#11: Make a Music Box and Program a Tune

Walter Bender & Devin Ulibarri

All musicians are subconsciously mathematicians.
—Thelonious Monk, jazz pianist and composer

Why Music?

Enumerated by Seymour Papert and Cynthia Solomon as "thing to do" #11, music is used as an entry point for engaging in computational thinking. Computation and music share important concepts. And music, like computation, requires a person to think on an abstract level. Moreover, music has some benefits that the traditional computing pedagogy does not, including the potential to improve social-emotional skills. See Table 1 for some examples of shared concepts between music and programming.

Music composition and performance require practitioners to follow basic control flow such as: sequences; conditionals and loops; data abstractions such as changes in timbre, tone, and meter; functions and operators such as transpositions, inversions, and retrograde; and debugging—making corrections to a composition, perfecting a transcription, or working through a section of music on an instrument—that leads to a deeper understanding of music theory.

The social aspect of musical performance also parallels the perspective that computing is both collaborative and creative (Brennan & Resnick, 2012). An analog can be built between the way programmers work together, building communities around sharing and remixing code, and the way in which musicians build communities of interest through performance, sharing, and debating best practices. Programmers review code and musicians critique performances. Both musicians and programmers modify, improvise, and derive inspiration from the work of peers and mentors.[1]

According to the National Endowment for the Arts: "Childhood arts education provides important gateway and formative experiences in the arts… School-based arts education is of particular importance because schools are the only institutions that reach vast numbers of children, particularly low-income children, who are unlikely to receive arts education any other way… [A] body of solid research and innovative practice continues to grow and show that arts education has serious benefits to students as students, and that arts learning is strongly associated with higher levels of achievement, positive social and emotional development, and successful transitions into adulthood" (Bawa et al., 2010).

In another study, "an analysis of longitudinal data on 25,000 students showed that those with higher levels of involvement with the arts did better across a wide range of outcome variables than those with lower arts involvement, and that low-income students benefited from their involvement in the arts more significantly than did higher-income students" (Catterall et al., 1999).

Students demonstrably benefit from an engaging music education. Demand today is in science, technology, engineering, and mathematics (STEM) and EdTech. Why not give people and institutions the option to learn both STEM and music?

Programming concept	**Musical counterpart**
Sequences	A series of notes (or phrases), in order
Loops	Repeating phrases, drum loops
Conditionals	Using conditionals for 1st and 2nd endings
Data structures	Note structure (note length, pitch name, and pitch octave) and phrase structure
Modularity and abstraction	Actions, transpositions, intervals, ornamentation, inversions, etc.
Debugging	Using one's ear, e.g., does the result sound correct? Understanding of music theory, e.g., where is the meaningful musical structure?

Table 1: Shared concepts: programming and music

Computation, Not Coding

As recently as 2006, during the launch of *One Laptop per Child* (Bender et al., 2012), there was pushback from educators and pundits regarding children using computers. But fifteen years and one global pandemic later, few argue against children having access to computing. The role of computation in learning, however, remains open to debate. In an article in the *New York Times* (Singer, 2017), Tim Cook, CEO of Apple, Inc., representing the point of view of Silicon Valley, said "Coding should be a requirement in every public school." But to what end? To ensure a "skilled workforce"? Beyond the false promise that by learning Java you will get a job at Google, we take the position that *learning to code* is not the same as becoming literate in *computational thinking*. Computational thinking, rather, is "solving problems using techniques from computing" (Sharples et al., 2015) and it has ramifications far beyond job training; it is about expression of ideas, problem solving, and creativity, all important life skills.

Acknowledging that computational thinking is about more than learning to program leads to a number of questions: How should we go about leveraging the latent capacity to learn? How can we transform a consumer-oriented culture into a learning-oriented culture? And is it possible to design a learning platform that respects the diversity of educational context found across a diversity of learning populations?

Music Is "Hard Fun"

In a 2008 memo "Questioning General Education," Marvin Minsky proposed that we "re-aim our schools towards encouraging children to pursue more focused hobbies and specialties—to provide them with more time for (and earlier experience with) developing more powerful sets of mental skills, which they later can extend to more academic activities." Minsky goes on to argue that the organization of our cognitive resources into towers with different levels of processes is what "enables our minds to generate so many new kinds of things and ideas." These levels span agencies, each of which specializes in areas such as gaining knowledge from experience, planning and causal attribution, the construction of models, and identifying values and ideals. A focus on achieving meaningful goals, not just the accumulation of simple knowledge objects, exercises all of the levels in a cognitive tower, helping a child "develop proficiencies that can be used in other domains" (Minsky, 2019).

A focus on achieving meaningful goals, not just the accumulation of trivial knowledge, helps a child "develop proficiencies that can be used in other domains." A focus on hobbies, where interest is authentic and sustained, as opposed to curricula organized around the sequential achievement of fragmented goals, has the potential for deep engagement across multiple levels. Albert Einstein summed up the focus on hobbies succinctly when he said, "Love is a better master than duty." It was in this spirit that Minsky's Music Box spawned the music ideas in the *Twenty Things* memo (Solomon et al., 2020).[2] Music was and is a vehicle for deep engagement.

Note the emphasis on "deep" engagement. There is a strong temptation to make things as simple as possible so as to reach the broadest possible audience. However some things (e.g., music and computation) are inherently complex. The hard part of "hard fun" of learning is in reaching towards complexity. Using an app is easy. Writing an app is harder. Listening to music is easy.[3] Performing and composing music is harder. Children should not miss out on the learning that takes place when engaging with complexity.

Musical Microworlds

Papert used the term *microworld* to describe the world of geometry explored when children used Logo. A microworld is a "subset of reality or a constructed reality whose structure matches that of a given cognitive mechanism so as to provide an environment where the latter can operate effectively. The concept leads to the project of inventing microworlds so structured as to allow a human learner to exercise particular powerful ideas or intellectual skills." (Papert, 1980)

In a microworld, an individual is able to use a technological tool for thinking and cognitive exploration that would not be possible without the technology. But not just any technology. "The use of the microworld provides a model of a learning theory in which active learning consists of exploration by the learner of a microworld sufficiently bounded and transparent for constructive exploration and yet sufficiently rich for significant discovery" (Papert, 1980).

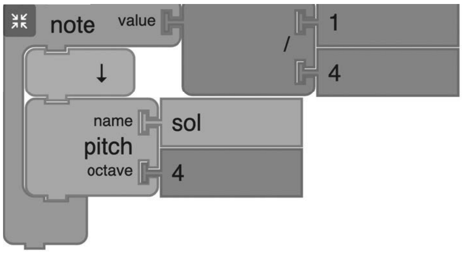
In a microworld of music, a student might start by exploring pitch and rhythm while using affordances for repetition, transposition, etc. The tools are more than an interface to a synthesizer and more than a transcription/engraving tool (e.g., Finale, Sibelius, Musescore, etc.)—they are scalable and modular collections of essential building blocks that are at the crux of all powerful ideas in music.

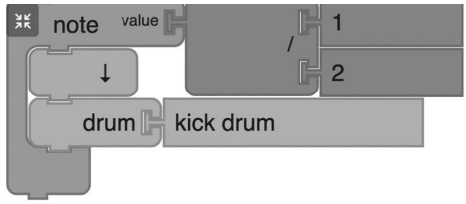
The microworld is designed to introduce a specific concept with parallels in both music and computer science. For example, a workshop on rhythm utilizes a rhythm making tool that introduces the concept of loops, which are used for drum machines implemented by *while* loops.

Music Blocks

"Music is a hidden arithmetic exercise of the soul, which does not know that it is counting." — Leibniz

Music Blocks[4] (musicblocks.sugarlabs.org) is a visual programming language and collection of manipulative tools for exploring musical and mathematical concepts in an integrative and fun way. Music Blocks is a fork of Turtle Blocks,[5] an activity with a Logo-inspired graphical turtle that draws colorful art based on snap-together visual programming elements. Its *low floor* provides an easy entry point for beginners. It also has *high ceiling* programming, graphics, mathematics, and computer science features that will challenge the more adventurous student.





Figures 1 and 2: Note-value representation as visual programming in Music Blocks. Note value is a quanta of time expressed as a ratio; ¼ is the same as a quarter note in music. Whatever is contained within the note value clamp will be done over the length of time of a quarter note. The example on the left will perform G (in the 4th octave) for the length of a quarter note; the example on the right will perform a kick drum sound for the length of a quarter note.

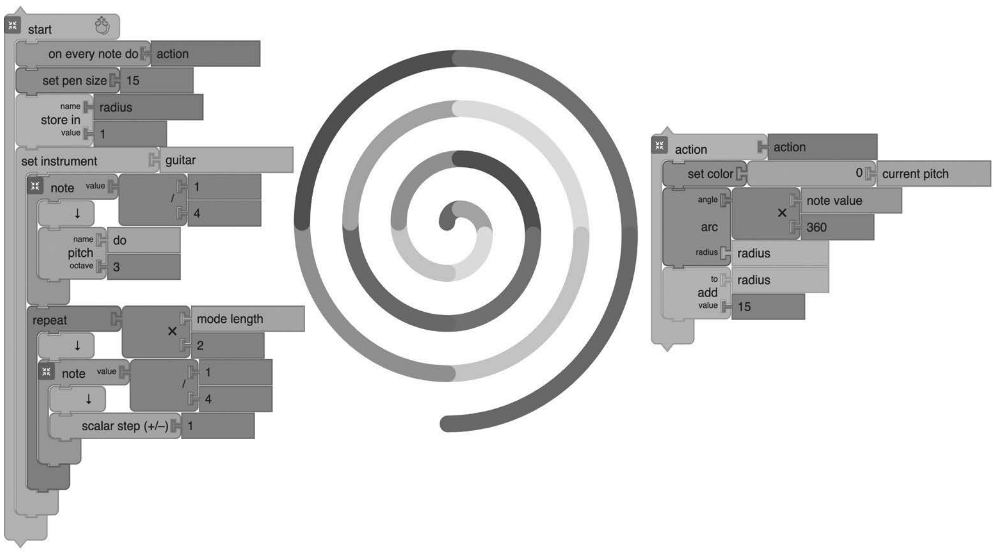


Figure 3: In this example, two octaves are played during which a portion of a spiral is drawn with each note. The angular distance traversed is determined by the note's value. The line color is determined by the note's pitch. The resulting spiral serves as an example of an alternative notation system created by a student using Music Blocks.

Music is not an add-on to the Music Blocks language; the *note* is a core datatype in the language (See Figures 1–3). The note expresses a quanta of time as a ratio, because the relative length of a note and their relationship to other notes is the fundamental basis of rhythm.[6] From there, Music Blocks has various ways to express pitch: solfege (e.g., do, re, mi), musical alphabet (e.g., C, D, E), and Hertz (e.g., 440Hz for A in an equal temperament tuning system), which are pedagogically important expressions of pitch for making music.[7]

Music, like computer science, offers a rich environment for exploration and problem solving, of which the intersection of their shared concepts allow for integrative learning. Music Blocks is designed to leverage the tools of the trade by using well-established scaffolding in music instruction and building analogous tools (or widgets) to help shape the introduction of powerful ideas, such as polyphonic rhythms, key and mode, intervals, tuning, and temperament. The software widgets are also tied to concepts in computer science. In other words, the widgets don't just produce music, they output code that is descriptive of concepts found in music, such as generating rhythms, changing tempo, and using samples.

The configuration of each widget is programmable (using blocks) and every widget can be used to write and export code (as blocks). Along with the activities defined by the widgets, students also: identifying and differentiate patterns; explore proportions, ratios, and relationships expressed through chords and intervals; read and represent musical ideas with a graph; use conditionals to express a melodic sequence; and program canons to explore concurrency.

The activities in which the students engage are constructive rather than instructive. Music Blocks provides scaffolding without over specifying the end results. Consequently, there are also numerous mechanisms to support *debugging* of both the music and the code.[8] In the spirit of Papert and Solomon, some things to do with Music Blocks are listed in Table 2.

1. Animate polyrhythms 2. Broadcast conductor instructions 3. Program a canon 4. Explore circles of fifths 5. Program a circular rhythm maker 6. Use conditionals in music 7. Explore representations 8. Musical cookie hunt 9. Design a crazy keyboard 10. Debug a composition 11. Design theme music for heroes and monsters for a video game 12. Use events for interactivity	13. Explore beat and rhythm 14. Explore chance 15. "Face the Music" 16. Use dictionaries to control ornamentation 17. Design a notation system 18. Explore intervals. 19. Make a piano 20. Build a better mousetrap 21. Make a music video 22. Program *Music Deducto* 23. Make music paint 24. Use an oscilloscope 25. Hunt for hidden music 26. Explore quiet and loud 27. Incorporate sensors for interactivity	28. Program strange loops (recursion) 29. Build a record player 30. Explore musical symmetry 31. Explore synesthesia 32. Use the heap to explore variations on a theme 33. Build a xylophone 34. Make a musical valentine 35. Program *Musical Racko* 36. Program a version of Set, where the dimensions are musical 37. Build a metronome 38. Build a one-string guitar 39. Explore harmonics 40. Invent 40 more ideas

Table 2: Some things to do with Music Blocks

Beyond Music Blocks

A musical microworld is not a destination. Rather, it is a waypoint along a road to achieving fluency in both musical comprehension and computational thinking. We want the students to dive deeper into musical representations and programming constructs than they could do in a single session. Therefore we provide mechanisms to go beyond tools such as Music Blocks to give the learner both the ability to communicate with the mainstream worlds of music and computer science and access to a rich set of tools that they may use to further augment their explorations. Our workshops, by design, do not confine a user to its tools—rather it is a tool to propel the ambitious learner to other rich and authentic discoveries (Bender et al., 2016).

```
\version “2.18.2”
mouse = {
\meter
\tempo 4 = 75
 \time 4/4
<e’ c’>2 <f’ a>4 <f’ a>4 <e’ c’>2 <d’ g>4 r4
<e’ c’>2 <g’ g>4 <g’ g>4 <g’ c’>2 <a’ f>4 g’4
<e’ c’>2 <f’ a>4 <f’ a>4 <e’ c’>2 <d’ g>4 r4
<e’ c’>2 <g’ g>4 <g’ g>4 <g’ c’>2 <a’ f>4 g’4
}
\score {
<<
\new Staff = “treble” {
\clef “treble”
\set Staff.instrumentName = #”mouse” \mouse
}
>>
\layout { }
}
```

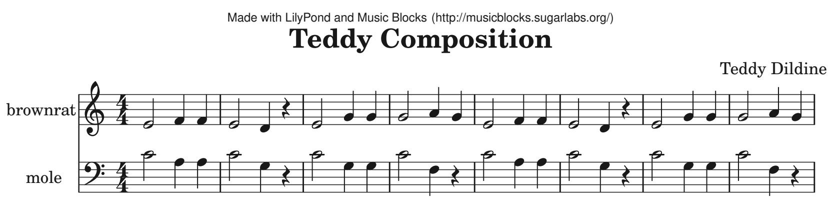


Figure 4: Lilypond is a music engraving program. The Save as Lilypond option in Music Blocks exports a Music Blocks composition into sheet music, thus connecting traditional musical notation with the block language. (Shown is part of a composition by then seven-year-old T. Dildine.)

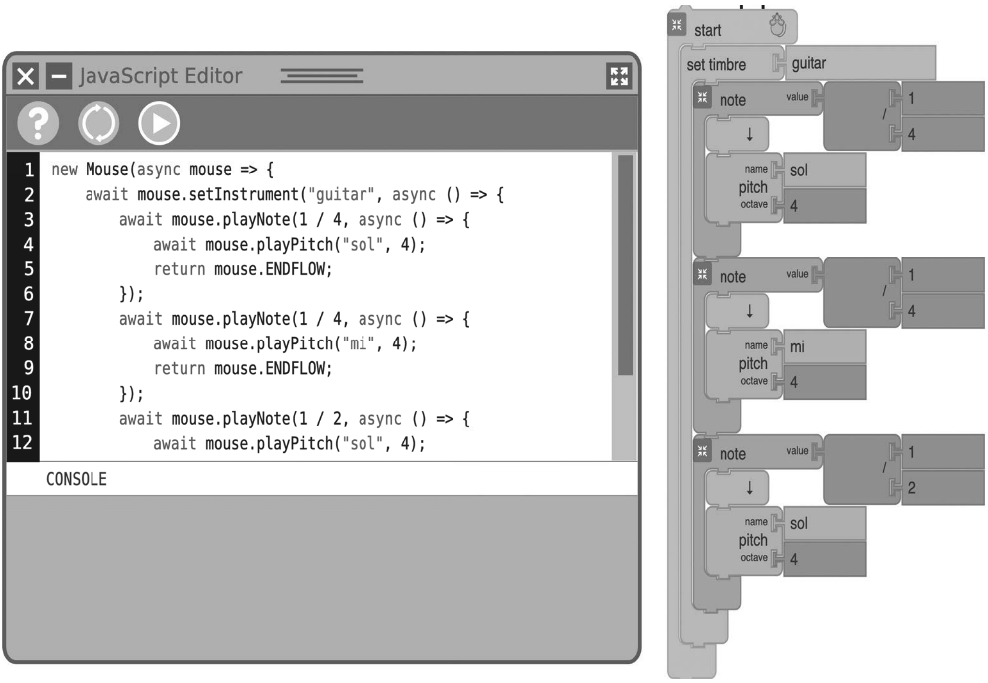


Figure 5: There are practical limits to the size and complexity of Music Blocks programs. At some point we expect Music Blocks programmers to move on to text-based programming languages. To facilitate this transition, there is a JavaScript widget that will convert a Music Blocks program into JavaScript. (This feature was added to Music Blocks by Anindya Kundu for Google Summer of Code.)

Music Blocks and Assessment

It is unrealistic to propose an intervention in education without acknowledging the current emphasis on measurement and evaluation. With Music Blocks, the goal is to ensure that any interventions have some positive socio-economic impact on children, so an evaluation is used that looks more broadly than those data that are captured by standardized tests. Evaluation occurs at different levels (Urrea & Bender, 2012): micro (at the level of individual students, teachers, and parents); mezzo (at the level of a classroom or school); and macro (national and global indicators). These mechanisms, briefly reviewed below, are orthogonal to the typical standardized-testing regimes; the two approaches—one serving administrators, the other serving learners—coexist.

At the micro level, Music Blocks maintains a digital portfolio to support reflection that can help students (as well as teachers and parents) become aware of their own learning, and do so by documenting their work and thinking over time. Digital portfolios are part of a "comprehensive system that combines formal, informal, and classroom assessment, including portfolios, to inform the state, the district, the school, and the teacher" (Stefanakis, 2002). Without a way to make visible what students do and what teachers teach, it is difficult to make changes to improve those dynamics. Both music and coding produce artifacts that are readily captured in a digital portfolio—for example, musical compositions, code, and geometric art. As with a source-code management system, each *commit* is accompanied by a *commit message*, where the learner is asked to document their work.

At a mezzo level, Music Blocks has tools that help teachers understand the impact and evolution of the program in a larger context—at the level of the classroom or the school. The goal is to navigate and visualize data automatically derived from the learning activities in which the learners are engaged. These data help teachers, administrators, and other stakeholders understand the impact of a program and make adjustments to it. Music Blocks has a built-in set of rubrics (Bender & Urrea, 2015) that can be used to visualize and track student progress.

As a macro level, Music Blocks supports strategies for understanding the use of computation in learning at a much larger scale. These strategies involve the design and implementation of a repository of objects or artifacts designed by children from different programs. There are a number of similar repositories with artifacts from an individual already in existence, for example, the Scratch website and the Music Blocks Planet (server with student published projects, accessible world-wide). Such collections make possible the analysis and understanding of impact at a large scale, and the learning that emerges, not only at the individual, but also at the collective level. In the respect that these sites allow users to *remix*, they bear a close resemblance to popular tools used in computer engineering, such as GitHub and GitLab.

Music+Code Teaching Artists

Around the time that he invented the Music Box, Minsky served as the AI consultant on Stanley Kubrick's movie adaptation of Arthur C. Clark's *2001: A Space Odyssey*. Both the HAL 9000 computer and the notion that children would be programming music were considered science fiction in 1971. Fifty years later, Siri et al. can "open the pod-bay doors" and play a competitive game of chess; any child with access to a web browser can program music with Music Blocks. But who will mentor these children? Who will guide them in their exploration of a musical microworld? Who will introduce them to the powerful ideas inherent in both music and computation?

STEM education is in high demand. Policy makers are mandating schools teach computer programming in their classrooms, which creates a demand for teachers trained in computation. Primary education, however, is not seen as a competitive or desirable career for many who have computer programming skills. So who will teach these skills?

Music education, with its rich blend of theory and practice, has proven benefits to early education and yet, because of financial and time pressures, it is largely sidelined in public education. Is there a way we can capture the benefits of music education that has been pushed aside in order to advance STEM education?

Musicians graduate from conservatories and colleges with few career prospects other than part time teaching jobs and freelance performance gigs. Professionals who specialize in music and who teach music both inside and outside of traditional settings are called "Teaching Artists" (Booth, 2009). Teaching Artists, even those working at the world's most prestigious institutions, typically work multiple part-time jobs in order to scrape by financially (National Endowment for the Arts, 2011b).

The many skilled Teaching Artists who are graduating at a steady rate from music schools can introduce young learners to programming and programming can be used as a vehicle to surface the powerful mathematical structure inherent in music. Teaching Artists can play a central role in primary STEM education, while simultaneously providing the proven cognitive and social-emotional benefits of a quality music education (Gaser & Schlaug, 2003; Hutchinson et al., 2003; Lee et al., 2003; Schlaug, 2001; Skoe & Kraus, 2012). Engaging and employing Teaching Artists into the booming economy of EdTech provides rich exposure to computational thinking, while not compromising music's proven benefits to a child's cognitive development and overall well-being.

While music is not typically associated with computer science, there is a depth of literature (Brindle, 1987; Garland & Kahn, 1995; Nierhaus, 2009) drawing parallels between music and mathematics. By focusing on the parallels between music and computation, and the knowledge and insights gained through designing and critiquing their musical and programmatic constructs, Teaching Artists offer novel perspectives of both music and computer science to learners, educators, and the public.

Music is a universal language. Leveraging music and music teachers has the potential to reach a global audience. Under-employed musicians, who are statistically proven to be highly educated and demographically diverse (National Endowment for the Arts, 2011a), are a global phenomenon, one that we leverage both in terms of mining latent skills and also filling a need for more diverse and engaging mentorship in computational thinking. Teaching Artists using Music Blocks will foster basic computer science skills and broaden public engagement in computer science in a large, diverse population that has been underserved by existing efforts for broadening participation in computer science and engage and enable a talented but underemployed group—music teachers—in fulfilling both pedagogical and societal needs.[9]

References

Bawa, S., Williams, K., & Dong, W. (2010). *Audience 2.0: How Technology Influences Arts Participation*. National Endowment for the Arts.

Bender, W., Kane, C., Cornish, J., & Donahue, N. (2012). *Learning to Change the World: The Social Impact of One Laptop Per Child*. St. Martin's Publishing Group.

Bender, W., Ulibarri, D., & Khandelwal, Y. (2016). Music Blocks: A Musical Microworld. *Constructionism 2016, Constructionism in Action*.

Bender, W., & Urrea, C. (2015). Visualizing Learning in Open-Ended Problem Solving in the Arts. *RED-Revista de Educación a Distancia*, *46*(2). doi.org/10.6018/red/46/2

Booth, E. (2009). *The Music Teaching Artist's Bible*. Oxford Univ. Press.

Brennan, K., & Resnick, M. (2012). New Frameworks for Studying and Assessing the Development of Computational Thinking. *Proceedings of the 2012 annual meeting of the American Educational Research Association (AERA)*, *1*, 25.

Brindle, R. S. (1987). *The New Music: The Avant-garde Since 1945*. Oxford Univ. Press.

Catterall, J., Chapleau, R., & et al. (1999). Involvement in the Arts and Human Development: General Involvement and Intensive Involvement in Music and Theatre Arts. In E. Fiske (Ed.), *Champions of Change: The Impact of the Arts on Learning* (pp. 1-18.11). Washington DC: Arts Education Partnership and President's Committee on the Arts and Humanities.

Garland, T. H., & Kahn, C. V. (1995). *Math and Music: Harmonious Connections*. Seymour (Dale) Publications.

Gaser, C., & Schlaug, G. (2003). Brain Structures Differ between Musicians and Non-Musicians. *Journal of Neuroscience*, *23*(27), 9240-9245. doi.org/10.1523/JNEUROSCI.23-27-09240.2003

Hutchinson, S., Lee, L. H., Gaab, N., & Schlaug, G. (2003). Cerebellar Volume of Musicians *Cerebral Cortex*, *13*(9), 943-949. doi.org/10.1093/cercor/13.9.943

Lee, D., Chen, Y., & Schlaug, G. (2003). Corpus Callosum: Musician and Gender Effects. *Neuroreport*, *14*(2), 205-209. doi.org/10.1097/00001756-200302100-00009

Minsky, M. (2019). Questioning General Education. In C. Solomon & X. Xiao (Eds.), *Inventive Minds: Marvin Minsky on Education*. The MIT Press.

National Endowment for the Arts. (2011a). Artists and Arts Workers in the United States. *NEA Research Note #105*.

National Endowment for the Arts. (2011b). Findings from the American Community Survey (2005-2009) and the Quarterly Census of Employment and Wages (2010). *NEA Research Note #105*.

Nierhaus, G. (2009). *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books.

Schlaug, G. (2001). The Brain of Musicians: A Model for Functional and Structural Adaptation. *Annals of the New York Academy of Sciences*, *930*, 281-299. doi.org/10.1111/j.1749-6632.2001.tb05739.x

Sharples, M., Adams, A., Alozie, N., & Ferguson, R. (2015). *Innovating Pedagogy 2015* (Report No: 4).

Singer, N. (2017, June 17). How Silicon Valley Pushed Coding Into American Classrooms. *New York Times*

Skoe, E., & Kraus, N. (2012). A Little Goes a Long Way: How the Adult Brain Is Shaped by Musical Training in Childhood. *The Journal of Neuroscience*, *32*(34), 11507-11510. doi.org/10.1523/JNEUROSCI.1949-12.2012

Solomon, C., Harvey, B., Kahn, K., Lieberman, H., Miller, M., Minsky, M., Papert, A., Silverman, B., & Shepard, T. (2020). History of Logo. *Proceedings of the ACM on Programming Languages*, *4*(HOPL). doi.org/10.1145/3386329

Stefanakis, E. H. (2002). *Multiple Intelligences and Portfolios: A Window into the Learner's Mind*. Heinemann.

Urrea, C., & Bender, W. (2012). Making Learning Visible. *Mind, Brain, and Education*, *6*(4), 227-241.

Notes

1 With both Music and Free/Libre Software, there is complete transparency—nothing is hidden from the student, giving them the opportunity to both debug and remix.

2 It is worth noting that Wally Feurzeig—one of the co-inventors of Logo—was both a mathematician and a pianist (Solomon et al. 2020).

3 Our intention is not to demean the demands of serious listening, but popular culture so often invites people to be passive consumers of music.

4 Music Blocks is available under the GNU Affero General Public License (AGPL) v3.0, a free, copyleft license.

5 Turtle Blocks is a fork of Brian Silverman's Turtle Art.

6 Other languages such as MAX, Scratch, or SuperCollider require the user to specify quantas of time from seconds and create their own functions in order to express time in this manner—perhaps this is why musicians do not universally find those languages useful for teaching concepts from music.

7 Music Blocks also allows the student to explore concepts such as mode, key, timbre, and temperament (See github.com/sugarlabs/musicblocks/blob/master/guide/README.md).

8 See github.com/sugarlabs/musicblocks/blob/master/Debugging.md

9 Some Music Blocks lesson plans are available at mapflc.com/lesson-plans/