BeatMohan's Symphony: Designing a Game Environment to Scaffold Pattern-Based Musical Organization

Noel Alben
noelalben@gatech.edu
Georgia Institute Of Technology
Atlanta, U.S.A

Jason Freeman
jason.freeman@gatech.edu
Georgia Institute Of Technology
Atlanta, U.S.A

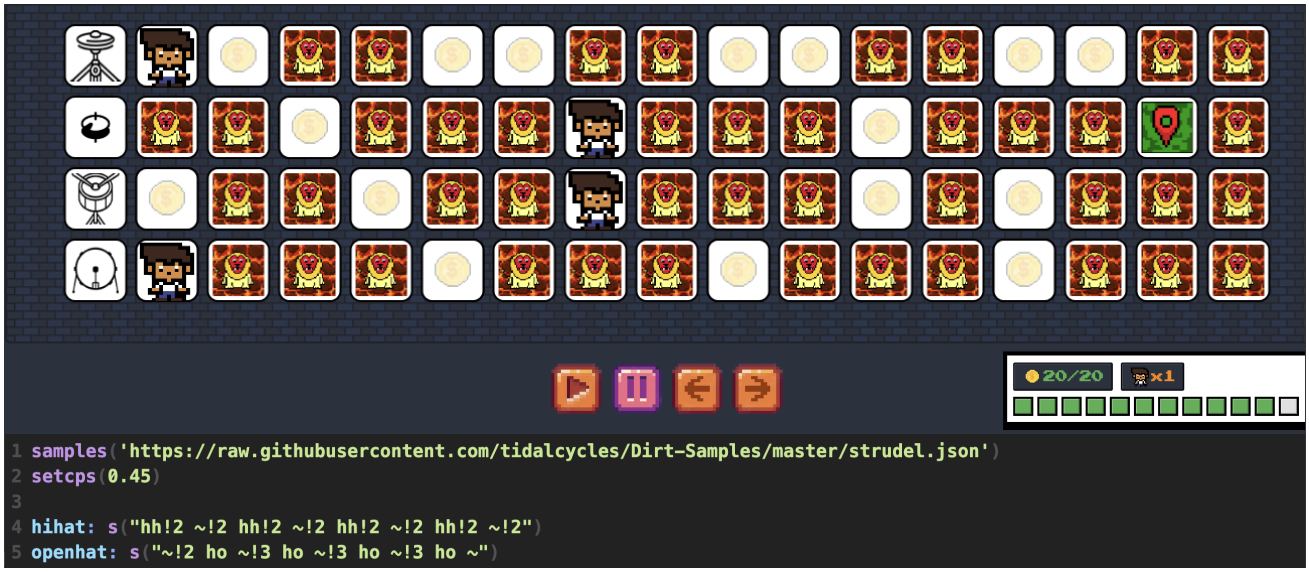


Figure 1: BeatMohan's Symphony gameplay interface

Abstract

BeatMohan's Symphony is a 2D musical game environment designed to scaffold pattern-based music organization through live coding. The work is grounded in a tradition established by Jeanne Bamberger, whose music pedagogy techniques emphasize active engagement with musical structures to support the development and refinement of learners' mental models for music making.

We translate this experience design learning framework into a goal-directed musical game in which pattern-based live coding functions as both the primary musical interaction and the core gameplay mechanic. The 2D character, *BeatMohan*, visually anchors the musical events. Learners modify code to guide the character's movement in response to spatial and rhythmic constraints presented at each level. The game's modular architecture is intended to allow educators to foreground specific aspects of pattern-based music organization.

This paper presents the design rationale and implementation of *BeatMohan's Symphony*. We report on a preliminary qualitative analysis of user feedback, examining whether learner interactions align with the intended pedagogical framing. Through gameplay observations and post-activity interviews, we analyzed how participants interacted with the environment, interpreted constraints, and responded to in-game feedback when organizing rhythmic patterns through code. Our analysis suggests that the system provides a context for eliciting and observing the development of musicality within a short gameplay session.

Keywords

live-coding, educational digital music interface, construction-ism, interaction design

1 Introduction

Modern text-based interfaces for live coding offer musicians a precise and concise means to define and manipulate musical patterns. In live coding systems such as TidalCycles and Strudel, musical patterns are defined as abstract, time-based structures that describe when musical events occur [16, 30]. Musicians use these systems in performance contexts to write code that is interpreted in real time to generate musical patterns. The music then evolves through iterative modifications of the code—live, in front of an audience [16, 18]. For novice learners, educational text-based live coding interfaces provide a medium to render musical thinking in code. They are designed to scaffold the abstract vocabulary of writing music in code and have proven to be effective music learning platforms as they support iterative experimentation where modifications to the code result in immediate aural feedback [6]. *BeatMohan's Symphony* adopts pattern-based live coding as the core mechanic for an educational 2D puzzle game. Players write short musical patterns as code into an editor console, which animates a 2D character *BeatMohan*, who moves across a grid map that resembles a rhythmic sequencer. Each level represents a rhythmic puzzle that must be solved by encoding an appropriate musical pattern in code.

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

This work is informed by Jeanne Bamberger's seminal music education research and *the experience design* framework, which emphasizes that the design of interactive environments must support learners' musical thinking through direct manipulation of musical structures (patterns of rhythmic and/or melodic elements) [4]. Central to this framework is the principle that engaging with multiple representations for musical structures can lead to a deeper musical understanding; *BeatMohan's Symphony* presents rhythmic patterns as manipulable code alongside a visual sequencer-like puzzle map, in which character motion is directly coupled to pattern evaluation. Together, these representations are intended to support exploration and the development of what Bamberger terms musicality — the ability to listen for, recognize, and organize musical structures in order to create new ones [1].

In this paper, alongside the system's design, we present a pilot qualitative study to examine how novice participants learned to structure rhythmic patterns after a short play-through of the game. Participants progressed through a sequence of game levels designed to scaffold the representation of simple rhythmic patterns in code, followed by post-activity interviews and a free-form music improvisation task using a four-voice drum sequencer. Throughout these activities, we analyzed how the learners interpreted the gameplay constraints, responded to in-game feedback, and reasoned about timing, repetition, and layering—structural components that directly inform the interpretation of rhythm [10].

The paper is structured as follows: We first detail the related works that served as design motivations and inspiration for the representational choices and interaction mechanics underlying *BeatMohan's Symphony*. We then describe the implementation of the system and outline the qualitative methodology used to examine whether these design decisions achieve their intended pedagogical outcomes. We position this work as an early-stage investigation into the system's capability to surface specific mental models for organizing rhythmic patterns through code, establishing *BeatMohan's Symphony* as a platform that will support future research into pattern-based musical learning with live coding.

2 Related Works

The design and implementation of *BeatMohan's Symphony* is influenced by existing visualization models of live coding in musical spaces, the pedagogical techniques outlined by the Experience Design framework, and gamified music-learning platforms.

2.1 Visualization of Live Code in Music Education

Over the past decade, numerous languages and environments have been developed to support text-based live coding, with techniques and tools continually evolving to support broader participation within the community [3]. Educationally oriented live-coding environments such as Sonic Pi, Jython Music, and Gibber adopt simplified musical notations that allow novices to create, modify, and investigate abstract musical structures with immediate auditory feedback [14, 26, 28]. However, for a non-programmer looking at the code alone to make sense of the musical logic and structure is challenging. Many systems integrate **visualizations** that are responsive to the musical patterns to expose the internal processes of live code and connect it to the music during a performance, thereby enhancing interpretability for non-programmers [15]. Web-based live coding environments such as Gibber and Strudel also offer a secondary syntax that highlights elements of the code to serve as parallel representational layers communicating how music generated by the code unfolds over time [28, 30].

Beyond direct visualization of the music, agent-based animations offer an alternative strategy for representing musical processes through character movement within a spatial environment [29]. Early examples, such as Griffiths' *Al Jazari*, allowed users to control robots moving across a 3D terrain to complete loops that trigger audible musical patterns, with thought bubbles illustrating the commands guiding their actions [8]. Such systems externalize temporal processes through synchronized movement, providing spatial metaphors for musical patterns along with the logic underlying them.

BeatMohan's Symphony is developed as an educational live coding game environment inspired by platforms such as Sonic Pi and EarSketch, which have been shown to provide a positive and impactful introduction to live coding for novices. They have also proven effective in fostering engagement among broad communities [7, 26]. While these platforms primarily emphasize code-to-sound interaction, *BeatMohan's Symphony* explores live coding within a goal-directed, puzzle-based environment where the visual constraints of solving the puzzle shape the resulting rhythmic pattern. The game is built upon the Strudel live coding engine, borrowing its mini-notation and syntax highlighting to support code interpretation. We combine Strudel's secondary syntax visualizations with musically synchronized movements of a 2D character to provide additional representational layers that support pattern organization through comparison, linking symbolic code to the evaluated audio and animated rhythmic events.

2.2 Experience Design Framework

Seymour Papert's theory of *Constructionism* proposes that learners build knowledge most effectively by creating artifacts through active engagement with tools and media relevant to their domain [22, 24]. Influenced by Constructionism, Jeanne Bamberger integrated computing technologies into music education research, enabling novices to engage with simple musical structures of rhythm and melody in ways that draw on their intuitions to support the development of musicality [1, 4].

These systems were further treated as epistemic tools, allowing researchers to investigate how learners develop an understanding of musical organization through structured activities embedded within software environments. Observations of learner interaction with the system were paired with situated conversations following the activities, in which participants articulated and reasoned through their musical choices and the system's output. These discussions supported learners' musical understanding and contributed to researchers' investigation of how musical reasoning developed through interaction with the system. Analysis of interaction data and post-activity conversations informed the iterative refinement of Bamberger's music education systems *MusicLOGO* and *Impromptu* [2, 4].

Andrew Brown formalizes this iterative, practice-led methodology as *Experience Design*, comprising three intertwined activities: (1) designing musical tasks aligned with specific learning goals, (2) developing interactive software to support those tasks, and (3) observing learners engaging with the system to guide ongoing refinement. A canonical example of this approach is his generative improvisation system *jam2jam* [4].

BeatMohan's Symphony adopts this lineage by translating the *Experience Design* framework into a puzzle game that scaffolds writing code to represent rhythmic patterns. Bamberger argued that computational representations can serve as cognitive tools for musical thinking, enabling learners to externalize, test, and revise their intuitions about musical organization through writing code and reflection on evaluated audio [2]. *BeatMohan's Symphony* intentionally foregrounds learning goals related to rhythmic timing, repetition, layering, and groove organization, all within the constraints of the Strudel live coding engine. In alignment with the *Experience Design* framework, our qualitative methodology aims to observe how learners engage with the game, and we examine their interactions and post-activity reflections to understand emerging mental models for representing rhythm through code. Our analysis identifies which elements of the system support their understanding, and we use these insights to guide iterative refinement of the design.

2.3 Game-Based Learning and Musical Interaction

In contrast to the open-ended musical environments of *Impromptu* and Jam2Jam discussed above, *BeatMohan's Symphony* embeds musical activities within goal-directed gameplay, using puzzle constraints and structured level progression to focus attention on specific learning goals. The gameplay provides a context for engaging in musical tasks, while the level progression scaffolds each rhythmic concept and its representation in code. This integration of musical activity within a game structure positions the system within broader discussions of game-based learning and musical interaction. Game-based approaches in music education have been shown to support learning across notation, rhythm, performance, and composition, often without requiring learners to explicitly focus on formal instruction [12].

Many music games embed musical concepts directly within gameplay mechanics, allowing understanding to emerge through repeated interaction and problem-solving rather than through explicit theoretical explanation. For example, in the game Mario Paint, the Composer mode presents composition as a spatial puzzle in which symbolic musical events are placed on a grid, encouraging reasoning about timing and sequence through playful construction [21]. Rhythm-based games such as Beat Saber, Just Dance, and Fitness Boxing emphasize timing, anticipation, and coordination, supporting implicit learning and transfer in areas such as rhythmic perception and sensorimotor synchronization [11]. Other systems, including Rocksmith and Song2See, integrate physical instrumental performance directly into gameplay, where musical skill development emerges as a byproduct of goal-directed interaction and immediate feedback [25].

Inspired by musical games and learning through play, *BeatMohan's Symphony* leverages goal-directed gameplay to engage learners in musical activities through problem solving. Puzzle constraints and level progression provide a structured environment to revise the code, not solely for the purpose of making music but within the context of solving in-game challenges. In this paper, we examine how this game-based context functions within the broader Experience Design framework and whether embedding musical tasks within gameplay introduces a distinct layer of engagement that shapes how learners interact with and reason about the musical activity.

Figure 2: Interface of *BeatMohan's Symphony*, showing the animation panel (top:1) and the live-coding console (bottom:2).

3 Interface Design

The interface of *BeatMohan's Symphony* consists of two primary components, shown in Figure 2:

(1) The animation panel, which displays the puzzle layout, in-game instructions, BeatMohan's movement, remaining lives, and coin progress.
(2) A live-coding console through which users interact with and manipulate rhythmic patterns.

3.1 Animation Panel

The visual aesthetics and interaction style of the animation panel draw inspiration from 8-bit platform and puzzle games such as *Donkey Kong*, *Super Mario Bros.*, and *Baba Is You* [19, 20, 23]. These games are characterized by simple rules and spatial constraints that govern character movement, where incorrect or mistimed actions often result in collisions that interrupt gameplay and trigger a reset, prompting players to reassess their strategy.

Mohan's animations are confined to a grid which resembles a step-based drum sequencer. Each horizontal tile corresponds to a position in the rhythmic cycle. When an onset is heard, Mohan moves to the corresponding tile. If that tile contains a coin, the beat continues; if the onset corresponds to a lava-lion, the beat is interrupted and Mohan is reset to the start, signaling that a rest should have been placed at that position. In Figure 2 the character's movement is synchronized with the playback of the kick.

Each level is defined by a structured JSON object that specifies three primary elements:

- The starting code state.
- The target rhythmic pattern (encoded as an array of symbolic onsets and rests).
- The visual layout of the puzzle, including instruments, coins, lava-lions, and the goal flag.

The top panel also displays gameplay status information, including the number of coins collected, Mohan's remaining lives, and overall level progression. In this iteration of the game, there are no penalties for reaching 0 lives; instead, we provide a leaderboard to showcase how many lives were saved, incentivizing players to save as many lives as possible. Audio evaluation is controlled through play and pause buttons, while the arrow navigation buttons allow users to move between levels.

3.2 Live Coding Console

Interacting with a code editor to generate audiovisual output follows a design lineage found in systems such as Replicube, Scratch, and EarSketch, where the code editor and its resulting

audiovisual output are displayed simultaneously within a single interface. Although presented in separate panels, both the symbolic code and its visual and sonic effects remain visible at all times, allowing users to directly compare their edits with the resulting behavior [7, 9, 27].

Users interact with a browser-based code editor that supports continuous editing, evaluation, and syntax highlighting of live-coded rhythmic patterns. The editor is built upon Strudel's pattern engine, allowing authored code to be evaluated in real time and immediately rendered as sound [30]. Strudel is a browser-based implementation of the TidalCycles approach to live coding, bringing Tidal's pattern representation, transformations, and mini-notation into a text-driven environment for encoding and manipulating musical patterns on the web [16, 30].

Figure 3: Synchronized representations of rhythmic structure. A highlighted bd (kick drum) event in the Strudel console corresponds to Mohan landing on the associated tile in the animation panel.

The decision to use Strudel was motivated by two considerations. First, it provides an authentic and transferable live coding experience grounded in a system used in contemporary live coding practice [5]. Second, Strudel's syntax highlighting emphasizes elements of the code in synchrony with the musical events they encode. When combined with the real-time animation of the puzzle layout described above, this creates multiple coordinated representations of rhythmic structure unfolding together in musical time. The live-coded patterns evaluated in the console directly drive Mohan's movement within the animation panel, linking symbolic input to spatial and auditory feedback, as shown in Figure 3.

4 Implementation

BeatMohan's Symphony is implemented in JavaScript. The core game engine couples 2D animation rendered in an HTML5 canvas with musical events evaluated by Strudel. Strudel evaluates an entire rhythmic pattern over a fixed duration referred to as a *cycle*. A cycle represents one complete iteration of the authored pattern from its beginning to its end. Users control how quickly patterns repeat by setting the number of cycles per second.

As Strudel's internal scheduler advances through each cycle, it queries the pattern and generates time-stamped musical events, referred to as *Haps*. Each Hap contains both musical information, such as the instrument to be triggered, and timing information indicating when the event should occur. The timing of a Hap is represented as a phase value ranging between 0 and 1, specifying the event's relative position within the current cycle, with 1 being at the completion of a cycle.

Our implementation is intentionally designed to scale across different animation metaphors and gameplay contexts. By separating pattern evaluation from gameplay logic, we can modify how musical events are mapped to visual elements, rules, and feedback without changing the underlying live-coding or scheduling system. This makes it possible to design entirely new levels or alternative visual representations while preserving the same musical timing infrastructure. A demo of the interface can be found here.[1]

4.1 Constraining the Musical Domain for Study

In the current iteration of the system, we deliberately restrict the musical activity space of *BeatMohan's Symphony* to drum-machine–style, grid-based, multi-voice rhythmic pattern construction. Although the underlying live-coding engine can support a wide range of musical expression, this study focuses specifically on stacked percussive patterns inspired by step sequencers and 808-style paradigms.

This constraint allows us to design puzzle activities that scaffold writing rhythmic patterns in code and foreground organizational concepts such as timing, repetition, layering, and the functional roles of percussive elements. Narrowing the domain in this way limits direct generalization to melodic or more open-ended live coding practices. However, it enables a more focused investigation of how learners construct, articulate, and revise mental models of pattern-based rhythm through interaction with code, animation, and gameplay constraints.

5 Experience-Centered Activity Design

To engage with the system, we recruited participants with little to no prior experience authoring rhythmic patterns in code or producing drum-based beats. Each participant completed a single session lasting approximately 45–60 minutes.

At the beginning of the session, participants completed a brief questionnaire about their musical background, adapted from the Goldsmith Musical Sophistication Index (Goldsmith-MSI) [17]. This measure was used to contextualize participants' prior musical experience and situate their interactions with the system along a broader spectrum of musical engagement.

Participants first interacted with a simple web-based drum sequencer, shown in Figure 4, for approximately three minutes. The interface reflects common step-based drum sequencers found in contemporary music production tools. They were invited to freely construct rhythmic patterns without additional rules or imposed constraints. This activity allowed us to observe how participants initially approached pattern construction in a sequencer paradigm, including the assumptions they made, the strategies they employed, and how they organized rhythmic material prior to interacting with *BeatMohan's Symphony*.

Participants then engaged with a fixed sequence of twelve levels in *BeatMohan's Symphony*. All participants completed the same progression to ensure consistency of experience across sessions. Gameplay interactions and audio output were recorded to document participants' strategies, revisions, and decision-making processes.

Following gameplay, participants responded to a set of open-ended reflection prompts (see Table 1). These prompts invited them to describe what they learned about rhythmic patterns during interaction, how they made decisions about rhythm and pattern placement, and which musical ideas or strategies became salient through writing rhythmic patterns in code. Questions were asked in direct reference to participants' gameplay, often building on observations of their actions or comments during

[1] Interface Demo

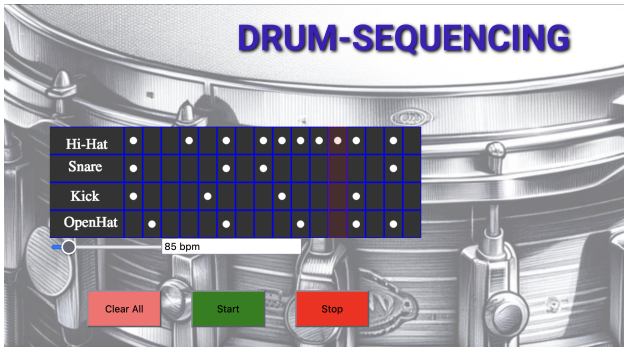


Figure 4: Web-based drum sequencer, the interface presents a four-voice, step-based grid (Hi-Hat, Snare, Kick, OpenHat) with a 16-step cycle, tempo control, and playback buttons.

the session. Prompts were adapted conversationally to maintain continuity with what participants had already articulated, rather than delivered as a fixed survey script. The interview was conducted as a reflective dialogue to encourage elaboration and reasoning about their musical and coding choices.

As a concluding activity, participants returned to the same web-based drum sequencer for approximately three minutes to construct additional rhythmic patterns. In several sessions, this second interaction followed immediately after the initial couple of reflection questions, allowing participants to articulate their thinking and then apply those ideas directly within the grid interface. This ordering enabled us to observe how verbalized reasoning translated into action and how newly articulated ideas about rhythm were enacted outside the gameplay context.

5.1 Gameplay Progression and Scaffolding

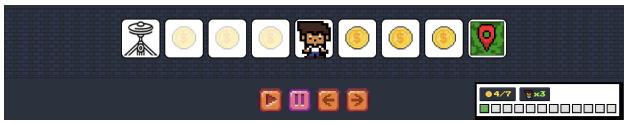
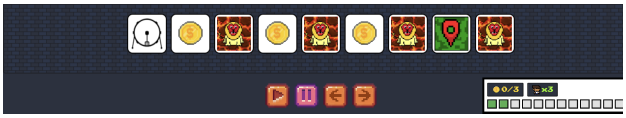


Figure 5: Tutorial level introducing synchronization between code evaluation, audio playback, and character movement.

The tutorial level introduced the relationship between code, sound, and animation (Figure 5). Participants evaluated a pre-completed 8-onset pattern and observed how code execution simultaneously triggered audio playback and Mohan's synchronized traversal across the grid. On-screen instructions directed attention to editable parameters, syntax highlighting, and tempo controls.

The second level introduced failure feedback through the lava-lion obstacle (Figure 6). A single incorrectly placed onset caused a collision, requiring participants to revise the code according to the on-screen instruction beneath the comment line. This established the core interaction loop of inspecting visual constraints, editing symbolic rhythm, and re-evaluating the pattern.

Subsequent levels introduced parallel character movement and multi-layer rhythmic organization. Puzzle 4 introduced unknown tiles that could resolve as coins, lava-lions, or the goal flag (Figure 7). These elements were not visually revealed until evaluation,



(a) Level layout with lava-lion obstacle.

```
samples('https://raw.githubusercontent.com/tidalcycles/Dirt-Samples/master/strudel.json')
setcps(0.6)

// change a bd below to '~' to quietly avoid the lava-lion
kick: s("bd ~ bd ~ bd bd bd ~")
```

(b) Corresponding code requiring revision.

Figure 6: Early level introducing failure feedback and pattern revision.

Figure 7: Multi-layer level introducing unknown tiles that resolve during evaluation.

requiring participants to anticipate rhythmic placement rather than rely solely on visible confirmation. Combined with a limited life counter and a running total of collected coins, this design discouraged random trial-and-error and encouraged deliberate reasoning about event spacing across the cycle. Participants were required to anticipate where onsets and rests should occur before evaluating the pattern. This reasoning became more complex as the puzzles progressed and expanded to 16 slots per voice.

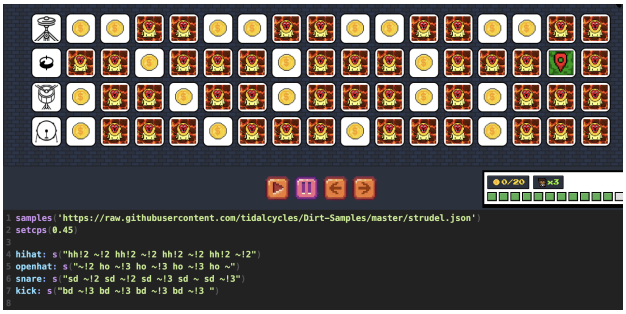


Figure 8: Multi-layer level introducing repetition operators in the code editor.

The final levels introduced Strudel repetition operators (e.g., `sd!5` (Figure 8)), shifting from event-by-event counting to more compact symbolic representation. This progression moved participants from direct spatial matching toward more abstract reasoning about rhythmic structure. All of our rhythmic patterns were adapted from Pocket Operations, a curated collection of drum machine patterns spanning genres including Latin grooves and hip-hop [32].

The progression described above was intentionally structured to foreground specific aspects of rhythmic organization through constrained interaction. In the following section, we examine how participants engaged with these tasks and how their reflections

reveal emerging mental models of rhythm and code-based pattern representation.

6 Observations and Discussion

Five participants completed the full study protocol. Three participants had no formal musical training, while two participants had more than ten years of formal training. One was trained in Western classical piano, and one in Indian classical music. None had prior experience writing rhythmic drum patterns or using live coding environments. The observations presented below are organized into recurring themes identified across gameplay interactions, sequencer activities, and post-activity conversations. Themes were identified through an inductive thematic analysis of session recordings and post-activity interview transcripts, coded using Atlas.ti [31]. Codes were then grouped into recurring patterns across sessions. The proceeding subsections go over the relevant themes that emerged in detail.

Given the small sample, we cannot make any generalizable claims on the comparative effectiveness of our system over other live coding scaffolding techniques; however, our analysis treats learner behavior, post-activity reflections, and breakdowns as evidence that reveals how the system helps surface musical reasoning. This aligns with work in the NIME community that frames digital musical interfaces as epistemic instruments: tools that externalize and make observable the knowledge they help construct [13].

6.1 From Random Placement to Patterned Organization

Across the participants with little or no prior music experience, initial interactions with the drum sequencer were exploratory and often unsystematic. Participant 2 described their first attempt as "just putting the dots wherever I could," noting that it sounded "just like noises playing in my head." Their initial sequencer artifact reflects this description, with sparse placement and limited repetition across layers (Figure 9).

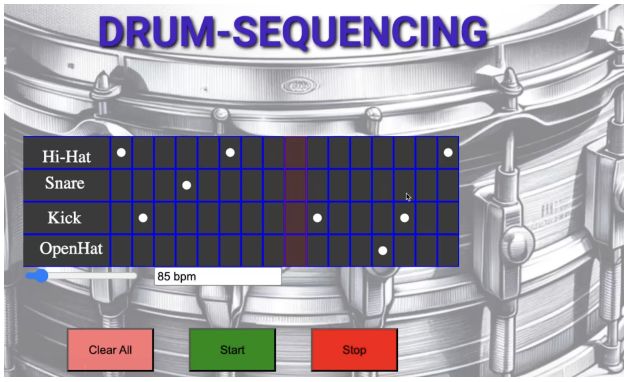


Figure 9: Participant 2's pre-game sequencer pattern. Onsets are sparsely and irregularly distributed across layers, with limited repetition or structural organization.

After progressing through the game levels, Participant 2 described approaching the sequencer differently:

> "I noticed that there was a pattern in almost every level… and it sounded much more palatable as compared to the first time when I did it. So then… I kind of tried to make it interesting."

When returning to the sequencer, this participant organized onsets with clearer repetition across layers and more consistent spacing. Rather than filling slots freely, they appeared to draw on patterns encountered during gameplay (Figure 10).

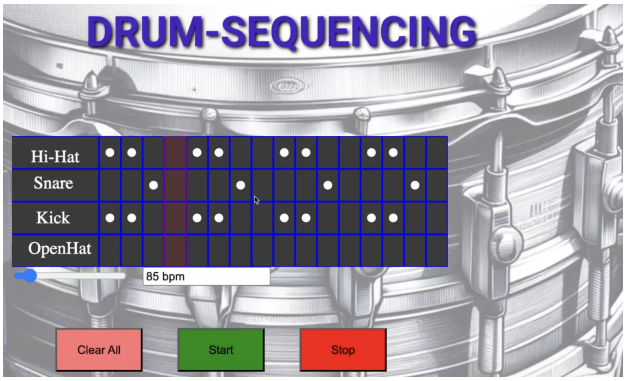


Figure 10: Participant 2's post-game sequencer pattern. Compared to their initial attempt, the pattern exhibits clearer repetition across layers and more consistent spacing of onsets.

After gameplay, participant 3 described rhythm construction as "just math" and as "knowing when a beat should occur," framing pattern building as reasoning about position within a cycle rather than random placement.

The musically trained participant (who had never made beats on a sequencer before) described the grid as a familiar layout for interpreting music, explaining that it allowed them to "split it up into different fractions, and then decide where a sound happened and where it didn't." Suggesting that the visual representation resonated with their prior experience of reading and subdividing Western musical notation.

Across all our participants, compared to their initial interaction with the sequencer, playing the game was followed by more deliberate organization of rhythmic onsets. Participants also regularly referenced musically significant concepts such as repetition, spacing, layering, and expectation in their reflections.

6.2 Layering and the Emergence of Instrument Roles

Participant 5, who had no formal musical training, had begun noticing that certain instruments "always have very simple beats, and you just repeat it," referring to the kick and hi-hat. They explained that once those parts were established, "the other two, this is where I can experiment." When returning to the sequencer, this participant intentionally experimented with a repetitive foundation using the kick and hi-hat before adding variation to the remaining layers.

Participant 4 similarly described wanting each drum to have its own "individual part" and noted that adding too many elements in the same layer "takes your attention away." They reflected that some sounds would "overlap with what I wanted to hear," leading them to remove events to clarify the groove. Across participants, instrument rows were described as having "roles," "stable parts," or "personality," indicating that rhythmic organization was understood as coordination between multiple interacting layers rather than isolated event placement.

These accounts suggest that the scaffolded progression supported a shift from editing single onsets toward reasoning about

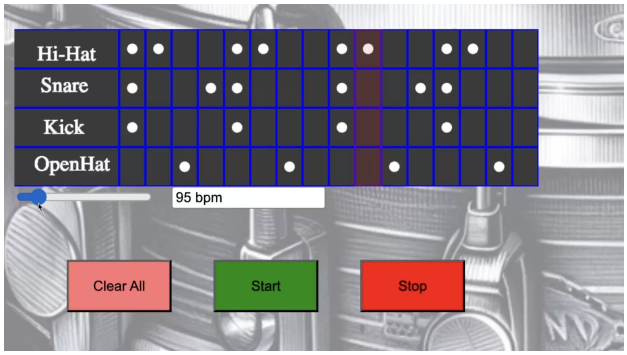


Figure 11: Participant 5's post-game sequencer pattern. The kick and hi-hat layers form a repetitive structural backbone, while other layers introduce variation.

the functional relationships between rhythmic layers within a short game-play session. Especially in the levels with unknown tiles, participants were required to consider how each layer contributed to the overall rhythmic structure. Since the count of their lives lost was shown constantly, participants described becoming more deliberate and error-avoidant, anticipating how each onset placement would affect the outcome before evaluating the pattern.

6.3 Reasoning Through Symbolic Representation: Intentional Pattern Construction in Code

Across participants, writing rhythmic patterns in code was associated with more intentional decisions about placement, spacing, and structure. Unlike the sequencer, where events could be toggled directly on a grid, the live-coding environment required participants to explicitly encode onsets and rests symbolically before hearing the result.

Several participants described inspecting the code before pressing play. Participant 3 explained that they would "check the code and see if it matches what's available" in order to avoid collision. Participant 4 experienced repeated lava-lion collisions on a near-complete level, which had no obvious visual indication of what was wrong. It was only through inspecting the code that they identified the issue: "the part of the drum was missing" — one layer had fewer onsets than the pattern required. Participant 5 used the **cps** function in strudel and slowed playback to follow isolated events more closely, this was done unprompted. They noted that hearing alone was sometimes insufficient and that they needed to examine the code and Mohan's possible positions slowed down.

These behaviors indicate that the code functioned as a representational space for rhythmic reasoning. Participants inspected it, compared it against visual constraints, and revised it deliberately. Rather than reacting only to auditory output, they started anticipating outcomes and spent time analyzing the animation grid and symbolic pattern before evaluation.

The introduction of repetition operators further extended the value of writing the patterns in code. Earlier in the session, participants manually counted visible grid positions and tracked onsets and rests across the cycle. When repetition syntax was introduced '!', Participant 2 reflected that "especially with the counts and all of that later on, that made it even more interesting and simpler also to kind of calculate the beat." The repetition operators formalized an activity that participants were already performing. Instead of enumerating each onset individually, they could encode their counting strategy in a compact symbolic form.

6.4 Game Structure as Motivational Context

The goal-directed structure of the puzzles played a central role in how participants engaged with the musical tasks. For some, the problem-solving frame made the activity feel accessible even without prior musical or coding experience. Participant 3, who has been exposed to music making software with very little experience using them, explained that they preferred the coded, problem-solving format: "I liked that it was coding instead of just a really easy user interface… being engaged in the problem solving of it made me want to play the game." They added that if it were simply "playing around on GarageBand," they would have found it overwhelming.

Participant 2, who had no formal musical training, described the experience as an introduction to both coding and rhythm. At the beginning of the session, they expressed doubt about being able to complete all the levels, but later reflected: "If you had shown me this at the beginning and told me I'd be able to code this pattern, I would not have believed it." The structured progression appeared to support a sense of capability that emerged through successfully completing each puzzle.

Participant 5 similarly described the system as an approachable entry point into coding music, stating that they "wish they had this in school." They noted that the gradual introduction of syntax and repetition made the coding feel "user-friendly" rather than intimidating. The puzzle format provided a clear objective, which they described as helping them stay focused: "If it was too creative, I might just get distracted, but having that challenge kept me going."

Across participants, the game structure provided a concrete context for interacting with rhythmic ideas. *Mohan's* collisions and interruptions of the audio often prompted participants to ask why certain placements worked or failed. These moments prompted brief discussions about rhythmic spacing, accent, pulse, and instrument roles, grounded directly in their gameplay experience. The puzzle structure supported engagement while also creating opportunities for musical concepts to be introduced in response to participants' own reasoning and questions.

7 Limitations

This study involved five participants in single-session interactions lasting approximately 45–60 minutes. The observations presented here are qualitative and exploratory in nature. Thematic coding was conducted by a single researcher without inter-rater reliability checks, which is a recognized limitation of the analysis. Future studies will include multiple coders to strengthen the validity of theme identification. We did not conduct a controlled learning experiment isolating specific variables or comparing *BeatMohan's Symphony* to other rhythm-learning platforms. As such, we do not claim measurable gains in rhythmic proficiency or live-coding skill.

Instead, the present work examines whether the system functions as intended within the Experience Design framework: to foreground specific live coding and rhythmic pattern organization learning goals, which in turn surface learners' understanding through game-play and post-activity conversation. While participant interactions were broadly positive, moments of friction and interface difficulty were also observed; these are addressed

in the Conclusions as areas for iterative refinement. Although limited in scale, the findings provide early evidence that the game-based context for live-coding meaningfully shaped participants' musical engagement.

8 Conclusions and Looking Ahead

This study represents one iteration in an ongoing Experience Design process. Observations of gameplay and participant reflections directly informed areas for refinement. For example, several participants relied heavily on visual organization strategies, suggesting the need for editable visual markers or highlights within the code editor. We also identified the need for clearer console-based error feedback to support more flexible code manipulation.

Future versions of the system will introduce optional hint mechanisms that provide contextual musical information during puzzle solving, particularly in levels with unknown tiles. These additions aim to support curiosity while maintaining the problem-solving structure of the game.

Beyond interface refinements, future research will more systematically investigate how *BeatMohan's Symphony* supports the development of algorithmic live-coding proficiency in Strudel, including the introduction of melodic structures alongside rhythmic patterns. Controlled studies will be required to examine how engagement within the game-based context compares to other sequencer or live-coding environments in supporting musical understanding. We are also curious to understand if the code-based music interaction produces a richer conceptual understanding than direct manipulation. Additionally, the game's punishment mechanic raises questions worth examining in future iterations: whether platformer conventions for musical error feedback shape how learners interpret precision and musicality, and whether alternative models might better reflect the contextual and interpretive nature of musical decisions.

Taken together, this work positions *BeatMohan's Symphony* as an evolving educational digital musical interface. In alignment with Bamberger's Experience Design framework, designed activities, learner interaction, and qualitative evaluation operate as mutually informing components within an ongoing iterative refinement process guided by our learning goals.

9 Ethical Standards

This study was reviewed and approved by the Institutional Review Board (IRB) at Georgia Institute Of Technology prior to data collection (Protocol IRB2025-1298). All procedures complied with institutional guidelines for research involving human participants.

Gameplay sessions, screen recordings, and post-activity interviews were collected solely for research purposes. All data were de-identified prior to analysis, and no personally identifying information is reported in this paper.

Acknowledgments

I would like to thank Dr. Betsy DiSalvo, Associate Professor in the School of Interactive Computing for her class Educational Game Design at Georgia Tech, from which the concept for this project originated.

References

[1] Jeanne Bamberger. 1973. Learning to Think Musically. *Music Educators Journal* 59, 7 (March 1973), 53–57. https://doi.org/10.2307/3394328

[2] Jeanne Bamberger. 2015. A Brief History of Music, Computers and Thinking: 1972–2015. *Digital Experiences in Mathematics Education* 1, 1 (April 2015), 87–100. https://doi.org/10.1007/s40751-015-0003-3

[3] Alan F Blackwell, Emma Cocker, Geoff Cox, Alex McLean, and Thor Magnusson. 2022. *Live coding: a user's manual.* MIT Press.

[4] Andrew Brown. 2012. Experience design and interactive software in music education research. *Visions of Research in Music Education* 20, 1 (2012), 1–38.

[5] Sarah Davis, Jack Armitage, and Gus Lobban. 2024. Pop Live Coding Encounters: Reflections on Practice. In *Proceedings of the International Conference on Live Coding (ICLC).*

[6] Jason Freeman and Brian Magerko. 2016. Iterative composition, coding and pedagogy: A case study in live coding with EarSketch. *Journal of Music, Technology & Education* 9, 1 (May 2016), 57–74. https://doi.org/10.1386/jmte.9.1.57_1

[7] Jason Freeman, Brian Magerko, Doug Edwards, Tom Mcklin, Taneisha Lee, and Roxanne Moore. 2019. EarSketch: engaging broad populations in computing through music. *Commun. ACM* 62, 9 (2019), 78–85.

[8] D. Griffiths. 2008. Al-Jazari. https://runme.org/project/+aljazari/index.html. Accessed April 28, 2025.

[9] Paul Hayes. 2023. Replicube. Video game. https://store.steampowered.com/app/2153030/Replicube/

[10] Henkjan Honing. 2013. Structure and interpretation of rhythm in music. *The psychology of music* 3 (2013), 369–404.

[11] Samuli Laato, Sampsa Rauti, Alexander Espeseth, Heinrich Söbke, Juho Hamari, et al. 2023. Composing Music Through Tile-based Games. In *2023 IEEE International Symposium on Multimedia (ISM).* IEEE, 309–314.

[12] Andrew J Lesser. 2024. *Gamifying the Music Classroom: Digital Tools for Practical Application.* Oxford University Press.

[13] Thor Magnusson. 2010. An epistemic dimension space for musical devices. In *Proceedings of the 2010 conference on new interfaces for musical expression (NIME 2010).* University of Technology, Sydney, 43–46.

[14] Bill Manaris, Blake Stevens, and Andrew R Brown. 2016. Jythonmusic: An environment for teaching algorithmic music composition, dynamic coding and musical performativity. *Journal of Music, Technology & Education* 9, 1 (2016), 33–56.

[15] Alex McLean, Dave Griffiths, Nick Collins, and Geraint Wiggins. 2010. Visualisation of live code. https://doi.org/10.14236/ewic/EVA2010.6

[16] Alex McLean and Geraint Wiggins. 2010. Tidal–pattern language for the live coding of music. In *Proceedings of the 7th sound and music computing conference.* 331–334.

[17] Daniel Müllensiefen, Bruno Gingras, Jason Musil, and Lauren Stewart. 2014. The Musicality of Non-Musicians: An Index for Assessing Musical Sophistication in the General Population. *PLOS ONE* 9, 2 (2014), e89642. https://doi.org/10.1371/journal.pone.0089642

[18] Click Nilson. 2007. Live Coding Practice. In *Proceedings of the International Conference on New Interfaces for Musical Expression.* New York City, NY, United States, 112–117. https://doi.org/10.5281/zenodo.1177209

[19] Nintendo. 1981. Donkey Kong. Arcade game. Original arcade release.

[20] Nintendo. 1985. Super Mario Bros. Nintendo Entertainment System game. Original NES release.

[21] Nintendo. 1992. Mario Paint (Composer Mode). Super Nintendo Entertainment System game.

[22] Michael Orey. 2010. *Emerging perspectives on learning, teaching and technology.* CreateSpace North Charleston.

[23] Hempuli Oy. 2019. Baba Is You. Video game. PC and console release.

[24] Seymour A Papert. 2020. *Mindstorms: Children, computers, and powerful ideas.* Basic books.

[25] JR Parker and John Heerema. 2007. Musical interaction in computer games. In *Proceedings of the 2007 conference on Future Play.* 217–220.

[26] Christopher Petrie. 2022. Programming music with Sonic Pi promotes positive attitudes for beginners. *Computers & Education* 179 (2022), 104409.

[27] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (2009), 60–67.

[28] Charlie Roberts, Jesse Allison, Daniel Holmes, Benjamin Taylor, Matthew Wright, and JoAnn Kuchera-Morin. 2016. Educational design of live coding environments for the browser. *Journal of Music, Technology & Education* 9, 1 (2016), 95–116.

[29] Gerard Roma. 2023. Agent-Based Music Live Coding: Sonic adventures in 2D. *Organised Sound* 28, 2 (Aug. 2023), 231–240. https://doi.org/10.1017/S1355771823000274

[30] Felix Roos and Alex McLean. 2022. Strudel: Live Coding Patterns on the Web. https://strudel.cc/ Web-based port of the TidalCycles pattern language to JavaScript.

[31] Gareth Terry, Nikki Hayfield, Victoria Clarke, Virginia Braun, et al. 2017. Thematic analysis. *The SAGE handbook of qualitative research in psychology* 2, 17-37 (2017), 25.

[32] Paul Wenzel. 2024. *Pocket Operations: A Collection of Drum Machine Patterns* (second edition, revision 3.1 ed.). Shitty Recording Studio, Minneapolis, MN, USA. https://shittyrecording.studio/ PDF available from the author's website; print editions sold via Lulu and Amazon.

Table 1: Post-Gameplay Reflection Interview Questions

Reflection Interview Questions
1. What did you feel like you learned about building rhythmic patterns from this experience?
2. Were there moments where you paused to listen before deciding what to code next? What were you listening for?
3. What was your favorite pattern to code and why?
4. How did the feedback from the puzzles influence your actions in the free-play sections?
5. Were there specific patterns or musical rules you tried to experiment with in the free-play section?
6. What criteria or judgment did you use to decide whether a pattern you created was "good" or "interesting"?
7. What did you think about writing rhythmic patterns in code? Would you interact with live coding for music in the future without the visual cues? Why or why not?
8. If you could continue using *BeatMohan's Symphony*, what would you want to learn, create, or explore next? Why?