АЛГОРИТМЫ ГЕНЕРАЦИИ ФРАКТАЛЬНЫХ ЛАНДШАФТОВ

Барилко М.А.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Оношко Д.Е. – старший преподаватель

Произведена сравнительная характеристика алгоритма Diamond-Square и алгоритма с использованием шума Перлина. Сравнение производилось по таким параметрам как: сложность понимания и программной реализации, степень свободы изменения параметров генерации, ограничения алгоритмов.

Процедурная генерация контента – создание при помощи алгоритмов игрового контента с частичным или полностью отсутствующим участием человека. «Игровой контент» – это широкое определение, включающее в себя: уровни, карты, квесты, текстуры, персонажей, растительность, правила, разнообразную динамику и структуры, но не сам игровой движок или поведение не игровых персонажей.[1] Широкое применение получил шум Перлина. Он и его модификации позволяют создавать различные текстуры, анимировать и создавать облака и волны на воде или создавать разнообразные поверхности и ландшафты. В свою очередь алгоритм Diamond-Square, основанный на Midpoint displacement, направлен на генерацию именно поверхностей.

Было выделено четыре параметра, по которым можно сравнить данные алгоритмы:

- сложность понимания;
- сложность алгоритмов;
- количество степеней свободы изменения параметров генерации;
- ограничения алгоритмов.

Сложность понимания оценивалась на основе сложности математического аппарата необходимого для понимания и реализации данного алгоритма.

Реализация алгоритмов была произведена на языке ассемблера, а именно FASM. Для отрисовки конечного результата программы использовался программный интерфейс OpenGL.

Алгоритм Diamond-Square состоит из двух шагов: Diamond и Square. В плане сложности понимания алгоритм очень прост, так как он полностью помещается в один рисунок (см. рис.1). На рисунке показана очерёдность работы с узлами для поверхности 5 на 5. Сначала задаётся случайное значение угловых узлов. По шагу Square находится и задаётся центральный узел квадрата значением среднего арифметического 4 угловых узлов и случайного смещения. Затем шаг Diamond задаёт значение ещё четырём краевым точкам, находящимся на пересечении двух угловых и центральной, используя значение случайного смещения и среднего арифметического узлов, на пересечении которых находится данный узел. Затем шаг Diamond и Square повторяются до момента, когда расстояние между узлами не будет равно единице.

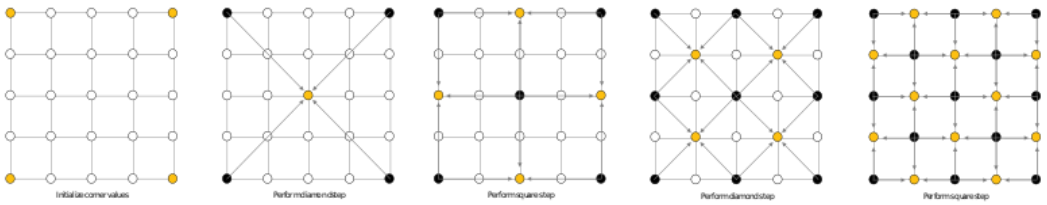


Рисунок 1 – алгоритм Diamond-Square[2]

Сложность данного алгоритма $O(n^2)$, так как нужно обратиться ко всем узлам. Степени свободы, которые влияют на вид поверхности, представлены коэффициентом шероховатости и разрешением. Коэффициент шероховатости отвечает за дельту высот между смежными узлами: чем он больше, тем больше будет разность высот. Разрешение – это количество узлов, которые будут иметь собственную высоту. Ограничения алгоритма – это то, что он подходит только для определённых размеров (длина стороны карты должна быть $2^n + 1$), и то, что после генерации необходима повторная обработка, так как на поверхности появляются пики, которые необходимо сглаживать.

Рассмотрим алгоритм на основе шума Перлина. Для понимания алгоритма необходимы базовые знания линейной алгебры и аналитической геометрии, т.е. требуется знать, что такое векторы и операции над ними и что такое интерполяция. В сравнении с предыдущим алгоритмом, этот требует знания выше школьной программы. Суть алгоритма для двумерной плоскости в том, что задаётся массив значений, представляющий из себя сетку как на рисунке 2, где каждое значение – это случайный двумерный юнит-вектор градиента. После для любой случайной точки из данной сети, причём не обязательно чтобы она лежала на узлах, нужно найти в какой ячейке данной сетки

находится данная точка. Далее для каждого угла необходимо вычислить вектор смещения относительно начала сетки/массива и посчитать скалярное произведение между смешением и вектором произведением и далее интерполировать между 4-мя значениями чтобы найти значение в данной точке.

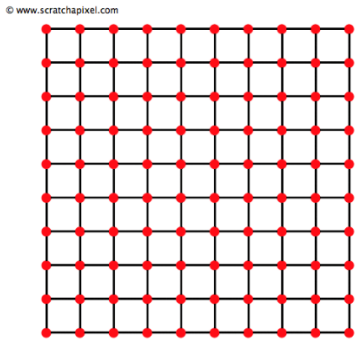
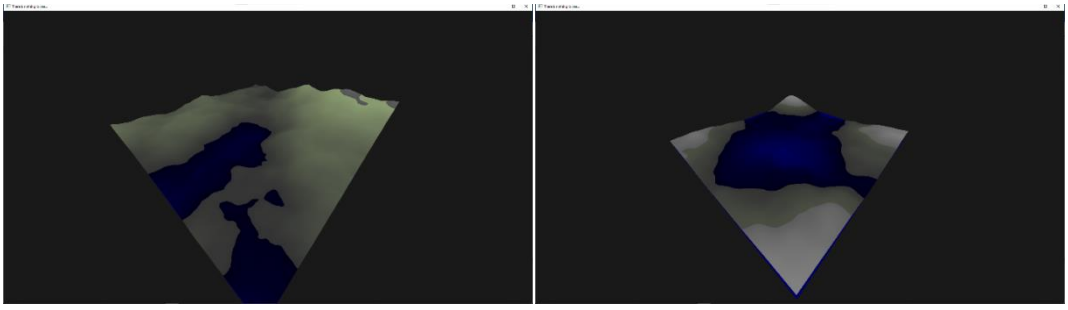


Рисунок 2 – представление массива в виде [3]

Сложность данного алгоритма $O(n^2)$(Необходимо пройтись по всем узлам, а для каждого узла сложность константная), однако для каждого узла необходимо рассчитать и сложить значения нескольких октав, чтобы получить поверхность, напоминающую ландшафт. Тогда можно ввести m – количество октав, и сложность алгоритма можно записать *как* $O(mn^2)$. Октавами называют функции шума Перлина, у которых частота различается в два раза. Степеней свободы у алгоритма множество и количество зависит от разрешения. Во-первых, на генерацию влияет количество октав, каждая октава добавляет шероховатость поверхности. Во-вторых, для каждой октавы может быть определён множитель, который влияет на то, насколько сильно октава будет выражена в конечной поверхности. Ограничениями данного алгоритма является память, ведь кроме самой поверхности необходимо хранить векторы-градиенты для каждой из октав. Однако в подготовленной реализации расход памяти был исправлен, при помощи функции генерации псевдослучайных чисел по двум координатам, которая позволила генерировать единичный вектор для каждой точки, в место того, чтобы хранить его в заранее подготовленном массиве. Таким же образом было встроено *зерно* для генерации, позволяющее переносить сгенерированный ландшафт, сохраняя только *зерно*, и, предположим, на другой машине, используя всё то же *зерно* получить идентичный ландшафт.

Следует отметить, что, несмотря на отличия двух алгоритмов, конечный результат у них схож (см. рис. 3 и 4). Тем не менее алгоритм Diamond-Square может позволить сгенерировать ландшафт без каких-либо углублённых знаний. В любом случае оба этих алгоритма подходят в учебных целях. С другой стороны, алгоритм на основе шума Перлина имеет значительное преимущество благодаря его степеням свободы. Это позволяет генерировать более разнообразные вариации поверхностей чем в предыдущем алгоритме. Также он имеет потенциал к модификации. Таким образом алгоритм на основанный на шуме Перлина можно сравнить с каркасом, на который можно и нужно навешивать разнообразные модификации, когда как алгоритм Diamond-Square не поддаётся модификациям столь же просто, однако представляет законченный алгоритм полностью выполняющий своё назначение.



Рисунки 3 и 4 – результаты работы алгоритма Diamond-Square и алгоритма основанного на шуме Перлина соответственно

Список использованных источников:

1. *Procedural content generation [Электронный ресурс]. — Режим доступа: https://www.researchgate.net/publication/285878527_Procedural_content_generation_goals_challenges_and_actionable_steps — Дата доступа: 07.03.2023.*
2. *Diamond-square algorithm [Электронный ресурс].—Режим доступа: https://en.wikipedia.org/wiki/Diamond-square_algorithm. — Дата доступа: 09.03.2023.*
3. *Scratchapixel [Электронный ресурс]. — Режим доступа: https://www.scratchapixel.com/lessons/procedural-generation-virtual-worlds/procedural-patterns-noise-part-1/introduction.html. — Дата доступа: 15.03.2023.*