МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ

ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»



Направление подготовки   
09.03.01 Информатика и вычислительная техника

Направленность (профиль)

«Технологии разработки программного обеспечения»

**Выпускная квалификационная работа**

Разработка VST плагина для цифровых звуковых рабочих станций

|  |
| --- |
| Обучающегося 4 курса  очной формы обучения  Столяренко Кирилла Анатольевича |
|  |
| Руководитель выпускной квалификационной работы:  Кандидат педагогических наук, доцент  Гончарова Светлана Викторовна |
|  |

Санкт-Петербург

2024

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 2](#_Toc167276856)

[**INTRODUCTION** 5](#_Toc167276857)

[**Глава 1 ПОСТРОЕНИЕ ТЕОРЕТИЧЕСКОЙ БАЗЫ ПРОЕКТА И ПОТГОТОВКА К РАЗРАБОТКЕ** 7](#_Toc167276858)

[1.1 Обзор форматов модульных расширений – плагинов и плагинов-эффектов. 7](#_Toc167276859)

[1.2 Сравнение средств разработки аудио плагинов. 19](#_Toc167276860)

[1.3 Вывод по главе 1. 26](#_Toc167276861)

[**Глава 2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА VST ПЛАГИНА – ЭФФЕКТА ДЛЯ ЦИФРОВЫХ ЗВУКОВЫХ РАБОЧИХ СТАНЦИЙ** 27](#_Toc167276862)

[2.1 Построение логики работы программного продукта с использованием языка UML. 27](#_Toc167276863)

[2.2 Создание и структурирование проекта JUCE. 33](#_Toc167276864)

[2.3 Написание кода. 35](#_Toc167276865)

[2.3.1 Создание параметров и логики их работы. 35](#_Toc167276866)

[2.3.2 Программирование эффекта перегрузки hard clipping. 40](#_Toc167276867)

[2.3.3 Программирование эффекта перегрузки soft clipping. 41](#_Toc167276868)

[2.3.4 Программирование эффекта сатурации. 42](#_Toc167276869)

[2.4 Тестирование итогового продукта. 42](#_Toc167276870)

[**СПИСОК ЛИТЕРАТУРЫ** 43](#_Toc167276871)

# **ВВЕДЕНИЕ**

В современном информационном обществе музыка имеет особую значимость, выступая не только как форма искусства и развлечения, но и как элемент культурной идентичности и социальной интеграции. В эру глобализации и цифровых технологий музыка становится мостом, соединяющим различные культуры и обеспечивающим обмен культурными ценностями на международном уровне. Кроме того, музыка играет ключевую роль в образовательных процессах, стимулируя когнитивное развитие, улучшая память и концентрацию, а также способствуя эмоциональному благополучию.

В эпоху всеобщей цифровизации технологические инновации неизбежно внедряются во все аспекты человеческой деятельности. Очень наглядно этот процесс отразился на творчестве. Можно долго говорить о том, как облегчилась жизнь тех же художников, или сколько принципиально новых видов самовыражения породили цифровые технологии, но данное исследование направлено на отражение влияния цифровизации на музыку.

Технологические инновации сопровождают музыку на протяжении всей истории ее существования. Начиная от изобретения струны, появления возможности записывать звук на носители, расширения горизонтов звучания различными обработками и звуковым дизайном, заканчивая возможностью полностью синтезировать любое, пришедшее автору на ум звучание, не выходя из комнаты.

Цифровой инструментарий звукового инженера растет и совершенствуется. На данный момент одним из самых распространенных средств, применяющихся как тинэйджерами, записывающими в своей спальне популярную музыку, так и композиторами, сочиняющими саундтрек к высокобюджетным фильмам - являются Цифровые звуковые рабочие станции (Digital Audio Workstations, далее - DAW). DAW представляют собой комплексные программные системы, играющие ключевую роль в современном культурном ландшафте. Они объединяют функционал огромного количества классических устройств, привнося при этом новые элементы звучания. Осуществляется это при помощи системы встраиваемых модулей – плагинов (от англ. Plug in - вставить).

Идея о разработке плагина - эффекта для DAW в рамках выполнения выпускной квалификационной работы пришла к автору настоящего исследования в связи с личной заинтересованностью в использовании и изучении внутреннего устройства таковых.

Актуальность работы обусловлена тем, что музыкальная индустрия переживает стремительный переход от аналоговых технологий к цифровым. Спрос на качественные аудиоинструменты и эффекты продолжает расти. Разработка новых плагинов удовлетворяет потребности музыкантов и звукорежиссеров, желающих расширять возможности звукового оформления. Также разработка плагина обладает крайне высокой образовательной ценностью, так как включает в себя множество аспектов: проектирование, программирование, цифровая обработка сигналов, пользовательский интерфейс и тестирование. Такое комплексное исследование награждает студента ценным опытом в различных областях компьютерных наук и аудиотехнологий. Публикация результатов разработки в свою очередь способствует расширению знаний в области звукотехнических разработок, а также укрепляет репутацию учебного заведения в академической и профессиональной сферах.

Разработка плагина - эффекта для DAW, как цель настоящего исследования позволит в результате его успешного завершения пополнить предложение качественным бесплатным программным продуктом с открытым кодом. При должном подходе проект обещает быть продуктивным как для обучающегося, так и для образовательного учреждения.

Для достижения поставленной цели были сформулированы следующие задачи:

- изучить теоретическую базу, необходимую для разработки плагина – эффекта для DAW;

- определиться с выбором вида эффекта, реализуемого в рамках выполнения выпускной квалификационной работы;

- изучить существующие инструменты разработки плагинов;

- сравнить существующие инструменты и выбрать наиболее подходящий;

- разработать и визуализировать информационно-логическую модель продукта средствами языка UML;

- разработать программный продукт;

- протестировать получившийся плагин на предмет работоспособности и совместимости в нескольких DAW.

# **INTRODUCTION**

In today's information society, music has a special significance, acting not only as a form of art and entertainment, but also as a significant element of cultural identity and social integration. In the era of globalization and digital technology, music becomes a bridge that connects different cultures and enables the exchange of cultural values on an international level. In addition, music plays a key role in educational processes, stimulating cognitive development, improving memory and concentration, and promoting emotional well-being.

In an era of universal digitalization, technological innovation is inevitably embedded in all aspects of human activity. This process has been reflected very clearly in creativity. We can talk for a long time about how the life of artists has become easier, or how many fundamentally new types of expression digital technologies have given rise to, but this study aims to reflect the impact of digitalization on music.

Technological innovation has accompanied music throughout its history. From the invention of the string, to the ability to record sound on media, to the expansion of sound horizons with various treatments and sound design, to the ability to fully synthesize any sound that comes to mind without leaving the room.

The sound engineer's digital toolkit is growing and improving. Currently, one of the most common tools used both by teenagers recording popular music in their bedrooms and by composers composing soundtracks for high-budget movies are Digital Audio Workstations (DAWs). DAWs are complex software systems that play a key role in today's cultural landscape. They combine the functionality of a huge number of classic devices, while introducing new elements of sound. This is accomplished through a system of plug-ins (Plug in).

The idea to develop a plug-in effect for DAW as part of the final qualification work came to the author of this study due to his personal interest in using and studying the internal structure of such plug-ins.

The relevance of the work is based on the fact that the music industry is experiencing a transition from analog to digital technologies. The demand for quality audio tools and effects continues to grow. The development of new plug-ins fulfills the needs of musicians and sound designers who wish to expand the possibilities of sound design. Also, plug-in development has an extremely high educational value because it involves many aspects: design, programming, digital signal processing, user interface, and testing. Such a comprehensive study rewards the student with valuable experience in various areas of computer science and audio technology. The publication of the development results in turn enhances knowledge in the field of audio engineering, and strengthens the reputation of the institution in academic and professional spheres.

The development of a plug-in effect for DAWs as a goal of this research will allow, as a result of its successful completion, to replenish the offer with a high-quality free and open source software product. With a proper approach, the project promises to be productive for both the student and the institution.

To achieve the goal, the following tasks were formulated:

- To study the theoretical basis necessary for the development of plug-in - effect for DAW;

- Determine the choice of the type of effect realized in the framework of the final qualification work;

- To study the existing tools of plug-in development;

- Compare existing tools and choose the most suitable one;

- Design a future program product using the UML language;

- Develop a program product;

- Test the resulting plug-in for performance and compatibility in several DAWs.

The object of this study is VST plug-in effect for digital sound workstations, and the subject is digital tools for synthesis and processing of sound signal.

# **Глава 1 ПОСТРОЕНИЕ ТЕОРЕТИЧЕСКОЙ БАЗЫ ПРОЕКТА И ПОТГОТОВКА К РАЗРАБОТКЕ**

* 1. Обзор форматов модульных расширений – плагинов и плагинов-эффектов.

На данный момент, если человек занимается какой-либо деятельностью, связанной с синтезированием и обработкой звуковой информации – он скорее всего использует DAW. Данное программное обеспечение крайне необходимо музыкантам, звукорежиссерам, дизайнерам звуковых эффектов для фильмов и игр, и присутствует как в домашней, так и в профессиональной звукозаписывающей студии. Цифровые станции объединяют в себе огромное количество функционала и объединяют в рамках одной программы на компьютере, или ноутбуке все громоздкие и дорогостоящие аппаратные приборы, которыми ранее пользовались для записи и сведения звука.

Цифровизация звука восходит к первой половине 20 века. В 1938 году Алеком Ривзом была подана заявка на патент фундаментальной по сей день для большинства методов записи и обработки звука в цифровом формате технологии конвертации аналогового звукового сигнала в цифровой формат - Pulse Code Modulation (PCM). Это изобретение положило начало долгому пути развития цифровых инструментов, что привело в конечном итоге к созданию DAW, которые развиваются и совершенствуются по сей день.

Сами по себе цифровые звуковые рабочие станции возникли в начале 90-х годов 20 века, как итог постепенного эволюционного развития и взаимодействия науки и инженерии. DAW были продуктом интеграции традиционных решений в новую оболочку, но в месте с этим и полем для совершенно новых изобретений. PCM, технология MIDI, являющиеся по полному праву фундаментами подобного ПО на ряду с многими другими нововведениями являются результатами многолетнего процесса развития. Подробный анализ такового даст возможность понять, какие программные и аппаратные нововведения способствовали настолько широкой экспансии DAW в рамках музыкальной индустрии.

Временной промежуток с 1950-х по 1970-е был достаточно продуктивным. Цифровой формат звука укреплялся и рос, ученые продолжали активные исследования и эксперименты в этой области. Альбомы классических музыкантов и джазовых исполнителей постепенно перекочёвывали в цифровую форму.

В 1977 году компания MICRO TECHNOLOGY UNLIMITED разработала и запустила дистрибьюцию программного обеспечения, позволяющего синтезировать музыку при помощи актуальнейшего на тот момент компьютера Apple II, произведенного на свет Стивом Джобсом и Стивом Возняком

Всего через год, компания Sound Stream разработала систему под названием Digital Editing System. Эта система была оснащена интерфейсами Unibus, что позволяло осуществлять ввод и вывод аудиосигналов как с аналоговых, так и с цифровых источников, включая магнитофонные аппараты. Основой системы был миникомпьютер DEC PDP-11/60, который функционировал с помощью программного обеспечения Digital Audio Processor (DAP). Комплекс также оснащался 14-дюймовым жестким диском, осциллографом для визуализации сигналов и видеодисплеем, который предоставлял пользователям возможность управления системой.

Следующее событие можно считать отправным пунктом развития DAW, ведь в 1979 году Питером Вогелем была создана система, позволяющая записывать звук на дискеты - FAIRLIGHT CMI (смотреть приложение B), Она объединила в себе синтезатор, сэмплер и секвенсор. Подобно Digital Editing System FAIRLIGHT CMI состояла из нескольких модулей, объединенных в единое целое. К системному блоку были подключены:

- клавиши в 5 октав;

- сэмплер;

- монитор;

- перо;

- преобразователь сигнала (ADC/DAC);

- QWERTY клавиатура;

Звук синтезировался путем синтеза Фурье. Монитор поддерживал возможность манипуляции пером для ручной отрисовки синусоидных звуковых волн, а также для манипуляций внутри семплера, позволяющего управлять записанными в цифровом формате кусками нарезанной звуковой дорожки.

Введение стандарта MIDI (Musical Instrument Digital Interface) в 1983 году стало революционным событием в музыкальной индустрии, оказав существенное влияние на развитие цифровых аудиостанций (DAW - Digital Audio Workstations). Это событие не только изменило подход к созданию и редактированию музыки, но и заложило основу для эры цифровой музыки, как мы её знаем сегодня. Рассмотрим подробнее, каким образом MIDI способствовал эволюции DAW и какие ключевые преимущества он принес в музыкальное производство.

До появления MIDI каждый производитель электронных музыкальных инструментов использовал собственные уникальные стандарты для передачи музыкальных данных, что делало невозможным их взаимодействие между собой. MIDI решил эту проблему, предложив универсальный формат данных, который позволял устройствам разных брендов «общаться» друг с другом. Эта стандартизация была критически важна для развития DAW, так как музыканты и продюсеры теперь могли интегрировать разнообразное оборудование в единую систему.

С помощью MIDI музыканты получили возможность использовать широкий арсенал звуков и инструментов, которые были бы недоступны или невозможны для воспроизведения вживую. Такая универсальность делает MIDI незаменимым инструментом в современных DAW, позволяя музыкантам экспериментировать с различными музыкальными жанрами и стилями.

MIDI значительно упростил процесс создания музыки. Не требуется больше владеть навыками игры на всех музыкальных инструментах, которые звучат в композиции, или арендовать профессиональную студию для записи трека. MIDI-контроллеры и DAW позволяют музыкантам записывать и редактировать свои произведения в комфортных домашних условиях, делая музыкальное творчество доступным широкому кругу лиц.

Одной из важнейших особенностей MIDI является возможность точно контролировать каждый аспект музыкального выражения — от силы звука до тонкости вибрато. Это позволяет музыкантам добиваться высокой выразительности и эмоциональной насыщенности в своих записях, что раньше было возможно только при живом исполнении.

MIDI-данные, записанные в DAW, могут быть легко отредактированы и изменены, что предоставляет музыкантам возможность доработать свои треки после первоначальной записи.

Необходимо оговориться о том, что персональные компьютеры в 80-х и сейчас – это разные вещи и многие из вышеперечисленных возможностей были накоплены в последствии.

Одними из самых распространенных ПК на тот момент были COMMODORE 64 и ATARI ST. На эти платформы и пал выбор первопроходцев, основавших в 1984 году компанию Steinberg. Эта компания в дальнейшем создаст ту самую технологию VST, позволяющую устанавливать модульные расширения в широчайший спектр DAW.

Первой разработкой компании Steinberg был MIDI - секвенсор – PRO -16, получивший свое логическое продолжение в лице усовершенствованной версии для ATARI под названием CUBASE, которая переросла в полноценную DAW, поддерживающуюся по сей день с тем же названием.

Первой увидевшей свет DAW принято считать детище компании Steinberg - Cubase Audio Mac выпущенное для mac в 92 году. Первым ПО, имеющим возможность обрабатывать сигнал без внешнего устройства - также было детище Steinberg. Это был рекордер для записи, поставляющийся для компьютера ATARI Falcon – CUBASE AUDIO FALCON.

Внедрение технологии Virtual Studio Technology (VST) компанией Steinberg в 1996 году в своей программе Cubase VST представляет собой один из ключевых моментов в истории развития цифровых аудиостанций (DAW). Это нововведение не просто расширило функциональные возможности программного обеспечения для создания музыки, но и радикально изменило саму концепцию звукозаписи, положив начало переходу от аналоговых методов к полностью цифровым рабочим процессам.

Технология VST включала в себя несколько прорывных для того времени особенностей: возможность использования до 32 звуковых дорожек, встроенное редактирование и запись MIDI, а также поддержка многочисленных встроенных и сторонних плагинов для обработки звука. Это позволило VST стать первой в мире технологией, которая могла полностью воссоздать парадигму современной студии звукозаписи в цифровом формате, включая многодорожечную запись с набором эффектов и автоматическим микшированием.

Прежде всего, VST разрушило традиционные барьеры между музыкальными производителями и технологиями, позволяя разработчикам создавать "плагины", которые могут быть интегрированы в любую DAW, поддерживающую VST. Это означало, что музыканты и продюсеры получили доступ к неограниченному количеству аудиоинструментов и эффектов, значительно расширяя их творческие возможности.

С появлением плагинов VST и VSTi (Virtual Studio Technology instruments) произошло ещё одно значительное новшество — возможность программного моделирования музыкальных инструментов. Например, первый стандартный синтезатор NEON для Cubase представил новую эру в создании звуков, где физические устройства могли быть заменены их цифровыми аналогами. Это не только упростило процесс создания музыки, но и сделало его более доступным.

С технической точки зрения, VST представляет собой комплексный алгоритмический подход, позволяющий обрабатывать цифровой звуковой сигнал в реальном времени. Разработка таких плагинов требует глубоких знаний в области цифровой обработки сигналов, программирования и акустики. Эти плагины могут имитировать всё, от простых эффектов (реверберация, дилэй) до сложных модуляций и синтеза.

Технология VST значительно повлияла на стандартизацию в индустрии звукозаписи. Благодаря её широкому распространению многие программы для создания музыки начали поддерживать общий стандарт плагинов, что облегчило совместную работу музыкантов и продюсеров из разных частей мира. Также VST ускорило отказ от аналоговой звукозаписи, так как музыканты получили возможность делать записи высокого качества без необходимости использования дорогостоящего аналогового оборудования.

Таким образом, технология VST от Steinberg не только улучшила технические возможности музыкального производства, но и преобразила подходы к записи и производству музыки на глобальном уровне. VST является не просто технологическим достижением, это культурный феномен, который помог музыкантам и продюсерам реализовать их творческий потенциал в полной мере, делая музыкальное искусство более доступным и многообразным.

В 1997 году компания Image-Line представила первую версию программы Fruity Loops, которая в дальнейшем была переименована в FL Studio. Изначально разработанная как четырёхканальная MIDI драм-машина, FL Studio со временем трансформировалась в мощную полноценную цифровую аудиостанцию (DAW). Эта эволюция отражает общий тренд перехода от простых музыкальных инструментов к комплексным системам для профессиональной звукозаписи, используемым ведущими музыкальными продюсерами.

Процесс окончательного становления DAW как доминирующего формата ПО занял около десяти лет, что довольно немного. Основные причины этого — высокое качество звука, возможности неразрушающего редактирования и автоматизированного микширования, которые стали доступны в цифровом формате. Эти характеристики существенно превосходили возможности аналоговых магнитофонных лент, которые требовали использования отдельных микшеров и процессоров и предлагали ограниченные функции редактирования. Сложность работы с аналоговым оборудованием и ограниченные возможности пост-продакшена стимулировали переход к более гибким и доступным цифровым системам, поддерживающим интенсивное развитие музыкальной индустрии.

Плагин (от английского "plugin") - это программное дополнение, которое расширяет функциональность или возможности программного обеспечения. Подобно модулям или расширениям, плагины позволяют пользователям настраивать и дополнять основной функционал программы в соответствии с их потребностями или предпочтениями. Плагины могут выполнять различные задачи, от добавления новых функций до обеспечения интеграции с другими программами или сервисами.

Основные характеристики плагинов включают:

- Расширение функционала: Плагины могут добавлять новые функции или возможности к основному программному обеспечению, что позволяет пользователям адаптировать программу под свои конкретные потребности.

- Модульность: Плагины обычно являются модульными и могут быть добавлены или удалены из программы без изменения ее основного кода. Это обеспечивает гибкость и расширяемость программного обеспечения.

- Интеграция: Плагины могут интегрироваться с другими программами или сервисами, расширяя возможности взаимодействия между ними.

- Переносимость: Плагины могут быть разработаны для работы с различными платформами или операционными системами, что обеспечивает их переносимость и доступность для широкого круга пользователей.

Примеры плагинов включают расширения браузера, добавляющие новые функции (например, блокировщики рекламы), плагины для графических редакторов, предоставляющие дополнительные инструменты обработки изображений, и плагины для аудио-видео плееров, позволяющие воспроизводить файлы в различных форматах.

В современной музыкальной индустрии плагины являются неотъемлемым компонентом цифрового аудиопроизводства, представляя собой программные модули, которые дополняют или расширяют функциональность базовых аудиоприложений. Плагины для цифровых аудиостанций (DAW — Digital Audio Workstation) обеспечивают музыкантов и звукорежиссеров инструментами для имитации акустических эффектов, звуковых обработок и виртуальных инструментов, которые традиционно требовали физического оборудования. Эти плагины могут включать в себя синтезаторы, семплеры, эквалайзеры, компрессоры, ревербераторы и множество других обработок звука, инструментов и тому подобного, позволяя пользователям манипулировать аудиосигналами в высокой степени детализации и контроля.

Технологически плагины для DAW интегрируются через стандартизированные форматы, такие как VST (Virtual Studio Technology), AU (Audio Units), AAX (Avid Audio eXtension) … каждый из которых обладает собственными особенностями в плане совместимости с различными операционными системами и хост-приложениями. Эти стандарты не только упрощают процесс интеграции новых плагинов в различные DAW, но и поддерживают обмен данными между плагином и хостом, обеспечивая эффективную обработку и минимизацию задержек.

Введение VST2 в 1999 году компанией Steinberg стало критическим моментом в эволюции цифровых аудио рабочих станций (DAW). Этот формат не только позволил копировать и воспроизводить звуковые эффекты и обработки, но и принёс возможность интеграции виртуальных музыкальных инструментов. Таким образом, компьютеры начали заменять не только аппаратные устройства, но и реальных исполнителей, что стало возможным благодаря разработке сторонними производителями виртуальных версий гитар, ударных установок и синтезаторов.

Создание конкурирующих форматов плагинов крупными игроками технологической индустрии, такими как Apple, Avid (ранее Digidesign) и Microsoft, подчёркивает важность и популярность VST. Apple адаптировала VST для macOS, создав формат AU, в то время как Avid предложила RTAS, совместимый с Pro Tools. Microsoft внесла свой вклад, разработав формат DX, основанный на DirectX. Эти разработки не только поддерживали инновации, но и способствовали стандартизации и унификации в области цифровой звукозаписи.

Несмотря на разнообразие форматов плагинов, многие музыканты и продюсеры по-прежнему используют термин "VST-плагин" в обобщающем смысле для обозначения любых виртуальных инструментов и эффектов. Это обобщение, хотя и удобно в разговорной речи, может вносить путаницу, особенно для начинающих пользователей, которые могут не понимать различия между форматами и их совместимость с различными DAW. Определение и понимание этих различий имеет важное значение для эффективного выбора инструментов для обработки звука.

В рамках данного параграфа настоящего исследования для полного отражения многообразия будут перечислены даже устаревшие форматы плагинов и форматы со слишком узкой областью применения. Таковыми на момент проведения настоящего исследования являются:

- DX – устаревший стандарт, разработанный Microsoft в коллаборации с Cakewalk. Формат непопулярен из-за поддержки в малом количестве DAW, но до сих пор может быть актуален из-за возможности его применения внутри среды более ранних версий 32-битной операционной системы Windows благодаря использованию технологии DirectX.

- RTAS – устаревший стандарт плагинов, применявшийся ранее эксклюзивно для модификации DAW под названием Pro Tools. На данный момент неактуален в следствие смены формата расширений для Pro Tools на новый AAX

- ReFill – стандарт, использующийся эксклюзивно в DAW под именем Reason от компании Reason Studios. Reason более не поддерживается разработчиками как самостоятельная DAW, она стала отдельным плагином, в форматах VST и AU.

- LV2 – стандарт, использующийся в основном для расширения DAW на платформе Linux. Позиционируется как бесплатная и открытая альтернатива популярным стандартам, но в виду довольно узкого комьюнити музыкантов использующих Linux имеет не такую большую библиотеку портированных плагинов.

Основными же для сегодняшней музыкальной индустрии являются форматы: VST, AU и AAX.

- VST – это самый массовый стандарт, который поддерживается огромным спектром DAW как на Windows и Mac, так и на Linux. Компания Steinberg, разработавшая его являлась первопроходцем, что не вызвало затруднений и никак не помешало успешно закрепиться в лидерской позиции на рынке. Этот формат плагинов переживает уже третью свою итерацию.

- AU – формат разработанный компанией Apple. Представляет собой слегка модифицированную для лучшего функционирования с macOS версию VST. Большая часть плагинов выпущенных во второй и третей итерациях VST портированы также и в AU формат. Перенос осуществляется легко благодаря сходному алгоритму работы. AU также пережила несколько итераций. Самой свежей на данный момент является AUv3, поддерживающая совместимость с iOS и iPadOS для работы в DAW на планшетах и смартфонах от Apple.

- AAX является новым эксклюзивным форматом для Pro Tools, поддерживающимся с 11 версии данной DAW. Отличительной особенностью является ориентированность на уменьшение нативности, присущей VST и AU расширениям. ОС от Avid и расширения, разработанные на безе их стандарта отличается возможностью полноценного использования DSP (Digital Signal Processing) чипов и прочих компонентов внешних аудио интерфейсов.

Ввиду личных предпочтений и наиболее высокой популярности стандарта VST выбор был отдан в пользу разработки плагина на его основе.

В прошлых параграфах первой главы настоящего исследования часто упоминались плагины – эффекты.

Плагины-эффекты, также известные как аудио-эффекты или просто эффекты, это программные инструменты, которые применяются к аудиосигналам для изменения их звучания или обработки.

Плагины-эффекты могут выполнять различные функции, такие как изменение тембра (частотного спектра), времени (задержки, эхо), динамики (компрессия, лимитирование), пространственных характеристик (реверберация, стереоэффекты) и многое другое. Они используются как для творческих целей, так и для улучшения качества звука или коррекции ошибок.

Плагины-эффекты играют важную роль в современной музыкальной и звуковой продукции, позволяя звукорежиссерам, музыкантам и звуковым инженерам создавать уникальные звуковые образы и достигать желаемого звучания. Они предоставляют широкий спектр инструментов для творчества и профессиональной обработки звука.

На основе анализа наиболее широко используемых расширений автором данной выпускной квалификационной работы была составлена следующая классификация плагинов – эффектов по принципу их работы.

- Динамические обработчики

* Компрессоры: уменьшают динамический диапазон аудиосигнала, снижая уровень громких звуков и увеличивая тихие. Это улучшает общую громкость и четкость звучания.
* Лимитеры: ограничивают максимальный уровень сигнала, предотвращая его искажение.
* Экспандеры и шумоподавители: увеличивают динамический диапазон, уменьшая уровень звуков ниже определённого порога, что помогает убирать фоновый шум.
* Гейты: позволяют сигналу проходить только когда его уровень превышает установленный порог, что используется для устранения нежелательных шумов и эхо.

- Фильтры

* Эквалайзеры (EQ): изменяют баланс частот в аудиосигнале, позволяя увеличивать или уменьшать определённые диапазоны частот для формирования тембра.
* Фильтры высоких/низких частот: блокируют частоты выше или ниже заданного порога.

- Модуляционные эффекты

* Фланжеры, фазеры и хорусы: создают эффекты движения и пространства за счет модуляции исходного сигнала с помощью переменной задержки.
* Тремоло и вибрато: модулируют амплитуду или частоту сигнала, добавляя ритмичные изменения громкости или высоты тона.

- Эффекты временной задержки

* Эхо и реверберация: имитируют звуковые отражения от поверхностей, добавляя пространственное ощущение и глубину звука.
* Дилей (Delay): задерживает аудиосигнал на определенное время, создавая эффект эха или повторения.

- Спектральные обработчики

* Эксайтеры: улучшают яркость и четкость звучания за счет добавления гармоник в верхние частоты.
* Сатураторы: имитируют эффект насыщения, который происходит при записи на аналоговые носители, добавляя теплоту и плотность звука.

- Эффекты перегруза

* Дисторшн: эффект, создающий перегруженное и жесткое звучание путем повышения амплитуды и клиппинга (ограничения амплитуды) для получения более «плоской» волны.

- Утилиты и специализированные эффекты

* Метрономы, тюнеры, анализаторы спектра: предоставляют вспомогательные функции для контроля и анализа аудиосигналов.
* Vocoder и Auto-Tune: используются для обработки голоса, изменения его тембра и коррекции высоты тона.

По итогам проведенного анализа наиболее реальной и практически значимой в осуществлении целью представляется разработка VST плагина, сочетающего в себе функционал дисторшена (реализованного при помощи 2-х алгоритмов) и сатуратора. Данные эффекты выполняют сходные функции и служат для предания звуку большей окрашенности, полноты, плотности, а также имеют характерное и явно различимое звучание

Практическая значимость по мнению автора данного исследования заключается в механизме работы данных эффектов, ведь их работа заключается не в манипуляциях с буфером и памятью, а в изменении формы звуковой волны, что можно продемонстрировать наглядно при помощи графиков, а также рассчитать, учитывая законы физики.

* 1. Сравнение средств разработки аудио плагинов.

Для разработки аудио плагинов на данный момент существует не мало средств. В большинстве своем такие средства используют два наиболее популярных подхода к программированию, а именно:

* Модульное конструирование (визуальное программирование);
* Написание кода с использованием специализированных фреймворков.

Конструкторы гораздо легче в освоении по причине отсутствия надобности в изучении языка программирования и синтаксиса используемого фреймворка. Далее будет рассмотрено несколько подобных сред, а именно:

- SynthEdit

- FlowStone

- Max/MSP

SynthEdit – это среда, разработанная Джеффом МакКлинтоком и выпущенная летом 2005 года, предназначается для создания собственных синтезаторов в форме VST плагинов. Она была разработана с учетом потребностей аудио-инженеров и музыкантов в создании индивидуальных звуковых решений, основанных на графическом подходе к программированию.

SynthEdit предлагает широкий набор готовых модулей и блоков, которые пользователи могут комбинировать и настраивать в соответствии с требуемыми звуковыми характеристиками. Эти модули включают в себя различные типы осцилляторов, фильтров, усилителей, эффектов и множество других аудио-компонентов. Пользователи могут соединять эти модули, используя графический интерфейс SynthEdit, что позволяет создавать сложные аудио-схемы и довольно эффективно управлять звуковыми процессами.

Одной из ключевых особенностей SynthEdit является его простота в использовании. Благодаря интуитивно понятному интерфейсу и множеству доступных модулей, даже непрофессиональные пользователи могут быстро создавать свои собственные звуковые эффекты и синтезаторы. Благодаря этому SynthEdit часто используется как обучающий инструмент для введения в мир звукового программирования.

Технический аспект SynthEdit также заслуживает внимания. Он предоставляет механизмы для оптимизации и тестирования созданных плагинов.

FlowStone — это IDE, предназначенная для создания собственных звуковых эффектов в форме VST. Разработанная фирмой DSPRobotics, эта платформа обеспечивает мощные инструменты для аудио-программирования, позволяя пользователям создавать сложные звуковые решения с использованием графического подхода.

FlowStone по аналогии с SynthEdit предоставляет широкий набор функциональных блоков, известных как "компоненты", которые пользователи могут комбинировать и настраивать, создавая уникальные аудио-процессы. Эти компоненты включают в себя осцилляторы, фильтры, генераторы оболочки, эффекты и другие звуковые модули. Пользователи могут соединять эти компоненты вместе, создавая сложные аудио-схемы, которые реагируют на входные сигналы и производят желаемые звуковые результаты.

Одной из ключевых особенностей FlowStone является его гибкость и расширяемость. Пользователи могут создавать собственные компоненты и библиотеки, расширяя функциональные возможности платформы и адаптируя ее под свои потребности. Это делает FlowStone привлекательным выбором для разработчиков, стремящихся создать уникальные звуковые эффекты и инструменты.

Платформа также обеспечивает низкую задержку обработки аудио, что делает ее идеальным выбором для профессиональных звукорежиссеров и музыкантов.

Max/MSP — это программное обеспечение для создания мультимедийных приложений, включающее в себя среду для визуального программирования. Оно позволяет пользователям создавать и модифицировать звуковые и видеоэффекты в реальном времени, используя графический интерфейс.

В Max/MSP пользователи могут создавать аудио и видеопроцессы, используя различные модули и объекты, которые связываются вместе в виде "патчей" - графических представлений алгоритмов обработки сигнала. Эти объекты могут быть использованы для генерации, обработки, модуляции и манипуляции аудио и видеосигналами.

Кроме того, Max/MSP позволяет создавать интерфейсы для управления аудиоэффектами и параметрами, что делает его очень гибким инструментом для создания пользовательских звуковых приложений.

В контексте создания VST плагинов, Max/MSP может быть использован для разработки звуковых алгоритмов, создания пользовательского интерфейса и интеграции с другими программами и устройствами. Это позволяет разработчикам создавать инновационные звуковые эффекты и инструменты, которые могут быть интегрированы в существующие музыкальные среды.

На основании вышеперечисленных особенностей трех различных сред разработки можно сделать вывод о том, что средства, предлагающие функционал конструкторов для визуального программирования плагинов, часто используют набор готовых модулей, соединяющихся между собой цепью посылов. Данный подход очевидно гораздо менее гибкий и не способен выдать полностью уникальное звучание. Хотя существуют и более усредненные варианты, использующие кастомные модули, или позволяющие самостоятельно запрограммировать какой-либо блок, разработанные в таких средах плагины, как правило более ресурсоемкие, чем плагины, разработанные с использованием фреймворков, из-за чего не подходят для больших проектов с длинной цепью обработок. Для реализации плагина – эффекта, подразумевающего своей сутью участие в цепях обработок различной сложности, такой подход видится автору данного исследования непригодным.

Говоря о втором подходе к разработке следует понимать, что хотя визуальное программирование также имеет свои преимущества, такие как более интуитивный интерфейс, возможность быстрого прототипирования и более низкий порог вхождения для новичков в программировании, для разработки крупных и профессиональных VST-плагинов обычно предпочтительнее использовать написание кода.

Самостоятельная работа с кодом позволяет разработчику иметь полный контроль над каждым аспектом плагина. Это обеспечивает большую гибкость в реализации сложных функций и алгоритмов обработки звука. Также данный подход дает возможность оптимизировать производительность плагина за счет оптимизации кода, управления памятью и использования эффективных алгоритмов обработки звука. Визуальные среды программирования могут ограничивать возможности оптимизации. Самостоятельное написание кода требует более глубокого понимания алгоритмов обработки звука и архитектуры VST-плагинов. Это позволяет разработчику лучше понимать внутреннее устройство и поведение плагина, что важно для его оптимизации и отладки.

Существует широкий выбор инструментов, библиотек и фреймворков для разработки аудио-приложений на языках программирования. Эти инструменты обеспечивают разработчиков мощными средствами для создания высококачественных и производительных VST-плагинов.

Подавляющее большинство библиотек, предназначенных для создания аудио плагинов действует в рамках языка C++. Этот факт заранее определил выбор языка программирования, использующегося в дальнейшей разработке.

На основе проведенного анализа форумов, где разработчики аудио софта общаются между собой автор настоящего исследования, выделил три лидирующих по предпочтению пользователей фреймворка на базе языка C++, которыми являются:

- WDL-OL;

- iPlug2.

- JUCE;

Проведем сравнительный анализ, на основе которого будет отдано предпочтение одному из трех фреймворков, для дальнейшего его применения в разработке.

WDL-OL (WDL-Object Language) является частью WDL (Whirlwind Development Language), который представляет собой диалект языка C++ и используется для создания аудио-плагинов.

WDL-OL расширяет возможности WDL, добавляя поддержку объектно-ориентированного программирования (ООП). Он позволяет разработчикам описывать аудио-эффекты, синтезаторы и другие аудио-процессы в виде объектов с использованием классов и методов.

Основные характеристики WDL-OL:

- Объектно-ориентированное программирование (ООП): WDL-OL поддерживает основные принципы объектно-ориентированного программирования. Это позволяет разработчикам создавать модульные и гибкие архитектуры плагинов.

- Простота и читаемость кода: WDL-OL обладает простым и понятным синтаксисом, который упрощает написание и понимание кода. Это делает разработку плагинов более эффективной и удобной.

- Поддержка VST: WDL-OL предоставляет инструменты и библиотеки для создания аудио-плагинов формата VST (Virtual Studio Technology). Это позволяет разработчикам создавать плагины, которые могут использоваться в различных аудио-средах и DAW (Digital Audio Workstation).

- Кроссплатформенность: WDL-OL обеспечивает кроссплатформенность, что позволяет разработчикам создавать плагины, которые могут работать на различных операционных системах, таких как Windows, macOS и Linux.

- Активное сообщество: WDL-OL поддерживается активным сообществом разработчиков, которые предоставляют учебные материалы и примеры кода для помощи новичкам и опытным разработчикам.

iPlug2 — это современный фреймворк для создания аудио-плагинов, который предоставляет разработчикам удобные инструменты для быстрой и эффективной имплементации плагинов. Вот несколько удобств, связанных с использованием iPlug2:

- Кросс-платформенность: iPlug2 поддерживает разработку аудио-плагинов для различных операционных систем, включая Windows, macOS, iOS и Android. Это позволяет создавать плагины, которые могут работать на разных платформах без изменений в исходном коде.

- Модульность и расширяемость: Фреймворк iPlug2 построен на основе модульной архитектуры, что облегчает добавление новых функций и компонентов в плагины. Разработчики могут легко создавать собственные аудио-эффекты, инструменты и другие компоненты, используя предоставляемые инструменты и API.

- Интеграция с современными технологиями: iPlug2 поддерживает интеграцию с современными технологиями, такими как OpenGL и WebAssembly, что позволяет создавать графически интерактивные пользовательские интерфейсы и расширять возможности плагинов.

- Богатая функциональность GUI: iPlug2 предоставляет разработчикам широкий набор инструментов для создания пользовательских интерфейсов плагинов. Он поддерживает различные виджеты, анимации и стилизацию интерфейсов, что позволяет создавать современные и привлекательные GUI для плагинов.

JUCE (Jules' Utility Class Extensions) - это кросс-платформенная библиотека для разработки мультимедийных приложений и плагинов. Она была создана Джулзом Стэнтоном и используется для создания музыкальных инструментов, аудио-программного обеспечения, игр и других приложений, работающих на различных операционных системах. JUCE предоставляет обширный набор инструментов для работы с аудио, графикой, пользовательским интерфейсом и другими аспектами разработки программного обеспечения. Он поддерживает такие платформы, как Windows, macOS, iOS, Android и Linux, что делает его очень гибким инструментом для создания кроссплатформенных приложений. JUCE также имеет широкое сообщество разработчиков, что делает его популярным выбором для профессиональной разработки аудио-приложений и плагинов. Комьюнити является особенностью, которая отличает JUCE от его конкурентов, он попросту популярнее. Наличие очень подробной официальной документации, которая значительно более развернутая, чем у двух представленных конкурентов упрощает работу с ним. Различных видеоуроков, связанных с программированием плагинов при помощи C++ и Juce значительно больше чем с применением WDL-OL и iPlug2. Такая мощная поддержка послужила фактором, снижающим порог вхождения и для разработчика слабо знакомого с синтаксисом C++ ключевым аргументом, обусловившим выбор, сделанный автором данной работы.

* 1. Вывод по главе 1.

В процессе написания первой главы настоящей исследовательской работы были выполнены следующие задачи:

- Были даны определения основным понятиям;

- Была изучена история появления и развития цифровых звуковых рабочих станций;

- Была выстроена теоретическая база, необходимая для понимания дальнейшей практической части исследования;

- Были изучены форматы плагинов, а также обоснован выбор в пользу разработки плагина формата VST;

- Были выбраны эффекты, которые будет необходимо реализовать в практической главе настоящей выпускной квалификационной работы;

- Был проведен сравнительный анализ двух подходов к разработке;

- Был проведен сравнительный анализ средств разработки, с помощью которого сделан выбор в пользу фреймворка JUCE на базе языка программирования C++.

# **Глава 2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА VST ПЛАГИНА – ЭФФЕКТА ДЛЯ ЦИФРОВЫХ ЗВУКОВЫХ РАБОЧИХ СТАНЦИЙ**

* 1. Построение логики работы программного продукта с использованием языка UML.

Первым шагом в реализации подобных проектов как правило является создание некой первичной абстрактной схемы. Такой подход позволяет еще на ранних этапах определиться с функционалом, что в свою очередь хорошо сказывается на таких вещах, как оценка собственных возможностей и построение реалистичного графика работы, позволяет создавать архитектуру будущего продукта последовательно, что значительно снижает процент ошибок в коде, хорошо отражается на его читаемости и просторе для оптимизации.

Традиционно на этапах, связанных с проектированием какого-либо программного продукта, используется язык UML. Использование UML позволяет улучшить качество и эффективность разработки, обеспечивая более глубокое понимание требований к системе и ее структуры, а также обеспечивая единый и понятный язык коммуникации между участниками проекта.

Unified Modeling Language (UML) является стандартом для визуализации, проектирования и документирования системных архитектур и процессов в программной инженерии. Разработанный объединением усилий лучших специалистов в области информационных технологий, UML предоставляет мощный набор инструментов для моделирования различных аспектов программных систем.

Важность этого этапа разработки трудно переоценить, ведь UML помогает разработчикам лучше понять требования к системе и ее структуру. Это позволяет избежать недопонимания и ошибок во время разработки. Создание UML-диаграмм позволяет визуализировать архитектуру программы, выделяя компоненты, их связи и взаимодействия. Это помогает выявить потенциальные проблемы и улучшить дизайн системы. Использование UML способствует поддержанию согласованности между членами команды разработки и заказчиком. Все участники проекта могут использовать общий язык для обсуждения требований и принятия решений. UML-диаграммы могут служить отличной формой документации, описывающей структуру и поведение программы. Это упрощает поддержку и развитие системы в дальнейшем, а также обучение новых членов команды.

Это облегчает понимание системы как разработчиками, так и пользователями. Используя UML, разработчики могут провести более глубокий анализ требований к системе и разработать более точные и эффективные решения на этапе проектирования. Это позволяет выявить потенциальные проблемы и улучшить качество конечного продукта. UML дает возможность создавать наглядную документацию, описывающую архитектуру и функциональные требования к системе. Это помогает упростить поддержку и развитие системы в будущем.

Первой диаграммой, созданной на этапе проектирования была диаграмма деятельности, представленная после данного абзаца. Эта диаграмма представляет собой графическое представление динамического поведения системы, где действия представлены узлами, а переходы между действиями - стрелками. При помощи данной специфики была построена диаграмма, воссоздающая процесс взаимодействия пользователя с плагином.

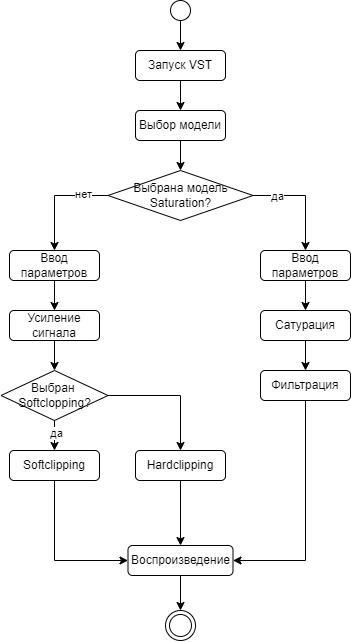


Рисунок 1 "Диаграмма деятельности"

Концепцией, наиболее откликающейся в диаграмме, является наличие трех разных алгоритмов обработки звука. Алгоритмы, подразумевающие собой дисторшн объединены в одной ветке, так как их объединяет усиление сигнала перед обоими частными случаями клиппинга. Сатурация же вынесена в отдельную ветку по причине существенной разницы в методе искривления волны, а также наличия фильтра. Диаграмма была создана в графическом редакторе draw.io.

Далее было принято решение построить диаграмму, отражающую логику взаимодействия между пользователем, VST и цифровой звуковой рабочей станцией (DAW). Для отражения подобного рода отношений обычно применяют диаграмму последовательности, которая в рамках Unified Modeling Language (UML) представляет собой графическое изображение последовательности взаимодействия между объектами в системе в течение определенного времени. Этот тип диаграммы акцентирует внимание на временных и пространственных аспектах взаимодействия, иллюстрируя порядок отправки сообщений между объектами в терминах временных линий, называемых "жизненными линиями". Каждый объект представлен своей жизненной линией, а сообщения между объектами отображаются в виде стрелок, указывающих направление передачи информации. Диаграммы последовательности позволяют детально анализировать взаимодействие между объектами, учитывая их последовательность и условия выполнения.

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

Рисунок 2 "Диаграмма последовательности"

Во время изучения данного процесса взаимодействия было понято, что логика запуска и модульного включения плагина в цепь обработок реализована полностью на стороне DAW. Данная диаграмма была создана при помощи web версии PlantUML. Код представлен далее.

Код:

@startuml

actor User

participant VST

participant DAW

User -> DAW: Запуск плагина

activate DAW

DAW -> VST: Запуск

activate VST

VST -> User: Интерфейс

User -> VST: Ввод параметров

VST -> DAW: Обработка сигнала

deactivate VST

DAW -> User: Воспроизведение

deactivate DAW

@enduml

Следующей диаграммой, выбранной для создания, была диаграмма состояний, являющаяся в рамках UML инструментом моделирования, который позволяет описывать динамическое поведение объекта или системы в зависимости от его состояния и событий, которые влияют на это состояние. Каждое состояние представляется как прямоугольник с указанием его имени, а переходы между состояниями представлены стрелками, аннотированными условиями, при которых происходит переход. Диаграммы состояний акцентируют внимание на переходах между состояниями объекта или системы, а также на действиях, которые выполняются при смене состояний.

Изображение выглядит как текст, диаграмма, снимок экрана, План

Автоматически созданное описание

Рисунок 3 "Диаграмма состояний"

Эта диаграмма в состоянии, актуальном на момент проектирования сильнее всего отражает многофункциональность плагина и указывает на концепцию смены модели обработки. Выполнена данная диаграмма состояний при помощи web версии PlantUML. Код представлен ниже.

Код:

@startuml

state VST {

[\*] --> Запуск\_плагина

Запуск\_плагина --> Ввод\_параметров

Ввод\_параметров --> Выбор\_модели\_обработки

Выбор\_модели\_обработки --> Hard: Hard

Выбор\_модели\_обработки --> Soft: Soft

Выбор\_модели\_обработки --> Saturation: Saturation

Hard --> Воспроизведение

Soft --> Воспроизведение

Saturation --> Воспроизведение

}

state Hard {

[\*] --> Обработка\_H

}

state Soft {

[\*] --> Обработка\_S

}

state Saturation {

[\*] --> Обработка\_Sn

Обработка\_Sn --> Фильтрация

}

@enduml

* 1. Создание и структурирование проекта JUCE.

Создание проекта в JUCE происходит по средствам взаимодействия с приложением Projucer, собирающимся на этапе установки Фреймворка. Приложение предоставляет удобный графический интерфейс, позволяющий создать проект, настроить параметры автоматически генерирующихся фреймворком блоков, а также создать, или удалить директории и файлы проекта для его эффективного структурирования.

Во время первоначальной настройки проекта при его создании было указано имя компании - «Herzen» и проставлены маркеры у необходимых параметров.

Также для улучшения читаемости кода и облегчения дальнейшей работы с ним были созданы директория «DSP»[[1]](#footnote-1), содержащая файлы «Distortion.cpp» и «Distortion.h» и директория «Parameters», содержащая файлы «Parameters.cpp» и «Patameters.h». Директория «DSP» включает в себя файлы, содержащие код, осуществляющий работу трех, запланированных на этапе проектирования моделей обработки, а директория «Parameters» содержит файлы, в которых задаются глобальные параметры, использующиеся далее повсеместно.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 4 "Меню настроек проекта в приложении Projucer»

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 5 "Файловый менеджер в приложении Projucer»

После создания и настройки проекта JUCE автоматически генерирует классы и методы в соответствии с выставленными на этапе настройки проекта параметрами. Данные конструкции сопровождаются комментариями с пояснением их функционала и советами по написанию кода внутри них.

* 1. Написание кода.

На данном этапе работа с приложением Projucer закончена. Дальнейшее редактирование кода будет осуществляться при помощи IDE - Microsoft Visual Studio 2022.

* + 1. Создание параметров и логики их работы.

Для того, чтобы последовательно конструировать и тестировать три модели обработки звука, включающиеся в итоговый продукт - необходимо реализовать придуманную на этапе проектирования концепцию переключения между таковыми. Также необходимо создать параметры и прописать логику их работы.

Важно отметить, что на такой ранней стадии разработки уже будет сконструирован графический интерфейс плагина. Это возможно благодаря компоненту фреймворка JUCE - «GenericAudioProcessorEditor», который автоматически выводит все параметры в виде слайдеров, комбинированных списков и переключателей. Для его использования необходимо следующим образом изменить «createEditor» в файле «PluginProcessor.cpp».

juce::AudioProcessorEditor\* DistortionBOXAudioProcessor::createEditor()

{

//return new DistortionBOXAudioProcessorEditor (\*this);

return new juce::GenericAudioProcessorEditor (\*this);

}

Листинг 1 «сreateEditor»

В файлах директории «Parameters» создаются параметры «Distortion Model», «Input», «Output» и «Mix». В дальнейшем данные параметры будут принимать в себя значения через «GenericAudioProcessorEditor».

#pragma once

#include <JuceHeader.h>

extern const juce::String disModelID;

extern const juce::String disModelName;

extern const juce::String inputID;

extern const juce::String inputName;

extern const juce::String outputID;

extern const juce::String outputName;

extern const juce::String mixID;

extern const juce::String mixName;

Листинг 2.1 «Parameters.h»

#include "Parameters.h"

const juce::String disModelID = "disModel";

const juce::String disModelName = "Distortion Model";

const juce::String inputID = "input";

const juce::String inputName = "Input";

const juce::String outputID = "output";

const juce::String outputName = "Output";

const juce::String mixID = "compMix";

const juce::String mixName = "Mix";

Листинг 2.2 «Parameters.cpp»

Для корректной работы плагина необходимо также сконструировать блок «process» внутри модуля «Distortion». Эту логику можно найти в готовом виде в папке JUCE Modules через обозреватель решений Microsoft Visual Studio. Фреймворк предоставляет большую библиотеку готовых решений, которые используются разработчиками для построения аналогичных конструкций в своих плагинах.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 6 "Скриншот части библиотеки модулей»

В приватной секции класса Distortion параметры были инициализированы, как «SmoothedValue». Это сделано во избежание появления нежелательных звуковых искажений при их изменении.

private:

juce::SmoothedValue<float> \_input;

juce::SmoothedValue<float> \_mix;

juce::SmoothedValue<float> \_output;

float \_sampleRate = 44100.0f;

DistortionModel \_model = DistortionModel::kHard;

Листинг 3 «Distortion private»

«SmoothedValue» предполагает обновление частоты дискретизации при изменении значения. Данный алгоритм осуществлен внутри блока «reset» модуля «Distortion».

template <typename SampleType>

void Distortion<SampleType>::reset()

{

if (\_sampleRate <= 0) return;

\_input.reset(\_sampleRate, 0.02);

\_input.setTargetValue(0.0);

\_mix.reset(\_sampleRate, 0.02);

\_mix.setTargetValue(1.0);

\_output.reset(\_sampleRate, 0.02);

\_output.setTargetValue(0.0);

}

Листинг 4 «Обновление частоты дискретизации»

Для создания функционала смены модели обработки необходимо создать класс «DistortionModel» в файле заголовков модуля.

enum class DistortionModel

{

kHard,

kSoft,

kSaturation

};

Листинг 5 «Получение параметров»

Далее необходимо прописать оператор switch, который будет получать модель обработки.

template <typename SampleType>

void Distortion<SampleType>::setDistortionModel(DistortionModel newModel)

{

switch (newModel)

{

case DistortionModel::kHard:

{

\_model = newModel;

break;

}

case DistortionModel::kSoft:

{

\_model = newModel;

break;

}

case DistortionModel::kSaturation:

{

\_model = newModel;

break;

}

}

}

Листинг 6 «Получение параметров»

Для того, чтобы принимать значения через «GenericAudioProcessorEditor» необходимо использовать стандартный класс фреймфорка JUCE -«AudioProcessorValueTreeState» для создания функции «createParameterLayout».

juce::AudioProcessorValueTreeState::ParameterLayout DistortionBOXAudioProcessor::createParameterLayout()

{

std::vector <std::unique\_ptr<juce::RangedAudioParameter>> params;

juce::StringArray disModels = { "Hard", "Soft", "Saturation" };

auto pDriveModel = std::make\_unique<juce::AudioParameterChoice>(disModelID, disModelName, disModels, 0);

auto pDrive = std::make\_unique<juce::AudioParameterFloat>(inputID, inputName, 0.0f, 24.0f, 0.0f);

auto pMix = std::make\_unique<juce::AudioParameterFloat>(mixID, mixName, 0.0f, 1.0f, 1.0f);

auto pOutput = std::make\_unique<juce::AudioParameterFloat>(outputID, outputName, -24.0f, 24.0f, 0.0f);

params.push\_back(std::move(pDriveModel));

params.push\_back(std::move(pDrive));

params.push\_back(std::move(pMix));

params.push\_back(std::move(pOutput));

return { params.begin(), params.end() };

}

Листинг 7 «Передача параметров»

Далее при помощи «parameterChanged» и «updateParameters» параметры передаются в модуль «Distortion», где применяются для осуществления переключения между моделями обработки, выставления уровня входящего сигнала, выходящего сигнала и уровня подмешивания обработанного сигнала к необработанному.

void DistortionBOXAudioProcessor::parameterChanged(const juce::String& parameterID, float newValue)

{

updateParameters();

}

void DistortionBOXAudioProcessor::updateParameters()

{

auto model = static\_cast<int>(\_treeState.getRawParameterValue(disModelID)->load());

switch (model)

{

case 0: \_distortionModule.setDistortionModel(Distortion<float>::DistortionModel::kHard); break;

case 1: \_distortionModule.setDistortionModel(Distortion<float>::DistortionModel::kSoft); break;

case 2: \_distortionModule.setDistortionModel(Distortion<float>::DistortionModel::kSaturation); break;

}

\_distortionModule.setDrive(\_treeState.getRawParameterValue(inputID)->load());

\_distortionModule.setMix(\_treeState.getRawParameterValue(mixID)->load());

\_distortionModule.setOutput(\_treeState.getRawParameterValue(outputID)->load());

}

Листинг 8 «Передача параметров в модуль Distortion»

* + 1. Программирование эффекта перегрузки hard clipping.

Эффекты, запланированные для реализации в рамках настоящего исследования, представляют собой намеренное искажение входного сигнала для получения более жесткого и насыщенного звучания. Реализовать технически данную обработку можно по-разному, но принцип работы будет в любом случае сводиться к ограничению амплитуды сигнала.

Для наибольшей наглядности за исходный звуковой сигнал будет взята стандартная синусоида. Она мягкая, ровная, естественно достигающая верхнего и идущая на спад к нижнему экстремуму, после чего вновь возрастающая.

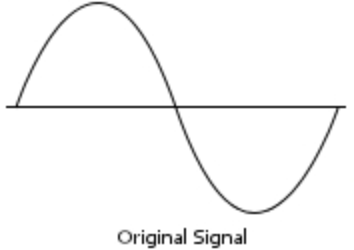


Рисунок 7 «Синусоида»

Эффект перегрузки «hard clipping» задает границу (как верхнюю, так и нижнюю), за которую сигнал никаким образом не может выйти, из-за чего пики волны жестко срезаются, она уплощается и становится более «острой», что и провоцирует искажение звука.

Изображение выглядит как линия, диаграмма, дизайн

Автоматически созданное описание

Рисунок 8 «Hard Clipping»

Функционал данного алгоритма перегрузки звукового сигнала реализован в блоке «processHardClipper». Первым делом амплитуда сигнала намеренно повышается, после чего сигнал проходит через тот самый «clipping» с потолком в 0.99.

if (std::abs(wetSignal) > 0.99)

{

wetSignal \*= 0.99 / std::abs(wetSignal);

}

Листинг 9 «Алгоритм hard clipping»

Логика подмешивания обработанного сигнала к необработанному с помощью ручки «Mix» выглядит следующим образом:

auto mix = (1.0 - \_mix.getNextValue()) \*

inputSample + wetSignal \* \_mix.getNextValue();

Листинг 10 «Mix»

Благодаря данному выражению при параметре «mix», равном одному - значение «wetSignal», означающее обработанный сигнал также будет равно одному, а значение «inputSample», означающее необработанный сигнал - будет равно нулю. При повороте ручки Mix значение «\_mix.getNextValue()» будет меняться соответственно, изменяя соотношение значений «wetSignal» и «inputSample». Эта логика далее применяется повсеместно.

* + 1. Программирование эффекта перегрузки soft clipping.

Текст данного подпараграфа находится в разработке.

* + 1. Программирование эффекта сатурации.

Текст данного подпараграфа находится в разработке

* 1. Тестирование итогового продукта.

Текст данного параграфа находится в разработке

# **СПИСОК ЛИТЕРАТУРЫ**

1. Унгер, А. Ю. Паттерны проектирования на C++ : учебное пособие / А. Ю. Унгер. — Москва : РТУ МИРЭА, 2023. — 74 с. — ISBN 978-5-7339-1753-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/368645 (дата обращения: 21.05.2024). — Режим доступа: для авториз. пользователей.
2. Андерсен, А. В. Современные музыкально-компьютерные технологии : учебное пособие / А. В. Андерсен, Г. П. Овсянкина, Р. Г. Шитикова. — 4-е, стер. — Санкт-Петербург : Планета музыки, 2021. — 224 с. — ISBN 978-5-8114-7389-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/160198 (дата обращения: 21.05.2024). — Режим доступа: для авториз. пользователей.
3. Зубец, А. И. Основы музыкальных технологий: компьютерная аранжировка и оркестровка, электронная музыка : учебное пособие / А. И. Зубец. — Санкт-Петербург : Планета музыки, 2024. — 332 с. — ISBN 978-5-507-48903-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/383063 (дата обращения: 21.05.2024). — Режим доступа: для авториз. пользователей.
4. Сутягин, Н. Р. Принципы и инструменты разработки аудио плагинов / Н. Р. Сутягин // РАЗВИТИЕ СОВРЕМЕННОЙ науки и ОБРАЗОВАНИЯ: АКТУАЛЬНЫЕ ВОПРОСЫ, ДОСТИЖЕНИЯ и ИННОВАЦИИ : сборник статей III Международной научно-практической конференции, Пенза, 17 июня 2022 года. – Пенза: Наука и Просвещение (ИП Гуляев Г.Ю.), 2022. — С. 12-14. — EDN MXTQOH.
5. Прата Стивен. Язык программирования C++. Лекции и упражнения. 6-е изд.: пер. с англ. / Стивен Прата. – Москва : Вильямс, 2020. – 928 с.: ил. ISBN 978-5-907114-14-2.
6. Магазинникова, А. Л. Основы цифровой обработки сигналов / А. Л. Магазинникова. — 5-е изд., стер. — Санкт-Петербург : Лань, 2024. — 132 с. — ISBN 978-5-507-48636-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/359951 (дата обращения: 21.05.2024). — Режим доступа: для авториз. пользователей.
7. Русакова, З. Н. Структуры данных в C++ : учебное пособие / З. Н. Русакова, И. В. Рудаков. — Москва : МГТУ им. Баумана, 2020. — 158 с. — ISBN 978-5-7038-5256-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/205832 (дата обращения: 21.05.2024). — Режим доступа: для авториз. пользователей.
8. Конюшик, Н. А. Плагин для цифровой звуковой рабочей станции и его эргономическое обеспечение / Н. А. Конюшик // Электронные системы и технологии : Сборник материалов 58-й научной конференции аспирантов, магистрантов и студентов БГУИР, Минск, 18–22 апреля 2022 года. – Минск: Белорусский государственный университет информатики и радиоэлектроники, 2022. – С. 716-718. – EDN IOLNQL.
9. Kasimkhodzhaeva, A. A. MODERN SOUND ENGINEERING - ACHIEVEMENTS AND CHALLENGES / A. A. Kasimkhodzhaeva // Проблемы современной науки и образования. — 2022. — № 9. — С. 94-98. — ISSN 2304-2338. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/journal/issue/333350 (дата обращения: 21.05.2024). — Режим доступа: для авториз. пользователей.
10. Гусаров, Д. Н. Программная реализация средства для генерации и обработки звуковых сигналов / Д. Н. Гусаров, О. В. Куликова, Г. С. Домабян // Интеллектуальные информационные технологии и математическое моделирование : Труды Международной научной конференции, пос. Дивноморское, Краснодарский край, 26–29 августа 2022 года / Под редакцией В.В. Долгова. – Ростов-на-Дону: Донской государственный технический университет, 2022. – С. 195-199. – EDN XBTNGS.
11. DAW: история создания и развития // Pop-Music.ru : сайт. — URL: https://pop-music.ru/articles/daw-istoriya-sozdaniya-i-razvitiya/ (Дата обращения 21.05.2024).
12. С++ // Wikipedia : сайт. — URL: https://ru.wikipedia.org/wiki/C%2B%2B. (Дата обращения 21.05.2024).
13. DAW // Wikipedia : сайт. — URL: https://ru.wikipedia.org/wiki/Цифровая\_звуковая\_рабочая\_станция (Дата обращения 21.05.2024).
14. Virtual Studio Technology // Wikipedia : сайт. — URL: https://en.wikipedia.org/wiki/Virtual\_Studio\_Technology (Дата обращения 21.05.2024).
15. Juce // Wikipedia : сайт. — URL: https://ru.wikipedia.org/wiki/Juce. (Дата обращения 21.05.2024).
16. Master Branch of JUCE documentation // juce.com : сайт. — URL: https://docs.juce.com/master/index.html (Дата обращения 21.05.2024).
17. JUCE — Кроссплатформенный C++ фреймворк для разработки приложений с пользовательским интерфейсом // Хабр : сайт. — URL: https://habr.com/ru/articles/209956/ (Дата обращения 21.05.22024).
18. JUCE forum // juce.com : сайт. — URL: https://forum.juce.com (Дата обращения 21.05.2024).
19. JUCE GitHub repository // github.com : сайт. — URL: https://github.com/juce-framework/JUCE (Дата обращения 21.05.2024).
20. How I made my own VST audio plugin (Dyst JSFX & JUCE Code Walkthrough) // YouTube : сайт. — URL: https://www.youtube.com/watch?v=ebLoEOWpgBo (Дата обращения 21.05.2024).
21. Audio Processing Tutorial: How To Create an AWESOME Distortion VST/AU Plugin In C++ (JUCE Framework) // YouTube : сайт. — URL: https://www.youtube.com/watch?v=iNCR5flSuDs&t=1556s (Дата обращения 21.05.2024).
22. JUCE 6 (2021) – playlist // YouTube : сайт. — URL: https://www.youtube.com/playlist?list=PLLgJJsrdwhPyNsICl0\_gSGF7owIow\_cfA (Дата обращения 21.05.2024).

1. Сокращение от Digital Sound Processor. [↑](#footnote-ref-1)