УДК 004

Реализация простого алгоритма шума Перлина на C++

Ульянов Егор Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается и описывается реализация алгоритма шаума Перлина. Алгоритм будет разрабатываться на языке программирования C++ с среды разработки Visual Studio. Практическим результатом является разработанный алгоритм.
Ключевые слова: Алгоритм, шум Перлина, C++

Implementation of a simple Perlin noise algorithm in C++

Ulianov Egor Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses and describes the implementation of the Perlin noise algorithm. The algorithm will be developed in the C++ programming language from the Visual Studio development environment. The practical result is the developed algorithm.
Keywords: Algorithm, Perlin noise, C++

Шум Перлина — это тип градиентного шума, разработанный Кеном Перлином в 1983 году. Алгоритм имеет множество применений: процедурную генерацию ландшафта, применение псевдослучайных изменений к переменной и помощь в создании текстур изображений. Чаще всего реализуется в двух, трех или четырех измерениях, но может быть определен для любого количества измерений.

Шум Перлина — это процедурный примитив текстуры, тип градиентного шума, используемый художниками по визуальным эффектам для повышения реализма в компьютерной графике. Функция имеет псевдослучайный вид, однако все визуальные детали имеют одинаковый размер. Это свойство позволяет легко управлять алгоритмом; несколько масштабированных копий шума Перлина можно вставлять в математические выражения для создания самых разнообразных процедурных текстур. Синтетические текстуры с использованием шума Перлина часто используются в компьютерной графике, чтобы сделать сгенерированные компьютером визуальные элементы, такие как поверхности объектов, огонь, дым или

облака, более естественными, имитируя контролируемое случайное появление текстур в природе.

Шум Перлина также часто используется для создания текстур, когда память крайне ограничена, например, в демонстрациях. Преемники алгоритма, такие как фрактальный шум и симплексный шум, стали почти повсеместными в графических процессорах как для графики в реальном времени, так и для процедурных текстур не в реальном времени во всех видах компьютерной графики. Также алгоритм часто используют в видеоиграх, чтобы процедурно сгенерированный ландшафт выглядел естественно.

Цель данной статьи является реализация простого, но мощного алгоритма генерации псевдослучайного шума Перлина на языке C++.

В статье Н. А. Грузина провел сравнительный анализ двух популярных языков программирования: C++ и Python. Описал основные особенности, преимущества и недостатки каждого из них, а также рассмотрел области применения и целесообразность использования того или иного языка в различных ситуациях [1]. В своей работе К. Ф. Асанбаев рассмотрел один из самых популярных языков программирования C++. Приведел его описание и примеры реализации. [2]. В статье С. О. Слеповичева рассматривается Шум Перлина - алгоритм, используемый для создания разнообразных природных спецэффектов на компьютере. Обсуждаются основные принципы работы алгоритма, его применение для генерации различных типов поверхностей, таких как горы, пещеры, волны, и т.д. [3].

Создаем проект консольного приложение и называем произвольным именем см. рисунок 1.

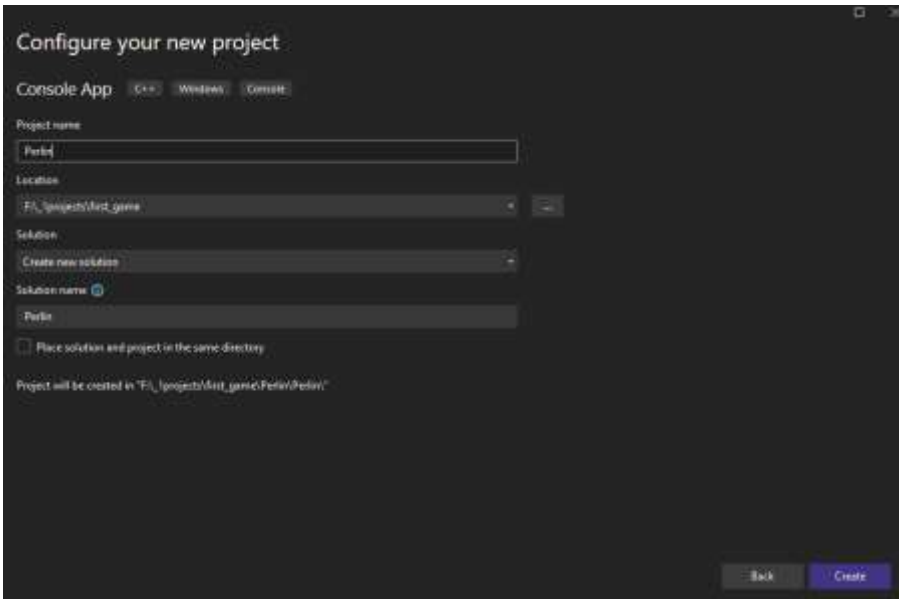


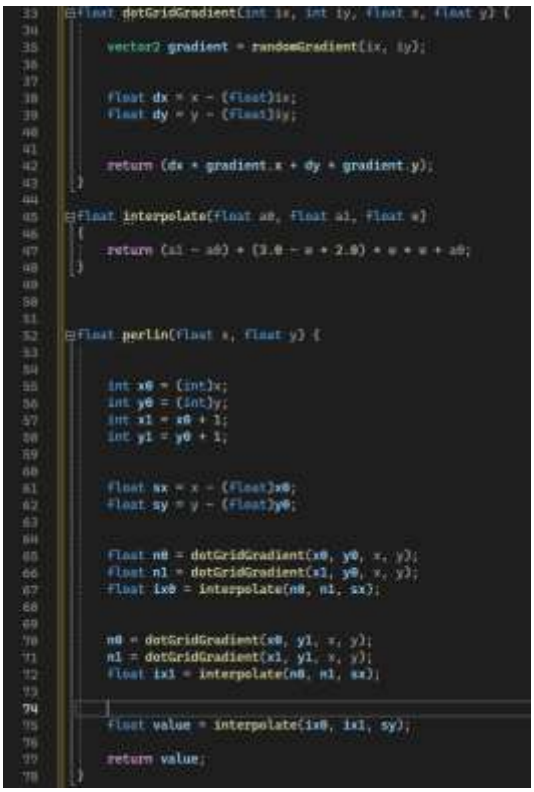
Рис. 1. Создание проекта

Сначала необходимо подключить библиотеки: math.h - содержит функции для работы с математическими операциями, SFML - для работы с графикой, iostream – для доступа к стандартным потокам ввода-вывода. Объявляем структуры данных vector2, которые представляют собой кортеж из

двух значений типа float. Функция randomGradient принимает два целых числа (ix, iy) в качестве аргументов и возвращает случайный вектор (sin(random)) с координатами x и y. Числа ix и iy используются для генерации псевдослучайного числа random с использованием алгоритма Xorshift. Это число затем преобразуется в значение синусоидальной функции, которое используется как координата x вектора. Координата y вектора устанавливается равной 1 см. рисунок 2.

Рис. 2. Подключение библиотек, функция randomGradient

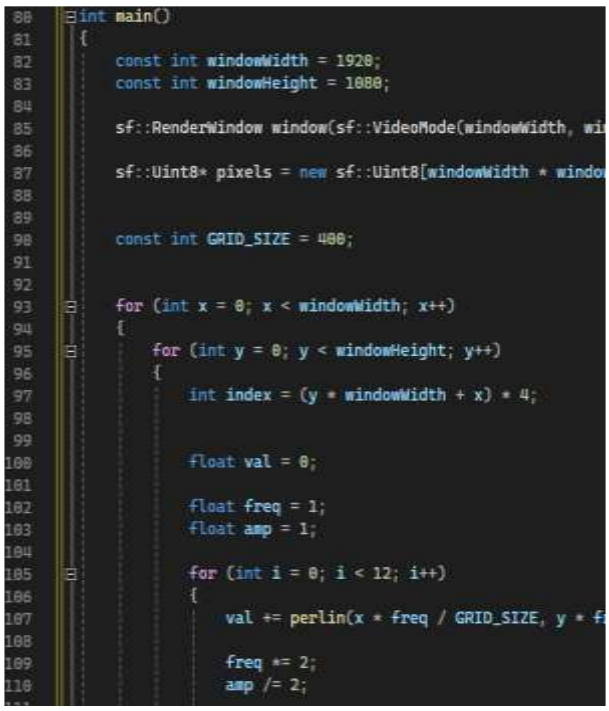
Переобъявляем структуру vector2, которая будет представлять кортежи из двух float-значений. Функция randomGradient возвращает случайный вектор на основе псевдослучайного числа, сгенерированного по алгоритму Xorshift, используя входные числа ix и iy. Функция dotGridGradient принимает четыре аргумента: координаты ячейки (ix, iy), координаты точки (x, y), и возвращает скалярное произведение вектора градиента и вектора от центра ячейки до точки. Функция interpolate принимает три аргумента: a0 и a1 - значения, которые нужно интерполировать, и w - вес интерполяции. Метод возвращает значение, интерполированное между a0 и a1 с учетом веса w. Функция perlin принимает два аргумента: x и y, и возвращает значение, рассчитанное с использованием интерполяции и градиента. Метод включает в себя структуры vector2 и функцию randomGradient, которая возвращает случайный градиент на основе входных координат. Функция dotGridGradient вычисляет скалярное произведение градиента и координат точки. Функция interpolate осуществляет линейную интерполяцию двух значений с заданным весом. Функция perlin производит вычисление значения шума Перлина с использованием градиентов и интерполяции см. рисунок 3.



```
float dotGridGradient(int ix, int iy, float x, float y) {

    vector2 gradient = randomGradient(ix, iy);


    float dx = x - (float)ix;
    float dy = y - (float)iy;


    return (dx * gradient.x + dy * gradient.y);
}

float interpolate(float a0, float a1, float w)
{
    return (a1 - a0) * (3.0 - w * 2.0) * w * w + a0;
}



float perlin(float x, float y) {


    int x0 = (int)x;
    int y0 = (int)y;
    int x1 = x0 + 1;
    int y1 = y0 + 1;


    float sx = x - (float)x0;
    float sy = y - (float)y0;


    float n0 = dotGridGradient(x0, y0, x, y);
    float n1 = dotGridGradient(x1, y0, x, y);
    float ix0 = interpolate(n0, n1, sx);


    n0 = dotGridGradient(x0, y1, x, y);
    n1 = dotGridGradient(x1, y1, x, y);
    float ix1 = interpolate(n0, n1, sx);


    float value = interpolate(ix0, ix1, sy);

    return value;
}
```

Рис. 3. Функция dotGridGradient, функция interpolate, функция perlin

Функция main создает окно SFML и заполняет его цветами, сгенерированными на основе шума Перлина. Основная часть кода содержит вложенный цикл, который проходит по всем пикселям окна и вычисляет цвет на основе шума Перлина. Функция "perlin" представляет собой специальную реализацию шума Перлина, этот код создает визуальный эффект, изменяя цвета пикселей в окне на основе шума Перлина см. рисунок 4.



```
int main()
{
    const int windowWidth = 1920;
    const int windowHeight = 1080;

    sf::RenderWindow window(sf::VideoMode(windowWidth, wi

    sf::Uint8* pixels = new sf::Uint8[windowWidth * windo


    const int GRID_SIZE = 480;


    for (int x = 0; x < windowWidth; x++)
    {
        for (int y = 0; y < windowHeight; y++)
        {
            int index = (y * windowWidth + x) * 4;


            float val = 0;

            float freq = 1;
            float amp = 1;

            for (int i = 0; i < 12; i++)
            {
                val += perlin(x * freq / GRID_SIZE, y * f

                freq *= 2;
                amp /= 2;
```

Рис. 4. Функция main

Далее используя математическую операция для умножения переменной "val" на значение 1.2, проверяем, больше ли значение переменной "val" 1.0 или меньше -1.0, и, если это так, оно устанавливается на 1.0 или -1.0 соответственно. Затем рассчитываем значение цвета (int "color") на основе значения "val". Затем этот цвет назначаем пикселям в массиве "pixels" на соответствующих позициях с индексом "index". После установки пикселей создаем текстуру SFML "texture" и спрайт "sprite". Текстура создается с размерами окна (windowWidth, windowHeight), после чего текстура обновляется пикселями из массива "pixels". Затем создаем цикл, который позволяет пользователю закрыть окно, текстура рисуется и отображается, а затем окно обновляется и отображает отображенные изменения.

```
115                 val *= 1.2;
116
117
118                 if (val > 1.0f)
119                     val = 1.0f;
120                 else if (val < -1.0f)
121                     val = -1.0f;
122
123                 int color = (int)(((val + 1.0f) * 0.5f) * 255);
124
125
126                 pixels[index] = color;
127                 pixels[index + 1] = color;
128                 pixels[index + 2] = color;
129                 pixels[index + 3] = 255;
130             }
131         }
132
133         sf::Texture texture;
134         sf::Sprite sprite;
135
136         texture.create(windowWidth, windowHeight);
137
138         texture.update(pixels);
139
140         sprite.setTexture(texture);
141
142         while (window.isOpen())
143         {
144             sf::Event event;
145             while (window.pollEvent(event))
146             {
147                 if (event.type == sf::Event::Closed)
148                     window.close();
149             }
150
151             window.clear();
152             window.draw(sprite);
153
154             window.display();
155         }
156
157         return 0;
158     }
```

Рис.5. Продолжение функции main

Далее проверяем алгоритм, запуском консольного приложения см. рисунок 6.

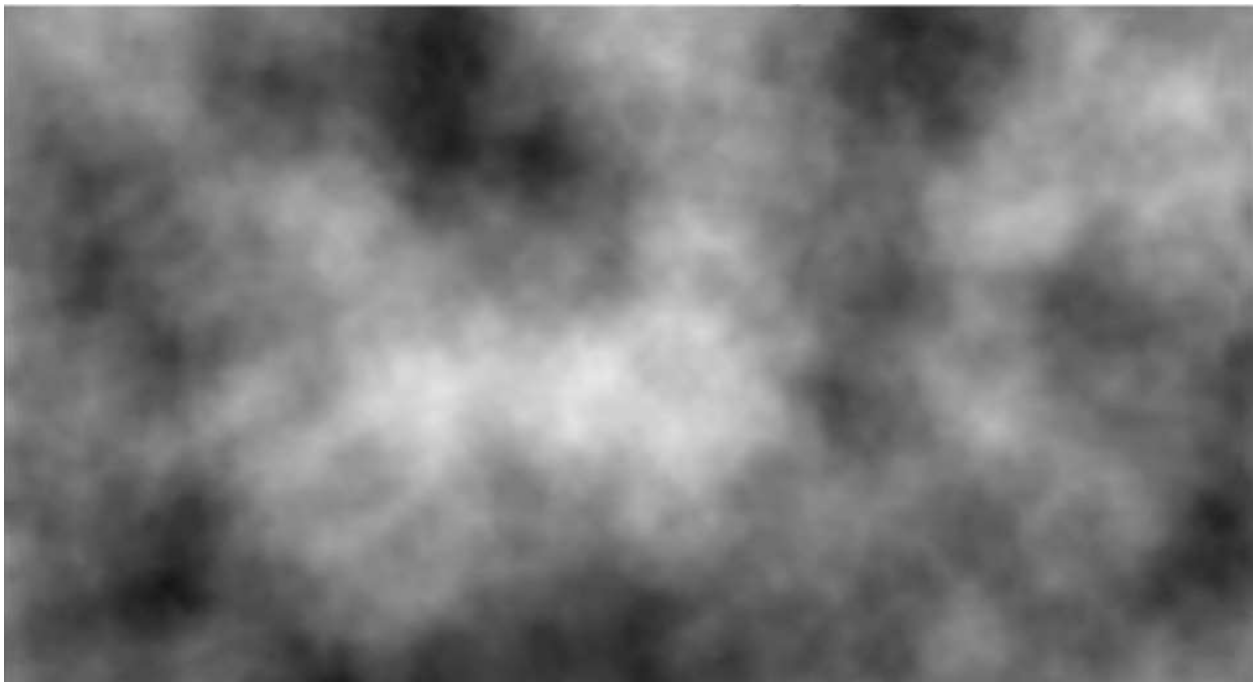


Рис. 6. Результат работы алгоритма

Были проанализированы существующие аналоги и методы разработки, а также выбрана среда разработки. Для реализации поставленной задачи отлично подошла разработка с помощью Visual Studio и языка программирования C++. Такой выбор заметно упростил разработку алгоритма, так как в интернете имеется достаточное кол-во документации. Во время создания шума Перлина был полученный ценный опыт работы с этим средством разработки.

В итоге был разработан алгоритм генерации шума Перлина. Созданный генератор имеет потенциал к развитию, а именно: добавление новых функций; добавление цветов шума.

Библиографический список

1. Грузин Н. А. Сравнение языков программирования C++ и Python //Modern Science. 2020. №. 2-1. С. 343-348.
2. Асанбаев К. Ф. и др. Язык программирования C++ //Фундаментальные и прикладные научные исследования: актуальные вопросы, достижения и инновации. 2020. С. 62-64.
3. Слеповичев С. О. Шум Перлина как способ получения компьютерных спецэффектов природных явлений //Инновационное развитие. 2017. №. 2. С. 32-33.