Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт математики и фундаментальной информатики
Базовая кафедра вычислительных и информационных технологий

УТВЕРЖДАЮ
Заведующей кафедрой
____ /В.В. Шайдуров

«___» _____2019г.

БАКАЛАВРСКАЯ РАБОТА

Направление 02.03.01 Математика и компьютерные науки

РАЗРАБОТКА АЛГОРИТМА ГЕНЕРАЦИИ ЛАНДШАФТА

Научный руководитель
кандидат физико-математических наук,
доцент _____/С.Н. Баранов

Выпускник _____/В.Е. Репп

Красноярск 2019

Содержание

Введение

В настоящее время очень популярной стала компьютерная графика по многим причинам. Во-первых, постоянно растущие компьютерные мощности дают большие возможности для реализации разного рода идей. Во-вторых, активно развивается киноиндустрия, становится популярной мультипликация, для создания которых используется компьютерная графика. Помимо этого, данное направление может использоваться в моделировании разных ситуаций, где необходимо построить визуальную модель события или объекта на местности. Также может использоваться при реализации разных дизайнерских проектов.

Сейчас очень интенсивно развивается компьютерная игровая индустрия. Многие разработчики игр пользуются разными программами обеспеченья, средами разработки, которые упрощают и ускоряют процесс создания видеоигр. В таких программах, как правило, уже реализовано много разных возможностей, которые и упрощают процесс создания продукта. Это могут быть различные плагины, функции, объекты, сценарии и т.д. Например, при создании разных уровней, разработчики могут обратиться к таким возможностям, чтобы сэкономить свои ресурсы.

Цель моей работы - создать программу, которая будет генерировать псевдослучайным образом 3D ландшафт в реальном времени. От разработчиков потребуется ввести несколько параметров для генерации ландшафта и на выходе они получат нужный результат, то есть сгенерированный ландшафт, с которым можно работать.

Еще один вариант использования данной программы, это использования ее в дизайнерских проектах. При создании своей работы, дизайнеру не придется тратить время на поиски нужного ландшафта, если он необходим, а просто ввести параметры и получить необходимый результат, что сэкономит время и силы.

Таким образом, мы видим, что созданная мной программа актуальна в разных направлениях, связанных с компьютерной графикой и генерацией ландшафта.

Исходя из поставленной цели, были определены следующие задачи:

- Задача 1: определение способа генерации ландшафта;
- Задача 2: определение способа представления данных о ландшафте;
- Задача 3: определение алгоритма построения ландшафта;
- Задача 4: выбор платформы для реализации данного алгоритма;
- Задача 5: создание программы.

На основании поставленных нами задач будет осуществляться работа по данному проекту.

1. Конструкторский раздел

1.1 Способы генерации ландшафта

Для начала необходимо определить, какие способы генерации ландшафта существуют, и выбрать наиболее подходящий для реализации данного проекта.

Разберем два основных способа генерации ландшафта.

Первый способ заключается в использовании монохромного изображения, то есть ландшафт строится на основе заранее созданного монохромного изображения. Монохромное изображение – это изображение, содержащее свет одного цвета.

Положительные стороны данного способа в том, что не нужно генерировать монохромное изображение, то есть не нужны большие вычислительные мощности компьютера. Но тогда потребуется больше памяти компьютера, так как данные изображения нужно хранить. Так же, если необходимо построить определенный ландшафт, то сначала придется подобрать необходимое изображение. Если такого нет, то сгенерировать или найти подходящее изображение, что является не самым эффективным способом.

Второй способ, это использование алгоритма генерации шума и последующего использования полученного результата для генерации ландшафта. Его положительные стороны: экономия памяти, так как не нужно хранить монохромные изображения, и можно задать необходимые параметры, чтобы получить нужный ландшафт. Но у данного способа есть свой недостаток, это требования к вычислительным мощностям компьютера, так как необходимо сгенерировать шум для построения ландшафта.

Для реализации своего проекта я выбрал второй способ генерации ландшафта. Так как данный способ даст возможность сгенерировать ландшафт с необходимыми параметрами.

1.2 Представление данных о ландшафте

После того, как выбран способ генерации ландшафта, необходимо определить, как представляются данные о ландшафте в компьютере. Есть несколько способов хранения информации о ландшафте:

Первый – использование регулярной сетки высот (другое название Карта Высот - HeightMap).

Второй – использование иррегулярной сетки вершин и их соединяющими связями (т.е. хранение триангулизированной карты поверхности).

Третий – хранение карты ландшафта, т.е. информации об использованном сегменте.

Рассмотрим каждый способ хранения информации о ландшафте по отдельности, чтобы лучше понять, какой из них наиболее подходящий для реализации данного проекта.

Использование регулярной сетки высот.

Данный способ заключается в том, что данные представляются в виде двухмерного массива. Каждый элемент массива имеет свой индекс, данный индекс соответствует координате расположения точке. В массиве хранится информация о высоте всех точках.

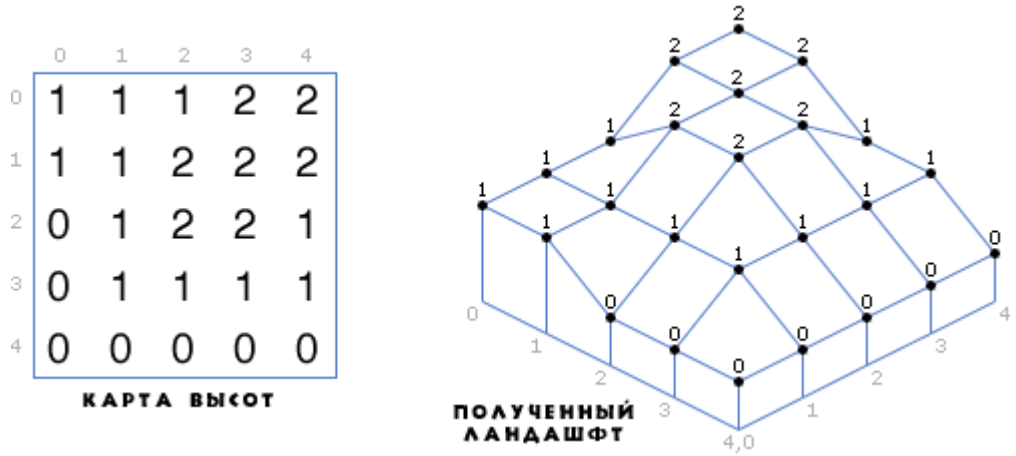

Рисунок 1. Пример карты высот и полученного ландшафта на ее основе.

Обычно карты высот хранят в файлах картинках. Это позволяет легко менять параметры и наглядно просматривать данные. Как правило, для этого используются монохромные изображения, но можно использовать все цвета.

У данного способа есть несколько плюсов:

- Наглядность, можно сразу увидеть всю информацию;
- Простота изменения данных;
- В таких картах можно хранить информацию не только о высоте;
- Так как точки расположены регулярно и близко друг к другу, то можно более правильно произвести динамическое освещение.

У данного способа есть один недостаток, это избыточность данных. Например, при создании простой плоскости будет использоваться информация множества точек, хотя хватило бы трех, четырех.

Иррегулярная сетка вершин.

Данный метод представления данных чаще всего используется в специальных пакетах для работы с трехмерной графикой. Данные хранятся в виде трехмерных моделей.

Плюс данного метода - это использование нужного количества данных для построения ландшафта.

Но у этого метода есть много недостатков:

- В основном, алгоритмы построения ландшафта, предназначены для работы с картой высот. А перестройка таких алгоритмов под данный метод может оказаться достаточно сложной;
- Возникают сложности при динамическом освещении, так как вершины расположены далеко друг от друга и не равномерно;
- Хранение, просмотр и модификация данных для такого ландшафта вызывает свои трудности.

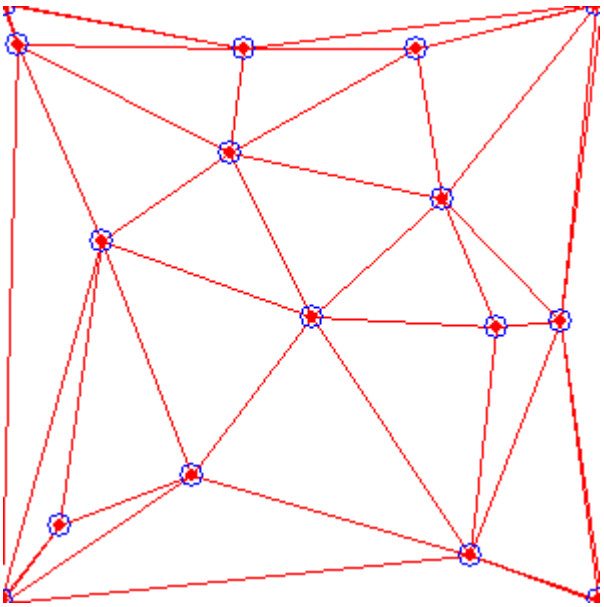

Рисунок 2. Пример иррегулярной сетки вершин.

Хранение карты ландшафта.

Данный метод использует карты высот. Только хранит информацию не о высоте, а о ландшафтных сегментах. Сегменты могут, представлены в виде регулярной, иррегулярной сетке или сочетать те и другие.

Преимущества данного метода:

- Возможность представления данных огромных территорий;
- Кроме хранения информации о ландшафте, в таких блоках можно хранить информацию и о других объектах (зданиях, реках, пещерах и т.д.);
- Возможность создать данные для одного и того же сегмента с разным уровнем детализации.

Недостатки данного метода:

- Проблемы взаимодействия таких сегментов;
- Данные не могут дать наглядность;
- Проблемы модификации. Так как, если для разных сегментов применяются разные редакторы, то не получится изменить все данные в одном редакторе.

Для данного проекта наиболее удобным будет представление данных о ландшафте в виде карты высот.

1.3 Алгоритмы генерации ландшафта

Далее рассмотрим несколько алгоритмов генерации ландшафта, проанализируем их и сделаем вывод, какой наиболее удобен для данной программы.

Генерация ландшафта с использованием Холмового алгоритма (Hill Algorithm)

Это простой итерационный алгоритм, который зависит от нескольких параметров. Алгоритм заключается в следующих шагах:

2. Создается двумерный массив и инициализируется нулевым уровнем;
3. Берется случайная точка и заранее заданный радиус;
4. В выбранной точке «поднимается» холм заданного радиуса;
5. Возвращаемся ко второму шагу, и так до выбранного количества повторений;
6. Проводится нормализация ландшафта;
7. Проводится «долинизация» ландшафта. Склоны делаются более пологими.
8. Генерация одного холма.

Рассмотрим шаг с «поднятием» холма. В данном случае, холм, это половина шара с заданным радиусом. С математической стороны, это похоже на перевернутую параболу. Формула, для генерации одного холма выглядит следующим образом:

$$z = r^2 - ((x_2-x_1)^2 + (y_2-y_1)^2)$$

Рисунок 3. Формула генерации одного холма.

Где, (x1, y1) – заданная точка, r – выбранный радиус, (x2, y2) – высота холма. Одиночный холм выглядит следующим образом:

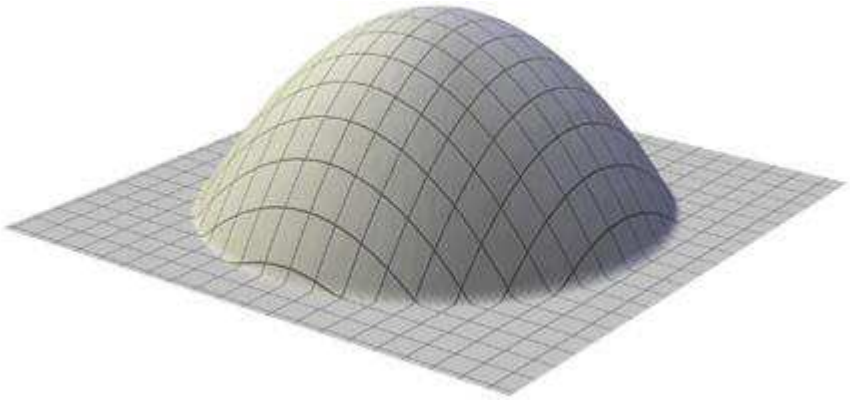

Рисунок 4. Пример одиночного холма.

Чтобы получился ландшафт, необходимо проделать несколько повторений, для создания холмов.

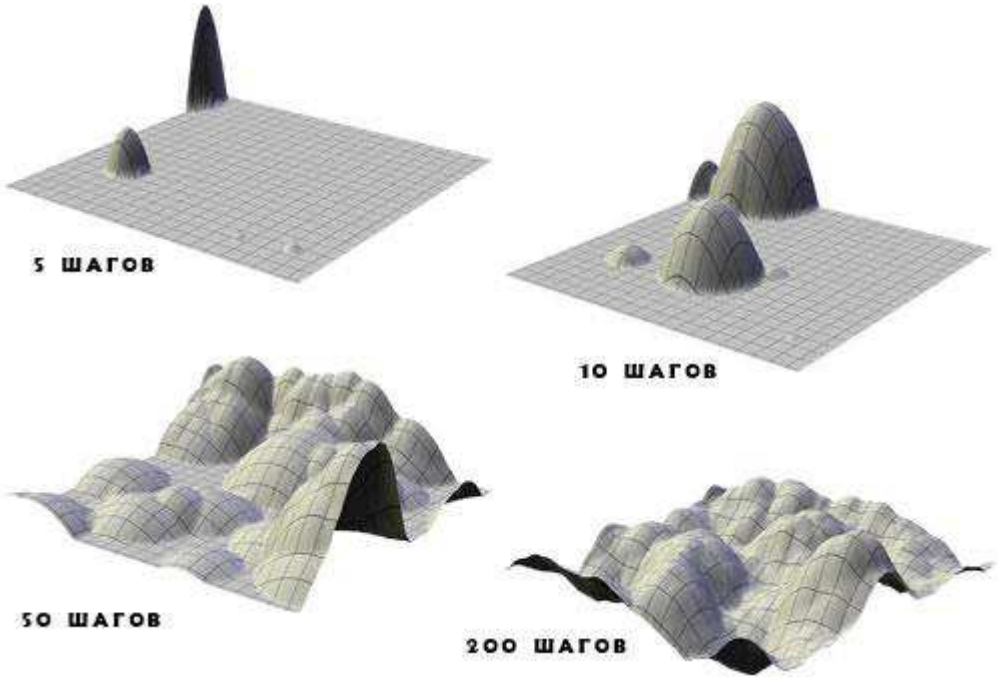

Рисунок 5. Пример ландшафта при большом количестве итераций.

Теперь, полученный ландшафт необходимо нормализовать.

Математически нормализация – это процесс получения значения из одного предела, и последующий перевод его в другие пределы. Графически выглядит так:

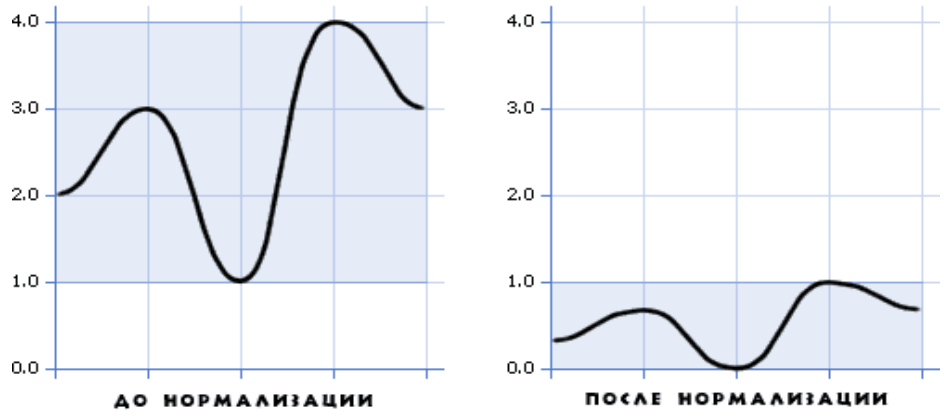

Рисунок 6. Нормализация данных.

Для этого необходимо сделать следующее:

- Проходим по всему массиву данных и запоминаем наибольшие и наименьшие значения;
- После этого, снова проходим по ландшафту и производим нормализацию конкретных данных по формуле:

$$z_{norm} = \frac{z - min}{max - min}$$

Рисунок 7. Формула нормализации ландшафта.

После этого остается только «долинизация» ландшафта. Этот шаг нужен для того, чтобы сделать все холмы более пологими. После нормализации, все значения находятся в пределах от 0 до 1. Тогда можно взять квадратный корень от значений, что повлияет в большей степени на средние значения, а не на максимальные и минимальные. Графически, это можно представить так:

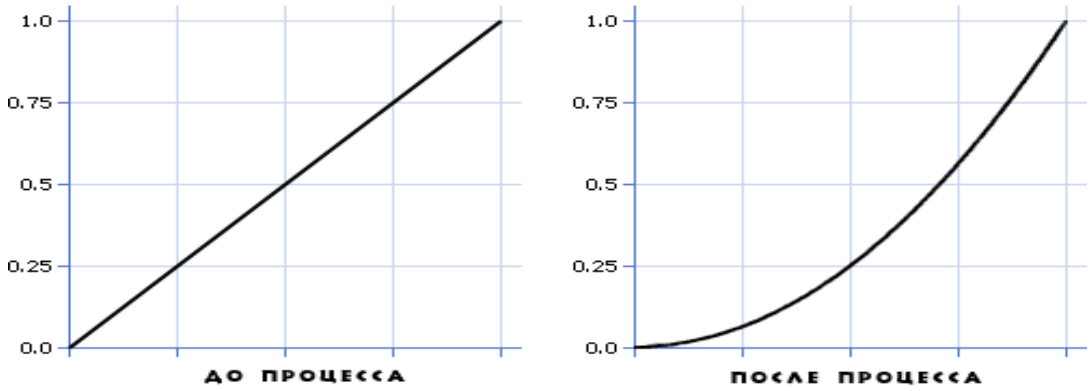

Рисунок 8. Пример взятия квадратного корня значения данных ландшафта.

На ландшафт это повлияет следующим образом:

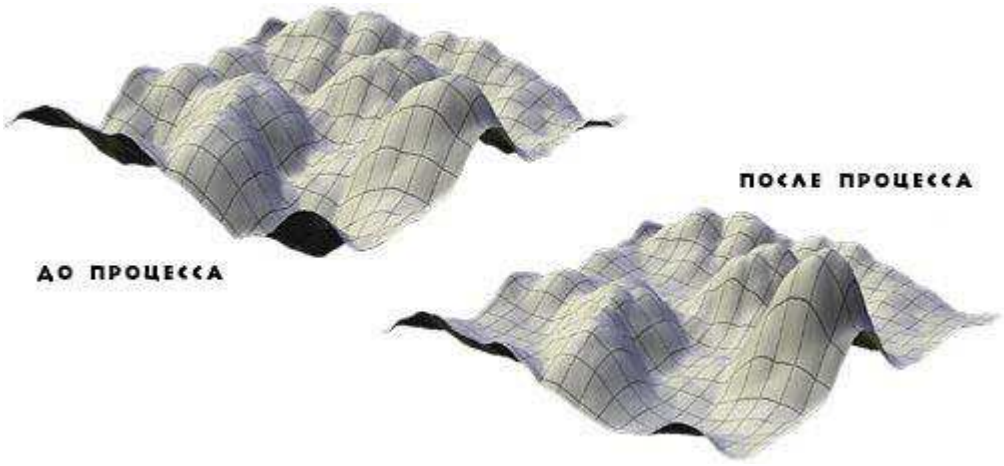

Рисунок 9. Пример применения «долинизации» ландшафта.

Данный алгоритм применяется в генерации простого холмистого или гористого ландшафта.

Генерация ландшафта с использованием шума Перлина.

Для начала, нужно понять, что такое шум. В обычной жизни, шум – это беспорядочное колебание звуковых волн, которые мы слышим. В математике, шум – это генерация случайных чисел. В компьютерной графике принято понимать под шумом следующее изображение:

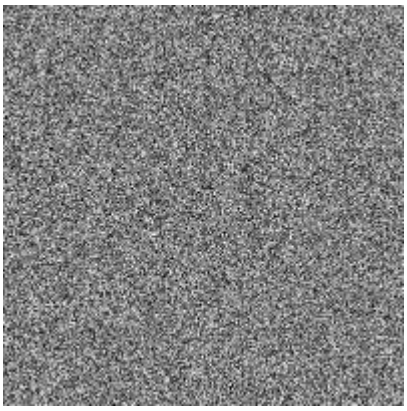

Рисунок 10. Пример шума в компьютерной графике.

Теперь можно перейти к шуму Перлина. Шум Перлина – это математический алгоритм по генерации процедурной текстуры псевдослучайным методом. Это градиентный шум, состоящий из случайно сгенерированных единичных векторов, расположенных в определенных точках пространства и интерполированных функцией сглаживания между

собой. Данный шум используется в компьютерной графике для увеличения реализма и генерации разных эффектов, таких как дым, облака, огонь и т.д.

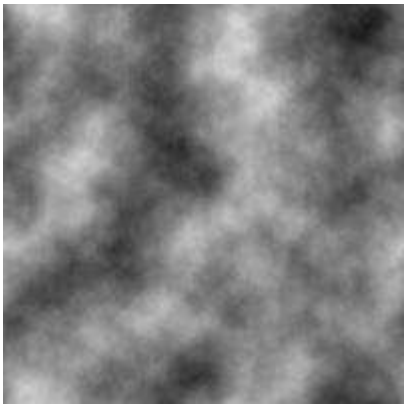

Рисунок 11. Пример шума Перлина.

Шум Перлина можно сгенерировать в любом количестве измерений. Объяснение алгоритма генерации шума Перлина, для наглядности, можно проиллюстрировать на обычном графике. По этому, сначала рассмотрим одномерный случай.

Сначала необходимо для каждого целого значения x=0, 1, 2, 3, … выбрать случайное вещественное из диапазона [-1;1]. Это будет угол наклона прямой, проходящей через данную точку на числовой оси.

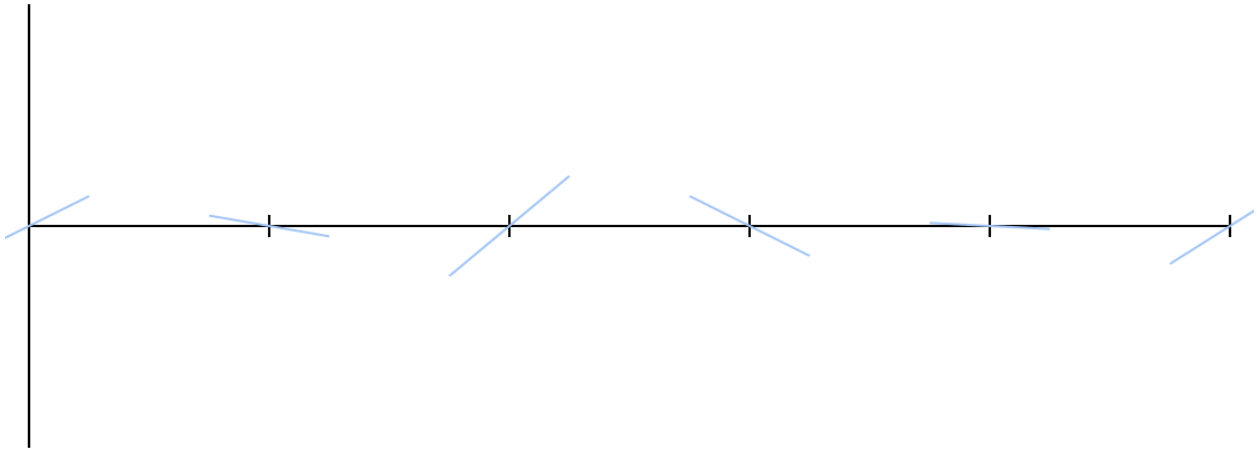

Рисунок 12. Пример генерации угла наклона.

Каждое значение x_0 лежит между двумя целыми значения x(x_0 є [0; 1], [1; 2] и т.д.) и между двумя наклонными прямыми. Пусть прямые имеют наклоны a и b. Тогда, уравнение прямой можно записать в виде $y = m(x - x_0)$, где m –наклон, x_0 – значение x, где прямая пересечет ось абсцисс.

Теперь, можно легко найти, где эти прямые пересекают заданное значение x. Например, если взять значение $x = 0$ и $x = 1$, получим точки ax и $b(x\text{-}1)$.

На графике показано, как оранжевые точки продолжают прямые, и места пересечения с осью абсцисс соответствующих значений:

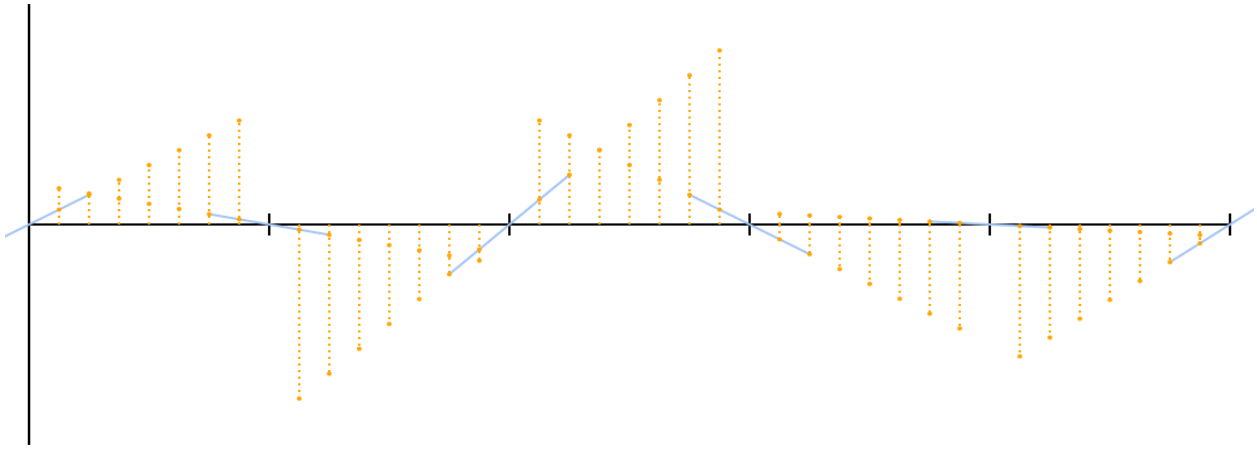

Рисунок 13. Пример пересечения заданных значений x.

Теперь появляется необходимость гладко соединить эти точки между собой. Для этого будет использоваться линейная интерполяция. Линейная интерполяция: если находимся в точке t на пути из точки A в B, линейная интерполяция будет равна $A + t(B - A)$.

Теперь, берем интерполированные точки. На графике они показаны красным:

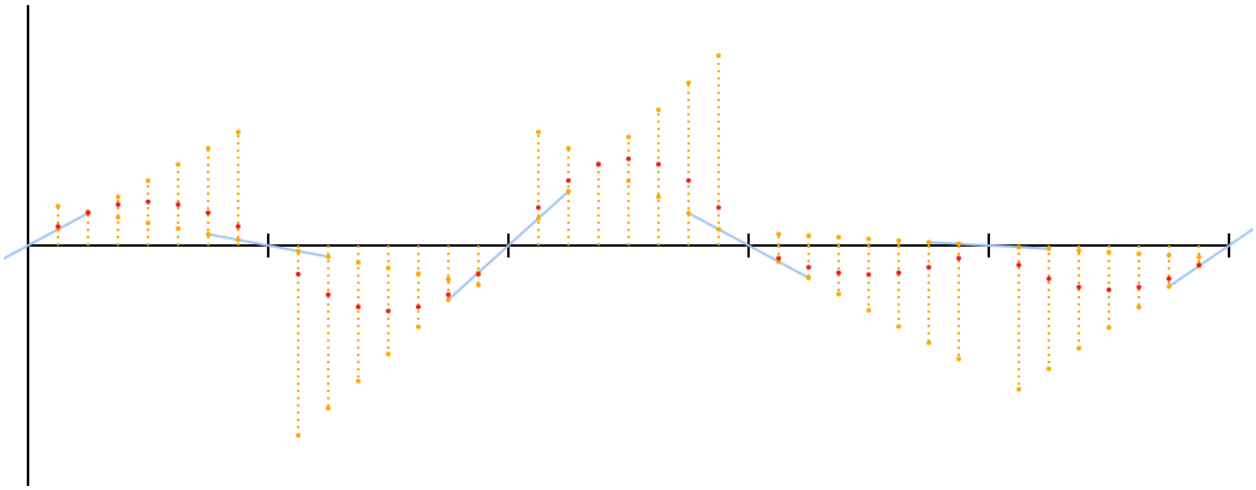

Рисунок 14. Пример интерполированных точек на графике.

Соединив эти точки со сглаживанием, получим следующий результат:

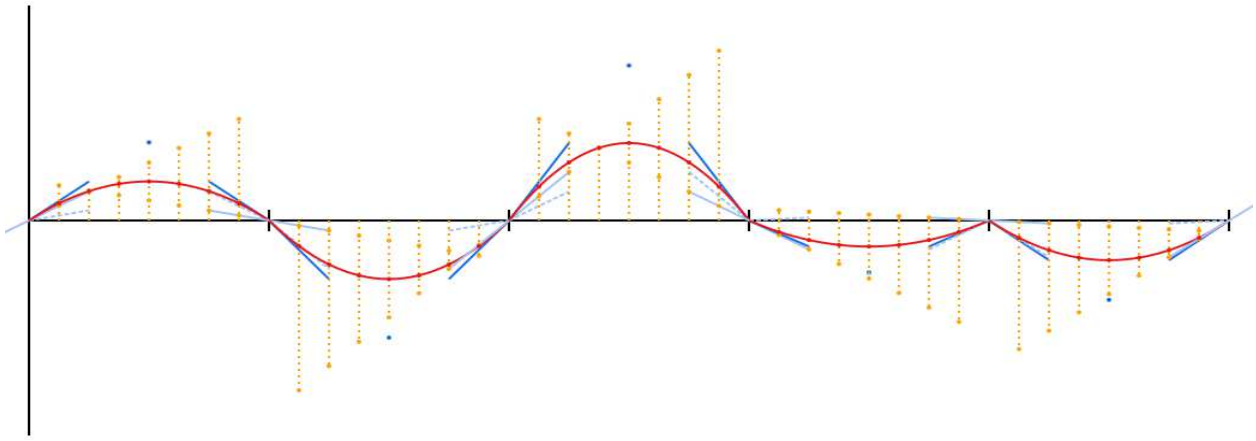

Рисунок 15. Пример соединения точек интерполяции со сглаживанием.

После соединения точек со сглаживанием, появляется проблема: эти кривые не переходят друг в друга плавно. В каждом сегменте виден излом. Получается, что линейной интерполяции не достаточно в этой ситуации. Так как, даже если находится достаточно близко к одному концу, влияние второго достаточно высоко, которое оттопыривает кривую от исходной случайной направляющей.

Для решения данной проблемы необходимо сделать следующее. До того, как происходит линейная интерполяция, можно сместить значение t при помощи функции smoothstep. Это s-образная кривая, похожая на диагональную прямую $x = y$, но сплющенная в нуле и единице.

Тогда график примет вид:

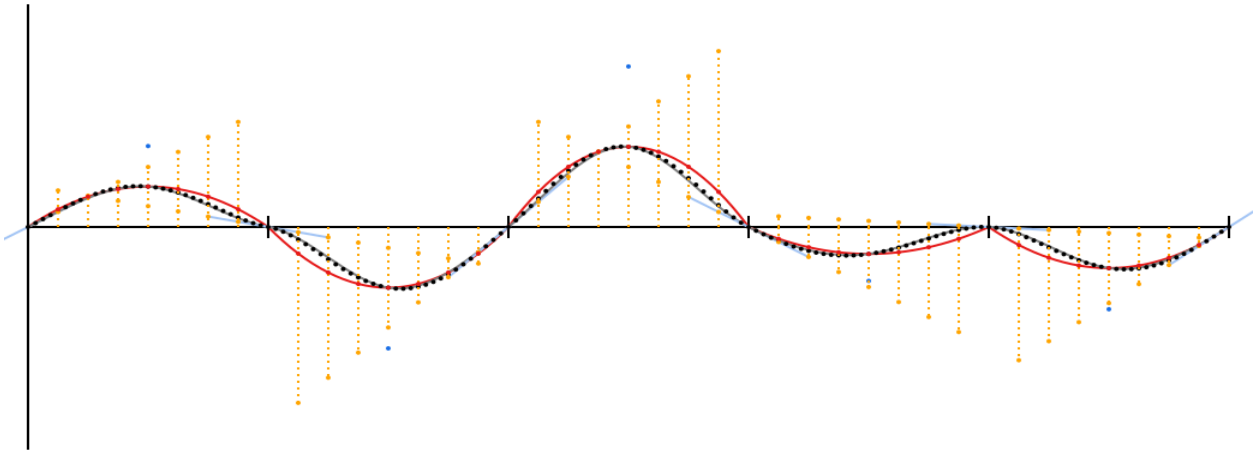

Рисунок 16. Пример более правильной интерполяции.

Уравнение прямой для каждого сегмента выглядит так:

$y = 2\,(a - b)x^4 - (3a - 5b)x^3 - 3bx^2 + ax.$

Разберемся с понятием октав. Создадим другую кривую Перлина. Например, такую:

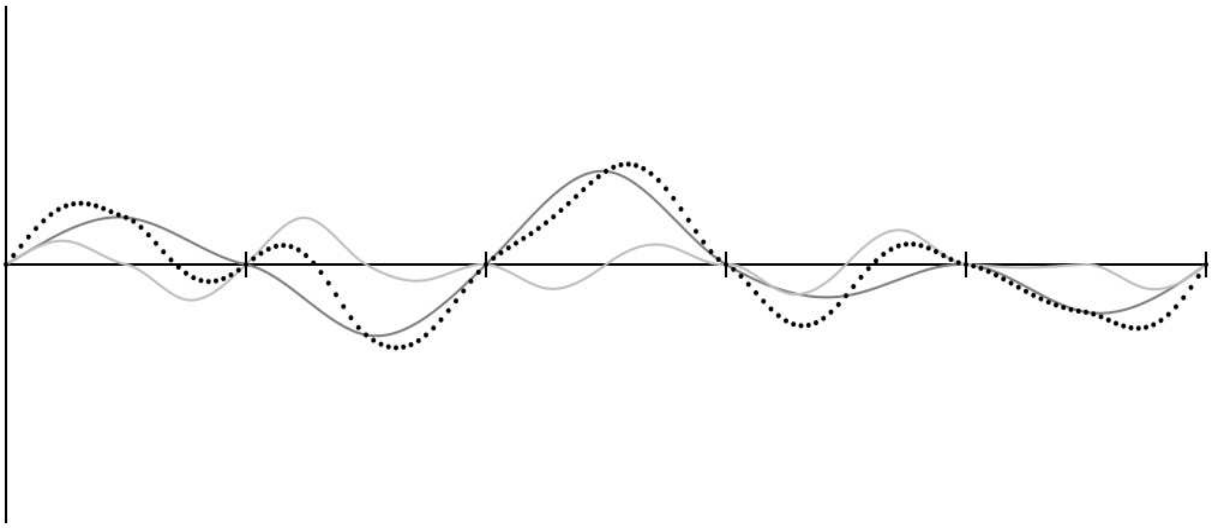

Рисунок 17. Пример кривой Перлина.

Если весь график уменьшить до половины размера и повторить дважды. То каждый отдельный график, то есть уменьшенная часть, называется октавой.

Пример кривой Перлина после применения 4-5 октав:

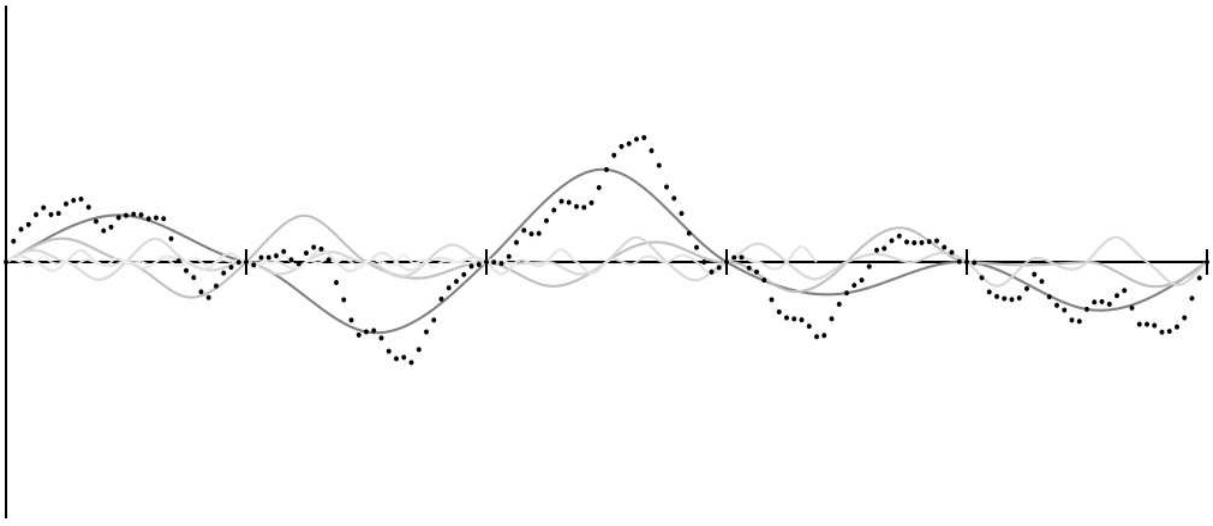

Рисунок 18. Пример применения 4-5 октав к кривой Перлина.

Рассмотрим в двумерном пространстве. Теперь, вместо числовой оси на входе имеется сетка. Так же, в двумерной плоскости для более точного понятия «наклона» появляется понятия вектора.

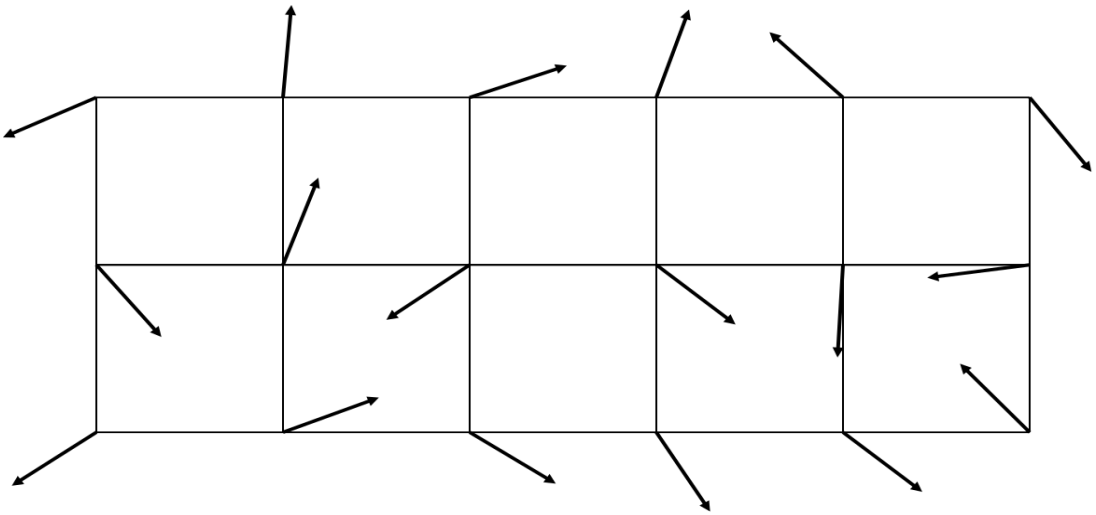

Рисунок 19. Пример сетки с векторами наклона.

В одномерном случае нужно сделать:

1. Найти расстояние до ближайшей точки слева и умножить его на наклон в этой точке;
2. Повторить это же действие для точки справа;
3. Интерполировать эти значения.

Для двумерного случая необходимы следующие изменения. Теперь, каждая точка имеет четыре соседних узла сетки, и требуется находить произведение наклона на расстояние.

Для этого можно использовать скалярное произведение. Если есть два вектора (a, b) и (c, d), тогда их скалярное произведение равняется сумме попарных произведений соответствующих компонент: $ac + bd$.

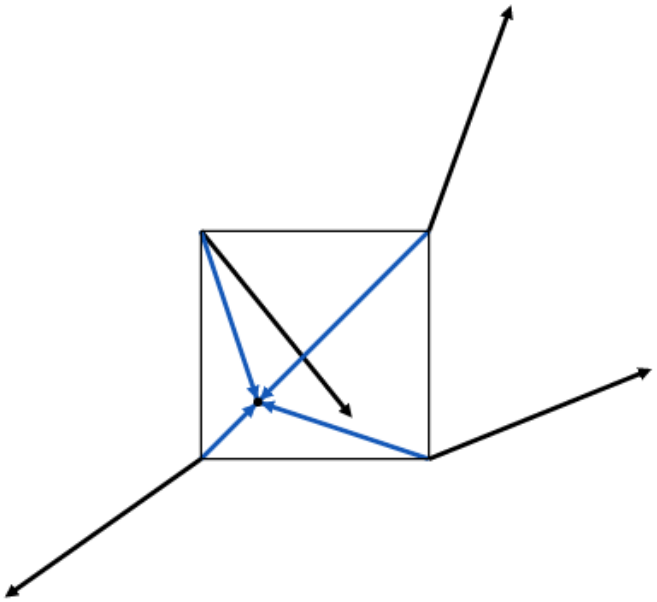

Рисунок 20. Пример узла сетки.

Единственное скалярное произведение между точкой и ближайшей к узлу сетки, выглядит следующим образом:

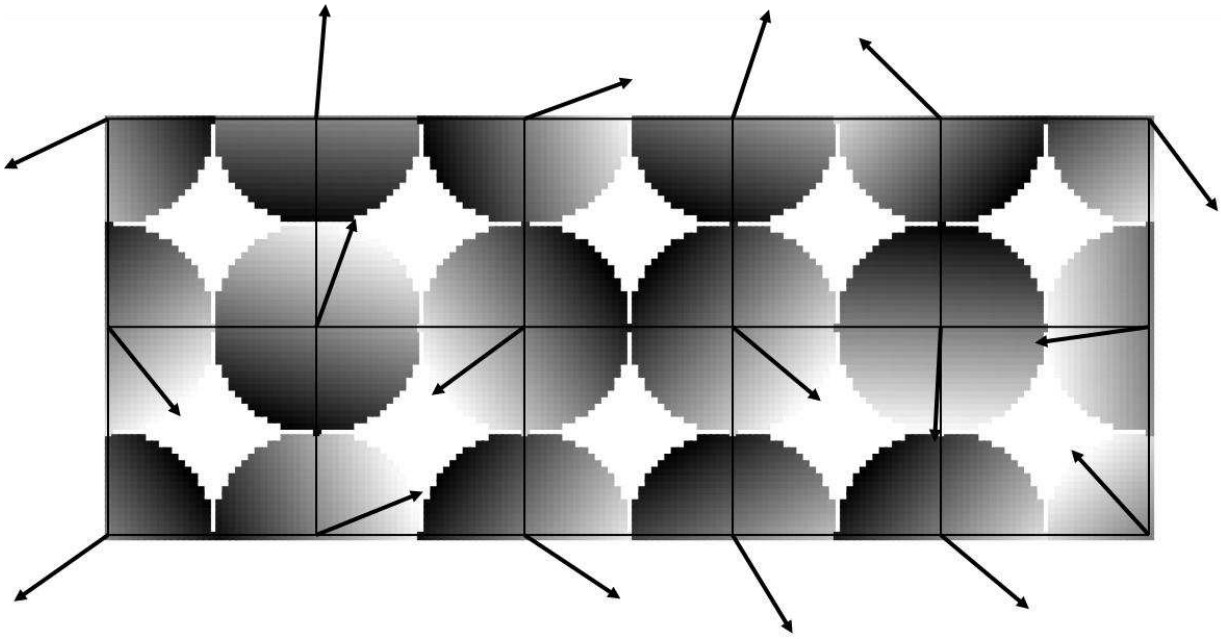

Рисунок 21. Пример единственного скалярного произведения между точкой и ближайшей к узлу сетки.

На основе данного метода можно построить более сложную сетку. Из этой сетки сформировать графическое изображение данных, а это и будет сгенерированная текстура шума Перлина.

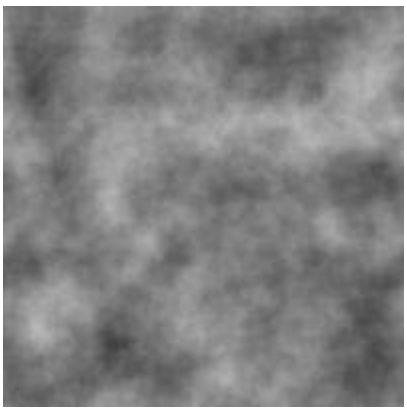

Рисунок 22. Пример текстуры шума Перлина.

С текстуры шума Перлина считывается карта высот, которая в свою очередь применяется к генерации ландшафта.

Для реализации данного проекта я выбрал алгоритм генерации шума Перлина, для создания карты высот, которую применим к ландшафту.

1.4 Программное обеспечение для реализации проекта

Программное обеспечение для реализации своего проекта я выбрал Unity.

Unity – это межплатформенный игровой движок с большими возможностями для реализации своих проектов. Этот движок дает возможность создавать свои проекты на большое количество платформ без особого труда. Основным плюсом данного движка является наличие визуальной среды разработки, межплатформенной поддержи и модульной системы компонентов. Также, данный движок является очень популярным как среди крупных создателей игр, так и среди малоизвестных.

Интерфейс Unity легко настраиваемый, каждый сможет сделать для себя удобное рабочее пространство. Отладку своих проектов можно делать непосредственно в редакторе, что тоже очень удобно. На движке можно писать на двух языках программирования, первый это C#, и JavaScript. Физические расчеты происходят за счет физического движка PhysX и NVIDIA.

Unity дает ряд преимуществ над остальными игровыми движками. Первое преимущество, это визуальная среда разработки. С ней гораздо легче

увидеть и настроить необходимые компоненты. Второе - кроссплатформенность, которая дает возможность сделать свой проект для разных устройств, не связанных между собой. Третье – модульность - очень удобно, что все необходимое расположено там, где ты этого ожидаешь, так как эти модули организованны просто и понятно. И четвертое - это относительная простота.

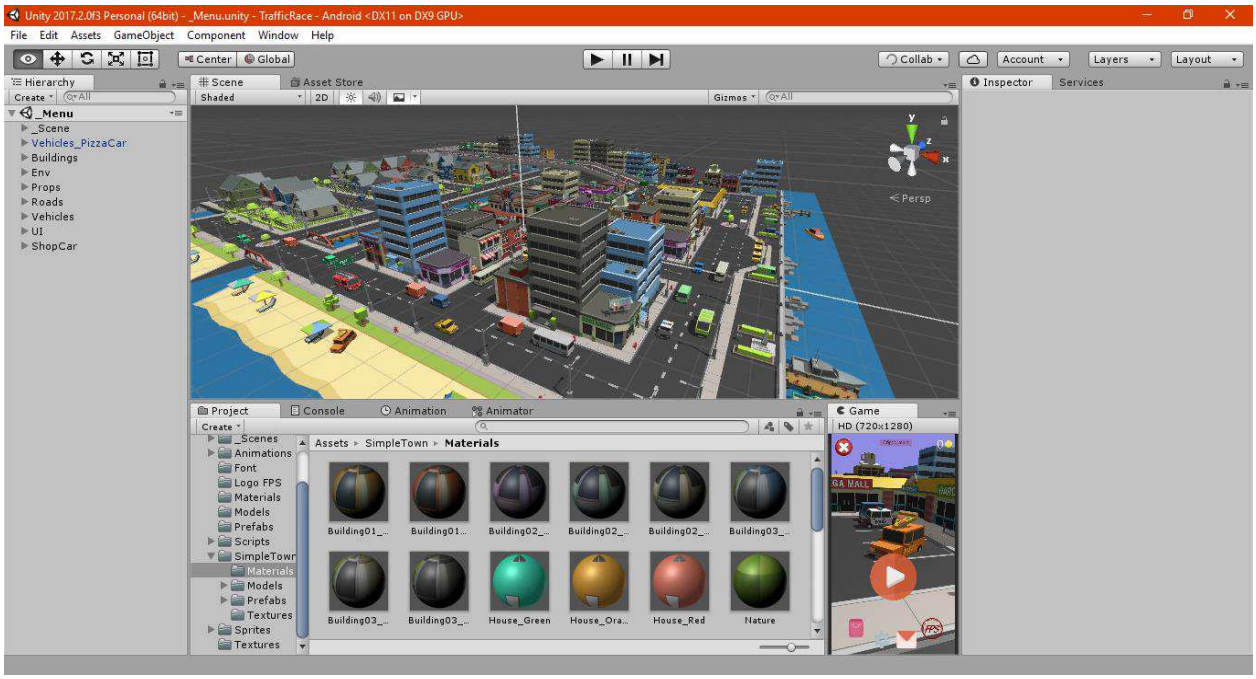

Рисунок 23. Пример рабочей области Unity.

Рисунок 24. Пример рабочей области Unity.

Список литературы

1. Jeff W. Murray C# Game Programming Cookbook for Unity 3D. - New-York: CRC Press, 2014. - 429 с.
2. Авдеева С.М., Куров А.В. Алгоритмы трехмерной машинной графики: Учебное пособие. - М.: Изд-во МГТУ им. Н.Э. Баумана, 1996. - 60 с.: ил.
3. Снук Г. Книга «Real-Time 3D Terrain Engines Using C++ and DirectX 9» – 2005.
4. Роджерс Д. Алгоритмические основы машинной графики: пер. с англ.— М.: Мир, 1989.— 512 с.: ил.
5. Официальный сайт Unity: https://unity.com/ru
6. Unity documentations: https://docs.unity3d.com/Manual/index.html

Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт математики и фундаментальной информатики
Базовая кафедра вычислительных и информационных технологий

УТВЕРЖДАЮ
Заведующей кафедрой
____/В.В. Шайдуров
«17» 06 2019г.

БАКАЛАВРСКАЯ РАБОТА

Направление 02.03.01 Математика и компьютерные науки

РАЗРАБОТКА АЛГОРИТМА ГЕНЕРАЦИИ ЛАНДШАФТА

Научный руководитель
кандидат физико-математических наук,
доцент ____/С.Н. Баранов
17.06.2019

Выпускник ____/В.Е. Репп
17.06.2019

Красноярск 2019