Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

«Генерация рельефа местности с применением карты высот»

Руководитель курсового проекта _____ Н.Ю.Рязанова
(Подпись, дата) (И.О.Фамилия)

Студент _____ А.О.Найденышев
(Подпись, дата) (И.О.Фамилия)

Москва 2020 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ7
(Индекс)
_____ И.В.Рудаков
(И.О.Фамилия)
« ___ » _____ 2020 г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине_____Компьютерная графика_____

_____Генерация рельефа местности с применением карты высот_____
(Тема курсового проекта)

Студент_____Найденышев А.О._____гр. ИУ7-55Б_____
(Фамилия, инициалы, индекс группы)

График выполнения проекта: 25% к _4 нед., 50% к _7_ нед., 75% к _11нед., 100% к 14 нед.

1. Техническое задание Реализовать программный комплекс для построения трёхмерного изображения ландшафта местности, используя карту высот. Должна быть обеспечена возможность загрузки пользовательской карты высот и ее генерации для демонстрации. Разработать интерфейс, позволяющий пользователю перемещать и поворачивать ландшафт. Предоставить возможность пользователю результирующее изображение сохранять в формате BMP. Источник света должен находиться на бесконечно удаленном расстоянии. Размер загружаемой пользователем карты не должен превышать 300x300 пикселей, формат изображения - BMP.

2. Оформление курсового проекта

2.1. Расчетно-пояснительная записка на 25-30 листах формата А4.

Расчетно-пояснительная записка должна содержать постановку введение, аналитическую часть, конструкторскую часть, технологическую часть, экспериментально-исследовательский раздел, заключение, список литературы, приложения.

2.2. Перечень графического материала (плакаты, схемы, чертежи и т.п.)__На защиту проекта должна быть представлена презентация, состоящая из 15-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, диаграмма классов, интерфейс, характеристики разработанного ПО, результаты проведенных исследований.

Дата выдачи задания « ____ » _____ 20__ г.

Руководитель курсового проекта _____ Н.Ю.Рязанова
(Подпись, дата) (И.О.Фамилия)

Студент _____ А.О.Найденышев
(Подпись, дата) (И.О.Фамилия)

Оглавление

Введение

Все большую роль в современной жизни играют компьютеры. После того как они вышли за пределы военных и научно-исследовательских организаций, они приобрели статус неотъемлемой части практически любого современного вида деятельности, будь то производство машин или операции с ценными бумагами, проектирование зданий или транспортировка грузов, обучение или индустрия развлечений. Компьютеры используются везде.

И для каждой области необходимы свои собственные приложения, адаптированные для выполнения конкретных функций и задач. Благодаря этому в настоящее время постоянно появляются новые предметы изучения и исследования. Так, например, достаточно новым проявлением таких тенденций выступает машинная графика. Это наука, занимающаяся построением графических изображений посредством вычислительных систем. Появление машинной графики было вызвано многими важными задачами, среди которых присутствуют такие, как визуализация результатов, полученных при обработке данных, моделирование реальных процессов и др. Одной из важнейших областей, вызвавших формирование машинной графики как науки, послужила военная область. Для обучения пилотов самолетов, водителей танков и другой техники, подготовки к действиям в боевых условиях было необходимо создавать симуляторы реальных технических средств. Было гораздо безопасней посадить человека за симулятор для получения первоначальных навыков, чем на реальный объект. А для создания симуляторов потребовалось получать реалистичные изображения различных объектов, например местности, на

которой ведутся учения, причем эти изображения должны быть получены в реальном времени, то есть так, чтобы в зависимости от действий обучаемого соответственно менялись и параметры системы, такие как положение на местности, высота и др. Через несколько лет стало понятно, что такие системы можно использовать не только для обучения новичков, но и для планирования реальных боевых операций на любой территории, о которой есть определенные данные.

В ответ на эти запросы сначала появилась такая область машинной графики, как «трехмерное моделирование»(3D-моделирование), а затем, некоторое время спустя, ветвь 3D-моделирования, которая занималась проектированием и созданием трехмерных реалистичных изображений ландшафтов.

1 Аналитическая часть

1.1 Обзор и анализ существующего программного обеспечения и обоснование необходимости разработки

На сегодняшний день существуют множество программ, генерирующих 3D ландшафт. К более известным можно отнести следующие программы:

- Vue 5 Espirit;
- VistaPro.

1.1.1 Vue 5 Espirit

Vue 5 Esprit позволяет создавать как фотореалистичные, так и великолепные фантастические ландшафты с горами и водной гладью, великолепными атмосферными эффектами и разнообразной растительностью.
Недостатки:

- очень плохая поддержка импорта и экспорта;
- слишком много ненужных возможностей.

Преимущества:

- более привычный интерфейс;

- небольшой объем.

На Рисунке 1 показан внутренний интерфейс программы Vue 5 Esprit

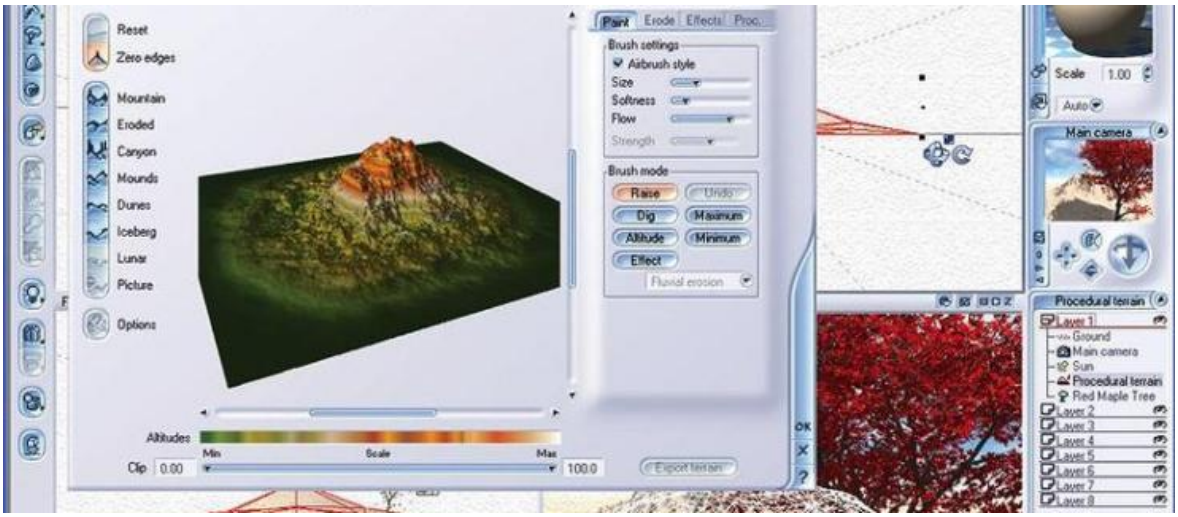


Рисунок 1 – Интерфейс программы Vue 5 Esprit.

1.1.2 VistaPro

VistaPro имеет самую длительную историю развития и когда-то работал еще под MS-DOS, позволяет строить как фотореалистичные пейзажи, полностью соответствующие существующим в действительности в том или ином уголке планеты, так и совершенно фантастические ландшафты, возникающие в воображении художника.

Недостатки:

- ограниченность баз данных объектов;
- малое количество настроек для регулирования небесных, атмосферных и прочих ландшафтных параметров.

Преимущества:

- невысокие требования к аппаратным ресурсам;
- поддержка баз данных ГИС;
- небольшая цена.

На Рисунке 2 показан внутренний интерфейс программы VistaPro.

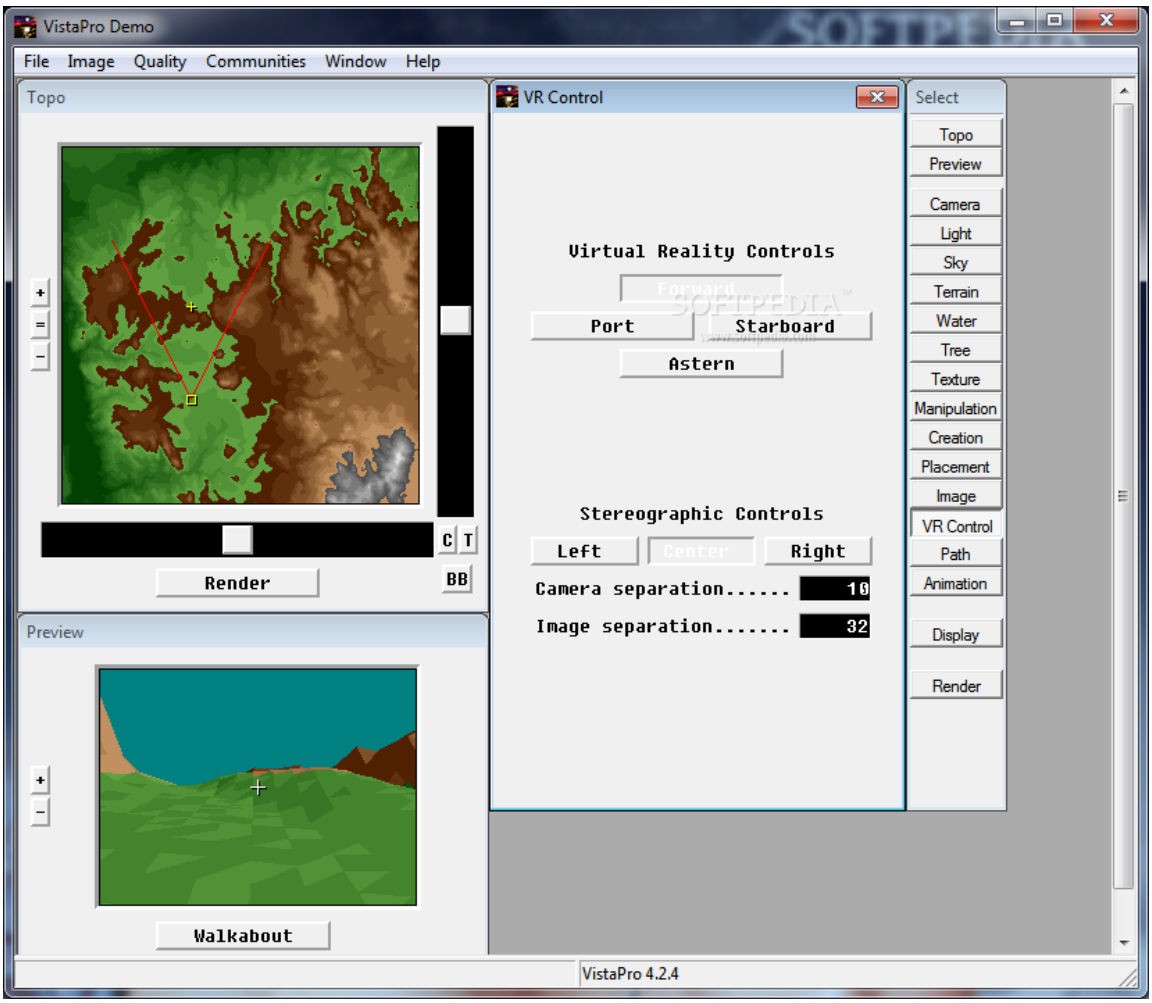


Рисунок 2 – Интерфейс программы VistaPro.

1.1.3 Вывод

Не смотря на то, что существует более чем достаточное количество программного обеспечения для генерации ландшафта, характерными минусами для всего программного обеспечения является высокие требования к аппаратным ресурсам и большой объем пакетов. В своей программе я устраню эти недостатки, предоставив пользователю только вывод результата генерации и интерфейс для работы с изображением, тем самым снизив ресурсные требования к ПО.

1.2 Представление данных о ландшафте

Существует несколько основных принципов представления данных для хранения информации о ландшафтах:

1. использование регулярной сетки высот (или еще другое название Карта Высот - HeightMap);
2. использование иррегулярной сетки вершин и связей, их соединяющих (т.е. хранение простой триангулизированной карты);
3. хранение карты ландшафта, но в данном случае хранятся не конкретные высоты, а информация об использованном блоке. В этом случае создается некоторое количество заранее построенных сегментов, а на карте указываются только индексы этих сегментов.

В этом проекте был выбран первый способ представления ландшафтов

Height Map: Данные представлены в виде двухмерного массива. Уже заданы две координаты (x, y - по высоте и ширине массива), и третья координата задается значением в конкретной ячейке, это высота.

Плюсы данного подхода:

1. простота реализации: легкость нахождения координат (и высоты) на карте, простая генерация ландшафта по карте высот или методом шума Перлина;
2. наглядность: в любой программе просмотра графических файлов можно сразу увидеть или изменить всю информацию;

Минусом данного подхода является большое количество избыточных данных (особенно для поверхностей, близких к плоским).

На Рисунке 3 показана визуализация данного подхода.

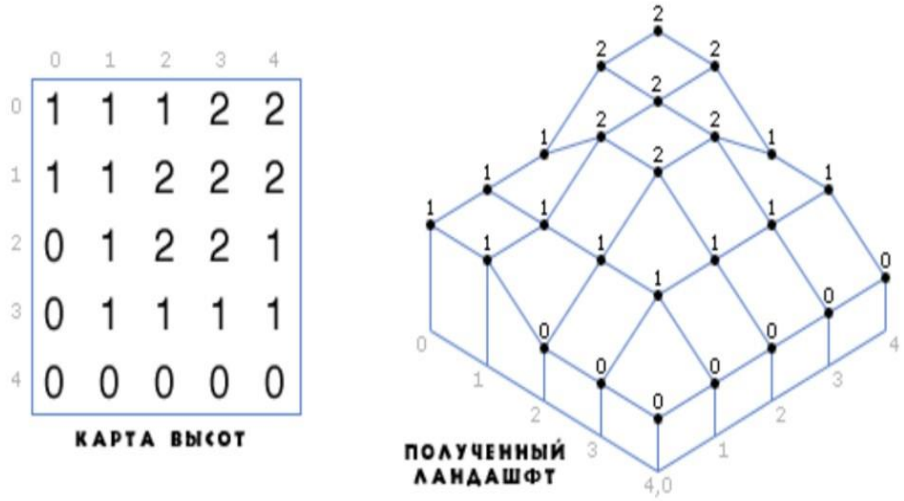


Рисунок 3 – Графическое представление метода Height Map.

Иррегулярная сетка вершин и связей: Еще один способ представления данных для ландшафтов про которую стоит упомянуть — иррегулярная сетка вершин и связей их соединяющих. Зачастую такие решения применяются в специализированных пакетах для игр. И хранятся в виде трехмерных моделей.

Это дает основной выигрыш по сравнению с картами высот:

1. Используется значительно меньше информации для построения ландшафта. Здесь необходимо хранить только значения высот каждой вершины и связи эти вершины соединяющие. Это дает выигрыш в скорости при передаче огромных массивов информации, в процессе визуализации ландшафта.

Но также имеются множество недостатков:

1. алгоритмы построения ландшафтов в основном предназначены для регулярных карт высот. Оптимизация таких алгоритмов под этот способ потребует значительных усилийа;
2. сложности при динамическом освещении — вершины расположены достаточно далеко друг от друга и неравномерно;
3. хранение, просмотр, модификация такого ландшафта также представляет сложности. При использовании карт высот достаточно применить простые и "стандартные" средства пиксельной графики.

Далее на Рисунке 4 показан метод иррегулярной сетки вершин и связей:

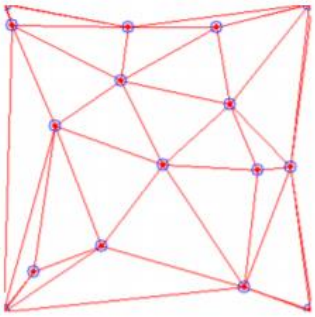


Рисунок 4 – Графическое представление метода иррегулярной сетки вершин и связей

1.2.1 Вывод

В данном разделе были рассмотрены алгоритмы представления данных. Из перечисленных выше алгоритмов представления данных был выбран алгоритм использования регулярной сетки высот, так как является самым популярным и простым в реализации методом.

1.3 Анализ алгоритмов генерации рельефа

Шум Перлина: При разработки данной программы возникает вопрос: откуда брать информацию для генерации карты высот? Конечно же можно просто загружать монохромное изображение и на его основе генерировать ландшафт. Но если нужно каждый раз генерировать разные карты высот (например, для компьютерных игр или демонстрационных программ, таких как эта), то на помощь приходит следующий метод на основе шума Перлина.

Изображение (или какой-либо другой объем) – вне зависимости от диапазона значений его элементов – полностью накрывается сеткой, представляющей диапазон вещественных чисел. Таким образом, создается шум на сетке,

представляющей по всему изображению значения между 0 и 4. Каждое число порождает линию сетки, а значит, все стороны каждого квадрата последней имеют длину, равную одной единице.

Выбранный масштаб влияет на сложность шума. Большое число квадратов на сетке изображения создает более «плотно упакованный» шум, подобный белому шуму на экране плохо настроенного телевизора. Меньшее число квадратов на сетке порождает «клубящийся» шум, внешне похожий на облака. В каждой точке на сетке строится случайный вектор нормали. Это обычный двумерный вектор единичной длины, который указывает в случайном направлении в пределах каждого из квадратов. Традиционный способ создания таких векторов – организация справочной таблицы из 256 векторов, которые охватывают полный круг, и последующий случайный выбор одного из них для каждой точки на сетке. Это гарантирует распределение векторов, которые могут с равной вероятностью указывать в любом направлении. Далее для каждого пикселя изображения находится та из ячеек сетки, где он находится.

Таким образом, определяется значение, которое основано исключительно на данных этой ячейки. Следующий шаг – создать четыре диагональных вектора, соединяющих углы ячейки с текущим пикселем.

Ниже на Рисунке 5 приведено графическое представление данного принципа:

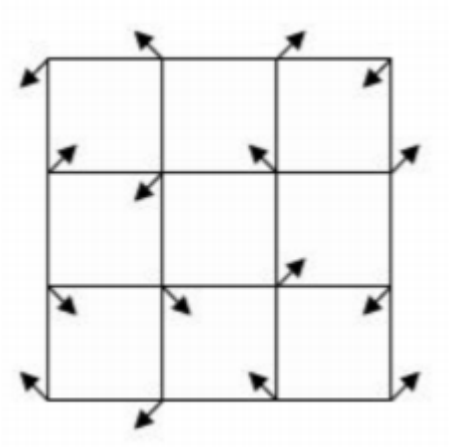


Рисунок 5 – Графическое представление одного из этапов построения шума

Каждый угол ячейки сетки теперь является базой для двух векторов – случайного единичного вектора и вектора в направлении пикселя, который необходимо построить. Для каждой пары таких векторов находиться скалярное произведение. Оно даст скалярное значение высоты каждого из углов сетки. Далее необходимо объединить эти четыре значения и найти высоту пикселя, который надо сгенерировать. Делать это можно по-разному, получая различные результаты, однако чаще всего используется взвешенная интерполяция четырех значений с учетом близости текущей позиции к каждому углу сетки.

Основным плюсом использования шумовой функции при генерации ландшафта является то, что нет необходимости хранить карту высот, а достаточно лишь использовать данные справочной таблицы векторов, – все остальное для восстановления конечной карты высот ландшафта сделает шумовая функция.

Ниже на Рисунке 6 показан шум Перлина:

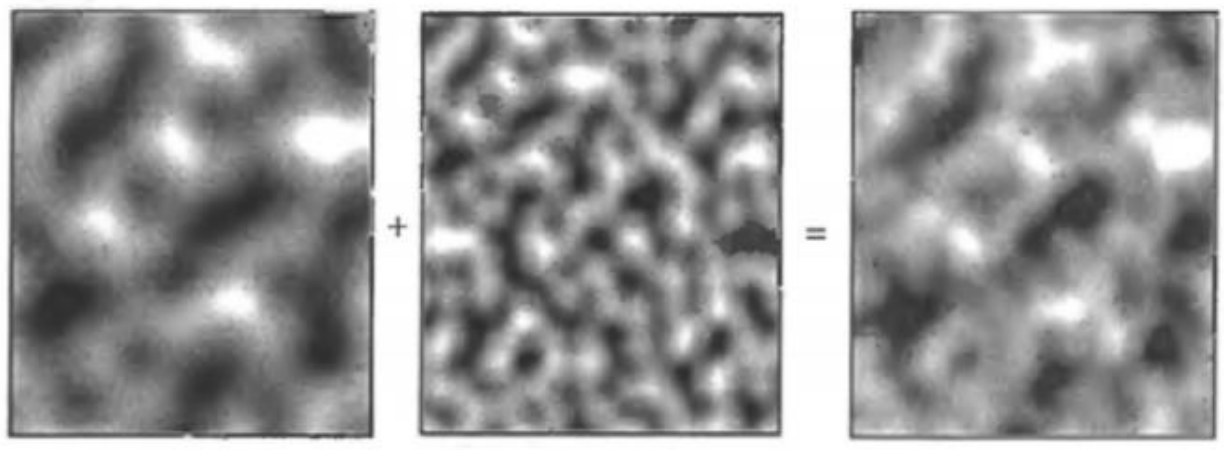


Рисунок 6 – Пример изображения шума Перлина, полученный сложением двух октав

Холмовой алгоритм: Это простой итерационный алгоритм, основанный на нескольких входных параметрах.

Алгоритм изложен в следующих шагах:

1. создание и инициализация двухмерного массива с нулевым уровнем (все ячейки заполнены нулями);
2. выбирается случайная точка и случайный радиус в заранее заданных пределах. Выбор этих пределов влияет на вид ландшафта - либо он будет пологим, либо скалистым;
3. в выбранной точке "поднимается" холм заданного радиуса;
4. возвращение ко второму шагу и так далее до выбранного количества шагов. От него будет зависеть внешний вид ландшафта;
5. нормализация;
6. долинизация.

Первый, второй и четвертые шаги тривиальны, пятый и шестой шаг будет рассмотрен далее. Теперь же изучим третий шаг. Фактически холм в этом случае половина шара, чем больше радиус - тем больше холм (и выше). Математически это похоже на перевернутую параболу:

$$z = r^2 - ((x_2 - x_1)^2 + (y_2 - y_1)^2)$$

здесь (x1, y1) - заданная точка, r - выбранный радиус, (x2, y2) - высота холма.

На Рисунке 7 показан одиночный холм:

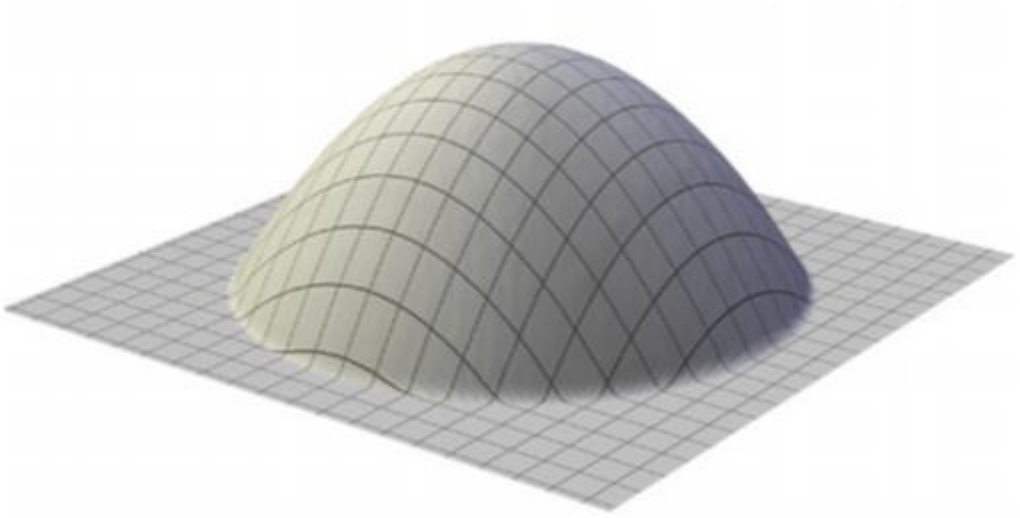


Рисунок 7 – Одиночный холм

Чтобы сгенерировать ландшафт полностью необходимо построить множество таких холмов. Но есть еще две вещи на которые необходимо обратить внимание. Первое - игнорирование отрицательных значений высоты холма. Второе - при генерации последующих холмов лучше добавлять полученное значение для данного холма к уже существующим значениям. Это позволяет строить более правдоподобный ландшафт, нежели правильно очерченные округлые холмы. Ниже на Рисунке 8 представлен ландшафт при большом количестве итераций:

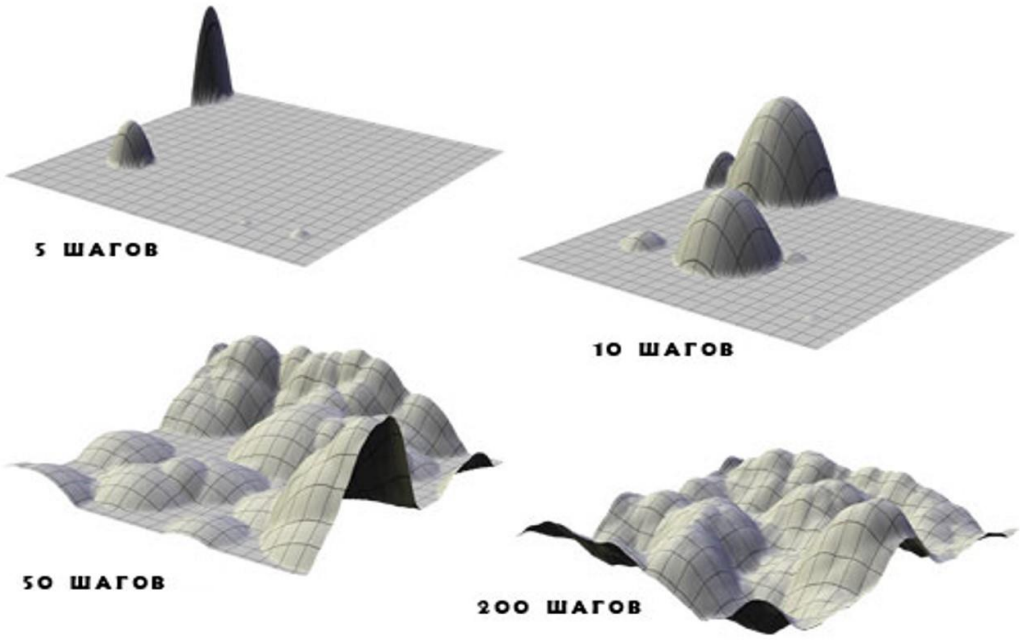


Рисунок 8 – Ландшафт при большом количестве итераций

Нормализация ландшафта

При генерации значений для ландшафта не учитывались выходы этих значений за некоторые пределы (например, если ландшафт будет храниться в монохромной картинке, то необходимо, чтобы все значения находились в пределе от 0 до 256). Для этого необходимо произвести нормализацию значений. Математически нормализация — это процесс получения значений из одного предела, и перевод его в другие пределы.

Вот как это выглядит графически на Рисунке 9:

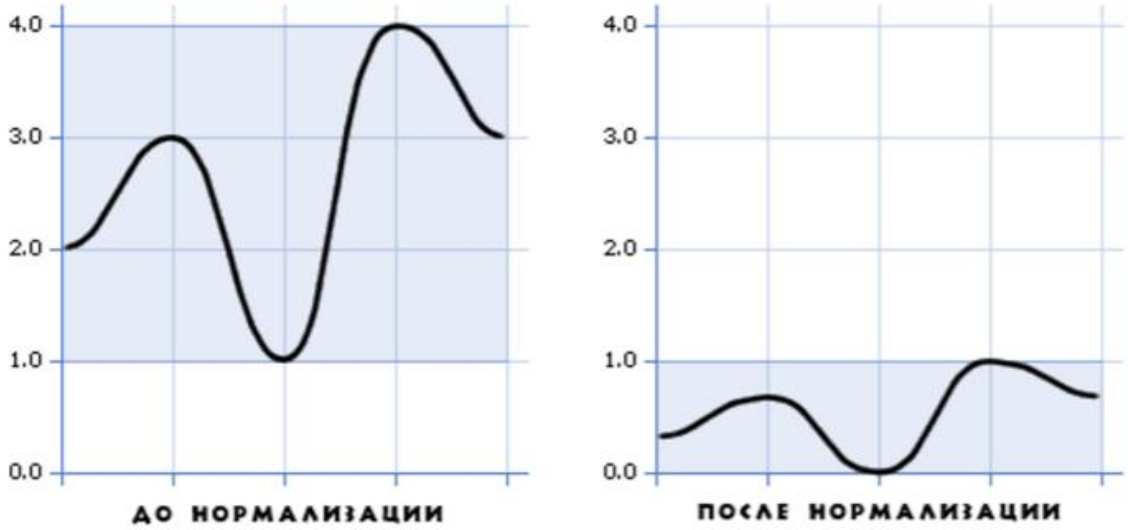


Рисунок 9 – Разница использования «Нормализации»

Чтобы это сделать произведем следующие действия:

1. сперва проходим по всему массиву и запоминаем наибольшее и наименьшее значения;
2. после этого производится нормализация конкретных значений в пределах от 0 до 1. Формула выглядит так:

$$z = \frac{z - min}{\max - min}$$

Долинизация ландшафта

Данный ландшафт уже можно использовать, но если присмотреться, то в нем достаточно мало долин. Склоны холмов излишне крутые, необходимо сделать их более пологими. Идея «Долинизации» состоит в следующем - взять от каждого значения квадратный корень. Это в большей степени влияет на средние значения, практически не затрагивая минимумов и максимумов. Графически это выглядит так как на Рисунке 10:

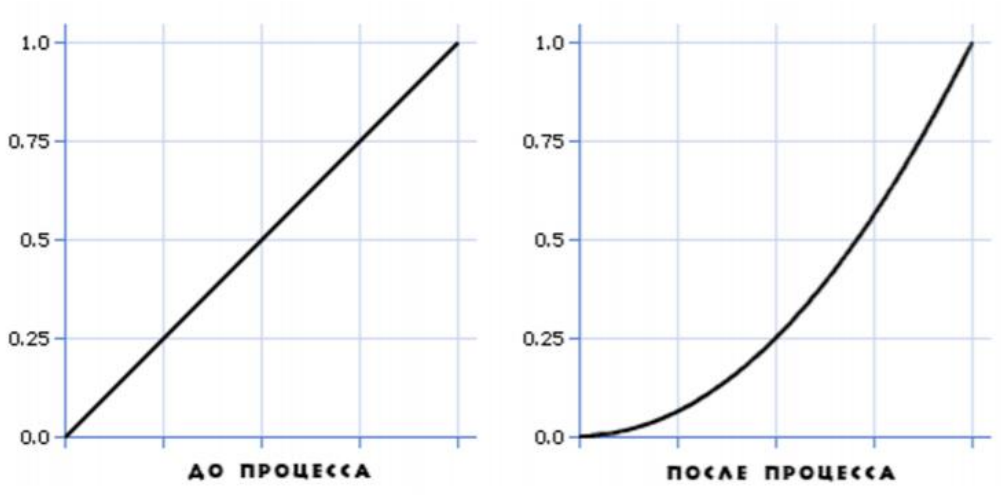


Рисунок 10 – Разница использования «Долинизации»

На Рисунке 11 можно увидеть преобразования ландшафта после «Долинизации» и «Нормализации»:

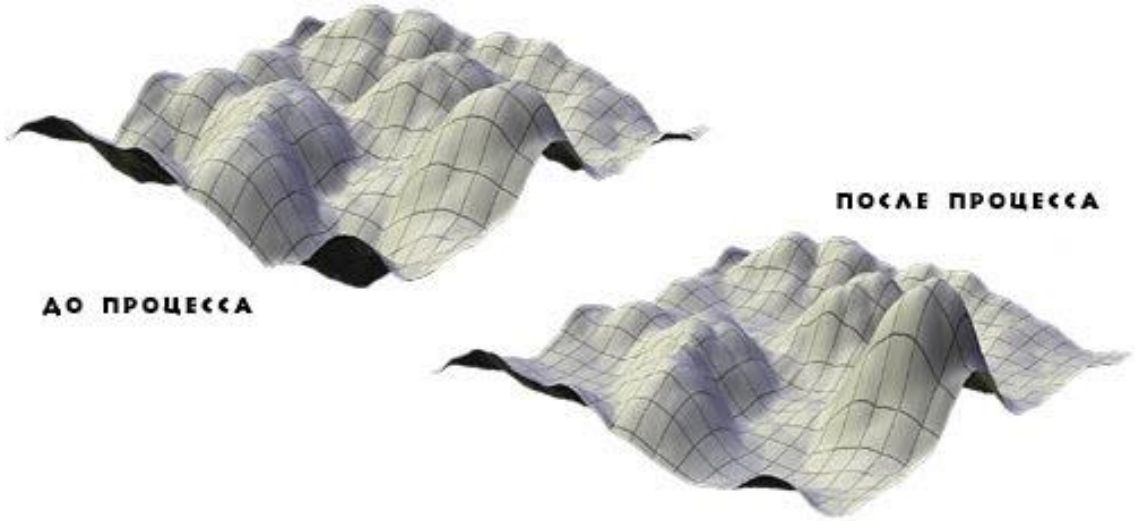


Рисунок 11 – Разница ландшафтов при использования «Долинизации» и «Нормализации»

1.3.1 Вывод

В данном разделе были рассмотрены алгоритмы генерации ландшафта. Оба алгоритма будут реализованы в проекте.

1.4 Алгоритмы визуализации ландшафта и окружающей среды

1.4.1 Алгоритм Z-буфера

Это один из простейших алгоритмов удаления невидимых поверхностей. Работает этот алгоритм в пространстве изображения. Идея z-буфера является простым обобщением идеи о буфере кадра. Буфер кадра используется для запоминания атрибутов (интенсивности) каждого пикселя в пространстве изображения, z-буфер – это отдельный буфер глубины, используемый для запоминания координаты z, или глубины каждого видимого пикселя в пространстве изображения.

Главное преимущество алгоритма – его простота. Кроме того, этот алгоритм решает задачу об удалении невидимых поверхностей и делает тривиальной визуализацию пересечений сложных поверхностей. Сцены могут быть любой сложности. Поскольку габариты пространства изображения фиксированы, оценка вычислительной трудоемкости алгоритма не более чем линейна. Поскольку элементы сцены или картинки можно заносить в буфер кадра или в z-буфер в произвольном порядке, их не нужно предварительно сортировать по приоритету глубины. Поэтому экономится вычислительное время, затрачиваемое на сортировку по глубине. При этом, алгоритм z-буфера, будучи реализованный аппаратно, является самым быстрым алгоритмом удаления невидимых поверхностей.

Основной недостаток алгоритма – большой объем требуемой памяти. Другой недостаток алгоритма z-буфера состоит в трудоемкости и высокой стоимости устранения лестничного эффекта, а также реализации эффектов прозрачности и просвечивания.

При визуализации изображения, как пиксельная информация, так и глубина усредняются. В этом методе требуются очень большие объемы памяти. Например, изображение размером 512x512x24 бита, использующее z-буфер размером 20 бит на пиксель, разрешение которого повышено в 2 раза по осям x и y, и на котором устранена ступенчатость методом равномерного усреднения, требует почти 6 МБ памяти.

1.4.2 Вывод

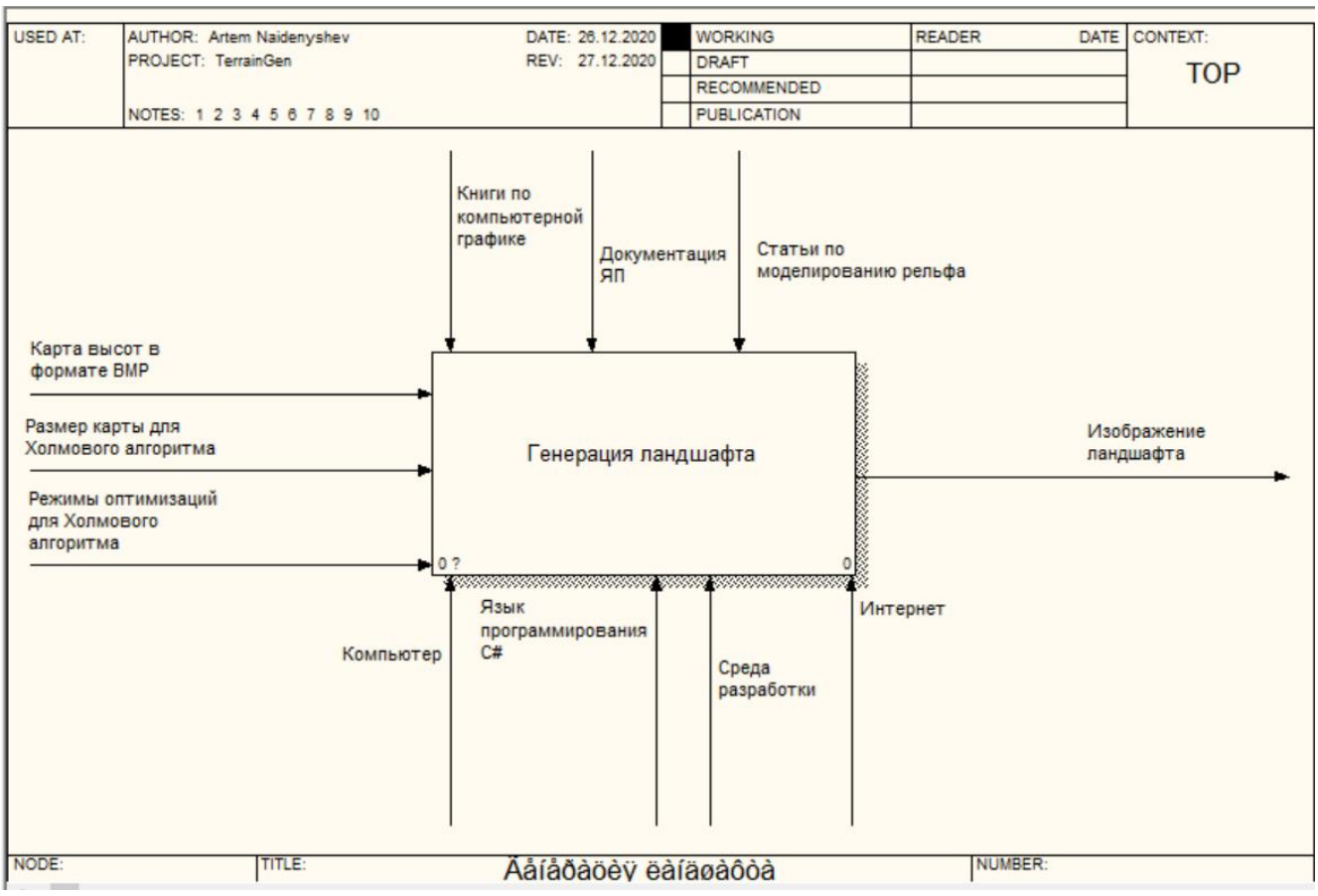
Здесь были рассмотрены алгоритмы визуализации ландшафта и окружающей среды.

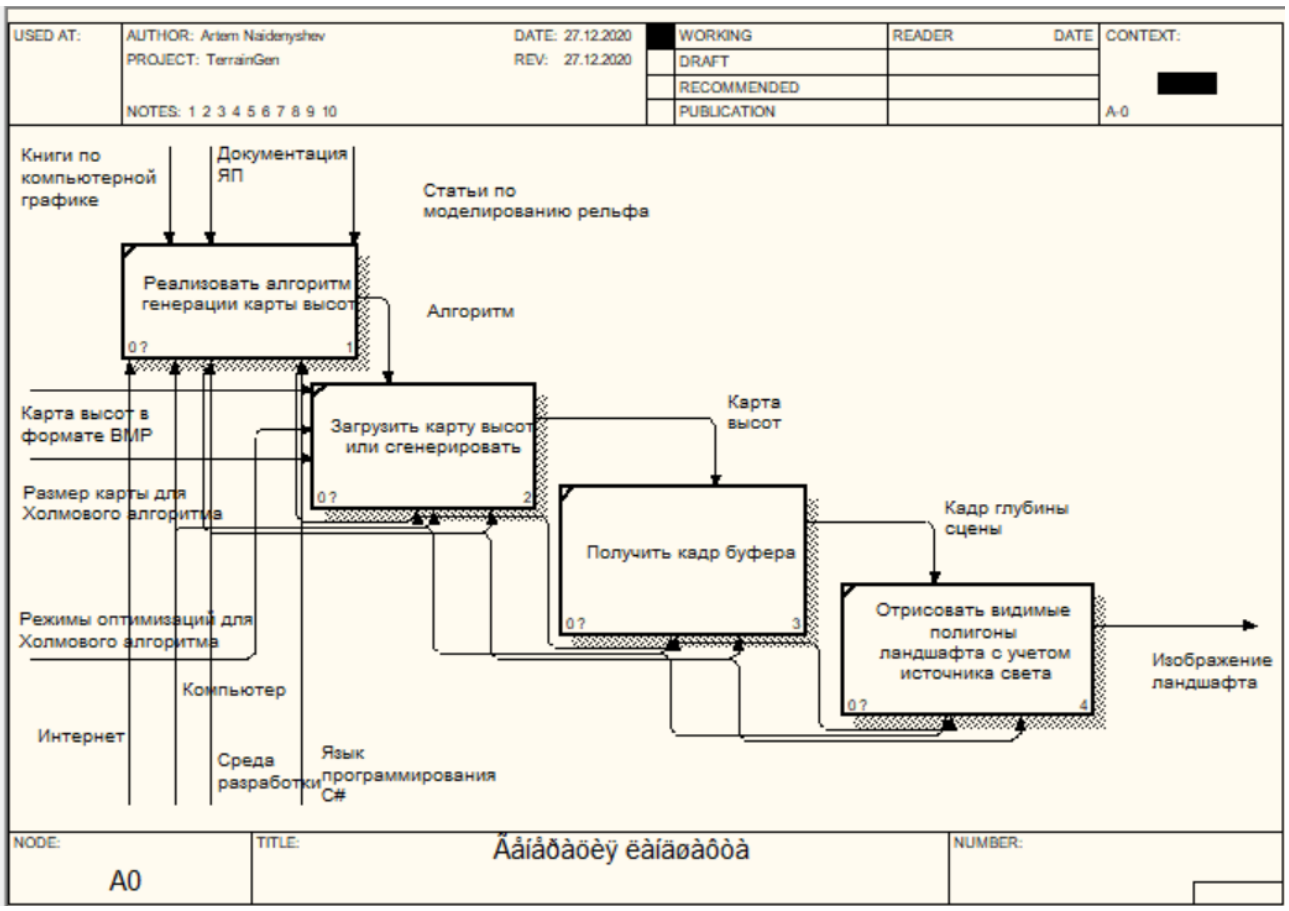
1.5 Выводы из аналитической части

В данном разделе были представлены алгоритмы загрузки данных о ландшафте. Были предоставлены преимущества и недостатки этих методов. Также было подробно описано про порядок визуализации ландшафта и окружающей среды. Подробная схема реализация все этих алгоритмов будет представлена в следующем разделе.

2 Конструкторская часть

2.1 Последовательность преобразований





2.2 Холмовой алгоритм

На Рисунке 12 представлена схема Холмового алгоритма:

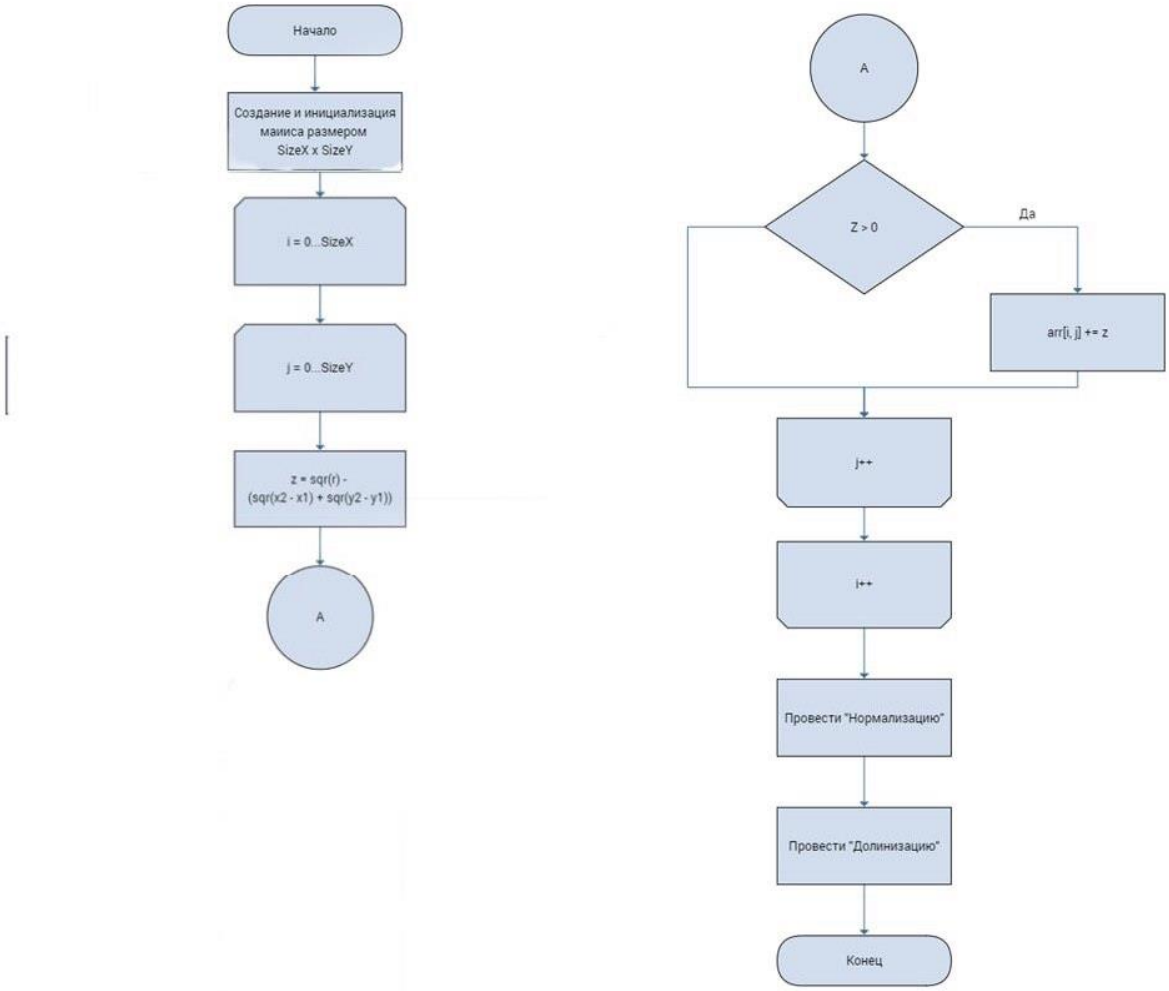


Рисунок 12 – Схема холмового алгоритма

2.3 Алгоритм Z-буффера

На Рисунке 13 представлена схема алгоритма z-буфера:

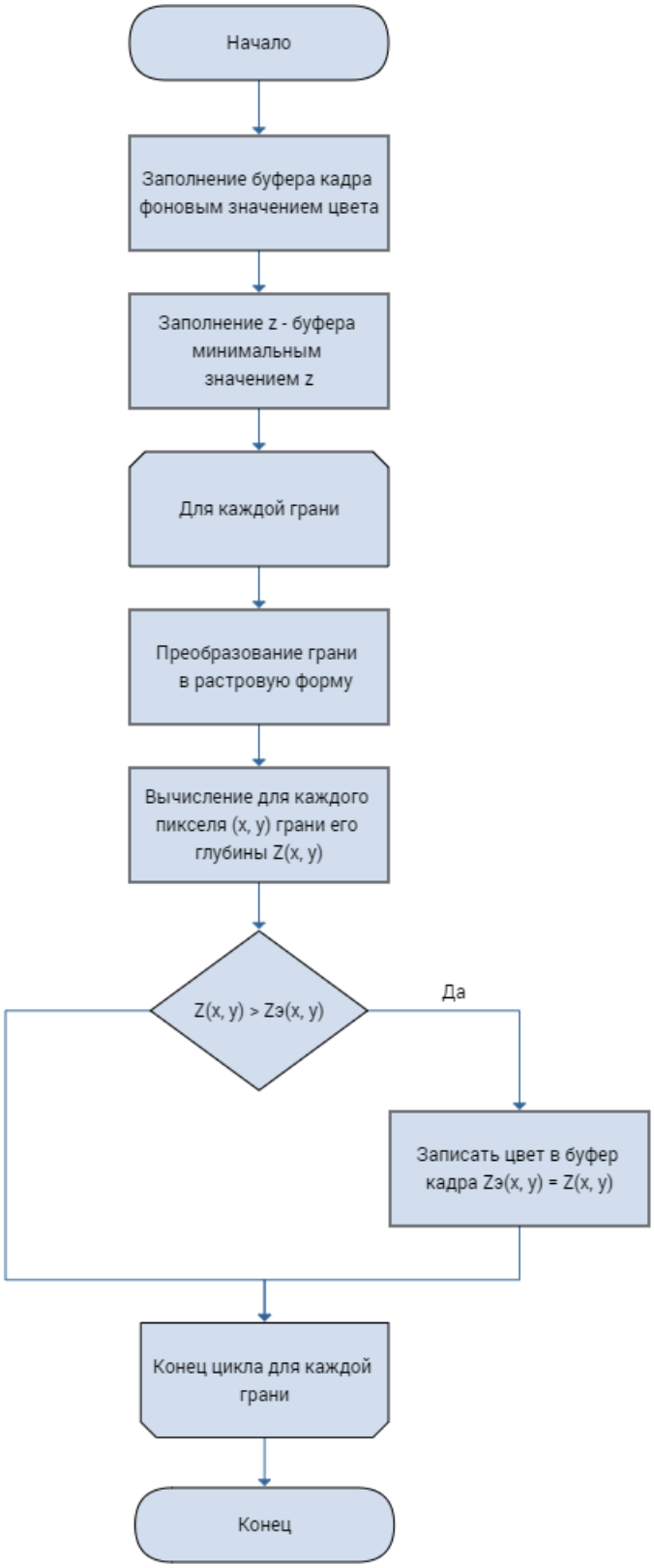


Рисунок 13 – Схема алгоритма Z-буфера

2.4 Освещение

В любом трёхмерном приложении использование какой-либо модели освещения всегда придаёт реалистичность обрабатываемой сцене. Как правило, в неё включается закон, по которому рассчитывается освещённость точки в пространстве, и метод закраски освещённого многоугольника. От выбора той или иной модели освещения зависит качество изображения, построенного компьютером, и скорость работы программы.

Обычно освещённость некоторой точки, принадлежащей грани в пространстве, складывается из рассеянной освещённости и диффузного отражения — потока света, отражающегося от поверхности объекта. Иногда к ним добавляют зеркальное отражение — поток света, отражающийся от внешней поверхности объекта под тем же углом, под которым он падал на эту поверхность.

Диффузное отражение присуще матовым поверхностям. Матовой можно считать такую поверхность, размер шероховатостей которой настолько велик, что падающий луч рассеивается неравномерно во все стороны. Такой тип отражения характерен, например, для гипса, песка, бумаги. Диффузное отражение описывается законом Ламберта, согласно которому интенсивность отраженного света пропорциональна косинусу угла между направлением на точечный источник света и нормалью к поверхности.

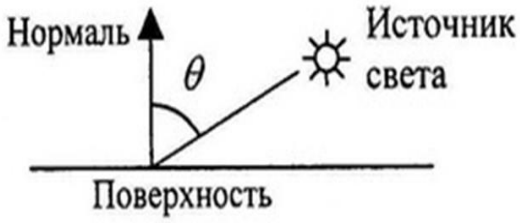


Рисунок 12 – Матовая поверхность

$$I_d = I * K_d * \cos\theta$$

Здесь I_d - интенсивность источника света, K_d - коэффициент, который учитывает свойства материала поверхности. Интенсивность отраженного света не зависит от расположения наблюдателя.

Матовая поверхность имеет свой цвет. Наблюдаемый цвет матовой поверхности определяется комбинацией собственного цвета поверхности и цвета излучения источника света (в данной работе цвет излучения источника считается белым, поэтому учитывается только цвет поверхности).

2.5 Модель освещения Фонга

Модель Фонга состоит из трех главных компонентов: фонового (ambient), рассеянного/диффузного (diffuse) и бликового (specular). Ниже вы можете видеть, что они из себя представляют:

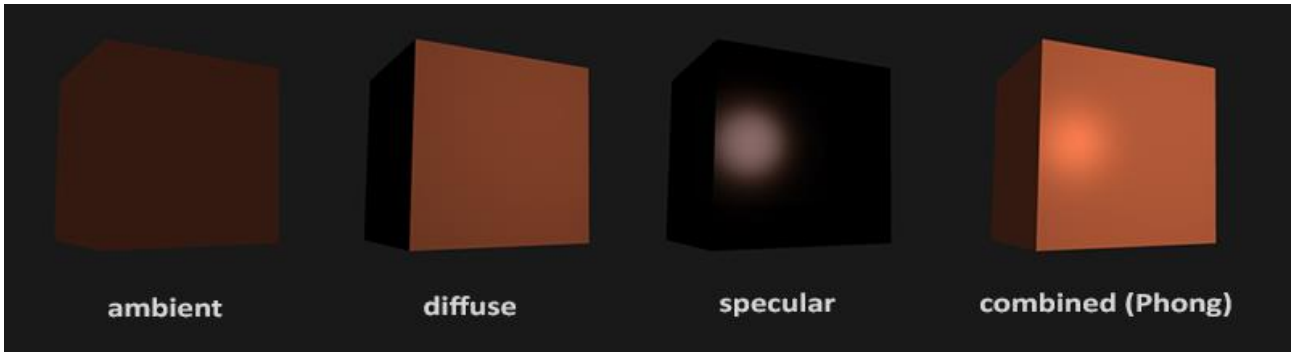


Рисунок 13 – Модель освещения Фонга

- **Фоновое освещение:** даже в самой темной сцене обычно всегда есть хоть какой-нибудь свет (луна, дальний свет), поэтому объекты почти никогда не бывают абсолютно чёрными. Чтобы имитировать это, мы используем константу окружающего освещения, которая всегда будет придавать объекту некоторый оттенок;
- **Диффузное освещение:** имитирует воздействие на объект направленного источника света. Это наиболее визуально значимый компонент модели освещения. Чем большая часть поверхности объекта обращена к источнику света, тем ярче он будет освещен;
- **Освещение зеркальных бликов:** имитирует яркое пятно света (блик), которое появляется на блестящих объектах. По цвету зеркальные блики часто ближе к цвету источника света, чем к цвету объекта.

В данном проекте было решено отказать от просчета бликовой составляющей, так как это ресурсоемко и ландшафту не подходят блики на результирующем изображении.

Последовательность действий для определения интенсивности цвета каждой точки при модели освещения Фонга такова:

1. Вычисляется нормаль к полигону;
2. Скалярное произведение между вектором нормали и вектором света (в данном проекте источник света находится на бесконечно удаленном расстоянии – все вектора света параллельны);
3. И прибавляется фоновая составляющая.

2.6 Вывод из конструкторской части

В данном разделе были представлены схемы используемых алгоритмов, а именно алгоритмы z - буфера и холмовой алгоритм. Также было подробно описана модель освещения, применяемая в данном проекте.

3 Технологическая часть

3.1 Выбор языка и среды программирования

В данной работе используется язык программирования C#. С помощью C# можно быстро писать программы под Windows, благодаря удобному синтаксису, интеллектуальным подсказкам и управляемому коду. Используя управляемый код, не нужно заботится о сборке мусора, об указателях и о некоторых базовых структурах и алгоритмах – все это уже реализовано. Главным минусом разработки на C# является низкая производительность программ. Но это уже компенсируется современным аппаратным обеспечением.

В качестве среды разработки была выбрана IDE Visual Studio 2019. Эта среда разработки может сильно оптимизировать код, используя раскрытие

циклов, встраивание функций, использование суперскалярных команд процессора SSE и другие, тем самым увеличив производительность.

В качестве технологического подхода к разработке проекта был выбран объектно-ориентированный подход. Этот выбор обуславливается легкостью и быстротой разработки программы

3.2 Структура программы

GenerateHeightmap – модуль, в котором реализованы алгоритмы генерации и чтения из файла карты высот. Список функций, реализующие алгоритмы:

1. static public double[][] Hills(int SizeX, int SizeY, bool Smoothing, bool Valley, int countHills) – функция генерации карты высот методом Холмового алгоритма
    - SizeX, SizeY – размер генерируемой карты
    - Smoothing – оптимизация алгоритма. «Смягчение» ландшафта;
    - Valley – оптимизация алгоритма. «Долинизация» ландшафта;
    - countHills – количество холмов на карте.
2. private void LoadFromFile(Bitmap bmp, Color[] colorPix) – функция загрузки карты высот
    - bmp – переменная, хранящая в себе изображение загружаемой карты высот;
    - colorPix – массив цветов для каждого полигона ландшафта.
3. private void GenPerlin(Bitmap bmp, Color[] colorPix) – функция генерации карты высот методом шума Перлина.
    - bmp – переменная, хранящая в себе изображение загружаемой карты высот;
    - colorPix – массив цветов для каждого полигона ландшафта.

Transformation – модуль для преобразований ландшафта в пространстве. Список функций, реализующие преобразования ландшафта на сцене:

1. public ParallMove(ref Vector3[] arrayPoints, float dx, float dy, float dz, int to, int from) – перемещение ландшафта на заданную величину;
    - arrayPoints – массив точек ландшафта;
    - dx, dy, dz – на сколько переместить ландшафт по осям;
    - to, from – переменные для реализации распараллеливания преобразования.
2. public ParallRotate(ref Vector3[] arrayPoints, double angle, float xCen, float yCen, int to, int from) – поворот ландшафта;
    - arrayPoints – массив точек ландшафта;
    - angle – угол поворота;
    - xCen, yCen – координаты центра, относительного которого происходит поворот;
    - to, from - переменные для реализации распараллеливания преобразования.
3. public ParallScale(ref Vector3[] arrayPoints, float kx, float ky, float kz, float xCen, float yCen, int to, int from) – масштабирование ландшафта.
    - arrayPoints – массив точек ландшафта;
    - kx, kx, kz – коэффициенты масштабирования;
    - xCen, yCen – координаты центра масштабирования;
    - to, from - переменные для реализации распараллеливания преобразования.

Rander – модуль отрисовки ландшафта в 3D пространстве. Список функций, реализующие отрисовку ландшафта:

1. static public float[][] GenBuffer(int width, int height, ref Bitmap landscape, int[][] faces, Vector3[] wordHeightmap, Vector3 lightDir, Color[] colorPix) – функция, реализующая алгоритм Z-буффер;
    - width – ширина сцены
    - height – высота сцены
    - landscape – переменная, хранящая в себе изображение сцены

- faces – массив, хранящий в себе связи между точками ландшафта;
- wordHeightmap – массив точек ланшафта;
- lightDir – направление веткора света
- colorPix – массив, хранящий в себе информацию о цвете каждого полигона ландашфта.

3.3 Входные и выходные данные

Входными данными для данной программы является карта высот в формате ".bmp", не превышающая 300x300 пикселей. Так же при генерации ландшафта Холмовым алгоритмом принимаются данные с клавиатуры и мышки, которые интерпретируются как команды пользователя, и в зависимости от них строится последующее изображение.

Выходными данными программы является анимационный ряд с изображением трехмерного ландшафта, построенного и отображенного на основании входных данных.

3.4 Описание интерфейса программы

Ниже представлен интерфейс полученного программного обеспечения

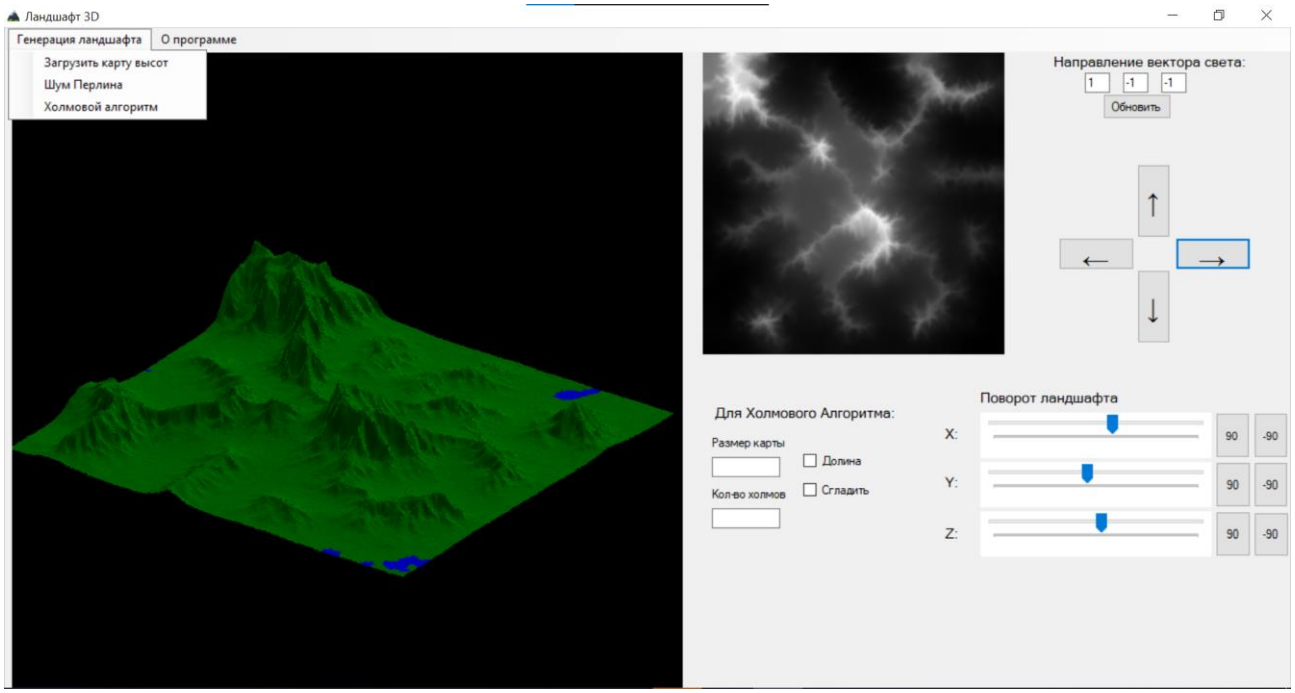


Рисунок 13 – Интерфейс программы

В верхнем меню пользователь может выбрать сгенерировать ему ландшафт одним из алгоритмов или загрузить карту высот. При загрузке карты высот и при генерации ландшафта шумом Перлина, справа отобразится карта высот.

Пользователь может менять направление вектора света, указав составляющие компоненты под строкой «Направление вектора света» и нажав кнопку «Обновить».

Четыре кнопки ниже позволяют перемещать ландшафт, стрелочки указывают в какую сторону произойдет перемещение. Шаг перемещения по умолчанию удобен для детального просмотра результата.

Масштабирование происходит колесиком мыши при наведение его на сцену, где генерируется ландшафт.

Далее располагается три трекбара, с помощью которых можно поворачивать ландшафт вокруг соответствующих осей, для удобства добавлены кнопки поворота на 90/-90 градусов.

И остается блок виджетов для Холмового алгоритма.
Можно задать размер генерируемой карты числом (карта генерируется квадратной), количество холмов, а также отметить нужные оптимизации для алгоритма.

Получившееся изображение можно сохранить на своем ПК, дважды нажав на сцену. Появится диалоговое окно:

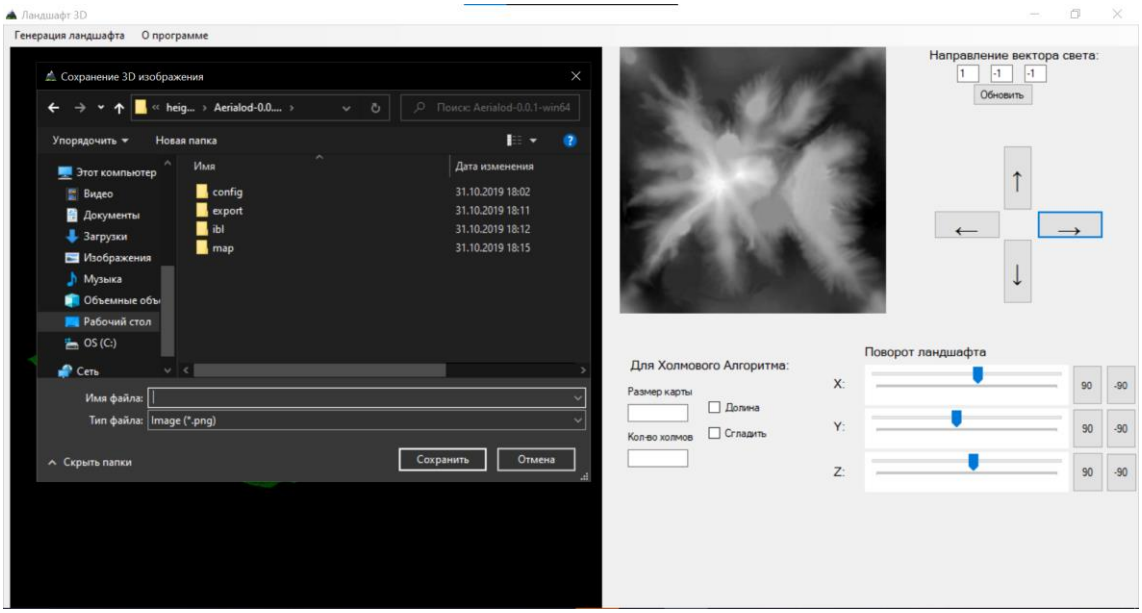


Рисунок 14 – Диалоговое окно

3.5 Вывод из технологической части

В данном разделе было описано и обосновано выбор языка и средств реализации данного проекта. Также была описана структура полученного ПО. Был продемонстрирован внутренний интерфейс программы.

4 Исследовательская часть

4.1 Системные характеристики

Характеристики компьютера на котором проводился эксперимент:

1. Операционная система – Windows 10;
2. Процессор - - Intel(R) Core(TM) i5-8250U CPU @1.80GHz;
3. Объем оперативной памяти – 8 Гб;
4. Количество ядер – 4;
5. Количество логических процессов – 8;

4.2 Постановка эксперимента

В рамках данного проекта были проведены эксперименты, описанные ниже:

1. сравнение и анализ времени отрисовки ландшафта разного размера алгоритмом Z-буффер;

2. сравнение и анализ времени генерации ландшафта двумя способами.

4.3 Сравнительный анализ на основе замеров времени работы программы

На рисунке 15 показаны результаты первого эксперимента, суть которого заключается в анализе времени построения ландшафта z – буффером от размера карты. Для данного эксперимента было выбрано 4 размера карты:

1. 128x128;
2. 256x256;
3. 512x512;
4. 1024x1024.

Ниже приведена полученная диаграмма:

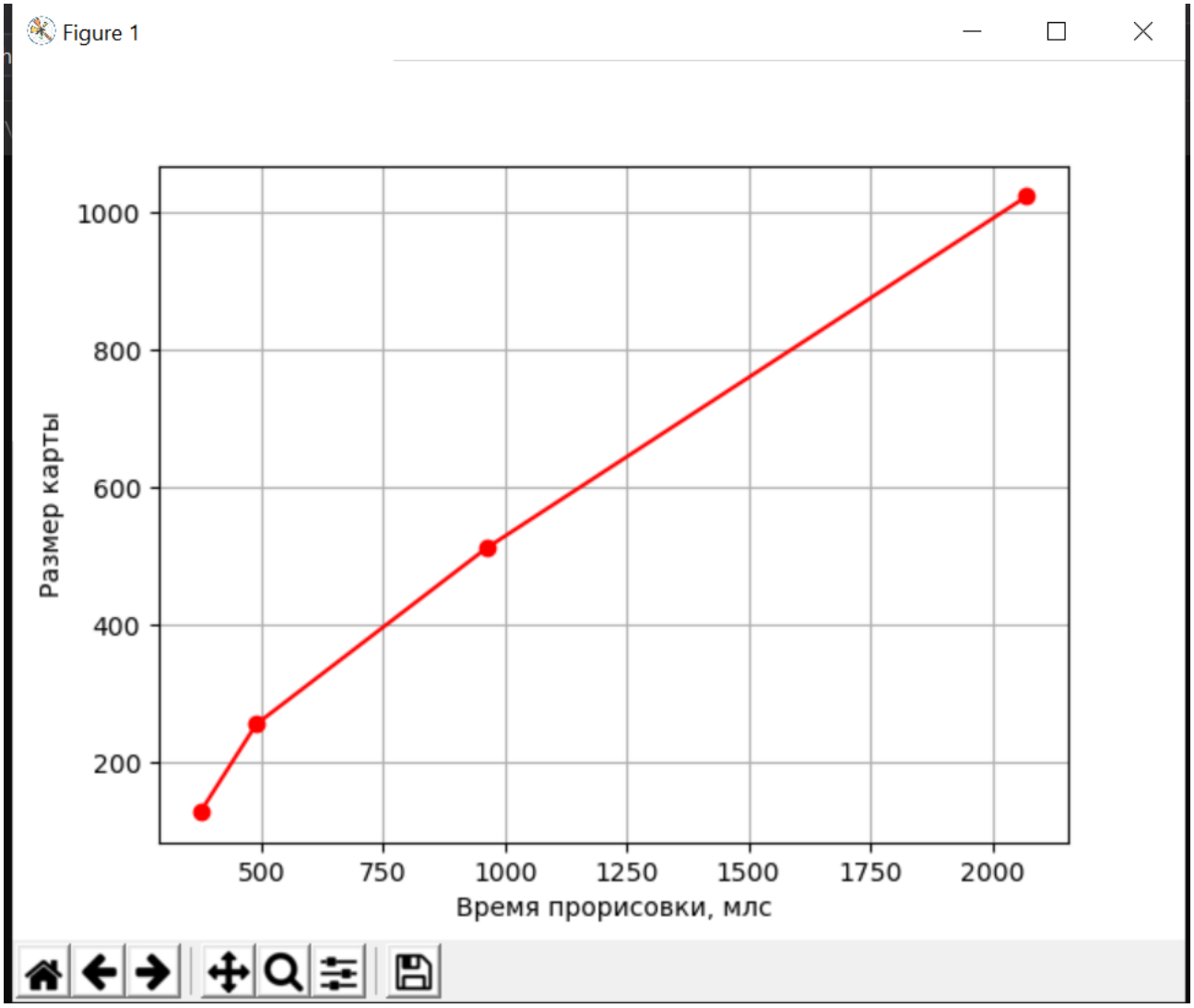


Рисунок 15

Из данного графика отчетливо видно, что скорость прорисовки ландшафта прямо пропорционально от размера самого ландшафта. Это соответствует и теоретическим расчетам.

На рисунке 16 показаны результаты второго опыта. Красным цветом построен график Холмового алгоритма, зеленым – шум Перлина. Для эксперимента размеры карт выбирались, как и для предыдущего.

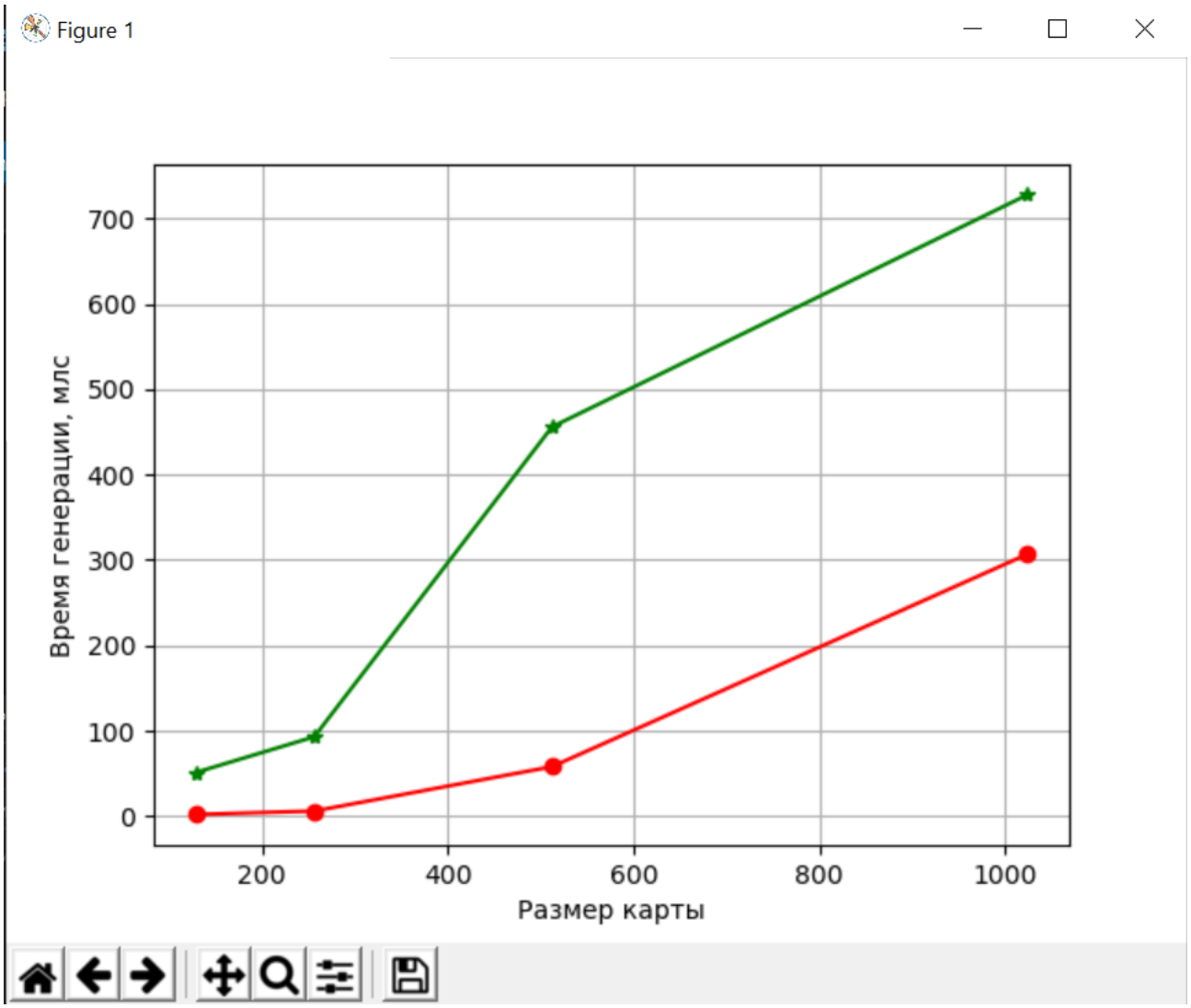
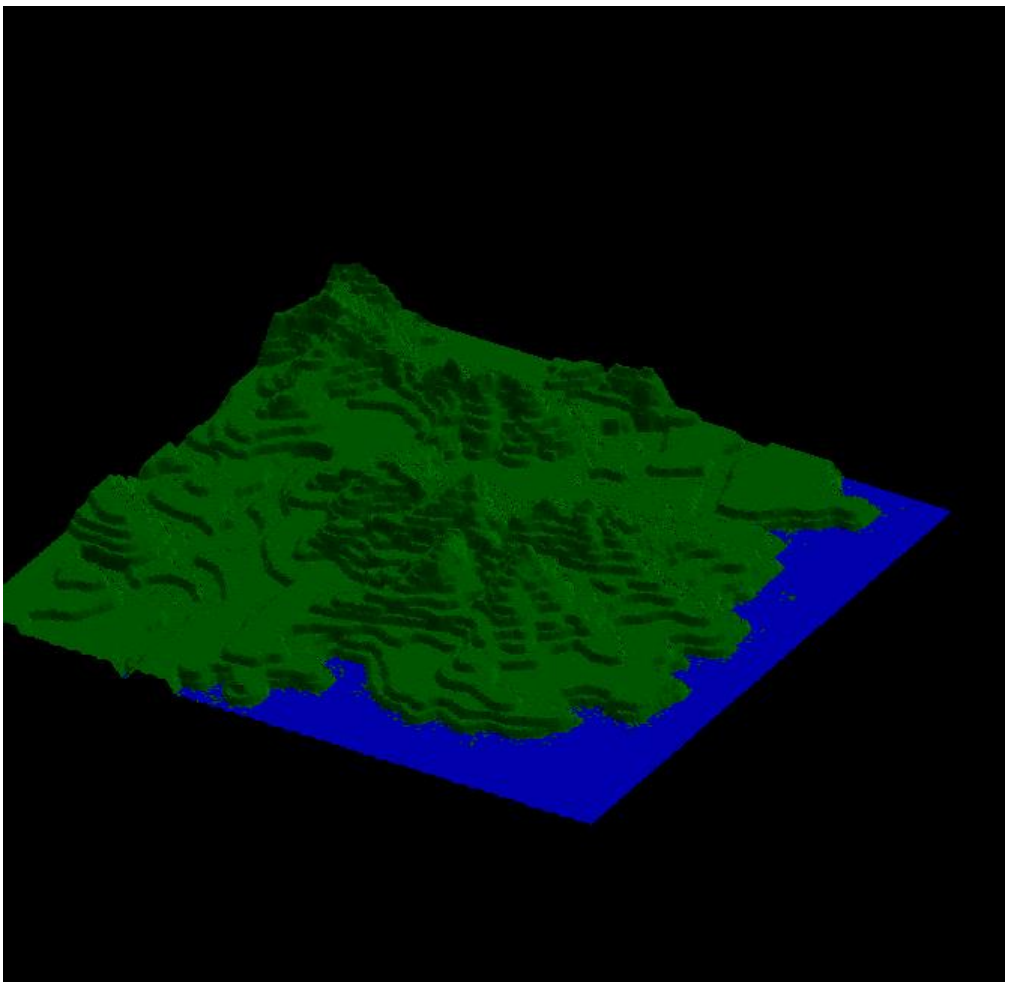


Рисунок 16

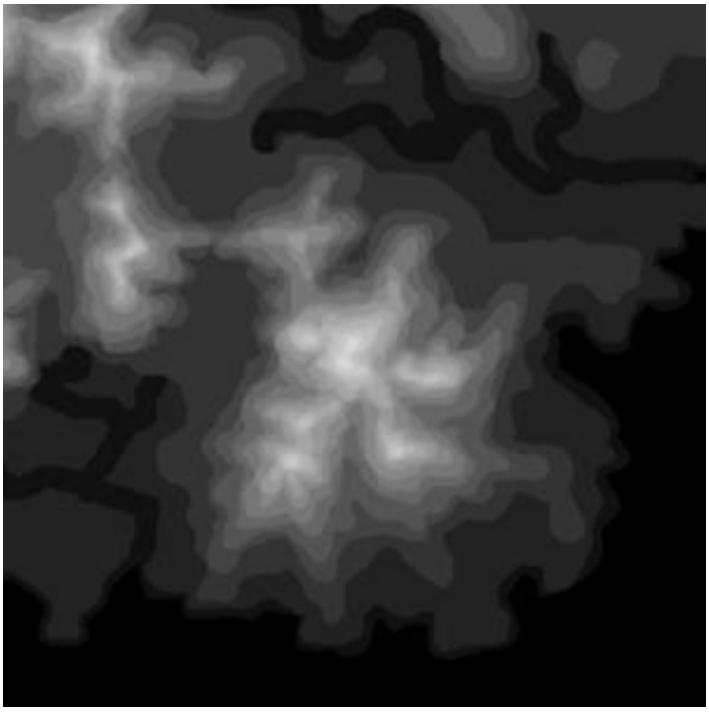
Из данного графика можно сделать вывод, что Холмовой алгоритм генерирует карту высот быстрее, чем шум Перлина.

4.4 Примеры работы программы

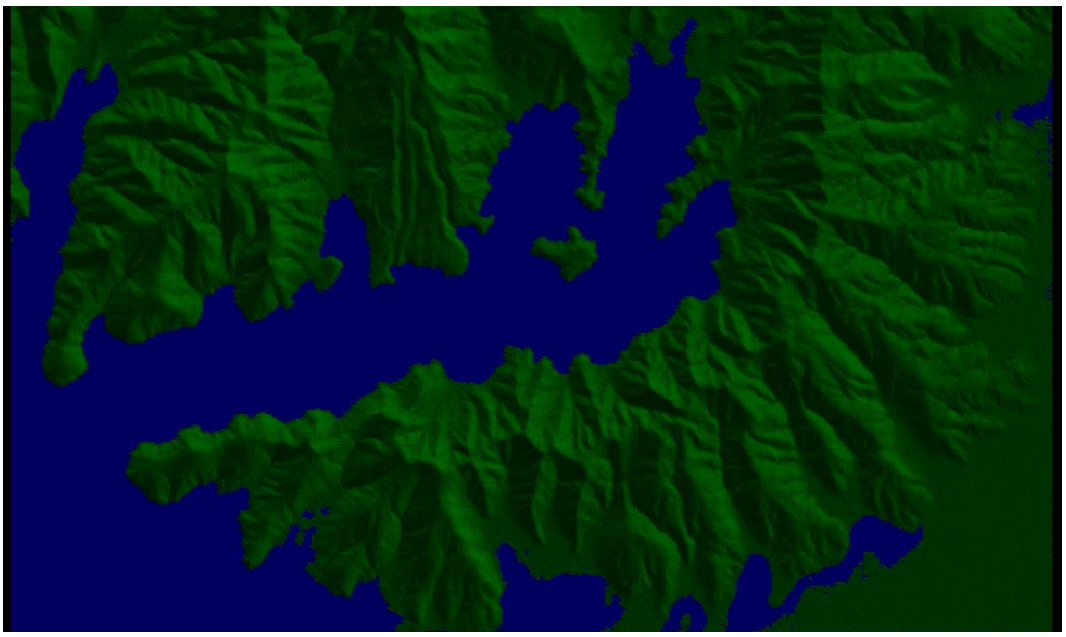
1.



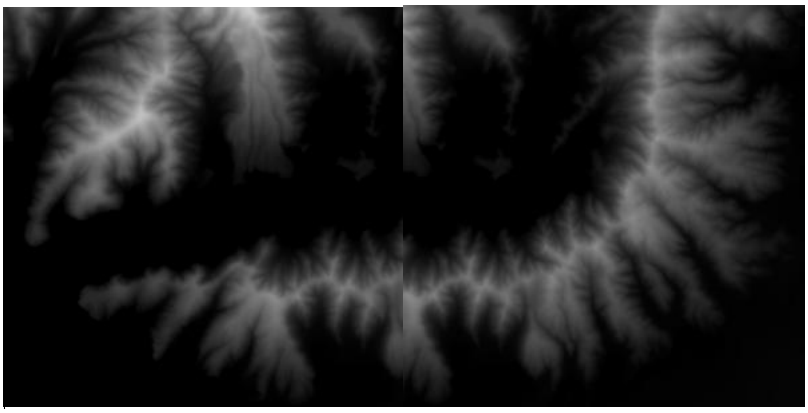
Карта высот:



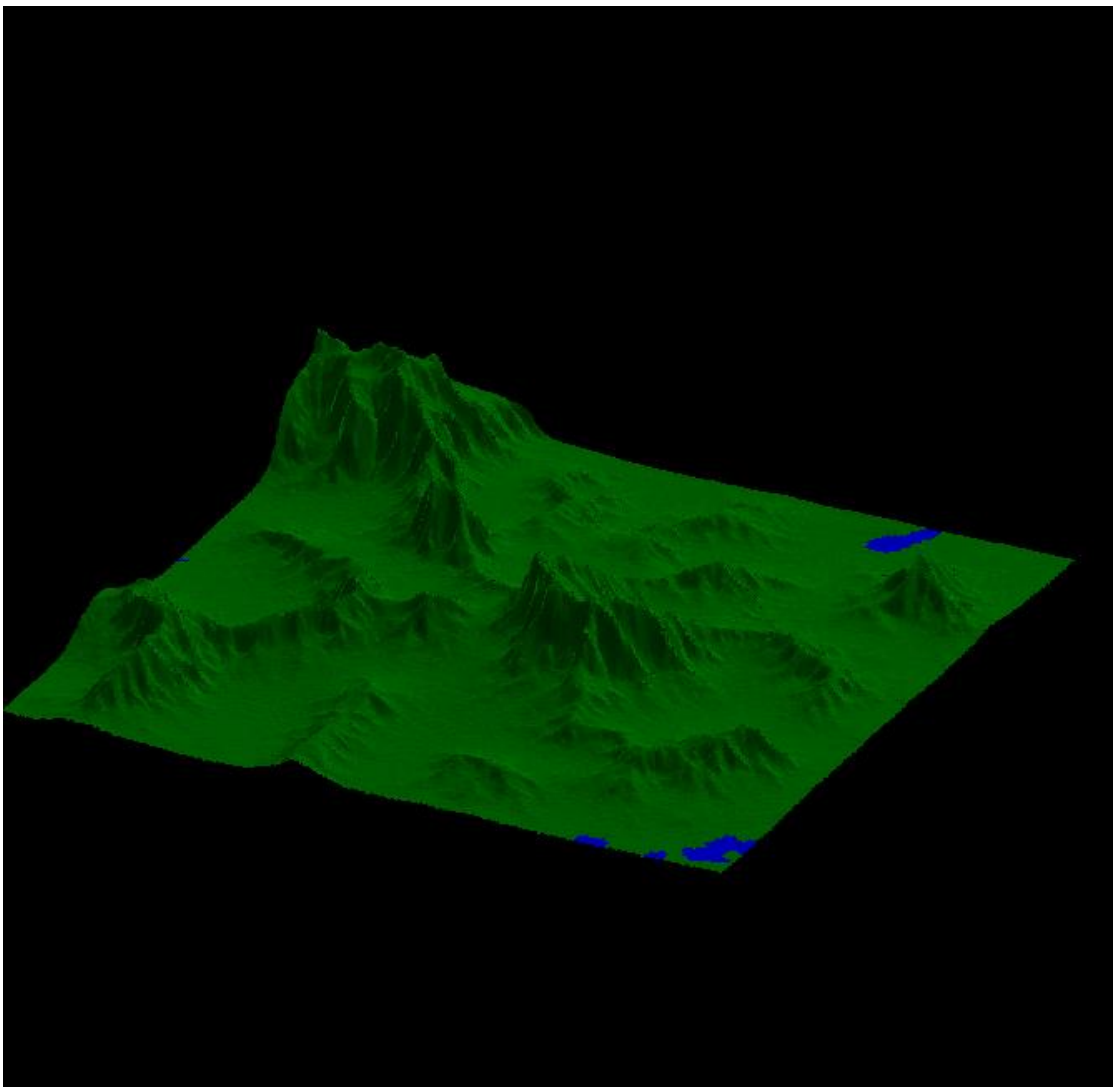
2.



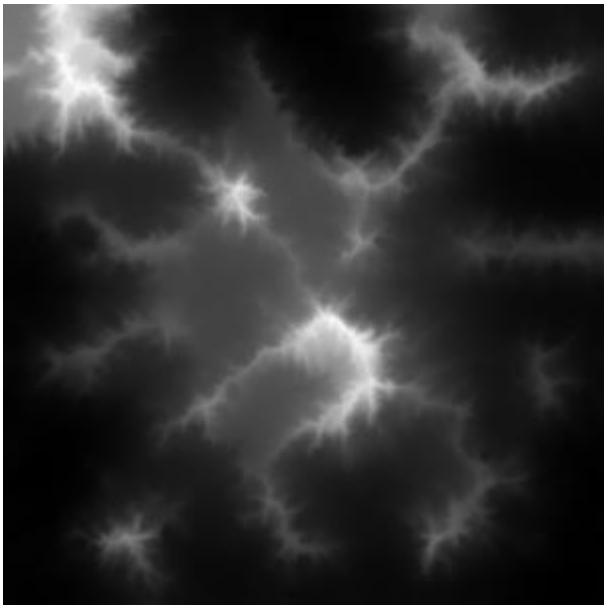
Карта высот:



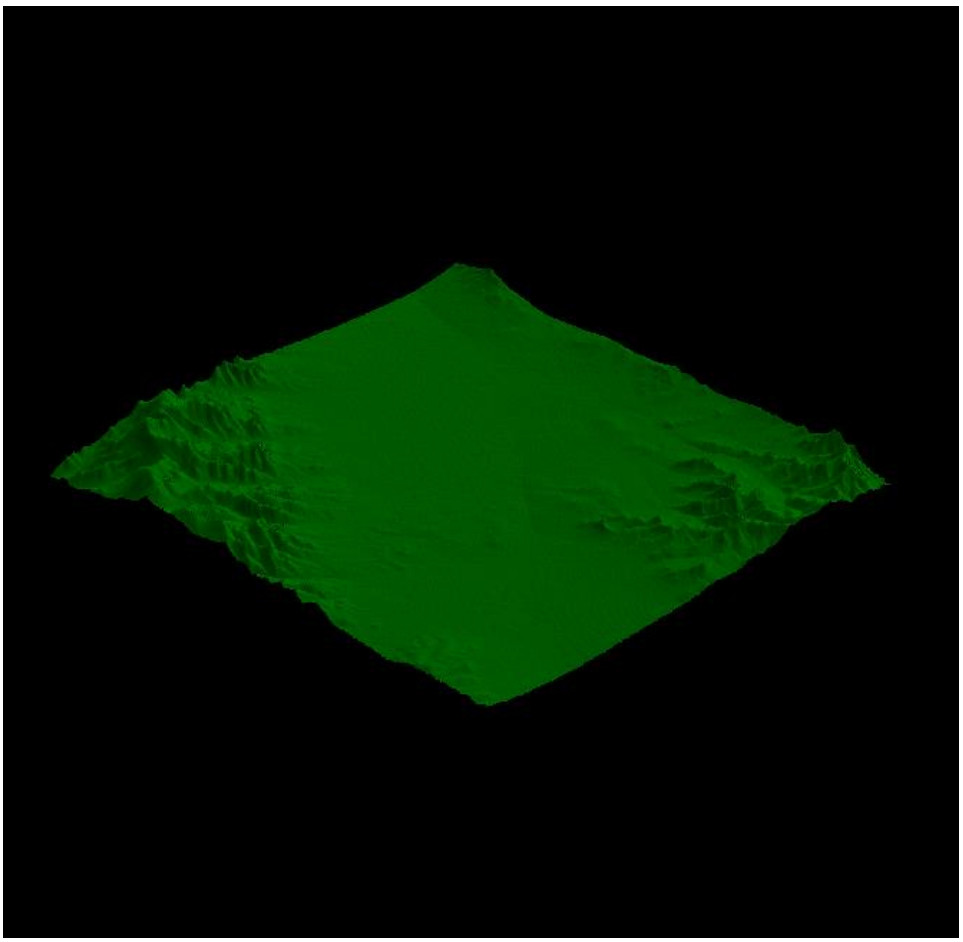
3.



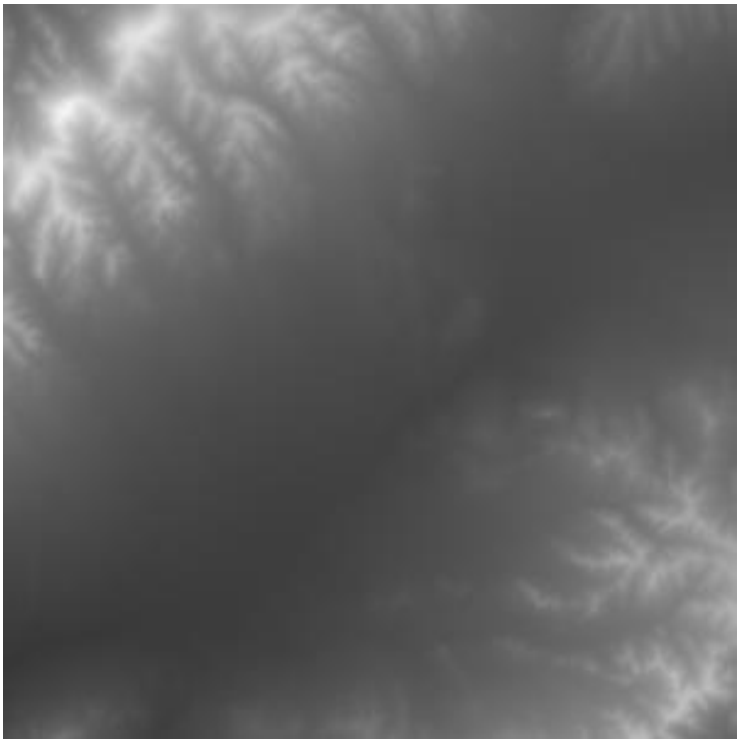
Карта высот:



4.

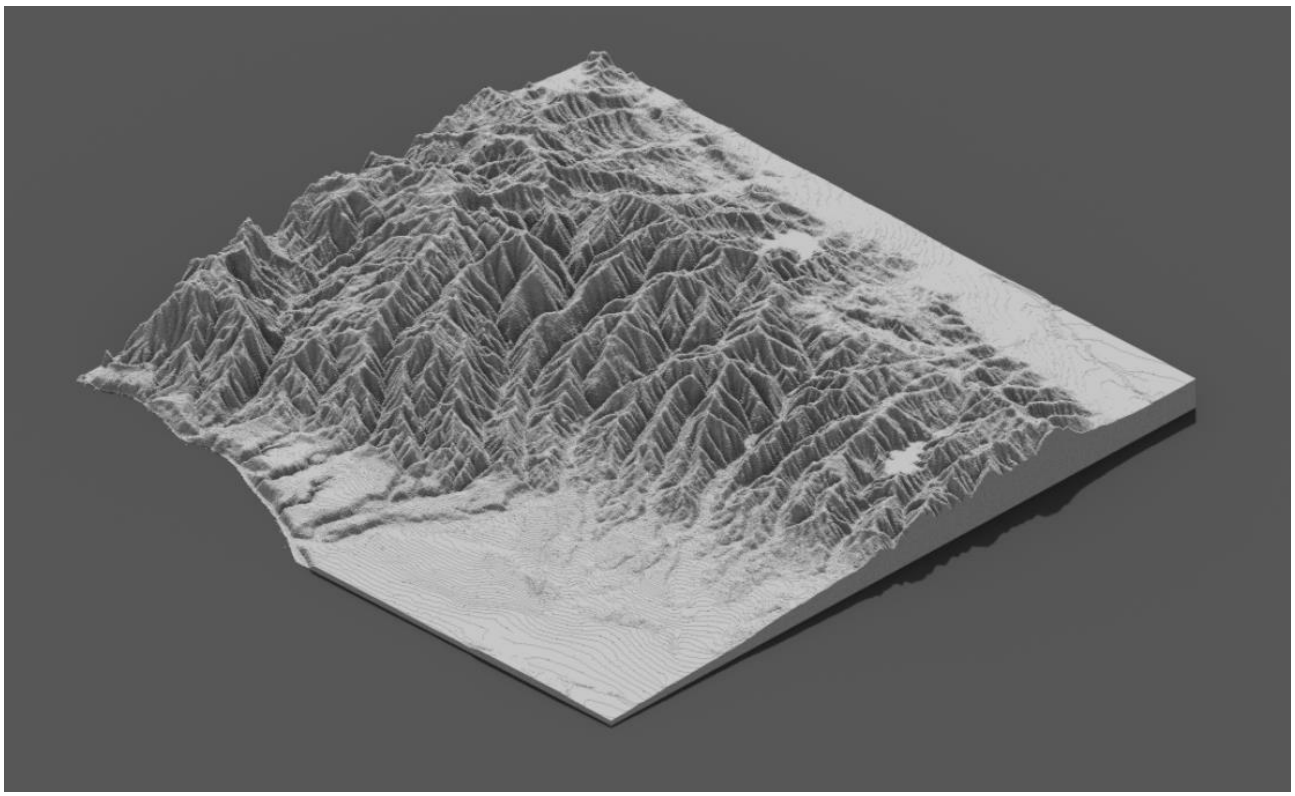


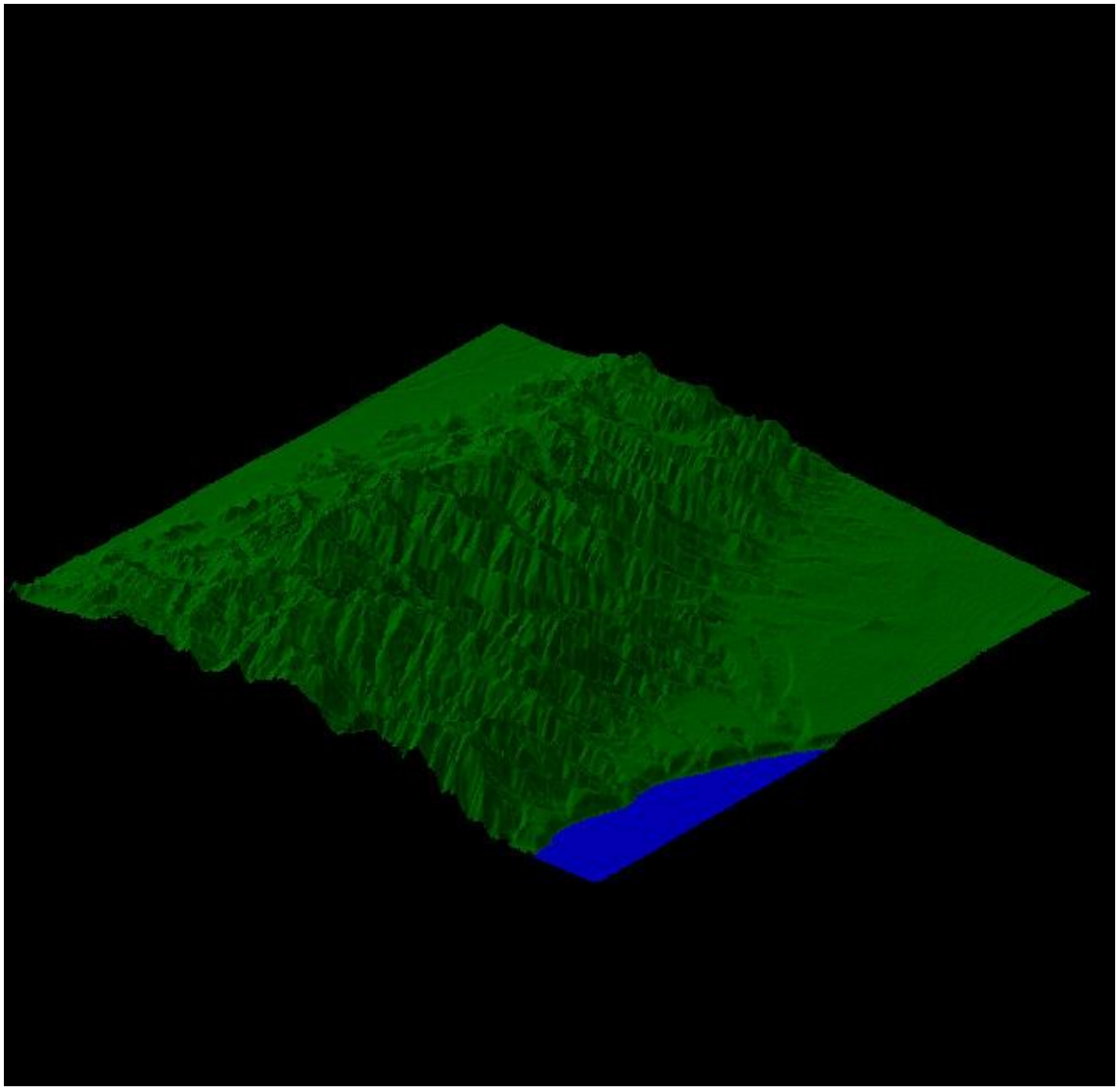
Карта высот:



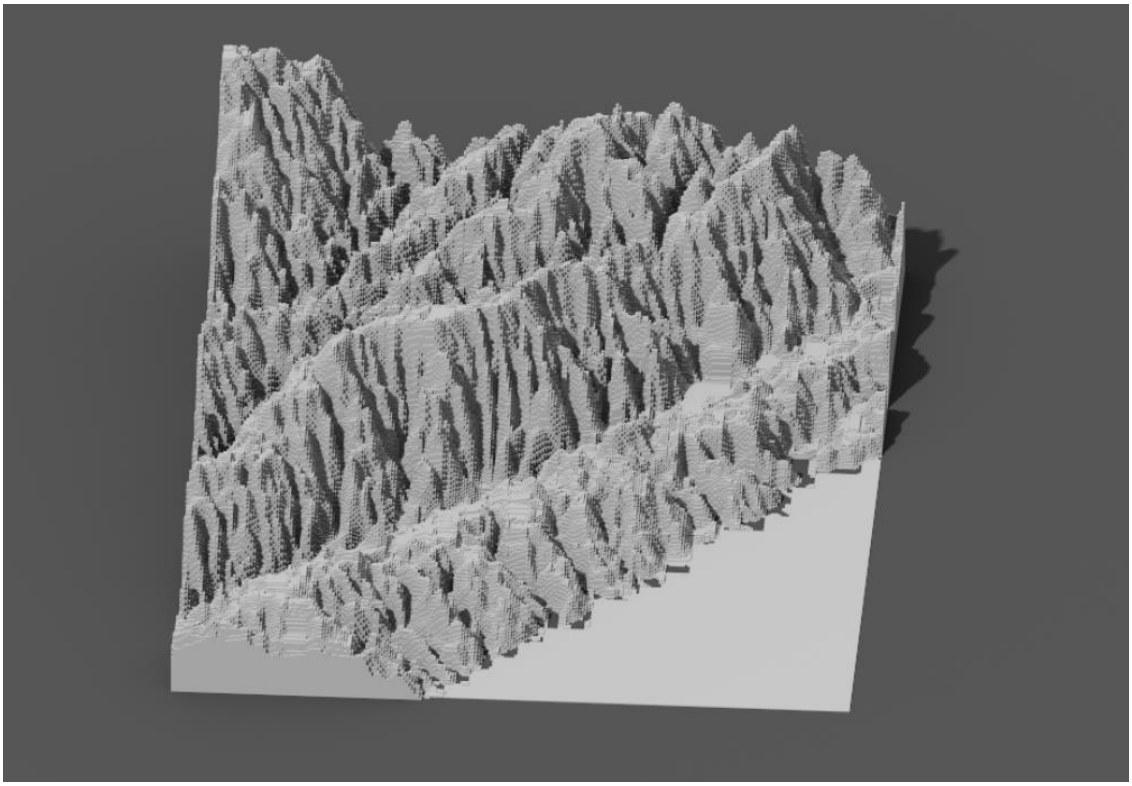
Далее я приведу результат своей программы и программы Aerialod, которая также генерирует ландшафт по карте высот:

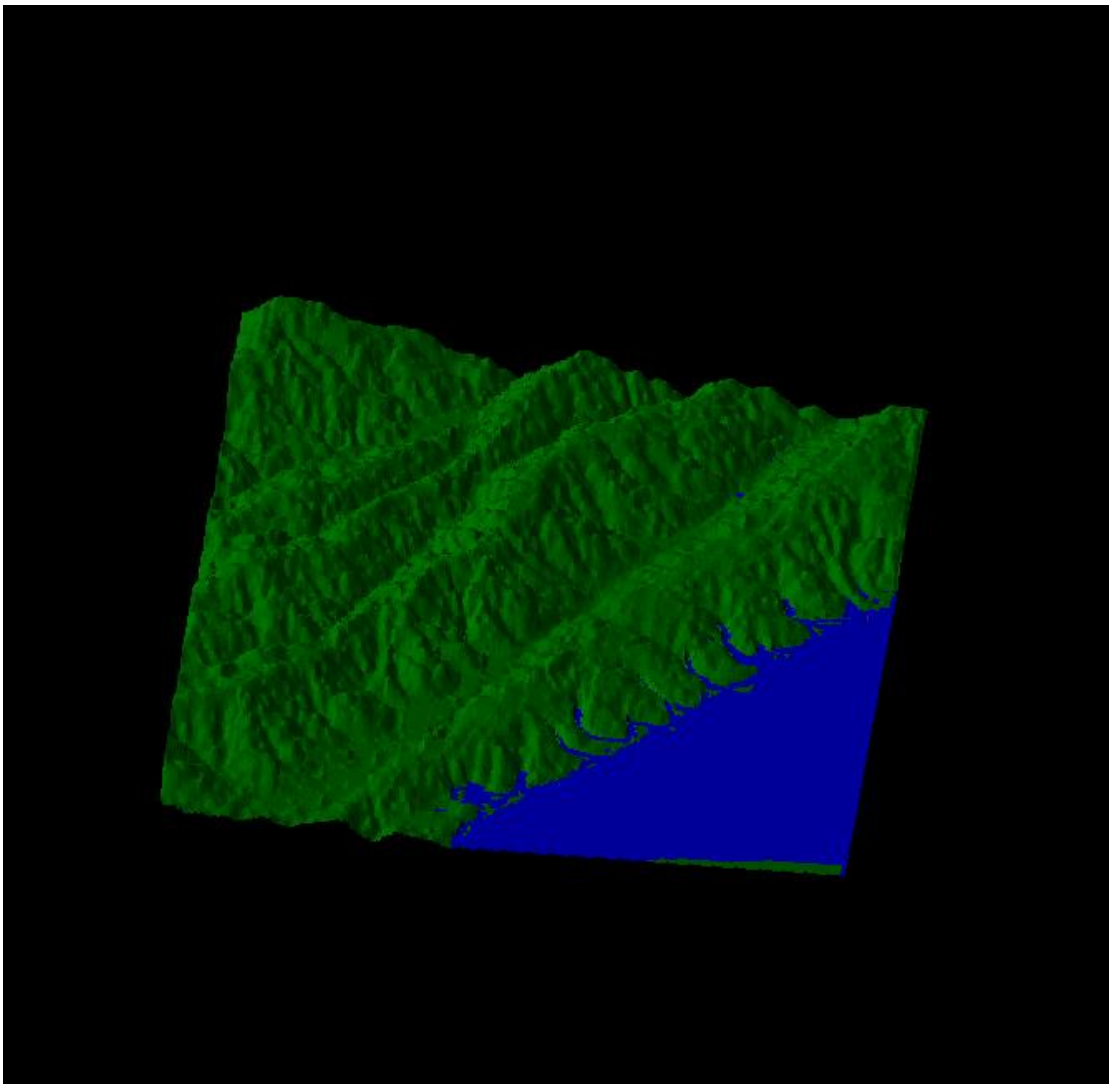
1.



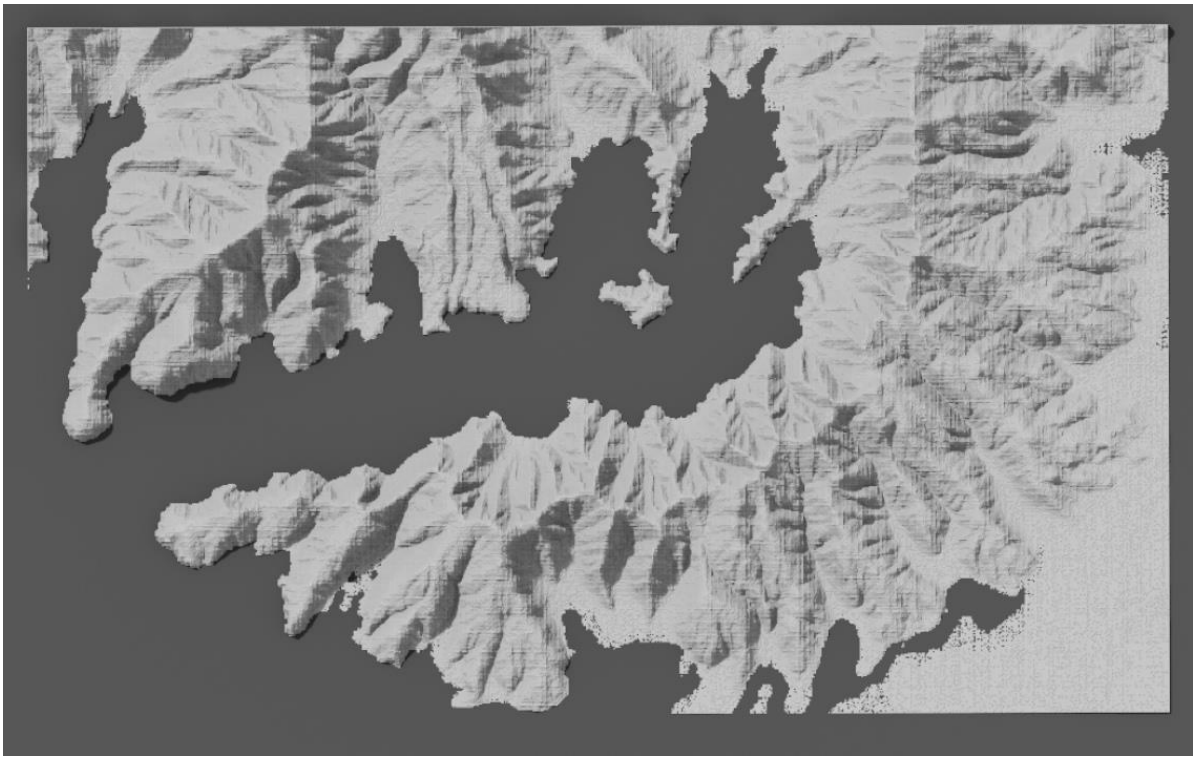


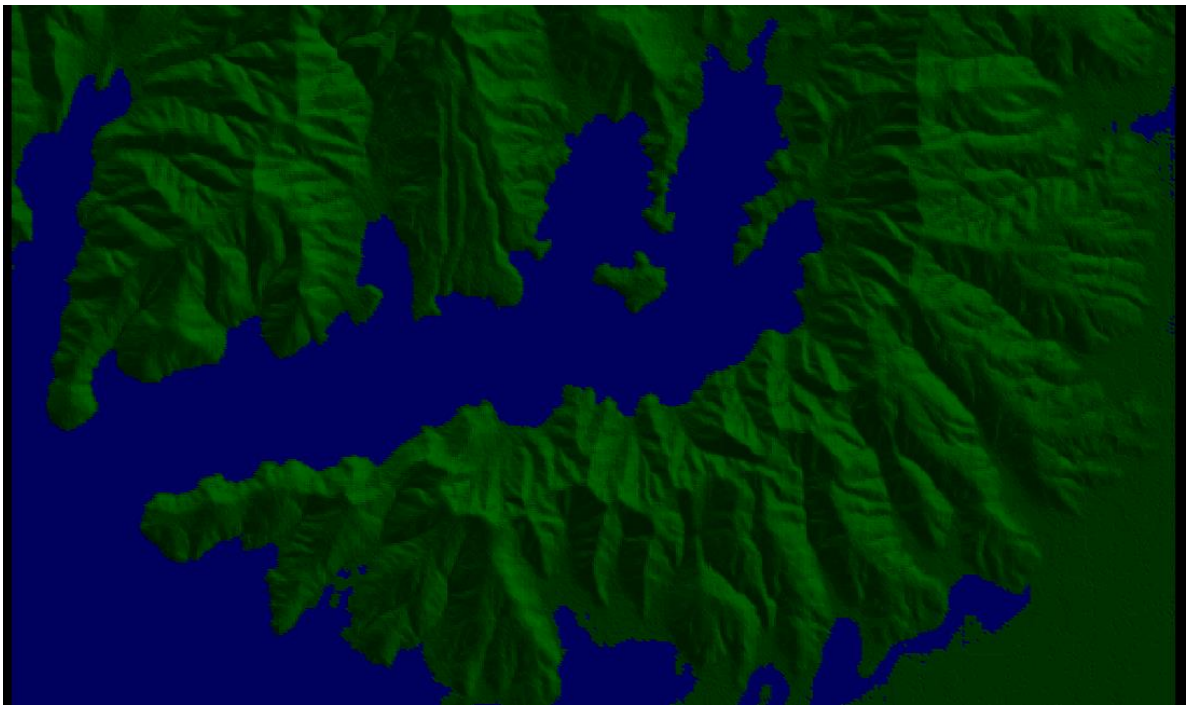
2.





3.





4.5 Вывод из исследовательской части

В данном разделе были показаны примеры работы программы. Также были продемонстрированы результаты экспериментов.

Заключение

Разработанный программный комплекс обеспечивает возможность генерации или загрузки карт высот. Также он позволяет строить изображения ландшафта по карте высот. Проведены исследования работы алгоритмов на различных наборах исходных данных.

Тем не менее, существует множество путей усовершенствования описанного программного комплекса:

1. генерируемые текстуры ландшафта все-таки выглядят не очень реалистично, из-за того, что на них не накладывается текстура грунта, травы;
2. в данной программе нет генерируемых растительности и деревьев. Внесение данных нововведений, несомненно, добавило бы реалистичности в генерируемый мир и расширило бы его возможности.

Список литературы

1. Scott Turner. Шум Перлина, процедурная генерация контента и интересное пространство[ЭЛ. РЕСУРС] Режим доступа: url:https://habr.com/ru/post/440286/.
2. Wikipedia. RGB[ЭЛ. РЕСУРС] Режим доступа: url: https://ru.wikipedia.org/wiki/RGB.
3. Кулагин Денис. Модель отражения Фонга[ЭЛ. РЕСУРС] Режим доступа: url: https://compgraphics.info/3D/lighting/phong_reflection_model.php.
4. Денис Кожухов. Генерация трехмерных ландшафтов[ЭЛ. РЕСУРС] Режим доступа: url:https://www.ixbt.com/video/3dterrains-generation.shtml.
5. Вадим Репин. Иерархический Z-буфер[ЭЛ. РЕСУРС] Режим доступа: url: https : / / www .ixbt.com/video/hierar-z.html.
6. Светлана Шляхтина. Генераторы ландшафтов [ЭЛ. РЕСУРС] Режим доступа: url:https://compress.ru/article.aspx?id=16597