|  |  |
| --- | --- |
| https://lh4.googleusercontent.com/W6y9JO8W24lRISAWmEBOzUdjU3TtANfCyRNti7vDgns3PvJrRnSgI2XoKdFNiv8C22Z8lO-B8TYCLTjOs0z9_2kTe5XJQTY0eHZ407Air-HIRm-iFqK95ADIlXZW44kPjp8FFY1sofQ2MeG5Ww | **Министерство образования и науки 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\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Студент\_\_\_Турсунов Жасурбек Рустамович\_\_\_

*фамилия, имя, отчество*

Группа\_\_\_\_ИУ7-46Б\_

Тип практики \_\_Стационарная\_\_

Студент                                 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     \_\_**Турсунов Ж.Р.**\_\_\_\_**

*подпись, дата                   фамилия, и.о.*

Руководитель практики  \_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     \_\_**Куров А.В.**\_**

*подпись, дата                   фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2020 г.*

**Министерство образования и науки 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)**

УТВЕРЖДАЮ

Заведующий кафедрой \_ИУ7\_\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_  И.В.Рудаков \_\_\_\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

**ЗАДАНИЕ**

**на выполнение производственной практики**

по дисциплине \_\_\_\_\_\_\_\_\_\_\_\_\_\_Компьютерная графика\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_\_\_ИУ7-46Б\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Турсунов Жасурбек Рустамович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Тема производственной практики \_\_ Генерация трехмерного ландшафта\_\_\_\_

Направленность практики (учебный, исследовательский, практический, производственный)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_производственный\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_\_\_\_\_\_кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Техническое задание*** Разработать алгоритм генерации трехмерного ландшафта в реальном времени с учетом местности.

***Оформление производственной практики:***

Расчетно-пояснительная записка на \_\_\_\_\_ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2020г.

**Руководитель производственной практики** \_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_А. В. Куров \_

(Подпись, дата)                             (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_**Ж.Р. Турсунов**\_\_\_**

(Подпись, дата)                             (И.О.Фамилия)

Оглавление

[**Введение** **3**](#_Toc52197414)

[**1.** **Аналитическая часть** **5**](#_Toc52197415)

[**1.1** **Предметная область** **5**](#_Toc52197416)

[1.2 Представление данных о ландшафте 5](#_Toc52197417)

[1.3 Алгоритмы генерации ландшафта 8](#_Toc52197418)

[**2.** **Конструкторская часть** **11**](#_Toc52197419)

[2.1 Генерация рельефа с помощью шума Перлина 11](#_Toc52197420)

[2.2 Генерация рельефов с использованием Холмового алгоритма (Hill Algoritm) 14](#_Toc52197421)

[**Заключение** **20**](#_Toc52197422)

[**Список литературы** **21**](#_Toc52197424)

**Введение**

Роль компьютеров с каждым днём становится всё больше и больше. С каждым разом увеличивается область применения этой технологии. Для каждой области применения необходимы свои собственные программы, для решения конкретных задач. Благодаря этому в настоящее время постоянно появляются новые предметы изучения и исследования. Это целая наука, занимающаяся построением реальных графических изображений посредством вычислительных действий. Компьютерная графика закрепила за собой много важных задач, среди которых присутствуют такие, как визуализация результатов, полученных при обработке данных, моделирование реальных процессов и объектов. Военная область – одна из тех, кто не только первыми опробовали компьютеры, но и послужили в формирование компьютерной графики как науки. Например, для обучения и подготовки к действиям в боевых условиях пилотов самолетов, танков или какой-нибудь другой бронетехники, было необходимым создание симуляторов реальных технических средств. Было гораздо безопасно и эффективно, в плане военного бюджета посадить человека за симулятор для получения первоначальных навыков, чем на реальный объект. А для создания симулятора требовалось получать реалистичные изображения различных объектов, например, местности, на которой ведутся учения, причем эти изображения должны быть получены в реальном времени, то есть так, чтобы в зависимости от действий обучаемого соответственно менялись и параметры системы, такие как положение на местности, погодные условия. Спустя время стало ясно, что такие системы можно использовать не только для обучения новичков, но и для планирования реальных боевых операций на любой территории, о которой есть определенные данные.

«Трехмерное моделирование», ветвь 3D-моделирования, которая занимается проектированием и созданием трехмерных реалистичных изображений всех видов поверхностей и ландшафтов.

В настоящее время 3D-моделирование является важнейшей областью компьютерной графики, так как построение трехмерного изображения, близкого к реальному, является достаточно сложной задачей. Но, благодаря получаемым знаниям каждый день, невообразимому быстрому росту производительности вычислительных систем, эта область активно развивается.

Для простоты область 3D-моделирования можно разделить на 2 подобласти. Одна из них – это генерация трехмерных изображений в реальном времени, а другая – генерация высоко реалистичных трехмерных изображений. Высоко реалистичные изображения требуют очень больших объемов вычислений и производительности от машин, в то время как в первой подобласти применяется система условных допущений, позволяющая применять более простые алгоритмы, и тем самым значительно уменьшить объем вычислений, что, правда, сказывается на качестве изображения. Как уже было сказано, алгоритмы 3D-моделирования делятся на две группы.

В первую входят такие алгоритмы, как алгоритм Робертса, алгоритм Варнока, алгоритм, использующий z-буфер, которые в свою очередь используют простейшие алгоритмы построения линий, треугольников, закрашивание замкнутых областей. То есть алгоритмы, изученные нами в ходе курса «Компьютерной графики». Также к этой группе можно отнести алгоритмы, выполняющие отсечение.

При проектировании были сформулированы и поставлены следующие задачи:

1. Постановка задачи
2. Описание сущностей
3. Разработка алгоритма и структур данных
4. Разработка архитектуры системы
5. Создание программы
6. Тестирование
7. **Аналитическая часть**
   1. **Предметная область**

Безусловно главной целью является создание трехмерного ландшафта. Ландшафт может быть представлен несколькими способами, а именно использованием регулярной сеткой высот или же иррегулярной сеткой вершин и связей. Параметры, характеризующие ландшафт не менее важны чем сам способ представления данных о местности. В этом проекте допускается возможность создания двух видов местности: острова и долины. Такие параметры как: размер, тип и метод генерации местности – являются базовыми. Но этими параметрами нельзя ограничиваться. Чтобы предать ландшафту более реалистичный вид, вводятся новые параметры, которые отвечают за текстуру неба и воды, за освещение, за материал местности. В последующих разделах подробно описаны все достоинства и недостатки методов хранения данных о ландшафте и об алгоритмах их генерации.

* 1. **Представление данных о ландшафте**

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

* Первый - использование регулярной сетки высот (другое название Карта Высот - *HeightMap*).
* Второе - использование иррегулярной сетки вершин и связей, их соединяющих (т.е. хранение простой триангулизированной карты).
* Третий - 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.

В этом проекте был выбран первый способ представления ландшафтов.

Данные представлены в виде двухмерного массива. Уже заданы две координаты (x, y - по высоте и ширине массива), и третья координата задается значением в конкретной ячейке, это высота.

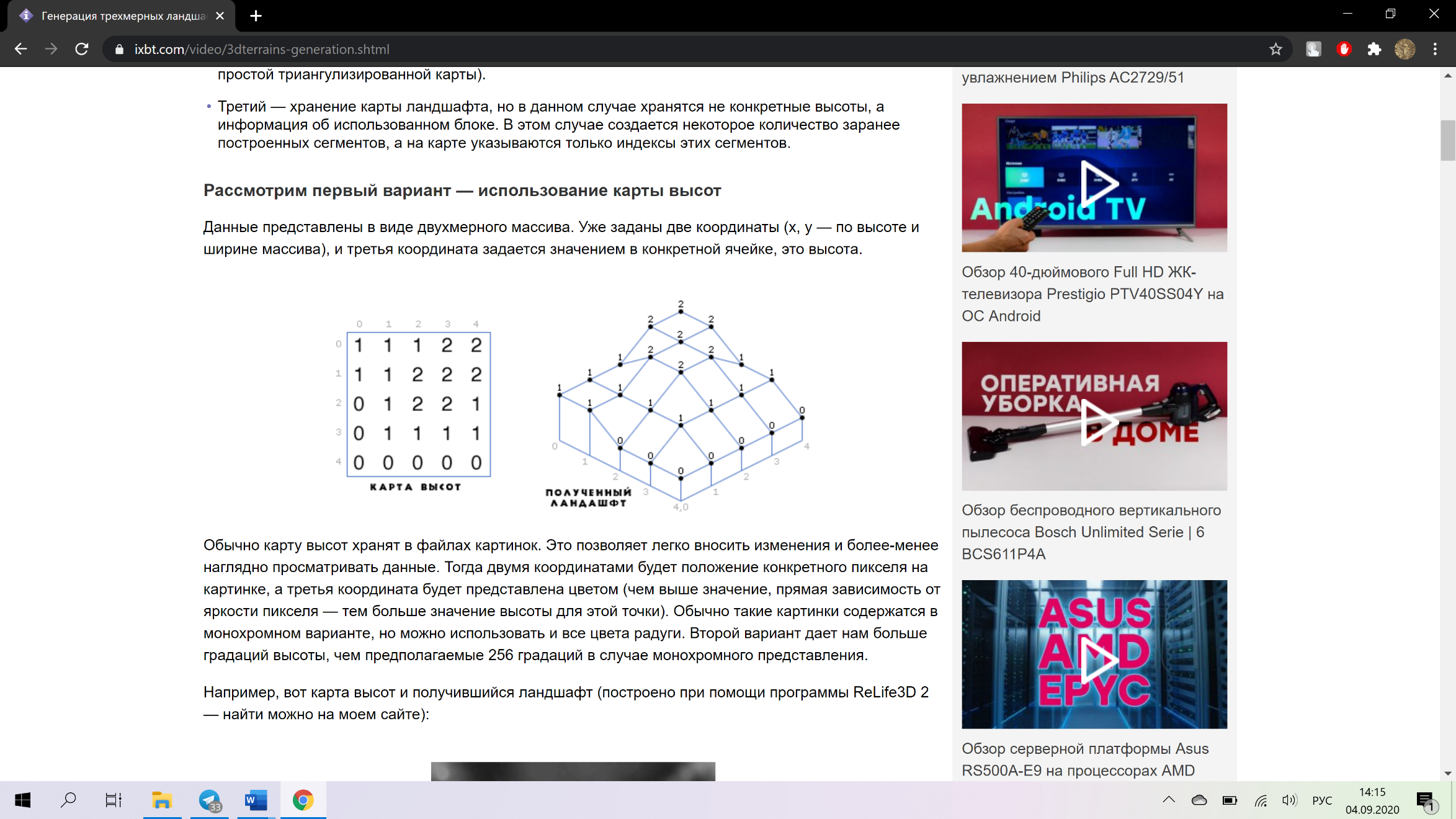


Рисунок 1.1

Плюсы данного подхода:

1. Простота реализации: легкость нахождения координат (и высоты) на карте, простая генерация ландшафта по карте высот или методом шума Перлина.
2. Наглядность: в любой программе просмотра графических файлов можно сразу увидеть или изменить всю информацию.
3. Скорость: благодаря конвейерной архитектуре процессора, просчет и вывод таких карт высот производится очень быстро (динамическое освещение, так как освещенность вершины напрямую зависит от расстояния от этой вершины до источника освещения).

Также есть минусы:

Большое количество избыточных данных (особенно для поверхностей, близких к плоским).

Еще один способ представления данных для ландшафтов про которую стоит упомянуть — иррегулярная сетка вершин и связей их соединяющих. Зачастую такие решения применяются в специализированных пакетах для игр. И хранятся в виде трехмерных моделей.

Это дает основной выигрыш по сравнению с картами высот:

* Используется значительно меньше информации для построения ландшафта. Здесь необходимо хранить только значения высот каждой вершины и связи эти вершины соединяющие. Это дает выигрыш в скорости при передаче огромных массивов информации, в процессе визуализации ландшафта.

Но также имеются множество недостатков:

* Алгоритмы построения ландшафтов в основном предназначены для регулярных карт высот. Оптимизация таких алгоритмов под этот способ потребует значительных усилий;
* Сложности при динамическом освещении — вершины расположены достаточно далеко друг от друга и неравномерно;
* Хранение, просмотр, модификация такого ландшафта также представляет сложности. При использовании карт высот достаточно применить простые и "стандартные" средства пиксельной графики. Например MS Paint. Тут же потребуются "весомые" пакеты.

Ниже приведено графическое представление данного принципа:

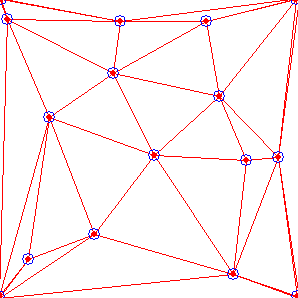


Рисунок 1.2.

В данном способе также используются карты высот. Только вместо высот в ней хранятся индексы *ландшафтных сегментов*. Способ представления этих сегментов не имеет значения. Они могут быть и регулярными, и иррегулярными.

Это дает следующие преимущества:

1. Возможность представления огромнейших открытых пространств;
2. 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);
3. 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 (так называемые LOD ландшафты — LOD — Level Of Detail).

Минусов данного способа:

1. Первый минус — проблема стыковки разных сегментов.
2. Второй — неочевидность данных. Невозможно моментально представить, как это должно будет выглядеть в программе.
3. Следовательно, встает и проблема модификации. Если при первом варианте вы используете *Paint*, во втором *3DMax*, то тут вам, скорее всего, потребуются писать редактор.
   1. **Алгоритмы генерации ландшафта**

Здесь показаны все достоинства и недостатки всех алгоритмов для генерации ландшафта. Для начала рассмотрим самый простой и примитивный алгоритм, который так и называется *Простой алгоритм*.

Алгоритм заключается в том, что программа в случайных координатах заполняет карту прямоугольниками случайного размера. Карта имеет вид двухмерного массива, представляющего карту высот нашего ландшафта. Этот алгоритм не дает реалистичный ландшафт, так что данный метод поможет новичку ознакомиться с этой тематикой. Следует заметить, что главными достоинства этого метода, является его простота и скорость исполнения. А из недостатков нужно подчеркнуть его примитивность. Также при маленьком шаге заполнения карты, ландшафт становится «квадратным».

Следующий метод генерации ландшафта – это *Холмовой алгоритм* Суть этого алгоритма заключается в «поднятии» холм. Так как холм похож на половину шара, поэтому достаточно поднять его половину. Чем больше количество холмов, тем более реалистично выглядит местность. Чтобы предать более реалистичный вид, необходимо провести «Долинизацию», «Нормализацию», «Сглаживание». Этот метод хорош тем, что здесь используется самые простые формулы для создания холма или для его модификации.

Также в разных источниках встречается понятие *шум Перлина*. Шум – набор случайных пикселей. Его можно часто наблюдать по телевизору, когда пропадает сигнал.

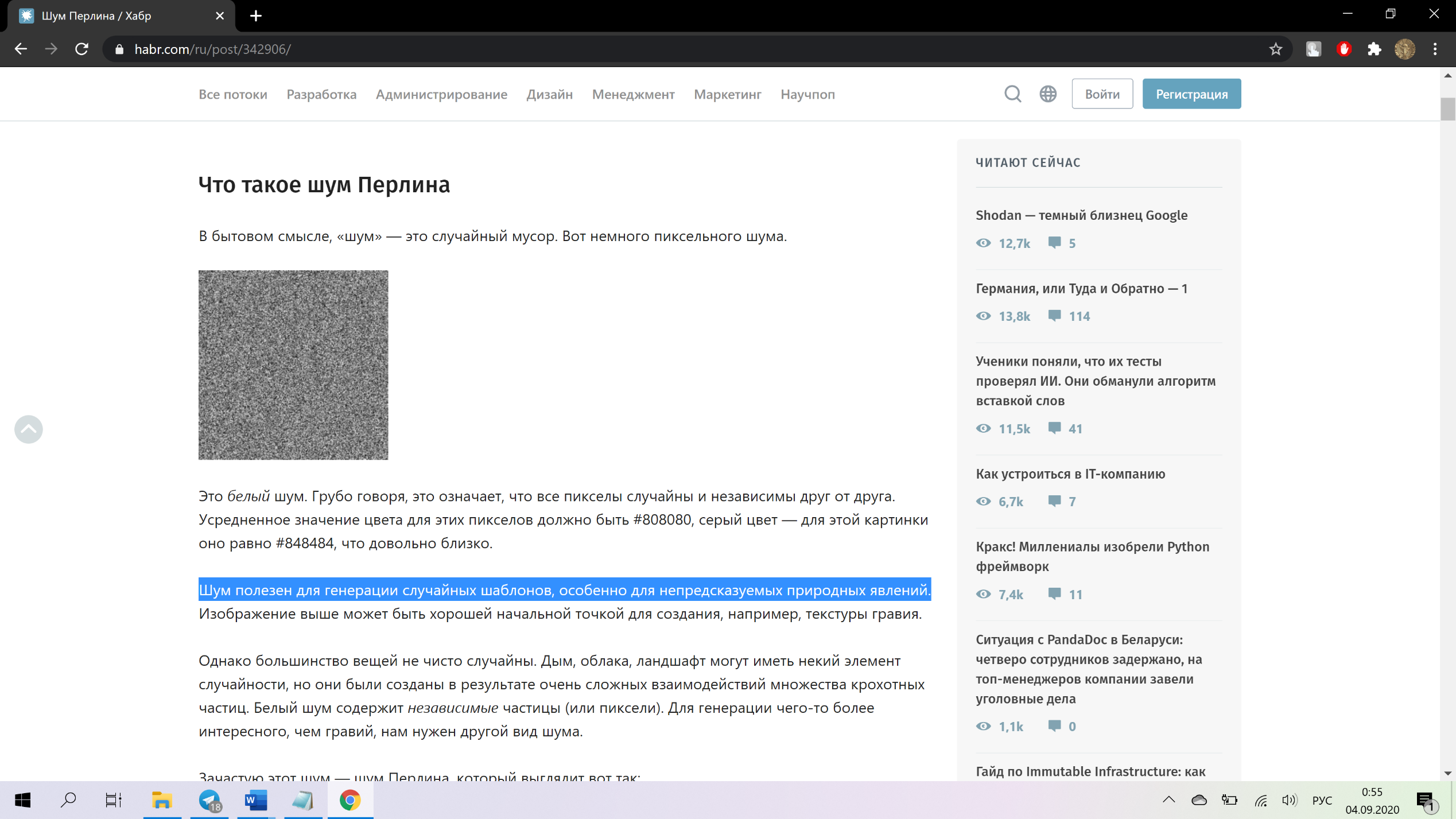


Рисунок 1.3.

Шум полезен для генерации случайных шаблонов, особенно для непредсказуемых природных явлений. Однако большинство вещей не чисто случайны. Дым, облака, ландшафт могут иметь некий элемент случайности, но они были созданы в результате очень сложных взаимодействий множества крохотных частиц. Белый шум содержит *независимые* частицы (или пиксели). Для генерации чего-то более интересного, нужен другой вид шума. Зачастую этот шум — шум Перлина, рисунок которого, приведен ниже:

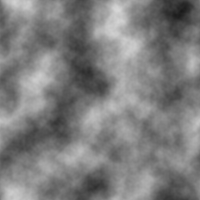


Рисунок 1.4.

Для генерации шума Перлина в одномерном пространстве необходимо для каждой точки этого пространства вычислить значение шумовой функции, используя направление градиента (или наклон) в указанной точке.

Функция «Perlin noise» проводит генерацию текстур методом генерации псевдослучайных чисел, однако все визуальные детали текстуры имеют одинаковый размер.

Это свойство делает шум Перлина легко управляемым; множество масштабированных копий шума Перлина могут быть вставленными в математические выражения для создания самых разнообразных процедурных текстур.

В отличие от растровых текстур, шум Перлина является процедурной текстурой, и поэтому он не занимает память, но вместе с тем исполнение алгоритма требует неких вычислительных ресурсов.

Выше были выделены три алгоритма генерации ландшафта и было решено, что в этом проекте будет сделано некое разнообразие, то есть генерация местности будет происходить двумя методами. Было решено сделать так, потому что «Холмовой алгоритм» и шум «Перлина» довольно одинаковы с точки зрения эффективности и производительности. В них нет сложных математических формул, всё довольно просто и понятно. Поэтому пользователю даётся возможность генерировать любым доступным способом. Какими бы не были схожими эти методы, в них все равно есть различие. Поэтому добавляется еще одна задача к проекту, а именно сделать тестирование обеих алгоритмов.

1. **Конструкторская часть**

**2.1** **Генерация рельефа с помощью шума Перлина**

При разработки данной программы возникает вопрос: откуда брать информацию для генерации карты высот? Конечно же можно просто загружать монохромное изображение и на его основе генерировать ландшафт. Но если нужно каждый раз генерировать разные карты высот (например, для компьютерных игр или демонстрационных программ, таких как эта), то на помощь приходит следующий метод на основе шума Перлина.

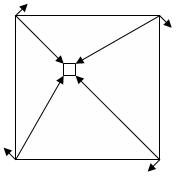


Рисунок 2.1.1.

Изображение (или какой-либо другой объем) – вне зависимости от диапазона значений его элементов – полностью накрывается сеткой, представляющей диапазон вещественных чисел.

Таким образом, создается шум на сетке, представляющей по всему изображению значения между 0 и 4. Каждое число порождает линию сетки, а значит, все стороны каждого квадрата последней имеют длину, равную одной единице.

Выбранный масштаб влияет на сложность шума. Большое число квадратов на сетке изображения создает более «плотно упакованный» шум, подобный белому шуму на экране плохо настроенного телевизора. Меньшее число квадратов на сетке порождает «клубящийся» шум, внешне похожий на облака.

В каждой точке на сетке строится случайный вектор нормали. Это обычный двумерный вектор единичной длины, который указывает в случайном направлении в пределах каждого из квадратов. Традиционный способ создания таких векторов – организация справочной таблицы из 256 векторов, которые охватывают полный круг, и последующий случайный выбор одного из них для каждой точки на сетке. Это гарантирует распределение векторов, которые могут с равной вероятностью указывать в любом направлении. Далее для каждого пикселя изображения находится та из ячеек сетки, где он находится.

Таким образом, определяется значение, которое основано исключительно на данных этой ячейки. Следующий шаг – создать четыре диагональных вектора, соединяющих углы ячейки с текущим пикселем.

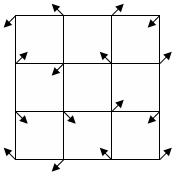




Рисунок 2.1.2.

Каждый угол ячейки сетки теперь является базой для двух векторов – случайного единичного вектора и вектора в направлении пикселя, который необходимо построить. Для каждой пары таких векторов находиться скалярное произведение. Оно даст скалярное значение высоты каждого из углов сетки. Далее необходимо объединить эти четыре значения и найти высоту пикселя, который надо сгенерировать. Делать это можно по-разному, получая различные результаты, однако чаще всего используется взвешенная интерполяция четырех значений с учетом близости текущей позиции к каждому углу сетки.

Основным плюсом использования шумовой функции при генерации ландшафта является то, что нет необходимости хранить карту высот, а достаточно лишь использовать данные справочной таблицы векторов, – все остальное для восстановления конечной карты высот ландшафта сделает шумовая функция.

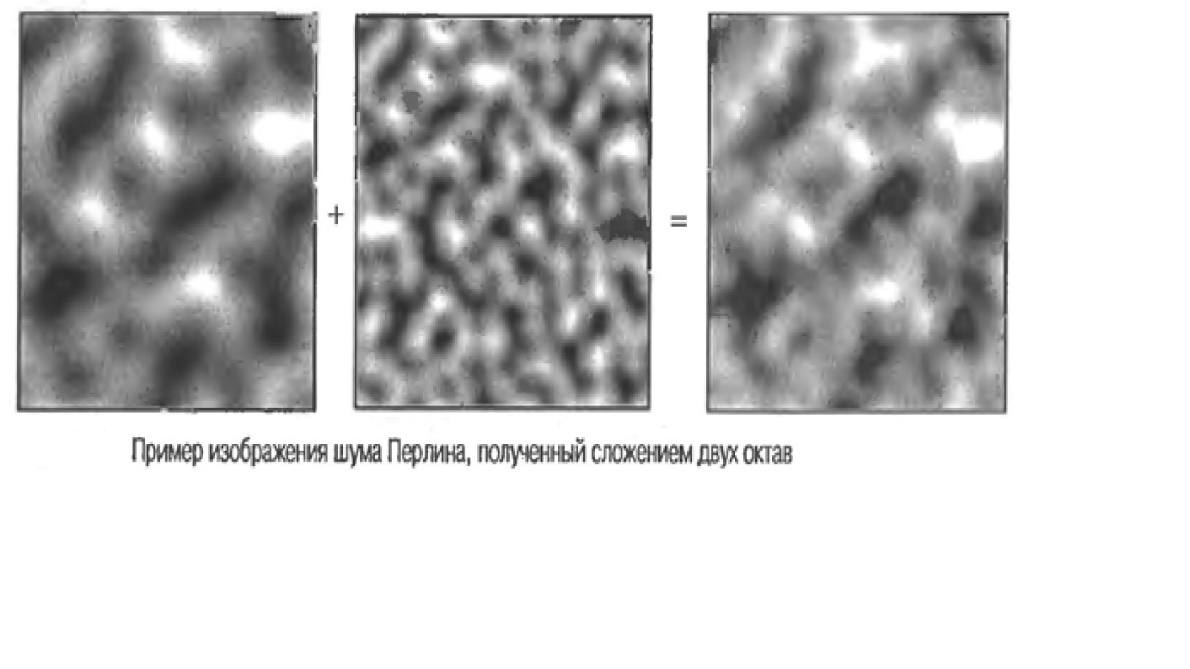


Рисунок 2.1.3.

## 2.2 Генерация рельефов с использованием Холмового алгоритма (Hill Algoritm)

Это простой итерационный алгоритм, основанный на нескольких входных параметрах.

Алгоритм изложен в следующих шагах:

1. Создание и инициализация двухмерного массива с нулевым уровнем (все ячейки заполнены нулями);
2. Выбирается случайная точка 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. Возвращение ко второму шагу и так далее до выбранного количества шагов. От него будет зависеть внешний вид ландшафта;
5. «Нормализация» ландшафта;
6. «Долинизация» ландшафта.

Первый, второй и четвертые шаги тривиальны, пятый и шестой шаг будет рассмотрен далее. Теперь же изучим третий шаг. Фактически холм в этом случае половина шара, чем больше радиус - тем больше холм (и выше). Математически это похоже на перевернутую параболу.

http://www.ixbt.com/video/theor/3dterrains-generation/picture_07.png

здесь (x1, y1) - заданная точка, r - выбранный радиус, (x2, y2) - высота холма.

Вот как выглядит одиночный холм:

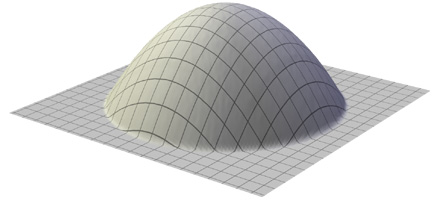


Рисунок 2.3.1

Чтобы сгенерировать ландшафт полностью необходимо построить множество таких холмов. Но есть еще две вещи на которые необходимо обратить внимание. Первое - игнорирование отрицательных значений высоты холма. Второе - при генерации последующих холмов лучше добавлять полученное значение для данного холма к уже существующим значениям. Это позволяет строить более правдоподобный ландшафт, нежели правильно очерченные округлые холмы. Ниже представлен ландшафт при большом количестве итераций:

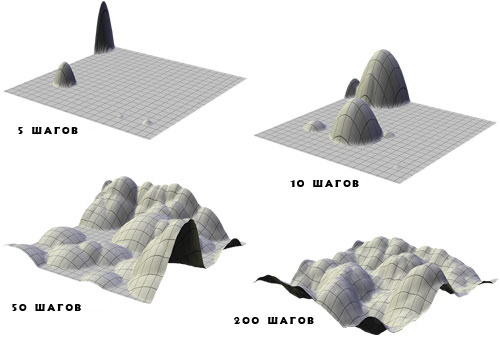


Рисунок 2.3.2.

**Нормализация ландшафта**

При генерации значений для ландшафта не учитывались выходы этих значений за некоторые пределы (например, если ландшафт будет храниться в монохромной картинке, то необходимо, чтобы все значения находились в пределе от 0 до 256). Для этого необходимо произвести нормализацию значений. Математически нормализация — это процесс получения значений из одного предела, и перевод его в другие пределы. Вот как это выглядит графически:

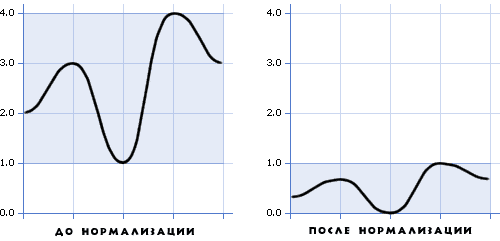


Рисунок 2.3.3.

Чтобы это сделать произведем следующие действия:

* Сперва проходим по всему массиву и запоминаем наибольшее и наименьшее значения;
* После этого производится нормализация конкретных значений в пределах от 0 до 1. Формула выглядит так:

picture_11

**Долинизация ландшафта**

Данный ландшафт уже можно использовать, но если присмотреться, то в нем достаточно мало долин. Склоны холмов излишне крутые, необходимо сделать их более пологими. Идея "Долинизации" состоит в следующем - взять от каждого значения квадратный корень. Это в большей степени влияет на средние значения, практически не затрагивая минимумов и максимумов. Графически это выглядит так:

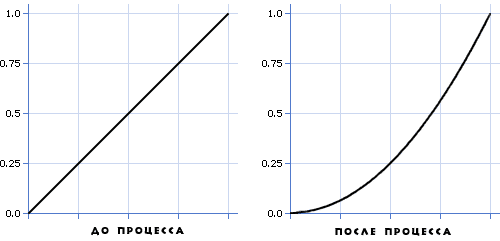


Рисунок 2.3.4.

А вот, как это повлияло на наш ландшафт:

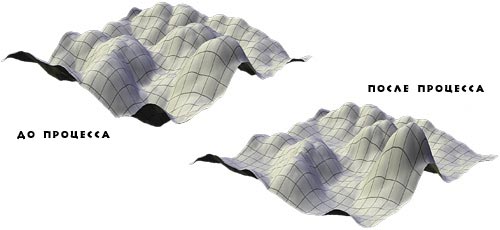


Рисунок 2.3.5.

В основном, рассмотренные выше алгоритмы предназначены для создания простого холмистого или гористого ландшафта. Но существуют и другие типы ландшафтов. Например, острова, озерные ландшафты. Их можно реализовать достаточно просто:

1. Создание простого, достаточно холмистого ландшафта;
2. Перемещение уровня воды вверх или вниз. .

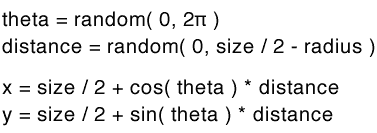
Так как уже было сказано выше, что ландшафт будет генерироваться двумя видами (остров и холмы). Ввиду того, что холмы уже рассмотрены. Необходимо понять, как правильно создать остров, чтобы он был реалистичным.

Во многих случаях используется уже рассмотренный алгоритм для генерации ландшафтов. Но иногда необходимо сгенерировать острова, или остров. В этом поможет слегка модифицированный алгоритм.

В исходном алгоритме выбиралась центральная точка случайным образом, и она могла располагаться в любой части ландшафта. Теперь же необходимо, чтобы холмы были расположены ближе к центру.

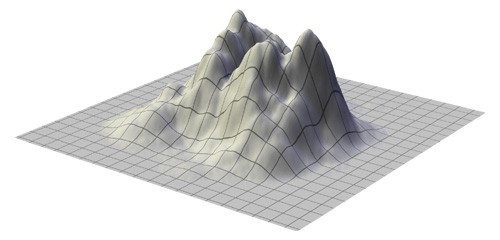
Чтобы сделать это, вводятся две переменные (которые потом будем случайным образом изменяться), назовем их расстояние и угол. Расстояние будет означать, как далеко от центра находится центральная точка для одиночного холма. Оно может изменяться от ноля (прямо по центру карты высот) до половины величины карты высот минус радиус холма. Это позволит избежать ситуаций пересечения холмов с краем карты высот. Угол будет показывать, в каком направлении от центра нужно будет поставить холм. Изменяется угол будет в пределах от 0 до двух Пи. Используя эти два значения, можно получить значения (x, y) для центральной точки конкретного холма и использовать их как и в простом алгоритме.

Ниже представлен способ получения значений для x и y:



здесь **size** - размер карты высот, **distance** - расстояние, **theta** - угол. Важно, чтобы радиус был меньше половины размера карты высот.

Поработав с этими величинами, можно получить довольно прилично выглядящий остров:



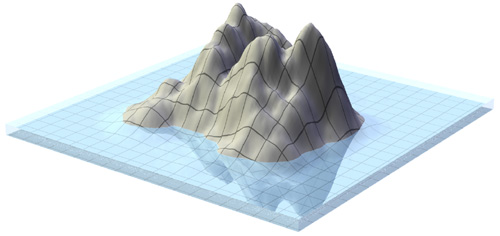


Рисунок 2.3.7

Выше были разобраны некоторые алгоритмы построения карт высот для ландшафтов. Рассмотрим еще некоторые сопутствующие операции:

1. Сглаживание (или размытие). Низкочастотный фильтр для уменьшения эффектов угловатости. При многократном применении позволяет добиться очень гладких очертаний ландшафта.
2. Превращение гористой местности в холмистую. Различия выделяются по очертанию вертикальных разрезов у вторых более пологие края.
3. Создание пляжей и отмелей в случае с островами и берегами. Для этого места соприкосновения с водой сглаживают. Хотя могут существовать и скалистые пляжи и просто скалы.

**Заключение**

В ходе практики разработана аналитическая часть курсовой работы по теме «Генерация трехмерного ландшафта», сделан анализ методов, подходящих для решения поставленной задачи, а также был выбран метод, которым будет реализовываться курсовая работа.

**Список литературы**

1. Генерация трехмерного ландшафта,/ [Электронный ресурс]. - Режим доступа: URL: https://www.ixbt.com/video/3dterrains-generation.shtml (дата обращения: 15.07.2020).
2. Простой и быстрый алгоритм генерации,/ [Электронный ресурс]. - Режим доступа: URL: https://habr.com/ru/post/319226/ (дата обращения: 15.07.2020).
3. Алгоритм «diamond-square» для построения фрактальных ландшафтов,/ [Электронный ресурс]. - Режим доступа: URL: https://habrahabr.ru/post/111538/ (дата обращения: 21.07.2020).
4. Diamond-Square algorithm,/ [Электронный ресурс]. - Режим доступа: URL: https://en.wikipedia.org/wiki/Diamond-square\_algorithm/ (дата обращения: 21.07.2020).
5. Red Blop Games Polygonal Map Generation for Games,/ [Электронный ресурс]. - Режим доступа: URL: http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/ (дата обращения: 29.07.2020).