Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)
(СПбГЭТУ «ЛЭТИ»)

Направление	09.04.04 – Программная инженерия
Профиль	Без профиля
Факультет	КТИ
Кафедра	МО ЭВМ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

ТЕМА: РАЗРАБОТКА АЛГОРИТМА ГЕНЕРАЦИИ ЛАНДШАФТА НА ОСНОВЕ ГРАФА СВЯЗЕЙ ТРЕХМЕРНЫХ МОДЕЛЕЙ

Студент		_____ *подпись*	Скиба А.С.
Руководитель	д.т.н., профессор	_____ *подпись*	Геппенер В.В.
Консультанты		_____ *подпись*	Шевская Н.В.
	к.э.н., доцент	_____ *подпись*	Магомедов М.Н.
	к.т.н.	_____ *подпись*	Заславский М.М.

Санкт-Петербург

2021

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ Кринкин К.В.
«___»_____2021 г.

Студент Скиба А.С. Группа 5304

Тема работы: Разработка алгоритма генерации ландшафта на основе графа связей трехмерных моделей

Место выполнения ВКР: Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), кафедра МО ЭВМ

Исходные данные (технические требования): необходимо разработать программный и визуальный интерфейсы для создания трехмерных ландшафтов, на основе структуры связного ациклического графа.

Содержание ВКР: «Введение», «Анализ предметной области», «Формулировка требований к решению», «Описание метода решения, «Исследование свойств решения», «Оценка и защита результатов интеллектуальной деятельности», «Заключение».

Перечень отчетных материалов: пояснительная записка, презентация.

Дополнительный раздел: Оценка и защита результатов интеллектуальной деятельности.

Дата выдачи задания «08» февраля 2021 г.

Дата представления ВКР к защите «03» июня 2021 г.

Студент _____ Скиба А.С.

Руководитель д.т.н., профессор _____ Геппенер В.В.

Консультант _____ Шевская Н.В.

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ К.В. Кринкин
«___»_____2021 г.

Студент Скиба А.С. Группа 5304
Тема работы: Разработка алгоритма генерации ландшафта на основе графа связей трехмерных моделей

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	08.02 – 22.02
2	Анализ предметной области	22.02 – 10.03
3	Составление требований к приложению	10.03 – 13.03
4	Проектирование приложения	13.03 – 16.03
5	Разработка приложения	16.03 – 05.05
6	Тестирование приложения	05.05 – 07.05
7	Оформление пояснительной записки	07.05 – 14.05
8	Оформление иллюстративного материала	14.05 – 17.05
9	Предзащита	20.05.2021

Студент _____ Скиба А.С.
Руководитель д.т.н., профессор _____ Геппенер В.В.
Консультант _____ Шевская Н.В.

РЕФЕРАТ

Пояснительная записка 90 стр., 34 рис., 6 табл., 29 ист.

ЛАНДШАФТ, ПСЕВДОСЛУЧАЙНАЯ ГЕНЕРАЦИЯ, ГРАФ, ДЕРЕВО КВАДРАНТОВ, ДЕТАЛИЗАЦИЯ, КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ, ТРЕХМЕРНАЯ ГРАФИКА, API, REACT, БИОМ, МЕСТНОСТЬ, МОДЕЛЬ.

Объектом исследования является сервис для получения данных о процедурной генерации ландшафта, как в виде набора бинарных данных, так и в виде визуального представления работы алгоритма.

Целью работы является разработка специализированного алгоритма, реализующего псевдослучайную устойчивую генерацию ландшафта на основе вводных данных, представляющих из себя различные параметры генерации и пользовательские модели.

В данной работе были исследованы программные средства для создания трехмерных ландшафтов. На основе исследования аналогов были составлены общие характеристики разрабатываемого сервиса. Были рассмотрены задачи, которые требуется решить для получения результата. На основе существующих алгоритмов и структур данных были разработаны методы процедурной псевдослучайного устойчивого создания поверхности ландшафта. Исследовался один из возможных способов преобразования структуры связного ациклического графа биомов в климатическую карту будущей планеты. Были рассмотрены критерии для реализации псевдослучайной расстановки объектов на получаемых участках ландшафта. Для оптимизации рендеринга на клиентской стороне были рассмотрены методы построения частичных ландшафтов. В работе также исследовались основные характеристики для визуализации ландшафта на клиентской стороне, а также характеристики программного интерфейса.

ABSTRACT

In this work, software tools for creating three-dimensional landscapes were investigated. Based on the study of analogs, the general characteristics of the developed service were compiled. The tasks that need to be solved to obtain the result were considered. On the basis of existing algorithms and data structures, methods of procedural pseudo-random sustainable terrain surface creation have been developed. One of the possible ways of transforming the structure of a connected acyclic graph of biomes into a climate map of the future planet was investigated. Criteria for the implementation of pseudo-random placement of objects on the resulting areas of the landscape were considered. Methods for constructing partial landscapes were considered to optimize rendering on the client side. The work also explored the main characteristics for rendering the landscape on the client side, as well as the characteristics of the software interface.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

API – программный интерфейс приложения, набор классов и процедур предоставляемые стороннему приложения для расчета результатов или получения данных.

3D – трехмерная графика, в данной работе используются трехмерные модели объектов, а также карт местности, выражающихся в высотах.

Javascript – интерпретируемый язык программирования, поддерживающийся браузерами.

HTML – стандартизированный язык разметки, используется для построения визуального интерфейса приложения в браузерах и оконных приложения операционной системы.

CSS – каскадный язык стилизации компонентов интерфейса.

React – javascript-библиотека с открытым исходным кодом для создания и компоновки пользовательских интерфейсов.

Node.js – серверный язык программирования, основанный на javascript

Псевдослучайная генерация – метод получения определенных данных на основе ключевых параметров.

Simplex Noise (Симплексный шум) [1] – один из способов построения n-мерной функции шума для получения плавных псевдослучайных искажений.

Quad Tree (Дерево квадрантов) – связный ациклический граф, в котором у каждого внутреннего узла ровно 4 потомка [2]. В данной работе используется как алгоритм уточнения местоположения камеры пользователя над участком ландшафта для детализации.

Шейдер – компьютерная программа, исполняемая процессорами видеокарты.

Рендеринг [3] – процесс преобразования трехмерных моделей и структур в двухмерное изображение на компьютере. В данной работе используется рендеринг в реальном времени, то есть выполняется преобразование в

двухмерное изображение несколько раз в секунд, таким образом появляется возможность управления и действия над целью рендеринга.

WebGL – кроссплатформенный API для реализации 3D-графики в браузере.

Three.js – кроссбраузерная библиотека, используемая для визуализации трехмерных моделей. Взаимодействует с графической картой компьютера на низком уровне с помощью WebGL и шейдеров.

Биом – экосистема одной природно-климатической зоны, включающая в себя типы растительности, места обитания животных, а также особенности образования высот местности.

Климатическая карта – структура данных, является одним из способов визуализации и приданию характеристик местности на основе параметров высоты, влажности и температуры каждого участка ландшафта. Климатическая карта представляет из себя совокупность биомов для будущего ландшафта.

Диаграмма Вороного – структура данных, представляющая конечное множество точек на плоскости, где разбиение точек происходит по принципу близости к одному из элементов множества точек [4]. В данной работе используется для построения климатических карт на основе ациклического связного графа биомов.

Ландшафт – вид земной поверхности, представляющей совокупность параметров высотности, климатических условий, и определенной местности

REST – архитектурный стиль взаимодействия компонентов распределенного приложения.

HTTP – протокол прикладного уровня обмена данных между программными средствами, структурирует и описывает шаблоны запросов взаимодействия.

ВВЕДЕНИЕ

В современном мире все чаще используются технологии с применением или основанные на трехмерном моделировании тех или иных процессов. Также на данный момент активно развивается игровая индустрия. Стоит учитывать, что визуальные эффекты также стали неотъемлемой частью многих фильмов или сериалов в киноиндустрии.

С массовой потребностью в визуализации трехмерной графики стало появляться огромное количество инструментов для моделирования. Таким образом были созданы различные программные средства, в которых пользователь сможет сам воссоздать необходимые модели. Многие научные или игровые проекты требуют моделирования ландшафтов и к сожалению ручное воссоздание местности не всегда является возможным, так как требует огромного количества времени прямо-пропорционального размерам местности. Как следствие тратится большое количество времени на моделирование самой местности, а после требуется еще и разместить собственные разработанные трехмерные модели объектов на этой местности, например – деревья, дома и т.д.

Таким образом возникает необходимость в инструментарии для создания бесконечно больших процедурных псевдослучайных [5] ландшафтов с возможностью процедурного размещения готовых трехмерных моделей на создаваемой местности.

Целью работы является разработка специализированного алгоритма, реализующего псевдослучайную устойчивую генерацию ландшафта на основе вводных данных, представляющих из себя различные параметры генерации и пользовательские модели.

Задачи, которые требуется решить для достижения поставленной цели:

1. Рассмотреть существующие решения в сфере псевдослучайной генерации ландшафтов.

2. На основе существующих аналогов определить необходимые функциональные возможности будущего сервиса, а также выбрать основные вводные параметры для создания местности.
3. Необходимо выбрать средства разработки, и спроектировать архитектуру приложения и интерфейс визуализации.
4. Проработать систему взаимодействия пользователя как с визуальной частью приложения, так и с API – предоставляемым интерфейсом для сторонних программных средств.
5. Разработать алгоритм создания ландшафта с объектами на основе структуры связного ациклического графа.
6. Внедрить методы детализации массивных ландшафтов для оптимизации разработки.
7. Протестировать готовое приложение.

Объектом исследования является сервис для получения данных о процедурной генерации ландшафта, как в виде набора бинарных данных, так и в виде визуального представления работы алгоритма.

Предметом исследования является разработка специализированного сервиса, совмещающего алгоритмы визуализации, детализации и процедурной генерации местности и объектов на ней.

Практическая значимость решения заключается в предоставлении инструмента для создания псевдослучайных реалистичных ландшафтов, совмещающего в себе устойчивые алгоритмы генерации местности на основе вводимых параметров, алгоритмы оптимизации в виде пошаговой детализации данных ландшафтов, а реализующего расстановку готовых трехмерных моделей на основе размещения их на структуре связного ациклического графа.

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Описание проблемы

Проблемой работы является структуризация и объединение алгоритмов и методов для создания бесконечно больших реалистичных трехмерных ландшафтов, с учетом уже готовых трехмерных моделей, без дальнейшей работы над их расстановкой на сгенерированной местности. Возможность реализации и оптимизации таких ресурсоемких задач в условиях браузера. Обеспечение скорости работы приложения в реальном времени в виде API для сторонних программ. Автоматизация процессов создания игровых и научных ландшафтов и минимизация времени взаимодействия пользователя и приложения для получения заполненных объектами, ландшафтов.

1.2. Сравнение существующих аналогов

1.2.1. Обоснование отбора существующих решений

В данной работе исследуются вариации устойчивого создания псевдослучайных ландшафтов на основе вводных данных. Таким образом поиск аналогов сводится к группе программ по трехмерному моделированию, способных воссоздать некоторые случайные участки ландшафта.

Следующим немаловажным критерием для отбора является доступность сервисов, он выражается в возможности единожды приобрести сервис или использовать его бесплатно в собственных проектах, так как арендная плата за услуги сервиса или плата за реализацию определенной трехмерной модели полностью исключает возможность создания участков в реальном времени.

В приложениях должна обязательно присутствовать загрузка трехмерных моделей для программной или ручной расстановки объектов на полученной местности. В ином случае возникает лишний шаг для экспорта модели ландшафта в сторонние программы, позволяющие загрузить пользовательские модели.

Все приложения должны соответствовать средне статистическим требованиям к устройству, на котором запускаются, для реализации ландшафтов за реальное время и установленные объемы памяти. То есть в требованиях перед покупкой и установкой приложения должны быть четко прописаны требования к характеристикам устройства.

Рассматриваемые программные средства должны иметь наглядную документацию или обучающие инструкции для работы с данными приложениями.

Аналоги должны соответствовать требованию о наличии визуального или программного интерфейса для взаимодействия соответственно с пользователем или другим программным средством, так-как все вычисления происходят в контексте трехмерного моделирования. Таким образом исключаются программы имеющие только консольный вывод вычислительных процессов.

Таким образом основными параметрами для отбора аналогов являются:

- Возможность создания трехмерных моделей ландшафтов;
- Возможность экспорта результатов в виде бинарных файлов или структурированных данных;
- Доступность в плане взаимодействия в реальном времени [6];
- Четко установленные требования по характеристикам устройства для установки и взаимодействия с программой;
- Наличие документации или обучающих инструкций по работе с программой.

1.2.2. Описание аналогов

На основе списка требований из главы 1.3.1. был составлен определенный перечень программно-вычислительных средств, предоставляющих возможности по созданию трехмерных моделей ландшафтов.

World Creator [7] - клиент-серверное приложение, создающее естественные ландшафты, генерируя их на сервере с помощью тысяч ядер. Позволяет редактировать ландшафт и добавлять туда свои детали.

Данное приложение устанавливается и имеет визуальный редактор ландшафтов. Программа разработана на основе графического модуля Unity, поэтому требует также сторонней установки данного программного средства, что значительно влияет на объем памяти, выделяемый на хранение данного приложения. Работает в реальном времени за счет вычислительных мощностей серверной стороны.

Алгоритм генерации, используемый в данной программе, позволяет точно настраивать как фильтры для составления карты высотности ландшафта, так и отдельные параметры данных фильтров. Фильтрами в данном случае называются искажения плоскости на основе шумовых функций таких как шум Перлина, симплексный шум, а также шум Вейвлет.

Страница приложения в сети Интернет имеет разделы по обучению работы с программой, с описанием характеристик устройства для развертывания, с описанием предоставляемых возможностей в зависимости от режима использования программного средства, платного, бесплатного и стандартного. Бесплатная версия предоставляет крайне небольшую функциональность.

Таким образом к плюсам данного приложения можно отнести:

- Комбинированное генерирование поверхности на основе нескольких шумовых функций и их точная настройка;
- Наличие визуального обучающего материала;
- Множество точных настроек для создания реалистичного ландшафта, эрозии поверхности, образование скал и гор, озер и рек;
- Выполнение сложно-вычислительных задач на стороне сервера;
- Наличие требований для устройств
- Ручное корректирование созданной карты

Минусами данного аналога является

- Бесплатная версия предоставляет самую базовую функциональность по созданию и выгрузке карт высот. Ограничена размерами и детализацией карты, максимальное разрешение 4096x4096
- Отсутствие возможности загрузки собственных трехмерных моделей для дальнейшего размещения
- Установка дополнительного программного обеспечения для взаимодействия с приложением
- Отсутствие программного интерфейса (API), при наличии серверной части приложения

Таким образом, данное приложение позволяет создавать реалистичные трехмерные модели местности. Предоставляет точные настройки ландшафтов в платной версии, а также имеет собственный онлайн-сайт, на котором можно изучить программу. При этом довольно слабо реализована бесплатная часть приложения, и даже в платной версии большая часть разработки местности перекладывается на пользователя, а отсутствие API не позволит использовать данное программное средство в собственных разработках.

Instant Terra [8] – устанавливаемое оконное приложение, позволяющее генерировать небольшие участки ландшафта, на основе различных алгоритмов создания карт высот.

Данное приложение предоставляет гораздо меньше функциональности чем предыдущее, но при этом не требует установки дополнительного ПО, а также особенностью является большой выбор для экспорта созданной модели в различных форматах. Имеет только два варианта платных версий.

Также имеет небольшой онлайн-сервис с описанием функциональности предоставляемой в зависимости от тарифа. На сайте находится небольшая документация для работы с приложением.

Плюсами данного приложения являются:

- Не требует дополнительного ПО

- Множество вариантов экспорта
- Основа вводных параметров в виде связного графа
- Минимизация шагов для получения результата

Минусами данного приложения являются:

- Поставляется только в платной версии, нет возможности попробовать приложение перед покупкой
- Отсутствие возможности загрузки собственных моделей
- Неустойчивое образование ландшафтов, одни и те же параметры могут привести к различным результатам
- Ограничено по размерам создаваемых карт

Данное приложение имеет довольно слабую функциональность, поставляется только в платной версии, но при этом не требует лишнего дополнительного ПО и позволяет получить простой реалистичный ландшафт для дальнейшего редактирования и наполнения объектами в других программах, так как есть большой выбор для экспорта модели ландшафта

World Machine [9]– полнофункциональное средство для моделирования мира с возможностью загрузки и расстановки собственных моделей, текстурирования и выгрузки готового ландшафта. Помимо всех возможностей предоставляет вариант создания ландшафта на основе шаблона определенного биома.

Также имеет бесплатную и платную версию, но в отличии от предыдущих аналогов, единственным ограничением платной версии является размер создаваемой карты.

Сайт приложения в сети Интернет имеет полное описание функциональности приложения и будущих нововведений.

К плюсам рассматриваемой программы можно отнести:

- Отсутствие дополнительного ПО
- Наличие шаблонов, на основе которых можно строить ландшафты
- Устойчивость к вводным параметрам

- Возможность загрузки собственных моделей
- Наличие API для взаимодействия с другими программными средствами

Минусами можно назвать следующее:

- Долгое вычисление местности, что создают определенные трудности в использовании приложения как API в реальном времени
- Ограничение по размерности создаваемой карты
- Отсутствие документации или обучающих материалов

Данное приложение в полной мере дает попробовать функциональность алгоритмов создания ландшафта как в среде визуального интерфейса, так и с помощью сырых данных в собственных приложениях.

3D Map Generator [10] – является плагином для популярного программного средства Photoshop, соответственно устанавливается вместе с ним. Работает на самом просто алгоритме генерации карты высот – шуме Перлина. Стоимость плагина полностью определяется стоимостью Photoshop. Имеет самую базовую функциональность для создания карт, но при этом является лишь дополнением к приложению, поэтому требует минимальное количество памяти для установки и использования. Позволяет выгрузить готовый ландшафт для дальнейшего редактирования.

Плюсами данного аналога являются:

- Небольшие требования для установки
- Быстрота работы, данный плагин мог бы работать в реальном времени для получения участков ландшафта
- Имеет базовое текстурирование на основе карт высот
- Бесплатное использование при наличии Photoshop
- Устойчивая генерации на основе ключевого значения для шума Перлина

Минусами плагина являются следующие пункты:

- Отсутствие любого вмешательства пользователя по конфигурации ландшафта
- Отсутствие импорта собственных моделей
- Ограничение детализации ландшафта
- Отсутствие API, при условии быстрого создания местности, но в контексте плагина для основной программы Photoshop

1.2.3. Критерии сравнения аналогов

При рассмотрении плюсов и минусов всех представленных аналогов, были выдвинуты следующие критерии сравнения аналогов:

Устойчивость – возможность алгоритма или комплекса алгоритмов, которые представлены в программе создавать те же данные, что и ранее на основе тех же параметров [11]. Выражается в наличии данного свойства приложения.

Загрузка объектов – критерий говорит о том, что приложение не ограничивает пользователя в загрузке и конфигурации пользовательских трехмерных моделей на сгенерированном ландшафте. Выражается в трех вариантах: полностью не поддерживаются, ручная конфигурация, автоматическая конфигурация.

Размер – выражает количественную характеристику максимально возможного создаваемой детализации и размеров ландшафта, выражается в разрешении максимально возможного изображения карты высот, полученного с помощью данного приложения, обозначается в пикселях (px).

Дополнительное ПО – критерий показывает, нужно ли приложению устанавливать дополнительное стороннее программное обеспечение или библиотеки.

API – присутствует ли в приложении возможность взаимодействия в реальном времени для поэтапной процедурной генерации ландшафта, выражается в трех вариантах, так как приложение может предоставлять

программный интерфейс, но при этом, не имеет возможности процедурной генерации: в реальном времени, длительные вычисления, отсутствует.

1.2.4. Сводная информация результатов сравнения

Описание критериев в контексте каждого аналога:

1) World Creator

1.1) Устойчивость

Данное приложение не смотря на совокупность нескольких алгоритмов создания карт высот, полностью строится на основе вводных параметров, поэтому обладает устойчивостью в генерации местности.

1.2) Загрузка объектов

Программа полностью позиционируется только для создания местности, поэтому факт отсутствия загрузки и размещения собственных моделей является неудивительным.

1.3) Размер

Размер полностью сгенерированной карты, получаемый с помощью этого приложения действительно внушающий, составляет $268435456 * 268435456\,px$. Достигается путем многоядерного вычисления на стороне сервера, но создание такого ландшафта требует огромного количества времени, а также мощностей со стороны клиента для выгрузки и обработки такого масштаба.

1.4) Дополнительное ПО

Для визуализации потока данных с сервера требует дополнительной установки Unity – средства для отображения трехмерных моделей.

1.5) API

Приложение способное работать в реальном времени все же не предоставляет программный интерфейс сторонним разработчикам, он не доступен как в открытом доступе, так и для приобретения, но утверждать, что API у такого клиент-серверного приложения нет – нельзя.

2) Instant Terra

2.1) Устойчивость

Приложение генерирует небольшие участки ландшафта основываясь на вводимых параметрах, смешивая их с собственными, поэтому получение устойчивой карты не представляется возможным.

2.2) Загрузка объектов

Приложение предоставляет внутренние шаблоны моделей для конфигурации местности, но загрузить пользовательские объекты нельзя

2.3) Размер

Приложение является небольшим, но позволяет создавать карты высот размером в $64000 * 64000\ px$.

2.4) Дополнительное ПО

Так как само приложение поставляется только в платной форме, оно не требует дополнительного программного обеспечения, и реализует технологии визуализации и создания ландшафтов само.

2.5) API

Не предоставляет API, но присутствует возможность выгрузки данных в виде бинарных файлов трехмерных моделей,

3) World Machine

3.1) Устойчивость

Карта высот строится на основе различных устойчивых алгоритмов генерации. Приложение является полностью устойчивым к вводимым параметрам.

3.2) Загрузка объектов

Данное ПО предоставляет возможность полностью конфигурировать местность с объектами. Пользовательские модели можно загружать, но размещать их придется вручную.

3.3) Размер

Приложение предоставляет самую обширную функциональность из всех представленных аналогов, но выходная карта ограничена $12000 * 12000\ px$

3.4) Дополнительное ПО

Приложение работает автономно как в бесплатной версии, так и в платной, дополнительно ПО не требуется.

3.5) API

Единственное приложение, которое предоставляет программный интерфейс, но, к сожалению, в реальном времени достраивать участки ландшафта не представляется возможным.

4) 3D Map Generator

4.1) Устойчивость

Основным алгоритмом, на основе которого строится карта высот, является шум Перлина, особенностью данного алгоритма является устойчивость при одинаковом ключевом значении.

4.2) Загрузка объектов

Так как данная разработка является лишь плагином для основного приложения Photoshop, загрузка моделей не реализована.

4.3) Размер

Минимальное значение размеров из всех аналогичных сервисов, равное $1750 * 1100\,px$. С помощью этого достигается почти моментальное построение ландшафта, а также малое потребление памяти приложением.

4.4) Дополнительное ПО

Данный плагин является лишь дополнением, поэтому для работы с ним требуется основное платное приложение Photoshop

4.5) API

С учетом особенности реализации, данная разработка могла бы предоставить самое быстрое взаимодействие с API и процедурной генерации карт, но из-за отсутствия прослойки API у Photoshop на данный момент это не представляется возможным.

Сводная информация по результатам сравнения каждого аналога представлена в табл. 1. Обозначения, используемые в таблице для каждого критерия:

Устойчивость – Да / Нет

Загрузка объектов – НП (не поддерживается) / РК (ручная конфигурация) / АК (автоматическая конфигурация)

Размер - $N * M\ px$

Дополнительное ПО – Нет / Да

API – РВ (в реальном времени) / ДО (длительные операции) / Нет (отсутствует)

Таблица 1 – Сводная таблица по результатам сравнения аналогов

Критерий / **Сервис**	**Устойчивость**	**Загрузка объектов**	**Размер**	**Доп. ПО**	**API**
World Creator	Да	НП	$268\,435\,456 * 268\,435\,456\ px$	Да	Нет
Instant Terra	Нет	НП	$64\,000 * 64\,000\ px$	Нет	Нет
World Machine	Да	РК	$12\,000 * 12\,000\ px$	Нет	ДО
3D Map Generator	Да	НП	$1\,750 * 1\,100\ px$	Да	Нет

Данная таблица четко показывает достоинства и недостатки аналогичных сервисов, описанных ранее.

1.3. Выводы

В результате сравнения аналогичных сервисов можно сделать следующие выводы. Представленная в табл. 1 информация показывает, что основная тенденция в приложениях для создания процедурных псевдослучайных ландшафтов является устойчивость получаемых данных по отношению к вводным параметрам.

Работа с пользовательскими объектами развита наименее всего. Только одно приложение предоставляет возможность ручной конфигурации трехмерных моделей на создаваемом ландшафте. Большая часть позволяет только выгрузить трехмерную модель для дальнейшей работы с ней в средствах визуализации трехмерной графики. Автоматизированное размещение объектов не было представлено ни в одном из рассматриваемых сервисов, а это как уже было сказано в пункте 1.2 занимает большее количество времени у разработчиков сторонних приложений или программных комплексов.

Сравнение по максимальному размеру показало, что все приложения предоставляют лишь конечное разрешение карт местности. Так как уклон в основном делается на максимально точную конфигурацию самой карты высот, а не на оптимизацию и детализацию итогового ландшафта.

Дополнительное ПО является большим ограничением по работе с приложением, так как требует долгого изучения самих дополнительных программ. Также является ограничением для создания в будущем программного интерфейса, так как его поддержка должна происходить на всех уровнях взаимодействия с дополнительным ПО.

Приложения в основном предлагают готовые результаты работы для выгрузки в виде бинарных файлов, но в нынешнем мире создание процедурных ландшафтов стало довольно востребованным механизмом, и на данный момент разработчикам прикладных проектов приходится обходится собственными решениями по процедурной генерации ландшафтов. Наличие базиса в виде API позволило бы ускорить процесс разработки таких проектов.

Соответственно будущее приложение должно обладать следующими характеристиками:

- Построение ландшафтов на основе устойчивых алгоритмов псевдослучайной генерации данных.

- Предоставление возможности загрузки и автоматической конфигурации пользовательских объектов на сгенерированной местности.
- Создание условий для генерации массивных ландшафтов, ограниченных в размерах только устройством, в котором запускается приложение.
- Работа с сервисом без установки стороннего дополнительного программного обеспечения
- Наличие программного интерфейса для получения в реальном времени потока данных о создаваемой модели трехмерного ландшафта.

ГЛАВА 2. ФОРМУЛИРОВКА ТРЕБОВАНИЙ К РЕШЕНИЮ

2.1. Постановка задачи, выделение подзадач

Основной задачей данной работы является исследование алгоритма устойчивой процедурной генерации трехмерных ландшафтов с учетом вводных параметров и трехмерных моделей и разработка сервиса, предоставляющего программный и оконный интерфейсы взаимодействия с приложением. Из сравнения аналогичных сервисов были сделаны выводы о наличии основных критериев для будущего приложения. На основе этих критериев были выдвинуты следующие подзадачи:

1) Выбор и интеграция алгоритма создания карт высот на основе ключевых параметров.

В рамках данной подзадачи будет достигнута устойчивость создания ландшафтов на основе некоторых ключевых значений, вводимых пользователем. Предполагается, что в пользовательском интерфейсе будут присутствовать некоторые поля ввода на основе которых будут строится шумовые функции карты высот и влажности местности.

2) Реализация механизма загрузки, обработки и рендеринга внешних трехмерных моделей.

Данная задача предполагает разработку участка пользовательского интерфейса, который позволит выбрать бинарный файл трехмерной модели, загрузить его, обработать и отобразить в приложении. Данный интерфейс должен позволять редактировать размеры моделей, а также автоматическую корректировку позиционирования модели в пространственных координатах.

3) Разработка оконного интерфейса для визуального представления приложения.

Подзадача предполагает выбор средств для компоновки всех участков пользовательского интерфейса, реализацию и унификацию всех его компонентов. Интерфейс должен содержать в себе окна редактирования пользовательских объектов, редактирования биомов, а также два основных

окна: конфигурация древовидного графа и выбор всех параметров для будущего ландшафта, окно рендеринга основной полученной местности на основе заданных параметров и сконфигурированного графа.

4) Разработка программного интерфейса, для предоставления данных сторонним программам.

Приложение должно быть разработано на основе клиент-серверной архитектуры. Серверная составляющая должна предоставлять программный интерфейс в виде публичных методов, получаемых с помощью HTTP запросов в соответствии с REST архитектурой построения приложения.

5) Разработка и внедрение алгоритма детализации ландшафта для оптимизации рендеринга.

Подзадача предполагает решение проблемы оптимизации массивных ландшафтов. Требуется разработать алгоритм детализации на основе доступных структур хранения данных и алгоритмов. Требуется внедрить данный алгоритм для визуальной части реализации приложения, а также добавить в программный интерфейс методы, позволяющие частично детализировать участки ландшафта.

6) Разработка алгоритма создания климатической карты на основе ациклического связного графа.

Требуется разработать алгоритм преобразования структуры древовидного графа в вид климатической климатической карты. В графе должны присутствовать вершины биомов, которые также конфигурирует пользователь, и в реальном времени видит образование карты в окне приложения. Требуется выбрать структуру данных, способную по нескольким ключевым точкам получить карту температуры и влажности определенного участка местности.

7) Разработка алгоритма псевдослучайной расстановки пользовательских моделей на созданной местности.

Требуется разработать оптимальный алгоритм образования массивов объектов на сгенерированной карте высот, основываясь на параметрах

местности, к которой прикреплен конкретный объект. Данные параметры пользователь указывает при создании вершины биомов и дополняя данные вершины связью с родительскими биомами или же корневой вершиной.

2.2. Обоснование методов решения подзадач

Каждая подзадача предоставляет выбор инструментов ее решения. Такими решениями могут служить готовые библиотеки исходного кода, алгоритмы и структуры данных, которые позволяют приблизиться к решению определенной подзадачи. Соответственно для каждой подзадачи можно рассмотреть общие положительные или отрицательные стороны того или иного подхода, и выбрать оптимальный вариант в контексте разрабатываемого приложения.

1) Выбор и интеграция алгоритма создания карт высот на основе ключевых параметров.

Для начала требуется исключить алгоритмы генерации неравномерных сеток местности, так как в данном приложении конфигурируются в будущем автоматически объекты на этих сетках, следовательно, вычислительные задачи усложняются расчетом высоты поверхности в конкретной точке пространства с помощью расчета средних значений ключевых точек, окружающих объект. Основой таких алгоритмов являются полигоны – выпуклые многоугольники, лежащие на одной плоскости в пространстве. Один из вариантов полигональной карты построения ландшафта представлен на рис. 1. Вариант результата построения равномерной карты местности представлен на рис. 2.

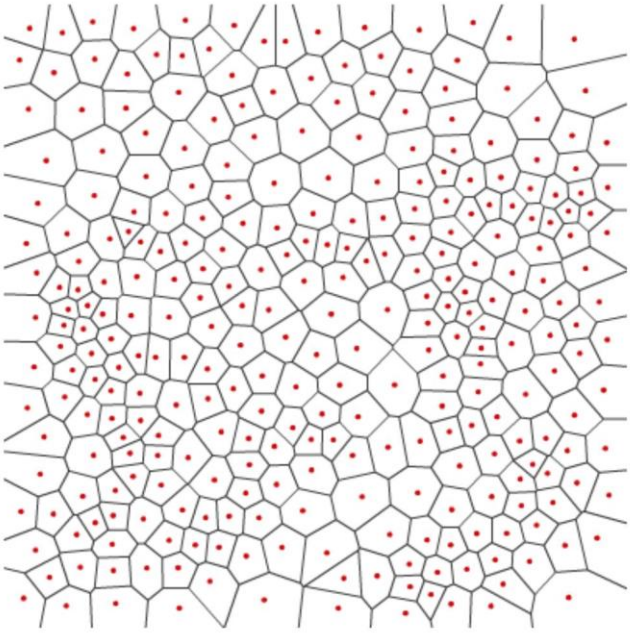

Рисунок 1 – Карта высот, построенная на основе полигонов

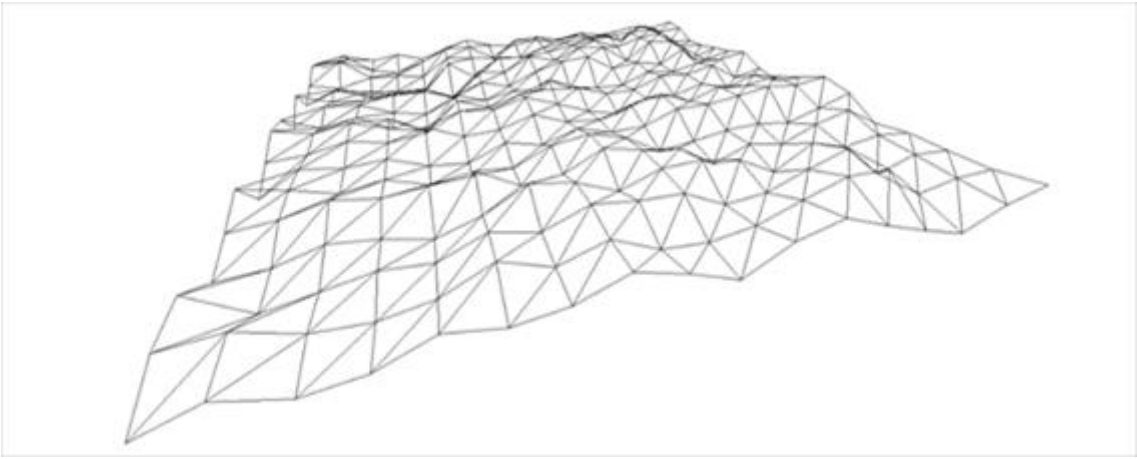

Рисунок 2 – Равномерная карта высот, построенная на основе пар треугольников

Следующим шагом требуется исключить алгоритмы создания ландшафтов, которые строятся на основе содержащихся в них параметров, то есть такие алгоритмы, которые не способны обеспечить абсолютно эквивалентные выходные данные, основанные на ключевых параметров.

Из вышесказанного следует, что требуется рассмотреть алгоритмы, способные отдавать устойчивые равномерные сетки на выходе на основе ключевых параметров. Именно такой категорией алгоритмов являются методы градиентной шумовой генерации значений [12]. Можно рассмотреть две

вариации алгоритмов на основе градиентного шума: шум Перлина, симплексный шум:

Шум Перлина был разработан как один из первых алгоритмов градиентной шумовой генерации, поэтому имеет ряд недостатков, таких как множественные артефакты в виде изотропности выбора направлений для интерполяции между выбранными псевдослучайным способом векторами. Также данный алгоритм ограничен трехмерным пространством, что исключает возможность полнофункционального использования шумовых методов.

Симплексный шум решает проблему образования артефактов в классическом варианте шума Перлина, визуально полностью изотропен, но в срезе n-мерного пространства ситуация ухудшается с каждым дополнительным измерением. Также преимуществами такого метода являются расширения на четырехмерное и пятимерное пространство, с гораздо меньшей вычислительной сложность по отношению к шуму Перлина: $O(n^2)$ для $n * n$ сетки вместо $O(n2^n)$ классического алгоритма. Симплексный шум является более простым для аппаратной реализации. В то время как шум Перлина интерполируется между градиентами конечных точек пространства, симплексные шум делит пространство на n-мерные треугольник – симплексы, что позволяет вычислять градиенты процедурно, а не для каждого участка отдельно. Симплексы строятся на основе параметров той или иной реализации этого метода.

Таким образом оптимальным вариантом для реализации равномерной сетки высот является метод симплексного шума. Данный алгоритм реализуется в 4 основных этапа: отклонение координат, разделение на симплексы, выбор градиента, суммирование ядер.

Основные математические вычисления данной модели:

Отклонение координат:

Входные точки преобразуются по формуле:

$$x' = x + (x + y + \cdots) * F$$

$$y' = y + (x + y + \cdots) * F$$

...

Где

$$F = \frac{\sqrt{n+1} - 1}{n}$$

Данное преобразование приводит к перемещению координаты на решетке, которое является расположением вершин из гиперкубической соты, сжатой по главной диагонали.

Каждая из n + 1 вершины симплекса учитывается путем суммирования радиально симметричных ядер, центрированных вокруг каждой вершины.

Симплексный шум строится на основе ключевого значения, а преобразование выходных значений с помощью параметров чистоты, степеней наложения нескольких значений шума, а также возведение в степень результата дает визуально разные результаты.

2) Реализация механизма загрузки, обработки и рендеринга внешних трехмерных моделей.

Для реализации трехмерной графики в браузерах существуют различные варианты. От более низкоуровневых до высокоуровневых библиотек. Например, все сторонние библиотеки строятся на основе WebGL. Данный вариант можно было бы использовать, если бы требовались в основном низкоуровневые операции с графической картой компьютера.

Следующим вариантом для рассмотрения стала библиотека p5.js, но за счет своей простоты и высокоуровневого программного интерфейса она полностью исключает возможность взаимодействия с графической картой.

Самым близким вариантом к реализации данного приложения стала библиотека Three.js, которая предоставляет удобные механизмы для работы с трехмерными моделями как на базовом уровне в виде буфферных структур

данных с атрибутами для шейдерной пост-обработки, так и с оптимизированными изменяемыми готовыми фигурами [13].

В данной библиотеки присутствует механизм для обработки полученного бинарного файла с расширением .obj трехмерной модели.

После обработки Three.js модель представляет собой сущность объекта фигуры. Такую модель можно обернуть в контейнер, представляющего собой прямоугольный параллелепипед, и в данном варианте найти минимальную нижнюю точку фигуры и ее центр, и сместить соответственно данную фигуры в центр пространственных координат {0, 0, 0}.

3) Разработка оконного интерфейса для визуального представления приложения.

Представление оконного интерфейса в браузере может выполняться с помощью совершенно различных фреймворках, а также на базовом языке разметки HTML, каскадных стилях CSS, и установленного за стандарт языка программирования Javascript. Но такой подход наполняется сложностями компонентной реализации приложения, то есть где каждый компонент можно будет повторно использовать изменяя лишь его опции.

Для конкретной задачи подходит большая часть фреймворков для разработки на основе компонентного подхода. Из всех вариантов был выбран фреймворк React. Его особенностью является реактивное изменение DOM-дерева страницы, при изменении состояния компонента [14]. Также React полностью инкапсулирует контекст выполнения Three.js, что позволяет разрабатывать отдельные компоненты с визуализацией трехмерной графики. Таким образом появляется возможность реализовать окна редактирования пользовательских моделей, а также биомов. А также разделение и унификация компонентов позволяет динамически изменять структуру графа с сущностью вершин, которые инкапсулируют в себе собственные конфигурации.

4) Разработка программного интерфейса, для предоставления данных сторонним программам.

Существует огромное количество языков программирования, способных обеспечить REST-архитектуру взаимодействия с программой. Но в данной работе от сервера требуется преобразование трехмерных моделей и генерация данных. Таким образом было бы удобно использовать язык программирования который предоставляет библиотеку Three.js, которая также используется и на клиентской стороне. Наиболее подходящим решением стал Node.js – программный инструмент, основанный на системе V8 использующейся в браузерах. Транслирует Javascript в машинный код.

Node.js предоставляет сервис для получения различных пакетов – NPM. Одним из таких пакетов и является библиотека Three.js, но для взаимодействия с серверной стороной, написанной на Node.js требуется также пакет распределения и управления запросами Express [15].

Express позволяет работать в классической методологии объектно-ориентированного программирования на стороне сервера. Предоставляет механизмы роутинга, то есть распределения запросов на отдельный сущности класса Router.

5) Разработка и внедрение алгоритма детализации ландшафта для оптимизации рендеринга.

В ходе разработки приложения большим ограничением является контекст разрабатываемого приложения в браузере для реализации как визуальной части, так и серверного API. Таким образом требуются алгоритмы оптимизации приложения. Были разработаны несколько вариантов алгоритма детализации.

Первым вариантом служило первоначальное разделение ландшафта на определенные участки, при приближении к которым они начинали перестраиваться с более высокой точностью, пример такого алгоритма представлен на рис. 3.

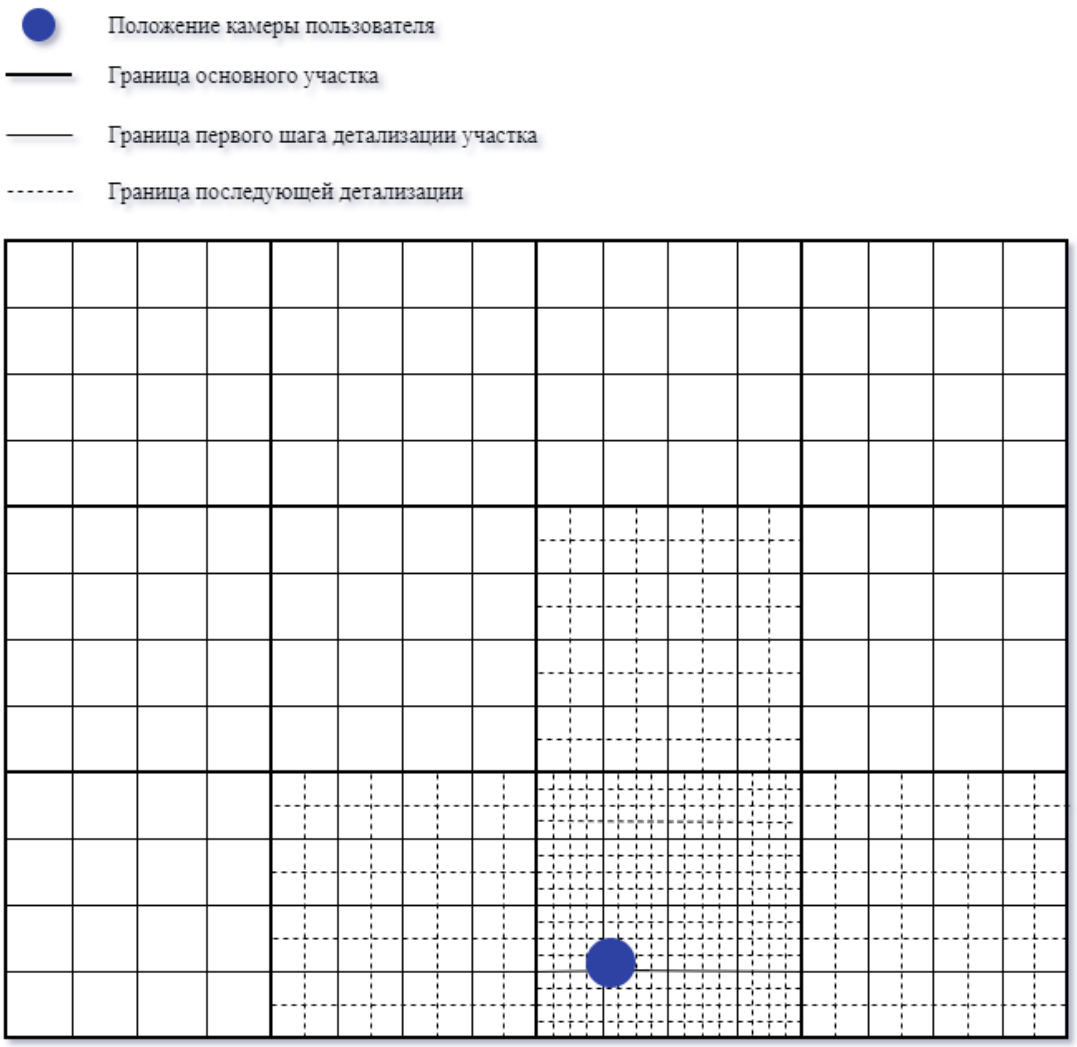

Рисунок 3 - Алгоритм детализации на основе установленного размера участка

Минусами такого подхода является перебор всех первоначальных участков и вычисление расстояния от ключевых центральных точек данных каждого участков до положения камеры пользователя. Также возникает проблема с увеличением уровня детализации, так как каждый раз нужно все больше и больше детализировать участок установленного размера, таким образом на первых уровнях происходит быстрая детализация, но с каждым новым уровнем время нарастает прямо-пропорциональной уровню.

Вторым подходом для детализации участков ландшафта стал метод построения на основе структуру Quad Tree. Данная структура представляет из

себя связные ациклический граф, в котором у каждого узда может быть либо 0, либо 4 дочерних потомка. Таким образом задача поиска детализируемого участка преобразуется в задачу вставки элемента в дерево квадрантов, которая создаст новые потомки узлов и удалит старые. Схематическая работа алгоритма представлена на рис. 4.

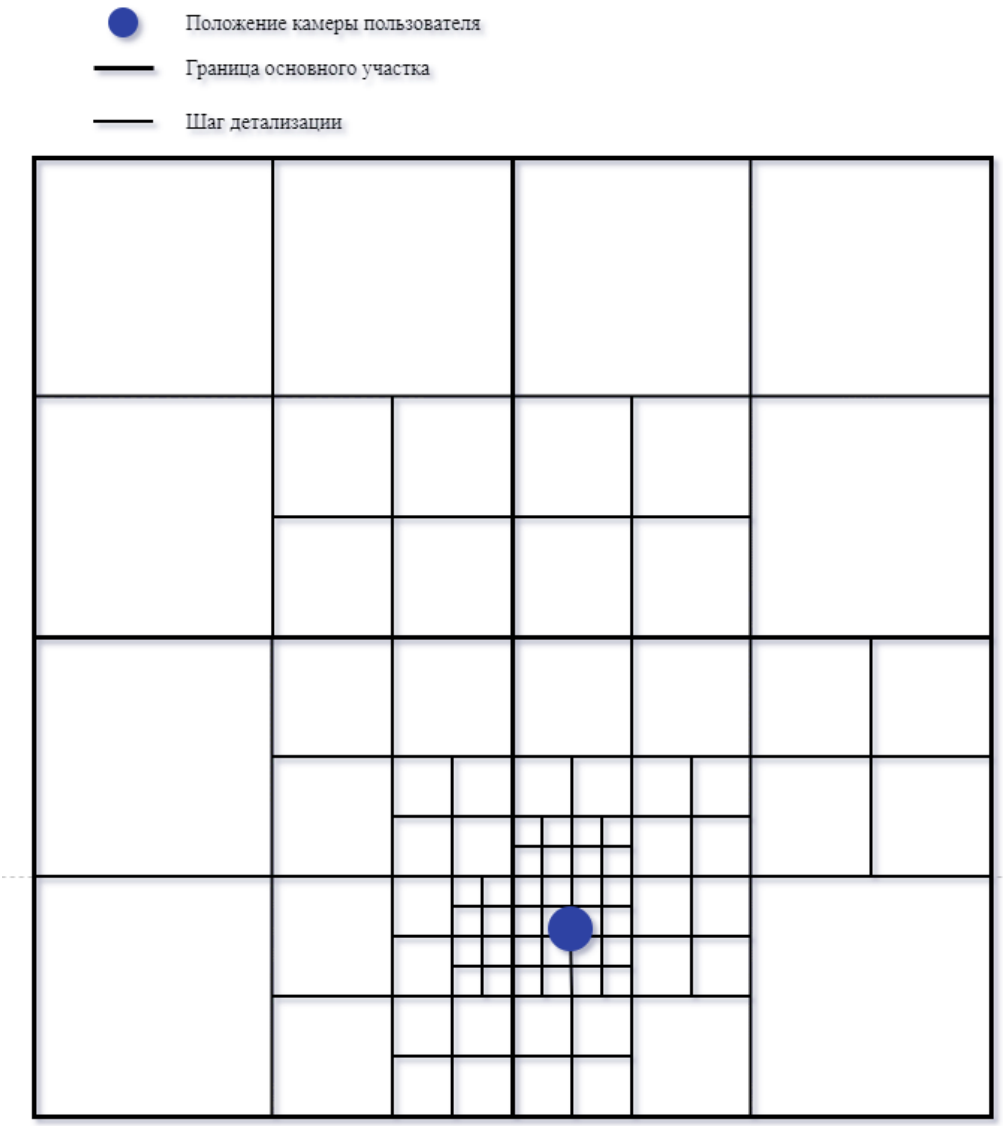

Рисунок 4 – Алгоритм на основе дерева квадрантов (Quad Tree)

Данный алгоритм ускоряет поиск нахождения участка для детализации, а также обеспечивает равное время для детализации каждого участка, так как детализация всегда остается неизменной, меняется только отступы участка, которые можно передать в алгоритм процедурной генерации ландшафта x.

6) Разработка алгоритма создания климатической карты на основе ациклического связного графа.

На данный момент автоматическая конфигурация климатических карт, на основе выбранных биомов еще нигде не используется. Был предложен свой вариант решения данной задачи.

Для каждого биома указана средняя температура воздуха и средняя влажность, выражаются в градусах и процентах соответственно. Первоначальный набор биомов представляет из себя множество точек на плоскости. Связь вершины биома с корнем конфигурируемого графа означает, что средние значения температуры и влажности для данного биома берется первоначальное, указанное в конфигурации биомов. Далее биом прикрепленный к другому биому строится относительно него. Для начала выбирается направление от родительской вершины до рассматриваемой. Вектор направления представляет собой смещение данного биома относительно родительского, величина смещения зависит от уровня вложенности вершины. Иллюстрация разработанного алгоритма представлена на рис. 5.

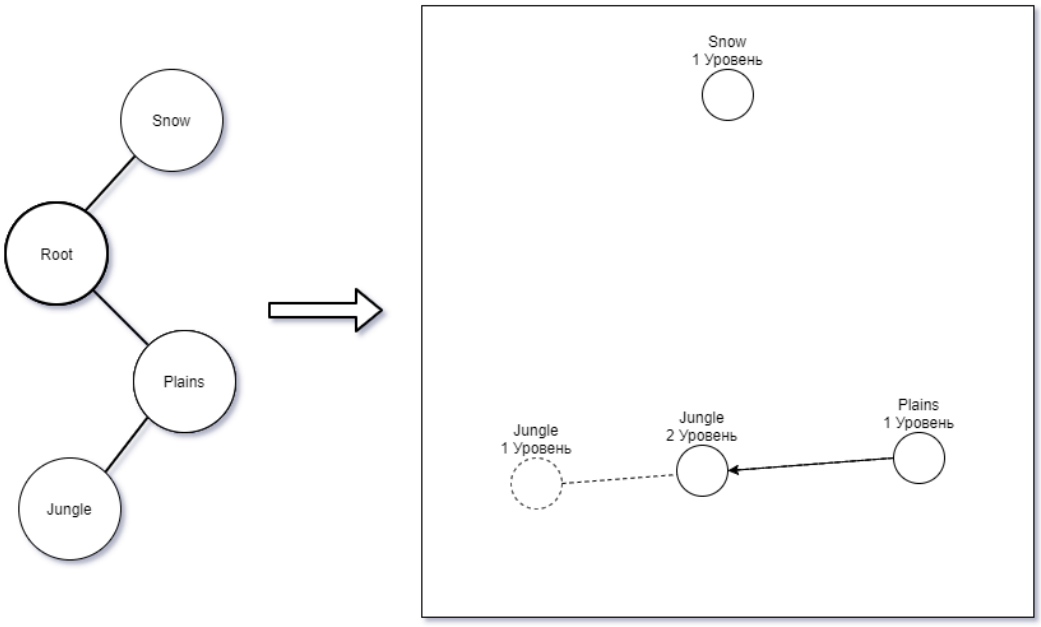

Рисунок 5 – Пример работы разработанного алгоритма

На основе полученного множества точек биомов на плоскости с помощью диаграммы Вороного определяются прилежащие множества для каждой точки исходной плоскости в зависимости от ее детализации, пример построения диаграммы Вороного на основе множества точек из примера представленного на рис. 5 иллюстрируется на рис. 6.

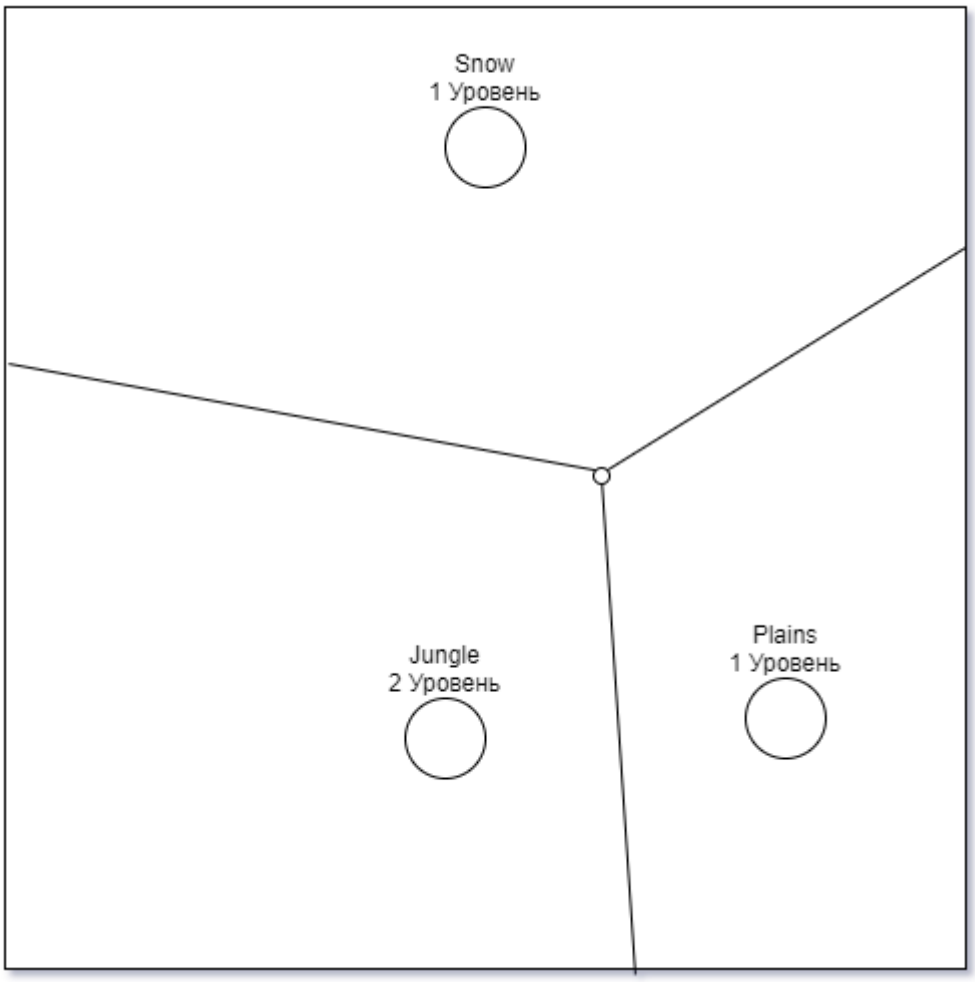

Рисунок 6 – Пример построения диаграммы Вороного на основе множества точек распределения биомов

Данный алгоритм полностью подходит для данной работы, так как позволяет строить климатические карты на основе конфигурируемых данных. Количество и связи биомов пользователь сможет выбирать самостоятельно. Обновление карты будет происходить в реальном времени на странице пользовательского интерфейса, рядом с настраиваемым графов.

7) Разработка алгоритма псевдослучайной расстановки пользовательских моделей на созданной местности.

Данный механизм автоматизированного размещения объектов на поверхности еще не был реализован в аналогичных приложения как видно из сравнения. По этой причине предлагается собственный алгоритм для реализации этой возможности.

Пользовательские объекты, также, как и биомы являются вершинами, но в отличии от вершин-мест, объекты не могут иметь связей с другими объектами или корнем. То есть сама структура графа должна предоставлять возможность создания связей только с вершинами биомов.

В вершинах биомов определяются параметры на основе которых будут распределены объекты на представленном биоме. Были выдвинуты следующие параметры:

Тип биома – при конфигурации вершины биома можно выбрать определенный тип, на примере из рис. 5 такими типами могут служить джунгли, равнины, снежные горы и т.д [16]. Выбирая тип пользователь обозначает в каком биоме на будущем ландшафте будут размещаться объекты.

Кластеризация (Clustering) – Обозначает насколько объекты будут разбиты на определенные кластеры [17] на представленном биоме. Позволяет, например, создавать сильно-распределенные массивы лесов, и группировать здания для образований поселений и городов.

Насыщенность (Saturation) – Описывает насколько плотно участок или кластер будут заполнены объектами, прикрепленными к этой зоне. Позволяет более точно конфигурировать редкость определенных моделей, встречающихся на будущем ландшафте.

Хаотичность (Chaotic) – Определяет дополнительное генерируемое смещение от первоначальной точки сгенерированной для объекта, а также определяет поворот модели на местности. Например, для создания плантаций растений, можно указать нижнюю границу этого параметра, тогда смещение будет минимальным, объекты будут образовывать определенную структуру,

повороты будут осуществляться только на 0, 90, 180, 270 градусов по оси нормали к центру ландшафта. При указании верхней границы параметра, смещение и повороты будут основываться на параметрах модели, а также параметрах окружающей местности, то есть также генерируемые смещение и поворот будут устойчивыми к вводным данным.

Далее при построении участка ландшафта для каждой модели определяется родительская вершина биома с определенными параметрами. Модель конфигурируется основываясь на этих параметрах и отправляется для рендеринга.

Полнота (Fullness) – Обозначает порог значения сгенерированной шумовой функции объекта, после которого объект может быть расположен на местности в зависимости от прочих параметров.

2.3. Выводы

В результате задача была разбита на более небольшие подзадачи. Были исследованы подходы решения аналогичных задач. Исследованы основные методы и решения, и на основе плюсов и минусов были выбраны наиболее оптимальные алгоритмы, структуры данных и архитектура приложения для выполнения набора подзадач.

ГЛАВА 3. ОПИСАНИЕ МЕТОДА РЕШЕНИЯ

3.1. Пользовательский интерфейс

3.1.1. Структура пользовательского интерфейса

Для реализации клиентской части данной работы используется web-фреймворк React, который реализует компонентный подход для построения интерфейса приложения. Следовательно, были разработаны следующие базовые компоненты для построения основных интерфейсных участков:

Кнопка-переключатель

Компонент обеспечивает последовательное переключение значений в зависимости от передаваемого списка. Две возможные вариации использования компонента представлены на рис. 7.

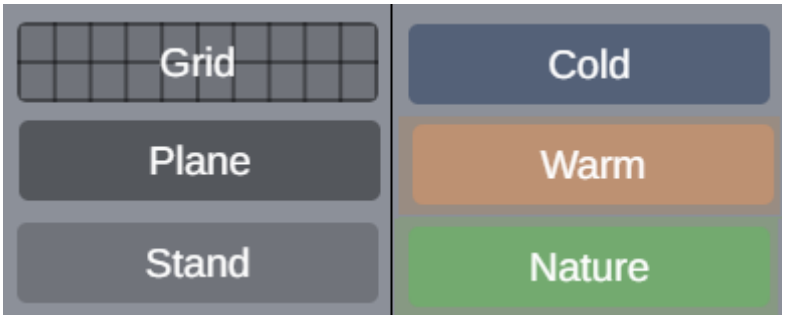

Рисунок 7 – Два варианта использования кнопки переключателя

Данный компонент принимает параметры стилизации фона кнопки, а также список возможных параметров значения, при переключении отправляет событие о том, что значение изменилось.

Диалог редактирования

Базовый компонент, которое обеспечивает открытие отдельного всплывающего окна. Реализует затемнение основного контента, и построение передаваемых компонентов.

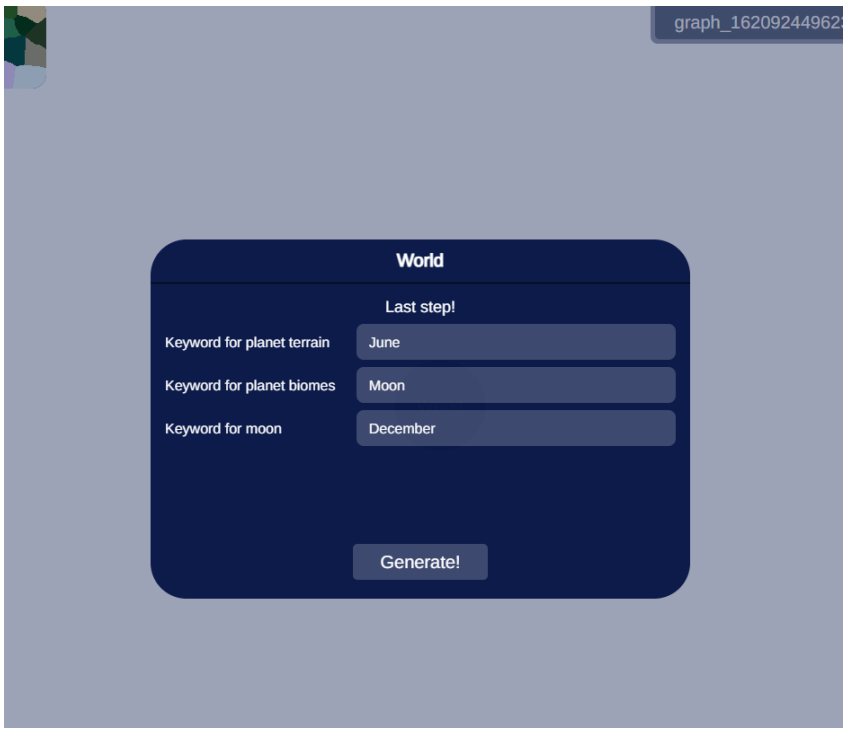

Рисунок 8 – Одно из представлений базового компонента диалога

Принимает три компонентных параметра: header, body, footer. Передаваемые компоненты размещаются в соответствии с типом опции. А также можно сконфигурировать закрытие по нажатию левой кнопки мыши по затемненной области с помощью отдельного параметра.

Текстовое поле ввода

Обеспечивает ввод текста, принимает параметры проверки вводимых значений. Предоставляет возможность сконфигурировать подпись к полю с помощью отдельной опции caption. Передает вводимое значение в родительский компонент. Три поля ввода представлены на рис. 9.

Рисунок 9 – Текстовые поля ввода

Выпадающее меню

Обеспечивает выбор из списка значений, используется, когда список значений может изменятся динамически с помощью обращений к серверу. А также предоставляет просмотр информации по каждому пункту меню. Примеры выпадающих меню представлены на рис. 10.

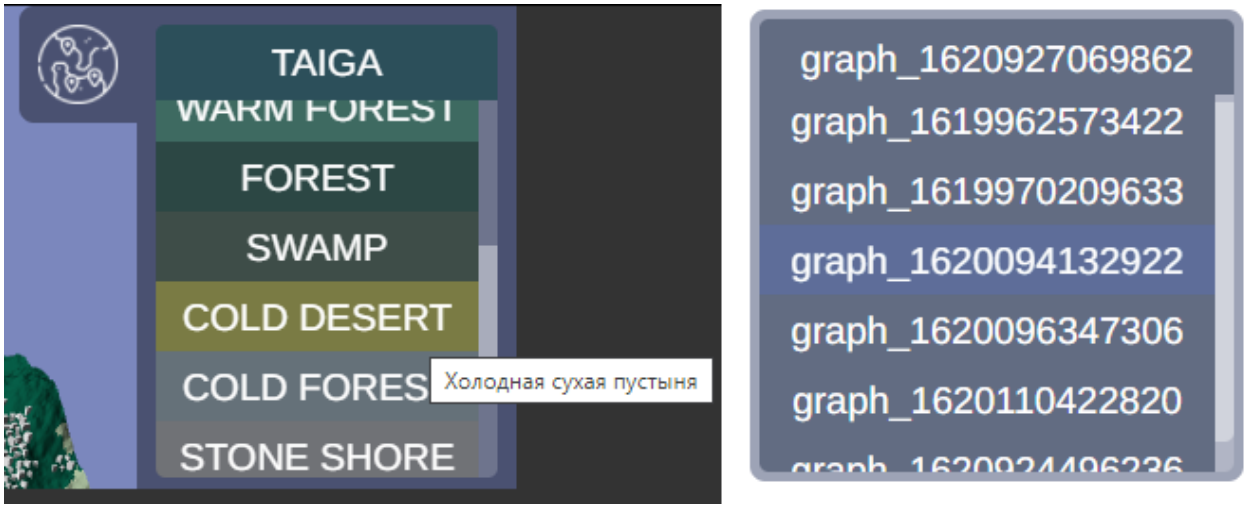

Рисунок 10 – Выпадающее меню

Вершина графа

Обеспечивает изображение редактируемой модели, также принимает характеристики стилизации рамки.

Вершина предоставляет возможность перемещения с помощью перетягивания мыши пользователем. Также отправляет события о перемещении, для информирования родительского компонента. Позволяет посылать событие о клике на вершину для дальнейшей обработки, как например открытия диалогового окна редактирования вершины. Представлено в виде окружностей на рис. 11.

Ребро графа

Компонент создается между двумя вершинами для визуализации связи между ними. Принимает положения и стилизацию вершин для собственной конфигурации цвета, а также положения между двумя точками – центрами вершин.

Может работать в двух вариантах как потенциальное ребро, или как установленное. Потенциальное ребро можно перевести в установленное с

помощью нажатия на ребро. Удалить установленное ребро и перевести его в режим потенциального можно с помощью двойного нажатия на установленное ребро. Ребра графа показаны в виде соединительных линий между вершинами на рис. 11.

Рисунок 11 – Представление вершин и ребер графа

Таким образом были разработаны базовые компоненты, которые позволяют повторно использоваться в любом месте приложения.

Информационное окно

Компонент представляет из себя одну небольшую кнопку, которая разворачивается в окно, компонент и пример разворота, см. рис. 12.

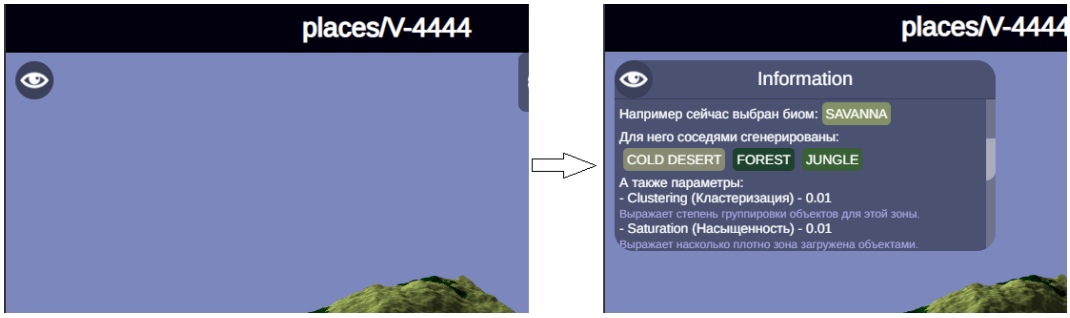

Рисунок 12 – Информационное окно

Данный компонент предоставляет возможности отображения любых компонентов который будут переданы в опцию content.

Кнопка слайдер

Компонент представляет из себя ползунок, который можно передвинуть для определения желаемого значения. С помощью опций max, min, step можно задать соответственно максимальное, минимальное значение, а также значение шага. См. рис. 13.

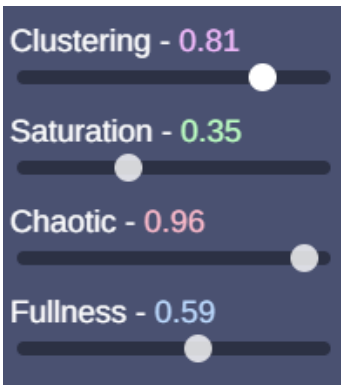

Рисунок 13 – Группа кнопок слайдеров

3.1.2. Описание взаимодействия с приложением

На основе базовых компонентов были разработаны две основные страницы пользовательского интерфейса:

1) Страница конфигурации графа пользовательских объектов.

Представляет из себя набор сложных компонентов, то есть тех, которые строятся на основе базовых. Позволяет полностью задать настройки будущего ландшафта. Страница графа см. рис. 14.

Рисунок 14 – Страница конфигурации графа

Компоненты, представленные на данной странице:

Модель климатической карты будущего ландшафта

Представляет из себя изображение, находящееся в левом верхнем углу. Данное изображение является кнопкой, и открывает диалог просмотра подробной информации о будущем климате ландшафта. Данный диалог строится на основе базового диалога редактирования, представленного в пункте 3.1.1. Окно, открываемое при нажатии на данную кнопку представлено на рис. 15.

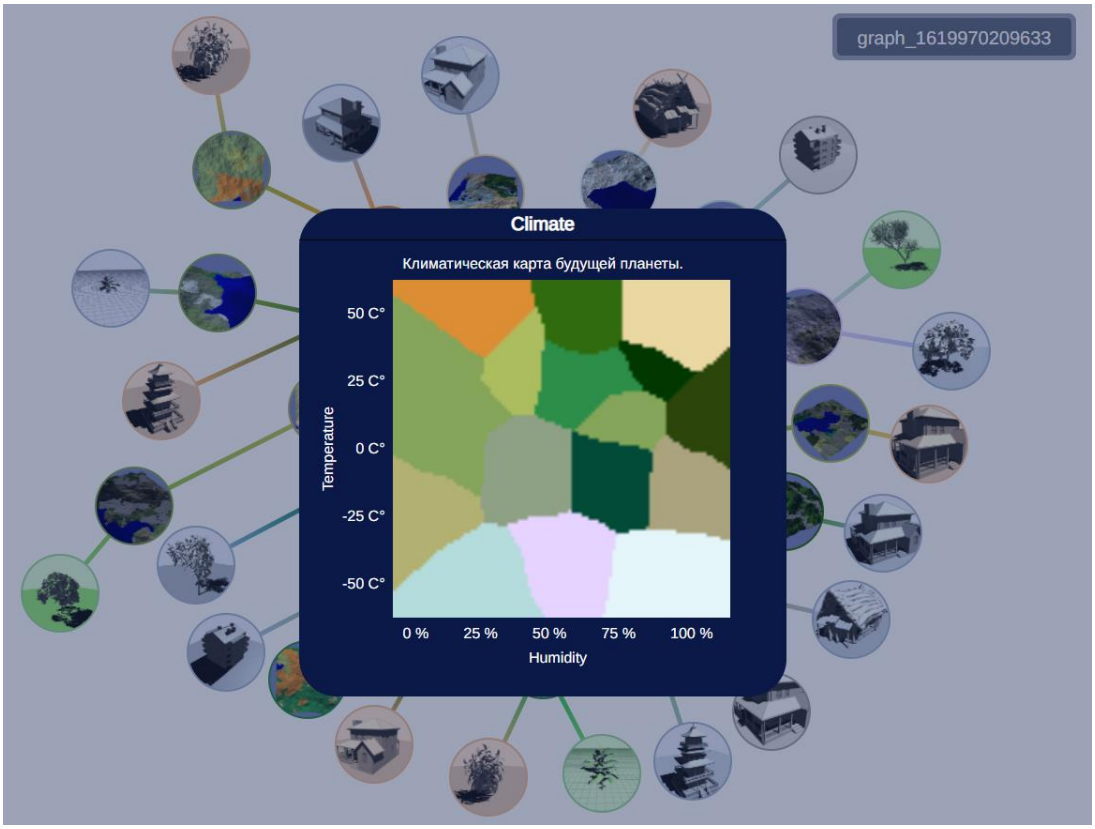

Рисунок 15 – Диалоговое окно просмотра климата ландшафта

В данном диалоге можно полностью просмотреть образование климата на основе температуры и влажности.

Изображение климатической карты формируется в реальном времени на основе алгоритма генерации климатической карты из связного ациклического графа биомов. Пример формирования такой карты представлен на рис. 16.

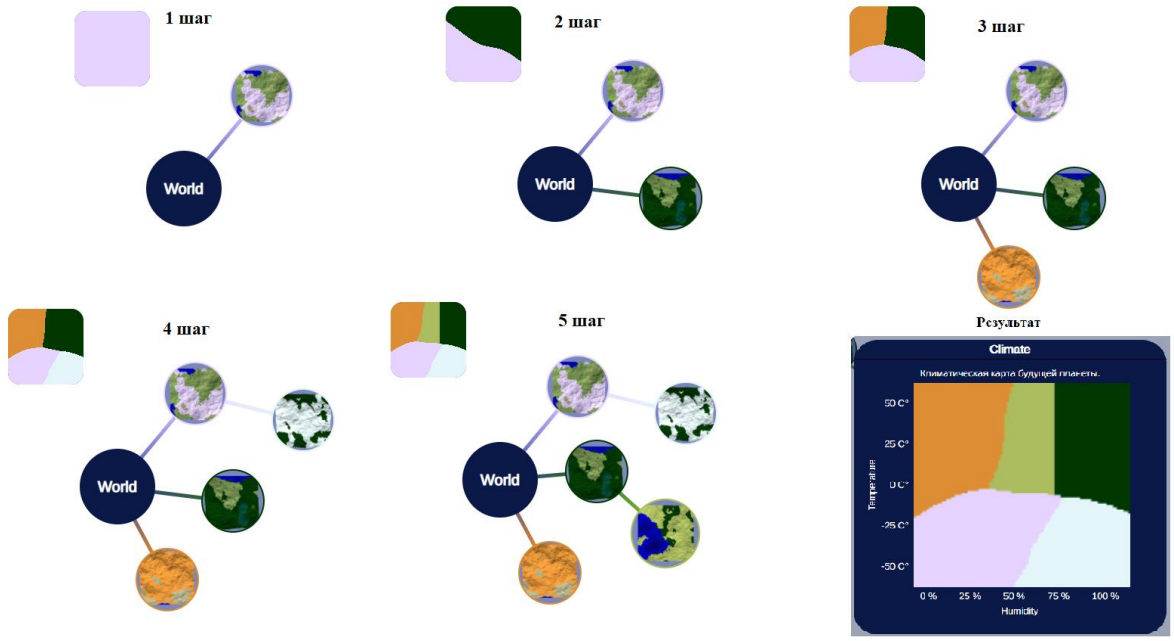

Рисунок 16 – Пример пошаговой конфигурации климатической карты

Граф пользовательских объектов

Организует возможность конфигурации структуры связного ациклического графа [18]. Содержит в себе динамический набор добавляемых вершин местности и загружаемых моделей.

Ограничивает возможности конфигурации только определенными связями между вершинами. Биомы могут быть связаны с вершиной корнем, или же с другим биомом, который уже привязан к графу. Объекты могут быть связаны только с определенным одним биомом. Данное ограничение позволяет создавать именно связные ациклические графы, где корнем является синтетическая вершина, в которой задаются ключевые значения для построения шумовых функций влажности, высоты и дополнительного ландшафта луны.

Реализация выбора потенциальных вершин представлена на рис. 17.

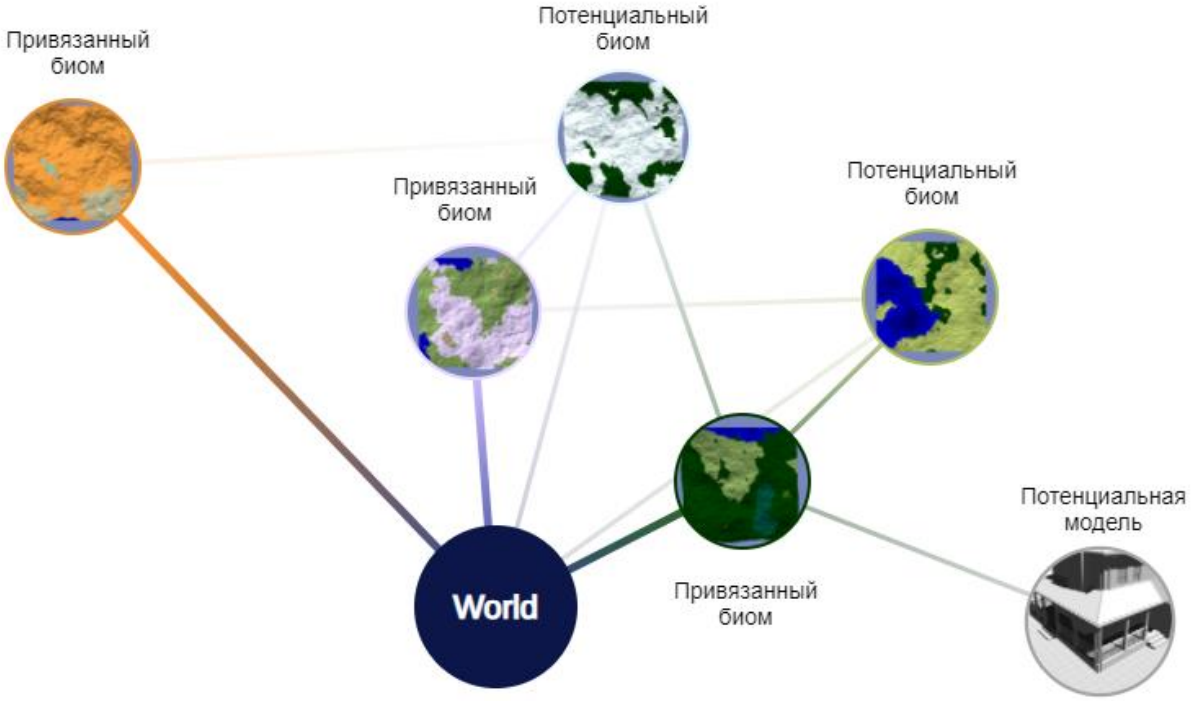

Рисунок 17 – Механизм привязки вершин в графе

Выпадающий список созданных на сервисе графов

Находится в правом верхнем углу страницы на рис. 14. Позволяет загрузить любой граф, для дальнейшего редактирования или построения на его основе ландшафта, представлен на рис. 18.

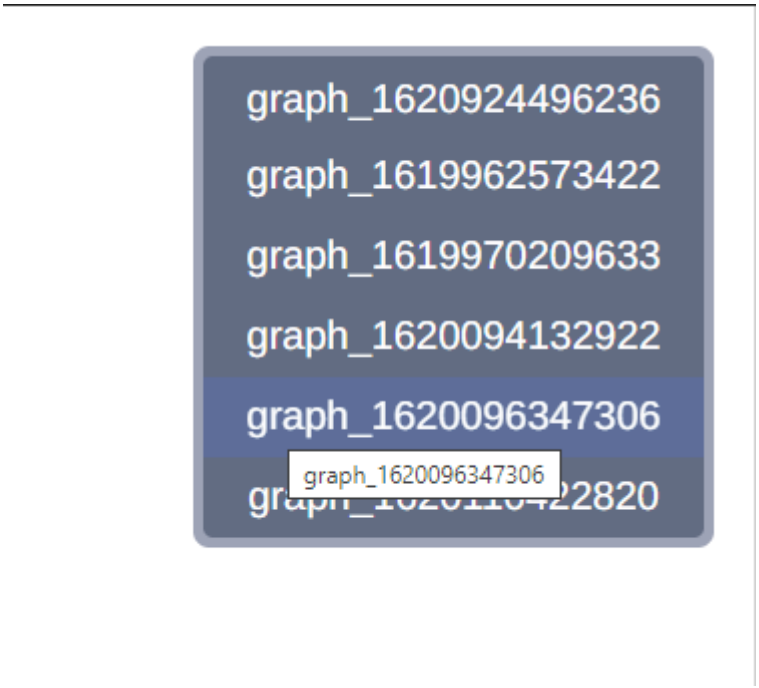

Рисунок 18 – Список созданных на сервисе графов

Контекстное меню страницы

Данное меню открывается по нажатию правой кнопки мыши в любом месте приложения. Данное меню представлено на рис. 19.

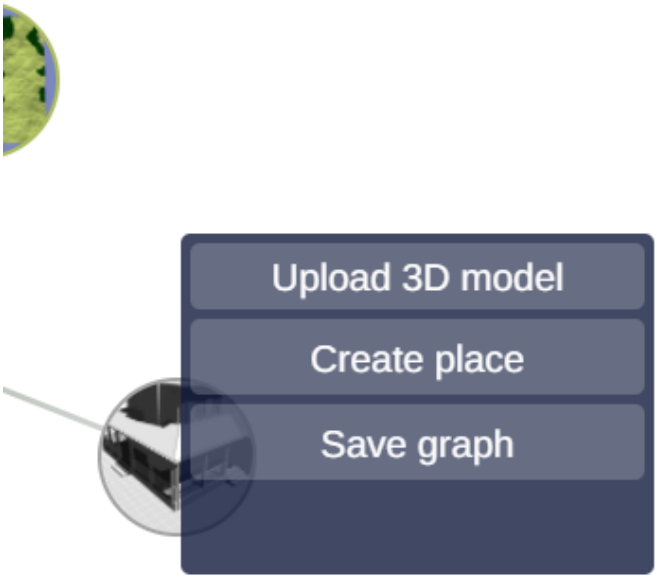

Рисунок 19 – Контекстное меню приложения

Содержит три кнопки:

Кнопка загрузки объектов – открывает окно выбора файлов соответствующее API операционной системы. Выбранный файл проверяется на наличие расширения трехмерного объекта. А также на объем загружаемого файла, который не должен превышать 300мб. После загрузки и проверки файла открывается диалог редактирования модели.

Кнопка создания места – открывает диалоговое окно конфигурации вершины с биомом.

Кнопка сохранения графа – позволяет добавить собственный созданный граф на сервере, для дальнейшей работы над ним.

Диалог редактирования объекта

Данный составной компонент выполняет предобработку загружаемой модели, а также позволяет пользователю выбрать набор стилей для

отображения данной модели в графе, а именно поверхности на которой находится модель и света, который падает на трехмерную модель. Позволяет изменить размер загруженной модели для дальнейшего позиционирования на ландшафте с определенным множителем первоначального размера модели. Диалог представлен на рис. 20.

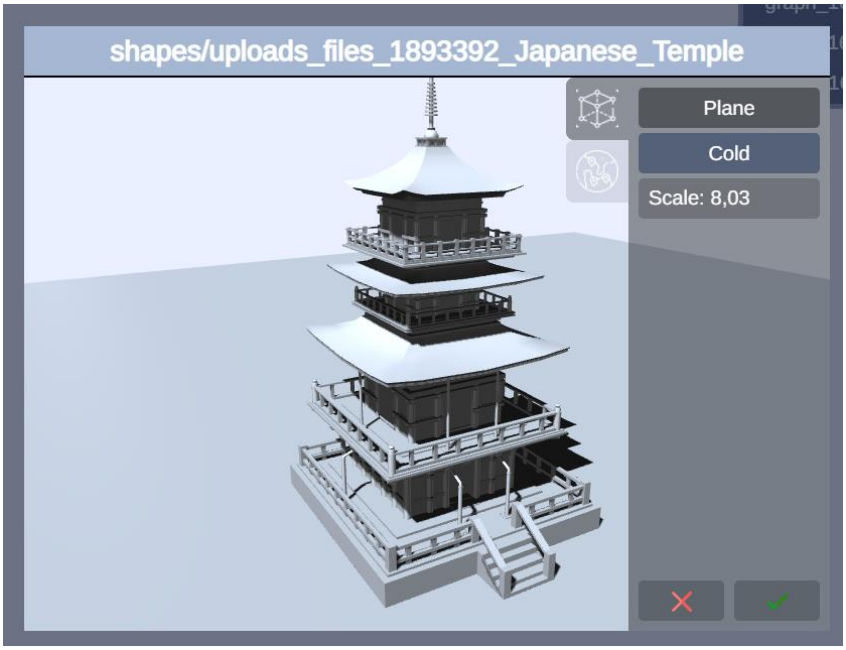

Рисунок 20 – Диалог редактирования загружаемой модели

После изменения характеристик, пользователь может как сохранить загруженный объект в вершину, так и отменить его изменение, если, например, данная модель была открыта уже из существующей вершины.

Диалог редактирования места

Данное окно позволяет полностью сконфигурировать вершину биома. Содержит пункты редактирования, которые были описаны ранее, такие как выбор типов биома, кластеризацию, насыщенность, хаотичность, полноту.

Визуализирует упрощенный алгоритм расположения объектов на ландшафте. Объектами служат псевдослучайные прямоугольники, распределенные на основе параметров. Данный ландшафт обновляется в режиме реального времени в зависимости от выбора параметров в кнопках слайдерах.

Построение биомов для небольшого участка ландшафтов производиться по принципу выбранного основного биома и четырех псевдослучайных соседей. Информация по каждому отдельному параметру, а также по определению места, представлена в специальном информационном окне. Данный диалог обращается к серверу для получения списка возможных биомов, это сделано для того, чтобы в будущем можно было без проблем расширять этот список. При этом упрощенный алгоритм выполняется на стороне пользователя, то есть на клиентской части приложения. Диалог редактирования места представлен на рис. 21.

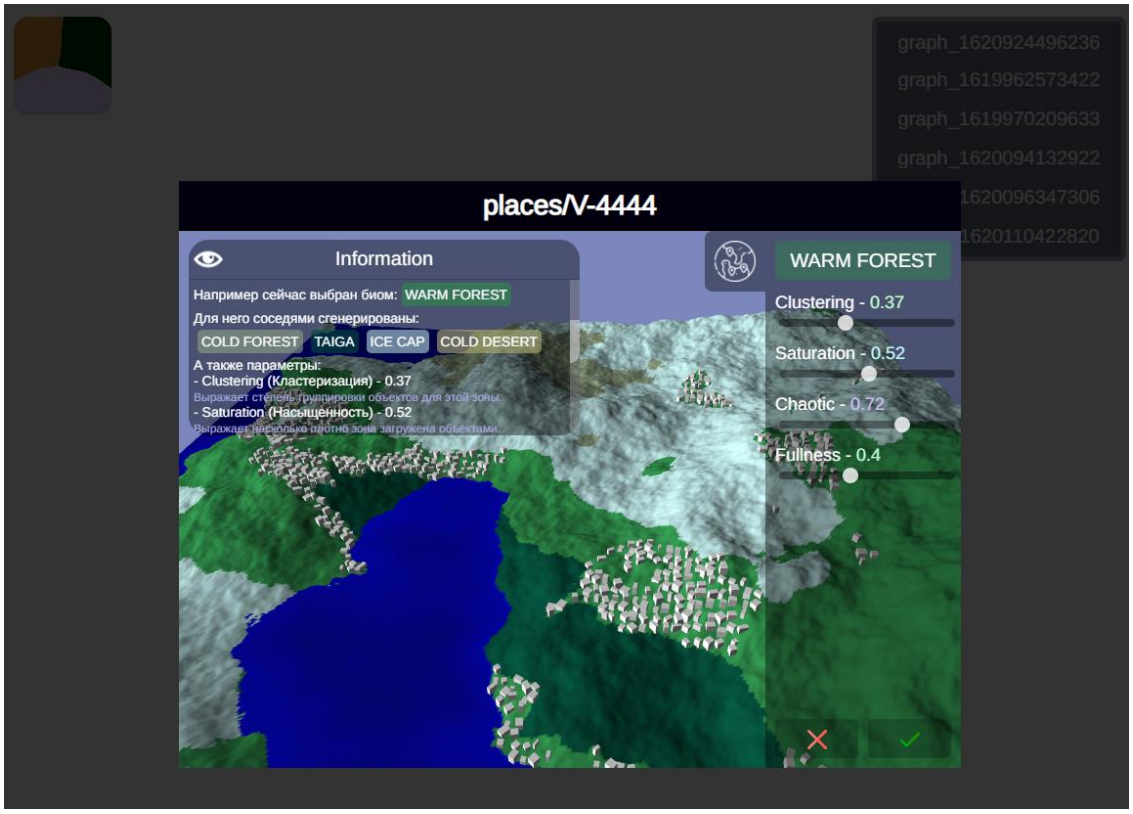

Рисунок 21 – Диалог создания и редактирования вершины местности

Диалог создания конечного ландшафта

Данный диалог открывается из синтетической вершины, предустановленной для каждого графа, которая является корнем для всех остальных вершин, конфигурируемых пользователем.

Открывается соответственно с вершины “World”, которую можно заметить на рис. 14, 16, 17. Предоставляет три поля задания ключевых

значений, которые для первой генерации случайно выбираются из справочника, заданного в компоненте графа.

Данный диалог имеет одну кнопку, на которую соответственно и запускается процесс создания ландшафта. Представлено окно на рис. 22.

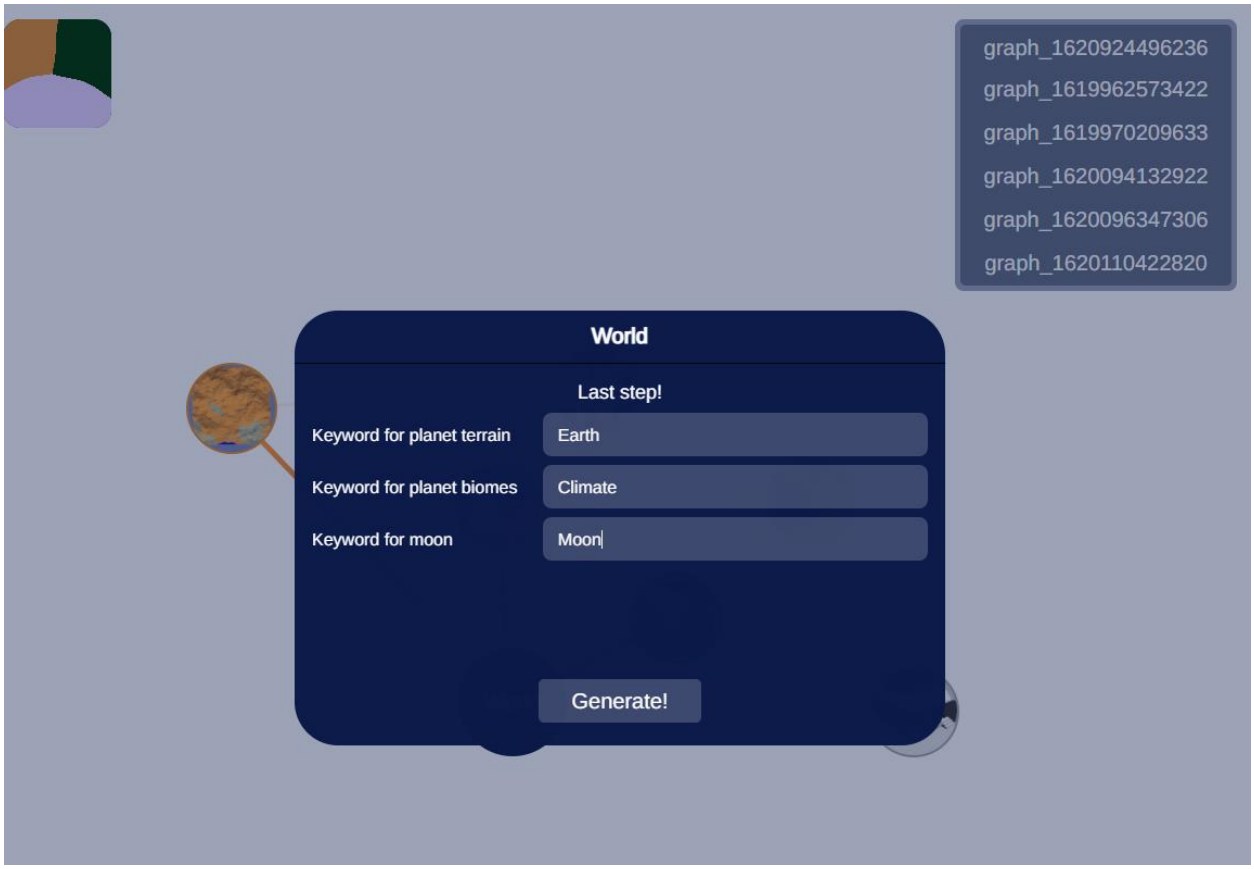

Рисунок 22 – Диалоговое окно создания конечного ландшафта

2) Страница визуализации ландшафта

Представляет из себя интерфейс созданного трехмерного ландшафта в планетарном масштабе. Страница представлена на рис. 23.

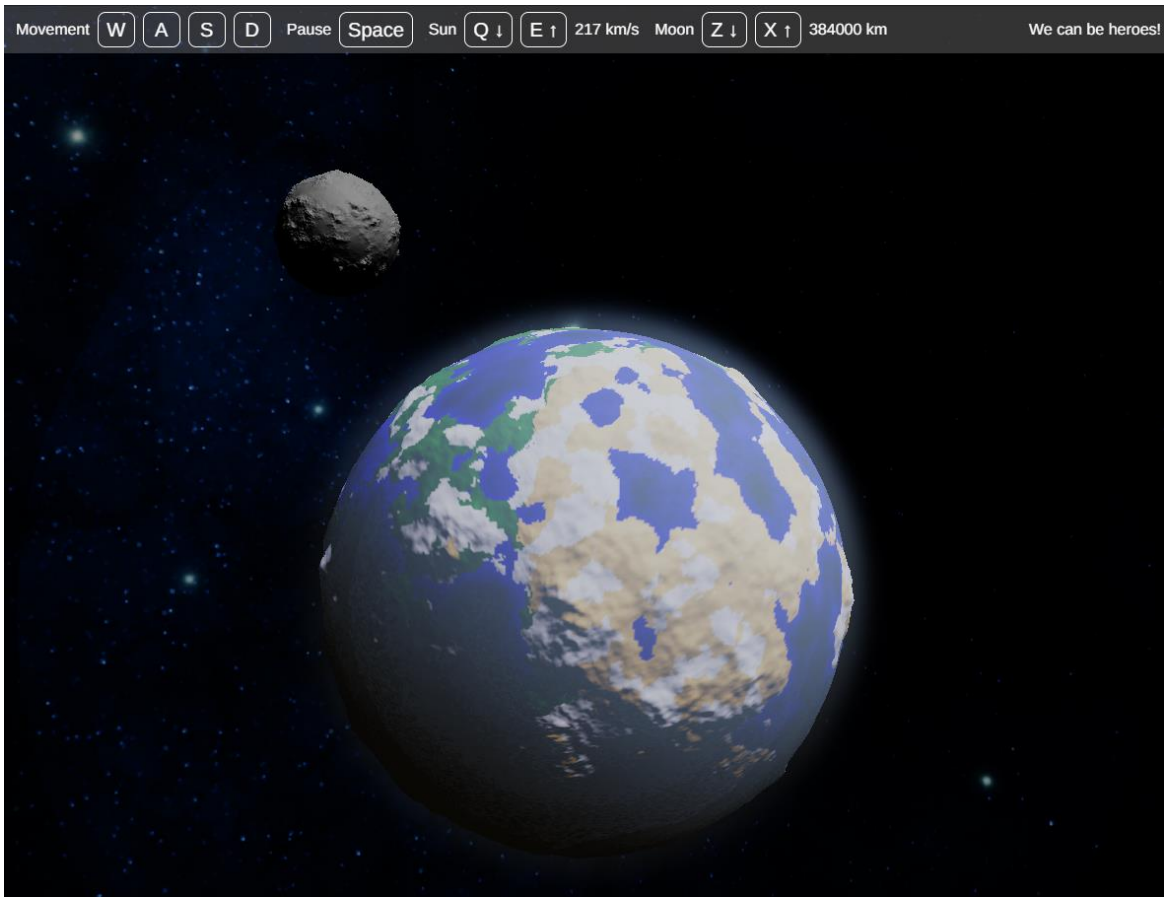

Рисунок 23 – Страница визуализации сгенерированного ландшафта

На данной странице присутствует шапка, в которой описаны основные клавиши для управления камерой пользователя, описаны характеристики состояния ландшафта на данный момент, а именно скорость движения солнца, которую можно увеличить с помощью заданных клавиш, расстояние от луны до земли, а также время, текущее на созданном ландшафте планеты в зависимости от координат камеры и положения солнца в данных координатах. При выходе из атмосферы планеты, вместо времени отображается сгенерированная фраза из справочника в компоненте визуализации ландшафта.

Все клавиши, задействованные пользователем в данный момент отображаются с помощью подсветки их в шапке страницы. Пример с

движением назад, включенной паузой, увеличенной скоростью движения солнца, а также с визуализацией процесса течения времени, см. рис. 24.

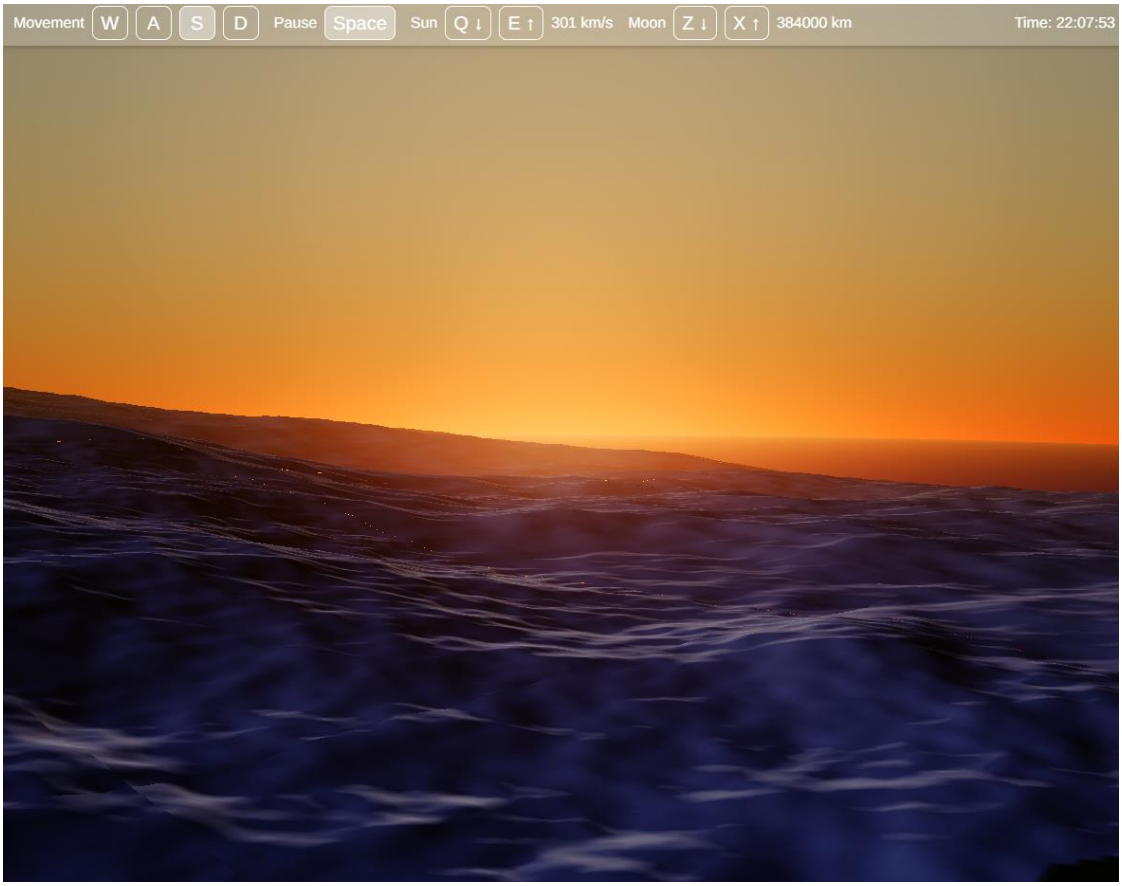

Рисунок 24 – Визуализация процессов взаимодействия пользователя со страницей

Таким образом, были реализованы основные страницы и компоненты, используемые для создания пользовательского интерфейса.

3.2. Архитектура программной реализации

3.2.1. Общая архитектура приложения

Для реализации данной разработки выбрана клиент-серверная архитектура приложения. Такой способ организации приложения представляет собой распределенную по задачам приложения сферам. Основной сферой для реализации большинства вычислений является сервер. Клиент формирует запросы, а также служит выводом данных приходящих с сервера. Сервер обрабатывает данные запросы и отправляет необходимые данные.

Язык и правила запросов определяются протоколами связи между двумя сторонами обмена. Для формализации обмена данными сервер реализует интерфейс прикладной реализации запросов – API. API является уровнем абстракции для доступа к различным методам, использующимся на сервере, через отправку запросов.

В данном приложении сервер организует большие вычисления с помощью разработанных алгоритмов, а также является посредником для сохранения конфигурационных файлов в системе хранения данных.

3.2.2. Описание используемых библиотек

На рис. 25 представлены основные технологии, библиотеки и алгоритмы, используемые на стороне клиента и сервера

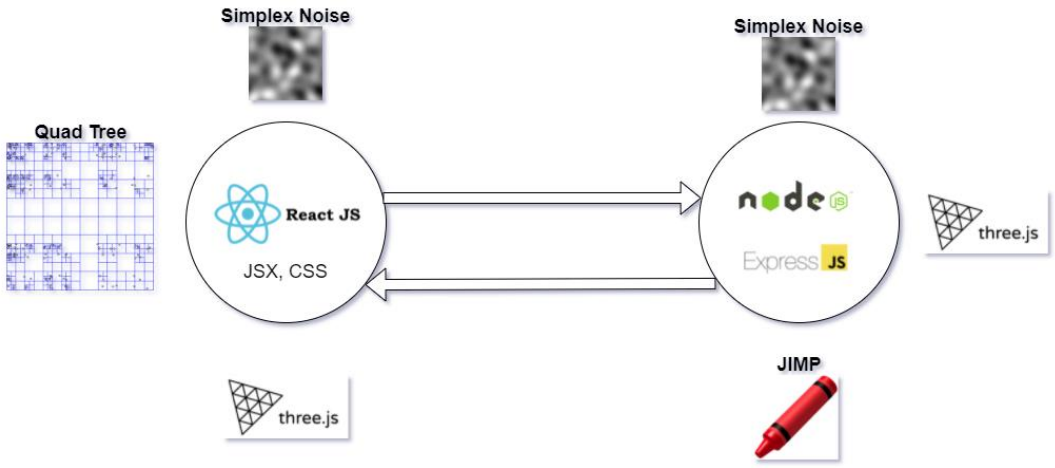

Рисунок 25 – Технологии, используемые на сервисе

На представленной схеме видно, что алгоритм симплексного шума используется как на сервере, так и на клиенте. Это реализовано для того, чтобы клиентская сторона приложения могла моментально производить операции по созданию упрощенных ландшафтов в диалоге редактирования местности.

Также Three.js, являясь средством визуализации, используется на серверной стороне приложения для достижения однородности воспроизводимых данных, посылающихся на клиент для рендеринга в визуальной части, также Three.js использует стандартные математические

преобразования трехмерных моделей, что не противоречит использованию сервиса с помощью программного интерфейса.

Quad Tree строится исключительно на клиентской стороне, сервер в конечном итоге в запросах получает просто смещение участка ландшафта в пространственных координатах. Это, во-первых, не ограничивает прикладных разработчиков в выборе своего средства детализации ландшафта и передачи собственных смещений участков местности, а во-вторых позволяет делегировать обязанности по вычислению смещении участка на сторону клиента, что значительно ускоряет процесс получения итоговых участков, так как пока алгоритм моментально просчитывает каждый кадр рендеринга на смещение камеры в дереве квадрантов, сервер выполняет работу по детализации только новых участков, которые появляются по мере движения камеры пользователя.

Jimp – это сторонняя библиотека, которая используется для преобразований изображений [19]. Так как шумовые функции изначально являются фильтрами для определенных изображений, а также климатическая карта является изображением, то в контексте выполнения работы требуются механизмы оптимизации работы над изображениями, которые и предоставляет данная библиотека.

Технологии Express JS, Node.js, React были описаны ранее в пункте 2.2 с обоснованием выбора данных программных средств в разрабатываемом приложении.

3.2.3. Архитектура реализации клиентской части

Клиентская сторона построена на основе компонентного подхода, компоненты представляют из себя классы с определенными состояниями и методами. UML диаграмма взаимодействия, см. рис. 26.

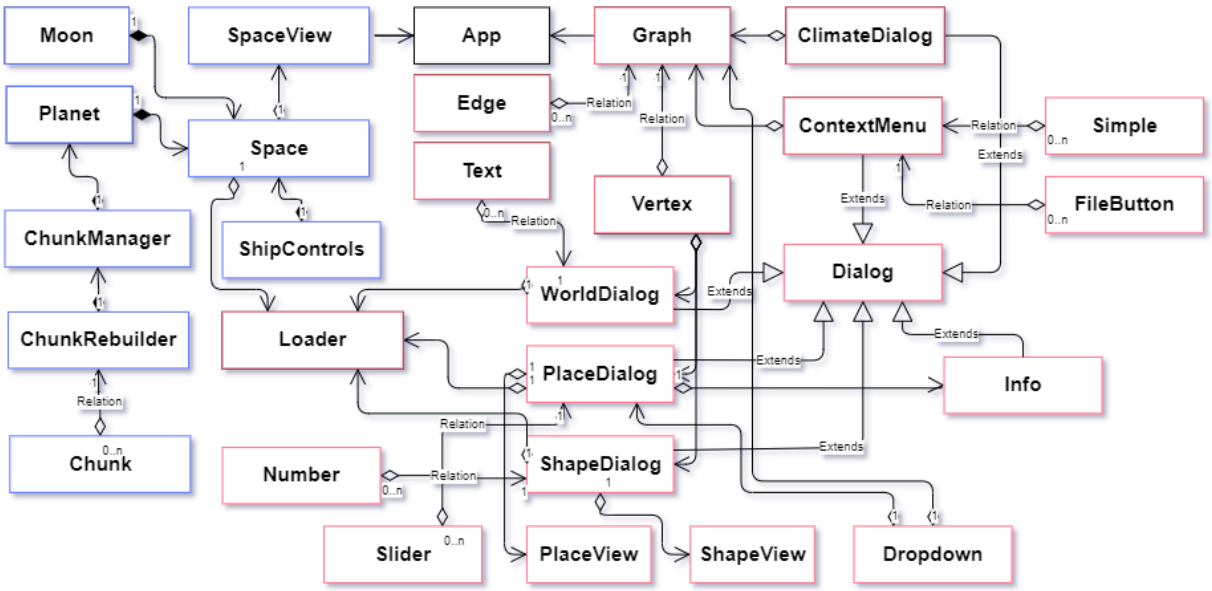

Рисунок 26 – UML-диаграмма взаимодействия классов клиентской части приложения

Из представленной диаграммы видно, что классы в основном делятся на две непересекающиеся категории, соответствующие страницами приложения. Данные категории размечены цветами, синий и красный соответственно для страницы ландшафта и страницы графа соответственно. Классы со стороны страницы с графом уже были описаны в виде компонентов в разделах 3.1.1 и 3.1.2. Более подробного рассмотрения требуют сущности, составляющие страницу визуализации ландшафта.

SpaceView – класс контейнер, является прослойкой между чистой реализации трехмерного моделирования в браузере и компонентом React. Передает DOM-элемент для создания контекста рендеринга моделей. А также реализует взаимодействия пользователя и приложения, путем обработчиков ввода с клавиатуры. Вызывает публичные методы класса Space для обновления параметров с учетом поступающих событий. Также реализует шапку страницы, с подробной информацией о состоянии ландшафта на данный момент.

Space – является потомком SpaceView, реализует рендеринг трехмерных фигур в контексте DOM-элемента переданного из SpaceView. В данном классе реализуется установка рендера элементов трехмерного моделирования, сцены

для изображения данных моделей, происходит также настройка конфигурации камеры пользователя и установка света на сцене рендеринга. Создает новые экземпляры классов планеты и луны. Предоставляет публичные методы для обновления состояния скорости движения солнца (угол наклона света), и положения луны. Запускает цикл анимации. Подключает пространственный шейдер для реализации эмуляции реалистичной атмосферы планеты, рис.1.

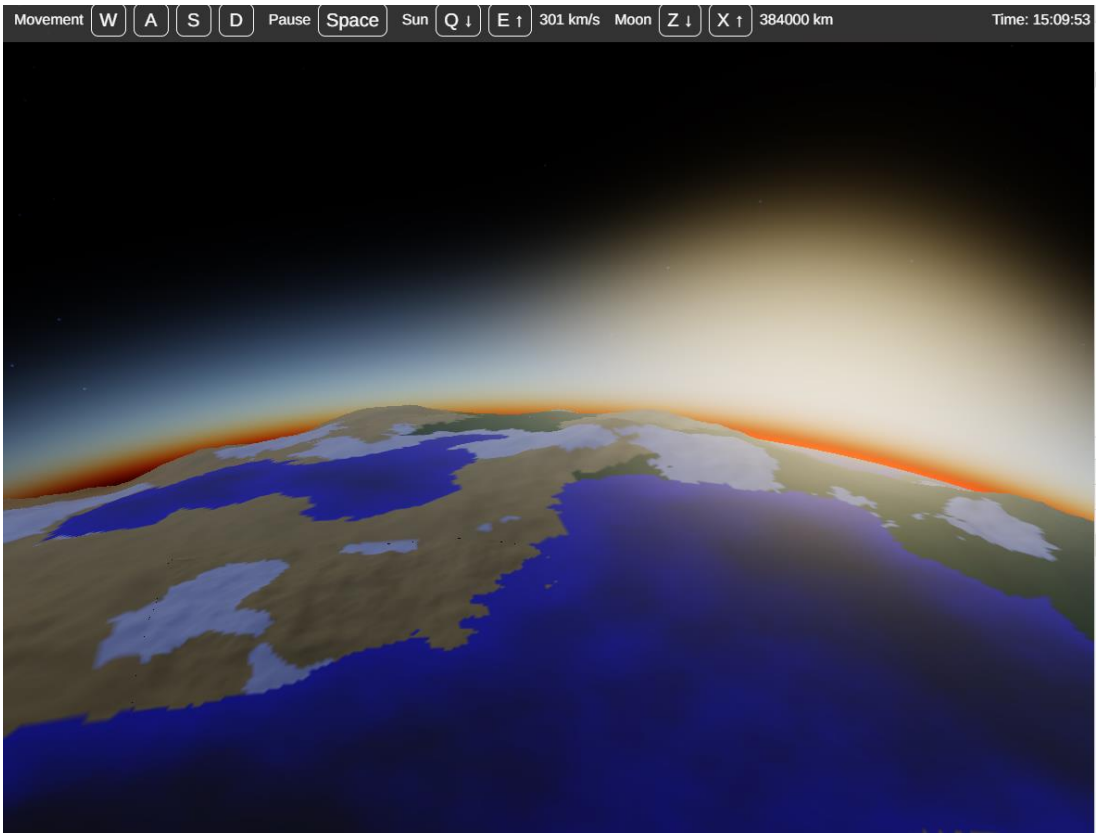

Рисунок 27 – Иллюстрация работы шейдера

Данный шейдер реализует упрощенное преломление света в атмосфере некоторого объекта, с помощью критерия Рэлея на основе рассеивания воздуха над планетой. [20] В космическом масштабе с помощью этого шейдера появляется небольшая дымка при просмотре неосвещенной стороны земли. При приближении к планете, свет распределяется в голубое небо с более светлым пятном солнца.

Planet – класс инициализирует новую планету с помощью запроса на сервер, в который передает сконфигурированный граф связей биомов и

объектов. Создает экземпляры класса ChunkManager и QuadTree для каждой стороны планеты.

Планета строится на основе кубической сферы [21], что позволяет использоваться алгоритм детализации с помощью дерева квадрантов, а также такой подход менее подвержен коллизиям, возникающим с детализацией обычной сферы, так как размеры базовых фигур для стандартной сферы уменьшаются к полюсам сферы. Кубическая сфера позволяет создать для каждой стороны отдельную сущность дерева квадрантов и с большей точностью определять местоположение камеры пользователя для детализации. Пример преобразования из куба в данную сферу представлен на рис. 28.

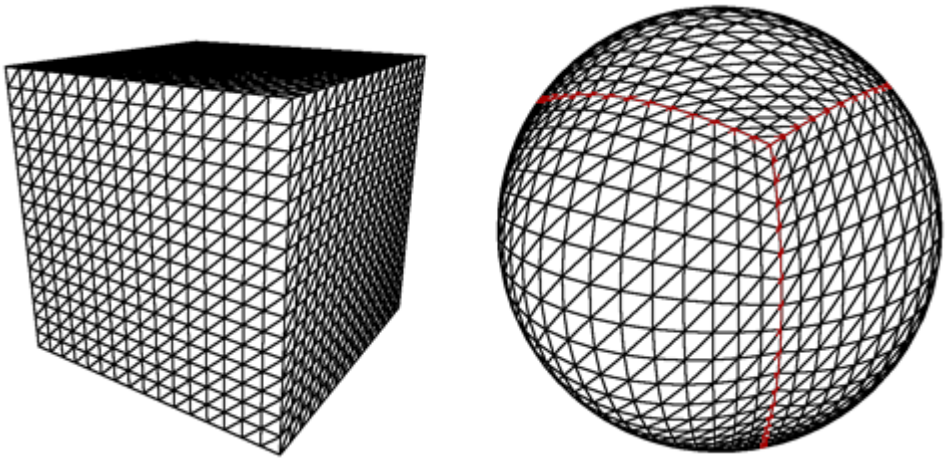

Рисунок 28 – Преобразование куба в сферу

ChunkManager – класс обеспечивает распределение нагрузки детализации. Создает пулл, в который помещаются участки требующие детализации, и отправляет данный пулл в ChunkRebuilder. Управляет обновлением дерева квадрантов.

Занимается выгрузкой и предобработкой пользовательских моделей с сервера, для этого на основе идентификатора графа запрашивает сохраненные бинарные файлы всех прикрепленных моделей и создает для них позиционирование в центре координат {0, 0, 0}. Передает базовые модели в ChunkRebuilder для дальнейшего рендеринга на основе карты координат каждой модели.

ChunkRebuilder – класс предназначен для отправки и получения данных о участках с серверной части приложения. Распределяет нагрузку на сервер с учетом заданного количества открытых запросов, синхронизирует обновление и создание участков с учетом ответом от сервера. Параметрами для создания участков служат уровень детализации участка, смещение относительно базовой стороны, матрица поворота участка в пространстве, а также отправляется радиус и максимально возможная высота ландшафта.

На заданном уровне, дополнительно в параметрах, которые передаются на сервер, отправляется флаг о том, что требуется сформировать карту расстановки моделей на участках. Это сделано для того чтобы, не было наложения одной карты на другую на каждом уровне детализации, а также это очень сильно оптимизирует визуализацию и время обработки запроса на получения карты координат моделей, так как не имеет смысла конфигурировать карту на тех участках, которые не попадают в поле зрения камеры пользователя.

Chunk – Класс реализует буфферную геометрию участка, и добавляет в атрибуты буффера данные, полученные из ChunkRebuilder.

ShipControls – Реализует преобразования движения камеры в виде “от первого лица” с учетом поворота над планетой. Создает объект управления камерой. Преобразует сферические координаты движения мышью в зоне SpaceView в декартовы координаты цели просмотра камеры с помощью формул

$$\begin{cases} x = r * \sin(\theta) * \cos(\phi) \\ y = r * \sin(\theta) * \sin(\phi) \\ z = r * \cos(\theta) \end{cases}$$

Где r – является радиусом заданной планеты, θ – угол поворота мыши относительно горизонтали, а ϕ – угол поворота относительно вертикали [22].

Поскольку данная реализация просто преобразует сферические координаты в пространственные, камера в таком случае, не будет описывать окружность вокруг движения планеты, также камера будет всегда

перпендикулярная плоскости основания. Поэтому требуется повернуть камеру на определенный кватернион, который строится на основе поворота нормали к центру планеты и единичного вектора нормали к первоначальной плоскости.

3.2.4. Архитектура реализации серверной части

Серверная часть приложения также следует принципам ООП и распределяет запросы на некоторые сущности, отвечающие за собственную функциональность. UML диаграмма серверной части приложения представлена на рис. 29.

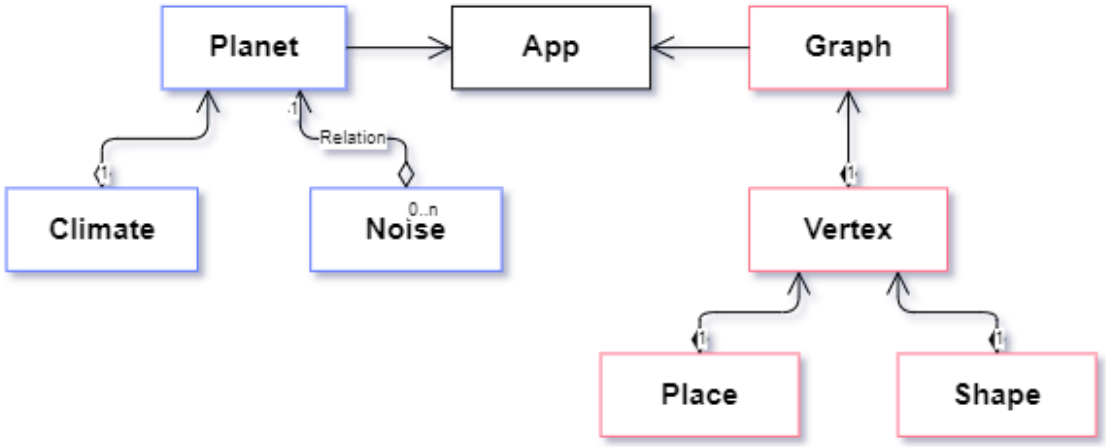

Рисунок 29 – UML диаграмма серверной стороны приложения.

Как видно из данной диаграммы она также разделена на две части по взаимодействию со страницей графа и построением основного ландшафта. В данной структуре сущность вершины уже не представляет повторно используемый компонент, на серверной стороне вершины определяют API запросов, которые предоставляются клиентскому интерфейсу.

Planet – класс реализует основное создание ландшафта и расстановки на нем объектов. Предоставляет два запроса для инициализации и процедурной генерации участков ландшафта.

При инициализации, с клиентской стороны приходят параметры графа, радиуса, и максимальной высоты карты. Далее создается новый экземпляр планеты, то есть массивного ландшафта. Создаются экземпляры класса Noise

предоставляющие шумовые методы для построения карт высот и карт влажности. Создается и сохраняется конфигурация планеты (прилож. json), обрабатывается и сохраняется климатическая карта ландшафта.

При процедурном строении ландшафта, в метод приходят параметры очередного участка. Такими параметрами являются:

1) Детализация определенного участка (resolution) – позволяет расширить выбор метода детализации на стороне клиентской программы. В нынешней реализации визуальной части каждый участка дерева квадрантов детализируется размером $100 * 100\ px$

2) Смещение участка по трем пространственным координатам (offset) – реализует правильное размещение сетки точек относительно своих соседей.

3) Матрица поворота участка (worldMatrix) – в параметр передается поворот первоначальной стороны кубической сферы.

4) Флаг для реализации карты объектов для участка (objects) – данный параметр приходит для определенного участка с уровнем, определенным на стороне клиента, что позволяет выбирать момент, в котором пользовательский интерфейс готов отобразить созданные трехмерные модели.

Полный алгоритм создания ландшафта выглядит следующим образом:

Для начала определяются сам вершины сетки на плоскости Z:

$$\begin{cases} xp_i = \dfrac{x_i}{resolution - 1} - 0.5 \\ yp_j = \dfrac{y_j}{resolution - 1} - 0.5 \end{cases}$$

Где i, j – обозначают индексы вершины в проходе по двумерному циклу.

Далее полученному вектору $\{xp_i, yp_j, 0\}$ прибавляется, заданное смещение $\{oy, ox, oz\}$. Полученный вектор $\{xp_i + ox, yp_j + oy, oz\}$ нормализуется и перемножается с матрицей поворота [23]. Таким образом получается участок кубической сферы, который передается параметром в метод объектов класса Noise для получения некоторого шумового равномерного значения. На основе полученных значениях высоты и климатического индекса определяется биом для обработки на будущем

ландшафте. В зависимости от наличия флага о создании объектов запускается алгоритм создания карты объектов на определенном участке местности с учетом смещение конкретного участка.

При создании карты объектов происходит следующее:

Совершается обход всех моделей, прикрепленных к графу, для каждой модели создается экземпляр класса Noise для определения шумовой функции на основе идентификатора объекта. Устойчивая процедурная генерация обеспечивает создание одной и той же шумовой функции для одного и того же объекта.

На основе параметра кластеризации формируется частота для экземпляра класса Noise. После генерации значения в заданной точке на участке кубической сферы рассчитывается шаг с которым нужно расставлять объекты на сетке с помощью формулы

$$step = Max(0.01, 1 - sat) * oz$$

Где sat – это заданный для родительского участка параметр насыщенности равный [0.01, 1.00], oz – смещение участка по оси z выражает уровень участка, например, для каждого участка $oz = 2^{level}$, где level = [-1, n] – уровень детализации участка. С помощью step отбрасываются координаты, которые не соответствуют полученному шагу, то есть при делении очередных координат на step остается остаток от деления.

Далее значение проверяется в соответствии с параметром полноты, при успешном прохождении порога и в соответствии родительской зоны и рассматриваемой в контексте ландшафта, ключ объекта добавляется в карту расстановки моделей на ландшафте, ключ в данный момент соответствует определенной модели, загруженной на клиентской стороне. Также добавляется хаотичное смещение и поворот в зависимости от параметра хаотичности.

Пример работы алгоритма построения пользовательских моделей на ландшафте с помощью методов шумовой псевдо случайно генерации представлен на рис. 30.

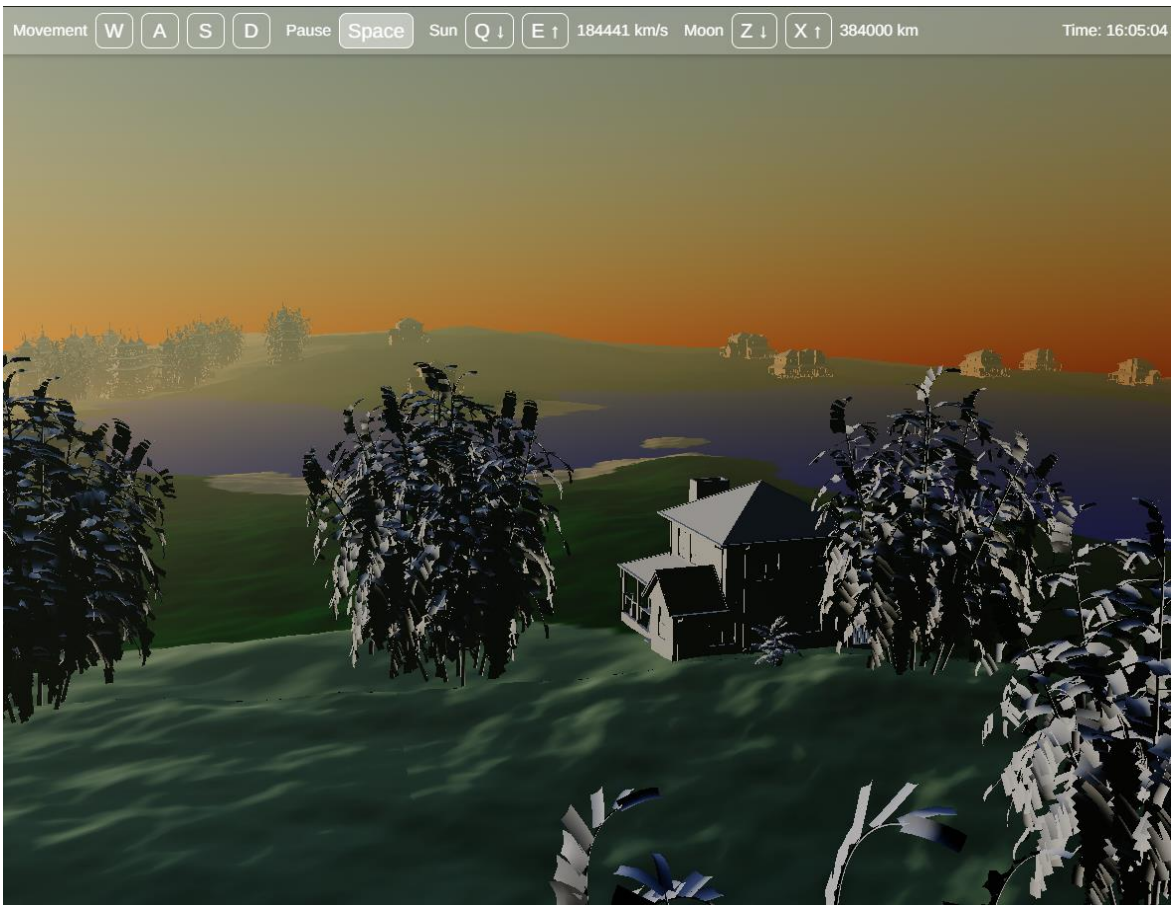

Рисунок 30 – Демонстрация результатов работы алгоритма псевдослучайной расстановки пользовательских моделей

Climate – объекты данного класса создаются в классе Planet и в классе Graph. Класс предназначен для построения климатической карты на основе связного ациклического графа. Описание алгоритма было представлено ранее.

Для перебора узлов дерева и составления множества точек используется рекурсивная функция, в ходе которой формируется массив значений будущих биомов. Далее на основе полученных значений формируется диаграмма Вороного с определением близости к ключевой точке с помощью метрики нормы Минковского:

$$\rho(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Где p – является порядком нормы, в данной работе было принято равным 3. Также предполагается наличие двух других метрик для определения расстояния, а именно Манхэттенское расстояние и Евклидова метрика [24].

Noise – класс реализует метод генерации значений шумовой функции на основе симплексного подхода. При инициализации задаются параметры:

Seed – зерно, ключевое значение для симплексного шума

Octaves – количество наложений шума

Frequence – частота распределения шума

Flatness – сглаживание поверхности

Amplitude – амплитуда разброса значений шума

На основе данных значений генерируется определенная величина в рамках переданной вершины в трехмерном пространстве.

Класс реализует статические методы для получения конфигурации биомов по ключу каждого из них. А также метод для генерации температуры на основе высоты определенной вершины, а также близости или отдаленности к экватору создаваемой планеты.

Graph – класс-модуль, который предоставляет внешнему интерфейсу следующие запросы:

POST-запрос: api/graph/climate – формирует климатическую карту на основе передаваемого графа.

POST-запрос: api/graph/save – сохраняет граф в системе хранения данных.

GET-запрос: api/graph/list – предоставляет список всех, созданных графов на сервисе.

GET-запрос: api/graph/load?=name – загружает конфигурацию определенного графа по его имени.

Vertex – класс модуль, реализует сохранение вершины любого типа, автоматически создает директории в системе хранения данных, если данная вершина например создается впервые.

Place – класс модуль, предоставляет запрос для получения актуального списка биомов представленных на сервисе, которые можно выбирать и конфигурировать на их основе вершину местности.

Shape – класс модуль, реализует два запроса по работе с вершинами объектами:

POST /api/shape/upload – сохраняет выбранный пользователем файл на сервере.

GET /api/shape/download?=file – отправляет файл объекта с сервера для дальнейшего преобразования на клиентской стороне.

3.3. Реализация системы хранения данных

Данный сервис предполагает хранение нескольких структур данных распределенных по категориям. Категории должны быть выделены в отдельные директории проекта. При взаимодействии с серверным API, сохраняются следующие основные директории:

- Planets/ – содержит в себе конфигурации планет, в частности всех ключевых параметров, ссылки на используемый граф, а также изображение карты биомов.
- Graphs/ – содержит конфигурации графов, в виде вершин и связей между ними, с визуальными составляющими, такими как стилизацию вершины и ее местоположение на холсте конфигурации графа
- Files/ – содержит исходные загружаемые файлы.
- Vertices/
    - Places/ – содержит конфигурации вершин биомов, со всеми, выбранными ранее характеристиками.
    - Objects/ – содержит конфигурации объектов, загружаемых пользователем, в частности ссылки на используемые файлы из директории Files.

Далее во всех директориях кроме Files, создаются поддиректории, которые пользователь указывает при именовании вершины биома, объекта, графа или планеты. Таким образом обеспечивается легкий способ поиска различных конфигураций для будущего проекта.

3.4. Выводы

В результате выполнения задач данной главы было реализовано приложение, которое предоставляет пользовательский интерфейс для визуализации работы, используемых алгоритмов, а также для ввода пользовательских объектов и ключевых значений. Также был разработан и описан программный интерфейс, то есть API, которое можно использовать в сторонних проектах, приложениях и сервисах.

ГЛАВА 4. ИССЛЕДОВАНИЯ СВОЙСТВ РЕШЕНИЯ

4.1. Быстродействие приложения

4.1.1. Быстродействие интерфейса

1) Количественные характеристики.

1.1) Количество шагов для получения новой вершины объекта [25].

Минимальным путь для получения объекта:

Открыть контекстное меню, нажать кнопку выбора объекта, выбрать файл, открыть меню в диалоге редактирования, сохранить.

Итого получается 5 шагов.

Максимальное взаимодействие с объектом:

Открыть контекстное меню, нажать кнопку выбора объекта, выбрать файл, повернуть модель с помощью мыши, открыть меню, изменить поверхность, изменить освещение, изменить размер, сохранить.

Итого 9 шагов.

Следовательно, можно сделать вывод, что среднее значение шагов для получения вершины объекта равно 7, стоит учитывать, что все шаги кроме поворотов и изменения размера, представляют собой простые действия.

1.2) Количество шагов для получения вершины места

Минимальное значение:

Контекстное меню, нажатие кнопки создания места, открытие меню, сохранение.

Итого 4 шага.

Максимальное значение:

Контекстное меню, нажатие кнопки создания места, поворот, просмотр информации, открытие меню, изменение типа биомов, настройка каждого из 4 параметров.

Итого 10 шагов

Среднее количество шагов равно 7, но конфигурация места, является более сложным процессом, так как требует точной настройки желаемого распределения объектов на местности.

2) Графики измерения скорости работы с пользовательским интерфейсом.

2.1) Зависимость времени предобработки модели от размера бинарного файла, представлен на рис. 31.

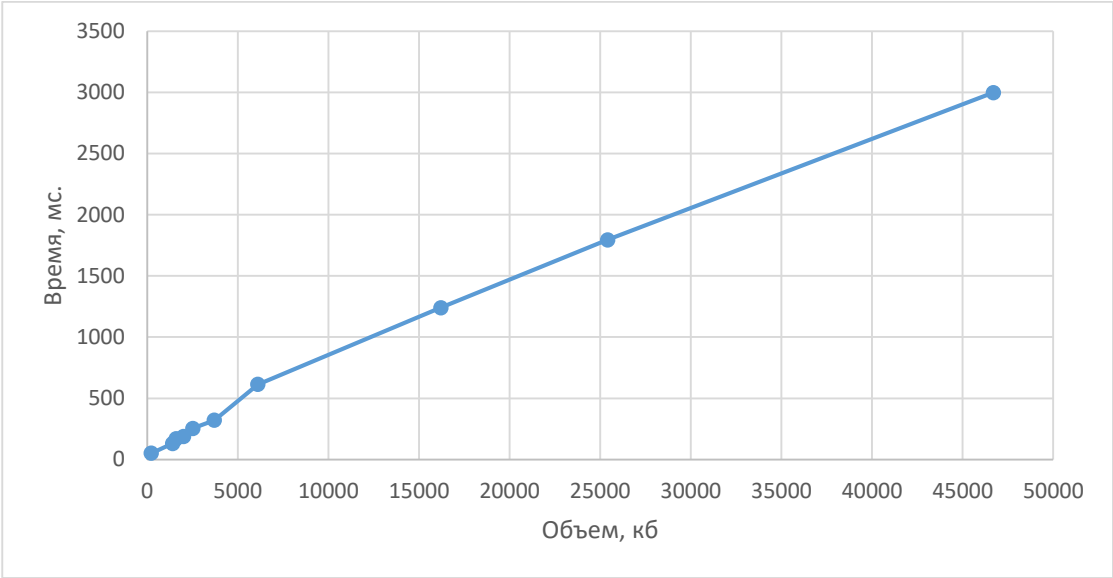

Рисунок 31 – График зависимости времени обработки модели, от первоначального объема бинарного файла

Из данного графика видно, что зависимость является линейной, а это означает что в разработанном механизме нет каких-либо сторонних эффектов над процессом загрузки модели.

2.2) График зависимости скорости загрузки графа от количества вершин представлен на рис. 32.

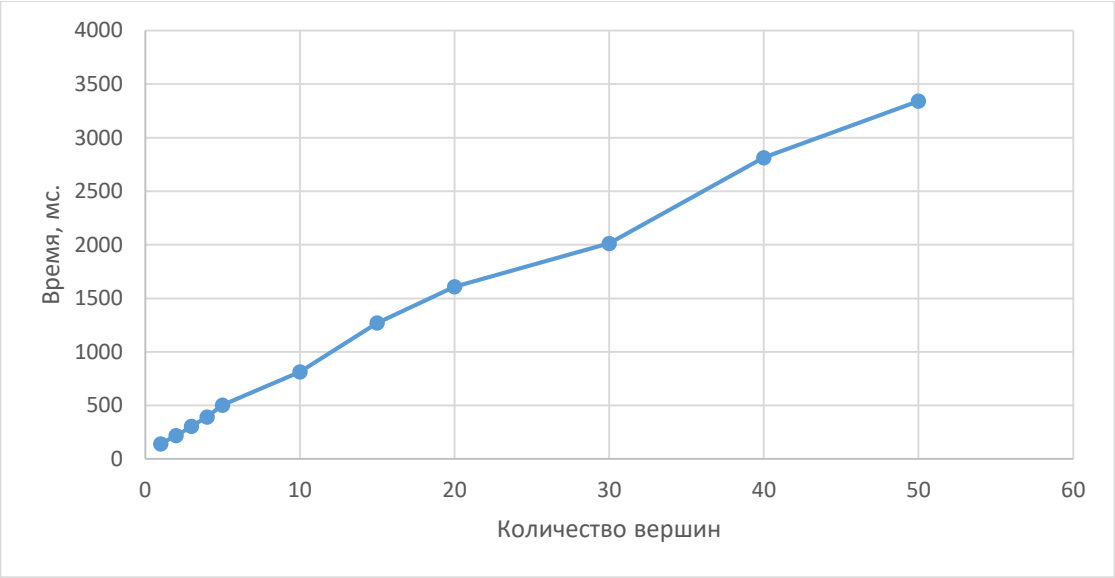

Рисунок 32 – График зависимости времени загрузки графа от количества вершин

В данном графике также наблюдается линейная зависимость, это подтверждает то, что граф как структура сконфигурирован правильно, нет никой дополнительной динамической информации, все вершины ссылаются на оригинальные вершины, сохраненные ранее.

4.1.2. Быстродействие сервера

1) Результаты сравнения скорости создания участков массивного ландшафта с объектами и без объектов, см. табл. 2. Также для каждого уровня расписано сгенерированное количество объектов в одной и той точке пространства, а также общее количество потенциальных объектов на всей стороне планеты с помощью формулы: $n * 4^l$, где n это количество объектов участка, а l – уровень детализации. Исследование данного свойства покажет оптимальность выбора 6 уровня детализации с объектоми.

Для тестирования быстродействия алгоритма, были выбраны следующие характеристики:

Граф с одной прикрепленной местностью, и одной вершиной объекта, см. рис. 33.

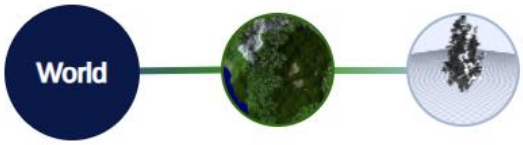

Рисунок 33 – Граф для тестирования быстродействия работы алгоритма по уровням

Характеристики для равномерного расположения объектов на местности, см. рис. 34.

Рисунок 34 – Характеристики единственной местности графа для тестирования

Таблица 2 – Результаты вычисления участков на каждом уровне детализации с объектами и без них

Уровень	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Кол-во объектов	461	215	323	281	180	78	20	3
Общее кол-во об.	461	860	5168	17984	46080	79872	81920	49152
С объектами, мс	229	136	196	145	154	168	184	150
Без объектов, мс	28	27	24	25	28	23	24	21

Данная таблица показывает, что выбор уровня детализации с объектами с 0 по 3 не имеет смысла, так как при относительно равных показателей скорости создания участка с объектами, на этих уровнях генерируется слишком малое количество объектов для одной стороны планеты. Уровни 4 и 7, генерируют примерно одинаковое количество объектов на весь участок, но проблема 7 уровня, в том, что явно видны границы где объекты перестают генерироваться. Таким образом оптимальными вариантами являются 4, 5, 6 уровни. На 6 уровне происходит генерируется максимальное количество объектов на сторону, при этом для рендеринга приходит всего 20 объектов, что позволяет довольно быстро выводить их на сцену ландшафта.

4.2. Сравнение с аналогами

Соответствие разработанного сервиса критериям, выдвинутым в подразделе 1.3.3.

Устойчивость

Разработанное приложение использует алгоритмы шумовой псевдослучайной генерации на основе ключевых значений. Пользователь вручную вводит ключи для создания карты высот и влажности ландшафта, а построение шума для распределения объектов на этом ландшафте происходит с помощью ключевых значений самих объектов и указанных для них характеристик. Следовательно, данное приложение можно считать полностью устойчивым к вводным данным.

Загрузка объектов

Приложение имеет отдельные механизмы для загрузки и конфигурации трехмерных объектов. Объект автоматически позиционируется в пространстве, и объекту можно присвоить собственные размеры. Приложение само конфигурирует области с объектами на ландшафте, то есть является пользователю не требуется вручную расставлять объекты.

Размер

Используются процедурные алгоритмы генерации псевдослучайных чисел, поэтому программный интерфейс позволяет с помощью смещения задавать сколь угодно большие отступы, при этом создавая очередной участок ландшафта, являющийся продолжением предыдущих. Пользовательский интерфейс в данной работе ограничен планетой, максимальный уровень детализации которой на данный момент равен 14, а каждый участок детализируется на основе 100 базовых пар треугольников, всего в планете участвует 6 сторон, это означает что конечный размер ландшафта равен:

$$s = 100 * 4^{14} * 6\ px^2 = 161\ 061\ 273\ 600\ px^2 = 401\ 324 * 401\ 324\ px$$

Таким образом с помощью оптимизации в браузере удалось создать больший ландшафт, чем представлен в трех аналогах, не удалось только достичь размеров многоядерных вычислений приложения World Creator, при это API может работать постепенно с любыми значениями размеров.

Дополнительное ПО

Приложение разработано с помощью клиент-серверной архитектуры, это позволяет обращаться к серверу через доступные протоколы обмена сообщениями, например, через HTTP в сети Интернет, поэтому для установки ничего не требуется. Пользовательский интерфейс реализован как онлайн сервис в браузере, поэтому также не требует никакого программного обеспечения.

API

Приложение четко распределяет задачи исполнения сервером и клиентской частью. Серверная сторона позволяет свободно подключаться к ее программному интерфейсу, единственным требованием является поддержка формата запросов и ответов со стороны клиентской части. За счет сведения всех механизмов псевдослучайной генерации к симплексному шуму, API может работать в реальном времени, что демонстрируется в процессе детализации ландшафта.

Сводная информация по результатам сравнения с аналогами представлена в табл. 3, разработанный сервис обозначен как Space World. Обозначения, используемые в таблице для каждого критерия:

Устойчивость – Да / Нет

Загрузка объектов – НП (не поддерживается) / РК (ручная конфигурация) / АК (автоматическая конфигурация)

Размер - $N * M\ px$

Дополнительное ПО – Нет / Да

API – РВ (в реальном времени) / ДО (длительные операции) / Нет (отсутствует)

Таблица 3 – Результаты сравнения разработанного сервиса с аналогами

Критерий / **Сервис**	**Устойчивость**	**Загрузка объектов**	**Размер**	**Доп. ПО**	**API**
World Creator	Да	НП	$268\,435\,456 * 268\,435\,456\ px$	Да	Нет
Instant Terra	Нет	НП	$64\,000 * 64\,000\ px$	Нет	Нет
World Machine	Да	РК	$12\,000 * 12\,000\ px$	Нет	ДО
3D Map Generator	Да	НП	$1\,750 * 1\,100\ px$	Да	Нет
Space World	Да	АК	$401\,324 * 401\,324\ px$	Нет	РВ

4.3. Выводы

Исследование свойств решения показало, что сервис полностью удовлетворяет выдвинутым критериям. Для реализации псевдослучайной генерации были выдвинуты верные параметры, в частности уровень детализации, на котором строятся списки объектов – трехмерных моделей для позиционирования их на будущем ландшафте – планете.

ГЛАВА 5. ОЦЕНКА И ЗАЩИТА РЕЗУЛЬТАТОВ ИНТЕЛЛЕКТУАЛЬНОЙ ДЕЯТЕЛЬНОСТИ

5.1. Описание результатов интеллектуальной деятельности

Разработанная программа для ЭВМ, представляет из себя онлайн-сервис для моделирования ландшафтов на основе загружаемых трехмерных моделей. Данный сервис предоставляет все механизмы для получения процедурно сгенерированного ландшафта. Таким образом сервис направлен на упрощение создания игровых или дизайнерских ландшафтов, так как предоставляет возможность генерации сразу с нужными моделями.

Пользователю предоставляется возможность сконфигурировать свой граф, на основе которого будет строится будущий ландшафт. Граф конфигурируется на основе двух пользовательских типов вершин и одной вершины корня. Существуют вершины для конфигурации места и загрузки объектов, и предустановленный корень, к которому данные вершины прикрепляются для создания структуры, пример представлен на рис. 14.

Вершина с типом объекта создается путем загрузки трехмерной модели на данный сервис. После загрузки пользователь попадает в окно редактирования модели (см. рис. 20) для более точной настройки, если требуется, например, изменения размера. Такие вершины могут быть только дочерними узлами вершин – мест.

Вершина с типом места создается через контекстное меню в программе, в окне редактирования места (см. рис. 21) появляется небольшой участок ландшафта, позволяющий настроить характеристики расположения объектов, прикрепленных к этому месту, на будущем ландшафте.

После настройки всех параметров и окончания конфигурации графа, вся информация отправляется на сервер для создания итогового ландшафта. На сервере выполняются основные вычисления, а пользователю уже приходят данные для рендеринга ландшафта.

Итоговый ландшафт и/или граф и каждые отдельные вершины можно сохранить на данном сервисе указав собственную директорию при сохранении и далее использовать уже готовые модели, например, загружая сконфигурированные вершины мест или объекты, а также целые графы для дальнейшего редактирования. Можно сразу сохранить весь ландшафт, если пока что не требуется его выгружать.

5.2. Оценка рыночной стоимости результата интеллектуальной деятельности.

5.2.1. Идентификация объекта оценки

Объектом оценки является программа для ЭВМ [26], позволяющая процедурно генерировать псевдослучайные ландшафты, основанные на ключевых вводных параметрах.

Процедурная генерация является основой для быстрого получения данных, то есть в реальном времени. Псевдослучайные методы генерации позволяют принимать определенные вводные параметры и получать на выходе одинаковые результаты, для тех же параметров, таким образом данная программа является устойчивой для длительной генерации ландшафтов.

В программе реализованы методы детализации массивных ландшафтов, таким образом сокращается использование памяти, и появляется возможность реализовывать массивные ландшафты в планетарном масштабе, см. рис. 23.

Программа позволяет генерировать ландшафты на основе пользовательских объектов, что позволяет создавать собственные ландшафты под конкретные проекты.

Из вышесказанного следует, что программа предоставляет возможность генерировать массивные ландшафты. Преимуществами данной программы являются возможность работы в реальном времени в игровых проектах в виде API, то есть обращению к программе с собственными параметрами для получения готовых данных, быстрая скорость такого взаимодействия обеспечивается алгоритмами детализации и процедурной генерации на основе

ключевых параметров. Также программа имеет собственный интерфейс, доступный в сети Интернет для наглядного примера работы по составлению структур данных вводных параметров (пункт 1.1, рис. 1, 2, 3), а также для получения готовых ландшафтов в виде файлов с расширениями трехмерных объектов для реализации данного ландшафта в других программах моделирования или собственных игровых или дизайнерских проектах.

5.2.2. Анализ рынка объекта оценки

Данную программу можно отнести к комплексу средств для моделирования ландшафтов, как было сказано в пункте 1.2.1 преимуществами программы на данном рынке являются массивность ландшафтов, процедурная генерация, устойчивость к вводным параметрам, построение на основе пользовательских трехмерных моделей.

Данная программа может использоваться в игровой, дизайнерской индустрии и научной сфере. Таким образом продукты, реализуемые с помощью данной программы могут представлять из себя другие программы, к примеру игры, или программы для моделирования взаимодействий массивных космических объектов, программы для генерации городов или лесных массивов, изображения основанные на данных полученных из программы, могут быть вырезаны как из готовой визуальной реализации программы, так и на основе собственной визуализации данных, видеоролики основанные на тех же принципах реализации, что и изображения.

5.2.3. Анализ использования объекта оценки

Данная программа прежде всего нужна для ускорения производства продуктов - программ, требующих наличия процедурных ландшафтов. Таким образом сложно оценить показатели производства других научных или развлекательных программ, где будет использоваться данная разработка. Сфера дизайна также является по большей части творческой деятельностью, и использование продукции в данной сфере является дополнением к основным возможностям программы. Таким образом значительного ускорения

реализации продукции в индустрии дизайна не предполагается, но получить статистическую оценку использования в данной сфере также не представляется возможным.

5.2.4. Расчет рыночной стоимости

Данная программа на данный момент не используется в коммерческих целях, а оценка использования является крайне неточной и до момента выхода на рынок нет возможности уточнить данную оценку, поэтому для оценки рыночной стоимости был выбран затратный подход, метод учета затрат на замещение или восстановление продукта.

Стоимость разработки программы для ЭВМ будет осуществляться по следующим параметрам:

- Расходы на оборудование;
- Основная заработная плата;
- Дополнительная заработная плата;
- Отчисления на социальные нужды;
- Накладные расходы.

Расходы на оборудование представляют собой арендную плату за использование компьютерного времени. Величина арендной платы рассчитывается по формуле:

$S_{\text{ар}} = T_{\text{ар}} * O_{\text{ар}}$, где $S_{\text{ар}}$ – сумма арендной платы, $T_{\text{ар}}$ – время аренды компьютера, $O_{\text{ар}}$ – часовая ставка использования компьютера, которая рассчитывается по формуле:

$O_{\text{ар}} = \frac{C_{\text{пк}}}{T_{\text{пк}} * D_{\text{пкг}} * T_{\text{пкд}}}$, где $C_{\text{пк}}$ – первоначальная стоимость оборудования (руб.); $T_{\text{пк}}$ – нормативный срок службы оборудования (лет); $D_{\text{пкг}}$ – количество дней работы оборудования в календарном году; $T_{\text{пкд}}$ – продолжительность рабочего дня (час).

Составим сводную таблицу расходов на оборудование, см. табл. 4.

Таблица 4 – Данные для расчета расходов на оборудование

Показатель	**Значение**
Время аренды компьютера $T_{\text{ар}}$, час	1650 (275д. * 6ч.)
Первоначальная стоимость оборудования $C_{\text{пк}}$, руб.	145 000
Срок службы оборудования $T_{\text{пк}}$, год	3
Количество дней работы оборудования в календарном году $D_{\text{пкг}}$, день	275
Продолжительность рабочего дня $T_{\text{пкд}}$, час	6

Таким образом часовая ставка получается:

$$O_{\text{ар}} = \frac{145\,000}{3 * 275 * 6} = 29.3 \text{ руб.}$$

А сумма арендной платы: $S_{\text{ар}} = 1\,650 * 29.3 = 48\,345$ руб.

Основная заработная плата рассчитывается как сумма произведений трудоемкости работ каждого исполнителя на его среднюю дневную ставку, так как в данной работе один исполнитель, то применимо обычное произведение.

$S_{\text{озп}} = \sum_{i=1}^{1} T_{\text{авт},i} * O_{\text{авт},i} = T_{\text{авт}} * O_{\text{авт}}$, где $T_{\text{авт}}$ – трудоемкость выполнения работа автором в днях, $O_{\text{авт}}$ – средняя дневная ставка автора, руб.

Для определения стоимости разработки составим перечень выполненных работ и их трудоемкость для одного автора, см. табл. 5.

Таблица 5 – Трудоемкость работ по созданию программы для ЭВМ

Вид работ	**Трудоемкость (чел./дни)**
	Автор
1. Изучение литературы	40
2. Сбор информации в интернете	25
3. Разработка макетов	20
4. Разработка архитектуры	25
5. Разработка структуры хранения	20
6. Реализация программы	125
7. Отладка и тестирование	15
8. Оформление документации	5
Всего	**275**

Дополнительная заработная плата $S_{\text{доп}}$, начисления на зарплату $S_{\text{сн}}$ и накладные расходы $S_{\text{нр}}$ определяются как:

$$S_{\text{доп}} = S_{\text{озп}} * P_{\text{доп}}/100,$$

$$S_{\text{сн}} = \left(S_{\text{доп}} + S_{\text{озп}}\right) * P_{\text{сн}}/100,$$

$$S_{\text{нр}} = \left(S_{\text{доп}} + S_{\text{озп}}\right) * P_{\text{нр}}/100,$$

Где $P_{\text{доп}}$ – процент дополнительной заработной платы; $P_{\text{сн}}$ – процент начислений (единый социальный налог); $P_{\text{нр}}$ – процент накладных расходов.

К статье «Дополнительная заработная плата» относятся выплаты, предусмотренные законодательством за непроработанное время, оплата отпусков, выплаты премий, вознаграждения и т. п. К статье «Отчисления на социальные нужды относятся отчисления на социальное страхование; отчисления на пенсионное обеспечение; отчисления на медицинское страхование; отчисления в фонд занятости и т. д. К статье «Накладные расходы» относятся расходы на управление и хозяйственное обслуживание. Размеры дополнительной заработной платы и отчислений на социальные нужды устанавливаются нормативными документами, накладные расходы определяются организацией:

Размеры отчислений на социальные нужды устанавливаются нормативными документами, накладные расходы и размеры дополнительной заработной платы определяются организацией, сводная информация для расчета заработной платы представлена в табл. 6.

Таблица 6 – Сводная информация для расчета заработной платы

Показатель	**Автор**
Трудоемкость выполнения работы, день	275
Дневная ставка, руб	1500
Дополнительная зарплата, %	25
Начисления на зарплату (единый социальный налог), %	30
Накладные расходы, %	5

Таким образом получается:

$$S_{\text{озп}} = 275 * 1\,500 = 412\,500 \text{ руб.}$$

$$S_{\text{доп}} = 412\,000 * \frac{25}{100} = 103\,125 \text{ руб.}$$

$$S_{\text{сн}} = (103\,125 + 412\,500) * \frac{30}{100} = 154\,687.5 \text{ руб.}$$

$$S_{\text{нр}} = (103\,125 + 412\,500) * \frac{5}{100} = 21\,656.25 \text{ руб.}$$

На основании полученных данных по отдельным статьям затрат можно составить калькуляцию себестоимости разработки в целом, см. табл. 7.

Таблица 7 – Результаты расчета себестоимости разработки.

Статья затрат	**Сумма, руб.**
Расходы на оборудование	48 345
Основная заработная плата	412 500
Дополнительная заработная плата	103 125
Отчисления на социальные нужды	154 687.5
Накладные расходы	21 656.25
Итого себестоимость $S_{\text{себ}}$	**740 313.75**

Итоговая рыночная стоимость разработки данного приложения составила 740 313.75 рублей. Данная сумма была посчитана на основе затратного метода на замещение или полное восстановление приложения.

5.3. Правовая защита результатов интеллектуальной деятельности

Правовая защита результатов интеллектуальной деятельности регулируется Гражданским кодексом Российской Федерации (часть IV), Кодексом об административных правонарушениях Российской Федерации, Уголовным кодексом Российской Федерации.

5.3.1. Перечень нормативно-правовых актов

Гражданско-правовые способы защиты

В соответствии со статьями 12, 1252 Гражданского Кодекса (далее - Кодекс) обладатели исключительных прав на объекты промышленной собственности, авторских и смежных прав вправе требовать от нарушителя [27]:

1) Признания прав.

Признание прав может относиться к личным неимущественным правам авторов и исполнителей или к имущественным правам.

Признание права может сопровождаться публичным объявлением о существовании определенного права, которое делается нарушителем или за его счет.

2) Восстановления положения, существовавшего до нарушения права, и пресечение действий, нарушающих право или создающих угрозу его нарушения.

Восстановление прежнего положения возможно далеко не всегда, эта мера защиты в некоторых случаях все же может быть применена (уничтожение экземпляров, проставление на экземплярах имен создателей и т. п.).

Прекращение действий, составляющих правонарушение или создающих угрозу правонарушения в сфере авторского права и смежных прав, может быть применено практически всегда. Это запрет рекламы, предложения продажи экземпляров, запрет продажи, допечатки тиража и т. п.

3) Возмещения убытков, включая упущенную выгоду.

Обладатель исключительных прав, выдвигая требование о возмещении убытков, должен доказать факт наличия убытков, их размер, а также то, что убытки были причинены действиями нарушителя.

Обычно в сфере интеллектуальной собственности убытки проявляются в форме упущенной выгоды, т. е. той суммы, которую правообладатель мог бы

получить, если бы нарушитель заключил договор с правообладателем и использовал охраняемый результат интеллектуальной деятельности возмездно, на законных основаниях.

4) Взыскания дохода, полученного нарушителем вследствие нарушения авторских и смежных прав, вместо возмещения убытков.

Обладатель исключительных прав может отказаться от требования о возмещении своих убытков и потребовать взыскать с нарушителя тот доход, который нарушитель получил от использования объектов промышленной собственности или авторского и смежного права (п. 2. ч. 2 ст. 15 Кодекса).

Доход должен быть получен нарушителем именно в результате нарушения прав патентообладателей, авторских и смежных прав, а не в результате законных действий.

Обладатель исключительных прав может потребовать выплаты ему доходов, полученных нарушителем, не отказываясь от возмещения реального ущерба.

5) компенсации морального вреда;

В сфере интеллектуальной собственности моральный вред подлежит компенсации лишь в тех случаях, когда нарушены какие-либо личные неимущественные права.

Требования о компенсации морального вреда регулируются статьями 150–152 и 1099–1101 Кодекса.

Размер компенсации морального вреда определяется судом в зависимости от характера причиненных потерпевшему физических и нравственных страданий, а также степени вины нарушителя в случаях, когда вина является основанием возмещения вреда. При определении размера компенсации вреда должны учитываться требования разумности и справедливости.

6) изъятия оборудования, прочих устройств и материалов, главным образом используемые или предназначенные для совершения нарушения исключительных прав на результаты интеллектуальной деятельности и на средства индивидуализации.

Требования предъявляются к изготовителю, импортеру, хранителю, перевозчику, продавцу, иному распространителю, недобросовестному приобретателю. По решению суда изъятое оборудование, устройства и материалы подлежат уничтожению за счет нарушителя, если законом не предусмотрено их обращение в доход Российской Федерации.

7) принятия иных предусмотренных законодательными актами мер, связанных с защитой интеллектуальных прав.

Помимо возмещения убытков, взыскания дохода или выплаты компенсации в твердой сумме суд или арбитражный суд за нарушение интеллектуальных прав взыскивает штраф в размере 10 % от суммы, присужденной судом в пользу истца. Сумма штрафов направляется в установленном законодательством порядке в соответствующие бюджеты.

Меры административно правового характера

Согласно пункту 2 статьи 7.12 Кодекса Российской Федерации об административных правонарушениях от 30 декабря 2001 г. № 195-ФЗ (далее – КоАП РФ) незаконное использование изобретения, полезной модели либо промышленного образца, за исключением случая недобросовестной конкуренции, выражающейся во введении в оборот товара с незаконным использованием результатов интеллектуальной деятельности, разглашение без согласия автора или заявителя сущности изобретения, полезной модели либо промышленного образца до официального опубликования сведений о них, присвоение авторства или принуждение к соавторству, влечет наложение административного штрафа на граждан в размере от одной тысячи пятисот до двух тысяч рублей; на должностных лиц - от десяти тысяч до двадцати тысяч рублей; на юридических лиц - от тридцати тысяч до сорока тысяч рублей [28].

Меры уголовно-правового характера

В соответствии со статьей 147 Уголовного кодекса Российской Федерации от 13 июня 1996 г. № 63-ФЗ (далее – УК РФ) незаконное использование изобретения, полезной модели или промышленного образца, разглашение без согласия автора или заявителя сущности изобретения,

полезной модели или промышленного образца до официальной публикации сведений о них, присвоение авторства или принуждение к соавторству, если эти деяния причинили крупный ущерб, наказываются штрафом в размере до двухсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до восемнадцати месяцев, либо обязательными работами на срок от ста восьмидесяти до двухсот сорока часов, либо лишением свободы на срок до двух лет [29].

Те же деяния, совершенные группой лиц по предварительному сговору или организованной группой, наказываются штрафом в размере от ста тысяч до трехсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период от одного года до двух лет, либо арестом на срок от четырех до шести месяцев, либо лишением свободы на срок до пяти лет.

В отношении объектов авторского права или смежных прав, согласно статье 146 УК РФ, их незаконное использование, а равно присвоение авторства, если это деяние причинило крупный ущерб, наказываются штрафом в размере до двухсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до 18 месяцев, либо обязательными работами на срок от ста восьмидесяти до двухсот сорока часов, либо арестом на срок от трех до шести месяцев.

Незаконное использование объектов авторского права или смежных прав, а равно приобретение, хранение, перевозка контрафактных экземпляров произведений или фонограмм в целях сбыта, совершенные в крупном размере, наказываются штрафом в размере от двухсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до восемнадцати месяцев, либо обязательными работами на срок от ста восьмидесяти до двухсот сорока часов, либо лишением свободы на срок до двух лет.

Если те же деяния совершены либо группой лиц по предварительному сговору или организованной группой, либо в особо крупном размере, либо лицом с использованием своего служебного положения, то в данном случае наказание предусматривается в виде лишения свободы на срок до шести лет

со штрафом в размере до пятисот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до трех лет либо без такового.

За защитой своего права обладатели исключительных авторских и смежных прав вправе обратиться в установленном порядке в суд, арбитражный суд, третейский суд, орган дознания, органы предварительного следствия в соответствии с их компетенцией.

По решению суда, арбитражного суда или судьи единолично контрафактные экземпляры произведений или фонограмм подлежат обязательной конфискации и уничтожению, за исключением случаев их передачи обладателю авторских или смежных прав по его просьбе.

Контрафактными являются экземпляры произведения и фонограммы, изготовление или распространение которых влечет за собой нарушение авторских и смежных прав.

Если изобретения, созданные в России и заявленные от имени российских юридических или физических лиц, первоначально патентуются в других странах, это явится прямым нарушением законодательства по интеллектуальной собственности. Согласно ст. 7.28 Кодекса Российской Федерации об административных правонарушениях от 30 декабря 2001 г. № 195-ФЗ (с последующими изменениями и дополнениями) нарушение установленного порядка патентования объектов промышленной собственности в иностранных государствах влечет наложение административного штрафа на граждан в размере от одной тысячи до двух тысяч рублей, на юридических лиц - от пятидесяти тысяч до восьмидесяти тысяч рублей.

5.3.2. Объем и сроки правовой защиты объекта

Программа для ЭВМ – это представленная в объективной форме совокупность данных и команд, которая предназначена для функционирования ЭВМ и прочих компьютерных устройств, чтобы получить определенный результат, включая подготовительные материалы, полученные

в ходе разработки программы для ЭВМ и порождаемые ею аудиовизуальные отображения (статья 1261 Гражданского кодекса).

Авторские права на любые программы для ЭВМ (в т. ч. на ОС и программные комплексы), которые могут быть выражены на любом языке и в любой форме, включая исходный текст и объектный код, охраняются так же, как авторские права на произведения литературы.

На программу для ЭВМ распространяются интеллектуальные права, включающие исключительное право, которое является имущественным правом, и личные неимущественные права.

Срок действия исключительных прав на программу для ЭВМ действует в течение всей жизни автора, который пережил своих соавторов, и 70 лет, считая с 1 января года, который следует за годом его смерти (это оговорено в Гражданском кодексе). Авторство, имя автора и неприкосновенность произведения будут находиться под охраной бессрочно.

Для возникновения, осуществления и защиты авторских прав нет необходимости в регистрации произведения или соблюдении каких-либо еще формальностей. Программу для ЭВМ можно официально зарегистрировать в Федеральной службе по интеллектуальной собственности, патентам и товарным знакам РФ, если этого хочет обладатель права (статья 1262 Гражданского кодекса).

5.4. Выводы

В результате работы по оценки результатов интеллектуальной деятельности, был определен тип объекта оценки – программа для ЭВМ, а также были описаны основные характеристики и особенности данного программного обеспечения.

Была получена рыночная стоимость разработанной программы для ЭВМ с помощью затратного подхода, так как в коммерческих целях представленная разработка на данный момент не используется. Результаты представлены в табл. 7. Полученная стоимость характеризует весь путь разработки от поиска

информации до выпуска готовой версии приложения, учитывая оборудование, задействованное в разработке и заработную плату автору проекта.

Был составлен перечень нормативно-правовых актов для защиты приложения по гражданско-правовым способам защиты, а также с помощью мер административно и уголовно правового характера.

ЗАКЛЮЧЕНИЕ

В исследовательской работе был проведен анализ предметной области. Были рассмотрены положительные и отрицательные стороны каждого приложения. На их основе были сформулированы свойства, которыми должен обладать разрабатываемый сервис.

В ходе решения поставленных задач были разработаны новые алгоритмы для псевдослучайной расстановки пользовательских объектов на местности, а также для создания климатической карты на основе связного ациклического графа пользовательских объектов. Данные алгоритмы смогут послужить основой или толчком для развития автоматизированного создания, наполненных объектами, карт местности.

В рамках задачи о предоставлении программного интерфейса были приняты решения, позволяющие свести все алгоритмы псевдослучайного генерации к реализации через устойчивые методы симплексного шума. Данное решение обеспечило доступ к получению небольшого участка ландшафта в реальном времени, большая часть параметров передается при первоначальной инициализации ландшафта, в дальнейшем используется только уточнение местоположения очередного участка. Данный подход был продемонстрирован на основе клиентской части приложения, которая запускается в браузере, с помощью алгоритма детализации, основанном на деревьях квадрантов.

Данная работа комбинирует в себе множество существующих на данный момент алгоритмов, методов и математических аппаратов создавая с помощью них совершенно новые алгоритмы и направления для изучения разработки процедурной псевдослучайно генерации ландшафтов на основе сторонних трехмерных моделях.

СПИСОК ИСТОЧНИКОВ ЛИТЕРАТУРЫ

1. Ken Perlin. «Noise hardware. In Real-Time Shading» SIGGRAPH Course Notes (2001), Olano M.
2. Hanan Samet, Robert Webber. «Storing a Collection of Polygons Using Quadtrees». // ACM Transactions on Graphics July 1985: 182-222. *InfoLAB*. Web. 23 March 2012
3. «The Science of 3D Rendering» // The Institute for Digital Archaeology. 2019-01-19
4. Ф. Препарата, М. Шеймос. «Вычислительная геометрия: Введение.». — М.: Мир, 1989. Стр. 295.
5. N.G. Bardis, A.P. Markovskyi, N. Doukas, N. V. Karadimas. «True Random Number Generation Based on Environmental Noise Measurements for Military Applications». // Proceedings of the 8th WSEAS International Conference on SIGNAL PROCESSING, ROBOTICS and AUTOMATION. — 2009. — С. 68—73.
6. Huss, S.A. «Advances in Design and Specification Languages for Embedded Systems: Selected Contributions from FDL’06». — Springer, 2007. — P. 345. — 368 p.
7. WORLD CREATOR. [Электронный ресурс] URL: https://www.world-creator.com/ (дата обращения: 15.05.2021)
8. INSTANT TERRA 2.2. [Электронный ресурс] URL: https://www.wysilab.com/ (дата обращения: 15.05.2021)
9. World Machine. [Электронный ресурс] URL: https://www.world-machine.com/ (дата обращения 15.05.2021)
10. 3D MAP GENERATOR. [Электронный ресурс] URL: https://www.3d-map-generator.com/ (дата обращения 15.05.2021)
11. Себеста Р.У. «Основные концепции языков программирования» = «Concepts of programming languages». — 5-е изд. — М.: Вильямс, 2001.

12. David Ebert, Kent Musgrave, Darwyn Peachey, Ken Perlin, and Worley. «Texturing and Modeling: A Procedural Approach». // Academic Press, October 1994.
13. Williams, James. «Learning HTML5 game programming : a hands-on guide to building online games using Canvas, SVG, and WebGL» // Upper Saddle River, NJ: Addison-Wesley, 2012. — P. 117—120, 123—131, 136, 140—142.
14. Бэнкс Алекс, Порселло Ева. React и Redux: функциональная веб-разработка. — СПб.: «Питер», 2018. — С. 336
15. Итан Браун. «Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript» = «Web Development with Node and Express» // Итан Браун. — Санкт-Петербург: Питер, 2017. — 336 с.
16. Миркин Б. М., Наумова Л. Г. «Краткий курс общей экологии. Часть II. Экология экосистем и биосферы: Учебник». — Уфа: Изд-во БГПУ, 2011. — 180 с.
17. Shalamov Viacheslav, Valeria Efimova, Sergey Muravyov, and Andrey Filchenkov. «Reinforcement-based Method for Simultaneous Clustering Algorithm Selection and its Hyperparameters Optimization». Procedia Computer Science 136 (2018): 144-153.
18. Определение дерева // Лекции по теории графов / Емеличев В. А., Мельников О. И., Сарванов В. И., Тышкевич Р. И.. — М.: Наука, Физматлит, 1990. — С. 53. — 384 с.
19. Слюсар В. И. Методы передачи изображений сверхвысокой четкости. //Первая миля. Last mile. — 2019, № 2. — С. 46 — 61
20. Храмов, Ю. А. Рэлей (Стретт) Джон Уильям (Rayleigh (Strutt) John William) // Физики : Биографический справочник / Под ред. А. И. Ахиезера. — Изд. 2-е, испр. и доп. — М. : Наука, 1983. — С. 239. — 400 с.
21. Chan, F.K.; O'Neill, E. M. (1975). Feasibility Study of a Quadrilateralized Spherical Cube Earth Data Base (CSC - Computer Sciences Corporation, EPRF Technical Report 2-75). // Monterey, California: Environmental Prediction Research Facility.

22. Брюнелли Б. Е., Намгаладзе А. А. Физика ионосферы. М.: Наука, 1988. § 3.5, С. 172—173
23. Лурье А. И. Аналитическая механика. — М.:Физматлит. — 1961. — 824 с
24. Шрейдер Ю. А. Что такое расстояние? // «Популярные лекции по математике». — М.: Физматгиз, 1963 г. — Выпуск 38. — 76 с.
25. Paas F. et al. Cognitive load measurement as a means to advance cognitive load theory //Educational psychologist. — 2003. — Т. 38. — №. 1. — С. 63-71.
26. Федеральный Закон «Об оценочной деятельности в Российской Федерации» от 29 июля 1998 года № 135 — ФЗ, «Российская газета» № 148 — ФЗ от 06.08.1998 года.
27. Гражданский кодекс Российской Федерации: Часть первая – четвертая: [Принят Гос. Думой 23 апреля 1994 года, с изменениями и дополнениями по состоянию на 10 апреля 2009 г.] // Собрание законодательства РФ. – 1994. – № 22. Ст. 12, ст. 1252.
28. Кодекс Российской Федерации об административных правонарушениях от 30 декабря 2001 г. № 195-ФЗ. Ст. 7.12, п. 2
29. Уголовный кодекс Российской Федерации от 13 июня 1996 г. № 63-ФЗ. Ст. 147.