Monte-Carlo Pre-Roll

Complex Phase with i

Elias, M. (2025)

Abstract

`preroll.py` generates a vivid pre-roll intro (4–16 s) preceding a musical drop. The sound arises as a sum of complex, exponentially decaying oscillations with minimal frequency detuning (beating), a 1/f noise texture (pink noise), slow phase diffusion, and smooth stereo width modulation. All computations are performed explicitly in the complex plane (Python: $1j \equiv i$). The result is a stereo WAV file, ready to drop into your DAW.

CAUTION

Deterministic modeling can cause unnatural distortions and algorithmically triggered responses. Independent safety and risk management strategies are essential.

1 Core Idea & Equations

1.1 Complex Features

The raw signal is generated as the sum of K beating pairs:

$$z(t) = \sum_{k=1}^{K} A_k \, e^{-(t/\tau_k)} \left(e^{i(2\pi(f_k-\Delta_k)t+\phi_k^1+\psi_k^1(t))} + e^{i(2\pi(f_k+\Delta_k)t+\phi_k^2+\psi_k^2(t))} \right)$$

- A_k: start amplitude (dB→linear), τ_k: decay time,
- f_k: base frequency, Δ_k: small offset → beating,
- $\phi_k^{1,2}$: initial phases, $\psi_k^{1,2}(t)$: phase diffusion (random walk).

Stereo channels receive a mini phase offset:

$$z_L \leftarrow z\, e^{+i\theta_k}, \quad z_R \leftarrow z\, e^{-i\theta_k}$$

Output: $x_L = \Re\{z_L\}$, $x_R = \Re\{z_R\}$.

1.2 Pink Noise (1/f)

Generated in the frequency domain:

$$Y(f) = \frac{\mathcal{F}\{W\}(f)}{\sqrt{\max(f,\varepsilon)}} \quad \Longrightarrow \quad y(t) = \mathcal{F}^{-1}\{Y(f)\}$$

where W is white noise and $\varepsilon \ll 1$ protects the DC component.

1.3 Complexity Gate

$$\mathrm{incoh} = |e_L - e_R|, \quad g(t) = 1 - \alpha\,\mathrm{incoh}(t), \quad x_{L/R} \leftarrow g(t)\,x_{L/R}$$

α controls damping strength when stereo envelopes diverge.

1.4 Master Envelope & Saturation

$$w(t) = t^2(3 - 2t), \quad y \leftarrow \tanh(\mathrm{drive} \cdot y), \quad \mathrm{Normalize}_{\mathrm{peak}}(y)$$

2 Pipeline (High-Level)

1. Complex bank: sum $z(t) \to x_L, x_R$
2. Whoosh layer: pink noise + pitch glide
3. Mix: $x \leftarrow x + \mathrm{whoosh}$
4. Apply complexity gate
5. Musical envelope: smooth riser to 100% at drop
6. Fades, softclip, normalize
7. Export WAV (16-bit PCM)

3 CLI Examples

```
# 12 s, 48 kHz, EDM riser
python mc_preroll_i.py --outfile preroll.wav --seconds 12 --bpm 128 --sr 48000 --seed 42

# brighter shimmer
python mc_preroll_i.py --seconds 10 --pairs 9 --fmin 300 --fmax 6000 --beatmin 1.2 --beatmax 3.0

# darker pull
python mc_preroll_i.py --seconds 16 --pairs 7 --fmin 40 --fmax 800 --beatmin 0.2 --beatmax 1.0
```

4 Parameter Guide (Practical)

Group	Field	Effect	Typical Values
Length/Tempo	seconds, bpm	Riser duration & drop timing	8–16 s, 120–140 BPM
Complex bank	pairs	Density/complexity	5–9
	f_min, f_max	Timbre	60–3000 Hz
	beat_hz_range	Beat frequency	0.3–2.5 Hz
	tau_range	Decay-time spectrum	0.2–3.0 s
	amp_db_range	Pair dynamics	22 to 6 dB
	phase_diffuse_strength	"Organic/lively"	0.6–1.0
	stereo_phase_max	Width (subtle)	0.10–0.25 rad
Whoosh	pink_db	Noise layer loudness	24 to 12 dB
	riser_octaves	Pitch glide	1–3 oct
Master	gate_strength	Incoherence damping	0.10–0.25
	fade_in/out	Click avoidance	15–60 ms
	drive	Saturation	1.1–1.6
	headroom_db	Export headroom	0.8–1.5 dB

5 Audio Quality & Checks

- Peak 1.0 dBFS
- DC offset $< 10^{-3}$
- Stereo correlation: 0.1–0.9
- RMS energy increasing over time
- No NaNs/Infs

```
import numpy as np, soundfile as sf
y, sr = sf.read("preroll.wav")
assert np.isfinite(y).all()
assert np.max(np.abs(y)) <= 1.0
assert abs(y.mean(axis=0)).max() < 1e-3
corr = np.corrcoef(y.T)[0,1]
print("stereo␣corr:", corr)
```

6 Reproducibility & Performance

- Determinism: set `-seed` (NumPy PCG64)
- Complexity: $O(KN)$, pink noise: $O(N \log N)$
- RAM: stereo float32 $8N$ bytes
- Example: 16s @ 48kHz → N=768,000 → 6 MB

7 Presets

- **EDM Neutral:** 12s, 6 pairs, 60–3000Hz, beat 0.3–2.5Hz
- **Cinematic Warm:** 16s, 7 pairs, 40–1200Hz
- **Airy Techno:** 10s, 9 pairs, 300–6000Hz, pink_db = 18

8 DAW Integration

1. Render WAV, import into DAW
2. Sidechain to kick ($\frac{1}{4}$ notes)
3. Gentle EQ/saturation (2 dB @ 10 kHz, low-cut @ 30 Hz)
4. Time-stretch if BPM mismatch

9 Troubleshooting

- Too sharp → lower fmax, drive, or pink_db 3 dB
- Too muddy → fewer pairs, reduce phase diffusion
- Mono issues → stereo_phase_max 0.2
- Clipping → increase headroom_db

10 Mini Validation (Unit-ish)

```
def test_basic_shape():
    y = render_preroll(PreRollCfg(seconds=4.0, sr=48000, seed=7))
    assert y.ndim == 2 and y.shape[1] == 2
    assert np.isfinite(y).all()

def test_no_clipping_dc():
    y = render_preroll(PreRollCfg(seconds=4.0, sr=48000, seed=1))
    assert np.max(np.abs(y)) <= 1.0 + 1e-7
    assert abs(y.mean(axis=0)).max() < 1e-3
```

Author: Martha Elias
Version: v1.0 (October 2025)
`marthaelias [at] protonmail [dot] com`