На тему Разработка программной среды для генерации ландшафта по карте биомов

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Цифровое изобразительное искусство имеет большой спрос: в киноиндустрии, рекламе, компьютерных играх, при создании чертежей, карт и так далее. Речь может идти о двухмерной (плоские изображения в растровом или векторном формате) или трехмерной графике (при которой строятся трехмерные модели различных реальных или вымышленных объектов). Моделируется все: начиная от пластиковых бутылок до целых кварталов городов.

Существует частная задача: моделирование рельефа и ландшафта. Построение природы в цифровом виде особенно актуально для проектов с открытым миром, там, как правило, требуется создавать большие природные пространства.

Ландшафт довольно утомительно создавать вручную. Например, можно рисовать карту высот (цифровое изображение, белый цвет на котором – самая высокая точка местности, черный – самая низкая) в графическом редакторе. Посредством мыши можно нанести в нужные области светлый или темный цвет. В целом, черно–белое изображение дает человеку примерное представление о рельефе местности. Однако существуют следующие трудности. Во–первых, карта высот сама по себе недостаточно наглядна. Это лишь высота, которая абстрагирована от всего остального. Было бы намного лучше, если человек будет видеть образ природы: то, что он создает.

Объектом выпускной квалификационной работы (ВКР) является процесс генерации цифрового ландшафта. Предметом исследования – автоматизация процесса по генерации цифрового ландшафта.

Целью выпускной квалификационной работы является проектирование и разработка программного обеспечения для генерации ландшафта с простым и удобным интерфейсом.

Для достижения данной цели необходимо выполнить следующие задачи:

– провести анализ предметной области и доступных аналогов программного обеспечения, эксплуатируемых в предметной области тематики выпускной квалификационной работы;

– разработать техническое задание на проектируемое и разрабатываемое программное обеспечение;

– осуществить проектирование разрабатываемого программного продукта;

– разработать программный продукт и его интерфейс.

В соответствии с намеченной целью и задачами, были определены следующие методы исследования:

– теоретический анализ систем–аналогов, распространенных алгоритмов генерации ландшафта;

– программные эксперименты: выставление различных входных параметров у алгоритмов, наблюдение и оценка результатов.

1. Аналитическая часть

1.1 Проблематика

Представим, что у нас есть среда, которая предоставляет наглядный интерфейс с возможностью удобного редактирования (то есть с картой высот мы непосредственно не работаем): подъема, понижения высоты, сглаживания. Даже в этом случае не пропадают следующие трудности: рисование ручьев, рек, линий берегов. Особенно, если требуется отразить большую площадь местности. Несложно отразить несколько простых речных изгибов, круглых озер. Однако, если мы нацеливаемся на объемы, реалистичность, возникают большие трудности. Человеку приходится имитировать случайность и при этом полагаться на интуицию. В одном месте – «скосить» береговую линию, в другом – «скруглить». При этом постоянно следить за тем, чтобы какие-то фрагменты ландшафта повторялись как можно меньше. Такое в природе «не принято». Впоследствии картина может выглядеть не очень хорошо. Приходится вносить еще какие-то изменения. Это может повторяться больше число раз. Впоследствии, это может отнять большое количество времени и сил.

Не только речные массивы могут являться проблемой для любителей реализма. Общие очертания гор, холмов вполне можно отразить без особых усилий. Однако обсудим детализацию. Поверхность подвергается выветриванию, эрозии. Свои особенности имеет и ландшафт каменной местности, пустынной и снежной. Данные явления привносят истинную красоту ландшафту, и большие проблемы человеку, которые собирается их рисовать вручную.

1.2 Обзор аналогов

Ввиду раннее описанных сложностей для моделирования ландшафта разрабатываются специальные программные продукты, которые служат решению именно этой проблемы. Следует отметить такие программные продукты как «World machine», «Terragen», «InstantTerra». Это большие, серьезные средства с объемным функционалом, посредством которого можно получить очень реалистичную природу. Однако существуют и проблемы. Это сложные в эксплуатации системы. Для получения приемлемой картины ландшафта потребуется приложить определенные усилия.

Генератор ландшафта «WorldMachine» представлен на рисунке 1.1. Данный аналог способен дать реалистичную картину природы, имеет объемный функционал. Есть возможность определять текстуры, генерировать речные массивы.

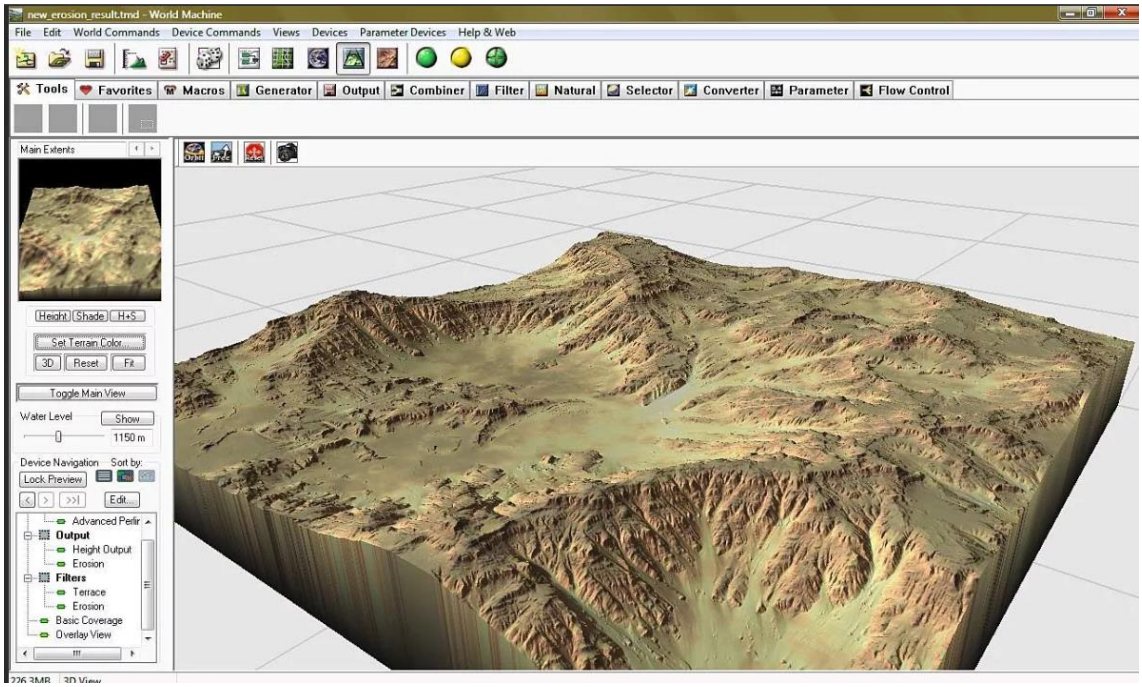

Рисунок 1.1 – генератор ланшдафта «WorldMachine»

Аналог «InstantTerra» представлен на рисунке 1.2. Он также дает возможность пользователю получить фотореалистичные изображения. Здесь

присутствует система функциональных блоков. Посредством них пользователь задает преобразования, определяет выходное изображение. Например, в виде блоков можно задать преобразование ландшафта: подсоединить блок генерации шума (первичной карты высот). Затем – выход последнего «присоединить» ко входу блоку генерации эрозии. Иными словами, так мы определяем поток данных и преобразований. Блоки в данном случае являются преобразователями. За счет их подстановок, перестановок таких блоков достигается гибкость функционала.

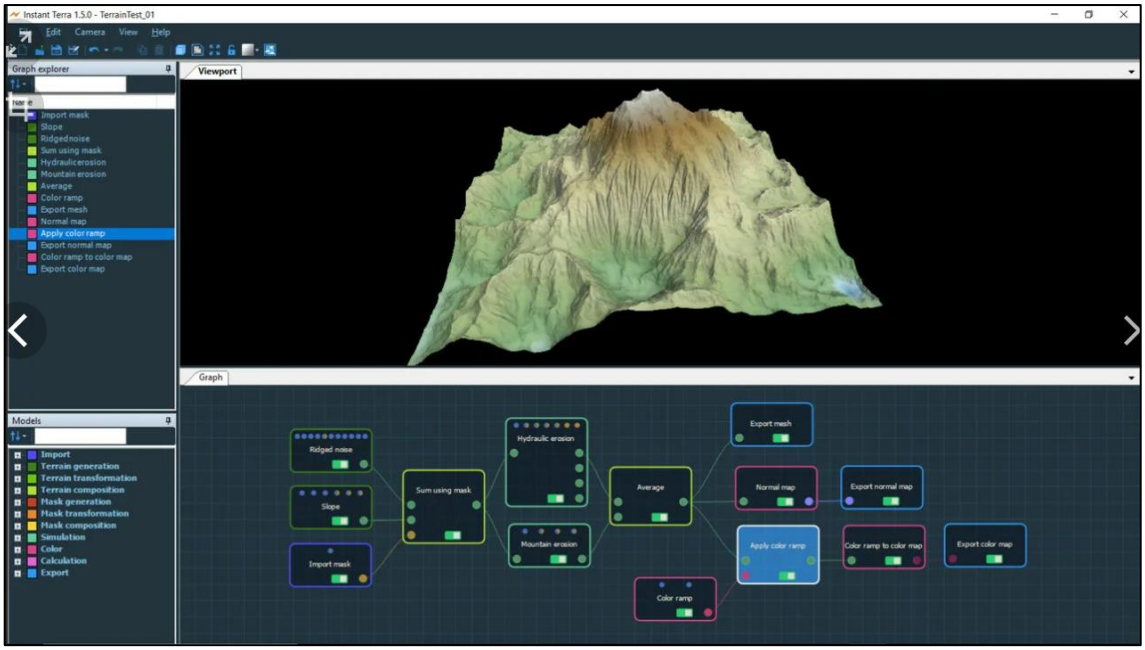

Рисунок 1.2 – генератор ланшдафта «InstantTerra»

Аналог «TerraGen» (рисунок 1.3), в целом, не уступает остальным. Здесь так же присутствует система блоков, как следствие: здесь присутствует обширный функционал.

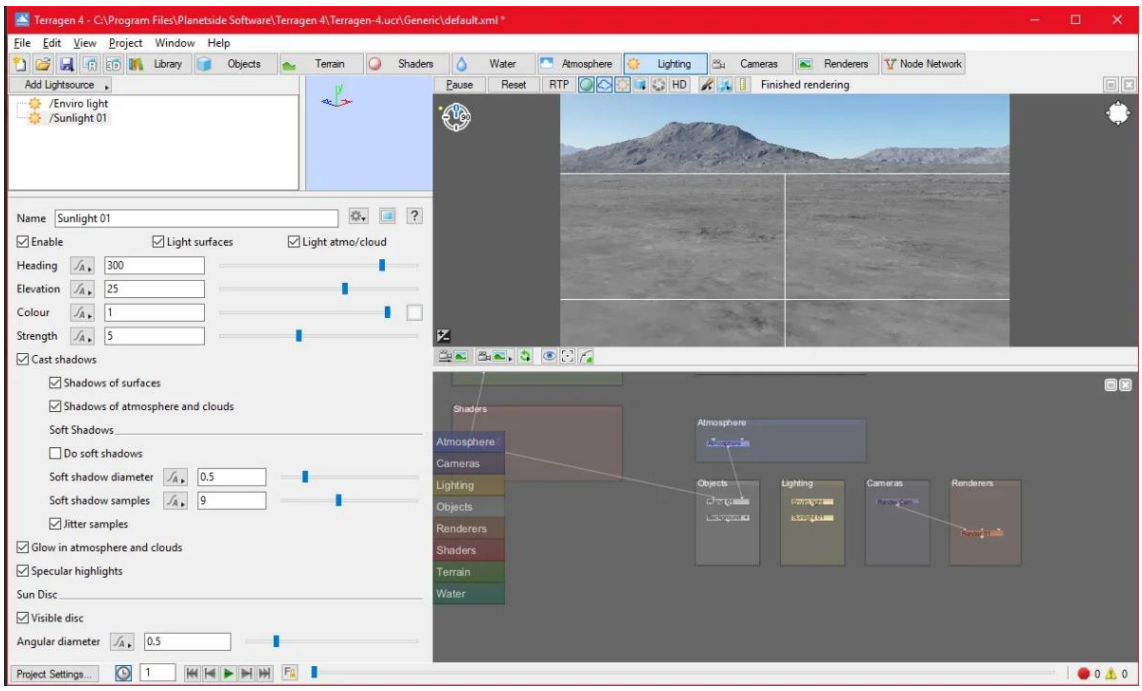

Рисунок 1.3 – генератор ланшдафта «Terragen»

В аналоге «MapMagic» (рисунок 1.4) так же представлена система блоков. Особенность данной системы заключается в том, что она нацелена на игровой движок «Unity». То есть, создаваемые карты очень удобно встраивать в системы, написанные на данном движке. Представленные раннее аналоги в этом плане не имеют нацеленность на определенный движок. Они в этом плане являются более универсальными. Функционал так же обширен, результат картинки реалистичен. Однако, по последнему параметру данная система немного отстает от раннее представленных.

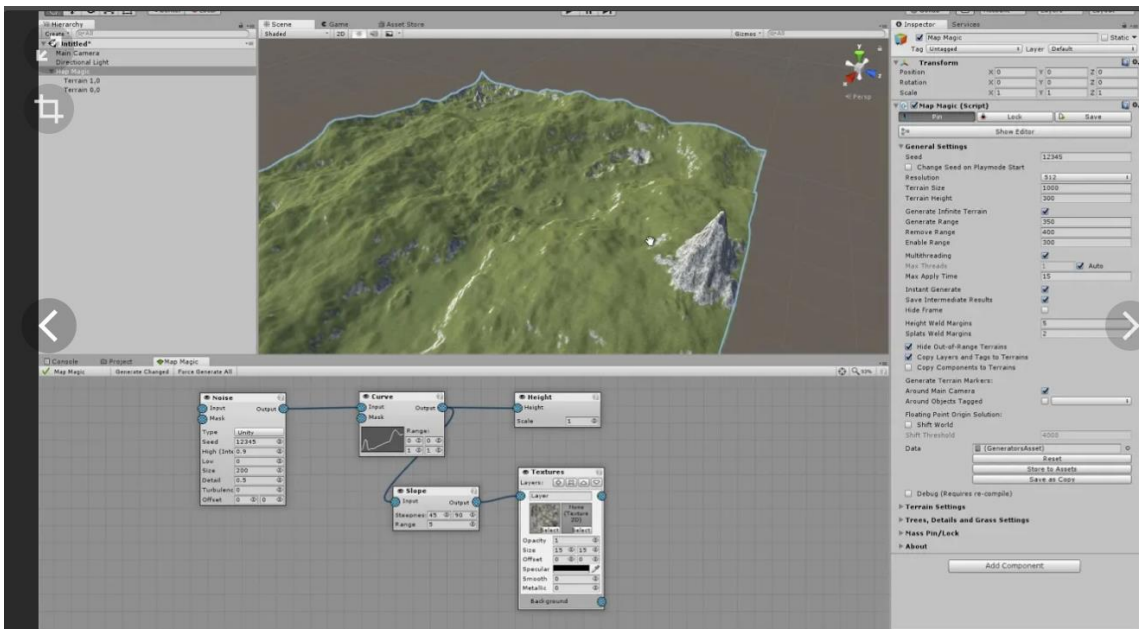

Рисунок 1.4 – генератор ланшдафта «MapMagic»

Несмотря на преимущество описанных систем, они не предоставляют простого способа решения следующих проблем. Например, в одном месте мы обязательно хотим видеть равнину. В другом: обязательно холмистый ландшафт. Ожидать того, что все сгенерируется именно таким образом хотя бы с двадцатой попытки – опрометчиво.

Система блоков, с одной стороны, очень практична. Но с другой, все же, не является достаточно простой в эксплуатации. Пользователю требуется предварительно освоить этот принцип, научиться пользоваться: это может быть довольно затруднительным процессом.

1.3 Выбор подхода

С целью упростить функционал будущей системы, было принято решение рассмотреть биомный подход. Речь идет об интерфейсе, где можно было бы определить области (далее будем называть их биомами во избежание ассоциаций с административно-территориальными единицами),

границы между ними (рисунок 1.5). И для каждой – назначить собственные параметры генерации.

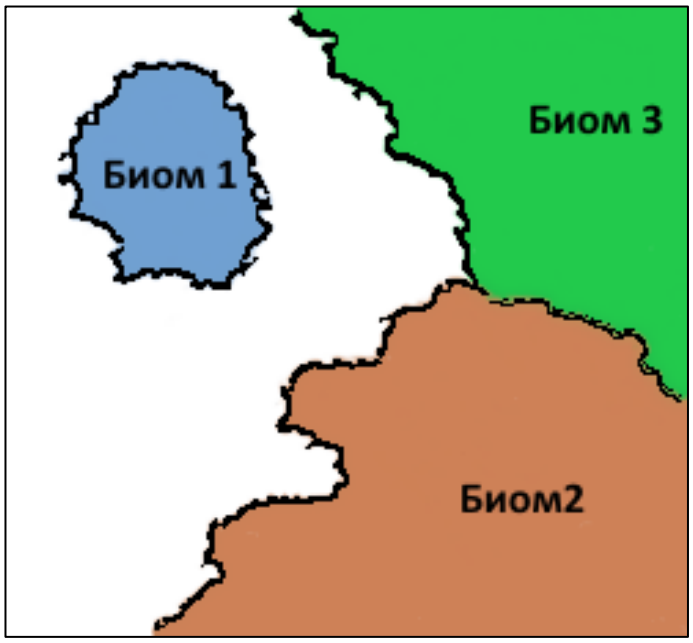

Рисунок 1.5 – Карта биомов

Данный подход имеет распространение. Можно рассмотреть статью [6]. В ней описывается генерация полигонов, на их основе которых строятся регионы, реки и так далее. Если говорить именно про биомный подход в генерации ландшафта, он также хорошо описан. Например, он приводится в статье [5]. В целом, его можно наблюдать в компьютерных играх.

Следует провести разницу между вышеописанными концепциями и нашей. В них не предполагается явное задание биомов: их количество, свойства не определяются напрямую пользователем. То есть, в них результат определяется фактором случайности. Трудно добиться того, что было задумано. Подход, описываемый в данной работе нацелен на уменьшение фактора случайности. При нем пользователь будет иметь четкое, ясное представление о том, что он получит в итоге.

1.4 Определение функционала

Было написано техническое задание на разработку программного обеспечения, в котором, помимо нефункциональных требований и графика работ, определяется функционал системы. Полный текст технического задания представлен в приложении А.

Были определены следующие задачи:

– реализовать функционал работы с сеткой шестиугольников;

– преобразовать вышеописанное в сетку многоугольников (т.е пространство, состоящее из многоугольников; пример представлен на рисунке 1.6);

– сделать границы сетки менее угловатыми, более реалистичными; сгенерировать шум (например, шум Перлина или Diamond–Square) для каждого биома со своими параметрами минимальной и максимальной высоты;

– сгладить все резкие перепады на карте (нет никакой гарантии, что после генерации шума с разными параметрами на их границах биомов не будет несостыковок;);

– применить ко всей карте алгоритм эрозии для повышения уровня детализации, придания реалистичности;

– реализовать функционал для просмотра и редактирования сетки шестиугольников в трехмерном пространстве; установки параметров высоты для каждого биома, запуска генерации и просмотра результата.

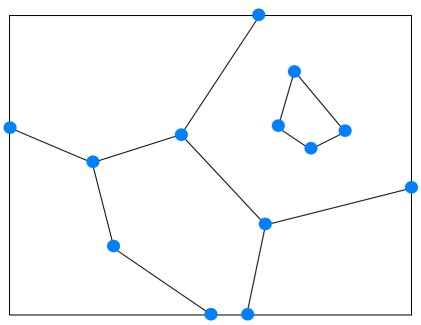

Рисунок 1.6 – Сетка многоугольников

2. Проектная часть

2.1 Диаграмма компонентов

До разработки трехмерного интерфейса редактирования сетки шестиугольников было принято решение разработать простое и небольшое ПО, которое позволит создавать файлы с данными сетки (компонент «HexGridEditor»). Данное решение разрешит проблему независимости разработки: на трехмерный интерфейс нужно больше времени, больше вероятность багов, прочих проблем. Имея простой двухмерный интерфейс можно быстро и удобно отработать, протестировать логическое поведение сетки, создать множество входных файлов для тестов.

Компонент геометрии («Geometry») будет содержать в себе функционал шестиугольной сетки, полигональной сетки, детализации отрезков и все прочее, тематически связанное с этим.

Компонент карты высот («Heightmap») будет хранить класс карты высот и те, которые нацелены на ее генерацию и преобразование (состыковка и эрозия).

Компонент «Graphics» содержит графический функционал трехмерной графики, модуль отображения ландшафта и шестиугольной сетки.

Были спроектированы компоненты и связи между ними (рисунок 2.1).

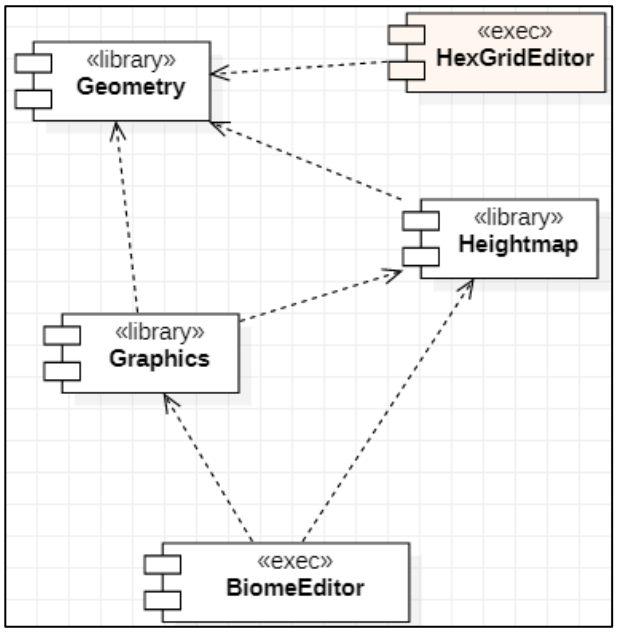

Рисунок 2.1 – Диаграмма компонентов

2.2 Компонент геометрии

Были спроектирован функционал, реализующий сетку шестиугольников. Классы представлены на рисунке 2.2. Полная диаграмма представлена на рисунке Б1. Был построен класс HexagonGrid. В наследнике RegionHexagonGrid добавляется функционал разметки пространства. За счет метода setRegion(index, regionId) мы указываем, что шестиугольник с данным индексом теперь будет относиться к региону с данным id.

У шестиугольника реализованы методы для получения его точек – верхнюю, нижнюю, верхнюю левую и т.д. Это все может оказаться полезным в случае, если мы будем проходить по точкам. У шестиугольной сетки созданы методы получения ширины, высоты, преобразования точек.

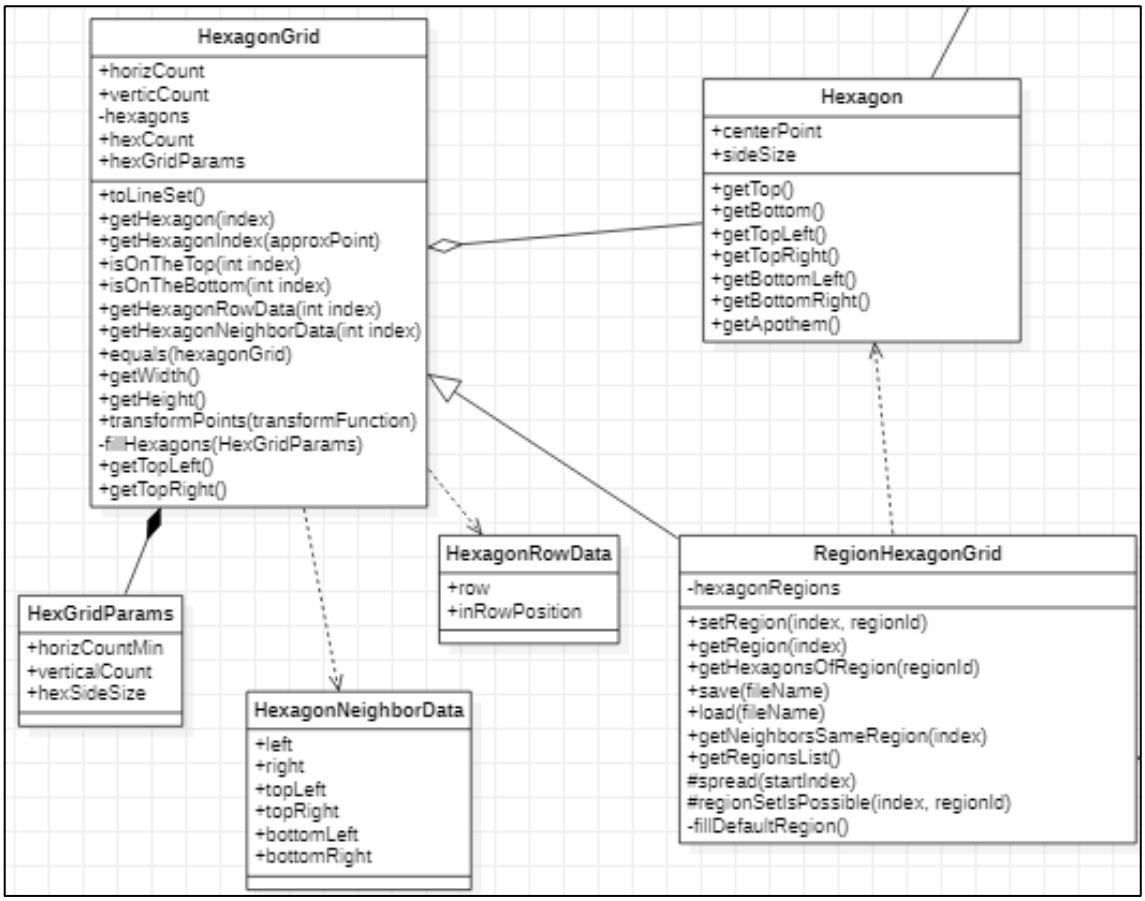

Рисунок 2.2 – Классы, относящиеся к сетке шестиугольников

Были спроектированы классы, относящиеся к сетке полигонов (рисунок 2.3). Функционал детализации («LinesDetalization») предназначен не напрямую для сетки полигонов («PolygonWithHolesGrid»), а для набора отрезков. Данное решение делает этот класс более универсальным: можно детализировать не только сетку, но и любой произвольный набор отрезков. Получается, чтобы продетализировать сетку, необходимо получить ее набор отрезков. Это было реализовано посредством метода «getLineSegments».

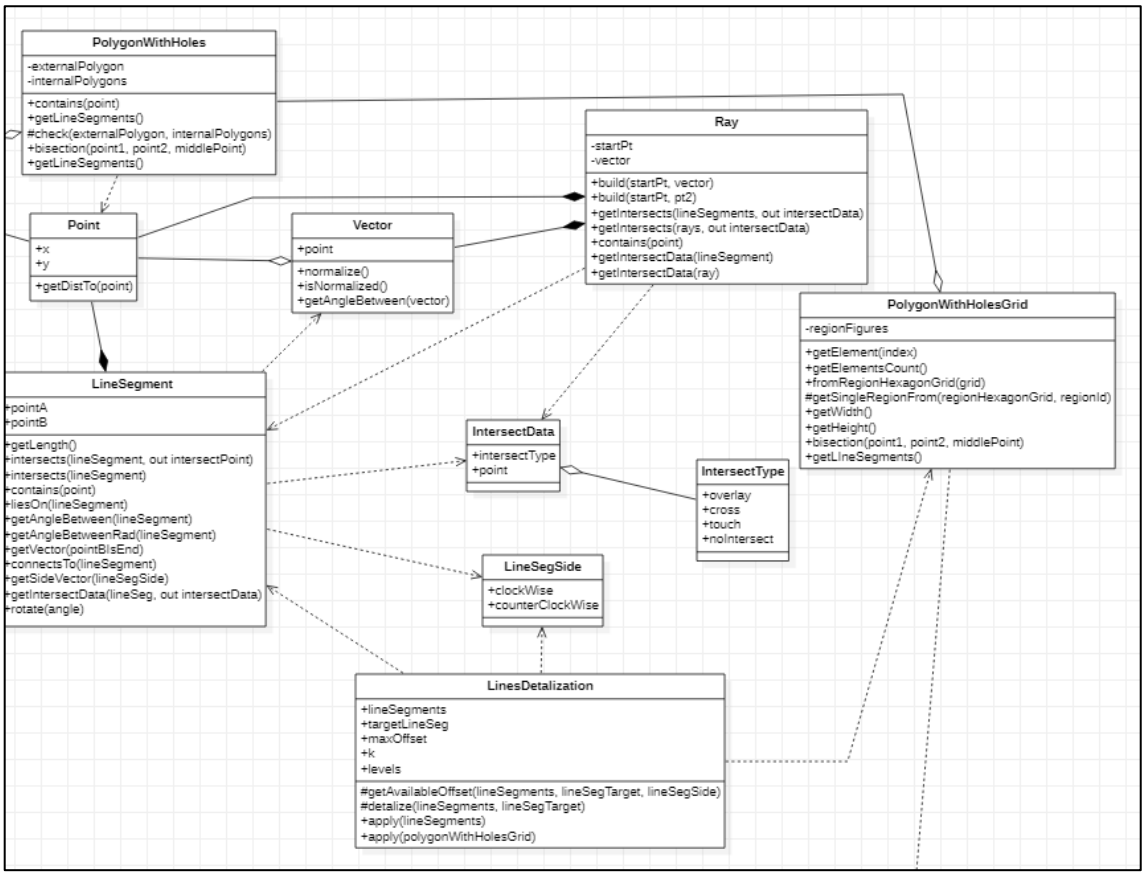

Рисунок 2.3 – класс сетки полигонов («PolygonWithHolesGrid»), детализации («LinesDetalization») и прочие

2.3 Компонент генерации карты высот

Спроектирован класс карты высот и относящиеся к ней (рисунок 2.4). Полная диаграмма представлена на рисунке Б2. Описан базовый функционал:

– получение, запись высоты в нужную ячейку (методы «getHeight» и «setHeight» соответственно);

– получение размера массива карты высот (методы «getCellsCountX» и «getCellsCountY»).

Запланировано сохранение и загрузка карты в виде изображения («saveToBmp» и «loadFromBmp» соотвественно), сохранение и загрузка карты в виде файла («saveToBmp» и «loadFromBmp» соотвественно). Это необходимо потому, что пиксели на изображении могут содержать

дискретный набор значений; если нам нужен неограниченный набор значений высот, требуется сохранять данные другим способом. В данном случае – это сохранение в файл набора вещественных чисел.

Определен ввод и получение физических размеров карты («getSideX», «getSideY»); это может понадобиться для некоторых алгоритмов, которые применяют законы физики, основываются на размерах, скоростях, расстояниях объектов.

Подготовлены и вспомогательные методы для удобного написания кода: получение соседних ячеек относительно выбранной («getTop», «getTopLeft», «getBottom», «getNeighbors» и другие);

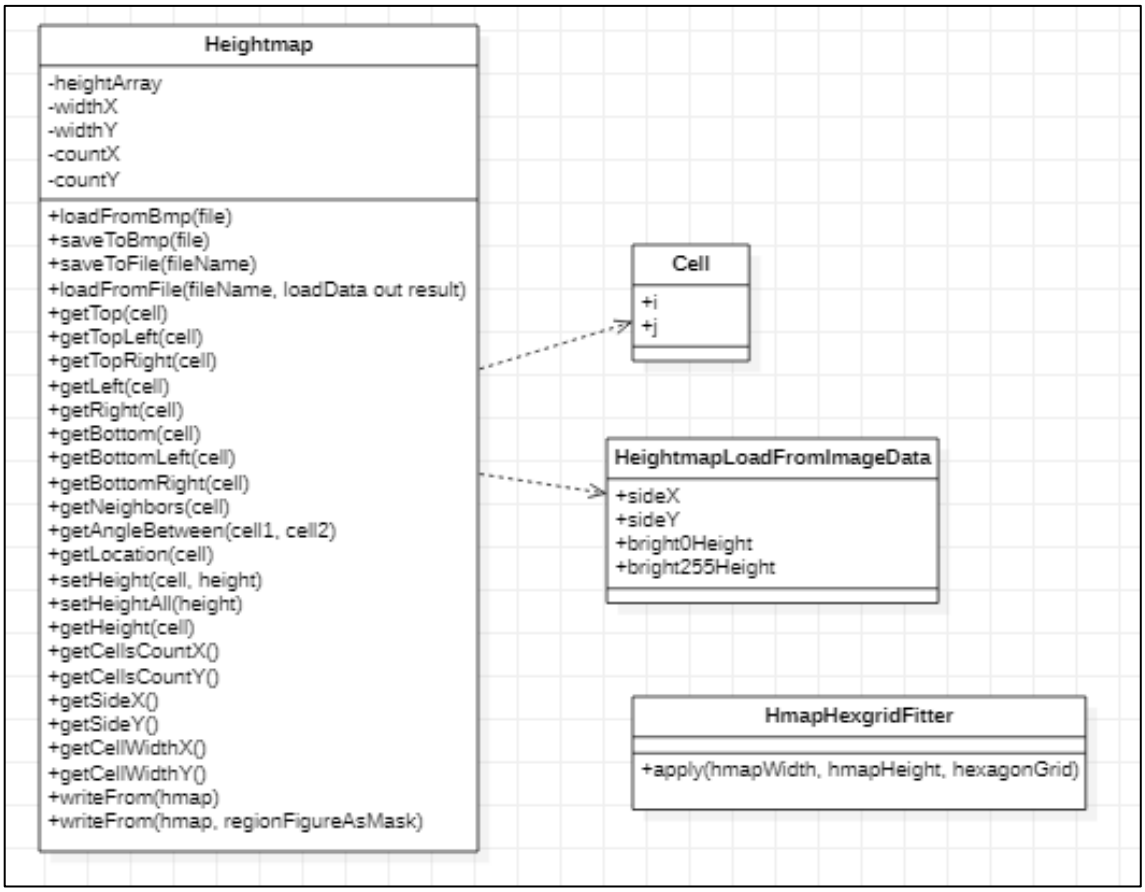

Рисунок 2.4 – класс карты высот «Heightmap», подгона размера карты высот и сетки шестиугольника («HmapHexgridFitter»)

Из-за того, что класс генерации, состыковки и эрозии имеют общее свойство – они изменяют входную карту высот, было принято решение сделать для них общий родительский класс «HeightmapTransformer».

Подготовлены классы генерации и преобразования карты высот (рисунок 2.5). Раннее было отмечено, что при генерации разных участков по разным параметрам, на границах в большинстве случаев будут резкие перепады высот. Класс «Docking» определен для решения данной проблемы. На вход будем подавать максимальный угол (например, если подадим 45 градусов, не будет ни одной местности на карте, где угол между соседними точками будет больше 45).

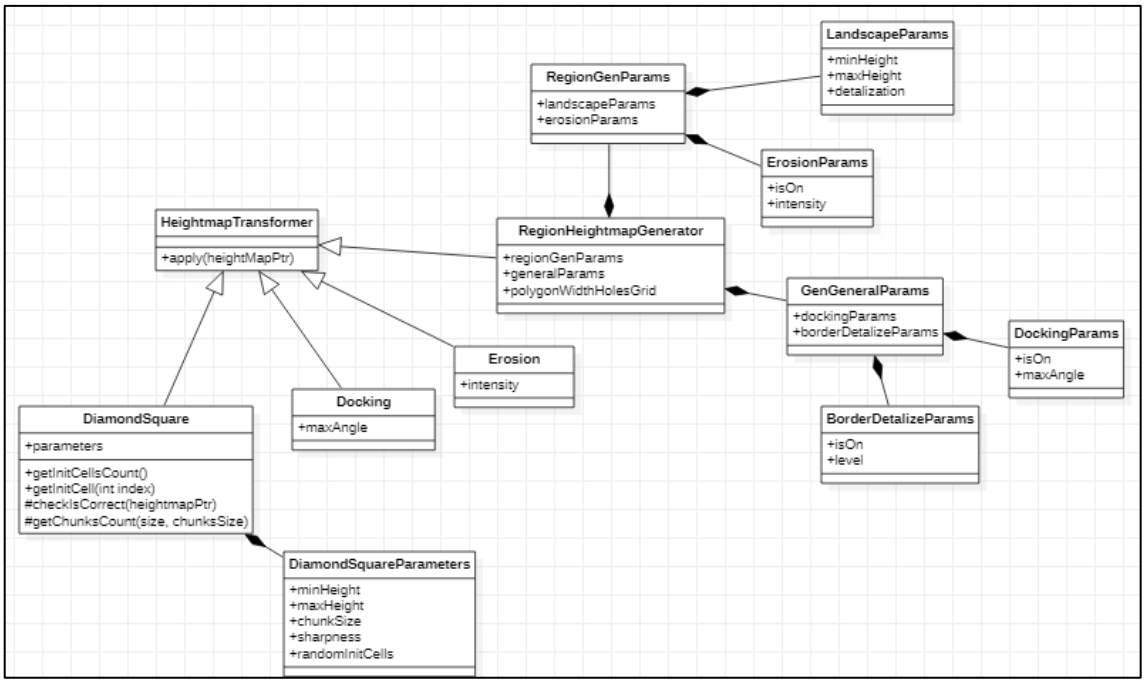

Рисунок 2.5 – класс генерации карты высот «DiamondSquare», состыковки карты высот «Docking», применения эрозии «Erosion» и другие

Класс «RegionHeightmapGenerator» предназначен для объединения функционала вышеописанного: то есть, программисту не придется явно взаимодействовать с классами эрозии, генерации и состыковки.

2.4 Компонент графики

Спроектирован класс отрисовки графики («Graphics») и относящиеся к ней (рисунок 2.6). Полная диаграмма представлена на рисунке Б3. Общий принцип взаимодействия заключается в следующим. Подаются на вход вершины (посредством «AddVertices»). Определяем, как нужно интерпретировать данные («setVerticesInterpret»). Например, мы их можем определить их как набор треугольников: каждые три вершины в массиве – это один треугольник. Иной вариант – как набор прямых.

Посредством «SetDiffuseByTexture» мы задаем текстуру, которую хотим отрисовать на треугольниках (имеет смысл только если интерпретация – набор треугольников). Спроектирован класс камеры: посредством него задается позиция и поворот просмотра сцены.

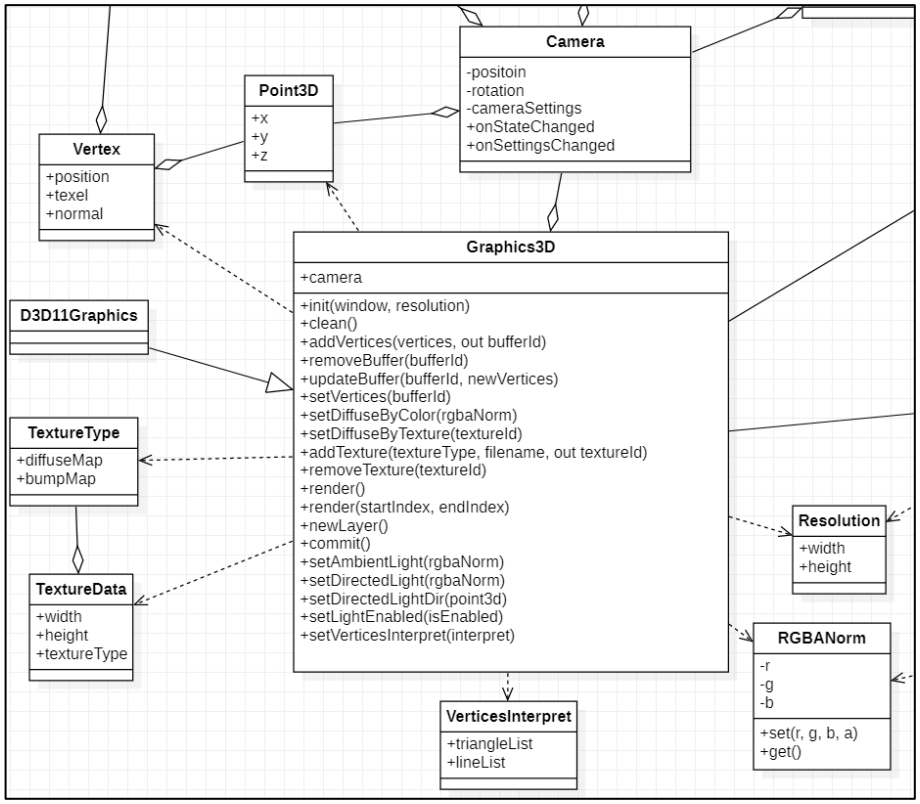

Рисунок 2.6 – класс «Graphics» и связанные с ним

Непосредственно из кода вызывать класс «Graphics» будет непрактичным решением ввиду того, что он предоставляет низкоуровневый функционал. Имеет смысл создать класс, «оборачивающий» данный. Он был спроектирован – это класс «ModelApp» (рисунок 2.7). Напрямую указывать вершины было бы неудобно: с целью упростить работу описан класс «Model». Он содержит в себе метод «load», позволяющий загружать данные из файла.

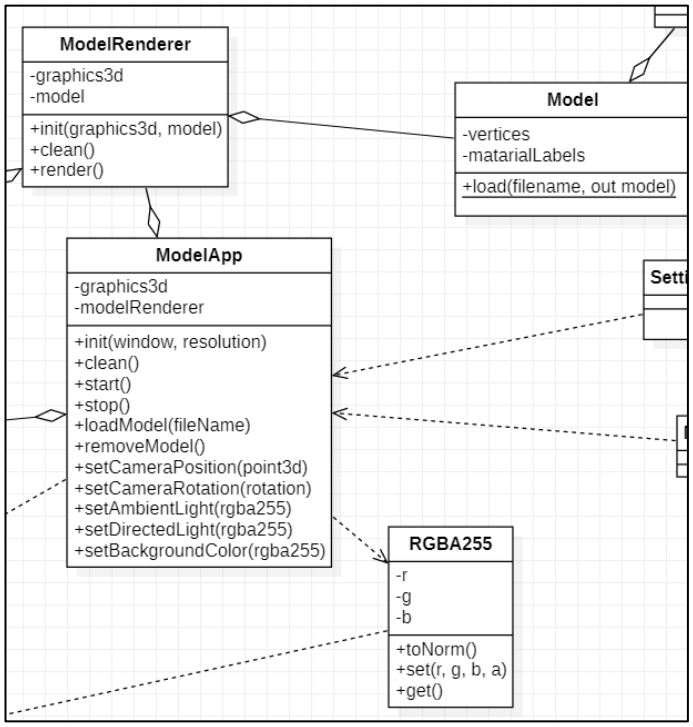

Рисунок 2.7 – класс «ModelApp», «Model» и связанные с ними

3. Практическая часть

3.1 Сетка шестиугольников

Была разработана небольшая программа, предоставляющая интерфейс создания и редактирования сетки шестиугольников (рисунок 3.1) на плоскости. Для итогового варианта ПО будет создан трехмерный интерфейс. Описываемая программа понадобится для первичного этапа разработки, для проведения тестов, когда еще нет функционала трехмерной графики. В ней мы можем выделить регионы, сохранить результат в виде файла, а затем – или загрузить с целью изменения.

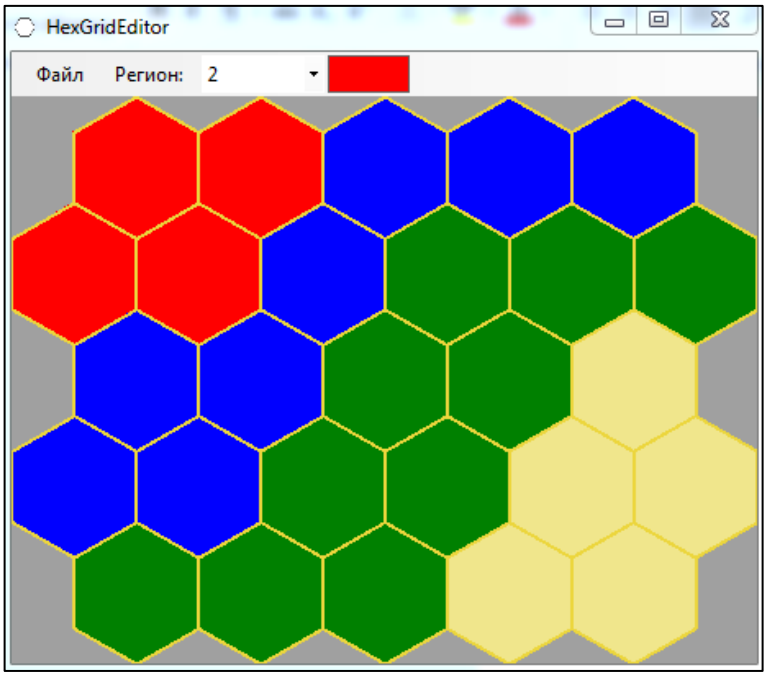

Рисунок 3.1 – редактор сетки шестиугольников

3.2 Детализация границ

Допустим, мы имеем карту биомов на шестиугольной сетке. Есть проблема: эти границы слишком нереалистичны, угловаты. Их следует изменить, повысить их реалистичность. То есть предстоит следующая задача:

совершить детализацию границ (то есть, увеличить число входящих в них отрезков, чтобы добиться большей реалистичности).

Был реализован алгоритм, решающий данную задачу (рисунок 3.2). Во время прохождения данной практики он был доработан (листинг кода В1, В2).

Далее излагается общий принцип. Просчитываются отрезки, которые есть в сетке. Каждый разбивается на два отрезка. Их середина перемещается перпендикулярно изначальному отрезку случайным образом в одну из двух сторон.

При всем вышеописанном отслеживается, чтобы новые отрезки не пересекались с остальными в сетке: рассчитывается максимальное расстояние, на которое можно передвинуть среднюю точку.

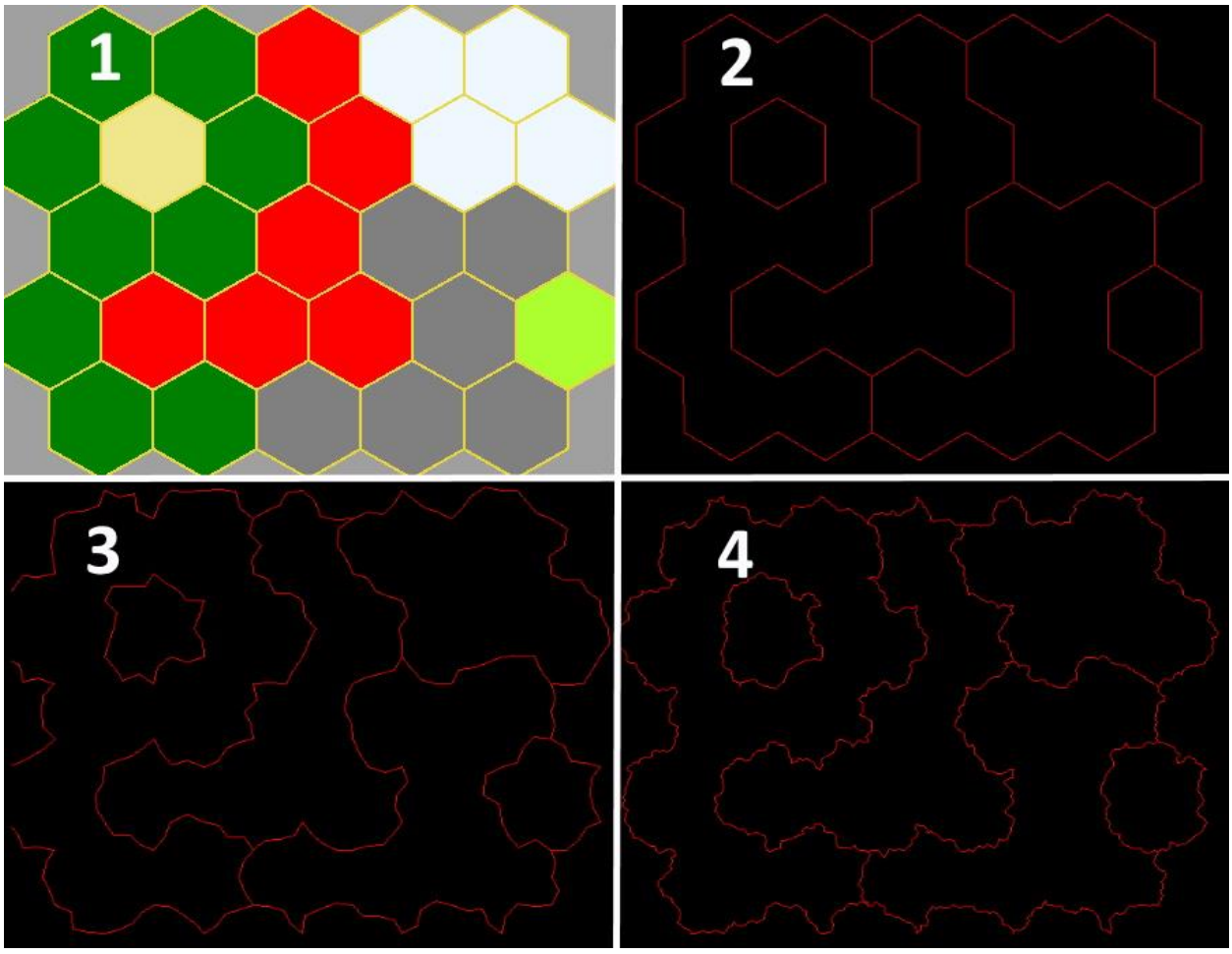

Рисунок 3.2 – сетка шестиугольников (1), получение полигонов из сетки (2), детализация границ (3, 4)

Был внедрен в проект алгоритм diamond-square [5, 6]. Пример работы алгоритма представлен на рисунке 3.3. Посредством него генерируется ладншафт. Можно установить минимальную и максимальную высоту.

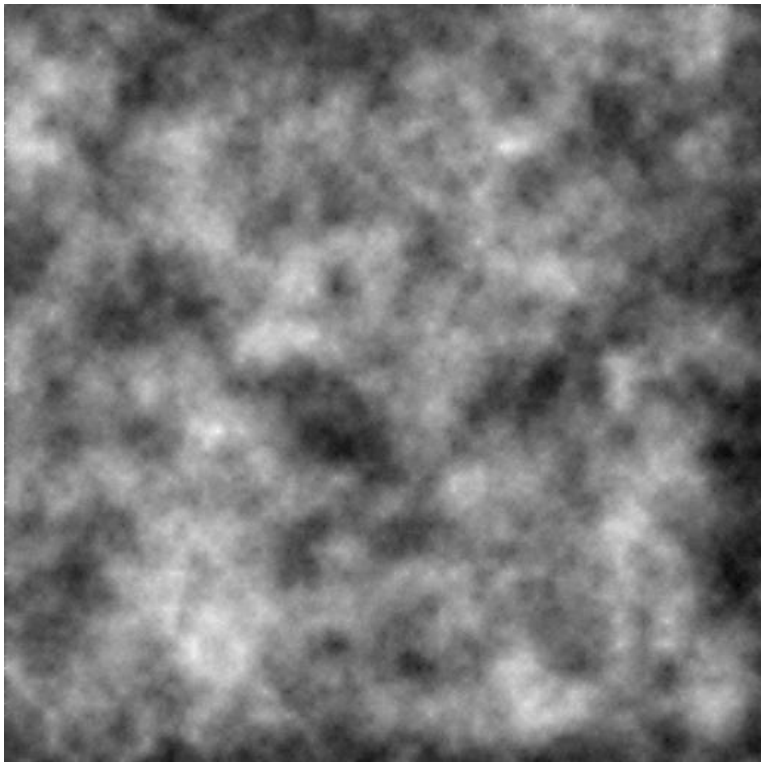

Рисунок 3.3 – сгенерированная карта высот алгоритмом diamond–square

У нас есть продетализированные границы биомов. Теперь нужно сделать следующее: сгенерировать карту высот посредством diamond–square для каждого биома в соответствии с их параметрами. В нашем случае параметры – это пока что лишь минимальная и максимальная высота. То есть, один биом в среднем располагается пониже, другой – повыше. Результат можно наблюдать на рисунке 3.4.

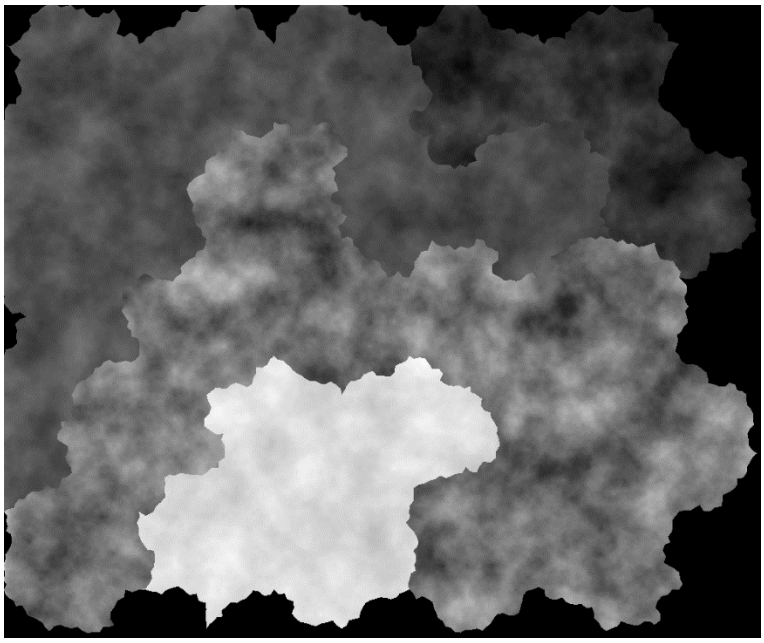

Рисунок 3.4 – карта высот, сгенерированная по разным параметрам

3.3 Состыковка

Проблема заметна: на границах существуют разрывы (рисунок 3.4). Их надо каким–либо образом сгладить, удалить. Чтобы решить данную задачу, был разработан алгоритм состыковки ландшафта.

Общий принцип заключается в следующем: одни ячейки изменяют высоту других ячеек в случае, если разница между ними слишком велика. Затем управление переходит измененным ячейкам. Они, в свою очередь, изменяют высоту своих соседей (так же, в случае максимальной разницы, выше установленной). Процедура повторяется до тех пор, пока не останется ячеек, которые могли мы изменить высоту других.

Разработано два варианта алгоритма (представлены на рисунке 3.5):

– понижение: когда более низкие ячейки понижают высоту более высоких в случае большой разницы;

– повышение: наоборот, более высокие повышают более низкие ячейки.

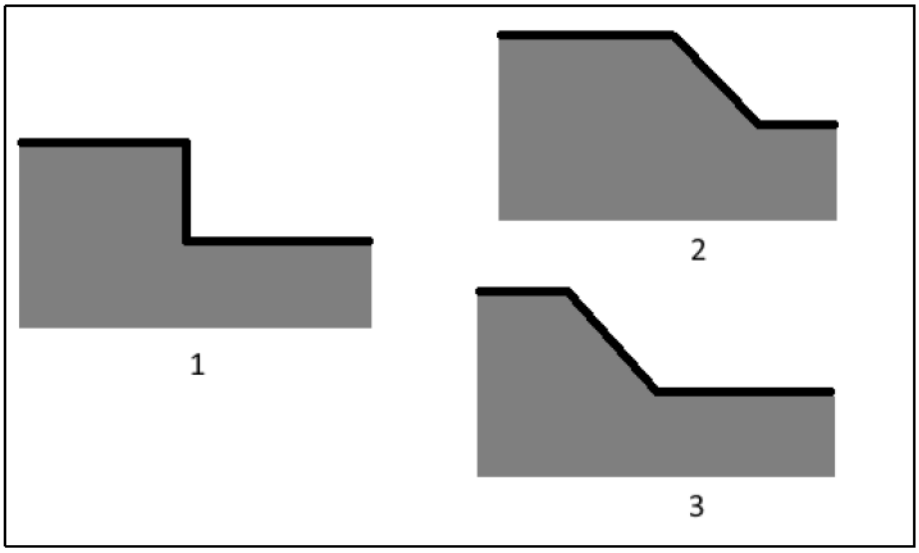

Рисунок 3.5 – до стыковки (1), после повышающей стыковки (2) и понижающей (3)

В итоге мы решили остановиться на понижающем варианте. Он нам может пригодиться далее, если мы захотим состыковывать воду и остальной ландшафт. Так, если вода будет на самом нижнем уровне высоты, в случае понижения ландшафта появится плавный переход между реками и землей: речное пространство не будет затронуто. Если же включить алгоритм на повышение: окружающие, более высокие ячейки просто поднимут все то место, которое должно было быть под водой.

Алгоритм был реализован (листинг кода В.3, В.4, В.5). Первый тест: имеем набор ячеек одной высоты. Но в центре находится точка ниже всех остальных. Второй: круг на большой высоте, все остальное – ниже. Оба теста представлены на рисунке 3.6.

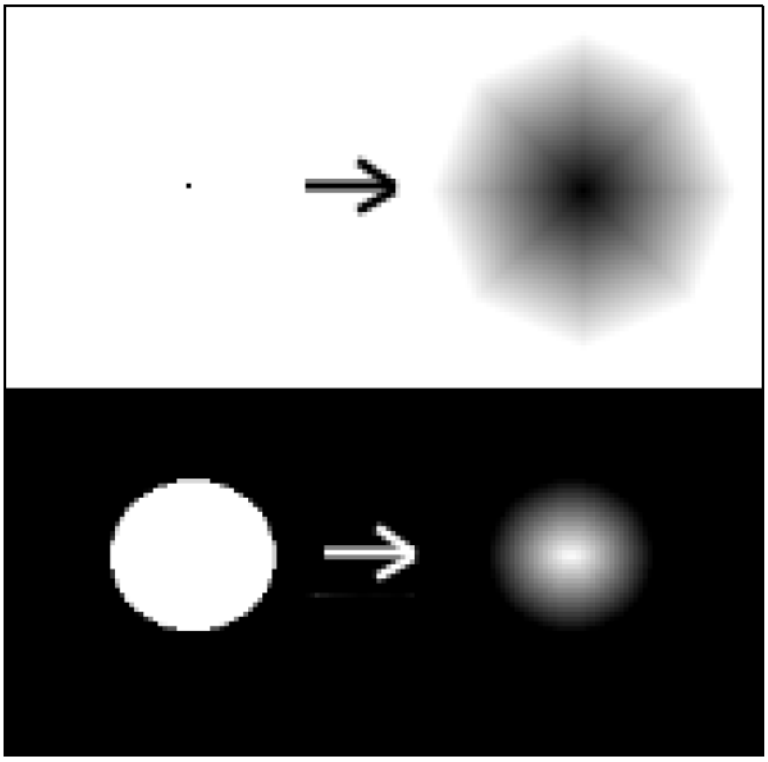

Рисунок 3.6 – работа понижающей стыковки

Далее опробуем алгоритм в обработке сформированного ландшафта. Результат представлен на рисунке 3.7. Можем наблюдать действие во времени: ландшафт постепенно шлифуется, начиная от самых нижних областей до самых высоких.

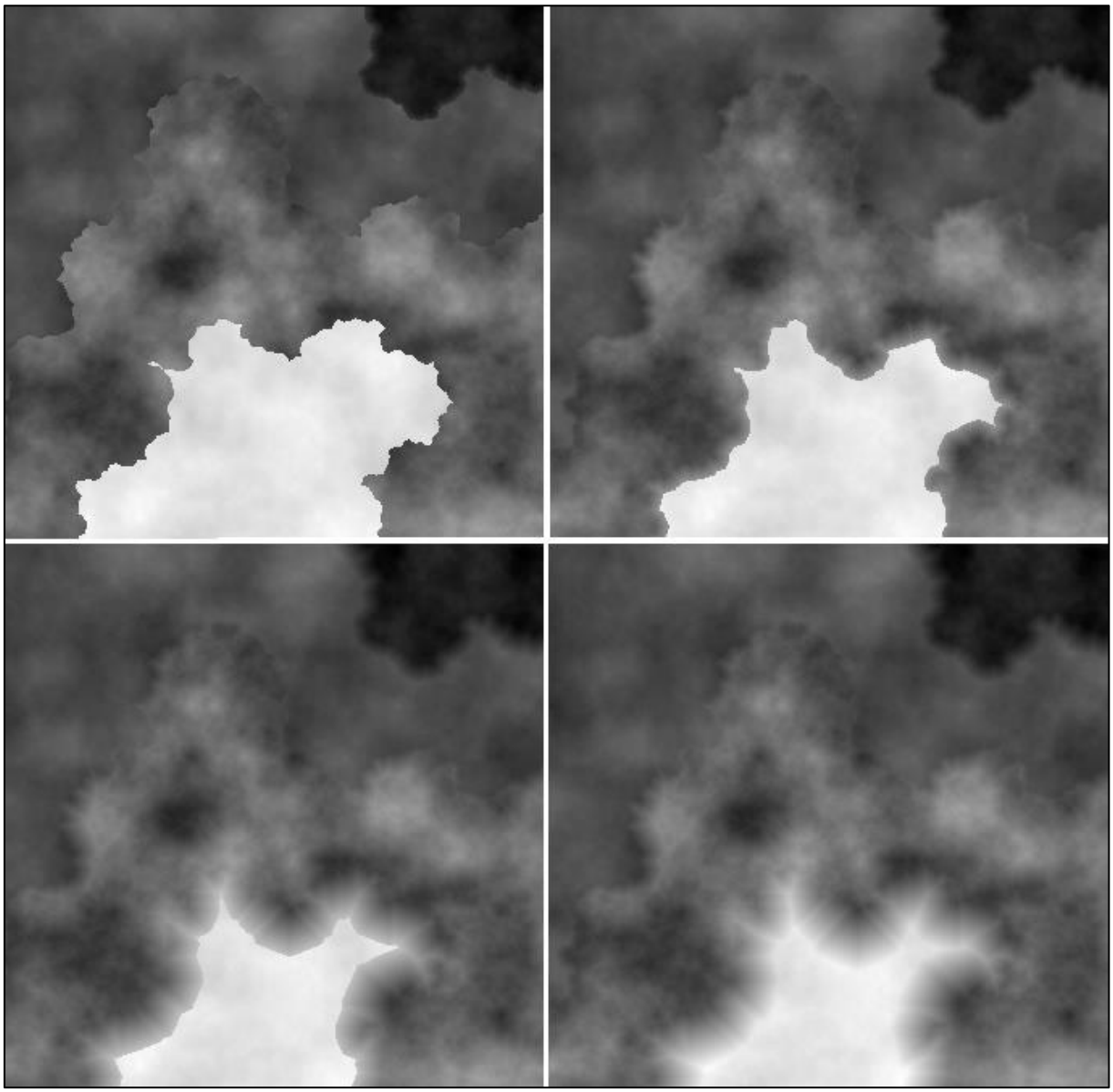

Рисунок 3.7 – работа алгоритма состыковки в последовательности

Попробуем выставить небольшой угол стыковки. Можем наблюдать результат на рисунке 3.8. Ландшафт был полностью выровнен, но при этом практически полностью потерял свой первоначальный облик.

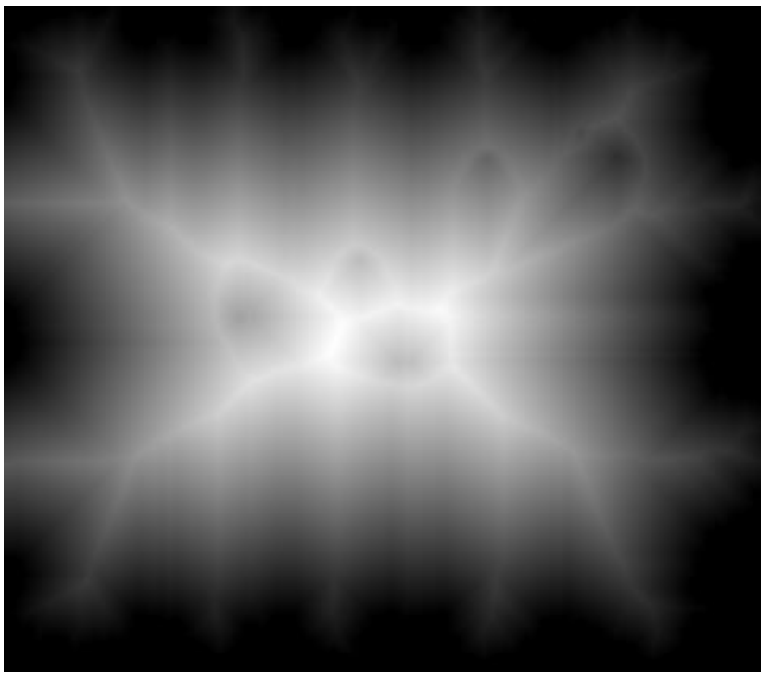

Рисунок 3.8 – результат состыковки в случае малого максимального угла

Изначально в алгоритме задается единственный максимальный угол. То есть, каждая ячейка «руководствуется» одним значением максимального угла. В рамках улучшения работы алгоритма можно задать разные значения максимального угла. Это поднимет уровень реалистичности итоговой картины после стыковки. Она не будет теперь строго линейной, должна отличаться от места к месту.

Результаты работы можем наблюдать на рисунке 3.9. Действительно, очертания карт высот приобрели более уникальный вид. Были взяты те же тесты: точка и линия, которая была нарисована в графическом редакторе.

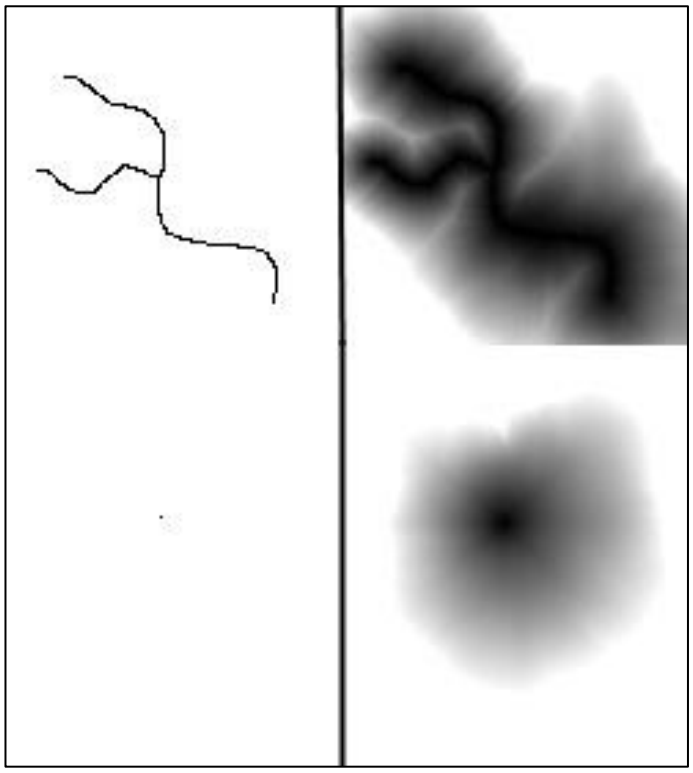

Рисунок 3.9 – стыковка с разными максимальными углами (слева – «до», справа – «после»)

На рисунке 3.8 отражена ситуация: ландшафт может потерять свой облик в случае большого значения угла. Но если точно известно, что нам нужно выставить именно этот угол, чтобы «швы» состыковались как следует. Если не внести дополнительные изменения в алгоритм, нужную стыковку не провести без повреждения облика всего ландшафта.

Для этого поступим на границах областей можно усилить влияние стыковки (то есть, уменьшить максимальный угол), а в других местах – уменьшить.

То есть, на вход алгоритму подадим карту максимальных углов (рисунок 3.10). Каждая ячейка будет брать значения максимального угла из нее. На границах регионов максимальный угол самый низкий (соответствует на рисунке белым точкам). Был реализован функционал для генерации данной карты. Эта карта совпадает с размерами входной карты.

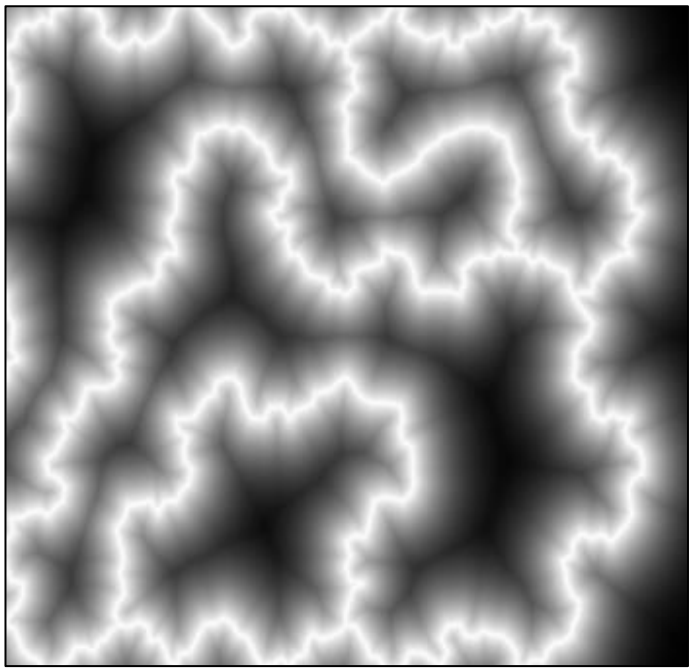

Рисунок 3.10 – карта максимальных углов (белый цвет – минимальное значение, черный – максимальное)

Следует отметить: данный подход можно объединить с предыдущим. То есть, максимальные углы могут быть случайными, но все же, в среднем, на границах пониже, в остальных местах – повыше.

Была произведена стыковка по картам углов. Однако принципиальных различий между данным вариантом и без этого не было замечено. Поэтому данное улучшение было временно убрано. Требуется проверка, дальнейшая доработка.

3.4 Эрозия

Далее был внедрен в проект алгоритм эрозии, представленный в статье «Симуляция эрозии рельефа на основе частиц» [7]. Алгоритм изменениям не был подвернут. Результаты можно наблюдать на рисунках 3.11 и 3.12. Можем наблюдать, как карта была продетализирована, приобрела более реалистичный вид.

Впоследствии был найден подобный вариант алгоритма, реализованный на видеокарте (GPU) [8]. Однако подобный алгоритм сложнее внедрить в проект. Из-за этого было принято решение рассмотреть его позже.

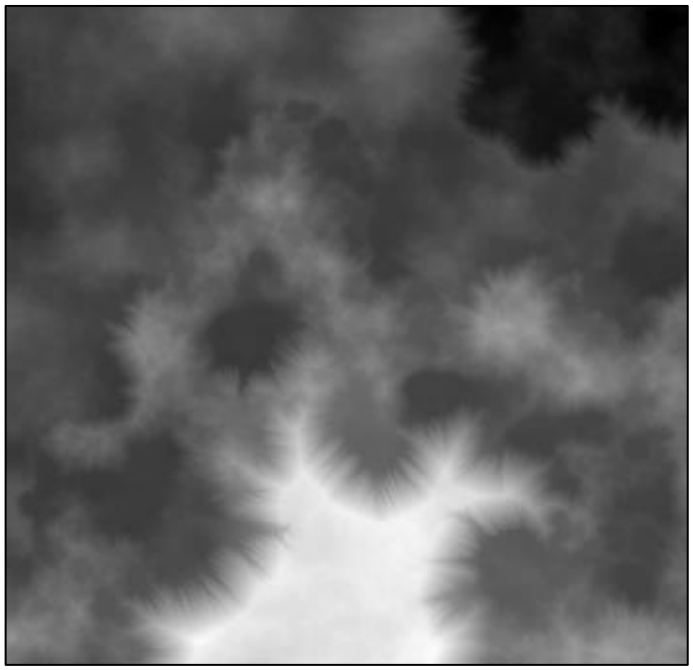

Рисунок 3.11 – средняя эрозия

Принцип следующий: на карте случайным образом появляются частицы. Они перемещаются по физическим законам по карте, «забирая» с собой часть высоты и переносят ее в другое место. В итоге они «испаряются» со временем. Это можно назвать аналогом частиц воды, которые переносят в себе землю, горные породы.

Рисунок 3.12 – сильная эрозия

3.5 Функционал трехмерной графики

Была реализована отрисовка цвета. Есть возможность посредством аргументов функции указать цвет закрашивания посредством задания красного, зеленого и синего компонентов. Далее был реализована отрисовка модели (рисунок 3.13). В данном случае модель – это набор вершин с указанием текстуры, текстурных координат.

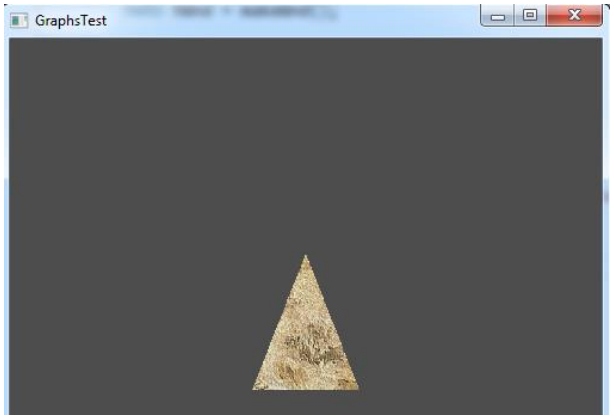

Рисунок 3.13 – отрисовка треугольника с текстурой

Чтобы иметь возможность тестировать координаты мыши, перемещение, луч курсора, требуется сначала реализовать отрисовку текста с возможностью указывать, позицию текста на экране.

Далее следует реализовать рендеринг в текстуру. Это было выполнено. Под этим понимается следующее. До этого речь шла о рендеринге (отрисовке) прямо в окно. Может возникнуть потребность сначала отрисовать на текстуру модели, а затем – отобразить все это вместе. Результат представлен на рисунке 3.14. Видим прямоугольник, на текстуру которого была отражена сцена с зеленым фоном и треугольником.

Рисунок 3.14 – отрисовка текстуры

Базовый графический функционал был реализован раннее. Теперь следует реализовать обработку входных сигналов. Под этим понимается следующее: нажатие кнопок мыши и перемещение мыши, нажатие клавиш.

Отлов нажатия мыши и клавиш был реализован. Принцип следующий: если сигналы были получены, отображаются соответствующие записи в левом углу экрана.

Для перемещения камеры из стороны в сторону требуется фиксация факта перемещения мыши в сторону. Данная функция является основной в сфере трехмерной графики. Чтобы ее осуществлять, необходимо подготовить «отлов» позиции мыши на экране и ее перемещения во времени. Это было сделано. Можем наблюдать тест данного функционала на рисунке 3.15. В левом углу отображается позиция мыши на экране, изменение позиции за недавнее время.

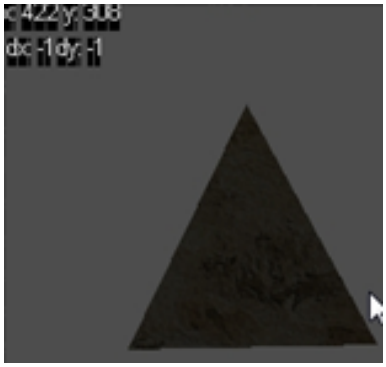

Рисунок 3.15 – фиксация позиции курсора, его перемещения

Следует решить другую задачу: отыскать, куда «указывает» курсор в трехмерном пространстве. За счет данной функции мы сможем в дальнейшем отлавливать пересечение курсора с объектами на сцене. В нашем случае это будет нужно для того, чтобы «закрашивать» курсором сетку шестиугольников.

Тестовый запуск представлен на рисунке 3.16. Принцип теста в следующем: на экране отображаются координаты и вектор луча курсора. Если курсор находится на треугольнике, отображается надпись «Intersection». В ином случае – надпись пропадает.

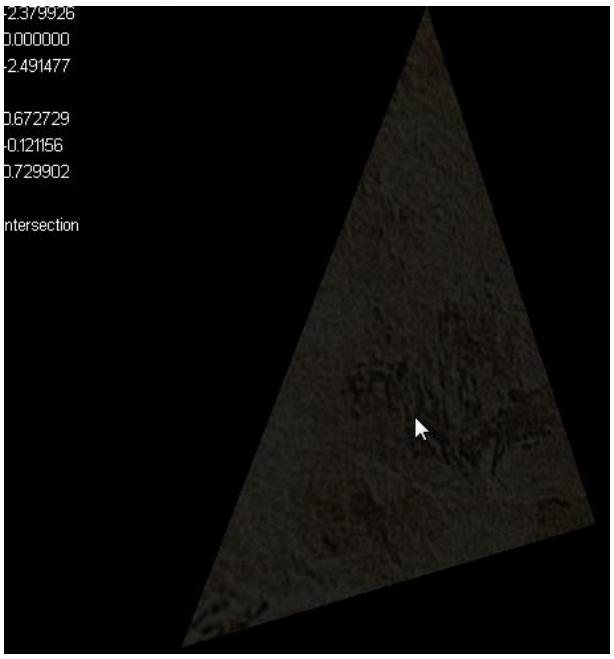

Рисунок 3.16 – определение пересечение курсора с объектом

3.6 Пользовательский интерфейс

Трехмерный интерфейс шестиугольной сетки был реализован (рисунок 3.17). Было реализовано нажатие на шестиугольники. Предоставлен оконный, кнопочный интерфейс для того, чтобы можно было задавать настройки, определять: каким регионом будем закрашивать шестиугольник после нажатия. Выполнено отображение того, какой шестиугольник к какому региону принадлежит (как это было сделано раннее в 2D–интерфейсе).

Пользователь переходит к окну «параметры биомов». Выбирает порядковый номер биома. После выбора в нижней части окна отобразятся окна для ввода минимальной и максимальной высоты для данного биома. Далее в процессе перемещения в трехмерном пространстве, наводя курсор на нужные шестиугольники, пользователем создается разметка карты. В описываемом примере для «белого» биома указаны максимальная высота побольше, для «синего» – поменьше. У «зеленого» были указаны средние значения минимальной и максимальной высоты.

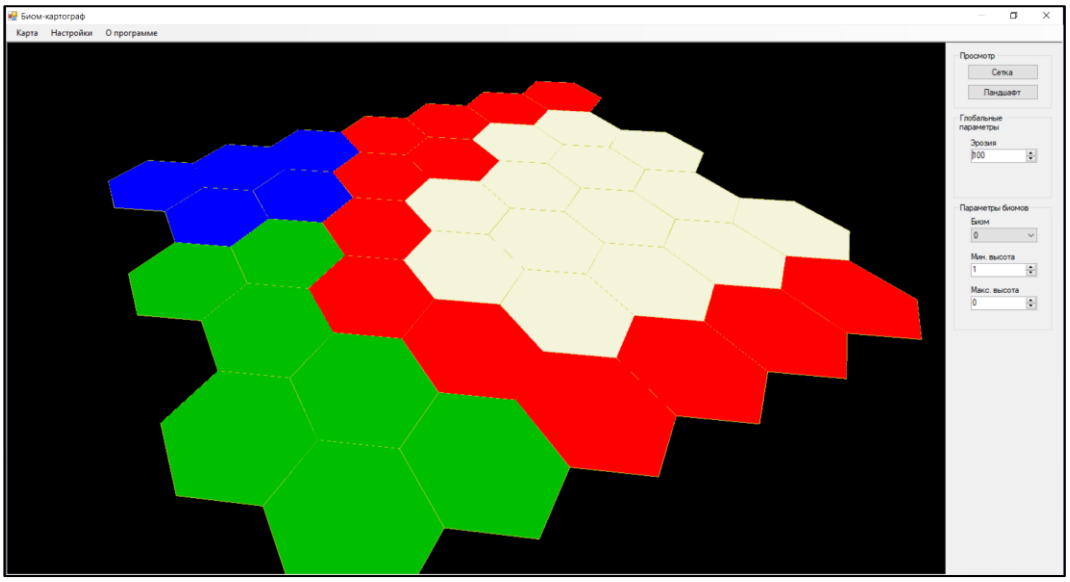

Рисунок 3.17 – карта биомов, нанесение биомов на шестиугольную сетку

Далее, для запуска генерации следует нажать на соответствующую кнопку (находится в меню «Карта», которое находится в верхнем левом углу экрана). Откроется окно с полосой загрузки, после ожидания в случае нормальной работы автоматически откроется окно просмотра ландшафта (кнопка «Ландшафт» в верхнем правом углу экрана). Результат отображен на рисунке 3.18. Можем наблюдать сгенерированную карту высот по нашей разметке биомов. В случае, если результат генерации пользователя не устроил, он может перейти обратно к редактированию сетки (кнопка «Сетка» в верхнем правом углу экрана) и повторить описанный раннее процесс.

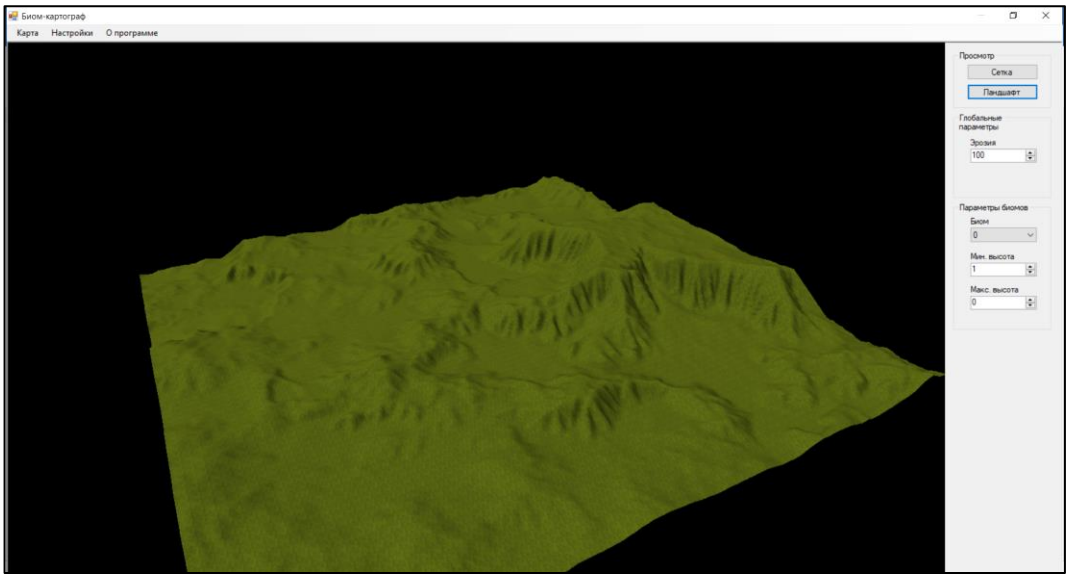

Рисунок 3.18 – сгенерированный ландшафт

ЗАКЛЮЧЕНИЕ

Цель выпускной квалификационной работы была выполнена: выполнено проектирование и разработка программного обеспечения для генерации ландшафта с простым и удобным интерфейсом. Поставленные задачи были выполнены:

- проведен анализ предметной области и доступных аналогов программного обеспечения, эксплуатируемых в предметной области тематики выпускной квалификационной работы;
- разработано техническое задание на проектируемое и разрабатываемое программное обеспечение;
- осуществлено проектирование разрабатываемого программного продукта;
- разработан программный продукт и его интерфейс.

Были получены новые, закреплены раннее полученные знания:

- в сфере информационных технологий;
- в частности, в сфере разработки программного обеспечения;
- в работе с языком программирования C++.

Можем наблюдать острые углы на сетке полигонов после детализации. Это можно доработать в дальнейшем. На основании углов между соседними отрезками можно рассчитать, куда следует перемещать среднюю точку, чтобы не появлялись острые углы, либо способствовать уменьшению этой остроты.

Касательно состыковки: было бы лучше, если бы линия перехода не «уходила» в сторону более высокой или низкой части (в зависимости от вида алгоритма, см. рисунок 18), а находилась строго в середине. Есть предположения как это примерно можно сделать: посредством специального поочередного применения обоих алгоритмов. Однако данная концепция не была разработана до конца, подлежит уточнению, доработке.

Что можно сказать в итоге касательно алгоритма эрозии: в дальнейшем он может быть подвергнут улучшению. То есть, все места, ячейки на карте изначально обладают одинаковой «твердостью». То есть, размываются одинаково. Можно сделать так, чтобы одни области размывались сложнее, другие – проще. Так, это будет похоже на ситуацию в природе: есть камни, почва, песок. И в итоге, возможно, мы сможем наблюдать большее разнообразие ландшафта. Например, сохранившиеся, выступающие пики каменных глыб, которые мы можем наблюдать в реальной природе в горах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Технологии разработки программного обеспечения: Учебник/ С. Орлов. 5–е издание обновленное и дополненное — СПб.: Питер, 2018. — 640 с.: ил.

2. Уроки программирования на языке C++. – URL: https://ravesli.com/uroki-cpp/ (Дата обращения: 11.06.2021)

3. Документация по Microsoft C++, C и ассемблеру – URL: https://docs.microsoft.com/ru-ru/cpp/?view=msvc-160 (Дата обращения: 11.06.2021)

4. Polygonal Map Generation for Games – URL: http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/ (Дата обращения: 14.06.2021)

5. AutoBiomes: procedural generation of multi-biome landscapes– URL: https://link.springer.com/article/10.1007/s00371-020-01920-7 (Дата обращения: 14.06.2021)

6. Алгоритм «diamond–square» для построения фрактальных ландшафтов. – URL: https://habr.com/ru/post/111538/ (Дата обращения: 11.06.2021)

7. Симуляция эрозии рельефа на основе частиц. – URL: https://habr.com/ru/post/496762/ (Дата обращения: 11.06.2021)

8. Real–Time Massive Terrain Generation using Procedural Erosion on the GPU. – URL: https://web.mit.edu/cesium/Public/terrain.pdf (Дата обращения: 11.06.2021)

9. Windows API – URL: https://docs.microsoft.com/en-us/windows/win32/apiindex/windows-api-list (Дата обращения: 11.06.2021)

10. DirectX graphics and gaming – URL: https://docs.microsoft.com/en-us/windows/win32/directx (Дата обращения: 11.06.2021)

11. Визуальное моделирование систем в StarUML: Учебное пособие/ А.В. Каюмова. Казань. – Казанский федеральный университет, 2013. – 104с.

12. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя. М.: ДМК Пресс, 2007 – 489 с. (http://www.knigafund.ru/books/106240)

13. StarUML. Руководство пользователя. Перевод Д.В. Летуновского, 2007. – 207 с.

14. Процедурная генерация ландшафтов. – URL: masters.donntu.org/2019/fknt/poletaev/ind/index.htm#:~:text=Под%20процедурной%20генерацией%20понимается%20использование,целых%20планет%2C%20сюжеты%20литературных%20произведений. (Дата обращения: 11.06.2021)

15. Процедурная гидрология: динамическая симуляция рек и озёр. – URL: https://habr.com/ru/post/498290/ (Дата обращения: 13.03.2021)

16. Уроки программирования на языке C++. – URL: https://metanit.com/cpp/tutorial/1.1.php (Дата обращения: 11.06..2021)

17. Скайбокс (объект в трехмерной графике) – URL: https://ru.wikipedia.org/wiki/Скайбокс (Дата обращения: 11.06..2021)

18. Группы сглаживания (объект в трехмерной графике) – URL: https://render.ru/ru/s.vatkin/post/11656 (Дата обращения: 11.06..2021)

19. Уроки по DirectX 11. – URL: http://www.rastertek.com/tutdx11.html (Дата обращения: 11.06..2021)

20. Direct3D 11 на C++ с нуля. – URL: http://d3dbegin.narod.ru/dx11tutorial01.htm (Дата обращения: 11.06..2021)

ПРИЛОЖЕНИЯ

Приложение А

Техническое задание

УТВЕРЖДАЮ

Научный руководитель

_____/Е.И. Сафонов /

«____» _____ 2021 г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ
«РАЗРАБОТКА ПРОГРАММНОЙ СРЕДЫ ДЛЯ ГЕНЕРАЦИИ ЛАНДШАФТА ПО КАРТЕ БИОМОВ»

Ханты-Мансийск 2021 г.

1. Общие сведения

Настоящее техническое задание распространяется на разработку программы по генерации цифрового ландшафта. Под ним понимается файл, который представляется следующими данными:

– размеры рассматриваемого участка (в условных единицах);

– высота пространства в той или иной области (где равнины, где холмы, а где – горы или овраги);

Разрабатываемое ПО нацелено на то, чтобы предоставить функционал и интерфейс пользователю для работы с такими файлами, для настройки и запуска генерации ландшафта.

В данном документе были применены сокращения. Их список, с указанием полных названий, представлен в таблице 1.1.

Таблица 1.1 – сокращения и их полные названия

Сокращение	Полное название
ПО	Программное обеспечение
ЛКМ	Левая кнопка мыши
ПКМ	Правая кнопка мыши

2. Основание для разработки

Большинство аналогов («World machine», «Terragen», «InstantTerra», «MapMagic» и другие), имеют сложный в эксплуатации инструментарий. Требуется простое в освоении и использовании ПО, посредством которого пользователь сможет указать, что ему примерно хочется видеть там или здесь, не вдаваясь в детали.

3. Назначение

Построение природы в цифровом виде может понадобиться разработчикам компьютерных игр. Особенно это ПО подойдет для игровых проектов с открытым миром. Там, как правило, требуется создавать большие природные пространства. Также оно может принести пользу тем, кто занимается компьютерной графикой в целом. Например, для ролика, анимации или картинки с трехмерным пейзажем.

4. Требования к программному изделию

4.1 Требования к функциональным характеристикам

4.1.1 Общая концепция

Работу с ПО пользователем в общих чертах можно описать следующим образом (все описываемые функции следует реализовать):

– определение границ биомов;

– для каждого биома задаются параметры рельефа;

– пользователь по своей инициативе начинает генерацию содержимого для всех биомов сразу; каждый биом будет наполнен тем, что указано в конкретно его параметрах;

– выгрузка данных файла: карты высот, трехмерной модели ландшафта (в формате «.obj»); указание пути к сохранению соответствующих файлов;

При удержании ЛКМ и перемещении мыши в нужную сторону камера просмотра поворачивается соответственно. При удержании определенных клавиш клавиатуры должно происходить перемещение камеры вперед, назад, влево, вправо.

Требуется интерфейс, который предоставит возможность пользователю работать с сеткой шестиугольников (множество шестиугольников, состыкованных друг с другом.). Пример такой сетки представлен на рисунке 4.1. Посредством нее пользователь должен размечать пространство. Речь идет о том, чтобы отметить, какой шестиугольник к какому биому принадлежит. То есть, предварительно выбирается биом, которым будет происходить «закраска» (требуется соответствующая интерфейсная панель). Затем, при нажатии ЛКМ и наведении курсора пользователем на нужный шестиугольник, он должен «закраситься» соответствующим биомом.

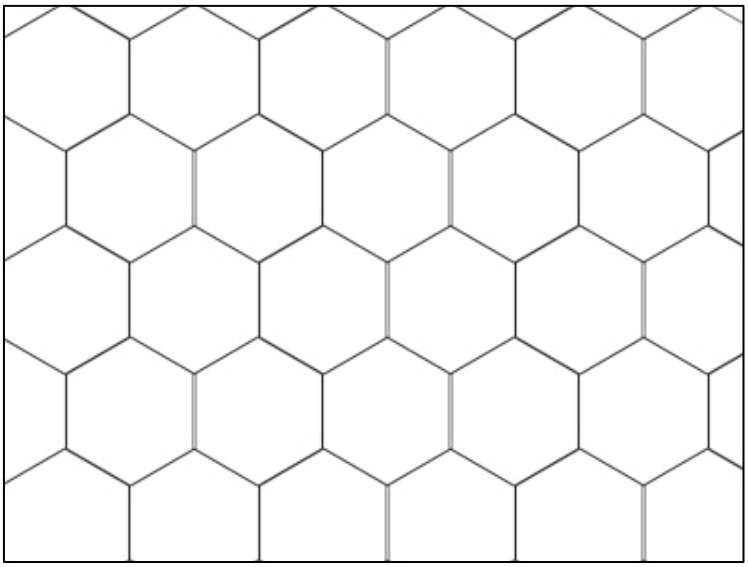

Рисунок 4.1 – сетка шестиугольников

4.1.2 Генерация

Биом – это местность, которая располагается внутри одного многоугольника. Установленные пользователем параметры для каждого биома должны сохраняться на протяжении работы программы, чтобы для повторной генерации вводить все сначала.

Для доступа к настройкам генерации биома требуется выбрать данный биом среди списка (требуется соответствующая панель). После этого должна отобразиться его параметры. В нее входит название: можно выбирать свое. Изначально автоматически задается системой.

Должны быть определены следующие параметры биома:

– минимальная высота;
– максимальная высота;

Также должен быть еще один параметр, актуальный для обоих типов биомов: алгоритм эрозии (под ним понимается некоторый алгоритм, который преобразовывает ландшафт, имитируя процессы вымывания грунта, камня; как следствие, он добавляет детализацию и реализм ландшафту), его интенсивность.

4.1.3 Многоугольная и реальная граница

Как было описано раннее, биомы имеют многоугольные границы. Но после генерации мы не должны видеть четкие прямоугольные границы между биомами: это совершенно нереалистично. Они должны быть размыты, детализированы. Такую границу будем называть «реальной». Пример таковой представлен на рисунке 4.2. Однако сама сетка шестиугольников не должна быть изменена таким образом. Вместо этого эта новая детализированная граница должна быть актуальна только на момент генерации.

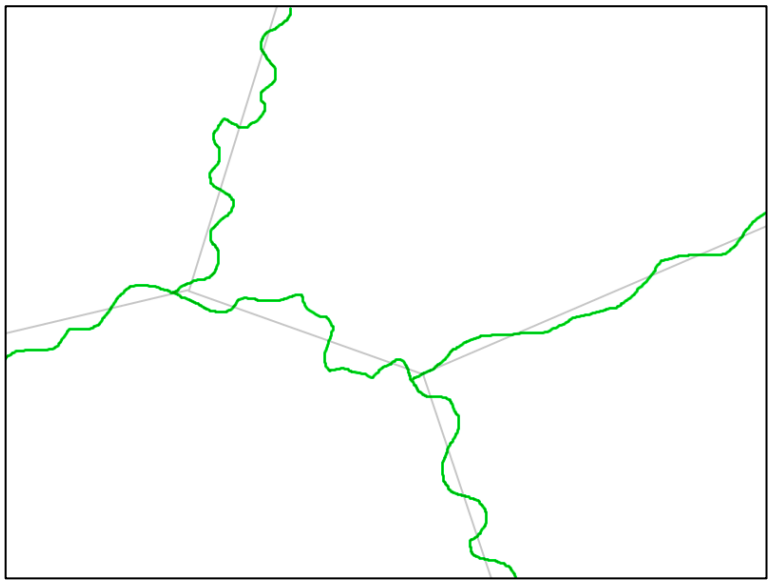

Рисунок 4.2 – многоугольные границы (светло-серые) и реальные границы (зеленые)

4.1.4 Настройки

Должна быть возможность определять и изменять следующие параметры:

– чувствительность мыши при повороте камеры;

– какие клавиши назначены для перемещения камеры вперед, назад, влево, вправо;

– клавиша, по которой включается и выключается вращение камеры курсором.

4.2 Требования к надежности

Предусмотреть контроль вводимой информации. Предусмотреть блокировку некорректных действий пользователя при работе с системой.

4.3 Требования к составу и параметрам технических средств, программной совместимости

Требуется следующие параметры технических средств:

- периферийное оборудование: монитор, клавиатура, мышь;
- операционная система: windows 7 и выше.
- программное обеспечение: Microsoft .NET framework 4.5 и выше, Microsoft Visual C++ 2017 Redistributable, DirectX 11.

5. Требования к программной документации

Разрабатываемые программные модули должны быть самодокументированы, т. е. тексты программ должны содержать все необходимые комментарии.

Разрабатываемая программа должна включать справочную информацию о работе программы.

В состав сопровождающей документации должны входить:

- пояснительная записка, содержащая описание разработки;
- руководство пользователя.

6. Технико–экономические показатели

Эффективность системы определяется удобством использования системы, реалистичностью сгенерированного ландшафта.

7. Календарный план работ

Создание ПО подразумевает под собой выполнение этапов, описанных в таблице 7.1.

Таблица 7.1 – календарный план выполнения работ

Наименование этапа работы над ВКР	Начало	Окончание
Разработка концепции	10.03.2021	Разработанная концепция проекта
Анализ требований к ПО	27.03.2021	Use–Case диаграммы, текстовое описание требований
Проектирование ПО	25.04.2021	Диаграммы классов, компонентов
Реализация генерации ландшафта	15.05.2021	Разработанный программный компонент по генерации ландшафта
Реализация графического трехмерного интерфейса	15.06.2021	Разработанный программный компонент графического интерфейса. Готовое ПО.