробки програмного забезпечення із доповненою реальністю для Web для початківців, що володіють основами веб-розробки: учнів ліцеїв та студентів молодших курсів інформатичних спеціальностей.

Мета дослідження – розробити окремі елементи методики навчання розробки програмного забезпечення із доповненою реальністю для Web.

Для досягнення мети дослідження були поставлені такі **завдання**:

1. Виконати порівняльний аналіз засобів розробки доповненої реальності для Web з метою добору засобів, доступних для початківців.

2. Схарактеризувати дібрані засоби та розробити інструкції з їх налаштування та використання.

3. Розробити прототипи програмного забезпечення із маркерною та безмаркерною доповненою реальністю для Web.

Об'єкт дослідження – методи розробки програмного забезпечення із доповненою реальністю для Web.

Предмет дослідження – методика навчання початківців розробки програмного забезпечення із доповненою реальністю для Web.

Методи дослідження: *аналіз* джерел та програмного забезпечення з метою визначення стану розв'язання проблеми дослідження та добору засобів розробки доповненої реальності для Web, *синтез* методичних вказівок з налаштування та використання дібраних засобів, *методи програмної інженерії* (проектування, розробка, тестування) для досягнення мети дослідження.

Наукова новизна результатів дослідження полягає в тому, що розроблено окремі елементи методики навчання розробки програмного забезпечення із доповненою реальністю для Web.

Практичне значення одержаних результатів полягає в тому, що розроблено прототип програмного забезпечення, що надає можливість використання наочних (фотографічних та рисункових) маркерів для підготовки профорієнтаційних веб-квестів.

Структура роботи. Робота складається із вступу, трьох розділів, висновків, списку використаних джерел (34 найменування).

РОЗДІЛ 1
ОГЛЯД ЗАСОБІВ РОЗРОБКИ ДОПОВНЕНОЇ РЕАЛЬНОСТІ ДЛЯ WEB

1.1 Доповнена реальність: поняття та апаратне забезпечення для Web

Програмувати доповнену реальність (AR – augmented reality) – інноваційно (модно, цікаво, корисно та ін.) останні 60 років [22], а її використання у веб-середовищі зумовлено виключно поточним станом розвитку технологій. Для початку роботи з нею необхідно мати лише AR-сумісний браузер, такий як Firefox або Chrome, та текстовий редактор (типу mcedit або Sublime). Базові знання основ веб-розробки (HTML, CSS та JavaScript) є необхідними для початківця, а досвід роботи з веб-API та GitHub стануть у пригоді.

Надалі під AR будемо розуміти здатність пристрою, зокрема мобільного пристрою або веб-браузера, відстежувати зображення або відображати 3D-об’єкт поверх цього зображення. Головна ідея AR полягає в тому, щоб відобразити комп’ютерну модель у реальному часі та реальному просторі з метою взаємодії між користувачем у реальному просторі та 3D-моделі у віртуальному (рис. 1.1).

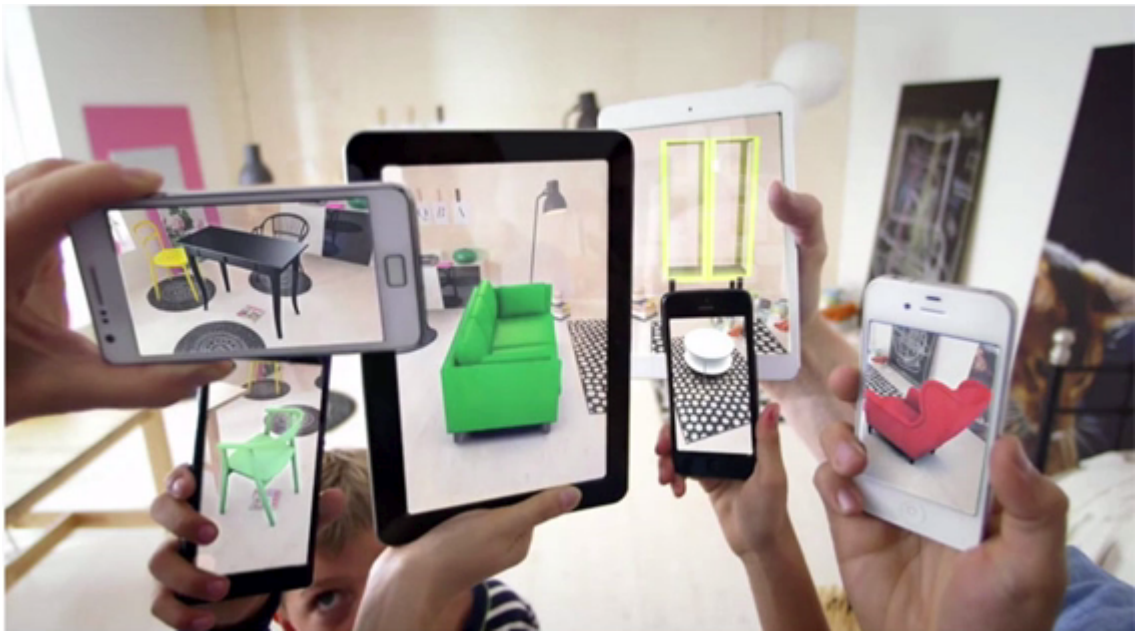

Рис. 1.1. Приклад роботи засобів AR на різних типах мобільних пристроїв

AR може бути як маркерним, так й безмаркерним. У маркерній AR пристрій відстежує 2D-маркер: коли він знаходиться, на ньому фактично відображається 3D-об’єкт (рис. 1.2). У безмаркерному варіанті пристрій буде шукати плоску поверхню (стіл, підлогу тощо), і розташовуватимемо 3D-об’єкт на ній.

Рис. 1.2. Приклад роботи маркерної AR

Використовуючи камеру пристрою, AR надає можливості відображення комп’ютерно згенерованих об’єктів в ігрових, маркетингових та інших програмах – наприклад, для розстановки меблів у вітальні або примірки одягу перед їх покупкою. Це дійсно велика можливість для бізнесу – показати, як виглядає продукція, перш ніж будь-який споживач дійсно її купує [14].

Для AR розробляються спеціальні пристрої, як правило, у вигляді шоломів та гарнітур, що надають можливість занурення користувача у модельне середовище.

AR доповнює реальний світ 3D-моделями, якими можна керувати за допомогою мобільного пристрою в будь-якому місці. Віртуальна реальність (Virtual Reality – VR) занурює користувача у модельний світ, для чого, як правило, необхідні наголовні дисплеї (Head Mounted Devices – HMD) (рис. 1.3).

а

б

Рис. 1.3. Використання шоломів віртуальної реальності для самостійної (а) та спільної (б) роботи

Інтерактивність у програмах для AR і VR забезпечується дуже схоже. Так, наприклад, VR фактично використовує контролери, а у деяких випадках й відстеження рук, що надає змогу користувачеві взаємодіяти з 3D-об’єктами всередині сцени, у якій вони знаходяться.

До головних небезпек використання HMD для роботи у VR відносяться:

– напруження очей;

– запаморочення і головні болі після використання HMD.

На відміну від VR, AR не має таких значних ризиків для здоров’я. Тим не менш, викликає занепокоєння можливість користувачів залишатися зосередженими на тому, що вони роблять, під час використання AR – зокрема, з причин безпеки.

Найбільш поширений тип пристосованих пристроїв, готових для AR – смартфони та планшети (рис. 1.4) з операційними системами iOS (версія 11 та вище під управлінням iPhone та iPad) та Android (версія 7 та вище).

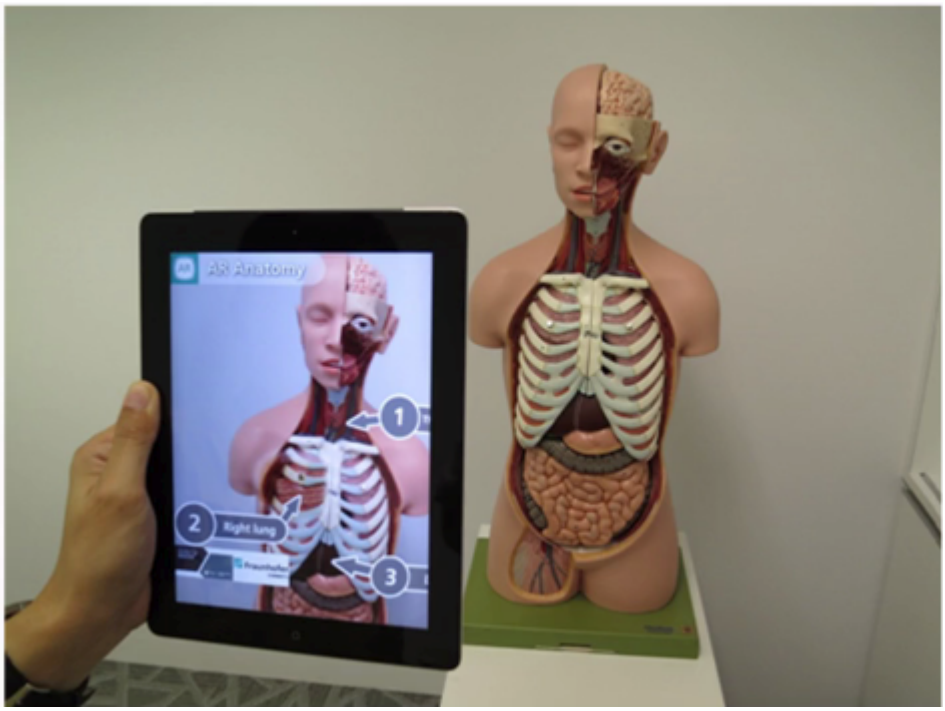

Рис. 1.4. AR Anatomy [2] на планшеті

Для веб-браузерів AR доступна, якщо у них реалізована підтримка WebRTC [30] та WebGL [29] – насамперед Google Chrome та Mozilla Firefox.

Microsoft HoloLens є HMD-подібною гарнітурою для AR, що знаходиться у активній розробці [13]. Так само, як і Google Glass [11], вона розрахована на корпоративне використання, але, на відміну від Glass, спільно не з Android, а з Windows 10.

1.2 Засоби візуалізації комп'ютерних моделей у Web

WebGL (OpenGL ES for the Web) є API для 3D-графіки у веб-браузері, що розробляється The Khronos Group Inc [28]. WebGL використовує мову програмування шейдерів GLSL (OpenGL Shader Language) та є частиною об'єктної моделі документа (DOM API) браузера. Всі провідні розробники браузерів Apple (Safari), Google (Chrome), Microsoft (Edge), та Mozilla (Firefox) є членами WebGL Working Group. Поточна версія WebGL – 2.0 відповідає стандарту OpenGL ES 3.0 API.

Програма, описана за допомогою WebGL, містить як код на JavaScript, та й С-подібний код GLSL. Приклад простої програми для побудови трикутника (рис. 1.5) з [27] містить 105 рядків коду, шо утворюють 5 блоків:

1. Підготовка полотна (HTML-об'єкт canvas) та отримання контексту рендерингу WebGL.

2. Визначення атрибутів геометрії, таких як вершини, індекси та ін., і збереження їх у буферних об'єктах.

3. Створення та компіляція програм для вершинних і фрагментних шейдерів.

4. Зв'язування шейдерних програми з буферними об'єктами.

5. Відображення потрібного об'єкту.

Рис. 1.5. Вихідний код та результат роботи простої програми на WebGL

Використання WebGL для візуалізації комп'ютерних моделей у Web – не найлегших шлях: обсяг необхідного коду буде вимірюватись тисячами рядків та потребуватиме високого рівня підготовки з програмування 3D-графіки за допомогою OpenGL. Ураховуючи, що першим завданням дослідження є добір засобів, доступних для початківців, доцільним є застосування бібліотек, що спрощують використання WebGL.

У таблиці 1.1 подано результати оцінювання доцільності використання бібліотек мовою JavaScript для роботи з об'єктами WebGL.

Таблиця 1.1

Оцінка доцільності використання бібліотек для роботи з WebGL

Назва	**Анімація**	**Убудоване аудіо**	**Доступ до мережі**	**Урахування законів фізики**	**Версія WebGL**	**WebVR**	**Імпорт**	**Експорт**	**Ліцензія**	**Загальна оцінка**
A-Frame	+	+	–	–	1.0	+	Багато форматів (3)	HTML, three.js (2)	Вільна	**10**
CopperLicht	+	+	–	+	1.0	–	–	–	Вільна	5
OSG.JS	+	+	–	–	1.0	+	–	–	Вільна	5
Three.js	+	+	–	–	2.0	+	Багато форматів (9)	Багато форматів (4)	Вільна	**19**
Verge3D	+	+	–	+	1.0	+	Багато форматів (3)	glTF (1)	Комерційна	9
Clara.io	+	–	–	+	1.0	+	Багато форматів (5)	Багато форматів (6)	Комерційна	15
Babylon.js	+	+	–	+	2.0	+	Багато форматів (5)	Багато форматів (9)	Вільна	**21**

Ураховуючи спрямованість результатів проведеного аналізу на навчання початківців, були визначені кращі за рейтингом некомерційні бібліотеки для роботи з WebGL: 1 місце – Babylon.js, 2 місце – Three.js та 3 місце – A-Frame.

Babylon.js [7] надає можливість створювати складні 2D-об'єкти з використанням суттєво меншого обсягу коду, ніж при застосуванні WebGL: так, обсяг коду, необхідний для створення об'єкту, поданого на рис. 1.6, у 5 разів менше, ніж WebGL-коду для суттєво простішої сцени на рис. 1.5.

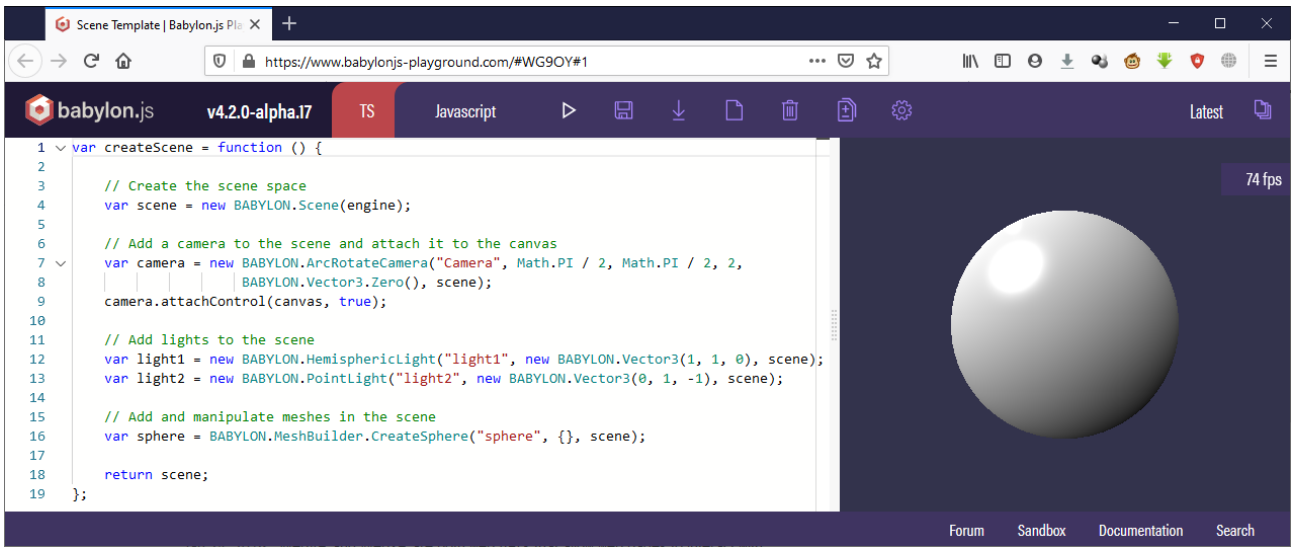

Рис. 1.6. Вихідний код та результат роботи простої програми на Babylon.js

Створена сцена є керованою за допомогою стандартних елементів управління. На відміну від інших учасників рейтингу, Babylon.js є «фізичним» рушієм, що широко застосовується при створенні комп'ютерних ігор для Web.

Three.js – це відкрита 3D-графічна бібліотека загального призначення, написана на JavaScript. На сайті https://threejs.org/ можна знайти велику кількість якісних демонстрацій, створених із її використанням [23]. Її автор Рікардо Мігель Кабелло, також відомий як mrdoob, є одним із піонерів використання WebGL, тому ця бібліотека часто використовується при побудові інших бібліотек.

Рис. 1.7. Вихідний код та результат роботи простої програми на Three.js

Створення об’єктів у Three.js відбувається у три кроки:

1) визначення геометрії об’єкту – векторів позиції, кольорів та ін.;

2) визначення матеріалу – способу рендерингу об’єкту;

3) композиція геометрії та матеріалу.

Обсяг коду, необхідний для створення 3D-сцени за допомогою Three.js, є дещо більшим, ніж коду Babylon.js – це пов’язано із об’єднанням у Babylon.js операцій створення об’єкту та додавання об’єкту до сцени в один виклик конструктора відповідного класу.

A-Frame фактично, є засобом швидкого прототипування, і значна частина програми з його використанням – це HTML-подібний код. Команди A-Frame описуються тегами, які подібні до тегів HTML, але, на відміну від останніх, інтерпретуються не у веб-браузері на боці клієнта, а є способом доступу до JavaScript, що виконується на боці сервера [1].

Рис. 1.8 є дуже показовим – обсяг коду A-Frame, необхідний для створення все тієї ж сфери, утричі менше, ніж коду з використанням Babylon.js/Three.js:

Рис. 1.8. Вихідний код та результат роботи простої програми на A-Frame

Таким чином, незважаючи на третє місце в рейтингу функціональних можливостей, A-Frame є лідером за наочністю та доступністю серед розглянутих бібліотек для роботи з 3D-графікою у Web. Ураховуючи, що A-Frame є надбудовою над Three.js, доцільним є їх циклічне опанування, розпочинаючи із A-Frame.

1.3 Засоби відстеження реальних об’єктів

У [22] було виконано порівняльний аналіз найбільш поширених AR SDK, тому зосередимось лише на тих із них, які придатні для розробки у Web – WebAR SDK. На жаль, серед вільних засобів таких станом на травень 2020 року лише три, і кожен з них тісно пов’язаний із розглянутими засобами побудови комп’ютерних моделей.

Babylon.js у серпні 2019 року анонсував **Babylon AR** [19] – проект з інтеграції Babylon.js та бібліотеки комп’ютерного зору OpenCV. На поточний момент у рамках проекту реалізовано відслідковування стандартних маркерів, подібних до QR-кодів. Не зважаючи на те, що самі розробники Babylon.js визначають Babylon AR як проект на дуже ранній стадії, створені за його допомогою веб-програми працездатні лише на мобільних пристроях, які підтримують WebXR [31]. Це звужує сферу використання Babylon AR до пристроїв з Android версії 8.1 та вище. Станом на квітень 2020 року, в Україні таких пристроїв біля 65 % від загальної кількості усіх Android-пристроїв [16]. Ураховуючи тенденцію до зростання цієї частки, Babylon AR можна розглядати як перспективний проект із розробки програмного забезпечення із доповненою реальністю для Web (починаючи з 2021-2022 рр.).

ARToolKit, на 20 років старший за Babylon AR, є однією з найбільш широко використовуваних бібліотек для AR. **JSARToolKit** (ARToolKit.js) на поточний момент підтримує такі 3 типи квадратних маркерів (з довільним рисунком, двовимірним кодом та набори маркерів) й NFT-маркери (natural feature tracking – відслідковування довільних зображень) [5]. JSARToolKit не є специфічною для певної бібліотеки WebGL, проте найчастіше застосовується разом із Three.js: приклад [21] їх спільної роботи, розроблений Л. Стемкоскі (Lee Stemkoski), подано на рис. 1.9.

Обсяг коду, необхідній для реалізації функціональності прикладу – біля 100 рядків – є суттєво меншим, ніж за автономного застосування JSARToolKit. Це досягається застосування бібліотеки THREEx – розширення Three.js для розробки комп’ютерних ігор, що наближує її функціональність до Babylon.js

[24].

Рис. 1.9. Куб на маркері Hiro (спільне використання Three.js та JSARToolKit)

Її автором є Жером Етьєнн (Jerome Etienne) [12] – основний розробник першої та другої версії бібліотеки **AR.js**, яка на травень 2020 року має версії, адаптовані для роботи як з A-Frame, так й з Three.js. Так само, як A-Frame є надбудовою над Three.js, в основу AR.js була покладена JSARToolKit, тому дана бібліотека підтримує всі типи квадратних (Marker tracking) та NFT-маркерів (Image Tracking), що й JSARToolKit, а також надає можливість розміщення комп'ютерних моделей за їх географічними координатами (Location based AR) [3].

При застосуванні спільно із A-Frame код, необхідний для реалізації прикладу, подібного до поданого на рис. 1.9, займає кілька рядків:

```
<a-scene embedded arjs>
     <a-marker preset="hiro">
          <a-box></a-box>
     </a-marker>
     <a-entity camera></a-entity>
</a-scene>
```

Висновки до розділу 1

1. Доповнена реальність – це штучне (синтетичне) середовище, створене шляхом об’єднання об’єктів реального світу та даних, згенерованих комп’ютером (комп’ютерних моделей).

2. Огляд засобів розробки програмного забезпечення із доповненою реальністю для Web надав можливість рекомендувати для опанування початківцями такі комбінації засобів візуалізації комп’ютерних моделей у Web та засобів відстеження реальних об’єктів:

а) A-Frame та AR.js – API для швидкого прототипування, значна частина програм з використанням яких є HTML-подібний код. A-Frame використовується для створення сцен, об’єктів, анімації та інших 3D-елементів у веб-браузері. AR.js використовується для відслідковування маркерів і надає можливість сцені, сконструйованій за допомогою A-Frame, відображатися безпосередньо на маркері;

б) Three.js та ARToolKit.js – для поглибленого рівня, що передбачає створення програм засобами JavaScript. Three.js використовує WebGL, що надає можливість створення якісних 3D-сцени безпосередньо у веб-браузері. JSARToolKit використовується для відслідковування маркерів та низькорівневого доступу до даних із камери пристрою.

3. Програмні засоби із доповненою реальністю, розроблені із використанням вказаних пар засобів, можуть бути розміщені в Інтернет на одному із хмарних сервісів. Виходячи з того, що бібліотека Three.js є основною A-Frame так само, як JSARToolKit є основою AR.js, для досягнення мети дослідження необхідною є розробка інструктивних матеріалів насамперед із спільного використання A-Frame та AR.js.

РОЗДІЛ 2
МЕТОДИЧНІ ОСНОВИ ВИКОРИСТАННЯ A-FRAME ТА AR.JS ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІЗ ДОПОВНЕНОЮ РЕАЛЬНІСТЮ

2.1 Налаштування проекту та створення сцени

Для початку роботи з A-Frame необхідно перейти на сайт A-Frame за посиланням https://aframe.io/docs/1.0.0/introduction/installation.html (рис. 2.1) та завантажити з нього файл збірки JavaScript (JS Build, Production Version – https://aframe.io/releases/1.0.4/aframe.min.js).

Рис. 2.1. Документація до A-Frame із посиланнями на завантаження

Завантажений файл збірки необхідно зберегти у окремому каталозі (наприклад, aframe) всередині робочого каталогу, після чого створити у останньому індексний файл HTML ARindex.html:

```
code
|    ARindex.html
|
└──aframe
        aframe.min.js
```

із наступним вмістом

```
<!DOCTYPE html>
<html>
  <script src="aframe/aframe.min.js"></script>
```

```
  <a-scene>
    <a-sky color="grey"></a-sky>
  </a-scene>
</html>
```

Файл містить команду `script` із посиланням на завантажений файл – вона завжди використовується для початку роботи з бібліотекою JavaScript. Саме у ньому інтерпретується тег `a-scene`, в якому знаходитиметься більша частина коду A-Frame. Тег `a-sky` створюватиме фоновий колір (його також можна застосувати для розміщення 360° зображення на сцені). Після відкриття файлу ARindex.html у веб-браузері отримаємо вікно із сірим фоном (рис. 2.2).

Рис. 2.2. Порожня сцена у віртуальній реальності

Про те, що A-Frame API дійсно працює, свідчить режим VR (Virtual Reality) у нижньому правому куті. Суттєво більше відомостей можна отримати, увімкнувши режим візуального 3D-інспектора (Ctrl-Alt-I, рис. 2.3).

Додавши до індексного файлу команду

```
<a-torus position="-2 1 -5" color="green" radius="1.2"></a-torus>
```

отримаємо зображення зеленого тору на сірому тлі (рис. 2.4).

У новостворених об’єктів є певні атрибути – ознайомитись із ними можна у документації до A-Frame. Поточна версія підтримує наступні примітиви:

`a-box` – прямокутний паралелепіпед;

`a-camera` – камера (визначає, що бачить користувач);

`a-circle` – круг;

`a-collada-model` – 3D-модель у форматі COLLADA (.dae);

`a-cone` – конус;

`a-cursor` – опрацювання подій від миши;

`a-curvedimage` – панорамне зображення;

`a-cylinder` – циліндричні поверхні;

`a-dodecahedron` – дванадцятигранник;

`a-gltf-model` – 3D-модель у форматі glTF (.gltf);

`a-icosahedron` – двадцятигранник;

`a-image` – пласке зображення;

`a-light` – джерела світла;

`a-link` – гіперпосилання;

`a-obj-model` – 3D-модель у форматі Wavefront (.obj/.mtl);

`a-octahedron` – восьмигранник;

`a-plane` – площина;

`a-ring` – пласке кільце або диск;

`a-sky` – додає фоновий колір або 360° зображення;

`a-sound` – джерело звуку;

`a-sphere` – сфера або багатогранник;

`a-tetrahedron` – трикутна піраміда;

`a-text` – плаский текст;

`a-torus-knot` – кренделеподібна фігура;

`a-torus` – тор;

`a-triangle` – трикутна поверхня;

`a-video` – відео як текстура на площині;

`a-videosphere` – 360° фонове відео.

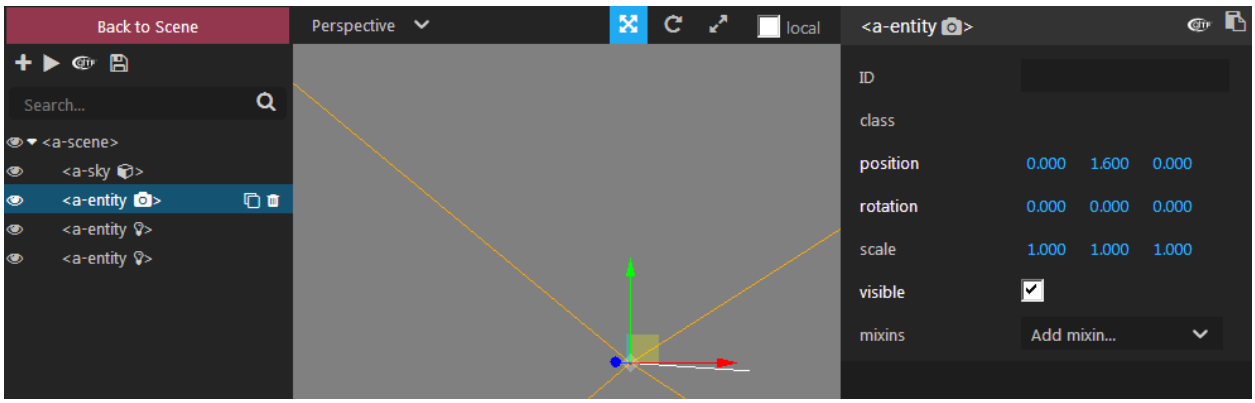

Рис. 2.3. Інспектор A-Frame

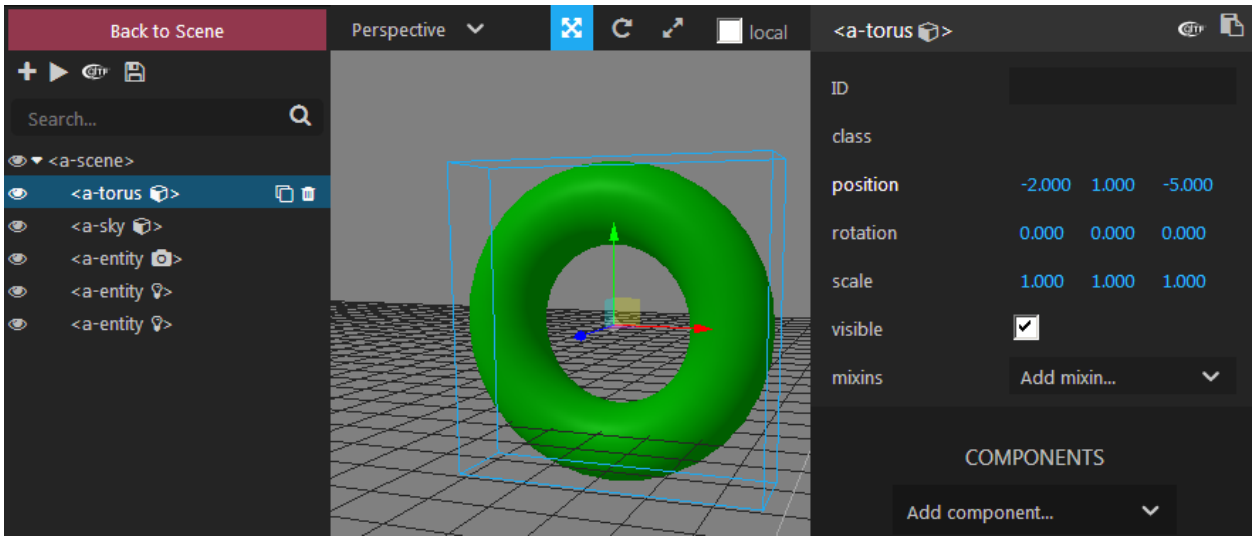

Рис. 2.4. Модифікована сцена в інспекторі A-Frame

2.2 Об’єктні сітки та атрибути сітки

Додамо до попередньої сцени, що містить зелений тор та сірий фон, ще кілька примітивів. Спочатку вставимо між тором та фоном площину:

```
<a-plane width="7" height="7" rotation="50 0 0"
         position="3 -2 -3" color="purple"></a-plane>
```

Для перегляду параметрів `a-plane` слід скористатись документацією за посиланням https://aframe.io/docs/1.0.0/primitives/a-plane.html – тут можна знайти різні атрибути площини, які можна застосувати до неї в тегу A-Frame. Параметри `width` та `height` відповідають за ширину та висоту прямокутника, `rotation` вказує на необхідність її повороту на 50° відносно вісі *x* та на 0 –

відносно *y* та *z*, `position` – координати початку площини за відповідними осями, а `color` – обраний колір.

Площину, налаштовану у такий спосіб, на сцені можна побачити лише в режимі інспектора, тому що вона розміщена дуже далеко на задньому плані. Якщо змінити координати початку площини на "-2 -2 -5", її можна побачити (рис. 2.5).

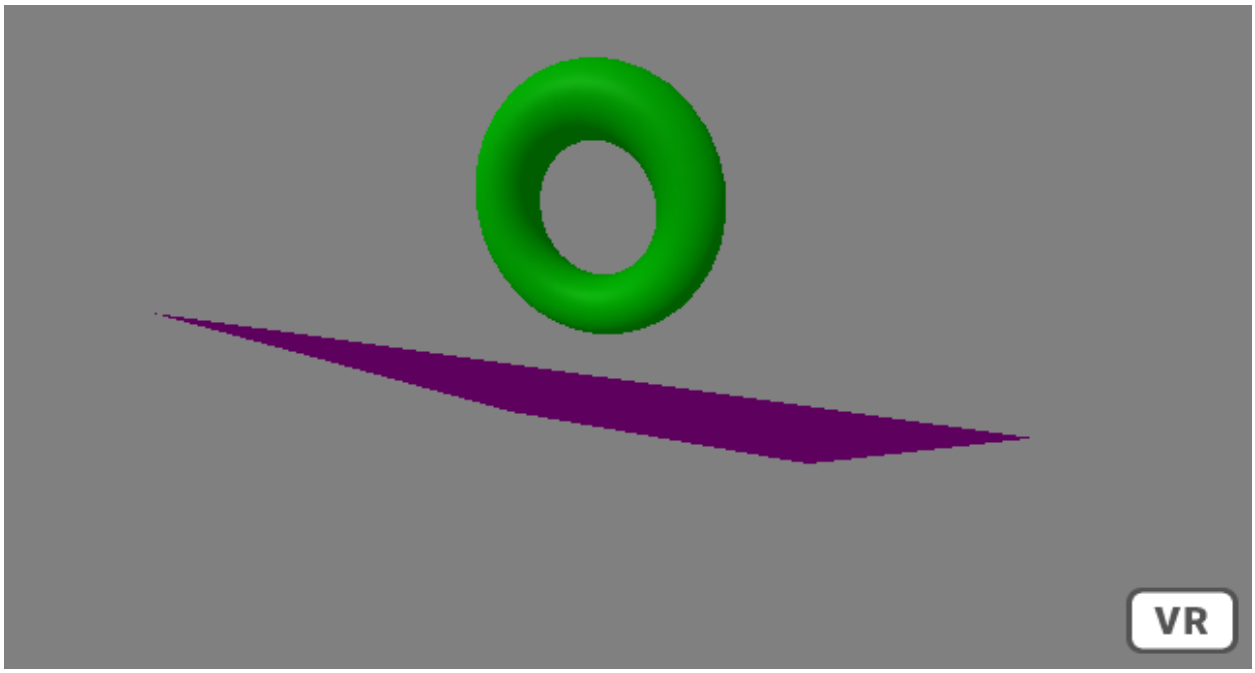

Рис. 2.5. Сцена із тором та площиною

Отже, якщо в певній позиції об’єкт фактично не відображається, це може бути проблемою з позиціонуванням самого об’єкту всередині коду. Тепер, коли у нас є площина, додамо ще кілька об’єктів. Створимо циліндричний об’єкт у формі льодяника жовтого кольору, розташованого під самим тором (рис. 2.6):

```
<a-cylinder color="yellow" height="2" radius="0.05"
            position="-2 -1 -5"></a-cylinder>
```

Додамо ще один циліндр (рис. 2.7):

```
<a-cylinder color="blue" height="2" radius="0.05"
            position="-3 -1 -5"></a-cylinder>
```

І, нарешті, кренделеподібну фігуру (рис. 2.8):

```
<a-torus-knot color="orange" radius="1.2"
              position="-3 1 -5"></a-torus-knot>
```

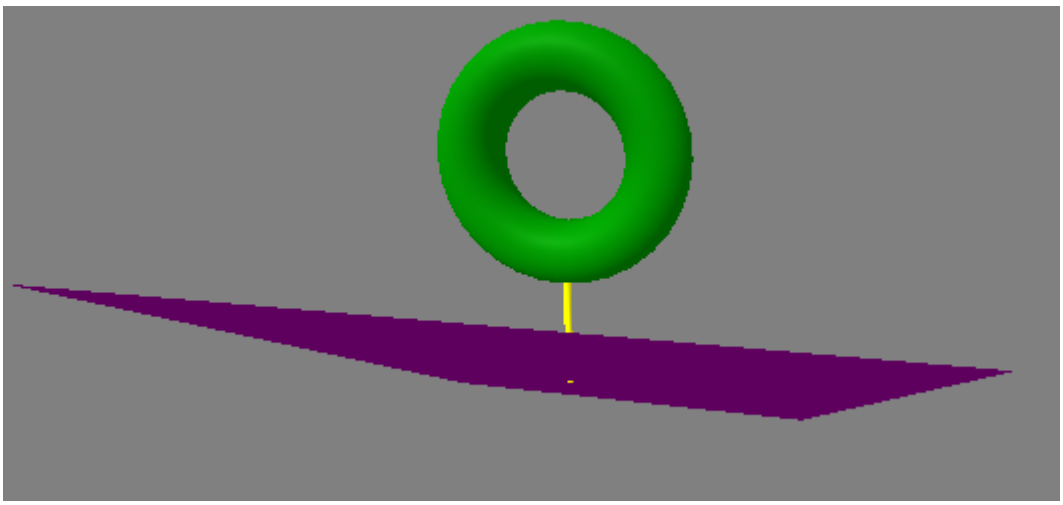

Рис. 2.6. Подальша модифікація сцени

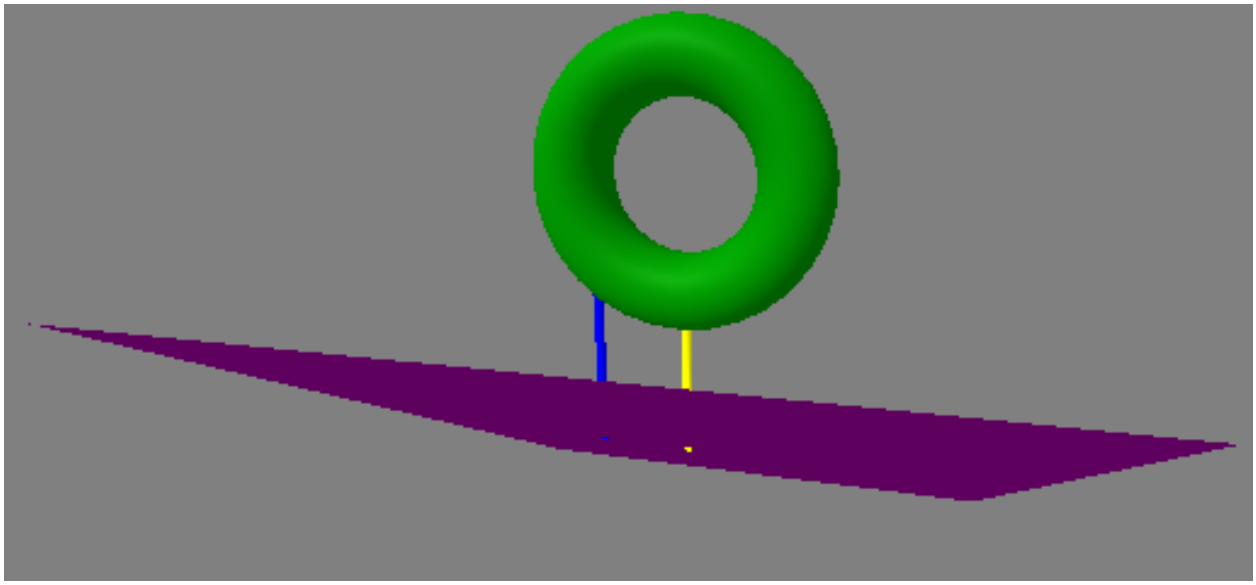

Рис. 2.7. Сцена із тором, площиною та двома циліндрами

2.3 Додавання тексту та анімації до об’єктів

Виконаємо невеликі зміни у індексному файлі в атрибуті обертання площини – зміна кута огляду на "-55 0 0" надає їй суттєво кращого вигляду (рис. 2.9).

Додамо до сцени площину, перпендикулярну попередній:

```
<a-plane width="9" height="2" position="3 1 -9"></a-plane>
```

За замовчанням її колір буде білим – достатньо зручний для того, щоб перед ним розмістити напис (рис. 2.10):

```
<a-text value="Welcome to browser's VR" color="black"
        width="10" position="-0.5 1 -6"></a-text>
```

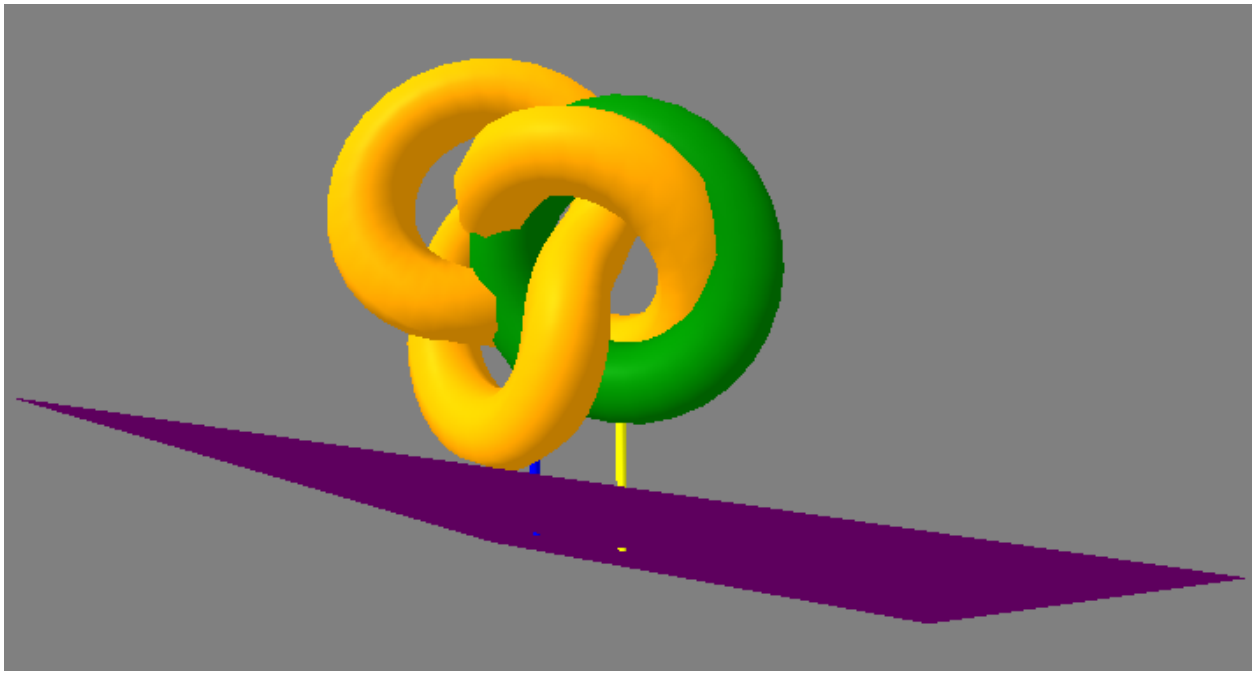

Рис. 2.8. Приклад роботи команди a-torus-knot

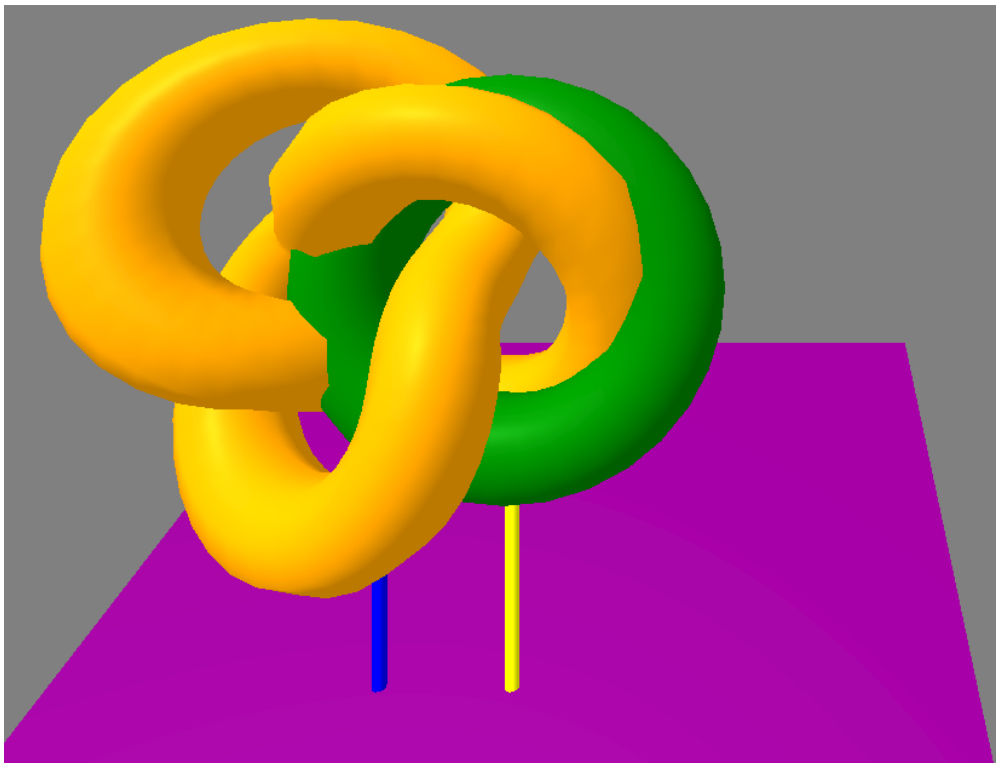

Рис. 2.9. Зміна кута нахилу площини

«Лобова» спроба замінити даний напис на кириличний, скоріше за все, виявиться невдалою без використання відповідного шрифта. Більш ніж 2000

різних шрифтів та способи їх використання можна знайти у репозитарії https://github.com/etiennepinchon/aframe-fonts. Будь-який шрифт із репозитарію може бути підключений за посиланням виду `https://raw.githubusercontent.com/etiennepinchon/aframe-fonts/master/fonts/[FONT_NAME]/[FONT_TYPE].json`

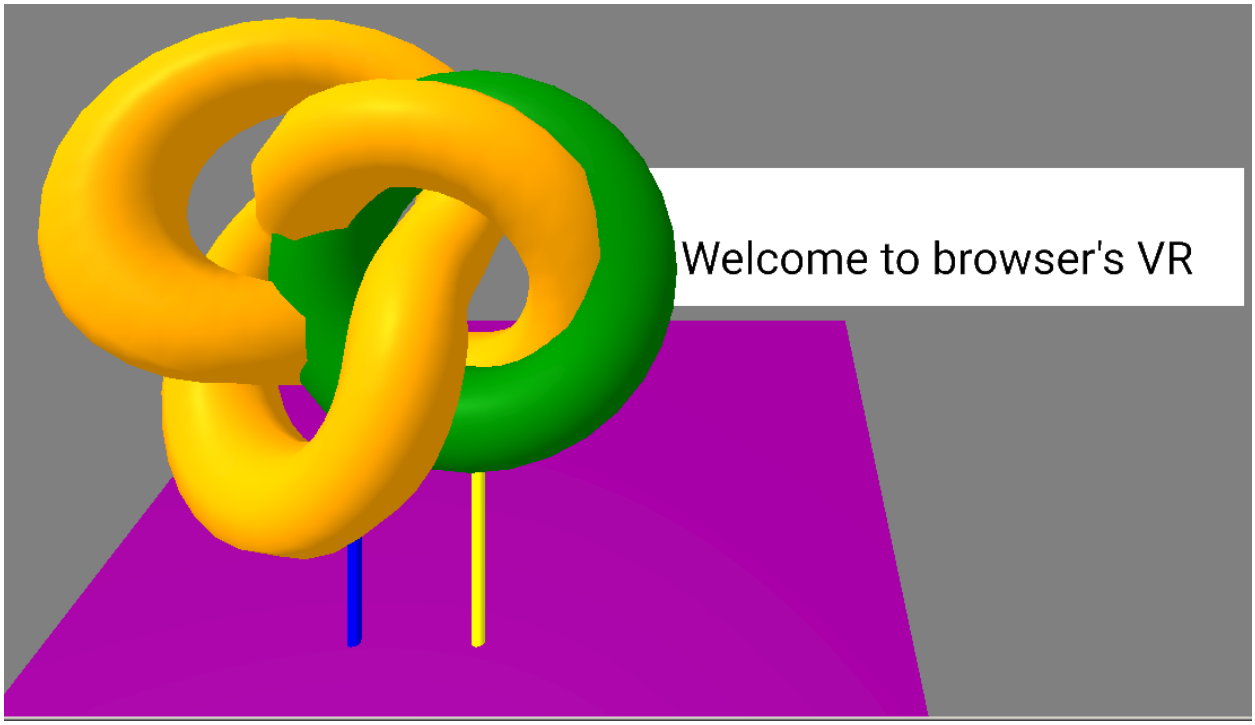

Рис. 2.10. Сцена із написом англійською

Інший варіант – створення власного шрифту за допомогою MSDF font generator (рис. 2.11).

Архів, завантажуваний з https://msdf-bmfont.donmccurdy.com, містить шрифт у форматах JSON та PNG (для вихідного шрифту arial.ttf це будуть arial-msdf.json та arial.png відповідно – розташуйте їх там само, де знаходиться й ARindex.html). Чим більше символів обирається, тим більше має бути розмір PNG-файлу (до 1024x1024). Для економії трафіку до них включені лише ті символи, які були обрані користувачем – всі інші не відображатимуться.

Унесемо зміни до параметру `value` та додамо параметри `font` і `negate` (рис. 2.12):

```
<a-text value="Вітаємо у браузерній VR!" color="black"
        width="10" position="-0.5 1 -6"
```

```
font="arial-msdf.json" negate="false"></a-text>
```

Рис. 2.11. Генерація власного шрифта для використання у A-Frame

Якщо останній матиме істинне значення, обраний колір стане фоновим для кожної з літер.

Новий текст не відображається – згідно повідомлень із консолі браузера, локальне завантаження (за протоколом `file://`) шрифту (так само, як і багатьох інших ресурсів), не дозволено. Для того, щоб виправити цю помилку, необхідно файли шрифту розмістити на сервері з доступом за протоколом HTTP або HTTPS. Де та як це зробити, суттєво залежить від власних можливостей користувача – студенти зазвичай можуть звернутись до викладача для того, щоб отримати доступ до зовнішнього серверу, або налаштувати власний.

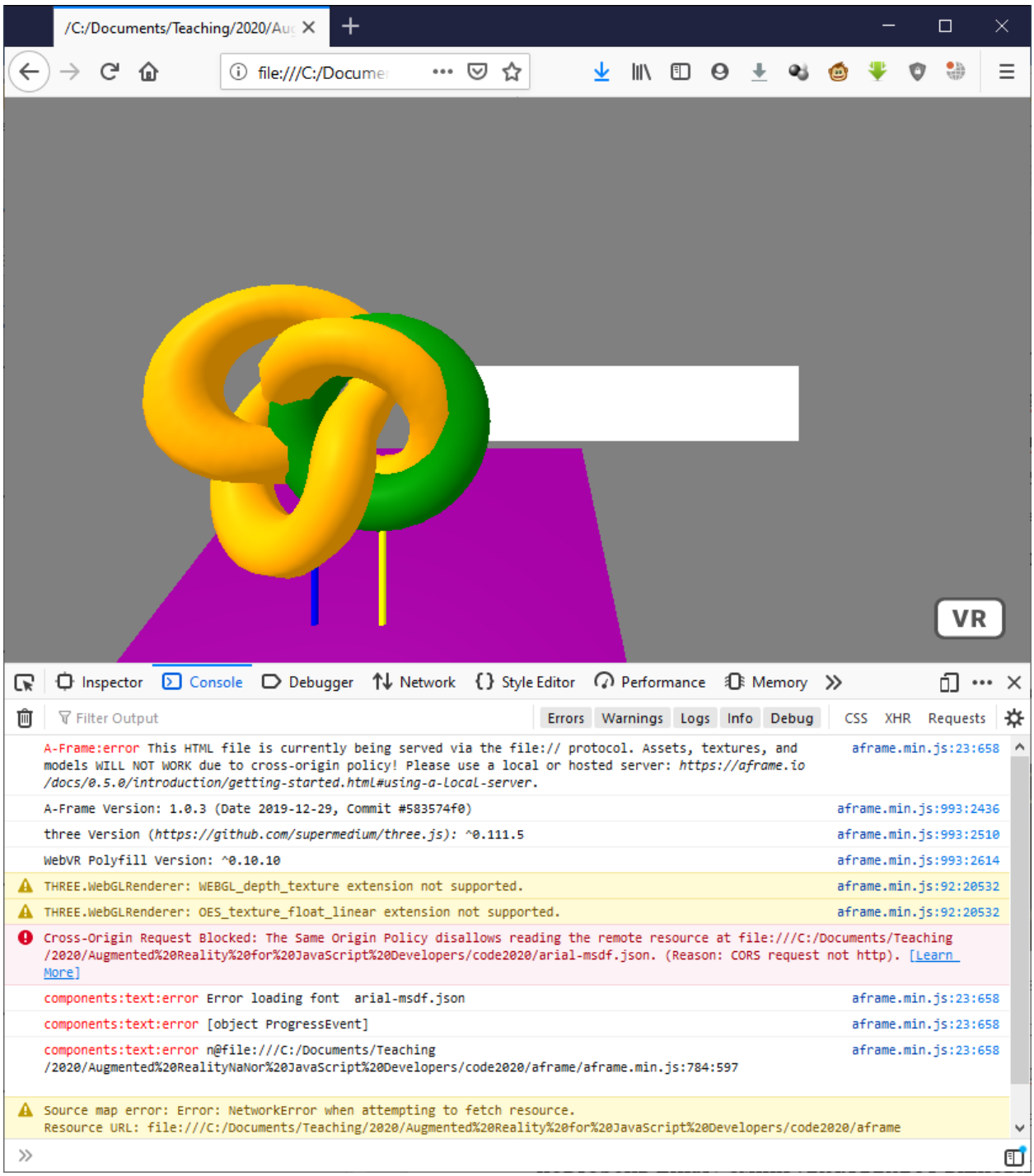

Рис. 2.12. Невдала спроба застосування згенерованого шрифта

Наприклад, для отримання доступу за протоколом FTP до власного розділу на сервері https://playground2.ccjournals.eu (рис. 2.13) користувачеві необхідні знати ім’я або адресу FTP-серверу (host, hostname), ім’я (login, username) та пароль (password). Після реєстрації користувач побачить поточний вміст його розділу на сервері (рис. 2.14).

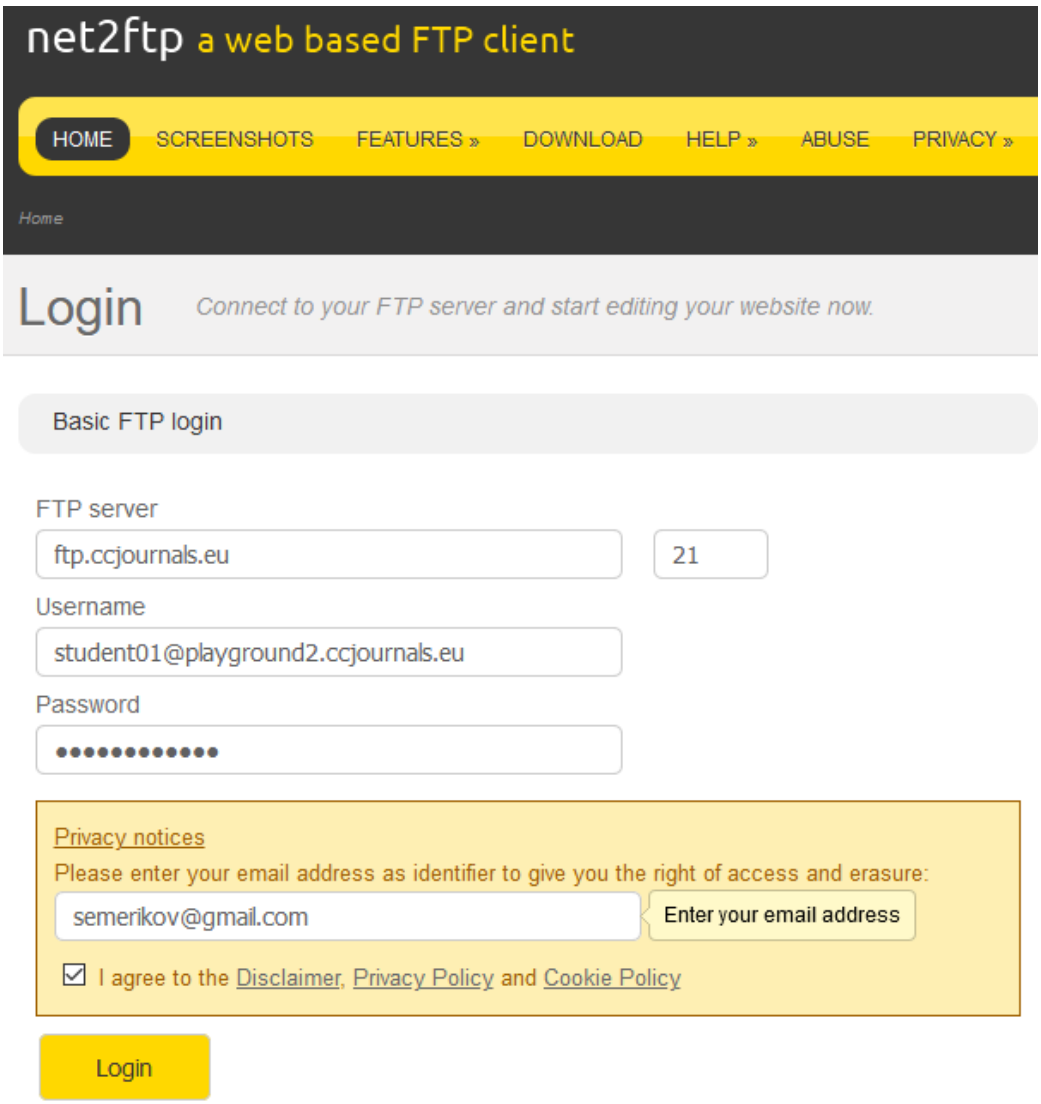

Рис. 2.13. Web-клієнт для доступу до хмарного сервера за протоколом FTP

У файлі index.html доцільно розмістити посилання на інші файли, створені користувачем. Редагувати їх можна як безпосередньо на сервері, так й локально із подальшим завантаженням. Для того, щоб поточний приклад був працездатним, необхідно завантажити як сам ARindex.html, так й конвертовані файли із шрифтами arial-msdf.json та arial.png (рис. 2.15).

Для масового завантаження файлів доцільно скористатись можливістю завантаження архіву з ними.

Перевірка виконання ARindex.html на сервері показує, що результат вже суттєво кращий (рис. 2.16).

Для завершення локалізації надпису необхідно масштабувати його до розміру площини, вказавши параметр `scale`, та додати до набору символів, для

яких генерується шрифт, пропуск. Застосування символів, що не входять до набору ASCII, потребує додаткового вказання одного з найпоширеніших кодувань тексту – UTF-8 – після тегу `<html>`:

```
<meta charset="utf-8">
```

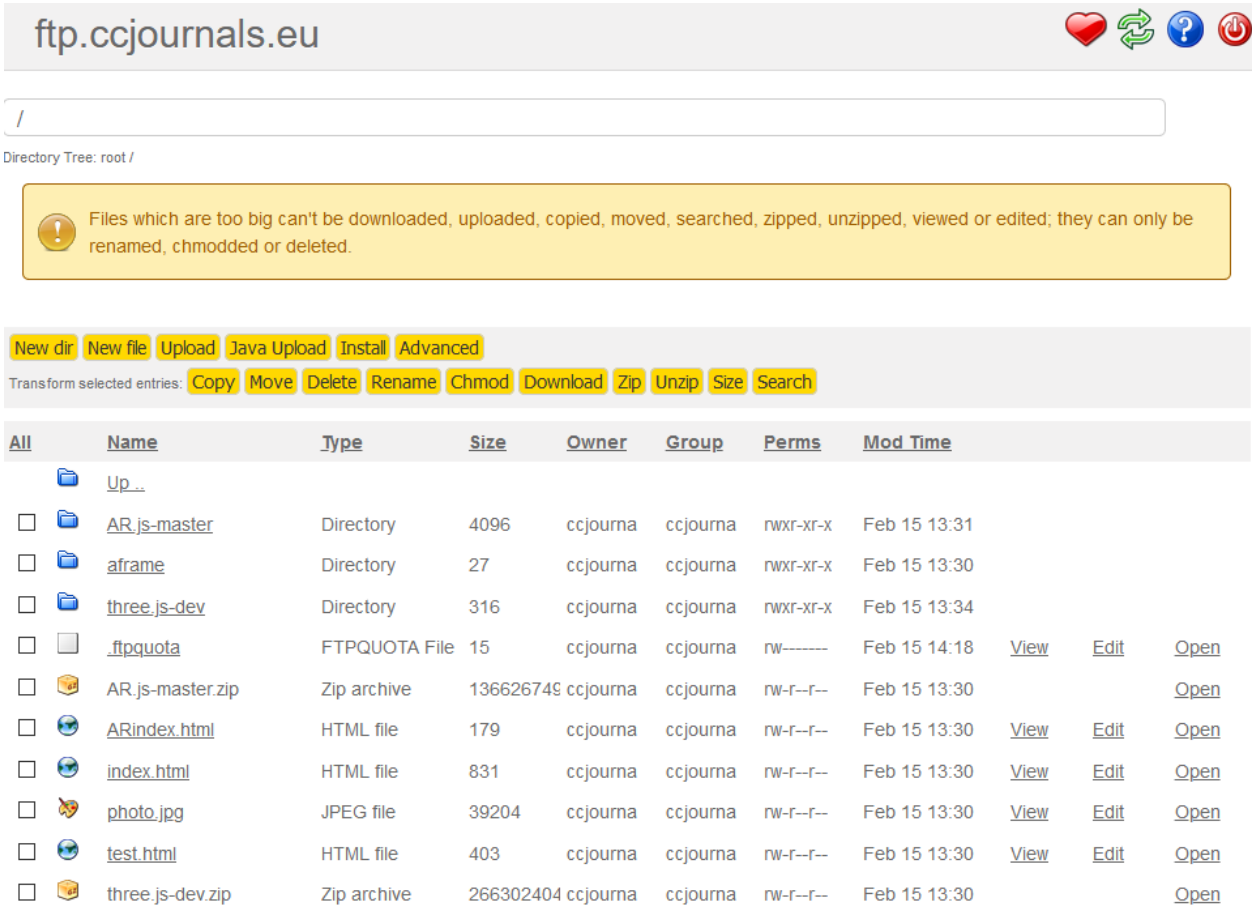

Рис. 2.14. Список файлів на сервері

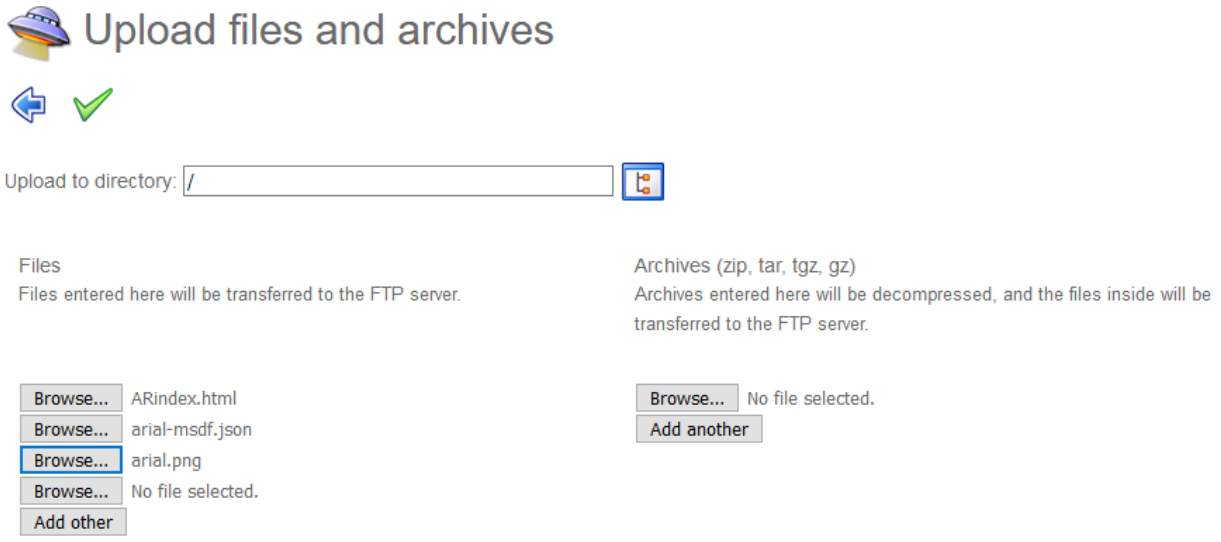

Рис. 2.15. Завантаження файлів на сервер

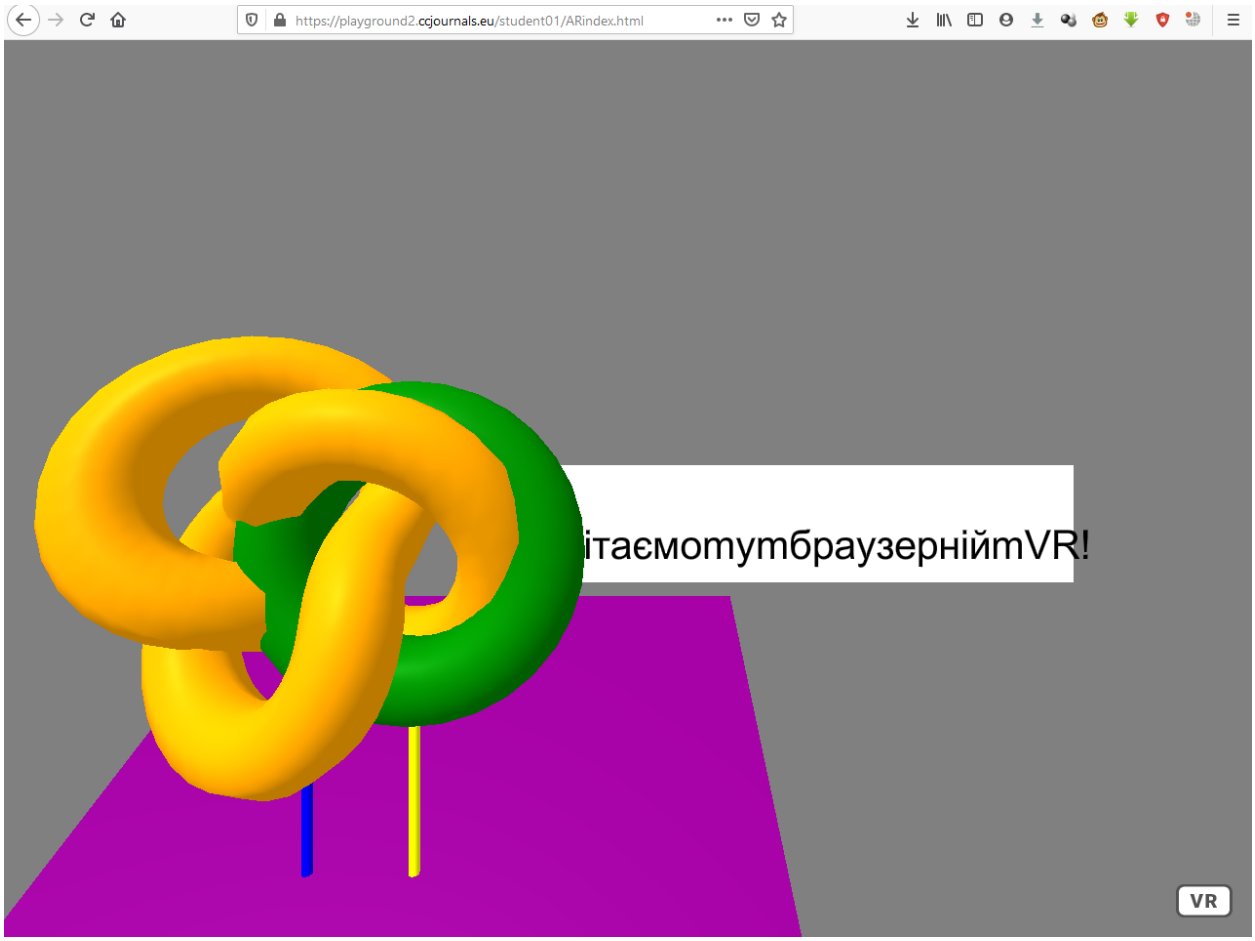

Рис. 2.16. Сцена, доступна у мережі Інтернет

Для того, щоб анімувати примітив, необхідно додати до нього параметр `animation`.

```
<a-torus position="-2 1 -5" color="green" radius="1.2"
     animation="property: components.material.material.color; type:
     color; from: green; to: red; loop: true; dur: 10000">
</a-torus>
```

У даному прикладі основним в параметрі `animation` є `property` – та складова об'єкту, яка буде змінюватись (`components.material.material.color` – колір). `dur` визначає тривалість анімації в мілісекундах (10000 мс = 10 с), `from` задає початкове значення атрибуту, `to` – кінцеве, а `loop` визначає кількість повторень циклу зміни (`true` відповідає нескінченному циклу).

Наступний приклад демонструє зміну атрибуту `rotation` для того, щоб

змусити тороподібний об’єкт обертатися навколо центру:

```
<a-torus-knot color="orange" radius="1.2" position="-3 1 -5"
        animation="property: rotation; to: 0 0 360; loop: true; dur: 10000">
</a-torus-knot>
```

Більш детально про параметр `animation` можна прочитати у довідці з A-Frame Core API за посиланням https://aframe.io/docs/1.0.0/components/animation.html

2.4 Додавання текстур до об’єктів

Головна сторінка http://aframe.io містить посилання на вихідні коди бібліотеки. На рівень вище серед 18 репозиторіїв A-Frame можна знайти sample-assets – приклади файлів зображень, моделей та аудіозаписів для використання у A-Frame. Для того, щоб скористатися ними, необхідно отримати пряме посилання на об’єкт репозитарію, та додати його у якості атрибуту `src` до об’єкту, для якого обране зображення виступатиме текстурою. Змінимо площину під торами (рис. 2.17):

```
    <a-plane
src="https://raw.githubusercontent.com/aframevr/sample-
assets/master/assets/images/illustration/758px-
Canestra_di_frutta_(Caravaggio).jpg"      width="7"      height="7"
rotation="-55 0 0" position="-2 -2 -5" color="purple"></a-plane>
```

Посилання на файл текстури може бути узяте із будь-якого місця, але не завжди сайти дозволяють використовувати прямі посилання на файли сайту поза його межами. «Політика того ж походження» (same origin policy) є важливим механізмом безпеки у сучасних веб-браузерах, що стосується як виконуваних у браузері файлах, так й використовуваних.

2.5 Додавання AR.js до програми

Перш за все необхідно перейти до сховища AR.js у GitHub за посиланням https://github.com/jeromeetienne/AR.js і завантажити ZIP-файл, натиснувши «Clone or download» (рис. 2.18). Станом на початок 2020 року розмір репозитарію AR.js – більше 170 Мб, тому завантаження архіву може тривати

певний час.

Рис. 2.17. Сцена із текстурою

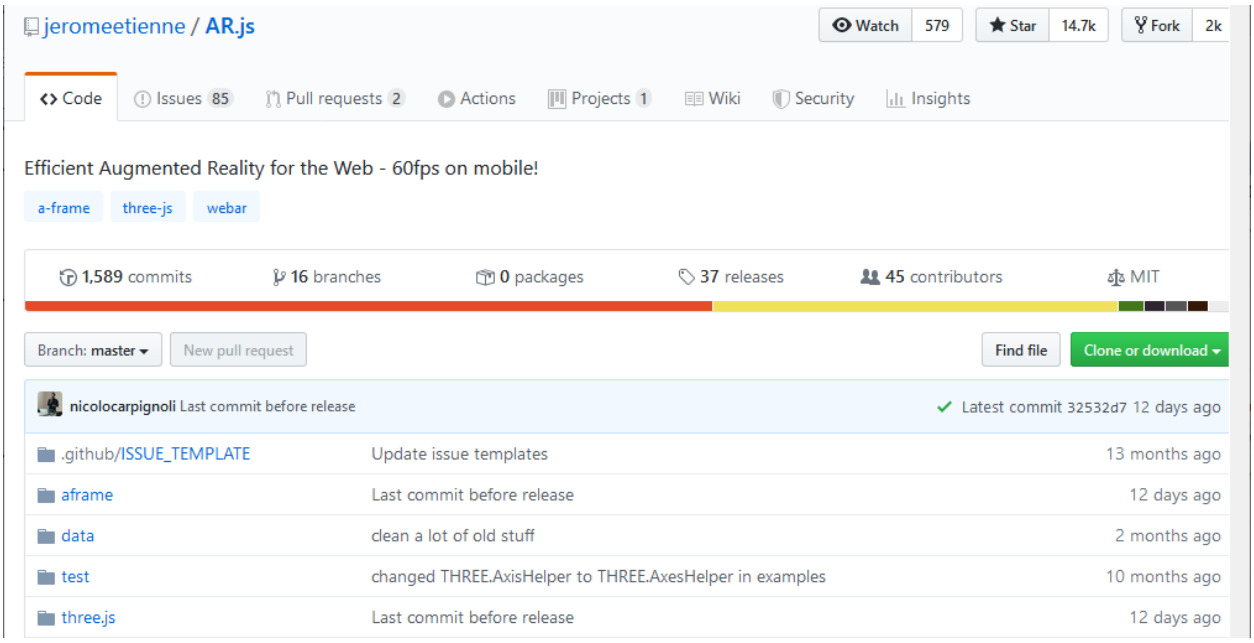

Рис. 2.18. Репозитарій AR.js 2.1

Завантажений архів доцільно розпакувати у каталог, де зберігаються

індексний файл та каталог aframe:

```
│   ARindex.html
│
├───aframe
│       aframe.min.js
│
└───AR.js-master
    │   .gitignore
    │   .travis.yml
    │   AR.js-1920-1080-HD.png
    │   CHANGELOG.md
    │   CODE_OF_CONDUCT.md
    │   CONTRIBUTING.md
    │   HOW_TO_RELEASE.md
    │   ISSUE_TEMPLATE.md
    │   LICENSE.txt
    │   Makefile
    │   package-lock.json
    │   package.json
    │   PULL_REQUEST_TEMPLATE.md
    │   README.md
    │   TODO.md
    │
    ├───.github
    │   └───ISSUE_TEMPLATE
    │           bug_report.md
...
```

Структура каталогів репозитарію досить розгалужена, але прозора. Так, у AR.js-master\data\images\ можна знайти файл HIRO.jpg (рис. 2.19) – дане зображення буде маркером, необхідним для того, щоб при наведенні на нього веб-камери відображалась сцена.

Рис. 2.19. Убудований маркер Hiro

Для початку використання AR.js необхідно внести кілька змін до вихідного індексного файлу ARindex.html. По-перше, після тегу `script` для підключення стандартної версії бібліотеки A-Frame додається ще один тег для роботи із AR.js та захоплення даних із веб-камери:

```
<script src="AR.js-master/aframe/build/aframe-ar.min.js"></script>
```

Для того, щоб побудовані об’єкти краще відображались на маркері, змінимо деякі їх атрибути:

– позицію тора встановимо у (0; 0,5; 0), а радіус – у 0,5;

– ширину та висоту площини зменшимо до 3,5, а позицію встановимо у (0; -1; 0);

– позицію першого циліндра змінімо на (0; 0; 0), а другого – на (1; 0; 0);

– для кренделеподібної фігури радіус встановимо у 0,5, а позицію – у (1; 0,5; 0);

– до площини, на яку накладатиметься текст, застосуємо текстуру за посиланням https://raw.githubusercontent.com/aframevr/sample-assets/master/assets/images/uvgrid/UV_Grid_Sm.jpg та встановимо її ширину у 2,5, висоту – в 1,5, а позицію – у (0; 1; -1);

– ширину тексту встановимо у 3, а позицію – у (-1; 0,5; -1).

Створена сцена була побудована для використання у веб-VR. Для використання об’єктів на ній у доповненій реальності спочатку позбавимось сірого фону, доданого тегом `a-sky`, шляхом його видалення. Далі необхідно вбудувати у сцену AR.js шляхом додавання атрибутів `embedded` (застосовувати вбудований розпізнавач маркерів) та `arjs = 'trackingMethod: best;'` (використовувати найкращий метод відстеження):

```
<a-scene embedded arjs = 'trackingMethod: best;'>
```

Крім того, доведеться створити тег `a-anchor` для того, щоб виконати прив’язку до доповненої реальності:

```
<a-anchor hit-testing-enabled='true'>
```

Всі об’єкти, розміщені на сцені, слід розмістити між тегами `<a-anchor>` та `</a-anchor>`. Останній крок – додавання статичної камери:

```
<a-camera-static/>
```

На відміну від попередніх, це тег не є парним, що, за стандартом XHTML, потребує додавання слешу наприкінці тега.

```
<!DOCTYPE html>
<html>
  <meta charset="utf-8">
  <script src="aframe/aframe.min.js"></script>
  <script src="AR.js-master/aframe/build/aframe-ar.min.js"></script>
  <a-scene embedded arjs = 'trackingMethod: best;'>
    <a-anchor hit-testing-enabled='true'>
    <a-torus position="0 0.5 0" color="green" radius="0.5"
             animation="property: components.material.material.color; type: color; from: green; to: red; loop: true; dur: 10000"></a-torus>
    <a-plane width="3.5" height="3.5" rotation="-55 0 0" position="0 -1 0" color="purple" src=
     "https://raw.githubusercontent.com/aframevr/sample-assets/master/assets/images/illustration/758px-Canestra_di_frutta_(Caravaggio).jpg"
     ></a-plane>
    <a-cylinder color="yellow" height="2" radius="0.05" position="0 0 0"></a-cylinder>
    <a-cylinder color="blue" height="2" radius="0.05" position="1 0 0"></a-cylinder>
    <a-torus-knot color="orange" radius="0.5" position="1 0.5 0" animation="property: rotation; to: 0 0 360; loop: true; dur: 10000">
    </a-torus-knot>
    <a-plane width="2.5" height="1.5" position="0 1 -1"
      src="https://raw.githubusercontent.com/aframevr/sample-assets/master/assets/images/uvgrid/UV_Grid_Sm.jpg"></a-plane>
<!--     <a-text value="Вітаємо у браузерній VR!" color="black" width="10" position="-0.5 1 -6"
       font="arial-msdf.json" negate="false"></a-text> -->
    <a-text value="Welcome to browser's VR" color="black" width="3" position="-1 0.5 -1"></a-text>
    </a-anchor>
    <a-camera-static/>
  </a-scene>
</html>
```

Після відкриття індексного файлу у веб-браузері, за наявності веб-камери, необхідно надати дозвіл на доступ до неї (рис. 2.20).

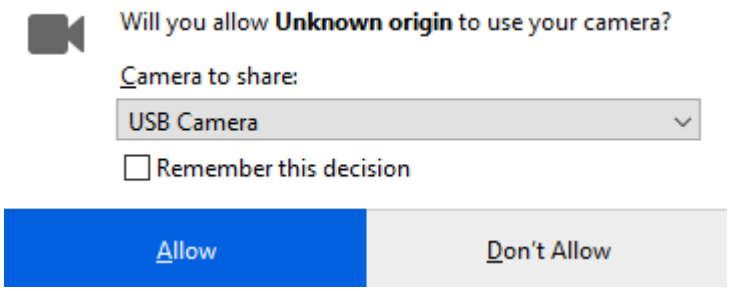

Рис. 2.20. Надання веб-браузеру дозволу на використання камери

Об’єкти на сцені з’являться, коли камера буде спрямована на маркер (рис. 2.21).

Файл маркеру за замовчанням можна відкрити на будь-якому мобільному пристрої або роздрукувати.

До параметру `arjs` можуть бути передані наступні значення:

`trackingMethod` – спосіб відстежування маркерів (за замовчанням `'best'`);

`debugUIEnabled` – показувати додаткові відомості для налагодження (за замовчанням `true`);

`debug` – вмикає налагоджувальний режим (за замовчанням `false`) (рис. 2.22);

`detectionMode` – тип маркеру (можливі `'color'`, `'color_and_matrix'`, `'mono'`, `'mono_and_matrix'`);

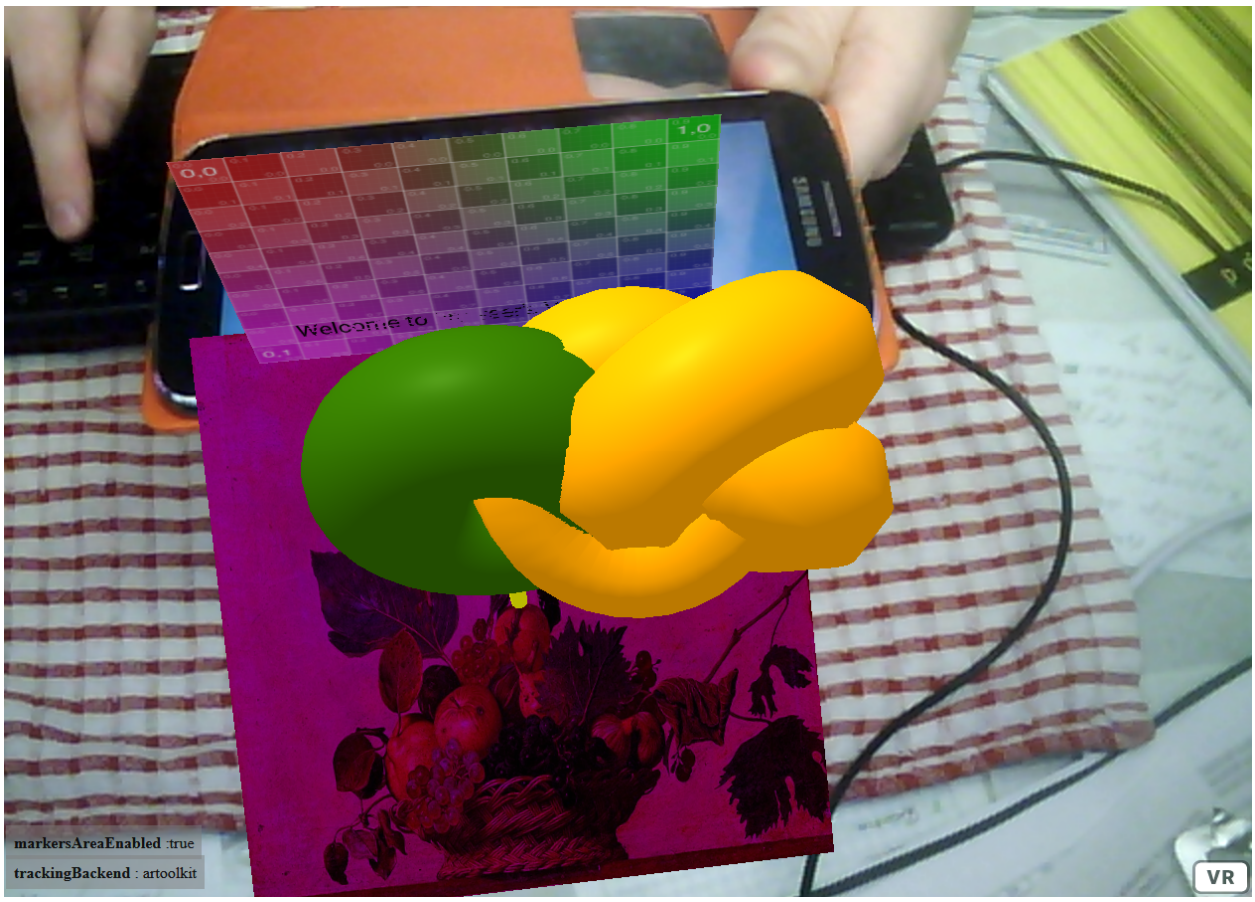

Рис. 2.21. Сцена у доповненій реальності

`matrixCodeType` – тип матричного коду для `detectionMode` `'color_and_matrix'` та `'mono_and_matrix'` (можливі `3x3`, `3x3_HAMMING63`, `3x3_PARITY65`, `4x4`, `4x4_BCH_13_9_3`, `4x4_BCH_13_5_5`);

`patternRatio` – співвідношення сторін для користувацьких маркерів (за замовчанням не використовується: -1);

`cameraParametersUrl` – посилання на параметри камери (наприклад, "AR.js-master\data\data\camera_para.dat");

`maxDetectionRate` – максимальна частота, з якою бібліотека намагається знайти маркер на зображенні з камери (за замовчанням не використовується: -1);

`sourceType` – джерело зображення (можливі `'webcam'`, `'image'`, `'video'`);

`sourceUrl` – посилання на джерело (якщо `sourceType` – `'image'` або `'video'`);

`sourceWidth` та `sourceHeight` задають роздільну здатність вихідного зображення (наприклад, 640x480), `displayWidth` та `displayHeight` – відповідно відображуваного зображення;

`deviceId` – опціональний ідентифікатор камери;

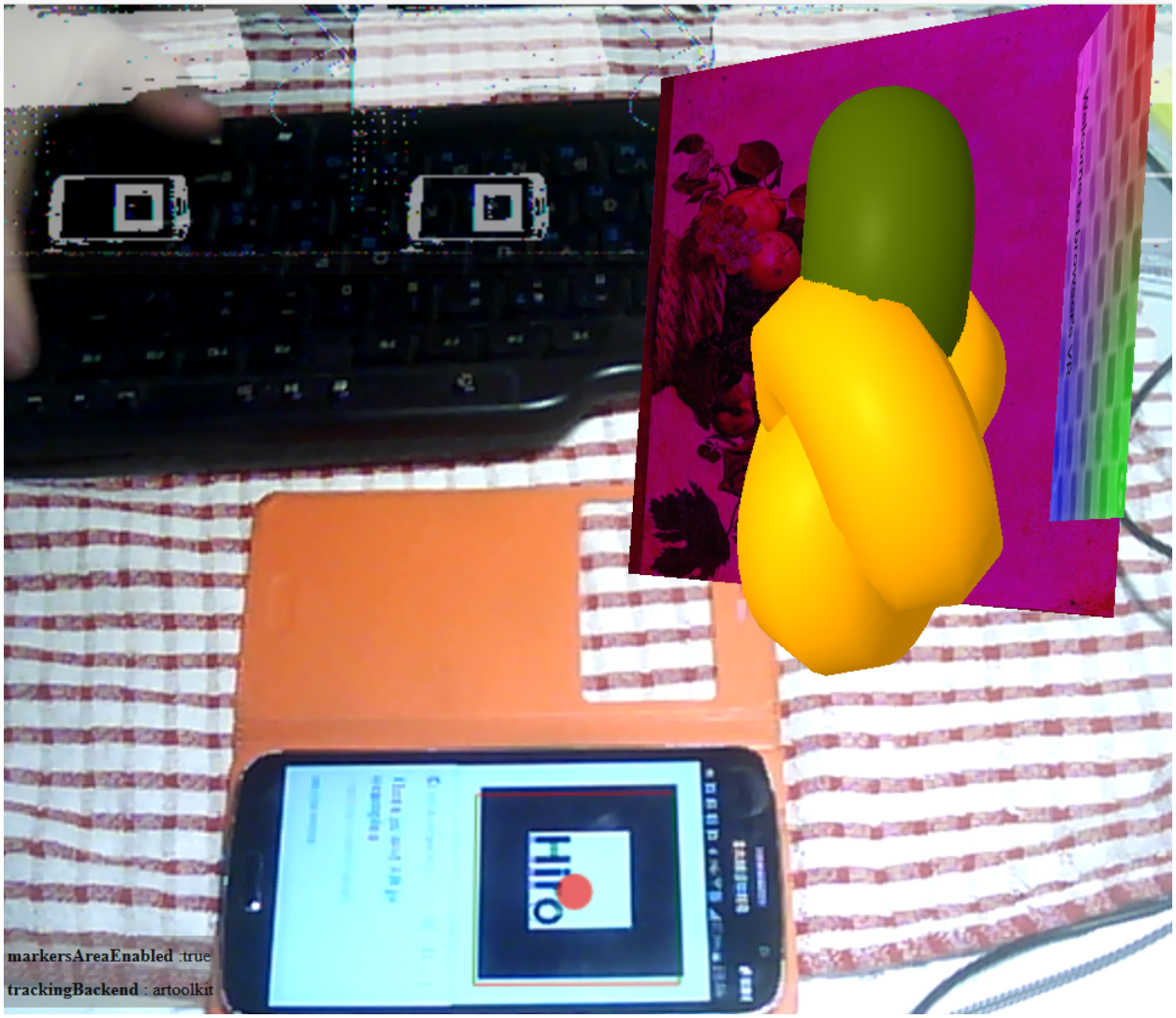

Рис. 2.22. AR.js у налагоджувальному режимі

Більшість із цих значень використовує найстаріша бібліотека для доповненої реальності – ARToolKit, явне використання якої може бути задано параметром `artoolkit` примітиву `a-scene`. Детальні керівництва та приклади створення веб-AR за допомогою AR.js можна знайти у репозитарії його автора – Жерома Етьєнна.

2.6 Об’єктна модель A-Frame

Використовуючи тег `script` із атрибутом `src`, бібліотеки (і взагалі будь-який текст мовою JavaScript) можна включати до індексного файлу як з локального розташування, так й з віддаленого. Наприклад, замість завантаження бібліотек A-Frame та AR.js можна було б послатися на їх розташування в мережі Інтернет. Якщо тег `a-anchor` виконує загальну прив’язку групи об’єктів, що він їх охоплює, до маркеру, то у даному прикладі явно вказано, який маркер буде використовуватись, за допомогою тегу `a-marker`: у такий спосіб різним маркерам можна співставити різні групи об’єктів:

```
<!doctype HTML>
<html>
<script src="https://aframe.io/releases/1.0.4/aframe.min.js"></script>
<script src="https://raw.githack.com/jeromeetienne/AR.js/2.1.8/aframe/build/aframe-ar.js"></script>
  <body style='margin : 0px; overflow: hidden;'>
    <a-scene embedded arjs="debugUIEnabled:false">
      <a-marker preset="hiro">
          <a-box position='0 0.5 0' material='color: red;'></a-box>
      </a-marker>
      <a-marker preset="kanji">
          <a-box position='0 0.5 0' material='color: blue;'></a-box>
      </a-marker>
      <a-entity camera></a-entity>
    </a-scene>
  </body>
</html>
```

У даному прикладі використовуються два стандартні маркери – "hiro" та "kanji" (рис. 2.23).

Крім того, одинарний тег `<a-camera-static/>` замінено на об’єкт `camera` за допомогою пари тегів `<a-entity camera></a-entity>`. Зауважимо, що, хоча у цьому прикладі різниця несуттєва, дана заміна не є однозначною. Головна її мета – показати, що JavaScript є об’єктно-орієнтованою мовою. Будь-який приклад, з якого розпочинається використання цієї мови, яскраво це ілюструє:

```
<!DOCTYPE html>
<html> <body>
      <script type="text/javascript">
         document.write("<font size=7>Hello World!</font>");
      </script>
```

```
    </body>
</html>
```

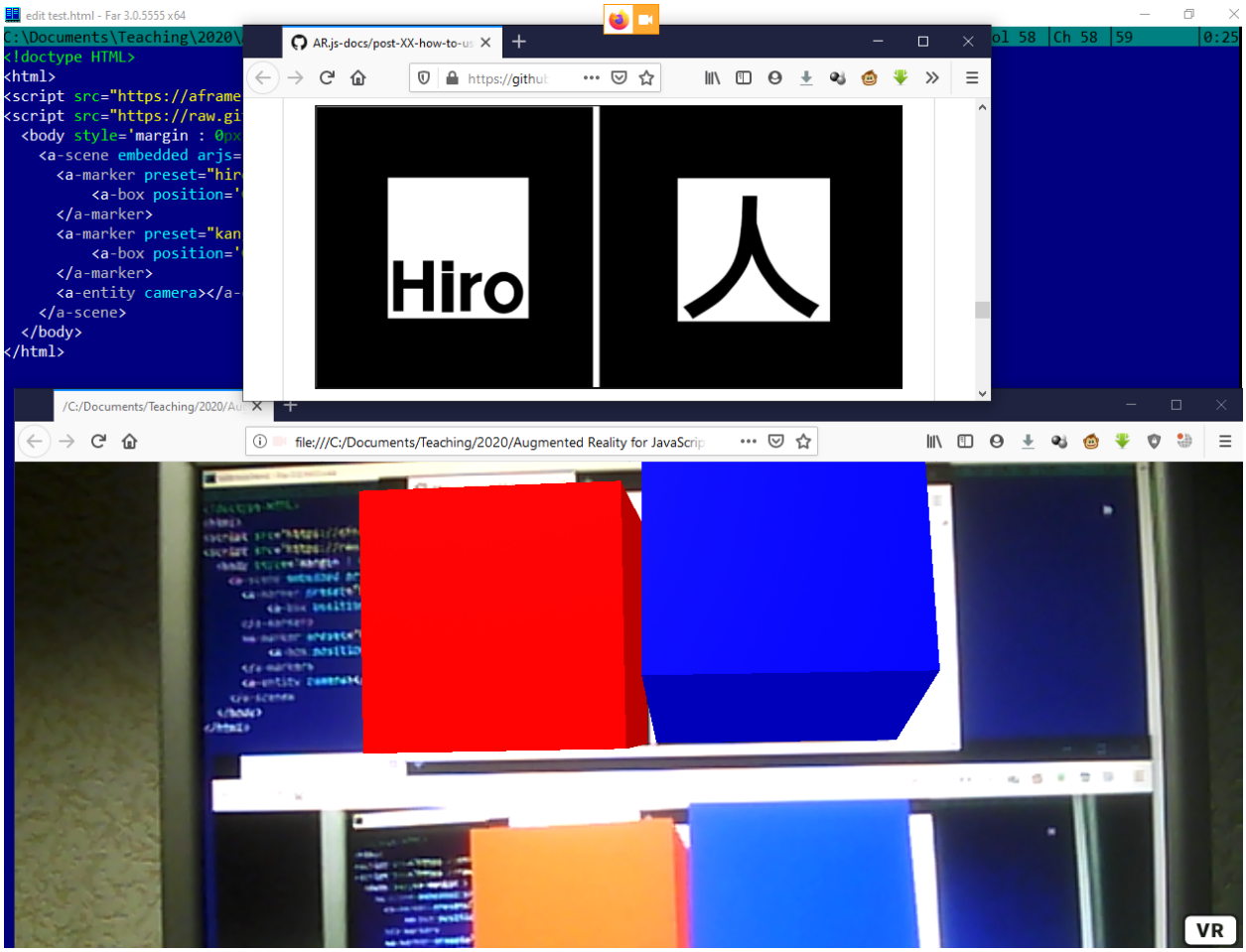

Рис. 2.23. Приклад використання двох маркерів

Тег `script` доповнений атрибутом `type`, у якому конкретизовано мову. Програма складається з одного єдиного рядка, в якому з об'єкту `document` викликається метод `write`, параметром якого є рядок, що з'являється у вікні браузера (рис. 2.24).

Hello World!

Рис. 2.24. Результат роботи найпростішої програми мовою JavaScript

Для початку застосування JavaScript достатньо знати мінімальний синтаксис цієї мови: способи створення змінних, коментарів, прості типи даних, спеціальні типи даних, основні оператори, способи визначення функцій,

умовний вираз, оператор вибору, цикли, примітивні та об’єктні типи даних. Все, що стосується синтаксису JavaScript/ECMAScript, можна знайти у багатьох джерелах – наприклад, якісних відеолекціях Дугласа Крокфорда (Douglas Crockford) [9].

A-Frame використовує ECS (Entity – Component – System) – шаблон проектування комп’ютерних ігор, основними поняттями якого є Entity (сутність), Component (компонент) та System (система). Сутність – це контейнер для компонентів. Сутності є основою всіх об’єктів на сцені, але без компонентів сутності нічого не роблять і не надають. Компонент – це невеликий об’єкт, який реалізує певну структуру даних та відповідає за окрему частину логіки роботи програми. Кожен тип компонента можна прикріпити до сутності, щоб надати їй певної властивості. Системи управляють набором сутностей, об’єднаних деякими компонентами. Вони не є обов’язковими.

У A-Frame цей шаблон проектування реалізований за допомогою атрибутів. Як сутностей використовуються будь-які примітиви A-Frame – `a-scene`, `a-box`, `a-sphere` та ін. Але особливе місце займає `a-entity`, ім’я говорить саме за себе. Всі інші примітиви є по суті обгортками для компонентів і зроблені для зручності, тому що будь-який елемент можна створити і за допомогою `a-entity`. Наприклад, примітив `a-box` можна реалізувати у такий спосіб:

```
<a-entity
  geometry="primitive: box; width: 1; height: 1; depth: 1">
</a-entity>
```

`geometry` у даному випадку є компонентом, який був доданий до сутності `<a-entity>`. Сам по собі `<a-entity>` не має будь-якої логіки (в глобальному сенсі), а компонент `geometry` по суті перетворює його на куб або що-небудь інше. Іншим, не менш важливим, ніж `geometry`, компонентом є `material`. Він додає до геометрії матеріал. Матеріал відповідає за те, чи буде наш куб блищати як метал, чи буде мати будь-які текстури та ін.

Будь-компонент у A-Frame повинен бути зареєстрований глобально через

спеціальну конструкцію:

```
AFRAME.registerComponent('hello-world', {
  init: function () {
    console.log('Hello, World!');
  }
});
```

Створений компонент можна додати на сцену, як і будь-який інший елемент:

```
<a-entity hello-world></a-entity>
```

Створення компоненту приводить до виклику методу `init`, що виводить повідомлення до консолі веб-браузера. У компоненті можуть бути визначені також методи:

`update` – викликається при ініціалізації разом з `init` та при оновленні будь-якого властивості компонента;

`remove` – викликається після видалення компонента або сутності, що його містить;

`tick` – викликається кожного разу перед відображенням чи оновленням (рендерингом) сцени;

`tock` – викликається після рендерингу сцени;

`play` – викликається кожного при поновленні рендерингу сцени;

`pause` – викликається при зупинці рендерингу сцени;

`updateSchema` – викликається кожен раз після оновлення схеми.

Схема описує властивості компонента та визначається у такий спосіб:

```
AFRAME.registerComponent('my-component', {
  schema: {
    arrayProperty: {type: 'array', default: []},
    integerProperty: {type: 'int', default: 5}
  }
}
```

У даному випадку компонент `my-component` буде містити дві властивості – `arrayProperty` та `integerProperty`. Щоб передати їх до компонента, потрібно задати значення відповідного атрибута.

```
<a-entity my-component="arrayProperty: 1,2,3; integerProperty: 7"></a-entity>
```

Отримати ці властивості всередині компонента можна через властивість `data`. Отримати властивість `data` компонента із сутності, до якої він доданий, можна за допомогою `getAttribute`, а за допомогою `setAttribute` – встановити властивість у певне значення.

Системи у A-Frame задаються параметрами `a-scene` та реєструються через `AFRAME.registerSystem (name, definition)` – саме так бібліотекою Ar.js зареєстрована система `arjs`. На відміну від компонентів, системи надають лише методи `init`, `play`, `pause`, `tick` та `tock`.

A-Frame активно використовує об'єктну модель веб-браузера:

а) доступ до будь-якого об'єкту A-Frame може бути отриманий через `document.querySelector`, `document.getElementById` тощо;

б) різні компоненти можуть обмінюватись повідомленнями, для чого один з них повинен генерувати повідомлення за допомогою функцією `emit` (приймає три параметри: назву події; дані, які треба передати; ознака спливання (bubbling) події), а інший – обробляти («прослуховуючи» чергу події) за допомогою методу, визначеного у `addEventListener`;

в) до об'єктів A-Frame можуть бути застосовані методи `setAttribute` (надання атрибуту значення), `removeAttribute` (видалення атрибуту), `createElement` (створення елементу) та `removeChild` (видалення).

Постійно поповнюваний перелік нових компонентів A-Frame, розроблених користувачами та оформлених у пакети npm, доступний за посиланням https://www.npmjs.com/search?q=aframe-component

Для того, щоб застосувати компонент із списку, необхідно традиційно перейти до його репозитарію, завантажити та підключити відповідний файл. Інший спосіб – скористатись сервісом unpkg, який надає можливість швидко завантажити будь-який файл з npm-пакету за посиланням вигляду

unpkg.com/:package@:version/:file

В якості прикладу скористаємось компонентом для побудови 3D-

поверхонь:

```
<script   src="https://unpkg.com/aframe-plot-component/dist/aframe-plot-component.min.js"> </script>
```

Для його використання замінимо блакитний куб, що пов’язаний із другим маркером, на поверхню (рис. 2.25):

```
<a-entity plot="function: ((3*x)^2 - (4*y)^2)/4;
                order: 32;
                show_zero_planes: true;
                bounds: -0.5 0.5 -0.5 0.5 -0.5 0.5;
                color: #04F"> </a-entity>
```

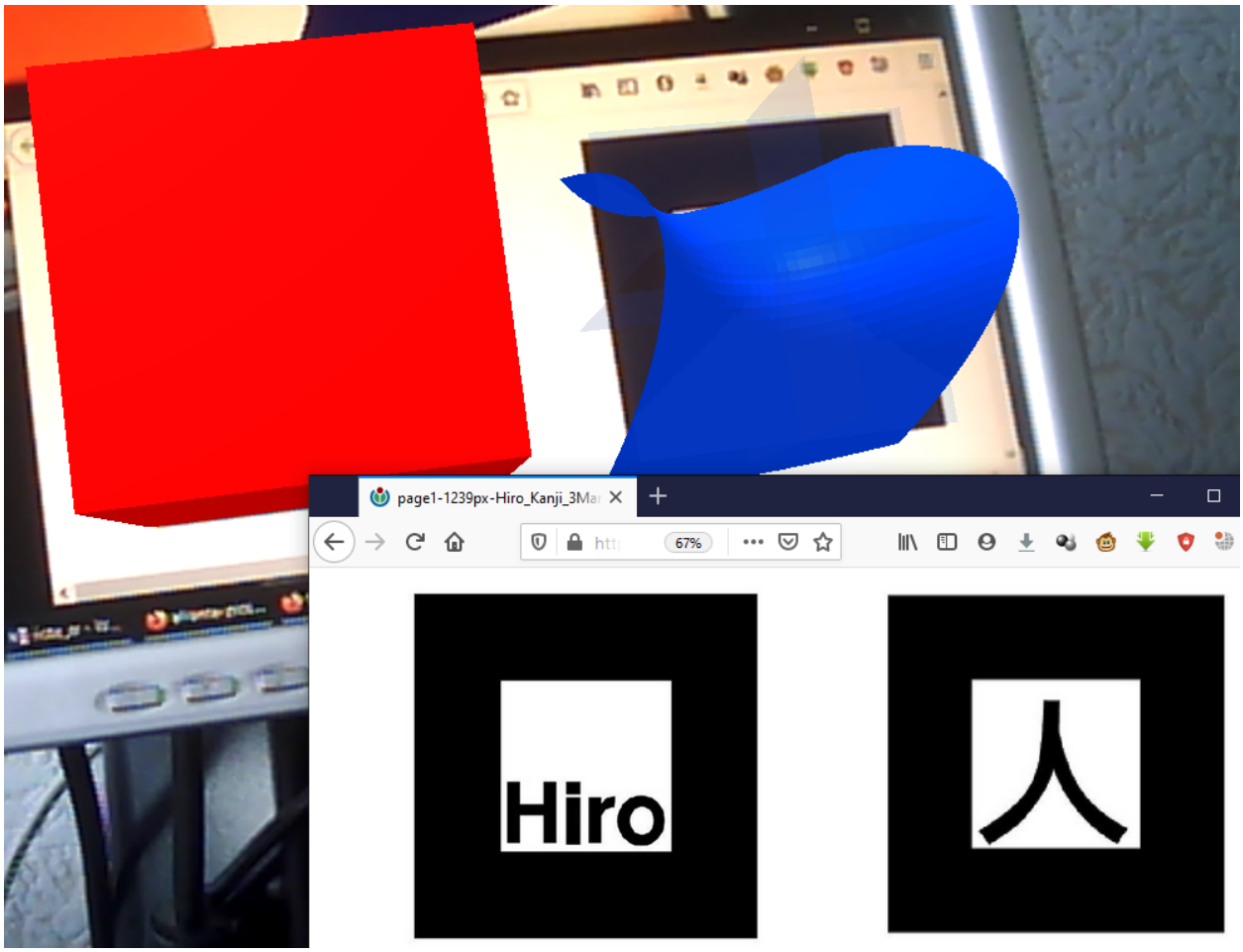

Рис. 2.25. Сцена із компонентом «3D-поверхня»

2.7 Застосування квадратних маркерів

Тег `<a-marker>` має такі основні атрибути:

`type` – тип маркеру: `'pattern'`, `'barcode'`, `'unknown'`;

`size` – розмір маркеру в метрах;

`url` – посилання на шаблон маркеру, якщо його тип – `'pattern'`;

`value` – значення коду, якщо тип маркеру – `'barcode'`;

`preset` – вибір стандартного маркеру (`'hiro'`, `'kanji'`);

`emitevents` – генерування подій `'markerFound'` та `'markerLost'`, якщо встановлений у `'true'`;

`smooth` – вмикає/вимикає згладжування зображення з камери (за замовчанням – `'false'`).

Тип `'barcode'` відповідає двовимірному (матричному) коду ARToolKit. У репозитарії поточного розробника AR.js Ніколо Капріньолі за посиланням https://github.com/nicolocarpignoli/artoolkit-barcode-markers-collection можна знайти зображення для всіх типів матричних кодів, що задаються атрибутом `matrixCodeType` параметру arjs примітиву a-scene. Чим простіше матричний код, тим легше його розпізнати, але й тим менше маркерів можна створити за його допомогою: `3x3_HAMMING63` підтримує 8 маркерів (відстань Хеммінга – 3), `3x3_PARITY65` – 32 (1), `4x4_BCH_13_5_5` – 32 (5), `4x4_BCH_13_9_3` – 512 (3), `5x5_BCH_22_7_7` – 128 (7), `5x5_BCH_22_12_5` – 4096 (5). Чим більше відстань Хеммінга, тим краще маркер розпізнаватиметься.

Приклад налаштування сцени для використання матричних кодів (рис. 2.26):

```
<a-scene vr-mode-ui="enabled: false" embedded arjs="debugUIEnabled:false; detectionMode: mono_and_matrix; matrixCodeType: 3x3;">
```

Відключення компонента `vr-mode-ui` надає можливість прибрати стандартні іконки A-Frame для переходу до повноекранних VR/AR режимів. Сам маркер створюється так:

```
<a-marker type="barcode" value='7>' ... </a-marker>
```

Для опрацювання подій від клавіатури та миші необхідно вказати відповідний обробник події та підключити курсор до камери. У наступному прикладі показується, як можна анімувати об’єкт за подіями від миші, курсор

якої має форму чорного кільця (рис. 2.27):

```
<a-scene vr-mode-ui="enabled: false" embedded arjs='sourceType: webcam; debugUIEnabled: false; detectionMode: mono_and_matrix; matrixCodeType: 3x3;'>
  <a-marker type='barcode' value='6'>
    <a-entity geometry="primitive: box" material="color: red" animation__mouseenter="property: rotation; from:0 0 0; to: 0 0 360; startEvents: mouseenter; dur: 1000"; animation__mouseleave="property: components.material.material.color; type: color; from: blue; to: red; startEvents: mouseleave; dur: 1000";> </a-entity>
  </a-marker>
  <a-entity camera><a-cursor></a-cursor></a-entity>
</a-scene>
```

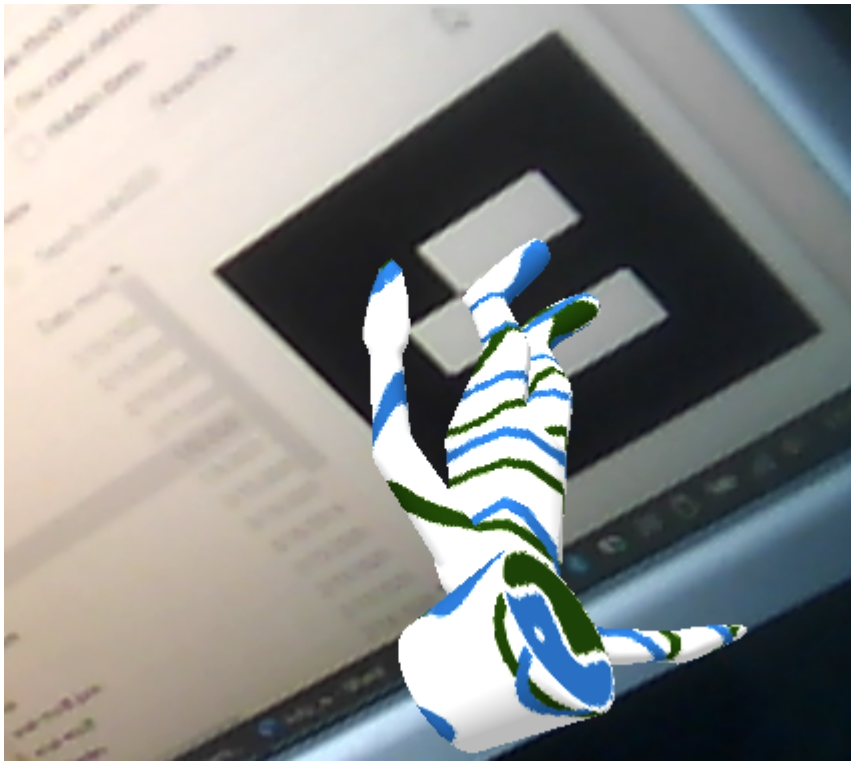

Рис. 2.26. Модель, розташована на матричному маркері

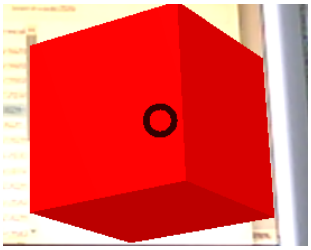

Рис. 2.27. Модель, керована користувачем

Інший спосіб – явне використання обробників подій через виклик `addEventListener`. Наприклад, на рівні вікна веб-браузера можна визначити параметри підключеної камери під час її ініціалізації або подію припинення відеопотоку з камери:

```
window.addEventListener('camera-init', (data) => {
    console.log('camera-init', data); })
window.addEventListener('camera-error', (error) => {
    console.log('camera-error', error); })
```

Якщо необхідно опрацьовувати події, пов’язані із певною сутністю A-Frame, доцільно увести до неї новий компонент та зареєструвати його. Наприклад, компонент `registerevents`, доданий до маркера, допоможе розібратись, коли він потрапляє до поля зору камери (розпізнається), а коли – виходить з поля зору («губиться»):

```
AFRAME.registerComponent('registerevents', {
  init: function () {
    var marker = this.el;
    marker.addEventListener('markerFound', function() {
      console.log('markerFound', marker.id); });
    marker.addEventListener('markerLost', function() {
      console.log('markerLost', marker.id); });
  }
});
```

Приклад компонента для видимої сутності, що опрацьовує натискання кнопки миші на ньому:

```
AFRAME.registerComponent('jump', {
  init: function () {
    var obj = this.el;
    obj.addEventListener('click', function () {
    this.setAttribute('position', (Math.random()-0.5)+ " 0.5 0");
    });
  }
});
```

Приклад сцени з трьома незалежними маркерами, об’єкти яких реагують

на натискання миші переміщенням (рис. 2.28):

```
<a-marker preset="hiro" id='marker-hiro' registerevents>
  <a-box position='0 0.5 0' material='color: red;' jump></a-box>
</a-marker>
<a-marker preset="kanji" id='marker-kanji' registerevents>
  <a-box position='0 0.5 0' material='color:blue;' jump></a-box>
</a-marker>
<a-marker type='barcode' value='6' id='marker-barcode-7'
          registerevents>
  <a-sphere position='0 0.5 0' radius="0.5"
            material='color: green;' jump></a-sphere>
</a-marker>
<a-entity camera><a-cursor></a-cursor></a-entity>
```

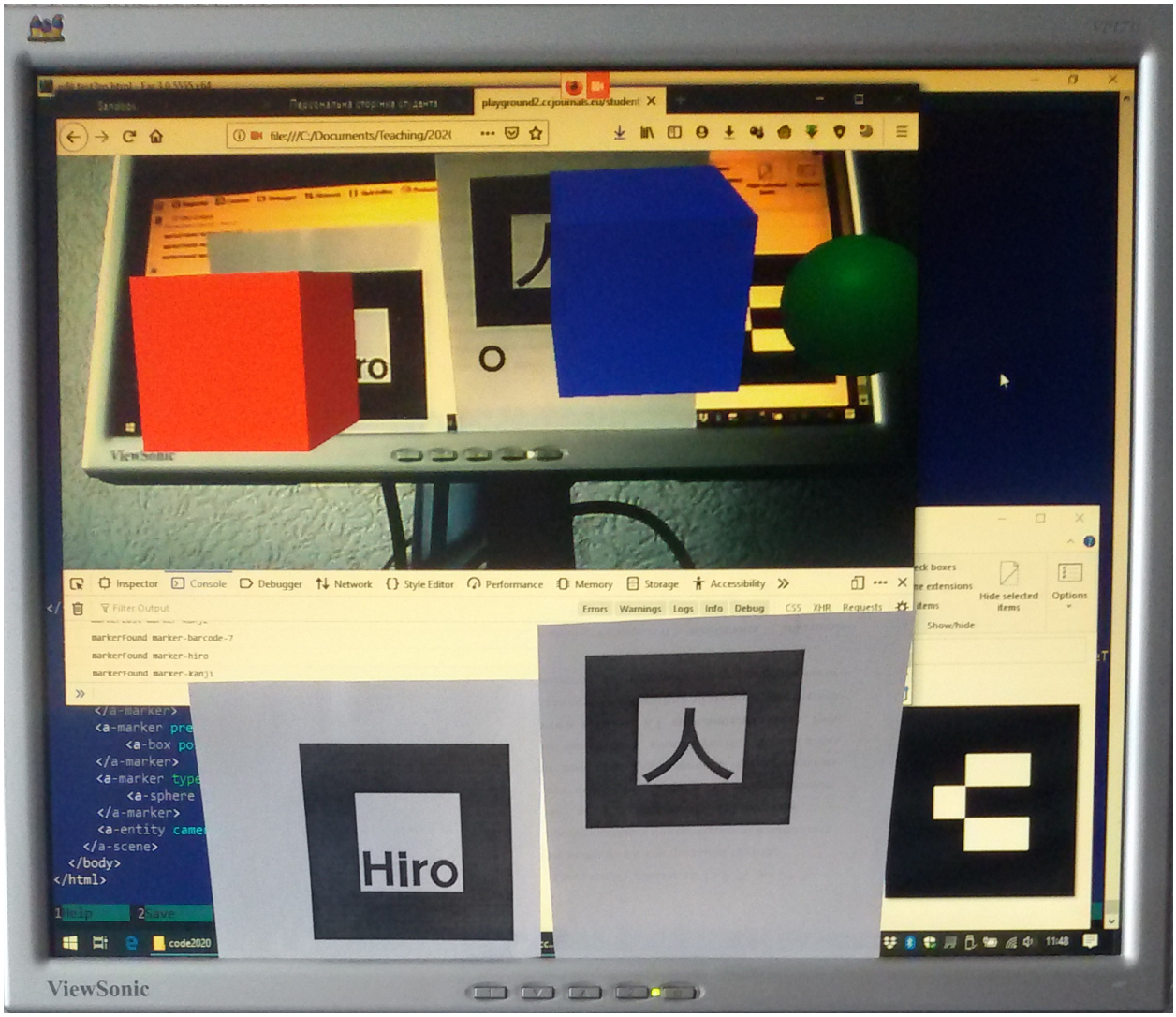

Рис. 2.28. Сцена із трьома керованими маркерами

AR.js надає можливість використання усіх стандартних маркерів, що підтримує ARToolKit. Крім вже використаних 'hiro' та 'kanji', це 'letterA', 'letterB', 'letterC', 'letterD', 'letterF', 'letterG' та ряд інших. Частина маркеру, що детектується – чорний квадрат зі стороною 1, усередині якого розміщено зображення. Опис розміщується у текстовому файлі (як правило, його ім'я звершується на .patt) – саме він використовується при розпізнаванні, а не звичне зображення (рис. 2.29).

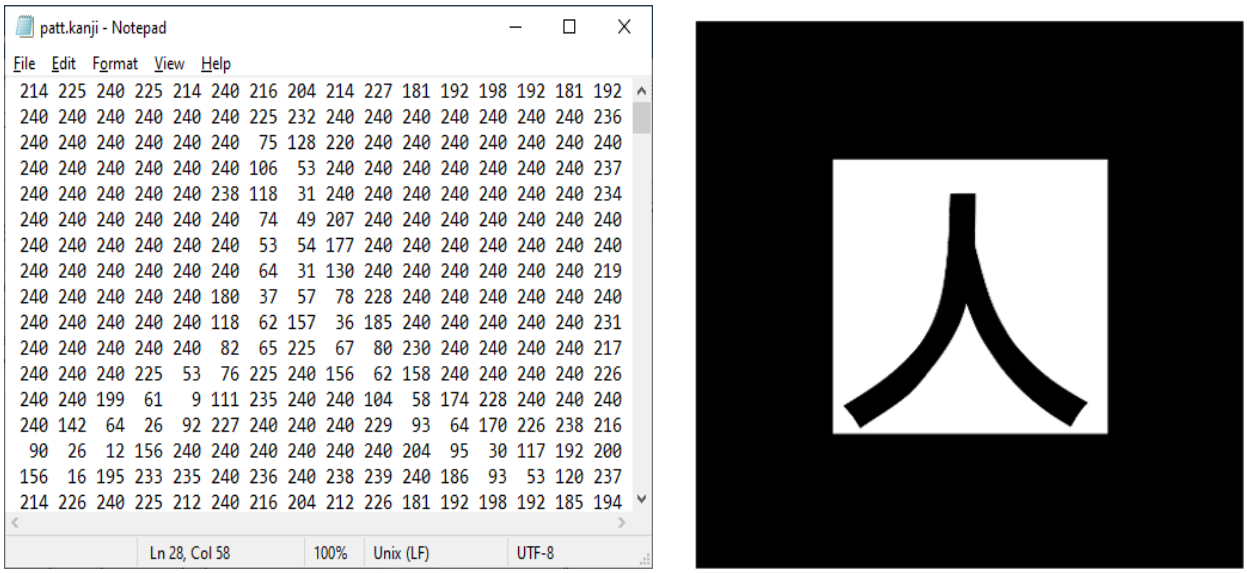

Рис. 2.29. Внутрішнє (фрагмент) та зовнішнє подання маркеру 'kanji'

Усього файл шаблону маркера зберігає 12 його зображень у 4 орієнтаціях. Створення маркеру, що гарно розпізнається – непроста задача: яким би не був розмір зображення, для використання ARToolKit воно буде закодоване усього 256 елементами (16 рядків на 16 стовпців), тому надмірно деталізоване зображення буде неминуче спрощене. Приклад гарного маркеру поданий на рис. 2.30.

Рис. 2.30. Зовнішнє подання користувацького маркеру 'pi'

Наведемо його кодування (перше зображення):

```
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 242  95  11   0   0   0   0   0   0 114 255 255 255
255 255 255 101  72  82   0 141 149   0   1 159 202 255 255 255
255 255 255 232 253 141   0 226 239   0   2 255 255 255 255 255
255 255 255 255 255 135   0 226 239   0   2 255 255 255 255 255
255 255 255 255 255 119   0 226 239   0   2 255 255 255 255 255
255 255 255 255 255  90   0 226 241   0   2 254 255 255 255 255
255 255 255 255 255  38   0 226 254  37   0  90 182 255 255 255
255 255 255 255 255 195 191 247 255 229 164 210 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
```

Простий спосіб створити власний маркер – скористатись генератором маркерів https://jeromeetienne.github.io/AR.js/three.js/examples/marker-training/examples/generator.html (рис. 2.31).

Отриманий текстовий файл із описом маркеру можна завантажити разом із його зображенням та використати для розпізнавання:

```
<a-marker type='pattern' url='pattern-pi.patt'>
...
</a-marker>
```

`a-marker` використовує статичну камеру, що розташована на початку координат та напрямлена у від'ємний бік вісі *z*. Видимість об'єктів, пов'язаних із маркером, встановлюється залежно від того, чи розпізнаний маркер камерою. Якщо один маркер використовується для того, щоб ініціювати певний процес, доцільно скористатись `a-marker-camera`. Після виходу маркеру з поля зору камери створені об'єкти прив'язуються до переміщення камери.

Для покращення розпізнавання доцільно використовувати асиметричні маркери, наприклад:

Д,Б!

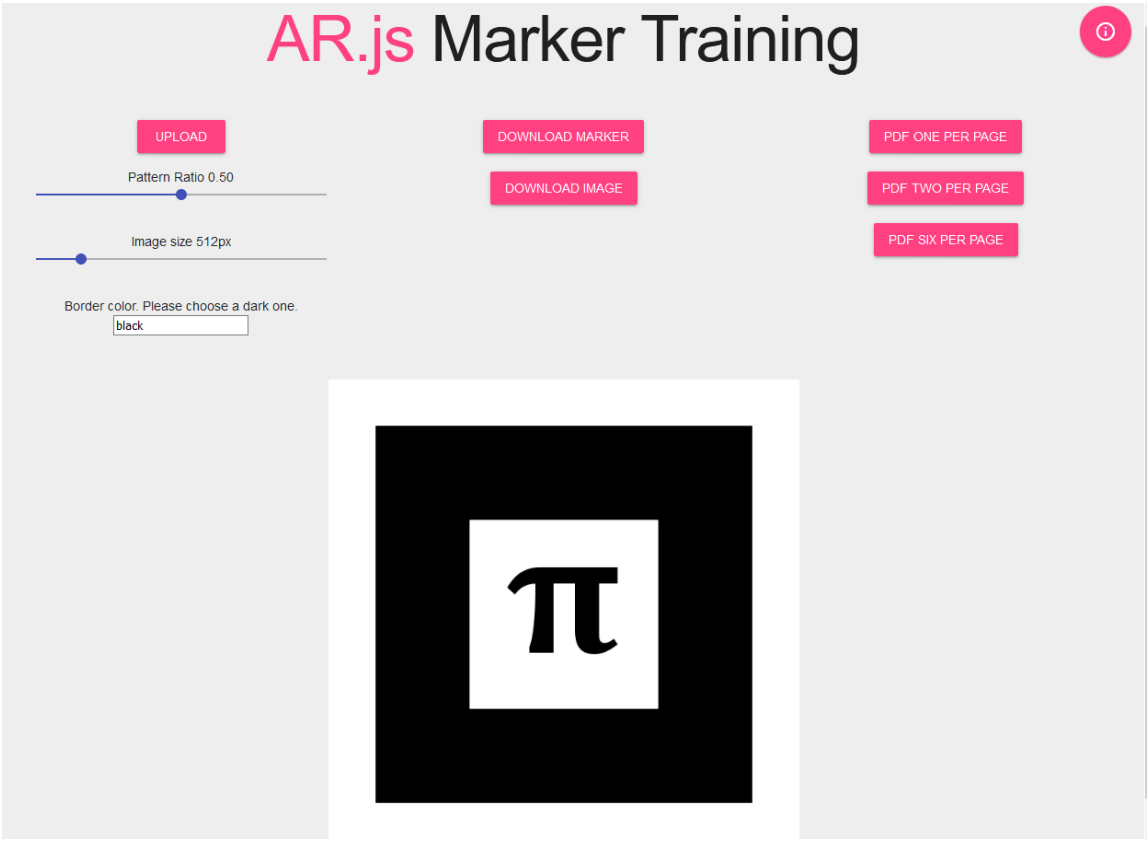

Рис. 2.31. Використання генератору користувацьких маркерів

Ще один спосіб покращити розпізнавання – використання друкованих маркерів достатнього розміру. Вирізаючи маркер, залишайте біле поле шириною принаймні в половину чорної рамки – контраст є суттєвим при розпізнаванні. Якщо й це не допомагає, можна збільшити роздільну здатність камери у компоненті `arjs` – за замовчанням вона 640x480. Наприклад, для FullHD камери можна вказати `arjs="sourceType: webcam; sourceWidth:1280; sourceHeight:960; displayWidth: 1280; displayHeight: 960; "`.

У лютому 2020 року Ніколо Капріньолі анонсовано створення AR.js Studio, а також підтримка NFT (Natural Feature Tracking) – використання довільних зображень в якості маркерів (https://carnaux.github.io/NFT-Marker-Creator/) – у третій версії AR.js.

Висновки до розділу 2

1. Для опанування початків розробки програмного забезпечення із доповненою реальністю для Web доцільним є спільне використання A-Frame та AR.js, причому проектуванню сцени у доповненій реальності має передувати її апробація у віртуальній реальності.

2. У міру збільшення складності сцени постає питання її керованості, що актуалізує застосування JavaScript для створення об’єктів A-Frame та опрацювання подій. Розробка мультимаркерної сцени потребує застосування різних типів матричних маркерів, що надає JSARToolKit, а спільна робота моделей, прив’язаних до різних маркерів, потребує застосування засобів бібліотеки Three.js. Це надає процесу навчання розробки програмного забезпечення із доповненою реальністю для Web діалектичної природи: уведення нових засобів відбувається для подолання проблем, що не можуть бути розв’язані із використанням наявних.

3. Технологічними умовами навчання розробки програмного забезпечення із доповненою реальністю для Web є наявність довільної операційної системи, доступ до мережі Інтернет, можливість використання хмарних сховищ для розміщення власних матеріалів та наявність принаймні двох класів пристроїв для тестування програм.

РОЗДІЛ 3
РОЗРОБКА ПРОТОТИПУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРОФОРІЄНТАЦІЙНИХ AR-КВЕСТІВ

3.1 Поняття про профорієнтаційний квест

Квест (з англ. quest) – мандрівка з метою виконання певної місії або досягнення цілі. У літературних творах квест найчастіше використовується для того, щоб показати зростання його героя – він змінюється сам та змінює інших. В. Я. Пропп вказує, що в усіх таких казкових історіях (або, більш правильно – tales of quests) є загальна структура, що складається із близько 150 кроків [20, с. 134].

Федерація квесту України [34] визначає квест як аматорське спортивно-інтелектуальне змагання, основою якого є послідовне виконання заздалегідь підготовлених завдань командами або окремими гравцями.

Під час гри команди вирішують логічні завдання, здійснюють пошук на місцевості, будують оптимальні маршрути переміщення, шукають оригінальні рішення і підказки. Після завершення чергового завдання команди переходять до виконання наступного. Перемагає команда, що виконала всі завдання швидше за інших.

Під час профорієнтаційних заходів на території Криворізького державного педагогічного університету організуються пішохідні квести (як у приміщенні, так й на відкритому повітрі). При проходженні квесту його організаторам необхідно контролювати час проходження, виконання завдань, виконувати перевірку присутності команди на відповідному завданні та контролювати коди підтвердження проходження завдання. Учасникам квесту пропонується розв’язати інтелектуальні та пошукові завдання в ігровій формі, пов’язані із майбутньою професією, тому надалі такий тип квестів називатимемо профорієнтаційним.

3.2 Засоби WebAR для реалізації профорієнтаційного квесту

Профорієнтаційний квест має проходити по наперед заданому маршруту, на якому необхідно фіксувати проходження команд по станціях. Наявність активних GPS-навігаторів на мобільних пристроях учасників квесту не лише допоможе відслідкувати проходження, а й надає можливість їм зібратись у певній точці маршруту, ознакою якої є прив'язаний до географічних координат об'єкт доповненої реальності.

Прототип коду, що виконує такі дії, поданий у [3]:

```
<script
src="https://aframe.io/releases/1.0.4/aframe.min.js"></script>
<script src="https://unpkg.com/aframe-look-at-
component@0.8.0/dist/aframe-look-at-component.min.js"></script>
<script src="https://raw.githack.com/AR-js-
org/AR.js/master/aframe/build/aframe-ar-nft.js"></script>
<a-scene vr-mode-ui="enabled: false" embedded
  arjs="sourceType: webcam; debugUIEnabled: false;">
  <a-text
  value="ТЕКСТ ІЗ ПОЗНАЧКОЮ СТАНЦІЇ ТА НАСТУПНОГО ЗАВДАННЯ"
  look-at="[gps-camera]"
  scale="120 120 120"
  gps-entity-place="latitude: 47.908387; longitude: 33.411848;">
</a-text>
  <a-camera gps-camera rotation-reader> </a-camera>
</a-scene>
```

На рис. 3.1 показано місце на площі перед головний корпусом університету, що відповідає наведеним у коді координатам. На жаль, використання виключно GPS-локації для визначення досягнення командою станції (проміжного етапу квесту, за який нараховуються бали), не завжди можливо – у багатоповерхових будівлях, якими є майже усі корпуси університету, одна й та сама координата буде відповідати станціям на різних поверхах.

Для точного визначення станції доцільним є поєднання декількох типів

доповненої реальності – координатної та квадратної маркерної (aframe-ar.js – версія AR.js 3 для даної комбінації) або координатної та NFT-маркерної (aframe-ar-nft.js – версія AR.js 3 для цієї комбінації).

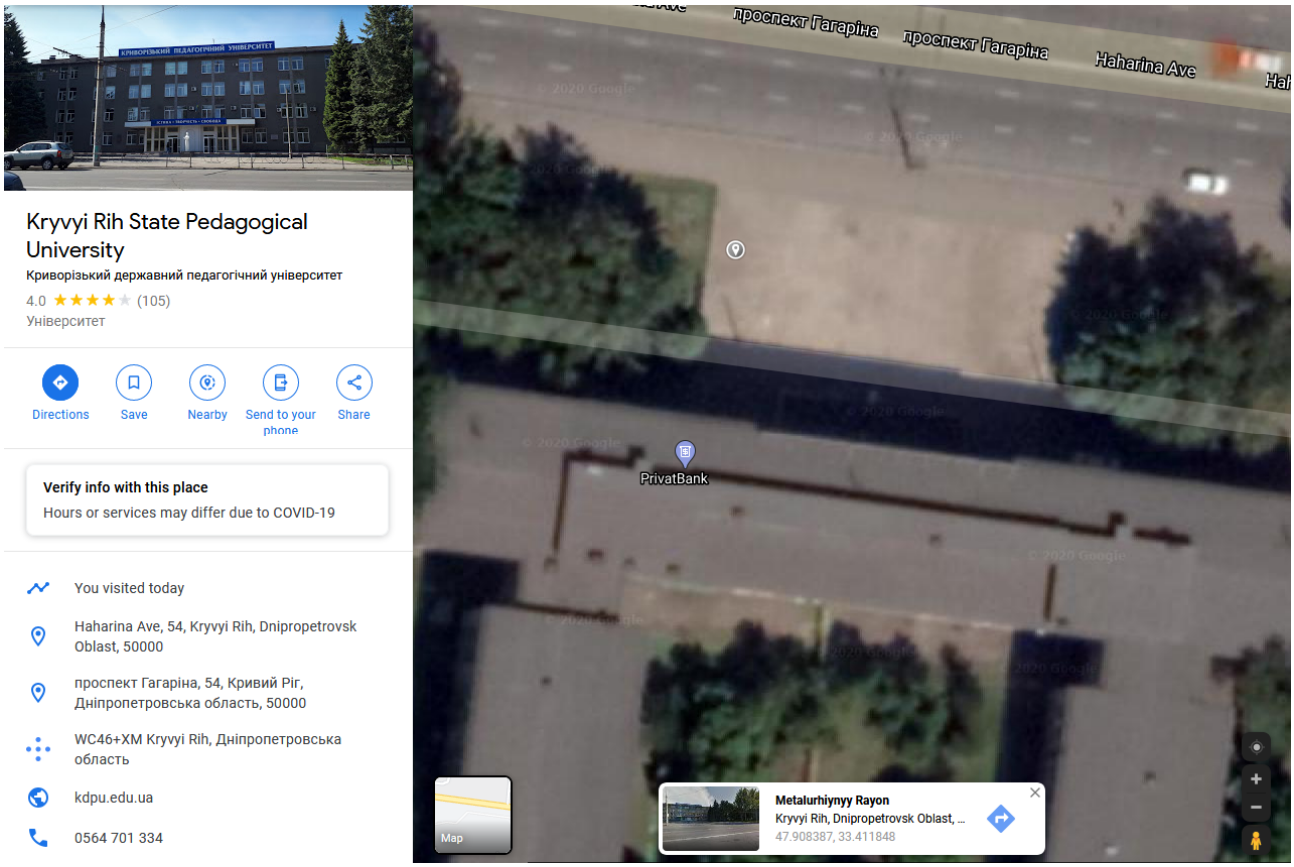

Рис. 3.1. Визначення координат квест-станції за допомогою Google Maps

Традиційний квадратний маркер може бути легко встановлений, але сам факт його наявності сигналізує учасникам квесту про те, що на них чекає завдання. Для уникнення можливості фальсифікації результатів шляхом перенесення маркеру в інше місце й застосовується додаткове визначення координат.

Зробити квести більш трудним, але й більш цікавим надає можливість Ar.js 3 відслідковувати природні зображення, які можуть бути частиною інтер'єру. Для використання таких зображень в якості NFT-маркерів доцільно скористатись генератором за посиланням https://carnaux.github.io/NFT-Marker-Creator/ та ознайомитись із рекомендаціями зі створення гарних маркерів [8].

NFT Marker Creator (рис. 3.2) має вимірювач впевненості (confidence),

мета якого – показати, чи буде маркер гарним чи поганим: чим більшу кількість зірок він показує, тим краще буде розпізнаватись маркер.

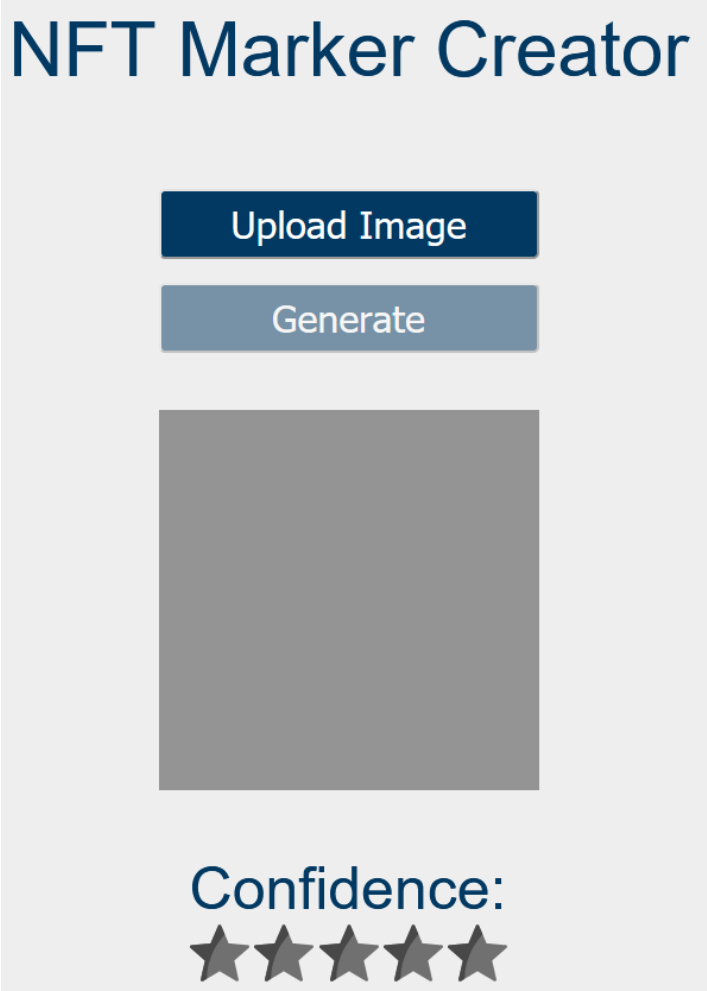

Рис. 3.2. NFT Marker Creator

За гарного освітлення та камери впевнено розпізнаються й зображення, для яких рівень впевненості дає одну зірку, проте краще добирати зображення, що даватимуть 4 або 5 зірок. Таке зображення повинно бути візуально складним та мати високу роздільну здатність (на відміну від квадратних маркерів, тут деталі мають значення). Візуально складне зображення надає програмному забезпеченню багато можливостей для відстеження унікальних частин зображення, що легко розпізнаються.

Від фізичного розміру маркера також залежить якість його розпізнавання – до малих за розміром зображень мобільний пристрій необхідно наблизити, у той час як від великих навпаки – тримати подалі.

Якість розпізнавання також залежить від освітленості екрану мобільного пристрою; крім того, камери з низькою роздільною здатністю зазвичай працюють краще, коли вони знаходяться близько до маркерів.

При використанні друкованих маркерів вони мають бути розміщені або на непрозорому паперів, або на стенді. Останні є природним джерелом маркерів, адже усі факультети університету мають власні стенди.

3.3 Розробка та тестування програмного забезпечення

Розглянемо створення прототипу програмного забезпечення фрагменту квесту, пов’язаним із стендом фізико-математичного факультету Криворізького державного педагогічного університету (рис. 3.2).

Рис. 3.2. Колективна фотографія кафедри інформатики та прикладної математики перед стендом фізико-математичного факультету

Рис. 3.2 дає найвищий коефіцієнт упевненості – 5 зірок. Після друку на

непрозорому папері кольорова гама зображення зміниться, що призведе до зменшення коефіцієнта до 2 зірок (рис. 3.3).

а б

Рис. 3.3. Генерування маркера для оригінального (а) та сканованого після друку (б) фотографічного зображення

Порівняння зображень на рис. 3.3а та рис. 3.3б надає можливість зробити додатковий висновок про важливість зовнішнього освітлення для корекції балансу білого та якісного розпізнавання NFT-маркерів.

Після опрацювання зображення генератор створює 3 файли маркеру (для даного випадку – kafera.fset, kafera.fset3 та kafera.iset) – всі вони є необхідними для розпізнавання. Останній файл містить оригінальне зображення, перетворено на відтінки сірого (рис. 3.4б).

У наступному фрагменті коду після розпізнавання фотографічного NFT-маркеру з’являються відомості про осіб, зображених на фотографії (рис. 3.5) як об’єкт доповненої реальності (площина з текстурою), що активується натисканням на ньому – у новому вікні браузера відкривається посилання на сторінку кафедри інформатики та прикладної математики (рис. 3.6, 3.7).

а б

Рис. 3.4. Зображення високої якості (а) та внутрішнє подання зображення (б)

Стоять (зліва направо):
1.Мерзлікін Олександр Володимирович - к.пед.н., асистент ;
2. Моісеєнко Наталя Володимирівна - к. ф. -м.н., доцент;
3. Моісеєнко Михайло Вікторович - старший викладач ;
4. Шокалюк Світлана Вікторівна- к.пед. н., доцент;
5. Медведєв Дмитро Геннадійович;
6. Степанюк Олександр Миколайович - асистент;
7. Семеріков Сергій Олексійович - д.пед.н., професор;
8. Юрко Олександр Володимирович - старший викладач;
9. Міненко Павло Олександрович - д.ф. -м.н., професор;
10. Мерзліик Павло Володимирович - к.ф.-м.н., доцент ;
11. Соловйов Володимир Миколайович -д.ф.-м.н., професор,завідувач кафедри
Сидять(зліва направо):
12. Мінтій Іріна Сергіївна - к.пед. н.,доцент;
13. Шасмутдінова Наталя Олександрівна - старший лаборант;
14. Закарлюка Ірина Станіславівна - старший викладач;
15. Хараджян Наталя Анатоліївна - к.пед.н.,доцент.

Рис. 3.5. Об’єкт доповненої реальності для NFT-маркеру

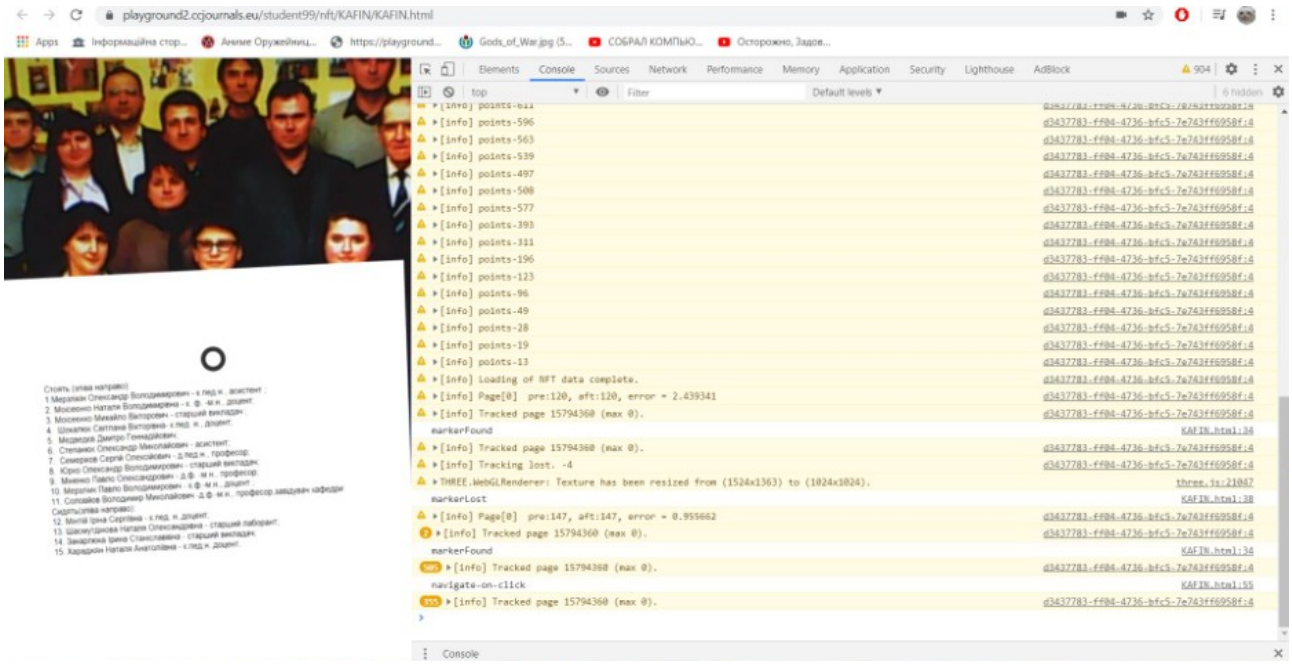

Рис. 3.6. Тестування програмного забезпечення у налагоджувальному режимі

Рис. 3.7. Робота програмного забезпечення на мобільному інтернет-пристрої

Код описаного фрагменту профорієнтаційного квесту подано у додатку А.

Висновки до розділу 3

1. Квест – аматорське спортивно-інтелектуальне змагання, основою якого є послідовне виконання заздалегідь підготовлених завдань командами або

окремими гравцями. Під час профорієнтаційних заходів на території Криворізького державного педагогічного університету організуються пішохідні квести (як у приміщенні, так й на відкритому повітрі). При проходженні квесту його організаторам необхідно контролювати час проходження, виконання завдань, виконувати перевірку присутності команди на відповідному завданні та контролювати коди підтвердження проходження завдання. Учасникам квесту пропонується розв'язати інтелектуальні та пошукові завдання в ігровій формі, пов'язані із майбутньою професією, тому такий тип квестів називається профорієнтаційним.

2. Профорієнтаційний квест має проходити по наперед заданому маршруту, на якому необхідно фіксувати проходження команд по станціях. Наявність активних GPS-навігаторів на мобільних пристроях учасників квесту не лише допоможе відслідкувати проходження, а й надає можливість їм зібратись у певній точці маршруту, ознакою якої є прив'язаний до географічних координат об'єкт доповненої реальності. Для точного визначення станції доцільним є поєднання декількох типів доповненої реальності – координатної та квадратної маркерної або координатної та NFT-маркерної.

Зробити квести більш трудним, але й більш цікавим надає можливість Ar.js 3 відслідковувати природні зображення, які можуть бути частиною інтер'єру, за допомогою NFT-маркерів. Такі зображення повинні бути візуально складними та мати високу роздільну здатність. Якість розпізнавання NFT-маркеру також залежить від його фізичного розміру, освітленості екрану мобільного пристрою, відстані пристрою до маркеру, роздільною здатністю камери, зовнішнього освітлення, яскравості екрану та якості паперу, на якому надруковано маркер.

3. Розроблені прототипи програмного забезпечення профорієнтаційного квесту надають можливість визначити як координати учасника квесту, прив'язавши до них об'єкти доповненої реальності, так само як і до будь-яких зображень фотографічної якості у просторі профорієнтаційного квесту.

ВИСНОВКИ

У процесі дослідження проблеми добору та апробації засобів розробки програмного забезпечення із доповненою реальністю для Web для початківців, що володіють основами веб-розробки, було розв'язане завдання порівняльного аналізу засобів розробки доповненої реальності для Web з метою добору засобів, доступних для початківців. Для дібраних засобів були розроблені інструкції з їх налаштування та використання, на основі яких розроблено прототипи програмного забезпечення із маркерною та безмаркерною доповненою реальністю для Web. Розв'язання поставлених завдань надало можливість досягнути його мету та зробити наступні висновки:

1. Огляд засобів розробки програмного забезпечення із доповненою реальністю для Web надав можливість рекомендувати для опанування початківцями такі комбінації засобів візуалізації комп'ютерних моделей у Web та засобів відстеження реальних об'єктів: а) A-Frame та AR.js – API для швидкого прототипування, значна частина програм з використанням яких є HTML-подібний код; б) Three.js та ARToolKit.js – для поглибленого рівня, що передбачає створення програм засобами JavaScript.

2. Технологічними умовами навчання розробки програмного забезпечення із доповненою реальністю для Web є наявність довільної операційної системи, доступ до мережі Інтернет, можливість використання хмарних сховищ для розміщення власних матеріалів та наявність принаймні двох класів пристроїв для тестування програм. Для опанування початків розробки програмного забезпечення із доповненою реальністю для Web доцільним є спільне використання A-Frame та AR.js, причому проектуванню сцени у доповненій реальності має передувати її апробація у віртуальній реальності. У міру збільшення складності сцени актуалізується застосування JavaScript для створення об'єктів A-Frame та опрацювання подій. Розробка мультимаркерної сцени потребує застосування різних типів матричних маркерів, що надає JSARToolKit, а спільна робота моделей, прив'язаних до

різних маркерів, потребує застосування засобів бібліотеки Three.js. Це надає процесу навчання розробки програмного забезпечення із доповненою реальністю для Web діалектичної природи: уведення нових засобів відбувається для подолання проблем, що не можуть бути розв’язані із використанням наявних.

3. Квест – аматорське спортивно-інтелектуальне змагання, основою якого є послідовне виконання заздалегідь підготовлених завдань командами або окремими гравцями. При проходженні квесту його організаторам необхідно контролювати час проходження, виконання завдань, виконувати перевірку присутності команди на відповідному завданні та контролювати коди підтвердження проходження завдання. Якщо учасникам квесту пропонується розв’язати інтелектуальні та пошукові завдання в ігровій формі, пов’язані із майбутньою професією, такий тип квестів називається профорієнтаційним. Профорієнтаційний квест має проходити по наперед заданому маршруту, на якому необхідно фіксувати проходження команд по станціях. Наявність активних GPS-навігаторів на мобільних пристроях учасників квесту не лише допоможе відслідкувати проходження, а й надає можливість їм зібратись у певній точці маршруту, ознакою якої є прив’язаний до географічних координат об’єкт доповненої реальності. Для точного визначення станції доцільним є поєднання декількох типів доповненої реальності – координатної та квадратної маркерної або координатної та NFT-маркерної. Якість розпізнавання NFT-маркеру залежить від роздільної здатності та ступеня деталізації зображення, його фізичного розміру, освітленості екрану мобільного пристрою, відстані пристрою до маркеру, роздільною здатністю камери, зовнішнього освітлення, яскравості екрану та якості паперу, на якому надруковано маркер. Розроблені прототипи програмного забезпечення профорієнтаційного квесту надають можливість визначити як координати учасника квесту, прив’язавши до них об’єкти доповненої реальності, так само як і до будь-яких зображень фотографічної якості у просторі профорієнтаційного квесту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A-Frame – Make WebVR [Electronic resource]. – [2020?]. – Access mode : https://aframe.io/
2. Applications | Jump Simulation [Electronic resource] / OSF HealthCare. – 2019. – Access mode : http://web.archive.org/web/20191130085337mp_/https://jumpsimulation.org/education/applications
3. AR-js-org/AR.js: Image tracking, Location Based AR, Marker tracking. All on the Web [Electronic resource]. – 2020. – Access mode : https://github.com/AR-js-org/AR.js
4. artoolkitx/jsartoolkit5: Javascript ARToolKit v5.x [Electronic resource]. – 2020. – Access mode : https://github.com/artoolkitx/jsartoolkit5
5. artoolkitx/jsartoolkit5: Javascript ARToolKit v5.x [Electronic resource] / GitHub. – 2020. – Access mode : https://github.com/artoolkitx/jsartoolkit5
6. Babylon AR Preview CDN [Electronic resource]. – [2019?]. – Access mode : https://ar.babylonjs.com/
7. Babylon.js: Powerful, Beautiful, Simple, Open - Web-Based 3D At Its Best [Electronic resource]. – [2020?]. – Access mode : https://www.babylonjs.com/
8. Creating good markers · Carnaux/NFT-Marker-Creator Wiki [Electronic resource]. – 2020. – Access mode : https://github.com/Carnaux/NFT-Marker-Creator/wiki/Creating-good-markers
9. Crockford on JavaScript [Electronic resource]. – 2011. – Access mode : https://youtu.be/playlist?list=PLEzQf147-uEpvTa1bHDNlxUL2klHUMHJu
10. Dev.Opera – webkit [Electronic resource] / Opera Software AS. – 2019. – Access mode : https://dev.opera.com/tags/webkit/
11. Glass – Glass [Electronic resource] / [Google]. – [2020]. – Access mode : https://www.google.com/glass/start/
12. jeromeetienne/threex: Game Extensions for three.js [Electronic resource] / [Jerome Etienne]. – Apr 15, 2017. – Access mode :

https://github.com/jeromeetienne/threex

13. Microsoft HoloLens | Mixed Reality Technology for Business [Electronic resource] / Microsoft. – 2020. – Access mode : https://www.microsoft.com/en-us/hololens
14. Mintii I. S. Augmented Reality: Ukrainian Present Business and Future Education [Electronic resource] / Iryna S. Mintii, Vladimir N. Soloviev // Augmented Reality in Education : Proceedings of the 1st International Workshop (AREdu 2018). Kryvyi Rih, Ukraine, October 2, 2018 / Edited by : Arnold E. Kiv, Vladimir N. Soloviev. – P. 227-231. – (CEUR Workshop Proceedings (CEUR-WS.org), Vol. 2257). – Access mode : http://ceur-ws.org/Vol-2257/paper22.pdf
15. Mobile & Tablet Browser Market Share Ukraine | StatCounter Global Stats [Electronic resource] / StatCounter. – 2020. – Access mode : https://gs.statcounter.com/browser-market-share/mobile-tablet/ukraine/
16. Mobile Android Version Market Share Ukraine | StatCounter Global Stats [Electronic resource] / StatCounter. – 2020. – Access mode : https://gs.statcounter.com/android-version-market-share/mobile/ukraine
17. Mobile Games - Ukraine | Statista Market Forecast [Electronic resource] / Statista. – 2020. – Access mode : https://www.statista.com/outlook/211/338/mobile-games/ukraine
18. Mobile Operating System Market Share Ukraine | StatCounter Global Stats [Electronic resource] / StatCounter. – 2020. – Access mode : https://gs.statcounter.com/os-market-share/mobile/ukraine
19. Murray J. Babylon AR [Electronic resource] / Justin Murray, Babylon.js. – Aug 1, 2019. – Access mode : https://medium.com/@babylonjs/babylon-ar-7823ab4a80c1
20. Propp V. Morphology of the Folktale [Electronic resource] / V. Propp. – 2nd ed. – Austin : University of Texas Press, 1968. – Access mode : https://monoskop.org/images/f/f3/Propp_Vladimir_Morphology_of_the_Folktale_2nd_ed.pdf

21. Stemkoski L. Basic Cube [Electronic resource] / Lee Stemkoski. – May 31, 2018. – Access mode : https://stemkoski.github.io/AR-Examples/hello-cube.html
22. Syrovatskyi O. V. Augmented reality software design for educational purposes / Oleksandr V. Syrovatskyi, Serhiy O. Semerikov, Yevhenii O. Modlo, Yuliia V. Yechkalo, Snizhana O. Zelinska // Computer Science & Software Engineering : Proceedings of the 1st Student Workshop (CS&SE@SW 2018), Kryvyi Rih, Ukraine, November 30, 2018 / Edited by : Arnold E. Kiv, Serhiy O. Semerikov, Vladimir N. Soloviev, Andrii M. Striuk. – P. 193-225. – (CEUR Workshop Proceedings (CEUR-WS.org), Vol. 2292). – Access mode : http://ceur-ws.org/Vol-2292/paper20.pdf
23. three.js – JavaScript 3D library [Electronic resource]. – [2020?]. – Access mode : https://threejs.org/
24. THREEx Game Extensions for Three.js [Electronic resource] / [Jerome Etienne]. – Apr 15, 2017. – Access mode : https://www.threejsgames.com/extensions/
25. Ukraine: popularity of app categories by age 2018 | Statista [Electronic resource] / Statista. – 2020. – Access mode : https://www.statista.com/statistics/1023304/ukraine-popularity-app-categories/
26. WebAssembly [Electronic resource]. – [2020?]. – Access mode : https://webassembly.org/
27. WebGL - Sample Application - Tutorialspoint [Electronic resource] / Tutorialspoint. – 2020. – Access mode : https://www.tutorialspoint.com/webgl/webgl_sample_application.htm
28. WebGL Overview - The Khronos Group Inc [Electronic resource] / The Khronos Group Inc. – 2020. – Access mode : https://www.khronos.org/webgl/
29. WebGL: 2D and 3D graphics for the web – Web APIs | MDN [Electronic resource] / Mozilla and MDN contributors. – 2020. – Access mode : https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API
30. WebRTC API – Web APIs | MDN [Electronic resource] / Mozilla and MDN contributors. – 2020. – Access mode : https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API

31. WebXR Device API – Web APIs | MDN [Electronic resource] / Mozilla and MDN contributors. – 2020. – Access mode : https://developer.mozilla.org/en-US/docs/Web/API/WebXR_Device_API
32. Заклади вищої освіти [Електронний ресурс] / Держстат України. – 19.03.2020. – Режим доступу : http://www.ukrstat.gov.ua/operativ/operativ2005/osv_rik/osv_u/vuz_u.html
33. Розподіл постійного населення України за статтю та віком на 1 січня 2019 року : статистичний збірник [Електронний ресурс] / Державна служба статистики України ; відповідальний за випуск М. Б. Тімоніна. – Київ, 2019. – 345 с. – Режим доступу : http://database.ukrcensus.gov.ua/PXWEB2007/ukr/publ_new1/2020/zb_chuselnist%202019.pdf
34. Что такое КВЕСТ? [Електронний ресурс] / AutoQuest ® Group. – 2017. – Режим доступу : https://web.archive.org/web/20081013022413/http://www.kbect.net.ua/gamma_igr/chto_takoe_kvest

ДОДАТКИ

Додаток А Фрагмент коду профорієнтаційного квесту

```
<script
src="https://cdn.jsdelivr.net/gh/aframevr/aframe@1c2407b26c61958ba
a93967b5412487cd94b290b/dist/aframe-master.min.js"></script>
<script src="https://raw.githack.com/AR-js-
org/AR.js/master/aframe/build/aframe-ar-nft.js"></script>

<style>
  .arjs-loader {
    height: 100%;
    width: 100%;
    position: absolute;
    top: 0;
    left: 0;
    background-color: rgba(0, 0, 0, 0.8);
    z-index: 9999;
    display: flex;
    justify-content: center;
    align-items: center;
  }

  .arjs-loader div {
    text-align: center;
    font-size: 1.25em;
    color: white;
  }
</style>

<body style="margin : 0px; overflow: hidden;">

<script>
     AFRAME.registerComponent('registerevents', {
```

```
          init: function () {
               var marker = this.el;

               marker.addEventListener('markerFound', function() {
                    console.log('markerFound', marker.id);
               });

               marker.addEventListener('markerLost', function() {
                    console.log('markerLost', marker.id);
               });
          }
     });

  AFRAME.registerComponent('navigate-on-click', {
    schema: {
      url: {default: ''}
    },

    init: function () {
      var data = this.data;
      var el = this.el;

      el.addEventListener('click', function () {
     console.log('navigate-on-click');
     window.open(data.url, '_blank' );
      });
    }
  });

     AFRAME.registerComponent('jump', {
          init: function () {
               var obj = this.el;
               obj.addEventListener('click', function () {
                         this.setAttribute('position',
150+100*(Math.random()-0.5)+ " 150 0");
     console.log('jump');
```

```
                    });
          }
     });
</script>

  <!-- minimal loader shown until image descriptors are loaded -->
  <div class="arjs-loader">
    <div>Loading, please wait...</div>
  </div>
  <a-scene vr-mode-ui="enabled: false;"
    renderer="logarithmicDepthBuffer: true;" embedded
    arjs="trackingMethod: best; sourceType: webcam;debugUIEnabled:
false;">
    <!-- url should contain 3 files: .fset, .fset3 and .iset -->

 </a-nft>
  <a-nft registerevents
      type="nft"

url="https://playground2.ccjournals.eu/student99/nft/KAFIN/kafera"
      smooth="true"
      smoothCount="10"
      smoothTolerance=".01"
      smoothThreshold="5">
<a-entity geometry="primitive: box; width: 150; height: 150;
depth: 1" material="src: url(kaferain.png)"
 position= "150 -100 0" scale = "4 4 2"     rotation="-90 0 0"
navigate-on-click="url:https://kdpu.edu.ua/informatyky-ta-
prykladnoi-matematyky/zahalna-informatsiia/vykladachi-
kafedry"></a-entity>

</a-nft>

  <a-entity camera><a-cursor></a-cursor></a-entity>
  </a-scene>
</body>
```