The Good, the Bad and the Ugly: Can Live Coding and Large Language Models Ride Together?

Uandha Fernandes Barbosa[1]
Department of Design and Computation Arts
Concordia University
uandha.fernandesbarbosa@mail.concordia.ca

Dominic Thibault
Faculté de Musique
Université de Montréal
dominic.thibault@umontreal.ca

Abstract

Advancements in Large Language Models (LLMs) and agent workflows have expanded coding capabilities across programming languages. In live coding environments, however, their use raises questions about alignment with the TOPLAP manifesto's philosophy and commitment to transparency and the visibility of human and machine performative processes. Framed by the cinematic metaphor of *The Good, the Bad and the Ugly* (1966), this research-creation investigates, using a confrontational ride metaphor, the arrangement of a Claude Large Language Model and Strudel REPL, connected through a Model Context Protocol saddled with artist-driven musical preferences—rhythmic material from Brazilian São Paulo funk and a selection of non-Western scales. This AI-live coding desert is crossed through an iterative process of reflection, translating specific musical knowledge into machine-readable logic, in search of understanding how, and whether, LLMs may be steered to support such practices. By embracing the limitations and unpredictability of these models, this investigation explores a landscape where the artist continuously negotiates artificial intelligence's companionship, finding "beauty"—and maybe joy—in such an "ugly" unresolved ride.

1 IN THE HORIZON OF BEGINNINGS

In the cinematic world of The Good, the Bad and the Ugly (Leone, 1966), each character embodies a different mode of engaging with uncertainty. The Good seeks alignment and clarity, the Bad navigates conflict and instability, and the Ugly persists in chaos and unpredictability.

This metaphor provides the core lens for this investigation, in which the live coder moves between these modes, continuously adapting their approach in response to unstable and imperfect artificial intelligence (AI) generated outputs, finding their utility to one another.

This research-creation asks how an artist's situated musical knowledge—preferences, cultural background and ways of working—can be encoded as machine-readable context so that a general-purpose Large Language Model (LLM) becomes a workable companion within one's practice, and what the live coder surrenders, in transparency and control, in exchange.

Such reflections are not new to this field, though its terms have changed. In 2000, Adrian Ward and Alex McLean presented the Generative Manifesto (Ward & McLean, 2000), which demanded that artists use only software they had written themselves, on the grounds that software dictates output and that authorship cannot be granted to those who have not authored.

Twenty-six years later, this investigation does not meet that demand. What follows may suggest that the value of integrating an LLM into the live coding equation lies in the acceptance of an imperfect companionship—one where the questions between human and machine remain unresolved and must be continuously (re) negotiated.

This ride runs as follows. Section 2 sets out the values the live coding community has built for itself. Section 3 situates the work in relation to decades of machine learning in live coding. Section 4 describes the tools and the reasons for choosing them. Section 5 sets out the musical knowledge encoded in the Model Context Protocol (MCP) server: rhythmic material from Brazilian São Paulo funk and a selection of non-Western scales. Section 6 addresses natural language as the interface and outlines the criteria for assessing the collaboration. Section 7 reports what the sessions showed, organized under our cinematic metaphor. Section 8 states the limits of what can be claimed from them and future work, and Section 9 returns to the principles set out inside the live coding community and asks which of them survive to this companionship arrangement.

2 THE SADDLE BEFORE RIDING: TOPLAP MANIFESTO

Live coding includes a variety of approaches and skills, all sharing one main principle: the real-time writing of algorithmic instructions (Magnusson, 2014). Before starting our ride into this present AI-live coding desert, it is useful to understand the philosophical foundations on which this practice and its community were built, and the values they still share.

[1] Throughout this paper, "I" refers to the first author, whose practice is the subject of the investigation; "we" refers to the authors jointly.

The Generative Manifesto introduced above (Ward & McLean, 2000) was among the first statements of those values. While live coding remains open and evolving without strict boundaries, the later TOPLAP manifesto, drafted in 2004 and maintained since as a living document (TOPLAP, 2004/2024), has played the central role in defining its ethos—particularly around transparency, liveness, and the visibility of computational processes. A fragment:

> *We demand:*
> *•Give us access to the performer's mind, to the whole human instrument.*
> *•Obscurantism is dangerous. Show us your screens.*
> *•Programs are instruments that can change themselves*
> *•The program is to be transcended - Artificial language is the way.*
> *•Code should be seen as well as heard, underlying algorithms viewed as well as their visual outcome.*
> *•Live coding is not about tools. Algorithms are thoughts. Chainsaws are tools. That's why algorithms are sometimes harder to notice than chainsaws.* (TOPLAP, 2004/2024)

The manifesto goes on to acknowledge a limit to its own demand. It accepts that a lay audience need not understand the code to appreciate the performance, drawing a comparison itself: "it is not necessary to know how to play guitar in order to appreciate watching a guitar performance" (TOPLAP, 2004/2024).
The demand, then, could be perceived as a matter of legibility per se, where nothing essential happens that cannot be seen. Thus, what matters is not how much computation is involved but whether it can be shown. A complex machine learning system can still be opened; an undisclosed one cannot, and that is where the real tension lies in this investigation.
One further passage is worth noting, since this paper returns to it. Among the manifesto's stated preferences is the extemporization of the algorithm—inventing code live, on the spot —as a display of mental dexterity and the refusal of any safety net during performance. If the code is extemporized by an AI model, whose dexterity is on display? And a Large Language Model standing by to generate patterns is, arguably, the safety net a live coder could perform with.

3 BEHIND THE GOOD AND THE BAD: LIVE CODING AND LLMS IN CONTEXT

When thinking about artistic production, the image that often comes to mind is rarely a creator sitting in front of a computer, immersed in lines of code. The abstract symbols and syntax of algorithms do not immediately align with conventional expectations and notions of musical expression.
As a live coder, this human-computer relationship in the practice feels inherently "mathmagical," even self-hypnotic. I particularly found deep enjoyment in writing structured lines of code and in witnessing how such processes, grounded in mathematics and physics, turn into dynamic art and personal expression.
From day one, this investigation began in a classroom. Among the curiosities of studying generative AI models and their biases, a question emerged that seemed more than just technical: when a model—trained on vast, undisclosed amounts of material—is asked to make specific personal music, whose music does it make? Biases in these systems are usually presented as imbalances, with related questions about which cultures are fairly represented and which are not.
As a consequence, live coding is a community undergoing a visible, and at times touchable, tension over these tools. On one side of the coin, the capacity of an LLM to write correct code is not in doubt; on the other, correctness was never what was at stake. The question is not whether it can produce patterns, but what kind, and whose.
Inevitably, these reflections found their way into my practice. As a live coder and contemporary composer, my artistic production is inseparable from my cultural background. In my first experiments live coding with an LLM, I asked the model to write some of my patterns. What returned was syntactically correct but felt musically hollow—a type of music that meant nothing, drawn from the massive data of everything it was trained on.
That was the point where our investigation started. Although the output was not wrong in any technical sense, the model missed—and "failed"—because what it wrote belonged to a "musical world" that was far apart from mine. This explains the motivations and choices that follow inside this investigation: choosing a general-purpose Large Language Model and further providing it with a specific context built from my own practice. It also explains why the material encoded into the mediating layer matters here, and the final paper's purpose: to understand what such a model is capable of generating afterwards.

3.1 Old Boots: Live Coding and Machine Learning

Back to its early days, live coding as an artistic practice has embraced a shared, exposed collective experience, per se. The community has been guided by principles articulated in the TOPLAP manifesto, which emphasizes transparency, liveness, and the visibility of the process as a whole. By demanding to "show us your screens" (TOPLAP, 2004/2024), the central idea is reinforced: the audience should have access not only to the artistic output but also to the performer's thinking process.
Also, as it resists a single unified definition, it can be understood within a broader trajectory of "live programming," where liveness is defined not only by the presence of an audience but by the immediacy of such human-computer interaction (Magnusson, 2014). In this context, the code becomes both medium and message, something to be heard, seen, and understood as part of the whole experience.
With the rise of LLMs and their current boom, a new layer of doubts and tensions has entered this domain. As Knotts and Paz (2021) argue, the integration of machine learning into live coding challenges core principles of the practice, particularly those related to transparency, authorship, and the role of the performer.
Machine learning is indeed not new to live coding. Collins (2011) built a music information retrieval library for SuperCollider that has since been used by others in performance systems, and later surveyed the crossover between live coding and machine listening (Collins, 2015). Navarro Del Angel and Ogborn (2017) built Cacharpo, an agent that listens to a human live coder

through machine listening and music information retrieval, and responds by typing code within the genre of cumbia sonidera. Xambó (2021) documents a further lineage of such systems reaching back to 2007. Alongside these sits a long tail of practice that also uses clustering, Markov models, and similar techniques, demonstrating that ML applications have long been part of this community's toolkit.

At the same time, some approaches suggest that these tensions are not necessarily obstacles, but opportunities for artistic and research exploration. One possible direction is to position machine learning as a collaborator or agent within the live coding environment, rather than attempting to fully automate one's creative processes. As discussed by Wilson et al. (2023), machine agents can be integrated as co-creative participants that should not merely generate material but engage in a collaborative process that includes reflection, aesthetic consideration, and evaluation. For example, these virtual agents may take on specific roles, such as retrieving or transforming material (Xambó, 2021). In doing so, they may introduce concepts such as negotiation and shared agency, expanding the scope of live coding practice (Xambó, 2021). Related experiments have taken natural language as the performed medium; for example, Lee, Essl and Martinez (2016) turn the typing of a poem into a real-time audiovisual performance, a precedent discussed in Section 4.4.

Virtual agents are not new to live coding either. Xambó (2021) reviews a lineage framed through notions of virtual agency, machine musicianship and the musical cyborg, and analyzes it along two dimensions: social interactivity and learnability. Within that taxonomy, this project sits closest to systems in which an agent generates code alongside a human, but differs on the second dimension: learnability. As Xambó (2021) discusses, Cibo was trained on recordings of live coding performances (Stewart & Lawson, 2019), and Mégra learns during online training (Reppel, 2020). This arrangement, by contrast, leaves the model's weights untouched and intervenes only in the context supplied. What is authored is not a model but a vocabulary.

That distinction becomes clear when set alongside the earlier history of machine agency in this practice. In the systems Collins (2015) surveys, agents were interfaced through audio—onset detection, pitch tracking, timbral feature extraction—and control passed between performer and system through sound itself. The interface here instead inputs natural language. This is the substance of an LLM-mediated arrangement and also its principal liability: language admits ambiguity.

Collins (2015) speculates about something structurally similar. Among the future possibilities he ponders is a unique constructed language emerging between a human and a machine symbiont, built as they grow up together, with a virtuosity he suggests would be considerable given substantial investment of learning. The alias tagging described in Section 7 arrives differently; the vocabulary was built across sessions, refined whenever outputs missed what was meant, and the development is one-sided.

Questions of evaluation remain open in this field. Xambó (2021) observes that the literature on evaluating virtual agents in live coding is scarce, and asks who should carry out such evaluation: the audience, the live coder, or the agent itself. Bernardo, Kiefer and Magnusson (2021) assess Sema, a browser-based environment for live coding with machine learning, applying the Creativity Support Index across its subsystems and feeding the resulting insights into a subsequent design iteration. Notably, they close by questioning the applicability of that to systems supporting non-linear or variable tasks such as live coding and machine learning, where they found the analysis remained rather superficial. This investigation takes that finding into account. It does not conduct a user study; the criteria set out in Section 6.1 are the live coder's own, and are applied to a practice from inside it.

Finally, the tension that motivates this paper is therefore not between live coding and Machine Learning. It is between a community whose tools are free, open and modifiable by their users, and the new generation of AI models that are black boxes, closed-source and controlled by large commercial entities. Many of the systems exemplified above were built by the artists who performed with them, or by researchers within the community, and can be inspected, modified, or rewritten. Differently, an LLM accessed through a commercial API cannot. This distinction is what makes the arrangement described in this paper confrontational with the values articulated in the manifestos, as well as with those held by artists inside the live coding community.

3.2 New Boots: Vibe Coding

Coined by Andrej Karpathy in February 2025 (Karpathy, 2025), and widely adopted since, vibe coding in its original formulation involves relinquishing oversight.

Indeed, this project used it for its own development; however, it differs from that relation in one relevant aspect. The material the model draws from in the intermediary Model Context Protocol (MCP) layer is a body of musical knowledge transcribed, encoded and structured by the artist's hands, preferences, and cultural background. Outputs are then auditioned, curated, and accepted or rejected within one cycle—or across many — of a manually iterative development loop.

4 THE GUNSLINGERS' TOOLS

The approach taken here does not frame the model as an autonomous creator but as a co-creative participant whose behaviour we try to shape and constrain to the artist's own practice. What follows describes the arrangement built to do that.

The specific combination—Strudel[2], Claude[3], and a Model Context Protocol[4] server—was chosen based on practical reasons. Strudel runs in a browser and requires no local configuration, which makes it automatable by an external process without a bespoke integration layer. It

[2] https://strudel.cc/workshop/getting-started

[3] https://www.anthropic.com/

[4] MCP: The Model Context Protocol is an open standard, introduced by Anthropic in November 2024, that allows language models to connect to external tools and data sources through a common interface.

also shares mini-notation with TidalCycles[5], so patterns remain legible to a wider live coding community. The MCP was selected because it is a mechanism for supplying a commercial LLM model with structured external context. Claude utilizes the MCP since the protocol was developed by Anthropic, the company that developed this model. A final, and substantial reason, is that these tools have been widely and routinely employed, including in courses where students encounter them.

4.1 The Good's Simple Boots: Uzulang and Strudel REPL

Strudel earns the Good's boots by being the most straightforward member of this arrangement: open, accessible, and requiring no configuration. It belongs to a family of pattern-based languages known as Uzulang (Uzu, n.d.), which share a mini-notation inspired by polymetric expressions from the Bol Processor[6]. One of them is the paradigmatic TidalCycles, developed by Alex McLean in 2010, which adopted a pattern-based approach to live coding. More recently, it was extended to the browser via the Strudel REPL, a JavaScript-based implementation developed by Felix Roos and Alex McLean in 2023.

Being web-based, it enables immediate interaction and accessibility for users, since no setup or configuration of any kind is required. Central to this is the concept of the REPL[7], which consists of an interactive interface inherited from computing traditions. In this context, the REPL, as shown in Figure 1, allows users to write code and instantly evaluate it by hearing or seeing the result in real time. Such an approach is a core feature of live coding practice, where composition and performance occur simultaneously.

Figure 1: Strudel REPL web environment with the default mini notation pattern ready to play.

4.2 The Bad's Black Hat: LLMs and Anthropic's Claude

Claude wears the black hat not because it "fails", but because nothing about how it works can be seen, as discussed below.

Based on transformer architectures and trained on massive amounts of data—often undisclosed—LLMs not only require significant computational resources, but often are described as "black boxes." This designation basically means they are not open source, do not allow access to their architecture, and do not offer options to make substantial modifications.

It's also no novelty that these models have been evolving at fast and large pace, demonstrating increasing capabilities across a wide range of tasks, including applications in programming languages (Jiang et al., 2026). Claude, the model used in this project, was developed by Anthropic, a company founded in 2021 with a stated focus, as they themselves describe, on building safer and more controllable AI systems (Anthropic, n.d.).

For those engaged in programming, tools such as Claude (and more recently Claude Code) have become recognized for their ability to assist with writing, debugging, and structuring code.

However, these capabilities translate less directly into live coding environments. Due to the nature of LLMs in producing statistical outputs, combined with their almost untouchable "black box" architecture, disconnected or even nonsensical sonic code is often generated within the Strudel context, as exemplified in this project's repo on GitHub[8].

4.3 The Ugly's Leather Gloves: Model Context Protocol

The MCP wears the Ugly's gloves: it is neither open like Strudel nor sealed like Claude. It is the part of this arrangement that is authored—built by hand to work around a closed AI box one cannot see inside.

Introduced in late 2024, it functions as an intermediary that facilitates interaction between LLMs and external data sources or tools (Anthropic, 2024). It allows developers to ground model responses in specific datasets, countering generalist outputs by providing the LLM with the precise local context destined for specific tasks.

That means MCP opens the door to building upon an AI model's internally trained knowledge, such as Anthropic's Claude, by providing a direct bridge to specialized external data. In this investigation, the protocol provides a structured musical knowledge context and enables its integration by mediating between Claude's generative capabilities and the Strudel REPL live-coding environment.

The MCP used here was originally developed by William Zujkowski[9]. While the initial server provides a solid foundation of implemented musical tools (e.g., scales, chords and chord progressions, drums, bass, melody and

[5] https://tidalcycles.org/

[6] https://bolprocessor.org/

[7] REPL: Read-Eval-Print Loop is an interactive programming cycle, in which expressions are entered, evaluated and returned immediately. In live coding environments the expansion is sometimes given as Read-Eval-Print/Play Loop, since evaluation produces sound.

[8] https://github.com/uandhafb/The-Good-The-Bad-and-The-Ugly

[9] https://github.com/williamzujkowski/live-coding-music-mcp

audio analysis), this project builds upon and extends those capabilities by embedding musical preferences drawn from my own practice.

In doing so, a central aspect was to understand what Claude is capable of producing independently, as well as how its outputs change when interacting with musical knowledge already implemented within this server. It became evident that, when supported by existing structured knowledge within the MCP, the model could produce more elaborate musical results. For example, the output evolved to more advanced code syntax, such as related to the usage of random modifiers (e.g., every, often, rarely) and polyrhythmic Euclidean.

Among experimentations and reflections discussed later, the decision was to incorporate non-Western scales, rhythmic patterns derived from Brazilian Funk, and curated examples for textural sonic production, which include other pattern modifiers (e.g., degradeBy, stack, struct) based on my personal practice-led research.

4.4 The Three Horses: Claude - MCP - Strudel

This architecture can be understood as a multi-layered pipeline. At the primary level, Claude LLM receives user prompts that describe specific musical intentions (e.g., create a dark melody). These instructions are translated into structured tool calls within the MCP server, which employs browser automation to control the Strudel environment, effectively enabling the LLM to write, play, and modify code in real time. Consequently, a typical workflow involves initializing the local Strudel REPL via Claude, generating a musical pattern through a prompt, auditioning the result, and optionally modifying the code either manually in Strudel or through subsequent requests to the LLM.

As previously noted, curated material was developed and integrated as templates within the MCP server, allowing Claude to consult artist-driven data to generate and execute mini-notation patterns. This approach enables decisions not only regarding which musical material to feature but also the specific ways in which its representation could directly shape Claude's generative outputs. A schematic of this process is presented in Figure 2.

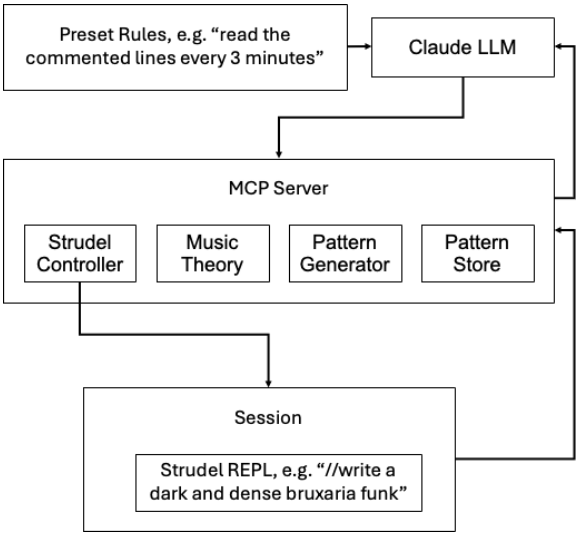

Figure 2: Schematic of Claude - MCP - Strudel REPL.

Finally, a feature was implemented that enables Claude to periodically parse commented lines within the Strudel REPL, for instance, every three minutes at the start of each new session. This feature incorporates prompt instructions directly into the performative code, eliminating the need to exit the live coding interface. By maintaining the interaction within the environment, this functionality preserves the continuity and immersive nature of the session.

This gesture has a precedent outside live coding proper. Lee, Essl and Martinez (2016) propose live writing, in which the act of typing a poem is captured and augmented so that writing itself becomes a real-time audiovisual performance, and, as described by the authors, an idea drawn explicitly from live coding's practice of revealing the programming process to an audience. They describe live coding as a special case of live writing in which the text happens to be executable, and distinguish the two by what the writing acts upon—a program in one case, the reader's mind in the other.

In our investigation, prompting through commented lines does not sit neatly in either case. The text is natural language, and Strudel never executes it. It is addressed to the model, which reads it and acts. But it is also readable by anyone watching, and by the artist writing it. When the model explains its reasoning, the reply arrives in the same display and in the same register, to be read in turn. Thus, the writing acts on the machine and on its readers at once. What remains untested in this paper is how this reads from outside: whether an audience receives the exchange as part of the performance.

5 THE GUNSLINGERS' KNOWLEDGE

The specific musical preferences incorporated into the MCP seek to address gaps and bias in LLM outputs by steering them toward musical knowledge that is embedded in my practice, listening and interpretation.

Before presenting these elements, it is important to contextualize the reasons behind their selection. Originally from Brazil and growing up in São Paulo, I was exposed to many genres, but around the time of the millennium bug, Brazilian funk began to gain national success. Whether intentionally or not, I was deeply influenced by this movement.

In recent years, my work as a composer has shifted toward experimental music—the kind that can be difficult for an uninitiated audience to digest —and is highly focused on the importance of how sounds do (or do not) fit into a narrative. For this reason, two primary directions were explored here: the incorporation of musical structures derived from Brazilian Funk (particularly from the state of São Paulo) and the usage of non-Western scales available within the original Strudel's library.

5.1 The Desert Tips: Brazilian Funk

Nowadays, Brazilian funk is a popular music genre that emerged in the 1980s in the favelas—marginalized communities that still exist—of Rio de Janeiro, influenced by soul music (Frazão, 2021) and styles such as Los Angeles electrofunk and hip-hop freestyle (Jornal da USP, 2024). It can be described as dance-oriented rhythms and patterns, such as tamborzão and mandelão, combined with layered vocal lines and melodic elements. Lyrically, funk often engages with themes related to everyday life, social

realities, and cultural expression within these communities (Frazão, 2021). An example of the recent global expansion of Brazilian funk is the international success of artists such as Anitta, who made history in March 2022 by becoming the first Brazilian artist to reach number one on Spotify's global charts (Guinness World Records, 2022).

Over time, Brazilian funk developed into a diverse musical practice, expanding beyond its origins and establishing distinct regional identities. Consequently, in the city of São Paulo, funk manifests as localized practices that vary across different regions of the city, as shown in Figure 3, rather than a single unified artistic expression. As highlighted in recent discussions in Brazilian media, different geographical city zones, which are the North, South, East, and West, have developed distinct rhythmic and melodic tendencies that emerge organically through listening and are shaped by local producers and cultural dynamics (G1, 2024).

Figure 3: Regional areas of different funk styles across São Paulo, from Funk paulista tem batidas diferentes por regiões de SP (2024).

In the East Zone (Zona Leste), the style often referred to as magrão, is characterized by faster tempos, dense rhythmic layering, and strong low-frequency presence. Such elements tend to emphasize continuous energy and physical impact, often blending elements from earlier carioca funk styles with more recent production approaches (G1, 2024). In contrast, the South Zone (Zona Sul) tends to feature heavier grooves, incorporating "horror" and distorted structural elements, a style commonly called bruxaria funk (G1, 2024).

The North Zone (Zona Norte) is frequently linked to the beat ZN, which reflects a more experimental approach in which funk interacts with genres such as trap and electronic music (G1, 2024). Meanwhile, the West Zone (Zona Oeste), the ritmado often explores more minimalistic rhythmic structures, relying on repetition and subtle variation to create a hypnotic effect, sometimes incorporating timbral influences from instruments such as Brazilian capoeira berimbau and saxophone.

It's important to note that Brazilian funk does not follow compositional paradigms typically associated with Western art music traditions. That means, it is not, in most cases, a practice grounded in written notation or formal institutionalized rules. Instead, it is primarily an oral, performative, and production-based practice, developed through listening, repetition, and experimentation. Also, a key and common musical feature across Brazilian funk practices is the presence of cyclic rhythmic structures that may evoke a "clave-like" organization.

These distinctions are crucial for understanding how such material was translated into musical theory and musical examples in this investigation. Since Brazilian funk is not traditionally notated, the rhythmic material used here was based on the research produced by Souza (2019). This process involved identifying rhythmic patterns, interpreting them for syntax into Strudel REPL code structure, and combining elements of texture, instrumentation, and playability— all by hand —as shown in Figure 4.

Following this, the Strudel code material was further integrated into the MCP, where it becomes accessible to the Claude LLM as structured musical logic. In order to achieve this, Strudel examples are transformed and organized into progressive layers of complexity, or alternatively provided as complete reference examples within the MCP environment.

```
let um= stack(
                    s("tg33_bd:5").struct("[x [~ x]][~ x] [~ x] [x ~]").cutoff(1000),
                    s("tg33_bd:8").struct(" ~ ~ [x ~] ~")
                   )

                let dois= stack (
                 s("tg33_bd:5").struct("[x [~ x]][~ x] [~ [x x]] [x x]").cutoff(1000),
                 s("tg33_bd:8").struct(" ~ ~ [x ~] ~")
                 )

                let tres= stack(
                 s("tg33_bd:8").struct("[x ~] ~ [~ [x x]] [~ x]").cutoff(1000),
                 s("tg33_bd:5").struct(" ~ ~ [x ~] ~")
                 )

                $: arrange(
                [1, um],
                [1, dois],
                [1, um.room(saw.slow(4)).gain(saw.slow(4).range(0.5, 1))],
                [1, um],
                [1, dois],
                [1, um],
                [1, tres.room(saw.slow(4)).gain(saw.slow(4).range(0.5, 1))]
                )
```

Figure 4: Example of beat Zn funk drums into mini notation patterns inside the Strudel REPL environment.

For instance, in the case of drum patterns in Figure 4, the MCP includes three levels of complexity, implemented through Python-based code representations, as shown in Figure 5. These range from minimal rhythmic structures to more intricate and texturally dense patterns, enabling the AI model to generate preferable outputs that evolve in complexity while remaining grounded in the provided musical framework.

```
`arrange(
  [1, stack(
      s("tg33_bd:5").struct("[x [~ x]][~ x] [~ x] [x ~]").cutoff(1000),
      s("tg33_bd:8").struct("~ ~ [x ~] ~")
  )],
      [1, stack(
     s("tg33_bd:5").struct("[x [~ x]][~ x] [~ [x x]] [x x]").cutoff(1000),
     s("tg33_bd:8").struct("~ ~ [x ~] ~")
     )],
     [1, stack(
      s("tg33_bd:5").struct("[x [~ x]][~ x] [~ x] [x ~]").cutoff(1000),
      s("tg33_bd:8").struct("~ ~ [x ~] ~")
      ).room(saw.slow(4)).gain(saw.slow(4).range(0.5, 1))]
)`,
```

Figure 5: Example of a Python-coded beat ZN drum pattern (medium complexity layer) within the MCP server.

Within this framework, the model does not "learn" funk in the conventional machine learning sense. Instead, it interacts with Python-predefined templates and rules– devised by the artist–that reflect the author's own musical practice. This approach shifts general large-scale data training toward a more personal and situated context, in which musical knowledge is reflected, curated, encoded,

and made available for interaction with the Claude LLM. As such, the MCP functions as a mediating layer that enables, to some degree, the incorporation of specific musical practices into large model AI-generated production.

5.2 The Duel Tips: non-Western Scales

In parallel, this investigation also incorporates non-Western scale systems into the MCP environment. The Strudel REPL includes a library of over ninety scale definitions, encompassing a wide range of tonal, modal, and microtonal possibilities. From this collection, six scales—ritusen, pelog, hirajoshi, iwato, enigmatic and prometheus—were selected and integrated into the MCP based on their relevance to my personal compositional interests and artistic identity as an experimental composer.

The selection reflects a practice that prioritizes the sound itself and its affective qualities over conventional tonal frameworks. By introducing non-Western scales into the MCP, which already included options such as major, minor, pentatonic, and blues, the project attempts to expand the range of sonic outputs from the AI-mediated live code. At the same time, it addresses the inherent biases present in many LLMs, which tend to reproduce dominant musical structures found in their training data.

Similar to the incorporation of Brazilian Funk, these musical preferences are selected, interpreted, and translated into Strudel code syntax. These examples are then programmatically implemented in Python code to be introduced as contextual reference knowledge, allowing the Claude LLM to access and combine this specific vocabulary in its musical outputs.

6 THE UGLY'S HORSE: NATURAL LANGUAGE INTO MCP

How can ideas be communicated as clearly and simply as possible without reducing their intent? While the use of domain-specific nomenclature can support this process, language alone may still be insufficient to fully capture a personal vision. Even within fields that possess extensive terminology, such as music, two descriptions of the same idea may lead to significantly different interpretations.

In this investigation, the interaction between the artist and the LLM necessarily occurs through natural language, which serves as the primary interface for communicating one's musical intent. However, descriptors such as stylistic references or affective qualities are inherently ambiguous. Terms like dense or dark do not correspond to fixed or universal representations, but instead depend on context and the experience of one's expressing them.

As a result, natural language prompts must be understood not as a precise instruction, but as a starting point in a broader process of intent translation. Here, the starting point of musical ideas expressed in prompt natural language (e.g., a dense magrão funk rhythm with strong dark emphasis) is interpreted by the artist-researcher and tagged as key musical features inside the MCP. For example, as currently implemented, a scale described as "dark" now has an equal probability of being applied as a minor scale or a newly implemented scale, such as the iwato. At this stage, specific musical knowledge becomes part of the model's operational context, allowing the Claude LLM to access and manipulate it during the generative process. Rather than relying solely on its internal training data, the model interacts with these structured elements and their respective descriptions, producing outputs that are directly guided by these embedded musical preferences.

This process is not linear but iterative and experimental. The translation from natural language intent to functional machine descriptors often requires multiple refinements, as initial representations may present flaws in capturing important nuances of the intended musical idea. Adjustments to code syntax, word descriptors, and rule combinations were necessary to refine the output and achieve musically interesting production.

By continuously refining both the natural language tags and the structured musical material embedded within the MCP, the artist actively steers the generative process. In this sense, the ongoing practice of "word engineering" reflects not only the evolution of Natural Language Processing (NLP) from early rule-based systems such as ELIZA (Weizenbaum, 1966) to contemporary deep learning models like Claude (Le et al., 2024), but also the possibility of shaping in some degree the behaviour of a commercial pre-trained LLM through a designed framework as an MCP server.

6.1 The Horseshoes: Evaluating the Companionship

As Section 3.1 noted, evaluating such systems remains an evolving process. Assessing a collaboration of this kind requires criteria that do not reduce to whether the model produced functioning code. The four below were conceived during the work and applied to the documented sessions. They are subjective by necessity: whether a rhythm carries the intended sound, or a scale sounds *dark* in the sense meant, is a judgement available only from inside the practice. They are also a first attempt, and are offered as such.

Stylistic alignment. Do the model's outputs draw on the material encoded in the MCP or revert to the metrical defaults dominant in its training data? This tests whether the mediating layer does any work at all.

Recoverability. When an output is unwanted, can it be overridden or repaired within a cycle or two, or does the intervention derail what is running? This measures control not as the absence of surprise but as the capacity to respond to it.

Legibility. Is the exchange with the model externalized in the performative code itself? This concerns the location of the interaction.

Continuity. Does interacting with the model sustain or interrupt temporal flow? This assesses latency and context-switching.

7 THE GOOD, THE BAD AND THE UGLY: IMPERFECT RIDE REFLECTIONS

The design and implementation of this investigation were a process of deep artistic self-reflection combined with a technical exercise. I was faced with a challenging task of determining which aspects of my musical practice should be "hard-coded" and which could be delegated to the LLM's generative "token agency". This involved identifying the specific voids in the existing AI-MCP framework and translating those into a mediating layer. What follows organizes what the sessions showed under

the three figures of our metaphor, assessed against the criteria set out above.

7.1 The Good: Delegation and Stylistic Coherence

One effective technique identified during this adaptation was the use of broader alias tagging within the MCP. As an example, by hard-coding variations of natural language prompts, such as tense, dark, primal, haunting, and raw for the iwato scale, the model was provided with a dense semantic web that allowed it to better recognize the user's intent.

Assessed against stylistic alignment, the mediating layer improved the model's generated outputs. A comparison run for this project, documented in the accompanying GitHub repository, contrasts Brazilian funk material generated with and without the encoded context; without it, outputs bore little resemblance to the genre. With the artist-driven descriptors in place, and a set of scales beyond the usual major and minor implemented in the mediating layer, the model opened onto a broader range of possibilities. Its outputs also grew more elaborate, drawing on random modifiers such as every, often and rarely, and on polyrhythmic combinations of Euclidean rhythms. This comparison is necessarily time-bound, and the baseline described here is the one that existed, with the specific Claude model in use, when the sessions were run.

7.2 The Bad: Errors, Divergence and Loss of Control

This investigation also highlighted a phenomenon of "syntax stickiness," where the LLM might latch onto a particular new code syntax found in MCP-provided examples and repeat it throughout an entire session, sometimes resulting in an improvement and sometimes in a completely nonsensical application.

Despite these structured boundaries, working with a generative LLM remains a box of surprises. Even with a more refined MCP over the course of this investigation, the model's outputs are not so commonly predictable. The same prompt sent twice rarely resulted in the same code, unless the user sends the exact Python code in the prompt. A comparable negotiation appears in Lee, Essl and Martinez's account (2016), where one performer describes having to respond to what has appeared on screen; improvising back toward the intended phrase, or trusting the moment and letting it stand.

Assessed against recoverability, the picture is uneven. An unwanted output can also be addressed through the commented lines, asking the model to remove or revise it, but this request is itself unpredictable, and the following output is sometimes worse than the one it replaced. More disruptive still are the occasions when the model erases or overwrites code the live coder wrote, despite explicit instruction not to.

Assessed against continuity, the delay is not in the pipeline. The server's documentation reports pattern writes at 50-80 ms and playback control at 100-150 ms. What interrupts flow is the model's own response time, which is measured in seconds rather than milliseconds and cannot be reduced by improving the automation.

More common than outright failure is divergence. The model sometimes takes a direction I did not envision, or returns something below the level I would have written myself, and the work becomes one of adapting to what arrived on my screen.

While these moments of unpredictability can be frustrating, such as when the model unexpectedly changes the tempo or replaces entire blocks of code without permission, we did not approach them as technical failures. Instead, this research embraced these instabilities as a fundamental part of the artistic experience, where the characterization of what constitutes a meaningful musical output becomes inherently subjective and personal.

7.3 The Ugly: Opacity, Authorship and Platform Dependence

The performative use of the Strudel REPL also shifted the nature of its interaction during this project. By using commented lines in the code editor as a mechanism for prompting, the act of communicating with the AI moved from an external command to an internal performative gesture.

Assessed against legibility, this is the criterion the arrangement satisfies most fully, where the exchange is externalized in the same document as the music. The model can also be asked, through those same commented lines, to explain its reasoning, which then appears alongside the patterns.

However, that transparency has a price. Asking the model to explain itself consumes tokens, and tokens are billed; so the more visible the negotiation, the more the session costs. A practice built on this arrangement pays, in the most literal sense, for the TOPLAP manifesto's demand. It is also partial at best, as the model's explanation is an account of its output, not access to the process that produced it.

The use of commented lines as prompts was designed to align with the manifesto's demand and live coding values to "show us your screens" (TOPLAP, 2004/2024), by exposing the imperfections of the AI "black box" architecture while defining its operational boundaries through the MCP. Whether an audience would read it that way—as the negotiation made visible rather than as an interruption—remains to be tested in performance.

Authorship is unsettled in a related way. A pattern may originate in rhythmic material transcribed from research into São Paulo funk, encoded manually as Python templates, retrieved by a model I did not train, recombined by processes I cannot inspect, and then kept or discarded at my judgment. Thus, this paper takes the position that curation under real-time constraint is itself the authorial act here, and records that the position is a claim rather than a settled matter.

One session took the form of a duel, where the terms were set at the outset: my AI companionship and I would trade patterns, each responding to what the other had left in the editor, and at the end the model would name a winner, itself or me. When it moved in a direction I had not intended, the work became one of turning it back or adapting to it, and the reasoning it printed in the commented lines made that negotiation legible. In the end, the AI declined the choice, and instead, it assigned the roles: I was the Good, it was the Bad, and the Ugly had won.

The judgement is a joke, and it is also the argument of this paper, delivered by the very arrangement it describes. Neither party prevailed; what prevailed was the imperfect thing produced between them. That the model reached

this says nothing about its understanding—it is a plausible completion of a cinematic reference it has certainly read many times. But the fit is exact, and the moment is reported here for what it did to the session rather than for what it revealed about the AI model.

Ultimately, the interactions presented here as "ugly rides," suggest that the value of integrating commercial LLMs into live coding lies in the negotiation of an unresolved relationship between human and machine. This is framed through the metaphor of The Good, the Bad and the Ugly, representing a spectrum of engagement: seeking clarity, navigating conflict, and persisting within the chaos of unpredictable outputs. Rather than evaluating the system solely on its ability to produce "perfect" code, this investigation finds the imperfect ride as a means to a possibility of AI-live coder companionship.

8 LIMITATIONS AND FUTURE RIDES

This investigation rests on documented sessions, in which the criteria were applied by a single practitioner to their own work, configuring a research-creation.

Extensions might follow future work. One is performance before an audience, which alone can test whether prompting through commented lines reads as a performative gesture or as an interruption. Another is a study with multiple live coders using the same MCP server, comparing how different practitioners build prompt vocabularies and whether alias tagging developed by one artist is usable by another. One more is a smaller MCP server, focused on a single aspect of practice without the general music-theory definitions, to see whether a narrower context produces a closer fit.

9 IN THE HORIZON OF CONCLUSIONS

Three claims follow from this ride.

First, reflecting an artist's musical preferences and translating natural language and practice-based knowledge into MCP-compatible structures highlights the central argument of this investigation: meaningful artistic results do not emerge solely from the capabilities of a commercial Large Language Model like Claude, but from the intentional design of the context in which that AI operates.

Second, steering an LLM is knowledge-based as well as a linguistic craft with its own particularities. For example, the alias tagging in the MCP—attaching a fixed vocabulary of descriptive words to particular musical choices—proved to be a successful technique attempted in this paper that directly shapes the generated output. This vocabulary is particular to my own practice and to this corresponding system. Whether it would be legible to another practitioner, or whether another practitioner would build a comparable one, remains untested.

Third, the arrangement compromises precisely where live coding has invested its values. What is gained with such LLM companionship is paid for in transparency, authorship, and independence from a commercial platform—and that includes the monetary cost of tokens to the artist.

Altogether, this returns us to the saddle. Do the manifestos—Generative and TOPLAP—still hold?

"Show us your screens" (TOPLAP, 2004/2024) survives in principle, and prompting through commented lines was designed to honour it. "Give us access to the performer's mind" (TOPLAP, 2004/2024) survives in part; some reasoning is visible, but a second layer is now involved—the "black box" architecture of an AI like Claude—that neither the audience nor the artist can truly inspect. One demand does not survive the ride: the Generative Manifesto's fifth, that we use only software we have written ourselves, on the grounds that authorship cannot be granted to those who have not authored. By that standard, this work is split. The MCP server—the encoded knowledge and vocabulary made by the artist's hands—is authored in exactly the sense the manifesto demands. What it steers is not: the LLM beneath it was written by no one in this paper, and no amount of care in the mediating layer (MCP) changes that.

Thus, what we might propose is not a relaxation but an addition. If the tools of a practice can no longer all be fully inspected by their users, the practice might demand: show us what you did not write. Declare the model, the version, the protocol, the context supplied to it—or any additional information—and what was kept from it. The MCP server accompanying this paper is offered in that spirit—not as a solution that makes an opaque model transparent, which is impossible, but as a full account of the only part of the arrangement the artist actually authored.

Through an ongoing process of reflection, interpretation, encoding, and refinement, artistic expression takes form, enabling human-AI interaction to produce outputs that are not merely functional but, more importantly, meaningful. Within this horizon, between each technological sunset and sunrise where new AIs are introduced, it becomes essential to ask how we can preserve what defines us as artists.

This project suggests that the path forward does not lie in completely rejecting these tools, but in understanding how to shape them according to our own aesthetic visions. In the context of Large Language Models and live coding, this investigation does not advocate either in favour of or against them riding together. Instead, it turns toward "finding beauty in the ugly," embracing and confronting the imperfections, inconsistencies, and bitterness that emerge when a "black box" AI is included in the creative equation.

As in the world of The Good, the Bad and the Ugly, there is no pure hero, no absolute control—only negotiation within a shifting landscape.

ACKNOWLEDGMENTS

This project is available on GitHub as free and open source. The MCP server it builds on is described by its author as an unofficial project, not affiliated with or endorsed by the Strudel project, and is distributed under AGPL-3.0-or-later; the extension inherits those terms.

AUTHOR DECLARATIONS

In the spirit of the demand proposed in Section 9—show us what you did not write—this project used Anthropic's Claude, accessed through the Claude Code CLI, as the companion of this ride, between February and May 2026. Claude and Codex were also used for coding support and structural writing assistance. The ideas presented in this article are the authors' own, and no content was generated without human direction and intervention.

REFERENCES

Anthropic. (n.d.). Anthropic. Retrieved April 20, 2026, from https://www.anthropic.com/

Anthropic. (2024, November 25). *Introducing the model context protocol*. https://www.anthropic.com/news/model-context-protocol

Bernardo, F., Kiefer, C., & Magnusson, T. (2021). Assessing the support for creativity of a playground for live coding machine learning. In J. Baalsrud Hauge, J. C. S. Cardoso, L. Roque, & P. A. González Calero (Eds.), *Entertainment Computing – ICEC 2021* (Lecture Notes in Computer Science, Vol. 13056, pp. 449–456). Springer. https://doi.org/10.1007/978-3-030-89394-1_38

Bol Processor. (n.d.). *Bol Processor*. https://bolprocessor.org/

Collins, N. (2011). SCMIR: A SuperCollider music information retrieval library. *Proceedings of the International Computer Music Conference*.

Collins, N. (2015). Live coding and machine listening. *Proceedings of the International Conference on Live Coding*, 4–11. https://iclc.toplap.org/2015/meta/5_Live_Coding_and_Machine_Listening.pdf

Frazão, M. F. C. (2021). *Funk no Brasil*. Universidade Estadual de Campinas (UNICAMP). https://www.iar.unicamp.br/wp-content/uploads/2021/07/V3_ED05_A3_funknoBrasil.pdf

G1. (2024, June 29). *Funk paulista tem batidas diferentes por regiões de SP e reconhecimento internacional: "Não é coisa de drogado, bandido, é arte."* https://g1.globo.com/sp/sao-paulo/noticia/2024/06/29/funk-paulista-tem-batidas-diferentes-por-regioes-de-sp-e-reconhecimento-internacional-nao-e-coisa-de-drogado-bandido-e-arte.ghtml

Guinness World Records. (2022, July 18). *Rio pop star Anitta becomes first solo Latin artist to reach No. 1 on Spotify*. https://www.guinnessworldrecords.com/news/2022/7/rio-pop-star-anitta-becomes-first-solo-latin-artist-to-reach-no-1-on-spotify-710368

Jiang, J., Wang, F., Shen, J., Kim, S., & Kim, S. (2026). A survey on large language models for code generation. *ACM Transactions on Software Engineering and Methodology*, *35*(2), Article 58. https://doi.org/10.1145/3747588

Jornal da USP. (2024). *"É som de preto, de favelado": A evolução do funk no Brasil*. https://jornal.usp.br/diversidade/e-som-de-preto-de-favelado-a-evolucao-do-funk-no-brasil/

Karpathy, A. (2025, February 2). *There's a new kind of coding I call "vibe coding"* [Post]. X. https://x.com/karpathy/status/1886192184808149383

Knotts, S., & Paz, I. (2021). Live coding and machine learning is dangerous: Show us your algorithms. *Proceedings of the International Conference on Live Coding.* https://iclc.toplap.org/2021/

Le, D.-V.-T., Bigo, L., Keller, M., & Herremans, D. (2024). *Natural language processing methods for symbolic music generation and information retrieval: A survey*. arXiv. https://doi.org/10.48550/arXiv.2402.17467

Lee, S. W., Essl, G., & Martinez, M. (2016). Live writing: Writing as a real-time audiovisual performance. *Proceedings of the International Conference on New Interfaces for Musical Expression*, 212–217. https://doi.org/10.5281/zenodo.1176060

Leone, S. (Director). (1966). *The Good, the Bad and the Ugly* [Film]. Produzioni Europee Associate.

Magnusson, T. (2014). Herding cats: Observing live coding in the wild. *Computer Music Journal, 38*(1), 8–16. https://doi.org/10.1162/COMJ_a_00216

Navarro Del Angel, L., & Ogborn, D. (2017). Cacharpo: Co-performing cumbia sonidera with deep abstractions. *Proceedings of the International Conference on Live Coding*, Morelia, Mexico.

Python Software Foundation. (n.d.). *Python* [Computer software]. https://www.python.org/

Reppel, N. (2020). The Mégra system: Small data music composition and live coding performance. *Proceedings of the 2020 International Conference on Live Coding*, 95–104.

Souza, T. (2019). *Estruturas e sonoridade afro latentes no funk*. In *Jornada de Pesquisa em Arte UNESP PPG IA 2019 – 3ª edição internacional*. Universidade Estadual Paulista (UNESP).

Stewart, J., & Lawson, S. (2019). Cibo: An autonomous TidalCycles performer. *Proceedings of the Fourth International Conference on Live Coding*.

Strudel. (n.d.). *Getting started*. Retrieved April 20, 2026, from https://strudel.cc/workshop/getting-started/

TidalCycles. (n.d.). *Tidal Cycles*. Retrieved April 20, 2026, from https://tidalcycles.org/

TOPLAP. (2024). *Manifesto draft*. TOPLAP Wiki. https://toplap.org/wiki/ManifestoDraft (Original work published 2004)

TOPLAP. (n.d.). *TOPLAP blog*. https://blog.toplap.org/

Uzu. (n.d.). *Uzu: Learning and sharing patterns with Tidal, Strudel and friends*. Retrieved August 13, 2026, from https://uzu.lurk.org/

Ward, A., & McLean, A. (2000). *The Generative Manifesto*. Institute of Contemporary Arts, London. https://slab.org/2015/07/28/the-generative-manifesto-august-2000/

Weizenbaum, J. (1966). ELIZA-A computer program for the study of natural language communication between man and machine. *Communications of the ACM, 9*(1), 36–45. https://doi.org/10.1145/365153.365168

Wilson, E., Fazekas, G., & Wiggins, G. (2023). On the Integration of Machine Agents into Live Coding. *Organised Sound*, *28*(2), 305–314. https://doi.org/10.1017/S1355771823000420

Xambó, A. (2021). *Virtual agents in live coding: A short review.* arXiv. https://arxiv.org/abs/2106.14835

Zujkowski, W. (2025). *Live-coding-music-mcp: A Model Context Protocol server for AI-assisted live coding music* [Computer software]. GitHub. https://github.com/williamzujkowski/live-coding-music-mcp