TEACHING A COMPUTER TO SING: INTEGRATING COMPUTING AND MUSIC IN A MIDDLE SCHOOL, AFTER-SCHOOL PROGRAM

Jesse M. Heines
Dept. of Computer Science
Univ. of Massachusetts Lowell
Jesse_Heines@uml.edu

Daniel A. Walzer
Dept. of Music
Univ. of Massachusetts Lowell
Daniel_Walzer@uml.edu

ABSTRACT

This paper reports on an after-school program that introduced middle school students to computing through music. The program ran for two years, from October 2015 through April 2017. It involved singing, encoding music with ABC notation, and programming music with Pencil Code. We describe the program's goals and the activities students pursued, as well as suggestions for improvement. While rigorous evaluation of such a program is difficult, we present survey and focus group results that show that students' attitudes toward the program were positive and that they did learn some programming.

FROM SINGING TO PROGRAMMING

It probably goes without saying that most educators, regardless of field, think it's important for students to be computer literate and even to learn at least a little about how to program. But how do we "hook 'em," especially in an after-school program that doesn't even begin until after they've been in school for seven hours and are ready to just play and hang out with their friends?

Our approach has been to work with a dynamic music teacher whom the children adore, have her teach them songs with multiple parts and modest complexity, and then have the children encode those songs in ABC notation [2, 27] and program them in Pencil Code [16]. (In the first year we also used Audacity [1] and Scratch [20], but we switched to ABC notation and Pencil Code in our second year.) The students "performed" their creations for each other using a computer projector and the room's sound system, thereby enjoying an outlet for their work and learning from what others had produced.

While this approach sounds straightforward, it wasn't easy to implement. Despite their love of listening to—and singing along with—popular music with their friends, the children were initially shy about singing in a class setting. The music teacher worked hard to get them to overcome their reluctance. The team struggled to engage them in the computer part of the program. Due to our lack of experience with this age group, it took some time to "get our footing." However, as discussed in this paper, persistence paid off and the children indeed made progress in both singing and programming.

ENABLING RESEARCH

Our research explores the intersection of computing and music at the middle school level. The basis for this research is our prior work in Performamatics [6, 14, 15]. The first of our prior projects explored interdisciplinary connections between computer science and art, music, and theater [5, 7, 8, 9, 13, 18]. It then focused on computing+music [8, 19], which for us turned out to be the collaboration that had the most "traction." The popularity of the university courses we built and the positive research results we observed enticed us to see if we could achieve similar gains with younger students.

Teaching a Computer to Sing builds on the premise outlined by Magerko et al. [12] that creative coding can enhance musicianship by helping students identify a song's structure. It also builds on the concept of "coding through play" described by Stinson [23], who taught Scratch coding by having children follow robots through a series of imaginative storytelling activities. The key here is that children saw immediate, tangible results as the robots responded to various commands. Programming music also yielded immediate results, which children learned to identify as "right" or "wrong" by what they heard.

Ho [10] conducted a five-year study to evaluate the ability of information technology to inspire learner-centered music creation practices in young students. Both students and teachers remarked that the use of MIDI improved pitch and rhythmic accuracy when added to traditional choral rehearsal techniques and pedagogy. Our work builds on this finding by having students work with the same music in multiple modes.

Studer [24] explored the use of computers in choral rehearsals and noted that music notation programs are highly useful for isolating each part. We took an analogous approach: having children write programs that "sang" multiple parts to help them learn each one. Studer also found that music notation programs enhance STEM learning along with musical competency. Our project went one step further: writing computer programs to play music using standard computing constructs.

PROGRAM LOGISTICS

Teaching a Computer to Sing is an after-school program. It ran for 2½ hours on Tuesdays and Thursdays from October 2015 through April 2017. Students spent the first half of the session singing songs and then, after a break, spent the second coding those songs using ABC notation and Pencil Code. Sometimes we used the last 10-15 minutes of the computing session for children to show their work to the entire group. At other times, when students were fully engaged in the day's activities, we felt it best to let them continue what they were doing rather than interrupt them to share their creations.

Software Choices

We began the program in 2015 using Scratch, but the middle school students had a lot of trouble converting music notation to MIDI numbers [28]. First, converting musical scores to MIDI involved two steps: figuring out a note's alphabetic name (such as C), and then converting that note to its MIDI value (such as 60). Second, the resulting code bore little resemblance to the musical notation. For example, it's pretty hard to know that the code in Figure 1(a) plays *Frère Jacques* even if one compares it to the sheet music in Figure 1(b). The Pencil Code version in Figure 1(c) that uses ABC notation still requires a bit of interpretation, but it is clearly easier to explain to students than the Scratch version.

To help students make the transition from sheet music to ABC notation, we used EasyABC [21]. This free program allows one to enter ABC notation in one window and see the resultant score in another window. Students can thus compare the resultant score with the original score that they have on paper to ensure that they are the same. If the two scores are not identical, the ABC notation is easily edited to correct any problems.

Once the ABC notation is entered into EasyABC, it is easy to copy and paste that into Pencil Code. The final step is to enclose the Pencil Code ABC notation in quotes and add code to pass it to a function that plays it. "Pure" Pencil Code using the built-in "play" function is shown in Figure 1(c), but a little additional code displayed a dynamic keyboard

that showed which key was being played as each note sounded. We wrote a custom function "sing" that allowed playing up to four parts simultaneously by specifying which phrase to play on which piano.

The code in Figure 2 (shown in text rather than block mode) calls our custom "sing" function to play *Frère Jacques* as a round (in the key of C). Note that line 10 rests part 2 for 8 beats before "singing" that part, thus creating the round. The graphic in Figure 2 shows the keyboards as they appear during playback.

In the second year we introduced Soundtrap [22]. This online tool allowed students to record their own songs and sounds to create their own mashups. This is a different type of coding than done with the other tools, but it proved interesting to students and allowed them to extend their work ABC notation and Pencil Code. That is, Soundtrap allowed them to capture what they had done in the other programs and remix it along with pre-recorded sounds and effects as well as their own voices to create new compositions.

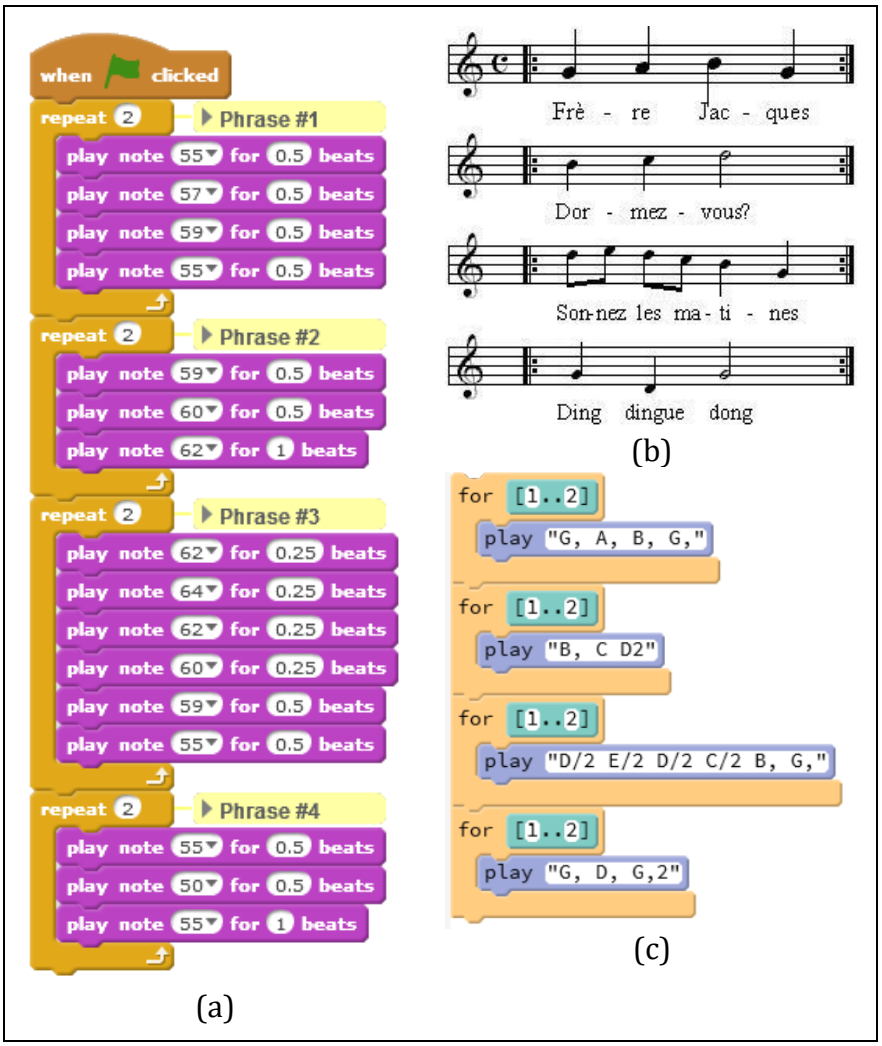

Figure 1. Frère Jacques in (a) Scratch, (b) standard music notation, and (c) Pencil Code using ABC notation.

Song Choices

Songs were mostly chosen by the music teacher. We began with popular songs such Rachel Platten's *Fight Song* [17], Taylor Swift's *Shake It Off* [25], and Shawn Mendes's *Stitches* [4]. We had an arranger create simple harmony parts for these songs, but these did not prove popular because students had trouble with the complexity of the songs' rhythms.

The music teacher suggested that we switch to "partner songs," which were sets of three simple, complementary melodies meant to be sung together. One such set consisted of *One Bottle of Pop*, *Don't Throw Your Trash in My Backyard*, and *Fish and Chips and Vinegar* [youtu.be/u-TdsmPHjo0]. It was much easier for the students to sing and code these songs in multiple parts.

Student Assistants

One of the program's key features was the relatively large number of university student assistants we employed. We had originally budgeted for two, but it quickly became apparent that that was not enough. We really needed one university student for each pair of middle school students, so we increased the number of assistants to six.

To establish rapport, the university students (and professors) sang with the middle schoolers as well as assisted them both with reading music (which was necessary to translate scores into ABC notation) and actual coding. About half the assistants were music majors, while the other half were computer science majors.

In the first year we had only one female assistant: Nicole. With 12 female and two male children in the program, we felt that it was important to recruit more, especially since Nicole provided us with invaluable insights about the complex issues that middle school girls deal with and thus helped us weather a number of storms. In the second year we had three female assistants and two males.

```
1  for [1..2]   # part 1, phrase 1
2    sing 1, "C D E C"
3  for [1..2]   # part 1, phrase 2
4    sing 1, "E F G2"
5  for [1..2]   # part 1, phrase 3
6    sing 1, "G/2 A/2 G/2 F/2 E C"
7  for [1..2]   # part 1, phrase 4
8    sing 1, "C G, C2"
9
10 sing 2,"Z8" # rest for 8 beats
11 for [1..2]   # part 2, phrase 1
12   sing 2,"C D E C"
13 for [1..2]   # part 2, phrase 2
14   sing 2,"E F G2"
15 for [1..2]   # part 2, phrase 3
16   sing 2, "G/2 A/2 G/2 F/2 E C"
17 for [1..2]   # part 2, phrase 4
18   sing 2, "C G, C2"
```

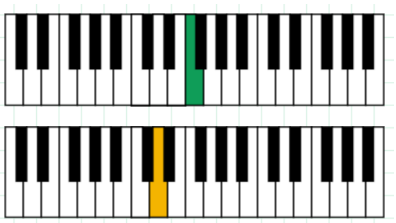

Figure 2. Playing Frère Jacques as a round.

As with any program that uses student assistants, some worked out well and some did not. Most were good at helping to keep the children on task and getting them past hurdles such as computer freeze-ups and simple programming issues. Some made excellent suggestions during our activity planning sessions, and some provided insightful observations when we reviewed each day's experiences.

Some assistants even established strong personal relationships with the middle schoolers and functioned as role models, which contributed significantly to the clubhouse atmosphere we were trying to maintain. When one assistant was absent, the children were always disappointed and asked if they would be there the next time we met.

Additional Incentives

During the second year we actively sought ways to motivate the children. We produced a CD of their work that proved to be a very popular and motivating activity. Every student contributed at least one track and many contributed multiple tracks. We made about 80 copies to fulfill all of the students' requests, and they gave them to their friends and families for the holidays.

Additional Resources

Throughout the program we created handouts with titles such as *Getting Started*

with Pencil Code, *Understanding Note and Rest Values*, and *Using the TACTS Pencil Code Functions*. These handouts, as well as links to demonstration programs and other resources, are available at jheines.github.io/tacts/Workshops, a website we created to support teacher workshops that we taught at the 2017 CCSCNE and CSTA conferences.

QUESTIONS AND FINDINGS

Our project investigated two instructional questions and two supporting questions. We attempted to answer these questions both by documenting our own informal observations and by employing surveys and focus groups conducted by the evaluation team. The former obviously suffer somewhat from observer bias, while the latter suffer somewhat from small sample size. Nevertheless, we feel that we can draw a number of conclusions from our experiences that can be reasonably supported by credible anecdotal evidence if not by hard numbers and statistical significance.

Instructional Questions

(1) Can middle schoolers follow the connections from singing to digitized sound to computer notation and back to music to help them learn to program using songs they like to sing?

Our answer to this question is an unqualified "yes." We would go even further to say "yes, and sometimes with enthusiasm." Of course, not every student was "into" the coding part of the program, and some days it seemed like none wanted to do any coding at all. (As every parent knows, such is life when working with children.) But on other days a good number of students exhibited real excitement about their ability to "teach a computer to sing" and eagerly lined up for "show and tell" at the end of the day to demonstrate their accomplishments to others.

We began the year having students code music as a series of Pencil Code play blocks. As the year progressed, we introduced progressively more advanced computing concepts. The first of these was simple loops (like those in Figure 2(c)), which allowed students to replay musical phrases that repeated in succession. Next, we introduced loops with control variables, such as `for k in [1..3]`. This allowed students to make the connection between songs with larger repetitive structures that used first and second endings and conditional `if` statements in code. We then introduced general variables, which allowed us to store and reuse musical phrases coded as ABC notation strings. A couple of students even got as far as indexed variables (one-dimensional arrays), which we used as two parallel structures to pair each note with its lyric.

Only one student in our program was a 7th grader, while all others were 5th and 6th graders. Only one had prior experience with computer programming, and about 25% did not have computers at home. Given these demographics, the fact that the students had already been in school for more than seven hours by the time they began programming with us, and our own inexperience in teaching middle school students, we feel that the programming concepts we were able to introduce represent a reasonable level of learning.

(2) Conversely, can programming their individual parts help students learn to sing in three- and four-part harmony?

During initial discussions, the music teacher told the professors that she thought it would be difficult for the children to sing in more than two parts because they had never done it before and had no familiarity with it. By the end of the year, however, they were

successfully singing the "partner songs" in three parts. They even cheered when they all finished at the same time! We can't give all the credit for that progress to the work in programming those songs, but the music teacher felt that at least some credit was due there.

One of the programming techniques that seemed to help students learn multipart songs was the use of variables to store and reuse musical phrases coded as ABC notation strings. This helped students see song structures, notice where phrases repeated, and understand how the melody lines went together. Again, we do not want to overstate this result, as it was impossible to measure objectively. However, we have learned to trust the music teacher's instincts, as she is clearly highly attuned to the students' capabilities.

Supporting Questions

(3) What resources, models, and tools are necessary to integrate STEM into a middle school, after-school choral program?

The school resources were severely limited. The computer network and Internet access were so severely tied down that Windows systems could not access the network at all and sites such as YouTube were not accessible without teacher credentials. Luckily, access to all the music sites mentioned earlier was available.

We were also unable to install software on the school systems. No one in the school had authority to do so, either. We had to make a request of the central school district office, and that took weeks to be fulfilled. To resolve this issue, we were fortunate to be able to buy systems specifically for the project's exclusive use.

Our model for the after-school program was initially what we were used to: a laboratory class. This did not prove viable, as the children were simply unable to pay attention to instructions for more than a minute or two in the after-school environment. We therefore transitioned toward a clubhouse model, where students worked one-on-one or two-on-one with a professor, university student assistant, or another middle school student.

As noted earlier, we prepared handouts with instructions and illustrations so that the children could work on their own rather than listen to us explain how to accomplish the day's goals. We also hired three times the number of university student assistants than we had originally planned, as it became evident that they were needed in the clubhouse model.

The tools we used have been discussed previously, but it is important to reemphasize that they changed throughout the program. The switch from Scratch to Pencil Code was the biggest unanticipated change at the beginning of the first year, and the discovery of EasyABC for writing ABC notation proved to be a godsend.

(4) Can the involvement of older students and teachers who match the students' racial and/or cultural backgrounds have a positive effect on the "people like me don't (or can't) do that" belief that some researchers claim is a factor in underrepresented groups' disproportionally small participation in STEM?

One of the issues that concerned us was our ability to "connect" with the middle school students. Almost all of the children had very different racial profiles from our own, and, as noted earlier, the vast majority (86%) were female. Numerous authors such as Kohl [11], Delpit [3], and Tatum [26] have written about issues related to race in the classroom, and we feared that at least some of those issues extended to gender, as well.

Looking back, it appears that we need not have been so concerned, as racial and cultural attributes did not appear to be major stumbling blocks. The children did build

relationships faster with some university students than with others or with the professors, but with time all facilitators were able to build relationships with all of the students.

Peer pressure seemed to play a larger role in relationship-building. In follow-up interviews conducted by our evaluation team, one professor observed that he had no trouble working with any child one-on-one. As soon as two or three of the students got together, however, they seemed to shut him out. He said he felt that the children—especially the girls—appeared reluctant to admit to their peers that they were comfortable talking to a teacher, especially an older white male.

The bottom line is that while we can only report observational and anecdotal evidence, in our case we feel that the premise of this question does not appear to be supported. That said, we acknowledge that our project had a small sample size and that our results are dependent on the many specific personalities involved. Thus, it is difficult—if not impossible—to generalize these results.

Lessons Learned

Based on these experiences, the main lessons we learned are as follows.

- Working with children after they've been in school for seven or more hours is hard. There are times when one has to just let them play. In addition, one must understand that some days they just won't want to code, and that pushing them to do so is futile. There were even days when the beloved and experienced music teacher found it difficult to get them to sing. "Go with the flow."
- Knowing how the children perceive a program such as this is also hard. Attitudes often cannot be seen, and one must be very careful not to make assumptions about observed behaviors.
- When meeting only once or twice a week, there is a strong need for concrete, over-arching goals to tie sessions together.
- University student assistants must be vetted carefully. We had no major problems with any assistants' interactions with the children, but it was clear that some were far better than others at helping the children learn and remain on task.
- There is simply no substitute for partnering with an experienced teacher who has a strong rapport with the children and understands where they're coming from. On several occasions, our music teacher partner pointed out where some of our assessments and impressions of how the program was going were wrong. For example, we thought that one student who didn't seem to engage with the program simply didn't want to be in it. Our teacher partner pointed out that all of this student's friends had dropped out of the program for one reason or another, and the fact that she was still with us was a strong indicator of her desire to be there.
- Despite all the time and patience it takes to get children to focus on learning in an after-school program and the inevitable ups and downs of such an endeavor, there are numerous, priceless, unforgettable "ah-ha" moments that make the effort worthwhile.

FORMAL EVALUATION

Our project was supported by an evaluation team centered in the UMass Lowell Center for Program Evaluation. The team observed after-school sessions and administered surveys and conducted focus groups with both students and faculty. We present just a few

of their more interesting findings here in descriptive terms, because all but one of the statistical measures were not significant due to small sample sizes.

Student Surveys and Focus Groups

Student surveys and focus groups revealed two major outcomes, although we stress again that these are not statistically significant.

- Measures of students' attitudes toward music and their perceptions of their own music-related abilities both increased slightly from pre- to post-program assessments.
- Measures of students' attitudes toward computer programming remained the same, but their perceptions of their own computer-related abilities increased slightly from pre- to post-program assessments.

To demonstrate just how tricky it is to conduct formal evaluation on this type of program, consider the following student responses to open-ended questions. The reasons cited most frequently for *liking* computer programming were "making games, music, and websites," "coding," and "I don't know." Those cited most frequently for *not* liking computer programming were "boring," "hard," "so much to do," and "I don't know." We find it interesting that "coding" shows up in the positive list, while "hard" and "so much to do" show up in the negative list. These seem contradictory, and of course the "I don't know" response is not helpful, particularly because it appears in both lists.

A more positive outcome that the evaluation team reported was that 67% of students responded "yes" when asked if computer science and music were related in any way. We interpret this as at least an indication that most students were able to achieve one of our primary goals: to have them follow the connections from singing to digitized sound to computer notation and back to music.

The focus group discussion clearly revealed that students preferred working with EasyABC and Pencil Code to Audacity and Scratch. They like the visual aspects of EasyABC and the versatility of Pencil Code, that is, its ability to support various types of projects.

It is also telling that the students' two most prominent suggestions across both years for improving the program were related to having more snacks and more fun. We feel that such comments are typical for an after-school program, and we tried to accommodate them by allowing snacks during the program and giving students more free time after they completed coding activities.

Facilitator Surveys and Focus Groups

As discussed above, a number of changes were made in Year 2 of the program. Facilitator surveys and focus groups attempted to assess the effect of these differences, at least from the facilitators' perspective.

Using a five-point Likert scale, facilitators were asked whether students were able to follow the connections from singing to digitized sound to computer notation and back to music. As shown in Table 1, their responses were far more positive in Year 2 than in Year 1, and even with the small sample size the difference was statistically significant ($p < .05$).

Facilitators were also asked if they felt that (a) using songs that students like to sing helped them learn to program, (b) students' efficacy for programming had improved throughout the program, and (c) working with adults who match the students' racial or cultural backgrounds had a positive effect on the students. The Likert scale results for these questions increased from Year 1 to Year 2, but not enough to be statistically significant.

Table 1. Facilitators' Perspectives on Program Objectives

Objective	Year	N	Mean	Std.Err.	Sig.
Follow connections both ways	1	4	2.00	0.577	$p < .05$
	2	8	3.88	0.350	
Learn to program using songs	1	4	3.50	0.289	no
	2	8	4.13	0.295	
Improve efficacy for programming	1	4	3.50	0.289	no
	2	7	3.71	0.184	
Racial/cultural matching positive	1	4	2.75	0.479	no
	2	7	3.71	0.184	

The differences between facilitators' responses to open-ended questions in Year 1 and Year 2 are telling with regard to the program's development.

- In Year 1, responses focused on what they needed to teach. In Year 2, they focused on the need for more structured plans.
- In Year 1, their favorite songs to work on were the partner songs. In Year 2, their favorites were those that supported loops.

A major theme in the focus group discussion was relationships, that is, establishing a rapport with the children so that they would be receptive to instruction. This improved greatly in Year 2 with more careful vetting of university student assistants and the professors' acceptance of the need to work one-on-one or one-on-two. In addition, we all had to learn to allow for the ebb and flow of the children's learning and attention span in an after-school program.

RECOMMENDATIONS FOR FUTURE PROGRAMS

- Structure the program to include ample time for facilitator planning and preparation outside of the time spent with the children.

Comment: This was difficult for us given our office locations of different campuses and the differences in our teaching schedules. We also would have benefitted from more outside the classroom time with the university student facilitators.

- Have many structured activities available to keep the children engaged.

Comment: In most sessions, we had structured activities for the main theme of the day, but it would have been good to have had several backup structured activities, as well.

- Ensure that all facilitators circulate throughout the room and work with the children one-on-one or in pairs.

Comment: Some of the university student assistants thought that they should wait until a child asked for assistance, which they seldom did. We talked with the student assistants about this, but some were still reluctant to simply sit down next to a child and ask him or her to show them what they were doing. On the other hand, the children were very receptive to help and advice when it was offered.

- Present short narratives or movie clips and discuss famous minorities in STEM.

Comment: The children were very receptive to clips that we did show them, and it would have been good to do that more often.

- Provide more opportunities for students to share what they have done in the program.

Comment: The children were all strongly engaged with the creation of the holiday CD. In retrospect, we could have created a second CD or had some other culminating joint project to further motivate their participation.

ACKNOWLEDGMENTS

This work is supported by Award No. 1515767 from the National Science Foundation Division of Research on Learning. Any opinions, findings, conclusions, or recommendations expressed in this proposal are solely those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Audacity Open Source Development Team (2011). *Audacity: The Free, Cross-Platform Audio Editor and Recorder*. audacity.sourceforge.net *accessed* Oct. 25, 2014.
[2] Chambers, J. (2016). *An ABC Primer*. trillian.mit.edu/~jc/music/abc/doc/ABCprimer.html *accessed* Aug. 16, 2016.
[3] Delpit, L. (2006). *Other People's Children: Cultural Conflict in the Classroom*. New York: The New Press.
[4] Geiger, T., Parker, D., & Kyriakides, D. (2015). *Stitches*: Hal Leonard Music Publishing.
[5] Greher, G.R., & Heines, J.M. (2009). *Sound Thinking: Conceptualizing the Art and Science of Digital Audio for an Interdisciplinary General Education Course*. Assoc. for Technology in Music Instruction (ATMI) 2009 Conf. Portland, OR.
[6] Greher, G.R., & Heines, J.M. (2014). *Computational Thinking in Sound*. New York: Oxford University Press.
[7] Heines, J.M., Jeffers, J., & Kuhn, S. (2008). *Performamatics: Experiences with Connecting a Computer Science Course to a Design Arts Course.* Int'l. Jrnl. of Learning **15**(2):9-16.
[8] Heines, J.M., Greher, G.R., & Kuhn, S. (2009). *Music Performamatics: Interdisciplinary Interaction*. Proc. of the 40th ACM Tech. Symposium on CS Education, pp. 478-482. Chattanooga, TN: ACM.
[9] Heines, J.M., Greher, G.R., Ruthmann, S.A., & Reilly, B. (2011). *Two Approaches to Interdisciplinary Computing+ Music Courses.* IEEE Computer **44**(12):25-32.
[10] Ho, W.-C. (2004). *Use of information technology and music learning in the search for quality education.* British Jrnl. of Educational Technology **35**(1):57-67.
[11] Kohl, H. (1994). *"I Won't Learn from You" and Other Thoughts on Creative Maladjustment*. NY: The New Press.
[12] Magerko, B., Freeman, J., McKlin, T., McCoid, S., Jenkins, T., & Livingston, E. (2013). *Tackling engagement in computing with computational music remixing*. Proc. of the 44th ACM Tech. Symposium on CS Education, pp. 657-662.
[13] Martin, F., Greher, G.R., Heines, J.M., Jeffers, J., Kim, H.-J., Kuhn, S., Roehr, K., Selleck, N., Silka, L., & Yanco, H. (2009). *Joining Computing and the Arts at a Mid-Size University.* Jrnl. of Computing Sciences in Colleges **24**(6):87-94.
[14] National Science Foundation (2007). *NSF Award #0722161 - CPATH CB: Performamatics: Connecting Computer Science to the Performing, Fine, and Design Arts*. CNS: Division of Computer and Network Systems, www.nsf.gov/awardsearch/showAward?AWD_ID=0722161 *accessed* Nov. 4, 2013.
[15] National Science Foundation (2011). *NSF Award #1118435 - Computational Thinking through Computing and Music*. DUE: Division of Undergrad. Ed., www.nsf.gov/awardsearch/showAward?AWD_ID=1118435 *accessed* Nov. 4, 2013.
[16] Pencil Code (2016). *Dream it. Code it*. pencilcode.net *accessed* 8/16/2016.
[17] Platten, R., & Bassett, D. (2015). *Fight Song*: Columbia Records.
[18] Ruthmann, S.A., & Heines, J.M. (2009). *Designing Music Composing Software with and for Middle School Students: A Collaborative Project among Senior Computer Science and Music Education Majors*. Association for Technology in Music Instruction (ATMI) 2009 Conference. Portland, OR.
[19] Ruthmann, S.A., Greher, G.R., & Heines, J.M. (2012). *Real World Projects for Developing Musical and Computational Thinking*. 30th Int'l Society for Music Education (ISME) World Conf. on Music Ed. Thessaloniki, Greece.
[20] Scratch (2016). *Create stories, games, and animations; Share with others around the world*. scratch.mit.edu *accessed* Aug. 16, 2016.
[21] Shlien, S. (2017). *EasyABC*. easyabc.sourceforge.net *accessed* Aug. 8, 2017.
[22] Soundtrap (2017). *Soundtrap Education*. www.soundtrap.com/edu/ *accessed* Aug. 13, 2017.
[23] Stinson, L. (2013). *Google and apple alums invent adorable robots that teach kids to code*. *Wired*.
[24] Studer, K. (2005). *Maximum Technology in the Music Classroom: Minimum Requirements.* Teaching Music **13**(3):44-47.
[25] Swift, T., Martin, M., & Shellback (2014). *Shake It Off*: Big Machine Records.
[26] Tatun, B. (1997). *"Why Are All the Black Kids Sitting Together in the Cafeteria" and Other Conversations About Race*. New York: Basic Books.
[27] Walshaw, C. (2017). *ABC Notation*. abcnotation.com *accessed* Aug. 8, 2017.
[28] Wolfe, J. (2017). *Note names, MIDI numbers and frequencies*. newt.phys.unsw.edu.au/jw/notes.html *accessed* Aug. 8, 2017.