СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc479940284)

[**1** ОБЗОР ЛИТЕРАТУРЫ 8](#_Toc479940285)

[**1.1** Цифровые аудио-станции и плагины 8](#_Toc479940286)

[**1.2** Octaver 10](#_Toc479940287)

[**1.3** Delay 13](#_Toc479940288)

[**1.4** Reverb 14](#_Toc479940289)

[**2** СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ 16](#_Toc479940290)

[**2.1** Структура программного модуля 16](#_Toc479940291)

[**2.2** Выбор программных средств 17](#_Toc479940292)

[**3** ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 21](#_Toc479940293)

[**4** РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 22](#_Toc479940294)

[**5** ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ 23](#_Toc479940295)

[**6** РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 24](#_Toc479940296)

[**7** ЭКОНОМИЧЕСКАЯ ЧАСТЬ 25](#_Toc479940297)

[ЗАКЛЮЧЕНИЕ 26](#_Toc479940298)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc479940299)

[ПРИЛОЖЕНИЕ А 28](#_Toc479940300)

[ПРИЛОЖЕНИЕ Б 29](#_Toc479940301)

[ПРИЛОЖЕНИЕ В 30](#_Toc479940302)

# 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.

# **1** ОБЗОР ЛИТЕРАТУРЫ

## **1.1** 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.

## **1.2** 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].

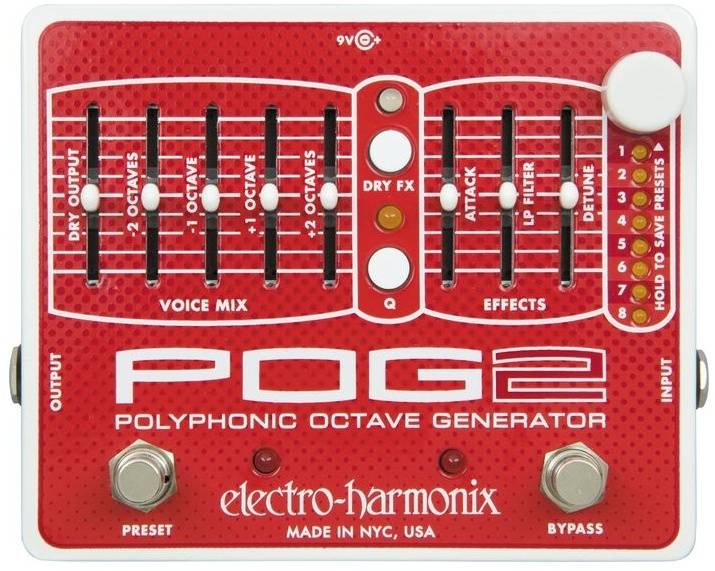


Рисунок 1.1 – Electro-Harmonix POG 2.

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).

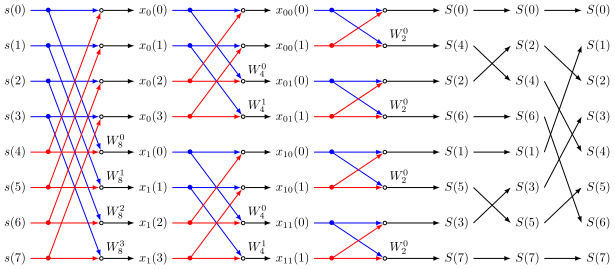


Рисунок 1.2 – Полный граф алгоритма БПФ с прореживанием по частоте для .

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) |
|  |  |

## **1.3** 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.

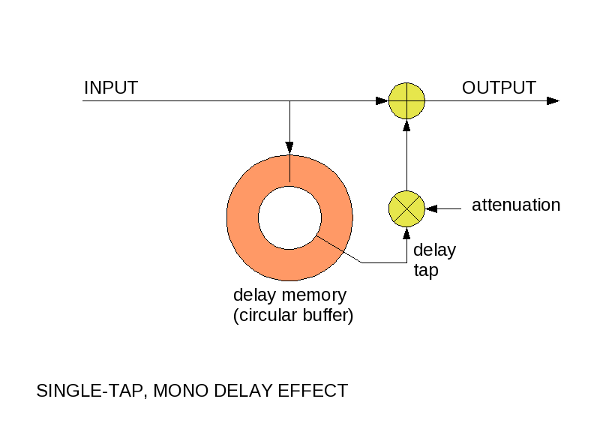


Рисунок 1.3 Схема простого эффекта delay.

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, задающимся хост-программой, а также с возможностью выбора режима reverse.

## **1.4** 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.

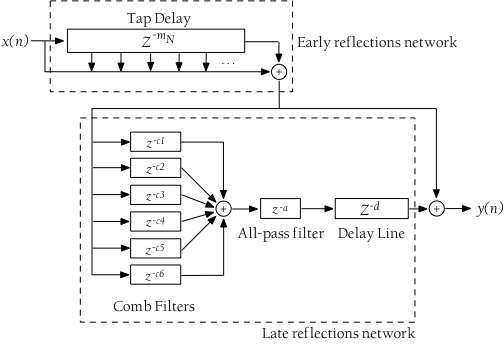


Рисунок 1.4 – Структурная схема ревербератора Мурера.

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.

# **2** 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.

## **2.1** 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.

## **2.2** 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.

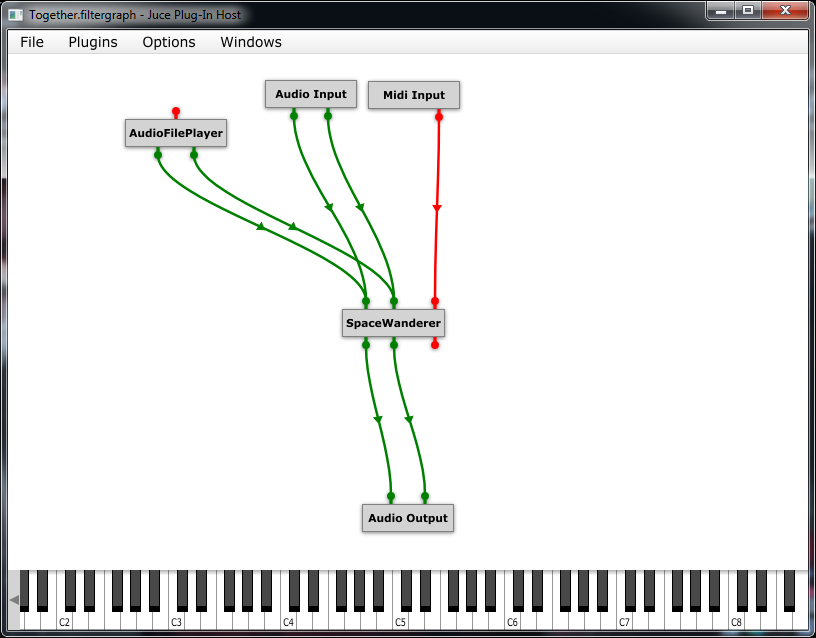


Рисунок 2.1 – Интерфейс хост-программы Juce.

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.

# **3** ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

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.

# **4** 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.

# **5** 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.

# **6** РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

В руководстве пользователя дается описание работы с программой. Указываются требования к аппаратному (процессор, объем ОЗУ, объем дискового пространства и так далее) и программному (ОС, необходимое ПО, драйверы и так далее) обеспечению. Описывается процесс инсталляции с указанием каталогов, ключей реестра, конфигурационных файлов и так далее. Также описывается пользовательский интерфейс с указанием элементов управления (пунктов меню, кнопок, закладок и так далее), режимов работы и последовательности действий. Здесь могут приводиться скриншоты работы программы.

# **7** ЭКОНОМИЧЕСКАЯ ЧАСТЬ

Экономика

# 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