МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка алгоритма генерации псевдослучайных текстур на основе шума Перлина

Обучающийся А.В. Пеков
(Инициалы Фамилия) (личная подпись)

Руководитель М.А. Тренина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.п.н., доцент С.А. Гудкова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Тема данной выпускной квалификационной работы: «Разработка алгоритма генерации псевдослучайных текстур на основе шума Перлина».

Целью работы является разработка и реализация метода генерации псевдослучайных текстур.

Данная выпускная квалификационная работа состоит из: введения, трёх основных глав, заключения, а также списка используемой литературы и задействованных источников.

Введение раскрывает актуальность темы, область применения, поставленную цель и задачи для реализации алгоритма генерации псевдослучайных текстур.

Первая глава содержит обзор предметной области, сравнительный анализ существующих подходов для генерации текстур и описание алгоритма шума Перлина.

Вторая глава содержит определение функциональных требований и разработку концептуальной модели системы.

Третья глава представляет собой программную реализацию и тестирование системы генерации псевдослучайных текстур с графическим интерфейсом пользователя.

Заключение содержит выводы, которые были сделаны в результате проведенной работы.

Результатом работы является программа процесса генерации псевдослучайных текстур для создания различных видов ландшафта.

В работе было использовано 21 рисунка и 25 ссылок на внешние ресурсы. Общий объем выпускной квалификационной работы 42 страницы.

Abstract

The topic of this graduate qualification work is "Development of an algorithm for generating pseudorandom textures based on Perlin noise".

The aim of the work is to develop and implement a method for generating pseudorandom textures.

This graduate qualification work consists of: introduction, three main chapters, conclusion, as well as a list of literature and sources used.

The introduction reveals the relevance of the topic, the field of application, the set goal and objectives for implementing an algorithm for generating pseudorandom textures.

The first chapter contains an overview of the subject area, a comparative analysis of existing approaches for texture generation and a description of the Perlin noise algorithm.

The second chapter contains the definition of functional requirements and the development of the conceptual model of the system.

The third chapter is a program implementation and testing of the pseudorandom texture generation system with a graphical user interface.

The conclusion contains the conclusions that were made as a result of the work.

The result of the work is a program of pseudorandom texture generation process to create different types of landscape.

21 pictures and 25 references to external resources were used in the work. The total volume of the final qualification work is 42 pages.

Оглавление

Введение

В современном мире компьютерная графика имеет большую область применения в играх, медицине, на производстве и другие. При моделировании различных объектов и поверхностей происходит наложение текстур.

Главной проблемой генерации псевдослучайных текстур является нахождение алгоритма, способного создавать параметризированные текстуры, которые будут способны имитировать реальные объекты и поверхности.

Одной из основных целей является задача определения подхода для генерации псевдослучайных текстур, создание которых можно будет контролировать, задавая параметры их генерации, таким образом, сгенерированные текстуры будут приближены к реальности.

Одним из таких методов генерации псевдослучайных текстур является алгоритм шума Перлина.

Актуальность бакалаврской работы заключается в применении алгоритма шума Перлина для генерации текстур. Такой подход позволяет создавать более реалистичные текстуры различных объектов и ландшафтов. Кроме того, тема является актуальной с точки зрения широкого применения во многих сферах жизнедеятельности человека.

Целью данной работы является разработка и реализация алгоритма генерации псевдослучайных текстур на основе шума Перлина.

На основе этого шума будет реализован программный модуль для создания параметризированных текстур.

Конечным этапом демонстрации работы алгоритма генерации псевдослучайных текстур на основе шума Перлина будет наложение созданной текстуры на поверхность. Таким образом будет создаваться разнообразный ландшафт. Разнообразность создаваемого ландшафта будет достигаться путём изменения различных входных параметром алгоритма шума Перлина.

Для достижения выше поставленной цели, необходимо решить следующие задачи:

- проанализировать существующие подходы и методы генерации псевдослучайных текстур;
- описать алгоритм генерации параметризированных текстур;
- разработать требования к программе и пользовательскому интерфейсу;
- реализовать программу для генерации текстур;
- провести тестирование программы;
- сделать вывод о применимости получившегося программного продукта.

Первая глава работы посвящена изучению предметной области, сравнительному анализу различных актуальных и современных подходов генерации псевдослучайных текстур в области компьютерного моделирования и описанию метода генерации параметризированных текстур на основе алгоритма шума Перлина.

Во второй главе описываются требования к разрабатываемой системе и пользовательское взаимодействие с ней, также разрабатывается концептуальная модель работы всей системы в целом.

Третья глава посвящена разработке пользовательского интерфейса и программной реализации алгоритма генерации псевдослучайных текстур, тестированию разрабатываемого продукта и анализу полученных результатов.

Результатом работы является анализ, проведённый в области подходов генерации псевдослучайных текстур, программная реализация алгоритма генерации параметризированных текстур и создание программного модуля с пользовательским интерфейсом для создания и наложения сгенерированных текстур на ландшафт.

Глава 1 Математическое описание метода генерации псевдослучайных текстур

1.1 Обзор предметной области

В последние годы использование компьютерной графики быстро растет. Многие привычные для нас вещи создаются с её помощью.

Компьютерная графика (машинная графика) представляет собой способ создания графических изображений и визуальной информации с помощью специальных программ [2]. У компьютерной графики есть различные сферы применения:

- широкое использование в различных отраслях промышленности, например, инженерия, архитектура, медицина и многое другое;
- в сфере развлечений, например, в создании видеоигр, спецэффектов для кино и телевидения [19];
- для визуализации данных, что способствует сделать информацию менее сложной и более доступной [19];
- в сфере образования для реализации интерактивных учебных материалов [2];
- для создания логотипов, рекламных материалов в сфере интернет-маркетинга.

Таким образом, благодаря машинной графики происходит влияние на различные области деятельности.

Существуют различные виды компьютерной графики:

- растровая графика;
- векторная графика;
- 3-D графика;
- компьютерная анимация;
- инфографика.

У каждой из видов есть свои характерные особенности. Например, растровая графика создается с помощью пикселей и подходит для фотографий и изображений с большой детализацией. Векторная графика напротив, основывается на математических формулах и в основном применяется для создания логотипов. 3D-графика создает трехмерные модели, широко используется в видеоиграх, кино и славится своей реалистичностью [10]. Компьютерная анимация также, как и 3D-графика используется в видеоиграх, но с её помощью создают различного вида анимации. Последним видом графики является инфографика, которая комбинирует текст и изображение для визуализации информации.

Выбор подходящего вида графики зависит от конкретных задач и целей создаваемого проекта.

В компьютерной графике все создаваемые модели имеют текстуру, которая позволяет однозначно идентифицировать образ объекта. Следовательно, введем понятие текстуры.

Текстура – это изображение объекта или поверхности приближающее его к реальности. За счёт наложение текстуры можно создавать различные визуальные эффекты, сделать поверхность более шероховатой, а объекты более плавными.

Текстура представляет собой объёмную плоскость, которая накладывается на полигональную сетку.

Полигональная сетка – это совокупность точек или вершин, лежащих в одной плоскости, соединённых ребрами. Простыми словами, многоугольная фигура, лежащая в определённой плоскости. Таким образом, большинство объёмных поверхностей и объектов в компьютерной графике представляют собой много-полигональные фигуры.

Следовательно, для таких фигур необходимо программно-генерировать текстуры, с возможностью влиять на параметры их генерации, например, на их размер, степень сглаженности и т.д. Создаваемые, таким образом, текстуры

будут максимально приближены к реальности и с их помощью можно создать ландшафт.

Ещё одним из главных понятий является ландшафт. Ландшафт представляет собой обозримый участок поверхности, имеющий свои характерные черты. Благодаря полигональной сетке возможно создавание реалистичного ландшафта. Реалистичность ландшафта достигается с помощью ключевых аспектов:

- сглаженность подъёмов и падений рельефа;
- подбор правильной цветовой палитры с учетом времени суток, освещения;
- правильное построение высот для создания необходимой рельефности с учётом окружающей среды.

Благодаря выделенным аспектам, в компьютерной графике можно добиться максимально реалистичного ландшафта. Ландшафт создается с помощью полигональной сетки и текстур, а текстуры в свою очередь создаются с помощью алгоритмов шума.

Алгоритмы шума основаны на различных математических преобразованиях псевдослучайных данных, что позволяет в той или иной степени контролировать и видоизменять созданные псевдослучайные текстуры. Одним из таких подходов является алгоритм шума Перлина.

1.2 Сравнительный анализ существующих подходов для генерации текстур с алгоритмом шума Перлина

На текущий момент существует большое количество алгоритмов создания шума. Каждый из них находит своё применение в различных областях компьютерной графики. У каждого из алгоритмов шума есть свои достоинства и недостатки. Таким образом, необходимо провести сравнительный анализ некоторых из методов генерации псевдослучайных текстур с алгоритмом шума Перлина.

Первым из рассматриваемых алгоритмов шума будет белый шум. Белый шум – это стационарный шум, основанный на равномерно-стохастическом распределении точек на плоскости со случайными высотными значениями. Данный шум является наиболее распространённым и, помимо генерации текстур, имеет широкую сферу применения.

Белый шум имеет ряд достоинств:

- простота реализации;
- сравнительно не высокая нагрузка на вычислительную систему;
- равномерность распределения по всей поверхности текстуры.

Помимо этого, белый шум имеет существенные недостатки:

- слишком высокая степень распространения сплошного шума;
- малое количество контролируемых параметров, что приводит к меньшей степени реалистичности текстуры;
- очень высокая степень сходства результата при одинаковой размерности поверхности текстуры.

Следовательно, белый шум является менее подходящим алгоритмом генерации псевдослучайных текстур, так как основная цель алгоритма генерации является создание наиболее приближённой к реальности текстуре.

Ещё одним рассматриваемым алгоритмом является Diamond-square. Diamond-square – это алгоритм генерации псевдослучайных фрактальных текстур. Данный алгоритм основан на рекурсивном разбиении поверхности на квадраты и ромбы, присваивая их центрам псевдослучайные высотные значения. Этот метод хорошо подходит для генерации карт высот при создании текстур для поверхностей.

У Diamond-square есть ряд достоинств:

- позволяет создать достаточно реалистичные ландшафты;
- имеет ряд модификаций, что позволяет создать наиболее качественные текстуры;
- имеет высокую скорость генерации небольших текстур [1].

У данного метода есть свои недостатки:

– линейная зависимость создаваемых текстур;
– иногда встречаются участки, состоящие либо из «мелких островков», либо из сплошного шума;
– при генерации текстур, для создания граничных точек поверхности, происходят выходы за границу области.

Самым популярным и универсальным алгоритмом генерации псевдослучайных текстур является алгоритм шума Перлина (Perlin noise). Шум Перлина – это градиентный шум, состоящий из набора псевдослучайных единичных векторов (направлений градиента), расположенных в определенных точках пространства и интерполированных функцией сглаживания между этими точками [17]. Для генерации шума Перлина в пространстве необходимо для каждой точки этого пространства вычислить значение шумовой функции, используя направление градиента (или наклон) в указанной точке [13]. Данный метод хорошо подходит для создания как текстур поверхностей, так и для текстур объектов.

Шум Перлина имеет ряд достоинств:

– позволяет создать наиболее реалистичные ландшафты [13];
– имеет большое количество параметров и модификаций, что позволяет создать наиболее качественные текстуры и задавать характер их распространения [17];
– является не линейным алгоритмом генерации псевдослучайных текстур, за счёт функции сглаживания и случайно-направленных вектор-градиентов.

Главными недостатками шума Перлина являются:

– сложность и трудоёмкость алгоритма;
– высокая нагрузка на вычислительную систему;

– иногда встречаются равномерно-распределённые участи текстур, имеющие практически одинаковое значение высотных коэффициентов.

Несмотря на сложность и трудоёмкость алгоритма, шум Перлина является наиболее подходящим метод генерации псевдослучайных текстур. Он подходит для большинства создаваемых текстур, делая их более реалистичными, и имеет большое количество параметров, что позволяет контролировать характер создаваемых текстур.

Таким образом, выбор алгоритма шума Перлина является полностью оправданным и целесообразным, так как позволяет решит все поставленные задачи.

1.3 Описание алгоритма генерации псевдослучайных текстур на основе шума Перлина

Для генерации псевдослучайных текстур на основе шума Перлина используется следующий алгоритм.

Шаг 1. На двумерном пространстве размерностью $N \times N$ наносятся M случайно-расположенных вещественных точек.

Шаг 2. Выбирается точка m_k $(k = 1, 2, \dots, M)$ с вещественными координатами (x, y).

Шаг 3. Для этой точки находится целый координатный узел (i, j) и рассчитывается вектор до этого узла

$$\vec{a}_1(v, u) = (x - i, y - j). \quad (1)$$

В результате строится вектор $\vec{a}_1(v, u)$, который изображен на рисунке 1.

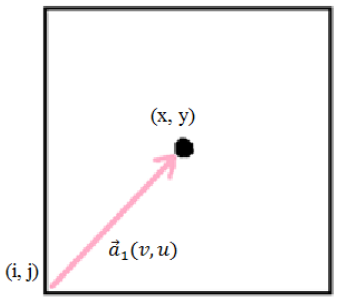

Рисунок 1 – Рассчитанный вектор из координатного узла

Шаг 4. Аналогично вычисляются векторы и до других узлов (рисунок 2).

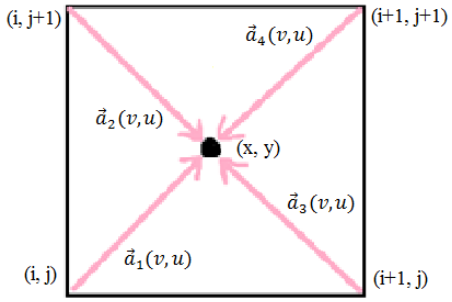

Рисунок 2 – Векторы из оставшихся узлов

Шаг 5. Полученные векторы умножаются на псевдослучайные градиентные векторы

$$\overrightarrow{R_l}(v,u) = (\overrightarrow{G_l}, \overrightarrow{a_l}), \quad (2)$$

где $\overrightarrow{G_l}$ – псевдослучайно-направленный единичный градиент-вектор.

Построение псевдослучайных градиентных векторов изображено на рисунке 3.

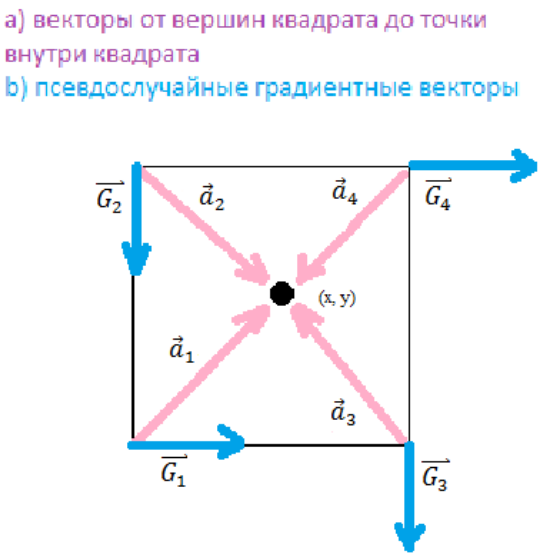

Рисунок 3 – Нахождение градиентов

Шаг 6. Вектор из шага 3 сглаживается степенной функцией

$$fade(x) = 6 * x^5 - 15 * x^4 + 10 * x^3. \quad (3)$$

Шаг 7. Полученные четыре градиента попарно интерполируются по x, используя координату x сглаженного вектора

$$L_{x_i} = a + fade(x) * (b - a), \quad (4)$$

где a, b – значения градиентов $\overrightarrow{R_l}(v, u)$ изменяющихся по оси x;

$fade(y)$ – значение координаты x сглаженного вектора.

Шаг 8. В результате, полученные два значения интерполируются по y, используя координату y сглаженного вектора

$$L = L_{x_1} + fade(y) * (L_{x_2} - L_{x_1}), \quad (5)$$

где $fade(y)$ – значение координаты y сглаженного вектора.

Шаг 9. Затем берется следующая точка m_{k+1} и происходит возврат к шагу 3.

Шаг 10. Таким образом, находятся значения всех M точек на плоскости размерностью $N \times N$. Полученный результат является шумом Перлина или начальной матрицей высот.

Алгоритм шума Перлина имеет ряд модификаций и параметров, которые накладываются на решение после генерации начальной матрицы высот.

Следовательно, для контроля и способности видоизменения результата алгоритма шума Перлина, необходимо воздействовать на алгоритм и его результат следующими переменными: seed, amp, octaves.

Seed – начальное значение псевдослучайного распределения, другими словами коэффициент нормального распределения. Оно влияет на распределение M случайно-расположенных вещественных точек на пространстве размерностью $N \times N$ и направление псевдослучайных единичных градиентных векторов.

Amp – абсолютное максимальное значение пиков. Это значение является положительным и влияет на результат созданных текстур. Если значение данного параметра не велико, то в результате созданные текстуры представляют «слегка волнистую», плавную поверхность, но при больших значениях результатом будет являться «грубая» текстура с множеством пригорков и ям [18].

Octaves – количество итераций расщепления шума. Данное расщепление происходит по алгоритму.

Шаг 1. Задаются начальные значения количества итераций, амплитуда изменения расщепления и шаг регулировки изменения, который будет умножаться на амплитуду.

Шаг 2. Результат алгоритма шума Перлина умножается на амплитуду

$$M_{ij} = M_{ij} * A, \tag{6}$$

где M_{ij} – элемент матрицы высот;

A – амплитуда.

Шаг 3. Амплитуда умножается на шаг регулировки

$$A = A * \Delta\alpha, \tag{7}$$

где $\Delta\alpha$ – шаг регулировки.

Шаг 4. Частота шума удваивается. Другими словами, количество псевдослучайных вещественных точек, которыми задавалась матрица шума Перлина, увеличивается в два раза.

Шаг 5. Значение octaves уменьшается на один. Если значение больше нуля, то осуществляется переход к шагу 2.

Шаг 6. Полученный на последней итерации результат алгоритма шума Перлина умноженный на амплитуду на последней итерации делиться на сумму амплитуд на каждой итерации, что позволяет сохранить начальный размер итоговой матрицы высот.

$$M = \frac{M_K * A_K}{\sum_{k=1}^{K} A_k}, \tag{8}$$

где M – итоговая матрица высот;

M_K – матрица высот на последней итерации;

A_K – значение амплитуды на последней итерации;

A_k – значение амплитуды на k-й итерации.

С помощью всех вышеперечисленных параметров можно воздействовать на создаваемые текстуры меняя их форму, делая их более плавными или гористыми, сплошными или хаотичными. Благодаря этому, генерируемые текстуры будут максимально приближены к реальности, исходя из желаний пользователя.

Выводы к первой главе

В данной главе была рассмотрена разработка модели генерации псевдослучайных текстур на основе алгоритма шума Перлина. Первым этапом были описаны предметная область и основные понятия, такие как компьютерная графика, текстуры, полигональная сетка и ландшафт. Помимо этого, были определены условия для генерации параметризированных текстур.

Затем был проведён сравнительный анализ других подходов генерации текстур с алгоритмом шума Перлина. На основе данного анализа были выделены достоинства и недостатки используемого метода для понимания целесообразности и актуальности его использования.

В конечном итоге, были описаны алгоритм генерации псевдослучайных текстур на основе шума Перлина и ряд его модификаций, с помощью которых появляется возможность влияния на характер создаваемых текстур. Данный алгоритм необходим для дальнейшей программной реализации создания псевдослучайного ландшафта.

Глава 2 Проектирование системы генерации псевдослучайных текстур

2.1 Определение функциональных требований к разрабатываемой системе

Главной целью проектирования системы является создание такой концепции создаваемого программного продукта, которая будет понятной и легко осваиваемой для любого пользователя.

Следовательно, для определения требований к проектируемой системе необходимо учитывать следующие аспекты:

- наличие интуитивно понятного интерфейса [6];
- отсутствие сбоев программы и точность результатов;
- наличие справочного материала (мануала пользователя);
- наличие советов и подсказок для пользователя;
- наличие предупреждений об неверно сделанных действиях, с подсказкой о способе их решения [4].

Исходя из вышеперечисленных требований, целей и задач можно определить функциональные требования к разрабатываемой системе:

- просмотр мануала пользователя [21];
- ввод необходимых параметров для генерации текстуры;
- возможность сгенерировать шум Перлина (матрицу высот);
- возможность сгенерировать ландшафт на основе шума Перлина.

После определения функциональных требований к разрабатываемой системе, необходимо выбрать методологию для дальнейшего проектирования. Для наглядности отлично подойдет методология UML.

UML представляет собой стандартный язык моделирования, который используется для визуализации и проектирования структуры и поведения системы [7]. UML отлично подойдёт для создания различных типов диаграмм,

которые помогут визуализировать работу системы для дальнейшего её проектирования и программной реализации.

2.2 Разработка концептуальной модели системы

Для создания концептуальной модели, необходимо прежде всего определить возможные взаимодействия предполагаемого пользователя с разрабатываемой системой. Наиболее подходящим способом представления этого взаимодействия будет диаграмма вариантов использования. Для создания этой диаграммы необходимо учитывать функциональные требования к системе, созданные ранее. Диаграмма вариантов использования представлена на рисунке 4.

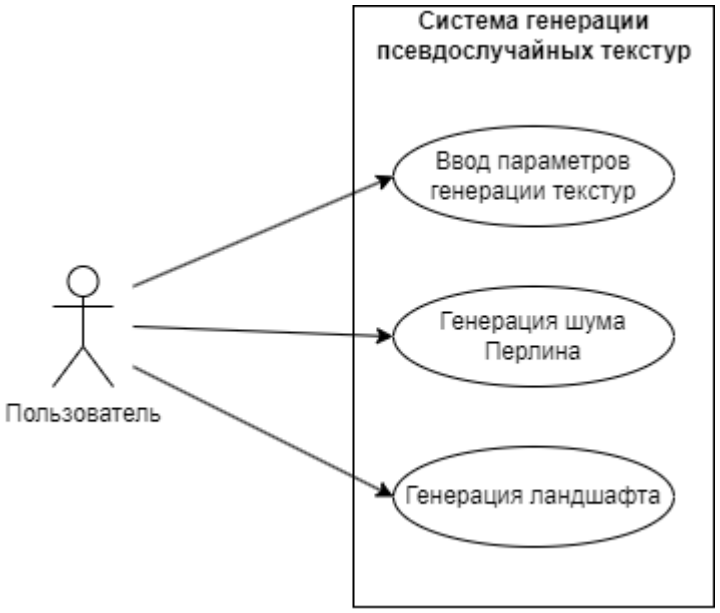

Рисунок 4 – Диаграмма вариантов использования

Исходя из диаграммы вариантов использования становится ясно каким функционалом должна обладать будущая система. Таким образом, можно определить из каких страниц, кнопок и функций будет состоять программный модуль генерации псевдослучайных текстур на основе шума Перлина.

Отобразить этот перечень функционала и помочь пользователю ориентироваться в системе позволяет карту навигации. Данная карта представлена на рисунке 5.

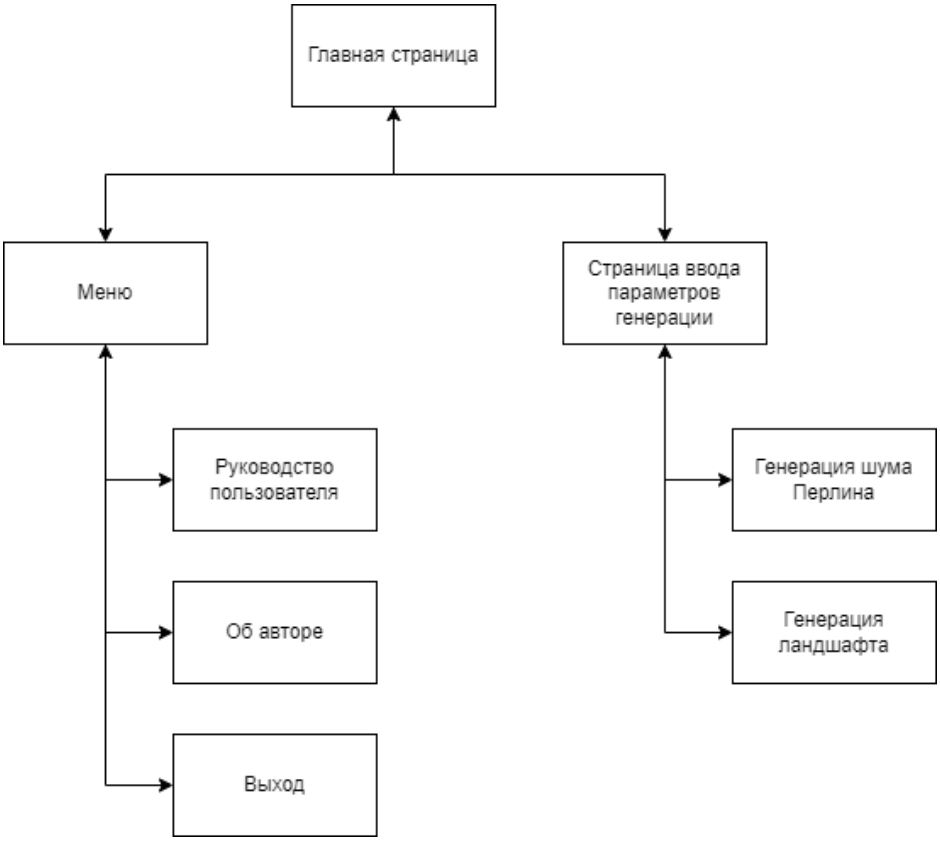

Рисунок 5 – Карта навигации программы генерации псевдослучайных текстур на основе шума Перлина

На основе составленных диаграмм можно перейти к конечному этапу проектирования системы, а именно к созданию концептуальной модели. Эта модель системы является абстрактным представлением структуры, функций и взаимосвязей элементов программы [11]. С помощью этой модели можно описать средства разработки, программное обеспечение, строки состояний пользователя, рабочего пространства и самого меню приложения и взаимосвязь между ними. Концептуальная модель системы генерации псевдослучайных текстур на основе шума Перлина представлена на рисунке 6.

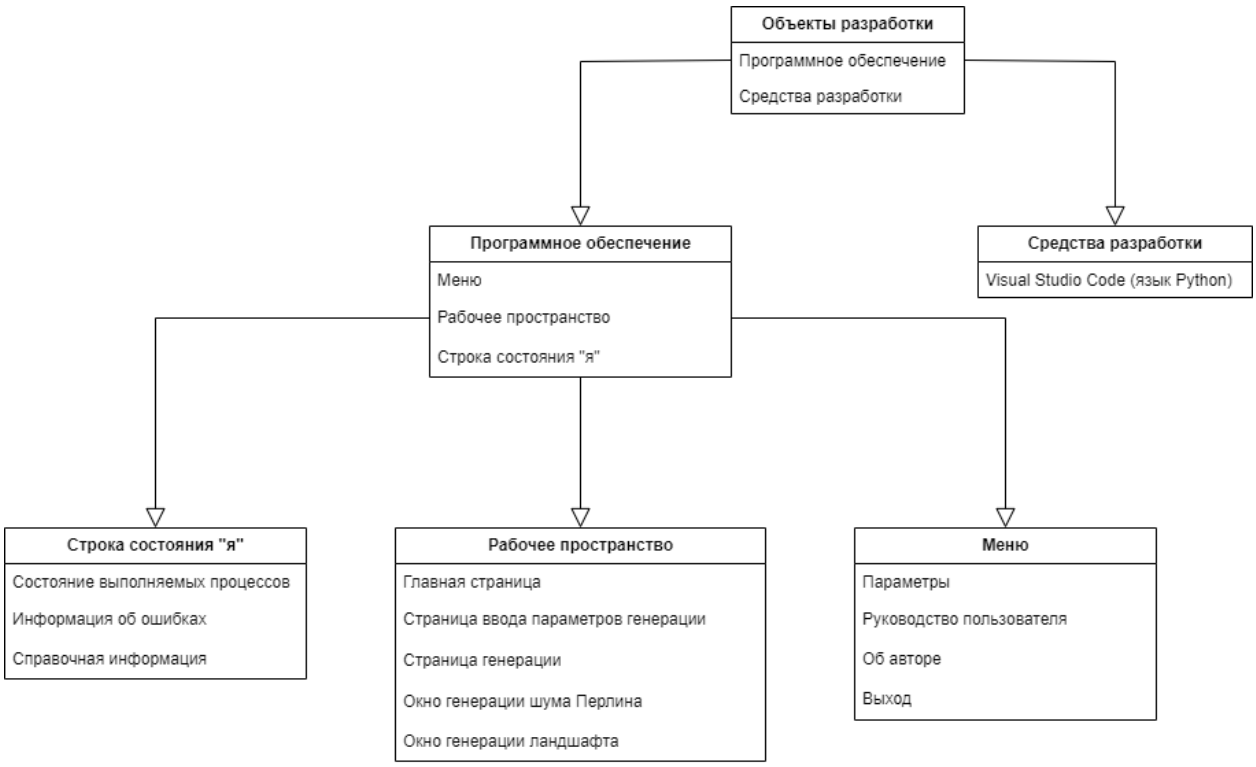

Рисунок 6 – Концептуальная модель системы

На основе спроектированной концептуальной модели можно будет разработать полноценный интерфейс пользователя и создать необходимый функционал системы генерации псевдослучайных текстур.

Выводы к главе 2

В данной главе была спроектирована система генерации текстур.

В первую очередь были определены функциональные требования к разрабатываемой системе с целью создания концепции приложения, которая будет интуитивно понятной и легко осваиваемой для любого пользователя.

Затем была создана диаграмма вариантов использования и карта навигации для определения взаимодействий предполагаемого пользователя с программой. На основе этого была составлена концептуальная модель системы для определения интерфейса, функционала программы и её компонентов.

Глава 3 Программная реализация системы генерации псевдослучайных текстур на основе шума Перлина

3.1 Описание работы системы генерации псевдослучайных текстур

На основе описанных функциональных требований к системе и концептуальной модели становится возможным реализация системы процесса генерации псевдослучайных текстур. Для реализации системы был выбран высокоуровневый язык программирования Python, а средой разработки является текстовый редактор кода Visual Studio Code.

Выбор данного языка программирования обусловлен наличием большого количество современных библиотек, технологий, движков для реализации пользовательского интерфейса и компьютерной графики, для генерации, отображения текстур и ландшафта [8].

Для того, чтобы сгенерировать псевдослучайные текстуры предполагаемому пользователю необходимо будет перейти в меню генерации текстур. В данном меню можно вводить основные параметры для генерации, такие как размерность генерируемой текстуры, значение октав, коэффициент распределения псевдослучайных точек и коэффициент высот сгенерированной матрицы.

На основе введённых параметров будет сгенерирована матрица высот с помощью алгоритма шума Перлина. Затем результат алгоритма можно вывести на экран в виде высотного изображения методом imshow() библиотеки matplotlib. Данная библиотека позволяет изображать данные в виде графиков, диаграмм и рисунков [9].

Следующим этапом является наложение сгенерированной текстуры на ландшафт и его отображение.

Блок-схема алгоритма работы системы генерации псевдослучайных текстур представлена на рисунке 7.

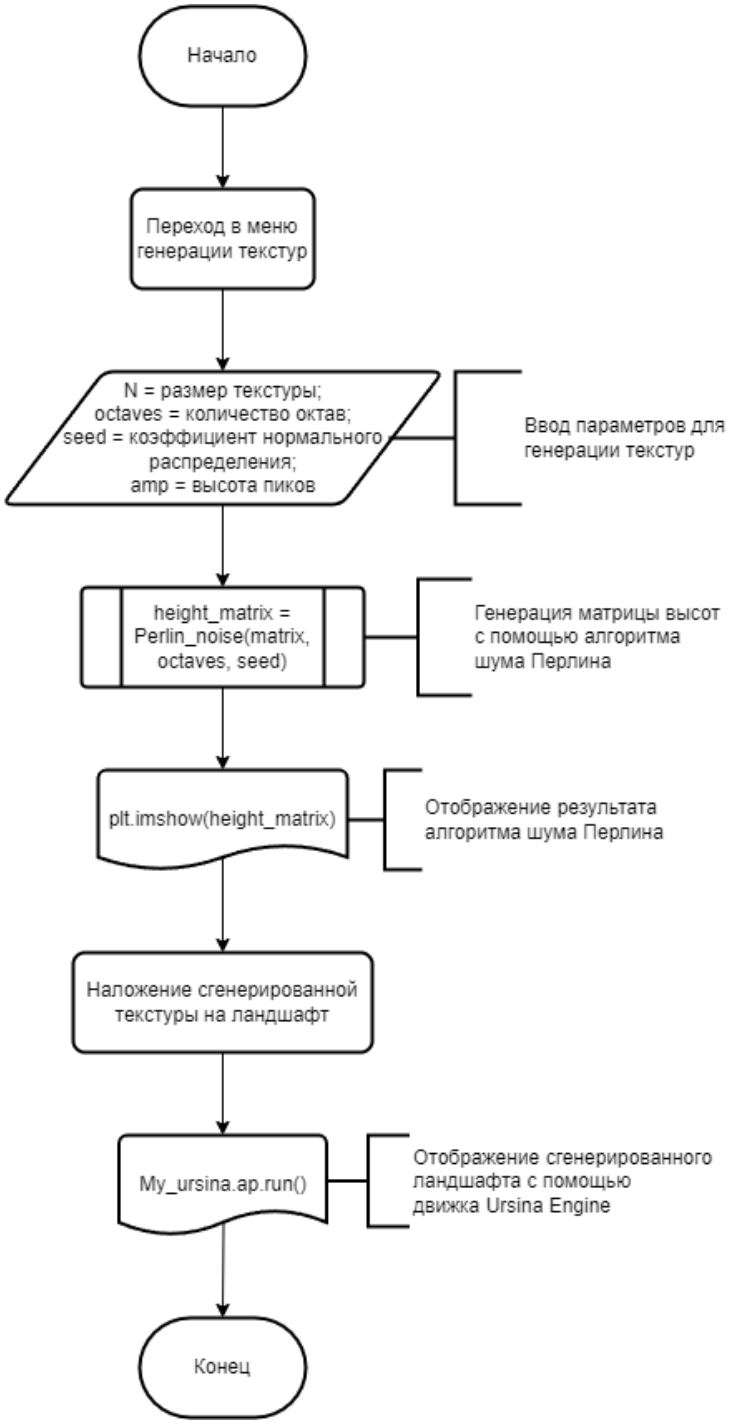

Рисунок 7 – Блок-схема алгоритма работы системы

Данная блок-схема показывает алгоритм работы системы, что в свою очередь поможет правильно реализовать структуру пользовательского интерфейса и программной реализации.

3.2 Разработка и реализация интерфейса приложения

В современном мире большинство программных продуктов имеют графический интерфейс пользователя (GUI). GUI – это система средств для взаимодействия пользователя с программой, представленная в виде графических компонентов экранных форм, таких как окна, значки, меню, кнопки, списки и прочие [12]. Разрабатываемый интерфейс пользователя должен соответствовать следующим требованиям:

- иметь возможность для решения всех задач в соответствии функциональными требованиями;
- быть привлекательным и интуитивно понятным для любого пользователя;
- иметь множество подсказок и предупреждений для обеспечения защиты от ошибок;
- иметь руководство пользователя и (или) справочную информацию.

Для реализации интерфейса будет использоваться библиотека Tkinter. Данная библиотека достаточна проста в использовании и имеет большое множество различных экранных форм и методов взаимодействия с ними [22]. Имеющиеся в библиотеке экранные формы, обладающие собственными встроенными методами, в компьютерной графике имеют название виджеты [16].

Таким образом, начальная страница расположенная на главном окне приложения включает в себя такие виджеты, как Label, для вывода статического текста, PhotoImage, для отображения картинки и Button – это виджет кнопка, который имеет основной триггерный метод «command()», который срабатывает при нажатии на кнопку [20]. Объединять все эти виджеты в группу, расположенную на одной странице, позволяет виджет Frame. Реализация начальной страницы приложения представлена на рисунке 8.

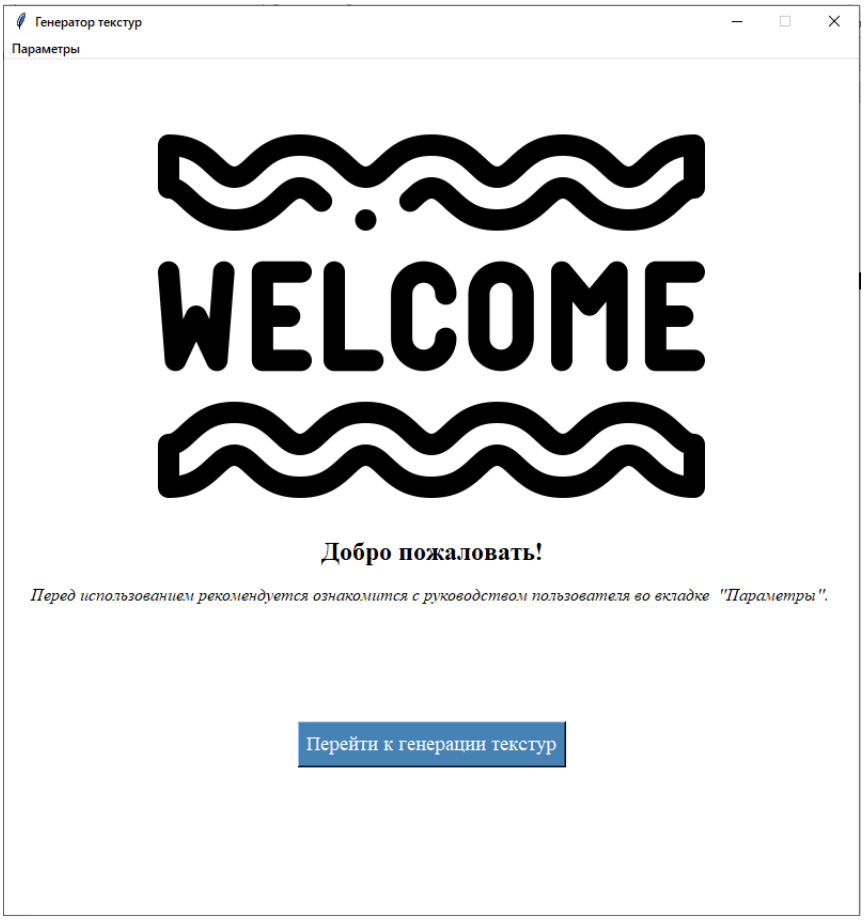

Рисунок 8 – Начальная страница входа в программу

После нажатия пользователем на кнопку «Перейти к генерации текстур» открывается страница для ввода параметров генерации текстур. Данная страница, помимо вышеперечисленных виджетов, также содержит виджет Entry, который предназначен для ввода параметров пользователем. Помимо этого, на данной странице содержатся специальные кнопки, на которых изображены картинки с вопросительным знаком. Эти кнопки служат для вызова одной из функций специального модуля оконных сообщений MessageBox, а именно «showinfo()», которая предназначена для вывода справочной информации о параметрах генерации. Страница для ввода параметров генерации текстур представлена на рисунке 9.

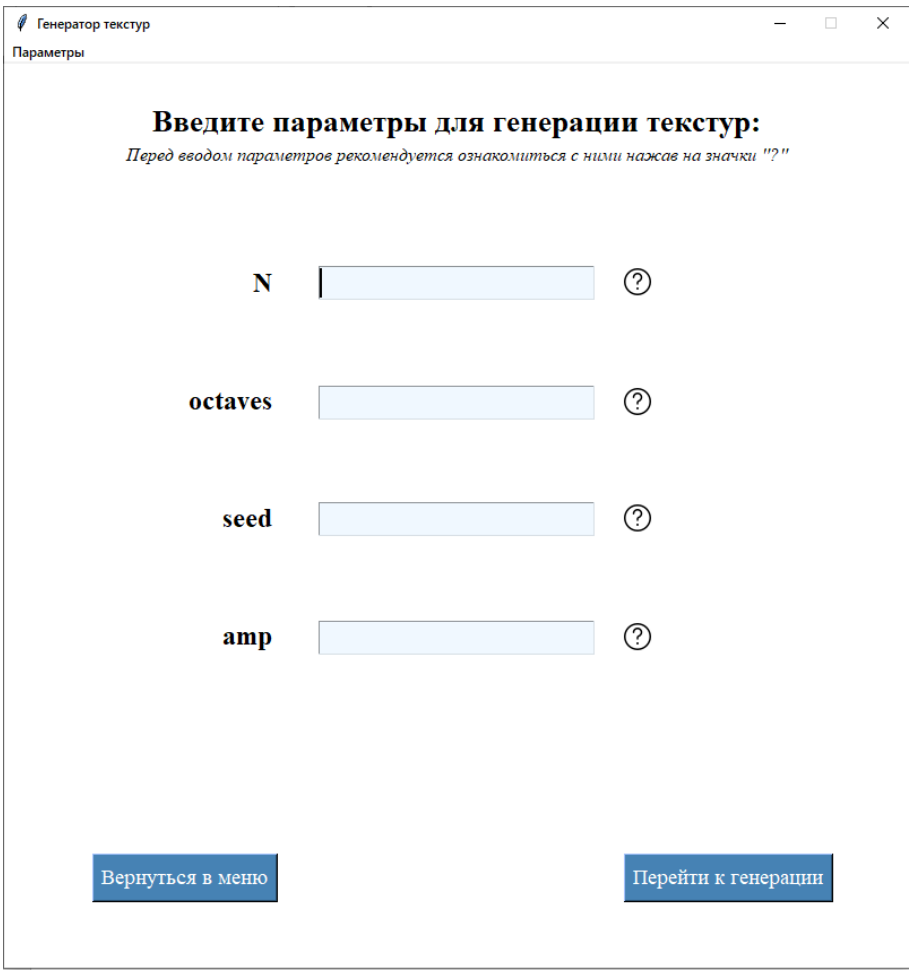

Рисунок 9 – Страница для ввода параметров

На рисунке 10 изображено справочное оконное сообщение одного из вводимых параметров для генерации текстур.

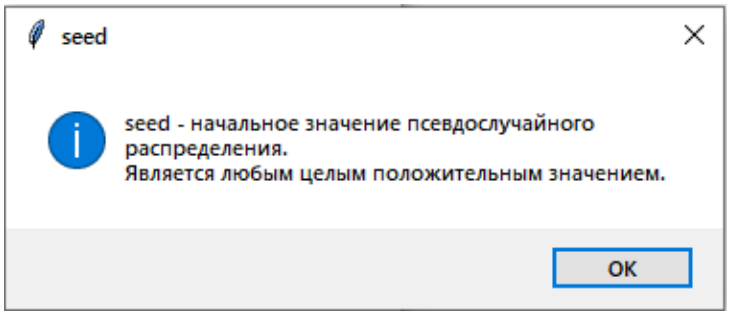

Рисунок 10 – Справочное оконное сообщение

Следующим этапом, после ввода параметров, перед пользователем откроется главная страница генерации. На этой странице находятся все вышеупомянутые виджеты и справочное оконное сообщение о режиме демонстрации. На главной странице содержатся две основные кнопки. Кнопка «Сгенерировать шум Перлина» отвечает за расчёт на основе введённых параметров матрицы высот алгоритма и вывод их в виде высотного изображения. Следующая кнопка «Генерация ландшафта» накладывает сгенерированную текстуру на ландшафт умножая введённое ранее значение параметра amp на матрицу высот и запускает режим демонстрации сгенерированного ландшафта. Реализация главной страницы генерации изображена на рисунке 11.

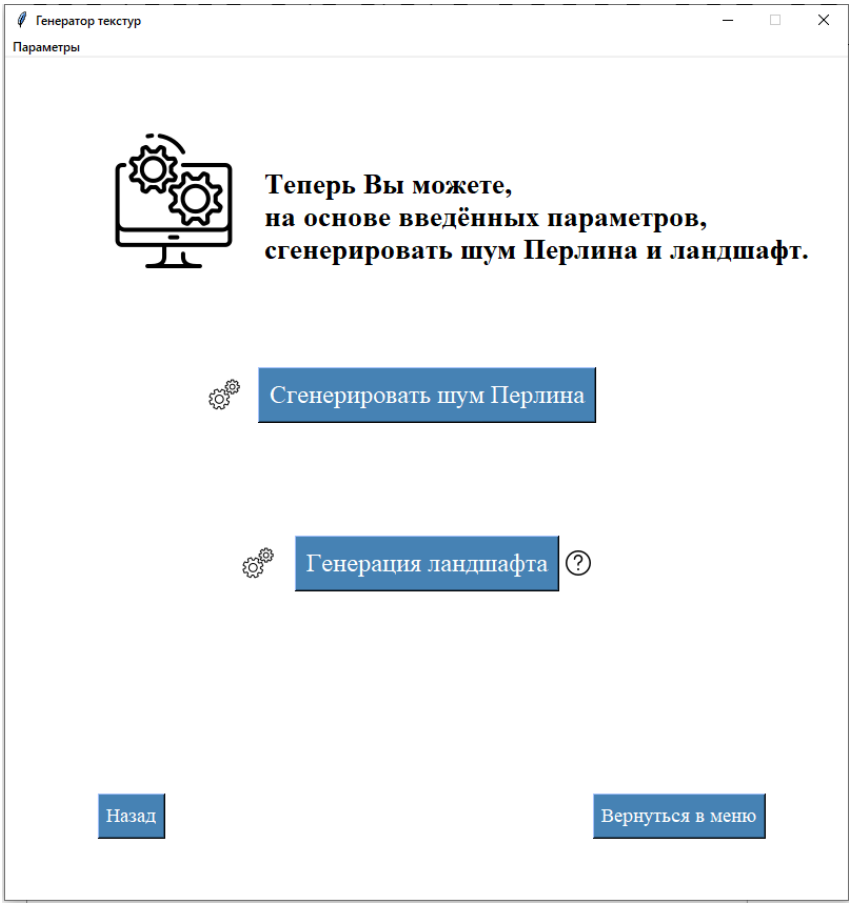

Рисунок 11 – Главная страница генерации текстур

На рисунке 12 показано справочное оконное сообщение о выходе из режима демонстрации кнопки «Генерация ландшафта».

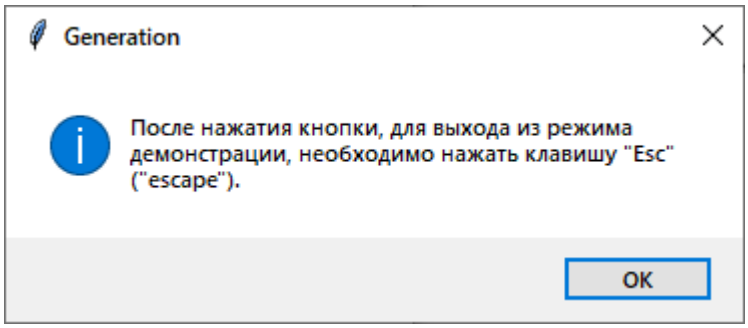

Рисунок 12 – Главная страница генерации текстур

Также на главном окне приложения в левом верхнем углу находится иерархическое меню, представленное в виде виджета Menu. Это меню содержит вкладку параметры, содержащую подменю, в котором можно ознакомится с руководством пользователя, информацией об авторе и выйти из приложения.

Таким образом, созданный интерфейс соответствует всем заявленным требованиям, является привлекательным, понятным и имеет достаточное количество справочного материала.

3.3 Программная реализация процесса генерации и наложение текстур

Одной генерации матрицы высот на основе алгоритма шума Перлина недостаточно, для дальнейшего генерирования текстуры и ландшафта. Поэтому необходимо решить проблему программной реализации процесса генерации и наложение текстур.

На первом этапе, для того чтобы сгенерировать текстуру, необходимо проверить на адекватность введённые пользователем параметры и считать их.

Затем нужно создать M случайно-расположенных вещественных точек на матрице размером N на N. Далее сгенерировать матрицу высот на основе алгоритма шума Перлина и создать двумерное-растровое (высотное) изображение с помощью метода «imshow()» библиотеки matplotlib [5]. Реализация этих действий в виде функции «move_to_generation()» представлена на рисунке 13.

```
# Генерация текстуры
def move_to_generation():
    # Параметры генерации
    global N, my_octaves, my_seed, my_amp, X, X_min
    # Проверка правильно введённых параметров
    if check():
        page4.pack_forget()
        page5.pack(fill=BOTH, expand=1)
        # Чтение введённых параметров
        N = int(entry4_1.get())
        my_octaves = int(entry4_2.get())
        my_seed = int(entry4_3.get())
        my_amp = float(entry4_4.get())

        # Генерация равномерно распределенных координат для функции Noise
        lin_array = np.linspace(1, 10, N, endpoint=False)
        x, y = np.meshgrid(lin_array, lin_array)
        # Генерация матрицы высот
        X = Noise(x, y, octaves = my_octaves, seed = my_seed)
        X_min = looking_min(Noise(x, y, octaves = my_octaves, seed = my_seed))
        plot.title('Шум Перлина')
        # Генерация высотного изображение
        plot.imshow(X)

# Отображение текстуры
def gen_noise():
    plot.show()
```

Рисунок 13 – Программная реализация процесса генерации текстуры

На рисунке 13 также представлен метод «gen_noise» для отображения сгенерированной текстуры, который вызывается при нажатии пользователем на кнопку «Сгенерировать шум Перлина».

Следующим этапом будет наложение текстуры на ландшафт. Для этого каждое высотное значение точки на матрице необходимо умножить на введённый пользователем параметр amp.

Чтобы сгенерировать и отобразить ландшафт будет использоваться современный движок Ursina Engine для 3D-моделирования. Ursina Engine – это открытый игровой 3D-движок под все операционные системы для языка программирования Python, базирующийся на основе движка Panda3D [24]. Генерированный ландшафт будет иметь блочный вид, для этого будет создаваться пользовательский класс Voxel. У этого класса определены размер, позиция в пространстве, центрирование, цвет, модель и встроенные шейдеры [15]. Помимо этого, будет использоваться объект встроенного класса FirstPersonController для перемещения в режиме демонстрации по созданной текстуре [23]. Наложение текстуры, создание объектов и запуск режима демонстрации реализован в виде функции «gen_ursina()», которая вызывается при нажатии пользователем на кнопку «Генерация ландшафта». Данная функция представлена на рисунке 14.

Рисунок 14 – Программная реализация процесса наложения текстуры и генерация ландшафта

Таким образом, пользователь, поэтапно нажимая на кнопки, расположенные на главной странице генерации, сможет увидеть сгенерированную текстуру на основе алгоритма шума Перлина и сгенерированный ландшафт.

3.4 Тестирование процесса генерации текстур

Одним из ключевых моментов разработки приложения является тестирование системы. В крупных компаниях на проведение правильного и досконального тестирования порой выделяется большое количество ресурсов и времени, практически сопоставимых с затратами на разработку самой системы [14]. Без тестирования любая компания рискует понести огромные

потери в прибыли, за счёт нарушение целостности данных и постоянных сбоев в работе приложения [25]. В следствии чего, смысл в разработке программного продукта просто будет исчерпан.

Таким образом, выбор правильных подходов и методов тестирования имеет ключевую роль в разработке программного продукта. Существует несколько основных видов тестирования.

Первым и основополагающим видом тестирования является модульное тестирования. Данное тестирование направленно на проверку одного логически выделенного модуля, класса, функции, цикла. Основной целью такого тестирования является выявление вычислительных (физических) ошибок в работе программы.

Следующий вид – это системное тестирование. Оно направленно на проверку работы функций и системы в целом. Поэтому этот вид включает в себя не только функциональное тестирование, но и оценку качества работы системы. За счёт этого тестирование можно проверить систему на удовлетворение ранее поставленным функциональным требованиям.

Интеграционное тестирование также является одним из важных подходов. Его целью является проверка взаимодействия между собой нескольких модулей или компонентов программы [3].

При работе с заказчиком также уместно проводить приёмочное тестирование. Данное тестирование проводится совместно с предполагаемым пользователем и позволяет провести оценку системы на соответствия потребностям, требованиям и удобству пользования.

Помимо вышеупомянутых видов тестирования существует ещё немалое количество и других. Их целями может быть не только сама разработка программы, но и её сопровождение, оценка бизнес-процессов и составление сценария работы предполагаемого пользователя с целью проверки его удовлетворённости во время использования, выявления сбоев и ошибок программы.

Для полноценной проверки работы системы необходимо включать сразу несколько видов тестирования.

Одним из реализованных подходов в приложении является модульное тестирование. Для этого был использован модуль Unittest, основанный на тестах JUnit языка Java. Unittest является встроенным модулем в Python, включающим все необходимые инструменты для проведения тестирования. В данном модуле содержится класс TestCase с множеством проверочных методов assert для проверки выполнения условия в определенной части кода [25].

Данный вид тестирования используются для проверки корректности вводимых пользователем значений параметров генерации псевдослучайных текстур на основе алгоритма шума Перлина. Все вводимые пользователем параметры генерации имеют числовое значение. Параметры N, octaves, seed являются целочисленными и положительными, а параметр amp имеет вещественный тип.

При введении пользователем параметров и нажатием на кнопку «Перейти к генерации», значения параметров считываются. Ошибкой пользователя может быть неверно введённые типы значений. Для определения данных ошибок использовался метод «assertIsInstance(a, type)», имеющий два входных параметра, а именно значение и тип, с которым оно сравнивается.

После проведения тестирования были приняты меры по обнаружению и сообщению пользователю о допущенных ошибках. Для этого необходимо, при переводе введённых пользователем значений использовать встроенный в Python механизм исключений «try except», который в случае ошибки вернёт значение «False». Помимо проверки введённых значений на тип данных, некоторые параметры должны иметь положительные значения.

Если хоть какие-то из выше перечисленных условий не выполняются, то при обнаружении ошибки, используется метод «showerror()» класса MessageBox, который выводит на экран сообщение об ошибке. Одним из

примеров оконного сообщения об допущенной ошибке приведён на рисунке 15.

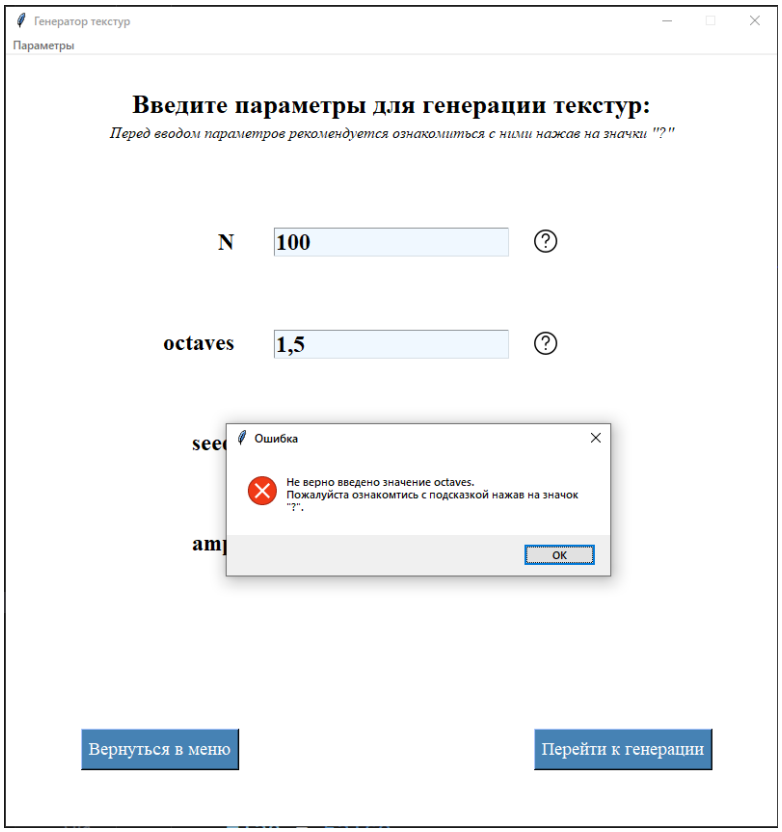

Рисунок 15 – Не верно введённое значение параметра octaves

Следующим этапом необходимо провести проверку работы программы с помощью системного тестирования. Все функции переходов и взаимодействий с страницами и справочными окнами были проверены во время разработки интерфейса программы, кроме основных двух кнопок генерации, а именно кнопка «Сгенерировать шум Перлина» и кнопка «Генерация ландшафта». Для проведения тестирования, проверки работоспособности системы и адекватности, анализа полученных результатов проведём несколько разноплановых тренировочных запусков программы генерации псевдослучайных текстур.

Первым этапом создадим сглаженные текстуры. Для этого значение параметра octaves необходимо задать больше единицы, а значение amp наиболее близкое к единице (рисунок 16).

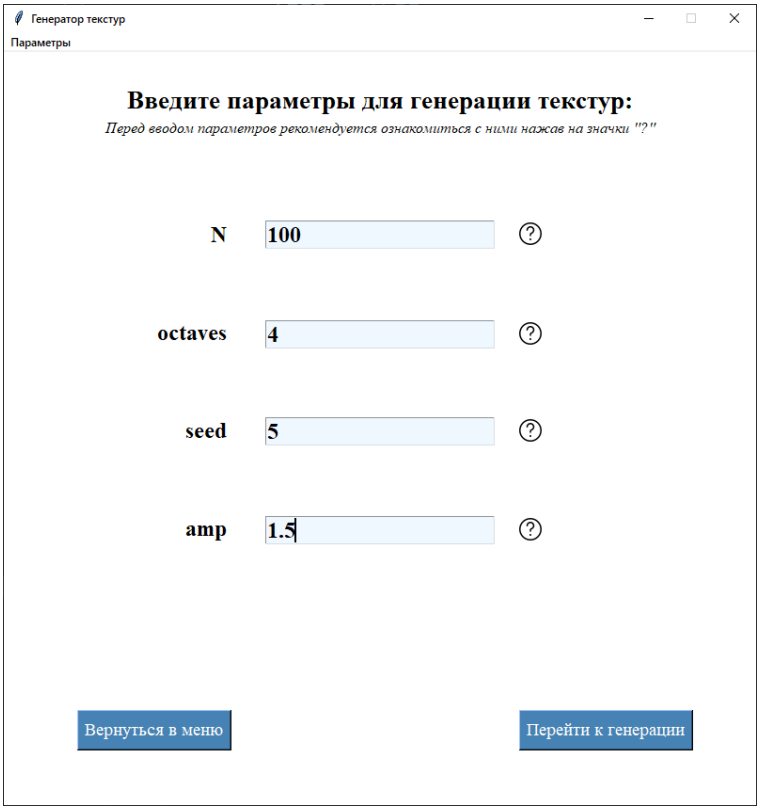

Рисунок 16 – Задание входных параметров генерации для сглаженной текстуры

Теперь проверим полученные результаты. Для этого сгенерируем шум Перлина и ландшафт. Результат приведён на рисунках 17 и 18.

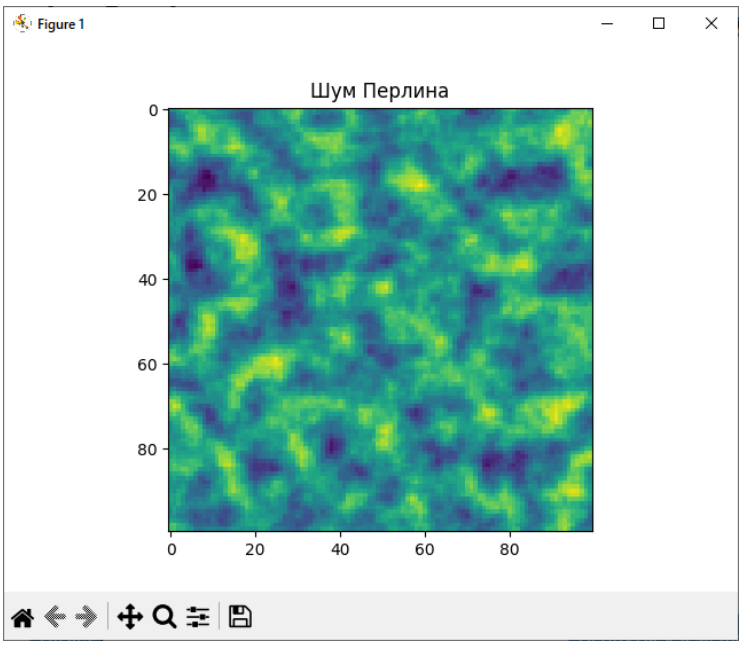

Рисунок 17 – Сгенерированный шум Перлина при большом значении параметра octaves

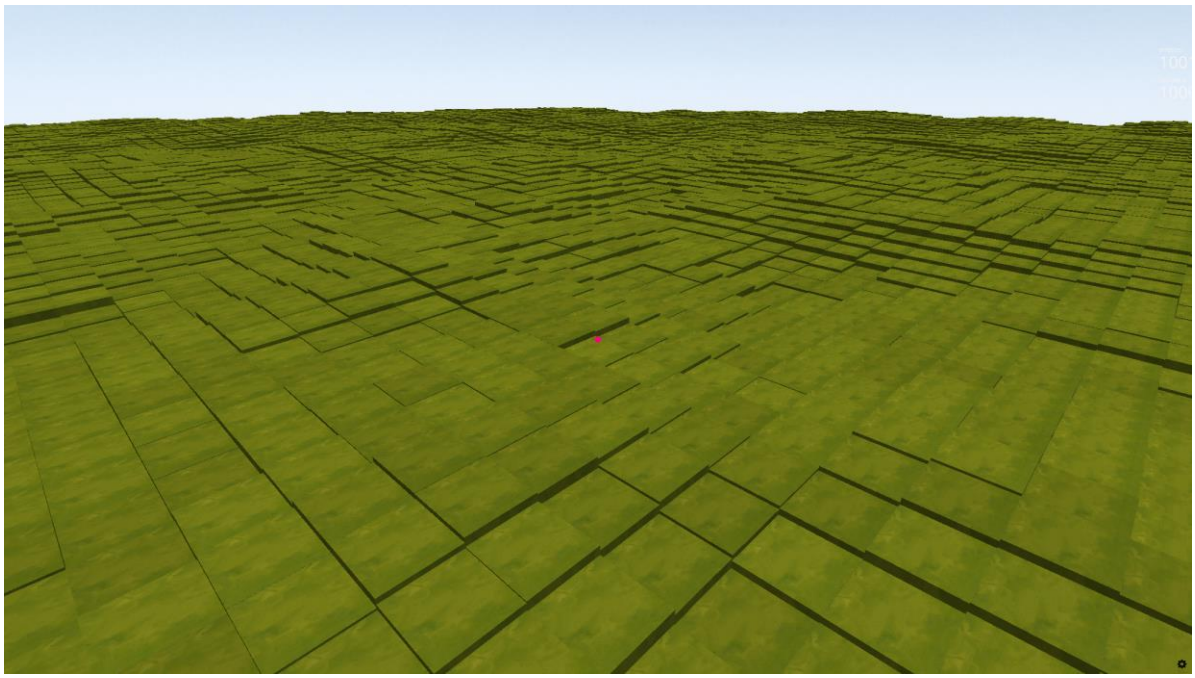

Рисунок 18 – Просмотр сглаженного ландшафта в режиме демонстрации

Затем создадим текстуры с повышенной рельефностью. Размер ландшафта, контролируемый параметром N, оставим тот же для сравнения,

значение параметра octaves сделаем равным единицы, а значение параметра amp гораздо больше единицы (рисунок 19).

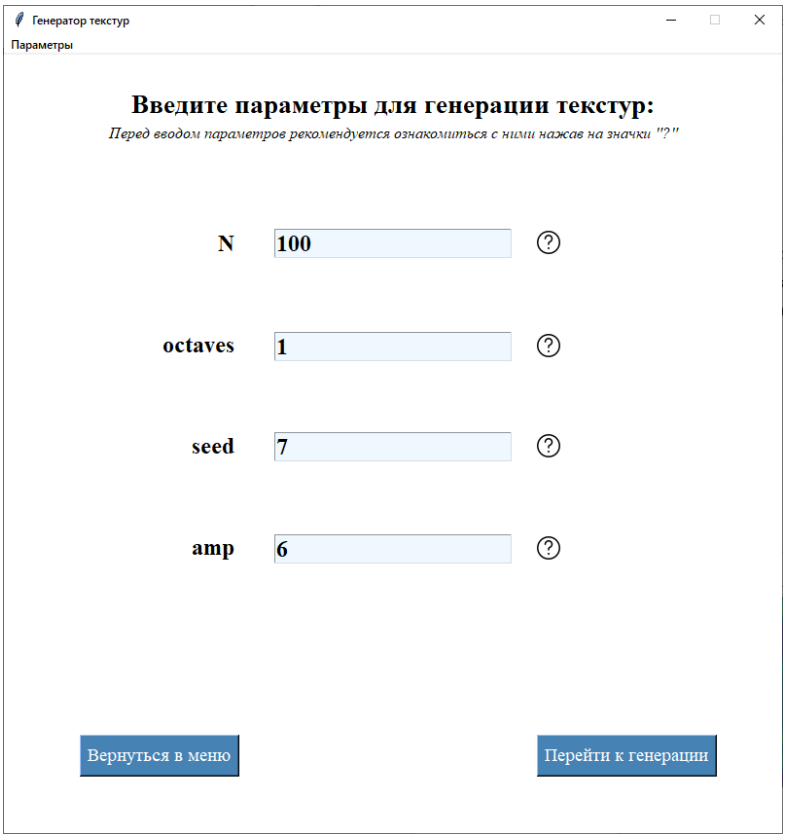

Рисунок 19 – Задание входных параметров генерации для рельефной текстуры

Снова проверим полученные результаты. Для этого нажмём на кнопку «Сгенерируем шум Перлина», затем на кнопку «Генерация ландшафта». Результат генерации псевдослучайной текстуры и ландшафта приведён на рисунках 20 и 21.

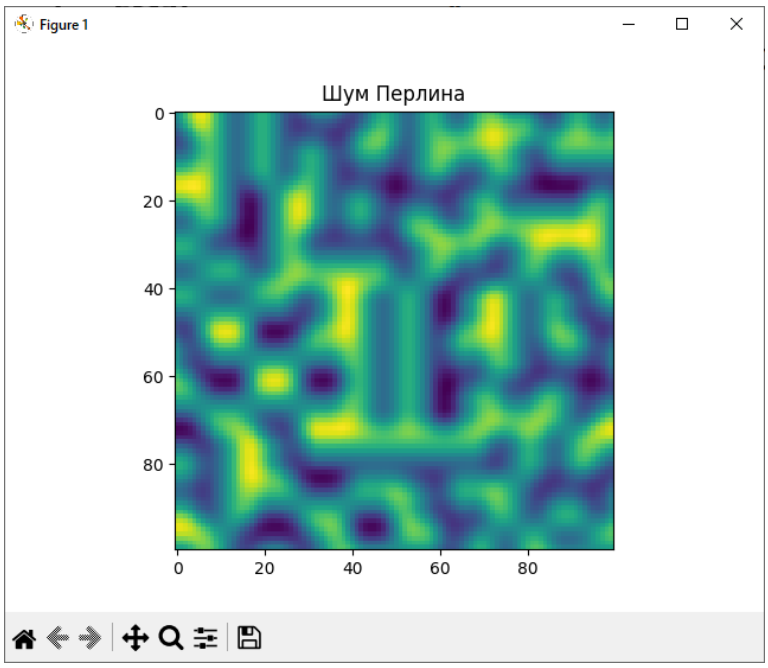

Рисунок 20 – Сгенерированный шум Перлина для рельефной текстуры

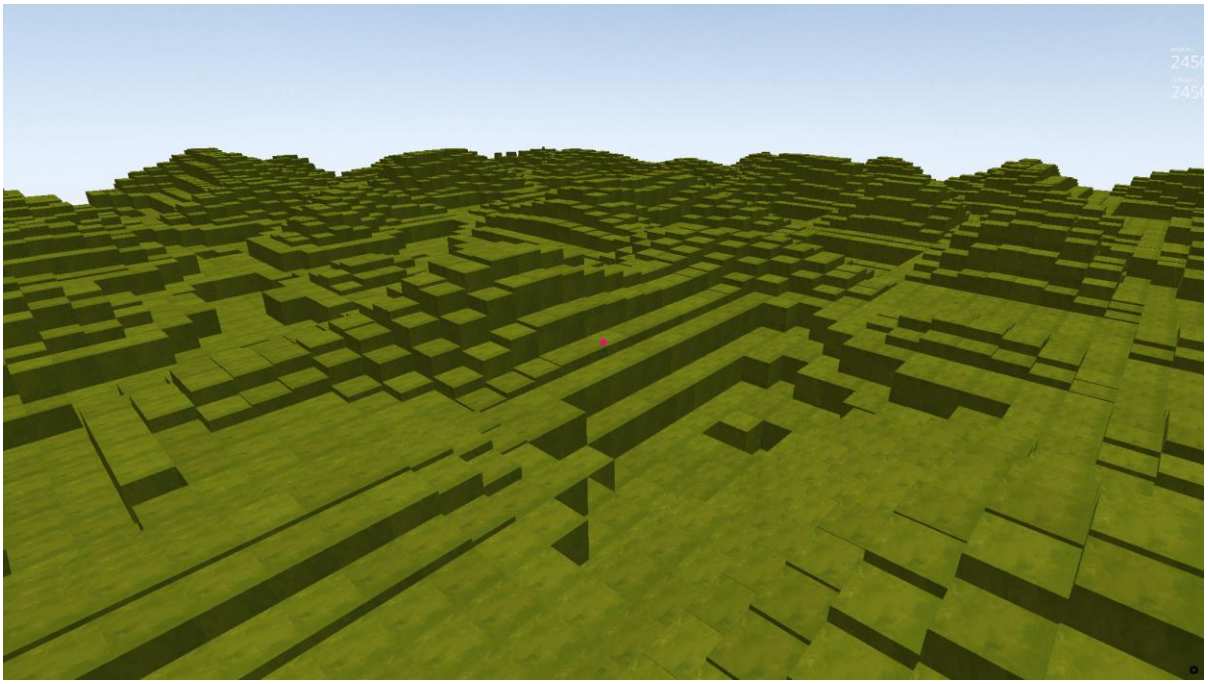

Рисунок 21 – Просмотр рельефного ландшафта в режиме демонстрации

По итогам тестирования генерации видно, что при задании большего значения параметра octaves текстура дополнительно расщепляется и

сглаживается, а в сочетании с малым значением параметра amp максимальное возвышение блоков мало. Таким образом, сгенерированная текстура получается более сглаженной. При большом значении коэффициента возвышении и малом значении коэффициента расщепления в текстуре появляются явно выраженные неровности, ямы, пригорки, что делает созданный ландшафт более рельефной.

Благодаря модульному и системному тестированию можно сделать вывод, что программа работает корректно и способна предотвращать ошибки со стороны пользователя оповещая его об этом. А по результатам тестирования процесса генерации видно, что реализованная программа является хорошим инструментом для создания текстур.

Выводы к главе 3

В данной главе была описана и программно-реализована работа системы генерации псевдослучайных текстур на основе шума Перлина.

Первым шагом был разработан алгоритм программной реализации работы системы с помощью блок-схемы. Выбраны все компоненты и необходимые инструменты для программной реализации.

На основе алгоритма был спроектирован и реализован интерфейс пользователя. А также разработаны функции генерации и наложения созданной текстуры на ландшафт.

Последним этапом было проведено тестирование как отдельных модулей, функций, так и всей системы в целом. Были разработаны подходы по предотвращению ошибок, вызванных неправильной работой пользователя, посредством оповещения. Также проведено тестирование процесса генерации текстур на различных параметрах, и сделаны выводы о его работе.

Заключение

Итогом выпускной квалификационной работы является программная реализация алгоритма генерации псевдослучайных текстур на основе шума Перлина.

Основная проблема в рамках работы заключалась в разработке подхода для генерации и применении псевдослучайных текстур в компьютерной графике. Для решения данной проблемы требовалось создать программный модуль, который позволил бы быстро и удобно генерировать параметризированные текстуры для их дальнейшего применения.

Для решения поставленной цели первым этапом была проанализирована предметная область, а также проведён сравнительный анализ существующих алгоритмов генерации шума. Это было необходимо для оценки правильности и оптимальности выбора алгоритма шума Перлина для генерации текстур. Затем был описан алгоритм генерации текстур на основе шума Перлина, его параметры и модификации.

Следующим этапом была спроектирована система генерации псевдослучайных текстур. На данном этапе были определены функциональные требования к разрабатываемой системе. На основе этих требований была составлена структура системы с помощью концептуальной модели.

Последним этапом была программная реализация системы генерации псевдослучайных текстур. Для этого была сделана блок-схема алгоритма работы программы. Следующим действием был разработан и реализован интерфейс программы. После реализации программы произведено тестирование. Это было сделано для того, чтобы предотвратить ошибки и сбои работы программы и убедиться в правильности полученных результатов.

Таким образом, разработанное приложение является инструментом для генерации параметризированных, псевдослучайных текстур.

Список используемой литературы

1. Алгоритм «diamond-square» для построения фрактальных ландшафтов / [Электронный ресурс]. URL: https://habr.com/ru/articles/111538/, (дата обращения: 20.01.2024).
2. Боресков, А. В. Основы компьютерной графики: учебник и практикум для вузов / А. В. Боресков, Е. В. Шикин. – Москва: Издательство Юрайт, 2019. – 219 с. – (Серия: Профессиональное образование). – ISBN 978-5-534-11630-4.
3. Васильев, Ю. Обработка естественного языка. Python и spaCy на практике. – СПб.: Издательский дом «Питер», 2021. – 256 с.: ил. – (Серия «Библиотека программиста»). ISBN 978-5-4461-1506-8.
4. Вигерс, К. Разработка требований к программному обеспечению: учебное пособие / К. Вигерс, Д. Битти. – Издательство «БХВ», 2019. – 736 с. – ISBN 978-5-99-098053-2.
5. Ганков, М. С. Разработка программ на языке Python для графической интерпретации точечных отображений. / М. С. Ганков, В. Ю. Ильичев // Научное обозрение. Технические науки. 2021.
6. Гниденко, И. Г. Технология разработки программного обеспечения: учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. – 2-е изд., перераб. и доп. – Москва: Издательство Юрайт, 2024. – 248 с. – (Профессиональное образование). – ISBN 978-5-534-18131-9.
7. Забродин, А. В. Основы проектирования информационных систем с помощью языка UML: учебное пособие / А. В. Забродин, В. П. Бубнов. — Санкт-Петербург: ПГУПС, 2018. — 46 с. – ISBN 978-5-7641-1133-9.
8. Златопольский, Д. М. Основы программирования на языке Python. 2-е изд. – Москва: ДМК Пресс, 2018. – 396 с.: ил. – ISBN 978-5-97060-641-4.
9. Ильечев, В. Ю. Визуализация масштабируемых 3d-моделей с помощью модуля Matplotlib для Python. // Системный администратор. 2020. № 12 (217). С. 86-89.

10. Лисяк, В. В. Основы компьютерной графики: 3D-моделирование и 3D-печать : учебное пособие / В. В. Лисяк. – Ростов-на-Дону: ЮФУ, 2021. – 109 с. – ISBN 978-5-9275-3825-6.

11. Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: Издательский дом «Питер», 2018. – 352 с.: ил. – (Серия «Библиотека программиста»). ISBN 978-5-4461-0772-8.

12. Мартынов А.А. Принципы проектирования графического интерфейса пользователя. // Журнал "Информатика и вычислительная техника", 2018. - С. 89-102.

13. Медведева, О.А. Генерация карт высот с использованием шума Перлина для построения ландшафтов / О.А. Медведева, К.И. Гордеев, Г.Ю. Гуляев // Фундаментальные и прикладные научные исследования: актуальные вопросы, достижения и инновации. – Пенза: Наука и Просвещение, 2017. С. 105-108.

14. Пероцкая, В. Н. Основы тестирования программного обеспечения: учебное пособие / В. Н. Пероцкая, Д. А. Градусов; Владимирский. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир: Издательство ВлГУ, 2017. – 100 с. – ISBN 978-5-9984-0777-2.

15. Реализация генерации и отображения ландшафта в трехмерной плоскости / [Электронный ресурс]. URL: https://www.bibliofond.ru/id=877950, (дата обращения: 14.01.2024).

16. Руководство по Tkinter / [Электронный ресурс]. URL: https://metanit.com/python/tkinter/, (дата обращения: 17.02.2024).

17. Слеповичев, С. О. Шум Перлина как способ получения компьютерных спецэффектов природных явлений. / С. О. Слеповичев // Инновационное развитие. 2017. № 2 (7). С. 32-33.

18. Тарасян, В. С. Интерполяция распределённых данных горизонталей для получения цифровой модели рельефа / В. С. Тарасян, Н. В. Дмитриев // Инженерный вестник Дона. 2018. №1. URL: http://ivdon.ru/ru/magazine/archive/n1y2018/4774, (дата обращения: 11.02.2024).

19. Тозик, В. Т. Компьютерная графика и дизайн: учебник для нач. проф. образования / В. Т. Тозик, Л. М. Корпан. – 3-е изд., стер. – Москва: Издательский центр «Академия», 2013. – 208 с. – ISBN 978-5-7695-9718-3.

20. Шелудько, В. М. Основы программирования на языке высокого уровня Python: учебное пособие; Южный федеральный университет. – Ростов-на-Дону; Таганрог: Издательство Южного федерального университета, 2017. – 112 с.

21. BPMN Specification – Business Process Model and Notation / [Электронный ресурс] URL: https://www.bpmn.org/, (дата обращения: 12.02.2024).

22. Moore, A. D. Python GUI Programming with Tkinter, 2nd Edition: expert insight / A. D. Moore. – UK, Birmingham: published by Packt Publishing Ltd, 2021. – 860 p. – ISBN 978-1-80181-592-5.

23. Perlin noise 1.12 / [Электронный ресурс] URL: https://pypi.org/project/perlin-noise/, (дата обращения: 10.01.2024).

24. Romano, F. Learn Python Programming, 2nd Edition: beginner’s guide to programming / F. Romano. – UK, Birmingham: published by Packt Publishing Ltd, 2018. – 469 p. – ISBN 978-1-78899-666-2.

25. The Art of Software Testing, 3rd Edition / [Электронный ресурс] URL: https://malenezi.github.io/malenezi/SE401/Books/114-the-art-of-software-testing-3-edition.pdf, (дата обращения: 18.02.2024).