When Literacies Collide

The Role of Translation in Music+Coding Activities

Cameron L. Roberts
Northwestern University, Computer Science and Learning Sciences
cameronroberts2020@u.northwestern.edu

Michael S. Horn
Northwestern University, Computer Science and Learning Sciences
michael-horn@northwestern.edu

ABSTRACT

The integration of computer programming and music-making has a rich history dating back to the 1950s. While there has been substantial prior work on the creative and cognitive affordances of programming languages for engaging in musical tasks, there is less work that attempts to understand the theoretical implications of music and code as literacies in collision. In this paper, we report on a study in which five undergraduate students with experience in both music and coding completed two creative musical tasks: one using conventional instruments and tools and one using Python code in an online music-coding environment. In combining representational infrastructures from music and code, both undergo transformations. We introduce semiotic theories of translation and transcription to make sense of the music-coding process and describe strategies that participants devised in their creative process.

CCS CONCEPTS

• **Human-centered computing**; • **Human computer interaction (HCI)**; • **Collaborative and social computing**;

KEYWORDS

Computer Science Education, Music, Computational Literacy

ACM Reference Format:
Cameron L. Roberts and Michael S. Horn. 2023. When Literacies Collide: The Role of Translation in Music+Coding Activities. In *Learning, Design and Technology (LDT '23), June 23, 2023, Evanston, IL, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3594781.3594795

1 INTRODUCTION

Since the 1950s, researchers and artists have sought to harness the creative potential that exists at the intersection of music and computer programming languages [20, 34]. Advances in computational technology and programming language design have helped support thriving communities interested in the affordances of computer languages for expressing musical ideas and supporting creative expression. Examples include explorations of live coding [1], functional programming paradigms for representing musical structure [1], tools for modular synthesis [34], and data-flow languages [28]. Other research has focused on educational questions: Can the intersection of music and code help students learn about music-making? Or computer programming? Can this intersection change students' relationships with both domains in terms of identity and self-expression?

While there is a growing body of research exploring these questions, the field is still in need of theoretical foundations that help conceptualize music and coding as bidirectional and mutually influencing. In this paper, we report on a study in which five undergraduate students with experience in both music and coding completed two creative musical tasks—one using traditional musical tools, instruments, and representations and one using the Python programming language and an online learning platform called TunePad [17]. Using a constructivist grounded theory approach [9], we observe the critical role of *translation* [11] between representational systems in the music-coding process and the potential for computation to serve as a restructuration of musical knowledge.

2 RELATED WORK

Of the many technical domains in which coding interacts, music is a fascinating case. Many of the domains that researchers have investigated rely on formal representational systems. Examples include physics [30], chemistry [21], and the life sciences [37]. Due to the technical nature of these domains, researchers are able to ask if code can supplement, augment, or even replace conventional representational systems, and, in the process, restructure learners' cognitive engagement with the concepts themselves [10, 30, 36]. While music involves formal and technical representational systems, it is also a sociocultural phenomenon that cannot be separated from existing encultured knowledge. Music-making, or *musicking* [31], is a natural human activity. We are surrounded by music from birth in countless different forms. Through this process of enculturation, individuals acquire "basic musical competencies through everyday exposure to music" [15]. This manifests itself as an increased affinity and intuition for music from one's own culture through culture-specific cognitive representations. This contrasts with formal musical training, which gives individuals explicit power over this musical intuition by introducing new categories, abstractions, and processes. Research has also shown that individuals with musical training have improved cognitive abilities related to sound—such as pitch discrimination and tonal pattern recognition—which correlates with levels of expertise [7]. In other words, formal music instruction directly changes how we process and encode sound information. Therefore, in introducing code as a form of representation of music, this new computational literacy must come into

LDT '23, June 23, 2023, Evanston, IL, USA

ACM ISBN 979-8-4007-0736-0/23/06…$15.00
https://doi.org/10.1145/3594781.3594795

contact with both encultured and formal aspects of an individual's musical literacy.

Prior work at the intersection of music and coding has either tended to emphasize the technical aspects of computer language design; or it has thought about music as a motivational context within which to learn computer science concepts and potentially broaden participation in computing fields. What's missing is work that considers the cognitive implications of music and computation colliding as literacies. How do these two rich domains act on one another, and what implications does this have for designing learning environments that utilize both? In our view, when literacies collide, transformative spaces can be opened in which both domains are reimagined. Combining coding and arts education—as music-coding systems such as TunePad, EarSketch, and SonicPi seek to do—can alter a learner's experience with both domains. On the one hand, we are given new tools for understanding the algorithmic, mathematical, and structural aspects of music. But what is less often considered is that music might also change how we think about code.

The use of music and code to enhance learning has a long history. Work from Jeanne Bamberger laid the foundation with projects such as MusicLOGO and Impromptu [2, 6]. Many general-purpose learning environments such as Scratch [29] have music creation capabilities, even if they are not optimized for musical expression [26]. Tools like Web Audio, SuperCollider, and Max/MSP are designed for professionals or advanced learners and may not be suitable for beginners. On the other hand, more recent domain-specific projects like Sonic Pi [1, 27], Jython Music [24], EarSketch [12], and TunePad [18] have gained worldwide popularity among hundreds of thousands of students. These platforms make use of text-based programming languages such as Python or JavaScript and are tailored specifically towards music creation.

2.1 Literacies

There have been many attempts to define musical literacy, both in connection with traditional language literacy [14, 16] and domain-specific conceptions [8, 35]. We conceive of musical literacy as the skills and competencies needed to negotiate and interact with musical texts in a culturally defined, substantive manner. These interactions may include conventional musical activities such as creating (composing or improvising), interpreting (listening or analyzing), and performing. They may also include interacting with music in participatory ways such as remixing, covering, satirizing, or sampling [32]. A musical literacy is necessarily specific to a musical culture rather than universal, although there may be overlap of competencies. We consider the ability to think abstractly in sound, or *audiation*, as the core competency of our concept of musical literacy rather than privileging symbolic forms. Audiation is the process of hearing and comprehending sound in one's mind when the sound is not present. This is separate from the concept of aural perception, which occurs when sound *is* present [13]. Audiation may be engaged for a range of musical activities: imagining the resulting sound while reading sheet music, improvising, or writing music. Those who are skilled readers of musical notation also show a link between the notation and sound. That is, they can audiate or subvocalize the imagined sound of a piece without any intermediary representation. In contrast, more novice readers may link notation not with a sound result, but with a kinesthetic means of sound production—that is, for example, they may associate a note on the staff with a key on a piano or a specific fingering on the flute [25]. Other supporting competencies include (but are not limited to) working with theoretical and notational systems as well as instrumental proficiency and collaboration.

Our view of computational literacy derives from the work of diSessa [10, 22] who thinks of coding (and computing more broadly) as a sort of technical literacy with cognitive, social, and cultural dimensions. On a cognitive level, calling computation a *literacy* implies fluency with technical representation systems; however, that fluency also reflects back on us, shaping how we think about the world, how we solve problems, and how we engage in creative tasks. As a person becomes more computationally literate, they may be able to think more abstractly in terms of tools, design patterns, or approaches to solving a problem without worrying about language-specific syntax or formalisms. On a social level there is an aspect of communication and sharing of ideas that is facilitated through technical inscriptions. The shared understanding of technical inscriptions supports communities of people engaged in a variety of endeavors (e.g. a team of software engineers or an online community such as Stack Overflow). Finally, on a cultural level, diSessa argues that literacies should have society-wide implications that transcend any one domain or discipline. For this last point, diSessa might expect different genres of computational literacy to emerge when applied to specialized tasks such as music creation.

Our notion of musical literacy overlaps with computation in several ways. For instance, both require working with abstractions. In music, this may involve recognizing chord progressions or melodic motifs, while in computation, it may involve recognizing common programming structures such as loops or functions. Both may involve a specialized symbolic language as well as iterative problem-solving routines (i.e. practicing and debugging).

2.2 Translation

To mediate between the two representational systems associated with these forms of literacy, we chose the analogy of *translation*, building on the work of Umberto Eco [11]. This choice is due in part to the ambiguities that are inherent in making a translation. To use language as an example, this is to say that meaning is fundamentally altered in translating from one language to another, both due to cultural differences and the abstractions that are available in each language. Through the process of translation, meaning—which may or may not accurately reflect the meaning of the original text—is transmuted and reconstructed in the new language.

Eco's model of translation builds on the work of Jakobson [19] and expands his model of translation to encompass translation both between as well as within languages and other sign systems. Intrasemiotic translation (or intralinguistic in the case of language) is a reformulation within a single sign system. In language, this may manifest as rewording of a thought. Intersemiotic translation (interlinguistic in language) encompasses translation between sign systems. This may manifest as a change of media, such as adapting a novel to the stage. Both forms of translation can be contrasted with *transcription*, which Eco describes as being able to be carried

out by "automatic substitution", such as moving from a natural language to morse code [11].

In the context of this paper, we are taking the set of musical and computational signs to be our two principal systems of interest—that is, the set of signs and relations that convey musical and computational meanings. Because we are considering the set of signs, an intrasemiotic translation in this context could be considered moving from one computational representation to another within a single system—for instance, refactoring code or moving from one coding language to another. We consider intersemiotic translation to be the transfer of knowledge from one system to another. Again, using computation as an example, this could manifest as the implementation of a computer program (i.e. moving from a list of requirements to code) [23].

3 METHODS

The next subsections provide an overview of our participants, data collection, and analysis methods.

3.1 Participants

All five of our participants were advanced undergraduate students participating in a two-week training program to prepare coaches for teaching summer programs with TunePad. Each coach had some amount of experience with both coding and music. Each had at least one STEM major and some level of formal Computer Science background. Four participants were involved in informal musical activities (e.g. a cappella or songwriting), and one participant was studying viola performance as a second major. Two of the participants had been TunePad coaches the previous summer and had prior experience with TunePad.

3.2 Data Collection

The five coaches participated in two sessions, each around 45 minutes long and on separate days. For both sessions, participants were given approximately 20 minutes to compose a short original melody and then reflected on their compositional process through a semi-structured interview. For the first session, participants were instructed to bring any non-coding tools of their choice to aid their composition. For the second, participants were instructed to code their melody in TunePad. Participants were not informed of the specific musical task they were to complete ahead of each session to avoid any potential preparation. While composing, participants were instructed to think aloud about their musical process as well as to vocalize musical ideas by humming or singing. Every participant provided signed informed consent as well as verbal consent to be video recorded.

In the reflection portion of the interview, participants were asked to describe their musical process in greater detail. They were asked questions such as: *Why did you choose your specific musical tool? How did you structure your musical composition? What strategies did you use to write your musical ideas as code? How do your coding skills transfer into music-coding? How do your music-coding strategies differ from your regular coding strategies?*

Interviews were audio and video recorded and then transcribed for analysis.

3.3 Research Questions

The purpose of this study was to investigate how computation affected the musical creative process and how music affected the coding process. The following questions guided the analysis:

1. How did participants engage in both the musical and computational aspects of the task?
2. How did processes and artifacts between the trials differ?
3. What are the limitations and affordances of code as a form of musical representation?
4. How did the nature of the musical task affect the code that participants created?

3.4 Analysis and Positionality

To make sense of our data we applied a constructivist grounded theory approach [9], looking at both interviews as well as the musical artifacts created. Transcripts were inductively coded sentence-by-sentence. After initial coding, transcript data was edited for clarity and conciseness, removing filler words and redundancy. Then initial codes were grouped into larger emergent themes which were refined and abstracted through constant comparison. These categories included representation (e.g. notational or instrumental), musical literacy (both formal and informal), computational literacy, compositional process, and knowledge transfer. Individual data were segmented and aggregated into the major thematic areas and once again refined. A key component of this approach is acknowledging the researcher's own subjectivity in building a theory. CR is a software developer who has contributed to the TunePad as a curriculum developer, software developer, and coach. MH is a professor of computer science and learning science and created the TunePad platform. MH was conducting the coach training workshop. Both MH and CR have training in Western musical traditions.

3.5 TunePad

In this study we used TunePad, a free, online music+code environment that was developed as a complement to EarSketch [17]. TunePad projects take the form of specialized computational notebooks called playbooks that consist of an arbitrary number of cells containing Python code, multimedia content, and text. Code in TunePad can represent individual musical elements such as notes, rests, chords, or percussion sounds. This research could also have been conducted using similar music+code environments that make use of text-based programming languages.

4 FINDINGS

For the first task, the participants created a variety of different musical compositions. Jan (pseudonym), a singer-songwriter and musician, created a country-western inspired verse using GarageBand, MIDI keyboard, and her voice. Ian, a classically trained violist, composed a simple classical melody using his viola. Lewis, an *a capella* singer and arranger, created an indie-pop inspired song with two contrasting sections using a MIDI keyboard and voice. Sal, an amateur guitarist and saxophonist, wrote a full verse-chorus song using guitar and voice. Morgan, who participated in high school band and college *a capella* has no formal training in music theory. Morgan used online reference material as well as a MIDI

keyboard, notation software, and whistling to compose a short melodic fragment.

For the second task, participants used TunePad and Python code to compose an entirely new composition. Jan started by creating drum and bass accompaniment and then composed an eight measure pop piano melody over that accompaniment. Ian wrote three separate motifs inspired by EDM but struggled to line up the different cells rhythmically. Like Ian, Lewis created short motifs rather than developing a longer melody; he wrote four different motifs which were two measures in duration. Sal wrote a two measure piano motif outlining two chords and a drum beat and then composed a four measure piano melody over this texture; like Ian, Sal also struggled to rhythmically align the different cells. Morgan was the fastest to complete his melody; he first developed a short eight measure chord progression and then composed a piano melody on top of this.

4.1 Translation and the Creative Process

In our analysis, we began to see the role that *translation*—both conscious and automatic—played in negotiating representations of both music and code. In both the first and second task, we found that participants had to negotiate a variety of constraints, representations, and sensory-perceptual resources including:

- **Sound**: the aural feedback a participant receives
- **Theoretical knowledge**: both the intuitive and formalized means participants have of structuring musical knowledge
- **Cultural knowledge**: the aesthetics and values encultured both by specific genres/subcultures and of larger Western culture
- **Instrumental**: the link between the sensorimotor network and perception
- **Notational**: written systems for representing music, such as standard music notation and tablature

Even in the non-coding task, participants had to *translate* between different musical texts and representations. This is to say, they were "reading" or interpreting musical texts and negotiating musical meaning across related representational forms. We see this negotiation as an *intrasemiotic translation*, or a reformulation within a single sign system—in this case, music.

This translation manifests in many different ways depending on the representations in play. For instance, for some participants the physical instruments (e.g. guitar or viola) afford for automaticity in which there is no conscious translation between thought and action. This automaticity is of course predicated on some degree of virtuosity, otherwise there must be an additional translational step. In the reflection after her second task, Jan remarked how this extra step could serve as a barrier to creative expression:

> *"I feel like I have so much control over my voice. When I was playing the saxophone, if I was trying to do a saxophone solo, I would have to think what I want to hear and how [to] translate that to my fingers and actually play it. If I really practiced my scales and got to that point, like if you're like an insane saxophone player, you kind of just think it and play. You don't do it in two steps; you just do that one step. That's how I feel a little bit with guitar and piano."*

The music-coding process in the second task involved moving from a variety of musical forms—notational, instrumental, cultural, and others—to the language of computation through what we saw as a recursive process of refining towards higher fidelity representations. As in the first task, translation was also present in how participants negotiated non-coding constraints and representations; in this instance, however, the participants demonstrated an *intersemiotic* form of translation: that is, movement from one sign system to another. We observed that in their compositional processes, none of the participants formulated musical ideas in code, but instead converged on their musical ideas through a combination of audiating and vocalizing. They also leveraged musical representations in the TunePad platform—including standard music notation, piano roll, and track instruments (keyboard, guitar, and drums). Participants then engaged in an active process in which semi-formed musical ideas were transformed from cognitive auditory structures (which may also leverage the appropriation of other forms of musical representations) into computer code. In other words, they enacted an informal process in which they transformed their musical idea into code.

As part of their creative process, the output of the code itself (i.e. the sound produced by TunePad) was evaluated and compared to the original musical idea of the participant. If necessary, the participant engaged in debugging and refining this code. This step was predicated on the participant being able to discern an incongruity or dissonance between their initial idea and the audio output of the code (related to musical literacy) as well as their ability to understand and debug the code they wrote (related to computational literacy). The translational process continued iteratively until the participant was able to find coherence between their mental representation of the sound and the musical artifact.

Computational and musical literacies work in tandem in a music-coding setting, but the ability to make the translational step into computer code serves as a major pain point in music-coding environments. Similar to the example of instrumental proficiency serving as a barrier to creative expression, coding fluidity has obvious implications for the success of a music-coding task, as one participant remarked:

> *"[There's] a clear difference between the interns who had CS experience and [those] who didn't. The ones who had experience could make a lot cooler stuff, but I think that's just because we could do it faster and didn't have to be like, 'I want this thing to happen; how do I do it?' For me, it's 'I want something to happen. I kind of know how to do it because I know how coding works.'"*

In the example above, the participant reflected on how one's degree of computational knowledge can act as a constraint on musical expression; this can certainly work in reverse as well. Morgan, who was one of the most adept coders, lamented that his lack of music theory knowledge was his primary challenge in both tasks. His musical strategy involved a large degree of trial-and-error (divergent thinking), testing out many ideas without clear direction. This was time-consuming and caused frustration:

> *"Because all the aural skills that I had are gone, when I hear what would sound right in my head, there's no translation from that sound into a chord, or a knowledge*

Code without variables	Equivalent code with variables for notes	With variables for notes and duration
`playNote(48, beats=1)`	`from constants import *`	`from constants import *`
`playNote(48, beats=1)`	`playNote(C3, beats=1)`	`QUARTER = 1.0`
`playNote(55, beats=1)`	`playNote(C3, beats=1)`	`HALF = 2.0`
`playNote(55, beats=1)`	`playNote(G3, beats=1)`	`playNote(C3, beats=QUARTER)`
`playNote(57, beats=1)`	`playNote(G3, beats=1)`	`playNote(C3, beats=QUARTER)`
`playNote(57, beats=1)`	`playNote(A3, beats=1)`	`playNote(G3, beats=QUARTER)`
`playNote(55, beats=2)`	`playNote(A3, beats=1)`	`playNote(G3, beats=QUARTER)`
	`playNote(G3, beats=2)`	`playNote(A3, beats=QUARTER)`
		`playNote(A3, beats=QUARTER)`
		`playNote(G3, beats=HALF)`

Figure 1: Use of variables as a translational aid.

> *of what comes next. So, there are things where I'm like, 'I wish that sounded like something else.' I would've either spent more time playing around with stuff, but in the time I had, playing around didn't work."*

The process of translating to code also at times diverged from the intended outcome, generating new creative possibilities. The act of searching for coherence (and sometimes failing) often revealed new ideas:

> *"Very [little] of what I wrote down is what I was hearing [in my head]. It's more like, 'oh, this sounds good anyway.' I didn't mean to mis-translate that, but I think in the end, I was able to make more efficient decisions where I was making more intelligent guesses."*

4.2 Translational Strategies

Beyond coding or musical competencies, the ability to make translational steps is dependent on building conceptual links across representation systems and domains. Some participants took advantage of the abstractions and representational affordances of code to help with this process. In our programming with K-12 students, understanding the mathematical and fractional nature of music rhythms has often been a pain point. For example, a quarter note in musical notation translates into 1 beat in TunePad Python code. An eighth note translates into 0.5 beats, and so on. Even the concept that beats are conventionally subdivided by powers of 2 (½, ¼, $\frac{1}{8}$) is a difficult concept that is far from immediately obvious to beginners. Even an experienced music-coding participant struggled to make the link between music notation and beats without use of a translational aid. Speaking about this particular pain point, Ivan says:

> *"You have to really focus on the math side of music in terms of the length of the notes and the articulation you want. [. . .] In terms of just straight up note length, that's a lot easier to do as a musician. [. . .] Coming from a music background, I have the intuition that a quarter of a beat is a 16th note, and a half a beat is an eighth note. But all that kind of gets thrown out the window."*

One such strategy that other participants used for this particular pain point was through use of the built-in *constants* library—which defines variables for notes and rhythmic values—or by defining their own variables (see figure 1). One participant remarked that code written in this way "looked like music" even though he understood that it was functionally the same. Another, that using variables in this way allowed them to leverage their knowledge of sheet music to better visualize the end product. This is not to mention the added bonus in terms of readability and explainability.

The organizational affordances that participants took advantage of were not limited to variables. Comments allowed them to leave notes and reminders for things such as the notes which are a part of the harmony. Participants also added line breaks and whitespace as a kind of equivalent to musical barlines by adding them in between measures. Morgan said,

> *"I like being able to separate what each measure looks like and write little notes to myself. It's nice and helpful. I kind of wanted this [in task #1] but writing out the notes of the chords would have been a nice tool."*

The parameterized nature of coding allowed participants to decompose the problem of translation into two separate problems: translating pitches and translating rhythmic values. One participant described how they filled in placeholder values for rhythmic values to denote relative length before they figured out the exact value:

> *"Generally I get the notes first and then deal with the rhythms after. . . Sometimes if I know that one note is shorter than the next one, I will just put like 0.5 instead. It's like I'm doing little annotations and then going back and editing."*

4.3 Transforming Coding Practices

4.3.1 Debugging and Multiple Hearings. We observed that music-coding as a process altered typical computational practices. In the process of translation, participants used multiple forms of debugging. On one level, they practiced conventional debugging: both syntactic and semantic [33]. For this, they were able to use the Python interpreter and their computational knowledge. In the evaluation of the correctness of their code (i.e. the output matching their musical ideas), they had to employ an entirely different form of debugging through musical aural skills. They had to find errors in the program through sound, first identifying the incongruity in sound and then mapping that to code. One participant remarked on the difference between this sound debugging and conventional debugging:

> *"The way that I'm testing is by listening; it's more of an interactive way. How I would normally debug is [through] test code or maybe a print statement or [with] an output. It's kind of like this is the output except instead of visually seeing an output, I'm hearing it and processing myself. [Instead of] the computer telling me what's wrong, I'm telling me what's wrong."*

In the debugging process, listeners must manage and remedy multiple incongruent hearings of a musical idea in order to find coherence as well as contend with the ephemeral nature of musical memory. Bamberger characterizes musical development in terms of this sort of disequilibrium among representations [4]:

> "Musical development is enhanced by continuously evolving interactions among multiple organizing constraints along with the disequilibrium and sensitivity to growing complexity that these entanglements entrain [...] Rather than being a uni-directional process, musical development is a spiraling, endlessly recursive process in which organizing constraints such as those above are concurrently present creating an essential, generative tension as they play a transformational dance with one another." (page 4)

This process has the capacity to engage both musical and computational literacies and also to reinforce both. The dissonance between the mental idea and its computational form can highlight the ambiguities in a listener's mental representation and results in a spiraling process in which both are refined.

4.3.2 Refactoring as Restructuration. Key to our concept of translation is the fact that coding is itself a creative process in which there are multiple "correct" solutions; that is, there are multiple ways to formulate code which have the same musical output. Unlike canonical musical notations, the difference between these code representations is not syntactical but structural—although we have also described strategies employed on the level of syntax (i.e. comments, variables). There are heuristics for evaluating and choosing between different viable solutions—such as readability, modularity, and conciseness—and we saw participants actively employing these heuristics in the refinement of their code. There are certainly translations which obscure the musical structure in the exact same way as musical notation (see figure 2 below). Sequential representations—essentially a list of notes and rhythmic values—are arguably indistinguishable in form from standard musical notation (but are nonetheless an important stage towards thinking programmatically about music). These are arguably more akin to a *transcription* rather than a translation [11].

Although the time constraints of the task discouraged some participants from refining their code, more advanced coders were able to encapsulate elements of musical structure using coding abstractions such as loops and functions. Morgan describes this as a matter of convenience:

> *"...as soon as I find a pattern—which is very, very common in popular music—my code starts to look much more modular because it's easy to break things up and make my life easier."*

Another participant echoed this preference for convenience and conveyed that they were evaluating their code based on metrics of readability and conciseness:

> *"I defined two little functions—my A and B functions—so I don't have to write these over and over again. Even though I'll only use the B function once—I wouldn't need to create a function because I'm not using it again—it's nice to have it more organized. I think it's pretty important when you're writing music in TunePad being able to use variables, comments, functions because it really does clean it up and make it tidier and easier."*

As this shows, the translational process does not end when participants reach a "correct" answer—and in fact, as we have seen, it may stop before then—it ends when they reach coherence between the musical idea and its representation. We view computer code as a restructuration of musical knowledge [36]. As Bamberger and diSessa [5] proposed with music and mathematics, this encapsulation has the capacity to reveal the underlying formal and mathematical structure of music—what Bamberger might call "units of perception" [3]—in a way that other forms of notation obscure. This is to say that the translational process serves as a form of musical analysis. We argue that the goal of music-coding education must be the continuous refinement of the translations, which in turn spurs the development of both musical and computational literacies. Musical literacy, through the illumination of these larger structural components; computational literacy, through the utilization of principles of encapsulation, abstraction, and decomposition.

5 DISCUSSION AND FUTURE WORK

In this paper, we have begun to outline the role of translation as a key process in music-coding. In doing so, we propose that music-coding introduces shifts in both forms of literacy and begins to emerge as something new. We observed changes, spurred by the specific considerations of the new domain, which operated on the levels of procedure, structure, and aesthetics. On the level of aesthetics, participants sought to make their code "look more musical" through imposing additional organizational constraints. On the level of procedure, participants leveraged language and platform affordances to aid in the translational process. On the level of structure, participants encapsulated elements of musical structure into loops and functions. Our findings suggest that prior music and

TunePad code for *Hot Crossed Buns*

```
playNote(52, beats=1)
playNote(50, beats=1)
playNote(48, beats=2)
playNote(52, beats=1)
playNote(50, beats=1)
playNote(48, beats=2)
playNote(48, beats=0.5)
playNote(48, beats=0.5)
playNote(48, beats=0.5)
playNote(48, beats=0.5)
playNote(50, beats=0.5)
playNote(50, beats=0.5)
playNote(50, beats=0.5)
playNote(50, beats=0.5)
playNote(52, beats=1)
playNote(50, beats=1)
playNote(48, beats=2)
```

Refactored code to encapsulate repeating sections

```
def phrase_a:
    playNote(52, beats=1)
    playNote(50, beats=1)
    playNote(48, beats=2)

def phrase_b:
    playNote(48, beats=0.5)
    playNote(48, beats=0.5)
    playNote(48, beats=0.5)
    playNote(48, beats=0.5)
    playNote(50, beats=0.5)
    playNote(50, beats=0.5)
    playNote(50, beats=0.5)
    playNote(50, beats=0.5)

phrase_a()
phrase_a()
phrase_b()
phrase_a()
```

Figure 2: Use of encapsulation to illuminate musical structure.

coding training did not directly transfer to the music-coding context, even in participants experienced with both music and coding. Rather, musical experiences—both formal and informal—develop a toolset of musical skills which may (or may not) aid in adapting to new representational forms. As is implied by our definition of musical literacy, musical skills are not necessarily transferable to different contexts, meaning that there is no generalized or universal form of music literacy that applies to all musical genres and cultures. Musical knowledge and familiarity with a particular musical representation are culturally specific and are often acquired through enculturation, which means that skills and knowledge that are specific to one context may not be directly transferable to another context.

In music-coding, this cultural specificity means that it is not enough simply to drop music into coding. This involves recognizing that music-coding is not an amalgamation of the two content areas, but rather a unique domain that requires specialized skills. In our music-coding task, musical thinking formed the foundation of creative expression, but this was highly interconnected with the coding process—with translation serving as mediation. The back-and-forth between musical and computational knowledge allowed for new divergent creative possibilities, as we observed with Morgan. The refinement of both musical and computational representations that was achieved through the translational process suggests promising pedagogical implications to both domains. In a future paper, we will take a closer look at how these three processes intertwine and present a more complete model of the music-coding process.

The implications of our findings suggest that curricula should focus on actively cultivating this translational linkage between musical knowledge and coding. Educators should aim to equip students with the necessary tools to navigate the complexities of musical representation in the coding context. This involves developing a deep understanding of musical concepts such as rhythm, melody, harmony, and timbre through coding and non-coding musical activities, as well as the ability to translate these concepts into code.

6 LIMITATIONS

A limitation of this study was sample size and participant demographics. In future studies, we would like to expand our dataset to include participants from more diverse musical backgrounds and a wider set of ages and abilities as well as to compare multiple music-coding platforms. In future papers, we plan to describe the implications of these findings on the design of our interfaces and curricula. We also note that the task used in this research design was artificially constrained and may not resemble how people use similar platforms to make music in real life. As such we lack thick descriptions that could be gained from an ethnographic immersion within creative communities.

ACKNOWLEDGMENTS

This research was supported by grants DRL-1612619, DRL-1451762, DRL-1837661, and DRL-2119701 from the National Science Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. The second author is co-owner of a commercial entity established to generate revenue to support the ability to continue to offer TunePad as a free platform available for anyone to use.

REFERENCES

[1] Samuel Aaron and Alan F. Blackwell. 2013. From sonic Pi to overtone: creative musical experiences with domain-specific and functional languages. In *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design*, ACM, Boston Massachusetts USA, 35–46. DOI:https://doi.org/10.1145/2505341.2505346

[2] Jeanne Bamberger. 1979. Logo Music projects: Experiments in musical perception and design. *LOGO Memo* 52 (1979). Retrieved from http://hdl.handle.net/1721.1/5726

[3] Jeanne Bamberger. 1996. Turning music theory on its ear Do we hear what we see; Do we see what we say? *Int J Comput Math Learning* 1, 1 (1996). DOI:https://doi.org/10.1007/BF00191471

[4] Jeanne Bamberger. 2006. What develops in musical development? A view of development as learning. *The Child* (2006), 69–92.

[5] Jeanne Bamberger and Andrea diSessa. 2003. Music as Embodied Mathematics: A Study of a Mutually Informing Affinity. *International Journal of Computers for Mathematical Learning* 8, 2 (2003), 123–160. DOI:https://doi.org/10.1023/B:IJCO.0000003872.84260.96

[6] Jeanne Bamberger and Armando Hernandez. 2000. *Developing musical intuitions: a project-based introduction to making and understanding music*. Oxford University Press, New York.

[7] Karen Chan Barrett, Richard Ashley, Dana L. Strait, and Nina Kraus. 2013. Art and science: how musical training shapes the brain. *Front. Psychol.* 4, (2013). DOI:https://doi.org/10.3389/fpsyg.2013.00713

[8] Paul Broomhead. 2021. A New Definition of Music Literacy: What, Why, and How? *Music Educators Journal* 107, 3 (March 2021), 15–21. DOI:https://doi.org/10.1177/0027432121991644

[9] Kathy Charmaz. 2006. *Constructing grounded theory*. Sage Publications, London; Thousand Oaks, Calif.

[10] Andrea A. diSessa. 2018. Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education. *Mathematical Thinking and Learning* 20, 1 (January 2018), 3–31. DOI:https://doi.org/10.1080/10986065.2018.1403544

[11] U. Eco. 2008. *Experiences in Translation*. University of Toronto Press. Retrieved from https://books.google.com/books?id$=$0dVYaP9VukIC

[12] 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 (August 2019), 78–85. DOI:https://doi.org/10.1145/3333613

[13] E. Gordon. 2012. *Learning Sequences in Music: A Contemporary Music Learning Theory*. GIA Publications. Retrieved from https://books.google.com/books?id$=$wPjRMgEACAAJ

[14] Suzanne N. Hall and Nicole R. Robinson. 2012. Music and Reading: Finding Connections From Within. *General Music Today* 26, 1 (October 2012), 11–18. DOI:https://doi.org/10.1177/1048371311432005

[15] Erin E. Hannon and Laurel J. Trainor. 2007. Music acquisition: effects of enculturation and formal training on development. *Trends in Cognitive Sciences* 11, 11 (November 2007), 466–472. DOI:https://doi.org/10.1016/j.tics.2007.08.008

[16] Dee Hansen, Elaine D. Bernstorf, and Gayle M. Stuber. 2007. *The music and literacy connection*. Rowman & Littlefield Education, Lanham, Maryland.

[17] Michael Horn, Amartya Banerjee, Melanie West, Nichole Pinkard, Amy Pratt, Jason Freeman, Brian Magerko, and Tom McKlin. 2020. TunePad: Engaging learners at the intersection of music and code. (2020).

[18] Mike Horn, Amartya Banerjee, and Matthew Brucker. 2022. TunePad Playbooks: Designing Computational Notebooks for Creative Music Coding. In *CHI Conference on Human Factors in Computing Systems*, ACM, New Orleans LA USA, 1–12. DOI:https://doi.org/10.1145/3491102.3502021

[19] Roman Jakobson. 1959. On linguistic aspects of translation. In *On translation*. Harvard University Press, 232–239.

[20] Victor Lazzarini. 2013. The Development of Computer Music Programming Systems. *Journal of New Music Research* 42, 1 (March 2013), 97–110. DOI:https://doi.org/10.1080/09298215.2013.778890

[21] Sharona T. Levy and Uri Wilensky. 2009. Students' Learning with the Connected Chemistry (CC1) Curriculum: Navigating the Complexities of the Particulate World. *J Sci Educ Technol* 18, 3 (June 2009), 243–254. DOI:https://doi.org/10.1007/s10956-009-9145-7

[22] Yeping Li, Alan H. Schoenfeld, Andrea A. diSessa, Arthur C. Graesser, Lisa C. Benson, Lyn D. English, and Richard A. Duschl. 2020. Computational Thinking Is More about Thinking than Computing. *Journal for STEM Educ Res* 3, 1 (April 2020), 1–18. DOI:https://doi.org/10.1007/s41979-020-00030-2

[23] Olimpia Giuliana Loddo, Andrea Addis, and Giuseppe Lorini. 2022. Intersemiotic translation of contracts into digital environments. *Front. Artif. Intell.* 5, (October 2022), 963692. DOI:https://doi.org/10.3389/frai.2022.963692

[24] 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 and educat* 9, 1 (May 2016), 33–56. DOI:https://doi.org/10.1386/jmte.9.1.33_1

[25] Janet Mills and Gary E. McPherson. 2006. Musical Literacy. In *The Child as Musician*, Gary McPherson (ed.). Oxford University Press, 155–172. DOI:https://doi.org/10.1093/acprof:oso/9780198530329.003.0008

[26] William Payne and S Alex Ruthmann. 2019. Music Making in Scratch: High Floors, Low Ceilings, and Narrow Walls. *Journal of Interactive Technology and Pedagogy* 15, (2019), 2019.

[27] Christopher Petrie. 2022. Programming music with Sonic Pi promotes positive attitudes for beginners. *Computers & Education* 179, (April 2022), 104409. DOI:https://doi.org/10.1016/j.compedu.2021.104409

[28] Miller Puckette and others. 1996. Pure Data: another integrated computer music environment. *Proceedings of the second intercollege computer music concerts* (1996), 37–41.

[29] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (November 2009), 60–67. DOI:https://doi.org/10.1145/1592761.1592779

[30] Bruce L. Sherin. 2001. A Comparison of Programming Languages and Algebraic Notation as Expressive Languages for Physics. *International Journal of Computers for Mathematical Learning* 6, 1 (2001), 1–61. DOI:https://doi.org/10.1023/A:1011434026437

[31] Christopher Small. 1998. *Musicking: the meanings of performing and listening*. University Press of New England, Hanover.

[32] Evan S. Tobias. 2013. Toward Convergence: Adapting Music Education to Contemporary Society and Participatory Culture. *Music Educators Journal* 99, 4 (June 2013), 29–36. DOI:https://doi.org/10.1177/0027432113483318

[33] Iris Vessey. 1986. Expertise in Debugging Computer Programs: An Analysis of the Content of Verbal Protocols. *IEEE Transactions on Systems, Man, and Cybernetics* 16, 5 (1986), 621–637. DOI:https://doi.org/10.1109/TSMC.1986.289308

[34] Ge Wang. 2017. A history of programming and music. In *The Cambridge Companion to Electronic Music* (Second edition), Nick Collins and Julio d'Escrivan (eds.). Cambridge University Press, Cambridge, United Kingdom; New York, NY, 55–71.

[35] Brian N. Weidner. 2018. Content Area Literacy in Ensemble Music Education: The Before-During-After Instructional Framework. *Journal of Music Teacher Education* 27, 3 (June 2018), 10–23. DOI:https://doi.org/10.1177/1057083717732512

[36] Uri Wilensky and Seymour Papert. 2010. Restructurations: Reformulations of knowledge disciplines through new representational forms. *Constructionism* 17, (2010), 1–15.

[37] Uri Wilensky and Kenneth Reisman. 2006. Thinking Like a Wolf, a Sheep, or a Firefly: Learning Biology Through Constructing and Testing Computational Theories—An Embodied Modeling Approach. *Cognition and Instruction* 24, 2 (June 2006), 171–209. DOI:https://doi.org/10.1207/s1532690xci2402_1