МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

Обробка Аудіо

(Розробка еквалайзера для цифрових звукових робочих станцій)

Текстова частина до курсової роботи за спеціальністю “Інженерія програмного забезпечення”

Керівник курсової роботи

доцент Афонін А.О.

_____

“____” _____ 2021 р.

Виконав студент

Ставровський М.А.

“____” _____ 2021 р.

Київ 2021

Зміст

Тема: Обробка аудіо. Розробка еквалайзер плагіну на основі фреймворку JUCE

Календарний план виконання роботи:

№ п/п	Назва етапу	Термін виконання етапу	Примітка
1	Отримання теми курсової роботи	Листопад 2020р.	
2	Дослідження предметної області	Лютий 2021р.	
3	Розробка застосунку	Квітень 2021р.	
4	Написання текстової частини	Квітень 2021р.	
5	Коригування роботи згідно зауважень керівника	Травень 2021р.	
6	Створення презентації	Травень 2021р.	

Студент Ставровський М.А.

Керівник Афонін А.О.

“_____” _____

Анотація

Ця курсова робота досліджує цифрову обробку аудіо у вигляді процесу еквалізації. Розроблено додаток еквалайзер на основі фреймворку JUCE та описаний процес його розробки.

Вступ

Дуже важко уявити сучасне життя без аудіо. Ми можемо цього не помічати, але ми споживаємо звукозапис кожного дня. Це може бути телевізійна програма, відео, музика, аудіокнига, радіо-трансляція, тощо. І якими б тривіальними для нас не здавалися ці речі, насправді, за кулісами завжди працюють, так звані, аудіо-інженери, які стараються зробити звук якомога чистішим та приємнішим для людського вуха. Не залежно від напряму, основна робота аудіо-інженерів - це обробка аудіо сигналу. І найбільш універсальним інструментом для цього є еквалайзер.

Еквалізація - це процес зміни балансу частот електронного сигналу. Ціль еквалізації - посилення або приглушення певних частот для досягнення сигналу потрібного для певної задачі. Інструмент, який здійснює цей процес називається *еквалайзером*.[1] Раніше такий інструмент мав вигляд фізичного приладу, який обробляв аналоговий сигнал, та у сучасній ері оцифрування цей прилад частіше можна побачити у формі програмного застосунку.

Мета курсової роботи - дослідження процесу еквалізації, аналіз функціоналу сучасних еквалайзерів, та розробка власного програмного застосунку.

У першому розділі розглянуті теоретичні відомості про еквалізацію та роботу з аудіо в цілому. У другому розділі проведений аналіз типів еквалайзерів та архітектури сучасного середовища роботи з аудіо. У фінальному розділі описаний процес створення власного додатку.

Розділ 1: Теоретичні відомості

Курсова робота зосереджена на роботі з цифровим аудіо, відповідно інформація не буде стосуватися акустичного та аналогового звуку.

Звук у цифровому вигляді зберігається у формі двійкових даних, які розбиваються на багато невеликих сегментів інформації, розміром 16 або 24 біти, які називають семплами. Саме за допомогою семплів відбувається обмін інформацією, який призводить до запису або відтворення звуків. [2]

Якість звуку залежить від таких двох параметрів:

- *Частота дискретизації* (sample rate) - це кількість семплів які обробляються за секунду. Від цього залежить кількість частот можливих до відтворення. Ця величина вимірюється в герцах.
- *Амплітудна роздільність* (bit depth) - це кількість бітів, яку містить у собі семпл. Від цього параметру буде залежати наскільки точно сигнал буде оцифровано і наскільки багато пам'яті буде займати аудіо.

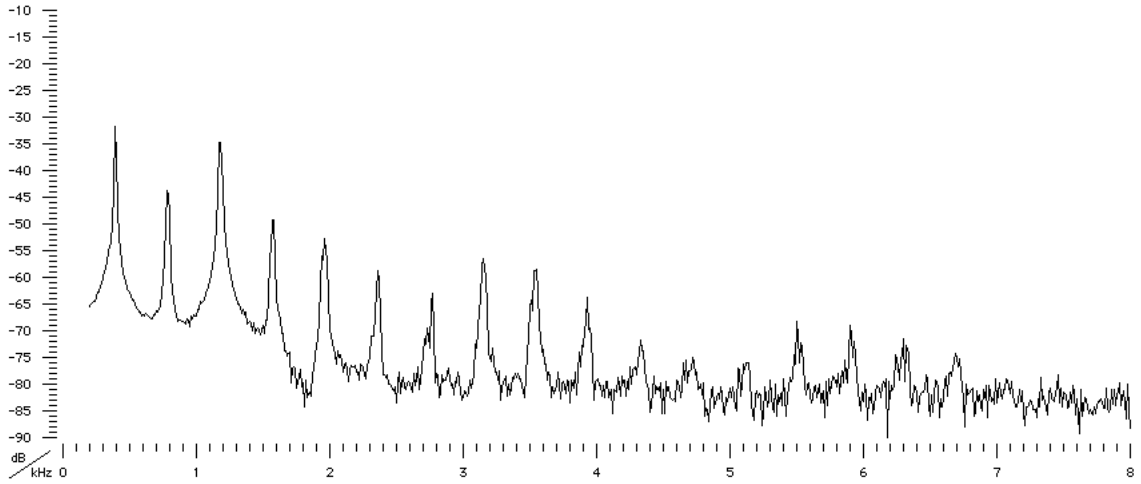
Зазвичай пропускна частота семплів становить не менше 40 тис. семплів за секунду, або ж 40 кГц. Чим більша ця частота та амплітудна роздільність - тим більше буде навантажуватися процесор для того щоб обробити дані.

Еквалізація - це один із найбільш вагомих аспектів будь-якої роботи з аудіо. Еквалізація дуже часто використовується для того щоб "чистити" звук, тобто прибирати шуми на певних частотах. Також еквалізацію використовують для того, щоб посилювати тихі частоти або ж послаблювати гучні частоти заради досягнення балансу.

Важливо зрозуміти, яким чином усе це досягається. В цьому розділі я детальніше проаналізую як працює еквалізація та як використовується еквалайзер.

Спершу дамо визначення спектру частот. *Спектр частот* - це відображення звуку, найчастіше у вигляді графіку, на якому відображається кількість вібрацій, або іншими словами, гучність кожної окремої частоти. Гучність вимірюється у децибелах, а частота в герцах.[3]

В умовах роботи з аудіо на спектрі частот відображається лише діапазон, який може почути людське вухо. Загально прийнятим діапазоном частот, що може чути людина, вважається діапазон від 20 Гц до 20 кГц. Спектр частот графічно виглядатиме приблизно так:



(Рис. 1.1) Візуальна репрезентація спектру частот[3]

Безпосередньо саме відображення частот на спектрі також називають *частотною характеристикою* (frequency response). Спектр частот використовується для наглядної роботи з аудіо. На ньому дуже просто побачити та проаналізувати будь-які зміни частотної характеристики звуку.

Зазвичай, в контексті еквалізації, спектр частот ділять на такі сегменти: низькі частоти (20 - 250 Гц), середні (250 Гц - 4к Гц) та високі (4 - 20 кГц).

Еквалайзер повинен мати змогу модифікувати цю частотну характеристику, що в результаті відобразиться на спектрі. Досягається це за допомогою фільтрів.

Фільтр - це певна математична формула, за якою вхідні дані модифікуються для того щоб отримати бажаний відфільтрований вихідний сигнал.[4] Існує багато різноманітних видів фільтрів.

За типом імплементації фільтри розділяють на:

- *СІХ фільтри* (FIR filter) - це фільтр з обмеженим часом імпульсної характеристики. Це означає, що в певний момент часу, якщо на вхід

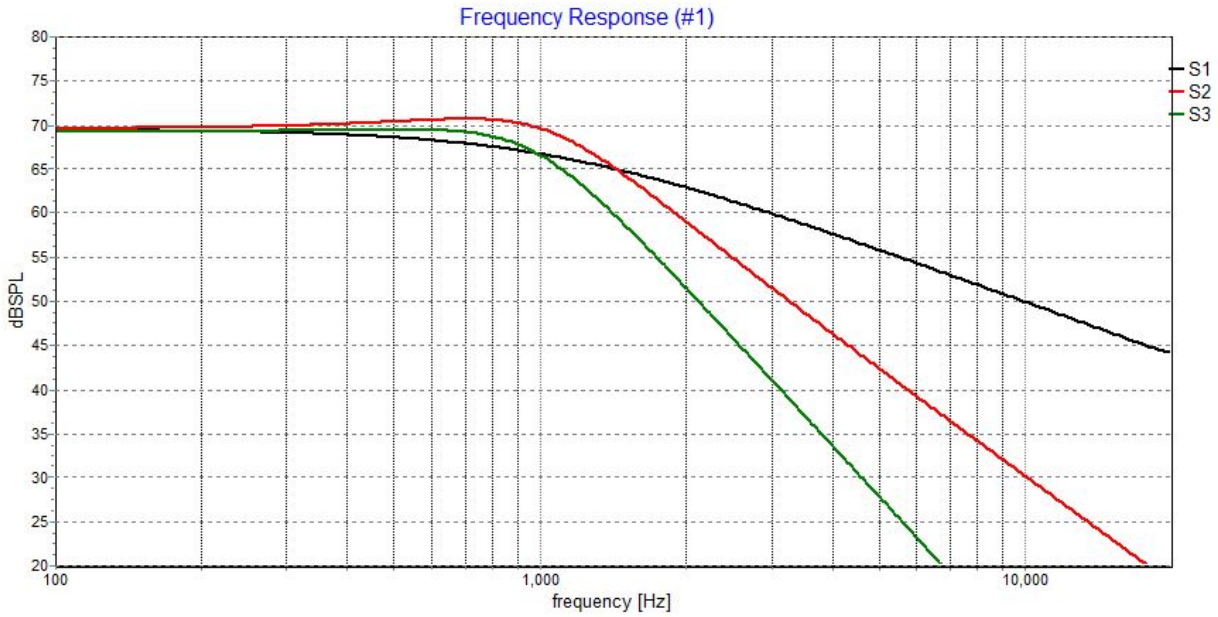
приходить багато семплів з 0 значеннями, фільтр також дає на вихід нульовий сигнал.[5]

- *НІХ фільтри* (IIR filter) - це фільтр, з необмеженою імпульсною характеристикою. Це досягається за допомогою використання одного з виходів фільтру як входу. Таким чином, якщо прийде семпл з одним значенням 1 і багатьма 0, то фільтр теоретично продовжить нескінченно подавати ненульовий сигнал.[6]

Порівнюючи, FIR фільтр є більш надійним але споживає набагато більше ресурсів. IIR фільтр же є більш точним та витрачає менше ресурсів, проте може мати деякі артефакти у роботі. Загалом, частіше використовують FIR фільтри. [7][8]

Зазвичай, фільтри мають два діапазони(дві смуги) частот - пропускну (passband) і смугу зрізу (stopband)[9]. Пропускна смуга - це діапазон частот, що “пропускається”, не змінюється в результаті роботи фільтра. Відповідно, смуга зрізу - це смуга частот, яка “зрізається”, прибирається фільтром. Можна сказати, що пропускна смуга множить сигнал на значення більше 1, а смуга зрізу на значення менше 1.

Характеристиками фільтру є сила або гучність фільтру(gain) частота зрізу (cut off), крутизна зрізу (slope) та ширина пропускної смуги (bandwidth). Частота зрізу визначає смугу зрізу. Крутизна зрізу визначає те наскільки різко смуга пропуску переходить у смугу зрізу (Див. Рис. 1.2). Вона вимірюється в Дб/октаву. Ширина пропускної смуги відповідно визначає діапазон частот пропуску та вимірюється в октавах.

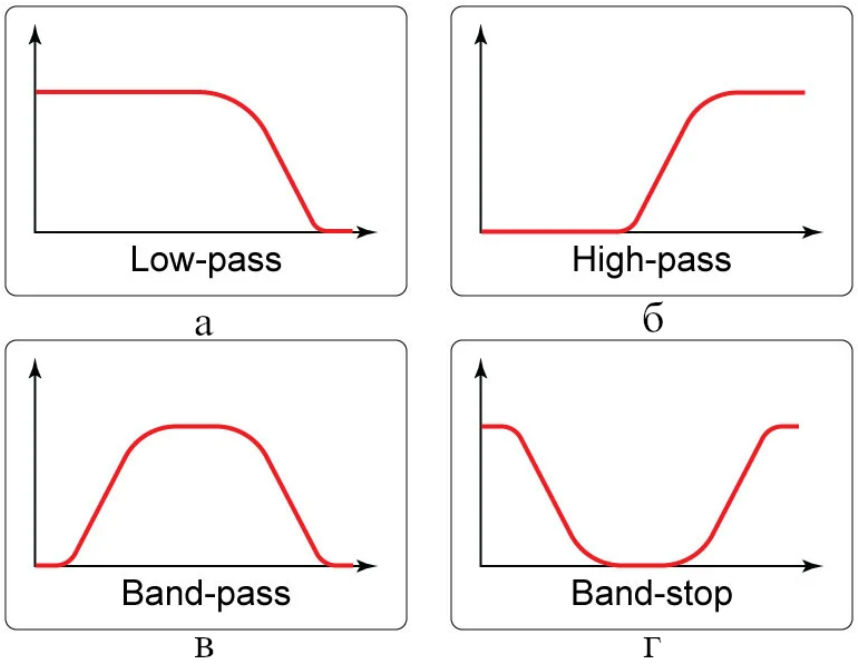


(Рис. 1.2) Візуалізація різної крутизни зрізу[10]

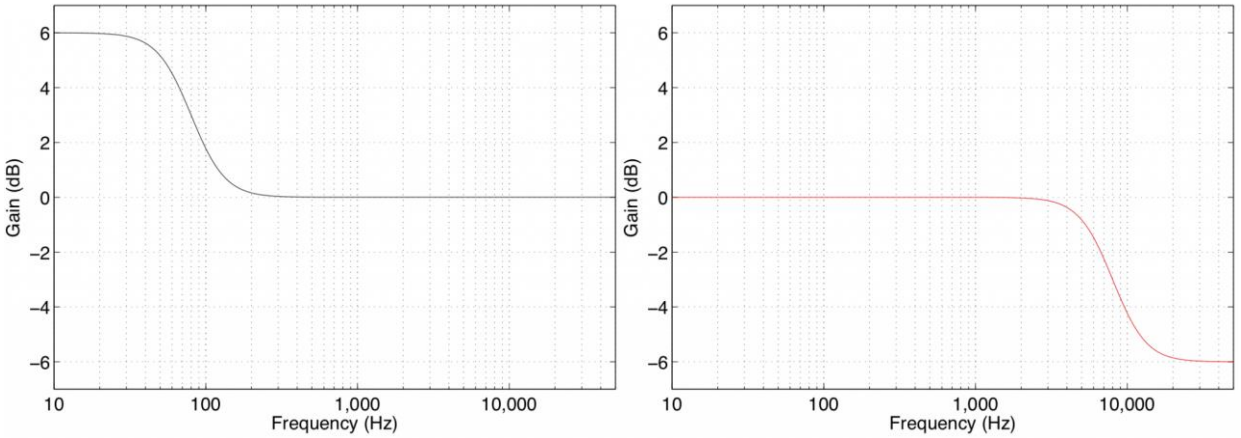
В залежності від розташування смуги зрізу та смуги пропуску, фільтри можуть мати різне призначення. За своєю функцією фільтри поділяються на:

- *Фільтр нижніх частот* (low pass filter) - фільтр, який зрізає високі частоти, залишаючи лише низькі. (Рис. 1.3 а)
- *Фільтр верхніх частот* (high pass filter) - фільтр, який зрізає низькі частоти, залишаючи лише високі. (Рис. 1.3 б)
- *Смуговий фільтр* (bandpass filter) - фільтр, який залишає лише обраний проміжок частот, та зрізає усе інше. (Рис. 1.3 в)
- *Режекторний фільтр* (bandstop filter) - фільтр, який залишає усі частоти, крім обраного проміжку. (Рис. 1.3 г)
- *Піковий фільтр* (peaking filter) - фільтр, який маніпулює певним проміжком частот, з можливістю його посилювати або послаблювати. (Рис. 1.5)
- *Поличкові фільтри* (shelf filters) - вид фільтрів, які працюють схожим чином до фільтрів низьких та високих частот, але зріз відбувається не до 0, а до певної конкретної величини (Рис. 1.4):

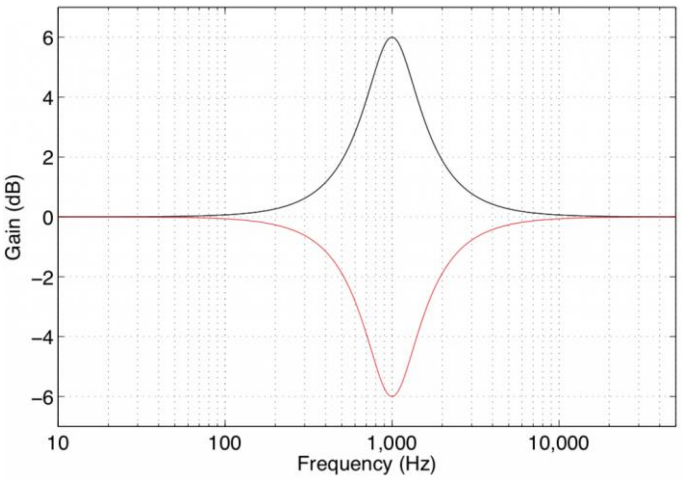
- *Низький поличковий фільтр* (low shelf filter)
- *Високий поличковий фільтр* (high shelf filter)



(Рис 1.3)[11] а) Фільтр нижніх частот; б) Фільтр високих частот; в) Смуговий фільтр; г) Режекторний фільтр



(Рис. 1.4)[12] Поличкові фільтри, зліва низький, що посилює сигнал, справа високий, що послаблює сигнал



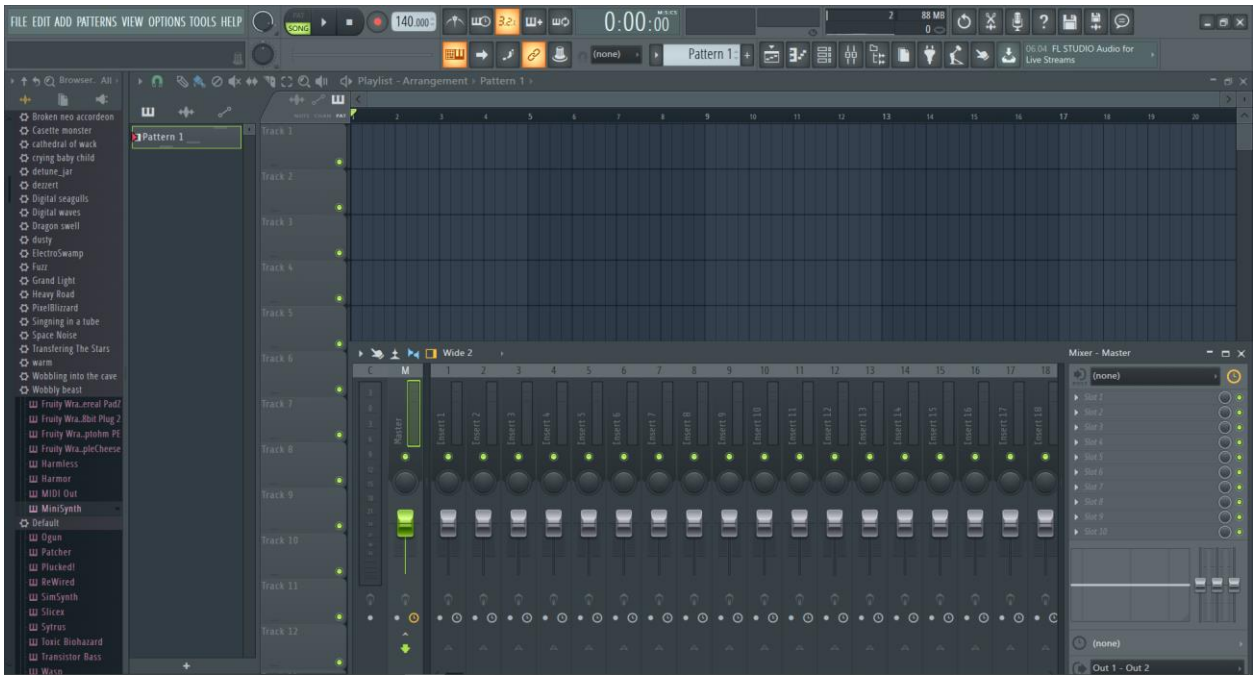
(Рис. 1.5)[12] Піковий фільтр

Випливаючи з теоретичної інформації про фільтри, виходить, що, узагальнено, еквалайзер - це набір фільтрів на різних частотах, які в сукупності дають контроль над усім спектром. Від того як реалізована робота з цими фільтрами залежить тип еквалайзера. Оскільки моя курсова робота присвячена цифровим еквалайзерам у вигляді програмного забезпечення, у наступному розділі я опишу як вони застосовуються, а у третьому розділі більш детально проаналізую види таких еквалайзерів.

Розділ 2: Робота плагіну всередині DAW

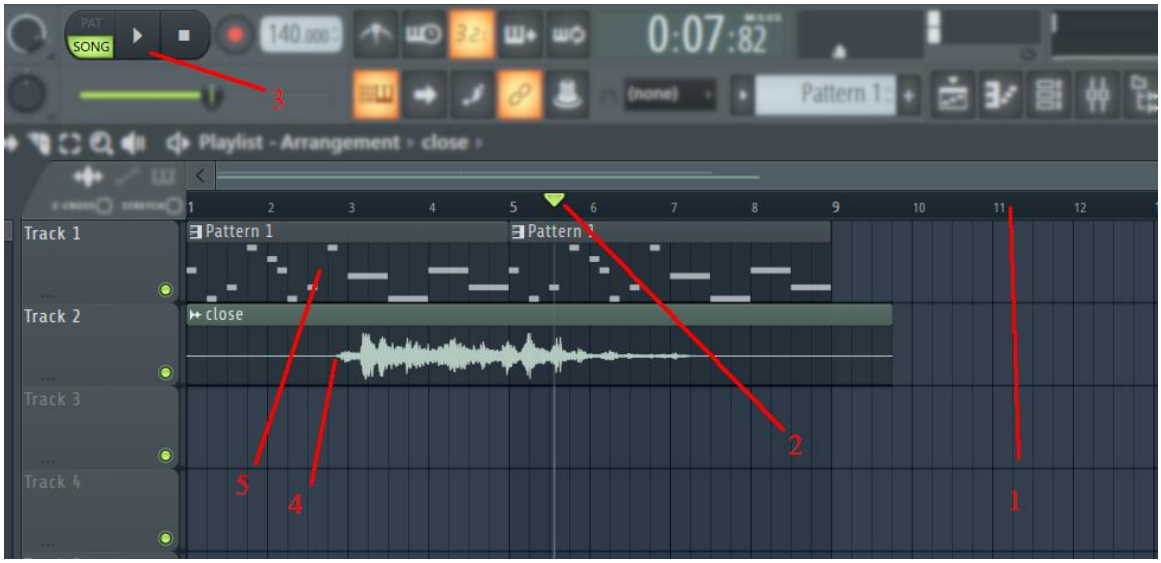
Як програмне забезпечення, сучасний еквалайзер має вигляд плагіну до інтерфейсу для обробки аудіо, який мають назву *цифрова звукова робоча станція*, або англійською, Digital Audio Workstation (DAW). Метою цього розділу є дослідити як плагін взаємодіє з DAW.

DAW має вигляд застосунку, який дає дуже широкі можливості для роботи з аудіо (Рис 2.1). Такий застосунок дає приблизно ті самі можливості, що і фізична студія звукозапису, але у компактній формі додатку на комп'ютері. Це означає, що програма дозволяє записувати звук, синтезувати звук, обробляти, компонувати, зводити та інше. В межах цієї курсової роботи, детально буде розглядатися лише застосування плагінів та все що для цього потрібно.



(Рис 2.1) DAW яким я користуюсь, FL Studio

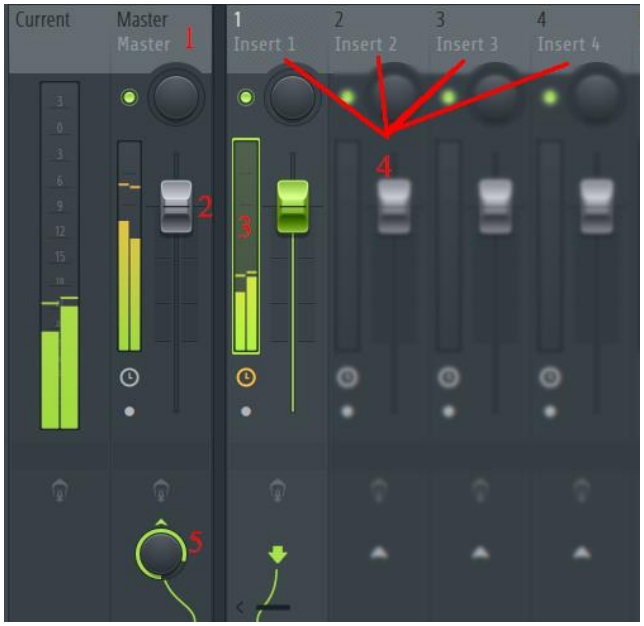
Для того щоб взагалі було з чим працювати, потрібно мати певний вхідний сигнал. Це досягається за допомогою патернів синтезаторів або існуючих аудіо-файлів. Ці елементи, розставляються на часову лінію, яку потім можна відтворити (Рис 2.2).



(Рис 2.2) Вигляд часової лінії та звуків розставлених на ній: 1) часова лінія; 2) маркер поточного часу; 3) кнопки для керування відтворенням; 4) імпортований аудіофайл; 5) Midi паттерн записаний синтезатором

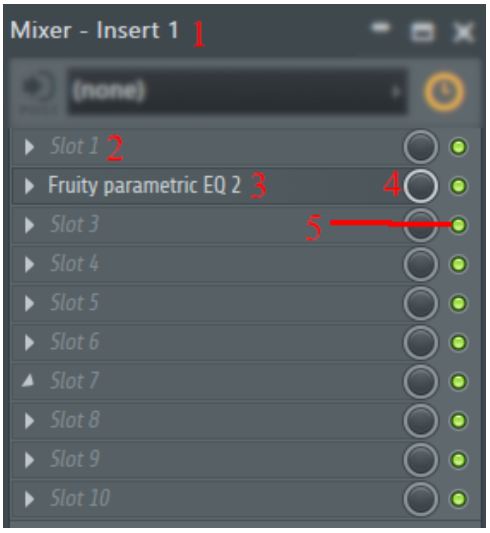
Загалом, серед інструментів, які використовуються у DAW розрізняють ефекти та синтезатори. Синтезатори генерують сигнали, ефекти модифікують уже існуючі сигнали. Еквалайзер належить до розряду ефектів, адже він не генерує сигнал, а модифікує його.

Відтворення будь-яких з розставлених на часовій лінії елементів створює сигнал. Цей сигнал надсилається на інтерфейс під назвою міксер. Міксер розбитий на канали, кожен з яких приймає певний сигнал, обробляє відповідно тому як він налаштований, та відправляє на вихід у якийсь інший канал. У міксері є головний канал, вихід якого - це те що ми чуємо в результаті. Усі інші канали повинні до нього приєднуватись, щоб їх було чути.



(Рис 2.3) Міксер: 1) головний канал; 2) налаштування гучності каналу; 3) Індикатор гучності каналу; 4) додаткові канали, що приєднуються до головного; 5) гучність виходу з одного каналу на інший

Кожен із каналів містить у собі певну обмежену кількість “місць” для ефектів (Рис 2.4). У випадку DAW, якою я користуюсь, кожен канал має 10 слотів для ефектів. Коли сигнал проходить через канал, він послідовно, по порядку, виконує кожен з накладених ефектів, передаючи сигнал від одного ефекту до іншого. Порядок виконання ефектів може бути критичним у роботі аудіо-інженера. Крім цього кожен з ефектів має характеристики ввімкнення\вимкнення та силу застосування ефекту.



(Рис 2.4) Слоти для ефектів каналу: 1) індикатор каналу; 2) пустий слот; 3) слот заповнений ефектом; 4) сила застосування ефекту; 5) кнопка ввімкнення\вимкнення ефекту

Саме тут і буде застосовуватись еквалайзер. DAW розроблені таким чином, що будь-які синтезатори або ефекти не є вбудованими. Замість цього, DAW імпортує ці інструменти як плагіни. Така система дозволяє мати свободу в розробці та використанні будь-яких інструментів, та також створює навколо себе окремий бізнес. Існує багато комерційних проектів, які займаються суто розробкою таких плагінів, якими потім можуть користуватися будь-хто у будь-якому персонально зручному середовищі.

Найчастіше аудіо плагіни розробляють у форматі VST, адже цей формат підтримується більшістю DAW. Іншими менш розповсюдженими форматами плагінів є AAX, який підтримується лише DAW компанії Pro Tools, та AU розроблений Apple лише для Mac систем. В курсовій роботі розглядатимуться лише VST плагіни.

Розділ 3: Види еквалайзерів

Знаючи яким чином еквалайзер застосовується в DAW, можна розглянути та порівняти їхні види. Я буду порівнювати їх за кількістю фільтрів, можливістю їх змінювати та додатковим функціоналом. Еквалайзери поділяються на[13]:

1. *Параметричний еквалайзер*
2. *Семі-параметричний еквалайзер*
3. *Динамічний еквалайзер*
4. *Графічний еквалайзер*
5. *Поличковий еквалайзер*

Параметричний еквалайзер - це найбільш популярний вид еквалайзерів. Такий еквалайзер може мати варійовану кількість фільтрів, яка може досягати п’ятнадцяти і більше фільтрів. Для кожного фільтру можна індивідуально налаштовувати:

- Тип фільтру
- Частоти на яких фільтр застосовується
- Ширину пропускної смуги фільтру
- Гучність вихідного сигналу фільтру
- Крутизну зрізу

Серед додаткового функціоналу, такий еквалайзер зазвичай має вбудований аналізатор спектру частот, на якому графічно видно які частоти приходять на вхід, які йдуть на вихід та на якому можна графічно керувати фільтрами. За рахунок додаткового функціоналу та змінної кількості фільтрів, такий еквалайзер є складним у своїй реалізації.

Семі-параметричний еквалайзер - це еквалайзер, схожий до параметричного, але без певного функціоналу. Такі еквалайзери мають фіксовану кількість фільтрів, зазвичай не більше п’яти. У порівнянні з параметричними еквалайзерами, вони рідко мають можливість змінити ширину смуги пропуску фільтру та зазвичай не мають графічного аналізатора частот. Можливість змінювати тип фільтру відсутня, або

обмежена. Через спрощений функціонал, такий еквалайзер простіший у реалізації та може використовуватись для більш вузьких задач, які не потребують точності та потужності параметричного еквалайзера.

Динамічний еквалайзер - це еквалайзер схожий до параметричного, але, на відміну від семі-параметричного, розширює функціонал. Динамічний еквалайзер має усі ті ж характеристики, що і параметричний, але додатково замінює статичне налаштування фільтрів на динамічне. Це означає, що сила кожного фільтру може змінюватись в залежності від вхідного сигналу, наприклад посилюватись або послаблюватись проходячи певний поріг гучності. [14]

Такий еквалайзер є дуже потужним та суміщає у собі функціонал еквалайзера та інших інструментів по типу компресора, тощо. Проте, таким еквалайзером набагато складніше оперувати та, за рахунок додаткових алгоритмів, його набагато складніше реалізувати.

Як приклад наведу Surfer EQ 2 (Рис 3.3). Це динамічний еквалайзер, який крім звичного функціоналу також має розроблений власний вид динамічної фільтрації, який заснований не на гучності сигналу, а на різних частотних серіях.

Графічний еквалайзер - це більш обмежений вид еквалайзерів. Він має статичну але велику кількість пікових фільтрів, яка може досягати тридцяти і більше. Такий еквалайзер не має можливості змінювати тип, частоту або ширину фільтрів, лише гучність. Також, він не має графічного аналізатора частот. Перевага такого еквалайзеру - це велика місткість фільтрів. У порівнянні з семі-параметричним, такий еквалайзер простіше реалізовувати.

Зазвичай цей інструмент виглядає схожим до міксера, де кожний слайдер відповідає за певний статичний діапазон частот. Наприклад на Рис 3.4 зображено Voxengo Marvel GEQ, який має у собі 16 фільтрів.

Поличковий еквалайзер - це найпростіший тип еквалайзерів. Зазвичай, вони мають 2 поличкових фільтри, або 3 фільтри, два крайніх з яких є поличковими а той, що посередині - піковий. Фільтри часто є статичними, без можливості зміни типу,

частоти застосування або ширини. Зазвичай поличковий еквалайзер не має додаткового функціоналу. Через свою простоту підходить для менш точних робіт, та, у порівнянні з іншими видами, він найпростіший у реалізації.

Для прикладу наведу 3-Band EQ компанії Kilohearts (Рис 3.5). Це надзвичайно простий поличковий еквалайзер на 3 фільтри, який навіть має простий аналізатор частот.

Підсумовуючи, можна утворити таку порівняльну таблицю:

Тип	Кількість фільтрів	Можливі модифікації	Додатковий функціонал
Параметричний	Варійована, до 15	Тип фільтру, частоти, ширина, гучність	Графічний аналізатор
Семі-параметричний	Статична, до 5	Тип фільтру (обмежено), частоти, ширина(рідко), гучність	Немає
Динамічний	Варійована, до 15	Тип фільтру, частоти, ширина, гучність	Графічний аналізатор, динамічність
Графічний	Статична, до 30	Гучність	Немає
Поличковий	Статична, до 3	Гучність	Немає

(Таблиця 3.1) Порівняльна таблиця видів еквалайзерів

Виходячи з цього порівняння, я вирішив розробляти семі-параметричний еквалайзер із трьома фільтрами та можливістю змінювати тип, частоти, ширину та гучність кожного. Я так вирішив, тому що за обраними характеристиками цей тип є найбільш оптимальним серед наведених.

Для розробки додатку я обрав фреймворк JUCE. Це фреймворк на основі мови програмування C++, який має дуже широкий набір інструментів для роботи з аудіо, зокрема для створенні плагінів для DAW.

Розділ 4: Розробка додатку

В цьому розділі буде описаний процес розробки семі-параметричного еквалайзера на основі фреймворку JUCE. Результатом має бути додаток формату VST з графічним інтерфейсом. У функціонал додатку входитимуть:

- Керування формою фільтру
- Керування силою застосування фільтру
- Керування частотами фільтру
- Керування шириною фільтру
- Прив’язка параметрів до DAW
- Можливість зберігати стан плагіну

Очевидно, що додаток матиме frontend та backend. Для цього в JUCE проект розбивається на два класи: AudioEditor та AudioProcessor відповідно.

4.1 Базовий фронтенд застосунку

Першим я почав створювати базовий графічний інтерфейс, для якого я потім буду реалізовувати функціонал. Для першого прототипу мені потрібно створити: загальний слайдер гучності, слайдери гучності кожного з фільтрів та слайдери налаштування частоти та ширини(Q) фільтрів. Усі відповідні елементи були створені в AudioEditor як екземпляри класу Slider.(Рис 4.1).

Кожен із них налаштовується у конструкторі AudioEditor, де встановлюється їхній тип, можливі значення, текстова анотація, тощо. Там же вони додаються на головну сцену за допомогою методу addAndMakeVisible().

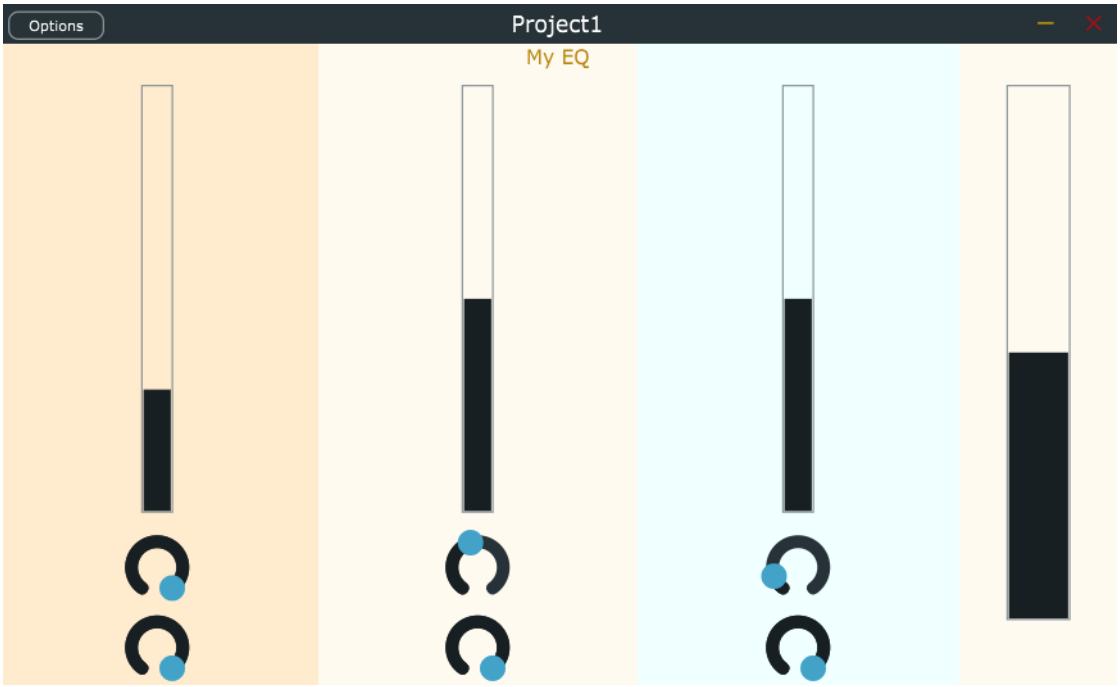
```
Project1AudioProcessor& audioProcessor;
juce::Slider main_volume;
juce::Slider lo_bus_vol;
juce::Slider lo_bus_cutoff;
juce::Slider lo_bus_q;
juce::Slider mid_bus_vol;
juce::Slider mid_bus_cutoff;
juce::Slider mid_bus_q;
juce::Slider hi_bus_vol;
juce::Slider hi_bus_cutoff;
juce::Slider hi_bus_q;
```

```
// Low bus setup
lo_bus_vol.setSliderStyle(juce::Slider::LinearBarVertical);
lo_bus_vol.setRange(0.01f, 2.0f, 0.01f);
lo_bus_vol.setTextBoxStyle(juce::Slider::NoTextBox, true, 30, 0);
lo_bus_vol.setPopupDisplayEnabled(true, false, this);
lo_bus_vol.setTextValueSuffix("Db");
lo_bus_vol.setValue(audioProcessor.lo_sliderVel);
```

(Рис 4.1) Зліва, декларація потрібних слайдерів; справа, приклад налаштування одного з них

За допомогою методу paint(), можна створювати певні графічні елементи, як наприклад елементи фону, текст, тощо. Він виконується кожного разу, коли вікно ініціалізується і його потрібно промалювати. Це відбувається за допомогою окремого класу Graphics. В ньому я зробив 4 колонки різних кольорів, у яких потім будуть знаходитись потрібні слайдери.

В AudioEditor також є метод resized(), який виконується, коли відбуваються зміни розмірів вікна. Оскільки моє вікно статичного розміру, цей метод виконуватиметься лише раз. В цьому методі задаються розміри кожного функціонального елемента. Вікно плагіну розміром 700 x 400 і умовно я розділив його на 4 колонки: 100 пікселів ширини на головну гучність та по 200 пікселів ширини на кожен піксель. У кожного фільтру є один лінійний вертикальний слайдер, та два кругові слайдери знизу. В результаті був створений перший макет інтерфейсу (Рис 4.2):



(Рис 4.2) Перший варіант графічного інтерфейсу

4.2 Прив'язка бекенду до фронтенду

Далі потрібно було прив'язати інтерфейс до певного функціоналу. Для кожного параметра я створив окремі змінні всередині AudioProcessor, які мають змінюватись при відповідних змінах в графічному інтерфейсі. В AudioEditor я додав перевизначення вбудованого методу sliderValueChanged(), в якому в залежності від того, який слайдер змінюється, змінюю відповідне значення в AudioProcessor.

Далі я реалізував зміну гучності. AudioProcessor має метод processBlock(), який працює з буфером вхідних даних. В ньому я додав циклічний обхід по семплах кожного вхідного каналу, де вхідне значення домножується на гучність яку ми отримуємо з графічного інтерфейсу (Рис 4.3). Для гучності я використовую діапазон значень від -20 до 20 Дб. Людське вухо чує гучність нелінійно, а логарифмічно. Проте, цифрова гучність є лінійною, де значення одиниці дорівнює нулю децибел. Існує формула для конвертації гучності з децибел у потрібне лінійне значення: $y = 10^{\frac{\Delta L}{20}}$, де y - лінійна гучність, ΔL - гучність у децибелах[15]. Для такої конвертації JUCE має окремий метод decibelsToGain().

```
for (int channel = 0; channel < totalNumInputChannels; ++channel)
{
    auto* channelData = buffer.getWritePointer (channel);
    const auto readPointer = buffer.getReadPointer(channel);
    for (int sample = 0; sample < buffer.getNumSamples(); ++sample) {
        channelData[sample] = channelData[sample] * juce::Decibels::decibelsToGain(main_volume);
    }
}
```

(Рис 4.3) Цикл для застосування загальної гучності

4.3 Фільтрація

Наступним кроком мені потрібно було знайти як саме реалізовувати фільтрацію. Мені здалося, що найзручніше реалізовувати фільтрацію буде за допомогою модуля DSP(Digital Signal Processing). DSP містить у собі багато додаткових інструментів для спрощення роботи із обробкою звуку. Зокрема, DSP має у собі імплементації FIR та IIR фільтрів. Я обрав IIR фільтри.

Спершу я створив один фільтр, для перевірки його справності. Для цього в AudioProcessor я додав приватне поле IIR фільтру lowPassFilter. Проте, цього було б не достатньо, адже такий фільтр є моно-орієнтованим, що означає, що він не зміг би обробити стерео сигнал. Для цього щоб такої проблеми не було, в DSP існує додатковий клас ProcessorDublicator, який дублює певний клас обробки на два різних, щоб один обробляв лівий сигнал, а інший правий. Таким чином ProcessorDublicator може перетворити моно-фільтр на стерео-фільтр.

Для того, щоб будь-які елементи модулю DSP працювали, потрібно ініціалізувати ProcessSpec, клас, який надає DSP контекст необхідний для коректної роботи фільтрів. Це відбувається у методі AudioProcessor prepareToPlay(), який розрахований на попередню ініціалізацію певних параметрів, модулів тощо. Тут я створюю новий екземпляр класу ProcessSpec та передаю у нього частоту дискретизації, амплітуду роздільності та кількість активних каналів. Цей ProcessSpec передається у створений до цього фільтр, за допомогою методу prepare для завчасного налаштування коректної роботи фільтру.

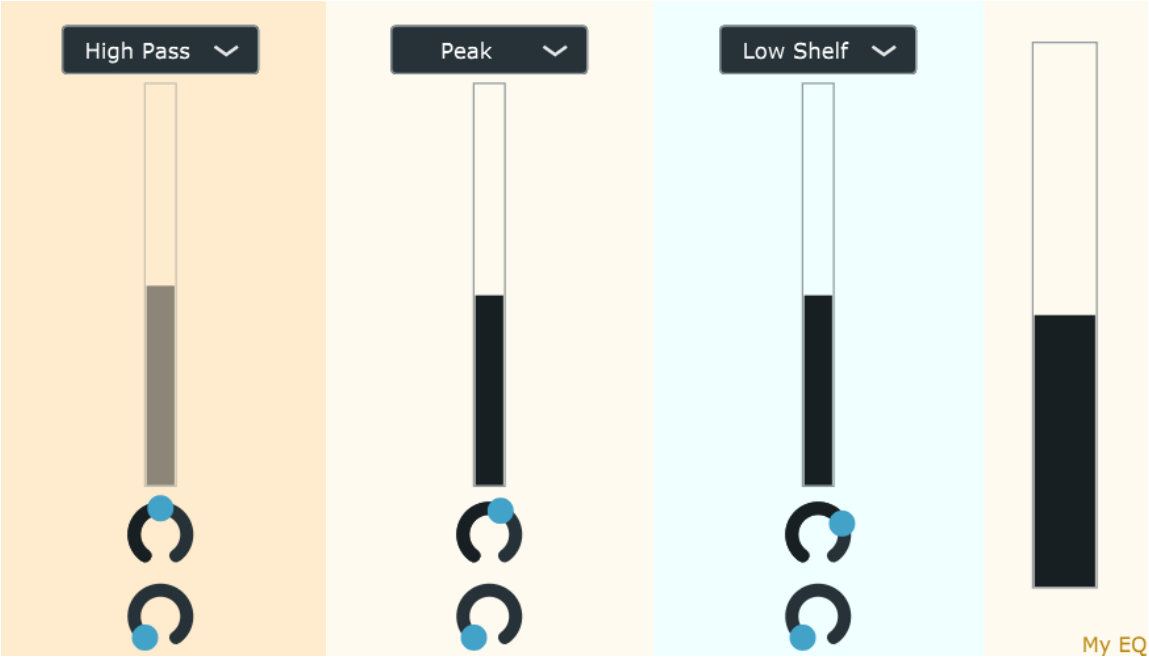
Далі потрібно було додати саму обробку фільтром сигналу. Для цього в processBlock створюється екземпляр класу dsp::AudioBlock - клас на основі буферу даних, який використовує DSP для коректної роботи. Після цього створений аудіо-блок передається у метод process фільтру. Цей метод буде опрацьовувати сигнал. Проте, так фільтр буде статичним і зміни у графічному інтерфейсі нічого не дадуть. Для того, щоб це виправити я створив додатковий метод updateFilters(), в якому кожен фільтр змінюватиме стан в залежності від потрібних аргументів. Цей метод буде виконуватись на кожній ітерації processBlock(). В тому ж методі, при оновленні стану фільтру передається тип цього фільтру. Це відбувається через dsp::IIR::Coefficients, де для IIR фільтрів створюються різні види фільтрів. Таким чином, мені вдалося зробити перший low pass фільтр використавши метод makeLowPass. Такий метод приймає у себе поточну частоту дискретизації, частоти застосування, Q-фактор та, у випадках інших фільтрів, лінійну гучність фільтру.

Частоти застосування фільтрів налаштовані на весь спектр від 20 Гц до 20 кГц, а гучність, аналогічно до загальної, від -20 Дб до 20 Дб. Проте, варто пояснити, що таке Q-фактор. Q-фактор - це певний безрозмірний коефіцієнт, який використовується для обрахування ширини фільтру, де менше значення Q даватиме більшу ширину і навпаки. Ширина вимірюється у октавах. Мені здалося, що з користувацькою точки зору, простіше сприймати реальну ширину. Тому я налаштував відповідні слайдери на значення від 0.05 до 20 октав, а на бекенд частині за значенням ширини вираховую значення Q. Воно рахується за формулою: $Q = \frac{\sqrt{2^N}}{2^N - 1}$, де N - ширина в октавах. [16]

Аналогічним чином я створив ще два фільтри, досягаючи потрібного набору трьох фільтрів. Наступним кроком була зміна фільтрів. Для цього я ввів три нових змінних, які позначатимуть стан кожного фільтру. Стан матиме значення від 1 до 8, кожен з яких позначає певний тип фільтру: 1 - вимкнений, 2 - low pass, 3 - low shelf, 4

- peak, 5 - high shelf, 6 - high pass, 7 - band pass, 8 - band stop. Безпосередня зміна типу відбувається в updateFilters().

Для того, щоб ці зміни запрацювали, потрібно оновити графічний інтерфейс інтерфейс. Для кожного фільтру я додав згори комбобокси, які містять потрібні назви фільтрів. Крім цього, деякі фільтри не мають характеристики гучності, тому відповідно до фільтру вона може ставати неактивною. Оновлений інтерфейс зображено на Рис. 4.4.



(Рис 4.4) Оновлений інтерфейс з можливістю міняти тип фільтру

4.4 Розширена взаємодія з DAW. Параметри

На даному етапі, додаток легко експортується у формат vst та може запускатися всередині DAW. Але цього не достатньо для коректної роботи. У плагіні потрібно прописати логіку взаємодії із DAW.

Очевидно, що кожен плагін має певні параметри, за допомогою яких користувач керує ним. DAW дають можливість керувати цими параметрами на вищому рівні, за допомогою автоматизації або MIDI пристроїв. Я розгляну як приклад лише автоматизацію. Автоматизація певних параметрів плагіну дає

можливість надати інструкції зміни цих параметрів в часі. У DAW, якою я користуюсь, найпростіша автоматизація виглядає так: (Рис 4.5)

(Рис 4.5) Автоматизація в FL Studio

На зображеній автоматизації, заданий параметр змінюється з найвищого значення до найнижчого, проходячи через кілька різних точок. Ці точки задаються користувачем. Коли відбуватиметься відтворення, поки маркер поточного часу проходить в діапазоні цієї автоматизації, параметр буде змінюватись відповідно. Ця функція є дуже важливою для роботи з будь-яким інструментом, адже це надає можливість застосовувати його динамічно. Проте, наразі, моїм плагіном це не передбачено.

Для того, щоб параметри плагіну коректно комунікували з DAW, до AudioProcessor потрібно додати нове поле у вигляді екземпляру класу AudioProcessorValueTreeState. Цей клас зберігає поточний стан плагіну, дозволяє напряму підєднуватися до слайдерів графічного інтерфейсу та має утиліти для прямого підключення до DAW. Спершу, на стороні бекенду я додав до ValueTreeState усі існуючі параметри. Для того, щоб ці параметри змінювались при взаємодії з інтерфейсом, на стороні фронтенду я створив відповідні екземпляри SliderAttachment та ComboBoxAttachment. Ці два класи слугують як прив’язка графічних елементів до дерева станів, і приймають у конструкторі саме дерево, ідентифікатор потрібного параметру та графічний елемент який буде керувати цим параметром.

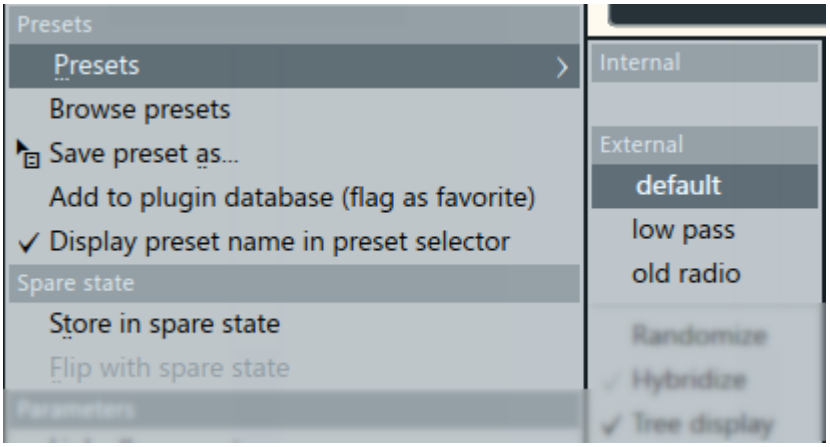
Тепер, якщо завантажити плагін у DAW, то створені параметри з’являються в окремому меню, де ними можливо керувати поза інтерфейсом плагіну та використовувати для автоматизації, прив’язки до MIDI пристрою, тощо (Рис 4.6).

(Рис 4.6) Зліва зображено меню параметрів всередині DAW, справа видно можливість редагувати значення параметра поза плагіном

4.4 Розширена взаємодія з DAW. Збереження стану плагіну

Наступним кроком у взаємодії з DAW є збереження стану плагіну у файл. Зберігаючи стан можна створювати шаблони, які можна зручно і швидко застосувати для виконання певної задачі.

Всередині PluginProcessor існують два методи: getStateInformation та setStateInformation, які викликаються при збереженні та завантаженні стану із файлу. Оскільки всі аргументи стосовно файлу передаються через DAW, потрібно лише правильно обробити стан самого плагіну. Універсально, стани плагінів зберігаються у вигляді файлу розмітки XML. Тож для збереження стану, в getStateInformation спершу потрібно конвертувати вигляд ValueTreeState до XML та відправити у файл. Для підвантаження файлу, в setStateInformation потрібно перевірити, чи обраний файл коректний та задати новий стан ValueTreeState. Після цього, всередині DAW можна зберігати та підвантажувати стан плагіну. До них можна доступитися через додаткове меню (Рис 4.7).

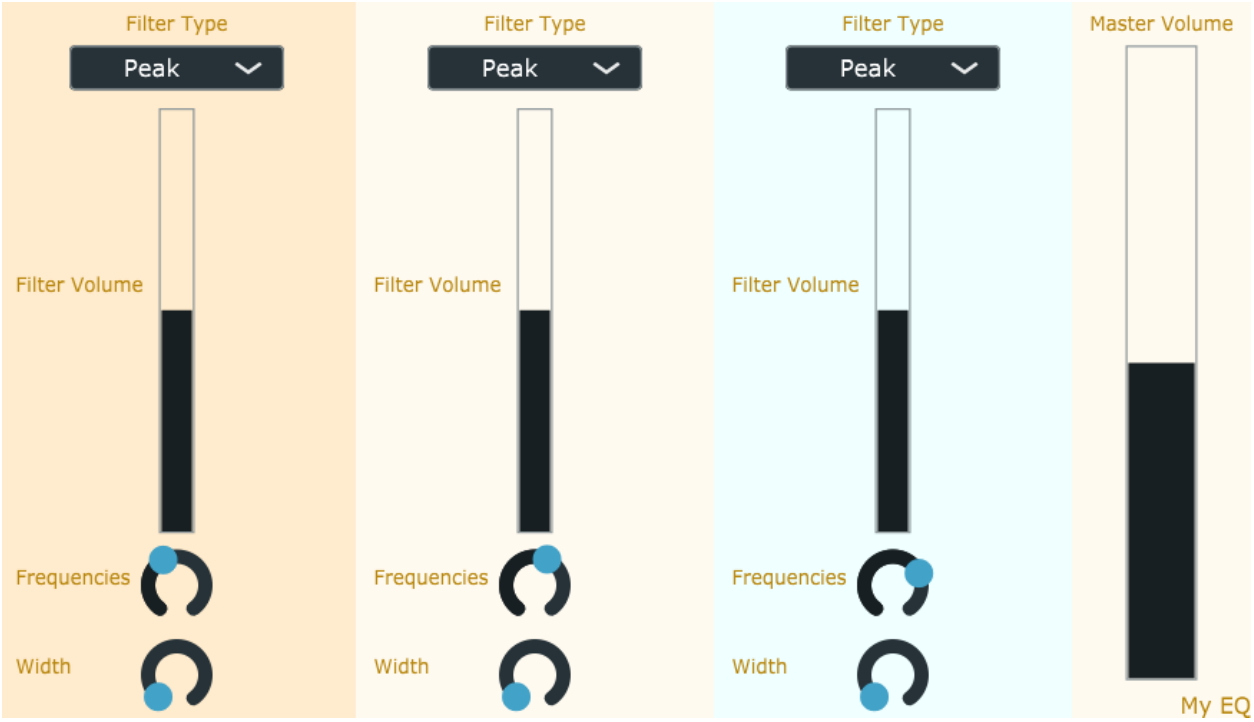


(Рис 4.7) меню для зберігання та підвантаження стану плагіну

4.5 Завершальні зміни

До цього моменту, додаток мав би бути повністю справним. Для перевірки, я надіслав готовий vst3 файл кільком людям, які користуються DAW. Плагін коректно відображався та працював. Проте, було виявлено, що при використанні автоматизації параметрів вона не була помітна у експортованому аудіо-файлі. Пізніше, я знайшов причину цієї проблеми. До того, як додавати TreeValueState, на бекенді я використовував окремі змінні на кожен з параметрів. TreeValueState працює асинхронно, відповідно дані могли не синхронно оновлюватись, з певними запізненнями. За рахунок цього, при експорті аудіо файлу, не очікувалось коректного оновлення даних і параметр лишався зі своїм попереднім значенням. Для виправлення цього, я прибрав оновлення значень параметрів, які прописував раніше, та знайшов метод TreeValueState, який за назвою параметра синхронно дістає значення параметра - getRawParameterValue. Таким чином, у методі оновлення фільтрів, я став використовувати значення, які діставалися синхронно, що давало фільтрам можливість коректно оновлюватись при автоматизації параметрів.

Завершальним кроком була зміна вигляду графічного інтерфейсу, адже він був не достатньо інтуїтивно зрозумілим. Для цього я додав підписи кожному з параметрів (Рис 4.8).



(Рис 4.8) Фінальний вигляд графічного інтерфейсу

4.5 Тестування

Свій застосунок я тестував використовуючи його для кількох різних робіт:

- Очистка аудіо від шуму
- Посилення, послаблення певних ділянок частот для збалансування
- Автоматизоване посилення або згасання аудіо

Для тестування *очистки* від шуму, я використав аудіофайл із голосом записаним не якісним мікрофоном. В цьому аудіо файлі було чути гул на низьких частотах (20-80Гц), гудіння на середніх (250-400Гц) та чіткий шум на високих(3-15кГц). Для очистки шуму на низьких частотах, я спробував використати high pass фільтр, проте в результаті він був занадто пологим та позбувався корисних частот. Натомість, спрацював band stop фільтр шириною в 6 октав, який зміг більш ефективно прибрати гул.

На середніх частотах знаходиться багато важливих деталей, тому для більшої точності, спершу, за допомогою band pass фільтру я знайшов частоти на яких шум був найгучнішим, та щоб не вирізати забагато, використав піковий фільтр шириною

1.5 октави щоб зменшити гучність на 5 децибел. Це допомогло трохи зменшити гучність шуму але не змогло прибрати його повністю.

На високих частотах шум чітко чути починаючи з 3 кГц. Через таку концентрацію шуму, його складно прибирати таким еквалайзером, бо можна зрізати багато важливого сигналу. Єдине що було можливо зробити - обрізати найбільш різкі частоти (близько 5.5кГц) за допомогою вузького пікового фільтру.

В результаті такої обробки, аудіо стало менш шумним, проте також стало звучати “плоско”. Ця проблема виникає через те, що деякі фільтри є занадто пологими, через що обрізаються важливі частоти. На програмному рівні, цю проблему можна було вирішити додавши можливість змінювати крутизну фільтру. У цьому ж тесті проблему неякісного звуку можна було виправити додавши ще один еквалайзер, який би підвищив важливі низькі та високі частоти, які могли бути послаблені. Такий варіант справді робить звук якіснішим, але шуми на високих частотах стають дуже помітними.

Загалом, очистка шуму потребує високої точності, чого не вистачає моєму плагіну. Також, цей процес був би зручнішим, якби мій плагін мав аналізатор частот. Без нього, можна використовувати аналізатор частот у вигляді окремого плагіну.

Для тестування *посилення та послаблення частот задля збалансування,* я використав інструментальний запис, який звучить дещо приглушено, особливо на нижніх частотах. Я посилив нижні частоти піковим фільтром близько 250Гц, та також застосував високий поличковий фільтр на частотах вище 3кГц. Останнє посилення також посилило високі частоти де стало чути статичний шум, тож третім піковим фільтром я послабив частоти близько 8кГц на 18Дб. Оскільки більшість фільтрів були посилюючі, я також зменшив головну гучність аби отримати приблизно той самий рівень, що і був на вході. Отриманий аудіо сигнал звучить більш повно та чітко. Отже, мій еквалайзер без проблем упорався зі збалансуванням аудіо.

Для тестування *автоматизованого згасання та посилення* аудіо, я використав low pass фільтр, для якого автоматизував частоту застосування, щоб вона з часом рухалась від низьких до високих. За рахунок цього фільтр з часом пропускає більше частот, і звук посилюється. Потім для згасання, я автоматизував форму іншого фільтру, щоб той із вимкненого стану перемкнувся у high pass фільтр, і також автоматизував його частоти від менших до більших. Таким чином, отримуємо протилежний ефект - згасання звуку, починаючи з низьких частот. Виходячи з цього тесту, мій плагін без проблем працює з автоматизацією.

Крім цього, з тестів я переконався, що додаток коректно працює зі збереженням та підвантаженням шаблонів та працює у інших DAW.

Підсумовуючи, серед переваг, мій плагін без проблем взаємодіє з DAW та гарно виконує загальні задачі як формування тону, балансу, автоматизованої фільтрації. Але серед недоліків, він може бути не підходящим для точних робіт, як усунення шуму. Для підвищення точності можливо додати реалізацію зміни крутизни фільтрів та додати аналізатор частот для наглядної роботи зі спектром частот. Також можливим покращенням була б власна математична реалізація фільтрів.

Висновки

У сучасному світі, зі зростанням попиту на створення мультимедіа, обробка аудіо як ніколи є актуальною. Вона присутня практично всюди і її неможливо уявити без еквалізації.

Впродовж курсової роботи, я дослідив як відбувається робота з цифровим аудіо у сучасних умовах. Також я детальніше дізнався як відбувається еквалізація, які існують типи фільтрації та які існують еквалайзери у вигляді додатків. В результаті, я розробив власний три смуговий семі-параметричний еквалайзер на основі обраного фреймворку JUCE.

Технічно, я ближче познайомився з обробкою звукового сигналу та отримав базові навички у роботі із фреймворком JUCE. Також я поглибив своє розуміння роботи цифрових звукових робочих станцій. Найголовнішим досягненням є сам додаток, який універсально працює у будь-якій DAW, яка підтримує формат VST. Усі задані технічні вимоги до проекту були виконані.

Я практично переконався у потужності фреймворку JUCE та можу відзначити, що він містить дуже обширний набір інструментів та детальну документацію, яка була надзвичайно корисною під час розробки власного додатку.

Також я усвідомив наскільки важливим є дослідження предметної області, адже на початку маючи лише тему роботи, я намагався відразу почати практичну частину, але мав дуже погане розуміння куди рухатися. Пошук та дослідження теоретичних матеріалів допомогли мені зрозуміти та набагато краще структурувати фронт робіт.

Список літератури

1. Еквалізація [Електронний ресурс], 03.05.2021 https://en.wikipedia.org/wiki/Equalization_(audio)
2. Цифровий звук [Електронний ресурс], 18.04.2021 https://en.wikipedia.org/wiki/Digital_audio
3. Joe Wolfe, What is a Sound Spectre [Електронний ресурс], 2005 http://newt.phys.unsw.edu.au/jw/sound.spectrum.html
4. ``Introduction to Digital Filters with Audio Applications", by Julius O. Smith III, (September 2007 Edition), What is a Filter [Електронний ресурс] https://ccrma.stanford.edu/~jos/filters/What_Filter.html
5. FIR filter basics [Електронний ресурс], 25.02.2017 https://dspguru.com/dsp/faqs/fir/basics/
6. IIR filter basics [Електронний ресурс], 25.02.2017 https://dspguru.com/dsp/faqs/iir/basics/
7. Digital Sound And Music [Електронний ресурс], chapter 7.3.1, 12.07.2014 http://digitalsoundandmusic.com/7-3-1-convolution-and-time-domain-filtering/
8. Цифровий фільтр [Електронний ресурс], 27.04.2021 https://en.wikipedia.org/wiki/Digital_filter
9. “Introduction to Digital Filters with Audio Applications”, by Julius O. Smith III, (September 2007 Edition), The simplest low pass filter [Електронний ресурс] https://ccrma.stanford.edu/~jos/fp/Simplest_Lowpass_Filter_I.html
10. Crossover basics [Електронний ресурс] 13.02.2016 https://speakermakersjourney.blogspot.com/2016/02/crossover-basics.html
11. What Is a Low Pass Filter? [Електронний ресурс] 12.05.2019 https://www.allaboutcircuits.com/technical-articles/low-pass-filter-tutorial-basics-passive-RC-filter/

12. A very brief introduction to Parametric Equalizaion [Електронний ресурс], 06.02.2018 https://www.tonmeister.ca/wordpress/2018/02/06/bo-tech-a-very-brief-introduction-to-parametric-equalisation/
13. WHAT ARE THE DIFFERENT TYPES OF EQ AND FILTERS? [Електронний ресурс], 19.07.2019 https://iconcollective.edu/types-of-eq/
14. Dynamic Equalizer [Електронний ресурс] 02.05.2018 https://soundbridge.io/dynamic-equalizer/
15. The human perception of loudness, sound pressure [Електронний ресурс], Eberhard Sengpiel, 01.08.2014 http://www.sengpielaudio.com/calculator-loudness.htm
16. Relation between Q and bandwidth BW [Електронний ресурс], Eberhard Sengpiel, 01.06.2014 http://www.sengpielaudio.com/calculator-bandwidth.htm