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 КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Генератор трехмерного ландшафта»

Студент ИУ7-53Б (Группа) _____ (Подпись, дата) Лысцев Н. Д. (И. О. Фамилия)

Руководитель курсовой работы _____ (Подпись, дата) Филлипов М. В. (И. О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время технологии трехмерной графики стремительно развиваются. Одним из направлений применения этих технологий является создание видеоигр. Наибольшую популярность набирают игры с так называемым открытым миром.

Основой для открытого мира служит трехмерный ландшафт значительных размеров, который для реалистичности должен быть разнообразным и детализированным. Традиционные методы ручного моделирования ландшафта являются крайне громоздкими, трудоемкими и ограниченными в своей вариативности.

Возникает потребность в создании программного обеспечения, которое бы позволяло автоматизировать процессы создания реалистичного ландшафта, чтобы ускорить разработку игр и обеспечить большую степень креативной свободы для разработчиков.

Целью работы является разработка программного обеспечения для генерации и визуализации трехмерного ландшафта.

Для достижения желаемых результатов необходимо решить следующие задачи:

1) Выполнить формализацию объектов синтезируемой сцены;
2) Провести анализ существующих алгоритмов создания ландшафта и визуализации сцены;
3) Выбрать подходящие алгоритмы для решения поставленной задачи.
4) Реализовать выбранные алгоритмы.

1 Аналитический раздел

В данном разделе дано формальное описание объектов синтезируемой сцены, проанализированы разные способы представления данных о ландшафте, рассмотрены алгоритмы процедурной генерации ландшафта, алгоритмы удаления невидимых линий и поверхностей, алгоритмы закраски и модели освещения.

1.1 Формализация объектов синтезируемой сцены

Сцена состоит из следующих объектов:

– ландшафт – трехмерная модель, описываемая полигональной сеткой [**lands**];

– источник света – материальная точка, испускающая лучи света.

1.2 Анализ способов представления данных о ландшафте

Существует несколько основных принципов представления данных для хранения информации о ландшафте [1]:

– регулярная сетка высот (карта высот);

– иррегулярная сетка вершин и связей, их соединяющих;

– посегментная карта высот.

1.2.1 Регулярная сетка

Данные представлены в виде двумерного массива [2]. Каждый элемент массива имеет свои индексы $[i, j]$, являющиеся координатами расположения точки на плоскости. Для каждой вершины с индексами $[i, j]$ в двумерном массиве определяется соответствующее значение высоты h_{ij}. На рисунке 1.1 показан пример представления ландшафта с помощью карты высот.

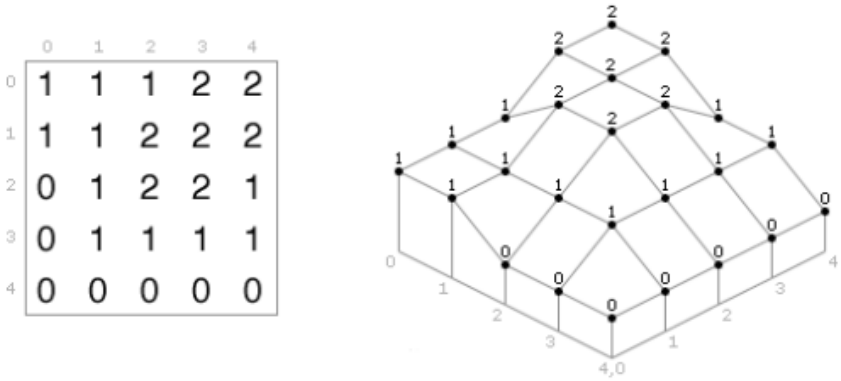

Рисунок 1.1 – Пример представления ландшафта с помощью карты высот [2]

К преимуществам данного способа можно отнести наглядность и простоту изменения данных, легкость нахождения координат и высоты на карте, возможность более точно производить динамическое освещение. Однако, у такого метода есть один существенный недостаток – избыточность данных (например, при задании плоскости) [1].

1.2.2 Иррегулярная сетка

Иррегулярная сетка позволяет размещать точки или узлы в произвольных местах в зависимости от необходимости и особенностей ландшафта.

У такого метода есть несколько существенных недостатков [1]:

- многие алгоритмы построения ландшафтом предназначены для регулярных сеток высот, поэтому оптимизация этих алгоритмов под иррегулярную сетку потребует значительных усилий и времени;
- из-за неравномерного расположения вершинных точек друг к другу возникает сложность при создании динамического освещения;
- сложности при хранении, модификации и просмотре такого ландшафта.

К плюсам данного способа можно отнести использование меньшей информации для построения ландшафта [1].

1.2.3 Посегментная карта высот

Суть этого метода заключается в разделении изначальной области на небольшие участки (например, квадратной формы), и генерации высот на каждом участке отдельно [3].

К преимуществам данного способа можно отнести возможность представления больших открытых пространств, возможность хранения информации о других объектах (строения, пещеры, скалы), возможность создания нескольких вариантов одного и того же сегмента, но с разным уровнем детализации. К недостаткам можно отнести проблему стыковки разных сегментов, неочевидность представления данных, проблему модификации этих данных [1].

1.3 Анализ алгоритмов процедурной генерации ландшафта

Основным критерием выбора алгоритма будет качество получаемого ландшафта, поскольку для решения задачи необходимо создавать правдоподобный рельеф.

1.3.1 Алгоритм Diamond-Square

Данный алгоритм является расширением одномерного алгоритма *midpoint displacement* [4] на двумерную плоскость. Алгоритм $Diamond - Square$ [5] выполняется рекурсивно и начинает работу с двумерного массива размера $2^n + 1$. В четырёх угловых точках массива устанавливаются начальные значения высот. Шаги *diamond* и *square* выполняются поочередно до тех пор, пока все значения массива не будут установлены.

1) Шаг *diamond* – для каждого квадрата в массиве, устанавливается срединная точка, которой присваивается среднее арифметическое из четырёх угловых точек плюс случайное значение.

2) Шаг *square* – берутся средние точки граней тех же квадратов, в которые устанавливается среднее значение от четырёх соседних с ними по осям точек плюс случайное значение.

На рисунке 1.2 показаны последовательность действий алгоритма $Diamond - Square$ на примере массива 5x5.

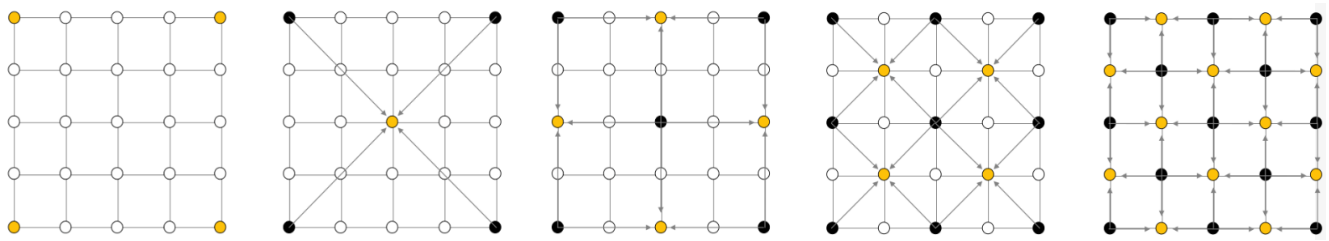

Рисунок 1.2 – Шаги, проходимые алгоритмом Diamond-Square на примере массива 5x5 [6]

К преимуществам данного алгоритма можно отнести простоту его реализации и возможность генерации различных ландшафтов с разнообразной детализацией.

К недостаткам можно отнести создание вертикальных и горизонтальных «складок» на краях карты из-за наиболее значительного возмущения, происходящего в прямоугольной сетке [6], а также сложности с контролем получаемого ландшафта.

1.3.2 Холмовой алгоритм

Это простой итерационный алгоритм, основанный на нескольких входных параметрах. Алгоритм изложен в следующих шагах [7]:

1) Создается и инициализируется нулевым уровнем двухмерный массив;

2) Берется случайная точка на ландшафте или около его границ (за границами) 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) Выполнение пунктов 1) - 3) n раз, где n – выбранное количество шагов.

5) Проведение нормализации ландшафта;

6) Проведение «долинизации» ландшафта.

К преимуществам данного алгоритма можно отнести простоту его реализации, а также возможность контроля «гористости» ландшафта за счет этапов нормализации и долинизации.

Данный алгоритм имеет несколько недостатков:

– с помощью этого алгоритма тяжело смоделировать склон холма, или горы, так как в процессе генерации ландшафта используются только гладкие полусферы [8];

– этот алгоритм неприменим, когда требуется детализировано отобразить лишь часть всего ландшафта [8].

1.3.3 Алгоритм шума Перлина

Это математический алгоритм по генерированию процедурной текстуры псевдо-случайным методом [9]. Этот алгоритм может быть реализован для n-мерного пространства, но чаще его используют для одно-, двух-, трехмерного случая.

Рассмотрим версию этого алгоритма для двумерного случая:

Для карты высот создается сетка точек и в каждой точке сетки генерируется псевдослучайный единичный вектор-направления градиента. Для каждой точки с координатами (x, y) из карты высот определяется, в какой ячейке сетки находится точка, генерируются вектора, идущие от точек ячейки сетки до этой точки. На рисунке 1.3 показан пример векторов для точки (x, y).

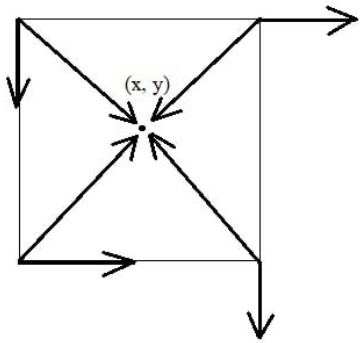

Рисунок 1.3 – Пример сгенерированных векторов для точки (x, y)

Вычисляются четыре скалярных произведения векторов-направлений градиента и вектора, идущего от связанной с ним точки сетки к точке с координатами (x, y).

Далее, чтобы получить значение высоты z в точке (x, y), необходимо провести двумерную интерполяцию на основе полученных скалярных произведений. Для создания плавного перехода между значениями предварительно производятся вычисления весов измерений по x и по y. Веса определяются функцией

$$smootherstep(t) = 6t^5 - 15t^4 + 10t^3 \tag{1.1}$$

где вместо t подставляются значения x и y. Далее, используя метод последовательной интерполяции по каждому измерению, получаем значения двух одномерных линейных интерполяций с использованием веса по x, и в конце проводим одну одномерную линейную интерполяцию с этими вычисленными значениями, но уже по весу измерения y. Полученное значение и будет высотой в данной точке (x, y) карты высот [10].

Чтобы контролировать генерацию шума, существует набор параметров:

- октавы – количество уровней детализации шума;
- лакунарность – множитель, который определяет изменение частоты с ростом октавы;
- стойкость – множитель частотной амплитуды, который определяет изменение амплитуды с ростом октавы.

Для достижения качественного детализированного ландшафта можно смешивать шумы разных частот и амплитуд.

Преимущества:

- комбинация различных октав в алгоритме шума Перлина позволяет добавлять дополнительные уровни детализации к сгенерированному шуму, тем самым повышая реалистичность и детализацию ландшафта;
- зная лишь параметры генерации, можно получить высоту в любой точке карты без необходимости знания высот в соседних точках карты.

Недостатком является низкая детализация и реалистичность сгенерированного ландшафта без использования механизма комбинации октав.

1.4 Анализ алгоритмов удаления невидимых линий и поверхностей

Главным требованием при выборе алгоритма будут высокая скорость работы, чтобы пользователю не приходилось ждать долгой загрузки изображения.

1.4.1 Алгоритм Робертса

Алгоритм Робертса работает в объектном пространстве только с выпуклыми телами. Если тело не является выпуклым, то его предварительно нужно разбить на выпуклые составляющие [11].

Этот алгоритм выполняется в 4 этапа:

1) Подготовка исходных данных – составление матрицы тела для каждого тела сцены;

2) Удаление ребер, экранируемых самим телом;

3) Удаление ребер, экранируемых другими телами;

4) Удаление линий пересечения тел, экранируемых самими телами и другими телами, связанными отношением протыкания.

Алгоритм работает только с выпуклыми телами, что является недостатком данного алгоритма. Также к недостатку можно отнести то, что вычислительная сложность теоретически растет как квадрат числа объектов. К преимуществам можно отнести высокую точность вычислений.

1.4.2 Алгоритм, использующий Z-буфер

Это один из простейших алгоритмов удаления невидимых поверхностей. Работает этот алгоритм в пространстве изображения[11].

В данном алгоритме используется два буфера: буфер кадра и z-буфер. Буфер кадра используется для заполнения атрибутов (интенсивности) каждого

пикселя в пространстве изображения. z-буфер – отдельный буфер глубины, используемый для запоминания координаты z или глубины каждого видимого пикселя в пространстве изображения [11].

Вначале в z-буфер заносятся минимально возможные значения z, а буфер кадра заполняется заполняются фоновым значением интенсивности или цвета. Затем каждый многоугольник преобразовывается в растровую форму. Суть работы алгоритма заключается в следующем: в процессе работы глубина (значение координаты z) каждого нового пикселя, который надо занести в буфер кадра сравнивается с глубиной того пикселя, который уже есть в z-буфере. Если новый пиксель оказывается расположен ближе к наблюдателю, то новый пиксель заносится в буфер кадра. При этом в z-буфер заносится глубина нового пикселя. Если сравнение дало противоположный результат, то никаких действий не производится.

К преимуществам данного алгоритма относится простота его реализации, возможность работы со сценами любой сложности, отсутствие необходимости в сортировке по приоритету глубины.

К недостаткам можно отнести большой объем требуемой памяти, трудоемкость и высокая стоимость устранения лестничного эффекта, трудоемкость реализации эффектов прозрачность и просвечивания.

1.4.3 Алгоритм обратной трассировки лучей

Наблюдатель видит объект посредством испускаемого источником света, который падает на этот объект и согласно законам оптики некоторым путем доходит до глаза наблюдателя. Алгоритм имеет такое название, потому что эффективнее с точки зрения вычислений отслеживать пути лучей в обратном направлении, то есть от наблюдателя к объекту.

Преимуществами данного алгоритма являются возможность его использования в параллельных вычислительных системах и высокая реалистичность получаемого изображения, а недостатком – большое количество вычислений и медленная работа алгоритма.

1.5 Анализ моделей освещения

В данном разделе будут рассмотрены две модели освещения: модель Ламберта и модель Фонга.

1.5.1 Модель освещения Ламберта

Модель Ламберта моделирует идеальное диффузное освещение. Считается, что свет при попадании на поверхность рассеивается равномерно во все стороны. При расчете такого освещения учитывается только ориентация поверхности (нормаль N) и направление на источник света (вектор L). Рассеянная составляющая рассчитывается по закону косинусов (закон Ламберта) [12].

1.5.2 Модель освещения Фонга

Модель Фонга – классическая модель освещения. Модель представляет собой комбинацию диффузной составляющей (модели Ламберта) и зеркальной составляющей и работает таким образом, что кроме равномерного освещения на материале может еще появляться блик. Местонахождение блика на объекте, освещенном по модели Фонга, определяется из закона равенства углов падения и отражения. Если наблюдатель находится вблизи углов отражения, яркость соответствующей точки повышается [12].

1.6 Анализ алгоритмов закраски

Методы закраски используются для затенения полигонов модели в условиях некоторой сцены, имеющей источники освещения. Существует три основных алгоритма, позволяющих закрасить полигональную модель.

1.6.1 Алгоритм простой закраски

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. При данной закраске все плоскости (в том числе и те, что аппроксимируют фигуры вращения), будут закрашены однотонно, что в случае с фигурами

вращения будет давать ложные ребра [11].

К преимуществам можно отнести простоту реализации алгоритма и его быстродействие. В качестве недостатков можно выделить нереалистичность результата.

1.6.2 Алгоритм закраски по Гуро

Данный алгоритм позволяет получить более сглаженное изображение. Это достигается благодаря тому, что разные точки грани закрашиваются разным значением интенсивности.

Алгоритм состоит из следующих шагов:

1) Вычисление векторов нормалей к каждой грани;
2) Вычисление векторов нормали к каждой вершине грани путем усреднения нормалей примыкающих граней;
3) Вычисление интенсивности в вершинах грани в соответствии с выбранной моделью освещения;
4) Выполнение линейной интерполяции интенсивности вдоль ребер;
5) Выполнение линейной интерполяции вдоль сканирующей строки.

Преимуществами данного алгоритма являются хорошее сочетание с моделью освещения с диффузным отражением, а также более высокая, по сравнению с алгоритмом простой закраски, реалистичность получаемого изображения.

К недостаткам можно отнести отсутствие учета кривизны поверхности [11].

1.6.3 Алгоритм закраски по Фонгу

В алгоритме закраски по Фонгу используется билинейная интерполяция не интенсивностей в вершинах полигона, а билинейная интерполяция векторов нормалей. Благодаря такому подходу изображение получается более реалистичным. Однако для достижения такого результата требуется больше вычислительных затрат [11].

Данный алгоритм состоит из следующих шагов:

1) Вычисление векторов нормалей к каждой грани;

2) Вычисление векторов нормали к каждой вершине грани путем усреднения нормалей примыкающих граней;

3) Выполнение линейной интерполяции нормалей вдоль ребер;

4) Выполнение линейной интерполяции нормалей вдоль сканирующей строки.

5) Вычисление интенсивности в каждой вершине

Вывод

Таблица 1.1 – Сравнение способов представления данных о ландшафте

Способ	Наглядность представления данных	Сложность модификации данных
Регулярная сетка	Высокая	Низкая
Иррегулярная сетка	Высокая	Средняя
Посегментная карта высот	Средняя	Высокая

Таблица 1.2 – Сравнение алгоритмов процедурной генерации ландшафта

Алгоритм	Качество ландшафта	Отстутствие артефактов	Контроль ландшафта
Diamond-Square	Среднее	–	Низкая
Холмовой алгоритм	Среднее	+	Средний
Шум Перлина	Высокое	+	Высокая

Таблица 1.3 – Сравнение алгоритмов удаления невидимых линий и поверхностей

Алгоритм	Сложность алгоритма	Скорость работы	Типы объектов
Алгоритм Робертса	$O(n^2)$	Средняя	Выпуклые многогранники
Алгоритм с z-буфером	$O(np)$	Высокая	Произвольные
Алгоритм с обратной трассировки лучей	$O(np)$	Низкая	Произвольные

Таблица 1.4 – Сравнение моделей освещения

Модель освещения	Реалистичность изображения	Объем вычислений
Модель Ламберта	Средняя	Средний
Модель Фонга	Высокая	Большой

Таблица 1.5 – Сравнение алгоритмов закраски

Алгоритм закраски	Скорость работы	Реалистичность изображения	Сочетание с диффузным отражением
Простая закраска	Высокая	Низкая	Высокое
Закраска по Гуро	Средняя	Средняя	Высокое
Закраска по Фонгу	Низкая	Высокая	Средняя

В данном разделе были рассмотрены существующие методы для визуализации и построения трехмерного ландшафта. В качестве способа представления данных о ландшафте была выбрана регулярная карта высот ввиду своей простоты представления данных, в качестве алгоритма генерации карты высот был выбран алгоритм шума Перлина. Поскольку основным критерием выбора алгоритмов была скорость их работы, то для удаления невидимых линий и поверхностей был выбран алгоритм, использующий Z-буффер, а в качестве модели освещения и алгоритма закраски были выбраны модель Ламберта и алгоритм закраски по Гуро.

2 Конструкторский раздел

В данном разделе ...

2.1 Требования к программному обеспечению

Разрабатываемое программное обеспечение должно предоставлять пользователю следующую функциональность:

- изменение параметров алгоритма шума Перлина для генерации ландшафта;
- изменение положение источника света;
- возможность управления положением модели (перемещение, масштабирование, поворот);
- задание и изменение уровня моря в интерактивном режиме;

При этом разрабатываемая программа должна удовлетворять следующим требованиям:

- время отклика программы должно быть менее 1 секунды для корректной работы в интерактивном режиме;
- программа должна корректно реагировать на любые действия пользователя.

2.2 Разработка алгоритмов

2.2.1 Алгоритм процедурной генерации ландшафта на основе шума Перлина

На рисунке 2.1 представлена схема алгоритма генерации карты высот на основе шума Перлина.

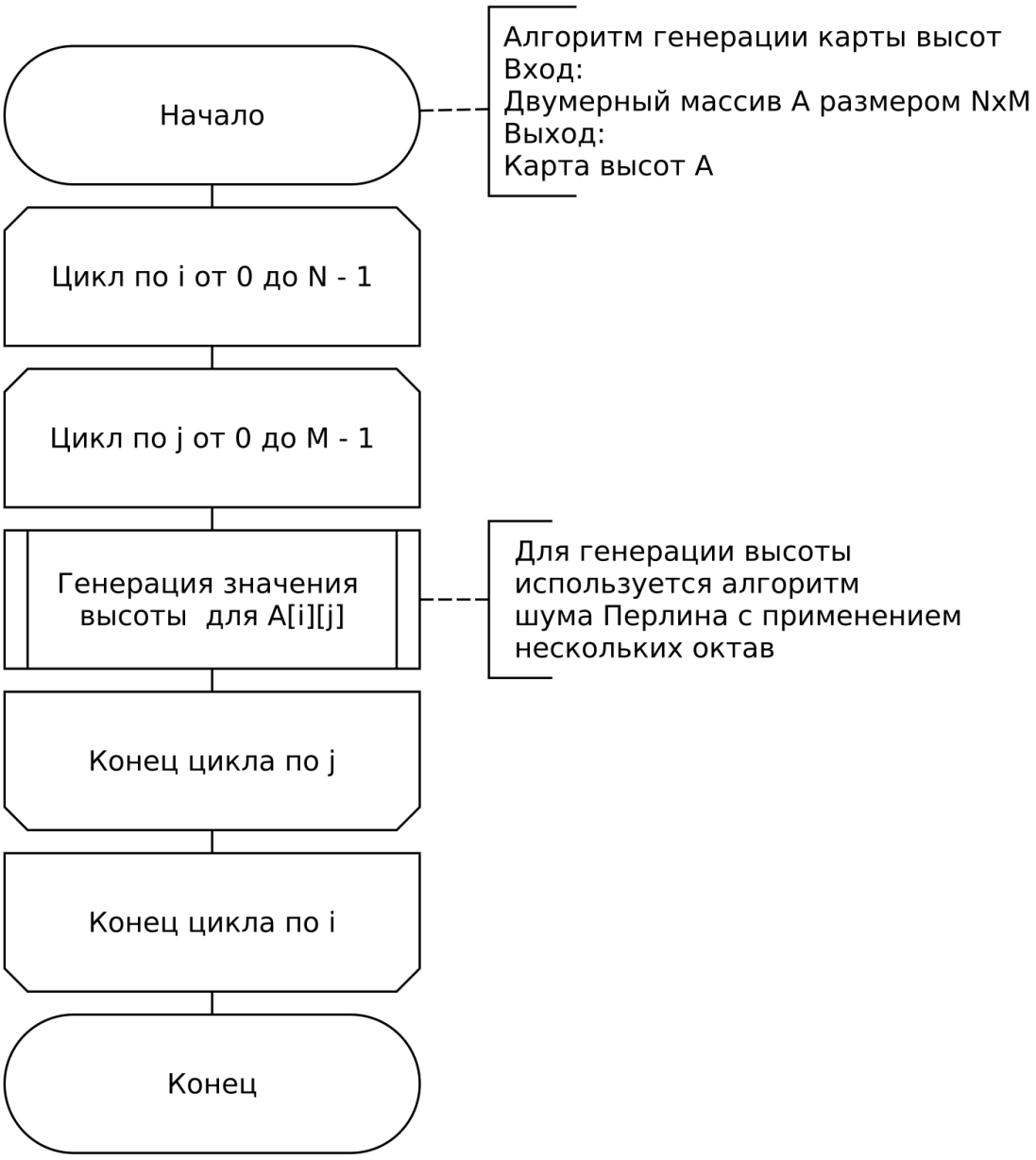

Рисунок 2.1 – Схема алгоритма генерации карты высот на основе шума Перлина

На рисунке 2.2 представлена схема алгоритма шума Перлина с применением нескольких октав.

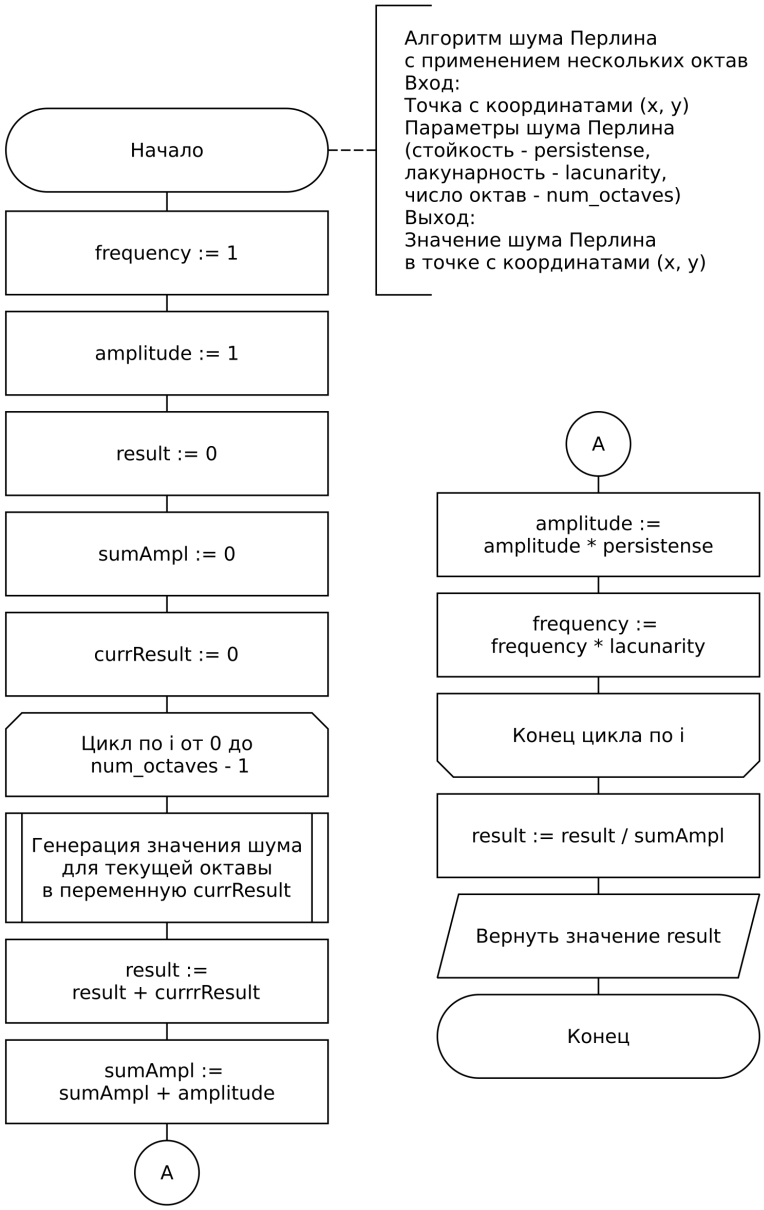

Рисунок 2.2 – Схема алгоритма шума Перлина с применением нескольких октав

На рисунке 2.3 представлена схема алгоритма шума Перлина.

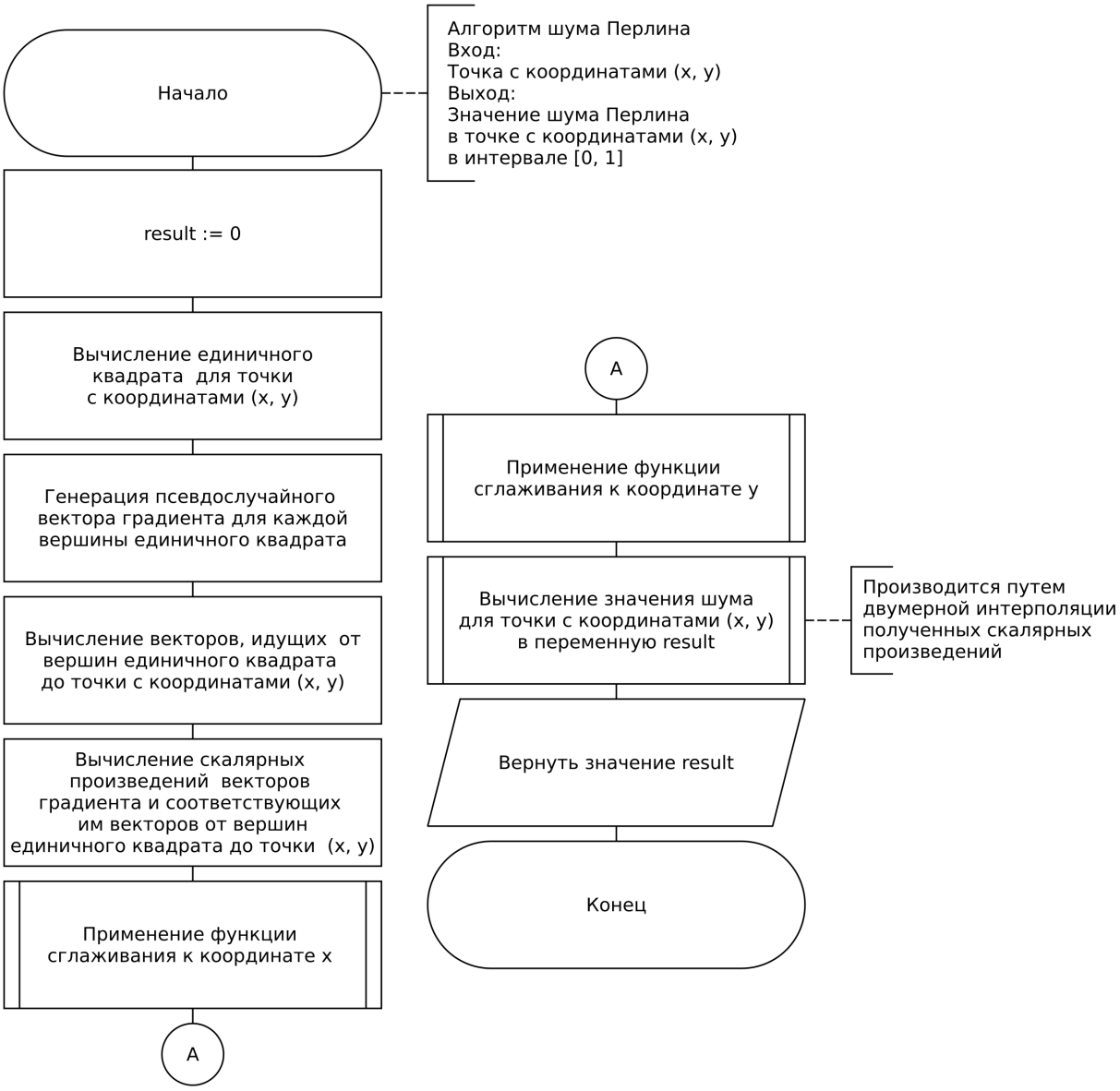

Рисунок 2.3 – Схема алгоритма шума Перлина

2.2.2 Модель освещения Ламберта

В основе модели Ламберта лежит закон Ламберта [13], который утверждает, что интенсивность света, отраженного от материала, пропорциональна косинусу угла между нормалью к поверхности и направлением света. Это означает, что поверхности, ориентированные более к световому источнику, будут ярче, а те, которые ориентированы более в сторону от источника света, будут менее освеще-

ны. На рисунке 2.4 показан пример взаимного расположения вектора нормали (вектор N) и вектора направления от точки к источнику света (вектор L).

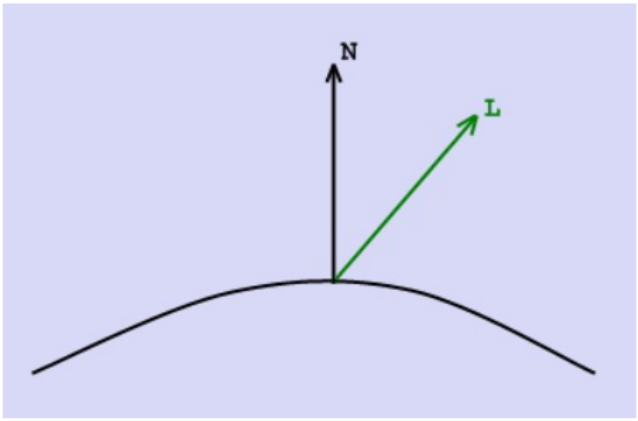

Рисунок 2.4 – Пример расположения векторов N и L в модели освещения Ламберта [12]

Пусть:

- $\vec{L}$ — единичный вектор от точки до источника;
- $\vec{N}$ — единичный вектор нормали;
- I — результирующая интенсивность света в точке;
- I_0 — интенсивность источника;
- K_d — коэффициент диффузного освещения.

Формула расчета интенсивности имеет следующий вид:

$$I = I_0 \cdot K_d \cdot cos(\vec{L}, \vec{N}) \cdot I_d = I_0 \cdot K_d \cdot (\vec{L}, \vec{N}) \tag{2.1}$$

2.2.3 Алгоритм, использующий Z-буфер

На рисунке 2.5 представлена схема алгоритма z-буфера.

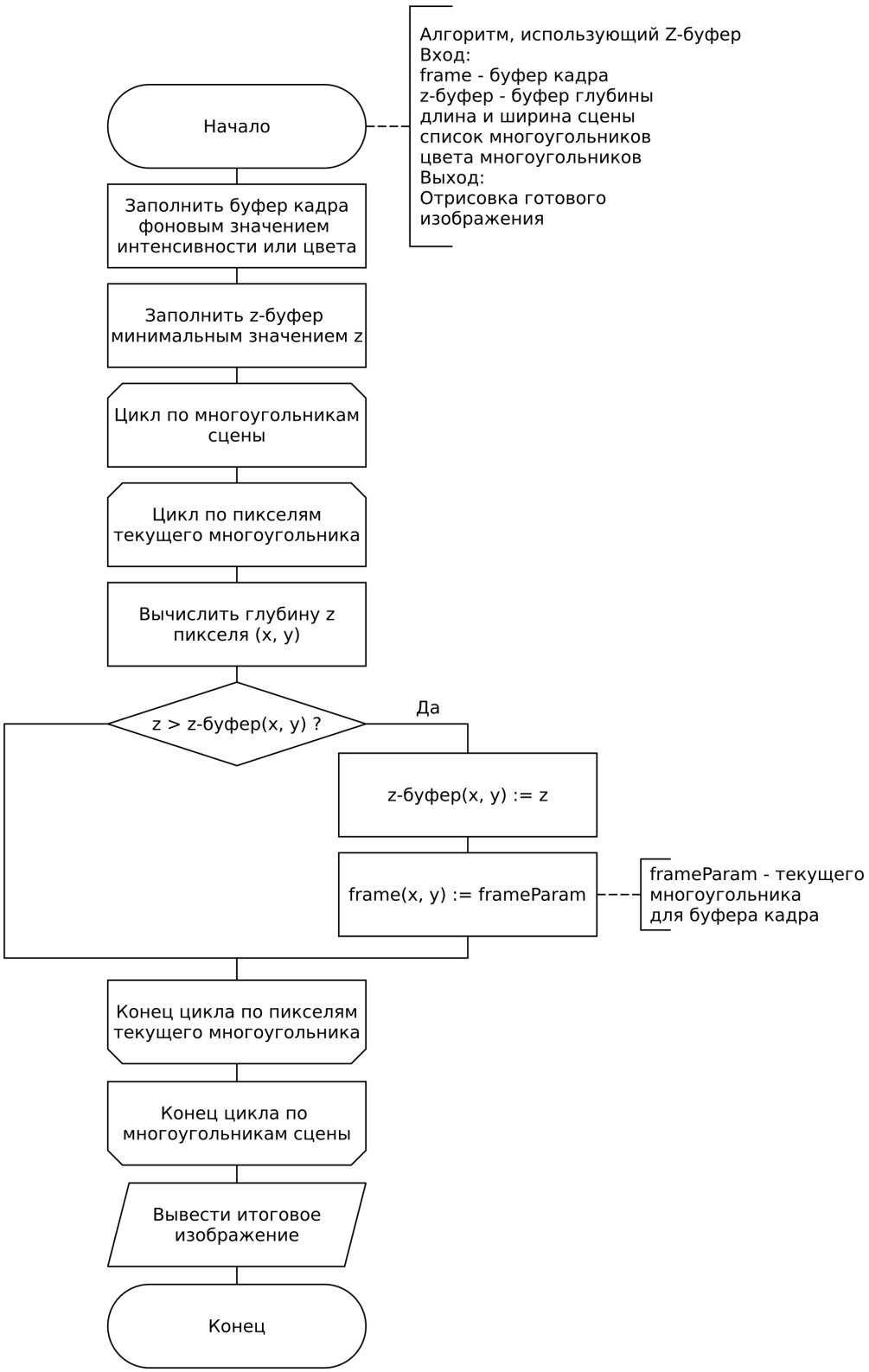

Рисунок 2.5 – Схема алгоритма z-буфера

2.2.4 Алгоритм закраски по Гуро

На рисунке 2.6 представлена схема алгоритма закраски по Гуро.

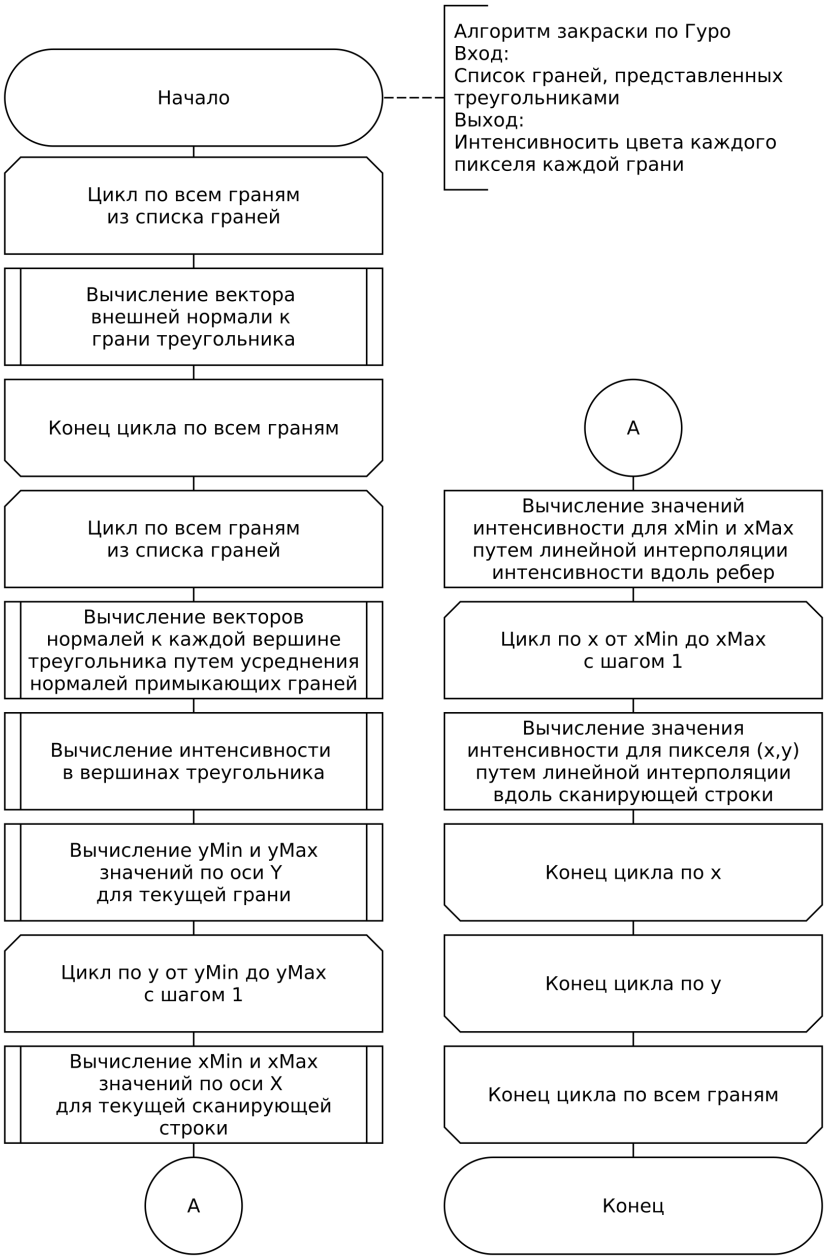

Рисунок 2.6 – Схема алгоритма закраски по Гуро

2.2.5 Алгоритм, использующий Z-буфер, объединенный с закраской по Гуро

На рисунке 2.7 представлена схема алгоритма, использующего Z-буфер, объединенного с закраской по Гуро.

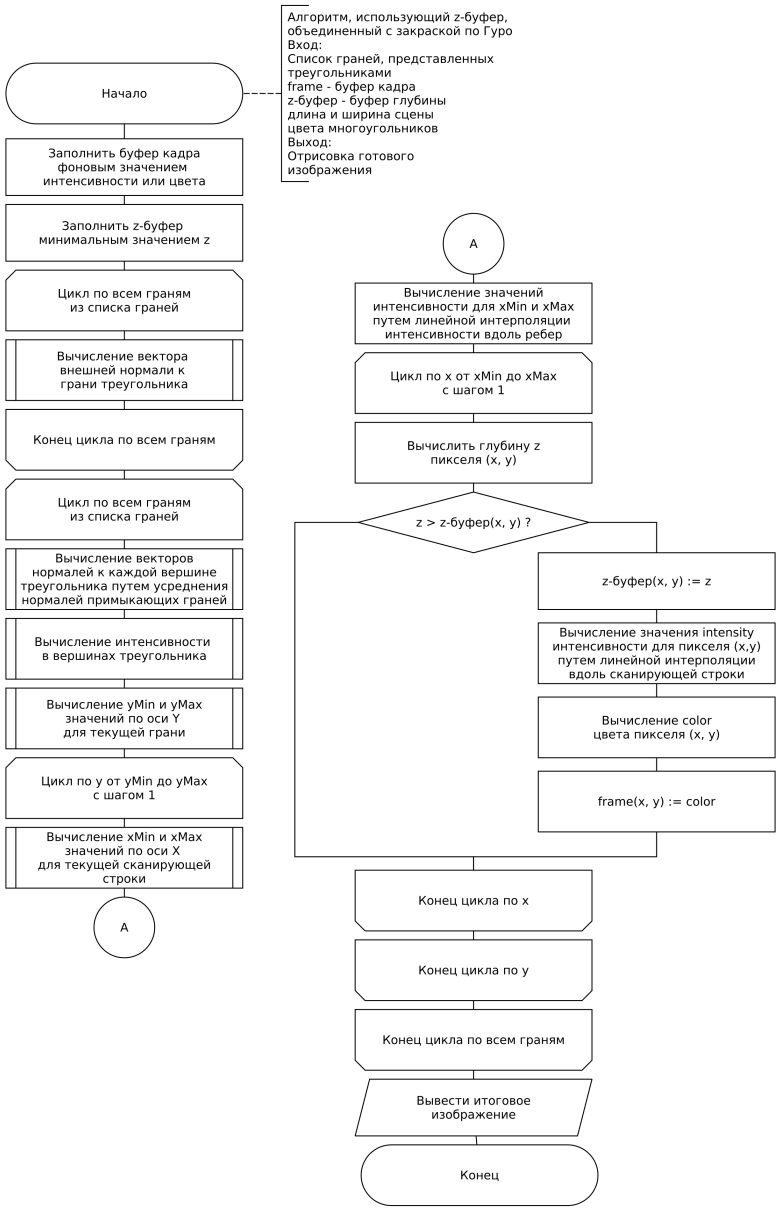

Рисунок 2.7 – Схема алгоритма, использующего Z-буфер, объединенного с закраской по Гуро

3 Технологический раздел

4 Исследовательский раздел

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Генерация трехмерных ландшафтов [Электронный ресурс]. — Режим доступа: https://www.ixbt.com/video/3dterrains-generation.shtml (дата обращения: 03.07.2023).
2. Применение дерева квадрантов в визуализации ландшафтов с изменяемым уровнем детализации [Электронный ресурс]. — Режим доступа: https://www.elibrary.ru/download/elibrary_34887617_70147073.pdf (дата обращения: 19.10.2023).
3. Исследование методов компьютерного моделирования поверхностей Земли для тестирования программного обеспечения обработки изображений со спутников [Электронный ресурс]. — Режим доступа: https://www.elibrary.ru/download/elibrary_39141278_24936959.pdf (дата обращения: 19.10.2023).
4. Применение алгоритма midpoint-displacement в процедурной генерации ландшафтов [Электронный ресурс]. — Режим доступа: https://cyberleninka.ru/article/n/primenenie-algoritma-midpoint-displacement-v-protsedurnoy-generatsii-landshaftov (дата обращения: 19.10.2023).
5. Процедурная генерация текстур на основе алгоритма Diamond-Square [Электронный ресурс]. — Режим доступа: https://libeldoc.bsuir.by/bitstream/123456789/31470/1/Multan_Protsedurnaya.PDF (дата обращения: 19.10.2023).
6. Diamond-square algorithm [Электронный ресурс]. — Режим доступа: https://en.wikipedia.org/wiki/Diamond-square_algorithm (дата обращения: 04.07.2023).
7. Terrain Generation Tutorial: Hill Algorithm [Электронный ресурс]. — Режим доступа: https://www.stuffwithstuff.com/robot-frog/3d/hills/hill.html (дата обращения: 08.07.2023).

8. *Аникеев А.* Генерация и моделирование 2D ландшафта по контрольным значениям с использованием Unity на языке программирования c# //. — — Новосибирск, Новосибирский государственный технический университет, 2020. — С. 9.

9. Perlin noise [Электронный ресурс]. — Режим доступа: `https://en.wikipedia.org/wiki/Perlin_noise` (дата обращения: 08.07.2023).

10. *Снук Г.* 3D-ландшафты в реальном времени на C++ и DirectX 9 //. — Пер. С англ М.: КУДИЦ-ОБРАЗ, 2007. — С. 368.

11. *Д. Р.* Алгоритмические основы машинной графики: Пер. с англ. //. — — СПб: БХВ-Петербург, 1989. — С. 512.

12. Простые модели освещения [Электронный ресурс]. — Режим доступа: `https://cgraph.ru/node/435` (дата обращения: 16.07.2023).

13. *Иванов В.П. Б. А.* Трехмерная компьютерная графика //. — — под ред. Г.М. Полищука. М.: Радио и связь, 1995. — С. 224.