Monte-Carlo Pre-Roll

Complex Phase with i

Elias, M. (2025)

Abstract

`preroll.py` erzeugt ein lebendiges Pre-Roll-Intro (4–16 s) vor einem musikalischen Drop. Der Klang entsteht als Summe komplexer, exponentiell abklingender Schwingungen mit minimalen Frequenzversätzen (Schwebungen), 1/f-Rauschtextur (Pink Noise), langsamer Phasen-Diffusion und sanfter Stereo-Breitensteuerung. Alles wird explizit in der komplexen Ebene gerechnet (Python: $1j \equiv i$). Das Resultat ist ein zweikanaliges WAV, ready-to-drop in deiner DAW.

CAUTION

Deterministic modeling is vulnerable to unnatural distortions and algorithmically triggered reactions. Independent safety and risk management strategies are essential.

1 Kernidee & Formeln

1.1 Komplexe Features

Das Rohsignal entsteht als Summe aus K Schwebungspaaren:

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

- A_k: Start-Amplitude (dB→Linear), τ_k: Abklingzeit,
- f_k: Grundfrequenz, Δ_k: kleiner Versatz → Schwebung,
- $\phi_k^{1,2}$: Startphasen, $\psi_k^{1,2}(t)$: Phasen-Diffusion (Random-Walk).

  Stereo:

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

Audio-Ausgabe: $x_L = \Re\{z_L\}$, $x_R = \Re\{z_R\}$.

1.2 Pink Noise (1/f)

Erzeugung im Frequenzraum:

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

mit W = weißem Rauschen und $\varepsilon \ll 1$ zum Schutz des DC-Anteils.

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 steuert die Dämpfungsstärke bei Inkohärenz (sanftes Stereo-Gating).

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: Summe $z(t) \to x_L, x_R$
2. Whoosh-Layer: Pink-Noise + Pitch-Glide
3. Mischen: $x \leftarrow x + \mathrm{whoosh}$
4. Complexity-Gate anwenden
5. Musikalische Hüllkurve: Smooth-Riser
6. Fades, Softclip, Normalize
7. WAV-Export (16-bit PCM)

3 CLI & Minimalbeispiele

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

# hellerer "Schimmer"
python mc_preroll_i.py --seconds 10 --pairs 9 --fmin 300 --fmax 6000 --beatmin 1.2 --beatmax 3.0

# dunkler "Sog"
python mc_preroll_i.py --seconds 16 --pairs 7 --fmin 40 --fmax 800 --beatmin 0.2 --beatmax 1.0
```

4 Parameter-Leitfaden (Praxis)

Gruppe	Feld	Wirkung	Praxiswerte
Länge/Tempo	seconds, bpm	Riser-Dauer & Drop-Timing	8–16 s, 120–140 BPM
Komplexbank	pairs	Dichte/Komplexität	5–9
	f_min, f_max	Timbre	60–3000 Hz
	beat_hz_range	Schwebungsgeschw.	0.3–2.5 Hz
	tau_range	Abklingzeit-Spektrum	0.2–3.0 s
	amp_db_range	Dynamik je Paar	22 bis 6 dB
	phase_diffuse_strength	„Organisch/lebendig“	0.6–1.0
	stereo_phase_max	Breite (dezent!)	0.10–0.25 rad
Whoosh	pink_db	Lautheit Noise-Layer	24 bis 12 dB
	riser_octaves	Pitch-Glide	1–3 Oct
Master	gate_strength	Dämpfung Inkohärenz	0.10–0.25
	fade_in/out	Klickfreiheit	15–60 ms
	drive	Sättigung	1.1–1.6
	headroom_db	Export-Headroom	0.8–1.5 dB

5 Audio-Qualität & Checks

- Peak 1.0 dBFS
- DC-Offset $< 10^{-3}$
- Stereo-Korrelation: 0.1–0.9
- RMS-Steigerung über Zeit
- Keine 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 Reproduzierbarkeit & Performance

- Determinismus: `-seed` setzen (NumPy PCG64)
- Komplexität: $O(KN)$, Pink-Noise: $O(N \log N)$
- RAM: Stereo-Float32 $8N$ Bytes
- Beispiel: 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. WAV rendern, in DAW importieren
2. Sidechain mit Kick ($\frac{1}{4}$ Notes)
3. Sanfte EQ/Sättigung (2 dB @ 10 kHz, Low-Cut 30 Hz)
4. Time-Stretch bei BPM-Abweichung

9 Troubleshooting

- Zu spitz → fmax ↓, drive ↓, pink_db 3 dB
- Zu matschig → pairs ↓, phase_diffuse ↓
- Mono-Probleme → stereo_phase_max 0.2
- Clippt → headroom_db ↑

10 Mini-Validierung (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
```


Note on Terminology

All terms, metaphors, and model names used in this repository (e.g. "Pseudoscalar Score", "Schrödinger Zone", "OddSpin", "MaxwellFlux" etc.) are original to the author. These names are not in the public domain. Any use of these names, terms, or model identifiers — especially for commercial or branding purposes — is prohibited without prior written permission from the author. This restriction applies regardless of whether the underlying code or methods are licensed under open-source terms.

I'd be happy if you like my work: https://buymeacoffee.com/marthafay

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