МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

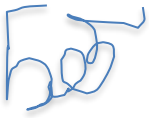
«До захисту допущено»
В.о. завідувача кафедри
Ігор
_____ ШЕЛЕХОВ
(підпис)
червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Вебдодаток для симуляції гри на селесті»
здобувача групи ІН – 02, Бражніка Дмитра Костянтиновича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.



Дмитро БРАЖНІК

Керівник, старший викладач,
кандидат технічних наук Артем КОРОБОВ

СУМИ – 2024

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук

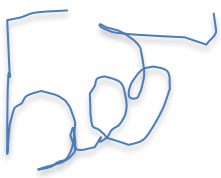
«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття освітнього ступеня бакалавра
зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН- 02, Бражніка Дмитра Костянтиновича

1. Тема роботи: « Вебдодаток для симуляції гри на селесті »
затверджую наказом по СумДУ від _«01» травня 2024 р. № 0475-VI___
2. Термін здачі здобувачем кваліфікаційної роботи до 28 травня 2024 року.
3. Вхідні дані до кваліфікаційної роботи _
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Дослідити існуючі технології, які можна використати для реалізації даного вебдодатку, постановка й формування завдань дослідження.
2) Визначити оптимальний підхід для створення вебдодатку, який зможе симулювати основні аспекти ігрового процесу.
3) Розробка вебдодатку використовуючи вибрані технології та дотримуючись завдань.
4) Аналіз отриманих результатів.
5) Оформлення пояснювальної записки до кваліфікаційної роботи.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

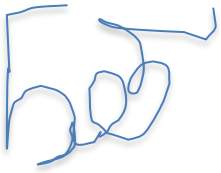
Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання  Керівник 

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	Дослідити існуючі технології, які можна використати для реалізації даного вебдодатку, постановка й формування завдань дослідження	06.05-10.05.2024	Виконано
2	Визначити оптимальний підхід для створення вебдодатку, який зможе симулювати основні аспекти ігрового процесу	10.05-13.05.2024	Виконано
3	Розробка вебдодатку використовуючи вибрані технології та дотримуючись завдань	13.05-17.05.2024	Виконано
4	Аналіз отриманих результатів	17.05-20.05.2024	Виконано
5	Оформлення пояснювальної записки до кваліфікаційної роботи	20.05-28.05.2024	Виконано

Здобувач вищої освіти  Керівник 

АНОТАЦІЯ

Записка: 54 стр., 12 рис., 1 додаток, 15 використаних джерел.

Обґрунтування актуальності теми роботи – селеста це музичний інструмент, що востаннє з'явився в обіг у кінці XIX століття, і відроджується у сучасній музичній практиці. Актуальність створення симулятора для гри на селесті полягає у популяризації цього інструменту серед музикантів, дослідників музичної технології та широкого загалу. Важливість збереження та просування культурної спадщини стає особливо актуальною в епоху швидких технологічних змін. Створення симулятора для гри на селесті сприяє залученню молодого покоління до вивчення та використання унікальних музичних інструментів, що відкриває нові можливості для музичної творчості та освіти.

Об’єкт дослідження - об'єктом дослідження є розробка віртуального симулятора для гри на селесті з використанням передових технологій у сфері комп'ютерної графіки та обробки звуку. Дослідження включає аналіз структури селести, реалізацію інтерактивного інтерфейсу та оптимізацію звукової моделі для максимальної реалістичності віртуального інструменту.

Мета роботи — основною метою роботи є створення інноваційного симулятора, який дозволить користувачам не лише віртуально грати на селесті, а й досліджувати його звукові можливості та інтерактивний характер. Проект спрямований на розвиток музичних навичок та творчого мислення через віртуальний досвід музикознавства.

Методи дослідження — в роботі використано методи комп'ютерного моделювання, програмування на мовах високого рівня (наприклад, JavaScript), технології візуалізації у реальному часі та цифрової обробки звуку. Для досягнення поставлених цілей було застосовано алгоритми генерації звуків, взаємодію з користувачем та оптимізацію роботи додатку.

Результати — результатом роботи є функціональний симулятор для гри на селесті з інтерактивним інтерфейсом, відтворенням реалістичних звуків та можливістю навчання та творчості в області музики. Проект показує потенціал використання віртуальних інструментів у навчальних та творчих цілях, що розширює можливості доступу до музичної освіти та виконавства.

ІНФОРМАЦІЙНА СИСТЕМА, СИМУЛЯТОР, СЕЛЕСТА, JAVA SCRIPT,HTML, CSS, PYTHON

ЗМІСТ

Вступ

Веб-додаток “Celesta” - це революційна онлайн-платформа, яка привносить чарівний звук музичного інструменту селеста прямо на ваші пальці. Цей унікальний веб-додаток дозволяє користувачам грати на селесті віртуально, реплікувати чарівний звук цього рідкісного і вишуканого інструменту.

Селеста, відома своїм дзвінким, ефірним звуком, є надзвичайним та захоплюючим музичним інструментом. Однак, незважаючи на її захоплюючий звук, існує відчутність відсутності інтерактивних онлайн-платформ, які дозволяють користувачам досліджувати та грати на селесті віртуально. Щоб заповнити цю прогалину, розпочато проект розробки веб-додатка “Celesta”, який надасть користувачам неперевершений музичний досвід.

В додатку “Celesta” внесено значні зміни, доданий функціонал зміни фону, щоб створити атмосферу, яка відповідає магічному звуку селеста. Також, додано можливість запису мелодій, які користувачі грають на селесті, щоб вони могли зберігати та відтворювати свої творіння у будь-який час.

Завдяки цим інноваціям, “Celesta” стає не просто додатком для гри на музичному інструменті, але і платформою для творчості, де кожен може відчути справжню магію музики.

Обґрунтування вибору теми роботи: обрана тема дослідження, створення симулятора для гри на селесті, вибрана з огляду на її великий потенціал у сфері музичної освіти та розвитку творчих навичок. Селеста, як маловідомий музичний інструмент, має багато особливостей, які можна досліджувати та впроваджувати у віртуальному середовищі для залучення користувачів до музичного навчання та творчості.

Актуальність: актуальність теми полягає в розвитку інтерактивних методів навчання та сприянні доступності музичної освіти для широкого кола користувачів. Використання сучасних технологій у створенні віртуального симулятора дозволить підвищити інтерес до вивчення музики та розвинути творчі здібності.

Об’єкт дослідження: процес створення та використання віртуального симулятора для гри на селесті є об’єктом дослідження. Досліджуються технології візуалізації, генерації звуків та інтерактивного взаємодії з користувачем у контексті музичної освіти та творчості.

Предмет дослідження: предметом дослідження є вплив використання віртуального симулятора для гри на селесті на процес навчання музики та розвитку музичних навичок у користувачів, а також можливості його використання у різних сферах освіти та розваг.

Гіпотеза: гіпотеза роботи полягає в тому, що використання віртуального симулятора для гри на селесті сприятиме підвищенню інтересу до вивчення музики, розвитку музичних навичок та творчого мислення у користувачів, що знайде відображення в покращенні якості музичної освіти.

Новизна: новизна роботи полягає в поєднанні передових технологій комп'ютерної графіки та звуку з вивченням та популяризацією маловідомого музичного інструменту через віртуальне середовище, що створює нові можливості для музичної освіти та творчості.

С**труктура**: дане робота складається зі вступу, аналітичного огляду, постановки задачі, вибір методу розв’язання поставленої задачі, опису

програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

1 Огляд Проекту

1.1 Про проект

Веб-додаток "Celesta" розроблений для того, щоб користувачі могли грати на селесті за допомогою клавіатури свого комп'ютера. Додаток пропонує широкий набір клавіш, ретельно розроблених для реплікації справжнього музичного інструменту селеста. Користувачі можуть насолоджуватися різноманіттям музичних нот, білих і чорних клавіш, створюючи гармонії і мелодії, які відзвучують у їхніх серцях. Цей додаток перетворює вашу клавіатуру в справжній музичний інструмент. Ви маєте можливість грати на селесті, використовуючи ретельно відтворені клавіші, які дозволяють створювати різні мелодії та гармонії. Кожна клавіша відтворює чарівний звук селести, даруючи вам неперевершений музичний досвід**.** Веб-додаток розроблено з метою заповнення прогалини у доступі до музичних інструментів онлайн. Celesta пропонує вам унікальний і захоплюючий досвід, який робить гру на селесті доступною для всіх, незалежно від вашого музичного досвіду. Дозвольте собі зануритися у світ чарівної музики разом з Celesta!

1.2 Основні Особливості

а) Реалістична 3D Візуалізація: Celesta надає вам можливість насолоджуватися візуальною привабливістю справжньої селести завдяки вражаючій 3D-моделі. Кожен елемент, від клавіш до корпусу інструменту, відтворено з неймовірною реалістичністю, що робить ваш досвід гри на селесті ще більш захоплюючим.

б) Висока якість звуку: Завдяки вдосконаленій звуковій системі, користувачі отримують неперевершений музичний досвід, слухаючи звуки селести, які максимально наближені до справжнього інструменту, щоб забезпечити найвищу якість звуку селести, кожна нота, кожен акорд звучить так, ніби ви граєте на справжньому інструменті. Інтенсивність, яскравість і чистота

звуку дарують вам можливість насолоджуватися музикою в її найпрекрасніших виразах..

в) Інтерактивний інтерфейс: Користувачам пропонується зручний інтерфейс з інтерактивними контрольними елементами, що дозволяють налаштувати звук, вибрати акорди та ритми для безлічі музичних композицій, створюючи унікальні музичні композиції. Інтерфейс дозволяє вам вільно виразити свою творчість та взаємодіяти з музикою за вашим бажанням.

1.3 Постановка задачі

Метою роботи є створення веб-додатка, який надаватиме користувачам можливість зануритися в світ музики, вдосконалювати свої музичні навички та брати участь у захоплюючих навчальних сесіях, пропонуючи інтерактивний і ефективний підхід до навчання музики.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

1. Розробити інтерфейс для взаємодії з користувачем під час гри на селесті.
2. Додати кнопку вибору фону, на інтерфейсі додатку з'явиться додаткова кнопка, яку можна натиснути для вибору фону з п'яти запропонованих варіантів. Це дозволить користувачам налаштовувати зовнішній вигляд додатку відповідно до їхніх вподобань та настрою.
3. Додати кнопку Старт/Стоп запису мелодії, з'явиться ще одна кнопка, яка дозволить користувачам починати та припиняти запис мелодій, які вони грають на селесті. Це важлива функція для тих, хто бажає зберігати свої музичні твори та вивчати їх пізніше.
4. Додати список збережених мелодій, додаток отримає також новий функціонал - список, де будуть зберігатися всі записані мелодії користувача. В цьому списку будуть кнопки для програвання та видалення

мелодій, що надасть можливість легко управляти своїми музичними творами.

5. Забезпечити можливість відтворення музичних нот та мелодій за допомогою комп'ютерної клавіатури або миші.
6. Додати вікно в якому під час гри будуть виводитись ноти які грає користувач

2 Аналіз існуючих веб-додатків для музичного навчання

2.1. Переваги та недоліки існуючих рішень

Під час аналізу існуючих веб-додатків для музичного навчання було виявлено декілька переваг та недоліків. Деякі з переваг таких додатків включають доступність з будь-якого пристрою з підключенням до Інтернету, інтерактивний інтерфейс, можливість навчатися у власному темпі. Однак, існуючі додатки мають деякі недоліки, такі як обмежений функціонал, недостатній рівень взаємодії з користувачем, а також обмеження у виборі музичних інструментів для навчання.

Переваги:

- Доступність: Можливість використання з будь-якого пристрою з Інтернетом, що робить навчання доступним і зручним.
- Інтерактивність: Інтерактивний інтерфейс, що сприяє залученню користувача та покращує засвоєння матеріалу.
- Темп навчання: Можливість навчатися у власному темпі, встановлюючи власний розклад та обсяг занять.

Недоліки:

- Обмежений функціонал: Деякі існуючі додатки мають обмежений функціонал, що може обмежувати можливості користувачів.
- Недостатній рівень взаємодії: Деякі додатки можуть мати недостатній рівень взаємодії з користувачем, що може впливати на зручність використання.
- Обмеження вибору інструментів: Можливі обмеження у виборі музичних інструментів для навчання, що не дозволяє користувачам розширювати свої навички в широкому спектрі музичних напрямків.

2.2. Відсутність аналогічного веб-додатка Celesta на ринку

- Під час аналізу популярності та пошукових запитів за допомогою Google Trends[3] було виявлено, що існує велика відсутність запитів про Celesta як музичний інструмент для навчання. Запити за рік не перевищували 100, що свідчить про низький рівень інтересу до цього напрямку. Дані пошуку веб-додатка також показали відсутність запитів на ринку.
- Це відкриває можливості для успішного впровадження Celesta та зайняття унікальної ніші на ринку музичних додатків для навчання. Відсутність аналогічного додатка у конкурентів також може допомогти залучити увагу користувачів та стати лідером у даній області.

Ці аспекти дозволяють побачити переваги та можливості Celesta на тлі існуючих рішень і показують потенціал для успішного впровадження та розвитку у цій сфері.

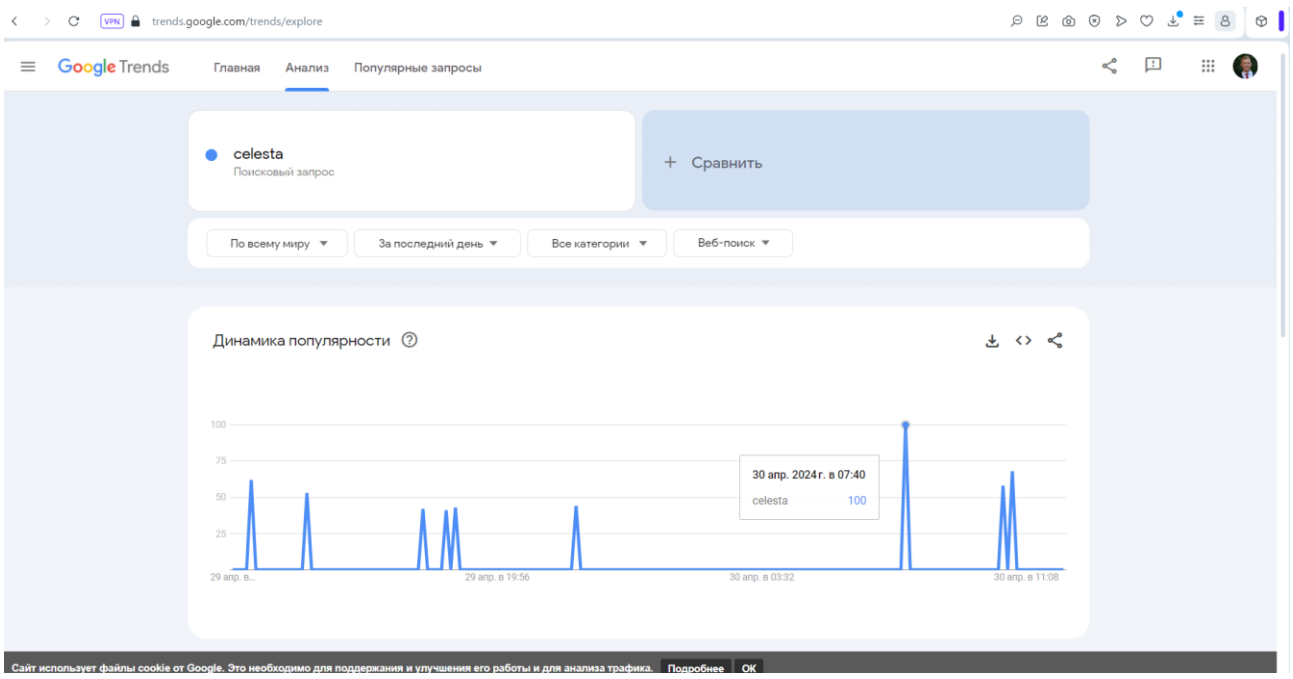


Рисунок 2.1 - запит Celesta

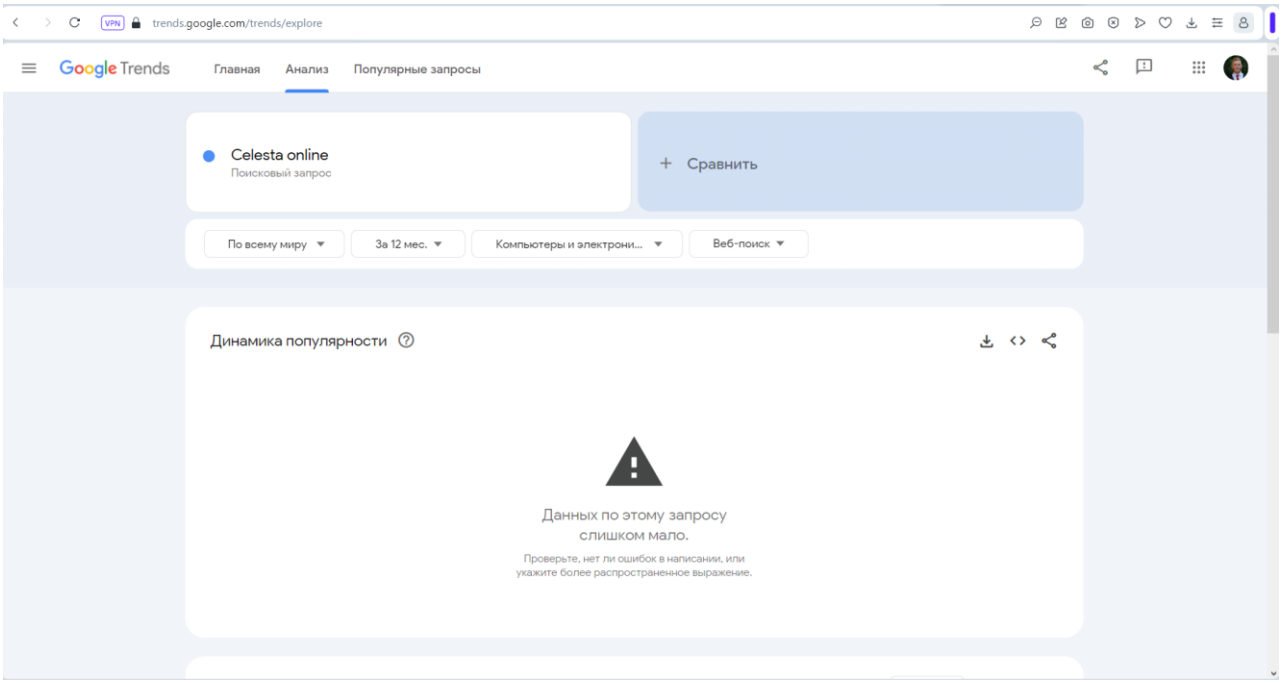


Рисунок 2.2 - запит Celesta online

3 Технічне завдання для розробки веб-додатка Celesta

3.1. Функціональні вимоги

- Розробити інтерфейс для взаємодії з користувачем під час гри на селесті.
- Додати кнопку вибору фону, на інтерфейсі додатку з'явиться додаткова кнопка, яку можна натиснути для вибору фону з п'яти запропонованих варіантів. Це дозволить користувачам налаштовувати зовнішній вигляд додатку відповідно до їхніх вподобань та настрою.
- Додати кнопку Старт/Стоп запису мелодії, з'явиться ще одна кнопка, яка дозволить користувачам починати та припиняти запис мелодій, які вони грають на селесті. Це важлива функція для тих, хто бажає зберігати свої музичні твори та вивчати їх пізніше.
- Додати список збережених мелодій, додаток отримає також новий функціонал - список, де будуть зберігатися всі записані мелодії користувача. В цьому списку будуть кнопки для програвання та видалення мелодій, що надасть можливість легко управляти своїми музичними творами.
- Додати вікно в якому під час гри будуть виводитись ноти які грає користувач
- Забезпечити можливість відтворення музичних нот та мелодій за допомогою комп'ютерної клавіатури або миші.

3.2. Нефункціональні вимоги

- Додаток повинен бути доступний з будь-якого пристрою з підключенням до Інтернету.
- Інтерфейс додатка повинен бути інтуїтивно зрозумілим та зручним для користувачів будь-якого рівня музичної підготовки.

- Додаток повинен працювати стабільно та без збоїв навіть при великому навантаженні.
- Доступність та зручність користування, додаток "Celesta" має залишатися доступним з будь-якого пристрою з підключенням до Інтернету, а його інтерфейс повинен бути інтуїтивно зрозумілим для користувачів будь-якого рівня музичної підготовки. Досягнення цієї цілі гарантує зручність та приємний досвід користування додатком незалежно від технічних знань користувача.
- Стабільність та продуктивність, додаток повинен працювати стабільно та без збоїв навіть при великому навантаженні. Це важливо для забезпечення комфортного користування та уникнення ризику втрати даних чи незручностей під час гри на селесті та роботи з записаними мелодіями.

3.3. Опис інтерфейсу користувача

Інтерфейс користувача повинен бути зрозумілим та привабливим. Основними елементами інтерфейсу будуть клавіатура для вибору нот, в майбутньому кнопки запису та відтворення композицій, індикатори прогресу навчання та досягнень.

Основними елементами інтерфейсу додатку будуть:

- Клавіатура для вибору нот та гри на селесті.
- Кнопка вибору фону для зміни зовнішнього вигляду додатку.
- Кнопка старт/стоп запису мелодії для збереження та відтворення творів користувача.
- Список збережених мелодій з кнопками програвання та видалення для зручного управління записами.
- Вивід нот в спеціально відведеному вікні, які грає користувач

Ці функції та елементи забезпечать користувачам комфортний, зручний та захоплюючий досвід гри на селесті та роботи з музичними творами у додатку "Celesta".

4 Вибір технологій для розробки веб-додатка

4.1. Вибір мови програмування

При розробці веб-додатка "Celesta" було обрано мову програмування JavaScript. Це стратегічне рішення, оскільки JavaScript є однією з найпопулярніших мов для веб-розробки з декількома важливими перевагами:

- Широке поширення: JavaScript є стандартом для веб-розробки та має велику спільноту розробників. Це забезпечує доступність багатьох ресурсів, бібліотек та фреймворків для покращення продуктивності розробки.
- Висока швидкість: JavaScript добре оптимізований для веб-додатків, що дозволяє досягати високої швидкості роботи та ефективності виконання завдань навіть у великих проектах.
- Можливості взаємодії з користувачем: JavaScript надає широкий функціонал для створення інтерактивних елементів, анімацій, взаємодії з користувачем та обробки подій, що дозволяє реалізувати багатофункціональний та захоплюючий інтерфейс веб-додатка.
- Підтримка браузерами: JavaScript є стандартом для веб-розробки, тому підтримується всіма сучасними браузерами, що забезпечує сумісність та доступність додатка для широкого кола користувачів.

Обрана мова програмування відповідає вимогам нашого проекту та дозволить забезпечити високу якість та продуктивність веб-додатка "Celesta".

4.2. Використання бібліотек та фреймворків

Для реалізації інтерфейсу та взаємодії з користувачем веб-додатка "Celesta" використовуються наступні бібліотеки та фреймворки:

- Babylon.js [1][8]: Бібліотека Babylon.js використовується для реалізації 3D візуалізації та інтерактивних елементів на стороні клієнта. Вона дозволяє створювати реалістичні 3D моделі та обробляти їхнє відображення у веб-середовищі.

- Babylon.GUI [5]: Для створення інтерактивного інтерфейсу та управління компонентами використовується бібліотека Babylon.GUI. Вона надає зручні засоби для створення кнопок, вікон, текстових полів та інших елементів інтерфейсу.
- Web Audio API [2]: Для роботи зі звуком у веб-додатку використовується Web Audio API. Ця бібліотека надає потужні інструменти для запису, відтворення та обробки звукових файлів безпосередньо у веб-середовищі, що дозволяє нам реалізувати функціональність запису та відтворення мелодій у додатку "Celesta".

Вибір цих мов програмування, бібліотек та фреймворків дозволяє нам ефективно реалізувати поставлені завдання та надати користувачам високоякісний та захоплюючий досвід користування нашим веб-додатком. Реалізація програми наведена у **Додаток А**.

4.3 Використання серверу Python

Python[15] - це високорівнева, інтерпретована, інтерактивна та об'єктно-орієнтована мова програмування, яка широко використовується для веб-розробки, наукових досліджень, штучного інтелекту та багатьох інших областей. Один із способів використання Python для створення веб-сервера на вашому ПК - це використання модуля http.server, який є частиною стандартної бібліотеки Python.

Загальна ідея сервера полягає в тому, що він приймає запити від клієнтів (наприклад, веб-браузерів) і надає їм відповіді, використовуючи різноманітні протоколи, такі як HTTP, FTP, SMTP і т.д. У веб-розробці сервер відповідає за надання вмісту веб-сторінок, обробку запитів користувачів і взаємодію з базою даних, якщо це необхідно.

Команда python -m http.server використовує вбудований HTTP-сервер Python, який має декілька параметрів налаштування, таких як порт, на якому

сервер працює, та директорія, з якої він обслуговує запити. Наприклад, команда python -m http.server 8080 запустить сервер на порту 8080, а команда python -m http.server --directory /path/to/directory вказує серверу обслуговувати файли з певної директорії.

Щодо вибору конкретного сервера, це залежить від конкретних потреб і вимог вашого проекту. Вбудований HTTP-сервер Python чудово підходить для швидкого тестування і локальної розробки, але у вас можуть бути інші вимоги, які потребують більш розширених функцій сервера, наприклад, використання спеціалізованих фреймворків веб-розробки, підтримки динамічного контенту, роботи з базою даних тощо. У таких випадках вибір сервера може бути здійснений на основі його можливостей, продуктивності, масштабованості та інших факторів, що важливі для вашого проекту.

Вибір сервера - це компроміс між простотою використання, потребами вашого проекту і ваших власних знань та навичок у роботі з різними серверними технологіями.

Команда python -m http.server запускає вбудований HTTP-сервер Python на вашому комп'ютері. Основні характеристики цього сервера включають:

Простота використання: Команда python -m http.server дозволяє легко запустити HTTP-сервер без необхідності встановлення додаткових пакетів чи налаштувань.

Реалізація простого сервера: Цей сервер надає простий спосіб відображення файлів у веб-браузері. Він обробляє HTTP-запити і відповідає з вмістом файлів, які знаходяться в поточній директорії або її піддиректоріях.

Локальний доступ: HTTP-сервер запускається на вашому ПК, тому ви можете відкривати веб-сторінки з вашого веб-браузера, використовуючи адресу http://localhost:8000.

Тепер про сервери загалом: сервер - це комп'ютер або програмне обладнання, яке надає послуги іншим комп'ютерам, які звертаються до нього через мережу. Основні причини вибору певного сервера можуть включати його

характеристики, продуктивність, надійність, підтримку необхідного програмного забезпечення тощо.

У нашому випадку, ми використовуєм вбудований HTTP-сервер Python через команду python -m http.server, через його простоту використання та можливість швидкого запуску локального сервера для розробки та тестування веб-застосунків або статичних веб-сайтів без необхідності встановлення окремого серверного ПО.

4.4 Babylon.js – як основний інстурмент

Babylon.js][11] - це потужна JavaScript бібліотека для створення тривимірних веб-додатків та ігор. Вона пропонує широкий набір інструментів і функціональностей для розробки вражаючих візуальних ефектів, інтерактивних сцен та анімацій. Давайте розглянемо детальніше основні компоненти та можливості цієї бібліотеки.

- Графічний движок та візуалізація:
- Babylon.js використовує потужний графічний движок, який дозволяє створювати складні тривимірні сцени з високою деталізацією.
- Підтримка широкого спектру візуальних ефектів, таких як реалістичні тіні, світлові ефекти, ефекти частинок та багато інших.
- Можливість роботи з текстурами, матеріалами та освітленням для створення реалістичних та привабливих візуальних ефектів.
- Взаємодія та управління:
- Можливість взаємодії з об'єктами на сцені, включаючи обробку кліків, перетягування, обертання та інші дії.
- Підтримка різних типів камер та управління їх положенням і орієнтацією для кращої перспективи та інтерактивності.
- Фізичне моделювання:
- Можливість створення фізично реалістичних сцен з колізіями, гравітацією та іншими фізичними ефектами.

- Реалізація анімації об'єктів за допомогою фізичних законів для створення реалістичних рухів та ефектів.
- Анімація та ключові кадри:
- Підтримка анімації об'єктів та камер з використанням ключових кадрів для створення складних та динамічних анімаційних сцен.
- Можливість реалізації різноманітних рухомих ефектів, змін положення та форми об'єктів, анімації текстур і матеріалів.
- Крос-платформеність та веб-сумісність:
- Babylon.js працює на різних платформах та пристроях, що дозволяє створювати веб-додатки та ігри для широкої аудиторії.
- Підтримка сучасних веб-браузерів з можливістю оптимізації та використання різних функцій HTML5 та WebGL для покращення продуктивності та швидкості відтворення.

Babylon.js - це не лише інструмент для створення вражаючих тривимірних сцен, але й потужний набір інструментів для реалізації ваших творчих ідей у веб-просторі. Його широкий функціонал, висока продуктивність та можливості для взаємодії з користувачем роблять його одним з найбільш популярних інструментів для тривимірної веб-розробки.

4.5 Web Audio API

Web Audio API - дозволяє відтворювати, створювати та керувати звуком, додавати звукові ефекти, створювати візуалізацію аудіо та багато іншого за допомогою JavaScript у браузері.

Об'єкт AudioContext - це основа Web Audio, який має методи для створення інших об'єктів для обробки аудіо.

Основні об'єкти які використовуються для робити Web Audio:

- AudioContext - аудіо контекст для обробки аудіо.
- AnalyserNode - об'єкт який аналізує частоти звуку.
- AudioListener - об'єкт який вказує положення слухача звуку.

- BiquadFilterNode - фільтр звуку низького порядку.
- ConvolverNode - об'єкт для створення ефекту реверберації звуку.
- GainNode - об'єкт для регулювання рівня гучності звукового сигналу.
- MediaElementAudioSourceNode - об'єкт для отримання звуку з медіа елемента.
- OscillatorNode - об'єкт для створення звукової хвилі.
- StereoPannerNode - стерео панорамування звуку.

Основні аудіо операції виконують аудіо вузли.

Не всі браузери підтримують Web Audio API.

Рекомендовано перевіряти чи підтримує браузер Web Audio API.

Робота зі звуком:

1. Завантаження звуку за допомогою AJAX

За допомогою технології AJAX і Web Audio можна завантажити аудіо файл, декодувати методом AudioContext.decodeAudioData() та відтворити.

2. Відтворення звуку з елемента Audio

Для відтворення аудіо з елемента Audio використовується аудіо вузол MediaElementAudioSourceNode.

3. Зміна гучності

Зміна рівня сигналу можна за допомогою аудіо вузла GainNode.

4. Змішування двох звуків

Змішування двох аудіо сигналів відбувається за допомогою відтворення з двох джерел.

5. Візуалізація

Візуалізація аудіо звуку.

Для того щоб створити візуалізацію звуку на полотні canvas використовується об'єкт AnalyserNode для аналізації звукового сигналу.

6. Фільтр звука низького порядку

Для фільтрації аудіо звуку через фільтр низького порядку використовується об'єкт BiquadFilterNode.

7. Створення звуку

Створення звуку відбувається за допомогою об'єкта OscillatorNode. [6] [7]

5 Архітектура веб-додатка Celesta

5.1. Структура базових компонентів

Веб-додаток "Celesta" буде побудований на наступних базових компонентах, які забезпечать його функціональність та ефективну роботу:

- Інтерфейс користувача: Основний інтерфейс, який надає зручний доступ до функцій додатка та взаємодію з користувачем.
- Система звуку: Відповідає за відтворення музичних нот та мелодій, забезпечуючи високу якість звуку із застосуванням Web Audio API[2][13][14].
- Система запису та відтворення композицій: Надає можливість користувачам записувати свої мелодії та відтворювати їх, створюючи власні музичні твори.
- Система відстеження прогресу користувачів: Забезпечує ведення статистики та відстеження прогресу користувачів у вивченні та використанні додатка.

5.2. Організація роботи зі звуком

Для реалізації звукового компоненту буде використана бібліотека Web Audio API[2][9][12]. Ця технологія дозволяє відтворювати реалістичний та якісний звук, що є важливим аспектом для нашого додатка, орієнтованого на музичні можливості.

5.3. Інтеграція зовнішніх сервісів

Додаток буде підтримувати інтеграцію зовнішніх сервісів для збереження та обробки даних користувачів. Планується розміщення веб-додатка на GitHub, а також створення GitHub Pages для зручного доступу користувачів. Рисунок 5.1 – GitHubдемонструє архітектурну діаграму веб-додатка Celesta та його взаємодію зовнішніми сервісами[4][10].

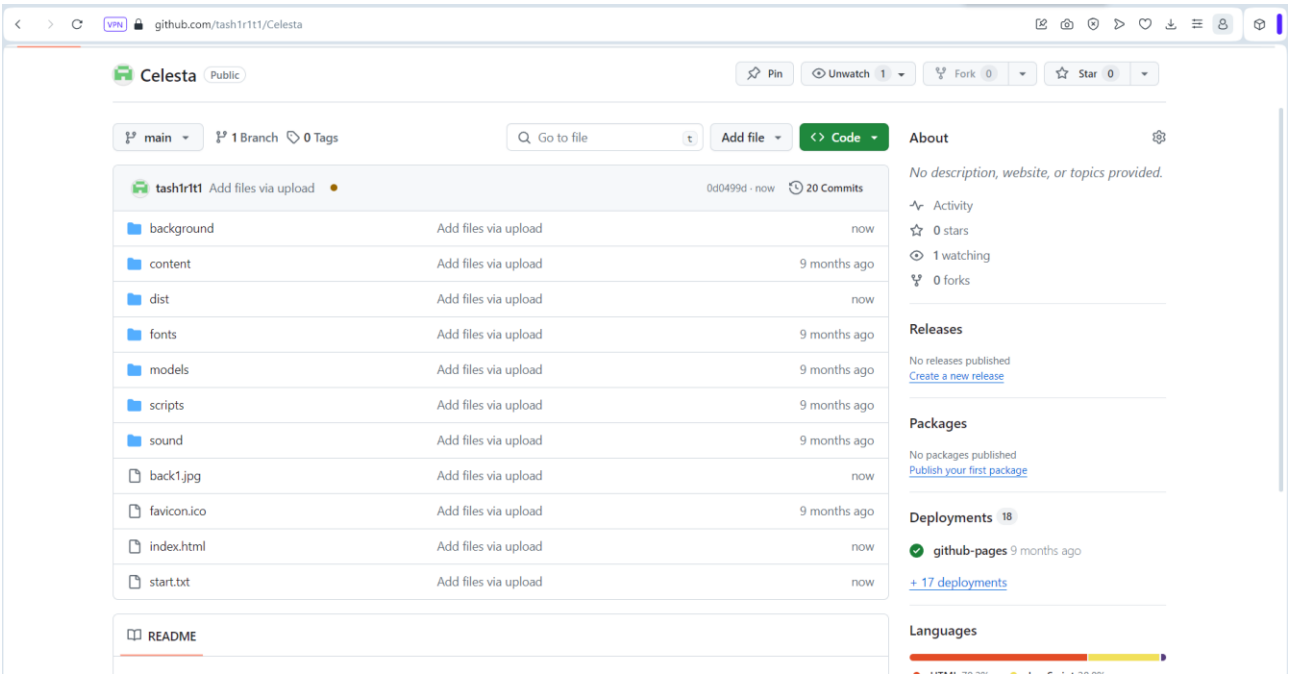


Рисунок 5.1 – GitHub

6 Розробка функціоналу веб-додатка

6.1. Реалізація клавіатурної нотації

Буде розроблений механізм клавіатурної нотації, який надасть користувачам можливість вибирати музичні ноти та мелодії за допомогою комп'ютерної клавіатури або комп’ютерної миші. При натисканні клавіш будуть відтворюватися відповідні звуки нот, що дозволить користувачам створювати свої власні музичні твори за допомогою зручного та інтуїтивного інтерфейсу.

6.2 Взаємодія з користувачем через інтерфейс

Створений інтерфейс дозволить користувачам здійснювати запис та відтворення створених композицій. Користувачі зможуть бачити свій прогрес навчання та досягнення. Також інтерфейс забезпечить можливість вибору фону для заднього плану. Інтерфейс можно переглянути на Рисунок 6.1 - Інтерфейс селести - 6.8.

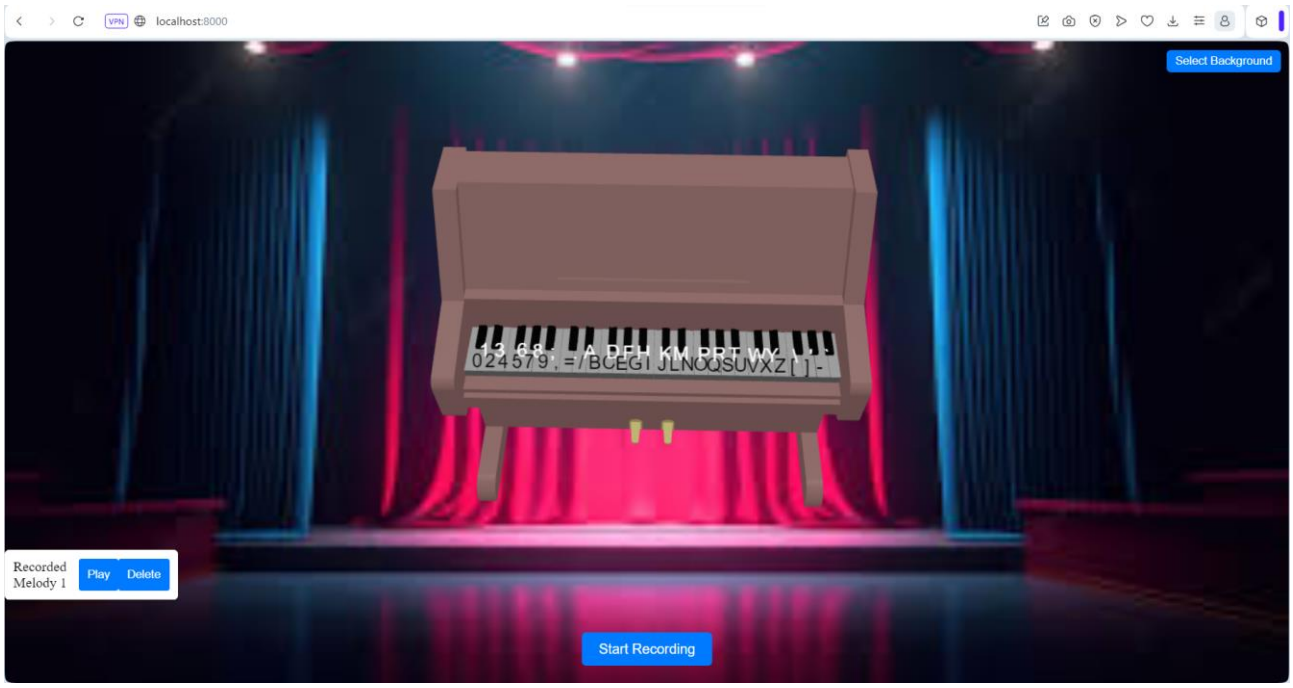


Рисунок 6.1 - Інтерфейс селести

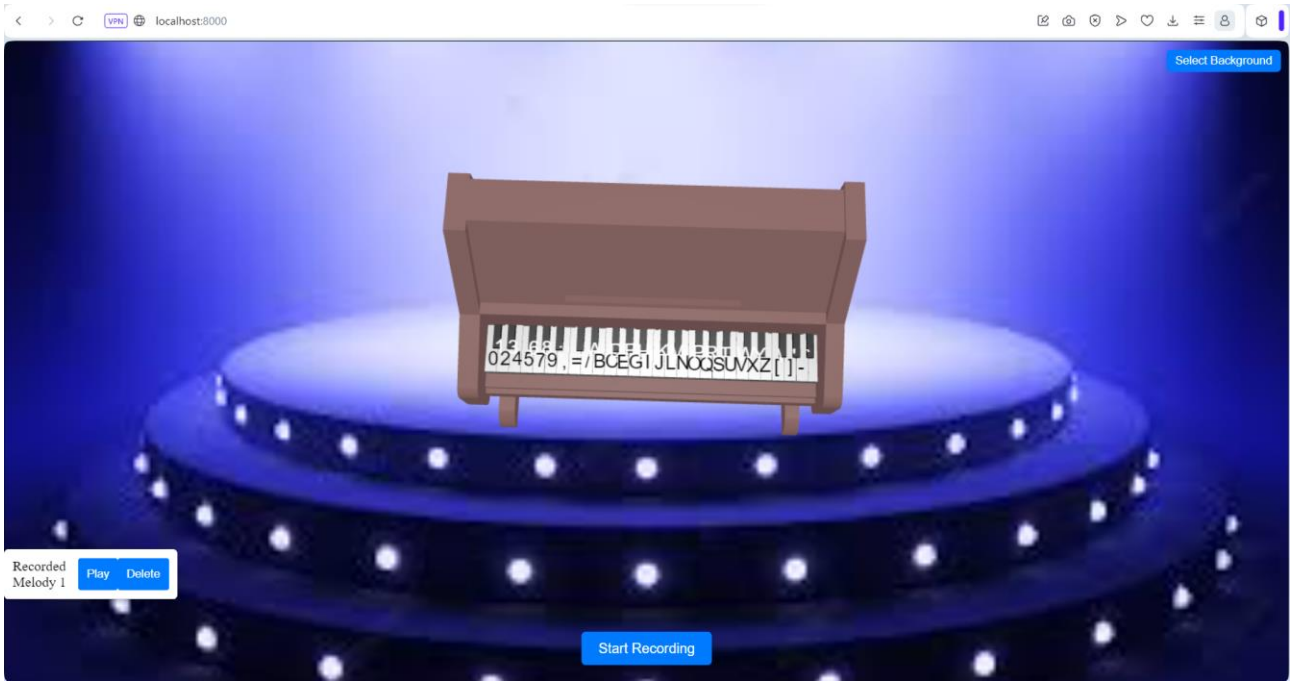


Рисунок 6.2 - Інтерфейс селести

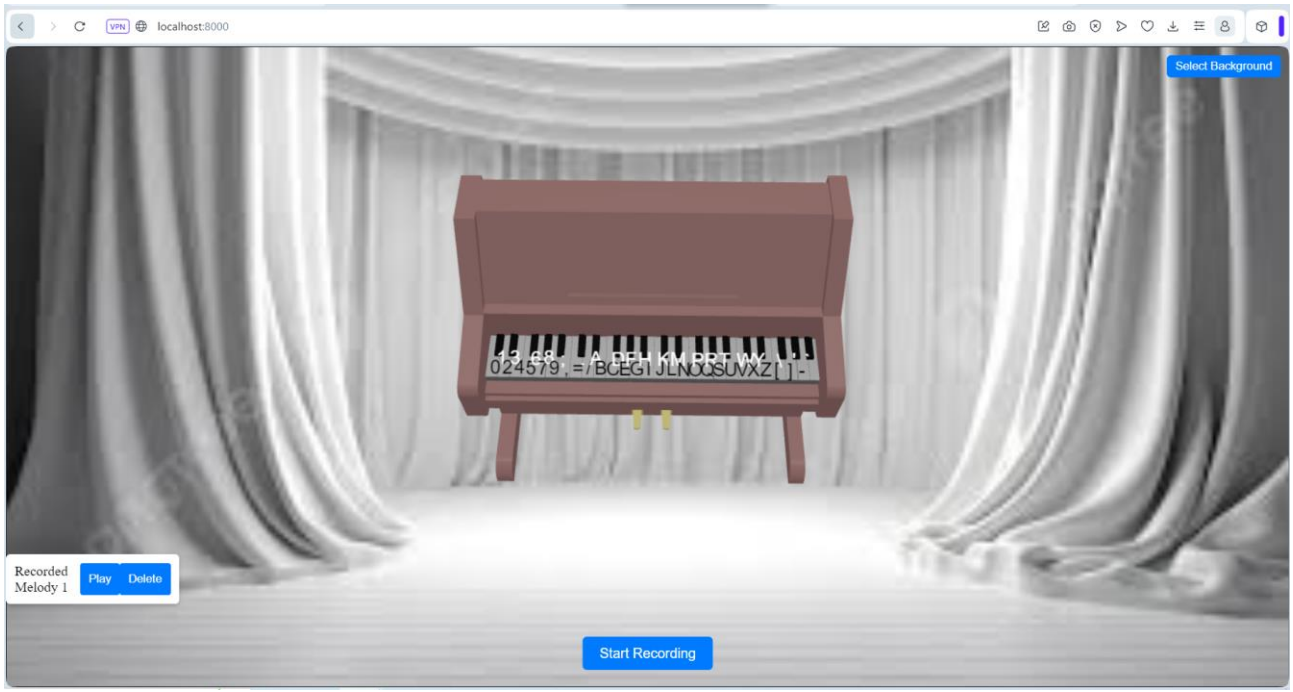


Рисунок 6.3 - Інтерфейс селести

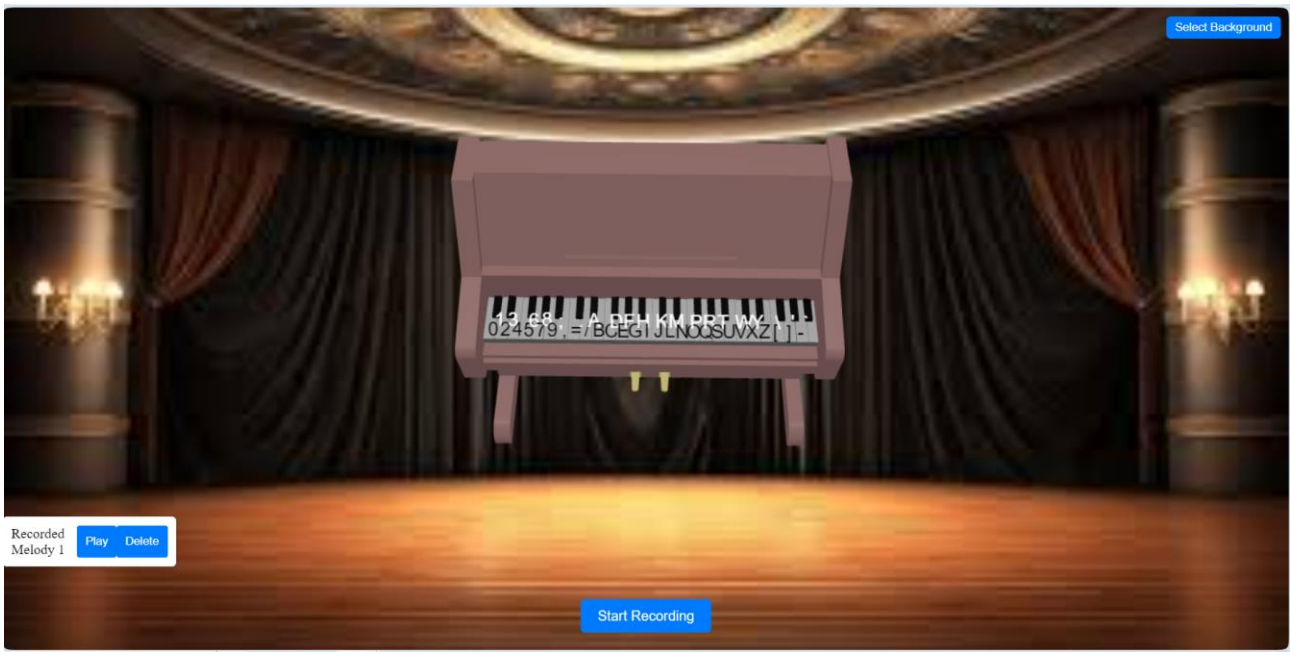


Рисунок 6.4 - Інтерфейс селести

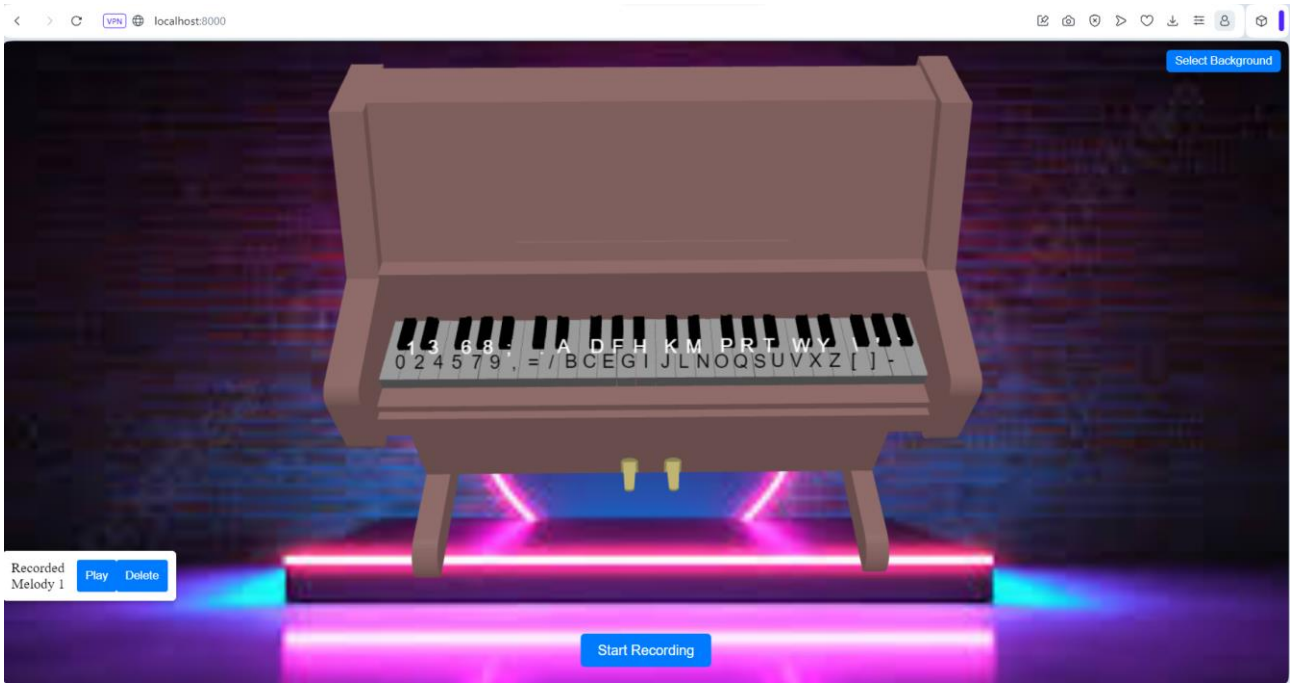


Рисунок 6.5 - Інтерфейс селести

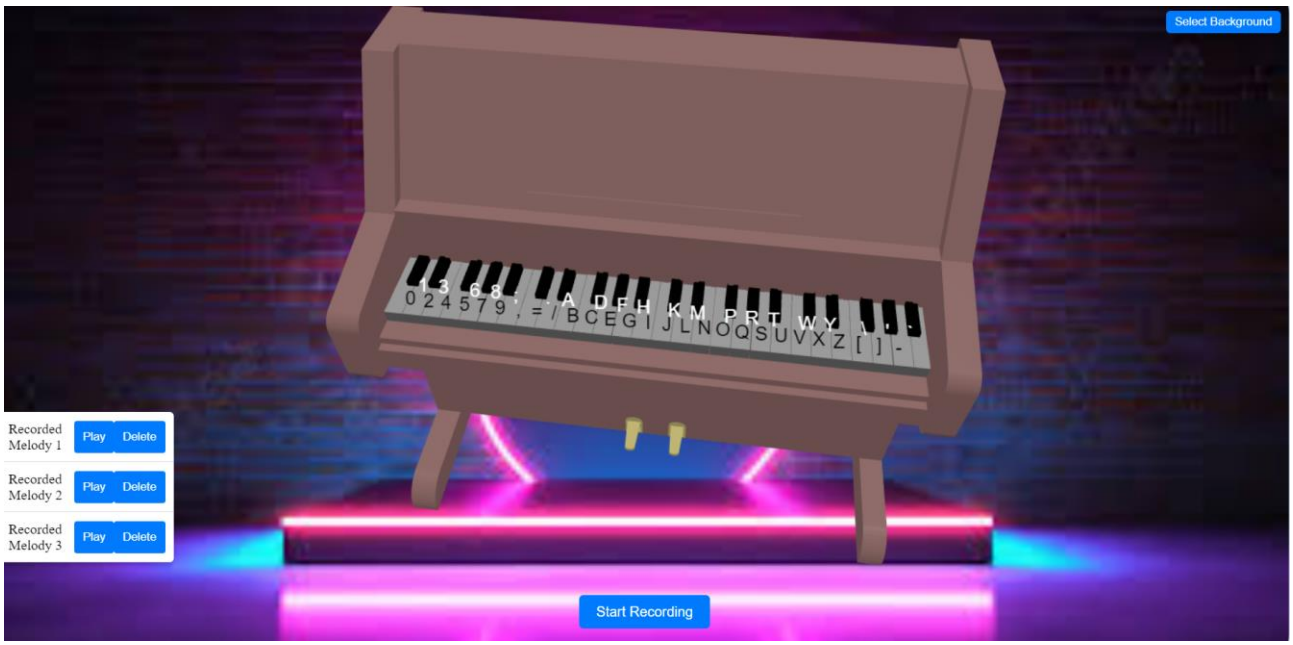


Рисунок 6.6 - Інтерфейс селести зі списком мелодій

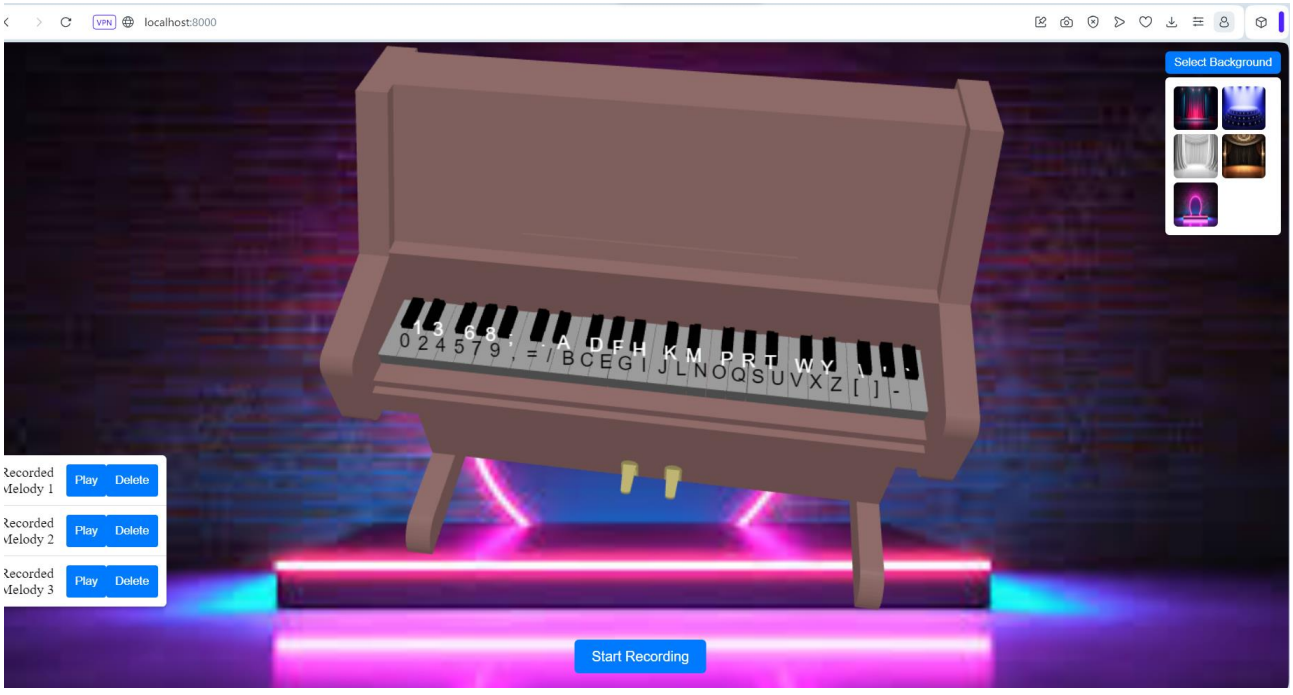


Рисунок 6.6 - Інтерфейс селести, вибір фону

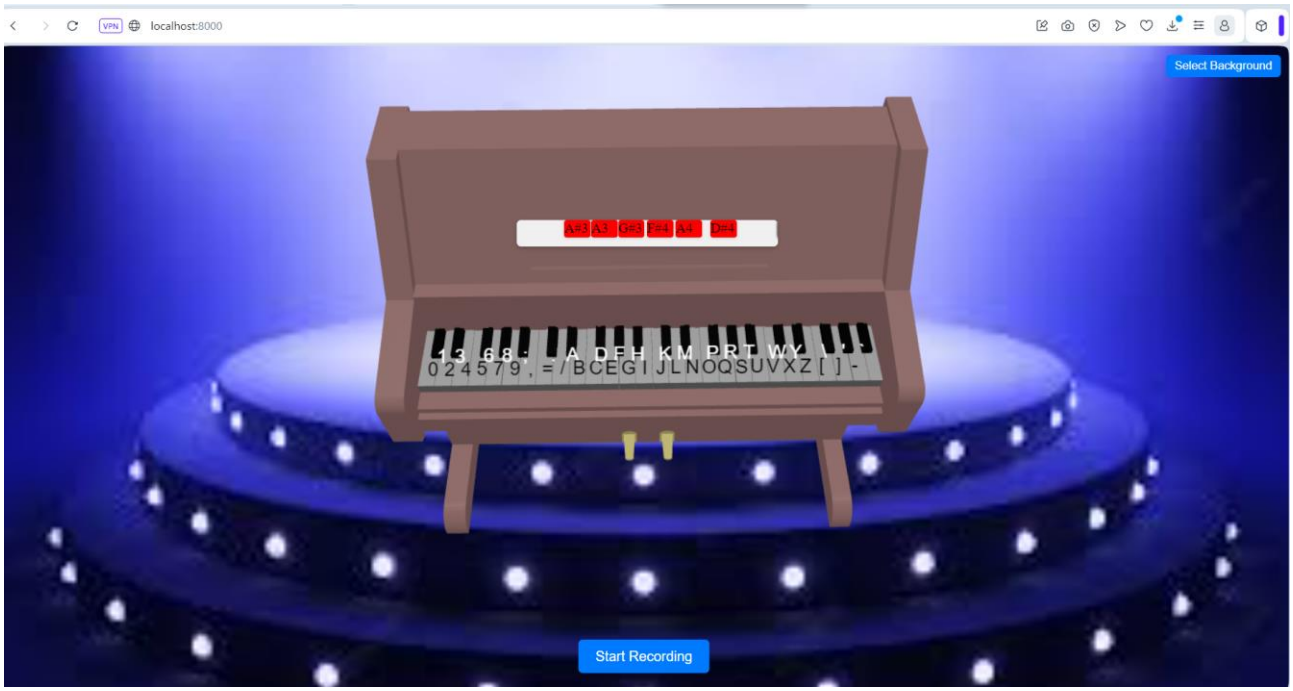


Рисунок 6.7 - Інтерфейс селести, вивід нот

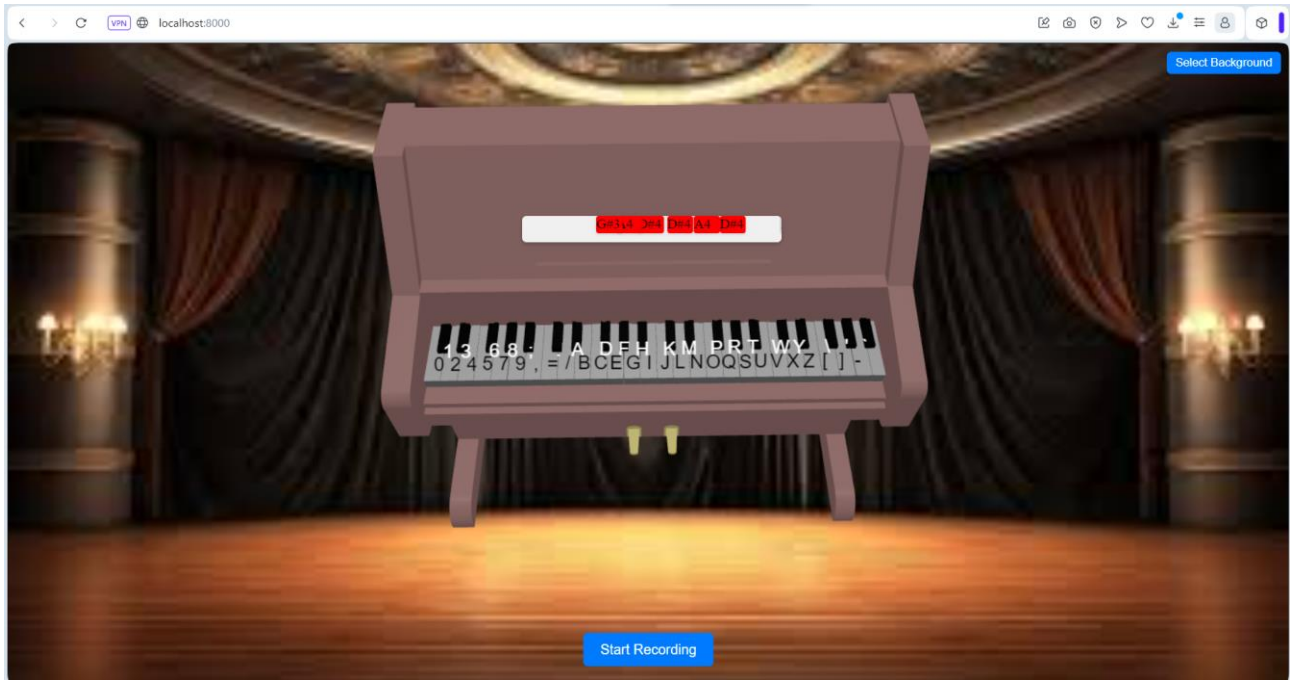


Рисунок 6.8 - Інтерфейс селести, вивід нот

Висновки

У данному проекті був розглянутий процес розробки веб-додатка "Celesta" - інноваційного та унікального музичного інструменту для онлайн музичного навчання. Метою проекту було створення веб-додатка, який надаватиме користувачам можливість зануритися в світ музики, вдосконалювати свої музичні навички та брати участь у захоплюючих навчальних сесіях, пропонуючи інтерактивний і ефективний підхід до навчання музики.

В ході реалізації проекту було вивчено існуючі веб-додатки для музичного навчання, їх переваги та недоліки, а також виявлено відсутність аналогічного веб-додатка "Celesta" на ринку, що зумовило актуальність та значущість цього проекту.

На етапі розробки технічного завдання були сформульовані функціональні та нефункціональні вимоги до додатка, а також детально описано інтерфейс користувача для зручності його використання.

При розробці веб-додатка були здійснені вибір мови програмування та використання бібліотек та фреймворків для забезпечення оптимальної продуктивності та функціональності. Архітектура веб-додатка була створена з урахуванням структури базових компонентів, організації роботи зі звуком та інтеграції зовнішніх сервісів.

В результаті реалізації дипломного проекту був успішно створений веб-додаток "Celesta", який надає користувачам унікальний досвід онлайн музичного навчання та стимулює їх музичний розвиток. Проект виконаний згідно поставлених цілей та завдань і є перспективним рішенням для всіх, хто прагне розвивати свої творчі здібності та навички в музиці.

У даному проекті було успішно реалізовано функціонал запису мелодій зі списку, заміна фону та вивід нот які програються в спеціально відведеному вікні що розширює можливості веб-додатка "Celesta". Реалізація цих функцій дозволяє користувачам більш гнучко керувати контентом, персоналізувати досвід

використання додатка та зберігати свої унікальні творіння для подальшого використання.

Ці можливості сприяють покращенню взаємодії користувачів з додатком та збільшують його привабливість для широкого кола аудиторії. Разом з іншими ключовими функціями, які вже реалізовані у веб-додатку "Celesta", ці можливості створюють унікальний та захоплюючий досвід для всіх, хто цікавиться музичним навчанням та творчістю.

СПИСОК ЛІТЕРАТУРИ

1. «babylonjs.com» [Електронний ресурс]: [Інтернет-портал]. Портал для вивчення документації по Babylon.js – Режим доступу: babylonjs.com

2. «developer.mozilla.org» [Електронний ресурс]: [Інтернет-портал]. Портал для вивчення документації по Web Audio API – Режим доступу: developer.mozilla.org

3. «trends.google.com» [Електронний ресурс]: [Інтернет-портал]. Портал для аналізу пошукових запитів – Режим доступу: trends.google.com

4. «github.com» [Електронний ресурс]: [Інтернет-портал]. Портал для розміщення веб-додатку – Режим доступу: github.com

5. «doc.babylonjs.com» [Електронний ресурс]: [Інтернет-портал]. Портал для вивчення документації по Babylon.GUI – Режим доступу: doc.babylonjs.com

6. «яваскрипт.укр» [Електронний ресурс]: [Інтернет-портал]. Портал для вивчення документації по Web Audio API – Режим доступу: яваскрипт.укр

7. «ux-republic.com» [Електронний ресурс]: [Інтернет-портал]. Портал для вивчення документації по Web Audio API – Режим доступу: ux-republic.com

8. «uk.softoware.org» [Електронний ресурс]: [Інтернет-портал]. Портал для вивчення документації по Babylon.js – Режим доступу: uk.softoware.org

9. «devzone.org.ua» [Електронний ресурс]: [Інтернет-портал]. Портал для вивчення документації по Web Audio API – Режим доступу: devzone.org.ua

10. «drukarnia.com.ua» [Електронний ресурс]: [Інтернет-портал]. Портал з інформацією щодо розміщення веб-додатку на GitHub – Режим доступу: drukarnia.com.ua

11. Moreau-Mathis Julien. Babylon.js Essentials , 2016. — 200 p. — ISBN: 978-1-78588-479-5.

12. Smus B. Web Audio API: Advanced Sound for Games and Interactive Apps, 2013. — 75 p

13. Turner William. JavaScript for Sound Artists Learn to Code with the Web Audio API, 2023. — 276 p. — ISBN: 978-1-003-20149-6.

14. Pfeiffer S., Green T. Beginning HTML5 Media: Make the most of the new video and audio standards for the Web, 2015. — 304 p. — ISBN: 1484204611, ISBN13: (electronic): 978-1-4842-0460-3.

15. Stack Overflow Contributors. Python Notes for Professionals, 2018. — 813 p.

Додаток А

```
<!DOCTYPE html>
<html>
<head>
   <title>CELESTA</title>
   <script src="https://cdn.babylonjs.com/babylon.js"></script>
   <script src="https://cdn.babylonjs.com/loaders/babylonjs.loaders.js"></script>
   <script src="https://cdn.babylonjs.com/loaders/babylonjs.loaders.min.js"></script>
   <script src="https://cdn.babylonjs.com/gui/babylon.gui.min.js"></script>
   <script
src="https://cdnjs.cloudflare.com/ajax/libs/howler/2.2.3/howler.min.js"></script>
   <style>
      body {
         margin: 0;
         padding: 0;
         overflow: hidden;
         background-size: cover;
         background-position: center;
         background-repeat: no-repeat;
         height: 100vh;

      }

      #renderCanvas {
         width: 100%;
         height: 100%;
         display: block;
      }

      #background-menu {
         position: fixed;
         top: 10px;
         right: 10px;
         z-index: 1000;
      }

      #background-select {
         display: none;
         position: absolute;
         top: 30px;
         left: 0;
         z-index: 1001;
         background-color: white;
```

```
    padding: 5px;
    border-radius: 5px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
}

.background-thumbnails {
    display: flex;
    flex-wrap: wrap;
    gap: 5px;
}

.background-thumbnail {
    width: 50px;
    height: 50px;
    cursor: pointer;
}

#select-background-btn {
    cursor: pointer;
    padding: 5px 10px;
    border: none;
    background-color: #007bff;
    color: white;
    border-radius: 5px;
}
#UI {
    position: fixed;
    top: 10px;
    left: 10px;
    z-index: 1000;
}

#canvasContainer {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
}

#record-btn {
    position: fixed;
    bottom: 20px;
    left: 50%;
```

```
    transform: translateX(-50%);
    cursor: pointer;
    padding: 10px 20px;
    border: none;
    background-color: #007bff;
    color: white;
    border-radius: 5px;
    font-size: 16px;
}

#melodies-list {
    position: fixed;
    bottom: 80px;
    left: 100px;
    transform: translateX(-50%);
    list-style-type: none;
    padding: 0;
    max-width: 200px;
    width: 30%;
    overflow-y: auto;
    background-color: white;
    border-radius: 5px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
}

.melody-item {
    padding: 10px;
    border-bottom: 1px solid #ddd;
    display: flex;
    justify-content: space-between;
}

.melody-item button {
    cursor: pointer;
    padding: 5px 10px;
    border: none;
    background-color: #007bff;
    color: white;
    border-radius: 3px;
}

#background-menu {
    position: fixed;
    top: 10px;
```

```
    right: 10px;
    z-index: 1000;
}

#background-select {
    display: none;
    position: absolute;
    top: 30px;
    right: 0;
    z-index: 1001;
    background-color: white;
    padding: 10px;
    border-radius: 5px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
}

.background-thumbnails {
    display: flex;
    flex-wrap: wrap;
    gap: 5px;
}

.background-thumbnail {
    width: 50px;
    height: 50px;
    cursor: pointer;
    border-radius: 5px;
    transition: transform 0.3s ease;
}

.background-thumbnail:hover {
    transform: scale(1.1);
}
 .pianoRoll {
    width: 300px; /* Примерная ширина полосы */
    height: 30px; /* Примерная высота полосы */
    background-color: #f0f0f0; /* Цвет фона полосы */
    position: absolute;
    top: 200px; /* Примерное расположение сверху */
    left: 595px; /* Примерное расположение слева */
    overflow: hidden; /* Скрытие содержимого, выходящего за пределы */
    border-radius: 5px; /* Скругление углов */
    box-shadow: 0px 2px 5px rgba(0, 0, 0, 0.2); /* Тень */
}
```

```
        .pianoNote {
            position: absolute;
            left: 0;
            top: 0;
            width: 30px; /* Примерная ширина ноты */
            height: 20px; /* Примерная высота ноты */
            background-color: #ff0000; /* Цвет ноты */
            animation: moveNote 5s linear forwards; /* Анимация движения ноты */
            border-radius: 3px; /* Скругление углов */
            box-shadow: 0px 1px 3px rgba(0, 0, 0, 0.2); /* Тень */
        }

        @keyframes moveNote {
            from {
                left: 0; /* Начальная позиция ноты */
            }
            to {
                left: 100%; /* Конечная позиция ноты (исчезание справа) */
            }
        }
    </style>

    <link rel="icon" type="image/png" href="models/logo_celesta.png"
sizes="16x16">
</head>
<body>
    <div id="UI"></div>
    <div id="canvasContainer">
        <canvas id="renderCanvas"></canvas>
    </div>
    <div id="pianoRoll" class="pianoRoll"></div>

    <button id="record-btn">Start Recording</button>
    <ul id="melodies-list"></ul>
    <div id="background-menu">
        <button id="select-background-btn">Select Background</button>
        <div id="background-select">
            <div class="background-thumbnails">
                <img src="background/back1.jpg" class="background-thumbnail" data-
src="background/back1.jpg">
                <img src="background/back2.jpg" class="background-thumbnail" data-
src="background/back2.jpg">
```

```
            <img src="background/back3.jpg" class="background-thumbnail" data-
src="background/back3.jpg">
            <img src="background/back4.jpg" class="background-thumbnail" data-
src="background/back4.jpg">
            <img src="background/back5.jpg" class="background-thumbnail" data-
src="background/back5.jpg">
        </div>
    </div>
</div>
<script type="text/javascript">
    // Конфігурація параметрів клавіш, включаючи тип (біла чи чорна), ноту,
розміри та позиції.
    const keyParams = [
        { type: "white", note: "C", topWidth: 1.4, bottomWidth: 2.3, topPositionX: -
0.45, wholePositionX: -14.4 },
        { type: "black", note: "C#", wholePositionX: -13.45 },
        { type: "white", note: "D", topWidth: 1.4, bottomWidth: 2.4, topPositionX: 0,
wholePositionX: -12 },
        { type: "black", note: "D#", wholePositionX: -10.6 },
        { type: "white", note: "E", topWidth: 1.4, bottomWidth: 2.3, topPositionX:
0.45, wholePositionX: -9.6 },
        { type: "white", note: "F", topWidth: 1.3, bottomWidth: 2.4, topPositionX: -
0.55, wholePositionX: -7.2 },
        { type: "black", note: "F#", wholePositionX: -6.35 },
        { type: "white", note: "G", topWidth: 1.3, bottomWidth: 2.3, topPositionX: -
0.2, wholePositionX: -4.8 },
        { type: "black", note: "G#", wholePositionX: -3.6 },
        { type: "white", note: "A", topWidth: 1.3, bottomWidth: 2.3, topPositionX:
0.2, wholePositionX: -2.4 },
        { type: "black", note: "A#", wholePositionX: -0.85 },
        { type: "white", note: "B", topWidth: 1.3, bottomWidth: 2.4, topPositionX:
0.55, wholePositionX: 0 },
    ]
    // Відповідність клавіш на клавіатурі їхнім музичним значенням.
    const keyMappings = {
        "1": "C#3",
        "2": "D3",
        "3": "D#3",
        "4": "E3",
        "5": "F3",
        "6": "F#3",
        "7": "G2",
        "8": "G#2",
        "9": "A2",
```

```
    "0": "C3",
    "-": "A5",
    "=": "C4",
    "`": "A#5",
    "q": "G4",
    "w": "C#6",
    "e": "G3",
    "r": "G#4",
    "t": "A#4",
    "y": "D#6",
    "u": "B4",
    "i": "B3",
    "o": "F5",
    "p": "F#5",
    "a": "D#4",
    "s": "A4",
    "d": "F#4",
    "f": "G#3",
    "g": "A3",
    "h": "A#3",
    "j": "C5",
    "k": "C#5",
    "l": "D5",
    ";": "A#2",
    "'": "G#5",
    "\\": "F#6",
    "z": "E6",
    "x": "D6",
    "c": "F4",
    "v": "C6",
    "b": "E4",
    "n": "E5",
    "m": "D#5",
    ",": "B2",
    ".": "C#4",
    "/": "D4",
    "[": "F6",
    "]": "G5",
    " ": "B5"

};
// Змінні та елементи DOM для роботи з сценою та записом мелодій.
let scene;
const canvas = document.getElementById("renderCanvas");
```

```
const engine = new BABYLON.Engine(canvas, true);

// Функція для створення сцени та камери з подіями для зміни фону та взаємодії з музичною клавіатурою.
const createScene = async function (engine) {

    scene = new BABYLON.Scene(engine);
    const alpha = 3 * Math.PI / 2;
    const beta = Math.PI / 50;
    const radius = 220;
    const target = new BABYLON.Vector3(0, 0, 0);
    const camera = new BABYLON.ArcRotateCamera("Camera", alpha, beta, radius, target, scene);
    camera.attachControl(canvas, true);

    // Загрузка фонового зображення та налаштування сцени.
    let backgroundTexture = new BABYLON.Texture("background/back1.jpg", scene);
    scene.clearColor = new BABYLON.Color4(0, 0, 0, 0);
    scene.backgroundTexture = backgroundTexture;

    // Обробники подій для зміни фонового зображення.
    document.addEventListener('DOMContentLoaded', function () {
      const thumbnails = document.querySelectorAll('.background-thumbnail');
      const backgroundSelect = document.getElementById('background-select');
      const canvasContainer = document.getElementById('canvasContainer');

      function setBackground(imageUrl) {
        if (scene.backgroundTexture) {
        scene.backgroundTexture.dispose();
      }
        document.body.style.backgroundImage = `url(${imageUrl})`;
      }

      function thumbnailClickHandler() {
        const selectedSrc = this.getAttribute('data-src');
        setBackground(selectedSrc);
        backgroundSelect.style.display = 'none';
      }

      thumbnails.forEach(thumbnail => {
      thumbnail.addEventListener('click', function () {
        const selectedSrc = this.getAttribute('data-src');
        setBackground(selectedSrc);
```

```
            backgroundSelect.style.display = 'none';
        });
    });

    const selectBackgroundBtn = document.getElementById('select-background-
btn');
    selectBackgroundBtn.addEventListener('click', function () {
        backgroundSelect.style.display = 'block';
    });
});
    // Створення світла та клавіатури.
    const light = new BABYLON.HemisphericLight("light", new
BABYLON.Vector3(0, 1, 0), scene);
    light.intensity = 0.6;

    // Створення вузла клавіатури та графічного інтерфейсу користувача для
клавіатури.
    const keyboard = new BABYLON.TransformNode("keyboard");

    const gui =
BABYLON.GUI.AdvancedDynamicTexture.CreateFullscreenUI("UI");

    // Функція для створення клавіш з вказаними параметрами та текстом.
    const buildKey = function (scene, parent, props, gui, text) {
        if (props.type === "white") {// Логіка створення білої клавіші та
текстового блоку.
            const bottom = BABYLON.MeshBuilder.CreateBox("whiteKeyBottom",
{ width: props.bottomWidth, height: 2, depth: 6.5 }, scene);

            const top = BABYLON.MeshBuilder.CreateBox("whiteKeyTop", {
width: props.topWidth, height: 2, depth: 6.5 }, scene);
            top.position.z = 4.75;
            top.position.x += props.topPositionX;

            const key = BABYLON.Mesh.MergeMeshes([bottom, top], true, false,
null, false, false);
            key.position.x = props.referencePositionX + props.wholePositionX;
            key.name = props.note + props.register;
            key.parent = parent;

            const container = new BABYLON.GUI.Rectangle();
            container.width = `${props.bottomWidth * 100}%`;
            container.height = "100%";
            container.background = "transparent";
```

```
            container.color = "rgba(0, 0, 0, 0)";
            container.linkOffsetY = 15;

            const textBlock = new BABYLON.GUI.TextBlock();
            textBlock.text = String.fromCharCode(text);
            textBlock.color = (props.type === "black") ? "white" : "black";
            textBlock.fontSize = 22;
            textBlock.textHorizontalAlignment =
BABYLON.GUI.Control.HORIZONTAL_ALIGNMENT_CENTER;
            textBlock.textVerticalAlignment =
BABYLON.GUI.Control.VERTICAL_ALIGNMENT_CENTER;

            container.addControl(textBlock);
            gui.addControl(container);

            container.linkWithMesh(key);
            container.isPointerBlocker = false;

            return key;
        } else if (props.type === "black") {// Логіка створення чорної клавіші та
текстового блоку.
            const blackMat = new BABYLON.StandardMaterial("black");
            blackMat.diffuseColor = new BABYLON.Color3(0, 0, 0);

            const key = BABYLON.MeshBuilder.CreateBox(props.note +
props.register, { width: 1.4, height: 3, depth: 6.5 }, scene);
            key.position.z += 4.75;
            key.position.y += 0.25;
            key.position.x = props.referencePositionX + props.wholePositionX;
            key.material = blackMat;
            key.parent = parent;
            key.name = props.note + props.register;

            const container = new BABYLON.GUI.Rectangle();
            container.width = "100%";
            container.height = "100%";
            container.background = "transparent";
            container.color = "rgba(0, 0, 0, 0)";
            container.linkOffsetY = 15;

            const textBlock = new BABYLON.GUI.TextBlock();
            textBlock.text = String.fromCharCode(text);
            textBlock.color = (props.type === "black") ? "white" : "black";
            textBlock.fontSize = 22;
```

```
            textBlock.textHorizontalAlignment =
BABYLON.GUI.Control.HORIZONTAL_ALIGNMENT_CENTER;
            textBlock.textVerticalAlignment =
BABYLON.GUI.Control.VERTICAL_ALIGNMENT_CENTER;

            container.addControl(textBlock);
            gui.addControl(container);

            container.linkWithMesh(key);
            container.isPointerBlocker = false;

            return key;
        }
    };

    // Змінні для позицій та тексту клавіш.
    let referencePositionX = -2.4 * 7;
    let num = 0;
    let text_note = 48;
    let text = 0;

    // Цикл для створення клавіш на клавіатурі за вказаними параметрами.
    for (let register = 3, register1 = 2; register <= 6; register++, register1++) {
        num = 0;
        keyParams.forEach(key => {// Логіка визначення параметрів клавіші та
тексту для клавіші.
            let keyParams = {
                register: (num >= 7) ? register1 : register,
                referencePositionX: referencePositionX,
                ...key
            };
            if (text_note == 58 || text_note == 64) {
                text_note++;
                text = text_note;
            }
            else if (text_note == 94) {
                text = 39;
            }
            else if (text_note == 97) {
                text = 32;
            }
            else if (text_note == 95) {
                text = 45;
            }
```

```
            else if (text_note == 60) {
                text = 44;
            }
            else if (text_note == 62) {
                text = 46;
            }
            else if (text_note == 63) {
                text = 47;
            }
            else {
                text = text_note;
            }
            buildKey(scene, keyboard, keyParams, gui, text);
            text_note++;
            num++;
        });
        referencePositionX += 2.4 * 7;
    }

    // Завантаження моделі Celesta та її розміщення в сцені.
    const celesta = new BABYLON.TransformNode("celesta");
    BABYLON.SceneLoader.ImportMesh("", "models/", "Celesta.glb", scene,
function (meshes) {
        meshes.forEach(function (mesh) {
            mesh.parent = celesta;
            mesh.scaling = new BABYLON.Vector3(1.55, 1.55, 1.8);
            mesh.position.y = -48;
            mesh.position.z = 106;
            mesh.position.x = -80;
        });
        celesta.rotation.y = Math.PI / 2;
    });

    // Створення мапи для відслідковування клавіш та обробники подій
натискання клавіш.
    const pointerToKey = new Map();
    const pressedKeys = {};
    const isPlaying = {};

    // Обробники подій натискання та відпускання клавіш на клавіатурі.
    function keyDownHandler(event) {
        const keyCode = event.key;
        const pianoKeyName = keyMappings[keyCode];
        if (pianoKeyName && !pressedKeys[keyCode]) {
```

```
        pressedKeys[keyCode] = true;
        const pianoKeyMesh = scene.getMeshByName(pianoKeyName);
        if (pianoKeyMesh) {
            pianoKeyMesh.position.y -= 0.5;
        }
        playAudio(pianoKeyName);
        showNoteInPianoRoll(pianoKeyName);
        console.log('Pressed note:', pianoKeyName);
        event.preventDefault();
        return false;
    }
}


function keyUpHandler(event) {
    const keyCode = event.key;
    const pianoKeyName = keyMappings[keyCode];
    if (pianoKeyName && pressedKeys[keyCode]) {
        pressedKeys[keyCode] = false;
        const pianoKeyMesh = scene.getMeshByName(pianoKeyName);
        if (pianoKeyMesh) {
            pianoKeyMesh.position.y += 0.5;
        }
        const keyData = pointerToKey.get(pianoKeyName);
        if (keyData && keyData.audio) {
            stopAudio(pianoKeyName);
        };
        event.preventDefault();
        return false;
    }
}


function showNoteInPianoRoll(note) {
    const pianoRoll = document.getElementById('pianoRoll');
    const noteElement = document.createElement('div');
    noteElement.textContent = note;
    noteElement.className = 'pianoNote';

    pianoRoll.appendChild(noteElement);
}

window.addEventListener('keydown', keyDownHandler);
window.addEventListener('keyup', keyUpHandler);
```

```
        // Функції для відтворення та зупинки аудіо під час натискання клавіш.
        function playAudio(pianoKeyName) {
            if (!isPlaying[pianoKeyName]) {
                const musicFileName =
`${encodeURIComponent(pianoKeyName)}.wav`;
                const audio = new Audio();
                audio.src = `sound/${musicFileName}`;
                audio.play();

                currentAudio = audio;
                isPlaying[pianoKeyName] = true;
                pointerToKey.set(pianoKeyName, { audio: audio });
            }
        }

        function stopAudio(pianoKeyName) {
            const keyData = pointerToKey.get(pianoKeyName);
            if (keyData && keyData.audio && isPlaying[pianoKeyName]) {

                setTimeout(() => {
                    keyData.audio.pause();
                    keyData.audio.currentTime = 0;
                    isPlaying[pianoKeyName] = false;
                    pointerToKey.delete(pianoKeyName);
                    currentAudio = null;
                }, 1000);
            }
        }

        // Обробник подій для клавіатури з можливістю взаємодії з клавішами.
        scene.onPointerObservable.add((pointerInfo) => {
            switch (pointerInfo.type) {
                case BABYLON.PointerEventTypes.POINTERDOWN:
                    if (pointerInfo.pickInfo.hit) {
                        let pickedMesh = pointerInfo.pickInfo.pickedMesh;
                        let pointerId = pointerInfo.event.pointerId;
                        if (pickedMesh.parent === keyboard) {
                            pickedMesh.position.y -= 0.5;
                            const pianoKeyName = pickedMesh.name;
                            playAudio(pianoKeyName);
                            pointerToKey.set(pointerId, { mesh: pickedMesh });
                        }
                    }
```

```
                break;
            case BABYLON.PointerEventTypes.POINTERUP:
                let pointerIdUp = pointerInfo.event.pointerId;
                if (pointerToKey.has(pointerIdUp)) {
                    const keyData = pointerToKey.get(pointerIdUp);
                    if (keyData && keyData.mesh) {
                        keyData.mesh.position.y += 0.5;
                        stopAudio(keyData.mesh.name);
                        pointerToKey.delete(pointerIdUp);
                    }
                }
                break;
        }
    });

    // Створення досвіду розширеної реальності (XR) для сцени.
    const xrHelper = await scene.createDefaultXRExperienceAsync();
    // Повернення створеної сцени для відображення у вікні браузера.
    return scene;
}

// Змінні та функції для запису та відтворення мелодій.
let isRecording = false;
let melodies = [];
const melodiesList = document.getElementById('melodies-list');
const recordedAudio = document.getElementById('recordedAudio');
const recordBtn = document.getElementById('record-btn');
let mediaRecorder;
let recordedChunks = [];

// Функція для початку або зупинки запису мелодій.
async function toggleRecording() {
    if (!mediaRecorder) {
        await startRecording();
    } else {
        stopRecording();
        await startRecording();
    }
}

async function toggleRecording() {
if (!isRecording) {
    await startRecording();
} else {
```

```
        stopRecording();
    }
}

// Функція для початку запису мелодій
async function startRecording() {
    recordedChunks = [];

    try {
        const stream = await navigator.mediaDevices.getDisplayMedia({ audio: true });
        mediaRecorder = new MediaRecorder(stream);

        mediaRecorder.ondataavailable = function (event) {
            if (event.data.size > 0) {
                recordedChunks.push(event.data);
            }
        };

        mediaRecorder.onstop = function () {
            const blob = new Blob(recordedChunks, { type: 'audio/wav' });
            const url = URL.createObjectURL(blob);
            const audio = new Audio();
            audio.src = url;
            melodies.push(audio);

            renderMelodies();
        };

        mediaRecorder.start();
        recordBtn.textContent = 'Stop Recording';
        isRecording = true;
    } catch (error) {
        console.error('Error accessing display media:', error);
    }
}

// Функція для зупинки запису мелодій.
function stopRecording() {
    if (mediaRecorder && mediaRecorder.state === 'recording') {
        mediaRecorder.stop();
        recordBtn.textContent = 'Start Recording';
        isRecording = false;
    }
}
```

```
function handleDataAvailable(event) {
    if (event.data.size > 0) {
        recordedChunks.push(event.data);
        const blob = new Blob(recordedChunks, { type: 'audio/wav' });
        const url = URL.createObjectURL(blob);
        recordedAudio.src = url;
        recordedAudio.controls = true;
        addMelodyToList(recordedChunks);
    }
}

recordBtn.addEventListener('click', toggleRecording);

// Додавання записаної мелодії до списку для відтворення.
function addMelodyToList(melody) {
    melodies.push(melody);
    renderMelodies();
}

// Оновлення відображення списку записаних мелодій на сторінці.
function renderMelodies() {
    melodiesList.innerHTML = '';
    melodies.forEach((melody, index) => {
        const item = document.createElement('li');
        item.classList.add('melody-item');
        item.textContent = 'Recorded Melody ' + (index + 1);

        const playBtn = document.createElement('button');
        playBtn.textContent = 'Play';
        playBtn.addEventListener('click', () => playMelody(melody));

        const deleteBtn = document.createElement('button');
        deleteBtn.textContent = 'Delete';
        deleteBtn.addEventListener('click', () => deleteMelody(index));

        item.appendChild(playBtn);
        item.appendChild(deleteBtn);
        melodiesList.appendChild(item);
    });
}

// Відтворення обраної мелодії.
```

```
function playMelody(melody) {
    melody.play();
}

// Видалення обраної мелодії зі списку.
function deleteMelody(index) {
    melodies.splice(index, 1);
    renderMelodies();
}

renderMelodies();

// Створення сцени та рендеринг у вікні браузера.
createScene(engine).then(sceneToRender => {
    engine.runRenderLoop(() => sceneToRender.render());
});

// Обробник зміни розміру вікна браузера для адаптації сцени.
window.addEventListener("resize", function () {
    engine.resize();
});
</script>
</body>
</html>
```