НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

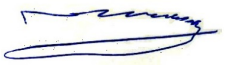
"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО"

Факультет електроніки
(повна назва інституту/факультету)

Акустичних та мультимедійних електронних систем
(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

 С. А. Найда

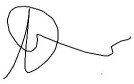
«_14» _____06_____ 2024 р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності (спеціалізації) 171 Електроніка (Електронні системи мультимедіа та засоби Інтернету речей)

на тему: Підготовка звукового контенту в інформаційному середовищі

Виконав: студент IV курсу, групи ДВ-01 

Алексеєнко Андрій Сергійович _____

Керівник Старший викладач Батіна О.А. 

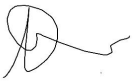
_____

Рецензент

доц. кафедри ЕІ, к.т.н. Попов А.О. 

_____

Засвідчую, що в цій дипломній роботі немає запозичень із праць інших авторів без відповідних посилань.

Студент Алексеєнко А.С. 

_____

Київ — 2024

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет електроніки

Кафедра Акустичних та мультимедійних електронних систем

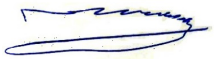
Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 171«Електроніка»

Освітньо-професійна програvма «Електронні системи мультимедіа та засоби Інтернету речей»

ЗАТВЕРДЖУЮ

Завідувач кафедри

 Сергій НАЙДА

«14 » травня 2024 р.

ЗАВДАННЯ

на дипломну роботу студенту

Алексеєнко Андрію Сергійовичу

1. Тема роботи «Підготовка звукового контенту в інформаційному середовищі», керівник роботи Батіна Олена Анатоліївна

затверджен наказом по університету від «23»травня2024р. №2076-с

2. Термін подання студентом роботи «07» червня 2024р.

3. Вихідні дані до роботи: Microsoft Silverlight, AdobeFlash, Java FX, HTML5, Ujam, RIA, WebAudioAPI

4. Зміст роботи: Характеристика мережних технологій для створення та обробки аудіо, огляд сучасних мережних технологій для роботи з аудіо файлами в мережі, створення онлайн додатків для роботи з аудіо.

5. Перелік ілюстративного матеріалу:набір слайдів презентації з узагальнюючими матеріалами та описом проведеного дослідження.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання: «15» травня 2024 р.

Календарний план

№ з/п	Назваетапіввиконання дипломноїроботи	Терміннвиконання етапівроботи	Примітка
1	Написання першого розділу	15.04.2024	виконано
2	Написання другого розділу	20.05.2024	виконано
3	Написання третього розділу	30.05.2024	виконано
4	Підготовка матеріалів до друку та оформлення пояснювальної записки	07.06.2024	виконано
5	Підготовка та оформлення презентації для доповіді	12.06.2024	виконано

Студент  Андрій АЛЕКСЕЄНКО

Керівник  Олена БАТІНА

УДК 621.397

РЕФЕРАТ

Алексеєнко А. С. Підготовка звукового контенту в інформаційному середовищі : дипломна робота бакалавра : 171 Електроніка. – Київ, 2024. – 75с.

Дипломну роботу виконано на 75 аркушах, вона містить 55 рисунків, 1додаток та перелік посилань на використані джерела з 10 найменувань.

Ключові слова: звуковий контент, обробка звуку, аудіоаналіз, WebAudio API, Full-stack, метадані, WEB-API.

Мета роботи: Метою роботи є дослідження та аналіз онлайн-інструментів та редакторів для створення, тестування та дебагування аудіо-додатків з використанням WebAudio API.

Об'єкт дослідження: Об'єктом дослідження є онлайн-інструменти та редактори для реалізації WebAudio API в браузері.

Методи дослідження: Методом дослідження є аналіз особливостей використання WebAudio API за допомогою різних онлайн-інструментів та редакторів, порівняння їх можливостей та функціоналу, а також практичне тестування шляхом створення коду для реалізації звукових ефектів та візуалізації аудіо. Крім того, застосовується науковий метод узагальнення для визначення ключових факторів, що впливають на ефективність розробки аудіо-додатків у браузері.

UDC621.397

ABSTRACT

Alekseienko A. S. Preparation of audio content in the information environment: bachelor's thesis: 171 Electronics. - Kyiv, 2024. - 75 p.

The thesis is presented in 75 pages. It contains 1appendixe and bibliography of 10 references.

Keywords: audio content, sound processing, audio analysis, Web Audio API, Full-stack, metadata, WEB-API.

Purpose: The purpose of the study is to research and analyze online tools and editors for creating, testing and debugging audio applications using the Web Audio API.

Object of research: The object of study is online tools and editors for implementing the Web Audio API in a browser.

Research methods. The research method is to analyze the features of using the Web Audio API with various online tools and editors, compare their capabilities and functionality, and conduct practical testing by creating code to implement sound effects and audio visualization. In addition, the scientific method of generalization is used to identify the key factors that affect the efficiency of developing audio applications in the browser.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (ApplicationProgrammingInterface) – Прикладний програмний інтерфейс

HTML (HyperTextMarkupLanguage) – Мова гіпертекстової розмітки

VPS (Virtual private server) – Віртуальний виділений сервер

SCP (Securecopyprotocol) – Захищений протокол копіювання даних

FTP (File transfer protocol) – Протокол передачі даних

RAM (Random Access Memory)–Пам’ять з довільним доступом

SQL(Structuredquerylanguage) - Мова структурованих запитів

S3 (SimpleStorageService) – Простий сервіс зберігання даних

SCP (Securecopyprotocol) – Захищений протокол копіювання даних

THD (Total Harmonic Distortion) – Коефіцієнтгармонік

IMD (InterModulationDistortion) - Рівень інтермодуляційних спотворень

FM (FrequencyModulation) - Частотно-модуляційний синтез

FFT (FastFourierTransform) – Алгоритм швидкого перетворення Фур’є

PCM (Pulsecodemodulation) - Імпульсно-кодова модуляція

SDM (Sigmadeltamodulation) - Сигма-дельта-модуляція

ВСТУП

Актуальність теми.На сьогоднішньому етапі розвитку сучасних технологій важливим аспектом є технічна реалізація сервісів для створення та цифрової обробки аудіо контенту, особливо з використанням мережевих технологій для дистанційного доступу. Такі сервіси повинні забезпечувати повний набір функцій для організації процесів зведення, мікшування та обробки аудіо даних, включаючи фінальний мастеринг. Їхнє використання спрощує процес запису та обробки аудіо, адже відпадає потреба в великому приміщенні для обладнання та придбанні дорогої апаратури для регулювання і мікшування звуку.

Постановка задачі. Проаналізувати літературні джерела щодо організації інформаційних технологій. Розглянути основні методи створення, обробки та мастерингу аудіо. Провести аналіз сучасного програмного забезпечення, визначивши його особливості та відмінності.

Обґрунтування необхідності проведення дослідження. Проведення цього дослідження є доцільним, оскільки інформаційні технології широко застосовуються в сучасному світі. Вони корисні як для новачків, що лише починають працювати зі звуком, так і для досвідчених користувачів, які вже мають певні навички у цій сфері.

Мета та завдання дослідження .Метою роботи є вивчення проблеми вибору програмного забезпечення для створення та обробки аудіо контенту з використанням WEB-технологій, аналіз технічних аспектів з виявленням основних переваг і недоліків.

Об’єкт дослідження. Об’єктом дослідження є інформаційні технології та методи реалізації створення і обробки аудіо контенту на прикладі провідних компаній у сфері хмарних сервісів і веб-технологій.

Предмет дослідження. Предметом дослідження є методи та способи отримання високоякісного аудіо контенту.

Методи дослідження. Методом дослідження є порівняльний аналіз способів реалізації різних провідних компаній у сфері інформаційних технологій та обробки аудіо контенту.

Практичне значення одержаних результатів.Отримані результати можуть бути використані для вибору оптимальних методів створення та обробки аудіо контенту.

1 СТВОРЕННЯ І ОБРОБКА АУДІО КОНТЕНТУ

Синтез звуку переслідує дві основні цілі: перша — це імітація різноманітних природних звуків, таких як шум вітру, дощу, звуки кроків, спів птахів тощо, а також звуків музичних інструментів (імітаційний синтез). Друга мета полягає в отриманні нових, унікальних звуків, які не зустрічаються в природі (чистий синтез).

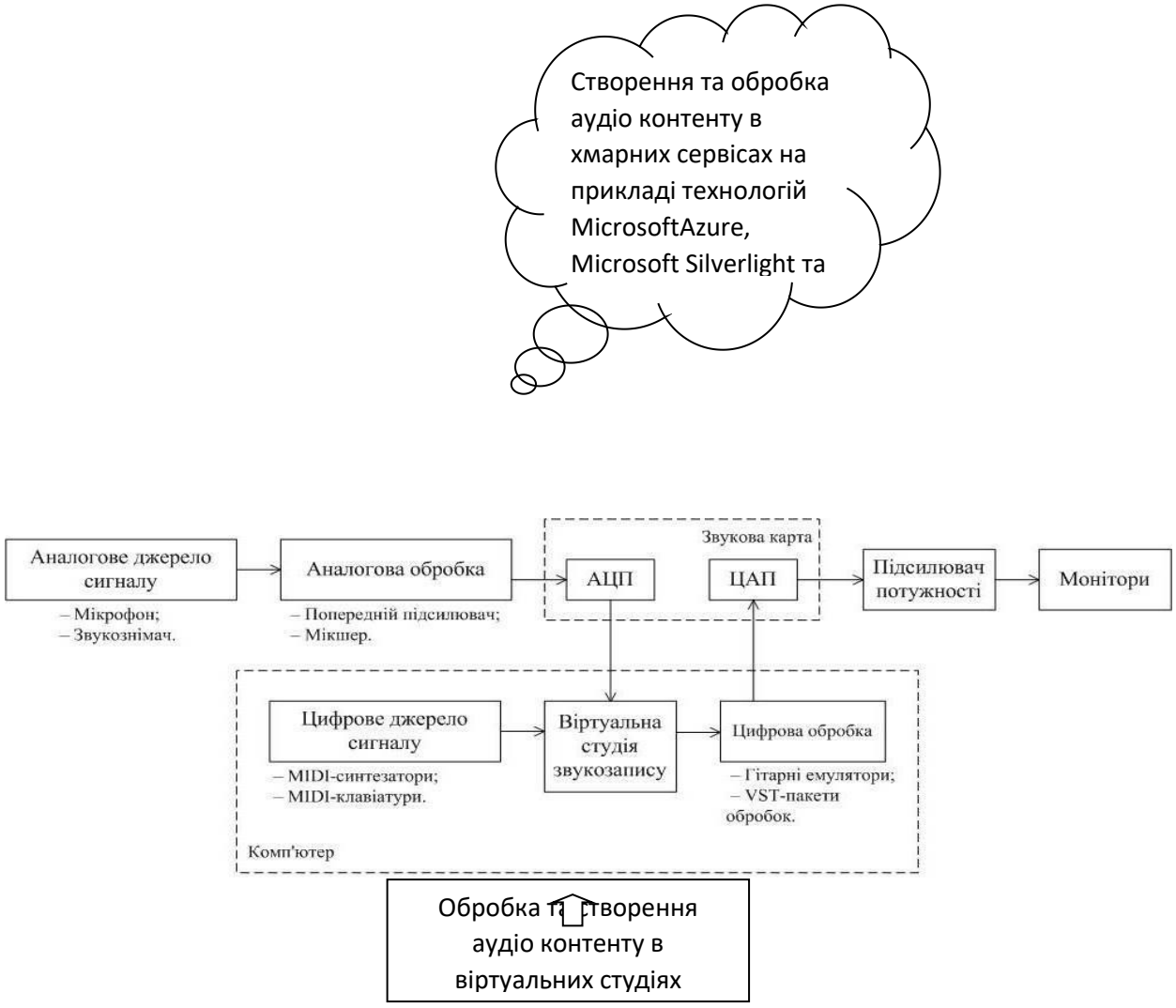


Рисунок 1.1 – Функціональна схема звукозапису

Обробка звуку зазвичай спрямована на трансформацію існуючих звуків для отримання нових (наприклад, створення ефекту «роботизованого голосу») або надання їм додаткових властивостей чи усунення недоліків (наприклад, додавання ефекту хору, видалення шуму або клацань). Кожен метод синтезу та обробки має свою математичну і алгоритмічну модель, що дозволяє реалізувати його на комп'ютері. Однак, точна реалізація багатьох методів вимагає значних обчислювальних ресурсів, тому вони часто реалізуються з певними спрощеннями.

На рис.1.1 приведена функціональна схема звукозапису, із зазначенням створення та обробкии аудіо контенту з використанням ресурсів комп’ютера (настільні застосунки), і безпосередньо в хмарних сервісах.

1.1 Властивості звуку

У звуці зазвичай розглядають амплітуду, спектральний склад звукового коливання та їхні зміни в часі.

Амплітуда визначає максимальну інтенсивність коливань, тобто гучність або силу звуку. На осцилограмі амплітуда представлена розмахом сигналу — найвищими та найнижчими рівнями відносно середнього значення. Спектральний склад визначає забарвлення або тембр звуку. Будь-яке періодичне коливання можна представити рядом Фур'є, тобто сумою кінцевого числа синусоїдальних коливань. Спектр звуку є графіком інтенсивностей цих частотних складових, представлених як вертикальні лінії. Спектр чистого тону має лише одну лінію, відповідну його частоті, тоді як спектр іншого коливання містить більше однієї лінії. Якщо на спектрі є чітко виражений пік, то звук сприймається як тон певної висоти, а інші складові визначають його тембр. Якщо пік відсутній, звук сприймається як суміш кількох тонів або шум. Частотні складові, кратні основній частоті тону, називаються гармоніками або обертонами. Гармоніки нумеруються від основного тону (перша гармоніка), а обертони — від першої кратної складової (перший обертон — друга гармоніка і т.д.)[2].

Через особливості слухового сприйняття висота звуку визначається більше за його спектральним складом, ніж за основним тоном. Наприклад, суб'єктивна висота більшості низькочастотних звуків із багатим спектром практично не змінюється навіть при видаленні основного тону, який відновлюється слуховим апаратом за різницевими частотами перших обертонів.

Зміна амплітуди в часі називається амплітудною обвідною звуку. На амплітудному графіку вона огинає графік коливання. Також існує поняття спектральної обвідної, яка є тривимірним графіком зміни спектра (і, відповідно, тембру) в часі.

Крім періодичних коливань (тонів), розглядаються також неперіодичні коливання — шуми. Для шуму характерне більш-менш рівномірне розподілення інтенсивності по спектру без виражених піків. Основні види шуму — білий і рожевий. Білий шум має рівномірну спектральну щільність і не зустрічається в природі, але часто трапляється в електронних приладах. Рожевий шум, щільність якого зменшується з ростом частоти (1/f), характерний для шумів дощу, прибою, вітру та інших природних шумів. Іноді розглядається також коричневий шум із щільністю (1/f^2), що швидко спадає з ростом частоти та близький до звуків ударного походження (грім, обвал)[2].

1.2 Звуковий тракт і його параметри

Звуковим трактом називають будь-який пристрій, що здійснює передачу або перетворення звуку. Звуковий тракт характеризується такими параметрами[10]:

- Номінальний вхідний і вихідний рівень (Input/OutputLevel): величина сигналу на вході та виході тракту, при якому він зберігає зазначені параметри. Вказується у вольтах і зазвичай приймається за 0 дБ.

- Максимальний вхідний і вихідний рівень: найбільша величина сигналу, при якій тракт зберігає працездатність. Рівні сигналів від номінального до максимального завжди мають ненульовий позитивний рівень.
- Коефіцієнт підсилення: відношення величини вихідного сигналу до вхідного. Вказується в разах, відсотках або децибелах.
- Діапазон частот (FrequencyResponse): частотний інтервал, в якому тракт зберігає свої основні характеристики. Якщо нижня межа діапазону дорівнює нулю, це означає, що пристрій може працювати з постійним струмом.
- Форма амплітудно-частотної характеристики (АЧХ): графік залежності амплітуди сигналу на виході від його частоти при незмінній амплітуді сигналу на вході. Тракти з горизонтальним АЧХ у межах частотного діапазону називають частотно-незалежними.
- Нерівномірність АЧХ: відхилення графіка від заданої форми. Вказується у відсотках або децибелах.
- Рівень шуму (NoiseLevel): величина шуму відносно номінального рівня сигналу. Вказується у децибелах і завжди має негативне значення. Також може вказуватися співвідношення сигнал/шум (SignaltoNoiseRatio, SNR), яке має позитивне значення. Іноді рівень шуму приводять до входу, припускаючи, що весь шум надходить тільки на вхід, а сам тракт власного шуму не має.
- Коефіцієнт гармонік (TotalHarmonicDistortion, THD): величина побічних гармонійних складових, що вносяться нелінійністю тракту. Вказується у відсотках від величини сигналу. У деяких випадках зазначають коефіцієнти для різних гармонік, причому найбільші спотворення зазвичай вносять непарні гармоніки вищих порядків.
- Рівень інтермодуляційних спотворень (InterModulationDistortion, IMD): відносний рівень паразитних частотних компонентів, породжених

взаємною модуляцією корисних складових сигналу. Вказується у відсотках від величини сигналу.

- Перехресне загасання (StereoCrosstalk): ступінь ослаблення сигналу при його проникненні в сусідній стереоканал. Вказується у децибелах.
- Динамічний діапазон (DynamicRange): діапазон найбільшого і найменшого рівнів сигналу, всередині якого зберігаються основні характеристики тракту. Знизу обмежений рівнем шуму, зверху — номінальним рівнем. Часто дорівнює співвідношенню сигнал/шум, проте нелінійність тракту може звужувати динамічний діапазон.

1.3 Частотні характеристики музичного звукоряду

Усі музичні звукоряди ґрунтуються на понятті октави – звуковисотного діапазону, де частоти крайніх звуків відрізняються вдвічі. Музичний звукоряд розбиває октаву на кілька ступенів (в європейській системі – дванадцять), які в будь-якій октаві мають однакові назви та зміст.

Існують два основних музичних звукоряди: натуральний і хроматичний. Натуральний звукоряд будується з обертонів базового звуку, зведених в одну октаву, тоді як хроматичний заснований на рівномірному розподілі октави на дванадцять ступенів. Співвідношення частот у натуральному звукоряді є раціональними дробами, тоді як сусідні ступені хроматичного звукоряду відрізняються на корінь 12-го ступеня з двійки – приблизно в 1.059 рази. Опорною нотою вважається нота Ля першої октави з частотою 440 Гц.

Використання натурального звукоряду дозволяє отримати більш гармонійне звучання, проте нерівномірність його ступенів ускладнює транспонування музики на інтервали, що не кратні октаві. Хроматичний звукоряд не забезпечує таких гармонійних співзвуч, проте завдяки рівномірності ступенів отримав ширше застосування[10].

1.4 Методи синтезу звуку

Адитивний синтез (additive)

Заснований на принципі Фур'є, що будь-яке періодичне коливання можна представити як суму чистих тонів (синусоїдальних коливань з різними частотами і амплітудами). Для цього використовують кілька синусоїдальних генераторів з незалежним управлінням, сигнали яких підсумовуються для створення остаточного звуку.

Різницевий синтез (subtractive)

Протилежний адитивному методу. Основу складає генерація звуку з багатим спектром (безліччю частотних компонентів), який потім фільтрується для виділення необхідних частот і ослаблення інших. В якості вихідних сигналів використовуються прямокутні, пилкоподібні і трикутні хвилі, а також різні види шумів. Основні інструменти синтезу – керовані фільтри (резонансний і фільтр нижніх частот).

Частотно-модуляційний синтез (frequencymodulation - FM)

Заснований на взаємній модуляції за частотою між кількома синусоїдальними генераторами, які називаються операторами. Різні способи з'єднання операторів, коли вихідні сигнали одних керують роботою інших, називаються алгоритмами синтезу. Завдяки простоті цифрової реалізації, метод широко використовується в студійній і концертній практиці (наприклад, Yamaha DX).

Семплерний метод (sample)

Передбачає запис реального звуку (семплу), який потім відтворюється. Для зміни висоти звуку відтворення прискорюється або сповільнюється. Щоб тембр звуку не змінювався при зміні висоти, використовують кілька записів через певні інтервали. Для зменшення обсягу пам'яті застосовується зациклення семпла.

Таблично-хвильовий синтез (wavetable)

Різновид семплерного методу, при якому записуються не всі звучання, а його окремі фази – атака, початкове загасання, середня фаза і кінцеве загасання. Ці фази записуються на різних частотах і при різних умовах, утворюючи сімейство звучань одного інструмента. При відтворенні ці фази комбінуються, що дозволяє при невеликому обсязі семплів отримати широкий спектр звучань.

Метод фізичного моделювання (physicalmodelling)

Моделює фізичні процеси, що визначають звучання реального інструмента, на основі його параметрів (наприклад, для скрипки – порода дерева, матеріал струн тощо). Цей метод дуже складний і вимагає великого обсягу обчислень, тому поки що використовується лише в студійних і експериментальних синтезаторах.

WaveGuide технологія

Розроблена в Стенфордському університеті і застосовується у деяких промислових моделях електронних роялів (наприклад, Baldwin). Моделює розповсюдження коливань по струні (одновимірне моделювання), по резонансним поверхням (двовимірне моделювання) або в об'ємному резонаторі (тривимірне моделювання). Це дозволяє моделювати нелінійні ефекти, такі як удар молоточка і дотик струни демпфером, а також взаємодію струн і зв'язок горизонтальних і вертикальних мод[10].

1.5 Методи обробки звуку

Монтаж

Включає вирізання, вставку, заміну та розмноження ділянок запису. Цей метод також називають редагуванням. Всі сучасні звукові та відеозаписи піддаються монтажу в тій чи іншій мірі.

Амплітудні перетворення

Виконуються шляхом змінення амплітуди сигналу, що зводиться до множення значень семплів на постійний коефіцієнт (підсилення/ослаблення) або до змінення амплітуди в часі за допомогою функції-модулятора (амплітудна модуляція). Окремим випадком є формування обвідної для надання звуку динамічного розвитку. Ці перетворення виконуються послідовно з окремими семплами, що робить їх простими в реалізації і не потребує великого обсягу обчислень.

Частотні (спектральні) перетворення

Виконуються над частотними складовими звуку. Використовуючи спектральне розкладання, де по горизонталі відкладені частоти, а по вертикалі – інтенсивності, можна реалізувати багато частотних перетворень як амплітудні перетворення над спектром. Наприклад, фільтрація (посилення або ослаблення певних частотних смуг) зводиться до накладення відповідної амплітудної обвідної на спектр. Однак частотну модуляцію таким чином не уявити – вона виглядає як зсув спектра або його окремих ділянок у часі. Частотні перетворення зазвичай виконуються за допомогою спектрального розкладання методом Фур'є, який потребує значних ресурсів. Проте існує алгоритм швидкого перетворення Фур'є (FFT), який дозволяє розгортати спектр сигналу середньої якості в реальному часі навіть на старих комп'ютерах.

Фазові перетворення

Зводяться до постійного зсуву фази сигналу або її модуляції деякою функцією чи іншим сигналом. Оскільки слух людини використовує фазу для визначення напрямку джерела звуку, фазові перетворення стереозвуку дозволяють створити ефекти обертання звуку, хору тощо. Зсув фази на 90-180 градусів (останнє досягається простим інвертуванням відліків) реалізує ефект «псевдооб'ємності» звуку (Surround).

Часові перетворення

Полягають у додаванні до основного сигналу його копій, зсунених у часі на різні величини. При зсувах, порівнянних з періодом сигналу, ці перетворення стають фазовими; при менших зсувах (менше 20 мс) це створює ефект хору (розмноження джерела звуку), при більших – ефекти багаторазового відбиття: реверберацію (20-50 мс) і відлуння (більше 50 мс).

Формантні перетворення

Є окремим випадком частотних перетворень і працюють з формантами – характерними смугами частот, що зустрічаються у звуках людської мови. Кожному звуку відповідає своє співвідношення амплітуд і частот кількох формант, яке визначає тембр і розбірливість голосу. Змінюючи параметри формант, можна підкреслювати або приглушувати окремі звуки, змінювати голосні, зрушувати регістр голосу тощо[10].

1.6 Звукові ефекти що застосовуються в музиці

Звукові ефекти значно покращують суб’єктивне сприйняття слухачем звукових образів, додаючи нові тембри, тональності тощо. Сучасні технології значно спрощують процес запису та обробки, зменшуючи потребу у великому просторі для інструментів та дорогій апаратурі для регулювання звуку і мікшування.

За допомогою різних комбінацій описаних вище перетворень створюються різні звукові ефекти. Найпоширеніші з них[7]:

Динамічна фільтрація (wah-wah)

Реалізується зміною частоти зрізу або смуги пропускання фільтра з невеликою частотою. На слух сприймається як обертання або закриття/відкриття джерела звуку, де збільшення високочастотних складових асоціюється з джерелом, спрямованим на слухача, а їх зменшення – з відхиленням від цього напрямку.

Вібрато

Амплітудна або частотна модуляція сигналу з невеликою частотою (до 10 Гц). Амплітудне вібрато також називають тремоло; на слух воно сприймається як тремтіння або завмирання звуку, а частотне – як «завивання» або «плавання» звуку.

Фленжер

Полягає в додаванні до вихідного сигналу його копій, зсунених у часі на невеликі величини (приблизно 3-30 мс) з можливою частотною модуляцією копій або величин їх тимчасових зрушень. На слух це відчувається як «дроблення», «розмазування» звуку, виникнення биття.

Фейзер

Змішування вихідного сигналу з його копіями, зсунутими по фазі. Величина зсуву може модулюватися в часі, створюючи ефекти «розмазування» звуку.

Реверберація

Досягається додаванням до вихідного сигналу загасаючої серії його затриманих у часі копій. Це імітує загасання звуку в приміщенні, коли звук набуває повноти і гучності, а потім поступово затихає.

Відлуння (Echo)

Відлуння з великим часом затримки (понад 50 мс). Слух починає сприймати відображення як повторення основного сигналу.

Дістошн (Distortion)

Навмисне спотворення форми звуку, надаючи йому різкий, скреготливий відтінок. Найбільш використовується як гітарний ефект у жанрі heavymetal.

Компресія

Стиснення динамічного діапазону сигналу, коли слабкі звуки посилюються сильніше, а сильні – слабше. Це зменшує різницю між тихим і гучним звучанням, запобігаючи перевантаженням і знижуючи рівень шуму.

Вокодер (VoiceCoder)

Синтез мови на основі довільного вхідного сигналу з багатим спектром. Мовний синтез зазвичай реалізується за допомогою формантних перетворень, виділяючи з сигналу потрібний набір формант, щоб надати сигналу властивості відповідного голосного звуку.

1.7 Переваги та особливості цифорової обробки звуку

Більшість популярних аналогових синтезаторів, які працюють на різницевому принципі, побудовані за модульною технологією, що сформувалася до кінця 70-х років. Ці синтезатори містять блоки Key, Env, VCO, VCA, VCF, LFO, NG, Mix та інші. Кожен з цих блоків незалежний і може з'єднуватися різними способами для отримання різних режимів синтезу.

На початку 80-х років почали впроваджуватися цифрові методи обробки, які спочатку комбінувалися з аналоговими, виконуючи свої специфічні функції. Наприклад, блоки Key, VCO, LFO, NG і Env легше реалізувати цифровим способом, тоді як Mix і VCF - аналоговим. Цифрові блоки через ЦАП подавали керуючі напруги на аналогові. Перевага цифрових формувачів полягає у вищій стабільності, точності та повторюваності сигналів.

У той же час з'явилися повністю цифрові FM-синтезатори, які не містили складних для цифрової реалізації керованих фільтрів. В середині 80-х років був освоєний випуск швидкодіючих DSP, і з'явилися повністю цифрові різницеві та семплерні синтезатори. Цифровий синтезатор по суті є комп’ютером з пристроями введення (клавіатура, кнопки, важелі, датчики, MIDI), виведення (звук, індикатори, MIDI), обробки (генератори, перетворювачі, пам’ять тощо) і центральним процесором[5].

Секвенсори та цифрові аудіо станції

Існує велика кількість секвенсорів або цифрових аудіо станцій (DAW) - програм, що дозволяють записувати і відтворювати багато аудіофайлів одночасно.

Ці програми надають можливість редагувати запис як завгодно: розрізати файл на частини, рухати, перетасовувати їх між собою, змінювати висоту звуку, швидкість відтворення, розтягувати звук тощо. Вони також підтримують роботу з MIDI, що дозволяє семплувати інструменти. Найвідоміші з них - Audacity, Cubase, FL Studio, Logic та Ableton. Кожна з цих програм має свої особливості і відмінності у функціональності, проте всі вони призначені для створення, запису та обробки звуку.

Переваги цифрового запису і обробки звуку

Цифрові технології запису і обробки звуку мають значні переваги над аналоговими:

- **Мінімальна кількість апаратури**: Зменшення потреби в громіздкій апаратурі для запису та обробки звуку.
- **Зберігання звуку**: Можливість зберігати звук у цифровому вигляді на ПК і на будь-яких зовнішніх носіях інформації.
- **Мінімальні спотворення**: Сучасні пристрої оцифрування звуку вносять мінімум спотворень.
- **Багатий спектр алгоритмів**: За допомогою звукових редакторів можна додавати ефекти, видаляти шуми, підвищувати якість звучання і багато іншого.
- **Копіювання без втрат**: Цифровий звуковий сигнал можна копіювати без втрат якості.
- **Редагування і копіювання**: Можливість редагувати, копіювати і збирати партії музикантів без втрат якості.
- **Повторюваність налаштувань**: Можливість зберігати і відтворювати налаштування ефектів з тією ж точністю.

Недоліки цифрового звуку

- **Втрата інформації**: Втрата частини інформації при оцифруванні звуку, що може призводити до спотворень при його відтворенні.

Рекомендації для якісного запису звуку

Для досягнення високоякісного запису звуку необхідно дотримуватися ряду правил, що враховують обмеження, введені електронними компонентами звукової карти. Це допоможе мінімізувати спотворення і зберегти якість запису на високому рівні[5].

1.8 Цифрові аудіо формати

Цифровий аудіоформат - це спосіб представлення звукових даних, що використовується для цифрового звукозапису і зберігання на комп'ютері або інших електронних носіях. Аудіофайли (файли, що містять звукозапис) зберігають інформацію про амплітуду і частоту звуку, дозволяючи відтворювати їх на комп'ютері або програвачі[10].

Формат звукових даних у цифровому вигляді залежить від методу квантування, що використовується цифро-аналоговим перетворювачем (ЦАП). У сучасній звукотехніці найбільш поширені два методи квантування:

- Імпульсно-кодова модуляція (PCM)
- Сигма-дельта-модуляція (SDM)

Формат файлу визначає структуру і спосіб представлення звукових даних на носії. Для зменшення розміру файлів використовуються аудіокодеки, що здійснюють стиснення аудіоданих. Види звукових форматів поділяються на три групи:

- Формати без стиснення: WAV, AIFF
- Формати зі стисненням без втрат: APE, FLAC
- Формати зі стисненням з втратами: MP3, OGG

Окремо варто згадати модульні музичні формати, створені синтетично або з використанням заздалегідь записаних семплів живих інструментів. Ці формати

(наприклад, MOD) часто використовуються для створення сучасної електронної музики.

Формати представлення звуку в комп'ютері

Оцифрований звук: Це точна цифрова копія зовнішніх звуків, таких як запис голосу з мікрофону або копія звукових доріжок з компакт-диску. Одна хвилина цифрового запису займає приблизно 10 мегабайт, що вимагає застосування методів стиснення.

Синтезований звук (MIDI): MIDI-мелодії складаються з команд, що керують звуковою картою, які вказують ноти, які вона має відтворити. Ця технологія дозволяє легко змінювати параметри мелодії і займає небагато місця (кілька десятків кілобайт). Однак, якість звучання залежить від якості звукової карти. MIDI-файли можна легко конвертувати у формат цифрового запису.

Трекерна або семплерна технологія: Поєднує цифровий і синтезований звук. Музична композиція складається з невеликих повторюваних фрагментів (петель або семплів). Цей принцип часто використовується для створення простої ритмічної танцювальної музики.

Види цифрових звукових носіїв

Цифрові формати використовуються як для масового поширення звукових записів (CD, SACD), так і в професійній звукозапису (DAT, MiniDisc).

Переваги цифрового звуку

- **Мінімальна кількість апаратури:** Зменшення потреби в громіздкому обладнанні для запису і обробки звуку.
- **Зберігання:** Можливість зберігати звук у цифровому вигляді на ПК і зовнішніх носіях.
- **Мінімальні спотворення:** Сучасні пристрої оцифрування звуку забезпечують мінімальні спотворення.

- **Широкий спектр обробки:** Застосування математичних алгоритмів для додавання ефектів, видалення шумів, підвищення якості звучання.
- **Копіювання без втрат:** Можливість копіювання цифрового сигналу без втрати якості.
- **Редагування:** Легкість редагування і копіювання музичних партій без втрати якості.
- **Повторюваність налаштувань:** Можливість зберігати і відтворювати налаштування ефектів з високою точністю.

Недоліки цифрового звуку

- **Втрата інформації:** При оцифруванні звуку може втрачатися частина інформації, що призводить до спотворень при відтворенні.

Рекомендації для якісного запису

Для досягнення високоякісного запису звуку важливо дотримуватися правил, що враховують обмеження електронних компонентів звукової карти, щоб мінімізувати спотворення і зберегти високу якість запису[5].

2 ЗАСОБИ РЕАЛІЗАЦІЇ

2.1 HTML5 – технологія створення насичених веб-додатків

Минулі технології насичених веб-додатків

До появи HTML5, створення насичених веб-додатків було складним завданням, яке вимагало використання сторонніх технологій, таких як Flash, Silverlight та JavaFX. Кожна з цих технологій мала свої переваги та недоліки, які вплинули на їх прийняття та використання в індустрії веб-розробки.

Flash

Flash, розроблений компанією Macromedia, а згодом придбаний Adobe, був однією з найпопулярніших технологій для створення інтерактивного контенту, анімацій та ігор на веб-сторінках. Flash забезпечував потужні можливості для роботи з графікою, звуком та відео, що дозволяло створювати багатофункціональні веб-додатки[4].

Переваги Flash:

- Інтерактивна графіка та анімація: Flash дозволяв створювати складні анімації та інтерактивні елементи, що робило веб-сторінки привабливими та динамічними.
- Підтримка потокового відео та аудіо: FlashPlayer міг обробляти потоковий мультимедійний контент, що дозволяло транслювати відео та аудіо у режимі реального часу.
- Широка підтримка браузерами: На початку 2000-х FlashPlayer був встановлений майже на всіх персональних комп'ютерах, що робило його доступним для більшості користувачів.

Недоліки Flash:

- Потреба в плагіні: Flash вимагав встановлення стороннього плагіну, що створювало додаткові кроки для користувачів та потенційні проблеми з безпекою.
- Продуктивність: Flash додатки часто вимагали значних обчислювальних ресурсів, що могло негативно впливати на продуктивність комп'ютера та швидкість роботи браузера.

- Проблеми з безпекою: Flash був вразливий до численних вразливостей, що ставали цілями для хакерів і зловмисного програмного забезпечення.
- Відсутність підтримки на мобільних пристроях: Apple відмовилася підтримувати Flash на своїх мобільних пристроях, таких як iPhone та iPad, що суттєво знизило його популярність з появою мобільних додатків.

Silverlight

Silverlight, розроблений компанією Microsoft, був альтернативою Flash та пропонував можливості для створення багатофункціональних веб-додатків з інтерактивними елементами, графікою та мультимедіа[4,8].

Переваги Silverlight:

- Інтеграція з .NET Framework: Silverlight використовував .NET Framework, що дозволяло розробникам з легкістю створювати додатки з використанням мов програмування C# та VB.NET.
- Підтримка високоякісного відео: Silverlight забезпечував відмінну якість відтворення відео, включаючи підтримку HD відео.
- Можливості для бізнес-додатків: Завдяки підтримці інтерактивних форм та інших елементів UI, Silverlight був зручним для створення бізнес-додатків з багатими інтерфейсами.

Недоліки Silverlight:

- Обмежена підтримка браузерами: Silverlight працював тільки на Windows та MacOS, що обмежувало його використання.
- Потреба у плагіні: Як і Flash, Silverlight вимагав встановлення додаткового плагіна.
- Відсутність мобільної підтримки: Silverlight не підтримувався на багатьох мобільних платформах, що зменшувало його привабливість для розробників.

JavaFX

JavaFX, розроблений компанією SunMicrosystems, а згодом придбаний Oracle, був ще однією спробою створити багатофункціональні веб-додатки з використанням технологій Java[4].

Переваги JavaFX:

- Потужна мова програмування: JavaFX використовував мову програмування Java, що забезпечувало високу продуктивність та можливості для створення складних додатків.
- Підтримка 2D та 3D графіки: JavaFX дозволяв створювати графіку високої якості, включаючи 2D та 3D елементи.
- Кросплатформеність: JavaFX додатки могли працювати на різних операційних системах, включаючи Windows, MacOS та Linux.

Недоліки JavaFX:

- Великий об'єм: JavaFX додатки часто вимагали великого об'єму пам'яті та ресурсів.
- Потреба у плагіні: Як і інші технології, JavaFX вимагав встановлення додаткового плагіна.
- Відсутність підтримки на мобільних пристроях: JavaFX мав обмежену підтримку на мобільних платформах.

Переваги HTML5 над минулими технологіями

HTML5 став суттєвим кроком вперед у веб-розробці, вирішивши багато проблем та обмежень, що були притаманні попереднім технологіям. Ось деякі з ключових переваг HTML5:

- Відсутність потреби у плагінах: HTML5 працює безпосередньо у браузері без потреби у встановленні додаткових плагінів. Це забезпечує вищий рівень безпеки та зручності для користувачів.

- Широка підтримка браузерами: Всі сучасні браузери підтримують HTML5, що робить його доступним для більшості користувачів без необхідності додаткових налаштувань.
- Потужні мультимедійні можливості: HTML5 включає в себе вбудовані теги для роботи з аудіо та відео (наприклад, <audio> та <video>), що дозволяє легко інтегрувати мультимедійний контент у веб-сторінки.
- Семантичні теги: HTML5 включає нові семантичні теги, такі як <header>, <footer>, <article>, та <section>, що покращують структуру та читабельність коду.
- Підтримка сучасних веб-технологій: HTML5 забезпечує інтеграцію з іншими сучасними веб-технологіями, такими як CSS3 та JavaScript, що дозволяє створювати багатофункціональні та інтерактивні веб-додатки.
- Кросплатформеність: HTML5 додатки можуть працювати на різних пристроях та операційних системах, включаючи десктопи, планшети та смартфони.

Висновок

HTML5 значно спростив роботу з аудіо контентом у веб-додатках, надавши потужні інструменти для вбудовування та управління аудіо файлами. Відмова від використання сторонніх плагінів, широка підтримка браузерами та потужні мультимедійні можливості роблять HTML5 ідеальним вибором для створення насичених веб-додатків. Завдяки елементам, таким як <audio>, <source>, та <track>, розробники можуть легко інтегрувати аудіо контент у свої веб-проекти, забезпечуючи користувачів зручним та захоплюючим аудіо-експерієнсом[6].

2.1.1 ЕлементиHTML для роботи з аудіо контентом

HTML5 надає розробникам широкий набір інструментів для роботи з аудіо контентом. Основними елементами HTML5 для вбудовування, відтворення та управління аудіо файлами є `<audio>`, `<source>`, `<track>`, а також можливості

програмного інтерфейсу JavaScript через об'єкт `HTMLAudioElement`. Розглянемо докладніше ці елементи та їх використання у веб-додатках[6].

Тег <audio>

Тег <audio> є центральним елементом для вбудовування аудіо файлів у веб-сторінки. Він дозволяє додавати аудіо контент безпосередньо у HTML-код і надає різні атрибути для налаштування відтворення аудіо.

Основні атрибути тега<audio>:

- `src`: вказує шлях до аудіо файлу.
- `controls`: додає елементи керування для аудіо, такі як кнопки відтворення, паузи та регулювання гучності.
- `autoplay`: вказує, що аудіо має автоматично відтворюватися при завантаженні сторінки.
- `loop`: вказує, що аудіо має повторюватися після завершення відтворення.
- `muted`: вказує, що аудіо має бути без звуку за замовчуванням.
- `preload`: визначає, як браузер має завантажувати аудіо ("none", "metadata", "auto").

```
<audio src="audiofile.mp3" controls></audio>
```

Рис 2.1 - Приклад використання тега<audio>

Цей приклад вставляє аудіо файл `audiofile.mp3` у веб-сторінку та додає елементи керування для його відтворення.

```
<audio src="audiofile.mp3" controls autoplay loop muted preload="auto"></audio>
```

Рис 2.2 - Комбінація атрибутівтега<audio>

Цей приклад демонструє використання всіх основних атрибутів тега `<audio>`, що забезпечує автоматичне відтворення аудіо, його повторення, вимкнення звуку за замовчуванням та автоматичне завантаження файлу при завантаженні сторінки.

Тег <source>

Тег `<source>` використовується для вказання кількох джерел аудіо файлів у межах тега `<audio>`. Це дозволяє браузеру вибирати найкращий доступний формат аудіо файлу для відтворення залежно від його підтримки.

Атрибути тега<source>:

- `src`: вказує шлях до аудіо файлу.
- `type`: вказує тип аудіо файлу (наприклад, `audio/mpeg` для MP3, `audio/ogg` для Ogg).

```
<audio controls>
  <source src="audiofile.mp3" type="audio/mpeg">
  <source src="audiofile.ogg" type="audio/ogg">
  Your browser does not support the audio element.
</audio>
```

Рис 2.3 - Приклад використання тегів <audio> та <source>

У цьому прикладі браузер спочатку спробує відтворити аудіо файл у форматі MP3. Якщо цей формат не підтримується, браузер перейде до відтворення аудіо файлу у форматі Ogg.

Тег <track>

Тег <track> використовується для додавання текстових доріжок до аудіо та відео контенту, таких як субтитри, описи та метадані. Це робить мультимедійний контент більш доступним для користувачів з обмеженими можливостями.

Атрибути тега<track>`:

- `kind`: вказує тип текстової доріжки (наприклад, `subtitles`, `captions`, `descriptions`, `chapters`, `metadata`).
- `src`: вказує шлях до файлу текстової доріжки.
- `srclang`: вказує мову текстової доріжки.
- `label`: вказує назву текстової доріжки.
- `default`: вказує, що ця доріжка має бути активною за замовчуванням.

```
<audio controls>
  <source src="audiofile.mp3" type="audio/mpeg">
  <track kind="captions" src="subtitles_en.vtt" srclang="en" label="English">
</audio>
```

Рис 2.4 - Приклад використання тега<track>

У цьому прикладі додається текстова доріжка з субтитрами на англійській мові до аудіо файлу.

Використання JavaScript для управління аудіо

HTML5 також надає можливості для програмного управління аудіо контентом за допомогою JavaScript. Об'єкт `HTMLAudioElement` дозволяє розробникам динамічно створювати та керувати аудіо елементами на веб-сторінках.

```
var audio = new Audio('audiofile.mp3');
audio.controls = true;
document.body.appendChild(audio);
```

Рис 2.5 - Створення аудіо елементу за допомогою JavaScript

Цей приклад створює новий аудіо елемент з використанням JavaScript, додає елементи керування та вставляє його у тіло веб-сторінки.

Управління відтворенням аудіо за допомогою JavaScript:

```
<button onclick="playAudio()">Play Audio</button>
<button onclick="pauseAudio()">Pause Audio</button>
<script>
  var audio = new Audio('audiofile.mp3');

  function playAudio() {
    audio.play();
  }

  function pauseAudio() {
    audio.pause();
  }
</script>
```

Рис 2.6 - ВикористанняJavaScript функції для відтворення та паузи аудіо файлу.

Обробка подій аудіо елементу:

Об'єкт `HTMLAudioElement` підтримує різні події, які можуть використовуватися для реагування на зміни стану аудіо елементу.

```
audio.addEventListener('play', function() {
  console.log('Audio is playing');
});

audio.addEventListener('pause', function() {
  console.log('Audio is paused');
});

audio.addEventListener('ended', function() {
  console.log('Audio has ended');
});
```

Рис 2.7 - Приклад обробки подій за допомогою HTMLAudioElement

У цьому прикладі додаються обробники подій для подій `play`, `pause` та `ended`, які виводять повідомлення у консоль браузера при відповідних змінах стану аудіо.

Сумісність браузерів та підтримка форматів

Однією з ключових переваг HTML5 є його широка підтримка сучасними браузерами. Тег `<audio>` підтримується всіма основними браузерами, такими як GoogleChrome, MozillaFirefox, Microsoft Edge, Safari та Opera. Однак підтримка різних форматів аудіо файлів може відрізнятися залежно від браузера.

Підтримка форматів аудіо файлами основними браузерами:

Формат аудіо	**MIME-тип**	**Chrome**	Firefox	Safari	Edge	Opera
MP3	audio/mpeg	Так	Так	Так	Так	Так

OggVorbis	audio/ogg	Так	Так	Ні	Так	Так
WAV	audio/wav	Так	Так	Так	Так	Так
AAC	audio/aac	Так	Так	Так	Так	Так
WebM	audio/webm	Так	Так	Ні	Так	Так

Щоб забезпечити максимальну сумісність аудіо контенту з різними браузерами, рекомендується використовувати кілька джерел аудіо файлів у різних форматах.

```
<audio controls>
  <source src="audiofile.mp3" type="audio/mpeg">
  <source src="audiofile.ogg" type="audio/ogg">
  <source src="audiofile.wav" type="audio/wav">
  Your browser does not support the audio element.
</audio>
```

Рис 2.8 -Приклад забезпечення сумісності форматів у браузері

2.2 WEB Audio API

WebAudio API є сучасним інтерфейсом для роботи з аудіо у веб-браузерах. Цей API дозволяє створювати, обробляти та відтворювати складні аудіо ефекти та композиції у режимі реального часу, використовуючи JavaScript. WebAudio API є потужним інструментом для розробників, які хочуть інтегрувати аудіо у свої веб-додатки, надаючи широкі можливості для роботи з аудіо контентом.

2.2.1 Загальні концепції WebAudio API

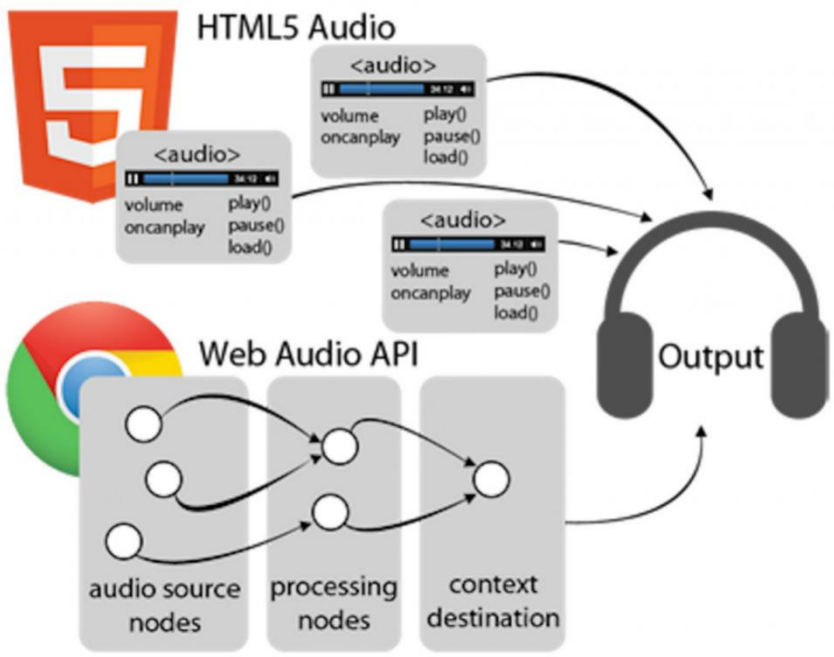


Рис. 2.9 – Схема обробки звуку за допомогою HTML5 + Web Audio Api

WebAudio API побудований на кількох основних концепціях, які дозволяють створювати складні аудіо-композиції та ефекти. Основними елементами цієї архітектури є аудіо-контекст, вузли (нод) та аудіо-граф. Розглянемо кожну з цих концепцій докладніше.

Аудіо-контекст (AudioContext)

Аудіо-контекст є основною точкою входу для роботи з WebAudio API. Він служить контейнером для всіх аудіо-вузлів і обробляє їх взаємодію. Аудіо-контекст надає можливість створювати джерела звуку, додавати ефекти, змішувати різні звукові сигнали і багато іншого.

```
var audioContext = new (window.AudioContext || window.webkitAudioContext)();
```

Рис 2.10 - Приклад створення аудіо-контексту:

Цей код створює новий аудіо-контекст, який можна використовувати для створення і обробки аудіо.

Вузли (AudioNodes)

Вузли є основними будівельними блоками в WebAudio API. Кожен вузол виконує певну функцію, наприклад, генерує звук, обробляє його або відтворює. Вузли можуть бути з'єднані між собою, утворюючи аудіо-граф, через який проходить звуковий сигнал.

Типи вузлів:

1. Джерела звуку (SourceNodes): ці вузли генерують звукові сигнали. Вони можуть бути створені з аудіо-файлів, потокового аудіо або синтетичних генераторів звуку.
2. Обробні вузли (ProcessingNodes): ці вузли модифікують або аналізують аудіо-сигнал. Приклади включають фільтри, динамічні компресори, ревербератори та інші ефекти.
3. Вузли призначення (DestinationNodes): ці вузли визначають кінцеву точку звукового сигналу, наприклад, динаміки або навушники.

```
var oscillator = audioContext.createOscillator();
oscillator.type = 'sine';
oscillator.frequency.setValueAtTime(440, audioContext.currentTime);
oscillator.connect(audioContext.destination);
oscillator.start();
```

Рис 2.11 - Приклад створення джерела звуку

Цей код створює простий генератор звуку (осцилятор), який генерує синусоїдальний сигнал частотою 440 Гц і відправляє його на відтворення.

Аудіо-граф (AudioGraph)

Аудіо-граф представляє собою набір вузлів, з'єднаних між собою для створення ланцюгів обробки звуку. Звуковий сигнал проходить через ці вузли в певному порядку, дозволяючи створювати складні звукові ефекти і композиції.

```
var gainNode = audioContext.createGain();
oscillator.connect(gainNode);
gainNode.connect(audioContext.destination);
gainNode.gain.setValueAtTime(0.5, audioContext.currentTime);
```

Рис 2.12 - Приклад побудови аудіо-графу

У цьому прикладі осцилятор з'єднується з вузлом підсилення (gainNode), який регулює гучність сигналу, а потім підсилений сигнал надходить до призначеного вузла (destinationnode) для відтворення.

Аудіо-буфери (AudioBuffers)

Аудіо-буфери використовуються для зберігання аудіо даних у пам'яті для подальшої обробки і відтворення. Вони можуть бути завантажені з файлів або створені динамічно.

```
var audioBuffer;
var request = new XMLHttpRequest();
request.open('GET', 'path/to/audio/file.mp3', true);
request.responseType = 'arraybuffer';

request.onload = function() {
  audioContext.decodeAudioData(request.response, function(buffer) {
    audioBuffer = buffer;
  });
}
request.send();
```

Рис 2.13 - Приклад завантаження аудіо-буфера

Цей код завантажує аудіо файл і декодує його в аудіо-буфер для подальшого використання.

Відтворення та обробка звуку

WebAudio API надає можливість не тільки відтворювати аудіо, але й здійснювати складні маніпуляції з ним. Це включає зміну частоти, додавання ефектів, обробку сигналу в режимі реального часу і багато іншого.

```
var source = audioContext.createBufferSource();
source.buffer = audioBuffer;
source.connect(audioContext.destination);
source.start();
```

Рис 2.14 - Приклад відтворення звуку з аудіо-буфера

Цей код створює джерело звуку з аудіо-буфера і відтворює його.

Обробка звуку в режимі реального часу

Однією з ключових особливостей WebAudio API є можливість обробляти звук в режимі реального часу, що дозволяє створювати інтерактивні додатки з динамічно змінюваним аудіо контентом.

```
var biquadFilter = audioContext.createBiquadFilter();
biquadFilter.type = 'lowshelf';
biquadFilter.frequency.setValueAtTime(1000, audioContext.currentTime);
biquadFilter.gain.setValueAtTime(25, audioContext.currentTime);

oscillator.connect(biquadFilter);
biquadFilter.connect(audioContext.destination);
```

Рис 2.15 - Приклад обробки звуку з використанням biquad фільтра

Цей код додає до осцилятора низькочастотний фільтр, що підсилює низькі частоти сигналу.

Аналіз звукового сигналу

WebAudio API також дозволяє аналізувати звукові сигнали, що може бути корисним для створення аудіо-візуалізацій та інших інтерактивних додатків.

```
var analyser = audioContext.createAnalyser();
oscillator.connect(analyser);
analyser.connect(audioContext.destination);

var dataArray = new Uint8Array(analyser.frequencyBinCount);
function draw() {
  requestAnimationFrame(draw);
  analyser.getByteFrequencyData(dataArray);
  // код для візуалізації даних
}
draw();
```

Рис 2.16 - Приклад використання аналізатора

Цей код створює аналізатор звуку, який зчитує дані частоти звукового сигналу і передає їх для візуалізації.

Просторовий звук

WebAudio API підтримує просторовий звук, що дозволяє створювати ефекти тривимірного звуку, коли джерела звуку розташовані у віртуальному просторі навколо користувача.

```
var panner = audioContext.createPanner();
oscillator.connect(panner);
panner.connect(audioContext.destination);
panner.setPosition(1, 0, 0); // розташування джерела звуку праворуч від слухача
```

Рис 2.17 - Приклад створення просторового звуку

Цей код створює панорамування звуку, що розташовує джерело звуку праворуч від користувача, створюючи ефект просторового звуку.

Використання зовнішніх бібліотек

Хоча WebAudio API є потужним інструментом сам по собі, існують також численні зовнішні бібліотеки, які спрощують роботу з ним і надають додаткові можливості. Деякі з популярних бібліотек включають Tone.js, Pizzicato.js та Howler.js.

```
var synth = new Tone.Synth().toMaster();
synth.triggerAttackRelease('C4', '8n');
```

Рис 2.18 - Приклад використання Tone.js

Цей код створює синтезатор з використанням бібліотеки Tone.js і відтворює ноту C4 тривалістю восьма нота.

Приклади використання WEB Audio API

- Музичні додатки: WebAudio API дозволяє створювати повноцінні музичні додатки, такі як синтезатори, секвенсери і драм-машини. Завдяки можливості обробки звуку в режимі реального часу, можна створювати інструменти, які реагують на дії користувача у реальному часі.
- Ігри: У ігровій розробці WebAudio API використовується для створення ефектів навколишнього середовища, звукових ефектів та музичних треків, що покращує ігровий досвід за рахунок високоякісного звуку і його інтерактивності.
- Аудіо-візуалізації: WebAudio API надає інструменти для аналізу звукового сигналу, що дозволяє створювати вражаючі візуалізації звуку. Зокрема, використання аналізатора частоти дозволяє отримувати дані про амплітуду на різних частотах, які можуть бути використані для створення інтерактивних графічних ефектів, що синхронізуються зі звуком.
- Програмне забезпечення для обробки звуку: WebAudio API дозволяє створювати онлайн додатки для обробки звуку, такі як редактори

звукових доріжок, додатки для обробки голосу та інші інструменти для роботи зі звуком. Це відкриває можливості для роботи зі звуком без необхідності встановлювати спеціалізоване програмне забезпечення на комп'ютер.

- Ефекти та обробка звуку в реальному часі: З використанням WebAudio API можна створювати різноманітні звукові ефекти, такі як реверберація, затримка, еквалізація та багато інших. Ці ефекти можуть бути застосовані до звукових сигналів у реальному часі, що дозволяє створювати інтерактивні додатки, де користувач може змінювати звук "на льоту".

Вузли обробки звуку

- GainNode: Вузол зміни гучності (GainNode) дозволяє регулювати амплітуду аудіосигналу. Це один з найпростіших і найчастіше використовуваних вузлів у WebAudio API.
- BiquadFilterNode: Вузол біквадратного фільтра (BiquadFilterNode) надає можливість застосовувати до сигналу різні типи фільтрації, такі як низькочастотний, високочастотний, смуговий фільтр тощо. Це дозволяє здійснювати тонке налаштування частотного спектра звуку.
- DelayNode: Вузол затримки (DelayNode) дозволяє додавати до сигналу ефект затримки, створюючи ехо або інші схожі ефекти. Можна налаштовувати час затримки, що дозволяє створювати різноманітні звукові ефекти.
- ConvolverNode: Вузол згортки (ConvolverNode) використовується для створення реверберації. За допомогою цього вузла можна моделювати акустичні властивості різних приміщень, застосовуючи до звукового сигналу відповідні імпульсні відповіді.

- AnalyserNode: Вузол аналізу (AnalyserNode) використовується для отримання даних про частотний спектр та амплітуду звукового сигналу. Ці дані можуть бути використані для візуалізації звуку або інших цілей, де потрібен аналіз сигналу.
- StereoPannerNode: Вузол панорамування (StereoPannerNode) дозволяє змінювати положення звукового сигналу у стереопанорамі, регулюючи його баланс між лівим і правим каналами.
- PannerNode: Вузол панорамування у тривимірному просторі (PannerNode) дозволяє розміщувати звукові джерела у тривимірному просторі, створюючи ефекти просторового звуку.

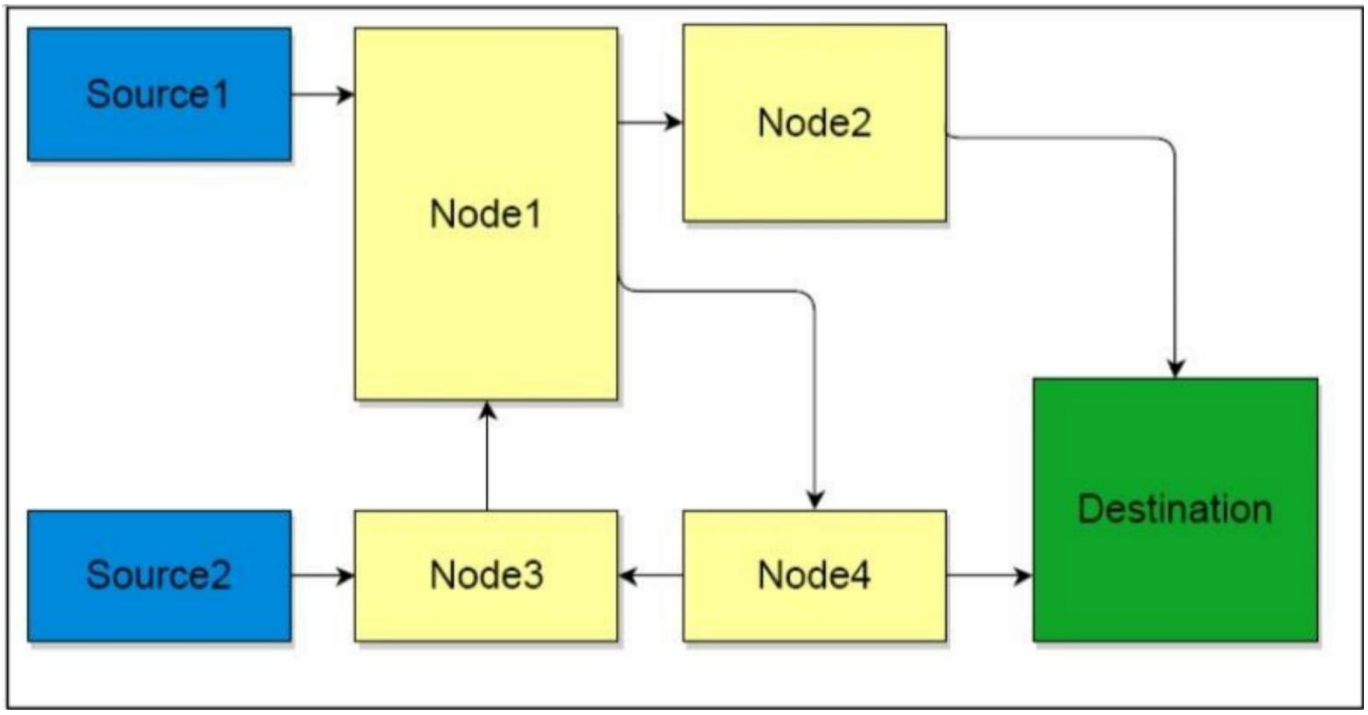


Рис 2.19 - Схема з'єднання вузлів WEBAudioAPI

2.2.2 Інтеграція WebAudio API з іншими технологіями

WebGL: Інтеграція WebAudioAPI з WebGL дозволяє створювати синхронізовані аудіо-візуалізації, де звук керує графічними ефектами. Це особливо корисно для створеннямузичнихкліпів, інтерактивнихпрезентацій та ігор.

MediaStream API: Використання Web Audio API разом з MediaStream API дозволяєобробляти аудіо дані з мікрофону або інших зовнішніх джерел у режимі

реального часу. Цевідкриваєможливості для створеннядодатків для відеоконференцій, онлайн-трансляцій та обробки живого звуку.

Service Workers: Застосування Web Audio API у поєднанні з Service Workersдозволяєстворюватиавтономніаудіо-додатки, які можуть працювати без доступу до інтернету. Це особливо корисно для музичнихплеєрів та іншихаудіо-додатків.

2.2.3 Використання WebAudio API для створення музичних інструментів

Синтезатори: За допомогою WebAudioAPI можна створювати віртуальні синтезатори, які дозволяють генерувати і модулювати звукові хвилі різних форм. Розробники можуть створювати інтерфейси для керування параметрами синтезаторів, такими як частота, форма хвилі, амплітуда та інші.

Драм-машини: Web Audio API дозволяєстворювати драм-машини, якіможутьвідтворюватиритмічніпатерни, використовуючипопередньозавантажені звуки ударних інструментів. Розробники можуть реалізовувати різноманітні функції для керування ритмом, темпом і гучністю.

Секвенсери: Використовуючи Web Audio API, можнастворюватисеквенсери, якідозволяютьпрограмувати та відтворюватимузичніпатерни. Цеможе бути корисним для створеннямузичнихдодатків, де користувачіможутьстворюватисвоївласні треки, розташовуючи звуки на часовійшкалі.

2.2.4 Недоліки та майбутнєWebAudio API

Продуктивність: Обробка аудіо у режимі реального часу може вимагати значних ресурсів процесора, особливо якщо одночасно обробляється багато звукових джерел або застосовуються складні ефекти. Цеможе бути викликом для мобільнихпристроївабокомп'ютерів з низькоюпродуктивністю.

Сумісність: Хоча Web Audio API підтримуєтьсябільшістюсучаснихбраузерів, все ще існують деякі проблеми сумісності між різними платформами. Розробникиповиннівраховуватиціособливості та забезпечуватисумісністьсвоїхдодатків.

Безпека: Робота з аудіо даними, особливо з мікрофону або інших зовнішніх джерел, може викликати проблеми безпеки та конфіденційності. Розробники повинні дотримуватися найкращих практик для забезпечення безпеки даних користувачів.

Web Audio API продовжуєрозвиватися, додаючинові функції та покращення. Очікується, що в майбутньому з'являться нові можливості для обробки звуку, підтримка нових форматів аудіо та покращення продуктивності. Цезробить Web Audio API щебільшпотужнимінструментом для розробкиаудіо-додатків.

2.2.5 WebAudio API інтерфейси

WebAudioAPI надає широкий набір інтерфейсів для створення, обробки і відтворення аудіо в веб-додатках. Ці інтерфейси дозволяють розробникам реалізовувати складні звукові ефекти, аналізувати аудіо-сигнали та створювати інтерактивні аудіо-додатки. Розглянемо основні інтерфейси WebAudioAPI.

AudioContext

AudioContext є основним інтерфейсом для роботи з WebAudioAPI. Він служить точкою входу для створення та управліннясімааудіо-операціями.

Методи:

`createBufferSource()`: створюєAudioBufferSourceNode для відтворення звукових буферів.

`createMediaElementSource()`: створюєMediaElementAudioSourceNode для використанняHTMLMediaElement як джерело звуку.

`createMediaStreamSource()`: створюєMediaStreamAudioSourceNodедлявикористанняMediaStreamякджерелозвуку.

`createGain()`: створюєGainNodедляуправліннярівнемгучності.

`createBiquadFilter()`: створюєBiquadFilterNodeдлязастосуваннячастотнихфільтрів.

`createAnalyser()`: створюєAnalyserNodeдляаналізузвуковихданих.

`createPanner()`: створюєPannerNode для створення ефекту просторового звуку.

`createConvolver()`: створюєConvolverNode для застосування ефекту реверберації.

`decodeAudioData()`: декодуєаудіодані з ArrayBuffer.

```
var audioContext = new (window.AudioContext || window.webkitAudioContext)();
```

Рис 2.20 - Приклад використанняAudioContext

AudioBuffer

AudioBufferпредставляєаудіо-дані у пам'яті. Вінвикористовується для зберігання та відтвореннязвуковихданих.

Властивості:

`length`: довжина буфера у семплах.

`duration`: тривалістьаудіоу секундах.

`sampleRate`: частота дискретизаціїаудіо.

`numberOfChannels`: кількістьканалівaудіо.

Методи:

`getChannelData(channel)`: повертає Float32Array, щоміститьдані для заданого каналу.

```
audioContext.decodeAudioData(arrayBuffer, function(buffer) {
  var audioBuffer = buffer;
});
```

Рис 2.21 -Приклад використанняAudioBuffer

AudioBufferSourceNode

AudioBufferSourceNode є джерелом звуку, щовідтворюєаудіо з AudioBuffer.

Властивості:

`buffer`: AudioBuffer, щомістить звукові дані.

`playbackRate`: швидкістьвідтворення.

`loop`: визначає, чи буде звук відтворюватися у циклі.

Методи:

`start(when, offset, duration)`: починає відтворення звуку.

`stop(when)`: зупиняєвідтворення звуку.

```
var source = audioContext.createBufferSource();
source.buffer = audioBuffer;
source.connect(audioContext.destination);
source.start();
```

Рис 2.22 - Приклад використанняAudioBufferSourceNode

GainNode

GainNodeвикористовується для управліннягучністюаудіо-сигналу.

Властивості:

`gain`: аудіо-параметр, щовизначаєрівеньпідсилення.

```
var gainNode = audioContext.createGain();
source.connect(gainNode);
gainNode.connect(audioContext.destination);
gainNode.gain.setValueAtTime(0.5, audioContext.currentTime);
```

Рис 2.23 - Приклад використанняGainNode

BiquadFilterNode

BiquadFilterNodedозволяєзастосовуватирізні типи частотнихфільтрів до аудіо-сигналу.

Властивості:

`type`: тип фільтра (lowpass, highpass, bandpass, і т.д.).

`frequency`: частота фільтрації.

`Q`: параметр якості фільтра.

`gain`: підсилення для деякихтипівфільтрів.

```
var biquadFilter = audioContext.createBiquadFilter();
biquadFilter.type = 'lowshelf';
biquadFilter.frequency.setValueAtTime(1000, audioContext.currentTime);
biquadFilter.gain.setValueAtTime(25, audioContext.currentTime);

source.connect(biquadFilter);
biquadFilter.connect(audioContext.destination);
```

Рис 2.24 - Приклад використанняBiquadFilterNode

AnalyserNode

AnalyserNodeвикористовується для аналізуаудіо-сигналу, наприклад, для створеннявізуалізацій.

Властивості:

`fftSize`: розмір FFT для аналізу частотного спектра.

`frequencyBinCount`: кількістьчастотнихбінів.

`minDecibels`: мінімальнийрівень в децибелах для аналізу.

`maxDecibels`: максимальнийрівень в децибелах для аналізу.

`smoothingTimeConstant`: коефіцієнтзгладжування для аналізу.

Методи:

`getByteTimeDomainData(array)`: отримуєданічасової області.

`getByteFrequencyData(array)`: отримуєданічастотноїобласті.

```
var analyser = audioContext.createAnalyser();
source.connect(analyser);
analyser.connect(audioContext.destination);

var dataArray = new Uint8Array(analyser.frequencyBinCount);
function draw() {
  requestAnimationFrame(draw);
  analyser.getByteFrequencyData(dataArray);
  // код для візуалізації даних
}
draw();
```

Рис 2.25 - Приклад використанняAnalyserNode

PannerNode

PannerNodeвикористовується для створенняефектупросторового звуку.

Властивості:

`positionX`, `positionY`, `positionZ`: координати розташування джерела звуку.

`orientationX`, `orientationY`, `orientationZ`: напрямокорієнтаціїджерелазвуку.

```
var panner = audioContext.createPanner();
source.connect(panner);
panner.connect(audioContext.destination);
panner.setPosition(1, 0, 0); // розташування джерела звуку праворуч від слухача
```

Рис 2.26 - Приклад використанняPannerNode

WaveShaperNode

WaveShaperNodeвикористовується для створеннянелінійнихефектів, таких як дисторшн.

Властивості:

`curve`: Float32Array, щовизначає форму вигину (кривої) для обробки сигналу.

`oversample`: метод підвищеннячастотидискретизації (none, 2x, 4x) для зменшенняартефактівобробки.

```
var waveShaper = audioContext.createWaveShaper();
waveShaper.curve = new Float32Array([0, 0.5, 1, 0.5, 0]); // Проста крива спотворення
waveShaper.oversample = '4x';

source.connect(waveShaper);
waveShaper.connect(audioContext.destination);
```

Рис 2.27 - Приклад використанняWaveShaperNode

StereoPannerNode

StereoPannerNodeвикористовується для управлінняпанорамуваннямаудіо-сигналу міжлівим і правим каналами у стереопанорамі.

Властивості:

`pan`: значенняпанорамування (від -1.0 для лівого каналу до 1.0 для правого).

```
var stereoPanner = audioContext.createStereoPanner();
stereoPanner.pan.setValueAtTime(-1, audioContext.currentTime); // Панорамування вліво

source.connect(stereoPanner);
stereoPanner.connect(audioContext.destination);
```

Рис 2.28 - Приклад використанняStereoPannerNode

ScriptProcessorNode (застарілий)

ScriptProcessorNodeвикористовується для обробкиаудіо-даних через JavaScript. ЗамістьногорекомендуєтьсявикористовуватиAudioWorklet.

Властивості:

`bufferSize`: розмір буфера для обробки даних (повинен бути ступенем двійки між 256 і 16384).

`onaudioprocess`: обробникподій для обробкиаудіо-даних.

```
var scriptNode = audioContext.createScriptProcessor(256, 1, 1);
scriptNode.onaudioprocess = function(audioProcessingEvent) {
  var inputBuffer = audioProcessingEvent.inputBuffer;
  var outputBuffer = audioProcessingEvent.outputBuffer;

  for (var channel = 0; channel < outputBuffer.numberOfChannels; channel++) {
    var inputData = inputBuffer.getChannelData(channel);
    var outputData = outputBuffer.getChannelData(channel);

    for (var sample = 0; sample < inputBuffer.length; sample++) {
      outputData[sample] = inputData[sample] * 0.5; // Приклад простого оброблення
    }
  }
};
source.connect(scriptNode);
scriptNode.connect(audioContext.destination);
```

Рис 2.29 - Приклад використанняScriptProcessorNode

AudioWorkletNode

AudioWorkletNodeвикористовується для обробкиаудіо-даних у окремомупотоці, щозабезпечуєкращупродуктивність і більшточний контроль. Властивості:

`parameters`: набіраудіо-параметрів для вузла.

`port`: MessagePort для комунікаціїміжголовним потоком і аудіо-процесором.

```
audioContext.audioWorklet.addModule('processor.js').then(() => {
  var audioWorkletNode = new AudioWorkletNode(audioContext, 'processor');
  source.connect(audioWorkletNode).connect(audioContext.destination);
});
```

Рис 2.30 - Приклад використанняAudioWorkletNode

MediaElementAudioSourceNode

MediaElementAudioSourceNode використовується для відтворення аудіо з HTMLMediaElement (наприклад, `<audio>` або `<video>`).

```
var audioElement = document.querySelector('audio');
var mediaElementSource = audioContext.createMediaElementSource(audioElement);

mediaElementSource.connect(audioContext.destination);
audioElement.play();
```

Рис 2.31 - ПрикладвикористанняMediaElementAudioSourceNode

MediaStreamAudioSourceNode

MediaStreamAudioSourceNodeвикористовується для відтворенняаудіо з MediaStream (наприклад, з мікрофона).

```
navigator.mediaDevices.getUserMedia({ audio: true })
  .then(function(stream) {
    var mediaStreamSource = audioContext.createMediaStreamSource(stream);
    mediaStreamSource.connect(audioContext.destination);
  });
```

Рис 2.32 - ПрикладвикористанняMediaStreamAudioSourceNode

MediaStreamAudioDestinationNode

MediaStreamAudioDestinationNodeвикористовується для надсиланняобробленогоаудіо до MediaStream, щоможе бути використане для записуабопередачі.

```
var mediaStreamDestination = audioContext.createMediaStreamDestination();
source.connect(mediaStreamDestination);
var stream = mediaStreamDestination.stream;
```

Рис 2.33 - ПрикладвикористанняMediaStreamAudioDestinationNode

AudioWorkletNode

AudioWorkletNodeдозволяєрозробникамстворюватисвоївласнівузлиобробки звуку, використовуючи JavaScript. Цевідкриваєвеликіможливості для створенняунікальнихзвуковихефектів та процесорів.

Властивості:

`parameters`: колекціяAudioParam, щоможуть бути використані в AudioWorkletProcessor.

```
audioContext.audioWorklet.addModule('processor.js').then(() => {
  var audioWorkletNode = new AudioWorkletNode(audioContext, 'processor');
  source.connect(audioWorkletNode).connect(audioContext.destination);
});
```

Рис 2.34 - Приклад використанняAudioWorkletNode

PeriodicWave

PeriodicWaveпредставляєперіодичнухвилю, щовикористовується для визначенняформихвилі в OscillatorNode.

Методи:

`createPeriodicWave(real, imag, constraints)`: створюєPeriodicWave з масивівдійсних і уявнихчастин.

```
var real = new Float32Array([0, 1, 0.5]);
var imag = new Float32Array([0, 0.5, 0.25]);
var wave = audioContext.createPeriodicWave(real, imag);

oscillator.setPeriodicWave(wave);
```

Рис 2.35 - ПрикладвикористанняPeriodicWave

ConstantSourceNode

ConstantSourceNodeгенеруєпостійний сигнал, щоможе бути використаний для модуляціїіншихпараметрів.

Властивості:

`offset`: AudioParam, щовизначаєзначенняпостійного сигналу.

Методи:

`start(when)`: починаєгенераціюпостійного сигналу.

`stop(when)`: зупиняєгенераціюпостійного сигналу.

```
var constantSource = audioContext.createConstantSource();
constantSource.offset.setValueAtTime(1, audioContext.currentTime);
constantSource.connect(audioContext.destination);
constantSource.start();
```

Рис 2.36 -Приклад використанняConstantSourceNode

2.3 Онлайн редактори для реалізації WebAudio API

Із зростанням популярності веб-технологій, з'являється все більше інструментів та онлайн редакторів, що допомагають розробникам ефективно використовувати WebAudioAPI. Ціінструментидозволяютьстворювати, тестувати та дебагуватиаудіо-додатки прямо у браузері. Вони часто надаютьзручнірафічніінтерфейси та підтримкурізнихфункцій Web Audio API, щозначноспрощуєпроцес розробки. У цьомурозділі ми розглянемодеякі з найбільшпопулярних онлайн редакторів та інструментів, щодопомагають у реалізації Web Audio API.

Audion

Audion є одним з найпопулярнішихінструментів для розробки з використанням Web Audio API. Церозширення для браузера Chrome, щодозволяєрозробникамбачитиаудіо-графи у реальному часі, відслідковувати звукові вузли та їх з'єднання, а також відображати інформацію про аудіо-сигнали. Audionзначнополегшуєдебаггінг та оптимізаціюаудіо-додатків.

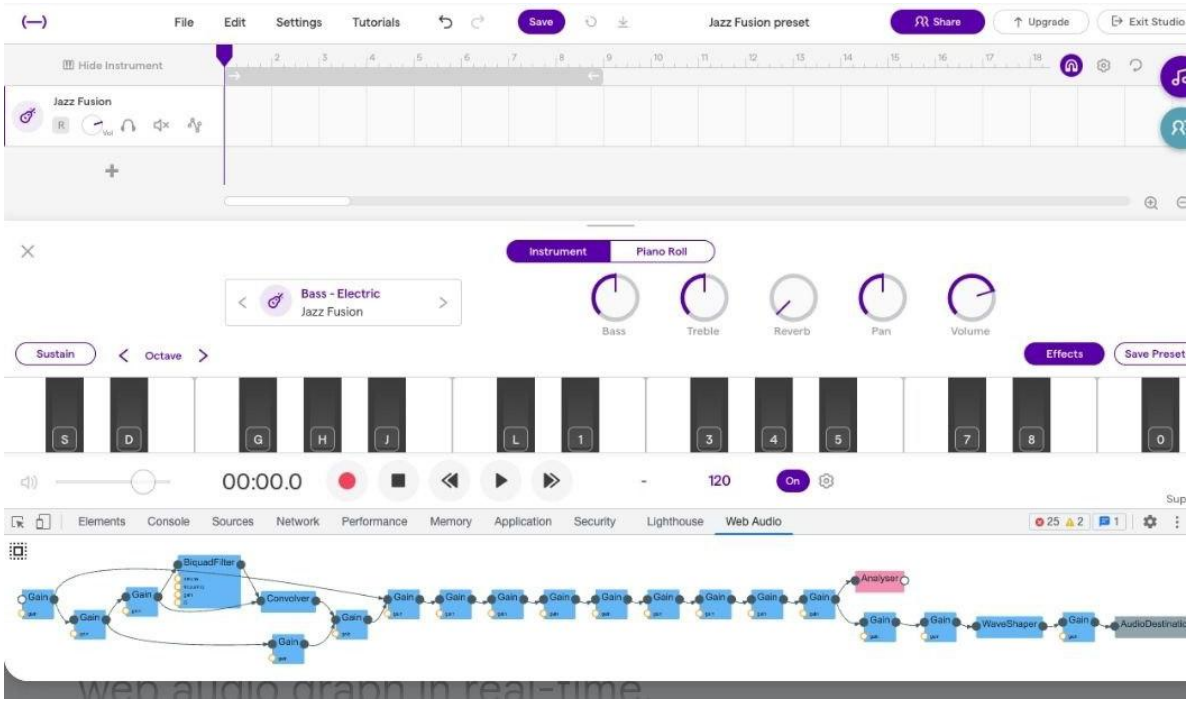


Рис 2.37

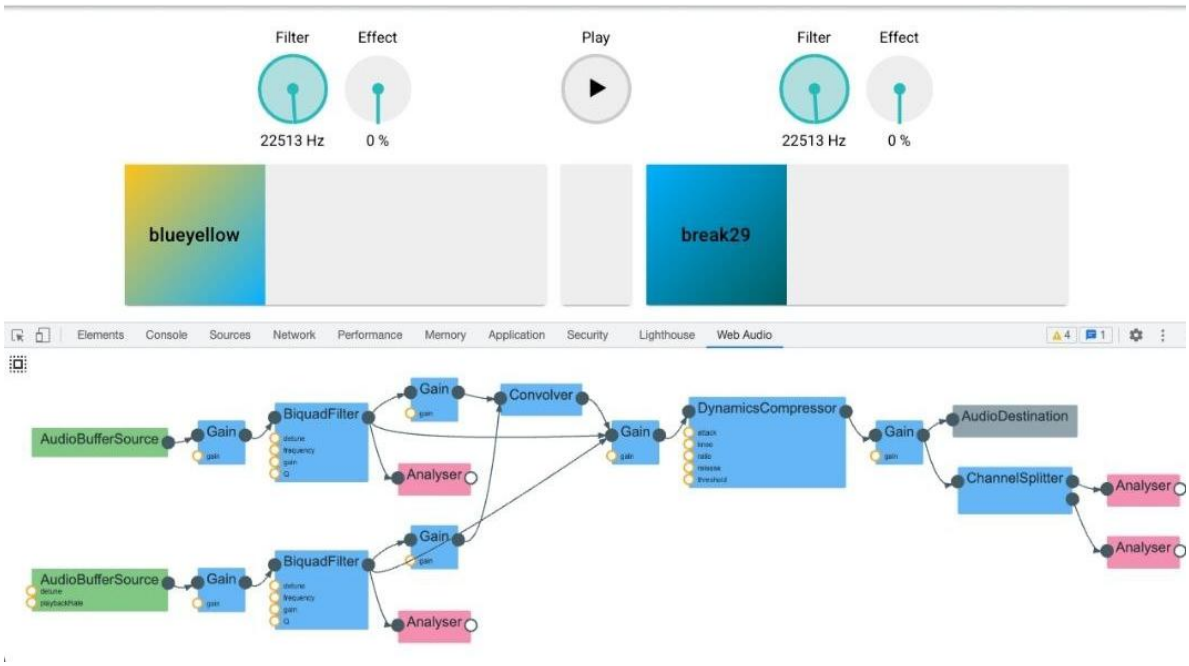


Рис 2.38

Рис 2.37, 2.38 - Інтерфейс Audion

Основні можливості Audion:

- Візуалізація аудіо-графів: Audion відображає всі вузли та з'єднання у вигляді графу, що дозволяє легко зрозуміти структуру аудіо-додатку.

- Інформація про вузли: Користувачіможуть переглядати детальну інформацію про кожен вузол, включаючи його параметри та стан.
- Моніторингаудіо-сигналів: Audiondозволяєвідстежуватиаудіо-сигнали в реальному часі, що допомагає виявляти проблеми та оптимізувати продуктивність.

ПрикладвикористанняAudion:

- ЗавантажтетавстановітьрозширенняAudion з Chrome Web Store.
- Відкрийте ваш веб-додаток, щовикористовує Web Audio API.
- ВідкрийтеAudion через меню розширень у браузері.
- Переглядайте та аналізуйте аудіо-граф вашого додатку.

Tone.js Playground

Tone.js - це популярна бібліотека для роботи з Web Audio API, яка значно спрощує створення музичних додатків та обробку звуку. Tone.js Playground - це онлайн інструмент, що дозволяє експериментувати з можливостями Tone.js прямо у браузері. Віннадаєінтерактивнесередовище для створеннязвуковихкомпозицій та ефектів за допомогою Web Audio API.

Основніможливості Tone.js Playground:

- Інтерактивнесередовище: Користувачіможутьписати та запускати код на JavaScript, використовуючиможливості Tone.js.
- Бібліотекаприкладів: Playgroundмістить велику кількістьприкладів, щодопомагають швидко освоїти основи роботи з Tone.js.
- Візуалізація аудіо: Інструмент надає можливості для візуалізації звукових сигналів та ефектів.

Приклад використання Tone.js Playground:

- Відкрийте Tone.js Playground у браузері.
- Виберіть один з прикладів або створіть новий проект.
- Напишіть код для створення аудіо ефектів та композицій.
- Запустіть код та експериментуйте зі звуком у реальному часі.

Web Audio API DevTools

Web Audio API DevTools - цещеоднерозширення для браузера Chrome, щонадаєінструменти для дебаггінгу та оптимізаціїдодатків, яківикористовують Web Audio API. Воно дозволяє переглядати структуру аудіо-графів, відстежувати стан вузлів та їх параметри, а також аналізувати продуктивність аудіо-додатку.

Основніможливості Web Audio API DevTools:

- Візуалізація аудіо-графів: Інструмент відображає всі вузли та з'єднання у вигляді графу, що дозволяє легко зрозуміти структуру аудіо-додатку.
- Інформація про вузли: Користувачі можуть переглядати детальну інформацію про кожен вузол, включаючи його параметри та стан.
- Аналіз продуктивності: Інструмент дозволяє відстежувати продуктивність аудіо-додатку та виявляти вузькі місця.

Приклад використання Web Audio API DevTools:

- Завантажте та встановітьрозширення Web Audio API DevTools з Chrome Web Store.
- Відкрийте ваш веб-додаток, щовикористовує Web Audio API.
- ВідкрийтеDevTools через меню розширень у браузері.
- Переглядайте та аналізуйте аудіо-граф вашого додатку.

Pizzicato.js Playground

Pizzicato.js - цеще одна бібліотека для роботи з Web Audio API, яка надає простий інтерфейс для створення звукових ефектів та композицій. Pizzicato.js Playground - це онлайн інструмент, що дозволяє експериментувати з можливостями Pizzicato.js прямо у браузері.

Основніможливості Pizzicato.js Playground:

- Інтерактивнесередовище: Користувачіможутьписати та запускати код на JavaScript, використовуючиможливості Pizzicato.js.
- Бібліотекаприкладів: Playgroundмістить велику кількістьприкладів, щодопомагають швидко освоїти основи роботи з Pizzicato.js.

- Простота використання: Інструмент надає зручний інтерфейс для створення аудіо ефектів та композицій.

Приклад використання Pizzicato.js Playground:

- Відкрийте Pizzicato.js Playground у браузері.
- Виберіть один з прикладів або створіть новий проект.
- Напишіть код для створення аудіо ефектів та композицій.
- Запустіть код та експериментуйте зі звуком у реальному часі.

Gibber

Gibber - цеінтерактивнесередовище для створеннямузики та аудіо-візуалізацій за допомогою Web Audio API та WebGL. Gibbernадаєпотужніінструменти для програмування звукових та візуальних ефектів, що дозволяє створювати складні мультимедійні проекти.

ОсновніможливостіGibber:

- Інтерактивнесередовище: Користувачіможутьписати код на JavaScript для створеннямузики та візуалізацій.
- Велика кількістьінструментів: Gibbernадаєбезлічінструментів для роботизі звуком, включаючисинтезатори, секвенсери та ефекти.
- ПідтримкаWebGL: Інструментдозволяєстворюватівізуалізації, синхронізованізі звуком.

ПрикладвикористанняGibber:

- ВідкрийтеGibber у браузері.
- Напишіть код для створення музики та візуалізацій.
- Запустіть код та експериментуйте з аудіо та візуальними ефектами у реальному часі.

CodePen

CodePen - цепопулярний онлайн редактор для фронтенд-розробки, щодозволяєстворювати та ділитися проектами, включаючиті, щовикористовують Web Audio API. CodePennадаєінтерактивнесередовище для написання HTML, CSS

та JavaScript коду, щодозволяєрозробникамшвидко створювати та тестувати свої аудіо-додатки.

ОсновніможливостіCodePen:

- Інтерактивне середовище: Користувачі можуть писати та запускати HTML, CSS та JavaScript код у браузері.
- Спільнота: CodePenмає велику спільнотурозробників, щодозволяє ділитися своїми проектами та отримувати зворотній зв'язок.
- Інтеграція з бібліотеками: Інструмент підтримує інтеграцію з багатьма популярними бібліотеками, включаючи Tone.js та Pizzicato.js.

Приклад використанняCodePen для Web Audio API:

- ВідкрийтеCodePen у браузері.
- СтворітьновийPen.
- Напишіть код для створенняаудіоефектів та композицій за допомогою Web Audio API.
- Запустіть код та експериментуйте зі звуком у реальному часі.

3 РОЗРОБКА ПРОЄКТУ

3.1 Короткі теоретичні відомості

WebAudio API - один із засобів, що значно розширює можливості Web - додатків при роботі зі звуком. Це найпотужніший інструмент, без якого складно буде обійтися в майбутньому при розробці сучасних ігор та інтерактивних веб-додатків.

Елементи <audio> і webAudio API практично ніяк не пов'язані між собою. Це два незалежних, самодостатніх API, призначених для вирішення різних завдань. Єдиний зв'язок між ними полягає в тому, що <audio> елемент може бути одним із джерел звуку для WebAudio API.

Завдання, які покликаний вирішувати елемент <audio>:

• Простий аудіо плеєр

• однопоточне фонове аудіо.

• Аудіо підказки, капчи і т.п.

Завдання, які покликаний вирішувати WebAudio API:

• Об'ємний звук для ігор та інтерактивних веб додатків

• Програми для обробки звуку

• Аудіо синтез

• Візуалізація аудіо і багато, багато, багато іншого.

Одним з основоположних понять при роботі з WebAudio API є аудіо контекст.

Поки специфікація знаходиться в чернетці, в webkit браузерах потрібно використовувати webkitAudioContext. Тобто щось на зразок:

```
var context;
window.addEventListener('load', function() {
    try {
        window.AudioContext = window.AudioContext || window.webkitAudioContext;
        context = new AudioContext();
    } catch(e) {
        alert('Oops.. Your browser does not support the audio API');
    }
}, false);
```

Рис 3.1 – Використання webkitAudioContext

У одного документа може бути тільки один контекст. Цього цілком достатньо для всього спектра завдань вирішуваного WebAudio API. Наявність одного аудіо контексту дозволяє будувати скільки завгодно складні аудіо графи з необмеженою кількістю джерел і одержувачів звукового сигналу. Практично всі методи і конструктори для створення аудіо модулів є методами аудіо контексту.

Можливі джерела звукового сигналу:

1. AudioBufferSourceNode - аудіо буфер (розглянемо нижче)

2. MediaElementAudioSourceNode - <audio> або <video> елемент

3. MediaStreamAudioSourceNode - зовнішній аудіо потік (стрім) (мікрофон або будь-який інший аудіо стрім, в тому числі зовнішній)

Можливі одержувачі звукового сигналу:

1. context.destination - системний звуковий вихід за замовчуванням (в типовому випадку - колонки).

2. MediaStreamAudioDestinationNode - аудіо потік (стрім). Цей потік може бути використаний таким же чином, як потік, отриманий через getUserMedia(), і, наприклад, може бути відправлений на віддалений RTCPeerConnection за допомогою методу addStream().

Творці WebAudio API зробили побудову будь-яких графів (схем) витонченим і простим для розуміння. У кожного модуля є метод.connect(...), який приймає один параметр, власне говорить про те, до чого потрібно під'єднатися. Ось все, що потрібно написати для побудови вищезгаданої схеми:

```
source1.connect(node1);
source2.connect(node3);
node1.connect(node4);
node1.connect(node2);
node2.connect(destination);
node3.connect(node1);
node4.connect(destination);
node4.connect(node3);
```

Рис 3.2 – Приклад застосування .connect

WebAudio API містить десятки високорівневих, конфігурованих і готових до використання модулів. Це підсилювачі, лінії затримки, фільтри, модулі згортки, сплітери і мержери каналів, 3D панери і т.д. Ви можете створювати складні графи обробки і синтезу звуку, просто поєднуючи готові блоки і конфігуруючи їх.

3.2 Порядок виконання роботи

1. Спочатку створюємо каркас для файлу .html.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Web Audio with 2D Visualization</title>
  <style>
  </style>
</head>
<body>
  <script>
  </script>
</body>
</html>
```

Рис 3.3 - Каркас для файлу .html

2. Тепер будуємо граф для розуміння роботи API. WebAudio API матиме такі функціональні можливості: вибір типу хвилі (Sine - синусоїдальна, Tringle - трикутна, Square - квадратна), встановлення частоти від 1 до 20 000 Гц, регулювання гучності та візуалізація частотного спектру (2d Visualization).

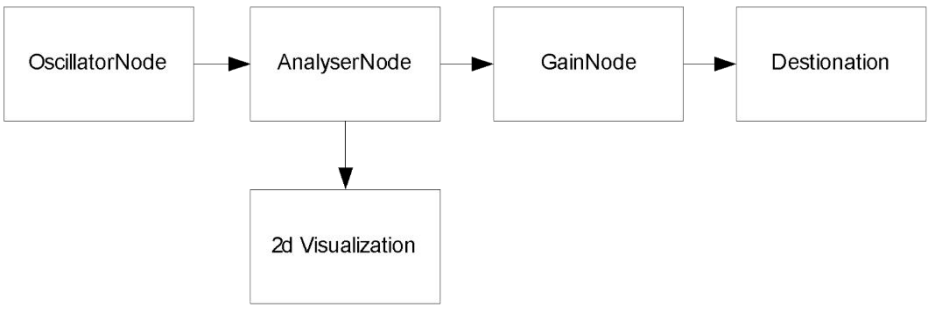


Рис. 3.4- Побудований граф

3. Додаємо таблицю стилів.

```
<style>
  body {
    margin: 0;
    overflow: hidden;
  }
  label {
    margin-right: 10px;
  }
  canvas {
    margin-top: 20px;
    width: 100%;
    height: 100px;
  }
</style>
```

Рис 3.5 – Таблиця стилів

4. Робимо оформлення сторінки: додаємо список типу хвилі, назви слайдерів, самі слайдери та canvas.

```
<body>
  <select id="waveType">
    <option value="sine">Sine</option>
    <option value="square">Square</option>
    <option value="triangle">Triangle</option>
  </select>
  <label for="frequency">Frequency</label>
  <input type="number" id="frequency" value="440" min="1" max="20000"> (Hz)
  <br><br>
  <label for="volume">Volume</label>
  <input type="range" id="volume" value="0.5" min="0" max="1" step="0.01">
  <p>2D visualization</p>
  <canvas id="canvas"></canvas>
  <script>
```

Рис 3.6 – Оформлення сторінки

На рис. 3.7 зображено отримане оформлення сторінки.

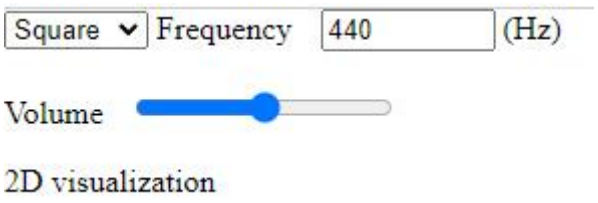


Рис. 3.7

5. Заповнюємо скрипт. Створюємо новий об’єкт AudioContext, який представляє граф обробки аудіо, побудований із зв’язаних аудіомодулів.

```
const audioCtx = new AudioContext();
```

Рис 3.8 – Створення AudioContext

Створюємо Analyser, який надає інформацію про аналіз частоти та часової області в реальному часі.

```
const analyser = audioCtx.createAnalyser();
```

Рис 3.9 – Створення Analyser

Встановлюємо розмір FFT (швидке перетворення Фур’є) для аналізатора. Це визначає розмір масиву даних, який використовується для частотного аналізу.

```
analyser.fftSize = 256;
```

Рис 3.9 - розмір FFT

Створюємо новий масив із довжиною, що дорівнює властивості frequencyBinCount аналізатора. Це буде використано для зберігання даних частоти.

```
const frequencyData = new Uint8Array(analyser.frequencyBinCount);
```

Рис 3.10 – створення нового масиву

Робимо функцію для створення вузла осцилятора з заданим типом сигналу та частотою.

```
function createOscillator(waveType, frequency) {
  const oscillator = audioCtx.createOscillator();
  oscillator.type = waveType;
  oscillator.frequency.setValueAtTime(frequency, audioCtx.currentTime);
  oscillator.connect(analyser);
  return oscillator;
}
```

Рис 3.11- Створення вузла осцилятора

Робимо функцію для підключення вузла осцилятора до місця призначення аудіоконтексту з додатковим керуванням гучністю.

```
function connectAudio(oscillator) {
  const gainNode = audioCtx.createGain();
  gainNode.gain.setValueAtTime(document.getElementById("volume").value, audioCtx.currentTime);
  analyser.connect(gainNode);
  gainNode.connect(audioCtx.destination);
}
```

Рис 3.12 - функцію для підключення вузла осцилятора

Робимо функцію для створення та відтворення осцилятора на основі введення користувача.

```
function playSound() {
  const waveType = document.getElementById("waveType").value;
  const frequency = parseFloat(document.getElementById("frequency").value);
  const oscillator = createOscillator(waveType, frequency);
  oscillator.start();
  connectAudio(oscillator);
}
```

Рис 3.13 - функцію для створення та відтворення осцилятора

Додаємо обробник подій для waveType, частоти, гучності.

```
document.getElementById("waveType").addEventListener("change", playSound);
document.getElementById("frequency").addEventListener("change", playSound);
document.getElementById("volume").addEventListener("input", function() {
  const gainNode = audioCtx.createGain();
  gainNode.gain.setValueAtTime(this.value, audioCtx.currentTime);
  analyser.disconnect();
  analyser.connect(gainNode);
  gainNode.connect(audioCtx.destination);
});
```

Рис 3.14 - обробник подій для waveType

Отримуємо посилання на елемент canvas та контекст 2D візуалізації для полотна.

```
const canvas = document.getElementById("canvas");
const ctx = canvas.getContext("2d");
```

Рис 3.15 - Посилання на елемент canvas та контекст 2D візуалізації

Робимо функцію для анімації візуалізації на основі аудіоданих та викликаємо її.

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

  analyser.getByteFrequencyData(frequencyData);

  const barWidth = canvas.width / frequencyData.length;
  let x = 0;

  ctx.clearRect(0, 0, canvas.width, canvas.height);
  for (let i = 0; i < frequencyData.length; i++) {
    const volume = document.getElementById("volume").value;
    const barHeight = frequencyData[i] * volume * 2;
    ctx.fillStyle = `rgb(${barHeight}, ${barHeight}, ${barHeight})`;
    ctx.fillRect(x, canvas.height - barHeight, barWidth, barHeight);
    x += barWidth;
  }
}
animate();
```

Рис 3.16 - Функція для анімації візуалізації

Запустимо сторінку і отримуємо такий результат як на рис. 3.17

Рис. 3.17

Повний код вказаний у додатку А.

В якості прикладу реалізовано онлайн аудіо-редактор, що використовує WebAudio API для створення, обробки та візуалізації звукових ефектів.

Код демонструє можливості WebAudio API для генерування різних типів звукових хвиль (синусоїдальна, квадратна, трикутна), а також для регулювання частоти та гучності звуку.

3.3 Створення більш складних додатків на прикладі онлайн аудіо-редактора

Створення онлайн аудіо-редактора за допомогою WebAudio API є складним завданням, яке включає в себе багато етапів, від початкового проектування до реалізації та тестування. WebAudio API – це потужний інструмент для обробки та маніпулювання звуком у веб-браузері. Далі наведені покрокові дії, які необхідно виконати для створення функціонального аудіо-редактора.

Крок 1: Визначення Вимог і Планування

Перед тим, як приступити до кодування, необхідно чітко визначити вимоги до вашого аудіо-редактора:

1. **Функціональність**: Які функції повинен виконувати редактор? Наприклад:
    - Завантаження аудіофайлів
    - Відтворення і пауза
    - Обрізка і нарізка аудіо
    - Зміна гучності
    - Додавання ефектів (наприклад, реверберація, фільтри)
    - Збереження відредагованих файлів
2. **Інтерфейс користувача**: Який буде дизайн інтерфейсу? Чи буде він зручним та інтуїтивно зрозумілим для користувача?

3. **Платформа**: На яких платформах буде працювати ваш редактор? (ПК, мобільні пристрої тощо)

Крок 2: Налаштування Середовища Розробки

Після визначення вимог потрібно підготувати середовище розробки:

1. **Інструменти**: Встановіть необхідні інструменти для розробки (редактор коду, веб-браузер для тестування, система контролю версій тощо).
2. **Бібліотеки і фреймворки**: Виберіть відповідні бібліотеки і фреймворки, які допоможуть у розробці (наприклад, React для інтерфейсу користувача).

Крок 3: Створення Базового Інтерфейсу Користувача

Створіть базову HTML-структуру для вашого аудіо-редактора. Основні елементи включають:

1. **Форма для завантаження файлів**: Дозволяє користувачам завантажувати аудіофайли.
2. **Поле для відображення звукової хвилі**: Використовуйте HTML5 <canvas> для візуалізації звукової хвилі.
3. **Кнопки керування**: Кнопки для відтворення, паузи, обрізки, додавання ефектів тощо.

Крок 4: Завантаження і Відтворення Аудіо

Використовуйте JavaScript і WebAudio API для завантаження та відтворення аудіофайлів:

1. **Обробка завантаження файлів**: Реалізуйте функцію, яка буде обробляти завантаження аудіофайлів за допомогою FileReader.
2. **Ініціалізація аудіо контексту**: Створіть новий аудіо контекст за допомогою AudioContext.
3. **Відтворення аудіо**: Реалізуйте функції для відтворення і паузи аудіо.

Крок 5: Візуалізація Аудіо

Створіть функцію для візуалізації аудіо хвилі:

1. **Отримання даних аудіо буферу**: Використовуйте методи WebAudio API для отримання аудіо даних з буфера.
2. **Малювання звукової хвилі**: Використовуйте Canvas API для малювання звукової хвилі на <canvas>.

Крок 6: Реалізація Функцій Редагування

Додайте можливості редагування аудіо:

1. **Обрізка аудіо**: Створіть функцію для обрізки аудіо, яка буде видаляти непотрібні частини аудіофайлу.
2. **Зміна гучності**: Додайте можливість змінювати гучність аудіо за допомогою GainNode.
3. **Додавання ефектів**: Використовуйте різні вузли WebAudio API, такі як BiquadFilterNode для фільтрації або ConvolverNode для реверберації.

Крок 7: Тестування

Після реалізації основних функцій ретельно протестуйте ваш аудіо-редактор:

1. **Тестування функціональності**: Переконайтеся, що всі функції працюють коректно і без помилок.
2. **Крос-браузерне тестування**: Перевірте роботу редактора у різних браузерах і на різних пристроях.
3. **Користувацьке тестування**: Отримайте відгуки від реальних користувачів і внесіть необхідні покращення.

Крок 8: Оптимізація і Деплоймент

Після завершення тестування оптимізуйте код і підготуйте його до розгортання:

1. **Оптимізація продуктивності**: Зменшіть розмір файлів, використовуйте кешування, мінімізуйте JavaScript та CSS.
2. **Розгортання**: Виберіть хостинг для вашого веб-додатка і розгорніть його.

3.4 Висновки до третього розділу

Створений код у пункті 3.2 демонструє можливості WebAudio API для створення звукових ефектів та візуалізації аудіо. Він дозволяє користувачу вибирати тип хвилі (синусоїдальна, квадратна або трикутна), задавати частоту звуку (від 1 до 20000 Гц) і регулювати гучність (від 0 до 1 з кроком 0.01.), що створює можливість експериментувати зі звуковими параметрами.

Код використовує WebAudio API для: створення генератора звуку (oscillator) з обраною формою хвилі та частотою; підключення генератора звуку до

аналізатора (analyser) для отримання даних про частоту; візуалізації частотного спектру звуку у вигляді 2D-графіку на канвасі.

У пункті 3.3 було розглянуто можливості для створення більш складних додатків з використанням WebAudioAPI на прикладі створення онлайн аудіо-редактора.

ВИСНОВКИ

У роботі описано та проаналізовно розвиток та створення додатків для роботи із аудіо контентом. Розглянуті методи котрими можна створити додаток для роботи з аудіо контентом. Описані перспективи розвитку цих технологій.

У першому розділі описана теоретична частина роботи. Розглянуті як фізичні параметри звуку так і методи взаємодії із ним. Розписані переваги використання цифрової обробки звуку та цифрові аудіо формати.

У другому розділі роботи детально розглянуто HTML5 та WebAudio API як інструменти для створення та обробки аудіо контенту.

Проведено аналіз можливостей HTML5 для роботи з аудіо файлами без необхідності використання сторонніх плагінів, що забезпечує вищий рівень безпеки та зручності для користувачів.

У третьому розділі створено онлайн аудіо-редактор, що використовує WebAudio API для створення, обробки та візуалізації звукових ефектів.

Код демонструє можливості WebAudio API для генерування різних типів звукових хвиль (синусоїдальна, квадратна, трикутна), а також для регулювання частоти та гучності звуку.

Нижче у тому ж розділі було описано як можна створити більш складний додаток для розкриття більшого потенціалу технологій HTML5 та WebAudioAPI, проте за браком часу, можливостей та потреби - в самій роботі описано виключно те як можна створити подібний додаток на прикладі онлайн аудіо-редактора.

Отримані результати можуть бути використані для подальшого вдосконалення методів створення та обробки аудіо контенту, що сприятиме підвищенню якості та доступності цифрових аудіо сервісів.

Ці висновки узагальнюють проведену роботу та вказують на перспективні напрями подальших досліджень і розробок у сфері звукового контенту в інформаційному середовищі .

ПЕРЕЛІК ПОСИЛАНЬ

1. Макаренко, К.О. Трапезон, А.М. Чермянін. Основні вимоги до оформлення атестаційних робіт, дипломних та курсових проектів: методичні рекомендації для студентів усіх форм навчання факультету електроніки. – К.: ФЕЛ НТУУ “КПІ”, 2006. – 112 с.
2. Wikipedia:Звук: https://uk.wikipedia.org/wiki/%D0%97%D0%B2%D1%83%D0%BA
3. IBM: Whatiscloudcomputing?. [Електронний ресурс] – 2024. – Режим доступу до журн.: https://www.ibm.com/topics/cloud-computing?mhsrc=ibmsearch_a&mhq=cloud%20computing
4. CloudComputing: Prof. Dr. O. Günther, Prof. Dr. W. Karl, Prof. Dr. R. Lienhart, Prof. Dr. K. Zeppenfeld. Springer,2010– 134c.
5. Аналоговий та цифровийзвукозапис. Порівняннятехнологій: https://zs-studio.com.ua/analogoviy-ta-cifroviy-zvukozapis/
6. HTML5 Basics For Everyone Tired Of Reading About Deprecated Code: https://html.com/html5/
7. Популярні ефекти обробки звуку: https://muzline.ua/uk/statti-ta-oglyadi/populyarni-efekti-obrobki-zvuku/
8. Microsoft Silverlight: https://uk.wikipedia.org/wiki/Microsoft_Silverlight
9. WebAudio API samples: https://webaudioapi.com/samples/
10. О.П. Гребінь, В.Б. Швайченко, Н.Ф. Левенець.Основи звукотехніки– К.: ФЕЛ НТУУ “КПІ”, 2023. – 342с.

ДОДАТОК А

Повний код практичної частини

```
<!DOCTYPE html>
<htmllang="en">
<head>
<metacharset="UTF-8">
<metaname="viewport" content="width=device-width, initial-scale=1.0">
<title>WebAudiowith 2D Visualization</title>
<style>
body {
margin: 0;
overflow: hidden;
    }
       label {
margin-right: 10px;
    }
canvas {
       margin-top: 20px;
width: 100%;
height: 100px;
    }
</style>
</head>
<body>
<selectid="waveType">
<optionvalue="sine">Sine</option>
<optionvalue="square">Square</option>
<optionvalue="triangle">Triangle</option>
</select>
<labelfor="frequency">Frequency</label>
<inputtype="number" id="frequency" value="440" min="1" max="20000"> (Hz)
<br><br>
<labelfor="volume">Volume</label>
<inputtype="range" id="volume" value="0.5" min="0" max="1" step="0.01">
<p>2D visualization</p>
<canvasid="canvas"></canvas>
<script>
constaudioCtx = newAudioContext();
constanalyser = audioCtx.createAnalyser();
analyser.fftSize = 256;
constfrequencyData = new Uint8Array(analyser.frequencyBinCount);

functioncreateOscillator(waveType, frequency) {
constoscillator = audioCtx.createOscillator();
oscillator.type = waveType;
oscillator.frequency.setValueAtTime(frequency, audioCtx.currentTime);
oscillator.connect(analyser);
returnoscillator;
    }

functionconnectAudio(oscillator) {
constgainNode = audioCtx.createGain();
      gainNode.gain.setValueAtTime(document.getElementById("volume").value,
audioCtx.currentTime);
analyser.connect(gainNode);
gainNode.connect(audioCtx.destination);
```

```
    }

functionplaySound() {
constwaveType = document.getElementById("waveType").value;
constfrequency = parseFloat(document.getElementById("frequency").value);
constoscillator = createOscillator(waveType, frequency);
oscillator.start();
connectAudio(oscillator);
    }

document.getElementById("waveType").addEventListener("change", playSound);
document.getElementById("frequency").addEventListener("change", playSound);
document.getElementById("volume").addEventListener("input", function() {
constgainNode = audioCtx.createGain();
gainNode.gain.setValueAtTime(this.value, audioCtx.currentTime);
analyser.disconnect();
analyser.connect(gainNode);
gainNode.connect(audioCtx.destination);
    });

constcanvas = document.getElementById("canvas");
constctx = canvas.getContext("2d");

functionanimate() {
requestAnimationFrame(animate);

analyser.getByteFrequencyData(frequencyData);

constbarWidth = canvas.width / frequencyData.length;
let x = 0;

ctx.clearRect(0, 0, canvas.width, canvas.height);
for (let i = 0; i <frequencyData.length; i++) {
constvolume = document.getElementById("volume").value;
constbarHeight = frequencyData[i] * volume * 2;
ctx.fillStyle = `rgb(${barHeight}, ${barHeight}, ${barHeight})`;
ctx.fillRect(x, canvas.height - barHeight, barWidth, barHeight);
        x += barWidth;
      }
    }
animate();
</script>
</body>
</html>
```

SUMMARY

Thepaperdescribesandanalyzesthedevelopmentandcreationofapplicationsforworking withaudiocontent.
Themethodsofcreatinganapplicationforworkingwithaudiocontentareconsidered.
Theprospectsforthedevelopmentofthesetechnologiesaredescribed.

Thefirstsectiondescribesthetheoreticalpartofthework.
Bothphysicalparametersofsoundandmethodsofinteractionwithitareconsidered.
Theadvantagesofusingdigitalsoundprocessinganddigitalaudioformatsaredescribed.

Thesecondsectionofthepaperdescribesindetail HTML5 andWebAudio API astoolsforcreatingandprocessingaudiocontent.

Weanalyzethecapabilitiesof HTML5 forworkingwithaudiofileswithouttheneedtousethird-partyplug-ins, whichprovides a higherlevelofsecurityandconvenienceforusers.

Inthethirdsection, wecreatedanonlineaudioeditorthatusestheWebAudio API tocreate, process, andvisualizesoundeffects.

ThecodedemonstratesthecapabilitiesoftheWebAudio API togeneratedifferenttypesofsoundwaves (sine, square, triangular), aswellastoadjustthefrequencyandvolumeofthesound.

Below, inthesamesection, wedescribehowtocreate a morecomplexapplicationtounlockthegreaterpotentialof HTML5 andWebAudio API technologies, butduetothelackoftime, opportunities, andtheneedfortheworkitself, wedescribeonlyhowtocreate a similarapplicationusingtheexampleofanonlineaudioeditor.

Theresultsobtainedcanbeusedtofurtherimprovethemethodsofcreatingandprocessinga udiocontent, whichwillhelpimprovethequalityandaccessibilityofdigitalaudioservices.

Theseconclusionssummarizetheworkdoneandpointtopromisingareasforfurtherresearc handdevelopmentinthefieldofaudiocontentintheinformationenvironment.