



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

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Расчетно-пояснительная записка к курсовому проекту

Тема: Генерация трехмерного ландшафта

Дисциплина: Компьютерная графика

Студент

ИУ7-55Б

(Группа)

(Подпись, дата)

Хетагуров П.К

(И.О. Фамилия)

Руководитель проекта

(Подпись, дата)

В.П. Степанов

(И.О. Фамилия)

Москва, 2020

Содержание

Введение	4
1 Аналитическая часть	5
1.1 Генерирование данных о структуре ландшафта	5
1.1.1 Представление данных о ландшафте	5
1.1.1.1 Карта высот	5
1.1.1.2 Вершины и связи между ними	6
1.1.1.3 Карта сегментов ландшафта	6
1.1.1.4 Сравнение	6
1.1.1.5 Вывод	6
1.1.2 Алгоритмы генерации карты высот	6
1.1.2.1 Шум Перлина	7
1.1.2.2 Холмовой алгоритм	7
1.1.2.3 Diamond-square алгоритм	8
1.1.2.4 Сравнение	8
1.1.2.5 Вывод	9
1.2 Построение трехмерного изображения на основе карты высот	9
1.2.1 Разбиение на полигоны	9
1.2.2 Удаление невидимых граней и поверхностей	9
1.2.2.1 Алгоритм, использующий Z-буфер	10
1.2.2.2 Алгоритм художника	10
1.2.2.3 Трассировка лучей	10
1.2.2.4 Сравнение	10
1.2.2.5 Вывод	11
1.2.3 Затенение	11
1.2.3.1 Затенение по Ламберту	12
1.2.3.2 Затенение по Гуро	12
1.2.3.3 Затенение по Фонгу	12
1.2.3.4 Сравнение	12
1.2.3.5 Вывод	13
1.3 Постановка задачи	13
1.4 Вывод	13
2 Конструкторская часть	14
3 Технологическая часть	15

4 Экспериментальная часть	16
4.1 Вывод	16
Заключение	17
Список литературы	18

Введение

Компьютерные системы уже глубоко проникли во все сферы жизни и являются неотъемлемыми составляющими все более различной человеческой деятельности. Организация работы предприятий, проектирование ракет, моделирование химических процессов - все это значительно облегчилось после широкого распространения компьютеров.

При все большем усложнении информационных систем и развитии компьютерной техники, росли и требования к таким системам. Очень быстро появилась потребность в визуализации данных, полученных или обработанных с помощью уже существующего программного обеспечения. Ответом на эту потребность стала машинная графика - область компьютерной науки, отвечающая за обработку, синтез и распознавание изображений. В частности выделилось такое направление машинной графики, как 3D-моделирование, отвечающее за синтез и обработку изображений объемных объектов.

В настоящее время существует большое количество задач, решаемых с помощью 3D-моделирования, например высокоточное моделирование деталей и объектов, добавление спецэффектов при производстве фильмов, компьютерные игры. Одной из таких задач является генерация трехмерного ландшафта.

Целью работы является создание программного продукта, позволяющего генерировать трехмерные модели ландшафта.

Для достижения поставленной цели необходимо решить следующие задачи:

1. проанализировать методы генерирования ландшафтов;
2. спроектировать программное обеспечение, позволяющее генерировать трехмерный ландшафт;
3. реализовать спроектированное программное обеспечение.

1 Аналитическая часть

В данном разделе будут предъявлены требования к разрабатываемому программному обеспечению, будут рассмотрены основные теоретические сведения связанные с трехмерной генерацией ландшафта, проанализированы существующие решения.

Целью работы является создание программного продукта, позволяющего генерировать трехмерные модели ландшафта.

Задачу генерации трехмерного ландшафта можно решать различными способами, но все из них можно разделить на следующие этапы:

1. генерация данных о структуре ландшафта;
2. построение трехмерного изображения по сгенерированным данным.

Рассмотрим первый этап.

1.1 Генерирование данных о структуре ландшафта

Сначала необходимо выбрать способ представления данных о ландшафте, ведь в зависимости от этого будут варьироваться алгоритмы генерации.

1.1.1 Представление данных о ландшафте

Существуют различные способы представления данных о ландшафте, вот некоторые из них:

1. карта высот (height map);
2. вершины и связи между ними;
3. карта сегментов ландшафта.

Рассмотрим каждый из них подробнее.

1.1.1.1 Карта высот

Представление данных в виде карты высот предполагает создание двумерного массива, являющегося "снимком" высоты местности. Каждый элемент массива представляет собой число, являющееся высотой в точке с координатами x и y , равными индексам в двумерном массиве с некоторым смещением.

Минусом данного способа представления данных является избыточность при представлении некоторых поверхностей. Например, для описания простой плоскости можно использовать всего 3 точки. Плюсом же является простота представления и изменения данных, наглядность, возможность быстро определить высоту в заданной точке. Этот формат хранения является наиболее распространенным и поэтому существует возможность работать с картой высот в других программах [1].

1.1.1.2 Вершины и связи между ними

Если хранить информацию о ландшафте как вершины и связи между ними, то основным преимуществом, по отношению к предыдущему случаю, будет существенно меньшее количество информации. Но этот способ лишается преимуществ, которыми обладало хранение информации в качестве карты высот. Так, изменение и просмотр данных, представленных таким образом, требует специализированного программного обеспечения, а основные алгоритмы построения ландшафта требуют адаптации, так как заточены на использование карты высот [2].

1.1.1.3 Карта сегментов ландшафта

Карты сегментов ландшафта - есть двумерный массив, похожий на карту высот, но определяющий не высоту в конкретной точке, а номер созданного заранее блока ландшафта. Плюсами данного способа является снижение требуемых данных для хранения, ведь ландшафтные блоки могут быть большого размера. Минусами является необходимость регенерации большого количества различных ландшафтных блоков и проблема совместимости этих блоков между собой [3].

1.1.1.4 Сравнение

В таблице 1 представлено общее сравнение вышеперечисленных способов представления ландшафтов.

Таблица 1 – Сравнение способов представления данных о ландшафте

	Наличие сторонних программ редактирования	Отсутствие необходимости адаптации алгоритмов генерации	Отсутствие избыточности данных
Карта высот	+	+	-
Иррегулярная сетка вершин	-	-	+
Карта сегментов ландшафта	-	-	-

1.1.1.5 Вывод

Из вышесказанного можно сделать вывод, что представление в виде карты высот наиболее простое и охватывает общий случай. Для дальнейших рассуждений выберем его.

1.1.2 Алгоритмы генерации карты высот

Второй этап в генерации данных о структуре ландшафта - собственно генерация. Существуют различные способы генерации карты высот, вот некоторые из них:

1. шум Перлина;
2. холмовой алгоритм;
3. diamond-square алгоритм.

Рассмотрим каждый из них подробнее.

1.1.2.1 Шум Перлина

При применении шума Перлина вся площадь карты покрывается сеткой. В каждом узле сетки строится случайный двумерный вектор единичной длины, указывающий в случайном направлении. Таким образом каждый пиксель изображения попадает в случайную клетку, вектора в углах которой указывают в случайном направлении. Далее создаются четыре вектора, соединяющие углы ячейки сетки с данным пикселем. После этого для каждой пары векторов, выходящих из одного угла, находится скалярное произведение. Полученные четыре значения надо объединить, для получения высоты в заданном пикселе. Наиболее распространена интерполяция этих четырех значений в зависимости от близости пикселя к углам сетки. При увеличении количества узлов сетки, получается более частый шум, похожий на белый, а при уменьшении наоборот, более гладкий [4].

На рисунке 1 представлены векторы, о которых шла речь.

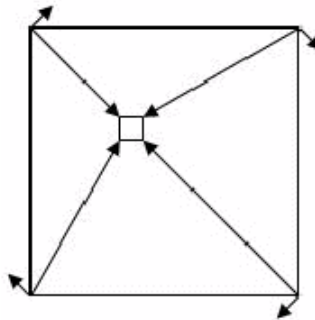


Рисунок 1 – Определение высоты в точке внутри ячейки сетки

Поверх уже сгенерированного шума можно наложить ещё один шум, меньшей частоты и более частой сетки, что придаст ландшафту реалистичный вид. Этот способ позволяет легко повторить генерацию, достаточно знать только вектора в каждом угле сетки.

1.1.2.2 Холмовой алгоритм

Изначально считается, что все точки находятся на одной высоте, в дальнейшем в произвольных местах задаются холмы различной высоты и диаметра. В результате наложения холмов друг на друга получается ландшафт [5].

1.1.2.3 Diamond-square алгоритм

Этот алгоритм рекурсивен. Изначально вся карта покрывается одной ячейкой и каждой вершине этой ячейки случайным образом присваивается высота. Далее происходит этап square - в квадрате определяется центральная точка, путем усреднения угловых и добавления случайного отклонения. Далее происходит этап diamond - определяются высоты точек, лежащих в серединах сторон. Для этого используются как точки, лежащие сверху и снизу, так и точки справа и слева, найденные на шаге square. Далее происходит рекурсивное повторение этого алгоритма для каждого квадрата. Заметим, что так как в шаге diamond используются точки, полученные на шаге square, то шаг square проводится для всех текущих квадратов, так же, как и шаг diamond для всех получившихся ромбов. Для простоты применения этого алгоритма необходимо использовать квадратную карту со стороной, равной степени двойки [6].

На рисунке 2 визуальна представлена одна итерация этого алгоритма.

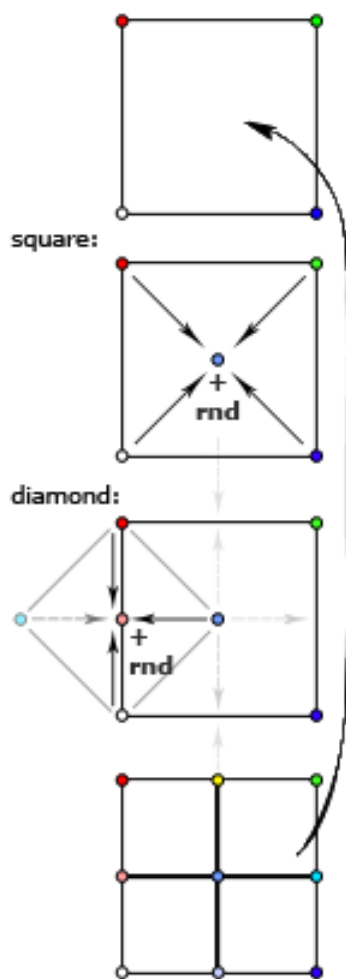


Рисунок 2 – Алгоритм diamond-square

1.1.2.4 Сравнение

В таблице 2 представлено общее сравнение вышеперечисленных способов генерации карты высот.

Таблица 2 – Сравнение способов генерации карты высот

	Возможность влиять на генерацию	Возможность повторения генерации
Шум Перлина	+	+
Холмовой алгоритм	-	-
Diamond-square	+	-

1.1.2.5 Вывод

Шум Перлина предоставляет более гибкую настройку, позволяя накладывать несколько шумов разной частоты и меняя шаг сетки. Для получения такого же предсказуемого эффекта при генерации другими алгоритмами потребуется дополнительная обработка шумом Перлина, поэтому было принято решение изначально использовать шум Перлина.

1.2 Построение трехмерного изображения на основе карты высот

Построение изображения на основе карты высот можно разделить на следующие этапы:

1. разбиение на полигоны;
2. удаление невидимых граней и поверхностей;
3. затенение.

Рассмотрим каждый этап.

1.2.1 Разбиение на полигоны

Самым простым многоугольником является треугольник и, как следствие, с первого взгляда было бы логично разбить модель на треугольные полигоны. Но у треугольных полигонов есть несколько недостатков. Так, треугольные полигоны хуже деформируются и могут создавать артефакты, что, при условии совместимости со сторонними редакторами, может привести к нежелательным сложностям. Четырехугольные же полигоны, помимо легкости в деформации, позволяют уменьшить количество вычислений без существенного снижения качества [?].

Было принято решение использовать четырехугольные полигоны.

1.2.2 Удаление невидимых граней и поверхностей

Рассмотрим некоторые алгоритмы удаления невидимых граней и поверхностей:

1. алгоритм, использующий Z-буфер;
2. алгоритм художника;
3. метод трассировки лучей.

Рассмотрим каждый из них.

1.2.2.1 Алгоритм, использующий Z-буфер

Алгоритм работает в пространстве изображения. Идея алгоритма крайне проста. Создается буфер, хранящий z-координату каждого пикселя изображения, и заполняется минимальным значением. При обработке очередного пикселя, значение его z-координаты сравнивается с хранящимся в буфере и, если его координата больше хранящейся, то её значение заносится в буфер, а сам пиксель выводится на экран, закрашивая предыдущий [8]. Заметим, что размер буфера будет равен размеру пространства изображения. Тем самым, память, требуемая для работы алгоритма, определяется размерами изображения. Основным плюсом данного алгоритма является простота реализации и возможность обрабатывать полигоны объекты в произвольном порядке.

1.2.2.2 Алгоритм художника

Идея алгоритма заключается в том, чтобы изображать объекты начиная с самых удаленных от наблюдателя. Для использования алгоритма художника сначала необходимо провести сортировку объектов по глубине и занести их в список приоритетов. В окончательном списке никакие из двух элементов не должны перекрывать друг друга. После этого объекты отображаются, начиная с самого удаленного от наблюдателя [9]. При использовании данного алгоритма существует сильная зависимость скорости работы от взаимного расположения объектов. Так, при циклическом перекрытии многоугольников используются сложные тесты, для определения ближайшего к наблюдателю. В худшем случае многоугольники необходимо разбивать на несколько новых.

1.2.2.3 Трассировка лучей

В этом методе через каждый пиксель картинной плоскости выпускается луч. Из всех граней, с которыми этот луч пересекается, выбирается ближайшая, и точка пересечения отображается.

С помощью данного метода возможно построения высококачественных изображений, алгоритм легко распараллеливается, вычислительная сложность слабо зависит от сложности сцены. Существенным же минусом является низкая производительность [10].

1.2.2.4 Сравнение

В таблице 3 представлено сравнение вышеперечисленных алгоритмов.

Таблица 3 – Сравнение алгоритмов визуализации

	Маленькая вычислительная сложность	Легкость реализации	Работа с произвольными входными данными
Алгоритм, использующий Z-буфер	+	+	+
Алгоритм художника	+	+	-
Трассировка лучей	-	-	+

1.2.2.5 Вывод

Из проведенного сравнения видно, что наиболее оптимальным алгоритмом, в отсутствии требования к фотореалистичному изображению, является алгоритм, использующий Z-буфер.

1.2.3 Затенение

В моделях затенения освещенность складывается фоновой, диффузной и зеркальной составляющих [11].

Фоновая составляющая - некоторая константа освещения, присутствующая в каждой точке, вне зависимости от расположения точки и источника света.

Диффузная составляющая - это компонента, отвечающая за рассеивание света после попадания на поверхность.

Зеркальная составляющая отвечает за моделирование отражающих свойств материала. Её добавление позволяет отобразить блики на поверхности.

На рисунке 3 визуальны представлены все компоненты затенения и получившееся изображения.

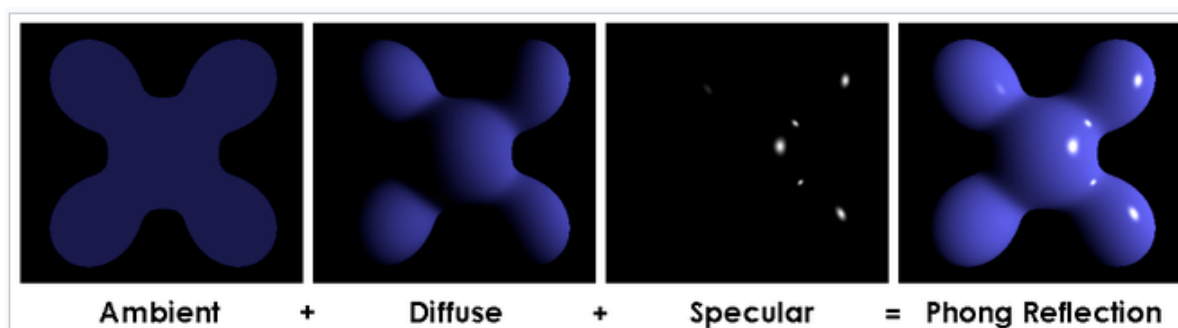


Рисунок 3 – Компоненты затенения

Основные модели затенения:

1. затенение по Ламберту;
2. затенение по Гуро;
3. затенение по Фонгу.

Рассмотрим каждый из них подробнее.

1.2.3.1 Затенение по Ламберту

Модель затенения по Ламберту является одной из самых простейших и позволяет описать идеальное диффузное освещение. Если N - вектор нормали в некоторой точке, L - вектор, направленный на источник света, а $I_{ambient}$ - фоновая составляющая, то освещённость по модели Ламберта считается по формуле (1):

$$I = I_{ambient} + \max(0, \text{dotProduct}(N, L)) \quad (1)$$

Затенение по Ламберту предполагает выбор случайной точки на полигоне и экстраполирование цвета этой точки на весь полигон, что создает явный полигональный характер изображения [12].

1.2.3.2 Затенение по Гуро

При затенении по Гуро сначала считаются интенсивности света в каждой вершине, а после используется билинейная интерполяция интенсивностей, что убирает явную дискретность освещения. В этом методе может быть неверно определена зеркальная составляющая, так как происходит усреднение векторов нормалей. В методе Гуро учитывается зеркальная составляющая, что отражено в формуле (2), использующейся для подсчета интенсивности [13].

$$I = I_{ambient} + kd * \max(0, \text{dotProduct}(N, L)) + ks * \max(0, \text{dotProduct}(N, \frac{L + V}{|L + V|}))^{SpecularPower}, \quad (2)$$

где N , L , $I_{ambient}$ тоже, что и в затенении по Ламберту, kd и ks - коэффициенты рассеивания и зеркального освещения соответственно, V - вектор, направленный из точки к наблюдателю, $SpecularPower$ - коэффициент блеска материала.

1.2.3.3 Затенение по Фонгу

Формула для расчета интенсивности при затенении по Фонгу совпадает с формулой (2), но происходит не интерполирование интенсивностей, а интерполирование нормалей, что дает более точный учет зеркальной составляющей. Изображение получается более реалистичным, чем в методе Гуро, но трудоемкость также повышается [11].

1.2.3.4 Сравнение

В таблице 4 представлено сравнение вышеперечисленных моделей затенения.

Таблица 4 – Сравнение моделей затенения

	Маленькая вычислительная сложность	Устранение дискретизации закраски	Учет зеркальной составляющей
Затенение по Ламберту	+	-	-
Затенение по Гуро	+	+	+
Затенение по Фонгу	-	+	+

1.2.3.5 Вывод

Из рассмотренных моделей затенения была выбрана модель затенения по Гуро, из-за небольшой вычислительной сложности и учета зеркальной составляющей.

1.3 Постановка задачи

Разработать программное обеспечение, позволяющее генерировать и отображать ландшафт, представленный в виде карты высот. При генерации использовать шум Перлина. Дать возможность пользователю влиять на генерируемый ландшафт, путем изменения количества узлов сетки и добавления шумов меньшей частоты поверх сгенерированного ландшафта. При отображении использовать алгоритм Z-буфера и затенение по Гуро.

1.4 Вывод

В данном разделе были рассмотрены и проанализированы существующие алгоритмы решения задачи генерации и отображения ландшафта, уточнены требования к разрабатываемому ПО.

2 Конструкторская часть

3 Технологическая часть

4 Экспериментальная часть

4.1 Вывод

Заключение

Список литературы

- [1] Карта высот. Wikipedia [Электронный ресурс]. Режим доступа: (дата обращения - 08.12.2020) Свободный. URL: <https://en.wikipedia.org/wiki/Heightmap>
- [2] Генерация трехмерных ландшафтов [Электронный ресурс]. Режим доступа: (дата обращения - 08.12.2020) Свободный. URL: <https://www.ixbt.com/video/3dterrains-generation.shtml>
- [3] Генерация трехмерных ландшафтов [Электронный ресурс]. Режим доступа: (дата обращения - 08.12.2020) Свободный. URL: <https://www.ixbt.com/video/3dterrains-generation.shtml>
- [4] Шум перлина [Электронный ресурс]. Режим доступа: (дата обращения - 08.12.2020) Свободный. URL: <https://medium.com/@yvanscher/playing-with-perlin-noise-generating-realistic-archipelagos-b59f004d8401>
- [5] Холмовой алгоритм [Электронный ресурс]. Режим доступа: (дата обращения - 08.12.2020) Свободный. URL: <https://www.ixbt.com/video/3dterrains-generation.shtml>
- [6] Diamond-square algorithm [Электронный ресурс]. Режим доступа: (дата обращения - 08.12.2020) Свободный. URL: <http://jmecom.github.io/blog/2015/diamond-square/>
- [7] Polygon Mesh [Электронный ресурс]. Режим доступа: (дата обращения - 08.12.2020) Свободный. URL: https://en.wikipedia.org/wiki/Polygon_mesh
- [8] Томский политехнический университет [Электронный ресурс]. Режим доступа: (дата обращения - 08.12.2020) Свободный. URL: <http://compgraph.tpu.ru/zbuffer.htm>
- [9] kursgm.ru [Электронный ресурс]. Режим доступа: (дата обращения - 08.12.2020) Свободный. URL: <http://kursgm.ru/grpraktik/grafikomp21.htm>
- [10] Томский политехнический университете [Электронный ресурс]. Режим доступа: (дата обращения - 08.12.2020) Свободный. URL: <http://compgraph.tpu.ru/ray.htm>
- [11] Bui Tuong Phong, Illumination for computer generated pictures, Communications of ACM 18 (1975), no. 6, 311–317
- [12] Ikeuchi, Katsushi (2014). "Lambertian Reflectance". Encyclopedia of Computer Vision. Springer. pp. 441–443.
- [13] Gouraud, Henri (1971). "Continuous shading of curved surfaces"(PDF). IEEE Transactions on Computers. C-20 (6): 623–629. doi:10.1109/T-C.1971.223313.
- [14] [Электронный ресурс]. Режим доступа: (дата обращения - 08.12.2020) Свободный. URL:
- [15] Golang [Электронный ресурс]. Режим доступа: (дата обращения - 20.11.2020) Свободный. URL: https://ru.wikipedia.org/wiki/C_Sharp
- [16] Visual Studio [Электронный ресурс]. Режим доступа: (дата обращения - 20.11.2020) Свободный. URL: <https://visualstudio.microsoft.com/ru/>