UDC 004.925.4(043.2)

R.O. Savelko
O.N. Romanyuk

PERLIN NOISE AND ITS DERIVATIVES AS A BASIS FOR PROCEDURAL TEXTURES

Vinnytsia National Technical University

Анотація

У тезах розглядається шум Перліна та його похідні як фундаментальна основа для синтезу процедурних текстур у комп'ютерній графіці. Описано математичний апарат класичного градієнтного шуму Перліна, алгоритм його обчислення та ключові властивості. Розглянуто вдосконалені варіанти – симплексний шум, хвильовий шум, а також метод дробового броунівського руху. Наведено порівняльний аналіз різновидів шуму за критеріями обчислювальної складності, артефактів та сфер застосування. Обговорено практичне використання шумових функцій у реальних задачах текстурування.

Ключові слова: шум Перліна, симплексний шум, процедурна текстура, градієнтний шум, дробовий броунівський рух, комп'ютерна графіка, хвильовий шум.

Abstract

This research paper examines Perlin noise and its derivatives as the fundamental basis for procedural texture synthesis in computer graphics. The mathematical formulation of classic gradient noise, its computation algorithm, and key properties are described. Improved variants – simplex noise, wavelet noise, and fractional Brownian motion – are discussed. A comparative analysis of noise variants is provided according to computational complexity, artefact characteristics, and application domains. The practical use of noise functions in real texturing tasks is also addressed.

Keywords: Perlin noise, simplex noise, procedural texture, gradient noise, fractional Brownian motion, computer graphics, wavelet noise.

Introduction

Procedural texturing stands as one of the most powerful paradigms in computer graphics, enabling the synthesis of complex surface appearances through mathematical algorithms rather than pre-recorded image data. At the heart of the overwhelming majority of procedural texture systems lies a family of algorithms rooted in a single foundational concept: noise – a pseudo-random continuous function whose statistical properties can be controlled to match the spectral characteristics of natural phenomena [1].

The modern era of procedural noise in computer graphics began in 1985 when Ken Perlin introduced a gradient noise function, subsequently named Perlin noise, in the context of the film Tron [2]. The algorithm addressed a critical shortcoming of earlier pseudo-random techniques: the ability to produce smooth, band-limited, and isotropic noise that could convincingly simulate clouds, terrain, fire, marble, and countless other natural textures. Perlin was awarded an Academy Award for Technical Achievement in 1997 in recognition of this contribution.

Since its introduction, Perlin noise has spawned a family of derivative algorithms – simplex noise, wavelet noise, sparse convolution noise, and others – each addressing specific limitations of the original formulation. Furthermore, by compositing multiple octaves of noise through the technique of fractional Brownian motion (fBm), artists and engineers can construct textures with the self-similar, multi-scale structure characteristic of many natural surfaces [3].

This research paper provides a systematic treatment of Perlin noise and its principal derivatives, examining both the mathematical foundations and the practical implications for procedural texture design. The goal is to furnish researchers and practitioners with a clear, comparative understanding of these tools as a foundation for software implementation.

Mathematical foundations of perlin noise

Classic Perlin noise is a gradient noise function defined on an n-dimensional integer lattice. The algorithm proceeds in three stages [2; 4]. First, the input coordinate vector x is located within the unit hypercube of the lattice; the integer lattice vertices of this hypercube are identified. Second, a pseudo-random gradient vector g

is assigned to each lattice vertex via a permutation table. Third, the dot product of the gradient vector and the displacement vector from each vertex to x is computed, and the results are interpolated using a smooth fade curve.

The fade (or smoothstep) function used in the improved version of Perlin noise [4] is the quintic polynomial:

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

This polynomial has zero first and second derivatives at t = 0 and t = 1, ensuring that both the gradient and the curvature of the noise function are continuous across lattice boundaries, eliminating the visible grid-aligned artefacts that affected the original cubic interpolant [4].

The noise value at a point x in two dimensions is computed as:

$$N(x, y) = lerp(lerp(d_{00}, d_{10}, u), lerp(d_{01}, d_{11}, u), v)$$

where $d_{ij} = g_{ij} \cdot (x - i, y - j)$ is the dot product of the gradient at lattice vertex (i, j) with the displacement vector, $u = f(x - \lfloor x \rfloor)$ and $v = f(y - \lfloor y \rfloor)$ are the interpolation weights, and lerp denotes linear interpolation. The function is designed to produce values in the range [−1, 1], with a statistical distribution concentrated around zero [1].

Simplex noise

Simplex noise was introduced by Perlin in 2001 as an improvement addressing three key limitations of the original algorithm [5]. First, the computational complexity of gradient noise scales as $O(2^n)$ with the number of dimensions n, since 2^n lattice vertices must be evaluated per sample. Simplex noise replaces the rectangular grid with a simplex lattice (e.g., triangles in 2D, tetrahedra in 3D), reducing the number of vertices to n + 1 per sample, giving $O(n^2)$ complexity.

Second, classic Perlin noise exhibits directional artefacts aligned with the axes of the rectangular grid. The simplex lattice has higher angular symmetry, reducing these artefacts significantly. Third, the gradient contribution function in simplex noise uses a radially symmetric kernel with compact support, ensuring that contributions from each vertex decay smoothly to zero before the next vertex begins to contribute, without requiring a fade function.

The simplex noise value at a point p is computed as the sum of contributions from the n + 1 vertices of the enclosing simplex:

$$N(p) = \Sigma_i \, max(0, r^2 - |p - v_i|^2)^4 \cdot (g_i \cdot (p - v_i))$$

where v_i are the simplex vertices, g_i are pseudo-random gradient vectors assigned to each vertex, and r^2 is the radius of the kernel support. The exponent 4 in the attenuation function ensures C^1 continuity of the noise function [5].

Wavelet noise

Wavelet noise, introduced by Cook and DeRose [6], addresses aliasing problems that arise when Perlin noise is used in high-quality rendering contexts. When a texture is viewed at distances where the sampling rate approaches or exceeds the Nyquist limit of the noise, high-frequency components alias and produce undesirable visual artefacts. Wavelet noise solves this by constructing a noise function that is explicitly band-limited: its energy is concentrated in a specific frequency band.

The construction projects a white noise signal onto the detail band of a wavelet decomposition, retaining only the mid-frequency content. The resulting function can then be evaluated analytically at any frequency band, enabling anti-aliased noise queries that correctly omit energy above the Nyquist frequency of the sampling context. Wavelet noise is particularly valuable in production rendering pipelines where aliasing control is critical [6].

Fractional Brownian motion and multiscale composition

A single octave of gradient noise has a well-defined characteristic frequency and therefore looks like a smooth, featureless bump pattern. Natural surfaces, however, are characterised by a continuous spectrum of spatial frequencies – their power spectral density follows a power law $P(f) \propto f^{-\beta}$, known as 1/f or pink noise statistics. Fractional Brownian motion (fBm) replicates this property by summing multiple octaves of a base noise function [3]:

$$fBm(x) = \Sigma_{i=0}^{n-1} A_i \cdot N(\lambda_i x)$$

where N is a base noise function, λ is the lacunarity (typically $\lambda = 2$, i.e. each successive octave doubles the frequency), A is the amplitude attenuation factor (typically $A = 0.5$, i.e. each octave has half the amplitude of the previous one), and n is the number of octaves summed. The Hurst exponent $H \in (0, 1)$ controls the fractal dimension of the result; for typical terrain synthesis, $H \approx 0.5–0.7$ is used [3; 7].

Turbulence is a variant in which the absolute value of each noise octave is taken before summation, introducing sharp ridges that simulate turbulent flow and fire. The ridge noise and ridged multifractal variants apply further non-linear transformations per octave to produce mountain ridge and eroded terrain appearances [7].

Table 1 – Comparison of gradient noise algorithm variants

Algorithm	Complexity (nD)	Continuity	Grid Artefacts	Primary Use
Classic Perlin (1985)	$O(2^n)$	C^1 (cubic)	Moderate	General textures
Improved Perlin (2002)	$O(2^n)$	C^2 (quintic)	Reduced	General textures
Simplex Noise (2001)	$O(n^2)$	C^1	Minimal	High-dim noise
Wavelet Noise (2005)	$O(2^n)$	C^1	Minimal	Anti-aliased rendering
fBm (composite)	$O(k \cdot 2^n)$	C^1	Inherited	Terrain, clouds, fire

Practical applications in procedural texture synthesis

The noise family described above serves as the building block for a remarkably broad range of procedural textures encountered in games, film, and visualisation. The following applications are particularly well-established [1; 7; 8].

Terrain and heightmap generation employs fBm with two to eight octaves of Perlin or simplex noise as a displacement function applied to a planar mesh. The lacunarity and Hurst exponent control the apparent geological character of the terrain, from smooth plains (H close to 1) to rugged, heavily eroded mountain ranges (H close to 0). Turbulence-based variants produce more irregular, cliff-like features.

Cloud and atmospheric simulation uses three-dimensional fBm noise to define an opacity or density field, which is then ray-marched from the camera. The characteristic low-frequency base shape with high-frequency detail structure maps naturally to the multi-octave fBm formulation. Domain warping – the technique of using one noise evaluation to distort the input coordinates of another – adds the swirling, self-referential complexity characteristic of real cumulus clouds [7].

Marble and stone material synthesis is achieved by using noise as a phase perturbation of a sine wave along one coordinate axis. The resulting pattern of alternating dark and light bands, smoothly distorted by the noise, faithfully reproduces the appearance of veined marble. Wood grain is produced by a similar technique with a radial distance function replacing the linear coordinate [8].

Water and fire animation exploits noise evaluated at time-varying coordinates: a three-dimensional noise function is evaluated at (x, y, t), where t is the animation time, producing a continuously evolving two-dimensional texture. The result is used as a normal map for water surface perturbation or as an opacity and emission mask for flame rendering.

Table 2 – Application domains of Perlin noise derivatives

Texture Type	Noise Variant	Key Parameters	Typical Octaves
Terrain heightmap	fBm (Perlin/Simplex)	$H = 0.5–0.7$, $\lambda = 2.0$	6–8
Clouds / atmosphere	fBm + domain warp	$H = 0.7$, $\lambda = 2.0$	4–6
Marble / stone	Perlin + sine warp	Frequency, amplitude	1–3
Fire / water animation	3D Perlin / Simplex	Time step Δt, scale	3–5
Surface normals (bump)	Improved Perlin	Derivative evaluation	2–4
Production rendering	Wavelet noise	Band, projection axis	1 (band-limited)

Discussion

The gradient noise family occupies a privileged position in procedural texture synthesis due to the favourable combination of properties it offers [1; 4]. The noise is smooth and band-limited, avoiding the visual roughness of white noise while retaining the aperiodicity necessary for large-scale texture generation. Its computation requires only table lookups, dot products, and polynomial interpolation, making it highly amenable to GPU parallelisation. The permutation table that underpins the pseudo-random gradient assignment provides a compact and deterministic random number generator that requires no external state.

However, the family is not without limitations. Classic Perlin noise exhibits a well-known bias: its histogram of output values is not uniform but instead follows a distribution concentrated near zero. This can cause procedural textures based on raw noise to appear washed-out, requiring remapping functions. The grid-aligned structure of the rectangular lattice introduces directional biases that are visible as diagonal streaking in some applications, a problem that simplex noise mitigates but does not entirely eliminate [5].

A practical concern for production use is the patent situation: the original simplex noise algorithm was patented by Perlin (US Patent 6,867,776), limiting its commercial use. Several alternative implementations that achieve similar properties – notably OpenSimplex noise and similar variants – have been developed to circumvent this restriction while providing comparable quality [9].

From a software engineering perspective, the improved Perlin noise algorithm [4] remains the most widely adopted baseline due to its simplicity of implementation, well-understood behaviour, and availability of the original reference code in the public domain. Simplex noise is preferred for high-dimensional applications and where isotropy is critical. Wavelet noise is the choice for offline rendering pipelines where aliasing must be controlled analytically [6].

Conclusions

This research paper has presented a systematic examination of Perlin noise and its principal derivatives – improved Perlin noise, simplex noise, wavelet noise, and the composite technique of fractional Brownian motion – as the foundational building blocks of procedural texture synthesis in computer graphics.

The mathematical analysis reveals that each variant represents a considered trade-off: improved Perlin noise eliminates second-order artefacts at modest additional cost; simplex noise reduces dimensionality scaling and grid bias; wavelet noise provides explicit frequency control; and fBm extends single-octave noise to multi-scale, fractal-like structures that faithfully replicate the power spectrum of natural surfaces.

The comparative analysis (Table 1) and application survey (Table 2) confirm that the choice of noise variant should be guided by the specific requirements of the rendering context: computational budget, dimension count, anti-aliasing requirements, and target appearance. For real-time GPU applications, improved Perlin or simplex noise with two to six fBm octaves provides an excellent balance of visual quality and performance. For production rendering, wavelet noise with analytic anti-aliasing is the preferred choice.

These findings directly inform the software implementation component of the broader research programme on procedural texture synthesis methods, which constitutes the motivation for this study. Future work will focus on GPU-optimised implementations of these noise variants and their integration into a parametric procedural texture authoring framework.

REFERENCES

1. Lagae A., Lefebvre S., Cook R., DeRose T., Drettakis G., Ebert D. S., Lewis J. P., Perlin K., Zwicker M. A survey of procedural noise functions. Computer Graphics Forum. 2010. Vol. 29, No. 8. P. 2579–2600.
2. Perlin K. An image synthesizer. ACM SIGGRAPH Computer Graphics. 1985. Vol. 19, No. 3. P. 287–296.
3. Mandelbrot B. B., Van Ness J. W. Fractional Brownian motions, fractional noises and applications. SIAM Review. 1968. Vol. 10, No. 4. P. 422–437.
4. Perlin K. Improving noise. ACM Transactions on Graphics. 2002. Vol. 21, No. 3. P. 681–682.
5. Perlin K. Noise hardware. Real-Time Shading SIGGRAPH Course Notes. 2001. P. 1–3.
6. Cook R. L., DeRose T. Wavelet noise. ACM Transactions on Graphics. 2005. Vol. 24, No. 3. P. 803–811.
7. Musgrave F. K. Procedural fractal terrains. In: Ebert D. S. et al. Texturing and Modeling: A Procedural Approach. 3rd ed. San Francisco: Morgan Kaufmann, 2003. P. 427–462.
8. Ebert D. S., Musgrave F. K., Peachey D., Perlin K., Worley S. Texturing and Modeling: A Procedural Approach. 3rd ed. San Francisco: Morgan Kaufmann, 2003. 693 p.
9. Gustavson S. Simplex noise demystified. Linköping University technical report. 2005. 11 p.

Савелко Ростислав Олегович – студент групи 1ПІ-22б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, Вінниця, e–mail: rostiksavelko@gmail.com.

Романюк Олександр Никифорович – доктор технічних наук, професор, професор кафедри програмного забезпечення, завідувач кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e–mail: rom8591@gmail.com.

Savelko Rostyslav O. – student of group 1PI-22b, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: rostiksavelko@gmail.com.

Romanyuk Oleksandr N. – Doctor of Technical Sciences, Professor, Professor of the Software Engineering Department, Head of the Software Engineering Department, Vinnytsia National Technical University, Vinnytsia, e–mail: rom8591@gmail.com.