Содержание

[**Введение** 2](#_Toc144408720)

[**1** **Аналитическая часть** 3](#_Toc144408721)

[**1.1** **Формализация объектов синтезируемой сцены** 3](#_Toc144408722)

[**1.2** **Анализ и выбор формы задания трехмерных моделей** 3](#_Toc144408723)

[**1.3** **Анализ способов представления данных о ландшафте** 4](#_Toc144408724)

[**1.3.1** **Карта высот** 5](#_Toc144408725)

[**1.3.2** **Иррегулярная сетка** 6](#_Toc144408726)

[**1.3.3** **Посегментная карта высот** 7](#_Toc144408727)

[**1.4** **Анализ алгоритмов процедурной генерации ландшафта** 8](#_Toc144408728)

[**1.4.1** **Алгоритм Diamond-Square** 8](#_Toc144408729)

[**1.4.2** **Холмовой алгоритм** 9](#_Toc144408730)

[**1.4.3** **Алгоритм Шума Перлина** 11](#_Toc144408731)

[**1.5** **Анализ алгоритмов удаления невидимых линий и поверхностей** 14](#_Toc144408732)

[**1.5.1** **Алгоритм Робертса** 14](#_Toc144408733)

[**1.5.2** **Алгоритм Варнока** 15](#_Toc144408734)

[**1.5.3** **Алгоритм, использующий Z-буфер** 16](#_Toc144408735)

[**1.5.4** **Алгоритм обратной трассировки лучей** 17](#_Toc144408736)

[**1.6** **Анализ моделей освещения** 17](#_Toc144408737)

[**1.6.1** **Модель освещения Ламберта** 18](#_Toc144408738)

[**1.6.2** **Модель освещения Фонга** 19](#_Toc144408739)

[**Литература** 21](#_Toc144408740)

**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:

* выполнить формализацию объектов синтезируемой сцены;
* провести анализ существующих алгоритмов создания ландшафта и визуализации сцены;
* выбрать подходящие алгоритмы для решения поставленной задачи;
* реализовать выбранные алгоритмы.

# **Аналитическая часть**

## **Формализация объектов синтезируемой сцены**

Сцена состоит из следующих объектов:

* Ландшафт – трехмерная модель, представляющая собой полигональную сетку, состоящую из связанных между собой плоских многоугольников;
* Источник света – представляет собой материальную точку пространства, принимающая ортогональную проекцию визуализируемой сцены из своего положения с некоторым ограниченным обзором, и в зависимости от расположения источника и направления распространения лучей света определяется тень от объектов, расположенных на сцене.

## **Анализ и выбор формы задания трехмерных моделей**

В компьютерной графике для описания трехмерных объектов существует три вида моделей: каркасная, поверхностная и твердотельная. Использование моделей позволяет правильно отображать форму и размеры объектов сцены.

* Каркасная модель – в трехмерной графике описывает совокупность вершин и ребер, которая показывает форму многогранного объекта. Это моделирование самого низкого уровня и имеет ряд серьезных ограничений, большинство из которых возникает из-за недостатка информации о гранях, которые заключены между линиями, и невозможности выделить внутреннюю и внешнюю область изображения твердого объемного тела. Однако каркасная модель требует меньше памяти и вполне пригодна для решения задач, относящихся к простым. Основным недостатком каркасной модели является то, что модель не всегда однозначно передает информацию о форме объекта [1];
* Поверхностная модель. Поверхностное моделирование определяется в терминах точек, линий и поверхностей. При построении поверхностной модели предполагается, что технические объекты ограничены поверхностями, которые отделяют их от окружающей среды. Недостатком поверхностной модели является отсутствие информации о том, с какой стороны находится поверхности материал [1];
* Твердотельная модель. Отличие данной формы задания модели от поверхностной формы состоит в том, что в объёмных моделях к информации о поверхностях добавляется информация о том, где расположен материал путём указания направления внутренней нормали.

Для решения моей задачи использование каркасной модели не подойдет, поскольку та дает не однозначного представления о форме ландшафта, также использование твердотельной модели не дает каких-либо преимуществ, поскольку для моей задачи нет необходимости знать физические свойства ландшафта. Поэтому наилучшим способом описания трехмерной модели ландшафта будет поверхностная модель.

## **Анализ способов представления данных о ландшафте**

Существует несколько основных принципов представления данных для хранения информации о ландшафтах [2]:

* Использование регулярной сетки высот (карта высот);
* Использование иррегулярной сетки вершин и связей, их соединяющих (хранение простой триангулизированной карты);
* 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.

### **Карта высот**

Данные представлены в виде двухмерного массива. Для каждой вершины с индексами [i][j] в двумерном массиве определяется соответствующее значение высоты hij.

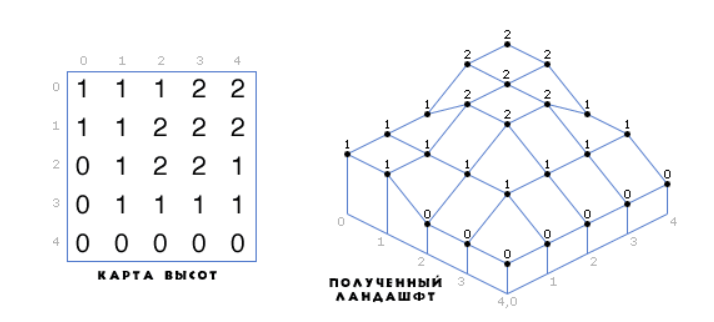


Рисунок 1.1 -- Представление ландшафта с помощью регулярной сетки высот

Преимущества:

* наглядность, простота изменения данных;
* легкость нахождения координат и высоты на карте;
* из-за близкого друг к другу расположения вершинных точек можно

более точно производить динамическое освещение.

Недостатки:

* слишком много описаний для точек;
* избыточность данных, например, при задании плоскости (задается множество точек, хотя можно обойтись и тремя).

### **Иррегулярная сетка**

Это метод представления ландшафта, который не использует равномерно распределенные узлы или точки, как в случае регулярной сетки. Вместо этого, иррегулярная сетка позволяет размещать точки или узлы в произвольных местах в зависимости от необходимости и особенностей ландшафта.

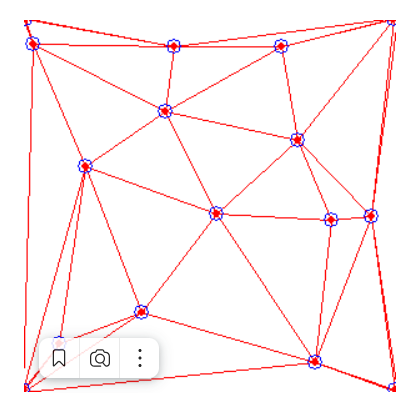


Рисунок 1.2 -- Представление ландшафта с помощью нерегулярной сетки высот

Преимущества:

* используется меньше информации для построения ландшафта. Необходимо хранить только значения высот каждой вершины и связи, эти вершины соединяющие;

Недостатки:

* 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.

### **Посегментная карта высот**

В данном способе также используются карты высот. Только вместо высот в ней хранятся индексы ландшафтных сегментов. Как эти сегменты представлены, в принципе, роли не играет. Они могут быть и регулярными, и иррегулярными (причем можно использовать и те и другие одновременно) [2].

Преимущества:

* возможность представления больших открытых пространств;
* кроме самих ландшафтов в таких блоках можно хранить и информацию о зданиях, строениях, растениях, специфических ландшафтных решениях (например, пещеры или скалы, нависающие друг над другом);
* возможность создания нескольких вариантов одного и того же сегмента, но при разной степени детализации. В зависимости от скорости или загруженности компьютера можно выбирать более или менее детализованные варианты.

Недостатки:

* проблема стыковки разных сегментов;
* неочевидность данных. Взглянув на картинку, вы не сможете моментально представить, как это должно будет выглядеть в игре;
* проблема модификации.

Проанализировав все три способа представления данных о ландшафте, выбор пал на регулярную сетку высот, поскольку:

* Первый способ дает более наглядное и понятное представление данных, позволяет легко их модифицировать. Также для этого способа существует множество алгоритмов построения ландшафта;
* При использовании иррегулярной сетки возникают сложности с хранением и модификацией данных. Также оптимизация существующих алгоритмов построения ландшафта под использование этого способа отнимет немало времени и усилий;
* Третий способ также влечет за собой проблемы с хранением и модификацией данных.

## **Анализ алгоритмов процедурной генерации ландшафта**

В данном разделе рассмотрены различные методы и алгоритмы процедурной генерации ландшафта. Также рассмотрены преимущества и недостатки каждого метода. Основным критерием выбора алгоритма будет качество получаемого ландшафта, поскольку для решения задачи необходимо создавать правдоподобный рельеф.

### **Алгоритм Diamond-Square**

Данный алгоритм является расширением одномерного алгоритма midpoint displacement на двумерную плоскость. Алгоритм Diamond-Square начинает работу с двумерного массива размера 2n + 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 [3].

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.

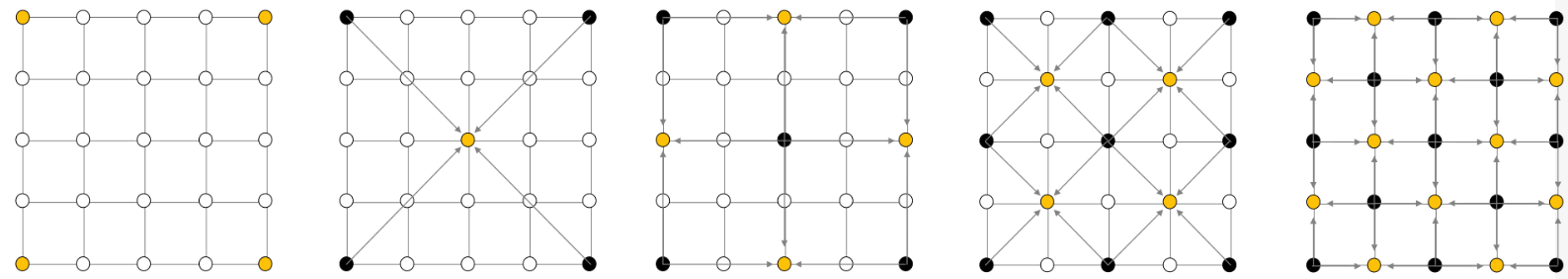


Рисунок 1.3 -- Шаги, проходимые алгоритмом Diamond-Square на примере массива 5х5

Преимущества:

* Алгоритм Diamond-Square достаточно прост в реализации, особенно по сравнению с некоторыми другими алгоритмами процедурной генерации ландшафта. Он может быть реализован относительно легко и быстро на различных платформах;
* Алгоритм Diamond-Square позволяет быстро генерировать реалистичные и природные ландшафты с разнообразной текстурой и детализацией.

Недостатки:

* Алгоритм создает заметные вертикальные и горизонтальные "складки" на краях карты из-за наиболее значительного возмущения, происходящего в прямоугольной сетке;
* При использовании этого алгоритма возникают сложность с контролем получаемого ландшафта.

### **Холмовой алгоритм**

Это простой итерационный алгоритм, основанный на нескольких входных параметрах. Алгоритм изложен в следующих шагах [2]:

1. Создаем двухмерный массив и инициализируем его нулевым уровнем (заполняем все ячейки нолями);
2. Берем случайную точку на ландшафте или около его границ (за границами), а также берем случайный радиус в заранее заданных пределах. Выбор этих пределов влияет на вид ландшафта — либо он будет пологим, либо скалистым;
3. В выбранной точке "поднимаем" холм заданного радиуса;
4. Возвращаемся ко второму шагу и так далее до выбранного количества шагов. От него потом будет зависеть внешний вид нашего ландшафта;
5. Проводим нормализацию ландшафта;
6. Проводим "долинизацию" ландшафта. Делаем его склоны более пологими.

Холм – половина шара, похоже на перевернутую параболу. Выбранный радиус определяет высоту холма и радиус его основания. Уравнение холма выглядит так:

Где (x1, y1) — заданная точка, r — выбранный радиус, (x2, y2) — высота холма. ы

Что бы сгенерировать ландшафт полностью нам необходимо построить множество таких холмов.  Но есть еще две вещи на которые нам необходимо обратить внимание. Первое — нам необходимо игнорировать отрицательные значения высоты холма. Второе — при генерации последующих холмов нам лучше добавлять полученное значение для данного холма к уже существующим значениям. Это позволяет нам построить более правдоподобный ландшафт, нежели правильно очерченные округлые холмы [2].

Поскольку алгоритм не фиксирует сгенерированные значение высоты заранее в какой-то числовой диапазон, а для модификации и масштабирования это необходимо, производится процесс нормализации ландшафта.

В полученном после нормализации ландшафте довольно много гор и мало долин. Чтобы добавить долин в полученный сгенерированный ландшафт, применяют процесс долинизации ландшафта. Один из наиболее простых способов сделать это – возвести в квадрат каждое значение высоты [4].

Преимущества:

* Алгоритм прост в реализации;
* Возможность контроля «гористости» ландшафта за счет этапов нормализации и долинизации.

Недостатки:

* С помощью этого алгоритма тяжело смоделировать склон холма, или горы, так как в процессе генерации ландшафта используются только гладкие полусферы [5];
* Этот алгоритм неприменим, когда требуется детализировано отобразить лишь часть всего ландшафта [5].

### **Алгоритм Шума Перлина**

Это математический [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC)  по генерированию  [процедурной текстуры](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D0%BA%D1%81%D1%82%D1%83%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) псевдо-случайным методом [6]. Этот алгоритм может быть реализован для n-мерного пространства, но чаще его использую 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 (x, y) из карты высот определяется, в какой ячейке сетки находится точка, генерируются вектора, идущие от точек ячейки сетки до координат (x, y).

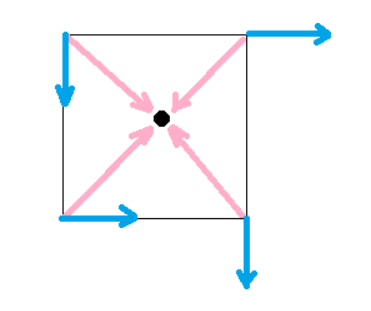


Рисунок 4. Расчёт значения высоты точки при помощи градиентов и векторов, соединяющих точки сетки с расчётной позицией

Синим цветом на рисунке 4 обозначены псевдослучайные вектора-направления градиента, розовым цветом – вектора, идущие от точек ячейки сетки до координат (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 (полином 6𝑡5 − 15𝑡4 + 10𝑡3, 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.

Чтобы контролировать генерацию шума, существует набор параметров:

* октавы (octaves) – количество уровней детализации шума;
* лакунарность (lacunarity) – множитель, который определяет изменение частоты с ростом октавы;
* стойкость (persistence) – множитель частотной амплитуды, который определяет изменение амплитуды с ростом октавы.

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

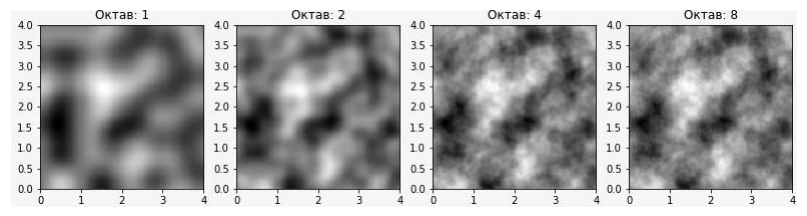


Рисунок 5 – Изменение детализации шума Перлина с применением нескольких октав

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.

Недостатки:

* Без использования механизма комбинации октав сгенерированный ландшафт не выглядит реалистичным и детализированным.

Результаты анализа алгоритмов процедурной генерации ландшафта можно представить в виде сравнительной таблицы (Таблица 1.1):

Таблица 1.1 – Сравнение алгоритмов генерации

|  |  |  |  |
| --- | --- | --- | --- |
| Алгоритм | Качество ландшафта | Отсутствие артефактов | Контроль ландшафта |
| Diamond-Square | Среднее | - | Низкий |
| Холмовой алгоритм | Среднее | + | Средний |
| Шум Перлина | Высокое | + | Высокий |

Из всех рассмотренных алгоритмов самым подходящим является алгоритм шума Перлина: он позволяет получить наиболее реалистичный и детализированный ландшафт, в нем отсутствуют какие-либо артефакты, а также именно этот алгоритм дает больше контроля при генерации ландшафта.

## **Анализ алгоритмов удаления невидимых линий и поверхностей**

Задача удаления невидимых линий и поверхностей является одной из наиболее сложных в машинной графике. Алгоритмы удаления невидимых линий и поверхностей служат для определения линий ребер, поверхностей или объемов, которые видимы или не видимы для наблюдателя, находящегося в данной точке пространства [7].

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

### **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 [7].

Этот алгоритм выполняется в 4 этапа:

* Подготовка исходных данных – составление матрицы тела для каждого тела сцены;
* Удаление ребер, экранируемых самим телом;
* Удаление ребер, экранируемых другими телами;
* Удаление линий пересечения тел, экранируемых самими телами и другими телами, связанными отношением протыкания.

Преимущества:

* Высокая точность вычислений.

Недостатки:

* Алгоритм работает исключительно с выпуклыми телами;
* Вычислительная сложность теоретически растет как квадрат числа объектов.

### **Алгоритм Варнока**

Алгоритм Варнока работает в пространстве изображения. Экран рассматривается как окно и решается вопрос о том, пусто ли оно или его содержимое достаточно просто для визуализации. Если это не так, то окно разбивается на подокна до тех пор, пока содержимое подокна не станет достаточно простым для визуализации или его размер не достигнет требуемого предела разрешения. Если информации достаточно, то происходит ее усреднение и результат отображается с одинаковой интенсивностью или цветом [7]. Пределом разбиения для растрового экрана в случае простого алгоритма является размер окна в 1 пиксель. Единой версии этого алгоритма не существует, есть только его различные модификации.

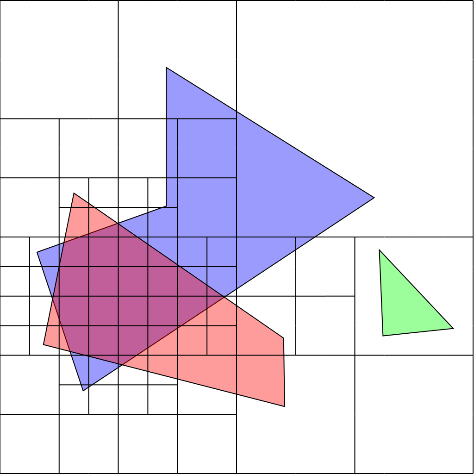


Рисунок 1.4 – Пример разбиения в алгоритме Варнока

Недостатки:

* При увеличении сложности сцены число разбиений может значительно увеличиться, что может негативно сказаться на скорости работы алгоритма;
* Рекурсивная работа алгоритма.

### **Алгоритм, использующий Z-буфер**

Это один из простейших алгоритмов удаления невидимых поверхностей. Работает этот алгоритм в пространстве изображения [7].

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 [7].

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.

Преимущества:

* Простота реализации алгоритма;
* Позволяет работать со сценами любой сложности;
* Линейная вычислительная сложность;
* Элементы сцены заносятся в буфер кадра в произвольном порядке, поэтому отсутствует необходимость в сортировке по приоритету глубины.

Недостатки:

* Большой объем требуемой памяти;
* Трудоемкость и высокая стоимость устранения лестничного эффекта;
* Трудоемкость реализации эффектов прозрачность и просвечивания.

### **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.

Преимущества:

* Возможность использования алгоритма в параллельных вычислитель системах;
* Высокая реалистичность получаемого изображения.

Недостатки:

* Большое количество вычислений и медленная работа алгоритма.

Произведя анализ нескольких алгоритмов удаления невидимых линий и поверхностей, можно прийти к следующим рассуждениям:

* Алгоритм Робертса не подходит ввиду того, что ландшафта не всегда является выпуклым многоугольником;
* Алгоритмы обратной трассировки лучей и Варнока также не подходят, поскольку они не удовлетворяет основному требованию – скорости работы;

Таким образом, в качестве алгоритма удаления невидимых линий и поверхностей был выбран алгоритм, использующий z-буфер.

## **Анализ моделей освещения**

Все модели освещения делятся на две группы: глобальные и локальные.

Алгоритмы глобальных моделей освещения учитывают не только свет, который поступает непосредственно от источника света (прямое освещение), но и последующие случаи, в которых световые лучи от того же источника отражаются другими поверхностями сцены, отражающими или нет (непрямое освещение) [8].

Алгоритмы локальных моделей освещения учитывают только тот свет, который поступает непосредственно от источника света.

Выделяют две основные модели локального освещения: модель Ламберта и модель Фонга.

### **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 и направление на источник света. Рассеянная составляющая рассчитывается по закону косинусов (закон Ламберта) [9]:

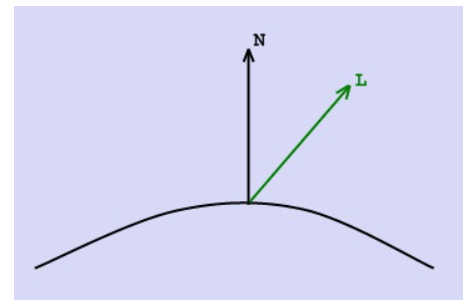


Рисунок 1.5 -- Модель освещения Ламберта

Для удобства все векторы, описанные ниже, берутся единичными. В этом случае косинус угла между ними совпадает со скалярным произведением. Формула расчета интенсивности имеет следующий вид:

Где 𝐼 — результирующая интенсивность света в точке; — интенсивность источника; — коэффициент диффузного освещения; — вектор от точки до источника; — вектор нормали в точке; — мощность рассеянного освещения.

Модель Ламберта является одной из самых простых моделей освещения. Данная модель очень часто используется в комбинации других моделей, практически в любой другой модели освещения можно выделить диффузную составляющую. Более-менее равномерная часть освещения (без присутствия какого-либо всплеска) как правило будет представляться моделью Ламберта с определенными характеристиками. Данная модель может быть очень удобна для анализа свойств других моделей (за счет того, что ее легко выделить из любой модели и анализировать оставшиеся составляющие) [9].

### **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 [9].

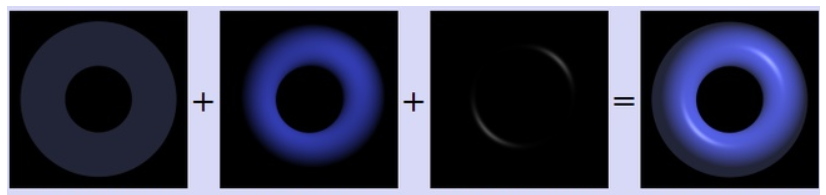


Рисунок 1.6 -- Составляющие модели Фонга (слева направо: фоновая, диффузная и зеркальная)

Для модели Фонга освещение в точке вычисляется по следующей формуле:

Где 𝐼 — результирующая интенсивность света в точке;  — фоновая составляющая;   — рассеянная составляющая;   — зеркальная составляющая.

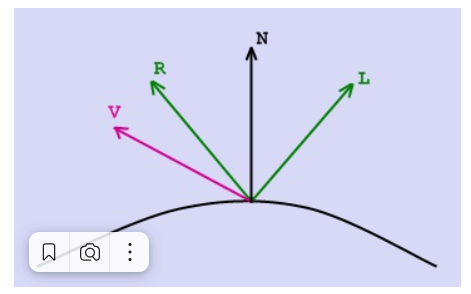


Рисунок 1.7 -- Модель освещения Фонга

Падающий и отраженный лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения, и эта нормаль делит угол между лучами на две равные части. Таким образом отраженная составляющая освещенности в точке зависит от того, насколько близки направления на наблюдателя и отраженного луча. Это можно выразить следующей формулой [9].

Формула для расчета интенсивности для модели Фонга имеет вид:

Где 𝐼 — результирующая интенсивность света в точке — коэффициент фонового освещения; — интенсивность фонового освещения; 𝐼0 — интенсивность источника; — коэффициент диффузного освещения; — вектор от точки до источника; — вектор нормали в точке; — интенсивность диффузного освещения; — коэффициент зеркального освещения; — вектор отраженного луча; — вектор от точки до наблюдателя; — коэффициент блеска; — интенсивность зеркального освещения.

Проанализировав вышеописанные модели освещения, можно сделать вывод, что наилучшим алгоритмом для решения задачи будет метод Фонга, поскольку он дает более реалистичную картинку с эффектами отражения и преломления лучей.

## **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.

### **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 [7].

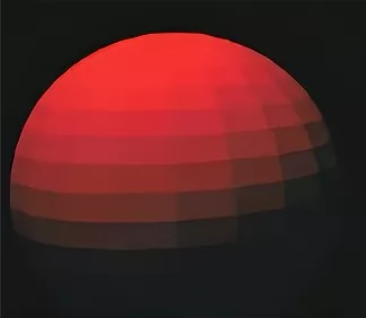


Рисунок 1.8 -- Пример простой закраски

Преимущества:

* Простота реализации;
* Быстродействие;

Недостатки:

* Нереалистичность результата.

### **Алгоритм закраски по Гуро**

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

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. 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.



Рисунок 1.9 -- Пример закраски по Гуро

Преимущества:

* Хорошо сочетается с простой моделью освещения с диффузным отражением;
* Получаемое изображение выглядит более реалистичным.

Недостатки:

* **Закраска по Гуро не предусматривает учет кривизны поверхности.** При применении закраски по Гуро мы можем потерять рёбра и получить плоское изображение. Это может произойти, когда мы каждую смежную грань закрасим одним уровнем интенсивность (когда углы, образованные гранями, одинаковые) [7].

### **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 [7].

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.



Рисунок 1.10 -- Пример закраски по Фонгу

Проанализировав вышеописанные алгоритмы закрасок, можно сделать вывод, что наилучшим алгоритмом для решения задачи будет алгоритм закраски по Фонгу, поскольку в сочетании с моделью освещения по Фонгу можно будет получить качественное и реалистичное изображение ландшафта.

**Литература**

1. Методы трехмерного моделирования [Электронный ресурс]. – Режим доступа: <https://studopedia.ru/19_307536_metodi-trehmernogo-modelirovaniya-karkasnoe-modelirovanie-poverhnostnoe-tverdotelnoe-modelirovanie-tipi-poverhnostey-chto-predstavlyayut-s-soboy-trehmernie-ob-ekti.html> (дата обращения 03.07.2023)
2. Генерация трехмерных ландшафтов [Электронный ресурс]. – Режим доступа: <https://www.ixbt.com/video/3dterrains-generation.shtml> (дата обращения 03.07.2023)
3. Diamond-square algorithm[Электронный ресурс]. – Режим доступа: <https://en.wikipedia.org/wiki/Diamond-square_algorithm> (дата обращения 04.07.2023)
4. Terrain Generation Tutorial: Hill Algorithm [Электронный ресурс]. – Режим доступа: <https://www.stuffwithstuff.com/robot-frog/3d/hills/hill.html> (дата обращения 06.07.2021)
5. А.С. Аникеев, Генерация и моделирование 2D ландшафта по контрольным значениям с использованием Unity на языке программирования c# – Новосибирск, Новосибирский государственный технический университет, 2020. – 9 с.
6. Шум Перлина [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Шум_Перлина> (дата обращения 08.07.2023)
7. Роджерс Д. Алгоритмические основы машинной графики: Пер. с англ. - М.: Мир, 1989. - 512 с.
8. Global Illumination [Электронный ресурс]. – Режим доступа: <https://en.wikipedia.org/wiki/Global_illumination> (дата обращения 16.07.2023)
9. Простые модели освещения [Электронный ресурс]. – Режим доступа: <https://cgraph.ru/node/435> (дата обращения 16.07.2023)