TunePad: Engaging Learners at the Intersection of Music and Code

Michael S. Horn, Amartya Banerjee, Melanie West, Nichole Pinkard, and Amy Pratt

Northwestern University

Jason Freeman,
Brian Magerko, *Georgia Institute of Technology*
Tom McKlin, *The Findings Group*

michael-horn@northwestern.edu

Abstract: TunePad is a free, online platform designed with the goal of empowering diverse communities of learners to create and share music through code. We are interested in the idea of music as a pervasive form of literacy with abundant connections to concepts of computer programming. Over the past three years we have developed and refined successive prototypes with over 500 middle school and high school students in a variety of learning spaces including schools, libraries, summer camps, and other out-of-school programs. This paper shares the current TunePad design along with data from three summer camps for middle school students that involved daily work with the platform. Through these camps we saw significant gains in learners' attitudes around computer programming as measured through pre-post surveys. We also share a theoretical perspective on music and coding as an intersection of literacies that we reflect on through student-created artifacts.

Keywords: Computational literacy; music; diversity; design

Introduction

Computer science education opportunities for learners of all ages have expanded rapidly in the last 15 years. Notable examples include platforms like Scratch (Resnick et al., 2009), code.org, and App Inventor (Tissenbaum, Sheldon, Ableson, 2019); products like Osmo Coding (Hu et al., 2015) and Code-a-Pillar; and all manner of open, online courses. At a policy level, funding agencies have launched CS for all initiatives, while cities, states, and countries have begun mandating CS curriculum across the K-12 spectrum. Underlying these efforts is a perspective that computer programming should be a foundational skill for *all* students regardless of their ultimate career pathways (Vee, 2017). This is coupled with an awareness that the participation rates of women and marginalized groups in computing remains discouragingly low. As just one example, according to data from the Computing Research Association's Taulbee survey, women made up only 20.9% of bachelor's degrees awarded in Computer Science in the United States and Canada in 2018, while underrepresented minority students accounted for slightly less than 15% (Zweben & Bizot, 2019). Recent data on student participation in the high school AP computer science exams in the United States suggest that this situation is slowly starting to improve (http://home.cc.gatech.edu/ice-gt/597), but it's also clear that much more needs to be done to ensure a diverse computational future.

This paper describes continuing work on a free, online platform called TunePad (Figure 1, 2) that has the goal of empowering diverse communities of learners to create and share music through code. We are especially interested in the idea of music as a pervasive form of literacy with abundant connections to concepts and practices of computer programming. Our hope is that this might serve as a foundation for prolonged interest, learning, and creative expression for young people whose participation in computing has been historically marginalized.

We have been developing TunePad for three years as part of the growing EarSketch (Freeman et al., 2019) ecosystem of learning environments that combine music and computing. Over this time we have refined successive prototypes with over 500 middle school and high school students in a variety of learning spaces including schools, libraries, summer camps, and other out-of-school programs. This work culminated in a public beta release in late 2018. In this paper we share our current design along with data from three summer camps for middle school students that involved daily work with TunePad. Through these camps we saw significant gains in learners' attitudes around computer programming as measured through pre-post surveys. We also share examples of students' musical and computational artifacts from these camps.

Our contributions are primarily in the space of design—we believe that TunePad is a significant departure from existing offerings at the intersection of music and coding, particularly for educational purposes. We outline several novel design features that we have refined through many rounds of field testing with our target user population. We also expand on existing perspectives of computation as a literacy (diSessa, 2018; Vee, 2017), taking into account the idea that cultural movements of empowerment have historically involved a *subversion* of dominant literary forms and conventions. This perspective raises tensions around our understanding of mainstream CS education paradigms, while also suggesting new opportunities for learning.

Related work in music and coding

The idea of using music and code to support learning has a rich history. Jeanne Bamberger (2013) pioneered early conceptual groundwork, while more recent projects such as EarSketch (Freeman et al., 2019), Sonic Pi (Aaron & Blackwell, 2013), and Jython Music (Manaris, Stevens, & Brown, 2016) have demonstrated the broad appeal of this approach by engaging hundreds of thousands of students around the world. Many general purpose learning environments such as Logo (Papert, 1980), Scratch (Resnick et al., 2009), Pencil Code (Bau et al., 2015), and Arduino (arduino.cc) also support musical creation. However, these are general-purpose environments provide basic building blocks for generating sounds and notes, they aren't primarily organized around musical expression. This can severely limit and frustrate the ability of learners to engage in more serious musical explorations (Payne & Ruthmann, 2019). At the other end of the spectrum are more professional languages and frameworks such as ChucK, Max/MSP, SuperCollider, and Web Audio. These tools tend to be designed for professionals or more advanced learners. The ceiling for these tools is high, but, for the most part, they aren't appropriate for beginners. In the middle are learning environments such as Sonic Pi, Jython Music, and EarSketch. These environments tend to use text-based programming languages such as Python and are specifically designed to support the creation of music. For example, EarSketch is a web-based environment that features either Python or JavaScript. The creators of EarSketch focus on engaging students in computational music remixing, meaning creating musical compositions by combining pre-recorded loops and beats. "EarSketch learners are able to compose music and learn how to program without the added barrier of entry of learning music theory about harmony, melody, chord progressions" (Magerko et al., 2016). EarSketch also offers integrated curriculum, an audio loop library, social media connections, and a visual digital audio workspace (DAW). EarSketch is particularly notable for engaging diverse youth from backgrounds underrepresented in computing fields. Quantitative studies with EarSketch have included over 500 students and have shown statistically significant increases in intention to persist and attitudes toward computing, typically with medium or large effect sizes (Magerko et al., 2016; Freeman et al., 2019; Wanzer et al., 2020). TunePad was developed as a companion to EarSketch that focuses more on music composition from notes and beats.

TunePad differs from these related music+coding learning platforms along several dimensions that we describe in detail below. Our innovations include a computational notebook architecture, a unique approach to parallel code execution and musical synchronization, interactive musical instruments, rhythmic code tracing, and an interactive piano rolls integrated directly into Python code editors (Figures 1, 2).

Subversive computational literacy

In this work we build on the idea of computation as a form of literacy (diSessa, 2018; Papert, 1980; Vee, 2017). We adopt definitions of literacy offered by diSessa (2018) and Vee (2017) as a broad-based social-material intelligence that fundamentally shapes how we think and understand the world. The process of becoming literate plays out over a long period of time, is supported by a wide range of learning activities (both in school and out), and implies profound shifts in identity. In other words, becoming a literate person is a complex social, technical, and cultural phenomenon that goes well beyond taking a course or two in school. Both diSessa and Vee understand literacy as impacting entire societies across a wide range of human endeavors. diSessa, in particular, has critiqued current CS education movements as being too narrowly tied to the field of Computer Science. Or, as Vee writes, "because programming is so infrastructural to everything we say and do now, leaving it to computer science is like leaving writing to English or other language departments" (2017, pg 7). This sets up a sort of cultural imperialism in which the values, norms, and practices of Computer Science have shaped the discourse around educational standards, curriculum, and pedagogy. Because CS is largely a discipline of science and engineering, certain concepts and values (debugging, problem solving, efficiency) are given prominence over others. It is important to note here that there are many alternative framings that emphasize broader objectives such creativity, personal expression, and community (e.g. Grover & Pea, 2013; Brennan & Resnick, 2012; Kafai, 2016; Papert, 1980).

Adopting a literacy framework also forces us to consider the history of literacy in society. Vee draw parallels to medieval Europe, pointing out that as societies became dependent on written forms of communications and official documents, those who could read and write had outsized power and influence over those who could not. As a consequence, literacy, and the educational systems that support it, have historically been used as tools to maintain inequitable social, class, and economic power structures along with systems that reinforce those structures (Vee, 2017). Put bluntly, literacy can become as much a tool of oppression as it is empowerment depending on how it is imposed or denied to certain segments of the population. There are many examples from colonial societies that have dictated what counts as literacy while repressing or marginalizing indigenous languages and cultural knowledge systems. In the process these societies have denied literacy to segments of populations while using literacy (or illiteracy) as a wedge of disenfranchisement. Given this history, it's not surprising that social and political movements of empowerment have often been marked by a *subversion* of dominant literary forms and conventions. For example, hip hop and jazz can be thought of as not just musical genres but also transformations of poetry, dance, and art that emerged from a rejection of mainstream culture. For such movements, creating new forms of literacy can become an act of reclaiming power and identity (Lee & Soep, 2016).

The TunePad project is interested in what happens when you bring musical, social, and political ideologies embodied in movements like hip hop into collision with more established forms of coding literacy and CS education. As the literacies of coding and music collide, can we recognize, value, and make space for diverse ways of knowing and understanding that learners bring with them when they first encounter programming? Can we be open to a reimagining of the dominant forms of coding literacy that might make us uncomfortable as CS educators? Can coding be transformed (and transformative) in ways that might lead us to a more diverse computational future?

Relationship to computational thinking

Computational thinking (CT) is a related and useful construct that has received a great deal of attention in the literature (Brennan & Resnick, 2012; Grover & Pea, 2013; Shute, Sun, & Asbell-Clarke, 2017; Weintrop et al., 2016; Wing, 2006). In the mid-2000s, the term was revived with a strong cognitive focus that was explicitly tied to the field of Computer Science (Wing, 2006). Later definitions have broadened what we think of as computational thinking, but it has retained a strong connection to science, engineering, and technology. We view CT as a valuable perspective, but we prefer the term computational literacy as a more holistic framing that encompasses modern view of CT, while also opening paths to issues of social justice in the context of music and code.

Brief design overview

The design of TunePad is influenced by a long history of Constructionist learning environments that feature programming as a medium for creation and sharing. We draw inspiration from groundbreaking projects such as Logo (Papert, 1980), Scratch (Resnick et al., 2009), Sonic Pi (Aaron & Blackwell, 2013), and EarSketch (Freeman et al., 2019). Following Bamberger and diSessa (2003), we think of programming as a representational form that can reveal hidden algorithmic qualities of musical compositions. Or, in Wilensky and Papert's framing, programming is a *restructuration* (Wilensky & Papert, 2010) of traditional music notation systems that can fundamentally change the way we express musical ideas. We are also inspired by apps like GarageBand that do remarkable jobs of providing a playful, exploratory experiences that scaffold musical expression. We have borrowed from GarageBand the idea of providing multiple musical instruments that can be played with the mouse, keyboard, or touchscreen (Figure 2). We also think that it's important to lead with music (not code). In other words, we hope that learners will first see TunePad as a platform for creative expression and later recognize the role of code in empowering music.

When someone first visits the TunePad site, they can browse through a library of popular songs (Figure 1), follow step-by-step interactive tutorials, or listen to music created by other users. All of the music on TunePad is built entirely with Python code and can be remixed as a starting point for new projects. TunePad adopts a computational notebook paradigm has been popularized by tools such as Jupyter (Kluyver, 2016) and Mathematica, both widely used in data science communities. Projects take the form of interactive web pages that combine playable musical instruments, text, lyrics, multimedia elements, and runnable segments of Python code. As learners develop their projects, they can add new cells containing text, multimedia, and instruments (drums, bass, synths, and so on). For example, adding drums reveals a drumpad that can be played directly with a mouse, keyboard, or touchscreen (or attached MIDI device). To "record" tracks, users write short Python scripts that can also play the instrument. Figure 2 shows an instrument programmed to play a simple run of hi-hats containing random stutter steps (a common pattern in popular music).

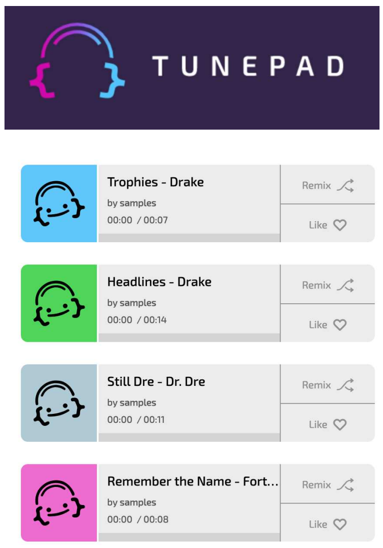
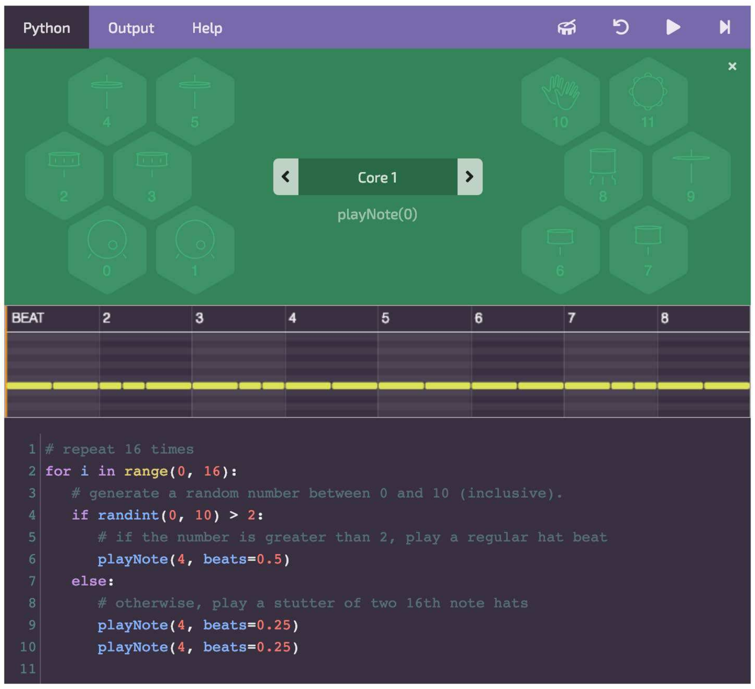


Figure 1. When users first visit TunePad, they can browse through a library of popular music or follow interactive tutorials. All of the music on TunePad is built with Python code, and anything can be remixed for new compositions.



```python
# repeat 16 times
for i in range(0, 16):
  # generate a random number between 0 and 10 (inclusive).
  if randint(0, 10) > 2:
    # if the number is greater than 2, play a regular hat beat
    playNote(4, beats=0.5)
  else:
    # otherwise, play a stutter of two 16th note hats
    playNote(4, beats=0.25)
    playNote(4, beats=0.25)
```

Figure 2. In TunePad, short Python scripts are tightly integrated with playable musical instruments (top) and interactive piano rolls (middle). These three elements form cells that can be embedded with other cells in interactive web pages called Dropbooks.

The computational notebook approach has a few important advantages for us. First, it organizes musical compositions around instruments while keeping Python code segments very short and manageable. This also creates a natural form of parallel code execution. As you press the Play button on each instrument on a project page in turn, the instruments synchronize themselves and layer together to form larger compositions. Second, the addition of text and multimedia elements can expand the opportunities for creative expression—students can add lyrics, personally meaningful images, or even video to their projects. Finally, it's easy to use projects to build interactive tutorials that describe the music and programming concepts involved in creating a project. We hope that TunePad users will develop their own tutorials to teach and learn from one another.

Instrument cells can also be added to a timeline view shown at the top of a project page. The timeline looks similar to the interface provided by most digital audio workspace (DAW) software packages. Once a cell is placed on the timeline, users can adjust the timing and duration of multiple audio samples using a simple drag-and-drop interface. This can facilitate the construction of more complex, full-length musical compositions.

One of our core innovations is to directly embed an interactive piano roll into the Python editor (Figure 2). This piano roll shows either the pattern of notes (pitches and durations) or the resulting audio waveform, depending on the user's preference. Every time a line of code changs, the piano roll automatically refreshes. This can even be done while the music is playing. Many integrated development environments (IDEs) provide visual debugging tools in which code can be stepped through line by line. Because music has tempo, we can automatically trace through code during playback in time with the music whenever a program is run.

Summer camps

We have been working on the design and development of TunePad for the past three years in collaboration with a number of partner organizations (including schools, libraries, community centers, and youth programs) serving primarily American and Latinx youth. In total, we have helped run over 20 events (ranging from around 60 minutes to 3-weeks in duration). More recently, we have begun formal data collection with IRB approval. Here we share data from three out-of-school camps with middle school learners conducted in the summer of 2019. All three camps emphasized technology and coding at the intersection of art, music, and performance. In collaboration with our partner organizations, our team participated in these events with daily sessions on music and coding involving around an hour of focused work with TunePad. A typical day would involve some form of independent warm-up activity, a 15-20 minute structured overview of core concepts from music and/or coding, and time for students to work individually or in pairs to practice applying the concepts. The first two camps were three weeks long and the third camp was one week long. All three camps featured musical performances and showcases as culminating events for parents, friends, peers, and families (Figure 3).

Our data included attitudinal and demographic surveys collected at the beginning and end of each camp; interviews and observations of student work; learner-created projects and artifacts; and TunePad log data. Before presenting the results, it is important to note that TunePad was only one part of larger programmatic activities that involved learning about the history of hip hop and social justice, engaging in making and tinkering activities, and learning about the work of real scientists. While TunePad was the major coding component of the camps, the results we present below should not be attributed to TunePad in isolation from these other activities.

Table 1: Participant responses to demographic questions on age, grade, gender, and ethnic/racial self-identity (N = 23). Participants could check multiple options for racial / ethnic self-identity

Racial / Ethnic Self-Identity	*Count*
Black or African American	14
Hispanic or Latino	0
White	6
Asian American	3
American Indian or Alaskan Native	1
Native Hawaiian or Other Pacific Islander	1
Multiple Race or Other	2

Age		*Grade*		*Gender*	
10 years	10	5th grade	7	Girls	13
11 years	5	6th grade	8	Boys	10
12 years	6	7th grade	5	Other	0
13 years	2	8th grade	3	No answer	0

Figure 3. Participants at a camp showcase presentation.

Participants
Table 1 provides an overview of camp students who participated in this study. We included all students who completed both the pre-and post-survey at any of the three summer camps. Students who were not present on either the first or last day of a camp were omitted. Some students also attended both of the three week camps. For these participants, we only included their pre- and post-responses from the first three-week session. The first 3-week camp had 10 participants who completed both surveys, while the second 3-week camp had 4 additional participants who had not attended the first session. The 1-week camp had 9 participants. Based on an analysis of log data, these students created over 215 projects including over 12,000 lines of code. It was also not uncommon for students to make edits outside of camp hours or on weekends.

Survey results
The attitudinal survey contained seven Likert-scale items on computer programming that asked learners to report on their self-confidence, enjoyment, interest, perceived usefulness, identity, and intention to persist (see Table 2). These items were derived from the validated Attitudes Towards Computing scale (Wanzer et al., 2019). The original scale had 19 items organized into 5 constructs. Because we were conducting a summer camp in an informal learning space, we reduced the number of items to shorten the time required to complete the survey.

To analyze student surveys, we constructed a composite scale from each students' average responses to the seven items on computer programming. Before constructing the scale, we computed Cronbach's alpha values for both pre- and post-responses. The results (pre = 0.897; post = 0.862) indicated high internal consistency for these items. We further conducted an exploratory factor analysis, revealing that all 7 items loaded onto the same factor.

Table 2: Participant responses to pre- and post-attitudinal items on a five-point Likert scale adapted from the Attitudes Towards Computing Scale (Wanzer et al., 2019).

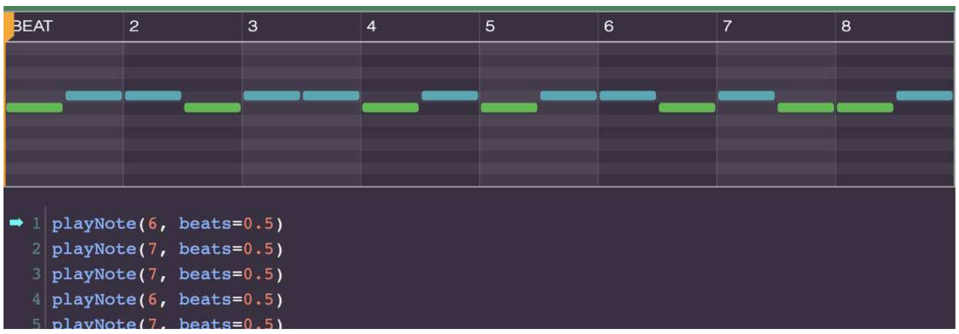
Item	**n**	**Pre (mean)**	**Post (mean)**
1. I have a lot of self-confidence when it comes to computer programming.	23	3.30	3.70

2. I enjoy computer programming.	23	3.83	4.09
3. I will use programming in many ways throughout my life.	22	3.17	3.64
4. I like solving programming problems.	23	3.39	3.70
5. I think of myself as a computer programmer.	22	2.64	3.13
6. I like sharing computer programming projects I've made with friends and family.	23	3.09	2.87
7. I can see myself taking computer science classes in school.	23	3.48	3.57
8. I enjoy making music.	21	4.33	4.22
9. I think of myself as a musician or music producer.	21	3.33	3.48
10. Composite Scale (items 1-7)		3.28	3.53

After constructing the composite scale, we conducted a paired, one-tailed t-test. Results indicate that there was a significant increase from the pre- to post-survey: $t(22)=2.19, p=0.048$, 95% CI [0.002, 0.503]; D=0.44. The effect size (Cohen's D) is considered moderate for educational interventions (Hatti, 2012). Looking at individual items, students seemed to more strongly identify with being a musician or music producer than a computer programmer, although the values for both increased from pre to post. This suggests some merit in the idea of leading with music rather than leading with code, but this is only one piece of evidence. Responses to "*I enjoy making music*" were also higher than "*I enjoy computer programming*" at both the pre and post, although the music enjoyment item was slightly lower on the post-survey. These results should be interpreted with the understanding that the N was small and that TunePad activities accounted for only one part of the broader summer programs.

Example student projects

Serena was a camp participant absorbed in the TunePad challenges we gave her, to the point that when it was time to go to lunch she often wanted to continue working. After a week of learning the fundamentals, participants were given the challenge of composing a song motivated by an image that inspired them. Serena chose a calla lily flower and her composition turned out to be quite complex filled with polyrhythms. Her project included 14 tracks of drums and bass, each with its own Python script featuring variables, loops, and even her own function. Serena was unusually creative with her rhythmic patterns. One of those interesting patterns was composed of two measures of eighth notes that playfully give the feeling of call and response when listened to repeatedly. There is a slight change of the motif in the second measure that reflects a variation on the initial theme. In this pattern, Serena uses 2 tom toms, one at a relatively high pitch and the other one at a lower pitch. She (perhaps unknowingly) invokes the quality of syncopation by placing some of the higher pitched tones on the up beat accentuating the weaker beats.



```
playNote(6, beats=0.5)
playNote(7, beats=0.5)
playNote(7, beats=0.5)
playNote(6, beats=0.5)
playNote(7, beats=0.5)
```

Figure 4. Serena's composition.

Keneda: Music was an inspiration for 10-year-old Keneda who we met as part of a local weekend STEM festival. Upon arriving each morning at camp, Keneda would immediately grab her laptop, and she often worked on projects outside of camp hours. She was focused and determined to translate Anna Kendrick's "Cups–When I'm Gone", a song that she loved, into Python code. Keneda found a piano tutorial on YouTube that showed what notes to play in both the bass and treble clefs. She coded the melody for the treble clef (right hand) using her ear for note duration and a tutorial for the actual pitches on the TunePad keyboard. After a long period of hard work on her project, she asked one of the facilitators how to play two notes at the same time. We suggested that she add a new Dropbook cell for the left hand part of her melody. When it came to coding the left hand she decided to improvise and come up with her own version of the bass part of the composition. She combined a piano riff with a pedal bass line. One of the main features in "Cups–When I'm Gone" is the distinctive rhythm made with a simple cup. Keneda included her version of that rhythm using finger snap and sidestick sounds she found in the TunePad instrument bank. She coded 4 bars of the rhythm using a for loop. Next, she had to figure out how to combine the parts and sync them up into her song structure. After hours of trial and error Keneda was not only able to compose her own version of the song (that she was proud of and eager to share with others), she had mastered some Python syntax and sequencing of code while engaging in authentic problem solving.

Discussion and future work

This paper presents TunePad, an online platform with the goal of empowering diverse communities of learners to create and share music with code. Although there are many related projects at the intersection of music and coding, we make the case that TunePad contributes several unique design innovations. Most important, TunePad projects take the form of interactive web pages that adopt a computational notebook paradigm. Building on this structure, we have developed Python code cells that closely integrate playable musical instruments, interactive piano rolls, and rhythmic code tracing. Multiple cells on a page operate independently but also synchronize themselves to a global clock. This allows for a unique form of parallel code execution and live coding.

We also share a perspective on music and coding as the intersection of literacies. Building on concepts of computational literacy, we add the observation that cultural movements of empowerment have often involved a subversion of dominant and established forms of literacy. With TunePad we are interested in what happens when the literacy of music collides with the literacy of coding. And, what happens when cultural movements like hip hop come into conjunction with more established forms of CS education?

Looking at survey data, our camp participants showed significant attitudinal gains in interest, self-confidence, enjoyment, and intention to persist in computing. More importantly, our descriptive cases of student engagement with TunePad highlight some of the motivation and authentic problem solving that learners bring into computational music projects. Our future work will involve UI improvements as well as enhancements to the quality of music it is possible to create. We will also conduct more longitudinal studies of student engagement as we are especially interested in the emergence of youth-driven communities at the intersection of music and code.

References

Aaron, S. & Blackwell, A.F. (2013). From Sonic Pi to Overtone: creative musical experiences with domain-specific and functional languages. In ACM SIGPLAN workshop on Functional art, music, modeling & design. ACM, 35–46.

Bamberger, J. (2013). *Discovering the musical mind: A view of creativity as learning*. Oxford University Press.

Bamberger, J. & diSessa, A. (2003). Music as Embodied Mathematics: A study of mutually informing affinity. *International Journal of Computers for Mathematical Learning 8*, 123–160.

Bau, D., Bau, D. A., Dawson, M., & Pickens, C. S. (2015). Pencil code: block code for a text world. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 445-448).

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In the annual meeting of the American Educational Research Association.

diSessa, A. (2018). Computational literacy and "the big picture" concerning computers in mathematics education. Mathematical thinking and learning 20, 1 (2018), 3–31.

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.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Ed. Researcher*, *42*(1), 38-43.

Hattie, J. (2012). *Visible learning for teachers: Maximizing impact on learning*. Routledge.

Hu, F., Zekelman, A., Horn, M., & Judd, F. (2015). Strawbies: explorations in tangible programming. In Interaction Design and Children. ACM, 410–413.

Kafai, Y. (2016). From computational thinking to computational participation in K–12 education. Comm. ACM 59, 8, 26–27.

Thomas Kluyver, et al. Positioning and Power in Academic Publishing: Players, Agents, and Agendas. IOS Press, Chapter Jupyter Notebooks—a publishing format for reproducible computational workflows, 87–90.

Clifford H Lee and Elisabeth Soep. 2016. None but ourselves can free our minds: Critical computational literacy as a pedagogy of resistance. Equity & Excellence in Education 49, 4 (2016), 480–492.

Brian Magerko, Jason Freeman, Tom Mcklin, Mike Reilly, Elise Livingston, Scott Mccoid, and Andrea Crews-Brown. 2016. Earsketch: A steam-based approach for underrepresented populations in high school computer science education. ACM Transactions on Computing Education (TOCE) 16, 4 (2016), 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, 33–56.

Margolis, J. (2010). Stuck in the shallow end: Education, race, and computing. MIT Press.

Seymour Papert. 1980. Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc.

Resnick, Maloney, Monroy-Hernández, Rusk, Eastmond, Brennan, Millner, Rosenbaum, Silver, Silverman, and others. (2009). Scratch: Programming for all. Comm. ACM 52, 11, 60–67.

Shute, V., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. Ed. Research Review 22, 142–158.

Sebastien Siva, Tacksoo Im, Tom McKlin, Jason Freeman, and Brian Magerko. 2018. Using Music to Engage Students in an Introductory Undergraduate Programming Course for Non-Majors. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education. ACM, 975–980.

Tissenbaum, M., Sheldon, J. & Abelson, H. (2019). From computational thinking to computational action. Comm. ACM 62, 3.

Annette Vee. 2017. Coding literacy: How computer programming is changing writing. MIT Press.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, *25*(1), 127-147.

Dana Wanzer, Tom McKlin, Doug Edwards, Jason Freeman, and Brian Magerko. 2019. Assessing the Attitudes Towards Computing Scale: A Survey Validation Study. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education. ACM, 859–865.

Wanzer, D. L., McKlin, T., Freeman, J., Magerko, B., & Lee, T. (2020). Promoting intentions to persist in computing: an examination of six years of the EarSketch program. *Computer Science Education*, 1-26.

Uri Wilensky and Seymour Papert. 2010. Restructurations: Reformulations of knowledge disciplines through new representational forms. Constructionism (2010).

Jeannette M Wing. 2006. Computational thinking. Commun. ACM 49, 3 (2006), 33–35.

Stuart Zweben and Betsy Bizot. 2019. 2018 CRA Taulbee Survey. Computing Research News 31, 5 (2019).