

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

**ВЫПУСКНАЯ** **КВАЛИФИКАЦИОННАЯ РАБОТА**

Тема: «ВЕБ-САЙТ ДЛЯ ИНТЕРАКТИВНОЙ ГЕНЕРАЦИИ МУЗЫКИ И ВИЗУАЛИЗАЦИИ НА ОСНОВЕ ФРАКТАЛОВ»

Направление подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем

Направленность Математическое обеспечение и администрирование информационных систем

Обозначение ВКР 02.03.03.570000.000 Группа ВМО41

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.А. Евсюкова

подпись, дата

Руководитель ВКР \_\_\_\_\_\_\_\_\_\_\_\_\_\_ доцент Т.А. Медведева

подпись, дата

Ростов-на-Дону

2025



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
 «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

**ЗАДАНИЕ**

на выполнение выпускной квалификационной работы

Тема: «ВЕБ-САЙТ ДЛЯ ИНТЕРАКТИВНОЙ ГЕНЕРАЦИИ МУЗЫКИ И ВИЗУАЛИЗАЦИИ НА ОСНОВЕ ФРАКТАЛОВ»

Обучающийся Евсюкова Софья Александровна

Обозначение ВКР 02.03.03.570000.000 Группа ВМО41

Тема утверждена приказом по ДГТУ от « 28 » апреля 2025 г. № 2156-ЛС-О

Срок представления ВКР к защите « » июня 2025 г.

Исходные данные для выполнения выпускной квалификационной работы:

Задание на выпускную квалификационную работу, Мандельброт, Б. Фрактальная геометрия природы / Б. Мандельброт. – Москва: Институт компьютерных исследований, 2002. – 468 с., Хольцман, Л. Фрактальный хаос в цифровом искусстве / Л. Хольцман. – Санкт-Петербург: Наука, 2006. – 198 с., Головнин, С.А. Web Audio API: основы синтеза звука в браузере / С.А. Головнин / Программные продукты и системы. – 2019. – 33 с.

Содержание выпускной квалификационной работы

Введение:

Описание сферы использования программного средства, обоснование актуальности темы.

Наименование и краткое содержание разделов:

1. Теоретический обзор. В данном разделе описывается предметная область, рассматриваются теория фракталов, описаны ключевые алгоритмы построения фракталов (IFS, L-системы). Рассмотрены особенности визуализации и генерации музыки на их основе. Определена цель, сформулированы задачи и требования к программному средству.

2. Проектирование веб-сайта. В данном разделе представлены алгоритмы генерации фрактальных структур в двух режимах: IFS и аксиомы. Построены логические схемы, описывающие процесс синтеза фрактальной визуализации и аудио.

3. Программное конструирование. В данном разделе обосновывается выбор стека технологий (Flask, PostgreSQL, Tone.js, HTML5 Canvas). Представлены реализованные модули интерфейса, генерации графики и звука, а также хранения и обработки данных пользователей. Описан механизм работы регистрации и сохранения композиций.

4. Тестирование программного средства. Приведена инструкция по использованию программного средства, а также протестирована его функционала генерации фракталов и аудио, реализовано два режима генерации (IFS и аксиомы). Представлены результаты тестов.

5. Безопасность и экологичность при разработке программного средства. Разработан комплекс мероприятий по организации рационального освещения в рабочем помещении. Выполнен расчет системы искусственного освещения.

Заключение:

В процессе написания данной работы было разработано интерактивное веб-приложение, позволяющее пользователю создавать фрактальные музыкальные композиции на основе визуальных структур. Система поддерживает регистрацию, выбор моделей фракталов, работу в двух режимах генерации, сохранение и управление композициями. Проведено тестирование, подтвердившее стабильность и корректность работы.

|  |  |  |
| --- | --- | --- |
| Руководитель ВКР | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | доцент Т.А. Медведева |
| Задание принял к исполнению | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | С.А. Евсюкова |

**Аннотация**

Отчёт включает: страниц – 105; рисунков – 38; таблиц – 7;   
источников – 18; приложений – 2;

Цель работы – разработка веб-сайта для генерации музыкальных композиций с визуализацией фракталов.

Разработано программное средства, позволяющее генерировать музыкальные последовательности на основе фрактальных структур. Описана реализация двух режимов генерации: с использованием IFS и L-систем. Представлены алгоритмы построения визуальных моделей и преобразования их параметров в звуковые элементы для реализации требуемого функционала. Программное средство поддерживает регистрацию пользователей, сохранение и управление композициями, а также визуализацию музыки.

**Abstract**

The report includes: pages – 105; drawings – 38; tables – 7; sources – 18; appendices – 2;

The purpose of the work is to develop a web application for generating fractal-based musical compositions with visualization.

This work is dedicated to the development of software that enables users to generate musical sequences based on fractal structures. The system implements two generation modes using IFS and stochastic L-systems. The report describes the algorithms for visual model construction and transformation of their parameters into sound. The application supports user registration, composition saving and management, and real-time music visualization. Functional testing of the key system features has been performed.

**Содержание**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

4

02.03.03.570000.000 ПЗ

Разраб.

Евсюкова С.А.

Провер.

Медведева Т.А..

Н.контр

Попова О.А.

Утвердил

Долгов В.В.

Веб-сайт для интерактивной генерации музыки и визуализации на основе фракталов

Лист.

Листов

101

ДГТУ

Кафедра «ПОВТиАС»

[Введение 6](#_Toc201436825)

[1 Теоретический обзор 7](#_Toc201436826)

[1.1 Понятие фракталов и их свойства 7](#_Toc201436827)

[1.2 Классификация фракталов 8](#_Toc201436828)

[1.3 Математические модели и алгоритмы фрактальной генерации 11](#_Toc201436829)

[1.4 Фрактальная генерация музыкальных структур 21](#_Toc201436830)

[1.5 Постановка задачи 26](#_Toc201436831)

[1.7 Выводы по главе 27](#_Toc201436832)

[2 Проектирование веб-сайта 28](#_Toc201436833)

[2.1 Структура программного средства 28](#_Toc201436834)

[2.2 Модули генерации фрактальных структур 30](#_Toc201436835)

[2.3 Архитектура базы данных 32](#_Toc201436836)

[2.4 Выводы по главе 34](#_Toc201436837)

[3 Программное конструирование 35](#_Toc201436838)

[3.1 Выбор языка программирования и среды разработки 35](#_Toc201436839)

[3.2 Архитектура клиент-сервер 37](#_Toc201436840)

[3.3 Структура базы данных 39](#_Toc201436841)

[3.4 Компоненты для визуализации и генерации музыки 42](#_Toc201436842)

[3.5 Выводы по главе 44](#_Toc201436843)

[4 Демонстрация работы программного средства 45](#_Toc201436844)

[4.1 Описание контрольного тестирования 45](#_Toc201436845)

[4.2 Инструкция по использованию программы 47](#_Toc201436846)

[4.3 Тестирование генерации фракталов 54](#_Toc201436847)

[4.3 Выводы по главе 64](#_Toc201436848)

[5 Безопасность и экологичность проекта 65](#_Toc201436849)

[5.1 Анализ опасность вредных факторов 65](#_Toc201436850)

[5.2 Расчет освещения на рабочем месте разработчика 68](#_Toc201436851)

[5.3 Экологичность рабочей зоны 72](#_Toc201436852)

[5.4 Пожарная безопасность рабочей зоны 74](#_Toc201436853)

[5.5 Выводы по главе 75](#_Toc201436854)

[Заключение 77](#_Toc201436855)

[Перечень используемых информационных источников 78](#_Toc201436856)

[Приложение А Техническое задание 80](#_Toc201436857)

[Приложение Б Исходный код программного средства 86](#_Toc201436858)

Введение

Современные веб-технологии позволяют создавать интерактивные мультимедийные приложения. Они объединяют визуализацию, звук и пользовательское взаимодействие. Особую популярность приобретают системы, которые используют алгоритмическое моделирование и генерацию звука для творчества. Фрактальные модели становятся популярным инструментом для создания уникальных музыкальных композиций.

Целью работы является создание интернет-ресурса с системой генерации фрактальных мелодий. Веб-сайт позволяет создавать музыкальные композиции на основе разных типов фракталов с их визуализацией. Пользователь может гибко настраивать параметры. Актуальность исследования связана с развитием алгоритмического искусства. Также важен поиск новых форм генеративного творчества. Фракталы широко применяются в компьютерной графике, мультимедийных проектах и аудиовизуальных инсталляциях.

В ходе работы создана платформа с использованием Python и Flask для серверной части. Клиентская часть реализована на JavaScript с библиотекой Tone.js. Система обеспечивает создание и воспроизведение музыкальных фрактальных шаблонов. Данные пользователей сохраняются в базе PostgreSQL. Веб-интерфейс предоставляет инструменты для настройки фракталов, управления композициями и избранными треками. Также реализована безопасная аутентификация и поддержка работы в реальном времени.

Практическая значимость заключается в использовании алгоритмов и моделей для создания музыкального и графического ПО. Эти методы могут применяться в фрактальной терапии, мультимедийных проектах, цифровом искусстве, генеративной музыке, аудиовизуальных инсталляциях и видеоиграх.

1 Теоретический обзор

В данной главе рассмотрены основные понятия и свойства фракталов, их классификация и примеры. Описаны методы фрактальной визуализации и их связь с генерацией музыки. Поставлены задачи, необходимые для реализации веб-сайта, и обоснованы выбранные подходы.

1.1 Понятие фракталов и их свойства

Фракталы – это математические объекты, которые обладают свойством самоподобия, т. е. структура объекта повторяется на разных уровнях масштабирования. Самоподобие фрактала означает, что его части, независимо от масштаба, в котором они рассматриваются, напоминают его целую форму [10].

Свойства фракталов:

* самоподобие – это одна из самых ярких характеристик фракталов. Примеры самоподобных объектов включают такие природные явления, как разветвления деревьев, кровеносные сосуды, молнии или даже береговые линии. В математике это свойство проявляется, например, в кривых Коха или в множествах Жюлиа [4];
* дробная размерность – фрактальные объекты часто не могут быть охарактеризованы целым числом размерности. Например, линия может иметь размерность 1, поверхность – 2, а объем – 3. Однако фракталы могут иметь дробную размерность, что отражает их сложную структуру [1]. Размерность фрактала можно измерить с помощью метода Хаусдорфа, который позволяет оценить, насколько сложно заполнить пространство объектом при его увеличении [10];
* рекурсивность фракталов означает, что их можно построить с помощью повторяющихся процедур или алгоритмов. Это свойство объясняется тем, что на каждом уровне масштаба процесс построения фрактала повторяется многократно, начиная с простых шагов и переходя к более сложным формам. Такой подход используется во многих алгоритмах фрактальной генерации, таких как метод итеративных функций (IFS) и системы Линденмайера.

Важным примером фракталов в природе являются структуры, подобные деревьям, где ветви и побеги напоминают основной стебель. Это явление также можно наблюдать в таких объектах, как облака или горные ландшафты, где элементы каждой из форм могут быть описаны с помощью фрактальных уравнений [10].

1.2 Классификация фракталов

Фракталы можно классифицировать по разным признакам, среди которых важнейшими являются типы, используемые для их создания, а также свойства, которые они проявляют в своем строении. Рассмотрим более подробно основные типы фракталов, которые играют ключевую роль в теории фракталов.

**1.2.1 Геометрические фракталы**

Геометрические фракталы представляют собой класс строгих самоподобных структур, формируемых с использованием детерминированных итерационных процедур на основе аффинных преобразований. Их построение опирается на систему итеративных функций (Iterated Function System, IFS), каждая из которых представляет собой преобразование, действующее на подмножества евклидова пространства.

На каждом шаге фигура разбивается на части, каждая из которых масштабируется, поворачивается и переносится в соответствии с фиксированными правилами. В результате многократного применения этих преобразований формируется фрактальная структура, обладающая инвариантностью относительно масштаба и высокой степенью   
регулярности [2].

Математически IFS определяется как конечный набор аффинных сжимающих отображений , каждое из которых имеет   
вид (1.1).

(1.1)

где – матрица линейного преобразования (масштабирование, поворот, отражение);

– вектор смещения.

Фрактал определяется как единственная инвариантная компактная точка по отношению к системе (1.2).

(1.2)

Такое множество существует и единственно по теореме Банаха о сжимающих отображениях. Реализация построения осуществляется через рекурсивную процедуру: начальная фигура на каждом шаге заменяется набором её образов под действием всех , порождая сходящееся к   
при .

Геометрические фракталы используются не только как математические объекты, но и в искусстве для создания симметричных, многократно повторяющихся форм и в **географических исследованиях** для моделирования природных объектов, таких как линии побережья или разветвления рек.

**1.2.2 Стохастические фракталы**

Стохастические фракталы – это фракталы, которые характеризуются введением элементов вероятности в процесс построения. Их основное отличие –наличие случайного выбора преобразования на каждом шаге генерации. Они описывают явления, которые не могут быть полностью предсказаны заранее, но имеют статистическую закономерность. Такие фракталы могут иметь самоподобие, однако оно проявляется не в строгой регулярности, а скорее в вероятностном распределении [10].

Математически, такие фракталы реализуются через **вероятностные итерационные системы функций (PIFS).** Формула отображения остаётся такой же, как в IFS, но теперь каждая функция выбирается не всегда, а с вероятностью представлена формулой (1.3).

. (1.3)

Стохастическая модель представляет собой марковский процесс над метрическим пространством , в котором последовательность точек формируется по правилу (1.4)

, (1.4)

где выбирается случайно с вероятностью .

Правило гарантирует, что независимо от начальной точки , процесс почти наверняка стремится к инвариантной мере, поддерживающей фрактальную структуру на множестве-аттракторе.

На практике стохастические фракталы широко применяются в моделировании природных объектов (растительности, ландшафтов), поскольку позволяют достоверно воспроизводить структурную неоднородность и вариативность.

**1.2.3 Алгоритмические фракталы**

**Алгоритмические фракталы** – это фракталы, которые строятся с использованием строгих математических алгоритмов, часто на основе итерационных процессов. Они обладают четко определенной структурой и обычно строятся с помощью численных методов [15].

Основной объект изучения – последовательность комплексных чисел.Рассматривается функция , обычно в форме полинома второй степени (1.5).

(1.5)

где – комплексные числа;

– параметр.

Функция рекурсивно применяется к начальному значению формируя последовательность (1.6).

(1.6)

Свойства сходимости или расходимости этой последовательности определяют принадлежность точки к множеству, называемому алгебраическим фракталом, чья граница обычно обладает фрактальной размерностью. Это отображение иллюстрирует чувствительную зависимость от начальных условий и демонстрирует хаотическое поведение в областях комплексной плоскости.

С вычислительной точки зрения, процедура построения заключается в применении алгоритма escape-time, в котором фиксируется максимальное число итераций и вычисляется, на каком шаге выполняется . Этот метод позволяет визуализировать структуру фрактала через цветовое кодирование скорости расходимости.

1.3 ****Математические модели и алгоритмы фрактальной генерации****

Фрактальные структуры обладают уникальными свойствами самоподобия и сложностью, которые можно моделировать с помощью различных математических моделей и алгоритмов. Эти модели не только помогают понять природу фракталов, но и являются основой для их применения в искусстве, научных исследованиях и инженерных задачах [15]. В этом разделе рассмотрены основные математические модели и алгоритмы, которые лежат в основе фрактальной генерации.

**1.3.1 Снежинка Коха**

**Снежинка Коха** – один из наиболее известных примеров фрактальной геометрии. Это замкнутый фрактал, являющийся частным случаем кривой Коха, предложенной шведским математиком Хельге фон Кохом в 1904 году. Построение этой фигуры основано на итеративном преобразовании отрезков, в результате чего формируется кривая с бесконечной длиной, ограничивающая конечную площадь [11].

Снежинка Коха представляет собой замкнутую фигуру, формируемую путём построения кривой Коха на каждой стороне равностороннего треугольника. Процесс построения начинается с начального треугольника с длиной стороны . На каждой итерации n его стороны делится на три равные части, и центральная часть заменяется на два отрезка, образующих новый треугольник. Далее данный шаг повторяется для каждого нового отрезка, полученного на предыдущем этапе.

После n-й итерации количество сторон выражается по формуле (1.7).

(1.7)

где количество сторон исходного треугольника.

Длина каждого отрезка на -й итерации уменьшается с каждым шагом по формуле (1.8).

(1.8)

где – длина стороны начального треугольника.

Периметр фигуры на n-м шаге определяется по формуле (1.9).

(1.9)

где – периметр фрактала.

Следовательно, при соответствует формула (1.10).

(1.10)

Это подчёркивает одну из ключевых особенностей фрактала: бесконечный периметр при конечной занимаемой площади. Построение снежинки Коха показано на рисунке 1.1.

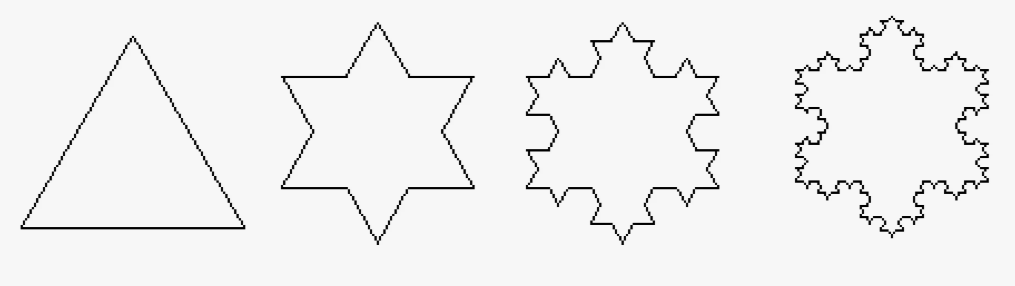


Рисунок 1.1 – Снежинка Коха

Снежинка Коха обладает свойством самоподобия – на каждом уровне построения её структура сохраняется, а каждая новая форма является уменьшенной копией всей фигуры.

Также одним из ключевых свойств снежинки Коха является её фрактальная размерность , отражающая степень её геометрической сложности. Она рассчитывается по формуле Хаусдорфа-Бесиковича (1.11).

(1.11)

где – количество отрезков после каждой итерации;

– коэффициент сокращения масштаба.

Тогда это даёт размерность:

Таким образом, снежинка Коха имеет фрактальную размерность, которая больше 1, но меньше 2, что указывает на её промежуточный характер между одномерными и двумерными объектами.

**1.3.2 Фрактальное дерево**

Фрактальное дерево – это геометрический фрактал, представляющий собой рекурсивную структуру с выраженным свойством самоподобия. Каждая ветвь такой структуры является уменьшенной копией всего дерева, формируемой путём последовательного разветвления отрезков под заданным углом и с постоянным коэффициентом масштабирования [7].

Построение фрактального дерева основано на итеративно-рекурсивном процессе, в котором каждая ветвь разделяется на несколько новых, укороченных ветвей, отклонённых от направления предыдущей на заданный угол. Этот процесс повторяется на каждом уровне, формируя сложную, самоподобную структуру. Алгоритм включает следующие этапы:

Шаг 1. Начальная ветвь (ствол) представляет собой вертикальный отрезок длины , направленным вверх из заданной точки начала координат.

Шаг 2. Из конца этого отрезка строятся две (или более) новые ветви, каждая из которых имеет длину, уменьшенную в раз, где ; отклоняется от предыдущего направления на угол .

Шаг 3. Каждая из вновь созданных ветвей становится основанием для следующего разветвления. Шаг 2 повторяется для каждой новой ветви.

Шаг 4. Процесс продолжается до достижения заданного уровня глубины , определяющего количество рекурсивных шагов построения. На каждом уровне , где, параметры ветвей вычисляются по следующим формулам:

1. Длина ветви на i-м уровне (1.12):

(1.12)

где – длина ветви на i-м уровне;

– длина предыдущей ветви;

– коэффициент уменьшения длины.

1. Угол наклона ветви: угол между соседними ветвями меняется по формуле (1.13).

, (1.13)

где – угол наклона ветви на i-м уровне;

– фиксированный угол поворота между ветвями,

знак указывает на возможность поворота как в положительную, так и в отрицательную сторону.

Такое разветвление может реализовываться с двумя, тремя и более ветвями. В простейшем случае – двоичное дерево, каждая вершина порождает две ветви, как показано на рисунке 1.2.

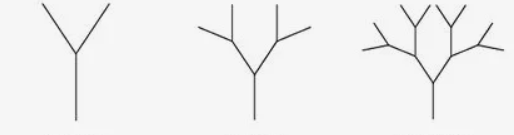


Рисунок 1.2 – Фрактальное дерево

Фрактальные деревья обладают несколькими ключевыми характеристиками, такими как самоподобие и фрактальная размерность.

Структура дерева сохраняет свою форму на каждом уровне рекурсии, но с уменьшением размеров. Это делает фрактальные деревья характерным примером самоподобных объектов, свойственных фрактальной геометрии [10].

Для фрактальных деревьев характерна дробная размерность, что означает, что дерево не укладывается в традиционную двумерную или трёхмерную геометрию. Фрактальная размерность может быть вычислена по формуле Хаусдорфа-Бесиковича (1.14).

(1.14)

где – фрактальная размерность;

– количество ветвей на каждом уровне дерева;

– коэффициент уменьшения длины ветвей.

**1.3.3. Дракон Хартера-Хейтуэя**

Дракон Хартера-Хейтуэя – это фрактальная кривая, обладающая самоподобной структурой, которая формируется посредством рекурсивного применения простых геометрических преобразований. Этот объект впервые описан в 1960-х годах как пример самопересекающейся кривой, характеризующейся конечной площадью при бесконечной длине. Дракон Хартера-Хейтуэя представляет интерес для изучения свойств фрактальных фигур и их применения в математике и физике [16].

Фрактал можно построить путём последовательных аффинных преобразований начального отрезка. Пусть исходный отрезок обозначен как . На каждой итерации из отрезка строятся два новых, повёрнутых и масштабированных отрезка, образующих «излом». Процесс описывается рекуррентной формулой (1.15).

(1.15)

где ;

и – аффинные преобразования, определяемые с помощью   
формул (1.16) и (1.17).

(1.16)

(1.17)

где – комплексное представление точки на плоскости.

Эти преобразования соответствуют повороту и сжатию отрезка с последующим объединением образов.

Фрактал дракона определяется как замыкание объединения всех итераций (1.18).

(1.18)

С каждым шагом кривая становится всё более «извилистой», стремясь к предельной форме, наглядно показано на рисунке 1.3.



Рисунок 1.3 – Дракон Хартера-Хейтуэя

Он обладает рядом фрактальных свойств:

* фрактальная размерность (Hausdorff-измерение) дракона равно 2 несмотря на то, что кривая строится из отрезков
* площадь, ограниченная кривой, стремится к фиксированной конечной величине;
* длина кривой бесконечна при стремлении числа итераций ;
* дракон инвариантен относительно преобразований и , это представлено в виде (1.19):

(1.19)

* кривая непрерывна, но нигде не дифференцируема.

**1.3.4. Множество Мандельброта**

Множество Мандельброта – одно из самых известных и изученных фрактальных множеств в комплексном анализе и динамических системах. Оно представляет собой множество точек в комплексной плоскости, для которых последовательность, задаваемая рекуррентным соотношением (1.20).

(1.20)

не расходится, то есть остается ограниченной при .

Для визуализации множества Мандельброта используется численный алгоритм, основанный на итерационном вычислении указанной последовательности для каждой точки c выбранной области комплексной плоскости. Процесс состоит из следующих этапов.

Шаг 1. Выбирается область на комплексной плоскости, обычно   
квадрат (1.21):

(1.21)

в пределах которой производится проверка принадлежности точек к множеству.

Шаг 2. Для каждой точки устанавливается начальное значение . Также задаётся максимальное число итераций от (обычно от нескольких сотен до нескольких тысяч) и порог радиуса выхода . Значение выбрано на основании теоремы, согласно которой если для некоторого n модуль превышает 2, то последовательность расходится к бесконечности

Шаг 3. Для каждого вычисляется последовательность (14) начиная с , и проверяется условие выхода: если , то последовательность расходится, и точка , где – множество Мандельброта. Если после итераций то точка считается принадлежащей множеству.

Шаг 4. Для визуализации множества обычно используется цветовая схема, где точки, принадлежащие множеству, отображаются чёрным цветом, а точки, для которых последовательность расходится, раскрашиваются в зависимости от номера итерации выхода, что позволяет продемонстрировать сложную структуру границы множества, это показано рисунке 1.4.

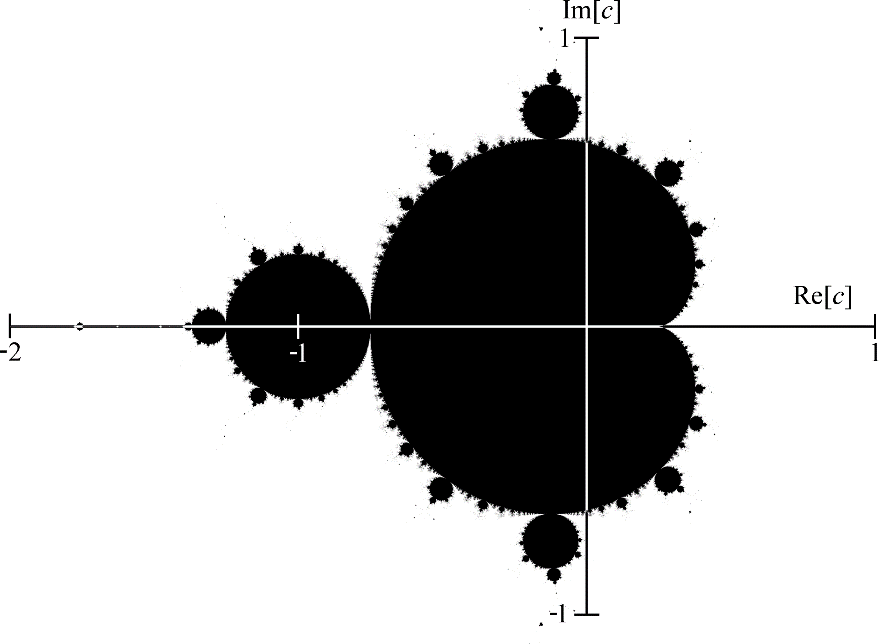


Рисунок 1.4 – Множество Мандельброта

Множество Мандельброта имеет уникальные свойства, которые делают его ключевым объектом в теории динамических систем и фрактальной геометрии. Рассмотрим главные из них:

* компактность и связность: множество является компактным и связным подмножеством комплексной плоскости ;
* фрактальная структура: граница множества Мандельброта обладает фрактальной структурой с бесконечным уровнем детализации. Это проявляется в его самоподобии и сложных узорах, возникающих при увеличении   
  масштаба [10];
* экспоненциальный рост: если для некоторого n выполняется , то последовательность (1.20) гарантированно уходит в бесконечность. Это свойство используется для численного построения множества.

**1.3.5 Папоротник Барнсли**

Папоротник Барнсли – это фрактальный объект, относящийся к классу самоподобных множеств, создаваемых с помощью итеративных аффинных преобразований с заданными вероятностями. Он является классическим примером системы итеративных функций (IFS), позволяющей моделировать сложные природные структуры, напоминающие листья папоротника [16].

Построение папоротника Барнсли основано на случайном итеративном применении четырёх аффинных преобразований , заданных матрицами и векторами смещения. Каждый шаг алгоритма выбирает одно из этих преобразований с определённой вероятностью и применяет его к текущей точке.

Шаг 1. Задаём начальную точку .

Определяем набор аффинных преобразований (1.22).

(1.22)

Шаг 2. Для каждого из четырёх аффинных преобразований задаются фиксированные вероятности, определяющие, как часто каждое из них применяется при итерациях:

Шаг 3. Для каждого шага :

* случайным образом выбирается одно из преобразований с вероятностью
* применяется выбранное преобразование к текущей точке (1.23):

(1.23)

* точка сохраняется для последующей визуализации.

Шаг 4. После большого числа итераций (обычно от нескольких тысяч до миллионов), множество полученных точек отображается на плоскости, образуя фрактальный рисунок 1.5 папоротника.

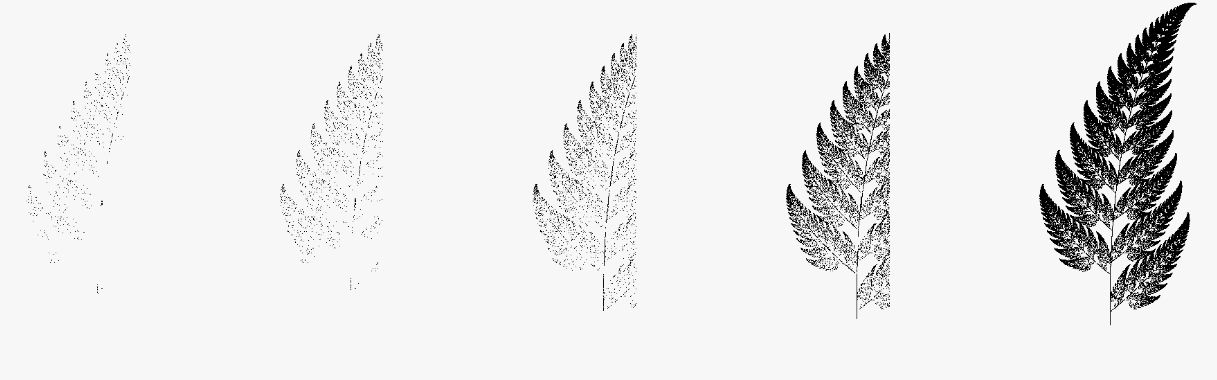


Рисунок 1.5 – Папоротник Барнсли

Благодаря своей математической структуре и способу построения, папоротник Барнсли обладает рядом характерных свойств, которые делают его важным примером стохастического фрактала в теории итеративных систем:

* структура папоротника Барнсли повторяется на разных масштабах с практически неограниченной степенью детализации [1];
* фрактал генерируется с помощью случайного выбора одного из набора аффинных преобразований с заданными вероятностями, что придаёт структуре естественную «живую» нерегулярность и разнообразие, характерное для природных объектов [1];
* построение папоротника основано на системе итеративных функций, каждая из которых является аффинным преобразованием, что позволяет эффективно моделировать сложные природные формы через простые математические операции [3];
* процесс генерации папоротника быстро сходится к фрактальному множеству, независимо от начальной точки, благодаря свойствам сжимающих отображений [4].

****1.4 Фрактальная генерация музыкальных структур****

В проекте используются фрактальные алгоритмы для генерации музыкальных паттернов, в основе которых лежат методы IFS и L-системы. Эти методы генерируют самоподобные структуры, которые затем преобразуются в музыкальные элементы, такие как мелодия, бас и ударные, с помощью синтезаторов. Параметры фракталов, такие как глубина, правила преобразования и вероятности, влияют на выбор нот и их последовательности.

В проекте используются гаммы, которые помогают структурировать музыкальные элементы, определяя выбор нот и гармоническую последовательность, синхронизируя музыку с фрактальными структурами.

1. Мажорная гамма используется для создания ярких и гармоничных мелодий. В проекте гамма мажорная применяется для генерации основных музыкальных фраз и аккордов. Мажорная гамма включает в себя следующие ноты:
2. Минорная гамма используется для более меланхоличных и эмоционально насыщенных музыкальных фрагментов. Она применяется для басовых линий, создавая более тёмное и напряжённое звучание. Минорная гамма состоит из следующих нот:

В проекте непосредственно минорная гамма не используется в явной форме, используется набор нот, который по своей природе можно ассоциировать с минорной гаммой. Басовые ноты генерируются в пределах , что соответствует низким частотам.

1. Пентатоническая гамма состоит из пяти нот и используется для создания более простых, но выразительных музыкальных паттернов, применяемых для создания ударных линий или основы баса. Пентатоническая гамма включает следующие ноты:

4. Для ударных инструментов в проекте используется отдельный набор нот, который не зависит от традиционных гамм, но имеет своё значение для создания ритма и динамики. Ноты для кика, снэйра и хи-хэта:

Ноты для ударных используются для создания ритмических структур, где каждой ноте (C2 для бочки, C3 для барабана и A2 для хи-хэта) соответствуют определённые ударные звуки. Эти ноты выбираются с помощью индексации, чтобы случайным образом или по заданному ритму генерировать последовательности для ударных инструментов:

**1.4.1 IFS-система**

Метод IFS (Iterated Function System) основывается на итеративном применении набора афинных преобразований для генерации фракталов. Каждый фрактал в IFS строится путём повторного применения одного из выбранных афинных преобразований, где каждое преобразование определяет изменение позиции, масштаба и ориентации точки или объекта. В случае фракталов каждый шаг итерации создаёт более сложную геометрическую структуру, которая в итоге может быть визуализирована как самоподобная фигура.

Математически процесс можно выразить формулой (1.24).

(1.24)

где – текущая точка или фигура на n-й итерации;

– одно из афинных преобразований (масштабирование, повороты, сдвиги и т. д.);

– индекс преобразования.

Афинные преобразования можно записываются формулой (1.25).

(1.25)

где – матрица, которая определяет повороты и масштабирование;

– вектор, который задаёт сдвиг.

Применение нескольких преобразований в сочетании с итерациями приводит к формированию фрактальных структур.

IFS создаёт структуру, которая затем переводится в музыкальную композицию. В проекте используется два типа преобразований: **детерминированные** и **стохастические**. Детерминированные преобразования создают предсказуемую музыкальную структуру, а стохастические добавляют элемент случайности, влияя на выбор музыкальных элементов.

* глубина фрактала (или количество итераций) связана с высотой ноты: чем глубже итерация, тем выше нота. Например, первая итерация может быть связана с нотой C4, в то время как глубже итерации могут быть связаны с более высокими нотами, такими как E4 или G4;
* положение по осям X и Y используется для определения временной позиции ноты в музыкальной последовательности, то есть, где нота будет воспроизведена по времени (на оси X);
* геометрические параметры фрактала, такие как угол поворота и масштаб, интерпретированы как длительность или громкость нот, при этом изменение масштаба фрактала влияет на динамику музыки.

Таким образом, с помощью IFS фрактальные структуры могут быть переведены в музыкальные паттерны, где каждое изменение геометрии фрактала является музыкальным событием, соответствующим определённой ноте.

**1.4.2 L-система**

L-система (или Lindenmayer System) – это формальная грамматика, используемая для моделирования процессов роста и развития, основанная на рекурсивном применении правил переписывания аксиом. L-система генерирует структуры через замены символов (аксиом), согласно набору, правил. Она определяется как упорядоченная тройка:

где V – алфавит, включая переменные и управляющие знаки   
(например: {F, X, +, −, [, ]});

ω – аксиома, определяющая начальное состояние;

P – набор правил переписывания, определяющих, как каждый символ заменяется на последовательность других символов при каждой итерации.

Каждое правило переписывания выглядит как:

где – символ, который заменяется на в процессе генерации.

Правила в L-системах применяются одновременно ко всем символам строки, что позволяет моделировать процессы разветвлённого и самоподобного роста. Каждая итерация приводит к экспоненциальному увеличению длины строки и усложнению структуры. Полученная последовательность символов интерпретируется как графическая инструкция – в том числе с использованием черепашьей графики,

где – движение вперёд,

+, − – повороты вправо и влево на заданный угол,

[ , ] – сохранение и восстановление состояния (позиции и направления), что позволяет строить ветвящиеся структуры.

L-системы могут быть как D0L-системами, так и стохастическими.

D0L-системами считают, если P: F → X, то независимо от того, что стоит до и после F, символ всегда заменяется на строку X. Правила переписывания имеют следующие особенности:

* детерминированность – каждому символу в алфавите соответствует только одно правило переписывания;
* контекстно-независимость – правило применяется к символу независимо от его окружения в строке.

D0L-системы имеют следующий вид:

Сначала применяем аксиому , затем применяем правила для получения более сложной структуры, что приводит к рекурсивному и самоподобному развитию структуры.

В стохастических системах правила имеют вероятностную природу. Один и тот же символ может иметь несколько возможных замен, выбор которых определяется случайным образом, согласно заданным вероятностям.

Стохастической L-системой называется четверка вида

где – конечный алфавит системы;

– аксиома, определяющая начальное состояние;

– набор правил переписывания, определяющих, как каждый символ заменяется на последовательность других символов при каждой итерации;

– функция вероятности, ставящая каждому правилу из R вероятность его применения.

На функцию P накладывается ограничение, что сумма вероятностей всех правил с одинаковой левой частью должна быть равна 1 (или не превосходит 1).

Связь L-системы с нотной грамотой осуществляется через аксиомы и правила переписывания, которые можно интерпретировать как музыкальные паттерны. Каждый символ в аксиоме или правиле ассоциирован с музыкальным событием, таким как нота или аккорд, а процесс переписывания аксиомы создает последовательность музыкальных событий, аналогичную развитию фрактальной структуры.

Начальная аксиома в L-системе первая нота в музыкальном паттерне. Например, аксиома связана с первой нотой в гамме (мажорной, минорной).

Каждое правило переписывания связано с изменением музыкальной фразы. Например, правило означает, что за нотой C4 последует нота D4, создавая динамичную музыкальную линию.

Каждый новый шаг переписывания интерпретирован как новая нота в последовательности, что создаёт музыкальный рост и развитие, аналогично тому, как L-система генерирует сложные структуры.

1.5 Постановка задачи

Целью данной работы является разработка программной реализации системы генерации музыкальных композиций на основе фрактальных структур, с использованием различных методов фрактальной генерации для синтеза музыкальных паттернов.

Для достижения цели решены следующие задачи:

1. Изучены теоретические основы фракталов, их классификацию и свойства, особенно применение фракталов в музыкальной генерации.
2. Проанализированы математические алгоритмы генерации ключевых фрактальных структур: снежинки Коха, фрактального дерева, дракона Хартера-Хейтуэя, множества Мандельброта и папоротника Барнсли.
3. Исследованы подходы к генерации фрактальной музыки, таких как IFS и L-системы, включая использование различных гамм.
4. Рассмотрены методы визуализации фракталов в интерактивной веб-среде, с возможностью динамического управления отображением.
5. Разработаны и реализованы программные модули генерации звука и визуализации фракталов, интегрированные в единую веб-платформу.
6. Спроектированы структуры базы данных и моделей взаимодействия, для хранения пользовательских настроек, музыкальных композиций и истории взаимодействий с платформой.
7. Реализована система регистрации и авторизации пользователей для персонализированного доступа к функционалу платформы, включая создание и сохранение музыкальных композиций.
8. Проведено тестирование веб-сайта и выполнить оценку его функциональности и пользовательского опыта.

Данная работа представляет собой создание доступного онлайн-инструмента, сочетающего фрактальную визуализацию и музыкальный синтез в интерактивной форме.

1.7 Выводы по главе

В данной главе проанализированы свойства и классификация фракталов, рассмотрены математические модели основных фрактальных структур, а также изучены алгоритмы генерации фрактальной музыки и визуализации, ставшие основой для разработки интерактивного веб-приложения.

2 Проектирование веб-сайта

В данном разделе представлены детализированные схемы, отражающие архитектуру программного средства для генерации музыки на основе фракталов, а также описание ключевых алгоритмов преобразования геометрических структур в звуковые паттерны.

2.1 Структура программного средства

Для проектирования программного средства, осуществляющего генерацию музыкальных композиций на основе фрактальных структур, разработан удобный и визуально понятный графический интерфейс. Пользователь может выбрать фрактальные модели для мелодии, баса и ударных, настраивая их параметры.

Доступны два режима генерации: IFS-система, где фрактал строится с использованием детерминированных или стохастических правил, и L-системы, в котором пользователь вводит собственные аксиомы и правила с возможностью указания вероятностей для каждого из них. В каждом режиме отображаются фрактальные визуализации для каждого компонента (мелодия, бас, ударные), что позволяет пользователю наблюдать изменения в реальном времени, контролируя процесс генерации композиции.

После настройки композиции пользователи могут сохранить её, добавить в избранное или экспортировать в аудиофайл.

Общая схема алгоритма программного средства представлена   
на рисунке 2.1.



Рисунок 2.1 – Схема программного средства

2.2 Модули генерации фрактальных структур

Генерация фрактальных структур в разработанной системе реализуется двумя основными методами: через IFS-системы и L-системы. Выбор метода зависит от выбранного пользователем режима: стандартный или аксиомный.

IFS-система основывается на итеративном применении афинных преобразований, таких как масштабирование, сдвиг и вращение, с использованием заранее определенных правил для геометрических преобразований фигуры. Преобразование может быть детерминированным или стохастическим, если применяется вероятность для выбора преобразования. Описанная IFS-система показана на рисунке 2.2.

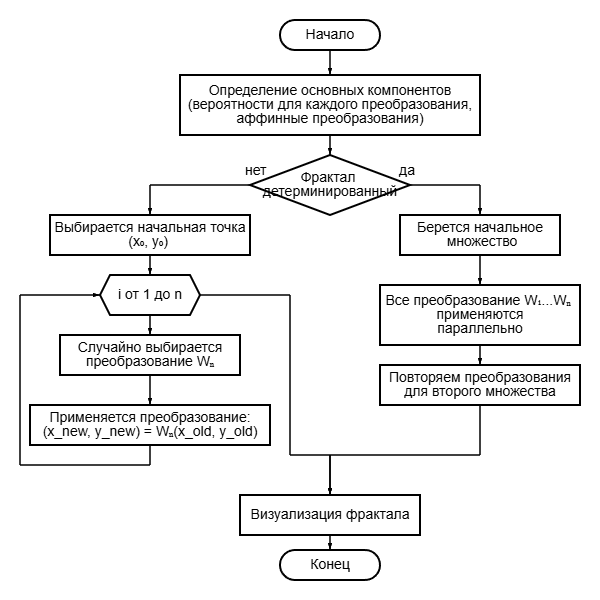


Рисунок 2.2 – Схема IFS-системы

L-система представляет собой набор формул, которые используются для генерации фракталов путем итеративного применения правил к аксиоме (начальной строке символов). В проекте используется режим аксиом, где пользователь может задавать аксиомы и правила для построения фракталов. Принцип работы заключается в том, что каждое правило применяется к символам строки, и в некоторых случаях результат применения правил может зависеть от заданных пользователем вероятностей, что позволяет создавать как детерминированные, так и стохастические фрактальные структуры. Структура L-системы приведена на рисунке 2.3.

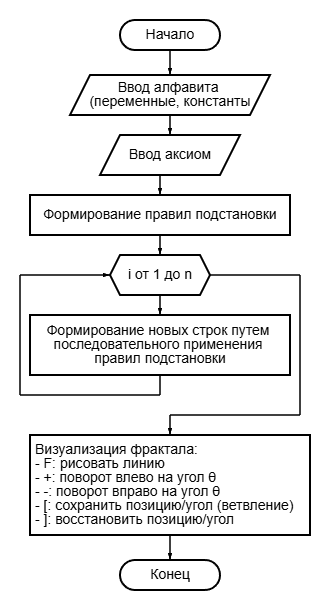


Рисунок 2.3 – Схема L-системы

2.3 Архитектура базы данных

Уровень представления:

Веб-интерфейс реализован с помощью HTML5, CSS3 и JavaScript. Для аудио-синтеза и визуализации используется библиотека Tone.js, которая позволяет динамически генерировать звук и отображать визуальные элементы. Интерфейс включает три основных модуля для музыкальных компонентов: мелодия, бас, ударные – каждый со своим выбором типа фрактала (дерево, дракон, фрактал Мандельброта и др.), глубиной и JSON-правилами генерации. Пользователь может управлять воспроизведением целиком или по отдельным компонентам, экспортировать результат в аудиофайл, сохранять и загружать пресеты. Предусмотрены разделы для просмотра истории созданных композиций и управления избранным.

Уровень бизнес-логики:

Серверная часть реализована на Python с использованием Flask – легковесного и гибкого веб-фреймворка. Сервер обрабатывает HTTP-запросы для регистрации и авторизации пользователей, а также REST API-запросы для работы с композициями и пользовательскими данными. Безопасность обеспечивается сессиями и хэшированием паролей (bcrypt). Используется PostgreSQL – мощная реляционная СУБД, которая хранит все данные пользователей, композиции в формате JSON, историю и избранное.

Для хранения информации используется реляционная база данных PostgreSQL, содержащая:

* + таблицу пользователей, отвечающую за аутентификацию и хранение данных пользователей;
  + таблицу музыкальных композиций, в которой сохраняются параметры фрактальных генераторов и информация о произведениях;
  + таблицу избранных композиций, обеспечивающую связь пользователей с их любимыми композициями.
  + таблицу истории действий пользователей, фиксирующую события создания, изменения и удаления композиций, а также управление избранным.

Особое внимание в архитектуре системы уделено обеспечению гибкости и масштабируемости за счёт многослойного подхода. Клиентская часть реализована на основе современных веб-технологий, включая HTML5, CSS3 и JavaScript, с использованием библиотеки Tone.js, позволяющей динамически генерировать аудиоконтент и обеспечивать высококачественную визуализацию музыкальных фрактальных узоров. Это обеспечивает мгновенный отклик интерфейса и удобство управления композициями.

Серверная часть построена с использованием фреймворка Flask и СУБД PostgreSQL. Такая связка обеспечивает надежную обработку REST API-запросов для регистрации, авторизации, управления пользовательскими композициями и избранным. Все операции с данными проходят через слой бизнес-логики, реализованный в Python, что позволяет выполнять необходимую валидацию и поддержку целостности данных.

Для защиты персональных данных реализованы сессии пользователя с использованием UUID, а пароли сохраняются в базе данных в виде криптографических хэшей с применением алгоритма bcrypt. Все обмены данными между клиентом и сервером проходят через защищённый протокол HTTPS, что предотвращает перехват и модификацию данных.

Система спроектирована с учетом возможного масштабирования: по мере увеличения нагрузки предусмотрена возможность перехода на более производительные СУБД и развертывание серверов в кластерном режиме с балансировкой нагрузки. Архитектура построена на основе модульного подхода, что облегчает сопровождение и расширение функционала.

На рисунке 2.4 представлена ER-диаграмма.

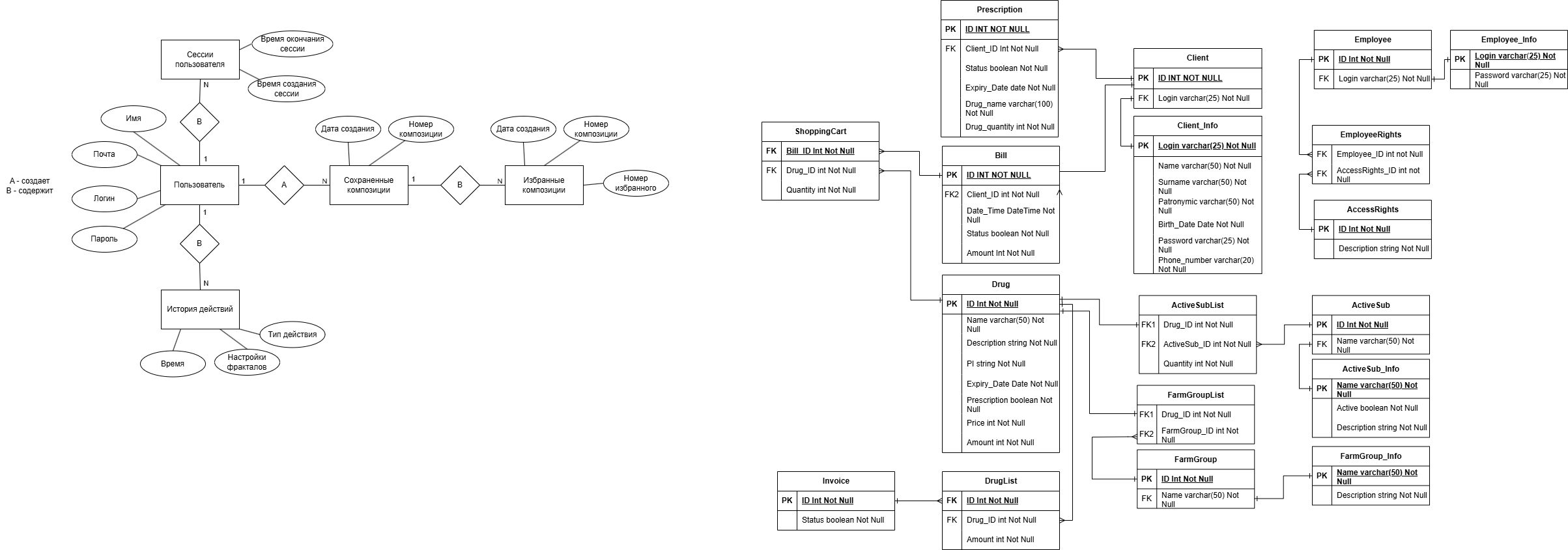


Рисунок 2.4 – ER-диаграмма

На рисунке 2.5 представлена UML-диаграмма компонентов системы, отражающая основные структурные элементы и их взаимосвязи.

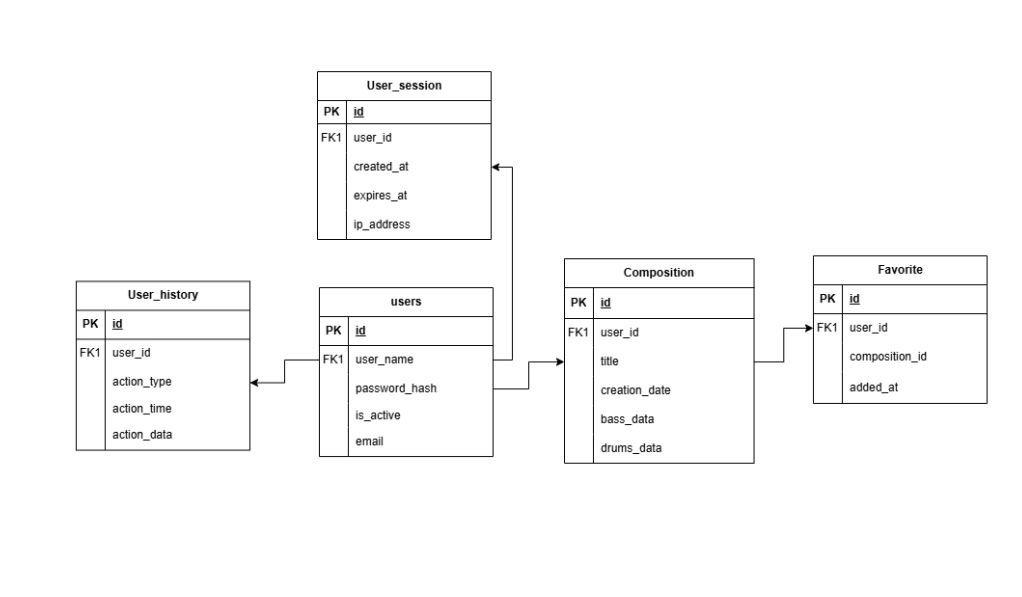


Рисунок 2.5 – UML-диаграмма

2.4 Выводы по главе

В данной главе рассмотрена общая схема программного средства, а также схемы моделей фракталов, применяемых для генерации музыки, использующиеся для дальнейшего решения поставленной задачи.

3 Программное конструирование

В данном разделе представлен выбор инструментов для разработки программного средства, включая обоснование выбора технологий, описание архитектуры клиент-серверного взаимодействия, проектирование структуры базы данных и основные классы.

3.1 Выбор языка программирования и среды разработки

При проектировании программной системы ключевым этапом является выбор технологического стека – совокупности инструментальных и программных средств, максимально отвечающих функциональным требованиям разработанного программного обеспечения. Современные задачи веб-разработки требуют поддержки высокоуровневой визуализации, обеспечения интерактивности пользовательского интерфейса и кроссплатформенности для охвата максимально широкой аудитории.

Для серверной части выбрана технология Python – язык, обладающий выразительным и лаконичным синтаксисом, идеально подходящий для создания вспомогательных API, прототипирования и реализации логики обработки пользовательских запросов. В рамках проекта применён микрофреймворк Flask, известный своей гибкостью, простотой освоения и широкими возможностями интеграции с внешними библиотеками. Flask обеспечивает реализацию REST API и веб-интерфейса с минимальными накладными расходами, а также эффективное управление аутентификацией и сессиями пользователей.

Данные хранятся в реляционной базе PostgreSQL, выбранной за надёжность, поддержку транзакций и тип данных JSONB, позволяющий сохранять сложные структуры, такие как параметры музыкальных композиций. В структуре базы реализованы таблицы пользователей, сессий, композиций, избранного и истории действий, что гарантирует целостность, безопасность и масштабируемость.

Клиентская часть построена на стандартах веб-технологий: HTML5 и CSS3 с использованием Tailwind CSS для эффективной и адаптивной стилизации. Язык JavaScript применяется для реализации динамической клиентской логики и интерактивности интерфейса, включая обработку пользовательских событий, обновление данных без перезагрузки страницы и взаимодействие с сервером посредством AJAX-запросов.

Для генерации и воспроизведения звука в браузере интегрирован JavaScript-фреймворк Tone.js, предоставляющий высокоуровневое API работы с Web Audio API. Он обеспечивает точное управление ритмом и нотами, поддерживает большинство современных браузеров и позволяет создавать сложные музыкальные композиции на стороне клиента.

Для графического отображения фрактальных структур используется элемент <canvas>, входящий в стандарт HTML5. Canvas даёт низкоуровневый доступ к пиксельной графике, что позволяет средствами JavaScript реализовывать высокопроизводительную интерактивную визуализацию и анимацию сложных геометрических объектов.

Средой разработки служил Visual Studio Code, обладающий расширенной поддержкой подсказок, диагностикой ошибок в реальном времени и интеграцией с системами контроля версий. Такой выбор ускорил выявление и исправление ошибок, способствовал эффективной организации разработки и сопровождению проекта.

Тестирование клиентского интерфейса проводилось итеративно в современных браузерах, преимущественно Google Chrome, благодаря мощным инструментам разработчика, позволяющим отслеживать выполнение кода, производительность и адаптивность дизайна.

Таким образом, выбранный стек технологий – Python 3.10 с Flask для серверной части, PostgreSQL для хранения данных, а также HTML5, CSS3, JavaScript с Tone.js и Canvas для клиентской – обеспечивает создание масштабируемой, надежной и удобной в использовании программной системы с высоким уровнем интерактивности и качественной визуализацией.

3.2 Архитектура клиент-сервер

Архитектура системы построена по классической модели с разделением на клиентскую и серверную части. Такое разделение обеспечивает чёткое разграничение ответственности, масштабируемость и гибкость развития   
веб-сайта.

Серверная часть. Сервер реализован на базе Python с использованием микрофреймворка Flask. Основные функции серверной части:

* обработка HTTP-запросов на регистрацию, вход и выход пользователей;
* управление сессиями с хранением информации о сессиях в базе данных;
* приём и сохранение музыкальных композиций и связанных данных в PostgreSQL;
* управление избранным и ведение истории действий пользователей;
* предоставление REST API для загрузки композиций, списка избранного и другой информации.

Входные данные сервер принимает через HTML-формы или JSON-запросы, возвращая в ответ HTML-страницы или JSON-объекты. Все операции с базой данных выполняются через драйвер psycopg2 с явным управлением транзакциями, что обеспечивает целостность и устойчивость данных.

В таблице 3.1 описаны функции серверной части.

Таблица 3.1 – Функции серверной части

| Функция / Маршрут | Описание |
| --- | --- |
| Регистрация (/register) | Приём данных формы регистрации, создание записи пользователя в базе |
| Вход (/login) | Аутентификация пользователя, создание сессии |
| Выход (/logout) | Завершение сессии пользователя |
| Сохранение композиции | Приём JSON с параметрами композиции, сохранение в таблицу compositions |
| Удаление композиции | Удаление записи композиции пользователя по ID |
| Управление избранным | Добавление и удаление композиций из списка избранных |
| Получение списков | Выдача сохранённых композиций, избранного, истории действий в формате JSON |

Клиентская часть. Клиент представлен современным веб-интерфейсом, построенным с использованием HTML5, CSS3 (Tailwind CSS) и JavaScript. Основные возможности клиентской части:

* управление параметрами генерации фрактальной музыки – выбор типа фрактала, глубины рекурсии, правил генерации;
* визуализация музыкальных фракталов с помощью HTML-элемента <canvas>;
* воспроизведение музыкальных компонентов через библиотеку Tone.js;
* управление воспроизведением и сохранением композиций;
* работа с сохранёнными композициями и избранным через асинхронные запросы к серверу.

В таблице 3.2 подробно описаны функции клиентской части.

Таблица 3.2 – Функции клиентской части

|  |  |
| --- | --- |
| Функция / Компонент | Описание |
| Интерфейс выбора параметров | Элементы управления для изменения настроек генерации (тип, глубина, правила) |
| Визуализация на Canvas | Отрисовка фрактальных структур мелодии, баса и ударных на отдельных холстах |
| Воспроизведение музыки | Запуск и остановка музыкальных паттернов с помощью Tone.js |
| Сохранение и загрузка композиций | Обмен данными с сервером через AJAX без перезагрузки страницы |
| Управление списками композиций | Отображение и управление сохранёнными и избранными композициями пользователя |

Взаимодействие клиент-сервер. Для обмена данными между клиентом и сервером используется AJAX через fetch API. Клиент формирует запросы на сохранение и получение данных, сервер обрабатывает их и возвращает ответы.

* Клиент отправляет запросы с параметрами музыкальной композиции в формате JSON;
* сервер сохраняет данные в базе, создаёт новые записи и возвращает подтверждения;
* клиент запрашивает списки сохранённых композиций и избранного, получает и отображает их без перезагрузки страницы;
* сервер обеспечивает аутентификацию и проверяет права доступа по сессиям.

В таблице 3.3 подробно описаны функции клиент-серверных запросов и ответов.

Таблица 3.3 – Клиент-серверные взаимодействия

|  |  |  |
| --- | --- | --- |
| Запрос (метод и путь) | Описание действия на сервере | Ответ |
| POST /save\_composition | Сохранение новой музыкальной композиции | JSON с ID сохранённой композиции |
| GET /my\_compositions | Получение списка сохранённых композиций пользователя | JSON-массив композиций |
| POST /add\_favorite | Добавление композиции в избранное | JSON-ответ с успешным статусом |
| DELETE /delete\_composition/<id> | Удаление композиции по ID | JSON с подтверждением удаления |
| GET /favorites | Получение списка избранных композиций | JSON-массив избранного |
| POST /login, /register | Аутентификация и регистрация пользователя | Перенаправление или ошибка |

3.3 Структура базы данных

Для хранения данных в программном средстве используется реляционная база данных PostgreSQL, обеспечивающая надежность, целостность и масштабируемость. Структура базы данных включает 5 таблиц, которые взаимосвязаны между собой для эффективного хранения информации о пользователях, их сессиях, музыкальных композициях, избранном и истории действий.

Структура базы обеспечивает целостность данных за счет использования внешних ключей и уникальных ограничений. Индексы ускоряют выборки по ключевым полям.

В таблице 3.4 содержится основная информация о зарегистрированных пользователях системы, с ограничениями уникальности для полей username и email, что предотвращает дублирование учетных записей.

Таблица 3.4 – Пользователь (users)

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Особенности и ограничения |
| user\_id | SERIAL | Уникальный идентификатор пользователя | Первичный ключ |
| username | VARCHAR(50) | Имя пользователя | Уникально, не может быть пустым |
| email | VARCHAR(100) | Электронная почта | Уникально, не может быть пустым |
| password\_hash | VARCHAR(255) | Хеш пароля пользователя | Обязательное поле |
| is\_active | BOOLEAN | Статус активности аккаунта | По умолчанию TRUE |

Таблица 3.5 хранит информацию о сессиях пользователей, что позволяет обеспечивать безопасность и управление доступом на уровне сессий. Связь с таблицей users реализована через внешний ключ с каскадным удалением.

Таблица 3.5 – Сессии пользователей (user\_sessions)

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Особенности и ограничения |
| session\_id | UUID | Уникальный идентификатор сессии | Первичный ключ, генерируется автоматически |
| user\_id | INTEGER | Идентификатор пользователя | Внешний ключ на users(user\_id), ON DELETE CASCADE |
| created\_at | TIMESTAMP WITH TIME ZONE | Дата и время создания сессии | По умолчанию CURRENT\_TIMESTAMP |
| expires\_at | TIMESTAMP WITH TIME ZONE | Дата и время окончания сессии | Обязательное поле |
| ip\_address | INET | IP-адрес клиента | Опциональное поле для мониторинга |

Сохранённые музыкальные композиции содержит таблица 3.6, связанные с пользователями. Использование типа JSONB для хранения параметров обеспечивает гибкость и возможность хранения сложных данных.

Таблица 3.6 – Музыкальные композиции (compositions)

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Особенности и ограничения |
| composition\_id | SERIAL | Уникальный идентификатор композиции | Первичный ключ |
| user\_id | INTEGER | Владелец композиции | Внешний ключ на users(user\_id), ON DELETE CASCADE |
| title | VARCHAR(100) | Название композиции | По умолчанию "Без названия" |
| Поле | Тип данных | Описание | Особенности и ограничения |
| creation\_date | TIMESTAMP WITH TIME ZONE | Дата и время создания | По умолчанию CURRENT\_TIMESTAMP |
| melody\_data | JSONB | Параметры основной мелодии | Хранятся в формате JSON |
| bass\_data | JSONB | Параметры басовой линии | Хранятся в формате JSON |
| drums\_data | JSONB | Параметры ударных | Хранятся в формате JSON |

Таблица 3.7 отражает отношения пользователей и их избранных композиций. Уникальность пары user\_id и composition\_id предотвращает дублирование.

Таблица 3.7 – Избранные композиции (favorites)

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Особенности и ограничения |
| favorite\_id | SERIAL | Уникальный идентификатор записи | Первичный ключ |
| user\_id | INTEGER | Идентификатор пользователя | Внешний ключ на users(user\_id), ON DELETE CASCADE |
| composition\_id | INTEGER | Идентификатор композиции | Внешний ключ на compositions(composition\_id), ON DELETE CASCADE |
| added\_at | TIMESTAMP WITH TIME ZONE | Дата и время добавления в избранное | По умолчанию CURRENT\_TIMESTAMP |

В таблице 3.8 предназначена для ведения журнала действий пользователей, что позволяет осуществлять аудит и анализ поведения.

Таблица 3.8 – История действий пользователя (user\_history)

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Описание | Особенности и ограничения |
| history\_id | SERIAL | Уникальный идентификатор записи | Первичный ключ |
| user\_id | INTEGER | Идентификатор пользователя | Внешний ключ на users(user\_id), ON DELETE CASCADE |
| action\_type | VARCHAR(50) | Тип действия | Пример: composition\_created, favorite\_added |
| action\_time | TIMESTAMP WITH TIME ZONE | Время действия | По умолчанию CURRENT\_TIMESTAMP |
| action\_data | JSONB | Дополнительные данные, связанные с действием | Позволяет хранить параметры действия |

3.4 Компоненты для визуализации и генерации музыки

Ядро музыкальной генерации – использование библиотеки Tone.js для формирования в реальном времени паттернов (эпизодов) мелодии, баса и ударных инструментов, основанных на структуре фрактала, глубине рекурсии и правилах. Основные методы приведены в таблице 3.9.

Таблица 3.9 – Методы модуля script.js

| Название метода | Функциональное назначение |
| --- | --- |
| generateMusicPattern() | Формирует последовательности нот и ритмов на основе выбранного типа фрактала и правил |
| playComponent() | Воспроизводит выбранный музыкальный компонент (мелодия, бас, ударные) с использованием Tone.js |
| exportMusic() | Экспортирует композицию в аудиоформаты (WAV/MP3) для локального сохранения |
| saveCurrentPreset() | Сохраняет текущие настройки фрактальных генераторов в локальное хранилище браузера |
| loadPreset() | Загружает сохранённые настройки и обновляет интерфейс и визуализацию |
| renderPresetsHistory() | Отображает список сохранённых пресетов, обеспечивает управление их выбором и загрузкой |

Функции отображения графических структур реализованы через Canvas API в отдельном модуле. Фрактальные генераторы используют правила, введённые пользователем, или стандартные шаблоны. Функции визуализации фракталов, приведены в таблице 3.10.

Таблица 3.10 – Методы визуализации

|  |  |
| --- | --- |
| Название | Функциональное назначение |
| drawTree() | Рекурсивное построение фрактального дерева |
| drawKochLine() | Рекурсивное построение кривой Коха |
| drawBarnsleyFern() | Отрисовка папоротника Барнсли |
| drawPreview() | Обновление Canvas с текущими параметрами фрактала |
| updateEditor() | Синхронизация текстового редактора с визуальными параметрами |
| getDefaultRules() | Загрузка стандартных параметров для каждого типа фрактала |

Разработан редактор отдельным классом с методами для интерактивной работы: смена типа фрактала, подсказки по параметрам, предпросмотр, интеграция с мышью/клавиатурой. Подробно методы класса представлены в таблице 3.11.

Таблица 3.11 – Методы класса редактора (FractalMusicSystem)

| Название методов | Функциональное назначение |
| --- | --- |
| constructor() | Инициализация всех внутренних компонентов, установка начальных параметров и привязка к DOM элементам |
| setupEventListeners() | Назначение обработчиков событий для элементов управления правилами и управления воспроизведением |
| drawPreview() | Основная функция отрисовки текущей визуализации на Canvas |
| playFractalMusic() | Связывает визуализацию с музыкальной генерацией для воспроизведения фрактальных паттернов |
| updateEditor() | Обновляет область редактирования параметров, синхронизируя её с визуальным представлением |
| updateDocumentation() | Динамическое обновление подсказок и описаний в блоке справки |

Пользователь имеет возможность в реальном времени изменять правила фракталов, наблюдать обновление визуальной части и одновременно слышать соответствующую музыкальную композицию, что обеспечивает высокую степень интерактивности и вовлечённости.

Полный листинг программы приведён в приложении Б.

3.5 Выводы по главе

В главе обоснован выбор технологий и представлена архитектура системы, включающая клиентскую и серверную части с централизованным хранилищем данных в PostgreSQL. Описана структура базы данных с ключевыми таблицами пользователей, композиций и избранного.

Класс FractalMusicSystem обеспечивает работу с фрактальной музыкой, визуализацию и синтез звука на клиенте, а сервер обрабатывает запросы, аутентифицирует пользователей и сохраняет данные.

Предложенная архитектура обеспечивает расширяемость, безопасность и удобство дальнейшей поддержки проекта.

4 Демонстрация работы программного средства

В данном разделе описаны и протестированы контрольные примеры с подробным описанием входных данных и подтверждающими работу программного средства изображениями. Приводится инструкция по использованию программного средства.

4.1 Описание контрольного тестирования

Целью тестирования системы было выявление и устранение ошибок, верификация соответствия функциональным требованиям, а также проверка её готовности к эксплуатации.

В процессе тестирования была использована методика «черного ящика», при которой проверялось функционирование программного продукта без анализа его внутренней структуры и кода. Такой подход позволил оценить работу системы с точки зрения пользователя и выявить возможные несоответствия между фактическим поведением и заявленными функциональными требованиями.

Таблица 4.1 демонстрирует результаты тестирования программного средства.

Таблица 4.1 – Результаты тестирования программного средства

| № п/п | Наименование проверки | Последовательность действий | Определение успешной проверки | Примечание |
| --- | --- | --- | --- | --- |
| 1 | Регистрация пользователя | 1. Запустить программное средство  2. Открыть страницу регистрации.  3. Заполнить поля: имя, email, пароль.  4. Нажать кнопку «Зарегистрироваться». | 1. При успешной регистрации происходит переход на страницу входа. | Ошибок нет |
| 2 | Вход пользователя | Предусловие: программное средство запущено  1. Перейти на страницу входа.  2. Ввести email и пароль.  3. Нажать кнопку «Войти». | 1. Переход на главную страницу при успешном входе. | Ошибок нет |
| 3 | Генерация фракталов в двух режимах | Предусловие: программное средство запущено  1. Открыть интерфейс генерации фракталов.  2. Выбрать режим генерации (IFS или L-системы).  3. Выбрать модель фрактала (дерево, дракон и т.д.).  4. настроить параметры  5. Нажать кнопку «Применить». | 1. Визуализация фрактала на холсте canvas.  2. Генерация музыки для выбранной модели. | Ошибок нет |
| 4 | Сохранение композиции | Предусловие: программное средство запущено и сгенерирована композиция  1. Выбрать параметры для мелодии, баса и ударных.  2. Нажать кнопку «Сохранить».  3. Проверить сохранение данных в базе. | 1.Программа сохраняет композицию и выводит её в список сохранённых. | Ошибок нет |
| 5 | Добавление композиции в избранное | Предусловие: программное средство запущено и сгенерирована композиция  1. Открыть список сохранённых композиций.  2. Выбрать композицию и нажать кнопку «В избранное». | 1. Композиция добавляются в избранное и сохраняются в базе данных. | Ошибок нет |
| 6 | Экспорт музыки в аудиофайл | Предусловие: программное средство запущено и сгенерирована композиция  1. После генерации композиции нажать на кнопку «Экспорт музыки». | 1. Файл экспортируется в формате \*.wav . | Ошибок нет |
| 7 | Просмотр сохранённых и избранных композиций | Предусловие: программное средство запущено, сохранена или добавлена в избранное композиция  1. Открыть вкладки «Сохранённые» или «Избранные».  2. Просмотреть список композиций. | 1. Список композиции отображаются корректно и могут быть загружены. | Ошибок нет |
| 8 | |  | | --- | | Загрузка сохранённых или избранных композиций на сайт | | |  | | --- | | Предусловие: программное средство запущено, сохранена или добавлена в избранное композиция  1. Открыть вкладки «Сохранённые» или «Избранные».  2.Нажать кнопку «Загрузить». | | 1. Программа выводит параметры композиции на сайт | Ошибок нет |

4.2 Инструкция по использованию программы

Разработанное программное средство представляет собой веб-приложение для интерактивной генерации музыки и визуализации на основе фрактальных структур. Пользователь может управлять музыкальными составляющими (мелодией, басом и ударными) с помощью визуального интерфейса, настраивая алгоритмы генерации и наблюдая соответствующие фрактальные изображения в режиме реального времени.

Для начала работы пользователю необходимо открыть веб-интерфейс программного средства, который запускается автоматически при выполнении скрипта app.py с помощью Flask-сервера. Пользователю отображается модальное окно авторизации, где есть два поля ввода для логина и пароля, и кнопка «Войти», ниже расположена кнопка для регистрации. На рисунке 4.1 изображено окно авторизации.

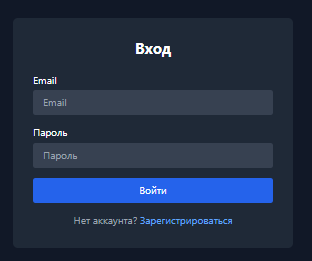


Рисунок 4.1 **– Окно авторизации**

**Если пользователь вводит логин и/или пароль, которых нет в базе, он выдаёт ошибку о том, что введено email или пароль. На рисунке 4.2 изображён вывод ошибки некорректного ввода.**

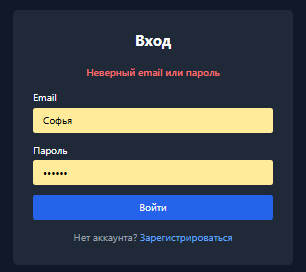


Рисунок 4.2 **–** Вывод ошибки некорректного ввода

При нажатии кнопки «Зарегистироваться», открывается новое окно регистрации, где есть поля для ввода e-mail, пароля и имя пользователя, что показано на рисунке 4.3.

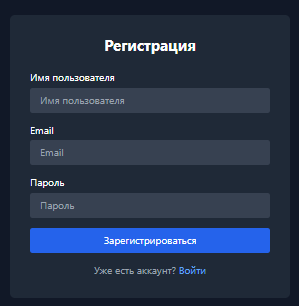


Рисунок 4.3 **– Окно для регистрации**

**Если при регистрации пользователь некорректно вводит адрес электронной почты, веб-сайт выдаст ошибку о неверном вводе e-mail. Это показано на рисунке 4.4.**

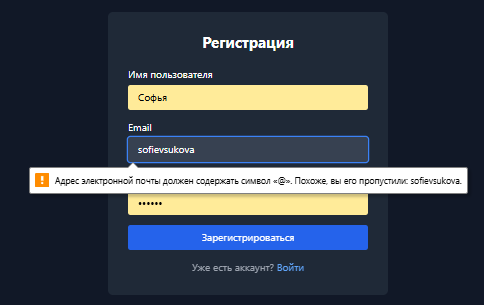


Рисунок 4.4 **– Вывод ошибки некорректного ввода e-mail**

После регистрации пользователь входит в свой профиль. Общий вид веб-сайта, с которым взаимодействует пользователь представлен на рисунке 4.5.

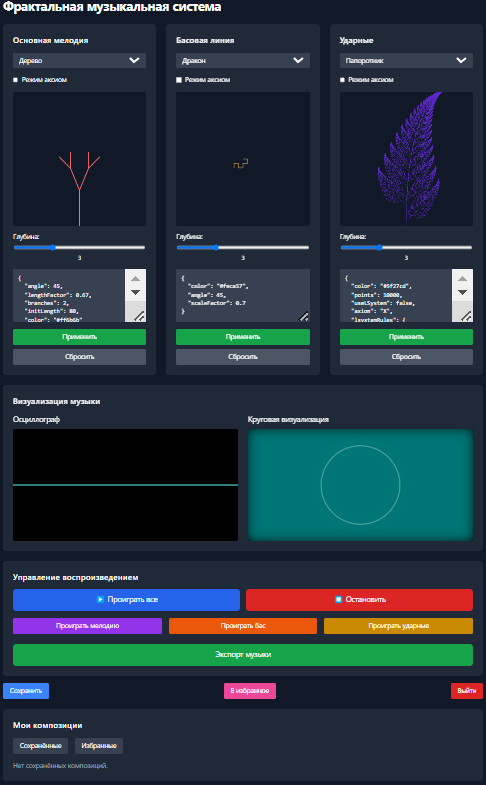


Рисунок 4.5 – Общий вид страницы веб-сайта

На главной странице отображается интерфейс с тремя основными секциями: основная мелодия, басовая линия и ударные. Для каждой из секций пользователь может выбрать тип фрактала из выпадающего списка, показанного на рисунке 4.6.

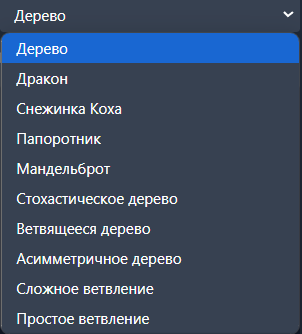


Рисунок 4.6 – Список фракталов

Пользователь выбирает один из двух режимов генерации мелодии: IFS или аксиомы. Для этого в интерфейсе предусмотрен переключатель, который активирует соответствующий режим.

В **режиме IFS** система автоматически заполняет поля параметров для выбранного типа фрактала, отображая его основные характеристики (глубина рекурсии, угол разветвления и т.д.), которые пользователь может при необходимости отредактировать.

В **режиме аксиом** пользователь может дополнитель вводить аксиомы, задавать правила и их вероятности, что предоставляет больше гибкости и контроля над результатом генерации.

Установить можно и глубину рекурсии, изменяя значение на ползунке от 1 до 8. Это определяет уровень детализации фрактала и влияет на сложность сгенерированной музыки.

Есть возможность редактировать правила генерации во встроенном текстовом поле.

Необходимо нажать кнопку «Применить», чтобы отобразить фрактал и сгенерировать соответствующую музыкальную последовательность. Можно сбросить параметры секции с помощью кнопки «Сбросить».

Ниже расположен блок управления воспроизведением:

* кнопка «Проиграть все» запускает одновременное воспроизведение мелодии, баса и ударных;
* кнопка «Остановить» завершает воспроизведение.

Доступны отдельные кнопки для воспроизведения каждой составляющей по отдельности: «Проиграть мелодию», «Проиграть бас», и «Проиграть ударные».

Кнопка «Экспорт музыки» позволяет экспортировать сгенерированную музыку в формате \*.wav. При нажатии создаётся файл, содержащий музыкальную последовательность, готовую для использования в аудиоредакторах.

Пользователь после настройки параметров может назвать и сохранить композицию на сайте. На рисунке 4.7 показан ввод названия композиции.

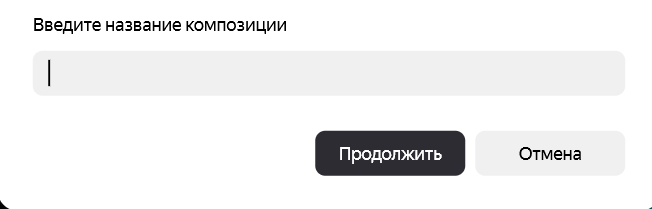


Рисунок 4.7 – Ввод название композиции

А на рисунке 4.8 отображается уведомление об успешном сохранении композиции.

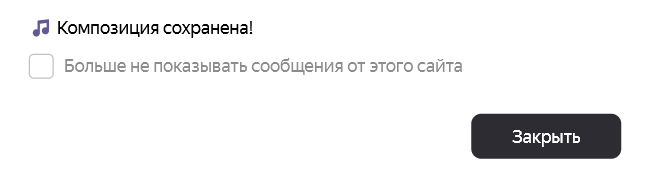


Рисунок 4.8 – Уведомление «Композиция сохранена»

На рисунке 4.9 показаны сохраненные мелодии пользователя. Есть возможность загрузки композиции для редактирования и воспроизведения, а также удаление с автоматическим обновлением списка.

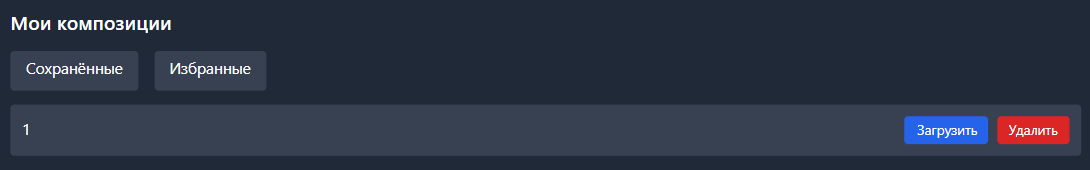


Рисунок 4.9 – Сохранение композиции

На рисунке 4.10 показано уведомление об успешной загрузки композиции.

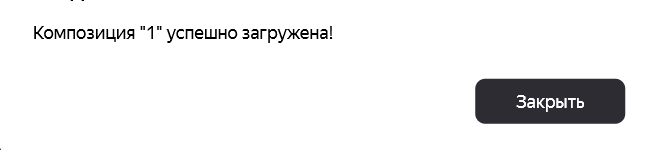


Рисунок 4.10 – Уведомление «Композиция успешно загружена»

Пользователь может отметить любую сохранённую композицию как избранную, после чего она появится в отдельной вкладке. Также реализована возможность удаления из избранного без потери самой композиции.

На рисунке 4.11 уведомление о добавлении композиции в избранное.



Рисунок 4.11 – Уведомление «Добавлено в избранное»

В процессе воспроизведения музыки одновременно отображается динамическая визуализация звукового сигнала с помощью двух независимых модулей: осциллографа и круговой спектральной визуализации.

Осциллограф предоставляет информацию о временной форме звуковой волны, отражая амплитудные изменения, тогда как круговая визуализация демонстрирует распределение частотных компонентов, что показано  
на рисунке 4.12.

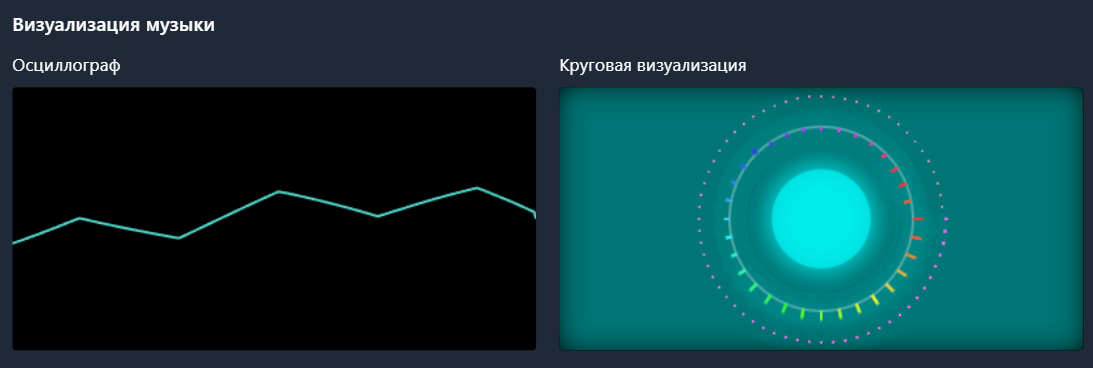


Рисунок 4.12 – Визуализация музыки

Такая интеграция аудио- и визуального ряда обеспечивает пользователю полное восприятие музыкальной композиции и фрактальной структуры, повышая интерактивность и наглядность работы системы.

4.3 Тестирование генерации фракталов

В процессе тестирования фрактальной музыкальной системы предусмотрены два основных сценария генерации фракталов. Первый сценарий основан на модели IFS, где фракталы создаются с помощью параметров. Второй сценарий – это режим аксиом, который позволяет пользователю вводить свои собственные аксиомы и задавать вероятности для генерации фракталов с использованием L-системы. Каждый из этих режимов требует проверки как в плане визуализации фракталов, так и в части их музыкального сопровождения, чтобы гарантировать корректную работу системы и соответствие заданным параметрам.

**4.3.1 Тестирование системы IFS**

Пример корректной отрисовки фрактала «Дерево» с параметрами количество ветвей = 2, угол наклона = 45°, глубина = 3, размер начальной ветви = 80, коэффицент длины ветви = 0.67, представлен   
на рисунке 4.13.

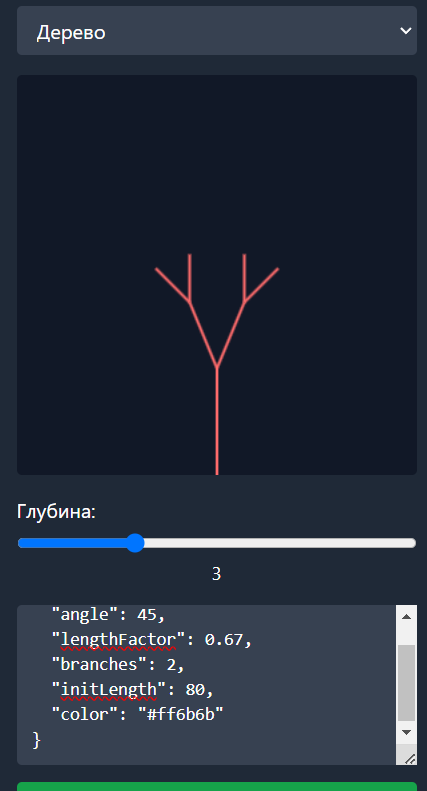


Рисунок 4.13 – Фрактальное дерево

При изменении количества ветвей (branches) от 2 до 5 – каждая итерация добавляет указанное число ответвлений. Угол наклона ветвей (angle) от 15° до 60° . Глубина рекурсии от 1 до 8 влияет на детализацию фрактала.

Мелодия − звуковая интерпритация введеных выше параметров. Каждая ветвь соответствует отдельной ноте, а увеличение параметров приводит к более сложной структуре мелодии.

Пример работы фрактала «Снежинка Коха» с параметрами глубина = 3, углом 60,° представлен на рисунке 4.14.

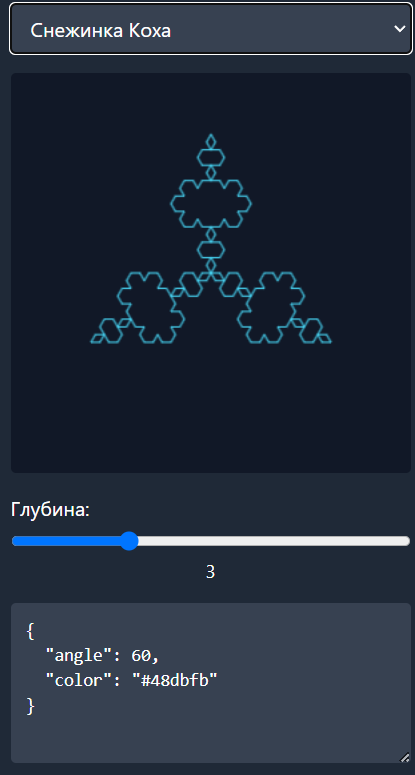


Рисунок 4.14 – Снежинка Коха

Изменяя глубину рекурсии каждый линейный отрезок исходного контура разбивается на четыре меньших сегмента, при этом средний сегмент заменяется на два, образующих острый выступ под углом 60°.

Мелодия строится на основе количества сегментов, формируя пилообразный музыкальный паттерн с чередованием высот.

Пример фрактала «множества Мандельброта» с параметрами глубина = 3, количества итераций = 100 представлен на рисунке 4.15.

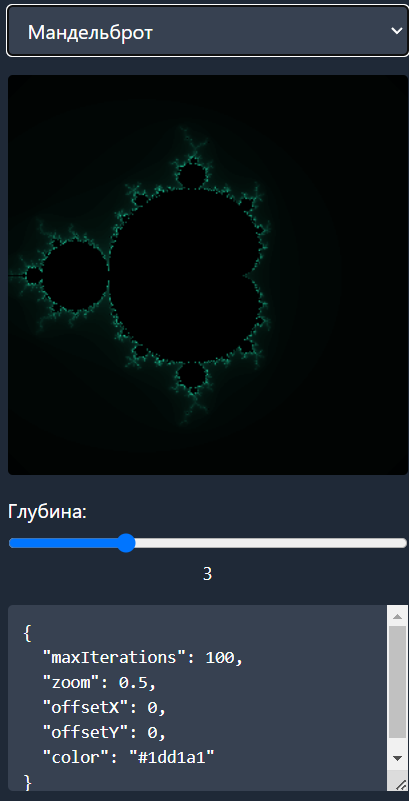


Рисунок 4.15 – Множества Мандельброта

Тестирование включает проверку при разных параметрах: масштабирования (zoom) от 0.1 до 5.0, смещения (offsetX/Y), глубины, количество итераций (maxIterations) от 20 до 500, влияя на детализацию фрактала. Цветовая палитра отражает скорость расходимости.

Параметры максимального числа итераций и глубины преобразуется в длину музыкального паттерна, где высота нот нормируется по степени сходимости точек.

Пример тестировании фрактала «дракон Хартера-Хейтуэя» с параметром глубины = 7 показано на рисунке 4.16.

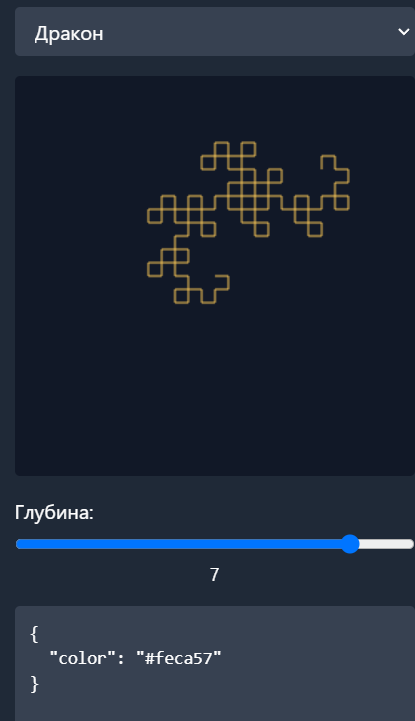


Рисунок 4.16 – Дракон Хартера-Хейтуэя

Количество изгибов увеличивается согласно глубине рекурсии, при которой формируется кривая с поворотами на 90°.

Длина мелодии связана с числом точек кривой, где ритм и высота нот формируются по синусоидальной функции, обеспечивая плавность и периодичность звучания.

Пример фрактала «папоротник Барнсли» с параметрами точек = 10000 и глубины = 3 показано на рисунке 4.17.

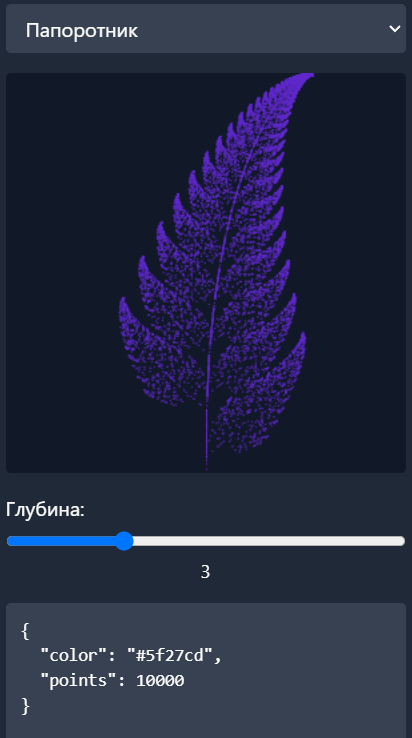


Рисунок 4.17 – Папоротник Барнсли

Тестирование фрактала включает проверку генерации 10 000-100 000 точек и глубины. Распределение точек соответствует вероятностным коэффициентам преобразований. Музыкальные ноты аналогично генерируются случайным образом, но с сохранением общей статистической структуры.

**4.3.2 Тестирование L-системы**

Для тестирования фракталов в режиме аксиом необходимо удостовериться, что каждый из поддерживаемых корректно генерируется с использованием введенных пользователем аксиом и вероятностей. Важно, что параметры для каждого фрактала будут такими же, как и при использовании модели IFS, включая глубину рекурсии, угол разветвления, длину ветвей и другие параметры.

Тестирование фрактального дерева начинается с выбора этого фрактала в интерфейсе. После включения режима аксиом, пользователю предоставляется возможность ввести аксиомы в формате JSON. Задаются аксиома «F»с правилом «F [+F][-F]», где каждая ветвь дерева будет делиться на два направления. После ввода аксиом пользователь нажимает кнопку «Применить», и система отображает дерево, что показано на рисунке 4.18.

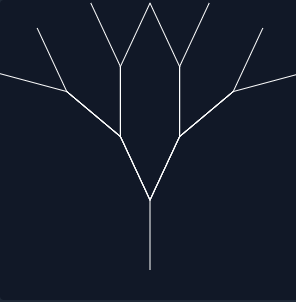


Рисунок 4.18 – Фрактальное дерево

Для фрактала снежинки Коха вводится аксиома «F» правило «F+F-F-F+F», где каждый отрезок будет повторяться несколько раз, образуя форму снежинки. Как и в случае с деревом, после ввода аксиом и применения их через кнопку "Применить.

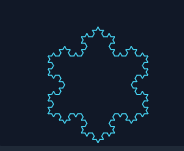


Рисунок 4.19 – Снежика Коха

Снежинка Коха отображается в соответствии с заданным правилом это показано на рисунке 4.19, а музыкальная последовательность отражает её симметричную и геометрическую структуру.

Для фрактала Дракон Хартера-Хейтуэя вводится аксиомы «FX» и соответствующие правила для «X» и «Y». Для «X» можно задать правило «X+YF+» и для «Y» правило «-FX-Y».

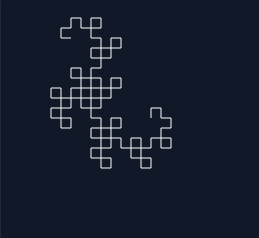


Рисунок 4.20 – Дракон Хартера-Хейтуэя

Фрактал генерируется в виде замкнутого узора с характерными резкими углами, продемонстрированно на рисунке 4.20, и музыкальное сопровождение отражает динамичность этого фрактала.

Для папоротника Барнсли процесс немного сложнее, поскольку в режиме аксиом используется не одно правило, а несколько с разными вероятностями. Применяется замены символа «X», где вводится правило «F+[[X]-X]-F[-FX]+X». Отрисовка папоротника показана на рисунке 4.21.

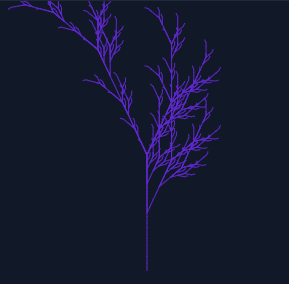


Рисунок 4.21 – Папоротник Барнсли

При каждом применении правил с учетом вероятности система генерует разные варианты фрактала, создавая визуально разнообразные папоротники, и соответствующее музыкальное сопровождение отражает каждый вариант.

В дополнение к основным фракталам, в системе предусмотрены расширенные варианты деревьев, основанные на L-системах. Эти модели отличаются разнообразием правил ветвления и позволяют добиться визуального и музыкального многообразия.

В ходе тестирования были проверены следующие варианты деревьев:

1. Стохастическое дерево. Данная модель использует вероятностное разветвление, что позволяет получать уникальные структуры при каждой генерации. Благодаря стохастике фрактал визуально приближен к естественным формам. Вводится аксиома «F» и правила (4.1)

(4.1)

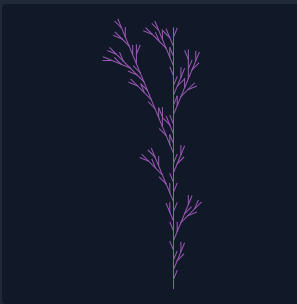


Рисунок 4.22 – Стохастическое дерево

На рисунке 4.22 при глубине 4 дерево принимает органичную форму с нерегулярным распределением веток, имитируя природный беспорядок. Музыкальный результат также варьируется: структура ритма становится менее предсказуемой, что усиливает эффект живого звучания.

1. Ветвящееся **дерево.** Структура с симметричным ответвлением по оси. Вводится аксиома «X» и правила: «X»: «F[+X][-X]FX»:, «X»: «FF». На рисунке 4.23 показана отрисовка фрактала.

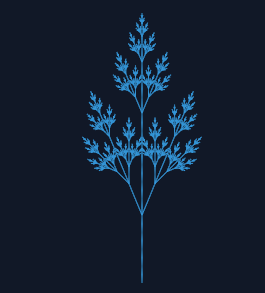


Рисунок 4.23 – Ветвящееся дерево

Фрактал демонстрирует чёткую структуру, каждая итерация даёт равномерное расхождение ветвей. Музыкальный шаблон, генерируемый на его основе, отличается ритмической стабильностью и регулярностью.

1. Асимметричное **дерево.** За счёт изменения порядка и знаков поворотов получается модель с выраженной асимметрией. Вводится аксиома «X» и правила: «X»: «F[+X][-X]+X»:, «X»: «FF», на рисунке 4.24 отображена отрисовка дерева.



Рисунок 4.24 – Ассиметричное дерево

При визуализации выявлено смещение центра тяжести структуры — дерево словно склоняется в сторону. Это создаёт эффект движения или ветра. В музыкальном плане результат выразительный: акценты сдвинуты, мелодия получается напряжённой и неровной.

1. Дерево со сложным ветвлением. Использует более сложное правило и аксиому с вложенными отклонениями. Визуально модель напоминает густые кроны или кустарники, отличается высоким уровнем детализации.

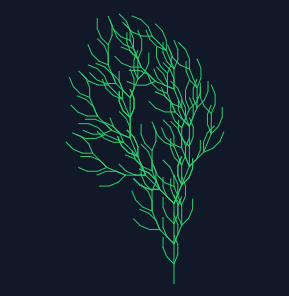


Рисунок 4.25 – Дерево со сложным ветвлением

Полученное дерево на рисунке 4.25, визуально напоминает заросли или кусты, особенно на глубине выше 4. Мелодия имеет плотную структуру и часто повторяет схожие ритмы с небольшими вариациями, создавая эффект насыщенности.

1. Дерево с простым ветвлением. Минималистичная модель с одним типом отклонений. Вводится аксиома и правило . На рисунке 4.26 отображена его отрисовка.

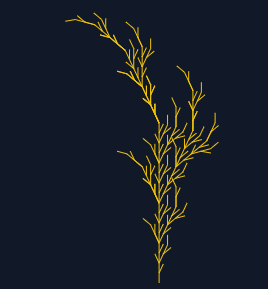


Рисунок 4.25 – Дерево с простым ветвлением

При глубине до 3 структура остаётся очень простой, но уже начиная с 4-го уровня наблюдается характерное веерообразное расхождение ветвей. Музыкально такой фрактал даёт ритмично повторяющиеся и легко предсказуемые шаблоны, хорошо подходящие для ударных или фонового сопровождения.

Тестирование фракталов с использованием L-системы подтвердило, что все функции работают как ожидается, и система успешно генерирует фракталы с заданными аксиомами и правилами. Визуализация и музыкальное сопровождение корректно синхронизированы, а интерфейс позволяет удобно взаимодействовать с системой для создания фрактальных композиций.

4.3 Выводы по главе

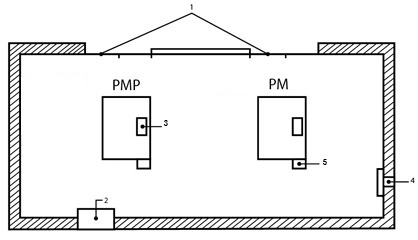
В ходе реализации практической части разработана веб-система для генерации музыки и визуализации на основе фрактальных структур. Тестирование показало, что приложение стабильно работает и предоставляет пользователю возможности по настройке и генерации музыкально-графических паттернов в реальном времени.

5 Безопасность и экологичность проекта

В данной главе рассматриваются меры по обеспечению безопасных условий при разработке и тестировании веб-сайта, а также рекомендации по оптимизации рабочего пространства и режима труда. Кроме того, затрагиваются аспекты экологичности проекта, включая минимизацию энергопотребления и рациональное использование вычислительных ресурсов. Предлагаемые мероприятия ориентированы на снижение вредных последствий длительной работы с ПЭВМ и создание комфортной и безопасной цифровой среды как для разработчиков, так и для конечных пользователей.

5.1 Анализ опасность вредных факторов

План по мерам и правилам безопасности разрабатывается для вводного инструктажа на рабочем месте в отделе программного обеспечения. В состав рабочего места входят персональный компьютер с видеодисплейным терминалом (ПК), принтер, ксерокс и искусственный светильник. План расположения рабочих мест представлен на рисунке 5.1.



1 – окна с форточкой; 2 – дверь; 3 – ПК; 4 – кондиционер; 5 – искусственный светильник; РМР – рабочее место руководителя; РМ – рабочее место сотрудника.

Рисунок 5.1 – План расположения рабочих мест в помещении

В соответствии с утверждённым регламентом, каждый сотрудник обязан пройти первичный инструктаж непосредственно на рабочем месте. Проведение инструктажа возлагается на непосредственного руководителя подразделения. Переход к самостоятельному выполнению должностных обязанностей возможен только после завершения периода стажировки и успешной проверки знаний в области охраны труда, включая как теоретические аспекты, так и практические умения.

В дополнение к составлению плана-конспекта инструктажа, на основании требований Федерального закона «Об основах охраны труда в Российской Федерации», обязательным условием обеспечения безопасности труда является проведение аттестации рабочих мест. При этом помещения, в которых выполняются трудовые функции, должны соответствовать действующим санитарным и гигиеническим нормативам.

В подразделении, специализирующемся на разработке программного обеспечения, значительная часть времени работников проходит за экраном монитора. В таких условиях уровень освещённости рабочего пространства играет ключевую роль в формировании безопасной и комфортной среды. Для контроля данного параметра используется прибор люксметр. Согласно нормам, установленным СНиП, освещённость должна находиться в пределах 150–300 лк в зависимости от характера выполняемых работ. В следующем разделе будет подробно рассмотрен подход к организации освещения в отделе разработки, а также представлены рекомендации по достижению нормативных показателей освещённости.

Разработка программного обеспечения, несмотря на внешнюю «безопасность» офисной среды, сопряжена с целым рядом факторов, которые могут негативно повлиять на здоровье и работоспособность программиста. Работа за компьютером в течение длительного времени требует постоянной концентрации внимания, пребывания в ограниченном пространстве и взаимодействия с техническими устройствами, создающими специфические физические условия.

В условиях стандартного рабочего места разработчика выделяются следующие потенциально вредные и опасные факторы.

* недостаточное или избыточное освещение – может привести к быстрой утомляемости глаз и снижению продуктивности;
* фоновый шум от систем охлаждения техники и другого оборудования – способен вызывать стресс и раздражение;
* электромагнитное излучение, создаваемое мониторами, системными блоками и сетевыми устройствами – при длительном воздействии может оказывать неблагоприятное влияние на нервную и иммунную систему;
* статическое электричество – скапливается при работе с компьютерной техникой, особенно при низкой влажности воздуха;
* нарушение микроклимата (перегрев/охлаждение помещения, сухой воздух) – способствует возникновению дискомфорта, головных болей и снижению внимания.

Для оценки уровня потенциальной опасности этих факторов составим таблицу 5.1.

Таблица 5.1 – Анализ опасных и вредных факторов

| № | Фактор | Чем выражается | Защита от фактора | Уровень опасности |
| --- | --- | --- | --- | --- |
| 1 | Недостаточное  освещение | Усталость глаз, снижение концентрации | Настройка уровня  освещённости, использование настольных ламп | 6 |
| 2 | Шум от оборудования | Раздражение, потеря внимания | Использование тихих  вентиляторов, шумоизоляция | 5 |
| 3 | Электромагнитные  поля | Влияние на ЦНС, общее утомление | Экранирование, соблюдение расстояния, качественные  экраны | 3 |
| 4 | Статическое  электричество | Микроразряды, риск выхода техники из строя | Заземление, увлажнение воздуха | 4 |
| 5 | Несоблюдение температурного режима | Головные боли, снижение работоспособности | Кондиционирование, проветривание | 5 |

На основе приведённого анализа можно сделать вывод, что наиболее значимым по уровню потенциальной опасности для разработчика является недостаточность освещение. Учитывая высокий балл этого фактора и его распространённость в условиях работы с компьютерной техникой, в следующем разделе будет произведён детальный расчёт параметров и меры по организации освещения в помещении.

5.2 Расчет освещения на рабочем месте разработчика

Для расчёта параметров общего равномерного освещения рабочей зоны, схематически представленной на рисунке 5, применяется метод определения светового потока с учётом коэффициента его использования. Этот подход позволяет учитывать не только прямой свет от источников освещения, но и долю света, отражённого от потолка, стен и других внутренних поверхностей помещения. Определение необходимого светового потока для группы светильников, оснащённых светодиодными лампами, производится в люменах на основании соответствующей расчетной формулы (5.1).

|  |  |
| --- | --- |
| , | (5.1) |

где – нормированная минимальная освещенность, лк;

– коэффициент минимальной освещенности (для светодиодных ламп принимает значение, равное 1);

– коэффициент запаса;

– коэффициент использования светового потока ламп, зависит от КПД и кривой распределения силы света светильника, коэффициента отражения от потолка и стен, а также от высоты подвеса светильников;

– количество светильников, шт.;

– количество ламп в одном светильнике, шт.

Показатель помещения , необходимый для определения величины рассчитывается по формуле (5.2).

|  |  |
| --- | --- |
| , | (5.2) |

где , – длина и ширина рабочего помещения соответственно, м; – высота подвеса светильника, м.

Исходные данные помещения, изображенного на рисунке 5.1, представлены в таблице 5.2.

Таблица 5.2 – Исходные данные помещения

|  |  |
| --- | --- |
| Параметр помещения | Значение параметра помещения |
| Длина помещения А, м | 9 |
| Ширина помещения В, м | 6 |
| Высота помещения h, м | 3 |
| Коэффициент отражения от потолка | 30 |
| Коэффициент отражения от стен | 10 |
| Коэффициент отражения от пола | 10 |
| Нормируемая освещенность , лк | 300 |
| Длина одного светильника , м | 1,534 |
| Количество ламп в одном светильнике, шт. | 2 |

Для начала необходимо вычислить высоту подвеса светильника над рабочей поверхностью, по формуле (5.3):

(5.3)

где – высота рабочей поверхности, м, принимает значение, равное 0,8;

– расстояние светового центра светильника от потолка(свес).

м,

м.

Далее определяем индекс помещения по формуле (5.2).

|  |  |
| --- | --- |
| . |  |

Значение коэффициента использования светового потока для светильников с светодиодными лампами составляет: .

Значение коэффициента запаса для помещения, с малыми выделениями пыли, составляет:

.

Оптимальное расстояние L между рядами светильников, м, определяется, из отношения (5.4).

(5.4)

где – коэффициент оптимального расстояния, который при освещении рядами светодиодных ламп принимается по умолчанию 1,4.

,

≈ 2,31 м.

Ближайшее расстояние от стен до светильников, принимает   
значение (5.5).

(5.5)

.

При выполнении условия равномерного освещения общее количество светильников, шт., находится по следующей формуле (5.6).

(5.6)

где – количество светильников по ширине (количество рядов), шт. полученное по формуле (5.7).

(5.7)

шт.

– количество светильников по длине, шт. рассчитанное по   
формуле (5.8).

(5.8)

шт.

Общее количество светильников по формуле 5.6 составляет:

шт.

Расчет светового потока одной лампы проведен по формуле (5.1) и определен тип лампы и ее мощность.

.

В соответствии с расчетами и параметрами светодиодных ламп общего назначения, в данном помещении необходимо установить лампы холодного света с мощностью = 10 Вт, силой тока = 0,35 А и напряжением = 104 ± 0,4 В.

Для удовлетворения проектируемой системой требований СНиП по освещенности должно быть выполнено следующее условие:

≥ (300 лк по усл.).

Тогда осуществим расчет фактической освещенности, лк по   
формуле (5.9).

(5.9)

лк.

Далее необходимо определить потребляемую мощность осветительной установки, Вт, по формуле (5.10).

(5.10)

где – мощность лампы, Вт;

– число светильников, шт.;

– число ламп в светильнике;

– коэффициент, учитывающий потери пускорегулирующей аппаратуры, принимает значение 1,25.

= 375 Вт.

В результате проведённых расчётов установлено, что при использовании светодиодных ламп с паспортным световым потоком 1462 лм обеспечивается фактическая освещённость на уровне 369,23 лк, что соответствует нормативу СНиП. Таким образом, проектируемая система освещения гарантирует комфортные и безопасные условия труда для сотрудников. Общая потребляемая мощность осветительной установки составляет 375 Вт, что позволяет достичь необходимого уровня освещённости без избыточного энергопотребления. Эти данные подтверждают эффективность выбранного решения для организации равномерного освещения рабочей зоны.

5.3 Экологичность рабочей зоны

Современное рабочее место программиста веб-сайта, должно соответствовать экологическим стандартам. Одним из ключевых аспектов устойчивого офисного пространства является грамотная организация раздельного сбора отходов, минимизирующая негативное воздействие на окружающую среду.

Анализ состава мусора в типичном офисе веб-разработчика показывает следующее распределение:

* бумага и картон (80%) – преобладающая категория, включающая черновики, документацию, упаковку от оборудования и использованную офисную бумагу;
* пластик (10%) – упаковка от доставки еды, бутылки, контейнеры и канцелярские принадлежности;
* пищевые отходы (5%) – остатки перекусов, чайная заварка, кофейная гуща;
* электронные компоненты и батарейки (5%) – отработанные элементы питания, сломанные кабели, неисправные гаджеты.

Одним из ключевых направлений в создании экологичной рабочей среды является внедрение системы раздельного сбора отходов. В офисах, где основную часть мусора составляет бумага (около 80%), важно организовать удобные точки сбора макулатуры. Для этого можно разместить специальные контейнеры возле рабочих мест и принтеров, а также разработать систему мотивации для сотрудников, чтобы повысить их вовлеченность в процесс сортировки. Дополнительно стоит сократить использование бумаги за счет перехода на электронный документооборот и настройки принтеров на двустороннюю печать по умолчанию.

Не менее важным аспектом является работа с пластиковыми отходами, которые составляют примерно 10% от общего объема мусора. Чтобы снизить их количество, можно полностью отказаться от одноразовой посуды в пользу многоразовых кружек и контейнеров. Также полезно договориться с поставщиками офисных принадлежностей о минимальной упаковке товаров или использовании перерабатываемых материалов. Для уже накопленного пластика необходимо организовать его сбор и передачу специализированным компаниям, занимающимся переработкой.

Особого внимания заслуживают пищевые отходы, на которые приходится около 5% мусора. Их можно утилизировать с помощью компактных компостеров, установленных в офисной кухне. Полученный компост в дальнейшем можно использовать для озеленения офисного пространства или передавать сотрудникам для личных нужд. Это не только сократит количество отходов, но и создаст дополнительную связь между коллективом и экологическими инициативами компании.

Отдельной задачей является утилизация электронных отходов – еще 5% мусора в IT-среде. Для этого необходимо организовать специальные пункты сбора старых кабелей, батареек и нерабочей техники. Важно заключить договоры с лицензированными организациями, которые обеспечат безопасную переработку или утилизацию такого рода отходов. Дополнительно можно проводить регулярные акции по сбору электронного мусора, вовлекая в процесс всех сотрудников и повышая их экологическую грамотность.

Все эти меры в комплексе способствуют созданию комфортной и экологически безопасной рабочей среды для программистов. Они не только снижают негативное воздействие на природу, но и формируют корпоративную культуру, основанную на принципах устойчивого развития. Внедрение таких практик делает компанию более привлекательной для клиентов и партнеров, разделяющих ценности ответственного отношения к окружающей среде.

5.4 Пожарная безопасность рабочей зоны

Электропитание оборудования в рабочей зоне разработчика осуществляется от стандартной сети переменного тока с напряжением 220–230 В и частотой 50 Гц. Все применяемые устройства относятся к нулевому классу электробезопасности, что означает наличие только основной изоляции без дополнительных мер, таких как заземление.

Риски поражения электрическим током могут возникнуть при:

* прямом контакте с оголёнными токоведущими элементами электронного оборудования;
* неисправности в конструкции устройств;
* повреждении розеток или внутренней электропроводки.

Для обеспечения электрической безопасности на рабочем месте требуется:

* надёжно изолировать все потенциально опасные участки оборудования с применением защитных оболочек и экранов;
* обеспечить заземление металлических конструкций, которые при аварии могут оказаться под напряжением;
* регулярно проводить инструктаж по охране труда с акцентом на правила работы с электрооборудованием.

Выбор средств изоляции обязан учитывать рабочие характеристики сети и устройств. Изоляционные материалы на всех доступных частях техники должны исключать возможность поражения током или возникновения электрической дуги. В данной ситуации, оборудование полностью изолировано, и дополнительные меры в виде заземления не требуются, что соответствует его классу безопасности.

Помещение, в котором располагается оборудование, относится к III степени огнестойкости и классифицируется по пожарной опасности как категория П. Основные потенциальные причины возгорания включают:

* эксплуатацию неисправной техники с внутренними повреждениями;
* короткие замыкания;
* перегрузку электросети за счёт одновременного подключения мощных потребителей;
* нарушение требований пожарной безопасности;
* износ или повреждение электропроводки.

Пожарная безопасность обеспечивается комплексным подходом, сочетающим превентивные меры и средства активной защиты, включая установку огнетушителей (например, ОХП-10 и ОУ-5), проведение технического обслуживания и инструктажей, а также организационные мероприятия. Согласно нормативам, риск воздействия опасных факторов пожара на человека не должен превышать одного случая на миллион человек в год. Для достижения этого необходимо внедрение эффективных систем пожарной сигнализации и тушения, регулярный контроль состояния электросети и использование безопасных отопительных систем.

5.5 Выводы по главе

В данной главе был проведён анализ условий труда при разработке программного обеспечения, включая оценку факторов, влияющих на здоровье и безопасность сотрудников. Сформированы рекомендации по организации безопасной рабочей среды, которые подлежат включению в план первичного инструктажа на рабочем месте. Особый акцент сделан на параметрах освещения: проведён расчет системы общего равномерного освещения с учётом действующих санитарных норм и требований. Полученные результаты подтверждают, что при соблюдении предложенных технических и организационных мер возможно обеспечить безопасные и комфортные условия труда для программиста.

Заключение

В ходе работы реализована идея создания веб-приложения, объединяющая визуализацию и генерацию звука на основе алгоритмических моделей. Реализованное приложение демонстрирует эффективность использования фрактальных структур для создания уникальных музыкальных композиций с возможностью гибкой настройки параметров.

Использование современных технологий, таких как Python с Flask на сервере и JavaScript с библиотекой Tone.js на клиенте, обеспечило возможность создания такого приложения, объединяющего творческие и математические функции с удобным, интуитивно понятным пользовательским интерфейсом.

Особое внимание уделено организации хранения данных с применением реляционной базы PostgreSQL. Она обеспечивает сохранность, целостность и безопасность пользовательских данных, включая информацию об учетных записях, музыкальных композициях и избранных треках. Интеграция базы данных с серверной частью позволяет эффективно управлять пользовательскими сессиями и историей действий, что расширяет функциональные возможности системы.

Основной результат работы – функциональный веб-сайт, позволяющее пользователю взаимодействовать с фрактальными структурами, прослушивать сгенерированные музыкальные композиции и наблюдать их визуализацию в реальном времени.

В перспективе возможны улучшения – добавление алгоритмов ИИ для генерации более сложных музыкальных структур, расширение палитры визуализации, и мобильных платформ.

Таким образом, работа достигла поставленных целей, а её результаты могут быть применимы в сфере цифрового искусства, образования и музыкальных технологий.

Перечень используемых информационных источников

1. Андреев, А. JavaScript и визуализация данных / А. Андреев. – Санкт-Петербург: БХВ-Петербург, 2021. – 336 с.
2. Бон, Д. Генеративное искусство: практическое руководство по использованию Processing / Д. Бон. – Москва: ДМК Пресс, 2016. – 240 с. – ISBN 978-5-97060-123-4.
3. Бондаренко, А.В. Web-программирование на JavaScript /   
   А.В. Бондаренко. – Москва: Лань, 2021. – 416 с.
4. Войцеховский, С.А. Основы фрактальной графики / С.А. Войцеховский. – Санкт-Петербург: Питер, 2003. – 256 с.
5. Головнин, С.А. Web Audio API: основы синтеза звука в браузере /   
   С.А. Головнин / Программные продукты и системы. – 2019. – 33 с.
6. Губарев, В.С. Введение в WebGL: графика нового поколения /   
   В.С. Губарев. – Москва: ДМК Пресс, 2014. – 304 с.
7. Коган, М. Креативное программирование: принципы и практика /   
   М. Коган. – Москва: ДМК Пресс, 2019. – 312 с. – ISBN 978-5-97060-456-3.
8. Кутепов, А.В. Основы программирования графики на WebGL /   
   А.В. Кутепов. – Москва: ДМК Пресс, 2022. – 220 с.
9. Маккормак, Дж. Компьютеры и творчество / Дж. Маккормак,   
   М. Д'Инверно. – Москва: ДМК Пресс, 2014. – 432 с. – ISBN 978-5-97060-789-2.
10. Мандельброт, Б. Фрактальная геометрия природы / Б. Мандельброт. – Москва: Институт компьютерных исследований, 2002. – 468 с.
11. Митчелл, М. Сложность: экскурсия по науке о сложных системах /   
    М. Митчелл. – Москва: Альпина, 2011. – 349 с. – ISBN 978-5-91671-123-4.
12. Хольцман, Л. Фрактальный хаос в цифровом искусстве / Л. Хольцман. – Санкт-Петербург: Наука, 2006. – 198 с.
13. Шиффман, Д. Природа кода / Д. Шиффман. – Москва: ДМК Пресс, 2022. – 520 с. – ISBN 978-5-97060-789-3.
14. Эно, Б. Генеративная музыка и искусство мышления в циклах / Б. Эно. – Лондон: Фабер, 2017. – 112 с. – ISBN 978-0-571-33000-3.
15. Юрасов, В.В. Алгоритмическое искусство: введение в теорию и практику / В.В. Юрасов. – Москва: URSS, 2021. – 256 с.
16. Юферева, С. Генеративное искусство: теория и практика / С. Юферева. – Москва: НИУ ВШЭ, 2020. – 234 с.

Приложение А Техническое задание

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Руководитель ВКР  \_\_\_\_\_\_\_\_\_\_ Т.А. Медведева  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г. | УТВЕРЖДЕНО  Зав. кафедрой «ПОВТиАС»  \_\_\_\_\_\_\_\_\_\_ В.В. Долгов  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2025 г. |

**А.1 Общие сведения**

**А.1.1 Наименование программного средства**

Веб-сайт «Fractune».

**А.1.2 Область применения**

Разработанное программное средство может быть использовано в сфере цифрового искусства для генерации музыки на основе реализованных фракталов и в научных исследованиях в качестве инструмента для изучения гармонических закономерностей. Оно также может применяться в образовании для изучения предметов, связанных с фрактальной геометрией и областью алгоритмических композиций.

**А.2 Основание для разработки**

Разработка ведется на основании документа «Учебный план для студентов ВУЗа» направление 02.03.03 «Математическое обеспечение и администрирование информационных систем» кафедры «Программное обеспечение вычислительное техники и автоматизированных систем» факультета «Информатика и вычислительная техника» Донского Государственного Технического Университета.

**А.3 Назначение разработки**

**А.3.1 Функциональное назначение**

Функциональное назначение программного средства заключается в генерации музыкальных композиций и их визуализации на основе фрактальных алгоритмов с возможностью интерактивной настройки параметров пользователем.

#### **А.3.2 Эксплуатационное назначение**

Эксплуатационное назначение состоит в использовании программного средства на компьютере.

#### **A.4 Требования к программе**

#### **А.4.1 Требования к функциональным характеристикам**

Программное средство должно осуществлять следующий функционал:

* регистрация пользователя в веб-сайте и хранение его данных (имя, e-mail, пароль, сохраненные и избранные композиции);
* выбор для генерации фракталов в двух режимах: IFS и L-системы;
* выбор моделей: дерево, дракон, снежинка Коха, Мандельброт, папоротник;
* поддержка настраиваемых параметров и правил построения фракталов;
* визуализация структуры фрактала на холсте canvas;
* преобразование фрактала в музыкальные шаблоны с использованием Tone.js;
* независимая генерация мелодии, баса и ударных по отдельным фракталам;
* воспроизведение каждой дорожки по отдельности и с объединением;
* круговая и линейная визуализация звука в реальном времени;
* сохранение и загрузка пользовательских композиций;
* управление избранными композициями пользователя;
* просмотр сохранённых и избранных композиций через вкладки;
* экспорт музыки в аудиофайл в форматах \*.wav;
* вход и выход пользователя.

#### **А.4.2 Требования к надежности**

Надежное функционирование программы должно быть обеспечено выполнением Заказчиком совокупности организационно–технических мероприятий, перечень которых приведен ниже:

* организация бесперебойного питания технических средств;
* организация логики работы модулей генерации фракталов и звука;
* защиты программного средства от несанкционированного проникновения;
* копирование (архивирование) данных, необходимых для восстановления работы программного средства;
* регулярным выполнением требований ГОСТ 51188-98. Защита информации. Испытания программных средств на наличие вирусов.

#### **А.4.2.1 Входные данные**

Входные данные формируются через веб-интерфейс и включают:

* данные пользователя (имя, e-mail, пароль);
* типы фрактальной модели для каждого звукового слоя;
* тип генерации фрактала для мелодии, баса и ударных;
* параметры глубины рекурсии;
* правила генерации, для режима L-системы – аксиомы и вероятности (в текстовом формате);
* тип музыкальной дорожки (мелодия, бас, ударные);
* название композиции.

#### **А.4.2.2 Выходные данные**

Выходные данные включают фрактальные визуализации в canvas, звуковые паттерны через Tone.js, список сохранённых и избранных композиций, экспортированные музыкальные файлы и параметры генерации.

#### **А.4.3 Условия эксплуатации**

Для функционирования программного продукта необходимо соблюдение всех требований и правил эксплуатации компьютерной техники. Для работы с программным средством необходимы навыки работы с персональным компьютером на уровне пользователя. Дополнительных требований и ограничений не вводится.

#### **А.4.4 Требование к составу и параметрам технических средств**

Состав технических средств пользователя:

* процессор не менее 4 ядер и 2.8 ГГц;
* оперативная память объемом не менее 8 Гб.

#### **А.4.5 Требования к исходным кодам и языкам программирования**

Программное обеспечение создано с применением Python и веб-фреймворка Flask для бэкенда, а также базы данных PostgreSQL для хранения информации о пользователях и музыкальных произведениях. Клиентская часть построена на JavaScript с использованием библиотеки Tone.js, отвечающей за синтез и воспроизведение фрактальных композиций, а оформление выполнено с помощью HTML и Tailwind CSS. Для отображения фрактальных изображений применяется элемент canvas. Система рассчитана на работу в современных браузерах и требует установленной среды с поддержкой Python и СУБД PostgreSQL.

#### **А.4.7 Требования к транспортировке и хранению**

Условия транспортирования, места хранения, условия складирования и сроки хранения в различных условиях должны соответствовать требованиям, предъявляемым к носителям информации, на которых будет содержаться данное программное изделие.

Допустимы все способы транспортирования и хранения, не нарушающие целостность используемого носителя данных. Программное средство может храниться на любом носителе информации, имеющее возможность подключения к персональному компьютеру.

#### **А.5 Требования к программной документации**

Программная документация должна состоять из следующих листов:

* титульный лист;
* пояснительная записка к дипломной работе;
* техническое задание по ГОСТ 15.016-2016 ЕСПД;
* исходный код программного средства по ГОСТ Р 51904-2002 ЕСПД.

#### **А.6 Стадии и этапы разработки**

Стадии и этапы разработки программного средства:

* постановка задачи (с 21.04.25 по 27.04.25);
* изучение предметной области (с 27.04.25 по 29.04.25);
* разработка алгоритма решения задачи (с 29.04.25 по 07.05.25);
* разработка программы (с 07.05.25 по 14.05.25);
* тестирование программы (с 14.05.25 по 16.05.25);
* подготовка отчета (с 16.05.25 по 17.05.25).

#### **А.7 Порядок контроля и приемки**

Порядок и контроль приемки определяются заведующим кафедрой «ПОВТиАС» и основаны на демонстрации знаний технологии и умении создавать программные средства для различных предметных областей.

Главным требованием к приемке является наличие правильно работающего программного средства с тестовым примером и отчета, представленного в печатном виде.

|  |  |
| --- | --- |
| Разработчик технического задания:  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г. | /Евсюкова Софья Александровна/  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Приложение Б Исходный код программного средства

Листинг 1 – Исходный код файла app.py

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route(«/»)

def index():

return render\_template(«index.html»)

if \_\_name\_\_ == «\_\_main\_\_»:

app.run(debug=True)

Листинг 2 – Исходный код файла script.js

class FractalMusicSystem {

constructor() {

this.presetsHistory = [];

this.currentPresetId = 0;

// Инициализация компонентов

this.melody = {

type: 'tree',

depth: 3,

canvas: document.getElementById('melody-canvas'),

ctx: document.getElementById('melody-canvas').getContext('2d'),

rules: this.getDefaultRules('tree')

};

this.bass = {

type: 'dragon',

depth: 3,

canvas: document.getElementById('bass-canvas'),

ctx: document.getElementById('bass-canvas').getContext('2d'),

rules: this.getDefaultRules('dragon')

};

this.drums = {

type: 'barnsley',

depth: 3,

canvas: document.getElementById('drums-canvas'),

ctx: document.getElementById('drums-canvas').getContext('2d'),

rules: this.getDefaultRules('barnsley')

};

// Инициализация Tone.js

this.melodySynth = new Tone.PolySynth(Tone.Synth).toDestination();

this.melodySynth.volume.value = -8;

this.bassSynth = new Tone.MonoSynth({

oscillator: { type: «sine» },

envelope: { attack: 0.1, decay: 0.3, sustain: 0.4, release: 1 }

}).toDestination();

this.bassSynth.volume.value = -12;

// В конструкторе оставьте только один синтезатор:

this.drumSynth = new Tone.MembraneSynth({

pitchDecay: 0.05,

octaves: 5,

oscillator: { type: «triangle» },

envelope: {

attack: 0.001,

decay: 0.4,

sustain: 0.01,

release: 0.4,

attackCurve: «exponential»

}

}).toDestination();

this.drumSynth.volume.value = -6;

this.drumReverb = new Tone.Reverb(1.5).toDestination();

this.drumSynth.connect(this.drumReverb);

this.initElements();

this.setupEventListeners();

this.updateAllEditors();

this.drawAllPreviews();

this.loadPresetsFromLocalStorage();

initElements() {

this.elements = {

// Melody

melodyType: document.getElementById('melody-type'),

melodyDepth: document.getElementById('melody-depth'),

melodyDepthValue: document.getElementById('melody-depth-value'),

melodyRules: document.getElementById('melody-rules'),

melodyApply: document.getElementById('melody-apply'),

melodyReset: document.getElementById('melody-reset'),

// Bass

bassType: document.getElementById('bass-type'),

bassDepth: document.getElementById('bass-depth'),

bassDepthValue: document.getElementById('bass-depth-value'),

bassRules: document.getElementById('bass-rules'),

bassApply: document.getElementById('bass-apply'),

bassReset: document.getElementById('bass-reset'),

// Drums

drumsType: document.getElementById('drums-type'),

drumsDepth: document.getElementById('drums-depth'),

drumsDepthValue: document.getElementById('drums-depth-value'),

drumsRules: document.getElementById('drums-rules'),

drumsApply: document.getElementById('drums-apply'),

drumsReset: document.getElementById('drums-reset'),

// Playback

playAll: document.getElementById('play-all'),

stopAll: document.getElementById('stop-all'),

playMelody: document.getElementById('play-melody'),

playBass: document.getElementById('play-bass'),

playDrums: document.getElementById('play-drums'),

// Presets

savePreset: document.getElementById('save-preset'),

exportMusic: document.getElementById('export-music'),

presetsHistory: document.getElementById('presets-history')

};

setupEventListeners() {

// Melody controls

this.elements.melodyType.addEventListener('change', (e) => {

this.melody.type = e.target.value;

this.melody.rules = this.getDefaultRules(this.melody.type);

this.updateEditor('melody');

this.drawPreview('melody');

});

this.elements.melodyApply.addEventListener('click', () => {

try {

this.melody.rules = JSON.parse(this.elements.melodyRules.value);

this.drawPreview('melody');

} catch (e) {

alert(`Ошибка в формате JSON: ${e.message}`);

});

this.elements.melodyReset.addEventListener('click', () => {

this.melody.rules = this.getDefaultRules(this.melody.type);

this.updateEditor('melody');

this.drawPreview('melody');

});

this.elements.melodyDepth.addEventListener('input', (e) => {

this.melody.depth = parseInt(e.target.value);

this.elements.melodyDepthValue.textContent = this.melody.depth;

this.drawPreview('melody');

});

// Bass controls

this.elements.bassType.addEventListener('change', (e) => {

this.bass.type = e.target.value;

this.bass.rules = this.getDefaultRules(this.bass.type);

this.updateEditor('bass');

this.drawPreview('bass');

});

this.elements.bassApply.addEventListener('click', () => {

try {

this.bass.rules = JSON.parse(this.elements.bassRules.value);

this.drawPreview('bass');

} catch (e) {

alert(`Ошибка в формате JSON: ${e.message}`);

});

this.elements.bassReset.addEventListener('click', () => {

this.bass.rules = this.getDefaultRules(this.bass.type);

this.updateEditor('bass');

this.drawPreview('bass');

});

this.elements.bassDepth.addEventListener('input', (e) => {

this.bass.depth = parseInt(e.target.value);

this.elements.bassDepthValue.textContent = this.bass.depth;

this.drawPreview('bass');

});

// Drums controls

this.elements.drumsType.addEventListener('change', (e) => {

this.drums.type = e.target.value;

this.drums.rules = this.getDefaultRules(this.drums.type);

this.updateEditor('drums');

this.drawPreview('drums');

});

this.elements.drumsApply.addEventListener('click', () => {

try {

this.drums.rules = JSON.parse(this.elements.drumsRules.value);

this.drawPreview('drums');

} catch (e) {

alert(`Ошибка в формате JSON: ${e.message}`);

});

this.elements.drumsReset.addEventListener('click', () => {

this.drums.rules = this.getDefaultRules(this.drums.type);

this.updateEditor('drums');

this.drawPreview('drums');

});

this.elements.drumsDepth.addEventListener('input', (e) => {

this.drums.depth = parseInt(e.target.value);

this.elements.drumsDepthValue.textContent = this.drums.depth;

this.drawPreview('drums');

});

// Playback controls

this.elements.playAll.addEventListener('click', () => this.playAll());

this.elements.stopAll.addEventListener('click', () => Tone.Transport.stop());

this.elements.playMelody.addEventListener('click', () => this.playComponent('melody'));

this.elements.playBass.addEventListener('click', () => this.playComponent('bass'));

this.elements.playDrums.addEventListener('click', () => {

this.elements.playDrums.classList.add('bg-green-600');

setTimeout(() => {

this.elements.playDrums.classList.remove('bg-green-600');

}, 200);

this.playComponent('drums');

});

this.elements.savePreset.addEventListener('click', () => this.saveCurrentPreset());

this.elements.exportMusic.addEventListener('click', () => this.exportMusic());

// Сохранение текущего пресета

saveCurrentPreset() {

const preset = {

id: Date.now(),

date: new Date().toLocaleString(),

melody: {

type: this.melody.type,

depth: this.melody.depth,

rules: this.melody.rules

},

bass: {

type: this.bass.type,

depth: this.bass.depth,

rules: this.bass.rules

},

drums: {

type: this.drums.type,

depth: this.drums.depth,

rules: this.drums.rules

};

this.presetsHistory.unshift(preset);

this.savePresetsToLocalStorage();

this.renderPresetsHistory();

// Создаем временную кнопку для скачивания JSON

this.downloadObjectAsJson(preset, `fractal\_music\_preset\_${preset.id}`);

// Загрузка пресета

loadPreset(preset) {

this.melody.type = preset.melody.type;

this.melody.depth = preset.melody.depth;

this.melody.rules = preset.melody.rules;

this.bass.type = preset.bass.type;

this.bass.depth = preset.bass.depth;

this.bass.rules = preset.bass.rules;

this.drums.type = preset.drums.type;

this.drums.depth = preset.drums.depth;

this.drums.rules = preset.drums.rules;

// Обновляем UI

this.elements.melodyType.value = this.melody.type;

this.elements.melodyDepth.value = this.melody.depth;

this.elements.melodyDepthValue.textContent = this.melody.depth;

this.elements.bassType.value = this.bass.type;

this.elements.bassDepth.value = this.bass.depth;

this.elements.bassDepthValue.textContent = this.bass.depth;

this.elements.drumsType.value = this.drums.type;

this.elements.drumsDepth.value = this.drums.depth;

this.elements.drumsDepthValue.textContent = this.drums.depth;

this.updateAllEditors();

this.drawAllPreviews();

// Рендер истории пресетов

renderPresetsHistory() {

this.elements.presetsHistory.innerHTML = '';

if (this.presetsHistory.length === 0) {

this.elements.presetsHistory.innerHTML = '<p class=«text-gray-400»>История пуста</p>';

return;

this.presetsHistory.forEach(preset => {

const presetElement = document.createElement('div');

presetElement.className = 'bg-gray-700 p-3 rounded flex justify-between items-center';

presetElement.innerHTML = `

<div>

<span class=«font-medium»>Пресет #${preset.id}</span>

<span class=«text-sm text-gray-400»>${preset.date}</span>

</div>

<div class=«space-x-2»>

<button class=«load-preset bg-blue-600 hover:bg-blue-700 px-3 py-1 rounded text-sm» data-id=«${preset.id}»>

Загрузить

</button>

<button class=«export-preset bg-green-600 hover:bg-green-700 px-3 py-1 rounded text-sm» data-id=«${preset.id}»>

Экспорт

</button>

</div>

;

this.elements.presetsHistory.appendChild(presetElement);

// Добавляем обработчики для новых кнопок

presetElement.querySelector('.load-preset').addEventListener('click', () => {

const presetToLoad = this.presetsHistory.find(p => p.id === preset.id);

if (presetToLoad) this.loadPreset(presetToLoad);

});

presetElement.querySelector('.export-preset').addEventListener('click', () => {

const presetToExport = this.presetsHistory.find(p => p.id === preset.id);

if (presetToExport) {

this.downloadObjectAsJson(presetToExport, `fractal\_music\_preset\_${presetToExport.id}`);

});

});

// Сохранение в LocalStorage

savePresetsToLocalStorage() {

localStorage.setItem('fractalMusicPresets', JSON.stringify(this.presetsHistory));

// Загрузка из LocalStorage

loadPresetsFromLocalStorage() {

const savedPresets = localStorage.getItem('fractalMusicPresets');

if (savedPresets) {

this.presetsHistory = JSON.parse(savedPresets);

this.renderPresetsHistory();

// Экспорт музыки в WAV

exportMusic() {

// Создаем рекордер

const recorder = new Tone.Recorder();

this.melodySynth.connect(recorder);

this.bassSynth.connect(recorder);

this.drumSynth.connect(recorder);

// Начинаем запись

recorder.start();

// Проигрываем композицию

this.playAll();

// Через 8 секунд (длина композиции) останавливаем запись

setTimeout(async () => {

Tone.Transport.stop();

const recording = await recorder.stop();

const url = URL.createObjectURL(recording);

// Создаем ссылку для скачивания

const anchor = document.createElement('a');

anchor.download = `fractal\_music\_${Date.now()}.wav`;

anchor.href = url;

anchor.click();

}, 8000);

// Вспомогательная функция для скачивания JSON

downloadObjectAsJson(exportObj, exportName) {

const dataStr = «data:text/json;charset=utf-8,» + encodeURIComponent(JSON.stringify(exportObj, null, 2));

const downloadAnchorNode = document.createElement('a');

downloadAnchorNode.setAttribute(«href», dataStr);

downloadAnchorNode.setAttribute(«download», exportName + «.json»);

document.body.appendChild(downloadAnchorNode);

downloadAnchorNode.click();

downloadAnchorNode.remove();

getDefaultRules(type) {

const defaults = {

tree: {

angle: 45,

lengthFactor: 0.67,

branches: 2,

initLength: 80,

color: '#ff6b6b'

},

koch: {

segments: 4,

angle: 60,

scaleFactor: 1/3,

color: '#48dbfb'

},

mandelbrot: {

maxIterations: 100,

zoom: 1,

offsetX: 0,

offsetY: 0,

color: '#1dd1a1'

},

dragon: {

color: '#feca57',

angle: 45,

scaleFactor: 0.7

},

barnsley: {

color: '#5f27cd',

points: 10000

};

return defaults[type];

updateAllEditors() {

this.updateEditor('melody');

this.updateEditor('bass');

this.updateEditor('drums');

updateEditor(component) {

this.elements[`${component}Rules`].value = JSON.stringify(this[component].rules, null, 2);

drawAllPreviews() {

this.drawPreview('melody');

this.drawPreview('bass');

this.drawPreview('drums');

drawPreview(component) {

const { ctx, canvas, type, rules, depth } = this[component];

ctx.clearRect(0, 0, canvas.width, canvas.height);

ctx.save();

switch(type) {

case 'tree':

this.drawTree(ctx, canvas, rules, depth);

break;

case 'koch':

this.drawKoch(ctx, canvas, rules, depth);

break;

case 'mandelbrot':

this.drawMandelbrot(ctx, canvas, rules);

break;

case 'dragon':

this.drawDragon(ctx, canvas, rules, depth);

break;

case 'barnsley':

this.drawBarnsleyFern(ctx, canvas, rules);

break;

default:

console.warn(`Unknown fractal type: ${type}`);

ctx.restore();

// Методы отрисовки фракталов (аналогичные оригинальным, но с передачей параметров)

drawTree(ctx, canvas, rules, depth) {

ctx.strokeStyle = rules.color || '#ff6b6b';

ctx.lineWidth = 2;

ctx.translate(canvas.width / 2, canvas.height);

this.drawBranch(

ctx,

rules.initLength || 80,

depth,

rules.angle \* Math.PI / 180,

rules.lengthFactor || 0.67,

rules.branches || 2

);

drawBranch(ctx, length, depth, angle, lengthFactor, branches) {

if (depth <= 0) return;

ctx.beginPath();

ctx.moveTo(0, 0);

ctx.lineTo(0, -length);

ctx.stroke();

ctx.translate(0, -length);

for (let i = 0; i < branches; i++) {

const branchAngle = angle \* (i - (branches - 1) / 2);

ctx.save();

ctx.rotate(branchAngle);

this.drawBranch(ctx, length \* lengthFactor, depth - 1, angle, lengthFactor, branches);

ctx.restore();

drawKoch(ctx, canvas, rules, depth) {

ctx.strokeStyle = rules.color || '#48dbfb';

ctx.lineWidth = 1;

const sideLength = Math.min(canvas.width, canvas.height) \* 0.6;

const height = sideLength \* Math.sqrt(3) / 2;

const centerX = canvas.width / 2;

const centerY = canvas.height / 2;

const angleDeg = rules.angle || 60;

const angleRad = angleDeg \* Math.PI / 180;

const scaleFactor = rules.scaleFactor || 1 / 3;

const p1 = {

x: centerX,

y: centerY - (2 / 3) \* height

};

const p2 = {

x: centerX - sideLength / 2,

y: centerY + (1 / 3) \* height

};

const p3 = {

x: centerX + sideLength / 2,

y: centerY + (1 / 3) \* height

};

this.drawKochSide(ctx, p1, p2, depth, angleRad, scaleFactor);

this.drawKochSide(ctx, p2, p3, depth, angleRad, scaleFactor);

this.drawKochSide(ctx, p3, p1, depth, angleRad, scaleFactor);

drawKochSide(ctx, p1, p2, depth, angleRad, scaleFactor) {

if (depth === 0) {

ctx.beginPath();

ctx.moveTo(p1.x, p1.y);

ctx.lineTo(p2.x, p2.y);

ctx.stroke();

return;

const dx = p2.x - p1.x;

const dy = p2.y - p1.y;

const x1 = p1.x + dx \* scaleFactor;

const y1 = p1.y + dy \* scaleFactor;

const x2 = p1.x + dx \* (1 - scaleFactor);

const y2 = p1.y + dy \* (1 - scaleFactor);

const angle = Math.atan2(dy, dx);

const length = Math.sqrt((dx \* dx + dy \* dy)) \* scaleFactor;

const xPeak = x1 + length \* Math.cos(angle - angleRad);

const yPeak = y1 + length \* Math.sin(angle - angleRad);

this.drawKochSide(ctx, p1, { x: x1, y: y1 }, depth - 1, angleRad, scaleFactor);

this.drawKochSide(ctx, { x: x1, y: y1 }, { x: xPeak, y: yPeak }, depth - 1, angleRad, scaleFactor);

this.drawKochSide(ctx, { x: xPeak, y: yPeak }, { x: x2, y: y2 }, depth - 1, angleRad, scaleFactor);

this.drawKochSide(ctx, { x: x2, y: y2 }, p2, depth - 1, angleRad, scaleFactor);

drawKochLine(ctx, x1, y1, x2, y2, depth, segments, angle, scaleFactor) {

if (depth === 0) {

ctx.beginPath();

ctx.moveTo(x1, y1);

ctx.lineTo(x2, y2);

ctx.stroke();

return;

const dx = x2 - x1;

const dy = y2 - y1;

const length = Math.sqrt(dx \* dx + dy \* dy) \* scaleFactor;

const initialAngle = Math.atan2(dy, dx);

let points = [{x: x1, y: y1}];

let currentX = x1;

let currentY = y1;

for (let i = 1; i <= segments; i++) {

const segmentAngle = initialAngle + angle \* (i - (segments - 1)/2);

currentX += length \* Math.cos(segmentAngle);

currentY += length \* Math.sin(segmentAngle);

points.push({x: currentX, y: currentY});

for (let i = 0; i < points.length - 1; i++) {

this.drawKochLine(

ctx,

points[i].x, points[i].y,

points[i+1].x, points[i+1].y,

depth - 1,

segments,

angle,

scaleFactor

);

drawMandelbrot(ctx, canvas, rules) {

const { maxIterations, zoom, offsetX, offsetY, color } = rules;

const width = canvas.width;

const height = canvas.height;

const imageData = ctx.createImageData(width, height);

const data = imageData.data;

for (let x = 0; x < width; x++) {

for (let y = 0; y < height; y++) {

let zx = 0, zy = 0;

const cx = (x - width / 2) / (200 \* zoom) + offsetX;

const cy = (y - height / 2) / (200 \* zoom) + offsetY;

let i = 0;

while (zx \* zx + zy \* zy < 4 && i < maxIterations) {

const tmp = zx \* zx - zy \* zy + cx;

zy = 2 \* zx \* zy + cy;

zx = tmp;

i++;

const p = (y \* width + x) \* 4;

const brightness = i === maxIterations ? 0 : (i / maxIterations) \* 255;

data[p] = parseInt(color.slice(1, 3), 16) \* brightness / 255;

data[p + 1] = parseInt(color.slice(3, 5), 16) \* brightness / 255;

data[p + 2] = parseInt(color.slice(5, 7), 16) \* brightness / 255;

data[p + 3] = 255;

ctx.putImageData(imageData, 0, 0);

drawDragon(ctx, canvas, rules, depth) {

const { color } = rules;

ctx.strokeStyle = color;

ctx.lineWidth = 1;

// L-система

let axiom = «FX»;

for (let i = 0; i < depth; i++) {

let next = ««;

for (let char of axiom) {

if (char === «X») {

next += «X+YF+»;

} else if (char === «Y») {

next += «-FX-Y»;

} else {

next += char;

axiom = next;

// Рисование по L-системе

const step = 10;

let x = canvas.width / 2;

let y = canvas.height / 2;

let angle = 0;

ctx.beginPath();

ctx.moveTo(x, y);

for (let char of axiom) {

if (char === «F») {

const rad = angle \* Math.PI / 180;

x += step \* Math.cos(rad);

y += step \* Math.sin(rad);

ctx.lineTo(x, y);

} else if (char === «+») {

angle += 90;

} else if (char === «-») {

angle -= 90;

ctx.stroke();

drawBarnsleyFern(ctx, canvas, rules) {

const { color, points } = rules;

ctx.fillStyle = color;

let x = 0, y = 0;

for (let i = 0; i < points; i++) {

const r = Math.random();

let nextX, nextY;

if (r < 0.01) {

nextX = 0;

nextY = 0.16 \* y;

} else if (r < 0.86) {

nextX = 0.85 \* x + 0.04 \* y;

nextY = -0.04 \* x + 0.85 \* y + 1.6;

} else if (r < 0.93) {

nextX = 0.2 \* x - 0.26 \* y;

nextY = 0.23 \* x + 0.22 \* y + 1.6;

} else {

nextX = -0.15 \* x + 0.28 \* y;

nextY = 0.26 \* x + 0.24 \* y + 0.44;

x = nextX;

y = nextY;

const plotX = canvas.width / 2 + x \* 30;

const plotY = canvas.height - y \* 30;

ctx.fillRect(plotX, plotY, 1, 1);

// Музыкальные методы

playAll() {

Tone.Transport.cancel();

Tone.Transport.bpm.value = 120;

// Основная мелодия

const melodyPattern = this.generateMusicPattern('melody');

const melodyPart = new Tone.Part((time, note) => {

this.melodySynth.triggerAttackRelease(note.note, note.duration, time);

}, melodyPattern.map((note, i) => ({

time: i \* 0.5,

note: this.getNoteForMelody(note),

duration: «8n»

})));

melodyPart.loop = true;

melodyPart.loopEnd = «4m»;

melodyPart.start(0);

// Басовая линия

const bassPattern = this.generateMusicPattern('bass');

const bassPart = new Tone.Part((time, note) => {

this.bassSynth.triggerAttackRelease(note.note, note.duration, time);

}, bassPattern.map((note, i) => ({

time: i \* 1,

note: this.getNoteForBass(note),

duration: «4n»

})));

bassPart.loop = true;

bassPart.loopEnd = «4m»;

bassPart.start(0);

// Ударные

const drumsPattern = this.generateMusicPattern('drums');

const drumsPart = new Tone.Part((time, note) => {

this.drumSynth.triggerAttackRelease(note.note, «8n», time, 0.5);

}, drumsPattern.map((note, i) => ({

time: i \* 0.25,

note: this.getDrumForPattern(note).note

})));

drumsPart.loop = true;

drumsPart.loopEnd = «2m»;

drumsPart.start(0);

Tone.Transport.start();

playComponent(component) {

if (Tone.context.state !== 'running') {

Tone.start();

const pattern = this.generateMusicPattern(component);

const now = Tone.now();

switch(component) {

case 'melody':

pattern.forEach((note, i) => {

this.melodySynth.triggerAttackRelease(

this.getNoteForMelody(note),

«8n»,

now + i \* 0.3

);

});

break;

case 'bass':

pattern.forEach((note, i) => {

this.bassSynth.triggerAttackRelease(

this.getNoteForBass(note),

«4n»,

now + i \* 0.5

);

});

break;

case 'drums':

pattern.forEach((note, i) => {

const drum = this.getDrumForPattern(note);

this.drumSynth.triggerAttackRelease(

drum.note,

«8n»,

now + i \* 0.2,

0.5 // velocity

);

});

break;

generateMusicPattern(component) {

const { type, depth, rules } = this[component];

const pattern = [];

const complexity[12] = depth \* 2;

switch(type) {

case 'tree':

const branches = rules.branches || 2;

for (let i = 0; i < complexity[12] \* branches; i++) {

pattern.push(i % 7);

break;

case 'koch':

const noteCount = 3 \* Math.pow(4, depth); // количество отрезков в снежинке

for (let i = 0; i < noteCount; i++) {

// например, цикл по гамме вверх и вниз

const value = i % 14;

const noteIndex = value < 7 ? value : 13 - value; // пилообразно

pattern.push(noteIndex);

break;

case 'mandelbrot':

const maxIterations = rules.maxIterations || 100;

for (let i = 0; i < maxIterations; i++) {

const normalized = Math.floor(i / maxIterations \* 7);

pattern.push(normalized);

break;

case 'dragon':

const dragonPoints = Math.pow(2, depth);

for (let i = 0; i < dragonPoints; i++) {

const angle = Math.sin(i \* Math.PI / dragonPoints) \* 7;

pattern.push(Math.floor((angle + 7) % 7));

break;

case 'barnsley':

const points = rules.points || 10000;

for (let i = 0; i < points / 1000; i++) {

const randomNote = Math.floor(Math.random() \* 7);

pattern.push(randomNote);

break;

default:

console.warn('Неизвестный тип фрактала, музыка не будет сгенерирована.');

break;

return pattern;

getNoteForMelody(index) {

const scales = {

major: [«C4», «D4», «E4», «F4», «G4», «A4», «B4», «C5»],

minor: [«A3», «B3», «C4», «D4», «E4», «F4», «G4», «A4»],

pentatonic: [«C4», «D4», «E4», «G4», «A4»]

};

return scales.major[index % scales.major.length];

getNoteForBass(index) {

const bassNotes = [«C2», «D2», «E2», «F2», «G2», «A2», «B2», «C3»];

return bassNotes[index % bassNotes.length];

getDrumForPattern(index) {

const drums = [

{note: «C2», type: «kick»},

{note: «C3», type: «snare»},

{note: «A2», type: «hihat»}

];

return drums[index % drums.length];

// Инициализация при загрузке страницы

document.addEventListener('DOMContentLoaded', () => {

new FractalMusicSystem();

});

Листинг 3 – Исходный код файла index.html

<!DOCTYPE html>

<html lang=«ru»>

<head>

<meta charset=«UTF-8»>

<title>Фрактальная музыкальная система</title>

<script src=«https://cdn.tailwindcss.com»></script>

<script src=«https://cdnjs.cloudflare.com/ajax/libs/tone/14.8.39/Tone.min.js»></script>

</head>

<body class=«bg-gray-900 text-white font-sans min-h-screen p-8»>

<div class=«max-w-6xl mx-auto»>

<h1 class=«text-3xl font-bold mb-6»>🎵 Фрактальная музыкальная система</h1>

<div class=«grid grid-cols-1 md:grid-cols-3 gap-6»>

<!-- Основная мелодия -->

<div class=«bg-gray-800 p-6 rounded-lg»>

<h2 class=«text-xl font-semibold mb-4»>Основная мелодия</h2>

<select id=«melody-type» class=«bg-gray-700 text-white px-3 py-2 rounded w-full mb-4»>

<option value=«tree»>Дерево</option>

<option value=«koch»>Снежинка Коха</option>

<option value=«mandelbrot»>Мандельброт</option>

</select>

<canvas id=«melody-canvas» width=«300» height=«300» class=«w-full bg-gray-900 rounded mb-4»></canvas>

<div class=«mb-4»>

<label class=«block mb-1»>Глубина:</label>

<input type=«range» id=«melody-depth» min=«1» max=«8» value=«3» class=«w-full»>

<div class=«text-center text-sm» id=«melody-depth-value»>3</div>

</div>

<textarea id=«melody-rules» class=«w-full h-32 bg-gray-700 text-white p-3 rounded font-mono text-sm mb-2»></textarea>

<button id=«melody-apply» class=«bg-green-600 hover:bg-green-700 px-4 py-2 rounded w-full mb-2»>

Применить

</button>

<button id=«melody-reset» class=«bg-gray-600 hover:bg-gray-700 px-4 py-2 rounded w-full»>

Сбросить

</button>

</div>

<!-- Басовая линия -->

<div class=«bg-gray-800 p-6 rounded-lg»>

<h2 class=«text-xl font-semibold mb-4»>Басовая линия</h2>

<select id=«bass-type» class=«bg-gray-700 text-white px-3 py-2 rounded w-full mb-4»>

<option value=«dragon»>Дракон</option>

<option value=«tree»>Дерево</option>

<option value=«koch»>Снежинка Коха</option>

</select>

<canvas id=«bass-canvas» width=«300» height=«300» class=«w-full bg-gray-900 rounded mb-4»></canvas>

<div class=«mb-4»>

<label class=«block mb-1»>Глубина:</label>

<input type=«range» id=«bass-depth» min=«1» max=«8» value=«3» class=«w-full»>

<div class=«text-center text-sm» id=«bass-depth-value»>3</div>

</div>

<textarea id=«bass-rules» class=«w-full h-32 bg-gray-700 text-white p-3 rounded font-mono text-sm mb-2»></textarea>

<button id=«bass-apply» class=«bg-green-600 hover:bg-green-700 px-4 py-2 rounded w-full mb-2»>

Применить

</button>

<button id=«bass-reset» class=«bg-gray-600 hover:bg-gray-700 px-4 py-2 rounded w-full»>

Сбросить

</button>

</div>

<!-- Ударные -->

<div class=«bg-gray-800 p-6 rounded-lg»>

<h2 class=«text-xl font-semibold mb-4»>Ударные</h2>

<select id=«drums-type» class=«bg-gray-700 text-white px-3 py-2 rounded w-full mb-4»>

<option value=«barnsley»>Папоротник</option>

<option value=«mandelbrot»>Мандельброт</option>

<option value=«dragon»>Дракон</option>

</select>

<canvas id=«drums-canvas» width=«300» height=«300» class=«w-full bg-gray-900 rounded mb-4»></canvas>

<div class=«mb-4»>

<label class=«block mb-1»>Глубина:</label>

<input type=«range» id=«drums-depth» min=«1» max=«8» value=«3» class=«w-full»>

<div class=«text-center text-sm» id=«drums-depth-value»>3</div>

</div>

<textarea id=«drums-rules» class=«w-full h-32 bg-gray-700 text-white p-3 rounded font-mono text-sm mb-2»></textarea>

<button id=«drums-apply» class=«bg-green-600 hover:bg-green-700 px-4 py-2 rounded w-full mb-2»>

Применить

</button>

<button id=«drums-reset» class=«bg-gray-600 hover:bg-gray-700 px-4 py-2 rounded w-full»>

Сбросить

</button>

</div>

</div>

<div class=«mt-6 bg-gray-800 p-6 rounded-lg»>

<h2 class=«text-xl font-semibold mb-4»>Управление воспроизведением</h2>

<div class=«flex space-x-4»>

<button id=«play-all» class=«bg-blue-600 hover:bg-blue-700 px-6 py-3 rounded-lg text-lg flex-1»>

▶️ Проиграть все

</button>

<button id=«stop-all» class=«bg-red-600 hover:bg-red-700 px-6 py-3 rounded-lg text-lg flex-1»>

⏹ Остановить

</button>

</div>

<div class=«mt-4 grid grid-cols-3 gap-4»>

<button id=«play-melody» class=«bg-purple-600 hover:bg-purple-700 px-4 py-2 rounded»>

Проиграть мелодию

</button>

<button id=«play-bass» class=«bg-orange-600 hover:bg-orange-700 px-4 py-2 rounded»>

Проиграть бас

</button>

<button id=«play-drums» class=«bg-yellow-600 hover:bg-yellow-700 px-4 py-2 rounded»>

Проиграть ударные

</button>

</div>

<div class=«mt-4 grid grid-cols-2 gap-4»>

<button id=«save-preset» class=«bg-indigo-600 hover:bg-indigo-700 px-4 py-2 rounded»>

Сохранить пресет

</button>

<button id=«export-music» class=«bg-green-600 hover:bg-green-700 px-4 py-2 rounded»>

Экспорт музыки

</button>

</div>

</div>

<!-- История пресетов -->

<div class=«mt-6 bg-gray-800 p-6 rounded-lg»>

<h2 class=«text-xl font-semibold mb-4»>История пресетов</h2>

<div id=«presets-history» class=«space-y-2 max-h-64 overflow-y-auto»>

<!-- Пресеты будут добавляться сюда -->

</div>

</div>

</div>

<script src=«/static/script.js»></script>

</body>

</html>

# **Приложение В Протокол предзащиты**

|  |  |
| --- | --- |
|  | Зав. кафедрой «ПОВТиАС»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. В. Долгов  (подпись) «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025г. |

**Протокол**

заседания комиссии по предварительной защите выпускной квалификационной работы

от «\_\_\_» июня 2025 г.

**Присутствовали:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**СЛУШАЛИ:** доклад студентки группы ВМО41 Евсюковой С.А. по защите выпускной квалификационной работы «Веб-сайт для интерактивной генерации музыки и визуализации на основе фракталов».

**Были заданы вопросы:**

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**решено отметить следующее:**

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
4. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПОСТАНОВИЛИ:** доклад студентки о проделанной работе и ответы на вопросы соответствуют требованиям, предъявляемым к выпускным квалификационным работам. Студентка Евсюкова С.А. рекомендована к защите.

Члены комиссии: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/