Содержание

Определения

В данной расчетно-пояснительной записке применяются следующие термины с соответствующими определениями:

- октава - повторение;
- лакунарность - величина, контролирующая значение частоты;
- постоянство - величина, контролирующая изменение амплитуды;
- зерно - значение, которое позволяет сформировать новые значения таблицы перестановок в алгоритме шума Перлина.

Обозначения и сокращения

В данной расчетно-пояснительной записке применяются следующие обозначения и сокращения:

- ПО - программное обеспечение.

Введение

По мере развития научных познаний и вычислительной техники человечество сталкивается с новыми запросами, которые необходимы для решения различных задач. Для автоматизации процессов человек создал вычислительную машину — компьютер и нашел ему множество применений в различных областях.

В последние десятилетия машинная графика приобрела высокую популярность. Появляется запрос на мониторинг и управление окружающей средой. Требовались различные средства для удовлетворения таких потребностей. Со временем в машинной графике стало выделяться такое направление как 3d-моделирование и визуализация ландшафта.

Ландшафт представляет собой видимые особенности участка земли и его формы рельефа. Создание его вручную занимает определенное число ресурсов, на это тратится время. В связи с этим возникает потребность в программном обеспечении, которое позволило бы автоматизировать эти процессы, чтобы можно было быстро работать с такой моделью.

Целью данного курсового проекта является разработка программного обеспечения для визуализации трехмерного ландшафта. Для достижения поставленной цели необходимо решить следующий набор задач:

- выполнить формализацию объектов синтезируемой сцены;
- провести исследование существующих алгоритмов решения поставленной цели;
- выбрать и описать подходящие алгоритмы для визуализации сцены и поставленной задачи;
- привести схемы используемых алгоритмов;
- описать использующиеся структуры данных;
- определить средства реализации ПО;
- реализовать ПО;

- выполнить исследование временных характеристик алгоритмов визуализации сцены.

1 Аналитический раздел

В данном разделе представлено описание 3D-сцены, а также рассмотрены алгоритмы визуализации сцены и генерации ландшафта.

1.1 Формализация объектов сцены

Сцена состоит из следующих объектов:

- ландшафт — представлен трехмерной моделью. Предусмотрено задание характеристик ландшафта для изменения вида. Доступны настройки изменения размера (длина, ширина);
- источник света — представляет собой материальную точку, которая испускает лучи света.

1.2 Представление данных о ландшафте

В настоящее время существует несколько основных принципов представления данных для хранения информации о ландшафте [1].

1. Использование регулярной сетки высот (карта высот).
2. Использование иррегулярной сетки высот (хранение триангулированной карты).
3. Посегментная карта высот.

Рассмотрим более детально первые два способа.

1.2.1 Регулярная сетка высот

Модель равномерной сетки описывает координаты отдельных точек поверхности следующим образом: каждому узлу сетки с индексами (i, j) приписывается значение высоты z_{ij} (рисунок 1.1).

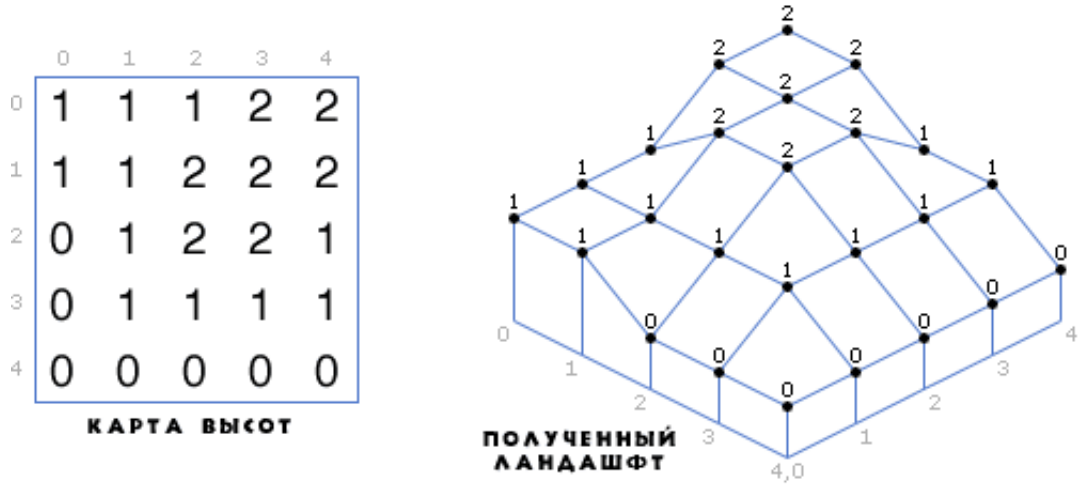

Рисунок 1.1 – Регулярная сетка высот

Индексам (i, j) соответствуют значения координат $(x,)$ точки. Расстояние между узлами одинаковое dx по оси OX, d по оси OY. Фактически такая модель - это матрица, каждый элемент которой сохраняет значение высоты z.

При помощи равномерной сетки высот часто описывают рельеф земной поверхности, так как такой подход имеет ряд преимуществ.

- Довольно просто описывается поверхность, легкая модификация данных.
- Легкость нахождения координат и их высот.
- Можно производить динамическое освещение ввиду того, что вершинные точки расположены равномерно и близко друг к другу.
- Наглядность - карту высот можно хранить в графическом виде.
- Промежуточные значения можно вычислить путем интерполяции значений в вершинах сетки.

Недостатки:

- при таком подходе невозможно описать все виды поверхностей, так как здесь используется для каждой координаты x, y единственное значение z (то есть получаем функцию $z = f(x, y)$). Это является той поверхностью, которую каждая вертикаль пересекает только один раз;

- для описания сложных поверхностей требуется большее число узлов.

1.2.2 Иррегулярная сетка вершин и их соединяющих

Неравномерной сеткой называется модель описания поверхности в виде множества отдельных точек $(x_0, y_0, z_0), (x_1, y_1, z_1), ...(x_{n-1}, y_{n-1}, z_{n-1})$, принадлежащих ей. Эти точки могут быть получены, например, в результате измерений поверхности какого-нибудь объекта с помощью определенного оборудования. Равномерная сетка могут считается разновидностью неравномерной сетки вершин.

Рассмотрим модель поверхности в виде множества точечных значений, логически никак не связанных между собой (рисунок 1.2).

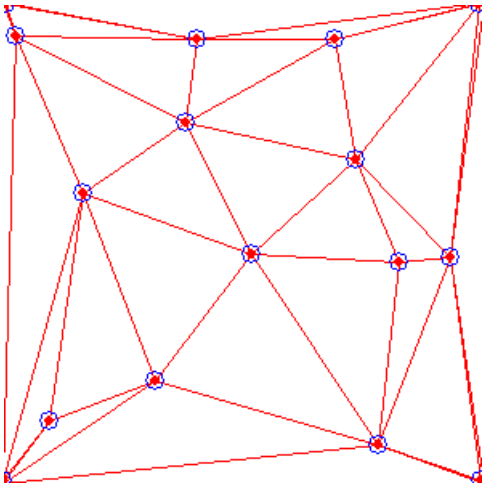

Рисунок 1.2 – Иррегулярная сетка высот

Неравномерность задания опорных точек усложняет определение координат для других точек поверхности, которые не совпадают с опорными точками. Требуются специальные методы пространственной интерполяции.

Выделяют следующие положительные черты такого подхода:

- использование отдельных опорных точек, наиболее важных для заданной формы поверхности (имеет меньший объем информации по сравнению с другими моделями, например с равномерной сеткой);

- применение изолиний на картах и планах позволяет наглядно отображать рельеф поверхности.

Но у такого метода содержатся существенные недостатки:

- невозможность или сложность выполнения многих операций над поверхностями;
- возникновение сложностей при динамическом освещении — вершины расположены достаточно далеко друг от друга и неравномерно.

1.3 Алгоритм генерации карты высот, использующий шум Перлина

Возникает проблема — как же получить эту карту высот. Для этого разработан алгоритм, который прост в понимании и являющийся некой базой для дальнейшей модификации и получении интересного результата — использование шума. Сам шум представляет собой набор чисел. Значение шума связывают с высотой. Таким образом, получаем карту высот. Первый революционный алгоритм, нашедший широкое применение в компьютерной графике, основан на шуме Перлина [2]. Шум Перлина - это функция генерации когерентного шума в пространстве (рисунок 1.3).

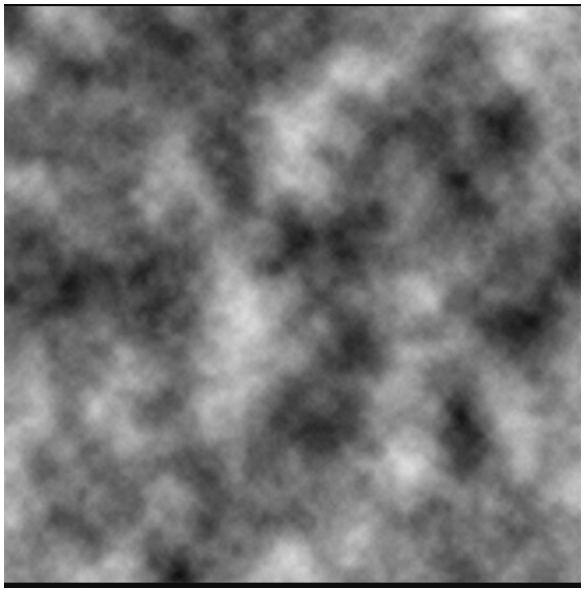

Рисунок 1.3 – Шум Перлина

Кен Перлин предложил использовать вместо случайных значений в углах ячейки градиентный шум. Данный алгоритм может быть реализован для n-мерного пространства, но чаще всего реализуется для двух-, трех- или четырехмерного. Визуализация осуществляется как восьмиразрядные изображения с градациями серого, где каждая точка хранит высоту ландшафта в соответствующей позиции. Значения этой карты варьируются в диапазоне от 0 до 255, где 0 представляет самую низкую высоту вершины (на карте отображается как черный цвет), а 255 - максимально возможную (белый цвет). Этот интервал можно расширить, используя коэффициент масштабирования, который умножается на заданное значение высоты. Такой способ обеспечивает больший интервал высот, но с меньшей точностью между значениями. По сравнению с обычной полигональной сеткой они требуют значительно меньше памяти для заданного уровня детализации.

Шум Перлина уникален тем, что ни одна из карта высот, созданная этим алгоритмом, не похожа на предыдущую.

Общая идея для создания n-мерной сетки: в каждом узле этой сетки создается вектор из n-мерного пространства.

Для начала следует разобрать, как работает данный алгоритм для одномерного случая: Пусть имеем оси Ox и Oy. На числовой оси в точках $x = 0, 1, 2, ..$ выберем случайным образом значение i из интервала $(-1; 1)$ и на этих точках отложим градиент функции $g(i)$ как на рисунке 1.4.

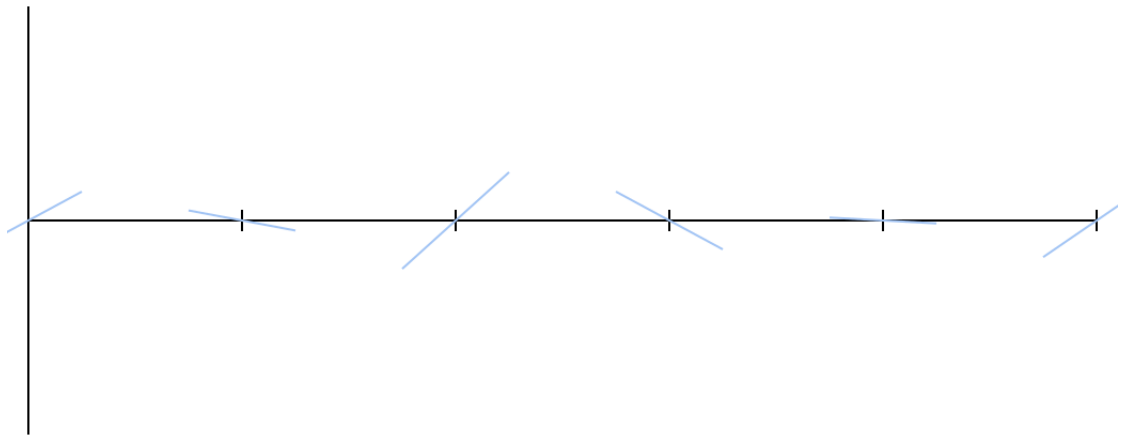

Рисунок 1.4 – Градиенты на оси Ox

Если продолжить эти линии и опустить перпендикуляры из некоторых точек этих прямых к оси Ox, то получим следующую картину

(рисунок 1.5).

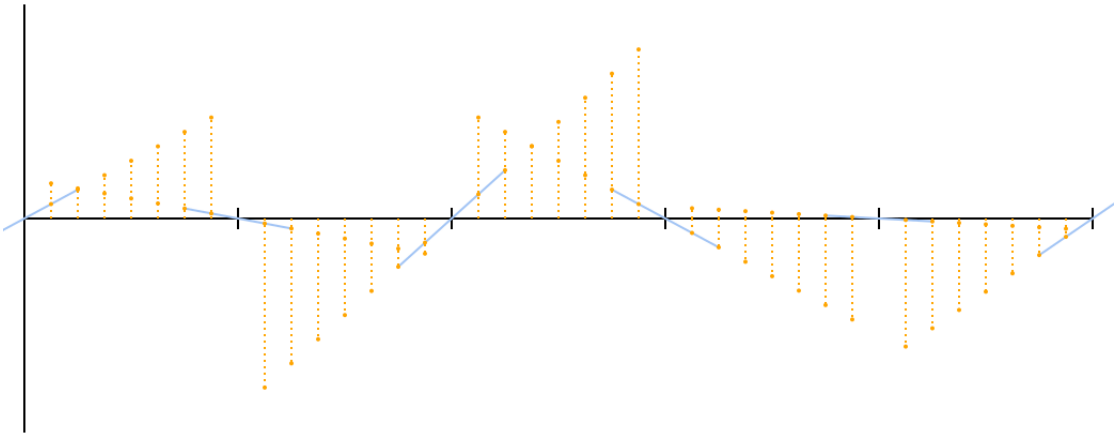

Рисунок 1.5 – Перпендикуляры к оси Ox

Теперь нужно выяснить, как можно пары точек для каждого «x» представить в виде гладкой кривой. Для решения этой проблемы применяют функцию, называемую smoothstep. Применяется полином третьей степени по формуле (1.1):

$$smoothstep(x) = 3x^2 - 2x^3 \tag{1.1}$$

Таким образом, выполнились необходимые условия: получен плавный переход между сегментами функции (производная в целочисленных значениях уна оси Ox равна нулю), а также альтернативу использования более сложных методов интерполяции. Следовательно, интерполируется не сам градиент, а его скалярное произведение на вектор от узла до точки.

Спустя некоторое время Кен Перлин предлагает улучшенную версию функции *smoothstep* по формуле (1.2), которая при $x = 0$ и $x = 1$ имеет нулевые производные первого и второго порядков:

$$smoothstep(x) = 6x^5 - 15x^4 + 10x^3 \tag{1.2}$$

Эти две функции очень похожи (рисунок 1.6), но кривая пятой степени также имеет нулевую вторую производную в своих конечных точках, что делает функцию шума всюду непрерывной и более подходящей для общих задач компьютерной графики, связанных со смещением поверхности и отображением рельефа [3].

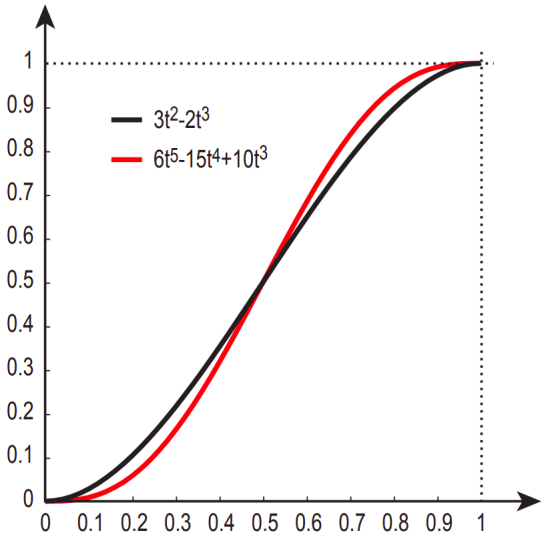

Рисунок 1.6 – Шум Перлина

Для создания более органичного и интересного результата необходимо добавить несколько шумовых функций, называемых октавами. Каждая функция шума имеет частоту, в два раза большую чем предыдущая. Поэтому градиенты будут находиться в точках ($x = 0, 0.5, 1, ..$). Применив до 4-5 октав к исходной функции для одномерного измерения, получим следующую картину (рисунок 1.7).

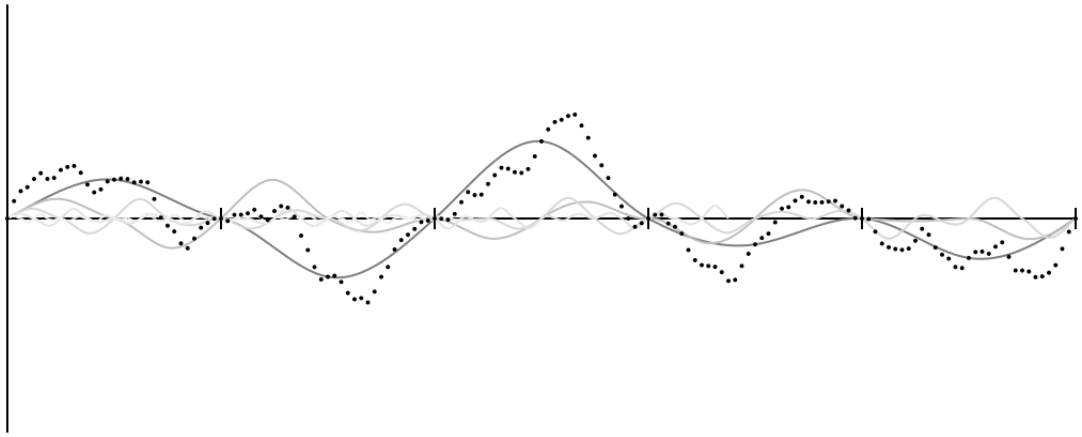

Рисунок 1.7 – Результат наложения нескольких октав

Использование 2d-измерения: Основное отличие от одномерного случая — целочисленные координатные точки определяют регулярную сетку (рисунок 1.8) [4]. В каждой точке выбирается псевдослучайный единичный вектор — градиент g, для которого важно только направление (рисунок 1.9).

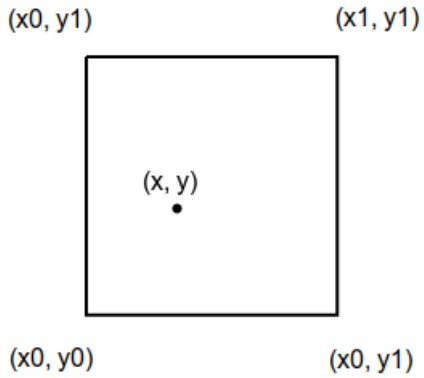

Рисунок 1.8 – Регулярная сетка

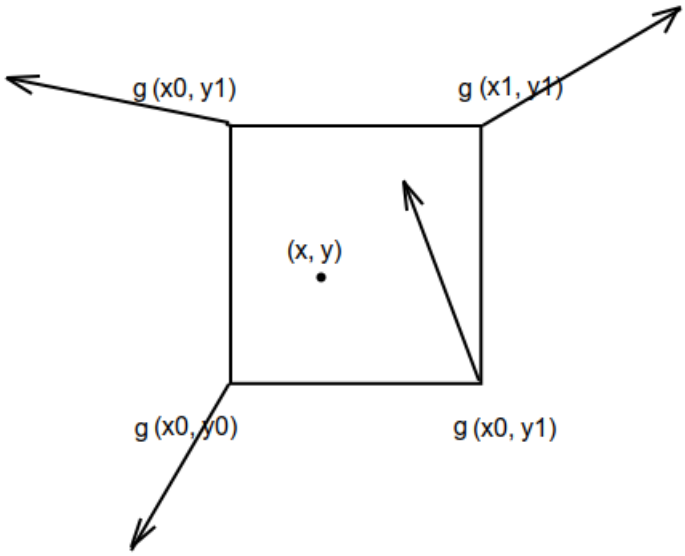

Рисунок 1.9 – Псевдослучайные градиенты, связанные с узлами сетки

Градиент g имеет вид случайности, но с важным учетом того, что он всегда возвращает один и тот же градиент для одной и той же точки сетки, каждый раз, когда он вычисляется. Также важно, чтобы у каждого направления были равные шансы быть выбранным. Для каждой точки сетки мы генерируем вектор, идущий от точки сетки до координат (x, y), который легко вычисляется путем вычитания точки сетки из (x, y) (рисунок 1.10).

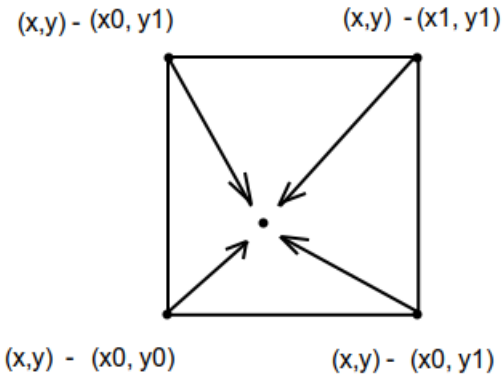

Рисунок 1.10 – Векторы из узлов сетки до точки (x, y)

Расчет влияния каждого псевдослучайного градиента на конечный результат позволяет сгенерировать результат как средневзвешенное значение этих влияний. Для этого вычисляется скалярное произведение градиента и вектора, идущего от связанной с ним точки сетки к точке с координатами (x, y) по формулам (1.3), (1.4), (1.5), (1.6):

$$s = g(x_0, y_0) \cdot ((x, y) - (x_0, y_0)) \tag{1.3}$$

$$t = g(x_1, y_0) \cdot ((x, y) - (x_1, y_0)) \tag{1.4}$$

$$u = g(x_0, y_1) \cdot ((x, y) - (x_0, y_1)) \tag{1.5}$$

$$v = g(x_1, y_1) \cdot ((x, y) - (x_1, y_1)) \tag{1.6}$$

Результат применения формул (1.3), (1.4), (1.5), (1.6) показан на рисунке 1.11.

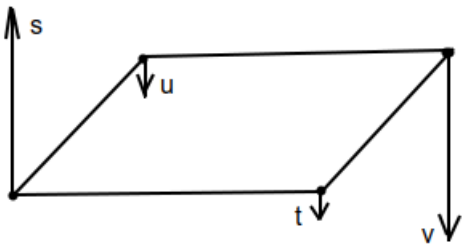

Рисунок 1.11 – Влияние на узлы сетки

Затем выполняется нахождение средневзвешенного значения s и t, построив билинейную функцию, которая отображает 0 в s и 1 в t, и оценив ее по весу измерения x. Мы назовем это среднее значение *lerpa*.

Выполним то же самое для u и v и запишем результат в $lerpb$. Математически можно рассчитать для точки P(x,y) по формулам (1.7), (1.8), (1.9), (1.10), (1.11), (1.12):

$$fade = 6t^5 - 15t^4 + 10t^3 \tag{1.7}$$

$$S_x = fade(x) \tag{1.8}$$

$$S_y = fade(y) \tag{1.9}$$

$$lerpa = s + S_x \cdot (t - s) \tag{1.10}$$

$$lerpb = u + S_x \cdot (v - u) \tag{1.11}$$

$$lerp = lerpa + sy(lerpb - lerpa) \tag{1.12}$$

Теперь находится вес для измерения y, S_y, оценивая кривую сглаживания при $y - y_0$, и, наконец, берется взвешенная сумма a и b, чтобы получить выходное значение z.

Применив для двумерного случая несколько октав, получим интересный результат (рисунок 1.12).

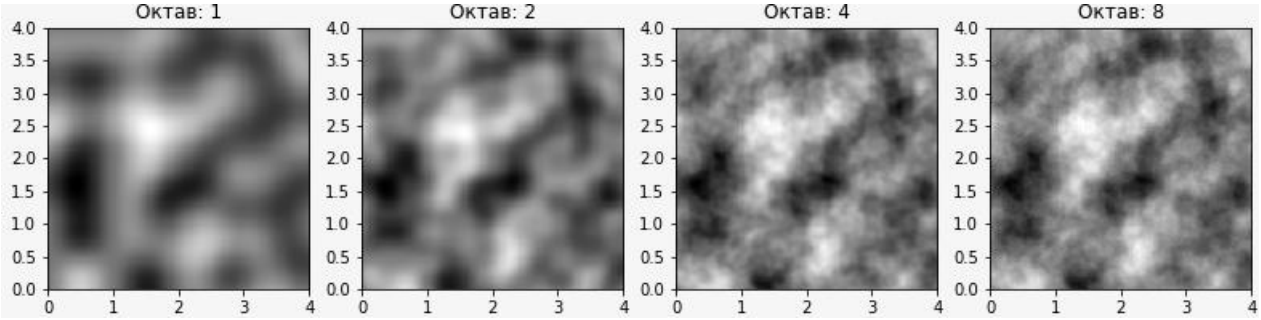

Рисунок 1.12 – Изменение детализации шума Перлина с применением нескольких октав

1.4 Алгоритмы удаления невидимых ребер и поверхностей

1.4.1 Алгоритм Варнока

Алгоритм работает в пространстве изображения.

Основная идея данного алгоритма заключается в том, что исходный экран рассматривается как окно. Производится анализ этого окна: пусто ли его содержимое или оно достаточно для визуализации. Если это не так, то производится разбиение данного окна на подобласти (рисунок 1.13) до тех пор, пока информация, содержащаяся в нем, не станет достаточной для визуализации данного фрагмента. Если полученной информации достаточно, то происходит её усреднение и результат изображается одинаковым цветом или интенсивностью. Для эффективной работы данного алгоритма требуется выбрать наиболее подходящий способ разбиения окна, тем самым усложнив способ и критерий разбиения. Для растрового дисплея на экране пределом деления окна является один пиксель - в случае простого алгоритма. Единой версии данного алгоритма не существует, существует только его различные модификации.

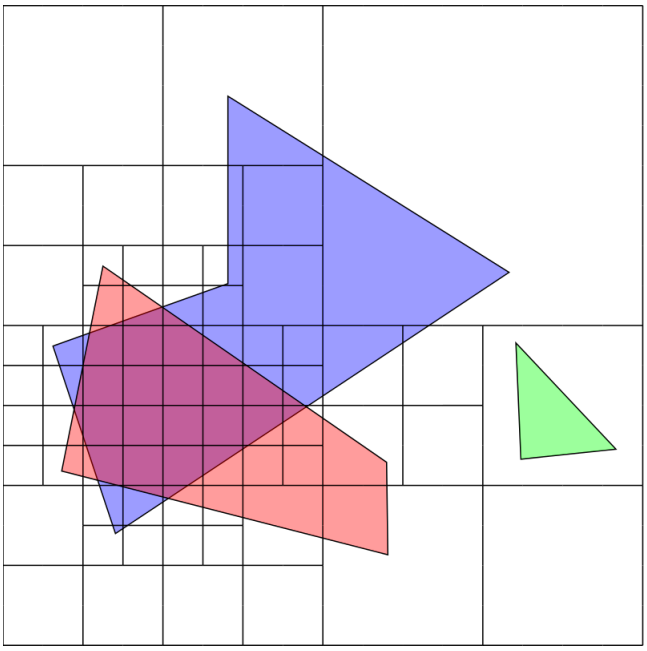

Рисунок 1.13 – Разбиение окна на подобласти

К недостатку данного алгоритма относится следующее: при увеличении сложности сцены число разбиений может так увеличиться, что приведет к снижению временной эффективности работы программы.

1.4.2 Алгоритм Z-буфера

Алгоритм работает в пространстве изображения.

Основная идея данного алгоритма — использование буфера глубины каждого видимого пикселя изображения сцены [5]. Для каждого объекта сцены сравнивается значение координаты z с буфером глубины, который изначально заполняется значением, соответствующим максимальному. Если рассматриваемый пиксель находится ближе пикселя, который находится в буфере кадра, то производится корректировка, и этот пиксель заносится в этот буфер. Используется также и другой буфер, который запоминает цвет пикселя изображения. Если сцена подвергается видовому преобразованию и отсекается до фиксированного диапазона координат z значений, то можно использовать z-буфер с фиксированной точностью. Информацию о глубине нужно обрабатывать с большей точностью, чем координатную информацию на плоскости (x, y); обычно бывает достаточно 20 бит.

Преимущества:

- позволяет работать со сложными объектами сцены — позволит достаточно легко выполнить визуализацию пересечения сложных поверхностей;
- объекты сцены могут заноситься в произвольном порядке — не нужно выполнять предварительную сортировку по глубине, что позволит сделать выигрыш по производительности;
- сцена может обрабатываться любого размера — вычислительная сложность не более чем O(n) — необходимо для быстрого рендеринга;
- буферы кадра и глубины вместе занимают памяти (буфер кадра размером $1920 \times 1080 \times 24$ бит в комбинации с z-буфером размером

1920 × 1080 × 20 бит требует около 11 Мб), что по сегодняшним темпам развития технологий не является критичным.

К недостаткам относится возникновение трудностей реализации эффектов прозрачности и просвечивания, а также устранение лестничного эффекта.

1.4.3 Алгоритм обратной трассировки лучей

Данный алгоритм работает в пространстве изображения.

В этом алгоритме происходит отслеживание лучей не от источников света, а наоборот, от точки наблюдателя — в обратном направлении (рисунок 1.14). При формировании конечного изображения учитываются только те лучи, которые вносят вклад в его формирование. То есть чтобы определить цвет пикселя экрана, необходимо из камеры провести луч и найти ближайшую точку пересечения со сценой, в которой и определяется освещенность этой точки. Она получается в результате сложения энергии отраженного и преломленного, которые получены непосредственно от источников света и также от других объектов сцены. В результат конечного изображения также учитывается и ослабление света, проходящий через прозрачный материал.

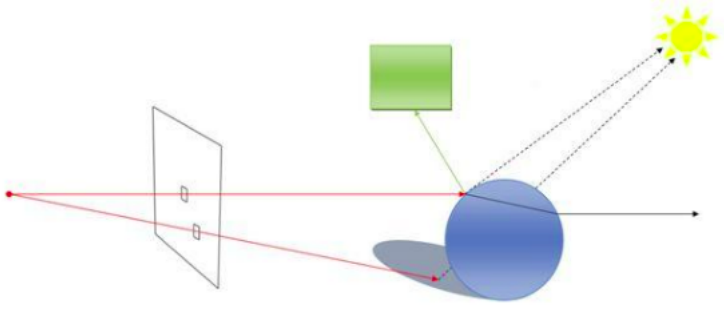

Рисунок 1.14 – Обратная трассировка лучей

Преимущества:

- позволяет получить сложные изображения, использующие законы геометрической оптики (преломление, отражение, тени);

- можно выполнить распараллеливание вычислений.

Недостатки:

- является одним из самых медленных алгоритмов синтеза изображений по сравнению с остальными.

1.5 Модель освещения

При работе с изображением для реалистичности необходимо также учитывать немало важный компонент — свет, который позволяет рассмотреть исходную модель более детально. Для этих целей существует модель освещения — выполнение расчетов интенсивности отраженного к наблюдателю света в каждом пикселе изображения.

Различают несколько видов модели освещения.

1. Локальная — в ней учитывается тот свет, который попал в рассматриваемую точку от источника света.

2. Глобальная — учитывает все свойства локального света, а также свет от других объектов, отраженный или пропущенный. Используя это, можно воспроизводить такие эффекты как преломление (прозрачность, полупрозрачность), отражение и затенение.

Существует несколько моделей освещенности, которые до сих пор пользуются популярностью и используются в современных программах: модель Ламберта и модель Фонга. Пусть заданы точечный источник света, расположенный в некоторой точке, поверхность, которая будет освещаться и наблюдатель. Будем считать, что наблюдатель точечный. Каждая точка поверхности имеет свои координаты, и в ней определена нормаль к поверхности. Для удобства все векторы, описанные ниже, берутся единичными. В этом случае косинус угла между ними совпадает со скалярным произведением.

1.5.1 Модель освещения Ламберта

Такое освещение называют диффузным, и его смысл заключается в том, что падающий свет отражается во всех направлениях (рисунок 1.15) [6].

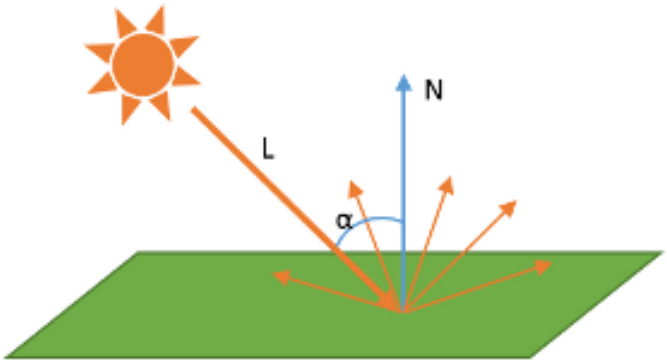

Рисунок 1.15 – Модель освещения Ламберта

Освещение рассчитывается по формуле (1.13):

$$I = (\overline{N}, \overline{L}) = \overline{N} \cdot \overline{L}, \tag{1.13}$$

где $\overline{N}$ - вектор нормали к поверхности в точке, $\overline{L}$ - падающий на точку луч света.

1.5.2 Модель освещения Фонга

Освещенность такой модели складывается из трех компонент: фоновое освещение, рассеянный свет и бликовая составляющая (рисунок 1.16). Свойства источника определяют мощность излучения для каждой из этих компонент, а свойства материала поверхности определяют её способность воспринимать каждый вид освещения.

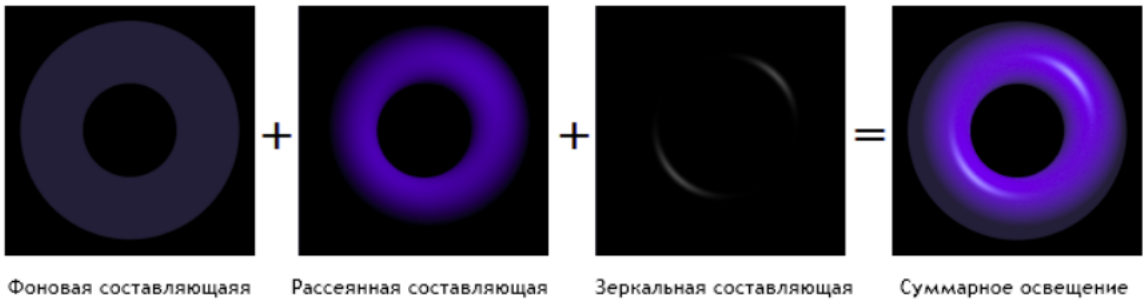

Рисунок 1.16 – Модель освещения Фонга

Освещение рассчитывается по формуле (1.14):

$$I = k_\alpha \cdot i_\alpha + k_d \cdot (\overline{N}, \overline{L}) + k_s \cdot (\overline{R}, \overline{V})^\rho, \tag{1.14}$$

где $\overline{N}$ - вектор нормали к поверхности в точке, $\overline{L}$ - падающий на точку луч света, $\overline{R}$ - отраженный луч, $\overline{V}$ - вектор, направленный к наблюдателю, k_α, k_d, k_s - коэффициенты фонового, диффузного и зеркального освещения соответственно; ρ - степень, аппроксимирующая пространственное распределение зеркально отраженного света. По сравнению с простой моделью освещения модель освещения Фонга обеспечивает большую реалистичность, но при этому требует больше вычислений.

1.6 Алгоритмы закраски

1.6.1 Простая закраска

Суть данного алгоритма заключается в том, что для каждой грани объекта находится вектор нормали, и с его помощью в соответствии с выбранной моделью освещения вычисляется значение интенсивности, с которой закрашивается вся грань.

Данный метод закраски обладает большим быстродействием по сравнению с другими методами, однако все пиксели грани имеют одинаковую интенсивность - сцена выглядит недостаточно реалистично.

1.6.2 Закраска по Гуро

В этом методе выполняется вычисление цвета модели для каждого пикселя[7]. Данный метод отличается от простой закраски тем, что разные точки грани закрашиваются при помощи разных значений интенсивности. Для это в каждой вершине грани находится усредненный вектор нормали всех плоскостей, в которых принадлежит эта вершина, и в них вычисляется значение интенсивности. Затем найденные значения интенсивности билинейно интерполируются по всем точкам грани (рисунок 1.17). При помощи такого алгоритма получится сглаженное изображение.

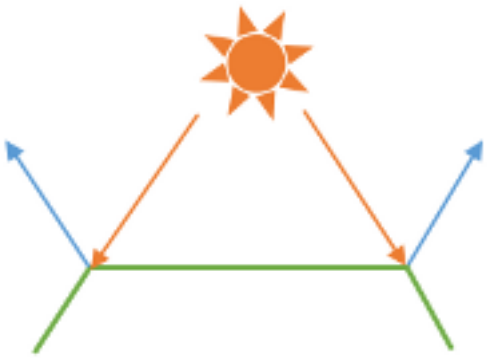

Рисунок 1.17 – Интерполяция интенсивностей закраски по Гуро

Недостатком такой интерполяции является блик как на рисунке 1.18, который будет правильно отрисован в вершине многоугольника, но неестественно распространен по соседним полигонам с помощью такого метода интерполяции.

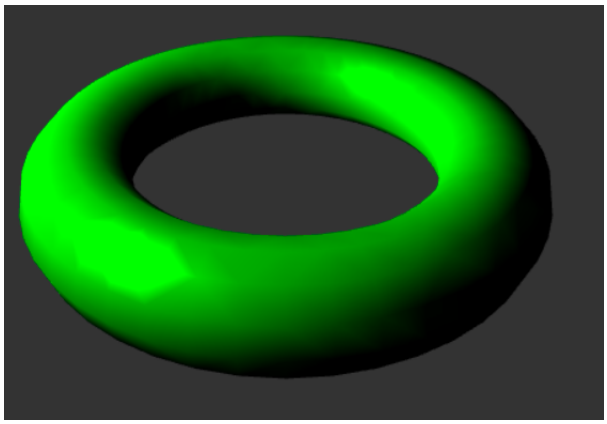

Рисунок 1.18 – Появление неестественного блика

1.6.3 Закраска по Фонгу

При таком подходе выполняется билинейная интерполяция не цвета, а нормали (рисунок 1.19). При таком подходе изображение получается более качественным, и исчезает проблема с бликами (рисунок 1.20). Но для достижения такого результата требуется больше вычислительных ресурсов — понижается скорость работы по сравнению с моделью Гуро. Из-за увеличения разрешения данный метод применятся в современных видеокартах при отрисовке изображения.

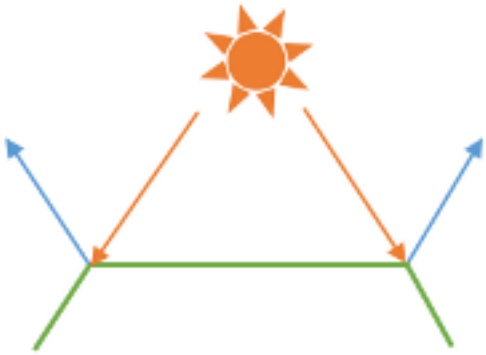

Рисунок 1.19 – Интерполяция нормалей закраски по Фонгу

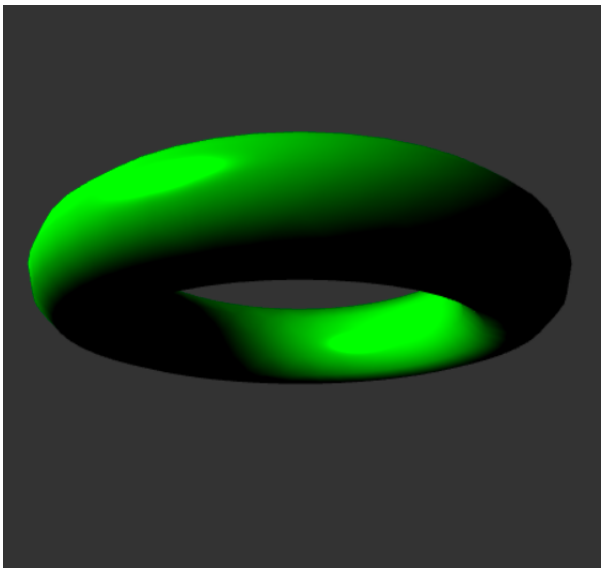

Рисунок 1.20 – Пример блика закраски Фонга

1.7 Вывод

В данном разделе был произведен анализ существующих методов, необходимых для построения и визуализации трехмерного ландшафта. Выбранный способ представления данных о трехмерном ландшафте — карта высот. Наряду с использованием шума Перлина будет использован алгоритм z-буфера. В качестве освещения выбрана модель Ламберта и модель интерполяции света по Гуро — в совокупности они приведут к качественному результату. Так как важна скорость отрисовки изображения на экране, то были выбраны именно данные алгоритмы.

На вход программе будут подаваться параметры шума Перлина, которые позволяют выводить различный вид ландшафта. Пользователю будет доступно изменять эти параметры, выполнять поворот и масштабирование сцены, а также увеличение размера исходной сцены. Возможно изменение положения точечного источника света. В результате программа должна выдавать сгенерированный трехмерный ландшафт. Программа должна корректно реагировать на действия пользователя.

2 Конструкторский раздел

В данном разделе представлены требования к ПО, описание работы алгоритмов построения и визуализации трехмерной сцены; приведена структура программного обеспечения.

2.1 Требования к программному обеспечению

Программа должна удовлетворять следующим требованиям:

- характеристики генерирования ландшафта задаются пользователем через графический интерфейс;
- расположение источника света задается через интерфейс;
- размер сцены, а также поворот и масштабирование ландшафта задаются в интерфейсе.

2.2 Описание работы алгоритмов

На рисунке 2.1 представлена схема алгоритма построения 3D изображения на экране.

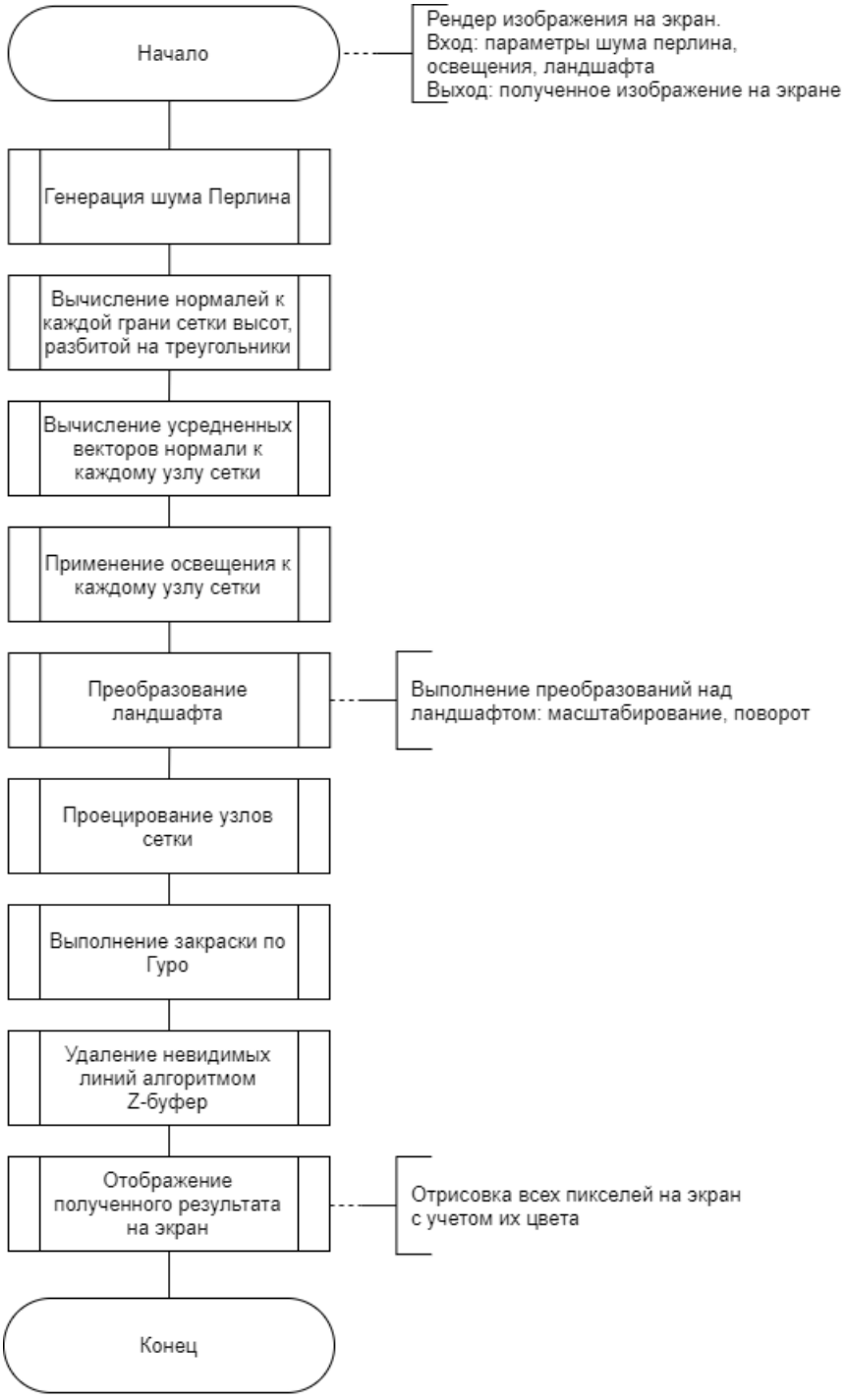

Рисунок 2.1 – Схема алгоритма построения 3D изображения на экране

На рисунках 2.2, 2.3 и 2.4 представлены схемы алгоритма генерирования карты высот на основе шума Перлина.

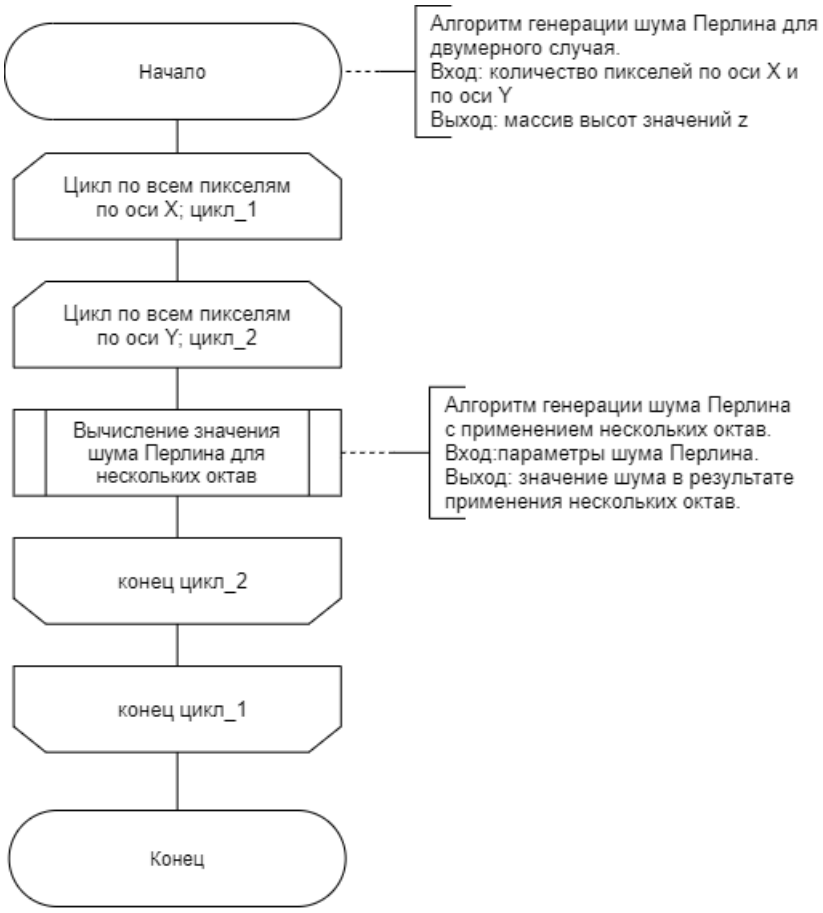

Рисунок 2.2 – Схема алгоритма построения шума Перлина для двумерного случая

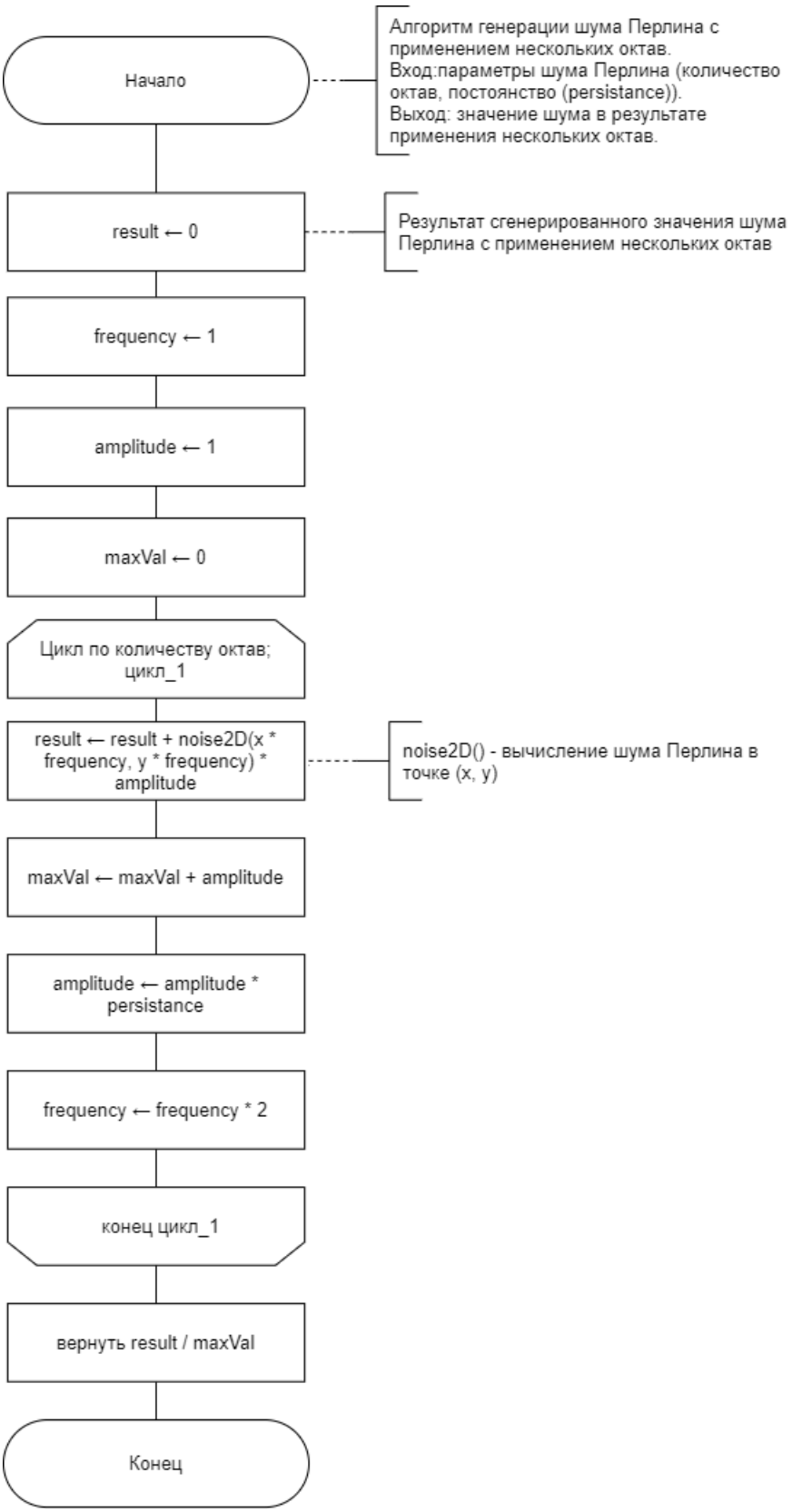

Рисунок 2.3 – Схема алгоритма шума Перлина в зависимости от числа октав

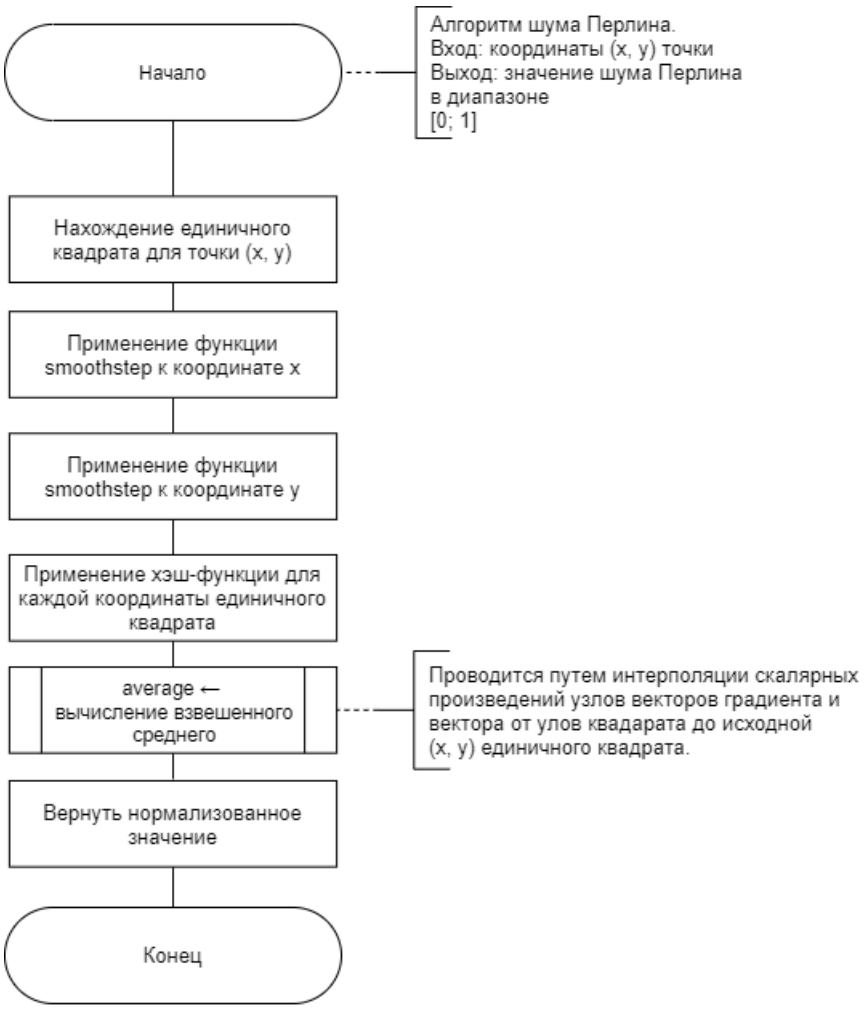

Рисунок 2.4 – Схема алгоритма шума Перлина

На рисунке 2.5 представлена схема алгоритма закраски Гуро.

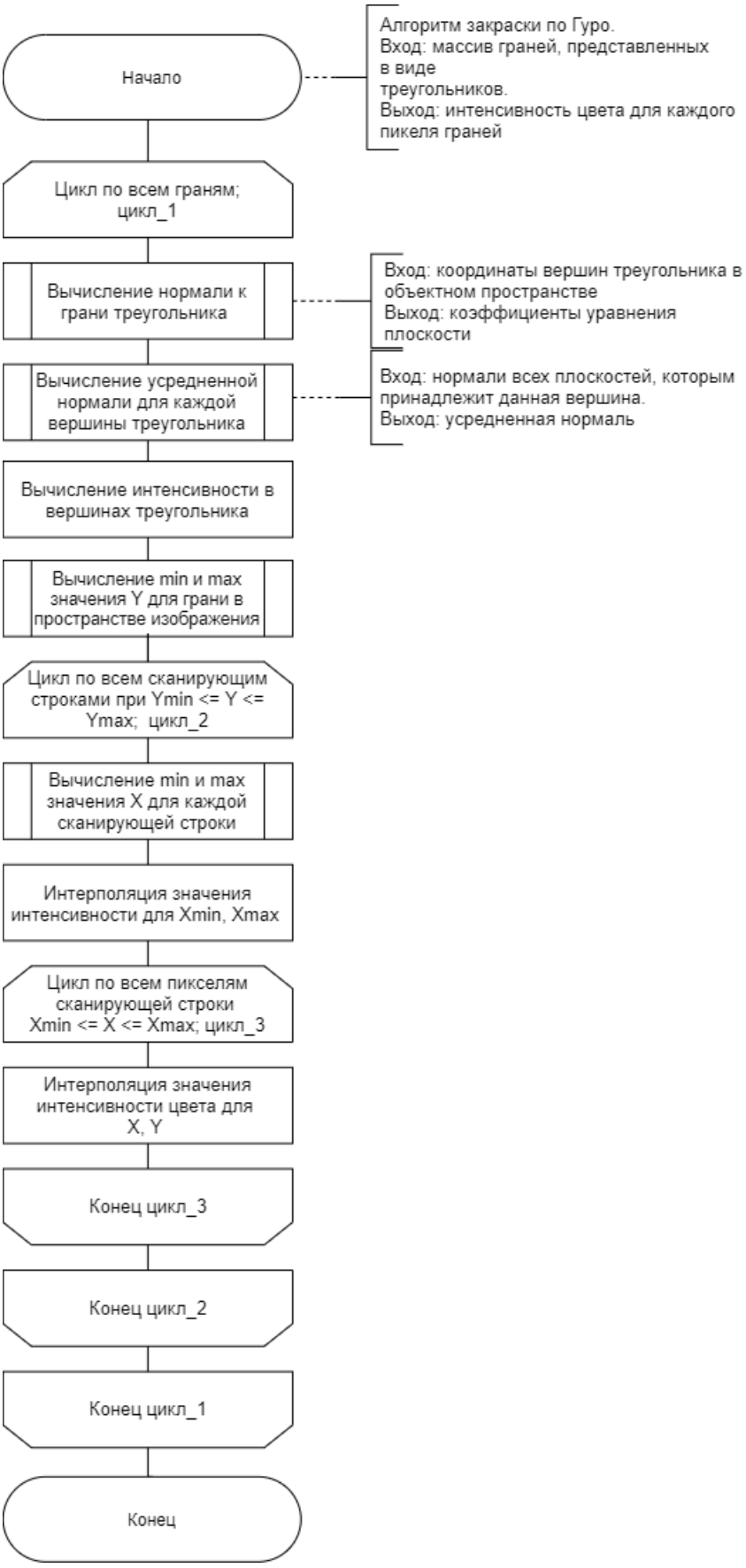

Рисунок 2.5 – Схема алгоритма закраски Гуро

На рисунке 2.6 представлена схема алгоритма Z-буфер.

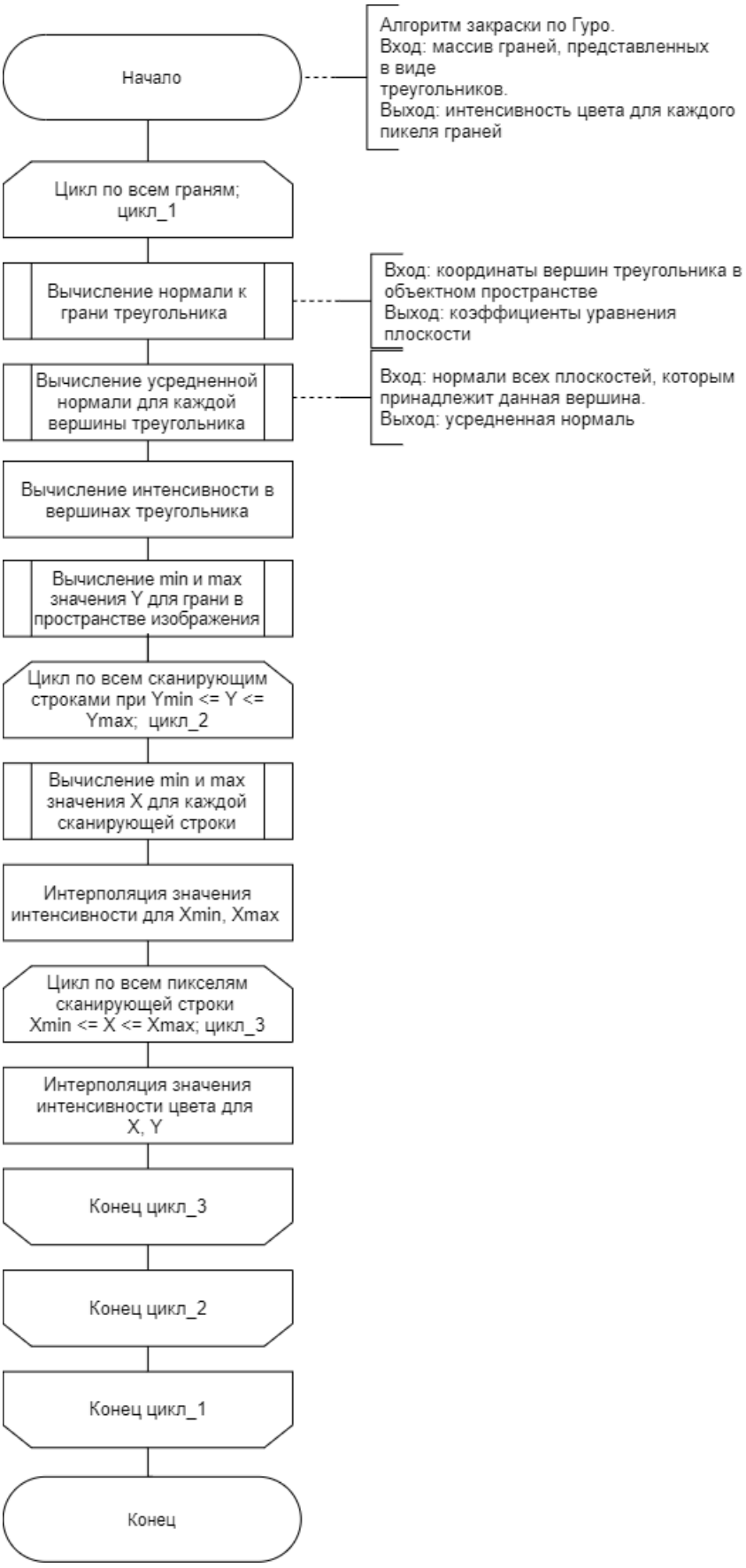

Рисунок 2.6 – Схема алгоритма Z-буфер

На рисунках 2.7 и 2.8 представлены схемы алгоритма Z-буфер в сочетании с закраской Гуро.

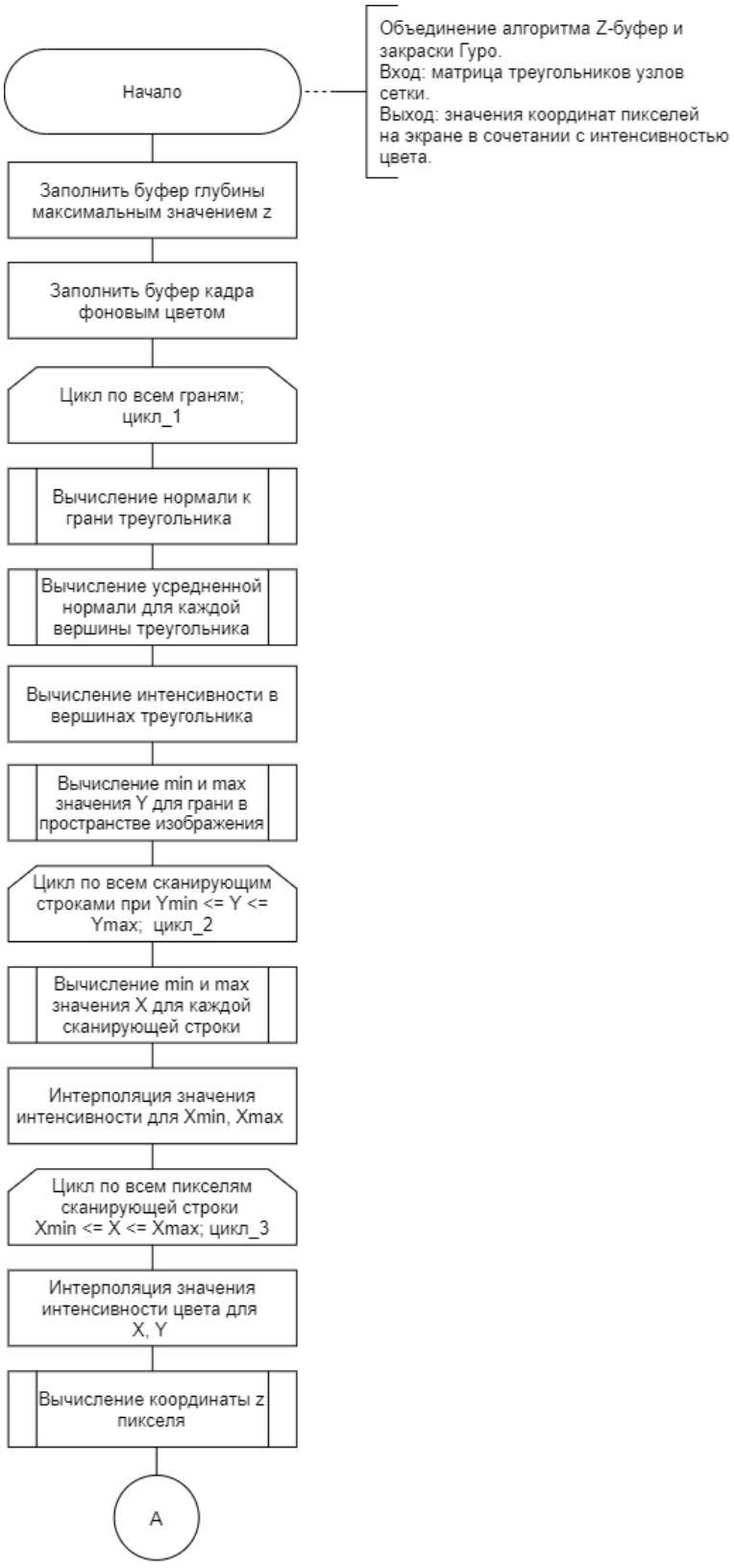

Рисунок 2.7 – Схема алгоритма Z-буфер в сочетании с закраской Гуро

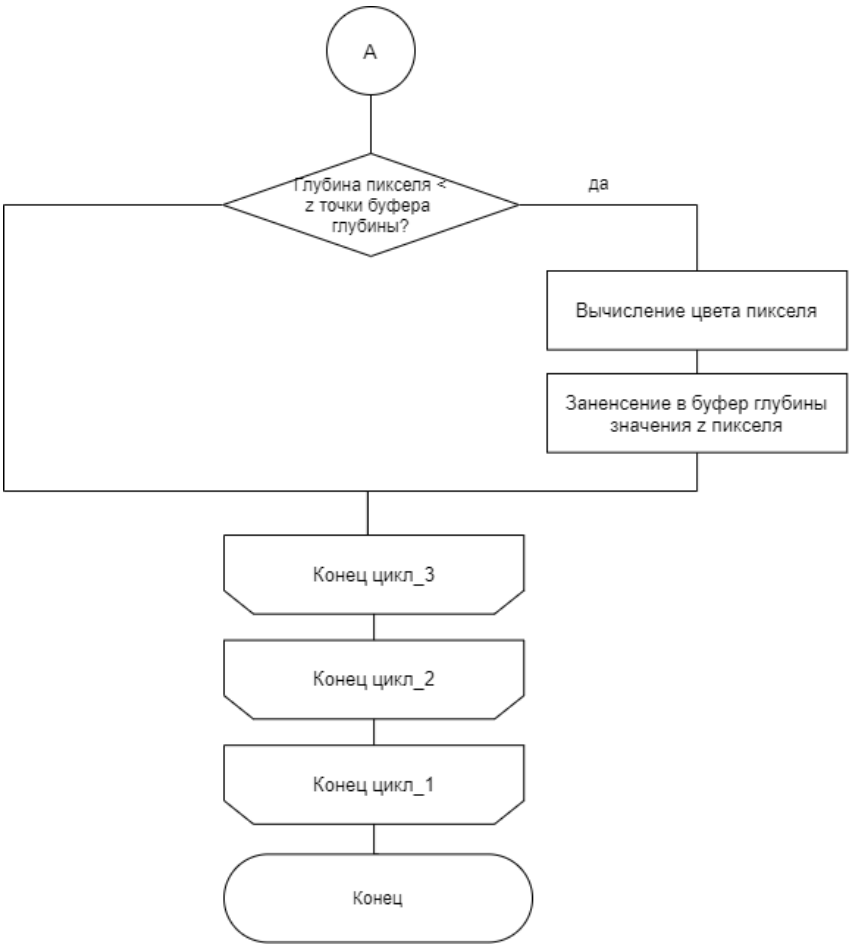

Рисунок 2.8 – Схема алгоритма Z-буфер в сочетании с закраской Гуро

2.3 Шум Перлина

Пусть (i,j) обозначают четыре точки на этом квадрате, где i - набор нижних и верхних ограничивающих целых чисел на $x : \{|x|, |x| + 1\}$, и аналогично $j = \{|y|, |y| + 1\}$. Четыре градиента задаются через $g_{i,j} = G[P[P[i] + j]]$, где предварительно вычисленные массивы P и G содержат, соответственно, псевдослучайную перестановку и псевдослучайную единичный вектор градиента. Четыре линейные функции $g_{i,j}(x_i, y_j)$ затем билинейно интерполируются с помощью $s(x - |x|),\ s(y - |y|), s(t) = 6t^5 - 15t^4 + 10t^3$.

2.4 Структура программного обеспечения

В качестве парадигмы было использовано объектно-ориентированное программирование. На рисунке 2.9 представлена диаграмма классов ПО.

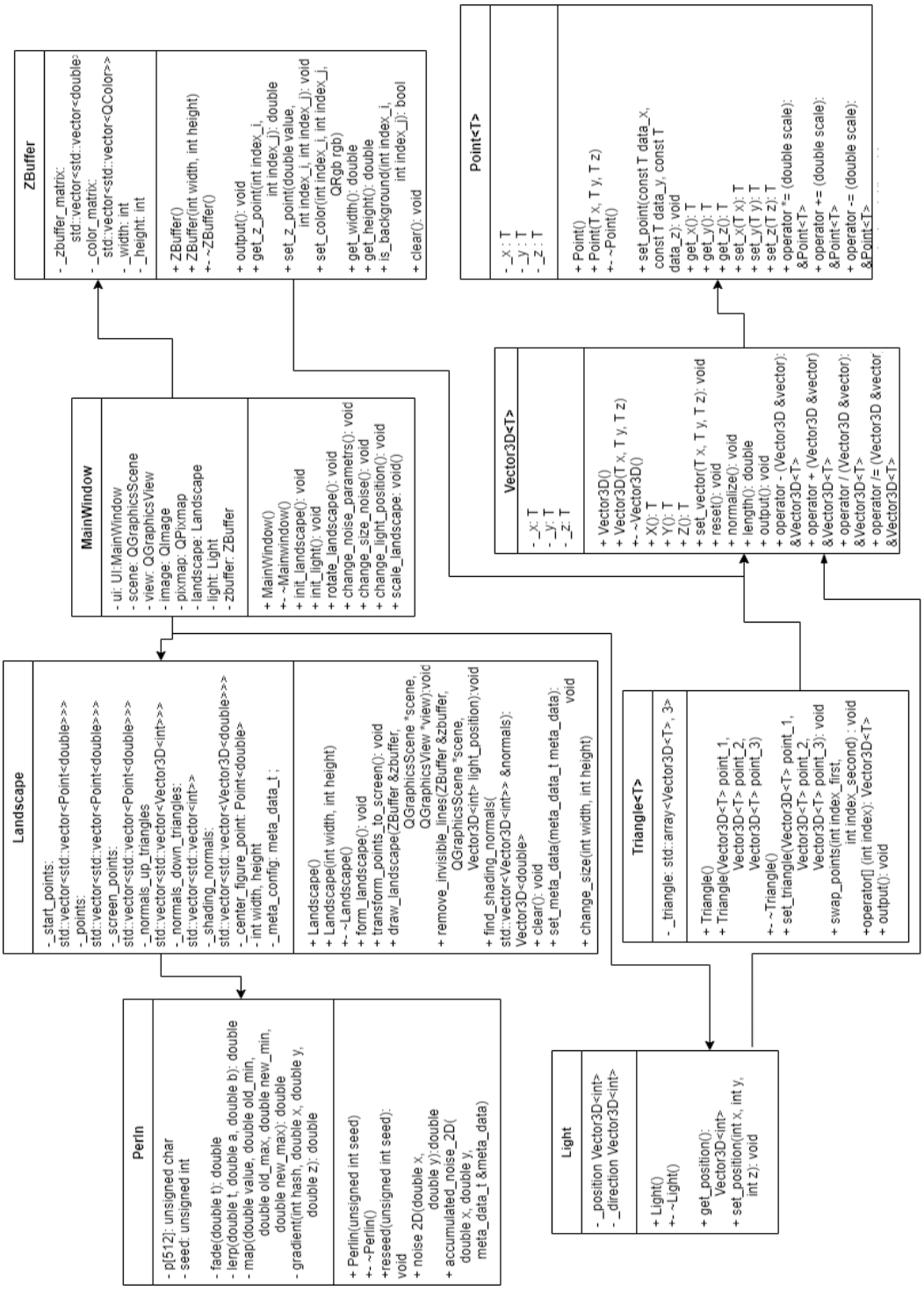

Рисунок 2.9 – Диаграмма классов

Были спроектированы следующие классы:

- Perlin - хранит таблицу перестановок p и значение "зерна" seed;
- Light - хранит позиию точечного источника освещения и его направление до точки (x, y);
- Landscape - хранит объектные точки, экранные точки, нормали к треугольникам, усредненные нормали, координату центра ландшафт, линейные размеры ландшафта и параметры шума Перлина;
- ZBuffer - хранит информацию о буфеер кадра и буфере глубины.

2.5 Описание оптимизации временных характеристик

Из схемы алгоритмов 2.5 и 2.6 следует, что каждый раз выполняется обработка всех пикселей карты высот. В связи с этим будет целесообразней совместить алгоритм Z-буфер и закраски Гуро как на рисунках 2.7 и 2.8.

2.6 Использование матричных фильтров

Для более гибкой работы с ландшафтом стоит учесть такой параметр как «редактирование». Этот этап позволит пользователю изменять желаемые настройки при работе с ландшафтом. Например, при использовании матричного фильтра, который может использовать матрицу размером $m \times n$, элементы которой соответствуют коэффициентам увеличения высоты в отдельной области ландшафта.

2.7 Использование стохастических алгоритмов

Стохастичность означает случайность. Стохастические алгоритмы могут вносить случайный фактор в формирование положения сетки ландшафта в мировом пространстве. Например, они могут изменять исход-

ный ландшафт при одинаковых параметрах таким образом, чтобы каждый раз выводился совершенно новый результат.

Например, можно при помощи случайности можно задавать настройки шума Перлина так, чтобы можно было выводить различные изображения на экране. Другим применением таких алгоритмов можно задавать положение ландшафта в пространстве путем переопределения его местоположения в пространстве или же вывод на экран только небольшой части ландшафта.

2.8 Вывод

Были описаны алгоритмы визуализации сцены и генерирования ландшафта предложен метод оптимизации. Было спроектировано программное обеспечение и представлены требования к ПО.

3 Технологический раздел

В данном разделе приведены средства реализации ПО, листинги реализованных алгоритмов, описание интерфейса.

3.1 Выбор средств реализации

В качестве языка программирования, на котором реализовано программное обеспечение, был выбран C++ [8]. Этот язык поддерживает объектно-ориентированную модель разработки, что позволяет четко структурировать программу и легко модифицировать отдельные ее компоненты независимо от других. Важной особенностью языка C++ является его высокая вычислительная производительность наряду с другими языками программирования. Для измерения процессорного времени была использована библиотека chrono [9].

В качестве среды разработки была выбрана среда Qt Creator по следующим причинам [10]:

- поддерживает фреймворк Qt, необходимый для создания графического пользовательского интерфейса и работы с ним;
- обладает функционалом, необходимым для написания, профилирования и отладки программ;
- имеет встроенный редактор QT Design для графического интерфейса.

3.2 Листинги программ

На листинге 3.1 разработанного метода удаления невидимых ребер и граней.

Листинг 3.1 – Программный код алгоритма удаления невидимых ребер и граней.

```
void Landscape::remove_invisible_lines(ZBuffer &zbuffer, QGraphicsScene
    *scene, Vector3D<int> light_position)
{
    plane_koeffs_t plane_koeffs_up, plane_koeffs_down;

    std::vector<std::vector<rasterised_points_t>> rasterized_points_up,
        rasterized_points_down;

    Vector3D<double> normal_up, normal_down;

    Triangle<double>triangle_up_normals, triangle_down_normals;

    Triangle<double>triangle_up_3d, triangle_down_3d;

    Point<int> middle_point_up, middle_point_down;

    for (int i = 0; i < _width - 1; i++){
        for (int j = 0; j < _height - 1; j++)
        {
            calculate_equation_plane(plane_koeffs_up, _points[i][j],
                _points[i][j+1], _points[i + 1][j + 1]);
            calculate_equation_plane(plane_koeffs_down, _points[i][j],
                _points[i+1][j], _points[i + 1][j + 1]);

            triangle_up_normals.set_triangle(_shading_normals[i][j],
                _shading_normals[i][j+1], _shading_normals[i+1][j+1]);
            triangle_down_normals.set_triangle(_shading_normals[i][j],
                _shading_normals[i+1][j], _shading_normals[i+1][j+1]);

            middle_point_up = rasterize_triangle(rasterized_points_up,
                triangle_up_normals, light_position,
            _screen_points[i][j], _screen_points[i][j+1], _screen_points
                [i + 1][j + 1],
            scene, zbuffer.get_color_matrix(), plane_koeffs_up);

            middle_point_down = rasterize_triangle(
                rasterized_points_down, triangle_down_normals,
                light_position,
            _screen_points[i][j], _screen_points[i+1][j], _screen_points
                [i + 1][j + 1],
            scene, zbuffer.get_color_matrix(), plane_koeffs_down);

            calculate_depth_pixels(zbuffer.get_zbuffer_matrix(), zbuffer
                .get_color_matrix(),
            rasterized_points_up, plane_koeffs_up, light_position,
                triangle_up_normals,
```

```
34              triangle_up_3d);
35              calculate_depth_pixels(zbuffer.get_zbuffer_matrix(), zbuffer
                    .get_color_matrix(),
36              rasterized_points_down, plane_koeffs_down, light_position,
                    triangle_down_normals,
37              triangle_down_3d);
38
39              rasterized_points_up.clear(), rasterized_points_down.clear()
                    ;
40          }
41      }
42  }
```

На листинге 3.2 представлен код вычисления шума Перлина для нескольких октав.

Листинг 3.2 – Программный код вычисления шума Перлина для нескольких октав.

```
1  double accumulatedNoise2D(double x, double y, meta_data_t &meta_data)
2  {
3      double lacunarity = meta_data.lacunarity;
4      double gain = meta_data.gain;
5      int octaves = meta_data.octaves;
6
7      double amplitude = 1;
8      double frequency = 1;
9      double result = 0.0;
10     double maxVal = 0.0;
11
12     for (; octaves > 0; octaves--)
13     {
14         result += noise2D(x * frequency, y * frequency) * amplitude;
15         maxVal += amplitude;
16
17         amplitude *= gain;
18         frequency *= lacunarity;
19     }
20
21     double e = result / maxVal;
22     return e;
23 }
```

На листинге 3.3 представлен метод генерации шума Перлина.

Листинг 3.3 – Программный код генерации шума Перлина.

```
double noise2D(double x, double y)
{
    int xi = (int)(std::floor(x)) & 255;
    int yi = (int)(std::floor(y)) & 255;

    x -= std::floor(x);
    y -= std::floor(y);

    double sx = fade(x);
    double sy = fade(y);

    unsigned char aa, ab, ba, bb;
    aa = p[p[xi] + yi];
    ab = p[p[xi] + yi + 1];
    ba = p[p[xi + 1] + yi];
    bb = p[p[xi + 1] + yi + 1];

    double average = lerp(sy, lerp(sx,
                                   gradient(aa, x, y, 0),
                                   gradient(ba, x - 1, y, 0)),
                              lerp(sx,
                                   gradient(ab, x, y - 1, 0),
                                   gradient(bb, x - 1, y - 1, 0)));

    return map(average, -1, 1, 0, 1);
}
```

3.3 Интерфейс ПО

Интерфейс ПО представлен на рисунке 3.1.

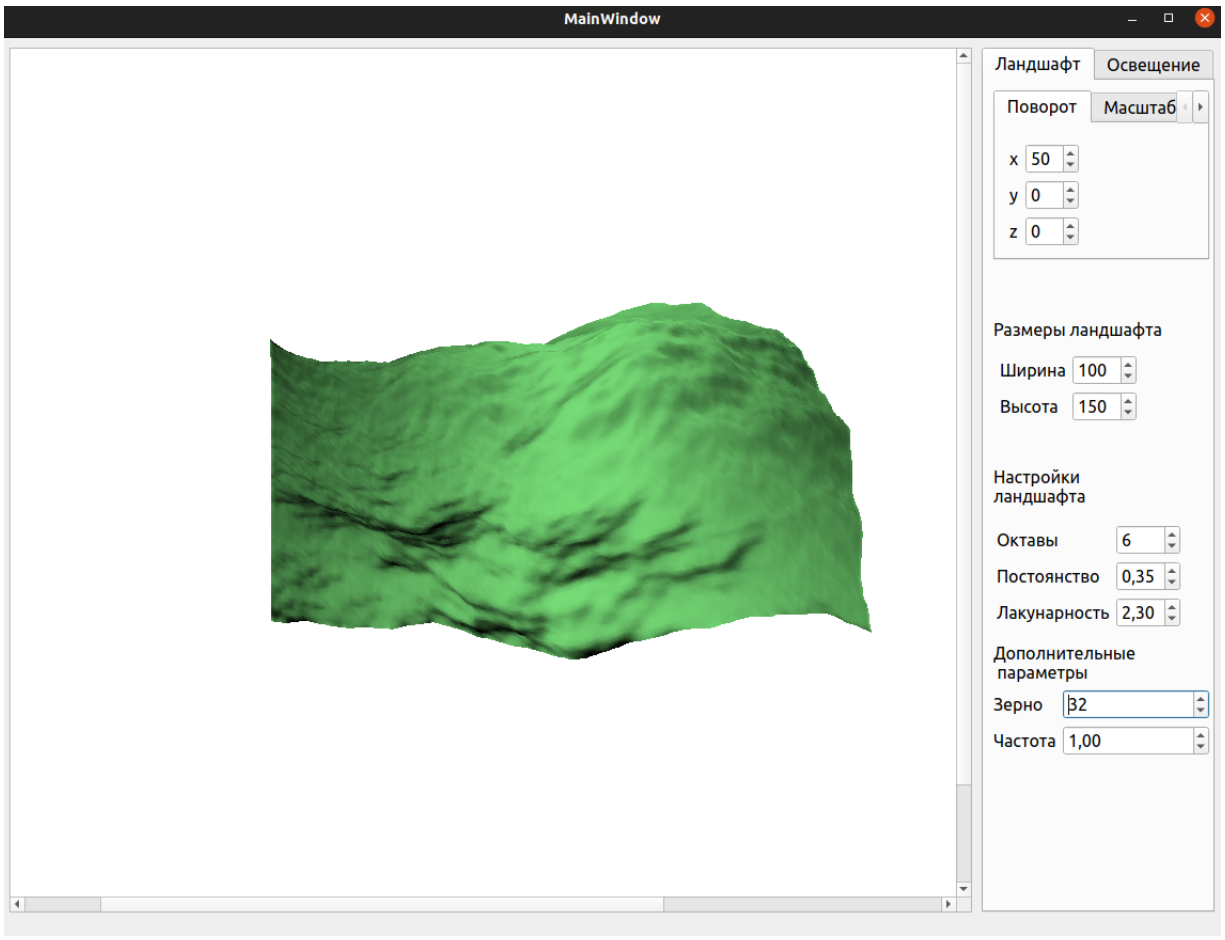

Рисунок 3.1 – Интерфейс ПО

Пользователь может изменять вид ландшафта следующим образом:

- при помощи изменения параметров ландшафта (октавы, постоянство, лакунарность);
- для получения новой модели можно задать дополнительный параметр «Зерно»;
- изменять положение точечного источника света;
- осуществить поворот и масштабирование ландшафта.

Настройка параметров приведена на рисунках 3.2, 3.3 и 3.4.

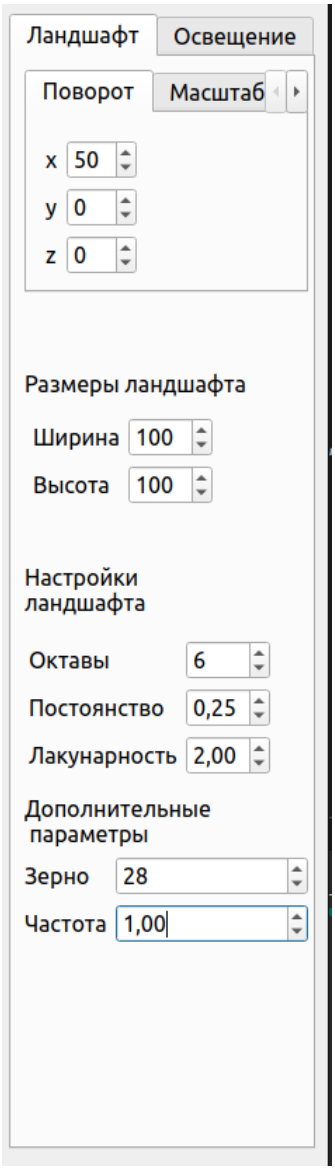

Рисунок 3.2 – Изменение настроек ландшафта

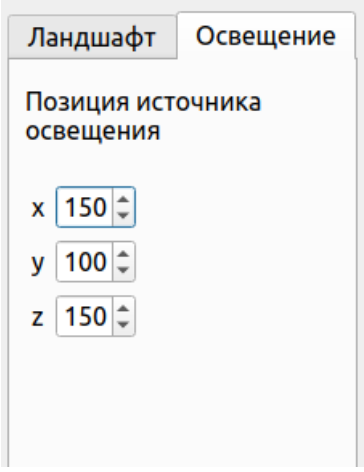

Рисунок 3.3 – Изменение позиции источника освещения

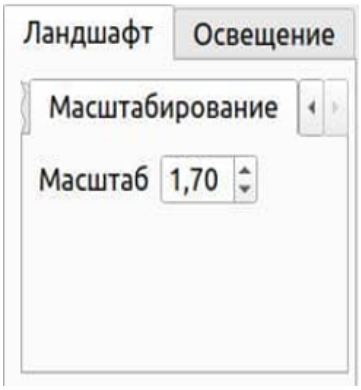

Рисунок 3.4 – Изменение масштаба ландшафта

3.4 Вывод

В данном разделе были выбраны средства реализации, среда разработки, приведены листинги модулей, представлен интерфейс ПО.

4 Исследовательский раздел

В разделе представлены результаты проведенного эксперимента, в котором сравниваются временные характеристики работы реализованного программного обеспечения.

4.1 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

- операционная система: Ubuntu 21.10;
- память: 8 ГГц;
- процессор: Intel(R) Core(TM) i5-8265U CPU @ 1.60 ГГц 1.80 ГГц.

Во время тестирования устройство было подключено к блоку питания и не нагружено никакими приложениями, кроме встроенных приложений окружения и самим окружением.

4.2 Апробация

На рисунках 4.1 - 4.5 представлены результаты синтеза изображения с различными параметрами ландшафта и освещенности сцены.

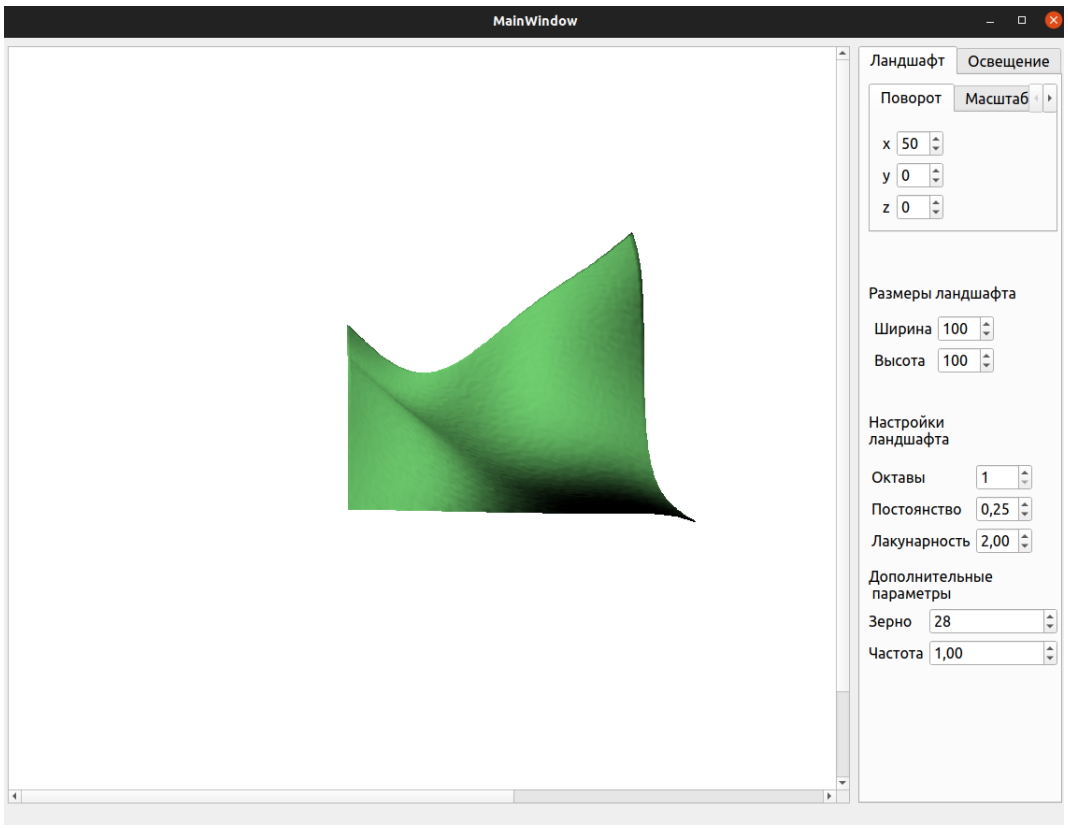

Рисунок 4.1 – Визуализация сцены в обычном режиме

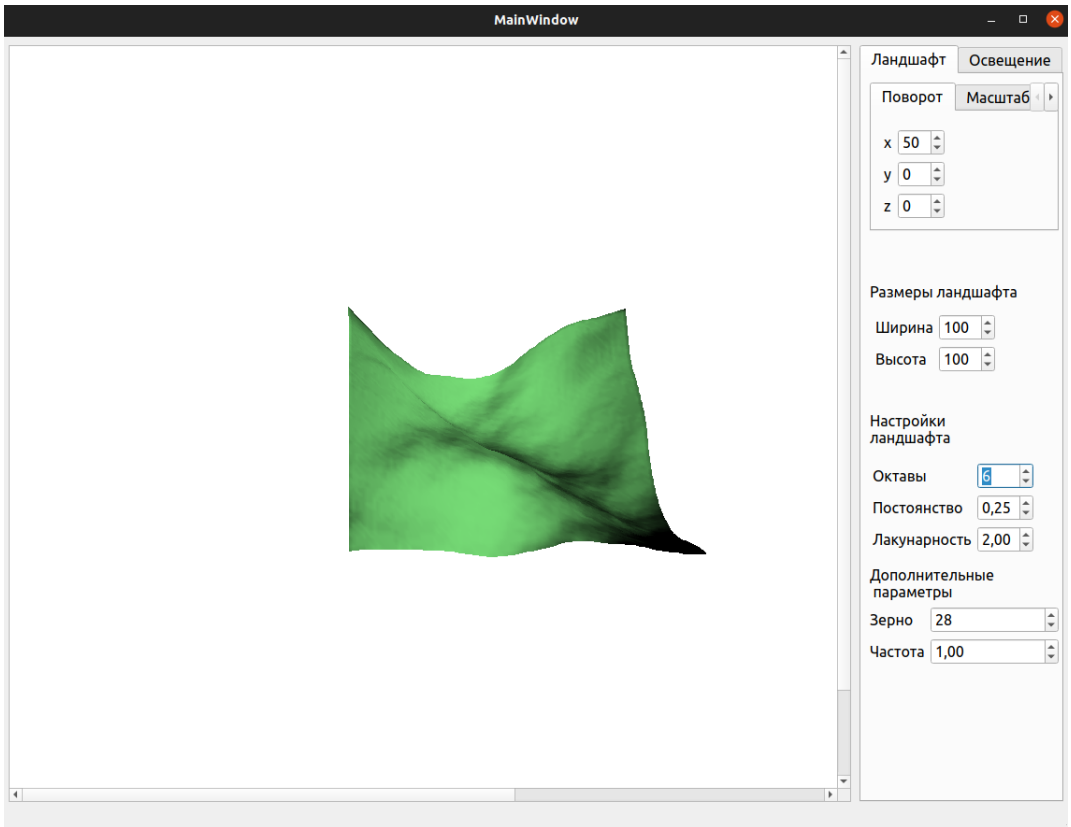

Рисунок 4.2 – Применение нескольких октав к исходному ландшафту

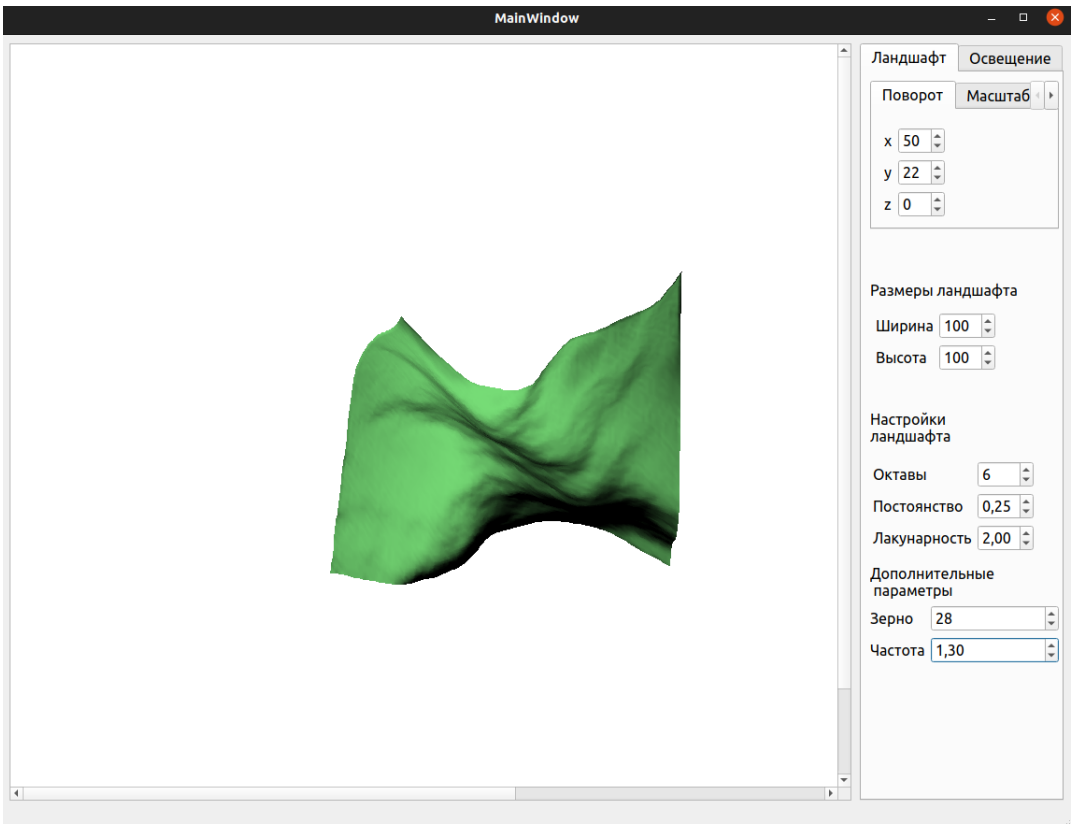

Рисунок 4.3 – Изменение частоты исходного ландшафтаы

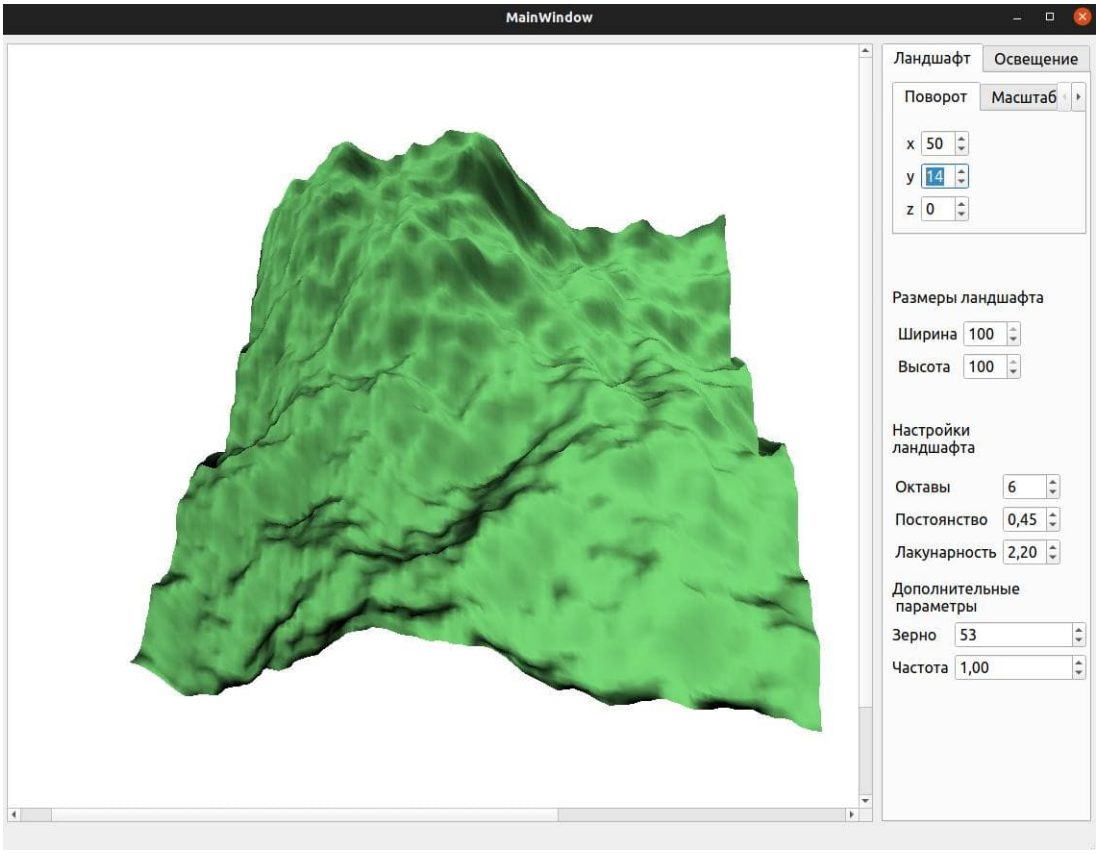

Рисунок 4.4 – Изменение параметра "зерно" ландшафта

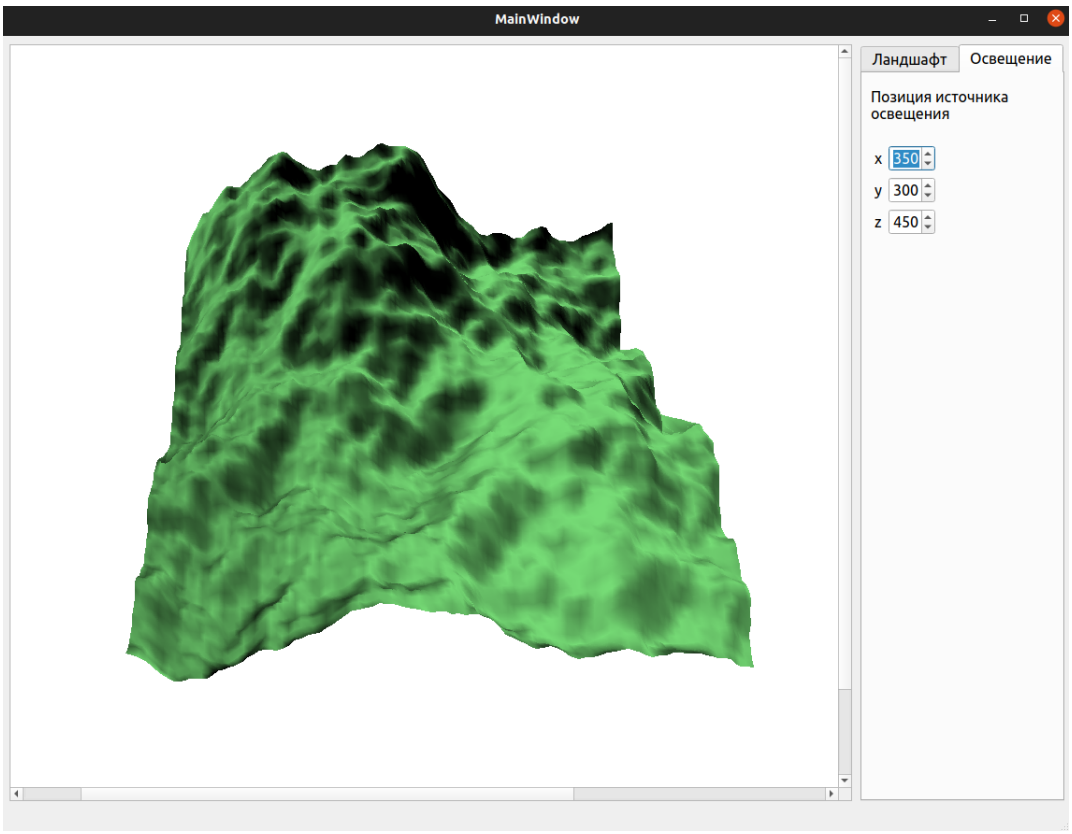

Рисунок 4.5 – Изменение позиции источника освещения

4.3 Постановка первого эксперимента

Целью эксперимента является определение зависимости времени отрисовки ландшафта от количества полигонов сетки, разбитых на треугольники. Эксперимент проводился на ландшафте, имеющем размер 10×10, 25×25, 50×50, 100×100, 150×150, 250×250, 350×350, 500×500.

Для снижения погрешности измерений один и тот же эксперимент будет проведен в количестве N=10, после что будет вычислять среднеарифметическое значение измеряемой величины.

4.4 Результат первого эксперимента

В таблице 4.1 приведены экспериментально полученные значения временных характеристик отрисовки ландшафта.

Таблица 4.1 – Таблица времени визуализации ландшафта в зависимости от его линейного размера (в секундах).

Размер	Время
10	0.336509
25	0.32979
50	0.342097
100	0.387329
150	0.468589
250	0.721446
350	1.06574
500	2.02038
700	19.6004

На рисунке 4.6 приведена зависимость времени отрисовки изображения от линейного размера ландшафта (в секундах).

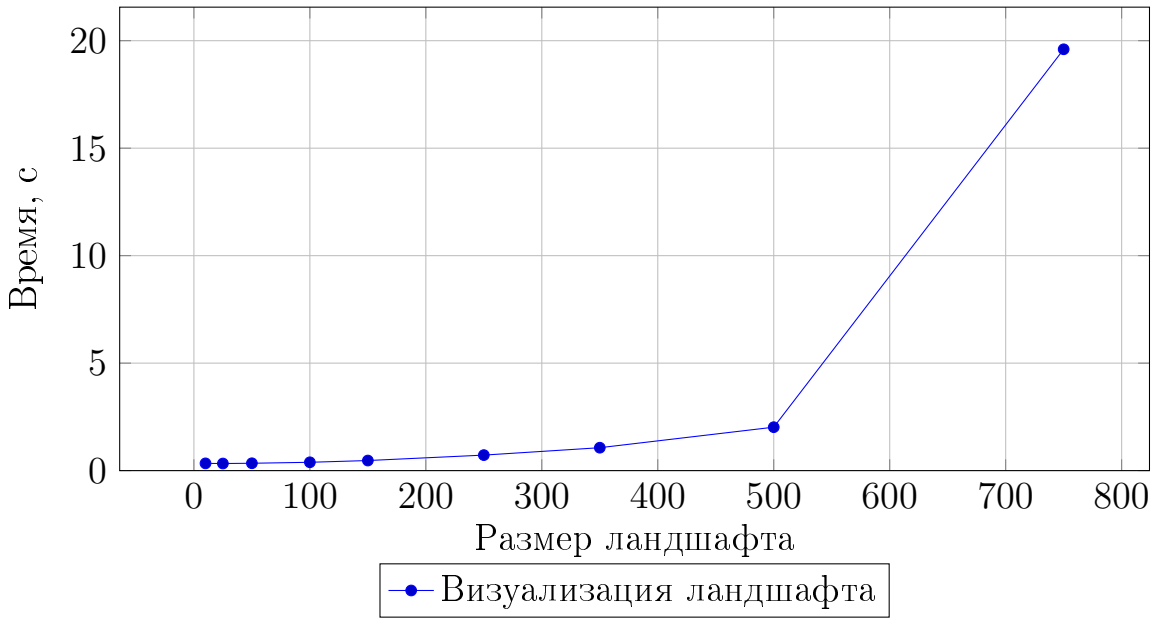

Рисунок 4.6 – Зависимость времени отрисовки ландшафта от его линейного размера.

4.5 Постановка второго эксперимента

Целью эксперимента является определение скорости формирования карты высот на основе шума Перлина в зависимости от числа октав для размера ландшафта 100×100. Диапазон значений октав берется от 1 до 8. Большее число октав не рассматривается, так как пользователю будет

уже неразличим отрисованный на экране результат.

4.6 Результат второго эксперимента

В таблице 4.1 приведены экспериментально полученные значения временных характеристик формирования карты высот в зависимости от числа октав.

Таблица 4.2 – Таблица времени формирования карты высот в зависимости от числа октав (в секундах).

Октавы	Время
1	0.000583146
2	0.00114126
3	0.00153122
4	0.0021004
5	0.00270201
6	0.00337686
7	0.0042514
8	0.00541033

На рисунке 4.7 приведена зависимость времени формирования карты высот от числа октав (в секундах).

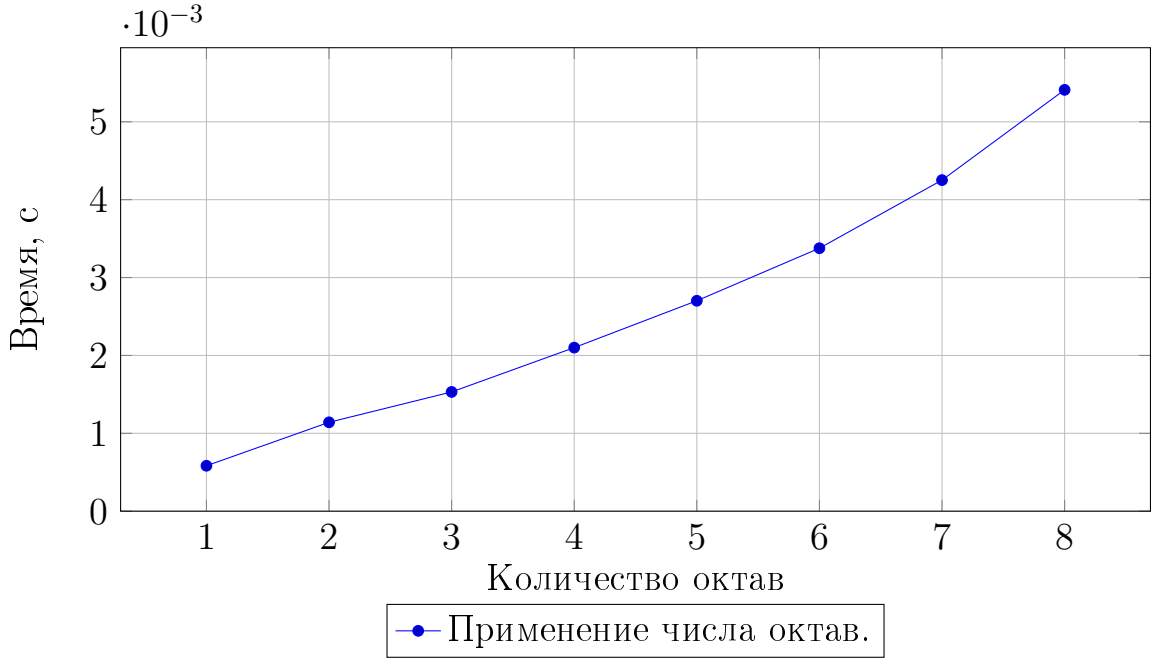

Рисунок 4.7 – Зависимость времени работы алгоритма шума Перлина от числа октав.

4.7 Вывод

По результатам эксперимента можно сделать вывод о том, что формирование карты высот на основе шума Перлина с использованием нескольких октав требует разные временные затраты. Время выполнения программы для 4 октав медленнее времени выполнения для 1 октавы в 3,6 раз. Для количества октав 4 и 8 эта разница составляет 2,57 раз. При этом общее время визуализации ландшафта начинает увеличиваться с размера 500×500. Общее время для размера 700×700 отличается уже в 9,68 раз.

Заключение

Цель данной курсовой работы достигнута - было написано программное обеспечение, выполняющее визуализацию трехмерного ландшафта на основе шума Перлина. Пользователь может задавать расположение источника точечного света, выполнять редактирование ландшафта и изменять размеры сцены.

В ходе работы были выполнены следующие задачи:

- выполнена формализация объектов синтезируемой сцены;
- проведено исследование существующих алгоритмов решения поставленной цели;
- выбраны и описаны подходящие алгоритмы для визуализации сцены;;
- приведены схемы используемых алгоритмов;
- описаны использующиеся структуры данных;
- определены средства реализации ПО;
- реализовано ПО;
- выполнено исследование временных характеристик алгоритмов визуализации сцены.

Разработанный программный продукт позволяет отображать на экране трехмерный ландшафт на основе шума Перлина. В дальнейшем ПО можно улучшить, добавив параллельную реализацию удаления невидимых линий и поверхностей, закраски Гуро; добавить возможность пользователю работать с несколькими ландшафтами и точечно изменять внешний вид.

Список литературы

[1] Генерация трехмерных ландшафтов [Электронный ресурс]. - Режим доступа: https://www.ixbt.com/video/3dterrains-generation.shtml (дата обращения: 20.10.2021)

[2] Шум Перлина [Электронный ресурс]. - Режим доступа: https://web.archive.org/web/20160308191327/http://noisemachine.com/talk1/2.html (дата обращения: 28.10.2021)

[3] Proceedings Template [Электронный ресурс]. - Режим доступа: https://mrl.cs.nyu.edu/ perlin/paper445.pdf (дата обращения: 28.10.2021)

[4] The Perlin noise math FAQ [Электронный ресурс]. - Режим доступа: https://web.archive.org/web/20150607183420/http://webstaff.itn.liu.se/šstegu/TNM022-2005/perlinnoiselinks/perlin-noise-math-faq.html (дата обращения: 1.11.2021)

[5] Роджерс Д.Ф. "Алгоритмические основы машинной графики. (Procedural Elements for Computer Graphics)". в: Москва: Издательство "Мир". Редакция литературы по математическим наукам, 1989. гл. Алгоритм, использующий z-буфер, с. 321

[6] Роджерс Д.Ф. "Алгоритмические основы машинной графики. (Procedural Elements for Computer Graphics)". в: Москва: Издательство "Мир". Редакция литературы по математическим наукам, 1989. гл. Простая модель освещения, с. 380

[7] Роджерс Д.Ф. "Алгоритмические основы машинной графики. (Procedural Elements for Computer Graphics)". в: Москва: Издательство "Мир". Редакция литературы по математическим наукам, 1989. гл. Закраска мтодом Гуро, с. 391

[8] C++11. [Электронный ресурс], режим доступа: https://en.cppreference.com/w/cpp/11 (дата обращения 12.11.2021)

[9] Data and time utilities. [Электронный ресурс], режим доступа: https://en.cppreference.com/w/cpp/chrono (дата обращения 20.11.2021)

[10] Qt 5.12. [Электронный ресурс], режим доступа: https://doc.qt.io/qt-5.12/index.html (дата обращения: 19.11.2021)