A Text-Steerable Instrument for Sketching Procedural Soundscapes via Language Models

Prabal Gupta
Rama Labs
Kitchener, Ontario, Canada
prabal@rjeinc.ca

Abstract

We present a real-time musical interface that converts natural-language scene descriptions into evolving procedural soundscapes. A performer types a prompt such as "warm jazz café at midnight" and steers it through direct parameter adjustments—stepping brightness down, switching a rhythm style—each producing a predictable, audible shift without re-prompting. Where GPU-bound text-to-audio systems synthesize monolithic waveforms, our instrument generates human-readable configurations over a categorical schema, enabling fine-grained performer control; most valid combinations are designed to sound musically coherent. Three interchangeable backends—embedding retrieval for sub-second CPU-only use, hosted LLMs via API, and a fine-tuned 270M local model—all emit the same schema. A live generator architecture continuously emits audio while resolving new instructions in the background, crossfading seamlessly when ready; even when an LLM takes 5–12 seconds to respond, the audience hears uninterrupted sound—reframing text-to-music as an ongoing performable stream rather than a one-shot generation. We evaluate text–audio semantic alignment using LAION-CLAP on held-out prompts as a technical proxy, finding that retrieval-based configuration outperforms random valid configurations on this metric, while noting that LAION-CLAP also informed retrieval-map construction. We report performance observations, informal listener feedback, and release materials for the SDK, dataset artifacts, model, and audiovisual performance interface.

Keywords

text-to-music, procedural synthesis, real-time performance, language models, parametric control, live coding

1 Introduction

In a live coding set, the performer types `"neon city at 2am"` and the ambient drone keeps playing. Five seconds later, the new configuration resolves—bright, electronic, fast—and the sound crossfades into it. The performer nudges brightness down two steps. The city gets darker. They switch rhythm to `"heartbeat"`. The pulse slows. None of this required stopping the music, waiting for a GPU, or understanding DSP.

Neural text-to-audio systems such as MusicGen [6], MusicLM [1], and Stable Audio [7] produce high-quality audio from text. However, such systems typically return generated audio rather than interpretable synthesizer parameters; CTAG explicitly frames this as a tweakability problem for neural audio systems [4]. Azimi and Zareei [2] fine-tuned Stable Audio Open for live improvisation, demonstrating that even with GPU acceleration, generation latency and unpredictable outputs remain challenges.

Our SDK addresses this tension by generating *interpretable parameter configurations* for a procedural synthesizer rather than raw audio. Its live generator architecture decouples instruction resolution from audio emission—the current configuration keeps sounding while the next one resolves in the background. This brings LLM-powered text-to-music into live performance without audible interruption.

For the performer, the key affordance is persistent steerability: a prompt establishes a scene, but the musical work happens through named parameter edits—darkening the spectrum, switching rhythm, widening the space—while the stream continues. Text becomes a playable control surface, not a one-shot trigger.

The system is not a neural audio generator: language models resolve prompts into symbolic synthesizer configurations, and a deterministic procedural engine renders the sound—less timbrally general than neural text-to-audio, but inspectable, steerable, and stable during performance.

We contribute: (1) a live generator architecture that hides resolution latency behind continuous audio, enabling the SDK to function as an instrument runtime rather than a generation endpoint; (2) a 34-field configuration schema whose categorical labels mean most valid combinations sound musically coherent—a mapping-design contribution in the sense of Hunt and Kirk [8] and Magnusson [11]; and (3) an open-source release.[1] A companion ACM SIGGRAPH 2026 talk addresses the broader system architecture and benchmark framing; the present paper focuses on the instrument contribution: the live generator as performance runtime, the configuration schema as mapping design, and the prompt–listen–steer workflow.

2 Related Work

Text-to-audio instruments. Azimi and Zareei [2] fine-tuned Stable Audio Open for improvisation, generating raw audio fragments. Shepardson et al. [16] used text notation to control neural vocal synthesis in Tungnáá. Both systems synthesize neural audio directly, with differing compute requirements and limited post-generation editability. Our SDK generates structured parameters instead—trading timbral fidelity for sub-second latency, direct editability, and reproducible output.

Text-to-synthesizer mapping. CTAG [4] maps text to a modular synthesizer with 78 parameters via iterative optimization, but its search runs at generation time. SynthScribe [3] supports synthesizer sound retrieval, creation, and modification through multimodal text/audio tools. Our system front-loads LLM generation into dataset construction and retrieves complete multi-layer configurations at runtime. Latent-space instruments like

NIME '26, June 23–26, 2026, London, UK

[1] SDK and demo: https://github.com/prabal-rje/latentscore, https://latentscore.com. Supplementary materials: https://zenodo.org/records/19944277.

```python
async def performance():
    yield "warm jazz cafe at midnight" # text -> config
    await asyncio.sleep(8) # play for 8s

    yield MusicConfigUpdate( # relative nudge
        brightness=Step(-2), # two steps darker
        echo="heavy") # absolute value
    await asyncio.sleep(8)

    yield "neon rain on empty streets" # new scene
    # current config keeps playing while backend resolves
        ...

session = live(performance())
session.play(seconds=60)
```

Figure 1: *The live generator programming model. The performer yields instructions; audio plays continuously from the current configuration.*

Stacco [13] and Latent Mappings [12] navigate continuous neural spaces through physical gesture; our SDK navigates a discrete parameter space through language.

Programming frameworks. ChAI [10] and ChuMP [15] added interactive AI tools and modular package management to ChucK.

3 The Instrument

3.1 Design Goals

Immediacy (G1): sub-second text-to-sound. *Controllability (G2):* all parameters exposed as named fields with human-readable labels. *Reproducibility (G3):* a given configuration and seed yield identical output. *Learnability (G4):* a single-line entry point that scales to full programmatic control. *Continuity (G5):* audio never stops during configuration resolution.

3.2 The Live Generator

The SDK's core contribution is the live generator—an async streaming interface where the instrument is always sounding. A performer (or game engine, installation, or MIDI controller) writes a Python generator that yields instructions: text prompts, absolute configurations, or relative parameter adjustments. The SDK continuously emits audio chunks from the current configuration. When a new instruction arrives, it resolves in the background—calling an LLM, querying the retrieval index, or applying a parameter update—while audio emission continues uninterrupted. Once resolved, the SDK crossfades into the new configuration.

The sleep duration controls *how long each configuration plays*, and new instructions can be yielded at any time—including while a previous resolution is still pending. Figures 2 and 3 show the timing relationship between the generator, resolver, and audio output for both backends. The SDK queues instructions and resolves them in order. Rapid yielding can accumulate pending instructions with slow backends; with the fast backend, rapid yields produce transitions too quick to sound musical. In practice, performers quickly learn to pace yields to the backend's resolution speed.

The SDK supports three backends. The **fast** mode (default) uses embedding retrieval—sub-second after warm-up, fully reliable, no network needed. The **external** mode routes to any LLM via the developer's API key, with seamless playback despite network latency; failed calls fall back to fast mode. A local **expressive** mode runs a 270M fine-tuned model on-device (experimental). A web-based UI demonstrates the SDK as a playable instrument: it displays each configuration's co-generated title and color palette alongside the parameter editor, with background particle effects responsive to playback state—creating an audio-visual performance surface.

3.3 Configuration Schema as Mapping Design

The schema contains 34 fields in five groups: global parameters (8: `tempo`, `root`, `mode`, `brightness`, `space`, `density`, `motion`, `attack`), six orchestration layers (`bass`, `pad`, `melody`, `rhythm`, `texture`, `accent`—each selecting a style from a curated set), spatial/texture (5: `stereo`, `depth`, `echo`, `human`, `grain`), melody generation (10), and harmony (5). All fields use categorical labels, designed for both human interpretability and reliable LLM generation—an LLM can produce `"bright"` far more reliably than a meaningful float like `0.73`.

Eight of these fields use ordered labels that support relative stepping: a performer can issue `Step(+1)` or `Step(-1)` to move along the label sequence, clamping at boundaries. The remaining fields use style selectors, booleans, or bounded values accepting absolute assignments only. Each resolved configuration also includes co-generated metadata—a descriptive title, three color palettes, and a brief justification of the parameter choices.

The schema was iteratively refined through an AI-human design loop. Categorical labels make each level perceptually distinct and enable reliable LLM generation. Constrained style sets reduce discordant combinations. The design goal: *most valid configurations should sound musically coherent.* The schema is biased toward coherence to support exploratory play—a decision in Magnusson's sense [11], not a limitation.

4 Retrieval-Based Configuration

Given a coherent parameter space, the remaining challenge is mapping text to configurations quickly enough for live use. The default fast backend uses a retrieval map of LLM-generated configurations curated offline. We distilled ~10,500 unique scene descriptions from the Common Pile [9], an openly licensed text corpus, prompted Gemini 3 Flash Preview to generate five candidate configurations per scene as valid schema instances, scored each rendered audio by measuring text–audio semantic similarity using LAION-CLAP [17], a contrastive language–audio model trained on diverse audio–text pairs, and selected the best, embedded each scene with a sentence transformer [14], and exported the map. At runtime, the nearest neighbor is returned—approximately one second on CPU.

5 Design Observations

The author used the instrument across multiple composition sessions. The live generator excels at gradual scenic evolution: yielding `"quiet midnight hush"`, waiting, then stepping `brightness` down and switching `rhythm` to `"heartbeat"` produces a coherent ambient environment that evolves naturally. This two-level control—text for macro-level scene changes, parameter stepping for micro-level refinement—emerged as the instrument's natural performance idiom. When using an external LLM backend, the 5–8 s wait for a new scene to resolve is noticeable but not disruptive—the current soundscape keeps playing, and the crossfade when it arrives feels like a natural transition rather than

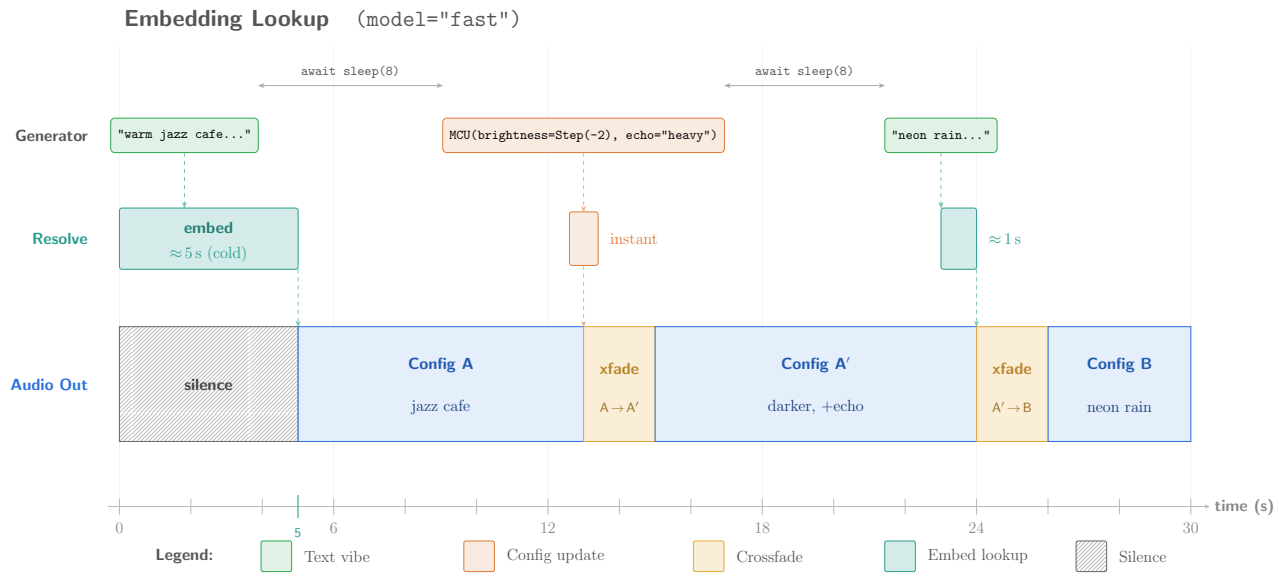

Figure 2: *Embedding Lookup (*`model="fast"`*) backend. The first resolve includes model loading (~5 s cold start); subsequent embed lookups resolve in ~1 s. Parameter updates (*`MusicConfigUpdate`*) apply instantly. Audio plays continuously after the first configuration resolves.*

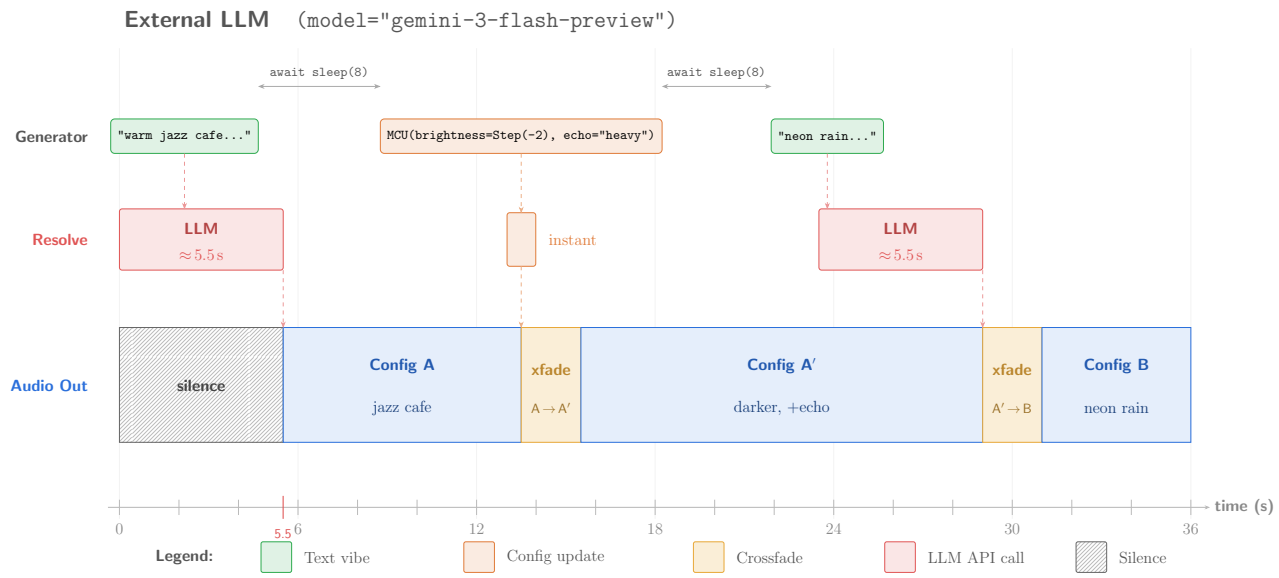

Figure 3: *External LLM (*`model="gemini-3-flash-preview"`*) backend. Each text prompt triggers a ~5.5 s API call; the current configuration keeps playing uninterrupted while the new one resolves. The time axis extends to 36 s for the same three instructions that the fast backend completes by 30 s—this gap compounds with each text prompt.*

a technical artifact. The instrument struggles with sharp stylistic pivots and cannot match timbral specificity for prompts like "upright bass."

Five lay participants each listened to four text-generated soundscapes paired with author-steered variations in randomized order, without being told which was which, and were asked to describe what they heard and whether changes felt intentional. All five independently described output as sounding musically intentional rather than random, offering informal support for the schema design goal. Limitations identified: no individual layer volume control, restricted stylistic range for culturally specific music, and occasional audio artifacts in dense configurations. These comments are treated as informal design feedback rather than evidence of general audience response: the sample was small, participants were lay listeners, and no statistical claims are made from these remarks.

A technical benchmark on 200 held-out prompts showed embedding retrieval scoring higher on text–audio semantic similarity than a random valid configuration, suggesting that retrieval adds semantic targeting beyond the schema's baseline coherence

under this proxy. This metric is a system-reliability and alignment proxy, not a perceptual-quality measure. LAION-CLAP also informed retrieval-map construction, so retrieval has an expected advantage on this metric; human perceptual validation is planned.

6 Discussion

G1 (immediacy) and G3 (reproducibility) are met. G5 (continuity) is met—in a 200-prompt benchmark, the embedding backend resolved all prompts successfully (~0.3 s config generation, ~1.2 s total with synthesis); the external LLM backend resolved 178/200 (~5.6 s config generation, 89% success rate), with failed calls falling back to the fast backend to maintain uninterrupted playback. G2 (controllability) is partially met—categorical labels provide precise steering, but retrieval coverage gaps cause occasional semantic mismatches. G4 (learnability) is informally supported but formally unevaluated.

The live generator reframes text-to-music from a one-shot generation into an *ongoing stream*—decoupling resolution from emission generalizes beyond music to any latency-sensitive domain. The CPU-only, reproducible design responds to Clarke et al.'s [5] concerns about longevity, reproducibility, documentation, and the resource demands of deep generative models in NIME practice—the default backend requires no inference at runtime, amortizing LLM cost into a one-time dataset construction step. The open-source SDK, extensible schema, and shared retrieval dataset contribute to the NIME 2026 *Communities* theme: the instrument's parameter space is a shared vocabulary that other developers can extend with new style sets, new retrieval entries, or new synthesizer backends—and the live generator architecture provides the runtime for any such extension to be performed live.

Musical outcomes and limitations. The strongest outputs are ambient, cinematic, and loop-adjacent soundscapes where texture and harmonic color matter more than authored melody. It is less successful for genre imitation or timbral specificity. This trade-off privileges predictable steering and live continuity over timbral range.

7 Ethical Standards

This work follows NIME ethical expectations. Informal listener feedback was solicited from consenting adult participants on coherence, controllability, and musical character; no personal data was collected, and the feedback is reported as informal design input, not a powered study. The system generates procedural audio rather than imitating recordings or performers; the retrieval dataset uses an openly licensed corpus [9] and releases structured configurations and rendered examples rather than copyrighted music recordings. The CPU-first default backend amortizes LLM cost across all use, addressing NIME's guidelines on lightweight models and AI environmental impact. AI coding tools assisted with codebase and schema design, and grammatical correction; the open-source release includes an `AI_DISCLOSURE.md` documenting their scope. A supplementary video demonstrates the live generator.

References

[1] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. 2023. MusicLM: Generating Music from Text. *arXiv preprint arXiv:2301.11325* (2023). https://doi.org/10.48550/arXiv.2301.11325

[2] Misagh Azimi and Mo H. Zareei. 2025. Live Improvisation with Fine-Tuned Generative AI: A Musical Metacreation Approach. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Canberra, Australia, Article 54, 389–393 pages. https://doi.org/10.5281/zenodo.15698902

[3] Stephen Brade, Bryan Wang, Mauricio Sousa, Gregory Lee Newsome, Sageev Oore, and Tovi Grossman. 2024. SynthScribe: Deep Multimodal Tools for Synthesizer Sound Retrieval and Exploration. In *Proceedings of the 29th International Conference on Intelligent User Interfaces*. https://doi.org/10.1145/3640543.3645158

[4] Manuel Cherep, Nikhil Singh, and Jessica Shand. 2024. Creative Text-to-Audio Generation via Synthesizer Programming. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*. PMLR, 8270–8285. https://doi.org/10.48550/arXiv.2406.00294

[5] Isaac Clarke, Francesco Ardan Dal Rí, and Raul Masu. 2025. Longevity of Deep Generative Models in NIME: Challenges and Practices for Reactivation. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Canberra, Australia, Article 32, 224–230 pages. https://doi.org/10.5281/zenodo.15735662

[6] Jade Copet, Felix Kreuk, Itai Gat, Tal Remez, David Kant, Gabriel Synnaeve, Yossi Adi, and Alexandre Défossez. 2023. Simple and Controllable Music Generation. In *Advances in Neural Information Processing Systems 36*. https://doi.org/10.48550/arXiv.2306.05284 arXiv:2306.05284

[7] Zach Evans, CJ Carr, Josiah Taylor, Scott H. Hawley, and Jordi Pons. 2024. Fast Timing-Conditioned Latent Audio Diffusion. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*. PMLR, 12652–12665. https://doi.org/10.48550/arXiv.2402.04825

[8] Andy Hunt and Ross Kirk. 2000. Mapping Strategies for Musical Performance. In *Trends in Gestural Control of Music*, Marcelo M. Wanderley and Marc Battier (Eds.). IRCAM – Centre Pompidou, Paris, France. https://www-media.idmil.org/media/Trends_Ircam/DOS/P.HunKir.pdf

[9] Nikhil Kandpal, Brian Lester, Colin Raffel, Sebastian Majstorovic, Stella Biderman, Baber Abbasi, Luca Soldaini, Enrico Shippole, A. Feder Cooper, Aviya Skowron, John Kirchenbauer, Shayne Longpre, Lintang Sutawika, Alon Albalak, Zhenlin Xu, Guilherme Penedo, Loubna Ben Allal, Elie Bakouch, John David Pressman, Honglu Fan, Dashiell Stander, Guangyu Song, Aaron Gokaslan, Tom Goldstein, Brian R. Bartoldson, Bhavya Kailkhura, and Tyler Murray. 2025. The Common Pile v0.1: An 8TB Dataset of Public Domain and Openly Licensed Text. *arXiv preprint arXiv:2506.05209* (2025). https://doi.org/10.48550/arXiv.2506.05209

[10] Yikai Li and Ge Wang. 2024. ChAI => Interactive AI Tools in ChucK. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Utrecht, Netherlands, Article 81, 553–559 pages. https://doi.org/10.5281/zenodo.13904949

[11] Thor Magnusson. 2019. *Sonic Writing: Technologies of Material, Symbolic, and Signal Inscriptions*. Bloomsbury Academic. https://doi.org/10.5040/9781501313899

[12] Tim Murray-Browne and Panagiotis Tigas. 2021. Latent Mappings: Generating Open-Ended Expressive Mappings Using Variational Autoencoders. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Shanghai, China, Article 66. https://doi.org/10.21428/92fbeb44.9d4bcd4b

[13] Nicola Privato, Victor Shepardson, Giacomo Lepri, and Thor Magnusson. 2024. Stacco: Exploring the Embodied Perception of Latent Representations in Neural Synthesis. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Utrecht, Netherlands, Article 62, 424–431 pages. https://doi.org/10.5281/zenodo.13904899

[14] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, Hong Kong, China, 3982–3992. https://doi.org/10.18653/v1/D19-1410

[15] Nicholas Shaheed and Ge Wang. 2025. ChuMP and the Zen of Package Management. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Canberra, Australia, Article 90, 610–617 pages. https://doi.org/10.5281/zenodo.15698984

[16] Victor Shepardson, Jonathan Reus, and Thor Magnusson. 2024. Tungnáá: a Hyper-realistic Voice Synthesis Instrument for Real-Time Exploration of Extended Vocal Expressions. In *Proceedings of the International Conference on New Interfaces for Musical Expression*. Utrecht, Netherlands, Article 78, 536–540 pages. https://doi.org/10.5281/zenodo.13904943

[17] Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. 2023. Large-Scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation. In *ICASSP 2023 – 2023 IEEE International Conference on Acoustics, Speech and Signal Processing*. 1–5. https://doi.org/10.1109/ICASSP49357.2023.10095969