Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра інженерії програмного забезпечення

Сичков Віталій Сергійович,
студент групи АС-172

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Програмна система паралельного цифрового оброблення звукових сигналів

Спеціальність:
121 – Інженерія програмного забезпечення

Освітньо-професійна програма:
Інженерія програмного забезпечення

Керівник:
Тройніна Анастасія Сергіївна,
канд. техн. наук, доцент

Одеса – 2022

ЗМІСТ

Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Навчально-науковий інститут комп'ютерних систем
Кафедра інженерії програмного забезпечення

Рівень вищої освіти: другий (магістерський)
Спеціальність: 121 – Інженерія програмного забезпечення
Освітньо-професійна програма: Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Комлева Н. О.
«___» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Сичкова Віталія Сергійовича, група АС-172

1. Тема роботи: «Програмна система паралельного цифрового оброблення звукових сигналів»
Керівник роботи: Тройніна Анастасія Сергіївна, канд. техн. наук, доцент
затверджені наказом ректора від «20» жовтня 2022 р. № 399-в.
2. Зміст роботи: опис потреб користувачів та проблем наявних засобів оброблення звукових сигналів для операційної системи Android, опис запропонованого методу паралельного оброблення, специфікація вимог до програмної системи, проєктування системи, програмна реалізація системи, опис результатів дослідження властивостей програмної розробки.
3. Перелік ілюстративного матеріалу: згідно з презентацією: актуальність розробки (слайди 2-3), мета роботи (слайд 4), задачі роботи (слайд 5), аналіз аналогічних програмних засобів (слайд 6), порівняльна характеристика продуктів-аналогів (слайд 7), традиційний послідовний метод оброблення звукових сигналів (слайд 8),

запропонований паралельний метод оброблення звукових сигналів (слайд 9), оцінка кількості виконуваних операцій накладання ефектів на звукові доріжки (слайд 10), діаграма варіантів використання (слайд 11), архітектура програмної системи (слайд 12), діаграма програмних класів (слайд 13), ER-діаграма сховища даних багатодоріжкових проєктів (слайд 14), використовувані інструменти розроблення (слайд 15), приклад використання системи (слайд 16), визначення характеристик програмної системи (слайд 17).

4. Консультанти розділів роботи

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

5. Дата видачі завдання: «__30__» __серпня__ 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назви етапів кваліфікаційної роботи	Строк виконання	Примітка
1	Вибір теми кваліфікаційної роботи магістра	29.08.2022	виконано
2	Складання плану роботи та погодження його з керівником	30.08.2022	виконано
3	Аналіз предметної області, проблем користувачів, аналогічних систем	12.09.2022	виконано
4	Розроблення методу паралельного оброблення звукових сигналів	26.09.2022	виконано
4	Специфікація вимог до програмної системи	10.10.2022	виконано
5	Проєктування архітектури, класів, сховища даних, GUI	24.10.2022	виконано
6	Програмна реалізація системи оброблення звукових сигналів	19.11.2022	виконано
7	Визначення властивостей програмної розробки	26.11.2022	виконано
8	Написання докладу і оформлення презентації для захисту	04.12.2022	виконано

Здобувач вищої освіти _____ В. С. Сичков

Керівник роботи _____ А. С. Тройніна

АНОТАЦІЯ

Метою роботи є скорочення кількості виконуваних користувачем операцій для накладання кінцевих ефектів оброблення сигналів за рахунок розроблення ефективного паралельного методу та його застосування під час етапу зведення звукових доріжок.

Технологіями розроблення є об’єктно-орієнтована мова програмування Java, засоби JDK 8 оновлення 351 і Android SDK 33.0.3, розширювана мова розмічення XML версії 1.0 та бібліотеки Android Architecture Components.

Як результат роботи виконана реалізація програмного забезпечення “Audiora”, яке дозволить користувачам працювати з аудіофайлами розповсюджених форматів, створювати багатодоріжкові проєкти та паралельно накладати різні ефекти оброблення сигналів на доріжки для операційної системи (ОС) Android.

Ключові слова: програмна система, звуковий сигнал, ефект оброблення, паралельне оброблення, Java, Android, XML.

ABSTRACT

The purpose of the work is to reduce the number of operations performed by user for layering final effects of signal processing due to the development of an effective parallel method and its application at the stage of mixing audio tracks.

An object-oriented programming language Java, JDK 8 Update 351, Android SDK 33.0.3, eXtensible Markup Language (XML) version 1.0 and Android Architecture Components libraries serve as technologies of development.

The result of the work is a developed software “Audiora”, which allows users to work with audio files of common encoding formats, create multitrack projects and apply various signal processing effects to tracks in parallel for Android operating system (OS).

Keywords: software system, audio signal, processing effect, parallel processing, Java, Android, XML.

ВСТУП

Оброблення звукового сигналу – це процес змінення певних характеристик аудіосигналу (як-от амплітудна, частотна та фазова), застосування компресії до нього або розширення його динаміки, використання різних типів модуляції, накладання ефектів реверберації, відлуння, його затримування тощо. Сучасне програмне забезпечення (ПЗ) дозволяє здійснювати перетворення таких сигналів, до того ж будь-якої складності, та вживати надзвичайні ефекти оброблення. Останнє можна здійснювати як до вже записаної фонограми, так і в реальному часі.

Необхідність у використанні засобів записування і оброблення сигналів на мобільних пристроях пояснюється виникненням різних життєвих ситуацій: треба записати лекцію в університеті, певну аудіонотатку, зробити пояснювальні коментарі до відеоролика або записати і обробити озвучення/дублювання чи вокальні партії. Також не завжди в кожного є змога працювати з десктоп-версіями такого ПЗ, а наявні аналогічні мобільні рішення не дозволяють повноцінно реалізувати власні художні задуми через відсутність певних інструментів для аналізу та ручної корекції звукових сигналів, підтримки всіх поширених ефектів оброблення та їх швидкого і гнучкого налаштування.

Метою роботи є скорочення кількості виконуваних користувачем операцій для накладання кінцевих ефектів оброблення сигналів за рахунок розроблення ефективного паралельного методу та його застосування під час етапу зведення звукових доріжок.

Для досягнення цієї мети необхідно розв’язати такі задачі:

а) вивчити проблеми користувачів та можливості наявних застосунків для оброблення звукових сигналів;

б) виявити та проаналізувати вимоги до розроблюваної програмної системи;

в) дослідити наявні підходи до розв’язання поставленої задачі та розробити власний;

г) розробити структуру для зберігання даних багатодоріжкових проєктів;

ґ) розробити алгоритми для виконання основних функцій системи;

д) виконати програмну реалізацію інтерфейсу користувача;

е) визначити та експериментально оцінити певні характеристики програмної розробки, узгоджені з метою роботи.

Використані методи наукового дослідження: аналіз проблем мобільних застосунків оброблення звукових сигналів, порівняння характеристик програмних аналогів, вимірювання властивостей розробленої програмної системи, експериментальне оцінювання результатів оброблення звукових сигналів.

Наукова новизна отриманих результатів полягає в такому: вперше запропоновано метод паралельного оброблення звукових сигналів на мобільних пристроях, який, над відміну від традиційного послідовного підходу, де ефекти оброблення поступово накладаються безпосередньо на вихідний сигнал один за одним, зберігає останній та водночас застосовує за одну операцію оброблення одразу ланцюжок ефектів до оперативної копії цього сигналу, що дозволяє зменшити кількість виконуваних операцій застосування результівних ефектів оброблення звукових сигналів.

Практичне значення отриманих результатів полягає у створенні програмної системи паралельного цифрового оброблення звукових сигналів, яка дозволить користувачам (це можуть бути студенти, музиканти, виконавці, композитори, аранжувальники, звукорежисери, музичні продюсери) працювати з аудіофайлами розповсюджених форматів, створювати багатодоріжкові проєкти, накладати різні ефекти оброблення сигналів на доріжки, використовуючи запропонований паралельний метод, підтримка якого є унікальною функціональною особливістю серед аналогічних продуктів.

Результати роботи було опубліковано у вигляді тез у матеріалах Дванадцятої міжнародної наукової конференції студентів та молодих вчених «Сучасні інформаційні технології»/“Modern Information Technology” (MIT-2022, див. додаток Б) [1].

У першому розділі «Проблеми наявних рішень для оброблення звукових сигналів» було проаналізовано потреби користувачів, використовуючи підхід дизайн-мислення, найпоширеніші програмні аналоги, наведено порівняльну характеристику останніх із зазначенням їх проблем, які покликана розв’язати розроблена програмна система (ПС), а також сформульовано постановку задачі дослідження.

У другому розділі «Розроблення ефективного методу паралельного оброблення звукових сигналів» було розглянуто традиційний метод оброблення та запропоновано власний підхід, а також наведено порівняльну оцінку кількості виконуваних операцій накладання ефектів на звукові доріжки за використання кожного з методів.

У третьому розділі «Специфікація вимог до програмної системи» було сформульовано технічне завдання на розроблення програмної системи як визначення функціональних (за допомогою UML-діаграми варіантів використання та відповідних сценаріїв) та нефункціональних вимог до неї.

У четвертому розділі «Проєктування програмної системи» було вибрано архітектуру розроблюваної системи, надано структуру й організацію програмних класів та сховища даних багатодоріжкових проєктів, описано застосовані шаблони проєктування, а також графічний інтерфейс користувача.

У п’ятому розділі «Програмна реалізація системи» було вибрано необхідні технології розроблення, описано деякі особливості реалізації, наведено один зі сценаріїв тестування отриманої програмної розробки та приклад її використання.

У шостому розділі «Визначення властивостей програмної системи» було описано результати дослідження властивостей програмної розробки. Спочатку було сформовано гіпотезу, які слід було перевірити, та визначено метрики, які були для цього застосовані. Далі було описано методику виконання досліджень характеристик системи, надано отримані результати та зроблено висновки щодо поставленої гіпотези.

1 ПРОБЛЕМИ НАЯВНИХ РІШЕНЬ ДЛЯ ОБРОБЛЕННЯ ЗВУКОВИХ СИГНАЛІВ

1.1 Аналіз предметної області та проблем користувачів щодо оброблення звукових сигналів

Під обробленням звукового сигналу розуміють процес змінення певних характеристик аудіосигналу (як-от амплітудна, частотна та фазова), застосування компресії до нього або розширення його динаміки, використання різних типів модуляції, накладання ефектів реверберації, його затримування тощо. Під час оброблення звукорежисер може виконувати як виключно технічні, так і художні завдання. До перших можна віднести погодження параметрів аудіосигналу з характеристиками обладнання. До других – застосування різних ефектів оброблення звуку, наприклад, деякі з них: тремоло, вібрато, хорус, відлуння.

Нині таке оброблення проводять здебільшого в цифровому вигляді за допомогою звукових процесорів. Колись студії, звукові та радіо, розміщували на кількох десятках кв. метрів… Однак, зараз хороший комп’ютер може витіснити їх, бо за можливостями перевищує десяток таких студій разом узятих, а за вартістю є набагато дешевшим навіть за одну студію. Завдяки цьому звукозапис став більш доступним як професіоналам, так і звичайним полюблювачам.

За допомогою сучасних програмних систем можна змінювати сигнали будь-якої складності та вживати до них надзвичайні ефекти оброблення. До речі, якщо розглядати аналоговий варіант, то практично для кожного окремого ефекту використовують свій пристрій, який може дуже дорого коштувати. У цифровому ж якість оброблення аудіосигналів набагато менше залежить від якості обладнання. Водночас не потрібно змінювати останнє, достатньо підібрати лише необхідне ПЗ.

Цифрове оброблення виконують із використанням програмних і апаратних засобів, зазвичай за допомогою спеціальних програм-редакторів та звукових карток різного призначення. Також його можна застосовувати до вже записаної фонограми

або до вхідного сигналу в режимі реального часу. Перше здійснюють на етапі «мастерінгу» або підготовки фонограм до тиражування, коли більш важливим є детальне опрацювання всіх тонкощів звучання результівного аудіосигналу, а не швидкість процесу.

Як зазначено в [2], «зараз звичайну пісню складають із частин, трек за треком: або декілька музикантів грають кожен свою партію, або це робить один музикант (грає за всіх). Використання міді-секвенсорів призводить до того, що деякі треки не мають нічого спільного з вихідним виконанням. Робота інженера та продюсера полягає в тому, щоб зібрати всі ці просторово-часові події та зробити з них музику, в якій кожна частина ідеально підходить до іншої. Щоб зробити це, треба бути трохи художником, трохи вченим. Необхідно знати фізичні основи перетворень, що здійснюються, і вміти грамотно користуватися обладнанням.

Науковий аспект роботи з оброблення звукових сигналів полягає в тому, щоб знати, як поєднати все в єдину систему і як керувати параметрами, що впливають на оброблення звуку. Художній аспект включають під час ухвалення рішення, які ефекти та звуки використовувати, яким має бути баланс та як розмістити різні партії в остаточному міксі».

Для визначення проблем користувачів таких систем було застосовано підхід дизайн-мислення. Це метод розроблення товарів, сервісів та послуг, орієнтованих на користувача, який завжди ставить у центр користувальницький запит і лише потім можливості технічної реалізації та економічні можливості [3, 4]. Для визначення досвіду та «болі» користувача було проведено інтерв’ю – первинний метод збору інформації.

Як виявилося, необхідність у використанні засобів записування і оброблення звукових сигналів на мобільних пристроях пояснюється виникненням різних життєвих ситуацій. Наприклад, необхідно записати лекції в університеті, певну аудіонотатку, коли під рукою немає ручки та аркушу паперу, зробити пояснювальні коментарі до відеоролика, записаного з екрана смартфону, або навіть обробити озвучення чи

дублювання фільму або записані вокальні партії для оформлення власної пісні чи кавера.

Отже, проблема полягає в тому, що не всі користувачі завжди мають змогу працювати з десктоп-версіями такого ПЗ, а доступні вільні засоби оброблення для ОС Android не дозволяють повноцінно реалізовувати власні художні задуми через відсутність можливості виконувати частотний, спектральний аналіз і ручну корекцію сигналів та накладати усі поширені ефекти оброблення, а також змінювати їх параметри, порядок і стан їх активності всього «у пару натисків».

1.2 Аналіз аналогічних рішень для оброблення звукових сигналів

Нині на ринку програмних продуктів існує чимало різних прикладних систем для оброблення звукових сигналів, зокрема для найпопулярнішої мобільної ОС. Для аналізу було вибрано такі три програмні аналоги: “SuperSound” [5], “Wave-Editor” [6] і “Lexis Audio Editor” [7]. На рис. 1.1-1.2 зображено відповідні віконні форми цих засобів.

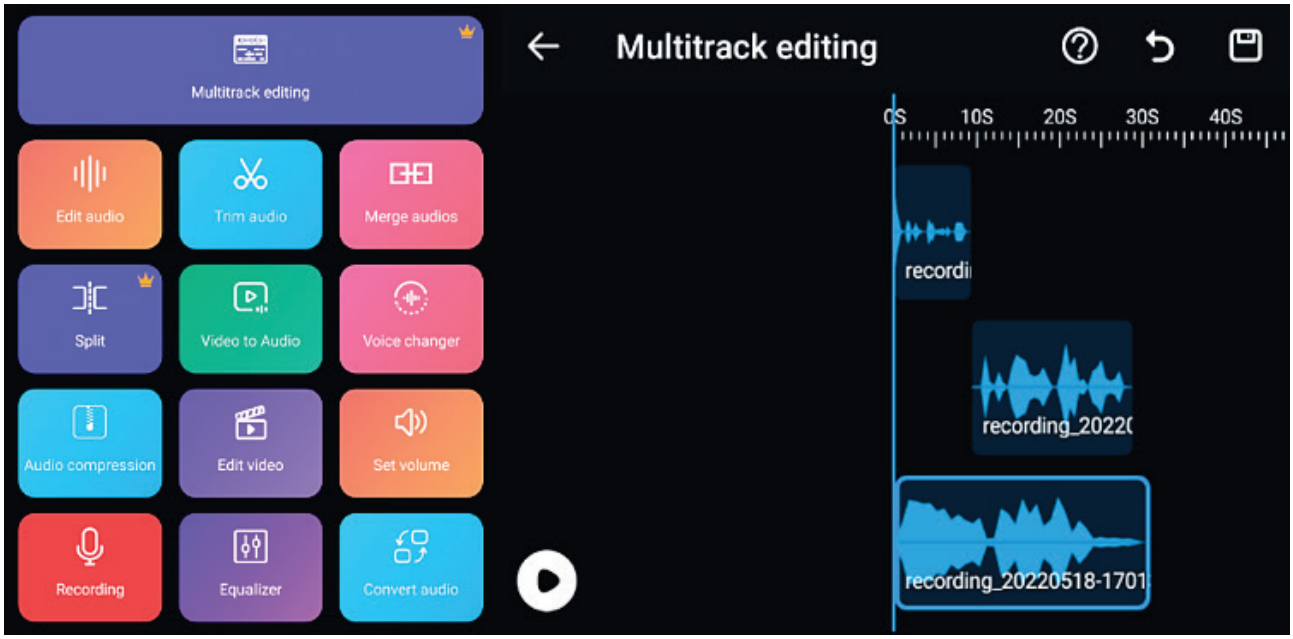
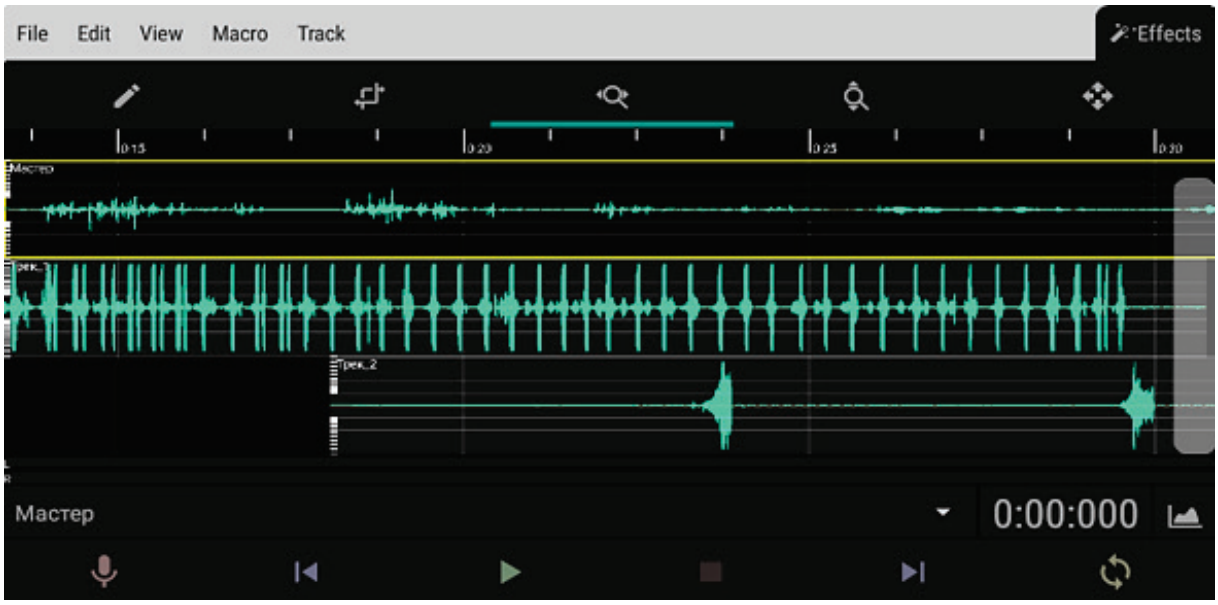
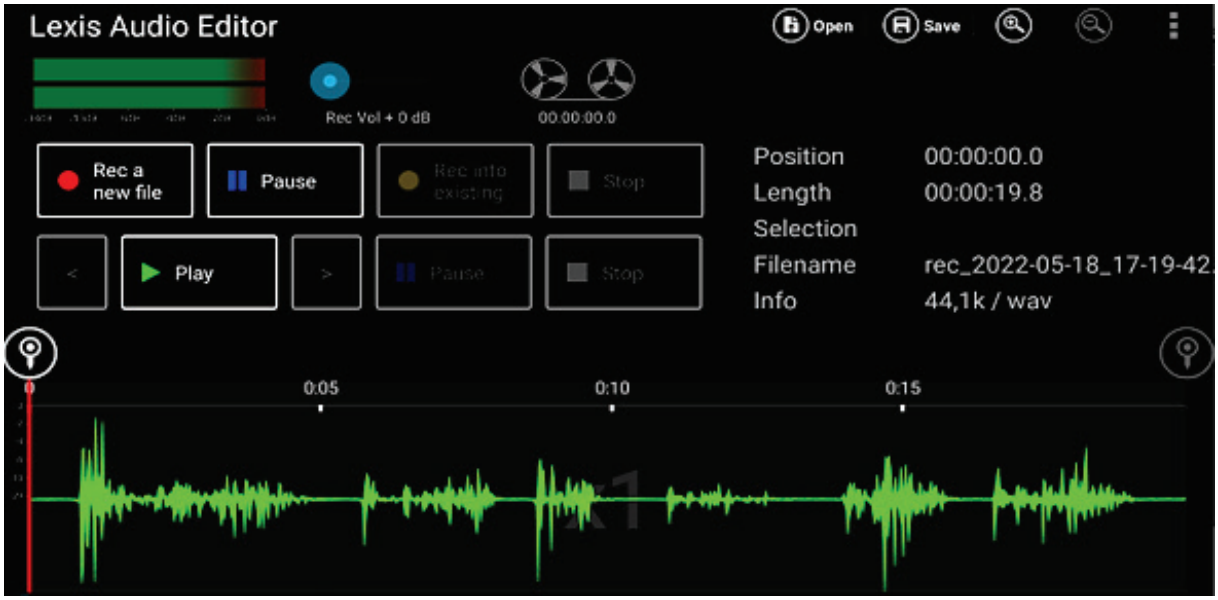


Рисунок 1.1 – Приклад віконної форми застосунка “SuperSound”



а)



б)

Рисунок 1.2 – Приклади віконних форм застосунків
“WaveEditor” (а) і “Lexis Audio Editor” (б)

Для опису можливостей кожної з ПС спочатку розглянемо такі основні поняття з оброблення звукових сигналів:

а) еквалізація – це процес регулювання гучності різних частотних діапазонів сигналу. Схема або обладнання, яке використовується для досягнення цієї мети, називається еквалайзером;

б) компресія – це процес скорочення динамічного діапазону сигналу (різниці між тихими і гучними звуками). Головний інструмент, за допомогою якого здійснюють процес стиснення аудіосигналу – це компресор;

в) реверберація – це процес плавного зниження гучності сигналу разом із його повторюваним відбиттям. Вона створюється, коли звукові хвилі відбиваються від будь-яких поверхонь і поступово згасають у просторі. Імітацію цього ефекту зазвичай здійснюють за допомогою ревербератора;

г) нормалізація – це процес вирівнювання гучності звукового сигналу за максимальним відхиленням його амплітуди до заданих меж, наприклад, гучності іншого звукового сигналу або до рівня 0 дБ;

ґ) затримування – це процес затримки сигналів на попередньо визначений проміжок часу. Одними з варіантів його використання є отримання одинарного відлуння або, наприклад, застосування до одного зі стереоканалів для отримання більш «широкого» звучання;

д) модуляція – це процес перетворення однієї чи кількох характеристик модулювального високочастотного коливання за впливу керувального низькочастотного сигналу. З огляду на те, яка саме характеристика синусоїдального коливання змінюється, розрізняють декілька типів модуляції (див. рис. 1.3):

1) амплітудна: на вхід модулювального пристрою передають модулювальний і опорний сигнали, як наслідок на виході маємо змодульований сигнал. Умовою коректного перетворення вважається подвоєне значення несучої частоти порівняно з максимальним значенням смуги модулювального сигналу;

2) частотна: сигнал модулює частоту опорного сигналу, а не потужність. Тому якщо величина сигналу збільшується, то відповідно зростає частота, зважаючи на те, що смуга одержуваного сигналу набагато ширша за вихідну величину сигналу;

3) фазова: модулювальний сигнал використовує фазу опорного сигналу. Результівний сигнал має досить широкий спектр, тому що фаза обертається на 180°;

4) імпульсна: як несучий сигнал використовують послідовність вузьких імпульсів, як модулювальний – дискретний чи аналоговий сигнал. Оскільки послідовність імпульсів характеризується 4 властивостями, то розрізняють 4 типи модуляції:

частотно-імпульсна, широтно-імпульсна, амплітудно-імпульсна та фазово-імпульсна модуляція.

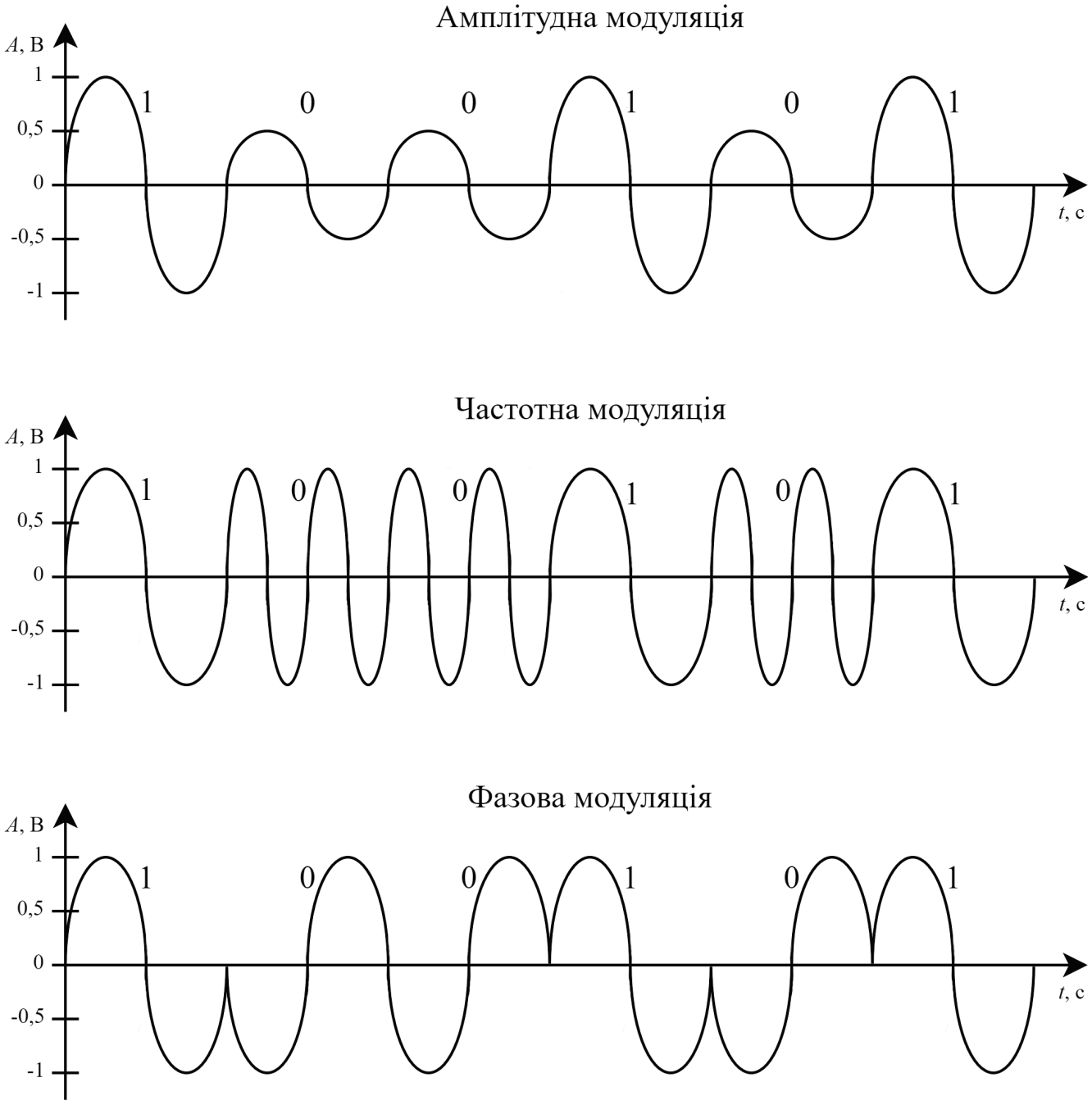


Рисунок 1.3 – Приклади результівного сигналу за різних типів його модуляції

Модулювальні ефекти можна назвати «найефектнішими», оскільки саме їх використання перетворює звучання на зовсім інший, булькотливий, коливний або нявкий звук. До найпоширеніших модуляційних ефектів можна віднести [8]:

1) хорус. На вхід подаємо звучання одного інструменту, а на виході отримуємо відразу звучання кількох, завдяки невеликому затримуванню сигналу. Під час мікшування основного і затриманого сигналів виходить ефект, схожий на одночасне звучання двох інструментів. Окрім зміщення за часом, деякі пристрої також зрушують висоту звучання, що водночас дозволяє цьому ефекту звучати ще яскравіше;

2) фленджер. Уперше цей ефект з'явився наприкінці 60-х років як наслідок експериментів інженерів звукозапису з 2-ма синхронно відтворюваними стрічками з одним і тим же записом. Під час одного з таких дослідів спеціаліст натискав на диск бобіни (англ. flange) одного з двох магнітофонів, на яких відтворювався запис, що призводило до уповільнення її обертання та порушення синхронного відтворення цих магнітофонів. Після відпускання бобіни пригальмований запис «наздоганяв» другий. Ця невелика затримка призвела до виникнення явища, яке отримало назву «гребінчаста фільтрація». У міру переміщення фільтра за спектром частот звучання набувало «охань» і «зітхань», що дуже припало до смаку психоделічному року;

3) фейзер (також зсув фази, фазер). В основі його роботи лежить використання осцилятора з еквалайзером та подальшим мікшуванням відфільтрованого та результівного сигналів. Під час переміщення фільтра тональним спектром за вертикаллю деякі частоти починають компенсувати одна одну за фазою. Фейзер схожий на попередній ефект, за певних налаштувань можна отримати майже однакове звучання. Проте створене звучання цим ефектом є зазвичай яскравішим та приємнішим. Найкраще фейзер підходить для ритм-енд-блюзу та фанку, а також його часто застосовують в рок-композиціях як альтернативу першим двом описаним ефектам;

4) тремоло. Уперше цей ефект було реалізовано в перших ламповых підсилювачах. Принцип його роботи полягає в періодичному змінення амплітуди звукового сигналу, або, інакше кажучи, у швидкому та рівномірному зміненні його гучності. Цей ефект нагадує звучання голосу перед увімкненим вентилятором. Дехто плутає тремоло з вібрато. Останній змінює не гучність сигналу, а висоту тону.

За допомогою застосунка “Super Sound” можна виконувати таке:

а) базові операції з доріжками, як-от їх об’єднання, розділення, записування, відтворення і приглушення;

б) еквалізувати звукові сигнали, змінювати темп, швидкість та висоту тону звучання;

в) імпортувати відеофайли та виокремлювати з них звукові доріжки;

г) працювати з багатьма доріжками одночасно (режим «мультитрек»).

Програмна система “WaveEditor” забезпечує підтримку:

а) усіх базових операцій з аудіотреками, що й у попередньому засобі;

б) таких процесів оброблення сигналу, як еквалізація, компресія, реверберація, нормалізація, затримування, модуляція (хорус, фейзер), змінення темпу, швидкості та висоти тону;

в) імпортування відеофайлів та відокремлення звукового сигналу від них;

г) роботи з багатодоріжковими проєктами.

Останній засіб “Lexis Audio Editor” також підтримує операції об’єднання, розділення, відтворення, записування і приглушення доріжок, а також їх еквалізації, компресії, реверберації, нормалізації, змінення темпу, швидкості, висоти тону, реверсії та відлуння.

У табл. 1.2 наведено порівняльну характеристику цих програмних засобів.

Як наслідок аналізу аналогічних програмних рішень було виявлено такі основні проблеми:

а) ці засоби дозволяють застосовувати лише певні підмножини звукових ефектів, які найбільше використовують у настільному ПЗ, що може обмежувати можливості користувача реалізувати свої художні задуми (див. рис. 1.3);

б) не всі розглянуті програмні аналоги дозволяють оцінювати ступінь і характер вираженості частот сигналу в часі як за осцилограмою, так і за спектрограмою, що може ускладнити користувачеві процес частотної корекції, а відсутність багатодоріжкового режиму може взагалі унеможливити їх мікшування;

Таблиця 1.2 – Порівняння функціональних характеристик продуктів-аналогів

Система / Характеристика	Назва програмного засобу			
	Super Sound	WaveEditor	Lexis Audio Editor	Audiora
Базові операції зі звуковими доріжками (об'єднання, розділення, відтворення, запис, приглушення)	+	+	+	+
Еквалізація, компресія, реверберація, нормалізація, затримування, модуляція (хорус, фейзер, фленджер, тремоло), змінення темпу, швидкості, висоти тону, реверсія, відлуння	–, окрім еквалізації та змінення I-III	+, окрім фленджеру, реверсії, тремоло і відлуння	+, окрім затриму-вання і модуляції	+
Частотний і спектральний аналіз звукових сигналів	–	+	–	+
Відокремлення звукового сигналу від потоку відео	+	+	–	+
Режим одночасної роботи з множиною доріжок	+	+	–	+
Паралельне оброблення звукових сигналів	–	–	–	+

в) критичною проблемою є наявність лише послідовного оброблення звукових сигналів у розглянутих засобах для ОС Android, які мають багатодоріжковий режим.

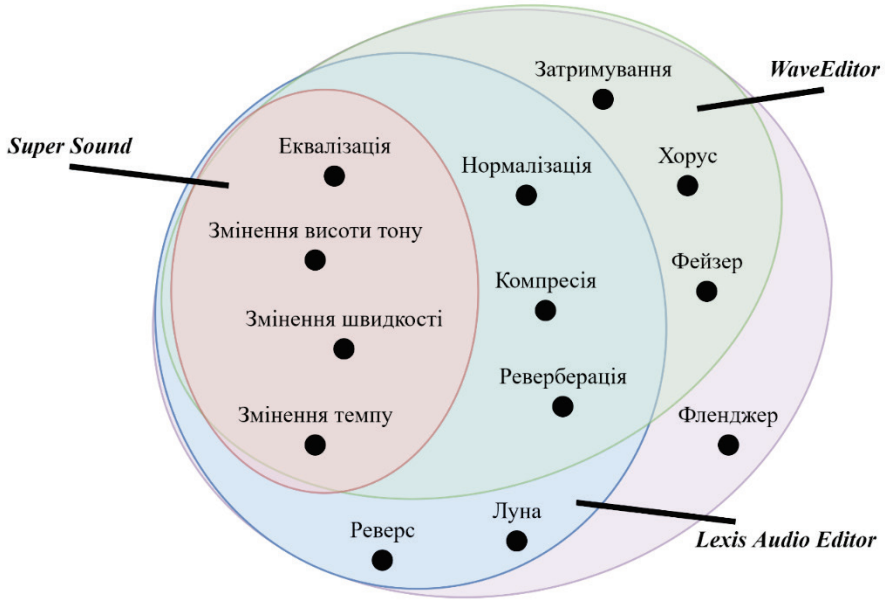


Рисунок 1.3 – Діаграма Венна, яка містить множини ефектів оброблення звукових сигналів, наявних в аналогічних рішеннях

У такому разі втрачаються («псуються») вихідні сигнали, відсутній гнучкий контроль застосування ефектів через лінійність процесу. Користувачеві складно порівнювати і підлаштовувати параметри тих чи тих ефектів, а за припущення помилки, наприклад, неправильно підібраного ефекту або його параметрів ще на початку зведення, користувач буде змушений «відкотитися» назад, утративши весь прогрес у проєкті та, як наслідок, змарнує свій час і зусилля на виконання операцій, яких узагалі можна було уникнути.

1.3 Висновки

У цьому розділі було проаналізовано потреби користувачів та можливості аналогічних програмних рішень. Як наслідок було виявлено, що розглянуті аналоги не задовольняють усім потрібним вимогам. Для розв’язання виявленої критичної проблеми було ухвалено рішення дослідити традиційний підхід до оброблення звукових сигналів та розробити ефективний паралельний метод, який описано в наступному розділі.

2 РОЗРОБЛЕННЯ ЕФЕКТИВНОГО МЕТОДУ ПАРАЛЕЛЬНОГО ОБРОБЛЕННЯ ЗВУКОВИХ СИГНАЛІВ

На підставі проведеного аналізу програмних засобів та наявних проблем можна дійти висновку щодо необхідності розроблення власної програмної системи паралельного цифрового оброблення звукових сигналів “Audiora”, яка дозволить користувачам працювати з аудіофайлами розповсюджених форматів, створювати багатодоріжкові проєкти та паралельно накладати різні ефекти оброблення сигналів на доріжки.

Останнє було визначено як вирішальну функціональну необхідність користувачів, тому на цьому етапі необхідно проаналізувати традиційний підхід, визначити його переваги і недоліки та розробити власний метод, що покликаний розв’язати виявлену критичну проблему аналогічних рішень.

2.1 Огляд наявного підходу до оброблення сигналів звукових доріжок

Звичайний (традиційний) метод оброблення сигналів у наявних програмних засобах полягає в безпосередньому послідовному обробленні вихідних сигналів у всьому багатоканальному проєкті.

Цей підхід використовує мінімальну кількість пам’яті, оперативно зберігаються попередні версії вихідних сигналів, отримані під час чергової сесії проєкту, останні з яких кодуються та зберігаються в сховищі даних до початкових аудіофайлів.

Проте у нього є низка недоліків:

а) втрачання вихідних сигналів через пряме застосування ефектів оброблення до них;

б) неможливість виправлення попередніх накладених ефектів без втрачання подальших операцій;

в) необхідність у надолужуванні всіх скасованих операцій у разі виправлення попередніх ефектів;

г) можливе псування сигналів шляхом неякісного послідовного намішування різних ефектів через небажання повертатися до попередніх станів або їх утрату.

Розглянемо приклад застосування такого методу оброблення (див. рис. 2.1).

Нехай є деякий сигнал S_1 і потрібно додати на його доріжку певний ефект e_1. Припустимо, що було ухвалено рішення додати такі ефекти, як-от хорус e_1, потім затримування e_2, а після нього – реверберацію e_3. Що відбувається в цьому разі:

а) звук S_1 обробляється послідовно, спершу проходячи через модулятор, який застосовує ефект хорусу;

б) далі проміжний сигнал S_1' зазнає певної попередньо визначеної затримки;

в) потім отриманий сигнал проходить через ревербератор. На виході отримуємо сигнал S_1'''.

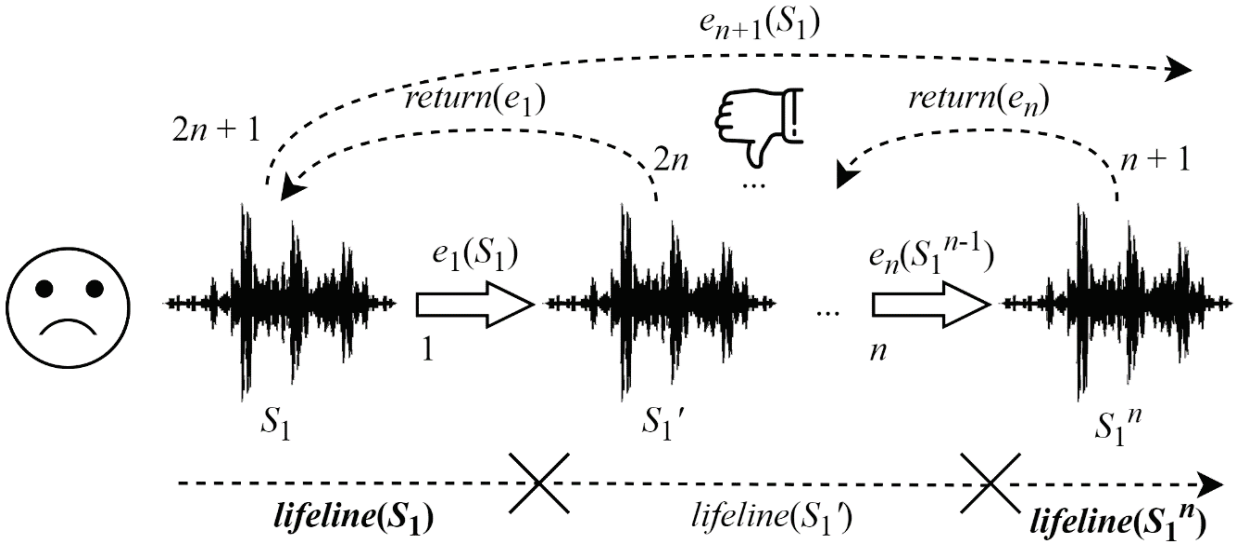


Рисунок 2.1 – Послідовний метод оброблення звукових сигналів

Таким чином, після застосування n ефектів до вихідного сигналу S_j із множини доріжок потужності m у багатодоріжковому проєкті перетворюється в сигнал S_j^n, де $j = \overline{1, m}$. Це і є лінійний, або послідовний, спосіб оброблення звуку. Далі розглянемо інший, принциповий для реалізації в розроблюваній ПС метод.

2.2 Паралельний метод оброблення звукових сигналів

Запропонований підхід полягає не в послідовному обробленні безпосередньо вихідного сигналу S_j покроково ефект за ефектом, а в паралельному застосуванні до копії цього сигналу $C_j = copy(S_j)$ деякого ланцюжка ефектів E_k, після чого сигнал зберігається в окремому оперативному аудіофайлі під час чергової сесії проєкту в системі (див. рис. 2.2).

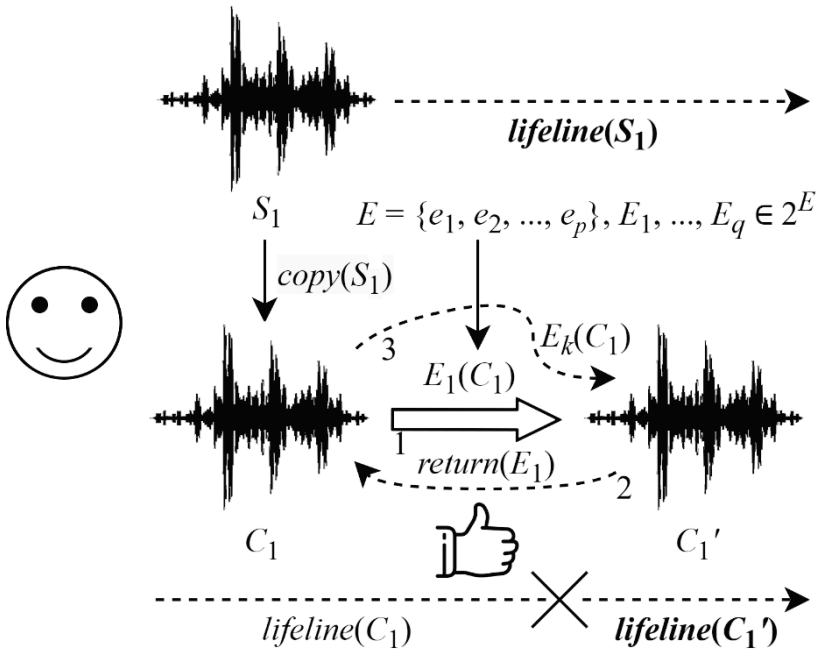


Рисунок 2.2 – Паралельний метод оброблення звукових сигналів

Цей метод надає користувачеві такі переваги:

а) зберігання вихідного сигналу від моменту його імпортування до проєкту до останньої операції оброблення, навіть після завершення роботи ПС. Як видно з рис. 2.2, його «лінія життя» (англ. lifeline) зберігається протягом усього процесу;

б) швидке повернення до первинної копії сигналу шляхом легкого скасування ланцюжка ефектів та навпаки. Тут кортеж E_k, – це упорядкована множина елементів, яка є підмножиною множини E всіх наявних у системі ефектів оброблення, $k = \overline{1, q}$;

в) тонке налаштування звучання звукових доріжок завдяки можливості частково або повністю змінити застосований набір ефектів – усього «в пару натисків».

2.3 Попередня оцінка ефективності застосування методів оброблення звукових сигналів

Спочатку введемо таку множину:

$$O = \{o_i\}, i = \overline{1, N}, \tag{2.1}$$

де o_i – i-та операція додавання чи видалення звукового ефекту, а також скасування однієї з цих двох операцій; позначимо їх відповідно як “+”, “–” і “×”,

N – кількість виконаних операцій від останнього збереження або створення проєкту (якщо до цього моменту перезапуски ПС відсутні).

Нехай користувач послідовно обробляє звукові сигнали і йому необхідно видалити ефект, застосований на j-й операції o_j, $1 \le j \le N$, та який не був видалений операцією o_{j+1}, або зробити n видалень ефектів, перший з яких було додано операцією o_j.

Для цього йому необхідно виконати:

а) $s = (N - j + 1)$ операцій “×”;

б) $d = (s - n - 2p)$ операцій “+” – вони були скасовані, проте їх не заплановано видаляти, тобто їх треба повторити, інакше вони входять до числа n, де p – кількість пар взаємовиключних операцій “+” і “–” серед множини $\{o_j, o_{j+1}, \ldots o_N\}$, виконаних над одним ефектом оброблення, $0 \le p \le s \text{ div } 2$, – їх не потрібно повторювати.

Водночас за використання запропонованого паралельного методу користувачеві потрібно виконати всього n операцій “–”. Проте виграш буде відсутнім при $n = 1$ та $j = N$.

2.4 Висновки

У цьому розділі було розглянуто традиційний метод оброблення звукових сигналів та запропоновано власний підхід, а також наведено порівняльну оцінку кількості виконуваних операцій накладання ефектів на звукові доріжки під час застосування кожного з методів.

3 СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

3.1 Функціональні вимоги до ПС

Для визначення функціональних вимог до розроблюваної системи необхідно описати варіанти її використання (ВВ) за допомогою відповідної UML-діаграми (див. рис. 3.1) та сценаріїв цих ВВ.

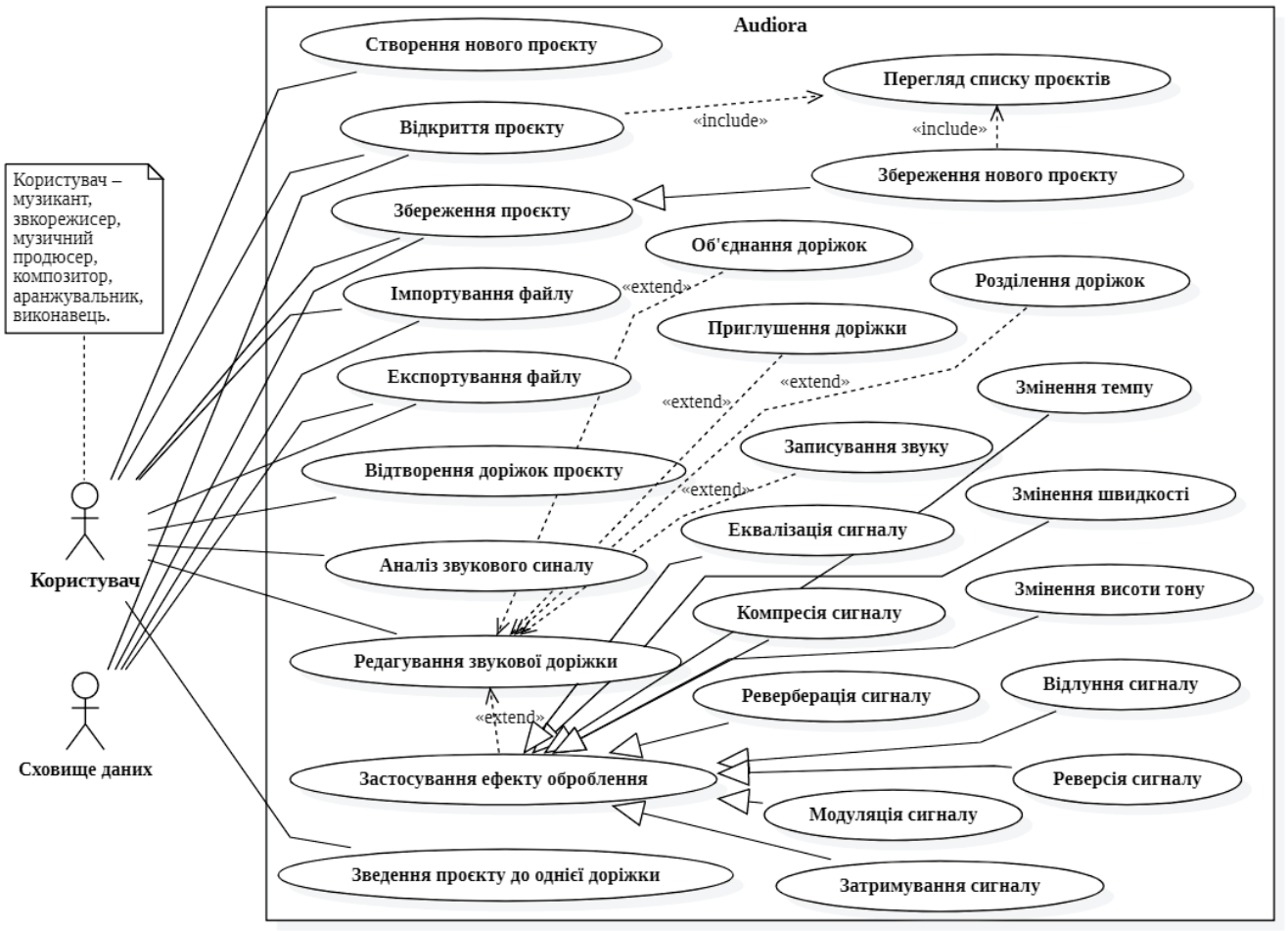


Рисунок 3.1 – Діаграма варіантів використання системи “Audiora”

У [9] наведено такі визначення понять щодо функціональних вимог:

а) «варіантом використання називають певний сценарій дій системи, який забезпечує відчутний і значущий для її користувачів результат. На практиці у вигляді одного ВВ оформляють сценарій дій системи, який буде, швидше за все,

неодноразово виникати під час її роботи та має досить чітко визначені умови початку та завершення виконання»;

б) «актор є будь-якою зовнішньою до модельованої системи сутністю, що взаємодіє із системою та використовує її функціональні можливості для досягнення визначених цілей або рішення особистих задач. Кожен актор може розглядатися як певна окрема роль щодо конкретного ВВ»;

в) «діаграма варіантів використання – це діаграма, на якій зображують відносини між акторами та ВВ»;

г) «сценарій – це певна послідовність дій, яка описує дії акторів і поведінку модельованої системи у формі звичайного тексту».

Сценарії певних ВВ розроблюваної системи наведено в табл. 3.1–3.8.

3.1.1 ВВ «Створення нового проєкту». Актори: Користувач (К). Мета: отримати необхідний шаблон багатодоріжкового проєкту для подальшого запису доріжок, їх оброблення та відтворення. Передумови: К відкрив головне меню системи (С). Тригер: К дає запит на створення нового проєкту. Посилання на інші ВВ: відсутні.

Таблиця 3.1 – Сценарій ВВ «Створення нового проєкту»

Типовий хід подій	
№ з/п	Крок
1	К дає запит на створення нового багатодоріжкового проєкту.
2	С запитує К основну інформацію про проєкт.
3	К уводить назву, вказує директорію розташування, частоту дискретизації, розрядність та режим каналів.
4	К завершує створення файлу проєкту.

Продовження таблиці 3.1

Типовий хід подій	
№ з/п	Крок
5	С обробляє введені К дані та відображає йому створений багато-доріжковий проєкт.
Альтернативний хід подій	
№ з/п	Крок
3a	К не вводить основну інформацію про проєкт, залишаючи значення за замовчуванням: назва – “Untitled_Project_N”, де N – порядковий номер проєкту серед решти з аналогічною назвою в директорії, розташування – “<сховище_даних>/Audiora/Projects”, частота – 48 кГц, розрядність – 16 біт, режим – стерео. 3a.1. Перехід до п. 4.
3b	К частково вводить основну інформацію про проєкт, залишаючи решту значень за замовчуванням. 3a.1. Перехід до п. 4.
3c	К скасовує створення проєкту. 3c.1. Завершення прецеденту.
4a	К вирішує змінити основну інформацію. 4a.1. Перехід до п. 3.
4b	К скасовує створення проєкту. 4b.1. Завершення прецеденту.

3.1.2 ВВ «Відкриття проєкту». Актори: Користувач (К), Сховище даних (СД). Мета: отримати вже створений багатодоріжковий проєкт для його аналізу, редагування і/або відтворення. Передумови: К відкрив головне меню системи (С).

Тригер: К дає запит на відкриття багатодоріжкового проєкту. Посилання на інші ВВ: Включає до себе ВВ «Перегляд списку проєктів».

Таблиця 3.2 – Сценарій ВВ «Відкриття проєкту»

Типовий хід подій	
№ з/п	Крок
1	К дає запит на відкриття багатодоріжкового проєкту.
2	ВВ «Перегляд списку проєктів».
3	К вибирає проєкт зі списку та підтверджує свій вибір.
4	С зчитує вибраний проєкт зі СД та відображає його К.
Альтернативний хід подій	
№ з/п	Крок
3a	К вирішив прискорити вибір проєкту і/або не може знайти його в наданому списку. 3a.1. К уводить назву необхідного проєкту для його пошуку. 3a.2. С зчитує зі СД оновлений список проєктів та виводить його. 3a.3. К вибирає необхідний проєкт із оновленого списку та підтверджує свій вибір.
3a.2a	Проєкт із уведеною К назвою відсутній. 3a.2a.1. С виводить відповідне повідомлення. 3a.2a.2. Перехід до п. 3a.1.
3b	К скасовує відкриття проєкту. 3b.1. Завершення прецеденту.
4a	СД відмовив у доступі, проєкт відсутній у ньому або її файл пошкоджено. 4a.1. С виводить повідомлення про відповідну помилку.

3.1.3 ВВ «Збереження проєкту». Актори: Користувач (К), Сховище даних (СД). Мета: оновити файл наявного багатодоріжкового проєкту із внесеними змінами та можливістю подальшої роботи з ним. Передумови: К відкрив проєкт у вікні редагування системи (С). Тригер: К дає запит на збереження багато-доріжкового проєкту. Посилання на інші ВВ: Предок відносно ВВ «Збереження нового проєкту».

Таблиця 3.3 – Сценарій ВВ «Збереження проєкту»

Типовий хід подій	
№ з/п	Крок
1	К дає запит на збереження багатодоріжкового проєкту.
2	С зберігає його до СД і виводить повідомлення про успіх операції.
Альтернативний хід подій	
№ з/п	Крок
2а	СД відмовив у доступі або недостатньо пам’яті в ньому. 2а.1. С виводить повідомлення про помилку збереження.

3.1.4 ВВ «Збереження нового проєкту». Актори: Користувач (К), Сховище даних (СД). Мета: записати файл нового багатодоріжкового проєкту або копію наявного із внесеними змінами та можливістю подальшої роботи з ним. Передумови: К відкрив проєкт у вікні редагування системи (С). Тригер: К дає запит на збереження нового багатодоріжкового проєкту. Посилання на інші ВВ: нащадок відносно ВВ «Збереження проєкту», включає до себе ВВ «Перегляд списку проєктів».

Таблиця 3.4 – Сценарій ВВ «Збереження нового проєкту»

Типовий хід подій	
№ з/п	Крок
1	К дає запит на збереження нового багатодоріжкового проєкту.

Продовження таблиці 3.4

2	ВВ «Перегляд списку проєктів».
3	К уводить назву проєкту та підтверджує його збереження.
4	С зберігає його і виводить відповідне повідомлення про успіх операції.
Альтернативний хід подій	
№ з/п	Крок
3а	К скасовує збереження нового багатодоріжкового проєкту. 3а.1. Завершення прецеденту.
4а	Проєкт із такою назвою вже існує, СД відмовив у доступі або недостатньо пам’яті в ньому. 4а.1. С виводить повідомлення про помилку збереження проєкту.

3.1.5 ВВ «Перегляд списку проєктів». Актори: Користувач (К), Сховище даних (СД). Мета: отримати уявлення про наявні багатодоріжкові проєкти. Передумови: К відкрив головне меню або вікно редагування системи (С). Тригер: К дає запит на відкриття або збереження нового проєкту. Посилання на інші ВВ: включений до ВВ «Відкриття проєкту» і «Збереження нового проєкту».

Таблиця 3.5 – Сценарій ВВ «Перегляд списку проєктів»

Типовий хід подій	
№ з/п	Крок
1	С зчитує зі СД список проєктів із вказанням їх назв і дат збереження.
2	К вибирає проєкт із наданого списку і запитує більше інформації про неї.

Продовження таблиці 3.5

3	С виводить основну інформацію про вибраний багатодоріжковий проєкт.
Альтернативний хід подій	
№ з/п	Крок
1а	СД відмовив у доступі. 1а.1. С виводить повідомлення про помилку відображення списку проєктів.
2а	К виконує одну з наступних дій, передбачених ВВ, які включають цей. 2а.1. Завершення прецеденту.

3.1.6 ВВ «Відтворення доріжок проєкту». Актори: Користувач (К). Мета: прослухати звучання наявних доріжок у багатодоріжковому проєкті. Передумови: К відкрив проєкт у вікні редагування системи (С). Тригер: К дає запит на початок відтворення відкритого проєкту. Посилання на інші ВВ: відсутні.

Таблиця 3.6 – Сценарій ВВ «Відтворення доріжок проєкту»

Типовий хід подій	
№ з/п	Крок
1	К дає запит на початок відтворення доріжок відкритого проєкту.
2	С враховує вміст проєкту і відповідні параметри та відтворює його.
3	К дає запит на завершення відтворення цього проєкту.
4	С зупиняє відтворення багатодоріжкового проєкту.

Продовження таблиці 3.6

Альтернативний хід подій	
№ з/п	Крок
2а	Багатодоріжковий проєкт порожній. 2а.1. С виводить відповідне повідомлення. 2а.2. Перехід до п. 4.
3а	К вирішує прослухати весь проєкт. 3а.1. С продовжує його відтворення до кінця (найдальшого з усіх доріжок) і зупиняє його.

3.1.7 ВВ «Зведення проєкту до однієї доріжки». Актори: Користувач (К). Мета: отримати один єдиний аудіофайл, який поєднує звучання усіх активних доріжок відкритого багатодоріжкового проєкту разом із урахуванням накладених на них звукових ефектів. Передумови: К відкрив проєкт у вікні редагування системи (С). Тригер: К дає запит на мікшування проєкту до нового файлу. Посилання на інші ВВ: відсутні.

Таблиця 3.7 – Сценарій ВВ «Зведення проєкту до однієї доріжки»

Типовий хід подій	
№ з/п	Крок
1	К дає запит на мікшування всього проєкту до нового файлу.
2	С отримує запит і виконує зведення, водночас показуючи прогрес і теоретичний часовий залишок.
3	Після завершення операції С відображає кінцевий файл в окремій доріжці багатодоріжкового проєкту.

Продовження таблиці 3.7

Альтернативний хід подій	
№ з/п	Крок
1а	К вказує певне виділення за часом, яке необхідно звести. 1а.1. Перехід до п. 2.
2а	К скасовує мікшування під час виконання операції. 2а.1. Завершення прецеденту.

3.1.8 ВВ «Застосування ефекту оброблення». Актори: Користувач (К). Мета: змінити звучання звукового сигналу згідно з попереднім задумом К. Передумови: К вибрав необхідну доріжку у вікні редагування системи (С). Тригер: К дає запит на додавання звукового ефекту до вибраної доріжки. Посилання на інші ВВ: предок відносно ВВ «Еквалізація сигналу», «Компресія сигналу», «Реверберація сигналу», «Модуляція сигналу», «Затримування сигналу», «Відлуння сигналу», «Реверсія сигналу», «Змінення темпу», «Змінення швидкості» та «Змінення висоти тону».

Таблиця 3.8 – Сценарій ВВ «Застосування ефекту оброблення»

Типовий хід подій	
№ з/п	Крок
1	К дає запит на додавання звукового ефекту до вибраної доріжки.
2	С отримує запит і виводить список доступних ефектів оброблення.
3	К вибирає необхідний ефект з наданого С списку.
4	С відкриває вікно налаштування вибраного К ефекту.

Продовження таблиці 3.8

Типовий хід подій	
№ з/п	Крок
5	К встановлює параметри рівня входу, виходу сумарного сигналу, початкового, результівного сигналу за гучністю та/або інші параметри, передбачені для кожного ефекту оброблення.
6	К завершує налаштування ефекту і підтверджує виконання операції оброблення сигналу.
7	С виконує оброблення, водночас показуючи прогрес і теоретичний часовий залишок.
8	С виводить повідомлення про успішне накладання звукового ефекту на доріжку після завершення операції.
Альтернативний хід подій	
№ з/п	Крок
3а, 5а, 6а	К скасовує налаштування вибраного ефекту оброблення. 3а.1, 5а.1, 6а.1. Завершення прецеденту.
7а	К скасовує операцію оброблення під час її виконання. 7а.1. Завершення прецеденту.

3.2 Нефункціональні вимоги до ПС

3.2.1 Вимоги до надійності. Необхідно забезпечити через запровадження перевірки полів для введення даних, графічного інтерфейсу, який не дозволяє користувачеві вносити зміни, що призводять до збоїв ПС, та періодичного збереження даних проєктів до СД (кожні 10 хвилин), а також архітектурних компонентів, які

забезпечують правильне та автоматичне оброблення ЖЦ складових системи, своєчасне оновлення даних відповідно до необхідного стану останніх, коректне змінення їх налаштувань та запобігання витоку пам’яті.

3.2.2 Вимоги до продуктивності. Максимальний час створення файлу нового проєкту (8 доріжок) та його відкриття у вікні редагування – 0,5 секунди, збереження даних – 2 секунди, побудова спектрограми сигналу – 1 секунда.

3.2.3 Вимоги до зручності використання. Користувач має витрачати на вивчення всіх можливостей ПС не більше 1 години, відчувати себе комфортно під час редагування звукової доріжки у 97% випадків та видаляти/деактивовувати будь-який застосований ефект оброблення звукової доріжки не більше 3-х секунд.

3.2.4 Системні вимоги. Центральний процесор із тактовою частотою ≥ 1,2 ГГц, мінімальний об’єм пам’яті ОЗП – 2 Гб, вільної пам’яті у внутрішньому сховищі для встановлення ПС – 100 Мб, ОС – Android 5.0 або новішої версії, необхідна підтримка технології USB OTG.

3.3 Висновки

У цьому розділі було сформульовано технічне завдання на розроблення програмної системи у вигляді специфікації вимог, де було визначено функціональні (за допомогою діаграми та сценаріїв варіантів використання) та нефункціональні вимоги до неї.

4 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

4.1 Архітектура програмного продукту

Розроблювана ПС є інтерактивним застосуванням із гнучким людино-машинним інтерфейсом, тому було ухвалено рішення використати архітектурний стиль MVVM (англ. Model-View-ViewModel – дослівно «модель, представлення, модель представлення») разом із Android Architecture Components – набором бібліотек від компанії Google, який дозволяє створювати програми, враховуючи підтримку ЖЦ користувальницького інтерфейсу та оброблення збереження даних [10].

Цей патерн дозволяє задати архітектуру всієї ПС та містить 3 складові: бізнес-логіка, дані, правила – «модель», дані та логіка представлення – «модель представлення» та графічний інтерфейс користувача – «представлення», або «подання» (див. рис. 4.1).

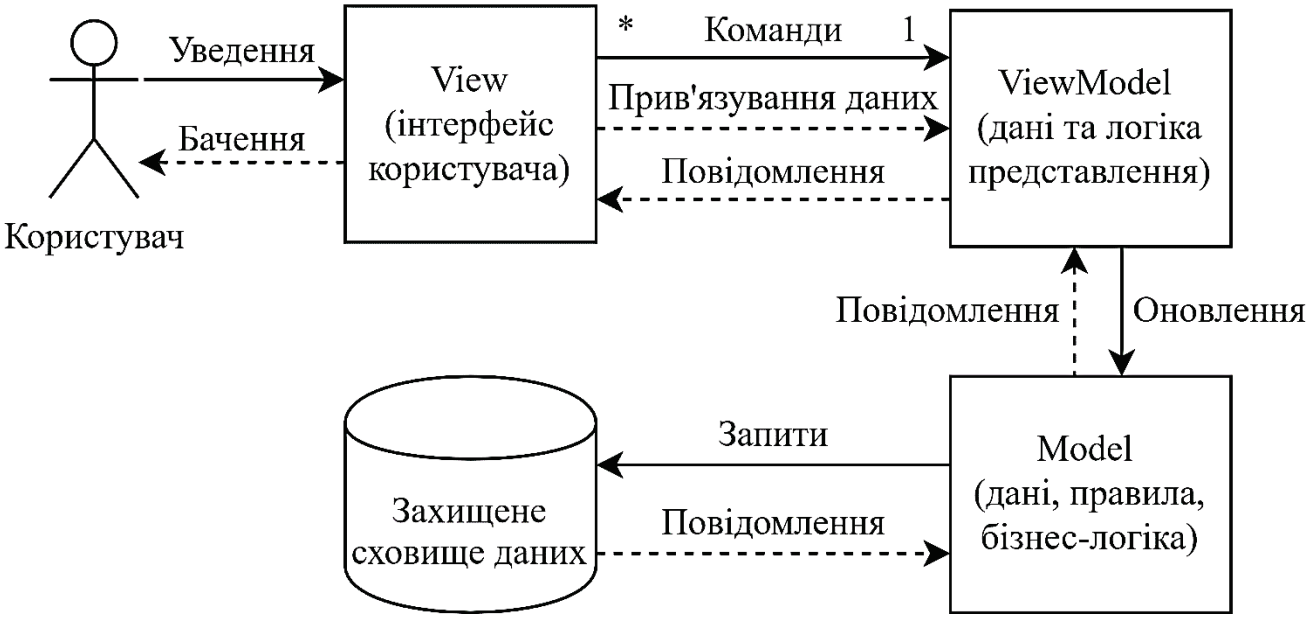


Рисунок 4.1 – Структура архітектурного шаблону MVVM

Цей архітектурний стиль є удосконаленою модифікацією стилів MVC і MVP, оскільки він передбачає краще структурування вихідного коду, зменшення його кількості завдяки механізму «прив’язування даних», поліпшення читабельності,

тестованості та супроводжуваності – моделі (бізнес- та подання) не залежать від самого подання.

Архітектурні компоненти Андроїд зазвичай застосовують для коректного автоматичного оброблення ЖЦ компонентів ПС, як-от активності та фрагменти, запобігання витоку пам’яті, вчасного оновлення даних відповідно до потрібного стану активності, правильної зміни конфігурації цих складових та кешування даних у разі поганого зв’язку з Інтернетом чи його відсутності.

4.2 Структура та організація програмних класів

Згідно з [11], «діаграма класів ілюструє специфікації програмних класів і інтерфейсів (наприклад, інтерфейсів Java) в застосуванні. Зазвичай на таку діаграму виносять таку інформацію: класи, асоціації та атрибути, інтерфейси зі своїми операціями і константами, методи, інформацію про типи атрибутів, способи навігації та залежності». На рис. 4.2 зображено діаграму спроєктованих основних програмних класів системи.

На цьому етапі вона складається із 13-ти класів, серед яких є: Project (багатодоріжковий проєкт), Track (доріжка), Clip (кліп), Signal (сигнал), Effect (звуковий ефект), Playback (програвач) тощо.

Між цими класами є відношення композиції («ціле – частина», де складові частини цілого не можуть існувати без нього: Project ↔ Playback, Project ↔ Track, Track ↔ Clip, Clip ↔ Signal) та агрегації («контейнер – частина»: Track ↔ Effect).

Розглянемо кожен із цих класів та опишемо їх атрибути і методи:

а) клас Project – клас моделі, який відповідає багатодоріжковому проєкту:

1) title – атрибут рядкового типу з приватним модифікатором доступу, який визначає назву проєкту, значення за замовчуванням – “Untitled_Project_”;

2) source – атрибут рядкового типу з приватним модифікатором доступу, який визначає директорію розташування проєкту, значення за замовчуванням – “/Audiora/Projects/”;

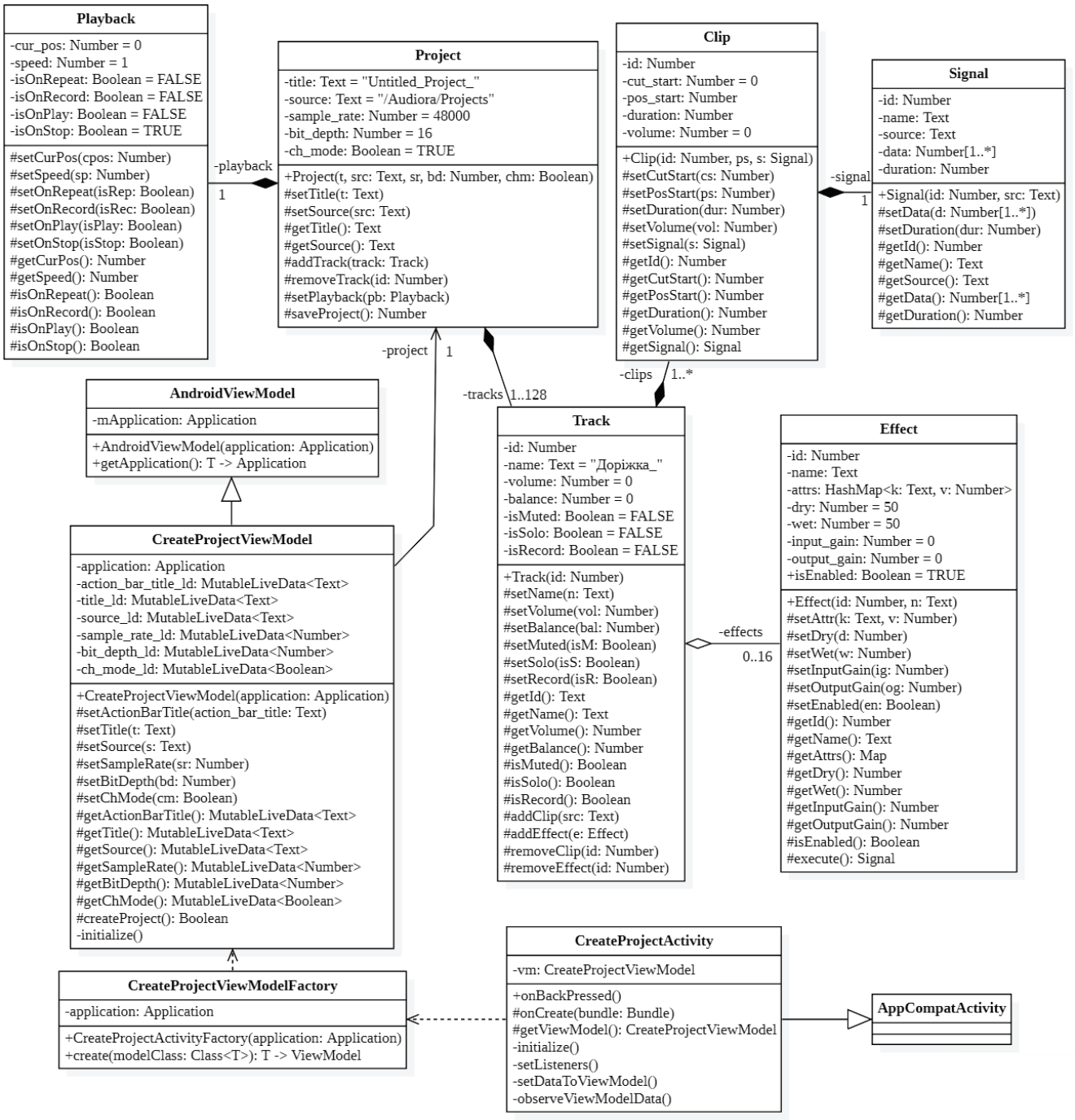


Рисунок 4.2 – Діаграма основних програмних класів ПС “Audiora”

3) sample_rate – атрибут числового типу з приватним модифікатором доступу, який визначає частоту дискретизації проєкту, значення за замовчуванням – 48000;

4) bit_depth – атрибут числового типу з приватним модифікатором доступу, який визначає розрядність проєкту, значення за замовчуванням – 16;

5) ch_mode – атрибут булевого типу з приватним модифікатором доступу, який визначає режим каналів проєкту (моно або стерео), значення за замовчуванням – TRUE;

5) Project – явно описаний конструктор з публічним модифікатором доступу, який приймає п’ять параметрів, що відповідають вищезазначеним атрибутам;

6) addTrack() – метод із захищеним модифікатором доступу, призначений для додавання нової доріжки до проєкту і який приймає об’єкт класу Track;

7) removeTrack() – метод із захищеним модифікатором доступу, призначений для видалення доріжки з проєкту і який приймає унікальний ідентифікатор цієї доріжки чисельного типу;

Тут і далі методи, назви яких мають префікс “set”, є такими, які встановлюють значення, які вони приймають (сетери), до полів, імена яких збігаються із залишками-суфіксами. Для прикладу, метод setSource() бере значення рядкового типу і встановлює його до поля source класу Project.

Аналогічно для гетерів – методів із префіксом “get”, які повертають певне значення поля відповідного типу. Також у класах присутні методи з префіксами “add” і “remove” у назві, які відповідають за додавання і видалення агрегованих об’єктів відповідно в списках цих класів. Конструктори всіх класів моделі мають публічний модифікатор доступу, методи – захищений, атрибути – приватний.

б) клас Track – клас моделі, який відповідає звуковій доріжці:

1) id – атрибут числового типу, який визначає унікальний ідентифікатор доріжки;

2) name – атрибут рядкового типу, який визначає назву доріжки, значення за замовчуванням – «Доріжка_»;

3) volume – атрибут числового типу, який визначає посилення доріжки за гучністю у дБ, значення за замовчуванням – 0;

4) balance – атрибут числового типу, який визначає відсоток відхилення звучання доріжки до одного з каналів (лівого або правого), значення за замовчуванням – 0;

5) isMuted() – атрибут булевого типу, який визначає приглушеність доріжки, значення за замовчуванням – FALSE;

6) isSolo() – атрибут булевого типу, який визначає, чи є доріжка сольною, значення за замовчуванням – FALSE;

7) isRecord() – атрибут булевого типу, який визначає, чи вибрано доріжку для записування, значення за замовчуванням – FALSE;

8) Track – явно описаний конструктор, який приймає один параметр – унікальний ідентифікатор доріжки;

в) клас Clip – клас моделі, який відповідає аудіокліпу:

1) id – атрибут числового типу, який визначає унікальний ідентифікатор кліпу;

2) cut_start – атрибут числового типу, який визначає початковий момент обрізання кліпу відносно файлу (сигналу), якому він відповідає, значення за замовчуванням – 0;

3) pos_start – атрибут числового типу, який визначає початок кліпу на часовій лінії проєкту;

4) duration – атрибут числового типу, який визначає тривалість кліпу в кількості семплів;

5) volume – атрибут числового типу, який визначає посилення власне кліпу за гучністю в дБ, значення за замовчуванням – 0;

6) Clip – явно описаний конструктор з публічним модифікатором доступу, який приймає два параметри: перший – числового типу (id – ідентифікатор кліпу), другий – типу Signal (s – об’єкт сигналу);

г) клас Signal – клас моделі, який відповідає сигналу – імпортованому аудіофайлу, перетвореному до формату лінійної імпульсно-кодової модуляції, або PCM:

1) id – атрибут числового типу, який визначає унікальний ідентифікатор сигналу;

2) name – атрибут рядкового типу, який визначає назву файлу, який містить цей сигнал;

3) source – атрибут рядкового типу, який визначає директорію розташування цього файлу;

4) data – атрибут числового типу – масив декодованих значень амплітуди сигналу у визначені моменти часу (згідно з частотою дискретизації останнього);

5) duration – атрибут числового типу, який визначає тривалість сигналу в кількості семплів;

4) Signal – явно описаний конструктор, який приймає два параметри: перший – числового типу (id, ідентифікатор сигналу), другий – рядкового (src, директорія розташування файлу з сигналом);

г) клас Effect – клас моделі, який відповідає ефекту звукового оброблення сигналу на доріжці;

1) id – атрибут числового типу, який визначає унікальний ідентифікатор застосованого ефекту;

2) name – атрибут рядкового типу, який визначає назву цього ефекту;

3) attrs – атрибут рядкового типу – хеш-таблиця, яка визначає інші властивості ефекту оброблення;

4) dry – атрибут числового типу, який визначає відсоток вихідного сигналу, значення за замовчуванням – 50;

5) wet – атрибут числового типу, який визначає відсоток перетвореного сигналу, значення за замовчуванням – 50;

6) input_gain – атрибут числового типу, який визначає посилення вхідного сигналу за гучністю в дБ, значення за замовчуванням – 0;

7) output_gain – атрибут числового типу, який визначає посилення результівного сигналу за гучністю в дБ, значення за замовчуванням – 0;

8) isEnabled – атрибут булевого типу, який визначає стан активності застосування ефекту, значення за замовчуванням – TRUE;

9) execute() – метод із захищеним модифікатором доступу, призначений для виконання обчислень, пов’язаних з ефектом, який обробляє вхідну доріжку і повертає результівний сигнал – об’єкт типу Signal;

д) клас Playback – клас моделі, який відповідає програвачеві проєкту:

1) cur_pos – атрибут числового типу, який визначає позицію курсора програвача на часовій лінії багатодоріжкового проєкту як кількість попередніх семплів, значення за замовчуванням – 0;

2) speed – атрибут чисельного типу, який визначає швидкість відтворення програвачем доріжок проєкту, значення за замовчуванням – 1;

3) isOnRepeat – атрибут булевого типу, який визначає, чи повторює програвач відвторення з початку проєкту після досягнення кінця останнього або у визначених часових межах;

4) isOnRecord – атрибут булевого типу, який визначає, чи виконується записування звуку на визначені відповідним атрибутом доріжки проєкту;

5) isOnPlay – атрибут булевого типу, який визначає, чи відвторює програвач доріжки проєкту, або він призупинений, значення за замовчуванням – FALSE;

6) isOnStop – атрибут булевого типу, який визначає, чи зупинене відтворення програвачем доріжок проєкту, чи ні, значення за замовчуванням – TRUE;

е) клас CreateProjectActivity – клас подання, що відповідає активності для створення багатодоріжкового проєкту:

1) vm – атрибут типу CreateProjectViewModel, який посилається на модель подання для створення цього проєкту;

2) onBackPressed() – публічний метод, що перевизначає дії, виконувані після натиснення кнопки «Назад»;

3) onCreate() – захищений метод, що задає початкову ініціалізацію параметрів, коли відбувається ініціювання активності;

4) initialize() – приватний метод для початкової ініціалізації полів цього класу (тут і далі);

5) setListeners() – приватний метод для оброблення зміни станів певних елементів подання;

6) observeViewModelData() – приватний метод для спостерігання за всіма потрібними даними в моделі подання.

ж) клас CreateProjectViewModelFactory – клас-фабрика для створення об’єкта класу моделі подання для створення проєкту;

з) клас CreateProjectViewModel – клас, що містить модель, перетворену до подання, а також команди, які може використовувати подання, щоб впливати на модель. Також містить поля типу MutableLiveData – це клас-тримач даних, які спостерігаються, явно описаний публічний конструктор з одним параметром типу Application – це базовий клас для підтримки глобального стану ПС, методи для перевірки, встановлення і отримання даних та створення проєкту.

У Додатку А наведено реалізацію цих програмних класів у вигляді лістингів системи паралельного цифрового оброблення звукових сигналів.

4.3 Опис застосованих шаблонів проєктування

До описаних у попередньому підрозділі класів додатково було вжито такі дизайн-патерни ПЗ [12]:

а) фасад (англ. Facade) – це структурний шаблон. Він полягає в об’єднанні складної системи класів, зокрема у фреймворку чи бібліотеці, під один простий уніфікований інтерфейс, який надає доступ до них через одну вхідну точку.

Перевагою цього патерну є ізолювання клієнтів від складових таких підсистем. Проте за необережного використання він може стати антипатерном «божественний об’єкт», який зберігає в собі або робить «надто багато», наприклад, взаємодіє зі всіма класами ПС.

У розроблюваній системі передбачено роботу з поширеними форматами кодування аудіофайлів, імпортування відео та виокремлення звукових сигналів, мікшування доріжок, забезпечити підтримку яких можна за допомогою впровадження пакету сервісів роботи з медіаданими, що представлені цілою низкою допоміжних програмних класів.

Шаблон проєктування «Фасад» надає єдиний метод, наприклад, для експортування звукового сигналу в форматі Waveform замість безпосередньої роботи з багатьма класами. Цей метод сам піклується про коректну конфігурацію потрібних об’єктів пакету і отримати необхідний результат.

Також його можна застосувати для реалізації накладання пресетів на доріжки, коли користувач викликає окремі методи, які потім налаштовують різні параметри ефектів оброблення;

б) одинак (англ. Singleton) – це породжувальний шаблон. Він забезпечує лише один екземпляр класу з глобальною точкою доступу, наданою цим класом.

Сильними сторонами «Одинака» є:

а) гарантія наявності єдиного екземпляра класу;

б) надання глобальної точки доступу;

в) реалізація відкладеної ініціалізації об’єкта-одинака.

Як його недоліки можна перерахувати таке:

а) порушення SRP (англ. single-responsibility principle) – принцип єдиної відповідальності класу, який полягає в тому, що кожен об’єкт повинен мати одну відповідальність і вона має бути цілком інкапсульована до класу. Уся його поведінка повинна бути націлена лише на забезпечення цієї відповідальності.

б) можлива націленість на маскування поганого дизайну;

в) можливе виникнення проблем багатопоточності.

г) вимога щодо постійного створення Mock-об’єктів під час модульного тестування.

Фасад можна використовувати з Одинаком, оскільки зазвичай необхідно мати лише один об’єкт-фасад. Також можна застосувати цей патерн до класу-програвача розроблюваної ПС, оскільки для багатодоріжкового проєкту потрібен тільки один екземпляр цього класу;

в) стан (англ. State) – це поведінковий шаблон. Він дозволяє об’єктам міняти поведінку відповідно до свого стану.

Переваги цього шаблону:

а) позбавлення безлічі великих умовних операторів скінченного автомату;

б) зосередження програмного коду, який належить до певного стану, в одному місці;

в) спрощення коду контексту.

Можливе невиправдане ускладнення коду за малої кількості станів та їх рідкісної зміни є слабкою стороною «Стану». Цей шаблон є застосовним для опису станів програвача багатодоріжкового проєкту.

г) команда (англ. Command) – це поведінковий шаблон. Він подає запити у вигляді об’єктів та в такий спосіб дозволяє передавати їх як аргументи під час викликання методів, а також чергувати, логувати запити та підтримувати скасування операцій.

Цей патерн позбавляє прямої залежності викликальних об’єктів та об’єктів, які безпосередньо виконують операції, дозволяє реалізувати скасування та повторення, відкладений запуск операцій та збирати прості команди в складні. Водночас він обтяжує код через уведення множини додаткових програмних класів.

д) знімок (Memento) – це також поведінковий шаблон, призначений для зберігання та відновлювання минулих станів об’єктів без розкривання подробиць їх реалізації.

Він не порушує інкапсуляції та полегшує структуру початкового об’єкта, оскільки той не повинен зберігати історію версій свого стану. Проте є низка недоліків: необхідно чимало пам’яті під час частого створення «знімків» користувачем, можливе спричинення додаткової витрати пам’яті за відсутності механізму видалення об’єктами застарілих знімків.

Останні два шаблони проєктування варто поєднати для реалізації скасування операцій у вікні редагування розроблюваної ПС. Так, об’єкти команд відповідатимуть за виконання дій над об’єктом (звуковою доріжкою), а знімки зберігатимуть резервні копії його стану, зроблену перед запуском команди.

4.4 Структура сховища даних багатодоріжкових проєктів

ER-модель (ER – від “entity” – «сутність» і “relationship” – «зв’язок») – це модель даних для описання концептуальних схем предметної області. Її використовують під час високорівневого проєктування БД. Вона дозволяє визначити головні сутності, а також можливі встановлювані між ними зв’язки [13].

Концептуальна ER-модель розроблюваного сховища даних проєктів для ПС “Audiora” має 8 сутностей: «Проєкт», «Програвач», «Доріжка», «Стан доріжки», «Кліп», «Сигнал», «Ефект», «Параметр» – із двома зв’язками «один до одного» (обов’язкові), чотирма зв’язками «один до багатьох» (також обов’язкові) та одним зв’язком «один обов’язковий до багатьох опціональних» (рис. 4.3):

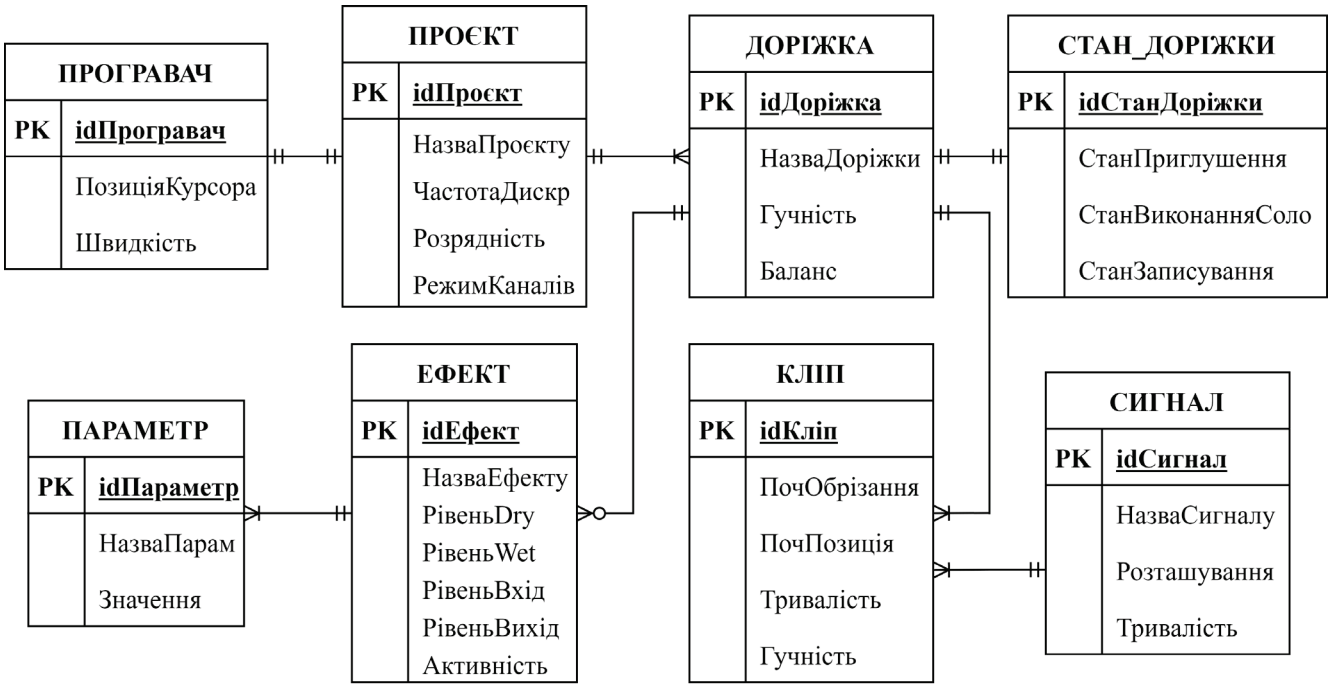


Рисунок 4.3 –ER-діаграма сховища даних багатодоріжкового проєкту

Проєкт характеризується унікальним ідентифікатором (код проєкту), назвою, частотою дискретизації, розрядністю і режимом каналів та має два зв’язки:

а) «один до одного» з сутністю «Програвач» – для відтворення проєкту потрібен лише один програвач, один програвач належить тільки одному проєкту;

б) «один до багатьох» із сутністю «Доріжка» – проєкт повинен мати щонайменше одну доріжку, певна доріжка обов’язково належить лише одному проєкту.

Програвач характеризується унікальним ідентифікатором (код програвача) позицією курсора і швидкістю програвання та має один зв’язок «один до одного» з сутністю «Проєкт»: один програвач належить тільки одному проєкту, для відтворення проєкту потрібен лише один програвач.

Доріжка характеризується унікальним ідентифікатором (код доріжки), назвою і значеннями гучності та балансу і має чотири зв’язки:

а) «один до одного» з сутністю «Стан доріжки» – доріжка має лише одну обов’язкову множину станів, ця конкретна множина належать тільки одній доріжці;

б) три «один до багатьох» з сутностями:

1) «Проєкт» – певна доріжка обов’язково належить лише одному проєкту, проєкт повинен мати щонай-менше одну доріжку;

2) «Кліп» – доріжка має хоча б один кліп (окремий відрізок сигналу), один кліп одночасно належить тільки одній доріжці;

3) «Ефект» – доріжка може мати декілька ефектів, один ефект із певними налаштуваннями накладають на одну доріжку.

Стан доріжки характеризується унікальним ідентифікатором (код станів доріжки) та множиною її станів: приглушення (англ. mute), виконання соло (англ. solo) і записування (англ. record) та має один зв’язок «один до одного» з сутністю «Доріжка»: певна множина станів належить тільки одній доріжці, ця доріжка має лише одну обов’язкову множину станів.

Кліп характеризується унікальним ідентифікатором (код кліпу), початком обрізання відносно сигналу, якому він відповідає, початком на часовій лінії проєкту, власною тривалістю і гучністю та має два зв’язки «один до багатьох» із сутностями:

а) «Доріжка» – один кліп одночасно належить тільки одній доріжці, доріжка має хоча б один кліп;

б) «Сигнал» – цей кліп відповідає тільки одному звуковому сигналу, сигнал можна застосовувати щонайменше в одному кліпі.

Сигнал характеризується унікальним ідентифікатором (код сигналу), назвою, розташуванням файлу і тривалістю та має один зв’язок «один до багатьох» із сутністю «Кліп» – сигнал може відповідати багатьом кліпам, один кліп – лише одному сигналу.

Ефект характеризується унікальним ідентифікатором (код ефекту), назвою, рівнем необробленого сигналу (англ. dry signal), рівнем обробленого сигналу (англ. wet signal), вхідним рівнем (англ. input), вихідним рівнем за гучністю (англ. output) та станом активності. Має два зв’язки «один до багатьох» із сутностями:

а) «Параметр» – ефект оброблення обов’язково має щонайменше один параметр, конкретний параметр належить лише одному ефекту;

б) «Доріжка» – один ефект із певними налаштуваннями накладають на одну доріжку, доріжка можна обробити багатьма ефектами, проте необов’язково.

Параметр ефекту характеризується унікальним ідентифікатором (код параметра), назвою і певним значенням та має один зв’язок «один до багатьох» із сутністю «Ефект» – один параметр із певним значенням відноситься тільки до одного ефекту, один ефект оброблення може мати багато параметрів налаштування.

4.5 Проєктування графічного інтерфейсу користувача

Для створення основних екранних форм ПС було застосовано спеціальний веб-інструмент для розроблення веб- та мобільних макетів Figma [14].

На рис. 4.4–4.6 зображено макети деяких екранних форм ПС і переходи між ними. Кожен із них виконано в темних кольорах із застосуванням одного стандартного системного шрифту узгоджених розмірів, містить чорний фон із зображенням темно-бірюзової абстракції звукового сигналу, а також заголовок із назвою ПС та відповідного відкритого вікна для зручності та комфорту під час користування розроблюваною системою.

Макет головного меню містить чотири кнопки з відповідними піктограмами зліва кожної – білими кругами з графічними позначками всередині них. Натискання першої кнопки призводить до запиту на створення нового багатодоріжкового

проєкту, другої – на відкриття проєкту або аудіофайлу. Користувач зможе налаштувати певні параметри системи, якщо натисне 3-ю кнопку. Також він матиме змогу переглянути інформацію про ПС, керівництво користувача та зв'язатися з розробником після натиснення чертвертої кнопки (див. рис. 4.4).

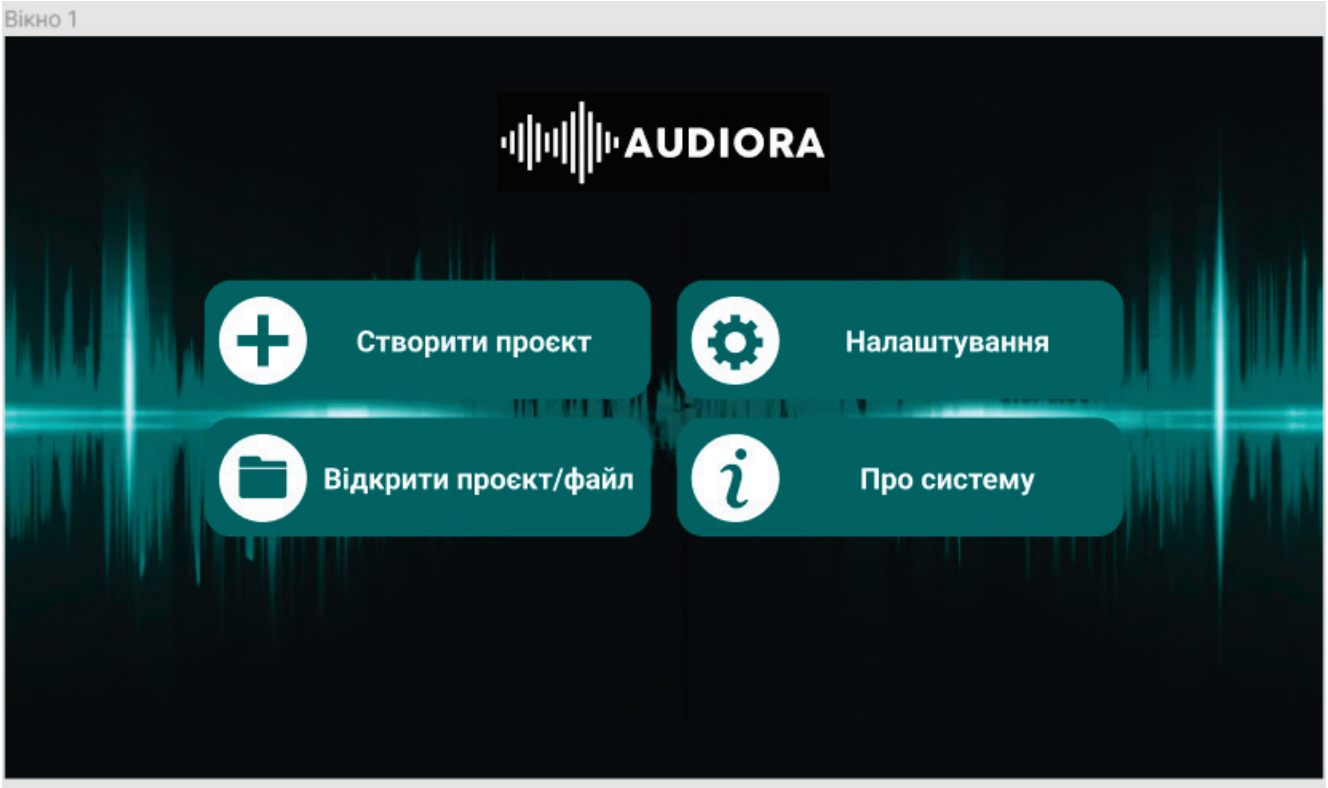


Рисунок 4.4 – Макет головного меню ПС «Audiora»

Макет вікна створення багатодоріжкового проєкту містить одне поле введення назви проєкту з кнопкою очищення цього поля, текстове поле відображення шляху розташування майбутнього проєкту з кнопкою, яка відкриває файловий провідник, після чого користувач може вибрати необхідний каталог і підтвердити свій вибір.

Також у цьому вікні розташовано три випадних списки для зазначення частоти дискретизації, розрядності та режиму каналів проєкту. За замовчуванням вони мають значення «48 кГц», «16 біт» і «стерео» відповідно. Біля кожного списку є відповідна позначка.

Внизу вікна створення проєкту розташовано дві кнопки: «Завершити» і «Скасувати». Натискання першої кнопки призводить до створення багатодоріжкового

проєкту, другої – скасування поточної дії та виходу з цього вікна шляхом повернення до вікна головного меню програмної системи (див. рис. 4.5).

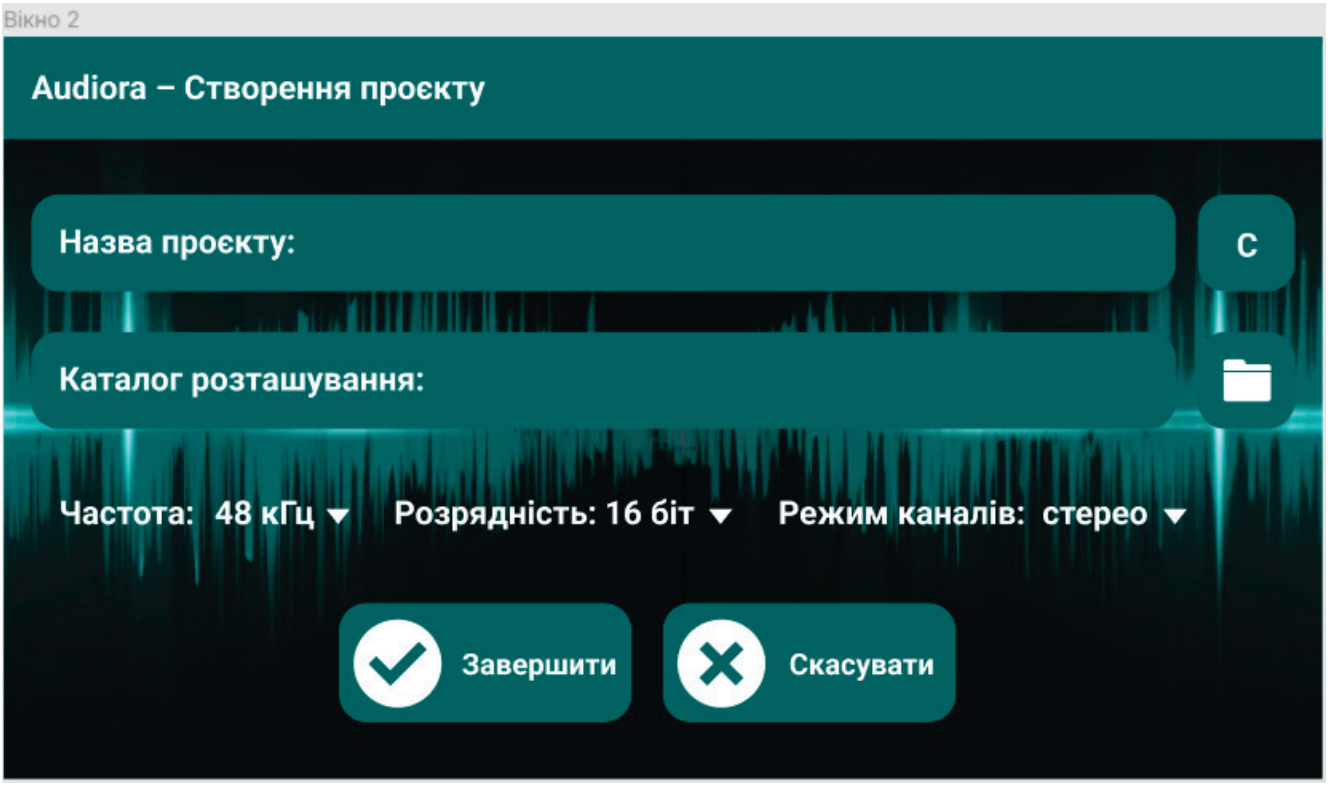


Рисунок 4.5 – Макет вікна створення багатодоріжкового проєкту в ПС “Audiora”

Макет вікна редагування доріжок проєкту складається з:

а) верхньої панелі, яка містить приховане меню, назву вікна, поточного тайм-коду програвача та елементи управління: скасування операції (англ. undo), повернення операції (англ. redo), а також решта елементів (відтворення доріжок, його призупинення, повна зупинка, режими повторення, записування), прихована під кнопкою з піктограмою трьох крапок;

б) робочої ділянки проєкту, де відображено список звукових доріжок, помічених різними кольорами, разом із спрощеним представленням сигналів, які вони містять, на часовій лінії проєкту, розташованій зверху цієї ділянки;

в) додаткової нижньої панелі, яка з’являється після вибору певної доріжки або кліпу, на якій відображено назву доріжки/кліпу, повзунки для змінення значень гучності, панорамування (для доріжки), три кнопки змінення її стану:

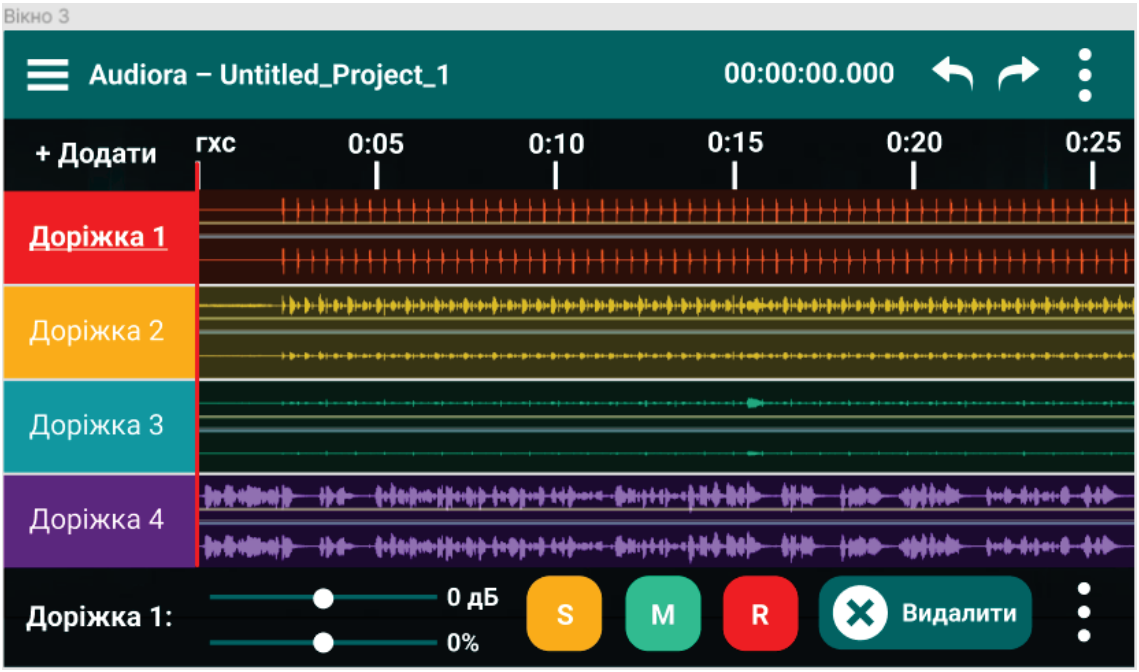


Рисунок 4.6 – Макет вікна редагування багатодоріжкового проєкту в ПС “Audiora”

1) “S” (від англ. solo) –виконання в соло. Якщо хоча б на одній звуковій доріжці ввімкнено режим «соло», то під час відтворення звучатимуть лише доріжки з цим активованим режимом;

2) “M” (від англ. mute) – приглушення. Доріжки в цьому режимі не відтворюються, поки його не буде вимкнено;

3) “R” (від англ. record) – записування. У разі, якщо користувач вибере однойменний режим програвача, то процес виконуватиметься лише на доріжках в стані “R”.

Також на цій панелі присутні кнопка для видалення доріжки/кліпу, а також інші операції, що сховані під кнопкою «трикрапка», як-от розділення треку, накладання ефектів оброблення, вибір файлу для імпортування в проєкт.

4.6 Висновки

У цьому розділі було вибрано архітектуру розроблюваної ПС, спроєктовано та надано основні програмні класи, структуру сховища даних багатодоріжкових проєктів, описано застосовані шаблони, а також графічний інтерфейс користувача.

5 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

5.1 Вибір технологій розроблення та особливості реалізації ПС

Для реалізації ПС “Audiora” було вирішено використати такі інструментальні засоби розроблення ПЗ, як-от:

а) Java – це строго типізована об’єктно-орієнтована мова програмування. Зазвичай програми мовою Java компілюють до байт-коду, який віртуальна машина інтерпретує для виконання на тій чи тій платформі [15];

б) JDK (англ. Java Development Kit) – це безкоштовний набір інструментів для розроблення застосунків вищезазначеною мовою, який містить стандартні бібліотеки, документацію, допоміжні службові програми, компілятор тощо [16];

в) Android SDK (англ. Software Development Kit) – це також набір для розроблення застосунків, конкретно для мобільної операційної системи Android. Дозволяє розробляти ПЗ для різних пристроїв: від тих, які можна носити, до габаритних технічних засобів [17];

г) Android Architecture Components – це ще один набір бібліотек, тільки від компанії Google, який дозволяє створювати ПЗ, враховуючи підтримку життєвого циклу користувальницького інтерфейсу та оброблення збереження даних (Lifecycle-Aware Components, Lifecycle, LifecycleOwner, LiveData, ViewModel та Navigation) [10];

г) XML (англ. eXtensible Markup Language) – це стандарт побудови мов розмічення даних з ієрархічною структурою. Його застосовують для приймання/передавання даних між різними застосунками.

Особливість XML полягає в тому, що розмітка в документах не є фіксованою. Розробник ПЗ створює власну згідно з потребами предметної області та обмежений лише синтаксичними правилами мови [18].

Цю технологію використано для зберігання даних багатодоріжкових проєктів у сховищі даних. Приклад такого файлу наведено в Додатку А.

д) Android Studio – це інтегроване середовище, яке надає інструменти для розроблення Android-застосувань для різних пристроїв, передбачених Android SDK [19].

5.2 Сценарії та результати тестування ПС

На цьому етапі розроблення програмної системи паралельного цифрового оброблення звукових сигналів необхідно провести тестування для визначення можливих помилок та їх виключення. Тестування ПЗ передбачає перевірку, чи відповідає реальна поведінка програм очікуваній на дискретній множині вибраних у певний спосіб тестів.

Тестова ситуація (англ. test case) – це кінцева множина входів та передбачувана відповідна множина виходів для перевірки ПС щодо дотримання зазначеної специфікації вимог.

Тестування на основі варіантів використання передбачає перевірку програмного модуля, утвореного на основі виконання деякої функції системи. Водночас тестові приклади (ТП) складають для перевірки коректного виконання пунктів сценаріїв ВВ, як основного, так і альтернативного [20].

Далі наведено один зі складених сценаріїв тестування варіанту використання «Створення нового проєкту».

Таблиця 5.1 – Сценарій тестування ВВ «Створення нового проєкту»

№ з/п	Опис тестового прикладу
1	Джерело ТП: ВВ «Створення нового проєкту».
	Вхідний стан системи: почати тестування з головного меню ПС.
	Вхідні дані: натискання кнопки «Створити проєкт».

Продовження таблиці 5.1

№ з/п	Опис тестового прикладу
1	Очікувані результати: відкривається вікно «Audiora – Створення проєкту» з одним активним та одним неактивним полем уведення основної інформації про проєкт (перше – назва, друге – розташування) та 2-ма кнопками – «Завершити» і «Скасувати».
2	Джерело ТП: ВВ «Створення нового проєкту», розширення 3a.
	Вхідний стан системи: почати тестування з вікна створення багатодоріжкового проєкту в початковому стані.
	Вхідні дані: натискання кнопки «Завершити».
	Очікувані результати: відкривається вікно редагування доріжок проєкту, де маємо проєкт із 6-ма порожніми стерео-доріжками, налаштованих на частоту дискретизації 48 кГц та розрядність 16 біт, іменованих підряд як «Доріжка 1», «Доріжка 2», …, «Доріжка 6». Зверху вікна – напис “Audiora – Untitled_ Project_N”, де N – порядковий номер проєкту серед наявних таких з аналогічною назвою. Файл проєкту розташовано в теці “/Audiora/Projects/”.
3	Джерело ТП: ВВ «Створення нового проєкту», розширення 3b.
	Вхідний стан системи: почати тестування з вікна створення багатодоріжкового проєкту в початковому стані.
	Вхідні дані: назва проєкту – “Test3_Project”, каталог розташування – “/Test3_ Dir/”, частота дискретизації – 44,1 кГц, режим каналів – моно, натискання кнопки «Завершити».
	Очікувані результати: відкривається вікно редагування доріжок проєкту, де маємо проєкт із 6-ма порожніми моно-доріжками, налаштованих на частоту дискретизації 44,1 кГц та розрядність 16 біт, іменованих підряд як «Доріжка 1», «Доріжка 2», …, «Доріжка 6». Зверху вікна – напис “Audiora – Test3_Project”. Файл проєкту розташовано в теці “/Test3_Dir/”.

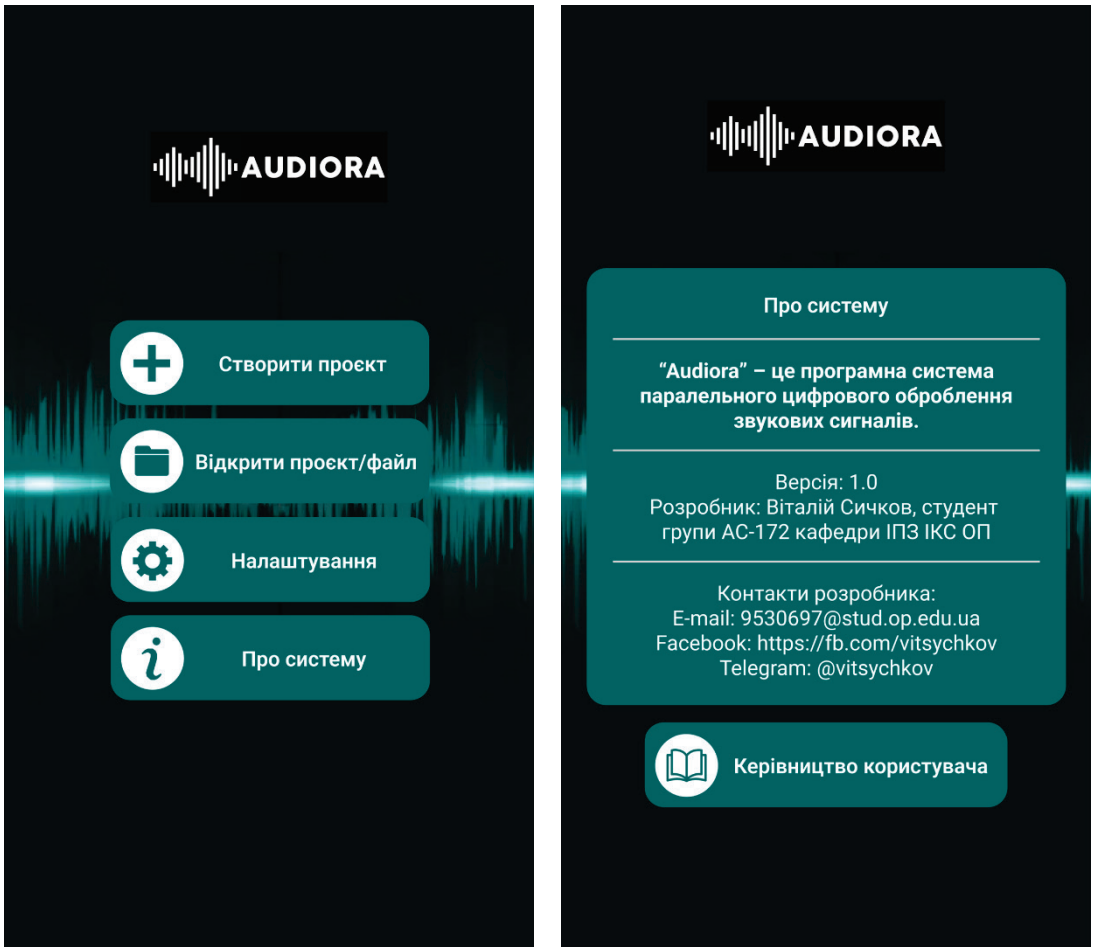
Продовження таблиці 5.1

№ з/п	Опис тестового прикладу
4	Джерело ТП: ВВ «Створення нового проєкту», розширення 4a.
	Вхідний стан системи: почати тестування з вікна створення багатодоріжкового проєкту після зазначення користувачем назви проєкту “Test4-1” та директорії “/Test4Dir/”.
	Вхідні дані: назва проєкту – “Test4-2”, натискання кнопки «Завершити».
	Очікувані результати: відкривається вікно редагування доріжок проєкту, де маємо проєкт із 6-ма порожніми стерео-доріжками, налаштованих на частоту дискретизації 48 кГц та розрядність 16 біт, іменованих підряд як «Доріжка 1», «Доріжка 2», …, «Доріжка 6». Зверху вікна – напис “Audiora – Test4-2”. Файл проєкту розташовано в теці “/Test4Dir/”.
5	Джерело ТП: ВВ «Створення нового проєкту», розширення 4b.
	Вхідний стан системи: почати тестування з вікна створення багатодоріжкового проєкту в початковому стані.
	Вхідні дані: натискання кнопок «Скасувати» → «Так».
	Очікувані результати: перехід до головного меню системи.

Отже, внаслідок тестування функціональності ПС було встановлено, що всі дійсні результати її роботи збіглися з очікуваними, тобто всі пред’явлені вимоги було успішно виконано.

5.3 Приклад використання системи

На рис. 5.1 зображено головне меню ПС, що має чотири кнопки для створення багатодоріжкового проєкту, відкриття такого проєкту або аудіофайлу, переходу до налаштувань системи та допоміжної інформації про неї.



а) *б)*

Рисунок 5.1 – Головне меню (а) та вікно «Про систему» (б) ПС “Audiora”

На рис. 5.2 зображено процес створення нового багатоканального проєкту у відповідному вікні програмної системи. Тут користувач зазначає необхідну інформацію про проєкт: уводить його назву, вказує місце для розташування файлу проєкту в сховищі даних та вибирає такі параметри, як частота дискретизації сигналів, їх розрядність та режим каналів звукових доріжок.

Також передбачено дві кнопки:

а) для скасування створення проєкту. Після натиснення на неї з’являється попереджувальне діалогове вікно з запитом підтвердити або відхилити цю дію. За першого випадку користувач повернеться до головного меню системи, за другого – залишиться у поточному вікні;

б) для підтвердження створення проєкту. Після натиснення на цю кнопку система виконає запис файлу до сховища даних у форматі XML та відобразить

порожній проєкт із шістьма звуковими доріжками та виділеною першою доріжкою у вікні редагування системи.

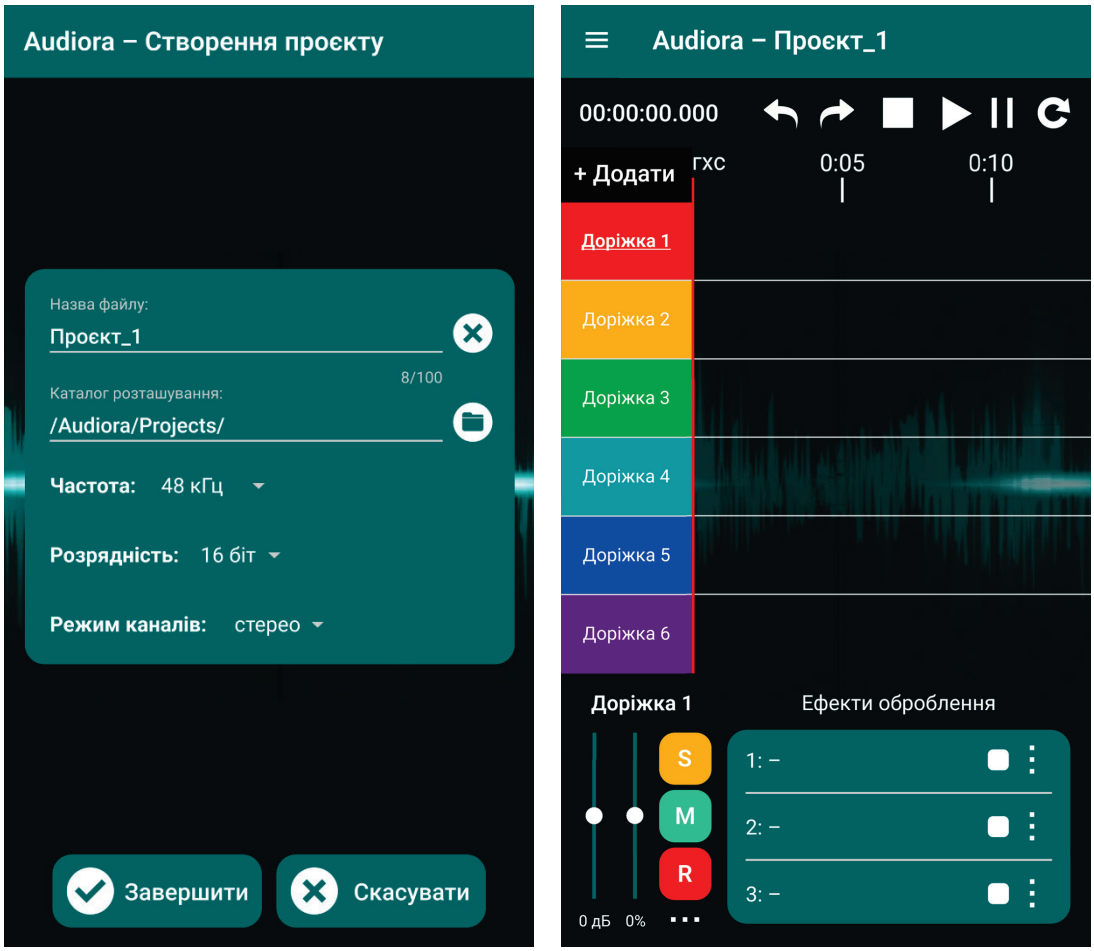


Рисунок 5.2 – Вікно створення нового багатодоріжкового проєкту (а) та його редагування (б) в ПС “Audiora”

Далі користувач може, наприклад, імпортувати аудіофайл до вибраної доріжки на поточній позиції курсора програвача. Для цього йому необхідно натиснути на кнопку з трьома крапками в додатковій нижній панелі поточного вікна системи та вибрати «Додати файл». Відкриється вікно з файловим провідником, де він указує на потрібний файл та підтверджує свій вибір.

Після цього користувач може додати деякі звукові ефекти на цю доріжку, наприклад, у такій послідовності: параметричний еквалайзер, компресор, ревербератор, затримка, нормалізатор (див. рис. 5.3).

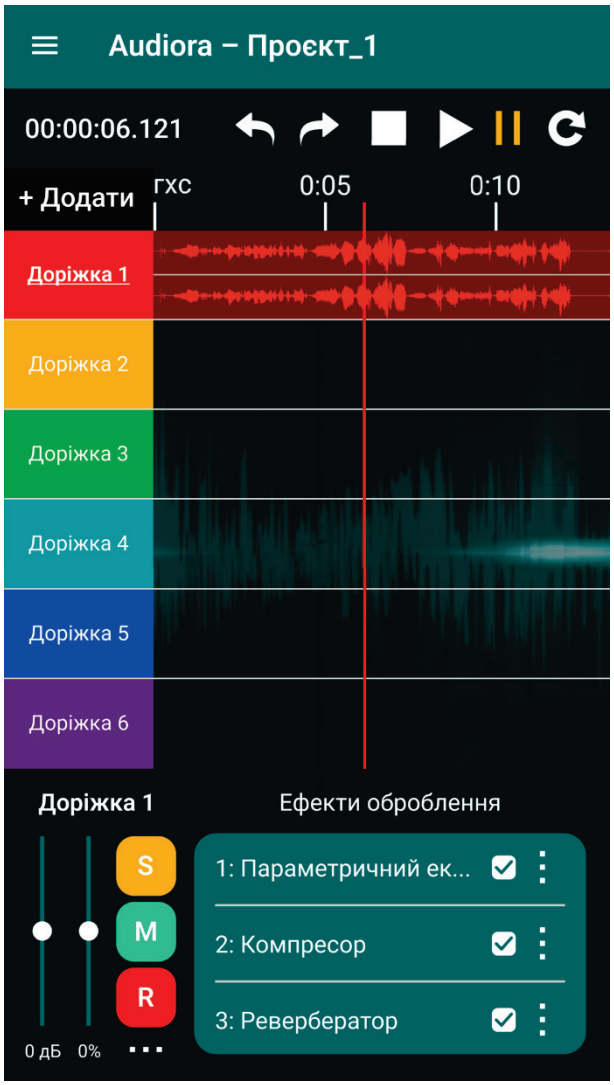


Рисунок 5.3 – Накладання ефектів оброблення на імпортовану звукову доріжку в розробленій програмній системі

Водночас у нього є доступ до паралельного налаштування всіх ефектів, він має змогу вмикати/вимикати їх у ланцюжку оброблення, а файли вихідних сигналів не змінюються, згідно з запропонованим методом у другому розділі роботи.

5.4 Висновки

У цьому розділі було вибрано необхідні засоби для розроблення ПС, описано деякі особливості її реалізації, наведено один зі сценаріїв тестування отриманої програмної розробки, а також приклад її використання.

6 ВИЗНАЧЕННЯ ХАРАКТЕРИСТИК ПРОГРАМНОЇ СИСТЕМИ

6.1 Формулювання гіпотези та метрик щодо використання ПС

Гіпотеза – це наукове припущення, яке висувають для ймовірного розв'язання певної проблеми чи правдоподібного пояснення певних явищ, що потребує подальшої експериментальної перевірки.

Метрика – це кількісний або якісний показник, який відображає ту чи ту характеристику програмного продукту. Кількісні показники легше відстежувати, тому зазвичай їх використовують частіше.

Сформулюємо твердження, узгоджене з метою цієї роботи, у якій спрогнозуємо наслідок подальшого експерименту: впровадження запропонованого методу оброблення звукових сигналів сприятиме зменшенню кількості операцій, необхідних для накладання ефектів на них, та часу, потрібного для коригування певного ефекту, порівняно з традиційним підходом.

Метрики, використані для перевірки цієї гіпотези:

а) кількість виконуваних операцій для застосування кінцевого варіанту певного ефекту оброблення, яку вимірюємо в операціях (оп.);

б) час редагування цього ефекту оброблення, який вимірюємо в секундах (с).

6.2 Опис методики виконання дослідження властивостей системи

Для дослідження вибраних характеристик розробленої ПС було розроблено і використано методику, яка дозволяє оцінити та порівняти ефективність застосування ефектів оброблення на звукові доріжки, яку визначатимемо за допомогою вимірювання значень запропонованих метрик.

Для експерименту було запрошено іншого користувача (не автора цієї роботи) з певним досвідом роботи в аналогічних програмних системах. Спочатку його було ознайомлено з функціональними можливостями ПС "Audiora". Після цього створено новий проєкт, до якого додано п'ять ідентичних доріжок, тобто таких, які містять по одному кліпу тривалістю 10 хвилин (від нульової позиції курсора програвача до

10-хвилинної), із посиланням на один і той же файл звукового сигналу, та кожну з яких попередньо і почергово оброблено 4-ма звуковими ефектами з однаковими параметрами за замовчуванням у тій же послідовності: еквалайзер, компресор, ревербератор та нормалізатор.

Методика передбачає проведення користувачем 20 (двадцяти) перевірок, тобто для кожного ефекту на кожній доріжці, використовуючи обидва методи, по 5 замірів часу, з яких знаходимо середнє арифметичне, – усього виходить 200 замірів. Їх можна проводити, наприклад, у парі, де один виконує системні операції, не відволікаючись під час процесу, а інший допомагає вимірювати необхідні показники. Операціями вважатимемо такі для додавання, редагування/видалення та скасування ефектів оброблення.

6.3 Аналіз отриманих результатів дослідження

У табл. 6.1 наведено дані, отримані експериментальним шляхом, де числові значення відповідають оцінкам кількості операцій для застосування кінцевого звукового ефекту (стовпець А) та часу коригування певного ефекту (стовпець Б) з використанням послідовного (ПМ1) і паралельного (ПМ2) методів для кожного з 20-ти ефектів у порядку від останнього до першого застосованого ефекту.

Таблиця 6.1 – Отримані дані внаслідок проведення експерименту

№ з/п	ПМ1		ПМ2		№ з/п	ПМ1		ПМ2	
	А, оп.	Б, с	А, оп.	Б, с		А, оп.	Б, с	А, оп.	Б, с
1	2	5,4	1	3,9	6	12	36,9	1	4,9
2	4	11,7	1	4,6	7	14	43,6	1	4,0
3	6	17,8	1	4,2	8	16	51,0	1	4,6
4	8	24,3	1	3,8	9	18	56,7	1	5,1
5	10	30,9	1	4,1	10	20	63,1	1	4,3

Продовження таблиці 6.1

№ з/п	ПМ1		ПМ2		№ з/п	ПМ1		ПМ2	
	А, оп.	Б, с	А, оп.	Б, с		А, оп.	Б, с	А, оп.	Б, с
11	22	69,3	1	4,6	16	32	102,7	1	4,3
12	24	76,2	1	3,8	17	34	109,5	1	5,2
13	26	82,6	1	4,3	18	36	115,8	1	4,5
14	28	89,4	1	4,1	19	38	122,0	1	4,8
15	30	96,1	1	4,8	20	40	128,4	1	4,4

Для наочності та облегшення аналізу результатів експерименту за отриманими даними побудуємо два графіки (див. рис. 6.1):

а) залежності відношення кількостей операцій для застосування одного кінцевого варіанту певного ефекту оброблення для обох методів від порядкового номеру застосування цього ефекту, яка відповідає функції $f = Q(n)$;

б) залежності відношення часів коригування цього ефекту для обох методів від порядкового номеру його застосування, яка відповідає функції $g = T(n)$.

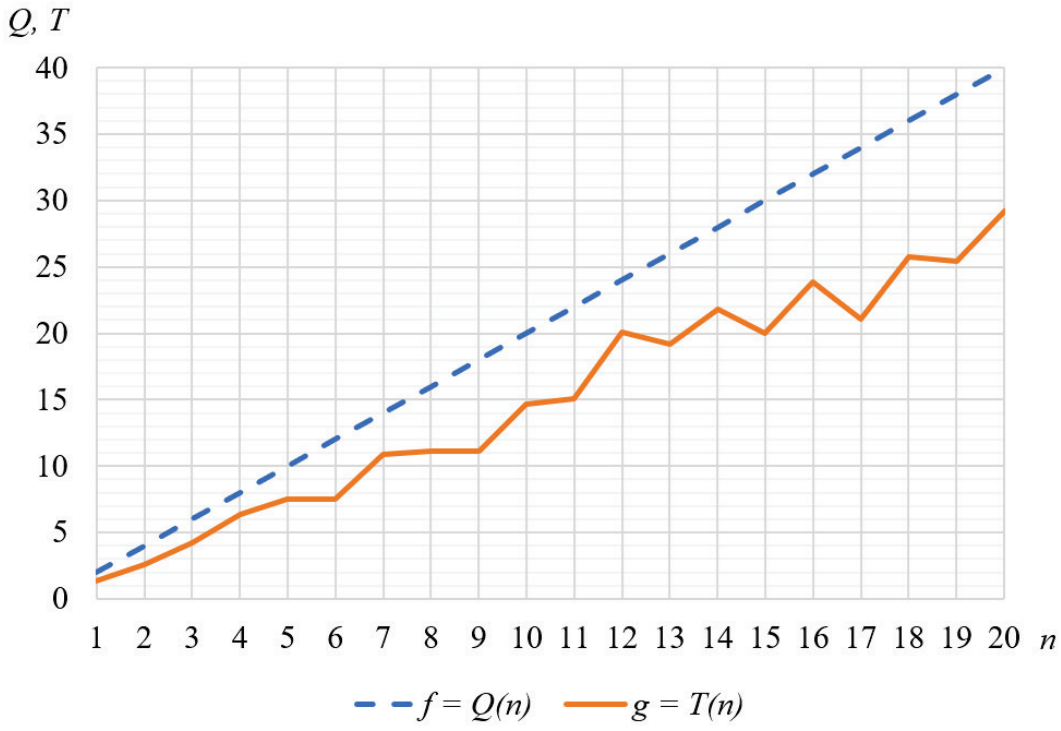


Рисунок 6.1 – Результівні графіки функцій f і g

Отже, можна дійти втішного висновку, що гіпотеза проведеного дослідження підтверджується. Застосування паралельного методу оброблення звукових сигналів дійсно сприятиме зменшенню кількості операцій, необхідних користувачеві для накладання ефектів, та часу, необхідного для редагування певного ефекту, порівняно з послідовним методом.

Варто зазначити, що максимальний показник виграшу Q в 40 разів та T в 29,18 разів було отримано за найбільшого значення порядкового номеру застосування ефекту (n = 20). Водночас виграш практично відсутній при n = 1. Тобто ефективність запропонованого підходу зростає зі збільшенням кількості ефектів, які потребують додаткового налаштування, віддаленості від операцій їх додавання, а також можливо загалом від кількості застосованих доріжок та ефектів у проєктах, що є доволі поширеними випадками під час створення сучасної насиченої аудіопродукції.

6.4 Висновки

У цьому розділі було описано результати дослідження властивостей програмної розробки. Спочатку було сформовано гіпотезу, яку слід було перевірити, та визначено метрики, які були для цього застосовані. Далі було описано методику виконання досліджень характеристик системи, надано отримані результати та зроблено висновки щодо поставленої гіпотези.

ВИСНОВКИ

Унаслідок виконання кваліфікаційної роботи магістра було спроєктовано і розроблено програмну систему паралельного цифрового оброблення “Audiora”, яка дозволяє скоротити кількість виконуваних операцій для накладання кінцевих ефектів на звукові доріжки під час зведення останніх завдяки реалізації паралельного методу оброблення сигналів.

На шляху досягнення поставленої мети було розв’язано такі задачі:

а) вивчено проблеми користувачів та можливості наявних мобільних систем для цифрового оброблення звукових сигналів;

б) виявлено та проаналізовано вимоги до розроблюваної ПС;

в) досліджено наявний послідовний підхід до розв’язання поставленої задачі та впроваджено власний паралельний метод;

г) розроблено структуру сховища даних для зберігання проєктів;

г) розроблено алгоритми для виконання основних функцій системи;

д) виконано програмну реалізацію інтерфейсу користувача;

е) визначено та експериментально оцінено певні характеристики системи.

Упровадження розробленого програмного забезпечення дозволить підвищити продуктивність праці користувачів завдяки зменшенню кількості необхідних операцій для накладання кінцевих ефектів на звукові сигнали, де маємо виграш, значення якого відрізняється від порядкового номеру застосування деякого ефекту (різниці віддаленості операції його додавання від поточного стану системи) в 2 рази, а також зменшенню часу на коригування ефектів.

У подальшому до розробленої системи можуть бути внесені покращення та додаткові можливості, як-от, наприклад, впровадження окремої результівної мікс-доріжки, яка поєднує усі оброблені сигнали в один, який можна одразу налаштувати на робочій ділянці проєкту (цей етап називають «мастерінгом»), додавання ефектів не тільки на доріжки, а на окремі кліпи, реалізація інших ефектів, тон-генерація, а також виправлено деякі її недоліки та продовжено дослідження в області ефективних алгоритмів та паралельних обчислень.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сичков В. С. Дослідження мобільних застосунків оброблення звукових сигналів. *Матеріали Дванадцятої Міжнародної наукової конференції студентів та молодих учених «Сучасні інформаційні технології - 2022» «Modern Information Technology - 2022» (19-20 травня 2022 р., м. Одеса)* / МОН України; Державний університет «Одеська політехніка»; Ін-т комп'ют. систем. Одеса: Наука і техніка, 2022. С. 13-14.

2. Вологдин Э. И. Методы и алгоритмы обработки звуковых сигналов: Курс лекций. СПб. : СПбГУТ, 2009. 96 с.

3. Дизайн-мислення: як застосовувати метод на практиці / Анна Каравайна, Blog Author. IT школа Beetroot Academy: інтенсивні IT-курси в дружній атмосфері. URL: https://beetroot.academy/blog/general/dizajn-mislennya-yak-zastosovuvati-metod-na-praktitsi (дата звернення: 31.08.2022).

4. Гогунський, В. Д., Лук'янов, Д. В., Колесніков, О. Є., & Шерстюк, О. І. (2018). The use of the "design-thinking" and "seven hats" methods at the project initiation and planning stage. *Herald of Advanced Information Technology, 1*(1), 62-68. Retrieved from http://hait.ccs.od.ua/index.php/journal/article/view/88.

5. SuperSound. Music Audio Editor, MP3 Cutter / Video Screen Recorder, Voice Audio Editor, Cut MP3. URL: https://play.google.com/store/apps/details?id=com.tianxing-jian.supersound (дата звернення: 05.09.2022).

6. WaveEditor Record & Edit Audio / Sound-Base Audio, LLC. URL: https://play.google.com/store/apps/details?id=io.sbaud.wavstudio (дата звернення: 05.09.2022).

7. Lexis Audio Editor / pamsys. URL: https://www.lexisaudioeditor.com (дата звернення: 05.09.2022).

8. Гітарні ефекти: хорус, фленджер, фазер, леслі, тремоло. URL: https://hitonline.ua/ua/articles/gitari-i-gitarnoe-oborudovanie/gitarnie-effekti--horus--flendzher--fazer--lesli--tremolo.html (дата звернення: 07.09.2022).

9. Кавіцька В. С., Любченко В. В. Конспект лекцій з дисципліни «Основи програмної інженерії» для студентів першого рівня вищої освіти (бакалаврів) спеціальності 121 «Інженерія програмного забезпечення». Одеса: ОНПУ, 2017. 90 с.

10. Guide to app architecture / Android Developers. URL: https://developer.android.com/topic/architecture/ (дата звернення: 11.10.2022).

11. Ларман К. Применение UML и шаблонов проектирования / пер. с англ. и ред. А. Ю. Шелестова. 2-е изд. М. : Издательский дом «Вильямс», 2002. 624 с.

12. Каталог патернів проєктування / Refactoring.Guru. URL: https://refactoring.guru/uk/design-patterns/catalog (дата звернення: 15.10.2022).

13. What is an Entity Relationship Diagram (ERD)? / LucidChart: Where seeing becomes doing. URL: https://www.lucidchart.com/pages/er-diagrams (дата звернення: 18.10.2022).

14. Figma: the collaborative interface design tool / Adobe. URL: https://www.figma.com (дата звернення: 21.10.2022).

15. What is Java technology and why do I need it? / Oracle. URL: https://www.java.com/download/help/whatis_java.html (дата звернення: 25.10.2022).

16. Java Downloads / Oracle. URL: https://www.oracle.com/java/technologies/downloads/ (дата звернення: 25.10.2022).

17. SDK Platform Tools release notes / Android Developers. URL: https://developer.android.com/studio/releases/platform-tools (дата звернення: 25.10.2022).

18. Extensible Markup Language (XML) 1.0 (Third Edition) / World Wide Web Consortium (W3C). URL: https://www.w3.org/TR/2004/REC-xml-20040204/ (дата звернення: 26.10.2022).

19. Download Android Studio & App Tools / Android Developers. URL: https://developer.android.com/studio (дата звернення: 26.10.2022).

20. О. Б. Кунгурцев. Конспект лекцій з дисципліни «Конструювання програмного забезпечення» для студентів інституту комп’ютерних систем першого рівеня вищої освіти (бакалавр) з базової підготовки за напрямком 6.050103 «Програмна інженерія». Одеса: ОНПУ, 2017. 66 с.

ДОДАТОК А

Лістинг програмної системи

А.1 Клас CreateProjectActivity

```
package com.svitstudio.audiora.createProject;

import android.os.Bundle;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.ViewModelProvider;

import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import com.svitstudio.audiora.R;

import java.util.Objects;

public class CreateProjectActivity extends AppCompatActivity
{
    private CreateProjectViewModel vm;
    private EditText title_et, source_et, sample_rate_et, bit_depth_et, ch_mode_et;
    private ImageButton clear_title_btn, clear_source_btn,
                    clear_sample_rate_btn, clear_ bit_depth_btn, clear_ch_mode_btn;
    private Button complete_btn, cancel_btn;

    @Override
    protected void onCreate(Bundle bundle)
    {
        setTheme(R.style.AppTheme);
        super.onCreate(bundle);
        setContentView(R.layout.activity_create_project);
        observeViewModelData();
        initialize();
    }

    public void onBackPressed()
    {
        AlertDialog.Builder builder = new
AlertDialog.Builder(CreateProjectActivity.this);
        builder.setMessage(R.string.cancel_project_creation_alert)
                .setCancelable(false)
                .setPositiveButton(R.string.yes, (dialog, which) -> finish())
                .setNegativeButton(R.string.no, (dialog, which) -> dialog.cancel());
```

```
        AlertDialog alert_dialog = builder.create();
        alert_dialog.setTitle(R.string.attention);
        alert_dialog.show();
    }
    protected CreateProjectViewModel getViewModel() { return vm; }

    private void initialize()
    {
        ActionBar action_bar = getSupportActionBar();
        vm = new ViewModelProvider(this,
                new CreateProjectViewModelFactory(getApplication()))
                .get(CreateProjectViewModel.class);
        vm.setActionBar(action_bar);
        vm.getActionBarTitle().observe(this,
Objects.requireNonNull(action_bar)::setTitle);

        title_et = view.findViewById(R.id.title_et);
        source_et = view.findViewById(R.id.source_et);
        sample_rate_et = view.findViewById(R.id.sample_rate_et);
        bit_depth_et = view.findViewById(R.id.bit_depth_et);
        ch_mode_et = view.findViewById(R.id.ch_mode_et);

        clear_title_btn = view.findViewById(R.id.clear_title_btn);
        clear_source_btn = view.findViewById(R.id.clear_source_btn);
        clear_sample_rate_btn = view.findViewById(R.id.sample_rate_btn);
        clear_bit_depth_btn = view.findViewById(R.id.clear_bit_depth_btn);
        clear_ch_mode_btn = view.findViewById(R.id.clear_ch_mode_btn);
        complete_btn = view.findViewById(R.id.complete_btn);
        cancel_btn = view.findViewById(R.id.cancel_btn);
    }

    private void setListeners()
    {
        clear_title_btn.setOnClickListener(v -> vm.clearTitle());
        clear_source_btn.setOnClickListener(v -> vm.clearSource());
        clear_sample_rate_btn.setOnClickListener(v -> vm.clearSampleRate());
        clear_bit_depth_btn.setOnClickListener(v -> vm.clearBitDepth());
        clear_ch_mode_btn.setOnClickListener(v -> vm.clearChMode());
        complete_btn.setOnClickListener(v -> vm.createProject());
        cancel_btn.setOnClickListener(v -> activity.onBackPressed());
    }

    private void setDataToViewModel()
    {
        vm.setTitle(title_et.getText().toString().trim());
        vm.setSource(source_et.getText().toString().trim());
        vm.setSampleRate(sample_rate_et.getText().toString().trim());
        vm.setBitDepth(bit_depth_et.getText().toString().trim());
        vm.setChMode(ch_mode_et.getText().toString().trim());
    }

    private void observeViewModelData()
    {
        vm.getTitle().observe(this, s -> title_et.setText(s));
        vm.getSource().observe(this, s -> source_et.setText(s));
```

```
        vm.getSampleRate().observe(this, s -> sample_rate_et.setText(s));
        vm.getBitDepth().observe(this, s -> bit_depth_et.setText(s));
        vm.getChMode().observe(this, s -> ch_mode_et.setText(s));
    }
}
```

A.2 Клас CreateProjectViewModelFactory

```
package com.svitstudio.audiora.createProject;

import android.app.Application;

import androidx.annotation.NonNull;
import androidx.lifecycle.ViewModel;
import androidx.lifecycle.ViewModelProvider;

public class CreateProjectViewModelFactory extends ViewModelProvider.Android-
ViewModelFactory
{
    private Application application;

    public CreateProjectViewModelFactory(@NonNull Application application)
    {
        super(application);
        this.application = application;
    }

    @NonNull
    @SuppressWarnings("unchecked")
    @Override
    public <T extends ViewModel> T create(@NonNull Class<T> modelClass)
    {
        return (T) new CreateProjectViewModel(application);
    }
}
```

A.3 Клас CreateProjectViewModel

```
package com.svitstudio.audiora.createProject;

import android.app.Application;
import android.content.res.Resources;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBar;
import androidx.lifecycle.AndroidViewModel;
import androidx.lifecycle.MutableLiveData;

import com.svitstudio.audiora.R;
import com.svitstudio.audiora.model.Track;
```

```
import com.svitstudio.audiora.model.Project;

import java.util.Objects;

public class CreateProjectViewModel extends AndroidViewModel
{
    private Application application;
    private MutableLiveData<String> action_bar_title_ld, title_ld, source_ld;
    private MutableLiveData<Integer> sample_rate_ld, bit_depth_ld;
    private MutableLiveData<Boolean> ch_mode_ld;
    private Project project;

    public CreateProjectViewModel(@NonNull Application application)
    {
        super(application);
        this.application = application;
        initialize();
    }

    protected void clearTitle() { title_ld.setValue(""); }
    protected void clearSource() { source_ld.setValue(""); }
    protected void clearSampleRate() { sample_rate_ld.setValue(0); }
    protected void clearBitDepth() { bit_depth_ld.setValue(0); }
    protected void clearChMode () { ch_mode_ld.setValue(false); }

    protected void setActionBar(ActionBar action_bar)
    {
        if (action_bar != null)
        {
            Resources resources = getApplication().getResources();
            setActionBarTitle(resources.getString(R.string.dash_delimeter,
action_bar.getTitle(), resources.getString(R.string.creation_project)));
        }
    }
    protected void setActionBarTitle(String action_bar_title) {
action_bar_title_ld.setValue(action_bar_title); }
    protected void setTitle(String title) { title_ld.setValue(title); }
    protected void setSource(String source) { source_ld.setValue(source); }
    protected void setSampleRate(int sample_rate) {
sample_rate_ld.setValue(sample_rate); }
    protected void setBitDepth(int bit_depth) { bit_depth_ld.setValue(bit_depth);
}
    protected void setChMode(boolean ch_mode) { ch_mode_ld.setValue(ch_mode); }

    protected MutableLiveData<String> getActionBarTitle() { return
action_bar_title_ld; }
    protected MutableLiveData<String> getTitle() { return title_ld; }
    protected MutableLiveData<String> getSource() { return source_ld; }
    protected MutableLiveData<Integer> getSampleRate() { return sample_rate_ld; }
    protected MutableLiveData<Integer> getBitDepth() { return bit_depth_ld; }
    protected MutableLiveData<Boolean> getChMode() { return ch_mode_ld; }

    private void initialize()
    {
        action_bar_title_ld = new MutableLiveData<>();
```

```
        title_ld = new MutableLiveData<>();
        source_ld = new MutableLiveData<>();
        sample_rate_ld = new MutableLiveData<>();
        bit_depth_ld = new MutableLiveData<>();
        ch_mode_ld = new MutableLiveData<>();
    }
}
```

A.4 Клас Project

```
package com.svitstudio.audiora.model;

import java.io.Serializable;
import java.util.ArrayList;
public class Project implements Serializable
{
    private String title;
    private String source;
    private int sample_rate;
    private int bit_depth;
    private boolean ch_mode;

    private ArrayList<Track> tracks;
    private Playback playback;

    public Project(String title, String source, int sample_rate, int bit_depth,
boolean ch_mode)
    {
        this.title = title;
        this.source = source;
        this.sample_rate = sample_rate;
        this.bit_depth = bit_depth;
        this.ch_mode = ch_mode;
        playback = new Playback();
    }

    protected String getTitle()
    {
        return title;
    }
    protected void setTitle(String title)
    {
        this.title = title;
    }

    protected String getSource()
    {
        return source;
    }

    protected void setSource(String source)
    {
        this.source = source;
```

```
    }

    protected int getSampleRate()
    {
        return sample_rate;
    }

    protected void setSampleRate(int sample_rate)
    {
        this.sample_rate = sample_rate;
    }

    protected int getBitDepth()
    {
        return bit_depth;
    }

    protected void setBitDepth(int bit_depth)
    {
        this.bit_depth = bit_depth;
    }

    protected boolean addTrack(Track track)
    {
              if (tracks.size() < 128)
              {
            tracks.add(track);
            return true;
        }
        return false;
    }
    protected boolean removeTrack(int id)
    {
        for (int i = 0; i < tracks.size(); i++)
        {
            Track track = tracks.get(i);
            if (track.getId() == id)
            {
                tracks.remove(track);
                return true;
            }
        }
        return false;
    }
}
```

A.5 Клас Track

```
package com.svitstudio.audiora.model;

import java.io.Serializable;
import java.util.ArrayList;
```

```
public class Track implements Serializable
{
    private int id;
    private String name = "Доріжка_";
    private int volume = 0;
    private int balance = 0;
    private boolean isMuted = false;
    private boolean isSolo = false;
    private boolean isRecord = false;
    private ArrayList<Effect> effects;
    private ArrayList<Clip> clips;

    public Track(int id) {
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getVolume() {
        return volume;
    }

    public void setVolume(int volume) {
        this.volume = volume;
    }

    public int getBalance() {
        return balance;
    }

    public void setBalance(int balance) {
        this.balance = balance;
    }

    public boolean isMuted() {
        return isMuted;
    }

    public void setMuted(boolean muted) {
        isMuted = muted;
    }
```

```
public boolean isSolo() {
    return isSolo;
}

public void setSolo(boolean solo) {
    isSolo = solo;
}

public boolean isRecord() {
    return isRecord;
}

public void setRecord(boolean record) {
    isRecord = record;
}

public ArrayList<Effect> getEffects() {
    return effects;
}

public boolean addEffect(Effect effect)
{
    if (effects.size() < 16)
    {
        effects.add(effect);
        return true;
    }
    return false;
}

public boolean removeEffect(int id)
{
    for (int i = 0; i < effects.size(); i++)
    {
        Effect effect = effects.get(i);
        if (effect.getId() == id)
        {
            effects.remove(effect);
            return true;
        }
    }
    return false;
}

public ArrayList<Clip> getClips()
{
    return clips;
}

public void addClip(Clip clip)
{
    clips.add(clip);
}
```

```
    public boolean removeClip(int id)
    {
        for (int i = 0; i < clips.size(); i++)
        {
            Clip clip = clips.get(i);
            if (clip.getId() == id)
            {
                effects.remove(clip);
                return true;
            }
        }
        return false;
    }
}
```

A.6 Приклад файлу проєкту “Project.xml”

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <id>1</id>
    <title>Project</title>
    <sampleRate>48000</sampleRate>
    <bitDepth>16</bitDepth>
    <chMode>true</chMode>
    <playback>
        <id>1</id>
        <curPos>499462</curPos>
        <speed>1</speed>
    </playback>
    <tracks>
        <track>
            <id>1</id>
            <name>Lead Vocals</name>
            <volume>2.5</volume>
            <balance>0</balance>
            <state>
                <mute>false</mute>
                <solo>true</solo>
                <record>false</record>
            </state>
            <clips>
                <clip>
                    <id>1</id>
                    <signalId>1</signalId>
                    <cutStart>498047</cutStart>
                    <posStart>0</posStart>
                    <duration>3016272</duration>
                    <volume>-1.5</volume>
                </clip>
                <clip>
                    <id>2</id>
                    <signalId>1</signalId>
```

```
            <cutStart>3754278</cutStart>
            <posStart>3030283</posStart>
            <duration>7219105</duration>
            <volume>0</volume>
        </clip>
    </clips>
    <effects>
        <effect>
            <id>1</id>
            <name>Параметричний еквалайзер</name>
            <dry>0</dry>
            <wet>100</wet>
            <inputGain>0</inputGain>
            <outputGain>0</outputGain>
            <active>true</active>
            <params>
                <shelfs>
                    <lowShelf>
                        <freq>40</freq>
                        <gain>0</gain>
                        <q_coef>2</q_coef>
                        <active>false</active>
                    </lowShelf>
                    <highShelf>
                        <freq>8000</freq>
                        <gain>3</gain>
                        <q_coef>1.5</q_coef>
                        <active>true</active>
                    </highShelf>
                </shelfs>
                <cuts>
                    <lowCut>
                        <freq>150</freq>
                        <gain>24</gain>
                        <active>true</active>
                    </lowCut>
                    <highCut>
                        <freq>18000</freq>
                        <gain>24</gain>
                        <active>false</active>
                    </highCut>
                </cuts>
                <bands>
                    <first>
                        <freq>50</freq>
                        <gain>0</gain>
                        <q_coef>2</q_coef>
                        <active>false</active>
                    </first>
                    <second>
                        <freq>300</freq>
                        <gain>2</gain>
                        <q_coef>3.5</q_coef>
                        <active>true</active>
                    </second>
```

```
                <third>
                    <freq>1000</freq>
                    <gain>-1</gain>
                    <q_coef>1.5</q_coef>
                    <active>true</active>
                </third>
                <fourth>
                    <freq>3200</freq>
                    <gain>0</gain>
                    <q_coef>2</q_coef>
                    <active>false</active>
                </fourth>
                <fifth>
                    <freq>12800</freq>
                    <gain>0</gain>
                    <q_coef>2</q_coef>
                    <active>false</active>
                </fifth>
            </bands>
        </params>
    </effect>
    <effect>
        <id>2</id>
        <name>Компресор</name>
        <dry>0</dry>
        <wet>100</wet>
        <inputGain>0</inputGain>
        <outputGain>0</outputGain>
        <active>true</active>
        <params>
            <points>
                <point>
                    <input>-100</input>
                    <output>-100</output>
                </point>
                <point>
                    <input>-20</input>
                    <output>-20</output>
                </point>
                <point>
                    <input>0</input>
                    <output>-10</output>
                </point>
            </points>
            <ptime>3</ptime>
            <attack>
                <input>1</input>
                <output>24</output>
            </attack>
            <release>
                <input>100</input>
                <output>100</output>
            </release>
            <rms>true</rms>
            <low_freq>20</low_freq>
```

```
                <high_freq>24000</high_freq>
            </params>
        </effect>
        <effect>
            <id>3</id>
            <name>Ревербератор</name>
            <dry>100</dry>
            <wet>20</wet>
            <inputGain>0</inputGain>
            <outputGain>0</outputGain>
            <active>true</active>
            <params>
                <decay>1200</decay>
                <predelay>10</predelay>
                <diff>2000</diff>
                <lowCut>500</lowCut>
                <highCut>3000</highCut>
            </params>
        </effect>
    </effects>
</track>
<track>
    <id>2</id>
    <name>BGVs</name>
    <volume>-3</volume>
    <balance>0</balance>
    <state>
        <mute>false</mute>
        <solo>true</solo>
        <record>false</record>
    </state>
    <clips>
        <clip>
            <id>1</id>
            <signalId>2</signalId>
            <cutStart>453712</cutStart>
            <posStart>0</posStart>
            <duration>5136184</duration>
            <volume>0</volume>
        </clip>
    </clips>
    <effects>
        <effect>
            <id>1</id>
            <name>Параметричний еквалайзер</name>
            <dry>0</dry>
            <wet>100</wet>
            <inputGain>0</inputGain>
            <outputGain>0</outputGain>
            <active>true</active>
            <params>
                <shelfs>
                    <lowShelf>
                        <freq>40</freq>
                        <gain>0</gain>
```

```
                <q_coef>2</q_coef>
                <active>false</active>
            </lowShelf>
            <highShelf>
                <freq>8000</freq>
                <gain>2</gain>
                <q_coef>1</q_coef>
                <active>true</active>
            </highShelf>
        </shelfs>
        <cuts>
            <lowCut>
                <freq>150</freq>
                <gain>24</gain>
                <active>true</active>
            </lowCut>
            <highCut>
                <freq>18000</freq>
                <gain>24</gain>
                <active>false</active>
            </highCut>
        </cuts>
        <bands>
            <first>
                <freq>50</freq>
                <gain>0</gain>
                <q_coef>2</q_coef>
                <active>false</active>
            </first>
            <second>
                <freq>300</freq>
                <gain>2</gain>
                <q_coef>3.5</q_coef>
                <active>true</active>
            </second>
            <third>
                <freq>1000</freq>
                <gain>0</gain>
                <q_coef>2</q_coef>
                <active>false</active>
            </third>
            <fourth>
                <freq>5000</freq>
                <gain>2</gain>
                <q_coef>3.5</q_coef>
                <active>true</active>
            </fourth>
            <fifth>
                <freq>12800</freq>
                <gain>0</gain>
                <q_coef>2</q_coef>
                <active>false</active>
            </fifth>
        </bands>
    </params>
```

```
    </effect>
    <effect>
        <id>2</id>
        <name>Компресор</name>
        <dry>0</dry>
        <wet>100</wet>
        <inputGain>0</inputGain>
        <outputGain>0</outputGain>
        <active>true</active>
        <params>
            <points>
                <point>
                    <input>-100</input>
                    <output>-100</output>
                </point>
                <point>
                    <input>-20</input>
                    <output>-20</output>
                </point>
                <point>
                    <input>0</input>
                    <output>-10</output>
                </point>
            </points>
            <ptime>3</ptime>
            <attack>
                <input>1</input>
                <output>24</output>
            </attack>
            <release>
                <input>100</input>
                <output>100</output>
            </release>
            <rms>true</rms>
            <low_freq>20</low_freq>
            <high_freq>24000</high_freq>
        </params>
    </effect>
    <effect>
        <id>3</id>
        <name>Ревербератор</name>
        <dry>100</dry>
        <wet>20</wet>
        <inputGain>0</inputGain>
        <outputGain>0</outputGain>
        <active>true</active>
        <params>
            <decay>1500</decay>
            <predelay>0</predelay>
            <diff>2500</diff>
            <lowCut>800</lowCut>
            <highCut>2500</highCut>
        </params>
    </effect>
</effects>
```

```
</track>
<track>
    <id>3</id>
    <name>Adlibs</name>
    <volume>0</volume>
    <balance>10</balance>
    <state>
        <mute>true</mute>
        <solo>false</solo>
        <record>false</record>
    </state>
    <clips>
        <clip>
            <id>1</id>
            <signalId>3</signalId>
            <cutStart>327024</cutStart>
            <posStart>0</posStart>
            <duration>4891910</duration>
            <volume>0</volume>
        </clip>
    </clips>
    <effects>
        <effect>
            <id>1</id>
            <name>Параметричний еквалайзер</name>
            <dry>0</dry>
            <wet>100</wet>
            <inputGain>0</inputGain>
            <outputGain>0</outputGain>
            <active>true</active>
            <params>
                <shelfs>
                    <lowShelf>
                        <freq>40</freq>
                        <gain>0</gain>
                        <q_coef>2</q_coef>
                        <active>false</active>
                    </lowShelf>
                    <highShelf>
                        <freq>8000</freq>
                        <gain>2</gain>
                        <q_coef>2</q_coef>
                        <active>true</active>
                    </highShelf>
                </shelfs>
                <cuts>
                    <lowCut>
                        <freq>150</freq>
                        <gain>24</gain>
                        <active>true</active>
                    </lowCut>
                    <highCut>
                        <freq>18000</freq>
                        <gain>24</gain>
                        <active>false</active>
```

```
            </highCut>
        </cuts>
        <bands>
            <first>
                <freq>50</freq>
                <gain>0</gain>
                <q_coef>2</q_coef>
                <active>false</active>
            </first>
            <second>
                <freq>300</freq>
                <gain>2</gain>
                <q_coef>3.5</q_coef>
                <active>true</active>
            </second>
            <third>
                <freq>1000</freq>
                <gain>0</gain>
                <q_coef>2</q_coef>
                <active>false</active>
            </third>
            <fourth>
                <freq>5000</freq>
                <gain>3</gain>
                <q_coef>2.5</q_coef>
                <active>true</active>
            </fourth>
            <fifth>
                <freq>12800</freq>
                <gain>0</gain>
                <q_coef>2</q_coef>
                <active>false</active>
            </fifth>
        </bands>
    </params>
</effect>
<effect>
    <id>2</id>
    <name>Компресор</name>
    <dry>0</dry>
    <wet>100</wet>
    <inputGain>0</inputGain>
    <outputGain>0</outputGain>
    <active>true</active>
    <params>
        <points>
            <point>
                <input>-100</input>
                <output>-100</output>
            </point>
            <point>
                <input>-20</input>
                <output>-20</output>
            </point>
            <point>
```

```
                        <input>0</input>
                        <output>-10</output>
                    </point>
                </points>
                <ptime>3</ptime>
                <attack>
                    <input>1</input>
                    <output>24</output>
                </attack>
                <release>
                    <input>100</input>
                    <output>100</output>
                </release>
                <rms>true</rms>
                <low_freq>20</low_freq>
                <high_freq>24000</high_freq>
            </params>
        </effect>
        <effect>
            <id>3</id>
            <name>Луна</name>
            <dry>100</dry>
            <wet>75</wet>
            <inputGain>0</inputGain>
            <outputGain>-1.5</outputGain>
            <active>true</active>
            <params>
                <channels>
                    <left>
                        <delay>667</delay>
                        <refl>50</refl>
                        <level>75</level>
                    </left>
                    <right>
                        <delay>333</delay>
                        <reflect>50</reflect>
                        <level>75</level>
                    </right>
                </channels>
                <low_freq>200</low_freq>
                <high_freq>6000</high_freq>
            </params>
        </effect>
    </effects>
</track>
<track>
    <id>4</id>
    <name>Instrumental</name>
    <volume>0</volume>
    <balance>0</balance>
    <state>
        <mute>false</mute>
        <solo>false</solo>
        <record>false</record>
    </state>
```

```
            <clips>
                <clip>
                    <id>1</id>
                    <signalId>4</signalId>
                    <cutStart>0</cutStart>
                    <posStart>0</posStart>
                    <duration>7259264</duration>
                    <volume>0</volume>
                </clip>
            </clips>
            <effects></effects>
        </track>
    </tracks>
    <signals>
        <signal>
            <id>1</id>
            <name>LDVs.wav</name>
            <source>/Audiora/Records/</source>
            <duration>7312529</duration>
        </signal>
        <signal>
            <id>2</id>
            <name>BGVs.wav</name>
            <source>/Audiora/Records/</source>
            <duration>6193821</duration>
        </signal>
        <signal>
            <id>3</id>
            <name>Adlibs.wav</name>
            <source>/Audiora/Records/</source>
            <duration>5382137</duration>
        </signal>
        <signal>
            <id>4</id>
            <name>I3YV.wav</name>
            <source>/Music/</source>
            <duration>7259264</duration>
        </signal>
    </signals>
</project>
```

ДОДАТОК Б

Скан-копії сторінок із тезами зі збірника матеріалів наукової конференції “МІТ-2022”

УДК 004.4'277

ДОСЛІДЖЕННЯ МОБІЛЬНИХ ЗАСТОСУНКІВ ЦИФРОВОГО ОБРОБЛЕННЯ ЗВУКОВИХ СИГНАЛІВ

Сичков Віталій Сергійович
к.т.н., доцент каф. ІПЗ Писаренко Катерина Олександрівна
Державний університет «Одеська політехніка», УКРАЇНА

АНОТАЦІЯ. У цій роботі проаналізовано програмні застосунки для роботи з аудіосигналами для ОС *Android* та описано їх основні проблеми. На основі цього запропоновано розроблення власного засобу, яке використовуватиме паралельне оброблення, а також оцінено скорочення кількості виконуваних операцій користувачем на застосування різноманітних ефектів до звукових доріжок на стадії мастерінгу.
Ключові слова: обробка звукових сигналів, цифрова обробка аудіосигналів/

RESEARCH OF MOBILE APPLICATIONS OF DIGITAL PROCESSING OF SOUND SIGNALS

Vitalii Sychkov
PhD, Associate Professor of the department of SE Kateryna Pysarenko
State University “Odessa Polytechnic”, UKRAINE

ABSTRACT. This paper analyzes software applications for working with audio signals for Android and describes their main problems. Based on this, it is proposed to develop its own tool that will use parallel processing, as well as estimated the reduction in the number of operations performed by the user to apply various effects to soundtracks at the mastering stage.
Keywords: Sound signals processing, digital audio signals processing

Вступ. Під обробленням звукового сигналу розуміють зміну частотної або фазової характеристики, звуження або розширення динамічного діапазону, застосування амплітудної, частотної або фазової модуляції, видалення шумів, а також створення затриманих за часом гаснучих копій цього сигналу. Сучасне програмне забезпечення дозволяє здійснювати перетворення аудіо-сигналів будь-якої складності та створювати найнеймовірніші звукові ефекти [1], проте кожне з них має як свої переваги, так і недоліки, які наведено в основній частині роботи.

Мета роботи. Оцінити зменшення кількості виконуваних операцій накладання різних ефектів на звукові сигнали користувачем, яке відбувається за рахунок використання методу паралельного оброблення під час мастерінгу вже зведених аудіодоріжок.

Основна частина роботи. Необхідність у використанні засобів записування і оброблення звукових сигналів на мобільних пристроях пояснюється виникненням різних життєвих ситуацій. Наприклад, необхідно записати лекції в університеті, певну аудіо-нотатку (оскільки не завжди під рукою є ручка та аркуш паперу), зробити пояснювальні коментарі до відеоролика, записаного з екрана смартфону, або навіть записати озвучку чи вокальну партію, коли під рукою немає мікрофона чи зовнішнього рекордера. Також не всі мають настільний ПК чи ноутбук, щоб працювати з десктоп-версіями таких програм, але внутрішній креатив вимагає негайних дій і звершень.

Для визначення власної програмної розробки насамперед проаналізуємо аналогічні засоби для мобільної операційної системи *Android*. У таблиці 1 наведено порівняльні характеристики найпопулярніших із них.

Таблиця 1 – Порівняльні характеристики програмних засобів оброблення звукових сигналів

Характеристика	Назва програмного засобу		
	Super Sound	*WaveEditor*	*Lexis Audio Editor*
Базові операції зі звуковими доріжками (об’єднання, розділення, відтворення, запис, приглушення тощо)	+	+	+
Еквалізація, нормалізація, компресія, реверберація, затримування, модуляція, змінення темпу, швидкості, висоти тону, ефекти реверс, луна, фейзер, фленджер	–, окрім, еквалізації та змінення I-III	+, окрім реверсу, луни і фленджеру	+, окрім затримування, модуляції, фейзеру і фленджеру
Частотний і спектральний аналіз звукових сигналів	–	+	–
Виокремлення звукового сигналу від потоку відео	+	+	–
Режим одночасної роботи з множиною доріжок	+	+	–
Паралельне оброблення звукових сигналів	–	–	–

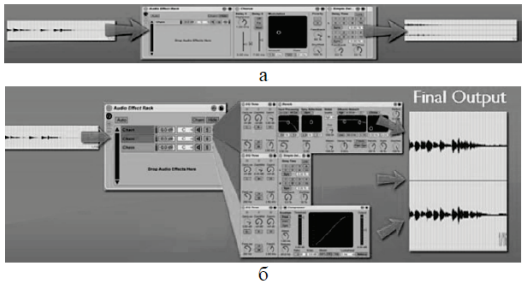
Як наслідок аналізу наявних програмних рішень було виявлено такі основні проблеми:

1. Ці засоби дозволяють застосовувати лише певні підмножини звукових ефектів, які є найрозповсюдженішими в настільних рішеннях, що може обмежувати можливості користувача реалізувати свої художні задуми.

2. Не всі розглянуті системи дозволяють оцінювати ступінь і характер вираженості частот сигналу в часі як за осцилограмою, так і за спектрограмою, що може ускладнити користувачеві процес частотної корекції, а відсутність режиму «мультитрек» може взагалі унеможливити їх мікшування (зведення), яке зазвичай виконується ще до стадії мастерінгу.

3. Критичною є проблема відсутності паралельного оброблення звукових сигналів у розглянутих засобах для ОС *Android*, які мають багатоканальний режим. У такому разі втрачаються вихідні сигнали (оскільки ефекти застосовуються безпосередньо до них), відсутній гнучкий контроль застосування ефектів через лінійність процесу. Користувачеві складно порівнювати і підлаш-товувати параметри тих чи тих ефектів, а за припущення помилки, наприклад, неправильно підібраного ефекту або його параметрів іще на початку мастерінгу, користувач буде змушений «відкотитися» назад, втративши весь прогрес та, як наслідок, змарнує свій час і зусилля на виконання операцій, яких узагалі можна було уникнути.

На підставі проведеного аналізу програмних засобів і наявних проблем можна зробити висновок щодо необхідності розроблення власного мобільного засобу паралельного цифрового оброблення звукових сигналів. На рисунку 1 зображено схеми послідовного та паралельного процесів на прикладі застосування ефектів модуляції (хорусу), затримання і реверберації.



а

б

Рис. 1 – Схеми оброблення сигналу: а – послідовне, б – паралельне

Для оцінки кількості виконуваних операцій для обох методів оброблення уведемо таку множину:

$$O = \{o_i\}, i = \overline{1, N}, \quad (1)$$

де o_i – i-та операція додавання (“+”) чи видалення (“–”) звукового ефекту, а також скасування (“×”) попередньої операції (“+” або “–”),

N – кількість виконаних операцій від останнього збереження або створення проєкту (якщо до цього моменту не було його збережень і перезапуску програмного засобу).

Нехай користувач послідовного обробляє аудіосигнали і йому необхідно видалити ефект, застосований на j-й операції o_j, $1 \le j \le N$, та який не був видалений операцією o_{j+1}, або зробити n видалень ефектів, для яких передує операція o_j. Для цього йому необхідно виконати $s = (N - j + 1)$ операцій “×” і $d = (s - n - 2p)$ операцій “+” (які були скасовані та які не заплановано видаляти, інакше вони входять до числа n), де p, $p \le 0 \le s$ div 2, – кількість пар взаємовиключних операцій “+” і “–” серед множини $\{o_j, o_{j+1}, \ldots o_N\}$, виконаних над одним звуковим ефектом. Водночас за паралельного оброблення користувачеві потрібно виконати всього n операцій видалення.

Висновки. Як наслідок проведеного дослідження мобільних застосунків оброблення звукових сигналів було визначено їх основні проблеми, зокрема відсутність паралельного накладання ефектів, та запропоновано розроблення програмного засобу, позбавленого цього, а також інших недоліків аналогічних розглянутих рішень.

Також було оцінено скорочення кількості операцій, виконуваних користувачем. Варто зауважити, що виграш відсутній при $n = 1$ та $j = N$, тому запропонований метод є доцільним у досить великих проєктах, де для множини сигналів застосовують достатню кількість звукових ефектів, які потребують ретельного налаштування, незалежно один від одного, що є досить поширеним випадком під час створення складної та насиченої аудіо-продукції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вологдин, Э. И. Методы и алгоритмы обработки звуковых сигналов: Курс лекций. — СПб.: СПбГУТ, 2009. — 96 с.