

Реализация алгоритма шума Перлина на языке Python

Болтовский Гавриил Александрович

Приамурский государственный университет им. Шолом-Алейхема

Студент

Аннотация

Целью данной работы является демонстрация классического алгоритма шума Перлина и его программной реализации на языке Python. Методика исследования включает разработку кода генерации двумерного шума Перлина, визуализацию полученных карт и сравнение результатов с картой случайного (белого) шума. Сделан вывод, что алгоритм шума Перлина формирует плавную псевдослучайную текстуру и может использоваться в обучении для иллюстрации принципов процедурной генерации.

Ключевые слова: шум Перлина, двухмерная графика, процедурная генерация

Implementation of Perlin noise algorithm in Python

Boltovsky Gavriil Alexandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

The purpose of this work is to demonstrate the classic Perlin noise algorithm and its software implementation in Python. The research methodology includes developing code to generate two-dimensional Perlin noise, visualizing the resulting maps, and comparing the results with a map of random (white) noise. It is concluded that the Perlin noise algorithm generates a smooth pseudorandom texture and can be used in teaching to illustrate the principles of procedural generation.

Keywords: Perlin noise, two-dimensional graphics, procedural generation

1 Введение

1.1 Актуальность исследования

Алгоритм шума Перлина изначально был разработан для создания реалистичных текстур в компьютерной графике и получил широкое применение в отрасли. Он представляет собой градиентный шум, генерируемый на основе набора псевдослучайных единичных векторов в узлах сетки с последующей сглаженной интерполяцией между ними. Благодаря этому свойству шум Перлина активно используется в двумерной и трёхмерной графике для синтеза природных эффектов (дым, облака, туман, огонь и т.д.) и процедурного формирования текстур и ландшафтов. Например, в игровых приложениях на основе шума Перлина создаются реалистичные ландшафты, как это имело место в ранних версиях игры Minecraft. Кроме прикладного

значения, данный алгоритм актуален и в образовательном плане: относительно простая реализация и наглядный результат делают шум Перлина удобным демонстрационным примером для изучения принципов процедурной генерации и псевдослучайности.

1.2 Обзор исследований

Шум Перлина как инструмент процедурной генерации рассматривается в ряде работ, где подчёркивается его универсальность и применимость как в визуализации природных явлений, так и в задачах компьютерной графики.

Е.А. Жучков и Т.Г. Жучкова [1] исследуют реализацию алгоритма генерации ландшафтов на основе шума Перлина и подчёркивают его эффективность в формировании рельефа с плавными переходами и реалистичной структурой. В работе В.Р. Зиновского и Е.А. Трифионовой [2] алгоритм используется в контексте анализа целостности цифровых изображений, где он применяется для моделирования преднамеренных искажений, что демонстрирует возможность его использования за пределами задач визуализации. С.О. Слеповичев рассматривает [3] шум Перлина как способ синтеза визуальных эффектов природных явлений, подчёркивая его применимость в создании облаков, огня и других динамичных текстур в реальном времени. О.А. Медведева и К.И. Гордеев анализируют [4] особенности генерации карт высот с использованием данного алгоритма и указывают на его устойчивость к искажениям и гибкость в управлении детализацией.

Н.А. Агафонов, А.М. Гиацинтов и А.В. Родителей описывают [5] методы визуализации погодных явлений в имитационных моделях и включают шум Перлина как одну из ключевых составляющих при синтезе текстур, имитирующих облачность и атмосферные эффекты. В работе Н.С. Иванова и П.Г. Горбуновой алгоритм рассматривается [6] как основа процедурной генерации ландшафтов в образовательных и прикладных средах, акцентируя внимание на его доступности для реализации. С.В. Колесников в контексте задач генерации рельефа подчёркивает [7] наглядность получаемых с помощью шума Перлина результатов и указывает на возможность использования его в задачах автоматизированного формирования геометрических моделей.

1.3 Цель исследования

Целью исследования является реализация алгоритма шума Перлина на языке Python с последующей визуализацией результатов и сравнительным анализом с классическим случайным шумом.

1.4 Постановка задачи

Для достижения указанной цели требуется решить ряд задач. Во-первых, необходимо разработать программную реализацию классического алгоритма шума Перлина для двумерного случая, вычисляющую значения шумовой функции в каждой точке заданной области. Во-вторых, следует сгенерировать

на основе этой реализации двумерную карту распределения значений шума Перлина, а для сравнения сформировать аналогичную карту несглаженного случайного шума (белого шума). Наконец, полученные карты подлежат визуализации и анализу с целью выявления отличий в характере псевдослучайных структур, порождаемых алгоритмом Перлина и обычным случайным распределением.

2 Методы исследования

В работе реализован алгоритм генерации классического двумерного шума Перлина средствами языка Python. Принцип действия алгоритма состоит в том, что вместо прямой интерполяции случайных чисел на сетке в узлах сетки задаются псевдослучайные градиентные векторы. Для произвольной точки плоскости значение шума определяется скалярными произведениями векторов градиента в ближайших узлах на векторы смещения от этих узлов до точки, с последующим сглаженным усреднением (интерполяцией) полученных величин. В отличие от обычного дискретного шума, где каждый узел задаёт фиксированное случайное значение и может приводить к резким переходам, градиентный подход Перлина обеспечивает непрерывность и естественность получаемой «случайной» картины за счёт корреляции соседних точек.

Для сглаживания интерполяции в алгоритме используется специальная пятой степени функция $f(t)$, обнуляющая производные на концах отрезка $[0,1]$:

$$f(t) = 6t^5 - 15t^4 + 10t^3$$

где t – локальная координата точки внутри элементарной ячейки решётки. Эта сглаживающая функция (quintic curve) деформирует параметр интерполяции, обеспечивая более плавный переход значений на границах ячеек. Затем вычисленные скалярные произведения для вершин ячейки комбинируются билинейной интерполяцией.

```
def fade(t):
    return 6 * t**5 - 15 * t**4 + 10 * t**3

def lerp(a, b, t):
    return a + t * (b - a)
```

Пусть n_{00} , n_{10} , n_{01} , n_{11} – значения, полученные в угловых узлах ячейки (левый верхний, правый верхний, левый нижний и правый нижний угол соответственно) как скалярные произведения вектора градиента узла на вектор до точки внутри ячейки. Обозначив $u = f(\Delta x)$, $v = f(\Delta y)$, где $\Delta x, \Delta y \in [0,1]$ – локальные координаты точки внутри ячейки по осям, результат вычисляется по формуле:

$$N(x, y) = (1 - v)[(1 - u)n_{00} + un_{10}] + v[(1 - u)n_{01} + un_{11}].$$

Таким образом получают значение шума Перлина $N(x,y)$ в координатах точки, лежащей внутри одной ячейки сетки.

```
def grad(hash, x, y):
    h = hash & 3
    u = x if h < 2 else y
    v = y if h < 2 else x
    return (u if (h & 1) == 0 else -u) + (v if (h & 2) == 0 else
-v)

def perlin(x, y, permutation):
    """Вычисление значения шума Перлина в точке (x, y)"""
    xi = int(x) & 255
    yi = int(y) & 255
    xf = x - int(x)
    yf = y - int(y)

    u = fade(xf)
    v = fade(yf)

    aa = permutation[permutation[xi] + yi]
    ab = permutation[permutation[xi] + yi + 1]
    ba = permutation[permutation[xi + 1] + yi]
    bb = permutation[permutation[xi + 1] + yi + 1]

    x1 = lerp(grad(aa, xf, yf), grad(ba, xf - 1, yf), u)
    x2 = lerp(grad(ab, xf, yf - 1), grad(bb, xf - 1, yf - 1), u)

    return (lerp(x1, x2, v) + 1) / 2 # Результат нормализован в
[0, 1]
```

Для генерации полной двумерной «карты» шума алгоритм применяют ко всем точкам дискретной сетки заданного разрешения. В рамках данного исследования реализация ограничена одной октавой шума (одним слоем без последующего наложения многомасштабных фрактальных компонент). Полученные численные значения $N(x,y)$ (находящиеся в диапазоне примерно $[-1,1]$) затем преобразованы в уровни серого для визуального анализа.

```
def generate_perlin_noise(width, height, scale=10.0):
    p = list(range(256))
    random.shuffle(p)
    p += p
    noise = np.zeros((height, width))
    for y in range(height):
        for x in range(width):
            nx = x / scale
            ny = y / scale
            noise[y][x] = perlin(nx, ny, p)
    return noise
```

```
def show_noise_image(noise_array):  
    plt.imshow(noise_array, cmap='gray', interpolation='nearest'  
)  
    plt.axis('off')  
    plt.show()
```

Пример запуска алгоритма для изображения 100 на 100, при масштабировании 20:

```
width, height = 100, 100  
scale = 20.0  
noise = generate_perlin_noise(width, height, scale)  
show_noise_image(noise)
```

Для сравнения с коррелированным шумом Перлина аналогичным образом сгенерирована карта белого шума, в которой каждой точке присваивается независимое случайное значение (равномерно распределённое в диапазоне $[0,1)$).

Для сравнения с коррелированным шумом Перлина аналогичным образом сгенерирована карта белого шума, в которой каждой точке присваивается независимое случайное значение (равномерно распределённое в диапазоне $[0,1)$).

```
def generate_white_noise(width, height):  
    return np.array([[random.random() for _ in range(width)] for  
_ in range(height)])
```

Методом оценки эффективности алгоритма выбран качественный визуальный анализ: сравнение структуры изображений, полученных на основе шума Перлина и на основе несвязного случайного шума.

2. Результаты и обсуждение

Пример разных видов шума представлен на рисунке (рис. 1)

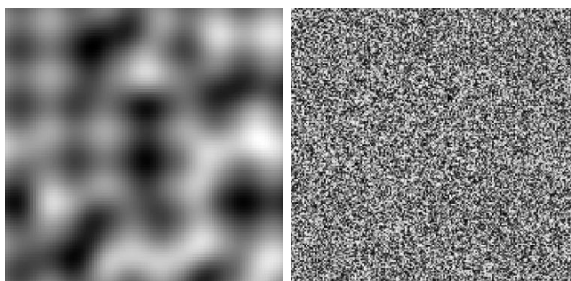


Рисунок 1 – Сравнение двумерных карт шума

Видно, что карта, сгенерированная по алгоритму Перлина (слева), имеет плавные переходы оттенков и выраженные областные структуры, тогда как

случайный (белый) шум (справа) представляет собой высокочастотную «зернистую» картину без каких-либо непрерывных областей. В вычисленном перлин-шуме соседние пиксели имеют близкие значения благодаря сглаженной интерполяции, что создаёт впечатление естественной изменчивости. Напротив, у белого шума каждый пиксель случаен и не зависит от соседних, вследствие чего изображение выглядит как мозаика из хаотичных точек.

Отмеченный контраст подтверждает известное преимущество алгоритма Перлина. Использование несглаженного случайного шума приводит к резкой, неестественной картине, напоминающей «снег» на экране телевизора, тогда как метод Перлина генерирует структурированную, плавную случайность. Фактически, алгоритм шума Перлина создаёт случайные вариации с определённой корреляцией на близких расстояниях, благодаря чему визуальный результат напоминает естественные узоры (например, облака или дым) вместо полного хаоса. Это демонстрирует, как введение математической структуры в процесс генерации случайности позволяет получать осмысленные (наглядные) результаты вместо простого набора разрозненных точек.

С практической точки зрения, реализация алгоритма шума Перлина на Python наглядно показывает принцип процедурной генерации текстур. Полученные изображения позволяют сделать качественное сравнение: перлин-шум даёт «гладкий» случайный рельеф, тогда как обычный белый шум не имеет крупных структур. Такой визуальный пример полезен в образовательном плане для формирования алгоритмического мышления. Анализируя различия между двумя видами шума, учащиеся могут увидеть влияние этапов алгоритма (градиенты, интерполяция) на характер результата. Таким образом, демонстрация шума Перлина служит средством визуального восприятия работы алгоритмов и подчёркивает значимость продуманного алгоритмического подхода при генерации сложных структур.

4 Выводы

Таким образом, алгоритм шума Перлина обеспечивает плавное псевдослучайное распределение значений и может эффективно использоваться в образовательных целях для развития алгоритмического мышления и наглядной демонстрации принципов процедурной генерации.

Библиографический список

1. Жучков Е.А., Жучкова Т.Г. Разработка и реализация алгоритма процедурной генерации ландшафта на основе шума Перлина // Актуальные проблемы модернизации математического и естественно-научного образования. 2020. С. 112 – 114.
2. Зиновский В.Р., Трифонова Е.А. Разработка программного обеспечения для обнаружения нарушения целостности цифрового изображения шумом Перлина // Актуальные научные исследования в современном мире. 2016.

- С. 16-19.
3. Слеповичев С.О. Шум Перлина как способ получения компьютерных спецэффектов природных явлений // Инновационное развитие. 2017. С. 32-33.
 4. Медведева О.А., Гордеев К.И. Генерация карт высот с использованием шума Перлина для построения ландшафтов // Фундаментальные и прикладные научные исследования: актуальные вопросы, достижения и инновации. 2017. С. 105-108.
 5. Агафонов Н.А., Гиацинтов А.М., Родителей А.В. Методы визуализации погодных явлений в имитационных системах // Вестник кибернетики. 2018. С. 61-65.
 6. Иванов Н.С., Горбунова П.Г. Процедурная генерация ландшафтов // Рост - развитие, образование, стратегии, технологии. 2018. С. 160-166.
 7. Колесников С.В. О решении задачи генерации ландшафтов // Решетневские чтения. 2013. С. 214-215.