# simplex∼: Exploring Musical Trajectories in Simplex Noise Space

**Benjamin Wesch**
Tangible Music Lab, University of Art and Design Linz, Linz, Austria
benjamin.wesch@kunstuni-linz.at

## ABSTRACT

*This paper introduces a novel approach for utilizing Simplex noise in musical exploration via a Pure Data (Pd) external object. While previous applications of gradient noise in sound synthesis have focused on sampling from one-dimensional noise spaces, we present a framework for navigating multi-dimensional noise spaces through carefully designed trajectories. Our implementation, based on Stefan Gustavson's optimized algorithm, enables exploration across multiple time scales — from micro-level sound synthesis to macro-level compositional patterns. The object provides variable octaves, persistence control, and derivative outputs, creating new possibilities for both precise timbral control and emergent musical structures.*

## 1. INTRODUCTION

Pure Data (Pd) [1] is a widely-used visual programming environment for audio processing. While Pd includes random number generators and noise-related externals, to our knowledge, it lacks a dedicated object for sampling from coherent gradient noise spaces like Simplex noise, which offers unique properties for musical applications. This paper introduces a new external object implementing Simplex noise, providing smooth, continuous random values suitable for parameter control and sound synthesis.

Our work treats multi-dimensional noise spaces as environments to be navigated through carefully designed trajectories rather than as simple waveform generators, enabling both complex sonic patterns and broader musical applications across different time scales.

## 2. RELATED WORK

While gradient noise has been extensively used in visual contexts, its application to sound synthesis represents a relatively unexplored area with significant potential. Popov [2] demonstrated the use of 1D Perlin noise and fractional Brownian motion for generating pitched tones with distinctive digital timbres, implementing these techniques in a VST/AU plugin called Andes. His work focused specifically on one-dimensional noise for direct waveform synthesis, choosing to concentrate on this dimension as most relevant for his sound synthesis goals.

## 3. BACKGROUND

### 3.1 Historical Context

Ken Perlin's gradient noise algorithm [3] revolutionized procedural content generation in computer graphics. The algorithm was equally groundbreaking for both terrain generation and texture synthesis, enabling the creation of natural-looking landscapes as well as organic textures like clouds and fire. The algorithm earned Perlin an Academy Award for Technical Achievement and has since become a fundamental tool in computer graphics. He later improved his original algorithm with Simplex noise [4], which addressed computational inefficiencies and directional artifacts in the original method. Gustavson further optimized the algorithm [5] and provided accessible implementations that have been widely adopted across various fields.

### 3.2 Characteristics of Gradient Noise

Unlike traditional random number generators, gradient noise functions like Perlin noise and Simplex noise have several key properties that make them suitable for both visual and audio applications:

- **Determinism:** Given the same input coordinates, the noise function always produces the same output value, allowing for predictable and reproducible results
- **Continuity:** The function produces smoothly varying values with no sudden jumps, creating natural-looking and natural-sounding variations

## 4. IMPLEMENTATION

### 4.1 Development Process

The development of this external object began with an exploration of noise algorithms using vanilla Pure Data objects. Initial prototypes included implementations of 1D, 2D, and 3D Perlin noise (Figure 1) as well as a 3D Simplex noise patch (Figure 2), both created entirely with vanilla Pd objects.

While these vanilla implementations worked well as proof-of-concept, a compiled version was preferred to avoid performance concerns when using multiple instances with many octaves for practical musical use. The core algorithm is based on Gustavson's optimized Simplex noise implementation [1], which supports up to 4D noise and was adapted for the Pure Data environment with additional features specifically designed for musical applications.

[1] https://github.com/stegu/perlin-noise

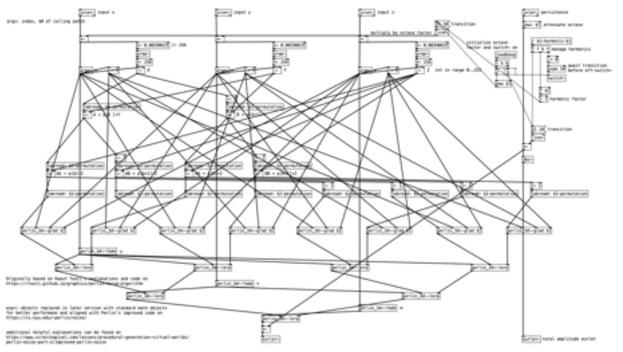


**Figure 1**. First attempt at a 3D Perlin noise patch in vanilla Pd, created based on Perlin's reference implementation

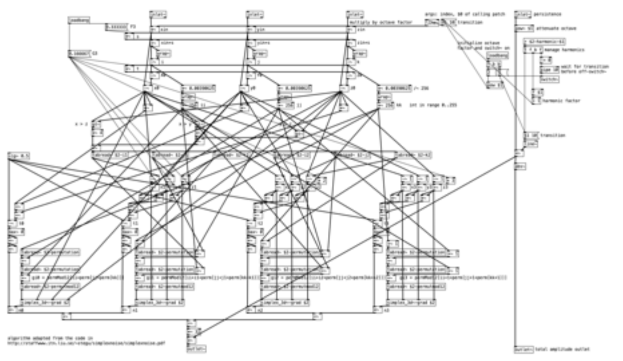


**Figure 2**. Vanilla Pure Data implementation of 3D Simplex noise

The resulting external is available on GitHub [2] and can be installed directly through Deken, Pure Data's package manager. This ensures easy accessibility for the Pd community and encourages further experimentation with gradient noise-based synthesis techniques.

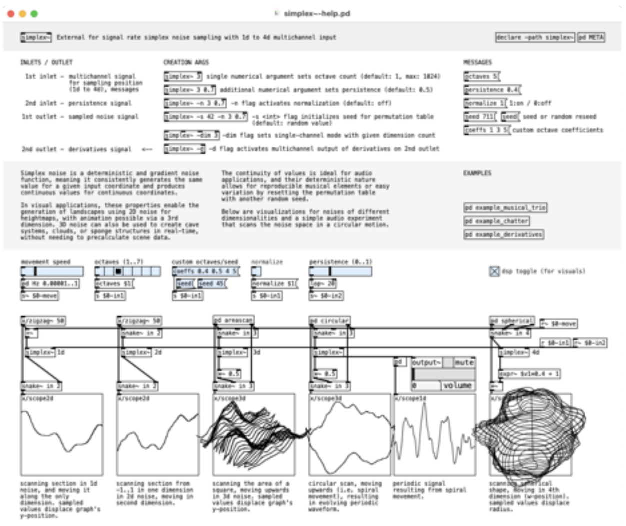
## 4.2 Object Design

The object automatically derives the noise's dimensionality based on the multichannel input signal, supporting 1D to 4D noise spaces. It is designed to facilitate the use of noise as a multi-dimensional space to be navigated.

Flexible control is provided through multiple inlets, outlets, and creation arguments:

- Primary inlet: Accepts up to 4 dimensions of input coordinates (as multichannel signal)
- Secondary inlet: Signal rate control of persistence value (relevant for multiple octaves)
- Main outlet: Outputs the noise value as a signal
- Optional derivative outlet: When enabled with the `-d` flag, outputs the derivatives of the noise function

[2] `https://github.com/ben-wes/pd-simplex`



```
simplex~--help.pd

simplex~  External for signal rate simplex noise sampling with 1d to 4d multichannel input     declare -path simplex~  pd META

INLETS / OUTLET                         CREATION ARGS                                                                          MESSAGES

 1st inlet - multichannel signal       simplex~ 3  single numerical argument sets octave count (default: 1, max: 1024)          octaves 5
             for sampling position     simplex~ 3 0.7  additional numerical argument sets persistence (default: 0.5)             persistence 0.4
             (1d to 4d), messages
 2nd inlet - persistence signal        simplex~ -n 3 0.7  -n flag activates normalization (default: off)                         normalize 1  1:on / 0:off
 1st outlet - sampled noise signal     simplex~ -s 42 -n 3 0.7  -s <int> flag initializes seed for permutation table            seed 711  seed  seed or random reseed
                                                                (default: random value)                                          coeffs 1 3 5  custom octave coefficients
                                       simplex~ -dim 3  -dim flag sets single-channel mode with given dimension count
 2nd outlet - derivatives signal  <--  simplex~ -d  -d flag activates multichannel output of derivatives on 2nd outlet

Simplex noise is a deterministic and gradient noise    The continuity of values is ideal for audio            EXAMPLES
function, meaning it consistently generates the same   applications, and their deterministic nature
value for a given input coordinate and produces        allows for reproducible musical elements or easy
continuous values for continuous coordinates.          variation by resetting the permutation table
                                                       with another random seed.                              pd example_musical_trio
In visual applications, these properties enable the                                                           pd example_chatter
generation of landscapes using 2D noise for            Below are visualizations for noises of different       pd example_derivatives
heightmaps, with animation possible via a 3rd          dimensionalities and a simple audio experiment
dimension. 3D noise can also be used to create cave    that scans the noise space in a circular motion.
systems, clouds, or sponge structures in real-time,
without needing to precalculate scene data.

movement speed       octaves (1..7)   custom octaves/seed      normalize        persistence (0..1)               dsp toggle (for visuals)
pd Hz 0.00001..1     octaves $1       coeffs 0.4 0.5 4 5       normalize $1     lop~ 20
s~ $0-move           s $0-in1         seed  seed 45            s $0-in1         s~ $0-in2
                                      s $0-in1

scanning section in 1d   scanning section from    scanning the area of a   circular scan, moving    periodic signal          scanning spherical
noise, and moving it     -1..1 in one dimension   square, moving upwards   upwards (i.e. spiral     resulting from spiral    shape, moving in 4th
along the only           in 2d noise, moving in   in 3d noise, sampled     movement), resulting     movement.                dimension (w-position).
dimension. sampled       second dimension.        values displace graph's  in evolving periodic                              sampled values displace
values displace graph's                           y-position.              waveform.                                         radius.
y-position.
```

**Figure 3**. Screenshot of the Simplex noise external help file in Pure Data, showing outputs for sampling from different dimensions.

Creation arguments and messages:

- `-n` flag: Normalizes output range to ensure it stays within -1 to 1
- `-d` flag: Enables the second outlet for derivatives output
- `seed` message or `-s` flag: Sets a specific seed value for reproducible results or reseeds randomly when no argument is provided
- `coeffs` message: Allows custom scaling of octaves beyond the default powers of 2

## 5. MUSICAL APPLICATIONS

### 5.1 Trajectories in Noise Space

N-dimensional noise space can be conceptualized as a continuous random field that can be "sampled" by moving through it along defined trajectories. Since the noise function is deterministic, following the same path through this space always yields the same sequence of values. In Gustavson's implementation, the noise space repeats every 768 units in one and three dimensions, which creates repeating patterns when trajectories exceed these boundaries.

Gradient noise itself is not inherently musical — it's the strategic traversal through the noise space that creates musically meaningful structures. To generate pitched sounds, we must sample the noise space periodically, with the sampling frequency determining the fundamental pitch. This represents a departure from traditional synthesis where oscillators sample from simple periodic functions.

While previous work by Popov [2] focused specifically on one-dimensional noise for direct waveform generation, our approach treats noise as a multi-dimensional space to be navigated. A popular technique in visual arts is to sample values with a circular movement in noise space to create seamless looping patterns [6].

We adapted this concept for musical applications:

$$\begin{aligned} x(t) &= r\cos(2\pi f t) \\ y(t) &= r\sin(2\pi f t) \end{aligned} \tag{1}$$

Where $r$ is the radius and $f$ determines the fundamental frequency of the resulting waveform. Figure 4 shows two cycles of such a circular movement through a noise space with three octaves (persistence 0.5, resulting in coefficients of 1, 0.5, and 0.25). While the combined octaves create a complex waveform characteristic of fractional Brownian motion, a pure circular trajectory in 2D noise would still produce the same pattern in each cycle, resulting in a static timbre.

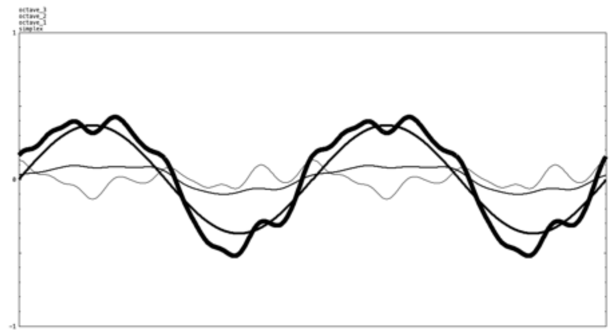


**Figure 4**. Visualization of combined octaves in noise output. The layered graphs show individual octave contributions (thin lines) and their sum (thickest line).

## 5.2 Multi-Dimensional Trajectories for Timbral Evolution

By extending the trajectory to higher dimensions, we can achieve more complex and evolving timbres. A spiral trajectory in three dimensions introduces variation over time while maintaining periodicity:

$$\begin{aligned} x(t) &= r\cos(2\pi f t) \\ y(t) &= r\sin(2\pi f t) \\ z(t) &= vt \end{aligned} \tag{2}$$

Here, the additional dimension $z(t)$ creates a slow evolution of the timbre over time as the trajectory moves through different "slices" of the noise space. The parameter $v$ controls the rate of timbral evolution. Figure 5 shows a basic Pure Data patch implementing this spiral movement.

The spectrograms in Figure 6 display frequencies up to 20kHz on a logarithmic scale, using the same seed across all examples. The baseline column shows single-octave noise sampled with a circular trajectory at 440Hz with radius 0.1, while other columns demonstrate the effect of adding higher octaves with different parameter combinations.

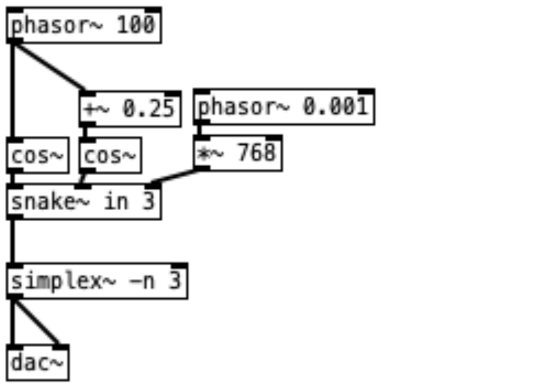


**Figure 5**. A basic Pure Data patch demonstrating spiral trajectory sampling in 3D noise space.

## 5.3 Multi-Scale Applications

The versatility of noise trajectories allows for application across multiple time scales:

- **Micro level (audio rate):** Generating waveforms with unique and evolving timbres
- **Meso level (control rate):** Creating modulation sources for parameters like filter cutoff, amplitude, or spatial position
- **Macro level (compositional):** Generating melodic and rhythmic patterns through low-frequency sampling and threshold detection

Figure 7 demonstrates melodic pattern generation through quantization to a major scale. Using a closed circular trajectory results in a repeating melodic pattern (a), while adding an offset to the circular trajectory creates an evolving melody that maintains coherent musical phrases (b).

Nested modulation chains extend these techniques: a slow-moving trajectory can modulate parameters of another trajectory used for audio synthesis, combining periodic and non-periodic elements into evolving timbral structures.

Multiple channels of coherent output can be achieved either through using multiple instances with different seeds or offset coordinates or through the derivative output, which efficiently provides additional dimensions of coherent variation from a single trajectory.

Beyond the circular and spiral trajectories described above, any closed trajectory can create repeating patterns and waveforms. With 4D noise, this extends to 3D trajectories moving through the fourth dimension, similar to how a circular motion along a third dimension creates a spiral trajectory.

## 6. EVALUATION

Initial performance testing shows that the external version significantly outperforms equivalent vanilla Pd patches, especially for higher-dimensional noise and multiple octaves. On a typical modern system, more than 1000 octaves of 3D noise at audio rate can be processed without buffer underruns, demonstrating its suitability for real-time applications.

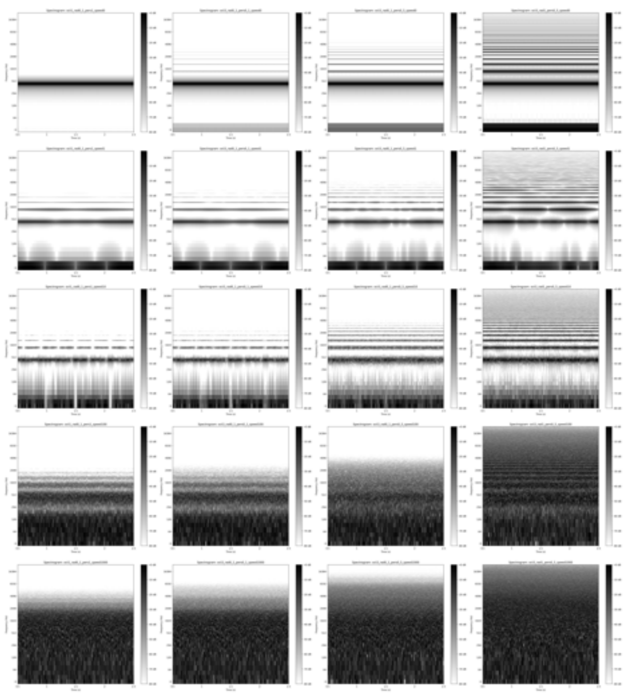


**Figure 6**. Spectrograms showing different parameter combinations of simplex noise output. Rows show increasing vertical movement speeds (0-1000 units/s), columns compare octave count, persistence, and radius variations.

## 7. CONCLUSIONS

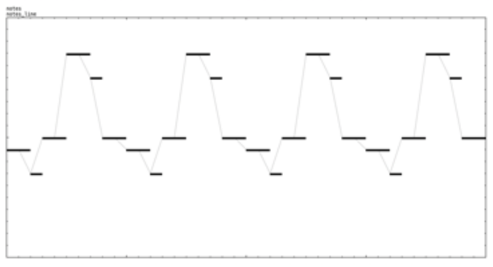
This paper has described an implementation of Simplex noise as a Pure Data external object, building upon previous work on gradient noise in musical applications. While earlier approaches like Popov's explored one-dimensional noise for waveform generation, our work investigates the potential of multi-dimensional exploration through trajectory-based sampling, offering possibilities for various musical applications across different time scales.

The combination of multi-dimensional sampling, variable octaves and derivative output provides musicians with a versatile tool, where coherent noise can serve different roles from sound synthesis to compositional aid.

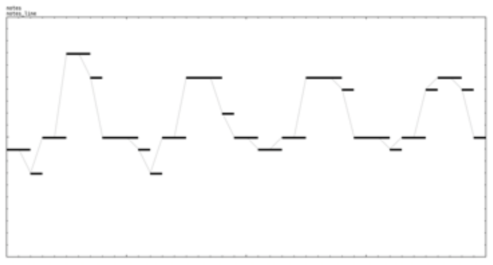
Additional areas for future research include extended trajectory designs, possibly higher dimensionalities and applications in spatial audio - primarily in the context of experiments with spatial synthesis, which is the author's current research focus, and where simplex noise itself can be used to create spatial trajectories.

**Acknowledgments**

Thanks to Miller Puckette for developing Pure Data, to Ken Perlin for the Simplex noise algorithm, and special thanks to Stefan Gustavson for his optimized implementation of the Simplex noise algorithms and his support and feedback on the audio applications presented in this paper.



(a) Repeating melody



(b) Evolving melody

**Figure 7**. Melodic patterns generated by quantizing noise values to a major scale

## 8. REFERENCES

[1] M. Puckette, "Pure Data: Recent Progress," in *Proceedings of the International Computer Music Conference*, 1997, pp. 224–227.

[2] A. Popov, "Using Perlin noise in sound synthesis," in *Proceedings of the Linux Audio Conference 2018*, 2018. [Online]. Available: https://lac.linuxaudio.org/2018/pdf/14-paper.pdf

[3] K. Perlin, "An Image Synthesizer," *ACM SIGGRAPH Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985. [Online]. Available: https://doi.org/10.1145/325334.325247

[4] ——, "Noise Hardware," in *Course Notes from Siggraph 2002, Course 36: Real-Time Shading Languages*, 2002, also published as "Improving Noise" in ACM Transactions on Graphics, Vol. 21, No. 3, pp. 681–682. [Online]. Available: https://www.csee.umbc.edu/~olano/s2002c36/ch02.pdf

[5] S. Gustavson, "Simplex Noise Demystified," 2005, self-published tutorial. [Online]. Available: https://www.researchgate.net/publication/216813608_Simplex_noise_demystified

[6] Necessary Disorder, "Drawing from noise, and then making animated loopy GIFs from there," Blog post, 2017, accessed: 2025. [Online]. Available: https://necessarydisorder.wordpress.com/2017/11/15/