МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
**«Сибирский государственный университет науки и технологий
имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

КУРСОВАЯ РАБОТА

по дисциплине *Конструирование программного обеспечения*

на тему: Разработка аудио редактора для наложения эффектов и фильтров

Выполнил(а) студент(ка) группы БПИ 15–01
Очной формы обучения
Мельников В.Е.
(Ф.И.О.)

Руководитель:
к.т.н., доцент, Зотин А. Г.
(ученая степень, ученое звание, Ф.И.О.)

Дата сдачи: «_____»_____201__ г.

Дата защиты: «____»_____201__ г.

Оценка:_____

_____
(подпись руководителя)

Красноярск 2018 г.

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

ЗАДАНИЕ

на курсовую работу по дисциплине *Конструирование программного обеспечения*

студенту Мельникову Владимиру Евгеньевичу

Группа *БПИ 15–01* Форма обучения *Очная*

1. Тема работы (проекта): Разработка аудио редактора для наложения эффектов и фильтров

2. Срок сдачи студентом работы *28.12.2018*

3. Перечень вопросов, подлежащих разработке при написании теоретической части:

4. Перечень вопросов, подлежащих разработке при написании практической части:

7. Дата выдачи задания: *04.09.2018*

Руководитель *Зотин А. Г., доцент кафедры ИВТ*

(Подпись)

Задание принял к исполнению (дата) _____

(подпись студента)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность. К преобразованию звука прибегают в основном с целью изменения каких-то характеристик звука. На основе этих преобразований реализуются звуковые эффекты. Целью обработки звука является придание существующему звуку каких-то новых качеств или устранение нежелательных. Звуковые эффекты относятся к тем преобразованиям звука, которые придают звучанию новые формы или полностью изменяют звуковую информацию.

Аппаратную реализацию звуковые эффекты находят в цифровых сигнальных процессорах (*DSP*). Любой *MIDI*-синтезатор имеет встроенный эффект-процессор той или иной сложности (эффект-процессор представляет собой один или несколько *DSP*). Сложные эффект-процессоры "умеют" накладывать на звуковой сигнал сразу несколько различных эффектов, причем, отдельно для каждого канала, позволяя регулировать параметры эффектов в режиме реального времени. Однако стоимость таких эффект-процессоров чрезвычайно высока (как и стоимость любого другого высокопроизводительного микропроцессора), поэтому профессиональные *DSP* устанавливаются только на качественной музыкальной аппаратуре. На более или менее дешевых звуковых платах часто устанавливается *DSP* с упрощенным набором возможностей: наложение одного или нескольких эффектов на все каналы одновременно.

Так же обработать звук можно и программным способом. Существует множество различных звуковых редакторов, позволяющих делать со звуком значительно более сложные вещи, чем это позволяют делать даже самые сложные эффект-процессоры. Кроме того, эффект-процессоры часто эмулируются в виртуальных *WT*-синтезаторах, а также находят программную реализацию в специальных программах для обработки звука в режиме реального времени.

В настоящее время аудио эффекты используются при записи музыкальных композиций, подготовке фонограмм для радио, теле и интернет-вещания,

озвучивании фильмов и компьютерных игр, реставрации старых фонограмм (предварительно оцифрованных), акустического анализа речи.

Применение аудио редактора позволит:

- отобразить звуковой сигнал;
- записывать и воспроизводить аудио;
- накладывать различные эффекты и фильтры на звук;
- анализировать звук;

Таким образом, создание аудио редактора для наложения эффектов и фильтров является актуальной задачей.

Цель и задачи. Целью курсовой работы является разработка аудио редактора, позволяющего накладывать эффекты и фильтры на аудио.

Для достижения поставленной цели необходимо решить следующие задачи:

- выполнить анализ процесса наложения эффектов;
- осуществить обзор аудио редакторов, позволяющих накладывать фильтры и эффекты;
- спроектировать структуру программного продукта;
- разработать и реализовать модуль защиты аудио редактора;
- осуществить программную реализацию приложения;
- разработать набор плагинов, реализующих обработку аудиоданных;
- провести тестирование разработанного приложения.

Структура работы. Курсовая работа состоит из введения, двух глав, заключения и списка использованных источников из 12 наименований. Изложена на 41 странице и содержит 21 рисунок и 2 таблицы.

В первой главе курсовой работы приводится анализ организации обработки аудио.

Во второй главе описана структура программного продукта. Представлены руководство программиста и руководство пользователя. Приводится тестирование программы.

В заключении подведены итоги работы по курсовой работе. Сделаны выводы по программе.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Аудио редакторы используются для записи музыкальных композиций, подготовки фонограмм для радио, теле и интернет-вещания, озвучивания фильмов и компьютерных игр, реставрации старых фонограмм (предварительно оцифрованных), акустического анализа речи.

1.1 Анализ задачи использования эффектов для обработки аудио

Возможности современных программ и компьютеров, а также их относительная доступность по цене позволяют выполнять профессиональную обработку звукового материала не только на специализированной звуковой рабочей станции в студии звукозаписи, но и на персональном компьютере в своей домашней студии.

Чтобы обработать звук, для начала нужно получить аудиоданные, которые представляют из себя последовательность байт. Чтобы получить их используются декодеры. Однако, здесь важно учитывать особенности структуры аудиофайла. Так, например, при использовании формата *wav,* сначала в файле записываются заголовки, отображающие полную информацию об аудиофайле, и лишь потом идут аудиоданные, при чем в одном четырех байтном числе два байта используются для указания амплитуды левого канала, и два – для правого, а при использовании формата *mp3*, все данные делятся на фреймы, в которых первые 32 байта используются для отображения информации о нем и все остальные являются аудиоданными.

Кроме того, следует учесть характеристики аудиофайла для того, чтобы иметь возможность вычислять длительность звучания, что позволит обработать не весь трек, а лишь заданный промежуток.

Для того, чтобы наложить какой-нибудь эффект, или же применить аудио фильтр используется таблица свертки. Свертка – это последовательный процесс, заключающийся в сложении N точек входной функции, умноженных на коэффициенты (таблицу свертки), для получения одной точки результирующей функции. Для дискретных сигналов x[n] и y[n], состоящих из конечного числа отсчетов, свертка обозначается следующим образом:

$z[n] = x[n] \times y[n]$, где z[n] – результирующий сигнал, а y[n] – ядро свертки (таблица свертки).

Когда речь идет о свертке двух конкретных сигналов, состоящих из M и N отсчетов соответственно, для расчета i-го отчета результирующего сигнала используется формула:

$$z[i] = \sum_{k=0}^{N+M-2} x[k] \times y[i-k]$$

Данная операция проводится столько раз, сколько точек будет содержать результирующая функция. Комбинируя различные приемы построения таблиц можно добиваться очень разнообразных эффектов, 90% всех функций типичного музыкального редактора можно реализовать с помощью сверток. Сверткой запросто делаются следующие эффекты (в любой комбинации):

- накладываемые задержки
- любая частотная фильтрация
- вариации фаз сигналов

С помощью этого набора процессов легко делаются такие эффекты как: хорус, вокодеры, фланжеры, любая реверберация (даже самая естественная) и эхо, любые эквалайзеры и фильтрация, а также великое множество других эффектов. Стоит один раз тщательно рассчитать таблицу свертки, и любой из этих эффектов можно запросто выполнять чуть ли не в реальном времени – так, например, реализовано большое число сложных эффектов в популярном редакторе *Audacity* [7].

1.2 Обзор основных аудио редакторов

Существует большое количество различных аудио редакторов.

Audacity – один из самых известных бесплатных аудио редакторов. Приложение завоевало популярность благодаря своему функционалу. В программе доступны инструменты для звукозаписи, пакет стандартных и «усиливающих» эффектов, режим удаления щелчков, поддержка видеофайлов и объектов в RAW-формате.

Audacity – единственный бесплатный редактор, позволяющий «из коробки» работать с мультитрековыми дорожками (студийными заготовками музыкальных произведений).

Минус программы – не совсем удобный интерфейс. Например, эффекты добавляются через огромное всплывающее меню [1].

Возможности программы:

- запись звука с микрофона;
- монтаж и редактирование аудиозаписей − любое изменение исходного звукового фрагмента: изменение длины, копирование, вырезание или вставка фрагмента, создание звуковой композиции из нескольких звуковых файлов;
- редактирование характеристик звуковых сигналов − изменение громкости, темпа, высоты звучания, добавление эффектов;
- наложение различных эффектов и фильтров;
- сохранение созданной звуковой композиции в отдельный звуковой файл.

Adobe Audition – Использование *Adobe Audition* позволяет получить широкий спектр возможностей полноценного звукового редактора, который является прекрасным инструментом для работы с аудио. Звуковая станция дает возможность осуществлять звукозапись в *Adobe Audition* в мультиканальном ре-

жиме. Источниками сигнала для записи может быть практически любое устройство с аудиовыходом: электрогитара, микрофон, плеер и т.д [1].

Возможности программы:

- работа со всеми современными аудио форматами;
- анализ спектра звука;
- измерение искажений (нелинейных, фазовых и др.);
- сведение проекта в режиме «мультитрек» с параллельным редактированием каждого отдельного канала;
- применение многочисленных эффектов (реверберация, эхо и другие);
- регулировка фазы, тона и скорости воспроизведения;
- *3D Stereo Sound*;
- работа с исправлением и удалением дефектов (щелчки, шумоподавление и др.);
- конвертирование аудио.

Free Audio Editor – бесплатный для некоммерческого использования редактор файлов звукозаписи для платформы Windows, ориентированный на работу с двумя дорожками [1].

Возможности программы:

- Создание, импорт и экспорт файлов звукозаписи.
- Отображение файлов звукозаписи в виде формы волны или в виде спектра с возможностью масштабирования.
- Запись (с последующим воспроизведением) с любого доступного источника звука.
- Встроенная поддержка форматов: *MP3, WMA, WAV* и *OGG*.
- Редактирование и монтаж:
    - Усиление амплитуды звука;
    - наложение эффекта «задержка»;
    - частоторегуляция (темброблок);

- наложение эффектов «нарастание» и «затухание»;
- наложение эффекта «фланжер»;
- зеркальное отражение формы волны;
- нормализация звука;
- обратное воспроизведение;
- вставка тишины;
- наложение эффекта «протяжка»;
- наложение эффекта «вибрато»;
- наложение эффекта «эхо»;
- Применение различных фильтров.

- Дополнительные возможности:
  - загрузка видео с *YouTube* с последующим импортом дорожки звукозаписи;
  - поставляемое в комплекте приложение *Import Audio from Video* также конвертирует звукозапись в форматы: *wav*, *mp3*, *wma*, *ogg*, *aac*, *m4a*, *FLAC*.

Бесплатная версия *Free Audio Editor* сохраняет результаты работы только в формат *.wav* [1].

Рассмотреть существующего ПО показал, что у аудио редактора для наложения эффектов и фильтров должны быть различные аудио эффекты, кроме того, аудио редактор должен позволять накладывать фильтры, и при этом предоставлять возможность работать не со всем треком, а только с его частью.

1.3 Обзор технологий

Разрабатываемая система должна обладать графическим пользовательским интерфейсом, а также иметь возможность расширения встроенного функционала за счет использования плагинов, которые реализуются как динамиче-

ски подключаемые библиотеки. В таблице 1.2 представлено сравнение самых популярным средств разработки, позволяющих реализовать пользовательские приложения на различных языках программирования, а также информация о том, поддерживается ли ими динамические библиотеки, написанные на языке C\C++.

Таблица 1.2

Язык программирования	**Библиотека графического интерфейса**	**Поддержка DLL, написанных на C\C++**
C++	Qt	Поддерживаются
C++	VCL	Поддерживаются
Delphi	VCL	Поддерживаются
C#	WPF	Поддерживаются
Java	Swing	Не поддерживаются

Для разработки приложения был выбран язык программирования C# и система для построения клиентских приложений *WPF*, так как в отличие от устаревшей технологии *Windows Forms, WPF* включает новую модель построения пользовательских приложений [2].

Для написания системы плагинов был так же выбран язык C++, так как он позволяет писать плагины в виде библиотек.

Для аудио обработки существуют такие библиотеки, как *BASS, NAudio*. Библиотека *BASS* позволяет работать с потоками, MOD файлами, MO3 файлами записывать аудио, конвертировать аудио. Несмотря на то, что библиотека *NAudio* предоставляет возможности работы с меньшим количеством аудио форматов, в остальном она предоставляет те же функции, что и библиотека *BASS*, а кроме того, по сравнению с библиотекой *BASS NAudio* имеет более удобный *API*, поэтому было принято решение для обработки звука использовать библиотеку *NAudio*.

Для защиты приложения от нелегального копирования существуют такие методы, как защита с использованием файла ключа, с использованием аппаратного ключа, с использованием интернет-активации. Подход с использованием аппаратного ключа выигрывает с точки зрения надежности, однако он сложен в

реализации, а приложение с таким ключом гораздо сложнее распространять. Подход с использованием интернет-активации обладает множеством преимуществ, например отсутствие физических носителей, что означает как отсутствие проблем с пересылкой электронных ключей и дисков, так и невозможность их механических поломок, относительная дешевизна внедрения и поддержки этого метода защиты и, конечно, возможность широкой цифровой дистрибуции и "мгновенной" покупки ПО из любой точки Земли. Однако данный метод сложен в реализации. Подход с использованием файла лицензии самый легкий в реализации, но является самым ненадежным с точки зрения защищенности приложения. Таким образом, для реализации модуля защиты был выбран подход с использованием файла лицензии, из-за простоты реализации.

Работа плагинов будет осуществляться с использованием механизма потоков для того, чтобы отдельные задачи могли выполняться параллельно друг с другом.

1.4 Обзор методов и алгоритмов

Плагины, которые требуется реализовать в программе условно можно разделить на две категории. К первой категории относятся плагины, реализующие аудио фильтры, а ко второй категории относятся плагины, реализующие аудио эффекты.

Плагины, относящиеся к первой категории используются для ограничения или изменения спектра звукового сигнала в каком-то определенном частотном диапазоне с целью избавления от нежелательных шумов или помех, подавления определенных частотных полос, и т.д.

Фильтры характеризуются с помощью амплитудно-частотной характеристики (АЧХ). Эта характеристика представляет собой график зависимости коэффициента передачи K(f) (амплитуды) от частоты f. То есть на таком графике можно увидеть, в какой полосе частот сигнал будет передаваться без измене-

ний, и в какой полосе частот сигнал будет ослаблен или не пропущен совсем [6].

В данной работе рассматривается фильтр Баттерворта. АЧХ фильтра Баттерворта максимально гладкая на частотах полосы пропускания и снижается практически до нуля на частотах полосы подавления. При отображении частотного отклика фильтра Баттерворта на логарифмической АФЧХ, амплитуда снижается к минус бесконечности на частотах полосы подавления. В случае фильтра первого порядка АЧХ затухает со скоростью -6 децибел на октаву (-20 децибел на декаду) (на самом деле все фильтры первого порядка независимо от типа идентичны и имеют одинаковый частотный отклик). Для фильтра Баттерворта второго порядка АЧХ затухает на -12 дБ на октаву, для фильтра третьего порядка на -18 дБ и так далее. АЧХ фильтра Баттерворта − монотонно убывающая функция частоты [7].

Фильтр Баттерворта − единственный из фильтров, сохраняющий форму АЧХ для более высоких порядков (за исключением более крутого спада характеристики на полосе подавления) тогда как многие другие разновидности фильтров (фильтр Бесселя, фильтр Чебышёва, эллиптический фильтр) имеют различные формы АЧХ при различных порядках.

Амплитудно-частотная характеристика G(ω) фильтра Баттерворта n-го порядка может быть получена из передаточной функции H(s):

$G^2(\omega) = |H(j\omega)|^2 = \frac{G_0^2}{1+\left(\frac{\omega}{\omega_c}\right)^{2n}}$, где: n − порядок фильтра, ω_c − частота среза (частота на которой амплитуда равна -3 дБ), G_0 − коэффициент усиления по постоянной составляющей (усиление на нулевой частоте) [8].

График фильтра Баттерворта представлен на рисунке 1.1.

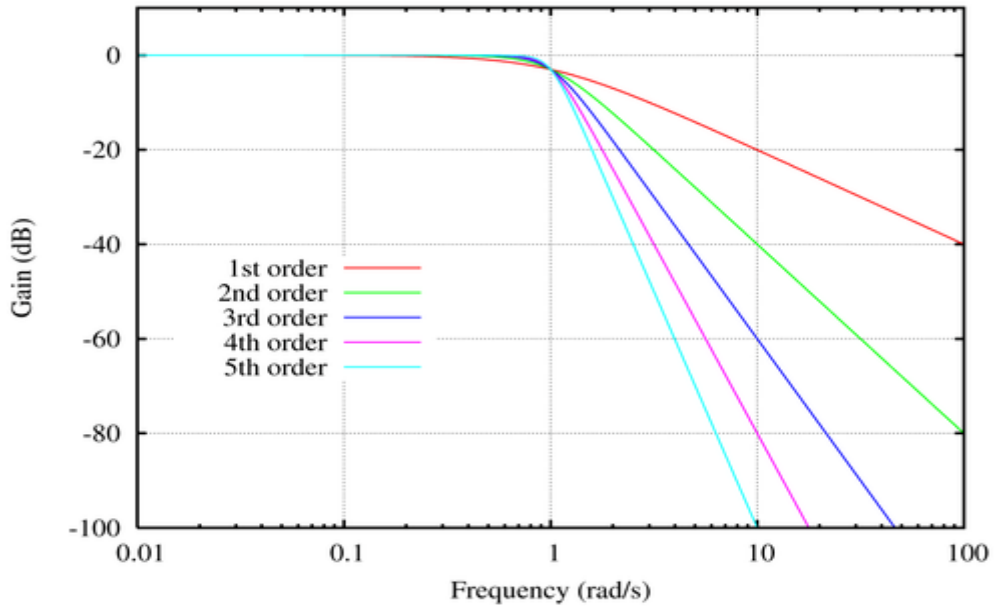

Рисунок 1.1. – фильтр Баттерворта

Обработка аудио происходит следующим образом: нахождение денормированной частоты среза, затем вычисление коэффициентов таблицы свертки, и, наконец, вычисление значения семпла.

Денормирование частоты среза выполняется по формуле:

$$\omega_p = \tan(\frac{\pi \times \omega}{F_c})$$

Формула вычисления значения семпла выполняется по формуле:

$$y(n) = b_0 \times x(n) + b_1 \times x(n-1) + b_2 \times x(n-2) - a_1 \times y(n-1) - a_2 \times y(n-2)$$

Вычисление значений коэффициентов таблицы свертки зависит от типа фильтра. При фильтре низких частот коэффициенты таблицы свертки вычисляются по формулам:

$$a_1 = -2 + 2 \times \omega^2$$

$$a_2 = 1 - \sqrt{2} \times \omega + \omega^2$$

$$b_1 = \omega^2$$

$$b_2 = 2 \times \omega^2$$

$$b_3 = \omega^2$$

При фильтре высоких частот:

$$a_1 = -2 + 2 \times \omega^2$$

$$a_2 = 1 - \sqrt{2} \times \omega + \omega^2$$

$$b_1 = 1$$

$$b_2 = -2$$

$$b_3 = 1$$

При полосном фильтре:

$$\omega_0 = \omega_1 \times \omega_2$$

$$\omega_d = \omega_2 - \omega_1$$

$$a_1 = 2 - 2 \times \omega_0$$

$$a_2 = -1 + \omega_d - \omega_0$$

$$b_1 = -\omega_d$$

$$b_2 = 0$$

$$b_3 = \omega_d$$

Ко второй категории относятся плагины, реализующие аудио эффекты. Они используются для придания существующему звуку каких-то новых качеств или устранения нежелательных. Звуковые эффекты относятся к тем преобразованиям звука, которые придают звучанию новые формы или полностью изменяют звуковую информацию.

В данной работе рассматриваются три аудио эффекта, а именно: *echo*, *reverberation* и *distortion*.

Создание эффекта "эхо" построено на использовании метода задержки. Фактически для получения эха необходимо на оригинальный входной сигнал

наложить его задержанную во времени копию. Для того, чтобы человеческое ухо воспринимало вторую копию сигнала как повторение, а не как отзвук основного сигнала, необходимо время задержки установить равным примерно 50 мс. Кроме того, на основной сигнал можно наложить не одну его копию, а несколько, что позволит на выходе получить эффект многократного повторения звука (многоголосного эха). Чтобы эхо казалось затухающим, необходимо на исходный сигнал накладывать не просто задержанные копии сигнала, а и приглушенные по амплитуде [7]. Так как происходит обработка дискретных данных требуется вычислять, количество семплов, через которое требуется обрабатывать сигнал. Данное значение считается по формуле:

$$n = t \times \frac{F_c}{1000}$$

Где n – количество семплов, F_c – частота дискретизации, t – время задержки.

Схематично механизм создания эха можно представить, как показано на рисунке 1.2:

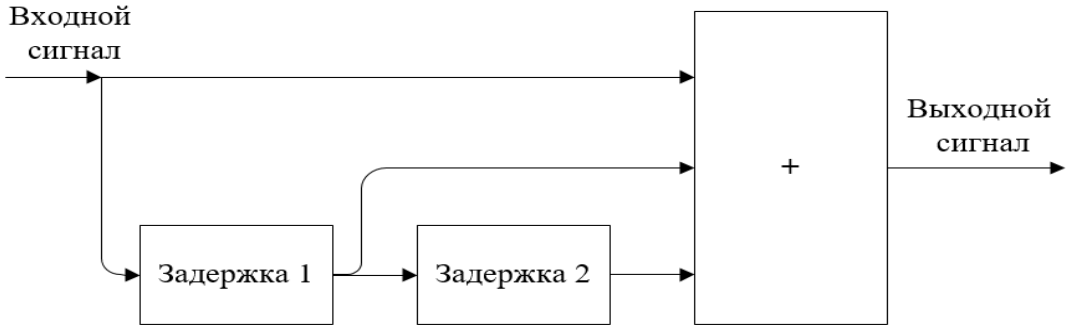

Рисунок 1.2. – механизм создания эхо

С использованием задержки можно добиться появления еще одного интересного эффекта − реверберации (от англ. "*reverberation*" − повторение, отражение). Эффект реверберации заключается в придании звучанию объемности, характерной для большого зала, где каждый звук порождает соответствующий, медленно угасающий отзвук. Таким образом, с помощью реверберации можно

"оживить", например, фонограмму, сделанную с заглушенном помещении. От эффекта "эхо" реверберация отличается тем, что на входной сигнал накладывается задержанная во времени не его копия, а выходной сигнал. Такой процесс происходит следующим образом. В первый момент времени входной сигнал проходит на выход без изменений. Затем, по истечении времени задержки, он снимается с выхода, его амплитуда умножается на какой-то коэффициент A (обычно имеющий значение меньше 1, что фактически приглушает сигнал) и суммируется со входным сигналом. И снова, по прошествии очередного промежутка времени задержки, уже смешанный сигнал снимается с выхода, снова перемножается на коэффициент A и в очередной раз суммируется с входным сигналом [7]. Так как происходит обработка дискретных данных требуется вычислять, количество семплов, через которое требуется обрабатывать сигнал. Данное значение считается по формуле:

$$n = t \times \frac{F_c}{1000}$$

Где n – количество семплов, F_c – частота дискретизации, t – время задержки.

Схематично механизм реверберации показан на рисунке 1.3:

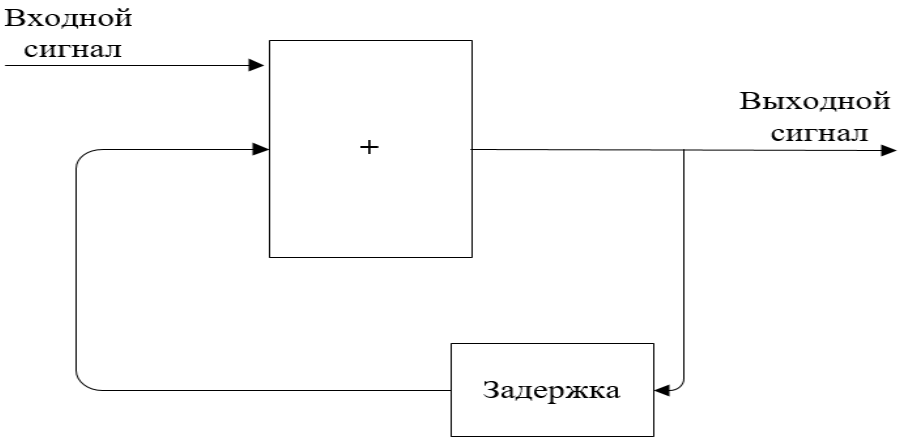

Рисунок 1.3 – механизм создания реверберации

Эффект дистошн (от англ. "distortion" − искажение) основывается на использовании амплитудной модуляции. Фактически это замена одних значений

амплитуд сигнала другими значениями. За счет переусиления, когда происходит срезание верхушек входного сигнала, можно получить, например, классический вариант гитары *heavy metal* (то есть сигналу придается скрежетание или своеобразная "хрипота"). Применение такого эффекта приводит к довольно резкому искажению входного сигнала (в зависимости от глубины модуляции), в результате чего сигнал становится похож на прямоугольный, и как следствие происходит расширение спектра сигнала. Классический механизм получения эффекта показан на рисунке 1.4:

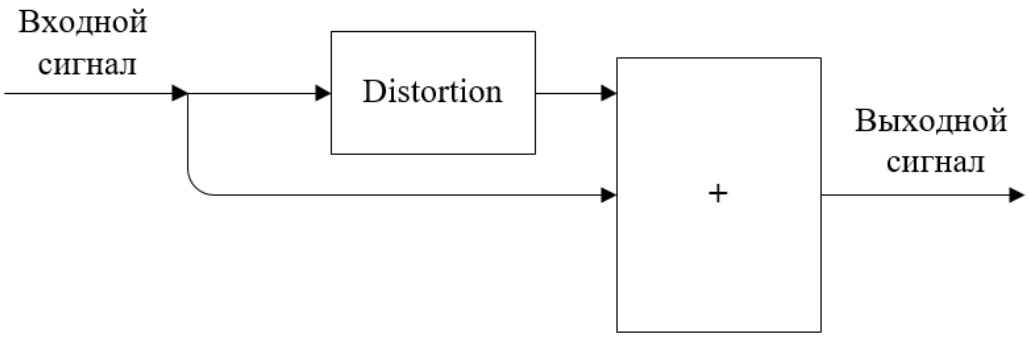

Рисунок 1.4 – механизм создания эффекта «*distortion*»

Входной сигнал смешивается с его копией, подвергнутой преобразованию в блоке *distortion*. Блок имеет два уровня сигнала: пороговый и верхний. Если амплитуда входящего в блок сигнала не превышает порогового уровня, то сигнал проходит на выход блока без изменений. Если же амплитуда сигнала выше порогового уровня, то блок усиливает такой сигнал до верхнего уровня [7]. Пример применения эффекта *distortion* к сигналу приведен на рисунке 1.5:

Рисунок 1.5 – применение эффекта «*distortion*» к синусоидальному сигналу

Для реализации файла лицензии используется алгоритм кодирования *Base64*. *Base64* – стандарт кодирования двоичных данных при помощи только 64 символов ASCII. Алфавит кодирования содержит текстово-цифровые латинские символы A-Z, a-z и 0-9 (62 знака) и 2 дополнительных символа, зависящих от системы реализации. Каждые 3 исходных байта кодируются 4 символами.

Для того, чтобы преобразовать данные в base64, первый байт помещается в самые старшие восемь бит 24-битного буфера, следующий — в средние восемь и третий – в младшие значащие восемь бит. Если кодируется менее, чем три байта, то соответствующие биты буфера устанавливаются в ноль. Далее каждые шесть бит буфера, начиная с самых старших, используются как индексы строки́ «ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/» и её символы, на которые указывают индексы, помещаются в выходную строку. Если кодируются только один или два байта, в результате получаются только первые два или три символа строки, а выходная строка дополняется двумя или одним символами «=». Это предотвращает добавление дополнительных битов к восстановленным данным. Процесс повторяется над оставшимися входными данными [8].

Например, цитата из "Левиафана" Томаса Гоббса:

Man is distinguished, not only by his reason, but by this singular passion from other animals, which is a lust of the mind, that by a perseverance of delight in the continued and indefatigable generation of knowledge, exceeds the short vehemence of any carnal pleasure.

перекодированная в base64, выглядит следующим образом:

TWFuIGlzIGRpc3Rpbmd1aXNoZWQsIG5vdCBvbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieSB0aGlzIHNpbmd1bGFyIHBhc3Npb24gZnJvbSBvdGhlciBhbmltYWxzLCB3aGljaCBpcyBhIGx1c3Qgb2YgdGhlIG1pbmQsIHRoYXQgYnkgYSBwZXJzZXZlcmFuY2Ugb2YgZGVsaWdodCBpbiB0aGUgY29udGludWVkIGFuZCBpbmRlZmF0aW

dhYmxlIGdlbmVyYXRpb24gb2Yga25vd2xlZGdlLCBleGNlZWRzIHRoZSBzaG9ydCB2ZWhlbWVuY2Ugb2YgYW55IGNhcm5hbCBwbGVhc3VyZS4=

Защищенность закодированной строки можно улучшить различными способами. Один из способов заключается в добавлении мусора в закодированную строку. Данный способ можно использовать в курсовой работе при кодировании файла лицензии.

Выводы по главе

В качестве прикладной задачи к разрабатываемой системе была выбрана задача обработки аудиоданных, в частности наложение эффектов и фильтров на аудио.

В качестве языка программирования был выбран C#, так он является статически типизированным и компилируемым. Среди двух рассмотренных библиотек для разработки графического пользовательского интерфейса была выбрана библиотека *WPF* из-за её гибкости при реализации сложного графического интерфейса. Для реализации считывания и записи аудио был выбран инструмент *NAudio*.

Были рассмотрены такие методы обработки звука как: фильтр Баттерворта и эффекты «*Echo*», «*Reverberation*» и «*Distortion*», однако, стоит отметить, что это далеко не все существующие на данный момент времени способы обработки звука.

К разрабатываемой системе предъявлено требование универсальности, т.е. она не должна быть ограничена рамками решаемой задачи прикладной области, поэтому все алгоритмы будут реализовываться в виде плагинов.

2 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

После выбора средств и методов разработки, приходит черед реализации программы, с которой взаимодействует непосредственно пользователь.

2.1 Структура программного продукта

Структура взаимодействия функциональных модулей программного продукта с плагинами представлена на рисунке 2.1.

Рисунок 2.1 – Структура взаимодействия функциональных модулей программного продукта

Детальное описание всех модулей и их составляющих частей представлена далее.

2.1.1 Головной модуль

Головной модуль – это сама программа, в которой работает пользователь. Он состоит из менеджера плагинов, загрузчика плагинов, менеджера задач, очереди задач, модуля базы данных, а также формы управления плагинами и формы управления задачами.

Загрузчик плагинов производит автоматический поиск плагинов в отдельном потоке, производит отсев плагинов, реализованных ненадлежащим образом. Он так же происходит сбор всей информации о плагинах Менеджер плагинов реализован как модель и используется при организации. Он получает всю информацию о плагинах от загрузчика плагинов.

Менеджер задач является посредником при взаимодействии с задачами. Менеджер задач осуществляет механизмы синхронизации между базой данных и программой, а также отдаёт задачи на выполнение в очередь задач. Менеджер задач реализован как модель и использует при организации MVC шаблона.

Очередь задач отвечает за выполнение задач. Очередь задач взаимодействует с менеджером задач и получает от него все задачи. Очередь задач управляет циклом событий и связывает события, генерируемые одними плагинами с обработчиками событий, загруженных из других плагинов.

Форма аудио редактора предоставляет графический интерфейс и является компонентом MVC шаблона, который отображает информацию об аудио и предоставляет компоненты для управления им.

Форма управления плагинами предоставляет также является компонентом MVC шаблона и предоставляет графический интерфейс, отображающий информацию о загруженных плагинах. Данная форма предоставляет возможность активировать или деактивировать загруженный плагин, заблокировать плагин, запрещая его загрузку, а также изменить директорию, из которой должна производиться загрузка плагинов.

2.1.2 Организация системы плагинов

Каждый плагин представляет собою динамически подключаемую библиотеку. Загрузчик плагинов при загрузке каждого плагина, ищет в директории с плагином конфигурационный файл плагина, имя которого задано в определённом формате: «название_плагина.propetries.json». В данном конфигурационном файле представлена такая информация о плагине, как название, имя автора, краткое описание плагина. Так же описывается пользовательский интерфейс,

который создается при выборе плагина. Формат данного файла представлен на рисунке 2.2. Конфигурационный файл описывается в формате JSON.

```
{
        "Name":"Distortion",
        "Description":"Плагин, отвечающий за эффект Distortion",
        "Author":"Melnikov V.E.",
        "Version":"1.0",
        "Components":[

                {
                        "Name":"SUpperLevel",
                        "Type":"Slider",
                        "Text":"Верхний уровень",
                        "Min":0,
                        "Max":255,
                        "Step":1
                },
                {
                        "Name":"SLineLevel",
                        "Type":"Slider",
                        "Text":"Пороговый уровень",
                        "Min":0,
                        "Max":255,
                        "Step":1
                }
        ]
}
```

Рисунок 2.2 – Пример конфигурационного файла плагина.

Все плагины имеют унифицированный протокол, предназначенный для упрощения работы с плагином. Согласно протоколу, взаимодействие с плагином осуществляется при помощи 2 функций. Описание данных функций представлено в таблице 2.2.

Таблица 2.1 – Описание функций взаимодействия программы с плагинами

Название функции	Описание
start	Реализует алгоритм обработки. Функция принимает 2 параметра, один из которых – целочисленный массив, содержащий аудиоданные, а другой – строка, содержащая параметры, необходимые для обработки данных.
initPlugin	Функция вызывается для инициализации экземпляра плагина. Данная функция вызывается при запуске плагина.

Описание плагинов предметной области будет представлено в следующем разделе. Схема, отображающая взаимодействие плагинов с основной программой представлена на рисунке 2.3

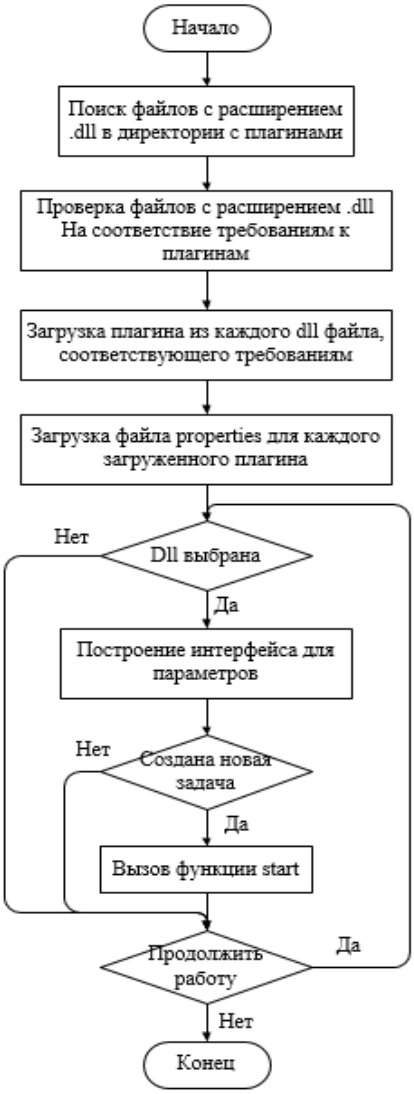

Рисунок 2.3 – Схема взаимодействия плагинов с основной программой

2.2 Алгоритмическая реализация

Для реализации задачи обработки аудиоданных были разработаны следующие плагины.

2.2.1 Плагины, реализующие фильтры

*З*адача фильтрации − изменить соотношение мощностей частот в звуке. Разные цифровые фильтры фильтруют сигналы совершенно по−разному, и надо хорошо понимать, какой фильтр где применять. Единственный неправильно примененный (не по назначению) фильтр, как правило, наносит звуку непоправимый ущерб. Малозаметный на глаз или при беглом прослушивании, но в дальнейшем − просто ощущаемый как смазанный или звенящий звук, и устранить его затем невозможно.

Плагин libButterworth

Данный плагин получает на вход два параметра. Первый – аудиоданные, представленные в виде целочисленного массива, второй – строка, состоящая из 3 параметров: тип фильтра (НЧ, ВЧ, полосовой), частота среза фильтра и временные рамки наложения фильтра.

Обобщенная схема алгоритма работы представлена на рисунке 2.4.

Рисунок 2.4 – Обобщенная схема алгоритма *libButterworth*

2.2.2 Плагины, реализующие эффекты

Плагин libEcho

Данный плагин получает на вход два параметра. Первый является массивом аудиоданных. Второй параметр является строкой, содержащей параметры. Она состоит из 3 параметров: времени задержки, коэффициента, который определяет «приглушенность» звука и временных рамок наложения эффекта.

Обобщенная схема алгоритма работы представлена на рисунке 2.5.

Рисунок 2.5 – Обобщенная схема алгоритма *libEcho*

Плагин libReverb

Данный плагин получает на вход два параметра. Первый является массивом аудиоданных. Второй параметр является строкой, содержащей параметры. Она состоит из 3 параметров: времени задержки, коэффициента, который определяет «приглушенность» звука и временных рамок наложения эффекта.

Обобщенная схема алгоритма работы представлена на рисунке 2.6.

Рисунок 2.6 – Обобщенная схема алгоритма *libReverb*

Плагин libDistortion

Данный плагин получает на вход два параметра. Первый является массивом аудиоданных. Второй параметр является строкой, содержащей параметры. Она состоит из 3 параметров: порогового уровня, верхнего уровня и временных рамок наложения эффекта.

Обобщенная схема алгоритма работы представлена на рисунке 2.7.

Рисунок 2.7 – Обобщенная схема алгоритма *libDistortion*

Модуль защиты

Для реализации модуля защиты был выбран подход с использованием файла лицензии.

Файл лицензии создается следующим образом: сначала формируются данные в виде строк, которые содержат в себе следующую информацию: кем выдана лицензия, кому выдана, когда выдана и до какого числа лицензия считается действительной. Структура файла выглядит следующим образом:

Выдано: «Славников В.А.»;

От: «Мельников В.Е.»;

Дата: «21012018»;

Годен до: «21012019»;

Затем используется алгоритм *Base64* для кодирования данных, после чего в закодированную строку на каждое пятое место вставляется случайный символ (строчная английская буква).

Обобщенная схема алгоритма работы модуля защиты представлена на рисунке 2.8.

Рисунок 2.8 – Обобщенная схема алгоритма модуля защиты

Декодирование состоит из 2-х этапов:

1. После считывания файла, из полученной строки удаляется каждый пятый символ;
2. Декодирование *Base64*.

2.3 Руководство программиста

Программный продукт разработан под операционную систему Windows. В качестве основного языка программирования, на котором были реализованы все компоненты системы, был выбран язык программирования C#. Для организации пользовательского графического интерфейса используется фреймворк *WPF*. Для реализации плагинов был выбран язык программирования C++.

Для выполнения возложенного функционала программой, необходимо чтобы система имела:

- процессор двухъядерный от 1.9 ГГц
- оперативную память от 2048 Мб;
- операционную систему Windows 7 и выше;
- клавиатуру и мышь

2.4 Краткое руководство пользователя

При запуске программы откроется основное окно (рисунок 2.9).

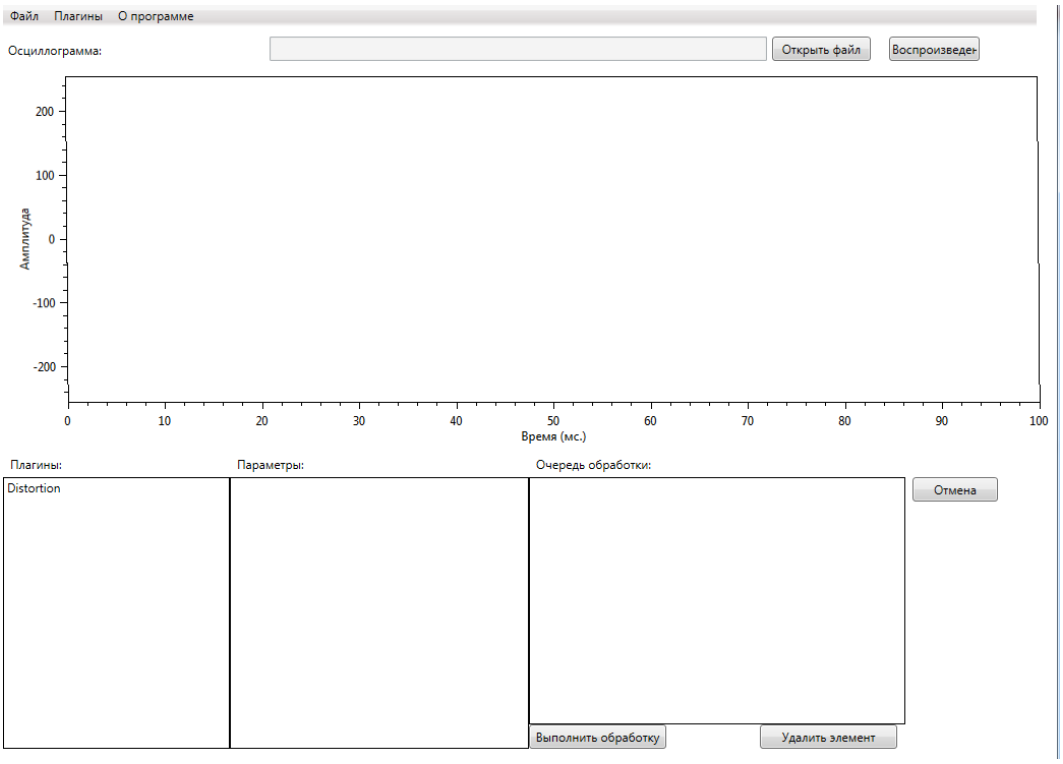

Рисунок 2.9 – главное окно программы

В левом нижнем углу окна находится список подключенных плагинов. Для того, чтобы загрузить плагины, или наоборот, отключить – требуется нажать на пункт главного меню окна «Плагины». После нажатия откроется окно со списком плагинов (рисунок 2.10).

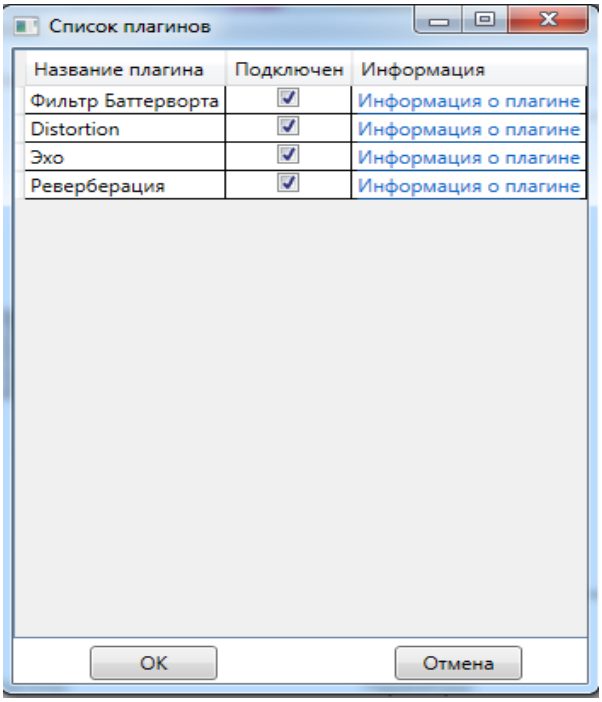

Рисунок 2.10 – окно со списком плагинов

Для того, чтобы загрузить или выгрузить плагины нужно:

1. выбрать загружаемые плагины, посредством отметки их в колонке «Включены»;
2. нажать на кнопку «ОК».

После произведения данных действий окно закроется, и список плагинов на главном окне обновится.

При нажатии на ссылку «Информация о плагине» откроется окно с информацией о плагине (Рисунок 2.11).

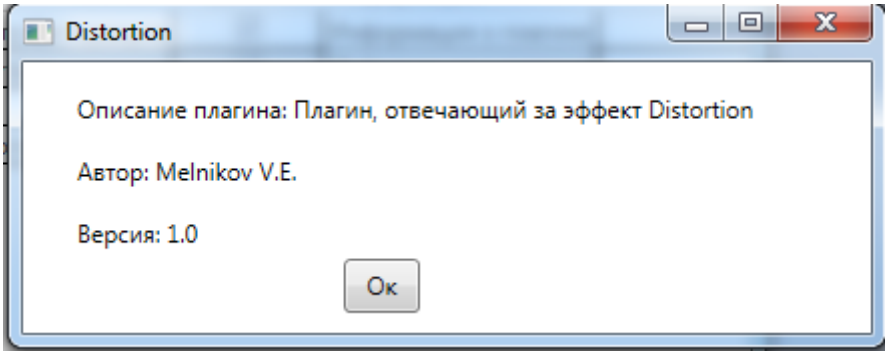

Рисунок 2.11 – окно с информацией о плагине

При нажатии на пункт меню «О программе» появится окно, содержащее информацию о программе, а также о лицензии (рисунок 2.12).

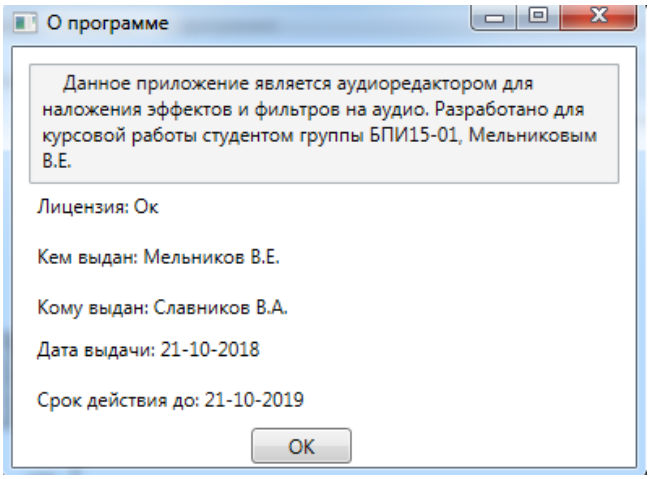

Рисунок 2.12 – окно с информацией о программе

В верхней части программы находятся кнопки «Загрузить аудио» и «Играть». При нажатии на кнопку «Загрузить аудио» появится окно выбора аудиофайла.

После выбора файла, он будет загружен и в главном окне будет выведена осциллограмма файла (рисунок 2.13).

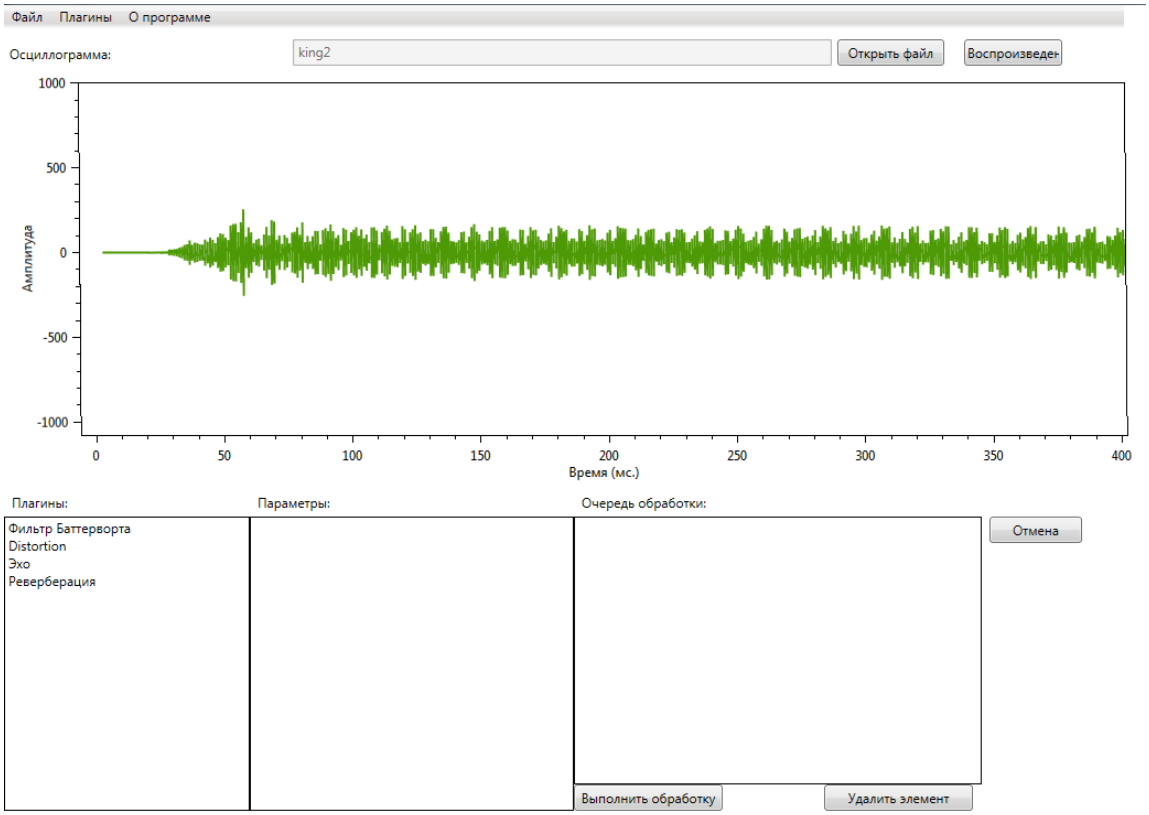

Рисунок 2.13 – главное окно после выбора аудиофайла

При нажатии кнопки «Воспроизведение» загруженный аудиофайл будет Воспроизведен. При повторном нажатии – остановлен.

При выборе какого-либо плагина в списке плагинов, будет выведены свойства данного плагина в список свойств (рисунок 2.14).

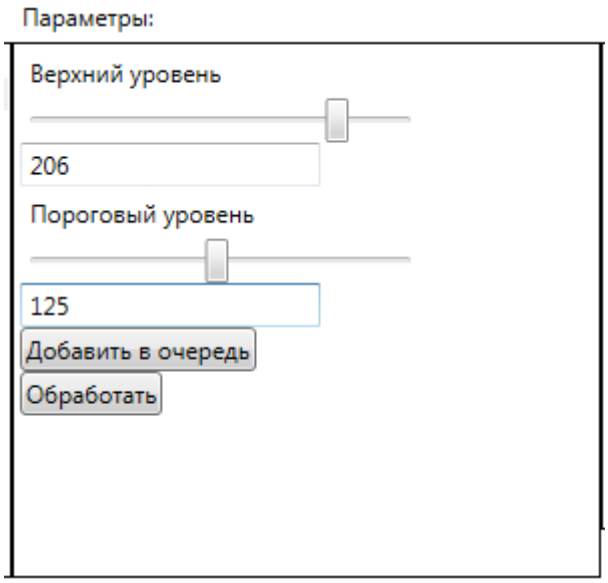

Рисунок 2.14 – свойства плагина

После задания свойств, при нажатии кнопки «Добавить в очередь» данный плагин с заданными свойствами добавляется в очередь на обработку, а при нажатии «Обработать» аудиофайл обрабатывается сразу с возможностью отмены обработки (посредством кнопки «Отмена»). Очередь обработки представлена на рисунке 2.15.

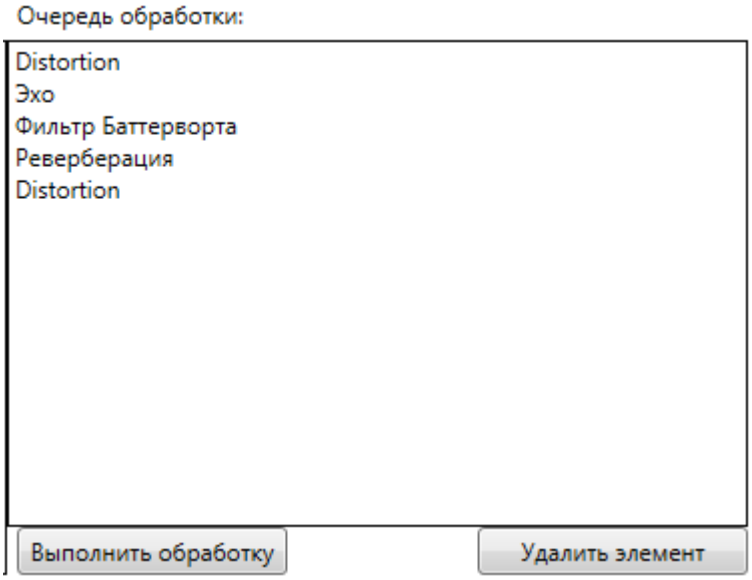

Рисунок 2.15 – очередь обработки

Для того, чтобы сохранить измененный аудиофайл – требуется нажать пункт «Сохранить» в выпадающем меню пункта «Файл» в главном меню.

Для закрытия аудио редактора требуется нажать пункт «Выход» в выпадающем меню пункта «Файл» в главном меню.

2.5 Тестирование

Тестирование программного продукта проводилось на всех этапах разработки, тестировалась как работа каждого плагина отдельна, работа основной программы, так и всей системы в целом. Плагины тестировались на правильную реализацию алгоритмов, правильное создание интерфейса (элементов для параметров), а также корректный вывод информации о плагине. Основная про-

грамма тестировалась на возможность пользователя добавлять задачи на выполнение. Общая система тестировалась на слаженную работу всех своих компонентов.

Результаты тестирования приведены в таблице 2.3.

Таблица 2.3 – результаты тестирования

№	Название	Цель	Объект	Значение	Результат
1	Создание задач	Определить, корректно ли создаётся задача	Основной модуль	Корректная работа	Задача создаётся корректно и связывает генерируемые события одного плагина с обработчиком другого.
2	Включение/Отключение плагинов	Определить, корректно ли включаются/отключаются плагины	Основной модуль	Корректная работа	Список плагинов обновляется, сохраняя лишь включенные плагины
3	Работа с фильтром	Определить, корректна ли работа накладываемого фильтра	Плагин *libButterworth*	Корректная работа	Плагин накладывается на аудио в соответствии с характеристикой фильтра
4	Работа с эффектом «Эхо»	Определить, корректна ли работа накладываемого эффекта	Плагин *libEcho*	Корректная работа	Плагин накладывается на аудио в соответствии с характеристикой эффекта
5	Работа с эффектом «Реверберация»	Определить, корректна ли работа накладываемого эффекта	Плагин *libReverb*	Корректная работа	Плагин накладывается на аудио в соответствии с характеристикой эффекта
6	Работа с эффектом «*Distortion*»	Определить, корректна ли работа накладываемого эффекта	Плагин *libDistortion*	Корректная работа	Плагин накладывается на аудио в соответствии с характеристикой эффекта
7	Работа модуля защиты	Определить, корректна ли проверка лицензии, а так же отсутствуют ли некоторые возможности в приложении с отсутствием лицензии	Модуль защиты	Корректная работа	При отсутствии лицензии приложение запускается с ограниченными возможностями

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы был разработан программный продукт, предназначенный для обработки аудио файлов.

В ходе выполнения курсовой работы были проанализированы способы обработки аудио и осуществлен обзор существующего программного обеспечения, такого как: *Free Audio Editor, Audacity, Adobe Audition*. Помимо этого, были выбраны технологии для реализации программного продукта. Для реализации основной программы и плагинов были выбраны язык C# и фреймворк *WPF*. Была рассмотрена организация процесса декодирования аудиоданных с помощью библиотеки *NAudio*.

Было разработано приложение по наложению аудио эффектов и фильтров. Кроме того, разработано 4 плагина на языке C++, реализующие фильтр Баттерворта, а также эффекты: эхо, реверберация и *distortion*. Разработан модуль защиты с использованием файла лицензии, закодированного с помощью алгоритма *Base64* и добавление мусора в закодированный файл.

Проведено тестирование разработанной системы методом «чёрный ящик». Программный продукт является законченным и готовым к использованию, однако, это не исключает возможность доработки и масштабирования, за счет разработки дополнительных плагинов, выполняющих новые функции.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Лучшие программы для работы со звуком (аудио конвертеры и редакторы) [Электронный ресурс] – URL: https://www.softhome.ru/article/luchshie−programmy−dlya−raboty−so−zvukom;
2. C# и .NET [Электронный ресурс] – URL: https://metanit.com/sharp/tutorial/11.1.php ;
3. JSON [Электронный ресурс] – URL: https://www.json.org/json−ru.html;
4. WPF и C# [Электронный ресурс] – URL: https://metanit.com/sharp/wpf/ ;
5. StackOverflow [Электронный ресурс] – URL: https://stackoverflow.com/ ;
6. AudioCoding [Электронный ресурс] – URL: https://audiocoding.ru ;
7. WebSound [Электронный ресурс]–URL: http://websound.ru/ ;
8. Wikipedia [Электронный ресурс]–URL: http://websound.ru/ ;
9. Wave File Format [Электронный ресурс] – URL: http://microsin.net/programming/pc/wav-format.html ;
10. Внутри MP3. А как оно всё устроено? [Электронный ресурс] – URL: https://habr.com/post/103635/ ;
11. Рисуем волну .wav-файла [Электронный ресурс] – URL: https://habr.com/post/113239/ ;
12. Теория звука. Что нужно знать о звуке, чтобы с ним работать. Опыт Яндекс.Музыки [Электронный ресурс] – URL: https://habr.com/company/yandex/blog/270765/ .

ПРИЛОЖЕНИЕ А. ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ ПРОГРАММНОГО ПРОДУКТА

Введение

В настоящее время аудио редакторы используются для записи музыкальных композиций, подготовки фонограмм для радио, теле и интернет–вещания, озвучивания фильмов и компьютерных игр, реставрации старых фонограмм (предварительно оцифрованных), акустического анализа речи. Использование аудио редакторов позволит записывать и редактировать звук не имея особой подготовки.

Основание для разработки

Разработка программного продукта ведется на основании учебного плана СибГУ им. М. Ф. Решетнева.

Назначение разработки

Разрабатываемый программный продукт предназначен для обработки аудио.

Требования к программе

Требования к функциональным характеристикам

Разрабатываемая программа должна:

- предоставлять графический интерфейс для наложения аудио эффектов и фильтров;
- отображать список подключенных плагинов с выводом информации о них (название, авторство, версия);
- настройка параметров работы алгоритмов обработки аудио через динамически формируемый приложением графический интерфейс;
- возможность добавлять задачи по обработке аудио;
- формировать осциллограмму обрабатываемого трека;

Ввод данных должен выполняться в программе пользователем, в виде текстовых, числовых данных. Вывод информации должен выполняться в виде аудио данных.

Требования к надежности

В программе должен быть обеспечен контроль над вводимыми данными. Возникновение каких−либо внутренних ошибок не должно приводить к утере данных. Программа должна полностью предусматривать ошибки, которые может совершить пользователь.

Условия эксплуатации

Приложение рассчитано на эксплуатацию пользователями имеющих опыт работы с компьютером.

Требования к составу и параметрам технических средств

Процессор с частотой не ниже 2.4 МГц.; оперативная память не меньше 1 Гб.; не менее 100 Мб свободного места на жестком диске; клавиатура, мышь.

Требования к информационной и программной совместимости

Операционная система Windows 7/8/10; .Net Framework 4.5.

Стадии и этапы разработки

- Анализ процесса наложения эффектов и фильтров.
- Обзор существующего программного обеспечения.
- Проектирование архитектуры приложения.
- Разработка системы плагинов.
- Разработка модуля защиты.
- Разработка алгоритмов эффектов и фильтров.
- Разработка пользовательского интерфейса.
- Программная реализация аудио редактора.
- Тестирование и отладка реализованного приложения.
- Оформление пояснительной записки.

ПРИЛОЖЕНИЕ Б. ПРИМЕРЫ ОТЧЕТОВ И ГРАФИКОВ

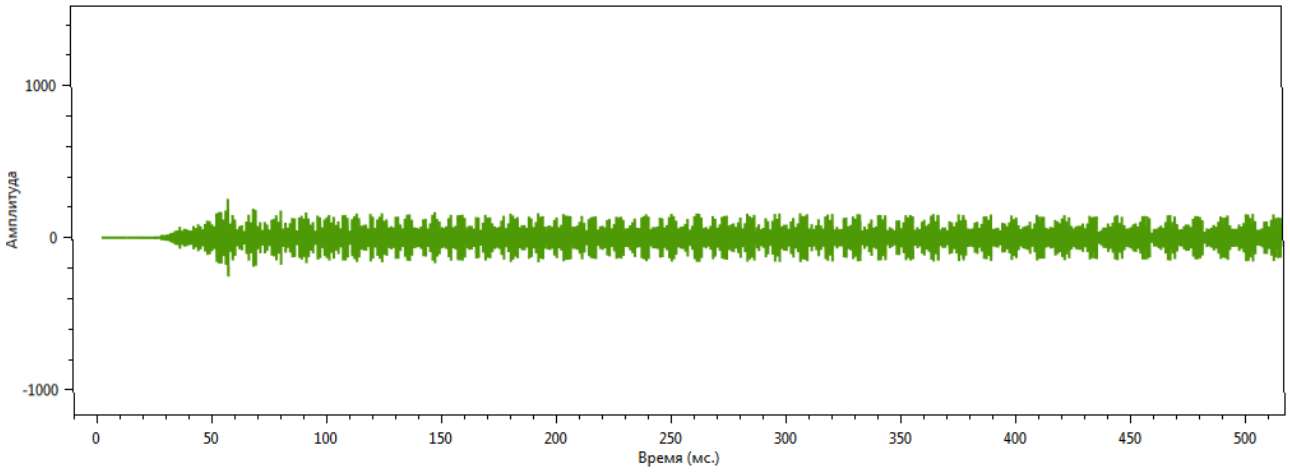

Рисунок Б1 – Осциллограмма аудио