СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Наименование программы

Название программы: "Программа алгоритмической генерации ландшафта, готового для импорта в межплатформенную среду разработки компьютерных игр Unity" ("The Programm for Algorithmic Generation of the Ready to Import Landscape to Unity IDE") (далее "Генератор ландшафтов).

1.2. Документы, на основании которых ведется разработка

Основанием для разработки программы является приказ декана ФКН И.В. Аржанцева № 2.3-02/1012-0 2 от 10.12.18

2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

2.1. Назначение программы

Программа генерирует 3d ландшафт в формате .obj для последующего импорта в Unity, другие движки для разработки игр или 3d редакторы. Процесс генерации является процедурным и может быть использован в играх для быстрой генерации контента.

2.2. Область применения программы

Программа может быть применима в процессе разработки игр с 3d ландшафтом как заготовка для последующей детализации или как конечный ландшафт открытого мира.

3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

3.1. Постановка задачи на разработку программы

Целью курсовой работы "Программа алгоритмической генерации ландшафта, готового для импорта в межплатформенную среду разработки компьютерных игр Unity" является исследование алгоритмов процедурной генерации ландшафтов - карт местности и разработка программы с графическим интерфейсом, предоставляющей пользователю возможность влиять на процесс генерации ландшафта и после его создания использовать в Unitu. В дальнейшем предполагается усовершенствовать методы генерации и превратить программу в ландшафтный движок.

Программная генерация моделей используется когда необходимо за короткий промежуток времени создать псевдо-уникальный объект, а затраты ручного труда несоизмеримо больше чем затраты времени на генерацию при помощи программы. Процедурная генерация карт обладает рядом преимуществ над стандартными(ручными) методами создания карт - ландшафтов:

- Масштабируемость модели.
- Сравнительная скорость получения готового объекта.
- Маленький объем необходимой памяти для хранения самого алгоритма

Для генерации необходимо разработать методы и структуры, позволяющие относительно быстро (быстрее n^2) генерировать модель, навигировать по всей структуре ландшафта и редактировать отдельные его участки. Также такая структура должна давать возможность относительно естественным образом генерировать реки и просчитывать эрозию.

Также необходимо разработать интуитивно понятный интерфейс с возможностью задания некоторых первоначальных параметров генерации и сохранения результатов.

3.2. Описание алгоритма и функционирования программы

Для достижения целей работы был выбран метод построения диаграммы Вороного на плоскости для создания структуры необходимой гибкости(реберный список с двойными связями) и генерации полигонов, а также последующее применение алгоритма DiamondSquare для задания высоты этих полигонов и получении “интересного” распределения высот.

3.2.1. Построение диаграммы Вороного

Диаграммой Вороного называется такое разбиение плоскости с находящимися на ней точками(пусть множество этих точек - “V”) на многоугольники, что в каждом многоугольнике находится только одна точка - f из множества V, а все точки плоскости внутри этого многоугольника находятся к этой точке f ближе чем ко всем остальным точкам из множества V (рис. 1).

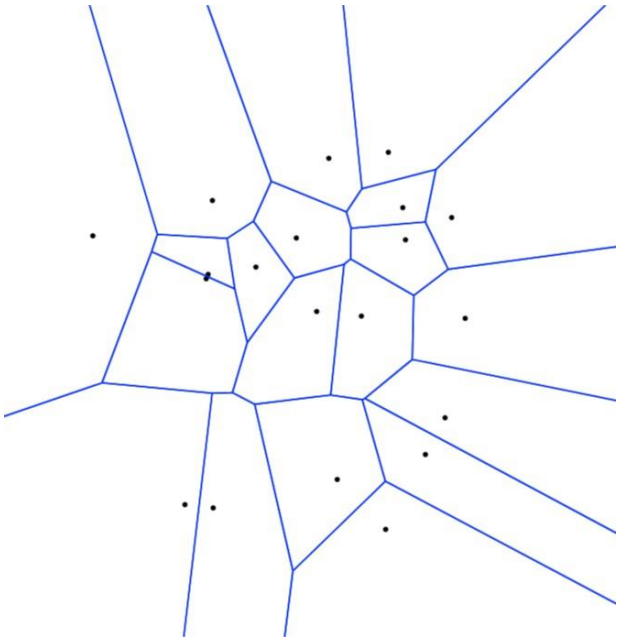


Рис. 1. Пример диаграммы Вороного на плоскости.

Вершины этих многоугольников называют вершинами диаграммы Вороного, грани многоугольников – ребра, а сами многоугольники – “сайты”. Из описания очевидно, что точки, принадлежащие ребрам равноудалены от двух точек из множества V, а вершины имеют кратчайшее расстояние до 3-х и более таких точек множества V одновременно. Для построения диаграммы был выбран алгоритм Стивена Форчуна 1986 года.

3.2.1.1. Термины алгоритма

Алгоритм основан на использовании метода заметающей прямой. В алгоритме Форчуна заметающая прямая - некоторая прямая, двигающаяся “сверху вниз” по плоскости и останавливающаяся на каждой точке события. События бывают двух типов: новая точка – (точка из множества V) и новая окружность, но подробнее события будут описаны ниже. На

каждой итерации строится парабола для каждой точки, находящейся над заметающей прямой, директрисой которой является заметающая прямая, а фокусом сама точка. Для каждой точки такая парабола – геометрическое место точек. Нижняя огибающая всех парабол над заметающей прямой называется береговой линией (рис. 2.). В процессе заметания эта береговая линия будет строить структуру диаграммы Вороного. Точки, лежащие на двух параболах одновременно будем называть узловыми точками -“break point”. Следует заметить, что эти точки в процессе заметания “рисуют” ребра Вороного и при пересечении ребер (столкновении двух узловых точек) образуется вершина Вороного.

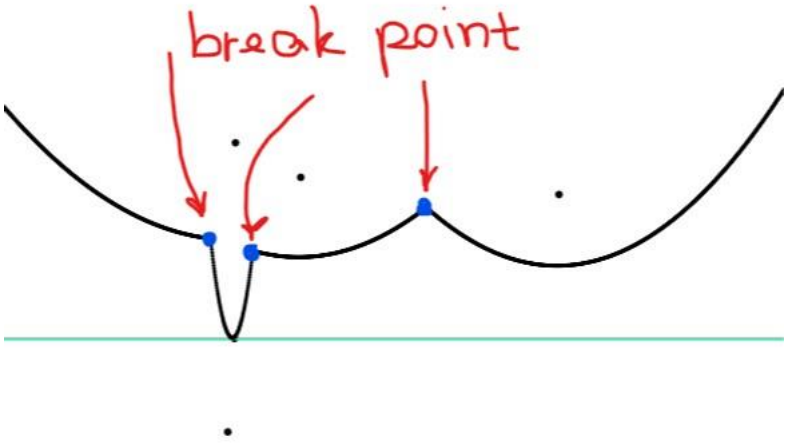


Рис. 2. Пример береговой линии и узловых точек - break point-ов.

При перемещении заметающей прямой к новой точке из множества V строится очередная парабола и добавляется к береговой линии. Новая парабола разбивает(рис. 3.) одну из парабол береговой линии, находящихся выше береговой линии. При этом необходимо проверить появление нового события вида “событие окружности” для новых дуг.

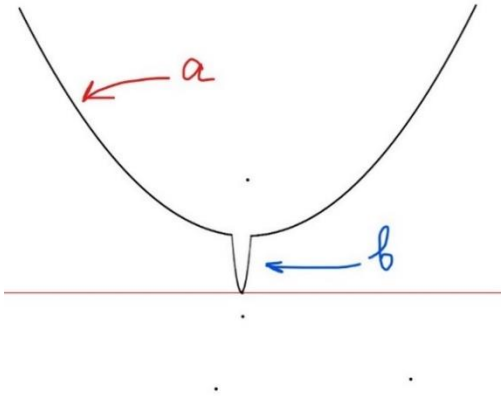


Рис. 3. Дуга "b” разбивает дугу “a” на две новых дуги при обработке события “событие точки”.

Событие окружности – момент, когда дуга из береговой линии “стягивается” в точку и находится на одинаковом расстоянии от 3-х и более точек множества V => несколько ребер Вороного соединяются в одной точке. (Рис. 4). В результате обработки этого события мы получаем вершину вороного в центре этой окружности. Появление таких событий необходимо проверять при обработке события точки и самого события окружности.

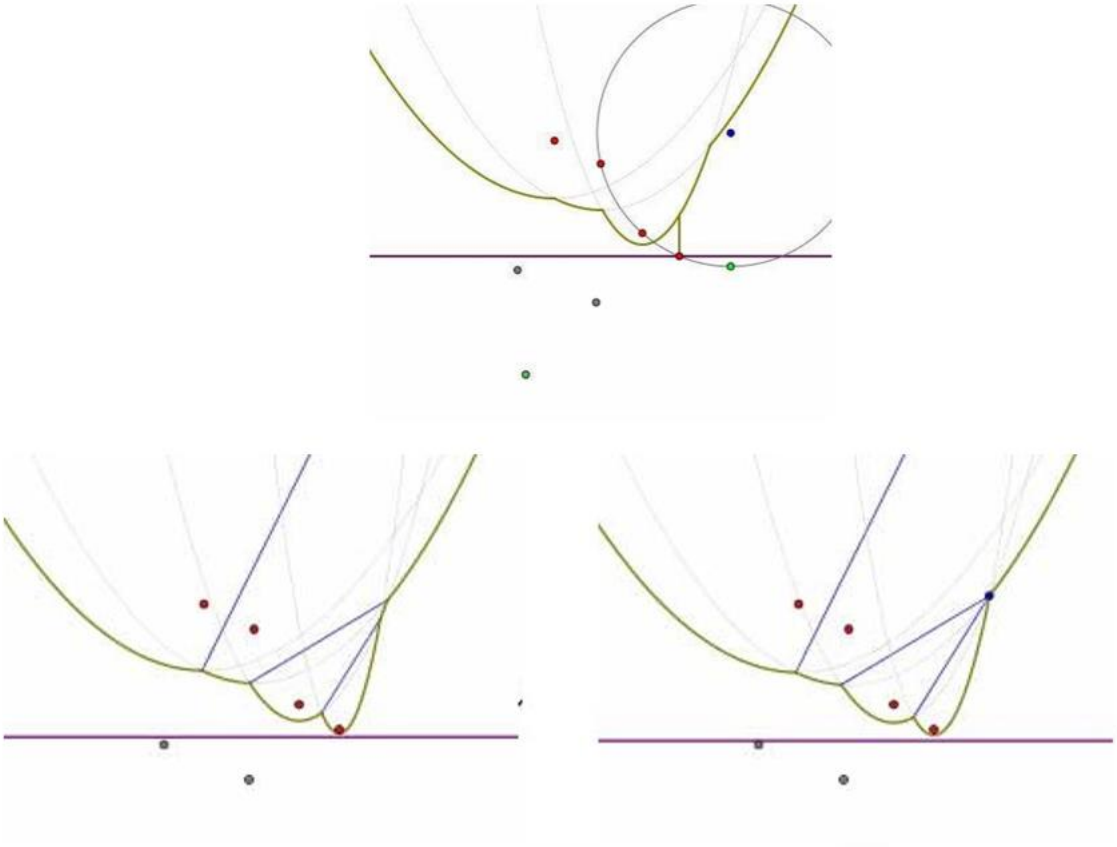


Рис. 4. Событие окружности.

3.2.1.2. Алгоритм Форчуна

Для построения диаграммы необходимо последовательно обрабатывать события, строить береговую линию, хранить порядок узловых точек и следить за исчезновением дуг из береговой линии. При обработке событий мы имеем два случая. Если это событие точки, то мы просто добавляем новую дугу в нашу береговую линию, обновляем порядок узловых точек и следим за появлением новых событий окружности. Если же это событие окружности, то мы фиксируем появление новой вершины в диаграмме Вороного и удаляем стянутую дугу из береговой линии.

Проблемы, которые возникают в процессе реализации алгоритма – это быстрый поиск дуги “b”, которую разбивает новая дуга “a” при вставке в береговую линию и быстрый способ удалить стянутую дугу. Все эти проблемы решаются хранением береговой дуги в виде самобалансирующегося двоичного дерева поиска. В данной курсовой работе было реализовано красно-черное дерево с узлами в виде дуг береговой линии. Следовательно манипуляции над

береговой линией будут требовать O(log(n)) операция и т.к. мы имеем линейное кол-во точек событий, то весь алгоритм будет работать за O(n*log(n)).

3.2.2. Алгоритм Ллойда

После построения диаграммы Вороного в работе применяется алгоритм Ллойда для достижения более равномерного распределения точек множества V по плоскости и получения более “красивой” формы ячеек Вороного. Алгоритм Ллойда заключается в вычислении “центроидов” для каждой ячейки Вороного и построения по множеству этих “центроидов” новой диаграммы Вороного.

Для вычисления “центроида” ячейки или “центра тяжести” полигона в нашем случае рассматривается сумма координат всех точек, принадлежащих текущей ячейке и делится на кол-во этих вершин. Такая операция проводится для каждого измерения координат центроида – x, y в нашем случае.

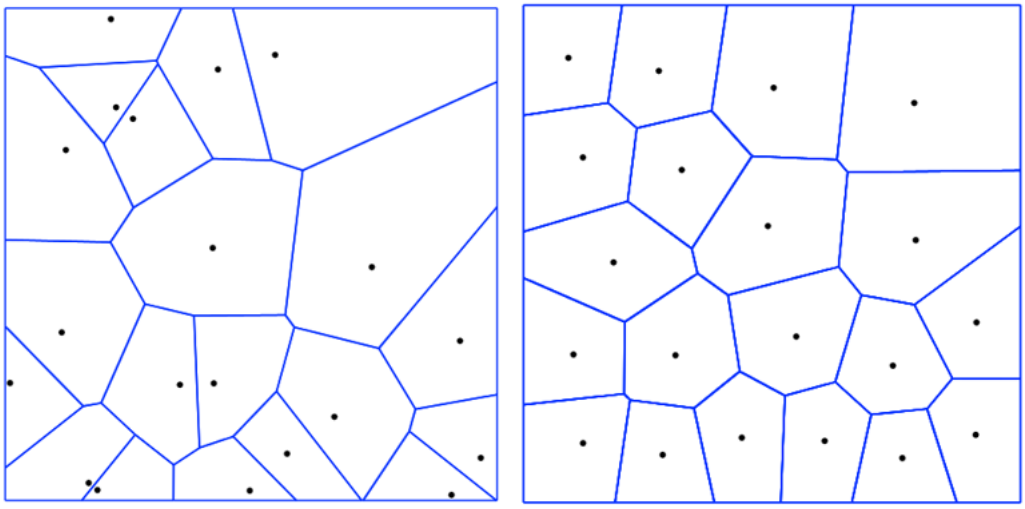


Рис. 5. Диаграмма до релаксации и после.

3.2.3. Алгоритм DiamondSquare

После генерации диаграммы Вороного и проведения на ней релаксации Ллойда в работе применяется алгоритм DiamondSquare для придания 3-х мерности имеющейся ldevthyjq карте

полигонов. В алгоритме рассматриваются только вершины Вороного и не затрагиваются основания сайтов – точки множества V.

В алгоритме DiamondSquare рекурсивно рассматриваются квадратные участки карты. В начале алгоритм получает на вход всю карту имеющую квадратную форму со стороной равной 2^n. После этого алгоритм вызывается для каждого из 4 квадратов на которые можно разделит первоначальный квадрат.

На каждом этапе обработки квадратного участка вычисляется точка в центре квадрата. Ее высота равна среднему арифметическому высот в углах этого квадрата. После этого вычисляются высоты точек на серединах сторон квадрата = среднему арифметическому точек находящихся слева, справа, сверху и снизу текущей точки на расстоянии = половине стороны квадрата(рис. 6.).

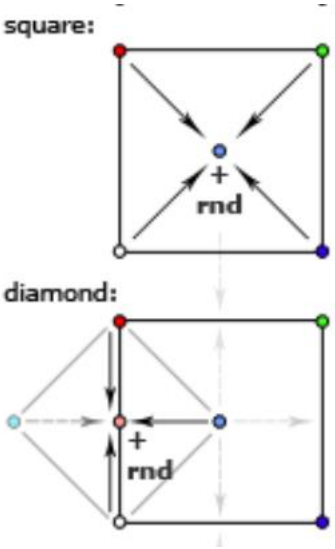


Рис.6. DiamondSquare

Таким образом формируется двумерный массив высот и при необходимости задать вершине Вороного высот мы обращаемся к элементу этого массива, имеющего индексы – координаты текущей вершины Вороного, округленные до ближайшего целого числа и умноженные на коэффициент отношения сторон карты и используемого массива, если используется массив большей размерности для большей детализации конечной карты.

3.3. Алгоритмы функционирования программы и ее интерфейс

Программа имеет удобный графический интерфейс (рис. 4), который позволяет:

1) Создать новую диаграмму.
2) Указать размеры ландшафта.
3) Указать степень детализации ландшафта.
4) Сохранить полученный ландшафт в формате .obj и текстуру .png.

Структура классов Программы приведена в Приложении 1, а описание методов приведено в Приложении 2.

Для создания новой диаграммы необходимо нажать кнопку “Generate” предварительно изменив стандартные параметры если это требуется. После создания в окне отобразится текстурированный “вид сверху” полученного ландшафта. Для его сохранения необходимо нажать кнопку “Save” и выбрать путь для сохранения файла.

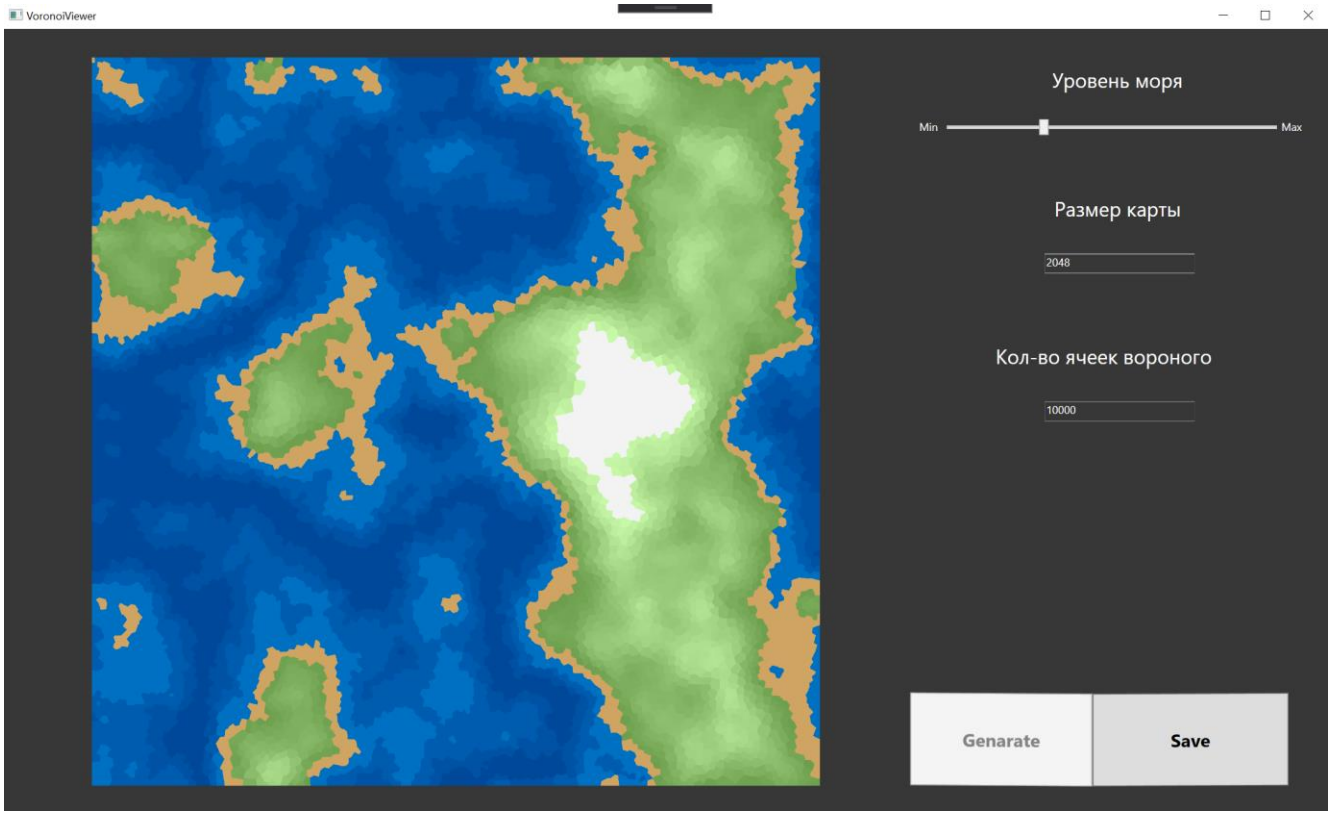


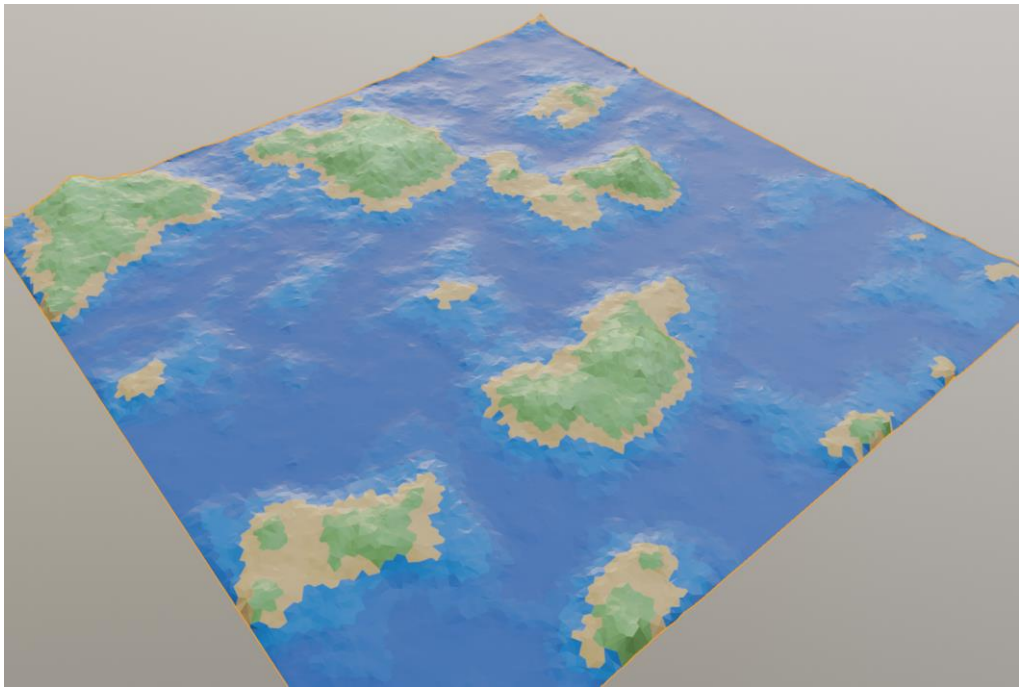
Рис. 7. Интерфейс программы.

3.4. Организация входных и выходных данных

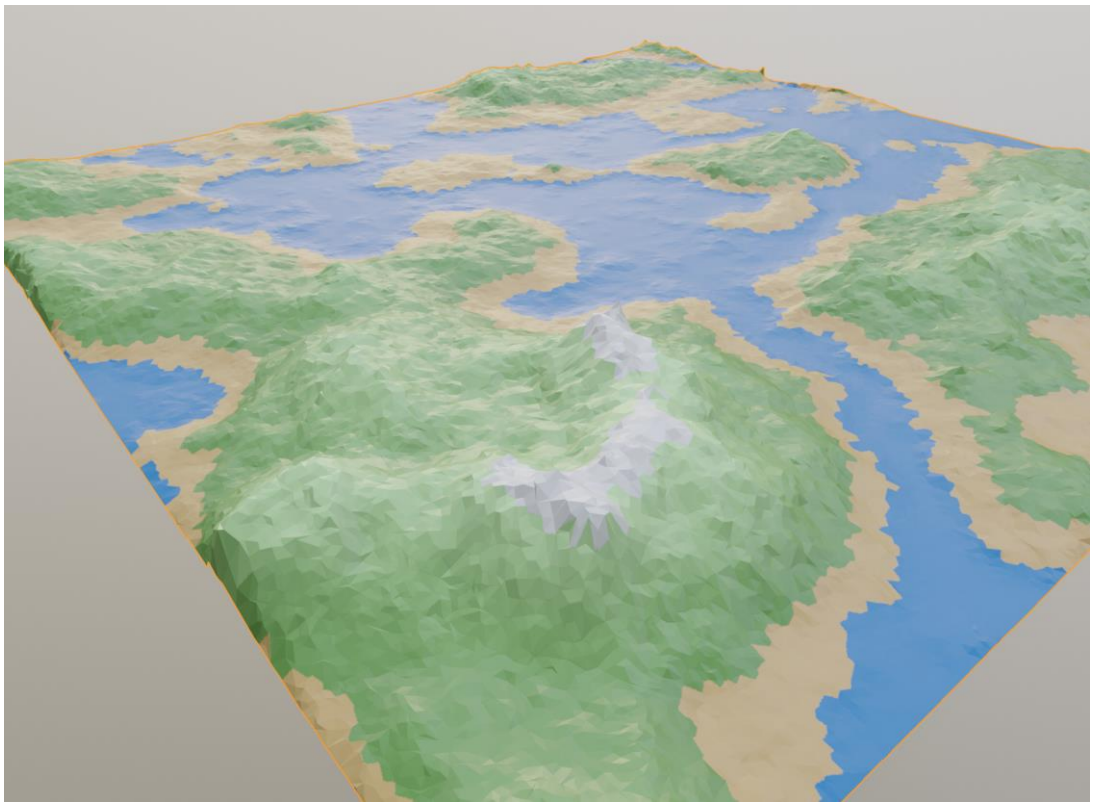
После распределения высот необходимо получить модель. Для импорта в unity был выбран формат 3d моделей .obj (рис.7.) т.к. имеет наиболее простую структуру и поддерживается большинством современных редакторов различного уровня.

В процессе формирования модели мы проходим по всем сайтам Вороного и записываем в файл все вершины Вороного, принадлежащие текущему сайту. После того как записаны все вершины записывается основание сайта – первоначальная точка из множества V имеющая высоту равную среднему арифметическому высот всех вершин Вороного, принадлежащих этому сайту. Для формирования треугольного полигона мы соединяем каждые две вершины, принадлежащие одному ребру Вороного с основанием текущего сайта. При это ребро также принадлежит рассматриваемому сайту.

Текстура формируется параллельно формированию 3d модели и представляет собой .png изображение, каждый бит которого имеет цвет, соответствующий среднему арифметическому высот трех точек диаграммы Вороного, образующих треугольный полигон. При записи в файл .obj для каждого полигона указываются координаты точки текстуры, которые должны соответствовать какой-то вершина полигона.



А.



Б.

Рис. 8. А, Б - Готовые модели с наложенной на них текстурой

```
# Blender v2.82 (sub 7) OBJ File: ''
# www.blender.org
mtllib untitled.mtl
o Cube
v 1.000000 1.000000 -1.000000
v 1.000000 -1.000000 -1.000000
v 1.000000 1.000000 1.000000
v 1.000000 -1.000000 1.000000
v -1.000000 1.000000 -1.000000
v -1.000000 -1.000000 -1.000000
v -1.000000 1.000000 1.000000
v -1.000000 -1.000000 1.000000
vt 0.625000 0.500000
vt 0.875000 0.500000
vt 0.875000 0.750000
vt 0.625000 0.750000
vt 0.375000 0.750000
vt 0.625000 1.000000
vt 0.375000 1.000000
vt 0.375000 0.000000
vt 0.625000 0.000000
vt 0.625000 0.250000
vt 0.375000 0.250000
vt 0.125000 0.500000
vt 0.375000 0.500000
vt 0.125000 0.750000
vn 0.0000 1.0000 0.0000
vn 0.0000 0.0000 1.0000
vn -1.0000 0.0000 0.0000
vn 0.0000 -1.0000 0.0000
vn 1.0000 0.0000 0.0000
vn 0.0000 0.0000 -1.0000
usemtl Material
s off
f 1/1/1 5/2/1 7/3/1 3/4/1
f 4/5/2 3/4/2 7/6/2 8/7/2
f 8/8/3 7/9/3 5/10/3 6/11/3
f 6/12/4 2/13/4 4/5/4 8/14/4
f 2/13/5 1/1/5 3/4/5 4/5/5
f 6/11/6 5/10/6 1/1/6 2/13/6
```

Рис. 9. Пример содержания фала в формате .obj.

3.5. Выбор технических и программных средств

- Операционная система Windows 10;
- Процессор архитектуры x64;
- .Net Framework 4.7.2.;
- DirectX 9 и позже.

4. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Процедурная генерация контента имеет множество применений. Данная программа может быть использована в небольших игровых проектах чтобы ускорить разработку игр или процесс прототипирования.

Сгенерированный программой ландшафт можно использовать для фона в процессе 3d моделирования или рендеринга.

Программа имеет большой потенциал для улучшения и доработки до ландшафтного движка.

5. ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ

1. ГОСТ 19.101-77 Виды программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

2. ГОСТ 19.102-77 Стадии разработки. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

3. ГОСТ 19.103-77 Обозначения программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

4. ГОСТ 19.104-78 Основные надписи. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

5. ГОСТ 19.105-78 Общие требования к программным документам. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом.// Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

8. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

9. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001

10. Статья «Doubly connected edge list» Wikipedia.org https://en.wikipedia.org/wiki/Doubly_connected_edge_list

11. Статья “Fortune's algorithm”Wikipedia.org https://en.wikipedia.org/wiki/Fortune%27s_algorithm

12. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein.

13. Статья “Красно-черное дерево” neerc.ifmo https://neerc.ifmo.ru/wiki/index.php?title=%D0%9A%D1%80%D0%B0%D1%81%D0%BD%D0%B

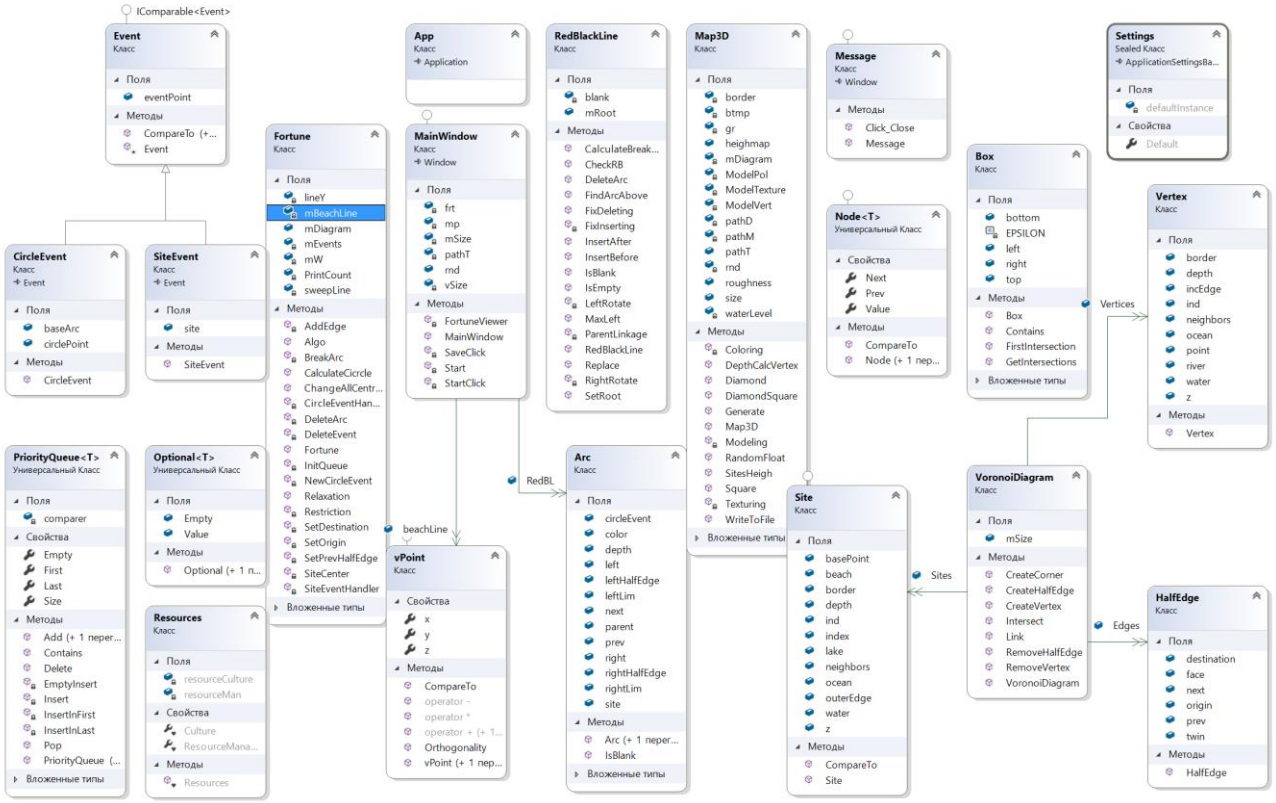
E-
%D1%87%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5
%D0%B2%D0%BE

14. Документация Microsoft C# https://docs.microsoft.com/ru-ru/dotnet/csharp/

15. Computational Geometry: Algorithms and Applications Third Edition (March 2008) Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars (7-я глава в переводе).

6. ПРИЛОЖЕНИЕ 1

Таблицы с описанием классов и интерфейсов



Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.1-01 81 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

7. ПРИЛОЖЕНИЕ 2

Таблицы с описанием методов

Таблица 1. Описание методов класса Arc

имя	модификаторы доступа	тип	аргументы	назначение
IsBlank()	public	void	-	Проверяет, является ли дуга пустым листом

Таблица 2. Описание методов класса Box

имя	модификаторы доступа	тип	аргументы	назначение
FirstIntersection	public	Intersection	vPoint origin, vPoint direction	Находит первое пересечение объекта Box и отрезка
GetIntersections	public	int	vPoint origin, vPoint destination, Intersection[] intersections	Находит пересечение отрезка с объектом Box

Таблица 3. Описание методов класса Fortune

имя	модификаторы доступа	тип	аргументы	назначение
Algo()	public	void	-	Запускает алгоритм Форчуна
SiteEventHandler	public	void	SiteEvent curEvent	Обрабатывает событие точки
CircleEventHandler	public	void	CircleEvent curCircleEvent	Обрабатывает событие окружности
BreakArc	public	Arc	Arc brokenArc, Site curSite	Разбивает дугу новой дугой на две и вставляет эту тройку в береговую линию

NewCircleEvent	public	void	Arc leftArc, Arc midleArc, Arc rightArc	Проверяет существование события окружности для 3-х дуг
DeleteArc	public	void	Arc curArc, ref Vertex newVert	Удаляет стянутую дугу
SetDestination	private	void	Arc left, Arc right, Vertex vertex	Устанавливает вершину в конец отрезка
SetOrigin	private	void	Arc left, Arc right, Vertex vertex	Устанавливает вершину в начало отрезка
SetPrevHalfEdge	public	void	HalfEdge prev, HalfEdge next	Связывает два подряд идущих ребра как предыдущее и следующее
CalculateCicrcle	public	vPoint	vPoint fPoint, vPoint sPoint, vPoint tPoint, ref vPoint eventPoint	Вычисляет точку окружности по 3-м другим точкам
AddEdge	private	void	Arc leftArc, Arc rightArc	Добавляет ребра между двумя дугами, которые они рисуют в процессе работы алгоритма.
DeleteEvent	private	void	ref Arc curArc	Удаляет событие окружности связанное с дугой curArc.
Restriction	private	void	Box box	Ограничивает

				диграмму объектом Box
Relaxation	public	void	-	Проводит релаксацию Ллойда
ChangeAllCentres	public	VoronoiDiagram	-	Пересчитывает все точки на центроиды
SiteCenter	private	vPoint	Site site	Вычисление центроида
InitQueue	private	void	-	Инициализирует очередь событий

Таблица 3. Описание методов класса MainWindow

имя	модификаторы доступа	тип	аргументы	назначение
StartClick	private	void	object sender, RoutedEventArgs e	Обработчик кнопки Generate
Start	private	void	-	Инициирует начало работы алгоритмаов
SaveClick	private	void	object sender, RoutedEventArgs e	Сохранение модели в файл
FortuneViewer	private	void	-	Генерирует первоначальные точки для построения лиграммы

Таблица 4. Описание методов класса Map3D

имя	модификаторы доступа	тип	аргументы	назначение

Modeling	private	void	-	Создает список точек для последующей записи в файл
Coloring	private	void	-	Задает цвет для определенной высоты
Texturing	private	void	vPoint[] verts	Создание текстуры
WriteToFile	public	void	-	Запись модели и текстуры в файл
DepthCalcVertex	public	Void	-	Обход всех вершин диаграммы и задание им высоты
SitesHeigh	public	void	bool perl	Задание высоты для каждого основания сайта
RandomFloat	private	float	float minimum, float maximum	Возвращает случайное число float в указанном диапазоне
Square	private	void	int lx, int ly, int rx, int ry	Шаг Square в алгоритме DiamodSquare
Diamond	private	void	int tgx, int tgy, int l	Шаг Diamond в алгоритме DiamondSquare
DiamondSquare	private	void	int lx, int ly, int rx, int ry	Запуск алгоритма DiamondSquare
Generate	public	void	int size	Подготовка к запуску алгоритма DiamondSquare и его запуск

Таблица 5. Описание методов класса PriorityQueue

имя	модификаторы	тип	аргументы	назначение

	доступа			
Add	public	void	T item	Добавляет новый элемент в очередь с приоритетом
Pop	public	void	-	Возвращает первый элемент очереди и удаляет его из самой очереди
Delete	public	void	Node<T> deleted	Удаляет элемент с заданным ключом из очереди
InsertInFirst	private	void	Node<T> ins	Вставляет новый элемент в начало очереди
InsertInLast	private	void	Node<T> ins	Вставляет новый элемент в конец очереди
Insert	private	void	Node<T> cur, Node<T> ins	Вставляет новый элемент между двумя элементами из очереди
EmptyInsert	private	void	Node<T> ins	Вставляет новый элемент в пустую очередь
Contains	private	bool	T item	Проверяет наличие элемента с заданным ключом в очереди

Таблица 6. Описание методов класса RedBlackLine

имя	модификаторы доступа	тип	аргументы	назначение
IsBlank	public	bool	Arc curArc	Проверяет дугу на пустоту
SetRoot	public	void	Arc curArc	Устанавливает корень дерева
IsEmpty	public	void	-	Проверяет береговую линию на пустоту
FindArcAbove	private	void	vPoint curPoint, double line	Находит дугу над точкой
Replace	pulic	void	Arc x, Arc y	Замена X->Y в береговой линии.
ParentLinkage	private	void	Arc x, Arc y	Связывает родителя дуги X

				с дугой Y, а дугу Y с родителем X.
MaxLeft	public	Arc	Arc x	Возвращает самую левую дугу
InsertBefore	public	void	Arc x, Arc y	Вставляет новую дугу в береговую линию перед x
InsertAfter	public	void	Arc x, Arc y)	Вставляет новую дугу в береговую линию после x
DeleteArc	public	void	Arc x	Удаляет дугу
FixInserting	private	void	ref Arc curArc	Восстановление свойств красно-черного дерева после вставки
LeftRotate	private	void	Arc x	Поворот влево вокруг узла x
RightRotate	private	void	Arc x	Поворот вправо вокруг узла x
FixDeleting	private	void	ref Arc curArc	Восстановление свойств красно-черного дерева после удаления
CalculateBreakPoint	private	double	vPoint leftPoint, vPoint rightPoint, double line	Вычисление точки пересечения двух парабол

Таблица 7. Описание методов класса VoronoiDiagram

имя	модификаторы доступа	тип	аргументы	назначение
CreateVertex	public	Vertex	vPoint curPoint	Создает вершину диаграммы
CreateHalfEdge	public	HalfEdge	Site curSite	Создает новое полу-ребро
RemoveHalfEdge	public	int	HalfEdge halfEdge	Удаляет полу-ребро ребро
RemoveVertex	public	void	Vertex vertex	Удаляет вершину
CreateCorner	public	Vertex	Box box,	Создает угол диаграммы

			Box.Side side	
Link	public	void	Box box, HalfEdge start, Box.Side startSide, HalfEdge end, Box.Side endSide	Связывает ребра после пересения с Box
Intersect	public	void	Box box	Пересекает диаграмму с Box

8. ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

зм.	Номера листов (страниц)				Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа и дата	Подпись	ата
	измененных	замененных	новых	анулированных					

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.04.1-01 81 01-1				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата
