

МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
"ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ"

Факультет компьютерных наук

Кафедра программирования и информационных технологий

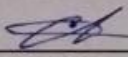
Разработка программного цифрового синтезатора звука

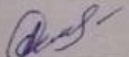
ВКР Бакалаврская работа

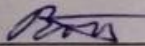
09.03.04 Программная инженерия

Информационные системы и сетевые технологии

Допущено к защите в ГЭК

Зав. кафедрой  С.Д. Махортов, д.ф.-м.н, доцент \_\_\_\_\_.20\_\_

Обучающийся  Н.А. Селиверстов, 4 курс, д/о

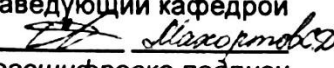
Руководитель  А.А. Вахтин, к.ф.-м.н, доцент

Воронеж 2022

МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет компьютерных наук

Кафедра программирования и информационных технологий


УТВЕРЖДАЮ  
заведующий кафедрой  
  
подпись, расшифровка подписи  
\_\_\_.2022

**ЗАДАНИЕ  
НА ВЫПОЛНЕНИЕ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ  
ОБУЧАЮЩЕГОСЯ ВВЕДЕНСКОГО АРТЕМА АЛЕКСАНДРОВИЧА**

1. Тема работы Разработка цифрового программного синтезатора звука, утверждена решением ученого совета ФКИ факультета от \_\_.\_\_.20\_\_
2. Направление подготовки / специальность 09.03.04 Программная инженерия
3. Срок сдачи законченной работы \_\_.\_\_.20\_\_
4. Календарный план: (строится в соответствии со структурой ВКР)

№	Структура ВКР	Сроки выполнения	Примечание
1	Введение	20.02.2022	
2	1 Постановка задачи	20.02.2022	
3	1.1 Цель работы	20.02.2022	
4	1.2 Этапы работы	20.02.2022	
4	2 Анализ предметной области	25.03.2022	
5	2.1 Обзор существующих решений	15.03.2022	
6	2.2 Обзор методов синтеза звука	20.03.2022	
7	2.3 Описание компонентов синтеза звука Синтезатора	25.03.2022	
8	2.4 Схема синтеза в Синтезаторе	25.03.2022	
9	3 Технические требования	10.04.2022	
10	4 Реализация	25.05.2022	
11	4.1 Средства реализации	20.04.2022	
12	4.2 Архитектура	30.04.2022	
13	4.3 Реализация компонентов синтеза звука	25.05.2022	
14	4.4 Реализация пользовательского интерфейса	25.05.2022	
15	5 Перспективы развития	30.05.2022	
16	Заключение	30.05.2022	
17	Список использованных источников	30.05.2022	
18	Приложения	30.05.2022	

Обучающийся



Селиверстов Н. А.

Руководитель



Вахтин А. А.

## Реферат

Бакалаврская работа 51 с., 19 рис., 1 прил.

Ключевые слова: СИНТЕЗАТОР, СИНТЕЗ ЗВУКА, ГЕНЕРАЦИЯ СИГНАЛОВ, ОБРАБОТКА СИГНАЛОВ, АЛГОРИТМ, PYTHON, ГРАФИЧЕСКИЙ ИНТЕРФЕЙС.

Объектом разработки является программный цифровой синтезатор звука.

Целью бакалаврской работы является разработка программного синтезатора – приложения, реализующего алгоритмы генерации и обработки звука в соответствии с параметрами, задаваемыми пользователем. Разработанное приложение может быть использовано музыкантами-любителями в качестве виртуальной творческой мастерской при изучении основ синтеза звука.

В процессе выполнения работы проведен анализ предметной области, включающий обзор существующих решений и методов синтеза звука, составлены схема и алгоритмы синтеза звука, сформулированы требования к программному продукту, спроектирована его архитектура.

В результате выполнения работы разработан программный синтезатор звука, удовлетворяющий поставленным требованиям и реализующий алгоритмы синтеза, составленные на этапе анализа.

# Содержание

Содержание .....	4
Введение .....	7
1 Постановка задачи .....	9
1.1 Цель работы .....	9
1.2 Этапы работы .....	9
2 Анализ предметной области .....	10
2.1 Терминология .....	10
2.2 Обзор существующих решений .....	10
2.2.1 Плагины реального времени .....	11
2.2.2 Standalone-приложения .....	12
2.2.3 Web-приложения .....	13
2.3 Обзор методов синтеза звука .....	14
2.3.1 Аддитивный синтез .....	14
2.3.2 Формантный синтез .....	14
2.3.3 Частотная модуляция .....	15
2.3.4 Субтрактивный синтез .....	15
2.3.5 Синтез на основе семплов .....	15
2.4 Компоненты синтеза звука в Синтезаторе .....	16
2.4.1 Осциллятор .....	16
2.4.2 Модулятор .....	17
2.4.2.1 Low-frequency oscillator .....	17
2.4.2.2 ADSR-оггибающая .....	18
2.4.3 Модификатор .....	19

2.4.3.1 Детюн .....	19
2.4.3.2 Панорама .....	19
2.4.3.3 Сумматор.....	20
2.4.4 Модуляция .....	21
2.4.4.1 Амплитудная модуляция осциллятора LFO.....	21
2.4.4.2 Фазовая модуляция осциллятора LFO .....	22
2.4.4.3 Частотная модуляция осциллятора LFO.....	23
2.4.4.4 Амплитудная модуляция осциллятора ADSR-огибающей.....	25
2.4.4.5 Модуляция коэффициента панорамы LFO.....	25
2.4.5 Метод генерации сигнала осциллятором в Синтезаторе .....	26
2.4.6 Компонентная схема синтеза звука в Синтезаторе .....	28
3 Технические требования к Синтезатору .....	32
4 Реализация Синтезатора .....	33
4.1 Средства реализации.....	33
4.1.1 Язык программирования Python.....	33
4.1.2 Библиотеки NumPy и SciPy.....	33
4.1.3 Библиотека Numba .....	34
4.1.4 Библиотеки RtMidi и PyAudio .....	34
4.1.5 Библиотека PyQt5 и среда разработки Qt Designer .....	34
4.1.6 Система контроля версий Git и репозиторий GitHub.....	35
4.2 Архитектура.....	35
4.2.1 Классы .....	35
4.3 Реализация компонентов синтеза звука.....	37
4.3.1 Реализация осцилляторов.....	37
4.3.2 Реализация модуляции осциллятора LFO .....	39

4.3.3 Реализация модуляции осциллятора ADSR-огибающей .....	39
4.3.4 Реализация детюна .....	39
4.3.5 Реализация панорамы .....	40
4.3.6 Реализация сумматора .....	40
4.4 Реализация контроллера .....	40
4.5 Реализация интерфейса.....	41
4.5.1 Интерфейс управления осциллятором.....	42
4.5.2 Интерфейс управления модуляцией осциллятора LFO .....	43
4.5.3 Интерфейс управления модуляцией ADSR-огибающей.....	44
4.5.4 Интерфейс управления детюном.....	44
4.5.5 Интерфейс управления сумматором .....	45
4.5.6 Интерфейс управления панорамой.....	46
4.5.7 Интерфейс управления уровнем сигнала .....	46
5 Перспективы развития Синтезатора .....	48
Заключение .....	49
Список источников .....	50
Приложение А .....	52

## **Введение**

Музыкальная индустрия является значимой частью современной индустрии развлечений. Согласно отчету Международной федерации звукозаписывающей индустрии (IFPI) ее глобальная выручка за 2020 год составила \$21,6 млрд., при этом 62% этой суммы было обеспечено стриминговыми сервисами, которые среди прочих преимуществ позволяют значительно облегчить дистрибуцию музыки для начинающих и независимых исполнителей [1]. Кроме того, стриминговые сервисы предоставляют исполнителям доступ к широкой аудитории, а также возможность продвижения музыки благодаря алгоритмам рекомендаций.

Одновременно с упрощением музыкальной дистрибуции наблюдается упрощение процесса производства музыки: появление большого количества электронных и бумажных ресурсов и курсов, обучающих игре на музыкальных инструментах и теории музыки, развитие интернет-площадок, предоставляющих возможность заказывать сочинение мелодий или обработку записанных композиций, увеличение доступности музыкального оборудования благодаря развитию торговых сетей. Значительную роль в этом процессе играет увеличение производительности персональных компьютеров, которое совместно с развитием программного обеспечения, позволяет людям самостоятельно производить запись и обработку музыки.

Упрощение дистрибуции, а также доступа к аудитории и производства музыки на домашних компьютерах делает разработку программного обеспечения в области музыки актуальной и перспективной. Одним из ее направлений является разработка программных средств для синтеза звуков.

Аналоговые и цифровые синтезаторы, появившись в прошлом веке, заняли место одного из главнейших инструментов в современной музыке благодаря тому, что они позволяют людям создавать уникальные звуки и достигать оригинального и недостижимого при помощи других инструментов

звучания [2]. Однако, несмотря на появление простых по устройству, но обладающих небольшой функциональностью домашних моделей, стоимость которых значительно ниже профессиональных, аппаратные синтезаторы остаются дорогими инструментами, и не каждый музыкант может позволить себе их приобретение. В связи с этим, а также причинами, перечисленными выше, все большую популярность обретают программные цифровые синтезаторы, позволяющие получить доступ к функциональности аппаратных моделей на персональных компьютерах и даже смартфонах.



## **1 Постановка задачи**

### **1.1 Цель работы**

Целью бакалаврской работы является разработка программного цифрового синтезатора (далее – Синтезатор) – приложения, реализующего алгоритмы генерации и обработки звука в соответствии с параметрами, задаваемыми пользователем. Целевой аудиторией Синтезатора являются музыканты-любители, начинающие изучение основ синтеза звука, для которых он станет виртуальной творческой мастерской.

### **1.2 Этапы работы**

Для достижения поставленной цели выполнены следующие этапы:

- анализ предметной области:
  - обзор существующих программных синтезаторов,
  - обзор методов синтеза звука,
  - обзор компонентов, используемых в процессе синтеза звука;
- формулирование технических требований к разрабатываемому приложению;
- реализация Синтезатора:
  - проектирование архитектуры,
  - выбор программных средств реализации,
  - разработка компонентов, осуществляющих синтез звука,
  - разработка графического интерфейса пользователя.

Результаты выполнения этапов работы оформлены в виде составных частей работы.

## **2 Анализ предметной области**

В целях формирования требований к разрабатываемому Синтезатору, а также составления алгоритмов синтеза проведен анализ предметной области, в ходе которого рассмотрены существующие решения, выявлены их преимущества и недостатки, а также изучены основные методы синтеза звука.

### **2.1 Терминология**

- Звук – в рамках данной работы звук рассматривается как цифровой сигнал;
- DAW (eng.: Digital Audio Workstation) – программная среда, предназначенная для записи, редактирования и воспроизведения цифрового звука;
- MIDI – стандарт цифровой звукозаписи, описывающий формат обмена данными между электронными музыкальными инструментами;
- VST – формат музыкального плагина реального времени;
- Частота дискретизации – частота, с которой получаются значения сигнала.

### **2.2 Обзор существующих решений**

Существует большое количество различных программных синтезаторов, которые можно разделить на три вида по способу программной реализации [2]:

- плагины реального времени для подключения к DAW;
- standalone-приложения;
- web-приложения.

Каждый из перечисленных видов обладает определенными преимуществами и недостатками.

Алгоритмы генерации и обработки сигналов, лежащие в основе синтеза звуков, не зависят от вида программной реализации синтезатора.

### **2.2.1 Плагины реального времени**

Программные синтезаторы, выполненные в виде плагинов реального времени, представляют собой библиотеки, динамически подключаемые к DAW. Таким образом работа с этими синтезаторами осуществляется из запущенной цифровой станции.

Основным преимуществом плагинов является возможность подключения синтезатора к DAW для записи звуковых сигналов в них напрямую.

Основной недостаток плагинов заключается в необходимости установки, запуска и подключения DAW для начала работы с синтезатором. Кроме того, несмотря на то, что большинство плагинов поддерживают работу во всех популярных DAW, могут возникать ошибки из-за несовместимости плагина и станции.

Примером программного синтезатора, выполненного в виде плагина, является VST версия синтезатора Analog Lab V, разработанного компанией Arturia. Данный синтезатор обладает большим количеством пресетов, однако для получения возможности полноценной настройки параметров синтеза необходимо приобрести полную версию синтезатора за 199 евро, что делает его недоступным для начинающих музыкантов. Снимок экрана синтезатора Analog Lab V приведен на рисунке 1.

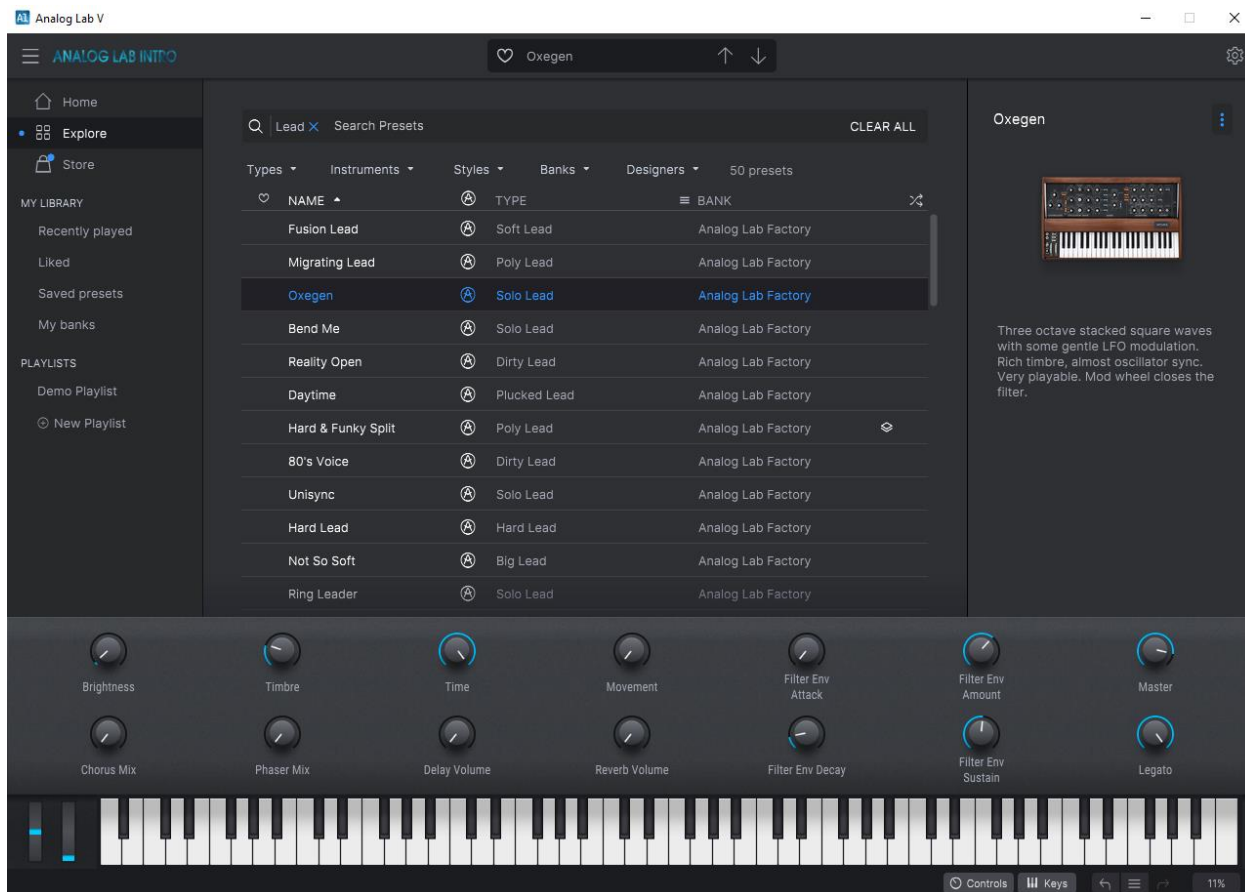


Рисунок 1 - Снимок экрана синтезатора Analog Lab V

### 2.2.2 Standalone-приложения

Программные синтезаторы, выполненные в виде standalone-приложений, способны работать независимо от DAW, что улучшает их доступность, но приводит к невозможности записи звука напрямую в рабочей станции.

Данный вид программной реализации позволяет уменьшить трудозатраты при разработке Синтезатора, а также обеспечивает приемлемый баланс между доступностью и производительностью программного продукта, соответствуя основным требованиям пользователей к синтезаторам [2]. Вышеуказанные факторы обусловили использование вида standalone-приложения в целях данной работы.

Примером программного синтезатора, выполненного в виде standalone-приложения, является standalone версия синтезатора Analog Lab V, которая

обладает интерфейсом и недостатками аналогичными VST версии данного синтезатора.

### 2.2.3 Web-приложения

Программные синтезаторы, выполненные в виде web-приложений, позволяют пользователю запускать синтезатор через Интернет-браузер, благодаря чему обладают наилучшей доступностью. Недостатками таких синтезаторов являются невозможность записи звука в DAW, а также зависимость производительности от скорости и качества Интернет-соединения.

Примером синтезатора, выполненного в виде web-приложения, является синтезатор Playground, разработанный компанией Ableton. Минусом данного синтезатора является довольно примитивная схема синтеза, ограничивающая количество получаемых при помощи него звуков. Снимок экрана синтезатора Playground приведен на рисунке 2.

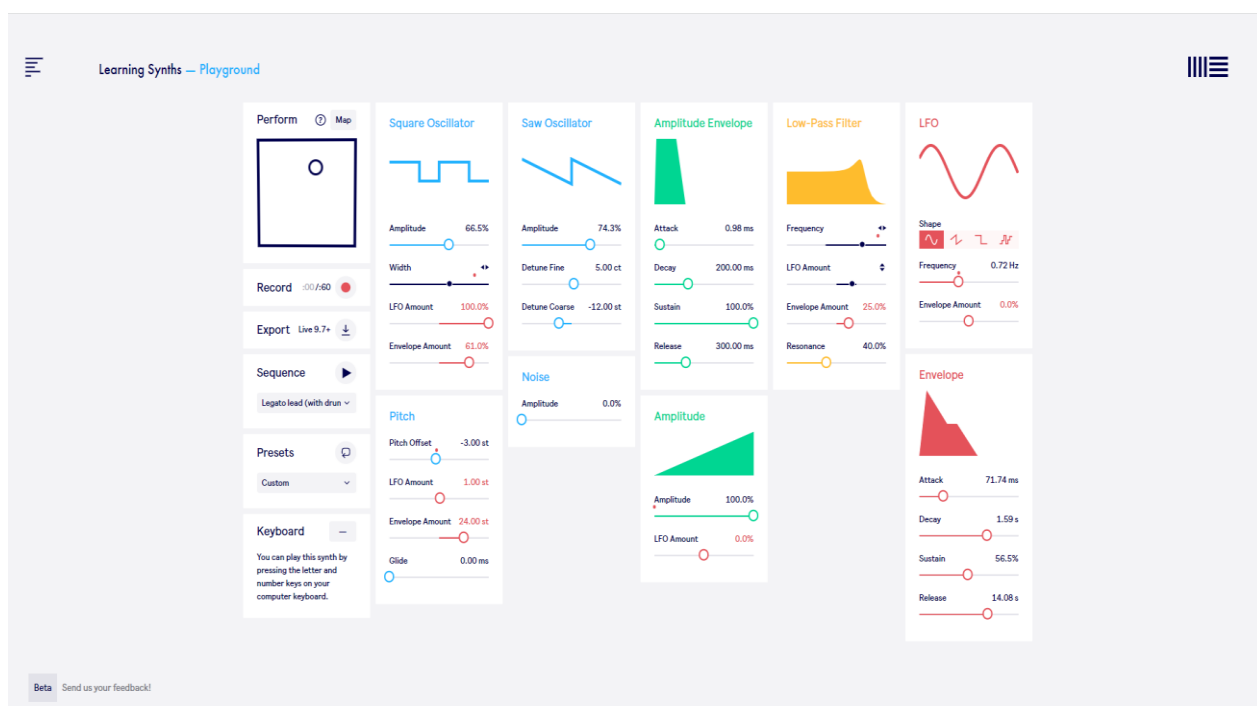


Рисунок 2 - Снимок экрана синтезатора Playground

## **2.3 Обзор методов синтеза звука**

Существует множество различных методов синтеза звуковых сигналов. Далее рассмотрены аспекты реализации и особенности получаемых звуков наиболее популярных методов синтеза.

### **2.3.1 Аддитивный синтез**

Аддитивный синтез основан на теории Фурье, которая описывает получение сигнала, обладающего сложным тембром, то есть сложным спектром, путем сложения простых синусоидальных тонов. Данный метод синтеза позволяет моделировать звучание живых инструментов благодаря возможности создания такого набора простых синусоидальных тонов, амплитуда которых изменяется в зависимости от времени, что их сумма будет представлять собой звуковой сигнал идентичный сигналу, воспроизводимому инструментом [3, 7].

Для реализации аддитивного синтеза может использоваться обратное преобразование Фурье. Данный способ обычно применяется в том случае, когда известен спектр необходимого звукового сигнала.

### **2.3.2 Формантный синтез**

Формантный синтез основан на получении звукового сигнала, напоминающего речь человека. Спектр такого сигнала содержит форманты – пики АЧХ, получаемые как резонанс определенных обертонов. Частоты, на которых выделяются форманты, определяют фонему и являются для нее константами, например, для фонемы «о» первая форманта выделяется на частоте 275 Гц, вторая – 850 Гц, третья – 2400 Гц.

Для получения сигнала при помощи метода формантного синтеза используется набор фильтров, который преобразует входной сигнал, выделяя в нем форманты на заданных частотах [4, 7]. Чем больше фильтров

применяется к сигналу, тем ближе полученный звук к фонеме. Входной сигнал для данного метода синтеза должен обладать богатым спектром.

### **2.3.3 Частотная модуляция**

Синтез методом частотной модуляции был разработан Джоном Чоунингом в 1973 году и реализован компанией Yamaha в 1983 году. [2] Он заключается в получении сложного сигнала путем модуляции частоты одних генераторов значениями амплитуды других [5, 7]. Данный вид синтеза позволяет получать необычные сигналы, а синтезаторы, реализующие его, предоставляют несколько вариантов последовательностей модуляций, каждый из которых воспроизводит уникальный звук.

### **2.3.4 Субтрактивный синтез**

Субтрактивный синтез основан на ослаблении участков спектра исходного сигнала. Данный метод синтеза начал широко применяться в аналоговых синтезаторах 60-х и 70-х годов прошлого века и продолжает использоваться в настоящее время. Для его реализации исходный сигнал пропускается через различные частотные фильтры, которые усиливают или ослабляют заданные участки его спектра [6, 7].

### **2.3.5 Синтез на основе семплов**

Данный метод синтеза объединяет множество методов, которые при генерации звуковых сигналов используют записанные в память семплы в целях сокращения вычислений, а также генерации изначально сложных сигналов. Например, таблично-волновой синтез основан на циклическом воспроизведении согласно заданным параметрам семплов, записанных в таблично-волновую таблицу и представляющих собой значения участка некоторого периодического сигнала [7].

По результатам проведенного анализа принято решение об использовании элементов таблично-волнового, аддитивного и субтрактивного методов синтеза звука, как наиболее эффективных и достаточных для выполнения поставленной задачи.

## **2.4 Компоненты синтеза звука в Синтезаторе**

Для реализации методов синтеза звука, а также его обработки применяются различные компоненты, выполняющие определенные функции по генерации и обработке сигналов. Совокупность компонентов синтеза звука, а также правил их взаимодействия определяет порядок преобразований, через которые проходит сигнал. Пользователь может настраивать параметры компонентов и их взаимодействий, что позволяет получить большое количество уникальных звуков.

Ниже рассматриваются компоненты синтеза звука, реализованные в Синтезаторе, а также приводится схема их взаимодействия.

### **2.4.1 Осциллятор**

Осциллятор — это компонент, генерирующий простейший периодический сигнал. В синтезаторе реализована возможность генерации осциллятором сигналов четырех видов:

- синусоидальный;
- пилообразный;
- квадратный;
- треугольный.

Графики всех видов сигналов приведены на рисунке 3.



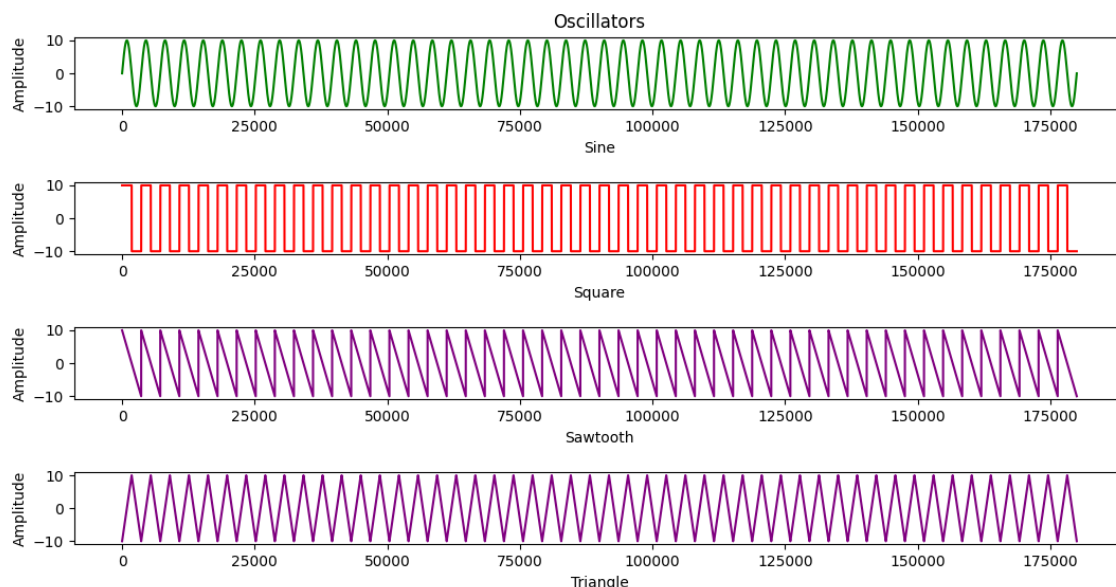


Рисунок 3 - Виды сигналов, генерируемых осцилляторами

В Синтезаторе реализована возможность одновременной работы двух осцилляторов.

## 2.4.2 Модулятор

Модулятор — это компонент, генерирующий собственную последовательность значений, которые в дальнейшем используются для изменения параметра определенного компонента, например, частоты осциллятора.

В Синтезаторе реализованы два вида модуляторов: low-frequency oscillator (LFO) и ADSR-ограничивающая.

### 2.4.2.1 Low-frequency oscillator

LFO — это осциллятор, генерирующий сигналы с частотой ниже порога слышимости человека, то есть ниже 20 Гц [8]. В Синтезаторе амплитуда сигнала LFO равна единице.

### 2.4.2.2 ADSR-оггибающая

ADSR-оггибающая (eng.: Attack-Decay-Sustain-Release) – это функция, описывающая изменение параметра во времени. На оггибающей выделяется четыре участка, первые буквы английских названий которых формируют сокращение ADSR [7, 9]:

- атака (eng.: attack) – период начального нарастания значений от нуля до максимума;
- спад (eng.: decay) – период снижения значений после нарастания до уровня поддержки;
- поддержка (eng.: sustain) – постоянный уровень значений;
- затухание (eng.: release) – окончательное снижение значений до нуля.

Значения, которые может принимать ADSR-оггибающая, находятся в промежутке между 0 и 1, благодаря чему при модуляции параметра оггибающей достаточно умножить его значение в определенный момент времени на значение оггибающей в тот же момент.

Пример ADSR-оггибающей и модулированного ею сигнала приведен на рисунке 4.

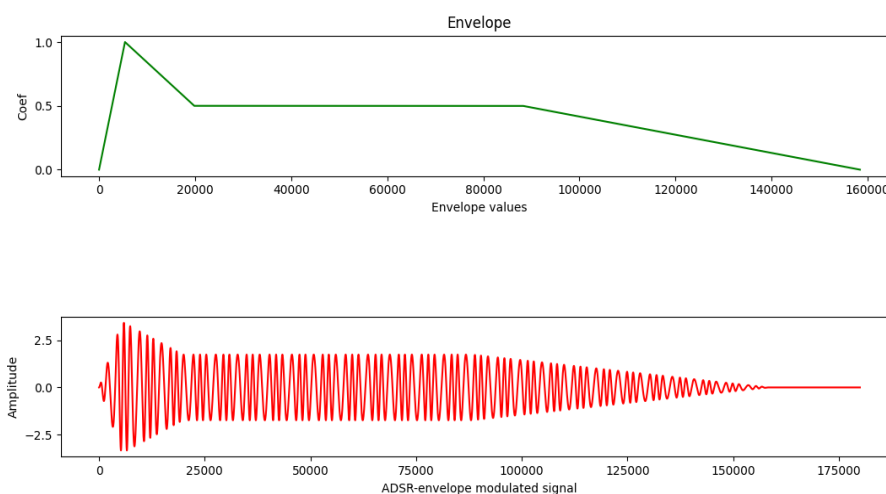


Рисунок 4 - ADSR-оггибающая и модулированный ею сигнал

### 2.4.3 Модификатор

Модификатор – это компонент, изменяющий значения сигнала, не генерируя для этого собственную последовательность значений.

В Синтезаторы реализованы следующие модификаторы: детюн, панорама, сумматор сигналов.

#### 2.4.3.1 Детюн

Детюн – это компонент, который позволяет получить массив, содержащий коэффициент смещения частоты для каждого из осцилляторов таким образом, чтобы среднее арифметическое смещенных частот равнялось передаваемой на вход частоте сигнала. Соотношение коэффициентов смещения частоты и несущей частоты осциллятора задается формулой (1):

$$f_c = \frac{f_c \sum_{i=0}^n (1 + k_i f_c)}{n}, \quad -k_{max} \leq k \leq k_{max} \quad (1)$$

где  $f_c$  – передаваемая на вход компонента частота;  $n$  – количество осцилляторов;  $k_i$  – коэффициент смещения для  $i$ -го осциллятора;  $k_{max}$  – коэффициент максимального смещения от передаваемой частоты задается пользователем,  $0 \leq k_{max} \leq 1$ .

Так как в синтезаторе реализована одновременная работа двух осцилляторов, то массив коэффициентов смещения состоит из двух элементов:  $k_1 = k_{max}$  и  $k_2 = -k_{max}$ .

#### 2.4.3.2 Панорама

Панорама – это компонент, который преобразует значение сигнала в массив из двух новых значений для левого и правого каналов звукового потока. Значения для левого и правого каналов рассчитываются по формулам (2) и (3):

$$l = (1 - k) * sample, \quad (2)$$

$$r = k * sample, \quad (3)$$

где  $l$  – значение сигнала в левом канале;  $r$  – значение сигнала в правом канале;  $sample$  – значение сигнала;  $k$  – коэффициент соотношения амплитуд левого и правого каналов, задаваемый пользователем,  $0 \leq k \leq 1$ . При  $k = 0.5$  амплитуды сигналов левого и правого каналов равны..

### 2.4.3.3 Сумматор

Сумматор – это компонент, который складывает значения всех осцилляторов в соответствии с задаваемыми пользователем долями каждого из сигнала в суммарном. Суммарное значение рассчитывается по формуле (4):

$$sum = \frac{\sum_{i=1}^n k_i * sample_i}{\sum_i k_i}, \quad 0 \leq k_i \leq 1 \quad (4)$$

где  $sum$  – суммарное значение всех сигналов в соответствии с их долями;  $k_i$  – коэффициент, определяющий долю  $i$ -го сигнала в суммарном сигнале, задается пользователем;  $sample_i$  – значение  $i$ -го сигнала;  $n$  – количество значений осцилляторов.

Пример работы сумматора по сложению двух сигналов, первому из которых соответствует коэффициент  $k_1 = 0.5$  и второму –  $k_2 = 1$ , приведен на рисунке 5.

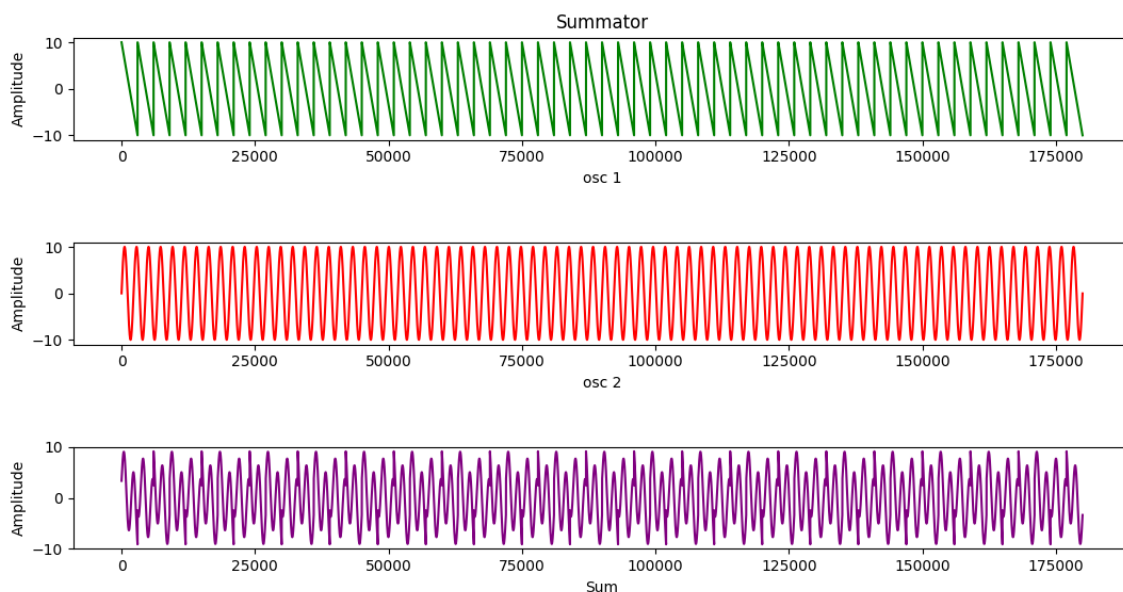


Рисунок 5 - Результат работы сумматора

## 2.4.4 Модуляция

Модуляция — это компонент, который изменяет определенные параметры компонентов в зависимости от значения модулятора.

В Синтезаторе реализованы следующие виды модуляций: амплитудная модуляция осциллятора LFO, фазовая модуляция осциллятора LFO, частотная модуляция осциллятора LFO, амплитудная модуляция осциллятора ADSR-огибающей, модуляция коэффициента панорамы LFO.

### 2.4.4.1 Амплитудная модуляция осциллятора LFO

Амплитудная модуляция осциллятора LFO (далее — AM LFO) — это изменение значения, генерируемого осциллятором в определенный момент времени, в зависимости от значения, генерируемого LFO в то же время, а также индекса модуляции. Амплитудная модуляция задается формулой (5) [10]:

$$am(t) = (1 + k * lfo(t)) * osc(t), \quad (5)$$

где  $am(t)$  – значение амплитудно-модулированного осциллятора, получаемое в момент времени  $t$ ;  $k$  – индекс модуляции, задаваемый пользователем;  $t$  – время, в которое получают значения осциллятора и LFO;  $lfo(t)$  – значение LFO, генерируемое в момент времени  $t$ ;  $osc(t)$  – значение осциллятора, генерируемое в момент времени  $t$ .

#### 2.4.4.2 Фазовая модуляция осциллятора LFO

Фазовая модуляция осциллятора LFO (далее – ФМ LFO) – это получение измененного значения, генерируемого осциллятором в определенный момент времени, за счет прибавления к полной фазе сигнала произведения индекса модуляции и значения, генерируемого LFO в то же время. Ниже приведено получение формулы, при помощи которой реализована фазовая модуляция осциллятора в Синтезаторе.

Пусть значение осциллятора в момент времени  $t$  вычисляется по формуле (6):

$$osc(t) = F(2\pi f_{osc}t), \quad (6)$$

где  $osc(t)$  – значение осциллятора в момент времени  $t$ ;  $F$  – некоторая периодическая функция;  $t$  – время, в которое получают значения осциллятора и LFO;  $f_{osc}$  – частота осциллятора.

Тогда фазово-модулированный сигнал вычисляется по формуле (7) [11]:

$$pm(t) = F(2\pi f_{osc}t + k * lfo(t)), \quad (7)$$

где  $pm(t)$  – значение фазово-модулированного сигнала в момент времени  $t$ ;  $k$  – индекс модуляции, задаваемый пользователем;  $lfo(t)$  – значение LFO, генерируемое в момент времени  $t$ .

Таким образом, задача по нахождению значению фазово-модулированного сигнала в том случае, когда известны только значения

осциллятора в моменты времени  $t$ , а не формула, по которой они получаются, сводится к нахождению такого смещения времени  $\delta$ , что выполняется соотношение (8):

$$\text{osc}(t + \delta) = \text{pm}(t) = F(2\pi f_{\text{osc}}t + k * \text{lfo}(t)). \quad (8)$$

Зная, что верно соотношение (9):

$$\text{osc}(t + \delta) = F(2\pi f_{\text{osc}}(t + \delta)), \quad (9)$$

проведем преобразования для получения формулы фазы модулированного сигнала (10):

$$\begin{aligned} F(2\pi f_{\text{osc}}t + k * \text{lfo}(t)) &= F(2\pi f_{\text{osc}}(t + \delta)), \\ F(2\pi f_{\text{osc}}t + k * \text{lfo}(t)) &= F(2\pi f_{\text{osc}}t + 2\pi f_{\text{osc}} * \delta), \\ \Phi = k * \text{lfo}(t) &= 2\pi f_{\text{osc}} * \delta, \end{aligned} \quad (10)$$

где  $\Phi$  – фаза модулированного сигнала.

Таким образом, значение смещения времени  $\delta$  вычисляется по формуле (11):

$$\delta = \frac{k}{2\pi f_{\text{osc}}} * \text{lfo}(t). \quad (11)$$

Итак, значение фазово-модулированного сигнала вычисляется по формуле (12):

$$\text{pm}(t) = \text{osc}\left(t + \frac{k}{2\pi f_{\text{osc}}} * \text{lfo}(t)\right). \quad (12)$$

#### 2.4.4.3 Частотная модуляция осциллятора LFO

Частотная модуляция осциллятора LFO (далее – ЧМ LFO) – это получение измененного значения, генерируемого осциллятором в определенный момент времени, засчет прибавления к полной фазе сигнала

произведения индекса модуляции,  $2\pi$  и интеграла функции, согласно которой генерируются значения LFO для того же времени. Формула вычисления значения частотно-модулированного сигнала получается аналогично формуле вычисления фазово-модулированного сигнала.

Зная формулу для получения значения частотно-модулированного сигнала (13) [11, 12]:

$$fm(t) = F\left(2\pi f_{osc}t + 2\pi k \int_0^t lfo(\tau)d\tau\right), \quad (13)$$

где  $fm(t)$  – значение частотно-модулированного сигнала в момент времени  $t$ ;  $F$  – некоторая периодическая функция;  $f_{osc}$  – частота осциллятора;  $k$  – индекс модуляции, задаваемый пользователем;  $lfo(t)$  – значение LFO, генерируемое в момент времени  $t$ .

Необходимо найти такое смещение времени  $delta$ , что выполняется соотношение (14):

$$osc(t + delta) = fm(t) = F\left(2\pi f_{osc}t + 2\pi k \int_0^t lfo(\tau)d\tau\right), \quad (14)$$

где  $osc(t + delta)$  – значение осциллятора в момент времени  $t + delta$ ;  $t$  – время, в которое получают значения осциллятора и LFO.

Проведя преобразования получим формулу для вычисления значения смещения (15):

$$delta = \frac{k}{f_{osc}} \int_0^t lfo(\tau)d\tau. \quad (15)$$

Итак, значение частотно-модулированного сигнала вычисляется по формуле (16):

$$fm(t) = osc\left(t + \frac{k}{f_{osc}} \int_0^t lfo(\tau)d\tau\right). \quad (16)$$



График, иллюстрирующий получение значения частотно-модулированного сигнала, приведен на рисунке 6.

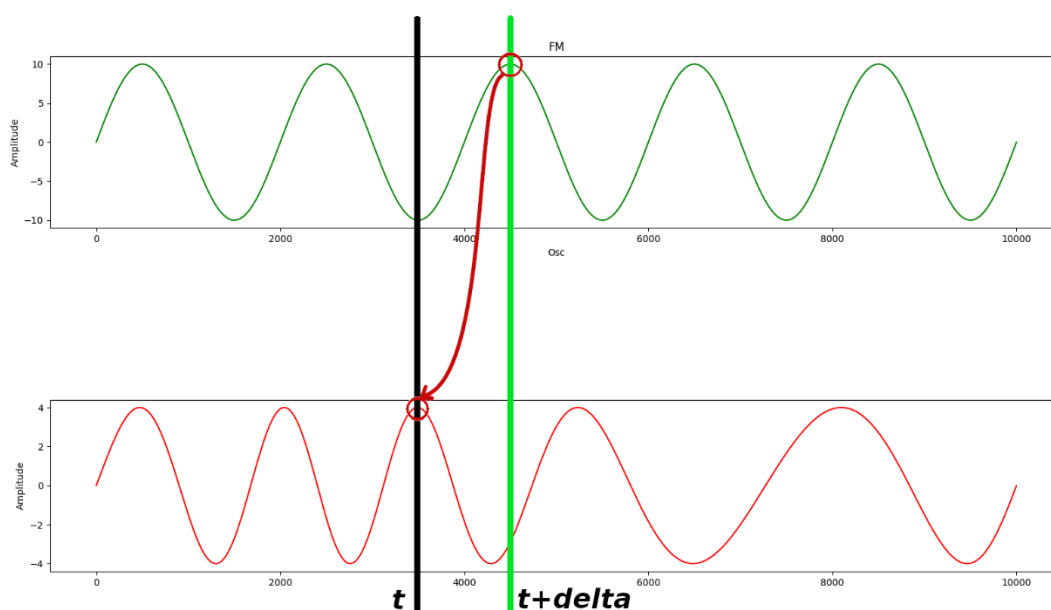


Рисунок 6 - получение значения частотно-модулированного сигнала

#### 2.4.4.4 Амплитудная модуляция осциллятора ADSR-огнивающей

Амплитудная модуляция осциллятора ADSR-огнивающей (AM ADSR) – это изменение значения, генерируемого осциллятором в определенный момент времени, получаемое путем его умножения на значение ADSR-огнивающей в тот же момент. AM ADSR задается следующей формулой (17):

$$am_{adsr}(t) = osc(t) * adsr(t), \quad (17)$$

где  $am_{adsr}(t)$  – значение модулированного сигнала в момент времени  $t$ ;  
 $osc(t)$  – значение осциллятора, генерируемое в момент времени  $t$ ;  
 $adsr(t)$  – значение ADSR-огнивающей, генерируемое в момент времени  $t$ .

#### 2.4.4.5 Модуляция коэффициента панорамы LFO

Модуляция коэффициента панорамы LFO – изменение коэффициента панорамы в зависимости от значения, генерируемого LFO в текущий момент времени. Модуляция коэффициента панорамы LFO задается формулой (18):

$$mod\_k(t) = lfo(t) * (k - 0.5) + 0.5, \quad 0 \leq k \leq 1 \quad (18)$$

где  $mod\_k(t)$  – модулированное значение коэффициента панорамы в момент времени  $t$ ;  $lfo(t)$  – значение LFO, генерируемое в момент времени  $t$ ;  $k$  – коэффициент панорамы, задаваемый пользователем.

Формула (18) позволяет получить значение коэффициента  $mod\_k$  в интервале между  $k$  и  $1 - k$ . При этом значения для правого и левого каналов вычисляются по формулам (19) и (20):

$$l = (1 - mod\_k) * sample, \quad (19)$$

$$r = mod\_k * sample, \quad (20)$$

где  $l$  – значение сигнала в левом канале;  $r$  – значение сигнала в правом канале;  $sample$  – значение сигнала.

Таким образом, со временем изменяется соотношение сигналов в каждом канале: пока уменьшается амплитуда сигнала в одном канале, в другом она увеличивается с той же скоростью.

#### 2.4.5 Метод генерации сигнала осциллятором в Синтезаторе

Можно выделить два основных метода генерации сигнала: Первый метод заключается в вычислении значения сигнала в заданный момент времени по формуле. Второй метод основан на предварительном вычислении всех возможных значений сигнала с последующим обращением к этим значениям в зависимости от заданного времени.

Второй метод выбран для осуществления генерации сигнала осцилляторами в Синтезаторе, благодаря следующим преимуществам:

- так как все возможные значения сигнала вычисляются предварительно, то при усложнении формулы вычисления сигнала не будет увеличиваться время генерации сигнала;

— данный метод позволяет задать любой набор значений, на основе которого будет генерироваться периодический сигнал. Таким образом, можно записать в память любой образец звука, например, инструмента или голоса, после чего осциллятор будет генерировать периодический сигнал на основе заданного образца. В качестве образца также может быть использован нарисованный пользователем график функции.

В Синтезаторе предварительно вычисляются значения одного периода сигнала частотой 1 Гц с заданной частотой дискретизации. Таким образом формируется массив всех возможных значений, содержащий количество элементов, равное частоте дискретизации. Для получения значения сигнала заданной частоты в заданный момент времени вычисляется индекс элемента массива по формуле (21):

$$index = [(f * t) \bmod sample\_rate], \quad (21)$$

где  $index$  – индекс элемента в массиве значений сигнала;  $f$  – частота сигнала;  $t$  – время, в которое получается значение сигнала;  $sample\_rate$  – частота дискретизации Синтезатора.

График, иллюстрирующий генерацию сигнала на основе предварительного вычисленных значений в Синтезаторе, приведен на рисунке 7.

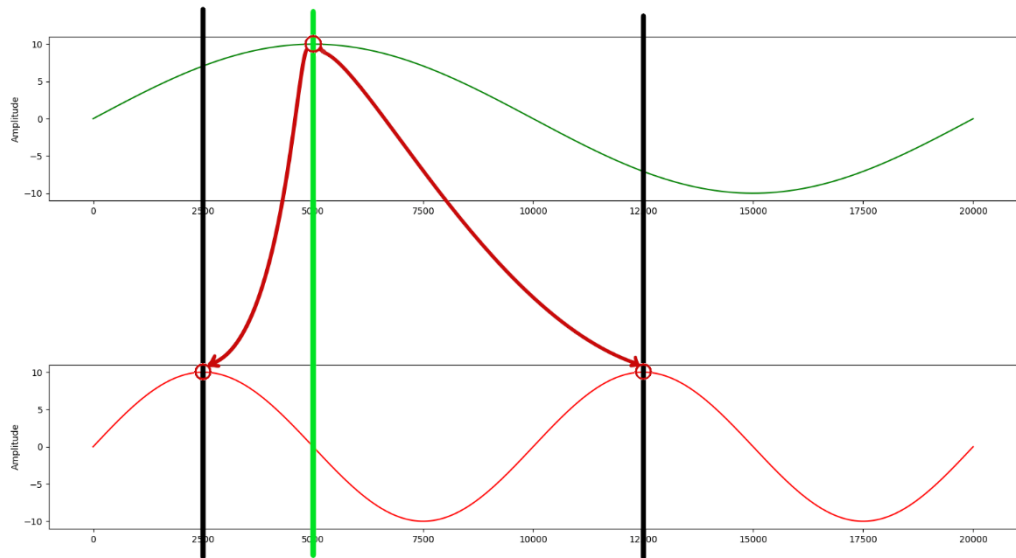


Рисунок 7 - генерация сигнала на основе предварительно вычисленных значений

Следует отметить, что при угловой модуляции сигнала, сгенерированного при помощи предварительно полученных значений, возникает сложность, вызванная невозможностью использования для модуляции классических формул (7) и (13), так как неизвестна периодическая функция  $F$  для вычисления сигнала. Для решения этой проблемы и были выведены формулы (12) и (16), позволяющие получать значение модулированных сигналов благодаря смещению времени.

#### 2.4.6 Компонентная схема синтеза звука в Синтезаторе

Схема взаимодействия компонентов синтеза звука приведена на рисунке 8.

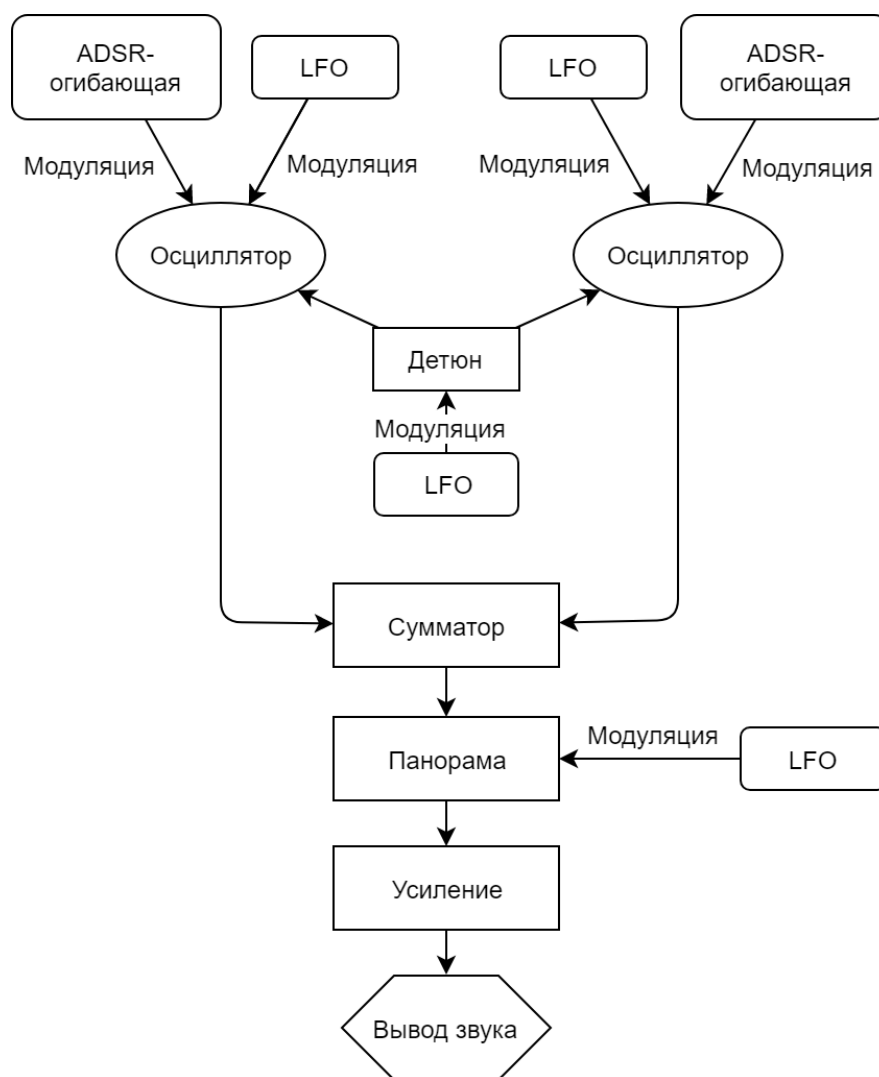


Рисунок 8 - Компонентная схема синтеза звука

Пользователь может управлять параметрами компонентов, а также может отключать все компоненты, кроме сумматора. Если компонент отключен, то он не генерирует и не преобразует сигнал.

Синтез звука осуществляется согласно следующему алгоритму:

1. пользователь вводит частоту сигнала, который должны генерировать осцилляторы;
2. на основании количества осцилляторов и значения коэффициента детюна вычисляются смещения частот для каждого осциллятора;
3. для каждого осциллятора:

- a. модифицируется значение частоты соответствующим значением смещения детюна,
  - b. вычисляются значения работающих модуляторов,
  - c. вычисляется модулированное LFO значение сигнала,
  - d. полученное значение модулируется ADSR-ограничивающей;
- 4. все полученные значения осцилляторов после модификации и модуляции складываются в сумматоре в соответствии заданными коэффициентами;
  - 5. в панораме вычисляются значения сигналов для каждого канала;
  - 6. синтезированный сигнал усиливается в зависимости от заданного пользователем значения и отправляется в буфер звукового потока.

Блок-схема алгоритма синтеза звука в Синтезаторе приведена на рисунке 9.

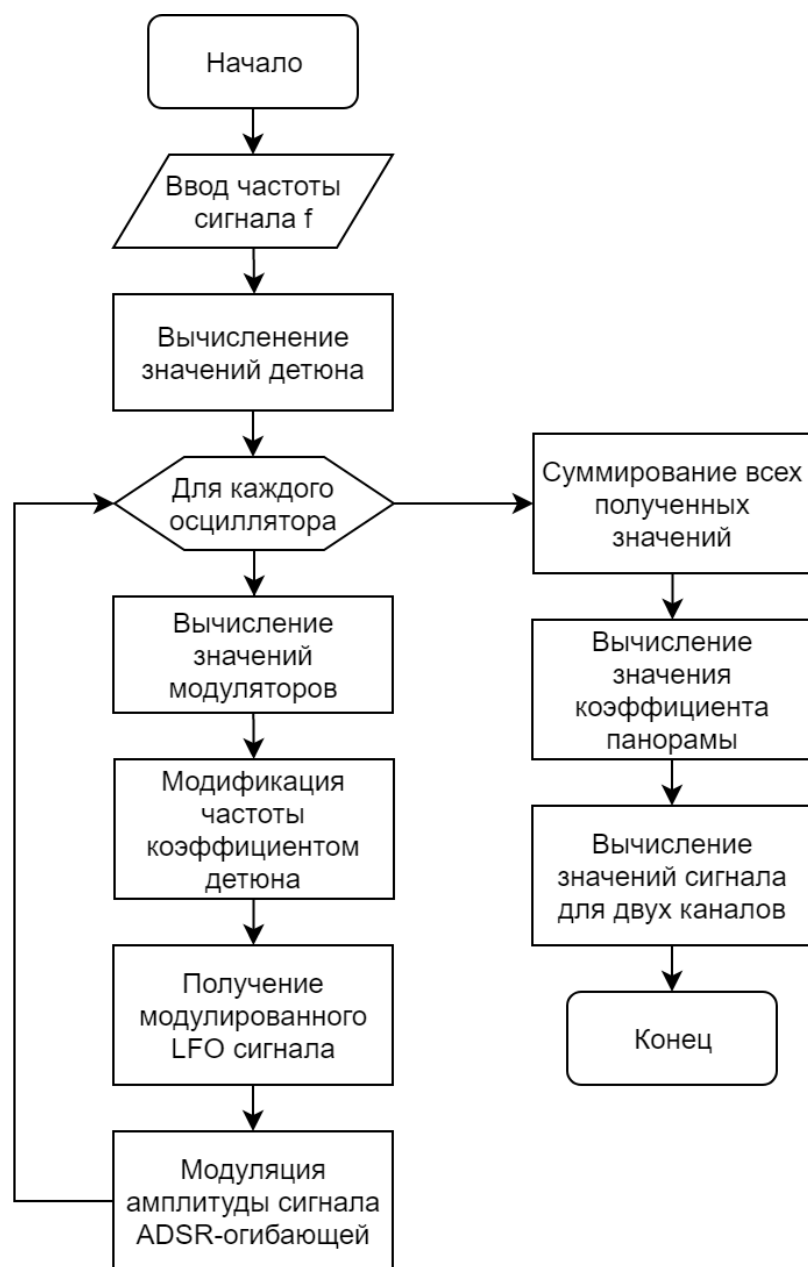


Рисунок 9 - Блок-схема алгоритма синтеза звука

### 3 Технические требования к Синтезатору

Синтезатор должен соответствовать следующим техническим требованиям:

- синтез звука должен осуществляться в реальном времени в соответствии с заданными пользователем параметрами компонентов;
- должна быть реализована поддержка возможности ввода высоты звука при помощи MIDI-клавиатуры;
- должен быть реализован графический интерфейс пользователя, который позволяет управлять параметрами компонентов синтеза звука.



## **4 Реализация Синтезатора**

### **4.1 Средства реализации**

Для реализации Синтезатора было использовано большое количество различных программных средств. Далее рассмотрены назначение и причины использования каждого из средств.

#### **4.1.1 Язык программирования Python**

Языком программирования для реализации Синтезатора выбран Python, так как он обладает рядом преимуществ:

- краткость и простота синтаксиса позволяют сосредоточиться на реализации математических алгоритмов Синтезатора, а также повысить эффективность разработки;
- удобство работы со списками и массивами также актуально при реализации алгоритмов Синтезатора, осуществляющих генерацию и преобразования цифровых сигналов;
- благодаря расширяемости, обусловленной популярностью и открытостью языка Python, для него существует большое количество различных по функциональности библиотек.

Основным недостатком языка Python в рамках поставленной задачи является низкая скорость его выполнения.

#### **4.1.2 Библиотеки NumPy и SciPy**

Библиотека NumPy предоставляет большое количество функций, выполняющих математические вычисления и преобразования. Например, в Синтезаторе она используется при формировании массивов значений интегрированного сигнала.

Библиотека SciPy основана на NumPy и предоставляет функции, выполняющие сложные математические вычисления. В Синтезаторе она используется при генерации пилообразного и треугольного сигналов.

#### **4.1.3 Библиотека Numba**

Библиотека Numba позволяет увеличивать скорость выполнения программ, написанных на языке Python. Это достигается за счет предварительной компиляции методов, имеющих специальную аннотацию @jit. Numba поддерживает не все структуры данных и конструкции языка Python и его библиотек, а ориентирован на ускорение выполнения математических вычислений. По этой причине в Синтезаторе данная библиотека используется для ускорения методов, осуществляющих различные вычисления и преобразования сигналов, например, метода, приведенного в приложении А.3.

#### **4.1.4 Библиотеки RtMidi и PyAudio**

Библиотека RtMidi предоставляет возможность получения и расшифровки MIDI-сообщений, поступающих от MIDI-клавиатуры пользователя.

Библиотека PyAudio предоставляет инструментарий для работы со звуком. В Синтезаторе она используется для вывода синтезированного звука на аудиоустройство пользователя.

#### **4.1.5 Библиотека PyQt5 и среда разработки Qt Designer**

Библиотека PyQt5 позволяет использовать фреймворк Qt GUI для разработки приложений, обладающих графическим пользовательским интерфейсом. Главными преимуществами библиотеки PyQt5 являются большое количество функциональных элементов интерфейса, а также удобство их использования при разработке.

Среда разработки Qt Designer позволяет сконфигурировать и настроить элементы графического интерфейса, а затем сгенерировать код на языке Python, реализующий его.

#### **4.1.6 Система контроля версий Git и репозиторий GitHub**

Git реализует распределенную систему контроля версиями файлов. Использование Git упрощает разработку за счет предоставления возможностей отслеживания истории изменения файлов программного продукта и отката неудачных обновлений.

Онлайн-сервис GitHub – это удаленный репозиторий, позволяющий сохранять версии файлов на сервер. Использование GitHub сокращает риск утери данных, а также позволяет упростить доступ к файлам программного проекта.

### **4.2 Архитектура**

#### **4.2.1 Классы**

Концептуальная диаграмма классов, отвечающих за синтез звука приведена на рисунке 10.



- `Envelope` – класс, отвечающий за получение значения ADSR-огибающей в заданный момент времени;
- `LFOModulation` – класс, отвечающий за вычисление модуляции заданного осциллятора в заданный момент времени значением LFO в тот же момент;
- `WaveAdder` – класс, отвечающий за сложение значений нескольких осцилляторов в соответствии со значениями коэффициентов для каждого из них;
- `Detune` – класс, отвечающий за получение значений коэффициентов смещения частоты для каждого осциллятора в зависимости от заданного коэффициента максимального отклонения;
- `Panner` – класс, отвечающий за получение значений сигнала для каждого из двух каналов в зависимости от заданного коэффициента;
- `ModulatedPanner` – класс, отвечающий за получение значений сигнала для каждого из двух каналов в зависимости от заданного коэффициента и значения LFO в заданный момент времени;
- `Synth` – класс, отвечающий за получение значений сигнала, синтезированного компонентами синтеза в соответствии с их параметрами;
- `Controller` – класс, реализующий ввод высоты звука при помощи MIDI-клавиатуры и вывод синтезированного звука на аудиоустройство.

## **4.3 Реализация компонентов синтеза звука**

### **4.3.1 Реализация осцилляторов**

Класс `Oscillator` реализует метод генерации сигнала на основе предварительно вычисленных значений, описанный в пункте 2.4.5 данной работы.

Экземпляр класса `WaveAdder` содержит массивы значений одного периода четырех видов сигналов с амплитудой равной единице. Массивы формируются единожды при инициализации экземпляра с заданной частотой дискретизации. Кроме того, Экземпляр класса `WaveAdder` содержит массивы значений интегрированных сигналов, которые используются при реализации частотной модуляции.

Класс `Oscillator` содержит ссылку на экземпляр класса `WaveAdder`, а также метод `get_next_sample`, возвращающий значение сигнала, частота которого равна `frequency` и амплитуда равна `amplitude`, в момент времени `time` путем обращения к нужному элементу массива значений.

Таким образом, метод `get_next_sample` является реализацией формулы (22):

$$osc(a, f_{osc}, t) = a * F(2\pi f_{osc}t), \quad (22)$$

где  $osc(a, f_{osc}, t)$  – значение сигнала осциллятора, получаемое в момент времени  $t$ ;  $a$  – амплитуда сигнала осциллятора, которая равная значению `amplitude`;  $F$  – периодическая функция, согласно которой вычисляются значения сигнала осциллятора;  $f_{osc}$  – частота сигнала осциллятора, которая равна значению `frequency`;  $t$  – момент времени равный `time`, в который получается значение сигнала осциллятора.

Для получения значения сигнала осциллятора в момент времени  $t$  в методе `get_next_sample` вычисляется индекс элемента массива значений по формуле (21). Полученное из массива значение сигнала умножается на переданное значение амплитуды.

Код метода `get_next_sample` приведен в приложении А.1.

Для реализации модуляции сигнала осциллятора LFO и ADSR-огибающей используется класс `ModulatedOscillator`, являющийся наследником класса `Oscillator` и переопределяющий метод `get_next_sample`. Переопределенный метод осуществляет модуляцию сигнала, вызывая методы модуляций, описанные далее.

#### 4.3.2 Реализация модуляции осциллятора LFO

Модуляция сигнала осциллятора LFO осуществляется в методе `get_modulated_sample` класса `LFOModulation`. Данный метод возвращает значение модулированного сигнала, генерируемого осциллятором `osc` и имеющего амплитуду равную `amplitude` и частоту равную `frequency`, в момент времени `time`. Для модуляции получается значение сигнала LFO или его интеграла в момент времени `time`, после чего вычисляется значение модулированного сигнала по формулам (5), (12) и (16).

Предварительное вычисление значений интегралов сигналов, осуществляемое при инициализации экземпляра класса `WaveAdder`, позволяет сократить время вычисления значения частотно-модулированного сигнала.

Код метода `get_modulated_sample` приведен в приложении А.2.

#### 4.3.3 Реализация модуляции осциллятора ADSR-огибающей

Класс `Envelope` содержит массив значений ADSR-огибающей для всех моментов времени. Для модуляции сигнала осциллятора получают значения огибающей и осциллятора в заданный момент времени, после чего находится их произведение. Таким образом реализуется формула (17).

#### 4.3.4 Реализация детюна

При инициализации экземпляра класса `Detune` вычисляется массив коэффициентов смещения частоты в зависимости от количества модифицируемых осцилляторов. Для получения смещения частоты сигнала

осциллятора значение его частоты умножается на соответствующий элемент массива коэффициентов.

Код метода для получения массива коэффициентов, удовлетворяющих соотношению (1), приведен в приложении А.3.

#### **4.3.5 Реализация панорамы**

Метод `get_stereo_sample` класса `Panner` формирует массив из значений сигнала для левого и правого каналов в зависимости от коэффициента согласно формулам (2) и (3).

Для реализации модуляции коэффициента панорамы используется класс `ModulatedPanner`, являющийся наследником класса `Panner` и переопределяющий метод `get_stereo_sample`. Переопределенный метод вычисляет значение LFO в заданный момент времени и на его основании модулирует коэффициент согласно формуле (18).

Код методов `get_stereo_sample` приведен в приложении А.4.

#### **4.3.6 Реализация сумматора**

Метод `get_sum` класса `WaveAdder` принимает массив значений осцилляторов в заданный момент времени, после чего суммирует их в зависимости от коэффициентов каждого из осцилляторов согласно формуле (4).

Код метода `get_sum` приведен в приложении А.5.

### **4.4 Реализация контроллера**

Класс `Controller` обеспечивает работу Синтезатора с устройствами ввода и вывода. В качестве устройства ввода выступает MIDI-клавиатура, а в качестве устройства вывода – динамики или наушники, подключенные к звуковой карте.



Работа класса Controller основана на одновременном выполнении трех потоков: поток ввода MIDI-данных, поток вывода на аудиоустройство, поток синтеза звука. Схема взаимодействия потоков приведена на рисунке 11.

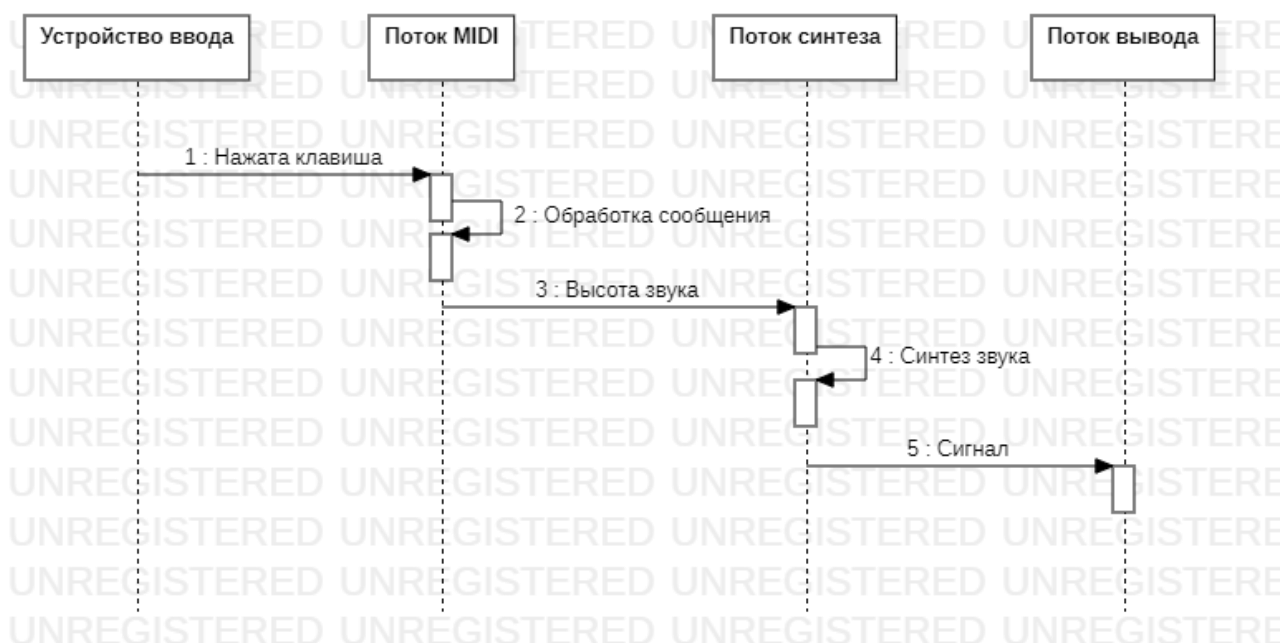


Рисунок 11 - Взаимодействие потоков в контроллере

Код метода, выполняющегося в потоке синтеза звука, приведен в приложении А.6.

#### 4.5 Реализация интерфейса

Для управления параметрами компонентов синтеза звука в Синтезаторе разработан графический пользовательский интерфейс, который приведен на рисунке 12. Далее рассмотрены элементы интерфейса, отвечающие за управление параметрами компонентов.



Рисунок 12 - Интерфейс Синтезатора

#### 4.5.1 Интерфейс управления осциллятором

Интерфейс управления осциллятором приведен на рисунке 13.

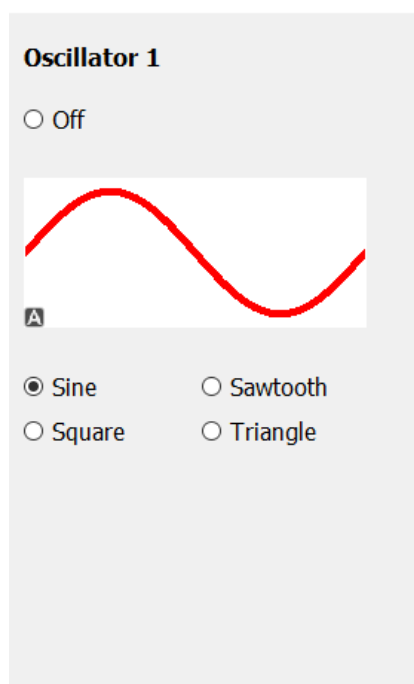


Рисунок 13 - Интерфейс управления осциллятором

Данный интерфейс содержит следующие элементы:

- переключатель, включающий и отключающий осциллятор;
- осциллограф, выводящий форму сигнала осциллятора;
- четыре переключателя для выбора формы сигнала осциллятора.

#### 4.5.2 Интерфейс управления модуляцией осциллятора LFO

Интерфейс управления осциллятором приведен на рисунке 14.

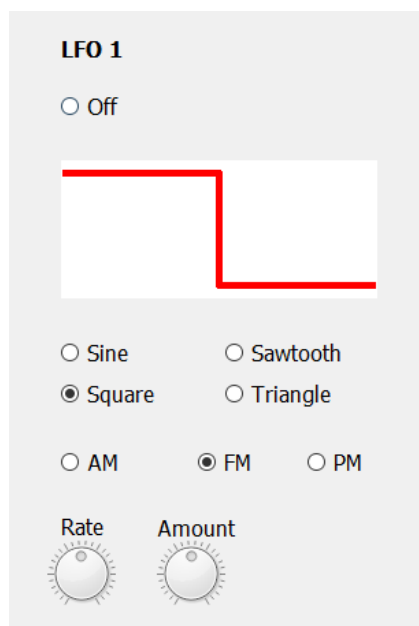


Рисунок 14 - Интерфейс управления модуляцией осциллятора LFO

Данный интерфейс содержит следующие элементы:

- переключатель, включающий и отключающий модуляцию осциллятора LFO;
- осциллограф, выводящий форму сигнала LFO;
- четыре переключателя для выбора формы сигнала LFO;
- три переключателя вида модуляции;
- потенциометр Rate, управляющий частотой сигнала LFO;
- потенциометр Amount, управляющий индексом модуляции.

### 4.5.3 Интерфейс управления модуляцией осциллятора ADSR-огибающей

Интерфейс управления модуляцией осциллятора ADSR-огибающей приведен на рисунке 15.



Рисунок 15 - Интерфейс управления модуляцией осциллятора ADSR-огибающей

Данный интерфейс содержит следующие элементы:

- переключатель, включающий и отключающий модуляцию осциллятора ADSR-огибающей;
- четыре слайдера, управляющих длительностью интервалов ADSR-огибающей;
- слайдер, управляющий уровнем поддержки ADSR-огибающей.

### 4.5.4 Интерфейс управления детюном

Интерфейс управления детюном приведен на рисунке 16.



Рисунок 16 - Интерфейс управления детюном

Данный интерфейс содержит следующие элементы:

- переключатель, включающий и отключающий детюн;
- потенциометр, управляющий коэффициентом детюна.

#### 4.5.5 Интерфейс управления сумматором

Интерфейс управления сумматором приведен на рисунке 17.

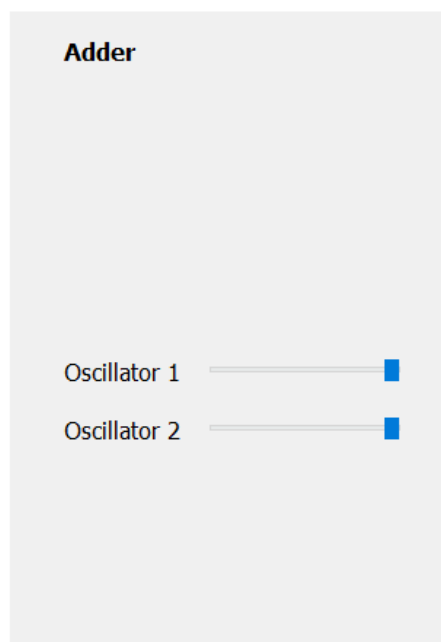


Рисунок 17 - Интерфейс управления сумматором

Данный интерфейс содержит два слайдера, управляющих коэффициентами, определяющими долю каждого из осцилляторов в суммарном сигнале.

#### 4.5.6 Интерфейс управления панорамой

Интерфейс управления сумматором приведен на рисунке 18.

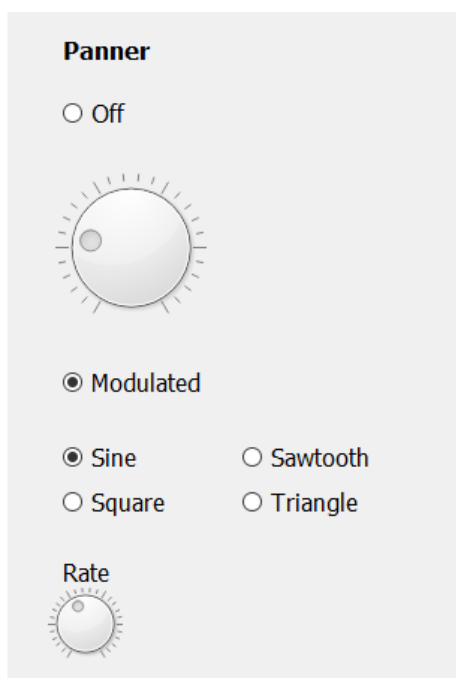


Рисунок 18 - Интерфейс управления панорамой

Данный интерфейс содержит следующие элементы:

- переключатель, включающий и отключающий панораму;
- потенциометр, управляющий коэффициентом панорамы;
- переключатель, включающий и отключающий модуляцию коэффициента панорамы LFO;
- четыре переключателя для выбора формы сигнала LFO;
- потенциометр Rate, управляющий частотой сигнала LFO.

#### 4.5.7 Интерфейс управления уровнем сигнала

Интерфейс управления сумматором приведен на рисунке 19.

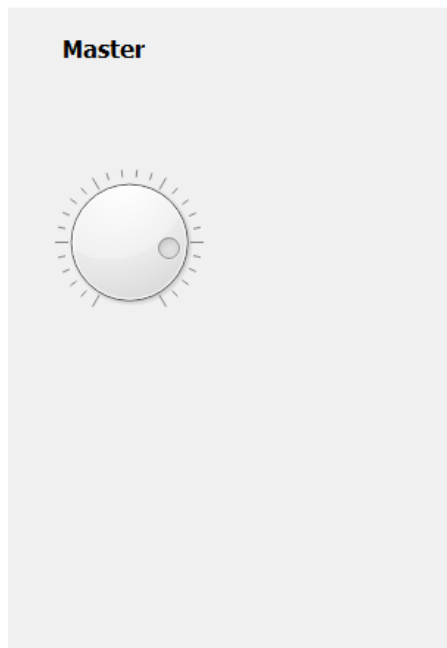


Рисунок 19 - Интерфейс управления уровнем сигнала

Данный интерфейс содержит потенциометр, управляющий коэффициентом усиления сигнала.

## 5 Перспективы развития Синтезатора

Разработанный в процессе выполнения бакалаврской работы программный синтезатор обладает перспективами для дальнейшего развития. Можно выделить несколько направлений потенциального развития:

- разработка программных синтезаторов в виде плагина реального времени и Web-приложения, основанных на методах и схемах синтеза звука, используемых в Синтезаторе;
- предоставление пользователю возможности задавать собственные образцы, на основе которых будут генерироваться сигналы осцилляторов. Например, для этой цели может быть использован конструктор формул;
- предоставление пользователю возможности составлять собственные схемы синтеза из набора реализованных компонентов. Таким образом может быть реализован модульный синтезатор;
- реализация дополнительных компонентов синтеза звука, например, эффектов и частотных фильтров.



## **Заключение**

В процессе выполнения бакалаврской работы была поставлена задача разработки программного синтезатора звука. Для выполнения поставленной задачи проведен анализ предметной области, включающий обзор существующих программных синтезаторов, а также методов синтеза звука. По результатам анализа были составлены схема и алгоритмы генерации и обработки сигналов. На основе анализа предметной области, а также полученных схем и алгоритмов синтеза были сформулированы требования к программному продукту, а также спроектирована его архитектура. Для реализации приложения был выбран набор необходимых программных средств.

Результатом бакалаврской работы является программный синтезатор звука, удовлетворяющий поставленным требованиям и реализующий составленные схемы и алгоритмы синтеза звука.

Разработанный программный продукт обладает перспективами дальнейшего развития, а составленные алгоритмы могут применяться при реализации различных задач по генерации и обработке цифровых сигналов.

## Список используемых источников

1. IFPI issues Global Music Report 2021 : [Сайт]. — URL : <https://www.ifpi.org/ifpi-issues-annual-global-music-report-2021/> (дата обращения 13.02.2022);
2. Селиверстов, Н.А. История развития цифровых программных синтезаторов / Н.А. Селиверстов, А.А. Вахтин // Труды молодых ученых факультета компьютерных наук. Выпуск 2. – Воронеж : ВЭЛБОРН, 2022. – С.144-149;
3. Reid, G. An introduction to additive synthesis / G. Reid : [Электронный ресурс]. – URL : <https://www.soundonsound.com/techniques/introduction-additive-synthesis> (дата обращения 10.06.2022);
4. Reid, G. Formant Synthesis / G. Reid : [Электронный ресурс]. – URL : <https://www.soundonsound.com/techniques/formant-synthesis?page=1> (дата обращения 10.06.2022);
5. Chowning, M. The Synthesis of Complex Audio Spectra by Means of Frequency Modulation / M. Chowning : [Электронный ресурс]. – URL : [https://ccrma.stanford.edu/sites/default/files/user/jc/fm\\_synthesis\\_paper.pdf](https://ccrma.stanford.edu/sites/default/files/user/jc/fm_synthesis_paper.pdf) (дата обращения 10.06.2022);
6. Papen, R. The 4 Element Synth: The Secrets of Subtractive Synthesis / R. Papen. – Rowman & Littlefield Publishers, 2020. – 224 p.
7. Roads, C. The computer music tutorial / C. Roads. – Cambridge : The MIT Press, 1996. – 1234 p.;
8. Stolet, J. Low-frequency Oscillators / J. Stolet : [Электронный ресурс]. – URL : <https://pages.uoregon.edu/emi/31.php> (дата обращения 10.06.2022);

9. Печатнов, Б. Синтез частотных и временных характеристик в ЭМС / Б. Печатнов, С. Сабуров // Радио. – 1980. – № 12. – С. 24-27;
10. Newkirk, D Mixers, modulators and demodulators. The ARRL Handbook for Radio Communications / D Newkirk, R. Karlquist : [Электронный ресурс]. – URL : [https://d1.amobbs.com/bbs\\_upload782111/files\\_26/ourdev\\_534476.pdf](https://d1.amobbs.com/bbs_upload782111/files_26/ourdev_534476.pdf) (дата обращения 10.06.2022);
11. Баскаков, С.И. Радиотехнические цепи и сигналы / С.И. Баскаков. – Москва : Ленанд, 2016. – 528 с.;
12. Der, L. Frequency Modulation (FM) Tutorial / L. Der : [Электронный ресурс]. – URL : <https://web.archive.org/web/20190303023627/http://pdfs.semanticscholar.org/c122/bb065f9a5540970b7a1298a908e88af6dfb9.pdf> (дата обращения 10.06.2022).

## Приложение А

```
1. def get_next_sample(self, amplitude, frequency, time, pressed):
    if(frequency<1) or not pressed:
        return 0
    wave = self.wave_generator.waves[self._type]
    t = (int)((frequency*time) % render_rate)
    return amplitude*wave[t]

2. def get_modulated_sample(self, osc, osc_amplitude, osc_frequency,
                           lfo, lfo_frequency, mod_index,
                           time, render_rate):
    if(self.type == 0):
        lfo_val = lfo.get_next_value(lfo_frequency, time)
        osc_val = osc.get_next_sample(osc_amplitude,
                                       osc_frequency, time)
        return (1+mod_index*lfo_val)*osc_val/mod_index
    elif(self.type == 1):
        lfo_val = lfo.get_next_integral(lfo_frequency, time)
        return osc.get_next_sample(osc_amplitude,
                                    osc_frequency,
time+render_rate*mod_index*lfo_val/osc_frequency)
    else:
        lfo_val = lfo.get_next_value(lfo_frequency, time)
        return osc.get_next_sample(osc_amplitude,
                                    osc_frequency,
time+render_rate*mod_index*lfo_val/(2*np.pi*osc_frequency))

3. @staticmethod
    @njit(cache=True)
    def set_ratio_numba(count_of_oscs : int):
        minus = []
        plus = []
        max = int(count_of_oscs/2)+1
        for i in range(1, max):
            minus.append(-1/i)
            plus.append(1/(max-i))
        if not (count_of_oscs % 2 == 0):
            minus.append(0)
        return minus + plus

4. '0 <= index <= 1'
    def get_stereo_sample(self, sample, time):
        return [(1-self.index)*sample, self.index*sample]

    def get_modulated_stereo_sample(self, sample, time):
        x = self.lfo.get_next_value(self.lfo_rate, time)
        y = x * (self.index-0.5) + 0.5
        return [(1-y)*sample, y*sample]
```

```

5. def get_sum(self, samples):
    if (len(samples) > len(self.indexes)):
        self.indexes += [0]*(len(samples)-len(self.indexes))
    res = 0
    for i in range(0,len(samples)):
        res += samples[i] * self.indexes[i]
    return res / sum(self.indexes)

6. def render(self):

    start = 0
    end = self.buffer_size
    prev_state = False
    time_up = 0

    while self.running:

        sample = []

        pressed = self.midi_interface.pressed
        currentFreq = self.midi_interface.currentFreq

        if pressed != prev_state:
            if pressed:
                start = 0
                end = self.buffer_size
            else:
                time_up = start

        for t in range(start, end):
            sample.append(self.synth.get_next_sample
                           (amplitude=0.6,
                            frequency=currentFreq,
                            time=t,pressed=pressed,
                            time_up=time_up))

        self.stream.write(np.array(sample,
                                   dtype=np.float32).tostring())

        start = end
        end += self.buffer_size
        prev_state = pressed

```