

CLASSIFICATION OF PROCEDURAL TEXTURE GENERATION ALGORITHMS IN COMPUTER GRAPHICS

Vinnitsia National Technical University

Анотація

У тезах розглядається класифікація алгоритмів процедурної генерації текстур, що широко застосовуються в комп'ютерній графіці. Описано основні категорії алгоритмів: шумові функції, фрактальні методи, клітинні алгоритми та методи на основі фізичного моделювання. Для кожної категорії наведено ключові представники, характеристики та типові застосування. Проведено порівняльний аналіз алгоритмів за такими критеріями, як швидкість виконання, контрольованість, реалістичність і використання ресурсів. Стаття призначена для дослідників і практиків у галузі комп'ютерної графіки, що прагнуть систематизованого огляду сучасних методів процедурної текстуризації.

Ключові слова: процедурна текстура, шум Перліна, фрактал, клітинний шум, комп'ютерна графіка, алгоритм генерації, текстурний синтез.

Abstract

This research paper presents a classification of procedural texture generation algorithms widely used in computer graphics. The main categories of algorithms are described: noise functions, fractal methods, cellular algorithms, and physically-based simulation methods. For each category, the key representatives, characteristics, and typical applications are provided. A comparative analysis of the algorithms is carried out based on such criteria as execution speed, controllability, realism, and resource usage. This research paper is intended for researchers and practitioners in the field of computer graphics seeking a systematic overview of modern procedural texturing methods.

Keywords: procedural texture, Perlin noise, fractal, cellular noise, computer graphics, generation algorithm, texture synthesis.

Introduction

Procedural texturing is a fundamental technique in modern computer graphics that allows surfaces to be described not by pre-recorded image data, but by mathematical algorithms that generate texture information on demand. Unlike raster textures, which are bitmap images stored in memory, procedural textures are computed at runtime and can, in principle, provide infinite detail at any resolution and scale [1].

The relevance of procedural texture generation is driven by several key factors. First, the growing demands of real-time rendering in video games and virtual reality require lightweight assets that do not consume large amounts of memory. Second, procedural methods allow parametric control over appearance, making it easy to introduce variation and avoid repetitive tiling artifacts. Third, they are invaluable in film and animation production, where physically plausible material synthesis is essential for achieving photorealism [2].

Despite the wide variety of algorithms available, there is no universally accepted classification that systematises procedural texture methods according to their mathematical foundations, computational properties, and application domains. This research paper aims to fill this gap by proposing a structured taxonomy and comparative evaluation of the principal families of procedural texture generation algorithms.

Noise-Based Algorithms

Noise functions form the cornerstone of procedural texture generation. The pioneering work of Ken Perlin introduced a gradient noise function that produces smooth, pseudo-random patterns with well-controlled spectral properties [3]. Perlin noise is computed by interpolating between pseudo-random gradient vectors defined on a regular integer lattice, resulting in a continuous, band-limited signal that resembles natural phenomena such as clouds, fire, and water.

Simplex noise, also developed by Perlin, addresses several shortcomings of classic gradient noise by using a simplex lattice instead of a rectangular grid [4]. This reduces computational complexity from $O(2n)$ to $O(n^2)$.

for n-dimensional noise and eliminates directional artefacts. Value noise is a simpler variant that interpolates between pseudo-random scalar values rather than gradients, at the cost of lower visual quality. Wavelet noise addresses aliasing in band-limited signals for high-quality rendering [5].

Fractional Brownian Motion (fBm) is obtained by summing multiple octaves of a base noise function at increasing frequencies and decreasing amplitudes, yielding a signal with a power spectrum characteristic of natural fractal phenomena. The resulting textures convincingly represent terrain heightfields, turbulent flow, and weathered surfaces.

Fractal and Self-Similar Algorithms

Fractal geometry provides a natural framework for modelling self-similar structures that appear in nature. The Mandelbrot set and Julia sets, derived from iteration of complex quadratic maps, produce infinitely detailed boundary structures used in artistic procedural textures [6]. Iterated Function Systems (IFS) define self-similar structures through a set of contractive affine transformations; the attractor of the IFS serves as the texture pattern.

L-systems (Lindenmayer systems) offer a grammar-based approach to generating self-similar branching structures. While primarily used for plant modelling, they can be projected onto surfaces to produce organic-looking procedural textures. Diamond-Square and midpoint displacement algorithms are simple yet effective methods for generating fractal terrain heightmaps, which are then used as displacement or colour maps in texturing pipelines.

Cellular and Voronoi Algorithms

Cellular textures are based on the Voronoi diagram – a partition of space according to the nearest feature point from a set of seed points distributed in the domain [7]. Steven Worley introduced the cellular noise function (also known as Worley noise or Voronoi noise) in 1996, defining texture values in terms of the distances to the k-th nearest seed point. By combining distance metrics (Euclidean, Manhattan, Chebyshev) and combining F1, F2, and F2-F1 distance components, a rich variety of organic-looking patterns can be generated, including cracked earth, skin, scales, and stone tiles.

Wang tiles are a specific cellular construction in which square tiles are coloured on their edges such that adjacent tiles always match; this allows aperiodic tiling of arbitrarily large surfaces without visible repetition [8]. Reaction-diffusion systems, such as those described by Alan Turing, simulate the interaction of two chemical species and produce spot-and-stripe patterns reminiscent of animal skin markings.

Physically-Based and Simulation Methods

Physically-based texture synthesis methods simulate real physical or biological processes to generate realistic surface appearances. Fluid simulation can produce marble-like swirling patterns by advecting a colour field with a simulated velocity field. Erosion simulation applied to a heightmap produces geologically plausible terrain textures, including gullies, alluvial fans, and sediment deposits [9].

Subsurface scattering models generate translucent material appearances such as skin and wax by simulating light transport beneath the surface. Crystallisation and crystal growth models based on diffusion-limited aggregation (DLA) produce dendritic, snowflake-like patterns. These methods typically require significant computation and are more common in offline rendering contexts, though hardware acceleration is increasingly making them viable for real-time use.

Machine Learning-Based Approaches

Recent advances in deep learning have introduced a new family of procedural texture methods. Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) can learn the distribution of textures from example images and generate new samples from that distribution [10]. Neural texture synthesis methods, such as those exploiting the feature statistics of convolutional neural networks (CNNs), allow example-based synthesis of highly complex textures that are difficult to model analytically.

Implicit neural representations (INR), where a neural network is trained to map spatial coordinates to texture values, provide an emerging framework that can be interpreted as a learned procedural texture. While these methods currently lack the mathematical transparency and real-time performance of classical approaches, they are rapidly advancing and represent a significant frontier in procedural texture research.

Table 1 – Classification of procedural texture generation algorithm families

Category	Key Algorithms	Typical Application	Memory Usage
Noise-based	Perlin, Simplex, Value, Wavelet, fBm	Clouds, terrain, water, fire	Very low
Fractal / Self-similar	Mandelbrot, Julia, IFS, L-systems, Diamond-Square	Terrain heightmaps, organic patterns	Low
Cellular / Voronoi	Worley noise, Wang tiles, Reaction-diffusion	Stone, skin, scales, tiles	Low
Physically-based	Fluid simulation, erosion, DLA, SSS	Marble, terrain, skin, crystal	Medium–High
ML-based	GANs, VAEs, Neural Texture Synthesis, INR	Complex natural & synthetic textures	High

Table 2 – Comparative characteristics of procedural texture algorithm families

Category	Speed	Controllability	Realism	Real-time Feasible
Noise-based	Very High	High	Medium	Yes
Fractal	High	Medium	Medium–High	Yes
Cellular	High	Medium	High	Yes
Physically-based	Low	Low–Medium	Very High	Partially
ML-based	Medium	Low	Very High	Partially

Discussion

The comparative analysis presented in Table 2 reveals clear trade-offs between algorithm families that guide their selection in practice. Noise-based algorithms occupy the sweet spot between performance and versatility: they are trivially parallelisable on modern graphics processing units (GPUs), support arbitrary parameter variation, and are sufficiently flexible to represent a broad class of natural phenomena. Their primary limitation is that they are not inherently tied to physical laws, which restricts their realism for specific material classes [11].

Fractal methods extend the expressiveness of noise by introducing explicit scale invariance, making them the method of choice for terrain and geological feature synthesis. Their recursive or octave-summing structure maps well to GPU computation. Cellular methods excel in producing structured, organic-looking patterns with clear geometric identity and are particularly suited to stone, biological tissue, and tiled surface simulation [7].

Physically-based methods achieve the highest degree of realism because they are grounded in the same processes that generate the appearances being modelled. However, the computational cost of simulating fluid dynamics, erosion, or light transport at texture resolution is prohibitive for interactive applications without approximation. The trend towards GPU-accelerated simulation is steadily narrowing this performance gap [12].

Machine learning methods represent the newest and fastest-evolving category. Their ability to capture complex statistical structure from training data enables synthesis of textures that are otherwise analytically intractable. However, they introduce new challenges: lack of mathematical interpretability, sensitivity to training data, large model sizes, and limited extrapolation capability. Hybrid approaches that combine learned components with classical procedural building blocks are an active research direction [10].

Conclusions

This research paper has presented a structured classification of procedural texture generation algorithms in computer graphics, organised into five principal families: noise-based, fractal/self-similar, cellular/Voronoi, physically-based, and machine learning-based methods. Each family has been characterised in terms of its mathematical foundations, representative algorithms, and application domains.

The comparative analysis demonstrates that no single algorithm family is universally superior; rather, the choice of method should be guided by the specific requirements of the application with respect to rendering

speed, visual realism, parameter controllability, and available computational resources. Noise-based and cellular methods remain the dominant paradigm for real-time applications, while physically-based and ML-based approaches are increasingly adopted in high-quality offline rendering pipelines.

The proposed classification serves as a foundation for the development of software tools for procedural texture synthesis, which constitutes the core objective of the broader research programme within which this study is embedded. Future work will focus on implementing representative algorithms from each category, benchmarking their performance on modern GPU hardware, and investigating hybrid procedural-neural approaches for controllable, high-fidelity texture synthesis.

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. 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.
3. Perlin K. An image synthesizer. *ACM SIGGRAPH Computer Graphics*. 1985. Vol. 19, No. 3. P. 287–296.
4. Perlin K. Noise hardware. *Real-Time Shading SIGGRAPH Course Notes*. 2001. P. 1–3.
5. Cook R. L., DeRose T. Wavelet noise. *ACM Transactions on Graphics*. 2005. Vol. 24, No. 3. P. 803–811.
6. Mandelbrot B. B. *The Fractal Geometry of Nature*. New York: W. H. Freeman, 1982. 460 p.
7. Worley S. A cellular texture basis function. *Proceedings of ACM SIGGRAPH 1996*. New York: ACM, 1996. P. 291–294.
8. Cohen M. F., Shade J., Hiller S., Deussen O. Wang tiles for image and texture generation. *ACM Transactions on Graphics*. 2003. Vol. 22, No. 3. P. 287–294.
9. Benes B., Forsbach R. Layered data representation for visual simulation of terrain erosion. *Proceedings of SCCG 2001*. Los Alamitos: IEEE, 2001. P. 80–86.
10. Zhou Y., Zhu J.-Y., Bautista E., Martin-Brualla R., Srinivasan P. P., Isola P., Freeman W. T. Non-stationary texture synthesis by adversarial expansion. *ACM Transactions on Graphics*. 2018. Vol. 37, No. 4. Research paper 75.
11. Lewis J. P. Algorithms for solid noise synthesis. *ACM SIGGRAPH Computer Graphics*. 1989. Vol. 23, No. 3. P. 263–270.
12. Stam J. Stable fluids. *Proceedings of ACM SIGGRAPH 1999*. New York: ACM, 1999. P. 121–128.

Савелко Ростислав Олегович – студент групи ІІІ-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.

Кот Сергій Олександрович – кандидат філологічних наук, доцент кафедри іноземних мов, Вінницький національний технічний університет, м. Вінниця, e-mail: kot.sergii@vntu.edu.ua.

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

Oleksandr N. Romanyuk – 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.

Sergii O. Kot – Candidate of Philological Sciences, Associate Professor of the Foreign Languages Department, Vinnytsia National Technical University, Vinnytsia, e-mail: kot.sergii@vntu.edu.ua.