Содержание

[Введение 2](#_Toc534840615)

[1. Существующие решения по моделированию рельефов 2](#_Toc534840616)

[2. Выбор среды разработки 3D-приложений 5](#_Toc534840617)

[3. Архитектура приложения 6](#_Toc534840618)

[4. Алгоритмы и подходы по генерации рельефа 7](#_Toc534840619)

[Заключение 13](#_Toc534840620)

[Список используемой литературы 13](#_Toc534840621)

## Введение

Трехмерные модели рельефа используются в составе программно-алгоритмических систем различного назначения. Сюда входят [**1**] тренажеры-симуляторы для обучения операторов робототехнических комплексов, военные тренировочные комплексы [**2**], имитационные моделирующие комплексы для виртуальных испытаний мобильных роботов [**3:** ссылка с примером], игровые [**4:** ссылка с примером, где шум Перлина используется] и другие приложения. Моделирование рельефа зачастую не является самоцелью, но все же представляет собой важный этап в работе программного комплекса. Так, без сложного, динамически изменяющегося виртуального пространства невозможно отрабатывать алгоритмы интеллектуального управления автономными аппаратами, адекватно моделировать работу некоторых датчиков, бортового оборудования и так далее.

[1] – 2011 – [С] Метод адаптивной генерации ландшафта.

[2] – 2010 – [С] Declarative Terrain Modeling for Military Training Games.

[3]

[4]

Здесь основные подходы обзорно к построению и моделированию рельефа. Главное – какие проблемы решаются, решены и не решены.

Целью данной работы является разработка алгоритмических и программных средств генерации подводного рельефа, приближенного к реальному. Основными требованиями к программной системе являются качество получаемого рельефа (отсутствие артефактов, высокая детализация карты высот), разнообразие типов поверхностей (песчаная отмель, песчаные холмы, россыпи камней, скальные образования, подводные горы и др.), а также контролируемость генерации и возможность настройки параметров рельефа. Генератор рельефа является частью программного модуля генерации подводной среды в составе имитационного моделирующего комплекса для отработки одиночных и групповых алгоритмов интеллектуального управления автономными подводными роботами.

В разделе 1 описываются известные алгоритмы генерации рельефа, их особенности и специфика использования, выделяются достоинства и недостатки алгоритмов относительно поставленной цели. Раздел 2 содержит информацию о состоянии отрасли игровых движков, один из которых используется при реализации программного модуля генерации рельефа; обосновывается выбор среды программирования и причины использования программного обеспечения подобного рода. Информацию об архитектуре генератора рельефа, предложенных подходах к генерации и представлению рельефа, а также особенности графического пользовательского интерфейса можно увидеть в разделах 3 и 4. Заключение содержит выводы о результатах проведенной работы и планы дальнейшей разработки программной системы.

## 1. Существующие решения по моделированию рельефа

Моделирование рельефа занимает отдельное место в структуре программного обеспечения, разрабатываемого для различных нужд [1–4], а инструменты и подходы по созданию 3d-моделей ландшафта делятся на три основные группы [**5**].

[5] – 2005 – [С] Terrain Generation Using Genetic Algorithms.

В первую группу входит создание рельефа путем «рисования» (sketching). Для этих целей используются растровые графические редакторы, такие как Photoshop, которые с помощью различных инструментов (кистей и фильтров) позволяют сформировать текстуру в градациях серого. Полученное изображение представляет собой карту высот, где абсолютно белый цвет соответствует самой высокой точке на карте со значением «1», а абсолютно черный – самой низкой со значением «0». Такую текстуру можно загрузить в какую-либо среду разработки 3D-приложений (Unity [ссылка], Unreal Engine [ссылка], Cry Engine [ссылка] и т.п.**)**, и восстановить рельеф по яркости пикселей. Кроме того, подобные среды предлагают собственные инструменты по изменению 3d-моделей рельефа в интерактивном режиме. Данный подход к созданию трехмерных моделей ландшафтов обеспечивает разнообразие и реалистичность картинки, позволяет провести проработку особенностей и деталей рельефа с желаемой точностью, но занимает в разы больше времени по сравнению со способами, основанными на автоматизации…. В тех случаях, когда создание рельефа является второстепенной задачей, подход на основе «рисования» является неэффективным и трудозатратным.

Ещё одна группа методик формирования виртуальных ландшафтов заключается в воссоздании карты высот по реальным снимкам или данным гидрографических промеров. Подобные модели предоставляют возможности по исследованию реальных физических процессов (течений, динамики изменения русел и пойм рек и другого) [ссылка на статью про подобное моделирование дна моря]. Данные методы отличаются высокой реалистичностью полученных рельефов, но не обеспечивает необходимого разнообразия, которое требуется при решении некоторых задач. Помимо этого, иногда возникает необходимость в проведении ручного или алгоритмического анализа снимков с целью определения типа рельефа (песчаный или каменистый, с растительностью или без и т.д.). Очевидно, что распознавание и классификация снимков усложняется в тех случаях, когда нужно выбрать часть поверхности, которая сочетает в себе два и более типа рельефов.

Последняя группа способов создания ландшафтов заключается в процедурной генерации карты местности. Генераторы рельефа, в зависимости от поставленных целей, используют различные алгоритмы или их комбинации для получения рельефов нужных типов. Применение процедурного подхода позволяет сократить затраты человеческих ресурсов, в разы повышает скорость создания рельефа, обеспечивают разнообразие, а также предоставляют возможности по гибкой настройке параметров генерации. Рассмотрим основные алгоритмы процедурной генерации карт высот.

1. *Использование функционального описания*. Идея алгоритма заключается в представлении трехмерной поверхности рельефа в виде аналитической функции двух переменных – пространственных координат горизонтальной плоскости, – которым ставится в соответствие значение высоты в каждой точке пространства. Основная сложность заключается в создании единой для всего рельефа формулы вида z = f(x, y), которая бы удовлетворяла требованиям к его внешнему виду.

Частным случаем использования функционального описания является *холмовой алгоритм* (*«hill rising»*): одна его итерация заключается в генерации поверхности вида:

z = r2 – ((x2 – x1)2 + (y2 – y1)2) (1),

где (x1, y1) – текущая точка вычислений, r – радиус холма, (x2, y2) – вершина холма.

Каждая полученная поверхность накладывается на рельеф, сформированный на предыдущем шаге. Таким образом, создание рельефа состоит в генерации и наложении некоторого количества поверхностей вида (1).

2. *Алгоритм смещения точки («particle deposition»).* Основная идея алгоритма заключается в том, чтобы имитировать процесс осыпания «частиц». На рельеф в заданное место помещается частица, и если она находится слишком высоко над окружающими частицами, то скатывается в случайном направлении. Размер частицы может варьироваться в различных пределах. После использования данного алгоритма может потребоваться применение сглаживающего фильтра.

[ССЫЛКИ на 1 и 2 пункты?]

*3. Фрактальные ландшафты* [2012 – [B.T.] Generation and Visualization of Terrain in Virtual Environment]*.* Поверхность генерируется с использованием стохастического алгоритма, воспроизводящего фрактальное поведение. Многие формы рельефа обладают статистическим самоподобием и могут быть представимы с достаточной степенью реалистичности с помощью фракталов. [1990 – [С] Язык фракталов] Результат процедуры генерации фрактального ландшафта не детерминирован, поверхность случайна, но проявляет в основе своей фрактальное поведение.

*Алгоритм линии разлома («fault line», «fault formation»)*, который является одним из примеров алгоритма генерации фрактального рельефа, заключается в итеративном процессе генерации случайной прямой и дальнейшем увеличении значений высот в одной полуплоскости и уменьшении в другой. Для достижения эффекта сглаженного рельефа необходима постобработка карты высот, например, с помощью фильтра Гаусса. [2003 – [К] Focus On 3D Terrain Programming]

*Алгоритм случайного смещения центральной точки («midpoint displacement»)* в двумерном случае заключается в разбиении первоначального отрезка пополам и смещении центральной точки на случайный коэффициент, находящийся в пределах, пропорциональных масштабу разбиения. [2003 – [К] Focus On 3D Terrain Programming – там же есть картинки для объяснения] *Алгоритм «ромб–квадрат» («diamond-square»)* [1996 – [С] Dynamic Terrain Generation Based on Multifractal Techniques], [2012 – [B.T.] Generation and Visualization of Terrain in Virtual Environment]и *алгоритм «деления треугольника» («triangle division»)* [2006 – [С] Algorithms for Generating Fractal Landscapes] – варианты модификации алгоритма случайного смещения центральной точки, обеспечивающие большую реалистичность генерируемых рельефов.

Фрактальные алгоритмы являются самой распространенной категорией алгоритмов по генерации реалистичных рельефов. По сравнению с другими, они отличаются одними из самых высоких показателей скорости вычислений [2011 – [С] Procedurally Generating Terrain – оценки эффективности алгоритмов], однако их использование не подразумевает генерации различных типов рельефов в пределах одной карты, следовательно, возникает необходимость модификации алгоритмов под конкретные нужды. Кроме того, текстуры, полученные с помощью данных алгоритмов, могут содержать артефакты (непредсказуемые скачки высот в пределах нескольких соседних значений). Их появление объясняется спецификой самого алгоритма и выбором коэффициентов генерации [2011 – [С] Procedurally Generating Terrain].

4. *Шумовые алгоритмы.* [2010 – [С] Challenges in Procedural Terrain Generation] Основными в данной категории алгоритмов являются *«value noise»*, *«Perlin noise» (шум Перлина)* и *«simplex noise»* [2011 – [С] Procedurally Generating Terrain], [2014 – [К] Framework for Real-Time Editing of Endless Procedural Terrains], [2014 – [С] Designer Worlds Procedural Generation of Infinite Terrain from Real-World Evelation Data]. Их особенность заключается в том, что значение каждого пикселя можно определить в любой момент времени без необходимости хранения всей карты высот в памяти. *Шум Перлина* и *simplex noise* кроме значения параметров генерации хранят лишь таблицы перестановок и градиентов. Считается, что данные алгоритмы позволяют генерировать карты высот хорошего качества (без артефактов), а также одним из их основных достоинств является «контролируемость» генерации, однако по скорости они уступают фрактальным алгоритмам *diamond-square* и *midpoint displacement* [2011 – [С] Procedurally Generating Terrain]. *Simplex noise* был разработан в качестве замены *шума Перлина* для уменьшения сложности вычислений и потребления ресурсов компьютера на задачах большой размерности. Он отличается сложностью для понимания, реализации и отладки, а также имеет намного меньше готовых реализаций [2011 – [С] Procedurally Generating Terrain].

В данной категории алгоритмов наиболее часто используемым является шум Перлина [Perlin, Ken (July 1985). "An Image Synthesizer". SIGGRAPH Comput. Graph. 19 (0097–8930): 287–296. doi:10.1145/325165.325247. Retrieved 9 February 2016.]. Он включает в себя три шага: задание сетки случайных векторов градиентов, вычисление скалярного произведения каждого градиента на вектор расстояния до текущей точки и интерполяция между полученными значениями.

Анализ литературы и программных реализаций рассмотренных алгоритмов показал, что для удовлетворения заявленных для разрабатываемого ПО требований лучше всего подходит группа шумовых алгоритмов, в частности, шум Перлина, который до сих пор сохраняет свою актуальность [2016 – [Th.] Procedural Terrain Generation Using Ray Marching] [2017 – [Th.] Terrain synthesis using noise] [2017 – [С] Генерация карт высот с использованием шума Перлина для построения ландшафтов], а также в некоторых случаях целесообразно использовать фрактальный алгоритм diamond-square.

## 2. Выбор среды разработки 3D-приложений

Генератор рельефа входит в подсистему моделирования подводной среды, которая, в свою очередь, является частью имитационного моделирующего комплекса (ИМК) по отработке одиночных и групповых алгоритмов интеллектуального управления автономными необитаемыми подводными роботами. Необходимость использования стороннего программного обеспечения вместо разработки собственной платформы по отображению графики обусловлена несколькими требованиями. Основное из них заключается в значительном сокращении временных и человеческих затрат. Использование стороннего ПО позволяет не вникать в вопросы текстурирования и написания шейдеров и предоставляет возможность сфокусироваться на основной задаче – разработке ИМК. Игровые механизмы (engines) являются таким типом платформ, которые удовлетворяют заданным требованиям и подходят для выполнения задачи по разработке алгоритмических и программных средств генерации рельефа и подводной среды в целом.

В настоящее время основными конкурентами среди игровых движков являются Unity, Unreal Engine 4 и CryENGINE.

* Unreal Engine 4 зарекомендовал себя как движок с возможностью создания реалистичного изображения, использования динамического освещения и около миллиона частиц в одной сцене, а наличие Blueprint Visual Scripting позволяет создавать небольшие проекты без особых навыков программирования. Основной язык программирования – C++.
* Unity поддерживает языки программирования UnityScript (похожий на JavaScript) и C#. C# обеспечивает легкий переход с Java, в отличие от перехода на C++ в UnrealEngine. Unity является кроссплатформенным движком, который поддерживает 25 платформ, что почти в два раза больше, чем у двух его основных конкурентов вместе взятых. Кроме того, Unity поддерживает различные форматы файлов 3D-приложений по работе с графикой (анимации, модели и так далее). Магазин Unity – Asset Store – предоставляет доступ к более чем 15000 бесплатных и платных 3D-моделей, анимаций, меню, расширений редактора, материалов, скриптов, шейдеров и т.д. Активное развитое сообщество позволяет получить помощь в краткие сроки. Расширенный редактор рельефа делает «рисование» рельефа очень удобным.
* Графические возможности CryENGINE лучше, чем у Unity, и находятся на уровне близком к Unreal Engine 4, однако для использования данной платформы необходима платная подписка и порог освоения среды является самым высоким из предоставленных движков.

Исходя из анализа перечисленных особенностей выбран Unity как подходящий вариант для выполнения поставленных в рамках данной работы. Объектно-ориентированный язык C# и компонентный подход к программированию в Unity обеспечивают модульность и удобство работы в команде.

Для работы с ландшафтами в Unity существует трехмерный игровой объект (GameObject), который называется Terrain. При его добавлении на сцену проекта создается соответствующий ассет (asset), представляющий собой плоскую поверхность, открытую для редактирования. Рельеф формируется с помощью карты высот и карты прозрачности. Первая задает высоты от 0 до 1 на регулярной квадратной сетке, вторая – описывает, какая текстура видна в данной точке карты (прозрачность составляет 0%), а какая – не видна (прозрачность – 100%). Кроме этого, задаются некоторые другие настройки рельефа, такие как его размер, связанные текстуры, разрешение карты высот, качество отображения рельефа в зависимости от удаленности камеры от него и так далее. Разрешение карты высот, согласно рекомендации из официальной документации, должно составлять 2n+1 единиц, что является удобным при работе с некоторыми видами алгоритмов (например, diamond-square). Таким образом, среда разработки приложений Unity предоставляет обширные возможности по созданию качественных рельефов.

## 3. Архитектура приложения

Генератор рельефа состоит из компонентов трех типов: обработчики, контроллеры и объекты. Обработчики выполняют вычислительную работу, контроллеры отвечают за состояние приложения, а объекты являются контейнерами для хранения информации.

Подсистема моделирования ландшафта включает в себя:

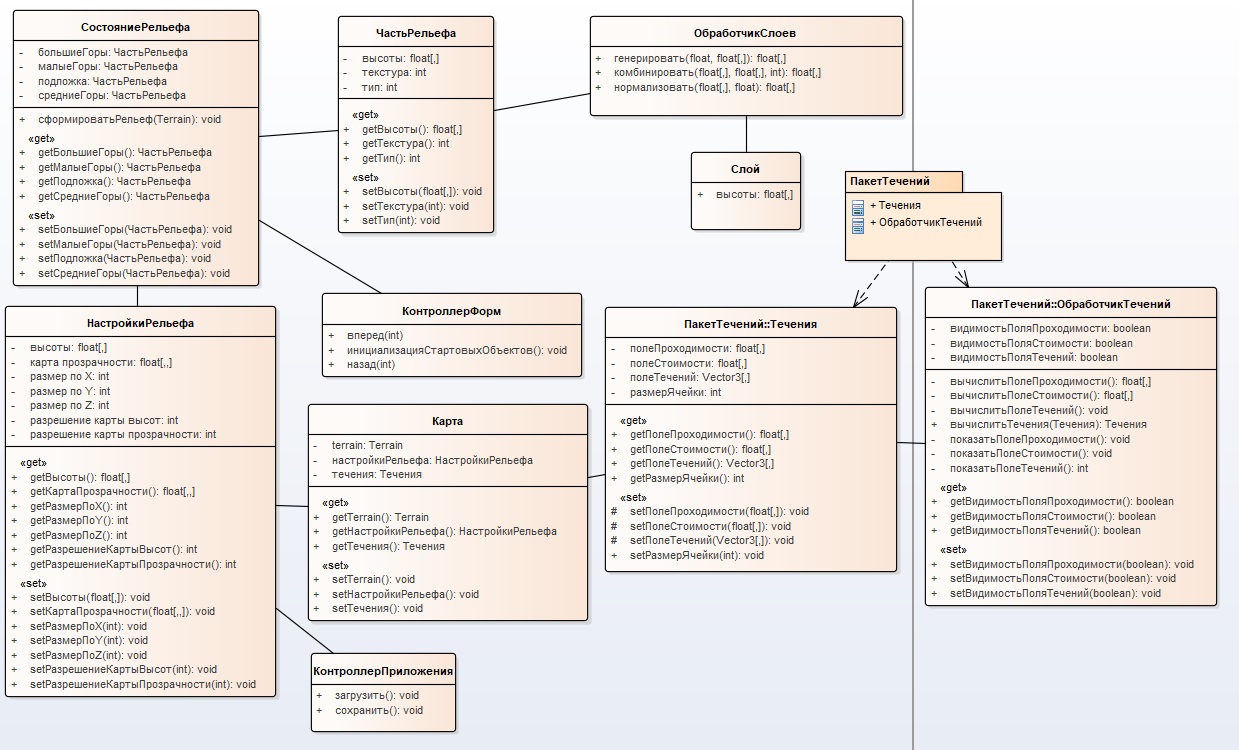
* *Обработчик* *слоев*.
* *Контроллер* *форм*, который переключает формы и сохраняет их состояние во время взаимодействия с пользователем.
* *Контроллер приложения*, который отвечает за сохранение и загрузку данных рельефа или карты.
* Объекты *Слой*, *Часть рельефа*, *Состояние рельефа*, *Настройки рельефа* и *Карта* хранят соответствующую информацию.

Рассмотрим подробнее внутреннюю логику взаимодействия компонентов приложения в процессе создания рельефа (рис. Х):

1. С помощью *Обработчика слоев* генерируется необходимое количество объектов типа *Слой*, которые представляют собой двумерные массивы высот.
2. На полученных объектах типа *Слой* в необходимой последовательности применяются методы *генерировать*, *комбинировать* и *нормализовать в диапазоне*. Так формируется объект *Часть* *рельефа*.
3. Заполняются поля типа *Часть рельефа* объекта *Состояние рельефа*: *Подложка*, *Малые* *горы*, *Средние* *горы, Большие* *горы*. Подложка – это песчаное или глинистое основание рельефа с разницей между максимальной и минимальной высотами составляющей от 10 до 120 метров (в среде Unity единица измерения unit соответствует одному метру). Малыми горами названы каменистые образования 1–30 метров в высоту. Средние горы обладают размерами от 30 до 100 метров, а большие – свыше 100 метров и не превышают максимальную высоту в настройках объекта Terrain.
4. С помощью метода *сформировать рельеф* объекта *Состояние рельефа* по заданным частям рельефа формируется объект *Настройки рельефа* и корректируются данные объекта Terrain.

Отметим, что для пользователя генерация рельефа представляется в виде четырех шагов, что соответствует логике генерации, отраженной в третьем пункте (создание *подложки*, *малых*, *средних* и *больших* *гор*).

Генератор рельефа используется в качестве модуля имитационного моделирующего комплекса для отработки одиночных и групповых интеллектуальных алгоритмов управления АНПА. Это означает, что помимо рельефа на карте виртуального полигона может храниться и другая информация о состоянии виртуальной среды, например, расположение объектов естественного и искусственного происхождения на дне водоема, подводные течения, уровень загрязненности и другое. Подходящим решением является использование объекта *Карта* для сохранения данных о настройках рельефа и объекте Terrain.

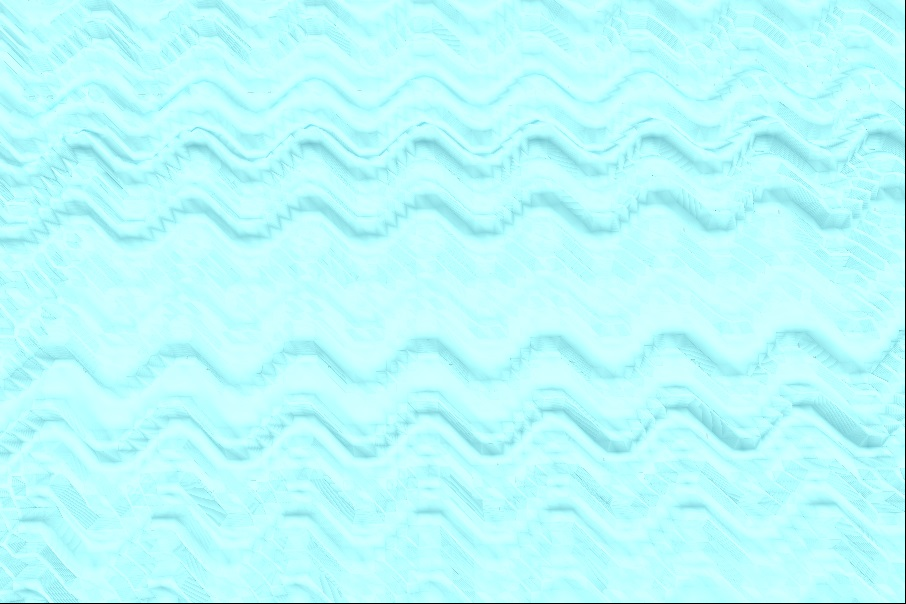


(на картинке должен быть еще объект terrain)

## 4. Алгоритмы и подходы по генерации рельефа

За используемые в процессе моделирования алгоритмы генерации рельефа отвечает обработчик слоев (в частности, его методы *генерировать* и *комбинировать*), который формирует *Части рельефа*, и метод *сформировать* *рельеф* объекта *Состояние* *рельефа*. Рассмотрим детальнее используемые алгоритмы и подходы.

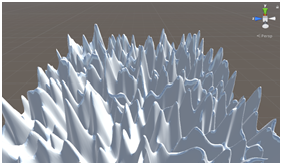
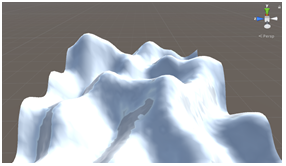
**Генерация слоев с помощью шума Перлина.** Как было сказано ранее, шум Перлина часто и весьма успешно используется при создании рельефа. В данной работе шум Перлина является основным алгоритмом, для которого разрабатывались способы применения, подходящие в рамках конкретной задачи. Результаты работы шума Перлина можно разнообразить несколькими способами. Предлагается использовать различные масштабы шума, изменение пропорций сетки путем равномерного или неравномерного растяжения или сжатия по одной из осей, задание зависимости между координатами горизонтальной плоскости. В качестве примера можно привести зависимость, выраженную в виде y = sin(x) (рис. 3). Данные изменения затрагивают генерацию одного слоя (метод *генерировать* *Обработчика* *слоев*). Комбинировать слои можно также несколькими способами по аналогии с использованием умножения двух карт высот на фрактальных ландшафтах [2006 – [С] Algorithms for Generating Fractal Landscapes]. В рамках данной работы реализованы алгоритмы сложения и умножения слоев, а также предложены алгоритмы наложения по высоте и по смежным зонам.



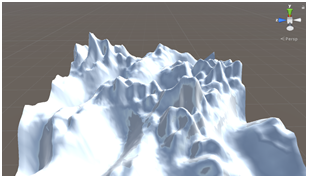
**Рисунок 3** – Добавление синусоидальной зависимости между координатами горизонтальной плоскости (вид на трехмерную поверхность сверху).

Рассмотрим подробнее правило наложения по высоте h двух слоев S1 и S2 масштабов scale1 и scale2 соответственно (рис. Х):

1. Создание слоев S1 и S2 масштабов scale1 и scale2.
2. Отсечение от S1 высот [h, 1] и их нормализация. Получаем временный слой Temp1.
3. Отсечение от S2 областей, где для тех же координат (x, z) слоя S1 высоты менее h. Получаем временный слой Temp2. Проводим его линейную нормализацию.
4. На данном этапе в наличии два промежуточных слоя. Необходимо наложить на первоначальный слой S1 начиная с высоты h слой Temp2, но избежать появления «швов» – видимых стыков. Для этого перемножим слои Temp1 и Temp2.
5. Наложим полученный на 4 шаге слой на ландшафт S1, ограниченный высотами [0, h]. Нормализуем результат.

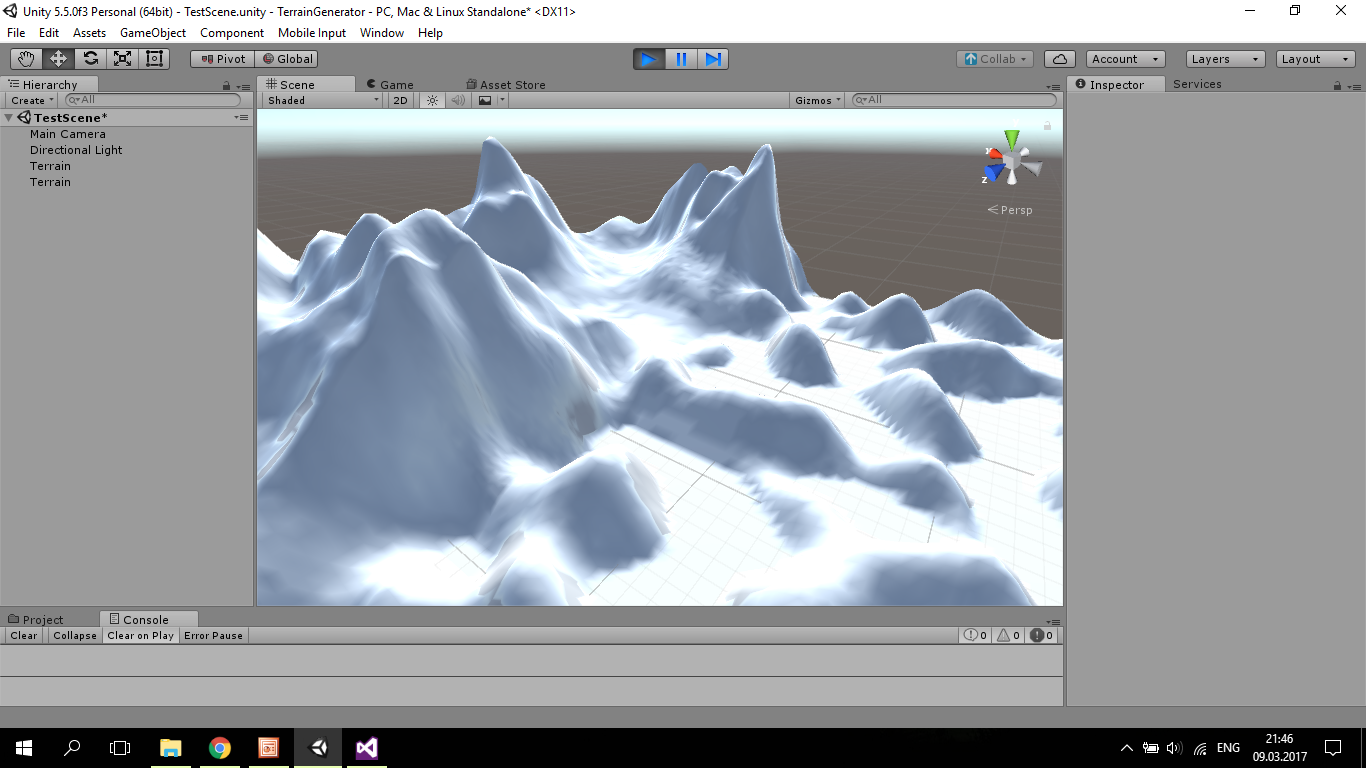
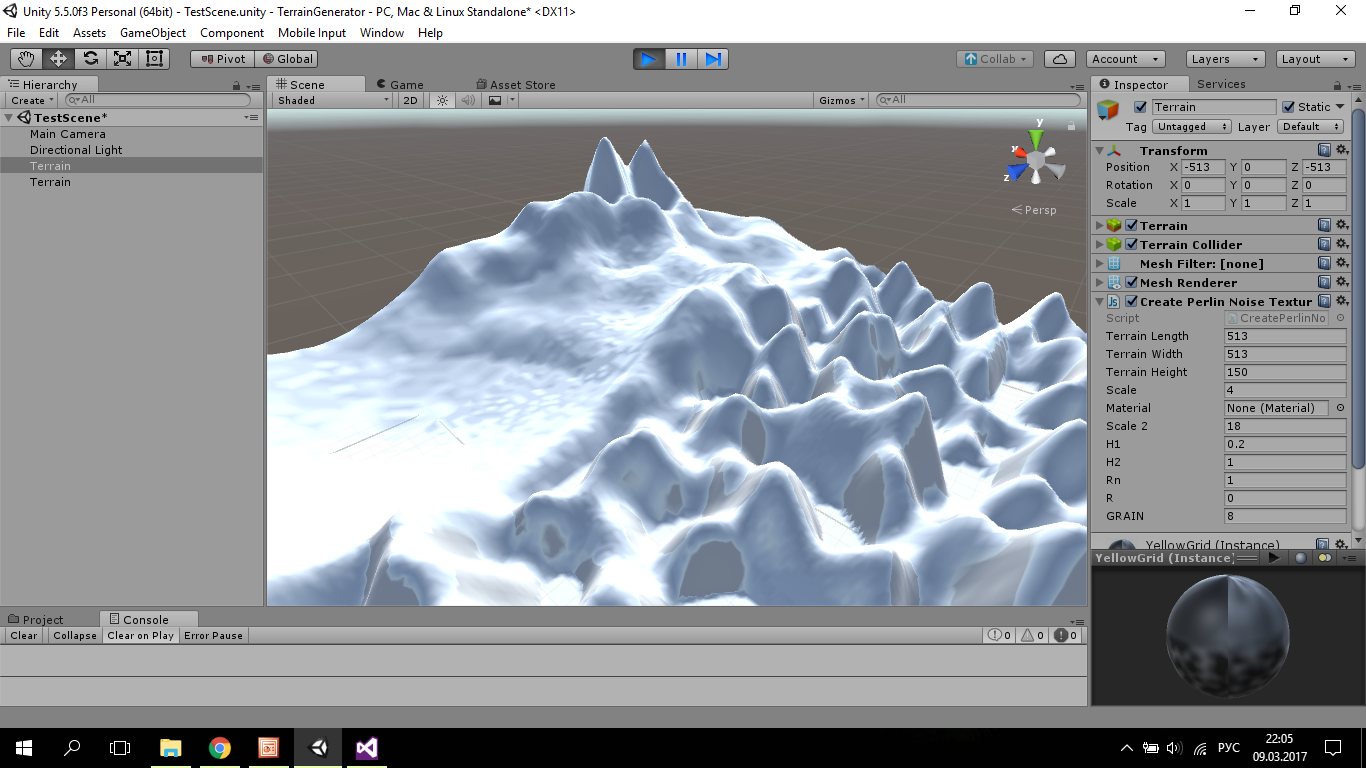


**Рисунок 4 –** Исходные слои S1 и S2 масштабами scale1 и scale2 соответственно.



**Рисунок 8** – Результат наложения произведения промежуточных слоев на исходный слой S1 на требуемой высоте h.

Похожим образом реализовано и наложение по смежным зонам (рис. Х), которое можно использовать не только для комбинирования слоев, но и для формирования рельефа из его частей.

**Рисунок Х** – Результат наложения нескольких слоев по смежным зонам. Оба скриншота иллюстрируют ситуацию наложения по линии: по одну сторону от нее рельеф гористый с острыми пиками, по другую – холмистый с различной степенью интенсивности.

**Соединение частей рельефа** происходит следующим образом. После генерации основания рельефа генерируются большие горы. Далее все высоты больших гор уменьшаются на коэффициент такой величины, чтобы нижняя граница высот гор была ниже некоторых (по выбору) высот песка. Этот коэффициент также необходимо учитывать при задании высоты больших гор. Подобным образом на рельефе размешаются средние и малые каменистые образования. Далее рельеф процедурно текстурируется и при необходимости размещается в водной «чаше» (рис. 9), поверхность которой имеет следующее представление:

C:\Users\enigm\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image001.png,

где x и z – координаты точки в горизонтальной плоскости,

a и b – коэффициенты сжатия по осям,

h – высота рельефа в точке (x, z).

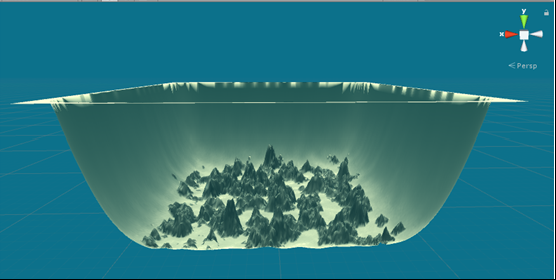
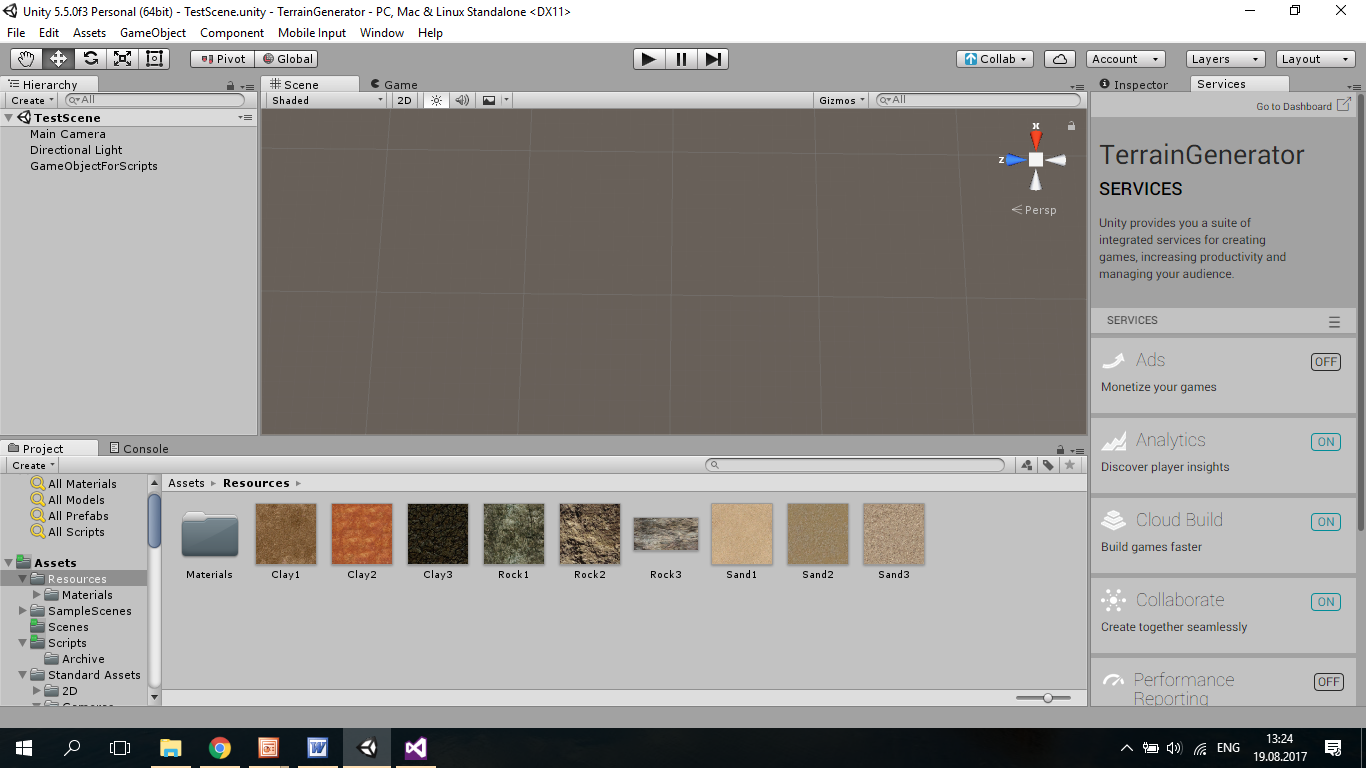
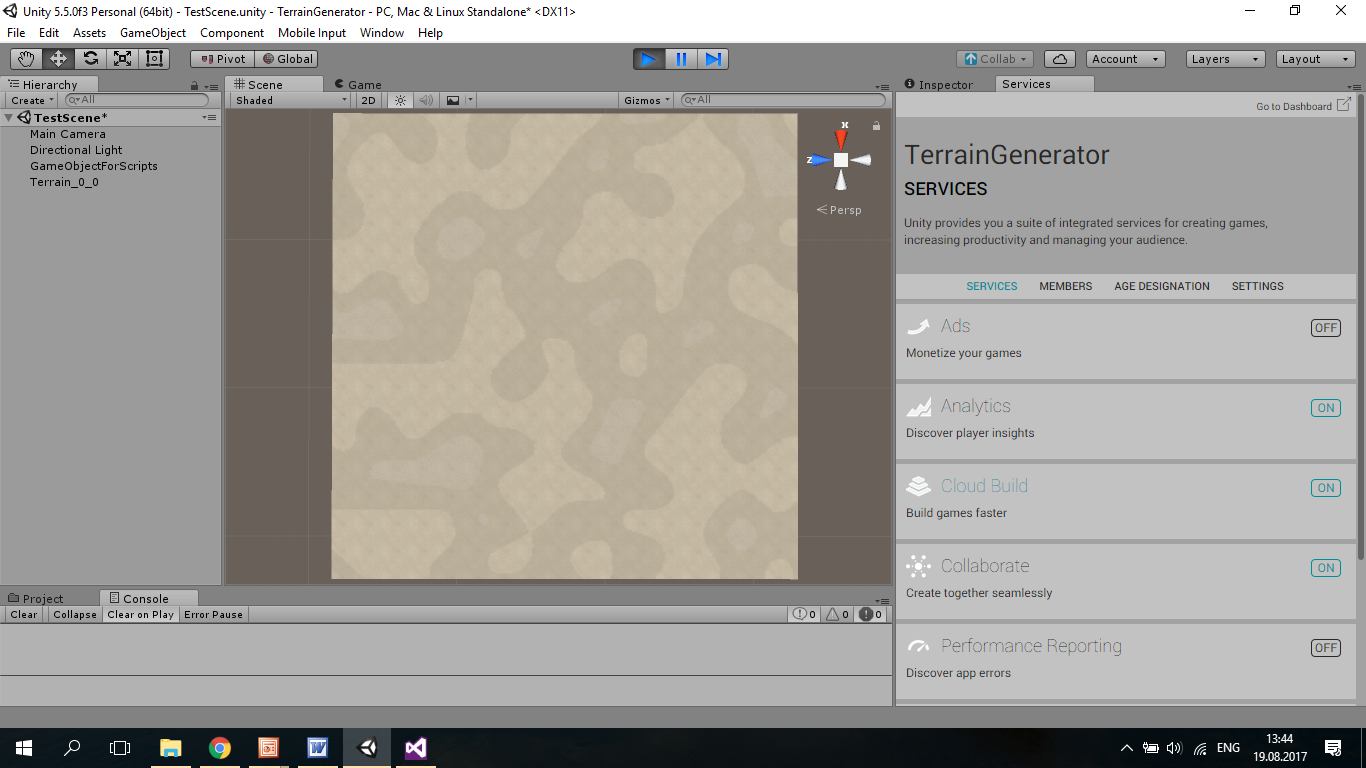
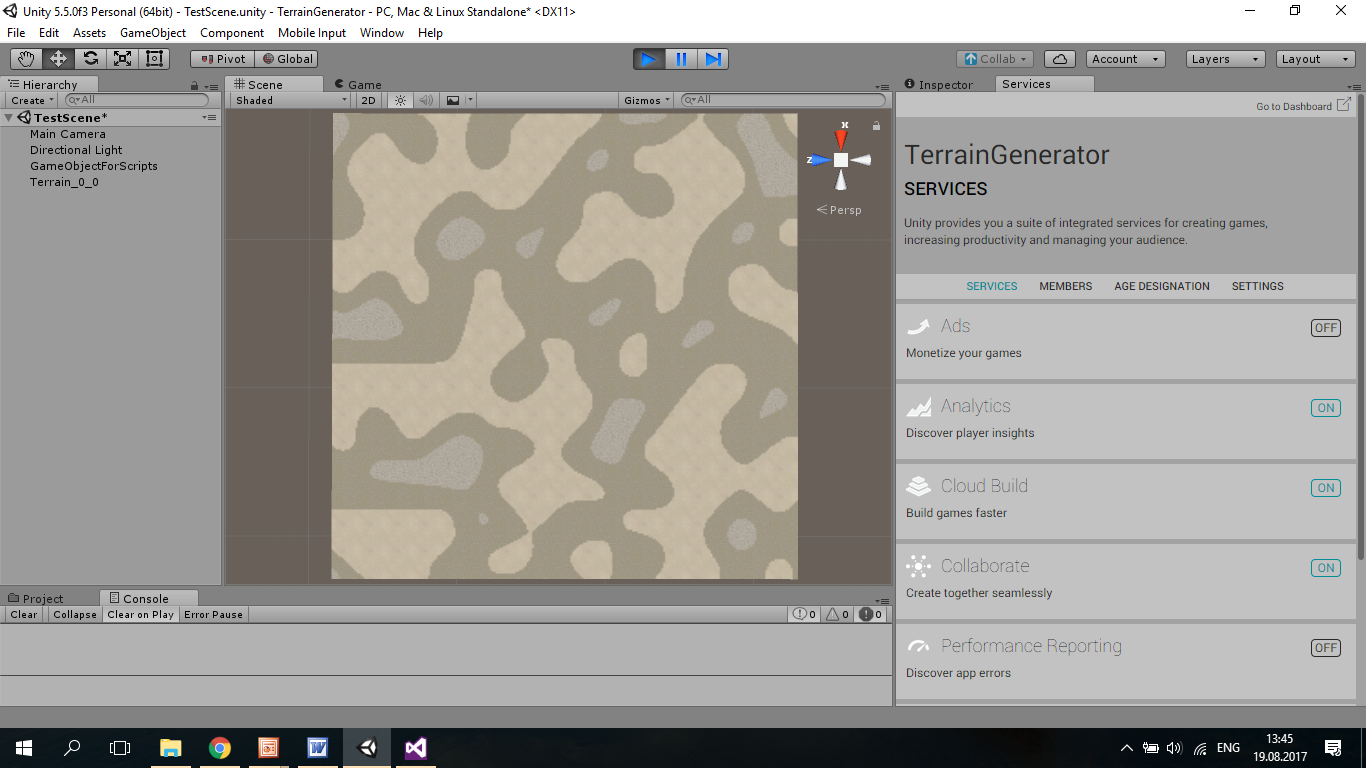


Рисунок Х – Текстурирование рельефа и размещение в водной «чаше».

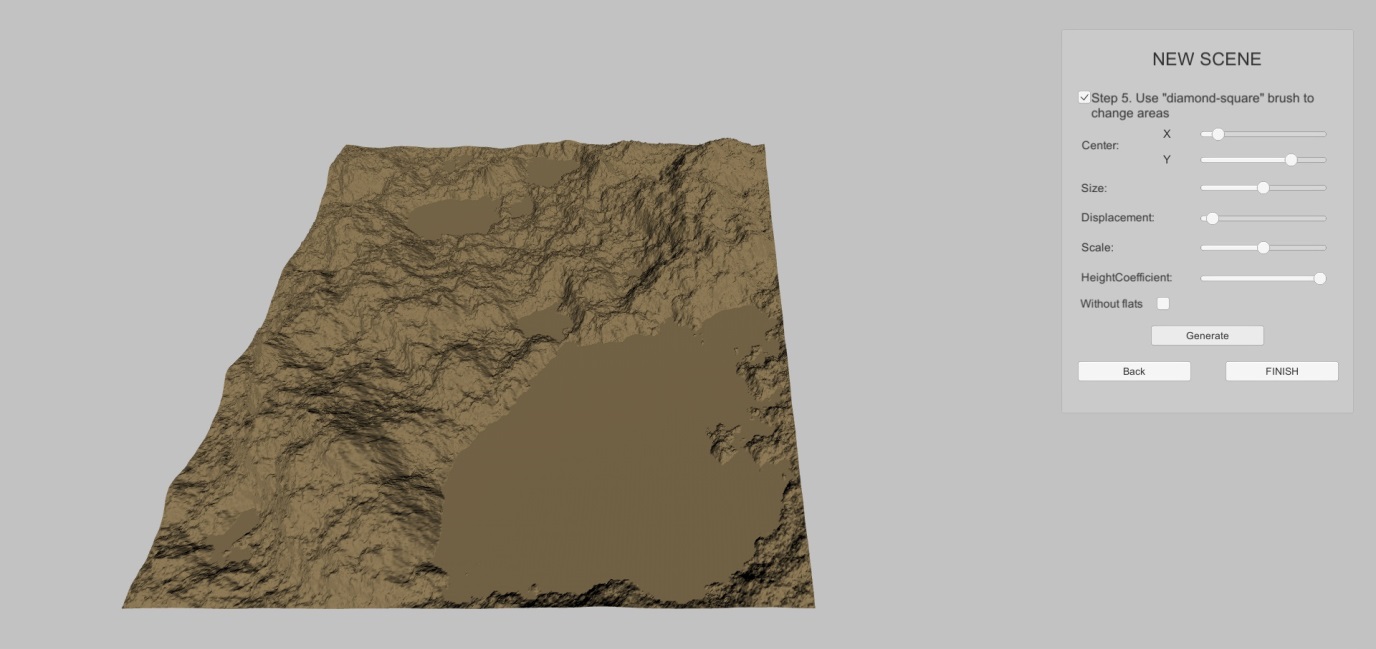
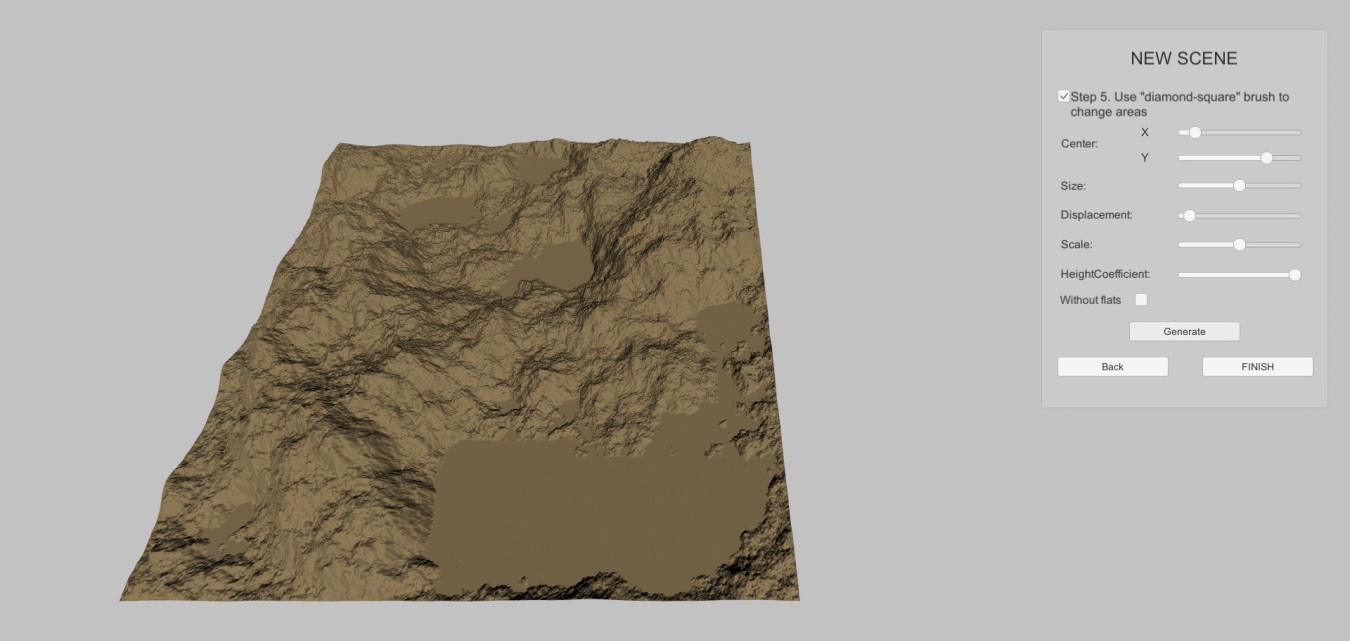
**Процедурное текстурирование** (рис. Х) в некоторых случаях позволяет внести разнообразие во внешний вид рельефа, например, добавить вариативности в цвет подложки. Прозрачность текстур можно задать как согласно рельефу, опираясь на его высоту и крутизну, так и случайным образом, в зависимости от целей.



**Рисунок Х** – Текстурирование с помощью шума Перлина и изменения значений карты прозрачности.

В данной работе в качестве еще одного метода генерации слоев предлагается **модифицировать алгоритм diamond-square** [] так, чтобы без изменения скорости выполнения сделать его более контролируемым. Для этого зафиксируем заранее или каким-либо образом сформируем значения высот на границах зоны карты высот. Помимо стандартных параметров, сделаем доступным для пользователя изменение центра зоны генерации и ее размера. В результате можно заново генерировать отдельные участки рельефа с другими или теми же самыми параметрами (рис. Х), не лишаясь при этом случайности.

**Рисунок Х** – Создание области с помощью модифицированного алгоритма diamond-square и повторная генерация нескольких зон.

**Сохранение и загрузка рельефа** осуществляется путем создания информационного «контейнера» с данными о *настройках рельефа* и дальнейшей сериализации объекта. Восстановление объекта Terrain происходит путем задания ему загруженных из памяти настроек. Все эти действия осуществляет контроллер приложения.

При переключении между шагами в редакторе рельефа сохраняется состояние рельефа. Формируется последовательность состояний, которая позволяет пользователю возвращаться назад, осматривая результаты выполнения предыдущих шагов и вновь двигаться вперед, подгружая уже совершенные ранее шаги из памяти, а также заменять на одном и том же шаге полученную часть рельефа на заново сгенерированную.

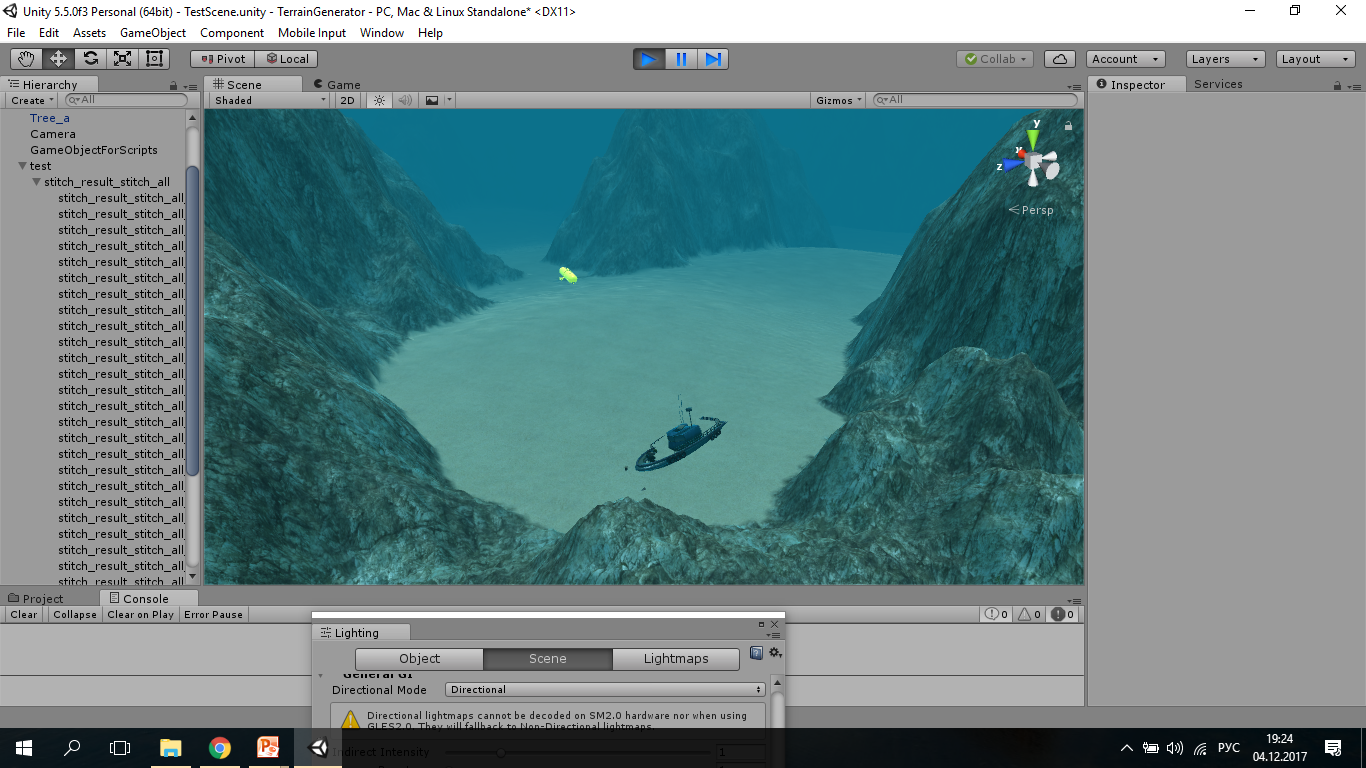
**Интерфейс пользователя.** Пользователю доступны следующие формы:

1. Начальное меню с возможностью создать рельеф, загрузить сохраненный или выйти из редактора рельефа.
2. Меню настроек рельефа, где задается максимальная высота рельефа и размеры его основания.
3. Шаг 1 – создание основы рельефа. Здесь осуществляется выбор текстуры, максимальной высоты холмов, частоты основного шума Перлина, который определяет, насколько крупными относительно общей площади будут холмы, и детализация карты, от которой зависит количество используемых слоев рельефа – то есть его внешний вид и наличие подробностей.
4. Шаг 2–4 содержит те же настройки для больших, средних и малых гор соответственно, а также еще один параметр – зернистость – который отвечает за то, насколько «гладкой» или «шероховатой» будет поверхность.
5. Меню для использования модифицированного алгоритма diamond-square с параметрами, отвечающими за координаты центра зоны генерации, ее размер, смещение точки, масштаб и флажок, отвечающий за возможность отключить генерацию равнин.



Рисунок Х – Пример интерфейса пользователя с рельефом и меню создания больших гор.

На полученном рельефе в режиме редактора сцены или программно **размещаются объекты**, а так же применяются **визуальные эффекты** для достижения большей реалистичности изображения.



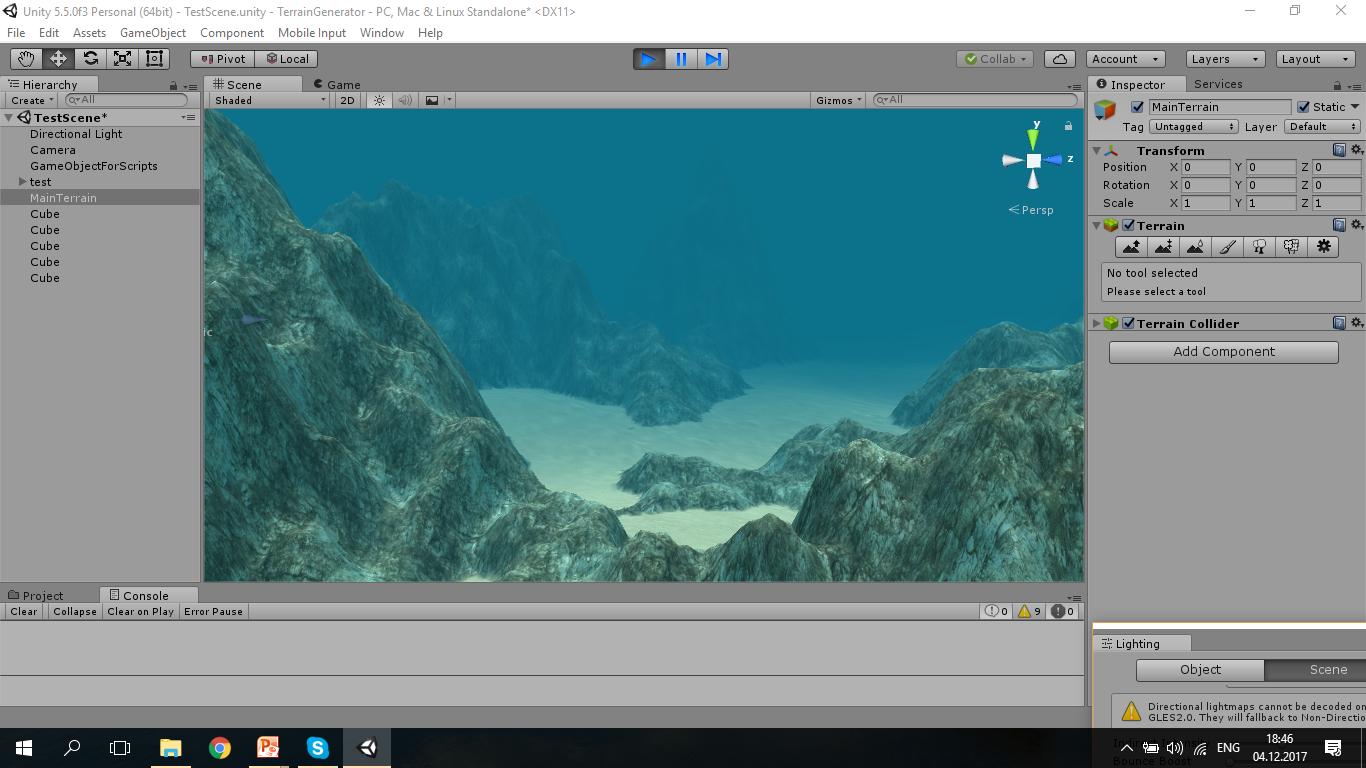


Рисунок Х – Иллюстрация размещения объектов и применения эффекта тумана в Unity.

Нужно добавить обсуждение результатов – привести какие-то сравнения (времени генерации, оценки «реалистичности», на сколько подходит к решению основных задач)…. Можно приводить результаты опросов пользователей, их замечания.

## Заключение

Моделирование подводного рельефа с предложенной организацией взаимодействия объектов, обработчиков и контроллеров позволяет использовать различные алгоритмы генерации для достижения разнообразного рельефа дна. Разработанные методики применения и комбинации различных слоев, заполненных шумом Перлина, повышают реалистичность конечного ландшафта. Реализованный модуль генерации рельефа используется в качестве составной части подсистемы генерации подводной среды разрабатываемого имитационного моделирующего комплекса. Есть нерешенные проблемы?

Дальнейшая работа над генерацией рельефа включает в себя использование алгоритма diamond-square в предложенной архитектуре в качестве генератора слоев, разработка соответствующих данному алгоритму методов комбинации слоев, использование некоторых значений шума Перлина на сетке для генерации фрактального ландшафта, обеспечение возможности более гибкой и настраиваемой генерации рельефа по зонам. Также в дальнейшую работу входит развитие ИМК: разработка модуля генерации течения, внедрение и тестирование одиночных и групповых алгоритмов интеллектуального управления АНПА.

## Список используемой литературы