СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc483227298)

[**1** ОБЗОР ЛИТЕРАТУРЫ 8](#_Toc483227299)

[**1.1** Цифровые аудио-станции и плагины 8](#_Toc483227300)

[**1.2** Octaver 10](#_Toc483227301)

[**1.3** Delay 13](#_Toc483227302)

[**1.4** Reverb 14](#_Toc483227303)

[**2** СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ 16](#_Toc483227304)

[**2.1** Структура программного модуля 16](#_Toc483227305)

[**2.2** Выбор программных средств 17](#_Toc483227306)

[**3** ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 21](#_Toc483227307)

[**3.2** Класс ShimmerAudioProcessor 22](#_Toc483227308)

[**3.3** Класс DelayLine 22](#_Toc483227309)

[**4** РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 24](#_Toc483227310)

[**5** ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ 25](#_Toc483227311)

[**6** РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 26](#_Toc483227312)

[**6.1** Системные требования 26](#_Toc483227313)

[**6.2** Процесс инсталляции 26](#_Toc483227314)

[**6.3** Пользовательский интерфейс 31](#_Toc483227315)

[**6.3.1** Элементы управления эффекта Octaver 31](#_Toc483227316)

[**6.3.2** Элементы управления эффекта Delay 31](#_Toc483227317)

[**6.3.3** Элементы управления эффекта Reverb 32](#_Toc483227318)

[**7** ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ И РЕАЛИЗАЦИИ ПРОГРАММНОГО ПРОДУКТА 34](#_Toc483227319)

[**7.1** Описание функций, назначения и потенциальных пользователей ПО 34](#_Toc483227320)

[**7.2** Расчет затрат на разработку ПО 34](#_Toc483227321)

[**7.3** Оценка эффекта от продажи ПО 39](#_Toc483227322)

[**7.4** Расчет показателей эффективности инвестиций в разработку ПО 40](#_Toc483227323)

[ЗАКЛЮЧЕНИЕ 43](#_Toc483227324)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 44](#_Toc483227325)

[ПРИЛОЖЕНИЕ А 45](#_Toc483227326)

[ПРИЛОЖЕНИЕ Б 46](#_Toc483227327)

[ПРИЛОЖЕНИЕ В 47](#_Toc483227328)

# ВВЕДЕНИЕ

Развитие вычислительной математики и техники привело к тому, что звуки, используемые человечеством для повседневной жизни, стали переводиться из аналоговой формы в цифровую. Цифровой звук имеет перед аналоговым огромные преимущества, хотя нельзя забывать и об определенных его недостатках. Основной ценностью цифрового звука является возможность бесконечно долгого хранения и бесконечного тиражирования материала без потери исходного качества, тогда как у аналогового звука качество теряется при каждой записи-перезаписи. Немаловажно и то, что цифровая техника, в отличие от аналоговой, позволяет добиваться идентичности параметров систем при их массовом производстве и сохраняет эту идентичность при эксплуатации, в то время как характеристики аналоговых изделий обычно отличаются на разных экземплярах и ухудшаются со временем. Кроме того, облегчаются передача звука и его обработка современными цифровыми средствами, в первую очередь, специализированными компьютерами.

Программные модули синтеза и обработки звука широко используются в уже давно популярных жанрах электронной музыки, создавая звуки, непривычные человеческому уху, а потому необычные, интересные. Но это только самое очевидное и, конечно же, не единственное приложение. Прогресс приводит к тому, что становится возможным создание модулей обработки звука, эмулирующих настоящие звукозаписывающие студии с любым оборудованием: педалями эффектов, гитарными усилителями и гитарными кабинетами, микрофонами, комнатами. Кроме того, каждый коммерческий и большинство некоммерческих музыкальных произведений проходит ряд обработок на стадии сведения и мастеринга, где исправляются некоторые дефекты записи, корректируются параметры для соответствия стандартам, добавляются эффекты и другое.

Ещё одним достоинством цифрового звука является возможность эмуляции любого аналогового прибора. В настоящее время любой реальный аналоговый прибор можно эмулировать с довольно высокой точностью, которая определяется в большей степени работой по сбору информации о приборе, нежели ограничениями компьютера. Более того, программные модули позволяют получить эффекты, которые невозможно создать с помощью аналоговых приборов. В настоящее время вычислительных ресурсов даже рядовых персональных компьютеров достаточно, чтобы запускать сразу несколько десятков программных модулей обработки звука в реальном времени.

Прогресс электротехники достиг такого уровня, что недостатки цифрового звука становятся ничтожными и меркнут перед достоинствами. Ведь уже несколько лет изготавливаются и широкодоступны АЦП и ЦАП с параметрами достаточными для того, чтобы разница между исходным аналоговым и оцифрованным звуком не была заметна человеческому уху.

Гибкость, удобство, универсальность, дешевизна программного обеспечения для обработки звука и хорошее качество цифрового звука практически не оставляют причин музыкантам, звукорежиссёрам и мастеринг-инженерам не переходить с аналогового оборудования на цифровое.

Таким образом, исходя из вышеперечисленных достоинств цифрового звука и его цифровой обработки, становится понятным, что создание программного обеспечения для обработки звука является актуальной темой. Этот факт, а также моя любовь к музыке обусловили выбор темы дипломного проекта. Цель проекта: разработать программный модуль обработки звуковой дорожки в реальном времени.

# ОБЗОР ЛИТЕРАТУРЫ

## Цифровые аудио-станции и плагины

Цифровая звуковая рабочая станция – электронная или компьютерная система, предназначенная для записи, хранения, редактирования и воспроизведения цифрового звука. Предусматривает возможность выполнения на ней законченного цикла работ, от первичной записи до получения готового результата. Плагины обычно выполняются в виде разделяемых библиотек [1]. Наиболее часто плагины для обработки звука используются в хост-программе DAW (Digital Audio Workstation – цифровая звуковая рабочая станция), где они применяются к отдельным дорожкам или их группам. Примеры наиболее популярных DAW с кратким описанием в таблице 1.1.

|  |  |
| --- | --- |
| Таблица 1.1 – Примеры и описание наиболее популярных DAW | |
| DAW | Описание |
| FL Studio | Цифровая звуковая рабочая станция (DAW) и секвенсер для написания музыки. Музыка создаётся путём записи и сведения аудио- или MIDI-материала. |
| Cubase | Программное обеспечение для создания, записи и микширования музыки. Обладает мощным аудиодвижком и встроенными профессиональными инструментами |
| Nuendo | Отличается от другой популярной линии программных продуктов — Steinberg Cubase тем, что ориентирована не только на музыкантов, но и на любую профессиональную деятельность по озвучиванию в том числе фильмов, телевизионных программ, реклам, радиопередач и прочего. |
| Ableton Live | Программа используется как для студийной работы (аранжировка, сведение), так и для живой игры (импровизация, DJ-инг), и имеет два режима: «Arrangement View» и «Session View». |
| REAPER | Аудиостанция обладает широкими функциональными возможностями и является развитой, профессиональной рабочей системой для создания, записи, редактирования и микширования аудио и MIDI материала, а также мастеринга композиций. При этом программа имеет относительно небольшие размеры. |
| Logic Pro X | Logic Pro X включает огромную коллекцию высококачественных музыкальных сэмплов, инструментов, эффектов и циклов — всё, что нужно для создания композиций профессионального уровня. |

В программах обработки звука плагины выполняют обработку и создание звуковых эффектов, например, мастеринг, применение эквалайзера и сжатие динамического диапазона [2]. Некоторые плагины изменяют технические характеристики звука: глубину, частоту дискретизации и прочее. Практически все аудио-плагины имеют графический пользовательский интерфейс. В GUI (graphical user interface – графический интерфейс пользователя) присутствуют элементы управления (чаще всего это какие-нибудь ручки), меняющие то, как плагин обрабатывает входящие данные. Часто плагин или standalone (автономное) приложение уже имеет встроенный набор пресетов (от английского preset – сохранённый набор настроек), в которых хранятся положения ручек и других параметров. Также часто имеется возможность сохранять свои собственные пресеты. Наиболее распространённый формат аудио-плагинов – VST.

Обычно виды обработок аудио дорожки классифицируют образом, отражённым в таблице 1.2.

|  |  |
| --- | --- |
| Таблица 1.2 – Типы и виды обработок звука | |
| Тип обработки | Вид обработки |
| Частотная | * filter, equalizer, pitchshifter. |
| Динамическая | noise gate, compressor, limiter, soft clipper, expander. |
| Модуляционная | * chorus, flanger, phaser. |
| Частотно-динамическая | multiband compressor. |
| Пространственная | delay, echo, reverb. |
| Искажения | overdrive, distortion, fuzz. |

Программные модули для обработки звука могут работать либо со стерео, либо с моно дорожкой, либо и с той и другой.

Virtual Studio Technology (VST) — формат ресурсозависимых (native) плагинов реального времени, которые подключаются к звуковым редакторам и музыкальным редакторам, секвенсорам. Формат был разработан совместно Propellerhead и Steinberg, впоследствии Propellerhead отказался от дальнейших работ над VST и дальнейшая разработка осуществлялась исключительно Steinberg. В настоящее время в этом формате существуют тысячи плагинов, он стал одним из самых распространённых для звуковых программ. Приложения VST отличаются от плагинов DirectX по нескольким параметрам; в частности, они существуют для Windows, Mac OS X и Linux. Кроме того, в отличие от ранних версий DirectX, плагины VST обладают развитым интерфейсом автоматизации [3].

В данном дипломном проекте разрабатывается программный модуль обработки моно и стерео дорожек с использованием следующих основных эффектов: octaver, delay и reverb. Кроме основных эффектов, разрабатывается обработка типа filter.

## Octaver

Octaver – звуковой эффект или соответствующее устройство, добавляющее к сигналу его копию на октаву или две ниже или выше основного тона. Само обозначение применяется преимущественно к обработке звука электрогитары [4].

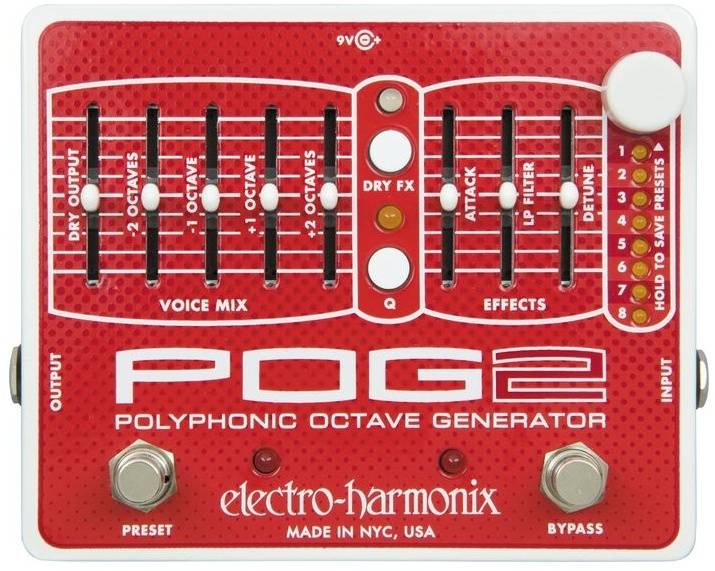


Рисунок 1.1 – Electro-Harmonix POG 2

Octaver является частным случаем эффекта pitch shifter (сдвиг тона). При использовании данного эффекта создаётся впечатление, что вместо одного инструмента играют два в разных октавах. В некоторых педалях эффектов, таких, как, например, Boss OC-2, имеется возможность добавлять два дополнительных тона одновременно – один на октаву ниже основного, другой на две октавы ниже основного. При этом каждый из них можно регулировать по громкости, смешивая в желаемой пропорции. Большинство педалей эффектов типа octaver монофонические, то есть не могут достраивать октавы к аккордам. Полифонические (цифровые) эффекты могут обрабатывать сигнал, содержащий несколько нот. Примером цифрового эффекта octaver является педаль Electro-Harmonix POG 2 (рисунок 1.1).

Для реализации этого эффекта необходимо получить представление звукового сигнала в частотной области, затем обработать его и перевести во временную область.

Дискретное преобразование Фурье (ДПФ) – одна из двух наиболее распространённых и мощных процедур цифровой обработки сигналов (другая процедура – цифровая фильтрация). ДПФ позволяет анализировать, преобразовывать и синтезировать сигналы такими способами, которые невозможны при непрерывной (аналоговой) обработке [5].

ДПФ – это математическая процедура, используемая для определения гармонического или частотного состава дискретных сигналов. Истоком ДПФ является непрерывное преобразование Фурье X(f), которое определяется как

|  |  |
| --- | --- |
| , | (1.1) |

где – некоторый непрерывный сигнал во временной области.

С появлением и развитием цифровых вычислительных машин, работающих с дискретным представлением сигналов, было разработано ДПФ. Оно определяется как дискретная последовательность X(m) в частотной области:

|  |  |
| --- | --- |
| , | (1.2) |

где – дискретная последовательность значений, полученных дискретизацией во временной области непрерывной переменной ;

– основание натурального логарифма;

– константа;

– мнимая единица.

Точные значения частоты разных синусоид зависят как от частоты дискретизации , с которой был дискретизирован исходный сигнал, так и от количества отсчётов N. Все частоты, соответствующие , кратны основной частоте. N разных частот анализа ДПФ определяются выражением

|  |  |
| --- | --- |
|  | (1.3) |

Практический интерес представляют амплитуда и фаза каждого отсчёта . Если представить произвольный отсчёт ДПФ как сумму действительной и мнимой частей

|  |  |
| --- | --- |
| , | (1.4) |

то амплитуда вычисляется как

|  |  |
| --- | --- |
| , | (1.5) |

а фазовый угол вычисляется как

|  |  |
| --- | --- |
| . | (1.6) |

Чтобы получить исходный сигнал во временной области необходимо выполнить обратное дискретное преобразование Фурье (ОДПФ). Выражение для ОДПФ имеет вид:

|  |  |
| --- | --- |
| . | (1.7) |

В 1965 году была опубликована статья Кули и Тьюки [6], описывающая эффективный алгоритм реализации ДПФ. Этот алгоритм сегодня известен как быстрое преобразование Фурье (БПФ).

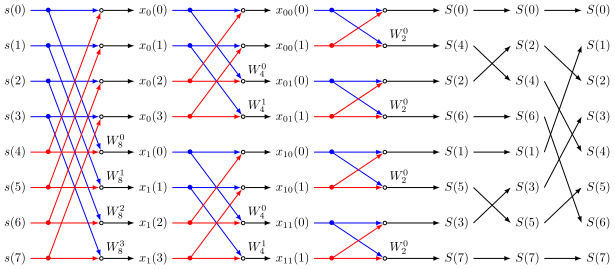


Рисунок 1.2 – Полный граф алгоритма БПФ с прореживанием по частоте для

Алгоритм БПФ по основанию 2 – это эффективный алгоритм вычисления ДПФ, когда длина ДПФ равна натуральной степени двух. Из-за специфической формы элементарных элементов графа он получил название «бабочка». Граф состоит из бабочек. Данная процедура объединения является основной при построении алгоритмов БПФ по основанию два. На рисунке 1.2 представлен полный граф алгоритма БПФ с прореживанием по частоте для N=8 [7].

Операция бабочки в данном алгоритме выполняется в соответствии с формулами 1.8 и 1.9.

|  |  |
| --- | --- |
| ; | (1.8) |

|  |  |
| --- | --- |
| . | (1.9) |
|  |  |

## Delay

Delay – звуковой эффект или соответствующее устройство, имитирующие чёткие затухающие повторы (эхо) исходного сигнала. Эффект реализуется добавлением к исходному сигналу его копии или нескольких копий, задержанных по времени. Под термином delay обычно подразумевается однократная задержка сигнала, в то время как эффект «эхо» - многократные повторы. По принципу действия является частным случаем ревербератора. Отличие заключается в том, что delay имеет одну линию задержки и больший временной интервал (не менее 50-60 мс), который позволяет отделить оригинальный звук от эффекта на слух [8].

Delay – довольно простой эффект для реализации. Входной сэмпл (от английского sample – элемент выборки, замер) сигнала сохраняется в буфер и суммируется с прошлым сэмплом, отстающим в буфере на некоторое число N сэмплов. Обычно прошлый сэмпл ослабляется. И ослабление и длина задержки (N) обычно управляются пользователем [9]. На рисунке 1.3 изображена схема простого эффекта delay.

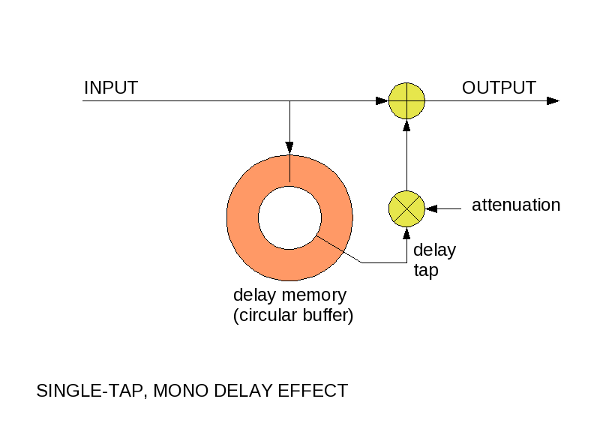


Рисунок 1.3 – Схема простого эффекта delay

Параметры эффекта:

* обратная связь (feedback, regeneration) – при отсутствии обратной связи на выходе будет одна задержка, при увеличении её значения растёт и количество сигналов на выходе;
* время задержки (delay, time) – промежуток времени между исходным сигналом и его задержкой (задержками);
* баланс (balance, mix) – соотношение исходного и задержанного сигналов.

Основные типы эффекта delay:

* slapback – одиночная задержка длительностью до 120 мс;
* echo – более длительная задержка с обратной связью;
* reverse – в цифровых эффектах возможно воспроизведение записанного в буфер сигнала в обратном порядке;
* ping pong – поочерёдное панорамирование задержанного сигнала в левый/правый канал.

В данном проекте реализуется эффект delay, с изменяемой задержкой в пределах от 20 до 1000 мс, с возможностью синхронизации с темпом музыкальной композиции.

## Reverb

Реверберация – это процесс постепенного уменьшения интенсивности звука при его многократных отражениях. Иногда под реверберацией понимается имитация данного эффекта с помощью ревербераторов [10].

Явление реверберации состоит в суперпозиции различных эхосигналов от одного источника звука. Эффект реверберации можно наблюдать в закрытых помещениях после выключения источника звука. Обычно избыточная длительность реверберации приводит к неприятной гулкости, «пустоте» помещения, а недостаточная – к резкому отрывистому звучанию, лишённому музыкальной полноты. Искусственно создаваемая реверберация в определённых пределах способствует улучшению качества звучания, создавая ощущение приятного «резонанса» помещения.

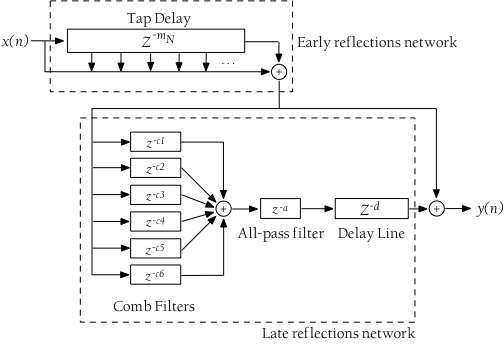


Рисунок 1.4 – Структурная схема ревербератора Мурера

Реверберацию можно разделить на два компонента, которые видны на представлении импульсного отклика в комнате:

1. Ранние отражения – первые отражения, которые мы слышим в течение приблизительно 100 мс вместе с прямым звуком от источника.
2. Поздняя реверберация – слышимый звук после 100 мс, до момента его полного затухания. Поздняя реверберация характеризуется плотной текстурой рассеянных отражений, которые достигают наших ушей несколькими путями. Эти рассеянные отражения находятся не в фазе относительно друг друга, что вызывает эффект гребенчатой фильтрации. Мы воспринимаем этот эффект как «атмосферу», характер места [11].

Структурная схема обработки сигнала ревербератором показана на рисунке 1.4. Такое решение предложил звуковой инженер, музыкант и доктор наук Стэнфордского университета Джеймс Энди Мурер, усовершенствовав алгоритм Шрёдера [12]. Исходя из Алгоритма Мурера, к исходному сигналу добавляется сигнал, прошедший через сеть ранних отражений, состоящую из множества delay -линий. Далее к нему добавляется сигнал обработанный сетью поздних отражений, состоящей из параллельных гребенчатых фильтров и фазового фильтра с delay-линией.

Гребенчатый фильтр — в обработке сигналов электронный фильтр, при прохождении сигнала через который к нему добавляется он сам с некоторой задержкой. В результате получается фазовая компенсация. АЧХ гребенчатого фильтра состоит из ряда равномерно распределённых пиков, так что она выглядит как гребёнка.

В цифровых системах, фильтр задаётся формулой 1.10:

|  |  |
| --- | --- |
| , | (1.10) |

где – коэффициенты;

– запаздывание сигнала;

– входной сигнал;

– выходной сигнал.

На данный момент существуют аналоги отдельных эффектов, реализуемых в данном дипломном проекте, как бесплатные, так и довольно дорогие. Комбинация этих эффектов создаёт новый, необычный, слабо представленный на рынке эффект. В определённых кругах этот эффект называют эффектом shimmer (от английского – мерцание). Целью данного дипломного проекта является создание программного модуля эффекта shimmer с возможностью настройки каждого из подкомпонентов в отдельности.

# СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

Разрабатываемый программный модуль разбит на отдельные логически взаимосвязанные блоки, что является необходимым условием для обеспечения гибкости его структуры. При данном подходе становится возможной выборочная модернизация отдельных частей программного кода, с минимальным влиянием на остальные части проекта, либо, в идеальном случае, вовсе без их изменения.

## Структура программного модуля

В соответствии с методологией системного подхода в разработке архитектуры, программный модуль разбивается на совокупность сущностей, представленных на структурной схеме (см. чертеж ГУИР.400201.161 С1).

Структурная схема была составлена исходя из основных стадий обработки сигнала, а также функций, которые должен предоставлять программный модуль обработки звуковой дорожки в реальном времени. Структурная схема состоит из следующих блоков:

* блок преобразования входного сигнала для обработки;
* блок эффекта «octaver»;
* блок эффекта «delay»;
* блок эффекта «reverb»;
* блок преобразования обработанных данных в выходной сигнал;
* блок настройки параметров обработки;
* блок интерфейса пользователя.

*Блок преобразования входного сигнала для обработки* принимает исходный сигнал. Перед передачей сигнала к блокам добавления эффектов в нём устраняются шумы. Это выполняется за счёт фильтра нижних частот, фильтра верхних частот и noise gate. Также корректируется уровень входного сигнала для обработки последующими блоками.

*Блок эффекта «*octaver*»* принимает уже подготовленный к обработке звуковой сигнал. Блок предназначен для добавления гармоник частотой степени двойки к исходному звуковому сигналу. Внутри блока с помощью ДПФ получается сигнал в частотном домене. Далее сигнал обрабатывается и с помощью ОДПФ переводится во временной домен. Затем обработанный сигнал складывается с исходным сигналом в задаваемой пользователем пропорции и подаётся на следующий логический блок.

*Блок эффекта «*delay*»* создаёт копию поступившего на вход сигнала и добавляет её к исходному через некоторое задаваемое пользователем время. Сигнал с выхода снова подаётся на вход этого блока уже с уменьшенным уровнем. Внутри этого блока также происходит обработка сигнала, подаваемого с выхода на вход. С выхода блока эффекта «delay» сигнал поступает на блок эффекта «reverb».

*Блок эффекта «*reverb*»* отвечает за эмуляцию постепенного уменьшения интенсивности звука при его многократных отражениях. Он включает в себя два компонента: блок ранних отражений и блок поздних отражений.

*Блок преобразования обработанных данных в выходной сигнал* стоит последним в тракте следования сигнала. Он включает в себя фильтры низких и высоких частот. Также этот блок отвечает за уровень сигнала, подаваемого на выход.

*Блок интерфейса пользователя* представляет собой совокупность средств, при помощи которых пользователь взаимодействует с программным модулем обработки звуковой дорожки в реальном времени. Для построения интерфейса используются компоненты, встроенные в фреймворк Juce (о фреймворке Juce написано в подразделе 2.2). Данный блок отвечает за получение ввода пользователя и связан лишь с блоком настройки параметров обработки двусторонней связью.

*Блок настройки параметров обработки* предназначен для преобразования ввода пользователя и хранения параметров, влияющих на работу блоков эффектов. Среди таких параметров обязательно будут присутствовать следующие:

* уровень входного сигнала (input level) для блока преобразования входного сигнала для обработки;
* соотношение обработанного сигнала к необработанному (mix) для всех эффектов;
* время задержки (delay time) для эффектов delay и reverb;
* уровень сигнала подаваемого с выход на вход (feedback) для эффекта delay;
* уровень выходного сигнала (output level) для блока преобразования обработанных данных в выходной сигнал.

Данный блок кроме двусторонней связи с блоком интерфейса пользователя имеет исходящие двусторонние связи с каждым из блоков эффектов обработки звуковой дорожки, а также блоками преобразования входного сигнала для обработки и блоком преобразования обработанных данных в выходной сигнал.

## Выбор программных средств

В качестве языка программирования задан язык C++, а в качестве среды разработки – Visual Studio 2012 Express. Для выполнения поставленной задачи необходимо выбрать фреймворк, который совместим с требованиями задания, позволяет избежать низкоуровневого программирования, включающий компоненты графического интерфейса пользователя, а также компоненты для работы со звуком. Фреймворк Juce удовлетворяет всем вышеперечисленным требованиям.

Juce – это открытый кроссплатформенный инструментарий разработки ПО (фреймворк) для языка C++, используемый для разработки GUI приложений и плагинов [13].

Цель Juce – позволить компилировать один и тот же исходный текст в программы, одинаково работающие на Windows, Mac OS X и Linux (последние версии – также iPhone и Android) платформах. Он поддерживает различные среды разработки и компиляторы, такие как GCC, Xcode и Visual Studio.

Juce впервые опубликован в 2004, держателем его кода является британская компания Raw Material Software. Имеет двойную GPL/коммерческую лицензию.

Juce содержит классы, позволяющие программе работать со звуком и графикой. За счёт этого нуждающиеся в дополнительных библиотеках программисты могут использовать только библиотеку Juce, или хотя бы сократить количество сторонних библиотек, которые они используют. На это разработчиков Juce вдохновил JDK языка Java. По их заявлению, они собирались из Juce сделать «что-то подобное для С++».

Наиболее важная особенность Juce по сравнению с другими аналогичными фреймворками — большой набор аудиофункций. Дело в том, что библиотека Juce сначала была разработана как часть аудиосеквенсора Tracktion, и лишь затем стала самостоятельным продуктом. Juce включает в себя поддержку воспроизведения звука через аудио и MIDI интерфейсы, полифонические синтезаторы, понимает файлы распространённых аудиоформатов (таких как WAV, AIFF, FLAC, и Vorbis). Он также содержит интерфейсы-оболочки для построения различных аудио плагинов, таких как эффекты и инструменты VST. Это привело к его широкому распространению в сообществе разработчиков аудио-ПО.

В поставку Juce входят классы-обёртки для создания аудиоплагинов. При сборке аудиоплагина, получается единый бинарный файл, который поддерживает несколько форматов: VST, RTAS, AU. Поскольку весь платформо- и форматозависимый код содержится в классах-обёртках, то пользователь может собирать плагины в формате VST/RTAS/AU для макинтошей и Windows из одного и того же исходного кода.

Имеется также неофициальное ответвление библиотеки, расширенное дополнительными возможностями, поддерживаемое сообществом, которое называется Juced.

На сайте этого варианта фреймворка можно найти также дополнительную документацию по Juce, которая поможет освоить библиотеку [14]. Последняя версия фреймворка на данный момент – 4.3.

Фреймворк Juce имеет встроенную хост-программу Juce Plug-In Host [15]. Она имеет только самый необходимый функционал для хост-программы и требует небольшого объёма вычислительных ресурсов компьютера, что позволяет быстро проверять работу плагина. Кроме того с её помощью можно производить отладку плагина, что очень важно для программиста. Чтобы разрабатываемый плагин обрабатывал входной сигнал необходимо соединить блок Audio Input и Midi Input с входами плагина, а выходы плагина соединить с блоком Audio output. К входам разрабатываемого плагина также подключён плагин AudioFilePlayer, который используется для проигрывания записанной звуковой дорожки. Интерфейс программы Juce Plug-In Host изображён на рисунке 2.1.

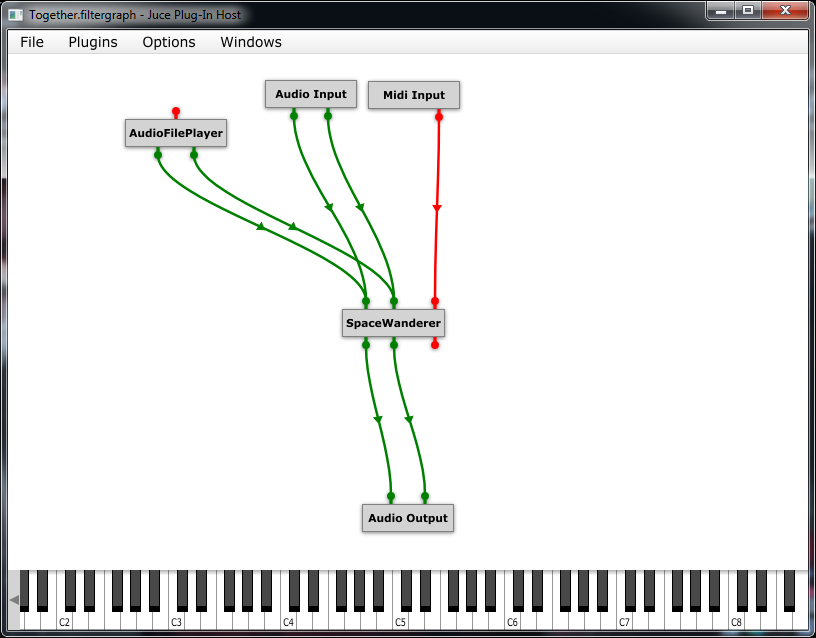


Рисунок 2.1 – Интерфейс хост-программы Juce

Для разработки VST плагинов необходимо скачать пакет VST SDK с официального сайта Steinberg – разработчика этого формата. Текущая версия этого средства разработки – 3.6.7.

Также при разработке программного модуля обработки звуковой дорожки в реальном времени будет использоваться система управления версиями – Git. Выбор сделан в пользу именно этой системы управления версиями из-за ряда причин:

1. Git использует распределённую модель вместо традиционной клиент серверной. Она не нуждается в централизованном хранилище: вся история изменения документов хранится на каждом компьютере в локальном хранилище. При необходимости локальный репозиторий может быть передан командой push на удалённый репозиторий.
2. Постоянное подключение к сети не является необходимым.
3. Git обладает высокой производительностью.
4. Продуманная и уже знакомая мне система команд.

В качестве сервиса предоставляющего хостинг для размещения git-репозиториев выбран веб-сайт GitHub [16].

Таким образом, в данном дипломном проекте при разработке программного модуля обработки звуковой дорожки в реальном времени

* используется модульное программирование;
* в качестве фреймворка выбрана библиотека Juce;
* в качестве формата ресурсозависимого плагина реального времени для DAW выбран VST;
* используется система контроля и управления версиями Git;
* удалённый репозиторий, содержащий пояснительную записку, чертежи, документы и проект программного модуля обработки звуковой дорожкой в реальном времени хранится на сайте GitHub по ссылке https://github.com/Andrewregrets/VST-Plugin.

# ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе подробно рассматривается функционирование программных модулей разработанного программного продукта. Перечислены соответствующие классы и большинство их компонентов. Состав и отношения основных классов показаны на диаграмме классов (см. чертёж ГУИР.400201.161 РР.1).

Основными классами, которые предоставляет фреймворк Juce при создании нового проекта типа AudioPlugin, являются:

* + - AudioProcessorEditor;
    - AudioProcessor.
  1. Класс ShimmerAudioProcessorEditor

Класс ShimmerAudioProcessorEditor унаследован от класса AudioProcessorEditor. Данный класс выполняет функцию *блока интерфейса пользователя*, а именно обработку пользовательского ввода и отображение графической информации на экран. Данный класс также унаследован от абстрактных классов SliderListener и ButtonListener.

Класс ShimmerAudioProcessorEditor имеет следующие поля:

1. window\_width – константная переменная типа int, ширина окна.
2. window\_height – константная переменная типа int, высота окна.
3. background\_image – объект класса Image. Используется для формирования фона приложения.
4. &processor – ссылка на объект типа ShimmerAudioProcessor, необходима для редактирования параметров обработки звука. ShimmerAudioProcessorEditor получает изменения параметров, которые вносит пользователь, и оповещает об этом ShimmerAudioProcessor.
5. delay\_gui\_components – структура, содержащая элементы управления для эффекта Delay. Элементы управления – ручки и кнопки.
6. reverb\_gui\_components – структура, содержащая элементы управления для эффекта Reverb.
7. octaver\_gui\_components - структура, содержащая элементы управления для эффекта Octaver.

Класс ShimmerAudioProcessorEditor имеет следующие основные методы:

* ShimmerAudioProcessorEditor(ShimmerAudioProcessor&);
* ~ShimmerAudioProcessorEditor();
* resized() – метод, в котором задаётся расположение и размеры компонентов графического пользовательского интерфейса.
* sliderValueChanged (Slider\* sliderThatWasMoved) – метод, обрабатывающий события, связанные с изменением значений ручек.
* buttonClicked (Button\* buttonThatWasClicked) ) – метод, обрабатывающий события, связанные с изменением состояния кнопок.

Первые два метода класса ShimmerAudioProcessorEditor – конструктор и деструктор. Параметром конструктора является ссылка на объект на объект ShimmerAudioProcessor.

1. resized() – метод, в котором задаётся расположение и размеры компонентов графического пользовательского интерфейса
2. paint (Graphics&) – метод, в котором происходит заполнение окна программного модуля графикой;
3. sliderValueChanged (Slider\* sliderThatWasMoved) – метод, обрабатывающий события, связанные с изменением значений ручек.
4. buttonClicked (Button\* buttonThatWasClicked) ) – метод, обрабатывающий события, связанные с изменением состояния кнопок.

## Класс ShimmerAudioProcessor

Класс ShimmerAudioProcessor унаследован от класса AudioProcessor и отвечает за аудио и миди входы и выходы, логику обработки звука. Главный метод этого класса, где непосредственно обрабатываются сэмплы цифрового звука.

## Класс DelayLine

Данный класс описывает поведение элементарной дилэй-линии, на основе которой функционируют гребенчатый (Comb) и всепропускающий (All-pass) фильтр. Это значит, что соответствующие классы фильтров содержат поля класса DelayLine. Также объект класса DelayLine является полем класса Delay,

Класс содержит следующие приватные поля:

* int sample\_rate – частота дискретизации;
* int writePos –текущая позиция для записи в буфер
* readPosA, MAX\_DELAY\_SAMPLES;
* float delay\_ms, delay\_samples, fraction, MAX\_DELAY\_MS;
* float \*buffer;

Функциональное проектирование – аналог соответствующего раздела аппаратного дипломного проекта. Это основной раздел ПЗ, дающий ключ к пониманию функционирования разрабатываемой программы и исчерпывающую информацию о ее структуре с точки зрения описания данных и обрабатывающих их подпрограмм – функций и процедур. Обработка данных, как известно, является основной целью работы любой программы. Поэтому здесь описываются заданные константы, пользовательские переменные, внутренние и внешние массивы и так далее. При определении необходимости в какой-либо обработке данных вводится соответствующая подпрограмма. При использовании объектно- ориентированного подхода могут описываться структура и взаимоотношения между классами со ссылками на чертеж диаграммы классов. Если при программировании (особенно в системах визуального программирования) используются стандартные библиотеки, то упор должен делаться на описание самостоятельно разработанных фрагментов кода. Стандартные функции лишь упоминаются. При использовании БД описывается структура таблиц и связи между ними со ссылками на чертеж модели данных. Здесь же раскрываются и потоки данных между подпрограммами, то есть входные и выходные аргументы процедур и функций. Рекомендуется структурировать раздел в соответствии с блоками, выделенными на этапе системного проектирования. Данный раздел должен сопровождать чертеж диаграммы последовательности и содержать ссылки на него. Могут быть ссылки на чертеж схемы данных.

# РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

Разработка программных модулей – аналог разработки принципиальной схемы аппаратного дипломного проекта. В этом разделе подробно описываются уже внутренние алгоритмы ключевых процедур и функций с разбиением на отдельные подразделы. Здесь же описывается реализация наиболее интересных алгоритмов, например, алгоритмов шифрования. Данный раздел может сопровождать чертежи схем программ и

содержать ссылки на них.

# ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

В разделе, посвященном программе и методике испытаний,

описываются внутренние (если самотестирование заложено в программу) и внешние средства тестирования. Могут использоваться как оригинальные, так и стандартные тесты. Рассматриваются способы проверки надежности (устойчивости, стабильности и так далее) разработанной программы в различных режимах, включая многопользовательский и многозадачный режимы, а также корректность обработки входных, промежуточных и выходных данных, в том числе: в области граничных значений допустимых диапазонов, заведомого неправильных данных, файлов большого размера и так далее. Для каждого из тестов приводятся исходные данные, параметры и результаты.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## Системные требования

Требования к аппаратному обеспечению в большей степени диктуются используемой DAW, нежели разработанным программным модулем. Требования к программному обеспечению зависят от используемой DAW. В качестве примера в таблице 6.1 приведены системные требования новой цифровой звуковой рабочей станции фирмы Steinberg Cubase Pro 9.

Формат программного модуля – VST. Данный формат широко распространён и поддерживается всеми известными DAW: от популярных и функциональных Cubase и Pro Tools, до малоизвестной и легковесной TunaFish.

Таблица 6.1 – Системные требования Cubase Pro 9

|  |  |
| --- | --- |
| Mac OS X | Windows |
| OS X 10.11 / macOS Sierra | 64-Bit Windows 7 / 8.x / 10 |
| Многоядерный процессор 64-bit Intel или AMD (рекомендован Intel i5 или более быстрый) | |
| 4 GB ОЗУ (рекомендовано 8 GB или больше) | |
| 18 GB свободного места на жёстком диске | |
| разрешение экрана 1366 x 768 (рекомендовано 1920 x 1080) | |
| Графическая карта с DirectX 10 и поддержкой WDDM 1.1 (для ОС Windows) | |
| USB порт для USB-eLicenser (управление лицензиями) | |
| Аудиоинтерфейс, поддерживаемый ОС (для высокой производительности, низких задержек в звуке рекомендуется поддержка ASIO) | |
| Соединение с интернетом (для активации, регистрации аккаунта и продукта, дополнительных загрузок в ходе инсталляции) | |

Файл разработанного модуля обработки звука Shimmer.dll имеет размер всего 5 MiB.

## Процесс инсталляции

Процесс инсталляции программного модуля обработки звуковой дорожки в реальном времени указан для DAW Cubase 7 Elements на операционной системе Windows 7 Service Pack 1. Для других операционных систем и цифровых звуковых рабочих станций процесс инсталляции может отличаться, но основные этапы будут такими же.

Алгоритм добавления и использования разработанного плагина в Cubase следующий:



Рисунок 6.1 – Создание проекта в Cubase 7 Elements

1. Скопировать с диска файл Shimmer.dll в папку с VST плагинами на вашем компьютере. В данном случае была выбрана папка D:\vst\_plugins\_folder.
2. Открыть цифровую звуковую рабочую станцию (2 раза кликнуть на исполняемый файл Cubase LE AI Elements 7.exe) и создать в ней проект. Для этого в меню More (см. рисунок 6.1) выбрать пункт Empty, указать папку, в которой будет создан проект, нажать Create.



Рисунок 6.2 – Открытие меню Plug-in Information

1. Открыть меню с информацией о плагинах. В строке меню выбрать пункт Devices, в выпадающем списке выбрать Plug-in Information (см. рисунок 6.2).
2. Указать путь к папке, в которой находится файл Shimmer.dll. Во вкладке VST Plug-ins нажать кнопку VST 2.x Plug-in Path. В появившемся окне нажать кнопку Add (см. рисунок 6.3). В появившемся окне выбрать местоположение папки и нажать OK. Снова нажать OK. Нажать кнопку Update. Как видно на рисунке 6.4, в списке доступных плагинов появился файл с именем Shimmer.dll.

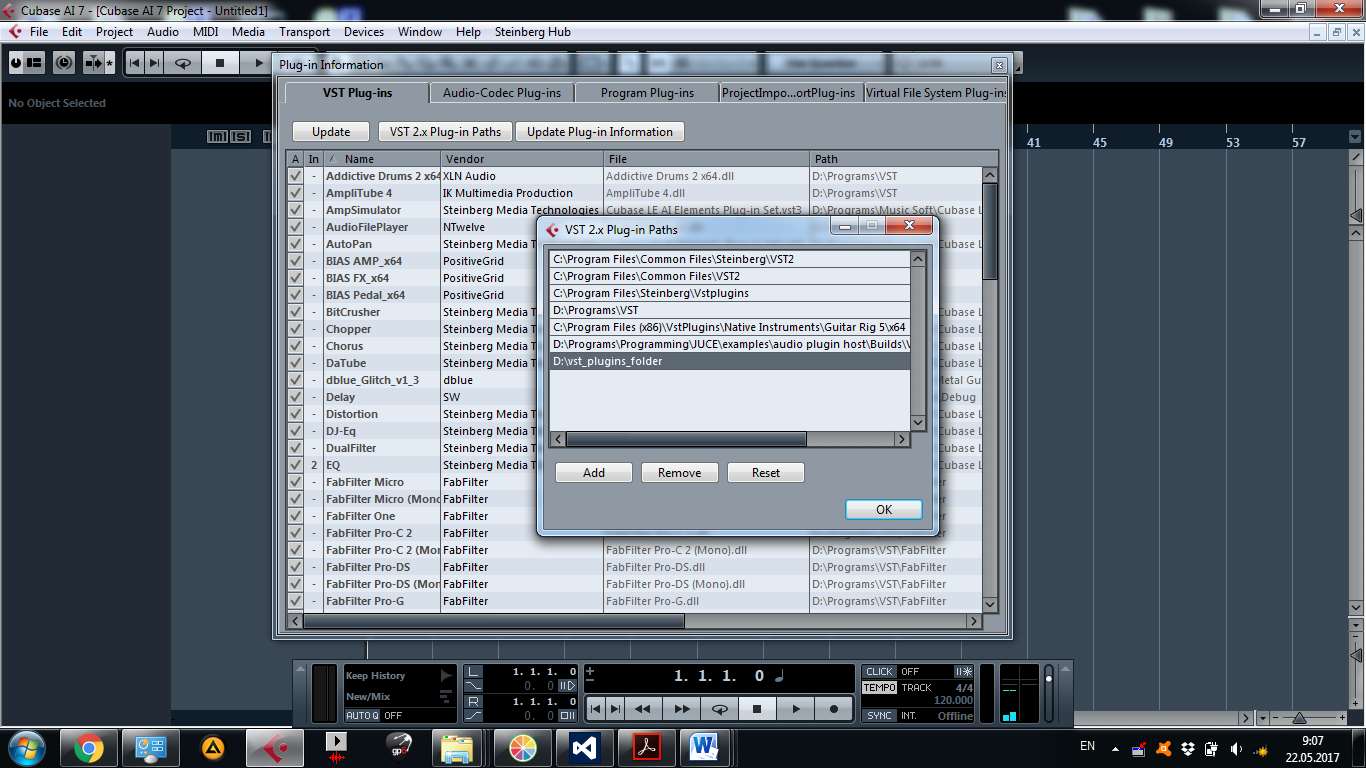


Рисунок 6.3 – Окно VST 2.x Plug-in Paths

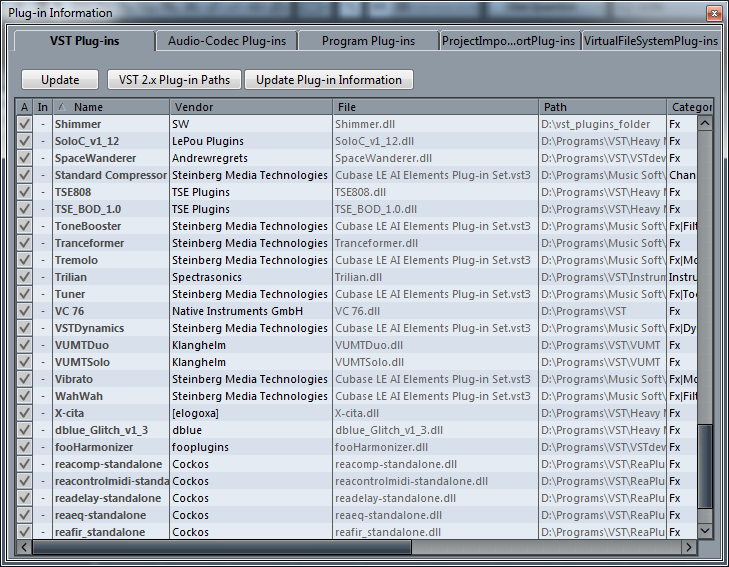


Рисунок 6.4 – Окно Plug-in Information

1. Чтобы использовать плагин на аудиодорожке нужно её создать. Нужно нажать правой кнопкой по области дорожек и выбрать Add Audio Track … (см. рисунок 6.5). В появившемся окне выбрать конфигурацию дорожки Mono, указать любое имя и нажать Add Track (см. рисунок 6.6).

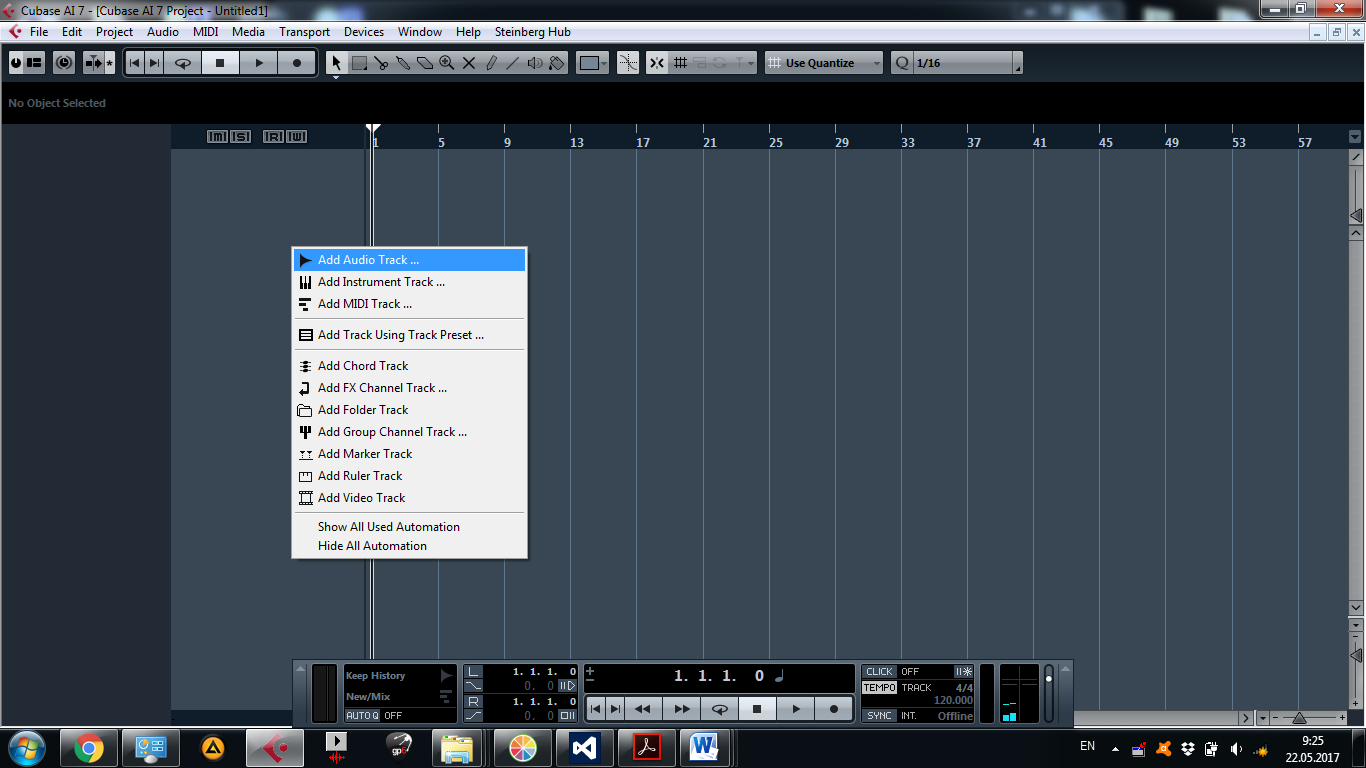


Рисунок 6.5 – Добавление звуковой дорожки 1

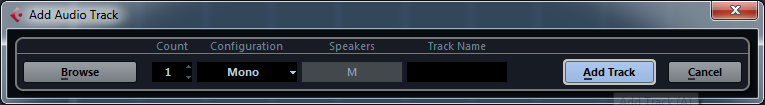


Рисунок 6.6 – Добавление звуковой дорожки 2



Рисунок 6.7 – Применение плагина к звуковой дорожке

1. Применить плагин к звуковой дорожке. Для этого необходимо выбрать дорожку, в меню Inserts нажать на пустой слот, выбрать необходимый плагин (см. рисунок 6.7). Теперь звуковая дорожка обрабатывается плагином в реальном времени.

## Пользовательский интерфейс

Графический пользовательский интерфейс представлен на рисунке 6.8

Программный модуль, как и описывалось ранее, состоит из трёх компонентов: Delay, Octaver, Reverb. Интерфейс представляет собой три секции, соответствующие этим трём компонентам.



Рисунок 6.8 – Графический интерфейс программного модуля обработки звуковой дорожки в реальном времени.

* + 1. Элементы управления эффекта Octaver

Компонент Octaver имеет следующие элементы графического интерфейса:

* кнопка Bypass;
* ручка Mix;
* ручка Octave low;
* ручка Octave high;

При нажатой кнопке Bypass, исходный звуковой сигнал не будет претерпевать обработку эффектом Delay и поступит на вход следующего компонента без изменений.

Ручка Mix регулирует соотношение обработанного сигнала эффектом Octaver к необработанному.

Ручка Octave low регулирует громкость добавляемого сигнала, который на октаву ниже основного.

Ручка Octave high регулирует громкость добавляемого сигнала, который на октаву выше основного.

* + 1. Элементы управления эффекта Delay

Эффект Delay управляется следующими элементами графического интерфейса:

* кнопка Bypass;
* ручка Mix;
* ручка Delay;
* ручка Feedback
* кнопка Synchronize
* кнопка Dotted;
* кнопка Second dotted.

При нажатой кнопке Bypass, звуковой сигнал, поступающий от блока Octaver, не будет претерпевать обработку эффектом Delay и поступит на вход следующего компонента Reverb без изменений.

Ручка Mix определяет количество эффекта в процентах, то есть громкость задержанного сигнала. Пределы изменения: от нуля до ста.

Ручкой Delay задаётся время, спустя которое сигнал будет повторяться. Пределы изменения этого параметра: от нуля до двух тысяч миллисекунд.

Ручка Feedback отвечает за обратную связь эффекта Delay. При минимальном значении равном нулю, компонент будет добавлять одну копию исходного сигнала. При максимальном – бесконечное количество копий. Если значение этого параметра находится в пределах от минимального, до максимального, громкость периодических копий сигнала будет постепенно уменьшаться.

Кнопка Synched меняет режим работы эффекта Delay на синхронный и обратно. В синхронном режиме программный модуль обработки звуковой дорожки в реальном времени получает темп от хост-программы, то есть от цифровой звуковой рабочей станции, и синхронизирует копии сигнала с полученным темпом. В этом режиме ручка Delay изменяет своё предназначение: с её помощью задаётся делитель темпа.

Кнопка Dotted добавляет одну вторую от времени между долями к синхронизированным с темпом повторам.

Кнопка Second dotted добавляет одну четверть от времени между долями к синхронизированным с темпом повторам.

* + 1. Элементы управления эффекта Reverb

Эффект Reverb управляется следующими элементами графического интерфейса:

* кнопка Bypass;
* ручка Decay;
* ручка Mix.

При нажатой кнопке Bypass, звуковой сигнал не будет претерпевать обработку эффектом Reverb и поступит на выход.

Ручка Mix определяет соотношение необработанного сигнала к обработанному эффектом сигналу. Этот параметр измеряется в процентах, то есть пределы изменения: от нуля до ста.

Ручкой Decay задаётся время отражений.

# ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТКИ И РЕАЛИЗАЦИИ ПРОГРАММНОГО ПРОДУКТА

## Описание функций, назначения и потенциальных пользователей ПО

Программный продукт предназначен для обработки звуковой дорожки в реальном времени. Он позволяет как немного обогатить звук дорожки гармониками, так и добавить объёма, эмулируя отражения комнаты. Программный продукт позволяет получить и довольно экстремальный эффект shimmer. Этот продукт может быть использован на обычных настольных компьютерах и ноутбуках во всех популярных цифровых звуковых рабочих станциях.

Потенциальные пользователи программного модуля обработки звуковой дорожки в реальном времени – музыканты, имеющие возможность подключить свой инструмент к компьютеру для получения цифрового представления звука и режиссёры, обрабатывающие записанные музыкантами дорожки.

В связи с распространением цифровой обработки звука с помощью персонального компьютера на рынке есть достаточный спрос на данный программный продукт. Программный модуль обработки звуковой дорожки в реальном времени предназначен для использования широким кругом пользователей и разрабатывается для свободной реализации на рынке IT.

Исходя из тенденций на рынке модулей обработки звука, а также дополнительного маркетингового исследования, приложение будет востребовано на рынке в течение четырех лет: в 2017 году планируется реализовать 30 копий, в 2018 году планируется реализовать 50 копий, в 2019 планируется реализовать 40 копий, в 2020 планируется реализовать 30 копий. Итого 150 копий. На основании анализа цен на аналогичные программные продукты цена программного модуля обработки звуковой дорожки в реальном времени будет составлять 50 руб.

Экономическая целесообразность инвестиций в разработку и реализацию представленного программного продукта определяется на основе расчета таких показателей, как:

* смета затрат и отпускная цена программного продукта;
* прибыль от реализации программного продукта;
* рентабельность инвестиций в разработку программного продукта.

## Расчет затрат на разработку ПО

Затраты на основную заработную плату команды разработчиков определяются исходя из состава и численности команды, размеров месячной заработной платы каждого из участников команды, а также общей трудоемкости разработки программного обеспечения.

Расчет величины основной заработной платы участников команды осуществляется по формуле 7.1:

|  |  |
| --- | --- |
|  | (7.1) |
|  |  |

где *n* – количество исполнителей занятых в разработке;

Т1ч*i* – часовая тарифная ставка *i*-го исполнителя, руб.;

Кт*i* – тарифный коэффициент разряда *i*-го исполнителя;

Тчi – количество часов работы исполнителя, ч;

К – коэффициент премирования, принятый равным 1,3.

Тарифная ставка первого разряда (на 01.03.2017) на предприятии составляет 120 рублей. Среднемесячная расчетная норма рабочего времени на 2017 год составляет 168 часов. Коэффициент премирования равен 1,3.

В процессе разработки приложения принимала участие команда из двух исполнителей: инженера-программиста 10 разряда с тарифным коэффициентом 2,48, и руководителя проекта 12 разряда с тарифным коэффициентом 3,25. Руководитель проекта ставит задачи перед инженером-программистом, среди которых следующие работы: проектирование интерфейса, написание кода, отладка и тестирование программного продукта.

Результаты расчёта основной заработной платы команды разработчиков представлены в таблице 7.1.

Таблица 7.1 – Расчет затрат на основную заработную плату команды разработчиков

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Участник  команды | Раз-  ряд | Тариф-  ный коэффи-  циент | Месячная  тарифная  ставка, руб. | Часовая  тарифная  ставка, руб. | Трудоём-  кость работ, ч. | Заработная  плата, руб. |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Инженер-програм-  мист | 10 | 2,48 | 297,60 | 1,77 | 400 | 708,57 |
| Руково-дитель проекта | 12 | 3,25 | 390 | 2,32 | 50 | 116,07 |
| Итого |  |  |  |  |  | 824,64 |
| Премия |  |  |  |  |  | 247,39 |

*Продолжение таблицы 7.1.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Основная заработная плата |  |  |  |  |  | 1072,04 |

Затраты на дополнительную заработную плату команды разработчиков включает выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей) и определяется по формуле 7.2:

|  |  |
| --- | --- |
|  | (7.2) |

где – норматив дополнительной заработной платы, 15%.

Размер дополнительной заработной платы исполнителей составит:

Отчисления на социальные нужды (в фонд социальной защиты населения и на обязательное страхование) определяются в соответствии с действующими законодательными актами по формуле 7.3:

|  |  |
| --- | --- |
|  | (7.3) |

где – норматив отчислений на социальные нужды, 34,6%.

Размер отчислений в фонд социальной защиты населения и на обязательное страхование составляет:

Расходы по статье «Машинное время» включают оплату машинного времени, необходимого для разработки и отладки ПО и определяются по формуле 7.4:

|  |  |  |
| --- | --- | --- |
|  |  | (7.4) |

где Цм – цена одного машино-часа, руб.;

Тч – количество времени, затраченного на разработку проекта, ч.

Расходы по статье «Машинное время», на разработку программного продукта составляют:



Размер прочих затрат, включающий затраты на аренду помещения, освещение и отопление, вычисляется по формуле 7.5:

|  |  |
| --- | --- |
|  | (7.5) |

Норматив прочих затрат принят равным 16%. Тогда прочие затраты на разработку программного продукта составляют:

Общая сумма расходов по смете (Ср) определяется как сумма выше рассчитанных показателей:

|  |  |  |
| --- | --- | --- |
|  |  | (7.6) |

Подставив значения в формулу расчета расходов по смете 7.6, получим:

В дополнение к выше рассчитанным параметрам, определяются расходы на сопровождение и адаптацию программного продукта (Зса):

|  |  |  |
| --- | --- | --- |
|  |  | (7.7) |

где Нрса – это норматив расходов обусловленных сопровождением и адаптацией ПО (5%).

Используя формулу 7.7, определим значение расходов:

Общая сумма расходов на разработку приложения, как полная себестоимость программного продукта (Сп) определяется по формуле 7.8:

|  |  |  |
| --- | --- | --- |
|  |  | (7.8) |

Общая сумма расходов на разработку:

В таблице 7.2 указаны рассчитанные значения затрат на разработку программного продукта.

Таблица 7.2 – Затраты на разработку программного обеспечения

|  |  |  |
| --- | --- | --- |
| Статья затрат | Условное обозначение | Сумма, руб. |
| Основная заработная плата |  | 1072,04 |
| Дополнительная заработная плата |  | 160,81 |
| Отчисления на социальные нужды |  | 426,56 |
| Расходы по статье «Машинное время» |  | 540,00 |
| Прочие затраты |  | 171,53 |
| Сумма расходов по смете |  | 2370,93 |
| Затраты на сопровождение и адаптацию |  | 118,55 |
| Полная себестоимость |  | 2489,48 |

Прогнозируемая прибыль программного продукта рассчитывается по формуле 7.9:

|  |  |  |
| --- | --- | --- |
|  |  | (7.9) |

где Пр – прибыль от реализации ПП;

Ур – уровень рентабельности(15%);

Сп – себестоимость программного продукта.

Получаем:

В связи с тем, что компания-исполнитель является резидентом ПВТ и по текущему законодательству РБ, такие предприятия освобождаются от уплаты налога на прибыль и на добавленную стоимость, то прогнозируемая отпускная цена ПП будет определяться по формуле 7.10:

|  |  |  |
| --- | --- | --- |
|  |  | (7.10) |

Подставляя значения в формулу (7.10), получаем прогнозируемую отпускную цену программного продукта:

## Оценка эффекта от продажи ПО

Экономический эффект организации-разработчика программного обеспечения в данном случае заключается в получении прибыли от его продажи множеству потребителей. Прибыль от реализации в данном случае напрямую зависит от объемов продаж, цены реализации и затрат на разработку ПО.

Как описано в подразделе 7.1, цена программного продукта выбрана 50 рублей. Запланированное количество проданных копий – 150. Прибыль от продажи одной копии рассчитывается по формуле 7.11:

|  |  |  |
| --- | --- | --- |
|  |  | (7.11) |

где Ц – цена реализации одной копии;

НДС – сумма налога на добавленную стоимость;

Зр – сумма расходов на разработку и реализацию;

N – количество реализованных копий.

Сумма налога на добавленную стоимость высчитывается по формуле 7.12:

|  |  |  |
| --- | --- | --- |
|  |  | (7.12) |

где %НДС – ставка налога на добавленную стоимость, 20%.

Тогда сумма налога на добавленную стоимость равна:

В связи с тем, что компания-исполнитель является резидентом ПВТ и по текущему законодательству РБ, такие предприятия освобождаются от уплаты налога на прибыль и на добавленную стоимость, тогда в подсчёте прибыли от продажи одной копии ПП НДС участвовать не будет.

Итого, прибыль за продажу одной копии программного продукта:

## Расчет показателей эффективности инвестиций в разработку ПО

Экономическая целесообразность инвестиций в разработку и использование программного продукта осуществляется на основе следующих показателей:

* чистая дисконтированная стоимость (ЧДД);
* срок окупаемости инвестиций (Ток);
* рентабельность инвестиций (Ри).

Суммарная прибыль за год рассчитывается, как прибыль от продажи одной копии на количество копий:

|  |  |  |
| --- | --- | --- |
|  |  | (7.13) |

где Пед – прибыль от продажи одной копии, руб;

N – количество копий за год.

Прибыль от продажи программного продукта по годам, рассчитанная по формуле 7.13 составляет:

В ходе реализации ПП чистая прибыль в конечном итоге возмещает расходы на разработку. Однако полученные при этом суммы результатов (прибыли) и затрат по годам следует привести к единому времени, а именно началу расчётного года. Для этого необходимо использовать дисконтирование путем умножения соответствующих результатов и затрат на коэффициент дисконтирования соответствующего года t, который определяется по формуле 7.14:

|  |  |
| --- | --- |
|  | (7.14) |

где – норма дисконта (в долях единиц), равная или больше средней процентной ставки по банковским депозитам действующей на момент осуществления расчетов, равная 0.14;

– порядковый номер года периода реализации инвестиционного проекта (предполагаемый период использования разрабатываемого ПО пользователем и время на разработку).

Чистый дисконтированный доход рассчитывается по формуле 7.15:

|  |  |
| --- | --- |
| , | (7.15) |

где – расчетный период, лет;

– результат (экономический эффект), полученный в году t, руб.;

– затраты (инвестиции в разработку программного обеспечения) в году t, руб.

Сведем данные расчета экономического эффекта в таблицу 7.3.

Таблица 7.3 – Расчет эффективности инвестиционного проекта по разработке

программного обеспечения

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Показатели | Ед. изм. | Годы | | | |
| 2017 | 2018 | 2019 | 2020 |
| Результаты: | | | | | |
| Экономический эффект | руб. | 902,53 | 1504,21 | 1203,37 | 902,53 |
| Коэффициент дисконтирования | доли ед. | 1 | 0,88 | 0,77 | 0,68 |
| Дисконтированный результат | руб. | 902,53 | 1319,48 | 925,95 | 609,18 |
| Затраты: | | | | | |
| Затраты на разработку | руб. | 2987,37 |  |  |  |
| Дисконтированные вложения | руб. | 2987,37 |  |  |  |
| Экономический эффект: | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 |

*Продолжение таблицы 7.3*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Чистый дисконтированный доход по годам | руб. | -2084,84 | 1319,48 | 925,95 | 609,18 |
| Чистый дисконтированный нарастающим итогом | руб. | -2084,84 | -765,36 | 160,59 | 769,77 |

Рентабельность инвестиций вычисляется по формуле 7.16:

|  |  |  |
| --- | --- | --- |
|  |  | (7.16) |

где ЧПср – среднегодовая величина чистой прибыли за расчетный период и высчитывается по формуле 7.17.

|  |  |  |
| --- | --- | --- |
|  |  | (7.17) |

Итого рентабельность инвестиций:

Чистый дисконтированный доход больше нуля, то есть проект эффективен, а значит инвестиции в разработку данного ПО экономически целесообразны.

Таким образом, чистый дисконтированный доход за четыре года работы программного продукта составит 769,77 рублей. Суммарный дисконтированный результат превысит дисконтированную сумму инвестиций в 2019 году, то есть срок окупаемости проекта равен трём годам. Рентабельность инвестиций составляет 38%. Исходя из вышеперечисленных показателей можно сделать вывод, что проект является прибыльным и затраты на его разработку целесообразны.

# ЗАКЛЮЧЕНИЕ

В Результате

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Wikipedia [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Цифровая звуковая рабочая станция.
2. Wikipedia [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Плагин>.
3. Wikipedia [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Virtual_Studio_Technology>.
4. Wikipedia [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Октавер>.
5. Лайонс, Р. Цифровая обработка сигналов: Второе издание/ Лайонс Р. – Пер с англ. – М. : ООО «Бином-Пресс», 2006 г. – 656 с.
6. Cooley J. W., Tukey J. W. An algorithm for the machine calculation of complex Fourier series // Mathematics of Computation. 1965. V. 19. No. 90. P. 297-301.
7. DSPLIB [Электронный ресурс]. – Режим доступа: <http://ru.dsplib.org/content/fft_dec_in_time.html>.
8. Wikipedia [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Дилэй>.
9. Audio Artillery [Электронный ресурс]. – Режим доступа: <http://audioartillery.com/projects/tonecore_dsp_dev_kit_guide>.
10. Wikipedia [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Реверберация>.
11. Open Source Acoustic Design [Электронный ресурс] : Algorithmic Reverberation. – Режим доступа: http://arqen.com/wp-content/docs/Hybrid-Convolution-Algorithmic-Reverb.pdf.
12. Christianfloisand wordpress [Электронный ресурс]. – Режим доступа: <https://christianfloisand.wordpress.com/2012/10/18/algorithmic-reverbs-the-moorer-design>/.
13. Wikipedia [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Juce>.
14. Juce [Электронный ресурс]. – Режим доступа: https://www.juce.com/doc/classes.
15. Juce [Электронный ресурс]. – Режим доступа: <https://www.juce.com/doc/tutorial_create_projucer_basic_plugin>.
16. GitHub [Электронный ресурс]. – Режим доступа: https://github.com/.

# ПРИЛОЖЕНИЕ А

(обязательное)

Исходный код некоторых алгоритмов программного модуля

# ПРИЛОЖЕНИЕ Б

(обязательное)

Спецификация программного дипломного проекта

# ПРИЛОЖЕНИЕ В

(обязательное)

Ведомость документов