МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Алгоритмічне та програмне забезпечення
для розширення можливостей програвача медіафайлів»

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

Виконав: студент групи МгІТ-21

Данов Данило Сергійович

Науковий керівник к.т.н., доц. Колиско О.З.

Рецензент

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
_____Володимир ЩЕРБАНЬ.
"____" _____2023 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Данову Данилу Сергійовичу

1. Тема кваліфікаційної роботи Алгоритмічне та програмне забезпечення для розширення можливостей програвача медіафайлів, науковий керівник роботи Колиско Оксана Зенонівна, доц.,к.т.н. затверджені наказом КНУТД від "_12_"вересня_2023 року №_210-уч_

2. Вихідні дані до роботи: Розробки кафедри комп'ютерних наук; рекомендована література, додатки.

3. Зміст дипломної роботи: Вступ; РОЗДІЛ 1 Постановка задачі; РОЗДІЛ 2 Проектування; РОЗДІЛ 3 Програмна реалізація; Висновки; Список літератури; ДОДАТОК А Окремі фрагменти програмного коду; ДОДАТОК Б Презентація.

4. Дата видачі завдання 1 вересня 2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапу кваліфікаційної роботи	Орієнтовний термін виконання	Примітка про виконання
1	Вступ	*15.09.2023*	
2	Розділ 1. Постановка задачі	*20.09.2023*	
3	Розділ 2 Проектування	*30.09.2023*	
4	Розділ 3. Програмна реалізація	*10.10.2023*	
5	Висновки	*25.10.2023*	
6	Оформлення (чистовий варіант)	*1.11.2023*	
7	Подача кваліфікаційної роботи науковому керівнику для відгуку (за 14 днів до захисту)	*4.11.2023*	
8	Подача кваліфікаційної роботи для рецензування (за 12 днів до захисту)	*6.11.2023*	
9	Перевірка кваліфікаційної роботи на наявність ознак плагіату (за 10 днів до захисту)	*8.11.2023*	
10	Подання кваліфікаційної роботи на затвердження завідувачу кафедри (з 7 днів до захисту)	*10.11.2023*	

З завданням ознайомлений:

Студент _____ Данило ДАНОВ

Науковий керівник _____ Оксана КОЛИСКО

АНОТАЦІЯ

Данов Д.С. Алгоритмічне і програмне забезпечення для розширення можливостей програвачів медіа файлів. – Рукопис.

Магістерська робота за спеціальністю 122 Комп’ютерні науки. – Київський національний університет технологій та дизайну, Київ, 2023.

У роботі досліджено роботу класичних медіаплеєрів з аудіофайлами, їх обробкою та запропоновано враінти для роширення їх можливотей.

Проведено аналіз роботи з файлами, наявність вже існуючих засобів покращення звуку та їх роботу. Розглянуто механізми які б покращили обробку звучання звичайним користувачем.

Здійснено тестування і порівняльний аналіз до і після застовання методів оброки звуку па визначено їх плюси та мінуси.

Визначено напрями покращення аудіфайлів. Запропоновано рекомендації щодо підвищення їх якості та майбутні можливості розвитку методів обробки.

ABSTRACT

Danov D.S. Algorithmic and software for expanding the capabilities of media file players.

Master's degree work on the specialty 122 Computer science. - Kyiv National University of Technology and Design, Kyiv, 2023.

The paper examines the operation of classic media players with audio files, their processing, and offers options for expanding their capabilities.

An analysis of working with files, the presence of already existing means of sound improvement and their operation was carried out. Mechanisms that would improve sound processing by an ordinary user are considered.

Testing and comparative analysis was carried out before and after the applying methods of sound modification, and their pros and cons were determined.

Areas of improvement of audio files are defined. Recommendations for improving their quality and future opportunities for the development of processing methods are offered.

Зміст

ВСТУП

За останні роки людство досягло небувалого прогресу в технологіях, що не оминуло і пристрої виведення інформацію. Зростання популярності смартфонів, смарт-телевізорів, аудіосистем і інших мультимедійних пристроїв вимагає розробки програмного забезпечення для оптимальної взаємодії з цими пристроями. Це не могло не вплинути на розвиток програмної складової, яка має відповідати їх можливостям, а іноді навіть і призводить до данного прогресу. Також з данним розвитком збільшилася кількість та зросла популярность відео та аудіо контенту. Споживачі медіаконтенту вимагають високої якості, покращеного відтворення, більшого різноманіття та персоналізованих рекомендацій. Виходячи з цього зростають і стандарти якості: Висока якість аудіо і відео стає стандартом, і розробники програмного забезпечення постійно шукають способи покращити роздавану якість. Це вимагає постійного вдосконалення алгоритмів і програмного забезпечення.

Також не можна не згадати про стрімкий зріст полулярності Стрімінгових платформ: Платформи для стрімінгу великої кількості медіаконтенту, такі як Netflix, YouTube, Spotify і інші, які вже стали невід'ємною складовою життя багатьох людей і мають постійно оновлювати свої можливості задля задоволення потреб користувачів.

З появою штучного інтелекту і машинного навчання для людства відкрилися нові можливості що в данному разі дозволяє створювати більш ефективні алгоритми для автоматичного покращення медіаконтенту та його відтворення а також персональних рекомендацій на основі вподобань користувачів.

Також не можна не згадати що зі збільшенням якості аудіо, відео та іших файлів з'являється проблема в їх обсязі а саме завантаженні передачі та зберіганні при умовах обмеженої пам'яті на пристрої або ж слабкому підключенні до мережі інтернет, для чого потрібна розробка ефективних кодеків та алгоритмів стиснення яка дозволяє зберігати медіаконтент високої якості при менших обсягах даних.

З урахуванням цих факторів, дослідження та розвиток алгоритмічного та програмного забезпечення для медіапрогравачів залишається важливою областю для подальших досліджень та розвитку.

По цим причинам мною було розглянуто питання з розширення можливостей програвачів медіа файлів. Виходячи з потреб та можливостей більшості користувачів в нашій країні було проийняте рішення звернути особливу увагу на впровадження таких доповнень.

Алгоритми усуненя шуму, технологія компресії та збереження файлів, а також розробка простого інтерфейсу для данних доповнень застосунку.

Шумогасіння - це процес зменшення рівня шуму або видалення небажаного звуку з середовища.

Компре́сія звуку — це процес зменшення/стиснення динамічного діапазону звукового сигналу.

Сьогодні ми маємо велику кількість аудіо і відео файлів записаних у низькій якості або в умовах аудіо поміх та шумів, абож ж з використанням слабких неякісних пристроїв запису(відео записані на телефон GoPro в умовах поганої погоди, зовнійніх шумів тощо, записи на вбудованні або ж неякісні мікрофони). Тому мною було прийняте рішення вбудувати ефект усунення шумів та компресії в в плеєр задля покращеняя аудіо та відео файлів. Данні доповнення дозволять покращувати вже записанні аудіодоріжки низької якості без використання зовнішніх редакторів звуку що набагато спростить роботу над редагуванням медіафайлів за короткий проміжок часу в умовах домашнього/аматорського викостання. Адже більшості людей процес редагування аудіо файлів здається заважким та задовгим через те що зазвичай застосунки з редактування звуку та відео мають в собі набагато більше функцій та можливостей які не потрібні абож незрозумілі звичайному користувачеві без знань в обробці звуку і час на освоєння даннного програмного забеспечення займає багато часу, що просто не має сенсу при одноразовому використанні або ж користуванні в край малому

обсязі. Тому данні доповнення при наявності в стандартних аудіо та відео програвачах допоможуть користувачам при відсутності або ж мінімальних знаннях в обробці звуку покращити якість необхідних їм медіа файлів задля власного абож загального користування.

Інтерфейс. Інтерфейс — сукупність засобів, методів і правил взаємодії (керування, контролю, тощо) між елементами системи. Інтерфейс являється невід'ємною частиною кожного застосунку, а також основним методом взаємодії між людиною та програмою, при його низькій якості це призводить до проблем у використанні, довгої адаптації або ж взагалі відмови від використання застосунку.

Тому на сьогоднішній день основними характеристиками інтерфейсу є простота, інтуїтивність та зручність використання, а виходячи з того що мною була поставленна задача покращення саме стандартого медіаплеєра який є на кожному пристрої, данний інтерфейс має бути максимально простим та зручним у використанні буль яким користувачем.

Об'єктом дослідження є програмне забезпечення що дозволить додати редактори звуку в звичайні медіа програвачі задля спрощення обробки аудіофайлів.

Предметом дослідження є методи та алгоритми за допомогю яких редагується звук а також відбувається інтеграція в медіаплеєр шумогасників та редакторів звучання.

Метою дослідження є проаналізувати взаємодію між додатковим забезпеченням з програвачем та вимоги до нього.

Практичне значення дослідження полягає у вивченні стандартних проблем з аудіозаписами в різних умовах. А також порівняти на їх основі відсоток збільшення якості записів задля підкреслення ефективності данної розробки.

Наукова новизна проекту на мою думку полягає в автентичності – проведенні мною дослідження показали що на данний момент не має аудіоплеєрів з схожими можливостями.

Розширення кількості можливих ефектів – на данний момент мною розглядаються лише самі необхідні додатки, але при необхідності в майбутньому можна розширити їх спектр та покращити роботу.

Також можливість покращнння роботи згаданих вище ефектів – на сьогоднішній день штучний інтелект активно застосовується в різних сферах діяльності особливо обробки аудіо відео та фото тому в майбутньому може бути розглянута можливість задіяння його задля підвищення якості результатів.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка задачі

Метою даного проекту є дослідження вимог, а у підсумку – програмна реалізація додатку, який би додав в медіа плеєр можливість шумоподавлення та компресії і міг би зменшити часові затрати користувача, на редагування аудіо-доріжок, що як наслідок призведе до покращення якості розповсюджуваного контенту а також частично прибере необхідність в використанні сторонніх редакторів.

Звук - коливальне переміщення частинок, що поширюються у вигляді хвиль в середовищі.

Програвач мультимедіа або ж медіа-плеєр — тип комп'ютерної програми, призначений для відтворення файлів мультимедіа. Більшість програмних мультимедіа програвачів підтримують значну кількість медіа-форматів, включаючи відео та аудіо файли.

Деякі мультимедіа програвачі призначені лише для відтворення аудіо або відео файлів і називаються, відповідно, програвачі аудіо (аудіоплеєри) і програвачі відео (відеоплеєри). В основному всі відеоплеєри діляться на кілька типів: Прості (для роботи на слабких ПК), Універсальні та Спеціалізовані (специфічні і професійні програми для вирішення унікальних завдань). Приблизно за таким же принципом діляться і аудіоплеєри. Розробники таких програвачів прагнуть зробити їх якомога зручнішим для відтворення відповідних форматів.

Аудіоредактор або звуковий редактор — програмне забезпечення для редагування звуку. Редактори пристосовані для роботи з музичним матеріалом.

Функції аудіоредакторів можуть відрізнятися в залежності від їх призначення. Найпростіші з них, часто вільно поширювані, мають обмежені можливості по редагуванню звуку та мінімальну кількість підтримуваних аудіоформатів. Професійні пакети можуть включати багатодоріжковий запис, підтримку професійних звукових плат, синхронізацію з відео, розширений набір кодеків, величезну кількість ефектів як внутрішніх, так і плагінів.

Перетворення звуку

Основне призначення аудіоредактора — це зміна або редагування аудіосигналу. Більшість видів перетворень звуку прийшли з ери аналогового звукозапису, однак деякі з них стали можливі тільки із застосуванням цифрового представлення аудіоданих.

Найбільш поширеними є:

- перетворення амплітуди
  - посилення
  - зміна динамічного діапазону
  - мікшування
  - нормалізація
  - панорамування
- ефекти, засновані на затримці звуку
  - хорус
  - затримка
  - ефект відлуння
  - реверберація
  - фленжер
- фільтрація звукового сигналу
  - графічні і параметричні еквалайзери
  - фільтри
- реставрація звукового сигналу
  - шумозаглушення
  - придушення клацань в записах з пластинок
  - відновлення кліпованного сигналу
- зміна висоти тону та / або тривалості звучання
- зациклення фрагмента звуку

Як правило, функції аудіоредактора можливо розширити, завдяки використанню модулів — плагінів. Вони можуть містити один або кілька ефектів

і різнитися за якістю обробки або кількістю параметрів, що налаштовуються вбудованими інструментами редагування.

Проте як ми знаємо звук це коливання, тому додаток має включати в себе варіативність налаштування задля уникнення можливості псування звучання замість його покращення.

Яскравими прикладами в компресії звуку та видалення шуму є Ableton Live та Sony Sound Forge – два всесвітньовідомих редактори звуку та запису які були визнані музикантами та звукорежисерами. Данні застосунки мають широкий набір ефектів високої якості які можна підлаштувати під потреби будь якого користувача. Проте найбільшу увагу на мій погляд заслуговують функції Sony Noise Reduction, Ableton Glue та Noise Gate.

Sony Noise Reduction – функцією редактора є захоплення та видалення шумів з файлу з можливістю коригування. При наявності чотирьох алгоритмів шумоподавлення які можна підлаштувати при необхідності він виступає корисним та ефективним інструментом. Данні алгоритми мають в собі налаштовувані функції:

Noise Reduction редактор - відповідає за ступінь шумоподавлення. Чим більше значення виставляється тим сильніше подавляється шум, але разом з цим можуть з’явитися дефекти такі як булькання, зрізання частот або дивні зайві звуки. Оптимальними значеннями є від 10 до 20 дБ.

Noise bias – виставляє рівень кривої для захопленого шуму.

Attack speed – відповідає за швидкість спрацювання подавлення шуму на участках доріжки без шумів.

Release time – відповідає за швидкість спрацювання на участках з шумами.

Windowing FFT size – виставляє дискретність обробки шумодавлення. Чим вище буде значення тим вище точність обробки, але вище нагрузка на процесор.

Ableton Glue – симулює роботу аналового компресора SSLL G Bus і може використовуватися для динамічної обробки аудіо доріжки.

Параметр Attack (Атака) - значення цього параметра визначає, як швидко компресор почне обробку сигналу. Значення задається в мілісекундах, і чим різкіше атака інструмента, тим воно повинне бути менше.

Параметр Release визначає скільки часу потрібно компресору, щоб повернутись у вихідний стан після того як рівень аудіо сигналу впаде нижче порогового значення (Threshold). Коли ручка параметра Release переведена в авто-режим час «розкриття» автоматично адаптується до аудіо-сигналу. Авто-розкриття використовує два часових режими: повільний, що базується на налаштуваннях компресора і швидкий, що базується на змінах в аудіо сигналі. Оскільки авто-розкриття може надто повільно реагувати такі зміни, цей режим корисний в випадках якщо ви хочете досягти легкої компресії.

Параметр Dry/Wet регулює баланс між обробленим (аудіо сигнал із компресією) та не обробленим сигналом (аудіо сигнал без компресії). При 100% чутно оброблений компресором сигнал, при 0% ви почуєте не оброблений сигнал. Ще один спосіб керування рівнем компресії – ручка Range, яка визначає скільки компресії буде застосовано до аудіо сигналу. Значення між 60 і 70 Дб емулюють роботу аналогового обладнання, тоді як значення між 40 і 15 Дб можуть бути альтернативою налаштування Dry/Wet. За значення 0 Дб компресія до аудіо сигналу не застосовується.

Параметр Makeup дозволяє компенсувати втрачену після компресії гучність.

Виходячи з можливостей редакторів звуку описаних вище, сформовано наступні необхідні вимоги до додатку.

- Зрозумілість та зручність для нового користувача
- Налагодженість взаємодії процесів між собою
- Створення збалансованих пресетів налаштувань для користувачів

1.2. Розгляд редагування звуку в сучасному використанні

На сьогодній день більшість інформації різного характеру (наприклад научна абож розважальна) розповсюджується за допомогою аудіо або відео файлів. Кіностудії, музичні гурти та контент мейкери яких з зростанням

популярності стрімігових платформ стає все більше й більше, вже давно користуються звуковими редакторами та ефектами задля покращення створюваного контенту. Проте данні записи проходять серйозну або ж професійну обробку яка вимагає великий багаж знань в плані обробки та редагування звуку, що займає багато часу через вивчення звукової теорії, вибору необхідного редактора та адаптації до його інтерфейсу і можливостей.

Сучасний процес обробки звуку базується на використанні різних технологій і методів, включаючи цифрову обробку сигналу (Digital Signal Processing, DSP), машинне навчання, та інші. Його можна описати такою схемою:

- Збір аудіоданих – збір даних, які можуть бути отримані з мікрофонів, аудіорекордерів, інтернет-потоків, аудіофайлів тощо.
- Аналогово-цифрове перетворення (ADC): Сигнали з аналогових джерел перетворюються в цифровий формат за допомогою ADC. Це крок дозволяє комп'ютеру легко опрацьовувати аудіодані.
- Обробка сигналу: Цифрові сигнали піддаються обробці за допомогою методів DSP. Це включає в себе:
- Фільтрація та еквалізація: Фільтрація дозволяє виділити або приглушити певні частоти в аудіосигналі. Наприклад, можна використовувати низькочастотну фільтрацію для приглушення низьких частот, щоб зменшити баси. Еквалізація дозволяє змінити співвідношення між різними частотами для керування якістю звучання.
- Зменшення шуму: застосування засобів що приглушують, обрізають або ж видаляють шум. Це особливо важливо в медіафайлах, які записані в шумних середовищах або з різних джерел шуму.
- Зміна темпу та тону: зміна темпу (швидкість) та тону (висота) аудіосигналу для створення різних ефектів. Наприклад, збільшення темпу може зробити аудіо більш жвавим, а зміна тону може вплинути на висоту голосу або музики.

- Видалення артефактів: видалення небажаних артефактів або помилок у записі, таких як стуки, тріскотіння, звуки вітру, тощо.
- Спектральний аналіз: Цей процес включає аналіз частотного складу аудіосигналу, що може бути корисним для визначення специфічних характеристик музичних треків, таких як інструменти, гармонія, ритм і багато іншого.
- Ефекти та обробка звуку: додавання різноманітних ефектів, таких як ехо, реверберація, флангер та інші. Ці ефекти додають художню виразність до аудіоматеріалу.
- Витягання ознак (Feature Extraction): На цьому етапі витягуються характеристики аудіосигналу, які важливі для подальшого аналізу. Це може включати в себе визначення спектральних характеристик, ритмів, гучності тощо.
- Машинне навчання: Завдяки машинному навчанню, звук може бути класифікований, аналізований та інтерпретований. Також воно використовується для розпізнавання голосу, розпізнавання мовлення, розпізнавання аудіошуму, автоматичної транскрипції та інших завдань.
- Розпізнавання мовлення: системи розпізнавання мовлення використовуються в голосових асистентах, системах контролю, в транскрипції аудіо та багатьох інших областях.
- Генерація звуку: звук також може бути створеним за допомогою синтезу звуку. Системи синтезу звуку використовуються у голосових движках, музичних синтезаторах та інших додатках для створення аудіосигналів.
- Аналіз та інтерпретація музики: для музичних додатків, обробка звуку може включати в себе аналіз музичних параметрів, таких як ритм, темп, мелодія, гармонія та інші аспекти, що дозволяють класифікувати та оцінювати музику.

Такий підхід призводить до постійного прогресу в редагуванні медіафайлів, нових методів їх обробки та комбінацій ефектів що використовуються в данному процесі. Саме тому зараз ми можемо спостерігати

тенденцію що з кожним роком фільми, музика та контент стрімінгових платформ стають якіснішими та яскравішими в плані відтворення звуку.

Це призводить до зростання стандартів вимог якості користувачами і контент низької-середньої якості зазвичай погано сприймається або ж навіть оминається в багатьох випадках, що призводить до втрачання великої кількості контенту який іноді містить дійсно корисну інформацію щодо вирішення зачасту унікальних проблем та питань.

Віднедавна навіть навчальні заклади почали практикувати данний метод викладення матеріалу(записи лекцій, семінарів, тощо). Проте я виявив що більшість такої інформації перебуває в поганій якості, адже кожному закладу чи компанії потрібен час на закупівлю якісної апаратури для запису, а матеріали відзняті або записані в «польових» умовах навіть при їх наявності можуть мати в собі неприємні фактори, які можуть завадити сприйняттю інформації.

1.3. Огляд наявних рішень

Зовнішні редактори звуку.

На сьогоднішній день є велика кількість зовнішніх редакторів як платних так і безкоштовних. В них наявна величезна кількість плагінів та ефектів які можуть задовольнити майже будь якого користувача, а також можливість установки додаткового забезпечення для редагування та обробки. Гарним прикладом таких будуть згадані вище *Sony Sound Forge* та *Ableton Live*.

Що собою являє *Ableton Live* 11 Компанія Ableton має давню традицію поступово покращувати свою програму DAW, не змінюючи її радикально, що дуже ціниться її користувачами (Рис.1-5). Зберігаючи цю концепцію, нова версія Live 11 отримала велику кількість змін і покращень. Загалом їх можна розділити на три категорії: розширені можливості відстеження, нові корисні для живих виступів інструменти і оновлення наявних можливостей Live у генерації та обробки звуку, а також секвенсорного двигуна. Програмний пакет Live 11 Suite включає в себе чотири нові ефекти: PitchLoop89 (ефект зміни висоти тону, за мотивами примітивного цифрового пітч-шифтера Publison DHM 89 B2), Hybrid Reverb (гібридний ревербератор на основі алгоритмів та імпульсної згортки), а

також Spectral Resonator і Spectral Time (які відповідно створюють ефект резонансного фільтру та гранулярний ефект затримки). До пакету Live 11 Suite також входять три нові інструменти, створені у співпраці зі Spitfire Audio: Upright Piano, Brass Quartet і String Quartet, три нові звукові пакети Voice Box, Mood Reel і Drone Lab, а також шість інструментів та ефектів, заснованих на випадкових процесах та ймовірності.

Крім цього деякі вже наявні ефекти отримали кардинальні зміни. Так, значно розширилась функціональність Redux і Chorus, а Phaser і Flanger тепер інтегровані в єдиний більш потужний ефект, як це було з Simple Delay та Ping Pong Delay в 10-й версії Live. В Live тепер підтримується поліфонічна MIDI експресія MPE, яка дозволяє більш чітко контролювати сумісні інструменти. MPE можна використовувати без сумісного апаратного контролера, запрограмувавши його на новій вкладці Expressions в редакторі MIDI кліпів. Це також дозволяє використовувати експресію поліфонічного післядотику за допомогою контролера Push. Функціональність та робочі характеристики Для тих, хто використовує Live для запису MIDI та аудіо треків, з’явились нові можливості компонування та лінкування треків. Компонування працює просто: записуйте аудіо або MIDI в зацикленій частині проекту, і дані при цьому записуватимуться в єдиний кліп, але кожен цикл буде розміщений на новій доріжці, як окремий дубль. Доріжки дублів за допомогою функції Show Take Lanes відображаються під головним треком так само, як кілька смуг автоматизації.

Для компонування записаного матеріалу можна вибрати потрібний відрізок і, натиснувши Enter або ж просто перетягнувши частину кліпу, розмістити його на головній доріжці. Лінкування треків також виконується дуже просто. Потрібно виділити кілька треків, клацнути правою кнопкою миші на одному із їхніх заголовків і вибрати опцію “Link Tracks”. Тепер вибір однієї із доріжок відображатиметься і на всіх інших лінкованих доріжках, що дає змогу одночасно змінювати кілька аудіотреків, наприклад, мультитрекові записи вокалу чи ударних.

Що стосується живих виступів, Live тепер має можливість використовувати вхідний аудіосигнал для керування внутрішнім темпом всього проекту. Просто виберіть вхідний аудіоканал на панелі Preferences > Link/Tempo/MIDI, потім перемикніть Show Tempo Follower на Show, і ліворуч від кнопки “Tap” з’явиться кнопка “Follow”.

З’явились також нові можливості виконання послідовних дій (Follow Actions), вони тепер можуть бути пов’язані з довжиною кліпу, сцени тепер можуть мати слідування дій, а нова опція Jump тепер дозволяє перейти до певного кліпу. На перегляді майстер треку сесії тепер є кнопка глобального увімкнення цієї функції Enable Follow Actions Globally.

Ще однією новою функцією, яка стане корисною в живих виступах, є доріжка ймовірностей в MIDI-редакторі, яка контролює ймовірність відтворення певноих ноти. Також можна встановити діапазон для MIDI параметра velocity, що дозволяє створювати більш природні звучання для живих виступів.

Розширилась функціональність макросів, в одному треці можна розмістити від 1го до 16ти макросів, а макроси снепшотів тепер дозволяють створювати та викликати кілька налаштувань одночасно, які також можна рандомізувати, створюючи нові звучання.

Хоча бувалим користувачам Live потрібно буде трохи часу на адаптацію до деяких змін (адже аудіоефекти тепер знаходяться в секціальних папках, які також включають модулятори Max for Live, і вигляд MIDI та аудіокліпів також оновився), команді Ableton вкотре вдалося розширити можливості застосунку, не ускладнюючи робочий процес цієї програми, відомої своєю простотою та зрозумілістю.

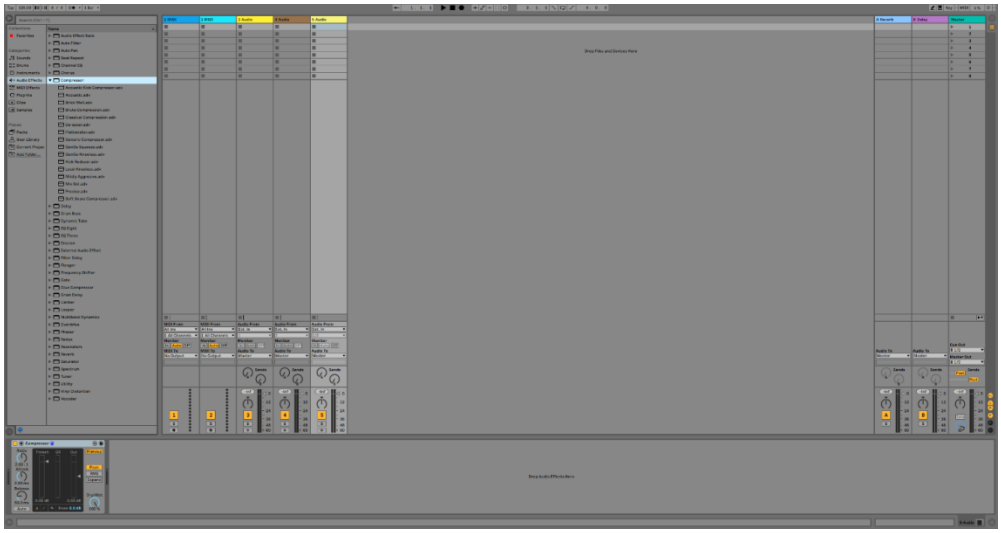

Рис.1 Загальний вигляд програми.

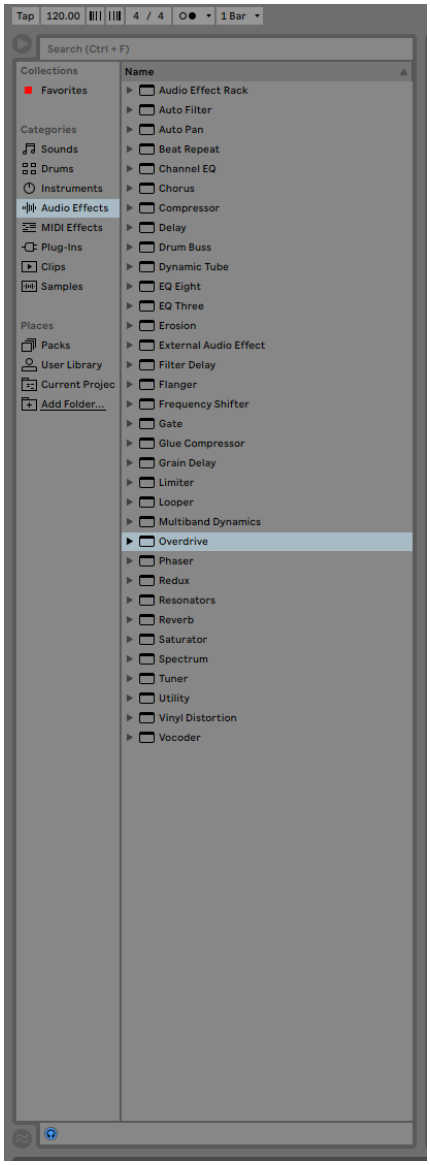

Рис.2 Вигляд меню плагінів та ефектів.

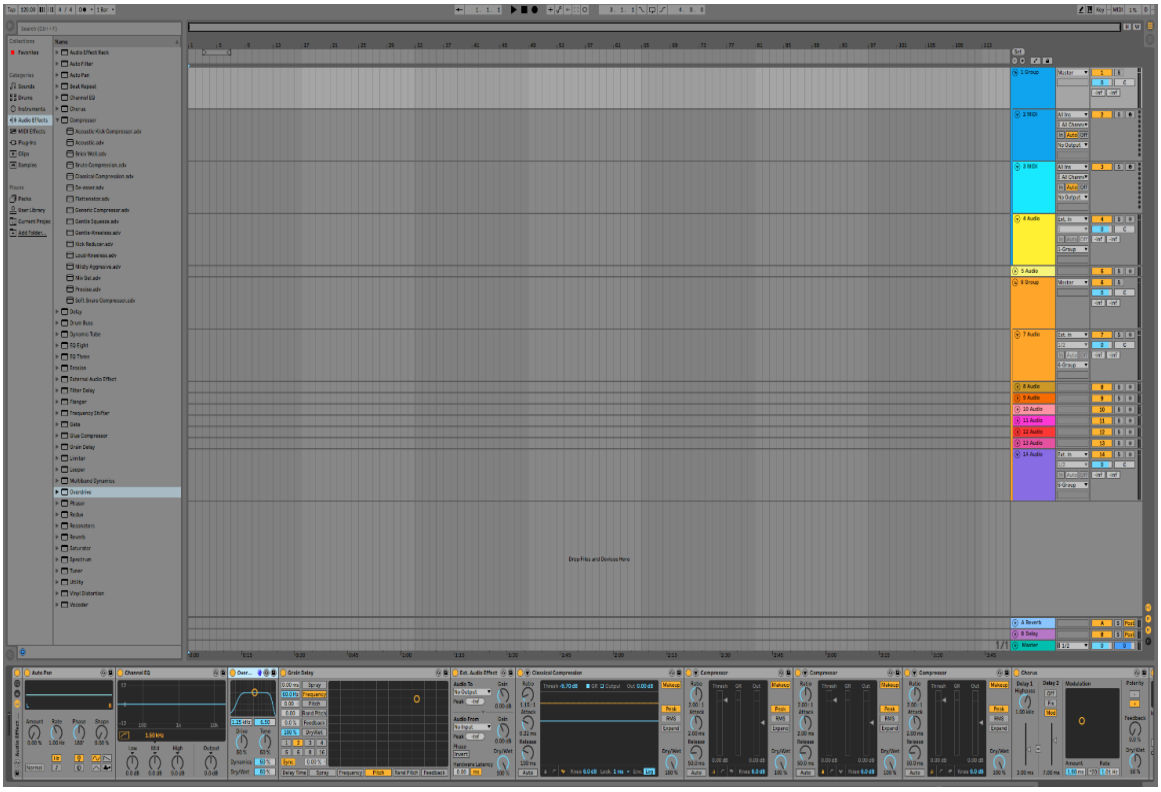

Рис.3 Вигляд програми з відкритими плагінами.

Рис.4 Вигляд програми з готовим проектом.

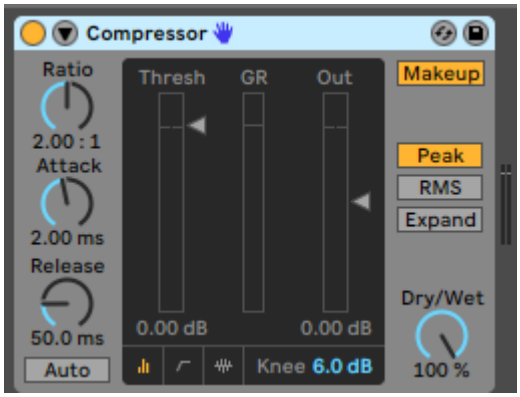

Рис.5 Вигляд компресору

Sony Sound Forge

Спочатку варто згадати про те, що може служити вихідним матеріалом для роботи з програмою. Це можуть бути записи у цифровому вигляді, отримані з диктофона, плеєра, жорсткого диска і т. д. - додаток підтримує величезний спектр форматів аудіо: як найбільш популярні MP3, WAV, WMA, Ogg Vorbis та ін., так і спеціалізовані типи файлів, створені в інших продуктах Sony для роботи зі звуком. При необхідності можна витягти доріжки з компакт-дисків - для цього необхідно скористатися пунктом Extract Audio from CD меню File.

Крім того, є можливість самому записати звукову доріжку підключивши мікрофон, безпосередньо в Sound Forge. Причому модуль запису при всій простоті реалізації надає досить обширні можливості налаштування. Ще одна цікава можливість програми - редагування звукової доріжки відеоролика. Звуковий супровід відкривається програмою як звичайний аудіофайл.

Для запису з мікрофона, як вже згадувалося, в Sound Forge передбачений окремий і досить простий у роботі модуль, який викликається кнопкою Record на панелі інструментів програми або комбінацією клавіш Ctrl+R. Прямо у вікні, можна побачити кнопки управління записом, а також шкалу інтенсивності звукової хвилі. При натисненні на червону кнопку, що запускає процес запису, програма автоматично створить нову доріжку для редагування і помістить в неї записаний матеріал для редагування. На вибір користувачу пропонується декілька режимів запису (список, що випадає Mode): звичайний режим за умовчанням (Automatic retake (automatically rewind)), створення безлічі варіантів запису що розташовані один за одним і зберігаються в одному звуковому файлі

який автоматично розділений на області (Multiple takes creating Regions) і без створення окремих областей для кожного варіанта (Multiple takes (no Regions)), запис кожної частини в окремому вікні (Create a new window for each take), а також запис інтервалів певної довжини (Punch-In (record a specific length)). В останньому випадку необхідно вказати час початку і закінчення запису. При цьому інша частина доріжки залишається незмінною - це зручно, коли необхідно переписати лише якийсь певну частину матеріалу.

Коли запис готовий, можна приступити до його редагування. У Sound Forge передбачено чотири основні інструменти для редагування звукових доріжок:

- Edit Tool призначений для виділення потрібних ділянок запису
- Magnify Tool служить для зміни масштабу звукової хвилі
- Pencil Tool дозволяє вручну малювати звукову хвилю (працює лише при сильному збільшенні масштабу)
- Event Tool дає можливость переміщати виділені області. Перемикатися між ними можна або на панелі інструментів програми клацанням по відповідній кожному засобу кнопці, або ж безпосередньо у вікні редагованого семплу.

Крім того, в Sound Forge є допоміжні об'єкти, які полегшують редагування звукової доріжки. Так щоб відзначити будь-який необхідний момент семплу, можна додати на доріжку маркер. А виділити певну ділянку на доріжці допоможе опція Область. Варто зазначити, що виділення частини доріжки між двома маркерами або в області здійснюється просто подвійним клацанням миші.

Одна з переваг Sound Forge полягає в тому, що багато завдань, що стосуються обробки аудіофайлу автоматизовані і вам не доведеться витрачати багато часу для редагування доріжки вручну. Наприклад прибрати паузи і ділянки з тишею дозволить інструмент Auto Trim / Crop в меню Process. Працює ця функція таким чином, що програма знайшовши частину даних, рівень сигналу яких вище ніж зазначений, і розцінює їх як прийнятну, а коли рівень сигналу падає нижче вказаного вами, Sound Forge розцінює таку частину як кінець

робочої секції, або початок фрагмента тиші. Потім програма шукає наступну секцію, що має необхідний рівень сигналу і видаляє все між двома секціями.

Auto Trim / Crop дозволяє використовувати декілька варіантів для очистки запису. Так Keep edges outside of the selection видаляє фрагменти тиші всередині виділеної області, не зачіпаючи дані поза цією ділянкою, Remove edges outsde of the selection вимикає паузи в межах виділеної області, а також видаляє всі дані, що знаходяться за її межами. Remove Silence between phrases (creates regions) видалить фрагменти тиші між фразами (при запису голосу), а Remove data from start and limit file length видалить тишу на початку звукового файлу, а також може обрізати кінець запису у зазначеній точці. Крім того, в списку Preset є кілька попередньо встановлених налаштувань для найбільш розповсюджених завдань.

Для створення ефекту перетікання однієї частини запису в іншу або плавного затухання можна скористатися функцією Fade. Регулювати гучність треку або окремих його частин можна за допомогою інструменту Volume, вирівняти частини запису різної якості дозволяє Normalize. А функція Mute допоможе повністю відключити звук у непотрібних ділянках запису.

Досить важливим інструментом є Bit-Depth Converter, що змінює значення розрядності аудіо файлу в бітах. Перед обробкою запису рекомендується підвищувати його розрядність - це не спричинить поліпшення якості доріжки, але збільшить її роздільну здатність, і подальше редагування та обробка файлу не призведуть до появи шумів. Зниження ж розрядності файлу понизить і його якість, проте в деяких випадках доводиться йти на такий крок - наприклад, щоб записати на компакт-диск 24-бітний файл, доведеться знизити його розрядність до 16 біт, оскільки аналоговий компакт-диск може використовувати тільки таку розрядність. В таких випадках варто залишати копію оригінального файлу - про всяк випадок.

Ще однією корисною функцією є можливість стиснення / розтягування запису за часом без змінення висоти тону. Це корисно, наприклад, при коригуванні звукової доріжки до відеоряду. Зробити це можна за допомогою

інструменту Time Stretch. Програма надає 19 різних режимів (Mode), які призначені для конкретного типу аудіоданих, які ви обробляєте. У вікні налаштування Time Stretch необхідно встановити в потрібну позицію на шкалі, спеціальний індикатор, який показує наскільки витягнулася або стиснулася доріжка.

Програма Sound Forge оснащена великим набором ефектів які можна застосовувати як до всього запису в цілому, так і до окремих її частин. Найбільш цікаві з них: реверберація, створює набір ефектів відлуння (Reverb), ефект трелі за рахунок вібрації висоти тону (Vibrato), спотворення звуку для вокалу або гітарних партій (Distortion), додавання «космічних» або «свистячих» звуків, характерних для психоделічної музики 1960х - 1970х років (Flange / Wah-Wah), ефекти хорового звучання (Chorus) і акустичного колориту (Acoustic Mirror) і т. д. Кожен з них потребує самостійного налаштування, тому на початку для отримання бажаного результату вам напевно доведеться використовувати метод наукового тику J. Благо, в налаштуваннях кожного з ефектів є функція попереднього перегляду дії. Крім того, деякі пункти налаштувань є схожими для більшості ефектів. Наприклад, Dry out - гучність «сухого», не обробленого ефектом сигналу, Wet out - гучність ефекту, Rate - швидкість модуляції, Depth - амплітуда або ж діапазон модуляції і т. д. Щоб краще розібратися в налаштуваннях ефектів, варто звернутися до довідки програми.

Коли обробка звуку завершена, отриману в результаті доріжку можна зберегти на жорсткому диску в один з багатьох форматів що підтримуваних програмою, з вибором якості треку (бітрейту і частоти дискретизації). Якщо ж планується подальша обробка треку, є сенс зберегти його як проект Sound Forge - файл із розширенням FRG. Це дозволить програмі в майбутньому швидше завантажити композицію для її подальшої обробки.

Готовий звуковий файл можна також записати на диск. Для цього в розділі меню Tools необхідно вибрати один з варіантів: Burn Track-at-Once Audio CD або Burn Disc-at-Once Audio CD.

Перший висновок, що можна зробити зі знайомства з Sony Sound Forge, полягає в тому, що програма не дуже складна в освоєнні - багато речей інтуїтивно зрозумілі навіть тим користувачам, які бачать програму вперше, не кажучи вже про всіх тих, хто працював з попередніми версіями програми. Однак для того щоб виконувати будь-які складні процеси з обробки звуку, однієї інтуїції недостатньо - треба буде запастися детальним посібником до застосунку, а також загальними підручниками по роботі зі звуком. Для простого редагування та обробки аудіо, наприклад нарізання рингтонів, з'єднання двох треків або ж застосування пари ефектів до звукового файлу все-таки краще використовувати безкоштовні звукові редактори. Не через те що Sound Forge не може цього робити, а з тієї причини, що додаток орієнтований на користувачів для яких обробка звуку - серйозне хобі, а також на людей, які займаються написанням музики у комерційних цілях. Та й ціна продукту, зазначена на сайті виробника (від $ 375), чітко натякає на цей факт.

Ефекти на звукові доріжки в Sound Forge можна застосовувати не тільки по одному, але й задавати з них послідовність. Для цього в програмі є модуль Plug-in Chainer. Для його запуску необхідно натиснути на кнопку у вікні редагування семпла. Натиснувши на кнопку Add Plug-Ins to chain, ви отримаєте доступ до менеджера ефектів і плагінів, з яких можете вибрати потрібні вам на данний момент. Після побудови послідовності ефектів кожен з них можна буде налаштувати окремо лише клацнувши на відповідну йому кнопку в ній. Також перед застосуванням ефектів можна буде попередньо оцінити, який вплив вони зроблять на запис.

В Sound Forge є і спеціальний модуль для автоматизованої обробки багатьох файлів - Batch Converter, що дозволяє переконвертувати кілька записів з одного формату в інший. Крім того цей модуль дає можливість обробляти групи файлів плагінами - наприклад, видалити шум з 20 файлів або нормалізувати їх рівень під необхідний параметр. Вдобавок Batch Converter підтримує групове перейменування файлів та внесення додаткової інформації (метаданих).

При записі дисків в Sony Sound Forge можна скористатися одним з двох режимів: Track-at-once (TAO) або Disc-at-once (DAO). Різниця між ними з технічної точки зору полягає в тому, що в першому випадку лазером записується за раз по одному треку, і між кожною композицією він вимикається, а в іншому - пише весь масив аудіоданих за раз. Що ж це дає з точки зору пересічного користувача? При методі запису TAO можна як фіналізувати диск, так і записати його без фіналізації, що дозволяє в подальшому додати на інші дані. За рахунок того, що між треками відбувається відключення лазера, виникають невеликі паузи, в результаті чого на деяких пристроях відтворення звуку між композиціями можуть лунати неприємні звуки. При використанні DAO диск автоматично фіналізується, і додавати на нього данні вподальшому не вийде. У Sound Forge режим Disc-at-once дозволяє також вбудовувати додаткову графіку і текст, додавати між треками різні інтродукції. Взагалі вважається, що саме цей режим якнайкраще підходить для запису концертних виступів наживо і створення майстер-дисків для подальшого тиражування.

Ручна автоматизація

У Sound Forge присутній спеціальний модуль Script Editor, в якому є можливість не тільки написати необхідний програмний код, але відкомпілювати його і запустити. Це означає, що якщо ви володієте можливостями написання програмного коду на C, Visual Basic або Java, то можете самостійно автоматизувати процеси в програмі для подальшого полегшення роботи з нею. Наприклад, розбити трек безліччю маркерів, що стоять через певні інтервали, імпортувати треки з CD і відразу їх переконвертувати в який-небудь доступний формат, створити певну кількість аудіофайлів і т. д. Такий підхід дозволяє заощадити безліч часу.

Інтерфейс Sony Sound Forge можна гнучко налаштувати - практично всі основні функції програми відображені кнопками на 12 панелях інструментів, розміщення яких можна підлаштовувати індивідуально.

Рис.4 Загальний вигляд додатку з відкритим проектом.

Рис.5 Панель Standard повторює основні пункти меню File і Edit, а також має набір з чотирьох різних типів курсору редагування.

Рис.6 Панель Transport: командне меню - пауза, перемотування, запис, відтворення і зупинка.

Рис.7 Панель Navigation: управління у вікні редагування звукової хвилі - масштабування, встановлення маркерів, швидка навігація.

Рис.8 Панель Views дозволяють запам'ятовувати поточні візуальні налаштування вікна редагування звукової хвилі, включаючи положення курсору, масштаб і т.п. Дана панель дуже корисна при роботі з файлами великого обсягу.

Рис.9 Панель Status / Selection: перехід від однієї тимчасової сітки до іншої, тобто відображення значень за семплом, секундами і хвилинами і різними протоколами синхронізації.

Рис.10 Панель Regions / Playlist: надає доступ до вікон зі списками регіонів і плейлисту, керування синхронізацією MIDI і відображення статусу синхронізації.

Рис.11 Панель Process повторює двадцять функцій однойменного пункту головного меню і є одним з ключових вікон при роботі з Sound Forge

Рис.12 Панель Effects повторює ключові функції однойменного пункту головного меню.

Рис.13 Панель Tools повторює ключові функції однойменного пункту головного меню, серед яких особливо варто відзначити запис CD та імпортування аудіотреків з дисків.

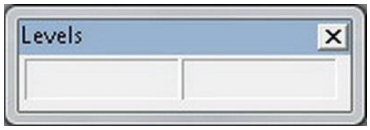

Рис.14 Панель Levels: цифровий індикатор, який показує поточне значення обраного параметра в місці розташування курсору (sample value, percent, dB, Peak, RMS Power).

Рис.15 Панель Insert: вставка об'єктів розмітки, таких як маркери, області і т. д.

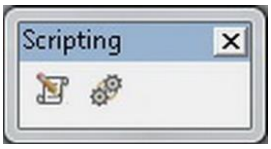

Рис.16 Панель Scripting дозволяє управляти роботою модуля створення скриптів для автоматизації роботи з програмою.

З головних переваг таких редакторів я б виділив:

- Варіативність плагінів та ефектів.
- Високу якість обробленого звуку.
- Велику кількість підтримуваних форматів звуку.

Недоліками ж я вважаю:

- Відносно складний в освоєні інтерфейс.
- Складність в налаштуванні ефектів та плагінів, а також їх підключення для пересічного користувача.
- Необхідність в знаннях щодо звукообробки та взвємодії між ефектами.
- Велика вартість (майже 400$ кожна)

Для початку я хочу розробити програму в вигляді плагіну. На мій погляд на початковій стадії це зручний варіант який в майбутньому можна буде повноцінно інтегрувати в медіаплеєр, можна легко протестувати його функціональність на існуючих аудіоредакторах, а також через те що деяких медіаплеєрів немає в open source, що дуже ускладнює роботу над моїм проектом. При розробці програмного продукту я намагався з одного боку поєднати всі переваги вище наведених програм, а з іншого врахувати всі їх недоліки та мінімізувати їх але з певними змінами адже мій плагін зорієнтований в основному на звичайного користувача: Наприклад, на відміну від величезної кількості ефектів в Ableton Live, звичайному користувачеві достатньо мати

компресор та шумодавлення, тому я зосередив свою увагу саме на них, щоб не розсіювати увагу користувачів на інші плагіни для яких потрібні певні спеціальні знання. Також саме через це було створенно два режими роботи автоматичний в якому просто обирається пресет ефекту що накладається та ручний де можна за потреби детальніше їх налаштувати.

РОЗДІЛ 2. МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Інструментальні засоби

Для моделювання майбутньої системи було обрано об’єктно-орієнтований підхід, оскільки, на мою думку, за такого підходу система стає зрозумілішою з технічної точки зору. Такой для спрщення роботи було обрано фреймворк JUCE. Існують різні підходи до розробки програмного забезпечення. Одним із найбільш популярних та ефективних є об’єктно-орієнтоване програмування (ООП). З його допомогою можна створювати, масштабувати та підтримувати досить складні проекти.

Коротко зупинимся на тому, що таке об’єктно-орієнтоване програмування, для чого воно потрібне, які його принципи та його плюси та мінуси, але спочатку розберемося з його попередником.

До появи ООП, щсновним підходом до розробки було процедурне програмування. У ньому програма розбивається на набір функцій та працює послідовно, виконуючи інструкції в строгому порядку. Це зручно для написання невеликих застосунків або скриптів, які виконують прості дії, як-от читання і запис даних, сортування тощо.

У разі складного софту з цим можуть виникнути проблеми. По-перше, зв’язки між окремими функціями не завжди очевидні. По-друге, в процедурному програмуванні не має механізмів для повторного використання коду. Кожна функція виконує своє окреме завдання, і немає способу використовувати її в інших частинах ПЗ. Це сильно ускладнює розробку, підтримку та розширення застосунків при зміні вимог. ООП позбавлене цих недоліків і дозволяє коригувати об’єкт, який є ключовим елементом програми.

ООП – це підхід до розробки програмного забезпечення що зосереджений на об’єктах, а не на функціях. Тобто програма розбивається на набір об’єктів, які взаємодіють між собою.

Структура ООП складається з класів, об’єктів, атрибутів та методів. Клас визначає абстрактні характеристики деякої сутності, включно з характеристиками самої сутності (її атрибутами або властивостями) та діями, які

вона здатна виконувати а саме її поведінкою, методами або можливостями. Наприклад, клас «Кіт» може характеризуватись рисами, притаманними всім котам, зокрема: порода, колір хутра, здатність нявкати. Класи вносять модульність та структурованість в об'єкто-орієнтовану програму. Зазвичай клас має бути зрозумілим людей для не-програмістів, що знаються на предметній області, що своєю чергою значить що клас повинен мати значення в контексті. Також, код реалізації класу має бути досить самодостатнім. Властивості та методи класу, разом називаються його членами.

Об'єкт – окремий екземпляр класу (створюється після запуску програми та ініціалізації полів класу). Клас Кіт відповідає всім котам шляхом опису їхніх спільних рис; об'єкт Пушок є одним окремим котом, окремим варіантом значень характеристик. Кіт має хутро; Пушок має біле хутро. Об'єкт Пушок є екземпляром (примірником) класу Кіт. Сукупність значень атрибутів окремого об'єкта називається станом. На основі цього класу можна, також, створити інший об'єкт Мурзік, який відрізнятиметься від об'єкта Пушок своїм станом (наприклад кольором хутра). Обидва об'єкти (Пушок і Мурзік) є екземплярами класу Кіт.

Методи це можливості об'єкта. Оскільки Пушок — Кіт, він може нявкати. Тому нявкати() є одним із методів об'єкта Пушок. Він може мати й інші методи, зокрема: лежати(), або їсти(). В межах програми, використання методу має впливати лише на один об'єкт; всі Коти можуть нявкати, але треба, щоб нявкав лише один окремий кіт.

Ось приклад для розуміння структури об’єктно-орієнтованого програмування:

Клас: музикант

Об’єкт: музикант Тоні

Атрибути: зарплата та обов’язки

Методи: створення музики

Як можна побачити, атрибути та методи тут є властивостями об’єкта. Такий підхід дає змогу спростити розробку комплексного ПЗ і писати гарно структурований код, з яким приємно та зручно працювати.

Принципи об’єктно-орієнтованого програмування. ООП грунтується на чотирьох основних принципах: інкапсуляція, успадкування, поліморфізм та абстракція. Далі вони будуть описані трохи детальніше.

Інкапсуляція – це данні та методи, пов’язані з ними, зберігаються всередині об’єкта. Це дає змогу приховувати його внутрішню реалізацію і надавати тільки інтерфейс необхідний для взаємодії. Так розробник може створювати надійно захищені програми, запобігаючи несанкціонованому доступу. Кібербезпека наразі є дуже важливою складовою.

Успадкування. В ООП можна створювати нові класи на основі вже існуючих. Новий клас називається похідним, а наявний – базовим або батьківським. Під час успадкування похідний клас отримує всі атрибути та методи базового і може додавати свої власні. Спадкування дає можливість повторного використання коду та створювання ієрархії класів «від загального до приватного» для реалізації складних схем. Це сприяє підвищенню ефективності розробки ПЗ та забезпечує більш логічну організацію коду. Не потрібно постійно переписувати однакові властивості для різних об’єктів, достатньо успадкувати їх від одного батьківського.

Поліморфізм. Об’єкти різних класів можуть мати однакові методи, але реалізовувати їх різними способами. Програма може працювати з усіма об’єктами використовуючи загальний інтерфейс, що робить код гнучкішим та універсальнішим.

Абстракція. Замість того щоб детально описувати кожну частину системи, абстракція фокусується на найважливішій її складовій. Вона дозволяє розробникам приховати складність реалізації та зосередитися на ключових аспектах об’єкта.

ООП здається ідеальним підходом. Але все таки має і свої недоліки.

Переваги та недоліки об’єктно-орієнтованого програмування. Плюси об’єктно-орієнтованого програмування є основною з причин його широкого застосування в різних галузях розробки. Ось деякі з них:

Модульність і повторне використання коду. ООП дає змогу розділити програму на невеликі та зрозумілі частини, які відповідають за певну функціональність. Модулі, класи та об’єкти можуть бути використані повторно, що сильно спрощує розробку, налаштування та підтримку ПЗ.

Гнучкість і масштабованість.За допомогою ООП програма стає гнучкішою і більш розширюваною, що дає змогу додавати нові класи і методи без необхідності переписування всього коду. Це забезпечує можливість адаптації софта до мінливих вимог.

Простота супроводу. Кожен об’єкт має свою власну функціональність і дані, що спрощує пошук та виправлення помилок. Код стає зрозумілішим, тому тестувальникам і програмістам простіше працювати з ним.

Безпека. Софт з інкапсульованим кодом складніше зламати.

Легка інтеграція. ООП дозволяє швидко інтегрувати різні компоненти програми, створюючи об’єкти які взаємодіють між собою. Це сильно спрощує розробку складних систем.

Попри безлічі переваг об’єктно-орієнтоване програмування має деякі недоліки, які слід врахувати під час розробки ПЗ:

Складніша крива навчання порівняно з процедурним програмуванням. Буде потрібно більше часу та зусилль щоб повністю освоїти концепції ООП і застосовувати їх на практиці.

Можливість дублювання коду і надмірності. У разі неправильного проєктування класів та об’єктів можуть виникнути ситуації, коли одна й та сама функціональність реалізована в декількох частинах коду що призводить до надмірності й складності обслуговування.

Перевантаження пам’яті та обробки даних. Класи та об’єкти потребують більше пам’яті ніж прості процедурні структури даних, оскільки вони містять додаткову інформацію, таку як методи та властивості.

ООП може бути корисним для більшості проєктів. Але розробники мусять враховувати всі фактори при виборі парадигми програмування, щоб забезпечити оптимальну продуктивність своїх рішень.

Приклади використання принципів ООП. Принципи об’єктно-орієнтованого програмування широко застосовуються в різних галузях розробки. Ось декілька прикладів із реального життя:

Веброзробка. Класи, об’єкти, успадкування і поліморфізм дають змогу заводити різні типи користувачів, товарів, замовлень та інших сутностей на основі загальних шаблонів. Це корисно при створенні соціальних мереж або інтернет-магазинів.

Розробка ігор. Щоб створювати персонажів, а також предмети, локації та інші елементи із загальними характеристиками.

Медицина. ООП застосовується для розробки медичних інформаційних систем, які зберігають і опрацьовують данні. Класи та об’єкти представляють пацієнтів, лікарів, ліки тощо. Інкапсуляція забезпечує безпечне зберігання інформації та доступ до неї.

Тепер розглянемо про фреймворки на прикладі веб-розробки.

Так, будь-який цифровий продукт можна написати чистим кодом, щоправда на сьогоднішній день це робиться вкрай рідко і це відбувається в особливих вмпадках. Насправді під кожну мову програмування вже вже вклика кількисть готових рішенн та бібліотек. Це означає, що розробникам не потрібно «винаходити велосипед». Вони можуть просто зібрати потрібний продукт із готових і напівготових вже створених деталей. Тобто користуватися фреймворками — готове робоче середовище для розробки, що пропонує використання готових наборів інструментів та структур. Все, що потрібно розробникам — налаштувати його під себе та свої потреби.

2.2 Розгляд фреймворку JUCE

Фреймворк (framework) – це набір бібліотек, інструментів, та правил, який використовується для створення додатків. Він зазвичай являє собою структуру, яка визначає як компоненти програми мають взаємодіяти між собою, які шаблони використовувати для створення інтерфейсу і які методи використовувати для роботи з базами даних та іншими зовнішніми ресурсами.

Фреймворк призначений для спрощення розробки додатків, оскільки він надає вже готові рішення для поширених завдань. Це дозволяє програмістам зосередитися на створенні логіки програми, а не витрачати час на написання коду для вирішення спільних завдань. Крім того він може бути спеціалізованим для певного типу програм, наприклад для веб-програм або мобільного софту. Він також може надавати інструменти управління проектом, такі як система контролю версій або інтеграція з іншими інструментами.

Навіщо потрібні фреймворки

Почнемо з того, що найчастіше Frameworks використовують для створення веб-додатків і для веб-дизайну. Вони чудово підходять як для розробки простих, так і для об’ємних корпоративних продуктів що мають складну логіку.

Основне завдання фреймворку — допомогти правильно налаштувати робочі процеси та побудувати бізнес-логіку. Багато комерційних проектів розроблено на основі нескладних фреймворків. Наприклад за їх допомогою не складе труднощів створити інтернет-магазин.

Звичайно, фреймворки не є універсальним рішенням для всіх завдань і іноді може знадобитися написати додаток «з нуля», щоб задовольнити свої або клієнтські унікальні вимоги. Однак у більшості випадків використання фреймворків дозволяє сильно прискорити та спростити процес розробки та створитення надійного та ефективного додатку.

Frameworks мають високу продуктивність і гнучкість. Це дозволяє отримати якісний каркас майбутнього продукту без втрати його функціоналу.

У результаті можна зробити висновок про можливості фреймворку:

- спрощують та прискорюють процес розробки
- реалізують базову функціональність веб-продукту
- прискорюють процес роботи
- знижують ймовірність виникнення помилок

Люди, які не працювали з фреймворками іноді плутають їх з бібліотеками, але насправді це різні інструменти. Бібліотека містить в собі певний набір

функцій, що дозволяє вирішити конкретне завдання певної галузі. Наприклад, бувають бібліотеки для роботи з датою або часом, HTTP-запитами та ін.

Фреймворк визначає архітектуру програми, забезпечуючи взаємодію між його компонентами. При цьому фреймворк може містити одразу кілька бібліотек. До речі, саме він встановлює, коли викликати функції користувача.

Особливості веб-фреймворків

З вищенаписаного можна зробити висновок, що таке фреймворки. Кожен із них має свої унікальні особливості. Це дозволяє розробникам підібрати конкретний стек технологій. Але при цьому в них є якості, які загалом поєднують будь-який фреймворк. Про них ми поговоримо трохи докладніше.

Універсальність

Фреймворки дають єдині стандарти розробки. Це сильно полегшує розвиток будь-якого проекту. Якщо ваш сайт або програма написана на поширеному фреймворку, то можна з легкістю знайти розробника/фахівця для роботи з ним. А все через те, що вони зобов’язують працювати з бібліотеками за конкретними правилами.

Ефективність та простота

Якщо ви хочете дійсно унікальний продукт, то доведеться написати код із нуля. Але саме використання Frameworks прискорює процес розробки. Та й з точки зору фінансів такий підхід явно коштуватиме дешевше. Все, що потрібно зробити, це доповнити готову архітектуру унікальними інструментами.

Завдяки використанню фреймворків можна швидко вивести готовий продукт на ринок. А наявність стандартизованої кодової бази сильно допоможе уникнути помилок під час розробки. До речі, це полегшує роботу не лише розробникам, а й тестувальникам.

Надійність

Хоча самописні сайти унікальні самі по собі, у них є одна вагома вада — чистота коду. Домогтися її вкрай важко, особливо якщо над проектом у різні періоди часу працює кілька команд. А ось у випадку з фреймворками ця

проблема легко вирішується. Шукати помилки тут набагато простіше, та й загалом процес тестування відбувається значно швидше.

Безпека

Ця особливість є основною їх перевагою. Усі фреймворки надають розробниками доступ до бібліотек та інструментів захисту даних, де дотримуються високі стандарти кібербезпеки. Вони надають механізми захисту від атак, таких як SQL-ін'єкції, міжсайтовий скриптинг (XSS), крос-сайтова підробка запиту (CSRF) та ін. Це дуже важливо, особливо якщо Framework використовують із метою розробки корпоративних продуктів.

Загалом безпека фреймворків залежить від кількох факторів, таких як активність спільноти, кількість, наявність механізмів захисту тощо.

Адаптивність

Використання CMS – це можливість створення швидкого та функціонального сайту. У цьому випадку швидкість розробки вища ніж із фреймворком, але останнє дозволяє глибоко працювати з бізнес-логікою програми. А це означає, що продукт можна адаптувати під конкретні завдання. Ви можете сміливо реалізувати частину функцій індивідуально і в результаті отримати унікальний продукт, якого не буде у ваших конкурентів.

Класифікація фреймворків у веб-розробці

Існує безліч фреймворків для різних мов програмування та завдань. Також їх поділяють на багатофункціональні (для універсальних рішень) та мікрофреймворки (для вирішення конкретних завдань).

Juce - це відкритий кросплатформовий інструментарій розробки програмного забезпечення (фреймворк) для мови C++, що використовується для розробки GUI додатків і плагінів.

Мета Juce - може дозволити компілювати один і той же вихідний текст у програми, що однаково працюють на Windows, macOS і Linux (останні версії - також iPhone та Android) платформах. Він підтримує різні середовища розробки та компілятори, такі як GCC, Xcode та Visual Studio.

Juce вперше опубліковано у 2004, власником його коду є британська компанія Raw Material Software. Має подвійну GPL/комерційну ліцензію.

Пакет Juce призначений для використання одним і тим же способом на безлічі платформ та компіляторів. Компанія Raw Material Software рекомендує наступний список платформ та компіляторів, підтримка яких офіційно підтверджена; інші можуть працювати, але не були офіційно випробувані.

Підтримувані платформи

Juce підтримується на наступних платформах.

Microsoft Windows XP, Vista, 7, 8

macOS, починаючи з версії 10.5

Linux із ядрами версій 2.6

iOS починаючи з версії 3

Android (NDK v.5)

Компілятори, що підтримуються

Офіційно підтверджено правильну роботу Juce, з наступними компіляторами:

GCC починаючи з версії 4.0

Microsoft Visual Studio - Visual C++ версії 2005 і вище

MinGW

Подібно до багатьох інших фреймвок (напр. Qt, wxWidgets, FLTK і т. д.), Juce містить класи, що дозволяють програмі працювати з графікою і звуком, розбирати XML, працювати з мережею і криптографією і т. д.. За рахунок цього потребують додаткових бібліотек програмісти можуть використовувати лише бібліотеку Juce, або хоча б скоротити кількість сторонніх бібліотек, які вони використовують. На це розробників Juce надихнув JDK мови Java. За їхньою заявою, вони збиралися із Juce зробити «щось подібне для C++».

Примітна особливість Juce, порівняно з іншими аналогічними фреймворками, - великий набір аудіофункцій. Справа в тому, що бібліотека Juce спочатку була розроблена як частина аудіосеквенсор Tracktion, і лише потім стала самостійним продуктом. Juce включає підтримку відтворення звуку через аудіо і MIDI інтерфейси, поліфонічні синтезатори, розуміє файли поширених

аудіоформатів (таких як WAV, AIFF, FLAC, і Vorbis). Він також містить інтерфейси-оболонки для побудови різних аудіо плагінів, таких як ефекти та інструменти VST. Це призвело до його широкого поширення у співтоваристві розробників аудіо-ПО.

У постачання Juce входять класи-обгортки для створення як аудіоплагінів, так і браузерних плагінів. При складанні аудіоплагіну виходить єдиний бінарний файл, який підтримує кілька форматів плагінів (VST, RTAS, AU). Оскільки весь платформо-і форматозалежний код міститься в класах-обгортках, то користувач може збирати плагіни у форматі VST/RTAS/AU для макінтошів та Windows з одного і того ж вихідного коду.

p align="justify"> Плагіни для браузерів підтримуються аналогічним чином: один і той же бінарний файл, функціонує і як NPAPI, і як ActiveX плагін.

NPAPI(Програмний інтерфейс модулів, що підключаються Netscape з англ. Netscape Plugin Application Programming Interface) –

крос-платформна архітектура розробки плагінів, що підтримується багатьма браузерами.

Інтерфейс був розроблений для сімейства браузерів Netscape Navigator, починаючи з Netscape Navigator 2.0, і надалі був реалізований багатьма іншими браузерами. Однак, Internet Explorer не підтримує цей інтерфейс, починаючи з версії 5.5

ActiveX є основою для визначення повторно використовуваних компонентів програмного забезпечення незалежно від мови програмування. Програмні застосунки можуть складатися з одного або декількох з цих компонентів з метою забезпечення їх функціональності.

ActiveX був введений в 1996 році Microsoft, як розвиток їхніх технологій Component Object Model (COM) і зв'язування і впровадження об'єктів (Object Linking and Embedding, OLE), і зазвичай використовується в операційній системі Windows. Хоча сама технологія не прив'язана до Windows, на практиці більшість елементів керування ActiveX працюють лише у ній і лише на платформі x86, через наявність у елементах машинного коду процесора.

Багато Windows застосунків — в тому числі від Microsoft, наприклад Internet Explorer, Microsoft Office, Microsoft Visual Studio і Windows Media Player — використовують елементи управління ActiveX, щоб побудувати свій набір функціоналу, а також інкапсулюції своїх функцій як елементів керування ActiveX, які можуть потім вкладатися в інші застосунки. Internet Explorer також дозволяє вбудовувати ці елементи управління на вебсторінках.

Керуючі елементи ActiveX — це як будівельні блоки для програм, вони можуть використовуватися для створення розподіленого додатка (клієнт-серверний додаток, що користовується технологією розподілених обчислень), що працює в браузері. Прикладами є налаштовуємі додатки по збору данних, перегляду файлів різного типу і відображення анімацій.

Програмісти можуть створювати керуючі елементи ActiveX за допомогою будь-якої мови програмування, що підтримує розробку компонентів Component Object Model (COM), зокрема:

- C ++;
- Delphi 7;
- Visual Basic;
- .NET Framework;

Jucer

Невід'ємна частина фреймворку Juce — програма Jucer (так само написана на Juce), яка використовується для візуального проектування та редагування графічних інтерфейсів. Jucer може згенерувати C++ код, що реалізує обрану структуру графічного інтерфейсу.

Juced. Є також неофіційне відгалуження бібліотеки, розширене додатковими можливостями, яке підтримує співтовариство, воно називається Juced. На сайті цього варіанта фреймворку можна знайти також додаткову документацію Juce, що допоможе освоїти бібліотеку.

Як відомо, звуковий сигнал у комп'ютері може представлятися у вигляді деякого набору підрахунків його амплітуд, що проводяться через певні проміжки часу (період дискретизації) і деякою кількістю двійкових розрядів (розрядність

вибірки). Таке уявлення зручне для зберігання звукового сигналу та його перетворення назад у безперервний сигнал. Однак деякі операції з обробки звукового сигналу в такому поданні буває не завжди зручні. Це пов'язано з тим, що реально звуковий сигнал складається з його частот з певною амплітудою і фазою. Таким чином, застосування таких операцій з обробки звукового сигналу як "фільтр нижніх частот" або "фільтр верхніх частот" вимагає перетворення звукового сигналу у вигляді підрахунків у його частотний спектр. Після цього перетворений звуковий сигнал буде представлений у вигляді коефіцієнтів відповідних амплітуд і фаз частот, що становлять цей сигнал. Тепер наприклад операція «фільтр нижніх частот», яка «вирізає» з сигналу всі частоти вище заданої, може просто обнулити коефіцієнти відповідні частотам, які необхідно «вирізати». Насправді коло застосування такого перетворення значно ширше: обробка растрових зображень, телекомунікації, дослідження та вимірювання сигналів, радіолокація тощо. Прикладом застосування перетворення може бути передача даних у цифровій формі аналоговими лініями телефонної мережі (модем). Для передачі даних у цифровій формі, вони спочатку перетворюються на деякий набір частот і передаються по лініях передач, а потім на приймальній стороні виконується зворотне перетворення і відновлюються вихідні дані. Далі буде розглянуто деякі фундаментальні поняття з математичного аналізу, необхідні розуміння роботи.

3.3 Перетворення Фур'є.

Ряд Фур'є Поруч фур'є називається нескінченна математична послідовність, що складається з коефіцієнтів при функціях синуса та косинуса виду:

$$\frac{a_0}{2} + \sum_{m=1}^{\infty}\left(a_m \cos\frac{m\pi x}{l} + b_m \sin\frac{m\pi x}{l} \right)$$

Доведено, що й деяка періодична функція з періодом 2l на інтервалі [-l, l] задовольняє умовам Діріхле (безперервна і має кінцеве число екстремумів і точок розриву I роду), вона може бути представлена вигляді суми низки Фур'є

(розкладений ряд Фур’е). Для визначення коефіцієнтів низки Фур'є справедливі такі формули:

$$a_m = \frac{1}{l}\int_{-l}^{l} f(x)\cos\frac{m\pi x}{l}\,dx$$

$$b_m = \frac{1}{l}\int_{-l}^{l} f(x)\sin\frac{m\pi x}{l}\,dx$$

Якщо функція, що розкладається, є парною (f(-x) = f(x)), то ряд Фур'є складається тільки з косинусів, тобто всі коефіцієнти при синусах рівні 0. Якщо функція, що розкладається, є непарною (f(-x) = - f(x)), то ряд Фур'є складається тільки з синусів, тобто всі коефіцієнти при косинусах дорівнюють 0. У загальному випадку, коефіцієнти при синусах і косинусах не дорівнюють 0.
Таким чином, будь-яку періодичну функцію, що задовольняє умовам Діріхле, можна розкласти в ряд Фур'є, тим самим представляючи її у вигляді суми синусів та косинусів.

Перетворення Фур'є у загальному вигляді

Припустимо тепер, що досліджувана функція перестав бути періодичної, т. е. її період повторення дорівнює нескінченності. У цьому випадку справедлива інтегральна формула Фур'є, одержувана шляхом граничного переходу з ряду Фур'є періодичної функції з періодом 2l при l ?

$$f(x) = \frac{1}{\pi}\int_{0}^{+\infty} dz \int_{-\infty}^{+\infty} f(u)\cos z(u-x)\,du$$

З цією формулою пов'язані так звані інтегральні перетворення Фур'є:

$$F(z) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} e^{izx} f(x)\,dx$$

$$f(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} e^{-izx} F(z)\,dz$$

Метод кореляцій

Метод кореляцій дозволяє визначити тісноту лінійної залежності між досліджуваною та базисною функціями. Це легше зрозуміти з прикладу. Нехай є імпульсна станція радіолокації. Для виявлення мети ця станція формує короткий високочастотний радіоімпульс, що огинає має прямокутну форму. Цей імпульс випромінюється у простір і відбивається від мети. Так як в однорідному середовищі електромагнітні хвилі поширюються з постійною швидкістю, близькою до швидкості світла у вакуумі, то знаючи час, через який відбитий від мети імпульс надійшов у приймач станції радіолокації, можна визначити відстань до мети. Однак тут виникає така проблема: так як тільки частина імпульсу відбивається від мети і надходить назад до приймача, а також до приймача надходять деякі перешкоди і сам приймач має деякий коефіцієнт шуму, результуючий імпульс буде дещо розмитий на тлі перешкод і шумів. Виникає питання, як визначити якась частина графіка, представленого на рис. буде представляти відбитий імпульс? Скористаємося наступним методом: візьмемо за основу функцію, яка на деякому інтервалі матиме стрибок, що є ідеальним відбитим імпульсом (при відсутності перешкод і послаблень). Далі будемо в кожній точці перемножувати цю базову функцію на функцію, що формується приймачем, а потім проінтегруємо отриману функцію. Таким чином, це перетворення дозволяє визначити наскільки велике значення базисної функції досліджуваної. Якщо досліджувана функція у кожній своїй точці дорівнює базисної, то інтеграл функції, отриманої внаслідок цього перетворення матиме максимальне значення. Якщо досліджувана функція ні як не відбиває базисну, то результат дорівнюватиме 0. У проміжних варіантах значення інтеграла результуючої функції відображатимуть те, наскільки точно досліджувана функція відповідає базисній. Це і є метод кореляцій. Повернемося до визначення відстані до мети. Тепер формуватимемо нову базисну функцію, інтервал стрибка якої зміщуватимемо вправо по осі абсцис.

Як тільки цей інтервал буде відповідати інтервалу функції, що відповідає відображеному імпульсу, значення інтеграла результуючої функції буде

максимальним. Таким чином, може вирішуватися завдання про визначення відстані до мети.

Дискретне перетворення Фур'є

У дискретному перетворенні Фур'є функція, що досліджується є періодичною має кінцевий період повторення, і є дискретною. Фактично дискретне перетворення Фур'є дозволяє уявити дискретну функцію як кінцевого числа частот з певними значеннями амплітуди і фази (розкладає функцію її спектр). Це ґрунтується на тому, що за наслідком з теореми Котельникова в дискретному сигналі період, що відповідає найвищій частоті, що відповідає частоті відповідає двом періодам дискретизації.

Для визначення амплітуд та фаз частотних складових сигналу, у дискретному перетворенні Фур'є використовується кореляція з базисними функціями синуса та косинуса. Спектр частот у дискретному перетворенні Фур'є визначається з амплітуд синусів і косінусів, з частотами повторення в досліджуваній вибірці від 0 до N/2 разів, де N - кількість елементів вибірки. Перетворення Фур'є розкладає дискретизований сигнал N вибірок на N/2 + 1 синусних і N/2 + 1 косинусних складових. Чому саме синусних та косинусних?
Тому можна сказати, що дискретне перетворення Фур'є розкладає досліджуваний сигнал за базовими функціями синуса та косинуса. Вони є аналогами двох взаємно перпендикулярних коливань, оскільки фазою зміщені один щодо одного на 90 градусів. Всі вищенаведені міркування призводять до таких формул дискретного перетворення Фур'є:

$$ReX[k] = \sum_{i=0}^{N-1} x[i] \cos(2\pi k i / N)$$

$$ImX[k] = -\sum_{i=0}^{N-1} x[i] \sin(2\pi k i / N)$$

Ці формули описують пряме перетворення Фур'є. ReX[x] - масив, що містить значення косинусоїдальних складових. ImX[x] - масив, що містить

значення синусоїдальних складових. Такі позначення введені з комплексного уявлення безперервного перетворення Фур'є. У цьому дійсній частини відповідають косинуси, а уявній - синуси. Також досліджувана функція є функцією дійсного змінного.

Комплексна форма перетворення Фур'є може вводитись для зручності запису двох інтегралів – для косинуса та синуса. Масиви Re[x] і Im[x] становлять так званий частотний домен (frequency domain), тоді як вихідна вибірка називається тимчасовим доменом (time domain).

Властивості дискретного перетворення Фур'є

За наведеними вище формулами проводиться розкладання досліджуваного сигналу його спектр. З'ясуємо тепер властивості перетворення Фур'є. Припустимо, що потрібно зробити зворотне перетворення - із частотних складових сформувати вихідний сигнал. Для цього справедливі наведені нижче формули:

$$x[i] = \sum_{k=0}^{N/2} Re\bar{X}[k]\cos(2\pi ki/N) + \sum_{k=0}^{N/2} Im\bar{X}[k]\sin(2\pi ki/N)$$

Коефіцієнти ReX[k] та ImX[k] визначаються за такими формулами:

$$Re\bar{X}[k] = \frac{ReX[k]}{N/2}$$

$$Im\bar{X}[k] = -\frac{ImX[k]}{N/2}$$

За винятком двох випадків:

$$Re\bar{X}[0] = \frac{ReX[0]}{N}$$

$$Re\bar{X}[N/2] = \frac{ReX[N/2]}{N}$$

Такий процес перетворення називається синтезом чи зворотним перетворенням Фур'є. Зауважимо, що формули зворотного перетворення аналогічні формулам прямого перетворення, тільки тепер підінтегральної функції є коефіцієнтами при синусах і косинусах. Ця властивість є дуже важливою і називається двоїстістю перетворення Фур'є. Властивість двоїстості дозволяє пояснити наступний факт: одиничний імпульс у тимчасовому домені (одиничне значення однієї вибірки при нульових значеннях інших) відповідає синусоїді та косинусоїді в частотному домені і навпаки.

У другому випадку все зрозуміло - є один коефіцієнт при синусі або косинусі - це означає, що вихідний сигнал (вибірка) містить складову однієї частоти синусоїдальної або косинусоїдальної форми. Перший випадок можна пояснити з урахуванням двоїстості перетворення Фур'є. Описаний факт використовується при побудові алгоритму швидкого перетворення Фур'є. Справа в тому, що наведені вище формули для прямого і зворотного перетворень мають тимчасову складності алгоритмів порядку O(n^2), що їх реалізують.

Таким чином, при великих обсягах вибірки, не вдається за реальний час зробити перетворення Фур'є. Для цієї мети в середині 60-х років було розроблено алгоритм швидкого перетворення Фур'є.

РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ

3.1. Опис апаратних та програмних вимог до реалізації проектованої системи

Система передбачає наявність наступних складових:

1 Компресія звуку

2 Нойз гейт

3 Простий та зручний інтерфейс для роботи з ефектами

4 Фунція збереження відредагованого файлу

6 Можливість інтегрування в Медіаплеєр

3.2 Компрессор

Першим модулем вирішено створити компресор або стиснення. Це характеризується процесом зменшення динамічного діапазону сигналу. Тобто його завданням є усунення чутної різниці між гучними та тихими частинами звуку.

Після обробки зникають перепади рівня: різкі ривки по гучності, що дають пік сигналу та несподівані падіння рівня. Більше того, у компресора є можливість збільшити вихід після компресії, тобто гучність, щоб не втратити початковий рівень гучності.

Зазвичай даний метод працює із загальним рівнем амплітуди всього сигналу. В цій роботі було вирішено реалізувати багатосмуговий компресор, який пропонує три діапазони для стиснення сигналу. Тобто для даного модуля також необхідна реалізація алгоритму БПФ для обчислення спектральних значень, оскільки багатосмуговий компресор працює із сигналом в амплітудно-частотній області.

За коректну роботу компресора відповідають такі характеристики:

- Поріг спрацьовування;
- Співвідношення;
- Атака;
- Час дії;
- Спад.

Поріг спрацьовування (Threshold) визначає, у який саме момент включається алгоритм роботи компресора. Цей параметр встановлюється в децибелах, і якщо сигнал переходить це значення, він стискається

Рис 17 візуалізація порогу спрацювання

Значення порога встановлює максимально можливу гучність сигналу до свого стиснення. При значенні порога ближче до нуля стиснення буде застосовано до меншої частини сигналу, лише до «агресивних» піків сигналу.

За значеннями ближче до -40 dB і далі ослаблення працюватиме на більшу частину сигналу.

Співвідношення (Ratio) визначає розмір послаблення сигналу, перейшовши за поріг спрацьовування. Наприклад, 2:1 або 4:1 і надалі. Коефіцієнт співвідношення вказує на відношення стисненого сигналу до вихідного, тобто скільки разів буде ослаблений вихідний сигнал. Чим більше перше число, тим більше робота компресора.

Атака і спад та час дії відповідають за часові рамки роботи компресора. Атака визначає швидкість спрацьовування стиснення сигналу, що вийшов за поріг. Спад визначає швидкість закінчення роботи, повертаючи сигнал у вихідний рівень та очікуючи наступного заходу за поріг.

Час очікування відображає час, протягом якого відбувається робота компресора

Рис 18 робота компресора

DAW поширені на різних операційних системах, як на windows, так і на Linux, і MacOS. Передбачається, що розроблений програмний модуль має бути кросплатформним.

Для реалізації модулів було вирішено використовувати мову C++, оскільки вона дозволяє створювати кросплатформне програмне забезпечення.

Для реалізації самого алгоритму плагіна існує бібліотека VST.NET від компанії Steinberg, але для роботи було вирішено використати інструмент JUCE framework.

JUCE framework - це кросплатформний фреймворк для розробки програмного забезпечення, створеного Джулсом Стоуеллом (Jules Storer). Він надає розробникам інструменти для створення програм для комп'ютерів, веб-браузерів та мобільних пристроїв. JUCE має компонентів, таких як графічний інтерфейс користувача, аудіо-движок, мережеві можливості, підтримку багатопоточності та багато іншого. Він написаний мовою C++, та його код є відкритим та доступним для всіх. Фреймворк має в своєму розпорядженні велику кількість бібліотек для розробки інтерфейсу додатків та також має великий обсяг бібліотек для використання математичних методів обробки звукового сигналу.

JUCE підтримує такі платформи:

- Mac OS X плагіни компілюються за допомогою Xcode;
- Windows плагіни компілюються за допомогою MS Visual Studio;
- Linux плагіни можуть бути зібрані для ядра версії 2.6 та вище;

- iOS плагіни збираються за допомогою Xcode;
- Android плагіни збираються за допомогою Ant або Eclipse з використанням Android NDK.

JUCE має в своєму розпорядженні велику кількість класів для роботи зі звуком, наприклад, клас AudioProcessor, який відповідає за запис вхідних значень звукового сигналу, а також за вихідні значення.

JUCE framework має в своєму розпорядженні свій окремий додаток з можливістю вибору налаштувань майбутнього проекту. Цей додаток називається Projucer.

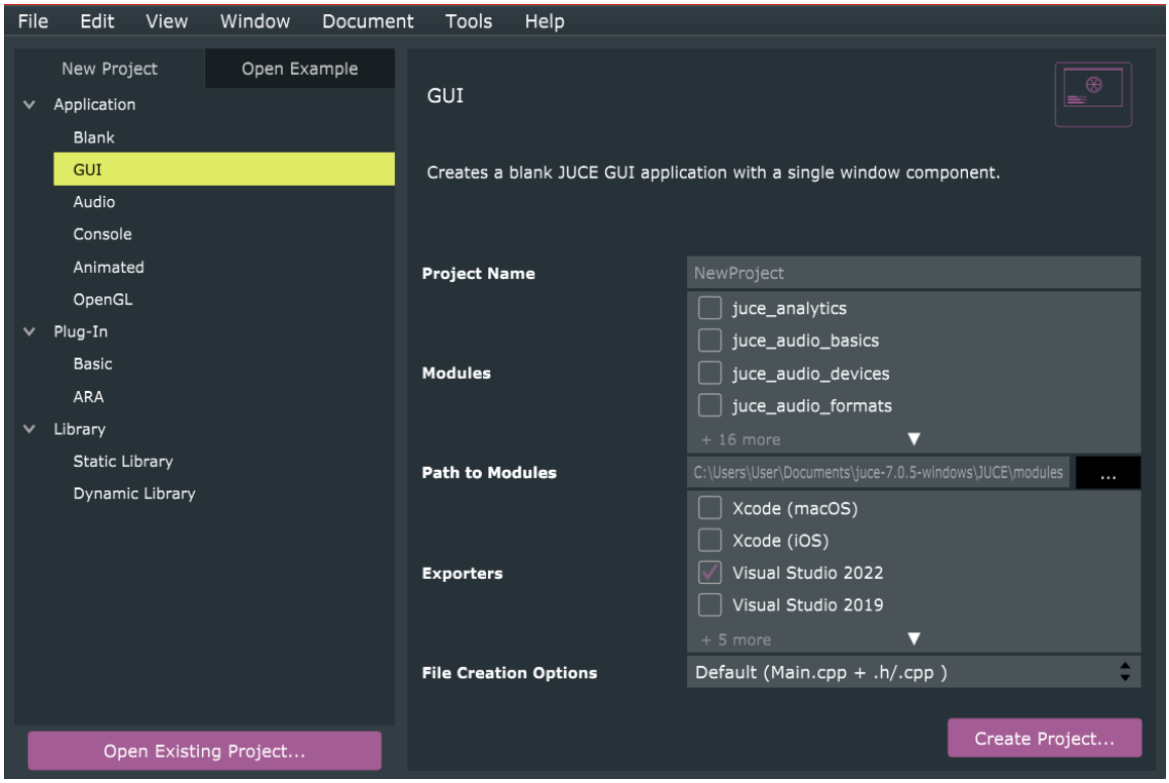

Рис 19 Вигляд стартового меню фреймворку

У цьому вікні можна вибрати шаблон для проекту, що розробляється.

З шаблонів представлені:

- Blank – порожній проект;
- GUI – створює порожній проект із можливістю додати додаткові функціональні можливості, такі як додаткові компоненти графічного інтерфейсу, використовуючи різні класи, які пропонує JUCE;
- Audio – створює проект на основі GUI, але додає шаблони класів

для роботи з сигналом аудіо-формату для реалізації простих операцій введення та виведення сигналу, зміни тривалості тощо;

- Console – створює проект під реалізацію роботи з командним рядком;
- Animated – створює проект, який надає шаблони під реалізацію анімацій у додатку;
- OpenGL – створює порожній проект, подібний до GUI, з додаванням підтримки OpenGL для роботи з двовимірною та тривимірною графікою
- Basic – створює порожній програмний модуль без реалізації методів обробки з підтримкою формату файлу VST
- ARA – створює проект програмного модуля з можливістю додавання бібліотек, що працюють поверх наявних у JUCE;

У даній програмній реалізації для цієї роботи використовується шаблон Basic, при виборі цього шаблону створюється репозиторій з такими файлами:

- PluginProcessor.cpp;
- PluginProcessor.h;
- PluginEditor.cpp;
- PluginEditor.h.

Також Projucer пропонує вибір середовища розробки для програмування програмного модуля. У разі роботи в операційній Windows, проект збирається під MS Visual Studio.

Файли заголовків формату «.h» необхідні для оголошення класів і функцій, файли «.cpp» служать для реалізації роботи з оголошеними функціями та класами.

Основна робота йде, якраз із файлами в папці Source, також через Projucer можливо додавати заголовні та виконувані файли для більш ефективної структуризації модуля, що розробляється.

Розберемо спочатку зібрані файли PluginEditor.cpp та PluginProcessor.cpp

Файл PluginEditor.cpp надає початковий набір функцій для реалізації інтерфейсу майбутнього програмного модуля Тут надається можливість для

встановлення розміру майбутнього вікна інтерфейсу, функцію для відтворення інтерфейсу та функцію для зміни розмірів відмальованих компнентів.

Цей файл реалізує функції формування входу (Input) та виходу (Output), які реалізують передачу аудіо-даних із звукового редактора.

Також реалізована функція prepareToPlay для підготовки даних до роботи програми. Функція processBlock необхідна для основної роботи з файлами.

Для реалізації програмних модулів було вирішено розділити реалізацію на три різні проекти під кожен плагін.

Вимоги до програмного модуля

Для програми виявлено такі функціональні вимоги, яким вона має задовольняти:

- надавати можливість регулювати рівень гучності сигналу;
- надавати можливість регулювати положення сигналу в стерео;
- Дозволяти прибирати зайві шуми з сигналу;
- Візуалізувати амплітудно-частотну характеристику (АЧХ) сигналу – спектр;
- Використовувати фільтри для редагування АЧХ;
- використовувати методи стиснення сигналу для вирівнювання амплітуди;
- Надавати можливість запуску програми з будь-якого аудіо редактора.

Реалізація компресії

У реалізації цього модуля було вирішено організувати більш складну структуру файлів, оскільки потрібна реалізація великої кількості компонентів взаємодії з даними сигналу.

Крім стандартних файлів шаблону Basic є такі компоненти:

- GlobalControls – відповідає за реалізацію функцій зміни вхідної та вихідний гучності, діапазон частот захоплюваних смугами компресора;
- CompressorBandControls – включає реалізацію класу Compressor та логіку роботи методів стиснення;
- CustomButtons – задає загальний стиль для кнопок у плагіні;

- RotarySlider – задає стиль та графічне виконання кругових слайдерів;
- SpectrumAnalyser – відповідає за візуалізацію спектра сигналу та смуг стискування;
- FFTDataGenerator – реалізація алгоритму БПФ для отримання значень спектра сигналуСтандартні файли відповідають за використання функцій, оголошениху додаткових компонент.

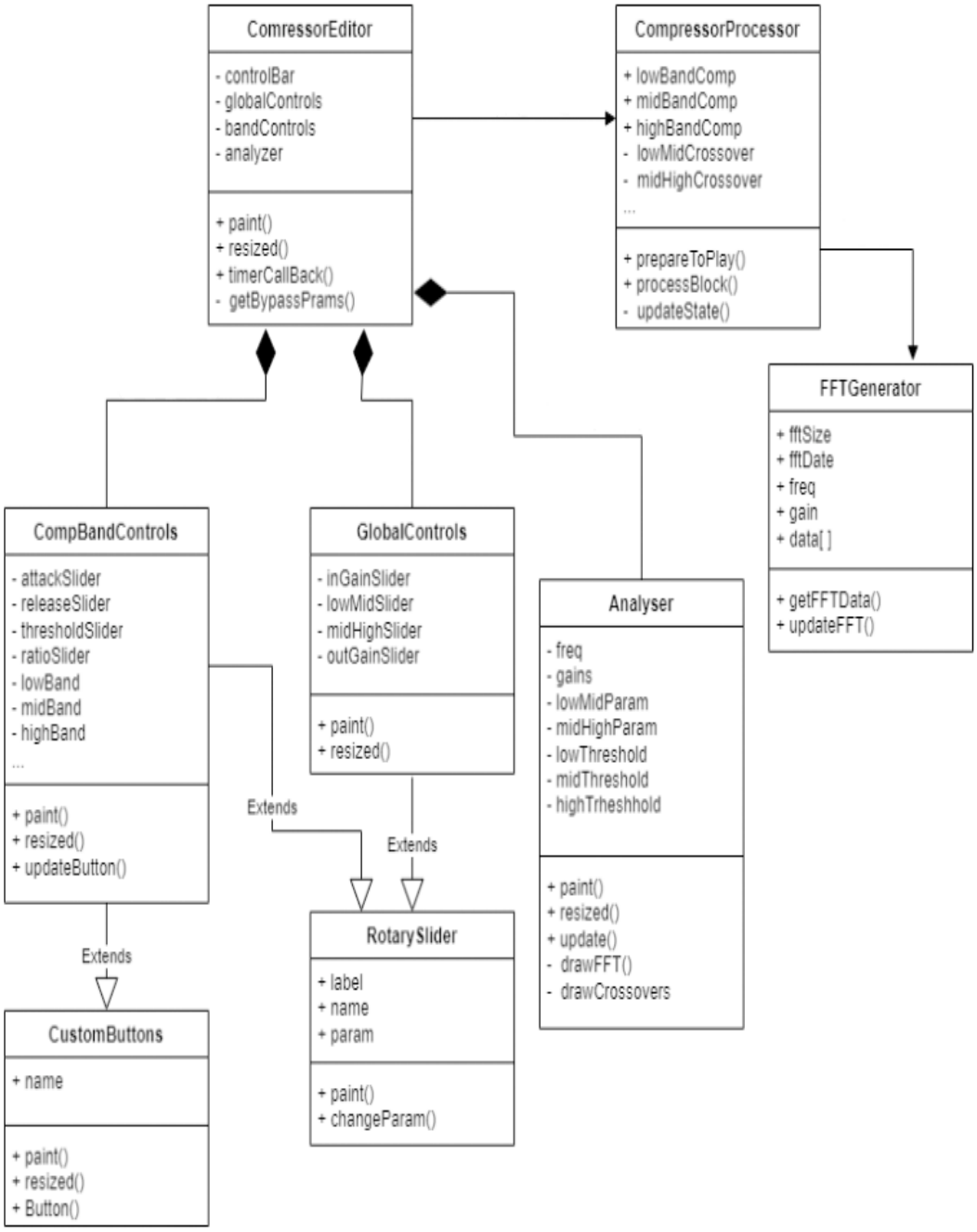

Рис 20 діаграма класів реалізації компресора

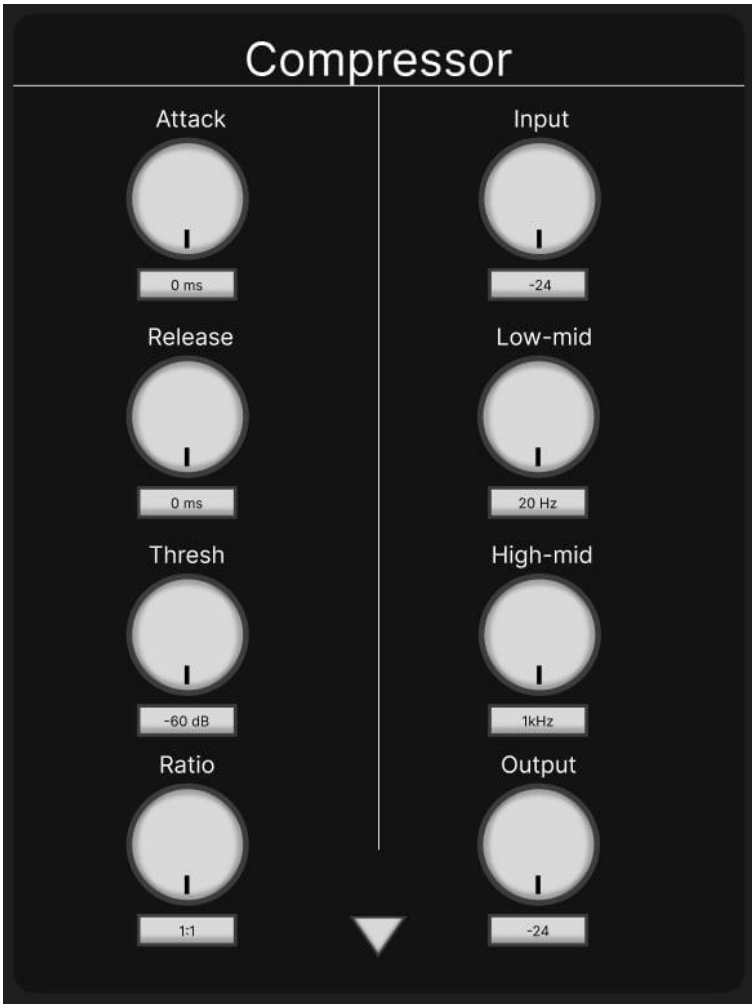

Рис 21 – Вигляд Компрессору

Компресор буде представлений такими компонентами:

- Вибір діапазону частот;
- Рівень вхідного сигналу;
- Рівень вихідного сигналу;
- Регулятор діапазонів частот;
- Атака (за який час після проходження порога спрацьовує стиск);
- Час дії (скільки часу працює стиск сигналу);
- Поріг (при переході якого значення в децибелах спрацьовує
- компресор);
- Ratio (співвідношення).

3.3 Noise Gate та інтерфейс

При розробці плагіну шумоподавлення було прийнято рішення створити Noise Gate. Таке рішення було прийняте через те що видалення шуму з вже записаної аудіо доріжки являється складним поцесом під час якого можна пошкодити або ж взагалі видалити корисний звук Шумовий гейт або просто гейт — це програмне забезпечення, яке використовується д ля керування гучністю аудіосигналу. Порівняно з компресором, який послаблює сигнали вище порогу, такі як гучні атаки з початку музичних нот, шумові гейти послаблюють сигнали, які реєструються нижче порогу. Проте шумові гейти послаблюють сигнали на фіксовану величину, відому як діапазон. У своїй найпростішій формі шумовий гейт пропускає основний сигнал лише тоді, коли він перевищує встановлений поріг: гейт «відкритий». Якщо сигнал падає нижче порогового значення, сигнал не може пройти (або сигнал суттєво ослаблений): ворота «закрито». Шумовий гейт використовується, коли рівень «сигналу» перевищує рівень небажаного «шуму». Поріг встановлюється вище рівня «шуму», тому, коли немає основного «сигналу», гейт закривається. А саме тому данна функція відповідає моїй задумці створення простого розширення фунціоналу медіплеєра. До йього ж він не потребує багато знань і його можна легко модифікувати для використання будь яким користувачем.

Розробляючи Noise Gate я реалізував такі компоненти:

Атака (Атака від 0.1 до 1000 мсек)

Встановлює час, по істеченні якого гейт відкривається при переключенні.

Утримання (Утримання від 0 до 2000 мсек)

Встановлює час, протягом якого гейт залишається відкритим, після того, як сигнал пускається нижче порога.

Release (Відпуск від 10 до 1000 мсек або режим Авто)

Встановлює час, по істеченні якого гейт закривається (після встановленого часу Утримання).

Поріг (Порог від -60 до 0 дБ) Встановлює рівень, при якому гейт спрацьовує. Сигнали з рівня вище порогового переключають гейт у відкритий стан, а сигнали з рівня нижче порогу закривають гейт.

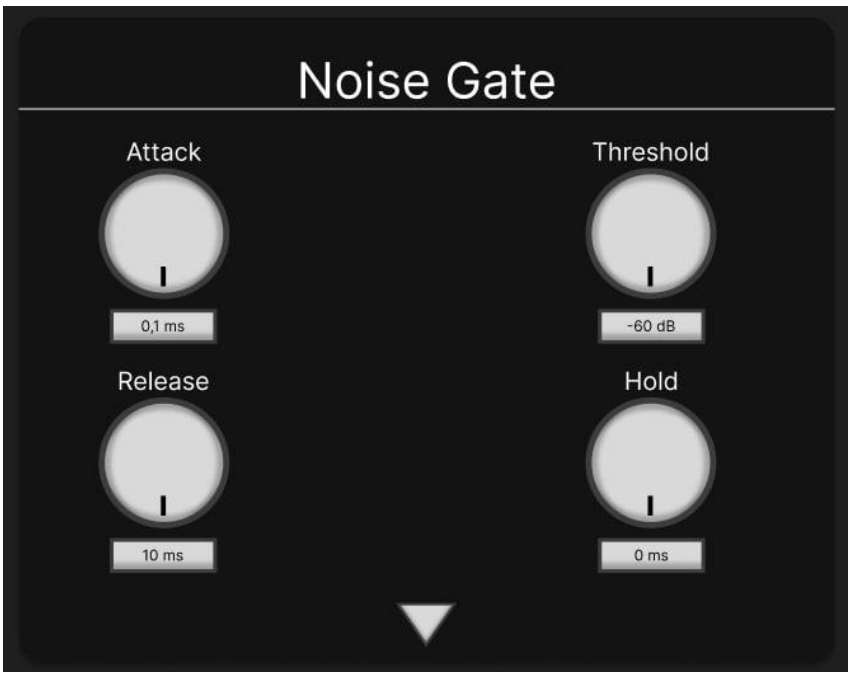

Рис 22 Вигляд Noise Gate

Реалізація зручного інтерфейсу

Оскільки якість процесу інтерактивної взаємодії користувача із системою (швидкість, зручність, низький рівень втоми) пов’язана з такими психологічними характеристиками людини як короткострокова та середньострокова пам’ять, час реакції, можливості сприйняття візуальної інформації, то при розробці інтерфейсу необхідно пам’ятати, що

Основними властивостями, яким повинні задовольняти інтерфейси, є такі:

- Ясність. Інтерфейс дозволяє уникнути двозначності, та робить все зрозумілішим через мову, потік, ієрархію та метафори для зорових елементів.
- Виразність. Легко зробити інтерфейс зрозумілим, завдяки надмірному уточненню та позначенню всього, але це призводить до його роздування, коли на екрані занадто багато елементів. Якщо на екрані занадто багато елементів, то знайти те, що ви шукаєте важко і через це, інтерфейс стає складним та нудним у використанні. Справжнє завдання у створенні

досконалого інтерфейсу полягає в тому, щоб одночасно зробити його і стислим і зрозумілим.

- Знайомство. Навіть якщо хтось користовує інтерфейс вперше, певні елементи можуть бути знайомі. Приклади з реального життя, може бути втілено для передавання інформації.
- Відповідність. Відмінний інтерфейс не повинен бути млявим. Це означає, що він мусить забезпечити користувача хорошим зворотним зв'язком відносно того, що саме відбувається, та чи успішно обробляються введені данні користувача.
- Послідовність. Важливо щоб інтерфейс було узгоджено з програмою, оскільки він дозволяє користувачеві розпізнавати шаблони використання.
- Адаптованість означає, що інтерфейс повинен бути:
  - сумісним з потребами та можливостями користувача; – забезпечувати простоту переходу від виконання однієї функції до іншої;
  - забезпечувати користувача на високому рівні вказівками стосовно його можливих дій, а також генерувати належний зворотний зв’язок на його запити;
  - надавати користувачу можливість відчувати себе повноправним керівником ситуації при розв’язанні всіх типів задач, тобто, забезпечувати його всією необхідною інформацією
- Естетика. Хоча це не так важливо, але створювати привабливий інтерфейс щоб він виконував своє завдання та робив щось добре, сприяє на час який витрачають користувачі за допомогою вашої програми, робить його приємнішим;
- «Прощення». Гарний інтерфейс не повинен карати користувачів за їх помилки, але мусить надавати засоби задля їх усунення.

Так як моя робота орієнтована на користувачів що майже не мають знань в плані обробки звуку мною було розроблено спрощений до мінімуму інтерфейс.

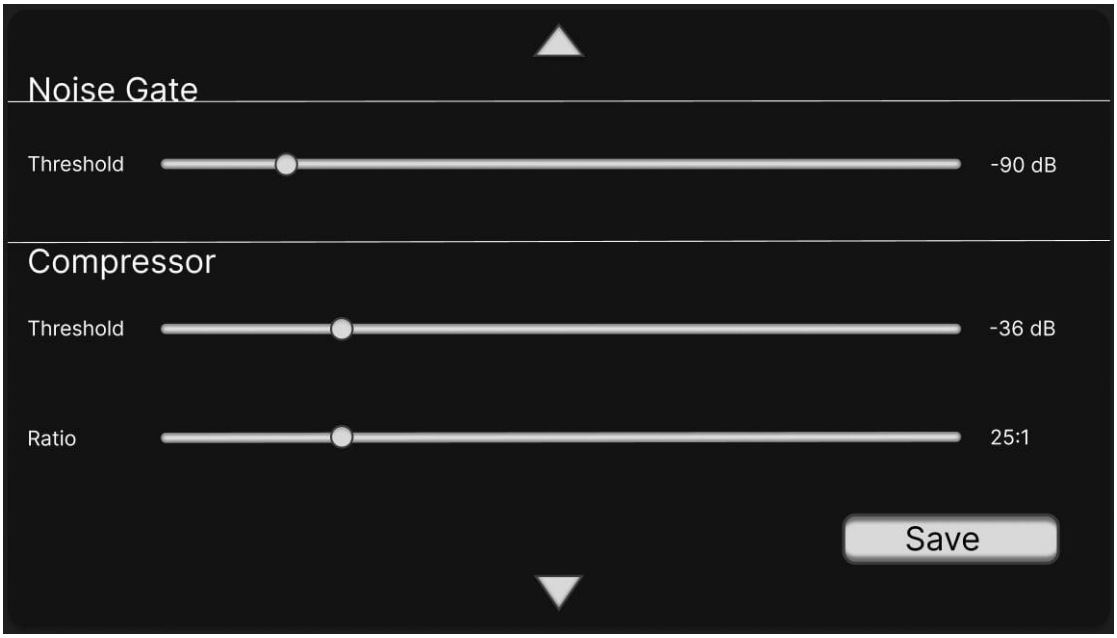

Рис.23 Вигляд мого спрощеного інтерфейсу

Рис.24 Вигляд нижньої панелі медапрогравача з доданим мною елементом меню (іконка ноти з плюсом)

Як видно з зображення в меню тільки 3 повзунки налаштувань. Один для Нойз гейту та два для компрессору, всі інші налаштування приховано в розширених меню та їх параметри виставленні автоматично, їх можна відкрити за допомогою стрілок вгору і вниз. При переході в розширене меню одного з ефектів авто-налаштування вимикається.

Також було розроблено функцію що відповвідає за збереження вже відредагованого файлу (кнопка Save).

3.4 Тестування з різними видами шуму.

При проведенні тестів задля отримання більш повної картини я розбив шуми на чотири види а саме: стаціонарні(білий шум), коливаючісь(шум вітру), перивчаті(гудки, стук ітд) та імпульсні(скрип, чихання), і якщо з першими двома випадками все відносно легко. Єдиний випадок в якому їх усунення ускладнюється або ж неможливе це перевищення шумом порогу звучання корисного звуку або ж дуже близьке розташування до нього(може призвести до

втрати частини потрібної інформації). То з наступними двома все набагато складніше через те що зазвичай такі шуми дуже різкі та високі що унеможливлює їх повне видалення моїми інструментами.

Трохи детальніше по кожному з них.

При стаціонарному шумі було достатньо виставити нойз гейт в необхідне положення задля відключення шуму. Проте при наявності тихих участків корисного звуку відбувалася або ж часткова втрата необхідного аудіо або при коректуванні налаштувань щоб уникнути першого шум просто ставав тихішим.

При коливальних шумах ситуація була схожою як і в випадку з стаціонарними шумами проте з певними нюансами. Так як коливальні шуми необов’язково бувають однієї гучності то їх редагування протягом всього запису може мати неоднозначний результат.

При преривчастих шумах якість обробки стала гіршою. Адже такі шуми мають дуже різкі перепади через що нозг гейт не може повністю їх прибрати. Натомість при застосуванні компрессору їх гучність часково можна звести до гучності голосу, музики ітд що покращує сприйняття доріжки.

В ситуації ж з імпульсними шумами результати погіршилися ще більше. При їх обробці я застосував той же метод що і при преривчастих шумах, але в результаті при пониженню їх рівня до корисного звуку іноді виникали звукові артефакти.

Висновки

Неможливо прибрати шум повністю

Повністю видалити шум з аудіозапису може бути дуже складним завданням зазвичай з численними технічними і практичними обмеженнями. Ось кілька ключових причин, чому це завдання може бути складним або навіть неможливим:

Співвідношення сигнал-шум: У багатьох аудіозаписах існує велика різниця між корисним сигналом (тобто тим, що вам потрібно зберегти) і шумом. Якщо ця різниця мала або сигнал і шум майже однакові, то важко або навіть неможливо визначити, де саме знаходиться корисна інформація.

Несприятливі умови запису: Якість запису може сильно вплинути на можливість видалення шуму. Якщо аудіозапис був створений у дуже шумному середовищі або з використанням поганого обладнання, шум може бути вкрай важко видалити без втрати корисної інформації.

Артефакти від процесу видалення шуму: Процеси видалення шуму, такі як фільтрація чи скасування шуму, можуть створювати артефакти або змінювати якість аудіо. Це може призвести до втрати деякої корисної інформації або створити неприйнятні аудіоартефакти.

Різні види шуму: Шум може бути різних видів, таких як шум від фону, електричний шум, резонансні шуми тощо. Видалення різних видів шуму може вимагати різних методів та інструментів, а інколи ці методи можуть бути непридатні для певних видів шуму.

Втрата даних: Під час обробки аудіо для видалення шуму може виникнути втрата деякої корисної інформації, особливо якщо шум і сигнал переплітаються або якщо виникають артефакти. Це може зробити аудіозапис менш прийнятним для відтворення.

Послідовність дій. Мій застосунок потребує певної послідовності при роботі з ним, а саме спочатку використання ефекту Noise gate а вже потім компресору. Адже при сильній компресії або низькому звучанні корисного звуку

після її застосування видалення фонового шуму може призвести до втрати або створення дефектів першого.

Можливі дефекти звуку при користуванні спрощенним інтерфейсом. Так як в спрощеному меню більшість налаштувань стоять в середніх оптимальних значеннях, пвидалення шуму такі як фільтрація чи скасування шуму, можуть створювати артефакти, змінювати якість аудіо або це може призвести навіть до втрати деякої корисної інформації.

Можливість покращення. В майбутньому буде розглянута можливість розширення можливостей мого додатку а саме додавання шумовирізання за допомогою штучного інтелекту.

Загалом мій додаток можна вважати корисним для простих задач по обробці аудіо, коли немає потреби в використанні професнійних аудіо редакторів та найманні людей які знаються на роботі з аудіофайлами. Також великим плюсом є те що він знаходиться безпоспосередньо в мудіаплеєрі тому не потрібно шукати аналоги зі схожим функціоналом в інтернеті.

Список використаних джерел

1. Шум. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Шум (дата звернення: 30.04.2022) 5. ДСТУ 2325-93. Шум. Терміни та визначення.
2. Білий шум. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Білий_шум (дата звернення: 20.10.2023)
3. What is Gray Noise? URL: https://www.techopedia.com/definition/ 27898/graynoise (дата звернення: 23.10.2023)
4. Dynamic Time Warping. URL: https://handwiki.org/wiki/ Dynamic_time_warping (дата звернення: 25.04.2022)
5. Suryo Wijoyo, "Speech Recognition Using Linear Predictive Coding and Artificial Neural Network for Controlling Movement of Mobile Robot", International Conference on Information and Electronics Engineering IPCSIT vol.6, IACSIT Press, pp.179-183, Singapore, 2011.
6. Neural network. Wikipedia. URL: https://en.wikipedia.org/w /index.php?title=Neural_network (дата звернення: 05.10.2023)
7. Viterbi algorithm. Wikipedia. URL: https://en.wikipedia.org/wiki/Viterbi _algorithm (дата звернення: 15.10.2023)
8. Черняхов А.В., Каук В.І. Використання дискретного перетворення Фур'є на прикладі мобільного застосунку для запису та аналізу звуку//Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. 2022. №2. С. 5
9. Verbytskyi, I. V. (2018). Швидке перетворення Фур'є модульованих сигналів, представлених рядом Фур'є двох змінних. Вісник Національного технічного університету «ХПІ». Серія: Нові рішення у сучасних технологіях.
10. Iain E. G. Richardson. Video Coding for Next-generation Multimedia. URL: 76 https://www.researchgate.net/publication/44512776_H264_and_MPEG4_Video _Compression_video_coding_for_nextgeneration_multimedia_Iain_E_G_Richa rdson (дата звернення 26.04.2023).

11. СкіббЛ.Дж.,Хейфмейстер С., Ческат А.М. Оптимізація мультимедиа ПК, К.: НІПФ Діа Софтлтд, 1997
12. Adaptive Multimedia Streaming. URL: https://www.intechopen. com/books/recenttrends-in-communication-networks/a-surveyon-adaptive-multimedia-streaming (дата звернення 26.04.2023)
13. David Austerberry. The technology of video and audio streaming. Tatlor&Francis Group. 2005. 356 c.
14. Wes Simpson. Video Over IP: IPTV, Internet and Video, H.264, P2P, Web TV, and Streaming: A Complete Guide to Understanding the Technology. Routledge. 2008. 518 c.
15. Ієрархія сервера. URL: https://www.akamai.com/glossary/what-is-a-cdn (дата звернення 26.04.2023).
16. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання : (ГОСТ 7.1-2003, IDT) : ДСТУ ГОСТ 7.1:2006. – Чинний з 2007–07–01. – К. : Держспоживстандарт України, 2007. – 47 с. – (Система стандартів з інформації, бібліотечної та видавничої справи) (Національний стандарт України). 1
17. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.
18. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. - Київ : Держстандарт України, 1994. – 88 с.
19. 14. Node.js Documentation: https://nodejs.org/en/docs/ (дата звернення 26.10.2023).
20. FFmpeg Documentation: https://ffmpeg.org/documentation.html (дата звернення 26.10.2023).